Date : 07-Feb-24

CHART.JS

Version: 1.0.3

1. **OVERVIEW**

[Chart.js](https://www.chartjs.org/) is an open-source JavaScript library that can help you to draw several types of charts. The Chart.js component brings this functionality to VoltMX Iris. You can use this component in scenarios where you want to display data in your app by using charts. This component supports Mobile, Tablet, and Web application channels.

In this component, only bare essential functionalities of the Chart.js library have been exposed by using Component Properties. However, you can access the entire Chart.js library by using the [createChart()](https://docs.kony.com/marketplace/Chartjs/Content/Chart.js/Chart.js_References.htm" \l "createChart) API.

**Note**: When you use createChart() API, the values provided through component properties are invalid.

You can use this component to display one chart or a bunch of charts as per your requirement. The component generates as many charts as you specify by using [the DataSet](https://docs.kony.com/marketplace/Chartjs/Content/Chart.js/Chart.js_References.htm#Data) property or the [createChart()](https://docs.kony.com/marketplace/Chartjs/Content/Chart.js/Chart.js_References.htm" \l "createChart) API.

1. **Usecase:**

I. Draw several types of charts.

## B. Percentage of re-use:

90% (Data and skins can be customizable)

# GETTING STARTED

**A.** [**Prerequisites**](javascript:void(0);)



Before you start using the Chart.js component, ensure you have the following:

* [HCL Foundry](https://manage.hclvoltmx.com/)
* Volt MX Iris

## B. Platforms Supported

### i. Mobile

#### 1. iOS

#### 2. Android

### ii. Tablets

### iii. PWA

## Importing the Component

## You can import the Forge components only into the apps that are of the Reference Architecture type.

## To import the Chart.js component, do the following:

## Open your app project in Volt MX Iris.

1. In the Project Explorer, click the **Templates** tab.

Graphical user interface, text, application

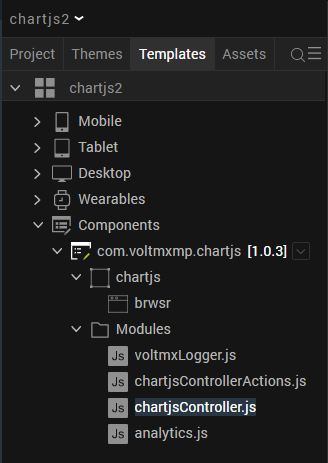
Description automatically generated

3.Right-click **Components**, and then select **Import Component**. The **Import Component** dialog box appears.

Graphical user interface, text, application, Teams

Description automatically generated

4.Click **Browse** to navigate to the location of the component, select the component, and then click **Import**. The component and its associated widgets and modules are added to your project.



Once you have imported a component to your project, you can easily add the component to a form. For more information, refer [Add a Component to a Form](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/C_UsingComponents.html#add-a-component-to-a-form).

# 3.REFERENCES

## A. Dynamic Usage

You can also add an Chart.js component dynamically. To do so:

1. In the **Project Explorer**, on the **Projects** tab, click **Controllers** section to access the respective **Form Controller**. Create a method and implement the code snippet similar to the sample code mentioned below.

In the code snippet, you can edit the properties of the component as per your requirement. For more information, see Setting Properties.

createComponent: function() {

/\* Creating the component's object \*/

var chartjs = new com.voltmxmp.chartjs(

{

"clipBounds": true,

"height": "100%",

"id": "chartjs",

"isVisible": true,

"left": "0dp",

"top": "0dp",

"width": "100%",

"zIndex": 1

}, {}, {});

/\* Setting the component's properties \*/

chartjs.gridChartDataSet = {

"data":

[{

"chartType": "Line",

"title": "Spending Trend (USD)",

"labels": "[January,February,March,April,May,June,July]",

"data": "[3000,2900,3300,3400,2800,2780,2550]",

"color": "#20639B",

"bgColor": "#ffffff"

}, {

"chartType": "Doughnut",

"title": "Spend by Category (USD)",

"labels": "[Travel,Food & Dining,Shopping,Bills, Health & Fitness]",

"data": "[250,190,120,100,80]",

"color": "[#ED553B, #F6D55C, #3CAEA3, #20639B, #172F5F]",

"bgColor": "#ffffff"

}, {

"chartType": "Vertical Bar",

"title": "Net Income over time (USD)",

"labels": "[January,February,March,April,May,June,July]",

"data": "[4500,4000,3500,3600,3400,3350,3800]",

"color": "#3CAEA3",

"bgColor": "#ffffff"

}, {

"chartType": "Horizontal Bar",

"title": "Budget (USD)",

"labels": "[Travel,Food & Dining,Shopping,Bills, Health & Fitness]",

"data": "[199,250,220,255,230]",

"color": "#3CAEA3",

"bgColor": ""

}, {

"chartType": "Radar",

"title": "Budget vs Spending (USD)",

"labels": "[Travel,Food & Dining,Shopping,Bills, Health & Fitness]",

"data": "Budget:[199,250,220,255,230],Spending: [250,190,120,100,80]",

"color": "Budget:[#3CAEA3],Spending:[#ED553B]",

"bgColor": ""

}, {

"chartType": "PolarArea",

"title": "Week over Week Spending (USD)",

"labels": "[Week 1, Week 2,Week 3,Week 4]",

"data": "[190,170,140,115]",

"color": "[#F6D55C, #3CAEA3, #20639B,#172F5F]",

"bgColor": ""

}, {

"chartType": "Pie",

"title": "Spend by Merchants (USD)",

"labels": "[Starbucks,iTunes,Uber,Amazon,Walmart]",

"data": "[45,60,80,110,145]",

"color": "[#ED553B, #F6D55C, #3CAEA3, #20639B, #172F5F]",

"bgColor": ""

}]

};

chartjs.renderType = true;

chartjs.noOfColumns = 2;

chartjs.enableLegend = true;

chartjs.legendPosition = "top";

chartjs.propertySyntax = "<property\_syntax>";

chartjs.titlePosition = "top";

chartjs.duration = 1000;

chartjs.easing = "easeOutQuart";

chartjs.fontColor = "#232425";

chartjs.fontSize = "12";

chartjs.fontFamily = "Arial";

chartjs.fontStyle = "normal";

/\*Adding the component to a Form\*/

//this.view.chartjs.renderType=true;

this.view.add(chartjs);

/\*Calling create chart\*/

var dataObject = {

"1": {

type: 'bar',

data: {

labels: ['Red', 'Blue', 'Yellow', 'Green', 'Purple', 'Orange'],

datasets: [{

label: '# of Votes',

data: [12, 19, 3, 5, 2, 3],

backgroundColor: ['#ff6384', '#36a2eb', '#cc65fe', '#ffce56', '#36a2eb', '#36a2eb'],

borderColor: ['rgba(255, 99, 132, 1)', 'rgba(54, 162, 235, 1)', 'rgba(255, 206, 86, 1)', 'rgba(75, 192, 192, 1)', 'rgba(153, 102, 255, 1)', 'rgba(255, 159, 64, 1)'],

borderWidth: 1

}, {

label: '# of Votes',

data: [2, 5, 12, 8, 2, 5],

backgroundColor: ['#ff6384', '#36a2eb', '#cc65fe', '#ffce56', '#36a2eb', '#36a2eb'],

borderColor: ['rgba(255, 99, 132, 1)', 'rgba(54, 162, 235, 1)', 'rgba(255, 206, 86, 1)', 'rgba(75, 192, 192, 1)', 'rgba(153, 102, 255, 1)', 'rgba(255, 159, 64, 1)'],

borderWidth: 1

}]

},

options: {

scales: {

yAxes: [{

ticks: {

beginAtZero: true

}

}]

},

legend: {

display: false,

labels: {

fontColor: 'rgb(255, 99, 132)'

}

}

}

}

};

// this.view.btnchartjs.onClick=this.view.chartjs.createChart(2,dataObject, {});

//this.view.chartjs.render();

this.view.btnchartjs.onClick=

this.view.chartjs.createChart(2,dataObject,{});

}

1. **Save** the file.

**B. Properties**

The properties provided on the **Component** tab allow you to customize the UI elements in the **Chart.js** component. You can set the properties directly on the **Component** tab or by writing a JavaScript.

1**. Render Automatic**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the type of rendering (automatic or forced). If you use a true component, the chart is rendered automatically. But in other cases, the render API must be called to display the charts. |
| Syntax: | renderType |
| **Type:** | boolean |
| **Read/Write:** | Write |
| **Example:** | this.view.<component\_id>.renderType = true; |

**2. Data set**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the input data set such as color, labels, and chart type. The data is then displayed in a chart format. Every row in the datagrid generates a chart. |
| **Syntax:** | gridChartDataSet |
| **Type:** | Data Grid |
| **Read/Write:** | Read + Write |
| **Example:** | |  | | --- | | this.view. < component\_id > .gridChartDataSet = {  "data":  [{  "chartType": "Line",  "title": "Spending Trend (USD)",  "labels": "[January,February,March,April,May,June,July]",  "data": "[3000,2900,3300,3400,2800,2780,2550]",  "color": "#20639B",  "bgColor": "#ffffff"  }, {  "chartType": "Doughnut",  "title": "Spend by Category (USD)",  "labels": "[Travel,Food & Dining,Shopping,Bills, Health & Fitness]",  "data": "[250,190,120,100,80]",  "color": "[#ED553B, #F6D55C, #3CAEA3, #20639B, #172F5F]",  "bgColor": "#ffffff"  }, {  "chartType": "Vertical Bar",  "title": "Net Income over time (USD)",  "labels": "[January,February,March,April,May,June,July]",  "data": "[4500,4000,3500,3600,3400,3350,3800]",  "color": "#3CAEA3",  "bgColor": "#ffffff"  }, {  "chartType": "Horizontal Bar",  "title": "Budget (USD)",  "labels": "[Travel,Food & Dining,Shopping,Bills, Health & Fitness]",  "data": "[199,250,220,255,230]",  "color": "#3CAEA3",  "bgColor": ""  }, {  "chartType": "Radar",  "title": "Budget vs Spending (USD)",  "labels": "[Travel,Food & Dining,Shopping,Bills, Health & Fitness]",  "data": "Budget:[199,250,220,255,230],Spending: [250,190,120,100,80]",  "color": "Budget:[#3CAEA3],Spending:[#ED553B]",  "bgColor": ""  }, {  "chartType": "PolarArea",  "title": "Week over Week Spending (USD)",  "labels": "[Week 1, Week 2,Week 3,Week 4]",  "data": "[190,170,140,115]",  "color": "[#F6D55C, #3CAEA3, #20639B,#172F5F]",  "bgColor": ""  }, {  "chartType": "Pie",  "title": "Spend by Merchants (USD)",  "labels": "[Starbucks,iTunes,Uber,Amazon,Walmart]",  "data": "[45,60,80,110,145]",  "color": "[#ED553B, #F6D55C, #3CAEA3, #20639B, #172F5F]",  "bgColor": ""  }] | |

1. **No of Colums**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the number of columns to be displayed in Responsive Web and Tablet. Not applicable for Mobile. In Mobile, the charts are always displayed in one column. |
| **Syntax:** | noOfColumns |
| **Type:** | String/Integer |
| **Read/Write:** | Write |
| **Example:** | this.view.<component\_id>.noOfColums = 2; |

1. **Enable Legend**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Enables the visibility of the chart legends. |
| **Syntax:** | enableLegend |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Example:** | this.view.<component\_id>.enableLegend = true; |

1. **Legend Position**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Enables you to set the position of the legends in the chart. |
| **Syntax:** | legendPosition |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.<component\_id>.legendPosition = "top"; |

1. **Enable Title**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Enables the visibility of the chart title. |
| **Syntax:** | enableTitle |
| **Type:** | Boolean |
| **Read/Write:** | Write | |

[Open](javascript:void(0);)

|  |  |
| --- | --- |
| **Example:** | this.view.<component\_id>.enableTitle = true; |

**7. Title Position**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Enables you to set the position of the title in the chart. |
| **Syntax:** | titlePosition |
| **Type:** | String |
| **Read/Write:** | write |
| **Example:** | this.view.<component\_id>.titlePosition= "top"; |

**8. Duration**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the time taken by the animation in milliseconds. If animation is not required, you must set the value of duration as 0. |
| **Syntax:** | duration |
| **Type:** | Integer |
| **Read/Write:** | write |
| **Example:** | this.view.<component\_id>.duration = 1000; |

**[Open](javascript:void(0);)9. Easing**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Sets the type of animation in the chart. |
| **Syntax:** | easing |
| **Type:** | String |
| **Example:** | this.view.<component\_id>.easing = "easeOutQuart"; |
| **Read/Write:** | write |

**[Open](javascript:void(0);)10. Font Color**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the font color of the displayed text. |
| **Syntax:** | fontcolor |
| **Type:** | String |
| **Example:** | this.view.<component\_id>.fontcolor = "#232425"; |

[Open](javascript:void(0);)

|  |  |
| --- | --- |
| **Read/Write:** | write |

**10. Font Size**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the font size of the displayed text. By default, the size is measured in pixels (px). |
| **Syntax:** | fontsize |
| **Type:** | String |
| **Example:** | this.view.<component\_id>.fontsize = "12"; |
| **Read/Write:** | write |

[Open](javascript:void(0);)

**11. Font Family**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the font family of the displayed text. |
| **Syntax:** | fontfamily |
| **Type:** | String |
| **Read/Write:** | write |
| **Example:** | this.view.<component\_id>.fontfamily = "Arial"; |

**[Open](javascript:void(0);)12. Font Style**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the font style of the displayed text.  **Note:** The fontstyle property does not apply to tooltip, footer, and title of the chart. |
| **Syntax:** | fontstyle |
| **Type:** | String |
| **Read/Write:** | write |
| **Example:** | this.view.<component\_id>.fontstyle = "normal"; |

**C. Events**

The Chart.js component invokes events when its corresponding action is performed. You can configure any logic that you want the component to perform whenever an event occurs. The events can be configured directly from the **Actions** tab or by writing a JavaScript snippet, For more information, refer [Add Actions](https://docs.kony.com/konylibrary/visualizer/visualizer_user_guide/Content/working_with_Action_Editor.htm).

**[Open](javascript:void(0);)1. errorCallback**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | The event is invoked when an error occurs. |
| **Syntax:** | errorCallback |
| **Example:** | this.view. < component\_id > .errorCallback = function(errorObject) {  alert(errorObject);  } |

1. **APIs**

You can use the following APIs for Chart.js component:

**[Open](javascript:void(0);)1. render**

|  |  |
| --- | --- |
| **Description:** | Renders or displays the charts as per the data provided through Properties. This API is called initially to render any chart. |
| **Syntax:** | render() |
| **Example:** | this.view.<component\_id>.render(); |

**[Open](javascript:void(0);)****2.** [**createChart**](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Creates and displays charts based on the data that you provide in the parameters.  The createChart API helps you to create |
| **Syntax:** | createChart(noOfColumns,dataObject,globalConfigurations) |
| **Parameters:** | * *noOfColumns* Specifies the number of columns to be displayed in Responsive Web and Tablet. For more information on the noOfColumns parameter, click [here](https://docs.kony.com/marketplace/Chartjs/Content/Chart.js/Chart.js_References.htm#No_of_Columns). * *dataObject* Specifies the data and other configuration options for every chart. For more information on configuration options for different chart types, click [here](https://www.chartjs.org/docs/latest/charts/).   dataObject : {  "<chart\_ID>" : {  "type" : "chart type",  "data" : "chart Data",  "options" : "chart options"  }  }   * *globalConfigurations* Global configurations allow you to change options globally across all the charts generated using the component, avoiding the need to specify options for each instance. These properties control styling, fonts, legends, and more. For more information on the globalConfigurations parameter, click [here](https://www.chartjs.org/docs/latest/configuration/).   {  "defaultColor" : "#234354",  "defaultFontColor" : "#000000",  "animation" : {  "duartion" : 1000,  "easing" : "easeOutQuart"  },  "events" : ["mosuemove"],  "hover" : {  "animationDuration" : 400,  "mode" : "nearest"  },  "layout" : {  "padding" : {  "top" : 0  }  },  "legend" : {  "display" : true  },  "plugins" : {  "responsive" : true  },  "title" : {  "display" : true,  "fontStyle" : "bold"  },  "tooltips" : {  "backgroundColor" : "#000000",  "caretSize" : 5  }  } |
| **Example:** | var dataObject = {  "1": {  type: 'bar',  data: {  labels: ['Red', 'Blue', 'Yellow', 'Green', 'Purple', 'Orange'],  datasets: [{  label: '# of Votes',  data: [12, 19, 3, 5, 2, 3],  backgroundColor: ['#ff6384', '#36a2eb', '#cc65fe', '#ffce56', '#36a2eb', '#36a2eb'],  borderColor: ['rgba(255, 99, 132, 1)', 'rgba(54, 162, 235, 1)', 'rgba(255, 206, 86, 1)', 'rgba(75, 192, 192, 1)', 'rgba(153, 102, 255, 1)', 'rgba(255, 159, 64, 1)'],  borderWidth: 1  }, {  label: '# of Votes',  data: [2, 5, 12, 8, 2, 5],  backgroundColor: ['#ff6384', '#36a2eb', '#cc65fe', '#ffce56', '#36a2eb', '#36a2eb'],  borderColor: ['rgba(255, 99, 132, 1)', 'rgba(54, 162, 235, 1)', 'rgba(255, 206, 86, 1)', 'rgba(75, 192, 192, 1)', 'rgba(153, 102, 255, 1)', 'rgba(255, 159, 64, 1)'],  borderWidth: 1  }]  },  options: {  scales: {  yAxes: [{  ticks: {  beginAtZero: true  }  }]  },  legend: {  display: false,  labels: {  fontColor: 'rgb(255, 99, 132)'  }  }  }  },  "2": {  type: 'line',  // The data for our dataset  data: {  labels: ['January', 'February', 'March', 'April', 'May', 'June', 'July'],  datasets: [{  label: 'My First dataset',  backgroundColor: 'rgb(255, 99, 132)',  borderColor: 'rgb(255, 99, 132)',  data: [0, 10, 5, 2, 20, 30, 45],  fill: false  }, {  label: 'My second dataset',  backgroundColor: 'rgb(99, 107, 255)',  borderColor: 'rgb(45, 196, 141)',  data: [0, 2, 5, 10, 38, 5, 25],  fill: false  }]  },  // Configuration options go here  options: {  scales: {  yAxes: [{  stacked: true  }]  },  title: {  display: true,  text: 'Custom Chart Title'  },  tooltips: {  // Disable the on-canvas tooltip  enabled: false,  custom: function(tooltipModel) {  // Tooltip Element  var tooltipEl = document.getElementById('chartjs-tooltip');  // Create element on first render  if (!tooltipEl) {  tooltipEl = document.createElement('div');  tooltipEl.id = 'chartjs-tooltip';  tooltipEl.innerHTML = '<table></table>';  document.body.appendChild(tooltipEl);  }  // Hide if no tooltip  if (tooltipModel.opacity === 0) {  tooltipEl.style.opacity = 0;  return;  }  // Set caret Position  tooltipEl.classList.remove('above', 'below', 'no-transform');  if (tooltipModel.yAlign) {  tooltipEl.classList.add(tooltipModel.yAlign);  } else {  tooltipEl.classList.add('no-transform');  }  function getBody(bodyItem) {  return bodyItem.lines;  }  // Set Text  if (tooltipModel.body) {  var titleLines = tooltipModel.title || [];  var bodyLines = tooltipModel.body.map(getBody);  var innerHtml = '<thead>';  titleLines.forEach(function(title) {  innerHtml += '<tr><th>' + title + '</th></tr>';  });  innerHtml += '</thead><tbody>';  bodyLines.forEach(function(body, i) {  var colors = tooltipModel.labelColors[i];  var style = 'background:' + colors.backgroundColor;  style += '; border-color:' + colors.borderColor;  style += '; border-width: 2px';  var span = '<span style="' + style + '"></span>';  innerHtml += '<tr><td>' + span + body + '</td></tr>';  });  innerHtml += '</tbody>';  var tableRoot = tooltipEl.querySelector('table');  tableRoot.innerHTML = innerHtml;  }  // `this` will be the overall tooltip  var position = this.\_chart.canvas.getBoundingClientRect();  // Display, position, and set styles for font  tooltipEl.style.opacity = 1;  tooltipEl.style.position = 'absolute';  tooltipEl.style.left = position.left + window.pageXOffset + tooltipModel.caretX + 'px';  tooltipEl.style.top = position.top + window.pageYOffset + tooltipModel.caretY + 'px';  tooltipEl.style.fontFamily = tooltipModel.\_bodyFontFamily;  tooltipEl.style.fontSize = tooltipModel.bodyFontSize + 'px';  tooltipEl.style.fontStyle = tooltipModel.\_bodyFontStyle;  tooltipEl.style.padding = tooltipModel.yPadding + 'px ' + tooltipModel.xPadding + 'px';  tooltipEl.style.pointerEvents = 'none';  }  }  }  },  "3": {  type: 'pie',  data: {  datasets: [{  data: [  this.randomScalingFactor(), this.randomScalingFactor(), this.randomScalingFactor(), this.randomScalingFactor(), this.randomScalingFactor(),  ],  backgroundColor: ['rgba(255, 99, 132, 1)', 'rgba(54, 162, 235, 1)', 'rgba(255, 206, 86, 1)', 'rgba(75, 192, 192, 1)', 'rgba(153, 102, 255, 1)'],  label: 'Dataset 1'  }],  labels: ['Red', 'Orange', 'Yellow', 'Green', 'Blue']  },  options: {  responsive: true  }  },  "4": {  type: 'doughnut',  data: {  datasets: [{  data: [  this.randomScalingFactor(), this.randomScalingFactor(), this.randomScalingFactor(), this.randomScalingFactor(), this.randomScalingFactor(),  ],  backgroundColor: ['rgba(255, 99, 132, 1)', 'rgba(54, 162, 235, 1)', 'rgba(255, 206, 86, 1)', 'rgba(75, 192, 192, 1)', 'rgba(153, 102, 255, 1)'],  label: 'Dataset 1'  }],  labels: ['Red', 'Orange', 'Yellow', 'Green', 'Blue']  },  options: {  responsive: true,  legend: {  position: 'top',  },  title: {  display: true,  text: 'Chart.js Doughnut Chart'  },  animation: {  animateScale: true,  animateRotate: true  }  }  }  };  /\* Render chart on a static chartjs component \*/  this.view.chart.createChart(noOfColumns, dataObject, {});  /\*Adding the component to a Form, refer to dynamic usage above \*/  this.view.add(chart);  this.view.chart.createChart(3, dataObject, {});  },  randomScalingFactor: function() {  try {  return Math.round(Math.random() \* 100);  } catch (exception) {  throw exception;  }  } |

For more information on how to create a chart by using **Chart.js**, click [here](https://www.chartjs.org/docs/latest/).

# 4. REVISION HISTORY

App version :1.0.3

1. **Known Issues**

No known issues