Date : 29-Sep-21

pie chart – partial data

version: 1.1.1

**1. OVERVIEW**

Pie Chart – Partial Data (also known as Pie Chart with a cutout) is a Volt MX component that creates a Pie chart with a cutout slice, based on the data that you provide. You can use the component in your mobile app to represent the comparison between distinct items or data in the form of pie. For examples, a company's profit percentage on annual basis.

Chart, pie chart

Description automatically generated

1. **Use Case:**

Consider a scenario that you want to develop a mobile app to provide a company's sales information. In the app, you want to build a feature to represent a company's profit percentage over a few years in the form of Pie Chart. You can use the Pie Chart component to represent the profit percentage with respect to the years in the form of Pie chart.

1. **Features:**
2. Ready-to-use Chart.
3. Useful component for project analysis.
4. Easy to plug into your app.
5. Facility to modify the UI.
6. **Percentage of re-use:**

90% (Data and skins can be customizable)

# 2. GETTING STARTED

## Prerequisites

Before you start using the Pie Chart with a cutout, ensure you have the following:

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

## Platforms Supported

### Mobile

#### iOS

#### Android

### Tablets

### PWA

## Importing the Component

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

## To import the Pie Chart with a cutout component, do the following:

## Open your app project in Volt MX Iris.

## 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.

Graphical user interface, text

Description automatically generated

# 3. REFERENCES

# A. Dynamic Usage

You can also add a **Pie Chart with a cutout**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.

/\* Creating the component s object \*/

var pieChart = new com.volmxmp.piechartwcut(

{

"clipBounds": true,

"id": "pieChart",

"height": "100%",

"width": "100%",

"top": "0%",

"left": "0%",

"skin": "CopyslFbox2",

"layoutType": voltmx.flex.FREE\_FORM,

"masterType": constants.MASTER\_TYPE\_USERWIDGET,

"isVisible": true,

"zIndex": 1

}, {}, {});

/\* Setting the component s properties \*/

pieChart.bgColor = "#FFFFFF";

pieChart.chartData =

{

"data":

[

{"colorCode": "#1b9ed9", "label": "data1", "value": 40},

{"colorCode": "#e8672b", "label": "data2", "value": 20},

{"colorCode": "#76c044", "label": "data3", "value": 10}

]

};

pieChart.enableStaticPreview = true;

pieChart.enableLegend = true;

pieChart.chartTitle = "Pie Chart";

pieChart.titleFontColor = "#000000";

pieChart.legendFontColor = "#000000";

pieChart.titleFontSize = “12”;

pieChart.legendFontSize = “8”;

pieChart.total = “120”;

/\* Adding the component to the form \*/

this.view.add(pieChart);

this.view.forceLayout();

In the code snippet, you can edit the properties of the component as per your requirement.

1. **Save** the file.

**B. Properties**

The properties provided on the **Component** tab allow you to customize the UI elements in the Pie Chart component. You can set the properties directly on the **Component** tab or dynamically through code. This section provides information on how to set the properties dynamically through code.

**Closed1. Background Color**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies background color of the chart. |
| **Syntax**: | bgColor |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.componentID.bgColor = "#ffffff"; |

**Closed2. Chart Data**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Enables a user to provide the data to generate the chart. |
| **Syntax:** | chartData |
| **Type:** | Data Grid |
| **Read/Write:** | Write |
| **Example:** | var data =  [  {"colorCode": "#1B9ED9", "label": "data1", "value": "25"},  {"colorCode": "#76C044", "label": "data2", "value": "20"}  ]; |

1. **Enable Static Preview**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies whether or not the chart should render in the data grids, with the given data. |
| **Syntax:** | enableStaticPreview |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Example:** | this.view.componentID.enableStaticPreview = true; |

1. **Enable Legends**

Closed

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Controls whether or not to enable the chart grid. |
| **Syntax:** | enableLegends |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Example:** | this.view.componentID.enableLegends = true; |
|  |  |

1. **Legend Font Color**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the font color of the Chart legend. |
| **Syntax:** | legendFontColor |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | The default value of the property is "#000000". |
| **Example:** | this.view.componentID.legendFontColor= "#000000"; |
|  |  |

1. **Legend Font Size**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the font size of the Chart legend. |
| **Syntax:** | legendFontSize |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | The default value of the property is "8".  The maximum and minimum recommended values are 12 and 8 respectively for better UI. |
| **Example:** | this.view.componentID.legendFontSize= "8"; |

1. **Chart Title**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the text to be displayed as the Chart title. |
| **Syntax:** | chartTitle |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.componentID.chartTitle = "Pie Chart"; |

1. **Title Font Size**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the font size of the Chart title. |
| **Syntax:** | titleFontSize |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.componentID.titleFontSize= "12"; |

1. **Title Font Color**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the font color of the Chart title. |
| **Syntax:** | titleFontColor |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.componentID.titleFontColor = "#000000"; |

1. **total**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the total value of the Cutout. |
| **Syntax:** | total |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | The value given must be always greater than the sum of the values passed as data to the chart. |
| **Example:** | this.view.componentID.total= "120"; |

## C. Events

-- None of the events are exposed.

## D. APIs

The following API pertains to the Pie Chart component.

The API creates a Pie Chart.

**Syntax**

createChart(data)

**Parameters**

*data:*   
JSON array contains the data based on which the Pie Chart is generated. The JSON array should contain data of column names and the corresponding values to generate a Pie Chart, in the key-value pair format. Here is the JSON array format:

## Return value:

None

**Sample Input Format**

var data =

[

{colorCode: <color> , label: "<label>", value: <value>},

{colorCode: <color>, label: "<label>", value: <value>}

];

this.view.componentID.createChart(data);

* The data must be passed in a proper format (without any missing values).
* The number of colour codes which are passed must be equal to the number of sectors.
* Never pass an empty JSON for colours.
* For example

var data =

[

{colorCode: #f0f1f2 , label: "Jan", value: 12},

{colorCode: #f0f1f2 , label: "Feb", value: 5}

];

this.view.componentID.createChart(data);

# 4. REVISION HISTORY

App version 1.1.1:

1. **Limitations**

The length of the data passed to the chart should be between **one** and **seven**. Exceeding the limit leads to UI distortions.

1. **Known Issues**

* On Android, when adding the component dynamically, the layout of the component does not show up as expected.
* Cannot handle layout properties of the component as per the device orientation. You must handle the properties at the form level.