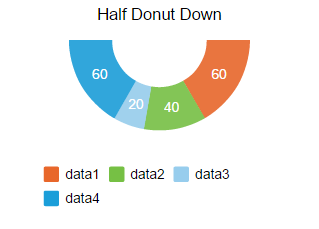
Date :  05-Feb-2023

Doughnut Chart - Inverse Half Circle

version: 1.1.2

# **Overview**

Doughnut Chart - Inverse Half Circle (also known as Half Donut Down Chart) is a Volt MX Iris component that creates a chart in the form of a half donut circle below the x-axis, 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 half donut. For examples, a company's profit percentage on annual basis.  


## **Use case:**

### Consider a scenario that you want to provide a company's sales information in your mobile application. In the app, you want to build a feature to represent a company's profit percentage over a few years in the form of half donut. You can use the Half Donut Down Chart component to represent the profit percentage with respect to the years in the form of half donut chart.

### You can also use the Half Donut Down Chart component to represent the department incomes.

## **Percentage of re-use:**

85%-90% (Data can be customizable and skins are customized and also can be changed manually).

# **Getting Started**

## **Prerequisites**

Before you start using the Half Donut Down Chart component, ensure you have the following:

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

## **Platforms Supported**

### Mobile

#### iOS

#### Android

### Tablets

### PWA

## **Importing the app**

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

**To import the Half Donut Down Chart component, do the following:**

1. Open your app project in Volt MX Iris.
2. In the Project Explorer, click the **Templates** tab.

Graphical user interface, text, application

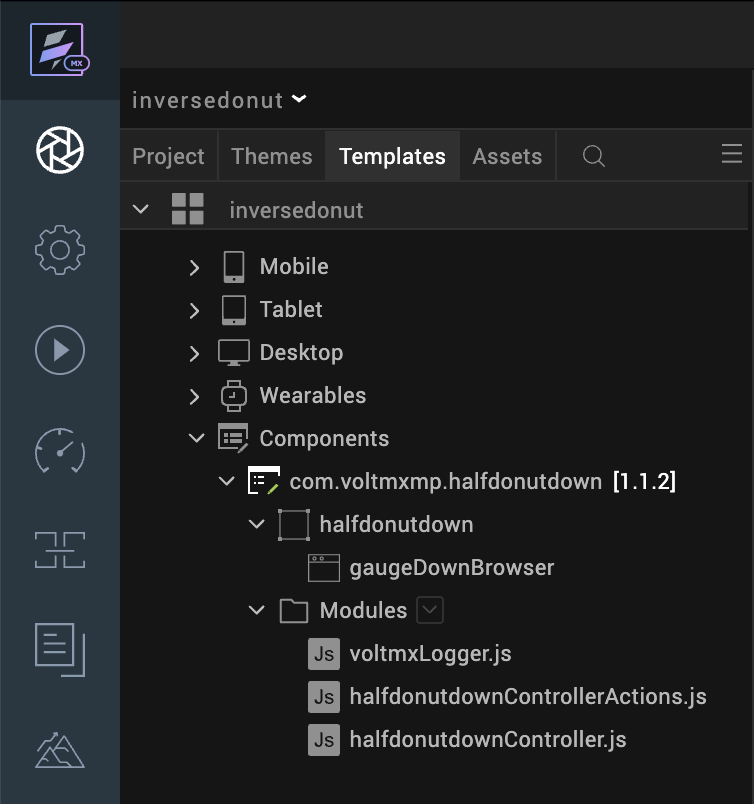
Description automatically generated

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

Graphical user interface, text, application, Teams

Description automatically generated

1. 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).

After adding a component to a form, you can configure the component the way you want it using the **Look**, **Skin**, and **Action** tabs on the **Properties** pane. Configuring the properties on the **Properties** pane is like configuring the properties of any widget in VoltMX Iris.

You can also see that a new tab, **Component**, is added on the **Properties** pane. The **Component** tab contains assorted properties relevant to the component that allow you to customize the component as required. The properties on the **Component** tab are categorized as **General**, **Legends**, and **Title** properties. The **General** properties are the default properties of individual widgets in the component.

# **References**

## **Dynamic Usage**

You can also add a Half Donut Down Chart component dynamically. To do so:

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

/\* Creating the component s object \*/

var Halfdonutdownchart = new com.voltmxmp.halfdonutdown(

{

"clipBounds": true,

"id": "Halfdonutdownchart",

"height": "60%",

"width": "100%",

"top": "0%",

"left": "0%",

"isVisible": true,

"zIndex": 1

}, {}, {});

/\* Setting the component s properties \*/

Halfdonutdownchart.bgColor = "#FFFFFF";

Halfdonutdownchart.enableChartAnimation = true;Halfdonutdownchart.enableStaticPreview = true;

Halfdonutdownchart.chartData =

{

"data":

[

{"colorCode": "#1B9ED9", "label": "Data1", "value": "40"},

{"colorCode": "#E8672B", "label": "Data2", "value": "20"},

{"colorCode": "#76C044", "label": "Data3", "value": "20"},

{"colorCode": "#FFC522", "label": "Data4", "value": "10"},

{"colorCode": "#97CDED", "label": "Data5", "value": "10"}

]

};

Halfdonutdownchart.enableLegend = true;

Halfdonutdownchart.legendFontColor= "#000000";

Halfdonutdownchart.legendFontSize= "8";

Halfdonutdownchart.chartTitle = "Half Circle below Horizontal";

Halfdonutdownchart.titleFontColor = "#000000";

Halfdonutdownchart.titleFontSize = 12;

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

this.view.add(Halfdonutdownchart);

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

1. Save the file.

## **Properties**

The properties provided on the **Component** tab allow you to customize the UI elements in the Half Donut Down 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.

### **General**

## **Background Color**

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

1. **Enable Chart Animation**

|  |  |
| --- | --- |
| **Description:** | Controls whether or not to enable the chart animation. |
| **Syntax:** | enableChartAnimation |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Remarks:** | Disabling the chart animation will also disable the grid animation. |
| **Example:** | this.view.componentID.enableChartAnimation= true; |

1. **Chart Data**

|  |  |
| --- | --- |
| **Description:** | Enables a user to provide the data to generate the chart. |
| Syntax: | chartData |
| **Type:** | Data Grid |
| **Read/Write:** | Write |
| **Remarks**: | The property is static and cannot be assigned dynamically. |

1. **Enable Chart with Static Data**

|  |  |
| --- | --- |
| **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; |
|  |  |

### **Legends**

1. **Enable Legends**

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

1. **Legend Font Color**

|  |  |
| --- | --- |
| **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**

|  |  |
| --- | --- |
| **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"; |

### **Title**

###### **i. Chart Title**

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

###### **ii. Title Font Size**

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

###### **iii. Title Font Color**

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

## **Events**

-- None of the events are exposed.

## **API’s**

The following API pertains to the Half Donut Down Chart component.

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

The API creates a Half Donut Down Chart.

**Syntax**

createChart(data)

**Parameters**

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

var data =

[

{"colorCode": "#1B9ED9", "label": "data1", "value": "25"},

{"colorCode": "#76C044", "label": "data2", "value": "20"}

];

In the above format, **label**, **value**, **colorCode** are keys and they are case sensitive.

* **label:** The key accepts string values, so define the value within the qoutation marks. You can specify upto 04 characters as a column name. For example, "data1". Specifying more than 04 characters results distortion in the component UI.
* **value:** The key accepts integer values.
* **colorCode:** The key accepts color codes. The value of the key must be a 6-character Hex color code preceded by the number sign (#).

The component can conveniently handle a maximum of 07 key-value pairs in the JSON array. Defining more than 07 key-value pairs results distortion in the component UI.

**Return Value**

None

**Example**

var chartData =

[

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

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

];

this.view.componentID.chartData={data: chartData};

this.view.componentID.createChart(chartData);

# **Revision History**

App version 1.1.2:

## **Known Issues**

Following are the known issues in the Half Donut Down Chart component:

* 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.
* If single data value is passed the data label is not visible if the background color is white.

## **Limitations**

Following are the limitations in the Half Donut Chart Down component:

* The length of the data passed to the chart should be between **one** and **seven**. Exceeding the limit leads to UI distortions.
* When creating the Half Donut Down chart the component is expecting the values for different data segments should not be too far off.  In case one segment’s data is much bigger than the remaining data segments it can be difficult to view the data with lower value clearly.  For example:

