Date: 05 Feb 2024

Vue Accordion

version: 1.0.2

# **Overview**

## The component contains a set of catalog items in a row. Every item contains a background image. When the user hovers on an item, the item expands. In the expanded state, the item contains the background image, a title and text. When the user moves the cursor to anywhere else on the screen, the item collapses. In the collapsed state, the item hides the title and text, but still shows a part of the background image.

## **Use case:**

1. You can use the component in scenarios such as: a retail app, where you want to display a catalog of product categories that a user can select to view the relevant products.
2. You can customize the image, the title, and the text that you want to display on every item. You can also customize the logic that you want to perform when a user clicks an item.

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

Approximate 90% of reuse.

## **Features**

* Display image, the title, and the text are customizable as required
* Business logic can be added easily on click of items

# **Getting Started**

## **Prerequisites**

Before you start using the Vue Accordion component, ensure the following:

• [HCL Foundry](https://manage.hclvoltmx.com/)

• Volt MX Iris

## **Platforms Supported**

## Responsive Web & PWA

## **Importing the app**

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

## **To import the Vue Accordion 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

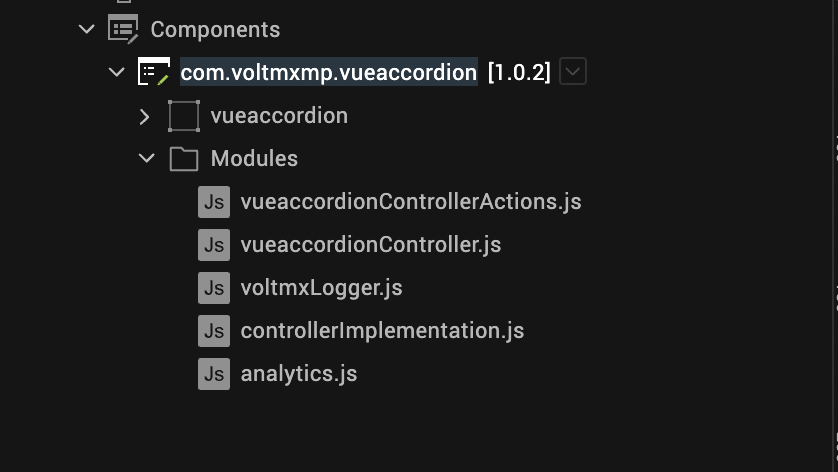
1. Right-click **Components**, and then select **Import Component**. The **Import Component** dialog box appears.

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" \l "add-a-component-to-a-form)



## **Building and previewing the app**

After performing all the above steps, you can build your app and run it on your device. For more information, you can refer to the [Building and Viewing an Application](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/Cloud_Build_in_VoltMX_Iris.html#cloud) section of the Volt MX User Guide.

You can then run your app to see the Vue Accordion work in real time.

# **3.References**

## **A. Dynamic Usage**

You can also addVue Accordioncomponent dynamically. To do so,

## 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.



/\*Creating Vue Accordion component instance\*/

createComponent: function()

{

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

var vueAccordion = new com.voltmxmp.vueaccordion(

{

"clipBounds": true,

"height": "265dp",

"id": "vueAccordion",

"isVisible": true,

"left": "0dp",

"top": "0dp",

"width": "100%",

"zIndex": 1

}, {}, {});

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

vueAccordion.masterData =

{

"data":

[

{

"id1": "1",

"id2": "First Title",

"id3": "This is the text for the first item.",

"id4": "https://unsplash.it/650/350/?image=101"

},

{

"id1": "2",

"id2": "Second Title",

"id3": "This is the text for the second item.",

"id4": "https://unsplash.it/650/350/?image=1"

},

{

"id1": "3",

"id2": "Third Title",

"id3": "This is the text for the third item.",

"id4": "https://unsplash.it/650/350/?image=20"

},

{

"id1": "4",

"id2": "Fourth Title",

"id3": "This is the text for the fourth item.",

"id4": "https://unsplash.it/650/350/?image=40"

},

{

"id1": "5",

"id2": "Fifth Title",

"id3": "This is the text for the fifth item",

"id4": "https://unsplash.it/650/350/?image=33"

},

{

"id1": "6",

"id2": "Sixth Title",

"id3": "This is the text for the sixth item.",

"id4": "https://unsplash.it/650/350/?image=44"

}

],

"schema":

[

{

"columnHeaderText":"id",

"columnHeaderType":"text",

"columnID":"id1",

"columnType":"text"

},

{

"columnHeaderText":"title",

"columnHeaderType":"text",

"columnID":"id2",

"columnType":"text"

},

{

"columnHeaderText":"text",

"columnHeaderType":"text",

"columnID":"id3",

"columnType":"text"

},

{

"columnHeaderText":"image",

"columnHeaderType":"text",

"columnID":"id4",

"columnType":"text"

}

]

};

vueAccordion.onClickItem = function(selectedItem){}.bind(this);

vueAccordion.onErrorCallback = function(errObj){}.bind(this);

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

this.view.add(vueAccordion);

}

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

**B. Properties**

You can use a component’s **Properties** to customize and configure the elements such as UI widgets, service parameters, and other elements.

You can set the properties from the **Properties** panel on the right side of Iris. You can also configure these properties by using a JavaScript code.

#### **General Properties**

1. **Master Data**

|  |  |
| --- | --- |
| **Description:** | Specifies the data that you want to display on the accordion. |
| **Syntax:** | masterData |
| **Type:** | * Data Grid * JSON |
| **Read/Write:** | Write |
| **Example:** | this.view.componentID.masterData =  {  “data”:  [  {  “id1”: “1”,  “id2”: “First Title”,  “id3”: “This is the text for the first item.”,  “id4”: “https://unsplash.it/650/350/?image=101”  },  {  “id1”: “2”,  “id2”: “Second Title”,  “id3”: “This is the text for the second item.”,  “id4”: “https://unsplash.it/650/350/?image=1”  },  {  “id1”: “3”,  “id2”: “Third Title”,  “id3”: “This is the text for the third item.”,  “id4”: “https://unsplash.it/650/350/?image=20”  },  {  “id1”: “4”,  “id2”: “Fourth Title”,  “id3”: “This is the text for the fourth item.”,  “id4”: “https://unsplash.it/650/350/?image=40”  },  {  “id1”: “5”,  “id2”: “Fifth Title”,  “id3”: “This is the text for the fifth item”,  “id4”: “https://unsplash.it/650/350/?image=33”  },  {  “id1”: “6”,  “id2”: “Sixth Title”,  “id3”: “This is the text for the sixth item.”,  “id4”: “https://unsplash.it/650/350/?image=44”  }  ],  “schema”:  [  {  “columnHeaderText”:”id”,  “columnHeaderType”:”text”,  “columnID”:”id1”,  “columnType”:”text”  },  {  “columnHeaderText”:”title”,  “columnHeaderType”:”text”,  “columnID”:”id2”,  “columnType”:”text”  },  {  “columnHeaderText”:”text”,  “columnHeaderType”:”text”,  “columnID”:”id3”,  “columnType”:”text”  },  {  “columnHeaderText”:”image”,  “columnHeaderType”:”text”,  “columnID”:”id4”,  “columnType”:”text”  }  ]  }; |

## **C. Events**

The component invokes events when its corresponding action is performed. You can configure logic that you want the component to perform when an event occurs.

You can configure the events on the **Actions** tab in the **Properties** panel. You can also configure the events by using a JavaScript code.

### **onClickItem**

|  |  |
| --- | --- |
| **Description:** | Invoked when the user clicks an item from the accordion. |
| **Syntax:** | onClickItem |
| **Parameters:** | *selectedItem[JSON]* : Contains the following keys   * id: The ID of the selected item * title: The title that is displayed on the selected item * text: The text that is displayed on the selected item * image: The URL of the background image of the selected item |
| **Example:** | this.view.componentID.onClickItem = function(selectedItem)  {  alert(“Item Clicked. Item Data: “+JSON.stringify(selectedItem));  }.bind(this); |

### [**onErrorCallback**](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Invoked when an error occurs in the component. |
| **Syntax:** | onErrorCallback |
| **Parameters:** | *errObj[JSON]* : Contains information about the error, such as the error code and the error message. |
| **Example:** | this.view.componentID.onErrorCallback = function(errObj)  {  alert(“Error: “+JSON.stringify(errObj));  }.bind(this); |
|  |  |

## **D. API’s**

The following APIs pertain to the Accordion – vue.js component:

###### [**setData**](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Displays the specified data on the accordion. |
| **Syntax:** | setData(data) |
| **Parameters:** | *data [Array of JSON]* : An array of objects that contain the ID, title, text, and image URL for every item on the accordion. |
| **Return Value:** | None |
| **Example:** | var data =  [  {  title: “First Title”,  text: “This is the text for the first item.”,  id: 1,  image: “https://unsplash.it/650/350/?image=101”  },  {  title: “Second Title”,  text: “This is the text for the second item.”,  id: 2,  image: “https://unsplash.it/650/350/?image=534”  },  {  title: “Third Title”,  text: “This is the text for the third item.”,  id: 3,  image: “https://unsplash.it/650/350/?image=9”  },  {  title: “Fourth Title”,  text: “This is the text for the fourth item.”,  id: 4,  image: “https://unsplash.it/650/350/?image=12”  },  {  title: “Fifth Item”,  text: “This is the text for the fifth item”,  id: 5,  image: “https://unsplash.it/650/350/?image=15”  }  ];  this.view.componentID.setData(data); |
|  |  |

# **REVISION HISTORY**

App version: 1.0.2

* + 1. **Known Issues**

No Known Issues

* + 1. **Limitations**

No Limitations