28 Oct 2021

Vue parallax (1.0.1)

1. Overview

Vue Parallax is a web component that contains multiple scenes. Every scene contains background image, captions, and a button. When a user scrolls the page, the background image moves slower than the rest of the content on the scene, creating a parallax effect. Vue Parallax is based on the Vue.js framework.

## Use case

### A Restaurant app, in which you want to showcase dishes or items on the home page.

## Features:

## Scrolls an image slower than the rest of the content on the scene, creating a parallax effect.

## You can customize the number of scenes.

## You can customize the background image and the captions.

## You can also customize the actions that you want to perform when the user clicks the button on a scene.

## Percentage of re-use:

Approximate 90% of reuse.

# Getting Started

## Prerequisites

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

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

• Volt MX Iris

## Platforms Supported

### PWA & Responsive Web

## Importing the app

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

## **To import the Vue Parallax component, do the following:**

## 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

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

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 Parallax work in real time.

## Modifying the UI of the scenes

You can modify the UI of the carousel by editing a CSS file. To access the CSS file, follow these steps:

1. From the **Project** explorer, go to **Web → Localfiles → vueparallax → custom-css.**
2. From the **custom-css** folder, select **overwrite.css**.
3. Use the text editor to edit the CSS based on your preference.

Text

Description automatically generated with medium confidence

## After you modify the CSS, you can Building and previewing the app to view the changes on the carousel.

## Setting actions for scene buttons

The buttons on every scene of the component invoke the **onSceneButtonClick** event. If you want to add different actions for every button, you need to perform the following steps:

1. Add a function to the form controller.
2. Invoke the function by using events.

1.) Add a function to the form controller

1. From the **Project** explorer, select the form controller of the form that contains the component.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. In the form controller, type the following code

|  |
| --- |
| navigateBetweenForms: function(id)  {  let formToNavigate = "frmHome";  switch (id)  {  case "location":  formToNavigate = "frmLocation";  break;  case "foodMenu":  formToNavigate = "frmFoodMenu";  break;  case "contact":  formToNavigate = "frmContact";  break;  }  var navObj = new voltmx.mvc.Navigation(formToNavigate);  navObj.navigate();  } |

**Note:**

The code snippet navigates to a form based on the ID of the scene that contains the button. You can edit the code and add the actions that you want to perform based on the ID of the scene.

2.) Invoking the function by using events

1. From the **Project** explorer, select the **vueparallax** component.

Text

Description automatically generated

1. From the **Properties** panel, go to **Action**.
2. For the **onSceneButtonClick** event, click **Edit**. The Action Editor opens.

A screenshot of a computer

Description automatically generated with medium confidence

1. From the list of actions, scroll to the **Functions** section, and then select **Add Snippet**.
2. In the code snippet, type the following code:

**this.navigateBetweenForms(id);**

**Note: The code snippet invokes the function that was added to the form controller. Make sure that you replace navigateBetweenForms with the name of the function in the form controller.**

# References

## Dynamic Usage

You can also add **Vue Parallax** 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.



|  |
| --- |
| /\*Creating the component's object \*/  var vueParallaxObj = new com.voltmxmp.vueparallax({  id: "vueparallax",  isVisible: true,  top:"0dp",  left:"0dp",  width:"100%",  height:"100%",  clipBounds: true,  layoutType: voltmx.flex.FREE\_FORM,  skin: "slFbox",  zIndex:1  },{},{});  /\* Setting the component's properties \*/  vueParallaxObj.masterData={  "data":  [  {  "id" : "parallax\_scene\_1",  "height" : "900",  "imageURL" : "resources/screen1.jpg",  "caption" : "FINE DINING",  "heading" : "Best Bistro",  "subHeading" : "Delicious French bistro style food in a relaxed casual atmosphere",  "btnText" : "VIEW OUR MENU"  },  {  "id" : "parallax\_scene\_2",  "height" : "900",  "imageURL" : "resources/screen2.jpg",  "caption" : "PERFECT GIFT",  "heading" : "Best Bistro Gift Card",  "subHeading" : "Gift a delicious experience today",  "btnText" : "BUY NOW"  }  ]  };  /\*Adding the component to a Form\*/  this.view.add(vueParallaxObj); |

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

2. Save the file

## Properties

The properties provided on the **Component** tab allows you to customize the elements in the **Vue Parallax** component. These elements can be UI elements, service parameters, and so on. You can set the properties from the Volt MX Iris Properties panel on the right-hand side. You can also configure these properties using a JavaScript code.

**Note:** If you copy code from any example, make sure that you replace componentID with the name of your component.

General Properties

1. masterData

|  |  |
| --- | --- |
| **Description:** | Specifies the data for the scenes that you want to display on the component. |
| **Syntax:** | masterData |
| **Type:** | * Data Grid * JSON |
| **Read/Write:** | Write |
| **Remarks:** | Every element in the data grid is a scene that contains the following keys:   * id *[String]* : An identifier for the scene * height *[String]* : The height (in px) of the background image of the scene * imageURL *[String]* : The URL of the image that you want to use as the background of the scene * caption *[String]* : The text that you want to display as the caption of the scene * heading *[String]* : The text that you want to display as the heading of the scene * subHeading *[String]* : The text that you want to display as the subheading of the scene * btnText *[String]* : The text that you want to display on the button that is on the scene |
| **Example:** | this.view.vueParallaxObj.masterData={  "data":  [  {  "id" : "parallax\_scene\_1",  "height" : "900",  "imageURL" : "resources/screen1.jpg",  "caption" : "FINE DINING",  "heading" : "Best Bistro",  "subHeading" : "Delicious French bistro style food in a relaxed casual atmosphere",  "btnText" : "VIEW OUR MENU"  },  {  "id" : "parallax\_scene\_2",  "height" : "900",  "imageURL" : "resources/screen2.jpg",  "caption" : "PERFECT GIFT",  "heading" : "Best Bistro Gift Card",  "subHeading" : "Gift a delicious experience today",  "btnText" : "BUY NOW"  }  ]  }; |

## Events

1.onSceneButtonClick

|  |  |
| --- | --- |
| **Description:** | Invoked when the user clicks the button on a scene. |
| **Syntax:** | onSceneButtonClick |
| **Parameters:** | *id [String]* : ID of the screen that contains the button that was clicked. |
| **Example:** | this.view.componentID.onSceneButtonClick = function(id)  {  alert("Button Clicked on Scene: "+id);  }.bind(this); |

2. onErrorCallback

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

## API’s

### setData

|  |  |
| --- | --- |
| **Description:** | Creates and displays scenes on the component based on the specified data. |
| **Syntax:** | setData(sceneData) |
| **Parameters:** | *sceneData [Array of JSON]* : The data for the scenes that you want to display on the component. |
| **Return Value:** | None |
| **Example:** | var sceneData =[{"id" : "parallax\_scene\_1","height" : "900","imageURL" : "resources/screen1.jpg","caption" : "FINE DINING","heading" : "Best Bistro","subHeading" : "Delicious French bistro style food in a relaxed casual atmosphere","btnText" : "VIEW OUR MENU"},{"id" : "parallax\_scene\_2","height" : "900","imageURL" : "resources/screen2.jpg","caption" : "PERFECT GIFT","heading" : "Best Bistro Gift Card","subHeading" : "Gift a delicious experience today","btnText" : "BUY NOW"}]this.view.componentID.setData(sceneData); |

# Revision History

App version 1.0.1:

## Known Issues

## -

## Limitations

-