28 Oct 2021

**Vue Hamburger Menu Component (1.0.1**)

# Overview

## Hamburger menu allows you to hide the navigation beyond the edge of the

## screen and reveal it only after a user’s action.

## Use case

## Screen space is a precious commodity on mobile and the slide menu is one of the most popular mobile navigation patterns that helps you save it.

## The drawer panel allows you to hide the navigation beyond the edge of the screen and reveal it only after a user’s action.

## This pattern can be particularly useful if you want your user to focus on the main content.

## This component provides a ready to use Sliding menu for VoltMX Apps. This component is based on the Vue.js framework.

## Percentage of re-use:

Approximate 85% of reuse.

## Features

* Enriches your app with popular navigation pattern, Hamburger Menu
* Flexible to display and hide the menu
* Saves the screen space
* Menu slides with animation onto the screen
* Provides controls to customize the UI and to set events

# Getting Started

## Prerequisites

Before you start using the Hamburger menu 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 Hamburger menu component, do the following:**

## Open your app project in Volt MX Iris.

## In the Project Explorer, click the Templates Tab

 

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



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.

## Text  Description automatically generated

Once you have imported a component to your project, you can easily add the component to a form.

## D. Building and previewing the app

After performing all the above steps, you can build your app and run it on your device.

 You can then run your app to see the Hamburger menu work in real time.

# 3.References

## Dynamic Usage

You can also add **Hamburger Menu** component 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 Hamburger Menu component instance\*/

define({

 //Type your controller code here

 onNavigate : function(){

 this.view.preShow=this.preShow();

 },

 preShow : function(){

 this.createComponent();

 },

 createComponent: function()

 {

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

 var vueHamburgerMenu = new com.voltmxmp.vuehamburgermenu(

 {

 "clipBounds": true,

 "height": "100%",

 "id": "vueHamburgerMenu",

 "isVisible": true,

 "left": "0dp",

 "top": "0dp",

 "width": "300px",

 "zIndex": 1

 }, {}, {});

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

 vueHamburgerMenu.menuItems =

 {

 "data":

 [

 {

 "icon": "resources/home.png",

 "id": "01",

 "title": "HOME"

 },

 {

 "icon": "resources/pro.png",

 "id": "02",

 "title": "PRODUCT"

 }

 ]

 };

 vueHamburgerMenu.defaultWidth = 300;

 vueHamburgerMenu.defaultVisibility = false;

 vueHamburgerMenu.defaultMenuItem = "01";

 vueHamburgerMenu.logoVisibility = true;

 vueHamburgerMenu.logoURL = "./resources/voltmx\_logo.png";

 vueHamburgerMenu.menuIconURL = "./resources/menu.png";

 vueHamburgerMenu.closeIconURL = "./resources/close.png";

 vueHamburgerMenu.onMenuItemClick = function(id) {}.bind(this);

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

 vueHamburgerMenu.onShowHamburgerMenu = function(){}.bind(this);

 vueHamburgerMenu.onHideHamburgerMenu = function(){}.bind(this);

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

 this.view.add(vueHamburgerMenu);

 }

});

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

## Properties

The properties provided on the **Component** tab allow you to customize the UI elements in the Hamburger Menu component. You can set the properties directly on the **Component** tab or by writing a JavaScript. This section provides information on how to set the properties by writing a JavaScript.

####  General properties

Menu Items

|  |  |
| --- | --- |
|  **Description:** | Specifies the data that you want to show as the items on the menu. |
|  **Syntax:** | menuItems |
|  **Type:** | * Data Grid
* JSON
 |
|  **Read/Write:** | Write |
|  **Default Value:** |  {"data":[ { "icon": "resources/home.png", "id": "01", "title": "HOME" }, { "icon": "resources/pro.png", "id": "02", "title": "PRODUCT" }]} |
| **Example:** | this.view.componentID.menuItems ={"data":[ { "icon": "resources/home.png", "id": "01", "title": "HOME" }, { "icon": "resources/pro.png", "id": "02", "title": "PRODUCT" }]}; |
|  |  |

Default Width in PX

|  |  |
| --- | --- |
|  **Description:** | Specifies the width of the menu (in pixels) in the expanded state. |
|  **Syntax:** | defaultWidth |
|  **Type:** | Integer |
| **Read/Write:** | Write |
| **Default Value:** | 300 |
| **Example:** | this.view.componentID.defaultWidth = 300; |

Default Visibility

|  |  |
| --- | --- |
| **Description:** | Specifies whether the menu should expand by default. |
| **Syntax:** | defaultVisibility |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Default Value:** | true |
| **Example:** | this.view.componentID.defaultVisibility = false; |

Default Menu ID

|  |  |
| --- | --- |
| **Description:** | Specifies the ID of the menu item that should be highlighted by default. |
| **Syntax:** | defaultMenuItem |
| **Type:** | String |
| **Read/Write:** | Write |
| **Default Value:** | 01 |
| **Example:** | this.view.componentID.defaultMenuItem = "01"; |

Logo Visibility

|  |  |
| --- | --- |
| **Description:** | Toggles the visibility of the logo. |
| **Syntax:** | logoVisibility |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Default Value:** | true |
| **Example:** | this.view.componentID.logoVisibility = true; |

Logo URL

|  |  |
| --- | --- |
| **Description:** | Specifies the URL of the image that you want to set as the logo. |
| **Syntax:** | logoURL |
| **Type:** | String |
| **Read/Write:** | Write |
| **Default Value:** | ./resources/quantum.png |
| **Example:** | this.view.componentID.logoURL = "./resources/quantum.png"; |

Burger Menu Icon URL

|  |  |
| --- | --- |
| **Description:** | Specifies the URL of the image that you want to set as the burger menu icon. |
| **Syntax:** | menuIconURL |
| **Type:** | String |
| **Read/Write:** | Write |
| **Default Value:** | ./resources/menu.png |
| **Example:** | this.view.componentID.menuIconURL = "./resources/menu.png"; |

Close Icon URL

|  |  |
| --- | --- |
| **Description:** | Specifies the URL of the image that you want to set as the close icon. |
| **Syntax:** | closeIconURL |
| **Type:** | String |
| **Read/Write:** | Write |
| **Default Value:** | ./resources/close.png |
| **Example:** | this.view.componentID.closeIconURL = "./resources/close.png"; |

### C. Events

You can define events to be executed when an action is performed. You can configure the events directly on the **Actions** tab or by writing a JavaScript. To configure the events on the **Action** tab, click **Edit** against each event.

 onMenuItemClick

|  |  |
| --- | --- |
| **Description:** | Invoked when the user selects an item from the menu. |
| **Syntax:** | onMenuItemClick |
| **Parameters:** | *id [String]* :The ID of the item that the user selected from the menu. |
| **Example:** | this.view.componentID.onMenuItemClick = function(id){ alert("Menu Item Selected: "+id);}.bind(this); |

onErrorCallback

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

onShowHamburgerMenu

|  |  |
| --- | --- |
| **Description:** | Invoked when the user clicks the burger menu icon and expands the menu. |
| **Syntax:** | onShowHamburgerMenu |
| **Parameters:** | None |
| **Example:** | this.view.componentID.onShowHamburgerMenu = function(){ alert("Menu Opened.");}.bind(this); |

onHideHamburgerMenu

|  |  |
| --- | --- |
| **Description:** | Invoked when the user clicks the close icon or a menu item and hides the menu. |
| **Syntax:** | onHideHamburgerMenu |
| **Parameters:** | None |
| **Example:** | this.view.componentID.onHideHamburgerMenu = function(){ alert("Menu hidden.");}.bind(this); |

## D. **APIs**

The following APIs pertain to the Hamburger Menu - vue.js component:

setMenuItems

|  |  |
| --- | --- |
| **Description:** | Sets the items on the menu based on the specified data.. |
| **Syntax:** | setMenuItems(data) |
| **Parameters:** | *data [Array of JSON]* :Contains the **ID**, the **Title**, and the **Icon URL** for the items that you want to display on the menu. |
| **Return Value:** | None |
| **Example:** | var data =[ { "icon": "resources/home.png", "id": "01", "title": "HOME" }, { "icon": "resources/pro.png", "id": "02", "title": "PRODUCT" }];this.view.componentID.setMenuItems(data); |

 selectMenuItem

|  |  |
| --- | --- |
| **Description:** | Selects the specified item from the menu. |
| **Syntax:** | selectMenuItem(id) |
| **Parameters:** | *id [String]* :The ID of the item that you want to select from the menu. |
| **Return Value:** | None |
| **Remarks:** | Selecting an item by using the **selectMenuItem** API does not close the menu. |
| **Example:** | var id = "02";this.view.componentID.selectMenuItem(id); |

# REVISION HISTORY

 App version: 1.0.1

## Limitations

#### Vue Hamburger menu is a web component. So, it doesn’t support for Android and IOS devices.