Miva Merchant

MMMenuButton Guide

Miva Merchant 9

MM9-MISC-306-1.0 3/31/15

Scope

This document describes how to create an **MMMenuButton** in Miva Merchant by defining a JavaScript class. Experience with JavaScript is presumed.

MMMenuButton is a complex class, designed for flexibility. This document covers the basics. For more in depth customization, a thorough examination of the source code (available in **admin/mmmenubutton.js**) is recommended.

Creating an MMMenuButton

The basic setup of an **MMMenuButton** is as follows:

```
...
var menubutton, item1, item2;


menubutton = new MMMenuButton( 'Button Text', element_parent );
menubutton.ContainedButton().SetImage( 'cancel' );
 // ContainedButton() gives you access to ALL functions from the MMButton class
menubutton.ContainedButton().SetHoverText( 'Stop the save process' );
item1 = menubutton.Menu_Append_Item( 'Option 1', function( e, data ) { alert(
 data + ' clicked' ); }, 'option_1' );
item2 = menubutton.Menu_Append_Item( 'Option 2', function( e, data ) { alert(
 data + ' clicked' ); }, 'option_2' );
...
```

MMMenuButton uses the **MMButton** class to control the button that toggles the visibility of the menu. There are a number of other **Set**/**Get** functions that allow finer control over your **MMButton**. Following is a list of all current modification functions:

- **menubutton.ContainedButton().[*any* Set*XXX function available to the* MMButton *class*]**
- **menubutton.SetText(text);**
- **menubutton.Feature_Dropdown_Enable(padding_right);**
  /* Shows the dropdown arrow. Purely a visual change (functionality does not change by calling this function) */
- **menubutton.SetClassName(classname);**
- **menubutton.SetAllowMiddleClick(true | false);**
- **menubutton.SetAllowRightClick(true | false);**
- **menubutton.SetOnClickHandler(function(e) { ; });**
- **menubutton.Menu_Empty();**

Note: All the **Menu_Append_*XXX*** functions return the appended item class (not a DOM element) that can be used to apply actions in certain cases — such as calling **item.SetSelectable(true | false)** to enable or disable that item's ability to be clicked.

Adding Menu Options

Following is a list of available functions for adding options to the menu:

menubutton.Menu_Append_Section_Header(text);	Appends a header element (not clickable) that can be styled differently than a normal menu item
menubutton.Menu_Append_Item(text, callback, data);	Generic function to append a menu item; "text" can be a text string OR an element
menubutton.Menu_Append_Item_Upload(text, multiple, callback, data);	Appends an **upload button** option. Clicking the option will open a file-open dialog. The callback will be passed the selected file for you to handle.
menubutton.Menu_Append_Item_Formatted(element, callback, data);	(Legacy) Identical to **Menu_Append_Item** when passing a DOM element for the text parameter. Use **Menu_Append_Item (element, callback, data)** instead of this function.
menubutton.Menu_Append_Item_Formatted_ NonSelectable(element);	Appends a non-selectable/non-clickable item to the list
menubutton.Menu_Append_Item_Sort(text, callback, data);	Appends a **sort** menu item that shows an "ascending", "descending", or "none" icon. Set by calling **item.SetItemState_Ascending()**, **item.SetItemState_Descending()**, or **item.SetItemState_None()** on the appended item.
menubutton.Menu_Append_Item_Toggle(text, callback, data, active);	Appends a **toggle** menu item that shows an "on" or "off" state for the menu item. Click once and its state is "on", click again and its state is "off". Manual setting of the active state can be triggered by calling **item.SetActive(true | false)** on the appended item.
menubutton.Menu_Append_Item_SubMenu(text, callback, data);	Appends a **sub menu container**. Sub menu containers have the ability to have their own actions. For example, clicking the text of the submenu container can have an action, while clicking the expand arrow will open the sub menu.
menubutton.Menu_Append_Divider();	Appends a horizontal line separator to separate groups of items in the menu

Adding Sub Menus

Using the **Menu_Append_Item_SubMenu** function allows you to add an entirely new set of menu items as a sub menu to the added item.

Example:

```
/* Structure:
    Root Menu:
        Option 1
        Option 2
        Option 3
            Option 3-1
            Option 3-2
            Option 3-3
        Option 4
*/

var menubutton, submenu;

menubutton = new MMButton( 'Button Text', element_parent );
menubutton.ContainedButton().SetImage( 'cancel' );
 // ContainedButton() gives you access to ALL functions from the MMButton class
menubutton.ContainedButton().SetHoverText( 'Stop the save process' );
menubutton.Menu_Append_Item( 'Option 1', function( e, data ) { alert( data + '
 clicked' ); }, 'option_1' );
menubutton.Menu_Append_Item( 'Option 2', function( e, data ) { alert( data + '
 clicked' ); }, 'option_2' );
submenu = menubutton.Menu_Append_Item_SubMenu( 'Option 3', function( e, data )
 { alert( data + ' clicked' ); }, 'option_3' );
menubutton.Menu_Append_Item( 'Option 4', function( e, data ) { alert( data + '
 clicked' ); }, 'option_4' );


submenu.Menu_Append_Item( 'Option 3-1', function( e, data ) { alert( data + '
 clicked' ); }, 'option_3-1' );
submenu.Menu_Append_Item( 'Option 3-2', function( e, data ) { alert( data + '
 clicked' ); }, 'option_3-2' );
submenu.Menu_Append_Item( 'Option 3-3', function( e, data ) { alert( data + '
 clicked' ); }, 'option_3-3' );
```

Controlling the MMMenuButton

The following list of functions is used to control the **MMMenuButton**.

menubutton.MenuParentContainer();	Returns the menu element's **parentNode** (menu element is the dropdown)
menubutton.MenuItemContainer();	Returns the menu element
menubutton.GetZIndex();	Returns the z-index value of the menu
menubutton.Menu_IsEmpty();	Check whether the menu is empty
menubutton.Menu_Toggle();	Toggle the visibility of the menu
menubutton.Menu_Show());	Show the menu (if there are any menu items)
menubutton.Menu_Hide();	Hide the menu
menubutton.Container();	Return the container element (DOM)
menubutton.ContainedButton();	Return the **MMButton** class for the menu button (used to toggle the menu)
menubutton.Show();	Show the menu button
menubutton.Hide();	Hide the menu button
menubutton.Visible();	Check the visibility of the menu button (not the menu dropdown)

Specific to items added to the **MMMenuButton:**

item.Expand();	For items that have submenu items, this shows the submenu
item.Collapse();	For items that have submenu items, this collapses the submenu
item.Select();	
item.Deselect();	
item.SetSelectable(true | false);	

Callback Functions

menubutton.onmenuresize();	Called any time **Menu_Resize** is called
menubutton.onmenushow();	Called when the menu dropdown is made visible
menubutton.onmenuhide();	Called when the menu dropdown is hidden

Examples

Examples of **MMMenuButtons** can be found on the Miva Merchant admin screens.

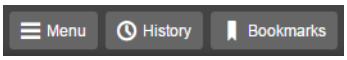

Figure 1: Buttons with text and image

Figure 2: Buttons with image only

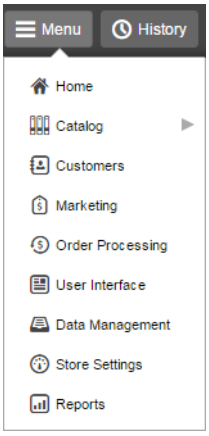

Figure 3: Menu button with dropdown list

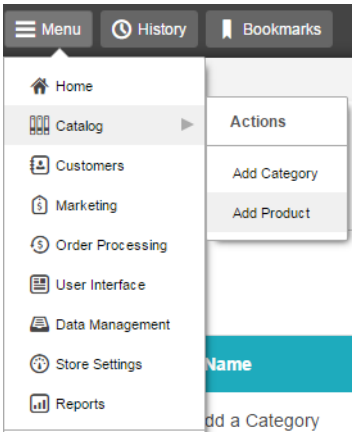

Figure 4: Menu button with submenu

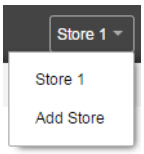

Figure 5: Menu button with text only