



Miva Merchant
MMMenuButton Guide

Miva Merchant 9

© Copyright 2005–2015, Miva®, Inc.

Miva Merchant® and Miva Central® are registered trademarks of Miva®, Inc.

UPS, THE UPS SHIELD TRADEMARK, THE UPS READY MARK, THE UPS DEVELOPER KIT MARK AND THE COLOR BROWN ARE TRADEMARKS OF UNITED PARCEL SERVICE OF AMERICA, INC. ALL RIGHTS RESERVED.

All rights reserved. The information and intellectual property contained herein is confidential between Miva® Inc and the client and remains the exclusive property of Miva® Inc. If you find any problems in the documentation, please report them to us in writing. Miva® Inc does not guarantee that this document is error free. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of Miva® Inc.

This document, and all materials, products and postings are made available on an “as is” and “as available” basis, without any representation or warranty of any kind, express or implied, or any guaranty or assurance the document will be available for use, or that all products, features, functions or operations will be available or perform as described. Without limiting the foregoing, Miva® Inc is not responsible or liable for any malicious code, delays, inaccuracies, errors, or omissions arising out of your use of the document. As between you and Miva® Inc, you are assuming the entire risk as to the quality, accuracy, performance, timeliness, adequacy, completeness, correctness, authenticity, security and validity of any and all features and functions of the document.

The Miva Merchant® logo, all product names, all custom graphics, page headers, button icons, trademarks, service marks and logos appearing in this document, unless otherwise noted, are trademarks, service marks, and/or trade dress of Miva® Inc (the “Marks”). All other trademarks, company names, product names, logos, service marks and/or trade dress displayed, mentioned or otherwise indicated on the Web Site are the property of their respective owners. These Marks shall not be displayed or used by you or anyone else, in any manner, without the prior written permission of Miva® Inc. You agree not to display or use trademarks, company names, product names, logos, service marks and/or trade dress of other owners without the prior written permission of such owners. The use or misuse of the Marks or other trademarks, company names, product names, logos, service marks and/or trade dress or any other materials contained herein, except as what shall be permitted herein, is expressly prohibited.

© Copyright 2005–2015, Miva®, Inc. All Rights Reserved.

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 MMenuButton

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

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

menubutton = new MMenuButton( 'Button Text', element_parent );
menubutton.ContainedButton().SetImage( 'cancel' );
// ContainedButton() gives you access to ALL functions from the MMenuButton 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 **MMenuButton** 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 **MMenuButton**. Following is a list of all current modification functions:

- **menubutton.ContainedButton().[any SetXXX function available to the MMenuButton 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:

<code>menubutton.Menu_Append_Section_Header(text);</code>	Appends a header element (not clickable) that can be styled differently than a normal menu item
<code>menubutton.Menu_Append_Item(text, callback, data);</code>	Generic function to append a menu item; “text” can be a text string OR an element
<code>menubutton.Menu_Append_Item_Upload(text, multiple, callback, data);</code>	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.
<code>menubutton.Menu_Append_Item_Formatted(element, callback, data);</code>	(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.
<code>menubutton.Menu_Append_Item_Formatted_NonSelectable(element);</code>	Appends a non-selectable/non-clickable item to the list
<code>menubutton.Menu_Append_Item_Sort(text, callback, data);</code>	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.
<code>menubutton.Menu_Append_Item_Toggle(text, callback, data, active);</code>	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.
<code>menubutton.Menu_Append_Item_SubMenu(text, callback, data);</code>	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.
<code>menubutton.Menu_Append_Divider();</code>	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 MMenuButton

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

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 MButton 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 **MMenuButton**:

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 **MMenuButtons** can be found on the Miva Merchant admin screens.

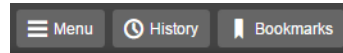


Figure 1: Buttons with text and image



Figure 2: Buttons with image only

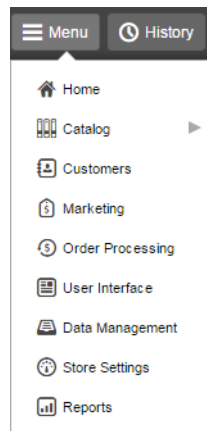


Figure 3: Menu button with dropdown list

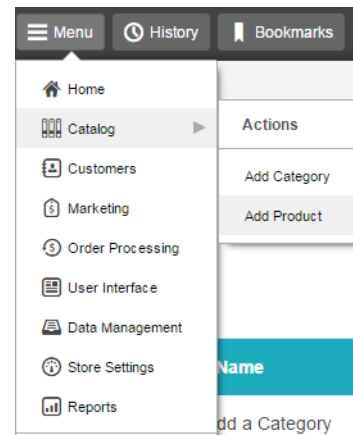


Figure 4: Menu button with submenu

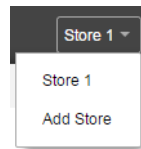


Figure 5: Menu button with text only

