



AccordionTreeMenu version 1.2.1

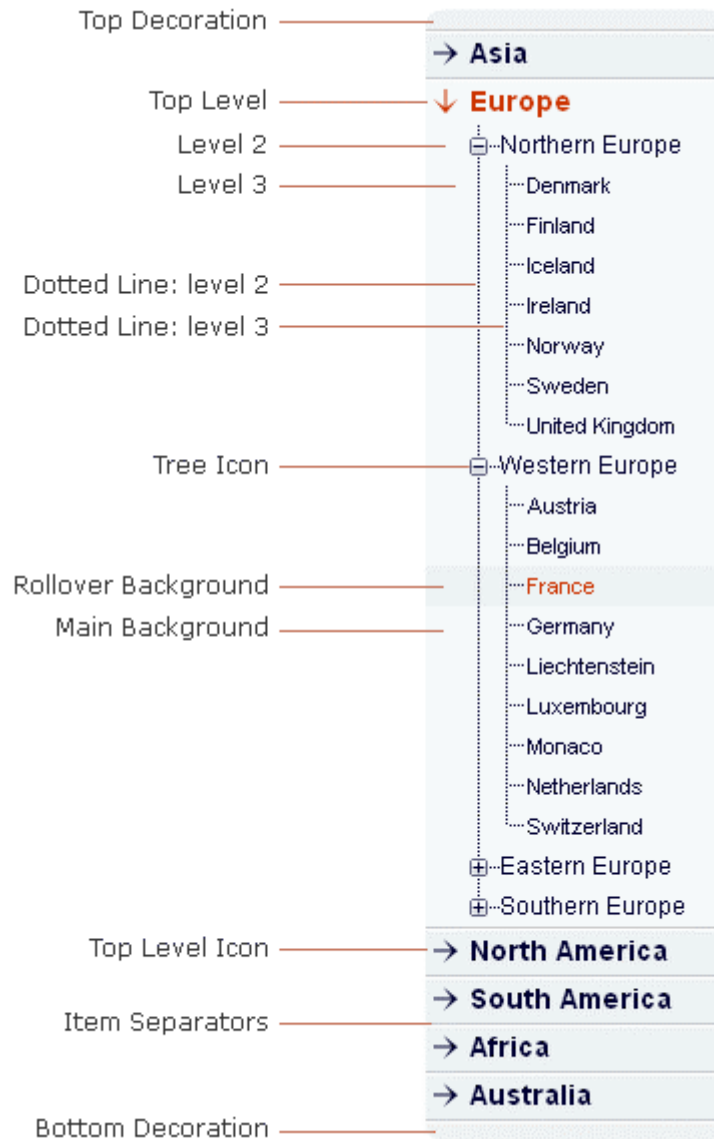
ADVANCED COMPONENT (FLA version)
FOR FLASH 8, FLASH CS3/CS4

Overview

The Accordion Tree Menu is a Flash component that represents a tree menu structure in an accordion fashion. It allows you to implement a hierarchical menu into your website navigation system.

The component provides with great usability supporting most complete feature set:

- multilevel structure (up to 3 levels) and unlimited menu items
- menu data is loaded using XML
- component can be completely configured through the XML file
- separate style declaration for each of three levels of menu structure
- alpha transparency support for all backgrounds
- adding URLs or custom Flash functions to any menu click event (through the XML file)
- adding sounds to click or rollover event (through the XML file)
- ActionScript API available (methods and properties of the AccordionTreeMenu class).



The structure of Accordion Tree Menu includes three possible levels:

1. *Top level* — consists of items expanding and folding in accordion fashion. Move your mouse over a top level item, and nested second level sub-menu (if one exists) drops down. Only one top level item can be open at a time.
2. *Second level* (optional) — consists of items expanding and folding in tree fashion. Click on tree icon, and nested third level sub-menu moves up or down. Multiple second level items can stay open at the same time.
3. *Third level* (optional) — the deepest level of Accordion Tree Menu hierarchy.

You can work with the Accordion Tree Menu in one of the following ways:

1. Using the source FLA file (editable Flash file) for further publishing (creating SWF file).
2. Using the SWF file (fully working Flash movie). In this case, you don't need Flash software or any programming knowledge.

Using Accordion Tree Menu

The Accordion Tree Menu can be used to create the professionally looking navigation system for websites of any complexity. Using parameter settings in the XML file and ActionScript methods and properties available for the component, you can build advanced and full-featured navigation systems. Elegant design and excellent dynamics will create unique and attractive appearance for different types of Flash and HTML-based web applications.

Use Accordion Tree Menu to create any of the following:

- Accordion menus
- Tree structure menus
- Combinations of accordion menu and tree structure menu
- Vertical rollover menus

This is not the complete list of possible usages, just ones that are likely to be used most often. Your creative power and imagination can greatly extend the scope of the Accordion Tree Menu component.

You can fully use the Accordion Tree Menu by editing the provided XML file using any text editor (such as Notepad for example). This will allow you to edit all the parameters.

You can create an application with the Accordion Tree Menu in one of the following ways:

1. Using the source FLA file (editable Flash file) for further publishing (creating SWF file)

In this case, you will need Macromedia Flash 8 or Adobe Flash CS3/CS4 software. The following files are required (included in the package):

- AccordionTreeMenu.fla;
- AccordionTreeMenu.as;
- menu.xml.

Edit the parameters in the XML file (menu.xml) according to your needs. Open the source FLA file (AccordionTreeMenu.fla) and publish the Flash document with appropriate name.

2. Copying the Accordion Tree Menu library assets from the source FLA file into another document

In this case, you will need Macromedia Flash 8 or Adobe Flash CS3/CS4 software. The following files are required (included in the package):

- AccordionTreeMenu.fla;
- AccordionTreeMenu.as;
- menu.xml.

Edit the parameters in the XML file (menu.xml) according to your needs. Open the source FLA file (AccordionTreeMenu.fla) and the destination FLA file. Drag the "Accordion Tree Menu" folder from the source document library onto the Stage of the destination document. Delete the copied assets from the Stage. Select the "AccordionTreeMenu" symbol (Movie Clip) in the Library panel in the destination document and drag it onto the Stage. When publishing the Flash document, note that the ActionScript file — AccordionTreeMenu.as — must be in the same directory with the destination FLA file.

3. Using the SWF file (fully working Flash movie)

In this case, you don't need Flash software or any programming knowledge. The following files are required (included in the package):

- menu.swf;
- menu.xml.

Edit the parameters in the XML file (menu.xml) according to your needs.

If you need to change the name of the XML file, you can easily do it. Just open the AccordionTreeMenu.as file (before publishing the Flash document) and replace "menu.xml" with the name of your XML file. You can also specify the name for the XML file when embedding the Accordion Tree Menu into HTML page.

Below is the code that describes how to embed the Accordion Tree Menu using SWFObject 2:

```
<head>
  <script type="text/javascript" src="swfobject.js"></script>
  <script type="text/javascript">
    var flashvars = {xmlURL: "menu.xml"};
    var params = {};
    var attributes = {};
    swfobject.embedSWF("menu.swf", "menu", "290", "800", "8.0.0", false, flashvars, params, attributes);
  </script>
</head>
<body>
  <div id="menu">
    <a href="http://www.adobe.com/go/getflashplayer">
      
    </a>
  </div>
</body>
```

XML file structure

The basic template of the XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<menu width="200">
  <properties>
    <treeStructure state="open"/>
    <itemHeight level1="24" level2="20"/>
    <topLevelIcons enabled="true"/>
    <dottedLine level2="true" level3="true"/>
    <decoration height="5" rounded="right"/>
    <leftMargin value=""/>
    <itemIndent level2="true" level3="true"/>
    <itemSeparators level1="true" level2="false"/>
    <tween duration="0.3" easing="regular" method="easeOut"/>
    <rolloverSound src="" volume=""/>
    <clickSound src="hifi.mp3" volume="80"/>
    <initialStateReturn enabled="true"/>
    <selectedStateMode enabled="true"/>
    <internalSelectionMode enabled="false"/>
  </properties>
```

```

<style>
  <mainBackground color="F5F9FA" alpha="30"/>
  <decorationBackground color="E8EEF0" alpha=""/>
  <topLevelBackground color="E8EEF0" alpha=""/>
  <treelcon color="" bgcolor="F5F9FA"/>
  <dottedLine color=""/>
  <separatorLine color1="CCCCCC" color2="FFFFFF"/>
  <rolloverBackgroundColor level1="F5F9FA" level2="E8EEF0"/>
  <rolloverBackgroundAlpha level1="" level2="60"/>
  <selectedBackgroundColor level2="E8EEF0"/>
  <selectedBackgroundAlpha level2="60"/>
  <fontSize level1="14" level2="12" level3="11"/>
  <fontFamily name="Arial"/>
  <embedFonts enabled="false"/>
  <fontColor level1="000033" level2="000033" level3="000033"/>
  <fontColorRollover level1="CC3300" level2="CC3300" level3="CC3300"/>
  <fontColorSelected level2="CC3300" level3="CC3300"/>
  <fontWeight level1="bold" level2="" level3=""/>
  <fontWeightRollover level1="bold" level2="" level3=""/>
  <fontWeightSelected level2="" level3=""/>
  <fontStyle level1="" level2="" level3=""/>
  <textDecorationRollover level1="" level2="" level3=""/>
</style>
<items>
  <item label="Europe" action="getUrl" url="/world/europe/" target="_self" font_color="">
    <item label="Northern Europe" action="" url="" target="" font_color="">
      <item label="Norway" action="showMap" url="map3" target="_level0.maps"
        font_color=""/>
    </item>
  </item>
  .....
</items>
</menu>

```

Note: If the value of a property in the XML file is undefined (omitted), it will be ignored.

The description of the XML file elements

1. The **<menu>** element has one optional attribute `width` which sets the width of an Accordion Tree Menu component instance.

Note: You can transform an Accordion Tree Menu component horizontally and vertically through the XML file and at runtime. At runtime, use the `menuWidth` ActionScript property.

The width of the bounding box of an Accordion Tree Menu component designates the width of the hit area for the component. If you increase the width of the component, you also increase the width of the hit area. The default value for a component's size is 200 x 100 pixels. If the component's bounding box is too small to fit the component's labels and graphic elements, they are clipped to fit.

Transforming a component instance horizontally reduces or increases its width and does not resize the graphic elements and item labels of the menu. Change the component's width to fit the component labels if necessary.

2. The **<properties>** element determines mostly how Accordion Tree Menu component behaves. The following table describes the attributes of nodes that the **<properties>** element contains:

Node name	Attribute name	Default	Description
treeStructure	state	"close"	<p>Specifies the appearance and initial state of Accordion Tree Menu structure. Possible values: "open", "close", "static".</p> <p>a) "open" — Each second level menu item that has nested third level sub-menu is in open state, and can be closed by clicking on appropriate tree icon.</p> <p>b) "close" — Each second level menu item that has nested third level sub-menu is in closed state, and can be opened by clicking on appropriate tree icon.</p> <p>c) "static" — Each second level menu item that has nested third level sub-menu stays open permanently. No tree icons are drawn.</p> <p><i>Note:</i> The component's structure appearance is also depends on values combination of the following parameters: topLevelIcons, itemIndent (level2) and dottedLine (level2).</p>
itemHeight	level1	24	Indicates the height of each top level item, in pixels. The value of this parameter is controlled programmatically, considering specified font size and font family.
	level2	20	Indicates the height of each second and third level item, in pixels. The value of this parameter is controlled programmatically, considering specified font size and font family.
topLevelIcons	enabled	"true"	<p>Determines whether to draw the arrow icons for the top level menu items. Possible values: "true", "false".</p> <p><i>Note:</i> If both topLevelIcons and itemIndent (level2) parameters are set to "false", then neither tree icons nor dotted lines for second level sub-menus are drawn, and each second level menu item (if it has nested third level sub-menu) stays open permanently.</p>
dottedLine	level2	"true"	Indicates whether the dotted lines for second level sub-menus to be drawn. Possible values: "true", "false". If dottedLine (level2) parameter is set to "false", then no tree icons are drawn, and each second level menu item (if it has nested third level sub-menu) stays open permanently.

	level3	"true"	Indicates whether the dotted lines for third level sub-menus to be drawn. Possible values: "true", "false".
decoration	height	0	Indicates the height of the top and bottom decorations, in pixels. Possible values: from 0 to the value of itemHeight (level1). If decorationHeight is equal to 0, then no decoration will be drawn.
	rounded	"all"	Specifies which corners of the menu decoration to be drawn rounded. Possible values: "left", "right", "all", "none".
leftMargin	value	0	Indicates the left margin of the menu, in pixels. This parameter accepts values from 0 to 100.
itemIndent	level2	"true"	Indicates whether to indent the second level items. The amount of indent is determined programmatically. Possible values: "true", "false". <i>Note:</i> If both itemIndent (level2) and topLevelIcons parameters are set to "false", then neither tree icons nor dotted lines for second level sub-menus are drawn, and each second level menu item (if it has nested third level sub-menu) stays open permanently.
	level3	"true"	Indicates whether to indent the third level items. The amount of indent is determined programmatically. Possible values: "true", "false".
itemSeparators	level1	"true"	Determines whether to draw separator lines between the top level menu items. Possible values: "true", "false".
	level2	"false"	Determines whether to draw separator lines between the second level menu items and between the third level menu items. Possible values: "true", "false".
tween	duration	0.3	A number indicating the duration of the tweened animation in seconds. The tweened animation is applied to each third level sub-menu during its movement up and down. Possible values: from 0.05 to 0.5.
	easing	"regular"	Specifies which easing class to be applied when opening or closing a second level menu item. Possible values: "regular", "none".
	method	"easeOut"	Specifies which easing method to be applied when opening or closing a second level menu item. Possible values: "easeIn", "easeOut", "easeNone".

rolloverSound	src	undefined	The path to an external mp3 file loaded into a movie clip and attached to the menu item's rollover event. If no path is specified, the rollover sound will not be used. If the path is incorrect, the rollover sound will not be used.
	volume	100	A number representing a volume level. Possible values: from 0 to 100.
clickSound	src	undefined	The path to an external mp3 file loaded into a movie clip and attached to the menu item's click event. If no path is specified, the click sound will not be used. If the path is incorrect, the click sound will not be used.
	volume	100	A number representing a volume level. Possible values: from 0 to 100.
initialStateReturn	enabled	"true"	Instructs Accordion Tree Menu how to behave when the mouse pointer rolls out of an open top level item outside the menu area. Possible values: "true", "false". a) "false" — The open top level item remains open. b) "true" — The menu returns to its initial state. If selectedIndexMode is set to "true" and the selectedIndex property has a definite value (see selectedIndexMode for more information), then the top level item including nested item, which index is equal to the selectedIndex value, will be opened. If there is no item selected, the open top level item will be closed.
selectedStateMode	enabled	"true"	Turns on/off the component's ability to make an item selected if the selectedIndex property has a definite value. Possible values: "true", "false". <i>Note:</i> The selectedIndex property has a definite (not undefined) value in the following three cases: 1) The variable selIndex is passed to a SWF (containing Accordion Tree Menu component instance) through HTML tags (see example 1 for more information). 2) The internalSelectionMode parameter is set to "true" and a menu item has been clicked. 3) The setItemSelected() method has been called.

internalSelectionMode	enabled	"false"	Turns on/off the component's ability to make an item selected when it is clicked. That means Accordion Tree Menu can change the value of selectedIndex property inside a movie clip without reloading HTML page. This option is useful, first of all, for Flash-based web applications. Possible values: "true", "false".
-----------------------	---------	---------	---

3. The **<style>** element determines the appearance of Accordion Tree Menu component. The following table describes the attributes of nodes that the **<style>** element contains:

Node name	Attribute name	Default	Description
mainBackground	color	"FFFFFF"	The main background color of the menu. If omitted, the background is transparent.
	alpha	100	The main background alpha of the menu. Possible values: from 0 to 100.
decorationBackground	color	"FFFFFF"	The background color of the top and bottom decorations. If omitted, the background is transparent.
	alpha	100	The background alpha of the top and bottom decorations. Possible values: from 0 to 100.
topLevelBackground	color	"FFFFFF"	The background color of the top level menu items. If omitted, the background is transparent.
	alpha	100	The background alpha of the top level menu items. Possible values: from 0 to 100.
treelcon	color	–	The color of the tree icons. If omitted, the color is the same as the text color of a top level item.
	bgcolor	–	The background color of the tree icons. If omitted, the background color is the same as the main background color of the menu.
dottedLine	color	–	The color of the dotted lines. If omitted, the color is the same as the text color of a top level item.
separatorLine	color1	undefined	The color of the top separator line. If omitted, the line will not be drawn.
	color2	undefined	The color of the bottom separator line. If omitted, the line will not be drawn.

rolloverBackgroundColor	level1	"FFFFFF"	The background color of a rolled-over top level item. If omitted, the background is transparent.
	level2	"FFFFFF"	The background color of a rolled-over second or third level item. If omitted, the background is transparent.
rolloverBackgroundAlpha	level1	100	The background alpha of a rolled-over top level item. Possible values: from 0 to 100.
	level2	100	The background alpha of a rolled-over second or third level item. Possible values: from 0 to 100.
selectedBackgroundColor	level2	"FFFFFF"	The background color of a selected second or third level item. If omitted, the background is transparent.
selectedBackgroundAlpha	level2	100	The background alpha of a selected second or third level item. Possible values: from 0 to 100.
fontSize	level1	14	The point size for the font of a top level item text. Possible values: from 9 to 30.
	level2	12	The point size for the font of a second level item text. Possible values: from 9 to 22.
	level3	11	The point size for the font of a third level item text. Possible values: from 9 to 22.
fontFamily	name	"_sans"	The font name for text.
embedFonts	enabled	"false"	A Boolean value that indicates whether the font specified in fontFamily is an embedded font. This parameter must be set to "true" if fontFamily refers to an embedded font. Otherwise, the embedded font is not used. If this parameter is set to true and fontFamily does not refer to an embedded font, no text is displayed. Possible values: "true", "false".
fontColor	level1	"000033"	The text color of a top level item.
	level2	"000033"	The text color of a second level item.
	level3	"000033"	The text color of a third level item.
fontColorRollover	level1	"CC3300"	The color of text when the pointer rolls over a top level item.
	level2	"CC3300"	The color of text when the pointer rolls over a second level item.
	level3	"CC3300"	The color of text when the pointer rolls over a third level item.

fontColorSelected	level2	"CC3300"	The color of text in the selected second level item.
	level3	"CC3300"	The color of text in the selected third level item.
fontWeight	level1	"none"	The font weight for the text of a top level item. Possible values: "none", "bold".
	level2	"none"	The font weight for the text of a second level item. Possible values: "none", "bold".
	level3	"none"	The font weight for the text of a third level item. Possible values: "none", "bold".
fontWeightRollover	level1	"none"	The font weight for the text when the pointer rolls over a top level item. Possible values: "none", "bold".
	level2	"none"	The font weight for the text when the pointer rolls over a second level item. Possible values: "none", "bold".
	level3	"none"	The font weight for the text when the pointer rolls over a third level item. Possible values: "none", "bold".
fontWeightSelected	level2	"none"	The font weight for the text in the selected second level item. Possible values: "none", "bold".
	level3	"none"	The font weight for the text in the selected third level item. Possible values: "none", "bold".
fontStyle	level1	"normal"	The font style for the text of a top level item. Possible values: "normal", "italic".
	level2	"normal"	The font style for the text of a second level item. Possible values: "none", "bold".
	level3	"normal"	The font style for the text of a third level item. Possible values: "none", "bold".
textDecorationRollover	level1	"none"	The text decoration when the pointer rolls over a top level item. Possible values: "none", "underline".
	level2	"none"	The text decoration when the pointer rolls over a second level item. Possible values: "none", "underline".
	level3	"none"	The text decoration when the pointer rolls over a third level item. Possible values: "none", "underline".

Note: All colors should be expressed in RRGGBB format.

4. The **<items>** element can contain an unlimited number of item nodes. Each **<item>** node represents a menu item object and should have one required attribute label. You can also add four optional attributes in the **<item>** node: action, url, target, font_color.

Node name	Attribute name	Default	Description
item	label	undefined	The text that is displayed to represent a menu item.
	action	undefined	(Optional) This attribute can be either "getUrl" or the name of your custom Flash function (e.g., "showMap"). In the first case, Flash getURL() method is called — it loads a document from the specified URL into the specified window. In the second case, your custom Flash function is called.
	url	undefined	(Optional) This attribute defines either the URL from which to obtain the document (if action attribute is set to "getUrl") or any parameters to be passed to your function (if action attribute is set to the name of your custom Flash function). You can specify zero or more parameters, separating them by commas.
	target	"_self"	(Optional) This attribute can be either a parameter ("_self", "_blank", "_parent", "_top" or your custom value) that specifies the window or HTML frame that the document is loaded into (if action attribute is set to "getUrl") or a target movie clip where your function is placed (if action attribute is set to the name of your custom Flash function).
	font_color	undefined	(Optional) The text color of a single item (overrides the color setting for appropriate level in the style element).

Creating an application with Accordion Tree Menu component

The following procedure explains how to create an application with Accordion Tree Menu component.

Example 1

In this example, Accordion Tree Menu is used to create the navigation system for HTML-based web application.

To create an application with the Accordion Tree Menu component:

1. Open the source FLA file (AccordionTreeMenu.fla) and set the Stage size to 120 x 580 pixels. Specify "#89A7B1" as background color. For frame rate enter the value not less than 30 fps.
2. Set the coordinates for the Accordion Tree Menu component instances to (0, 0).
3. Publish the Flash document with appropriate name — for example, "menu.swf" (note that the ActionScript file — AccordionTreeMenu.as — must be in the same directory with the source FLA file).
4. Using your favorite text editor, create a new document and enter the following code:

```
<?xml version="1.0" encoding="UTF-8"?>
<menu width="">
  <properties>
    <treeStructure state="close"/>
    <itemHeight level1="20" level2="20"/>
    <topLevelIcons enabled="false"/>
    <dottedLine level2="true" level3="false"/>
    <decoration height="2" rounded="none"/>
  </properties>
</menu>
```

```

<leftMargin value="10"/>
<itemIndent level2="true" level3="false"/>
<itemSeparators level1="true" level2="false"/>
<tween duration="" easing="" method=""/>
<rolloverSound src="" volume=""/>
<clickSound src="" volume=""/>
<initialStateReturn enabled="true"/>
<selectedStateMode enabled="true"/>
<internalSelectionMode enabled="false"/>
</properties>
<style>
<mainBackground color="E8EEF0" alpha=""/>
<decorationBackground color="00202B" alpha=""/>
<topLevelBackground color="C5D4D9" alpha=""/>
<treeIcon color="" bgcolor=""/>
<dottedLine color=""/>
<separatorLine color1="004B64" color2=""/>
<rolloverBackgroundColor level1="C5D4D9" level2="C5D4D9"/>
<rolloverBackgroundAlpha level1="" level2="30"/>
<selectedBackgroundColor level2=""/>
<selectedBackgroundAlpha level2=""/>
<fontSize level1="13" level2="12" level3="11"/>
<fontFamily name="Verdana"/>
<embedFonts enabled="false"/>
<fontColor level1="00202B" level2="00202B" level3="004B64"/>
<fontColorRollover level1="00202B" level2="00202B" level3="004B64"/>
<fontColorSelected level2="FF3300" level3="FF3300"/>
<fontWeight level1="" level2="" level3=""/>
<fontWeightRollover level1="" level2="" level3=""/>
<fontWeightSelected level2="" level3=""/>
<fontStyle level1="" level2="" level3=""/>
<textDecorationRollover level1="" level2="" level3=""/>
</style>
<items>
<item label="Austria" action="getUrl" url="destinations.php?index=0" target="">
<item label="Vienna" action="getUrl" url="destinations.php?index=1" target="">
<item label="Overview" action="getUrl" url="destinations.php?index=2" target=""/>
<item label="Places to See" action="getUrl" url="destinations.php?index=3" target=""/>
<item label="Money" action="getUrl" url="destinations.php?index=4" target=""/>
<item label="Transport" action="getUrl" url="destinations.php?index=5" target=""/>
<item label="Culture" action="getUrl" url="destinations.php?index=6" target=""/>
</item>
.....
</item>
.....
</items>
</menu>

```

Note: In our case, the url attribute of item nodes refers to the same PHP-file — destinations.php. The index variable passed with GET method is used to make the necessary item of Accordion Tree Menu selected.

This is your configuration XML document. Save the XML file in preferred location (for example, in the same directory as the SWF file that contains an Accordion Tree Menu component) and name it, for example, "menu.xml".

5. Create a PHP file and place the following HTML code into it — to embed SWF file (in our case, menu.swf):

```

<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.adobe.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0"
width="120" height="580" id="menu" align="middle">
  <param name="allowScriptAccess" value="sameDomain"/>
  <param name="movie" value="menu.swf"/>
  <param name="FlashVars" value="xmlURL=menu.xml&selIndex=<?=$index;?>"/>
  <param name="quality" value="high"/>
  <param name="bgcolor" value="#89A7B1"/>
  <embed src="menu.swf" FlashVars="xmlURL=menu.xml&selIndex=<?=$index;?>" quality="high"
bgcolor="#89A7B1" width="120" height="580" name="menu" align="middle"
allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
pluginspage="http://www.adobe.com/go/getflashplayer"/>
</object>

```

Note: Accordion Tree Menu can accept two external variables: 1) xmlURL — the path to configuration XML file, 2) selIndex — the zero-based index of a menu item to be selected. In our case, selIndex is equal to the value of index variable passed with GET method in url.

6. Save the PHP file in preferred location (for example, in the same directory as the SWF file that contains an Accordion Tree Menu component) and name it "destinations.php".
7. Open destinations.php in a web browser and test the behavior of the Accordion Tree Menu component.
8. You can view the completed application [here](#).

Example 2

In this example, Accordion Tree Menu is used to create the navigation system for Flash-based web application.

Before creating an application, download the images required for this example from the following URLs and save them to your hard disk:

http://www.e-merald.com/pic/components/car_bg.jpg

<http://www.e-merald.com/pic/components/beep.mp3>

To create an application with the Accordion Tree Menu component:

1. Open the source FLA file (AccordionTreeMenu.fla) and set the Stage size to 520 x 500 pixels. Specify "#000000" as background color. For frame rate enter the value not less than 30 fps.
2. Import the saved image file (car_bg.jpg) to the Stage by selecting File > Import > Import to Stage. Set the coordinates for the imported image to (0, 0).
3. Set the coordinates for the Accordion Tree Menu component instances to (20, 130). In the Property inspector, set the Instance Name value for the component instance to "atm".
4. Using the Text tool, create a dynamic text field on the Stage. With the text field selected, in the Property inspector, set the Variable value to "page_title". Create two more dynamic text fields and set the Variable value for them accordingly to "section_title" and "page_text". Set appropriate values for other text field properties.
5. Open the Actions panel, select Frame 1 in the main Timeline, and enter the following ActionScript code:

```

function xmlOnLoadEvent () {
    page_title = "Overview";
    page_text = "Description for this section...";
}

```

```

function clickEvent(index:Number) {
    if (index == 0) atm.setItemSelected(1);
    if (index == 5) atm.setItemSelected(6);
    if (index == 11) atm.setItemSelected(12);
    if (index == 14) atm.setItemSelected(15);
    section_title = atm.getItemAt(atm.selectedIndex).label;
    page_text = "Description for this section...";
}
function pageTitle(topLevelIndex:Number):Void {
    page_title = atm.getItemAt(topLevelIndex).label.substr(0,1) +
    atm.getItemAt(topLevelIndex).label.substr(1).toLowerCase();
}

```

6. Publish the Flash document with appropriate name — for example, "menu.swf" (note that the ActionScript file — AccordionTreeMenu.as — must be in the same directory with the source FLA file).
7. Using your favorite text editor, create a new document and enter the following code:

```

<?xml version="1.0" encoding="UTF-8"?>
<menu width="">
  <properties>
    <treeStructure state="static"/>
    <itemHeight level1="22" level2="18"/>
    <topLevelIcons enabled="true"/>
    <dottedLine level2="false" level3="false"/>
    <decoration height="3" rounded="none"/>
    <leftMargin value=""/>
    <itemIndent level2="false" level3=""/>
    <itemSeparators level1="false" level2="false"/>
    <tween duration="" easing="" method=""/>
    <rolloverSound src="" volume=""/>
    <clickSound src="beep.mp3" volume="70"/>
    <initialStateReturn enabled="true"/>
    <selectedStateMode enabled="true"/>
    <internalSelectionMode enabled="true"/>
  </properties>
  <style>
    <mainBackground color="" alpha=""/>
    <decorationBackground color="FFFFFF" alpha="30"/>
    <topLevelBackground color="FFFFFF" alpha="15"/>
    <treeIcon color="" bgcolor=""/>
    <dottedLine color=""/>
    <separatorLine color1="FFFFFF" color2=""/>
    <rolloverBackgroundColor level1="FFFFFF" level2=""/>
    <rolloverBackgroundAlpha level1="15" level2=""/>
    <selectedBackgroundColor level2=""/>
    <selectedBackgroundAlpha level2=""/>
    <fontSize level1="11" level2="10" level3=""/>
    <fontFamily name="Tahoma"/>
    <embedFonts enabled="false"/>
    <fontColor level1="CCCCCC" level2="CCCCCC" level3=""/>
    <fontColorRollover level1="CCCCCC" level2="FFB13E" level3=""/>
    <fontColorSelected level2="FFB13E" level3=""/>
    <fontWeight level1="bold" level2="" level3=""/>
    <fontWeightRollover level1="bold" level2="" level3=""/>
    <fontWeightSelected level2="" level3=""/>
    <fontStyle level1="" level2="" level3=""/>
    <textDecorationRollover level1="" level2="" level3=""/>
  </style>

```

```

<items>
  <item label="FEATURES" action="pageTitle" url="0" target="" font_color="">
    <item label="Exterior" action="pageTitle" url="0" target="" font_color=""/>
    <item label="Interior" action="pageTitle" url="0" target="" font_color=""/>
    <item label="Safety" action="pageTitle" url="0" target="" font_color=""/>
    <item label="Price" action="pageTitle" url="0" target="" font_color=""/>
  </item>
  <item label="SPECS" action="pageTitle" url="5" target="" font_color="">
    <item label="Mechanical" action="pageTitle" url="5" target="" font_color=""/>
    <item label="Dimensions" action="pageTitle" url="5" target="" font_color=""/>
    <item label="Capacities" action="pageTitle" url="5" target="" font_color=""/>
    <item label="Tires" action="pageTitle" url="5" target="" font_color=""/>
    <item label="MPG" action="pageTitle" url="5" target="" font_color=""/>
  </item>
  <item label="OPTIONS" action="pageTitle" url="11" target="" font_color="">
    <item label="Available options" action="pageTitle" url="11" target="" font_color=""/>
    <item label="Packages" action="pageTitle" url="11" target="" font_color=""/>
  </item>
  <item label="ACCESSORIES" action="pageTitle" url="14" target="" font_color="">
    <item label="Exterior" action="pageTitle" url="14" target="" font_color=""/>
    <item label="Interior" action="pageTitle" url="14" target="" font_color=""/>
  </item>
</items>
</menu>

```

Note: In our case, the action attribute of each item node is equal to the name of Flash function to be called on menu item's click event. The url attribute of each item node is equal to the index of corresponding top level menu item. Thus, the value of url is a parameter for the pageTitle() function (see the ActionScript code above).

This is your configuration XML document. Save the XML file in preferred location (for example, in the same directory as the SWF file that contains an Accordion Tree Menu component) and name it, in our case, "menu.xml".

8. Open the Flash document (menu.swf).
9. You can view the completed application [here](#).

AccordionTreeMenu class

Inheritance MovieClip > AccordionTreeMenu class

Methods unique to the AccordionTreeMenu class

The following table lists methods of the AccordionTreeMenu class.

Method	Description
AccordionTreeMenu.getItemAt()	Gets a reference to a menu item at a specified location.
AccordionTreeMenu.setItemSelected()	Determines the index position of the menu item to be selected and changes the current selection.

Properties unique to the AccordionTreeMenu class

The following table lists properties of the AccordionTreeMenu class.

Property	Description
AccordionTreeMenu.decorationHeight	The height of the top and bottom decorations, in pixels.
AccordionTreeMenu.decorationRounded	Specifies which corners of the menu decoration to be drawn rounded.
AccordionTreeMenu.dottedLineL2	Indicates whether the dotted lines for second level sub-menus to be drawn.
AccordionTreeMenu.dottedLineL3	Indicates whether the dotted lines for third level sub-menus to be drawn.
AccordionTreeMenu.easingClass	Specifies which easing class to be applied when opening or closing a second level menu item.
AccordionTreeMenu.easingMethod	Specifies which easing method to be applied when opening or closing a second level menu item.
AccordionTreeMenu.initialStateReturn	Instructs the menu how to behave when the mouse pointer rolls out of an open top level item outside the menu area.
AccordionTreeMenu.internalSelectionMode	Turns on/off the component's ability to make an item selected when it is clicked. That means the Accordion Tree Menu can change the value of selectedIndex property inside a movie clip without reloading HTML page.
AccordionTreeMenu.itemHeightL1	The height of each top level item, in pixels.
AccordionTreeMenu.itemHeightL2	The height of each second and third level item, in pixels.
AccordionTreeMenu.itemIndentL2	Indicates whether to indent the second level items.
AccordionTreeMenu.itemIndentL3	Indicates whether to indent the third level items.
AccordionTreeMenu.itemSeparatorsL1	Determines whether to draw separator lines between the top level menu items.
AccordionTreeMenu.itemSeparatorsL2	Determines whether to draw separator lines between the second level menu items and between the third level menu items.

AccordionTreeMenu.leftMargin	The left margin of the menu, in pixels.
AccordionTreeMenu.menuWidth	The width of the menu, in pixels.
AccordionTreeMenu.selectedIndex	Read-only; the index position of the selected menu item.
AccordionTreeMenu.selectedStateMode	Turns on/off the component's ability to make an item selected.
AccordionTreeMenu.topLevelIcons	Determines whether to draw the arrow icons for the top level menu items.
AccordionTreeMenu.treeStructure	Specifies the appearance and initial state of Accordion Tree Menu structure.
AccordionTreeMenu.tweeningDuration	A number indicating the duration of the tweened animation in seconds. The tweened animation is applied to each third level sub-menu during its movement up and down.

Events unique to the AccordionTreeMenu class

There are no events unique to the AccordionTreeMenu class. However, you can use the functions `xmlOnLoadEvent` and `clickEvent` (called from `AccordionTreeMenu.as`) as analogs of `onLoadXML` and `click` listeners, respectively.

AccordionTreeMenu.decorationHeight

Usage

```
my_menu.decorationHeight
```

Description

Property; the height, in pixels, of the top and bottom decorations. Possible values: from 0 to the value of a top level item height. If `decorationHeight` is equal to 0, then no decoration will be drawn. The default value is 0.

Example

The following example sets the `decorationHeight` property to 5 pixels:

```
my_menu.decorationHeight = 5;
```

AccordionTreeMenu.decorationRounded

Usage

```
my_menu.decorationRounded
```

Description

Property; a string that specifies which corners of the menu decoration to be drawn rounded. This property can be one of four predefined values: "left", "right", "all", "none". The default value is "all".

Example

The following example sets the `decorationRounded` property to "right":

```
my_menu.decorationRounded = "right";
```

AccordionTreeMenu.dottedLineL2

Usage

```
my_menu.dottedLineL2
```

Description

Property; indicates whether the dotted lines for second level sub-menus to be drawn ("true") or not ("false"). If dottedLineL2 is set to "false", then no tree icons are drawn, and each second level menu item (if it has nested third level sub-menu) stays open permanently. The default value is "true".

Example

The following example sets the dottedLineL2 property to "false":

```
my_menu.dottedLineL2 = false;
```

AccordionTreeMenu.dottedLineL3

Usage

```
my_menu.dottedLineL3
```

Description

Property; indicates whether the dotted lines for third level sub-menus to be drawn ("true") or not ("false"). The default value is "true".

Example

The following example sets the dottedLineL3 property to "false":

```
my_menu.dottedLineL3 = false;
```

AccordionTreeMenu.easingClass

Usage

```
my_menu.easingClass
```

Description

Property; a string that specifies which easing class to be applied when opening or closing a second level menu item. This property can be either "regular" or "none". The default value is "regular".

Example

The following example sets the easingClass property to "none":

```
my_menu.easingClass = "none";
```

AccordionTreeMenu.easingMethod

Usage

```
my_menu.easingMethod
```

Description

Property; a string that specifies which easing method to be applied when opening or closing a second level menu item. This property can be one of three predefined values: "easeIn", "easeOut" or "easeNone"; the default value is "easeOut".

Example

The following example sets the `easingMethod` property to "easeIn":

```
my_menu.easingMethod = "easeIn";
```

AccordionTreeMenu.getItemAt()

Usage

```
my_menu.getItemAt(index)
```

Parameters

index An integer indicating the index of the node in the menu. This is a zero-based index, so 0 retrieves the first item, 1 retrieves the second item, and so on.

Returns

A reference to the specified node.

Description

Method; returns a reference to the specified child node of the menu.

Example

The following example demonstrates how to retrieve the values of attributes available for an item node:

```
my_menu.getItemAt(0).label;  
my_menu.getItemAt(0).action;  
my_menu.getItemAt(0).url;  
my_menu.getItemAt(0).target;  
my_menu.getItemAt(0).fontColor;
```

AccordionTreeMenu.initialStateReturn

Usage

```
my_menu.initialStateReturn
```

Description

Property; instructs the menu how to behave when the mouse pointer rolls out of an open top level item outside the menu area. There are two values available:

true The open top level item remains open.

false The menu returns to its initial state. If `selectedStateMode` is set to "true" and the `selectedIndex` property has a definite value, then the top level item including nested item, which index is equal to the `selectedIndex` value, will be opened. If there is no item selected, the open top level item will be closed.

The default value is "true".

Example

The following example sets the `initialStateReturn` property to "false":

```
my_menu.initialStateReturn = false;
```

AccordionTreeMenu.internalSelectionMode

Usage

```
my_menu.internalSelectionMode
```

Description

Property; turns on/off the component's ability ("true" or "false") to make an item selected when it is clicked. That means the Accordion Tree Menu can change the value of the selectedIndex property inside a movie clip without reloading HTML page. This option is useful, first of all, for Flash-based web applications. For this property to be activated, selectedIndexMode has to be set to "true". The default value is "false".

Example

The following example sets the internalSelectionMode property to "true":

```
my_menu.internalSelectionMode = true;
```

AccordionTreeMenu.itemHeightL1

Usage

```
my_menu.itemHeightL1
```

Description

Property; the height, in pixels, of each top level item. The value of this property is controlled programmatically, considering specified font size and font family. The default value is 24.

Example

The following example sets the itemHeightL1 property to 30:

```
my_menu.itemHeightL1 = 30;
```

AccordionTreeMenu.itemHeightL2

Usage

```
my_menu.itemHeightL2
```

Description

Property; the height, in pixels, of each second and third level item. The value of this property is controlled programmatically, considering specified font size and font family. The default value is 20.

Example

The following example sets the itemHeightL2 property to 24:

```
my_menu.itemHeightL2 = 24;
```

AccordionTreeMenu.itemIndentL2

Usage

```
my_menu.itemIndentL2
```

Description

Property; indicates whether to indent the second level items. The amount of indent is determined programmatically. The default value is "true". If both itemIndentL2 and topLevelIcons properties are set to "false", then neither tree icons nor dotted lines for second level sub-menus are drawn, and each second level menu item (if it has nested third level sub-menu) stays open permanently.

Example

The following example sets the itemIndentL2 property to "false":

```
my_menu.itemIndentL2 = false;
```

AccordionTreeMenu.itemIndentL3

Usage

```
my_menu.itemIndentL3
```

Description

Property; indicates whether to indent the third level items. The amount of indent is determined programmatically. The default value is "true". If itemIndentL3 is set to "false", then no dotted lines for third level sub-menus are drawn.

Example

The following example sets the itemIndentL3 property to "false":

```
my_menu.itemIndentL3 = false;
```

AccordionTreeMenu.itemSeparatorsL1

Usage

```
my_menu.itemSeparatorsL1
```

Description

Property; determines whether to draw separator lines between the top level menu items. The default value is "true".

Example

The following example sets the itemSeparatorsL1 property to "false":

```
my_menu.itemSeparatorsL1 = false;
```

AccordionTreeMenu.itemSeparatorsL2

Usage

```
my_menu.itemSeparatorsL2
```

Description

Property; determines whether to draw separator lines between the second level menu items and between the third level menu items. The default value is "false".

Example

The following example sets the itemSeparatorsL2 property to "true":

```
my_menu.itemSeparatorsL2 = true;
```

AccordionTreeMenu.leftMargin

Usage

```
my_menu.leftMargin
```

Description

Property; the left margin of the menu, in pixels. Accepts values from 0 to 100. The default value is 0.

Example

The following example sets the leftMargin property to 20:

```
my_menu.leftMargin = 20;
```

AccordionTreeMenu.menuWidth

Usage

```
my_menu.menuWidth
```

Description

Property; the width of the menu, in pixels. The default value is 100.

Example

The following example sets the menuWidth property to 200:

```
my_menu.menuWidth = 200;
```

AccordionTreeMenu.selectedIndex

Usage

```
my_menu.selectedIndex
```

Description

Property (read-only); the index position of the selected menu item. The value is undefined if nothing is selected. The selectedIndex property has a definite (not undefined) value in the following three cases:

1. The variable selIndex is passed to a SWF (containing Accordion Tree Menu component instance) through HTML tags (see example 1 for more information). If the component instance is not at the top level of a movie, you must specify the selIndex variable at the component's level.
2. The internalSelectionMode property is equal to "true" and a menu item has been clicked.
3. The setItemSelected() method has been called.

Example

The following example gets the value of the selectedIndex property:

```
trace(my_menu.selectedIndex);
```

AccordionTreeMenu.selectedStateMode

Usage

```
my_menu.selectedStateMode
```

Description

Property; turns on/off the component's ability ("true" or "false") to make an item selected if the selectedIndex property has a definite value. The default value is "true".

Example

The following example sets the selectedStateMode property to "false":

```
my_menu.selectedStateMode = false;
```

AccordionTreeMenu.setItemSelected()

Usage

```
my_menu.setItemSelected(index)
```

Parameters

index An integer indicating the zero-based index of the menu item to be selected. If this parameter is omitted, no item will be selected and the selectedIndex property will be set to the undefined value.

Returns

Nothing.

Description

Method; determines the index position of the menu item to be selected and changes the current selection. If you call `setItemSelected()` method, any current selection is cleared and the indicated item is selected.

Example

The following example sets the index of the menu item to be selected to 10:

```
my_menu.setItemSelected(10);
```

AccordionTreeMenu.topLevelIcons

Usage

```
my_menu.topLevelIcons
```

Description

Property; determines whether to draw the arrow icons for the top level menu items. If both `topLevelIcons` and `itemIndentL2` properties are set to "false", then neither tree icons nor dotted lines for second level sub-menus are drawn, and each second level menu item (if it has nested third level sub-menu) stays open permanently. The default value is "true".

Example

The following example sets the `topLevelIcons` property to "false":

```
my_menu.topLevelIcons = false;
```

AccordionTreeMenu.treeStructure

Usage

```
my_menu.treeStructure
```

Description

Property; specifies the appearance and initial state of Accordion Tree Menu structure. This parameter can be one of three predefined values:

open Each second level menu item that has nested third level sub-menu is in open state, and can be closed by clicking on appropriate tree icon.

close Each second level menu item that has nested third level sub-menu is in closed state, and can be opened by clicking on appropriate tree icon.

static Each second level menu item that has nested third level sub-menu stays open permanently. No tree icons are drawn.

The component's structure appearance is also depends on values combination of the following properties: `topLevelIcons`, `itemIndentL2` and `dottedLineL2`. The default value is "close".

Example

The following example sets the `treeStructure` property to "static":

```
my_menu.treeStructure = "static";
```

AccordionTreeMenu.tweeningDuration

Usage

```
my_menu.tweeningDuration
```

Description

Property; a number indicating the duration of the tweened animation in seconds. The tweened animation is applied to each third level sub-menu during its movement up and down. This parameter accepts values from 0.05 to 0.5. The default value is 0.3.

Example

The following example sets the tweeningDuration property to 0.5:

```
my_menu.tweeningDuration = 0.5;
```

xmlOnLoadEvent()

Usage

```
function xmlOnLoadEvent() {  
    your code  
}
```

Returns

Nothing.

Description

Function; is called from the AccordionTreeMenu.as class when the XML file is loaded and processed. This function is an analog of onLoadXML listener.

Example

The following example, written on a frame of the Timeline, uses the xmlOnLoadEvent function to output a message:

```
function xmlOnLoadEvent() {  
    trace("The XML file is loaded");  
}
```

clickEvent ()

Usage

```
function clickEvent(index) {  
    your code  
}
```

Parameters

index An integer indicating the zero-based index of the clicked menu item.

Returns

Nothing.

Description

Function; is called from the AccordionTreeMenu.as class when the mouse is clicked (pressed) over a menu item. This function is an analog of click listener.

Example

The following example, written on a frame of the Timeline, uses the `clickEvent` function to output a message:

```
function clickEvent(index:Number) {  
    trace("The index of the menu item that has been clicked is " + index);  
}
```