Accessible Drop-Down Menu

# Date last modified:

5/8/2014

# URL:

<http://www.csun.edu/udc/alen/drop-down/dev1/menu.html>

# Summary:

This document describes the technical and analytical aspects of creating accessible Drop-down menu. It introduces a coding methodology that is used to overcome the challenges in developing an accessible Drop-down menu that includes links as main menu items.

# Proper Keyboard Navigation

## Purpose:

The purpose of this technique is to provide proper keyboard functionality to navigate through the drop down menu. The most common navigational strategy for drop down navigation includes:

* Tab key: to navigate through the main menu item
* Up + down arrow: to navigate through the sub-menu items
* Spacebar: to expand the drop down menu and activate the links of the sub-menu items
* Enter key: to activate the links of the main menu items
* Esc key: to exit (collapse) the drop down

## Technique:

This technique uses jQuery to handle the functionality of the keys. It is used to modify the default behavior of the spacebar key, arrow key, ESC key and Enter key. For example, this technique modifies the Enter key functionality to activate the main menu items without expanding the drop down menu. The same way all other navigational keys’ functionalities were modified to implement the proper behaviors. Therefore it works consistently on all common browsers (Firefox, Internet Explorer and Chrome).

## Source Code

### jQuery:

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>

<script>

$(document).ready(function() {

 var menu1 = new menubar('nav', false);

}); // end ready()

// Modified and Developed by: Universal Design Center - California State University, Northridge

// Date Developed: 04/22/2014 to 04/24/2014

// This code is modified and customized to comply with the best navigational practices

/////////////// begin menu widget definition /////////////////////

//

// Function menubar() is the constructor of a menu widget

// The widget will bind to the ul passed to it.

//

// @param(id string) id is the HTML id of the ul to bind to

//

// @param(vmenu boolean) vmenu is true if menu is vertical; false if horizontal

//

// @return N/A

//

function menubar(id, vmenu) {

 // define widget properties

 this.$id = $('#' + id);

 this.$rootItems = this.$id.children('li'); // jQuerry array of all root-level menu items

 this.$items = this.$id.find('.menu-item').not('.separator'); // jQuery array of menu items

 this.$parents = this.$id.find('.menu-parent'); // jQuery array of menu items

 this.$allItems = this.$parents.add(this.$items); // jQuery array of all menu items

 console.log('All items:');

 console.log(this.$allItems);

 console.log('Parents: ');

 console.log(this.$parents);

 this.$activeItem = null; // jQuery object of the menu item with focus

 this.vmenu = vmenu;

 this.bChildOpen = false; // true if child menu is open

 this.keys = {

 tab: 9,

 enter: 13,

 esc: 27,

 space: 32,

 up: 38,

 down: 40,

 F5: 116

 };

 // bind event handlers

 this.bindHandlers();

};

//

// Function bindHandlers() is a member function to bind event handlers for the widget.

//

// @return N/A

//

menubar.prototype.bindHandlers = function() {

 var thisObj = this;

 //////////// bind key event handlers //////////////////

 // bind a keydown handler

 this.$allItems.keydown(function(e) {

 return thisObj.handleKeyDown($(this), e);

 });

 // bind a keypress handler

 this.$allItems.keypress(function(e) {

 return thisObj.handleKeyPress($(this), e);

 });

 // bind a focus handler

 this.$allItems.focus(function(e) {

 return thisObj.handleFocus($(this), e);

 });

 // bind a blur handler

 this.$allItems.blur(function(e) {

 return thisObj.handleBlur($(this), e);

 });

 // bind a document click handler

 $(document).click(function(e) {

 return thisObj.handleDocumentClick(e);

 });

} // end bindHandlers()

// Function handleFocus() is a member function to process focus events

// for the menu.

//

// @param($item object) $item is the jquery object of the item firing the event

//

// @param(e object) e is the associated event object

//

// @return(boolean) Returns true;

//

menubar.prototype.handleFocus = function($item, e) {

 // if activeItem is null, we are getting focus from outside the menu. Store

 // the item that triggered the event

 if (this.$activeItem == null) {

 this.$activeItem = $item;

 }

 else if ($item[0] != this.$activeItem[0]) {

 return true;

 }

 // get the set of jquery objects for all the parent items of the active item

 var $parentItems = this.$activeItem.parentsUntil('div').filter('li');

 // remove focus styling from all other menu items

 this.$allItems.removeClass('menu-focus');

 // add styling to the active item

 this.$activeItem.addClass('menu-focus');

 // add styling to all parent items.

 $parentItems.addClass('menu-focus');

 if (this.vmenu == true) {

 // if the bChildOpen is true, open the active item's child menu (if applicable)

 if (this.bChildOpen == true) {

 var $itemUL = $item.parent();

 // if the itemUL is a root-level menu and item is a parent item,

 // show the child menu.

 if ($itemUL.is('.root-level') && ($item.attr('aria-haspopup') == 'true')) {

 $item.children('ul').show().attr('aria-hidden', 'false');

 }

 }

 else {

 this.vmenu = false;

 }

 }

 return true;

} // end handleFocus()

//

// Function handleBlur() is a member function to process blur events

// for the menu.

//

// @param($item object) $item is the jquery object of the item firing the event

//

// @param(e object) e is the associated event object

//

// @return(boolean) Returns true;

//

menubar.prototype.handleBlur = function($item, e) {

 // $item.find('a').removeClass('menu-focus');

 $item.removeClass('menu-focus');

 return true;

} // end handleBlur()

//

// Function handleKeyDown() is a member function to process keydown events

// for the menus.

//

// @param($item object) $item is the jquery object of the item firing the event

//

// @param(e object) e is the associated event object

//

// @return(boolean) Returns false if consuming; true if propagating

//

menubar.prototype.handleKeyDown = function($item, e) {

 if (e.altKey || e.ctrlKey) {

 // Modifier key pressed: Do not process

 return true;

 }

 switch(e.keyCode) {

 **case this.keys.tab:** {

 // hide all menu items and update their aria attributes

 this.$id.find('ul').hide().attr('aria-hidden', 'true');

 // remove focus styling from all menu items

 this.$allItems.removeClass('menu-focus');

 this.$activeItem = null;

 this.bChildOpen == false;

 return true;

 break;

 }

 **case this.keys.esc:** {

 var $itemUL = $item.parent();

 if ($itemUL.is('.root-level')) {

 // hide the child menu and update the aria attributes

 $item.children('ul').first().hide().attr('aria-hidden', 'true');

 }

 else {

 // move up one level

 this.$activeItem = $itemUL.parent();

 // reset the childOpen flag

 this.bChildOpen = false;

 // set focus on the new item

 this.$activeItem.focus();

 // hide the active menu and update the aria attributes

 $itemUL.hide().attr('aria-hidden', 'true');

 }

 e.stopPropagation();

 return false;

 }

 **case this.keys.enter:** {

 window.location = $item.find('a').attr('href');

 return false;

 }

 **case this.keys.space:** {

 if (!($item.hasClass('menu-parent'))) {

 // user pressed enter or space on a dropdown menu item,

 // not an item on the menu bar

 // get the target href and go there

 window.location = $item.find('a').attr('href');

 return false;

 }

 var $parentUL = $item.parent();

 if ($parentUL.is('.root-level')) {

 // open the child menu if it is closed

 $item.children('ul').first().show().attr('aria-hidden', 'false');

 this.bChildOpen = true;

 }

 else {

 // remove hover styling

 this.$allItems.removeClass('menu-hover');

 this.$allItems.removeClass('menu-focus');

 // close the menu

 this.$id.find('ul').not('.root-level').hide().attr('aria-hidden','true');

 // clear the active item

 this.$activeItem = null;

 }

 e.stopPropagation();

 return false;

 }

 **case this.keys.up:** {

 if (!(this.vmenu == true && $itemUL.is('.root-level'))) {

 this.$activeItem = this.moveUp($item);

 }

 this.$activeItem.focus();

 e.stopPropagation();

 return false;

 }

 **case this.keys.down:** {

 if (!(this.vmenu == true && $itemUL.is('.root-level'))) {

 this.$activeItem = this.moveDown($item);

 }

 this.$activeItem.focus();

 e.stopPropagation();

 return false;

 }

 //prevent open dropdown when pressing F5 to reload the page

 **case this.keys.F5:** {

 location.reload();

 return false;

 }

 } // end switch

 **return false;** //prevent any other keys to function or have behavior of the defined spacebar functionality

} // end handleKeyDown()

//

// Function moveDown() is a member function to select the next item in a menu.

// If the active item is the last in the menu, this function will loop to the

// first menu item.

//

// @param($item object) $item is the active menu item

//

// @param(startChr char) [optional] startChr is the character to attempt to match against the beginning of the

// menu item titles. If found, focus moves to the next menu item beginning with that character.

//

// @return (object) Returns the item to move to. Returns $item is no move is possible

//

menubar.prototype.moveDown = function($item, startChr) {

 var $itemUL = $item.parent(); // $item's containing menu

 var $menuItems = $itemUL.children('li').not('.separator'); // the items in the currently active menu

 var menuNum = $menuItems.length; // the number of items in the active menu

 var menuIndex = $menuItems.index($item); // the items index in its menu

 var $newItem = null;

 var $newItemUL = null;

 if ($itemUL.is('.root-level')) { // this is the root level menu

 if ($item.attr('aria-haspopup') != 'true') {

 // No child menu to move to

 return $item;

 }

 // Move to the first item in the child menu

 $newItemUL = $item.children('ul').first();

 $newItem = $newItemUL.children('li').first();

 // make sure the child menu is visible

 $newItemUL.show().attr('aria-hidden', 'false');

 this.bChildOpen = true;

 return $newItem;

 }

 // if $item is not the last item in its menu, move to the next item. If startChr is specified, move

 // to the next item with a title that begins with that character.

 //

 if (startChr) {

 var bMatch = false;

 var curNdx = menuIndex+1;

 // check if the active item was the last one on the list

 if (curNdx == menuNum) {

 curNdx = 0;

 }

 // Iterate through the menu items (starting from the current item and wrapping) until a match is found

 // or the loop returns to the current menu item

 while (curNdx != menuIndex) {

 // Use the first of the two following lines if menu does not contain anchor tags.

 // Otherwise use the second

 // var titleChr = $menuItems.eq(curNdx).html().charAt(0);

 var titleChr = $menuItems.eq(curNdx).find('a').html().charAt(0);

 if (titleChr.toLowerCase() == startChr) {

 bMatch = true;

 break;

 }

 curNdx = curNdx+1;

 if (curNdx == menuNum) {

 // reached the end of the list, start again at the beginning

 curNdx = 0;

 }

 }

 if (bMatch == true) {

 $newItem = $menuItems.eq(curNdx);

 // remove the focus styling from the current item

 $item.removeClass('menu-focus');

 return $newItem

 }

 else {

 return $item;

 }

 }

 else {

 if (menuIndex < menuNum-1) {

 $newItem = $menuItems.eq(menuIndex+1);

 }

 else {

 $newItem = $menuItems.first();

 }

 }

 // remove the focus styling from the current item

 $item.removeClass('menu-focus');

 return $newItem;

}

//

// Function moveUp() is a member function to select the previous item in a menu.

// If the active item is the first in the menu, this function will loop to the

// last menu item.

//

// @param($item object) $item is the active menu item

//

// @return (object) Returns the item to move to. Returns $item is no move is possible

//

menubar.prototype.moveUp = function($item) {

 var $itemUL = $item.parent(); // $item's containing menu

 var $menuItems = $itemUL.children('li').not('.separator'); // the items in the currently active menu

 var menuNum = $menuItems.length; // the number of items in the active menu

 var menuIndex = $menuItems.index($item); // the items index in its menu

 var $newItem = null;

 var $newItemUL = null;

 if ($itemUL.is('.root-level')) { // this is the root level menu

 // nothing to do

 return $item;

 }

 // if $item is not the first item in its menu, move to the previous item

 if (menuIndex > 0) {

 $newItem = $menuItems.eq(menuIndex-1);

 }

 else {

 // loop to top of menu

 $newItem = $menuItems.last();

 }

 // remove the focus styling from the current item

 $item.removeClass('menu-focus');

 return $newItem;

}

//

// Function handleKeyPress() is a member function to process keydown events

// for the menus.

//

// The Opera browser performs some window commands from the keypress event,

// not keydown like Firefox, Safari, and IE. This event handler consumes

// keypresses for relevant keys so that Opera behaves when the user is

// manipulating the menu with the keyboard.

//

// @param($item object) $menu is the jquery object of the item firing the event

//

// @param(e object) e is the associated event object

//

// @return(boolean) Returns false if consuming; true if propagating

//

menubar.prototype.handleKeyPress = function($item, e) {

 if (e.altKey || e.ctrlKey || e.shiftKey) {

 // Modifier key pressed: Do not process

 return true;

 }

 switch(e.keyCode) {

 case this.keys.tab: {

 return true;

 }

 case this.keys.esc:

 case this.keys.enter:

 case this.keys.space:

 case this.keys.up:

 case this.keys.down: {

 e.stopPropagation();

 return false;

 }

 default : {

 var chr = String.fromCharCode(e.which);

 this.$activeItem = this.moveDown($item, chr);

 this.$activeItem.focus();

 e.stopPropagation();

 return false;

 }

 } // end switch

 return true;

} // end handleKeyPress()

/////////////// end menu widget definition /////////////////////

</script>

---------------------------------------------------------------------------------------------------

### HTML:

<body>

<h1>Accessible Menu - Dev 1</h1>

<p> This is a paragraph, and this is a <a href="www.csun.edu">link to CSUN </a> </p>

<div role="application" aria-label="Global Menu" tabindex="0">

 <ul id="nav" class="menubar root-level" role="menubar">

 <li class="menu-parent" role="menuitem" tabindex="0" aria-haspopup="true" title="Item one">

 <a href="somepage.html" tabindex="-1">Item one</a>

 <ul role="menubar" class="menu" aria-hidden="true">

<li role="menuitem" class="menu-item" tabindex="-1" title="item 1.1">

 <a href="somepage-submenu.html">item 1.1</a></li>

<li role="menuitem" class="menu-item" tabindex="-1" title="item 1.2">

 <a href="somepage-submenu.html">item 1.2</a></li>

<li role="menuitem" class="menu-item" tabindex="-1" title="item 1.3">

 <a href="somepage-submenu.html">item 1.3</a></li>

<li role="menuitem" class="menu-item" tabindex="-1" title="item 1.4">

 <a href="somepage-submenu.html">item 1.4</a></li>

</ul>

</li>

<li class="menu-parent" role="menuitem" tabindex="0" aria-haspopup="true" title="Item two">

 <a href="somepage.html" tabindex="-1">Item two</a>

 <ul role="menubar" class="menu" aria-hidden="true">

<li role="menuitem" class="menu-item" tabindex="-1" title="item 2.1">

 <a href="somepage-submenu.html">item 2.1</a></li>

<li role="menuitem" class="menu-item" tabindex="-1" title="item 2.2">

 <a href="somepage-submenu.html">item 2.2</a></li>

<li role="menuitem" class="menu-item" tabindex="-1" title="item 2.3">

 <a href="somepage-submenu.html">item 2.3</a></li>

<li role="menuitem" class="menu-item" tabindex="-1" title="item 2.4">

 <a href="somepage-submenu.html">item 2.4</a></li>

</ul>

</li>

<li class="menu-parent" role="menuitem" tabindex="0" aria-haspopup="true" title="Item three">

<a href="somepage.html" tabindex="-1">Item three</a>

<ul role="menubar" class="menu" aria-hidden="true">

<li role="menuitem" class="menu-item" tabindex="-1" title="item 3.1">

<a href="somepage-submenu.html">item 3.1</a></li>

<li role="menuitem" class="menu-item" tabindex="-1" title="item 3.2">

<a href="somepage-submenu.html">item 3.2</a></li>

<li role="menuitem" class="menu-item" tabindex="-1" title="item 3.3">

<a href="somepage-submenu.html">item 3.3</a></li>

</ul>

</li>

</ul>

</div>

<span> </span>

</body>

# ARIA for Proper Assistive Technology Functionality

## Purpose:

Users should be properly acknowledged when they land on a drop down menu region while navigating through the web page. They should be informed about how to access the sub-menu items and how to navigate through them. This information should be provided when the drop down menu item is activated while receiving focus.

## Technique:

This technique utilizes **ARIA** roles and attributes to announce the general navigational behavior of the drop down menu to Assistive Technology agent. Aria-hidden attribute is used to change the visually hidden status of the drop down when the user activates drop down main menu items. In addition, the combination of aria-role=”application” and aria-label provides general instruction to the Assistive Technology users that they have landed on the drop down menu region. Using proper ARIA roles and attributes such as application, menubar, menuitem and aria-haspopup, a menu region is built that Assistive Technology users can easily interact and get the information in a meaningful ways.

### Roles:

*Role= “application”:* this indicates that the keyboard events will be switched to an application browsing mode.

*Role= “menubar”:* this role is used to create a presentation of menu that usually presented horizontally

*Role= “menuitem”:* this role is used to indicate the child element of the element that has menubar role. It can be used with aria-haspopup attribute to indicate that the item can be used to launch the sub-menu level.

### **Attributes:**

*Aria-label:* this attribute is used to provide the Assistive Technology user with a recognizable name of the object

*Aria-haspopup:* this attribute is used to indicate that the element has a popup sub-menu level.

*Aria-hidden:* this attribute is used to indicate that the element and all of its descendants are not perceivable to any users

## Source Code

### HTML Code:

<div **role="application"** **aria-label="Global Menu"** tabindex="0">

<ul id="nav" class="menubar root-level" **role="menubar"**>

<li class="menu-parent" **role="menuitem"** tabindex="0" **aria-haspopup="true"** title="Item one">

<a href="somepage.html" tabindex="-1">Item one</a>

<ul **role="menubar"** class="menu" **aria-hidden="true"**>

<li **role="menuitem"** class="menu-item" tabindex="-1" title="item 1.1">

<a href="somepage-submenu.html">item 1.1</a></li>

<li **role="menuitem"** class="menu-item" tabindex="-1" title="item 1.2">

<a href="somepage-submenu.html">item 1.2</a></li>

<li **role="menuitem"** class="menu-item" tabindex="-1" title="item 1.3">

<a href="somepage-submenu.html">item 1.3</a></li>

<li **role="menuitem"** class="menu-item" tabindex="-1" title="item 1.4">

<a href="somepage-submenu.html">item 1.4</a></li>

</ul>

</li>

<li class="menu-parent" **role="menuitem"** tabindex="0" **aria-haspopup="true"** title="Item two">

<a href="somepage.html" tabindex="-1">Item two</a>

<ul **role="menubar"** class="menu" **aria-hidden="true"**>

<li **role="menuitem"** class="menu-item" tabindex="-1" title="item 2.1">

<a href="somepage-submenu.html">item 2.1</a></li>

<li **role="menuitem"** class="menu-item" tabindex="-1" title="item 2.2">

<a href="somepage-submenu.html">item 2.2</a></li>

<li **role="menuitem"** class="menu-item" tabindex="-1" title="item 2.3">

<a href="somepage-submenu.html">item 2.3</a></li>

<li **role="menuitem"** class="menu-item" tabindex="-1" title="item 2.4">

<a href="somepage-submenu.html">item 2.4</a></li>

</ul>

</li>

<li class="menu-parent" **role="menuitem"** tabindex="0" **aria-haspopup="true"** title="Item three">

<a href="somepage.html" tabindex="-1">Item three</a>

<ul **role="menubar"** class="menu" **aria-hidden="true"**>

<li **role="menuitem"** class="menu-item" tabindex="-1" title="item 3.1">

<a href="somepage-submenu.html">item 3.1</a></li>

<li **role="menuitem"** class="menu-item" tabindex="-1" title="item 3.2">

<a href="somepage-submenu.html">item 3.2</a></li>

<li **role="menuitem"** class="menu-item" tabindex="-1" title="item 3.3">

<a href="somepage-submenu.html">item 3.3</a></li>

</ul>

</li>

</ul>

</div>

# References:

* <http://www.w3.org/TR/wai-aria/roles#application>
* <http://www.w3.org/TR/wai-aria/roles#menuitem>
* <http://www.w3.org/TR/wai-aria/roles#menubar>
* <http://www.w3.org/TR/wai-aria/states_and_properties#aria-label>
* <http://www.w3.org/TR/wai-aria/states_and_properties#aria-haspopup>
* <http://www.w3.org/TR/wai-aria/states_and_properties#aria-hidden>
* <http://oaa-accessibility.org/example/26/> (Open Ajax)