ATLS 3519 – Lab #8 – (12 pts.)

Lab#8: Using WAI-ARIA to make complex pages & interactions accessible

## Objectives:

* To learn how to use WAI-ARIA roles and functions with javascript to increase the accessibility of complex pages (i.e. rich Internet applications)

# The files you need to start this lab are at <http://webdevgroupcu.org/atlas3519/wai-aria/>

## Scenario 1 – prompting a user with a dialog box to confirm an action

(for example, before the user leaves a page or form without saving or submitting)

Problem: Assistive Technology may not know a dialog box has opened and focus (i.e. the cursor) will likely not be routed to the new window.

WAI-ARIA used in this exercise

role="alertdialog"

aria-labelledby=”[element id of your choice]”

Open ***Alert\_dialog\_inaccessible.htm***

### Test the page with VoiceOver

1. What does VO read when the dialog box opens?
2. What happens when the “stay on this page” button is selected?

### Correcting the problem

The div that holds the dialog box is below (id=”b”)

|  |
| --- |
| <div id=a”>  <div id="b>  <h2>Do you want to stay on this page or do you want to visit Google?</h2>  <button onclick="hide();" id="StayButton">Stay on this Page</button>  <a href="http://www.google.com" id="GoogleLink">Go to Google</a>  </div>  </div> |

### Add the attributes:

***role="alertdialog" aria-labelledby="mymessage"***

to the b div. It should now look as follows:

*<div id="b"* ***role="alertdialog" aria-labelledby ="mymessage"*** >

### Add the id of “mymessage” to the h2 header:

*<h2* ***id="mymessage"****>*

### Test the page again with VoiceOver

Is the problem corrected?

### Adding focus

To ensure that focus moves to the dialog box and back to the original window (unless the user chooses to “go to Google”, add the following to the javascript as shown below:

***document.getElementById("StayButton").focus();***

***document.getElementById("ShowButton").focus();***

|  |
| --- |
| *function show() {*  *disableButtons();*  *window.myparent.appendChild(window.myobj);*  ***document.getElementById("StayButton").focus();***  *}*  *function hide() {*  *enableButtons();*  *window.myparent = document.getElementById("a");*  *window.myobj = document.getElementById("b");*  *window.myparent.removeChild(window.myobj);*  ***document.getElementById("ShowButton").focus();***  *}* |

### Now test it out again with VoiceOver (after you save and post it).

Does this solve the problem?

### What about for the visual user? How can we improve the visual focus on the dialog box?

How can we change the background color of the original page when the dialog appears?

Need help finding a color and associated code: <http://www.dasplankton.de/ContrastA/>

Scenario 2 – We want to the screenreader user to know when an error message has appeared on a form (right after leaving the field with the error)

Open ***Alert\_message\_inaccessible.htm***

### Test the page with VoiceOver

1. What happens when a required field is not entered? (i.e. the email address)

### Correcting the problem

This time we will use the WAI-ARIA “role=alert”.

The javascript shown below places the error message on the page when an improper email is not entered.

To have the screenreader read this error message it needs to be “alerted” to the message. We accomplish this by adding the attribute “role=alert” to the object that contains the error message – in this case, a “<p>” element.

|  |
| --- |
| <script language="JavaScript" type="text/javascript">  function show\_new\_object() {  // create a paragraph to be the alert message  var myAlert = document.createElement("p");  myAlert.setAttribute('id','mess2');  ***myAlert.setAttribute('role','alert');***  // make it look nice  myAlert.style.color='#AF3D5C';  myAlert.style.fontWeight='bold';  // include some text  var msg = document.createTextNode("Please provide a valid email address.");  myAlert.appendChild(msg);  // place the alert on the page  window.myparent = document.getElementById("place4alert");  window.myparent.appendChild(myAlert);  } |

After making your changes, upload the new file and see if it is now read by VoiceOver.

Scenario 3 – We want to the screenreader user to know when a timer updates on the screen

Open ***Timer\_Problem.htm***

Access the page with VoiceOver and start the time. What’s the problem?

Based on the code in the document and our previous discussion regarding ARIA roles, how do you think we could solve this problem?

## Scenario 4 – We have a long complex form we want to make simpler through progressive disclosure but we also want to maintain accessibility

If you haven’t already done so, post ***proposalform2014-problem1.html*** and its associated style sheet ***ahg\_scrn\_wform\_atlas.css*** onto the your server directory (as always, I recommend creating a separate directory for each project)

Open ***proposalform2014-problem1.html*** in your browser and then compare it to:

<http://accessinghigherground.org/wp/proposalform.php>

What else can we do to make this form more useable?

Creating a expandable/collapsible section

Using jquery accessible accordion (the plugin is referred to as accordeon by the author)

View plugin page at: <http://www.sitepoint.com/introducing-accessible-accordion-widget/>

## Hiding the second speaker section

Add the following code below the *<div id="formcol2">* code which heads the second speaker section:

*<div id="formcol2">*

***<ul class="a11yAccordeon" id="Accordeon1">***

***<li class="a11yAccordeonItem">***

***<div class="a11yAccordeonItemHeader">***

***<strong>Second Speaker Section</strong>***

***</div>***

***<div class="a11yAccordeonHideArea">***

Where do we put the closing div for ***a11yAccordeonHideArea***?

## Linking to the plugin and the script

Add the following code at about line 29 – just before the first javascript code and after the tests for versions of IE

<!-- js & other assets for accessible accordion -->

<script src="js/a11yAccordeon-0.2.2.min.js"></script>

<!--jQuery Library -->

<script src="http://code.jquery.com/jquery-latest.js"></script>

## Adding the javascript

We’ll add the code in with the <body> tag – more or less the same code from the a11yaccordeon site

*<body* ***onload="load();"****>*

***<script type="text/javascript">***

***var load = function () {***

***var myAccordeon = a11yAccordeon({***

***parentSelector: '#Accordeon1'***

***});***

***};***

***</script>***

After saving and uploading, you should see a collapsed section for “Second Speaker Section”

## Check with VoiceOver

Open page with VoiceOver and move through the page. Is it easy to move through the page. What happens when you open and close the second speaker section?

## Other Enhancements

We don’t need the search function for this use of the accordion. We can remove it by adding the following code to our “onload” script:

***showSearch: false***

We can also provide an audio-only message to the screenreader user by adding the following line:

***hiddenLinkDescription: 'select to expand',***

You script should look as follows:

<body onLoad="load();">

<script type="text/javascript">

var load = function () {

var myAccordeon = a11yAccordeon({

parentSelector: '#Accordeon1',

***hiddenLinkDescription: 'select to expand', // or choose your own phrase***

***showSearch: false***

});

};

</script>

## Adding focus

We can also improve both the accessibility and usability by adding focus after the opening of the accordion by moving focus to the first form field in the section – second speaker’s first name.

Add the following code the onload script after the first script and just below the closing bracket and closing </script> tag.

***$("a.a11yAccordeonItemHeaderLink").click(function(){***

***$("#secondspeak").attr("role", "alert");***

***document.getElementById("secondspkr").focus();***

***});***

You script should now look like:

<body onLoad="load();">

<script type="text/javascript">

var load = function () {

var myAccordeon = a11yAccordeon({

parentSelector: '#Accordeon1',

hiddenLinkDescription: 'select to expand',

showSearch: false

});

***$("a.a11yAccordeonItemHeaderLink").click(function(){***

***$("#secondspeak").attr("role", "alert");***

***document.getElementById("secondspkr").focus();***

});

};

</script>

Test it with VoiceOver – does it work better for the screenreader user.

## Adding Landmarks

We can also add semantics to the page by adding aria landmarks. The following is a list of aria landmarks. Add the ones where you think it is the most appropriate;

### 5.3.4. Landmark Roles

[#](http://www.w3.org/TR/wai-aria/roles#landmark_roles)

The following [*roles*](http://www.w3.org/TR/wai-aria/terms#def_role) are regions of the page intended as navigational [*landmarks*](http://www.w3.org/TR/wai-aria/terms#def_landmark). All of these roles inherit from the landmark base type and, with the exception of application, all are imported from the [*Role Attribute*](http://www.w3.org/TR/2011/WD-role-attribute-20110113/#s_role_module_attributes) [[*ROLE*](http://www.w3.org/TR/wai-aria/references#ref_ROLE)]. The roles are included here in order to make them clearly part of the WAI-ARIA Role [*taxonomy*](http://www.w3.org/TR/wai-aria/terms#def_taxonomy).

* [application](http://www.w3.org/TR/wai-aria/roles#application)
* [banner](http://www.w3.org/TR/wai-aria/roles#banner)
* [complementary](http://www.w3.org/TR/wai-aria/roles#complementary)
* [contentinfo](http://www.w3.org/TR/wai-aria/roles#contentinfo)
* [form](http://www.w3.org/TR/wai-aria/roles#form)
* [main](http://www.w3.org/TR/wai-aria/roles#main)
* [navigation](http://www.w3.org/TR/wai-aria/roles#navigation)
* [search](http://www.w3.org/TR/wai-aria/roles#search)

## 5.4. Definition of Roles

[#](http://www.w3.org/TR/wai-aria/roles#role_definitions)

At <http://www.w3.org/TR/wai-aria/roles#landmark_roles> is an alphabetical list of WAI-ARIA [*roles*](http://www.w3.org/TR/wai-aria/terms#def_role) to be used by rich internet application authors.

## Finding landmarks with VoiceOver

VO with left & right arrow keys takes you into nav mode

“w” should then take you to the landmarks.

Due Date: March 19, 2014

Submit your urls to: Dropbox 8