Introduction to WAI-ARIA 1.0

Mike Elledge
Assistant Director
Usability & Accessibility Center
Topics

• Brief History of Web Development
• Current Accessibility Issues
• What is AJAX and how does it work?
• AJAX and Accessibility
• WAI-ARIA
• Examples of Accessible AJAX Using ARIA
• ARIA Status, Tips and Resources

Note: Unless otherwise indicated, information for this presentation has been adapted from the WAI-ARIA document suite.
Brief History of Web Development

- Static web pages
  - Most basic of web pages
  - Simple table-based layouts, HTML used for formatting
  - **Accessibility issues**: nested tables, illegal code (blockquote, headings), missing alt descriptions
- Static web pages with JavaScript (DOM-scripting, DHTML)
  - Web pages with features that changed with user actions
  - Drop down menus, text highlighting
  - Browser and not server-side changes
  - **Accessibility issues**: Unreadable JS (AT), unnecessary animation/images (“eye candy”), inconsistency between browsers, missing alt descriptions.
Now

• Static web pages using CSS (& JS)
  – CSS used in place of or in concert with JS
  – CSS-based positioning, dropdowns, alternative style sheets
  – Accessibility issues: Inconsistent display (browsers), fixed fonts, missing alt descriptions and headings

• AJAX using JS and CSS
  – Stocktickers, clocks, interactive maps, autocomplete dropdowns
  – Accessibility issues: Page changes not recognized by AT, underlying code not readable by older browsers, missing alt descriptions

Source: AJAX, Creating Web Pages with Asynchronous JavaScript and XML, Woychowky
Current Accessibility Issues

Weighted average of worst:

Worst:
- CAPTCHA (28%)
- Flash (22%)
- Keyboard Accessibility (10%)
- Ambiguous Links (10%)

Source: WebAIM Screen Reader User Survey, 10/09
ARIA Related Results

- Which of the following best describes your use of landmarks?

<table>
<thead>
<tr>
<th>Response</th>
<th># of Respondents</th>
<th>% of Respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>I didn’t know this functionality existed</td>
<td>240</td>
<td>42.1%</td>
</tr>
<tr>
<td>I sometimes use landmarks for navigation</td>
<td>183</td>
<td>32.1%</td>
</tr>
<tr>
<td>I use landmarks for navigation whenever they are present</td>
<td>117</td>
<td>20.5%</td>
</tr>
<tr>
<td>My screen reader does not support landmarks</td>
<td>30</td>
<td>5.3%</td>
</tr>
</tbody>
</table>

Source: WebAIM Screen Reader User Survey, 10/09
What is AJAX?

- AJAX = Asynchronous JavaScript + XML
- AJAX Components:
  - HTML and CSS for marking up and styling information.
  - The DOM accessed with JavaScript to dynamically display and interact with the information presented.
  - A method for exchanging data asynchronously between browser and server, thereby avoiding page reloads. The XMLHttpRequest (XHR) object is usually used, but sometimes an IFrame object or a dynamically added <script> tag is used instead.
  - A format for the data sent to the browser (such as XML).

Static Web Page

Server

Browser

Page 1 Page 2 Page 3 Page 4

Topics

Source: “Ajax, JavaScript and PHP,” Ballard & Moncur
Static Web Page with Form

Server Running PHP

- Processing
- Processing
- Processing

Page 1  Page 2  Page 3  Page 4

Browser

Topics

Source: After “Ajax, JavaScript and PHP,” Ballard & Moncur
Drawbacks to Static Web Pages

1. They are slow
   a) Each time you submit something, the information is sent to the server where it is then processed
   b) In the meantime you are waiting for the next page

2. They are inefficient
   a) After it is processed, a full page is returned to you: header, top navigation bar, side navigation, footer, plus new content
   b) Uses internet bandwidth and further slows things down

3. They are an old paradigm
   a) Computer applications provide a more responsive and interactive experience by comparison
   b) People are beginning to expect Rich Interactive Applications

Source: “Ajax, JavaScript and PHP,” Ballard & Moncur
AJAX-based Web Page

Server Running PHP

AJAX

Page 1

Browser

Topics

Source: “Ajax, JavaScript and PHP,” Ballard & Moncur
AJAX Examples

- Google Maps (http://maps.google.com/)
- Yahoo! Search (autocomplete)

- Both are accessible applications by the way, although the Google map image is not...
How Screen Readers Work

• A screen reader (like JAWS) gives information about a page and its content to persons who cannot see it.
  1. JAWS user types webpage address into browser, browser sends info to server, server returns page, page appears in browser.
  2. Browser sends info about page elements and content based on html markup to the MSAA API, where it is interpreted and sent to screen reader.
• The API identifies what is known as “semantic content.”
  4. Screen reader stores the info in its cache and binds with the computer keyboard.
  5. User navigates and operates page with their keyboard by pressing specific keys (like Tab) and keystroke combinations (like insert + down arrow)
  6. Screen reader interprets those keystrokes, reads them, and announces additional information (like “header level one”) to the user according to the html tags and attributes on the page.
Topics

Accessibility Interaction
No JavaScript

Screen reader: “I have saved the page information and will describe its contents out loud based on the keystrokes someone presses. I understand what the list is because the API has told me about it.”

Source: WAI-ARIA, http://www.w3.org/TR/wai-aria/

DOM: “I am a page that contains an HTML element called an ordered list. The list contains three items that have links to other pages. You use it to navigate to another page. The browser makes it visible and interactive.”

MSAA API: “I understand that a list contains semantic information (items are in a group and have a parent) and text. I will translate this information in the DOM to AT.”
Screen Readers and AJAX

- AJAX, and until recently JavaScript, caused the understanding between the DOM and API to breakdown because there was no definition provided to the screen reader by the API for what the JavaScript was causing to happen.
“I am JavaScript. I’m going to change the list when you tab to an item so that it gives you another list of pages to navigate to. You can see the changes in the browser, but you won’t be told about it, since the API doesn’t understand what I am doing and doesn’t know what to tell the screen reader.”

Source: WAI-ARIA, http://www.w3.org/TR/wai-aria/
The Role of ARIA

• The JavaScript now directs the UI, data, and content to behave differently than in a static page, changing their roles, properties and states...

• For this to be accessible to a person who cannot see it, there has to be a way to describe those changes and interactions to the API, so it can in turn correctly inform the AT, which also has to be able to understand those changes...

• That is ARIA!
The WAI-ARIA solution

Source: WAI-ARIA, http://www.w3.org/TR/wai-aria/
WAI-ARIA 1.0

- WAI-ARIA 1.0 is a specification that provides an ontology (representation of concepts) of roles, states and properties for accessible user interface elements
  - The roles, states and properties can be used by developers to describe user interface behavior and document structure to Assistive Technology (AT)
  - Doing so will improve (enable) accessibility, usability and interoperability of web content and applications

- Accessibility is impossible without providing a mechanism for AT to understand the semantics of content as implemented through JavaScript
Status of ARIA

- WAI-ARIA is a working draft, the first of four levels to becoming a recommendation
  - This allows the community to try out the spec, identify missing pieces or problems, and make suggestions
- Overall, 47 of the 59 WAI-ARIA roles can be exposed to AT by browsers (via the MSAA API)
- Specific browser support:
  - Firefox (Minefield) 3.6: 47 role values
  - IE 8RC1: 43 role values
  - Opera 10.10 beta: 21 role values
  - Webkit Nightly 23/09/09: 10 role values
  - Chrome 3.0: 18 role values

Source: [http://www.paciellogroup.com/blog/?p=474](http://www.paciellogroup.com/blog/?p=474)
WAI-ARIA Components

- ARIA uses three terms to describe interactive content: Roles, Properties and States.
  - **Roles**: Identify widget types and structure of elements, such as landmark areas or buttons, to AT.
    - Four types: base types, user input widgets, user interface elements, document structure, specialized regions and landmark roles.
  - **States and properties**: Identify (declare) attributes of elements that affect and describe interaction, such as navigation or whether a button is pressed.
    - Five categories: widgets, live regions, drag-and-drop and relationships (all attributes).

- Together they enable the accessibility API to inform the AT how to describe page content and interaction to the user.
Example of Role--checkbox

checkbox (role)

• A checkable input that has three possible values: true, false, or mixed.

• The aria-checked attribute of a checkbox indicates the state of the input, whether it is checked (true), unchecked (false), or represents a group of elements that have both checked and unchecked values (mixed).

• Characteristics of checkbox (next page)
ARIA Roles

Base Types (Abstract)
- composite (abstract role)
- landmark (abstract role)
- roletype (abstract role)
- structure (abstract role)
- widget (abstract role)
- window (abstract role)

Landmark
- application
- banner
- complementary
- contentinfo
- main
- navigation
- search
ARIA Roles

Document Structure
article
columnheader
definition
directory
document
grid
gridcell
group
heading
img
list
listitem
math
note
presentation

Topics

region
row
rowheader
section (abstract role)
sectionheader (abstract role)
separator

Application Structure
alert
alertdialog
dialog
log
marquee
progressbar
status
timer
ARIA Roles

User Interface Elements
button
link
menu
menubar
menuitem
menuitemcheckbox
menuitemradio	
tablist
tabpanel	
tab	
toolbar
tooltip
tree
treegrid
treeitem

User Input Widgets
checkbox
combobox
input (abstract role)
listbox
option
radio
radiogroup
range (abstract role)
select (abstract role)
slider
spinbutton
textbox

Topics
Example of State—aria-checked

aria-checked (state)

- Indicates the current "checked" state of checkboxes, radio buttons, and other widgets.

- The aria-checked attribute indicates whether the element is checked (true), unchecked (false), or represents a group of other elements that have a mixture of checked and unchecked values (mixed).

- The mixed value is supported by certain tri-state inputs such as a checkbox or menuitemcheckbox.
ARIA States and Properties

Global

aria-atomic
aria-busy (state)
aria-controls
aria-describedby
aria-disabled (state)
aria-dropeffect (state)
aria-flowto
aria-grabbed (state)
aria-haspopup

aria-hidden (state)
aria-invalid (state)
aria-label
aria-labelledby
aria-live
aria-owns
aria-relevant
aria-required
ARIA States and Properties

Widgets

aria-autocomplete
aria-checked (state)
aria-disabled (state)
aria-expanded (state)
aria-haspopup
aria-hidden (state)
aria-invalid (state)
aria-level
aria-multiline
aria-multiselectable

aria-pressed (state)
aria-readonly
aria-required
aria-selected (state)
aria-sort
aria-valuemax
aria-valuemin
aria-valuenow
aria-valuetext
ARIA States and Properties

Live Regions
aria-atomic
aria-busy (state)
aria-live
aria-relevant

Drag and Drop
aria-dropeffect (state)
aria-grabbed (state)

Relationships
aria-activedescendant
aria-controls
aria-describedby
aria-flowto
aria-label
aria-labelledby
aria-owns
aria-posinset
aria-setsize
WAI-ARIA Widgets
Design patterns in ARIA Best Practices document

alert (1) autocomplete (1) button (1) checkbox (4) combobox (3) datepicker (1) dialog_modal (2) dialog_nonmodal (--) dialog_tooltip (--) draganddrop (2) grid (3) landmarks (3) link (2) listbox (2) mediaplayer menu (1) menu_button (1) popup_menu (1) popup_help (--) radiobutton (3) richtext_editor (1) slider (4) slidertwothumb (2) spinbutton (1) tabpanel (2) toolbar (1) tooltip (1) treegrid (2) treeview1 (2) windowsplitter (--) wizard (--)
ARIA Tips

- Works best with most current browsers and AT
  - Firefox 3.5, JAWS 10
  - Write for older browsers and AT: graceful degradation
- Common conventions
  - Tab = Move to item
  - Arrows = Move within item
  - ? = Set focus on item
  - Enter = Initiate action
- Provide description of how widget functions for AT users
- If widget doesn’t work, try turning off virtual cursor (insert + z) and try it again
Some ARIA-Based Examples

- Drag and Drop Widget
  - [http://test.cita.uiuc.edu/aria/draganddrop/draganddrop2.php](http://test.cita.uiuc.edu/aria/draganddrop/draganddrop2.php)
  - Roles: application, listbox, option
  - Properties: aria-labelledby, aria-disabled, aria-grabbed, aria-selected, aria-dropeffect

- Landmark Navigation Widget (ins+ctrl+;) 
  - Roles: main, search, navigation, and complementary

- Slider Widget 
  - [http://www.paciellogroup.com/blog/misc/ARIA/slider/](http://www.paciellogroup.com/blog/misc/ARIA/slider/)
  - Role: slider 
  - Properties: aria-valuetext, aria-valuenow, aria-valuemax, aria-valuemin, aria-labelledby
ARIA Resources—W3C WAI

- **Overview** ([http://www.w3.org/WAI/intro/aria.php](http://www.w3.org/WAI/intro/aria.php))
  - Introduction to WAI-ARIA publications
- **Accessible Rich Internet Applications 1.0** ([http://www.w3.org/TR/wai-aria/](http://www.w3.org/TR/wai-aria/))
  - Definitions & descriptions of roles, states, properties, and requirements
- **Best Practices** ([http://www.w3.org/TR/wai-aria-practices/](http://www.w3.org/TR/wai-aria-practices/))
  - Descriptions of ARIA implementations and examples
- **Primer** ([http://www.w3.org/TR/wai-aria-primer/](http://www.w3.org/TR/wai-aria-primer/))
  - Accessibility issues and technical approach to creating ARIA
- **Roadmap for Accessible Rich Internet Applications** ([http://www.w3.org/TR/wai-aria-roadmap/](http://www.w3.org/TR/wai-aria-roadmap/))
  - Past efforts and future steps to make RIA accessible
- **User Agent Implementation Guide** ([http://www.w3.org/TR/wai-aria-implementation/](http://www.w3.org/TR/wai-aria-implementation/))
  - How implementations (like widgets) should show content to Accessibility APIs
Other ARIA Resources

  - Descriptions, links to ARIA demos
- **DOJO Toolkit**: [http://dojotoolkit.org/](http://dojotoolkit.org/)
  - AJAX and DHTML code for numerous widgets
- **Fluid project**: [http://fluidproject.org/releases/1.1.2/demos/](http://fluidproject.org/releases/1.1.2/demos/)
  - Accessible widgets including code
- **iCITA**: [http://test.cita.illinois.edu/aria/](http://test.cita.illinois.edu/aria/)
  - Many examples of ARIA widgets including code
- **jQuery**: [http://jquery.com/](http://jquery.com/)
  - Code for numerous JavaScript and AJAX functions
  - Information about ARIA support, examples
Next

• Carl Bussema and Carl Raymond, UOE CIT
• ARIA examples and explanations
  – Alert/Dialog
  – Dropdown Menu/Selector