International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 9 (2014), pp. 925-930
© International Research Publications House
http://www. irphouse.com

WAI-ARIA: Method To Develop Disable Friendly Websites

Fazia Fatima[1], Shipra Rawal[2],
Chinmay Garg[3] and P N Barwal[4]

[1] Project Associate, [2] Project Engineer, [3]Technical Officer, [4]Joint Director
[1,2,3,4] e-Governance, Centre for Development of Advanced Computing.
C-56/1, Anusandhan Bhawan, C block, Sector 62, Noida – 201307.

Abstract

Web accessibility is a serious concern to provide universal access to a website. Inaccessible websites can put a significant barrier to people with disabilities. This paper focuses on identifying a technique which improves the accessibility and interoperability of web content and applications. The paper discusses about Web Accessibility Initiative – Accessible Rich Internet Application (WAI-ARIA) that defines a way to make Web content and Web applications more accessible. This method especially deals with the dynamic content and advanced user interface controls which are developed using HTML, JavaScript, Ajax and other related technologies. We finally propose that using this methodology will not only make the website globally accessible but also provides additional benefits.

Keywords- Accessibility; Disable Friendly ; WAI-ARIA; SEO; Web usability

1. Introduction

Accessibility can be viewed as the "ability to access" and benefit from some system or entity. The concept often focuses on people with disabilities or special needs. It is essential that the Web be accessible in order to provide equal access and equal opportunity to people with diverse abilities.

This research paper on WAI-ARIA enlightens the different roles, states and properties. For the past few years, we are in a state of dilemma for accessible websites, may be due to the lack of awareness.

2. Accessibility Problems

Web developers nowadays use client-side scripts to create user interface controls that cannot be created with HTML alone. They also use client-side scripts to update sections of a page without requesting a completely new page from a web server. Such techniques on websites are called rich Internet applications. Visually, emulating rich components and making server requests in the background creates a richer experience for users. Unfortunately, this results in accessibility problems that are particular bad for users of assistive technologies, such as screen reader user.

- Widgets which are built in this way are rarely keyboard accessible.
- The role of the widget, what it does, is not available to assistive technology.
- States and properties of the widget are not available to assistive technology.
- Updates and discovery of the updates are not reported to assistive technology.

3. WAI-ARIA

Fortunately, the problems listed above are solved by the WAI-ARIA (Web Accessibility Initiative -Accessible Rich Internet Application). It is a positive, enabling technology — rather than telling developers what they cannot do, ARIA allows developers to create rich web applications. It is a technical specification published by the World Wide Web Consortium that defines a way to make Web content and Web Applications (especially those which are developed with Ajax and JavaScript) more accessible to people with disabilities. It enables accessible navigation landmarks, JavaScript widgets, form hints and error messages, live content updates and more.

3.1 Keyboard Navigation

Along with providing alternative text for non-text objects, being able to interact with interface elements using the keyboard alone is one of the most basic accessibility provisions. The tabindex attribute from HTML 4 accepts a positive value between 0 and 32767. Elements with a value of 0 are visited in the order they appear in the markup. ARIA extends the tabindex attribute so that it can be used on all visible elements. ARIA also allows a negative value to be specified for elements that should not appear in the keyboard tab order, but can be programmatically. The following example uses a negative tabindex attribute value, so that the element can receive programmatic focus.

```
<div id=”divFocus” tabindex = “-1”>
//code
</div>
```

In this example, the div element is not placed in the tab order, but having a tabindex attribute value of “ -1” means that it can receive programmatic focus. The following snippet of JavaScript selects the element defined above, and uses the focus method to place focus on the element.

```
var  objDiv = document.getElementById(‘divFocus’);
//focus
objDiv.focus();
```

4. ARIA States and Properties

ARIA states and properties allow information about the widget to be provided to assistive technology to help the user understand how to interact with the widget. The state identifies a unique configuration of information for an object.The various aria-properties are shown below:

1. *aria-valuemin:* Stores the lowest value a range may have.
2. *aria-valuemax:* Stores the highest value a range may have.
3. *aria-valuenow:* Stores the current value in a range.
4. *aria-valuetext:* Stores readable text to help the user understand the context. E.g., "42 percent".
5. *aria-labelledby:* Stores the id attribute of a text label containing an appropriate prompt for this widget.

It also includes aria-autocomplete, aria-checked (state), aria-disabled (state), aria-expanded (state),aria-has popup, aria-hidden (state), aria-invalid (state), aria-label, aria-level, aria-multiline, aria-multiselectable, aria-orientation, aria-pressed (state), aria-readonly, aria-required, aria-selected (state) and aria-sort.

5. WAI-ARIA Roles

This specification categorizes roles that define user interface widgets, document structure and those that regions of the page. Roles are categorizes as follows:

5.1 Widget Roles

They act as a standalone user interface widgets or as part of larger composite widgets. Some of these are:

1. *Slider:* A user input where the user selects a value from within a given range. A slider represents the current value and range of possible values via the size of the slider and position of the thumb.
2. *Status:* A container whose content is advisory information for the user but is not important enough to justify an alert, often but not necessarily presented as a status bar.

Widget roles also includes dialog, alert, alert dialog, gridcell, link, marquee, menuitem, menuitemcheckbox, menuitemradio, option, progressbar, radio, scrollbar, button, log, spinbutton, tab, tabpanel, textbox, timer, tooltip and treeitem

5.2 *Document Structure Roles*

The following roles describe structures that organize content in a page. Document structures are not usually interactive. Some of these are explained as under

1. *Article:* A section of a page that consists of a composition that forms an independent part of a document, page, or site.

2. *Columnheader:* A cell containing header information for a column. Column header can be used as a column header in a table or grid.
3. *Row:* Rows contain grid cell elements and thus serve to organize the grid. Authors MUST ensure elements with role row are contained in, or owned by, an element with the role grid, row group and tree grid.

Document Structure roles also include definition, directory, document, region, rowheader, group, heading, img, list, listitem, math, note, presentation, separator and toolbar.

5.3 Landmark Roles

The following roles are regions of the page intended as navigational landmarks. The roles are included here in order to make them clearly part of the WAI-ARIA Role taxonomy.

1. *Banner:* A region that contains mostly site-orientated content, such as the title of the page and the logo.
2. *Main:* This marks the content that is directly related to or expands on the central content of the document.
3. *Navigation:* A collection of navigational elements (usually links) to navigate the document and/or related documents.
4. *Search:* A region that contains a collection of items and objects that, as a whole, combines to create a search facility. This section contains a search form to search the site.

```
<div role="banner">
</div>
<div id="nav" role="navigation">
<form id="form1"role="search"...>
</form>
</div>
<div id="content" role="main">
</div>
```

Landmark Roles also include Application, Complementary, Contentinfo and form.

6. WAI- ARIA Live Regions

Live regions allow elements in the document to be announced if there are changes, without the user losing focus on their current activity. This means users can be informed of updates without losing their place within the content.

6.1 The aria-live Property

The discoverability of updated content is one of the biggest obstacles for screen reader users. ARIA provides an aria-live property that has a value indicating the verbosity level for the region. They are ***off*** (default value, and changes will not be announced), ***polite*** (updates are announced at the next graceful interval i.e. when the user stops typing) and ***assertive*** (when the update is important and the user should be informed immediately)

6.2 The aria-atomic Property

The aria-atomic property is an optional property of live regions having values true or false (default). When the region is updated, the aria-atomic property is used to indicate if assistive technology should present all or part of the changed region to the user. If this property is set to true, assistive technology should present the entire region as a whole
In the following example, if a change is made anywhere in the div element, the whole content is announced to the user.

```
<div aria-atomic="true" aria-live="polite">
<h4>Currently Working On<h4>
<p>Web Accessibility Initiative – Accessible Rich Internet Application (WAI-ARIA)
</p>
</div>
```

6.3 The aria-busy Property

The aria-busy property should be used if multiple parts of a live region need to be loaded before changes are announced to the user, the aria-busy property can be set to true until the final part is loaded, and then set to false when the updates are complete. This property prevents assistive technologies announcing changes before the updates are complete.

```
<ol aria-busy="true" aria-live="polite">
<li></li>
</ol>
```

6.4 The aria-relevant Property

The aria-relevant property is an optional property of live regions that indicates what changes are considered relevant within a region. It indicates the type of update that should be announced in a region. They are ***Additions (a***nnounces when the elements are added to the DOM within the region), ***Removals*** (when the elements are removed), ***Text*** (when the text is added or removed) and A***ll: (***All of the above). In the absence of an explicit aria-relevant property, the default is to assume there are text changes and additions (aria-relevant="text additions").

7. Conclusion

There are no negative side effects from using ARIA. Often, script-based customization for assistive technologies in the workplace tackles exactly those custom controls that WAI-ARIA could make accessible. Better support for WAI-ARIA may save much time and cost. This, however, presupposes that in creating web or intranet applications, developers use WAI-ARIA consistently and according to best practices. Some of the additional benefits of web accessibility are listed below:

1. ***Search engine optimization***: SEO is something companies don't usually hesitate to spend money on. And yet many accessibility guidelines are the same as SEO techniques, for example valid HTML, clear link names, using text rather than images of text, descriptive 'title' tags, providing text equivalents for multimedia, creating a site map, etc. This means that incorporating accessibility will at the same time help to improve websites' search engines ranking;
2. ***Increased usability:*** In general, accessibility increases usability of a website, and in effect improves quality of user experience. As accessible websites are easier to find, access and use, they maximize the number of possible visitors. Some accessibility guidelines are similar to the usability ones, such as promoting clear and consistent design and navigation, dividing blocks of information into logical sections, etc.
3. ***Reduced site development and maintenance time:*** Although incorporating accessibility can increase site development time initially, in long term it reduces time spent on site improvements and maintenance. Using style sheets and coding to standards reduces effort needed to change presentation across a site.
4. ***Reputation:*** Last but not least, a company's efforts towards making their website accessible can have a positive impact on company's reputation, as it creates an image of ethical and socially responsible organization.

Acknowledgements:
We express our deep thanks to Mr. B.K. Murthy, Executive Director; CDAC Noida (India) with the help of his vast experience, efforts, valuable suggestion, support and motivation helped us to great extent for the design and development of this utility.

References

[1] http://www.w3.org/TR/wai-aria/
[2] http://msdn.microsoft.com/en-us/magazine/ff743762.aspx