Quick Developers Guide for ARIA

(Accessible Rich Internet Applications)

Version Log

Version	Description	Author	Date
0.1	First Draft	Kandagatla, Sravan K	02/24/2014
0.2	Added Mandatory column with proof of concepts and added Sources and References	Kandagatla, Sravan K	03/12/2014
0.3	Added Quick Tips	Kandagatla, Sravan K	04/08/2014
0.4	Enhanced the doc with valid suggestions and comments	Tyler, Bill	04/09/2014
0.5	Enhanced further by moving non-mandatory and deprecated items. Updated the doc with few more examples, sources & polyfill and also added "say-instead" – A new CSS3 Speech Module property example.	White, Tim Tyler, Bill Kandagatla, Sravan K	04/10/2014
0.6	Removed duplicate records. Example on live update region. Explained the difference between Readonly and Disabled fields. Example on ordered or un-ordered list with structural role attributes.	Tyler, Bill Peter, Stema Kandagatla, Sravan K Kandagatla, Sravan K	04/23/2014

Item / Tag	**Mandatory?**	Accessible with Strong **Native** and **ARIA** Semantics	Sample Code
Body <body	Situational	role="document" role can be either: document or application	<body role=”application”>
Section <section	Situational	role="region" role can be either: region, application, log, contentinfo, document, main, marquee, search, or status	<section role=”main”>
Header <header	Yes	role="banner" this role should be placed only once for main header and should be applied for only one tag in a page.	<header role=”banner”> or
Navigation <nav	Yes	role="navigation"	<nav role=”navigation”>
Range <input type="range"	Yes	role="slider" max="100" aria-valuemax="100" min="50"	<input type=”range” role="slider" max="100" aria-valuemax="100" min="50"

		aria-valuemin="50" value="55" aria-valuenow="55"	aria-valuemin="50" value="55" aria-valuenow="55" />
Telephone <input type="tel"	Situational	role="textbox"	<input type=”tel” role=”textbox” />
Text <input type="text"	Situational	role="textbox"	<input type=”text” role=”textbox” />
URL <input type="url"	Situational	role="textbox"	<input type=”url” role=”textbox” />
Required Field <input type="text" required	Yes	aria-required="true"	<input type=”text” required aria-required=”true”>
Progress <progress	Yes	role="progressbar" aria-valuemax="100" min="0" aria-valuemin="0" value="50" aria-valuenow="50"	<progress max=”100” role="progressbar" aria-valuemax="100" min="0" aria-valuemin="0" value="50" aria-valuenow="50"
Image <img	Yes	role="img" //only if there is no alt attribute	
Figure <figure	Yes	role="img" aria-labelledby=" "	<figure role="img" aria-labelledby="image-caption"> <figcaption id="image-caption"> You can describe the image or figure here in text </figcaption> </figure>

Article <article	Yes	role="article" role must be either: alert, alertdialog, dialog, article, document, application, presentation or main	<article role=”document”>
Aside <aside	Yes	role="complementary" role must be either: complementary, note, or search	<aside role=”search”>
Ordered List <ol	Situational	role="list" role can be either: list, directory, group, listbox, menu, menubar, tablist, toolbar, or tree	<ol role=”group”> Default (content): <ol role=”list”> Directory (document structure, example table of contents, not file directory): <ol role=”directory”> Group (document structure): <ol role=”group”> Listbox (container of options): <ol role=”listbox”> Menu (static, pop-up, pull-down, context): <ol role=”menu”> Menubar (top-level menu of menus): <ol role=”menubar”>

			Tab selection (list of tabs in tabbed dialog): <ol role=”tablist”> Toolbar (collection of controls, buttons): <ol role=”toolbar”> Tree (contains sub-tree and treeitems): <ol role=”tree”>
Unordered List <ul	Situational	role="list" role can be either: list, directory, group, listbox, menu, menubar, tablist, toolbar, or tree	<ul role=”group”>
List Item <li	Situational	role="listitem" role can be either: listitem, menuitemcheckbox, menuitemradio, option, tab, or treeitem	<li role=”menuitemradio”>
Main <main	Yes	role="main" role can be either: main, document, or	<main role=”main”>

		application	
Menu <menu type="popup"	Yes	role="toolbar" role can be either: directory, list, listbox, menu, menubar, tablist, toolbar, or tree	<menu type=”popup” role=”menu”>
Footer <footer	Yes	role="contentinfo" this role should be placed only once for main header, but not for the headers within the same page	<footer role=”contentinfo”>
Details <details	Yes	role="group"	<details role=”group”>
Dialog <dialog	Yes	role="dialog" aria-hidden="true"	<dialog role=”dialog” aria-hidden=”true”>
Hidden Any Tag	Situational	aria-hidden="true" //Can be applied to any element which you want to have it defined as hidden	<div role=”dialog” class=”hide” aria-hidden=”true” />
Read-only Any tag	Situational	aria-readonly="true" //only if the element has readonly attribute	<input readonly aria-readonly=”true” />

A big NO to the Equivalent ARIA Attributes

Item / Tag	**Mandatory?**	**Equivalent** - Native and ARIA Semantics
Multi Select Box <select multiple	No	multiple role="listbox" aria-multiselectable="true">
Single Select Box <select	No	role="listbox" aria-multiselectable="false" role can be either listbox or menu
Option Selected <option value="default"	No	selected aria-selected="true" aria-checked="true"
Data List <datalist	No	role="listbox" aria-multiselectable="false"
Checkbox <input type="checkbox"	No	role="checkbox" aria-checked="true" //If the checkbox is checked aria-checked="false" //If the checkbox is not checked aria-checked="mixed" //If the checkbox is partially checked
Date <input type="date"	No	aria-readonly="true" //only if the element has readonly attribute
Date & Time <input type="datetime"	No	aria-readonly="true" //only if the element has readonly attribute
Local Date & Time <input type="datetime-local"	No	aria-readonly="true" //only if the element has readonly attribute
Email <input type="email"	No	role="textbox" aria-readonly="true" //only if the element has readonly attribute

Number <input type="number"	No	role="spinbutton" max="100" aria-valuemax="100" min="50" aria-valuemin="50" value="55" aria-valuenow="55" aria-readonly="true" //only if the element has readonly attribute
Password <input type="password"	No	role="textbox" aria-readonly="true" //only if the element has readonly attribute
Radiobutton <input type="radio"	No	role="radio" aria-checked="true" //If the radio button is checked aria-checked="false" //If the radio button is not checked
Reset Button <input type="reset"	No	role="button"
Search <input type="search"	No	role="textbox" aria-readonly="true" //only if the element has readonly attribute
Submit Button <input type="submit"	No	role="button"
Text Area <textarea	No	role="textbox" aria-multiline="true" aria-readonly="true" //only if the element has readonly attribute
Template <template	No	aria-hidden="true"
Button <button	No	role="button" role can be either button, menuitem
Embed <embed	No	role="application" role can be either application, document, or img
Heading <h1 <h2 <h3 <h4 <h5 <h6	No	role="heading" aria-level="1" aria-level is the element's outline depth and role can be either heading or tab

Iframe <iframe	No	role="application" here role is not mandatory - if specified - it should be either application, document, or img
Object <object	No	role="application" if specified – role can be either application, document, or img
Video <video	No	role="application"
Table <table	No	role="grid" aria-readonly="true"
Table Body <tbody	No	role="rowgroup"
Table Cell <td	No	role="gridcell"
Table Row <tr	No	role="row"
Table Header Cell <th	No	role="gridcell" role can be either gridcell, columnheader, or rowheader
Table Header <thead	No	role="rowgroup"
Table Footer <tfoot	No	role="rowgroup"
Audio <audio	N/A	role="application"

Quick Accessibility Tips

Tip 1: Form fields with hints

```
1.     <label for="searchField" class="hide" aria-hidden="true">Search:</label>
2.     <input type="search" name="search" id="searchField" aria-describedby="hint" />
3.     <span id="hint">Search for a Physician, Facility, Treatment, or Condition</span>
```

If there are scenarios where we need to hide the main label but display a hint in UI. Then we can consider the above example for accessibility.

Observe class="hide" and aria-hidden="true" for Search: label and aria-describedby="hint" for hint.

Tip 2: Anchor tag / Hyperlink Accessibility

No need to declare role="link" for <a tag.
role="link" should be used only when you want other tags to define as a link for Screen Readers.

Title attribute should be mentioned only when the link text doesn't describe what the link is all about and what it does etc. for example:

```
1.  <a href=" " title="Delete Mr.X from your favorites">Delete This</a>
```

And if the link text is more descriptive like below - then no need to use Title attribute:

```
1.  <a href=" "> Delete Mr.X from your favorites</a>
```

Tip 3: Required Attribute for Form Inputs

If you are having a required / mandatory input field like:

```
1.  <input type="text" required>
```

To make the above field accessible friendly, it is must to have aria-required="true" as mentioned below:

```
2.  <input type="text" required aria-required="true">
```

Tip 4: “say-instead” – A new CSS3 Speech Module property

There is a new CSS3 property called "say-instead" which allows screen readers to read the content defined in this property. Let's look into some example:

In CSS let's define a class like this:

```
1.  .additional-info{ say-instead="Jurassic Park Part Two"; }
```

In HTML let's refer this class as:

```
1.  <span class="additional-info">Jurassic Part II</span>
```

So the screen reader will read this span content as:

Jurassic Park Part Two - instead of Jurassic Park II

This example is picked from the nice and informative presentation on Accessibility at: Is Your Website Accessible?

Tip 5: Creating an Accessible Live Updatable Region

The role attribute

```
1.  <div role="document">
```

Identifies responsibility of an element to screen readers. Using role="document" identifies the region as content, as opposed to a web application.

The aria-live="polite" attribute

```
1.  <div aria-live="polite" aria-atomic="true">
```

Identifies responsibility of an element to screen readers to not interrupt the user's workflow.

The aria-live="assertive" attribute

```
1.  <div aria-live="assertive" aria-atomic="true">
```

Identifies responsibility of an element to screen readers to stop and begin reading the new content.

The aria-hidden attribute

```
1.  <div aria-hidden="true">
```

Identifies a region that a screen reader should ignore. Value can be true or false.

The aria-atomic attribute

```
1.  <div aria-live="polite" aria-atomic="true">
```

Identifies whether the entire contents of a live region should be read (aria- atomic="true"), or just the elements that changed should be read (aria- atomic="false").

http://www.w3.org/TR/wai-aria/states_and_properties

Hidden:
The parameter, aria-hidden="true". Identifies a region that a screen reader should ignore. Value can be true or false. An example might be a region that is hidden on initial page load. But is either shown or loaded when an event occurs.

Polite and Assertive Updating:
There are two ways for alerting the user to changes on the page when using aria-live. The polite method is designed to not interrupt the user's workflow. For example, if the user's screen reader is reading a sentence and another region of the page updates and the mode is set to polite, then the screen reader will finish reading the current sentence. However, if the mode is set to assertive, then the updated region is considered high priority, and the screen reader will stop and begin reading the new content. It's really important that you use the appropriate type of interruption when you're developing your site. Overuse of assertive can disorient and confuse your users. Use assertive only if you absolutely must. It may be the right choice you will be hiding other content regions.

Atomic Updating:
A second parameter, aria-atomic=true, instructs the screen reader to read the entire contents of the changed region. If we set it to false, it would tell the screen reader to read only nodes that changed. If you are replacing the entire contents telling the screen reader to read it all makes sense in this case. If we were replacing a single list item or appending to a table with Ajax, you may want to use false instead.

Some information on screen readers:
Home of the free NVDA screen reader (windows only)
Home of the JAWS screen reader
Built into the Macintosh and iOS (iPhone, iPod, iPad) operating systems. Similar functionality to JAWS
Screen reader Information by UofM

Tip 6: Difference between Readonly and Disabled fields

A "readonly" field is not editable and can be focused while tabbing, means screen readers can read the value/content within this readonly fields and value can be sent on form submit.

Whereas "disabled" field is not editable nor can be focused while tabbing, means screen readers cannot read the value/content within this disabled field and value will not be sent on form submit.

So use the appropriate attribute based on the above information.

Difference between readonly and disabled fields can be found in the following example at iCITA.

Tip 7: How to use or nest ordered or un-ordered list with structural role attributes

Role for ordered and un-ordered list can be used based on the type of functionality one wanted to achieve.

Example on “menuitemradio” for Single Select Menu:

```
<ul id="singleSelectMenu" class="menu" role="menu">
    <li id="optionOne"   class="menu-item checked" role="menuitemradio" aria-checked="true">Option One</li>
    <li id="optionTwo"   class="menu-item"         role="menuitemradio" aria-checked="false">Option Two</li>
    <li id="optionThree" class="menu-item"         role="menuitemradio" aria-checked="false">Option Three</li>
</ul>
```

Example on “menuitemcheckbox” for Multi Select Menu:

```
<ul id="multiSelectMenu"  class="menu" role="menu">
    <li id="optionOne"   class="menu-item checked" role="menuitemcheckbox" aria-checked="true">Option One</li>
    <li id="optionTwo"   class="menu-item checked" role="menuitemcheckbox" aria-checked="true">Option Two</li>
    <li id="optionThree" class="menu-item"         role="menuitemcheckbox" aria-checked="false">Option Three</li>
</ul>
```

Example on “tablist” for Tab Menu:

```
<ul id="tabMenu" class="tabmenu" role="tablist">
    <li id="tabOne" class="tab-item selected" role="tab" aria-controls="contentOne" aria-selected="true"
    tabindex="0">Tab One</li>
    <li id="tabTwo" class="tab-item" role="tab" aria-controls="contentTwo" aria-selected="false" tabindex="-1">Tab
    Two</li>
    <li id="tabThree" class="tab-item" role="tab" aria-controls="contentThree" aria-selected="false" tabindex="-1">Tab
    Three</li>
</ul>
<div id="contentOne"   class="content" role="tabpanel" aria-labelledby="tabOne">First Tab Content</div>
<div id="contentTwo"   class="content" role="tabpanel" aria-labelledby="tabTwo">Second Tab Content</div>
<div id="contentThree" class="content" role="tabpanel" aria-labelledby="tabThree">Third Tab Content</div>
```

Sources

- More about WAI-ARIA roles
- Using WAI-ARIA in HTML
- ARIA – Recommendations Table
- Accessibility Guide – Mozilla Developer Network
- WAI-ARIA States and Properties
- Native Semantics and ARIA Semantics
- HTML5 Accessibility

Further Reference

- http://oaa-accessibility.org/rules/
- http://msdn.microsoft.com/en-us/library/ie/hh801958%28v=vs.85%29.aspx
- https://www.utexas.edu/learn/accessibility/testing.html
- http://www.w3.org/TR/WCAG20/
- http://msdn.microsoft.com/en-us/magazine/jj863135.aspx
- http://www.w3.org/TR/wai-aria/
- http://alistapart.com/article/the-accessibility-of-wai-aria/
- Section 508
- http://www.w3.org/WAI/
- http://www.w3.org/TR/WAI-WEBCONTENT/

Polyfill

- AccessifyHTML5.js