

WAI-ARIA Best Practices

콘텐츠연합플랫폼
클라이언트개발부 지성봉

What Is WAI-ARIA

WAI

Web Accessibility Initiative

ARIA

Accessible Rich Internet Application



Accessible Rich Internet Applications (WAI-ARIA) 1.0

W3C Recommendation 20 March 2014

This version:

<http://www.w3.org/TR/2014/REC-wai-aria-20140320/>

Latest version:

<http://www.w3.org/TR/wai-aria/>

Previous version:

<http://www.w3.org/TR/2014/PR-wai-aria-20140206/>

Editors:

James Craig, Apple Inc.
Michael Cooper, W3C

Previous Editors:

Lisa Pappas, Society for Technical Communication
Rich Schwerdtfeger, IBM
Lisa Seeman, UB Access

Please check the [errata](#) for any errors or issues reported since publication.

This document is also available as a [single page](#) version.

See also [translations](#).

Copyright © 2008-2014 W3C® (MIT, ERCIM, Keio, Beihang), All Rights Reserved. W3C [liability](#), [trademark](#) and [document use](#) rules apply.

Abstract

#

Accessibility of web content requires semantic information about widgets, structures, and behaviors, in order to allow assistive technologies to convey appropriate information to persons with disabilities. This specification provides an ontology of roles, states, and properties that define accessible user interface elements and can be used to improve the accessibility and interoperability of web content and applications. These semantics are designed to allow an author to properly convey user interface behaviors and structural information to assistive technologies in document-level markup. This document is part of the WAI-ARIA suite described in the [WAI-ARIA Overview](#).

TABLE OF CONTENTS

1. **Introduction**
 - 1.1 Rich Internet Application Accessibility
 - 1.2 Target Audience
 - 1.3 User Agent Support
 - 1.4 Co-Evolution of `WAI-ARIA` and Host Languages
 - 1.5 Authoring Practices
 - 1.5.1 Authoring Tools
 - 1.5.2 Testing Practices and Tools
 - 1.6 Assistive Technologies
2. **Using `WAI-ARIA`**
 - 2.1 `WAI-ARIA` Roles
 - 2.2 `WAI-ARIA` States and Properties
 - 2.3 Managing Focus
3. **Conformance**
 - 3.1 Non-interference with the Host Language
 - 3.2 All `WAI-ARIA` in `DOM`
 - 3.3 Assistive Technology Notifications Communicated to Web Applications
 - 3.4 Conformance Checkers
 - 3.5 Deprecated Requirements
4. **Important Terms**
5. **The Roles Model**
 - 5.1 Relationships Between Concepts
 - 5.1.1 Superclass Role
 - 5.1.2 Subclass Roles
 - 5.1.3 Related Concepts
 - 5.1.4 Base Concept
 - 5.2 Characteristics of Roles
 - 5.2.1 Abstract Roles
 - 5.2.2 Required States and Properties
 - 5.2.3 Supported States and Properties

Accessible Rich Internet Applications (`WAI-ARIA`) 1.1



W3C Proposed Recommendation 02 November 2017

This version:

<https://www.w3.org/TR/2017/PR-wai-aria-1.1-20171102/>

Latest published version:

<https://www.w3.org/TR/wai-aria-1.1/>

Latest editor's draft:

<https://rawgit.com/w3c/aria/master/aria/aria.html>

Implementation report:

<https://w3c.github.io/test-results/wai-aria/>

Previous version:

<https://www.w3.org/TR/2016/CR-wai-aria-1.1-20161027/>

Latest Recommendation:

<https://www.w3.org/TR/2014/REC-wai-aria-20140320/>

Editors:

[Joanmarie Diggs](mailto:joanmarie@igalia.com), Igalia, S.L., jdiggs@igalia.com

[Shane McCarron](mailto:shane@spec-ops.io), Spec-Ops, shane@spec-ops.io

[Michael Cooper](mailto:cooper@w3.org), W3C, cooper@w3.org

Richard Schwerdtfeger, IBM Corporation, schwer@us.ibm.com (until October 2017)

[James Craig](mailto:jcraig@apple.com), Apple Inc., jcraig@apple.com (until May 2016)

[Copyright](#) © 2013-2017 [W3C](#)® ([MIT](#), [ERCIM](#), [Keio](#), [Beihang](#)). [W3C](#) liability, trademark and [permissive document license](#) rules apply.

Abstract

Accessibility of web content requires semantic information about widgets, structures, and behaviors, in order to allow assistive technologies to convey appropriate information to persons with disabilities. This specification provides an ontology of roles, states, and properties that define accessible user interface elements and can be used to improve the accessibility and interoperability of web content and

Why Use?

**Supplement For
Native Language Semantics,
Not A Replacement**

Role, Property, State

Role

Attaching a role gives
assistive technologies information
about how to handle each element

```
<tag role="keyword">
```

```
<div role="navigation">
```

list of role keywords

<https://www.w3.org/TR/wai-aria/#x5-4-definition-of-roles>

Property, State

Both provide specific information about an object,
and both form part of the definition
of the nature of roles.

How To Improve A11y With WAI-ARIA

Step 1

use native HTML

Then, When Use WAI-ARIA?

- the feature is available in HTML but it is not implemented or it is implemented, but accessibility support is not.
- visual design constraints rule out the use of a particular active element
- feature is not currently available in HTML

Step 2

Add ARIA

Inline Or Via Script?

Step 3

developing keyboard interface

Requirements

- All interactive ARIA controls must be usable with the keyboard.
- If can click or tap or drag or drop or slide or scroll, must also be able to perform an equivalent action using the keyboard
- All interactive widgets must be scripted to respond to standard key strokes or key stroke combinations where applicable

Best Practices

Landmark

Skip To

Main Content

Nav

Page Outline

h1: ARIA Landmarks Example

h1: Banner Landmark

h2: Design Patterns

h2: Landmarks

h2: Related W3C Documents

Complementary

Contentinfo

Form

Main

Navigation

Region

Search

Asst. Tech.

Resources

ARIA Landmarks Example

... outline the landmarks and/or headings on the page using the following buttons.

Landmarks

Show Headings

Banner Landmark

A **banner** landmark identifies site-oriented content at the top of each page within a website. Site-oriented content typically includes things such as the logo or name of the site sponsor, and site-specific search tool.

A banner usually appears at the top of the page and typically spans the full width.

ARIA 1.1 Specification: [banner](#), [landmark](#).

Design Patterns

- Each page may have one **banner** landmark.
- The **banner** landmark should be a top-level landmark.
- When a page contains nested **document** and/or **application** roles (e.g. typically through the use of **iframe** and **frame** elements), each **document** or **application** role may have one **banner** landmark.
- If a page includes more than one **banner** landmark, each should have a unique label.

ARIA Techniques

HTML5 Techniques

A **role="banner"** attribute is used to define a **banner** landmark.

ARIA Example

```
<div role="banner">
  <h1>page title identifying
  website</h1>
  ... banner content...
</div>
```

Landmarks

The following are landmarks defined on the page:

- Banner
- Complementary
- Contentinfo
- Main
- Navigation
- Region

Related W3C Documents

- [WAI-ARIA Authoring Practices 1.1](#)
- [WAI-ARIA 1.0 Specification](#)
- [WAI-ARIA 1.1 Specification](#)
- [Accessible Name and Description: Computation and API Mappings 1.1](#)
- [Core Accessibility API Mappings 1.1](#)
- [HTML Accessibility API Mappings 1.0](#)
- [HTML5 Specification](#)
- [ARIA in HTML](#)
- [Using ARIA in HTML](#)
- [WCAG Specification](#)

Legacy

```
<div class="header">
  <h1>ARIA Landmarks Example</h1>
</div>
<div class="navigation">
  <ul><li><a>...</a></li>...</ul>
</div>
<div class="main">
  <h2>Banner Landmark</h2>
  <div class="tab-container">
    ...
  </div>
</div>
<div class="sidebar">
  <h2>Landmarks</h2>
</div>
<div class="sidebar">
  <h2>Related W3C Documents</h2>
</div>
<div class="footer">
  Copyright
</div>
```

Print	navigation
HTML5	
Banner	
Complementary	
Contentinfo	
Form	
Main	
Navigation	
Region	
Search	
Asst. Tech.	
Resources	

Banner Landmark

main

A **banner** landmark identifies site-oriented content at the beginning of each page within a website. Site-oriented content typically includes things such as the logo or identity of the site sponsor, and site-specific search tool. A banner usually appears at the top of the page and typically spans the full width.

ARIA 1.1 Specification: [banner landmark](#).

Design Patterns

- Each page may have one **banner** landmark.
- The **banner** landmark should be a top-level landmark.
- When a page contains nested **document** and/or **application** roles (e.g. typically through the use of **iframe** and **frame** elements), each **document** or **application** role may have one **banner** landmark.
- If a page includes more than one **banner** landmark, each should have a unique label.

region

A `role="banner"` attribute is used to define a **banner** landmark.

ARIA Example

```
<div role="banner">
  <h1>page title identifying
  website</h1>
  .... banner content....
</div>
```

complementary

landmarks defined on the page:

- Banner
- Complementary
- Contentinfo
- Main
- Navigation
- Region

complementary

- [WAI-ARIA Authoring Practices 1.1](#)
- [WAI-ARIA 1.0 Specification](#)
- [WAI-ARIA 1.1 Specification](#)
- [Accessible Name and Description: Computation and API Mappings 1.1](#)
- [Core Accessibility API Mappings 1.1](#)
- [HTML Accessibility API Mappings 1.0](#)
- [HTML5 Specification](#)
- [ARIA in HTML](#)
- [Using ARIA in HTML](#)
- [WCAG Specification](#)

Use Native Language

```
<header>                                     <!-- banner landmark -->
  <h1>ARIA Landmarks Example</h1>
</header>
<nav>                                         <!-- navigation landmark -->
  <ul>...</ul>
</nav>
<main>                                        <!-- main landmark -->
  <h2>Banner Landmark</h2>
  <section>                                   <!-- region landmark -->
    ...
  </section>
</main>
<aside>                                       <!-- complementary landmark -->
  <h2>Landmarks</h2>
</aside>
<aside>                                       <!-- complementary landmark -->
  <h2>Related W3C Documents</h2>
</aside>
<footer>                                       <!-- contentinfo landmark -->
  Copyright
</footer>
```

Use ARIA Techniques

```
<div class="header" role="banner" >
  <h1>ARIA Landmarks Example</h1>
</div>
<div class="navigation" role="navigation" >
  <ul><li><a>...</a></li>...</ul>
</div>
<div class="main" role="main" >
  <h2>Banner Landmark</h2>
  <div class="tab-container" role="region" aria-label="Coding Tequniques" >
    ...
  </div>
</div>
<div class="sidebar" role="complementary" aria-labelledby="id3" >
  <h2 id="id3">Landmarks</h2>
</div>
<div class="sidebar" role="complementary" aria-labelledby="id4" >
  <h2 id="id4">Related W3c Documents</h2>
</div>
<div class="footer" role="contentinfo" >
  Copyright
</div>
```

Tab Contents

HTML

CSS

Javascript

HTML은 하이퍼텍스트 마크업 언어(HyperText Markup Language) 라는 의미의 웹 페이지를 위한 마크업 언어이다.

Legacy

```
<div class="tab-menu">
  <a href="#tab-panel1">HTML</a>
  <a href="#tab-panel2">CSS</a>
  <a href="#tab-panel3">JavaScript</a>
</div>
<div class="tab-panels">
  <div id="tab-panel1">
    <h3>HTML</h3>
    ...
  </div>
  <div id="tab-panel2">
    <h3>CSS</h3>
    ...
  </div>
  <div id="tab-panel3">
    <h3>JavaScript</h3>
    ...
  </div>
</div>
```

Use ARIA Techniques - Add Roles

```
<div class="tab-menu" role="tablist" >
  <a href="#tab-panel1" role="tab" >HTML</a>
  <a href="#tab-panel2" role="tab" >CSS</a>
  <a href="#tab-panel3" role="tab" >JavaScript</a>
</div>
<div class="tab-panels">
  <div id="tab-panel1" role="tabpanel" >
    ...
  </div>
  <div id="tab-panel2" role="tabpanel" >
    ...
  </div>
  <div id="tab-panel3" role="tabpanel" >
    ...
  </div>
</div>
```

Use ARIA Techniques - Add Properties, States

```
<div class="tab-menu" role="tablist">
  <a id="tab1" href="#tab-panel1" role="tab"
    aria-controls="tab-panel1" aria-selected="true">HTML</a>
  <a id="tab2" href="#tab-panel2" role="tab"
    aria-controls="tab-panel2" aria-selected="false">CSS</a>
  <a id="tab3" href="#tab-panel3" role="tab"
    aria-controls="tab-panel3" aria-selected="false">JavaScript</a>
</div>
<div class="tab-panels">
  <div id="tab-panel1" role="tabpanel"
    aria-labelledby="tab1">
    ...
  </div>
  <div id="tab-panel2" role="tabpanel"
    aria-labelledby="tab2" aria-hidden="true">
    ...
  </div>
  <div id="tab-panel3" role="tabpanel"
    aria-labelledby="tab3" aria-hidden="true">
    ...
  </div>
</div>
```

Use ARIA Technique

- Developing Keyboard Interface

Keyboard Support

Key	Function
Tab	<ul style="list-style-type: none">• when focus moves into the tab list, place focus on active tab element• When the tab list contains the focus, moves focus to the next element in the tab sequence, which is the tabpanel element
Right Arrow	<ul style="list-style-type: none">• Moves focus to the next tab.• If focus is on the last tab, moves focus to the first tab.• Activates the newly focused tab
Left Arrow	<ul style="list-style-type: none">• Moves focus to the previous tab.• If focus is on the first tab, moves focus to the last tab.• Activates the newly focused tab
Home	Moves focus to the first tab and activates it
End	Moves focus to the last tab and activates it

Should I Implement It Myself?

Yes, If You Can.

TABLE OF CONTENTS

1.	Introduction
2.	Design Patterns and Widgets
2.1	Generally Applicable Keyboard Recommendations
2.2	Accordion (Sections With Show/Hide Functionality)
2.3	Alert
2.4	Alert and Message Dialogs
2.5	Breadcrumb
2.6	Button
2.7	Checkbox
2.8	Combo Box
2.9	Dialog (Modal)
2.10	Dialog (Non-Modal)
2.11	Disclosure (Show/Hide)
2.12	Feed
2.13	Grids : Interactive Tabular Data and Layout Containers
2.14	Link
2.15	Listbox
2.16	Menu or Menu bar
2.17	Menu Button
2.18	Radio Group
2.19	Slider
2.20	Slider (Multi-Thumb)
2.21	Spinbutton
2.22	Table
2.23	Tabs
2.24	Toolbar
2.25	Tooltip Widget
2.26	Tree View
2.27	Window Splitter

2.17 Menu Button

A menu button is a [button](#) that opens a [menu](#). It is often styled as a typical push button with a downward pointing arrow or triangle to hint that activating the button will display a menu.

Examples

- [Navigation Menu Button](#): A menu button made from an HTML `a` element that opens a menu of items that behave as links.
- [Action Menu Button Example Using `element.focus\(\)`](#): A menu button made from an HTML `button` element that opens a menu of actions or commands where focus in the menu is managed using `element.focus()`.
- [Action Menu Button Example Using `aria-activedescendant`](#): A button that opens a menu of actions or commands where focus in the menu is managed using `aria-activedescendant`.

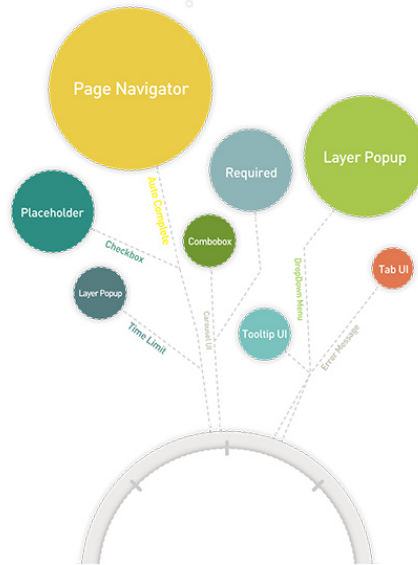
Keyboard Interaction

- With focus on the button:
 - `ENTER`: opens the menu and places focus on the first menu item.
 - `SPACE`: Opens the menu and places focus on the first menu item.
 - (Optional) `DOWN ARROW`: opens the menu and moves focus to the first menu item.
 - (Optional) `UP ARROW`: opens the menu and moves focus to the last menu item.
- The keyboard behaviors needed after the menu is open are described in [2.16 Menu or Menu bar](#).

WAI-ARIA Roles, States, and Properties

- The element that opens the menu has role `button`.
- The element with role `button` has `aria-haspopup` set to `true`.
- When the menu is displayed, the element with role `button` has `aria-expanded` set to `true`. When the menu is hidden, it is recommended that `aria-expanded` is not present. If `aria-expanded` is specified when the menu is hidden, it is set to `false`.
- The element that contains the menu items displayed by activating the button has role `menu`.
- Optionally, the element with role `button` has a value specified for `aria-controls` that refers to the element with role `menu`.
- Additional roles, states, and properties needed for the menu element are described in [2.16 Menu or Menu bar](#).

예제로 살펴보는 WAI-ARIA



미래창조과학부

NIA 한국정보화진흥원

WAI-ARIA 사례집

[http://www.wah.or.kr/_Upload/pds2/WAI-ARIA%20사례집\(온라인판\).pdf](http://www.wah.or.kr/_Upload/pds2/WAI-ARIA%20사례집(온라인판).pdf)

Otherwise

you can use jQueryUI, github, wai-aria 사례집, etc...



Your donations help fund the continued development and growth of jQuery.

SUPPORT THE PROJECT

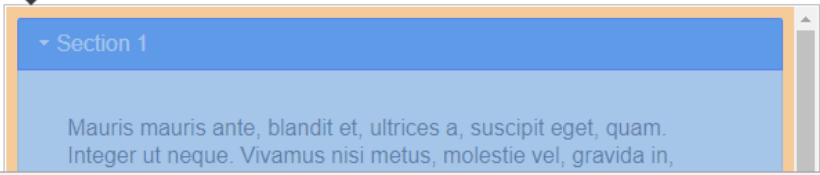
Search

Interactions

- Draggable
- Droppable
- Resizable
- Selectable
- Sortable

Accordion

Resizable content panels for presenting information in a limited amount of space.



Examples

- Default functionality
- Collapse content
- Customize icons

Elements Console Sources Network Memory Audits Performance Application Security

```
<html lang="en">
  <head>...</head>
  <body> == $0
    <div id="accordion" class="ui-accordion ui-widget ui-helper-reset" role="tablist">
      <h3 class="ui-accordion-header ui-corner-top ui-state-default ui-accordion-header-active ui-state-active ui-accordion-icons" role="tab" id="ui-id-1" aria-controls="ui-id-2" aria-selected="true" aria-expanded="true" tabindex="0">
        <span class="ui-accordion-header-icon ui-icon ui-icon-triangle-1-s"></span>
        "Section 1"
      </h3>
    </div>
  </body>
</html>
```

body 554 x 399

Styles Computed Event Listeners DOM Breakpoints

Filter :hov .cls +

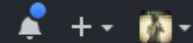
```
element.style {
}
body {
  font-family: Arial, Helvetica, sans-serif;
}
body {
  display: block;
  margin: 8px;
}
```

#content iframe html body div#accordion.ui-accordion.ui-widget.ui-helper-reset



accordion wai-aria

Pull requests Issues Marketplace Explore



Repositories **4** Code **108K** Commits **52** Issues **128** Wikis **43** Users

Advanced search

IIP-Design/a11y-accordion

JavaScript

jQuery *Accordion* that mostly complies with *WAI-Aria* 1.0 Authoring Practices.

Updated on 2 Sep 2016

Languages

JavaScript 4

AcceDe-Web/tablist

JavaScript

★ 8

WAI-ARIA accordion and tab plugin without dependencies

ISC license Updated on 16 Oct

NathanKleekamp/accessible-accordion

JavaScript

Accessible *accordion* that follows *WAI-ARIA* best practices

Updated on 6 Nov 2013

tariqkhan-co-uk/TabaKordion

JavaScript

★ 5

A fully accessible to *WAI* specification; tabs and *accordion* jQuery plugin. Makes use of *ARIA* and HTML data configurat...

MIT license Updated on 4 Feb 2016

This screenshot shows the GitHub repository page for 'niawa / ARIA'. The repository is on the '02. autocomplete' branch. The README content is as follows:

AutoComplete UI

사례집 수록 코드

“예제로 살펴보는 WAI-ARIA”에 실린 예제 코드 전문
쉬운 이해를 위해 절차지향적으로 작성

AutoComplete jQuery Plugin

검색어 자동완성기능 UI를 WAI-ARIA를 적용하여 jQuery 플러그인 형태로 제작

첨부된 데모는 DAUM API를 이용한 것으로, 일일 30,000콜의 제한이 있어 호출 횟수가 넘어갔을 시 추천 검색어가 나오지 않을 수 있습니다.

How to use

```
<form>
  <input type="text" id="test" />
</form>
```

```
$('#test').AutoCompleteUI({
  getSources : function(){
    var keyword = this.keyword;
    ...
    // setSources must be invoked with an array that is the suggested list related to keyword as argument
    this.setSources(source);
  }
});
```


References

- **WAI-ARIA Specification**
<https://www.w3.org/TR/wai-aria>
- **Using ARIA**
<https://www.w3.org/TR/using-aria>
- **WAI-ARIA Authoring Practices**
<https://www.w3.org/TR/wai-aria-practices>
- **HTML5 Accessibility**
<http://www.html5accessibility.com>

감사합니다.

`publisher@publisher.name`