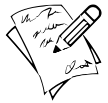
ARIA Foundations, Patterns and Components

Accessing Higher Ground 2021
with Dennis Lembrée

Agenda



- Introductions
- What is ARIA?
- Accessibility Tree
- Tools
- ARIA Ground Rules
- ARIA Mighty Rules
- General use cases
  - Landmarks
  - Labelling
  - Describing
  - Difference of 3 States
  - Presentation role
  - Current state
  - Live Regions
- Review
- Components
  - Disclosure
  - Modal Dialog
  - Tab Panel
- Quiz Time
- Resources
- Questions

This slide deck PDF:
https://bit.ly/AHGARIA

Introductions

About Me

Dennis E. Lembrée
Director of Accessibility, Diamond

Experience includes:
Deque Systems, eBay, PayPal, Blackberry, Ford, Google, a few startups

Twitter:

- @DennisL
- @WebAxe (first podcast on web accessibility)
- @EasyChirp (first accessible Twitter app)

Fun facts:

- Lived in Michigan, Florida, California, and now North Carolina
- I have 4 cats 😸

diamond

About You

What is:

- Your role?
- Level of ARIA?
- Your hometown?
- Your factoid?

Do you like stickers?

PREREQUISITE:
A basic understanding of HTML is recommended for this session.

What is ARIA?

Accessible Rich Internet Applications (WAI-ARIA)

- Code specification by W3C WAI (Web Accessibility Initiative)
  - w3.org/WAI
  - Other specs such as WCAG; training resources, testing resources, etc.
- ARIA 1.1 [current]
  - W3C Recommendation 14 December 2017
  - w3.org/TR/wai-aria-1.1
- ARIA 1.2
  - W3C Candidate Recommendation Snapshot 02 March 2021
  - w3.org/TR/wai-aria-1.2

W3C Recommendation

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
5.2.4 Inherited States and Properties
5.2.5 Required Owned Elements
5.2.6 Required Context Role
5.2.7 Accessible Name Calculation
5.2.7.1 Name Computation
5.2.7.2 Description Computation
5.2.7.3 Text Alternative Computation
5.2.7.4 Roles Supporting Name from Author
5.2.7.5 Roles Supporting Name from Content
5.2.8 Presentational Children
5.2.9 Implicit Value for Role
5.3 Categorization of Roles
5.3.1 Abstract Roles
5.3.2 Widget Roles
5.3.3 Document Structure
5.3.4 Landmark Roles
5.3.5 Live Region Roles
5.3.6 Window Roles
5.4 Definition of Roles

6. **Supported States and Properties**
6.1 Clarification of States versus Properties
6.2 Characteristics of States and Properties
6.2.1 Related Concepts
6.2.2 Used in Roles
6.2.3 Inherits into Roles
6.2.4 Value
6.3 Values for States and Properties
6.4 Global States and Properties
6.5 Taxonomy of WAI-ARIA States and Properties
6.5.1 Widget Attributes
6.5.2 Live Region Attributes
6.5.3 Drag-and-Drop Attributes

Accessible Rich Internet Applications (WAI-ARIA) 1.1

W3C Recommendation 14 December 2017

This version:
https://www.w3.org/TR/2017/REC-wai-aria-1.1-20171214/
Latest published version:
https://www.w3.org/TR/wai-aria-1.1/
Latest editor's draft:
https://w3c.github.io/aria/
Implementation report:
https://w3c.github.io/test-results/wai-aria/
Previous version:
https://www.w3.org/TR/2017/PR-wai-aria-1.1-20171102/
Previous Recommendation:
https://www.w3.org/TR/wai-aria-1.0/

Editors:
Joanmarie Diggs, Igalia, S.L., jdiggs@igalia.com
Shane McCarron, Spec-Ops, shane@spec-ops.io
Michael Cooper, W3C, cooper@w3.org
Richard Schwerdtfeger, IBM Corporation, schwer@us.ibm.com (until October 2017)
James Craig, Apple Inc., jcraig@apple.com (until May 2016)

Please check the **errata** for any errors or issues reported since publication.

See also **translations**.

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 version adds features new since WAI-ARIA 1.0 [wai-aria-1.0] to improve interoperability with assistive technologies to form a more consistent accessibility model for [html5] and [SVG2]. This specification complements both [html5] and [SVG2].

This document is part of the WAI-ARIA suite described in the WAI-ARIA Overview.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at https://www.w3.org/TR/.

This is the WAI-ARIA 1.1 W3C Recommendation by the Accessible Rich Internet Applications Working Group. The Working Group created a WAI-ARIA 1.1 Implementation Report to demonstrate that the specification is implementable. A history of changes to WAI-ARIA 1.1 is available in the appendix.

To comment on this document, file an issue in the W3C aria GitHub repository. If this is not feasible, send email to public-aria@w3.org (comment archive). Comments received on the WAI-ARIA 1.1 Recommendation cannot result in changes to this version of the specification, but may be addressed in errata or future versions of WAI-ARIA. The Working Group may not make formal responses to comments but future work undertaken by the Working Group may address comments received on this document. In-progress updates to the technology may be viewed in the publicly visible editors' draft.

This document was published by the Accessible Rich Internet Applications Working Group as a Recommendation.

Please see the Working Group's implementation report.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

This document was produced by a group operating under the W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

This document is governed by the 1 March 2017 W3C Process Document.

1. Introduction §

This section is non-normative.

The goals of this specification include:

- expanding the accessibility information that may be supplied by the author;
- requiring that supporting host languages provide full keyboard support that may be implemented in a device-independent way, for example, by telephones, handheld devices, e-book readers, and televisions;

What ARIA is

- A large set of **attributes** to add semantics
  - For HTML but could be SVG, etc.
  - Fixes communication gaps between code and assistive technology (AT)
- Roles
  - Widget, Document, Landmark
  - role attribute
- States and Properties
  - Widget, Live Region, Relationship
  - aria-* attribute

```
role=”tabpanel” aria-selected=”true”
```

ARIA attribute examples

Roles

- Widget:
  - button, dialog, menu, radio, tab, grid
- Document:
  - list, img, table, presentation, row, tooltip
- Landmark:
  - banner, main, navigation, contentinfo, region, search
- Live Region:
  - alert, log, status
- Window
  - dialog, alertdialog

States and Properties

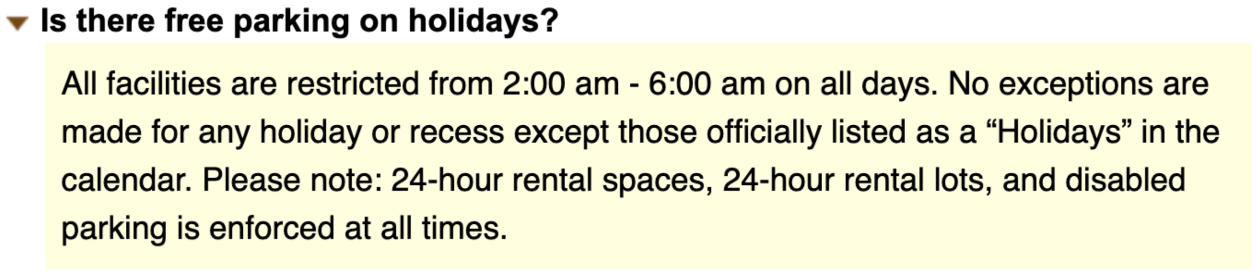
- Widget:
  - aria-checked, aria-expanded, aria-haspopup, aria-label, aria-readonly, aria-required, aria-modal, aria-pressed
- Live region:
  - aria-live, aria-atomic, aria-busy
- Relationship:
  - aria-describedby, aria-labelledby, aria-rowcount, aria-setsize, aria-controls

What ARIA is not

- ARIA doesn’t change visuals
- ARIA doesn’t “do” anything (creates no functionality)
  - Still need to do scripting for interaction
- ARIA isn’t magic!
  - Still need good content, solid design, semantic code

ARIA Example 1

```
<button aria-expanded="true" aria-controls="faq3_desc">
  Is there free parking on holidays?
</button>
```

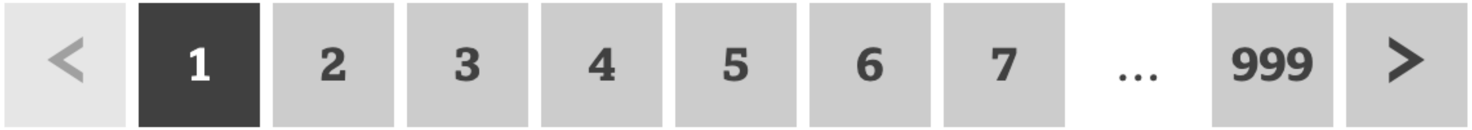


Is there free parking on holidays?

All facilities are restricted from 2:00 am - 6:00 am on all days. No exceptions are made for any holiday or recess except those officially listed as a “Holidays” in the calendar. Please note: 24-hour rental spaces, 24-hour rental lots, and disabled parking is enforced at all times.

ARIA Example 2

```
<div role=”navigation” class="gel-pages" aria-labelledby="gel-pages__label">
  <div id="gel-pages__label" hidden="">Page</div>
  <a class="gel-pages__prev">
    <span class="gel-sr">Previous page</span>
    <svg … aria-hidden="true">
  </a>
  <ol class="gel-pages__numbered">
    <li><a href="?page=1" aria-current="page">1</a></li>
    <li><a href="?page=2">2</a></li>
```



Summary: What Is ARIA?

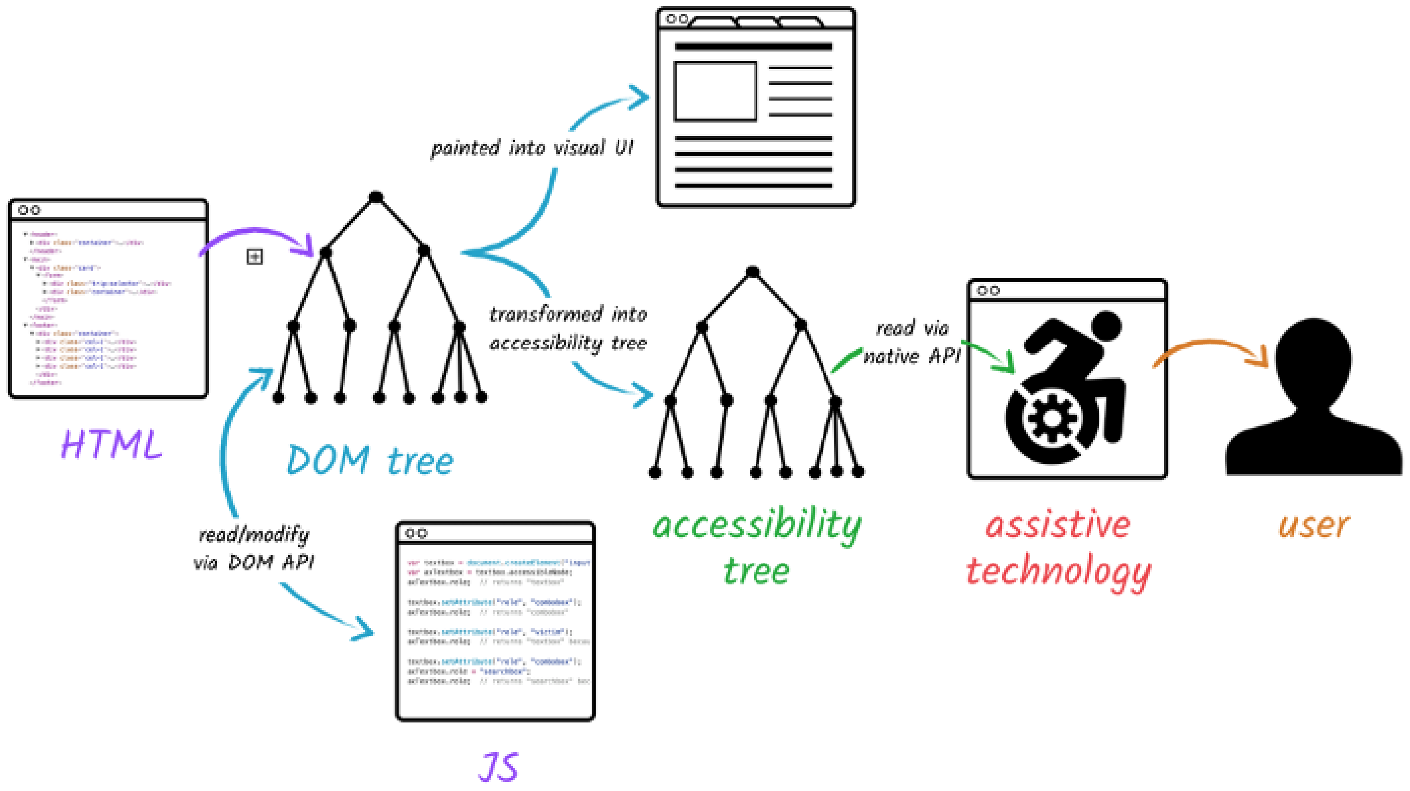
- Accessible Rich Internet Applications (WAI-ARIA), current version 1.1.
- ARIA is a large set of attributes to add semantics.
    - Roles, States, and Properties
- ARIA doesn’t add functionality (and is not magic!).

Accessibility Tree

About the Accessibility Tree 🌳

- Browsers convert webpage into a DOM tree.
- Browsers then create an Accessibility Tree based on the DOM tree.
- Accessibility Tree used by platform-specific Accessibility APIs to provide a representation that can be understood by assistive technologies.
    - SAA/UIA/IAccessible2 on Windows
    - OS X Accessibility Protocol on Mac and iOS
    - AT-SPI/IAccessible2 on Linux
- Accessibility tree contains:
    - Name, description, role, value, state, action
    - Covered in WCAG SC 4.1.2

Accessibility Tree Diagram



ARIA effects the Accessibility Tree

ARIA overrides HTML semantics.

So probably shouldn’t do this, because now there’s no heading:

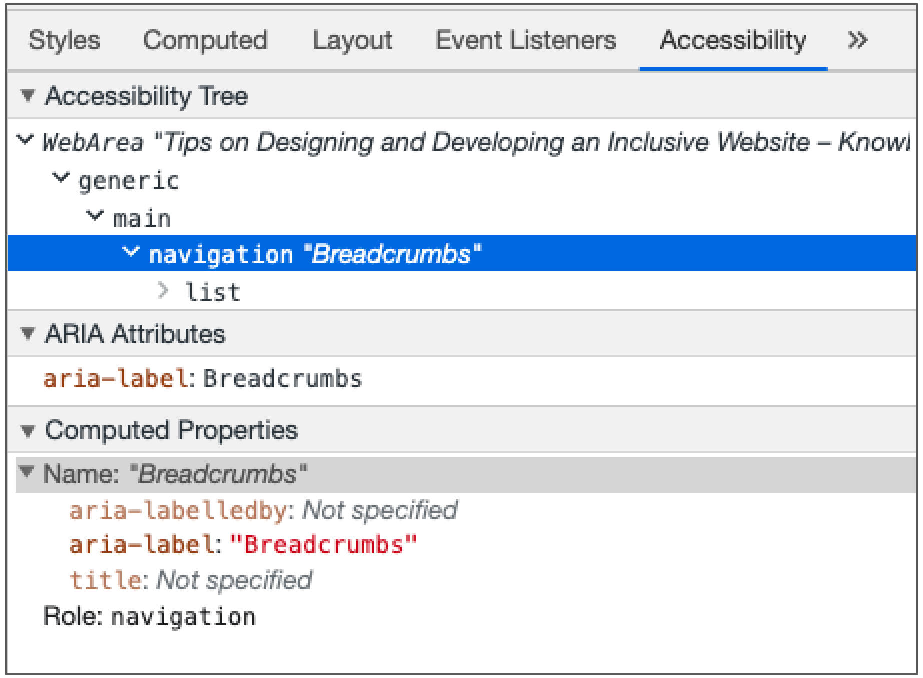
```
<h1 role=”link”>heading button</h1>
```

Do this instead:

```
<h1><a href=”foo”>heading button</a></h1>
```

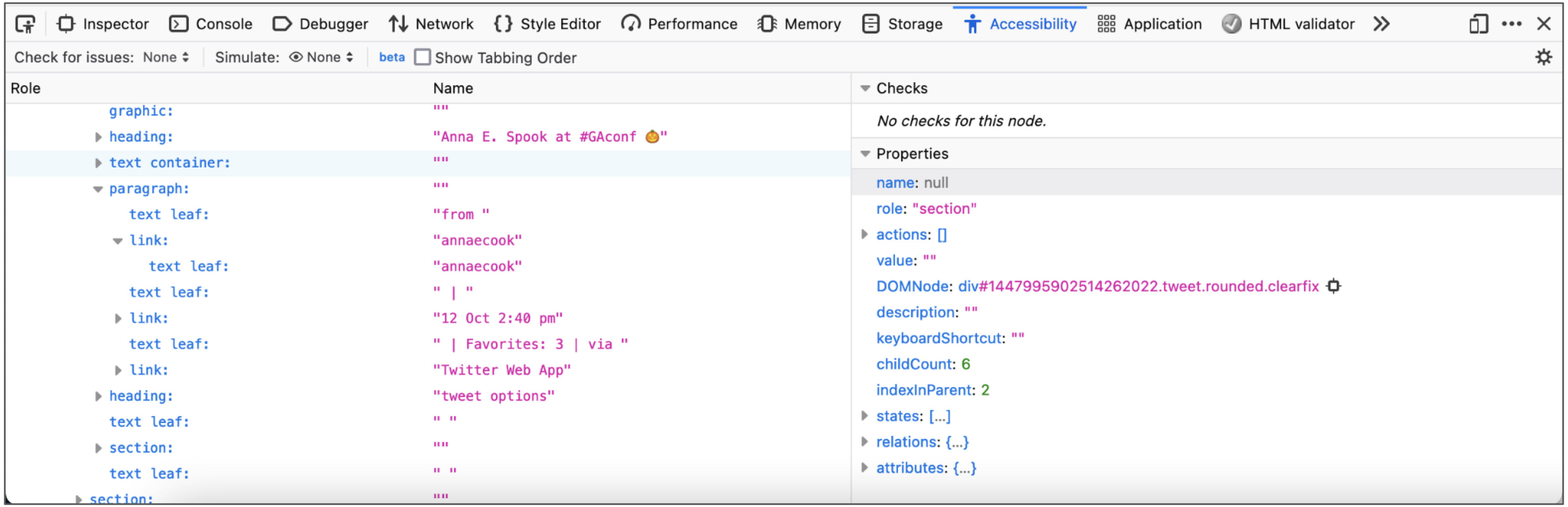
Viewing the Accessibility Tree

- Chrome and Firefox inspectors ☆
- Various browser extensions
- Windows: Active Accessibility Object Inspector
- OSX and iOS: Accessibility inspector via Xcode
- JAWS script called BX



```
▼ Accessibility Tree
⌄ WebArea "Tips on Designing and Developing an Inclusive Website – Knowl
  ⌄ generic
    ⌄ main
      ⌄ navigation "Breadcrumbs"
        › list
▼ ARIA Attributes
  aria-label: Breadcrumbs
▼ Computed Properties
▼ Name: "Breadcrumbs"
    aria-labelledby: Not specified
    aria-label: "Breadcrumbs"
    title: Not specified
  Role: navigation
```

Chrome and Firefox Exercise



Check for issues: None Simulate: None beta Show Tabbing Order

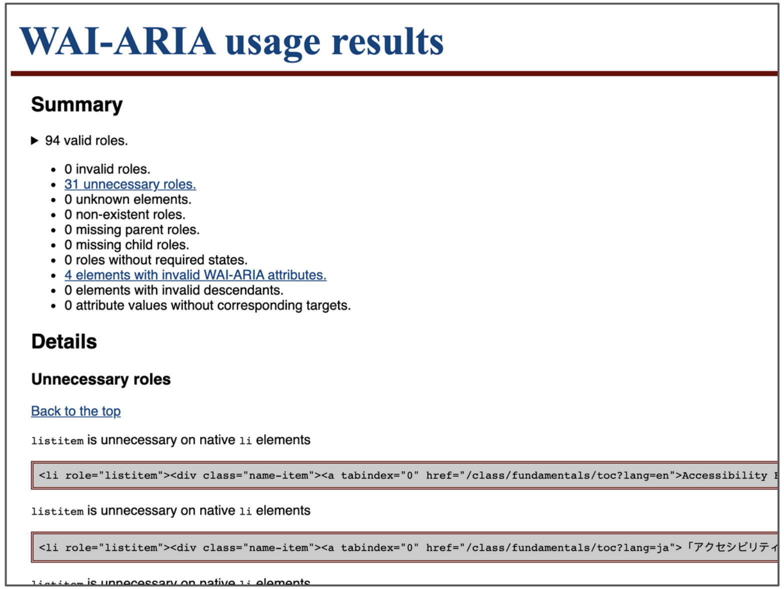
```
Role                        Name
    graphic:                ""
  ▸ heading:                "Anna E. Spook at #GAconf 🎃"
  ▸ text container:         ""
  ▾ paragraph:              ""
      text leaf:            "from "
    ▾ link:                 "annaecook"
        text leaf:          "annaecook"
      text leaf:            " | "
    ▸ link:                 "12 Oct 2:40 pm"
      text leaf:            " | Favorites: 3 | via "
    ▸ link:                 "Twitter Web App"
  ▸ heading:                "tweet options"
    text leaf:              " "
  ▸ section:                ""
    text leaf:              " "
▸ section:                  ""
```

```
▾ Checks
  No checks for this node.
▾ Properties
  name: null
  role: "section"
▸ actions: []
  value: ""
  DOMNode: div#1447995902514262022.tweet.rounded.clearfix
  description: ""
  keyboardShortcut: ""
  childCount: 6
  indexInParent: 2
▸ states: [...]
▸ relations: {...}
▸ attributes: {...}
```

Tools

ARIA Tools

- Chrome and Firefox inspectors (aforementioned)
- Automated testing
  - Axe, Tenon, WAVE, ARC
- WAI-ARIA usage bookmarklet
  - by TPGi/Gez Lemon
  - bit.ly/ariausage
- The Visual ARIA Bookmarklet
  - by Bryan Garaventa
  - bit.ly/visualaria



WAI-ARIA usage results

Summary

▶ 94 valid roles.

- 0 invalid roles.
- 31 unnecessary roles.
- 0 unknown elements.
- 0 non-existent roles.
- 0 missing parent roles.
- 0 missing child roles.
- 0 roles without required states.
- 4 elements with invalid WAI-ARIA attributes.
- 0 elements with invalid descendants.
- 0 attribute values without corresponding targets.

Details

Unnecessary roles

Back to the top

`listitem` is unnecessary on native `li` elements

```
<li role="listitem"><div class="name-item"><a tabindex="0" href="/class/fundamentals/toc?lang=en">Accessibility
```

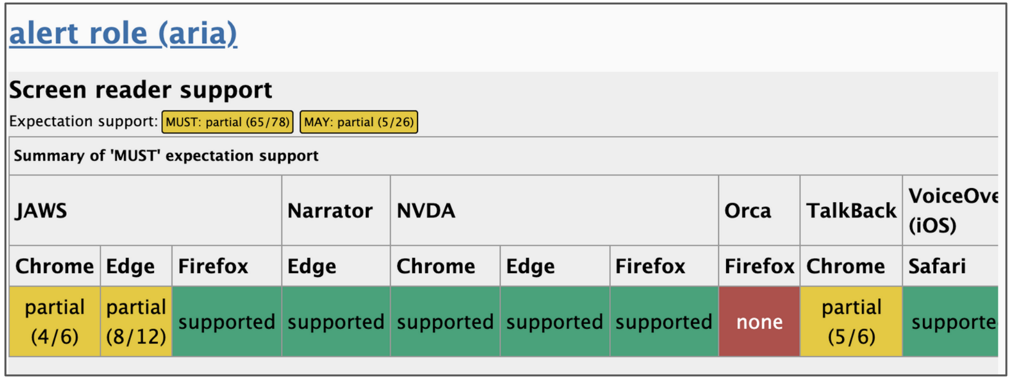
`listitem` is unnecessary on native `li` elements

```
<li role="listitem"><div class="name-item"><a tabindex="0" href="/class/fundamentals/toc?lang=ja">「アクセシビリティ
```

`listitem` is unnecessary on native `li` elements

ARIA Tools, cont.

- Accessibility Support
  - by Michael Fairchild
  - a11ysupport.io
- Nu Html Checker
  - validator.w3.org/nu/
- Landmarks browser add-on
  - by Matthew Atkinson
  - matatk.agrip.org.uk/landmarks/



alert role (aria)

Screen reader support

Expectation support: MUST: partial (65/78) MAY: partial (5/26)

Summary of 'MUST' expectation support

JAWS			Narrator	NVDA			Orca	TalkBack	VoiceOve (iOS)
Chrome	Edge	Firefox	Edge	Chrome	Edge	Firefox	Firefox	Chrome	Safari
partial (4/6)	partial (8/12)	supported	supported	supported	supported	supported	none	partial (5/6)	supporte

Summary: Tree & Tools

- Browsers create an Accessibility Tree (used by AT) from the DOM tree.
- ARIA change the Accessibility Tree
- There are great tools to help development, especially:
    - Chrome and Firefox inspectors
    - WAI-ARIA usage bookmarklet (by TPGi/Gez Lemon)
    - A11ySupport.io

ARIA Ground Rules

No ARIA is better than bad ARIA 🤔

If in doubt, leave it out.

Focus on:

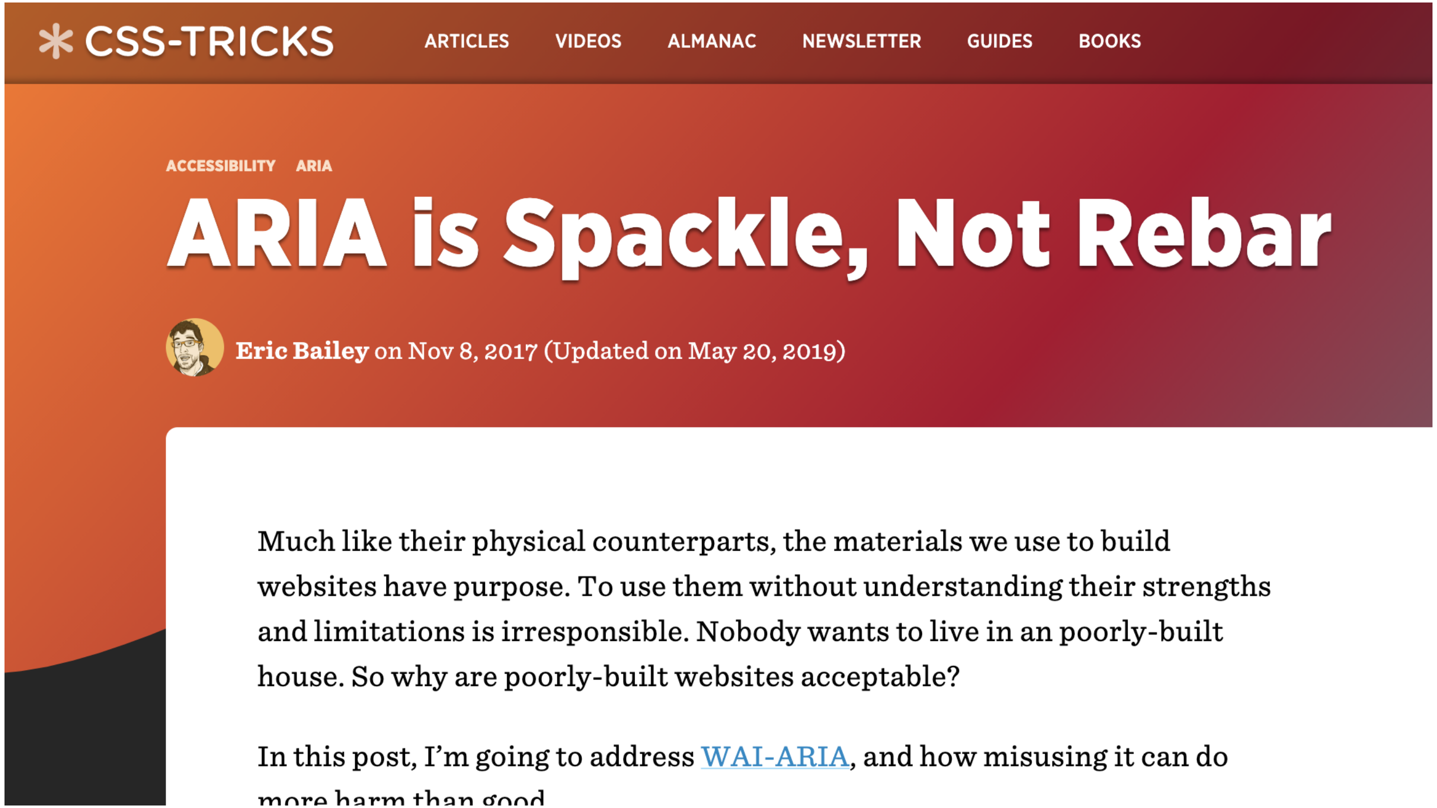
- HTML semantics
- Keyboard interaction (including focus management)

Caution ⚠

Use much caution/discretion with:

- `role=application`
- `role=form`
- `aria-hidden` and `role=presentation|none`
- Drag-and-drop
  - The ARIA specific to drag-and-drop is deprecated in version 1.2! (aria-dropeffect & aria-grabbed)
  - No standard code nor keyboard pattern

ARIA is Spackle, Not Rebar



CSS-TRICKS ARTICLES VIDEOS ALMANAC NEWSLETTER GUIDES BOOKS

ACCESSIBILITY ARIA

ARIA is Spackle, Not Rebar

Eric Bailey on Nov 8, 2017 (Updated on May 20, 2019)

Much like their physical counterparts, the materials we use to build websites have purpose. To use them without understanding their strengths and limitations is irresponsible. Nobody wants to live in an poorly-built house. So why are poorly-built websites acceptable?

In this post, I'm going to address WAI-ARIA, and how misusing it can do more harm than good.

WebAIM Million report says

“Pages with ARIA present have 27 more detectable errors (65% more errors) on average than pages without ARIA.”

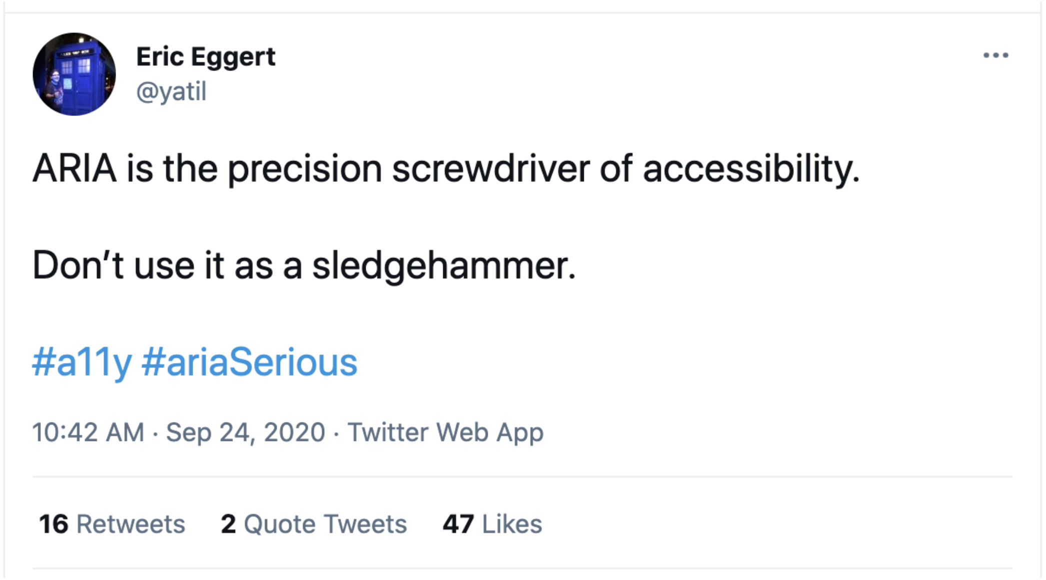
webaim.org/projects/million/update#aria

WebAIM
web accessibility in mind

ARIASerious?

ARIA Serious? talk by Eric Eggert: talks.yatil.net/5C4TU5

#ARIASerious on Twitter twitter.com/search?q=%23ARIASerious



Eric Eggert
@yatil

ARIA is the precision screwdriver of accessibility.

Don't use it as a sledgehammer.

#a11y #ariaSerious

10:42 AM · Sep 24, 2020 · Twitter Web App

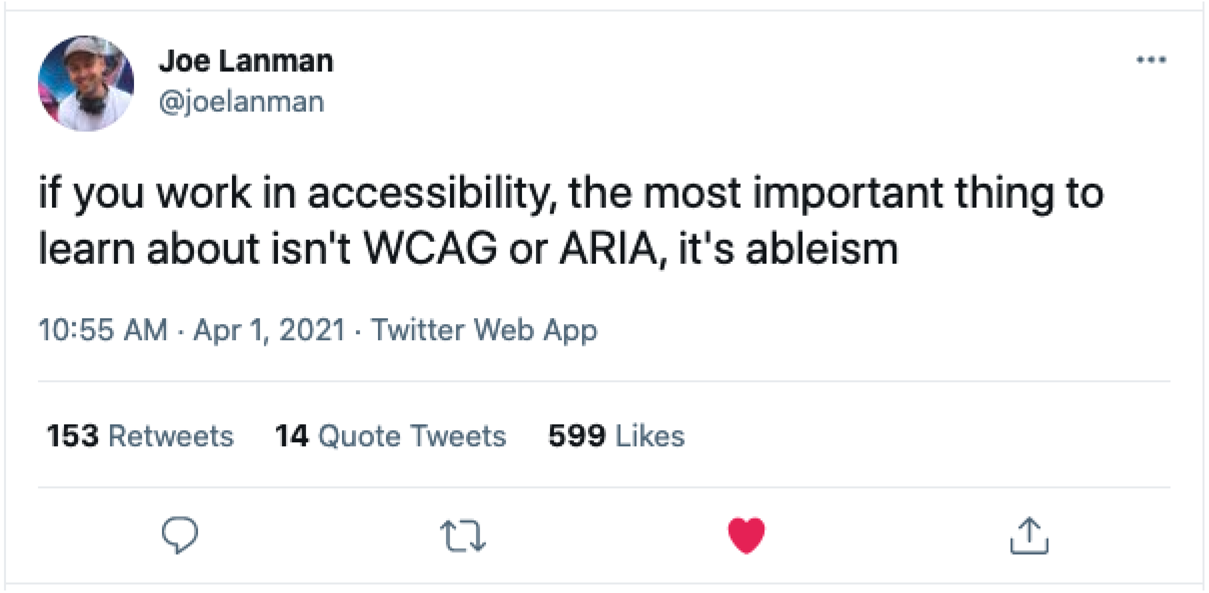
16 Retweets 2 Quote Tweets 47 Likes

Continue web development best practices

Using ARIA is not an excuse to avoid proper web development techniques

- Use semantic markup
- Separate content (HTML), design (CSS) and behavior (JS)
- Progressive enhancement
- First focus on HTML semantics & keyboard interaction
- Suggested order of implementing ARIA:
  - For labeling, structure: add ARIA with markup
  - For interactive widgets, add ARIA last

Most important thing...



Joe Lanman
@joelanman

if you work in accessibility, the most important thing to learn about isn't WCAG or ARIA, it's ableism

10:55 AM · Apr 1, 2021 · Twitter Web App

153 Retweets 14 Quote Tweets 599 Likes

Summary: ARIA Ground Rules

- No ARIA is better than bad ARIA.
- Always use web development best practices, especially semantic markup.
- Be cautious with:
    - `aria-hidden`
    - `role=presentation`
    - `role=application`
    - `role=form`

ARIA “Mighty” Rules

From W3C Using ARIA document: w3.org/TR/using-aria

Rule #1

If you can use a native HTML element or attribute instead of an ARIA role, state or property, then do so. [paraphrased]

Why is the following bad?

```
<span class=”btn” role=”button”>Go</span>
```

Rule #2

Do not change native semantics, unless you really have to.

Bad: `<h3 role=”button”>heading button</h3>`

Good: `<h3><button>heading button</button></h3>`

Rule #3

All interactive ARIA controls must be usable with the keyboard.

FYI:

> If using `role=button`, the element must be able to receive focus and a user must be able to activate it using both the Enter/Return and the Space keys.

Rule #4

Do not use role="presentation" or aria-hidden="true" on a focusable element.

Bad: `<button role="presentation">press me</button>`

Bad:

```
<div aria-hidden="true">
  <button>press me</button>
</div>
```

Rule #5

All interactive elements must have an accessible name.

Bad:

```
user name <input type="text">
```

Compliant:

```
user name <input type="text" aria-label="user name">
```

Good:

```
<label for="username">user name</label>
<input type="text" id="username">
```

Summary: ARIA “Mighty” Rules

- Always use an HTML element or attribute before using ARIA.
- Avoid changing native semantics; consider nesting HTML elements.
- All custom controls must be usable with the keyboard.

General use cases

HTML5 Structure

HTML5 Structural Elements

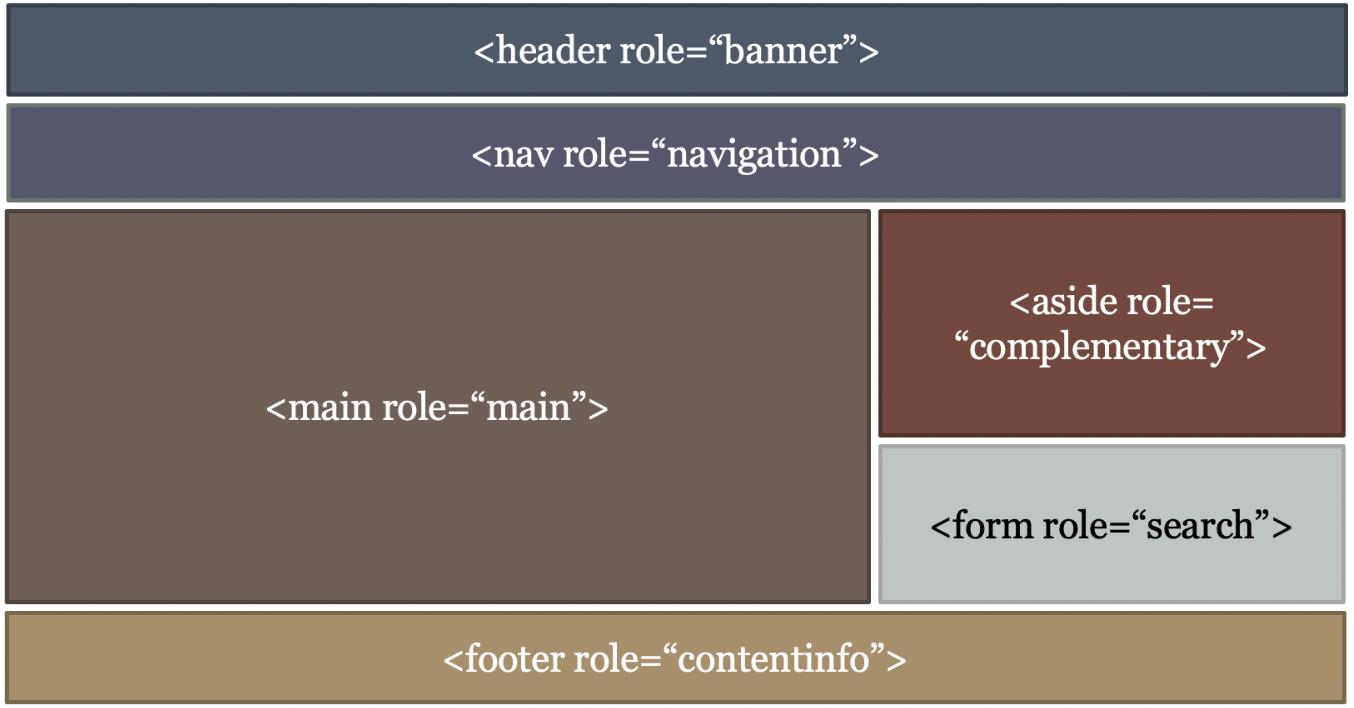
- Header
- Nav
- Main
- Section
- Article
- Aside
- Footer

HTML5 Structure + Landmarks

HTML5 Structural Elements + Landmark Roles

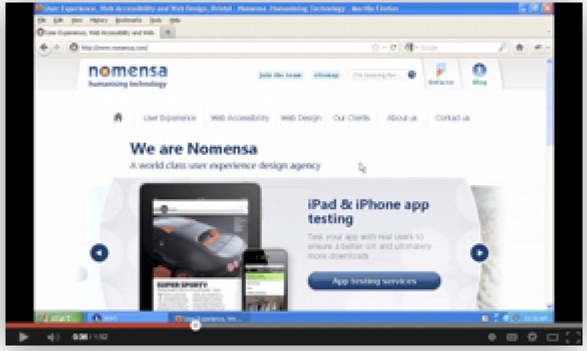
- Header + banner (page header only)
- Nav + navigation
- Main + main
- Section + region
- Article + [none]
- Aside + complementary
- [none] + search
- Footer + contentinfo (page footer only)

Landmarks example



Landmarks demo

youtu.be/IhWMou12_Vk


We are Nomensa

A world class user experience design agency

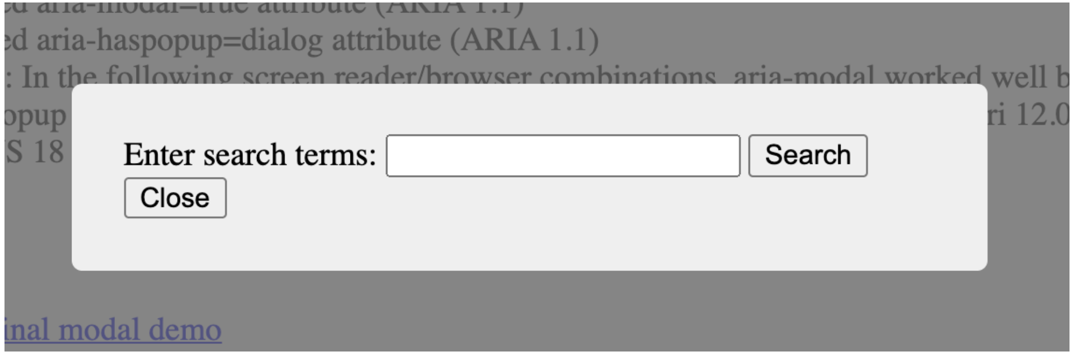
iPad & iPhone app testing

Labelling: aria-label, example 1

- Use `aria-label` when (visual text) label is not on screen
- Creates the accessible name
- Never use `aria-label` on a generic element (without role)

Example 1: modal dialog container

```
<div role="dialog" aria-label="Search Dictionary">
```


ed aria-modal=true attribute (ARIA 1.1)
ed aria-haspopup=dialog attribute (ARIA 1.1)
: In the following screen reader/browser combinations, aria-modal worked well b
opup ri 12.0
S 18

Enter search terms:

Labelling: aria-label, example 2

Example 2: SVG icon (without visual text label)

```
<svg aria-label="red square" role="img" focusable="false">
  <rect width="50" height="50" fill="#cc0000" />
</svg>
```

Labelling: aria-label, example 3

Example 3: Multiple navigation landmarks

```
<nav aria-label=”main menu”>
  <ul><li><a href…

<nav aria-label=”related topics”>
  <ul><li><a href...
```

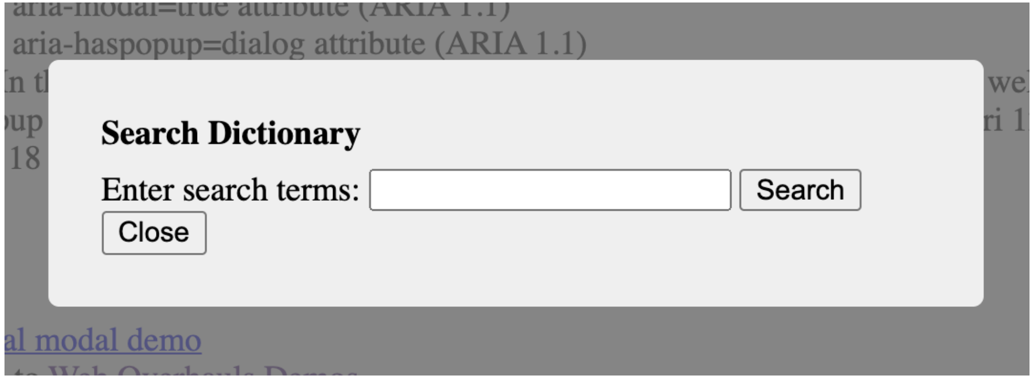
Labelling: aria-labelledby

- Use `aria-labelledby` when label text is already available
  - point to ID of that text element
- Creates the accessible name
- Never use `aria-labelledby` on a generic element (without role)

Labelling: aria-labelledby, example 1

Example 1: modal dialog container

```
<div role=”dialog” aria-labelledby=”hdgSearch”>
  <h1 id=”hdgSearch”>Search Dictionary</h1>
```


aria-modal=true attribute (ARIA 1.1)
aria-haspopup=dialog attribute (ARIA 1.1)

Search Dictionary

Enter search terms:

al modal demo

Labelling: aria-labelledby, example 2

Example 2: SVG icon

```
<svg role="img" aria-labelledby="title-0">
  <title id="title-0">red square</title>
  <rect width="50" height="50" fill="#cc0000" />
</svg>
```

Labelling: aria-labelledby, example 3

Example 3: Sections should have headings.

```
<section aria-labelledby=”news”>
  <h2 id=”news”>Newsletters</h2>
```

Describing, example 1

Use `aria-describedby` to associate advisory text with an element. This is often instructional text and error messages for form inputs.

Example 1: instructional text

```
<button aria-describedby="trash-desc">Move to trash</button>

<p id="trash-desc">Items in the trash will be permanently
removed after 30 days.</p>
```

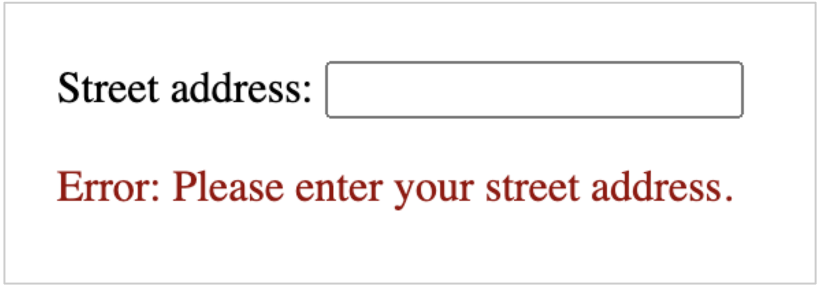
Describing, example 2

Example 2: error message

```
<label for=”text-address”>Street address:</label>

<input type=”text” id=”text-address” aria-describedby=”err-address” aria-required=”true”>

<p id=”err-address”>Error: Please enter your street address.</p>
```


Street address:

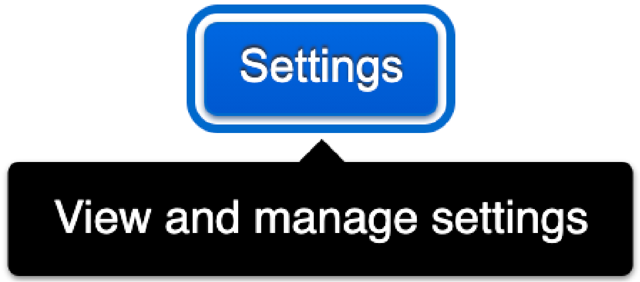
Error: Please enter your street address.

Describing, example 3

Example 3: tooltip

```
<button aria-describedby="desc">
  Settings
</button>

<p id="desc" role="tooltip" hidden>
View and manage settings</p>
```



Difference of 3 States

These ARIA states have similar meanings but must be used on specific roles:

- `aria-checked`
- `aria-selected`
- `aria-pressed`

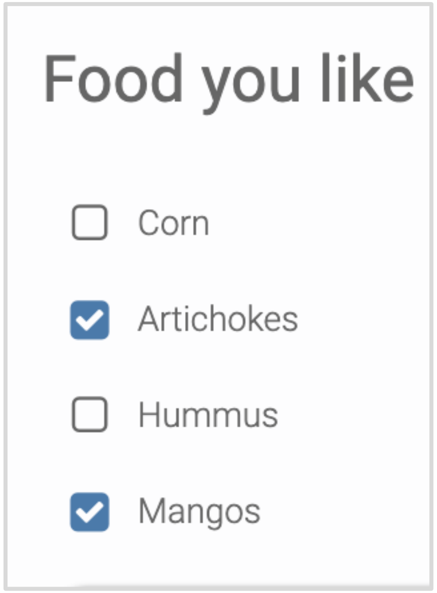
Difference of 3 States - Checked

Use `aria-checked` on:

- checkbox
- radio
- switch

```
<div role="checkbox" tabindex="0" aria-
labelledby="mangos" aria-checked="true"></div>
```

w3.org/TR/wai-aria-1.1/#aria-checked
pattern-library.dequelabs.com/components/checkboxes



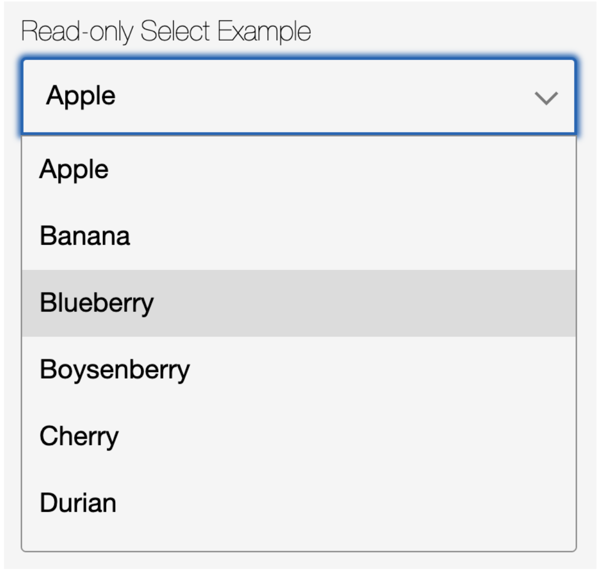
Difference of 3 States - Selected

Use `aria-selected` on:

- gridcell
- option
- row
- tab

```
<div role="option" aria-selected="true">
Blueberry</div>
```

w3.org/TR/wai-aria-1.1/#aria-selected
codepen.io/smhigley/pen/gObMVzv



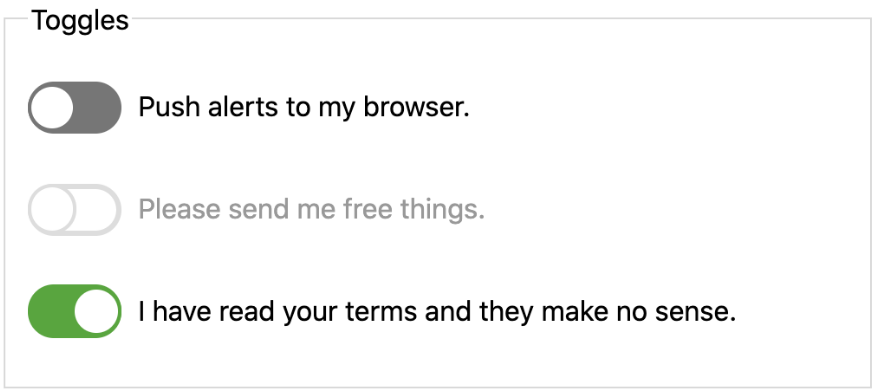
Difference of 3 States - Pressed

Use `aria-pressed` on:

- Button

```
<button type="button" id="alerts"
aria-pressed="false">
  Push alerts to my browser.
</button>
```

w3.org/TR/wai-aria-1.1/#aria-pressed
codepen.io/aardrian/pen/QemwBq


Toggles
Push alerts to my browser.
Please send me free things.
I have read your terms and they make no sense.

Presentation role

Removes the semantics from the element.

Note that synonym role of *none* added to ARIA 1.1; role="none". All modern browsers *do support* both role="none" and role="presentation" (not IE11).

Use role="presentation" to:

- Improve layout tables
- Repair parent-child relationships

Presentation role, example 1

Example 1: Improve screen reader experience for layout tables

```
<table role=”presentation” id=”layout-table”>
  <tr>
    <td>
```

Presentation role, example 2

Example 2: widgets, repair parent-child relationships

```
<ul role="tablist">
    <li role="presentation">
        <a href="#panel1" aria-describedby="tab-tip" role="tab"
aria-selected="false" tabindex="-1"...>The Cure</a>
</li>
...
```

Current state

Use `aria-current` to programmatically indicate a current item within a set of related elements. Often conveyed only visually.

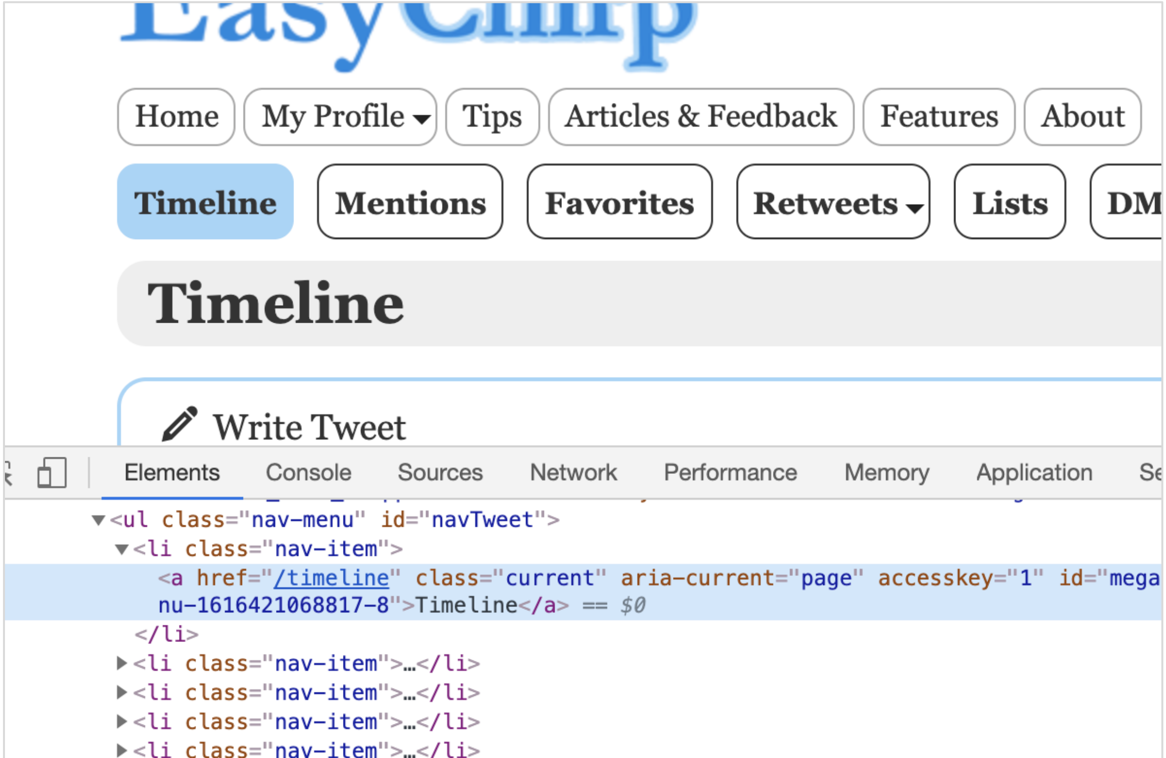
Values:

- true | false
- page, step, location, date, time

Previously, had to use visually hidden text to denote current state.

Current state, example 1

`aria-current="page"`

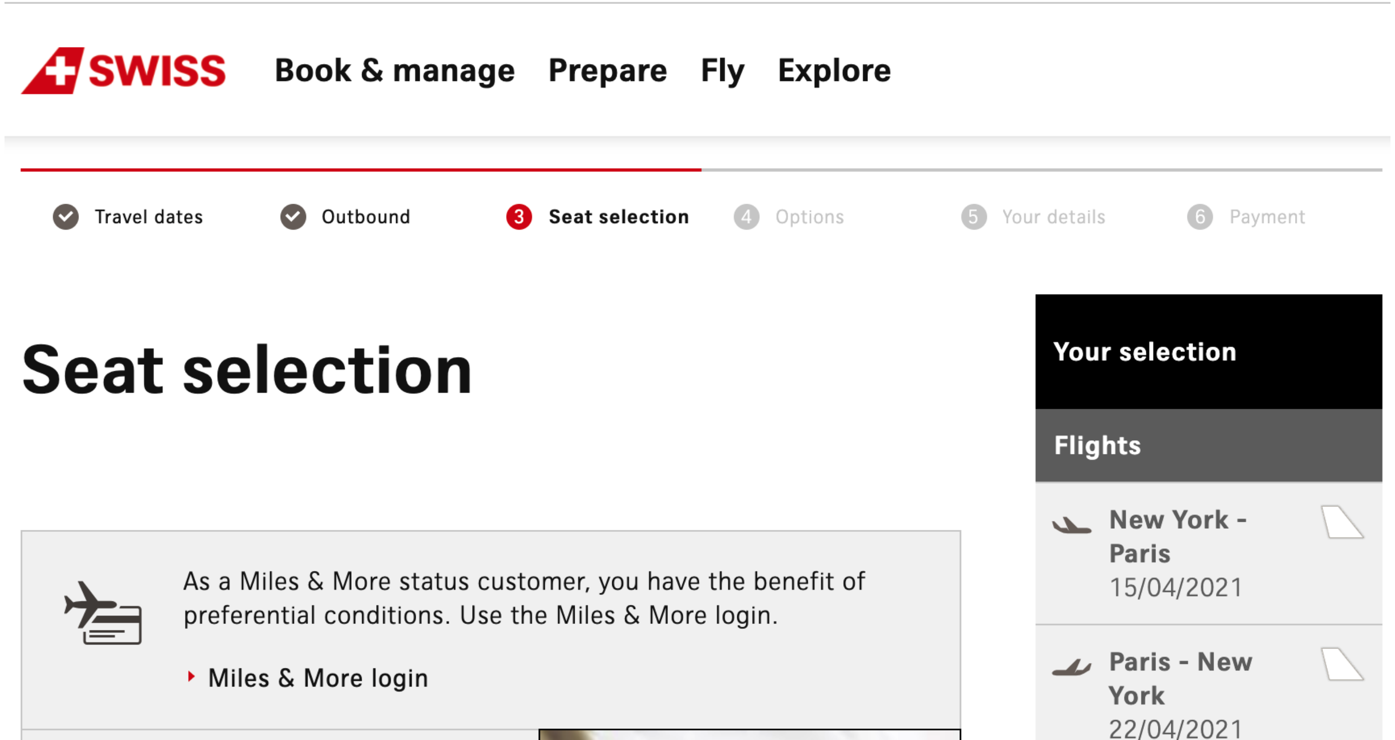


Timeline

```
▼<ul class="nav-menu" id="navTweet">
  ▼<li class="nav-item">
      <a href="/timeline" class="current" aria-current="page" accesskey="1" id="mega
      nu-1616421068817-8">Timeline</a> == $0
    </li>
  ▶<li class="nav-item">…</li>
  ▶<li class="nav-item">…</li>
  ▶<li class="nav-item">…</li>
  ▶<li class="nav-item">…</li>
```

Current state, example 2

`aria-current="step"`



Travel dates ✓ Outbound ✓ 3 Seat selection 4 Options 5 Your details 6 Payment

Seat selection

As a Miles & More status customer, you have the benefit of preferential conditions. Use the Miles & More login.

Miles & More login

Your selection

Flights

New York - Paris 15/04/2021

Paris - New York 22/04/2021

Live Regions

Screen reader will automatically output dynamic content changes within a live region element.

- Typical:
  `<p aria-live=”polite”>updating content here</p>`
- Live Region roles inherit the behavior.
  - alert, log, marquee, status, timer
  - Used rarely and support may be inconsistent
- With great power comes great responsibility!
  - The content may not be output (due to buffering, interruptions)
  - Too much content may be output

Live Regions - tech

- aria-live (property)
  - off, polite, assertive
- aria-atomic (property)
  - Defines whether all of the content in the ARIA live region should be announced (true), or only the part that's changed (false) (default).
- aria-relevant (property)
  - additions, removals, text, all
- aria-busy (state)

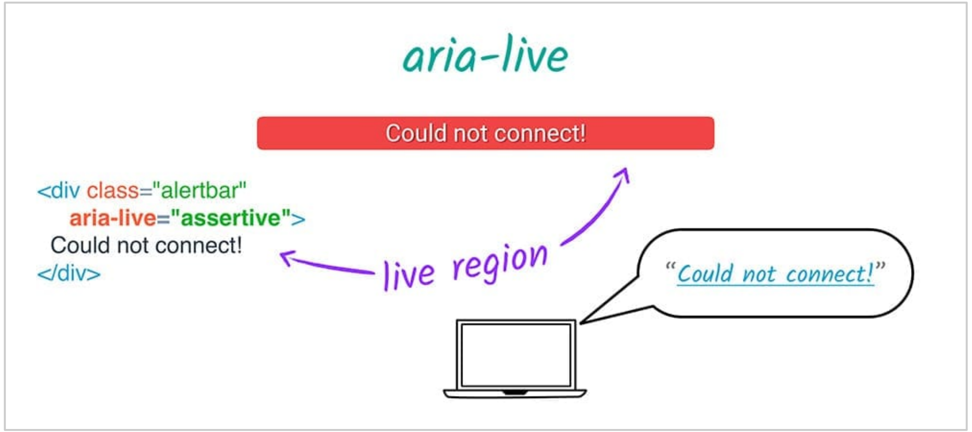
Live Regions - uses

Could be used for:

- Updating dynamic search results
- Updating results in autosuggest
- Updated shopping cart items
- Alert messages, “Toast” message
  - depending on importance
- Character counter
- Countdown timer (careful!)
- Sports score
- Stock price

Suggest *not* using for:

- Carousels (too verbose)
- Form error messages (output conflicts)



```
<div class="alertbar"
   aria-live="assertive">
 Could not connect!
</div>
```

Live Regions - sample code

Sample code:

```
<h2>Basket summary</h2>
<div aria-live=”polite” aria-atomic=”true”>
  <p>Cart contains <span id=”quantity”>0</span> items.</p>
</div>
```

Test pages:

- codepen.io/weboverhauls/pen/OJWEgpw
- weboverhauls.github.io/demos/chips/

Summary: General Use Cases

- Implement landmark roles, labelling and descriptions when needed.
- Use `aria-selected` on options & tabs.
- Use `aria-checked` on checkboxes & radios.
- `aria-current` is easy and useful!
- Be judicious when implementing live regions.

Components

Disclosure (show/hide)

ARIA usage:

- `<button>` OR `role=”button”`
- `aria-expanded=”true|false”`
- `aria-controls=”ID_VALUE”` (optional)

WAI-ARIA Authoring Practices 1.1:
w3.org/TR/wai-aria-practices-1.1/#disclosure

Disclosure Tips

Tips 💡

- If you customize button, ensure activates on Enter & Spacebar.
  - [Do this universally, not just for Disclosure.]
- Structure options
  - Could wrap button with heading
  - Could use <DL> structure

Disclosure Tips cont.

Tips 💡

- Could also use HTML5 `<details>` and `<summary>` (but no IE11 support)
  - Reference: adrianroselli.com/2020/05/disclosure-widgets.html
  - But don’t use for accordion twitter.com/ChrisFerdinandi/status/1381376045601992709
- Some say “aria-controls is poop”!
  - Reference: heydonworks.com/article/aria-controls-is-poop/
  - Poor support: a11ysupport.io/tech/aria/aria-controls_attribute

Disclosure Examples

Examples:

- weboverhauls.github.io/demos/expand-collapse/index2.html
- scottaohara.github.io/aria_disclosure_widget/
- w3.org/TR/wai-aria-practices-1.1/examples/disclosure/disclosure-navigation.html
- bbc.github.io/gel/components/accordions/

Modal Dialog

ARIA usage:

- `role=”dialog|alertdialog”`
- `aria-modal=”true”`
- `aria-labelledby` OR `aria-label`
- `aria-describedby` (optional)

WAI-ARIA Authoring Practices 1.1:
www.w3.org/TR/wai-aria-practices-1.1/#dialog_modal

Modal Dialog Tips

Tips 💡

- Use button to open; should add `aria-haspopup=”dialog”`
- When opened, set focus on or within dialog
- When open, keyboard focus must remain within dialog
- When closed, return focus to control that opened it (or other logical element)
- Provide visual close button
  - Escape key to close is optional but recommended

Modal Dialog Examples

Examples:

- weboverhauls.github.io/demos/modal/modal_demo2.html
- scottaohara.github.io/accessible_modal_window/
- www.w3.org/TR/wai-aria-practices-1.1/examples/dialog-modal/dialog.html

Tab Panel

ARIA usage:

- `role="tablist"`
- `role="tab"`
- `role="tabpanel"`
- `aria-labelledby` AND/OR `aria-label`
- `aria-controls="ID_VALUE"`
- `aria-selected="true|false"`
- `aria-orientation="horizontal|vertical"` (optional)

WAI-ARIA Authoring Practices 1.1:
www.w3.org/TR/wai-aria-practices-1.1/#tabpanel

Tab Panel Tips

Tips 💡

- Keyboard interaction pattern may vary (unfortunately)
  - Consider providing user with instructions
- Recommend adding `tabindex=0` to panel containers
  - Facilitates movement to panel content for keyboard & assistive technology users.
  - Especially helpful if panels do not contain a focusable element or if panel content requires scrolling.
- Use `aria-label` on tablist if more than one on a page.
- Only use `aria-orientation` when the tabs are aligned vertically
  - horizontal is default

Tab Panel Examples

Examples:

- weboverhauls.github.io/demos/tab-panel/
- ebay.github.io/mindpatterns/disclosure/tabs/index.html
- w3.org/TR/wai-aria-practices-1.1/examples/tabs/tabs-1/tabs.html

Quiz Time

Question 1

What Rule #1 of ARIA?

Question 2

Can you name 3 or more ARIA landmark roles?

Question 3

What is a use case role="presentation | none"?

Question 4

What is a use case for aria-labelledby?

Question 5

What are some use-cases for a live region?

What are the 3 values for aria-live?

Question 6

On what role could you use aria-checked?

Question 7

On what role could you use aria-selected?

Resources

Resource List

- WAI-ARIA Authoring Practices 1.1
  (includes Design Patterns and Widgets—be sure to test; also, mobile support needs improvement)
  w3.org/TR/wai-aria-practices-1.1/
- Getting started with ARIA: a11yproject.com/posts/2014-05-15-getting-started-aria/
- Blog: Sarah Higley sarahmhigley.com/writing/
- Blog: Adrian Roselli adrianroselli.com/
- Myth: ARIA Has Perfect Support:
  www.a11yproject.com/posts/2020-05-13-aria-has-perfect-support/
- A Complete Guide To Accessible Front-End Components - Smashing Magazine
  smashingmagazine.com/2021/03/complete-guide-accessible-front-end-components/
- Web Accessible Code Libraries and Design Patterns - Web Axe
  webaxe.org/web-accessible-code-library-design-systems-patterns/
- Enough with the role-play—let's get back to the basics
  https://www.tpgi.com/enough-with-the-role-play-lets-get-back-to-the-basics/

Questions?

Thank you!

Dennis Lembree
Director of Accessibility, Diamond
@DennisL / @DWSLA