Increasing Web Accessibity With ARIA: A Crash Course

Jeane Marty
Senior Web Developer
University of Washington

Hadi Rangin
IT Accessibility Specialist
University of Washington

Pete Graff
Senior Developer & UX Designer
University of Washington

Agenda

- Who Are We?
- Overview
- Demo: Screen Reading A Not-So-Accessible Page
- Semantic HTML
- ARIA Landmarks and Labeling

- Short Break

- Expanding/Collapsing Content
- Forms, Live Regions
- Modal Dialogs
- Demo: Screen Reading An Accessible Page
- Next Steps and Resources

Overview

First off, what is ARIA?

- **A**ccessible **R**ich **I**nternet **A**pplications
- A technical spec published by the W3C that explains how to increase the accessibility of web pages
- In particular, the ARIA spec addresses how to handle modern dynamic content and associated UI components (elements that weren't a reality when HTML was originally developed)
- Became "official" (via the W3C) in 2014

What does ARIA do?

It communicates roles, states, and properties of interface elements to accessibility APIs, for the benefit of AT (accessible technology) users.

It answers questions like:

- What is this?
- How do I use it?
- What is its state? (Is it on/selected/expanded/collapsed?)
- What just happened?

The First Rule of ARIA

Don’t use ARIA unless you have to.

In other words, if you can **use native HTML** elements with the attributes, semantics, and behavior you require, do so.

In other other words, whenever possible, **avoid unneeded complexity**.

Source: Notes on Using ARIA in HTML (W3C Working Draft)
https://www.w3.org/TR/aria-in-html/

Demo: Screen Reading A Not-So-Accessible Page

Semantic HTML

Proper semantic HTML markup answers the question:

What is this?

Semantic Headings

- Code with `<h1>`, `<h2>`, etc.
- Should form an outline of the page content
- Don't skip levels in the outline

```
<h1>Primary Heading</h1>
<h3>Subheading</h3>
<p>Lorem ipsum dolor sit amet,
consectetur adipiscing elit…</p>
```

Semantic Headings

- Code with `<h1>`, `<h2>`, etc.
- Should form an outline of the page content
- Don't skip levels in the outline

```
<h1>Primary Heading</h1>
<h2>Subheading</h2>
<p>Lorem ipsum dolor sit amet,
consectetur adipiscing elit…</p>
```

Semantic Form Field Labels

- Code with `<label>`

```
<label for="address">Address</label>
<input type="text" id="address">
```

Semantic Buttons

- Code with the `<button>` element
- Know the difference between buttons and links

Button

Triggers an **action** (submits a form, changes state of current page)

Link

Takes user to a **new location** (on a new page, or specific location on current page)

Exercise 1 - Semantic HTML

Directory

exercises/Exercise-1-Semantic-HTML/

File

_semantics.html

Instructions

There are multiple HTML elements in this file which are not fully semantic. Refactor to fix this.

Extra Credit

Once you add semantic structure to the collection of footer links, add some CSS (in the empty `<style>` tag) to improve its layout.

Solution

- *semantics-solution.html*
- *diff.txt*

But, what if even the most semantic HTML can’t do the job?

But, what if even the most semantic HTML can’t do the job?

HTML

+ ARIA =

ARIA Landmark Roles & Labeling

ARIA Landmark Roles

Provide a more descriptive way to represent blocks of content that commonly occur on web pages

- `role="main"`
- `role="banner"`
- `role="navigation"`
- `role="contentinfo"`
- `role="complementary"`
- `role="search"`
- `role="form"`
- `role="application"`

Some HTML5 elements map to ARIA Landmark Roles

- `<main>` → `role="main"`
- `<header>` → `role="banner"`*
- `<nav>` → `role="navigation"`
- `<footer>` → `role="contentinfo"`*
- `<aside>` → `role="complementary"`

*(sometimes)

Labeling/describing elements using ARIA

aria-label="*label*"

```
<nav role="navigation" aria-label="main menu">
   <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">About</a></li>
      <li><a href="#">Clients</a></li>
   </ul>
</nav>
```

Labeling/describing elements using ARIA

aria-labelledby="*id_of_element*"

```
<div aria-labelledby="headline">
   <h1 id="headline">Wild fires spread
   across the San Diego Hills</h1>
   Strong winds expand fires ignited by
   high temperatures ...
</div>
```

Labeling/describing elements using ARIA

aria-describedby="*id_of_element*"

```
<label for="question">What have you
learned today about ARIA?</label>

<textarea id="question" aria-
describedby="instructions"
required></textarea>

<div id="instructions">Please include at
least three lessons learned.</div>
```

Exercise 2: Landmark Roles and Labeling

Directory

exercises/Exercise-2-Landmark-Roles-Labeling/

File

_landmark.html

Instructions

Make use of semantic HTML elements (`<header>`, `<nav>`, `<main>`, `<aside>`, `<footer>`) as well as ARIA roles and labeling attributes to improve the overall accessibility of this page.

Extra Credit

Add CSS to improve overall layout.

Solution

- *landmark-solution.html*
- *diff.txt*

Break!

```
<coffee role=”crucial"
  aria-label=”break announcement">
</coffee>
```

Showing/Hiding & Expanding/Collapsing Content

Example: A "show more" button

University of Washington

ARIA Bootcamp

November 4, 2017

Menu Item 1 | Menu Item 2 | Menu Item 3

About this Page

This page includes a variety of accessibility problems. In this workshop we will explore these problems and their solutions. Show more

Accessible Web Design

Accessible web design is the practice of designing and developing websites that are usable by everyone, including individuals with disabilities. The World Wide Web Consortium (W3C) has developed Web Content Accessibility Guidelines that serve as the de facto standard for how to create accessible web pages and web applications.

ARIA

Accessible Rich Internet Applications (ARIA) is a W3C specification that is designed to make interactive web interfaces accessible. It does so by explicitly communicating the roles, states, and properties of user interface elements so assistive technologies such as screen readers and speech recognition technologies can understand them and present them in meaningful ways for their users. ARIA answers questions like:

- What is this element?
- What is its current state? For example, is it selected? expanded? visible?

Post-Workshop Questionnaire

Please submit the following form upon completion of the workshop. All fields are required.

Your name:

What have you learned today about ARIA?

Please include at least three lessons learned.

Submit

Resources

- UW Accessible Technology
- UW IT Accessibility Guidelines
- UW IT Accessibility Checklist
- Developing Accessible Websites
- Using ARIA for Web Applications

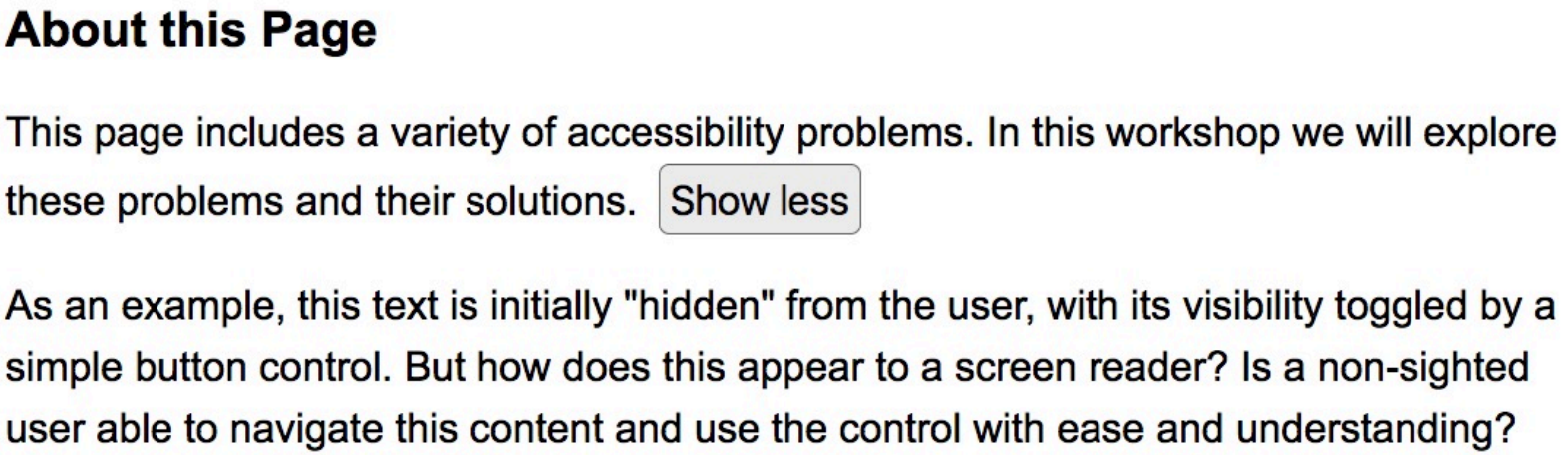
Example: A "show more" button

About this Page

This page includes a variety of accessibility problems. In this workshop we will explore these problems and their solutions. Show less

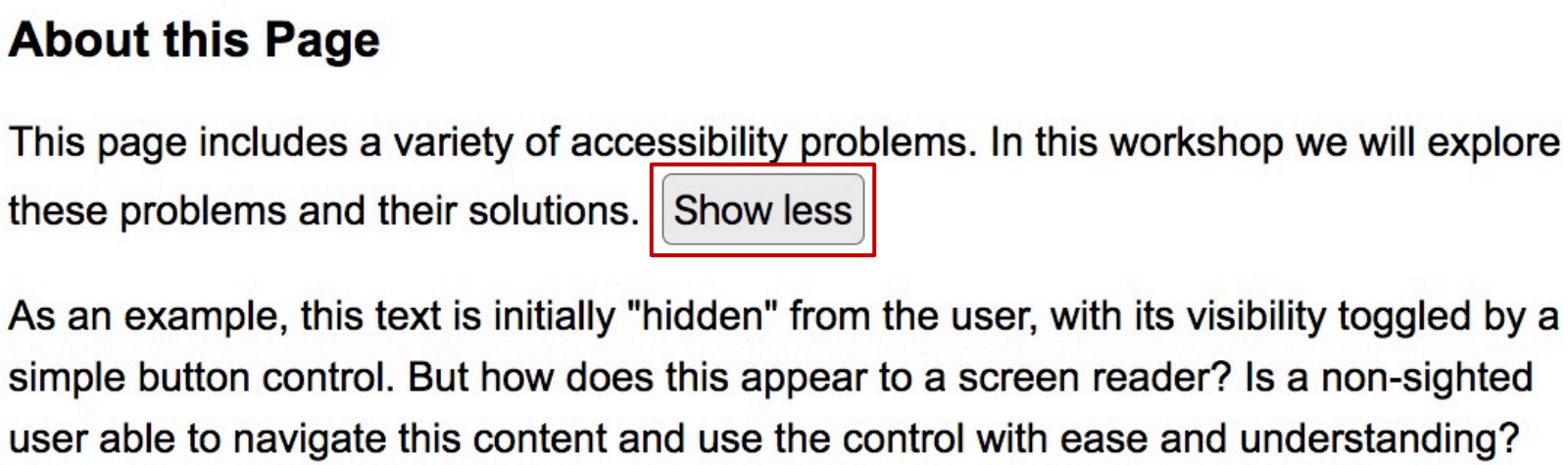
As an example, this text is initially "hidden" from the user, with its visibility toggled by a simple button control. But how does this appear to a screen reader? Is a non-sighted user able to navigate this content and use the control with ease and understanding?

Example: A "show more" button



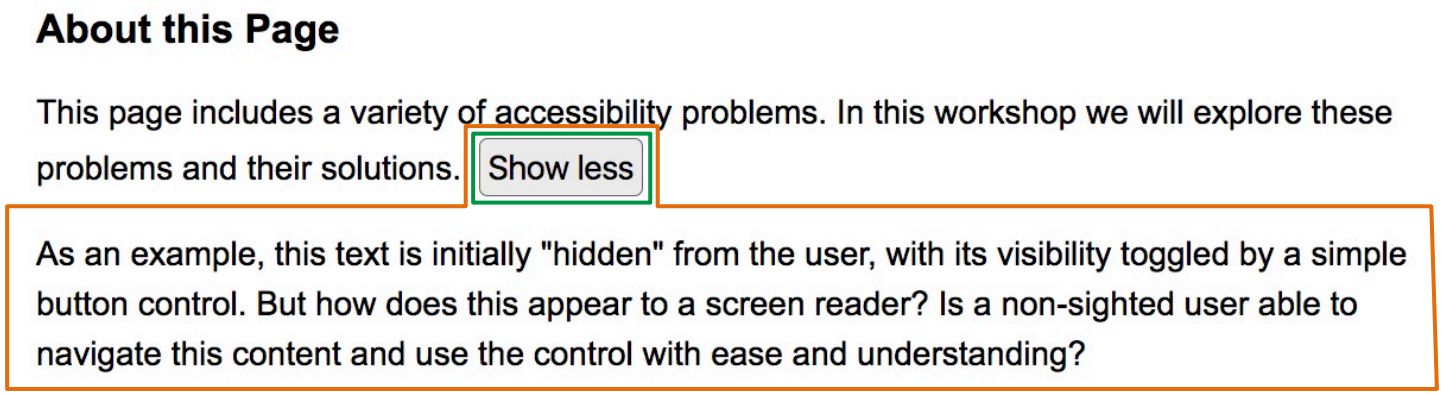
- When clicked, toggles the display of supplemental text
- When supplemental text is visible, button text changes to "Show less”
- Handled with JavaScript
- But how?

Make it a button



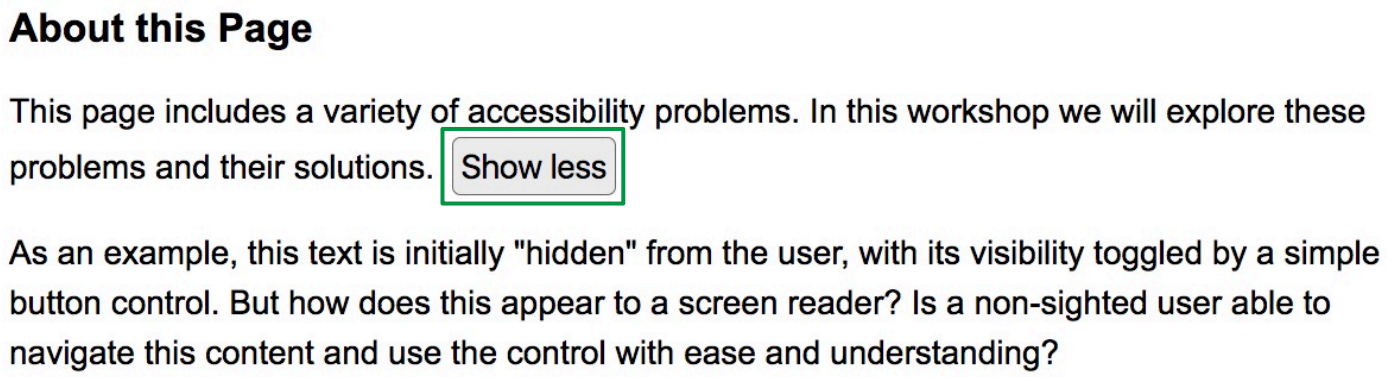

```
<p>This page … and their solutions.</p>
<button id="text-control">Show less</button>
```

Add aria-controls




```
<button id="text-control" aria-
controls="supplemental-text">Show
less</button>

<p id="supplemental-text">As an example, this
text is initially "hidden” from the user …
and understanding?</p>
```

Add aria-expanded




```
<button id="text-control" aria-controls="supplemental-text” aria-expanded="true">Show less</button>

<p id="supplemental-text">As an example, this text is initially "hidden” from the user … and understanding?</p>
```

(Optionally) add aria-hidden

About this Page

This page includes a variety of accessibility problems. In this workshop we will explore these problems and their solutions. Show less

As an example, this text is initially "hidden" from the user, with its visibility toggled by a simple button control. But how does this appear to a screen reader? Is a non-sighted user able to navigate this content and use the control with ease and understanding?

```
<button id="text-control" aria-
controls="supplemental-text" aria-
expanded="true">Show less</button>

<p id="supplemental-text" aria-
hidden="false">As an example, this text is
initially "hidden” from the user … and
understanding?</p>
```

Wire with JavaScript

About this Page

This page includes a variety of accessibility problems. In this workshop we will explore these problems and their solutions. Show more

As an example, this text is initially "hidden" from the user, with its visibility toggled by a simple button control. But how does this appear to a screen reader? Is a non-sighted user able to navigate this content and use the control with ease and understanding?

```
<button id="text-control" aria-controls="supplemental-text" aria-expanded="false">Show more</button>

<p id="supplemental-text" aria-hidden="true">As an example, this text is initially "hidden” from the user … and understanding?</p>
```

Review: Making a "show more" button accessible

1. Make it a `<button>`
   if `<button>` isn't possible, use `role="button"`
2. Add `aria-controls="`*`id_of_controlled_elem`*`"`
3. Add `aria-expanded="false"`
4. Optionally, add `aria-hidden="true"` to content being hidden/shown
5. As content is shown/hidden, use JavaScript to change values of `aria-expanded` and `aria-hidden` attributes

Exercise 3: Showing/Hiding Content

Directory

exercises/Exercise-3-Showing-Hiding-Content/

Files

- *_show-hide.html*
- *_show-hide.js*

Instructions

- In *_show-hide.html*, add appropriate ARIA attributes to the button element.
- In *_show-hide.js*, refactor to apply (and toggle) the `aria-expanded` attribute, as well as verify that the button text is toggling correctly. Optionally, implement use of `aria-hidden`.

Solution

- *show-hide-solution.html*
- *show-hide-solution.js*
- *diff.txt*

Forms

Simple form

Your name:

What have you learned today about ARIA?

Please include at least three lessons learned.

Submit

Please submit the following form upon completion of the workshop. All fields are required.

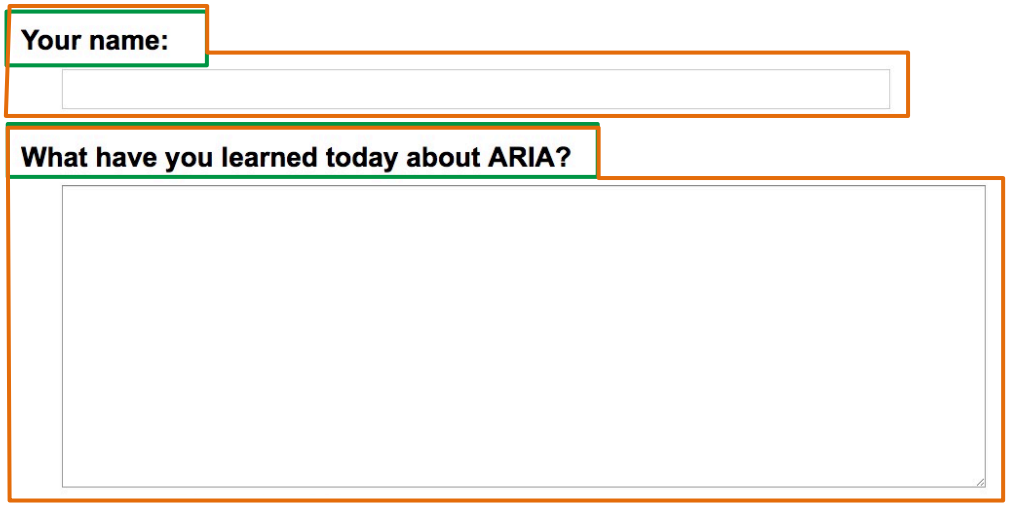
Your name:

What have you learned today about ARIA?

Please include at least three lessons learned.

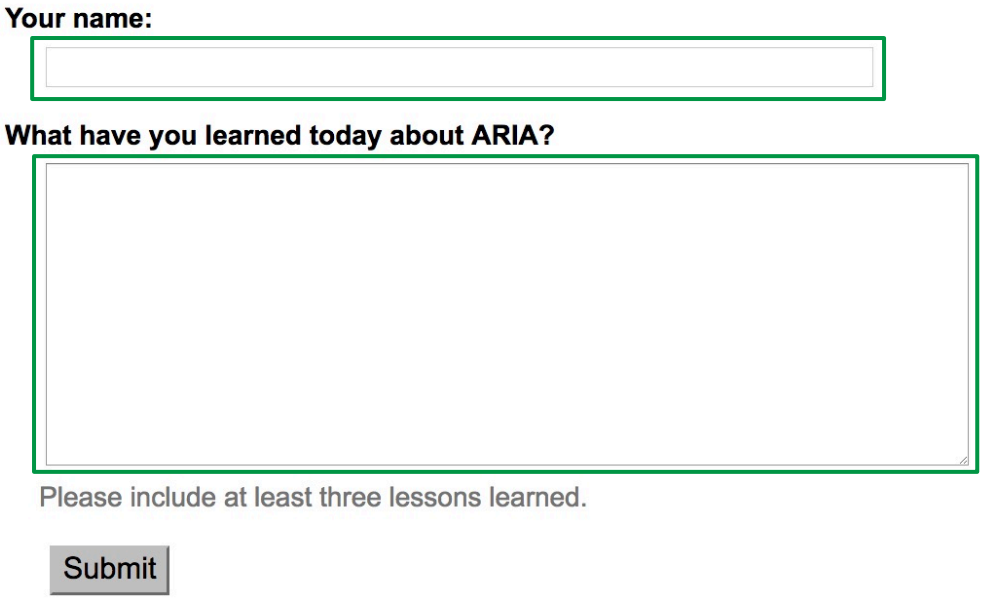
Submit

Label all inputs




```
<label for="name">Your name:</label>
<input type="text" id="name">

<label for="question">What have you learned
today about ARIA?</label>
<textarea id="question"></textarea>
```

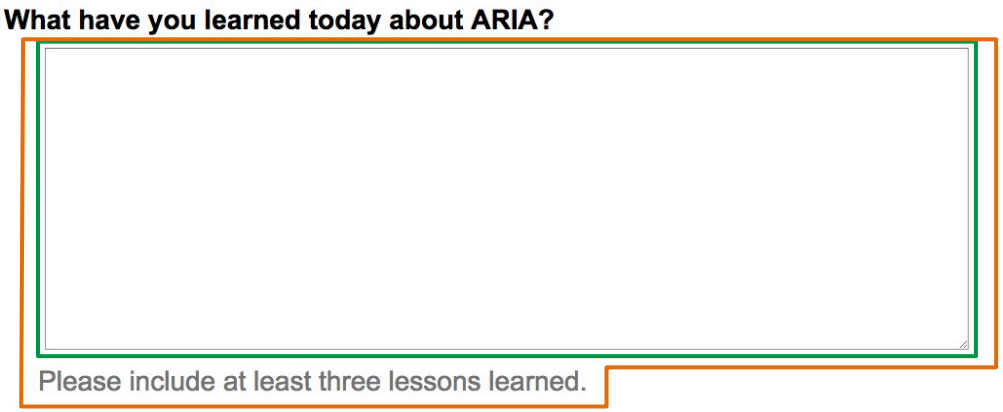
Use HTML5 `required` attribute



Specifies that an **input must have a value** before the form will be submitted

```
<input type="text" id="name" required>

<textarea id="question" required></textarea>
```

Use `aria-describedby` for supplemental help text



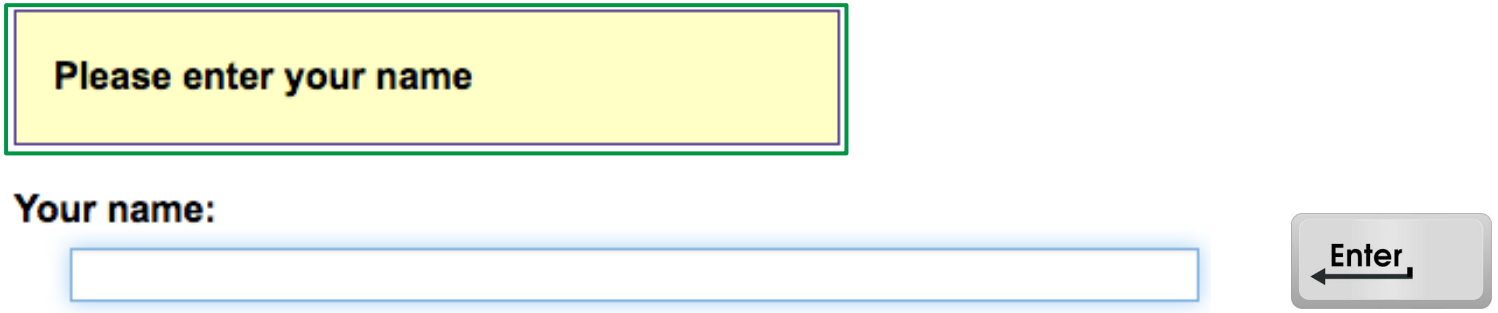

```
<textarea id="problem" required aria-describedby="help"></textarea>

<p id="help">Please include at least three lessons learned.</p>
```

ARIA Live Regions

- Screen readers announce any changes to content (useful for alerts & error messages)
- `aria-live="assertive"` interrupts user now
- `aria-live="polite"` interrupts when user is idle
- `aria-atomic="true"` announces *all* content within an element; if false, announces only changed content

ARIA Live Regions

- `role="alert"` behaves like `aria-live="assertive"`, and some screen readers preface with "Alert!"




```
<div id="error" role="alert"></div>
```

Exercise 4: Forms

Directory

exercises/Exercise-4-Forms/

Files

- *_forms.html*
- *_forms.js*

Instructions

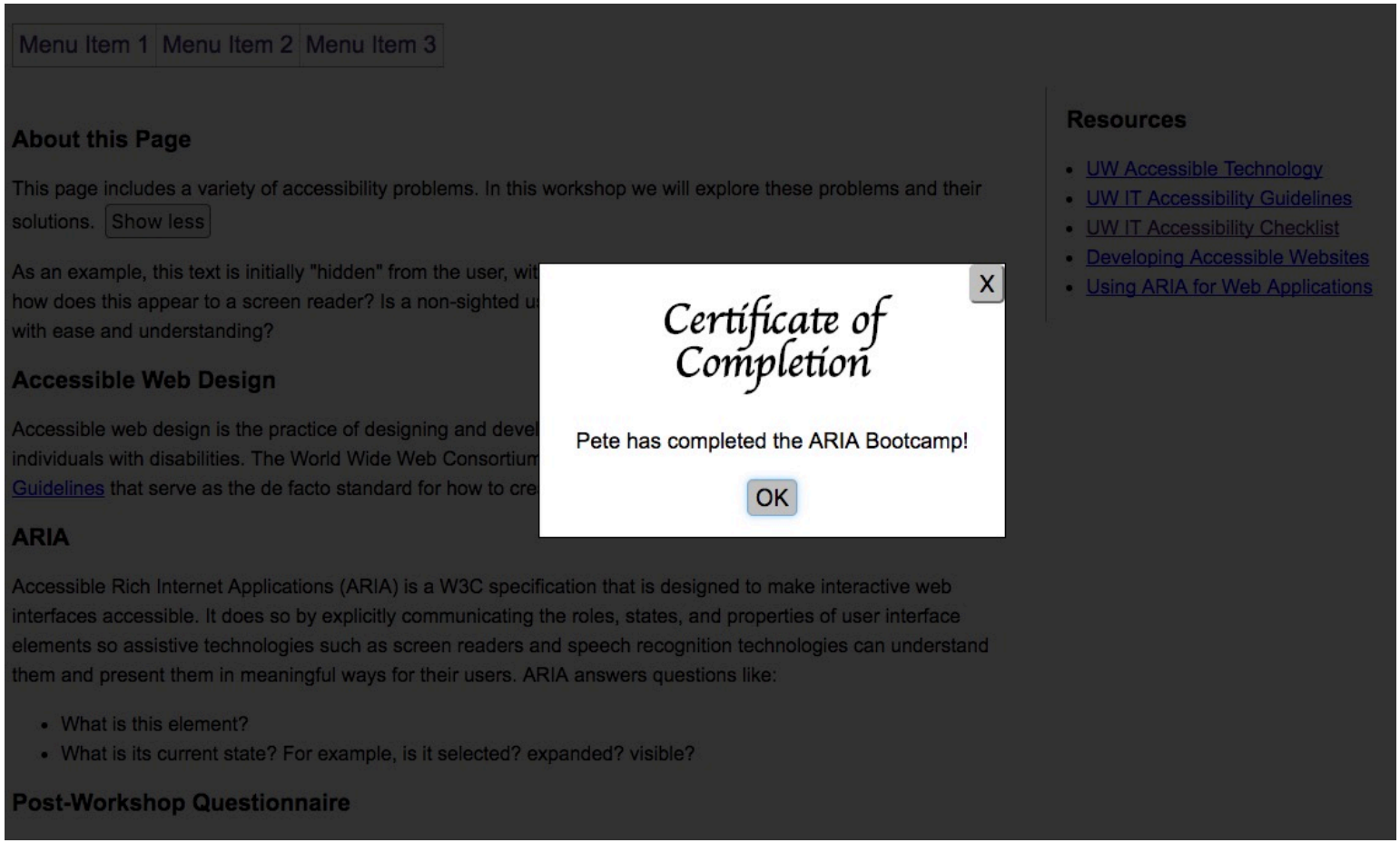
- In *_forms.html*, make use of the required attribute as well as the alert role to improve the overall accessibility of the form.
- In *_forms.js*, set the focus on the field or question which triggers an error.

Solution

- *forms-solution.html*
- *forms-solution.js*
- *diff.txt*

Modal Dialogs

Making a modal dialog accessible



Making a modal dialog accessible

```
<div id= "content”> … </div>

<div id="modal">
  <button type="button" id="close" aria-label="Close
   dialog">X</button>
  <h1 id="modal-heading">Certificate of Completion</h1>
  <div>
    <span id="participant">(name)</span> has completed the
     ARIA Bootcamp!
  </div>
  <button type="button" id="ok">OK</button>
</div>

<div id="modal-overlay"></div>
```

Add `role="dialog”` or `role="alertdialog"` to the outer dialog element

```
<div id= "content”> … </div>

<div id="modal" role="dialog">
  <button type="button" id="close" aria-label="Close
   dialog">X</button>
  <h1 id="modal-heading">Certificate of Completion</h1>
  <div>
    <span id="participant">(name)</span> has completed the
     ARIA Bootcamp!
  </div>
  <button type="button" id="ok">OK</button>
</div>

<div id="modal-overlay"></div>
```

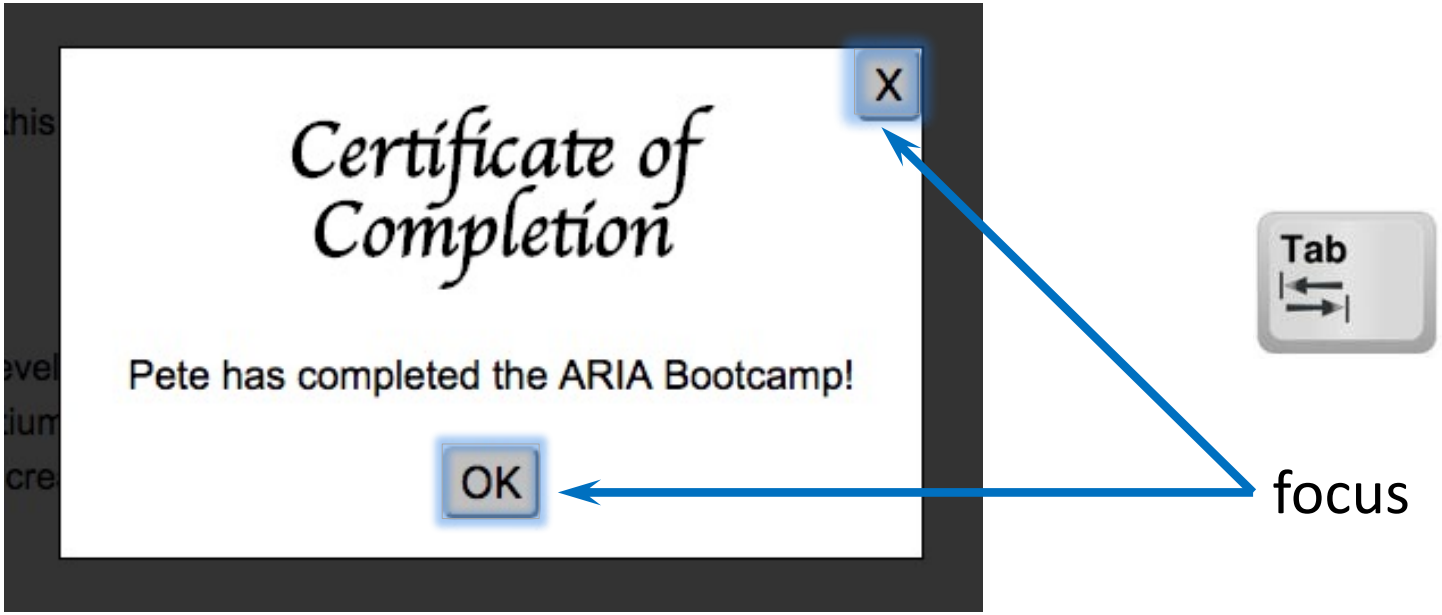
Be sure dialog has an <h1>

```
<div id= "content" > … </div>

<div id="modal" role="dialog">
  <button type="button" id="close" aria-label="Close
   dialog">X</button>
  <h1 id="modal-heading">Certificate of Completion</h1>
  <div>
    <span id="participant">(name)</span> has completed the
     ARIA Bootcamp!
  </div>
  <button type="button" id="ok">OK</button>
</div>

<div id="modal-overlay"></div>
```

Add `aria-labelledby` that references the `<h1>` to outer dialog element

```
<div id= "content”> … </div>

<div id="modal" role="dialog” aria-labelledby="modal-
heading">
  <button type="button" id="close" aria-label="Close
   dialog">X</button>
  <h1 id="modal-heading">Certificate of Completion</h1>
  <div>
    <span id="participant">(name)</span> has completed the
     ARIA Bootcamp!
  </div>
  <button type="button" id="ok">OK</button>
</div>

<div id="modal-overlay"></div>
```

Place keyboard focus on first focusable element within dialog and trap focus on subsequent tabs (so users can't tab out)

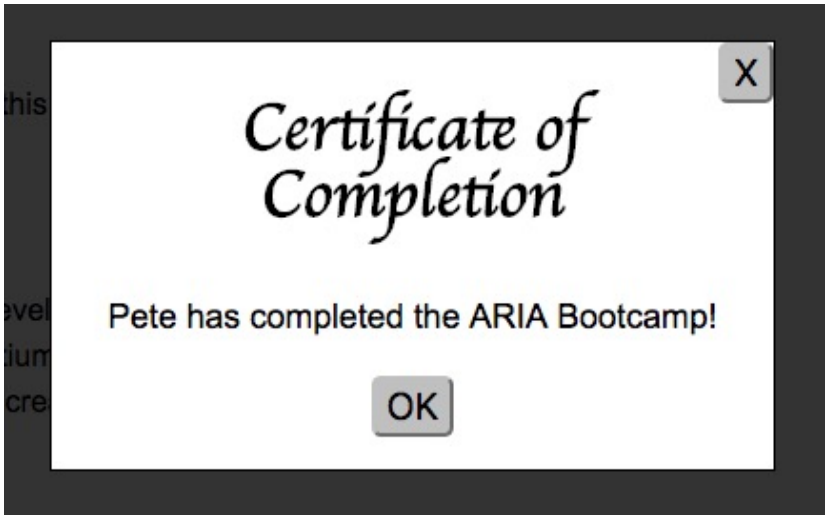


Be sure dialog is positioned outside of all other content in the DOM (e.g., as child of <body>)

```
<div id="content"> … </div>


<div id="modal" role="dialog” aria-labelledby="modal-
heading">
  <button type="button" id="close" aria-label="Close
   dialog">X</button>
  <h1 id="modal-heading">Certificate of Completion</h1>
  <div>
    <span id="participant">(name)</span> has completed the
     ARIA Bootcamp!
  </div>
  <button type="button" id="ok">OK</button>
</div>

<div id="modal-overlay"></div>
```

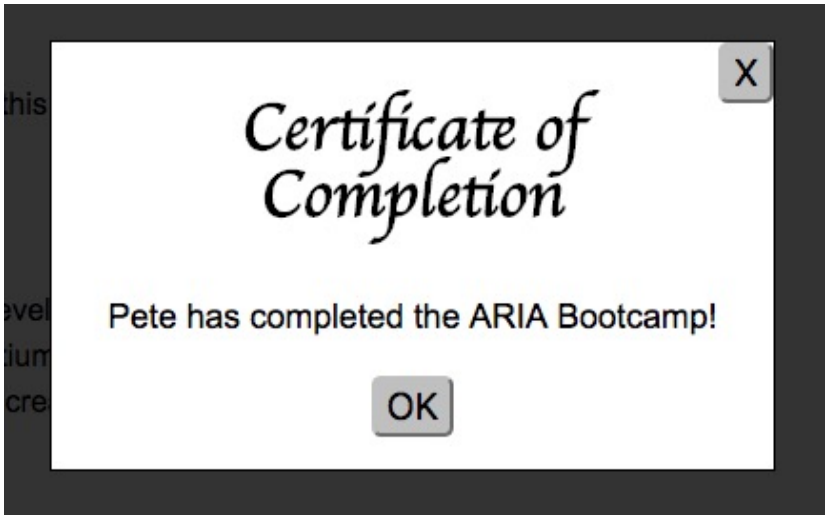
When dialog is visible, add `aria-hidden="true"` to all content *except* the dialog

```
<div id="content" aria-hidden="true"> … </div>

<div id="modal" role="dialog" aria-labelledby="modal-
heading"> … </div>
```

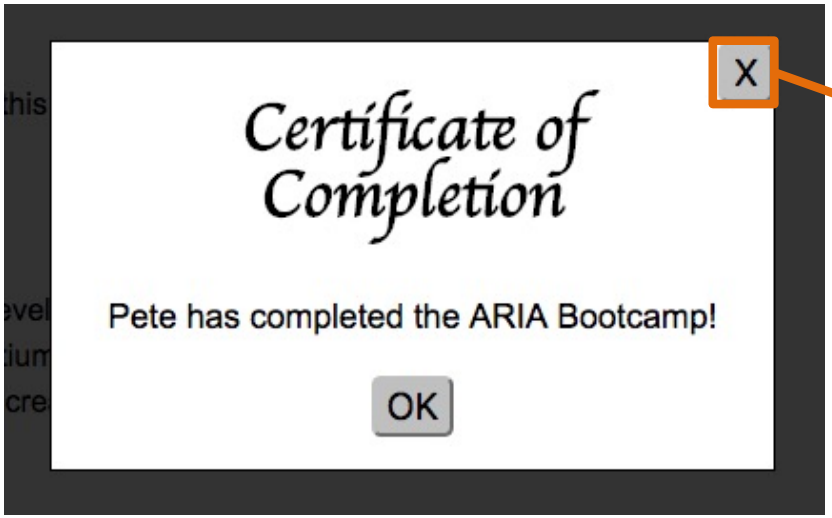


Then, when dialog is not visible, remove it (or set it to false)

```
<div id="content" aria-hidden="true"> … </div>

<div id="modal" role="dialog" aria-labelledby="modal-
heading"> … </div>
```



Add ability of escape key to close dialog



If dialog includes form fields, warn user before closing that data will be lost.

Review: Making a model dialog accessible

1. Add `role="dialog"` or `role="alertdialog"` on the outer dialog element
2. Be sure dialog has an `<h1>`
3. Add `aria-labelledby` that references the `<h1>` to outer dialog element
4. Place keyboard focus on first focusable element within dialog and trap keyboard within dialog (so users can't tab out)
5. Be sure dialog is positioned outside of all other content in the DOM (e.g., as child of `<body>`, a sibling of the page's content)
6. When dialog is visible, add `aria-hidden="true"` to all content *except* the dialog
7. Be sure to remove `aria-hidden` (or set to false) when dialog is closed
8. Add ability of escape key to close dialog (if dialog includes form fields, warn user before closing that data will be lost)

Demo: Screen Reading An Accessible Page

Next Steps & Resources

Resources

- UW Accessible Technology
  http://uw.edu/accessibility
- UW: Using ARIA for Web Applications
  http://uw.edu/accessibility/web/aria/
- WAI-ARIA Authoring Practices + Design Patterns
  https://www.w3.org/TR/wai-aria-practices/
- W3C Notes on using ARIA in HTML
  http://w3c.github.io/aria-in-html/

"Explore with Hadi" meetups

- A friendly, informal gathering to explore websites, provide feedback, and discuss web accessibility
- University of Washington: http://uw.edu/accessibility/events/
- University of Illinois: https://itaccessibility.illinois.edu/collaborate/explore

ARIA

Thank you!

Jeane Marty
jeanem@uw.edu

Hadi Rangin
hadir@uw.edu

Pete Graff
pgraff@uw.edu

Special thanks to Terrill Thompson (tft@uw.edu) for his work in the development of this course

Exercise 5: Modal Dialogs

Directory

exercises/Exercise-5-Modal-Dialogs/

Files

- *_modal.html*
- *_modal.js*

Instructions

- In *_modal.html*, restructure the modal to make it more accessible (hint: consider the `role`, `aria-label`, and `aria-labelledby` attributes, as well as a certain recommended html element)
- In *_modal.js*, refactor to:
  - Initially place the focus on the first focusable element in the modal
  - "trap" the focus inside the modal for all subsequent tabs, so that focus toggles between the modal's OK and close buttons
  - Close the modal when either the escape or enter key is pressed

Solution

- *modal-solution.html*
- *modal-solution.js*
- *diff.txt*