Is Your Applicant Tracking System Accessible?

Disability Inclusion of Greater Kansas City

2019 Summit

Presented by George Calvert

How to Fix Common Accessibility Problems

This section provides technical solutions for the accessibility barriers discussed in the presentation. It is intended for use by web developers and designers.

Problem: Fields are not announced by screen readers

Solution: Associate fields with labels.

Explicitly associate an HTML control with its label by setting the *for* attribute of the <label> element to the id of the control. Here is an example:

```
<label for="firstname">First Name:</label><input type="text"
id="firstname">
```

You can also implicitly associate a label with a form control by including the control within the <label> element. Here is an example:

```
<label>First Name:<input type="text"></label>
```

Problem: When moving from field to field with the Tab key, the order of fields is not logical

Solution Avoid using positive *tabindex* attribute values.

The order that fields are presented to a user when tabbing through a form is based on the order they appear in the Document Object Model (DOM). The DOM order is based on the order controls appear in the source HTML code. Fixing tab order takes two steps:

1. Adjust the source code order of form fields to place them in the desired sequence in the DOM.
2. Ensure that all form fields do not have a *tabindex* attribute, or if they do that the value is 0. Both of these mean the field should appear in the DOM order.

Note: A tabindex value of -1 is used to indicate that the control should not receive focus when the user tabs through a form. It is used when a control only should receive focus programmatically.

Problem: User input is not validated before the form is submitted

Solution: Use client-side validation in an accessible manner.

Performing client-side validation is a general best practice. The following techniques will make validation results more accessible.

- Trigger validation when the form is submitted rather than in response to unblur events. This helps prevent screen reader users from experiencing confusion moving from field to field.
- Take advantage of the built-in validation provided in HTML 5 controls. Adding the required attribute to a form control provides accessible validation if the form is submitted without a value. Here is an example:

  ```
  <form action="#" method="get" name="validationExampleFields"
  autocomplete="off">
  <label for="fname">First Name:</label>
  <input type="text"required id="fname">
  <br>
  <label for="lname">Last Name:</label>
  <input type="text" required id="lname">
  <br>
  <input type="submit" value="Submit">
  </form>
  ```

- HTML 5 also provides many new input types such as email, url, number, tel, date and so forth. Using the required attribute with one of these types provides type specific validation.

  For example, in a < date > field, you could not get away with typing "dog.”

For more complex validation, be sure to implement these accessibility techniques:

- Failure/Success Summary: If errors are detected when the submit button is activated, display a count and list of the fields with errors at the top of the page. Implement this container as a <div> element with tabindex=-1. Use CSS to set colors and font styles to alert sighted users and use the JavaScript focus method to trigger the screen reader to announce the message.
- Display error messages next to each error field: Implement each error message as a dynamically created <span> element with the *aria-label* attribute set to the text of the error message. Use CSS to set colors and font styles to alert sighted users.
- Mark Invalid Fields: For each field with an error, set the *aria-invalid* attribute to true to ensure it is spoken by a screen reader when the user tabs to the field.

Issue: Buttons that use icons do not have text that can be read by a screen reader

Solution: Add alternative text to the icon.

Add the *alt* attribute to the <img> element used for the button. Set its value to reflect the action that will be taken when the button is activated. Here is an example: code:

```
<img src="magnifying_glass.png" alt="Search">
```

Issue: Fields are not arranged in logical groups

Solution: Use HTML grouping elements to group similar elements together.

Use the <fieldset> element to group related fields together and then use the <legend> element to label the group of fields. Here is an example:

```
<fieldset>
<legend>Mailing Address</legend>
<label for="street1">Street Address 1</label>
<input type="text" id="street1">
... additional fields....
</fieldset>
```

Issue: In a multi-stage workflow, the current step or stage is not identified

Solution: Explicitly identify the current stage in the workflow.

Consider using the following design pattern:

1. At the start of the workflow state how many steps there are and list them using the <ol> element. Provide a Begin button to advance to step 1.
2. Insert "Step #" and the name of the step in the <title> and <h1> elements for each step. Provide a Back button to return to the prior step and a Next button to advance to the next step.
3. Validate user input when the Next button is activated and do not let users advance if any of the data values are invalid. Make sure error messages can be announced by a screen reader and appear close to the invalid field.
4. Allow the user to return to a prior step and change the earlier input.
5. For the final step, change the name of the Next button to Finish.
6. Once the final step has been completed and the user activates the Finish button display a message indicating whether the intended action of the workflow was successful.

Issue: Information that is updated on a page is not announced

Solution: Use an accessibility technique whenever a page is updated asynchronously.

In traditional web development, forms are submitted to a web server for processing and the server returns a new page based on the form data submitted. A screen reader begins reading the returned page as soon as it is loaded.

However, many web sites today use a Single Page Application (SPA) that relies on Ajax calls inside of JavaScript frameworks to submit and update page content. This has a number of advantages, but can create accessibility issues when content is updated without a new page being loaded. How does a screen reader user know that content somewhere within the SPA has been updated?

There are two types of solutions for this situation. One approach is to shift the user's attention to the new content. Once the content has been rendered, use the JavaScript focus method to set focus to the newly rendered content.

The second solution enables a screen reader to announce dynamic content without moving the current focus. This is accomplished using a technique called ARIA live regions. This is a non-semantic container such as a <div> that contains certain ARIA attributes that tell the screen reader when to announce content that is dynamically inserted into the container. The content can be visually displayed or hidden using CSS but will always be announced by a screen reader.

See the ARIA resources listed below for information on implementing ARIA live regions.

Issue: The color contrast between text and background is insufficient for low vision users

Solution: Choose colors with sufficiently high contrast ratios.

The contrast between two colors is called a contrast ratio. For normal text (less than 18 points for normal weight or less than 14 points for bold) the contrast ratio between the color of the text and the background color must be at least 4.5 to 1, and for larger text the ratio must be at least 3 to 1.

Contrast ratios can be calculated using the free tool mentioned below.

Issue Information is conveyed only through color, such as red warning text

Solution: Add text that conveys the same meaning as the color.

For example, prefix the red message with the word "Warning."

Issue: Videos and audio messages about the organization lack captions and text transcripts

Solution: Provide captions and text transcripts.

If the video shows a person delivering a spoken message, captions must be inserted into the video. Include relevant background such as applause or laughter. Captions can be permanently inserted into the video during production and will be displayed to all users. Alternatively, captions can be included in a separate file that is delivered along with the video itself. These are known as closed captions and can be turned on and off and their visual styling set by the user.

Additionally, a complete transcript of the spoken contents of a video or audio file must also be provided. For audio-only files, this is the only way to make them accessible to people who are deaf or hard of hearing. For video files it is the only way to make them accessible to people with deaf-blindness. As with captions, include relevant background sounds and identify the speaker if there are several speakers in the video. The transcript can appear on the page below the media player, or can be accessed from a link above or below the player.

For video that contains a mix of dialog, action, characters and scenery an additional assistive technology called audio descriptions is required to meet accessibility standards. Such videos are unlikely to be included in an ATS.

Issue: Data presented in tables lack headings

Solution: Use table headers in data tables.

Screen readers are able to make data tables accessible provided they contain <th> elements designating the column headers (the first row of data) and row headers (the first column of data). When marked up this way, the user is able to know the row and column headings for each cell of data.

Issue: CAPTCHAs that require sight or hearing

Solution: Remove CAPTCHAS or use non-sensory versions.

CAPTCHAS are always an accessibility problem, so avoid them if at all possible. If they are truly required, present a non-sensory version, such as answering this question: Mary has 3 cats and one dog. Jane gives Mary another dog. How many dogs does Mary have?

A series of random forms of this type of CAPTCHA can be used to make this technique more robust.

Resources for Managing Web Accessibility

This section contains links to resources for managing the accessibility of a web application such as an ATS.

Web Accessibility Testing Tools

Title: WAVE Web Accessibility Tool

Description: A free automated testing tool from WebAIM

URL: https://wave.webaim.org/

Title: WebAIM Color Contrast Checker

Description: A free tool from WebAIM to calculate the contrast ratio between 2 colors

URL: https://webaim.org/resources/contrastchecker/

Title: Photosensitive Epilepsy Analysis Tool

Description: A free, downloadable tool for analyzing whether web content is likely to cause seizures.

URL: https://trace.umd.edu/peat

Using Screen Readers to Test Web Accessibility

Below are resources for installing and learning to use screen readers to manually test web accessibility.

NVDA:

Title: WebAIM: Using NVDA to Evaluate Web Accessibility

Description: An introductory tutorial on using NVDA for accessibility testing

URL: https://webaim.org/articles/nvda/#maincontent

Title: NVAccess

Description: The official NVDA site – it contains the download link

URL: https://www.nvaccess.org/

JAWS:

Title: WebAIM: Using JAWS to Evaluate Web Accessibility

Description: An introductory tutorial on using JAWS for accessibility testing

URL: https://webaim.org/articles/jaws/

Title: JAWS Download

Description: Download an evaluation copy of JAWS

URL: https://www.freedomscientific.com/Downloads/JAWS

Windows Narrator:

Title: Complete Guide to Narrator

Description: Microsoft's user guide for its built-in Narrator screen reader

URL: https://support.microsoft.com/en-us/help/22798/windows-10-complete-guide-to-narrator

Voiceover – Mac:

Title: WebAIM: Using Voiceover to Evaluate Web Accessibility

Description: An introductory tutorial on using Voiceover for accessibility testing on a Mac

URL: https://webaim.org/articles/voiceover/

Additional General Resources

Disabilities and the Web:

Title: How People with Disabilities Use the Web

Description: An overview by the W3C WAI of how people with a variety of disabilities use assistive technology to interact with the web

URL: https://www.w3.org/WAI/people-use-web/

Web Content Authoring Guidelines:

Title: WCAG at a Glance

Description: A brief paraphrased presentation of the WCAG principles and guidelines prepared by the W3C WAI.

URL: http://www.w3.org/WAI/standards-guidelines/wcag/glance/

Title: WEBAim – Web Accessibility in Mind

Description: WEBAim is a non-profit founded in 1999 at the Center for Persons with Disabilities at Utah State. It provides web accessibility training and consulting, and its web site contains a large collection of articles, survey results, newsletters and an archive of mailing list posts.

URL: https://webaim.org/

Legal Issues:

Title: Legal Updates – Law Offices of Lainey Feingold

Description: Lainey Feingold is one of the leading attorneys in the area of digital access for people with disabilities. She is a champion of using structured negotiations in lieu of litigation to achieve positive change.

URL: https://www.lflegal.com/category/accessibility-laws-and-regulations/legal-updates/#content

Additional Technical Resources

Web Accessibility Blogs:

The following blogs are published by the three leading accessibility consulting firms. Staying current with each blog ensures you're up-to-date on web accessibility, and the blog archives contain a great deal of useful information.

Title: Level Access Accessibility Blog

Company: Level Access (formerly SSB Bart Group)

URL: https://www.levelaccess.com/blog/

Title: Deque Blog

Company: Deque Systems Inc.

URL: https://www.deque.com/blog/#content

Title: Blog TPG

Company: The Paciello Group

URL: https://developer.paciellogroup.com/blog/

WCAG:

Title: Web Content Accessibility Guidelines 2.1

Description: The current, official WCAG document

URL: https://www.w3.org/TR/WCAG21/

Title: How to Meet WCAG 2.1

Description: An interactive document that allows filtering to provide access to targeted WCAG Success Criteria.

URL: http://www.w3.org/WAI/WCAG21/quickref/

Title: WebAim WCAG Checklist

Description: A checklist approach to working with the WCAG.

URL: https://webaim.org/standards/wcag/checklist/

Accessible Rich Internet Applications (ARIA):

ARIA is a complicated subject, and the resources listed below appear in order from introductory to advanced.

Title: ARIA, what does it do for me, and what not?

Description: A quick overview of ARIA from one of the accessibility thought leaders at Mozilla

URL: https://www.marcozehe.de/2014/03/27/what-is-wai-aria-what-does-it-do-for-me-and-what-not/#content

Title: Using ARIA

Description: A more detailed guide to ARIA from its creators

URL: https://w3c.github.io/using-aria/

Title: ARIA Authoring Practices 1.1

Description: The definitive guide from the W3C WAI on how to use ARIA to implement custom widgets

URL: https://www.w3.org/TR/wai-aria-practices-1.1/

Title: ARIA 1.1 Specification

Description: The official ARIA specification from the W3C WAI. Not for the faint of heart.

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

Mobile Apps:

Title: Mobile Accessibility: How WCAG 2.0 and Other W3C/WAI Guidelines Apply to Mobile

Description: While the WCAG explicitly applies only to web applications, the WAI prepared this document as guidance on how to apply WCAG principles, guidelines and success criteria to native mobile apps.

URL: https://www.w3.org/TR/mobile-accessibility-mapping/#introduction