Lesson:

Accessibility in HTML

PW SKILLS

Topics

- What is accessibility
- Why to use accessibility
- Best practices to follow for accessibility

What is accessibility

Accessibility in HTML refers to the practice of designing and coding web content in a way that ensures it can be easily accessed, understood, and interacted with by people of diverse abilities, including those with disabilities. This includes using semantic markup, providing alternative text for images and videos, and making sure your web pages are navigable using a keyboard.

There are some assistive devices that play a major role in providing accessibility.

1. **Screen Reader:** A screen reader is software that reads out loud the content of a web page to individuals who are visually impaired. It can also interpret and communicate information about graphics, multimedia, and other elements on the page.
2. **Voice recognition software:** Voice recognition software enables users to navigate web pages and input text using voice commands. This technology is particularly useful for individuals with mobility impairments or those who have difficulty using a keyboard or mouse.
3. **Keyboard alternatives:** Keyboard alternatives such as sip-and-puff devices, head-tracking devices, and eye-tracking devices allow individuals with physical disabilities to navigate and interact with web pages without the use of a traditional keyboard or mouse.

Why to use accessibility

There are many reasons why you should use accessibility. Here are some of the important points

- To identify accessibility issues: - It can help you to identify accessibility issues in your website or web application before users with disabilities encounter them.
- To meet accessibility standards - it can help you to ensure that our website or web application meets accessibility standards.
- To improve usability - Accessibility automated testing software tools can help us to improve the usability of your website or web application for everyone, not just users with disabilities, as a result, it can help you to increase user satisfaction and engagement.

Best practices to follow for accessibility

Understanding Accessibility Guidelines and familiarisation of Web Content Accessibility Guidelines (WCAG), that provide the international standard for web accessibility. Understanding the different levels of standard rules and the principles, guidelines, and success criteria outlined in the guidelines of best practices to follow for accessibility.

Following the guidelines of the WCAG includes the best practices to follow for accessibility.

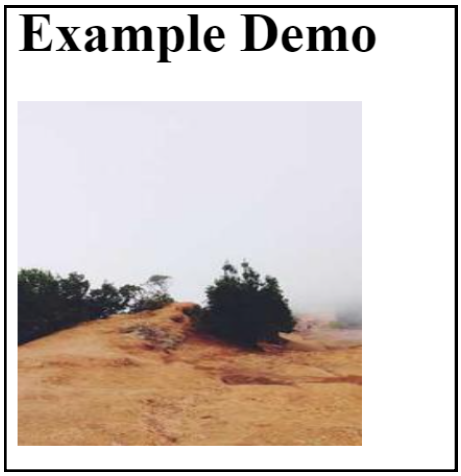
Some of the best practices are as follows -

1. Image alternative text - Add descriptive alternative text (alt text) to all images using the alt attribute

Example -

```
<img src="https://picsum.photos/200/300" alt="Random Pic"width="200" height="200" />
```

Browser output



If the above code is not working due to connectivity issues or a mistake in the URL link or Path. The alternate text will be displayed.
i.e.

2. Ensuring proper heading structure or Semantic Structure of HTML- practice the use of proper semantic structure, by using appropriate HTML elements like **<h1>, <nav>, <main>, <article>, <button>, and <label>** to convey the purpose and role of each element.

Example of HTML with proper semantic structure.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Delicious Chocolate Chip Cookies</title>
  </head>
  <body>
```

```
  <!— Heading section —>
<header>
      <h1>Delicious Chocolate Chip Cookies</h1>
 </header>

  <!— navbar section  —>
    <nav>
      <ul>
        <li><a href="#ingredients">Ingredients</a></li>
        <li><a href="#instructions">Instructions</a></li>
      </ul>
    </nav>

  <!— Main section —>
    <main>
      <article id="ingredients">
        <h2>Ingredients</h2>
        <p>Lorem ipsum dolor sit.....</p>
      </article>

      <article id="instructions">
        <h2>Instructions</h2>
        <p>Lorem ipsum dolor sit sciunt distinctio unde....</p>
      </article>
    </main>

  <!— Footer section —>
    <footer>
      <p>&copy; 2023 Delicious Cookie Recipes</p>
    </footer>
  </body>
</html>
```

3. Making links descriptive - to make a link descriptive in HTML, meaning full and descriptive text in the anchor **<a>** element should be used.

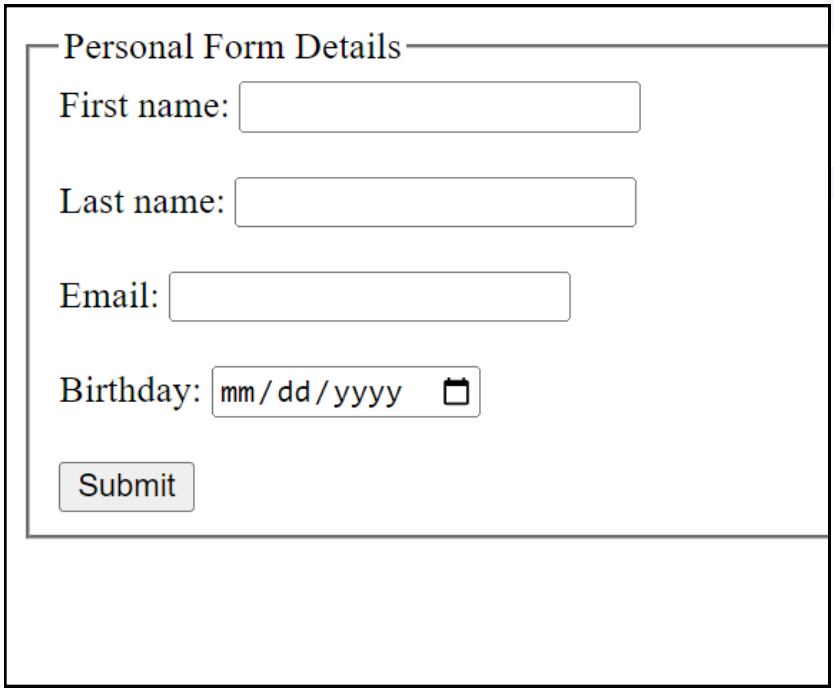
```
  <p>
      Check out our latest blog post about
      <a
        href="https://www.example.com/latest-blog"
        title="Read the latest blog post" >web-development</a>
  </p>
```

4. Accessibility in Form - use of **<label>** tag or **aria-label** attribute and Form fields should be properly grouped with **<fieldset>** and **<legend>** elements.

Example

```
<form action="/">
    <fieldset>
      <legend>Personal Form Details</legend>
      <label for="fname">First name:</label>
      <input type="text" id="fname" name="fname" /><br /><br />
      <label for="lname">Last name:</label>
      <input type="text" id="lname" name="lname" /><br /><br />
      <label for="email">Email:</label>
      <input type="email" id="email" name="email" /><br /><br />
      <label for="birthday">Birthday:</label>
      <input type="date" id="birthday" name="birthday" /><br /><br />
      <input type="submit" value="Submit" aria-label="Submit form" />
    </fieldset>
  </form>
```

Browser output -



5. Text resizes and Zoom testing - The website pages should be checked by resizing text or zooming in/out to ensure that the content remains readable and usable without causing layout or functionality issues.
6. Video and Audio Accessibility - Provide captions, transcripts, and audio descriptions for multimedia content. Ensure that video players are accessible and controllable with the keyboard.

4. HTML tables - Ensure that tables are used for tabular data and properly marked up with appropriate headers and scope attributes to provide context to screen readers.

Example -

```
<table>
      <caption>
        Monthly Employee Attendance
      </caption>
      <thead>
        <tr>
          <th scope="col">Month</th>
          <th scope="col">Product A</th>
          <th scope="col">Product B</th>
          <th scope="col">Product C</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <th scope="row">January</th>
          <td>19 days</td>
          <td>24 days</td>
          <td>19 days</td>
        </tr>
        <tr>
          <th scope="row">February</th>
          <td>24 days</td>
          <td>21 days</td>
          <td>19 days</td>
        </tr>
        <!— More rows go here —>
      </tbody>
    </table>
```

Browser output -

Monthly Employee Attendance

Month	Product A	Product B	Product C
January	19 days	24 days	19 days
February	24 days	21 days	19 days

8. Colour contrast - Verify that the colour contrast between text and background meets the WCAG contrast requirements. Ensure that information conveyed through colour is also available via other means (e.g., labels, symbols).

9. Tab-index - The tab-index is an HTML attribute that can be used to specify the order of navigating an HTML element when the tab button is pressed. The tab-index can be used on any HTML element but it should be used only on specific elements where it is needed.

It's best not to use the **"tabindex"** attribute on non-interactive elements to make them work like interactive elements when users use the keyboard, For example, don't use a ** ** to represent a button when you should use the **<button>** element

It accepts an integer as a value, with different results depending on the integer's value.

- tabindex="-1," indicates that the element cannot be accessed using standard keyboard navigation in a sequential manner.
- tabindex="0" makes the element focusable through keyboard navigation, following any elements with positive **tabindex** values
- A positive **tabindex** value determines the order of keyboard focus. Lower values are focused before higher ones. When elements share the same positive **tabindex**, their order follows the document source.

Recommend using only "0" and "-1" as **tabindex** values to maintain a logical and accessible tab order. Avoid using values greater than 0 or CSS properties that alter the order of focusable elements, as it can hinder keyboard navigation and accessibility for users who rely on it.

Example
Index.html (Basic example)

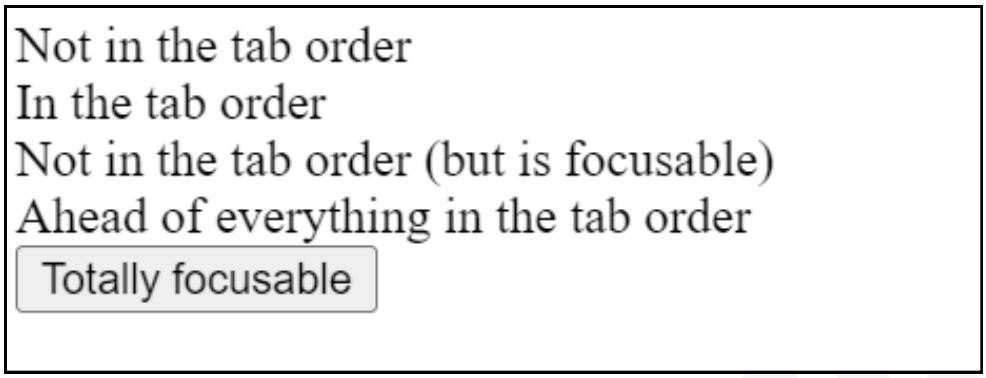
```
<body>
    <div tabindex="0">
      Click here!
    </div>
    <div tabindex="0" >
      Click right here!
    </div>
    <div tabindex="0" >
      Submit here
    </div>
</body>
```

From the above example, ** ** buttons have the ability to be focused (including via tab) by giving each on the attribute tabindex= "0". Basically, the tabindex attribute is primarily intended to allow tabbable elements to have a custom tab order.

More example of the tabindex
index.html

```
<body>
<div>Not in the tab order</div>
  <div tabindex="0">In the tab order</div>
  <div tabindex="-1">Not in the tab order (but is focusable)</div>
  <div tabindex="5">Ahead of everything in the tab order</div>
  <button>Totally focusable</button>
</body>
```

Browser output -



In the above example, the **tabindex="0"** option allows elements that are not usually able to be focused via the keyboard to become focusable. This value of the tab index is particularly beneficial. The **tabindex="-1**" option enables elements that are not typically focusable to receive focus programmatically, such as through JavaScript or as the target of links. The **tabindex="5"** moves the order ahead of everything. Additionally, the button is focusable by default.