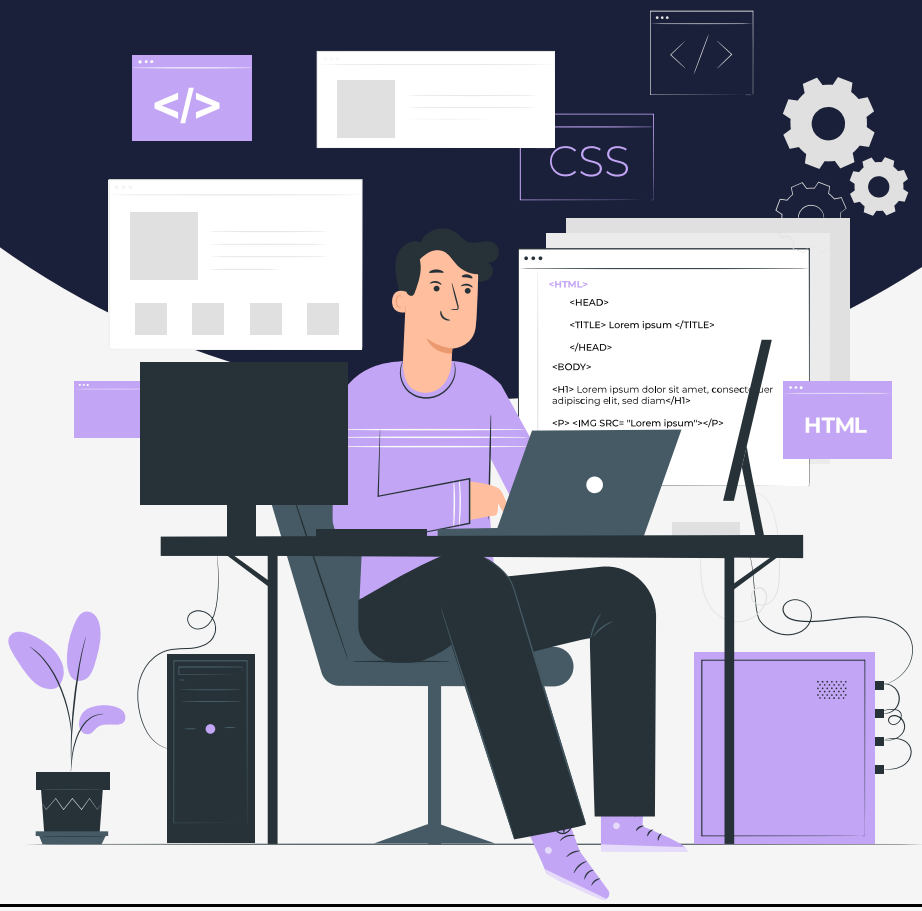
Lesson:

Accessibility in HTML



PW SKILLS

Topics Covered

- What is Web Accessibility?
- Several ways to improve accessibility of HTML
  - Text content
  - Page layout
  - UI control
  - alt attribute
  - title attribute
- Keyboard accessibility
- ARIA roles

What is web accessibility?

Web accessibility refers to the **practice of designing** and developing web applications that can be accessed and used by people with disabilities or different needs, without barriers or limitations.

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

1. **Screen Reader** : A screen reader is a 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.

Several ways to Improving accessibility of HTML

Text content

Having well-structured content that includes headings, paragraphs, and lists is one of the most helpful accessibility features for users who rely on screen readers. An example of a good semantic structure may resemble the following.

Example:-

```
Unset
<h2>Heading 1</h2>
<p> This is paragraph under the heading 1 </p>
<h2>Heading 2</h2>
<p> This is paragraph under the heading 2 </p>
```

Note: The screen reader reads each header out as you progress through the content, notifying you what a heading is and what is a paragraph etc.

Page layouts

Although it is possible to design a layout using nested ** ** elements, it is preferable to utilise proper sectioning elements to encapsulate your main navigation **(<nav>)**, footer **(<footer>)**, repeated content units **(<article>)**, and other relevant content.

These elements offer additional semantics to screen readers and other assistive tools, providing users with more context and information about the content they are browsing.

Example:-

```
Unset
<header>
  <h1>Header</h1>
</header>

<nav>
  <!-- main navigation in here -->
</nav>

<!-- Here is our page's main content -->
<main>
  <!-- It contains an article -->
  <article>
    <h2>Article heading</h2>

    <!-- article content in here -->
  </article>

  <aside>
    <h2>R
  <!-- aside content in here -->
  </aside>
</main>

<!-- And here is our main footer that is used across all the
pages of our website -->

<footer>
  <!-- footer content in here -->
</footer>
```

UI controls

When referring to UI controls, we are discussing the essential elements of web documents that users engage with, such as **buttons, links, and form controls.**

This section will cover fundamental accessibility considerations that one should keep in mind while developing these controls.

Example:-

```
Unset
<-- Link --!>

<p>This is a link to <a
href="https://www.mozilla.org">Mozilla</a>.</p>

<-- Buttons --!>
<p>
  <button data-message="First button">Click here!</button>
  <button data-message="Second button">Click right here!</button>
  <button data-message="Third button">Submit!</button>
</p>

<-- Form --!>
<form>
  <div>
    <label for="name">Fill in your name:</label>
    <input type="text" id="name" name="name" />
  </div>
  <div>
    <label for="age">Enter your age:</label>
    <input type="text" id="age" name="age" />
  </div>
  <div>
    <label for="mood">Choose your mood:</label>
    <select id="mood" name="mood">
      <option>Happy</option>
      <option>Sad</option>
      <option>Angry</option>
      <option>Worried</option>
    </select>
  </div>
</form>
```

This means define links, buttons, form elements, appropriately with proper labels using <label> tag.

Alt attribute

The alt attribute is an important attribute in HTML that is used to provide alternative text for an image if the image cannot be displayed or if the user is using a screen reader to access the page.

Example:

```
Unset
<img src="sample.jpg" alt="A sample image">
```

The alternate text will be displayed in case the user is not able to load the image.

title attribute

The title attribute is an important attribute in HTML that can be used to provide additional information about an element, such as a link or an image.
The title attribute can be used to provide additional information about the image or element.

Example:

```
Unset
 <h1 title="This is the h1 tag">Hover me</h1>
```

On hover of the <h1> tag the user will get a pop up of “This is the h1 tag”.

Keyboard accessibility

Keyboard accessibility is an important aspect of web accessibility that ensures users with mobility or visual impairments, or those who cannot use a mouse, are able to navigate and interact with web content using only a keyboard.

We can use **index** and **role** attributes to increase keyboard accessibility,

Let's take an example to understand it,

```
Unset
<div data-message="First button" tabindex="0" role="button">
  Click here!
</div>
<div data-message="Second button" tabindex="0" role="button">
  Click right here!
</div>
<div data-message="Third button" tabindex="0" role="button">
  Submit here
</div>
```

In the above example, ** ** buttons have the ability to be focused (including via tab) by giving each one the attribute
tabindex="0", and also **role="button"** so screen reader users know they can focus on and interact with the element

Basically, the **tabindex** attribute is primarily intended to allow tabbable elements to have a custom tab order (specified in positive numerical order), instead of just being tabbed through in their default source order.

There are two additional options available for tabindex:

1. **tabindex="0"** - this option allows elements that are not usually able to be focused via the keyboard to become focusable. This value of tabindex is particularly beneficial.
2. **tabindex="-1"** - this option enables elements that are not typically focusable to receive focus programmatically, such as through JavaScript, or as the target of links.

Accessible Rich Internet Applications (ARIA)

The Accessible Rich Internet Applications (ARIA) comprises **roles** that establish methods for improving the accessibility of web content and web applications, for individuals with disabilities.

Example 1 - List roles
The ARIA **list role** can be used to identify a list of items. It is normally used in conjunction with the **listitem role**, which is used to identify a list item contained inside the list.

For content that comprises an external container enclosing a group of items within it, assistive technologies can recognize the **“list”** and **“listitem”** containers, respectively.

```
Unset
<div role="list">
  <div role="listitem">List item 1</div>
  <div role="listitem">List item 2</div>
  <div role="listitem">List item 3</div>
</div>
```

Example 2 - img role
The ARIA **img role** can be used to identify multiple elements inside page content that should be considered as a single image.

These elements could be images, code snippets, text, emojis, or other content that can be combined to deliver information in a visual manner.

```
Unset
<div role="img" aria-label="Description of the overall image">
  <img src="image1.png" alt="alt text"/>
  <img src="image2.png" alt="alt text" />
</div>
```

Similarly, we have other ARIA roles like link roles, grid roles, form roles and many more, we use and study about them as per our requirement.