HTML Accessibility Review

Introduction to Accessibility

- **Web Content Accessibility Guidelines**: The Web Content Accessibility Guidelines (WCAG) are a set of guidelines for making web content more accessible to people with disabilities. The four principles of WCAG are POUR which stands for Perceivable, Operable, Understandable, and Robust.

Assistive Technology for Accessibility

- **Screen readers**: Software programs that read the content of a computer screen out loud. They are used by people who are blind or visually impaired to access the web.
- **Large text or braille keyboards**: Used by people with visual impairments to access the web.
- **Screen magnifiers**: Software programs that enlarge the content of a computer screen. They are used by people with low vision to access the web.
- **Alternative pointing devices**: Used by people with mobility impairments to access the web. This includes devices such as joysticks, trackballs, and touchpads.
- **Voice recognition**: Used by people with mobility impairments to access the web. It allows users to control a computer using their voice.

Accessibility Auditing Tools

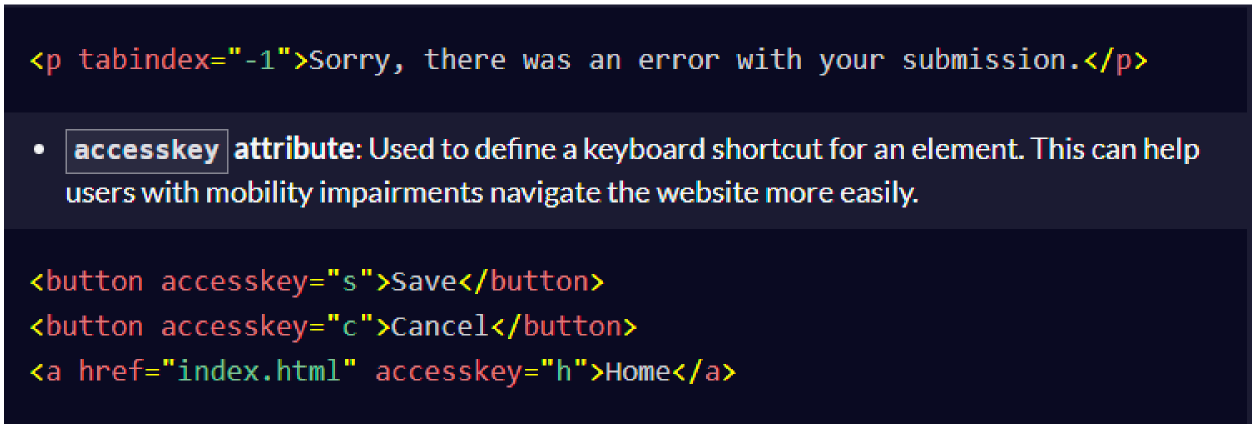
- **Common Accessibility Tools**: Google Lighthouse, Wave, IBM Equal Accessibility Checker, and axe DevTools are some of the common accessibility tools used to audit the accessibility of a website.

Accessibility Best Practices

- **Proper heading level structure**: You should use proper heading levels to create a logical structure for your content. This helps people using assistive technologies understand the content of your website.
- **Accessibility and Tables**: When using tables, you should use the th element to define header cells and the td element to define data cells. This helps people using assistive technologies understand the structure of the table. With
the caption element, you can write the caption (or title) of a table, so users, especially those who use assistive technologies, can quickly understand the table's

purpose and content. You should place the caption element immediately after the opening tag of the table element. This way, screen readers and other assistive technologies can provide more context by announcing the caption before reading the content.

- **Importance for inputs to have an associated label**: You should use the label element to associate labels with form inputs. This helps people using assistive technologies understand the purpose of the input.
- **Importance of good alt text**: You should use the alt attribute to provide a text alternative for images. This helps people using assistive technologies understand the content of the image.
- **Importance of good link text**: You should use descriptive link text to help users understand the purpose of the link. This helps people using assistive technologies understand the purpose of the link.
- **Best practices for making audio and video content accessible**: You should provide captions and transcripts for audio and video content to make it accessible to people with hearing impairments. You should also provide audio descriptions for video content to make it accessible to people with visual impairments.
- **tabindex attribute**: Used to make elements focusable and define the relative order in which they should be navigated using the keyboard. It is important to never use the tabindex attribute with a value greater than 0. Instead, you should either use a value of 0 or -1. For more information, review the prior lecture video on keyboard accessibility.



```
<p tabindex="-1">Sorry, there was an error with your submission.</p>
```

- `accesskey` **attribute**: Used to define a keyboard shortcut for an element. This can help users with mobility impairments navigate the website more easily.

```
<button accesskey="s">Save</button>
<button accesskey="c">Cancel</button>
<a href="index.html" accesskey="h">Home</a>
```

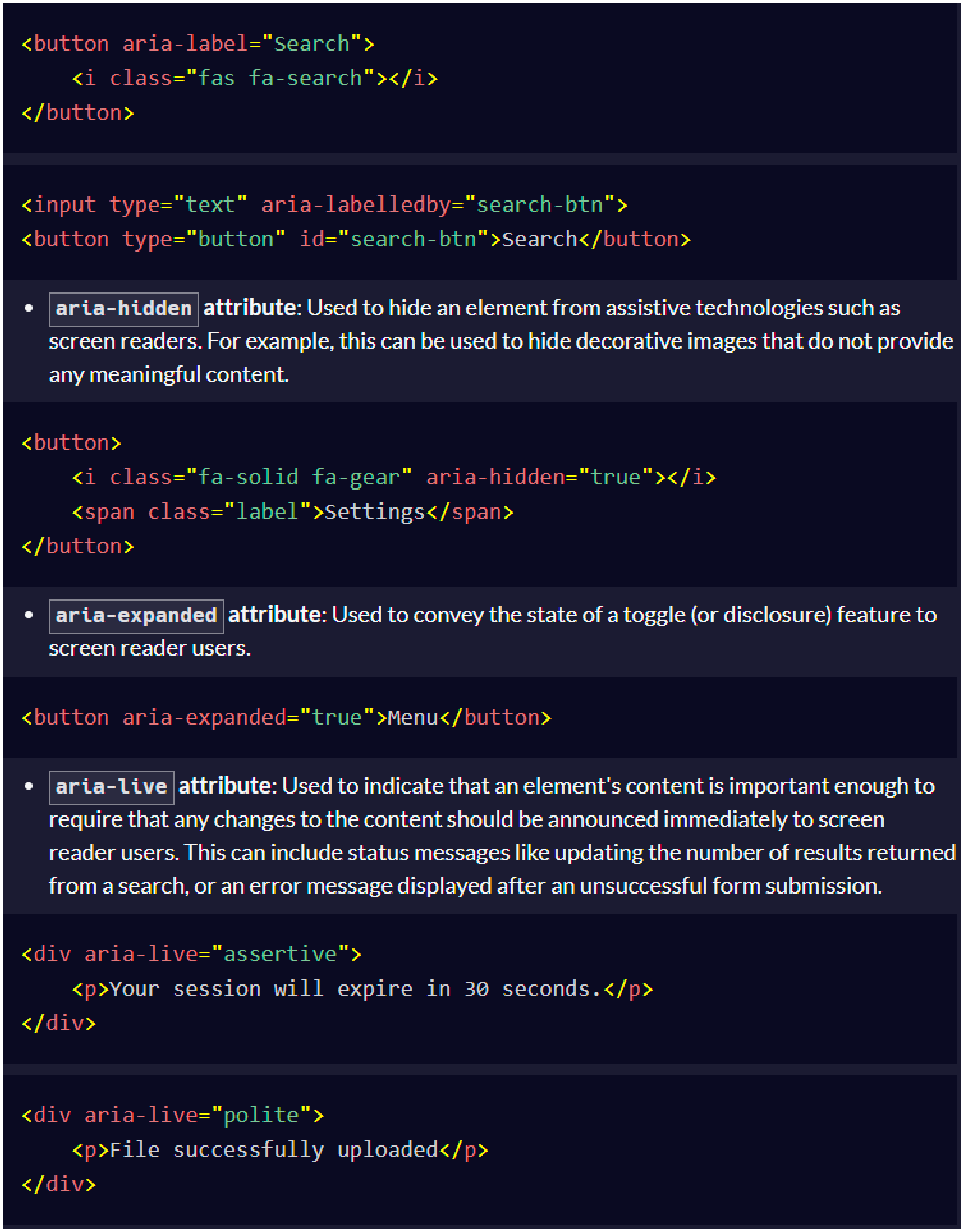
WAI-ARIA, Roles, and Attributes

- **WAI-ARIA**: It stands for Web Accessibility Initiative - Accessible Rich Internet Applications. It is a set of attributes that can be added to HTML elements to improve accessibility. It provides additional information to assistive technologies about the purpose and structure of the content.

- **ARIA roles**: A set of predefined roles that can be added to HTML elements to define their purpose. This helps people using assistive technologies understand the content of the website. Examples include role="tab", role="menu", and role="alert".

There are six main categories of ARIA roles:

- **Document structure roles**: These roles define the overall structure of the web page. With these roles, assistive technologies can understand the relationships between different sections and help users navigate the content.
- **Widget roles**: These roles define the purpose and functionality of interactive elements, like scrollbars.
- **Landmark roles**: These roles classify and label the primary sections of a web page. Screen readers use them to provide convenient navigation to important sections of a page.
- **Live region roles**: These roles define elements with content that will change dynamically. This way, screen readers and other assistive technologies can announce changes to users with visual disabilities.
- **Window roles**: These roles define sub-windows, like pop up modal dialogues. These roles include alertdialog and dialog.
- **Abstract roles**: These roles help organize the document. They're only meant to be used internally by the browser, not by developers, so you should know that they exist but you shouldn't use them on your websites or web applications.
- **aria-label and aria-labelledby attributes**: These attributes are used to give an element a programmatic (or accessible) name, which helps people using assistive technology (such as screen readers) understand the purpose of the element. They are often used when the visual label for an element is an image or symbol rather than text. aria-label allows you to define the name directly in the attribute while aria-labelledby allows you to reference existing text on the page.



```html
<button aria-label="Search">
    <i class="fas fa-search"></i>
</button>
```

```html
<input type="text" aria-labelledby="search-btn">
<button type="button" id="search-btn">Search</button>
```

- `aria-hidden` **attribute**: Used to hide an element from assistive technologies such as screen readers. For example, this can be used to hide decorative images that do not provide any meaningful content.

```html
<button>
    <i class="fa-solid fa-gear" aria-hidden="true"></i>
    <span class="label">Settings</span>
</button>
```

- `aria-expanded` **attribute:** Used to convey the state of a toggle (or disclosure) feature to screen reader users.

```html
<button aria-expanded="true">Menu</button>
```

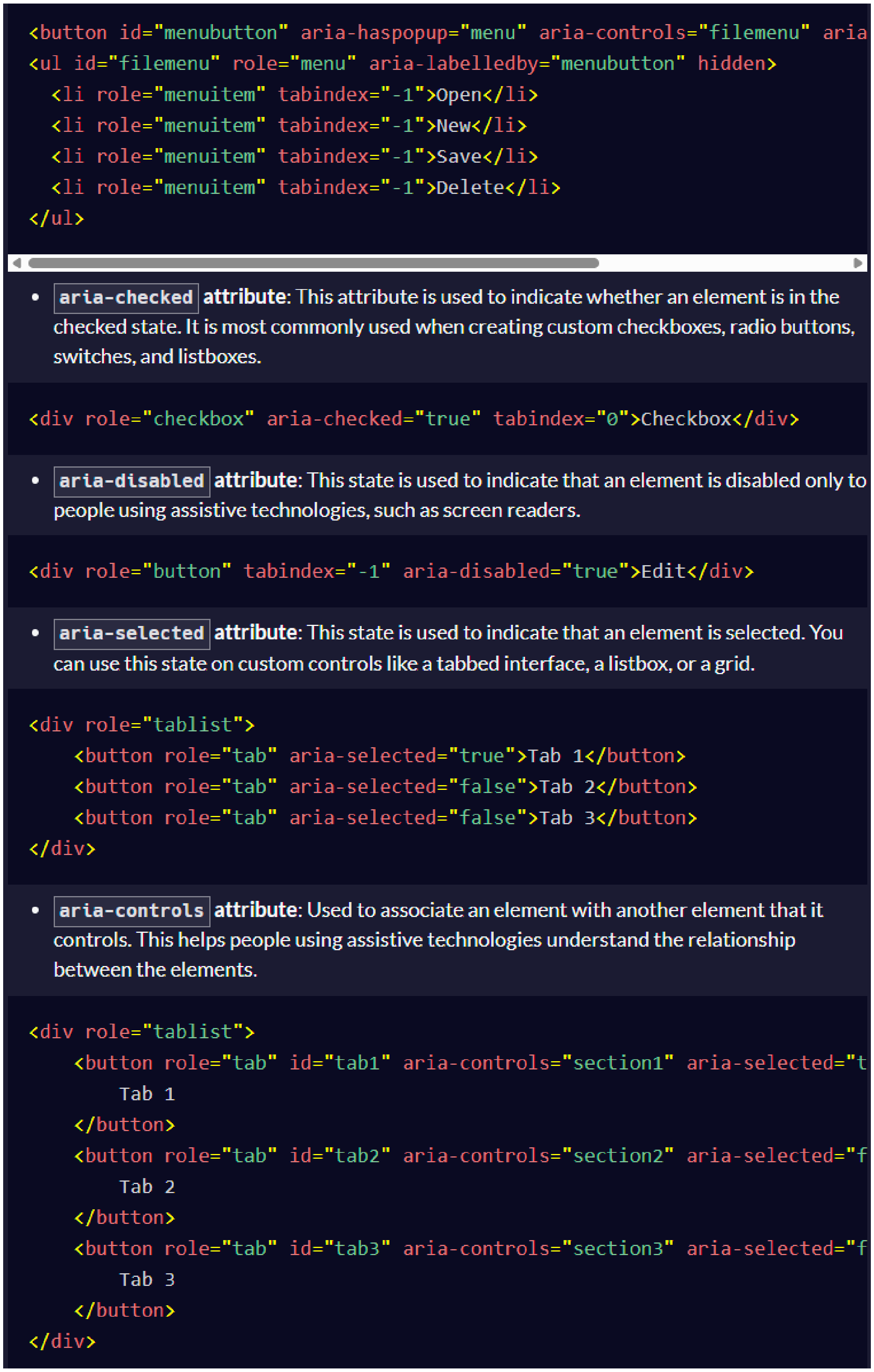
- `aria-live` **attribute**: Used to indicate that an element's content is important enough to require that any changes to the content should be announced immediately to screen reader users. This can include status messages like updating the number of results returned from a search, or an error message displayed after an unsuccessful form submission.

```html
<div aria-live="assertive">
    <p>Your session will expire in 30 seconds.</p>
</div>
```

```html
<div aria-live="polite">
    <p>File successfully uploaded</p>
</div>
```

- **Common ARIA states**: Common ARIA states include aria-haspopup, aria-checked, aria-disabled, and aria-selected. These attributes can be used to indicate the state of an element and help people using assistive technologies understand the content of the website.

- **aria-haspopup attribute**: This state is used to indicate that an interactive element will trigger a pop-up element when activated. You can only use the aria-haspopup attribute when the pop-up has one of the following roles: menu, listbox, tree, grid, or dialog. The value of aria-haspopup must be either one of these roles or true, which is the same as menu.



```
<button id="menubutton" aria-haspopup="menu" aria-controls="filemenu" aria
<ul id="filemenu" role="menu" aria-labelledby="menubutton" hidden>
  <li role="menuitem" tabindex="-1">Open</li>
  <li role="menuitem" tabindex="-1">New</li>
  <li role="menuitem" tabindex="-1">Save</li>
  <li role="menuitem" tabindex="-1">Delete</li>
</ul>
```

- `aria-checked` **attribute**: This attribute is used to indicate whether an element is in the checked state. It is most commonly used when creating custom checkboxes, radio buttons, switches, and listboxes.

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

- `aria-disabled` **attribute**: This state is used to indicate that an element is disabled only to people using assistive technologies, such as screen readers.

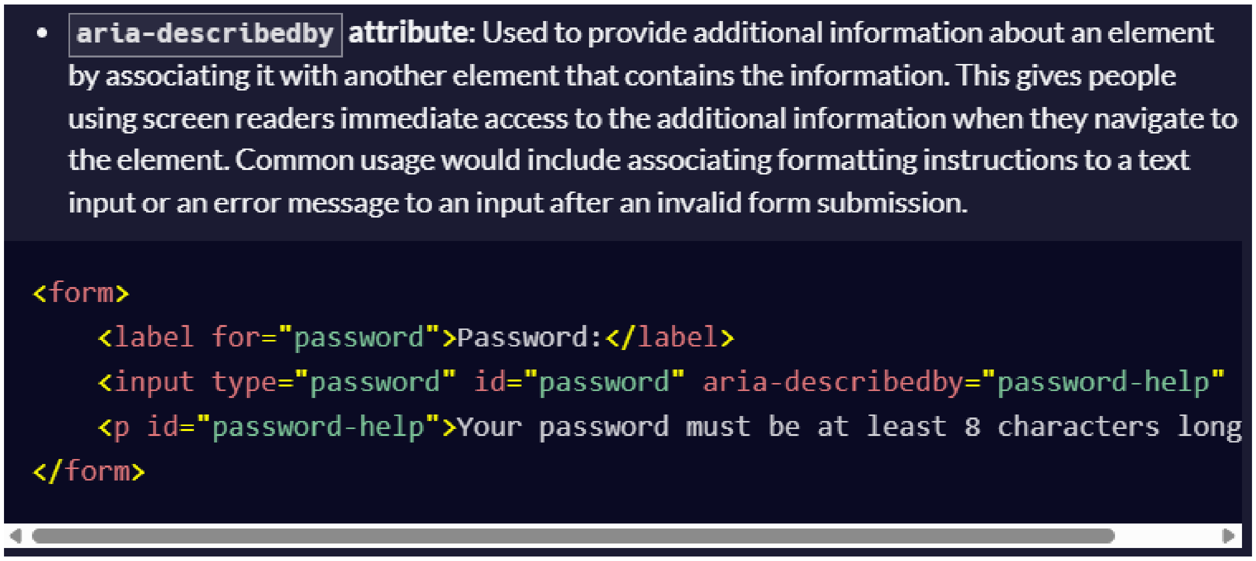
```
<div role="button" tabindex="-1" aria-disabled="true">Edit</div>
```

- `aria-selected` **attribute**: This state is used to indicate that an element is selected. You can use this state on custom controls like a tabbed interface, a listbox, or a grid.

```
<div role="tablist">
    <button role="tab" aria-selected="true">Tab 1</button>
    <button role="tab" aria-selected="false">Tab 2</button>
    <button role="tab" aria-selected="false">Tab 3</button>
</div>
```

- `aria-controls` **attribute**: Used to associate an element with another element that it controls. This helps people using assistive technologies understand the relationship between the elements.

```
<div role="tablist">
    <button role="tab" id="tab1" aria-controls="section1" aria-selected="t
        Tab 1
    </button>
    <button role="tab" id="tab2" aria-controls="section2" aria-selected="f
        Tab 2
    </button>
    <button role="tab" id="tab3" aria-controls="section3" aria-selected="f
        Tab 3
    </button>
</div>
```



- `aria-describedby` **attribute**: Used to provide additional information about an element by associating it with another element that contains the information. This gives people using screen readers immediate access to the additional information when they navigate to the element. Common usage would include associating formatting instructions to a text input or an error message to an input after an invalid form submission.

```
<form>
    <label for="password">Password:</label>
    <input type="password" id="password" aria-describedby="password-help"
    <p id="password-help">Your password must be at least 8 characters long
</form>
```