Frances “Reagan” James; 2025-RTT-23

Reflections on Learn-Accessible-Web-Design

1. The Accessibility enhancements that were the most challenging to implement were ones in the Aside: Accessible Javascript, Aside: Hiding content, and the Aside: Skip Navigation Link Part 1 and 2 sections of the course. I believe these were the most challenging sections of the course for me as I am not yet familiar with Javascript and how it is implemented. Not yet knowing how to correctly write the correct html and css that points to/interacts with the JavaScript functions in the index.js file was challenging as I am brand new to JavaScript.
2. Aria attributes improve the experience for users relying on assistive technologies by ensuring any elements not written in Semantic HTML have ARIA roles and properties which provide useful information about what the webpage contents are. If a custom button is made by a developer that is not written in Semantic HTML, using ARIA roles to define the purpose of an element button helps those who need assistance better know how to navigate and or interact with the webpage’s content. Instead of reading undescriptive “div” information---the screenreader that reads a webpage that has ARIA attributes would read <nav role=”navigation”> and the listener would know that the unordered list that follows is the navigation bar. This is just one of many examples of how ARIA attributes improve the experience of users relying on assistive technology. Another attribute that provides accessibility is the use of ARIA labels. They provide context by linking a label with an input field so that those who can not read the site know what the site “looks” like via how it is labeled by ARIA labels and so can navigate and interact more easily with a webpage’s content.
3. The tools I used to check the accessibility of a website, including its readability via a color contrast check where “Lighthouse” extension in the developer tools of a browser and the website Coolors.co. The “Lighthouse” extension I found (once installed) by right-clicking on the webpage I wanted to analyze. I then clicked ‘inspect’ and the developer tools window opened. Then I clicked on the “Lighthouse” extension, to then finally I clicked on “analyze page load”. Once Lighthouse finished analyzing the page the webpage’s scores for Performance, Accessibility, Best Practices, and SEO were displayed. I then checked the color contrast with Coolors.co/contrast-checker <https://coolors.co/contrast-checker/112a46-bdc8d3>. I entered in the text and background color values into the contrast checker on the site link above. To receive a WCAG 2.1 AAA rating, the webpage’s contrast score had to be above 7.1 on normal-sized text; I played with the different color values for text and background until I found a good contrast checker value of around 8.5 then entered that into the “styles.css” file.