# Accessibility

*by Sophia*

WHAT'S COVERED

In this lesson, you will learn about some of the considerations for accessibility and CSS. You will learn about the impact of color as well as font sizing. You will also learn about the rem unit of measurement for text and how it helps a site to easily adapt to the user's needs. You will learn about a technique for adding additional accessibility to a navigation menu using a hidden auditory marker. Finally, there will be a brief discussion about testing to help ensure accessibility and to see things from the user's perspective.

Specifically, this lesson will cover the following:



# 1. CSS Accessibility Best Practices

CSS can be used to help with accessibility in small but effective ways. The following are some best practices for improving accessibility using CSS.

## 1a. Colors

One of the easiest areas to provide an accessible experience is through colors. Careful color choices and attention to contrast can help visually convey the message or information to a user with poor eyesight. Furthermore, keeping in mind color choices for color-blind individuals will help ensure the color is perceived.

However, do not rely solely on color for emphasis and meaning; include other visual variations such as font weight, size, and decorations as well. Furthermore, depending on the project, it may also be worth giving users the option to enable a high-contrast theme. This can simply be a case of preparing a high-contrast stylesheet for the site and using a button and JavaScript to swap the stylesheets.

## 1b. Font Size Using rem

Recall that font sizes can be represented by px and em. Pixels (px) represent a static size of the text (i.e., 12px will always produce text of size 12 pixels). Em, on the other hand, produces a dynamic text size based on its parent.

EXAMPLE If the parent text size is 16px

```
1 em = 16px
2 em = 32px
3 em = 48px
```

There is a technique used by developers to ensure that font sizes are all adjustable by using the “rem” unit of measurement. The **root-em (rem)** and its values are multiples of the HTML root element’s font-size setting. Not only does this help ensure that when a user zooms the content (either with “Ctrl” + mouse wheel or the browser menu setting) all text will resize correctly, but the method also helps simplify font sizing for developers.

EXAMPLE By setting the HTML element’s font size to 62.5%, you can simplify your rem values to correspond to the number of pixels you want the font to be:

```
html { font-size: 62.5% } /* 62.5% of 16px browser font size is 10px */
```

Now the text on our page is set to 10 pixels, but that is too small for most users to read.

EXAMPLE We can fix this by setting the body’s font size to 1.6rem to scale the size back up to 16px for all visible text on the page:

```
body {font-size: 1.6rem } /* 10 * 1.6 = 16px */
```

EXAMPLE Other elements on the page can have their own sizes set as well:

```
h1 {font-size: 2.4rem } /* 10 * 2.4 = 24px */
```

**Root-Em (rem)**
A unit of font size measurement that is calculated as a multiple of the HTML root element's font-size setting.

## 1c. Auditory Navigation Marker

Another aspect where CSS can be useful in improving accessibility is adding a hidden auditory note, called an **auditory navigation marker**, in the navigation menu indicating where the user is currently. Normally, this is accomplished visually by adding a marker next to the current page's link in the navigation menu or changing its color. However, this will not be seen or mentioned by screen reader software. To get around this, we can add a hidden div just before the current navigation link with text that reads something like "You are currently on." Remember to use display: none to hide the division. If you use property and value, visibility: hidden, the div will be invisible but will still take up space in the menu.

EXAMPLE CSS

```
nav .marker { display: none; }
```

EXAMPLE HTML

```
<nav>
     <ul>
          <li><a href="index.html">Home</a></li>
          <li>
                 <div class="marker">You are currently on </div>
                 <a href="about.html">About</a>
          </li>
          <li><a href="contact.html">Contact Us</a></li>
     </ul>
</nav>
```

TERM TO KNOW

**Auditory Navigation Marker**
A hidden note meant to be read by a screen reader that is added to the navigation menu.

## 1d. Names, Roles, Buttons, and Links

Components and elements that make up a user interface need to possess a few attributes that allow assistive technology to fully understand and properly present the information to the user. To aid assistive technology, interactive elements should use the most appropriate HTML element. For example, when a clickable element performs an action on the current page, a <button> element should be used. When a clickable element is used to navigate to another page, the <a> element should be used. The problem arises when different frameworks are used during the development process. Frameworks may sometimes force the use of the <a> tag for assembling navigation menus. In such cases, the workaround will be to include the HTML attribute "role" in the <a> tag and give it the value of "button."

Accessible names are also important for assistive technology. A name that is accessible is simply a value that can be read by assistive technology. In the following example, we have a div, serving as a button, surrounding the button's name. The name in this case is accessible:

EXAMPLE

```
<div role="button">Add Comment</div>
```

However, if we examine a dropdown menu using a <select> tag, the select tag does not provide a place for an accessible name. We could add the "name" HTML attribute; however, this is NOT accessible. Name attributes are meant to be used by computers and scripts. Instead, we need to include an ARIA (Accessible Rich Internet Application) label HTML attribute. ARIA labels are used to provide assistive technology more access and a better understanding of the various elements within a webpage. The aria-label attribute can be used to define a string that states what the interactive element does or represents. This aria-label provides the element with an accessible name.

EXAMPLE

```
<select aria-label="Number of seats">
  <option value="1">1</option>
  <option value="2">2</option>
  <option value="3">3</option>
  <option value="4">4</option>
</select>
```

# 2. Accessibility Testing

Testing is an invaluable part of any development project. With regard to ensuring accessibility, the importance of this step cannot be overstated. By taking the time to examine a website from the perspective of someone with a disability, you can gain valuable insight as to how to make the user's experience better.

Select one of the following activities for practice with this concept.

TRY IT

**Directions for Activity Option 1:** Now that you know more about accessibility testing, visit a website and practice navigating using your keyboard.

1. Access the www.bettycrocker.com website.
2. Practice navigating the website with only a keyboard to simulate a person with limited movement.

REFLECT

Were there any challenges navigating the website? Did it take you longer to access specific pages or links than when using a mouse? Were you able to tab through the page properly? Were there visual indicators to let you know where you were when tabbing through the page?

TRY IT

**Directions for Activity Option 2:** Practice using a screen reader application to navigate your project.

1. Download and install one or more screen reader applications. We suggest ChromeVox as an option, which adds text-to-speech to Google Chrome.
2. Open the website you are developing for this class.
3. Put on a blindfold and attempt to navigate your website.

REFLECT

Were you able to make sense of where you were? Did you get lost? Did headings and seriations make sense? Were you able to tab through the page properly?

## 2a. Automated Accessibility Testing Tools

Tools such as Lighthouse, discussed in an earlier challenge, can be utilized to examine your page and automatically identify different types of accessibility deficiencies or problems. Lighthouse can be installed as an extension for the Google Chrome web browser. Navigate to your webpage, open the Developer Tools pane, and select the Lighthouse tab on the navigation bar. From there, make sure that the “Accessibility” category is checked, and then click “Analyze page load.” Lighthouse will reload the page, analyze it, and then produce a report of possible images and point out opportunities to improve the page’s accessibility. While automated checking tools can provide time-saving benefits, it is important to understand that they are limited in their ability to examine every aspect of accessibility and your webpage.

## 2b. Manual Accessibility Testing

Manual accessibility testing requires the web developer to view their site as a person with a disability would.

It is important to manually check for accessibility elements, such as the following:

- ARIA labels are used correctly.
  ARIA labels should be used with caution as much of the current HTML elements and semantic elements already contain much of the needed accessibility factors. A common saying with ARIA is “no ARIA is better than bad ARIA.” In other words, poorly implemented ARIA labels and roles can cause more problems than they solve.
- Associated Labels.
  In web forms, labels possess a `for` attribute that associates a label with its intended control, such as a text box.
- User Focus Trap.
  When a user cannot tab beyond a specific container, their focus is trapped. This commonly happens (on

purpose) when modal screens are displayed (a modal screen is a small popup that appears in front of the webpage and the webpage is usually faded and blurred.)

- Interactive elements are keyboard accessible.
  Users can access all interactive elements using their keyboard.
- Logical tab order.
  When users use the tab key to move throughout the page, they are able to access all elements in a logical and meaningful sequence.
- Focus users on new page content.
  When JavaScript is utilized to add new content to a webpage, the user's focus is taken to the new content.
- Offscreen content and elements are hidden from assistive technologies.
  When menus and other content are hidden off-screen, outside of the viewport boundaries, assistive technologies should not be able to see the content.

---

# 3. Audio and Video Content Considerations

Although not a CSS trick, if your site has videos or audio content with important information presented in an audio format, there should be a written version available for persons with auditory impairment. This should be, at a minimum, a written transcript. **Open captions** are written text of the spoken dialogue embedded within the video, and they cannot be turned off. **Closed captions**, the gold standard, are the written captions on screen, but they can be turned off. Captions have been benefit provide not only a benefit for the hearing impaired but has also been shown to assist neurodivergent users as well.


**Directions:** Now that you know more about accessibility and what to look and test for, it is time to practice!

1. Visit a webpage that you are familiar with and access frequently. Pretend that you only have one hand, your less dominant hand, and attempt to access the site's services and features as you normally would.
2. Now, visit a website that you are unfamiliar with.
3. Rate both sites on a scale of 1 to 10, 1 being inaccessible and 10 being very accessible.
4. Now, on both sites, run the Lighthouse tool to assess the sites for accessibility. Were their scores surprising?


Was it easier to navigate the site you are already familiar with? Did Lighthouse confirm your experience?


View the following video for a TED talk from website accessibility advocate Clive Loseby, who sheds light on why many parts of the web are closed off to those with disabilities and lays out some steps to make being online better for everyone.

**Open Captions**
Written text of the spoken dialogue that is embedded within the video and cannot be turned off.

**Closed Captions**
Dialog and sound descriptions written as text on screen that can be turned on or off depending on user preference.

## SUMMARY

In this lesson, you learned about some of the best practices in **CSS accessibility**. This included proper **color** selection and **font sizing**. Additionally, you learned about considerations for **auditory navigation**, as one of the nuances of navigation menus and screen reader software, and how to overcome this issue using a hidden auditory marker. Lastly, you learned about the importance of **testing** to help ensure accessibility and the value of putting yourself in the shoes of someone who has a disability.

As web designers and developers, it is important to remember that you are not the user. Remember that disabilities can come in many different forms and can vary with situations, temporary and permanent. Creating a website that is available and accessible for all users can help your clients meet their business needs and goals.

Source: This Tutorial has been adapted from "The Missing Link: An Introduction to Web Development and Programming " by Michael Mendez. Access for free at https://open.umn.edu/opentextbooks/textbooks/the-missing-link-an-introduction-to-web-development-and-programming. License: Creative Commons attribution: CC BY-NC-SA.

## TERMS TO KNOW

**Auditory Navigation Marker**
A hidden note meant to be read by a screen reader that is added to the navigation menu.

**Closed Captions**
Dialog and sound descriptions written as text on screen that can be turned on or off depending on user preference.

**Open Captions**
Written text of the spoken dialogue that is embedded within the video and cannot be turned off.

**Root-Em (rem)**
A unit of font size measurement that is calculated as a multiple of the HTML root element's font-size setting.