100 XP

Designing for accessibility

3 minutes

Accessibility is a relatively large topic. We can't cover it completely in a single Learn module. However, there are some core tenets that you'll want to implement in every page you create. Designing an accessible page from the start is always easier than going back to an existing page to make it accessible.

Use HTML the way it was designed

HTML provides many elements that you can use to create a page, including buttons, links, and form controls. Each of those elements has a set of built-in functionality, like being clickable, being linkable, or accepting focus.

> **Note**
>
> *Focus* is a web development term that means a control can accept input from a keyboard. A button can accept focus, allowing someone to activate or "click" it by selecting the Spacebar.

With CSS and JavaScript, it's possible to make any element look like any type of control. For example, you can use `<span>` to create a `<button>` element, and `<b>` can become `<a>`. Although this capability provides some shortcuts for styling or laying out your page, it removes the built-in functionality. Tools like a screen reader won't be able to understand that `<span>` is being used as `<a>`. Someone browsing with a keyboard won't be able to set focus on a `<div>` element that has been programmed to simulate a `<button>` element.

Another HTML element that's often skipped is headers (`<h1>` through `<h6>`). From a visual standpoint, header tags start from largest to smallest text size. This convention leads many developers to forgo header elements and instead stylize `<div>` or other generic elements.

Unfortunately, stylized generic elements convey only visual information rather than structural. Users of screen readers rely heavily on headings to find information and browse through a page. Writing descriptive heading content and using semantic heading tags are important for creating an easily navigable site for users of screen readers.

As a best practice, you should always use the appropriate HTML when creating controls on a page. If you want a hyperlink, use `<a>`, or use `<button>` for a button.

Use good visual cues

Developers often think about screen readers as the only accessibility tool. However, users might use numerous other tools, or they might not use tools at all. Users who are using the browser will rely on certain visual cues to understand how to interact with your page.

One of the great features of CSS is that it provides complete control over how to display a page, including removing certain display elements. For example, you can remove the outline from a text box or remove the underline from a hyperlink. Unfortunately, removing those types of cues can make it more challenging for someone who depends on them to recognize the type of control.

Consider the keyboard

Some users can't use a mouse or trackpad/touchpad. Instead, these users rely on keyboard interactions to tab from one element to the next. It's important for your pages to present your content in logical order so a keyboard user can access each interactive element as they move down.

When a user moves through a page by tabbing, focus moves from one control to the next based on the order in which the controls are listed in the HTML source. The controls for your page should be listed in the HTML source in the order in which you expect the page to be browsed, while relying on CSS to lay out the page visually to users.

For example, imagine creating a form with two columns. You'll want to consider what the natural flow is for someone filling out the form, and then list the controls in that order. Then you can use CSS to create the columns and display the controls in their appropriate locations.

Keyboard navigation relies heavily on semantic HTML. Certain controls (like buttons) accept focus, whereas `div` elements don't. If you're re-creating controls that already exist in HTML, you're making it more difficult for someone to use your page with a keyboard.

> **Important**

Keyboard navigation needs to be tested manually, and you should do it on every page that you create. WebAIM has more information about keyboard navigation strategies.

Next unit: Knowledge check

Continue >