Setting Up an Accessible Web Page: Site Navigation (A UX Accessibility Blog Post)

This UX Accessibility Blog Series, “Setting Up An Accessible Web Page,” will walk you through the basic elements of an accessible web page in hopes that you can take what you’ve learned an apply it to more complex pages. This blog series is loosely based on Heydon Pickering’s book [“Inclusive Design Patterns: Coding Accessibility Into Web Design.”](https://www.smashingmagazine.com/inclusive-design-patterns/)

**Why do we use an unordered list?**

Using an unordered list is something that we’ve been doing since about 1997-2000 when HTML4 and XHTML were becoming standard. At this point, it’s just convention. We can style anything however we want these days, but we keep using unordered lists as navigation. In the event of a CSS failure, the navigation bar will revert back to a regular unordered list. This bulleted list of blue, underlined links is a visual indicator that this list serves some kind of navigational purpose. This way, we’re including older users, older devices, assistive technology users, and those experiencing a CSS failure all in one!

We wrap the list in a <nav role=”navigation”></nav> tag to trigger additional semantics and behaviors that allow AT users to better understand the purpose of this list of links. For example, screen reader users going through the navbar might hear “navigation landmark. List. One of five. Link: home.” It’s easy for your user to understand that they are in the navigation bar, how many links it contains, and if they want to go home, they can do so immediately after hearing the “link: home.” The nav element also automatically offers some screen reader users the ability to hit the D key and hear the list of navigation links while on any other part of the page.

**More on Navigation Conventions**

Although we like to make things look cool and new, sometimes it’s just best to stick with what we’ve always been doing, especially when it comes to navigation conventions. The human brain uses *schemata* to understand new things. This schemata acts as a cache for understanding, if you will, constituting the prior experience which the current experience is compared to. As humans, we like what’s familiar to us, so we’ll compare things to other things we’ve seen just to make sense of it.

When it comes to navigation bars, we don’t want to make our users think, which is a common idea in usability. This is why we typically see navigation bars at the top of the page with font larger than the rest of the page and different links right next to each other. It’s familiar to us. We can look at that and immediately know that this is the part of the page we need to use to be able to see other pages on this website.

**Know where you are to know where you want to go**

Another helpful aspect of navigation bars is when they show which page you’re on by making that navigation link or button “active.” Just like when maps in malls have the “you are here” dot, we need some visual “you are here” indicator in the navigation bar for the user. Typically this is built into the framework, so all you need to do is add some Javascript that adds the active class to whatever link represents the current page.

The active class usually does a good job of showing the user where they are, but the problem is that the active indicator is usually only color. This violates WCAG 2.0 Success Criterion 1.4.1 (Level A): “Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.” To solve this problem, sometimes an arrow can be used to indicate the current page. For example, in the default template in Ensemble, the left-side navigation has an arrow that shows the user which page they are on.

[picture]

Although color alone isn’t a great indicator of showing the user which page they are on, it is important not to add other things to show the active page unless it is in Ensemble or it has been approved by UX.

**Accommodate Multiple Navigation Areas with aria-label**

When using a screen reader with more than one navigation bar (think secondary nav in Ensemble or a side navigation, as shown above), it is important to differentiate the two. A screen reader would label them both as “navigation”, as that is the default label for anything with the role=”navigation” or nav element. To solve this problem, WAI-ARIA has a global aria-labelledby relationship attribute to provide a label with heading text. For example:

<nav class="navbar navbar-fnc" role="navigation" aria-label="Main Navigation">

<div class="container-fluid">

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbarCollapseDiv">

<span class="glyphicon glyphicon-menu-hamburger"></span><span class="sr-only">Menu</span>

</button>

<a class="navbar-brand" href="#">Product/Client Logo</a>

</div>

<div class="collapse navbar-collapse" id="navbarCollapseDiv">

<ul class="nav navbar-nav navbar-right">

<li role="separator" class="divider-vertical"></li>

…

</ul>

</div>

</div>

</nav>

<nav class="navbar navbar-fnc-secondary" role="navigation" aria-label="Secondary Navigation">

<div class="container">

<ul class="nav nav-pills" role="tablist">

<li class="active"><a href="#" role="tab">Menu Item 1</a></li>

<li><a href="#" role="tab">Menu Item 2</a></li>

<li><a href="#" role="tab">Menu Item 3</a></li>

</ul>

</div>

</nav>