Semantics and navigating content

- Using headings effectively
- Other navigation options

You've learned about affordances, semantics, and how assistive technologies use the accessibility tree to create an alternative user experience for their users. You can see that writing expressive, semantic HTML gives you a lot of accessibility with very little effort, as many standard elements have both the semantics and supporting behavior built in.

In this lesson we'll cover some less obvious semantics that are very important to screen reader users, especially as regards navigation. In a simple page with lots of controls but not much content, it's easy to scan the page to find what you need. But on a content-heavy page, such as a Wikipedia entry or a news aggregator, it's not practical to read through everything from the top down; you need a way to efficiently navigate through the content.

Developers often have the misconception that screen readers are tedious and slow to use, or that everything on the screen has to be focusable for the screen reader to find it. That's often not the case.

Screen reader users often rely on a list of headings to locate information. Most screen readers have easy ways to isolate and scan a list of page headings, an important feature called the *rotor*. Let's see how we can use HTML headings effectively to support this feature.

Using headings effectively

First, let's reiterate an earlier point: *DOM order matters*, not only for focus order but for screen reader order. As you experiment with screen readers like VoiceOver, NVDA, JAWS, and ChromeVox, you'll find the heading list follows the DOM order rather than the visual order.

This is true for screen readers in general. Because screen readers interact with the accessibility tree, and the accessibility tree is based on the DOM tree, the order a screen reader perceives is thus directly based on the DOM order. This means that an appropriate heading structure is more

important than ever.

In most well-structured pages, the heading levels are nested to indicate parent-child relationships among content blocks. The WebAIM checklist repeatedly refers to this technique.

- 1.3.1{: .external } mentions "Semantic markup is used to designate headings"
- 2.4.1{: .external } mentions heading structure as a technique for bypassing blocks of content
- 2.4.6{: .external } discusses some details for writing useful headings
- 2.4.10{: .external } states "individual sections of content are designated using headings, where appropriate"

Not all headings have to be visible on-screen. Wikipedia, for instance, uses a technique that deliberately places some headings off-screen to specifically make them accessible *only* to screen readers and other assistive technology.

```
<style>
    .sr-only {
    position:absolute;
    left:-10000px;
    top:auto;
    width:1px;
    height:1px;
    overflow:hidden;
    }
</style>

<h2 class="sr-only">This heading is offscreen.</h2>
```

The WebAIM site discusses this technique at length in this article on offscreen content.

For complex applications, this can be a good way to accommodate headings when the visual design doesn't require or have room for a visible heading.

Caution: It's important not to go overboard with this technique. Remember that assistive technology users may also be able to see the screen for themselves, so going too far down the path of creating "screen reader only" content may actually degrade the user experience for some users. It can also create a maintenance headache for you later.

Other navigation options

Although pages with good headings help screen reader users navigate, there are other elements they can use to move around a page, including *links*, *form controls*, and *landmarks*.

Readers can use the screen reader's rotor feature (an easy way to isolate and scan a list of page headings) to access a *list of links* on the page. Sometimes, as on a wiki, there are many links, so the reader might search for a term within the links. This limits the hits to links that actually contain the term, rather than every occurrence of the term on the page.

This feature is useful only if the screen reader can find the links and the link text is meaningful. For example, here are some common patterns that make links hard to find.

- Anchor tags without `href` attributes. Often used in single-page applications, these link targets cause problems for screen readers. You can read more in this article on single-page apps.
- Buttons that are implemented with links. These cause the screen reader to interpret the content as a link, and the button functionality is lost. For these cases, replace the anchor tag with a real button and style it appropriately.
- Images used as link content. Sometimes necessary, linked images can be unusable to screen readers. To guarantee that the link is properly exposed to assistive technology, make sure the image has `alt` attribute text.

Poor link text is another problem. Clickable text such as "learn more" or "click here" provides no semantic information about where the link goes. Instead, use descriptive text like "learn more about responsive design" or "see this canvas tutorial" to help screen readers provide meaningful context about links.

The rotor can also retrieve a *form control list*. Using this list, readers can search for specific items and go directly to them.

A common error that screen readers make is pronunciation. For example, a screen reader might pronounce "Udacity" as "oo-dacity", or read a phone number as a large integer, or read capitalized text as though it were an acronym. Interestingly, screen reader users are quite used to this quirk and take it into consideration.

Some developers try to ameliorate this situation by providing screen-reader-only text that is spelled phonetically. Here's a simple rule for phonetic spelling: **don't do it**; it only makes the problem worse! If, for example, a user is using a braille display, the word will be spelled incorrectly, leading to more confusion. Screen readers allow words to be spelled aloud, so leave it to the reader to control their experience and decide when this is necessary.

Readers can use the rotor to see a *landmarks list*. This list helps readers find the main content and a set of navigational landmarks provided by HTML landmark elements.

HTML5 introduced some new elements that help define the semantic structure of the page, including `header`, `footer`, `nav`, `article`, `section`, `main`, and `aside`. These elements specifically provide structural clues in the page without forcing any built-in styling (which you should do with

CSS anyway).

Semantic structural elements replace multiple, repetitive `div` blocks, and provide a clearer, more descriptive way to intuitively express page structure for both authors and readers.

□□□ #3
□ □□□ □□□ 2026-10-31 22:04:36 UTC
□ cocobook-sync □□□ 2026-11-04 17:37:37 UTC