What are some ways to make web applications keyboard accessible?

Many users rely on keyboards instead of mice due to physical disabilities, repetitive strain injuries, or personal preference. This includes users of screen readers and those who may not have a working mouse. Keyboard accessibility ensures these users can navigate web applications effectively without barriers.

Let's look at some practical techniques you can employ to make web applications keyboard accessible.

Many users use the Tab key to navigate through interactive elements on a webpage. The tabindex attribute is used to make elements focusable and define the relative order in which they should be navigated using the keyboard. It takes a numerical value, which can be positive, zero, or negative.

Here is the basic syntax:

<element tabindex="number">Element Text</element>

It is important to never use the tabindex attribute with a value greater than 0. Instead, you should either use a value of 0 or -1.

You can set the tabindex to -1 to allow it to receive programmatic focus. This is used in a variety of situations for "focus management", such as when you need to set the focus on a heading, error message, or dialog, or you are building custom components such as tabbing interfaces or tree components.

<p tabindex="-1">Sorry, there was an error with your submission.</p>

Setting tabindex to 0 allows you to add an element that does not receive keyboard focus by default into the page's natural tab order. This allows keyboard users to Tab to the element. This is primarily used when building custom elements that need keyboard functionality.

<div role="combobox" tabindex="0">

accesskey is another attribute you can use to make your web project keyboard accessible. It allows you to define a key that focuses on or activates a particular element:

<button accesskey="s">Save</button>

<button accesskey="c">Cancel</button>

<a href="index.html" accesskey="h">Home</a>

In the above code:

* accesskey="s" assigns the key S to the Save button. On most browsers, pressing ALT + S (on Windows) and CTRL + Option + S (on Mac) will activate this button.
* accesskey="c" sets the key C to the Cancel button, allowing users to activate it using ALT + C (Windows) and CTRL + Option + C (Mac).
* accesskey="h" assigns the key H to the Home link, allowing users to navigate to the homepage using ALT + H (Windows) and CTRL + Option + H (Mac).

Please note that the exact key combination to activate the accesskey might vary depending on the browser and operating system. It's typically ALT + Specified Key on Windows and CTRL + Option + Specified Key on Mac.

Another way to make the keyboard accessible in your apps is to make sure you provide clear focus indicators. If you feel the default browser focus indicator is not enough, you can override it by targeting the focus state of the element.

Here is an example of styling the focus state for an HTML element:

element:focus {

outline: 2px solid #005fcc;

}

The outline property is used to define the outline around the element. This example sets the outline to a solid blue line with 2 pixels set for the thickness. The focus indicator should be styled in a way that makes it obvious which element currently has focus. In order to be accessible, the indicator must have a minimum color contrast of at least 3:1 with the background color it covers.

You should also avoid keyboard traps, which occur when a user cannot move focus away from a certain element in components like modals and popups.