Removing Headaches from Focus Management

By Rob Dodson
Rob is a contributor to Web**Fundamentals**

The 'sequential focus navigation starting point' feature defines where we start to search for focusable elements for sequential focus navigation ([Tab] or [Shift-Tab]) when there is no focused area. It's especially helpful for accessibility features like "skip links" and managing focus in the document.

Removing headaches from focus management

HTML provides us with a lot of built in support for dealing with keyboard interactions, which means it's pretty easy to write pages which can be used via the keyboard - whether a motor impairment prevents us from using a mouse, or we're just so efficient removing our hands from the keyboard wastes precious milliseconds.

Keyboard handling revolves around focus, which determines where keyboard events will go in the page. There are a few situations in which, up till now, we've needed to do some extra work to make things work well for keyboard users. The `focus()` method allows us to manage focus by selectively choosing an element to focus in response to a user action. However, this best practice suffers from a lot of gotchas and requires some tricky JavaScript hackery to provide a baseline experience.

While this technique isn't going to completely go away any time soon, in Chrome 50 it will be necessary in fewer cases thanks to the Sequential Focus Navigation Start Point. With this change, well-authored pages will automatically become more accessible without any need for extra manual focus management. Let's look at an example.

Linking within a page

Text heavy sites often interlink within the same page to help users quickly jump to important sections.

```
<!-- Table of Contents -->
<a href="#recipes">Recipes</a>
<a href="#ingredients">Ingredients</a>

<!-- Recipes Section -->
<h2 id="recipes">Recipes</h1>
<h3>Vegemite Cheesecake</h3>
<p>
  Vegemite cheesecake is delicious. We promise.
  <a href="cheesecake.html">Read More</a>
</p>
```

If I were a keyboard user (and a glutton for Australian foods) my next series of actions would go something like this:

- Press [Tab] twice to focus the Recipes link
- Press [Enter] to jump to the Recipes section
- Press [Tab] again to focus the Read More link

Let's see that in action using Chrome 49.

Oh. Well that didn't go quite according to plan did it?

Instead of focusing the Read More link, pressing [Tab] for the final time moved focus to the next item in the table of contents. This is because the developer did not set **`tabindex="-1"`** on the header to make it focusable. So clicking on the **`#recipes`** named anchor did not move focus. It's a subtle mistake, and not a big deal if you're a mouse user. But it's potentially a very big deal if you're a keyboard or switch device user. Consider the amount of interlinking on a typical Wikipedia page? It would be frustrating to have to constantly tab back and forth through all of those anchors!

Let's look at the same experience now using Chrome 50.

Wow that’s exactly what we wanted, and best of all, we didn’t have to change our code. The browser just figured out where focus should go based on where we were in the document.

How does it work?

Prior to the implementation of the focus starting point, focus would always move from either the previous focused element, or the top of the page. This means that choosing what gets focused next can involve moving focus to something which shouldn't really be focusable, like a container element or a heading. This causes all sorts of weirdness, including showing a focus ring if you happen to idly click such an element.

The focus start point, as the name suggests, provides a mechanism for suggesting where to start looking for the next focusable element when we press [Tab] or [Shift-Tab].

It can be set in a number of ways: If something has focus, it's also the focus navigation start point, just like before. Also, just like before, if nothing else has set the focus navigation start point, then it will be the current `document` or, if available and supported, the currently active `dialog`. If we navigate to a page fragment like in the example above, that will now set the focus start point. Also, if we click any element on the page, regardless of whether it is focusable, that will now set the focus navigation start point. Finally, if the element which was the focus start point is removed from the DOM, its parent becomes the focus start point. No more focus whack-a-mole!

Other use cases

Aside from the above example, there are many other scenarios where this feature can come in handy.

Hiding elements

There may be times when a user will be focused on an item that needs to be set to `visibility: hidden` or `display: none`. An example of this would be clickable items within a carousel. In prior versions of Chrome, hiding the currently focused item in this manner would reset focus back to the default starting point, turning the aforementioned carousel into a nasty trap for motor impaired users. With sequential focus starting point, this is no longer

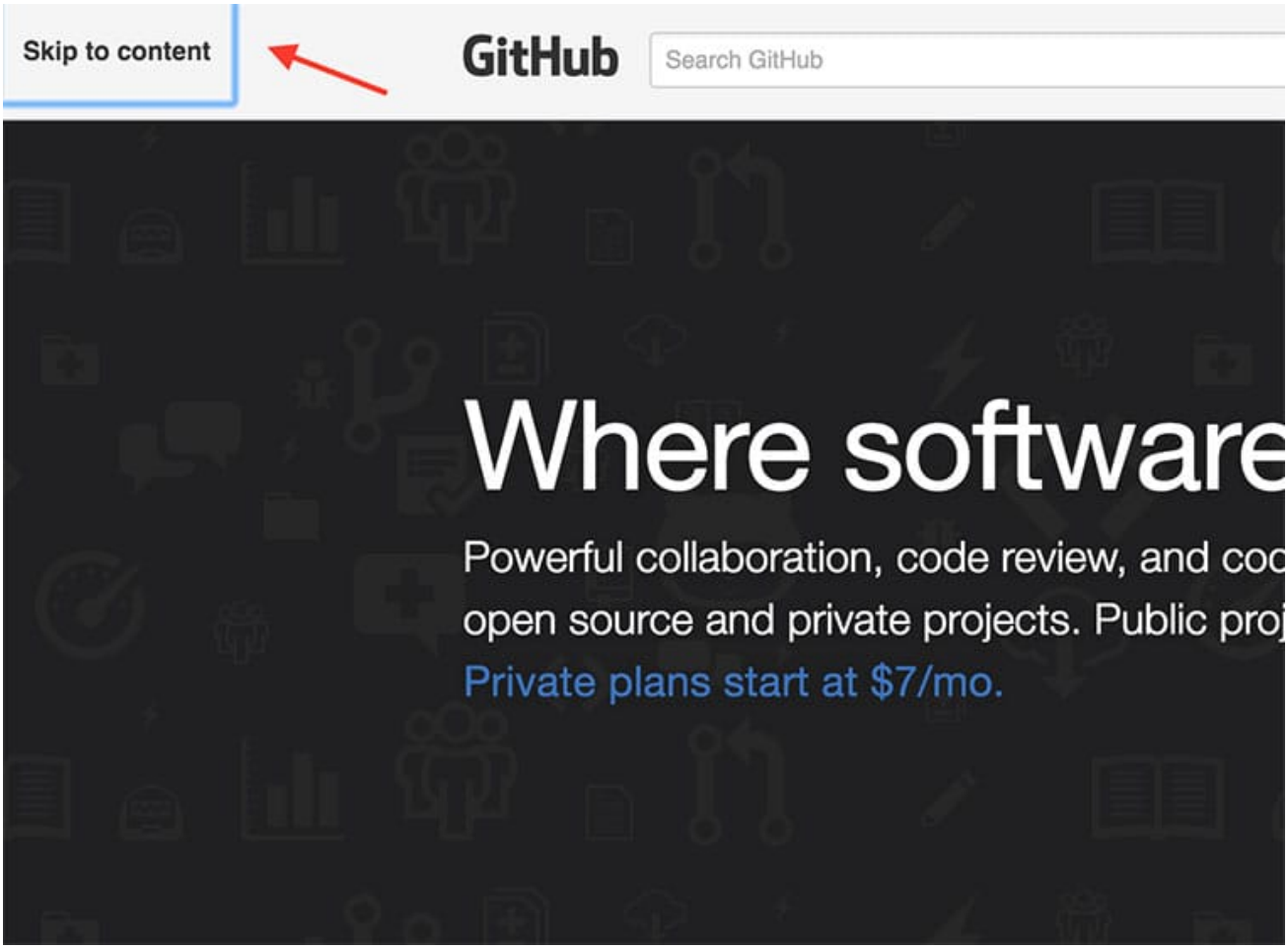
the case. If an element is hidden through either of the above methods, pressing the [Tab] key will simply move to the next focusable item.

Skip links

Skip links are invisible anchors which can only be reached via the keyboard. They allow users to “skip" navigation elements in order to jump straight into the content of a page and they can be extremely beneficial for keyboard and switch device users. As explained on the WebAIM site:

> Without some sort of system for bypassing the long list of links, some users are at a huge disadvantage. Consider users with no arm movement, who use computers by tapping their heads on a switch or that use a stick in their mouth to press keyboard keys. Requiring users to perform any action perhaps 100s of times before reaching the main content is simply unacceptable.

Many popular websites implement skip links, though you may have never noticed them.



Because skip links are named anchors, they work in the same fashion as our original table of contents example.

Caveats and support

Sequential focus navigation starting point is currently only supported in Chrome 50, Firefox, and Opera. Until it is supported in all browsers you'll still need to add `tabindex="-1"` (and remove the focus outline) to your named anchor targets.

Demo

Sequential focus navigation starting point is a great addition to the browser's set of accessibility primitives. It's easy to grok and actually lets us remove code from our application while improving the experience for our users. Double win! Take a look at the demo to explore this feature in more depth.

Last updated July 2, 2018.