Stick your landings! position: sticky lands in WebKit

By Eric Bidelman

Engineer @ Google working on web tooling: Headless Chrome, Puppeteer, Lighthouse

`position: sticky` is a new way to position elements and is conceptually similar to **`position: fixed`**. The difference is that an element with **`position: sticky`** behaves like **`position: relative`** within its parent, until a given offset threshold is met in the viewport.

Use cases

Paraphrasing from Edward O'Connor's original proposal of this feature:

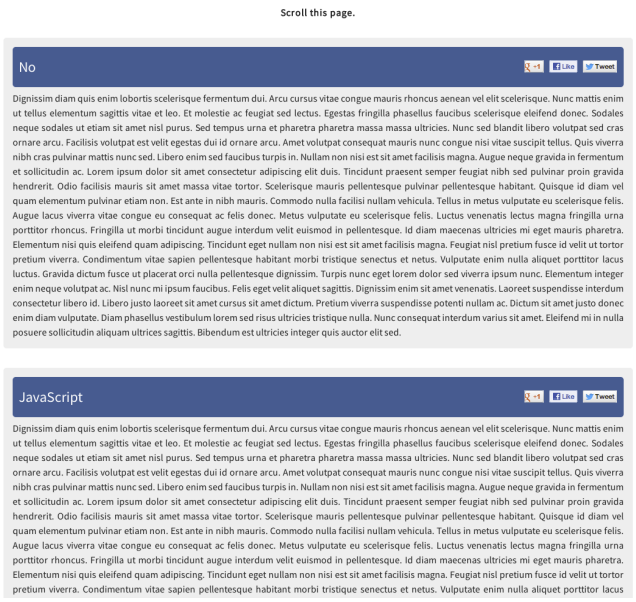
> Many websites have elements that alternate between being in-flow and having **`position: fixed`**, depending on the user's scroll position. This is often done for elements in a sidebar that the page author wants to be always visible as the user scrolls, but which slot into a space on the page when scrolled to the top. Good examples are news.google.com (the "Top Stories" sidebar) and yelp.com (search results map).

Introducing sticky positioning

LAUNCH DEMO

By simply adding **`position: sticky`** (vendor prefixed), we can tell an element to be **`position: relative`** until the user scrolls the item (or its parent) to be 15px from the top:

```
.sticky {
  position: -webkit-sticky;
  position: -moz-sticky;
  position: -ms-sticky;
  position: -o-sticky;
  top: 15px;
}
```



At `top: 15px`, the element becomes fixed.

To illustrate this feature in a practical setting, I've put together a DEMO which sticks blog titles as you scroll.

Old approach: scroll events

Until now, to achieve the sticky effect, sites setup `scroll` event listeners in JS. We actually use this technique as well on html5rocks tutorials. On screens smaller than 1200px, our table of contents sidebar changes to `position: fixed` after a certain amount of scrolling.

Here's the (now old way) to have a header that sticks to the top of the viewport when the user scrolls down, and falls back into place when the user scrolls up:

```
<div class="header"></div>

<script>
var header = document.querySelector('.header');
var origOffsetY = header.offsetTop;

function onScroll(e) {
  window.scrollY >= origOffsetY ? header.classList.add('sticky') :
                                  header.classList.remove('sticky');
}

document.addEventListener('scroll', onScroll);
</script>
```

Try it: http://jsbin.com/omanut/2/

This is easy enough, but this model quickly breaks down if you want to do this for a bunch of DOM nodes, say, every `<h1>` title of a blog as the user scrolls.

Why JS is not ideal

In general, scroll handlers are never a good idea. Folks tend to do too much work and wonder why their UI is janky.

Something else to consider is that more and more browsers are implementing hardware accelerated scrolling to improve performance. The problem with this is that on JS scroll handlers are in play, browsers may fall back into a slower (software) mode. Now we're no longer running on the GPU. Instead, we're back on the CPU. The result? User's perceive more jank when scrolling your page.

Thus, it makes a lot of sense to have such feature be declarative in CSS.

Support

Unfortunately, there isn't a spec for this one. It was proposed on www-style back in June and just landed in WebKit. That means there's no good documentation to point to. One thing to note however, according to this bug, if both `left` and `right` are specified, `left` wins. Likewise, if `top` and `bottom` are used at the same time, `top` wins.

Support right now is Chrome 23.0.1247.0+ (current Canary) and WebKit nightly.

Last updated July 2, 2018.