Stacking Changes Coming to position:fixed elements

By Tom Wiltzius

Tom is a contributor to Web**Fundamentals**

In Chrome 22 the layout behavior of `position:fixed` elements is slightly different than previous versions. All `position:fixed` elements now form new stacking contexts. This will change the stacking order of some pages, which has the potential to break page layouts. The new behavior matches the behavior of WebKit browsers on mobile devices (iOS Safari and Chrome for Android).

Stacking Whats?

Everyone knows and loves the `z-index` for determining depth ordering of elements on a page. Not all z-indexes are created equal, however: an element's `z-index` only determines its ordering relative to other elements in the same stacking context. Most elements on a page are in a single, root stacking context, but absolutely or relatively positioned elements with non-auto `z-index` values form their own *stacking contexts* (that is, all of their children will be z-ordered within the parent and not be interleaved with content from outside the parent). As of Chrome 22, `position:fixed` elements will also create their own stacking contexts.

For a general overview of stacking contexts this MDN article is a great read.

Compare `position:fixed` to the new position:sticky attribute: for reference, `position:sticky` always creates a new stacking context.

Motivation

Mobile browsers (Mobile Safari, Android browser, Qt-based browsers) put position:fixed elements in their own stacking contexts and have for some time (since iOS5, Android Gingerbread, etc) because it allows for certain scrolling optimizations, making web pages much more responsive to touch. The change is being brought to desktop for three reasons:

1. Having different rendering behavior on "mobile" and "desktop" browsers is a stumbling block for web authors; CSS should work the same everywhere when possible.

2. With tablets it isn't clear which of the "mobile" or "desktop" stacking context creation algorithms is more appropriate.
3. Bringing the scrolling performance optimizations from mobile to desktop is good for both users and authors.

Specifics of the Change

Here's an example showing the different layout behaviors:
http://codepen.io/paulirish/pen/CgAof

With the change, both versions will render like the right-hand version.

In this example, the green box has a `z-index: 1`, the pink box has a `z-index: 3`, and the orange box has a `z-index: 2`. The blue box is an ancestor of the orange box, and has `position:fixed`.

If the blue box gets its own stacking context, the orange box's `z-index` is computed relative to the blue box's stacking context. Since the blue box has a `z-index` of `auto`, giving it a stacking level of zero in the root stacking context, this means the orange box ends up behind the green and pink boxes, which have z-indexes of 1 and 3 (respectively) in the root context.

If the blue box does not get its own stacking context, the orange box's `z-index` is computed relative to the root stacking context (along with the green and pink boxes). Hence, the orange box ends up interleaved with the pink and green boxes.

For more detail about criteria for stacking context creation (and how stacking contexts behave in general), again refer to this MDN article. In the example the right-hand side version always gave the blue box its own stacking context because its opacity is less than 1. The behavior change being made effectively adds another criterion for creating a separate stacking context, namely an element being position:fixed.

Testing & The Future

To test if your page is going to change, go to Chrome's `about:flags` and turn on/off "fixed position elements create stacking contexts". If your layout behaves the same in both cases, you're set. If not, make sure it looks acceptable to you with that flag enabled, as that will be the default in Chrome 22.

This change removes one capability - the ability to interleave content within a position:fixed subtree with non-scrolling content from outside. It's unlikely that any web developers are

doing this on purpose, and the same effect can be created by giving multiple position:fixed elements the different portions of DOM. As an example, consider these two examples:

http://codepen.io/wiltzius/pen/gcjCk

This page attempts to take two child divs (overlayA and overlayB) of a position:fixed element and place one above a separate content div and one below that same separate content div. It's now impossible to do this because the position:fixed element is its own stacking context, and it (along with all of its children) will be either entirely above or entirely below the content div. Note that this example works in Chrome 21 and earlier but no longer in Chrome 22.

To fix this, the two overlays can be broken up into their own position:fixed elements. Each is its own stacking context, one of which can go above the content div and one of which can go below the content div. See the fixed example, which works in Chrome 21 and 22:

http://codepen.io/wiltzius/pen/vhFzG

Credit for the genesis of this example goes to the inimitable hixie.

Chrome is the first desktop browser to cause position:fixed elements to create their own stacking contexts. The relevant standard is the CSS z-index specification (see e.g. http://www.w3.org/TR/CSS21/zindex.html). There's not yet consensus over what to do about the difference between mobile and desktop browsers here, but given the confusion of having two different behaviors on mobile and desktop, Chrome has chosen to move to this single behavior on both platforms for the time being.

Updated Oct 1, 2012: The original version of this article suggested that the CSS `z-index` specification had already been changed to reflect the new behavior of position: fixed elements. This is inaccurate; it has been discussed on the www-style list but as of yet no change has been taken into the spec.
