Top Control Initiated Resizes

bokan@chromium.org

tl;dr

Chrome resizes the initial containing block(ICB) in response to the top controls, meaning elements with % heights or vh units get resized as a result of scrolling. We should match Safari here and make it static (pos:fixed elements should still resize), matching the viewport size with top controls showing.

~~Additionally, we should consider removing the Resize event and changing the timing of the resize to happen on top controls hitting their extent, rather than touch-end.~~

Background

On page load, the top controls (aka "The URL bar") start off shown. If the page is scrollable, and the user scrolls down, the top controls will begin to hide, as if they're attached to the content. Scrolling the page up brings the top controls back in. This means the amount of height available to the page changes as the top controls are shown and hidden.

However, there's two notions of height here: the content height the user sees (this changes as the top controls hide/show), and the layout height the page sees. Resizing the page can be a costly operation since it typically means we have to perform a layout. Also, the page may have a long running resize handler. For these reasons, we don't tell the page about the resize continuously during the scroll. Only when the user lifts their finger, and the top controls are either fully shown or fully hidden, will we resize the initial containing block (ICB) from which the page gets its layout size and send a resize event. During the scroll, we simply adjust of the height of the clipping layer, showing more or less of the page content to the user.

An added complication is position: fixed elements fixed to the bottom of the screen. Until we give the page the new height and perform a layout, simply adjusting the clipping layer means that the bottom fixed element wouldn't stay fixed in place as expected. We compensate for this with a small hack that nudges bottom fixed elements up or down, depending on the top controls position. This makes them appear to stay stuck to the page bottom.

Problems

Constantly resizing the page is jarring to users and gives an impression of poor performance - particularly as the page is loading; adding a resize and layout just adds to the load-time jank. Since we don't resize the page continuously during the scroll, there's an effect where the user lifts their finger and the page starts moving around (while we do layout). This is surprising to the user ("the top controls may have been hidden a while ago, why is the page going nuts?") and in the worst case can cause them to lose their place.

For developers, dealing with top controls can be difficult and restricts how they build their pages. For example, while it sounds like nice design to be able to peg the font size to the viewport size, doing so is currently impractical since every time the user scrolls in a new direction, the text will change size(bug). Also, developers don't expect frequent resize messages, particularly on mobile devices. This leads to surprising behaviour and/or janky pages.

What do other browsers do? (Notes)

Tested mostly on http://bokan.ca/resize.html

Safari

Safari has both a bottom and top control bars. They don't fully hide but shrink on scrolls to smaller versions of themselves.

Resize events fire as soon as the top controls hit their full shrunk/expanded state but before the touch end.

Percentage based heights relative to body don't change at all! document.documentElement.clientHeight remains constant. Seems like the ICB doesn't change size and is based on height with top controls shown. Strange that they send a resize at all.

Pos:fixed elements do change height. They disappear (strange!) when the controls are fully hidden, reappear with new height after touchend. Bottom fixed are counter scrolled like Chrome until disappearance.

IE

IE has a static URL bar that doesn't change size

Firefox

Doesn't fire resize events at all. (For top control resizes)

Non fixed elements with percentage heights remain constant. Like Safari, the ICB also looks static except that they use the height with top controls hidden.

Pos:fixed elements are resized to match the new height as soon as the top controls hit fully shown/hidden. Bottom fixed elements are counter scrolled like like Chrome.

Summary

	IE	***Chrome***	***Safari***	***Firefox***	***Chrome Proposed***
Resize Event Timing	N/A	Touchend	Top Controls Hit Extent	No Resize Event	Touchend
ICB Size (documentElement.clientHeight)	N/A	Dynamic - based on visual viewport* - changed on touchend	Static - smallest possible viewport (i.e. controls showing)	Static - largest possible viewport (i.e. controls hidden**)	Static - smallest possible viewport
window.{inner|outer}Height	N/A	Dynamic - based on visual viewport* - changed during scroll	Dynamic - based on visual viewport* - changed during scroll	Static - largest possible viewport	Dynamic - based on visual viewport
Pos: fixed elements resize timing	N/A	Touch End	Touch End	Top Controls Hit Extent	Touch End
pos: fixed height based on	N/A	Visual Viewport*	Visual Viewport*	Visual Viewport*	Visual Viewport
vh units relative to	N/A	ICB	Static - Largest possible viewport	ICB	Static - smallest or largest possible viewport?

**Visual Viewport here means the size of the content area, not including the top controls if they're showing*
***If possible. If the page is non-scrollable, the controls don't hide and the ICB doesn't include the top controls height. Hence, "largest possible".*

Proposal

Here's some changes I think would make this area more rational and interoperable:

1. Having a static ICB size would make developer's lives easier and improve performance on a subset of pages. The question is which size to use (the one with top controls showing or hidden). Sizing the ICB to the page with controls showing will make the initial page look more correct. We'd also be compatible with Safari which is a plus. This point is the most important and should be fairly non-controversial given we're the only ones that do it.

   A side effect of this is that elements with percentage based heights relative to the root element may not reflect the viewport when the top controls are hidden. For example, if we have a position: absolute; height: 100% element, it will fill the viewport when the page first loads. However, when we hide the top controls we won't resize the ICB so the element will now be shorter than the viewport by the amount of the top controls. Position: fixed elements with height 100% will resize as expected though (which is a natural consequence of their containing block directly being the viewport: see http://www.w3.org/TR/CSS21/visudet.html#containing-block-details).

2. ~~Don't send a resize event. IMO, if we're not changing the ICB then we shouldn't fire a resize. This might be controversial and we'd diverge from Safari here. On the other hand, I only rarely see bad resize handlers so this probably isn't as big a deal.~~

3. ~~Resize event (if we do keep it) and pos:fixed resize should happen on Top Controls hitting their extent. Safari's resize event and pos:fixed resize timing are inconsistent here and make no sense. This is more of a UX concern than compatibility and developer pain.~~