Fixed toolbars

In browsers that support CSS `position: fixed` (most desktop browsers, iOS5+, Android 2.2+, and others), toolbars that use the "fixedtoolbar" plugin will be fixed to the top or bottom of the viewport, while the page content scrolls freely in between. In browsers that don't support fixed positioning, the toolbars will remain positioned in flow, at the top or bottom of the page.

Quick Links

Fixed basics

To enable this behavior on a header or footer, add the `data-position="fixed"` attribute to a jQuery Mobile header or footer element.

Fixed header markup example:

```
<div data-role="header" data-position="fixed">
        <h1>Fixed Header!</h1>
</div>
```

Fixed footer markup example:

```
<div data-role="footer" data-position="fixed">
        <h1>Fixed Footer!</h1>
</div>
```

Top

Fullscreen Toolbars

Fullscreen fixed toolbars sit on top of the content at all times when they are visible, and unlike regular fixed toolbars, fullscreen toolbars do not fall back to static positioning when toggled. Instead they disappear from the screen entirely. Fullscreen toolbars are ideal for more immersive interfaces, like a photo viewer that is meant to fill the entire screen with the photo itself and no distractions.

To enable this option on a fixed header or footer, add the `data-fullscreen` attribute to the element.

```
<div data-role="header" data-position="fixed" data-fullscreen="true">
        <h1>Fixed Header!</h1>
</div>
```

Top

Forms in toolbars

Fixed footer

While all form elements are now tested to work correctly within *static* toolbars as of jQuery Mobile 1.1, we recommend extensive testing when using form elements within *fixed* toolbars or within any `position: fixed` elements. This can potentially trigger a number of unpredictable issues in various mobile browsers, Android 2.2/2.3 in particular (detailed in Known issues in Android 2.2/2.3, below).

Forms in toolbar example

Top

Changes in jQuery Mobile 1.1

Prior to version 1.1, jQuery Mobile used dynamically re-positioned toolbars for the fixed header effect because very few mobile browsers supported the `position:fixed` CSS property, and simulating fixed support through the use of "fake" JavaScript overflow-scrolling behavior would have reduced our browser support reach, in addition to feeling unnatural on certain platforms. This behavior was not ideal, and jQuery Mobile 1.1 took a new approach to fixed toolbars that allows much broader support. The framework now offers true fixed toolbars on many popular platforms, while gracefully degrading non-supporting platforms to static positioning.

Polyfilling older platforms

The fixed toolbar plugin degrades gracefully in platforms that do not support CSS

`position:fixed` properly, such as iOS4.3. If you still need to support fixed toolbars on that platform (with the show/hide behavior) included in previous releases, Filament Group has developed a polyfill that you can use.

- Slashdot
- Intel

- Github code repository with CSS, and JavaScript required for the fixed toolbars polyfill
- Preview URL using the code in the repo above

Just include the CSS and JS files after your references to jQuery Mobile and Fixed toolbars will work similarly to jQuery Mobile 1.0 in iOS4.3, with the inclusion of the new API for the 1.1 fixedtoolbar plugin.

If you have any improvements to suggest, fork the gist on github and let us know!

Top

Known issue with form controls inside fixed toolbars, and programmatic scroll

An obscure issue exists in iOS5 and some Android platforms where form controls placed inside fixed-positioned containers can lose their hit area when the window is programatically scrolled (using `window.scrollTo` for example). This is not an

issue specific to jQuery Mobile, but because of it, we recommend not programatically scrolling a document when using form controls inside jQuery Mobile fixed toolbars. This ticket from the Device Bugs project tracker explains this problem in more detail.

Top

Known issues in Android 2.2/2.3

Android 2.2/2.3's implementation of `position: fixed;` can, in conjunction with seemingly unrelated styles and markup patterns, cause a number of strange issues, particularly in the case of `position: absolute` elements inside of `position: fixed` elements. While we've done our best to work around a number of these unique bugs within the scope of the library, custom styles may cause a number of issues.

- Form elements elsewhere on the page—select menus in particular—can fail to respond to user interaction when an *empty* absolute positioned element is placed within a fixed position element. In rare cases—and specific to Android 2.2—this can cause *entire pages* to fail to respond to user interaction. This can seemingly be solved by adding any character to the absolute positioned element, including a non-breaking space, and in some cases even whitespace.
- The above-described issue can also be triggered by an absolute positioned image inside of a fixed position element, but *only* when that image is using

something *other than its inherent dimensions*. If a height or width is specified on the image using CSS, or the image src is invalid (thus having no inherent height and width), this issue can occur. If an image that is inherently, say, 50x50 pixels is placed in a fixed element and left at its inherent dimensions, this issue does not seem to occur.

- When a `position: fixed` element appears anywhere on a page, most 2D CSS transforms will fail. Oddly, only `translate` transforms seem unaffected by this. Even more oddly, this issue is solved by setting a CSS `opacity` of .9 or below on the parent of the fixed element.
- Combinations of `position: fixed` and overflow properties are best avoided, as both have been known to cause unpredictable issues in older versions of Android OS.
- Any element that triggers the on-screen keyboard, when placed inside a `position: fixed` element, will fail to respond to user input when using anything other than the default keyboard. This includes Swype, XT9 or, it seems, any input method apart from the standard non-predictive keyboard.

While we will continue to try to find ways to mitigate these bugs as best we can, we currently advise against implementing fixed toolbars containing complicated user styles and form elements without extensive testing in all versions of Android's native browser.

Top

Fixed footer

No longer supported: touchOverflowEnabled

Prior to jQuery Mobile 1.1, true fixed toolbar support was contingent on native browser support for the CSS property `overflow-scrolling: touch`, which is currently only supported in iOS5. As of version 1.1, jQuery Mobile no longer uses this CSS property at all. We've removed all internal usage of this property in the framework, but we've left it defined globally on the $.mobile object to reduce the risk that its removal will cause trouble with existing applications. This property is flagged for removal, so please update your code to no longer use it. The support test for this property, however, remains defined under `$.support` and we have no plans to remove that test at this time.