High performance Widgets: Optimize your JavaScript

Introduction

This article is the number three of the series, **High performance Widgets**, and continues what we started in High Performance Widgets: CSS Sprites and High performance Widgets: Combine your JavaScripts and CSS in external Files .

JavaScript are getting more important every day. They have broken the web barrier and now are used in different platforms, such as our beloved WRT. To supply these new needs, the applications are getting larger, more complex and sophisticated. But if not watched, this sophistication can cost you a slow performance for your app, widget or website.

Like Joseph Smarr said, Performance is something that you need to take seriously from the day one. If you don't, may be too late when you find out.

There are millions of techniques to speed up your JavaScript. The objective of this article is to show some of this tips, that when applied to your Widgets or mobile websites, will help you to make it faster and better for the user experience.

Watch your DOM

Well, DOM interaction is not fast. I know that, and you know that. The Document Object Model can run slowly by different motives. One of the most common things that make your JavaScript slow is when too much things happen inside your DOM, like redrawing or reflows.

Redraw

Redraw is when the DOM changes the User interface in real time, like changing the background color, or the visibility of elements, effects that are frequently used by widgets developers. These DOM changes demands a lot of loading process, since it requires to the JavaScript to search all elements to determinate and execute them. That's why they're so heavy to load.

Reflow

Reflow is one of the most expensive functions of a browser. This is a far more significant change. The reflow happens when:

- When you add or remove a DOM node.
- When you apply a inline style.
- When you get a measurement or info that must be calculated, such as accessing offsetWidth, clientHeight, or any computed CSS value (via getComputedStyle() in DOM-compliant browsers or currentStyle in IE), while DOM changes are waiting to be made.

After any of these examples above, the engine must then reflow all the related elements to work out which and where the various parts of it should now be displayed. Its children, Ancestors and all the elements that appear after it in the DOM tree, will alson be reflowed to calculate their new layout, as they may have been moved by the initial reflows. After all this processes, finally, everything is redrawed.

Visualize Redraw and Reflow examples

This videos, show how the Redraw and Reflow process works on the web browser.

Gecko Reflow Visualization - mozilla.org
Gecko Reflow Visualization - Wikipedia

Good with them, better off without them.

Reflows and Redraws, affect the whole widget performance, because they affect the whole document. The more it happens, the longer the document will take to make all the changes.

Reflows something, that developers should keep at minimum, as long as they want to make their scripts running fast. But, Though animations, transitions and stuff are all based on reflows, we still want them to make our app prettier.

Notice that some elements have significantly slower reflows than others. Reflowing an element with table display, can take as much as three times as long as reflowing an equivalent element with block display

Apply animations to elements fixed or absolute.

(C) Copyright Nokia 2014. All rights reserved.

Elements which are positioned as Fixed or absolute don't affect the document or other elements layout, so they will only cause a repaint instead of a total reflow. This is much lighter and better for your widget.

How to get rid of them

Another common source of multiple reflows is making changes to an element's appearance via the **Style** (CSS in case you're using jQuery) property. For example:

```
//Style Changes using jQuery

$('element').css('color','red');
$('element').css('backgroundColor','#CCC');
$('element').css('fontSize', '13px');
```

This jQuery code has three style changes, what gives us three reflows. Each reflow happens linked to every change in the style of this element. The solution to keep reflows at minimum in this case, is to group all this changes in a new CSS class, and than use the jQuery or JavaScript to change the class.

Example:

```
    /*The same changes now are in the CSS*/

.newStyle {

    background-color: #CCC;
    color: red;
    font-size: 13px;
}
```

Then you reduce the javaScript to one line.

```
$('element').addClass('newStyle');
```

Changing the class of an element counts allows all of the styles to be applied at once, within a single reflow. This is much more efficient and also more maintainable in the long run.

Trade smoothness for speed

Means that you should choose the best option for your user. As tempting as making a smooth animation transition can be, you should set them to animate the fastest as you can. Instead of setting the transition animation using a 1 pixel(10ms) at a time you shoul try something less smooth, like a 5 pixels (50ms).

It can be neccessary to swallow the developer pride, and trade some of the smoothness of the animation for speed instead. The sophisticated animations, may make your widget look slow, because it will need much more processing power. That is a bad response to the user. I recomend you to use the simple Hide/Show jQuery Solution, which is a less smooth solution, but is the fastest response your user can get on lower powered processors.

Reference

See another tips about good JavaScript pratices in:

- Opera.Dev
- Mozilla.org
- Reflows & Repaints: CSS Performance making your JavaScript slow?

Tools

(C) Copyright Nokia 2014. All rights reserved.

- Lindsey Simon wrote a bookmarklet for you to test reflows in any browser.
- John Resig's bookmarklet to visualize paint events.

See Also

- High Performance Widgets: CSS Sprites
- High performance Widgets: Combine your JavaScripts and CSS in external Files

(C) Copyright Nokia 2014. All rights reserved.