<<< previous article

Battle of the Browsers – BUI Wins

The World Wide Web has greatly matured over the last couple of decades and has now become an indispensable and integral part of our lives. While a browser's main job still is to deliver information to a user, the most radical change lately has been the aspect of interactivity. In the old days, Web pages presented static information with a few hyperlinks sprinkled throughout. In contrast, today's Web pages take advantage of several new technologies to present dynamic content. We can now view a Web site that customized all of its content – even advertisements! – to the visitor and allows the users to interact with the site to accomplish their specific goals, rather than being limited to passively viewing predefined content that may not apply. New technologies such as AJAX play a huge role in this change, allowing webmasters to design Web sites that change page content on-the-fly without having to redirect the user to a new page. The end result is a rich, immersive, and topical Web experience that customers have come to expect. BASIS' Browser User Interface (BUI) dovetails into this user-interactive experience perfectly, by running fully-functional Business BASIC applications natively in the client's Web browser.

By Nick Decker
Engineering Supervisor

A Browser Isn't a Browser, Isn't a Browser

Over the years, one of the biggest challenges for webmasters was ensuring that their Web sites looked good and performed well on a variety of different browsers. Although Web standards exist for the underlying technologies such as HTML and CSS specifications, it is well known that some browsers did not fully implement the specs, had bugs that prevented features from working properly, or in some cases, purposely eschewed proper behavior in favor of supporting older broken behavior for backwards compatibility. These challenges are trying at best for your typical webmaster, and as time goes on things haven't improved as much as one would think or hope.

On the the bright side, however, Microsoft's Internet Explorer 8 (IE8) serves as a example of how browser manufacturers are now striving to adhere to industry standards. As a case in point, IE8 offers the ability to render a page in 'strict mode' that boasts stringent adherence to W3C Web standards (in addition to the default 'quirks mode' that favors backwards compatibility with older versions of IE). This is good news for Web developers, as it eases the burden of development, but it is also telling in that it shows that browsers must now work well in order to remain competitive. The olden days of using the default browser installed on a system are over, as stand-alone browsers such as Mozilla's Firefox and Google's Chrome have become extraordinarily successful. Their increased development schedules, improved security mechanisms, advanced plug-in and extension architectures, and improved reliability and performance allowed them to leap-frog the competition from Redmond and have sparked a heated competition for the user's desktop and handheld devices.

The other bit of good news is that despite the vast number of differences in browsers today, most of the onus of making BUI programs run seamlessly on a variety of browsers has been taken on by the GWT (Google Web Toolkit) team and the BASIS engineers. Even though differences do exist from browser to browser, in many cases the BBx developer is shielded from these anomalies as the same application code runs similarly in multiple browsers. >>

There still are, and always will be, numerous differences between browsers but many of these differences are benign and workarounds exist for some of the more egregious ones.

Browser Differences for BUI Applications

In a nutshell, a Web browser's task is to take building blocks such as HTML code, CSS styles, text, fonts, images, and scripts such as JavaScript and put them all together to present a coherent user interface. Given the complexities of the source combined with adherence to various levels of specifications, it's easy to see how each browser's rendering engine could come up with a slightly different result. In some cases, the differences are subtle such as a control looking slightly different or sized differently in one browser compared to another. In other cases, the differences may be profound as various JavaScript speed tests have shown some browsers to be over 1,000% slower than their competitors in certain tests. All of these variations stem from the fact that browsers use proprietary engines to perform complex tasks such as layout and script execution.

As time goes by, browsers get more and more competitive with one another, vying for the title of fastest browser. Manufacturers realize that a fast browser results in a speedy, smoother, more satisfying Web experience. JavaScript performance in particular is highly contested, as it is ubiquitous and is responsible for critical concepts such as client-side validation, document manipulation, animation, and more. JavaScript speed is also one of the key factors that determines how quickly and responsively a BUI application performs, making it an important consideration when choosing a target browser to deploy a BUI application suite.

Look and Feel Considerations

As mentioned earlier, various controls may render slightly differently across various browsers. Often these differences are so minimal that it's not likely anyone would ever notice. However, differences become more prevalent when the browser changes the look of a particular control, such as a button, to adhere to the standard for that browser on the target operating system. To illustrate this point, take a closer look at the BBjListButton control that appears in the BUI Customer Maintenance demo. **Figure 1** also illustrates these; the shape and coloring of the control, the look of the drop down box on the right side of the control, and even the font and text alignment.

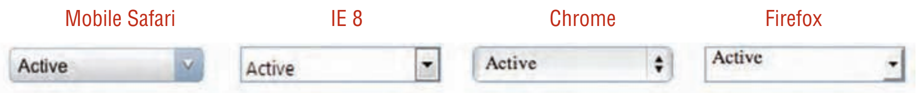


Figure 1. The BBjListButton in different browsers and platforms

In addition to the 'Look' portion, the 'Feel' of the controls varies as well. Using the same BBjListButton control example, selecting the drop down button on the right of the control (usually denoted by a disclosure triangle) causes the control to present the user with a list of possible choices. The type of list presented will differ, with some resulting in a drop down list, some with a pop up list (see **Figure 2**), and a native picker control on Mobile Safari.

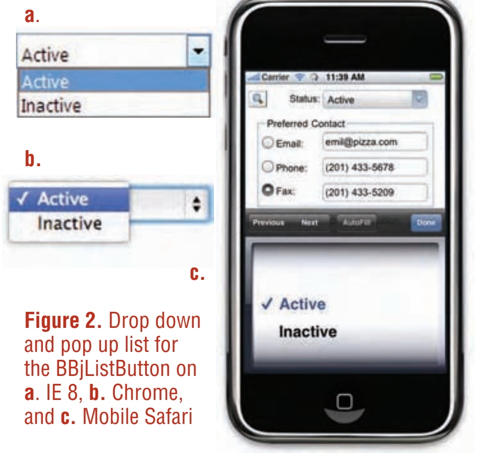


Figure 2. Drop down and pop up list for the BBjListButton on **a.** IE 8, **b.** Chrome, and **c.** Mobile Safari

Speed Considerations

Now that browsers play such a large role in our everyday computing life, browser manufacturers know that speed makes a huge difference in how well the market receives a browser and ultimately its popularity. Evidence of this comes from several sources, including the manufacturers themselves, who prominently display statistics as selling points like Opera's "*Our further-optimized JavaScript engine is over 50% faster than in Opera 10.5*" (and that's just one of the many differences in a 'point release' going from 10.5 to 10.6!). As manufacturers vie for the title of fastest browser on the planet, the big winner in the browser war are the end users. They, after all, get to enjoy the rewards that are a direct result of this heated competition.

Over the years, browsers have become several times faster due to this browser race and have been further spurred on by popular benchmark suites like SunSpider. Several different benchmarking suites exist, making it relatively easy to make direct JavaScript speed comparisons between multiple browsers. This information has proven interesting when comparing alpha and beta versions of upcoming browsers, but is extremely valuable and directly applicable when reviewing the current crop of released browsers in order to provide a recommendation for your department or end users.

One of the more challenging problems facing today's system administrators is that a browser like Microsoft's Internet Explorer is still the most popular, despite the fact that it's one of the slowest browsers available. According to StatCounter (see **Figure 3**), IE still holds the lion's share of the global browser market. However, it's popularity has been steadily dwindling over the years going from over 70% a few years ago to dipping below 50% for the first time in September of 2010. This is proof that newcomers, such as the super-fast Chrome, have been steadily gaining acceptance over the last year, slowly eating away at IE's dominance. The writing is on the wall for the older, slower browsers – improve or be left behind.

To Microsoft's credit, they have taken both speed and standards compliancy to heart with their upcoming Internet Explorer 9. The SunSpider result graph in **Figure 4**, taken from their recent tests, indicate how their various releases of IE9 Platform Preview compare to the older IE8 as well as the current crop of competitors. The latest pre-release of IE9 is 1,282% faster than IE8 – what a difference!

While those improvements are remarkable and give us hope for the future, using an unfinished Platform Preview release in a production environment is not at all feasible. If your customers and end users are still using IE6, 7, or 8, wouldn't it be great if there were a simple, free way to give >>

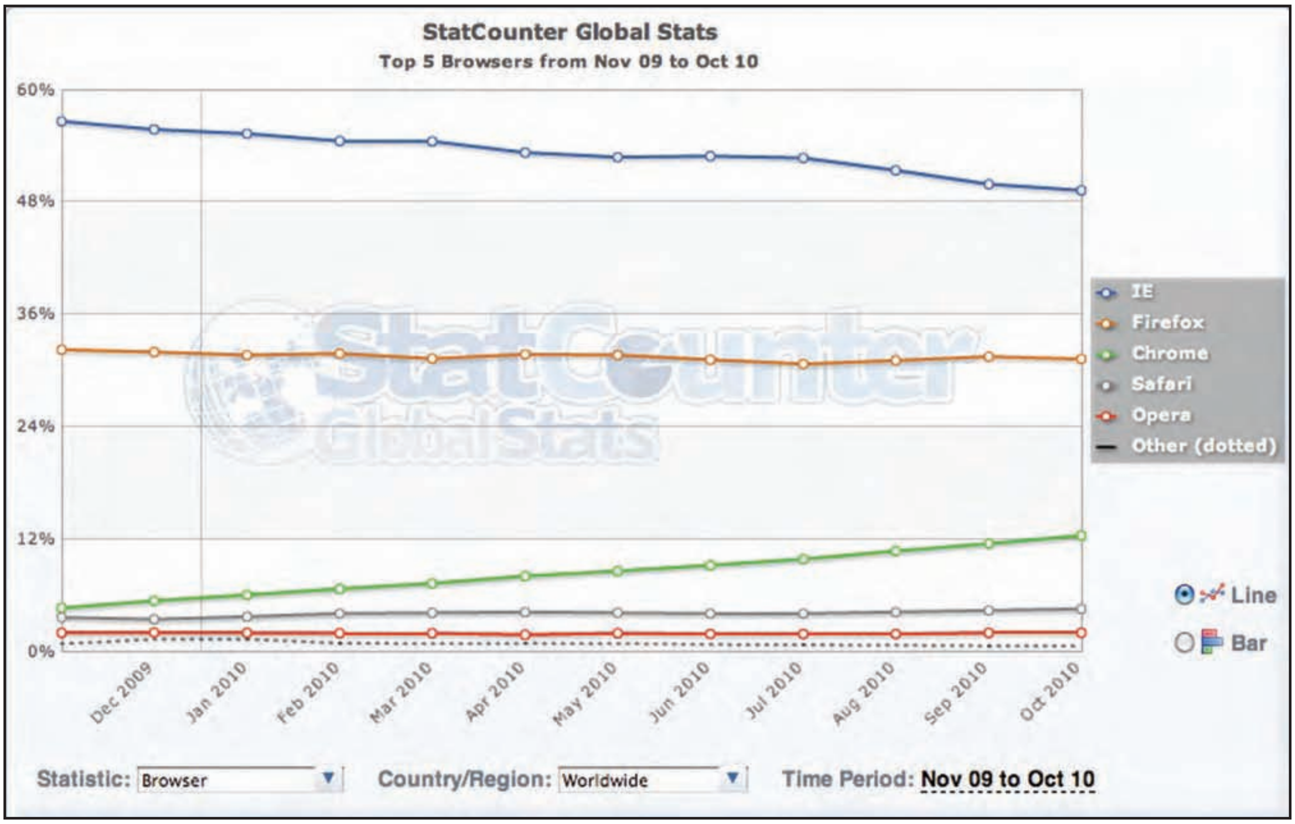


Figure 3. StatCounter's global browser share for the last year

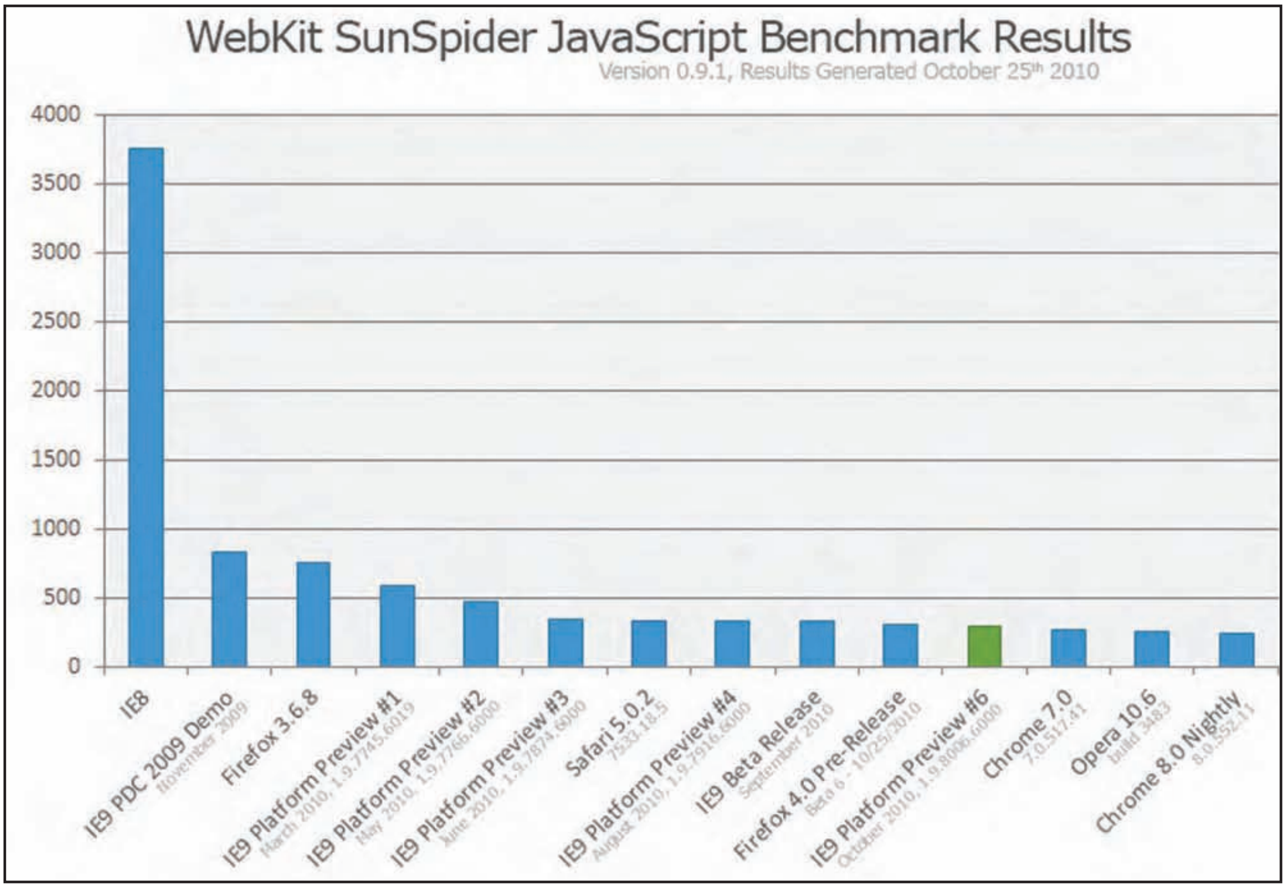


Figure 4. Benchmark results comparing various browser's JavaScript performance (lower is better)

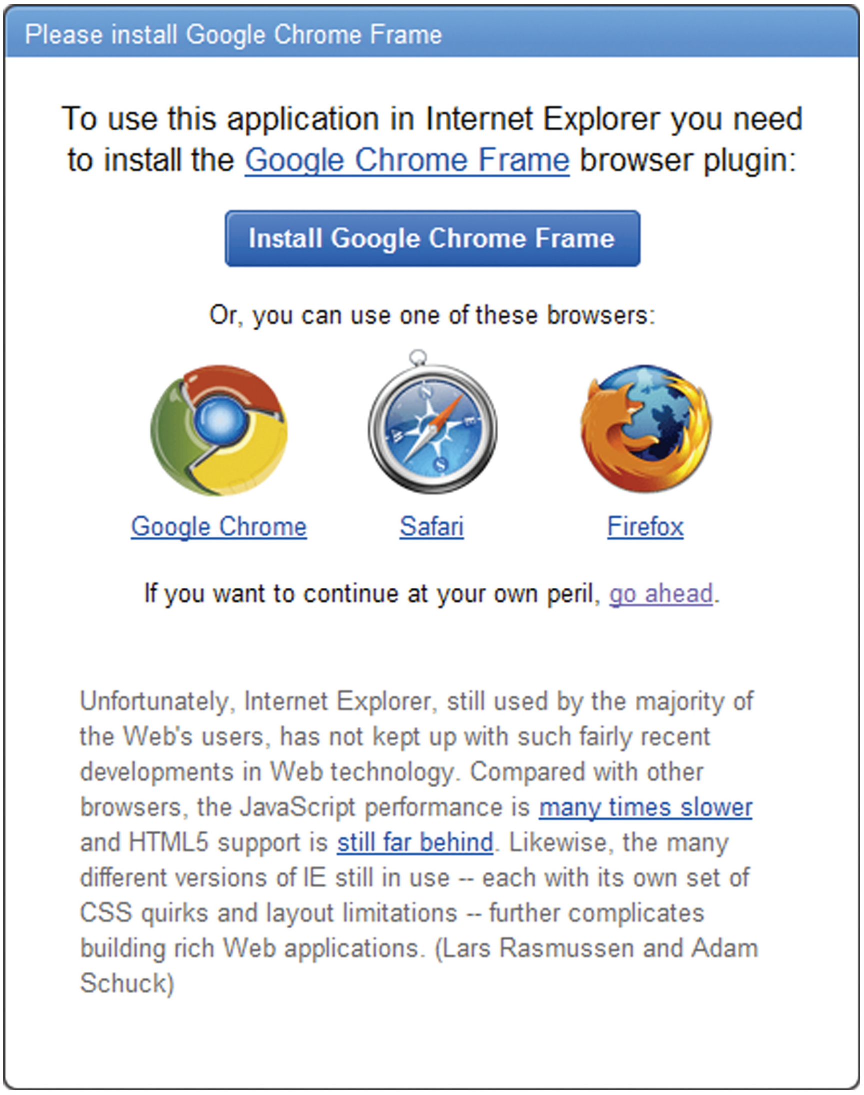
them a massive boost in speed and throw in support for progressive technologies like HTML5 and CSS3? It turns out that is not only possible, but only takes a minute to download and install via Google's Chrome Frame.

Google's Chrome Frame for Internet Explorer

To ameliorate the performance and compliance problems with the older versions of Internet Explorer (IE), Google released the Chrome Frame. Their Web site code.google.com/chrome/chromeframe/ aptly sums it up with the following text:

> Google Chrome Frame is an open source plug-in that seamlessly brings Google Chrome's open web technologies and speedy JavaScript engine to Internet Explorer. With Google Chrome Frame, you can:
>
> - Start using open web technologies – like the HTML5 canvas tag – right away, even technologies that aren't yet supported in IE 6, 7, or 8.
> - Take advantage of JavaScript performance improvements to make your apps faster and more responsive.

BASIS engineers added special code to the BUI system to automatically take advantage of Chrome Frame in IE if it is installed. This means that once Chrome Frame has been installed as a plug-in for IE, then running a BUI app in IE is just like running it in Google Chrome – it's fast and renders more accurately – all without the end user or BBx programmer having to do anything extra. BASIS also modified their online documentation to use the Chrome plug-in, when available, so that the documentation will render as quickly as possible. If you attempt to run a BUI application in a version of IE that does not yet have the Chrome Frame installed, the BUI system will bring up the screen in **Figure 5** to inform the user and facilitate the installation.



Please install Google Chrome Frame

To use this application in Internet Explorer you need to install the Google Chrome Frame browser plugin:

Install Google Chrome Frame

Or, you can use one of these browsers:

Google Chrome | Safari | Firefox

If you want to continue at your own peril, go ahead.

Unfortunately, Internet Explorer, still used by the majority of the Web's users, has not kept up with such fairly recent developments in Web technology. Compared with other browsers, the JavaScript performance is many times slower and HTML5 support is still far behind. Likewise, the many different versions of IE still in use -- each with its own set of CSS quirks and layout limitations -- further complicates building rich Web applications. (Lars Rasmussen and Adam Schuck)

Figure 5. The screen prompting the user to install the Google Chrome Frame for IE

Summary

Web browsers are now a vital part of virtually all desktop and handheld computers. With BUI, BASIS extends its promise of BBj's platform portability by running on an unprecedented number of platforms and devices. As browsers vary by manufacturer, platform, and device, the differences may become evident and have an impact on the performance and look and feel of your application running in BUI. While most of these differences are minor, speed is definitely worth looking into, as running your application on a slow browser instead of a fast browser is analogous to running your application on a slow computer instead of a fast one.

Now that the browser serves as the operating environment, choosing the right browser in which to run your application may mean the difference between a snappy, responsive application and a slow, lethargic one. As evidence of its commitment to the wide adoption of BUI, BASIS continues to test BUI applications on multiple browsers, and has made efforts to include support for Google's Chrome Frame plug-in in order to work around slower browsers. ■

Google Chrome Frame – www.google.com/chromeframe

next article >>>