The Browser Wars:

Chrome's Rise to Dominance

AlertSite
by SMARTBEAR

Contents

The Browser Wars

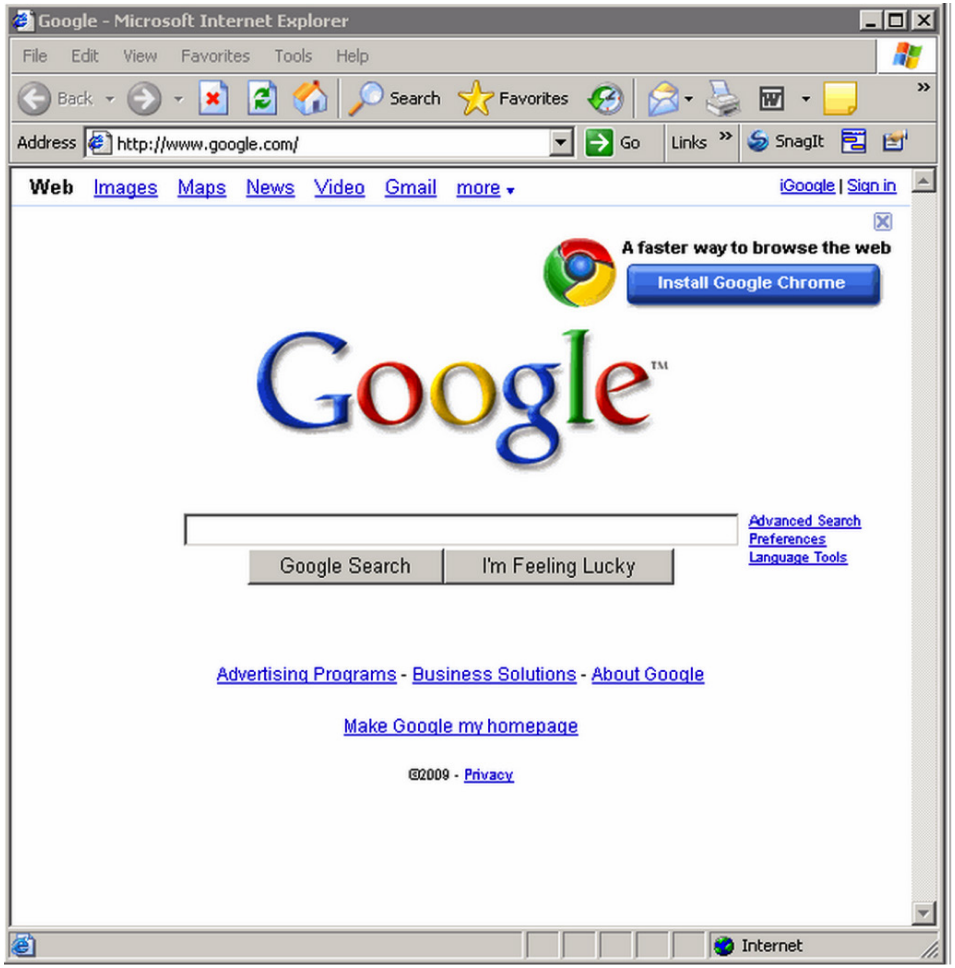
Introduction

Google Chrome came onto the scene in September 2008 — 43% of worldwide traffic was moving through it by December 2013, making it the most widely used desktop browser in the world.[1]

Chrome launched in a time period when Internet Explorer (IE) had entrenched itself as the undisputed leader in the browser market. Yet from day one, Chrome gained steady traction month after month. By the middle of 2013, Chrome had closed the gap on IE with each sharing 31% of all browser traffic. As of January 2014, Chrome had pushed IE's share down to just 25%.

So, what happened? After all, Microsoft had essentially been holding a monopoly over the industry for more than a decade.

Many say it was simply distribution. Chrome's success can be attributed to its presence on the Android line of mobile devices. That in effect, Android introduced mobile users to Chrome who then chose it for their desktops.[2] Then, of course, there is Google.com, the number one home page in the world, and Gmail, the number one web email client in the world — and here's what you see when you visit Google.com from an IE browser:



1. StatCounter Global Stats http://gs.statcounter.com/?PHPSESSID=bnp64opcri622hj9tvan6ulrj3#browser-ww-monthly-201212-201312
2. http://www.techspot.com/news/53166-statcounter-google-chrome-is-dominating-the-browser-industry.html

The Browser Wars

A lot of people have installed it — and love it. And why not? Chrome is faster than IE, often by a lot. It runs on every desktop platform anyone is likely to use. And it probably runs on their smartphones, too.[3]

Still, others say the single largest factor in Chrome's rise has been the rapid pace at which it adds new features. Google uses an agile approach to release management that doesn't wait for massive upgrades. The product is constantly evolving to appeal to new users and adapt to advancements in HTML5, CSS3 and JavaScript.

The Browser Wars

Google has made Chrome's rapid ascension look easy, but the fact is that the struggle for browser dominance, commonly referred to as "The Browser Wars", was vicious fight.

It began with Netscape launching in 1994, just as the Internet was taking off. But Netscape's hold on the browser market quickly crumbled after Microsoft started bundling IE with its Windows operating system.

By 2000, Microsoft's IE bundling strategy was so successful that the U.S. Justice Department said in a lawsuit that it constituted a monopoly and warranted breaking the company in two. Eventually, Microsoft won the suit — but the victory may have been what ultimately lead to losing the browser war.

From 2000 to 2006, Microsoft's IE5 and IE6 had a stranglehold on the browser market — a development that's been called "one of the worst things to happen to the Web as a platform." To maintain its dominance, the company embarked on a strategy of embracing, extending, and then extinguishing web standards. With IE6, Microsoft attained a 68% share of the market and virtually ceased development, leaving IE6 to stagnate.[4]

During this period, the browser business splintered across operating system lines. IE dominated Windows computers, while Safari dominated Apple machines.

By 2010, the landscape was far different. Just as the European Commission forced Microsoft to start offering Windows users a choice of browsers, many were already switching to various alternatives. Mozilla Firefox, being an open-source alternative, built a following of people who were drawn to its independence and vast array of community-built add-ons. Off in Eastern Europe and Western Asia, Opera was also quietly developing a loyal following.

The battle that got Microsoft into so much antitrust trouble didn't end but faded away, proving that distribution — no matter how tightly integrated — is not enough to win the hearts and minds of Web users. Performance matters too much.

"By 2000, Microsoft's IE bundling strategy was so successful that the U.S. Justice Department said in a lawsuit that it constituted a monopoly and warranted breaking the company in two."

3. http://www.zdnet.com/the-best-windows-browser-today-ie-10-or-chrome-25-7000011862/

4. http://farukat.es/journal/2013/02/682-why-opera-adopting-webkit-its-browser-engine-really-good-web

The Browser Wars

In the end IE's market share plummeted, first because of competition from Mozilla Firefox and Opera, and later because of Google Chrome. These alternatives were separated by more than just a few new features — they were each built on entirely different rendering engines, which, as we shall see, offered their respective users very different benefits.[5]

Engines that Drive Competition

A browser is transparent technology. For the most part, you don't really think about your browser much; you just use it.

If you've done any amount of front-end web development, you're bound to have noticed that not all browsers render all web content in exactly the same way. In the same way that appearance can be different, so too can performance. In fact, how a browser renders page content effects both appearance and performance.

The core of any browser is its rendering engine. As the name implies, the rendering engine's function is to take the code that has been received by the browser (mostly HTML, CSS, and JavaScript), interpret it, and present it. Each rendering engine works differently according to how it was programmed.

Today there are four mainstream rendering engines:

- Trident (used by Microsoft for Internet Explorer)
- WebKit (used by Apple for Safari, previously used by Google for Chrome)
- Blink (recently developed by both Google Chrome and Opera)
- Gecko (developed by Mozilla for Firefox)

Because they rely on different rendering engines, modern browsers offer varying levels of support for different coding techniques and advancements like HTML5

and CSS3. Something as simple as rounded corners on a CSS object can look radically different from one rendering engine to the next. When these new, cutting-edge web features don't work the same across modern browsers — even though they are "standards" — developers and webmasters need to a strategy for dealing with them.

Type rendering introduces yet another layer of complexity to the issue of browser performance. Each operating system contains a type-rendering engine — sometimes multiple engines from which to choose. And each browser controls which of these type-rendering engines is called on when displaying a webpage. So it's common for a website's type to appear completely different when the page is accessed through two different browsers on the same OS — because each browser is calling on a different type-rendering engine.

5. http://farukat.es/journal/2013/02/682-why-opera-adopting-webkit-its-browser-engine-really-good-web

The Browser Wars

Then there are differences in CSS handling. In an ideal world, you would only need one set of CSS style sheets for your website, and those styles will work across every browser. This, as every webmaster knows, is a CSS pipe dream — and it's an issue that is only exacerbated as the number of rendering engines continues to grow.

The use of conditional stylesheets became mainstream during IE's heyday. As stated earlier, Microsoft was notorious for picking and choosing what standards to support, including their support of non-standard features. Getting a site to render correctly with each successive version of IE was a challenging to say the least. To fix anything that didn't render correctly, site developers would use conditional style sheets — a stylesheet designed for a specific browser version. In some cases, a site might have 2 or 3 of these conditional stylesheets for each release of IE5 through IE8.

The problem with conditional style sheets, at least from the point of view of site administrators, is that each additional stylesheet creates an additional HTTP download request. When a page loads, it waits until stylesheets are fully loaded before rendering anything. This wait can leave users staring at a blank page.

Those that advise against conditional stylesheets often pursue a strategy of "graceful degradation." This is based on the general idea that rendering engines deal with things they don't understand pretty well. If a browser encounters an unknown CSS property, it will just ignore it and move on. If a browser encounters an unknown HTML element, it will treat it like an anonymous inline element. This means that if a browser is served some HTML or CSS features it can't understand, it will generally just ignore them and move on, rendering the rest of the content. Sometimes, these missing features may make the site unusable, but not often. In fact, because of this rendering engine design, even older browsers can render a usable result by gracefully degrading the page.[6]

While standards help ensure all browsers can handle all site features, they are far from perfect. As the number of rendering engines proliferates, expect to see less consistency in how browsers handle emerging standards, and that means more potential performance issues for web administrators to contend with.

> "Microsoft was notorious for picking and choosing what standards to support, including their support of non-standard features."

What Belarus Can Teach Us About Browsers

You'd probably be surprised at how many people have never heard of Opera, or only know it as a mobile browser. Given that, you might also be surprised to learn that it's the world's fifth most popular web browser. It's also the second-oldest active browser in the world.

6. http://www.w3.org/community/webed/wiki/Optimizing_content_for_different_browsers:_the_RIGHT_way

The Browser Wars

When it was released by Norwegian techies in 1996, Opera filled a void the same way that Firefox and Chrome would later. By offering new features the incumbents didn't, Opera created a browser that web geeks preferred. For a while, Opera was real competition for Microsoft's IE and Netscape's Navigator.

Opera's success was largely due to it its own rendering engine Presto. It was specifically created to challenge the dominance of companies who, at that time, were out to "win" the web — And its success was largely due to its incredible efficiency.

Long before the first iPhone, Opera was bundled with millions of feature phones. In those days there were no high-resolution, multi-touch screens — meaning that browsing the web on a mobile phone was tantamount to torture. Opera was the only thing that made it tolerable. Opera fed everything through its proxy server, which in turn reformatted pages designed for desktops so that they were viewable on smaller mobile screens.

While Opera's mobile browser enjoyed worldwide success, its desktop offering has always been considered a niche product. However, this niche laid claim to most of Eastern Europe and Central Asia. For years, Opera was the dominant browser throughout "The Stans" — Kazekstan, Uzbekistan, and Tajekistan — and in former soviet countries, the Ukraine and Belarus.

Opera and its Presto rendering engine were able to dominate in these countries due to their limited Internet infrastructure. The first Asymmetric digital subscriber line (ADSL) plans without monthly traffic limits didn't start to appear in this area of the world until around 2009-2010 — and when they did they offered very limited speed. The inherently efficient Presto engine combined with features like the ability to allow users to strip out images and other bandwidth-gobbling web extras made Opera the de facto choice of the region.

Opera's dominance came to an end in 2011 in every country but Belarus, the last country to upgrade its infrastructure. Opera was surpassed by... you guessed it . . . Chrome.[7] But rather than wallow in self pity Opera recognized that the benefits that had once raised Presto to the top had become less critical to their users. Rather than fight a losing battle Opera embraced the rise of Chrome and eventually joined Google in developing a new rendering engine, Blink.

WebKit and Beyond

In January 2003, long before the advent of Chrome, Apple built Safari — the first mainstream browser using the rendering engine WebKit. The decision was a major blow to Mozilla — who at the time stood alone, locked in a David and Goliath battle with Microsoft.

Apple said of its decision, "The number one goal for developing Safari was to create the fastest web browser on Mac OS X. When we were evaluating technologies over a year ago, KHTML and KJS stood out. Not only were they the basis of an excellent modern and standards compliant web browser, they were also less than 140,000 lines of code. The size of the code and ease of development within that code made it a better choice for us than other open source projects."

The "other open source projects" Apple was referring to was the Mozilla's Gecko engine that powered Firefox. Over the years since its debut it had bloated to several million lines of code, rendering it slow, cumbersome, and difficult to maintain. Mozilla went in the opposite direction of Opera. Rather than keeping Gecko as "built-for-purpose," Mozilla tried to make it "built-for-every-purpose," incorporating email, and even chat, into its core.

WebKit is a lightweight yet powerful rendering engine that emerged out of KHTML, an HTML rendering engine developed by the KDE community in 2001. Like Gecko, WebKit is also open source, although its direction today is largely controlled by Apple.

The consensus of developers who use WebKit is that it's an outstanding rendering engine that lends itself to an extremely diverse assortment of practical uses. It's everywhere, and it continues to gain traction at an impressive rate.

7. http://www.theatlantic.com/technology/archive/2012/11/why-is-belarus-the-only-country-where-opera-is-the-most-popular-browser/265406/

The Browser Wars

Google originally chose WebKit as the rendering engine for Chrome for many of the same reasons Apple chose it for Safari 4 years earlier. Google successfully used WebKit to supplant Internet Explorer. So why is Google leaving WebKit behind? Chrome uses a different multi-process architecture than other WebKit-based browsers, and supporting multiple architectures over the years has led to increasing complexity for both the WebKit and Chrome developers —which, in turn, has slowed down the collective pace of innovation.

In 2013, Google created Blink from WebKit specifically to support the Chrome architecture. Blink is considered a "forked version" or a divergence from WebKit. It will give Google a simpler code base to maintain and from which it can develop new features. As Google software engineer Adam Barth says, "That should translate into faster iteration — and innovation." For example, by moving forward with Blink, he estimates Google will soon be able to remove more than 7,000 files from Chrome, containing 4.5 million lines of code, which frees developer resources for innovation.[8]

Now that Google is joining the ranks of platform operators with their own rendering technology, the rendering engine world is diverging again. Given Google's massive market share with Chrome and Android in both desktop and mobile iterations, the decision has huge implications for how the web — both mobile and desktop — will evolve.

Polishing the Chrome Experience

Google Chrome's star is on the rise. It's not enough for a website to simply work with this new shining star — it has to be fast. AlertSite's web-performance-monitoring tools provide development and operations teams insight into the availability, response times and overall performance of their websites and applications from more than 80 locations across the globe. Since adding Chrome support to DejaClick, AlertSite's transaction recorder, AlertSite has monitored more than 1.3 billion web pages, allowing administrators to better understand and optimize the Chrome web experience.

8. http://blog.chromium.org/2013/04/blink-rendering-engine-for-chromium.html

About AlertSite

AlertSite, a part of SmartBear Software, is a global leader in Web, API and mobile performance monitoring solutions that continuously improve the Web user experience. AlertSite uniquely provides both technical performance measurement as well as visual user experience measurement from more than 80 locations around the globe and from within your environment with InSite, a private monitoring location. AlertSite's services measure basic availability, Web performance using real instances of IE, Firefox and Chrome browsers, API performance and mobile website performance. Gain a real-time view of service quality in terms of availability, performance, consistency and the user experience. Check AlertSite out at: alertsite.com

About SmartBear Software

More than one million developers, testers and operations professionals use SmartBear tools to ensure the quality and performance of their APIs, desktop, mobile, Web and cloud-based applications. SmartBear products are easy to use and deploy, are affordable and available for trial at the website. Learn more about the company's award-winning tools or join the active user community at http://www.smartbear.com, on Facebook or follow us on Twitter @smartbear and Google+.

SmartBear Software, Inc. 100 Cummings Center, Suite 234N Beverly, MA 01915
+1 978.236.7900 www.smartbear.com

EBOK_AS_20140210-Transaction-101