STATE OF THE UNION | SPRING 2015

PAGE SPEED & WEB PERFORMANCE

SHARE THIS WHITEPAPER

Table of Contents

Introduction

When it comes to website load times, user expectations are in a constant state of escalation. In 1999, the optimal load time for an ecommerce site was 8 seconds.[1] In 2006, that wait time had been slashed in half, to 4 seconds.[2] By 2010, 57% of online shoppers stated that they would abandon a web page after waiting 3 seconds for it to load.[3]

Three seconds. In case study after case study, this is the point at which most visitors will bounce if a page is not loading quickly enough. Not coincidentally, case study after case study shows that this is when business metrics – from page views to revenue – are affected by slow page rendering.

Whether your goal is to convert browsers into buyers or ensure that your content is served to as many eyeballs as possible, your eye should be on this 3-second target.

Why Test Leading Ecommerce Sites?

In February 2015, we tested the load times of the top 100 leading retail web sites' home pages, as ranked by Alexa.com. We also collected page metrics such as page size, composition, and adoption of key performance best practices. We then analyzed this data alongside previous benchmark tests.

The goals of this research are threefold:

1. To provide visibility into the performance and page composition of leading ecommerce sites.
2. To understand how these sites load for real people in real-world scenarios, via popular browsers and connection types.
3. To give a historical perspective into performance, seeing where sites have come from in terms of page size, composition, and adoption of performance best practices, and predicting their future direction.

Except where otherwise noted, the results discussed in this report are for pages tested in Chrome 40. At the time of conducting this research, Chrome was the most widely used browser in the United States, with a market share of 36.12%.[4]

Key Findings

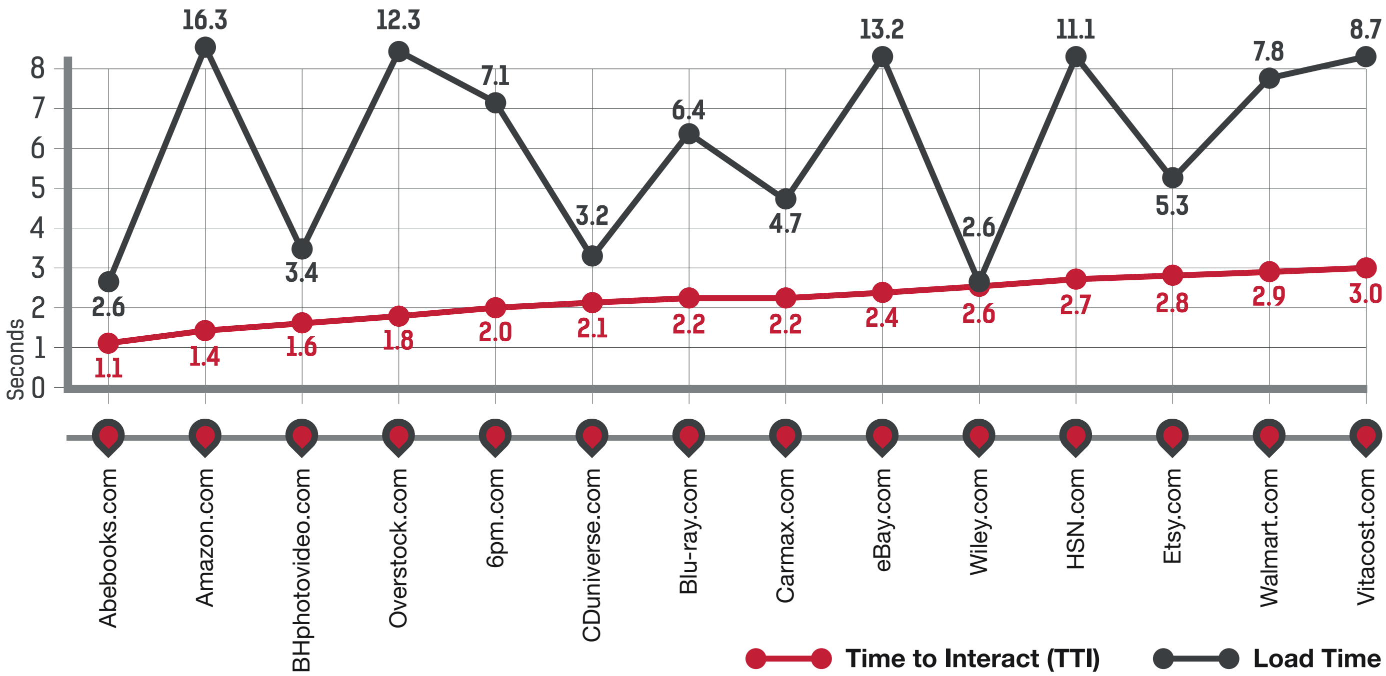
Our key findings are discussed in detail in this report and summarized below:

1. Among the top 100 pages, the median Time to Interact (TTI) is 5.2 seconds. This is considerably slower than users' reported wait-time threshold of 3 seconds.
2. Only 14% of the top 100 retail sites rendered feature content in fewer than 3 seconds.
3. 9% of the top 100 pages took 10 or more seconds to become interactive.
4. At the farthest end of the spectrum, the slowest page had a Time to Interact of 25.1 seconds.
5. The median page is 1354 KB in size and contains 108 resource requests. Page size and complexity typically correlate to slower load times.
6. Despite the fact that images comprise 50 to 60% of the average page's total weight, 43% of the top 100 sites failed to implement image compression, a core optimization technique.

Which Sites Were Fastest?

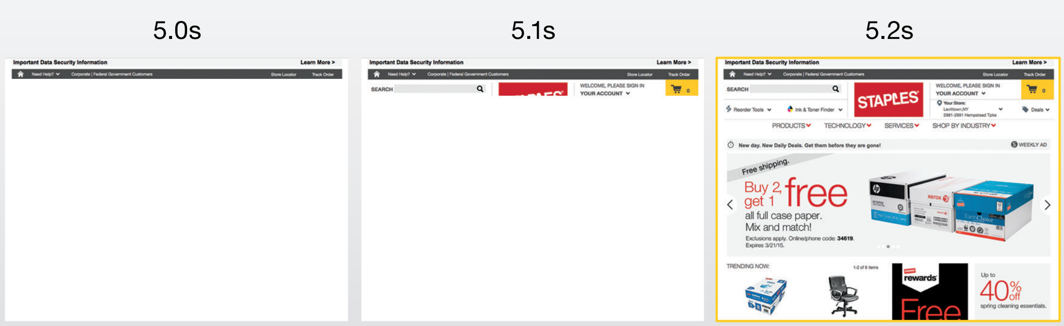
Among the top 100 ecommerce sites, these were the fastest home pages, defined by their ability to display meaningful, interactive content.

We have provided the time to interact alongside each page's full load time in order to give perspective into the distinction between the two metrics, and to illustrate **that load time is not always the most meaningful measure of a site's performance**. For example, while Amazon.com has a load time of 16.3 seconds, it has a TTI of 1.4 seconds; the TTI indicates that this site delivers a satisfactory user experience.



Finding 1: Among the Top 100 Pages, Median Time to Interact Is 5.2 Seconds

Time to interact (TTI) is a crucial indicator of a page's ability both to deliver a satisfactory user experience (by delivering content that the user is most likely to care about) and to fulfill the site owner's objectives (by allowing the user to engage with the page and perform whatever call to action the site owner has deemed the key action for that page). The median TTI in our research was 5.2 seconds.



PERFORMANCE METRICS DEFINED

Time to Interact (TTI)
The point at which primary page content renders and becomes interactive (e.g., feature banners with functional call-to-action buttons).

Load Time
The point at which all of a page's resources – from images to third-party scripts – have downloaded and rendered.

From a user experience perspective, TTI is a more meaningful performance metric than load time, as it indicates when a page begins to be usable.

Ideally, pages should be interactive in 3 seconds or less. Separate studies have found that:

- 57% of consumers will abandon a page that takes longer than 3 seconds to load.[5]
- A user who has to endure an 8-second download delay spends only 1% of their total viewing time looking at the featured space on a page. In contrast, a user who receives instantaneous page rendering spends 20% of their viewing time within the feature area of a page.[6]
- A site that loads in 3 seconds experiences 22% fewer page views, a 50% higher bounce rate, and 22% fewer conversions than a site that loads in 1 second, while a site that loads in 5 seconds experiences 35% fewer page views, a 105% higher bounce rate, and 38% fewer conversions.[7]

Given this proven relationship between a page's render time and business metrics, it is clear that a time to interact of 5.2 seconds fails to meet the needs of both consumers and site owners. Consumers are significantly more likely to abandon pages that take more than 3 seconds to load, and as a result, poorly performing sites can suffer conversion rate decreases of 22% or more.

In addition, we found that:

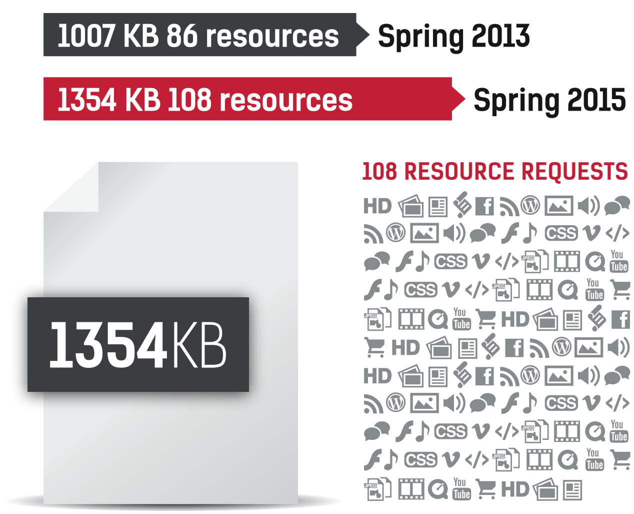
- Only 14% of the top 100 retail sites rendered feature content in fewer than 3 seconds.
- 9% took 10 or more seconds just to become interactive.
- At the farthest end of the spectrum, the slowest page had a Time to Interact of 25.1 seconds.

Finding 2: The Median Page is 1354 KB in Size and Contains 108 Resources

At 1354 KB, the median page is well over 1 MB in size and contains 108 resource requests (e.g., images, JavaScript, CSS files).

Page size is just one aspect of the problem. Page complexity is arguably an even greater performance challenge than page size. In the past two years, the median number of resources for a top 100 ecommerce page has grown by 26% -- from 86 resources in Spring 2013 to 108 resources in Spring 2015. Each of these page resources represents an individual server call.

For most sites, the greatest drain on performance is the need to complete dozens of network round-trips to retrieve resources such as style sheets, scripts, and images. Only 20% of the time required to display a typical web page is consumed by loading the page's HTML. The remaining 80% is spent loading the additional resources needed to render the page and performing client-side processing.



Each page resource makes an individual round trip from the user's browser, which requests the file from the host server, and in turn delivers the file to the browser. Each round trip can take 65-145 milliseconds (or more) for desktop browsers – numbers that add up quickly when a typical page contains more than 100 resources.

Not only does each server call introduce an incremental performance slowdown, it also increases the risk of page failure. For example, poorly executed CSS can create a host of performance problems, ranging from style sheets that take too long to download and parse to improperly placed CSS files that block the rest of the page from rendering.

Finding 3: Most Sites Fail to Take Advantage of Core Image Optimization Techniques

Images typically comprise between 50 to 60% of a page's total weight, making them fertile territory for optimization. Yet 43% of the top 100 retail sites fail to compress images. Only 10% received an 'A' grade for image compression.

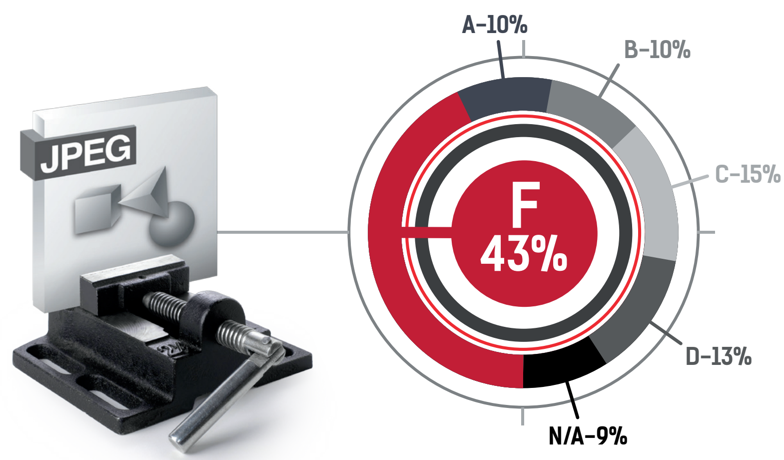


Image compression is a core performance technique that minimizes the size (in bytes) of a graphics file without degrading the quality of the image to an unacceptable level. Reducing an image's file size has two benefits:

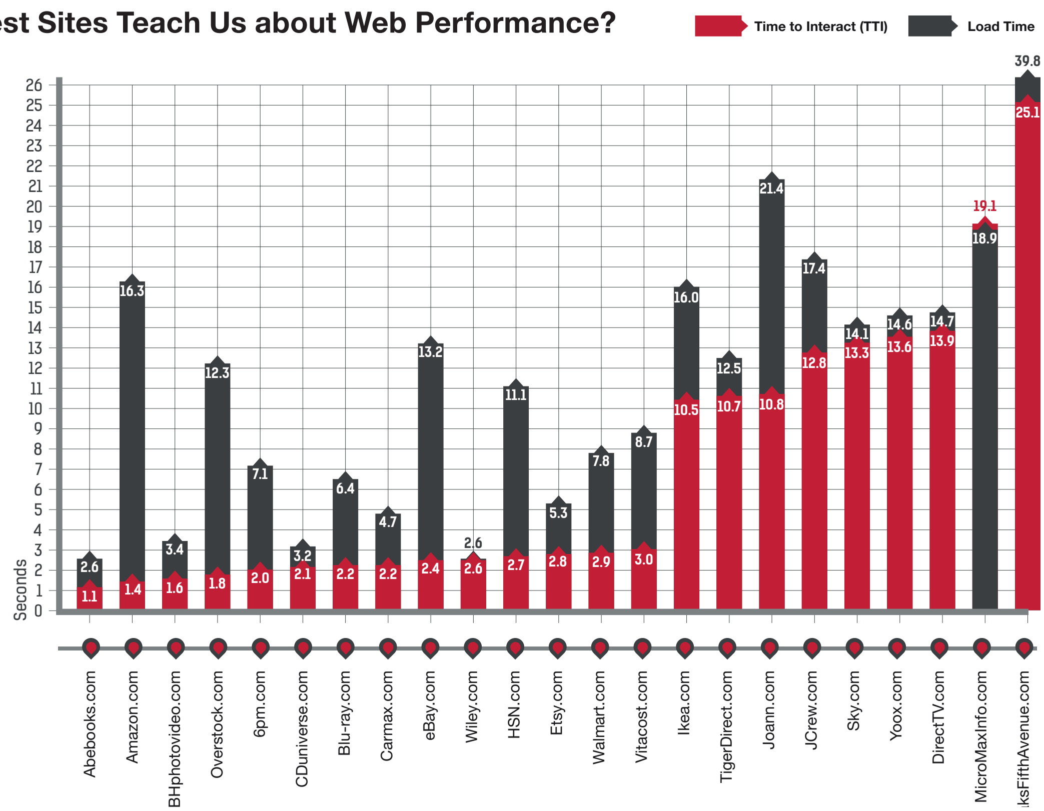
- Reducing the amount of time required for images to be sent over the internet or downloaded.
- Increasing the number of images that can be stored in the browser cache, thereby improving page render time on repeat visits to the same page.

Compressing image files lightens a web page's overall payload. Fewer bytes mean reduced bandwidth and faster pages.

With the advent of retina displays, consumer expectations of image quality have never been greater. Yet with the overwhelming advent of mobile usage, consumers also expect those same images to render quickly on their smartphones and tablets. In order to remain competitive, site owners must somehow miraculously meet consumers' demand for large, high-resolution product images while at the same time ensuring that those images don't clog the pipe to the user's screen.

What Can the Fastest Sites Teach Us about Web Performance?

Only 14% of the top 100 retail sites rendered feature content in fewer than 3 seconds, while 9% took 10 seconds or longer to become interactive. Let's take a deeper look at the fastest and slowest sites in our research and see what they have in common, where they differ, and what insights we can derive from this.

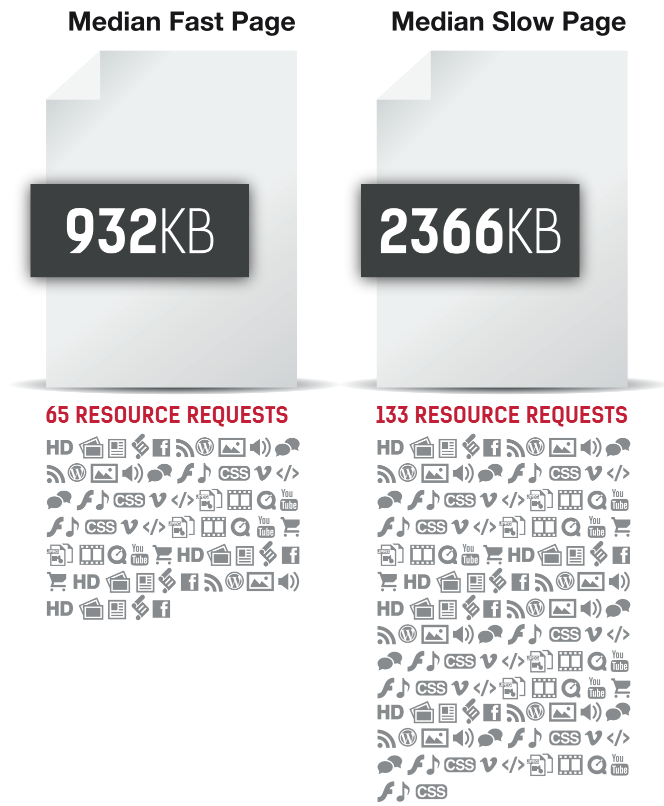


1. Faster pages are smaller.
Among the fastest pages, the median page contained 65 resource requests and was 932 KB in size. Among the slowest pages, the median page contained 133 resource requests and was 2366 KB in size.

In other words, **the median slow page was twice as large as the median fast page in terms of number of resources, and XX% larger in terms of size**.

Looking at the range of page sizes offers a bit more perspective. The smallest pages were as small as 337 KB and contained as few as 12 resources. At the other end of the spectrum, the largest pages were almost 6 MB in size and contained up to 375 resources.

(Note that these numbers refer to the moment the onLoad event fires in the browser, also known as “document complete time” – the amount of time it takes for *most* resources to render in the browser. Document complete time shouldn’t be confused with full load time – the amount of time it takes for *every* resource to render. There will be more discussion of this distinction in part 3 of this section.)



2. Faster pages have a faster Time to First Byte (TTFB).
Time to First Byte is the window of time between the browser asking the server for content and when it starts to get the first bit back. The user’s internet connection is a factor here, but there are other factors that can slow down TTFB, such as the amount of time it takes your servers to think of what content to send, and the distance between a site’s servers and its users. In other words, slow TTFB can be an indicator of a server problem or a content delivery network (CDN) (or lack thereof) problem — or both.

Among the fastest sites, the median TTFB was 428 milliseconds, compared to a median of 500 milliseconds for the slowest sites. This difference — 72 milliseconds — may sound insignificant, but it’s important to bear in mind that TTFB isn’t a one-time metric. It affects every resource on the page, meaning its effects are cumulative.

3. Faster pages understand their critical rendering path and know what content to defer.
Deferral is a fundamental performance technique. As its name suggests, deferral is the practice of deferring any page resources that are not part of a page’s critical rendering path, so that these non-essential resources load last. The optimal critical rendering path has been excellently defined by performance engineer Patrick Sexton as “a webpage that has only the absolutely necessary events occur to render the things required for just the initial view of that webpage”.[8]

Faster pages seem to have a better handle on deferral, which we can infer from looking at the difference between their page size metrics at doc complete versus their size when the page has fully loaded and all resources – including requests for “invisible” resources such as beacons and other third-party scripts – have rendered in the browser. As already mentioned, among the fastest pages in our study, the median page contained 65 resources and was 932 KB in size at doc complete. But looking at full load time, the median page contained 120 resources and was 1264 KB in size. In other words, the median number of page resources almost doubled between doc complete time and full load time, and the median page size increased by 36%.

Compare this to the ten slowest pages. The median page grew by only 11% -- from 2366 KB to 2616 KB, and from 133 resources to 134 resources – between document complete time and full load time. And in several cases, the difference between the doc complete and fully loaded metrics was either unchanged or only negligibly different. **These metrics indicate that these sites have not put a significant effort into leveraging deferral techniques to optimize the critical rendering path**.

A Tale of Two Websites: Walmart.com and eBay.com

Despite eBay and Walmart's status as retail giants, both sites have had less than stellar performance rankings in previous ecommerce performance studies. In our Fall 2014 report, these sites ranked 36th and 57th, respectively, out of 100. Our latest research, however, finds that both companies have made an impressive performance comeback – with each site's home page rendering primary content in fewer than 3 seconds.

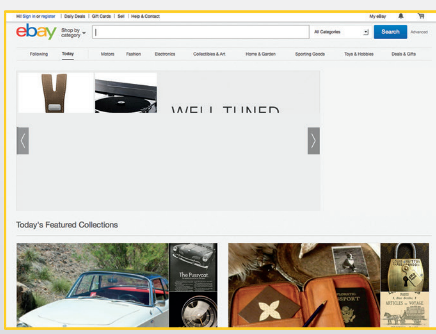
eBay.com: Sometimes Bigger Pages Can Be Faster

While eBay's home page in our Fall 2014 test was significantly smaller (in terms of both file size and total number of page resources) than the page tested for this report, that size did not equate to better performance. In Fall 2014, the page had a Time to Interact (TTI) of 5.2 seconds, while at the time of this test, the page had a TTI of just 2.4 seconds.

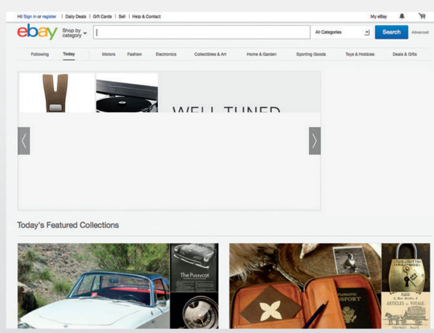
eBay.com	Fall 2014	Spring 2015
Speed Rank	36 out of 100	9 out of 100
Time to Interact	5.2 seconds	2.4 seconds
Load Time	12.1 seconds	13.2 seconds
Time to First Byte	611 milliseconds	655 milliseconds
Start Render Time	3.9 seconds	1.8 seconds
Page Size	786 KB	1326 KB
Resource Requests	181 resources	192 resources

Fall 2014

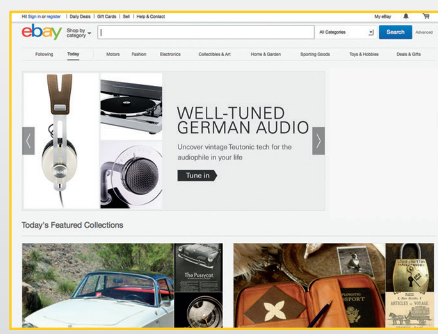
5.0s



5.1s

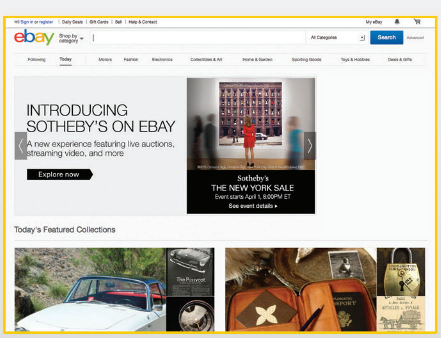


5.2s

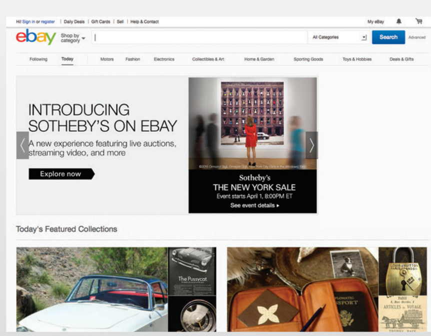


Spring 2015

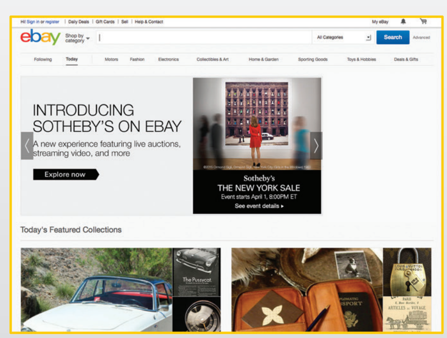
2.2s



2.3s



2.4s



While larger pages often correlate to slower load times, eBay.com demonstrates the value of leveraging a variety of optimization techniques to improve page speed. In the Fall 2014 page, the critical performance roadblocks were a small handful of feature images, each requiring between 3 to 6 seconds to download. The page also used a custom font file that was non-responsive for 2.7 seconds. By contrast, the page measured for this report contains leaner feature images, each requiring less than 2.5 seconds to download.

Walmart.com: Smaller Page Size Correlates to Faster Load Time

While the eBay case study goes against convention by demonstrating that a smaller page can, on some occasions, be slower than a larger one, Walmart.com offers a more conventional story. In our Fall 2014 research, the retail giant's home page contained 106 resources and came in at more than 2 MB in size. Visitors to the page were confronted with a blank screen until the 7.2-second mark. In our more recent test, Time to Interact had improved radically, from 7.2 seconds to 2.9 seconds.

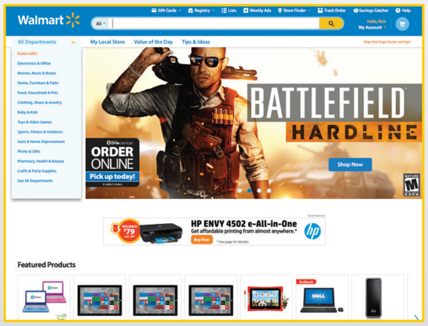
Walmart.com	Fall 2014	Spring 2015
Speed Rank	57 out of 100	13 out of 100
Time to Interact	7.2 seconds	2.9 seconds
Load Time	14 seconds	7.8 seconds
Time to First Byte	535 milliseconds	533 milliseconds
Start Render Time	7.2 seconds	2.2 seconds
Page Size	2282 KB	1183 KB
Resource Requests	106 resources	69 resources

Fall 2014

7.0s

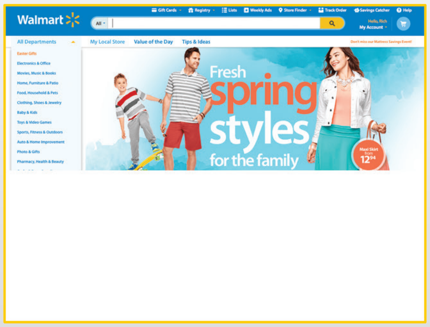
7.1s

7.2s

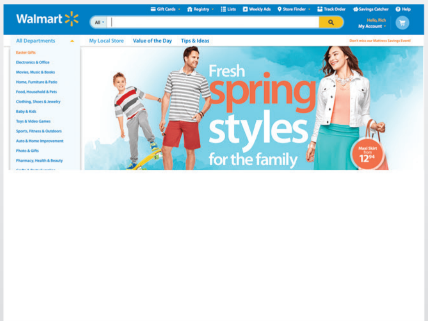


Spring 2015

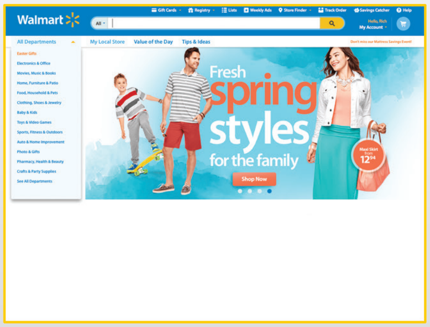
2.7s



2.8s



2.9s



Walmart.com shaved more than 4 seconds from its load time by removing several of the barriers that had been impeding the page from rendering: blocking JavaScript, slow custom fonts, and unoptimized image files that took up to 7.5 seconds to download. While the newer Walmart.com does contain some JavaScript and CSS files that are slower to render, these don't block the rest of the page from loading, hence the perception of faster load times.

14 Best Practices to Cure Your Web Site's Performance Pains

Throughout this report, there have been numerous references to performance techniques and best practices that site owners can leverage to optimize and accelerate page load times – ultimately improving both the real and perceived user experience for visitors. Some of these techniques can be implemented manually or via an automated solution, while others can only be performed by automated solutions.

1. Consolidate JavaScript and CSS

Consolidating JavaScript code and CSS styles into common files that can be shared across multiple pages should be a common practice. This technique simplifies code maintenance and improves the efficiency of client-side caching. In JavaScript files, be sure that the same script isn't downloaded multiple times for one page. Redundant script downloads are especially likely when large teams or multiple teams collaborate on page development.

2. Minify Code

Minification, which is usually applied to scripts and style sheets, eliminates non-essential characters such as spaces, newline characters, and comments. A correctly minified resource is used on the client without any special processing, and file-size reductions average about 20%. Script and style blocks within HTML pages can also be minified.

There are many good libraries available to perform minification, often along with services to combine multiple files into one, which additionally reduces requests.

3. Enable Keep-Alives

Enabling keep-alives is one of the easiest "low hanging fruit" on the performance optimization tree, yet a significant number of sites fail to do this. TCP connection is the process by which both the user and the server send and receive acknowledgment that a connection has been made and data can begin to be transferred. Too many TCP connections will slow down your site. It's not easy to speed up TCP connection, but you can control how many times the connection takes place. To enable keep-alives, make sure you have the proper configuration on your servers and load balancer.

4. Compress Text

Compression technologies such as gzip reduce payloads at the slight cost of adding processing steps to compress on the server and decompress in the browser. These operations are highly optimized, however, and tests show that the overall effect is a net improvement in performance. Text-based responses, including HTML, XML, JSON (JavaScript Object Notation), JavaScript, and CSS, can all be reduced in size by as much as 70%.

5. Sprite Images

Spriting is a CSS technique for consolidating images. Sprites are simply multiple images combined into a rectilinear grid in one large image. The page fetches the large image all at once as a single CSS background image then uses CSS background positioning to display the individual component images as needed on the page. This reduces multiple requests to only one, significantly improving performance.

6. Compress Images

Image compression is a performance technique that minimizes the size (in bytes) of a graphics file without degrading the quality of the image to an unacceptable level. Reducing an image's file size has two benefits:

- reducing the amount of time required for images to be sent over the internet or downloaded, and
- increasing the number of images that can be stored in the browser cache, thereby improving page render time on repeat visits to the same page.

7. Reformat Images

Inappropriate image formatting is an extremely common performance culprit. An image that is saved to the wrong format can be several times larger than it would be if saved to the optimal format. Images with unnecessarily high resolution waste bandwidth, processing time, and cache space.

As a general rule of thumb, these are the optimal formats for common image types:

- **Photos –** JPEG, PNG-24
- **Low complexity (few colors) –** GIF, PNG-8
- **Low complexity with transparency –** GIF, PNG-8
- **High complexity with transparency –** PNG-24
- **Line art –** SVG

8. Ensure That Feature Images Are Optimized to Load Early and Quickly

As discussed earlier in this report, site owners should be aware of the usability consequence of delaying the rendering of feature content: a user who experiences instantaneous page rendering spends 20% of their viewing time within the feature area of a page, whereas a user who endures an eight-second download delay spends only 1% of their total viewing time looking at the featured space on a page.

9. Rethink the Design and Location of Call-to-Action Links in Feature Graphics

While the accepted design convention has been to position CTA buttons at the bottom of feature banners, this convention does not always serve the best interests of users or site owners, as shoppers must wait for the image to fully render before taking their next action on the page. The simplest solution: Reposition the CTA.

10. Defer Rendering "Below the Fold" Content

Ensure that the user sees the page quicker by delaying the loading and rendering of any content that is below the initially visible area, sometimes called "below the fold." To eliminate the need to reflow content after the remainder of the page is loaded, replace images initially with placeholder <img> tags that specify the correct height and width.

11. Defer Loading and Executing Non-Essential Scripts

Many script libraries aren't needed until after a page has finished rendering. Downloading and parsing these scripts can safely be deferred until after the onload event. For example, scripts that support interactive user behavior, such as "drag and drop," can't possibly be called before the user has even seen the page. The same logic applies to script execution. Defer as much as possible until after onload instead of needlessly holding up the initial rendering of the important visible content on the page.

The script to defer could be your own or, often more importantly, scripts from third parties. Poorly optimized scripts for advertisements, social media widgets, or analytics support can block a page from rendering, sometimes adding precious seconds to load times.

12. Use AJAX for Progressive Enhancement

AJAX (Asynchronous JavaScript and XML) is a technique for using the XHR (XMLHttpRequest) object to fetch data from a web server without refreshing the page where the code is running. AJAX enables a page to display updated data in a section of a page without reconstructing the entire page. This is often used to respond to user interaction, but it can also enable your application to load a bare-bones version of a page quickly, and then to fill in more detailed content while the user is already viewing the page.

13. Preload Page Resources in the Browser

Auto-preloading is a powerful performance technique in which all user paths through a website are observed and recorded. Based on this massive amount of aggregated data, the auto-preloading engine can predict where a user is likely to go based on the page they are currently on and the previous pages in their path. The engine loads the resources for those "next" pages in the user's browser cache, enabling the page to render up to 70% faster.

Note that this is a data-intensive, highly dynamic technique that can only be performed by an automated solution.

14. Implement an Automated Web Performance Optimization Solution

While many of the performance techniques outlined in this section can be performed manually by developers, hand-coding pages for performance is specialized, time-consuming work. It is a never-ending task, particularly on highly dynamic sites that contain hundreds of objects per page, as both browser requirements and page requirements continue to develop. Automated front-end performance optimization solutions apply a range of performance techniques that deliver faster pages consistently and reliably across the entire site.

Conclusion: Faster Networks and Devices Are Not a Performance Cure-all

Page growth and complexity present critical web performance problems that cannot be entirely mitigated by technological advancement.

We typically assume that technology is getting better and faster. And in best-case scenarios – in which people are using great hardware, great networks, and well-designed, well-optimized pages – this is definitely the case.

But most of us tend to focus on this best-case scenario and live in denial of the fact that worst-case scenarios are rampant on the web. Site owners design their pages and applications with best-case hardware and networks in mind, to the detriment of every other technological use case.

Our use of the web today is highly situational. In the past, we could expect a relatively consistent user experience, as we used our desktop computers on speedy local networks to browse somewhat dynamic (but mostly static) pages that were hosted on, at most, three or four different servers. Today, we are more likely to use mobile devices on congested wireless networks to browse highly dynamic pages stuffed with rich content and hosted on dozens of different servers. Because of this, we are seeing an overall increase – not a decrease – in sub-optimal user experiences.

While we cannot affect the end-user environment, we do have a great deal of control over our web pages. Luckily, there is a wealth of opportunity to optimize our pages so that we can serve our visitors the user experience they expect and deserve.

If you care about delivering a faster user experience to your customers, then look to the fastest online retailers for insight. The most high-performing sites:

- contain smaller, leaner pages,
- understand the critical rendering path, and
- know what resources to defer.

The good news is that there are opportunities for every site — even the ones that are relatively fast already — to fine-tune performance by taking a more aggressive approach to front-end optimization.

Methodology

The tests in this study were conducted using an online tool called WebPagetest – an open-source project primarily developed and supported by Google – which simulates page load times from a real user's perspective using real browsers.

Radware tested the home page of every site in the Alexa Retail 500 nine consecutive times. The system automatically clears the cache between tests. The median test result for each home page was recorded and used in our calculations.

The tests were conducted on February 16, 2015, via the WebPagetest.org server in Dulles, VA, using Chrome 40 on a DSL connection.

In very few cases, WebPagetest rendered a blank page or an error in which none of the page rendered. These instances were represented as null in the test appendix.

Also, in very few cases WebPagetest.org rendered a page in more than 60 seconds (the default timeout for WebPagetest.org). In these cases, 60 seconds was used for the result instead of null.

To identify the Time to Interact (TTI) for each page, we generated a timed filmstrip view of the median page load for each site in the Alexa Retail 100. Time to Interact is defined as the moment that the featured page content and primary call-to-action button or menu is rendered in the frame.

About Radware

Radware (NASDAQ: RDWR), is a global leader of application delivery and application security solutions for virtual and cloud data centers. Its award-winning solutions portfolio delivers full resilience for business-critical applications, maximum IT efficiency, and complete business agility. Radware's solutions empower more than 10,000 enterprise and carrier customers worldwide to adapt to market challenges quickly, maintain business continuity, and achieve maximum productivity while keeping costs down. For more information, please visit www.radware.com.

Sources

1. Zona Research, The Economic Impacts of Unacceptable Website Download Speeds, 1999
2. Jupiter Research, Retail Web Site Performance: Consumer Reaction to a Poor Online Shopping Experience, June 2006
3. PhoCusWright, Consumer Response to Travel Site Performance, June 2010
4. StatCounter Global Browser Stats, February 2015
5. PhoCusWright, Consumer Response
6. Jakob Nielsen, Website Response Times, June 2010
7. Strangeloop Networks, The 90-Minute Optimization Life Cycle, June 2010
8. Patrick Sexton, Critical Rendering Path