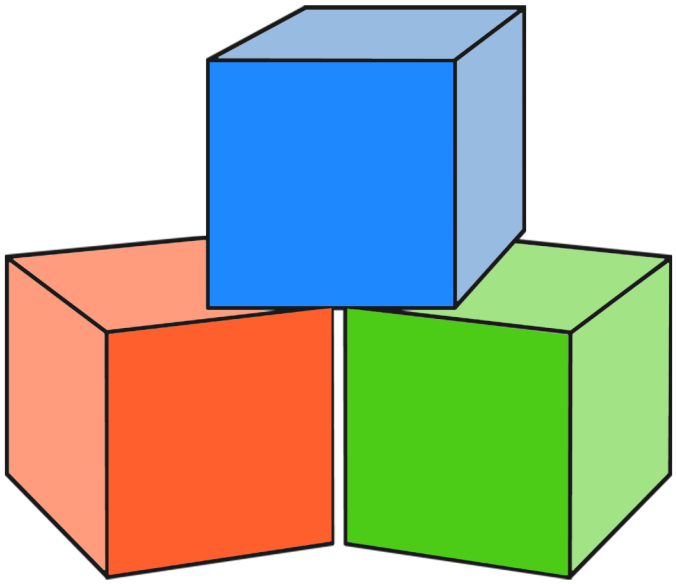


yottaa

A BEGINNER'S GUIDE TO

WEB PERFORMANCE

Table of Contents

Introduction to Web Performance...3

Executive Summary...4

The Causes of Poor Performance...5

ROI of Web Performance...10

How to Measure Performance...12

Web Performance Metrics - Front-End...13

Web Performance Metrics - Content...14

Basic Tools - Waterfall Chart...15

How to Improve Performance...17

Methods for FEO: Issue -> Resolution...18

Ongoing Action: Website Monitoring...19

WPO Checklist + Resources...20

Introduction to Web Performance

Web performance optimization (WPO) is the science of making websites load faster and more consistently for all visitors. The importance of WPO has been proven repeatedly: studies by Google, Wal-Mart, Amazon and others show that slower web pages have a direct and substantial impact on key business metrics such as conversion rate and sales.

This eBook is for anyone who is entirely new to WPO, or anyone who may be unfamiliar with its newer, more content-centric form. We cover:

- The **main causes** of poor performance
- Why WPO is important from the standpoint of **marketing and web operations ROI**
- How to **measure** performance
- How to **improve** performance
- How to **monitor** performance over time

These topics are covered at a high level, but suggestions for more in-depth reading are provided throughout the document. We hope you enjoy!

Executive Summary

Yottaa is a web performance optimization (WPO) company. Site speed is constantly on our minds. Finding new and creative ways to make sites faster is our passion.

We recognize, however, that not everyone shares our passion for speed. The challenges and concerns of running a web-based business are many. With marketing campaigns, SEO, building new website content, and everything else that comes with running a successful online business, it's no wonder that something seemingly as small as shaving a couple seconds off page load time often falls by the wayside.

In this eBook we hope to convince you otherwise: that web performance can and should be as integral as your other business processes. WPO is wide ranging field that has something for everyone. Whether your site has big images in need of compression, needs help to defend against the effects of sudden traffic spikes, or faster content delivery for visitors from the other side of the world, there is a tailor-made solution.

On the following pages we will describe how the industry has transitioned in focus over the years, and how the overwhelming bottleneck for performance is now related to content on your website. We'll review the ROI of WPO, backed up by metrics from the most successful web businesses. Then we'll cover the basics of how to find problems on your site, how to fix the problems, and what you can do to ensure continued performance over time.

We hope you enjoy reading and hope we can help you to speed up your website!

The Causes of Poor Performance

Performance problems stem from one of three stages in a website's journey to the visitor.

Backend

- DNS Lookup
- Server connection
- Server response

The first stage is datacenter performance -- also known as "**backend performance**." This involves the server processing the request sent by the browser and in turn sending out the correct website content.

The hangup: Heavy website traffic overloading servers, code bugs, inefficient code.

Middle Mile

- Content travels through networks to user

The second stage is the **"middle mile,"** where the content travels from the datacenter to the visitor's browser.

The hangup: When packets of information have to travel via inefficient routes and across great distances to arrive at the browser.

Front-end

- Browser download
- Browser rendering

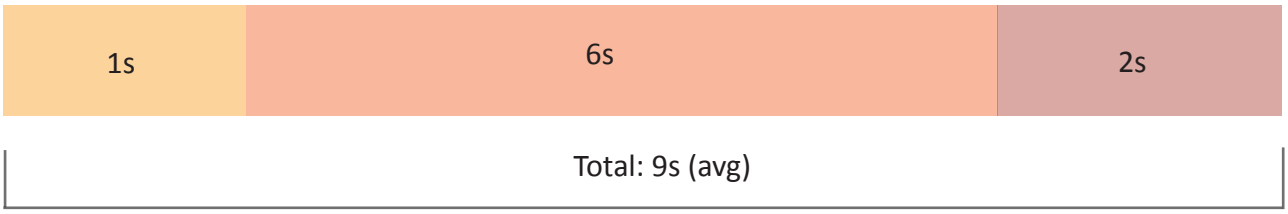
The final stage is "**front-end performance**." This stage consists of the visitor's browser rendering the page's content once the HTML file has been delivered.

The hangup: Downloading and rendering complex website content: numerous assets, heavy assets, and third party assets all result in problems.

The Causes of Poor Performance

What poor performance looked like: 1998

Sites should ideally load in 2 seconds or less to keep up with rising user expectations. Most sites do not. Here's what an average site's load process looked like in 1998. Middle-mile was by far the longest part of the process, because no solution had yet been invented to speed up delivery. In addition, the content of the sites was simple (primarily images and static HTML) so it was easy for browsers to render the content quickly. That's why the front-end only took 1-2 seconds.

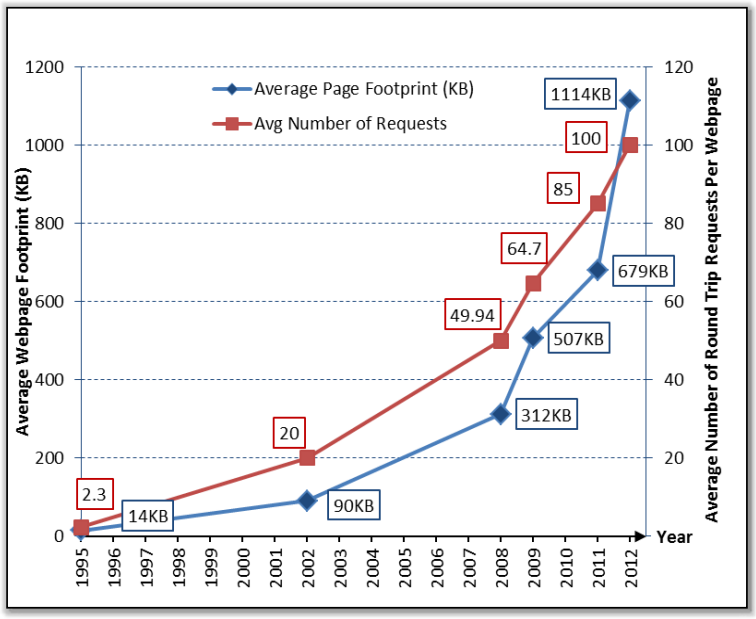


Key

Backend	Middle Mile	Front-end
• DNS Lookup • Server connection • Server response	• Content travels through networks to user	• Browser download • Browser rendering

The Causes of Poor Performance

Evolution of website content: massive growth in weight and number of assets per page



Sources: Demenech 2007, Gomez 2008, Charzinski 2010, Souders 2012

The growth of web pages: average number of assets per page grew from 20 to 100 between 2002 and 2012. Page weight grew from 90 KB to 1114 KB.

2000's - transition period

In the past decade, improvements to web infrastructure have made fast delivery across the "middle mile" easily achievable, particularly for sites willing to pay for add-on infrastructure services. This is mostly thanks to the creation and widespread adoption of **content delivery networks (CDNs)**, which bypass much of the middle mile by delivering website assets from "edge locations" much closer to the visitor.

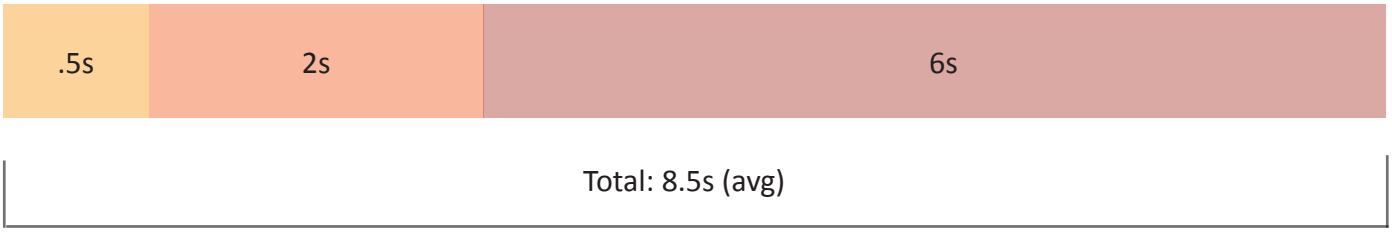
But as CDNs solved the middle mile bottleneck, increasingly rich content created a new one on the "front end" (the browser). In recent years website content has exploded in quantity, size, and complexity. The popularity of Social Media and advances in web development have made it commonplace for sites to be full of JavaScript, HTML5, media files, and other dynamic content. **In short, the average website has become a complex, distributed web application**. Put in numerical terms, the average website grew 13x between 2002 and 2012 (see left).

Rendering hundreds of kilobytes of complex content is a comparatively difficult task for browsers to accomplish, and the strain shows when one breaks down the page load process. **On today's websites, up to 90% of page load time is devoted to the "front-end" stage.**

The Causes of Poor Performance

What poor performance looks like today

Here's what a current major website might look like. The front-end, with today's heavy and complex content, is the primary bottleneck.

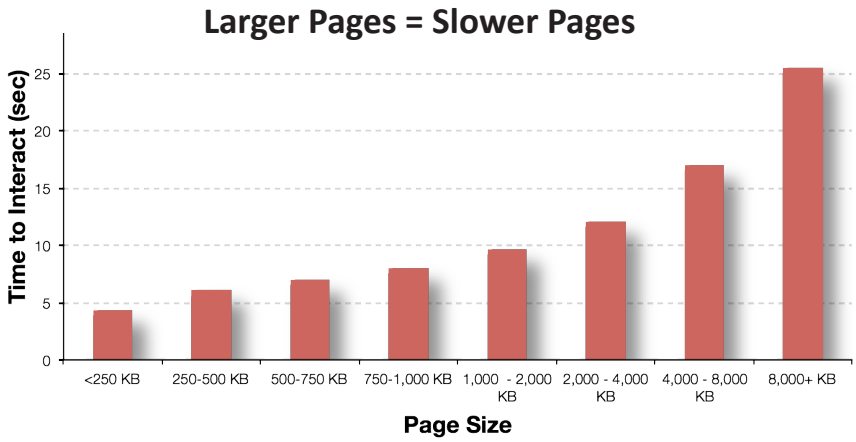


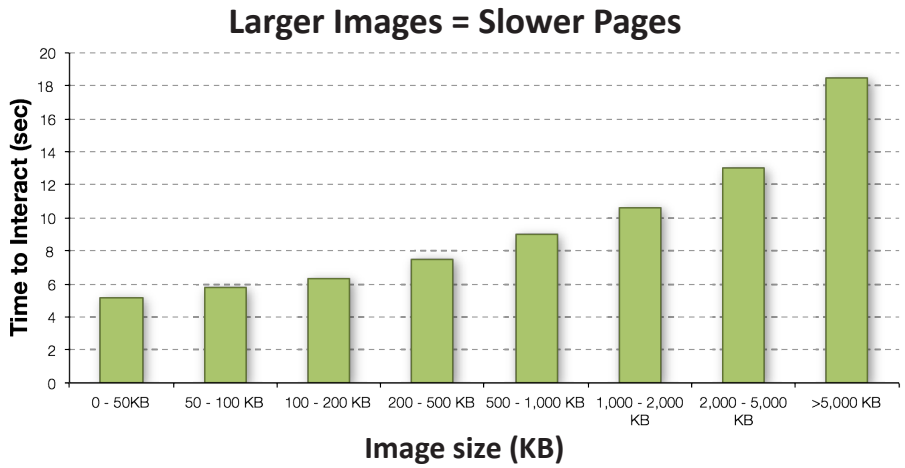
Key

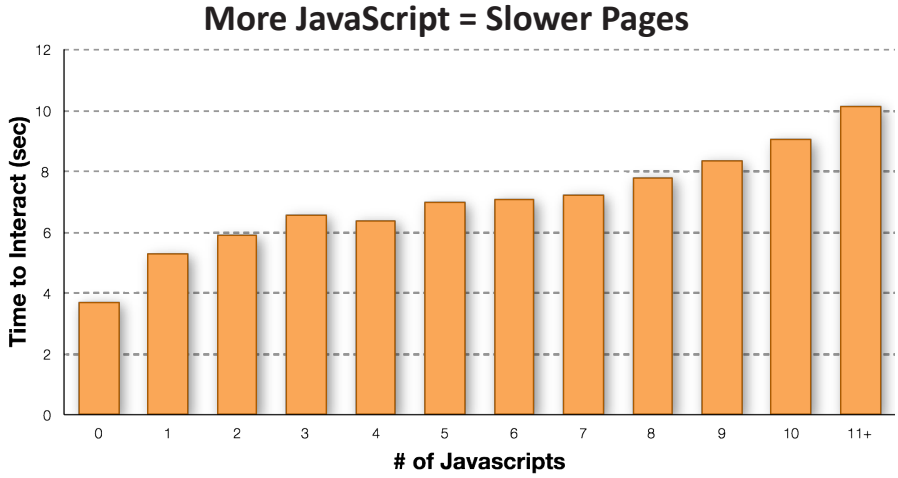
Backend	Middle Mile	Front-end
• DNS Lookup • Server connection • Server response	• Content travels through networks to user	• Browser download • Browser rendering

The Causes of Poor Performance

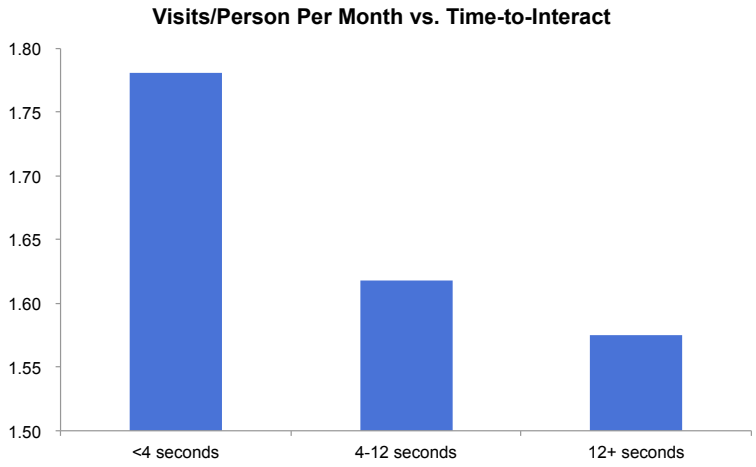
Focusing on the “front-end” side, below are graphs showing how the amount of assets and weight of content on a website can directly affect page load time. (The data was collected from a study Yottaa conducted of over 14,000 websites.)



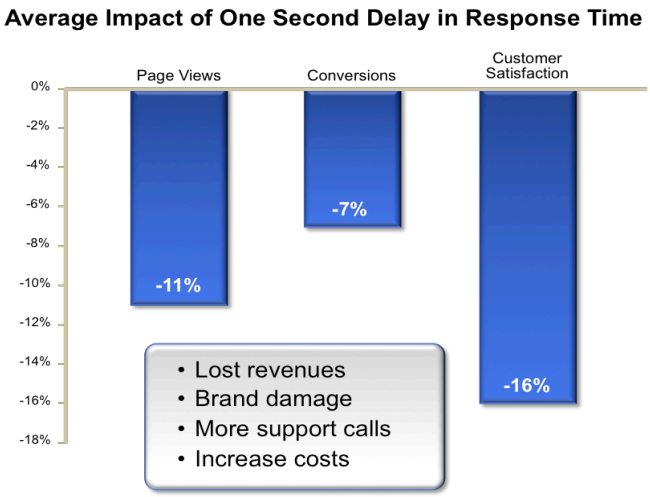




ROI of Web Performance



Yottaa performed a study on over 14,000 sites that showed increasing page load time resulted in decreasing visits per person, per month.



A study of websites from Aberdeen Group shows that a one second delay in page load time resulted in visitors viewing fewer pages, converting less often, and reporting a poorer overall experience when polled.

Poor performance affects websites both in the short run and the long run.

In the immediate term, a slow site can cause the loss of a potential customer who becomes impatient with the page load and bounces back to the search engine results page to find another site. A slow site can also cause a more invested visitor to abandon a shopping cart because of repeatedly slow page performance during the shopping process. Customers are less willing to trust their credit card information with a site that is struggling to render pages.

In the longer term, repeated events like those described above combine to impact business metrics. Sites with worse performance will attract fewer visits, worse rankings in search engines, and are more likely to be spoken poorly of by visitors who report to friends about their experience. In addition, studies have shown that:

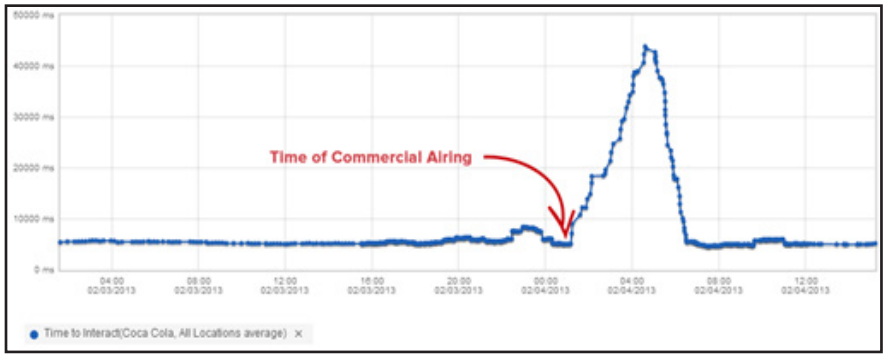
- A one second delay can reduce revenue per visitor by 3% [1]
- A one second delay reduces the number of pages viewed per visitor by 11% [2]
- Added delays result in a linear drop in visits per month [3]
- A 100 millisecond delay in page load can cause 1% drop in revenue [4]

[1+2] Aberdeen Group ; [3] Yottaa; [4] Amazon.com

ROI of Web Performance

If ROI statistics from web giants like Amazon seem less applicable to your business, consider the following case of a typical eCommerce site.

Let's imagine our site sells clothes, earning $10,000,000 per year in revenue. The site takes approximately 8 seconds to load (a bit worse than average, as we'll show later in this eBook). And let's assume that as a result of traffic spikes driven by successful marketing campaigns, as well as traffic from bots and crawlers, approximately 5% of the time the site performance is particularly slow or even experiences outages.



A good marketing campaign drives traffic, but can also drive up page load times.

So, what can be the impact of a performance improvement for this site? Let's assume that through a variety of performance optimization techniques, we could cut the site outages in half, and reduce page load time by 2 seconds.

There are two components:

1. First, the impact of reducing site outages: if 5% of the time the commerce site is largely unusable, that's a loss of $500,000 per year. **Reducing site outages by a factor of 2X would help reclaim $250,000 in revenue.**

2. There is also the impact of improving average performance from the current 8 seconds to 6 seconds. Various studies estimate the impact of improving performance by one second as somewhere in the 1.0%-10.0% range. Let's take a figure from the lower end of this estimate – a 1.5% revenue increase for each second gained in page load speed. **In our case, the 2 second gain would translate into a revenue increase of 3%, or $300,000.**

Company Metrics	Input
Annual revenue	$10,000,000
% of time with poor web performance	5%
Impact of Performance Improvements	
Reduction of requests	50%
Revenue Increase due to improved reliability ($/year)	$250,000
User Experience performance improvement (sec)	2.0
Impact of 1 sec improvement on conversions (%)	1.5%
Revenue increase due to improved performance	$300,000
Total	$550,000

The total gain for the site would thus be $550,000 – clearly a sizable improvement for a modest amount of work. **WPO impacts the bottom line, even for small or medium sized businesses!**

How to Measure Performance

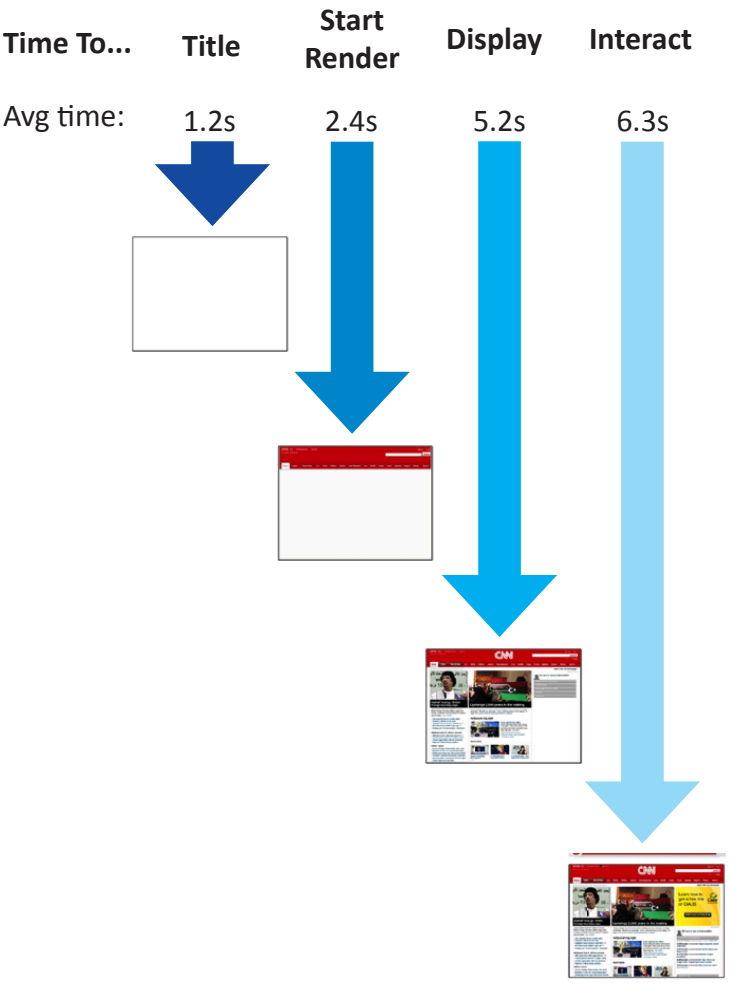
The first step in WPO is to learn more about your website's performance. A variety of website testing and monitoring tools have been developed to do this. Some operate online, through your web browser, and below are a few of our favorites from that category. Though these will not monitor your site continuously over time, they'll help you kick off the process with a baseline of information on your site's performance.

- Websitetest.com
- Webpagetest.org
- YSlow
- Mobitest (for mobile)

You can also check out this page on WPO guru Steve Souders' site for a more complete list of available tools for testing and monitoring web performance.

To learn about your website, a good place to start is from the user's perspective. This means looking at a combination of **front-end metrics** and **content metrics.** Front-end metrics show what users experience on your site, and content metrics help you understand the attributes of your site and the keys to improving it.

In the next two sections we will provide some basics to help you conceptualize web performance optimization. When you're ready to dive deeper into performance metrics, check out our eBook: 17 Performance Metrics You Should Care About.



Time to Title is the time elapsed from the moment a visitor's browser requests your site to the moment that your site's title appears in the tab of his or her browser.

Time to Start Render is the time elapsed from the request to when the visitor sees actual website content appear on the page. This stage is important since it assures the visitor that your site is loading. Nobody likes staring at a blank page. Assuring visitors that they are in the right place and will soon be seeing the content they expect will promote a good perception of your website.

Time to Display is the time elapsed from the request to when the browser has finished parsing the HTML page, constructed the Document Object Model (DOM), and displayed the HTML document. This all means that the page will have the general appearance of a web page, but there may yet be some images, interactive elements, and other media that haven't fully loaded.

Time to Interact is the time elapsed from the request to the moment the user can interact with the page. (By "interact" we mean the page will respond properly to the visitor clicking a link, scrolling, typing into a field, or activating an element like a hover effect). This does **not** mean that the page is fully loaded, as there may be scripts, trackers, and other assets that continue to load in the background. But it does mean that the visitor can use the web page, and that's most important. Many site owners choose Time To Interact as the principle index for overall web performance because of its relationship with user experience.

Weight

Typical asset weights	
JavaScript 116.6 KB	**Overall** 675 KB
Image 311.3 KB	**CSS** 17.7 KB

The overall weight of your website -- that is, the total number of bytes of all its assets -- factors into its speed. So does the weight of individual assets: **a single heavy asset can have a negative ripple effect on performance**.

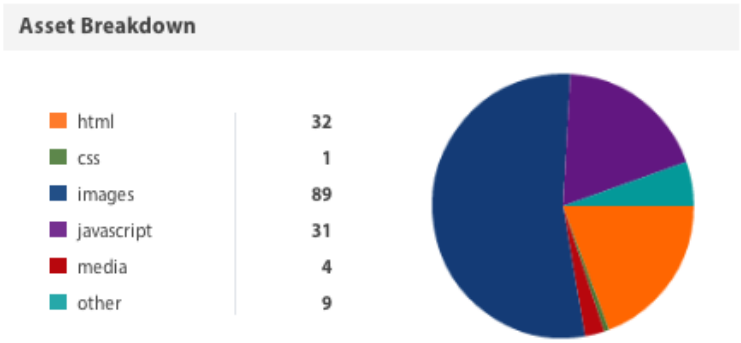
Use weight metrics to identify categories of assets that are too heavy in aggregate, then use a waterfall chart to zero in on specific assets within that category that can be fixed or cut.

For instance, if the total weight of your images is much higher than is typical (see left), click to the waterfall chart in your monitoring service and pay particular attention to images that have long load times. This will give you a starting point for deciding what images to compress or delete.

Asset Count

Typical asset counts	
Overall 47	**JavaScript** 8
Image 25	**CSS** 3

More assets necessarily mean more weight -- that's reason enough to keep track of asset count. But in addition, each time a visitor's browser makes the trip to your origin server to fetch an asset for your site, it adds time to the page load. That means **no matter how small or compressed an asset is, it still contributes to slowing down your site** to some extent. Some types of assets can be combined so that multiple assets load with a single request rather than individually.

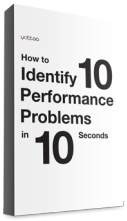


Example: pie graph, taken from the results of a WebsiteTest.com test of CNN.com, shows the variety of content that exists on a modern web page.

Basic Tools - Waterfall Chart

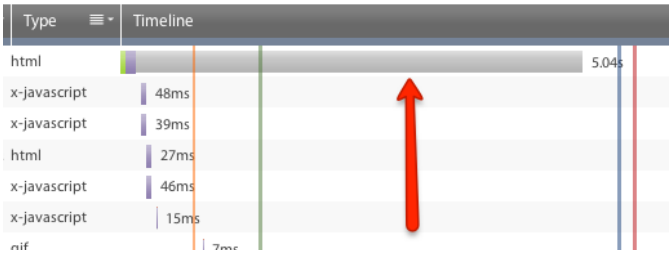
The **waterfall chart** is an indispensible tool for web performance. It visually displays each and every asset that composes a website, and uses horizontal bars to show how long each asset took to download. At the most basic level, a longer bar is a longer download time.

In addition, the other columns in the chart provide information on WHAT the asset is, WHERE the asset comes from (be it your origin server, or a third party), how HEAVY the asset is (in B, KB, MB), and the STATUS of the asset (if there was an error involved or not). With all this information you can quickly find out where the performance bottlenecks are.



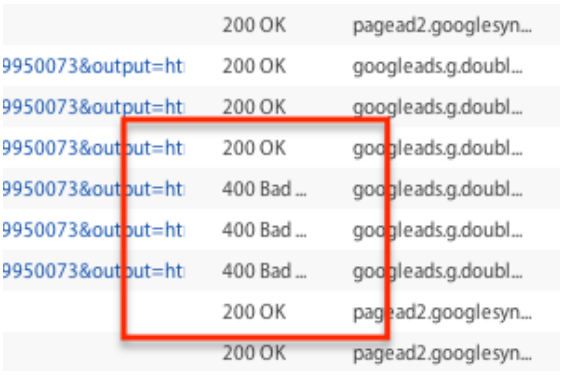
Below is an excerpt of a typical waterfall, and on the next page are several examples of how to identify website problems with the waterfall. For more tips on using the chart, check out our eBook: How To: Identify 10 Web Performance Problems in 10 Seconds

#	URL	Status	Host	Size	Type	Timeline
1	www.jamsadr.com	200 OK	www.jamsadr.com	14.7 KB	html	270ms
2	header.css	200 OK	www.jamsadr.com	20 B	css	56ms
3	general.css	200 OK	www.jamsadr.com	7.4 KB	css	59ms
4	menu.js	200 OK	www.jamsadr.com	1.7 KB	x-javascript	115ms
5	print.css	200 OK	www.jamsadr.com	1.4 KB	css	114ms
6	footer.css	200 OK	www.jamsadr.com	18 B	css	113ms
7	ntpagetag.js	200 OK	www.jamsadr.com	2.5 KB	x-javascript	62ms
8	spamproof.js	200 OK	www.jamsadr.com	786 B	x-javascript	59ms
9	JAMS-logo-blue-150x107.jpg	200 OK	www.jamsadr.com	14.3 KB	jpeg	61ms
10	logo_JAMS.gif	200 OK	www.jamsadr.com	2 KB	gif	69ms
11	tagline_the_resolution_experts.gif	200 OK	www.jamsadr.com	2.3 KB	gif	61ms
12	l_neutrals.gif	200 OK	www.jamsadr.com	716 B	gif	114ms
13	l_rules_clauses.gif	200 OK	www.jamsadr.com	1 KB	gif	59ms
14	l_locations.gif	200 OK	www.jamsadr.com	1.4 KB	gif	56ms
15	l_adr.gif	200 OK	www.jamsadr.com	1016 B	gif	59ms
16	l_practices.gif	200 OK	www.jamsadr.com	2.1 KB	gif	58ms
17	l_pubs.gif	200 OK	www.jamsadr.com	3 KB	gif	60ms
18	l_intern_4_07_2011_sb.gif	200 OK	www.jamsadr.com	4.6 KB	gif	55ms
19	ntpagetag.gif?js=1&ts=1358870279050.9	200 OK	www.jamsadr.com	85 B	gif	56ms
20	swfobject.js	200 OK	www.jamsadr.com	2.3 KB	x-javascript	59ms



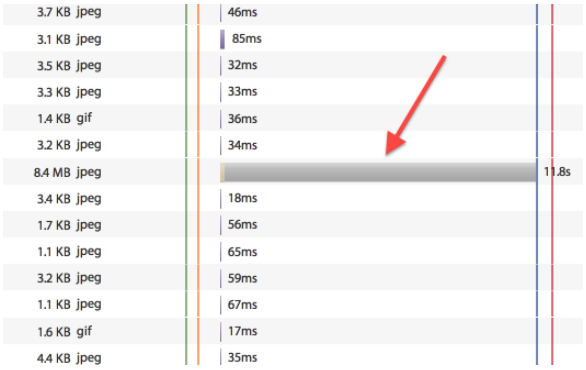
Slow Backend Performance

Look at the very first line in the waterfall chart. If it's longer than most of the other lines in the chart, that's not good. It should be very short. Improvements to web infrastructure in the last decade have made rapid content delivery the Internet standard. If your "time to last byte," aka total delivery time, is longer than 1 second, consider pursuing a plan to improve it.



Asset Errors

Scan the HTTP status column for any numbers 400 or over. Any time a 400- or 500-level error occurs it's a matter of concern since errors can often have ripple effects on performance, aside from the failure indicated for that one asset. Each case begs investigation.



One Bad Asset

Look for an extremely long bar in the waterfall. Any asset, be it a JPEG, JavaScript, or an asset served from a third party, can drag down your entire site. There are dozens of possible reasons why an otherwise innocent-looking asset could take seconds and seconds to load, but in the short term the important thing is identifying it and removing or fixing it so that the rest of your site doesn't suffer.

JavaScript Blocking Behavior

Ideally, the waterfall would be a flat diagonal line. Unfortunately, that's almost never the case. It's easy to see where the smooth pattern breaks: there's a left-to-right gap from one line to the next down. This gap indicates that an asset, usually a third party script, activated blocking behavior as it executed, which stalled other asset downloads. This behavior can be overcome by forcing that asset to load asynchronously.

How to Improve Performance

Bottleneck	Solution
Backend	• Adding/Upgrading servers • Adding bandwidth • Optimizing code/database
Middle Mile	• Content Delivery Network (CDN)
Front-end	• Reduce Number of Requests • Reduce Page Weight • Promote Serialization (Assets loading in parallel)

Once you find out what's dragging your site's performance down, there are several ways to approach improving the site.

If the bottleneck appears to be in the backend, there are two approaches: increasing the capacity of your servers by buying more or bigger ones, and adding bandwidth; and optimizing backend code. The latter includes optimizing the server side code (PHP, Java, Ruby, etc.) and optimizing the database (updating indices, queries).

If the problem is the middle mile, a CDN is the main solution. There's a wide range of CDN solutions, and they some considerations in picking a CDN are price (which ranges from hundreds per month to tens of thousands per month), whether the CDN provides whole site delivery, and whether the CDN serves dynamic content.

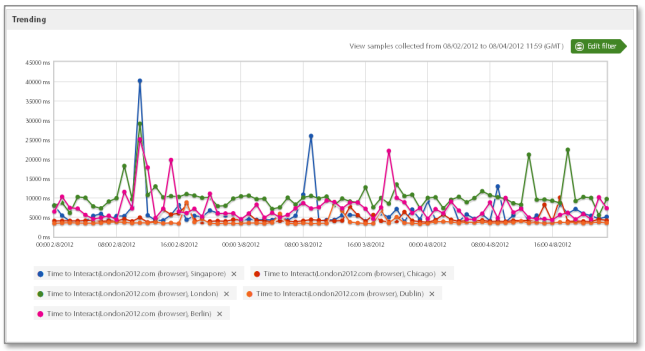
Most important is optimizing the front-end. This is by far the most common (and most devastating) bottleneck for web performance today. Fortunately, while server upgrades and CDNs have unavoidable cost, the front-end can be mitigated manually by developers. On the following page we explore some of these techniques.

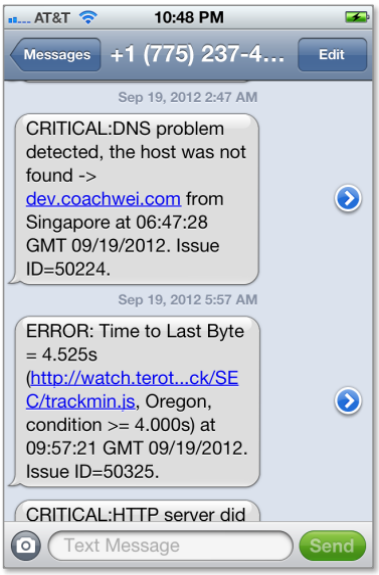
We explore these manual FEO techniques and others in greater detail in our eBook, 11 Techniques To Make Your Website Rock.

Methods for FEO: Issue -> Resolution

Issue	Resolution
Lots of Requests	**Reduce Number of Requests:** • **Combine Scripts -** Combining or "concatenating" files and scripts is accomplished by writing a line of code that encompasses information that was previously contained in two or more lines of code. This is a common practice that can be accomplished by a developer or using free script-based tools. • **Combine images with sprites -** A sprite is a single image that contains the visual information of multiple images. A CSS script then positions pieces of the image where they are supposed to be on the page. This allows the developer to make a single request for multiple images. • **Combine images etc. with "data: URIs" -** A data: URI is a method of including small images or other bits of data "inline" into an HTML file. That way, instead of loading as a standalone request, the image will be downloaded along with the HTML file that includes the basic framework of the page.
Large Assets	**Reduce Asset Weight:** • **Use "GZIP" Compression -** GZIP is a versatile script that can be applied to almost every asset on your website. As long as the script is called first, all files of that type (HTML, JS, CSS) on your site will be "gzipped," i.e. compressed and less heavy. • **Minify Scripts -** Script minification is the process of stripping out whitespace, comments, and erroneous characters from code. This makes the code more streamlined and less heavy. • **Use Lossy and Lossless image compression -** All images on a site should be compressed. Lossless compression can slightly reduce the weight of an image without reducing the display quality at all. Lossy image compression slightly reduces the display quality (though usually so little it's not noticeable to the visitor) but can reduce the weight of the image on the order of 50 to 90% depending on the image and degree of loss.
Serialization	**Parallel Processing:** • **Load 3rd party assets "asynchronously" -** Social media widgets, trackers, and other dynamic assets on your site are often hosted by third parties far and wide, on servers you have no control over. When those widgets have performance issues, their problems spill onto your site and drag down performance. Rewriting the script so that it loads asynchronously can help mitigate blocking behavior so that the rest of your site can load around the struggling asset. • **Use domain sharding -** Domain sharding is a way to overcome the limitation of browsers that only allow a handful of open connections with a server at a time. This has largely been solved by newer editions of browsers that allow for up to 12 open connections at a time - but not all web users are on updated browsers. To do this you rewite an asset's URL to include a made-up alias in the place of "www" that tricks the browser into thinking it's a different server.

Ongoing Action: Website Monitoring





In order to guarantee a return on the investment of time and effort you've made in web performance optimization, it's important to monitor website performance continuously over time. **The web is constantly shifting and changing, and your website's performance can change at any moment.**

There are several options for site performance monitoring including Pingdom, Yottaa Site Monitor, Keynote, and Gomez. Using these tools, set up your site to be monitored in real time and save data to show trends. With many monitoring systems you can also arrange alerts, so that you're notified instantly when any of a number of website problems that you've selected occur.

We recommend monitoring a sample of your site's pages. This would certainly include the home page. If your site is an eCommerce site, then product catalog pages, as well as mission-critical pages like a shopping cart check-out are important to monitor.

How frequently you monitor depends on how much traffic your site receives. Some site owners find it adequate to measure the site every 3 to 4 hours, knowing that only a few customers will be affected. Others measure pages multiple times an hour from a variety of locations.

To get a feel for monitoring before diving in, sign up for a free Yottaa Site Monitor account.

WPO Checklist + Resources

WPO Checklist:

- [x] Read the Beginner's Guide to Web Performance
- [ ] Test web performance with a free tool like Yottaa Site Monitor to identify areas for improvement
- [ ] Learn more about techniques for optimization; determine whether to perform manual optimization or enlist an automatic WPO service
- [ ] Review options for a Content Delivery Network (CDN) if delivery is slow
- [ ] Review and choose a monitoring service to gather performance data over time
- [ ] Follow the guidelines in our eBook on managing a WPO project and put all that research to work!

Yottaa blogs and eBooks:

Manual FEO:
11 Techniques To Make Your Website Rock

Entry-level analysis of performance tests:
10 Performance Problems Identified in 10 Seconds

Intermediate-level analysis of performance tests and monitoring data:
17 Performance Metrics You Should Care About

WPO project management:
Managing a Web Performance Optimization Project

Other Resources:

SteveSouders.com - Steve's blog, as well as information on his books

Yahoo Performance Page - Tips, tools, videos, a forum and more, all dedicated to WPO

Book of Speed - Stoyan Stefanov shares his expertise in this work-in-progress online book

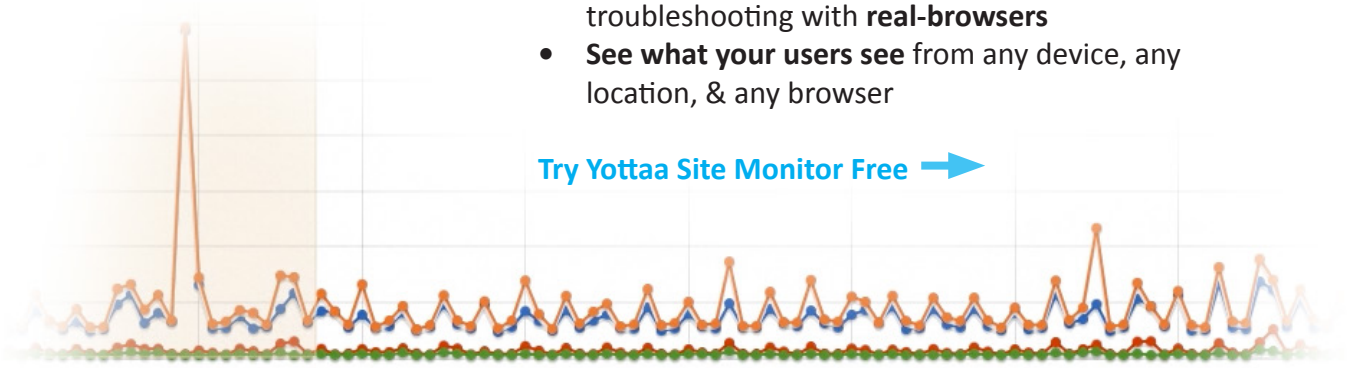
Velocity Conference Resources - Annual conference for the WPO industry

About Yottaa

Yottaa is an easy-to-use cloud service that optimizes, protects, and monitors any website.

Try it Free

Yottaa Site Monitor



- **24/7 Site Monitoring** is always on alert & ready to help
- Do multivariate testing & web performance troubleshooting with **real-browsers**
- **See what your users see** from any device, any location, & any browser

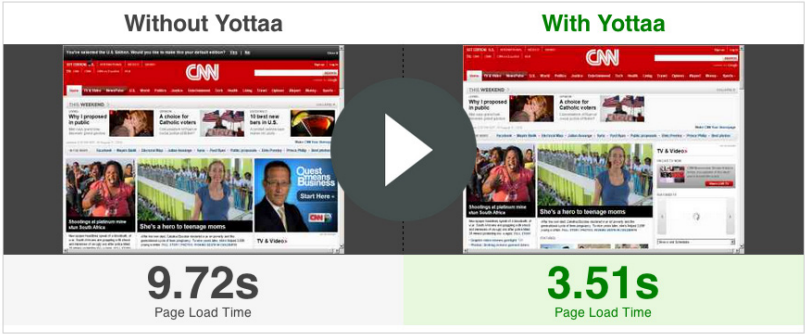
Try Yottaa Site Monitor Free →

Try it Free for 7 Days

Yottaa Site Optimizer

- On-the-fly **front-end optimization techniques** streamline content
- Yottaa's **hybrid-cloud network** provides infinite scalability
- Global **CDN ensures** great user experience from any location

Try Yottaa Site Optimizer Free →



Like this eBook? Share it!