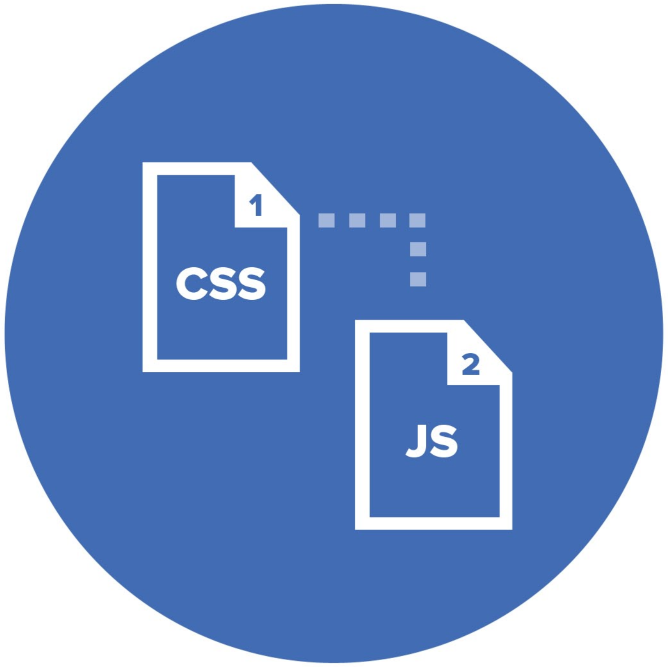


A PUBLICATION FROM yottaa

A STEP-BY-STEP GUIDE TO WEBSITE OPTIMIZATION

HOW TO OPTIMIZE ORDER OF EXECUTION

1

Contents

Table of Contents

Code Cleanup

All code is not created equal.

Two web pages that appear identical to a visitor may be underpinned by vastly different code, and the makeup of that code can mean the difference between quick, easy site maintenance and a developer's nightmare. That's why best practices were created to help developers write code that is **tidy, efficient, organized, and beautiful.**

One niche of coding best practices revolves around cleaning up code for the purpose of web performance optimization (WPO). This means making code lighter and streamlined for fewer total requests. Developers have used WPO best practices for years, but presently they are more important than ever. The main bottleneck slowing down page performance has shifted from the delivery stage to the front-end stage, where the browser downloads and renders the page. Front-end developers now have more power than ever to make their sites fast, and much of it comes down to simple code maintenance!

We've created a series of eBooks that include detailed how-tos of "Code Cleanup" best practices that are guaranteed to make pages load faster. This eBook concentrates on order of execution. The other two address code minification and script concatenation.

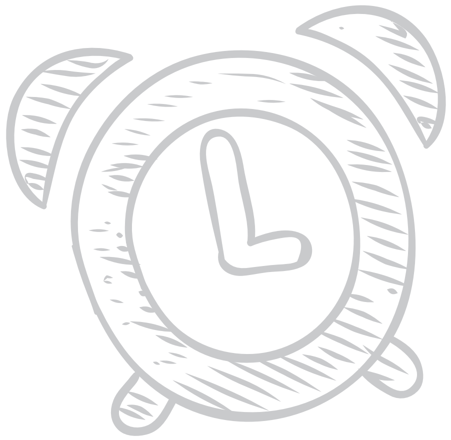
Order In The Court!

Optimizing order of execution for assets on a web page is one of the best ways to ensure a good user experience.

Here, “good user experience” isn’t just a stand-in term for “faster page load time” (as is often the case in web performance discussions) – it also indicates how users *perceive* load time.

Users like to see progress – reassurance that the page content is on the way. Giving a user some content to look at while the page loads not only inspires confidence in the site’s execution, but leads users to believe the page loads faster than it does.

However, ordering your assets incorrectly can cause a blank screen in place of a progressive render, indicating system errors (instead of coding mistakes) and eliminating progress indicators. Avoiding the “blank white screen of death”* can be achieved by optimizing the order that the assets in your site execute.

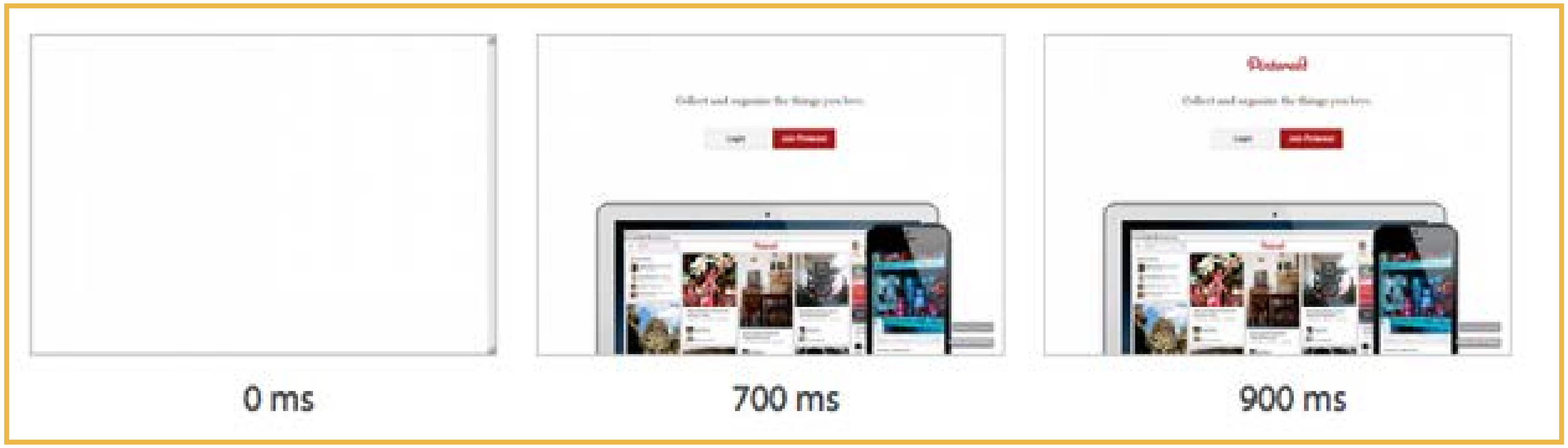


*Unofficial (but often used) term

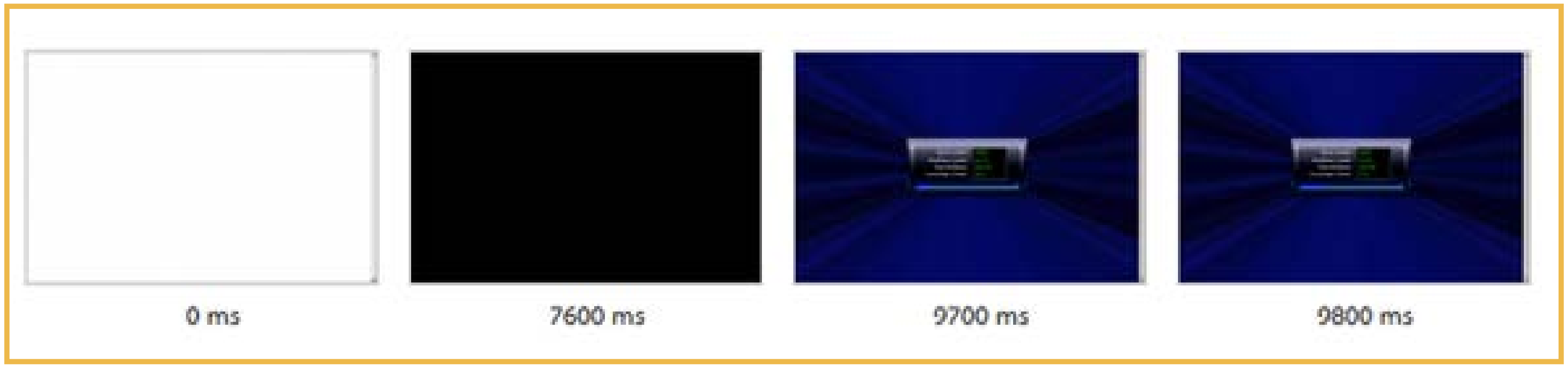
Why does it matter?

yottaa

Winning hearts and minds with a quick load time and important content fetched first.



Losing their faith and attention with empty screens.



Ok, so how do I do it?

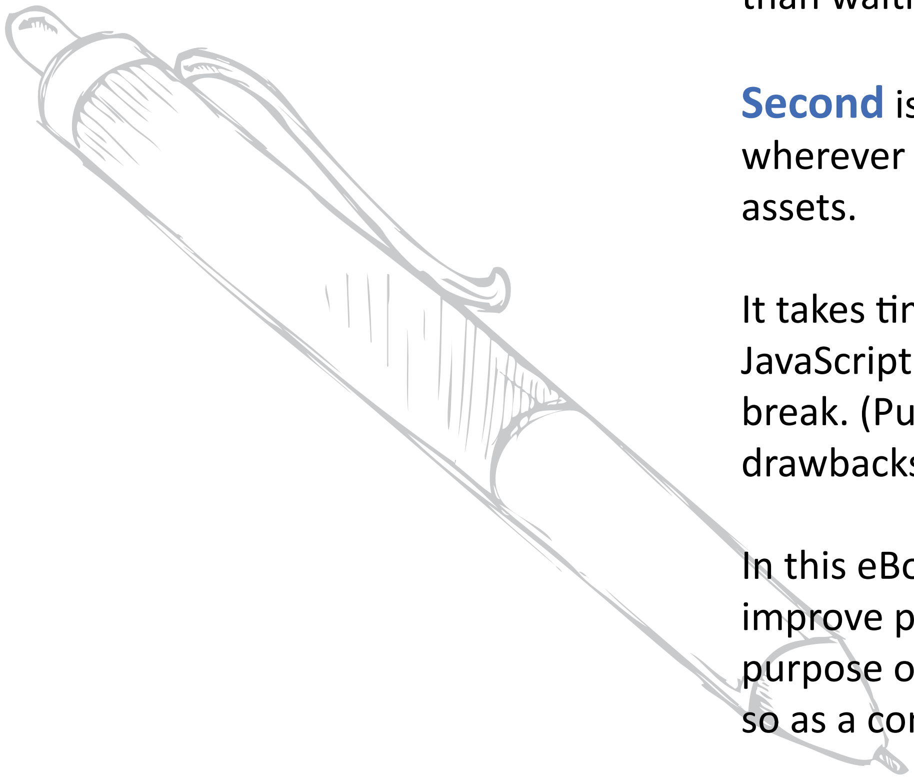
Improving the order of execution comes down to two main points.

First is putting CSS files at the top of the load order so that content like images and text can be rendered as soon as it's ready, rather than waiting for a CSS file to download (we'll explain).

Second is putting JavaScript (JS) at the bottom of the order wherever possible to avoid scripts blocking the download of other assets.

It takes time to find out what works on your site. The order of JavaScript execution may have to be preserved, lest something break. (Putting CSS at the top of the order has no notable drawbacks. Just do it.)

In this eBook we've also included two other techniques in that will improve performance. These techniques don't have the express purpose of changing the order of execution, but they end up doing so as a consequence of what they do to change the code.



Tip #1: JavaScript

Put JavaScript files at the bottom

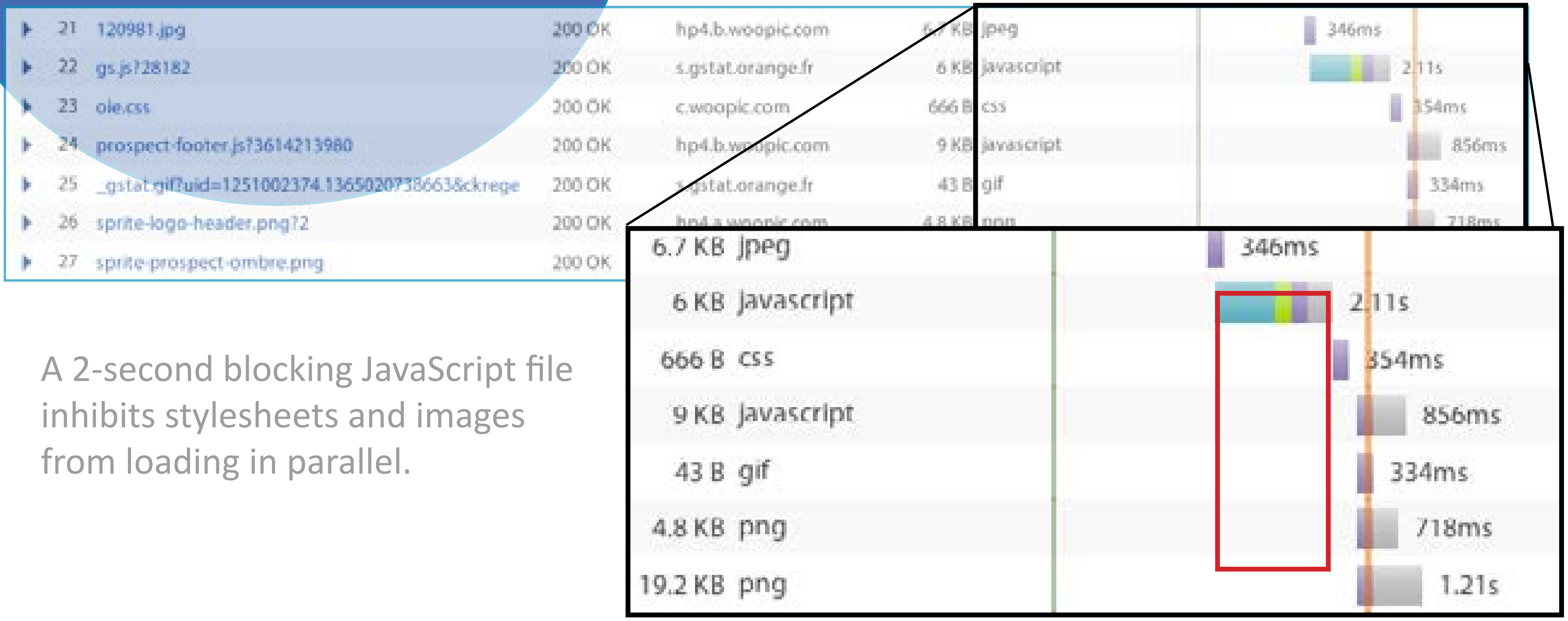
JavaScript files can block subsequent file downloads, like images and stylesheets, from downloading in parallel. This means the page load process is halted until a JS file has finished rendering, which has ripple effects on performance.

Many times JS files are related to content that's not important to the main messages of the page. For instance, a Facebook widget in the sidebar might be blocking a main navigation element on the page from downloading. That's a problem for user experience.

Moving JS files to the bottom of the code allows other assets to load initially, giving the visitor something to see and interact with on the website before it's completely loaded.

Even if the overall page load time stays the same, the site can reach a point where the user can start to view and use much of the site's content, unbothered by the fact that some JS assets are still loading.

yottaa



21	120981.jpg	200 OK	hp4.b.woopic.com	6.7 KB	jpeg	346ms
22	gs.js?28182	200 OK	s.gstat.orange.fr	6 KB	javascript	2.11s
23	oie.css	200 OK	c.woopic.com	666 B	css	354ms
24	prospect-footer.js?3614213980	200 OK	hp4.b.woopic.com	9 KB	javascript	856ms
25	_gstat.gif?uid=1251002374.13650207386638&ckrege	200 OK	s.gstat.orange.fr	43 B	gif	334ms
26	sprite-logo-header.png?2	200 OK	hp4.a.woopic.com	4.8 KB	png	718ms
27	sprite-prospect-ombre.png	200 OK		19.2 KB	png	1.21s

A 2-second blocking JavaScript file inhibits stylesheets and images from loading in parallel.

0 ms

8300 ms

This contributes to a long start render time, and a blank screen nightmare!

Tip #1: JavaScript

Challenges in moving the JS

Most times, it's simple enough to move the JavaScript as far down in the page code as possible, ideally right before the final `<body>` tag. With your CSS at the top (tip #2 - read on!), it's easy to identify where the other assets end.

Watch out for any script that uses `document.write` - this tells the page to insert part of the content at a specific point, and thus can't be moved to the bottom. Instead, seek out an alternative option, like Ajax style calls to change the contents of named elements, which can be added as a separate script.

Make sure any external script files are also placed in the bottom; even though they're external, they can still block parallel downloading and prevent progressive rendering.

With JavaScript, it's always important to pinpoint any tags that might become problematic when moved to the bottom, so be vigilant in your review.

Tip #2: CSS

Put stylesheets at the top

Some browsers automatically prevent content from rendering until all CSS files have downloaded. This results in a blank screen if even a single, unrelated CSS file happens to be further down in the order of execution.

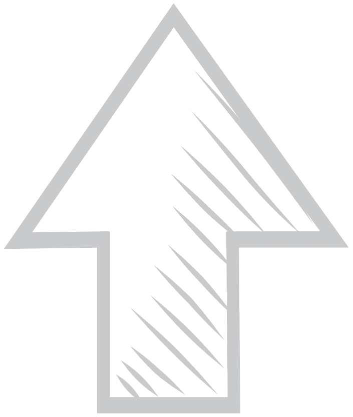
Other browsers allow content to render, but this can lead to a wonky rendering process when CSS files lower in the order introduce new styles that cause the already-visible content to change suddenly. That defeats the purpose of progressive rendering, as it disorients the visitor.

Putting CSS at the top of the order of execution can solve both of these unfortunate circumstances.

This is accomplished by putting the CSS files right into the HEAD section of the document via a LINK tag. This will allow content to display as soon as it’s downloaded, and with the proper appearance

Tip #2: CSS

```
<head>
<link rel=”stylesheet” href=”YOTTAASTYLE.css”>
</head>
```



Putting CSS in the HEAD section is the easiest and most straightforward way to move your stylesheets to the top.

For both performance and simplicity, this is the ideal.

Tip #3: Inline

Inline images with data URIs

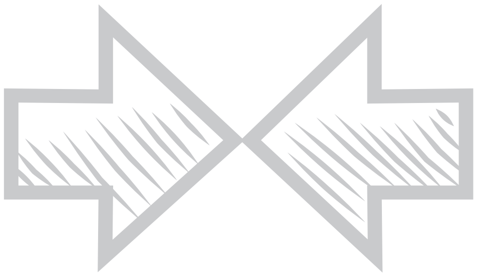
One way to cut down requests is to use data URIs to “inline” files. On a very high level, this means that instead of a file being fetched via its own URL — which requires an HTTP request — it is fetched along with a larger HTML or CSS stylesheet file that it’s been “inlined” into. To do this, you can create a data URI.

In a data URI, the information required to display an image or text exists in Base64 encoding or URL-encoding, within the document it’s inlined into. This means it can be accessed immediately when that document is fetched.

This also means that the inlined images use the data URI to embed the images directly within the website as “immediate” data, creating fewer external reference requests. It is essentially retrieved and read as any other part of the page text.

Tip #3: Inline

```
data:[<mime type>][;charset=<charset>][;base64],<encoded data>
```



To create a data URI one must follow the basic format above.

Larger images will not work as well with URI schemes; they may weigh down the HTML document, causing “bloat.” They will also benefit more directly from browser caching when they exist as a standalone request.

Tip #4: Post-load

Post load components

Post loading components allows you to prioritize the most important assets on your site and put a hold on the rest until after the initial load. This is most often done with JS by delaying which files are least important to the user experience.

For example, you can delay things that live below the fold, so the user sees a quick loading time on the top of the page and may not even notice a post load that happens before they scroll down.

Special note: If You Have a Responsive Site

Responsive images are designed to manipulate order of execution – once you configure a page to support responsive images, look at the page to make sure only visible images load according to what browser size the visitor is on.

Tools to Use

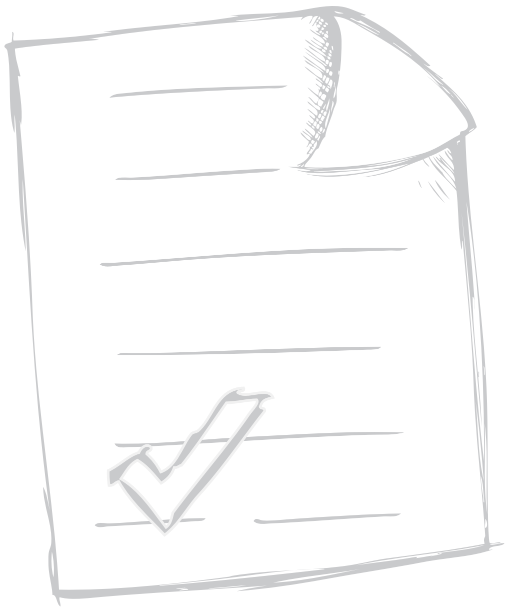
There are a few tools available to help in these techniques.

- YUI3 ImageLoader allows you to delay the loading of images on your web page until your user is likely to see them

- Similarly, YU3 Get gives you the ability to dynamically load JS and CSS resources

- Google's Closure Compiler is a JS optimization tool that includes inlining in its process

There are also free tools you can use to assess and pinpoint which assets are causing additional delays, and you can re-order accordingly. Websitetest.com and webpagetest.org are two common testing tools available that we use often.



Or You Can Automate It

With Yottaa Site Optimizer

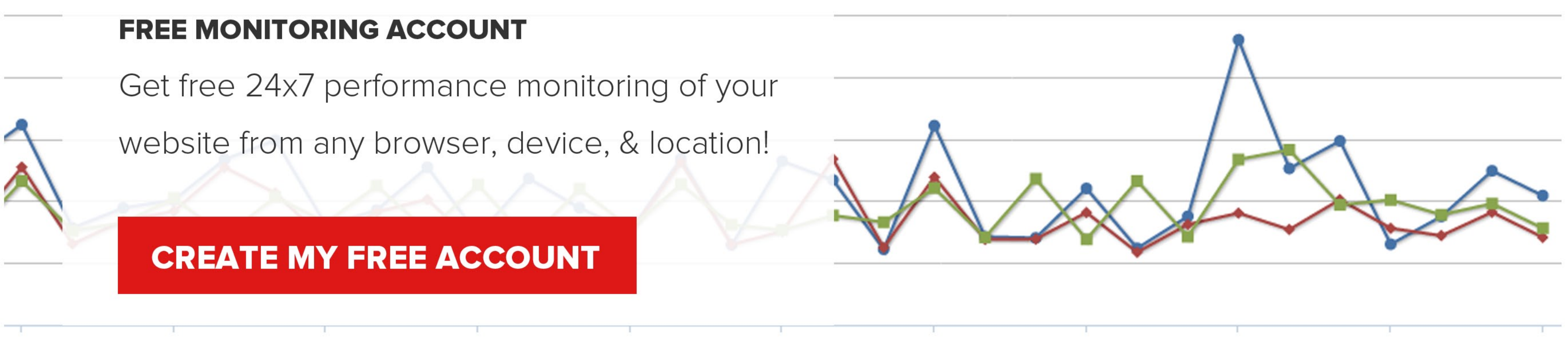
Yottaa's Site Optimizer software can automatically optimize your website assets for priority ordering and to decrease page load time.

Part of Yottaa's all-in-one web optimization solution, Site Optimizer automatically accelerates web sites and mobile apps, reducing page load times and improving your visitors' ability to interact with every page on your site.

Learn more about Site Optimizer by visiting our website or sign up for a free 7 day trial to see how Optimizer accelerates your website.

Want to learn more about website performance optimization?

Here are some of our awesome free resources for you.



FREE MONITORING ACCOUNT

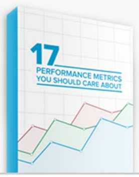
Get free 24x7 performance monitoring of your website from any browser, device, & location!

CREATE MY FREE ACCOUNT

FREE EBOOK

17 Performance Metrics You Should Care About

Download →



FREE EBOOK

Managing A WPO Project: A Step-by-Step Guide

Download →

LATEST FROM THE BLOG

The ROI of Website Performance: A Resources Guide

Read Post →