A PUBLICATION FROM yottaa

A *Designer's* GUIDE TO WEB PERFORMANCE

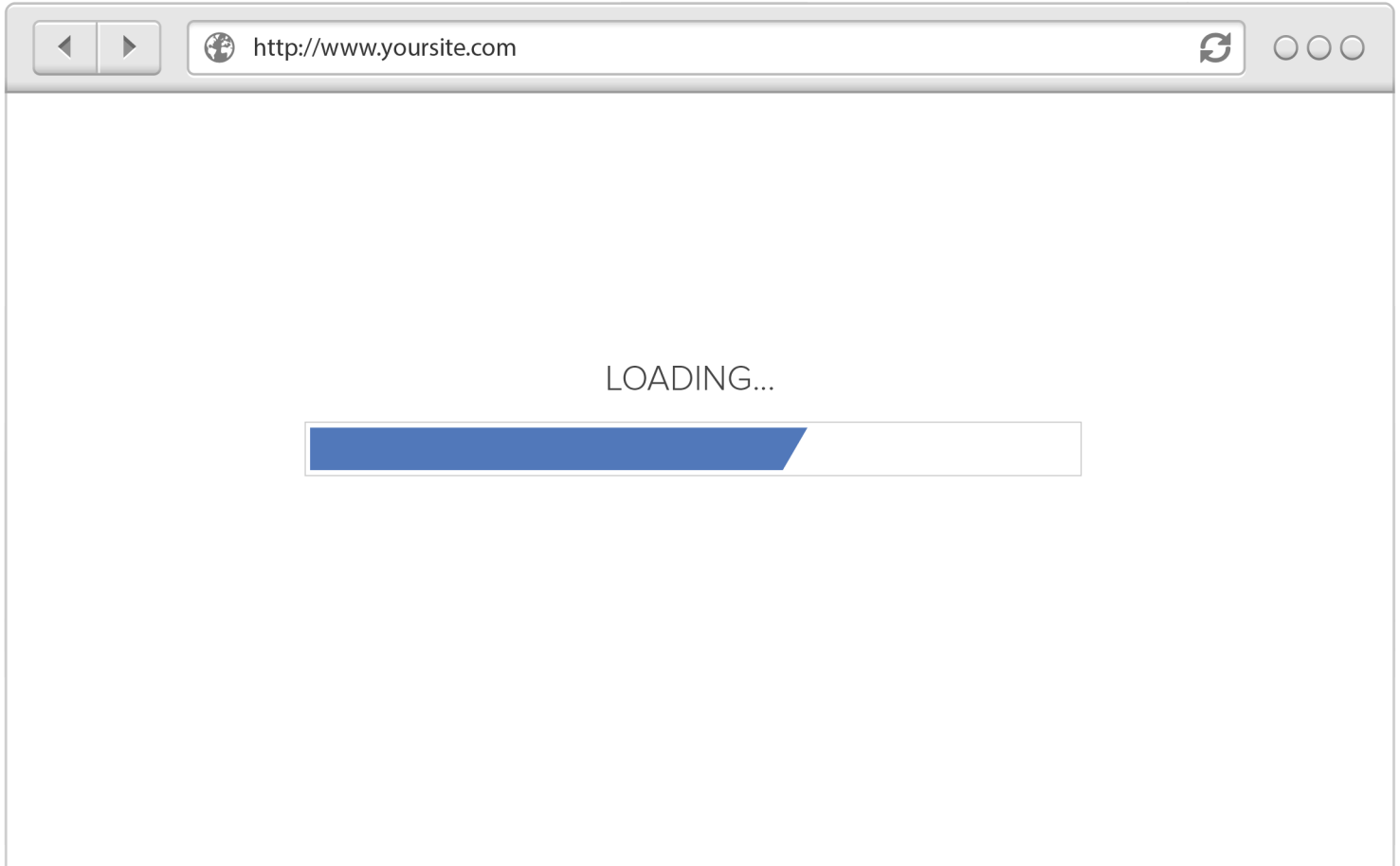


TABLE OF CONTENTS

Chapter 1

Why Web Performance **MATTERS FOR DESIGNERS**

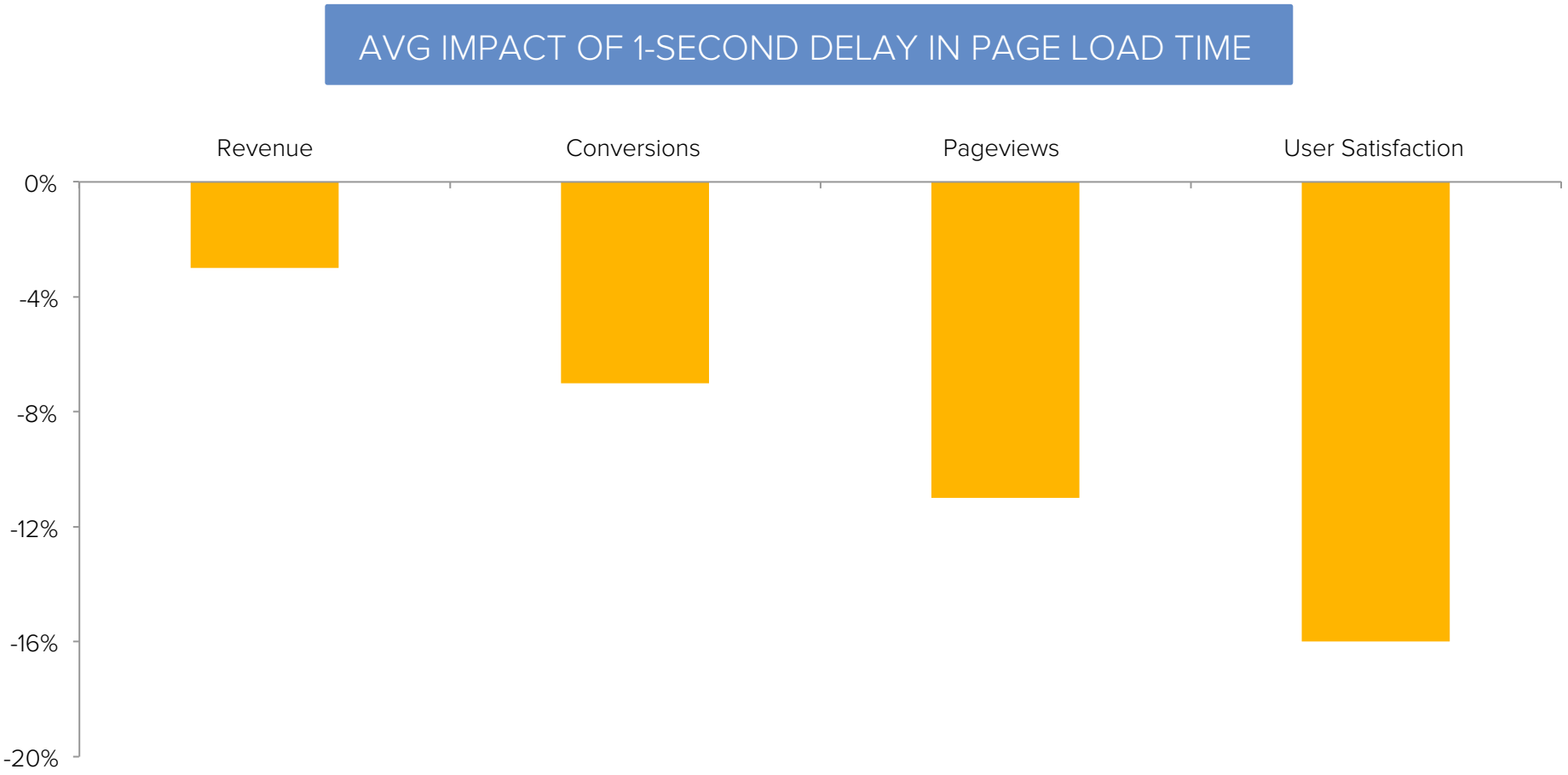
Speed is one of the most overlooked aspects when it comes to websites and web apps. Today, users expect a website to load in 2 seconds or less.

In fact, a 1-second delay in page load time equals a 3% drop in revenue per visitor, 7% fewer conversions, and a 16% decrease in customer satisfaction. When it comes to web performance, every millisecond counts.

> Speed is the *most important* feature. If your application is slow, people won't use it.
>
> ***Fred Wilson**, Managing Partner @ Union Square Ventures*

So, why should designers care? Because 80%-90% of performance occurs on the front-end of websites, which is where designers operate. Only 10%-20% happens via back-end performance, which falls under the responsibility of developers and operations. In other words, all of the content a designer is responsible for creating and building - HTML, CSS, JavaScript, images, etc. - accounts for almost all of a page's load time.

A faster website or web app will produce better business results and a better user experience. It will significantly increase visitors, conversions, average order size, and revenue both in the short and long-term. There are endless case studies proving the importance for web performance.



CASE STUDY #1

100ms = **1%**

DECREASE IN PAGE LOAD TIME

DROP IN TOTAL REVENUE

Amazon found that a 100ms delay in page load time caused a **1% drop in total revenue** - that's millions of dollars for them!

CASE STUDY #2

12%

INCREASE IN REVENUE

\+

25%

INCREASE IN PAGEVIEWS

Shopzilla sped up its average page load time from 6 seconds to 1.2 seconds and experienced a **12% increase in revenue** and a **25% increase in pageviews**.

CASE STUDY #3

mozilla Firefox®

15.4% INCREASE IN CONVERSIONS

=

60 million MORE DOWNLOADS PER YEAR

Mozilla shaved 2.2 seconds off their landing pages, increasing their download conversions by 15.4%, which will result in **60 million more Firefox downloads per year**.

Chapter 2

Why Designers Struggle to **BUILD FAST WEBSITES**

There are several key reasons why building a fast website or web app has become so challenging for designers.

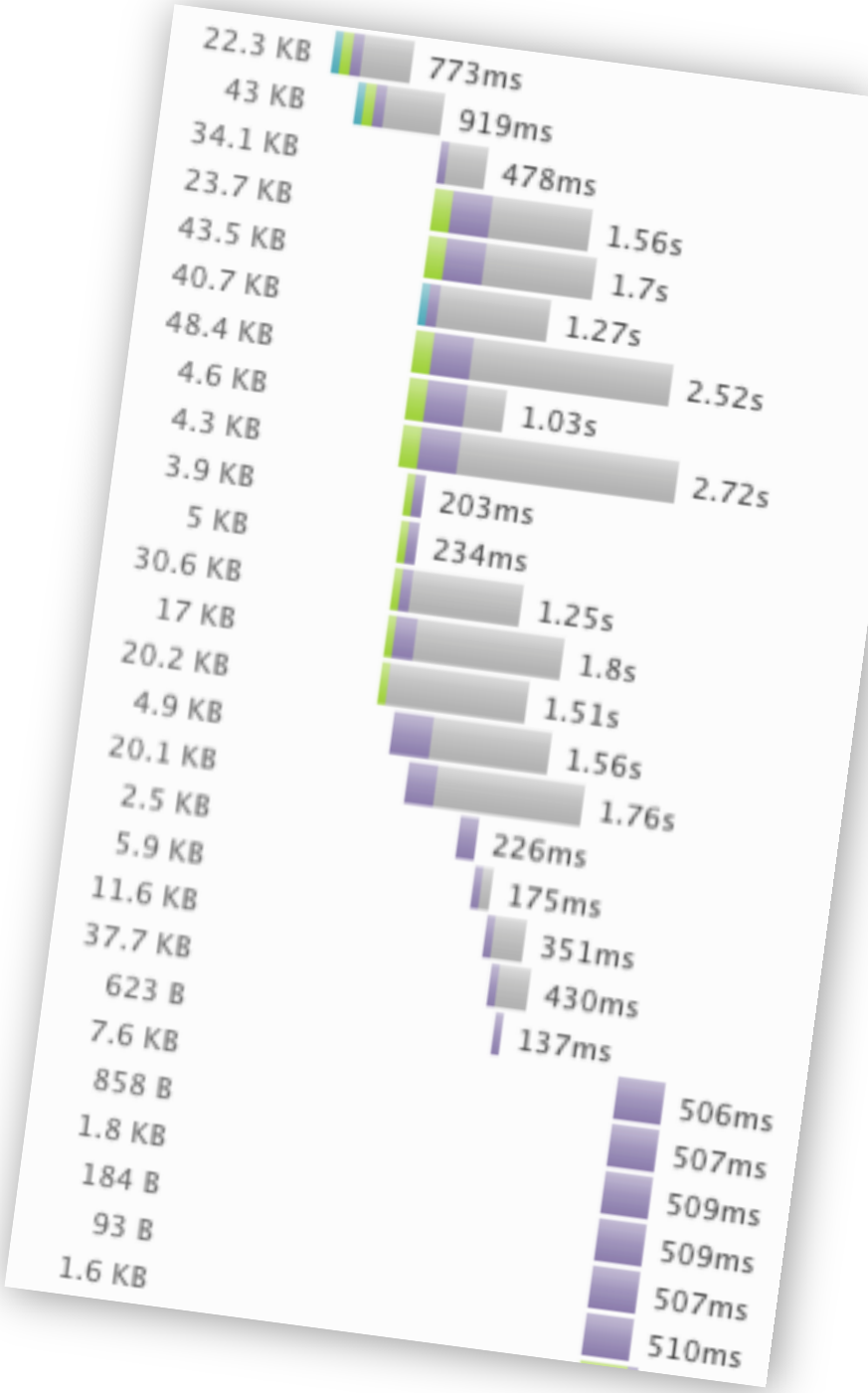
First, creating and building content by nature decreases performance. The more content a webpage has, the worse it will perform. As Steve Souders, legendary web performance optimization guru, once said, "The fastest HTTP request is the one not made."

> The *fastest* HTTP request is the one not made.
>
> ***Steve Souders**, Head Performance Engineer @ Google*

So, the best way to optimize web performance, in theory, is to have no content at all. But, users now expect websites to contain more content, media, and interactivity than ever before. Striking a balance of fast performance and content-rich sites is one of the biggest challenges to running a great website today.

Second, many designers are unaware of the performance consequences that their assets create.

They lack the visibility and insight into their web performance issues - for example, knowing that a specific JavaScript file is slowing down their page load time by 3 seconds. They don't have or use the tools necessary to assess their web performance. Additionally, many designers do not know what steps they must take to fix these performance problems - such as moving that specific piece of JavaScript to the bottom of their HTML code to prevent it from blocking the rest of the page downloading.



Lastly, without the help of a third-party service, the methods and techniques needed to optimize a website are very manual and time-consuming efforts.

With tight deadlines and limited cycles, many designers do not take the time necessary or make it a priority to build a high performance website. Maintaining fast and consistent performance is a highly complex and demanding job. As a reference, Google, who is renowned for their fast performance and user experience, has a team of over 100 engineers whose sole job is focused on web performance optimization!

Chapter 3

Understanding Key Web **PERFORMANCE METRICS**

There are 3 key metric areas for web performance: back-end performance, front-end performance, and content complexity.

Each key area contains several metrics that will provide actionable insight to improve the web performance and page load times of websites.

BACK-END PERFORMANCE METRICS

Back-end performance refers to the connection and delivery of a webpage's content. This accounts for about 10%-20% of web performance and is not visible to an end user.

DNS Resolution Time

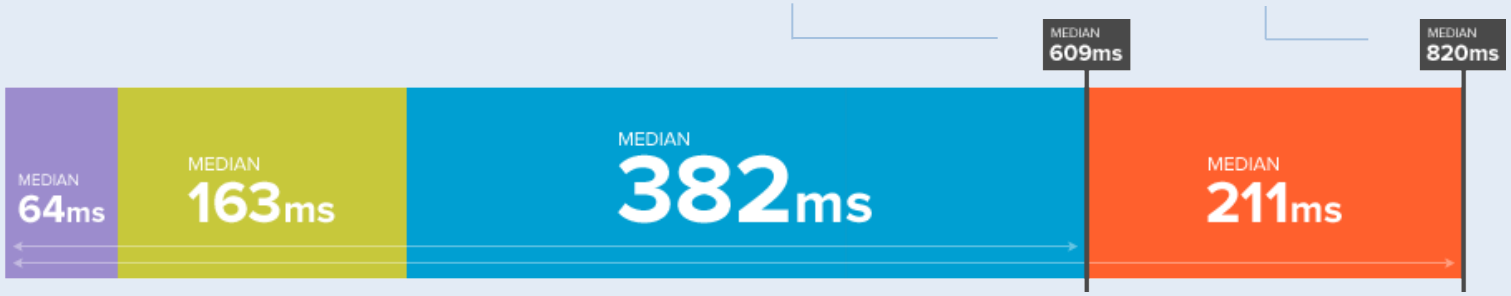
Time elapsed for a DNS provider to execute its service - to process a visitor's URL request and return the matching IP address.

Time to First Byte

Time elapsed for the first byte of a website to make it to the visitor's browser.

Time to Last Byte

Time elapsed when every byte of a website has made it to the visitor's browser.



Time to Connect

Time elapsed from initial request to when the connection between the visitor's browser and an origin server is established.

Waiting Time

Time elapsed from establishing a connection to delivering the first byte of a webpage.

Receiving Time

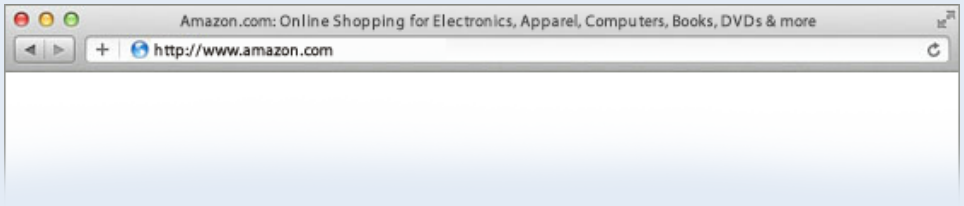
Time elapsed from downloading the first byte to the last byte of a webpage.

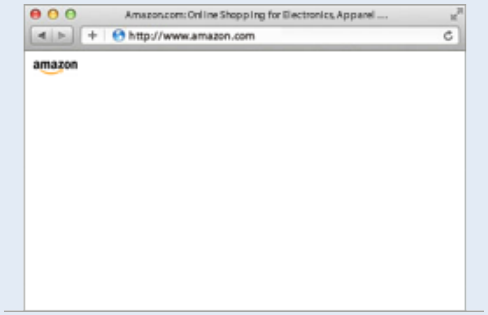
FRONT-END PERFORMANCE METRICS

Front-end performance refers to how quickly a visitor's browser executes and renders a webpage's content. They are each measured from the moment a request for the site is made - that is, when the visitor clicks a link or enters a URL.

Time to Title

Time elapsed when a user's browser downloads the first byte of your website and the webpage's title displays in the browser.





Time to Start Render

Time elapsed when the first visible element appears on the blank page.

Time to Display

Time elapsed when all visual elements of the page are in place.

Time to Interact

Time elapsed when a user has gained control of a webpage and can interact with content.

CONTENT COMPLEXITY METRICS

Content complexity refers to how complex a website's composition is in terms of the number and the size of requests, assets, and files. The more requests and the heavier a website's assets are, the more complex the website is, and vice versa.

Total # of Assets

Requests

Domains

HTML

JavaScript

CSS

Images

Media (Video, Audio, etc.)

Other

Total Size (KB)

Page Size (Sum of All Assets' Sizes)

HTML

JavaScript

CSS

Images

Media (Video, Audio, etc.)

Other

Chapter 4

What Causes Poor **WEB PERFORMANCE?**

What are the main culprits of poor performance? What components on the front-end cause the biggest problems?

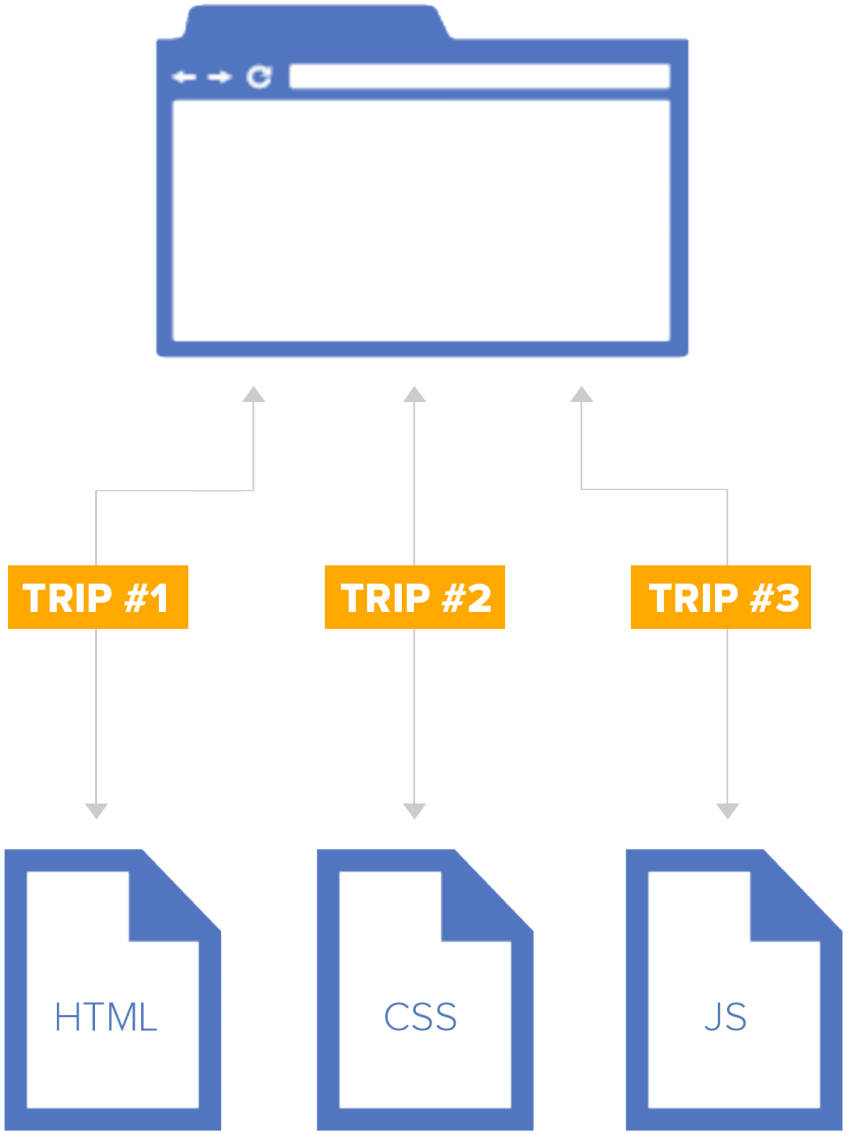
The key area we are focusing on improving performance in this eBook is a website's content complexity. Designers should mainly be concerned with optimizing assets such as HTML, CSS, JavaScript, and image files.

> If your site is 15MB, it's not HTML5 – it's *stupid*.
>
> ***Christian Heilmann**, Principal Developer Evangelist @ Mozilla*

As we previously alluded to, a high number of files or assets is the biggest indicator of poor performance. Reducing the number of round trips and requests the browser needs to take to download a webpage will make the biggest impact in improving web performance.

Picture yourself going to get a bunch of mail from your mailbox.

In this example, you are like the browser, having to make trips to retrieve all of the assets needed to download your webpage. Would it make sense to make a separate trip to get every single piece of mail in your mailbox? No. But that's how a browser is programmed to behave when it downloads assets of a webpage – it makes a separate trip for every individual file. Granted, the browser does this very quickly - often, in milliseconds. But the more individual files there are, the longer it will take the browser to process all of those requests.



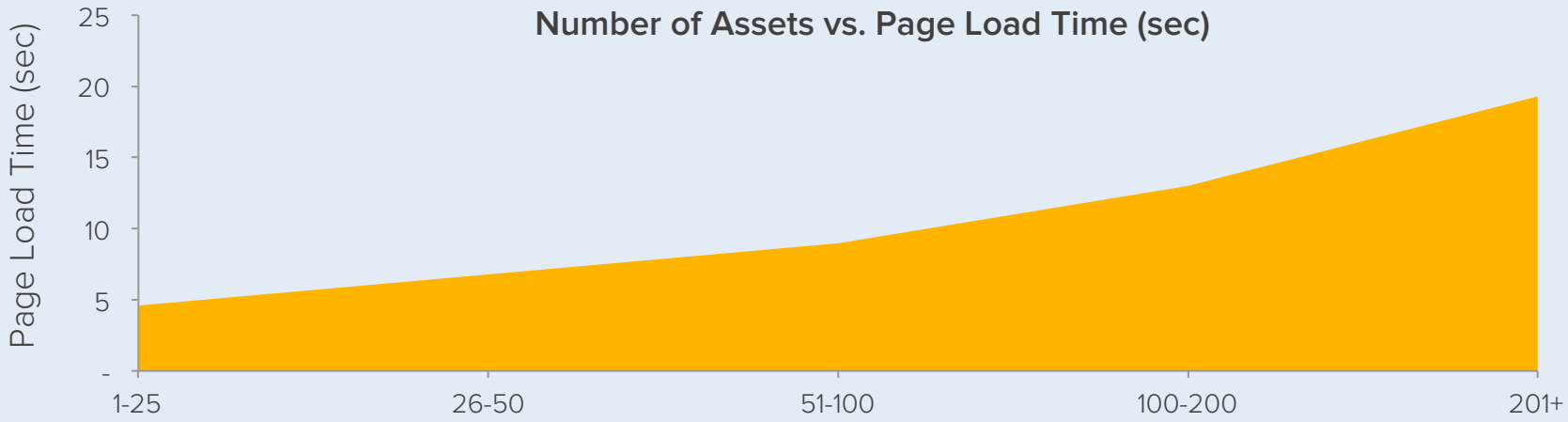
OF ASSETS BREAKDOWN

59

AVG # OF ASSETS

AVG # OF IMAGES	**33**
AVG # OF JAVASCRIPT FILES	**11**
AVG # OF CSS FILES	**5**
AVG # OF OTHER FILES (VIDEO, AUDIO, ETC.)	**10**

When optimizing for performance, designers should reduce the number of files above all else as the number one priority. The key method to reduce the total number of assets on a webpage is combining files.



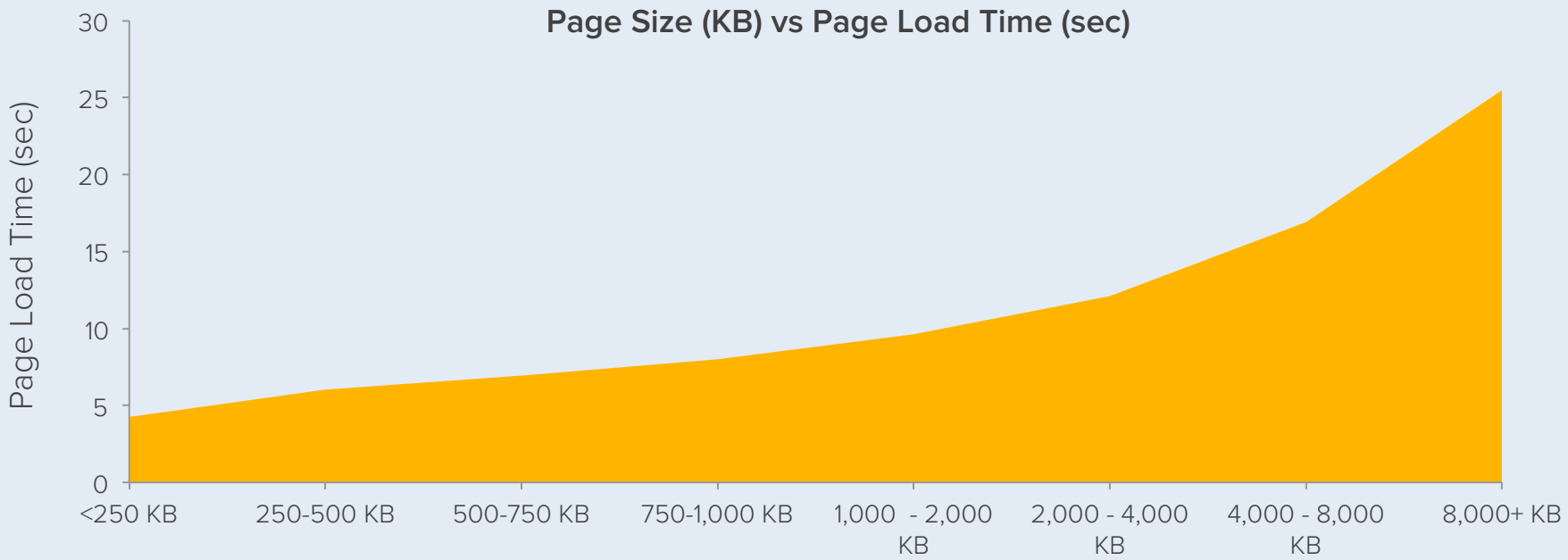
Additionally, large page size is another key indicator of poor performance. Bigger files take more time for browsers to download. Let's return to the mailbox analogy we just used. It's easy to carry a few envelopes of mail. But what if there were several heavy boxes and packages? All of a sudden, you won't be able to carry your mail as quickly or easily. The same holds true for browsers. If you have massive files on your website, browsers will have a tougher time downloading and delivering them to end users. So, designers need to compress and reduce file size as much as possible to achieve the fastest page load time possible.

ASSET SIZE BREAKDOWN

1.1MB

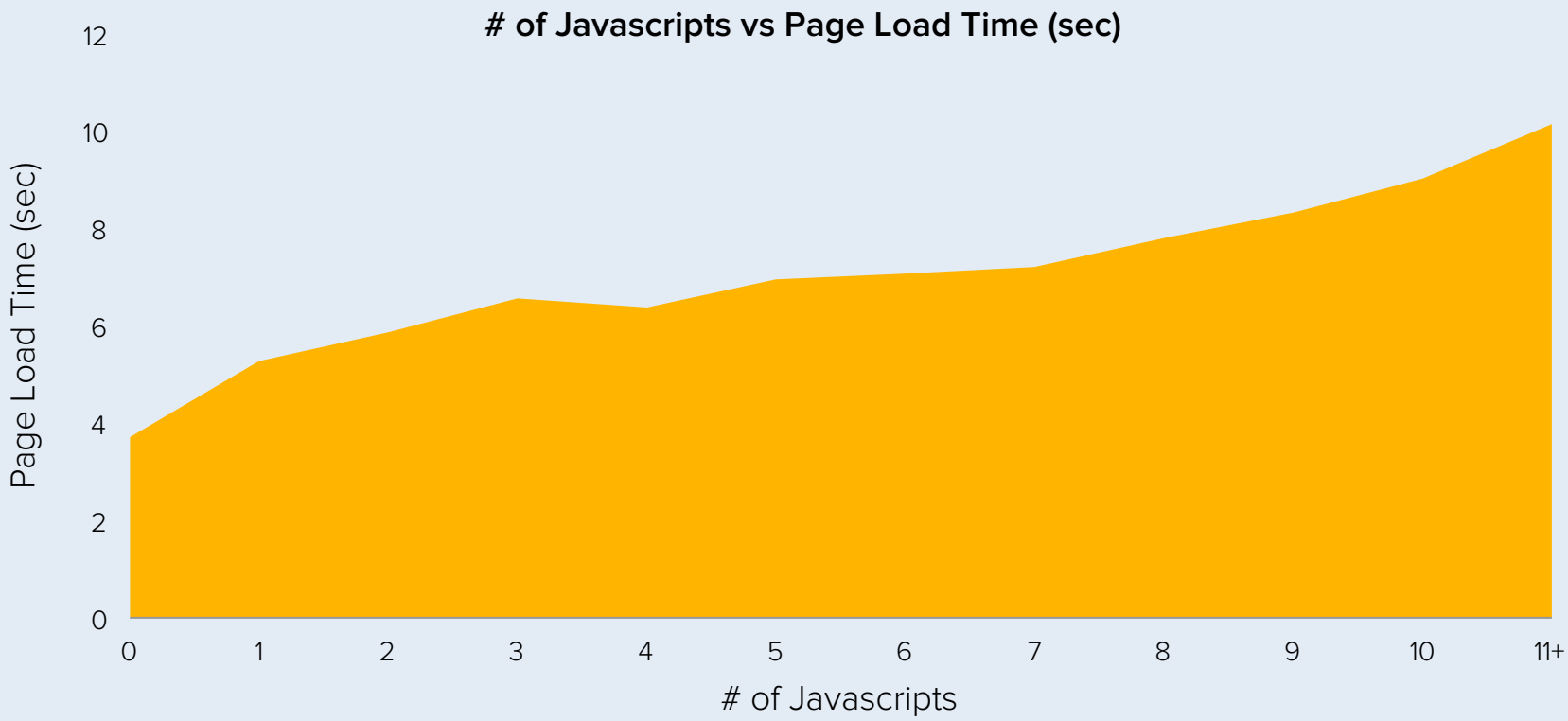
AVG PAGE SIZE

AVG SIZE OF TOTAL IMAGES	**653 KB**
AVG SIZE OF TOTAL JAVASCRIPT FILES	**173 KB**
AVG SIZE OF TOTAL CSS FILES	**34 KB**
AVG SIZE OF TOTAL OTHER FILES	**260 KB**



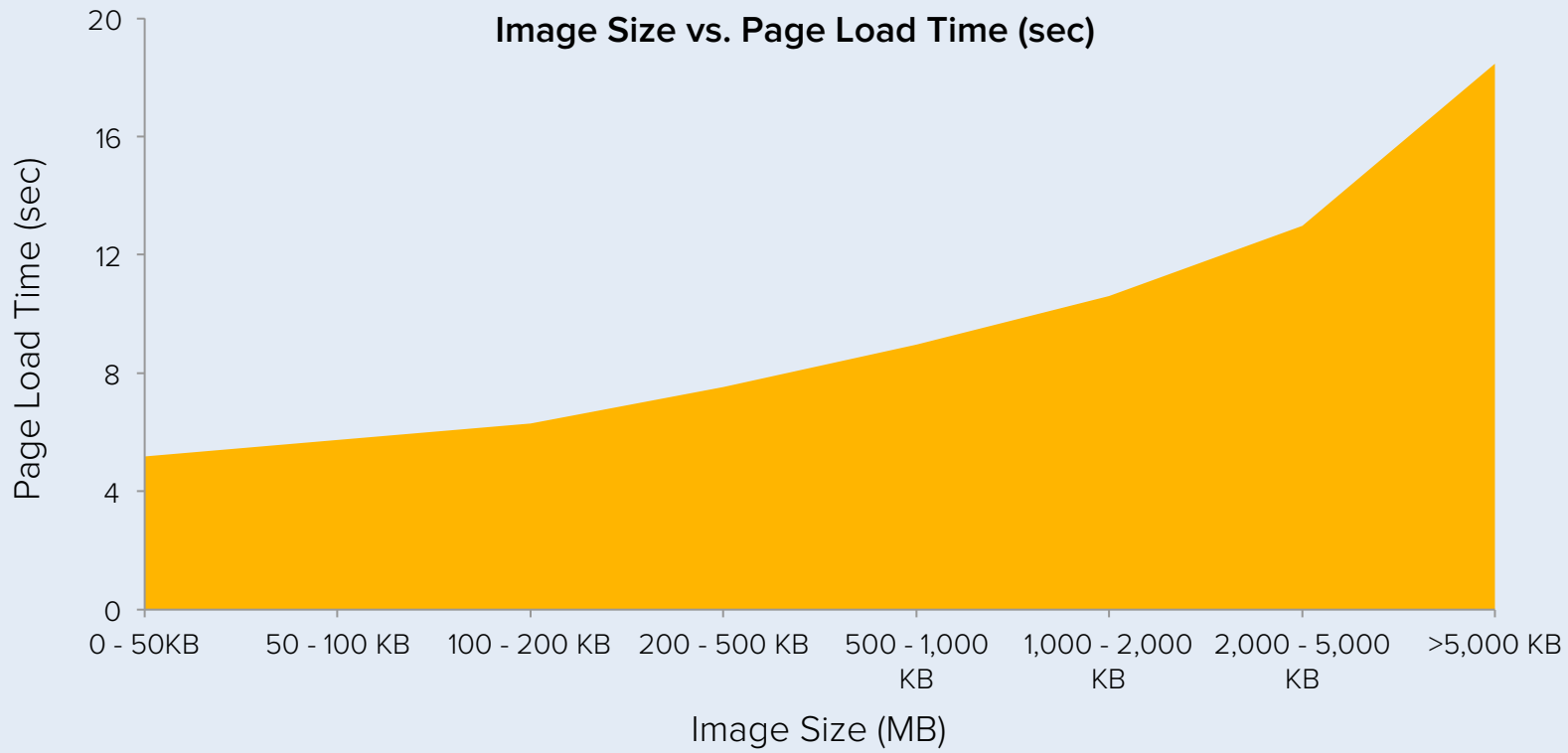
JAVASCRIPT BREAKDOWN

JavaScript is generally the worst offender of poor performance, especially as it relates to the number of JavaScript files.



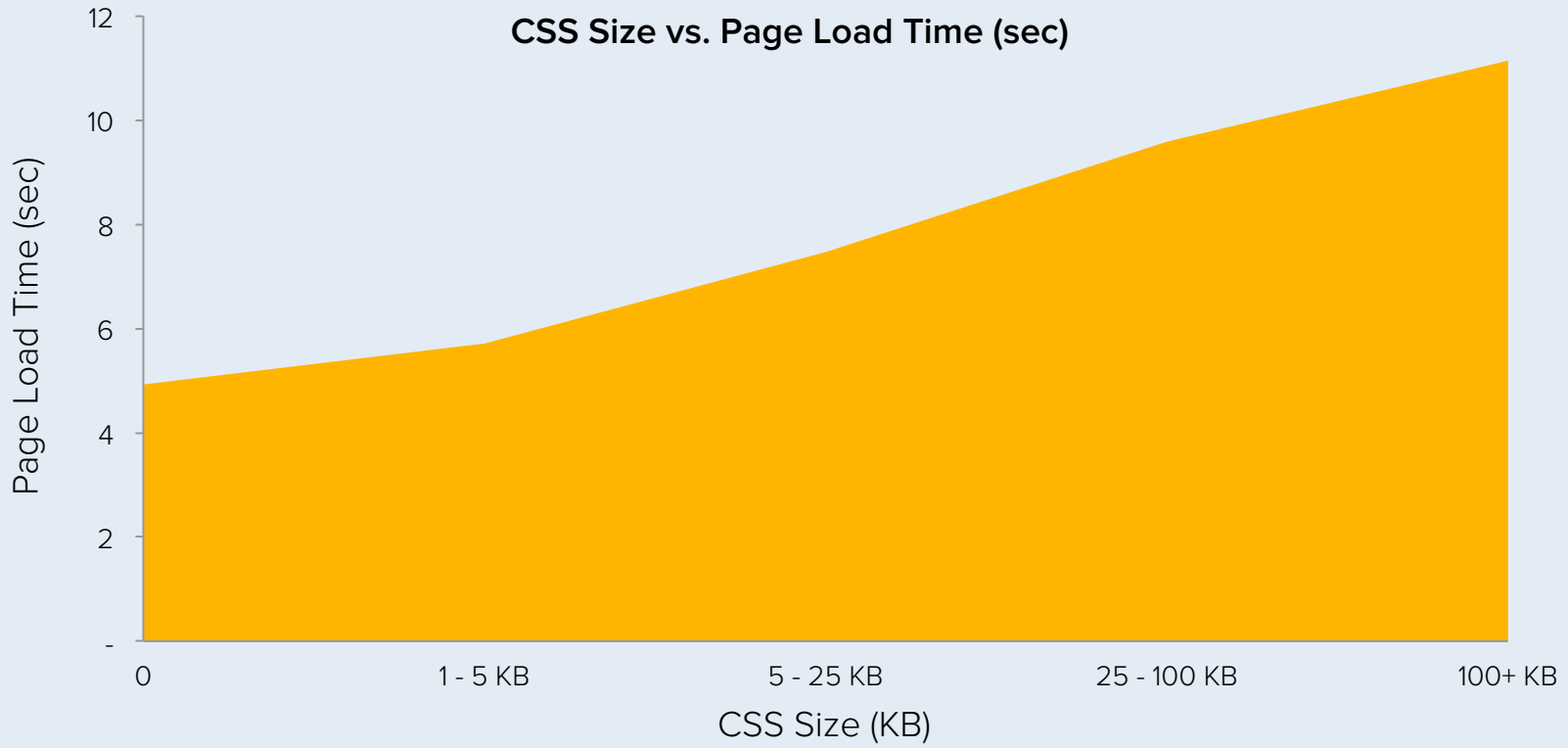
IMAGES BREAKDOWN

A high amount of images and large image sizes are also one of the top culprits when it comes to slowing down page loads.



CSS BREAKDOWN

CSS can be a contributing factor of poor performance, especially in terms of its size. Since CSS paints the visual display of a webpage, larger CSS files have a higher amount of display work needed to executed by the browser.



Chapter 5

How to Optimize Your **WEBSITE ASSETS**

There are many key methods and tactics you can use to optimize your web performance.

Let's take a deep dive into the best practices to optimize the individual assets of HTML, CSS, JavaScript, and images.

> The best strategy is to consider speed a *feature* just like one of the other features on your site.
>
> ***Stoyan Stefanov**, Engineer @ Facebook & Author*

Remember, as we discussed in the previous chapter, there are 2 key factors that will help improve our performance: (1) Reducing the total number of requests and (2) reducing the overall asset size. These will be the main actions we focus on for optimizing our web performance as designers.

OPTIMIZE ORDER OF EXECUTION

Always load your external CSS files first, and then your external and inline JavaScript second in your <head>. JavaScript files are generally larger than CSS files and take a longer time to download, which delays the download of other assets.

```
<head>
<link rel="stylesheet" type="text/css" href=”s1.css" />
<link rel="stylesheet" type="text/css" href="stylesheet2.css" />
<link rel="stylesheet" type="text/css" href="stylesheet3.css" />
<link rel="alternate" type="application/rss+xml" title="Say hello" href="front.xml" />
<link rel="shortcut icon" type="image/x-icon" href="favicon.ico">
<script type="text/javascript">
  document.write("Hello world!");
</script>
</head>
```

COMBINE CSS FILES

Combining, or “concatenating”, as many CSS files together decreases the number of requests and round trips the browser has to make, which increases the page load time and reduces latency.

```
<link rel="stylesheet" type=“text/css” href=”stylesheet1.css">
<link rel="stylesheet" type=“text/css” href=”stylesheet2.css">
<link rel=“stylesheet” type=“text/css” href=“stylesheet3.css”>
```

```
<link rel=“stylesheet” type=“text/css” href=“combined.css”>
```

COMBINE JAVASCRIPT FILES

Just like CSS, combining or “concatenating” as many JavaScript files together decreases the number of requests and round trips the browser has to make, which increases the page load time.

```
<script type="text/javascript" src=“javascript1.js"></script>
<script type="text/javascript" src=“javascript2.js"></script>
<script type="text/javascript" src=“javascript3.js"></script>
```

```
<script type="text/javascript" src=“combined.js"></script>
```

COMBINE IMAGES USING CSS SPRITES

Use CSS sprites to combine many image files into one. This significantly reduces the number of image requests and overall total page size. This technique also allows for faster parallelization of asset downloads.

Here are some great free CSS sprite creator tools:

- SpriteMe
- Compass
- SpritePad
- Spritebox

MINIFY HTML

Compressing your HTML file to remove comments, whitespace between tags, and unnecessary closing tags (such as) can reduce the overall HTML file size and speed up the parsing and execution of a webpage.

Here are some great tools to automatically minify your HTML files:

- HTML Compressor
- Google HTML Compressor and Minifier
- TextFixer

MINIFY JAVASCRIPT

Compressing, or “minifying”, your JavaScript files includes eliminating unnecessary line breaks, extra spaces, and indentation, which reduces the overall size of the JavaScript files and increase page load time.

Here are some great tools to automatically minify your JavaScript files:

- JavaScript Compressor
- JSCompress
- Online YUI Compressor

MINIFY CSS

Compressing, or “minifying”, your CSS files includes removing all whitespace and semicolons for the last property of a CSS declaration, which reduces the overall size of the CSS files and increases page load times.

Here are some great tools to automatically minify your CSS files:

- CSS Compressor
- CSS Drive
- Minify CSS
- Online YUI Compressor

COMPRESS IMAGES

Compress the file size of your images. You can manually tune the save settings for your images in design software such as Photoshop, Illustrator, etc. Or, you can use free online tools that automatically compress your images.

Here are some great tools to automatically compress your images:

- Yahoo! Smush.it
- JPEG-Optimizer
- Trimage

AVOID CSS @import

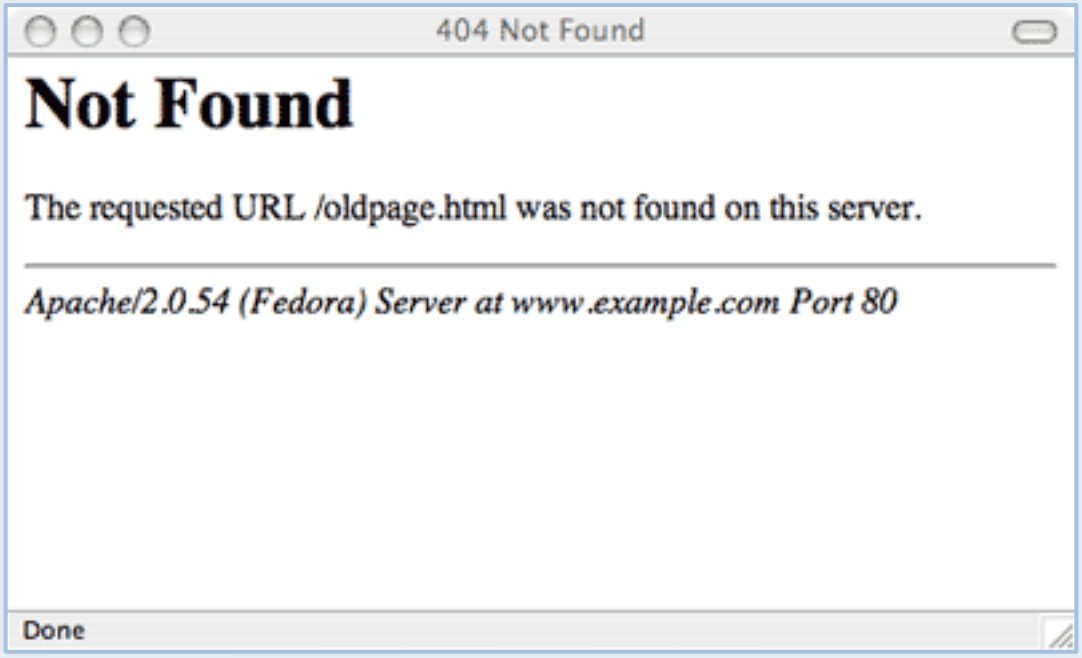
Using CSS @import prevents the browser from being able to download the CSS files in parallel, which increases the number of requests the browser needs to make. Always use the traditional <link> tag for CSS to ensure parallelized downloads.

```
@import url(“stylesheet2.css”)
```

```
<link rel="stylesheet" href=”stylesheet1.css">
<link rel="stylesheet" href=”stylesheet2.css">
```

AVOID BAD REQUESTS

Be sure to remove any broken links, missing images, or other asset requests that result in 404 errors. Failing to do so creates a (higher) number of requests for non-existent assets that slows down the loading of the page.



USE DATA URIs

For smaller images, data URIs are a great technique to cut down image requests by “inlining images” into HTML or CSS files. The image is immediately available when its host document is downloaded. Data URIs use Base64 encoding and follow this basic format:

---

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

LOAD 3RD-PARTY ASSETS ASYNCHRONOUSLY

Assets that aren't needed to construct the display of a webpage, most notably JavaScript files, can be loaded asynchronously so it does not block important resources from loading. These types of assets are most notably social share widgets (Facebook, Twitter, etc.) and analytics tracking (Google Analytics, etc.)

```
<script>
var node = document.createElement('script');
node.type = 'text/javascript';
node.async = true;
node.src = 'example.js';
// Now insert the node into the DOM, perhaps using insertBefore()
</script>
```

SPECIFY IMAGE DIMENSIONS

When possible, you should specify image dimensions in the HTML or CSS and serve a properly resized image that fits such dimensions. This will reduce the actual image file size and also prevent reflows the browser needs to make when loading the images.

```
<img src="myimage.jpg" height="225" width="150">
```

USE EFFICIENT CSS SELECTORS

You should write your CSS so that it is as efficient as possible. This includes avoiding a universal key selector (*), using class and ID selectors over tag selectors, removing redundant qualifiers, and avoiding using descendant selectors. The goal is to always be as specific and individual with your CSS as possible. For example:

```
ul li {color: blue;}
ol li {color: red;}
```

```
.unordered-list-item {color: blue;}
.ordered-list-item {color: red;}
```

SPECIFY A CHARACTER SET

By specifying a character set in the HTTP response headers of your HTML document, this allows the browser to begin parsing HTML and executing scripts immediately.

```
<meta http-equiv="content-type" content="text/html;charset=utf-8">
```

PUT CSS IN <HEAD>

It's a best practice to always place your CSS in the <head> section of your HTML document. Removing inline style blocks and using <link> CSS files in the <head> section improves the browser render and display load times.

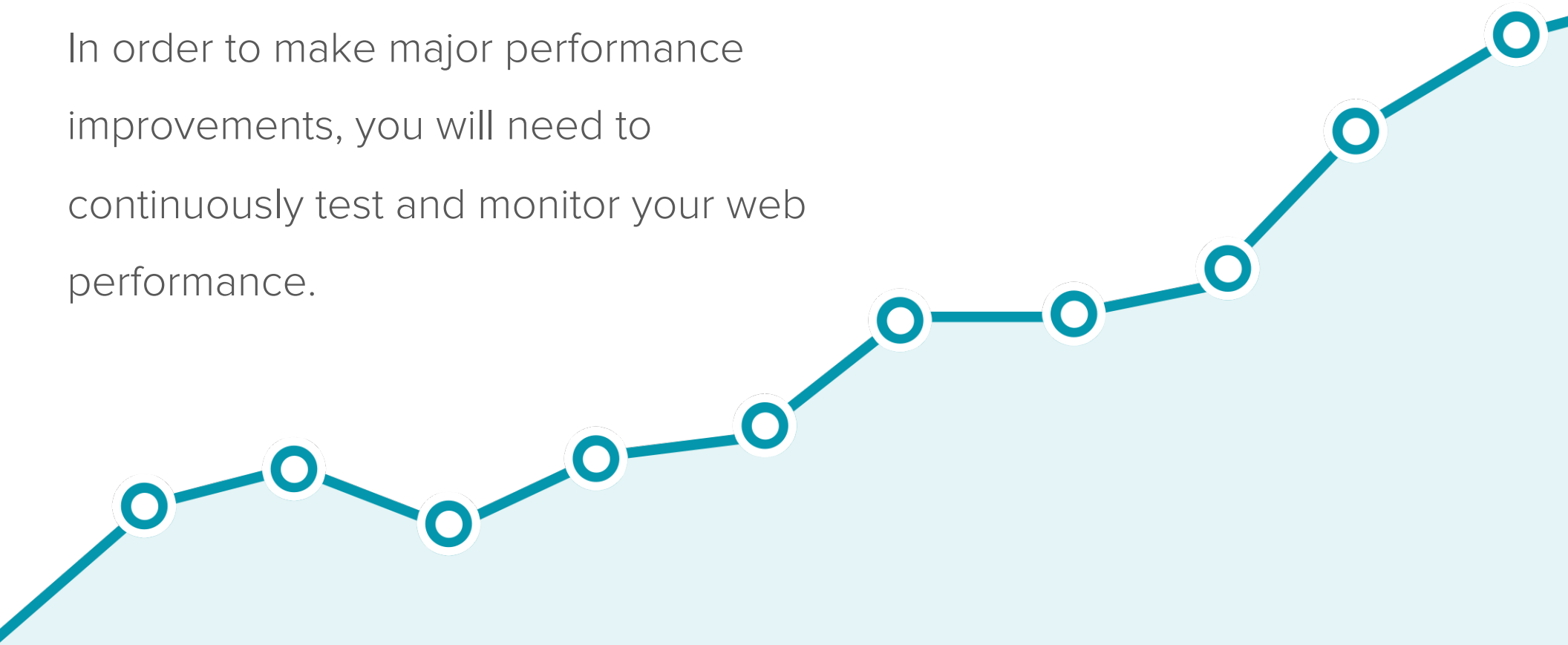
```
<p style="font-size: 1.2em; font-weight: bold; font-family: arial, helvetica;">
```

```
<head>
    <link rel="stylesheet" type="text/css" href="stylesheet.css">
</head>
```

Chapter 6

How to Monitor Your **WEB PERFORMANCE**

In order to make major performance improvements, you will need to continuously test and monitor your web performance.



> If you can't measure it, you can't *improve* it.
>
> ***Lord Kelvin**, 19th Century Physicist and Engineer*

The first thing you should do is test your website to establish a baseline of your current performance. You need to understand how your website and its assets load and where there are areas to improve. There are several key factors you should test your website against: different browsers, locations, and connectivities.

Testing your website only provides a single snapshot in time, however. To capture a holistic view, you'll need to monitor and track your performance over time. This way, you can understand your performance trends -- how your website performs across the world, with various browsers, different connectivities, and traffic spikes.

Here's a list of free resources to test and monitor your web performance:

- Websitetest.com
- Webpagetest.org
- YSlow
- Mobitest
- Yottaa

Chapter 7

Design With Performance **AS A PRIORITY**

The ultimate goal of this eBook is to create a fundamental shift in how you approach design: to design with performance as a priority.

By looking at web design through the lens of performance, you'll be able to build better, faster, and more scalable websites and web apps.

> Exceptional performance means being *faster* than a user expects us to be.
>
> ***Alois Reitebauer**, Product Manager & Evangelist @ Compuware*

What is the goal of design? Design should help users achieve tasks as fast as possible. Therefore, performance should be an integral part of your design process that drives decisions. Build performance best practices into your daily design habits. You should reframe your mindset so you are constantly thinking: "Do I really need this image on the webpage at all? How can I rewrite and condense this CSS file? What's the best way to implement this design so it loads fast?”

Here are a few ways you can keep performance at top of the mind in your design process:

- Create a performance checklist for every design or asset you create
- Build a performance budget for your website (i.e. our website must load in 2 seconds. If it doesn't, we must deliver our content in a way that it does meet this goal or we must remove some assets altogether)
- Conduct a regular review of your website performance and try to improve it just by optimizing your assets

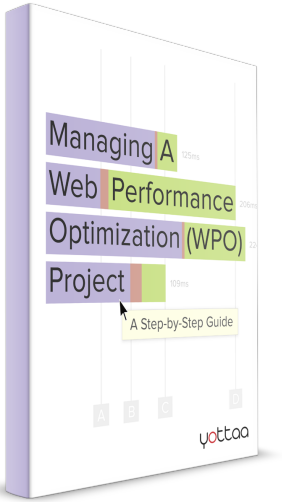
* FREE BONUS RESOURCES *

FREE GUIDE

9 TIPS FOR BENCHMARKING YOUR WEB PERFORMANCE

This guide gives you 9 key tips to ensure you get the most actionable insights from your benchmarking program!

DOWNLOAD GUIDE NOW



FREE EBOOK

HOW TO MANAGE A WPO PROJECT: A STEP-BY-STEP GUIDE

Designed to help web developers and website owners understand the entire WPO process, this eBook will help you to put together your own action plan to improve performance.

DOWNLOAD EBOOK NOW

FREE EBOOK

A Designer's Guide to Web Performance

This ebook gives any web designer the how-to steps, tools & best practices needed to design & build faster websites & web apps.

DOWNLOAD EBOOK NOW

yottaa

See how much faster your site can be on Yottaa

Create a faster site that delivers more sales, higher conversions, and better user engagement with Yottaa

FREE 14-DAY TRIAL

Share this eBook with your friends!