Performance Best Practices for Web Developers

Even Faster Web Sites

O'REILLY®

Steve Souders

Web Programming

O'REILLY®

Even Faster Web Sites

Performance is critical to the success of any web site, and yet today's web applications push browsers to their limits with increasing amounts of rich content and heavy use of Ajax. In this book, Steve Souders, web performance evangelist at Google and former Chief Performance Yahoo!, provides valuable techniques to help you optimize your site's performance.

Souders' previous book, the bestselling *High Performance Web Sites*, shocked the web development world by revealing that 80% of the time it takes for a web page to load is on the client side. In *Even Faster Web Sites*, Souders and eight expert contributors provide best practices and pragmatic advice for improving your site's performance in three critical categories:

- **JavaScript—Get advice for understanding Ajax performance, writing efficient JavaScript, creating responsive applications, loading scripts without blocking other components, and more.**
- **Network—Learn to share resources across multiple domains, reduce image size without loss of quality, and use chunked encoding to render pages faster.**
- **Browser—Discover alternatives to iframes, how to simplify CSS selectors, and other techniques.**

Speed is essential for today's rich media web sites and Web 2.0 applications. With this book, you'll learn how to shave precious seconds off your sites' load times and make them respond even faster.

"Even Faster Web Sites *has the latest up-to-date, curated wisdom on how to make your site scream. I loved the book's format—lots of topics, each explored deeply by some of the most respected authorities in the field. Everyone on my team will own a copy.*"

—Bill Scott, Director, UI Engineering, Netflix

Steve Souders works at Google on web performance and open source initiatives. He is the creator of YSlow, the performance analysis extension to Firebug, and is cochair of Velocity, O'Reilly's web performance and operations conference. Steve frequently speaks at conferences and at high-level companies including Microsoft, Amazon, MySpace, LinkedIn, and Facebook.

Contributing authors:
Dion Almaer, Douglas Crockford, Ben Galbraith, Tony Gentilcore, Dylan Schiemann, Stoyan Stefanov, Nicole Sullivan, and Nicholas C. Zakas

oreilly.com

US $34.99 **CAN $43.99**
ISBN: 978-0-596-52230-8

Even Faster Web Sites

Even Faster Web Sites

Steve Souders

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

Even Faster Web Sites
by Steve Souders

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Mary E. Treseler
Production Editor: Sarah Schneider
Copyeditor: Audrey Doyle
Proofreader: Sarah Schneider
Indexer: Lucie Haskins
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Robert Romano

Printing History:
June 2009: First Edition.

ISBN: 978-0-596-52230-8

[M]

1243719104

Table of Contents

Credits

Even Faster Web Sites contains six chapters contributed by the following authors.

Dion Almaer is the cofounder of Ajaxian.com, the leading source of the Ajax community. For his day job, Dion coleads a new group at Mozilla focusing on developer tools for the Web, something he has been passionate about doing for years. He is excited for the opportunity, and he gets to work with Ben Galbraith, his partner in crime on Ajaxian and now at Mozilla. Dion has been writing web applications since Gopher, has been fortunate enough to speak around the world, has published many articles and a book, and, of course, covers life, the universe, and everything else on his blog at *http://almaer.com/blog*.

Douglas Crockford was born in the wilds of Minnesota, but left when he was only six months old because it was just too damn cold. He turned his back on a promising career in television when he discovered computers. He has worked in learning systems, small business systems, office automation, games, interactive music, multimedia, location-based entertainment, social systems, and programming languages. He is the inventor of Tilton, the ugliest programming language that was not specifically designed to be an ugly programming language. He is best known for having discovered that there are good parts in JavaScript. This was an important and unexpected discovery. He discovered the JSON (JavaScript Object Notation) data interchange format. He is currently working on making the Web a secure and reliable software-delivery platform. He has his work cut out for him.

Ben Galbraith is the codirector of developer tools at Mozilla and the cofounder of Ajaxian.com. Ben has long juggled interests in both business and tech, having written his first computer program at 6 years old, started his first business at 10, and entered the IT workforce at 12. He has delivered hundreds of technical presentations worldwide, produced several technical conferences, and coauthored more than a half-dozen books. He has enjoyed a variety of business and technical roles throughout his career, including CEO, CIO, CTO, and Chief Software Architect roles in medical, publishing, media, manufacturing, advertising, and software industries. He lives in Palo Alto, California with his wife and five children.

Tony Gentilcore is a software engineer at Google. There, he has helped make the Google home and search results pages lightning fast. He finds that the days seem to fly by while writing web performance tools and techniques. Tony is also the creator of the popular Firefox extension, Fasterfox.

Dylan Schiemann is CEO of SitePen and cofounder of the Dojo Toolkit, an open source JavaScript toolkit for rapidly building web sites and applications, and is an expert in the technologies and opportunities of the Open Web. Under his guidance, SitePen has grown from a small development firm to a leading provider of inventive tools, skilled software engineers, knowledgeable consulting services, and top-notch training and advice. Dylan's commitment to R&D has enabled SitePen to be a major contributor to and creator of pioneering open source web development toolkits and frameworks such as Dojo, cometD, Direct Web Remoting (DWR), and Persevere. Prior to SitePen, Dylan developed web applications for companies such as Renkoo, Informatica, Security FrameWorks, and Vizional Technologies. He is a cofounder of Comet Daily, LLC, a board member at Dojo Foundation, and a member of the advisory board at Aptana. Dylan earned his master's in physical chemistry from UCLA and his B.A. in mathematics from Whittier College.

Stoyan Stefanov is a Yahoo! frontend developer, focusing on web application performance. He is also the architect of the performance extension YSlow 2.0 and cocreator of the Smush.it image optimization tool. Stoyan is a speaker, book author (*Object-Oriented JavaScript* from Packt Publishing), and blogger at *http://phpied.com*, *http://jspatterns.com*, and YUIblog.

Nicole Sullivan is an evangelist, frontend performance consultant, and CSS Ninja. She started the Object-Oriented CSS open source project, which answers the question, How do you scale CSS for millions of visitors or thousands of pages? She also consulted with the W3C for their beta redesign, and she is the cocreator of Smush.it, an image optimization service in the cloud. She is passionate about CSS, web standards, and scalable frontend architecture for large commercial websites. Nicole speaks about performance at conferences around the world, most recently at The Ajax Experience, ParisWeb, and Web Directions North. She blogs at *http://stubbornella.org*.

Nicholas C. Zakas is the author of *Professional JavaScript for Web Developers*, Second Edition (Wrox) and coauthor of *Professional Ajax*, Second Edition (Wrox). Nicholas is principal frontend engineer for the Yahoo! home page and is also a contributor to the Yahoo! User Interface (YUI) library. He blogs regularly at his site, *http://www.nczonline.net*.

Preface

Vigilant: alertly watchful, especially to avoid danger

Anyone browsing this book—or its predecessor, *High Performance Web Sites*—understands the dangers of a slow web site: frustrated users, negative brand perception, increased operating expenses, and loss of revenue. We have to constantly work to make our web sites faster. As we make progress, we also lose ground. We have to be alert for the impact of each bug fix, new feature, and system upgrade on our web site's speed. We have to be watchful, or the performance improvements made today can easily be lost tomorrow. We have to be vigilant.

Vigil: watch kept on a festival eve

According to the Latin root of *vigil*, our watch ends with celebration. Web sites can indeed be faster—dramatically so—and we can celebrate the outcome of our care and attention. It's true! Making web sites faster is attainable. Some of the world's most popular web sites have reduced their load times by 60% using the techniques described in this book. Smaller web properties benefit as well. Ultimately, users benefit.

Vigilante: a self-appointed doer of justice

It's up to us as developers to guard our users' interests. At your site, evangelize performance. Implement these techniques. Share this book with a coworker. Fight for a faster user experience. If your company doesn't have someone focused on performance, appoint yourself to that role. *Performance vigilante*—I like the sound of that.

How This Book Is Organized

This book is a follow-up to my first book, *High Performance Web Sites* (O'Reilly). In that book, I lay out 14 rules for better web performance:

- Rule 1: Make Fewer HTTP Requests
- Rule 2: Use a Content Delivery Network
- Rule 3: Add an `Expires` Header
- Rule 4: Gzip Components

- Rule 5: Put Stylesheets at the Top
- Rule 6: Put Scripts at the Bottom
- Rule 7: Avoid CSS Expressions
- Rule 8: Make JavaScript and CSS External
- Rule 9: Reduce DNS Lookups
- Rule 10: Minify JavaScript
- Rule 11: Avoid Redirects
- Rule 12: Remove Duplicate Scripts
- Rule 13: Configure ETags
- Rule 14: Make Ajax Cacheable

I call them "rules" because there is little ambiguity about their adoption. Consider these statistics for the top 10 U.S. web sites* for March 2007:

- Two sites used CSS sprites.
- 26% of resources had a future `Expires` header.
- Five sites compressed their HTML, JavaScript, and CSS.
- Four sites minified their JavaScript.

The same statistics for April 2009 show that these rules are gaining traction:

- Nine sites use CSS sprites.
- 93% of resources have a future `Expires` header.
- Ten sites compress their HTML, JavaScript, and CSS.
- Nine sites minify their JavaScript.

The rules from *High Performance Web Sites* still apply and are where most web companies should start. Progress is being made, but there's still more work to be done on this initial set of rules.

But the Web isn't standing still, waiting for us to catch up. Although the 14 rules from *High Performance Web Sites* still apply, the growth in web page content and Web 2.0 applications introduces a new set of performance challenges. *Even Faster Web Sites* provides the best practices needed by developers to make these next-generation web sites faster.

The chapters in this book are organized into three areas: JavaScript performance (Chapters 1–7), network performance (Chapters 8–12), and browser performance (Chapters 13 and 14). A roundup of the best tools for analyzing performance comes in the Appendix.

* AOL, eBay, Facebook, Google Search, Live Search, MSN.com, MySpace, Wikipedia, Yahoo!, and YouTube, according to Alexa.

Six of the chapters were written by contributing authors:

- Chapter 1, *Understanding Ajax Performance*, by Douglas Crockford
- Chapter 2, *Creating Responsive Web Applications*, by Ben Galbraith and Dion Almaer
- Chapter 7, *Writing Efficient JavaScript*, by Nicholas C. Zakas
- Chapter 8, *Scaling with Comet*, by Dylan Schiemann
- Chapter 9, *Going Beyond Gzipping*, by Tony Gentilcore
- Chapter 10, *Optimizing Images*, by Stoyan Stefanov and Nicole Sullivan

These authors are experts in each of these areas. I wanted you to hear from them directly, in their own voices. To help identify these chapters, the name(s) of the contributing author(s) are on the chapter's opening page.

JavaScript Performance

In my work analyzing today's web sites, I consistently see that JavaScript is the key to better-performing web applications, so I've started the book with these chapters.

Douglas Crockford wrote Chapter 1, *Understanding Ajax Performance*. Doug describes how Ajax changes the way browsers and servers interact, and how web developers need to understand this new relationship to properly identify opportunities for improving performance.

Chapter 2, *Creating Responsive Web Applications*, by Ben Galbraith and Dion Almaer, ties JavaScript performance back to what really matters: the user experience. Today's web applications invoke complex functions at the click of a button and must be evaluated on the basis of what they're forcing the browser to do. The web applications that succeed will be written by developers who understand the effects of their code on response time.

I wrote the next four chapters. They focus on the mechanics of JavaScript—the best way to package it and load it, and where to insert it in your pages. Chapter 3, *Splitting the Initial Payload*, describes the situation facing many web applications today: a huge JavaScript download at the beginning of the page that blocks rendering as well as further downloads. The key is to break apart this monolithic JavaScript for more efficient loading.

Chapters 4 and 5 go together. In today's most popular browsers, external scripts block everything else in the page. Chapter 4, *Loading Scripts Without Blocking*, explains how to avoid these pitfalls when loading external scripts. Loading scripts asynchronously presents a challenge when inlined code depends on them. Luckily, there are several techniques for coupling inlined code with the asynchronous scripts on which they depend. These techniques are presented in Chapter 5, *Coupling Asynchronous Scripts*.

Chapter 6, *Positioning Inline Scripts*, presents performance best practices that apply to inline scripts, especially the impact they have on blocking parallel downloads.

I think of Chapter 7, *Writing Efficient JavaScript*, written by Nicholas C. Zakas, as the complement to Doug's chapter (Chapter 1). Whereas Doug describes the Ajax landscape, Nicholas zooms in on several specific techniques for speeding up JavaScript.

Network Performance

Web applications aren't desktop applications—they have to be downloaded over the Internet each time they are used. The adoption of Ajax has resulted in a new style of data communication between servers and clients. Some of the biggest opportunities for growth in the web industry are in emerging markets where Internet connectivity is a challenge, to put it mildly. All of these factors highlight the need for improved network performance.

In Chapter 8, *Scaling with Comet*, Dylan Schiemann describes an architecture that goes beyond Ajax to provide high-volume, low-latency communication for real-time applications such as chat and document collaboration.

Chapter 9, *Going Beyond Gzipping*, describes how turning on compression isn't enough to guarantee optimal delivery of your web site's content. Tony Gentilcore reveals a little-known phenomenon that severely hinders the network performance of 15% of the world's Internet users.

Stoyan Stefanov and Nicole Sullivan team up to contribute Chapter 10, *Optimizing Images*. This is a thorough treatment of the topic. This chapter reviews all popular image formats, presents numerous image optimization techniques, and describes the image compression tools of choice.

The remaining chapters were written by me. Chapter 11, *Sharding Dominant Domains*, reminds us of the connection limits in the popular browsers of today, as well as the next generation of browsers. It includes techniques for successfully splitting resources across multiple domains.

Chapter 12, *Flushing the Document Early*, walks through the benefits and many gotchas of using chunked encoding to start rendering the page even before the full HTML document has arrived.

Browser Performance

Iframes are an easy and frequently used technique for embedding third-party content in a web page. But they come with a cost. Chapter 13, *Using Iframes Sparingly*, explains the downsides of iframes and offers a few alternatives.

Chapter 14, *Simplifying CSS Selectors*, presents the theories about how complex selectors can impact performance, and then does an objective analysis to pinpoint the situations that are of most concern.

The Appendix, *Performance Tools*, describes the tools that I recommend for analyzing web sites and discovering the most important performance improvements to work on.

Conventions Used in This Book

The following typographical conventions are used in this book:

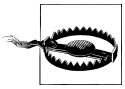
Italic
: Indicates new terms, URLs, email addresses, filenames, file extensions, pathnames, and directories

`Constant width`
: Indicates commands, options, switches, variables, attributes, keys, functions, types, classes, namespaces, methods, modules, properties, parameters, values, objects, events, event handlers, XML tags, HTML tags, macros, the contents of files, and the output from commands

`Constant width bold`
: Shows commands or other text that should be typed literally by the user

`Constant width italic`
: Shows text that should be replaced with user-supplied values

This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Comments and Questions

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596522308

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our web site at:

http://www.oreilly.com

Using Code Examples

You may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from this book *does* require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation *does* require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Even Faster Web Sites*, by Steve Souders. Copyright 2009 Steve Souders, 978-0-596-52230-8."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

When you see a Safari® Books Online icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at *http://my.safaribooksonline.com*.

Acknowledgments

I first want to thank the contributing authors: Dion Almaer, Doug Crockford, Ben Galbraith, Tony Gentilcore, Dylan Schiemann, Stoyan Stefanov, Nicole Sullivan, and Nicholas Zakas. They've made this a special book. Each of them is an expert in his or her own right. Most of them have written their own books. By sharing their expertise, they've helped create something unique.

I want to thank all the reviewers: Julien Lecomte, Matthew Russell, Bill Scott, and Tenni Theurer. I extend an especially strong thank you to Eric Lawrence and Andy Oram. Eric reviewed both this book as well as *High Performance Web Sites*. In both cases, he provided incredibly thorough and knowledgeable feedback. Andy was my editor on *High Performance Web Sites*. More than anyone else, he is responsible for improving how this book reads, making it flow smoothly from line to line, section to section, and chapter to chapter.

A special thank you goes to my editor, Mary Treseler. Coordinating a book with multiple authors is an opportunity that many editors will pass over. I'm glad that she took on this project and helped guide it from a bunch of ideas to what you're holding in your hands now.

I work with many people at Google who have a penchant for web performance. Tony Gentilcore is the creator of Fasterfox and the author of Chapter 9. He's also my officemate. Several times a day we'll stop to discuss web performance. Steve Lamm, Lindsey Simon, and Annie Sullivan are strong advocates for performance who I work with frequently. Other Googlers who have contributed to what I know about web performance include Jacob Hoffman-Andrews, Kyle Scholz, Steve Krulewitz, Matt Gundersen, Gavin Doughtie, and Bryan McQuade.

Many of the insights in this book come from my friends outside Google. They know that if they tell me about a good performance tip, it's likely to end up in a book or blog post. These performance cohorts include Dion Almaer, Artur Bergman, Doug Crockford, Ben Galbraith, Eric Goldsmith, Jon Jenkins, Eric Lawrence, Mark Nottingham, Simon Perkins, John Resig, Alex Russell, Eric Schurman, Dylan Schiemann, Bill Scott, Jonas Sicking, Joseph Smarr, and Tenni Theurer.

I've inevitably forgotten to mention someone in these lists. I apologize, and want to thank all of you for taking the time to send me email and talk to me at conferences. Hearing your lessons learned and success stories keeps me going. It's important to know there are so many of us who are working to make the Web a faster place.

Thank you to my parents for being proud to have a son who's an author. Most importantly, thank you to my wife and three daughters. I promise to take a break now.

CHAPTER 1

Understanding Ajax Performance

Douglas Crockford

Premature optimization is the root of all evil.

—Donald Knuth

Trade-offs

The design and construction of a computer program can involve thousands of decisions, each representing a trade-off. In difficult decisions, each alternative has significant positive and negative consequences. In trading off, we hope to obtain a near optimal good while minimizing the bad. Perhaps the ultimate trade-off is:

> I want to go to heaven, but I don't want to die.

More practically, the Project Triangle:

> Fast. Good. Cheap. Pick Two.

predicts that even under ideal circumstances, it is not possible to obtain fast, good, and cheap. There must be a trade-off.

In computer programs, we see time versus memory trade-offs in the selection of algorithms. We also see expediency or time to market traded against code quality. Such trades can have a large impact on the effectiveness of incremental development.

Every time we touch the code, we are trading off the potential of improving the code against the possibility of injecting a bug. When we look at the performance of programs, we must consider all of these trade-offs.

Principles of Optimization

When looking at optimization, we want to reduce the overall cost of the program. Typically, this cost is the perceived execution time of the program, although we could

optimize on other factors. We then should focus on the parts of the program that contribute most significantly to its cost.

For example, suppose that by profiling we discover the cost of a program's four modules.

Module	A	B	C	D
Cost	54%	4%	30%	12%

If we could somehow cut the cost of Module B in half, we would reduce the total cost by only 2%. We would get a better result by cutting the cost of Module A by 10%. There is little benefit from optimizing components that do not contribute significantly to the cost.

The analysis of applications is closely related to the analysis of algorithms. When looking at execution time, the place where programs spend most of their time is in loops. The return on optimization of code that is executed only once is negligible. The benefits of optimizing inner loops can be significant.

For example, if the cost of a loop is linear with respect to the number of iterations, then we can say it is $O(n)$, and we can graph its performance as shown in Figure 1-1.

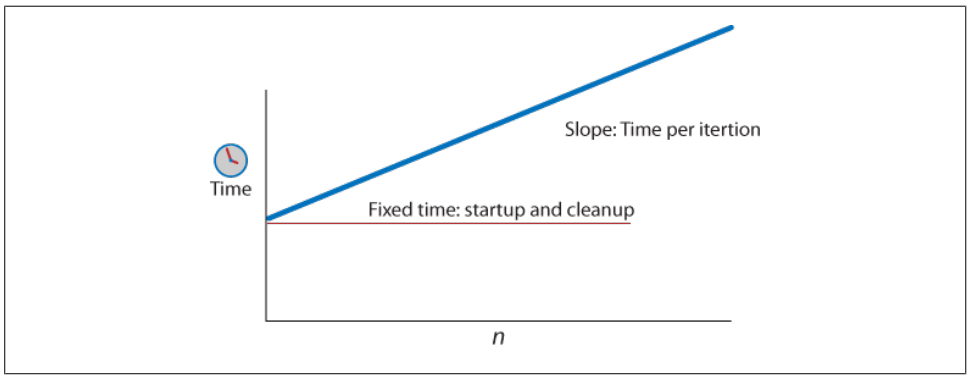


Figure 1-1. Performance of a loop

The execution time of each iteration is reflected in the slope of the line: the greater the cost, the steeper the slope. The fixed overhead of the loop determines the elevation of its starting point. There is usually little benefit in reducing the fixed overhead. Sometimes there is a benefit in increasing the fixed overhead if the cost of each increment can be reduced. That can be a good trade-off.

In addition to the plot of execution time, there are three lines—the Axes of Error—that our line must not intersect (see Figure 1-2). The first is the *Inefficiency* line. Crossing this line reduces the user's ability to concentrate. This can also make people irritable. The second is the *Frustration* line. When this line is crossed, the user is aware that he