

<<更快速网站>>

图书基本信息

书名：<<更快速网站>>

13位ISBN编号：9787564119348

10位ISBN编号：7564119349

出版时间：2009年12月

出版时间：东南大学出版社

作者：Steve Souders

页数：231

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

前言

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.

<<更快速网站>>

内容概要

《更快速网站(影印版)》内容简介：对于任何成功的网站来说，性能是至关重要的。但伴随着不断增长的丰富内容和Ajax的过度使用，如今的Web应用已经将浏览器推至性能极限。在《更快速网站(影印版)》中，Google的Web性能专家和前任雅虎首席网站性能官Steve Souders提供了宝贵的技术，来帮助你优化网站性能。

作者的上一《更快速网站(影印版)》是非常畅销的《High Performance Web Sites》，它透露了80%的网页加载时间是花在客户端，使网络开发世界为之震惊。

在《更快速网站(影印版)》中，Souders和8位专家撰稿人提供了最佳实践和实用建议，用于在三个范畴提高网站的性能：JavaScript——获取用于了解Ajax性能的建议，编写有效的JavaScript，创建响应程序，加载脚本时不阻止其他组件等等。

Network——学习穿过多个域共享资源，减小图片尺寸而不损失质量，以及使用分块编码（chunked encoding）来更快呈现页面。

Browser——探索内嵌框架（iframe）的替代方案、如何简化CSS选择器和其他技术。

对于当今的富媒体网站和Web 2.0应用来说，速度是至关重要的。

有了这《更快速网站(影印版)》，你将学习到如何减少你的网站的加载时间，让它们响应得更快。

作者简介

Steve Souders，在Google从事网络性能和开放源码计划方面的工作。他是YSlow（Firebug性能分析扩展）的创造者，并且担任Velocity（O'Reilly的网络性能和业务运营会议）的联合主席。Steve经常在会议上或者高级别公司中发言，包括微软、亚马逊、MySpace、LinkedIn、Facebook。

书籍目录

Credits Preface 1. Understanding Ajax Performance Trade-offs Principles of Optimization Ajax Browser Wow! JavaScript Summary 2. Creating Responsive Web Applications What Is Fast Enough? Measuring Latency When Latency Goes Bad Threading Ensuring Responsiveness Web Workers Gears Timers Effects of Memory Use on Response Time Virtual Memory Troubleshooting Memory Issues Summary 3. Splitting the Initial Payload Kitchen Sink Savings from Splitting Finding the Split Undefined Symbols and Race Conditions Case Study: Google Calendar 4. Loading Scripts Without Blocking Scripts Block Making Scripts Play Nice XHR Eval XHR Injection Script in Iframe Script DOM Element Script Defer document.write Script Tag Browser Busy Indicators Ensuring (or Avoiding) Ordered Execution Summarizing the Results And the Winner Is 5. Coupling Asynchronous Scripts Code Example: menu.js Race Conditions Preserving Order Asynchronously Technique 1: Hardcoded Callback Technique 2: Window Onload Technique 3: Timer Technique 4: Script Onload Technique 5: Degrading Script Tags Multiple External Scripts Managed XHR DOM Element and Doc Write General Solution Single Script Multiple Scripts Asynchronicity in the Real World Google Analytics and Dojo YUI Loader Utility 6. Positioning Inline Scripts Inline Scripts Block Move Inline Scripts to the Bottom Initiate Execution Asynchronously Use Script Defer Preserving CSS and JavaScript Order Danger: Stylesheet Followed by Inline Script Inline Scripts Aren't Blocked by Most Downloads Inline Scripts Are Blocked by Stylesheets This Does Happen 7. Writing Efficient JavaScript Managing Scope Use Local Variables Scope Chain Augmentation Efficient Data Access Flow Control Fast Conditionals Fast Loops String Optimization String Concatenation Trimming Strings Avoid Long-Running Scripts Yielding Using Timers Timer Patterns for Yielding Summary 8. Scaling with Comet How Comet Works Transport Techniques Polling Long Polling Forever Frame XHR Streaming Future Transports Cross-Domain Effects of Implementation on Applications Managing Connections Measuring Performance Protocols Summary 9. Going Beyond Gzipping Why Does This Matter? What Causes This? Quick Review The Culprit Examples of Popular Turtle Tappers How to Help These Users? Design to Minimize Uncompressed Size Educate Users Direct Detection of Gzip Support 10. Optimizing Images Two Steps to Simplify Image Optimization Image Formats Background Characteristics of the Different Formats More About PNG Automated Lossless Image Optimization Crushing PNGs Stripping JPEG Metadata Converting GIF to PNG Optimizing GIF Animations Smush.it Progressive JPEGs for Large Images Alpha Transparency: Avoid AlphaImageLoader Effects of Alpha Transparency AlphaImageLoader Problems with AlphaImageLoader Progressively Enhanced PNG8 Alpha Transparency Optimizing Sprites ?ber-Sprite Versus Modular Sprite Highly Optimized CSS Sprites Other Image Optimizations Avoid Scaling Images Crush Generated Images Favicons Apple Touch Icon Summary 11. Sharding Dominant Domains Critical Path Who's Sharding? Downgrading to HTTP/1.0 Rolling Out Sharding IP Address or Hostname How Many Domains How to Split Resources Newer Browsers 12. Flushing the Document Early Flush the Head Output Buffering Chunked Encoding Flushing and Gzip Other Intermediaries Domain Blocking During Flushing Browsers: The Last Hurdle Flushing Beyond PHP The Flush Checklist 13. Using Iframes Sparingly The Most Expensive DOM Element Iframes Block Onload Parallel Downloads with Iframes Script Before Iframe Stylesheet Before Iframe Stylesheet After Iframe Connections per Hostname Connection Sharing in Iframes Connection Sharing Across Tabs and Windows Summarizing the Cost of Iframes 14. Simplifying CSS Selectors Types of Selectors ID Selectors Class Selectors Type Selectors Adjacent Sibling Selectors Child Selectors Descendant Selectors Universal Selectors Attribute Selectors Pseudo-Classes and Pseudo-Elements The Key to Efficient CSS Selectors Rightmost First Writing Efficient CSS Selectors CSS Selector Performance Complex Selectors Impact Performance (Sometimes) CSS Selectors to Avoid Reflow Time Measuring CSS Selectors in the Real World Appendix: Performance Tools Index

章节摘录

插图：Refactoring the code can reduce its apparent complexity, making optimization and other transformations more likely to yield benefits. For example, adopting the YSlow rules can have a huge impact on the delivery time of web pages (see <http://developer.yahoo.com/yslow/>). Even so, it is difficult for web applications to get under the Inefficiency line because of the size and complexity of web pages. Web pages are big, heavy, multipart things. Page replacement comes with a significant cost. For applications where the difference between successive pages is relatively small, use of Ajax techniques can produce a significant improvement. Instead of requesting a replacement page as a result of a user action, a packet of data is sent to the server (usually encoded as JSON text) and the server responds with another packet (also typically JSON-encoded) containing data. A JavaScript program uses that data to update the browser's display. The amount of data transferred is significantly reduced, and the time between the user action and the visible feedback is also significantly reduced. The amount of work that the server must do is reduced. The amount of work that the browser must do is reduced. The amount of work that the Ajax programmer must do, unfortunately, is likely to increase. That is one of the trade-offs. The architecture of an Ajax application is significantly different from most other sorts of applications because it is divided between two systems. Getting the division of labor right is essential if the Ajax approach is to have a positive impact on performance. The packets should be as small as possible. The application should be constructed as a conversation between the browser and the server, in which the two halves communicate in a concise, expressive, shared language. Just-in-time data delivery allows the browser side of the application to keep n small, which tends to keep the loops fast. A common mistake in Ajax applications is to send all of the application's data to the browser. This reintroduces the latency problems that Ajax is supposed to avoid. It also enlarges the volume of data that must be handled in the browser, increasing n and again compromising performance.

<<更快速网站>>

媒体关注与评论

“ 本书拥有最近最新的专业知识，能使你的网站飞速运行。
我喜欢这本书的编排，有许多主题，每一个都被该领域最受人尊敬的权威人士所探究。
我的团队中的每个人都将拥有一本。

” ——Bill Scott，Netflix公司UI上程总监

<<更快速网站>>

编辑推荐

《更快速网站(影印版)》是由东南大学出版社出版的。

<<更快速网站>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>