《高性能网站建设指南》

图书基本信息

书名：《高性能网站建设指南》

13位ISBN编号：9787121066191

10位ISBN编号：712106619X

出版时间：2008年

出版社：电子工业出版社

作者：Steve Souders

页数：146 页

译者：刘彦博

《高性能网站建设指南》

内容概要

本书结合Web 2.0以来Web开发领域的最新形势和特点，介绍了网站性能问题的现状、产生的原因，以及改善或解决性能问题的原则、技术技巧和最佳实践。重点关注网页的行为特征，阐释优化Ajax、CSS、JavaScript、Flash和图片处理等要素的技术，全面涵盖浏览器端性能问题的方方面面。在《高性能网站建设指南》中，作者给出了14条具体的优化原则，每一条原则都配以范例佐证，并提供了在线支持。《高性能网站建设指南》内容丰富，主要包括减少HTTP请求、Edge Computing技术、Expires Header技术、Gzip组件、CSS和JavaScript最佳实践、主页内联、Domain最小化、JavaScript优化、避免重定向的技巧、删除重复JavaScript的技巧、关闭ETags的技巧、Ajax缓存技术和最小化技术等。《高性能网站建设指南》适合Web架构师、信息架构师、Web开发人员及产品经理阅读和参考。

《高性能网站建设指南》

作者简介

Steve Souders在Yahoo!担任Chief Performance。他于2000年加盟Yahoo!，在该公司的很多平台和产品团队中工作过。在他到达今天这个位置之前，他就职于My Yahoo!开发团队。.
作为Chief Performance Yahoo!，他开发了一系列优秀软件，可以使网站访问速度变得更快。他构建了用于进行性能分析的工具，并将这些优秀软件和工具传播到Yahoo!的各个产品团队中。
在到Yahoo!之前，Steve就职于很多小型或中型公司，包括他和别人一起创办的两个公司——Helix Systems和CoolSync。

《高性能网站建设指南》

书籍目录

Wikipedia
Yahoo
YouTube
索引

《高性能网站建设指南》

精彩短评

1、内容不错，应该有帮助
2、一层窗户纸，他就是捅破的那根手指
3、经典 提高网站速度的经典十四条 ,写了mind笔记 , 欢迎移步:sunlanda
4、出版年: 2008年。1. 内容就是雅虎军规；2. 看看今天前端框的更新速度，再看看这本书的出版年，感受一下。
5、非常好的一本实践指南，看完收获很大，很多东西都是一般书中不会提到的。顺便感慨一下，现在的一些网站服务也越来越人性化，基于这本书的一些内容提供网站优化建议。
6、都是经典的优化方法，但是现在这些都是大家都了解的了。
7、这书你不得不看啊
8、还不错
9、对当时来说也算是一本介绍全面的经典小作品了吧…也有对内容更细更新的增补版？…
10、简单实用
11、赞
12、书送过来就有破的地方
13、加快网站打开速度的前端优化方法
14、薄薄的一本，快速的过了一遍。
15、感觉并没有太多实质性的内容，不是很值得看
16、书是正品，我很满意，送货服务也好
17、前端优化必读
18、目前水平看一遍有些消化不良
19、薄薄的一本书，但是内容挺丰富的。虽然写的不详细，但是很容易读懂理解~只是我现在刚开始学前端，可能很多东西用不上，但是还是很有用的
20、不错，好书。
21、通俗易懂，内容浅显，作为前端优化的入门书籍挺好的。七年前的书，有些优化方案现在已经过时，当前前端优化情况更复杂，内容介绍的也不够深入。
22、小巧玲珑哈
23、2个小时看完，收获并不太大
24、本书介绍的技术虽然都很常见，但其中对技术产生的原因都作了很详细的分析。认真阅读此书对性能优化的细节考量会有很大收获
25、书中的14条早就看过，单是书中详细的分析还是值得一看。
26、核心思想：极力减少请求数
27、基本概况了前端性能优化的点，不过还不是特别完全，略简单了点，像一个小总结，但不深入。
28、前端视角下的网站性能优化
29、教材是本很好的教材，如果是新手学习的话 耶网科技 建议还是买近一两年的网页设计、网站建设教材吧，因为现在都到WEB3.0、HTML5 + CSS3了
30、前端工程师必看的书，我这个后端就看个大概，缓存相关的知识蛮有用的
31、黄金规则14条，对于网站性能提升很有帮助！就是翻译的一般般
32、书比较老，讲的很多ie5-7的东西，略过了几章服务器端的知识，看完了，nodejs学好应该还会再看一遍，学东西有趣就在于你知道了某个知识你发现很多人居然在用，让人觉得挺神奇的
33、13条法则～
34、内容不错，但是量少且浅。
35、侧重http协议 html的分析与优化
36、毕竟是2008年出的书，2016年读起来感觉很多东西都已经成为了标配（CSS Sprite、将脚本放在底部），可以看到前端领域这么多年的迅猛发展。最后，列举下平常都用过的规则：CSS Sprite、使用Expire过期头、减少HTTP请求、有效利用ETag、将CSS样式表放在头部、将脚本放在底部、使用gzip并精简JS代码、避免在IE中使用CSS表达式、减少重定向（本质还是为了减少HTTP请求）、使用webpack等工具来对前端资源打包和发布

37、第70页，节省那段，有句话”精简JavaScript代码的最流行的工具是JSMin，由我在Yahoo!的一个同事Douglas Crockford开发。“牛逼的人总是跟牛逼的人做同事，随便一个同事就在JS界大名鼎鼎。
38、前端性能上面的优化也是同等重要。
39、书很薄，容易理解和阅读。里面的方法很经典但很有用。
40、知识不错啊
41、有些东西之前也有了解过。但是看到了一些新的特别的信息。不错。
42、写的很用心一本书，如果以后涉及前端一定要会过头看一下
43、经典
44、还想再读
45、内容比较老了，里面提的很多东西现在自动化的构建工具都会帮你做，但是思想还是让人学习到很多，工作后参加实际工程化的东西以后可以再看一次
46、权当了解~书挺薄的，适合去图书馆借来看看~
47、没细看，大致了解了一下各个规则的应用及场景
48、雅虎的14的准则在这儿啊
49、读过好多oreilly的书，没想到工作2年后，觉得最庆幸的就是读了这本。许多Web基础技术都可以在实战中慢慢学习提高，只有优化的思想才是一开始就存在，能主导做更好的中心。有必要再看一次！
50、就一般般吧，作为前端新人，很多老旧的技术基本没在用过。所以，这本书里谈到的优化，已经不再适用目前前端的开发了。
51、看过以后我的那些低级错误就不要犯了，并且可以试着做优化。还拓展了解了下http协议和dns域名解析，总体来说这本书让我对BS的运行原理有了进一步理解。感谢leader的推荐。书不错。。
52、很不错，实用，短小精干
53、进修必备。
54、看完目录基本上就差不多了
55、有点旧，内容还不错

《高性能网站建设指南》

精彩书评

1、另外还有一些感悟：）我大学里学的是化学，很长一段时间里我每日的功课就是不断的做实验，去验证一个可能微不足道的化学原理。教授还告诉我们，一些化学原理仅是某个环境区间内的有效论证，也就是说如果某个参数超过某个极限值，这个化学原理就失效了。我还能回忆起当时我对这样的实验是如何的不屑。多年的工作经验最终让我明白，其实很多事物都是这样具有不确定性，很多时候，我们寻求的就是在特定条件下的最佳实践。从科学到生活都是如此。前端的环境够复杂了吧，众多的浏览器厂商，无数的浏览器版本，还有网络协议及操作系统环境的考虑。为了得到最佳实践，可以想象Steve做了多少求证的工作。感谢Steve，感谢Nate，感谢愿意沉下心来去探原、整理、归纳、总结、分享这些最佳实践的人。
2、YSlow for Firebug 插件: http://developer.yahoo.com/yslow/StanFord课程: http://cs193h.stevesouders.com/DBA Notes的读后笔记: http://www.dbanotes.net/web/high_performance_web_site.htmlYahoo!网站性能最佳体验的34条黄金守则(英): http://developer.yahoo.com/performance/rules.htmlYahoo!网站性能最佳体验的34条黄金守则(中): http://www.dudo.org/article.asp?id=214在线示例： http://stevesouders.com/hpws 示例包含在每章中讨论它们的上下文中。这里也列出一份，以便于查看。 无图片地图的示例（第1章）
http://stevesouders.com/hpws/imagemap-no.php 图片地图的示例（第1章）
http://stevesouders.com/hpws/imagemap.php CSS Sprites的示例（第1章）..
http://stevesouders.com/hpws/sprites.php 内联图片的示例（第1章）
http://stevesouders.com/hpws/inline-images.php 内联CSS图片的示例（第1章）
http://stevesouders.com/hpws/inline-css-images.php 分离脚本的示例（第1章）
http://stevesouders.com/hpws/combo-none.php 合并脚本的示例（第1章）
http://stevesouders.com/hpws/combo.php CDN的示例（第2章） http://stevesouders.com/hpws/ex-cdn.php 无CDN的示例（第2章） http://stevesouders.com/hpws/ex-nocdn.php 无Expires的示例（第3章）
http://stevesouders.com/hpws/expiresoff.php 长久的Expires的示例（第3章）
http://stevesouders.com/hpws/expireson.php 无压缩的示例（第4章）
http://stevesouders.com/hpws/nogzip.html 压缩HTML的示例（第4章）
http://stevesouders.com/hpws/gzip-html.html 压缩所有组件的示例（第4章）
http://stevesouders.com/hpws/gzip-all.html 将CSS放在底部的示例（第5章）
http://stevesouders.com/hpws/css-bottom.php 将CSS放在顶部的示例（第5章）
http://stevesouders.com/hpws/css-top.php 将CSS放在顶部并使用@import的示例（第5章）
http://stevesouders.com/hpws/css-top-import.php 无样式内容的CSS闪烁的示例（第5章）
http://stevesouders.com/hpws/css-fouc.php 将脚本放在中部的示例（第6章）
http://stevesouders.com/hpws/js-middle.php 脚本阻塞下载的示例（第6章）
http://stevesouders.com/hpws/js-blocking.php 将脚本放在顶部的示例（第6章）
http://stevesouders.com/hpws/js-top.php 将脚本放在底部的示例（第6章）
http://stevesouders.com/hpws/js-bottom.php 顶部脚本VS底部脚本的示例（第6章）
http://stevesouders.com/hpws/move-scripts.php 延迟脚本的示例（第6章）
http://stevesouders.com/hpws/js-defer.php 表达式计数器的示例（第7章）
http://stevesouders.com/hpws/expression-counter.php 一次性表达式的示例（第7章）
http://stevesouders.com/hpws/onetime-expressions.php 事件处理器的示例（第7章）
http://stevesouders.com/hpws/event-handler.php 内联JS和CSS的示例（第8章）
http://stevesouders.com/hpws/inlined.php 外部JS和CSS的示例（第8章）
http://stevesouders.com/hpws/external.php 可缓存的外部JS和CSS的示例（第8章）
http://stevesouders.com/hpws/external-cacheable.php 加载后下载的示例（第8章）
http://stevesouders.com/hpws/post-onload.php 动态内联的示例（第8章）
http://stevesouders.com/hpws/dynamic-inlining.php 一般的小脚本的示例（第10章）
http://stevesouders.com/hpws/js-small-normal.php 经过精简的小脚本的示例（第10章）
http://stevesouders.com/hpws/js-small-minify.php 经过混淆的小脚本的示例（第10章）

http://stevesouders.com/hpws/js-small-obfuscate.php 一般的大脚本的示例（第10章）
http://stevesouders.com/hpws/js-large-normal.php 经过精简的大脚本的示例（第10章）
http://stevesouders.com/hpws/js-large-minify.php 经过混淆的大脚本的示例（第10章）
http://stevesouders.com/hpws/js-large-obfuscate.php XMLHttpRequest信标的示例（第11章）
http://stevesouders.com/hpws/xhr-beacon.php 图片信标的示例（第11章）
http://stevesouders.com/hpws/redir-beacon.php 重复脚本——无缓存的示例（第12章）
http://stevesouders.com/hpws/dupe-scripts.php 重复脚本——有缓存的示例（第12章）
http://stevesouders.com/hpws/dupe-scripts-cached.php 重复脚本——10次缓存的示例（第12章）
http://stevesouders.com/hpws/dupe-scripts-cached10.php
3、前端开发可以优化网站剩余的70%~80%性能，这让我看到了前端开发工程师今后的前景，也感到了肩上的压力，推荐前端开发工程师必看。
4、这本译书的副标题是：前端工程师技能精髓。其实副标题应该是：14条让网站加速的“黄金定律”。这本书可谓是字字珠玑，虽内容很薄但层次很高级。它的作者：Steve Souders 说他在服务端开发领域中编程性能已做到极致了，这说明作者在服务端开发的领域已淫浸N久，到了无以复加的境界(可以了解国内的前端开发比国外的差距)。书中的14条“不二”规则是作者多年经验积累所得，这可不是一般人能告诉你的啊。因为前端开发的发展不长，能总结出这些条例规则的人，必须要有作者这样的高度和深度。绪言B HTTP协议的知识内容着实恶补了下。HTTP的相关书在国内是不多见的，所以不错的喔！该书适合所有Web开发人员认真仔细的阅读，不光只是前端开发人员。本文的第一句话就强调译书的副标题有误导读者的味道。而该书唯一的缺点是电工视点博文的书价太高（杀猪价，吓倒一批忠实读者）。还好的是，这本书比它出版的其它书要薄太多了，本人是在书店花三个小时看完的。更多请见blog:http://www.cnblogs.com/georgewing/archive/2009/09/14/1566558.html
5、大家好，我是这本书的译者Anders Liu，欢迎大家到这里提意见
：http://www.cnblogs.com/AndersLiu/archive/2008/04/23/high-performance-web-sites.html
6、每天都忙于应对一个又一个的项目，有时间坐下来看看，也是对自己的一个总结。闲下来看看，这本书还是值得去翻阅一下的。现在看来内容有些陈旧了，但是也可以给人一些启示，我们总是要在时间、质量和成本中找到平衡的。小高兴一下，看了这本书，让我去学了一下YSlow的使用。喜欢看老外写的书，不是因为崇洋媚外，而是老外的书，不仅只是带我进入他的书里，而是带我进入一个世界，一个笔者的世界。
7、这本书太爽了，一口气看完！真的很薄，薄得好，薄得强大。比起那些动不动就几百页的技术书，字字真经，更让人有读的兴趣。我觉得不管是前端工程师还是后台程序员，都可以看看，内容基本都是作者在工作中积累的经验和技巧，很容易引起共鸣，收获良多，往往能让人有“啊，原来可以如此！”的感慨。涉及到的知识大概有Html、CSS、Js、Ajax等前端技术，其实还涉及到一部分PHP和Apache服务器的配置命令等，没学过没关系，你能看懂的！
8、对于前端开发的工程师们，这些建议，确实是应该考虑的，能提高网站的性能，我自己在开发网站的时候，也会注意到里面的事项；我爱足球社区 我自己的网站：http://www.woizuqiu.com很多地方也是需要提高的
9、作为一个性能工程师，这本书是必须的。在我看来，国内的网站能知道这些指标的并不多，更不要说能遵循steve的advice了。。顺便说下，哥哥的这本书有steve的签名哦，当中有个小插曲，steve老大在签的时候说貌似他没看到过有这个版本（中文），额。。。什么情况？
10、小书一本,主要介绍网站在浏览器端的性能相关的影响因素, 脉络清晰,结构合理. 老外书.针对每个性能相关点,都尽量用图示方法来展现其性能差异, 然后提问题, 讲解提升性能的方法.之前没有注意前端的性能问题, 这本书给我带来诸多新思想, 一个小时的阅读,即可读完, 读后有收获, 我喜欢.....第二本<高性能网站建设进阶指南> 似乎也不错哦, 主要针对浏览器加载js的限制/ js 脚本运行性能 / ajax 低消耗/ iframe 的问题/ 图片格式特点及优化方法 提供了很多有用的建议, 另外还有 诸多工具推荐....
11、从14条黄金法则扩展开来每一条都是一支学习树，看完了解简单，一条一条掌握就来日方长了。最大的益处就是让你明白前端优化的准则，自己动手处理的时候有明确目标方向，遇到问题有逻辑上的推断能力，与技术人员提需求是能明确目的，减少沟通成本。
12、这本书出版得比较早了，但是其中的理论包括具体实施方法，在现在web前端开发中任然显得举足轻重。我一直都是在抱怨，在现实的项目环境下，我们前端工程师的生存环境并不是那么好，来自

时间和预算的压力，来自项目后端开发人员对我们页面的嵌套和整合的技术水平参差不齐。要按照我们提供的方案来实现前后端平缓过渡开发，还是很难（至少是在我现在所处的环境中做不到），并不是我做不到这些，但是推动的助力不是一般的大，项目的进度周期，产品线人员培训成本，上级领导的额外要求，我只能做到我认为能做到的那一步，不求最好，只求适应当前。前端工程师一直都在路上，干什么呢？就是把这些当前可行的方案慢慢融入到我们自己的产品中，力求提供优秀的产品体验。研发工程师可能好点，能尽快的去接触这些优秀方案，并进行评估，一致都在路上，致还在敲键盘的同仁们，继续加油！本书值得推荐阅读，能给你比较系统、比较整体的理解网站高性能建设的方案。

13、其实我搞WEB也就2年，接触的正式项目约5个（过小不计）。被迫学习CSS+DIV，然后慢慢的开始接触到一些问题。其实这些都是自主挖掘的。这本书来得太晚，实际上我个人更热爱自己去解决问题而不是靠别人告诉我。所以这本书，我随便翻了下，觉得意义不大。而且有些地方有点过时。做为基础概念巩固，还是有点用的吧。

14、1.减少HTTP请求,把能包在一起的都包在一起　　2.利用CDN,租用或者自建,看情况　　3.添加Expires头,长期缓存　　4.压缩组件传输,除了图片和PDF　　5.CSS扔在HTML最上面(只是看起来速度快,不过USER往往是SB)　　6.JS扔到HTML最下面　　7.尽量少使用CSS表达式　　8.使用外部JS和CSS(缓存)　　9.减少DNS查询(缓存)　　10.精简JS　　11.少使用重定向　　12.避免重复脚本　　13.配置ETag　　14.Ajax缓存以下内容出自：http://www.cnblogs.com/georgewing/archive/2009/09/14/1566558.html《高性能网站建设指南》书评这本译书的副标题是：前端工程师技能精髓。其实副标题应该是：14条让网站加速的“黄金定律”。这本书可谓是字字珠玑，虽内容很薄但层次很高级。它的作者：Steve Souders 说他在服务端开发领域中编程性能已做到极致了，这说明作者在服务端开发的领域已淫浸N久，到了无以复加的境界(可以了解国内的前端开发比国外的差距)。书中的14条“不二”规则是作者多年经验积累所得，这可不是一般人能告诉你的啊。因为前端开发的发展不长，能总结出这些条例规则的人，必须要有作者这样的高度和深度。绪言B HTTP协议的知识内容着实恶补了下。HTTP的相关书在国内是不多见的，所以不错的喔！该书适合所有Web开发人员认真仔细的阅读，不光只是前端开发人员。本文的第一句话就强调译书的副标题有误导读者的味道。而该书唯一的缺点是电工视点博文的书价太高（杀猪价，吓倒一批忠实读者）。还好的是，这本书比它出版的其它书要薄太多了，本人是在书店花三个小时看完的。在《Best Practices for Speeding Up Your Web Site》中共34条规则，前面的14条详细的写在书中，而后面20条规则是书中没有的：及早清除缓存AJAX 请求多用 GET 少用 POST延后加载组件预先加载组件减少 DOM 元素的数目通过域来切分组件iframe 的数目越少越好对 404 说不给 cookie 体积减肥组件用无 cookie 的域通过 DOM 访问越少越好开发灵活的事件处理选 <link> 而不选 @import规避过滤器优化图像优化 CSS Sprites不要在 HTML 中重新改写图像的宽高favicon.ico 越小越好并可缓存所有的组件要永远在25K以下将组件打包到多份文档中

15、从这本书里，让我意识到，除了标准的xHTML/CSS/JavaScript，一个优秀的前端工程师还应该具备什么样的技能呢？1. 精通浏览器工作原理及不同浏览器间的差异2. 理解HTTP协议3. 掌握服务器端编程4. 了解网络环境配置

16、前端优化的技巧指导和实践，对所有的网站都有借鉴的意义。看书的时候建议使用Yslow看看Google之类的网站是如何优化的。

17、自己是一个JavaCoder，主要是做业务实现的。需要写业务逻辑和一些前端代码。通过这本书的阅读，发现其实自己写的很多js，css其实会对网站性能有损失。不过因为是公司内部营业系统，加之网络够快，所以在这方面公司内部并不在乎。书中很多规则，对于我这样的开发者来说是超纲的，例如DNS查找，使用CDN，配置ETag等等。不过学到既是收获。也对前端的世界充满了好奇。

18、个人认为这是一本初中级网站开发人员必须拜读的一本书，里面介绍了很多网站性能优化的简单实用的技巧。这本书内容浅显易懂，书中介绍的每一个小技巧都是一个Web开发人员应该知道了解的。作者Steve Souders最近又出了一本《高性能网站建设进阶指南》听说比这本更加精彩，准备购买拜读

19、不错的书啊，不错的书啊。诺德是专业的互联网应用服务商，知名的网络品牌策划、网站建设、网络营销公司，拥有丰富的网站建设、网站策划、网络营销、网络运营经验。公司核心团队来自国内顶尖的互联　010-51266820/67695835　　http://www.prod.cn　　E-mail : zl@prod.cn网运...

20、对于高性能网站建设的指导方针，是按照Yahoo YSlow的那套规则做的，在具体实施过程中，感觉

有一项ETAG的设置效果不明显，其他几个优化方法进行实施，速度的确提升了一些。现在，Google也推出了类似的工具Page Speed，也非常好用。
21、都是前端技术1.减少HTTP请求,把能包在一起的都包在一起2.利用CDN,租用或者自建,看情况3.添加Expires头,长期缓存4.压缩组件传输,除了图片和PDF5.CSS扔在HTML最上面(只是看起来速度快,不过USER往往是SB)6.JS扔到HTML最下面7.尽量少使用CSS表达式8.使用外部JS和CSS(缓存)9.减少DNS查询(缓存)10.精简JS11.少使用重定向12.避免重复脚本13.配置ETag14.Ajax缓存
22、这本书很薄这本书操作性很强这本书里没有多余的“理论”很短的时间就能翻阅一遍，作为一名后端开发工程师（哈），后面的油水不好榨的时候，就只能打前面的主意了。pagespeed还有yslow现在都提供了很好的工具，然后再参照本书的原则进行改进，改起来比后面容易太多了。
23、对于前台工程师强力推荐。前台水分极大，前台工程师快来搞一下，做出一个让Boss看着开心的数据。Steve Souders老师现在在Stanford 大学计算机科学系教授 CS193H: High Performance Web Sites课程，http://cs193h.stevesouders.com/。一般人我不告诉他。
24、不管是否是前端，此书都有必要读一读！国内一条龙的开发模式不少，什么性能的也谈不上！但是我们努力成为艺匠的道路上，此书会帮你更上一层楼！读完此书，作者指出的观点，让人十分受用，但这只是偏理论的，如何运用到现实中，.NET程序员请拜读此书《构建高性能可扩展ASP.NET网站》
25、我正在阅读这本书，在没有读这本书之前对其中的rule都大致了解了，并且也使用了yahoo developer提供的页面性能分析工具yslow for fire fox，说实话，对中小型的站来说，掌握其中的几项规则就行，有些规则可能用不上，诸如：cdn
26、之前在网上找了本高性能网站建设指南电子书看，感觉很不错。这次看到出了本姐妹版，便决定买下来。翻了一下目录，看了两章关于解决脚本阻塞的，讲得很细致，而且看到了很多很有创新性的尝试并且找到很多很好有效解决方案。而且书上的很多Demo不像上一本只停留理论层面，而是有很多实实在在的代码演示。这次买英文版本来是想着练练英文阅读能力，但读了两章觉得还是蛮辛苦的，不是词句生涩，而是老外写得很罗嗦...或者也可以说他讲得细致吧。对于搞前端的，推荐收藏。
27、看着还行呀，总体而言呢，嗯，有很多收获。里面有很多经验，至于这些经验对您是否有用，就看您自己了。网站还是千差万别的，不好意思，说了一堆废话，凑字数。。。。。。。。。。。。。。。。。。。。。。。。。。。。。
28、如标题，确实很不错，讲解网站需要注意地方，同时也帮助一些网站建设人员需要哪些地方要提高，总体来说，书还写不错。当然也有不足地方，价格有点贵，书上插图好占位置，没安排好排版吧
29、　　以下部分摘自：www.lq580.cn　　这本书主要讲前段的优化，包括css，javascript，网页的头信息等等，让我对前端有了更新的认识。这本书还讲了10多个优化的规则，只是讲的比较少，当然以这本书开始很好，然后根据书上介绍的规则，可以通过其他书籍来加深，我是非常喜欢了，向大家推荐　　作为第一部，该作品还有后续大作“高性能网站建设进阶指南”，两本一起买了把！　　推荐前端人员必看，怎么建设高性能的网站，就是它了。　　虽然网上有很多优化网站的东西，可是还是觉得看书比较安全，哈哈　　这个本必须要有一定的网站建设的基础的人看了还能收益，对于那些只写写代码，网站pv在千万级以下的，看了也是白看。

《高性能网站建设指南》

章节试读

1、《高性能网站建设指南》的笔记-第57页

“ 如果你的网站能够为用户带来高完整缓存率，使用外部文件的收益就更大，如果不大可能产生完整缓存，则内联是更好的选择。

如果javascript和css是外部文件，浏览器就能缓存它们，html文件的大小减小，而且不会增加http请求的数量。 ”

2、《高性能网站建设指南》的笔记-第1页

例子地址：
http://stevesouders.com/hpws

1.减少http请求
2.使用内容发布网络（CDN）
3.为组建增加长久的Expires头
4.压缩脚本和样式表
5.使用link标签将样式表放在文档HEAD中
6.将脚本移到页面底部
7.避免css表达式
8.将javascript和css放到外部文件中
9.用过keep-Alive和较少的域名来减少DNS查找（http1.1可保持永久tcp连接）
10.对javascript源代码进行精简
11.寻找一种避免重定向的方法（记得最后加/）
12.确保脚本只被包含一次
13.配置或移除ETag（多台服务器时不宜用ETag）
14.确保Ajax请求遵守性能知道，尤其应具有长久的Expires头

抓包工具：IBM Page Detailer（http://alphaworks.ibm.com/tech/pagedetailer）

响应时间测量工具：（http://www.gomez.com）

页面性能工具：YSlow（http://developer.yahoo.com/yslow）

3、《高性能网站建设指南》的笔记-第89页

Expires头
浏览器下载组件时，会将他们存储到缓存中。在后续的页面查看中，如果缓存的组件是新鲜的，浏览器就会从磁盘上读取它，避免HTTP请求。如果组件没有过期，那么它就是新鲜的，这取决于Expires头的值。
--
条件GET请求
如果缓存的组件过期了（或者用户明确的重新加载了页面），浏览器在重用它之前必须首先检查它是否仍然有效。这称为一个条件GET请求（Conditional GET Requests）。不幸的是浏览器必须产生这个HTTP请求，执行有效性检查，但这仍比简单的下载所有已经过期的组件效率要高。如果浏览器缓存中的组件是有效的（即它能够和原始服务器上的组件相匹配），原始服务器不会返回整个组件，而

是返回一个“ 304 Not Modified ”状态码。
服务器在检测缓存的组件是否和原始服务器上的组件匹配时有两种方式：
1. 比较最新修改日期
2. 比较实体标签
--
最新修改日期
原始服务器通过Last-Modified响应头来返回组件的最新修改日期。在下一次请求时，浏览器会使用If-Modified-Since头将最新修改日期传回到原始服务器以进行比较。如果匹配，返回304，如果不匹配返回整个组件。
--
实体标签（ETag）
ETag提供了另外一种方式，用于检测浏览器缓存中的组件和原始服务器上的组件是否匹配。ETag是唯一标识了一个组件的一个特定版本的字符串。唯一的格式约束是该字符串必须用引号引起来。
ETag为验证实体提供了比最新修改日期更为灵活的机制。例如，如果实体依据User-Agent或Accept-Language头而改变，实体的状态可以反映在ETag中。
此后如果浏览器必须验证一个组件，会使用If-None-Match头将ETag传回原始服务器。

4、《高性能网站建设指南》的笔记-第20页

“ cdn用于发布静态内容，如图片、脚本、样式表盒flash。提供动态html页面会引入特殊的存储需求——数据库连接、状态管理、验证、硬件和os优化等。这些复杂性超越了cdn的能力范围。另一方面，静态文件更容易存储并具有较少的依赖性。这就是为什么对于地理上分散的用户人群来说，cdn能轻易地得到响应速度上的提高。 ”

5、《高性能网站建设指南》的笔记-第1页

提高网站性能的11个规则：

规则一、减少HTTP请求
方法：
1、CSS Sprites合并图片，减少HTTP请求
2、内联图片。浏览器不会缓存这种图像。dataurl节省了HTTP请求,但是如果这个图像在网页多个地方显示会加大网页的内容，延长下载时间。还有一点IE8以下都不支持这种图像,所以一般不用。
3、合并脚本和样式表，尽量减少js和css的请求数量

规则二、使用内容发布网络
内容发布网络（CDN）是一组分布在多个不同地理位置的Web服务器，用于更加有效地向用户发布内容。

规则三、添加Expires
页面中都包含大量的组件，通过web服务器使用一个长久的expires头，使得这些组件可以被缓存，这样在后续页面中避免不必要的HTTP请求
缺点：要求服务器和客户端的时间严格同步，过期日期需要检查，并提供新日期
换一种方式：cache-control使用max-age指令

规则四、压缩组件
方法：gzip压缩

规则五、将样式表放在顶部

规则六、将脚本放在底部，因为在下载脚本时浏览器会阻塞并行下载

规则七。避免CSS表达式。表达式的问题在于对其进行的求值【频率比人们期望的要高，它们不只在页面呈现和大小改变时求值，当页面滚动、甚至用户鼠标在页面上移过时都要被求值。

规则八、使用外部的JS和CSS

规则九、减少DNS查找
方法：通过使用Keep-Alive和较少的域名来减少DNS查找

规则十、精简JS，相当于压缩，且会除去注释及空格

规则十一、避免重定向。重定向会延迟整个HTML文档的传输

规则十二、移除重复脚本

规则十三、配置或移除Etag

规则十四、使用Ajax可缓存。确保ajax请求遵守性能指导，尤其应具有长久的expires头

6、《高性能网站建设指南》的笔记-第17页

减少HTTP请求
以前认为优化只要将后台做好，那么网站的响应速度应该自然没有问题，现在看来这种想法有点太想当然了。

7、《高性能网站建设指南》的笔记-第3页

这本书和进阶篇 基本上主要章节都是在讲http协议相关的前段优化技巧，还有部分程序框架上的设计指南。值得一读。

8、《高性能网站建设指南》的笔记-第4页

“ 所有这些网站在获取html文档时，花费的时间都不到总响应时间的20%－－－－在进行优化时，关键是剖析当前的性能，找到哪里能够获得最大的优化。

只有10%－20%的最终用户响应时间花在了下载html文档上，其余的80%－90%时间花在了下载页面中的所有组件上。 ”

9、《高性能网站建设指南》的笔记-第1页

压缩：浏览器使用Accept-Encoding
服务器：Content-Encoding

条件GET请求：如果浏览器在缓存中保留了组件的事个副本，但并不确定它是否还有效，就会生成一个条件GET请求.

304 Not Modified
Expires头：明确指出浏览器是否可以使用组件副本的缓存

Keep-Alive:持久连接，解决每次请求都要打开socket问题
Connection:keep-alive
Connection:close

10、《高性能网站建设指南》的笔记-第38页

“进度指示器有三个主要优势——它们让用户知道系统没有奔溃，只是正在为他或她解决问题；它们指出了用户大概还需要等多久，以便用户能够在漫长的等待中做些其他事情；最后，它们能给用户提供一些可以看的东西，使得等待不再那么无聊。最后一点优势不可低估，这也是为什么推荐使用图形进度条而不是仅仅以数字形式显示预期的剩余时间。”

11、《高性能网站建设指南》的笔记-第1页

《High Performance Web Sites》读书笔记
http://www.cainiao8.com/web/high_performace_web_sites.html
High Performance Web Sites
http://stevesouders.com/hpws/rules.php
规则1 减少http请求
图片地图 map
Server-side image maps
Client-side image maps
内联图片 Inline Images
ie不支持
php函数 file_get_contents
对文件进行精简ch10
规则2 使用内容分发网络 CDN啦
Akamai是CDN业界的领头羊
免费CDN服务可用 如Globule，CoDeeN，CoralCDN
无论如何不要使用HTTP重定向来将用户指向到本地服务器 ch11
规则3 添加Expires头
Max-Age和mod_expires
HTTP 1.1 引入了Cache-Control头来克服Expires头的限制
mod_expires Apache模块 http://httpd.apache.org/docs/2.0/mod/mod_expires.html
空缓存vs完整缓存
Empty Cache vs Primed Cached
html文档不应该使用长久的expires头。
Caching Tutorial for Web Authors and Webmasters
http://www.mnot.net/cache_docs/
最有效的解决方法是修改其所有连接，这样全新的请求将从原始服务器下载。
规则4 压缩组件
gzip编码压缩HTTP响应包
web客户端可以通过HTTP请求中的Accept-Encoding头来标识对压缩的支持
Accept-Encoding: gzip,deflate
web服务器通过响应中的Content-Encoding头来通知Web客户端
Content-Encoding:gzip
压缩what

js,css,html,xml,json
图片&pdf nope 已经被压缩了
apache 1.3使用mod_gzip
apache 2.x使用mod_deflate
代理缓存
Proxy Caching
边缘情形 Edge Cases
默认情况下 ETag 不能反映出内容是否被压缩，因此代理可能会想浏览器提供错误的内容。 此问题在Apache的缺陷数据库http://issues.apache.org/bugzilla/show_bug.cgi?id=39727 中有所描述。 最好的解决办法是禁用ETag。 ->ch13
规则5 将样式表放在顶部 （使用link方式放在head
将css放在底部会白屏 ie
规则6 将脚本放在底部
脚本阻止下载
如果它们之间存在着依赖关系，不按顺序执行就会导致js错误。
最差情况 将脚本放在顶部；最佳情况 将脚本放在底部
正确的放置
另外一种建议： 使用延迟Defferred脚本 differ属性表明脚本不包含document.write浏览器得到这一线索就可继续进行呈现。
如果一个脚本可以延迟 那么它一定可以一到页面底部。
规则7 避免css表达式
classname: expression
(原来这才是css表达式... a js 表达式)
规则8 使用外部js和css
主页 Homepages
页面查看 ： 每月很高的页面查看数量
空缓存vs完整缓存：
组件重用： 重用率低
首页倾向于使用内联。
两全其美 the best of both worlds
1. 加载后下载 post-onload download
通过onload事件
2. 动态内联 dynamic inlining
规则9 减少DNS查找 reduce DNS lookups
通常 浏览器查找一个给定主机名的ip地址要花费20~120毫秒
DNS缓存和TTL DNS Caching and TTLs
影响DNS缓存的因素 Factors Affecting DNS Caching
TTL Time-to-live http协议中的Keep-Alive特性可以同时覆盖TTL和浏览器的时间限制
TTL 建议值一天 一般没那么高 故障转移神马的
减少DNS查找
规则10 精简JavaScript minify JS
精简 minification
Obfuscation 书上翻译是混淆 模糊处理更好吧坑爹啊
精简不会出问题 obfuscation会
The Savings
JSMin http://crockford.com/javascript/
Dojo Compressor 已经改名为ShrinkSafe http://dojotoolkit.org/docs/shrinksafe
锦上添花 Icing on the Cake
inline scripts 精简

Gzip and Minification
minifying css
规则11 避免重定向 avoid redirects
缺少斜线 发送重定向是很多web服务器的默认行为，包括apache
连接网站 connecting web sites
Alias、mod_rewrite、DirectorySlash
tracking internal traffic referer 日志
tracking outbound traffic
规则12 移除重复脚本
规则13 配置ETag
组件如何缓存和确认。。。
ETag 实体标签 实体=组件
如果ETag匹配 就会返回304状态码 not-modified，使响应减少
ETag问题
使用了服务器集群的话，浏览器在一台服务器上获取组件，之后，又从另外一台不同服务器发起条件get请求时，ETag是不会匹配的
默认情况下，对于拥有多台服务器的网站，Apache和IIS向ETag中潜入的数据都会大大地降低有效性验证的成功率
ETag用还是不用
从ETag中移除ChangeNumber或完全移除ETag可以避免当数据已经位于浏览器缓存中进行不必要的和低效的下载。
规则14 使Ajax可缓存
优化ajax请求 optimizing Ajax Requests
3 使响应可缓存!important 4 压缩组件 9 减少DNS查找 10 精简JS 11 避免重定向 13 ETag
google spreadsheets中 请求有的使用了XMLHttpRequest 也有的使用了IFrame
ch15 Deconstruction 10 Top Sites
http请求图表 IBM Page Detailer http://alphaworks.ibm.com/tech/pagedetailer
响应时间 Gomez web监视服务 http://www.gomez.com
firebug分析各个页面中的javascript 和css yslow

12、《高性能网站建设指南》的笔记-第18页

“ 如果应用程序web服务器（application web server）离用户更近，则一个http请求的响应时间将缩短。另一方面，如果组件web服务器（component web server）离用户更近，则多个http请求的响应时间将缩短。与其开始重新应用程序这一艰难任务，以便将应用程序web服务器，不如首先将组件web服务器分散开。这不仅能达到响应时间大幅减少的目的，还很容易实现。 ”

13、《高性能网站建设指南》的笔记-第10页

改善响应时间的最简单途径就是减少组件（图片，脚本，样式表，flash等）的数量，并由此减少HTTP请求的数量

从页面中移除组件的想法会引发性能和产品设计之间的矛盾，解决办法：
图片地图：
允许在图片上放置多个url,目标url的选择取决于用户点击了图片的哪一块。

Css Sprites:
使用Css 整合图片
#navbar span {

```
 width:31px;
 height:31px;
 display:inline;
 float:left;
 background-image:url(/images/spritebg.gif?t=1389780616);
}
.home    { background-position:0 0; margin-right:4px; margin-left: 4px;}
.gifts   { background-position:-32px 0; margin-right:4px;}
.cart    { background-position:-64px 0; margin-right:4px;}
.settings { background-position:-96px 0; margin-right:4px;}
.help    { background-position:-128px 0; margin-right:0px;}
```

内联图片：
内联用法可能有些浏览器不支持
  data:[<mediatype>][;base64],<data>

```
&lt;img border=0
src="
t7e1uelMefn1ufn3ufn5+fv3u+MAO/v5+/v7/fGCPf35/f37//nY/////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////yH5BAEAAAAALAAAAAAfAB8AAAj+AAEI
HEhQAoICAxImLIBAAsGHEAc6KEDAgAIJGCU4UGCgooOIESUYYJjxgQSTGE0eNOAQpMCJBhycnGmyJ
s2RH0E6INAQpc2ZNDEiIJDzoQSSP5MCrXmwJUEDMWs6eCDTQYQDWKdOPTkV6sOdD8KKFRvhAlYK
Bx4IOCBA7M6iRxE4mEt3btkDDBgcQLtWQN2hLd8uWOBgcOG7DChQ0IuWrYLChAk8AGCwwODLgxEr
VsyYrYMEl0nGVUC6tObNnPceSFBaQVMJAxC4lo3gNOrUaFnTHoAxNm3XVxPfRq139e8BEGAjWD5bgIA
Lw287T8AcAXLly2kjOACdc17higXSIKDO/Lpv7Qq4bw7APgBq8eOzX69InrZ6xe3dbxZffyTGkb8tdx8F+b0Xn2
sFsCSBAgTM5lp63RHYnoHUudZgRgkGOGCB+43nGk4OGcQTabKx5dyJKJ7ImoUNCaRRAZYN1ppsrT3Y2
gIwyjSQBAtUpABml/0IJGYd6VjQUDH9uBFkGxGm5I8dPQaRUAQUMBdhhBV25ZYUJZBcSAtSJBddWZZ5
UAGPOTXlgkNVOSZdBxEwIkYu7VhYnAol5GaadRqF0Uaz0TgXnX2umVFyGakJUUAAADs="&gt;
```

合并脚本和样式表：
  引用分块的样式表和脚本文件会产生http请求
  可将多个样式表合成一个

规则2:使用内容发布网络(CDN)
如果http服务器与用户越近，响应越快，同样，组件服务器距离用户越近，响应也越快。

14、《高性能网站建设指南》的笔记-第1页

记录下这边书

15、《高性能网站建设指南》的笔记-第1页

《高性能网站建设指南》

这个价格的书，这些内容，确实性价比不怎么样，很多东西日常都在用到，只是没有归类整理罢了，而且其中很多的翻译感觉也欠缺水平。

文章最后提出的使用IBM page detailer，仅支持IE，专业点的还是用Wireshark NB些，Yslow这个工具集成到Firebug中确实不错。

个人感觉影响性能的最大的包括三个方面：
1. 连接资源消耗，包括HTTP请求，组件大小；
2. 本地缓存；
以上两个方面算是硬性指标，第三个，也就是css、js的位置， ajax cache 这些，对页面访问时间影响基本不大，但是对用户体验是有影响的。
等有时间了把这些个东西整理下。

16、《高性能网站建设指南》的笔记-第1页

重温yslow的13条建议（目前已与时俱进到34条），受益仍旧匪浅，特别是涉及到etag的部分，“ etag在某些情况下会损坏性能 ”

17、《高性能网站建设指南》的笔记-第33页

Web服务器可以告诉代理服务器根据一个或多个请求头来改变代理缓存的响应。由于压缩的决定是基于Accept-Encoding请求头的，因此需要在服务器的Vary响应头中包含Accept-Encoding。

18、《高性能网站建设指南》的笔记-第21页

规则3添加Expires头：
使用Expires 头来缓存组件
Cache-Control:public,max-age=172800
动态修改文件名

规则4:压缩组件
Accept-Encoding:gzip
压缩可减少文件传输体积，加快响应
压缩什么：xml,json,脚本，样式表等文本内容，不包括图片，pdf
压缩成本：服务器花cpu资源，客户端解压缩
配置apache 压缩

规则5：用link将样式表放在顶部<head><link rel="" href></head>
方式1：<link rel="stylesheet" href="style.css"/>
建议使用link方式
方式2：<style>
@import url('style.css')
@import url('style.css')
@import url('style.css')
....
</style>
必须放在其它规则之间，且容易产生白屏

白屏和无样式内容的css闪烁

规则6：脚本移动到页面底部
脚本放在顶部，脚本阻塞：会阻塞对其后面的内容的呈现及后面的组件的下载
移动脚本要注意的问题：例如存在document.write在页面写内容，就不能移到底部

规则7：避免CSS表达式
一次性表达式和事件处事器

规则8：使用外部javascritp和css
从几个方面考虑是使用内联还是外联:
页面查看数量，缓存，组件重用

规则9：减少DNS查找
DNS缓存,TTL
使用keep-alive和较少的域名来减少DNS查找

规则10：精简javascript
精简：去掉注释和空白，换行特等..

规则11：避免重定向
寻找一种避免重定向的方法

规则12：移除重复脚本
重复的脚本，重复的http请求和重复的响应时间

规则13：配置Etag
实体标签ETag:浏览器和服务器之间确认缓存组件的一种机制
检测缓存组件是否匹配：
比较最新修改日期，比较实例标签

规则14：使用AJAX可缓存
确保Ajax请求遵循性能指导，尤其应当具有长期Expires头

19、《高性能网站建设指南》的笔记-第1页

为何要关注前端性能：1.如果将后端响应时间缩短一半，则整体响应时间只能减少5%-10%
而关注前端，同样缩短一半，则整体响应时间可减少40%-45%

2.改进前段只需要较少的时间和资源。　　性能黄金法则：
只有10%～20%的最终用户响应时间花在了下载HTML文档上，其余80%～90%时间花在了下载页面的组件上

20、《高性能网站建设指南》的笔记-第21页

使用内容发布网络
实际情况是，我测试书中给的URL，没有使用CDN的比使用CDN的速度快……
应该是网络问题吧，哈哈。

《高性能网站建设指南》

21、《高性能网站建设指南》的笔记-1-14

1. 减少HTTP请求(使用CSS Sprite,CSS样式表中以)的方式内联图片
2. 使用CDN实现地理位置上离用户更近
3. 添加Expires头，添加Cache-Control中的max-age 利用缓存
4. gzip压缩文档、样式表、脚本(Accept-Encoding:gzip)
对代理缓存设置Vary:Accept-Encoding,User-Agent
5. 使用link标签将样式表放在顶部，使得内容逐步呈现。避免IE中的白屏、其他浏览器的闪烁
6. 脚本文件尽量放在底部，以免阻塞其他组件的预先下载
7. 避免CSS expression，使用one-time expression或者window.onresize等事件
例子：http://stevesouders.com/hpws/onetime-expressions.php
8. 外链js和css得到缓存的机会
9. 减少DNS查找（修改TTL值，使用keep-alive）
10. 精简JS
11. 避免重定向(301 moved permanently,302 moved temporarily,304 not modified)
避免缺少斜线的链接结尾
12. 移除重复脚本
13. 配置/移除ETag减少对服务器性能的影响。（因为不同服务器上ETag不同，导致对缓存的使用率下降）
14. 优化Ajax（可以用时间戳哦~~）
综合来讲，就是减少HTTP请求、减小被请求内容的大小。使用缓存、注意组件(CSS/JS)的位置对页面呈现的影响。

《高性能网站建设指南》

版权说明