**[Web 前端优化最佳实践之 内容篇](http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_content.html)**

作者: [Fenng](http://www.dbanotes.net/) | 可以转载, 但必须以超链接形式标明文章原始出处和作者信息及[版权声明](http://creativecommons.org/licenses/by/2.5/cn/)  
网址: <http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_content.html>

Yahoo! 的 Exceptional Performance team 在 Web 前端方面作出了卓越的贡献。广为人知的[优化规则](http://developer.yahoo.com/performance/rules.html)也由 13 条到 14 条，再到 20 条，乃至现在的 [34 条](http://developer.yahoo.com/performance/)--真是与时俱进啊。最新的 34 条也针对不同的角度做了分类。

面向内容的优化规则目前有 10 条。

**1. 尽量减少 HTTP 请求 (Make Fewer HTTP Requests)**

作为第一条，可能也是最重要的一条。根据 Yahoo! 研究团队的数据分析，有很大一部分用户访问会因为这一条而取得最大受益。有几种常见的方法能切实减少 HTTP 请求：

* 1) **合并文件**，比如把多个 CSS 文件合成一个；
* 2) **CSS Sprites** 利用 CSS background 相关元素进行背景图**绝对**定位；参见：[CSS Sprites: Image Slicing's Kiss of Death](http://alistapart.com/articles/sprites)
* 3) **图像地图**
* 4) **内联图象** 使用 [data: URL scheme](http://tools.ietf.org/html/rfc2397) 在实际的页面嵌入图像数据.

**2. 减少 DNS 查找 (Reduce DNS Lookups)**

必须明确的一点，DNS 查找的开销是很大的。另外，我倒是觉得这是 Yahoo! 所有站点的通病，Yahoo！主站点可能还不够明显，一些分站点，存在明显的类似问题。对于国内站点来说，如果过多的使用了站外的 Widget ，也很容易引起过多的 DNS 查找问题。

**3. 避免重定向 (Avoid Redirects)**

不是绝对的避免，尽量减少。另外，应该注意一些不必要的重定向。比如对 Web 站点子目录的后面添加个 / (Slash) ，就能有效避免一次重定向。http://www.dbanotes.net/arch 与 http://www.dbanotes.net/arch**/** 二者之间是有差异的。如果是 Apache 服务器，通过配置 Alias 或mod\_rewrite 或是 DirectorySlash 能够消除这个问题。

**4. 使得 Ajax 可缓存 (Make Ajax Cacheable)**

响应时间对 Ajax 来说至关重要，否则用户体验绝对好不到哪里去。提高响应时间的有效手段就是 Cache 。其它的一些优化规则对这一条也是有效的。

**5. 延迟载入组件 (Post-load Components)**

**6. 预载入组件 (Preload Components)**

上面两条严格说来，都是属于**异步**这个思想灵活运用的事儿。

**7. 减少 DOM 元素数量 (Reduce the Number of DOM Elements)**

**8. 切分组件到多个域 (Split Components Across Domains)**

主要的目的是提高页面组件并行下载能力。但不要跨太多域名，否则就和第二条有些冲突了。

**9. 最小化 iframe 的数量 (Minimize the Number of iframes)**

熟悉 SEO 的朋友知道 iframe 是 SEO 的大忌。针对前端优化来说 iframe 有其好处，也有其弊端，一分为二看问题吧。

**10. 杜绝 http 404 错误 (No 404s)**

对页面链接的充分测试加上对 Web 服务器 error 日志的不断跟踪能有效减少 404 错误，亦能提升用户体验。值得一提的是，CSS 与 Java Script 引起的 404 错误因为定位稍稍"难"一点而往往容易被忽略。

**[Web 前端优化最佳实践之 Server 篇](http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_server.html)**

作者: [Fenng](http://www.dbanotes.net/) | 可以转载, 但必须以超链接形式标明文章原始出处和作者信息及[版权声明](http://creativecommons.org/licenses/by/2.5/cn/)  
网址: <http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_server.html>

Web 前端优化最佳实践第二部分面向 Server 。目前共计有 6 条实践规则。【注，这最多算技术笔记，查看最原始内容，还请访问：[Exceptional Performance : Best Practices for Speeding Up Your Web Site](http://developer.yahoo.com/performance/rules.html) 】

**1. 使用 CDN (Use a Content Delivery Network)**

国内 CDN 的普及还不够。不过我们有**独特的电信、网通之间的问题**，如果针对这个作优化，基本上也算能收到 CDN 或类似的效果吧(假装如此)。【Tin 说国内 CDN 用的挺多，看看 CDN厂商的市场就知道了，还没走入寻常百姓家】

**2. 添加 Expires 或 Cache-Control 信息头 (Add an Expires or a Cache-Control Header)**

各个浏览器都有针对的方案, Apache 例子【注意：下面的说明例子还不够精细，具体的环境上还要加一些调整】:

ExpiresActive On

ExpiresByType image/gif "modification plus 1 weeks"

Lighttpd 启用 mod\_expire 模块 后：

$HTTP["url"] =~ "\.(jpg|gif|png)$" {

expire.url = ( "" => "access 1 years" )

}

Nginx 例子参考:

location ~\* \.(jpg|gif|png)$ {

if (-f $request\_filename) {

expires max;

break;

}

}

**3. 压缩内容 (Gzip Components)**

对于绝大多数站点，这都是必要的一步，能有效减轻网络流量压力。或许有人担心对 CPU 压缩对于 CPU 的影响，放心大胆的整吧，没事儿。Nginx 例子：

gzip on;

gzip\_types text/plain text/html text/css ext/javascript;

另外参见：

[IIS 如何启用 Gzip 压缩?](http://www.dbanotes.net/web/iis_gzip_compression.html)

**4. 设置 Etags (Configure ETags)**

对于 [Etag](http://www.dbanotes.net/web/http_11_etag_lastmodified.html)，可能是多数网站维护者都会忽略的地方。在这一系列优化规则出现之前，可能互联网上绝大多数站点都对这个问题忽略了。当然，Etag 对多数站点性能的影响并不是很大。除非是面向 RSS 的网站。【看到有朋友批评说写的简略，并且说 IE 不支持 ETag。明确说一下：IE 支持 ETag，倒是使用 IIS 要注意相关 Etag Bug。】

补充：我的意思是"很多网站在不注意的情况下都是打开 Etag 的，而没有网站关心如何用，消耗资源而不知。并不是说 Etag 不好，合理利用 Etag ，绝对能取得很好的收益.

**5. 尽早刷新 Buffer (Flush the Buffer Early)**

对这一条，琢磨了半天，貌似还是**异步**的思路。能更好的提升用户体验?

**6. 对 AJAX 请求使用 GET 方法 (Use GET for AJAX Requests)**

XMLHttpRequest POST 要两步，而 GET 只需要一步。但要注意的是在 IE 上 GET 最大能处理的 URL 长度是 2K。

**[Web 前端优化最佳实践之 Cookie 篇](http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_server_cookie.html)**

作者: [Fenng](http://www.dbanotes.net/) | 可以转载, 但必须以超链接形式标明文章原始出处和作者信息及[版权声明](http://creativecommons.org/licenses/by/2.5/cn/)  
网址: <http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_server_cookie.html>

Web 前端优化最佳实践第三部分面向 Cookie 。目前只有 2 条实践规则。

**1. 缩小 Cookie (Reduce Cookie Size)**

Cookie 是个很有趣的话题。根据 [RFC 2109](http://www.ietf.org/rfc/rfc2109.txt) 的描述，每个客户端最多保持 300 个 Cookie，针对每个域名最多 20 个 Cookie (实际上多数浏览器现在都比这个多，比如 Firefox 是 50 个) ，每个 Cookie 最多 4K，注意这里的 4K 根据不同的浏览器可能不是严格的 4096 。别扯远了，对于 Cookie 最重要的就是，尽量控制 Cookie 的大小，不要塞入一些无用的信息。

**2. 针对 Web 组件使用域名无关性的 Cookie (Use Cookie-free Domains for Components)**

这个话题在此前针对 [Web 图片服务器](http://www.dbanotes.net/web/web_image_server.html)的讨论中曾经提及。这里说的 Web 组件(Component)，多指静态文件，比如图片 CSS 等，Yahoo! 的静态文件都在 yimg.com 上，客户端请求静态文件的时候，减少了 Cookie 的反复传输对主域名 (yahoo.com) 的影响。

从这篇 [When the Cookie Crumbles](http://yuiblog.com/blog/2007/03/01/performance-research-part-3/) 能看出，MySpace 和 eBay 的 Cookie 都不小的，想必是对用户行为比较关心。[eBay 前不久构造了 Personalization Platform](http://www.dbanotes.net/database/ebay_personalization_platform_mysql.html) ，就是从 Cookie 的限制中跳出来。

**[Web 前端优化最佳实践之 CSS 篇](http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_css.html)**

作者: [Fenng](http://www.dbanotes.net/) | 可以转载, 但必须以超链接形式标明文章原始出处和作者信息及[版权声明](http://creativecommons.org/licenses/by/2.5/cn/)  
网址: <http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_css.html>

Web 前端优化最佳实践第四部分面向 CSS。目前共计有 6 条实践规则。另请参见 Mozilla 开发者中心的文章：[Writing Efficient CSS](http://developer.mozilla.org/en/docs/Writing_Efficient_CSS)

**1. 把 CSS 放到代码页上端 (Put Stylesheets at the Top)**

官方的解释我觉得多少有点语焉不详。这一条其实和**用户访问期望**有关。CSS 放到最顶部，浏览器能够有针对性的对 HTML 页面从顶到下进行解析和渲染。没有人喜欢等待，而浏览器已经考虑到了这一点。

**2. 避免 CSS 表达式 (Avoid CSS Expressions)**

个人认为通过 CSS 表达式能做到的事情，通过其它手段也同样能做到而且风险更小一些。

**3. 从页面中剥离 JavaScript 与 CSS (Make JavaScript and CSS External)**

剥离后，能够有针对性的对其进行单独的处理策略，比如压缩或者缓存策略。

**4. 精简 JavaScript 与 CSS (Minify JavaScript and CSS)**

如果没有 JavaScript 与 CSS 可能更好。但，这是不可能的，SO，尽量小点吧。语法能简写的简写。

**5. 使用 <link> 而不是@importChoose <link> over @import**

在 IE 中 @import 指令等同于把 link 标记写在 HTML 的底部。而这与第一条相违背。

**6. 避免使用Filter (Avoid Filters)**

**[Web 前端优化最佳实践之 JavaScript 篇](http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_javascript.html)**

作者: [Fenng](http://www.dbanotes.net/) | 可以转载, 但必须以超链接形式标明文章原始出处和作者信息及[版权声明](http://creativecommons.org/licenses/by/2.5/cn/)  
网址: <http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_javascript.html>

Web 前端优化最佳实践之 JavaScript 篇，这部分有 6 条规则，和 CSS 篇 重复的有几条。前端优化最佳实践，最重要的还是"实践"，要理解这东西容易得很，关键是要去"实践"，去"执行"，去"反馈"，去获取受益。

**1. 脚本放到 HTML 代码页底部 (Put Scripts at the Bottom)**

当一个脚本在下载的时候，浏览器干不了其它的事儿(串行了)。所以，把它扔到最后面去处理。对于一些功能性的脚本，可能实现起来有些两难。不过对于国内网站来说，有很多使用 Google Analytics 服务进行网站数据分析的。这这一点来说，绝对可行的建议，放到页面最底下。

**2. Make JavaScript and CSS External**

参见 [CSS 篇](http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_css.html)的描述

**3. 精简 JavaScript 与 CSS (Minify JavaScript and CSS)**

参见 [CSS 篇](http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_css.html)的描述

**4. 移除重复脚本 (Remove Duplicate Scripts)**

对于一些历史遗留站点或是论坛类的网站来说，这倒是比较常见的。接手维护人前后变化过多，每个人都有自己的一套。这就会带来一些潜在的麻烦。

**5. 减少 DOM 访问 (Minimize DOM Access)**

有三条指导建议:

* 缓存已经访问过的元素 (Cache references to accessed elements)
* "离线"更新节点, 再将它们添加到树中 (Update nodes "offline" and then add them to the tree)
* 避免使用 JavaScript 输出页面布局--应该是 CSS 的事儿 (Avoid fixing layout with JavaScript)

**6. Develop Smart Event Handlers**

**[Web 前端优化最佳实践之 图象篇](http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_image.html)**

作者: [Fenng](http://www.dbanotes.net/) | 可以转载, 但必须以超链接形式标明文章原始出处和作者信息及[版权声明](http://creativecommons.org/licenses/by/2.5/cn/)  
网址: <http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_image.html>

Web 前端优化最佳实践第六部分面向 图片(Image)，这部分目前有 4 条规则。在最近的 Velocity 2008 技术大会上，Yahoo! 的 Stoyan Stefanov 做的 [Image Optimization: How Many of These 7 Mistakes Are You Making](http://en.oreilly.com/velocity2008/public/schedule/detail/2405) 也非常有参考价值。结合一起说一下。

**1. 优化图片 (Optimize Images)**

使用 GIF 、JPG 还是 PNG 格式的图片? 尽可能的使用 PNG 格式的图片，更多的功能，更小的尺寸(与 GIF 相比)。

对于 PNG 图片，考虑用 [Pngcrush](http://pmt.sourceforge.net/pngcrush/) 或类似的工具进行优化。常见的工具如下表:

* [pngcrush](http://pmt.sourceforge.net/pngcrush/) http://pmt.sourceforge.net/pngcrush/
* [pngrewrite](http://www.pobox.com/~jason1/pngrewrite/) http://www.pobox.com/~jason1/pngrewrite/
* [OptiPNG](http://www.cs.toronto.edu/~cosmin/pngtech/optipng/) http://www.cs.toronto.edu/~cosmin/pngtech/optipng/ (refer: [教程](http://linuxtoy.org/archives/optipng.html))
* [PNGOut](http://advsys.net/ken/utils.htm)http://advsys.net/ken/utils.htm

另请参见: [Five Tips For the Effective Use of PNG Images](http://blog.httpwatch.com/2008/05/29/five-tips-for-the-effective-use-of-png-images/)

对 JPEG 图片的优化工具：

* [jpegtran](http://jpegclub.org/) (http://jpegclub.org/)

必需要强调的是，图片设计的同学啊，请考虑设计**面向 Web 的图片**，不要动不动就设计超过可接受尺寸之外大家伙，这应该是一种习惯，而不是什么高超的技能，只需要记住就成了。

**2. 使用 CSS Sprites 技巧对图片优化 (Optimize CSS Sprites)**

之前提到过，简单的说就是"利用 CSS background 相关元素进行背景图绝对定位"，把多次HTTP 调用变为一次调用，更多参考：[CSS Sprites: Image Slicing's Kiss of Death](http://alistapart.com/articles/sprites)

补充一下：对于这个技巧我曾经见到有人滥用的。把多个背景图片揉成一个，减少　HTTP 调用，这是一个很好的思路。但一定要记住这个大图片不能太"重"，我看到过 100 多K 的背景图。一个图片就把整个网站拖得很慢。比较好的例子可以参考[雅虎关系的这个图](http://cn.yimg.com/i/sns/img/ysred_1_0_6.png).

更新：使用 CSS Sprites 的一个潜在的副作用是客户端将消耗更多内存([参考](http://blog.vlad1.com/2009/06/22/to-sprite-or-not-to-sprite/))。

**3. 不要在 HTML 中使用缩放图片 (Don't Scale Images in HTML)**

更多的时候，可能是因为偷懒而没有制作合适大小的图片，如果是批量处理图片的话，可能一条 [ImageMagic](http://www.imagemagick.org/) 命令（convert ）就能搞定 。必须提及的是，看到太多的对图片拉伸很难看的页面，救救这些页面!

**4. 用更小的并且可缓存的 favicon.ico (Make favicon.ico Small and Cacheable)**

更小，可缓存，这两条可能都不是问题。问题是，**太多站点根本没有 favicon.ico** 。有的时候，判断独立域名的 Blog 是否专业，基本看一下是否有 favicon.ico 就差不多了。

**[Web 前端优化最佳实践之 Mobile(iPhone) 篇](http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_server_mobile.html)**

作者: [Fenng](http://www.dbanotes.net/) | 可以转载, 但必须以超链接形式标明文章原始出处和作者信息及[版权声明](http://creativecommons.org/licenses/by/2.5/cn/)  
网址: <http://www.dbanotes.net/web/best_practices_for_speeding_up_your_web_site_server_mobile.html>

Web 前端优化最佳实践最后一部分是针对移动应用的，其实只是针对 iPhone 的，目前只有两条规则。

**1. 单个数据对象小于 25K (Keep Components under 25K)**

这个似乎只是[针对 iPhone 研究](http://yuiblog.com/blog/2008/02/06/iphone-cacheability/)的。建议保持单个 Web 数据对象在 25 K 以下。为什么是 25K? Apple 官方信息指出可缓存到内存中的 Web 对象最大支持到 10M，但经过测试，发现也就是 25K 左右。

iPhone 在市场上的优异表现，让 Web 人员不得不考虑如何针对其进行优化。相信这部分内容也在不断变化中。

**2. Pack Components into a Multipart Document**

把Web 页面组件打包成一个多部分组成的文档。其目的是减少 HTTP 请求。对这部分语焉不详，等待后续更新吧。