Yahoo!团队实践分享：网站性能优化的34条黄金守则（一）内容

Yahoo!的 Exceptional Performance 团队为改善 Web 性能带来最佳实践。他们为此进行了一系列的实验、开发了各种工具、写了大量的文章和博客并在各种会议上参与探讨。最佳实践的核心就是旨在提高网站性能。

Excetional Performance 团队总结出了一系列可以提高网站速度的方法。可以分为7大类34条。包括内容、服务器、cookie、CSS、JavaScript、图片、移动应用等七部分。

其中内容部分一共十条建议：

一、内容部分

- 尽量减少 HTTP 请求
- 减少 DNS 查找
- 避免跳转
- 缓存 Ajxa
- 推迟加载
- 提前加载
- 减少 DOM 元素数量
- 用域名划分页面内容
- 使 frame 数量最少
- 避免404错误

1、尽量减少 HTTP 请求次数

终端用户响应的时间中，有80%用于下载各项内容。这部分时间包括下载页面中的图像、样式表、脚本、Flash 等。通过减少页面中的元素可以减少 HTTP 请求的次数。这是提高网页速度的关键步骤。

减少页面组件的方法其实就是简化页面设计。那么有没有一种方法既能保持页面内容的丰富性又能达到加快响应时间的目的呢？这里有几条减少 HTTP 请求次数同时又可能保持页面内容丰富的技术。

合并文件是通过把所有的脚本放到一个文件中来减少 HTTP 请求的方法，如可以简单地把所有的 CSS 文件都放入一个样式表中。当脚本或者样式表在不同页面中使用时需要做不同的修改，这可能会相对麻烦点，但即便如此也要把这个方法作为改善页面性能的重要一步。

CSS Sprites 是减少图像请求的有效方法。把所有的背景图像都放到一个图片文件中，然后通过 CSS 的 *background-image* 和 *background-position* 属性来显示图片的不同部分；

图片地图是把多张图片整合到一张图片中。虽然文件的总体大小不会改变，但是可以减少 HTTP 请求次数。图片地图只有在图片的所有组成部分在页面中是紧挨在一起的时候才能使

用，如导航栏。确定图片的坐标和可能会比较繁琐且容易出错，同时使用图片地图导航也不具有可读性，因此不推荐这种方法；

内联图像是使用 data:URL scheme 的方法把图像数据加载页面中。这可能会增加页面的大小。把内联图像放到样式表（可缓存）中可以减少 HTTP 请求同时又避免增加页面文件的大小。但是内联图像现在还没有得到主流浏览器的支持。

减少页面的 HTTP 请求次数是你首先要做的一步。这是改进首次访问用户等待时间的最重要的方法。如同 Tenni Theurer 的他的博客 Browser Cahe Usage - Exposed!中所说，HTTP 请求在无缓存情况下占去了40%到60%的响应时间。让那些初次访问你网站的人获得更加快速的体验吧！

2、减少 DNS 查找次数

域名系统（DNS）提供了域名和 IP 的对应关系，就像电话本中人名和他们的电话号码的关系一样。当你在浏览器地址栏中输入 www.dudo.org 时，DNS 解析服务器就会返回这个域名对应的 IP 地址。DNS 解析的过程同样也是需要时间的。一般情况下返回给定域名对应的 IP 地址会花费20到120毫秒的时间。而且在这个过程中浏览器什么都不会做直到 DNS 查找完毕。

缓存 DNS 查找可以改善页面性能。这种缓存需要一个特定的缓存服务器，这种服务器一般属于用户的 ISP 提供商或者本地局域网控制，但是它同样会在用户使用的计算机上产生缓存。DNS 信息会保留在操作系统的 DNS 缓存中（微软 Windows 系统中 DNS Client Service）。大多数浏览器有独立于操作系统以外的自己的缓存。由于浏览器有自己的缓存记录，因此在一次请求中它不会受到操作系统的影响。

Internet Explorer 默认情况下对 DNS 查找记录的缓存时间为30分钟，它在注册表中的键值为 DnsCacheTimeout。Firefox 对 DNS 的查找记录缓存时间为1分钟，它在配置文件中的选项为 network.dnsCacheExpiration（Fasterfox 把这个选项改为了1小时）。

当客户端中的 DNS 缓存都为空时（浏览器和操作系统都为空），DNS 查找的次数和页面中主机名的数量相同。这其中包括页面中 URL、图片、脚本文件、样式表、Flash 对象等包含的主机名。减少主机名的数量可以减少 DNS 查找次数。

减少主机名的数量还可以减少页面中并行下载的数量。减少 DNS 查找次数可以节省响应时间，但是减少并行下载却会增加响应时间。我的指导原则是把这些页面中的内容分割成至少两部分但不超过四部分。这种结果就是在减少 DNS 查找次数和保持较高程度并行下载两者之间的权衡了。

3、避免跳转

跳转是使用301和302代码实现的。下面是一个响应代码为301的 HTTP 头：

HTTP/1.1 301 Moved Permanently

Location: http://example.com/newuri

Content-Type: text/html

浏览器会把用户指向到 Location 中指定的 URL。头文件中的所有信息在一次跳转中都是必需的，内容部分可以为空。不管他们的名称，301和302响应都不会被缓存除非增加一个额外的头选项，如 Expires 或者 Cache-Control 来指定它缓存。<meat />元素的刷新标签和 JavaScript 也可以实现 URL 的跳转，但是如果你必须要跳转的时候，最好的方法就是使用标准的3XXHTTP 状态代码，这主要是为了确保“后退”按钮可以正确地使用。

但是要记住跳转会降低用户体验。在用户和 HTML 文档中间增加一个跳转，会拖延页面中所有元素的显示，因为在 HTML 文件被加载前任何文件（图像、Flash 等）都不会被下载。

有一种经常被网页开发者忽略却往往十分浪费响应时间的跳转现象。这种现象发生在当 URL 本该有斜杠（/）却被忽略掉时。例如，当我们要访问 http://astrology.yahoo.com/astrology 时，实际上返回的是一个包含301代码的跳转，它指向的是 http://astrology.yahoo.com/astrology/ （注意末尾的斜杠）。在 Apache 服务器中可以使用 Alias 或者 mod_rewrite 或者 the DirectorySlash 来避免。

连接新网站和旧网站是跳转功能经常被用到的另一种情况。这种情况下往往要连接网站的不同内容然后根据用户的不同类型（如浏览器类型、用户账号所属类型）来进行跳转。使用跳转来实现两个网站的切换十分简单，需要的代码量也不多。尽管使用这种方法对于开发者来说可以降低复杂程度，但是它同样降低用户体验。一个可替代方法就是如果两者在同一台服务器上时使用 Alias 和 mod_rewrite 和实现。如果是因为域名的不同而采用跳转，那么可以通过使用 Alias 或者 mod_rewirte 建立 CNAME（保存一个域名和另外一个域名之间关系的 DNS 记录）来替代。

4、可缓存的 AJAX

Ajax 经常被提及的一个好处就是由于其从后台服务器传输信息的异步性而为用户带来的反馈的即时性。但是，使用 Ajax 并不能保证用户不会在等待异步的 JavaScript 和 XML 响应上花费时间。在很多应用中，用户是否需要等待响应取决于 Ajax 如何来使用。例如，在一个基于 Web 的 Email 客户端中，用户必须等待 Ajax 返回符合他们条件的邮件查询结果。记住一点，“异步”并不异味着“即时”，这很重要。

为了提高性能，优化 Ajax 响应是很重要的。提高 Ajxa 性能的措施中最重要的方法就是使响应具有可缓存性，具体的讨论可以查看 Add an Expires or a Cache-Control Header。其它的几条规则也同样适用于 Ajax：

Gizp 压缩文件

减少 DNS 查找次数

精简 JavaScript

避免跳转

配置 ETags

让我们来看一个例子：一个 Web2.0的 Email 客户端会使用 Ajax 来自动完成对用户地址薄的下载。如果用户在上次使用过 Email web 应用程序后没有对地址薄作任何的修改，而且 Ajax 响应通过 Expire 或者 Cacke-Control 头来实现缓存，那么就可以直接从上一次的缓存中读取地址薄了。必须告知浏览器是使用缓存中的地址薄还是发送一个新的请求。这可以通过为读取地址薄的 Ajax URL 增加一个含有上次编辑时间的时间戳来实现，例如，&t=11900241612等。如果地址薄在上次下载后没有被编辑过，时间戳就不变，则从浏览器的缓存中加载从而减少了一次 HTTP 请求过程。如果用户修改过地址薄，时间戳就会用来确定新的 URL 和缓存响应并不匹配，浏览器就会重要请求更新地址薄。

即使你的 Ajxa 响应是动态生成的，哪怕它只适用于一个用户，那么它也应该被缓存起来。这样做可以使你的 Web2.0应用程序更加快捷。

5、推迟加载内容

你可以仔细看一下你的网页，问问自己“哪些内容是页面呈现时所必需首先加载的？哪些内容和结构可以稍后再加载？

把整个过程按照 onload 事件分隔成两部分，JavaScript 是一个理想的选择。例如，如果你有用于实现拖放和动画的 JavaScript，那么它就以等待稍后加载，因为页面上的拖放元素是在初始化呈现之后才发生的。其它的例如隐藏部分的内容（用户操作之后才显现的内容）和处于折叠部分的图像也可以推迟加载

工具可以节省你的工作量：YUI Image Loader 可以帮你推迟加载折叠部分的图片，YUI Get utility 是包含 JS 和 CSS 的便捷方法。比如你可以打开 Firebug 的 Net 选项卡看一下 Yahoo 的首页。

当性能目标和其它网站开发实践一致时就会相得益彰。这种情况下，通过程序提高网站性能的方法告诉我们，在支持 JavaScript 的情况下，可以先去除用户体验，不过这要保证你的网站在没有 JavaScript 也可以正常运行。在确定页面运行正常后，再加载脚本来实现如拖放和动画等更加花哨的效果。

6、预加载

预加载和后加载看起来似乎恰恰相反，但实际上预加载是为了实现另外一种目标。预加载是在浏览器空闲时请求将来可能会用到的页面内容（如图像、样式表和脚本）。使用这种方法，当用户要访问下一个页面时，页面中的内容大部分已经加载到缓存中了，因此可以大大改善访问速度。

下面提供了几种预加载方法：

无条件加载：触发 onload 事件时，直接加载额外的页面内容。以 Google.com 为例，你可以看一下它的 spirit image 图像是怎样在 onload 中加载的。这个 spirit image 图像在 google.com 主页中是不需要的，但是却可以在搜索结果页面中用到它。

有条件加载：根据用户的操作来有根据地判断用户下面可能去往的页面并相应的预加载页面内容。在 search.yahoo.com 中你可以看到如何在你输入内容时加载额外的页面内容。

有预期的加载：载入重新设计过的页面时使用预加载。这种情况经常出现在页面经过重新设计后用户抱怨“新的页面看起来很酷，但是却比以前慢”。问题可能出在用户对于你的旧站点

建立了完整的缓存，而对于新站点却没有任何缓存内容。因此你可以在访问新站之前就加载一部内容来避免这种结果的出现。在你的旧站中利用浏览器的空余时间加载新站中用到的图像的和脚本来提高访问速度。

7、减少 DOM 元素数量

一个复杂的页面意味着需要下载更多数据，同时也意味着 JavaScript 遍历 DOM 的效率越慢。比如当你增加一个事件句柄时在500和5000个 DOM 元素中循环效果肯定是不一样的。

大量的 DOM 元素的存在意味着页面中有可以不用移除内容只需要替换元素标签就可以精简的部分。你在页面布局中使用表格了吗？你有没有仅仅为了布局而引入更多的 元素呢？也许会存在一个适合或者在语意是更贴切的标签可以供你使用。

YUI CSS utilities 可以给你的布局带来巨大帮助：grids.css 可以帮你实现整体布局，font.css 和 reset.css 可以帮助你移除浏览器默认格式。它提供了一个重新审视你页面中标签的机会，比如只有在语意上有意义时才使用 ，而不是因为它具有换行效果才使用它。

DOM 元素数量很容易计算出来，只需要在 Firebug 的控制台内输入：

document.getElementsByTagName('').length*

那么多少个 DOM 元素算是多呢？这可以对照有很好标记使用的类似页面。比如 Yahoo!主页是一个内容非常多的页面，但是它只使用了700个元素（HTML 标签）。

8、根据域名划分页面内容

把页面内容划分成若干部分可以使你最大限度地实现平行下载。由于 DNS 查找带来的影响你首先要确保你使用的域名数量在2个到4个之间。例如，你可以把用到的 HTML 内容和动态内容放在 www.example.org 上，而把页面各种组件（图片、脚本、CSS)分别存放在 statics1.example.org 和 statics.example.org 上。

你可在 Tenni Theurer 和 Patty Chi 合写的文章 Maximizing Parallel Downloads in the Carpool Lane 找到更多相关信息。

9、使 iframe 的数量最小

ifrmae 元素可以在父文档中插入一个新的 HTML 文档。了解 iframe 的工作理然后才能更加有效地使用它，这一点很重要。

<iframe>优点：

- 解决加载缓慢的第三方内容如图标和广告等的加载问题
- Security sandbox
- 并行加载脚本

<iframe>的缺点：

- 即时内容为空，加载也需要时间
- 会阻止页面加载
- 没有语意

10、不要出现404错误

HTTP 请求时间消耗是很大的，因此使用 HTTP 请求来获得一个没有用处的响应（例如404没有找到页面）是完全没有必要的，它只会降低用户体验而不会有一点好处。

有些站点把404错误响应页面改为“你是不是要找***”，这虽然改进了用户体验但是同样也会浪费服务器资源（如数据库等）。最糟糕的情况是指向外部 JavaScript 的链接出现问题并返回404代码。首先，这种加载会破坏并行加载；其次浏览器会把试图在返回的404响应内容中找到可能有用的部分当作 JavaScript 代码来执行。

Yahoo!团队实践分享：网站性能优化的34条黄金守则（二）服务器

除了在网站在内容上的改进外，在网站服务器端上也有需要注意和改进的地方，它们包括：

- 使用内容分发网络
- 为文件头指定 Expires 或 Cache-Control
- Gzip 压缩文件内容
- 配置 ETag
- 尽早刷新输出缓冲
- 使用 GET 来完成 AJAX 请求

11、使用内容分发网络

用户与你网站服务器的接近程度会影响响应时间的长短。把你的网站内容分散到多个、处于不同地域位置的服务器上可以加快下载速度。但是首先我们应该做些什么呢？

按地域布置网站内容的第一步并不是要尝试重新架构你的网站让他们在分发服务器上正常运行。根据应用的需求来改变网站结构，这可能会包括一些比较复杂的任务，如在服务器间同步 Session 状态和合并数据库更新等。要想缩短用户和内容服务器的距离，这些架构步骤可能是不可避免的。

要记住，在终端用户的响应时间中有80%到90%的响应时间用于下载图像、样式表、脚本、Flash 等页面内容。这就是网站性能黄金守则。和重新设计你的应用程序架构这样比较困难的任务相比，首先来分布静态内容会更好一点。这不仅会缩短响应时间，而且对于内容分发网络来说它更容易实现。

内容分发网络（Content Delivery Network，CDN）是由一系列分散到各个不同地理位置上的 Web 服务器组成的，它提高了网站内容的传输速度。用于向用户传输内容的服务器主要是根据和用户在网络上的靠近程度来指定的。例如，拥有最少网络跳数（network hops）和响应速度最快的服务器会被选定。

一些大型的网络公司拥有自己的 CDN，但是使用像 Akamai Technologies，Mirror Image Internet， 或者 Limelight Networks 这样的 CDN 服务成本却非常高。对于刚刚起步的企业和个人网站来说，可能没有使用 CDN 的成本预算，但是随着目标用户群的不断扩大和更加全球化，CDN 就是实现快速响应所必需的了。以 Yahoo 来说，他们转移到 CDN 上的网站程序静态内容节省了终端用户20%以上的响应时间。使用 CDN 是一个只需要相对简单地修改代码实现显著改善网站访问速度的方法。

12、为文件头指定 Expires 或 Cache-Control

这条守则包括两方面的内容：

对于静态内容：设置文件头过期时间 Expires 的值为“Never expire”（永不过期）

对于动态内容：使用恰当的 Cache-Control 文件头来帮助浏览器进行有条件的请求

网页内容设计现在越来越丰富，这就意味着页面中要包含更多的脚本、样式表、图片和 Flash。第一次访问你页面的用户就意味着进行多次的 HTTP 请求，但是通过使用 Expires 文件头就可以使这样内容具有缓存性。它避免了接下来的页面访问中不必要的 HTTP 请求。Expires 文件头经常用于图像文件，但是应该在所有的内容都使用他，包括脚本、样式表和 Flash 等。

浏览器（和代理）使用缓存来减少 HTTP 请求的大小和次数以加快页面访问速度。Web 服务器在 HTTP 响应中使用 Expires 文件头来告诉客户端内容需要缓存多长时间。下面这个例子是一个较长时间的 Expires 文件头，它告诉浏览器这个响应直到2010年4月15日才过期。

Expires: Thu, 15 Apr 2010 20:00:00 GMT

如果你使用的是 Apache 服务器，可以使用 ExpiresDefault 来设定相对当前日期的过期时间。下面这个例子是使用 ExpiresDefault 来设定请求时间后10年过期的文件头：

ExpiresDefault "access plus 10 years"

要切记，如果使用了 Expires 文件头，当页面内容改变时就必须改变内容的文件名。依 Yahoo!来说我们经常使用这样的步骤：在内容的文件名中加上版本号，如 yahoo_2.0.6.js。

使用 Expires 文件头只有会在用户已经访问过你的网站后才会起作用。当用户首次访问你的网站时这对减少 HTTP 请求次数来说是无效的，因为浏览器的缓存是空的。因此这种方法对于你网站性能的改进情况要依据他们“预缓存”存在时对你页面的点击频率（“预缓存”中已经包含了页面中的所有内容）。Yahoo!建立了一套测量方法，我们发现所有的页面浏览量中有75~85%都有“预缓存”。通过使用 Expires 文件头，增加了缓存在浏览器中内容的数量，并且可以在用户接下来的请求中再次使用这些内容，这甚至都不需要通过用户发送一个字节的请求。

13、Gzip 压缩文件内容

网络传输中的 HTTP 请求和应答时间可以通过前端机制得到显著改善。的确，终端用户的带宽、互联网提供者、与对等交换点的靠近程度等都不是网站开发者所能决定的。但是还有其他因素影响着响应时间。通过减小 HTTP 响应的大小可以节省 HTTP 响应时间。

从 HTTP/1.1开始，web 客户端都默认支持 HTTP 请求中有 Accept-Encoding 文件头的压缩格式：

Accept-Encoding: gzip, deflate

如果 web 服务器在请求的文件头中检测到上面的代码，就会以客户端列出的方式压缩响应内容。Web 服务器把压缩方式通过响应文件头中的 Content-Encoding 来返回给浏览器。

Content-Encoding: gzip

Gzip 是目前最流行也是最有效的压缩方式。这是由 GNU 项目开发并通过 RFC 1952 来标准化的。另外仅有的一个压缩格式是 deflate，但是它的使用范围有限效果也稍稍逊色。

Gzip 大概可以减少70%的响应规模。目前大约有90%通过浏览器传输的互联网交换

支持 gzip 格式。如果你使用的是 Apache，gzip 模块配置和你的版本有关：Apache 1.3使用 mod_zip，而 Apache 2.x 使用 moflate。

浏览器和代理都会存在这样的问题：浏览器期望收到的和实际接收到的内容会存在不匹配的现象。幸好，这种特殊情况随着旧式浏览器使用量的减少在减少。Apache 模块会通过自动添加适当的 Vary 响应文件头来避免这种状况的出现。

服务器根据文件类型来选择需要进行 gzip 压缩的文件，但是这过于限制了可压缩的文件。大多数 web 服务器会压缩 HTML 文档。对脚本和样式表进行压缩同样也是值得做的事情，但是很多 web 服务器都没有这个功能。实际上，压缩任何一个文本类型的响应，包括 XML 和 JSON，都值得的。图像和 PDF 文件由于已经压缩过了所以不能再进行 gzip 压缩。如果试图 gizp 压缩这些文件的话不但会浪费 CPU 资源还会增加文件的大小。

Gzip 压缩所有可能的文件类型是减少文件体积增加用户体验的简单方法。

14、配置 ETag

Entity tags（ETags）（实体标签）是 web 服务器和浏览器用于判断浏览器缓存中的内容和服务器中的原始内容是否匹配的一种机制（“实体”就是所说的“内容”，包括图片、脚本、样式表等）。增加 ETag 为实体的验证提供了一个比使用“last-modified date（上次编辑时间）”更加灵活的机制。Etag 是一个识别内容版本号的唯一字符串。唯一的格式限制就是它必须包含在双引号内。原始服务器通过含有 ETag 文件头的响应指定页面内容的 ETag。

HTTP/1.1 200 OK

Last-Modified: Tue, 12 Dec 2006 03:03:59 GMT

ETag: "10c24bc-4ab-457e1c1f"

Content-Length: 12195

稍后，如果浏览器要验证一个文件，它会使用 If-None-Match 文件头来把 ETag 传回给原始服务器。在这个例子中，如果 ETag 匹配，就会返回一个304状态码，这就节省了12195字节的响应。 GET /i/yahoo.gif HTTP/1.1

Host: us.yimg.com

If-Modified-Since: Tue, 12 Dec 2006 03:03:59 GMT

If-None-Match: "10c24bc-4ab-457e1c1f"

HTTP/1.1 304 Not Modified

ETag 的问题在于，它是根据可以辨别网站所在的服务器的具有唯一性的属性来生成的。当浏览器从一台服务器上获得页面内容后到另外一台服务器上进行验证时 ETag 就会不匹配，这种情况对于使用服务器组和处理请求的网站来说是非常常见的。默认情况下，Apache 和 IIS 都会把数据嵌入 ETag 中，这会显著减少多服务器间的文件验证冲突。

Apache 1.3和2.x 中的 ETag 格式为 inode-size-timestamp。即使某个文件在不同的服务器上会处于相同的目录下，文件大小、权限、时间戳等都完全相同，但是在不同服务器上他们的内码也是不同的。

IIS 5.0 和 IIS 6.0 处理 ETag 的机制相似。IIS 中的 ETag 格式为 Filetimestamp:ChangeNumber。用 ChangeNumber 来跟踪 IIS 配置的改变。网站所用的不同 IIS 服务器间 ChangeNumber 也不相同。 不同的服务器上的 Apache 和 IIS 即使对于完全相同的内容产生的 ETag 在也不相同，用户并不会接收到一个小而快的304响应；相反他们会接收一个正常的200响应并下载全部内容。如果你的网站只放在一台服务器上，就不会

存在这个问题。但是如果你的网站是架设在多个服务器上，并且使用 Apache 和 IIS 产生默认的 ETag 配置，你的用户获得页面就会相对慢一点，服务器会传输更多的内容，占用更多的带宽，代理也不会有效地缓存你的网站内容。即使你的内容拥有 Expires 文件头，无论用户什么时候点击“刷新”或者“重载”按钮都会发送相应的 GET 请求。

如果你没有使用 ETag 提供的灵活的验证模式，那么干脆把所有的 ETag 都去掉会更好。Last-Modified 文件头验证是基于内容的时间戳的。去掉 ETag 文件头会减少响应和下次请求中文件的大小。微软的这篇支持文稿讲述了如何去掉 ETag。在 Apache 中，只需要在配置文件中简单添加下面一行代码就可以了：

FileETag none

15、尽早刷新输出缓冲

当用户请求一个页面时，无论如何都会花费200到500毫秒用于后台组织 HTML 文件。在这期间，浏览器会一直空闲等待数据返回。在 PHP 中，你可以使用 flush()方法，它允许你把已经编译的好的部分 HTML 响应文件先发送给浏览器，这时浏览器就会可以下载文件中的内容（脚本等）而后台同时处理剩余的 HTML 页面。这样做的效果会在后台烦恼或者前台较空闲时更加明显。

输出缓冲应用最好的一个地方就是紧跟在<head />之后，因为 HTML 的头部分容易生成而且头部往往包含 CSS 和 JavaScript 文件，这样浏览器就可以在后台编译剩余 HTML 的同时并行下载它们。 例子：

```
    ... <!-- css, js -->
</head>
<?php flush(); ?>
<body>
    ... <!-- content -->
```

为了证明使用这项技术的好处，Yahoo!搜索率先研究并完成了用户测试。

16、使用 GET 来完成 AJAX 请求

Yahoo!Mail 团队发现，当使用 XMLHttpRequest 时，浏览器中的 POST 方法是一个“两步走”的过程：首先发送文件头，然后才发送数据。因此使用 GET 最为恰当，因为它只需发送一个 TCP 包（除非你有很多 cookie）。IE 中 URL 的最大长度为2K，因此如果你要发送一个超过2K 的数据时就不能使用 GET 了。

一个有趣的不同就是 POST 并不像 GET 那样实际发送数据。根据 HTTP 规范，GET 意味着“获取”数据，因此当你仅仅获取数据时使用 GET 更加有意义（从语意上讲也是如此），相反，发送并在服务端保存数据时使用 POST。

Yahoo!团队实践分享：网站性能优化的34条黄金守则（三）—JavaScript 和 CSS

除此之外，JavaScript 和 CSS 也是我们页面中经常用到的内容，对它们的优化也提高网站性能的重要方面：

CSS：

- 把样式表置于顶部
- 避免使用 CSS 表达式（Expression）
- 使用外部 JavaScript 和 CSS
- 削减 JavaScript 和 CSS
- 用<link>代替@import
- 避免使用滤镜

JavaScript

- 把脚本置于页面底部
- 使用外部 JavaScript 和 CSS
- 削减 JavaScript 和 CSS
- 剔除重复脚本
- 减少 DOM 访问
- 开发智能事件处理程序

17、把样式表置于顶部

在研究 Yahoo!的性能表现时，我们发现把样式表放到文档的<head />内部似乎会加快页面的下载速度。这是因为把样式表放到<head />内会使页面有步骤的加载显示。

注重性能的前端服务器往往希望页面有秩序地加载。同时，我们也希望浏览器把已经接收到内容尽可能显示出来。这对于拥有较多内容的页面和网速较慢的用户来说特别重要。向用户返回可视化的反馈，比如进程指针，已经有了较好的研究并形成了正式文档。在我们的研究中 HTML 页面就是进程指针。当浏览器有序地加载文件头、导航栏、顶部的 logo 等对于等待页面加载的用户来说都可以作为可视化的反馈。这从整体上改善了用户体验。

把样式表放在文档底部的问题是在包括 Internet Explorer 在内的很多浏览器中这会中止内容的有序呈现。浏览器中止呈现是为了避免样式改变引起的页面元素重绘。用户不得不面对一个空白页面。

HTML 规范清楚指出样式表要放包含在页面的<head />区域内：“和<a />不同，<link />只能出现在文档的<head />区域内，尽管它可以多次使用它”。无论是引起白屏还是出现没有样式化的内容都不值得去尝试。最好的方案就是按照 HTML 规范在文档<head />内加载你的样式表。

[url=]18、避免使用 CSS 表达式（Expression）[/url]

CSS 表达式是动态设置 CSS 属性的强大（但危险）方法。Internet Explorer 从第5个版本开始支持 CSS 表达式。下面的例子中，使用 CSS 表达式可以实现隔一个小时切换一次背景颜色：

background-color: expression((new Date()).getHours()%2 ? "#B8D4FF" : "#F08A00");

如上所示，expression 中使用了 JavaScript 表达式。CSS 属性根据 JavaScript 表达式的计

算结果来设置。expression 方法在其它浏览器中不起作用，因此在跨浏览器的设计中单独针对 Internet Explorer 设置时会比较有用。

表达式的问题就在于它的计算频率要比我们想象的多。不仅仅是在页面显示和缩放时，就是在页面滚动、乃至移动鼠标时都会要重新计算一次。给 CSS 表达式增加一个计数器可以跟踪表达式的计算频率。在页面中随便移动鼠标都可以轻松达到10000次以上的计算量。

一个减少 CSS 表达式计算次数的方法就是使用一次性的表达式，它在第一次运行时将结果赋给指定的样式属性，并用这个属性来代替 CSS 表达式。如果样式属性必须在页面周期内动态地改变，使用事件句柄来代替 CSS 表达式是一个可行办法。如果必须使用 CSS 表达式，一定要记住它们要计算成千上万次并且可能会对你页面的性能产生影响。

19、使用外部 JavaScript 和 CSS

很多性能规则都是关于如何处理外部文件的。但是，在你采取这些措施前你可能会问到一个更基本的问题：JavaScript 和 CSS 是应该放在外部文件中呢还是把它们放在页面本身之内呢？

在实际应用中使用外部文件可以提高页面速度，因为 JavaScript 和 CSS 文件都能在浏览器中产生缓存。内置在 HTML 文档中的 JavaScript 和 CSS 则会在每次请求中随 HTML 文档重新下载。这虽然减少了 HTTP 请求的次数，却增加了 HTML 文档的大小。从另一方面来说，如果外部文件中的 JavaScript 和 CSS 被浏览器缓存，在没有增加 HTTP 请求次数的同时可以减少 HTML 文档的大小。

关键问题是，外部 JavaScript 和 CSS 文件缓存的频率和请求 HTML 文档的次数有关。虽然有一定的难度，但是仍然有一些指标可以一测量它。如果一个会话中用户会浏览你网站中的多个页面，并且这些页面中会重复使用相同的脚本和样式表，缓存外部文件就会带来更大的益处。

许多网站没有功能建立这些指标。对于这些网站来说，最好的坚决方法就是把 JavaScript 和 CSS 作为外部文件引用。比较适合使用内置代码的例外就是网站的主页，如 Yahoo!主页和 My Yahoo!。主页在一次会话中拥有较少（可能只有一次）的浏览量，你可以发现内置 JavaScript 和 CSS 对于终端用户来说会加快响应时 间。

对于拥有较大浏览量的首页来说，有一种技术可以平衡内置代码带来的 HTTP 请求减少与通过使用外部文件进行缓存带来的好处。其中一个就是在首页中内置 JavaScript 和 CSS，但是在页面下载完成后动态下载外部文件，在子页面中使用到这些文件时，它们已经缓存到浏览器了。

20、削减 JavaScript 和 CSS

精简是指从去除代码不必要的字符减少文件大小从而节省下载时间。消减代码时，所有的注释、不需要的空白字符（空格、换行、tab 缩进）等都要去掉。在 JavaScript 中，由于需要下载的文件体积变小了从而节省了响应时间。精简 JavaScript 中目前用到的最广泛的两个工具是 JSMin 和 YUI Compressor。YUI Compressor 还可用于精简 CSS。

混淆是另外一种可用于源代码优化的方法。这种方法要比精简复杂一些并且在混淆的过程更易产生问题。在对美国前10大网站的调查中发现，精简也可以缩小原来代码体积的21%，而混淆可以达到25%。尽管混淆法可以更好地缩减代码，但是对于 JavaScript 来说

精简的风险更小。

除消减外部的脚本和样式表文件外，<script>和<style>代码块也可以并且应该进行消减。即使你用 Gzip 压缩过脚本和样式表，精简这些文件仍然可以节省5%以上的空间。由于 JavaScript 和 CSS 的功能和体积的增加，消减代码将会获得益处。

21、用<link>代替@import

前面的最佳实现中提到 CSS 应该放置在顶端以利于有序加载呈现。

在 IE 中，页面底部@import 和使用<link>作用是一样的，因此最好不要使用它。

22、避免使用滤镜

IE 独有属性 AlphaImageLoader 用于修正7.0以下版本中显示 PNG 图片的半透明效果。这个滤镜的问题在于浏览器加载图片时它会终止内容的呈现并且冻结浏览器。在每一个元素（不仅仅是图片）它都会运算一次，增加了内存开支，因此它的问题是多方面的。

完全避免使用 AlphaImageLoader 的最好方法就是使用 PNG8格式来代替，这种格式能在 IE 中很好地工作。如果你确实需要使用 AlphaImageLoader，请使用下划线_filter 又使之对 IE7以上版本的用户无效。

23、把脚本置于页面底部

脚本带来的问题就是它阻止了页面的平行下载。HTTP/1.1 规范建议，浏览器每个主机名的并行下载内容不超过两个。如果你的图片放在多个主机名上，你可以在每个并行下载中同时下载2个以上的文件。但是当下载脚本时，浏览器就不会同时下载其它文件了，即便是主机名不相同。

在某些情况下把脚本移到页面底部可能不太容易。比如说，如果脚本中使用了 document.write 来插入页面内容，它就不能被往下移动了。这里可能还会有作用域的问题。很多情况下，都会遇到这方面的问题。

一个经常用到的替代方法就是使用延迟脚本。DEFER 属性表明脚本中没有包含 document.write，它告诉浏览器继续显示。不幸的是，Firefox 并不支持 DEFER 属性。在 Internet Explorer 中，脚本可能会被延迟但效果也不会像我们所期望的那样。如果脚本可以被延迟，那么它就可以移到页面的底部。这会让你的页面加载的快一点。

24、剔除重复脚本

在同一个页面中重复引用 JavaScript 文件会影响页面的性能。你可能会认为这种情况并不多见。对于美国前10大网站的调查显示其中有两家存在重复引用脚本的情况。有两种主要因素导致一个脚本被重复引用的奇怪现象发生：团队规模和脚本数量。如果真的存在这种情况，重复脚本会引起不必要的 HTTP 请求和无用的 JavaScript 运算，这降低了网站性能。

在 Internet Explorer 中会产生不必要的 HTTP 请求，而在 Firefox 却不会。在 Internet Explorer 中，如果一个脚本被引用两次而且它又不可缓存，它就会在页面加载过程中产生两次 HTTP 请求。即时脚本可以缓存，当用户重载页面时也会产生额外的 HTTP 请求。

除增加额外的 HTTP 请求外，多次运算脚本也会浪费时间。在 Internet Explorer 和 Firefox 中不管脚本是否可缓存，它们都存在重复运算 JavaScript 的问题。

一个避免偶尔发生的两次引用同一脚本的方法是在模板中使用脚本管理模块引用脚本。在 HTML 页面中使用<script />标签引用脚本的最常见方法就是：

<script type="text/javascript" src="menu_1.0.17.js"></script>

在 PHP 中可以通过创建名为 insertScript 的方法来替代：

<?php insertScript("menu.js") ?>

为了防止多次重复引用脚本，这个方法中还应该使用其它机制来处理脚本，如检查所属目录和为脚本文件名中增加版本号以用于 Expire 文件头等。

25、减少 DOM 访问

使用 JavaScript 访问 DOM 元素比较慢，因此为了获得更多的应该页面，应该做到：

- 缓存已经访问过的有关元素
- 线下更新完节点之后再将它们添加到文档树中
- 避免使用 JavaScript 来修改页面布局

有关此方面的更多信息请查看 Julien Lecomte 在 YUI 专题中的文章“高性能 Ajax 应该程序”。

26、开发智能事件处理程序

有时候我们会感觉到页面反应迟钝，这是因为 DOM 树元素中附加了过多的事件句柄并且些事件句病被频繁地触发。这就是为什么说使用 event delegation（事件代理）是一种好方法了。如果你在一个 div 中有10个按钮，你只需要在 div 上附加一次事件句柄就可以了，而不用去为每一个按钮增加一个句柄。事件冒泡时你可以捕捉到事件并判断出是哪个事件发出的。

你同样也不用为了操作 DOM 树而等待 onload 事件的发生。你需要做的就是等待树结构中你要访问的元素出现。你也不用等待所有图像都加载完毕。

你可能会希望用 DOMContentLoaded 事件来代替 onload，但是在所有浏览器都支持它之前你可使用 YUI 事件应用程序中的 onAvailable 方法。

有关此方面的更多信息请查看 Julien Lecomte 在 YUI 专题中的文章“高性能 Ajax 应该程序”。

Yahoo!团队实践分享：网站性能优化的34条黄金守则（四）—图片、Coockie与移动应用

图片和 Coockie 也是我们网站中几乎不可缺少组成部分，此外随着移动设备的流行，对于移动应用的优化也十分重要。这主要包括：

Coockie:

- 减小 Cookie 体积
- 对于页面内容使用无 coockie 域名

图片：

- 优化图像
- 优化 CSS Spirite
- 不要在 HTML 中缩放图像
- favicon.ico 要小而且可缓存

移动应用：

- 保持单个内容小于25K
- 打包组件成复合文本

27、减小 Cookie 体积

HTTP coockie 可以用于权限验证和个性化身份等多种用途。coockie 内的有关信息是通过 HTTP 文件头来在 web 服务器和浏览器之间进行交流的。因此保持 coockie 尽可能的小以减少用户的响应时间十分重要。
有关更多信息可以查看 Tenni Theurer 和 Patty Chi 的文章“When the Cookie Crumbles”。这们研究中主要包括：

- 去除不必要的 coockie
- 使 coockie 体积尽量小以减少对用户响应的影响
- 注意在适应级别的域名上设置 coockie 以便使子域名不受影响
- 设置合理的过期时间。较早地 Expire 时间和不要过早去清除 coockie，都会改善用户的响应时间。

28、对于页面内容使用无 coockie 域名

当浏览器在请求中同时请求一张静态的图片和发送 coockie 时，服务器对于这些 coockie 不会做任何地使用。因此他们只是因为某些负面因素而创建的网络传输。所有你应该确定对于静态内容的请求是无 coockie 的请求。创建一个子域名并用他来存放所有静态内容。

如果你的域名是 www.example.org，你可以在 static.example.org 上存在静态内容。但是，如果你不是在 www.example.org 上而是在顶级域名 example.org 设置了 coockie，那么所有对于 static.example.org 的请求都包含 coockie。在这种情况下，你可以再重新购买一个新的域名来存在静态内容，并且要保持这个域名是无 coockie 的。Yahoo!使用的是 ymig.com，YouTube 使用的是 ytimg.com，Amazon 使用的是 images-anazon.com 等等。

使用无 coockie 域名存在静态内容的另外一个好处就是一些代理（服务器）可能会拒绝对 coockie 的内容请求进行缓存。一个相关的建议就是，如果你想确定应该使用 example.org 还是 www.example.org 作为你的一主页，你要考虑到 coockie 带来的影响。忽略掉 www 会使你除了把 coockie 设置到*.example.org（*是泛域名解析，代表了所有子域名译者 dudo 注）外没有其它选择，因此出于性能方面的考虑最好是使用带有 www 的子域名并且在它上面设置 coockie。

29、优化图像

设计人员完成对页面的设计之后，不要急于将它们上传到 web 服务器，这里还需要做几件事：

- 你可以检查一下你的 GIF 图片中图像颜色的数量是否和调色板规格一致。 使用 imagemagick 中下面的命令行很容易检查：

identify -verbose image.gif

如果你发现图片中只用到了4种颜色，而在调色板的中显示的256色的颜色槽，那么这张图片就还有压缩的空间。

- 尝试把 GIF 格式转换成 PNG 格式，看看是否节省空间。大多数情况下是可以压缩的。由于浏览器支持有限，设计者们往往不太乐意使用 PNG 格式的图片，不过这都是过去的事情了。现在只有一个问题就是在真彩 PNG 格式中的 alpha 通道半透明问题，不过同样的，GIF 也不是真彩格式也不支持半透明。因此 GIF 能做到的，PNG（PNG8）同样也能做到（除了动画）。下面这条简单的命令可以安全地把 GIF 格式转换为 PNG 格式：

convert image.gif image.png

“我们要说的是：给 PNG 一个施展身手的机会吧！”

- 在所有的 PNG 图片上运行 pngcrush（或者其它 PNG 优化工具）。例如：

pngcrush image.png -rem alla -reduce -brute result.png

- 在所有的 JPEG 图片上运行 jpegtran。这个工具可以对图片中的出现的锯齿等做无损操作，同时它还可以用于优化和清除图片中的注释以及其它无用信息（如 EXIF 信息）：

jpegtran -copy none -optimize -perfect src.jpg dest.jpg

30、优化 CSS Spirite

- 在 Spirite 中水平排列你的图片，垂直排列会稍稍增加文件大小；
- Spirite 中把颜色较近的组合在一起可以降低颜色数，理想状况是低于256色以便适用 PNG8格式；
- 便于移动，不要在 Spirite 的图像中间留有较大空隙。这虽然不大会增加文件大小但对于用户代理来说它需要更少的内存来把图片解压为像素地图。100x100的图片为1万像素，而1000x1000就是100万像素。

31、不要在 HTML 中缩放图像

不要为了在 HTML 中设置长宽而使用比实际需要大的图片。如果你需要：

**

那么你的图片（mycat.jpg）就应该是100x100像素而不是把一个500x500像素的图片缩小使用。

32、favicon.ico 要小而且可缓存

favicon.ico 是位于服务器根目录下的一个图片文件。它是必定存在的，因为即使你不关心它是否有用，浏览器也会对它发出请求，因此最好不要返回一个404 Not Found 的响应。由于是在同一台服务器上，它每被请求一次 coockie 就会被发送一次。这个图片文件还会影

响下载顺序，例如在 IE 中当你在 onload 中请求额外的文件时，favicon 会在这些额外内容被加载前下载。

因此，为了减少 favicon.ico 带来的弊端，要做到：

- 文件尽量地小，最好小于1K
- 在适当的时候（也就是你不要打算再换 favicon.ico 的时候，因为更换新文件时不能对它进行重命名）为它设置 Expires 文件头。你可以很安全地把 Expires 文件头设置为未来的几个月。你可以通过核对当前 favicon.ico 的上次编辑时间来作出判断。

Imagemagick 可以帮你创建小巧的 favicon。

33、保持单个内容小于25K

这条限制主要是因为 iPhone 不能缓存大于25K 的文件。注意这里指的是解压缩后的大小。由于单纯 gizp 压缩可能达不要求，因此精简文件就显得十分重要。

查看更多信息，请参阅 Wayne Shea 和 Tenni Theurer 的文件“Performance Research, Part 5: iPhone Cacheability - Making it Stick”。

34、打包组件成复合文本

把页面内容打包成复合文本就如同带有多附件的 Email，它能够使你在一个 HTTP 请求中取得多个组件（切记：HTTP 请求是很奢侈的）。当你使用这条规则时，首先要确定用户代理是否支持（iPhone 就不支持）。