雅虎网站页面性能优化的34 条黄金守则

雅虎团队经验：网站页面性能优化的34 条黄金守则

1 、尽量减少HTTP 请求次数

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.rjboy.cn 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。

3 、避免跳转
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。

4 、可缓存的AJAX

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。

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。

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。

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。

9 、使iframe 的数量最小
<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

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。

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

13 、Gzip 压缩文件内容
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 。

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。

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 。

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。

18 、避免使用CSS 表达式（Expression ）

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

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

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。

27 、减小Cookie 体积

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

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。100×100 的图片为1 万像素，而1000×1000 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100×100 像素而不是把一个500×500 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。

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。

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）。