雅虎23条军规

## 减少HTTP请求次数

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

（1）CSS Sprites：将多个图片合并到一个单独的图片中，再结合css的background-position属性一起使用，通过合并图片减少http请求（合并后的图片会比分离的图片的总和要小，因为它降低了图片自身的开销（颜色表，格式信息等））

适用场景：若需要在页面中为背景，按钮，导航栏，链接等提供大量图片，可使用CSS Sprites

（2）合并脚本和样式表：若将css代码和javascript代码分别放在多个文件中，每个文件都会导致一个额外的http请求，将这些文件合并到一个文件中，可以减少http请求的数量并缩短最终用户时间

注：若不同页面需要不同的脚本，可能脚本合并的组合数会比较多

## 使用内容发布网络(CDN)

通常来说，Web客户端离当前Web服务器越近，响应时间会更快

（1）内容发布网络：内容发布网络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

（2）一些大型公司都拥有他们自己的CDN，但使用一个CDN服务提供商更为有效

优点：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除了缩短响应时间之外，CDN还提供备份，扩展存储功能和进行缓存等服务

缺点：CDN服务开销可能比较大，且CDN服务提供商在其所有客户之间共享其Web服务组，可能会受到其它网站流量的影响；无法直接控制组件服务器所带来的特殊麻烦，如修改HTTP响应头必须通过服务器提供商来完成

## 避免空的src和href

当link标签的href属性为空、script标签的src属性为空的时候，浏览器渲染的时候会把当前页面的URL作为它们的属性值，从而把页面的内容加载进来作为它们的值。

## 添加Expires头

（1）主要通过配置组件，使用一个长久的Expires头，使这些组件能够被缓存，使在后续的页面浏览中避免不必要的http请求（浏览器可以直接从硬盘上读取组件而无需生成任何http流量）,web服务器使用Expires头来告诉web客户端它可以使用一个组件的当前副本，直到指定的时间为止

（2）长久的Expires头最常用于图片，但应该将其应用在所有组件上，包括脚本、样式表和Flash等不经常变化的组件，但添加长久的Expires会带来额外的开发成本

（3）浏览器（和代理）通过使用缓存来减少http请求的数量，并减小http响应的大小，从而使页面加载得更快，Expires在http响应中发送，web服务器通过使用Expires头来告诉Web客户端它可以使用一个组件的当前副本

（4）Expires头的限制：Expires头使用一个特定的时间，它要求服务器和客户端的时钟严格同步，另外，如果到了过期时间，需要在服务器配置中提供一个新的日期

（5）Cache-Control头：http1.1引入的，主要用来克服Expires的限制

主要使用max-age指令指定组件被缓存多久

注：可以同时指定Expires和Cache-Control两个响应头，因为有些浏览器不支持http1.1

（6）修订文件名：为了确保用户能够获取组件的最新版本，需要在所有HTML页面中修改组件的文件名

（7）一个具有长久的Expires头的组件将会被缓存，在后续请求时浏览器直接从磁盘上读取它，避免了一个http请求。如果一个组件没有长久的Expires头，它仍然会存储在浏览器的缓存中，在后续请求中，浏览器会检查缓存并发现组件以及过期。为了提高效率，浏览器会向原始服务器发送一个条件GET请求，如果组件没有改变，原始服务器可以免于发送整个组件，而是发送一个很小的头，告诉浏览器可以使用其缓存的组件

## 使用gzip压缩内容

（1）主要通过http响应的大小来减少响应时间，从http1.1开始，Web客户端可以通过http请求中的Accept-Encoding来标识对压缩的支持。如果Web服务器看到请求中有这个头，就会使用客户端列出来的方法中的一种来压缩响应，Web服务器通过响应中的Content-Encoding头来通知Web客户端

（2）压缩什么：一般压缩HTML文档、样式表和脚本

（3）压缩的成本：服务器端会花费额外的CPU周期来完成压缩，客户端要对压缩文件进行解压缩

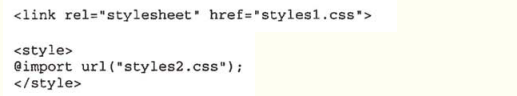
## 把CSS放到顶部

1）将样式表放在底部，可能会延迟页面的加载，会出现“白屏”和“无样式闪烁”，将样式表放在顶部，可以使页面逐步呈现，避免页面出现”白屏“和“无样式闪烁”

（2）白屏：浏览器延迟内容呈现，直到所有的样式表都下载完毕后才开始开始显示内容，即页面完全空白，直到页面所有内容同时涌上屏幕

无样式闪烁：样式表被正确地下载及解析，已经呈现的页面和图片要用新的样式重绘

（3）link和@import：



注：使用@import规则可能会导致白屏现象，即使是放在<head>中， 一般组件都是按照它们在页面中出现的顺序下载的，而使用@import会带来下载时的无序性

## 将脚本放在底部

（1）将脚本放在底部，既可以使页面逐步呈现，也可以提高下载的并行度

（2）将样式表放在<head>中，可以首先下载它们而不会阻止页面呈现，使用脚本时，对于所有位于脚本以下的内容，逐步呈现都被阻塞了。将脚本放在页面越靠下的地方，意味着越多的内容能够逐步地呈现

（3）http1.1建议浏览器从每个主机名并行地下载两个组件。然而，在下载脚本时并行下载其实是被禁用的，即使是用了不同的主机名，浏览器也不会启动其他的下载

（4）脚本对web页面的影响：脚本会阻塞其后面内容的呈现，脚本会阻塞其后面组件的下载（脚本会阻塞并行下载）

## 避免CSS表达式

（1）CSS表达式不只在页面呈现和大小改变时求值，当页面滚动，甚至连用户鼠标的页面移动时都要求值

## 将CSS和JS放到外部文件中

1）在某种程度上，内联会快一些，尽管总下载量时是相同的，但外部示例需要承担多个http请求带来的开销。但是现实中还是使用外部文件会产生比较快的页面，因为javascript和css文件有可能被浏览器缓存起来

（2）使用外部文件可以提高组件的重用率

## 减少DNS查找

（1）Internet是通过IP地址来查找服务器的，由于IP地址很难记忆，通常使用包含主机名的URL来代替它，但当浏览器发送其请求时，IP地址仍然是必需的，这就是Domain Name System(DNS)所处的角色。DNS将主机名映射到IP地址上，在输入一个URL时，连接到浏览器的DNS解析器会返回服务器的IP地址

（2）DNS也是开销，在DNS查找完成之前，浏览器不能从主机名那里下载到任何东西，响应时间依赖于DNS解析器（通常由你的ISP提供）

（3）DNS缓存：DNS查找可以被缓存起来以提高性能，在用户请求了一个主机名后，DNS信息会留在操作系统的DNS缓存中，之后对于该主机名的请求将无需进行过多的DNS查找

（4）通过使用keep-Alive和较少的域名来减少DNS查找，keep-Alive通过重用现有连接避免了重复的DNS查找，减少唯一主机名的数量可以减少DNS查找

（5）建议：可以将组件分别放在至少2个，但不要超过4个主机名下，这是在减少DNS查找和允许高度并行下载之间作出的很好的权衡

## 精简CSS和JS

（1）精简是从代码中移除不必要的字符以减小其大小进而改善加载时间，在代码被精简后，所有的注释以及不必要的空白字符都将被移除

（2）内联的javascript也应该被精简

（3）精简CSS能够带来的节省通常要小于精简javascript，因为css中的注释和空白比javascript中的少，最大的潜在节省来自于优化CSS——合并相同的类，移除不适用的类等

## 避免重定向

（1）重定向就是，在网页上设置一个约束条件，条件满足，就自动转入到其它网页、网址

（2）重定向用于将用户从一个url重新路由到另一个url，重定向有很多种——301和302是最常用的两种，重定向会使你的页面变慢，javascript可以通过document.location设置为期望的URL执行重定向

（3）当web服务器向浏览器返回一个重定向时，响应中就会返回一个范围在3xx的状态码，这表示用户代理必需执行进一步操作才能完成请求

注：304并不是真的重定向，它用来响应条件GET请求，避免下载已经存在于浏览器缓存中的数据

（4）重定向是如何损伤性能的：重定向会延迟整个HTML文档的传输，在HTML文档到达之前，页面中不会呈现出任何东西，也没有任何组件会被下载

## 移除重复脚本

（1）重复脚本损伤性能的方式有两种——不必要的HTTP请求和执行javascript所浪费的时间

（2）在页面中多次包含相同的脚本会使页面变慢； 在IE中，如果脚本没有被缓存，或在重新加载页面时，会产生额外的http请求; 在火狐和IE中，脚本会被多次求值

## 配置ETag

（1）ETag（实体标签）是web服务器和浏览器用于确认缓存组件的一种机制

（2）浏览器下载组件时，会将它们存储到缓存中，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，避免产生http请求，如果缓存的组件过期了（或者用户明确地重新加载了页面），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，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的状态码

（3）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：比较最新修改日期，比较实体标签（ETag是唯一标识了一个组件的一个特定版本的字符串）

（4）ETag带来的问题，当浏览器向一台服务器获取了原始组件，又向另一台不同的服务器发起条件get请求时，ETag是不会匹配的

（5）即使组件具有长久的Expires头，一旦用户单击了Reload或Refresh按钮。依然会产生条件get请求

## 可缓存的AJAX

"异步"并不意味着"即时"：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。

## 使用GET来完成AJAX请求

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获取数据时更加有意义。

## 减少DOM元素数量

是否存在一个是更贴切的标签可以使用?人生不仅仅是DIV+CSS

## 避免404

　　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。

## 减少Cookie的大小

## 使用无cookie的域

　　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。

## 不要使用滤镜

png24的在IE6半透明那种东西，别乱使，淡定的切成PNG8+jpg

## 不要在HTML中缩放图片

## 缩小favicon.ico并缓存