**为什么要提升web性能？**

　　Web性能黄金准则：只有10%~20%的最终用户响应时间花在了下载html文档上，其余的80%~90%时间花在了下载页面组件上。

　　web性能对于[用户体验](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=%E7%94%A8%E6%88%B7%E4%BD%93%E9%AA%8C)有及其重要的影响，根据著名的`2-5-8`原则：

　　当用户在2秒以内得到响应，会感觉系统的响应非常快 当用户在2-5秒之内得到响应，会感觉系统的响应速度还可以 当用户在5-8秒之内得到响应，会感觉系统的响应非常慢，但还可以接受 当用户在8秒之后都没有得到响应，会感觉系统糟透了，甚至系统已经挂掉；要么打开竞争对手的网站，要么重新发起第二次请求。

　　凡事都需要研究，通过科学的研究我们就可以找到事物的发展规律。这里要感谢[雅虎](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=%E9%9B%85%E8%99%8E)的工程师总结的14条前端优化法则，使得我们可以站在巨人的肩膀上。《高性能网站建设》这本书中的14条优化原则，总结起来主要是以下个方面的优化：

1. 减少HTTP请求
2. 页面内部优化
3. 启用缓存
4. 减少下载量
5. 网络连接上的优化

　　为什么减少HTTP请求可以提高Web性能？

　　要回答这个问题，我们就要了解当[浏览器](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=%E6%B5%8F%E8%A7%88%E5%99%A8)向服务器发送一个http请求知道获取数据都经历哪些过程：

　　开启一个链接（[tcp/ip](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=tcp%2Fip)的[三次握手](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=%E4%B8%89%E6%AC%A1%E6%8F%A1%E6%89%8B)过程） -> 发送请求 -> 等待（[网络延迟](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=%E7%BD%91%E7%BB%9C%E5%BB%B6%E8%BF%9F)跟服务器的处理时间） -> 下载数据

　　我们看一下百度首页中的http请求在各阶段耗费的时间，上面不同的颜色代表下图中的不同阶段



　　可以看到除了图片之外，其余大部分http请求的事件花在了建立连接与等待阶段。

　　http协议建立在TIC/IP协议之上，在[TCP/IP协议](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=TCP%2FIP%E5%8D%8F%E8%AE%AE)中，[TCP协议](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=TCP%E5%8D%8F%E8%AE%AE)提供可靠的连接服务，采用[三次握手](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=%E4%B8%89%E6%AC%A1%E6%8F%A1%E6%89%8B)建立一个连接。 简单来说[三次握手](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=%E4%B8%89%E6%AC%A1%E6%8F%A1%E6%89%8B)就是一个身份确认的过程：

　　（第一次握手：主机A发送位码为syn＝1,随机产生seq number=1234567的数据包到服务器，主机B由[SYN](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=SYN)=1知道，A要求建立联机；）

　　晴儿：你是潇哥哥吗，我是晴儿

　　（第二次握手：主机B收到请求后要确认联机信息，向A发送ack number=(主机A的seq+1),syn=1,ack=1,随机产生seq=7654321的包）

　　潇剑：这货是谁，[一箫一剑走江湖](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=%E4%B8%80%E7%AE%AB%E4%B8%80%E5%89%91%E8%B5%B0%E6%B1%9F%E6%B9%96)，下一句是什么？

　　（第三次握手：主机A收到后检查ack number是否正确，即第一次发送的seq number+1,以及位码ack是否为1，若正确，主机A会再发送ack number=(主机B的seq+1),ack=1，主机B收到后确认seq值与ack=1则连接建立成功。）

　　晴儿：这首诗。。。你真的是潇哥哥，一萧一剑走江湖，千古情愁酒一回。。。

　　潇剑：晴儿，你真的是晴儿。。。。

　　（啪啪啪啪啪啪啪啪啪啪啪啪啪啪啪。。。。。。。。。。。。）

　　言归正传，这个过程也是需要消耗时间的，在百度首页找到一个极端的例子：



　　而等待的时间通常也大于内容下载的时间，这里同样找到一个极端例子：



　　由此我们可以得出结论：一个http请求绝大多数的时间消耗在了建立连接跟等待的时间，优化的方法是减少http请求。

**如何提高web性能？**

　　1、减少HTTP请求

　　一般来说要减少http请求通常从两个方面下手：减少图片的请求、减少脚本文件与样式表的请求

　　图片的减少通常有两种方式：css sprites、内联图片、IconFont。

　　CSS Sprites：将多张图片合并成一幅单独的图片，使用css的background-position属性，将[html元素](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=html%E5%85%83%E7%B4%A0)的背景图片放到sprites 图片中的期望位置上。使用这项技术的附加优点是他降低了下载量，合并后的图片比分离的图片和更小，因为它降低了图片自身的开销（颜色表、格式信息等等）。实际项目中css sprites是一项体力活，因为开发过程中需要对这张大图进行维护（添加、减少图片），张鑫旭同学的文章中有介绍如何管理sprites图片可以作为参考（这里）。如果需要在页面中为背景、链接、导航栏提供大量的图片，css sprites绝对是一种优秀的解决方案（干净的标签、较少的图片、较短的响应时间）。

　　内联图片：通过使用data:URL模式可以再页面中包含图片而无需任何额外的请求。缺点就是[IE8](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=IE8)以下的[浏览器](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=%E6%B5%8F%E8%A7%88%E5%99%A8)不支持这种方式，而[IE8](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=IE8)在数据大小上有限制，只能支持23kb以内的数据。对于较小的图片来说可以直接内联到web页面中，但对于大图片内联到页面里会导致页面变大，聪明的做法是使用css，将内联的图片作为背景使用，并放到[外部样式表](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=%E5%A4%96%E9%83%A8%E6%A0%B7%E5%BC%8F%E8%A1%A8)中，这意味着数据可以缓存在样式表内部。使用[外部样式表](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=%E5%A4%96%E9%83%A8%E6%A0%B7%E5%BC%8F%E8%A1%A8)虽然增加了一个http请求，但样式可以被[浏览器](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=%E6%B5%8F%E8%A7%88%E5%99%A8)缓存，得到额外的收获。另外一点需要注意：[base64](http://zhannei.baidu.com/cse/search?s=17846539860057253473&entry=1&q=base64)是有损压缩。

Web性能优化 网站优化 性能优化

　　IconFont：图标字体，这是近年来新流行的一种以字体代替图片的技术。它可以适应任何分辨率而不会出现图片模糊问题，与图片相比它具有更小的容量，更高的灵活性（像字体一样可以设置图标大小、颜色、透明度、hover状态、反转等），IE8以上的浏览器都支持该技术。在使用IconFont之前，你首先要确定你选则的字体库是否是收费。详细内容可以参考这篇文章：图标字体化浅谈

　　减少脚本与样式表的请求主要原则就是合并。在实际开发中我们遵循模块化的原则将代码分散到许多小文件中，按照软件开发的原则这是完全正确的，但对于上线页面来说，每一个文件都会产生一个http请求，严重影响性能。和css sprites一样，将这些小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响应时间。在合并过程中我们还需要使用工具精简（移除不必要的字符以减小文件大小缩减下载时间）和混淆（除了移除不必要字符外，还会改写源代码，比如函数和变量名使用更短的标量名）Javascript代码。对于采用AMD或CMD进行模块化开发的同学，在合并过程中通常会将依赖的其他模块打包到一个文件中，而模板html通常以字符串的方式内联到Javascript文件中。目前最常用的前端构建工具就是glup，这里有一篇初步应用的文章：前端 | gulp 打包 require.js 模块依赖

　　2、页面内部优化

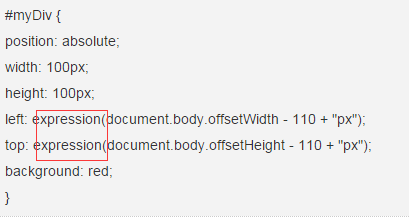
　　关于页面内部优化主要方向：样式表放在顶部、脚本文件放在底部、避免css表达式、把脚本的样式表放在外部、移除重复脚本

　　关心性能的工程师都希望页面能否尽快的展现在用户面前，对于页面中很多内容的页面我们都希望内容能够逐步加载，为用户提供可视化回馈。而将样式表放在底部会导致浏览器阻止内容逐步呈现。为避免当页面变化时重绘页面元素，浏览器会阻塞页面呈现，直到样式表解析完毕（详细内容可以查看我的这篇博客）。所以如果将样式表放在顶部并不会减少资源的加载时间，它减少的是页面的呈现时间。小米主页曾经犯过这样的错误：



　　将样式表放在底部会阻塞页面的逐步呈现，而将script文件放在页面顶部同样会阻塞页面的逐步呈现。script元素会阻塞后续内容的解析，因为script中可以同过document.write来更改页面。解决的办法就是将script标签放在页面底部。这样既可以让内容逐步呈现，也可以提高下载的并行度。如果我们确定不需要document.write那可以为script标签加上asyn属性（Ie中要加上defer）提高并行下载度。

　　CSS表达式是ie支持的可以用来动态更改css属性的一种方式，我们不需要了解太多，她的书写方式如下，一旦在产品中发现expression关键字就要彻底消灭。



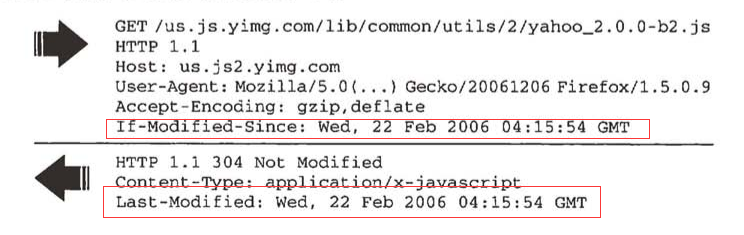
　　使用外部脚本和样式这一条，我想凡是有点经验的工程师都会这么干。

　　移除重复脚本：这条说的主要是避免在页面中多次加入同一份Javascript代码，如果我们的开发中有依赖管理的方式比如AMD、CMD，基本不会出现这种情况。

　　3、启用缓存

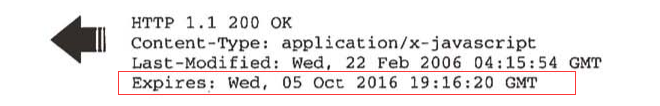
　　关于缓存的使用这里介绍两套方案：expires/If-Modified-Since、Cache-Control/Etag；前者是HTTP1.0中的缓存方案，后者是HTTP1.1中缓存方案，若http头部中同时出现二者，后者的优先级更高。

　　If-modified-since的方式通常被称为条件Get。浏览器缓存中保存了一个文件的副本，但需要向服务器询问此副本是否可用。If-Modified-Since是浏览器将最后修改时间发送给服务器，服务器相应头中Last-Modified进行对比；若If-Modified-Since <= Last-Modified 则浏览器读取本地副本。此时响应状态为304 Not Modified, 并不在发送响应体。



Web性能优化 网站优化 性能优化

　　Expries：虽然使用条件GET和304响应能够节省时间，但浏览器跟服务器端仍然要发送一次请求进行确认。通过明确设置副本的过期时间可以避免条件GET。当浏览器发现响应头中的expires时，会将过期时间和文件一起保存到缓存中去。在过期之前一直从缓存中读取。expires头使用一个特定的时间来指定缓存的有效期，他要求浏览器与服务器时间完全一致。而且一旦过期，服务器端配置中需要重新设顶一个过期时间。



　　ETag（实体标签）：是服务器用于检查浏览器缓存有效性的一种机制。ETag在HTTP1.1中引入，ETag是唯一标识了一个组件的一个特定版本的字符串。唯一的格式约束是这个字符串必须使用双引号。如果浏览器要验证一个组件是否有效他会使用If-None-Match将etag字符串传送给服务器。如果ETag是匹配的，服务器端会返回304.（如果实体数据需要根据User-Agent或Accept-Language来改变时，ETag提供了更高的灵活性）。对于使用服务器集群的网站来说，从一台服务器到另一台服务器，ETag通常是无法匹配的。这是ETag的问题。而且即便同时使用If-Modified-Since和If-None-Match也并不能达到预期效果。解决方法总是有的：自定义Etag格式

Web性能优化 网站优化 性能优化

　　Cache-Control：HTTP1.1引入了来代替Expires，它使用max-age指令来指定副本被缓存多久，该指令以秒为单位定义了一个更新窗，组件从被请求开始到现在的秒数小于设定值，则一直使用副本。避免了一次http请求。相比Expries，Cache-Control指令提供了更细粒度的控制。详细内容请看大额同学的文章：透过浏览器看HTTP缓存

　　4、减少下载量

　　减少下载量最有效的方式就是开启gzip压缩，gzip是GNU开发的一种免费格式。压缩组件通过减小http响应的大小来加快响应速度。HTTP1.1通过使用Accept-Encoding来标识支持的压缩，如果服务器看到这个标识，会使用请求头中的一种方式来压缩响应。并通过Content-Encoding来通知web客户端。很多网站会压缩html文件，实际上包括xml跟json在内的任何文本都可以压缩，但图片和pdf不应该压缩。根据经验通常可以对大于1kb或2kb的文件进行压缩。压缩通常能将响应的数据量减少70%。压缩的成本在于：服务器需要耗费额外的cpu进行压缩，客户端需要解压缩。所以需要在cpu的消耗和数据块的大小之间进行取舍。

　　5、优化网络连接

　　网络连接的优化主要有三个规则：使用CDN加速、减少DNS查找、避免重定向

　　CDN：CDN是地理上分布的web server的集合，用于更高效地发布内容。通常基于网络远近来选择给具体用户服务的web server。 这缩短了资源的传输响应时间，有效提高web性能。

　　DNS用于映射主机名和IP地址，一般一次解析需要20～120毫秒。浏览器会首先根据页面的主机名进行域名解析，在有ISP返回结果之前页面不会加载任何内容，所以减少DNS查找可以有效降低等待时间。为达到更高的性能，DNS解析通常被多级别地缓存，如由ISP或局域网维护的caching server，本地机器操作系统的缓存（如windows上的DNS Client Service），浏览器。IE的缺省DNS缓存时间为30分钟，Firefox的缺省缓冲时间是1分钟。 我们能做的是尽量减少一个页面的主机名，但要在浏览器最大并行下载数跟dns查找之间做权衡。根据雅虎的研究，最好将主机名控制在2-4个内。

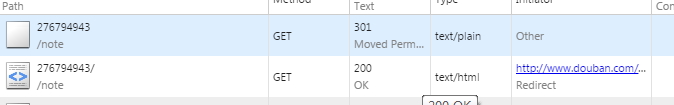
　　重定向：将一个URL重新路由到另一个URL。重定向功能是通过301和302这两个HTTP状态码完成的，如：

　　HTTP/1.1 301 Moved Permanently

　　Location: http://example.com/newuri

　　Content-Type: text/html

　　浏览器自动重定向请求到Location指定的URL上，重定向的主要问题是降低了用户体验。 种最耗费资源、经常发生而很容易被忽视的重定向是URL的最后缺少/，导致自动产生结尾斜线的原因是，浏览器在进行get请求是必须指定一些路径；如果没有路径它就会简单的使用文档根。（主机缺少结尾斜线是不会发生重定向：http://www.baidu.com）缺少结尾斜线发生重定向是很多web服务器的默认行为。需要在服务器端设置方可消除。以下图片是豆瓣的一个url请求：



　　雅虎的14条优化规则在很长的一段时间里发挥着重要作用，随着技术的发展，单单这十四条原则已经不能够满足前端性能优化。【cnblogs】