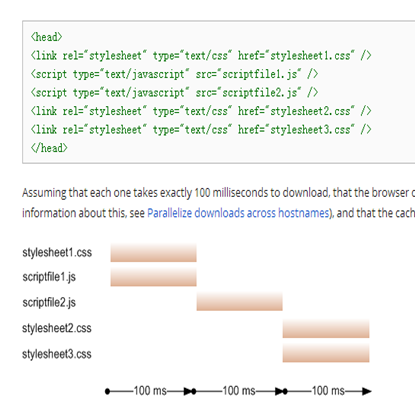
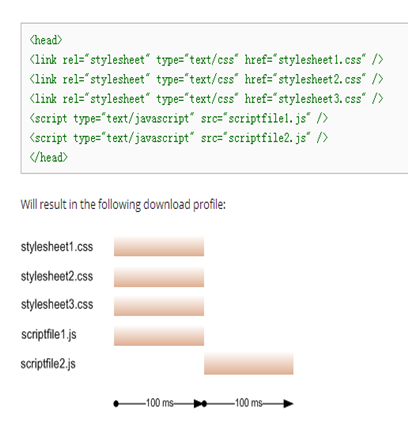
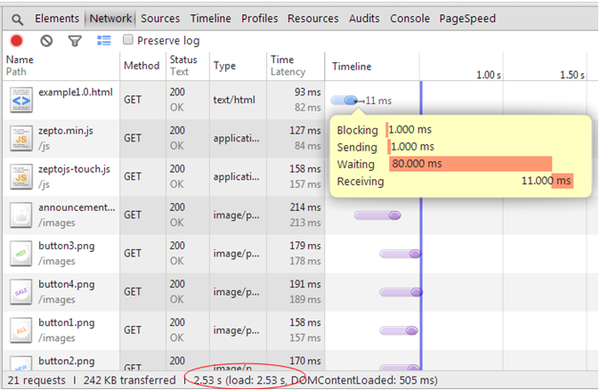
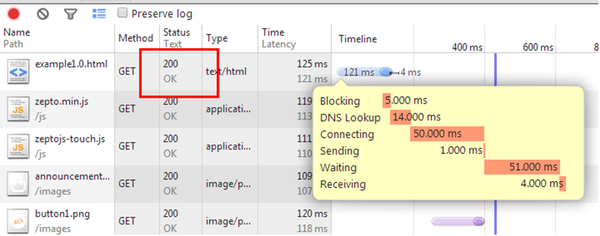
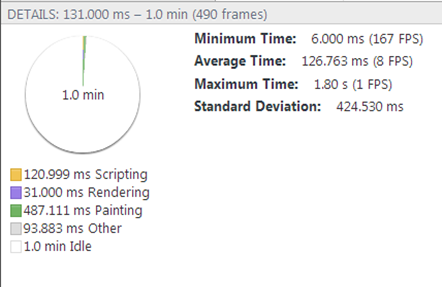
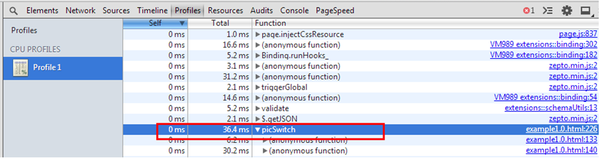
# 如何分析页面性能？

一、浏览器加载过程：  
1、建立连接过程  
  
(1) 浏览器查找域名的IP地址  
  
　　(2) 浏览器给web服务器发送一个HTTP请求  
  
　　(3) 服务器发送永久重定向响应  
  
　　(4) 浏览器跟踪重定向地址  
  
　　(5) 服务器“处理”请求  
  
　　(6) 服务器发回一个HTML响应  
2、浏览器渲染  
  
解析HTML  
↓  
构建DOM树  
↓  
渲染树构建  
↓  
渲染树布局  
↓  
绘制渲染树  
二、优化方法  
1、启用浏览器缓存  
　　浏览器要根据域名找出IP地址，而DNS查找过程的第一步是在浏览器缓存中查找，根据Chrome的PageSpeed给出的建议：要利用浏览器缓存存储可缓存的资源。在 HTTP 标头中为静态资源设置有效期或最长存在时间，可指示浏览器从本地磁盘中加载以前下载的资源而不是从网络中加载。  
HTTP 1.1提供的缓存方法主要有两种：  
(1) Expires和Cache-Control：max-age. 即内容在缓存中的生命有效期。第一次请求后将在生命有效期之内直接从本地缓存中拿取。  
　　(2) Last-Modified和ETag.：Last-Modified标记文件最后一次修改的时间，浏览器请求是在头部加入上次请求缓存下来的Last-Modified时间，若两次请求期间服务器的内容没有修改，服务器返回304 Not Modified，则不下载资源，浏览器直接使用本地缓存；否则，服务器会返回200以及更新后的版本。ETag是服务器对于文件生成的Hash散列，其生成算法与最后一次修改的时间相关。浏览器第二次请求发送上次的ETag信息，服务器通过简单的比对就知道是否应该返回304还是200。PageSpeed建议，要为资源指定Last-Modified或ETag标头，以便启动缓存验证。  
2、页面内容组织  
　　① 为 HTML 文档指定字符集，可让浏览器立即开始执行脚本。  
　　② 将内嵌样式块和 <link> 元素从文档主体移至文档标题，可改善显示性能。  
　　③ 将小型样式表或者脚本内嵌到主 HTML 网页中，可减少在下载其他资源时的往返时间 (RTT) 和延迟时间。  
④ 正确地排列外部样式表与外部和内嵌脚本的顺序，可增加下载时同时加载的数据量，并提高浏览器显示网页的速度。为确保能够并行下载这CSS 文件，始终将外部 CSS 文件排在外部 JavaScript 文件前面。   
PS：有图有真相  
<img src="https://pic1.zhimg.com/1d201894d45290fec37ad9d37f4b37b0\_b.jpg" data-rawwidth="417" data-rawheight="417" class="content\_image" width="417">VS：VS：  
<img src="https://pic3.zhimg.com/a12e36f60f950ff008d3674d93c98926\_b.jpg" data-rawwidth="410" data-rawheight="424" class="content\_image" width="410">  
3、HTML  
　　① 使用HTML5新标签，例如header、footer、section、nav、article。因为它们语义化，速度快，结构合理，浏览器上识别能力强。  
　　② 减少HTML标签嵌套深度，嵌套越深，在移动端的Web页面渲染速度以及滚动流畅度都会有所减低。  
　　③ 为图片指定大小，减少重排。  
　　④ 压缩HTML。  
4、 CSS  
　　① 使用CssSprite将零星的背景图包含到一张图中，通过background-position来使它显示在正确的位置；只请求一张图，减少了HTTP请求的次数。  
　　② 模块化、精简css，提高复用率。  
　　③ 减少渐变、阴影的使用。  
　　④ 合理使用CSS3高性能动画，Translate3d支持硬件加速。  
　　⑤ 避免使用滤镜。  
　　⑥ 不使用@import。  
　　⑦ 合并和压缩CSS代码。  
  
5、JavaScript  
　　① 暂缓 JavaScript 解析，暂缓解析不需要的 JavaScript（等到需要执行时再进行解析），可以提高网页的初始加载速度。  
　　② 使用事件委托机制，避免频繁操作DOM节点。  
　　③ 模块化代码：SeaJs。  
　　④ 合并和压缩JavaScript代码。  
　　⑤ 压缩工具：YUI Compressor或JSMin等。  
6、 图片优化  
① 图片时页面大部分加载时间所花的地方，在图片设计的时候应该考虑相应的图片大小和格式  
② 压缩图片  
三、实战分析  
1、 network  
<img src="https://pic3.zhimg.com/3a996de40148589cf5f5d80f7d04a22a\_b.jpg" data-rawwidth="674" data-rawheight="439" class="origin\_image zh-lightbox-thumb" width="674" data-original="https://pic3.zhimg.com/3a996de40148589cf5f5d80f7d04a22a\_r.jpg">2、status看资源请求情况2、status看资源请求情况  
<img src="https://pic3.zhimg.com/2716bb4ab251e514a5457d5a742e2ad6\_b.jpg" data-rawwidth="810" data-rawheight="319" class="origin\_image zh-lightbox-thumb" width="810" data-original="https://pic3.zhimg.com/2716bb4ab251e514a5457d5a742e2ad6\_r.jpg">3、timeline 可看页面渲染情况3、timeline 可看页面渲染情况  
<img src="https://pic4.zhimg.com/eaef805cabcfa48d9389fc4816e45433\_b.jpg" data-rawwidth="444" data-rawheight="289" class="origin\_image zh-lightbox-thumb" width="444" data-original="https://pic4.zhimg.com/eaef805cabcfa48d9389fc4816e45433\_r.jpg">4、profiles可看函数执行情况4、profiles可看函数执行情况  
<img src="https://pic3.zhimg.com/b1210e356668af38d784eb657e513d06\_b.jpg" data-rawwidth="949" data-rawheight="251" class="origin\_image zh-lightbox-thumb" width="949" data-original="https://pic3.zhimg.com/b1210e356668af38d784eb657e513d06\_r.jpg">四、工具四、工具  
1、Chrome Developer Tools —— 分析性能  
2、PageSpeed —— 查看性能建议  
　　3、Fiddler —— HTTP/HTTPS网络调试  
  
以上大多数总结自yahoo前端优化规则 & Chrome pagespeed的优化建议  
  
请指正。

## 转雅虎前端优化的几条军规

～～～有疑问欢迎讨论

雅虎给出了优化网站加载速度的34条法则（包括Yslow规则22条） 详细说明，下载转发 ponytail 的译文（来自帕兰映像）。  
  
1.Minimize HTTP Requests 减少HTTP请求  
  
图片、css、script、flash等等这些都会增加http请求数，减少这些元素的数量就能减少响应时间。把多个JS、CSS在可能的情况下写进一个文件，页面里直接写入图片也是不好的做法，应该写进CSS里，利用 CSS sprites 将小图拼合后利用background来定位。  
2.Use a Content Delivery Network 利用CDN技术  
  
CDN 确实是好东西，8过服务器提供商的这项服务一般是要收费的，我以前买的国内空间是有这个的但是我当时根本不知道啥用，现在没了。。。  
3.Add an Expires or a Cache-Control Header 设置头文件过期或者静态缓存  
  
浏览器会用缓存来减少http请求数来加快页面加载的时间，如果页面头部加一个很长的过期时间，浏览器就会一直缓存页面里的元素。不过这样如果 页面里的东西变动的话就要改名字了，否则用户端不会主动刷新，看自己衡量了~ 这项可以通过修改.htaccess文件来实现。  
4.Gzip Components Gzip压缩  
  
Gzip格式是一种很普遍的压缩技术，几乎所有的浏览器都有解压Gzip格式的能力，而且它可以压缩的比例非常大，一般压缩率为85%。压缩没压缩，可以到 这里 做下测试。  
5.Put Stylesheets at the Top 把CSS放顶部  
  
让浏览者能尽早的看到网站的完整样式。  
6.Put Scripts at the Bottom 把JS放底部  
  
网站呈现完毕后再进行功能设置，当然这些JS要在你的加载过程中不影响内容表现。  
  
7.Avoid CSS Expressions 避免CSS Expressions  
  
CSS表达式很可怕，这个只被IE支持的东西执行时候的运算量非常大，你移动一下鼠标它都要进行重计算的，但有时候为了做浏览器的兼容必须要用到这个||| IE6去死去死！~  
8.Make JavaScript and CSS External 将JS和CSS外链  
  
前面讲到了缓存这个事情，一些较为公用的JS和CSS，我们可以使用外链的形式，譬如我就是从Google外链来的Jquery文件，如果我的浏览者在浏览别的使用了这个外链文件的网站时已经下载并缓存了这个文件，那么他在浏览我的网站的时候就不需要再进行下载了！~  
9.Reduce DNS Lookups 减少DNS查找  
  
貌似是要减少网站从外部调用资源，我的Google分析和picasa的外链图片都算在里面了。  
10.Minify JavaScript and CSS 减小JS和CSS的体积  
  
写JS和CSS都是有技巧的，用最少的代码实现同样的功能，减少空白，增强逻辑性，用缩写方式等等，当然也有不少工具也能够帮你实现这一点。  
11. Avoid Redirects 避免重定向  
  
再写入链接时，虽然”[http://www](//link.zhihu.com/?target=http%3A//www" \t "_blank). today-s-ooxx. com”和”[http://www](//link.zhihu.com/?target=http%3A//www" \t "_blank). today-s-ooxx. com/” 仅有一个最后的”/”只差，但是结果是不同的，服务器需要花时间把前者重定向为后者然后进行跳转，这个要自己注意，也可以在Apache里用Alias 或者mod\_rewrite或者DirectorySlash解决。  
12. Remove Duplicate Scripts 删除重复脚本  
  
重复调用的代码浏览器并不会识别忽略，而是会再次运算一遍，这当然是大大的浪费。  
13. Configure ETags 配置ETags  
  
搞不清楚咋回事，总之我是在. htaccess里把它删除了。  
14. Make Ajax Cacheable 缓存Ajax  
  
Ajax是实时响应的，在浏览器接收到新的数据前，旧的数据被缓存，这样能够更好的提高效率。  
15. Flush the Buffer Early 尽早的释放缓冲  
  
当用户进行页面请求时，服务器端需要花费200到500毫秒时间来拼合HTML，将写在head与body之间，释放缓冲，这样可以将文件头先发送出去，然后再发送文件内容，提高效率。  
16. Use GET for AJAX Requests 用GET方式进行AJAX请求  
  
Get 方法和服务器只有一次交互（发送数据），而 Post 要两次（发送头部再发送数据）。  
17. Post-load Components 延迟加载组件  
  
最先加载必须的组件进行页面初始化，然后再加载其他，YUI Image Loader 是很好的例子。  
18. Preload components 预加载组件  
  
提前加载以后可能用到的东西，和延迟加载并不冲突，它的目的是为后续请求提供更快的响应，参见Google首页上的CSS sprites应用。  
19. Reduce the Number of DOM Elements 减少DOM元素数量  
  
复杂的页面结构意味着更长的下载及响应时间，更合理更高效的使用标签来架构页面，是好的前端的必备条件。  
20. Split Components Across Domains 跨域分离组件  
  
页面组件多个来源可以增大你的平行下载量，但注意不要过多，超过2-4个域名会引起上面说到的DNS查找浪费。  
21. Minimize the Number of iframes 减少iframe数量  
  
需要更有效的利用 ifames。  
iframe 优点：有利于下载缓慢的广告等第三方内容，安全沙箱，并行下载脚本  
iframe 缺点：即使为空也会有较大资源消耗，会阻止页面的onload，非语义  
22. No 404s 不要出现404页面  
  
站点本身里（非搜索结果）出现404页面，无意义的404页面会影响用户体验并且会消耗服务器资源。  
23. Reduce Cookie Size 减小Cookie  
  
Cookie在服务器及浏览器之间的通过文件头进行交换，尽可能减小Cookie体积，设置合理的过期时间，能够很好的提高效率。  
24. Use Cookie-free Domains for Components 对组件使用无Cookie的域名  
  
对静态组件的Cookie读取是一种浪费，使用另一个无Cookie的域名来存放你的静态组件式一个好方法，或者也可以在Cookie中只存放带www的域名。  
25. Minimize DOM Access 减少DOM的访问次数  
  
JS访问DOM是很慢的，尽量不要用JS来设置页面布局。  
26. Develop Smart Event Handlers 开发灵活的事件处理句柄  
  
DOM树上过多的元素被加入事件句柄的话，反应效率肯定会低，YUI事件工具有一个 onAvailable 方法可以帮助你灵活的设置DOM事件句柄  
27. Choose over @import 使用而非 @import  
  
在IE中使用@import就和在页面底部用一样，我们前面说要把放顶部的。  
28. Avoid Filters 避免过滤器的使用  
  
如果需要Alpha透明，不要使用AlphaImageLoader，它效率低下而且只对IE6及以下的版本适用，用PNG8图片。如果你非要使用，加上\_filter以免影响IE7+用户。  
29. Optimize Images 优化图片  
  
将你的GIF转为PNG8会是个减小体积的好办法，另外有很多方法处理你的JPG及PNG图片以达到优化效果。  
30. Optimize CSS Sprites 优化CSS Sprites  
  
在CSS Sprites中竖直并尽量紧凑的排列图片，尽量将颜色相似的图片排在一起，会减小图片本身的大小及提高页面图片显示速度。  
31. Don’t Scale Images in HTML 不要在HTML中缩放图片  
  
图片要用多大的就用多大的，1000X1000的图片被width=”100″ height=”100″以后，本身的KB数是不会减少的。  
32. Make favicon. ico Small and Cacheable 缩小favicon. ico的大小并缓存它  
  
站点的浏览器ICO应该不是经常换吧，那就长时间的缓存它，并且最好控制在1K以下。  
33. Keep Components under 25K 保证组件在25K以下  
  
iPhone不能缓存25K以上的组件，并且这还是要在被压缩前。  
34. Pack Components into a Multipart Document 将组件打包进一个多部分的文档中

作者：ScottLin  
链接：https://www.zhihu.com/question/21815101/answer/40131349  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。