目录

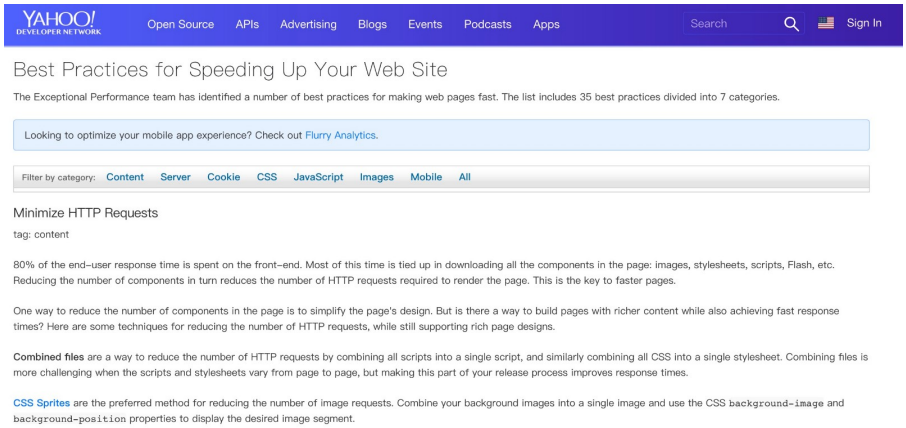
2 解读雅虎35条军规（上）

更新时间：2020-06-05 18:45:37

> 当你做成功一件事，千万不要等待着享受荣誉，应该再做那些需要的事。
>
> —— 巴斯德

相信大多数的前端开发者都接触过雅虎35条军规，可能有的没有实际去学习当中的理论知识，但是在日常的工作中你已经在使用了。

下面我们就来去解读一下雅虎的 35 条军规，这里我不会逐字逐句的去翻译，对一些比较拗口的，我就采用意译方式总结出来，同时我也会加入一些自己的理解。这里推荐英语比较好的同学去直接阅读原文，英文不太好的同学直接阅读我这里的解读就可以了。



上图就是原版的雅虎 35 条军规，大家不要觉得这是很神秘的技术，也就是雅虎开发人员日常总结出来的一些知识点，放到了他们的开发者网站，雅虎开发者网站还有很多其他有意思的技术总结，大家感兴趣可以去阅读，下面正式开始解读雅虎的 35 条军规。

1. Image(图片)

Optimize Images(优化图片)

- 检查 gif 图片的调色板大小是否匹配图片颜色数。
- 可以把 gif 转成 png 看看有没有变小。除了动画，gif 一般可以转成 png8。
- 运行 pngcrush 或其它工具压缩 png。
- 运行 jpegtran 或其它工具压缩 jpeg。

> Tips：这里推荐一个在线压缩图片网站，大家可以将开发当中遇到的较大的图片到这个网站压缩一下，再放到生产环境当中，至于图片格式的转换，推荐使用ps就好。

Optimize CSS Sprites(优化CSS雪碧图)

- 把图片横向合并而不是纵向，横向更小。
- 把颜色近似的图片合并到一张雪碧图，这样可以让颜色数更少，如果低于 256 就可以用 png8。
- `Be mobile-friendly` 并且合并时图片间的间距不要太大。这对图片大小影响不是太大，但客户端解压时需要的内存更少。100×100是10000个像素，1000×1000是1000000个像素。

Don’t Scale Images in HTML(禁止缩放图片在HTML文件当中)

不要在 HTML 中缩放图片

不要因为你可以设置图片的宽高就去用比你需要的大得多的图片。如果你需要100x100px的图片，那就不要用500x500px的。

```
<img width="100" height="100" src="mycat.jpg" alt="My Cat" />
```

Make favicon.ico Small and Cacheable(使 favicon小且缓存)

`favicon.ico` 是在你服务器根路径的图片。糟糕的是即使你不关心它，浏览器仍然会请求它。所以最好不要响应404。另外由于在同一服务器，每次请求 `favicon.ico` 时也会带上 cookie。这个图片还会影响下载顺序，比如在IE，如果你在 onload 时下载额外的组件，favicon会在这些组件之前被下载。

怎么减轻favicon.ico的缺点?

`favicon.ico` 的体积越小越好，最好1K以下。而且 favicon.ico 一般是不进行更换的，所以我们可以给它设置Expires头部，而且可以安全地设置为几个月，避免每一次打开页面都需要去进行请求。

> Tips： `favicon.ico` 就是我们打开的浏览器的tab上面的小图标，如果没有的话，控制台会报404错误。

2. CSS

Put Stylesheets at the Top(将css样式放在顶部)

关注性能的前端工程师希望页面被逐步渲染，这是因为，我们希望浏览器尽早渲染获取到的任何内容。这对大页面和网速慢的用户很重要。给用户视觉反馈，比如进度条的重要性已经被大量研究和记录。在我们的情况中，HTML 页面就是进度条。当浏览器逐步加载页面头部，导航条，logo 等等，这些都是给等待页面的用户的视觉反馈。这优化了整体用户体验。

把样式表放在文档底部的问题是它阻止了许多浏览器的逐步渲染，包括 IE。这些浏览器阻止渲染来避免在样式更改时需要重绘页面元素。所以用户会卡在白屏。

HTML规范清楚表明样式应该在 `<head>` 里。

> Tips：请注意这里提到的放到 `<head>` 标签里面指的是内联的形式，也就是使用 `<style>` 标签括起来的，而并非采用 `<link>` 的方式通过外联引入。

Avoid CSS Expressions(避免CSS表达式)

CSS 表达式是强大的（可能也是危险的）设置动态 CSS 属性的方法。IE5 开始支持，IE8 开始不赞成使用。例如，背景颜色可以设置成每小时轮换：

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

CSS 表达式的问题是它们可能比大多数人预期的计算的更频繁。它们不仅在页面载入和调整大小时重新计算，也在滚动页面甚至是用户在页面上移动鼠标时计算。比如在页面上移动鼠标可能轻易计算超过10000次。

要避免CSS表达式计算太多次，可以在它第一次计算后替换成确切值，或者用事件处理函数而不是CSS表达式。

Choose `<link>` over @import(选择 `<link>` 而不是@import)

之前的一个最佳原则是说 CSS 应该在顶部来允许逐步渲染。

在 IE 用 @import 和把 CSS 放到页面底部行为一致，所以最好别用。

Avoid Filters(避免使用（IE）过滤器)

IE专有的AlphaImageLoader过滤器用于修复IE7以下版本的半透明真彩色PNG的问题。这个过滤器的问题是它阻止了渲染，并在图片下载时冻结了浏览器。另外它还引起内存消耗，并且它被应用到每个元素而不是每个图片，所以问题的严重性翻倍了。

最佳做法是放弃 AlphaImageLoader，改用 PNG8 来优雅降级。

> Tips：其实IE浏览器现在绝大多数公司已经不再进行单独的适配，所以这条的使用场景非常有限。

Reduce Cookie Size(减少 cookie的体积)

http cookie 的使用有多种原因，比如授权和个性化。cookie 的信息通过 http 头部在浏览器和服务器端交换。尽可能减小cookie的大小来降低响应时间。

- 消除不必要的 cookie。
- 尽可能减小 cookie 的大小来降低响应时间。
- 注意设置 cookie 到合适的域名级别，则其它子域名不会被影响。
- 正确设置 Expires 日期。早一点的 Expires 日期或者没有尽早的删除 cookie，优化响应时间。

Use Cookie-free Domains for Components(对组件使用无Cookie域)

当浏览器请求静态图片并把 cookie 一起发送到服务器时，cookie 此时对服务器没什么用处。所以这些 cookie 只是增加了网络流量。所以你应该保证静态组件的请求是没有 cookie 的。可以创建一个子域名来托管所有静态组件。

比如，你域名是**www.example.org**，可以把静态组件托管在**static.example.org**。不过，你如果把cookie设置在顶级域名 example.org 下，这些cookie仍然会被传给static.example.org。这种情况下，启用一个全新的域名来托管静态组件。

另外一个用没有 cookie 的域名提供组件的好处是，某些代理可能会阻止缓存带 cookie 的静态组件请求。

> Tips：这里说明上面说的为什么要设置一个全新域名来托管静态组件，因为cookie是可以跨二级域名的，[所以如果你设置的顶级域名是**example.org**，那么**static.example.org**也是可以被访问到的，因此需要启用一个全新的域名。

4.Server(服务端)

Use a Content Delivery Network(使用CDN)

用户接近你的服务器会减少响应时间。把你的内容发布到多个地理上分散的服务器可以让页面加载更快。但怎么开始?

首先不要试图把你的架构重新设计成分布式架构。因为可能引进更多复杂性和不可控。

记住80-90%的终端用户响应时间花费在下载页面中的所有组件：图片、样式、脚本、falsh 等等。这是_Performance Golden Rule_。不要从困难的重新设计后台架构开始，最好首先分发你的静态内容。这不仅可以减少响应时间，用CDN还很容易来做。

CDN 是一群不同地点的服务器，可以更高效地分发内容到用户。一些大公司有自己的 CDN。

Add an Expires or a Cache-Control Header(加Expires或者Cache-Control头部)

这条规则有两个方面：

- 对静态组件：通过设置 Expires 头部来实现“永不过期”策略。
- 对动态组件：用合适的 Cache-Control 头部来帮助浏览器进行有条件请求。

一般用在图片上，但应该用在所有的组件上。

浏览器（以及代理）使用缓存来减少http请求数，加快页面加载。服务器使用http响应的Expires头部来告诉客户端一个组件可以缓存多久。比如下面：

```
Expires: Thu, 15 Apr 2010 20:00:00 GMT  // 2010-04-15之前都是稳定的
```

> Tips：注意，如果你设置了Expires头部，当组件更新后，你必须更改文件名。

Gzip Components(传输时用gzip等压缩组件)

http 请求或响应的传输时间可以被前端工程师显著减少。终端用户的带宽，ISP，接近对等交换点等等没法被开发团队控制，但是，压缩可以通过减少 http 响应的大小减少响应时间。

从 HTTP/1.1 开始，客户端通过http请求中的 Accept-Encoding 头部来提示支持的压缩：

```
Accept-Encoding: gzip, deflate
```

如果服务器看到这个头部，它可能会选用列表中的某个方法压缩响应。服务器通过Content-Encoding 头部提示客户端：

```
Content-Encoding: gzip
```

gzip 一般可减小响应的 70%。尽可能去gzip更多（文本）类型的文件。html，脚本，样式，xml和json 等等都应该被gzip，而图片，pdf等等不应该被gzip，因为它们本身已被压缩过，gzip 它们只是浪费 cpu，甚至增加文件大小。

Configure ETags(ETags 配置)

实体标记（Entity tags，ETag）是服务器和浏览器之间判断浏览器缓存中某个组件是否匹配服务器端原组件的一种机制。实体就是组件：图片，脚本，样式等等。ETag被当作验证实体的比最后更改（last-modified）日期更高效的机制。服务器这样设置组件的ETag：

```
HTTP/1.1 200 OK
Last-Modified: Tue, 12 Dec 2006 03:03:59 GMT
ETag: "10c24bc-4ab-457e1c1f"
Content-Length: 12195
```

之后，如果浏览器要验证组件，它用 If-None-Match 头部来传 ETag 给服务器。如果 ETag 匹配，服务器返回304：

```
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
```

方案。

如果不能解决多服务器间的 ETag 匹配问题，那么删除 ETag 可能更好。

Flush the Buffer Early(早一点刷新buffer)

当用户请求一个页面，服务器一般要花 200-500ms 来拼凑整个页面。这段时间，浏览器是空闲的（等数据返回）。在 php，有个方法 flush() 允许你传输部分准备好的 html 响应给浏览器。这样的话浏览器就可以开始下载组件，而同时后台可以继续生成页面剩下的部分。这种好处更多是在忙碌的后台或轻前端网站可以看到。

一个比较好的 flush 的位置是在 head 之后，因为浏览器可以加载其中的样式和脚本文件，而后台继续生成页面剩余部分。

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

Use GET for AJAX Requests(ajax请求用get)

Yahoo! Mail团队发现当使用 XMLHttpRequest，POST 被浏览器实现为两步：首先发送头部，然后发送数据。所以使用 GET 最好，仅用一个 TCP 包发送（除非cookie太多）。IE的url长度限制是2K。

POST 但不提交任何数据根 GET 行为类似，但从语义上讲，获取数据应该用 GET，提交数据到服务器用 POST。

Avoid Empty Image src(避免空src的图片)

空 `src` 属性的图片的行为可能跟你预期的不一样。它有两种形式：

- html标签： `<img src="">`
- js： `var img = new Image(); img.src = "";`

上面的两种形式都会造成同一种后果：浏览器会向你的服务器发请求。

IE向页面所在的目录发请求。 Safari 和 Chrome 请求实际的页面。 FireFox3 及之前和 Safari/Chrome 一样，但从3.5开始修复问题不再发送请求。 Opera 遇到空图片 src 不做任何事。 为什么这种行为很糟糕?

因为由于发送大量的意料之外的流量，会削弱服务器。尤其那些每天页面浏览量上百万的页面。浪费服务器计算周期去生成不会被浏览的页面，可能会破坏用户数据。如果你在跟踪请求状态，通过 cookie 或其它，你可能会破坏数据。即使 image 的请求不会返回图片，但所有的头部数据都被浏览器读取了，包括 cookie。即使剩下的响应体被丢弃，破坏可能已经发生。 这种行为的根源是 uri 解析发生在浏览器。

RFC 3986 定义了这种行为，空字符串被当作相对路径，Firefox, Safari, 和 Chrome都正确解析，而IE错误。总之，浏览器解析空字符串为相对路径的行为被认为是符合预期的。