[前端性能优化----yahoo前端性能团队总结的35条黄金定律](http://www.cnblogs.com/xiaobai110/p/6430488.html)



**网页内容**

[**减少http请求次数**](http://developer.yahoo.com/performance/rules.html#num_http)

80%的响应时间花在下载网页内容(images, stylesheets, javascripts, scripts, flash等)。**减少请求次数**是缩短响应时间的关键！可以通过简化页面设计来减少请求次数，但页面内容较多可以采用以下技巧。

**1. 捆绑文件**: 现在有很多现成的库可以帮你将多个脚本文件捆绑成一个文件，将多个样式表文件捆绑成一个文件，以此来减少文件的下载次数。例如在asp.net中可以使用[ScriptManager](http://msdn.microsoft.com/en-us/library/cc488552%28v=vs.90%29.aspx" \t "_blank)，asp.net MVC中的[Bundling](http://www.asp.net/mvc/tutorials/mvc-4/bundling-and-minification" \t "_blank)。

2. [**CSS Sprites**](http://www.w3schools.com/css/css_image_sprites.asp): 就是把多个图片拼成一副图片，然后通过CSS来控制在什么地方具体显示这整张图片的什么位置。给大家看个熟悉的Sprites实例。

豆瓣把他的图标集中在一起，然后我们看他如何控制只显示第一个图标的

[复制代码](javascript:void(0);)

复制代码

复制代码

复制代码

.app-icon-read {

background-position

: 0 0;

}

.app-icon {

background

: url("/pics/app/app\_icons\_50\_5.jpg") no-repeat scroll 0 0 transparent;

border-radius: 10px 10px 10px 10px;

box-shadow: 1px 1px 2px #999999;

display: inline-block;

height

: 50px;

width

: 50px;

}

复制代码

复制代码

复制代码

[复制代码](javascript:void(0);)

3. [**Image Maps**](http://en.wikipedia.org/wiki/Image_map)： 也是将多幅图拼在一起，然后通过坐标来控制显示导航。这里有个经典的例子，选中图片中的某个人就会将你带到不同的链接。

4. [**Inline images**](https://en.wikipedia.org/wiki/Data_URI_scheme): 通过编码的字符串将图片内嵌到网页文本中。例如下面的inline image的显示效果为一个勾选的checkbox。

[复制代码](javascript:void(0);)

复制代码

复制代码

复制代码

.sample-inline-png {

padding-left: 20px;

background: white url('data:image/png;base64,iVBORw0KGgoAA

AANSUhEUgAAABAAAAAQAQMAAAAlPW0iAAAABlBMVEUAAAD///+l2Z/dAAAAM0l

EQVR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83NDDeNGe4Ug9C9zwz3gVLMDA/A6

P9/AFGGFyjOXZtQAAAAAElFTkSuQmCC') no-repeat scroll left top;

}

复制代码

复制代码

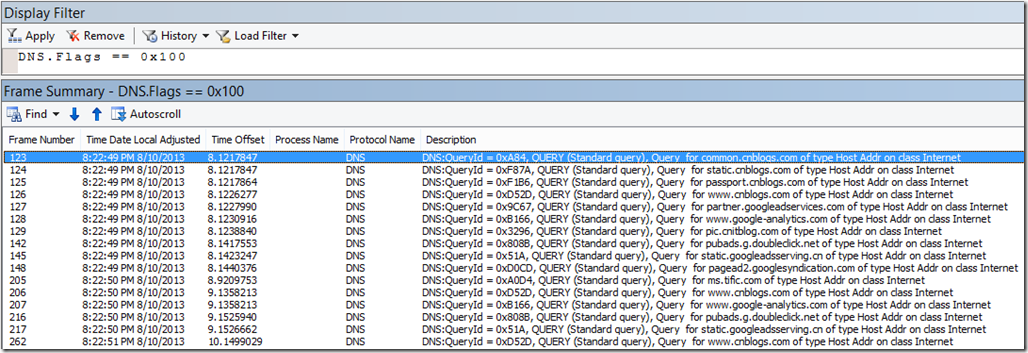
复制代码

[复制代码](javascript:void(0);)

图片显示效果如左图

[**减少DNS查询次数**](http://developer.yahoo.com/performance/rules.html#dns_lookups)

DNS查询也消耗响应时间，如果我们的网页内容来自各个不同的domain (比如嵌入了开放广告，引用了外部图片或脚本)，那么客户端首次解析这些domain也需要消耗一定的时间。DNS查询结果缓存在本地系统和浏览器中一段时间，所以DNS查询一般是对首次访问响应速度有所影响。下面是我清空本地dns后访问博客园主页dns的查询请求。看少去还不少哦。

[](https://images0.cnblogs.com/blog/502305/201308/10203233-1992fc4f7daf42a7b7b013158cc22eb2.png)

[**避免页面跳转**](http://developer.yahoo.com/performance/rules.html#redirects)

当客户端收到服务器的跳转回复时，客户端再次根据服务器回复中的location指定的地址再次发送请求，例如以下跳转回复。

HTTP/1.1 301 Moved Permanently

Location: http://example.com/newuri

Content-Type: text/html

当客户端遇到这种回复的时候，用户只能等待客户端再次发送请求，有的网站甚至会一直跳n次，跳到他想带你去的地方…当然在这个时候用户看不到任何页面内容，只有浏览器的进度条一直在刷新。

[**缓存Ajax**](http://developer.yahoo.com/performance/rules.html#cacheajax)

Ajax可以帮助我们异步的下载网页内容，但是有些网页内容即使是异步的，用户还是在等待它的返回结果，例如ajax的返回是用户联系人的下拉列表。所以我们还是要注意尽量应用以下规则提高ajax的响应速度。

* 添加Expires 或 Cache-Control报文头使回复可以被客户端缓存
* 压缩回复内容
* 减少dns查询
* 精简javascript
* 避免跳转
* 配置Etags

[**延迟加载**](http://developer.yahoo.com/performance/rules.html#postload)

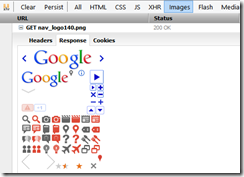
这里讨论延迟加载需要我们知道我们的**网页最初加载需要的最小内容集**是什么。剩下的内容就可以推到延迟加载的集合中。

Javascript是典型的可以延迟加载内容。一个比较激进的做法是开发网页时先确保网页在没有Javascript的时候也可以基本工作，然后通过延迟加载脚本来完成一些高级的功能。

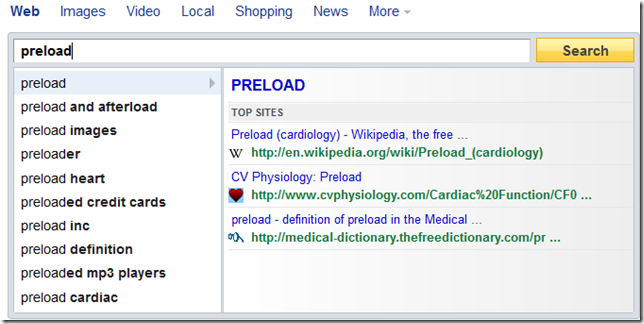
[**提前加载**](http://developer.yahoo.com/performance/rules.html#preload)

与延迟加载目的相反，提前加载的是为了提前加载接下来网页中访问的资源，下面是提前加载的类型

无条件提前加载：当前网页加载完成后，马上去下载一些其他的内容。例如google会在页面加载成功之后马上去下载一个所有结果中会用到的image sprite。

[](https://images0.cnblogs.com/blog/502305/201308/10214950-d7017f547cd14712a1fe2a7417cc8394.png)

有条件加载：根据用户的输入推断需要加载的内容，雅虎的示例是[search.yahoo.com](http://search.yahoo.com/)，

[](https://images0.cnblogs.com/blog/502305/201308/10221014-fd6a6979fcef46968ebcc3e97398c24e.png)

有预期的的加载：这种情况一般发生在网页重新设计时，由于用户经常访问旧网页，本地对旧的网页内容缓存充分从而显得旧网页速度很快，而新的网页内容却没有缓存，设计者可以在旧网页的内容中预先加载一些新网页中可能用到的内容，这样新的网页就会生下来一些需要下载的资源。

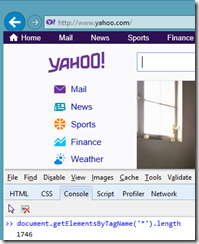
[**减少DOM元素数量**](http://developer.yahoo.com/performance/rules.html#min_dom)

网页中元素过多对网页的加载和脚本的执行都是沉重的负担，500个元素和5000个元素在加载速度上会有很大差别。

想知道你的网页中有多少元素，通过在浏览器中的一条简单命令就可以算出，

document.getElementsByTagName('\*').length

多少算是多了呢？雅虎在写这篇文章的时候号称主页只有700多元素，但现在接近多了一倍。我们的网页至少别比雅虎还多吧。。。

[](https://images0.cnblogs.com/blog/502305/201308/10221015-2e97f37f61144b29b657997ebed05b06.png)

[**根据域名划分内容**](http://developer.yahoo.com/performance/rules.html#split)

浏览器一般对同一个域的下载连接数有所限制，按照域名划分下载内容可以浏览器增大并行下载连接，但是注意控制域名使用在2-4个之间，不然dns查询也是个问题。

一般网站规划会将静态资源放在类似于static.example.com，动态内容放在[www.example.com](http://www.example.com/)上。这样做还有一个好处是可以在静态的域名上避免使用cookie。后面我们会在cookie的规则中提到。

[**减少iframe数量**](http://developer.yahoo.com/performance/rules.html#iframes)

使用iframe要注意理解iframe的优缺点

优点

* 可以用来加载速度较慢的内容，例如广告。
* [安全沙箱保护](http://www.html5rocks.com/en/tutorials/security/sandboxed-iframes/)。浏览器会对iframe中的内容进行安全控制。
* 脚本可以并行下载

缺点

* 即使iframe内容为空也消耗加载时间
* 会阻止页面加载
* [没有语义](http://www.w3schools.com/html/html5_semantic_elements.asp)

[**避免404**](http://developer.yahoo.com/performance/rules.html#no404)

404我们都不陌生，代表服务器没有找到资源，我们要特别要注意404的情况不要在我们提供的网页资源上，客户端发送一个请求但是服务器却返回一个无用的结果，时间浪费掉了。

更糟糕的是我们网页中需要加载一个外部脚本，结果返回一个404，不仅阻塞了其他脚本下载，下载回来的内容(404)客户端还会将其当成Javascript去解析。

**服务器**

[**使用CDN**](http://developer.yahoo.com/performance/rules.html#cdn)

再次强调第一条黄金定律，减少网页内容的下载时间。提高下载速度还可以通过CDN(内容分发网络)来提升。CDN通过部署在不同地区的服务器来提高客户的下载速度。如果你的网站上有大量的静态内容，世界各地的用户都在访问，我说的是youtube么？那CDN是必不可少的。事实上大多数互联网中的巨头们都有自己的CDN。我们自己的网站可以先通过[免费的CDN供应商](http://en.wikipedia.org/wiki/Content_delivery_network" \l "Free_CDNs" \t "_blank)来分发网页资源。

[**添加Expires 或Cache-Control报文头**](http://developer.yahoo.com/performance/rules.html#expires)

这条规则分为两个方面，

* 对于静态内容添加Expires，将静态内容设为永不过期，或者很长时间以后。在IIS中设置Expires可以看[Configure the HTTP Expires Response Header (IIS 7)](http://technet.microsoft.com/en-us/library/cc770661%28v=WS.10%29.aspx" \t "_blank)。
* 对于动态内容应用合适的Cache-Control，让浏览器根据条件来发送请求。关于asp.net的caching，可以看[asp.net cache feature](http://msdn.microsoft.com/en-us/library/xsbfdd8c%28v=vs.100%29.aspx" \t "_blank)和[asp.net caching best practices](http://msdn.microsoft.com/en-us/library/aa478965.aspx" \t "_blank)。

[**Gzip压缩传输文件**](http://developer.yahoo.com/performance/rules.html#gzip)

Gzip通常可以减少70%网页内容的大小，包括脚本、样式表、图片等文件。Gzip比deflate更高效，主流服务器都有相应的压缩支持模块。

IIS中内建了静态压缩和动态压缩模块，如何配制可以参考[Enable HTTP Compression of Static Content (IIS 7)](http://technet.microsoft.com/en-us/library/cc754668%28v=WS.10%29.aspx" \t "_blank)和[Enable HTTP Compression of Dynamic Content (IIS 7)](http://technet.microsoft.com/en-us/library/cc753681%28v=WS.10%29.aspx" \t "_blank)。

值得注意的是pdf文件可以从需要被压缩的类型中剔除，因为pdf文件本身已经压缩，gzip对其效果不大，而且会浪费CPU。

[**配置ETags**](http://developer.yahoo.com/performance/rules.html#etags)

虽然标题叫配制ETags，但是这里你要根据具体情况进行一些判断。首先Etag简单来说是通过一个文件版本标识使得服务器可以轻松判断该请求的内容是否有所更新，如果没有就回复304 (not modified)，从而避免下载整个文件。

但是Etags的版本信息即使主流服务器未能很好地支持跨服务器的判断，比如你从一个服务器集群中一台得到Etags，然后发送到了另一台那么校验很有可能会失败。

如果你遇到这样的问题，IIS 7中可以通过如下方法将Etag去掉，使用URL Rewrite，然后在web.config中添加如下配制

[复制代码](javascript:void(0);)

复制代码

复制代码

复制代码

<rewrite>

<outboundRules>

<rule name="Remove ETag">

<match serverVariable="RESPONSE\_ETag" pattern=".+" />

<action type="Rewrite" value="" />

</rule>

</outboundRules>

</rewrite>

复制代码

复制代码

复制代码

[复制代码](javascript:void(0);)

IIS8里提供了一个简单配制来直接关闭Etag，

[复制代码](javascript:void(0);)

复制代码

复制代码

复制代码

<element name="clientCache">

<attribute name="cacheControlMode" type="enum" defaultValue="NoControl">

<enum name="NoControl" value="0" />

<enum name="DisableCache" value="1" />

<enum name="UseMaxAge" value="2" />

<enum name="UseExpires" value="3" />

</attribute>

<attribute name="cacheControlMaxAge" type="timeSpan" defaultValue="1.00:00:00" />

<attribute name="httpExpires" type="string" />

<attribute name="cacheControlCustom" type="string" />

<attribute name="setEtag" type="bool" defaultValue="false" />

</element>

复制代码

复制代码

复制代码

[复制代码](javascript:void(0);)

[**尽早flush输出**](http://developer.yahoo.com/performance/rules.html#flush)

网页后台程序中我们知道有个方法叫[Response.Flush()](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.flush.aspx)，一般我们调用它都是在程序末尾，但注意这个方法可以被调用多次。目的是可以将现有的缓存中的回复内容先发给客户端，让客户端“有活干”。

那在什么时候调用这个方法比较好呢？一般情况下我们可以在对于需要加载比较多外部脚本或者样式表时可以提前调用一次，客户端收到了关于脚本或其他外部资源的链接可以并行的先发请求去下载，服务器接下来把后续的处理结果发给客户端。

[**使用GET Ajax请求**](http://developer.yahoo.com/performance/rules.html#ajax_get)

浏览器在实现XMLHttpRequest POST的时候分成两步，先发header，然后发送数据。而GET却可以用一个TCP报文完成请求。另外GET从语义上来讲是去服务器取数据，而POST则是向服务器发送数据，所以我们使用Ajax请求数据的时候尽量通过GET来完成。

关于GET和POST的详细对比可以查看[这里](http://www.w3schools.com/tags/ref_httpmethods.asp" \t "_blank)。

[**避免空的图片src**](http://developer.yahoo.com/performance/rules.html#emptysrc)

空的图片src仍然会使浏览器发送请求到服务器，这样完全是浪费时间，而且浪费服务器的资源。尤其是你的网站每天被很多人访问的时候，这种空请求造成的伤害不容忽略。

浏览器如此实现也是根据RFC 3986 - Uniform Resource Identifiers标准，空的src被定义为当前页面。

所以注意我们的网页中是否存在这样的代码

**straight HTML**   
<img src="">

**JavaScript**   
var img = new Image();   
img.src = "";

**Cookie**

[**减少Cookie大小**](http://developer.yahoo.com/performance/rules.html#cookie_size)

Cookie被用来做认证或个性化设置，其信息被包含在http报文头中，对于cookie我们要注意以下几点，来提高请求的响应速度，

* 去除没有必要的cookie，如果网页不需要cookie就完全禁掉
* 将cookie的大小减到最小
* 注意cookie设置的domain级别，没有必要情况下不要影响到sub-domain
* 设置合适的过期时间，比较长的过期时间可以提高响应速度。

关于asp.net中的cookie可以参考[ASP.NET Cookies Overview](http://msdn.microsoft.com/en-us/library/ms178194%28v=vs.100%29.aspx" \t "_blank)和[Configure Use Cookies Mode for Session State (IIS 7)](http://technet.microsoft.com/en-us/library/cc754725%28v=ws.10%29.aspx" \t "_blank)

[**页面内容使用无cookie域名**](http://developer.yahoo.com/performance/rules.html#cookie_free)

大多数网站的静态资源都没必要cookie，我们可以采用不同的domain来单独存放这些静态文件，这样做不仅可以减少cookie大小从而提高响应速度，还有一个好处是有些proxy拒绝缓存带有cookie的内容，如果能将这些静态资源cookie去除，那就可以得到这些proxy的缓存支持。

常见的划分domain的方式是将静态文件放在static.example.com，动态内容放在[www.example.com](http://www.example.com/)。

也有一些网站需要在二级域名上应用cookie，所有的子域都会继承，这种情况下一般会再购买一个专门的域名来存放cookie-free的静态资源。例如Yahoo!的yimg.com，YouTube的ytimg.com等。

**CSS**

[**将样式表置顶**](http://developer.yahoo.com/performance/rules.html#css_top)

经样式表(css)放在网页的HEAD中会让网页**显得**加载速度更快，因为这样做可以使浏览器逐步加载已将下载的网页内容。这对内容比较多的网页尤其重要，用户不用一直等待在一个白屏上，而是可以先看已经下载的内容。

如果将样式表放在底部，浏览器会拒绝渲染已经下载的网页，因为大多数浏览器在实现时都努力避免重绘，样式表中的内容是绘制网页的关键信息，没有下载下来之前只好对不起观众了。

[**避免CSS表达式**](http://developer.yahoo.com/performance/rules.html#css_expressions)

CSS表达式可以动态的设置CSS属性，在**IE5-IE8**中支持，**其他浏览器中表达式会被忽略**。例如下面表达式在不同时间设置不同的背景颜色。

复制代码

复制代码

CSS表达式的问题在于它被重新计算的次数远比我们想象的要多，不仅在网页绘制或大小改变时计算，即使我们滚动屏幕或者移动鼠标的时候也在计算，因此我们还是尽量避免使用它来防止使用不当而造成的性能损耗。

如果想达到类似的效果我们可以通过简单的脚本做到。

[复制代码](javascript:void(0);)

复制代码

复制代码

复制代码

<html>

<head>

</head>

<body>

<script type="text/javascript">

var currentTime = new Date().getHours();

if (currentTime%2) {

if (document.body) {

document.body.style.background = "#B8D4FF";

}

}

else {

if (document.body) {

document.body.style.background = "#F08A00";

}

}

</script>

</body>

</html>

复制代码

复制代码

复制代码

[复制代码](javascript:void(0);)

[**用<link>代替@import**](http://developer.yahoo.com/performance/rules.html#csslink)

避免使用@import的原因很简单，因为它相当于将css放在网页内容底部。

[**避免使用Filters**](http://developer.yahoo.com/performance/rules.html#no_filters)

[AlphaImageLoad](http://msdn.microsoft.com/en-us/library/ms532969%28v=vs.85%29.aspx)也是IE5.5 - IE8中支持，这种滤镜的使用会导致图片在下载的时候阻塞网页绘制，另外使用这种滤镜会导致内存使用量的问题。IE9中已经不再支持。

**Javascript**

[**将脚本置底**](http://developer.yahoo.com/performance/rules.html#js_bottom)

[HTTP/1.1 specification](http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.1.4)建议浏览器对同一个hostname不要超过两个并行下载连接， 所以当你从多个domain下载图片的时候可以提高并行下载连接数量。但是当脚本在下载的时候，即使是来自不同的hostname浏览器也不会下载其他资源，因为浏览器要在脚本下载之后依次解析和执行。

因此对于脚本提速，我们可以考虑以下方式，

* 把脚本置底，这样可以让网页渲染所需要的内容尽快加载显示给用户。
* 现在主流浏览器都支持[defer](http://www.w3schools.com/tags/att_script_defer.asp)关键字，可以指定脚本在文档加载后执行。
* HTML5中新加了[async](http://www.w3schools.com/tags/att_script_async.asp" \t "_blank)关键字，可以让脚本异步执行。

[**使用外部Javascirpt和CSS文件**](http://developer.yahoo.com/performance/rules.html#external)

使用外部Javascript和CSS文件可以使这些文件被浏览器缓存，从而在不同的请求内容之间重用。

同时将Javascript和CSS从inline变为external也减小了网页内容的大小。

使用外部Javascript和CSS文件的决定因素在于这些外部文件的重用率，如果用户在浏览我们的页面时会访问多次相同页面或者可以重用脚本的不同页面，那么外部文件形式可以为你带来很大的好处。但对于用户通常只会访问一次的页面，例如microsoft.com首页，那inline的javascript和css相对来说可以提供更高的效率。

[**精简Javascript和CSS**](http://developer.yahoo.com/performance/rules.html#minify)

精简就是将Javascript或CSS中的空格和注释全去掉，

[复制代码](javascript:void(0);)

复制代码

复制代码

复制代码

body {

line-height: 1;

}

ol, ul {

list-style: none;

}

blockquote, q {

quotes: none;

}

复制代码

复制代码

复制代码

[复制代码](javascript:void(0);)

精简后版本

复制代码

body{line-height:1}ol,ul{list-style:none}blockquote,q{quotes:none}

复制代码

统计表明精简后的文件大小平均减少了21%，即使在应用Gzip的文件也会减少5%。

例如我的网站上有5个CSS，4个Javascirpt，下面是分别经过bundling和minify之后的结果。

|  |  |  |
| --- | --- | --- |
| 没有任何处理之前 | 捆绑Javascript和CSS之后 | 精简Javascript和CSS之后 |
| [6](https://images0.cnblogs.com/blog/502305/201308/11162116-a63c6e9d45e44362aec24456f7f66a7e.png) | [7](https://images0.cnblogs.com/blog/502305/201308/11162116-ec89759b24124aae80db35f1712215ae.png) | [8](https://images0.cnblogs.com/blog/502305/201308/11162117-b450aed6cf5c4b76b5d33b7ca4e6ddfa.png) |

用来帮助我们做精简的工具很多，主要可以参考如下，

JS compressors:

* [Packer](http://dean.edwards.name/packer/)
* [JSMin](http://crockford.com/javascript/jsmin)
* [Closure compiler](http://code.google.com/intl/pl/closure/compiler/)
* [YUICompressor](http://developer.yahoo.com/yui/compressor/) (also does CSS)
* [AjaxMin](http://ajaxmin.codeplex.com/) (also does CSS)

CSS compressors:

* [CSSTidy](http://csstidy.sourceforge.net/)
* [Minify](http://code.google.com/p/minify/)
* [YUICompressor](http://developer.yahoo.com/yui/compressor/) (also does JS)
* [AjaxMin](http://ajaxmin.codeplex.com/) (also does JS)
* [CSSCompressor](http://www.csscompressor.com/)

与VS集成比较好的工具如下.

* [YUICompressor](http://yuicompressor.codeplex.com/) - 编译集成，包含在[NuGet](http://nuget.org/List/Packages/YUICompressor.NET).
* [AjaxMin](http://ajaxmin.codeplex.com/)  - 编译集成

[**去除重复脚本**](http://developer.yahoo.com/performance/rules.html#js_dupes)

重复的脚本不仅浪费浏览器的下载时间，而且浪费解析和执行时间。一般用来避免引入重复脚本的做法是使用统一的脚本管理模块，这样不仅可以避免重复脚本引入，还可以兼顾脚本依赖管理和版本管理。

[**减少DOM访问**](http://developer.yahoo.com/performance/rules.html#dom_access)

通过Javascript访问DOM元素没有我们想象中快，元素多的网页尤其慢，对于Javascript对DOM的访问我们要注意

* 缓存已经访问过的元素
* Offline更新节点然后再加回DOM Tree
* 避免通过Javascript修复layout

[**使用智能事件处理**](http://developer.yahoo.com/performance/rules.html#events)

这里说智能的事件处理需要开发者对事件处理有更深入的了解，通过不同的方式尽量少去触发事件，如果必要就尽早的去处理事件。

比如一个div中10个按钮都需要事件句柄，那么我们可以将事件放在div上，在事件冒泡过程中捕获该事件然后判断事件来源。

**图片**

[**优化图像**](http://developer.yahoo.com/performance/rules.html#opt_images)

当美工完成了网站的图片设计后，我们可以在上传图片之前对其做以下优化

* 检查GIF图片中图像颜色的数量是否和调色板规格一致。如果你发现图片中只用到了4种颜色，而在调色板的中显示的256色的颜色槽，那么这张图片就还有压缩的空间。可以使用[imagemagick](http://www.imagemagick.org/" \t "_blank)检查：   
  identify -verbose image.gif
* 尝试把GIF格式转换成PNG格式，看看是否节省空间。大多数情况下是可以压缩的。下面这条简单的命令可以安全地把GIF格式转换为PNG格式：   
  convert image.gif image.png
* 在所有的PNG图片上运行[pngcrush](http://pmt.sourceforge.net/pngcrush/" \t "_blank)（或者其它PNG优化工具）。例如：   
  pngcrush image.png -rem alla -reduce -brute result.png
* 在所有的JPEG图片上运行[jpegtran](http://jpegclub.org/jpegtran/" \t "_blank)。这个工具可以对图片中的出现的锯齿等做无损操作，同时它还可以用于优化和清除图片中的注释以及其它无用信息   
  jpegtran -copy none -optimize -perfect src.jpg dest.jpg

[**优化CSS Sprite**](http://developer.yahoo.com/performance/rules.html#opt_sprites)

* Spirite中水平排列图片，垂直排列会增加文件大小；
* 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；
* 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，1000×1000就是100万像素。

[**不要在HTML中缩放图片**](http://developer.yahoo.com/performance/rules.html#no_scale)

不要通过图片缩放来适应页面，如果你需要小图片，就直接使用小图片吧。

[**使用小且可缓存的favicon.ico**](http://developer.yahoo.com/performance/rules.html#favicon)

网站图标文件favicon.ico，不管你服务器有还是没有，浏览器都会去尝试请求这个图标。所以我们要确保这个图标

* 存在
* 文件尽量小，最好小于1k
* 设置一个长的过期时间

**移动客户端**

[**保持单个内容小于25KB**](http://developer.yahoo.com/performance/rules.html#under25)

这限制是因为iphone，他只能缓存小于25K，注意**这是解压后的大小**。所以单纯gzip不一定够用，精简文件工具要用上了。

[**打包组建成符合文档**](http://developer.yahoo.com/performance/rules.html#multipart)

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请求中取得多个组建。当你使用这条规则时，首先要确定用户代理是否支持（iPhone不支持）。

ths:

<http://www.cnblogs.com/developersupport/p/webpage-performance-best-practices.html>