## 高性能网站建设指南

### HTTP概述

HTTP是一种客户端与服务器端交换数据的协议，它由请求和响应组成，如果浏览器和服务器都支持的话，可以使用压缩来减小响应的大小，浏览器可以使用Accept-Encoding头来声明它支持压缩，服务器使用Content-Encoding头来确认被压缩

### 最佳实践1：减少HTTP请求

1. 使用图片地图

相邻的多个图片可以裁剪到同一张图片，通过一次请求取得资源，然后根据图片不同位置来控制对用户的响应

1. CSS Sprites

CSS Sprites就是将多个图片整合到一个图片上，通过一次请求取得，然后在网页中通过调节图片显示窗口的大小和图片的相对位置来显示指定图片的一种解决方案

1. 合并脚本和样式表

### 最佳实践2：使用内容发布网络

内容发布网络（CDN）是一组分布在不同地理位置的Web服务器，使用CDN服务后，用户能够访问就近的服务器，这样就能在最短的时间内得到响应。

尽量使用CDN缓存静态资源，因为动态的HTML会引入其他的需求——数据库连接，状态管理、验证、硬件和OS优化等，这些复杂性超过了CDN的能力范围

### 最佳实践3：添加Expires头

添加Expires头可以让一部分资源缓存在浏览器端直到过了指定的时间，这样用户下次访问时在缓存有效时间内就不必再请求已经缓存的资源

在图片上使用Expires头是非常普遍的做法，但这一最佳实践不应该仅仅限于图片，任何不经常变化的组件，包括脚本、样式表和Flash组件。但HTML文档不应该给使用长久的Expires头，因为它包含动态内容，这些内容在用户每次请求时都会更新。

如果在expires未到期之前修改了缓存的文件，那么用户访问时是无法得到最新资源的，如果想要用户得到最新资源，那么必须修改资源的名称

### 最佳实践4：压缩组件

Web客户端可以通过HTTP请求中的Accept-Encoding头来标识对压缩的支持，如Accept-Encoding: gzip, deflate，如果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，并通过响应中的Content-Encoding来通知客户端

Gzip是目前最流行和最有效的压缩方法

推荐对HTML、JS和CSS以及其他文本响应进行压缩，而不要对图片和PDF进行压缩，因为它们本来就已经被压缩了，再次压缩只是浪费CPU资源，甚至还有可能增加文件的大小

压缩的成本主要是CPU成本，服务器要压缩，客户端要解压缩，所以是否需要压缩通常是一个比较复杂的权衡过程，按照经验通常对大于1KB或2KB的文件进行压缩

代理缓存，如果资源请求来自于代理服务器，那么只提供压缩或者未压缩的文件是危险的，必须要同时提供两种，但需要在响应中添加Vary头，这样真正的用户可以从代理服务器根据自己的需要取用响应版本

Vary: Accept-Encoding

在某些边缘情况下会出现为不能解压缩的客户端提供压缩资源的情况，为了避免这种情况的出现，可以只为已经证实可以解压缩的浏览器提供压缩版本，这被称为浏览器白名单方式。

最佳实践5：将样式表放在顶部

将样式表放在文档的底部时，浏览器在样式表加载完成之前会拒绝显示任何可视化组件，这称为“白屏”，这是一种非常不好的用户体验，因为用户无法从白屏中获得任何有效信息，他无法判断自己的浏览器是不是崩溃了。这个最佳实践，对于加载页面所需要的实际时间没有太多影响，它影响的是浏览器对这些组件顺序的反应。

使用@import引入css文件时，@import规则必须放在其他所有规则之前