# 前端优化方案

优化的目的：

用户--页面加载更快

服务商角度--减少页面请求数或者减少请求所占的带宽能够节省可观的资源

一页面级别的优化：

1.减少HTTP请求数（1）设计实现层面简化页面（2）合理设置http缓存比如：原则很简单，能缓存越多越好，能缓存越久越好。例如，很少变化的图片资源可以直接通过 HTTP Header中的Expires设置一个很长的过期头 ;变化不频繁而又可能会变的资源可以使用 Last-Modifed来做请求验证。尽可能的让资源能够在缓存中待得更久（3）资源合并与压缩（4）css sprites

2.外部脚本置底：让脚本内容在页面信息内容加载后再加载

3.css放在head中：放在body中时有可能还没开始解析CSS就开始渲染页面了，用户体验差

4.减少不必要的http跳转

5.避免重复的资源请求。

二代码级优化

1.减少DOM操作，避免回流和重绘，减少作用域链查找（如果在循环中需要访问非本作用域下的变量时请在遍历之前用局部变量缓存该变量，并在遍历结束后再重写那个变量，这一点对全局变量尤其重要，因为全局变量处于作用域链的最顶端，访问时的查找次数是最多的。）

2.css选择器

3.图片压缩

# 如何提高页面加载速度

1.减少http请求（图片地图 css雪碧图，特别是图片比较多的网站如果能用css sprites降低图片数量带来的将是速度的提升）

2.使用CDN

3.添加expires头（对于很少变化的图片资源）

4.压缩组件

5.将样式表放在头部，希望页面尽快显示内容，为用户提供可视化的反馈

6.将脚本放在底部。因为js的下载和执行会阻塞DOM树的构建（严谨说是中断了dom树的更新）

7.使用外部的js和CSS，当脚本或样式是从外部引入的文件，浏览器有可能缓存它们，从而以后加载的时候能够直接使用缓存；

8.减少DNS查找

9.精简js和css，删除重复脚本

10.避免重定向

重定向：用于用户从一个URL重新路由到另一个URL

常用重定向的类型：

301：永久重定向（主要用于网站的域名发生变更后）

302：临时重定向（主要实现post请求后告知浏览器转移到新的URL）

304：not modified（主要用于当浏览器在其缓存中保留了组件的一个副本，同时组件已经过期了，浏览器就会生成一个get请求，如果服务器的组件没有被修改过就会返回304状态码，同时不携带主体，告知浏览器可以重用这个副本）

损伤性能：页面重定向后，延迟整个HTML文档的传输，在HTML文档到达之前页面不会显示任何东西，也没有任何组件被下载；

重定向的应用场景：跟踪内部流量：当拥有一个门户主页时，同时想对用户离开主页后的流量进行跟踪，这时可以使用重定向；跟踪出站流量（?）

11.在进行ajax请求的时候选择尽量使用get方法，这样可以使用客户端的缓存，提高请求速度。（因为post的请求是不可以在客户端缓存的，每次请求都需要发送给服务器进行处理）