性能黄金法则：

只有10%－20%的最终用户响应时间花在了下载HTML文档上（从Web服务器获取HTML文档，并传送到浏览器中）。**其余的80%－90%时间花在了下载HTML页面中的所有组件上。**

观察到：

有缓存的场景没有太多的下载活动，紧跟HTML文档的HTTP请求之后是一段空白，这段时间内，浏览器正在解析HTML、CSS、Javascript，并从缓存中获取组件。

大量的HTTP请求并行发生。

浏览器在下载脚本时会阻塞额外的HTTP请求。

如果浏览器在其缓存中保留了组件的一个副本，但并不确定它是否仍然有效，就会生成一个条件GET请求。如果确认缓存的副本仍然有效，浏览器就可以使用缓存中的副本，这会得到更小的响应和更快的用户体验。（先看本地时间，再发送GET）

1、减少HTTP请求：（限制不必要的HTTP请求）

改善响应时间的最简单途径就是减少组件的数量，并由此减少HTTP请求的数量。介绍一些技术，可以减少HTTP请求，又能避免在性能和设计时间进行艰难的抉择。

包括：本地图片、CSS Sprites、内联图片和脚本、样式表的合并。（减少50%）

本地图片：

在一个图片上关联多个URL。目标URL的选择取决于用户点击了图片上的哪个位置。

服务器端图片地图：

将所有点击提交到同一个目标RUL，向其传递用户点击的X、Y坐标

客户端图片地图：

可以用HTML的MAP标签实现。

CSS Sprits：

将多个图片合并到一个图片中。使用background-position -xx -xx来定

位

不仅减少了HTTP请求，还降低了下载量（因为合并后比原来小）

比图片地图灵活。图片地图中的图片必须是连续的，CSS Sprites不要求。

内联图片：

在src或者URL中使用“data:xxx”的形势，可以在页面中包含图片但无需任何额外的HTTP请求。IE不支持，且存在大小的限制。

可使用CSS设置内联图片，并将CSS规则放在外部样式表中。

合并脚本和样式表：

一般来说，使用外部脚本和样式表对性能更有利。将这些单独的文件合并到一个文件中，可以减少HTTP请求的数量并缩短最终用户响应时间。

解决的方法是：遵守编译型语言的模式，保持javascript的模块化，而在生成过程中从一组特定的模块生成一个目标文件。

确保组合的数量是可管理的。

2、使用内容发布网络（拉近HTTP请求）

在多个地理位置不同的服务器上部署内容

如果应用程序Web服务器离用户更近，则一个HTTP请求的响应时间将缩短；如果组件Web服务器离用户更近，则多个HTTP请求的响应时间将缩短。

内容发布网络（CDN）是一组分布在不同地理位置的Web服务器，用于更加有效的向用户发布内容。

CDN服务提供商。有些需要最终用户使用一个代理来配置他们的浏览器，有的需要开发者使用不同的域名修改他们的组件的URL。

优点：

1）缩短响应时间

2）有助于缓和Web流量峰值压力

3）备份、扩展存储能力和进行缓存

缺点：

1）响应时间可能会收到其他网站——甚至可能是竞争对手流量的影响。

2）无法直接控制组件服务器所带来的特殊麻烦。

3）如果CDN服务器的性能下降，你的工作质量耶随之下降。

CDN用户发布静态内容：图片、脚本、样式表和flash。静态文件更容易存储并具有较少的依赖。

3、添加Expires头：（限制不必要的HTTP请求）

Expires头用来告诉Web客户端它可以使用一个组件的当前副本，直到指定的时间为止。在HTTP响应头中。Expires头使用一个特定的时间，要求服务器和客户端的时钟严格同步。

Cache－Control使用max－age指令指定组件被缓存多久，以秒为单位定义了一个更新窗。只有HTTP1.1支持。

可以同时指定上面二者，同时指定的时候，如果都支持的话，后者会覆盖前者。

mod_expires Apache模块可以在使用Expires时能像max－age那样设置相对时间，通过Expires－Default完成。（支持HTTP1.0）

空缓存（缓存中没有当前页面的组件）或者完整缓存页面浏览的数量取决于Web应用程序的本质。

HTML文档不应该使用长久的Expires头，因为它包含动态内容

修订文件名：

为了确保用户能获取组件的最新版本，需要在所有HTML页面中修改组件的文件名

1、为所有的组件的文件名使用变量

2、将版本号嵌入到组件的文件名当中

4、压缩组件：（减小HTTP请求的大小）

Web客户端：请求头中有 Accept-Encoding 表示对压缩的支持

服务器：响应头中 Content-Encoding:gzip 通知浏览器采用了什么压缩

gzip是目前最流行最有效的压缩方法（deflate效率低且有些浏览器不支持）

压缩的队形：

1、脚本、样式表

2、HTML文档

3、XML和JSON在内的任何响应

不应该压缩图片和PDF文件（本来就压缩过，浪费CPU）

压缩的开销：

1、服务器CPU

2、客户端解压缩

一般是大于1K或者2K才进行压缩

当有代理服务器的时候，在服务器的响应头中加 Vary:Accept-Encoding，

代理服务器会根据情况缓存两份，分别为压缩和不压缩的情况

Vary:Accept-Encoding,User-Agent 添加白名单

Vary: *或者Cache-Control:private 禁用代理

5、将样式表放在顶部

实践发现，将DHTML特征的样式表放在文档顶部（head中），使页面加载得更快。将样式表放在文档底部会导致在浏览器中阻止内容逐步呈现（理想的方式）。在浏览器和用户等待位于底部的样式表时，浏览器会延迟显示任何可视化组件，就是白屏。

白屏：页面会完全空白，直到页面所有的内容同时涌上屏幕。

IE中，白屏的情形：在新窗口中打开、重新加载、作为主页

将CSS放在顶部：

link标签 href=

@import url()

注意：放在style中，可以加载多个，但必须在任何其他规则之前。使用@import会导致组件下载时的无序性，所以仍然可能会导致白屏。

将样式表放在页面底部也不一定会白屏，而是样式闪烁。有些浏览器在样式表未下载完毕之前会逐步显示页面，当样式表下载完毕，已经呈现的内容又要进行样式重绘。这就是样式闪烁。

将样式表放在顶部，可以同时避免白屏和样式重绘。

6、将脚本放在底部

脚本会阻塞并行下载。也会阻塞逐步呈现。

HTTP请求并行地执行，但是HTTP1.1规范建议浏览器从每个主机名并行地下载两个组件。不会同时下载所有。（http1.0，firefox可以并行下载8个）。可以简单地使用CNAME将组件分别放到多个主机名中，来增加并行下载量。

过多的并行下载会降低性能（CPU、带宽）

将脚本放在底部：

不会阻止页面内容的呈现

页面中的可视组件可以尽早下载

很多情况下无法将脚本移到底部，例如有document.write：

可以使用defer属性，表明脚本不包含document.write。firefox不支持。

7、避免CSS表达式(css中嵌入js代码)

CSS表达式: expression(document.body.clientWidth < 600? "600px" : "auto")

对CSS表达式的频繁求值使其得以工作，但也导致CSS表达式的性能降低。

避开该问题：

一次性表达式：在第一次表达式之后就重写这个样式，移除表达式。

事件处理器：使用事件处理器代替CSS表达式

8、使用外部JavaScript和CSS

纯粹而言，内联更快一些（http少）。

但是外部文件可以被浏览器缓存，HTML文件通常不会缓存。

基准：

如果浏览量很低，内联更有意义

如果不大可能产生完整缓存，内联是更好的选择

使用外部文件可以提高这些组件的重用率。将页面分成几种类型，为每种类型创建单独的脚本和样式表。如果重用度很低，还是内联更有意义。

主页：内联更好。

加载后下载：

对于那些作为多次页面浏览量中的第一次的主页。可以在主页中内联，同时又有外部组件，有两份。在主页加载完成后动态下载外部组件。必须双重定义，将组件放到看不见的iframe中。

动态内联：

先通过检查cookie，看浏览器缓存中是否有组件的缓存，如果没有就内联。

9、减少DNS查找：

通常浏览器查找一个给定主机名的IP地址要话费20－120毫秒。

DNS缓存可以被缓存起来以提高性能。浏览器在其缓存中保留DNS记录。

查找返回的DNS中包含了一个存活时间TTL值，用于告诉客户端可以对该记录缓存多久（一般几分钟到一小时）。Keep-Alive可以覆盖TTL。

减少唯一主机名可以减少DNS查找，但是主机名减少，会减少并行下载从而增加响应时间。建议将组件放在2－4台主机。

10、精简JavaScript

精简：

从代码中移除不必要的字符以减小其大小。所有的字符以及不必要的空白字符都将被移除。

混淆：

也会移除注释会空白，同时也会改写代码，例如函数和变量的名字将被转换为更短的字符串。

混淆更加复杂，混淆过程可能会引入错误。

要对任何不能改变的符号进行标记，防止混淆修改。

难阅读，难调试

节省：

精简JavaScript代码的最流行的工具是JSMin。

在结合使用gzip压缩之后，混淆和精简的差距将减小。精简不会带来混淆的风险。

gzip压缩比精简更能减小文件大小。

精简CSS的节省比JavaScript小。（0－0xp，#fff-#fffff）

11、避免重定向

重定向：

状态码：301（永久），302（临时），响应头中有location表示目标

<meta http-equiv="refresh" content="0;url=http...">

document.location=url

重定向会延迟整个HTML文档的传输。

重定向的情况：

却少结尾的斜线。当一个URL的结尾必须出现斜线而没有出现时

网站后端被重写，新的URL可能不一样。

跟踪内部流量。（可以用Referer代替）

跟踪出站流量

美化URL

12、删除重复脚本

导致脚本重复：团队大小、脚本数量

重复脚本损伤性能：

不必要的http请求

执行JavaScript所浪费的时间

13、配置ETag

Web服务器和浏览器用来确认缓存组件有效性的一种机制

如果Expires过期了，就要发送HTTP请求，条件GET请求。

两种方式：

比较最新修改日期：浏览器get－服务器返回Last-Modified－浏览器get发送If-Modefied-Since－服务器与最新修改日期进行比较决定发送304或者组件。

实体标签：浏览器get－服务器返回ETag－浏览器get发送If-None_Match－服务器进行比较决定发送304或者组件。ETag：唯一标识了一个组件的一个特定版本的字符串。

ETag的问题：

Etag使用组件的某些属性来构造，这些属性对于特定的、寄宿了网站的服务器来说是唯一的。对于拥有多台服务器的网站来说，即使相同的组件，ETag也是不一样的。导致性能下降。Apache和IIS向ETag嵌入的数据都会大大降低有效性验证的成功率。

降低了代理缓存的效率。因为代理后面用户缓存的ETag和代理缓存的ETag不匹配。

安全性弱点

同时设置If-None-Match和If-Modefied-Since，会禁止返回304

建议修改ETag的内容。

14、使Ajax可缓存

Web2.0

DHTML：通过CSS、JavaScript、DOM使得HTML页面在加载完毕后能够变化。Ajax是DHTML中使用的一项技术。

Ajax：异步的JavaScript和XML。Ajax层位于客户端，与Web服务器进行交互以获取请求的信息，并与表现层交互，仅更新那些必要的组件。

Ajax是异步的，但并不一定是即时的

被动请求：为了将来使用而预先发起，即时。

主动请求：可能需要等待。

优化Ajax：

抓包工具 IMB Page Detailer

改善主动Ajax请求最重要的方式就是使响应可缓存。第三条，Expires。

使Ajax可请求：

改变HTTP头

响应的个性化和动态本质：使用查询字符串参数

数据隐私而不能缓存，解决方案：采用SSL，SSL响应可缓存。确保数据隐私的同时在当前会话中缓存响应以改善用户体验。