**网站性能优化**

**服务器端优化**

1. **不经常改动的图片、文档进行缓存，不必每次访问都请求服务器**
2. **压缩内容，减小网络流量压力**
3. **设置Etags，客户端请求某文件时，服务端优先检查客户端存不存在该文件，如果不存在就从服务器端下载该文件并返回200，如果存在并且在缓存期内且无更改，则服务器值返回304，而不会返回资源内容，使用Etags会增加服务器端的负载，使用时自行平衡**

**Cookie优化**

1. **减小Cookie的体积，设置合理的过期时间**
2. **分域名存储Cookie，减少不使用Cookie时的Cookie传递**

**CSS处理**

1. **CSS放到代码页上端、避免CSS表达式、从HTML中剥离JS和CSS**
2. **精简CSS和JS**
3. **尽量使用CSS代替JS，js会降低性能，**

**图片处理：**

1. **尽量使用PNG而不是gif，因为PNG功能更多体积更小**
2. **不在HTML中缩放图片，尽量使用合成好的图片**

**内容优化**

1. **减少DNS查找**
2. **尽量减少重定向**
3. **切分多个组件到多个域**
4. **尽量减少404页面出现的情况**

[**大型网站--前端性能优化和规范**](http://www.cnblogs.com/and/p/3390676.html)

  Web性能涉及的范围太广，但一般web开发者在程序上线以后很多都曾遇到过性能的问题。普遍表现为页面速度开始急剧变慢，正常访问时间变的很长，或则干脆给你抛出异常错误页面。这里会涉及到很多可能发生的情况，举例几个最主要发生的情况：

 \* 数据库连接超过最大限制，一般表现为程序的连接池满，拒绝了与数据库的连接。

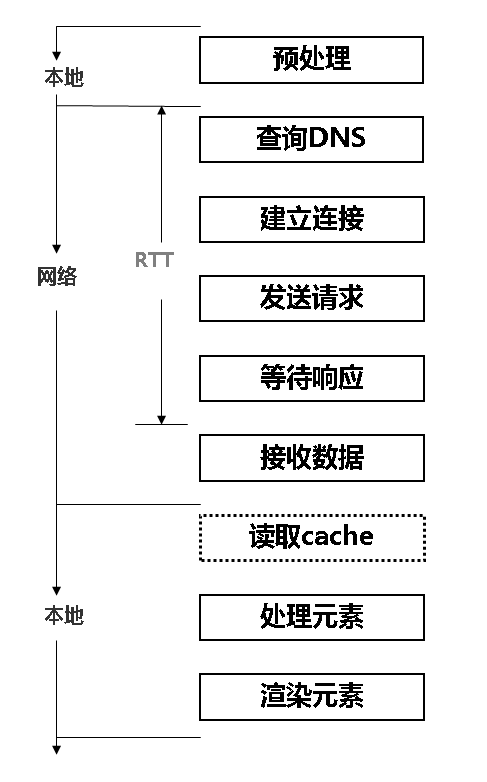
 \* 数据库死锁

 \* Web Server 超过最大连接数（一般在虚拟主机上才会限制）

 \* 内存泄漏

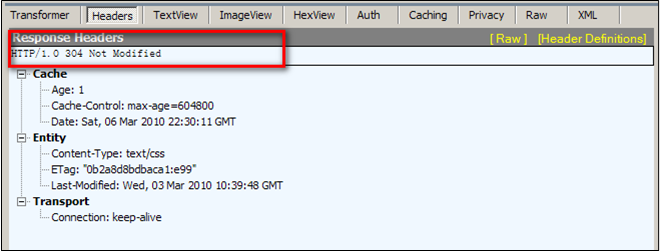
 \* Http连接数太多，即访问量超过了机器和软件设计正常所能提供的服务

**浏览器请求和响应的过程**

**

**第一步、浏览器预处理**

查询Cache：读取Cache 或者发送304请求



**第二步、查询DNS**

**优化规则--减少DNS查找**

DNS缓存

浏览器DNS缓存 计算机DNS缓存 服务器DNS缓存（TTL）

使用Keep-Alive特性   
减少DNS查找

当客户端的DNS缓存为空时，DNS查找的数量与Web页面中唯一主机名的数量相等。减少唯一主机名的数量就可以减少DNS查找的数量。

较少的域名来减少DNS查找（2-4个主机）

**第三步、建立连接**

**优化规则-- 使用内容分发网络**

美国十大Internet网站和CDN服务提供商

页面静态化,取决于发布系统

Ctrip使用的China-Cache和网宿

**优化规则--用域名划分页面内容**

按页面内容划分域名，在合适的资源服务器上存放文件

**第四步、发送请求**

**优化规则-- 减少HTTP请求**

 HTTP请求30-40,合并文件,图片地图,内联图像

a)js文件（不超过7个）

1.tuna\_090501\_base.js和tuna\_090501\_module.js(拆分tuna\_090501.js)

2.数据文件js（1-2个）

3.频道公用js(1个）和页面私有js（1-2个）

不含ga.js、uiscript.asp和外链其他网站的js

b) css文件不超过4个，各频道首页和全站首页不超过3个。

c) 目前无法解决的是allyes广告的请求数。

•

• 大量的广告和产品图片可能会造成，图片请求数很大，可能造成总请求数指标吃紧，

    这个只能从设计上搞定，需要权衡

• 目前老页面可能css和js文件请求数可能会超标

**优化规则- – 优化CSS Spirite**

  图片地图   Ctrip首页例子

**优化规则– 避免404错误**

避免内部无效的链接

**规则优化 –不要使用frameset，少使用iframe**

搜索引擎不友好、

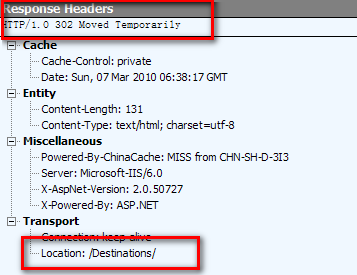
即时内容为空，加载也需要时间、会阻止页面加载

禁止使用iframe引入外部资源，不包括allyes广告，不包括about:blank的空页面。

**第五步、等待响应**

**优化规则 --避免重定向**

在重定向完毕并且HTML下载完毕之前，是没有任何东西显示给用户的



涉及服务器负载、数据查询、服务器端缓存等

**第七步、接收数据**

**优化规则 -- 压缩组件**

HTML文档、脚本和样式表、XML和JSON的文本响应 压缩如何工作  
压缩通常能将响应的数据量减少将近70%

**优化规则 -- 精简Javascript和Css**

从代码中移除不必要的字符以减少其大小，减少加载时间。

**规则规则– 尽量缩减页面大小**

页面必须小于150K(不含图片）  
a) 静态文件是否gzip  
b) 图片是否压缩优化过

**第八步、读取Cache**

**优化规则-- 添加Expire或Cache-Control**

应用于不经常变化的组件，包括脚本、样式表、Flash组件、图片  
Expires和Cache-Control

**规则规则 -- 使用外部的Js和Css文件**

尽可能使用外部Js和Css，因为我们目前大部分Js和Css都做了Gzip和缓存技术，可以充分利用。

**第九步、处理元素**

不要对image和pdf等二进制文件进行gzip压缩

**第十步、渲染元素**

**优化规则 -- 将样式表放在顶部**

 界面原型页面必须将样式表置于页面顶部，开发人员如无特殊原因也必须将样式表置于顶部。

以往多数是因为masterpage原因无法将所有样式表置顶，在改版修改masterpage时，尽可能按照此原则进行设计。

**优化规则 – 建议将脚本放在底部**

 一般浏览器可以允许并行下载，取决于主机个数、带宽等

（默认情况下，IE是2个而FF是8个）

下载脚本时并行下载实际上是被禁用的。

**优化规则-- 移除重复脚本**

 必须为0

**优化规则 -- 避免CSS表达式**

影响浏览器渲染时间

**优化规则 – 优化图像**

尽量使用GIF和PNG

尽量使用png/gif格式的图片，png的图片优先，但是必须注意如要兼容IE6，则png使用一定要注意透明问题。

图片在上次前一定要先用工具压缩优化（png、jpg）

**Javascript开发规范**

大型的项目在前端 JS 方面有几个需要达成的目标：

1. 代码逻辑分层
2. 避免全局变量
3. 便于多人协作开发
4. 各部分代码模块化，可以按需加载
5. 保持全局变量的清洁
6. 可进行单元测试