# 前端性能优化

## 前端性能优化

1. 减少http请求

雪碧图

Background-position属性设置背景图像的起始位置。

1. 使用HTTP2
2. 使用服务器端渲染
3. 静态资源使用CDN

内容分发网络（CDN）是一组分布在多个不同地理位置的Web服务器。我们都知道，当服务器离用户越远时，延迟越高。CDN就是为了解决这一问题，在多个位置部署服务器，当用户离服务器更近，从而缩短请求时间。

1. 将CSS放在文件头部，JavaScript文件放在底部

CSS不会阻塞html生成DOM树，但会阻塞浏览器布局。

1. 使用字体图标iconfont替换图片图标
2. 善用缓存，不重复加载相同的资源
3. 压缩文件

JavaScript ： UglifyPlugin

css : MiniCssExtractPlugin

HTML : HtmlWebpackPlugin

Gzip : 可以通过向HTTP请求头中的Accept-Encoding头添加gzip标识来开启这一功能。当然，服务器也得支持这一功能。

1. 图片优化
2. 通过webpack按需加载代码，提取第三库代码，减少ES6转为ES5的冗余代码
3. 减少重绘重排
4. 避免在页面的主体布局中使用table，table要等其中的内容完全下载之后才会先睡出来，显示比div+css布局慢
5. 使用事件委托/事件代理
6. 注意程序的局部性
7. If-else对比switch
8. 查找表
9. 避免页面卡顿
10. 使用requestAnimationFrame来实现视觉变化
11. 使用Web Workers
12. 使用位操作
13. 不要覆盖原生方法
14. 降低css选择器的复杂性
15. 使用flexbox而不是最早的布局模型
16. 使用transform和opacity [əʊˈpæsəti] 属性更改来实现动画

在css中，transforms和opacity这两个属性更改不会触发重排和重绘，它们是可以有合成器单独处理的属性。

1. 合理使用规则，避免过度优化

FCP（首屏渲染时间）

RTT（HTTP的往返时间）

前后端分离导致首屏需要请求更多的内容，导致更多HTTP的往返时间，这造成了白屏。

## 优化首屏渲染时间，减少白屏时间的一些方案

* 加速或减少HTTP请求耗损

1. 使用CDN加载公用库
2. 使用强缓存和协商缓存
3. 使用域名收敛
4. 小图片使用Base64代替
5. 使用Get请求代替Post请求
6. 设置Access-Control-Max-Age减少预检请求，
7. 页面内跳转其他域名或请求使用其他域名的资源时使用浏览器prefetch预解析等

延迟加载

* 减少请求内容的体积

非重要的库、非首屏图片延迟加载、SPA的组件懒加载等；

* 浏览器渲染原理

1. 开启服务器Gzip压缩
2. JS、CSS文件压缩合并
3. 减少cookies大小
4. SSR直接输出渲染后的HTML等

* 优化用户等待体验

白屏使用加载进度条、菊花图、骨架屏代替等。

## Css和JS的位置会影响页面效率

Css是页面渲染的关键因素之一，当页面存在外链css时，浏览器会等待全部的css下载及解析完成后再渲染页面。关键路径上的任何延迟都会影响首屏时间，因此我们需要尽快的将css传输到用户的设备，否则，用户只能看到一个空白的屏幕

Css在加载过程中不会影响到DOM树的生成，但是会影响render树的生成，进而影响到布局，所以一般来说，style的link标签需要尽量放在head里面，因为在解析DOM树的时候是自上而下的，而css样式又是通过异步加载的，这样的话，解析DOM树下的body节点和加载css样式能尽可能的并行，加快render树的生成速度。

Js脚本应该放在底部，原因在于js线程和GUI渲染线程是互斥的关系，如果js放在首部，当下载执行js的时候，会影响渲染线程绘制页面。