# 性能优化篇

1. **为什么要强调CSS要放在header里，js放在尾部？**

**Css为什么放在头部：**

CSS不会阻塞DOM的构建，并且它们是可以并行的，因此尽快的下载CSS文件以便构建CSSOM

**JS为什么放在尾部：**

Html遇到JS标签（非async和defer）都会停下下载js资源文件，并且执行完成之后才继续构建DOM，即JS阻塞DOM

**JS要在CSS之后：**

因为JS允许访问样式，一旦JS之前有CSS文件没有加载完成，JS一定会等到CSS文件加载完成之后，才开始下载执行JS， 所有CSS阻塞JS

1. **白屏解决方案**

含义：

白屏时间指的是浏览器开始显示内容的时间，一般认为浏览器开始渲染body或者解析完head标签的时候就是页面白屏结束的时间

计算方法：

title后输出一个时间pagestartime; head结束前 输出一个时间firstpaint。 白屏时间=firstpaint-pagestarttime/performance.timing.navigationStart

优化：

1. 加快js的执行速度，比如无限滚动的页面，可以用js先渲染一个屏幕范围内的东西
2. 减少文件体积
3. 首屏同步渲染html,后续的滚屏再异步加载和渲染
4. **首屏加载解决方案**

含义：

首屏时间是指用户打开网站开始，到浏览器首屏内容渲染完成的时间

计算方式：

1. 模块标签标记。适用于内容不需要拉取数据才能生存以及页面不考虑图片等资源的加载情况。结束位 置加时间戳输出时间
2. 统计首屏内图片加载最慢事件。 通常图片加载最慢，所以会把首屏内加载事件最慢的图片时间
3. 自定义计算

优化：

1. 首屏数据拉取逻辑放在顶部（数据最快返回）
2. 首屏渲染css及js逻辑优先内联html，返回时能立即执行
3. 次屏逻辑延后执行，分包加载
4. **如何减少回流和重绘**
5. 使用 transform 替代 top
6. 使用 visibility 替换 display: none ，因为前者只会引起重绘，后者会引发回流（改变了布 局），也可以考虑使用opacity，原因和transform类似
7. 不使用 table 布局，可能很小的一个小改动会造成整个 table 的重新布局
8. CSS选择符从右往左匹配查找，避免节点层级过多
9. 将频繁重绘或者回流的节点设置为图层，图层能够阻止该节点的渲染行为影响别的节点。（willchange属性或者video,iframe标签等）
10. 节点属性不要放在一个循环里当循环变量
11. **为什么频繁操作DOM性能很差**
12. 因为 DOM 是属于GUI渲染引擎中的东西，而 JS 又是 JS 引擎中的东西。当我们通过 JS 操作 DOM 的时候， 其实这个操作涉及到了两个线程之间的通信，那么势必会带来一些性能上的损耗
13. DOM对象上有太多的浏览器上的实现，每次去构建或者访问DOM元素相对于原生对象非常消耗性能
14. **插入几万个 DOM，如何实现页面不卡顿？**
15. 通过 requestAnimationFrame 的方式去循环的插入 DOM
16. 通过虚拟滚动 这种技术的原理就是只渲染可视区域内的内容，非可见区域的那就完全不渲染了，当用户在滚动的 时候就实时去替换渲染的内容。 即使列表很长，但是渲染的 DOM 元素永远只有那么几个，当我们滚动页面的时候就会实时去更新 DOM
17. InnerHTML，createDocumentFragement、分时插入setTimeout
18. **可以通过哪些方式来最快的渲染页面?**

不考虑缓存和优化网络协议的前提下，

1. .从文件大小考虑
2. 从 script 标签使用上来考虑 async和defer
3. 从需要下载的内容是否需要在首屏使用上来考虑，分包
4. .最后就是从 CSS、HTML 的代码书写上来考虑了
5. **图片大小如何优化**
6. 图片压缩
7. 图片大小
8. 图片像素点颜色
9. **图片加载如何优化**
10. 用 CSS 去代替
11. 用 CDN 加载，计算出适配屏幕的宽度，然后去请求相应裁剪好的图片
12. 小图使用base64
13. 雪碧图
14. 选择正确的图片格式，小图SVG考虑，照片使用JPEG
15. **防抖节流**

**节流：**每N秒后允许执行一次函数，N秒内触发事件无效，稀释函数执行频率

**防抖：**在N秒之后执行函数，但是N秒内再次触发事件，则重新计算N秒

1. **预渲染**

通过预渲染将下载的文件预先在后台渲染，可以使用以下代码开启预渲染

<Link ref=”prerender” href=”地址”/>

**注意：** 预渲染虽然可以提高页面的加载速度，但是要确保该页面大概率会被用户在之后打开，否则就是白白浪 费资源去渲染。

1. **懒加载和懒执行**

**懒加载：**懒加载就是将不关键的资源延后加载。

懒加载的原理就是只加载自定义区域（通常是可视区域，但也可以是即将进入可视区域）内需要加载的 东西。 对于图片来说，先设置图片标签的 src 属性为一张占位图，将真实的图片资源放入一个自定义属性 中，当进入自定义区域时，就将自定义属性替换为 src 属性，这样图片就会去下载资源，实现了图片 懒加载。 懒加载不仅可以用于图片，也可以使用在别的资源上。比如进入可视区域才开始播放视频等等

1. **CDN**

CDN 的原理是尽可能的在各个地方分布机房缓存数据，这样即使我们的根服务器远在国外，在国内的 用户也可以通过国内的机房迅速加载资源。 因此，我们可以将静态资源尽量使用 CDN 加载，由于浏览器对于单个域名有并发请求上限，可以考虑 使用多个 CDN 域名。并且对于 CDN 加载静态资源需要注意CDN 域名要与主站不同，否则每次请求都 会带上主站的 Cookie，平白消耗流量