前端性能优化

1. 在js中尽量减少闭包的使用 (原因是:闭包会产生不释放的栈内存)

a: 循环给元素做事件绑定的时候,尽可能把后期需要的信息(eg 索引)存储到元素的自定义属性中,而不是创建闭包存储

b:可以在最外层形成一个闭包,把一些需要的公共信息进行存储,而不是每一个方法都创建闭包(单例模式)

c:尽可能手动释放不被占用的内存

1. 尽量合并css和js文件(把需要引入的css&js合并为一个) 原理:减少HTTP请求次数,,尽可能把合并后的代码进行压缩,减小HTTP请求资源的大小

A:webpack这种自动化构建工具,可以帮我们实现代码的压缩与合并(工程化开发)

B:在移动开发(或者追求高性能的pc端开发 百度首页)若css或者js不是需要很多 ,可以选择把css和js编程内嵌式(也就是代码直接写在html中)

1. 尽量使用字体图标或者SVG图标,来代替传统的png等格式的图片,放大不会变形,,而且渲染速度快,相对比位图小一些
2. 减少对DOM的操作 (主要是减少DOM的重绘和回流(重排))

a:关于重排的分离读写

b:使用文档碎片或者字符串拼接做的数据绑定(DOM的动态创建)

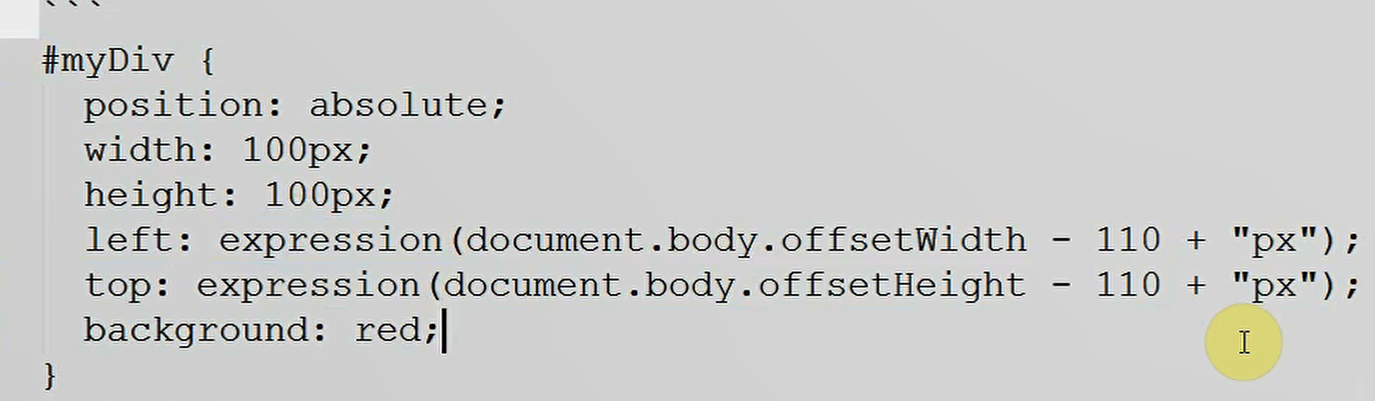
1. 在js中避免 “嵌套循环” (这种会额外增加循环次数)和 ‘死循环’
2. 采用图片的”懒加载” (延迟加载) 目的是为了减少页面第一次加载过程中HTTP的请求次数,让页面打开速度变快

步骤:开始加载页面的时候,所有的真实图片都不去发送http请求加载,而是给一张站位的背景图,当页面加载完,并且图片在可视区域内我们再去做图片加载

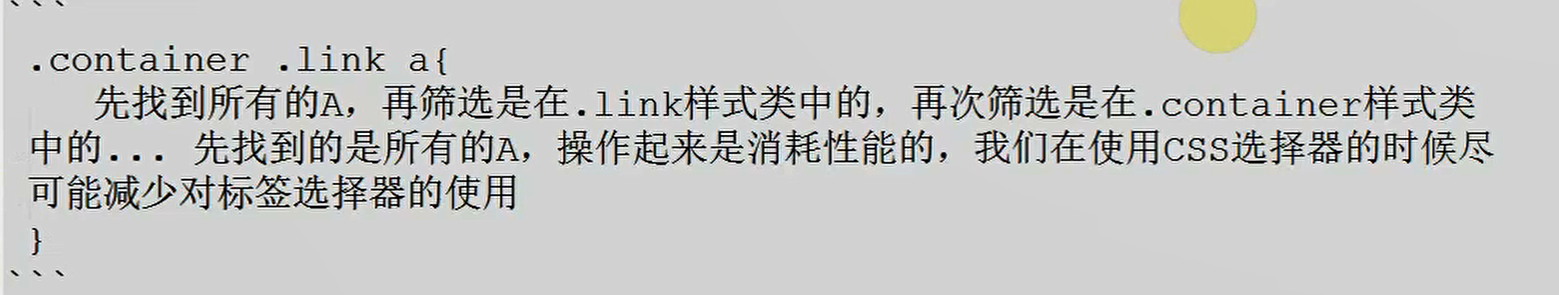
1. 利用浏览器和服务器的缓存技术(304缓存),把一些不经常更新的静态资源文件做缓存处理(eg js,css,静态图片等都可以做缓存)

原理**:**减少HTTP请求的次数及请求内容的大小

1. 尽可能使用事件委托(事件代理)来处理事件绑定的的操作,减少DOM的频繁操作,其中包括给每一个DOM元素做事件绑定
2. 尽可能减少css表达式的使用

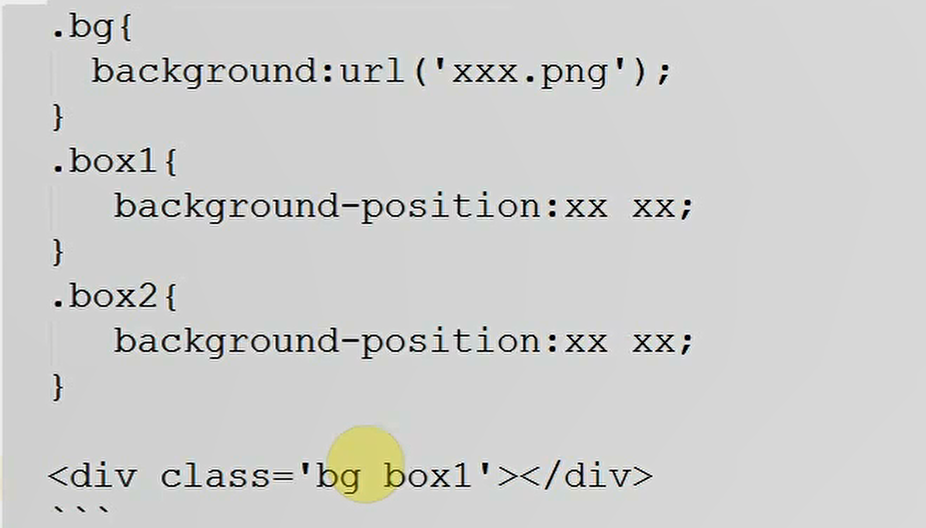


1. css解析规则是从右往左解析



1. 做CDN加速,重量级优化 烧钱机器
2. css雪碧图技术(css sprite /css 图片精灵)

把所有相对较小的资源图片汇总到一张大图上,后期我们只需要把大图加载下来.用背景定位的方式展示对应的小图即可



1. 避免重定向 减少对于cookie的使用(最主要是减少本地存储内容的大小),因为客户端操作cookie的时候,这些信息总是在服务端传来传去
2. 页面中各个的数据获取采用异步编程和延迟分批加载

使用异步获取数据,是为了降低HTTP通道的阻塞,不会因为数据没有 请求回来耽误下面信息的渲染,提高页面的打开速度,(需要动态绑定数据的区域先隐藏,,等数据返回并且绑定完成后在让其显示) 延迟分批加载类似与图片懒加载

1. 页面中出现音视频标签,我们不让页面加载的时候就去加载这些资源( 只需要设置preload=’none’ 即可)等待页面加载完成,音视频播放的时候我们再去加载音视频资源
2. 在客户端和服务器进行信息交互的时候 ,对于多项数据我们尽可能基于json格式来进行传送(jsom格式的数据处理方便,) 🡺相对于XML格式的传输才会有这个优势
3. 尽可能实现js的封装()低耦合高聚合,减少页面中的冗余代码(减少http请求资源的大小)
4. css设置定位后,,最好使用z-index改变盒子的层级,让盒子不在相同的平面上,这样后续处理的时候,性能有一点点的提高
5. 在基于ajax的get请求进行数据交互的时候,可以根据需求可以让其产生缓存(这个缓存不是304缓存),这样下一次从相同获取的数据是上一次缓存的数据(实际中很少用)
6. 尽量减少对于filter滤镜属性使用
7. 在css中导入的时候尽量减少使用@import导入式,因为这个是同步操作的,,只有对应的css导入,才会向下加载,而link是异步操作
8. 配置配置Etag
9. 使用window.requestAnimateFrame(js中的帧动画)代替传统中的定时器动画
10. 减少递归的使用,避免死递归,避免由于递归导致的栈内存嵌套(建议使用尾递归)
11. 避免使用iframe(不仅不好管控样式,而且相当于在A页面中加载了其他页面),消耗较大
12. 利用h5中提供的localstorage本地存储或者是manifest离线缓存,做一些信息的本地存储,下一次加载页面的时候直接从本地获取,减少HTTP请求次数
13. 基于script调取js的时候,可以使用defer或者async来异步加载

**额外技巧**

1. 我们一般把css放到body上,把js放到body下面 (需要等元素加载完再操作)
2. 能用css搞定的绝对不用js,能用原生js搞定的绝对不用插件,绝对不使用flash (除了音视频的低版本的浏览器播放)

css处理动画等功能的性能优于js,,而且css中的transform变形还开启了硬件加速

1. js中尽量减少对eveal的使用,因为js合并压缩的时候,可能出现符号不完善,,导致代码执行的优先级错乱问题.eveal处理起来的消耗更大一些
2. 使用keep-alive实现客户端和服务端的长连接
3. 尽量使用设计模式来管理我们的代码(单例,,构造,promise.发布订阅),方便后期的升级和维护
4. 开启服务器端的gzip压缩 (这个压缩可以有效减少请求资源的文件的大小 其实客户端的图片的资源也是可以进行压缩24位的会模糊)
5. 使用关键字,提高曝光率.标签语义化meta标签

**前端开发性能优化方案**

**j减少HTTP请求次数和请求大小**

**代码优化**

**->有利于SEO**

**->有利于扩展维护**

**->有利于减少性能消耗**

**[js代码优化的108条建议] [雅虎css优化的36条建议]**

**DNS以及HTTP通信方式的优化**