减少请求数量

【合并】

如果不进行文件合并，有如下3个隐患

1、文件与文件之间有插入的上行请求，增加了N-1个网络延迟

2、受丢包问题影响更严重

3、经过代理服务器时可能会被断开

但是，文件合并本身也有自己的问题

1、首屏渲染问题

2、缓存失效问题

所以，对于文件合并，有如下改进建议

1、公共库合并

2、不同页面单独合并

【图片处理】

1、雪碧图

CSS雪碧图是以前非常流行的技术，把网站上的一些图片整合到一张单独的图片中，可以减少网站的HTTP请求数量，但是当整合图片比较大时，一次加载比较慢。随着字体图片、SVG图片的流行，该技术渐渐退出了历史舞台

2、Base64

将图片的内容以Base64格式内嵌到HTML中，可以减少HTTP请求数量。但是，由于Base64编码用8位字符表示信息中的6个位，所以编码后大小大约比原始值扩大了 33%

3、使用字体图标来代替图片

【减少重定向】

尽量避免使用重定向，当页面发生了重定向，就会延迟整个HTML文档的传输。在HTML文档到达之前，页面中不会呈现任何东西，也没有任何组件会被下载，降低了用户体验

如果一定要使用重定向，如http重定向到https，要使用301永久重定向，而不是302临时重定向。因为，如果使用302，则每一次访问http，都会被重定向到https的页面。而永久重定向，在第一次从http重定向到https之后 ，每次访问http，会直接返回https的页面

【使用缓存】

使用cach-control或expires这类强缓存时，缓存不过期的情况下，不向服务器发送请求。强缓存过期时，会使用last-modified或etag这类协商缓存，向服务器发送请求，如果资源没有变化，则服务器返回304响应，浏览器继续从本地缓存加载资源；如果资源更新了，则服务器将更新后的资源发送到浏览器，并返回200响应

【不使用CSS @import】

CSS的@import会造成额外的请求

【避免使用空的src和href】

a标签设置空的href，会重定向到当前的页面地址

form设置空的method，会提交表单到当前的页面地址

减小资源大小

【压缩】

1、HTML压缩

HTML代码压缩就是压缩在文本文件中有意义，但是在HTML中不显示的字符，包括空格，制表符，换行符等

2、CSS压缩

CSS压缩包括无效代码删除与CSS语义合并

3、JS压缩与混乱

JS压缩与混乱包括无效字符及注释的删除、代码语义的缩减和优化、降低代码可读性，实现代码保护

4、图片压缩

针对真实图片情况，舍弃一些相对无关紧要的色彩信息

【webp】

在安卓下可以使用webp格式的图片，它具有更优的图像数据压缩算法，能带来更小的图片体积，同等画面质量下，体积比jpg、png少了25%以上，而且同时具备了无损和有损的压缩模式、Alpha 透明以及动画的特性

【开启gzip】

HTTP协议上的GZIP编码是一种用来改进WEB应用程序性能的技术。大流量的WEB站点常常使用GZIP压缩技术来让用户感受更快的速度。这一般是指WWW服务器中安装的一个功能，当有人来访问这个服务器中的网站时，服务器中的这个功能就将网页内容压缩后传输到来访的电脑浏览器中显示出来。一般对纯文本内容可压缩到原大小的40%

优化网络连接

【使用CDN】

CDN全称是Content Delivery Network，即内容分发网络，它能够实时地根据网络流量和各节点的连接、负载状况以及到用户的距离和响应时间等综合信息将用户的请求重新导向离用户最近的服务节点上。其目的是使用户可就近取得所需内容，解决 Internet网络拥挤的状况，提高用户访问网站的响应速度

【使用DNS预解析】

当浏览器访问一个域名的时候，需要解析一次DNS，获得对应域名的ip地址。在解析过程中，按照浏览器缓存、系统缓存、路由器缓存、ISP(运营商)DNS缓存、根域名服务器、顶级域名服务器、主域名服务器的顺序，逐步读取缓存，直到拿到IP地址

DNS Prefetch，即DNS预解析就是根据浏览器定义的规则，提前解析之后可能会用到的域名，使解析结果缓存到系统缓存中，缩短DNS解析时间，来提高网站的访问速度

方法是在 head 标签里面写上几个 link 标签

```
1 <link rel="dns-prefecth" href="
  https://www.google.com
  ">
2 <link rel="dns-prefecth" href="
  https://www.google-analytics.com
  ">
```

对以上几个网站提前解析 DNS，由于它是并行的，不会堵塞页面渲染，这样可以缩短资源加载的时间

【并行连接】

由于在HTTP1.1协议下，chrome每个域名的最大并发数是6个。使用多个域名，可以增加并发数

【持久连接】

使用keep-alive或presistent来建立持久连接，持久连接降低了时延和连接建立的开销，将连接保持在已调谐状态，而且减少了打开连接的潜在数量

【管道化连接】

在HTTP2协议中，可以开启管道化连接，即单条连接的多路复用，每条连接中并发传输多个资源，这里就不需要添加域名来增加并发数了

优化资源加载

【资源加载位置】

通过优化资源加载位置，更改资源加载时机，使尽可能快地展示出页面内容，尽可能快地使功能可用

1、CSS文件放在head中，先外链，后本页

2、JS文件放在body底部，先外链，后本页

3、处理页面、处理页面布局的JS文件放在head中，如babel-polyfill.js文件、flexible.js文件

4、body中间尽量不写style标签和script标签

【资源加载时机】

1、异步script标签

defer: 异步加载，在HTML解析完成后执行。defer的实际效果与将代码放在body底部类似

async: 异步加载，加载完成后立即执行

2、模块按需加载

在SPA等业务逻辑比较复杂的系统中，需要根据路由来加载当前页面需要的业务模块

按需加载，是一种很好的优化网页或应用的方式。这种方式实际上是先把代码在一些逻辑断点处分离开，然后在一些代码块中完成某些操作后，立即引用或即将引用另外一些新的代码块。这样加快了应用的初始加载速度，减轻了它的总体体积，因为某些代码块可能永远不会被加载

webpack 提供了两个类似的技术，优先选择的方式是使用符合 ECMAScript 提案 的 import() 语法。第二种则是使用 webpack 特定的 require.ensure

3、使用资源预加载preload和资源预读取prefetch

preload让浏览器提前加载指定资源，需要执行时再执行，可以加速本页面的加载速度

prefetch告诉浏览器加载下一页面可能会用到的资源，可以加速下一个页面的加载速度

4、资源懒加载与资源预加载

资源延迟加载也称为懒加载，延迟加载资源或符合某些条件时才加载某些资源

资源预加载是提前加载用户所需的资源，保证良好的用户体验

资源懒加载和资源预加载都是一种错峰操作，在浏览器忙碌的时候不做操作，浏览器空间时，再加载资源，优化了网络性能

减少重绘回流

【样式设置】

1、避免使用层级较深的选择器，或其他一些复杂的选择器，以提高CSS渲染效率

2、避免使用CSS表达式，CSS表达式是动态设置CSS属性的强大但危险方法，它的问题就在于计算频率很快。不仅仅是在页面显示和缩放时，就是在页面滚动、乃至移动鼠标时都会要重新计算一次

3、元素适当地定义高度或最小高度，否则元素的动态内容载入时，会出现页面元素的晃动或位置，造成回流

4、给图片设置尺寸。如果图片不设置尺寸，首次载入时，占据空间会从0到完全出现，上下左右都可能位移，发生回流

5、不要使用table布局，因为一个小改动可能会造成整个table重新布局。而且table渲染通常要3倍于同等元素时间

6、能够使用CSS实现的效果，尽量使用CSS而不使用JS实现

【渲染层】

1、此外，将需要多次重绘的元素独立为render layer渲染层，如设置absolute，可以减少重绘范围

2、对于一些进行动画的元素，使用硬件渲染，从而避免重绘和回流

【DOM优化】

1、缓存DOM

```
const div = document.getElementById('div')
```

由于查询DOM比较耗时，在同一个节点无需多次查询的情况下，可以缓存DOM

2、减少DOM深度及DOM数量

HTML 中标签元素越多，标签的层级越深，浏览器解析DOM并绘制到浏览器中所花的时间就越长，所以应尽可能保持 DOM 元素简洁和层级较少。

3、批量操作DOM

由于DOM操作比较耗时，且可能会造成回流，因此要避免频繁操作DOM，可以批量操作DOM，先用字符串拼接完毕，再用innerHTML更新DOM

4、批量操作CSS样式

通过切换class或者使用元素的style.csstext属性去批量操作元素样式

5、在内存中操作DOM

使用DocumentFragment对象，让DOM操作发生在内存中，而不是页面上

6、DOM元素离线更新

对DOM进行相关操作时，例、appendChild等都可以使用Document Fragment对象进行离线操作，带元素“组装”完成后再一次插入页面，或者使用display:none 对元素隐藏，在元素“消失”后进行相关操作

7、DOM读写分离

浏览器具有惰性渲染机制，连接多次修改DOM可能只触发浏览器的一次渲染。而如果修改DOM后，立即读取DOM。为了保证读取到正确的DOM值，会触发浏览器的一次渲染。因此，修改DOM的操作要与访问DOM分开进行

8、事件代理

事件代理是指将事件监听器注册在父级元素上，由于子元素的事件会通过事件冒泡的方式向上传播到父节点，因此，可以由父节点的监听函数统一处理多个子元素的事件

利用事件代理，可以减少内存使用，提高性能及降低代码复杂度

9、防抖和节流

使用函数节流（throttle）或函数去抖（debounce），限制某一个方法的频繁触发

10、及时清理环境

及时消除对象引用，清除定时器，清除事件监听器，创建最小作用域变量，可以及时回收内存

性能更好的API

1、用对选择器

选择器的性能排序如下所示，尽量选择性能更好的选择器

```
id选择器（#myid）
类选择器（.myclassname）
标签选择器（div,h1,p）
相邻选择器（h1+p）
子选择器（ul > li）
后代选择器（li a）
通配符选择器（*）
属性选择器（a[rel="external"]）
伪类选择器（a:hover,li:nth-child）
```

2、使用requestAnimationFrame来替代setTimeout和setInterval

希望在每一帧刚开始的时候对页面进行更改，目前只有使用 requestAnimationFrame 能够保证这一点。使用 setTimeout 或者 setInterval 来触发更新页面的函数，该函数可能在一帧的中间或者结束的时间点上调用，进而导致该帧后面需要进行的事情没有完成，引发丢帧

3、使用IntersectionObserver来实现图片可视区域的懒加载

传统的做法中，需要使用scroll事件，并调用getBoundingClientRect方法，来实现可视区域的判断，即使使用了函数节流，也会造成页面回流。使用IntersectionObserver，则没有上述问题

4、使用web worker

客户端javascript一个基本的特性是单线程：比如，浏览器无法同时运行两个事件处理程序，它也无法在一个事件处理程序运行的时候触发一个计时器。Web Worker是HTML5提供的一个javascript多线程解决方案，可以将一些大计算量的代码交由web Worker运行，从而避免阻塞用户界面，在执行复杂计算和数据处理时，这个API非常有用

但是，使用一些新的API的同时，也要注意其浏览器兼容性

webpack优化

【打包公共代码】

使用CommonsChunkPlugin插件，将公共模块拆出来，最终合成的文件能够在最开始的时候加载一次，便存到缓存中供后续使用。这会带来速度上的提升，因为浏览器会迅速将公共的代码从缓存中取出来，而不是每次访问一个新页面时，再去加载一个更大的文件

webpack 4 将移除 CommonsChunkPlugin, 取而代之的是两个新的配置项 optimization.splitChunks 和 optimization.runtimeChunk

通过设置 optimization.splitChunks.chunks: "all" 来启动默认的代码分割配置项

【动态导入和按需加载】

webpack提供了两种技术通过模块的内联函数调用来分离代码，优先选择的方式是，使用符合 ECMAScript 提案 的 import() 语法。第二种，则是使用 webpack 特定的 require.ensure

【剔除无用代码】

tree shaking 是一个术语，通常用于描述移除 JavaScript 上下文中的未引用代码(dead-code)。它依赖于 ES2015 模块系统中的静态结构特性，例如 import 和 export。这个术语和概念实际上是兴起于 ES2015 模块打包工具 rollup

JS的tree shaking主要通过uglifyjs插件来完成，CSS的tree shaking主要通过purify CSS来实现的

【长缓存优化】

1、将hash替换为chunkhash，这样当chunk不变时，缓存依然有效

2、使用Name而不是id

每个 module.id 会基于默认的解析顺序(resolve order)进行增量。也就是说，当解析顺序发生变化，ID 也会随之改变

下面来使用两个插件解决这个问题。第一个插件是 NamedModulesPlugin，将使用模块的路径，而不是数字标识符。虽然此插件有助于在开发过程中输出结果的可读性，然而执行时间会长一些。第二个选择是使用 HashedModuleIdsPlugin，推荐用于生产环境构建

【公用代码内联】

使用html-webpack-inline-chunk-plugin插件将mainfest.js内联到html文件中