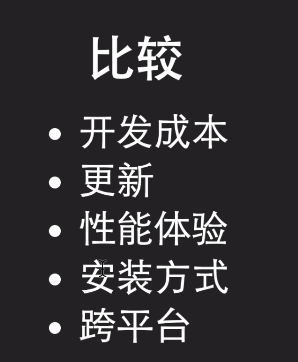
移动端的分类

一, 

二，

三，

四，

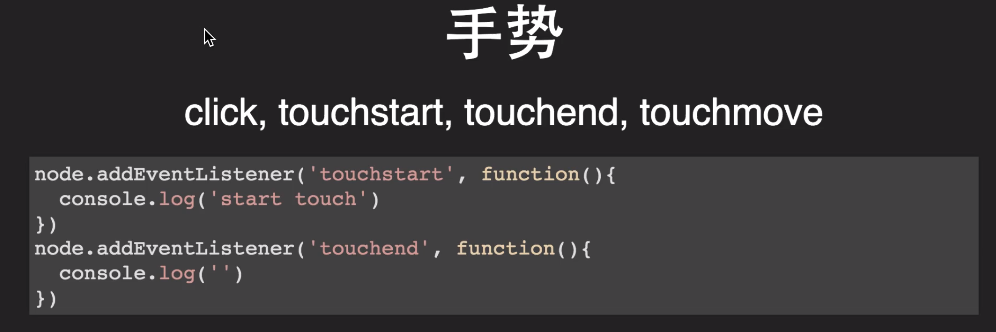


移动端和pc端的区别

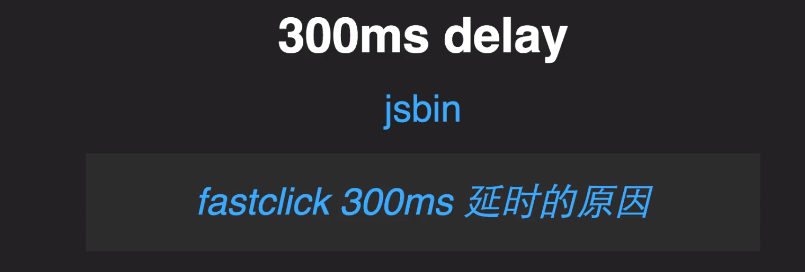




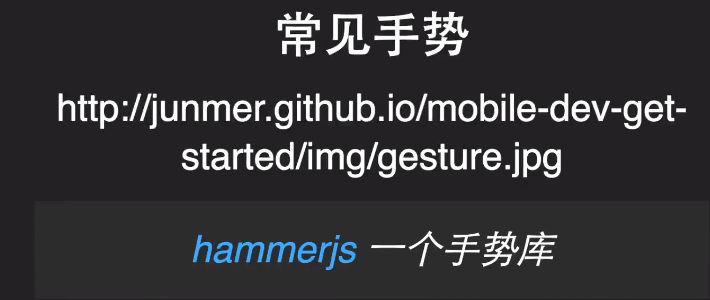




在pc端给某个元素绑定click点击事件，它会立即执行，在移动端给某个元素绑定click点击事件，它会延迟300ms后执行（这个延迟时浏览器做的）目的在手机端有个特性就是，单击放大和双击缩小，如果没有延迟，浏览器无法判断是单击还是双击。



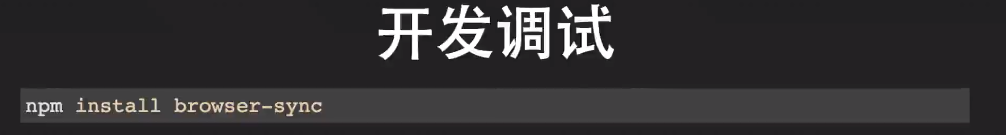
搜索文章300ms延迟的来龙去脉。





滚动 适配

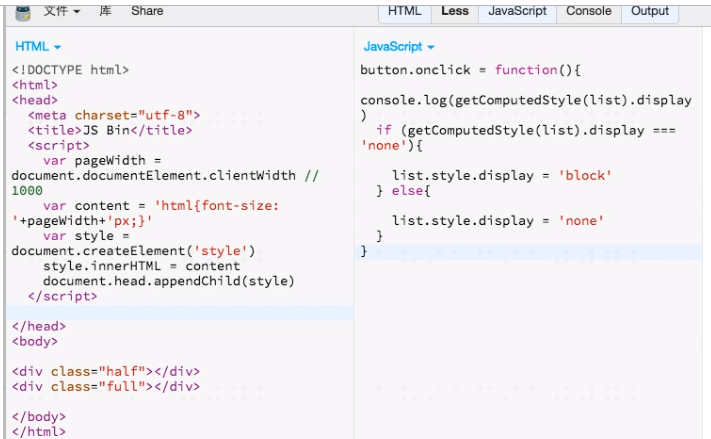




移动端适配之动态rem

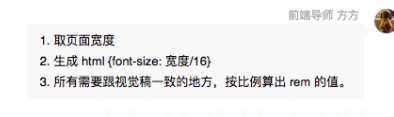






document.documentElement.clientWidth的宽度在pc上包含滚动条的宽度，在移动端不包含滚动条的宽度。所以pc上会出现滚动条，但是移动端不会。

Rem布局思路和步骤



## rem是什么？

    rem（font size of the root element）是指相对于根元素的字体大小的单位。简单的说它就是一个相对单位。看到rem大家一定会想起em单位，em（font size of the element）是指相对于父元素的字体大小的单位。它们之间其实很相似，只不过一个计算的规则是依赖根元素一个是依赖父元素计算。

谷歌搜索文章

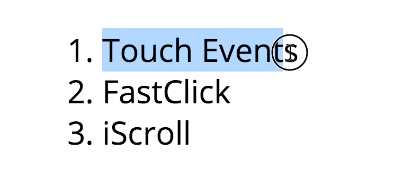


Bootstrap



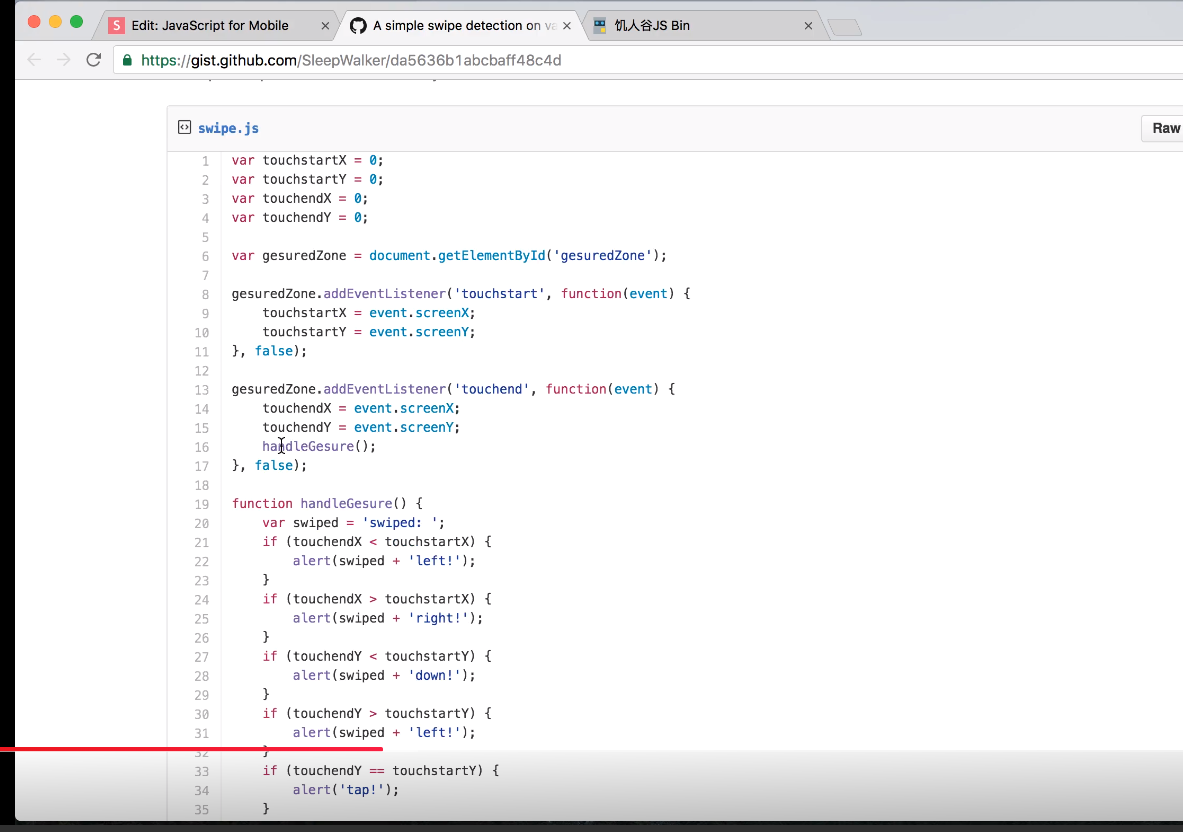
Bootstrap3手机优先Bootstrap2兼容IE8 PC优先



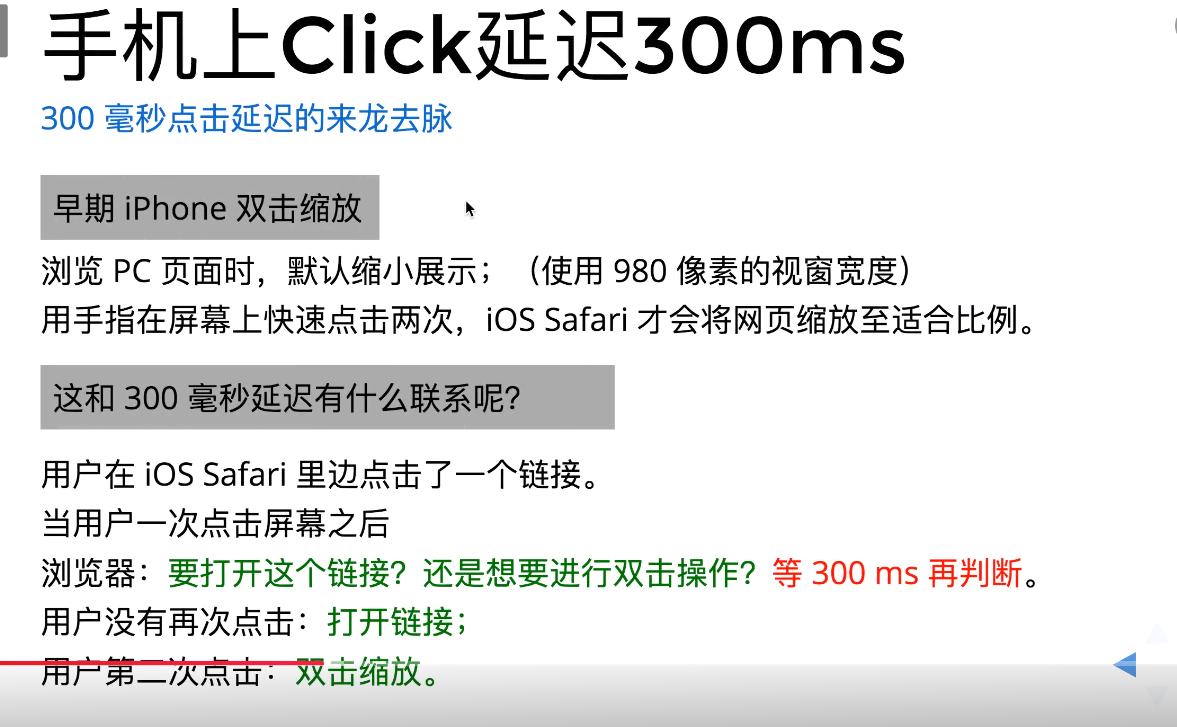




模擬js滑動事件

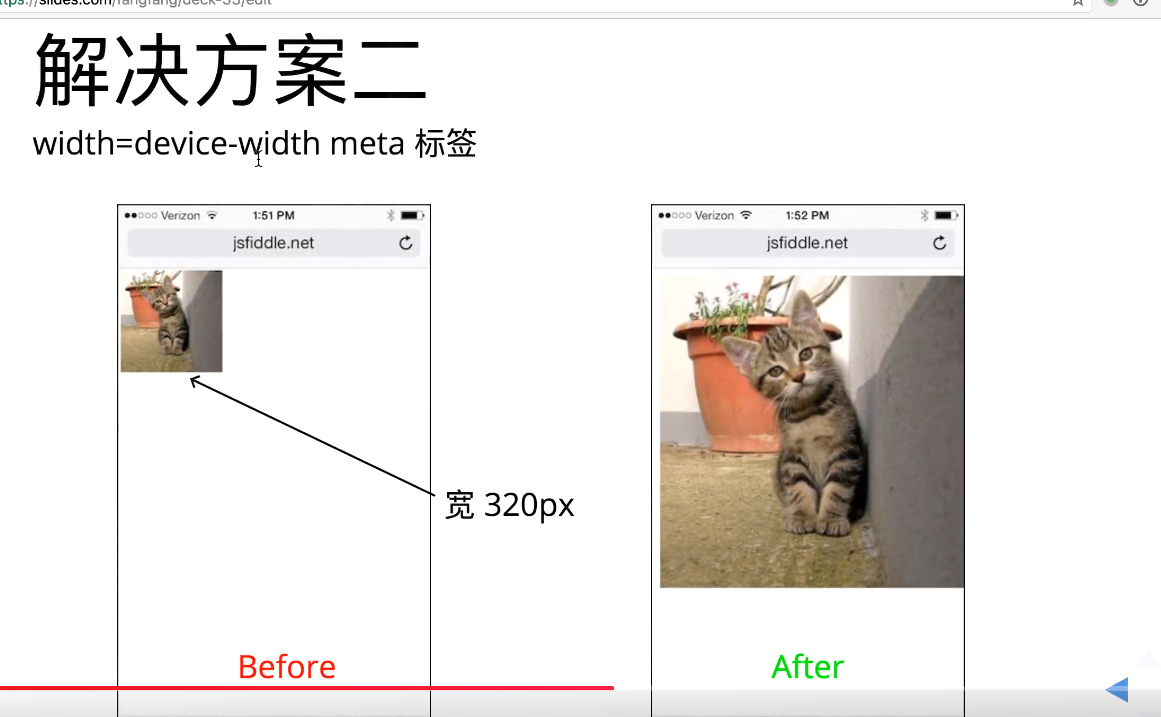




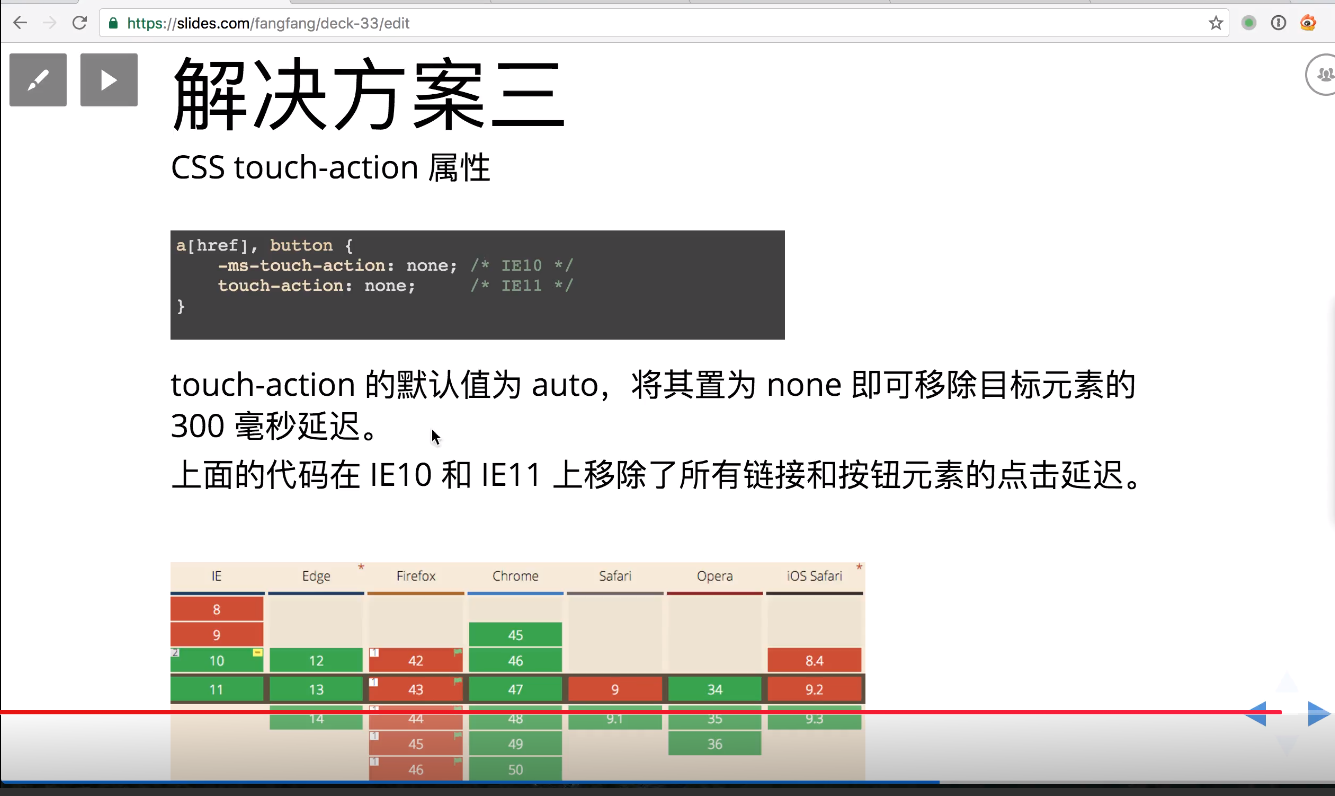


如何解決300ms延迟问题？









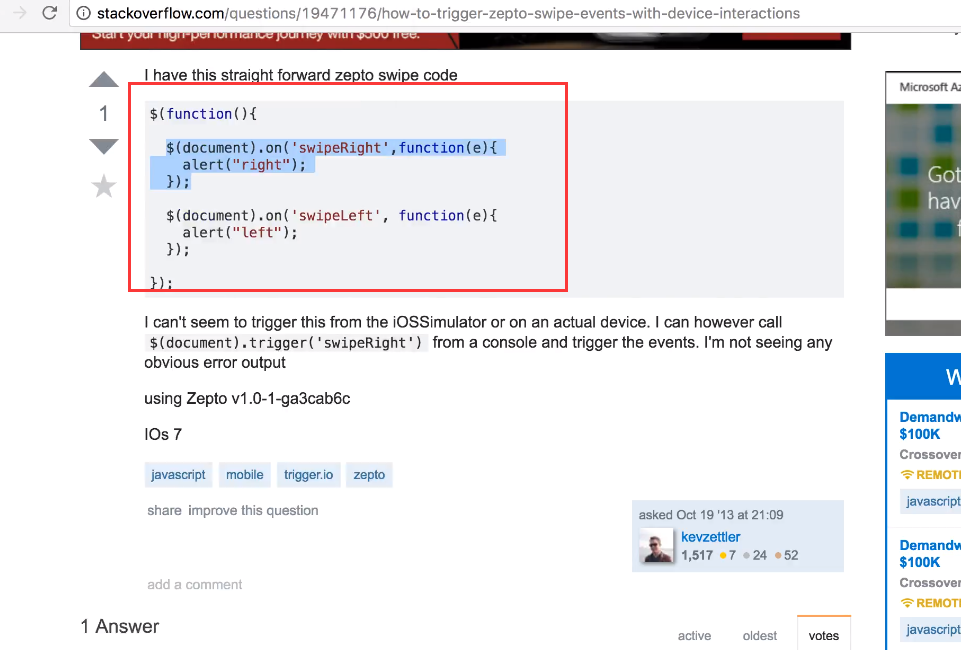


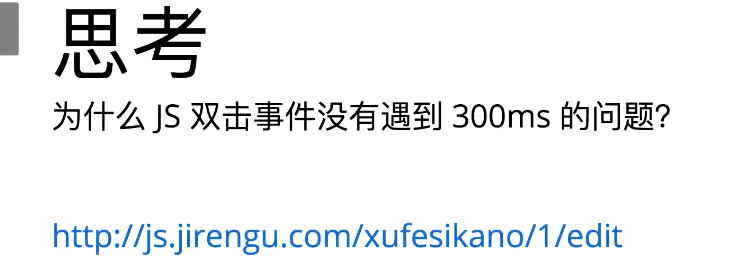




zepto插件 模拟滑动







## 响应式

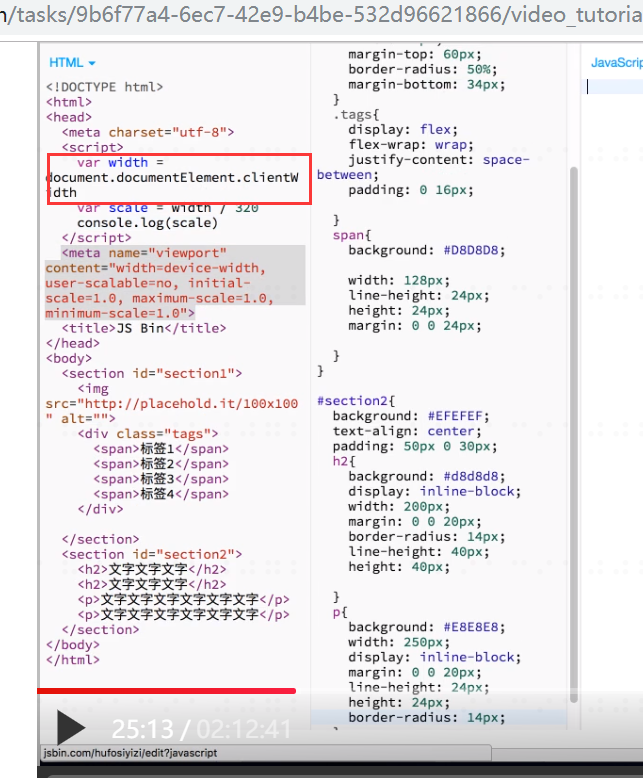


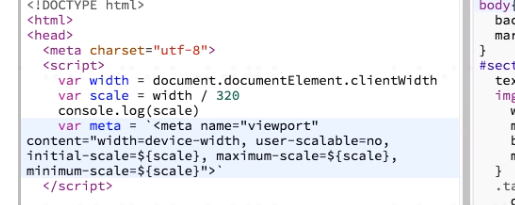
vw代表宽度的7.5%。

布局基本思路：

所有的元素尺寸全部应按照 设计稿的实际测量尺寸除以设备宽度的百分数来计算，这样就可以保证元素随着设备的不同等比例缩放从而占满整个屏幕

以320px为标准进行等比例缩放，计算缩放比例步骤：一，先获取设备屏幕的宽度，二，设备宽度除以定稿标准宽度得出缩放比例。三，利用js动态写meta标签。



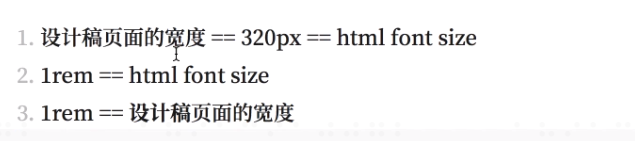


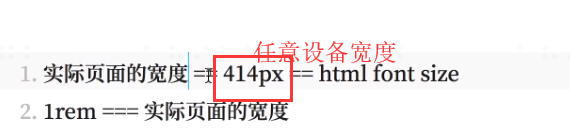
Meta标签里的它是获取设备屏幕的真实宽度，否则就会当成默认宽度来处理一般手机默认宽度为980px;

meta获取之后不能再次动态修改meta了。



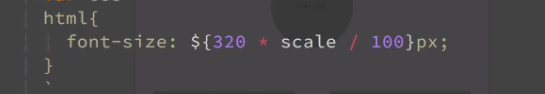
动态rem方案，先固定写死meta标签。

按图中设置后

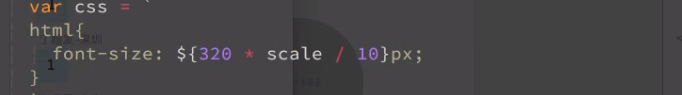


此方案特点保证1rem等于设备宽度

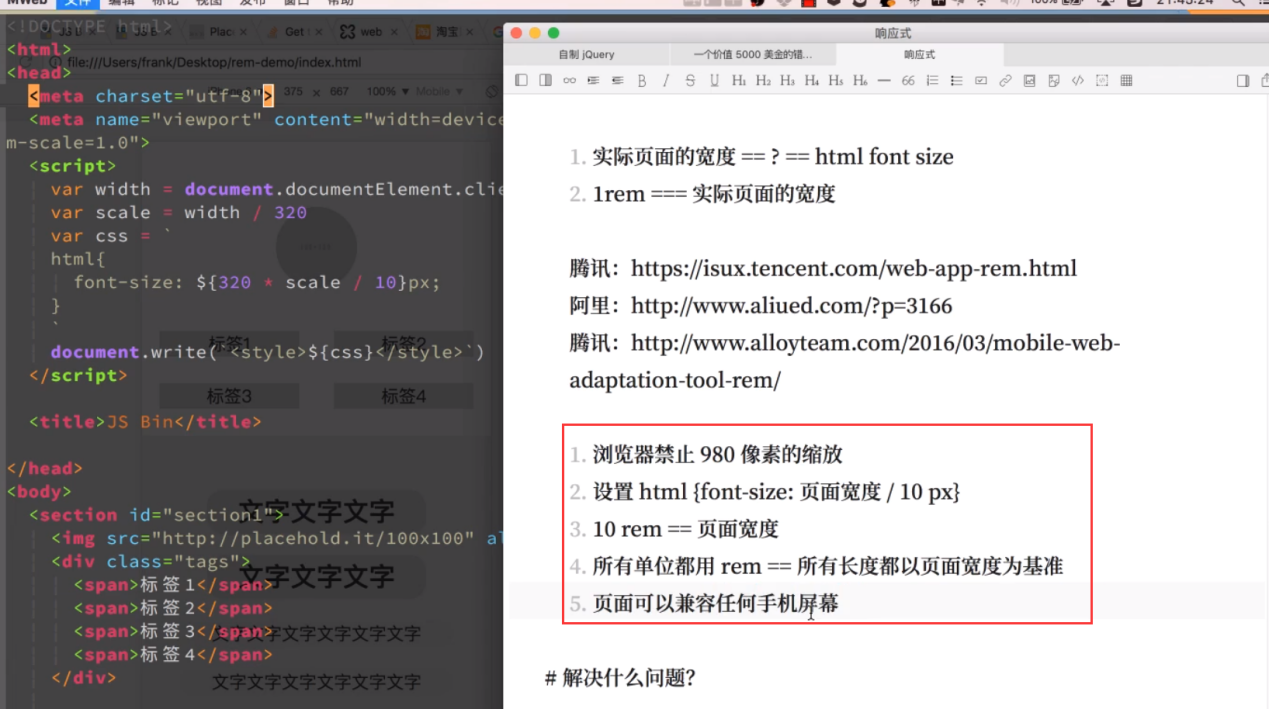
为了方便计算让1rem等于设备宽度的1%.但是这样设置完之后会造成html fontSize的值过小，而chrome浏览器由最小字体设置，一般为12px,也可以手动修改，设置完之后，如果html的fontSize值小于最小值就不会生效，所以导致一些问题，不能按照预期的效果出现，但是移动端一般不会出现这样的现象。



所以这个值可以设置的大一些。让1rem等于屏幕宽度的10分之一。

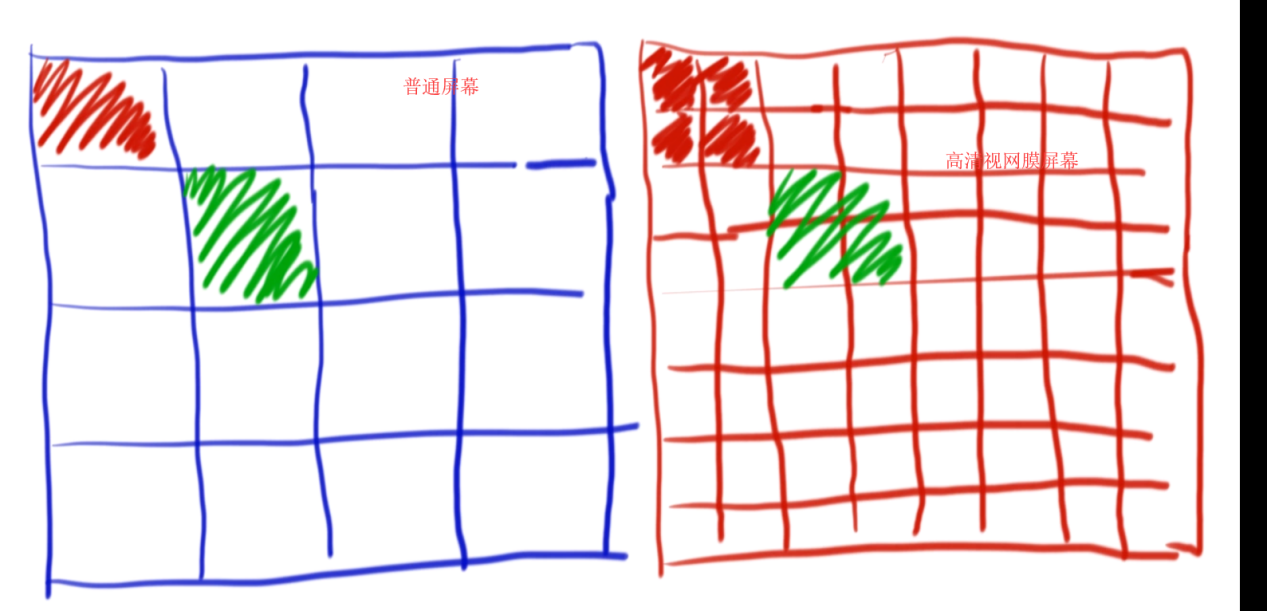


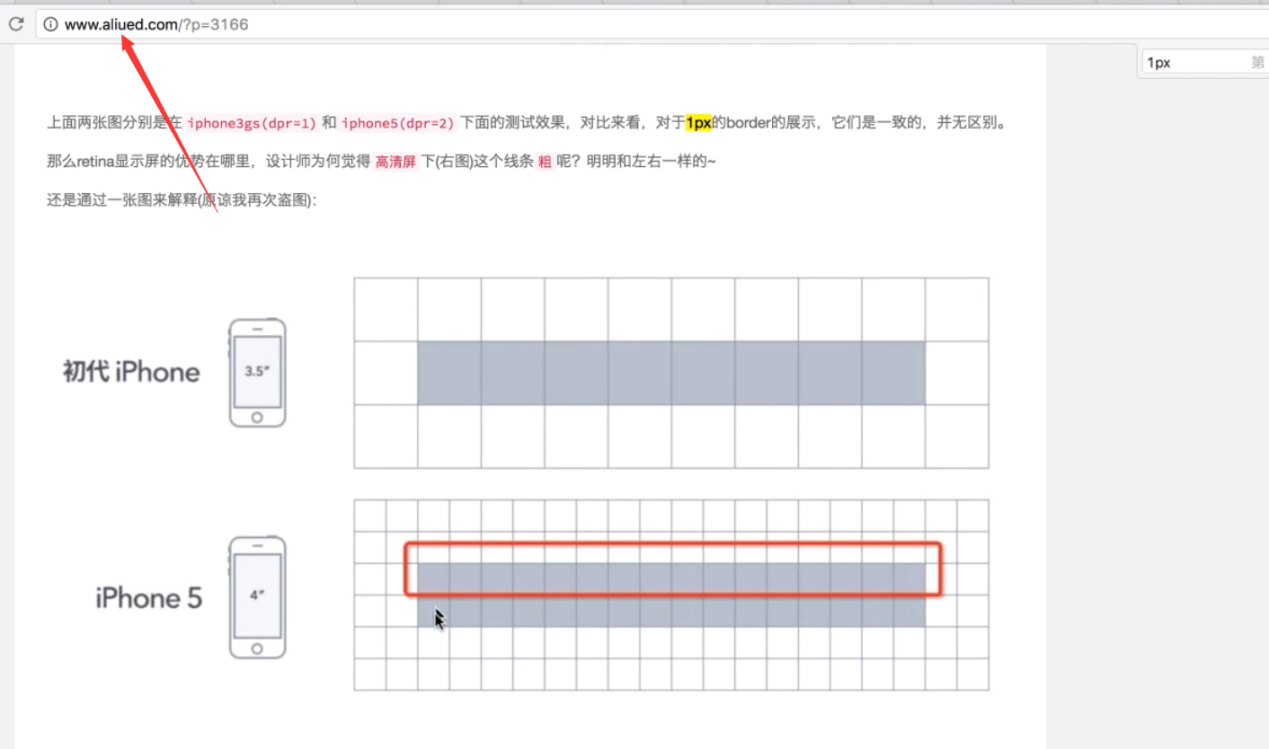




Rem动态方案的五个步骤,第一步，浏览器禁止980像素的缩放，方式：写上meta标签。

核心点在于：用js动态写html的font-size,然后以设备屏幕宽度为基准进行缩放，这样的话就可以兼容所有的手机。



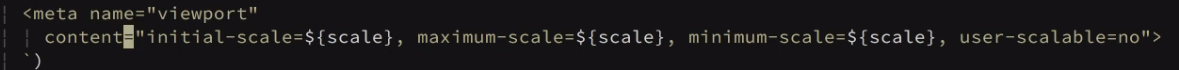


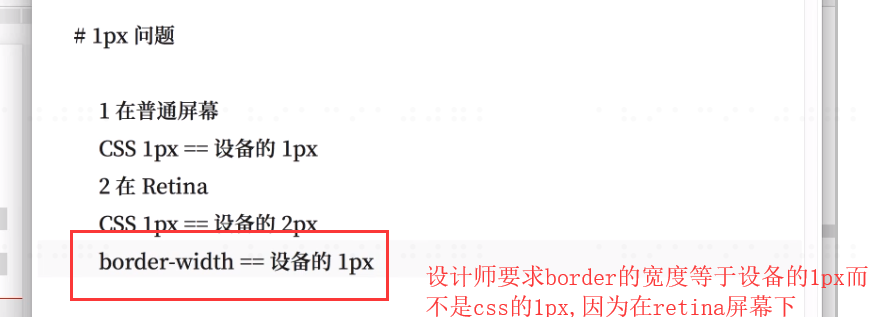
搜索相关帖子进行进一步研究。

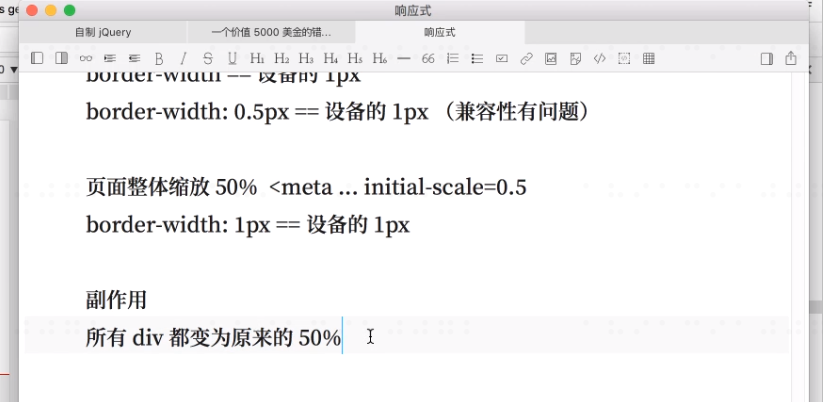
解决1px问题，先缩放整格设备然后扩大htmL的fontSize



如果想要缩放就不能写content=”width=device-width”它的意思是强制不缩放









解决1px问题的思路：先把页面整体缩放，然后扩大所有的rem,倍数等于缩放的倍数，然后把border剩下不扩大，这样就保证了border的像素永远为1px.

## 以下为vue工程化中rem方案

## Vue + vant组件工程化（vuecliX2.0）

#### 一，关于lib-flexible可伸缩布局方案

flexible方案是手淘经过多年的摸索和实战，总结出的一套移动端适配方案。早期的flexible的方案（lib-flexible）通过动态修改viewport的方式来设置视口，出现了很多问题：安卓端1像素问题 、不能与响应式兼容 等等。最新版(amfe-lexible)已经不再修改 viewport ，而是统一使用理想视口。

使用此方案不能在页面写<meta name=”viewport”>标签，否则lib-flexible中的<meta name=”viewport”>标签不会生效。既然会有问题那就淘汰。

原理：利用js动态更改 html的font-size大小

二，

#### **amfe-flexible是在lib-flexible的基础上开发的。解决了lib-flexible存在的一些问题。也可以在**viewport设置理想视口。

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, minimum-scale=1, user-scalable=no">

原理：利用js动态更改html的font-size大小。

1. 将px转化为rem

postcss-plugin-px2rem官方文档：[https://www.npmjs.com/package/postcss-plugin-px2rem](https://links.jianshu.com/go?to=https://www.npmjs.com/package/postcss-plugin-px2rem" \t "_blank)  
postcss-pxtorem官方文档：[https://www.npmjs.com/package/postcss-pxtorem](https://links.jianshu.com/go?to=https://www.npmjs.com/package/postcss-pxtorem" \t "_blank)  
postcss-px2rem官方文档：[https://www.npmjs.com/package/postcss-px2rem](https://links.jianshu.com/go?to=https://www.npmjs.com/package/postcss-px2rem" \t "_blank)

这三种插件postcss-pxtorem和 postcss-px2rem 类似，postcss-plugin-px2rem 这个插件 配置选项上有 exclude 属性，它可以配置 是否对 某个文件夹下的所有css文件不进行从px到rem的转换。我们可以利用这个特性，排除掉我们不要转换处理的文件夹。

如果使用postcss-plugin-px2rem npm install postcss-plugin-px2rem --save-dev

找到根目录下的.postcsssrc.js文件进行配置

module.exports = {

"plugins": {

"postcss-import": {},

"postcss-url": {},

// to edit target browsers: use "browserslist" field in package.json

"autoprefixer": {},

"postcss-plugin-px2rem": {

remUnit: 100,

mediaQuery: true,

exclude:"/node\_modules/i",//排除文件夹

selectorBlackList: ['html', 'mint-', 'mt-', 'mpvue-', 'calendar', 'iconfont'], //在rem.js全局作用下 排除指定的文件的影响

propBlackList: ['border']

}

}}

2,使用 postcss-pxtorem插件 不能排除文件及文件夹

npm install postcss-pxtorem --save-dev同样找到根目录下的.postcsssrc.js文件进行配置

module.exports = {

"plugins": {

"postcss-import": {},

"postcss-url": {},

// to edit target browsers: use "browserslist" field in package.json

"autoprefixer": {},

"postcss-pxtorem": { // 此处为添加部分

rootValue: 32, // 对应16px 适配移动端750px宽度

unitPrecision: 5,

propList: ['\*'],

selectorBlackList: [],

replace: true,

mediaQuery: false,

minPixelValue: 0

}

}}

在这里postcss-pxtorem还解决了1像素线的问题，如果想写1像素，写成1PX就不被解析，他就是真正意义上的1px

1. postcss-px2rem

vue项目中postcss-px2rem的三种使用方式

<https://blog.csdn.net/tomotoCoder/article/details/99464828>

1. vant组件

# **[使用vue vantUi框架 根字体是37.5 和我们设置的默认根字体75不一致，导致页面组件样式变小](https://www.cnblogs.com/yimei/p/11319657.html)所以解决办法如下**

<https://www.cnblogs.com/yimei/p/11319657.html>

总结：1，使用如下方式可以解决上述所有问题

一，

npm install amfe-flexible --save-dev

二，在main.js中引入amfe-flexible

import 'amfe-flexible'

三，viewport设置理想视口

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, minimum-scale=1, user-scalable=no">

1. npm install postcss-pxtorem --save-dev同样找到根目录下的postcsssrc.js文件进行配置 也可以安装postcss-px2rem

让vant使用37.5计算，让自己的页面使用75计算，这样不至于让页面变得太小，不能用postcss-plugin-px2rem 中的exclude来排除vant,否则在不同的屏幕设备上vant组建的展现效果也不一样，因为vant的单位是px.排除vant后,不让vant转化为rem,vant组件不会随屏幕设备等比例缩放。

module.exports = ({ file }) => {

let rootValue

// vant 37.5 https://github.com/youzan/vant/issues/1181

if (file && file.dirname && file.dirname.indexOf('vant') > -1) {

rootValue = 37.5

} else {

rootValue = 75

}

return {

plugins: [

autoprefixer(),

pxtorem({

rootValue: rootValue,

propList: ['\*'],

minPixelValue: 2

})

]

}

}