# webapp

# 前置知识

## 浏览器分类

**PC端常见浏览器：**360浏览器、**谷歌浏览器、火狐浏览器**、QQ浏览器、百度浏览器、搜狗浏览器、**IE浏览器**。

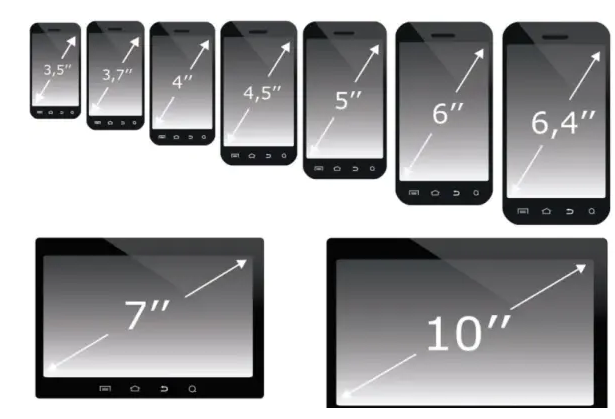
**移动端常见浏览器：**UC浏览器，QQ浏览器，欧朋浏览器，百度手机浏览器，360安全浏览器，谷歌浏览器，搜狗手机浏览器，猎豹浏览器，以及其他杂牌浏览器。

国内的UC和QQ，百度等手机浏览器都是根据webkit修改过来的内核，国内尚无自主研发的内核，就像国内的手机操作系统都是基于Android修改开发的一样。

**在写手机端网站时，只需要兼容webkit内核就OK了~**

## 屏幕尺寸

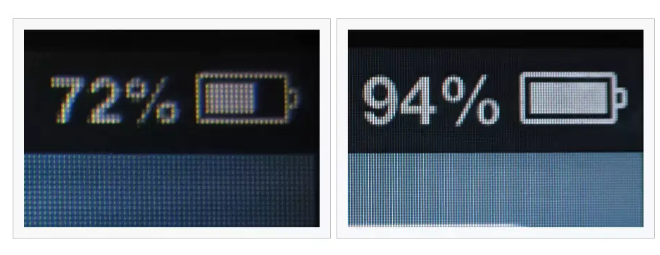
如图所示两个对角线的长度就是这个屏幕的尺寸。



1英寸(in)=2.54厘米(cm)

## 物理像素（屏幕像素/屏幕分辩率）

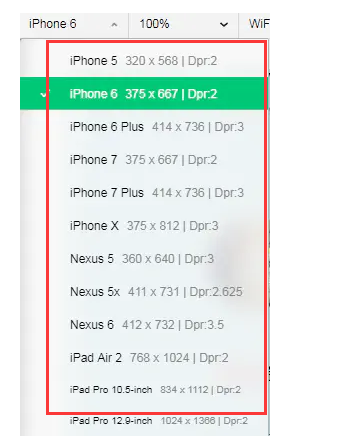
物理像素就是屏幕上面的一个个的发光点，苹果6的分辨率为750x1334，在横上有750个发光点，在垂直方向上有1334个发光点。



不同的手机，它的屏幕分辩是不同的，我们开发时，不需要关注屏幕分辩率。

## 设备独立像素

手机在出厂，都定了一个宽度，这个宽度在300~500之间左右，看浏览器中的手机模型，如下：



# 视口

## 为什么PC端的网页在移动设备下，整体被缩小

在N前年，那个时代，电脑显示器的主流分辩率是1024，把PC端网页它的版心定为980px。显示的非常完美的。后机电脑的显示的分辩也添加到了非常多的，也就是说，现在PC端网页的版以也变大，如果1000,1024....

一个PC端的网页宽度是980（不要想高度），在手机端访问如何？

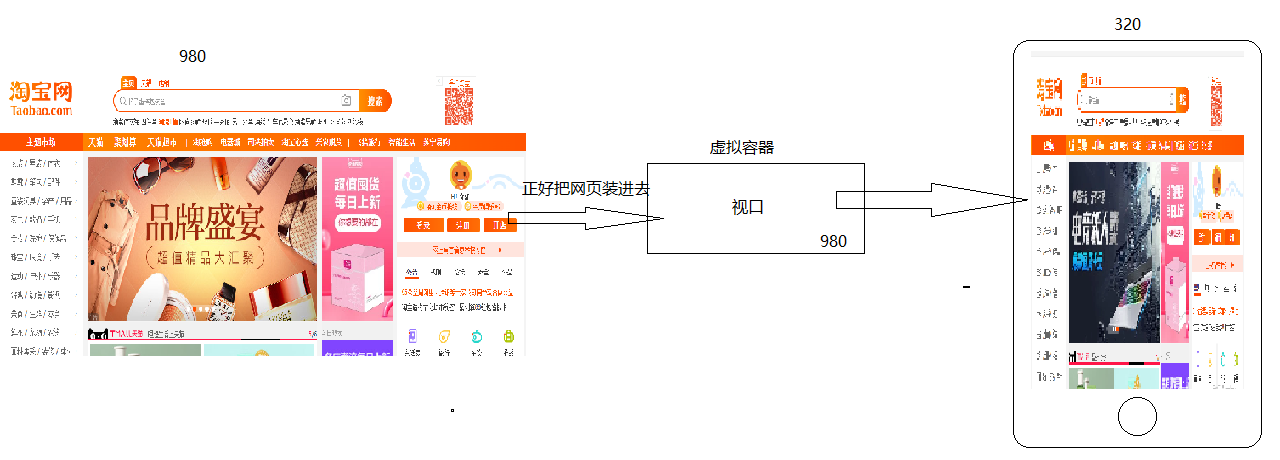
最早时期访问一个网站的效果，如下：



把上面的网站先装到一个虚拟的容器，假如这个容器也是980，这样正好卡住整个网页。接着再把这个虚拟的容器，塞到手机中。

**也就是说，一个PC端网页，到手机端，分两步走：**第一步：从PC端到虚拟容器中（视口 绝大部分的手机都是980）

第二步：从虚拟容器塞到手机中，会把虚拟容器进行压缩，



其中第一步如果PC上的网页大于980，会在虚拟容器中产出滚动条。

其中第二步假如手机是320，把980的虚拟容器塞到320里，会被整体压缩。

**320就是设备独立像素。**

## 为什么是980？

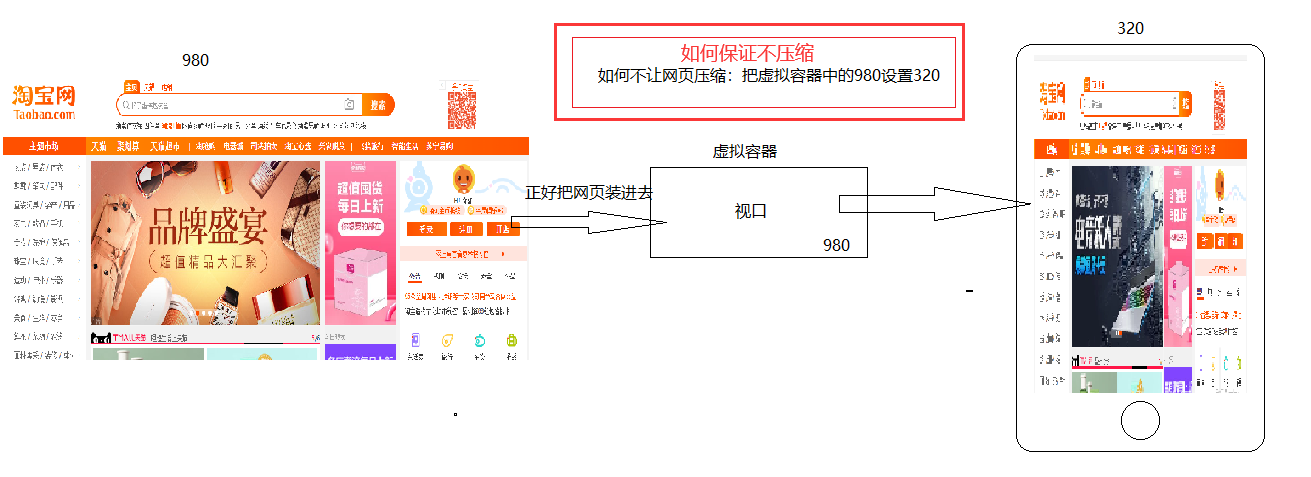
是这样的，最N年前，PC端的网页的版心通常是980，为了把这个网页塞到手中，乔帮主它就定了虚拟容器的宽度就是980，后来，安卓手机也遵循了这样一个约定，默认虚拟容器（默认视口）的宽度就定为了980。

大部的手机的默认视口都是980，但是黑莓，IE它们的默认是视口是1024。

使用980这个虚拟容器，放到手机端，还是会被压缩，但是有个好处，你可以看到整个网页，像在高空3000米，俯视整个网页。此时也有不足之处，是页面被压缩了，你想看清楚，两根手指去放到网页。我们就想，如果不让虚拟容器压缩，我们每一个手机在出厂时，都有一个叫设备独像素的东西，和手机的分辩率是没有关系。这就是约束之后的视口。

我们如果把这个虚拟容器设置小一点，小到正好等于手机的设备独立像素，例如，你把虚拟容器（视口）设备成375，手机的宽度（不是指分辩率，指设备独立像素）也是375，此时，你把虚拟容器放到手机里面，正好被手机卡住，也不会压缩了，这样就完美显示了。

如果保证不压缩呢？



## 设备独立像素

因为980大大，放手机中，会压缩，所以不同手机在出厂时，它又定义了一个叫约束视口的东西，默认视口是980，它定义的比980小的多，如果你把虚拟容器也定义和线束视口一样，这样就不会被压缩了。

常见手机的约束视口，在模拟器中看，如下：



....

默认视口是980，放到手机中会被压缩，如果**设置视口=手机的宽度（设备独立像素）**，此时虚拟容器就不会被压缩了

最好记一些，常见的手机的**设备独立像素**（**手机的宽度/手机的尺寸**（不是指手机分辩率））：

* iPhone5/SE 320
* iPhone6/7/8 375
* iPhone6/7/8plus 414
* 安卓端：320, 360, 420, 540....

## 设置视口的宽度

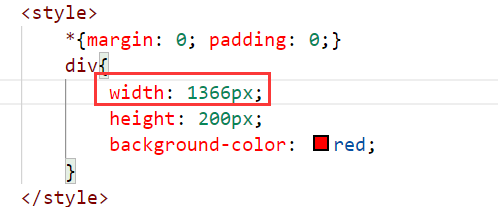
**看如下代码：**



**在PC端显示，如下：**



**把这个盒子定为1366，如下：**

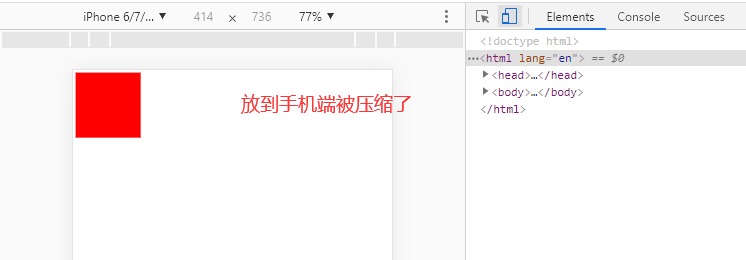




没有设备视口的情况下，视口的宽度是980，放到手机端会被压缩，如下：



**把上面的盒子放到手机端，如下：**



在手机端css代码中写的1px 并不和手机屏幕上的发光一一对应。分析如下：

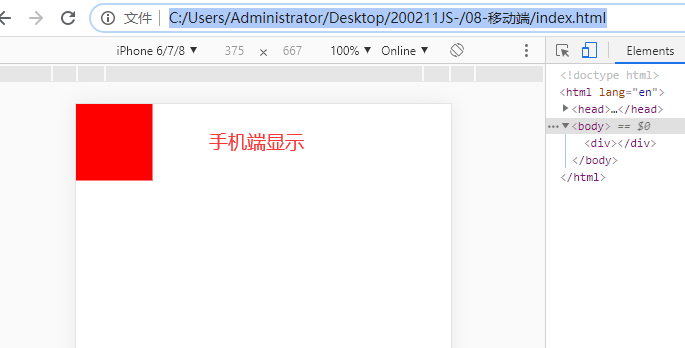


在水平方向上有750个发光点，但是水平方向上css像素点不是750px。



**问题：在电脑端的一个盒子，在手机端就被压缩了？**





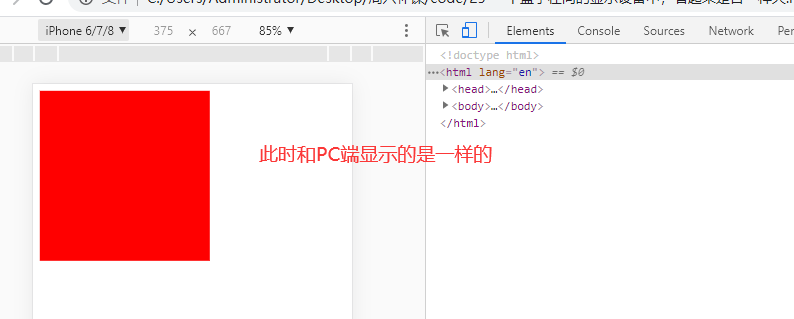
**再强调一下，把电脑端的网站放到手机端，分两步：**

1. 先放到980的虚拟容器中
2. 把虚拟容器压缩，压缩后放到手机中

**你想让它不压缩，你只需要把视口的宽度设置成手机的宽度（设备独立像素），如下：**



再看一下效果，如下：



但是我们知道，不同的手机，在出厂时，它的设备独立像素是不一样的，你直接写死了一个375，但是其它的手机可以不是375，怎么办，你可以这样做：



此时，就相当于把视口的宽度设置成了与当前移动设备的设备独立像素一样大了，所以就不会有压缩的情况了。

**总结：**



**为什么1个css像素 会 映射到两个物理像素呢？**

答：lRetina（视网膜屏幕） 在iphone3中一个css像素是映射成1个物理像素，从iphone4开始，提出来高清屏概念，这个时候，1个css像素就会映射到2个或3个物理像素。

## 5，viewport的属性

**width 视口的宽度 通常设置成device-width**

**initial-scale 设置页面的初始缩放值**

maxinum-scale 允许用户的最大缩放值 是一个数字 可以是小数

mininum-scale 允许用户的最小缩放值 是一个数字 可以是小数

user-scalable : 用户是否可以手动缩放，值可以是：①yes、 true允许用户缩放；②no、false不允许用户缩放

height 高度，没什么用

**width=device-width  等价于 initial-scale=1.0 为什么？**

答：**视口 = 设备独立像素宽度（375） / inital-scale**

上面的两行代码都有各自的兼容性问题，所以说通常，我们两个都写。如下：



### inital-scale

也是用来设置视口的，前面说了，width是用来设置视口，其实有这样一个公式，如下：

**视口 = 设备独立像素宽度 / inital-scale**

**可以通过inital-scale间接地设置视口，操作如下：**



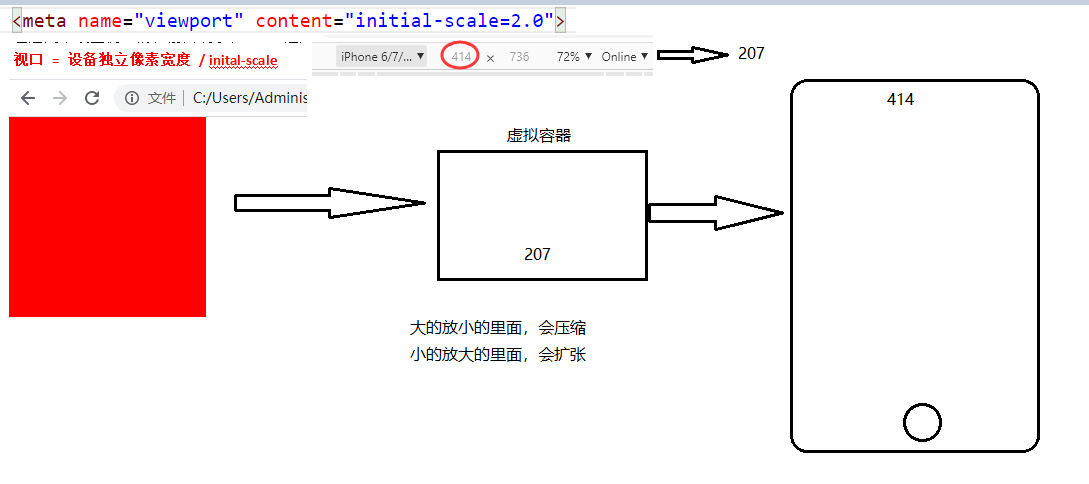


现在不让inital-scle是1，设置成2，如下：

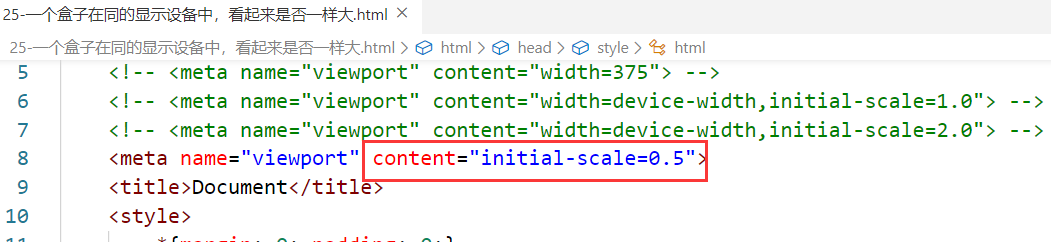




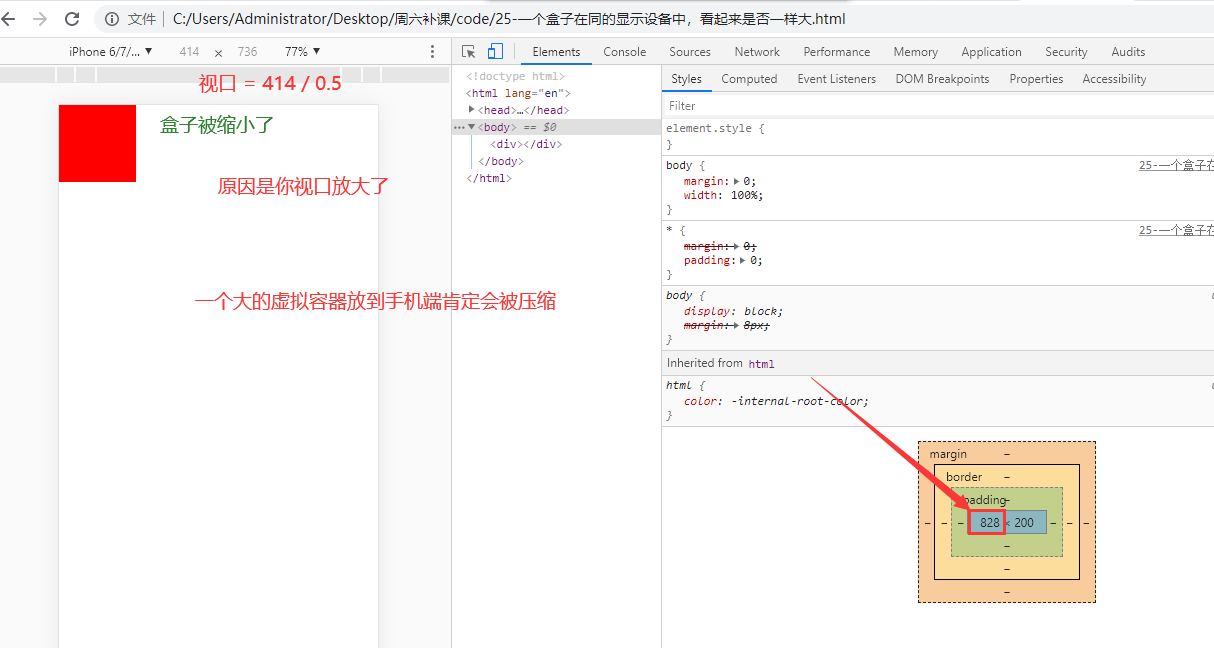
分析：



**那假如我把inital-scale=0.5，效果如下：**

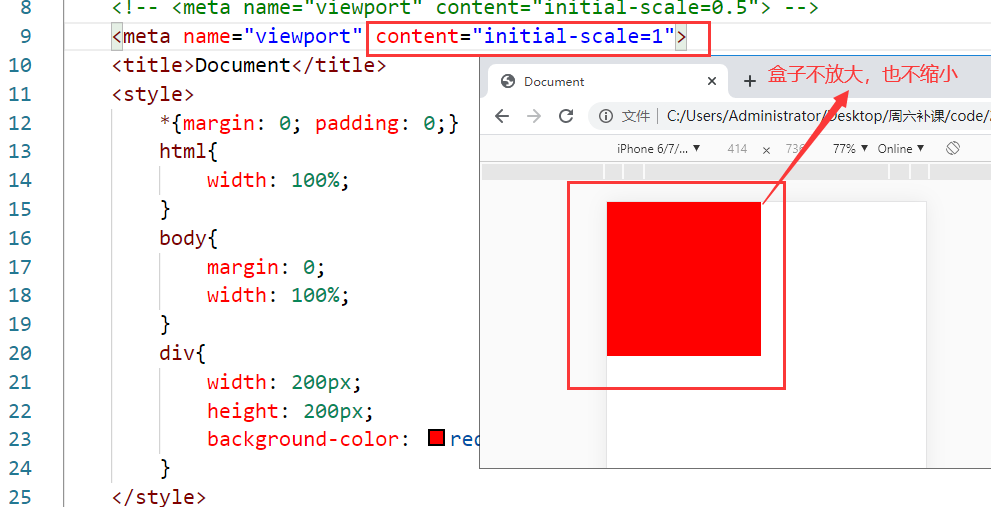


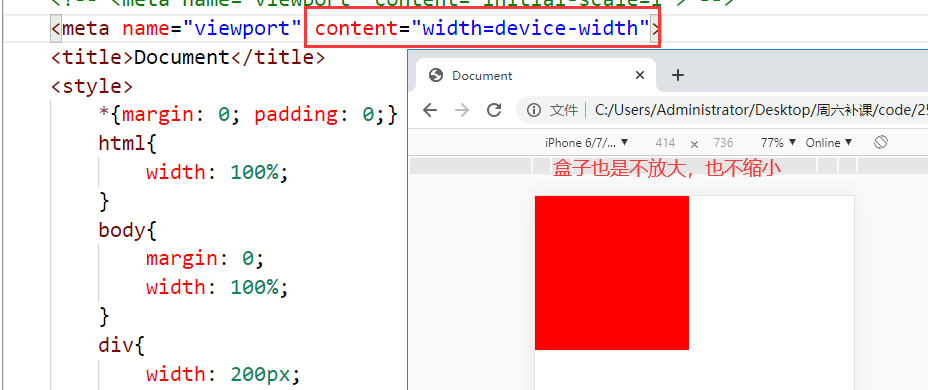
效果如下：



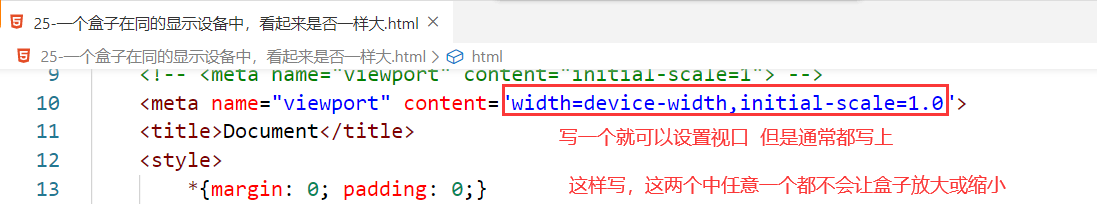
此时，我们设置视口，就有两种方式，如下：

* width=device-wdith
* inital-scale = 1

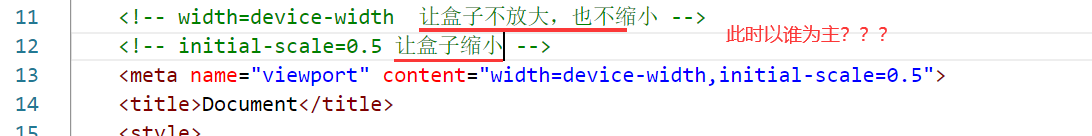


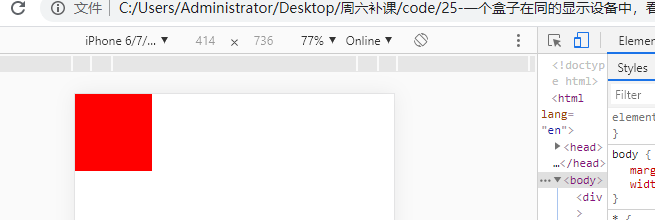


如果有一个人，它设置了width=device-wdith，又设置了inital-scale = 1，以谁为主。如果说你这样设置了，不放大，也不缩小，如下：

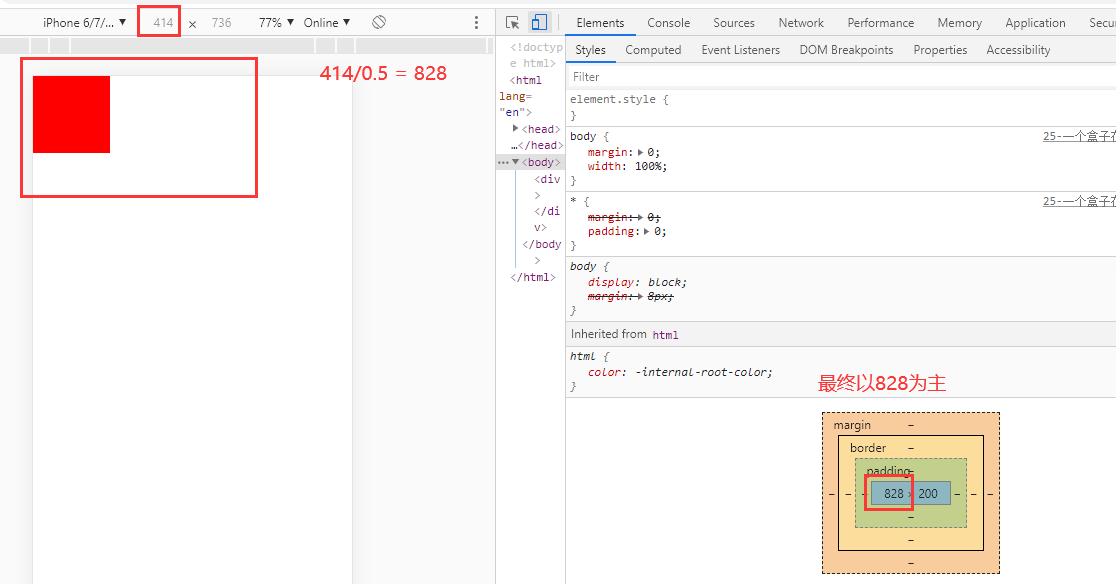


假如人家这样设置，如下：



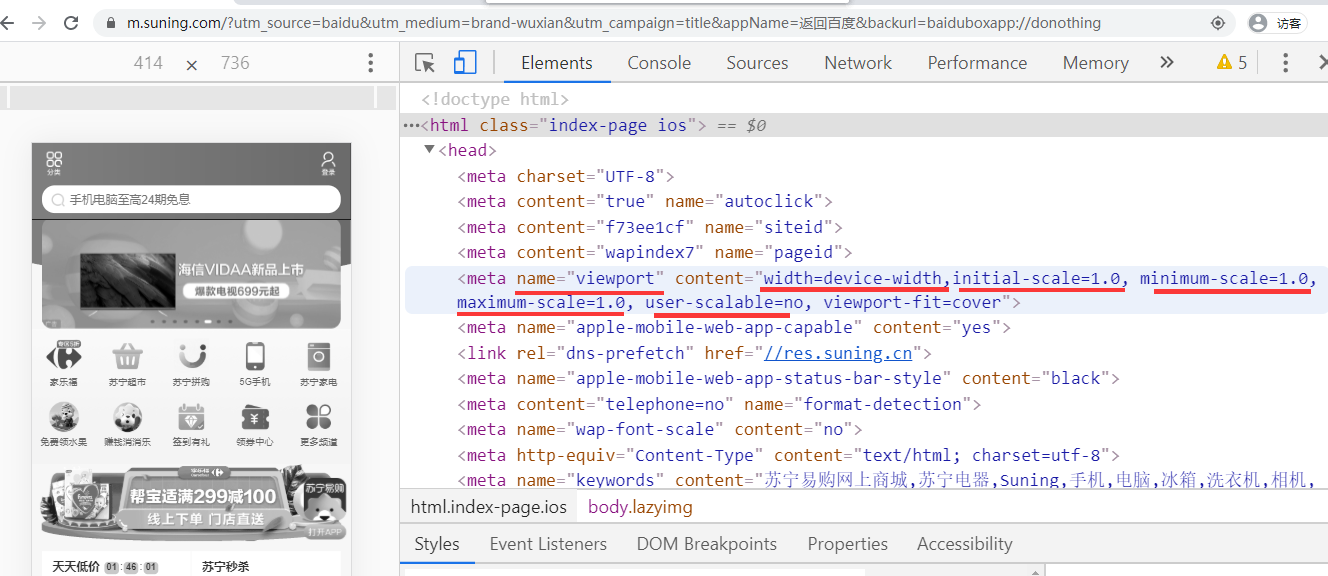
效果如下：  


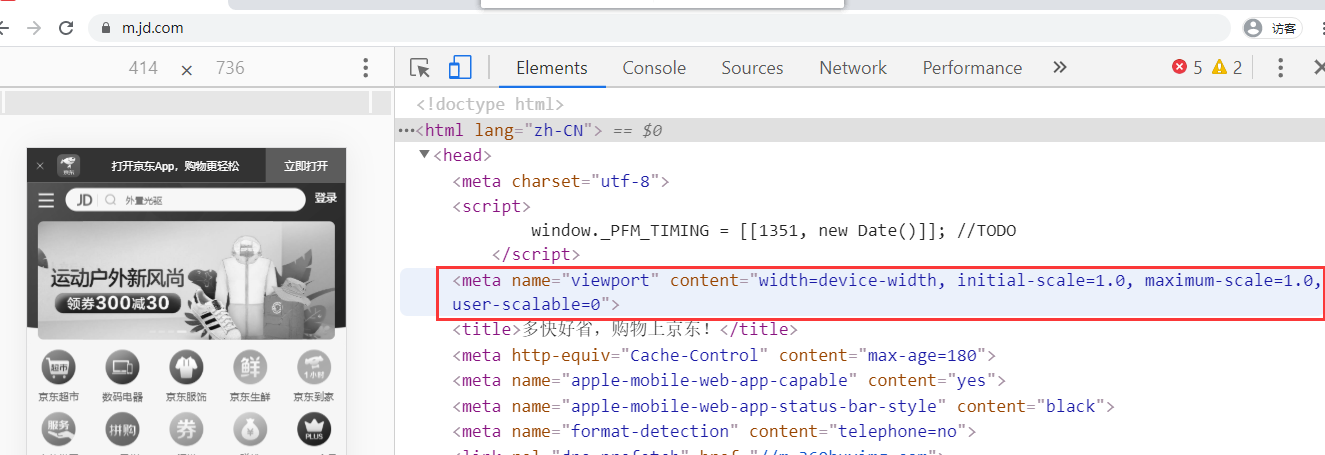
发现盒子缩小了，也就是说上面的两个属性设置矛盾了，最终的结果为大的为准。

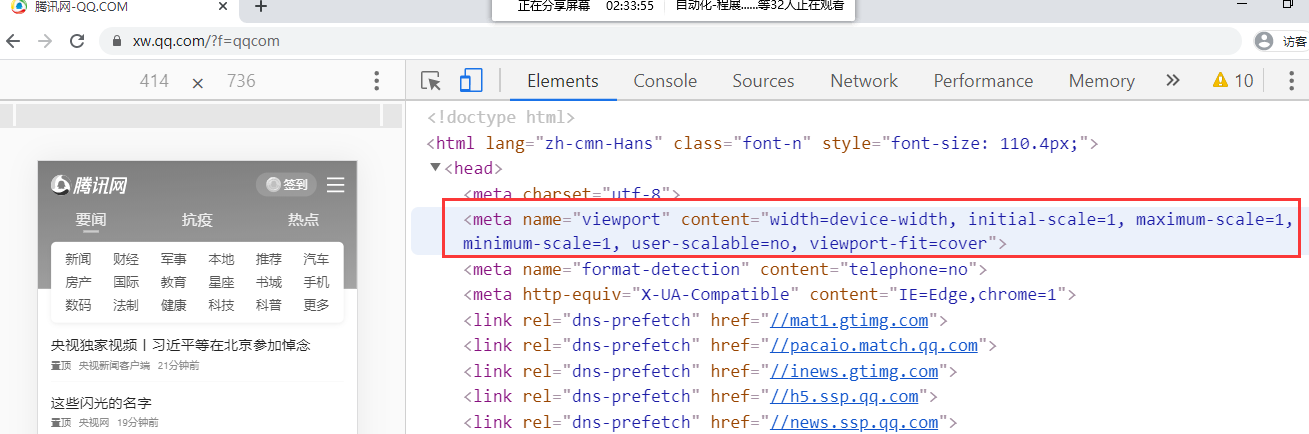


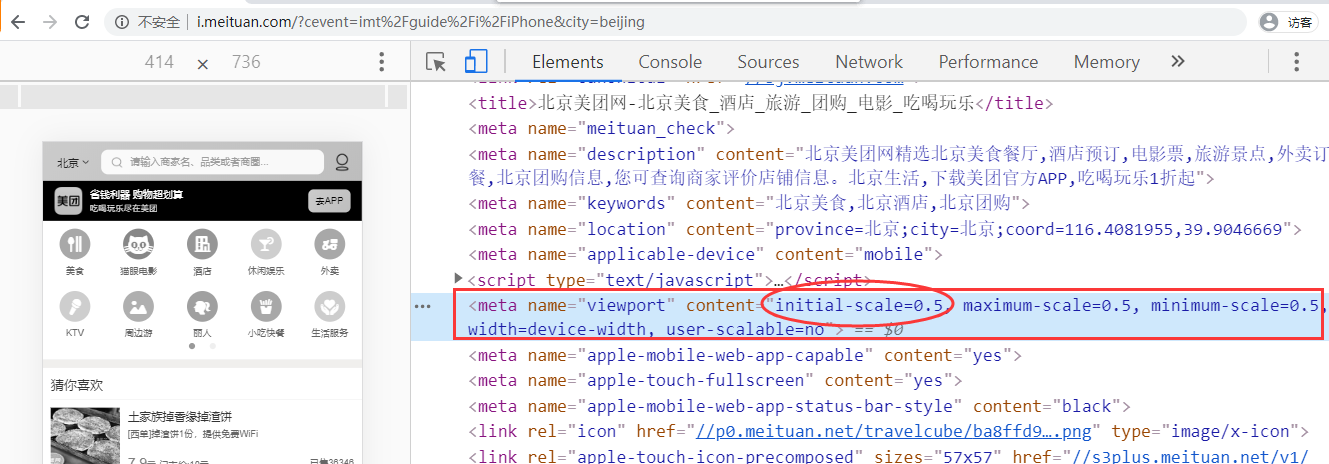


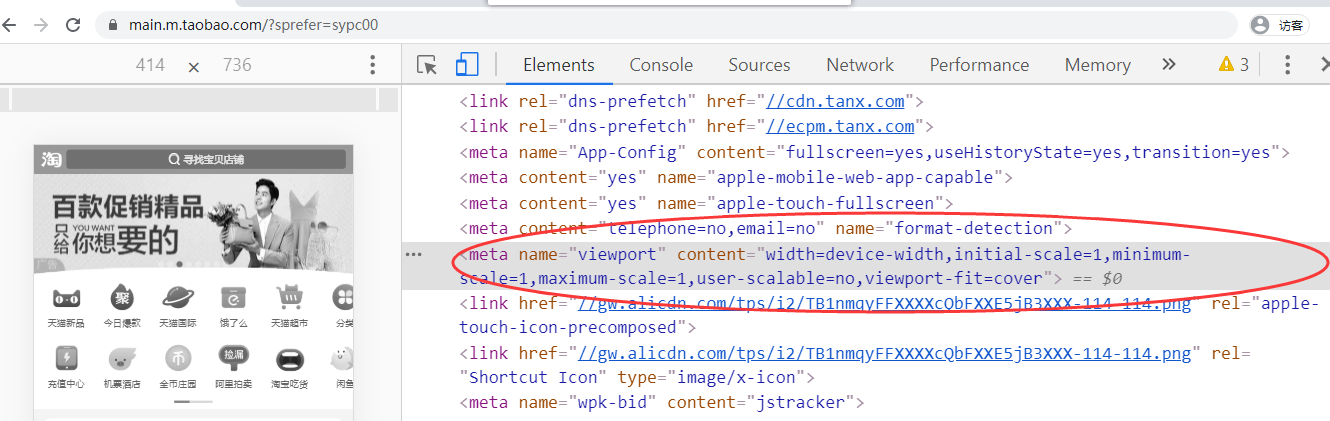
## 6，找一些移动端网站看一下



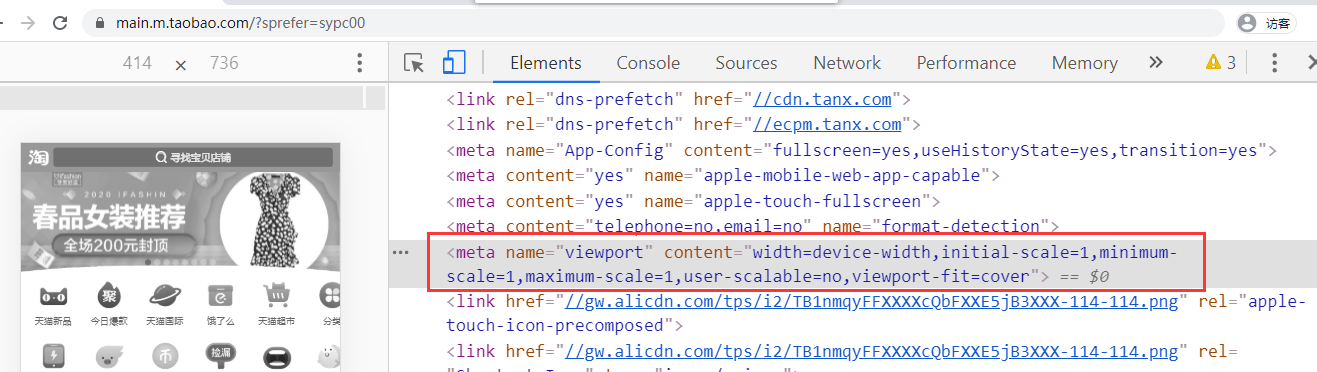








设置视口的经典代码，如下：



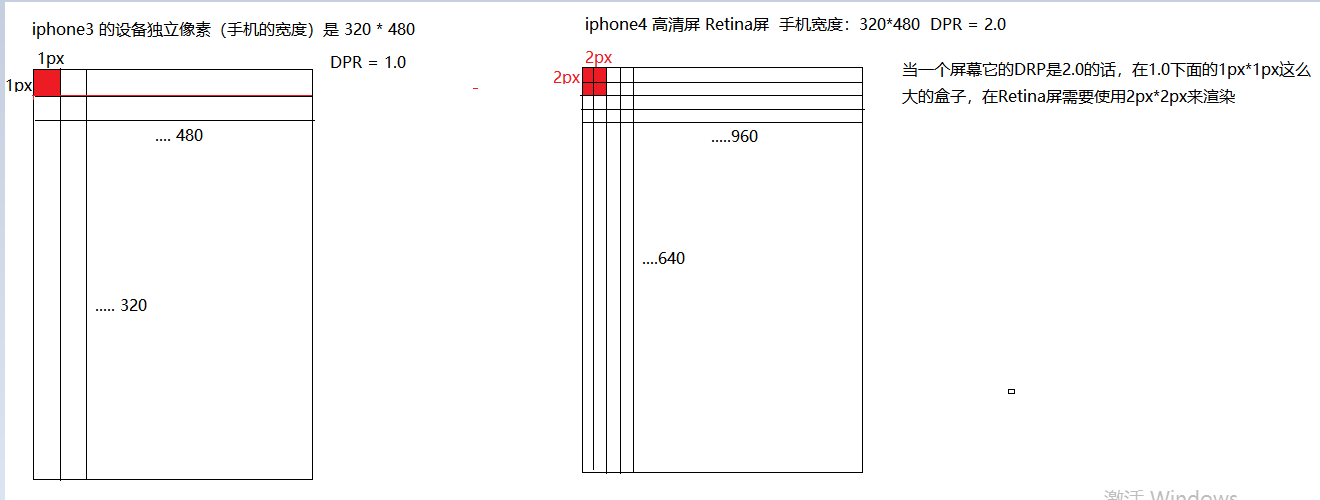
**最经典的设置：**

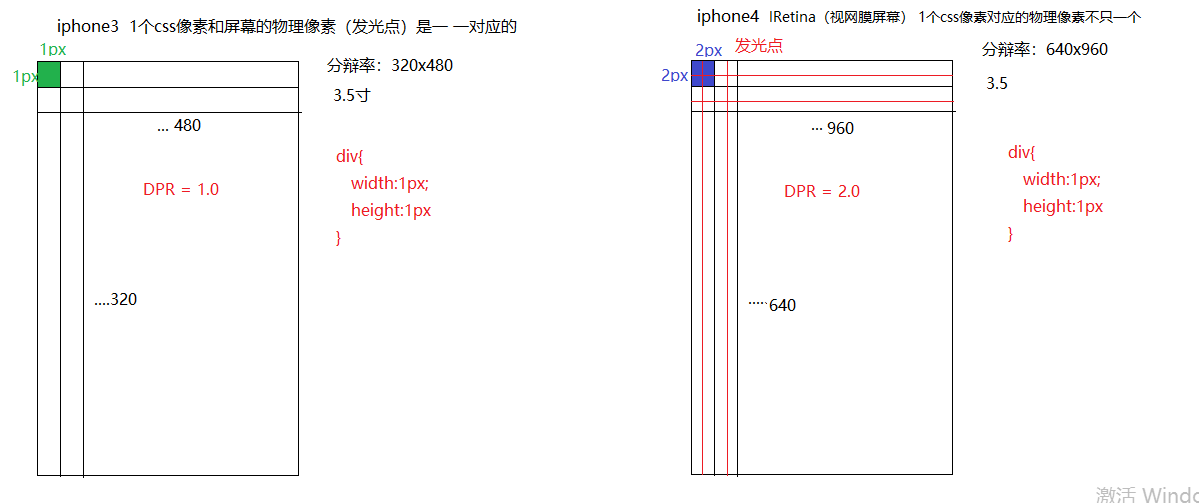
<meta name="viewport" content="width=device-width,initial-scale=1,minimum-scale=1,maximum-scale=1,user-scalable=no,viewport-fit=cover">

## DPR

前面，我们了解了手机端的设备独立像素，也就是常见手机的尺寸。以iphone6为例，它的是宽度是375。但是设计师，她给我们的设计图通常是iphone6作为一个标准，按理说，设计师应该给我一个375尺寸的设计图，但是我们拿到750宽的设计图，为什么需要把设计图设计成是手机宽度二倍呢？ 此时我们就需要了解一下，什么是dpi。

画图来说：



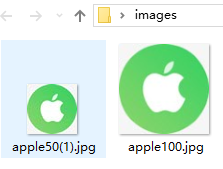


不同的手机的DPR，如下：



现在有一张100\*100的图片，它在DPR=1.0设备中可以完美显示，当我们把它放到DPR=2.0的设备中，实际呈现的时候，是按200\*200进行的，此时展示出来的图片相当于扩大了一倍来展示了，此时图片变模糊。

为了不让图片模糊，在DPR=2.0的设备中，我们需要准备一张200\*200的图片，这个图片，叫二倍图，就是说，在DPR=1.0的设备，使用原图，在DRP是=2.0中，使用二倍图，当然，在DPR=3.0，使用二倍图... 如下：



如一个Logo的图片，长宽是100\*100，在设计时，通常会有如下的几个尺寸：

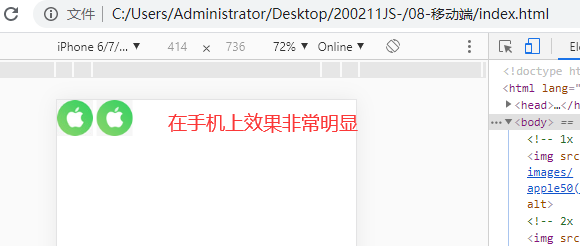
* logo.png
* [logo@2x.png](mailto:logo@2x.png)
* [logo@3x.png](mailto:Logo@3x.png)

目的：在不同DPR的设备中都可以完美的显示的图片。

但是，在真实开发，为了方便，我们通常直接使用2倍图，即使在DPR=1.0的设备中，也使用2倍图，也不会失真，只是图片比较大，加载会慢一点。

在手机上测试，原图和2x倍图的效果，如下：

也就是说，以iphone6为例，iphone6的尺寸是375，通过得到750的设计图。写页面，该怎么写，还怎么写。



## 二倍图

物理像素点指的是屏幕显示的最小颗粒，是物理真实存在的。这是厂商在出厂时就设置好了,比如苹果6 是 750\* 1334 。我们开发时候的1px 不是一定等于1个物理像素的，一个px的能显示的物理像素点的个数，称为物理像素比或屏幕像素比

lRetina（视网膜屏幕）是一种显示技术，可以将把更多的物理像素点压缩至一块屏幕里，从而达到更高的分辨率，并提高屏幕显示的细腻程度。对于一张 50px \* 50px 的图片,在手机或 Retina 屏中打开，按照刚才的物理像素比会放大倍数，这样会造成图片模糊，在标准的viewport设置中，使用倍图来提高图片质量，解决在高清设备中的模糊问题，通常使用二倍图， 因为iPhone 6 的影响背景图片 注意缩放问题。

# webapp分类

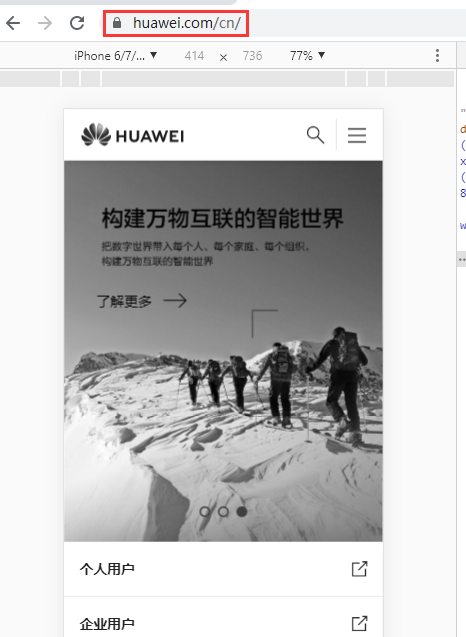
## PC端和移动共用一套代码（响应式网站开发）

比较适合页面比较简的网页，比如公司的官网。如：华为官网，猎豹官网...

**看一下，华为的官网，如下：**



**手机端，如下：**

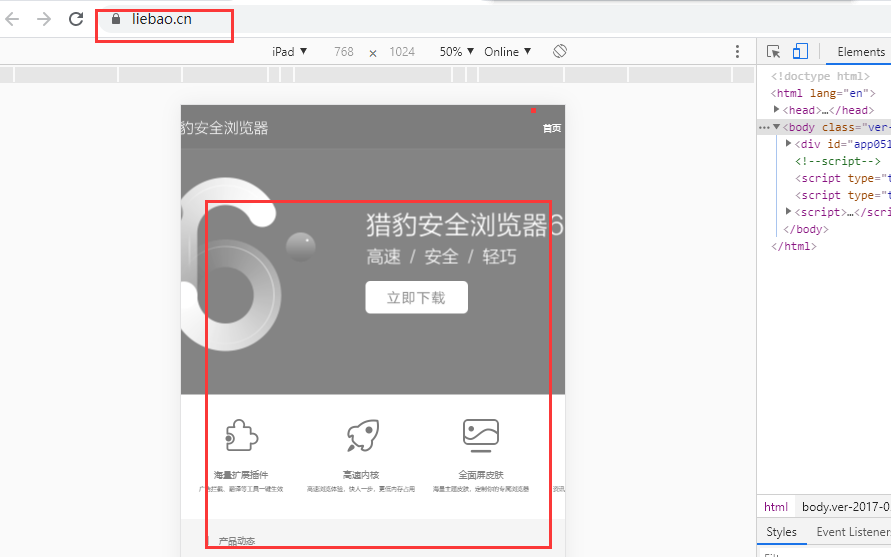


PAD端，如下  


**再看一下猎豹官网，如下：**



**手机端，如下：**



这属于PC端和移动端共用一套代码，我们叫这样的网站是响应式的网站。

**实现方案如下：**

先做PC端，在CSS中有一个叫@media的东西，类似于C语言中if判断，它可以根据不同的分辩率加载不同的样式，@media叫媒体查询，这样的网站一般都比较简单，学习@media也比较简单，在不同尺寸的设备下，使用样式是不一样的。

**结论：**

* 媒体查询
* bootstarp

## PC端和手机端是两套代码（针对手机端专门写一套代码）

如访问一些比较复杂的网站，很少做成响应式，如淘宝，京东....

看一下淘宝PC端，如 ：



换到移动端，如下：



换到手机端，就是另一个项目了。我们最近要学习的不是响应式开发，就是针对手机的网站，也叫webapp。

**布局方案：**

* 流式布局（百分比布局）
* flex 弹性布局（强烈推荐）
* less+rem+媒体查询布局
* 混合布局

# 移动端开发注意点

## 移动端兼容

移动端浏览器基本以 webkit 内核为主，因此我们就考虑webkit兼容性问题，我们可以放心使用 H5 标签和 CSS3 样式。同时我们浏览器的私有前缀我们只需要考虑添加 webkit 即可。

## 移动端初始化样式



移动端 CSS 初始化推荐使用： <http://necolas.github.io/normalize.css/>



## 使用新的盒子模型

传统盒子模型：盒子的宽度 = width + border + padding

新的盒子模型：CSS中设置的宽度width 里面包含了 border 和 padding

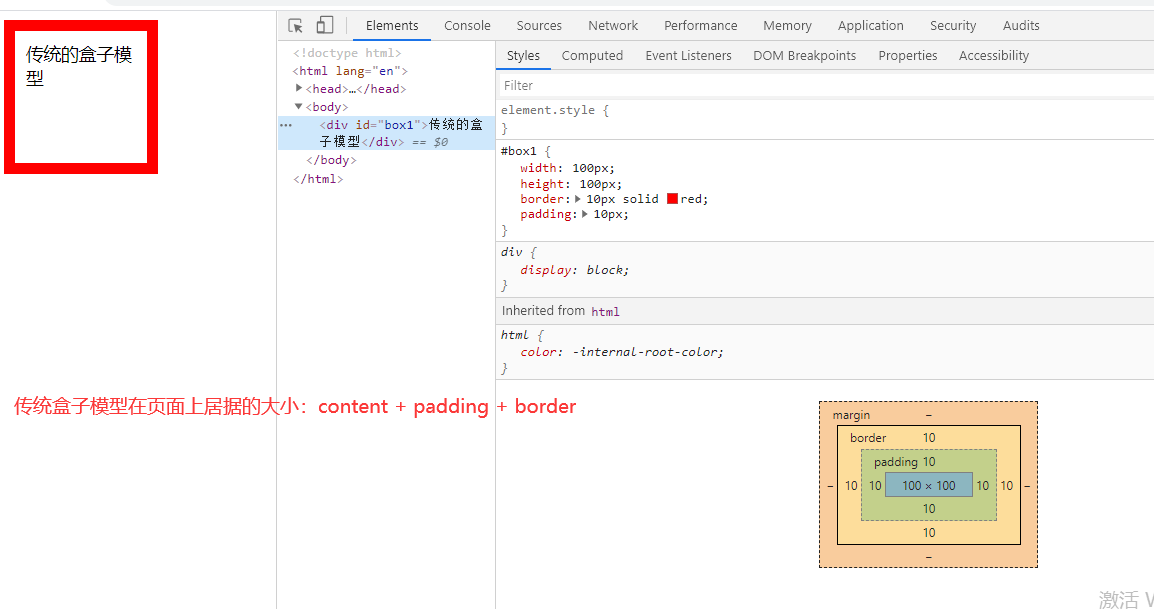
box-sizing: border-box;

**使用新的盒子模型：**

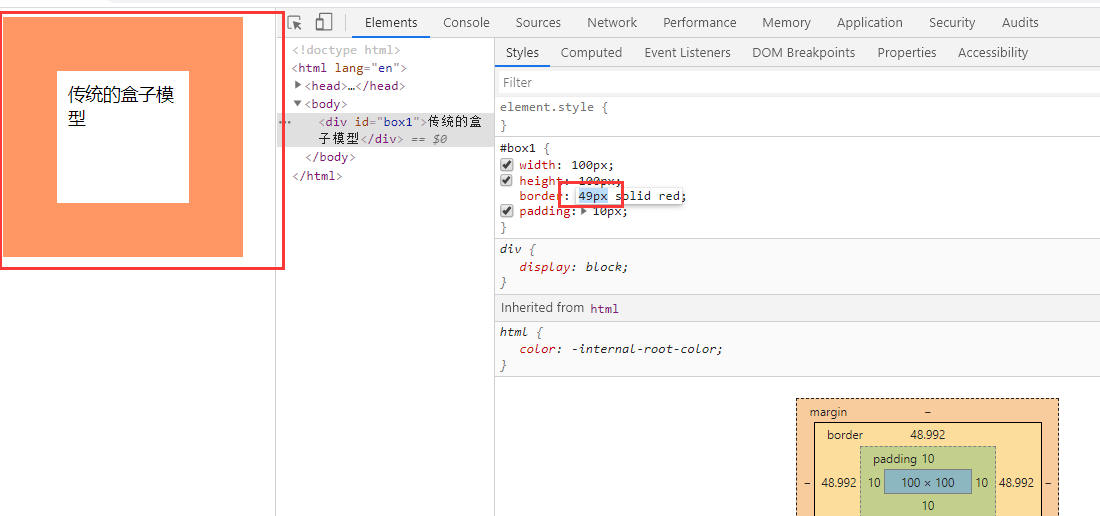
**box-sizing: border-box;**

**-webkit-box-sizing: border-box;**

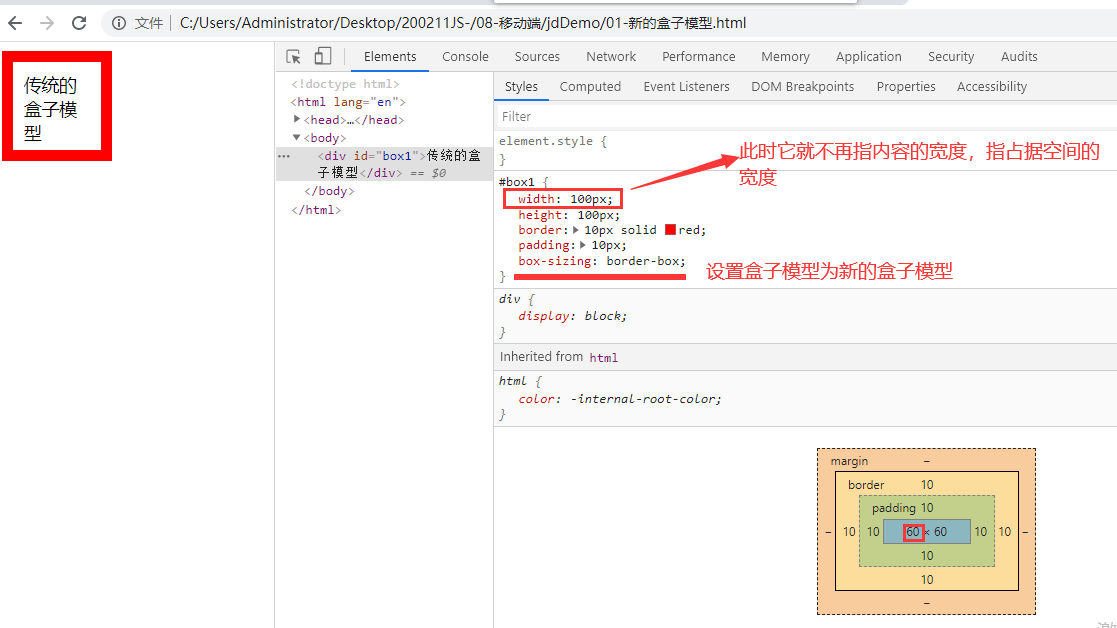
传统的盒子模型如下：



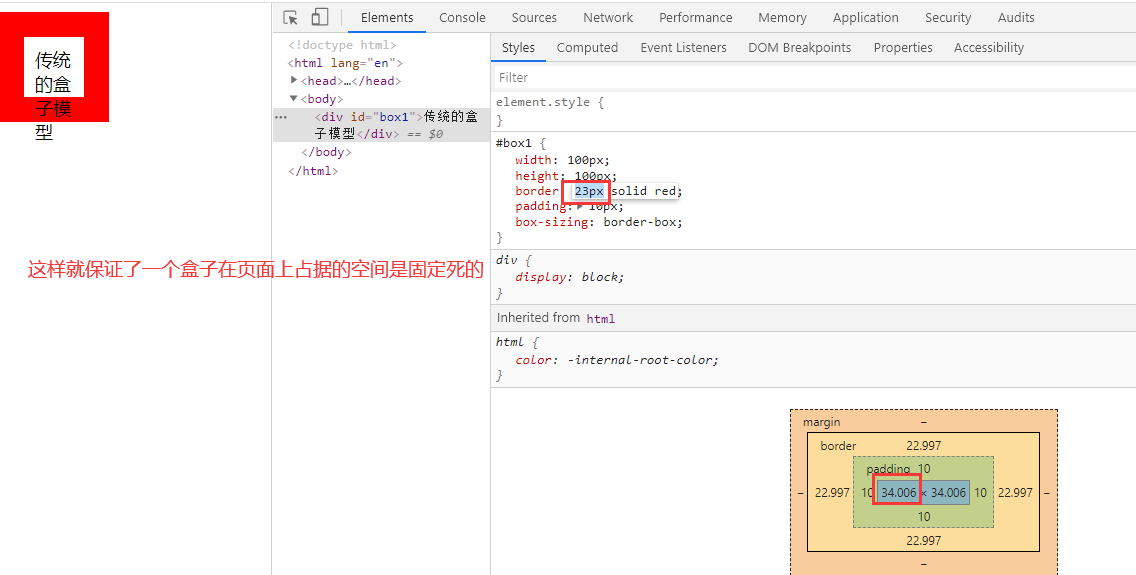
不足：当增加了padding和border时，盒子占的空间就变大了。如下：



新的盒子模型，只需要设置：box-sizing: border-box; 如下：



当我们增加了border和padding时，不会改变占据空间的大小：



## 特殊样式处理



**去掉点击高亮：**

-webkit-tap-highlight-color: transparent;

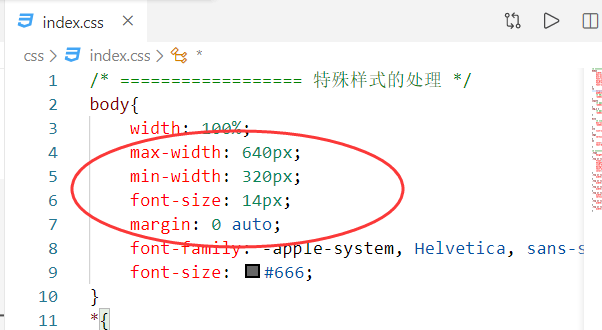
**在移动端浏览器默认的外观在iOS上加上这个属性才能给按钮和输入框自定义样式：**

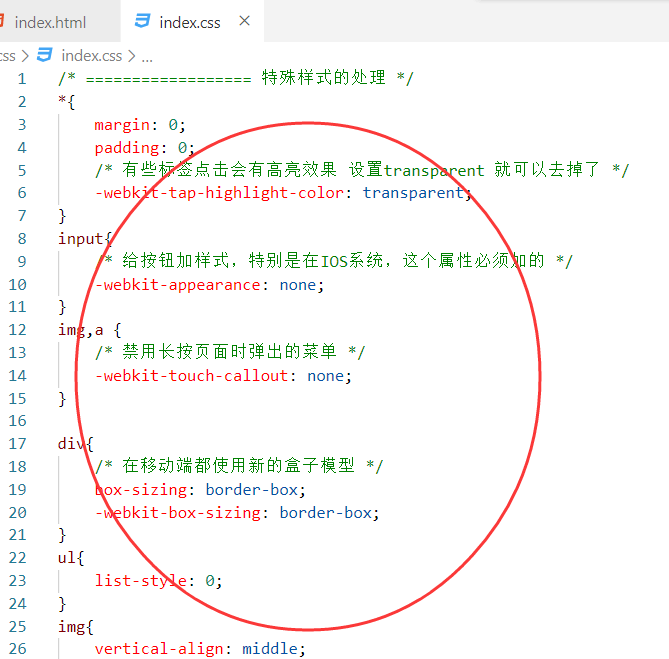
-webkit-appearance: none;

**禁用长按页面时的弹出菜单：**

img,a { -webkit-touch-callout: none; }

**代码如下：**



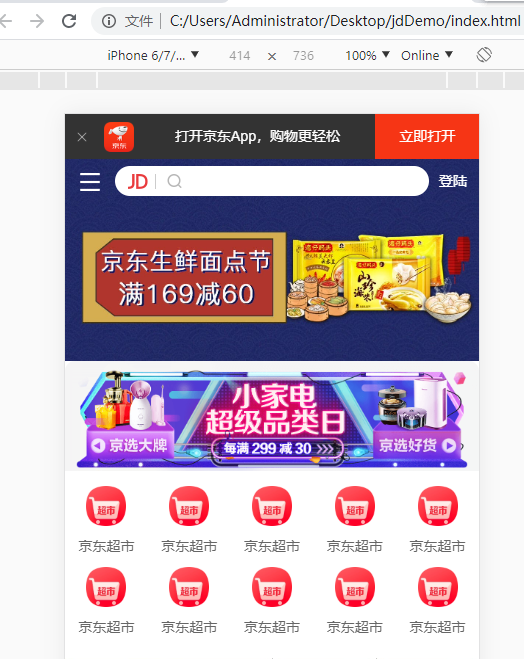


# 百分比布局

百分比布局，也叫流式布局或非固定像素布局。通过盒子的宽度设置成百分比来根据屏幕的宽度来进行伸缩，不受固定像素的限制，内容向两侧填充。通过max-width设置最大宽，通过min-width设置最小宽。

流式布局方式是移动web开发使用的比较常见的布局方式。

**利用百分比布局实现JD，目标：**



**常见的布局之固比固：**



# FLEX布局

参考博客：http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html

    flex布局：弹性布局  如果使用Flex，直接干死float。

        如果你要使用Flex布局，就不想浮动和清除浮动了。

    flex的目的：让男标签可以并排显示

## 1，四个概念

**flex布局四个概念：**

**容器：**如果给盒子加上了display:flex，那么这个盒子就是容器。

**项目：**容器的直接子元素就是项目（儿子），默认情况下，项目是水平向右排列。

**主轴：**默认主轴水平向右，项目就在主轴上进行排列，项目的排列方向就是主轴的方向。

             项目是从主轴的起点开始排列（把握住）

**交叉轴：**和主轴垂直的那个轴就是交叉轴。

    flex布局：

        关于容器的相关的属性（6个（其实就5个））

        关于项目的相关的属性（6个）

## 2，关于容器的相关的属性

**关于容器的相关的属性：  下面的6个属性一定是加给容器的**

* flex-direction：设置主轴的方向
* flex-wrap：控制是否换行
* flex-flow：是前两个是的简写
* justify-content：如果主轴上有富余空间，处理主轴上的富余空间
* align-items：如果有一根主轴，处理交叉轴上的富余空间
* align-content：如果有多根轴，处理多根主轴的富余空间

### flex-direction

**flex-direction： direction 是方向的意思   用来改变主轴的方向**

        row: 默认的方向  水平向右   起点在左  终点在右

        column：垂直向下  起点在上  终点在下

        row-reverse: 水平向右   起点在右  终点在左

        column-reverse: 垂直向上   起点在下  终点上

### (2)flex-wrap

    flex-wrap: 当项目一行装不下时，是否换行

        nowrap: 默认值  表示不管有多个项目 都进行宽度的压缩  不换行

        wrap: 当一行装不下时，自动换行

        wrap-reverse: 用的不多  从容器下面开始分布，满一行换行 了解就ok了~

### (3) flex-flow

    flex-flow：flex-direction属性和flex-wrap属性的简写形式

        flex-flow:row wrap; =  flex-direction:row; +  flex-wrap:wrap;

### (4)justify-content

    justify-content：  jutify意思是：调整使全行排满;使每行排齐;使齐行

        处理主轴上面的富余空间

        flex-start: 默认值  表示项目从主轴的开始位置排列  富余空间就是最后面

        flex-end：表示项目从后开始排列 富余空间在前面

        center：表示项目从中间开始排列  富余空间在两侧

        space-between: 富余空间在项目和项目之间

        space-around: 富余空间环绕项目

### (5) align-items

    align-items:

        项目在交叉轴上如何对齐   如何处理项目在交叉轴上的富余空间

        flex-start：表示项目在交叉轴的开始位置开始排列  富余空间在下面

        flex-end: 项目在交叉轴的终点位置开始排列  富余空间在上面

        center： 项目在交叉轴的中间位置开始排列  富余空间上下都有

        baseline: 项目的第一行文字的基线对齐

        stretch: 默认值  如果项目未设置高度或设为auto，将占满整个容器的高度(撑满整个交叉轴)

### (6) align-content

    align-content: 用来设置多根主轴的位置关系，如果就一个主轴，此属性没有效果

        有多根主轴后，如何处理它们在交叉轴上的富余空间。

        使用这个属性的前提：1）肯定要允许换行   3）有多根主轴

        flex-start： 多根主轴从上进行排列   富余空间在下面

        flex-end： 多根主轴从下进行排列   富余空间在上面

        center: 多根主轴从中间进行排列     富余空间上下都有

        space-between: 富余空间在多根主轴之间

        space-around ：富余空间环绕多根主轴

        stretch: 不说了~

## 3，关于项目的相关的属性

**关于项目的相关的属性：  下面的6个属性一定是加给项目的**

        order：定义项目的排列顺序。数值越小，排列越靠前，默认为0。

        flex-grow： grow 表示生长  如果有富余空间  可以让项目长大  了解

        flex-shrink： 压缩  在不换行情况下  一行装不下了  设置每一个项目的压缩比例  了解

        flex-basis：设置项目在主轴上占多少比例

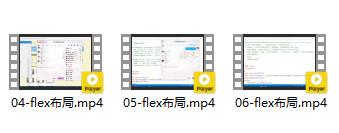
        flex：是flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto。

            flex:1, 1, auto; = flex-grow:1; + flex-shrink:1; + flex-basis：auto;

            通过会这样写：flex:1; // 把flex-shrink 和 flex-basis省略不写了   就剩flex-grow;

        align-self：处理单独一个项目它在交叉轴的富余空间

**视频看一下：**



**Flex布局实现一个页面 明天写页面**

# REM

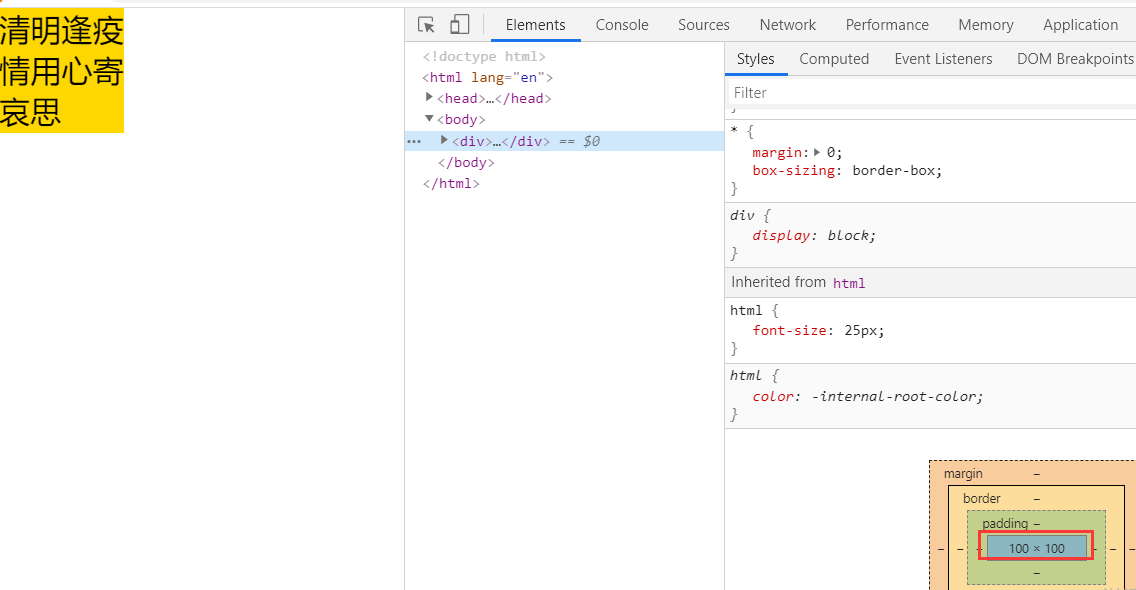
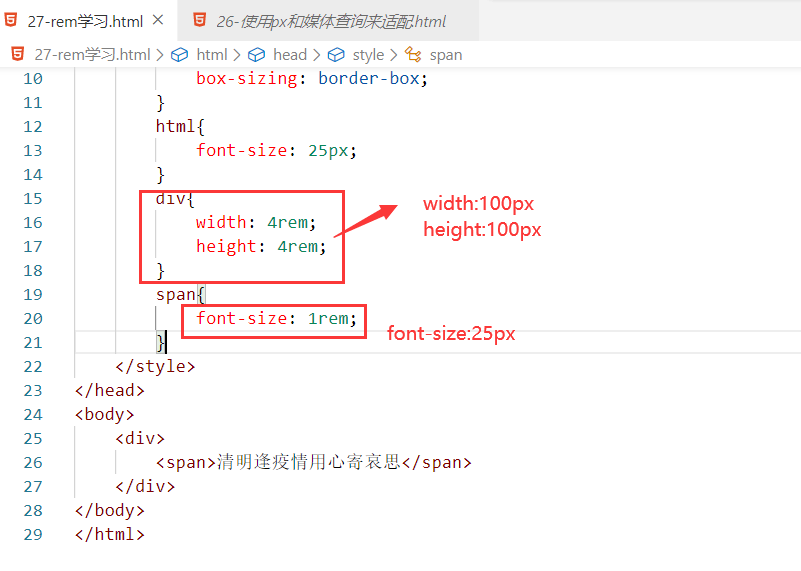
## 什么是rem

我们去找到一个标准，是一直不变的，此时我们就可以使用rem。r表示root，给html设置字体大小，后面所有的盒子都根据html的font-size这个标准来设置。

是相对于根元素的字体大小，根元素是独一无二的，就是html元素，如下：



**例子，如下：**



## 利用rem和媒体查询做适配

rem适配原理：通过缩放，大屏中盒子/文字大一点，小屏中盒子/文字小一点。保证不同屏中看到的内容是一样的。

**假如说：**我要看电子书，我买一个大屏，我的目的是**显示更多的内容**，此时使用rem布局就不行了。也就说除了rem还有一些其它的适配方案。

**基本步骤：**

1. 在写CSS时，所有尺寸，如marign,padding,top,left.... 全部使用rem来写，把之前所有的px变成rem。
2. 在不同的手机中，通过媒体查询来设置html标签的font-size，不同的手机中html的font-size是不一样的，从而达到适配目的。

**假如我们有珍上设计图，有一行文字和一张图，要求如下：**

1. 一行字在不同的设备中占满一行
2. 一张图片在不同的手机中只占一半

### 从美工手中得到设计搞，假如是750的宽。

### 严格按设计搞的尺寸，以px为单位，把页面进行还原。

有一个不技巧，调整屏幕的宽度，如下：



调整上面的目的就是，在750的手机上，完美显示，如果750上面效果OK。接着就把750的宽度推广到其它屏上。

写代码：



效果如下：



也就是说在750屏幕下，写的是OK的。

### 定义一个rem的基准

给html标签设置一个font-sie，为了好计算，我们就设置成100px，纯粹是为了好计算，也可以设置成其它的。如下：



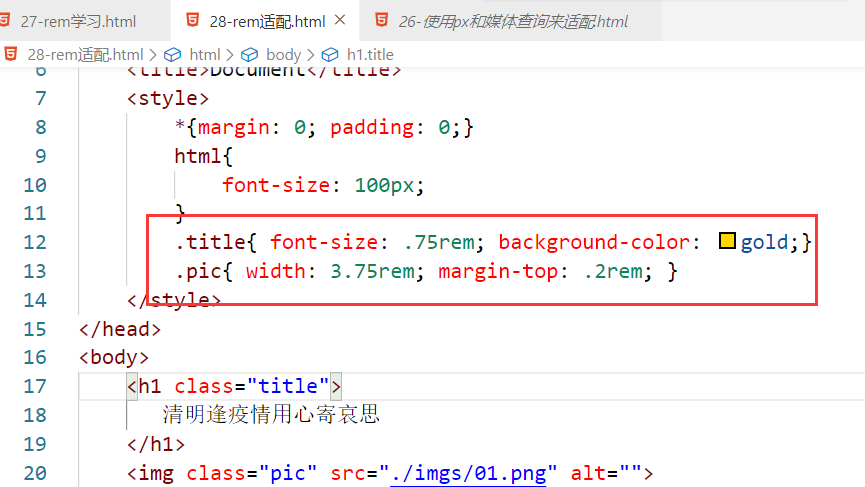
现在就意味着，1rem就是100px。

**假如我们一个盒子，**

在750中，宽是20px， 我们写rem应该写：0.2rem

在750中，高是54px，我们写rem应该：0.54rem

把之前的Px换成rem，如下：



此时，在750下面，没有任意的影响，如下：



### (4)通过媒体查询，把750下面的写好的迁移到其它的屏上

在宽为750屏， html中的font-size是100px。

换算了：

在宽为375屏，html中的font-size是50px。

在宽为320屏，html中的font-size是320\*100/750px。

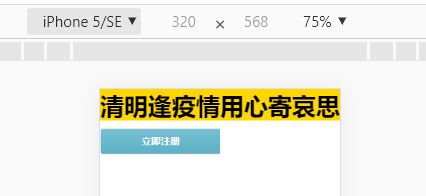
在宽为414屏，html中的font-size是414\*100/750px。

...

**写代码如下：**



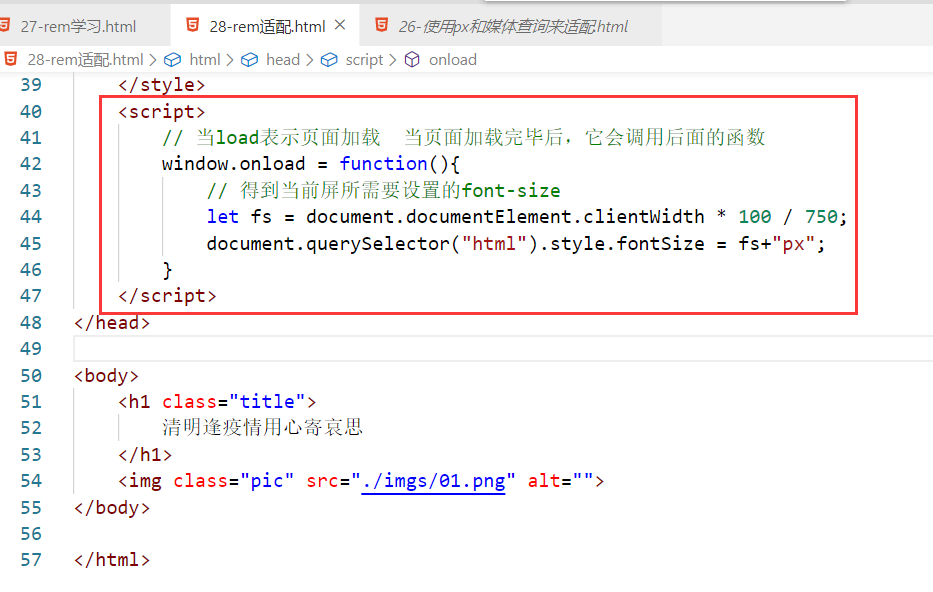
测试是否OK，如下：



上面虽然实现了，但是不完美，手机的尺寸各种各样，我们不可能市面上所有的机型都写一条媒体查询，可以通过JS代码获取不同的屏的尺寸。

### (5)通过JS来迁移到其它屏

**JS代码如下：**



测试OK~

# 媒体查询

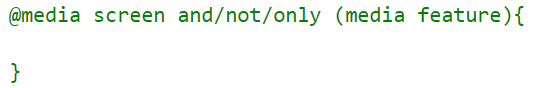
参考：<https://www.cnblogs.com/nyw1983/p/11409219.html>

是css3中提出的新语法。@media来实现媒体查询。根据不同的设备设计不同的样式。

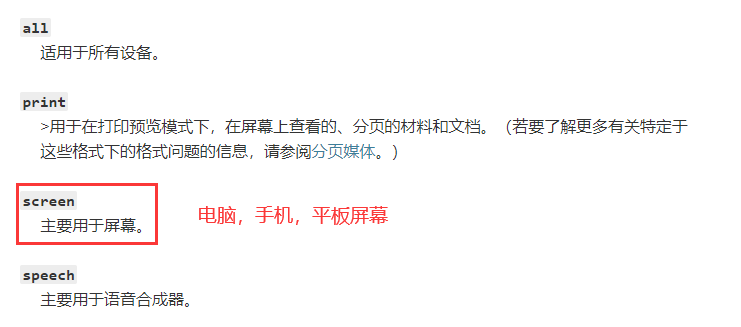
**语法：**

* 使用@media打头 @符号不能少
* mediatype 媒体类型
* 关键字 and not only
* 媒体特性，需要使用()包起来

**语法如下：**



## 媒体类型



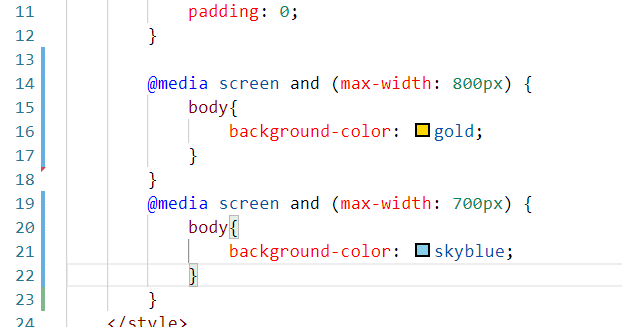
## 关键字

* and: 将多个媒体特性连接在一起，类似 且 的意思
* not: 排队某个媒体类型，相当于 非 的意思 可以不写
* only: 指定某个物定的媒体类型，可以不写

## 媒体特性

* widht 页面可见区域的宽度
* min-width 页面最小可见的宽度
* max-widht 页面最大可见的宽度

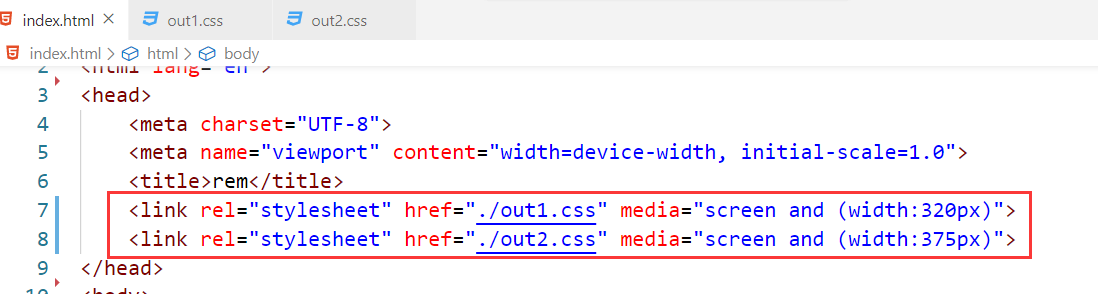
在没不同的屏幕中body颜色要不一样，代码如下：



max-width表示小于等于

也是有层叠性的

## 根据不同的设备载入不同的外部样式



# 九，less

**less目标：**让你更加优雅写CSS，浏览器不认识less，最终还需要编译成css，浏览器只认css。

## CSS不足

* CSS代码只能串起来起，大量使用了后代选择器，写起来不方便
* CSS不能计算 px---->rem
* CSS本身没有编程语言特点

## Less是什么

vue, react它们里面，通常也是不会直接写原生的css，都是使用预处理器，如下：

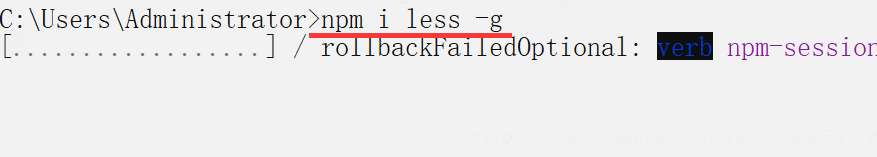
* less
* sass
* stylus



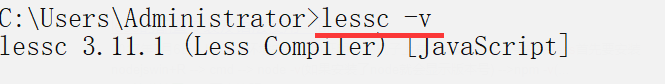
**大白话：**它就是一门新的语言，让你更加优雅地写CSS代码。

**先把node安装了。**

**安装Less:**



**检测是否安装成功：**

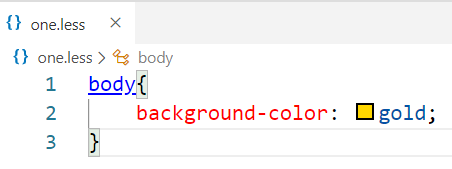


## less的编译

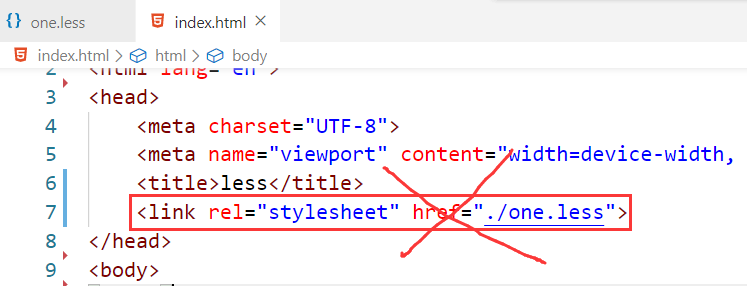
**创建一个less的文件，如下：**



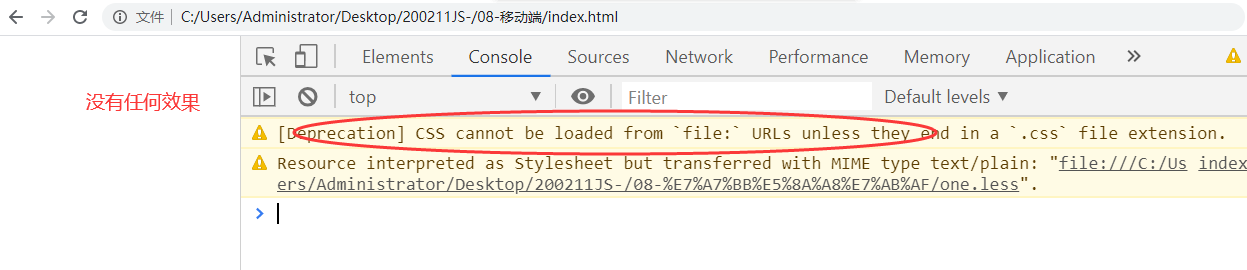
**在less文件中，就可以写css代码，如下：**



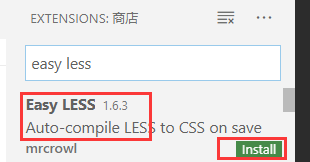
**不能在html文件中直接引入less，因为浏览器是不认识的，如下：**



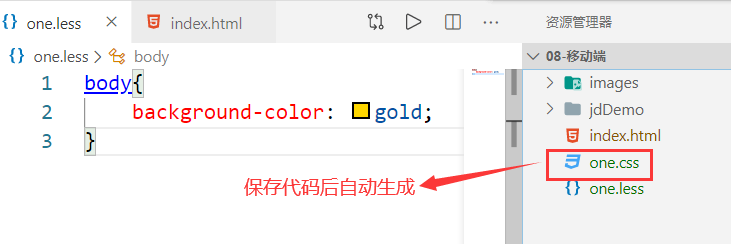
**效果：**



**此时我们需要把less编译成css，使用vscode的一个插件就OK了，插件叫easy less，如下：**



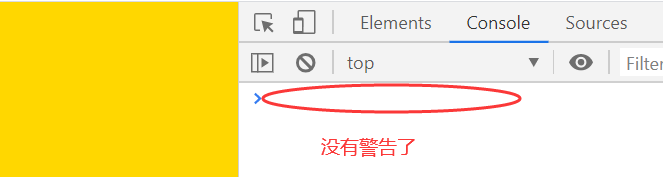
**安装完这个插件，当我们写less代码时，保存代码后，它会自动帮我们生成一个css文件，我们在html文件中引入就OK了。如下：**



**在html文件中引入css文件，如下：**



**效果：**



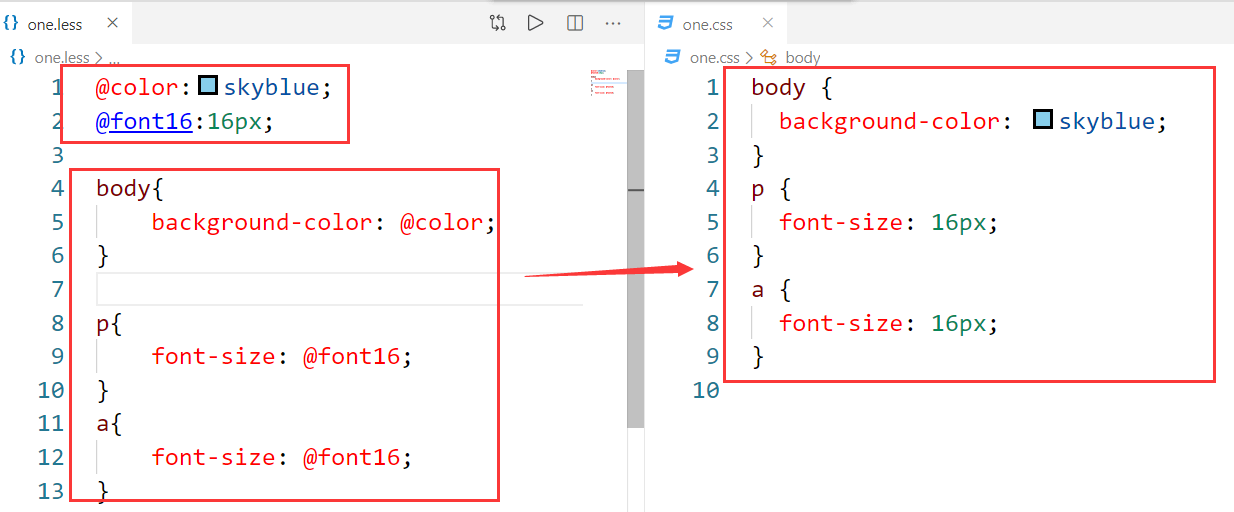
## less的变量

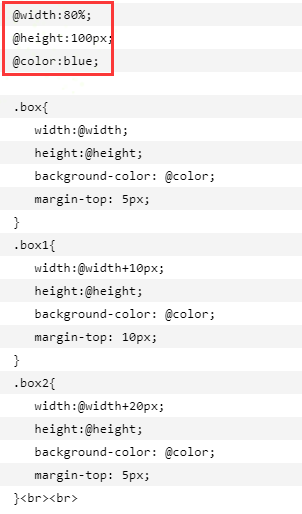
格式：@变量名:值

**要求：**

* 必须@打头
* 不能包含特殊字符
* 不能以数字打头
* 大小写敏感

**代码：**

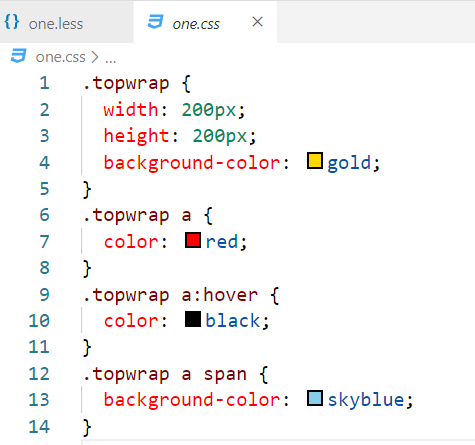




## 5，less中的嵌套

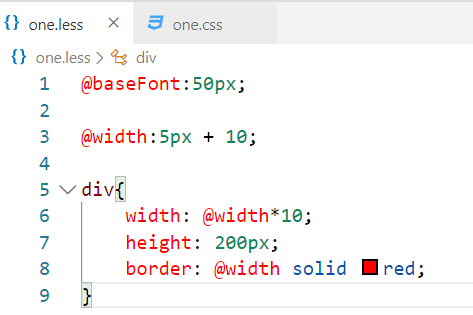


**最终也会编译成后之前写的css，如下：**



## less中的计算

算术运算+、-、\*、/可以对任何数字、颜色或变量进行运算。

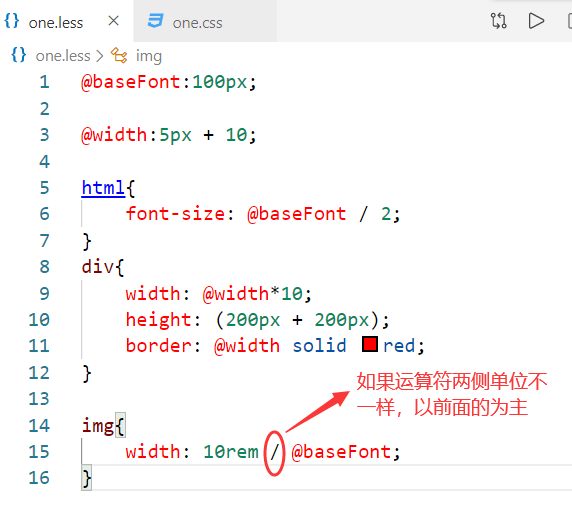


**注意点：**1）最好在运算符两侧加上空格

2）运算符两侧可能单位是相同的，也可能只有一个操作数，有单位，结果如下：



1. 还有一种情况，两侧都有单位，但是单位是不一样的，如下：

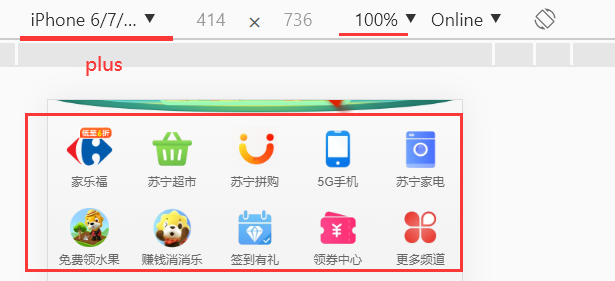
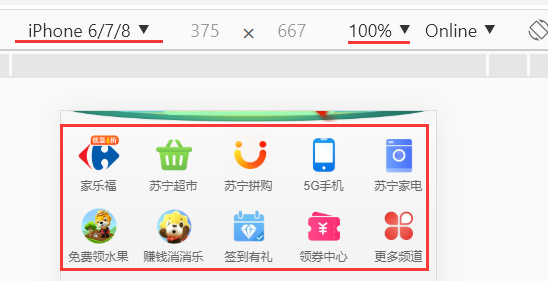
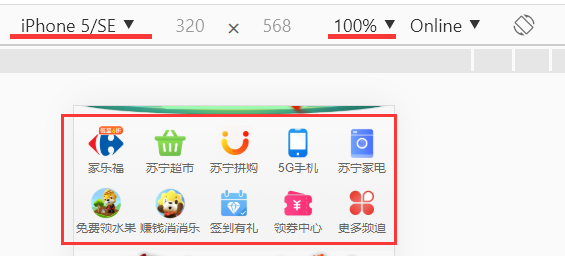


# 十，rem的适配方案

## 1，rem适配的本质

**rem适配的本质：大屏上面的显示大盒子，小屏上面显示小盒子。**

看苏宁易购效果图，如下：





在某些情景下面，是不能使用rem适配，如在大屏中显示更多的内容。

## 2，常用的适配方案

**两种：**

* Rem + 媒体查询 + less
* flexible.js + rem

### rem + 媒体查询 + less

**第一步：**得到设计稿尺寸750px

**第二步：**定义html标签的font-size的值（把屏幕分成N份） N = **15**/10/20...

**第三步：**就有了html中的font-size的值，50px 750/15 = **50**

**第四步：**适配时，把任何屏都分成15份，在320设备上，320/15 = **21.3333**

**这样就得到了，不同屏上的font-size了。**

**结论：**

1. **不同屏的font-size = 屏幕宽度 / 划分的份数**
2. **元素的rem值 = 设计图上的px / font-size**

750 分成7.5份 一份就是100

在750屏上：

html fontsize:100px 1rem = 100px

一个盒子量出它的高度是100px 宽度是100px

div{

width:1rem

}

在375屏上 分成7.5份 一份就是50 ：

html fontsize:50px 1rem=50px

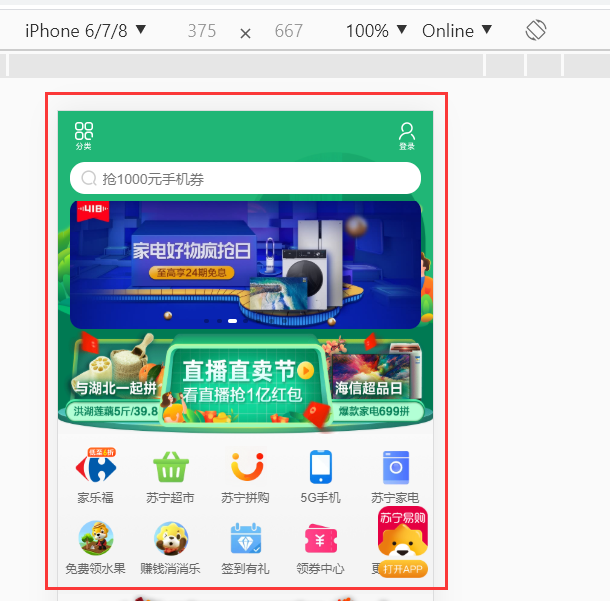
一个盒子量出它的高度是100px 宽度是100px

div{

width:2rem

}

**目标：**



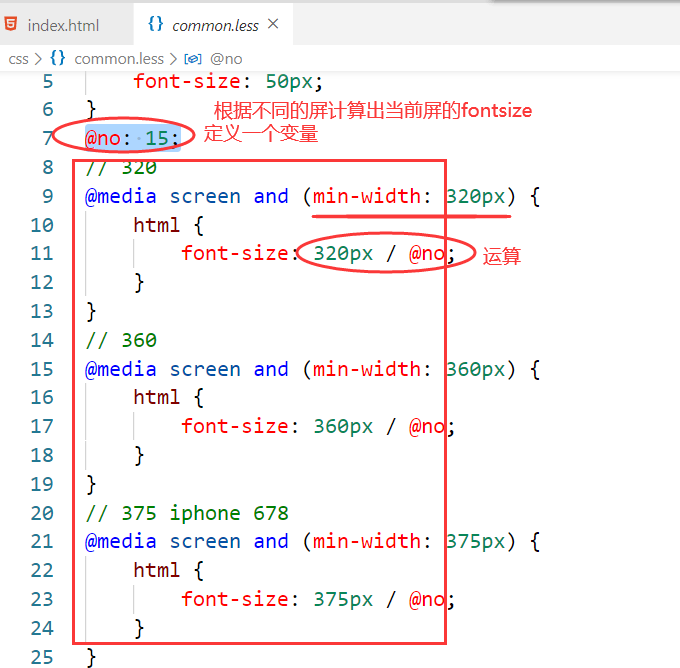
**结论：**

1. **不同屏的font-size = 屏幕宽度 / 划分的份数**
2. **元素的rem值 = 设计图上的px / font-size**

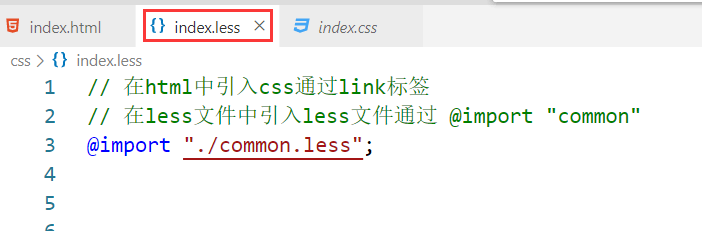
**现在开始写页面了，设计图是750px。 分15份，一份是50px**

* 在750屏中：html的fs:50px
* 在375屏中：html的fs:25px
* 在320屏中：html的fs:21.33333px
* 在414屏中：html的fs:27.6px
* 在500屏中：html的fs:33.3333px
* ....

在不使用js情况下，定义一个common.js文件，列出市面常用的手机屏的宽度，并计算出这个屏它的FontSize，如下：



开始写样式，定义一个index.less，表示首页面的样式，引入上面的common.less文件，，如下：

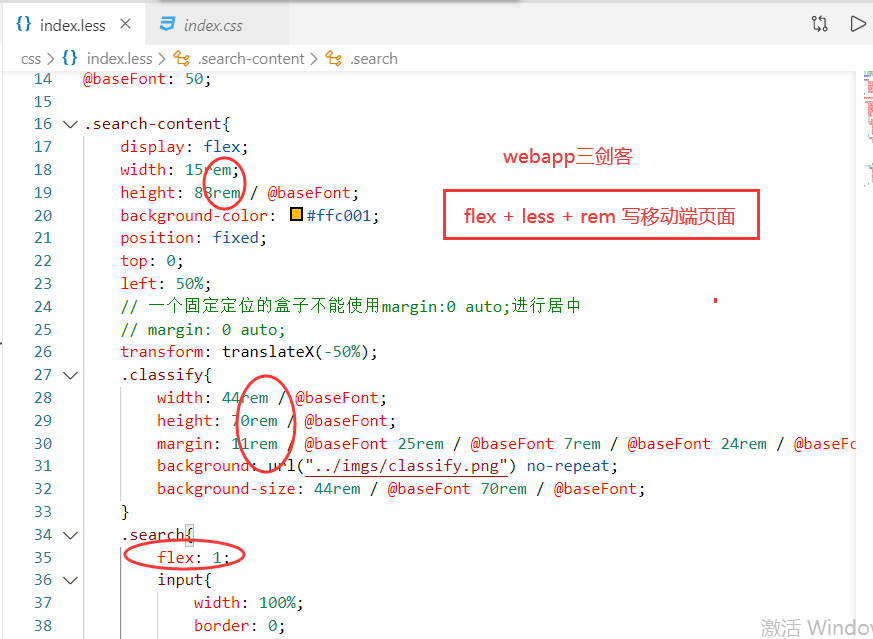


在index.html中引入index.css，如下：



接着在index.less中写样式就OK了，如下：





后面的代码就不截图了，直接看代码就OK

### flexible的使用

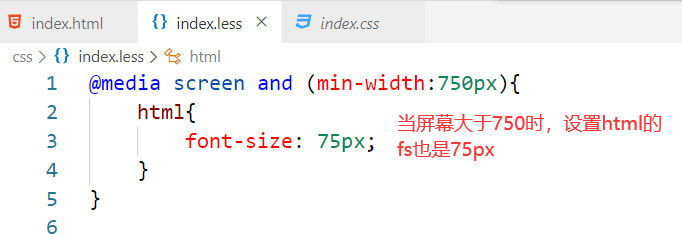
**手机淘宝团队出的解决方案。**

参考：<https://www.jb51.net/article/158894.htm>

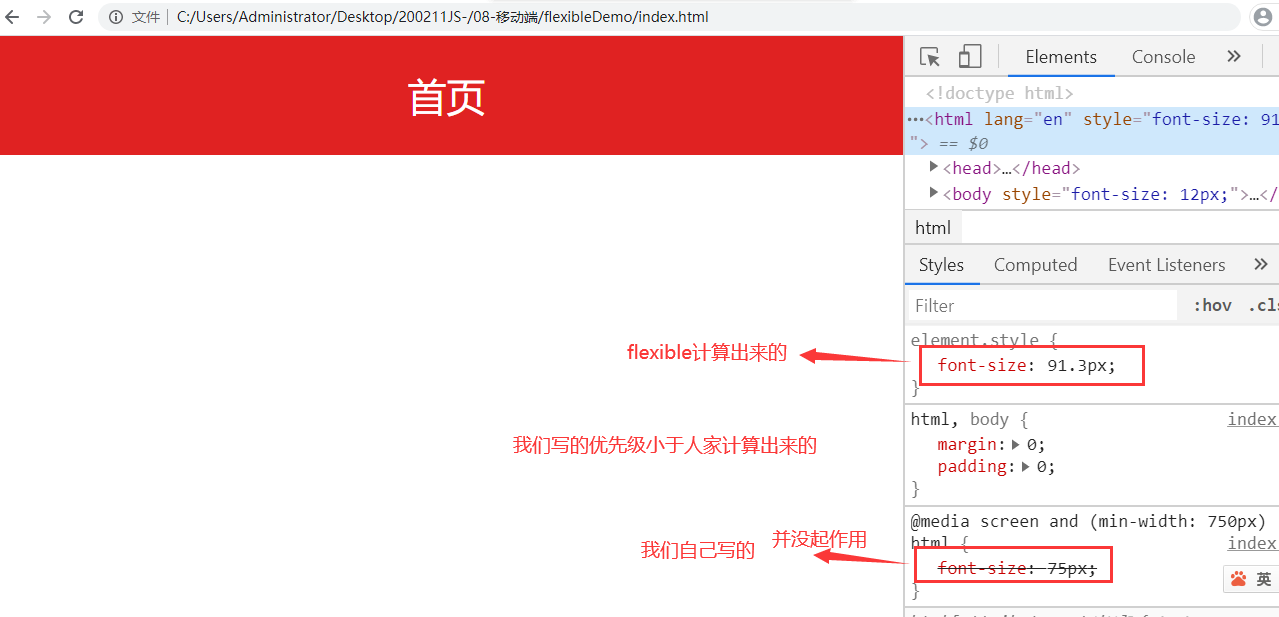
**它可以帮我解决如下事情：**

1. 不需要再写一块的媒体查询，库里帮处理好了
2. 它默认是划分成了10份 html的fs:750/10 = **75px**
3. 量一个盒子高度88px div{ height:88rem/75 }

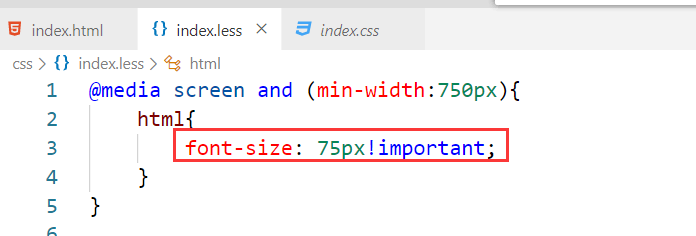
**需要注意：当屏幕大于750时，我们需要自己写媒体查询处理一下，代码如下：**



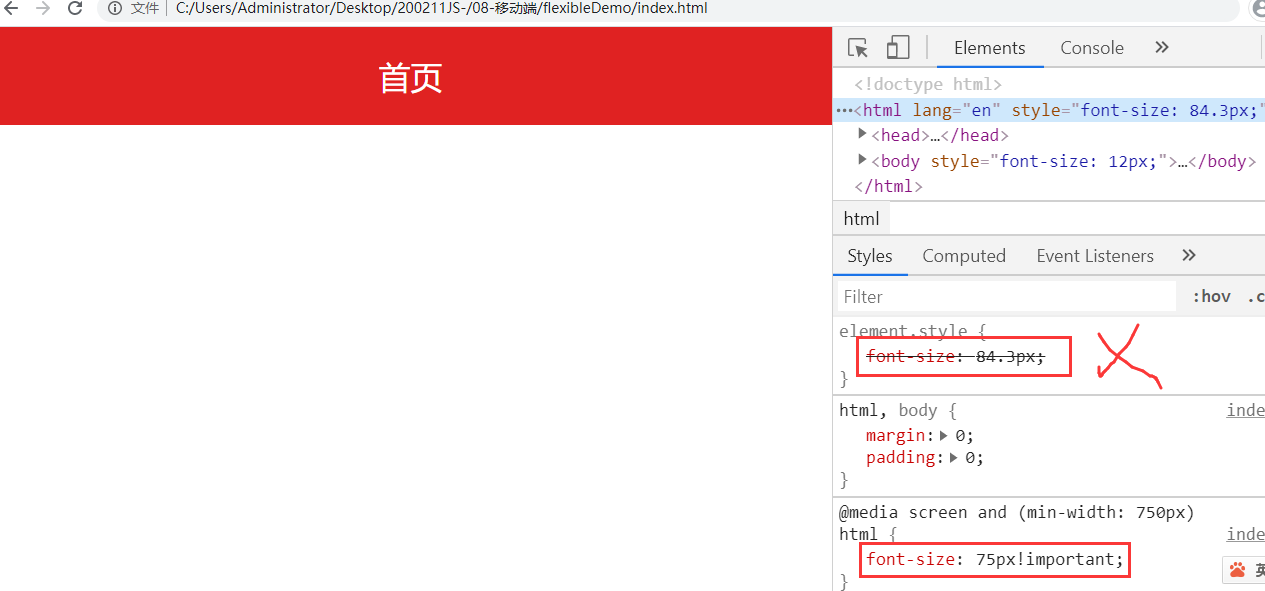
效果如下：



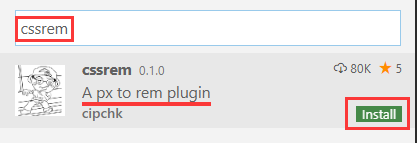
提高优先级，如下：



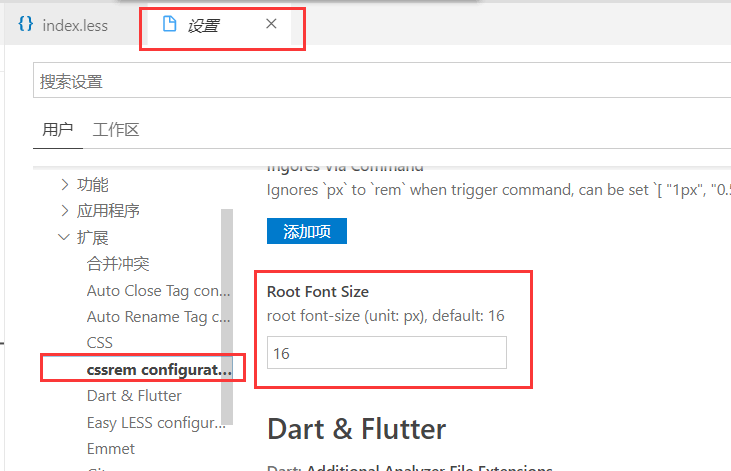
效果如下：



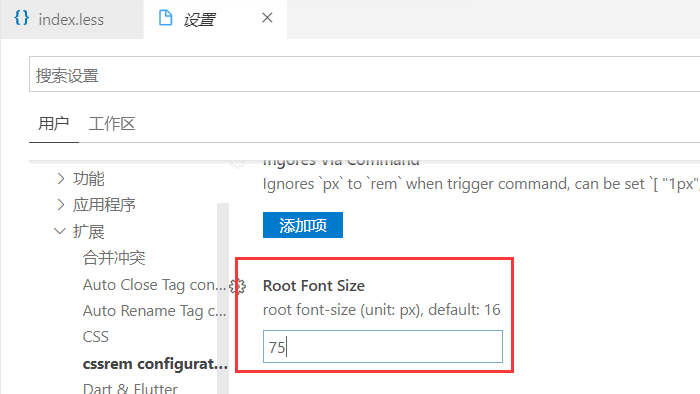
方案一还有一个恶心的地方，写样式时总要除一个基准，这个可以使用vscode插件来解决，插件叫**cssrem**。安装如下：



**cssrem默认基准是16px，如下：**

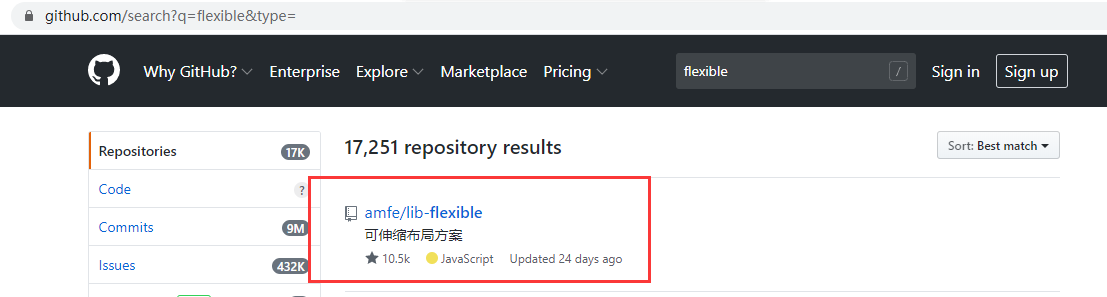


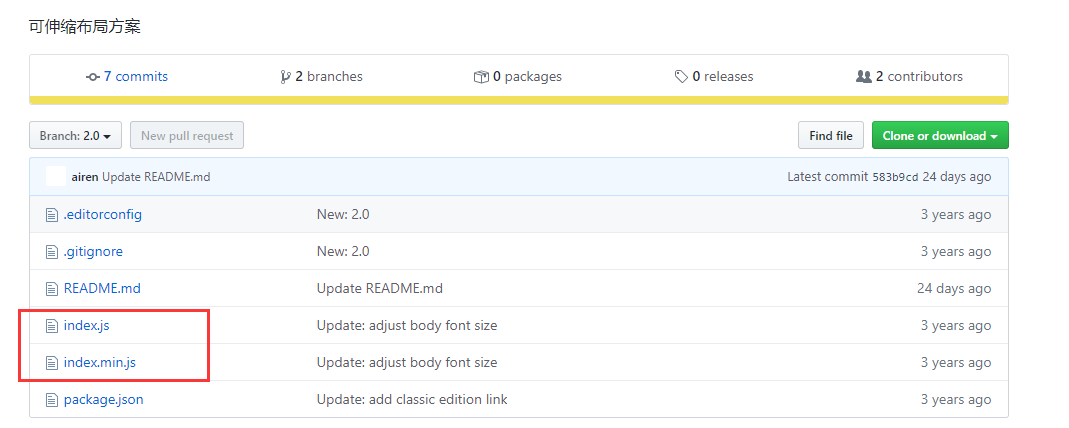
你的基准是多少，就修改为多少，我们使用flexible插件的基准是75，所以修改如下：



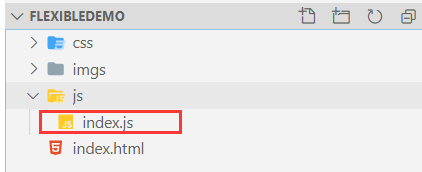
**使用flexible的步骤：**

第一步：下载flexible这个库

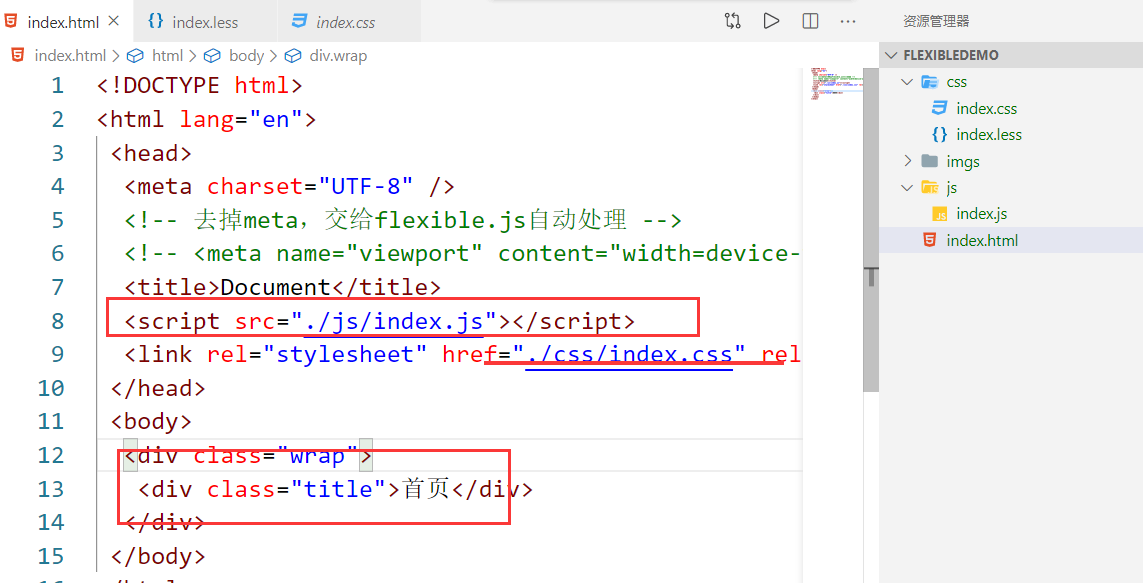




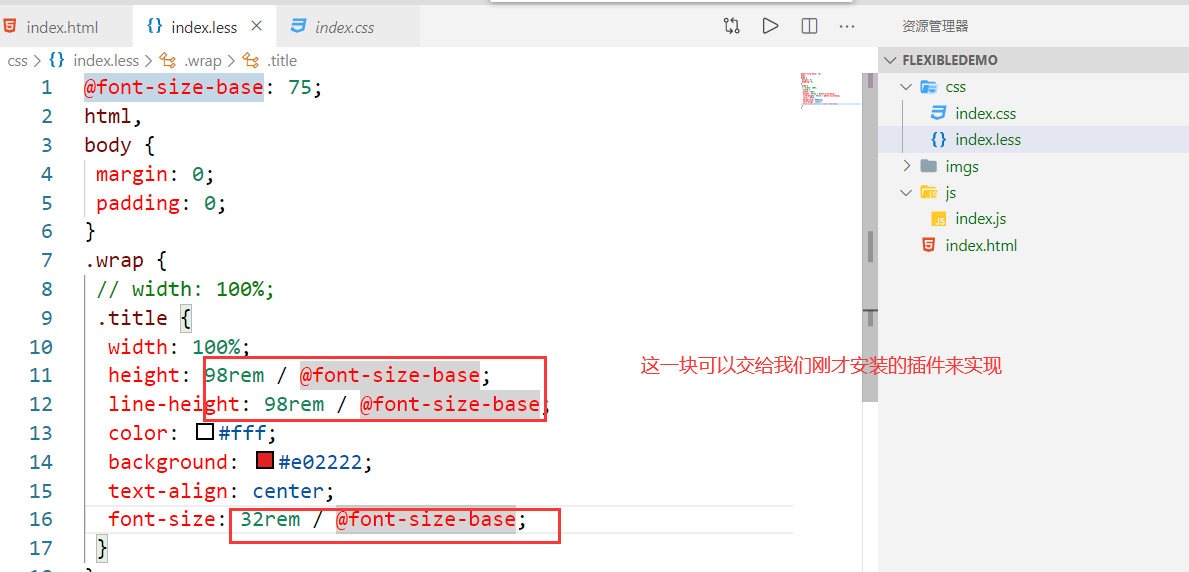
**放到项目中：**



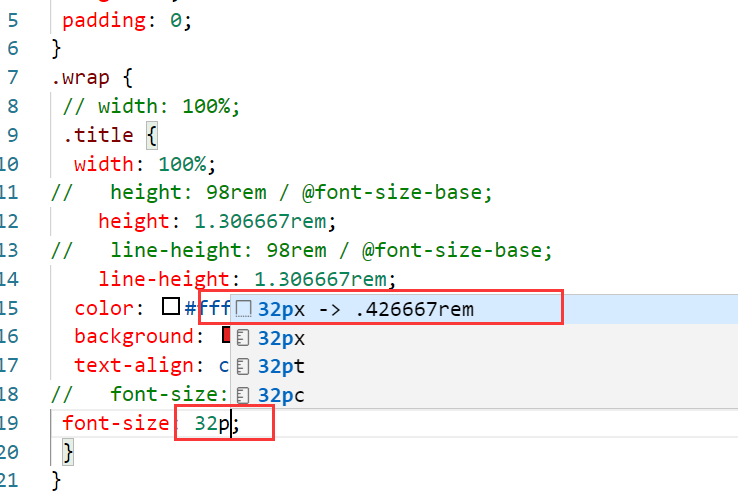
**第二步：写代码**

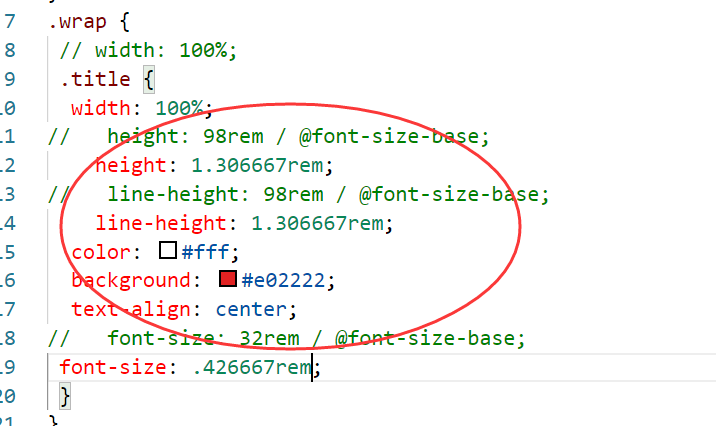


写Less代码，如下：

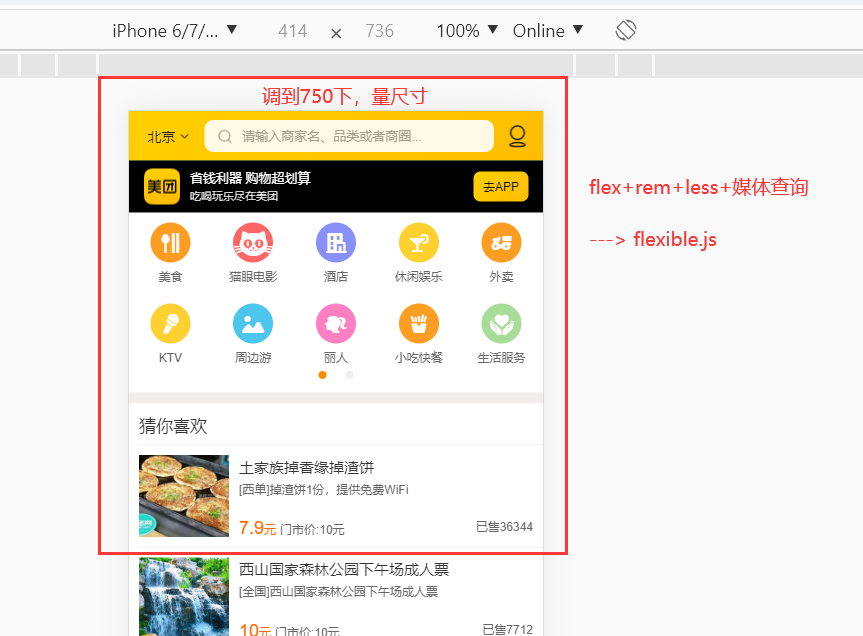


使用我们插件，如下：





**作业：**



# 其它的适配方案



还有一种叫vw适配方案，我是弥补rem的不足。