# 移动端理论

## 为什么学习移动web开发.avi

统计数据显示：

1. 成年人每天会花4小时左右来玩手机，移动端才是未来,得移动者得天下
2. 京东百分之七十的营业额来自于移动端，只有30%的营业额来自PC端。

## 基础知识-像素密度.avi

密度：同样体积的铁和棉花，铁更重一些，因为铁的密度比棉花要大

像素密度：同样尺寸的手机，有的手机更清晰一些，因为像素密度更大

### 引言





我们将从屏幕尺寸、分辨率、PPI的角度来思考一下，为什么小米要比红米贵那么多

FHD+的含义是：全高清屏

### 屏幕尺寸

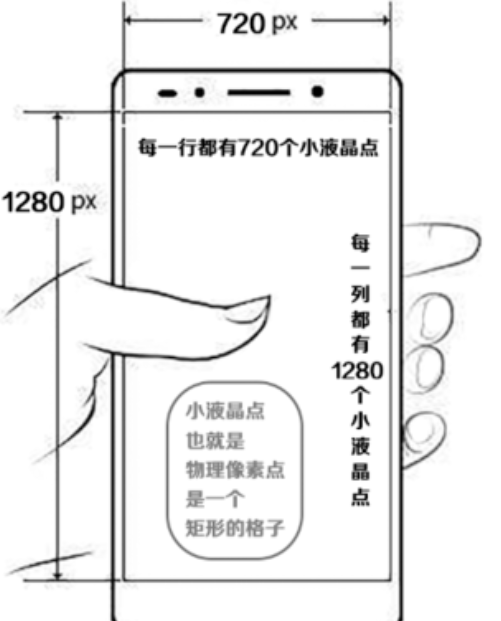


1英寸（inch）=2.54厘米（cm）

### 像素





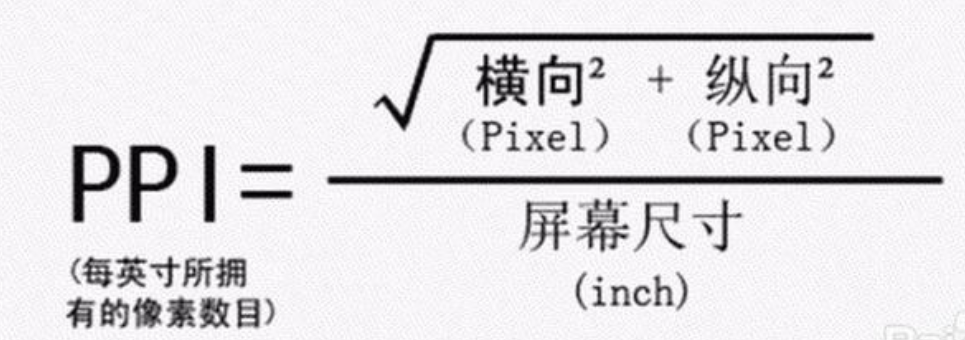


我们可以把像素理解为画图时候的笔刷的大小

### PPI

同样一片画图区域

PPI = 屏幕像素密度，即在一个对角线长度为1英寸的正方形内所拥有的像素数





PPI越小，一个像素越大，同样大小的面积的像素的数量越少

PPI越大，一个像素越小，同样大小的面积的像素的数量越多，画质越清晰越细腻

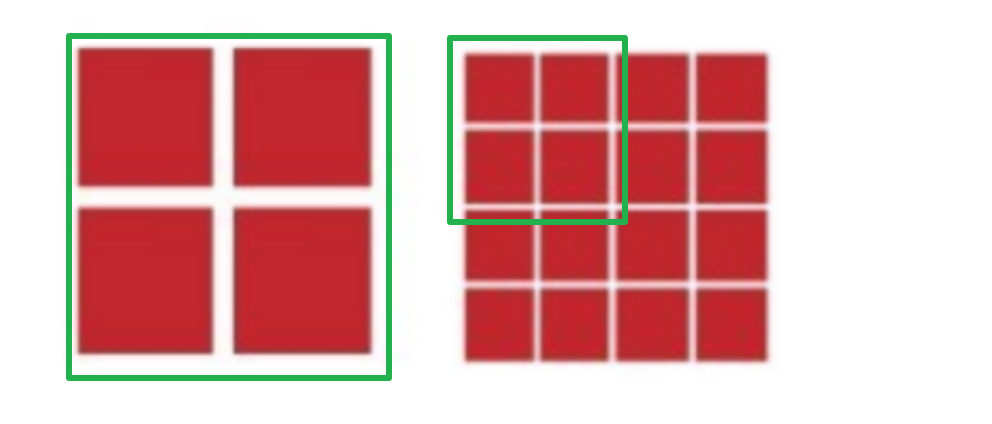


## 基础知识-设备独立像素.avi

设备独立像素是为了便于开发人员用代码绘制界面的时候方便而人为规定出来的一个东西。

1立方米的水和1立方米的空气，大小是一样的，但是里面实际的物质含量是不一样的

我们了解了像素的知识后，就引出了一个问题，布局的时候我们写的是10px\*10px,在不同的PPI值的手机上面，低PPI的可以占满全屏幕，而高PPI的不能占满全屏幕。



PPI越高，像素的数量越高，这里面，其实有一个比例的关系，比如A手机的像素刚好是B手机的2倍，我们平时布局更关注的是这里的倍数的关系，我们会用dpr来表示

dpr是用来连接现实物理像素和虚拟逻辑像素的桥梁

在iphone 3下面,dpr = 1，意味着1pt = 1个像素（2pt\*2pt相当于2物理像素\*2物理像素填充满了）

在iphone 4下面，dpr = 2,意味着1pt = 2个像素（2pt\*2pt相当于4物理像素\*4物理像素填充满了）

pt（dp）就是设备独立像素（逻辑像素）

一台手机在出厂的时候，像素宽度值（物理像素）确定下来了，PPI也确定下来了，dpr的值也确定下来了，不可人为进行改变，其值大小决定了屏幕渲染图像的品质，我们前面讨论的都是物理像素。PPI越高，物理像素越小，PPI越低，物理像素越大。

屏幕的清晰程度其实是分辨率和尺寸大小共同决定，用ppi或者dpr来衡量屏幕清晰程度更加准确。因为红米手机的PPI值比不上小米手机，所以要更加的便宜。

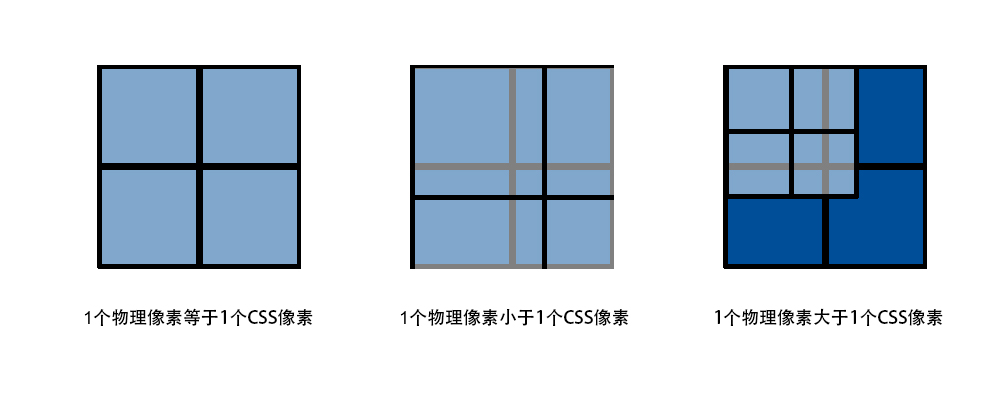
## 基础知识-css像素.avi

<script>  
*// 以像素计，屏幕的大小*var ***screenWidth*** = **window**.**screen**.**width**;  
var ***screenHeight*** = **window**.**screen**.**height**;  
  
**console**.log('屏幕的宽度为: ' + ***screenWidth***);  
**console**.log('屏幕的高度为: ' + ***screenHeight***);  
</script>

接下来，我们再来理解一个概念叫css像素，也就是我们平时切图写页面的时候的像素单位

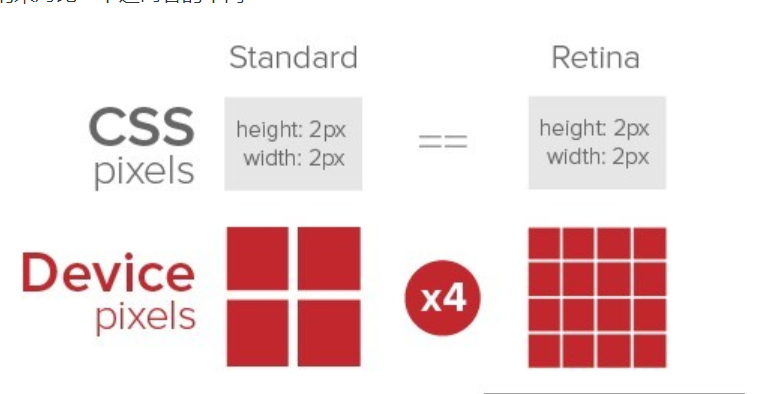
我们上面讲过了设备独立像素是pt,dp,但这个单位我们平时用不上，我们平时用的是px，其实我们平时写的css像素就是我们前端h5这块的所谓的设备独立像素（逻辑像素）。

PC端：1个单位的CSS像素等于1个单位的物理像素



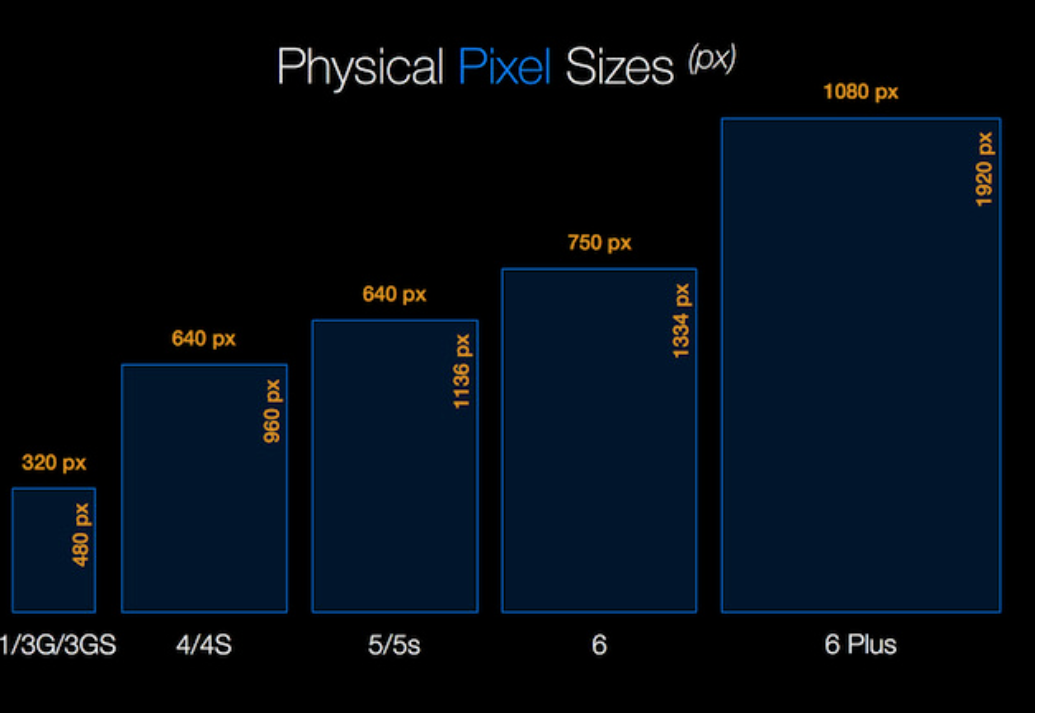
移动端dpr的值为1的话，则1个CSS像素等于1个物理像素

移动端dpr的值为2的话，则1个CSS像素等于2个物理像素



也就是说，在dpr为2的情况下，你写的1px，需要2个物理像素去渲染

理解：



上面显示的是每种苹果手机的物理像素

我们打开控制台看到的是css像素+dpr的值



如何理解：

320 --> css像素、设备独立像素

dpr 2 1个css像素实际包含2物理像素

## 基础知识-2倍图.avi

假如你有一个张图片，我们在PS里设计的时候物理像素只有1像素，则在dpr为2的手机上面，如果你写的是1px的话（这里的1px是指css像素），也就是设置这个图片的宽度是2个物理像素去渲染，则会拉伸模糊。

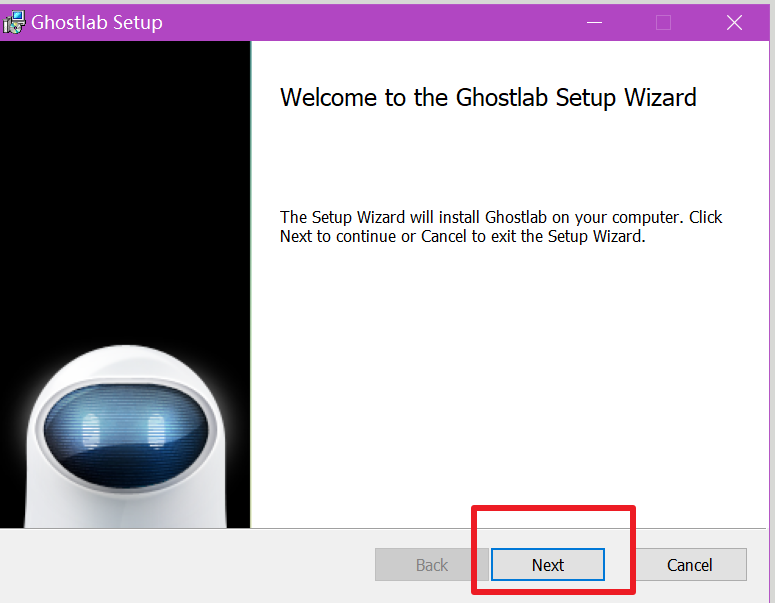
解决办法：在PS设计的时候，就针对于dpr=2的设计图片的宽高是原来的2倍，这样才能保证物理像素一对一，不会出现拉伸模糊。

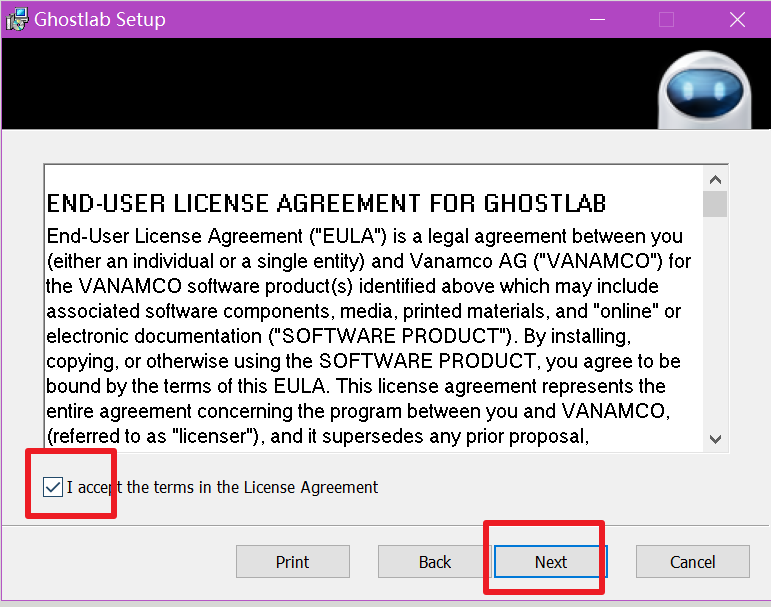
1. 图片的像素是构成图片的像素块
2. 我们可以对图片进行拉伸或者缩放
3. 拉伸或缩小并不会影响它的物理像素的个数
4. 比如一个图片的物理像素是10\*10，如果我们写的是10px\*10px，如果dpr的值是2，则相当于我们设置这个图片在占满20\*20物理像素的大小，这样就会出现拉伸
5. 为了解决这个问题：我们在dpr为2的情况下，就需要物理像素也是20\*20的，这样你写的虽然是10px\*10px,但是实际上就保证了一个图片的像素点对应一个屏幕的像素点，这样刚好不会有拉伸的情况，就能保证不会模糊
6. 有的同学说，既然这样，那干脆让图片越大越好，这样不是也可以保证吗？这是对的，但是我们要知道，图片越大，像素越多，页面的数据越多，用户上网会浪费流量，并且会很卡顿

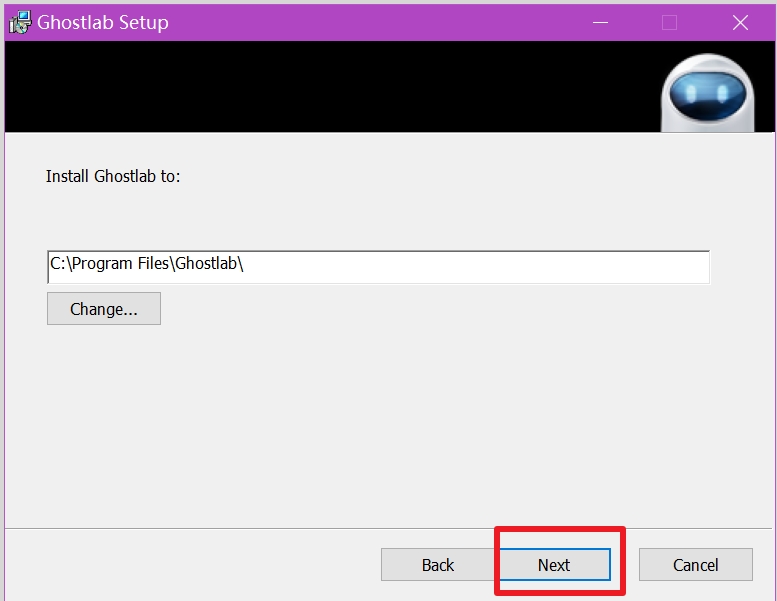
## 调试.avi

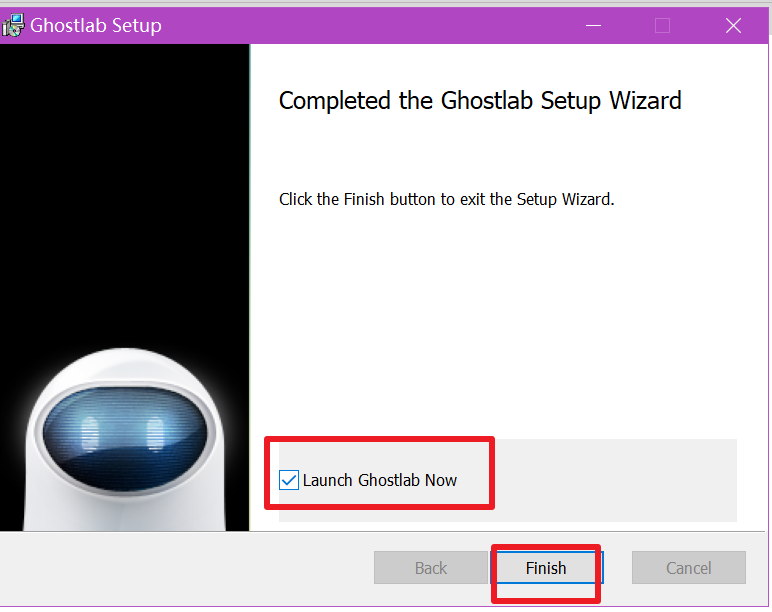
itools大家不用安装，我这边给大家共享我的屏幕用的，对我们写代码没啥用

安装ghostlab







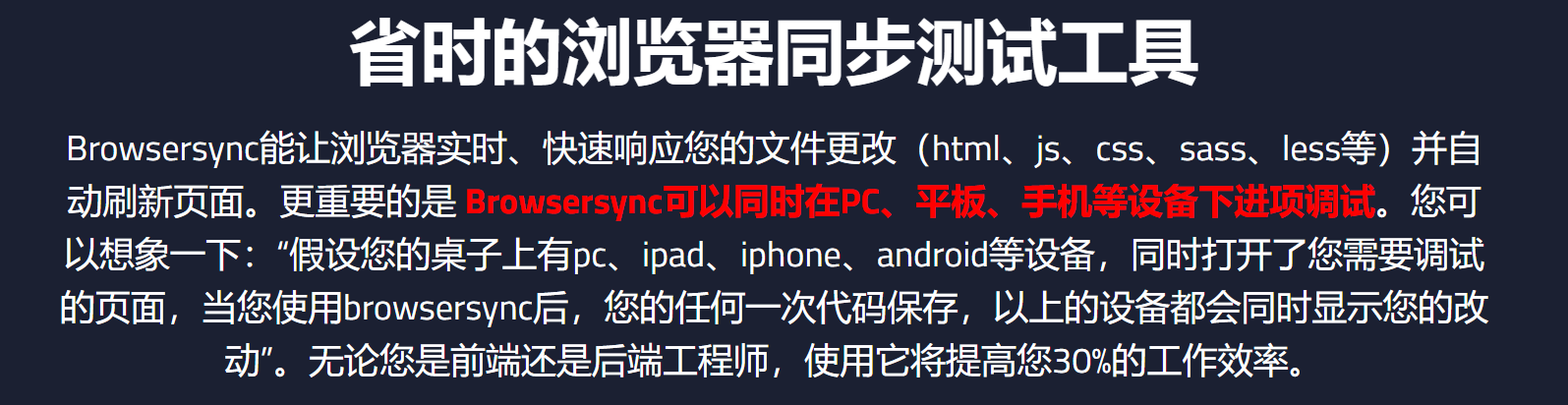


**真机调试必须保证移动设备同服务器间的网络是相通的。**

ghostlab这个工具虽然是付费的，但是没关系，我们 后面在学习了node.js之后，

我们会拥有类如live server, browsersync之类的工具都是免费的

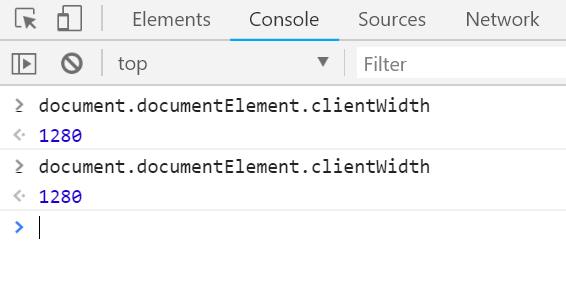
<http://www.browsersync.cn>



我们大部分学习和工作的阶段，我们还是以chrome的模拟工具为主进行开发

## PC端的视口

viewport：在PC设备上viewport的大小取决于浏览器窗口的大小



## 移动端的视口

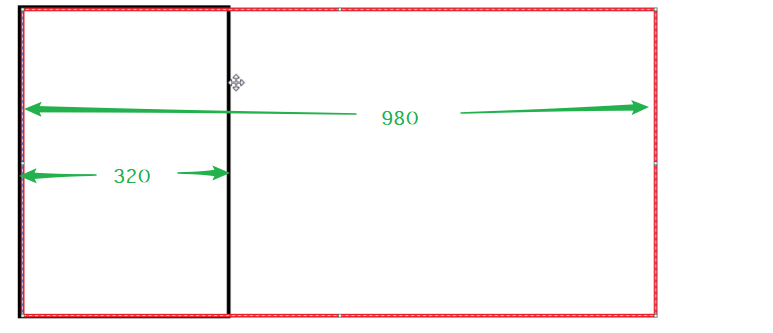


320指的是设备视口宽度，也是理想视口宽度

980指的布局视口宽度

如何理解：

1. 如果把移动设备的浏览器(也有可能是app中的webview) ,当做相框的话
2. viewport就相当于相框中的画,可能会比相框小,可能会比相框大,如果刚好一样大,那就皆大欢喜.



1. PC端的页面在移动端都有一个默认的viewport值

2.当盒子内容宽度等于viewport就显示一排，超出了就会换行

3. PC端的页面在移动端上显示文字被缩放了

如何检测宽度是980?

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <style>  \*{  margin: 0;  padding: 0;  }  .ceshi{  width:981px;  height:100px;  background: blue;  }  </style>  </head>  <body>  <div class="ceshi"></div>  </body>  </html> |

如何看出来是缩放了？

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <style>  \*{  margin:0;  padding:0;  }  .ceshi{  width:375px;  height:980px;  background: #000;  }  </style>  </head>  <body>  <div class="ceshi"></div>  </body>  </html> |

## 移动端页面正常展示的解决方案分析

让视口刚好等于屏幕的宽度，并且禁止掉放大缩小

这块可以直接用模拟器进行检测

1、布局视口和屏幕视口的概念

layout viewport(布局视口) :承载页面布局的那块区域

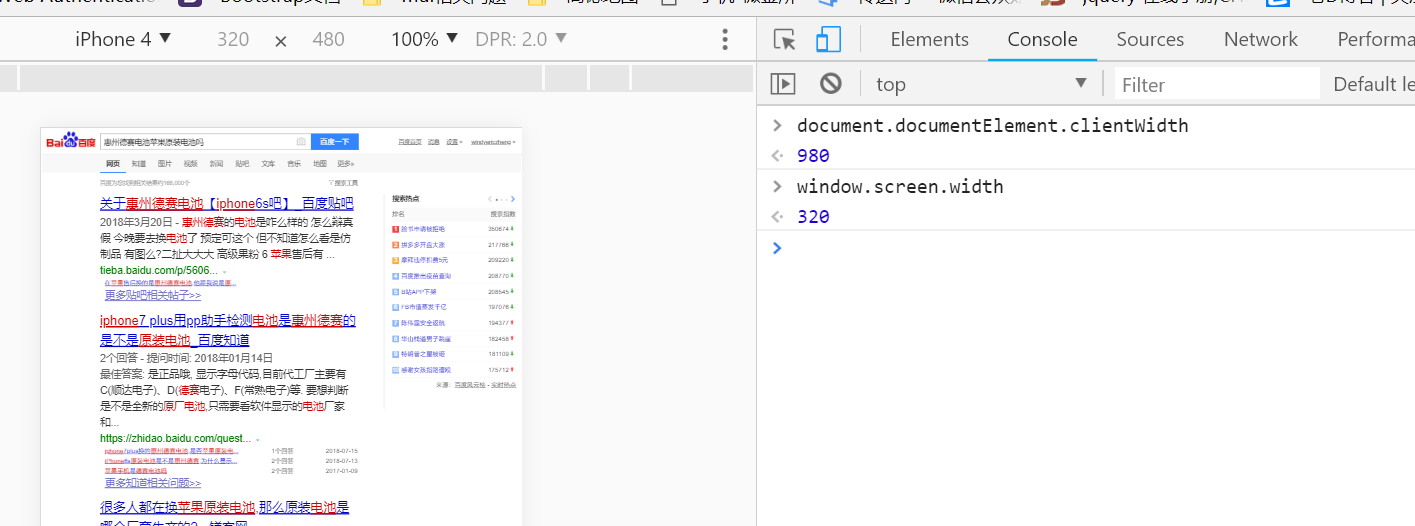
ideal viewport(屏幕视口) : 即设备窗口的尺寸

2、页面能正常的显示的原理

原理：页面能正常的显示是通过了一个缩放

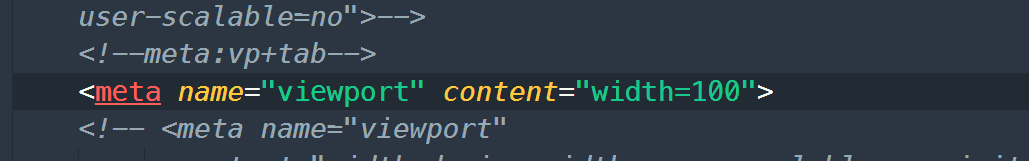
将layout viewport（布局视口）进行了缩放，使其恰好完整显示在ideal viewport（屏幕）里，其缩放比例为ideal viewport / layout viewport





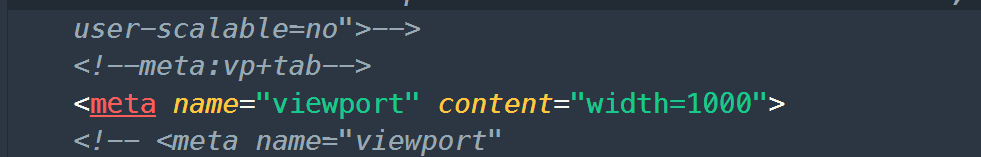
## 移动端viewport的设置(width)

极端一点测试



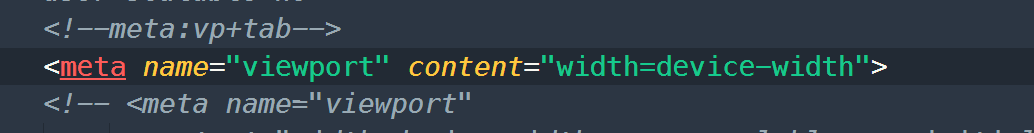
如何理解：

1. layout viewport是100
2. 缩放到与ideal viewport一样宽，所以字会变大



如何理解：

1. layout viewport是1000
2. 缩放到与ideal viewport一样宽，所以字会变小



1. 为什么要设置viewport

为了让页面能正常的显示在不同的设备上

1. 如何去设置viewport

通过meta标签来设置viewport的属性

<meta name="viewport" content="width=device-width">

1. viewport的其他属性
   1. name=”viewport” 说明meta是来设置viewport属性的
   2. width : viewport的宽度
   3. device-width:设备的宽度

## 移动端viewport的设置(initial-scale)

1. 设置<meta name="viewport" content="initial-scale=1">，这时我们发现网页没有被浏览器设置缩放。
2. 设置<meta name="viewport" content="width=device-width">，这时我们发现网页也没有被浏览器设置缩放。

**总结：**两种方式都可以控制缩放，开发中一般同时设置width=device-width和initial-scale=1.0（为了解决一些兼容问题）

即<meta name="viewport" content="width=device-width, initial-scale=1.0">

**提示：**initital-scale设置页面的初始缩放值，为一个数字，可以带小数。

## 移动端viewport的设置(其它属性)

1. **minimum-scale:**

允许用户的最小缩放值，为一个数字，可以带小数

1. **maximum-scale:**

允许用户的最大缩放值，为一个数字，可以带小数

1. **user-scalable:**

是否允许用户进行缩放，值为"no"或"yes", no 代表不允许，yes代表允许

标准的国际通用写法：

<metaname="viewport" content="width=device-width, initial-scale=1.0，user-scalable=0">

总结成一句话：webstorm的快捷键是:meta:vp按tab键就行了

## 移动站点和浏览器的说明

1. **移动端主要可以分为哪几类？**
   1. **系统内置浏览器**
      1. **safari**
   2. **第三方浏览器**
      1. **QQ手机浏览器**
      2. **UC手机浏览器**
   3. **应用内置浏览器(webview)**
      1. **微信里面别人发你一个链接，你点击可以打开，微信里面内置了一个浏览器**
         1. **这块大家应该体会得最深**

# 京东首页布局

## 京东首页公共样式

1、设置base样式

1. 公共样式
2. 样式重置 (包括清除一些默认的样式)

添加新的样式

/\*公共样式\*/  
/\*1.样式重置\*/  
**html**,**body**,**ul**,**li**,**img**,**a**,**p**,**div**{  
**padding**:0;  
**margin**:0;  
/\*改变盒子模型\*/  
**-webkit-box-sizing**: **border-box**;  
**-moz-box-sizing**: **border-box**;  
**box-sizing**: **border-box**;  
/\*防止a标签之类的点击高亮\*/  
**-webkit-tap-highlight-color**: **transparent**;  
}  
**body**{  
**font-family**:**"微软雅黑"**,**sans-serif**;  
**font-size**: 13**px**;  
}  
**a**,  
**a**:**hover**{  
**text-decoration**: **none**;  
**color**: **#666**;  
}  
**ul**{  
**list-style**:**none**;  
}  
**input**{  
/\*清除文本框获取焦点时默认的边框阴影\*/  
**outline**: **none**;  
/\*去除边框\*/  
**border**:**none**;  
/\*添加边框\*/  
**border**:1**px solid #ccc**;  
}

.**f\_left**{  
**float**: **left**;  
}  
.**f\_right**{  
**float**: **right**;  
}  
.**clearfix**::**before**,  
.**clearfix**::**after** {  
**content**: **""**;  
**display**: **block**;  
**height**: 0;  
**line-height**: 0**px**;  
**clear**: **both**;  
**visibility**: **hidden**;  
}  
/\*2.添加新的样式\*/  
.**m\_left10**{  
**margin-left**:10**px**;  
}  
.**m\_right10**{  
**margin-right**:10**px**;  
}  
.**m\_top10**{  
**margin-top**:10**px**;  
}  
.**m\_bottom10**{  
**margin-bottom**:10**px**;  
}

## 京东首页整体结构

步骤结构分析：

1. 引入移动端开发的meta标签

<metaname="viewport" content="width=device-width, initial-scale=1.0，user-scalable=0">

1. 模块化结构搭建
2. 搜索块 b、轮播图 c、导航块 d、产品块





1. 最大宽度和最小宽度的样式设置 max-width:640px; min-width:320px;

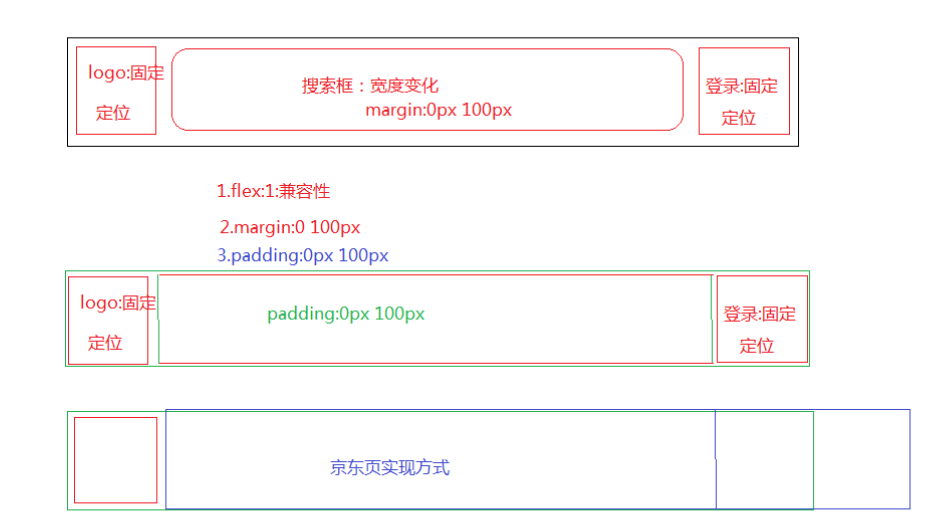
为了解决页面在一些常见的设备上都能正常的显示

max-width:640px 是根据ui最大设计尺寸为640px来的，例如我们的轮播图图片尺寸此时是640，如果超过640就会对图片拉伸图片就会失真，所以我们需要设置一个最大宽度

min-width:320px 因为我们设置的宽度都是100%，当它无限小的时候最小，页面布局就会掉下来或者挤在一起这样就会影响我们页面的布局，所以我们加一个最小宽度，基本上市面上最小屏的智能机是iphone4s它的尺寸为320px

## 京东首页顶部结构块的制作方式分析

1. 头部固定块结构
   1. 效果：
   2. Logo
   3. 表单元素-搜索栏
   4. 登录按钮
   5. 样式：

结构分析图：

### 用flex来实现

<!DOCTYPE **html**>  
<**html**>  
<**head**>  
<**meta charset="utf-8"**>  
<**style**>  
.**parent**{  
**height**:500**px**;  
**border**:1**px solid #222**;  
**display**:**flex**;/\*设为伸缩容器\*/  
}  
 .**stable**{  
**width**:200**px**;/\*固定宽度\*/  
**border**:1**px solid #ccc**;  
**margin**:20**px**;  
 }  
 .**change**{  
**flex**:1;/\*这里设置为占比1，填充满剩余空间\*/  
**border**:1**px solid #ff4400**;  
**margin**:20**px**;  
 }  
</**style**>  
</**head**>  
<**body**>  
<**div class="parent"**>  
<**div class="stable"**>stable 固定宽度200px</**div**>  
<**div class="change"**>changeable 可变宽度</**div**>  
</**div**>  
</**body**>  
</**html**>

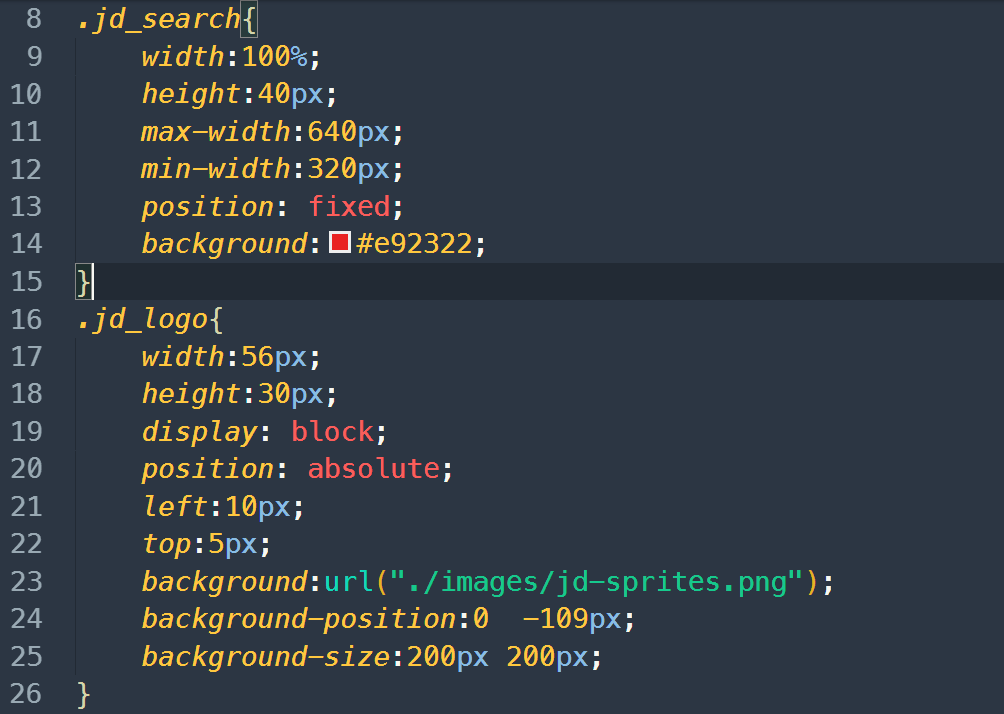
### 用margin:0 100px来实现

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
<**meta charset="UTF-8"**>  
<**title**>Title</**title**>  
<**style**>  
**\***{  
**padding**: 0;  
**margin**: 0;  
 }  
 .**jd\_search**{  
**width**: 100%;  
**height**: 40**px**;  
**background-color**: **#ccc**;  
**position**: **relative**;  
 }  
 .**jd\_logo**{  
**width**: 100**px**;  
**height**: 100%;  
**display**: **block**;  
**background-color**: **green**;  
**position**: **absolute**;  
**left**: 0;  
**top**:0;  
 }  
 .**jd\_login**{  
**width**: 100**px**;  
**height**: 100%;  
**display**: **block**;  
**background-color**: **blue**;  
**position**: **absolute**;  
**top**: 0;  
**right**: 0;  
 }  
 .**jd\_searchBox**{  
/\* !\*宽度参照父容器,不能人为设置具体的宽度值\*! \*/  
**margin**:0 100**px**;  
**height**: 100%;  
**background-color**: **red**;  
 }  
</**style**>  
</**head**>  
<**body**>  
<**div class="jd\_search"**>  
<**a href="javascript:;"class="jd\_logo"**></**a**>  
<**form action=""class="jd\_searchBox"**>  
<**input type="text"**>  
</**form**>  
<**a href="javascript:;"class="jd\_login"**>登录</**a**>  
</**div**>  
</**body**>  
</**html**>

### 用padding:0 100px来实现

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
<**meta charset="UTF-8"**>  
<**title**>Title</**title**>  
<**style**>  
**\***{  
**padding**: 0;  
**margin**: 0;  
 }  
 .**jd\_search**{  
**width**: 100%;  
**height**: 40**px**;  
**background-color**: **#ccc**;  
**position**: **relative**;  
 }  
 .**jd\_logo**{  
**width**: 100**px**;  
**height**: 100%;  
**display**: **block**;  
**background-color**: **green**;  
**position**: **absolute**;  
**left**: 0;  
**top**:0;  
 }  
 .**jd\_login**{  
**width**: 100**px**;  
**height**: 100%;  
**display**: **block**;  
**background-color**: **blue**;  
**position**: **absolute**;  
**top**: 0;  
**right**: 0;  
 }  
 .**jd\_searchBox**{  
**width**: 100%;  
**height**: 100%;  
**padding**:0**px** 100**px**;  
**background-color**: **red**;  
 }  
</**style**>  
</**head**>  
<**body**>  
<**div class="jd\_search"**>  
<**a href="javascript:;"class="jd\_logo"**></**a**>  
<**form action=""class="jd\_searchBox"**>  
<**input type="text"**>  
</**form**>  
<**a href="javascript:;"class="jd\_login"**>登录</**a**>  
</**div**>  
</**body**>  
</**html**>

## 京东首页顶部结构块制作



1、图片失真问题：精灵图长度和宽度通过background-size来设置取它的一半

2、表单宽度100% ; 给它设置padding-left和padding-right没出现滚动条原因是box-sizing属性

## 京东首页轮播图结构块制作

结构布局：

1. 图片布局
2. 图片盒子的宽度是每张图片100%的总和宽度;
3. 顶级父盒子设置overflow：hidden 让超出即隐藏
4. 每个图片盒子li的宽度：1/图片张数
5. 点盒子布局

a、点盒子居中：通过定位设置left:50%;在设置其位移宽度的一半

或者margin-left:-64px 也可以让其居中

B、当前的添加active类名

轮播图制作完成之后，还有一个小的问题：

1. 我们默认有一部分被挡住了
2. 而最终的效果是刚开始是透明的没有被挡住，只有轮播图上去之后才变成不透明的
3. 这块我们后面会通过js来实现这样的效果

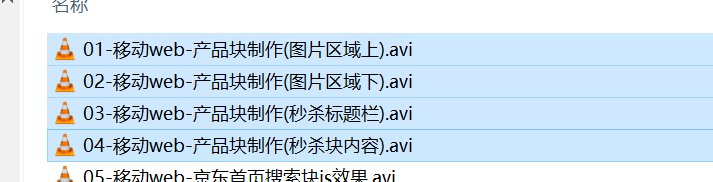
## 京东首页导航块制作

1. 清除浮动
2. 图片设置为原图的一半

轮播图中图片底部有黑色，一般直接对其图片设置一个display:block；

# 第一天作业

1. 完成如下几个视频对应的切图工作



1. 用onmousedown,onmousemove,onmouseup实现div的拖拽（PC端即可，不用考虑移动端,不考虑细节，能实现拖动就行了）

参考答案：

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <style>  .ceshi{  position: absolute;  left:100px;  top:100px;  width: 100px;  height: 100px;  background: blue;  }  </style>  <script>  window.onload = function(){  var ceshi = document.querySelector('#ceshi');  var startX,startY,moveX,moveY,distanceX,distanceY;  var isMove = false;  ceshi.onmousedown = function(e){  startX = e.clientX;  startY = e.clientY;  isMove = true;  }  ceshi.onmousemove = function (e) {  if (isMove == false) {  return;  }  moveX = e.clientX;  moveY = e.clientY;  distanceX = moveX - startX;  distanceY = moveY - startY;  ceshi.style.left = ceshi.offsetLeft + distanceX + 'px';  ceshi.style.top = ceshi.offsetTop + distanceY + 'px';  startX = moveX;  startY = moveY;  }  ceshi.onmouseup = function () {  isMove = false;  }  }  </script>  </head>  <body>  <div class="ceshi" id="ceshi"></div>  </body>  </html> |

## 产品块制作(图片区域上)

结构分析：

1. 对于秒杀和京东超市有公共区域就用公共的样式编写
2. 伪元素 :before和 :after 的使用，containt:””，以及绝对定位不能忘了

3、京东超市区块：左边和右边布局 添加左右浮动即可，**切记：一定要清除浮动**

## 产品块制作(图片区域下)

实现图片下面阴影运用的是CSS3中所学的

box-shadow ：1px 1px 1px 1px #ccc

## 产品块制作(秒杀标题栏)

**结构分析：**

1. 头部：左边左浮动，右边右浮动
2. 掌上秒杀和产品块用了一个公共的头部样式，为了只更改掌上秒杀需要重新定义一个样式，也就是 .js\_sk{}
3. 选择器 nth-of-type(3n) / nth-last-of-type(3n) 找到三的倍数的元素

nth-of-type(3n)：正数

nth-last-of-type(3n) 倒数

## 产品块制作(秒杀块内容)

**结构步骤：**

1. 最外层盒子类名为 .jd\_pContent设置内边距 padding
2. 每个图片盒子li 设置浮动均分成三等分、内容居中显示
3. 图片可以点击用a标签包裹课链接，图片需要设置尺寸
4. 最后一定不能忘了清除浮动

# 京东首页js效果

## 京东首页搜索块js效果

观察一下，有啥变化？？

*// 头部搜索块的js效果（function searchEffect）*

*// 1.获取当前banner轮播图的高度*

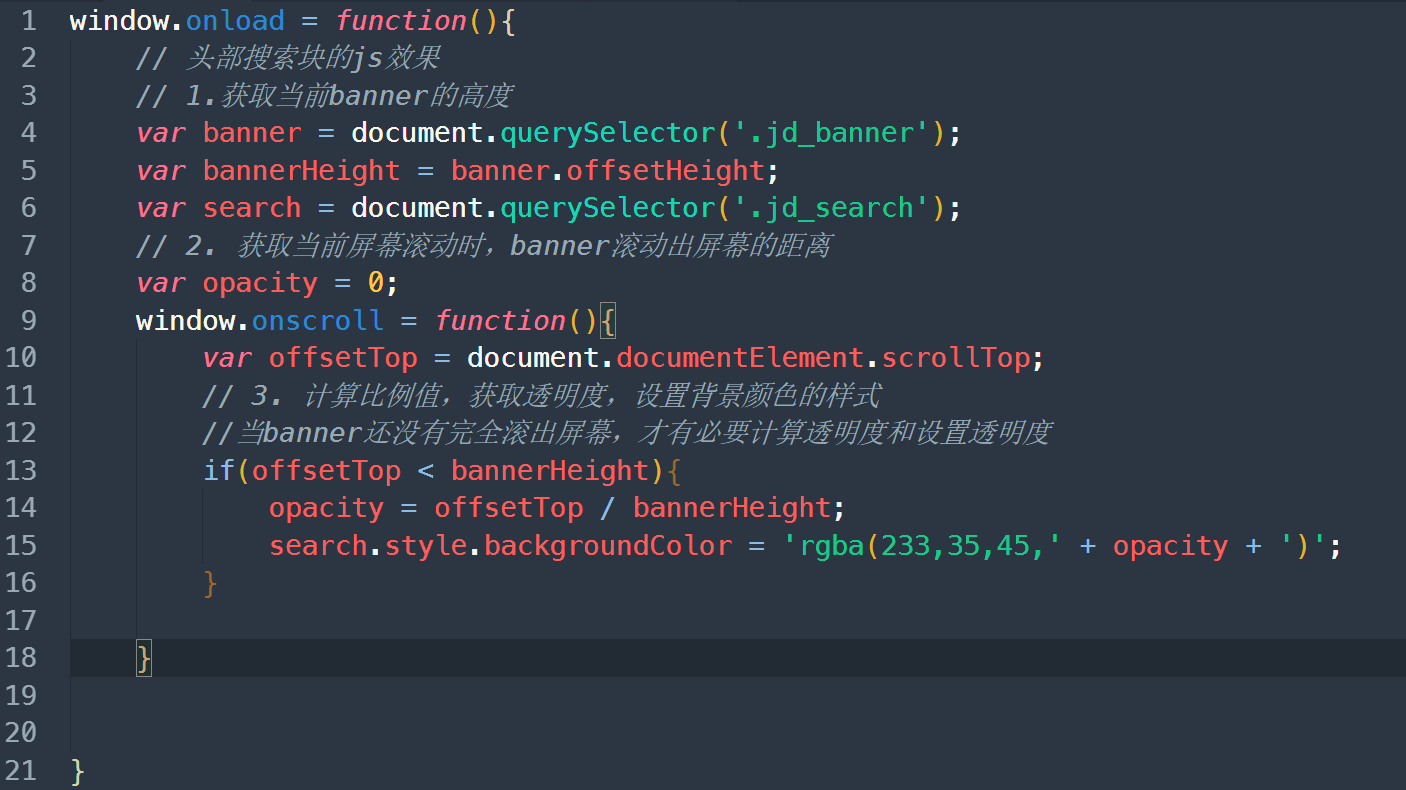
*// 2. 获取当前屏幕滚动时，文档滚动出屏幕的距离*

*// 3. 计算比例值，获取透明度，设置背景颜色的样式*

*//当banner还没有完全滚出屏幕，才有必要计算透明度和设置透明度*

需求分析：

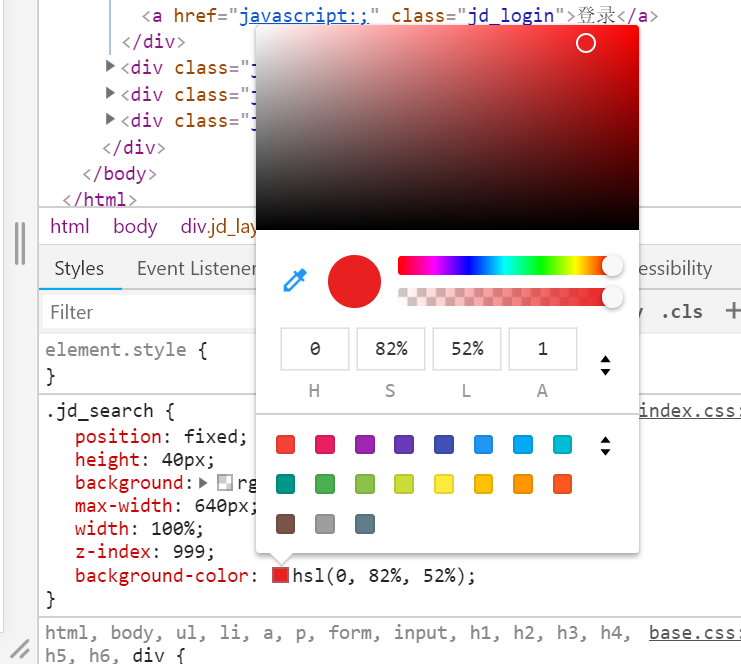
1. 默认情况下搜索部分是看不到的
2. 当页面往上滚动的时候，搜索慢慢的可以看到了，直到轮播图完全滚动上去了之后，搜索完全被看到
3. 首先求出来轮播图的高度值（固定的）
4. 接下来求出文档滚出去的高度
5. 当轮播图的高度和滚出去的高度相等的时候，这时候就全部可以看到了



Window.onload=function(){

}

如何通过浏览器查看颜色的rgba值：



注意：图片的宽高是自适应的，所以我们打印出来的offsetHeight不一定是视频中的238px

bug:

1. 如果设置了DTD头信息，则需要使用document.documentElement.scrollTop
2. 如果没有设置，可以用document.body.scrollTop
3. 给头部设置背景颜色rgba() 透明值为0
4. 分析渐变效果
   1. 开始为滚动透明度为0，随着滚动的距离加大，背景颜色逐渐加深
   2. 滚动超过一定高度时，背景颜色就不在改变
5. js效果实现步骤
   1. 引入的是外部js需要 js页面加载完入口函数: window.onload=function(){}
   2. 用CSS3的选择器quertSelector() 获去轮播图和搜索盒子对象，并得到轮播图盒子的高度
   3. 滚动事件用到 window.onscroll =function(){}，同时对滚动距离和轮播图高度进行判断 从而得到opacity的值

## 京东首页倒计时js效果

引言：

3700秒相当于多少小时 多少分钟 多少秒

*// 倒计时效果(timeBack)*

*// 1. 获取用于展示时间的span(qsa)*

*// 2.设置初始的倒计时时间（假设总时间是3700秒）,时间以秒做为单位*

*// 3.开启定时器(timerId)*

*//判断倒计时是否结束了，如果结束了，清除定时器,同时return*

*// 得到剩余时间中的 时 分 秒，这块先不要写代码，用3700把这几个数算出来，然后再用代码来写*



a/b 这个是取整 求商

a%b 这个是 求余

**Math**.floor()向下取整

**一、步骤分析：**

1. 获取所有的展示时间的span
2. 设置需要定时的时间
3. 开启定时器 (一秒一次自减 )
4. 计算时分秒

/\*获取时\*/  
var hour=*Math*.floor(totalTime/3600);  
/\*获取分\*/  
var minute=*Math*.floor(totalTime%3600/60);  
/\*获取秒\*/  
var second=*Math*.floor(totalTime%60);  
/\*赋值，将时间填充到span中 12\*/  
spans[0].innerHTML=*Math*.floor(hour/10);  
spans[1].innerHTML=*Math*.floor(hour%10);

spans[3].innerHTML=*Math*.floor(minute/10);  
spans[4].innerHTML=*Math*.floor(minute%10);

spans[6].innerHTML=*Math*.floor(second/10);  
spans[7].innerHTML=*Math*.floor(second%10);

1. 判断定时的时间小于0就清除定时器

if(totalTime <0){  
/\*清除时钟\*/  
clearInterval(timerId);  
return;  
}

## 无限轮播图

晚上写轮播图梳理思路：

1. 自动轮播：只能往一个方向轮播
   1. 首尾添加一张图片
   2. 如图，默认是index=1,下一张是index++
      1. 临界情况，假如最后一张了，则重置到index=1
2. 手动轮播：可以往两个方向轮播
   1. 手动轮播的一瞬间，把自动轮播的定时器关掉
   2. 上一张 index--
   3. 下一张 index++
   4. 手动轮播结束的时候，再开启定时器自动轮播
3. 点标记的设置
4. 优化代码
   1. 在定时器开启之前先清一下
   2. 默认isEnd = true
   3. 在所有的transition代码的时候加上开关isEnd = false
   4. 在move,end事件的前面判断如果isEnd的值为false,则直接return（比较粗暴，但是有效果）
   5. 在transitionEnd结束的时候，让isEnd = true
   6. 把定时器函数放到transitionEnd结束的地方即可

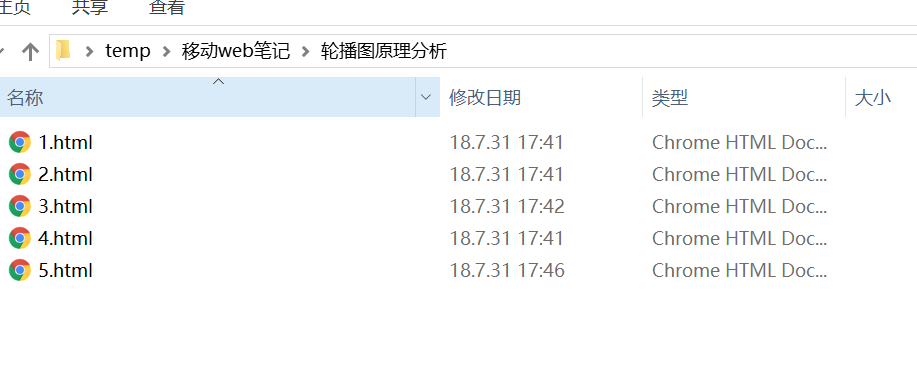
### 手动修改页面结构和样式

*//首添加最后一张图片，尾添加第一张图片*

*//修改图片盒子的宽度*

*//改变图片盒子为position:relative,并且默认初始left值*

分解一下进行理解思路：

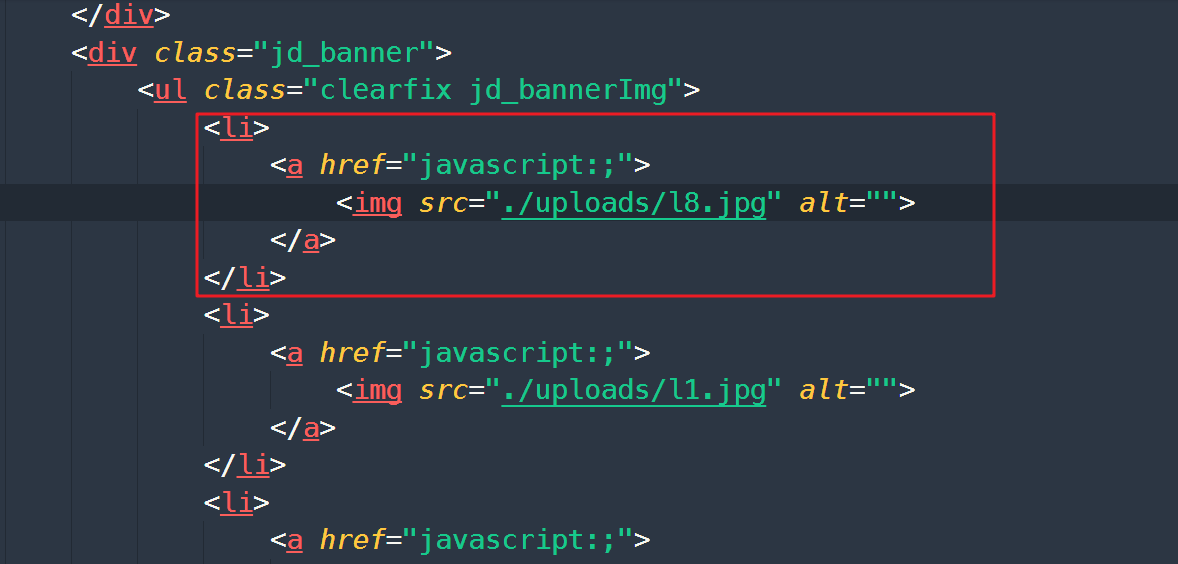


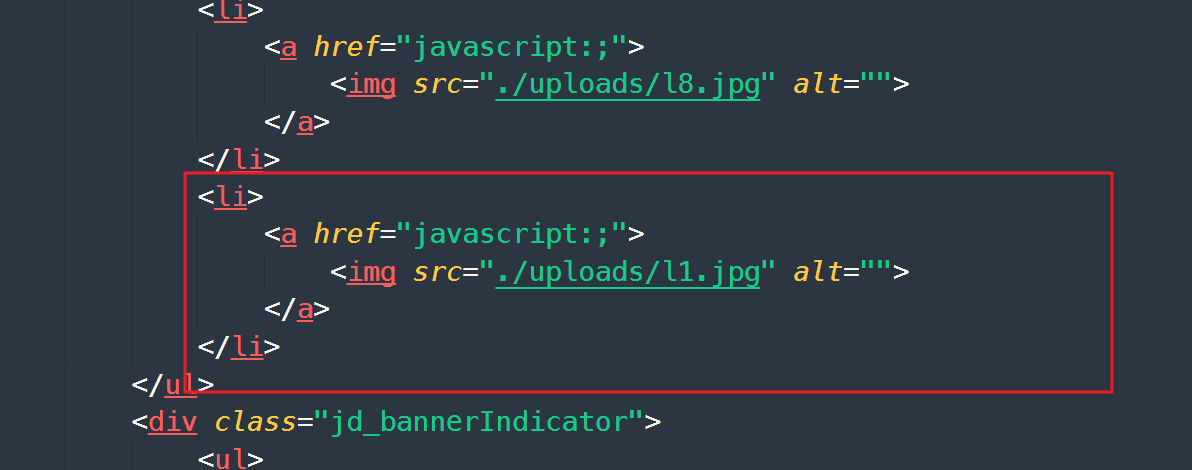
结论：

1. 假如原始有8张图片
2. 首尾添加两张总共是10张图片
3. 原来一个li占图片盒子总宽度的12.5%
4. 现在一个li占图片盒子总宽度的10%
5. 当切换到最后一个li的时候，重置到第二个li(假设索引值是从0开始，则当切换到length - 1的时候，让索引值重置为1)

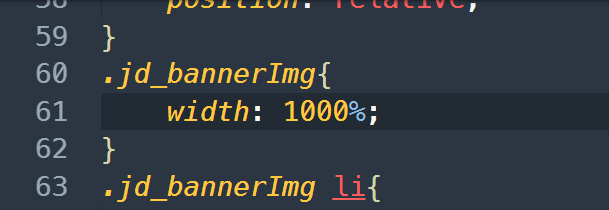
步骤：

第一步、首尾添加一张图片



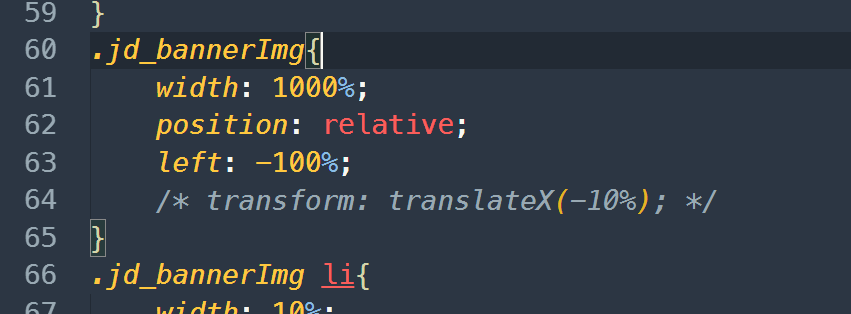


第二步、





第三步、



**思路：**

1.在首尾添加图片

a).在开始位置添加原始的最后一张图片

b).在最后位置添加原始的第一张图片

2.修改页面结构

3.修改对应的样式

.jd\_bannerImg{

width:1000%;

}

.jd\_bannerImg > li{

width:10%;

float: left;

}

4.设置默认的偏移：默认应该显示索引1的图片

transform:translateX(-10%):参照自身

left:-100%:参照父容器

使用定位实现的细节：当前元素必须使用**relative**定位，否则父容器无法获取正确的高度

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <style>  .ceshi{  position: relative;  overflow: hidden;//主要是这一行影响了  }  .wrapper{  margin:100px auto;  width: 100%;  border:1px solid black;  position: absolute;  left:100px;  }  .demo{  width: 100%;  }  .aaa{  height: 300px;  background-color: blue;  }  </style>  </head>  <body>  <div class="ceshi">  <div class="wrapper">  <div class="demo">  <div class="aaa"></div>  </div>  </div>  </div>  </body>  </html> |

### 通过js添加首尾图片

*// 1. 由于轮播图图片数量不确定，不能写死，必须要由js动态设置修改轮播图的页面结构*

*// 1.0 把上面修改的首尾添加的删除掉*

*// 1.1 获取轮播图结构 banner (.jd\_banner)*

*// 2.2 获取图片容器 imgBox (ul:first-of-type)*

*// 1.3 获取原始的第一张图片 first (li:first-of-type)*

*// 1.4 获取原始的最后一张图片 last (li:last-of-type)*

*// 1.5 在首尾插入两张图片appendChild*

*// insertBefore(需要插入的元素，位置)*

1. **选择器的回顾**
   1. first-of-type
   2. last-of-type
2. **深度克隆+添加元素+插入位置** 
   1. cloneNode(true)
   2. appendChild()
   3. insertBefore(插入对象，位置)



### 通过js设置样式

*// 2.设置对应的样式*

*// 2.1 获取所有的li元素(var lis); 一定要console.log测试是不是我们要的东西*

*// 2.2 获取li元素的数量 var count 一定要console.log测试是不是我们要的东西*

*// 2.3 获取banner的宽度 var bannerWidth一定要console.log测试是不是我们要的东西*

*// 2.4 设置图片盒子的总宽度(imgBox) 审查元素，看是否添加成功*

*//2.5 设置每个li元素的宽度为 bannerWidth*

*// 3. 设置默认偏移(负的一个bannerWidth)*

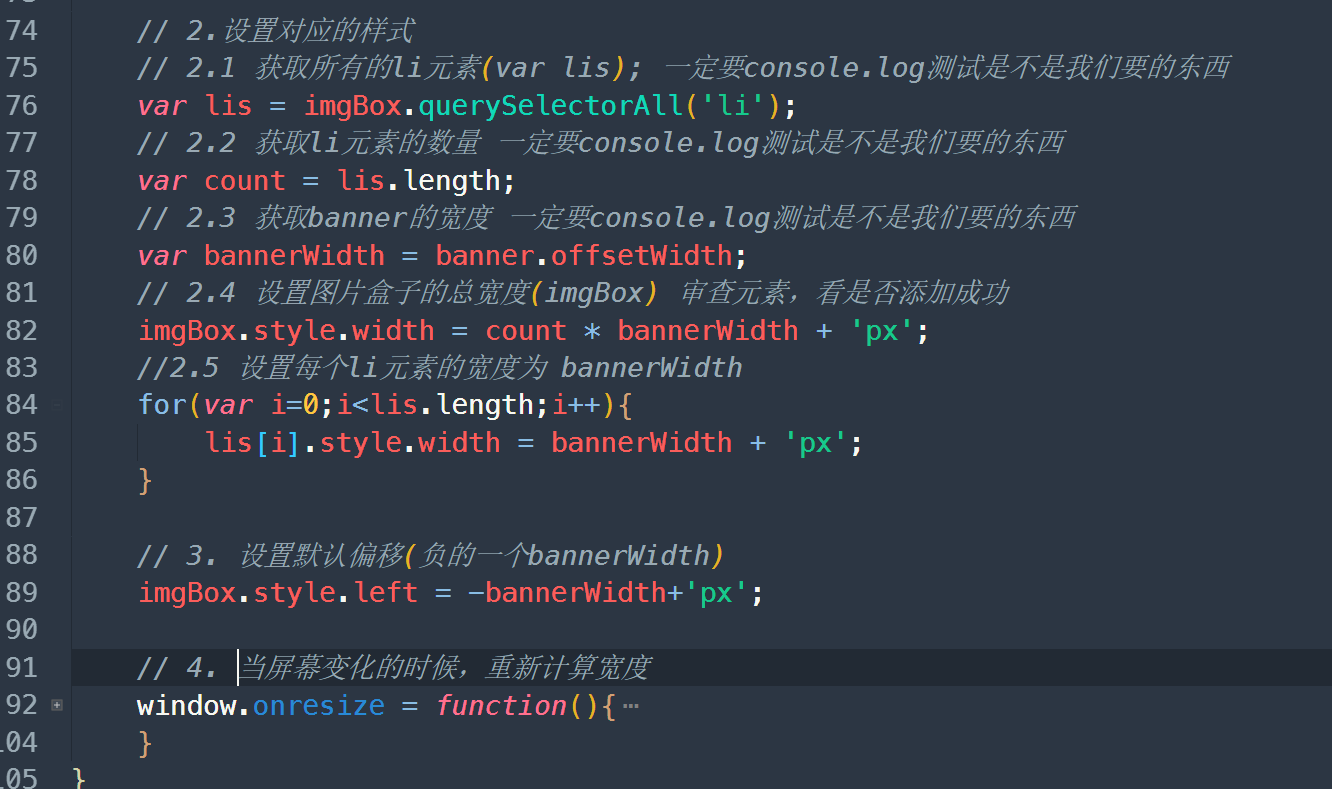
*// 4. 当屏幕宽度变化的时候，重新计算宽度*

*// 4.1.重新获取banner的宽度, 覆盖全局的宽度值*

*// 4.2 重新设置图片盒子的宽度*

*// 4.3 重新设置每一个li(图片)元素的宽度*

*// 4.4 重新设置定位值*



**步骤：**

**1、通过js设置整个轮播图盒子的宽度以及定位**

* 1. 获取所有的li的个数
  2. 获取轮播图的宽度
  3. 整个轮播图盒子的宽度=所有的li的个数\*轮播图的宽度
  4. 设置每个li的宽度，通过遍历获取每个li，在给其赋值为轮播图的宽度
  5. 设置第二张图片首先显示，需要让其位偏移一个负值的轮播图的宽度

/\*2.设置对应的样式\*/  
/\*2.1获取所有li元素\*/  
var lis=imgBox.querySelectorAll("li");  
/\*2.2 获取li元素的数量\*/  
var count=lis.length;  
/\*2.3.获取banner的宽度\*/  
var bannerWidth=banner.offsetWidth;  
/\*2.4 设置图片盒子的宽度\*/  
imgBox.style.width=count\*bannerWidth+"px";  
/\*2.5 设置每一个li(图片)元素的宽度\*/  
for(var i=0;i<lis.length;i++){  
 lis[i].style.width=bannerWidth+"px";  
};  
/\*3.设置默认的偏移\*/  
imgBox.style.left=-bannerWidth+"px";

1. **屏幕改变时重新计算宽度**
   1. 使用onresize事件对屏幕实时监听
   2. 在onresize事件监听下重新再获取轮播图的宽度+图片盒子的宽度+每个li的宽度+图片盒子的位偏移

/\*4.当屏幕变化的时候，重新计算宽度\*/  
window.onresize=function(){  
/\*4.1.获取banner的宽度,覆盖全局的宽度值\*/  
bannerWidth=banner.offsetWidth;  
/\*4.2 设置图片盒子的宽度\*/  
imgBox.style.width=count\*bannerWidth+"px";  
/\*4.3设置每一个li(图片)元素的宽度\*/  
for(var i=0;i<lis.length;i++){  
 lis[i].style.width=bannerWidth+"px";  
};  
/\*4.4重新设置定位值\*/  
imgBox.style.left=-index\*bannerWidth+"px";  
};

### js实现自动轮播

*//5. 实现自动轮播*

*//5.1 定义图片索引,索引值从0开始，但由于图片已经偏移了一个li的距离，所以默认初始值为1*

*// 5.2 开启定时器(setinterval 2000毫秒)*

*//5.3 索引自增*

*//5.4 添加过渡效果*

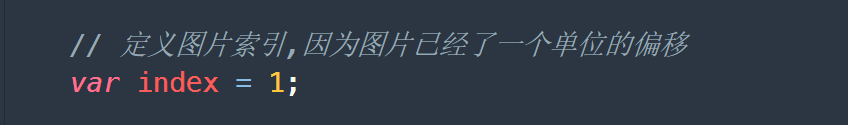
*//5.5 根据最新的索引值重新设置偏移*

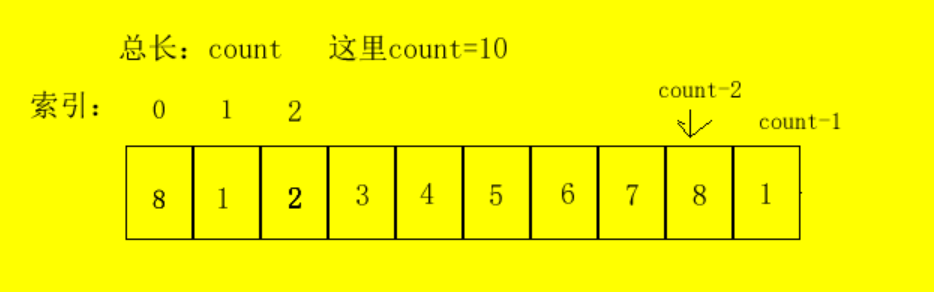
*// 5.6 500毫秒以后，判断是否是最后一张*

*//5.7 如果是最后一张，则让index重置*

*//5.8 去掉过渡效果*

*//5.9 重新修改图片盒子的偏移值*





**1、自动轮播**

* 1. setInterval(function(),2000) 定时器2s一次
  2. setTimeout(function(){},500) 定时器只执行一次，有500毫秒的延迟

1. **自动轮播+无缝轮播的原理**
   1. 自动轮播原理
      1. 首先申明全局的index=1(因位偏移首先显示的index=1的那张图片)
      2. 定时器1s执行一次，让索引自增一次，
      3. 设置图片盒子的位偏移 =-当前的索引\*图片盒子的宽度
      4. 添加过渡让其有动画效果

imgBox.style.transition="left 0.5s ease-in-out";  
imgBox.style.left=(-index\*bannerWidth)+"px";

b) 无缝轮播的原理

i. 添加定时延长

ii. 判断索引值为9(也就是最后一张),直接让其索引index=1

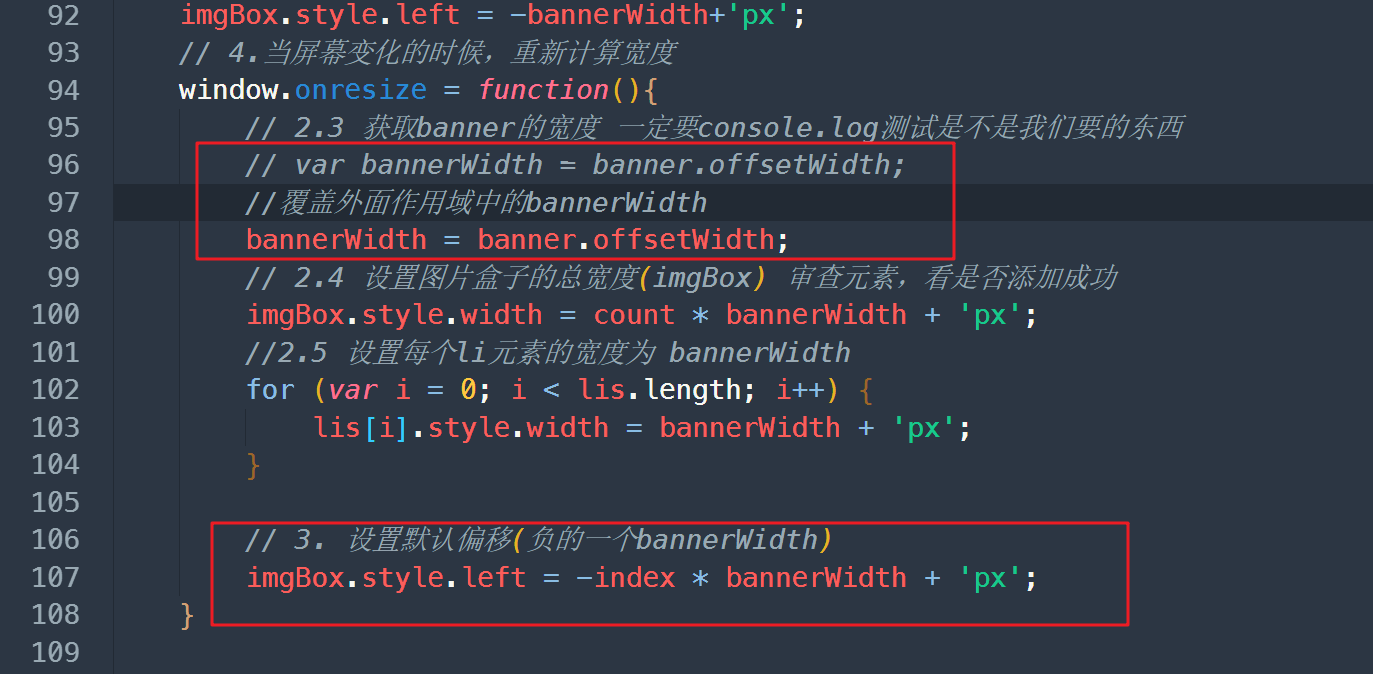
iii. 设置图片盒子的位偏移 = -当前的索引\*图片盒子的宽度

Iv. 关闭过渡

setTimeout(function(){  
if(index==count-1){  
 console.log(index);  
index=1;  
imgBox.style.transition="none";  
imgBox.style.left=(-index\*bannerWidth)+"px";  
}  
},500);

### 轮播图自动轮播完善(不放)

*//6 完善resize的时候当前的left值*



**1、为什么会出现屏幕变化时图片自动轮播还是会有错乱效果**

onresize事件监听后重新定义了一个bannerWidth ，而轮播图在定时器的作用下对图片盒子进行的位偏移值(-index\*bannerWidth)这里的bannerWidth还是调用的全局变量那个值，并不是获取屏幕改变后的值，为解决这个问题，我们只需要将监听的值覆盖全局变量的值即可

## 移动端touch事件

引言：

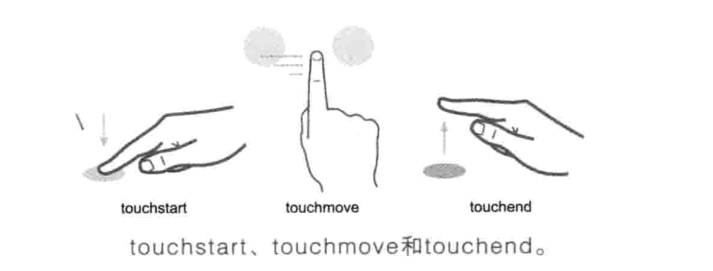
在移动端，手指替代了鼠标

onclick,onmousemove,onmouseup

### 移动端的touch事件简介

**1、touch事件有几种及什么情况下触发？**

* 1. touchsatrt 当手指触摸屏幕的时候触发
  2. touchmove 当手指在屏幕上滑动的时候连续触发
  3. touchend 当手指离开的时候触发



**2、touch事件触发的顺序**

touchsatrt>touchmove >touchend



|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0,user-scalable=no,maximum-scale=1.0,minimum-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <style>  div{  width: 100px;  height: 100px;  background-color:red;  }  </style>  </head>  <body>  <div></div>  <script>  // 1. 获取dom元素  var div = document.querySelector('div');  // 2.添加事件  // 添加开始触摸事件：当手指触摸到屏幕时触发  div.addEventListener('touchstart',function(){  console.log('touchstart');;  })  // 添加手指滑动事件，当手指在屏幕上滑动时触发 move事件是持续触发的  div.addEventListener('touchmove',function(){  console.log('touchmove');  })  //添加触摸结束事件，当手指离开屏幕时触发  div.addEventListener('touchend',function(){  console.log('touchend');  })  // 添加触摸意外中断事件  div.addEventListener('touchcancel',function(){  console.log('touchcancel');  })  </script>  </body>  </html> |

### 移动端的touch事件(手指对象数组)

touchstart,touchmove --> targetTouches

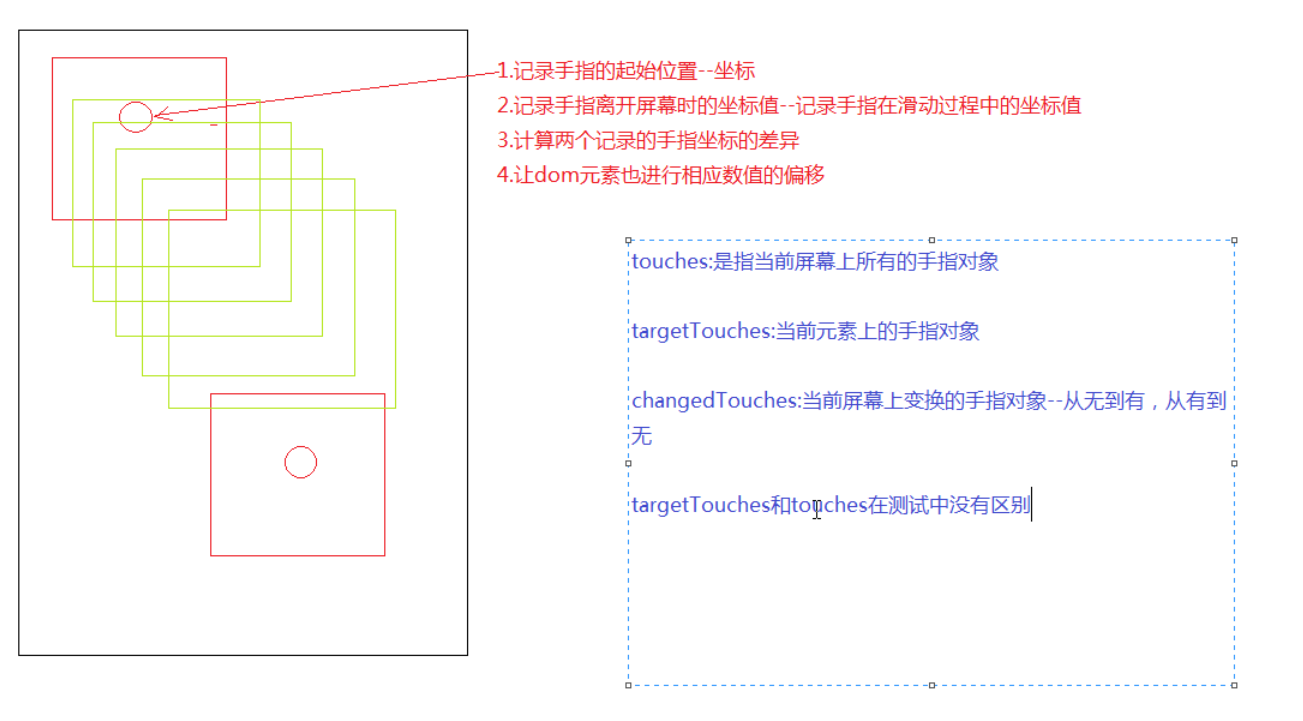
touchend --> changedTouches

### touches、targetTouches、changedTouches的意思是什么？

这块要补充的：changedTouches在什么情况下我们要用它

1. changedTouches 当前屏幕上变换的手指对象 从无到有或从有到无 (touchsatrt时包含与触摸屏接触的触点，touchend时包含离开触摸屏的触点)
2. targetTouches 位于该元素上的所有手指的列表
3. touches 位于屏幕上的所有手指列表

注：没有对比出touches同targetTouches的差异，推荐使用targetTouches







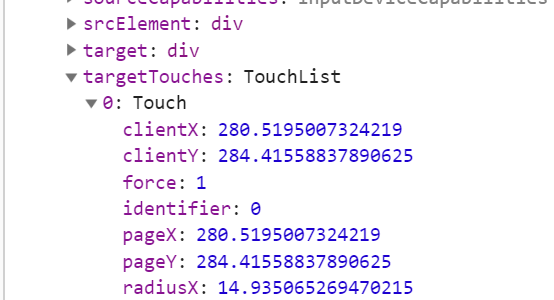
|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0,user-scalable=no,maximum-scale=1.0,minimum-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <style>  div{  width: 300px;  height: 300px;  background-color:red;  }  </style>  </head>  <body>  <div></div>  <script>  var div = document.querySelector('div');  // 开始触摸  div.addEventListener('touchstart',function(e){  console.log('touchstart');  console.log(e);  console.log('===============');  })  div.addEventListener('touchmove', function (e) {  console.log('touchmove');  console.log(e);  console.log('================');  })  div.addEventListener('touchend', function (e) {  console.log('touchend');  console.log(e);  console.log('=============');  })  // 触摸滑动  // 触摸结束  </script>  </body>  </html> |

我们记住targetTouches、changedTouches就行了

### 手指对象的坐标获取

**1、touches下面获取坐标是哪几个属性均是什么作用？**

* 1. screenX /screenY:手指的触摸点相对于屏幕左上角的坐标距离
  2. clientX /clientY:相对于当前窗口--移动端屏幕
  3. pageX /pageY:相对于当前页面的内容 会有滚动条 包含滚动条的



### 使用touch事件实现元素的拖拽

1. **元素实现拖拽的原理是什么？**
   1. 获取当前触摸点的坐标
   2. 获取滑动点的坐标
   3. 获取两者的差值
   4. 把差值传给对象就是它的位移值了

div.addEventListener("touchstart",function(*e*){  
*startX*= *e*.targetTouches[0].clientX;  
*startY*= *e*.targetTouches[0].clientY;  
console.log(*startX*+":"+*startY*);  
});  
/\*触摸滑动:持续触发\*/  
div.addEventListener("touchmove",function(*e*){  
/\*记录手指滑动过程中的坐标值\*/  
*moveX*= *e*.targetTouches[0].clientX;  
*moveY*= *e*.targetTouches[0].clientY;  
console.log(*moveX*+":"+*moveY*);  
/\*计算与上一次坐标的差异\*/  
*distanceX*=*moveX*-*startX*;  
*distanceY*=*moveY*-*startY*;  
console.log(*distanceX*+":"+*distanceY*);  
/\*设置偏移\*/  
div.style.transform="translate("+*distanceX*+"px,"+*distanceY*+"px)";  
});

当然，视频中的拖拽有一点小的问题，如果大家介意的话，可以按如下的方式来实现：

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0,user-scalable=no,maximum-scale=1.0,minimum-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <style>  \*{  margin:0;  padding:0;  }  div{  width: 100px;  height: 100px;  background: linear-gradient(to right,red,blue);  transform:translate()  }  </style>  </head>  <body>  <div></div>  <script>  var div = document.querySelector('div');  var startX,startY,moveX,moveY,distanceX,distanceY;  var currentDistanceX = 0;  var currentDistanceY = 0;  div.addEventListener('touchstart',function(e){  startX = e.targetTouches[0].clientX;  startY = e.targetTouches[0].clientY;  })  div.addEventListener('touchmove',function(e) {  moveX = e.targetTouches[0].clientX;  moveY = e.targetTouches[0].clientY;  distanceX = moveX - startX;  distanceY = moveY - startY;  if(getComputedStyle(div).transform != 'none'){  var arr = getComputedStyle(div).transform.slice(0, -1).split(",");  currentDistanceX = arr[4] - 0;  currentDistanceY = arr[5] - 0;  }  div.style.transform = 'translate(' + (currentDistanceX + distanceX) + 'px,' + (currentDistanceY + distanceY) + 'px)';  startX = moveX;  startY = moveY;  })  </script>  </body>  </html> |

### touch事件源参数中的target(事件委托)

这个视频讲的是为后面的分类页面的js特效做铺垫的，一堆的li需要绑定事件，就是复习了一下事件委托的原理

有三个同事预计会在周一收到快递。为签收快递，有两种办法：一是三个人在公司门口等快递；二是委托给前台MM代为签收。现实当中，我们大都采用委托的方案（公司也不会容忍那么多员工站在门口就为了等快递）。前台MM收到快递后，她会判断收件人是谁，然后按照收件人的要求签收，甚至代为付款。这种方案还有一个优势，那就是即使公司里来了新员工（不管多少），前台MM也会在收到寄给新员工的快递后核实并代为签收。

target在之前我们了解过，这里的意思也是一样的，代表的是事件源这个具体的元素

1. **任意对象能拖动原理**

通过对document添加事件可以捕获当前响应事件的对象,然后再获取当前对象的坐标来实现拖动偏移定位

1. **如何使用target**

通过对document添加事件可以捕获当前响应事件的对象e，从而获取当前响应事件对象e的target

## 轮播图-手动滑动

### 轮播图-实现滑动操作

*// 6. 实现手动轮播*

*// 6.1 触摸开始 记录手指的起始位置(startX)*

*//6.1.1 清除定时器*

*//6.2.2 记录手指的起始位置*

*// 6.2 滑动过程 记录手指在滑动过程中的位置(moveX)，，通过left样式实现图片的偏移*

*// 6.2.2 记录手指在滑动过程中的位置(moveX)*

*//6.2.3 计算出相对于起始位置的偏移值(distanceX)*

*//6.2.4 修改图片盒子的left值*

1. **touch事件的触发有什么前提条件**

**必须保证元素有具体的宽高，如果宽或高为0，则不会进行触发**

**2、分析实现手动轮播方法**

1.手指触发轮播图时清除定时器

2.记录手指的起始位置

3.记录手指在滑动过程中的位置，计算出相对于起始位置的偏移值，通过left样式实现图片的偏移 (最后这个偏移是当前图片相对于起始位置的偏移值+当前滑动的距离)

imgBox.style.left=(-index\*bannerWidth + distanceX)+"px";

4.在松开手指之后，判断当前滑动的距离，如果超出指定的范围，就翻页，否则回弹

5.松开手指之后，重新开启定时器

### 轮播图-手动滑动-细节



1. **滑动时图片产生延迟的原因是什么**

开启定时器的时候添加了过渡效果，在touchmove的时候没有清除这个过渡也就导致那个延迟了，解决的办法是在touchmove事件中清除过渡

### 轮播图-松开手指之后的处理

*//7 添加触摸结束事件*

*// 7.1 获取当前滑动的距离，判断距离是否超出了指定的范围 100px 如果超出了则翻页，否则回弹*

*//7.1.1 如果距离大于0，上一张*

*//7.1.2 如果距离小于0 下一张*

*//7.2 设置过渡效果*

*//7.3 翻页*

*//7.4 重新开启定时器(封装一个函数名字叫startTime)*

1. **滑动的方向如何确定**
   1. 根据distance从左往右为正值 (index--)

b) 根据distance从右往左为负值 (index++)

1. **松开手指后如何实现上一张或者下一张切换**

**原理分析：**

**A:滑动大于100时**

1. .通过滑动的距离(distance)的绝对值大于100时再来判断其方向
2. .滑动的方向根据distance值为正还是为负
3. .为负：从右往左滑 index++ ,为正：从左往右滑 index--
4. .给图片盒子添加过渡效果
5. .图片盒子位偏移=-当前的索引\*图片盒子的宽度

**B:有滑动且滑动距离大于0**

1. .给图片盒子添加过渡效果

b).图片盒子位偏移=-当前的索引\*图片盒子的宽度

**C:滑动结束后在重新开启定时器让其自动轮播**

if(*Math*.abs(distanceX) >100){  
/\*判断滑动的方向\*/  
if(distanceX >0){//上一张  
index--;  
}  
else{ //下一张  
index++;  
}  
/\*翻页\*/  
imgBox.style.transition="left 0.5s ease-in-out";  
imgBox.style.left=-index\*bannerWidth+"px";  
}  
else if(*Math*.abs(distanceX) >0){ //得保证用户确实进行过滑动操作  
/\*回弹\*/  
imgBox.style.transition="left 0.5s ease-in-out";  
imgBox.style.left=-index\*bannerWidth+"px";  
}  
//重新开启定时器  
startTime();

### 添加过渡结束的监听

500毫秒的确这样写是没问题的，但是如果时间变了，就得改好几个地方，这样代码容易被我们改错

1、

*// 8. 考虑手动的临界情况*

*// webkitTransitionEnd:可以监听当前元素的过渡效果执行完毕，当一个元素的过渡效果执行完毕的时候，会触发这个事件*

*//8.1 如果到了最后一张，回到索引1*

*//8.2 如果到了第一张(0),回到索引count-2*

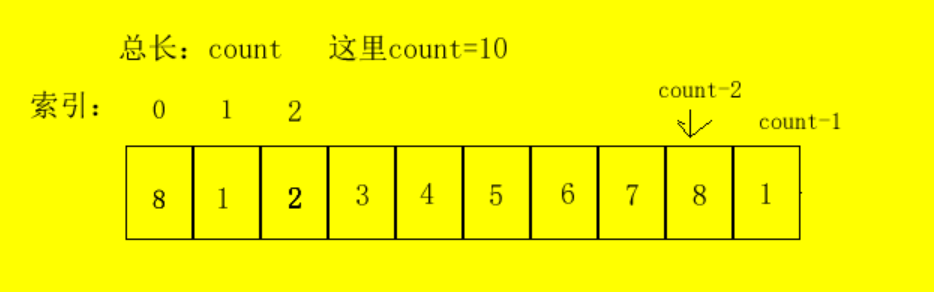
*// 8.3 清除过渡*

*// 8.4 设置偏移*

1. **索引index和count关系**

总长为count由于索引index是从0开始计算，最后一张的索引就和count相差1

1. 最后一张index=count-1
2. Index=0和count-2也是同一张



1. **过渡结束事件的兼容性写法**

webkitTransitionEnd

imgBox.addEventListener("webkitTransitionEnd",function(){  
/\*如果到了最后一张(count-1)，回到索引1\*/  
 /\*如果到了第一张(0)，回到索引count-2\*/  
if(index==count-1){  
 index=1;  
/\*清除过渡\*/  
imgBox.style.transition="none";  
/\*设置偏移\*/  
imgBox.style.left=-index\*bannerWidth+"px";  
}  
else if(index==0){  
 index=count-2;  
/\*清除过渡\*/  
imgBox.style.transition="none";  
/\*设置偏移\*/  
imgBox.style.left=-index\*bannerWidth+"px";  
}  
});

## 第二天作业

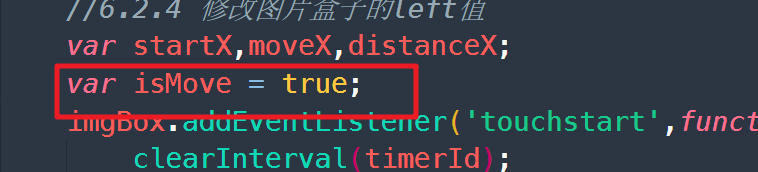
用js判断手指向上滑动还是向下滑动

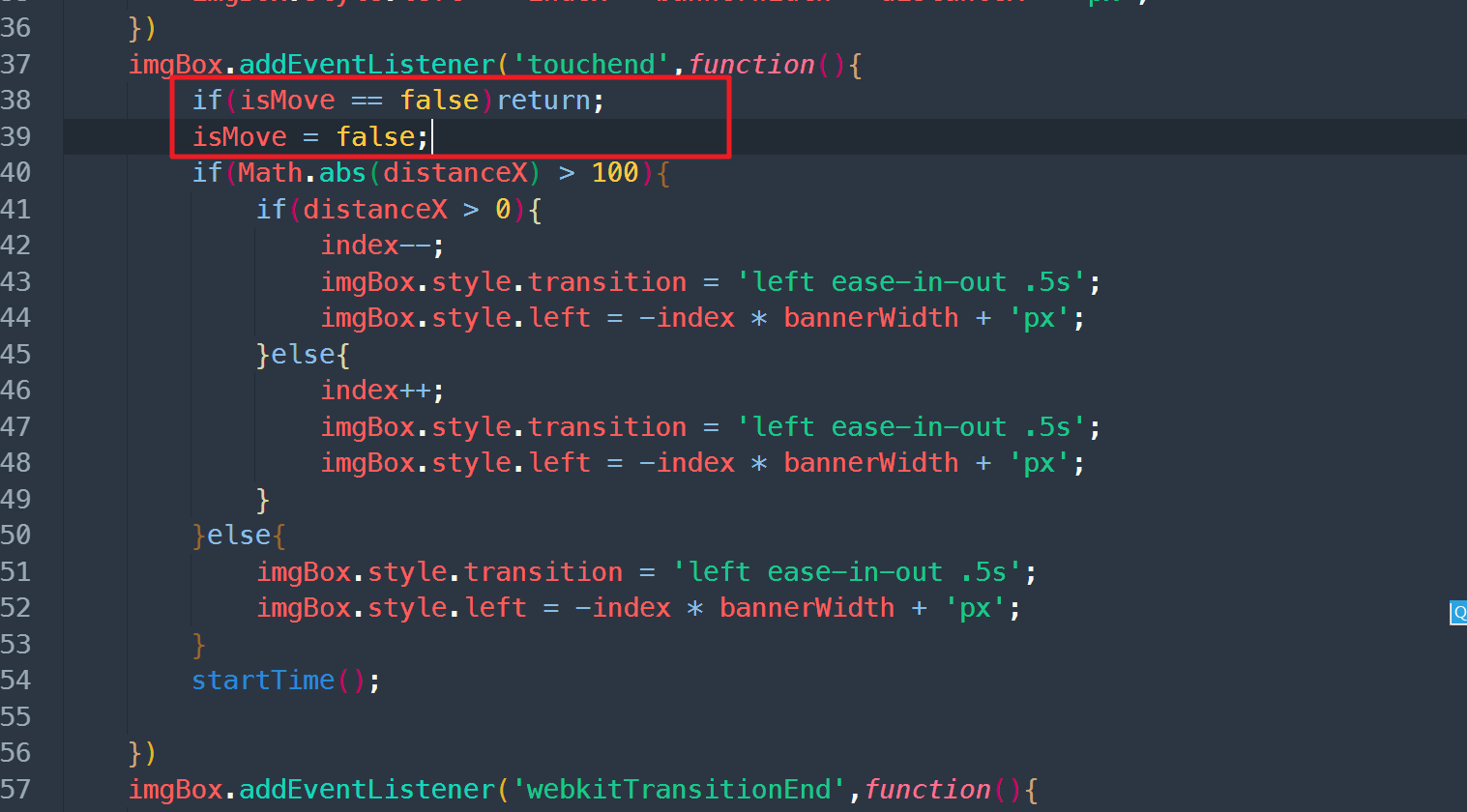
参考答案：

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0,user-scalable=no,maximum-scale=1.0,minimum-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <script>  window.onload = function(){  var startY,moveY,distanceY;  document.addEventListener('touchstart',function(e){  startY = e.targetTouches[0].clientY;  })  document.addEventListener('touchmove',function(e){  moveY = e.targetTouches[0].clientY;  distanceY = moveY - startY;  })  document.addEventListener('touchend',function(){  if(distanceY > 0){  console.log('向下滑');  }else{  console.log('向上滑');  }  })  }  </script>  </head>  <body>  </body>  </html> |

## 轮播图中一个bug的解决

这块可以简单化一些来实现：







只有轮播动画结束了，才可以允许用户操作

*// 1. 初始一个全局变量 设置其值为true,*

*// 2. 在touchmove事件判断阀门是否开启*

*// 3. 在touchend事件关闭阀门*

*// 4. 在transitionEnd事件在开启阀门 + 清定时器 + 重开定时器(100毫秒)*

1. **快速滑动会出现空白的问题？**

当前滑动距离超过100，图片完成过渡效果需0.5s，这时用户又接着多次滑动就导致此时的过渡没有触发，从而就导致了空白问题

1. **如何解决这个问题？**

节流阀的原理：

1. 初始一个全局变量 设置其值为true,
2. 在touchmove事件判断阀门是否开启
3. 在touchend事件关闭阀门
4. 在transitionEnd事件在开启阀门
5. **解决的步骤**
   1. 初始一个变量名:isEnd=true,
   2. 触发滑动touchmove事件时先判断isEnd是否为true, 是true在执行滑动
   3. 在touchend事件触发(即手指松开)在设置isEnd为false,限制拖拽
   4. touchmove和transitionEnd都没有触发轮播图却还可以滑动轮播 就是touchend在触发
   5. 上一次的touchmove虽然没在被触发，但touchmove中的distance值却存在导致后面滑动的distance是一个固定值，所这时我们需要对每次触摸后的值进行初始化为0，这时再次滑动就不会触发了
   6. 为了让当前的滑动完成后在可以滑到下一张，这时就可以在过渡结束(transitionEnd)后再次开启阀门：isEnd=true
   7. 在过渡结束后可以再重新开启定时器，让了看的更加明晰可以加一个定时器的事件延长，方可再次进行拖动，但是在两张图片过渡结束开启定时器之前我们还需要清除定时器

setTimeout(**function**(){  
 isEnd=**true**;  
clearInterval(timerId);  
startTime();},100);

## 轮播图点标记的设置

*// 初始图片索引为1，点的初始索引为0，让图片和点的索引对应，需将图片的索引减去1*

*// index-1个li添加active属性*

**步骤分析：**

1. 要达到同步效果需让图片的索引和点的索引一致
2. 调用封装好的方法传入当前图片的索引
3. 初始图片索引为1，点的初始索引为0，让图片和点的索引对应，需将图片的索引减去1

**var** setIndicator=**function**(index){  
**var** indicators=banner.querySelector("ul:last-of-type")

.querySelectorAll("li");  
/\*先清除其它li元素的active样式\*/  
**for**(**var** i=0;i<indicators.length;i++){  
 indicators[i].classList.remove("active");  
}  
/\*为当前li元素添加active样式\*/  
indicators[index-1].classList.add("active");  
}

## zepto的基本特点和下载

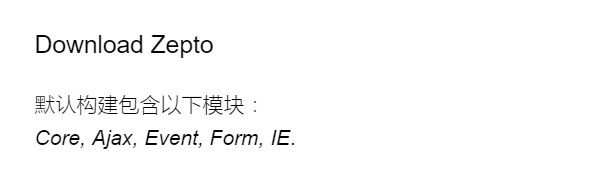
zepto就是套餐饭，我们都很穷，不可能一下子吃那么多菜，可以定制几个菜就行了

1. **为什么使用zepto**
   1. 轻量级体量小
   2. 可以提高我们的开发效率
   3. 针对的都是一些高级浏览器(即移动端浏览器)
2. **如何学习zepto**

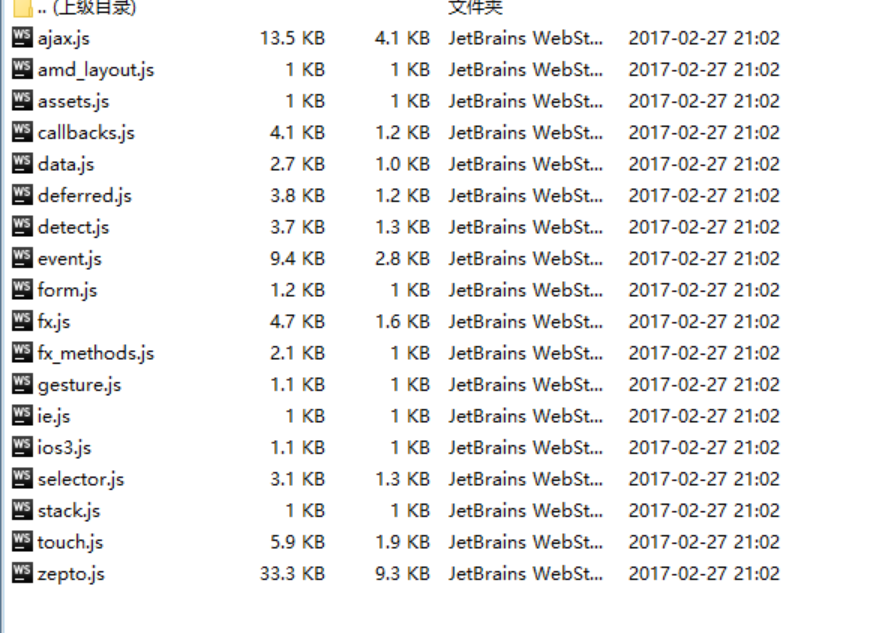
基本上我们会jquery就会使用zepto，跟jquery很多都是一样的,不需要我们去刻意的学习

1. **zepto的特点**

a) zepto是集成模块的，把所有的功能都集成在模块中，核心模块就是如下几种



b) 其他的模块js



## 使用zepto实现轮播图(上)

*// 1. 添加首尾两张图片*

*// 1.1 获取轮播图大的容器*

*// 1.2 获取轮播图大的容器的宽度*

*// 1.3 获取图片盒子*

*// 1.4 获取点标记*

*// 1.5 获取首尾两张图片*

*// 1.6 将两张图片添加到首尾 append,prepend*

*// 2. 重新设置图片盒子的宽度和图片的宽度*

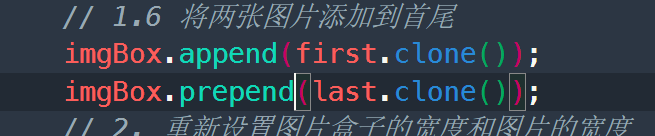
*// 2.1 获取所有的li元素*

*// 2.2 获取li元素的个数*

*// 2.3 设置图片盒子的宽度*

*// 2.4 设置所有li标签的宽度*

*// 2.5 设置图片盒子的默认偏移值*



1. **zepto的使用总结：**
   1. 会jquery就会使用zepto
   2. zepto分模块，如果只引入zepto.js只能用它的五个核心模块，如果要使用其他的功能需要引入其他的js文件
2. **zepto实现轮播图效果**
   1. **获取对象 (获取ul:eq(1)需要引入 selector.js)**
   2. **首尾各添加一张图片**

imgBox.append(first.clone());  
last.clone().insertBefore(first);

**c) 设置图片盒子的宽度和li盒子的宽度以及位偏移**

**var** lis=imgBox.find("li");  
**var** count=lis.length;  
imgBox.width(count\*bannerWidth);  
lis.each(**function**(index,value){  
$(lis[index]).width(bannerWidth);  
});  
imgBox.css("left",-bannerWidth);

## 使用zepto实现轮播图(中)

*// 3. 自动轮播*

*// 3.1 设置index索引，初始值为1*

*//3.2 开启定时器*

*//3.3 索引值自增*

*// 3.4 在zepto中直接使用animate函数来实现*

*//3.5 在回调函数中处理临界情况*

*//3.6 在回调函数中考虑轮播图点标记*

1. **animate()方法中一些属性的理解**
   1. 调用动画animate()方法 需要引入fx.js
   2. 需要添加动画效果的 对象
   3. 动画需要耗的时长
   4. 动画的速度
   5. 动画完成后执行的回调函数
      1. 判断当前的索引是否等于最后一张(index==count)还是第一张
      2. 当前点添加active类名

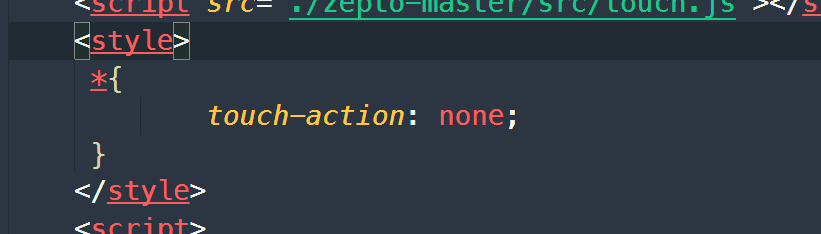
**var** imgAnimation=**function**(){  
 imgBox.animate(  
{"left":-index\*bannerWidth}, 执行动画效果的对象  
200,动画所需要的时长

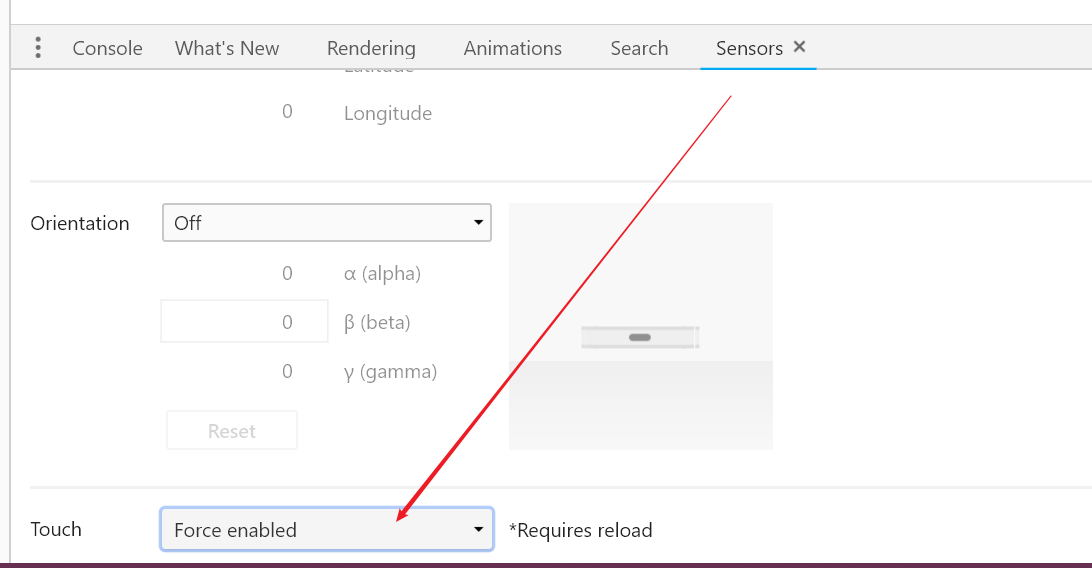
linear, 动画的速度(不写默认的是linear代表匀速)  
**function**(){ 动画完成后执行的回调函数  
**if**(index==count-1){ //最后张  
index=1;  
imgBox.css("left",-index\*bannerWidth);  
}**else if**(index==0){  
 index=count-2;  
imgBox.css("left",-index\*bannerWidth);  
}

indicators.removeClass("active").eq(index-1).addClass("active");  
}  
 );  
}

## 使用zepto实现轮播图(下)

如何解决chrome模拟器的的修复问题：





*// 4.1 封装动画函数*

*// 4.2 左右滑动（经测试遨游也用不了了）的时候清掉定时器 --> 自增、自减索引值 --> 执行动画函数*

<https://github.com/madrobby/zepto/issues/315#issuecomment-8386027>

这个案例视频中还没有做完，没有考虑添加开关之类的，这块大家有兴趣，可以在原生的写出来之后，自己再来用zepto尝试着写一下，参考代码如下：

|  |
| --- |
| <script>  $(function(){  var banner = $('.jd\_banner');  var imgBox = banner.find('.jd\_bannerImg');  var indicator = banner.find('.jd\_bannerIndicator');  var first = imgBox.find('li:first-of-type');  var last = imgBox.find('li:last-of-type');  imgBox.append(first.clone());  imgBox.prepend(last.clone());  var lis = imgBox.find('li');  var count = lis.length;  var bannerWidth = banner.width();  imgBox.width(count \* bannerWidth);  lis.width(bannerWidth);  imgBox.css('left',-bannerWidth);  var index = 1;  var isEnd = true;  function animateFn(){  imgBox.animate({ 'left': - index \* bannerWidth}, 500, 'ease-in-out', function () {  if (index == count - 1) {  index = 1;  imgBox.css('left', -index \* bannerWidth);  }  if(index == 0){  index = count - 2;  imgBox.css('left', -index \* bannerWidth);  }  isEnd = true;  clearInterval(timerId);  timerId = setInterval(function () {  index++;  animateFn();  }, 2000);  })  }  var timerId = setInterval(function(){  if (isEnd == false) return;  isEnd = false;  index++;  animateFn();  },2000);  imgBox.on('swipeLeft',function(){  if (isEnd == false) return;  isEnd = false;  clearInterval(timerId);  index++;  animateFn();  })  imgBox.on('swipeRight',function(){  if (isEnd == false) return;  isEnd = false;  clearInterval(timerId);  index--;  animateFn();  })  })  </script> |

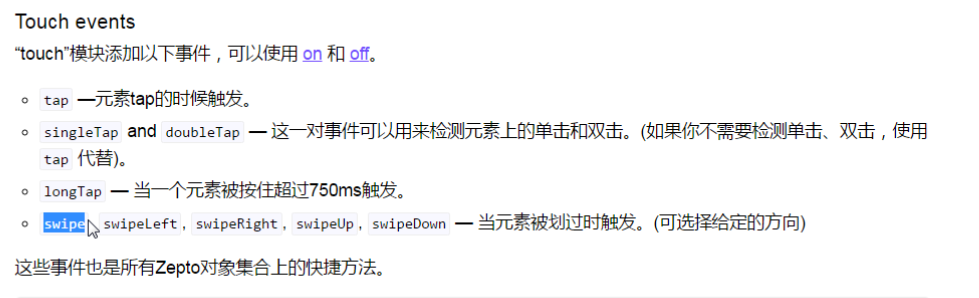
Zepto 的 'swipe', 'swipeLeft', 'swipeRight', 'swipeUp', 'swipeDown' 触摸事件在安卓4.4系统中除chrome + 火狐 之外的 浏览器 + 微信 都无效



|  |
| --- |
| .on('touchmove MSPointerMove pointermove', function(e){  if((\_isPointerType = isPointerEventType(e, 'move')) &&  !isPrimaryTouch(e)) return  firstTouch = \_isPointerType ? e : e.touches[0]  cancelLongTap()  touch.x2 = firstTouch.pageX  touch.y2 = firstTouch.pageY  deltaX += Math.abs(touch.x1 - touch.x2)  deltaY += Math.abs(touch.y1 - touch.y2)  /\*\*  \* 修复 android 4.4 swipe 事件  \* https://github.com/madrobby/zepto/issues/315#issuecomment-8386027  \*/  if (touch.x2 && Math.abs(touch.x1 - touch.x2) > 10)  e.preventDefault()  }) |

1. **如何实现触摸滑动**

用zepto其他模块所提供的on和off事件，



1. **步骤分析**
   1. **左滑动 在事件中传入方向(swipeLeft)** 
      1. 滑动的时候清除定时器
      2. 左滑就是让索引自增
      3. 让图片做动画同时需要判断当前属于第几张 (这里直接把刚才的动画函数给封装起来了)
   2. **右滑动 在事件中传入方向(swipeRight)** 
      1. 滑动的时候清除定时器
      2. 右滑就是让索引自减
      3. 让图片做动画同时需要判断当前属于第几张 (这里直接把刚才的动画函数给封装起来了)

/\*添加滑动事件\*/  
/\*左滑动\*/  
/\*在谷歌浏览器的模拟器中，无法正确的触发swipe相关事件，但是可以触发tap事件\*/  
imgBox.on("swipeLeft",**function**(){  
clearInterval(timerId);  
index++;  
imgAnimation();  
});  
/\*右滑动\*/  
imgBox.on("swipeRight",**function**(){  
clearInterval(timerId);  
index--;  
imgAnimation();  
});

## zepto的定制

会用就行了，原理这块后面学node的时候再说

**由于大家没有学过node.js,虽然视频中的操作方式比较笨，但还是尽量按视频中的来操作**

**步骤分析：**

zeptojs为我们封装了常用的触屏事件，需要touch模块支持，默认没有构建此模，我们可以自定义构建。

1、安装Nodejs环境

2、下载[zepto.js](https://github.com/madrobby/zepto)

3、解压缩

4、cmd命令行进入解压缩后的目录

菜单node>node.js command

cd+文件夹路径C:\Users\silence\Desktop\伍湖-移动web备课\zepto-master\zepto-master

5、执行npm install 命令

6、编辑make文件，添加自定义模块并保存，如下图



7、然后执行命令 npm run-script dist

8、查看目录dist即构建好的zepto.js

# 分类页面布局

## 分类页面结构的简单说明

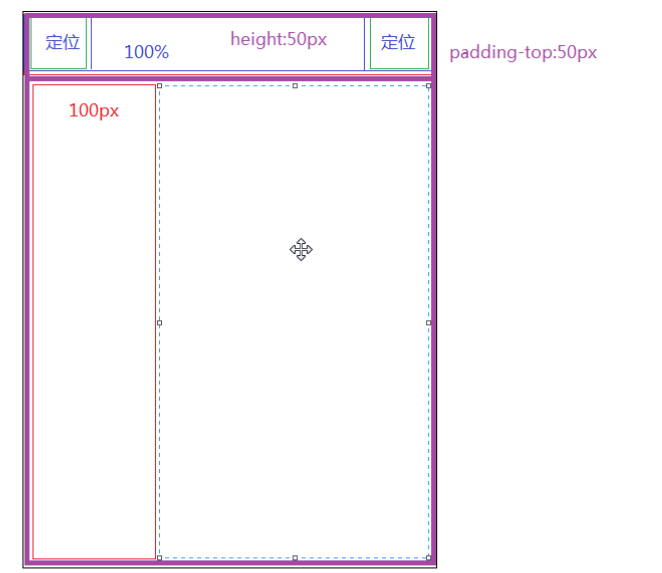
.ct\_layout>(.ct\_header>.ct\_back+.ct\_search+ct\_menu)+(ct\_content>ct\_cLeft+.ct\_cRight)



1.完成全屏页面的制作

2.回顾touch事件，使用touch事件实现左侧栏的滑动，同时使用touch事件来实现移动端的点击事件--重点

1. 使用第三方插件来实现右侧商品信息块的滚动操作



## 分类页面的全屏结构创建

1. **什么是全屏页面？**

内容的宽和高不要超出父容器(或者说内容的宽和高不要超出视口或屏幕)

1. **全屏页面怎么实现？**

/\*全屏页面的实现\*/  
html,body{width: 100%;height: 100%;}

1. **左边和右边怎么设置**

**左边和右边都是全屏布局，左边浮动，右边设置一个margin-left**

## 分类页面顶部搜索块的制作

1. **精灵图的显示用到哪两个属性？**
   1. background-clip:content-box 以内容来显示
   2. background-origin::content-box以内容来定位
2. **精灵图尺寸的设计**

二倍图的原理，直接背景尺寸为原图的一半

1. **form自适应的原理**

宽度百分比，设置padding会随着屏幕做自适应因为box-sizing:border-box

## 分类页面左侧导航块的制作

超出就设置overflow:hidden; 同时左侧是相对定位

## 分类页面右侧商品块的制作

1. **如何让右边盒子不超出屏幕的高度**
   1. 右盒子设置开启弹性布局 + flex-direction(改变主轴的方向)
   2. 给图片盒子设置flex:1

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0,user-scalable=no,maximum-scale=1.0,minimum-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <style>  \*{  margin:0;  padding:0;  }  html,body{  width: 100%;  height: 100%;  }  .wrapper{  height: 100%;  display: flex;  flex-direction: column;  }  .header img{  width: 100%;  vertical-align: top;  }  .content{  background: pink;  width: 100%;  flex: 1;  }  </style>  </head>  <body>  <div class="wrapper">  <div class="header">  <img src="./uploads/l1.jpg" alt="">  </div>  <div class="content">  </div>  </div>  </body>  </html> |

# 分类页面js效果

## 初步实现左侧栏的滑动

*// 1. 获取左侧栏*

*// 2. 获取用来滑动的列表*

*// 3. 初始化startY,moveY,distanceY,currentY*

*// 4. 设置touchstart,touchmove,touchend事件*

*// 5. 在touchstart的时候获取手指的起始坐标*

*//6.在touchmove的时候获取手指的坐标*

*// 7. 计算滑动开始与滑动的时候的距离的差异*

*// 8. 实现偏移操作*

*// 9. 在滑动结束的时候记录当前滑动的距离*

**1、基本滑动原理？**

通过touch事件来实现滑动效果，但每次滑动都是根据上一次的参考点来作相应的滑动

|  |
| --- |
| window.onload = function(){  // 获取左侧栏  var ct\_cLeft = document.querySelector('.ct\_cLeft');  // 获取用来滑动的列表  var ulBox = ct\_cLeft.querySelector('ul:first-of-type');  // 实现滑动  var startY = 0;  var moveY = 0;  var distanceY = 0;  var currentY = 0;  // 添加滑动事件  ulBox.addEventListener('touchstart',function(e){  // 获取手指的起始坐标  startY = e.targetTouches[0].clientY;  })  ulBox.addEventListener('touchmove',function(e){  moveY = e.targetTouches[0].clientY;  // 计算距离的差异  distanceY = moveY - startY;  // 实现偏移操作  ulBox.style.transition = 'none';  ulBox.style.top = currentY + distanceY + 'px';  })  ulBox.addEventListener('touchend',function(){  // 记录当前滑动的距离  currentY += distanceY;  })  } |

## 设置滑动区间

*//1. 获取左侧栏的高度(leftHeight)、左侧ul的高度(ulBoxHeight)*

*//2. 设置静态下的最小值minTop 最大值maxTop*

*//3. 设置动态下的最小值minBounceTop 最大值maxBounceTop*

*// 4. 判断是否超过最大值，最小值，如果是，则return*

**1、静止区间和滑动区间分别是哪里？**

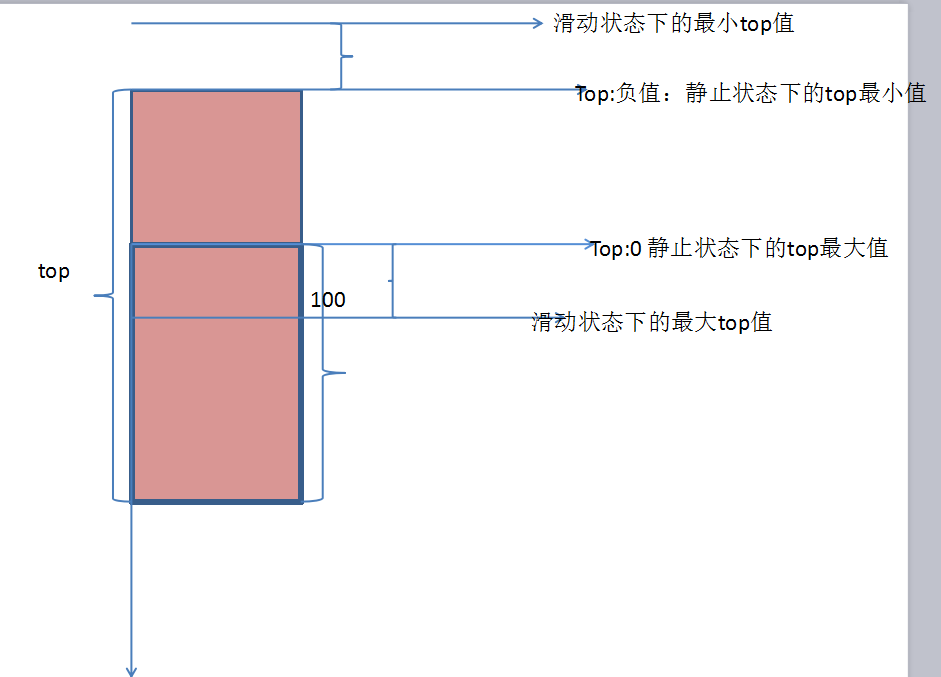
**静止区间**：初始存在一个最大静止值为0，向上滑动其值为负，也会存在一个最小的静止值（父容器高度-内容的高度）

**滑动区间**：最大滑动值：最大静止值+100(可向下滑动的距离) 最小滑动值：最小静止值-100(可向上滑动的距离)

**2、如何判断是否在滑动区间滑动**

在touchmove中，如果超出滑动最大值或者小于最小滑动值，就直接终止

**if**(currentY+distanceY > maxBounceTop || currentY+distanceY < minBounceTop){  
console.log("超出范围啦");  
**return**;  
}



## 松开手指之后的处理

1. **如何设置最终静止区间？**
   1. 当大于最大静止值是让其最终定位在最大静止值处
   2. 当小于最大静止值是让其最终定位在最小静止值处
   3. 当在最大和最小这个区间时 就直接让其定位在当前滑动的位置处

**if**(currentY+distanceY < minTop){  
 currentY=minTop;重置当前的currentY  
ulBox.style.transition="top 0.5s";  
ulBox.style.top=minTop+"px";小于最小静止值直接等于最小值  
}**else if**(currentY+distanceY > maxTop){  
 currentY=maxTop;重置当前的currentY  
ulBox.style.transition="top 0.5s";  
ulBox.style.top=maxTop+"px";大于最小静止值直接等于最大值  
}**else**{  
currentY+=distanceY; 重置当前的currentY

在最大和最小静止值之间直接等于当前滑动的的距离加当前的定位

}

1. **滑动出现闪动的原因？**

每次滑动之后当前的currentY没有重置

# tap事件封装及使用

## cilck等操作在移动端的延迟

1. **click事件为什么会有300毫秒延迟？**

因为双击会对其页面进行放大，当用户单击一个按钮时，移动设备会延时（约300ms）执行，判断用单是否要双击

1. **移动端哪个事件执行比较快？**

touch事件执行的快

**在移动端为什么使用touch事件而不使用click事件**

## 移动端的tap事件

1. **tap事件封装的条件？**

\* 1.单击只有一根手指  
\* 2.判断手指开始触摸和手指松开的时间差异不能大于指定的值 300/150  
\* 3.保证没有滑动操作，如果有抖动必须保证抖动的距离在指定范围内\*/

1. **touchend怎么获取到坐标值？**

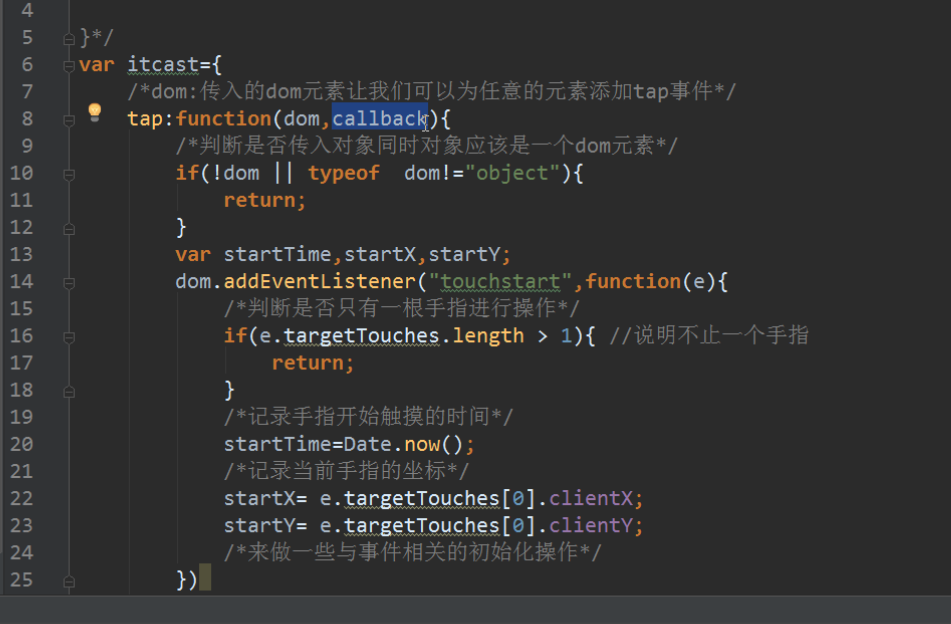
**通过changetouch来获取(touchend为离开触发，这是targetTouch就不会触发)**



## 移动端tap事件的封装

**步骤分析：**

* 1. 通过一个itcast的对象 对tap事件的封装 执行一个函数传入要执行这个事件的对象和回调函数
  2. 对传入的对象进行一个判断
  3. 就是刚才的事件触发(touchstart和touchend)直接调用
  4. 在touchend事件判断如果是移动端就执行这个回调函数





## 分类页面tap事件的添加(上)

1. **移动端tap事件的使用**
   1. 调用itcast.tap() 传入对象和要执行的的回调函数
   2. 给每个li移除样式给当前的添加active



## 分类页面tap事件的添加(下)

**步骤分析**

**1、获取每个li的索引**

/\*为每一个li元素设置添加一个索引值\*/  
**for**(**var** i=0;i<lis.length;i++){  
/\*lis[i].setAttribute("index",i);\*/  
lis[i].index=i;  
}

1. **给当前的li进行定位**
   1. 如果最终的偏移值小于最小静止值就直接等于这个最小值，反之就直接等于偏移值
   2. 完后最终的值需要对currentY重置不然还会出现异常

昨日知识点回顾：

01、京东页面轮播图的完善及完成

02、zepto实现京东轮播图+zepto的定制

03、分类页面全屏布局+结构创建

04、分类页面左侧实现滑动

05、移动端tap事件对tap事件封装及使用

今日知识点：

01、使用zepto实现左侧点击操作

02、移动端touch事件点透问题+fastclick插件的使用

03、iscroll插件的使用

04、swipe和swiper的插件的使用

05、响应式布局

# 1 使用zepto实现左侧点击操作

## 1.1 01-移动web-使用zepto来实现左侧单击操作

1. **如何使用zepto来代替封装好的tap事件**
   1. 把我们之前写的调用封装函数给注释掉

itcast.tap(ulBox,**function**(e){

* 1. 把里面的执行回调函数通过zepto的方法来执行

$(ulBox).on("tap",**function**(e){

* 1. 把之前引入的common.js注释掉，引入我们的zepto.min.js 就可实现zepto左侧单机操作

# 2 移动端touch事件点透问题+fastclick插件的使用

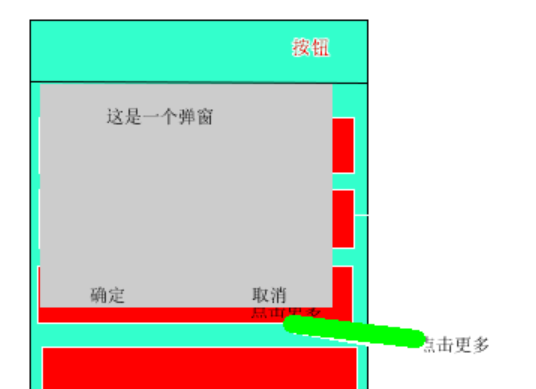
## 2.1 02-移动web-移动端touch事件所产生的点透问题

如果测不出来，可以尝试着用隐身模式来测试一下

1. **什么是点透**

当我们触发当前的事件时会导致跟它相关的对象也会触发绑定在他们身上的的事件就是点透。下面用例子和图展示一下

点击按钮会出现一个弹窗，弹窗上面有一个确定按钮和取消按钮 ，当我们点击取消按钮的时候，同时也会触发点击更多这个按钮，这个效果我们就叫做点透 (详图如下)



## 2.2 03-移动web-fastclick的使用

**1、touch和click在解决点透上有什么弊端？**

* 1. touch 只能在移动端使用 pc端会无效
  2. touch 会产生点透
  3. click 不会产生点透,但是会有时间延迟

**2、怎么通过fastclick解决点透问题**

* 1. **引入fastclick.js文件**
  2. **通过js代码**

**if** ('addEventListener' **in** document) {  
document.addEventListener('DOMContentLoaded', **function**() {  
/\***参数可以是任意的dom元素，如果写document.body，说明会将document.body下面的所的元素都绑定fastclick**\*/  
FastClick.attach(document.body);  
}, **false**);  
}

**或者jquery**

$(**function**() {  
FastClick.attach(document.body);  
});

* 1. **在对我们需要的对象绑定事件**

tap.addEventListener("click",**function**(){

**此处的click是fastclick.js封装好的click**  
tap.style.visibility="hidden";  
})  
click.addEventListener("click",**function**(){  
console.log(123);  
})

## 2.3 04-使用fastclick实现分类页面左侧栏的点击操作

1. 如何使用fastcick来替代之前tap实现的点击切换
   1. 引入fastclick.js文件
   2. 替换原始tap事件之前的代码

$(ulBox).on("tap",**function**(e){  
/\*1.修改li元素的样式：将所有li元素的active样式清除，再为当前被点击的li元素添加active样式\*/  
**for**(**var** i=0;i<lis.length;i++){  
 lis[i].classList.remove("active");  
}  
/\*为当前被单击的li元素添加样式\*/  
**var** li=e.target.parentNode;  
**var** liHeight=li.offsetHeight;  
li.classList.add("active");  
/\*2.移动当前的li元素到父容器的最顶部，但是不能超出之前设定了静止状态下的最小top值\*/  
 /\*获取当前li元素的索引值\*/  
**var** index=li.index;  
/\*开启过渡\*/  
ulBox.style.transition="top .5s";  
/\*设置偏移\*/  
**if**(-index\*liHeight < minTop){  
/\*只能偏移到minTop位置\*/  
ulBox.style.top=minTop+"px";  
currentY=minTop;  
}  
**else**{  
 ulBox.style.top=-index\*liHeight+"px";  
currentY=-index\*liHeight;  
}  
});

* 1. **用fastclick替换后**

/\*绑定fastclick\*/  
**if** ('addEventListener' **in** document) {  
document.addEventListener('DOMContentLoaded', **function**() {  
/\*参数可以是任意的dom元素，如果写document.body，说明会将document.body下面的所的元素都绑定fastclick\*/  
FastClick.attach(document.body);  
}, **false**);  
}  
/\*fastclick使用的时候就是来绑定添加click事件\*/  
ulBox.addEventListener("click",**function**(e){  
/\*1.修改li元素的样式：将所有li元素的active样式清除，再为当前被点击的li元素添加active样式\*/  
**for**(**var** i=0;i<lis.length;i++){  
 lis[i].classList.remove("active");  
}  
/\*为当前被单击的li元素添加样式\*/  
**var** li=e.target.parentNode;  
**var** liHeight=li.offsetHeight;  
li.classList.add("active");  
/\*2.移动当前的li元素到父容器的最顶部，但是不能超出之前设定了静止状态下的最小top值\*/  
 /\*获取当前li元素的索引值\*/  
**var** index=li.index;  
/\*开启过渡\*/  
ulBox.style.transition="top .5s";  
/\*设置偏移\*/  
**if**(-index\*liHeight < minTop){  
/\*只能偏移到minTop位置\*/  
ulBox.style.top=minTop+"px";  
currentY=minTop;  
}  
**else**{  
 ulBox.style.top=-index\*liHeight+"px";  
currentY=-index\*liHeight;  
}  
})

# 3、iscroll插件的使用

## 3.1 05-移动web-iscroll的使用说明(上)

1. **iscrolls 是什么？**

一个可以实现客户端原生滚动效果的类库。

## 3.2 06-移动web-iscroll的使用说明(下)

1. **Iscroll基本使用的结构：**

<div id="wrapper">  
<ul>  
<li>...</li>  
<li>...</li>  
 ...  
</ul>  
</div>

<script src="./js/iscroll.js"></script>  
<script type="text/javascript">  
**var** myScroll = **new** IScroll('.ct\_hotCategory',{  
mouseWheel: **true**,鼠标滚动也可以实现滑动  
scrollbars: **true添加滚动条**});  
</script>

1. **滚动条位置如何调整**

**只需要给id为wrapper的标签元素添加一个position:relative属性就可以实现**

# 4 swipe和swiper的插件的使用

## 4.1 07-移动web-swipe插件的使用

1. **加visibility:hidden的作用**

不加就会加载很慢会让图片都显示出来(呈现先伸开后隐藏)影响用户体验

1. **在加入visibility:hidden后如何让图片滑动轮播**

window.mySwipe = Swipe(document.querySelector(".jd\_banner"));

1. **其他参数的设置和了解**



1. **设置上一张和下一张**

document.getElementById("prev").onclick=**function**(){  
window.mySwipe.prev();**上一张**}

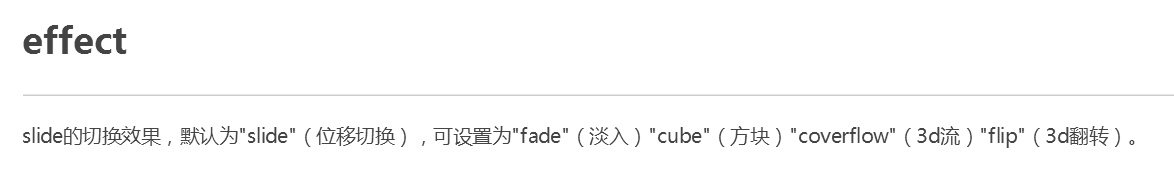
document.getElementById("next").onclick=**function**(){  
window.mySwipe.next();**下一张**}

## 4.2 08-移动web-swiper插件的使用介绍

**var** mySwiper = **new** Swiper('.swiper-container',{  
effect : 'cube',  
cube: {  
slideShadows: **true**,  
shadow: **true**,  
shadowOffset: 100,  
shadowScale: 0.6  
}  
})

参数的了解(也可通过文档详细查阅)





slideShadows：开启slide阴影。默认 true。  
shadow： 开启投影。默认 true。  
shadowOffset：投影距离。默认 20，单位px。  
shadowScale： 投影缩放比例。默认0.94。

## 4.3 09-使用iscroll插件完成分类页面右侧内容块的滑动操作

**步骤分析：**

1. 引入iscroll.js文件
2. 引入iscroll.js代码

<script src="./js/iscroll.js"></script>  
<script type="text/javascript">  
**var** myScroll = **new** IScroll('.ct\_hotCategory',{  
mouseWheel: **true**,  
scrollbars: **true**});  
</script>

1. 设置当前对象有滚动条，给当前对象加一个 position:relative

# 5 响应式布局

## 5.1 10-移动web-网页布局方式

1. **网页布局有几种方式？（红色加粗为即将要学的重点布局）**

a、固定宽度布局：为网页设置一个固定的宽度，通常以px做为长度单位，常见于PC端网页。

b、流式布局：为网页设置一个相对的宽度，通常以百分比做为长度单位。

c、**栅格化布局**：将网页宽度人为的划分成均等的长度，然后排版布局时则以这些均等的长度做为度量单位，通常利用百分比做为长度单位来划分成均等的长度。

d、**响应式布局**：通过检测设备信息，决定网页布局方式，即用户如果采用不同的设备访问同一个网页，有可能会看到不一样的内容，一般情况下是检测设备屏幕的宽度来实现。

注：以上几种布局方式并不是独立存在的，实际开发过程中往往是相互结合使用的。

## 5.2 11-移动web-响应式布局的简单说明

1. **什么是响应式布局？**

屏幕尺寸不一样展示给用户的网页内容也不一样

1. **响应式布局是如何实现的？**

利用媒体查询可以检测到屏幕的尺寸（主要检测宽度），并设置不同的CSS样式，就可以实现响应式的布局。

1. **响应式开发的优缺点**
   1. 一个页面可适配不同的终端
   2. 产生代码体积过大各种终端的资源都会存在加载
2. **常见的设备尺寸分类 (做专门的布局设计区分)**



1. **常见的设备尺寸对应的媒体查询条件**

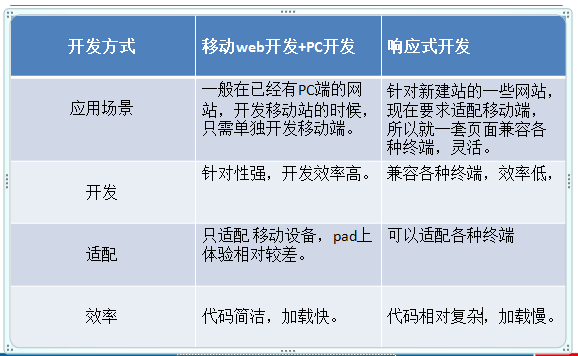


## 5.3 12-移动web-什么是响应式开发

**1、什么是响应式开发**

页面在不同的终端做不同的屏幕适配显示不同的页面效果

**2、移动开发和响应式开发的区别**



## 5.4 13-移动web-初体验媒体查询的语法

**1、响应式开发的原理：媒体查询：（**重点**）**

* 1. 查询媒介，查询到当前屏幕(媒介媒体)的宽度，针对不同的屏幕宽度设置不同的样式来适应不同屏幕。当你重置浏览器大小的过程中，页面也会根据浏览器的宽度和高度重新渲染页面。简单说，你可以设置 不同屏幕下面的不同的样式，达到适配不同的屏幕的目的。
  2. 实现方式：通过查询screen的宽度来指定某个宽度区间的网页布局。

超小屏幕 （移动设备） w<768px

小屏设备 768px-992px 768 <= w <992

中等屏幕 992px-1200px 992 =< w <1200

宽屏设备 1200px以上 w>=1200

1. 媒体查询语法 (**注意：空格不能省略，否则无法生效**)

@media screen and (max-width: 768px){  
body{  
background-color: green;  
}  
}  
@media screen and (min-width: 768px) and (max-width: 992px){  
body{  
background-color: blue;  
}  
}

## 5.5 14-移动web-媒体查询的条件判断的顺序说明(上)

**1、向上兼容和向下覆盖的原则？**

1.向上兼容：如果设置了宽度更小时的样式，默认这些样式也会传递到宽度更大的条件范围内 (宽度向上找，直到找到宽度最大的那个)  
2.向下覆盖：宽度更大的样式会将前面宽度更小的样式覆盖\*

## 5.6 15-移动web-媒体查询的条件判断的顺序说明(下)

1、如果判断max-width该怎么书写规则？

1.如果是判断最小值 (min-width)，那么就应该从小到大写--bootstrap就是判断最小值，从小到大写  
2.如果是判断最大值 (max-width)，那么就应该从大到小写

## 5.7 16-移动web-媒体查询中的媒体功能的说明

**1、min-width和min-device-width的区别**

1. min-width:在pc端和移动端能正常的响应，效果一致 。它是指当前可视区域的宽度

2、min-device-width:在移动端和期望值一样 device：设备。它是指当前设备的宽度。当拖动来改变浏览器大小的时候，当前终端设备的宽度并不会变化，意味着媒体查询条件不会响应

## 5.8 17-移动web-媒体查询补充说明

**1、如何在不同屏幕调用不同样式？**

**当屏幕尺寸宽度在992-1200就调用b.css ,不在这个范围就调用a.css**

<link rel="stylesheet" href="a.css">  
<link rel="stylesheet" media="screen and (min-width:992px) and (max-width:1200px)" href="b.css">

**2、not的使用**

**屏幕的尺寸不在768-992，body就显示背景颜色blue**

**屏幕在这个尺寸范围就显示红色**

<style>  
body{background-color: red;}  
@media notscreen and (min-width: 768px) and (max-width: 992px){  
body{  
background-color: blue;  
}  
 }  
</style>

## 5.9 18-移动web-使用媒体查询设置页面布局

通过媒体查询重置在不同尺寸下重置页面结构的样式

## 5.10 19-移动web-常见的响应式框架简单介绍





对于这类框架我们都是按照他所提供好的文档来学习，这时就需要同学们有一定的自学能力

## 5.11 20-移动web-bootstrap简单介绍

**1、bootstrap的特点**

简洁、直观、强悍的前端开发框架，让web开发更迅速、简单。

**提示：**本地官方页面：1-教学资料\bootcss\v3.bootcss.com\index.html

## 5.12 21-移动web-bootstrap简单介绍（2）

1. **要兼容media query需引入 respond.js**

为了让 IE 浏览器运行最新的渲染模式下  
<meta http-equiv="X-UA-Compatible"content="IE=edge">

## 5.13 22-移动web-bootstrap的模板文件

**使用框架前需要引入的一些文件，掌握需要引入哪些文件即可**

cc：ie6+tab

<!--  
 l less  
 t than  
 e equal  
 lte 小于等于  
-->

昨日知识点回顾：

01、使用zepto实现左侧点击操作

02、移动端touch事件点透问题+fastclick插件的使用

03、iscroll插件的使用

04、swipe和swiper的插件的使用

05、响应式布局

今日知识点：

01、bootstarp布局容器+栅格系统的使用

02、less环境搭建和语法说明

03、wjs页面结构搭建+wjs页面头部的完成

04、bootstarp导航组件说明及wjs导航的完成

05、轮播图效果的实现

# 1 bootstarp布局容器+栅格系统的使用

## 1.1 01-移动web-bootstrap中的布局容器

1. **container和container-fluid的区别？**
   1. **container (在不同屏幕尺寸container显示不同的一个默认宽度)**

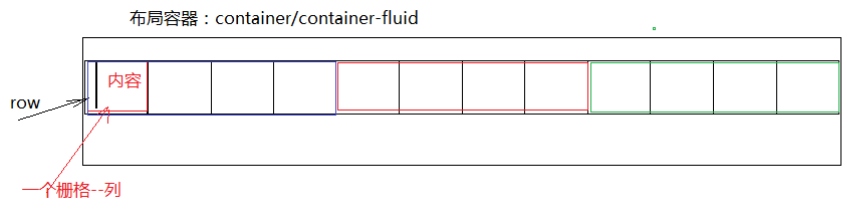
w>1200:1170  
w:992~1200:970  
w:768~992:750  
w<768:100%

**b) container-fluid 实时显示全屏 即宽度100%**

## 1.2 02-移动web-bootstrap中的栅格系统

**1、什么是栅格系统**

**栅格系统**就是来描述当前子元素 在指定屏幕下 占据当前屏幕默认12等分中的n（1-12）等分



1. **如何创建一个栅格系统**
   1. 先添加布局容器container/container-fluid
   2. 在布局容器中添加row
   3. 在row中添加子元素，设置栅格样式
   4. 在子元素中添加具体内容
2. **栅格系统类名的了解**
   1. xs 移动端： w<768 col-xs-3 （表示xs屏及以上屏占3等份） 宽度为33.33%
   2. sm 平板： 768-992 col-sm-3 （表示sm屏及以上屏占3等份） 宽度为33.33%
   3. md 小屏pc: 992-1200 col-md-3 （表示md屏及以上屏占3等份） 宽度为33.33%
   4. lg 大屏pc: w>1200 col-lg-3 （表示lg屏及以上屏占3等份） 宽度为33.33%

## 1.3 03-移动web-bootstrap栅格系统的其它样式说明(上)

1. **offset和push及pull的区别**

**offset:偏移**偏移的距离是以栅格做为单位 但是这种偏移会直接影响后面的元素， 有可能造成后面的元素换行显示 因为它是通过margin-left属性来实现的

**push(往后-右推)/pull(往前-左拉)**它是通过定位来实现的，所以不会影响其它元素， 但是有可能和其它元素重叠

## 1.4 04-移动web-bootstrap栅格系统的其它样式说明(下)

1. **栅格嵌套时候container的问题**

1.如果在外层没有再包含container,那么嵌套列的宽度就是参参照当前所在的栅格  
2.如果外层添加了container,那么参照就是核心样式文件所设置的容器宽度

**提示：如果加了container ,container的宽度会参照一下取值**

w>1200:1170  
w:992~1200:970  
w:768~992:750  
w<768:100%

1. **响应式工具的使用**
   1. **hidden-xs/hidden-sm/hidden-md/hidden-lg 分别在不同的屏幕尺寸下会隐藏**

# 2 less环境搭建和语法说明

## 2.1 05-移动web-less环境的搭建

1. **less是干什么的？**

Less 是动态的样式表语言，通过简洁明了的语法定义，使编写 CSS 的工作变得非常简单，本质上，less 包含一套自定义的语法及一个解析器。

1. **less如何搭建？**
   1. **安装node** 
      1. **通过命令行：win键+R > 输入cmd**
      2. **输入：node -v (检测node是否安装成功)**
      3. **输入：npm -v (检测node packag manager是否安装成功)**

**b) 安装less**

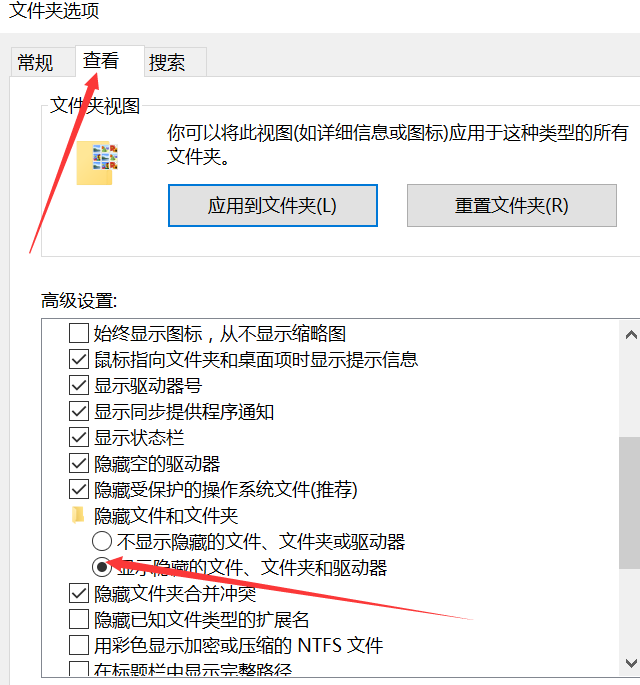
1. **在线安装 直接输入: npm install -g less (检验：lessc -v)**
2. **离线安装**

**i. 复制教学资料npm中的文件**

**ii. C盘 > 用户 > 自己起的电脑名 (如我的叫silence) > AppData > Roaming > npm (再把刚才复制的文件粘贴在npm文件中)**

**Iii. lessc -v检测是否安装成功**

**提示：**如果APPData是隐藏的，C>用户>当前用户目录>查看>选项



**3、webstorm中less文件的解析**

a.新建一个less文件

b在less文件里面建一个less.less

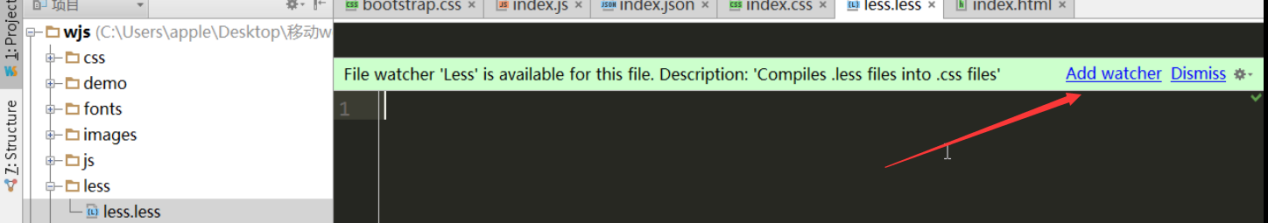
**i.命令行手动解析**less.less

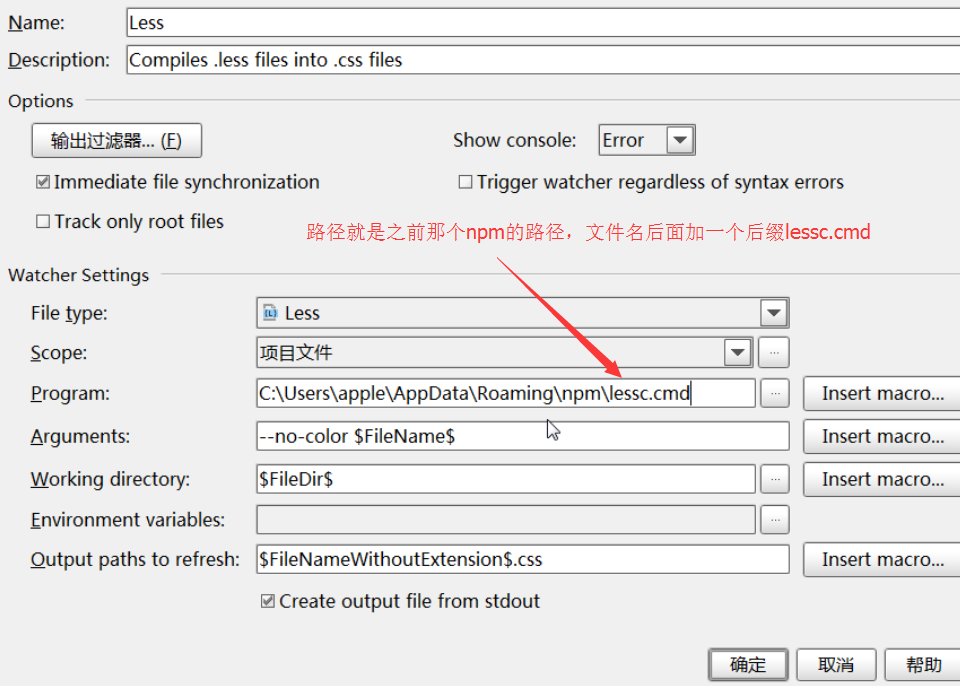
找到刚才创建的less.less >在命令行中输入cd less.less在项目中的路径 按下回车>检测是否正确输入：dir /w 》

在输入预编译less.css : **lessc less.less less.css**

**ii.编译器里面解析** text.less

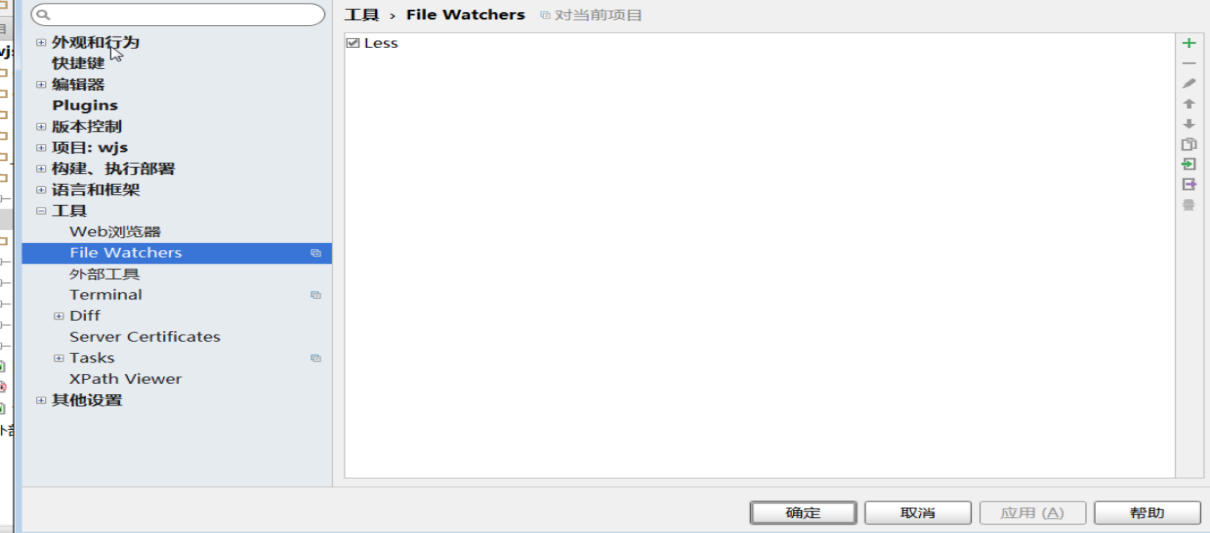
1》.出现如下提示点击add watch





2》检查是否添加好less

Setting>Tools>file watch (如下图)



**以上还是没有解析出来重新启动编译器**

## 2.2 06-移动web-less语法说明

**1、less.less注释：**

1. less中 // 在less.css中看不到
2. less中 /\* \*/ 在less.css中可以看到
3. **变量**
   1. @变量名：值； 例如：#aa:red; a{color:@aa}
4. **混入 (可以将一个样式引入到另外一个样式中)**
   1. .add{border-radius:20px}

.abc{height:100px ;.add}

* 1. .add(@aa){border-radius:@aa}

.abc{.add(200px)}

* 1. .add(@aa:10px){border-radius:@aa}

.abc{.add()} 此时不传值就是默认的 .abc{border-radius:10px}

.abc{.add(5px)} 此时传值 .abc{border-radius:5px}



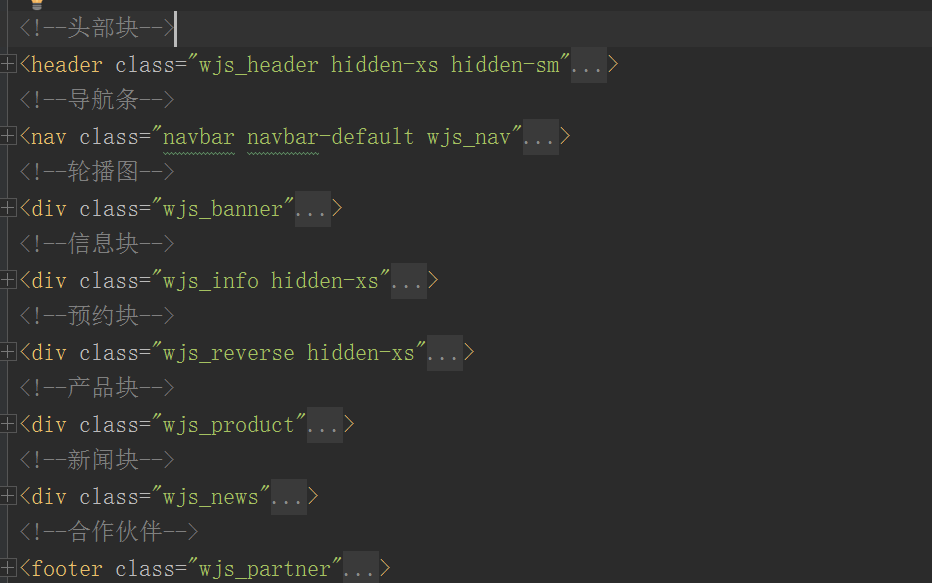
1. **嵌套**



# 3 wjs页面结构搭建+wjs页面头部的完成

## 3.1 07-移动web-wjs页面的整体结构

整体结构是分成了八大快



## 3.2 08-移动web-微金所页面的字体图标样式准备

@font-face {  
font-family: 'wjs';  
src: url('../fonts/MiFie-Web-Font.eot'); /\* IE9\*/  
src: url('../fonts/MiFie-Web-Font.eot') format('embedded-opentype'), /\* IE6-IE8 \*/  
url('../fonts/MiFie-Web-Font.woff') format('woff'), /\* chrome、firefox \*/  
url('../fonts/MiFie-Web-Font.ttf') format('truetype'), /\* chrome、firefox、opera、Safari, Android, iOS 4.2+\*/  
url('../fonts/MiFie-Web-Font.svg') format('svg'); /\* iOS 4.1- \*/  
}  
/\*自定义字体使用样式\*/  
.wjs\_icon{  
font-family: wjs;  
}

## 3.3 09-移动web-微金所头部块制作(上)

1. 页面的布局：栅格系统
2. 按钮调用框架提供好的 (框架提供位置 : 全局样式>按钮)

## 3.3 10-移动web-微金所头部块制作(中)

**步骤分析：**

1. less来编写css (先安装好)
2. 对容器设置高度和居中+底边框
3. 对前三个列加右边框 nth-of-type(-n+3)
4. 对图片定位,鼠标靠近就显示加边框

## 3.4 11-移动web-微金所头部块制作(下)

1. **如何修改bootstarp预定制样式？**

**找到具体的标签样式元素 在类名前面加上当前块类名 在重新更改它的属性**

# 4 bootstarp导航组件说明及wjs导航的完成

## 4.1 12-移动web-导航条组件的说明

1. 组件 > 导航条 可以找到我们类似的导航栏
2. 结构的选择( 选择需要的内容)

## 4.2 13-移动web-导航条组件的样式修改(上)



**1、导航条样式如何修改？**

* 1. 找到源文件中的样式 navbar-default，复制到当前页面的css文件中(wjs\_nav.css) 但这样会加大代码量
  2. 把原有的navbar-default用wjs\_nav替换，再重新定义新的属性，也就是重置覆盖框架所提供的css样式

## 4.3 14-移动web-导航条组件的样式修改(下)

**修改步骤：**

1. 鼠标靠近给hover,focus加底边框，
2. 给有active类名的加底边框，背景透明
3. Logo更换颜色和大小
4. 移动端就隐藏我要投资等导航栏
5. 更改移动端菜单样式

# 5 轮播图效果的实现

## 5.1 15-移动web-轮播图的效果分析

**1、移动端和pc端的机构**

w<768px-移动端：图片会随着屏幕的缩小自动适应--缩小

img的宽度为100%,通过img标签来实现

w>=768px：

图片做为背景，当屏幕宽度变宽的时候，图片的高度不在改变

1.background-image添加图片

2.添加background-position:center center

3.background-size:cover

## 5.2 16-移动web-轮播图插件的使用说明

**1、data-target和href的两种控制**

data-target="#carousel-example-generic"：当前这个li元素控制的是id为carousel-example-generic这个容器 data-target直接找到目标id容器

href="#carousel-example-generic"：设置控制的是id为carousel-example-generic这个容器 href是锚链接到id上，

## 5.3 17-移动web-移动端的轮播图效果实现

1、移动端图片是img标签图片，需要设置宽度100%，图片可以点击需要包含在a标签中

## 5.4 18-移动web-非移动端的轮播图效果实现

1. 高度不在改变，图片设置成背景图片。
2. 图片也是可以点击跳转链接，需要包含在a标签中
3. 图片居中显示:background-position:center center
4. 图片内容全部显示 banckground-size:cover

## 5.5 19-移动web-使用响应式工具实现轮播图

1. **响应式工具如何使用？**
   1. 非移动端就显示非移动端图片即背景图片

移动端就显示移动端图片即img标签图片

* 1. 背景图在移动端隐藏 hidden-xs, img标签图片在非移动端隐藏 hidden-sm/hidden-md/hidden-lg

1. **利用响应式工具实现会存在哪些问题？**
   1. 资源浪费 (在移动端也会加载非移动端图片，在非移动端也会加载移动端图片)
   2. 加载缓慢 (每次都会把所有的图片都加载一次)，用户体验不好

## 5.6 20-移动web-使用js实现轮播图

1. **如何获取data属性 (jquery的新方法)**

data-small-image=”” => data(“smallImage”)

data-large-image=”” => data(“largeImage”)

1. **js如何根据当前屏幕大小来获取对应图片**
   1. 页面结构item写成自定义data属性，其值分别对应移动端和非移动端的图片路径
   2. resize来获取屏幕尺寸的实时切换
   3. 判断当前是移动端就显示移动端图片，非移动端就显示非移动端图片
   4. 遍历再获取item自定义的data属性 ，添加子元素中，同时把data属性传给背景图还是标签图

$(**function**(){  
**var** items=$(".carousel-inner .item");  
$(window).on("resize",**function**(){  
**var** width=$(window).width();  
**if**(width>=768){/\*说明非移动端\*/  
 /\*为每一个item添加子元素--遍历\*/  
$(items).each(**function**(index,value){  
**var** item=$(**this**);/\*当前自定义属性中 存储的图片路径\*/  
**var** imgSrc=item.data("largeImage");  
console.log(imgSrc);  
/\*添加非移动端的子元素\*/  
item.html($('<a href="javascript:;" class="pcImg"></a>').css("backgroundImage","url('"+imgSrc+"')"));  
});  
}  
**else**{  
$(items).each(**function**(index,value){  
**var** item=$(**this**);  
**var** imgSrc=item.data("smallImage");  
item.html('<a href="javascript:;" class="mobileImg"><img src="'+imgSrc+'" alt="..."></a>');  
});  
}  
 }).trigger("resize");

## 5.7 21-移动web-实现轮播图的滑动操作

**1、如何实现滑动操作？**

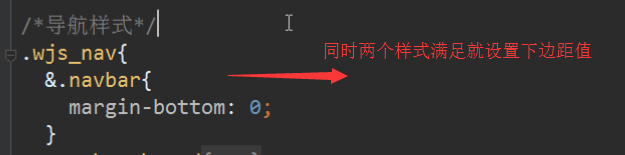
1. 移动端滑动事件：touch
2. 获取touchStart ：startX和touchEnd ：endX的坐标
3. 判断endX-startX的差值，

如果>0 表示向右滑，就上一张 carousel.carousel('prev');

如果<0,表示向左滑，就下一张 carousel.carousel('next');

## 5.8 22-移动web-微金所首页轮播图的完成

轮播图图片顶部距离的清除





昨日知识点回顾：

01、bootstarp布局容器+栅格系统的使用

02、less环境搭建和语法说明

03、wjs页面结构搭建+wjs页面头部的完成

04、bootstarp导航组件说明及wjs导航的完成

05、轮播图效果的实现

今日知识点：

01、微金所页面模块的完成

02、bootstarp定制的简单说明

03、rem的使用简单介绍+优惠活动案例

# 1 微金所页面模块的完成

## 1.1 01-移动web-微金所信息块制作

1. **信息块结构如何搭建？**
   1. >768不显示 hidden-xs
   2. 768-992 显示两列 栅格系统 col-sm-6
   3. >992 三列 栅格系统 col-md-4
   4. 栅格系统中内容跟 bootstrap中：组件>媒体对象 类似
   5. 栅格系统盒子是可以点击，都需包含在a标签中

## 1.2 02-移动web-微金所预约块制作

**1、预约块结构如何搭建？**

* 1. >768不显示 hideen-xs
  2. >768 二列 栅格系统 col-md-9 col-md-3
  3. 鼠标靠近颜色改变

## 1.3 03-移动web-标签页的使用

1. javascript插件>标签页

**href="#home"：锚链接，说明当前导航项链接到id号为home的面板上  
data-toggle="tab"：它会触发一个事件，这个事件就是切换到指定的面板**

## 1.4 04-移动web-标签页导航样式修改(上)



要更改框架样式还是复制出来，在通过绝对路径(加上当前块的类名)来修改 ，就不会改动框架的类名样式了

## 1.5 05-移动web-标签页导航样式修改(下)

鼠标靠近的一些默认样式的修改

## 1.6 06-移动web-详细产品块结构

**结构分析：**

1. **屏幕区分**

小屏显示两份 col-sm-6

中屏及以上三份 col-md-4

超小屏显示一份 col-xs-12

**2、左边内容设置：栅格系统 col-xs-6**

## 1.7 07-移动web-详细产品块样式(上)

1. **左边和右边布局**
   1. 右边position:absolute; width:100px
   2. 左边margin-right:100px
   3. 右边盒子通过::before,::after 设置半圆，内阴影用到box-shadow中的insert

## 1.8 08-移动web-详细产品块样式(中)

**对偶数设置：div:nth-of-type(even)**

## 1.9 09-移动web-详细产品块样式-宝北的添加

1. **Javascript > 工具提示**
2. **tooltip的一些属性**

type="button":说明当前工具提示的类型，类型默认是按钮，如果不需要， 可以修改为其它任意的元素类型

data-toggle="tooltip":说明当前插件/组件是一工具提示

data-placement="top"：提示出现的位置

title="提示文本"

**3、提示框需要初始化 引入js**

$(function(){

$('[data-toggle="tooltip"]').tooltip()

})

## 1.10 10-移动web-产品active块的制作

当前有active设置背景颜色为红色，字体颜色为白色

## 1.11 11-移动web-产品块标签页导航条的宽度计算

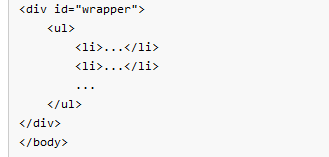
1. **宽度的几种计算方法**

**width():它只能得到当前元素的内容的宽度  
innerWidth():它能获取当前元素的内容的宽度+padding  
outerWidth():获取当前元素的内容的宽度+padding+border  
outerWidth(true):获取元素的内容的宽度+padding+border+margin**

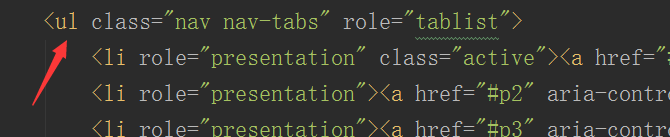
## 1.12 12-移动web-产品块标签页导航条的滑动

1. **如何使用iscroll.js让页面滑动**

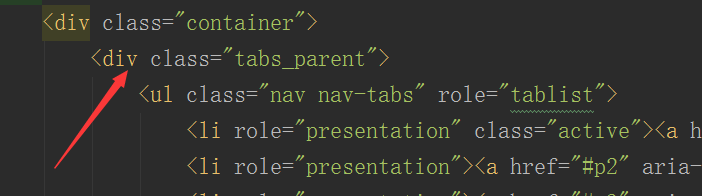
A) iscroll.js提供给我们的模板



B) 我们页面的模板



C) 我们只需要在他前面加一个div 包裹我们的ul就可



D) 引入js (是水平滑动所以需要开启水平滑动关闭竖直滑动)

传入的对象也是我们加的div中的类名

**var** myScroll = **new** IScroll('.tabs\_parent',{  
/\*设置水平滑动，不允许垂直滑动\*/  
scrollX: **true**, scrollY: **false**});

## 1.13 13-移动web-新闻块结构分析

**结构分析：**

1. >992
   1. 标题占一份 col-md-2 有位偏移col-md-offset-2
   2. 导航显示一列 col-md-1
   3. 导航栏对应的 占7分 col-md-7
2. <992 就显示一行 导航栏对应的也显示一行

## 1.14 14-移动web-新闻块结构样式(上)

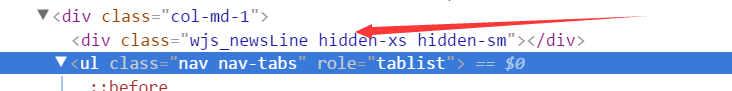
**步骤分析：**

1. **标题全部新闻地变宽设置**
   1. 右下角圆，通过伪元素定位，右位偏移圆的的宽度，往下位偏移圆的一半都是负值
2. **导航栏的设置**
   1. 给li设置margin-bottom
   2. a标签设置圆形，清除默认边框样式
   3. a：hover添加京东红背景色，去除边框
   4. Li标签有active的给a, a:hover, a:focus设置京东红背景色，去除边框

## 1.15 15-移动web-新闻块结构样式(中)

**步骤分析：**

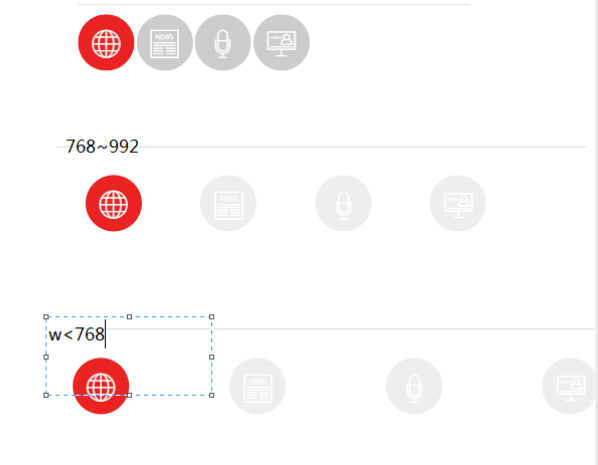
1. 导航栏最后一个删除底边距 li:last-of-type{margin-bottom:0}
2. 中间虚线的设置
   1. 重新设置一个跟ul同级的div 在设置定位，然后再通过位偏移盒子的内边距+宽度(45px)即可虚线居中



* 1. 虚线在md级以上显示，需设置hidden-xs,hidden-sm

## 1.16 16-移动web-新闻块结构样式(下)

1. **导航栏结构的分析**



1. **>992 就是显示一列 (也就是刚才所做的)**
2. **768-992 导航栏之间的边距是固定的 通过媒体查询**

@media screen and (min-width: 768px) and (max-width: 992px) {  
 .wjs\_news .nav-tabs >li {  
margin: 20px 30px;  
}  
}

1. **<768 导航栏之间是均匀分配的**

@media screen and (max-width: 768px) {  
 .wjs\_news .nav-tabs >li {  
margin: 20px 0px;  
width: 25%;  
}  
}

1. **导航栏对应的显示内容，需要设置行高和清除ul 的默认样式**

## 1.17 17-移动web-合作伙伴块制作

1. 步骤分析
   1. 文字和图标居中显示

footer标签类名设置text-align:center ul需设置display:inner-block

* 1. 每个li 设置浮动+左边距

# 2 bootstarp定制的简单说明

## 2.1 18-移动web-bootstrap定制的简单说明

**定制步骤：**

**1、官网:定制 >Grid system>@grid-columns**

* 1. **改变其其根据需求改成想要的**
  2. **点击最底下的编译并下载**

# 3 rem的使用简单介绍+优惠活动案例

## 3.1 19-移动web-em和rem的介绍

1. **浏览器默认字体如何修改**
   1. 设置 > 网络内容 > 设置成需要的字体大小
2. **设置font-size对ren和em的影响**
   1. em： 容器设置font-size对em有关，会做相应的倍数，

例：浏览器默认是16px, width:100em，这时容器宽度就是160px，

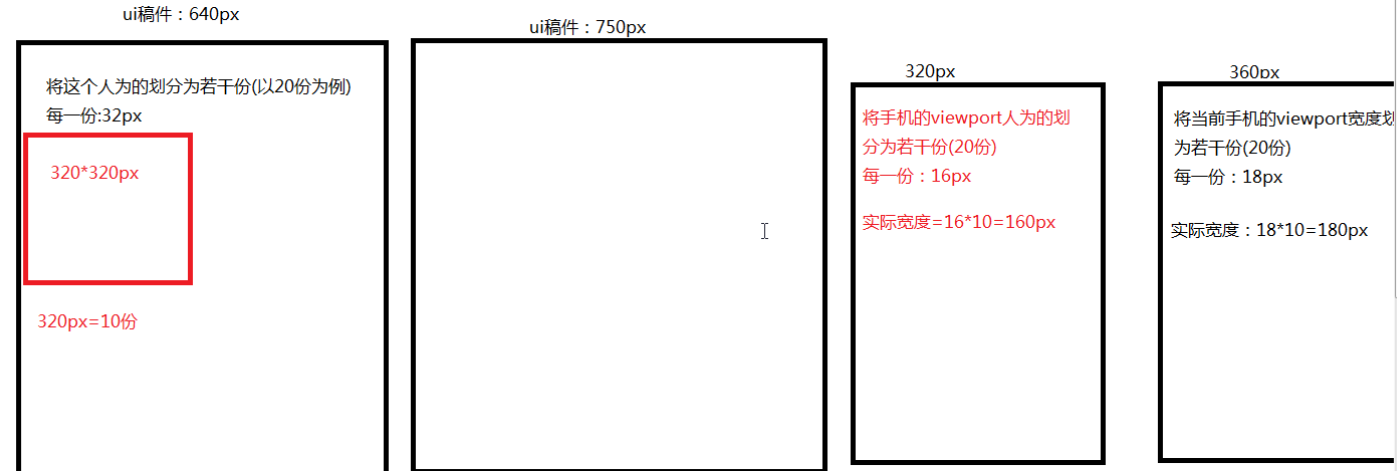
当设置font-size:12px, width:100em，这时容器宽度就是120px，

b) rem: 容器设置font-size对rem 没影响

例：浏览器默认是16px, width:100rem，这时容器宽度就是160px，

当设置font-size:12px, width:100rem，这时容器宽度就是160px，

## 3.2 20-移动web-rem实现适配的简单说明



@media screen and (device-width: 320px){  
html{font-size: 16px; }  
}  
@media screen and (device-width: 360px){  
html{font-size: 18px; }  
}  
@media screen and (device-width: 375px){  
html{font-size: 18.75px;}  
}  
@media screen and (device-width: 414px){  
html{font-size: 20.07px;}  
}  
div{  
width:10rem;  
height:10rem;  
background-color: red;  
}

说明：针对不同屏幕尺寸，设置不同的大小，都是分为20等分

## 3.3 21-移动web-rem案例-优惠活动(上)

1. **如何引入less文件**

<!--直接引入less文件-->  
<link rel="stylesheet/less"href="./css/index.less">  
<!--引入能够解析less文件的js插件-->  
<script src="./js/less.js"></script>

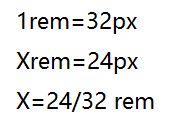
1. **less文件中如何导入其他的less文件 (也是存在路径的)**

**语法：@import ‘ 文件路径’**

/\*引入其它的less文件\*/  
@import 'base.less';  
/\*引入字体文件\*/  
@import 'fonts.less';  
/\*引入屏幕适配文件\*/  
@import 'adapter.less';

## 3.3 22-移动web-rem案例-优惠活动(中)

1. **Rem具体怎么计算？**
   1. **Ui设计为640px, 分成20份 每份就是640px/20=32px**
   2. **每份=32px,也就是1rem=32px**
   3. **24px=24/32 rem =0.75rem (表示占0.75份)**



在320px屏 每份=16px, 此时尺寸=16\*0.75rem=12px,

在360px屏 每份=18px, 此时尺寸=18\*0.75rem=13.5px,

## 3.3 23-移动web-rem案例-优惠活动(下)

总结：我们所有的单位都是以rem来计算的，而rem都是以开始的份数来计算，即我们开始20份=640px,每份=32px, 1rem=32px