# 触屏事件

细节:touch事件的触发,必须保证元素有具体的宽高值,如果宽或者高为0,则不会进行触发 touch仅移动端使用 pc端不可用

后面的zepto库对touch进行了封装



### 事件类型(前三个一定要掌握 中断事件仅了解)

touchstart: 手指触摸屏幕时触发

touchmove: 手指在屏幕上移动时触发

touchend: 手指离开屏幕时触发

扩展事件:touchcancel:触摸意外中断(例当打游戏时突然来了电话 就会显示电话接听界面 此时就是触摸意外中断,在手机中一些事件和通知会有较高优先级 如果遇到优先级高的就会触摸意外中断)

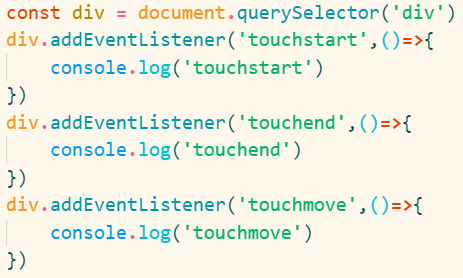
见示例代码7-1.html

我们所说的触屏事件都是对移动端而言 只有移动端才有 在pc端是没有的pc端是通过鼠标来实现 而手机端通过手指

**加事件的方式:**

在pc端通常用onclick onmouseover这种on方式添加事件,但是在移动端一般使用addEventListener添加

element. addEventListener(事件类型string,事件回调函数)



### TouchEvent对象

touches: 当前屏幕上的所有手指的列表

targetTouches: 位于该元素上的所有手指的列表

changedTouches：当前屏幕上变化的手指对象(从无到有,从有到无),touchstart时包含刚与触摸屏接触的触点，touchend时包含离开触摸屏的触点

总结:在使用时其实三个都差不多 但是我们取手指坐标值通常会取targetTouches的 因为它是针对元素的 targetTouches是伪数组可以[0]取

注意: 如果touchend事件时使用targetTouches是获取不到的 因为前面是手指离开时触发 此时手指已经离开 后者是获取位于该元素的手指列表 在touchend事件如果想获取手指就用changedTouches 其他时候就targetTouches

例

div**.**addEventListener('touchstart',(**e**)**=>**{

   console**.**log(e**.***targetTouches*[0]**.***clientX***+**'----------'**+**e**.***targetTouches*[0]**.***clientY*)

})

见代码示例7-2.html

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

(官方对上面三个是这么说明的 但是实际上touches和targetTouches没有区别 都只能在元素上触发 相当于targetTouches 如果在空白屏幕触摸是无效的不会触发事件 如果在元素上触摸相当于又触摸元素又触摸屏幕)

**三者区别:**

touches和targetTouches没有区别就是一样的,是记录所有手指的列表,如果有三个手指触摸就会记录三个手指对象

changedTouches:只会记录一个手指对象,touchstart时是从无到有包含刚与触摸屏接触的触点，touchend时从有到无包含离开触摸屏的触点.两个事件打印出的targetTouches分别是触摸或离开的触点 一个事件一个触点

touches 和changedTouches区别是前者记录所有手指列表 后者只记录一个手指(变化的那个)

### Touch对象

clientX/Y 相对于浏览器视口左上角的距离(首选)

pageX/Y 相对于文档左上角的距离 如果内容超出有滚动条则包含滚动距离

screenX/Y 是手指触摸点相对于屏幕左上角的距离 不是浏览器左上角而是屏幕设备左上角

target 手指最初与屏幕接触时的元素 和h5中的事件源对象没区别

移动开发通常会设置<meta name="viewport" content="width=device-width, initial-scale=1">，这时这三对坐标值是完全一样的。

见代码示例7-3.html

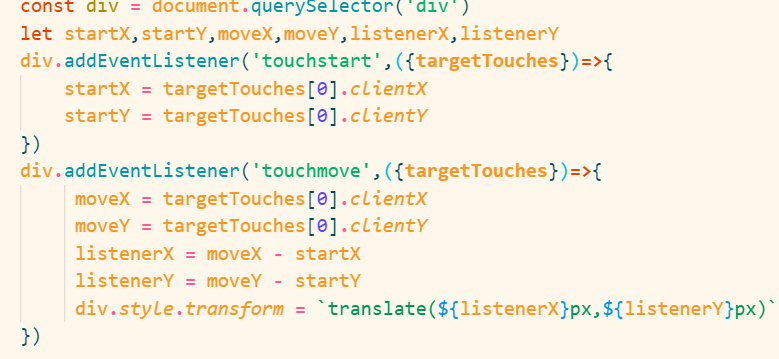
注意:在移动端我们会避免出现横向滚动条情况 最宽也只有设备屏幕宽 不会有内容超出的情况 所以不用过多关注clientX/Y和pageX/Y区别 相当于它们都一样 而在取值时我们一般取**clientX/Y** 这和pc端不同

取触摸点坐标 例:

div**.**addEventListener('touchstart',(**e**)**=>**{

   console**.**log(e**.***targetTouches*[0]**.***clientX***+**'----------'**+**e**.***targetTouches*[0]**.***clientY*)

})

案例:简单拖拽功能

以上案例只是对特定某个元素进行拖拽,如果想对屏幕上任意元素进行拖拽

通常我们会给documen加监听事件, e.target可以获取到触发的事件源对象, document的下面任意元素都可以获取到,而坐标都是针对事件源对象的

所以通常我们会为document或者body监听事件 获取的坐标就是事件源坐标不会变 而事件对象就通过e.target获取

例 只把这document和e.target写上 其他不变

document**.**addEventListener('touchmove',({**targetTouches**})**=>**{

     e**.***target***.***style***.***transform* **=** `translate(${listenerX}px,${listenerY}px)` })

**webkitTransitionEnd过渡效果完成后触发事件**

transition执行完毕后触发该事件,注意transition动画执行时不影响后续代码 后续代码也会同步继续执行

document.getElementById("myDIV").addEventListener("webkitTransitionEnd", myFunction);

注意:此事件仅在过渡效果完毕后才触发 如果在过渡时间还没结束又触发下一个过渡 则不会触发此事件 例如轮播图特别快速滑动的时候(设定0.5过渡 但是0.5时间还没到又快速滑动到下一张 这就会导致过渡还没完成不会触发事件)

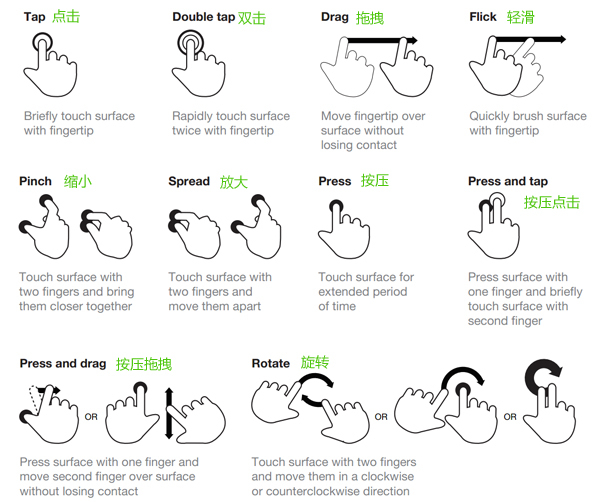
### click延时

早期移动设备浏览器网页时内容非常小，为了增强用户体验，苹果公司专门为移动设备设计了双击放大的功能，确保用户可以非常方便的放大网页内容，但是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。用触屏事件可以解决这个问题，见代码示例7-4.html

在移动端也可用click 但是会有300ms左右延迟 所以为了效率就使用touch事件

### 手势封装 zepto里面封装的收视函数

tap事件就是点击事件, 使用Zepto事件封装的tap来进行模拟，但是这会带来一个副作用，这个副作用就是“点透”(类似冒泡 实际又不是父子关系才有的冒泡),所谓的点透是哪怕没有父子或子孙关系 只要视觉上重合在一起 点击面上的 底下的也会触发click类似点击事件 后面的fastclick.js解决了这个问题



利用触屏事件简易封装手势，主要用途是熟悉触屏事件的使用

1、tap检测接触和离开屏幕的距离来实现，见代码示例7-5.html

2、drag跟踪手指移动位置，进而设置元素定位坐标，见代码示例7-6.html

3、swipe 判断手指滑动的方向，见代码示例7-7.html

### zepto.js

在移动端很少使用jquery, jquery主要解决浏览器兼容性问题,然而移动端浏览器都是比较先进的浏览器意味着它的兼容性问题非常容易解决 不需要我们的jquery大量的篇幅来解决 第二在jquery中很多函数都是实现动画的 但是现在在移动端浏览器里面它是完全支持CSS3的 而css3里面的动画和过渡的添加可以大大减少jquery函数的依赖 所以基于这两个原因我们很少在移动端使用jquery 因为在移动端jquery的体积过于庞大 效率也不是非常高 因为它做的很多事情在移动端根本就没有意义 在移动端类似jquery的是zeptojs

jquery是针对pc端的 zepto是针对移动端的

zepto的api和jquery类似 接近于翻版 所以会使用jquery就会使用zepto 顶多只有个别不同

在jquery中我们只需要引入一次js文件 但是zepto中它是分模块化的 需要哪个功能就引入对应模块 所以我们使用时可能会引入多次(如果不想多次引入 还可进行定制 打包为一个js 一次性引入 见后续操作--定制zeptojs)

zepto对移动端的touch事件也进行了封装 还封装了 左滑动 右滑动(左右滑动在谷歌模拟机会失效 但在真机有效 所以不用过多纠结)

定制zeptojs

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

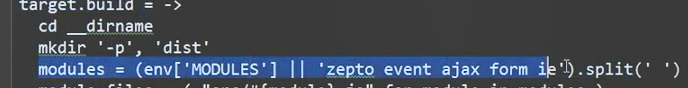
2、github下载[zepto.js](https://github.com/madrobby/zepto)

3、解压缩

4、cmd命令行进入解压缩后的目录

5、执行npm install 命令

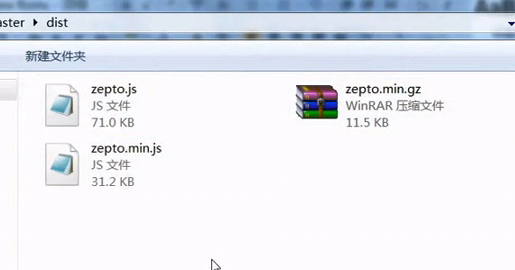
6、编辑make文件，添加自定义模块并保存，如下图



要加哪个模块就写模块名注意最后不能写空格末尾写空格会执行错误

7、保存后 然后执行命令 npm run-script dist 会执行脚本生成dist目录

8、查看目录dist即构建好的zepto.js 这个文件就包含新添加的模块 实现了定制js



见代码示例7-8.html

# 移动端类库

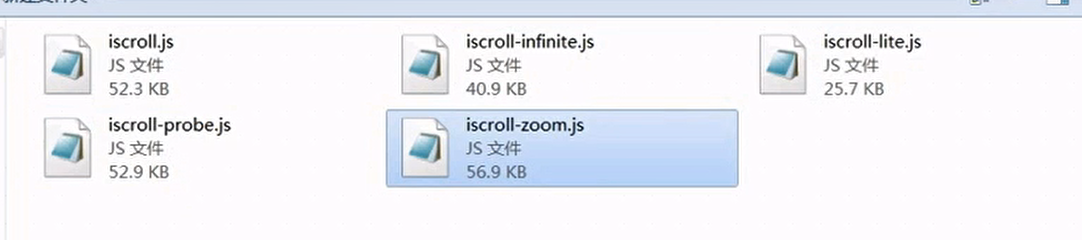


### iScroll.js滚动滑动插件

一个可以实现客户端原生滚动效果的类库。(这里只会讲解基本的滚动效果 如果要其他效果自行了解 下载的文件demo里面还有雷类似淘宝的放大镜案例)

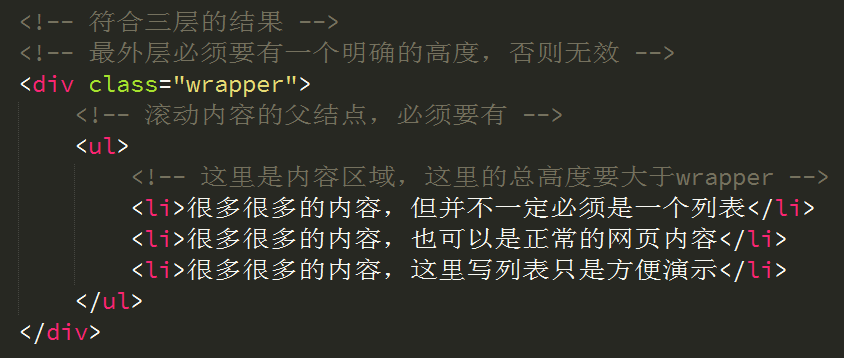
1、[下载iScroll](https://github.com/cubiq/iscroll)

2、build目录下提供了不同版本的iScroll(不同版本包含的功能不同)，可根据情况选择使用 (iScroll也是分模块的 不同模块在src文件下可找到需要哪个引哪个 而build目录类似dist 里面提供了几个定制好的文件 zoom是最全的 完整版 包含所有模块 但是使用时一般使用iScroll文件 因为里面包含了最常用的模块 使用也是使用这个)



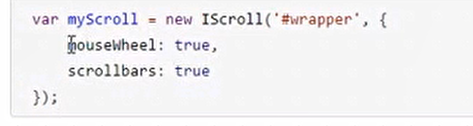
3、html要求有3层结构如下图

注意(这里并不是要这个标签 而是必须要这个结构 例如将ul换成div li换成p标签都可以)



4、获取wrapper这个最外层结点，然后实例化即可滚动生效， new Iscroll(一参,{})第一个参数可以传元素id例’#wrapper’或者dom元素 注意id命名随意 第二个参数(可选)是对象里面进行一些配置 这里mouseWheel表示用鼠标中键滚动 第二个参数是滚动条(注意滚动条是absolute定位的 如果当前元素没定位就会以浏览器参照 所以此时需对当前元素定位 让滚动条在合适位置)

步骤1.引入js 2.写正确结构 3.实例化





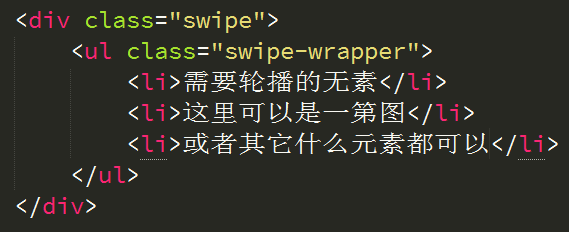
见示例代码iscroll.html

### swipe.js轮播图插件 推荐另一个swiper插件

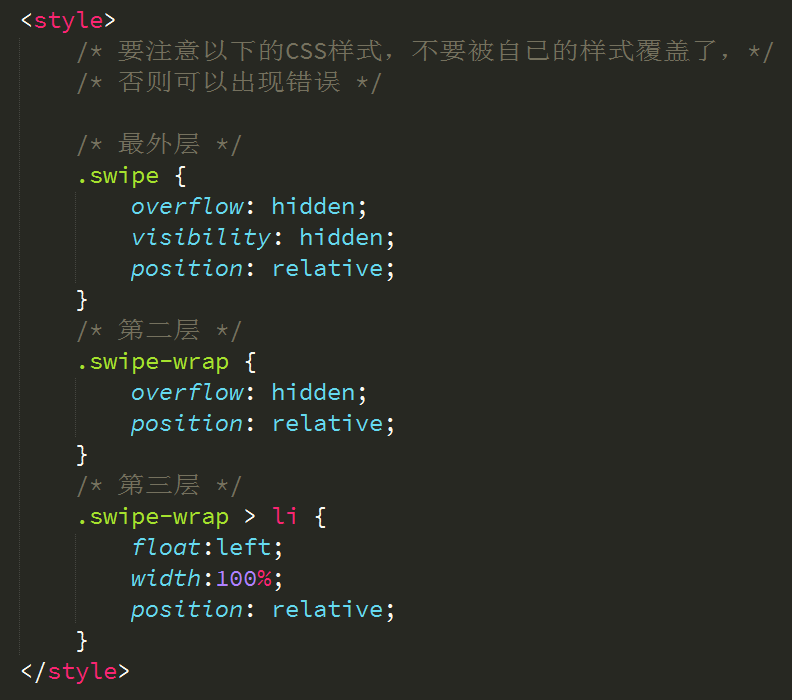
作用主要做轮播图滑动

1、[下载swipe.js](https://github.com/thebird/Swipe)

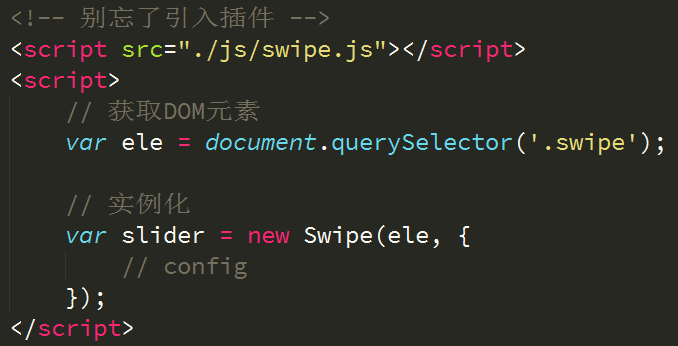
2、html结构要求有三层结构(对结构有要求 标签不要求 标签内的选择器命名例类名要有名字 名字任意不要求)，如下图

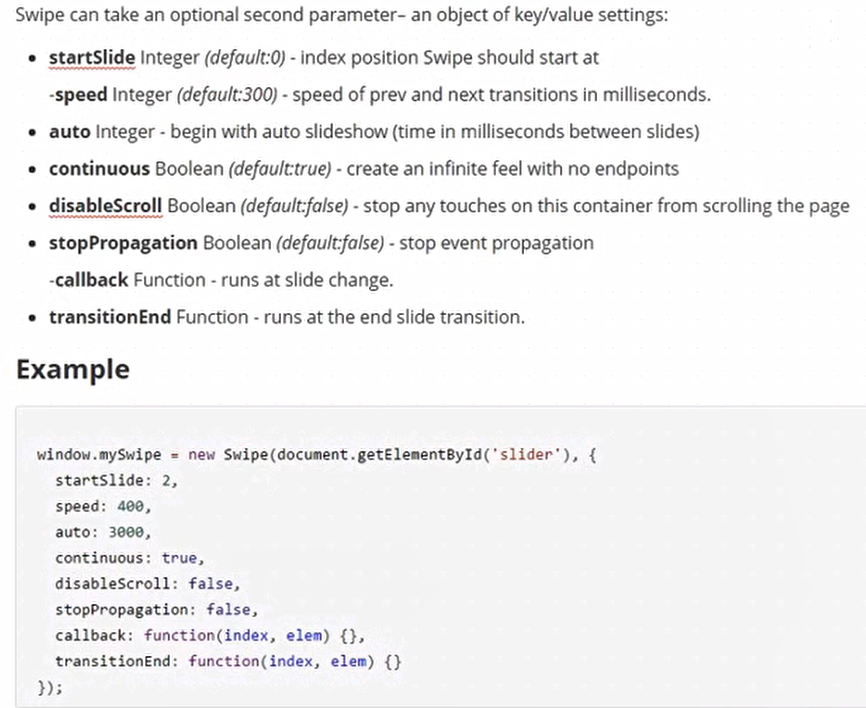


4、需要一些基础CSS样式，这些样式要对应到html结构上，如下图



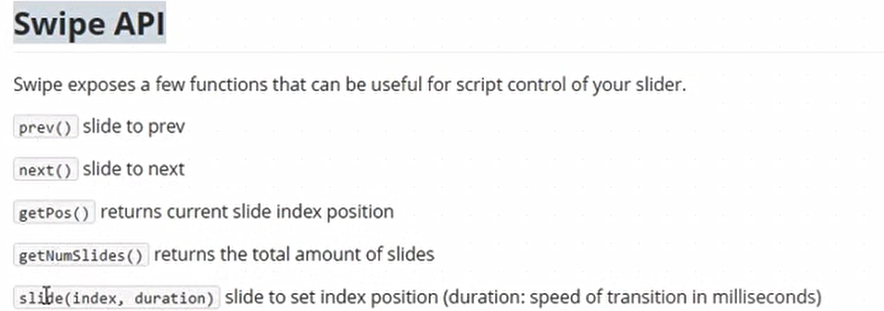
5、获取最外层的swipe元素，然后实例化，如下图 如果需要进行配置就传第二个参数 参数见图二 配置后可以实现拖拽滑动 自动轮播等 注意pc端不能拖拽滑动





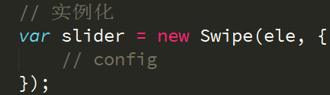
见示例代码swipe.html

**还有相关API(上一张 下一张 当前图片索引位置 图片总张数 可滑动到设置索引位置)**



例如移动端拖拽滑动 但是pc端不可以 pc只有点击滑动 这里就只有用api来操作

步骤1.自己设置点击按钮 实现上一张下一张 执行的函数里面用api 就写实例化后的元素点方法



前面实例化了 然后slider.prev() 即可实现上一张 到此结束

### swiper.js 轮播图插件(推荐) 触摸滑动插件 比较强大

1、[下载swiper.js](https://github.com/nolimits4web/Swiper/)，

2、其[中文网站](http://www.swiper.com.cn/)非常详细介绍了其使用方法

### fastclick.js点击插件 解决touch点透 实现click无延时

在移动设备上为了提升click的响应速度，我们选择了使用Zepto事件封装的tap来进行模拟，但是这会带来一个副作用，这个副作用就是“点透”，我们通过一个例子来解释“点透”，见示例代码fastclick.html（自行查阅点透现象发生的原因）

### fastclick.js作用:

1. pc端 移动端都可用
2. 解决点透问题
3. 解决click延迟问题

click本身就可以解决1.2问题 所以fastclick主要是解决延迟问题 这样一来 三者满足就可以成功解决了



如图实际应用的场景，当点击半闭按钮时，如果下面有click事件或链接则会被触发。

1、[下载fastclick](https://github.com/ftlabs/fastclick)

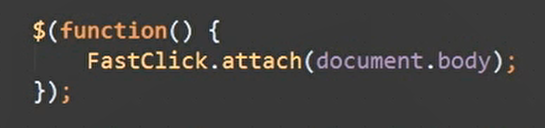
2、引入lib目录下的fastclick.js

2、调用方法即可，如下图

attach就是fastclick效果 可以写任意dom元素 但是推荐在整个body节点上调用 这样表示body下的所有元素都绑定了fastclick 写了以上操作后 就可以正常使用click且不会有延迟

只要使用了attach后 再使用click就不是原生的click了 而是做了一个简单的封装

方法一:jquery的attach



方法二:js的attach

