1. 什么是iscroll，产生

iScroll是一个高性能，资源占用少，无依赖，多平台的javascript滚动插件。它可以在桌面，移动设备和智能电视平台上工作。它一直在大力优化性能和文件大小以便在新旧设备上提供最顺畅的体验。

iScroll不仅仅是滚动。它可以处理任何需要与用户进行移动交互的元素。在你的项目中包含仅仅4kb大小的iScroll，你的项目便拥有了滚动，缩放，平移，无限滚动，视差滚动，旋转功能。给它一个扫帚它甚至能帮你打扫办公室。

即使平台本身提供的滚动已经很不错，iScroll可以在此基础上提供更多不可思议的功能。具体来说：



细粒度控制滚动位置，甚至在滚动过程中。你总是可以获取和设置滚动器的x，y坐标。



动画可以使用用户自定义的擦出功能（反弹'bounce'，弹性'elastic'，回退'back'，...）。



你可以很容易的挂靠大量的自定义事件（onBeforeScrollStart,\*开箱即用的多平台支持。

从很老的安卓设备到最新的iPhone，从Chrome浏览器到IE浏览器。

产生：iScroll的产生完全是因为移动版webkit浏览器，例如在iPhone，Android 的移动设备上。

iScroll的原理

是外层有一个溢出隐藏（overflow:hidden;）的DOM，然后这个区域内的第一个DOM结构会被实例化，其包裹的内容可 以纵向或者横向的滚动，所以在使用iScroll的时候，滚动元素要尽量的简单，减少DOM个数，减少嵌套，因为DOM结构越是复杂iScroll运行起 来就越是吃力，有可能会造成某些节点显示不正常的情况。

<div id="wrapper">//overflow:hidden;

    <ul>

    //只有第一个DOM结构（ul）被实例化，这个DOM可以纵向或者横向的滚动，

    //多出的内容会被wrapper的样式hidden。

        <li>1</li>

        <li>2</li>

        <li>3</li>

    </ul>

</div>

**注意：再次重申，只有wrapper里的第一个子元素（ul）才可以被实例化滚动，并且要结合外层的DOM（wrapper）才能实现滚动。**

如果 wrapper中有多个ul怎么办？很简单，记住那句话，只有wrapper里的第一个子元素（ul）才可以被实例化滚动：

<div id="wrapper">//overflow:hidden;

    <div id="first">

        //只有第一个DOM结构(ul)被实例化，这个DOM可以纵向或者横向的滚动，

        //多出的内容会被wrapper的样式hidden

        <ul>

            <li>1</li>

            <li>2</li>

            <li>3</li>

        </ul>

        <ul>

            <li>4</li>

            <li>5</li>

            <li>6</li>

        </ul>

    </div>

</div>

这里第一个DOM结构的ID（first）可以不写，我只是为了方便大家识别才写了个ID，但是最外层的ID（wrapper）一定要写，因为在JS实例化的时候需要填写这个ID：

|  |  |
| --- | --- |
| 1 | var myScroll = new iScroll("wrapper"); |

**iScroll中的参数：**

在实例化iScroll的时候，可以传入两个参数，第一个参数是实例化的外层的DOM的ID，第二个参数是iScroll执行方法的对象：

|  |  |
| --- | --- |
| 1 | var myscroll=new iScroll("wrapper",{hScrollbar:false}); |
|  |  |
|  |  |

hScroll        false 禁止横向滚动 true横向滚动 默认为true

vScroll        false 禁止垂直滚动 true垂直滚动 默认为true

hScrollbar     false隐藏水平方向上的滚动条

vScrollbar     false 隐藏垂直方向上的滚动条s’s

fadeScrollbar  false 指定在无渐隐效果时隐藏滚动条

hideScrollbar  在没有用户交互时隐藏滚动条 默认为true

bounce         启用或禁用边界的反弹，默认为true

momentum       启用或禁用惯性，默认为true，

               此参数在你想要保存资源的时候非常有用

lockDirection  false取消拖动方向的锁定，true拖动只能在一个方向上（up/down 或者left/right）

iscroll的方法  
当然在第二个参数中，也有一些方法可以执行：

(1)**scrollTo(x, y, time, relative)**方法：传入4个参数：X轴滚动距离，Y轴滚动距离，效果时间，是否相对当前位置。所以例如：

|  |  |
| --- | --- |
| 1  2  3  4 | //在200毫秒的时间内，Y轴向上滚动100像素；  uw3c.scrollTo(0, -100, 200)  //在200毫秒的时间内，相对于当前位置，X轴向左滚动100像素；  uw3c.scrollTo(-100, 0, 200, true) |

(2)**refresh()**方法：在DOM结构发生改变之后，需要刷新iScroll，否则滚动插件会实例化的不准确：

|  |  |
| --- | --- |
| 1 | uw3c.refresh();//刷新iScroll |

(3)**onPosChange**,有没有一个方法能返回位置的变化？你可以查询一下自己所用的iScroll中有没有onPosChange方法：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | onPosChange:function(x,y){      if(y < -200){       //如果Y周向上滚动200像素，$("#uw3c")就显示，否则就隐藏。          $("#uw3c").show();      }else{          $("#uw3c").hide();      }  } |

（4）**onScrollEnd**：滚动结束时执行的事件，如果想在滚动结束时出发摸个事件，这个方法就拍上用处了：

|  |  |
| --- | --- |
| 1  2  3  4 | //滚动结束后，执行的方法，滚动后会出现提示框alert("uw3c.com")  onScrollEnd:function(){      alert("uw3c.com");  } |

（5）**onRefresh**：在DOM结构发生改变之后，需要刷新iScroll，否则滚动插件会实例化的不准确，onRefresh是刷新完iScroll会执行的方法。

（6）**onBeforeScrollStart**：开始滚动前的时间回调，默认是阻止浏览器默认行为 。

（7）**onScrollStart**：开始滚动的回调。

（8）**onBeforeScrollMove**：在内容移动前的回调。

（9）**onScrollMove**：内容移动的回调。

（10）**onBeforeScrollEnd**：在滚动结束前的回调。

（11）**onTouchEnd**：手离开屏幕后的回调。

（12）**onDestroy**：销毁实例的回调。

Iscroll4在使用时常见的问题

1.在iscroll4的滚动容器范围内，点击input框、select等表单元素时没有响应  
这个问题原因在于iscroll需要一直监听用户的touch操作，以便灵敏的做出对应效果，所以它把其余的默认事件屏蔽了，解决的方法是，在iscroll4源码里面找到这一行，

     onBeforeScrollStart: function (e) { e.preventDefault(); }

然后把它改成：  
    onBeforeScrollStart: function (e) { var nodeType = e.explicitOriginalTarget？ e.explicitOriginalTarget.nodeName.toLowerCase():(e.target？ e.target.nodeName.toLowerCase():'');if(nodeType !='select'&& nodeType !='option'&& nodeType !='input'&& nodeType!='textarea') e.preventDefault(); }

    这样只要你touch的元素是 select || option || input || textarea时，它就不会执行e.preventDefault()，默认的事件就不会被屏蔽了。

    如果你有其他不想被屏蔽的元素，可以自己修改，不过需要注意onBeforeScrollStart里的屏蔽默认事件很重要，它是iscroll进行流畅滚动的基础，不要随便的把它去掉，否则你会发现滚动起来很卡顿。

2.往iscroll容器内添加内容时，容器闪动的bug

    我在做上拉加载更多内容的时候，肯定需要把新的内容插入到容器内，这时发现有时容器会出现闪动，一开始认为是insert进去的内容太多，后来又觉得是不是因为里面布局用了float的原因导致重新渲染，最后通通排除。

    其实病灶在于iscroll使用了太为先进的CSS3属性，可能web webkit对这些属性的支持力度还是不够好。

    涉及的两个属性是  translate3d 和 TransitionTimingFunction，或许是这两个属性在列表长度改变时会影响到渲染，所以导致页面闪动，解决办法就是找到源代码的，

     has3d = 'WebKitCSSMatrix' in window && 'm11' in new WebKitCSSMatrix()

改成：  
     has3d = false

和在配置iscroll时，useTransition设置成false就可以了（useTransition默认是false的）。

    这样做有一点瑕疵就是滚动起来和原来比没那么流畅了（原来的效果真的是可以媲美原生app的），但是假如你不对比的话，是看不出来了。

    在效果和体验上面选择，我更看重体验。

    不过如果你符合下面的条件，我还是不建议你修改成我这样：

        1)即使你不修改，无论你怎么往iscroll容器里面插内容，它都不会闪动，这种情况大多出现在纯文字的列表。假如列表涉及复杂的布局和图片，很多时候会出现闪动bug

        2)如果你的web app只是单纯在手机浏览器浏览。translate3d 和 TransitionTimingFunction只是在IOS里的uiwebview支持不成熟，但是在手机上的safari完全没有问题，所以如果你 不是用phonegap之类的框架开发混合app，你不需要担心这个问题。

        3)只针对android，因为android的webkit暂时还不支持translate3d，iscroll会自动选择不用。

3.过长的滚动内容，导致卡顿和app直接闪退

    说白了iscroll都是用js+css3实现的，对浏览器的消耗肯定是可观的，避免无限制的内容加载本身就是web产品应该避免的。

    假如无可避免，我们可以尽量减低iscroll对浏览器内存的消耗

        1)不要使用checkDOMChanges。虽然checkDOMChanges很方便，定时检测容器长度是否变化来refresh，但这也意味着你要消耗一个Interval的内存空间

        2)隐藏iscroll滚动条，配置时设置hScrollbar和vScrollbar为false。

        3)不得已的情况下，去掉各种效果，momentum、useTransform、useTransition都设置为false

4.左右滚动时，不能正确响应正文上下拉动

    在做这种效果时 ，假如这个幻灯片模块只是你页面的一部分，你还需要上下拉动页面去浏览其它内容时，你的手指在这个模块上做上下拨动时，恐怕会没有反应。原因还是和问题1一样的，因为屏蔽了默认事件。

完美的解决方法是没有的，如果把 e.preventDefault() 去掉，幻灯片的滚动效果就会大打折扣，而且有时用户上下拨动的操作会被误操作成幻灯片的滚动。所以在效果还是体验上，大家还是自己选择吧。在技术上解决不 了的问题，我认为还是多和产品和UI沟通比较好，共同协商一个良好的方案。（转）

**Iscroll-4和iscroll-5的区别？**

最近做滚动效果，用到IScroll，当然就直接使用IScroll5了，在移动端用的iscroll-lite，轻而小但是 没翻页无限加载的功能

主要是 用到了在IScroll 里面在嵌入 一个IScroll  ，在IScroll4 里还是有决绝的办法，IScroll5就无能为力了(ps:应该有办法，只是本人没找到)，最后只把两个IScroll 做为同一级的元素，实现的。  
iScroll5 事件使用  
var myScroll = new IScroll(‘#wrapper‘);  
myScroll.on('api事件'，function(){});  
  
  
iScroll4 事件使用  
var myScroll =new iScroll("wrapper",{  
onRefresh: function(e){...},  
onScrollMove: function(e){...},  
onScrollEnd: function(e)...{}  
}  
  
  
所以 iscroll4  里 子类 iScroll 可以这样阻止 事件冒泡  
onBeforeScrollStart :function(e){  
    e.stopPropagation();  
在 iScroll5里  
myScroll.on(‘beforeScrollStart'，function(e){e.stopPropagation()});  
却是不能使用的  
  
使用 iscroll 时 需要用到 click 事件的  click:true;  
就ok了  
当然 对于移动端 要是讲究都不会用默认的click 事件可以自定义tap 事件