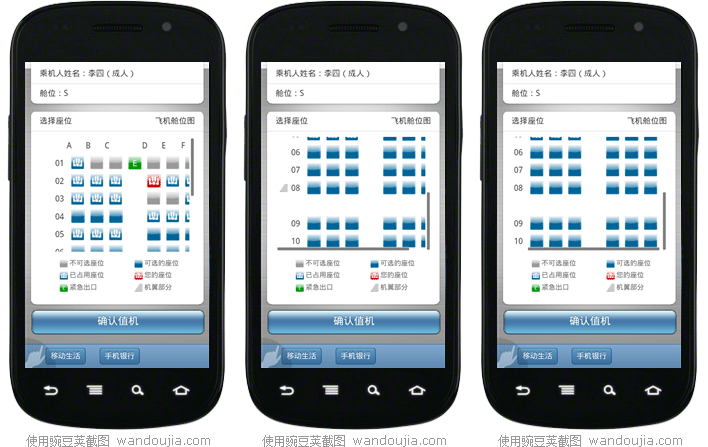
Iscroll材料收集

web移动平台前端UI开发工作，兼容问题超多，产品方要求在某固定尺寸容器内显示内容，但内容条数未知；如果条数过多，容器显示滚动条。这鸟需求按说是So easy，容器设死宽、高，CSS加属性overflow:scroll;完事！可拿设备去测的时候有趣了，Pc all browser正常，ios正常 ，android失效。泥马在android下效果与 overflow:hidden;一样，溢出部分隐藏了！  
水平有限，用尽吃奶力各种解决未果，Google搜索有人说是android系统BUG，最后查到比较靠谱的解决是国外某高手开发了个JS库，名曰iScroll 4 。 亲测，好用，唯独效率不敢恭维（拿别人的东西用，少TM废话）。  
下载地址：http://cubiq.org/iscroll-4  
使用方法：  
1. 引入 iscroll.js   
<script type="application/javascript" src="http://istyles.blog.163.com/blog/js/iscroll.js"></script>  
<script type="text/javascript">  
var myScroll;  
function loaded() {  
    myScroll = new iScroll('wrapper');  
}  
document.addEventListener('DOMContentLoaded', loaded, false);  
window.onload = function() {   
    setTimeout(function(){ new iScroll(document.getElementById('scroller')) }, 100)   
};  
</script>  
2. 添加必要的样式  
<style type="text/css">  
#wrapper {  
position:relative;  
z-index:1;  
height:200px;        /\* Desired element height \*/  
}  
</style>  
  
3. 容器加 id="wrapper" ， 内容加 id="scroller"   
<div id="wrapper">  
               <ul id="scroller">  
                    <li></li>  
                     ...  
                    <li></li>  
                </ul>  
</div>  
  
4. 有图有真相



iScroll框架解析（翻译）

**概要**

iScroll 4 这个版本完全重写了iScroll这个框架的原始代码。这个项目的产生完全是因为移动版webkit浏览器（诸如iPhone，iPad，Android这些系统上广泛使用）提供了一种本地化的方式来对一个限定了高度和宽度的元素的内容进行滑动。很不幸的是，这种情况下所有的web应用的页面就不能够包含具有position:absolute的头、页尾或者是一个内容可滚动的中间区域。

然而，Android系统最新修订版已经可以支持这种功能了（尽管支持的力度还不是特别好），Apple公司似乎不太情愿将one-finger滑动事件运用到div元素上。

除了以前版本的iScroll的特性以外，iScroll 4还包括如下的特性：

     （1）缩放（Pinch/Zoom）

    （2）拉动刷新（Pull up/down to refresh）

    （3）速度和性能提升

    （4）精确捕捉元素

    （5）自定义滚动条

      友情提示：iScroll 4并不是iScroll 3的简易替代版本，API文档已经不一样了。同时考虑到此版本正处于测试期，一些API 可能会有细微的变化。

**使用指南**

在此文档中你会发现很多例子来教会你如何快速上手iScroll脚本库。参看文中的demo小例子并仔细阅读此文档，可能有点小无聊，但是这篇文章中却是iScroll这个脚本库的精髓之所在哦。

iScroll需要对所要进行滚动的元素进行初始化，并且不限定一个页面中使用iScroll的元素的个数（这里不考虑您的硬件配置）。滚动元素中内容的类型和长度在一定程度上将会影响iScroll脚本库里可以同时使用的元素的个数。使用iScroll这个脚本库时，DOM树的结构要足够简单，移除不必要的标签，尽量避免过多的标签嵌套使用。最优的使用iScroll的结构如下所示：

<div id="wrapper">

        <ul>

               <li></li>

               .....

        </ul>

</div>

在这个小例子中，ul标签将会被滚动。iScroll一定要与滚动内容外面的wrapper进行联系才会产生效果。

【注意事项】：

只有wrapper里的第一个子元素才可以滚动，如果你想要更多的元素可以滚动，那么你可以试试下面的这种写法：

<div id="wrapper">

        <div id="scroller">

               <ul>

                    <li></li>

                     ...

                </ul>

                <ul>

                         <li></li>

                          ...

                </ul>

       </div>

</div>

在这个例子中，scroller这个元素可以滚动，即便它包含两个ul元素

iScroll必须在调用之前实例化，你可以在下面几种情况下对iScroll进行实例化：

    （1）onDOMContentLoaded

    （2）onLoad

    （3）以内联inline方式

下面我们逐个来讲讲这三种用法的优缺点

**ONDOMContentLoaded**

        适用于滚动内容只包含文字、图片，并且所有的图片都有固定的尺寸

        使用方法：（在head标签中添加如下代码）

        <script src="iscroll.js"></script>

        <script>

                 var myscroll;

                function loaded(){

                           myscroll=new iScroll("wrapper");

                 }

               window.addEventListener("DOMContentLoaded",loaded,false);

         </script>

         注意：myscroll这个变量是全局的，因此你可以在任何地方调用它的函数

**onLoad**

        有些时候在DOMContentLoaded的状态下就初始化iScroll其实是有点草率的，因此此时页面的资源可能还没有全部加载完毕。如果你遇到了一些很怪异的行为可以试试下面的写法：

        <script src="iscroll.js"><script>

        <script>

                    var myscroll;

                    function loaded(){

                   setTimeout(function(){

                            myscroll=new iScroll("wrapper");

                     }，100 );

                }

                window.addEventListener("load",loaded,false);

         </script>

         这种情况下iScroll会在页面资源（包括图片）加载完毕100ms之后得到初始化，这应该是一种比较安全的调用iScroll的方式。

**inline初始化**  
         这种情况会在页面加载到js的时候就进行调用，此方法不推荐使用，但是很多javascript的大牛都在用这种方式，我又有什么理由不赞成呢？

         <script src="iscroll.js"></script>

                    <div id="wrapper">

                            <ul>

                                <li></li>

                                 ...

                          </ul>

                  </div>

        <script>

                   var myscroll=new iScroll("wrapper");

        </script>

        不过建议你使用一些框架的ready方法来安全调用iScroll（比如jquery里的ready()）。

**iScroll里传递的参数**

        iScroll里的第二个参数允许你自定义一些内容，比如下面的这段代码：

       <script>

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

       </script>

       第二个参数通常都是一个对象，像上面的这个例子里就设定了不显示滚动条。常用的参数如下：

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

               vScroll               false 精致垂直滚动 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

               fixedScrollbar  在iOS系统上，当元素拖动超出了scroller的边界时，滚动条会收缩，设置为

                                        true可以禁止滚动条超出scroller的可见区域。默认在Android上为true，

                                        iOS上为false

               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）为了保持资源的完整性，建议去除滚动条

**拉动刷新（pull to refresh）**

        自从Twitter和一些Apple的本地化应用出现了这个效果之后，这个效果就变得非常流行。你可以看看[这儿](http://vimeo.com/20475828)先一睹为快。我最近把"pull to refresh"这个部分单分出来作为iScroll的一个额外插件。你可以点击[这儿](http://cubiq.org/dropbox/iscroll4/examples/pull-to-refresh/)看看pull to refresh是如何工作  滴。你只需要做的就是自定义pullDownAction()这个方法。你可能需要一个ajax来加载新的内容，不过一旦DOM树发生了变化要记得调用refresh这个方法来。需要记住的是在例子中我们加了1秒的延迟来模拟网络的延迟效果。当然，如果你不想使用这个延迟，那就把setTimeout方法去掉就行了。

**缩放(pinch / zoom)**

        我们不得不面对一个事实，那就是光有滚动其实没什么新意的。这就是为什么在iScroll 4这个版本里我们允许你可以放大和缩小。想要这个功能，只需要设置放大的参数zoom 为true即可实现利用手势来放大和缩小。你可以看看[这儿](http://vimeo.com/20475917)。双击放大和缩小的功能在iScroll 4里也是得到支持的。要使用缩放功能，你至少需要如下配置：

                var myScroll =new iScroll("wrapper",{zoom:true});

       如果你想对缩放功能进行深度的自定义的话可以使用下面的一些选项：

               zoomMax   指定允许放大的最大倍数，默认为4

      【注意事项】：如果想要图片缩放的效果很好的话要把他们放到硬件的合成层中。通俗点说，就是在所有需要缩放的img元素上使用-webkit-transform:translate3d(0,0,0)来实现，而且尤为重要的是，硬件的加速会占用大量资源，要谨慎使用，否则你的应用可能就此崩溃。

**捕捉元素（snap and snap to element）**

        捕捉的功能会促使scroller去重新定义位置，这样可以制作更加花哨的传送带效果。[点击这里](http://cubiq.org/dropbox/iscroll4/examples/carousel/)有个小例子。默认情况下，iScroll将scroller分成四分体，或者是相同大小的wrapper。iScroll 4加入了这个属性允许捕捉scroller里的 任何元素，不考虑外部wrapper的大小。如果你想要实现这样的传送带效果，我建议你使用“quadrant”分割。最佳的设置如下：

                 var myscroll=new iScroll("wrapper",{

                       snap:true,

                       momentum:false,

                       hScrollbar:false,

                       vScrollbar: false

                  });

          捕获元素，可以通过传递一个字符串来作为查询条件，如下：

                var myscroll=new iScroll("wrapper",{

                      snap:"li",

                      momentum:false,

                      hScrollbar:false,

                      vScrollbar:false

                });

           在这个例子中scroller可以捕捉到滚动区域中最左上角的li元素

**自定义滚动条（custom scrollbars）**

          在iScroll 4这个版本中，可以利用一系列的css来自定义滚动条的呈现。可以给滚动条添加一个class参数，如下：

                  var myscroll=new iScroll("wrapper",{

                     scrollbarClass: "myScrollbar"

                 });

          你可以[点击这里](http://cubiq.org/dropbox/iscroll4/examples/custom-scrollbar/)看一个小例子，在这个小例子里，myScrollbarH这个类被添加到了水平滚动条上，myScrollbarV这个类则被添加给了垂直方向上的滚动条上了。需要提醒的是，滚动条是由两个元素组合而成的：容器和显示器。容器同wrapper的高度相同，而显示器则代表的是滚动条本身。

           滚动条的HTML结构如下：

                  <div class="myScrollbarV">

                          <div></div>

                  </div>

                 .myscrollbarV{

                           position:absolute;z-index:100;width:8px;bottom:7px;top:2px;right:1px;

                  }

                .myScrollbarV > div {  
                           position:absolute;  
                           z-index:100;  
                           width:100%;  
                             /\* The following is probably what you want to customize \*/  
                           background:-webkit-gradient(linear, 0 0, 100% 0, from(#f00), to(#900));  
                           border:1px solid #800;  
                          -webkit-background-clip:padding-box;  
                          -webkit-box-sizing:border-box;  
                          -webkit-border-radius:4px;  
                          -webkit-box-shadow:inset 1px 1px 0 rgba(255,255,255,0.5);  
                 }

**通用方法：**

     （1）refresh                          在DOM树发生变化时，应该调用此方法

     （2）scrollTo()                     滚动到某个位置

               eg: myscroll.scrollTo(0,10,200,true);

     （3）scrollToElement()      滚动到某个元素

               eg: myscroll.scrollToElement("li:nth-child(10)",100);

     （4）detroy()                       完全消除myscroll及其占用的内存空间

                eg: myscroll.destroy()

【添加】

iScroll框架在部分三星手机上无法滑动或点击的解决方案

**一、测试平台**

    三星 i9023   android 4.1.2   （正常）

    三星 i9250   android 4.2.2   （存在问题）

    三星 i9300   android4.1.1    （存在问题）

    三星 i9308   android 4.0.4   （存在问题）

**二、问题描述**

    上述手机出现无法滑动或者无法触发点击事件的情况

**三、问题分析**

    1、onBeforeScrollStart方法

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

       此方法是在\_start里进行了调用，目的是为了阻止浏览器默认动作的执行，防止在

          滑动的过程中进行干扰，同时也就阻止了滑动区域里元素的事件的触发，这种处

          理方式也直接导致了必须要在\_end方法中再次触发元素的点击事件

    2、\_end方法

       if (target.tagName != 'SELECT' && target.tagName != 'INPUT'

             && target.tagName != 'TEXTAREA') {

           ev = document.createEvent('MouseEvents');

           ev.initMouseEvent('click', true, true,

                             e.view, 1,point.screenX,

                             point.screenY, point.clientX,

                             point.clientY,e.ctrlKey,

                             e.altKey, e.shiftKey,

                             e.metaKey,0, null);

            ev.\_fake = true;

            target.dispatchEvent(ev);

        }

        这个处理方式就是顺承了上面所提到的阻止了浏览器默认行为后，对滑动区域

            除SELECT、INPUT、TEXTAREA外的元素触发click事件，已完成对click绑定事件的调用

**【具体分析】**

         由于onBeforeScrollStart是在\_start方法中进行的调用，e.preventDefault();阻止了元

         素的默认行为，从而导致了元素绑定事件的失效，必须在\_end操作结束后进行

             绑定事件的模拟调用，原始的iScroll源代码中在\_end中最后创建了click事件的

             模拟，但是这里必须要清楚的一个原理就是，click其实是要依赖于其他事件的：

         1)   普通pc网页中，click需要依赖于mousedown、mouseup的相继触发

         2)   移动webkit中，click则需要依赖于touchstart、touchend

            （实际mousedown,mouseup在移动webkit上也存在）的相继触发

  “相继触发”的意思就是中间不会夹杂有其他的事件类型，这也就很容易理解

     iScroll中在\_end中对模拟事件调用的条件了，必须要判断that.moved才能直接触

    发模拟事件。iScroll中与\_start、\_move、\_end相关的三个事件类型是按照如

   下的规则来设置的：

        START\_EV = hasTouch ? 'touchstart' : 'mousedown'

        MOVE\_EV = hasTouch ? 'touchmove' : 'mousemove'

        END\_EV = hasTouch ? 'touchend' : 'mouseup'

    【注意事项】

          经测试，三星部分手机里默认浏览器里会默认所有元素都有一个默认的click

          事件（测试结果显示鼠标事件中默认事件包含mouseup,mousedown,click,

                    dblclick在移动webkit上不支持，为系统放大功能），e.preventDefault时会阻止

          掉默认click事件的执行，必须要人为的在\_end结束之后模拟click事件的调用，

               而其他正常手机即使调用了e.preventDefault也不会阻止click事件的触发，因为

               这些手机的默认浏览器上的元素的click事件的cancelable属性不为true，不可

               以被preventDefault取消掉，会正常执行，而如果在\_end中模拟了click事件则

               将会导致click的重复调用（在有toggle状态的事件上非常明显），因此折中

               的方式参见下面的解决方案。

**四、解决方案**

       1. 去除onBeforeScrollStart里的阻止默认行为

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

       2. onBeforeScrollMove设置为

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

         以保证手机上的正常滑动免受浏览器默认行为影响（如下滑时会有窗口的scroll事件），

         当然如果这里不添加的话也可以在document的END\_EV中阻止浏览器默认行为

      3.  \_end 中将模拟事件名更改为END\_EV或者直接去掉模拟事件的功能

iscroll.js是Matteo Spinelli开发的一个js文件，使用原生js编写，不依赖与任何js框架。旨在解决移动webkit系浏览器的区域滚动问题，兼容mobile safari、android默认浏览器、safari、chrome、firefox5+、opera11+、IE9+及其他webkit核心浏览器。最新版本为iscroll4。   
  
  
iscroll的用武之地   
  
1.区域滚动   
  
我们在pc端web开发中，有时会用固定某一区域的宽度和高度，然后使用overflow:auto，使其内容在该区域内滚动。   
  
iphone默认浏览器（mobile safari）也支持固定区域的滚动，但必须双指滑动操作，而且没有滚动条。mobile safari中的单指滑动只针对页面级别，不针对页面元素。这样的操作体验很麻烦，而且很多用户根本不知道双指能够区域滑动。   
  
android自带浏览器也支持区域滚动，且可以单指滑动操作，但是滑动起来卡的半死半活，很不流畅。   
  
使用iscroll,可以完美解决上述问题了，不仅可以在iphone和android上单指滑动，而且滑动起来流畅之极，酣畅淋漓。滑动过程中也有漂亮的滚动条提示，让你舒心不已。   
  
2.固定定位   
  
固定定位不是iscroll的原生用法，而是使用iscroll协助解决固定定位问题。   
  
话说iphone很先进，但就是不支持position:fixed。这下苦了我们了，固定定位怎么解决啊，我们会经常遇到固定标题栏、固定工具栏等情况啊！！   
  
使用iscroll协助解决：首先获取浏览器窗口高度；然后获取固定工具栏的高度；接着将除固定工具栏之外的内容全部放在一个固定区域内，该固定区域的高度=窗口高度-工具栏高度；之后对固定区域使用iscroll。这样，整个html页面的高度正好等于窗口高度，页面级别不会出现滚动，工具栏就一直固定在当前的位置。在固定区域内滑动，可以查看页面其他内容，不会影响工具栏的定位。   
  
  
鼠标滚轮滚动   
  
iscroll支持在pc端浏览器中使用鼠标滚轮控制区域滚动，但操作起来很不灵敏。这是由于iscroll对鼠标滚轮事件做了拦截，然后缩小了滚轮的滚动距离，详见iscroll4.js源代码608-609行：   
  
wheelDeltaX = e.wheelDeltaX / 12;//控制X轴鼠标滚轮速度   
wheelDeltaY = e.wheelDeltaY / 12;//控制Y轴鼠标滚轮速度   
    iscroll将每次的滚轮距离缩小为系统默认距离的12分之一，难怪滚起来很慢，感觉不灵敏。只需要将12改成1，滚轮的滚动速度就恢复正常了。你也可以根据实际需要设置成其他值。   
输入框聚焦不灵敏、无法选择文字   
  
    使用了iscroll之后，你会发现点击输入框时不灵敏，经常无法聚焦；页面文字也无法选择和复制。这是由于iscroll要监听鼠标事件和触摸事件来进行滚动，所以禁止了浏览器的默认行为，详见源代码92行：   
onBeforeScrollStart: function (e) { e.preventDefault(); },   
iscroll不分青红皂白，禁止了浏览器的一切默认行为，导致上述问题。所以我们需要稍作修改：   
onBeforeScrollStart: function (e) {   
var target = e.target;   
while (target.nodeType != 1) target = target.parentNode;   
if (target.tagName != ‘SELECT’ && target.tagName != ‘INPUT’ && target.tagName != ‘TEXTAREA’)   
e.preventDefault();   
},   
    判断触发事件的元素是不是input、select、textarea等表单输入类元素，如果不是就阻止默认行为。这样保证了输入类元素能正确快速聚焦。   
在上述修改中追加一个条件 target.tagName != ‘body’或者直接将onBeforeScrollStart修改为null，即可做到鼠标选取文字，但这样的修改会导致iscroll滚动失效或不灵敏，所以无法选取文字这个问题先放一边吧。如你有更好的方法，欢迎回复。   
select元素操作不灵敏或操作后白屏   
  
    使用iscroll后，还有一个较麻烦的问题，就是select元素点击不灵敏（滚动之后点击select，经常无响应）；有时点击有反应了，下拉菜单显示却错位（pc端）；选择一项之后，页面直接变成了空白（iphone、android中）或者页面位置向上偏离，滚动区域位置出现偏离。   
    说白了，就是导致select没法用。为此我调试了很久，最后发现了问题所在：iscroll默认使用的是css的transform变形中的translate3d实现区域滚动，每次滚动实际是改变translate3d中的y轴值，实际的dom位置并没有发生变化。translate3d会导致页面的焦点错误，系统级下拉菜单的显示则会导致其出现显示偏离。   
    控制滚动模式的代码在67行：useTransform: true。好在iscroll提供了另一种滚动方式，基于绝对定位位置变化的滚动。修改为useTransform: false之后，iscroll就会使用对定位位方式来实现滚动，该方式是我们在web开发中模拟动画的最常用方式，滚动之后dom的实际位置也同步发生了变化，不会出现错位偏离现象。   
    在pc端主流浏览器、iphone、android2.2下测试，select问题完美解决。   
    不过需要注意，使用对position决定定位后，滚动区的宽度默认会自适应内容宽度，而不是父元素的100%，所以最好将滚动区域宽度设为100%。   
美中不足   
  
    iscroll是小而精的经典作品，名字也带着apple范。但唯一美中不足的是，只能使用id调用。

**最近做移动平台的应用，使用iscroll使屏幕上下滑动。发现当使用iscroll后，input等不能输入内容了。只要在iscroll.js文件中加入如下代码就ok了。、**

function allowFormsInIscroll(){

[].slice.call(document.querySelectorAll('input, select, button')).forEach(function(el){

el.addEventListener(('ontouchstart' in window)?'touchstart':'mousedown', function(e){

e.stopPropagation();

})

})

}

document.addEventListener('DOMContentLoaded', allowFormsInIscroll, false);