**iScroll基本信息**

官网：http://cubiq.org/iscroll-4

更新：2012.07.14

版本：v4.2.5

兼容：iPhone/Ipod touch >=3.1.1, iPad >=3.2, Android >=1.6, Desktop Webkit, Firefox, Opera desktop/mobile.

[下载iScroll4.2.5](http://www.gafish.net/api/down/iScroll%20v4.2.5.js" \t "_blank)

**iScroll的历史**

之所以iscroll会诞生，主要是因为无论是在iphone、ipod、android 或是更早前的移动webkit都没有提供一种原生的方式来支持在一个固定高度的容器内滚动内容。

这个不幸的规则导致所有web-app要模拟成app的样子时，只能由一个绝对定位的header 或是footer再加上一个可以内容的滚动的中间区域组成。

幸运的是移动webkit提供了一种强大的硬件加速的CSS属性，这个属性可以用来模拟这个缺失的功能，Iscroll从这里开始了前进之路，但是没有不带刺的玫瑰。让内容滚动像原生方式一般比想象中要难

**Tigs：**截至2013年，IOS5 添加如下样式 overflow:scroll; -webkit-overflow-scrolling:touch; 已经能够支持区域滚动了。但是andriod4 还是不行...

**iScroll 4 新特性**

iScroll 4 这个版本完全重写了iScroll 3这个框架的原始代码，除了以前版本的iScroll特性以外，iScroll 4 还包括如下的特性：

1. 缩放
2. 拉动刷新
3. 速度和性能提升
4. 精确捕捉元素
5. 自定义滚动条

**快速上手**

**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

**实例化iScroll**

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

**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 参数**

iScroll里的第一个参数很简单就是外容器的id，或者css语法格式的class，如'.home .scroll'。

第二个参数允许你自定义一些属性配置和方法，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。所有参数如下（\*号为常见参数）：

| 基本参数 | | | |
| --- | --- | --- | --- |
| **参数名** | **说明** | **可选值** | **默认值** |
| hScroll \* | 是否允许水平滚动 | false 否 true 是 | true |
| vScroll \* | 是否允许垂直滚动 | false 否 true 是 | true |
| bounce \* | 是否超过实际位置反弹 | false 否 true 是 | true |
| bounceLock | 当内容少于滚动是否可以反弹 | false 否 true 是 | false |
| momentum\* | 是否开启拖动惯性 | false 否 true 是 | true |
| lockDirection | 当水平或垂直拖动时是否锁定另一边的拖动 | false 否 true 是 | true |
| useTransform | 是否使用CSS变形 | false 否 true 是 | true |
| useTransition | 是否使用CSS变换 | false 否 true 是 | false |
| checkDOMChanges | 是否自动检测内容变化 | false 否 true 是 | false |
| topOffset | 已经滚动的基准值(一般用在拖动刷新) | 数字值 | 0 |
| x | 滚动水平初始位置（负值） | 数字值 | 0 |
| y | 滚动垂直初始位置（负值） | 数字值 | 0 |

| Scrollbar 的相关参数 | | | |
| --- | --- | --- | --- |
| **参数名** | **说明** | **可选值** | **默认值** |
| hScrollbar \* | 是否显示水平滚动条 | false否 true 是 | true |
| vScrollbar \* | 是否显示垂直滚动条 | false否 true 是 | true |
| 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。 | false否 true 是 | 默认在Android上为true， iOS上为false |
| hideScrollbar | 是否隐藏滚动条 | false否 true 是 | 默认在Android上为false， iOS上为true |
| fadeScrollbar | 滚动条是否渐隐渐显 | false否 true 是 | 默认在Android上为false， iOS上为true |
| scrollbarClass \* | 字定义滚动条的样式名 | 自定义class | '' |

| Zoom 放大相关的参数 | | | |
| --- | --- | --- | --- |
| **参数名** | **说明** | **可选值** | **默认值** |
| zoom | 是否放大 | false 否 true 是 | false |
| zoomMin | 放大的最小倍数 | 数字值 | 1 |
| zoomMax | 放大的最大倍数 | 数字值 | 4 |
| doubleTapZoom | 双击放大倍数 | 数字值 | 2 |
| wheelAction | 鼠标滚动行为（还可以是zoom） | none 不支持 'scroll' 鼠标滚动 'zoom' 鼠标缩放 | 'scroll' |

| snap 捕捉元素相关的参数 | | | |
| --- | --- | --- | --- |
| **参数名** | **说明** | **可选值** | **默认值** |
| snap | 是否开启捕捉元素，snap值可以为true 或是 DOM元素的tagname，当为true 时，捕捉的元素会根据可滚动的位置和滚动区域计算得到可滑动几页。如果为tagname,则滑动会对齐到捕捉的元素上。 | false 否 true 是< DOM元素的tagname | false |
| snapThreshold | 滑动的长度限制，只有大于这个距离才会触发捕捉元素事件 | 数字值 | 1 |

| 自定义 Events 的相关参数 | | | |
| --- | --- | --- | --- |
| **参数名** | **说明** | **可选值** | **默认值** |
| onBeforeScrollStart | 开始滚动前的事件回调 默认是阻止浏览器默认行为 | null 不设置  Function 自定义函数 | function (e) { e.preventDefault(); } |
| onRefresh | refresh 的回调 | null |
| onScrollStart | 开始滚动的回调 |
| onBeforeScrollMove | 在内容移动前的回调 |
| onScrollMove | 内容移动的回调 |
| onBeforeScrollEnd | 在滚动结束前的回调 |
| onScrollEnd | 在滚动完成后的回调 |
| onTouchEnd | 手离开屏幕后的回调 |
| onDestroy | 销毁实例的回调 |
| onZoomStart | 放大开始时的回调 |
| onZoom | 放大的回调 |
| onZoomEnd | 放大结束后的回调 |

**iScroll 方法API**

**destroy**

销毁你实例化的iScroll 实例，包括之前绑定的所有iscroll 事件。

**refresh**

这个方法非常有用，当你的滚动区域的内容发生改变 或是 滚动区域不正确，都用通过调用refresh 来使得iscroll 重新计算滚动的区域，包括滚动条，来使得iscroll 适合当前的dom。

**scrollTo**

这个方法接受4个参数 x, y, time, relative x 为移动的x轴坐标，y为移动的y轴坐标, time为移动时间，relative表示是否相对当前位置。

**scrollToElement**

这个方法实际上是对scrollTo的进一步封装，接受两个参数(el,time)，el为需要滚动到的元素引用，time为滚动时间。

**scrollToPage**

此方法接受三个参数(pageX,pageY,time) 当滚动内容的高宽大于滚动范围时，iscroll 会自动分页，然后就能使用scrollToPage方法滚动到页面。当然，当hscroll 为false 的时候，不能左右滚动。pageX这个参数就失去效果

**currPageX/currPageY 获取当前页面**

**disable**

调用这个方法会立即停止动画滚动，并且把滚动位置还原成0，取消绑定touchmove, touchend、touchcancel事件。

**enable**

调用这个方法，使得iscroll恢复默认正常状态

**stop**

立即停止动画

**zoom**

改变内容的大小倍数,此方法接受4个参数，x,y,scale,time 分别表示的意思为，放大的基准坐标，以及放大倍数，动画时间

**isReady**

当iscroll 没有处于正在滚动，没有移动过，没有改变大小时，此值为true

**拉动刷新**

**说明**

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[点击这里（需要翻墙）](http://vimeo.com/20475828" \t "_blank)看看视频先一睹为快，也可以[点击这里](http://www.gafish.net/api/demo/pull-to-refresh/index.html" \t "_blank)试试拉动刷新是如何工作的。

**实现原理**

1. 通过 topOffset 参数属性设置iScroll已经滚动的基准值；
2. 通过 onScrollMove 参数方法判断当前滚动是到顶端还是底端；
3. 通过 onScrollEnd 参数方法触发加载新数据，再通过 refresh 方法重新渲染界面；
4. 通过 onRefresh 参数方法调整刷新后的界面结构；

<script>

var myScroll =new iScroll("wrapper",{

topOffset: 50,

onRefresh: function(){...},

onScrollMove: function(){...},

onScrollEnd: function()...{}

});

</script>

具体的实现代码可以查看[这里](http://www.gafish.net/api/demo/pull-to-refresh/index.html" \t "_blank)的源文件；

**缩放**

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。

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：

<script>

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

</script>

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：

| **参数名** | **说明** |
| --- | --- |
| 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**

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/" \t "_blank)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：

<script>

var myscroll=new iScroll("wrapper",{

snap:true,

momentum:false,

hScrollbar:false,

vScrollbar: false

});

</script>

捕获元素，可以通过传递一个字符串来作为查询条件，如下：

<script>

var myscroll=new iScroll("wrapper",{

snap:"li",

momentum:false,

hScrollbar:false,

vScrollbar: false

});

</script>

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**

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：

<script>

var myscroll=new iScroll("wrapper",{

scrollbarClass: "myScrollbar"

});

</script>

你可以[点击这里](http://cubiq.org/dropbox/iscroll4/examples/custom-scrollbar/" \t "_blank)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);

}