

javascripttable:JavaScript Table行定位效果

疯狂代码 <http://CrazyCoder.cn/> j:<http://CrazyCoder.cn/Javascript/Article65627.html>

近来有客户要求用table显示大串数据由于滚动后就看不到表头很不方便所以想到这个效果
上次做table排序对table有了些了解这次更是深入了解了番发现table原来是这么不简单
还不清楚这个效果叫什么就叫行定位吧本来想把列定位也做出来但暂时还没这个需求等以后有时间再弄吧
如果只是做个效果也不难但要做个通用无侵入就要考虑很多东西了

效果预览

为方便预览建议缩小浏览器

1 表头

1 表头

2 图片滑动切换效果

3 图片变换效果(ie _disibledevent=> <body>

```
<table cellpadding="5" cellspacing="0" border="1" width="100">
<tr style="position:relative; left:100px;">
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
</table>
</body>
</html>
```

给tr设置relative后就能相对table定位了看来很简单啊但问题是这个思路方法ie8和ff都无效而且存在很多问题
所以很快就被抛弃了

ps:该效果用来做tr拖动会很方便

接着想到是给table插入个新tr克隆原来tr并设置这个tr为fixed(ie6为absolute)例如:

Code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<table cellpadding="5" cellspacing="0" border="1" width="100">
<tr style="position:fixed; left:100px;">
<td>1</td>
<td>2</td>
</tr>
<tr style="position:absolute; left:200px;">
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
</table>
</body>
</html>
```

第个问题是fixedtr在ie7中不能进行定位而且td在定位后并不能保持在表格中布局这样在原表格插tr就没意义了
ps:fixed相关应用可参考仿LightBox效果

最后我用思路方法是新建个table并把源tr克隆到新table中然后通过对新table定位来实现效果
用这个思路方法关键有两点首先要做个仿真度尽可能高tr还有是要准确定位这些请看后面介绍说明

介绍说明

【克隆table】

克隆个元素用cloneNode就可以了它有个bool参数表示克隆是否包含子节点
第步就是克隆原table:

```
this._oTable = $(table);//源table
this._nTable = this._oTable.cloneNode(false);//新table
this._nTable.id = "";//避免id冲突
```

要注意虽然iecloneNode参数是可选(默认是false)但在ff是必须建议使用时都写上参数
还要注意是id属性也会被克隆也就是克隆后会有两个相同id元素(如果克隆对象有设置话)这很容易会导致其他问题会把克隆tableid属性设空
ps:table请用来绑定样式用id话新table就获取不了样式了

克隆的后再设置样式:

```
this._style.width = this._oTable.offWidth + "px";
this._style.position = isIE6 ? "absolute" : "fixed";
this._style.zIndex = 100;
```

般来说offWidth是width+padding+border结果但table比较特别测试下面代码:

Code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<table border="5" id="t" style="padding:10px; width:100px;">
<tr>
<td>1</td>
<td>1</td>
</tr>
</table>
<table width="100" id="t2" style="border:10px solid #000;">
<tr>
<td>1</td>
<td>1</td>
</tr>
</table>
<script>
alert(document.getElementById("t").offWidth);
```

```
alert(document.getElementById("t2").offsetWidth);
</script>
</body>
</html>
```

只要给table设置width(style或本身width属性)不管设置padding和border是多少offsetWidth都等于width值
经测量offsetWidth是没错那就是说是tablewidth设置问题

w3c table部分中说width属性是the desired width of the entire table我估计entire就是包含了padding和border找不到什么其他介绍说明先这么理解吧

定位方面除了不支持fixed用absolute其他都使用fixed定位

【克隆tr】

table有个rows集合包括了table所有tr(包括thead和tfoot里面)

Clone思路方法会根据其参数克隆对应索引tr:

```
this._index = Math.max(0, Math.min(this._oTable.rows.length - 1, isNaN(index) ? this._index : index));
this._oRow = this._oTable.rows[this._index];
var oT = this._oRow, nT = oT.cloneNode(true);
```

由于tr可能是包含在thead这些中所以还要判断下:

```
(oT.parentNode !== this._oTable){
nT = oT.parentNode.cloneNode(false).appendChild(nT).parentNode;
}
```

然后再插入到新table中:

```
(this._nTable.firstChild){
this._nTable.replaceChild(nT, this._nTable.firstChild);
}{
this._nTable.appendChild(nT);
}
```

允许修改克隆tr所以会判断有没有插入过没有就直接appendChild否则用replaceChild替换原来tr

【tableborder和frame属性】

tableborder属性用来指定边框宽度table特有frame属性是用来设置或获取表格周围边框显示方式
w3ctableframe部分介绍说明frame可以是以下值:

void: No sides. This is the default value.

above: The top side _disibledevent=> <body>

```
<table width="100" border="5" frame="lhs">
```

```
<tr>
```

```
<td>1</td>
```

```
<td>1</td>
```

```
</tr>
```

```
</table>
```

```
<table width="100" style="border:5px solid #000;" border="10" frame="lhs">
```

```
<tr>
```

```
<td>1</td>
```

```
<td>1</td>
```

```
</tr>
```

```
</table>
```

```
</body>
```

```
</html>
```

这里还可以看到如果同时设置tableborder和styleborder那tableborder就会失效

中为了更美观会自动去掉新table上面和下面边框包括frame和style:

Code

```
(this._oTable.border > 0){  
switch (this._oTable.frame) {  
  "above" :  
  "below" :  
  "hsides" :  
this._nTable.frame = "void";;  
  "" :  
  "border" :
```

```
"box" :
this._nTable.frame = "vsides";
}
}
this._style.borderTopWidth = this._style.borderBottomWidth = 0;
```

其中空值在设置collapse的后会比较麻烦在ie6/ie7中测试:

Code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" >
<html xmlns="http://www.w3.org/1999/xhtml" >
<body>
<style type="text/css">
.t{width:100px; border-collapse:collapse;}
.t td{border:5px solid #999;}
</style>
<table = "t" >
<tr>
<td>1</td>
<td>1</td>
</tr>
</table>
<br />
<table = "t" frame="vsides" >
<tr>
<td>1</td>
<td>1</td>
</tr>
</table>
<br />
<table = "t" border="1" >
<tr>
<td>1</td>
<td>1</td>
</tr>
```

```
</table>
<br />
<table = "t" border="1" frame="vsides">
<tr>
<td>1</td>
<td>1</td>
</tr>
</table>
</body>
</html>
```

后两个转换还可以接受所以在设置frame的前还是判断下border先

【获取背景色】

如果td是背景透明话显然不太美观最好是找个合适颜色来填充

用思路方法是从当前td开始找如果背景是透明话就再从父节点中找直到找到有背景色为止

一般来说透明属性值是"transparent"但在chrome里却是"rgba(0, 0, 0, 0)"所以用了个属性来保存透明值:

```
this._transparent = isChrome ? "rgba(0, 0, 0, 0)" : "transparent";
```

并在GetBgColor获取背景色中使用:

```
while (bgc this._transparent && (node = node.parentNode) != document) {
bgc = CurrentStyle(node).backgroundColor;
}
bgc this._transparent ? "#fff" : bgc;
```

如果全部都是透明话就会返回白色(#fff)

这里没有考虑图片背景情况毕竟图片不一定会覆盖整个背景

【parentNode/offParent/parentElement】

上面用到了parentNode这里顺便说说它跟offParentparentElement区别

先看看parentNode在w3c介绍说明:

The parent of this node. All nodes, except Document, DocumentFragment, and Attr may have a parent. However, a node has just been created and not yet added to the tree, or it has been removed from the tree, this is null.

很简单就是节点父节点看过dom都知道

再看看比较容易区分offParent它在mozilla和msdn都说得比较模糊在w3c就比较清楚了:

The offParent attribute, when called _disibledevent=> <body>

```
<table width="100" id="t" >
<tr>
<td><div id="t1"></div></td>
<td id="t2"><div style="position:absolute;">
<div id="t3"></div>
</div></td>
</tr>
</table>
<div id="t4" style="position:fixed;"></div>
<script>
var $ = function (id) {
  "" typeof id ? document.getElementById(id) : id;
};

alert($("#t").offParent)//body
alert($("#t1").offParent)//td
alert($("#t2").offParent)//table
alert($("#t3").offParent)//div
alert($("#t4").offParent)//null
</script>
</body>
</html>
```

可见offParent跟parentNode区别还是很大

而parentNode跟parentElement除了前者是w3c标准后者只ie支持其他区别就不是那么明显了
在ie中大部分情况下两者效果是样当然如果是模样话ie就没必要弄这么个东西出来了测试下面代码:

Code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" >
<html xmlns="http://www.w3.org/1999/xhtml" >
<body>
<script>
var o = document.createDocumentFragment.appendChild(document.createElement("div"));
alert(o.parentNode)
alert(o.parentNode.nodeType)//11
alert(o.parentElement)//null

alert(document.body.parentNode)
alert(document.body.parentNode.nodeType)//1
alert(document.body.parentElement)//html

alert(document.body.parentNode.parentNode)
alert(document.body.parentNode.parentNode.nodeType)//9
alert(document.body.parentElement.parentElement)//null
</script>
</body>
</html>
```

可以看到当父节点nodeType不是1即不是element节点话它parentElement就会是null
这就明白了名字中“Element”含义了

【设置td宽度】

接下来就要设置td宽度了要获取某元素宽度可以通过以下思路方法:

1支持defaultView可以直接用getComputedStyle获取width

2获取offsetWidth再减去border和padding宽度

这个本来也可以但tdborder宽度获取比较麻烦下面有更方便思路方法

3获取clientWidth再减去padding宽度

这个跟思路方法2差不多但更简单方便

注意iecurrentStyle不像getComputedStyle能获取准确值而只是个设置值像百分比、auto这些并不会自动转

成准确值即使是得到准确值也不定是实际值像td即使设置个很大准确值实际值也不会超过table本身宽度所以在td这种比较特殊结构中除非是很理想状况否则用currentStyle基本没戏而且在这个效果中即使是差了1px也会看不舒服

对于支持defaultView当然可以直接获取否则就用上面思路方法3来获取:

```
style.width = (document.defaultView ? parseFloat(css.width)
: (o.clientWidth - parseInt(css.paddingLeft) - parseInt(css.paddingRight))) + "px";
```

但这里不管哪个思路方法都有个问题就是出现scroll情况不过还好td这个元素即使设置了overflow为scroll也不会出现滚动条除了ie8和chrome

没对这个情况做处理毕竟给td设scroll也不常见而且支持这个浏览器不多没必要花太多时间在这里

ps:有关td宽度自动调整可以参考w3ctable-layout部分

如果有影响原td结构设置例如colspan的类就要留意结构很可能导致些异常变形

如果对原表格结构或内容做了修改应该执行次Clone思路方法重构新table

本部分对体验比较重要如果设置不当就会有变形感觉很不美观

【borderCollapse】

上面说到tdborder宽度获取比较麻烦那到底有多烦呢？

如果只是般情况话通过borderLeftWidth和borderRightWidth获取宽度就可以了

ps:如果borderStyle是"none"话那么border就会没效所以如果要取border宽度话最好先判断下borderStyle是不是"none"

但table有个特别样式borderCollapse设置table边框模型

它有两个值分别是separate(分开默认值)和collapse(合并)

separate就是我们般看到效果这里主要讨论collapse先看mozilla如何说:

In the collapsed border model, adjacent table cells share borders.

意思是在collapse border模型中相邻td会共用边框看下面例子会更明白:

Code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
```

```

<style type="text/css">
.t{line-height:40px;width:200px;}
.t td{border:5px solid #999;}
</style>
<table = "t" style="border-collapse:collapse;">
<tr>
<td width="50">&nbsp;</td>
<td style="border-left-width:10px; border-left-style:dotted;">&nbsp;</td>
<td width="50">&nbsp;</td>
</tr>
</table>
<table = "t">
<tr>
<td width="50">&nbsp;</td>
<td style="border-left-width:10px; border-left-style:dotted;">&nbsp;</td>
<td width="50">&nbsp;</td>
</tr>
</table>
</body>
</html>

```

可以看到使用collapse的后相邻td边框都合并成条而且是以相邻边框中宽度较大那条为准那td跟table的间呢参考下面例子:

Code

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<style type="text/css">
.t{line-height:40px;width:200px;border-collapse:collapse;}
.t td{border:5px solid #999;}
</style>
<table = "t" id="t1">
<tr>
<td width="50" style="border-left:10px dotted #999;">&nbsp;</td>

```

```

<td>&nbsp;</td>
<td width="50">&nbsp;</td>
</tr>
</table>
<br />
<table = "t" id="t2" style="border-left:10px dotted #999;">
<tr>
<td width="50">&nbsp;</td>
<td>&nbsp;</td>
<td width="50">&nbsp;</td>
</tr>
</table>
</body>
</html>

```

可见table和td的间也是遵从同样规则

还有是当设置了collapse那cellspacing就无效了顺便说说border-spacing它其实就是cellspacing在css中样式形式只是ie在ie8才开始支持详细可以看mozilla介绍说明

collapse个常见应用是做边框表格例如1px边框表格:

Code

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<style type="text/css">
.t{line-height:40px;width:200px;}

.t1{border-collapse:collapse;}
.t1 td{border:1px solid #999;}

.t2{background-color:#999;}
.t2 td{background-color:#FFF;}
</style>
<table = "t t1">

```

```

<tr>
<td width="50">&nbsp;</td>
<td>&nbsp;</td>
<td width="50">&nbsp;</td>
</tr>
</table>
<table border="1" cellspacing="1">
<tr>
<td width="50">&nbsp;</td>
<td>&nbsp;</td>
<td width="50">&nbsp;</td>
</tr>
</table>
</body>
</html>

```

前者用collapse后者是用table背景色模拟虽然效果都样但前者显然较好才是真正“边框”

在使用了collapse的后要写个通用获取边框宽度会变得十分麻烦而且有些情况下甚至没办法判断获取详细情况这里就不细说了有兴趣研究话可以看看w3cThe collapsing border model当然要想全部了解话还要在各个浏览器中研究

【元素位置】

table样式设置好后还需要获取原table和原tr位置参数为后面元素定位做准备
 要获取某个元素相对文档位置传统做法是获取对象offsetLeft/offsetTop然后不断获取offsetParent/offsetLeft/offsetTop直到找不到offsetParent为止
 得到结果就是相对文档位置了上面已经介绍过offsetParent原理应该都明白了吧
 SetRect设置区域属性思路方法中也使用了这个思路:

Code

```

//获取原table位置
var o = this._oTable, iLeft = o.offsetLeft, iTop = o.offsetTop;
while (o.offsetParent) { o = o.offsetParent; iLeft = o.offsetLeft; iTop = o.offsetTop; }
this._oTableLeft = iLeft;

```

```
this._oTableTop = iTop;
this._oTableBottom = iTop + this._oTableHeight;
//获取原tr位置
o = this._oRow; iTop = o.offTop;
while (o.offParent) { o = o.offParent; iTop = o.offTop; }
this._oRowTop = iTop;
this._oRowBottom = iTop + this._oRow.offHeight;
```

不过这里介绍个更好思路方法通过getBoundingClientRect思路方法来获取

在mozilla是这么介绍说明:

The ed value is a TextRectangle object, which contains read-only left, top, right and bottom properties describing the border-box, in pixels, with the top-left relative to the top-left of the viewport...

返回个TextRectangle对象包含left, top, right和bottom几个只读属性以px为单位来表示边界框相对视窗左上角位置(偶英文烂啊)

注意是相对视窗不是文档哦如果要相对文档还必须加上scrollLeft/scrollTop

通过下面测试可以看到两个思路方法返回结果都是相同:

Code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<style type="text/css">
.t{line-height:40px;width:200px; border:10px solid; margin-top:900px; margin-left:500px;}
</style>
<div = "t" id="t"></div>
<script>
var o = document.getElementById("t");

var rect = o.getBoundingClientRect;
var iLeft1 = rect.left + document.documentElement.scrollLeft, iTop1 = rect.top +
document.documentElement.scrollTop;

var iLeft2 = o.offLeft, iTop2 = o.offTop;
while (o.offParent) { o = o.offParent; iLeft2 = o.offLeft; iTop2 = o.offTop; }
```

```
alert(iLeft1+"_"+iLeft2)
alert(iTop1+"_"+iTop2)
</script>
</body>
</html>
```

中如果支持getBoundingClientRect就会用它来获取位置参数:

Code

```
//用getBoundingClientRect获取原table位置
var top = this._doc.scrollTop, rect = this._oTable.getBoundingClientRect();
this._oTableLeft = rect.left + this._doc.scrollLeft;
this._oTableTop = rect.top + top;
this._oTableBottom = rect.bottom + top;
//获取原tr位置
rect = this._oRow.getBoundingClientRect();
this._oRowTop = rect.top + top;
this._oRowBottom = rect.bottom + top;
```

显然用getBoundingClientRect更方便快捷

这个思路方法虽然是ie产物但已经是w3c标准而且ff3和Opera都已经支持了这个思路方法基本可以放心使用除了chrome

这里只是简单介绍想了解更多可以看w3cView Module部分

获取原table和tr位置后还需要计算新table位置

可以自定义新table位于视窗位置百分比例如顶部是0中间是0.5底部是1可以在化时或用SetPos思路方法来设置这里主要获取视窗高度和新table在视窗top值:

```
this._viewHeight = document.documentElement.clientHeight;
this._ntViewTop = (this._viewHeight - this._nTableHeight) * this._pos;
```

定位范围实际上是从视框顶部到视框高度减去新table高度范围内所以计算时要先把视窗高度减去新table高度

【元素定位】

万事俱备只欠定位了

由于要根据窗口滚动状态来判断计算定位scrollTop/scrollLeft获取必不可少

但在chrome中就算用了DOCTYPE也要用document.body来获取scrollTop/scrollLeft尽管它确实有document.documentElement

对chrome了解不多也不知哪里能查它相关文档里就直接做个判断算了:

```
this._doc = isChrome ? document.body : document.documentElement;
```

定位第步就是判断是否需要定位这里判断标准有两个第个是原tr是否超过了视窗范围还有是新table要显示位置是否在原table显示范围内

第点可以通过原tr位置顶部和底部是否超过视窗顶部和底部来判断:

```
var top = this._doc.scrollTop, left = this._doc.scrollLeft  
,outViewTop = this._oRowTop < top, outViewBottom = this._oRowBottom > top + this._viewHeight;  
(outViewTop || outViewBottom){}
```

在看第 2点的前先看看中Auto属性它是用来指定否自动定位

如果自动定位话当原tr离开视框顶部新table就会定位到视框顶部原tr离开底部新table就会定位到视框底部这样看上去会比较自然顺畅

如果不选择自动话就会根据SetPos思路方法中计算得到新table视窗top值来设置定位:

```
var viewTop = !this.Auto ? this._nTableViewTop  
: (outViewTop ? 0 : (this._viewHeight - this._nTableHeight))//视窗top  
,posTop = viewTop + top;//位置top
```

接着就判断新table要显示位置是否在原table显示范围内这个可以通过新table位置顶部和底部是否超过原table顶部和底部来判断:

```
(posTop > this._oTableTop && posTop + this._nTableHeight < this._oTableBottom){}
```

当符合所有条件就可以进行定位了如果是fixed定位就使用相对视窗top值:

```
this._style.top = viewTop + "px";  
this._style.left = this._oTableLeft - left + "px";
```



```
<select></select><br />
<select></select><br />
<select></select><br />
<select></select><br />
</body>
</html>
```

可以看到即使是rame在拖动滚动条时候select仍然在后面闪啊闪在本中这个现象会尤其明显
看来还得用隐藏select思路方法最好做法是只隐藏在新table后面select而不影响其他select正常显示
那关键就是如何判断select是否在新table后面这个可以通过位置坐标判断刚好可以用到上面
getBoundingClientRect
般思路是判断新table和select坐标根据位置判断select显示和隐藏
但如果有多例子可能会导致select在个例子中要隐藏却在另个要显示情况

为了解决冲突给select加了个_count属性作为计数器用来记录有多少例子把该select隐藏了
如果当前例子判断该select要隐藏就给其_count加1隐藏后存放到例子_selects集合中
在恢复显示_selects中select时先给select_count减1如果得到_count是0那介绍说明没有其他例子要隐藏它就可
以设置显示了最后清空_selects集合
在判断是否隐藏select前还必须恢复次该例子_selects里面select否则就会造成_count只加不减情况

中SetSelect思路方法就是用来判断和设置select:

Code

```
this.ReSelect;
var rect = this._nTable.getBoundingClientRect();
//把需要隐藏放到_selects集合
this._selects = Filter(this._oTable.getElementsByTagName("select"), Bind(this, function(o){
var r = o.getBoundingClientRect();
(r.top <= rect.bottom && r.bottom >= rect.top){
o._count ? o._count : (o._count = 1); //防止多个例子冲突
//设置隐藏
var visi = o.style.visibility;
(visi != "hidden"){ o._css = visi; o.style.visibility = "hidden"; }

true;
}
```

```
)))
```

其中ReSelect思路方法是用来恢复显示select:

```
forEach(this._selects, function(o){!--o._count && (o.style.visibility = o._css); });  
this._selects = ;
```

但这个思路方法在快速滚屏时还是无能为力而且select越多效率也随的下降各位有更好思路方法话欢迎交流

【Chrome个bug】

在测试时候发现Chrome个bug测试下面代码:

Code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" >  
<html xmlns="http://www.w3.org/1999/xhtml" >  
<body>  
<table border="1" >  
<tr>  
<td id="tt" > </td>  
</tr>  
</table>  
<div id="t" > </div>  
<script>  
document.getElementById("t").offWidth;  
document.getElementById("tt").innerHTML = "<select> <option>test</option> </select>";  
</script>  
</body>  
</html>
```

个毫不相干操作居然令table没有自动撑开加上前面问题看来Chrome路还很长啊

使用介绍说明

例子化个TableFixed对象只需要个参数tableid:

```
TableFixed("idTable");
```

例子化时有4个可选属性:

Index: 0, //tr索引

Auto: true, //是否自动定位

Pos: 0, //自定义定位位置百分比(0到1)

Hide: false //是否隐藏(不显示)

其中Index和Pos在例子化的后就不能使用

要修改克隆行可以用Clone思路方法其参数是要克隆tr索引

要修改自定义定位位置可以用SetPos思路方法其参数是要定位位置百分比

具体使用请参考例子

源码

Code

```
var TableFixed = function(table, options){
  this._oTable = $(table); //原table
  this._nTable = this._oTable.cloneNode(false); //新table
  this._nTable.id = ""; //避免id冲突

  this._oTableLeft = this._oTableTop = this._oTableBottom = 0; //记录原table坐标参数
  this._oRowTop = this._oRowBottom = 0; //记录原tr坐标参数
  this._viewHeight = this._oTableHeight = this._nTableHeight = 0; //记录高度
  this._nTableViewTop = 0; //记录新table视框top
  this._selects = ; //select集合用于ie6覆盖select
  this._style = this._nTable.style; //用于简化代码
  //chromescroll用document.body
  this._doc = isChrome ? document.body : document.documentElement;
  //chrome透明用rgba(0, 0, 0, 0)
  this._transparent = isChrome ? "rgba(0, 0, 0, 0)" : "transparent";
```

```
this.SetOptions(options);

this._index = this.options.Index;
this._pos = this.options.Pos;

this.Auto = !!this.options.Auto;
this.Hide = !!this.options.Hide;

addEventHandler(window, "resize", Bind(this, this.SetPos));
addEventHandler(window, "scroll", Bind(this, this.Run));

this._oTable.parentNode.insertBefore(this._nTable, this._oTable);
this.Clone;
};
TableFixed.prototype = {
//设置默认属性
SetOptions: function(options) {
this.options = {}//默认值
Index: 0,//tr索引
Auto: true,//是否自动定位
Pos: 0,//自定义定位位置百分比(0到1)
Hide: false//是否隐藏(不显示)
};
Extend(this.options, options || {});
},
//克隆表格
Clone: function(index) {
//设置table样式
this._style.width = this._oTable.offWidth + "px";
this._style.position = isIE6 ? "absolute" : "fixed";
this._style.zIndex = 100;
//设置index
this._index = Math.max(0, Math.min(this._oTable.rows.length - 1, isNaN(index) ? this._index : index));
//克隆新行
this._oRow = this._oTable.rows[this._index];
```

```

var oT = this._oRow, nT = oT.cloneNode(true);
(oT.parentNode != this._oTable){
nT = oT.parentNode.cloneNode(false).appendChild(nT).parentNode;
}
//插入新行
(this._nTable.firstChild){
this._nTable.replaceChild(nT, this._nTable.firstChild);
}
this._nTable.appendChild(nT);
}
//去掉table上面和下面边框
(this._oTable.border > 0){
switch (this._oTable.frame) {
"above" :
"below" :
"hsides" :
this._nTable.frame = "void";;
"" :
"border" :
"box" :
this._nTable.frame = "vsides";;
}
}
this._style.borderTopWidth = this._style.borderBottomWidth = 0;
//设置td样式
var nTds = this._nTable.rows[0].cells;
forEach(this._oRow.cells, Bind(this, function(o, i){
var css = CurrentStyle(o), style = nTds[i].style;
//设置td背景
style.backgroundColor = this.GetBgColor(o, css.backgroundColor);
//设置tdwidth,没考虑ie8/chrome设scroll情况
style.width = (document.defaultView ? parseFloat(css.width)
: (o.clientWidth - parseInt(css.paddingLeft) - parseInt(css.paddingRight))) + "px";
}));
//获取table高度
this._oTableHeight = this._oTable.offsetHeight;

```

```

this._nTableHeight = this._nTable.offHeight;

this.SetRect;
this.SetPos;
},
//获取背景色
GetBgColor: function(node, bgc) {
//不要透明背景(没考虑图片背景)
while (bgc this._transparent && (node = node.parentNode) != document) {
bgc = CurrentStyle(node).backgroundColor;
}
bgc this._transparent ? "#fff" : bgc;
},
//设置坐标属性
SetRect: function {
(this._oTable.getBoundingClientRect){
//用getBoundingClientRect获取原table位置
var top = this._doc.scrollTop, rect = this._oTable.getBoundingClientRect;
this._oTableLeft = rect.left + this._doc.scrollLeft;
this._oTableTop = rect.top + top;
this._oTableBottom = rect.bottom + top;
//获取原tr位置
rect = this._oRow.getBoundingClientRect;
this._oRowTop = rect.top + top;
this._oRowBottom = rect.bottom + top;
};//chrome不支持getBoundingClientRect
//获取原table位置
var o = this._oTable, iLeft = o.offLeft, iTop = o.offTop;
while (o.offParent) { o = o.offParent; iLeft o.offLeft; iTop o.offTop; }
this._oTableLeft = iLeft;
this._oTableTop = iTop;
this._oTableBottom = iTop + this._oTableHeight;
//获取原tr位置
o = this._oRow; iTop = o.offTop;
while (o.offParent) { o = o.offParent; iTop o.offTop; }
this._oRowTop = iTop;

```

```

this._oRowBottom = iTop + this._oRow.offHeight;
}
},
//设置新table位置属性
SetPos: function(pos) {
//设置pos
this._pos = Math.max(0, Math.min(1, isNaN(pos) ? this._pos : pos));
//获取位置
this._viewHeight = document.documentElement.clientHeight;
this._nTableViewTop = (this._viewHeight - this._nTableHeight) * this._pos;
this.Run;
},
//运行
Run: function {
(!this.Hide){
var top = this._doc.scrollTop, left = this._doc.scrollLeft
//原tr是否超过顶部和底部
,outViewTop = this._oRowTop < top, outViewBottom = this._oRowBottom > top + this._viewHeight;
//原tr超过视窗范围
(outViewTop || outViewBottom){
var viewTop = !this.Auto ? this._nTableViewTop
: (outViewTop ? 0 : (this._viewHeight - this._nTableHeight))//视窗top
,posTop = viewTop + top;//位置top
//在原table范围内
(posTop > this._oTableTop && posTop + this._nTableHeight < this._oTableBottom){
//定位
(isIE6){
this._style.top = posTop + "px";
this._style.left = this._oTableLeft + "px";
Timeout(Bind(this, this.SetSelect), 0);//iebug
}{
this._style.top = viewTop + "px";
this._style.left = this._oTableLeft - left + "px";
}
;
}
}

```

```
}
}
//隐藏
this._style.top = "-99999px";
isIE6 && this.ReSelect;
},
//设置select集合
SetSelect: function {
this.ReSelect;
var rect = this._nTable.getBoundingClientRect;
//把需要隐藏放到_selects集合
this._selects = Filter(this._oTable.getElementsByTagName("select"), Bind(this, function(o){
var r = o.getBoundingClientRect;
(r.top <= rect.bottom && r.bottom >= rect.top){
o._count ? o._count : (o._count = 1); //防止多个例子冲突
//设置隐藏
var visi = o.style.visibility;
(visi != "hidden"){ o._css = visi; o.style.visibility = "hidden"; }

true;
}
}))
},
//恢复select样式
ReSelect: function {
forEach(this._selects, function(o){ !--o._count && (o.style.visibility = o._css); });
this._selects = ;
}
};
```

2009-8-16 12:41:05

疯狂代码 <http://CrazyCoder.cn/>