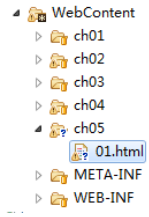
第五节 HTML&CSS -- 关于浮动和清除浮动的解说，以及两个大坑不要踩

1.随便唠叨几句

这一节课我会对浮动元素和怎样清除浮动相关的技术进行一个讲解，同时，我会列举一些我们前端开发中常见的坑，希望大家以后不要在这些地方犯错。在开始今天的课程之前，有一个东西我需要先讲一下，学网页，我认为最最重要的就是学布局，如果一个网页，你不去布局，学再多的div加css也无济于事，你就算学了再多的特效，又能怎么样？那么，何为布局呢？很简单，就是一句话，**把元素放在你想要放的地方**，这个就是布局了。对于一些后台管理系统，其实最最常见的还是table布局，包括现在很流行的前端框架 - easyui，你去看他的源码，肯定也是table布局嘛。为什么会导致这样的情况发生呢，原因很简单，因为后台管理系统主要是给工作人员去做系统维护的，比如一个新闻发布网站，工作人员要发布一条新闻，那么就得登录后台管理系统，进入一个什么新闻管理的菜单，这些操作注重的是简单，易用，不需要有多么华丽的效果。所以，table布局还是有必要意义的。那么，我们在学习的过程中，是不是一定要把table布局弄得非常精通呢？这倒也不必，我们以后开发项目的时候，基本上还是要运用一些前端框架的，我们更多的，应该是去学会如何使用一个框架，而不是自己去造轮子。当然，等你以后成为技术大牛，造一点轮子也无妨。但是现在，没必要。

OK，废话不多说，开始吧，我已经用table画了一个简单的布局，你没必要去深究这些代码的含义，当然能看懂最好，看不懂也无伤大雅。反正以后我们不太可能自己去布局，都是用框架，嗯，就是这么回事。不过呢，如果要定制一些东西，那么还是需要div加css技术出马。

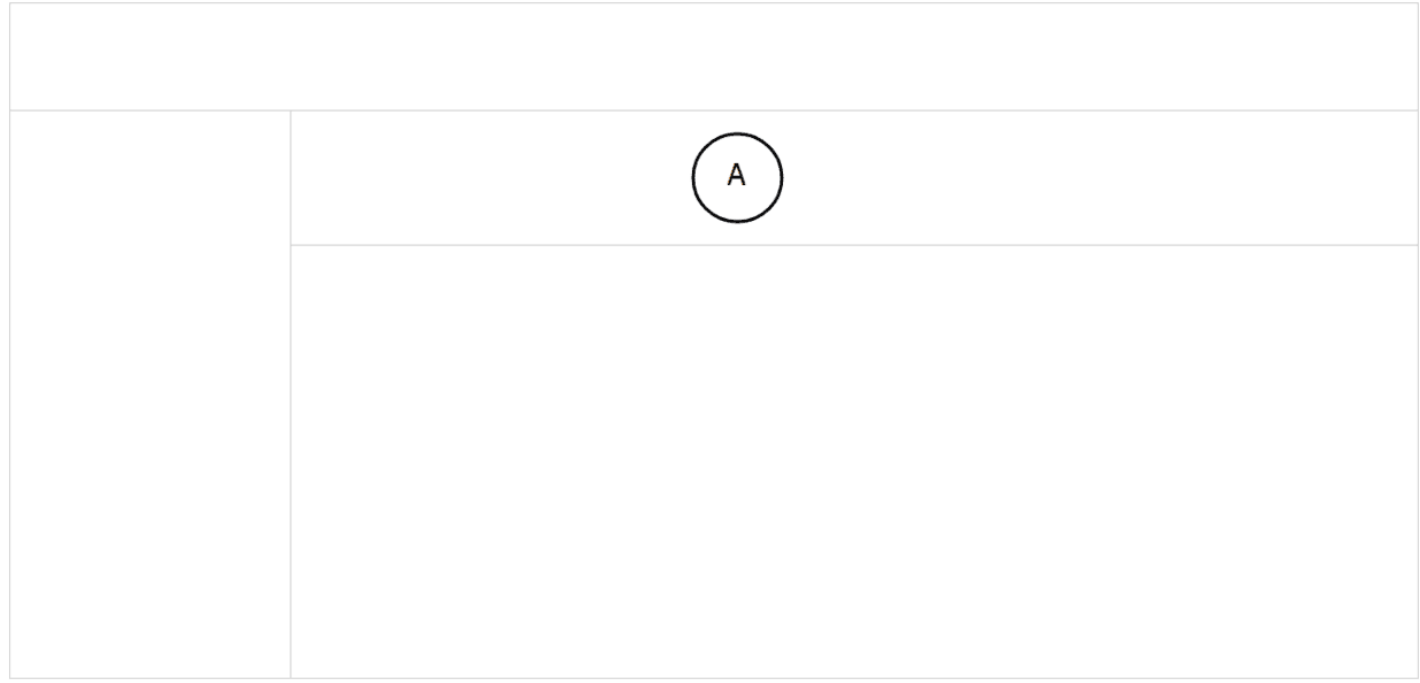


页面：

今天讲的东西，就在这个页面中实践。

2.无法想象，行内元素竟然这么任性，给了宽度和高度也不认帐！

额，是这样的哈，对于初学前端技术的萌新，往往都会在这一个地方吃亏，那就是给行内元素加了宽度和高度，怎么就没用呢？比如，我在A区域添加一个span元素。

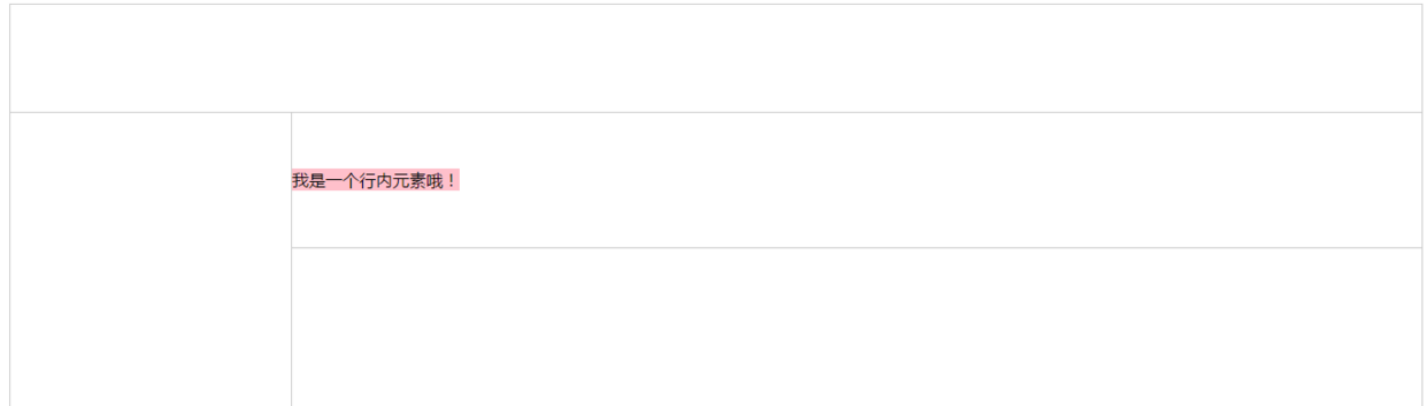



```
<table>

    <tr>
        <td colspan='2' style='height:16%'></td>
    </tr>

    <tr >
        <td rowspan='2' style='width:20%'></td>
        <td colspan='2' style='height:20%'>

            <span id='haha'>我是一个行内元素哦！</span>

        </td>
        <tr>
            <td colspan='2'></td>
        </tr>
    </tr>

</table>
```

然后，给它加一点样式：

```
#haha {
    width:300px;
    height:80px;
    background-color: pink;
}
```

再复习一遍，这个span元素的id是不是叫haha啊，那么我在给他设置样式表的时候，是不是要在前面要加一个#号呀？诶，就是这样的，如果这个span元素绑定了一个class属性，叫做xixi，那么前面加的就是一个点，应该是“.xixi”，不要混淆了。让我们来看一下效果：



我靠，郁闷的事情果然发生了，为什么加宽度和高度没有效果呀？呵呵，我相信很多人在这里吃过亏，不管怎么调就是没效果。原来，span是一个行内元素，而只有块级元素和行内块级元素才能够有自己的宽高，所以，我们给行内元素添加宽度和高度是没有效果的。解决方法有很多，最简单的，就是给这个span元素添加一个“display:block”或者“display:inline-block”的样式，就可以解决这个问题了。

```
#haha {
    width:300px;
    height:80px;
    background-color: pink;
    display:inline-block;
}
```

效果：

接下来一个问题，我们已经给这个行内元素升级为行内块级元素了，如何让里面的字相对于这个span居中定位呢？首先，我们可以给它加一个text-align:center

```
#haha {
    width:300px;
    height:80px;
    background-color: pink;
    display:inline-block;
    text-align:center;
}
```

然后，给它设置一个行高，和这个元素本身的高度相等

```
#haha {
    width:300px;
    height:80px;
    background-color: pink;
    display:inline-block;
    text-align:center;
    line-height:80px;
}
```

这样是不是就好了呀？嗯，这是今天第一个知识点。

3.震惊！P元素和div元素竟然这样，听说还有程序员不知道！

p元素是一个块级元素，所谓块级元素，就是那种会独自占满一行的元素，还有div也是块级元素。先来个例子吧。我现在要在B区域加一个P元素：


```
<span id='haha'>我是一个行内元素哦！</span>
```

我是一个行内元素哦！

哈哈，我是一个段落哦！

然后，我在这个p元素上挂载一个div元素：

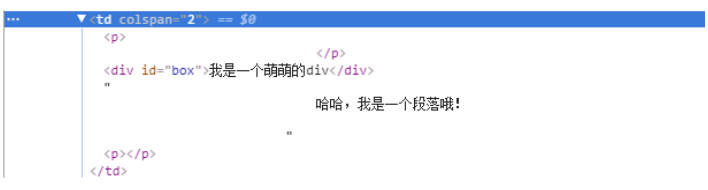
```
<tr>
    <td colspan='2'>
        <p>
            <div id='box'>我是一个萌萌的div</div>
            哈哈，我是一个段落哦！

        </p>


    </td>
</tr>


  #box {
      width:90px;
      height:80px;
      background-color: blue;
      color:#fff;
      text-align:center;
      line-height:80px;
  }
```

我是一个行内元素哦！

我是一个萌

哈哈，我是一个段落哦！

看起来似乎没什么，让我们打开F12看看：



发现问题了吗，各位？本来是一个p元素，当我在它里面放一个div元素的时候，一个p元素竟然被分成了两个，同学们，这样是不是很危险啊？所以，我们一定要警惕这种写法，p元素被设计出来的本意，就是用来存放文本内容的，不能够用来布局，不然的话，可能会产生很多意想不到的BUG。布局，我们一般就用table，或者div、span，加上css就可以了。p元素就是用来存放文字的，明白了吗？

4.惊！多个div元素为了并排显示，连这种事都干得出！

接下来，我们来说一个非常非常重要的知识点，那就是浮动布局。那么，到底什么是浮动呢，还是以案例为主吧。我在B区域画三个div盒子，给他们都绑定同一个class，叫做box。

```
<tr>
    <td colspan='2'>
        <div class='box'>第一个盒子</div>
        <div class='box'>第二个盒子</div>
        <div class='box'>第三个盒子</div>
    </td>
</tr>
```

```
.box {
    width:90px;
    height:80px;
    background-color: purple;
    color:#fff;
    text-align:center;
    line-height:80px;
    margin:2px;
}
```

可以看到，div元素是块级元素，再复习一下啊，块级元素有什么特点啊，是不是会独自占满一行呀？所以，这三个div元素就没法在同一行显示了。那么，有没有什么办法，让多个块级元素在同一行显示呢？当然是有的，我们可以采用浮动布局，也就是说，让这些div元素“飘起来”，注意哦，飘起来以后，有两个方向，一个是向左漂浮，另一个则是向右漂浮。是的，只有这两种状态，没有第三种了。明白了吗，只有两种，要么往左边浮动，要么往右边浮动。

我们来尝试一下，让所有class为box的元素都往左边飘。

```
.box {
    width:90px;
    height:80px;
    background-color: purple;
    color:#fff;
    text-align:center;
    line-height:80px;
    margin:2px;
    float: left;
}
```

哇，是不是飘起来了。没错，这个就叫做浮动布局。我现在问一下大家，你觉得，浮动布局的目的是什么？没错，就是为了让块级元素在同一行上显示，仅此而已。没有其他更玄妙的说法了，不论多么复杂的网页，如果用到了浮动布局，肯定就是这么回事，肯定是为了让某些div元素在一行上面显示。在这个例子中，我们如果让第三个盒子往右浮动，就给它单独加一个行内样式：

```
<tr>
    <td colspan='2'>
        <div class='box'>第一个盒子</div>
        <div class='box'>第二个盒子</div>
        <div class='box' style='float:right'>第三个盒子</div>
    </td>
</tr>
```

现在，我还有个非常非常重要的知识点要交给大家，到底是什么呢？我们直接来个案例好不好？

我把这几个盒子换成span元素：

```
<tr>
    <td colspan='2'>
        <span class='box'>第一个盒子</span>
        <span class='box'>第二个盒子</span>
        <span class='box'>第三个盒子</span>
    </td>
</tr>
```

然后把浮动去掉：

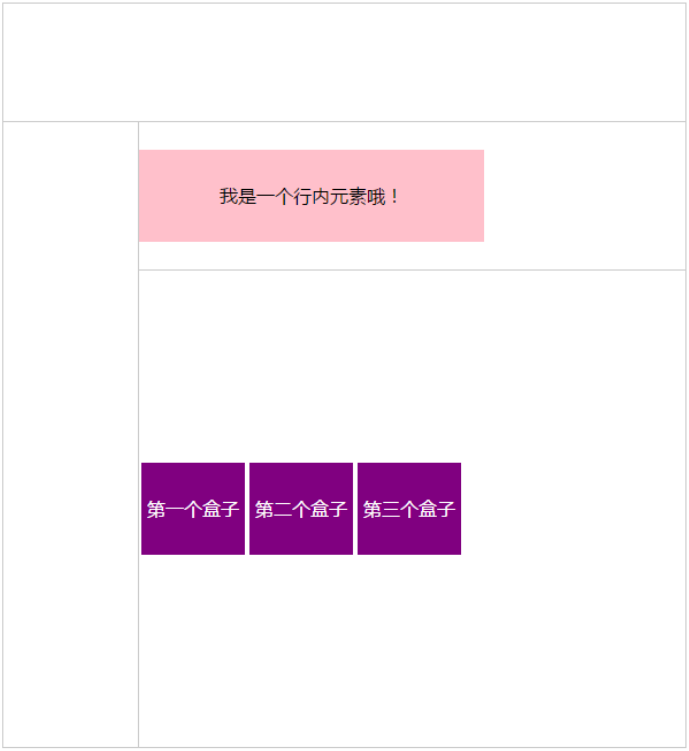
```
.box {
    width:90px;
    height:80px;
    background-color: purple;
    color:#fff;
    text-align:center;
    line-height:80px;
    margin:2px;
}
```

页面就是这样的：

为什么呀，刚才我们是不是已经说了呀，当我们给一个行内元素设置宽度和高度，是不是没有用的呀？如果我们要让它生效，是不是要加一个“display:inline-block”或者“display:block”，就好了呀。那么，在这个例子中，我直接加上一个浮动定位：

```
.box {
    width:90px;
    height:80px;
    background-color: purple;
    color:#fff;
    text-align:center;
    line-height:80px;
    margin:2px;
    float: left;
}
```

刷新页面：

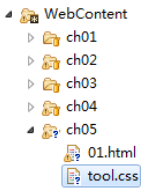


哇，天哪，怎么会这样？哦，原来，一旦我们给元素加了浮动定位，哪怕它是行内元素，也会默认转换成行内块级元素了。这也是一个比较重要的知识点，大家一定要记住。

5.惊！div元素因真爱成功私奔，还让空元素背锅，没WIFI也要看！

最后呢，我们要讲的，就是关于清除浮动的问题了，先来看下问题的由来。刚才，我们弄了三个div元素，都设置了float:left，于是，他们就脱离了原本的轨道，术语叫做脱离标准流。其实就是把自己的位置给腾出来了，来个例子吧。

首先，为了方便起见，我们把float：left，float:right这两个样式也封装一下，作为一个通用的工具样式，先把上一节课的tool.css拷贝过来：



然后，加上浮动的样式：

```
/* 浮动布局 */
.fl {
    float:left;
}
.fr {
    float:right;
}
```

然后，在这个html文件中，引入tool.css:

```
<link rel="stylesheet" type="text/css" href="tool.css" />
```

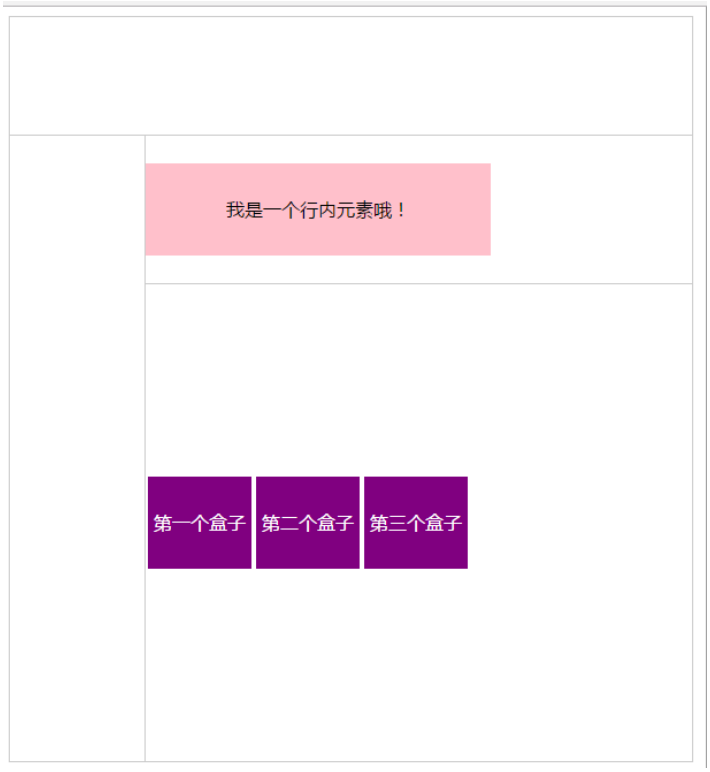
这样的好处就是，我们不必单独写浮动的样式表了，在box中，把浮动样式删掉：

```
    .box {
        width:90px;
        height:80px;
        background-color: purple;
        color:#fff;
        text-align:center;
        line-height:80px;
        margin:2px;
        float: left;
    }
```

删去

然后，在需要加浮动的地方，加上浮动的class：

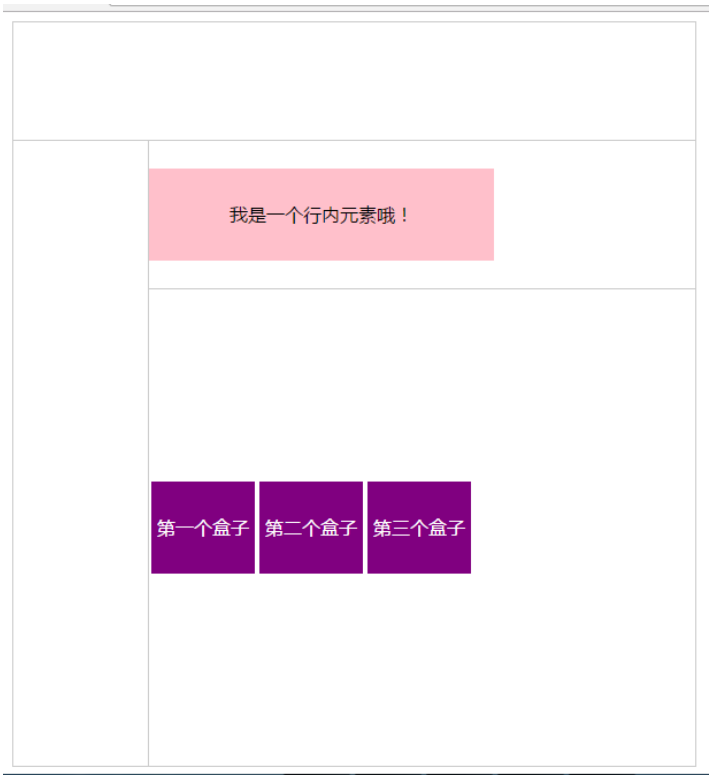
```
<tr>
    <td colspan='2'>
        <span class='box fl'>第一个盒子</span>
        <span class='box fl'>第二个盒子</span>
        <span class='box fl'>第三个盒子</span>
        <div ></div>
    </td>
</tr>
```



诶，这样是不是也可以呢？好的，接下来，我在第三个盒子后面再加一个盒子：

```
<tr>
    <td colspan='2'>
        <span class='box fl'>第一个盒子</span>
        <span class='box fl'>第二个盒子</span>
        <span class='box fl'>第三个盒子</span>
        <div class='box'>第四个盒子</div>
    </td>
</tr>
```

注意哦，第四个盒子我没有加上左浮动，会有怎样的效果呢？看：



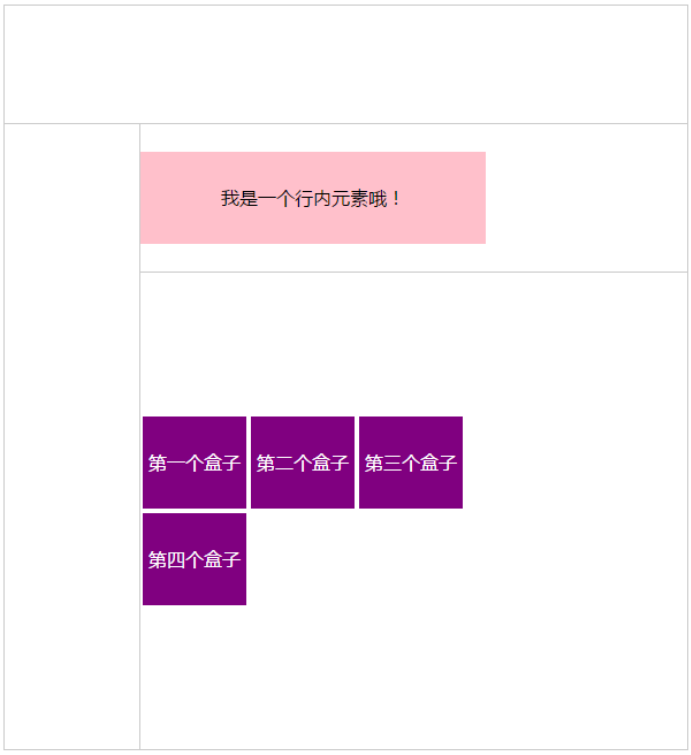
第四个盒子消失了，为什么呢？刚才我们说，当一个元素设置了浮动，那么就会飘起来，脱离标准流，也就是不占位置了。那么，在它后面的元素是不是就要紧跟上来啊？举一个现实生活中的例子，ABC三个人在排队买彩票，突然，B想到还有一件事情要做，就走了，那么C是不是要往前走一步，占据B的位置呀？注意哦，这个队伍就类似于标准流，现在B脱离了标准流，C是不是要跟上来，明白了吗？那么，在我们开发网页的时候，我们肯定不希望这样的事情发生吧，还记得我们当初采用浮动布局的目的是什么吗，是不是要让几个块级元素并排显示呀？现在虽然并排显示了，可是后面的元素自动顶上来，是不是布局就乱了呀？所以，我们需要清楚浮动。到底什么叫做清楚浮动呢，其实很简单，一句话的事：

清除浮动就是给飞出去的元素填坑，好让后面的元素不顶过来。这就是清除浮动，没有什么更加玄妙的东西了，就这么简单。

如何清除浮动呢？有一个办法就是在浮动定位的最后一个元素后面，加上一个空元素，比如div元素，里面啥也不写，然后加上一个叫做clear:both的样式，那么，浮动就被清除了。

```
<tr>
    <td colspan='2'>
        <span class='box fl'>第一个盒子</span>
        <span class='box fl'>第二个盒子</span>
        <span class='box fl'>第三个盒子</span>

        <div style='clear:both'></div>

        <div class='box'>第四个盒子</div>
    </td>
</tr>
```

刷新浏览器：

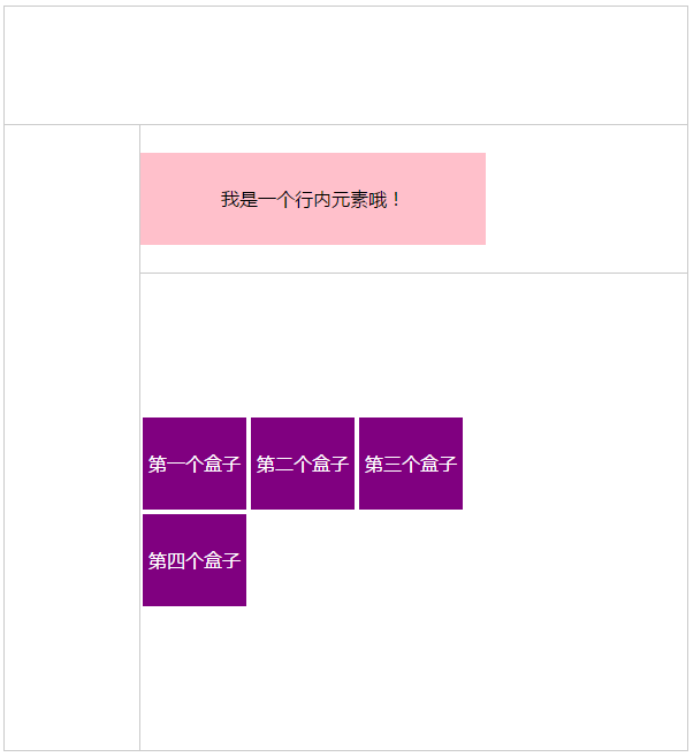


诶，是不是就好了呀？可是，你是否觉得，每次都这样去清除浮动的话，是不是会有很多个这样的空元素呀？这些空元素，仅仅是为了清除浮动而已，那么网页代码就显得有些乱。其实还有一种更好的方式，就是运用伪类，这是一个比较成熟的方式，也算是css中一种比较先进的技术，你就算看不懂也无所谓，只要会用就行。

```
/* 清除浮动 */
.clearfix:before,
.clearfix:after {
    content: " ";
    display: table;
}

.clearfix:after {
    clear: both;
}
```

我把这个样式也添加到了tool.css里面，然后，给需要清除浮动的那个元素添加上clearfix的class就OK了。比如，在这个例子中，我们就需要给三个盒子外面套一层div，加上clearfix的class，就可以了。以后我们都采用这种方式。

效果：



全文完。

源码下载地址：http://www.xiaotublog.com/getResourcePage.html?id=22
欢迎关注我的博客，长期更新各种JavaWeb相关的资料。