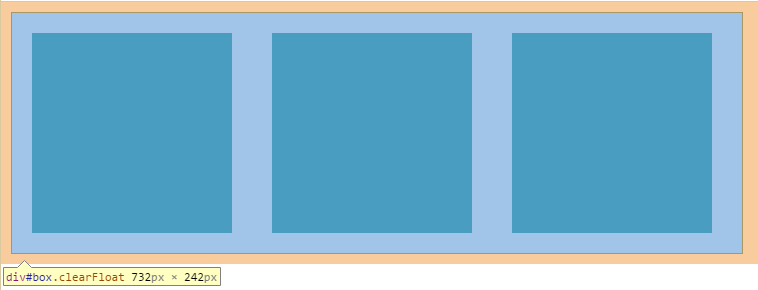
## 浮动从哪儿来

在我们进行页面布局的时候，常常会用到float，让元素浮动起来，这会让我们很方便地让块级元素并排显示。

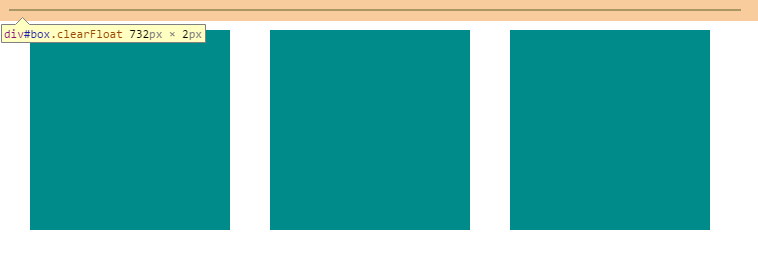
## 为什么要清除浮动

虽然float很好用，但也会给我们带来问题。

在父级元素没有设置高度的情况下，原本高度会让其内容撑开，但在我们使用float的时候，会导致元素脱离文档流，使父级元素高度塌陷——content高度变为0，会影响父元素兄弟元素的排版。



我们想要实现的效果



实际却是这样的

从上面我们可以看到，在使用浮动后，使父级元素的高度塌陷了！

## 怎样清除浮动

为了解决浮动使父元素高度塌陷的问题，我们就要想办法清除浮动，使父元素呈现正常高度。

想要清除浮动，我们可以使用这几种方法。

1. 给父元素设置高度

最简单粗暴的解决方案 首当其冲应该是 直接给父元素加 固定高度

1. 添加额外标签

在浮动元素的末尾添加一个宽高为0,的额外元素，并给它设置clear:both

**css代码:**

1. #d4{
2. width: 0;
3. height: 0;
4. clear: both;
5. }

**Html代码：**

1. **<div** id="box"**>**
2. **<div** id="d1"**></div>**
3. **<div** id="d2"**></div>**
4. **<div** id="d3"**></div>**
5. **<div** id="d4"**></div>**  //额外的标签
6. **</div>**
7. 给父级设置overflow: hidden

**css代码:**

1. #box{
2. margin: 10px;
3. width: 740px;
4. border: 1px solid #000000;
5. overflow: hidden;  //设置overflow: hidden（也可用auto）
6. }
7. 让父级也浮动

**css代码:**

1. #box{
2. margin: 10px;
3. width: 740px;
4. border: 1px solid #000000;
5. float: left;  //让父级也浮动
6. }
7. 给父级设置display：table

**css代码:**

1. #box{
2. margin: 10px;
3. width: 740px;
4. border: 1px solid #000000;
5. display: table;  //给父级设置display: table
6. }
7. 使用:after

通过伪元素:after给父元素追加如下clearFloat类

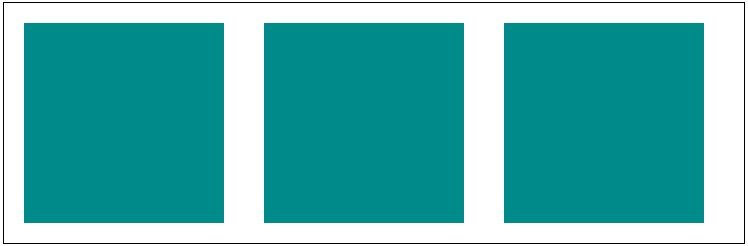
**css代码:**

1. .clearFloat:after{
2. content: "";
3. height: 0;
4. clear: both;
5. display: block;
6. visibility: hidden;
7. }

**HTML代码：**

1. **<div** id="box" class="clearFloat"**>**
2. **<div** id="d1"**></div>**
3. **<div** id="d2"**></div>**
4. **<div** id="d3"**></div>**
5. **</div>**

通过上面的方法清除浮动后我们最终都能实现如下效果：



清除浮动后的效果

上面方法三、四、五都用到了BFC的原理，我们就不得不提一下什么是BFC了，W3C CSS2.1规范中对BFC有如下概念：

## BFC概念

BFC决定了元素如何对其内容进行定位，以及与其他元素的关系和相互作用。当涉及到可视化布局的时候，Block Formatting Context提供了一个环境，HTML元素在这个环境中按照一定规则进行布局。一个环境中的元素不会影响到其它环境中的布局。

相信很多小伙伴听到这里都一脸懵逼，我们可以把BFC理解成是一个独立的容器，并且这个容器里的box的布局，与这个容器外的毫不相干。

那么，怎么才能触发BFC呢？

## 怎样触发BFC

1)float的值不为none。

2)overflow的值不为visible。

3)display的值为table-cell, table-caption, inline-block中的任何一个。

4)position的值不为relative和static。

很多小伙伴听到这里还是一脸懵逼，大家不要着急也不要懊恼，毕竟BFC的定义对于我们这种前端初学者来说理解起来还是很困难的，后面我们实际上手做的东西多了自然就理解了，我们不妨从它的作用来来让大家对BFC有一个初步的理解。

## BFC作用

一、清除浮动：

第一个作用就是我们刚刚讲的清除浮动了，子元素因为使用了float触发了BFC，使之不再处于普通文档流之中，这时候我们就可以让父元素也触发BFC，这样就能把子元素包起来了。

在我们让父元素触发BFC时，要考虑到给它新加的属性会不会影响到别的元素的排版，不然就相当于“挖东墙补西墙”了，所以要根据实际情况选择选择恰当的方式。

二、解决上下元素边距重叠：

根据 CSS 2.1 [8.3.1 Collapsing margins](https://www.cnblogs.com/nujufoul/p/7092520.html#collapsing-margins) 第一条，两个相邻的普通流中的块框在垂直位置的空白边会发生折叠现象。也就是处于同一个BFC中的两个垂直窗口的margin会重叠。

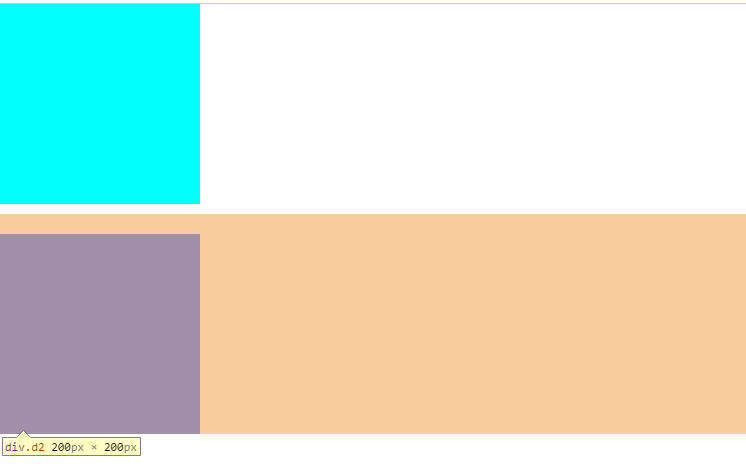
**CSS代码：**

1. .box1{
2. **height**: 200px;
3. **width**: 200px;
4. **background-color**: cyan;
5. **margin-bottom**: 30px;  //盒子一下边距为30个像素
6. }
7. .box2{
8. **height**: 200px;
9. **width**: 200px;
10. **background-color**: tomato;
11. **margin-top**: 20px;  //盒子二上边距为20个像素
12. }

**HTML代码：**

1. **<div** class="d1"**></div>**
2. **<div** class="d2"**></div>**

**效果如下：**



我们可以看到上面两个盒子边距重叠了，两个盒子之间的距离为其中较大的边距。

根据 CSS 2.1 [8.3.1 Collapsing margins](https://www.cnblogs.com/nujufoul/p/7092520.html#collapsing-margins) 第三条，生成 block formatting context 的元素不会和在流中的子元素发生空白边折叠。所以解决这种问题的办法是要为两个容器添加具有BFC的包裹容器。

**CSS代码：**

1. .box1{
2. **height**: 200px;
3. **width**: 200px;
4. **background-color**: cyan;
5. **margin-bottom**: 30px;  //盒子一下边距为30个像素
6. }
7. .box2{
8. **height**: 200px;
9. **width**: 200px;
10. **background-color**: tomato;
11. **margin-top**: 20px;  //盒子二下边距为20个像素
12. }
13. .BFC{
14. **overflow**: hidden;
15. }

**HTML代码：**

1. **<div** class="BFC"**>**
2. **<div** class="d1"**></div>**
3. **</div>**
4. **<div** class="BFC"**>**
5. **<div** class="d2"**></div>**
6. **</div>**

**效果如下：**



可以看到上下两个盒子距离变大，为两个margin相加变成了50个像素。

通过上面例子，通俗来讲，我们可以这样理解BFC的作用：

一、帮助爸爸找到离家出走的儿子；

二、亲兄弟，明算账。