第4讲 DIV的position定位

2018年3月29日 C16F38班

## 0、导语

关于DIV在浏览器中的布局位置，我们是在CSS中通过定位属性position来进行定位的，常见的属性有如下几个：

1. Static:默认值。没有定位。
2. Fixed:生成绝对定位的元素，相对于浏览器窗口进行定位。
3. Absolute:生成绝对定位的元素，
4. Relative:生成相对定位的元素，相对于其正常位置进行定位。
5. Inherit:父元素继承 position 属性的值。

和position属性密切相关的还有z-index属性，也经常在布局中与position属性配合使用。

## 1、position值的含义

Static：静态定位，是position的默认值，DIV正常生成，也就是没有定位时的正常显示。DIV出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。

fixed：固定定位，DIV脱离文本流，从普通文本中删除，并相对于浏览器视窗定位，不随文档滚动而移动。通过 "left", "top", "right" 以及 "bottom" 属性定位其位置。设置以后，DIV原来的位置将释放出来。

absolute：绝对定位，元素从文档流脱离，并相对于父级DIV定位。相对于static定位以外的第一个父元素进行定位。元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。使用absolute以后，DIV原来所在的位置释放。

relative：相对定位，最常见的使用方式有如下两种：

使用方法一：DIV身的原位置偏移某个距离，但是原本的空间依旧保留，但显示为空白。

使用方法二：把一个DIV设置为position: relative; 可以使它的子级DIV相对该元素绝对定位。

在网页设计中，position属性的使用是非常重要的。有时如果不能认识清楚这个属性，将会给我们带来很多意想不到的困难。

## 2、static定位

static定位是DIV的默认值，即没有定位，元素出现在正常的流中，因此，这种定位就不会受到top，bottom,left,right的影响。

.wrap{width: 300px;height: 300px; background: red;}

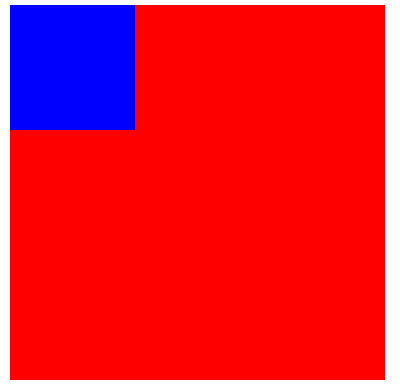
.content{position: static; top:100px; width: 100px;height: 100px; background: blue;}

<div class="wrap">

<div class="content"></div>

</div>

效果图如下：



可以发现，虽然设置了static以及top，但是DIV仍然按默认位置出现显示，不会移到TOP所对应的坐标。

## 3、fixed定位

fixed定位是指div的位置相对于浏览器窗口是固定位置，即使窗口是滚动的它也不会滚动，且fixed定位使div的位置与文档流无关，因此不占据空间，且它会和其他元素发生重叠。

body{height:1500px; background: green; font-size: 30px; color:white;}

.content{ position: fixed; right:0;bottom: 0; width: 300px;height: 300px; background: blue;}

<div class="content">我是使用fix来定位的！相对于浏览器窗口，一直不动。</div>

效果图如下：

即右下角的div永远不会动，网页中很多浮动广告经常采用这个作法。

## 4、relative定位

DIV的定位是相对它自己的原本正常位置的定位进行移动而形成新的定位。要弄明白相对定位，关键在于理解其DIV自身的坐标

我们先看一个例子：

.b2{position:relative;bottom:-20px;}

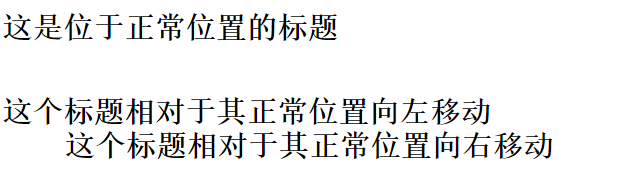
.b3{position:relative;left:50px;}

<h2>这是位于正常位置的标题</h2>

<h2 class="b2">这个标题相对于其正常位置向下移动</h2>

<h2 class="b3">这个标题相对于其正常位置向右移动</h2>

效果图如下：



看到上例的效果后，我们可以再结合一下HTML的坐标轴来理解：

0，0

width

height

这是位于正常位置的标题

这个标题相对于其正常位置向下移动

这个标题相对于其正常位置向下移动

弄清楚了relative是如何移动的，下面我们看一看移动之后是否会产生其他的影响。

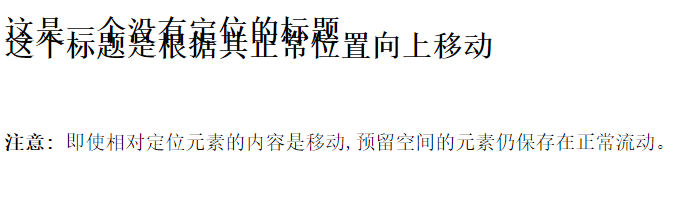
.pos\_top{position:relative;top:-35px;}

<h2>这是一个没有定位的标题</h2>

<h2 class="pos\_top">这个标题是根据其正常位置向上移动</h2>

<p><b>注意:</b> 即使相对定位元素的内容是移动,其原本的位置依然存在</p>

效果图如下：



根据坐标关系，pos\_top向上偏移了35px；我们发现移动后和上面的元素发生了重叠；但即使相对元素的内容移动了，它原本所在的位置空间并不释放，也就是说相对移动之后，不会对下面的其他元素造成影响。

## 5、absolute定位

绝对定位的元素相对于最近的已定位父元素（非static），如果元素没有已定位的父元素，那么它将一直向上找更高级的已定位元素，最高可至相对于<body>。

下面举几个例子：假设有以下层次关系的元素：

DIV(parent)

DIV(son)

span

### 例1：

<style>

body{background:green;}

.parent{ width: 500px;height: 500px;background: #ccc;}

.son{ width: 300px;height: 300px;background: #aaa;}

span{position: absolute; right: 30px; background: #888;}

</style>

<body>

<div class="parent">

<div class="son">

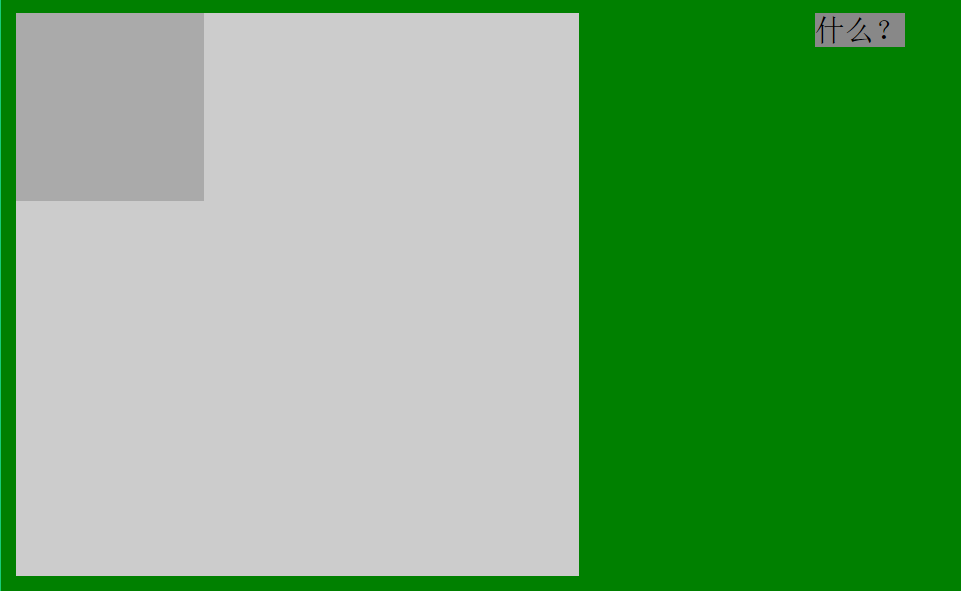
<span>什么？</span>

</div>

</div>

</body>

　效果如下：



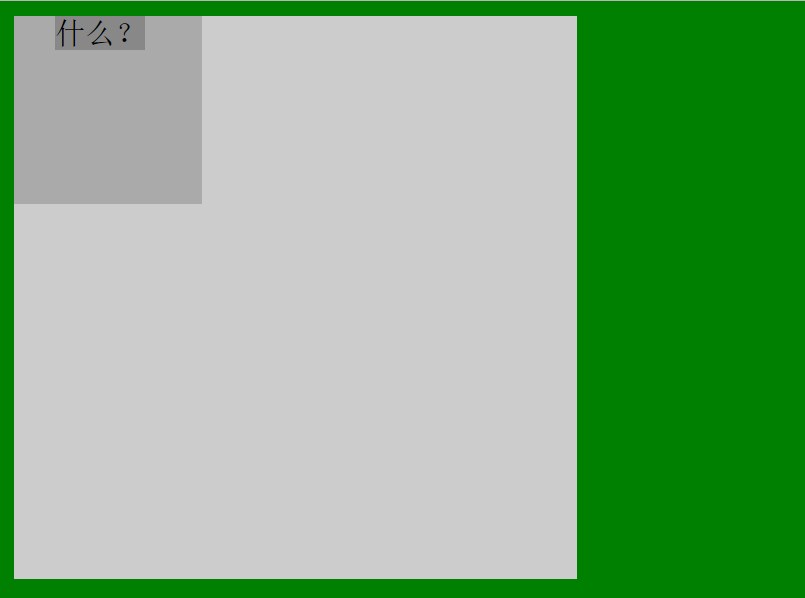
例子中只有span设置了position:absolute；而在其父级元素中都没有，因此它的位置是相对于<body>，位置是右边缘距离body的右边缘30px。

### 例2（基于例1的基础上修改）:

.son{position:relative;width:100px;

height:100px;background: #aaa; }

相较于上一个例子，修改了class为son的元素的css，设置为position：relative；效果图如下：



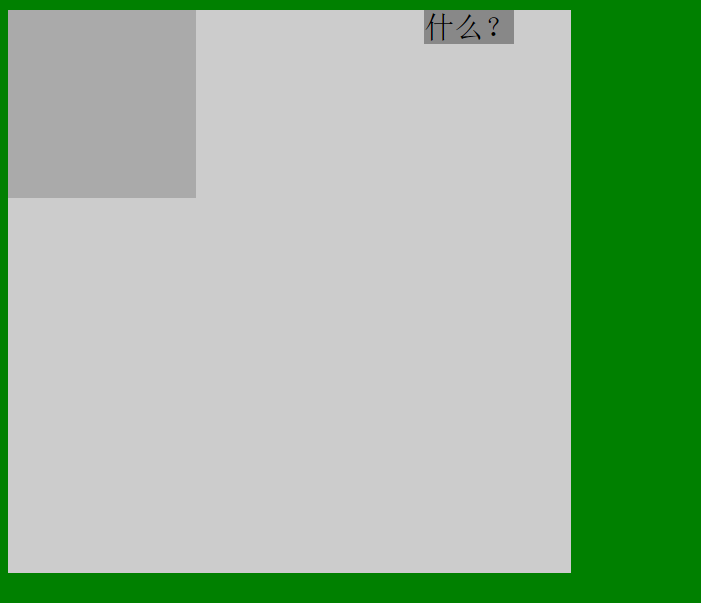
因为span是son的子元素，但son的定位有了设置以后，span的位置就相对于设置了position属性的父元素son的。

### 例3（基于例1的基础上修改）：

.parent{position:absolute;width:300px;

height: 300px;background: #ccc;}

修改了第一个例子中class为parent的元素的css,设置了position:absolute；效果如下：



可以发现，现在span的定位是相对于具有position:absolute的属性的class为parent的父元素。

### 例4（基于例1的基础上修改）：

.parent{position:fixed;width:300px;

height: 300px;background: #ccc;}

相对于例1，添加了fixed的position属性，可以看到发现结果和例3是一模一样的，因为都是parent的position作了定位。

### 例5（基于例1的基础上修改）：

.parent{position:static;width:300px;height:300px;

background: #ccc;}

相对于例1，添加了static的position属性（即html的默认属性），结果和例1是一样的,因为static设置与不设置，没有本质区别。

综上所述，当某个absolute定位元素的父元素具有position:relative/absolute/fixed时，定位元素就会依据其父元素的坐标定位，而父元素没有设置position属性或者设置了默认属性（static），那么该元素属就会依据body坐标来定位。

## 6、重叠的元素--z-index属性

首先注意：z-index属性只能在position属性值为relative或absolute或fixed的元素上有效。

基本原理是：z-index的值可以控制元素在眼睛与屏幕之间（z轴）的堆叠顺序,值大的元素发生重叠时会显示在值小的元素上面。见下图示意：



屏幕

z-index

0

1

2

下面我们通过几个例子继续来理解这个属性。假设有三个DIV，他们之间的层次关系如下图：

Div(parent)

Div(son1)

Div(son2)

### 例1：

<style type="text/css">

.parent

{

position:relative;width:400px;height:300px;

background-color:#09F;

}

.son1{

position:absolute;left:30px;width:100px;

height:100px;

background-color:#3C3;

}

.son2{

position:absolute;left:80px;width:100px;height:100px;

background-color:#F66;}

</style>

</head>

<body>

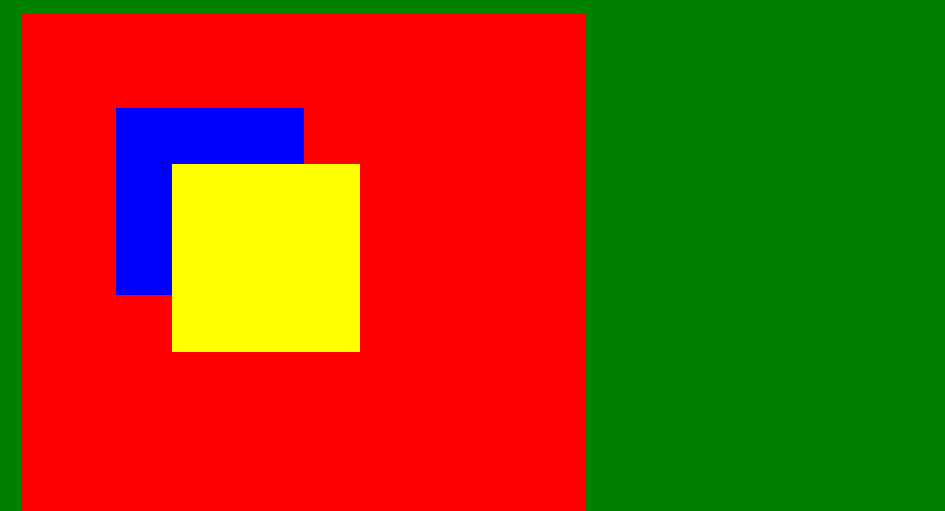
<DIV class="parent">

<div class="son1"> </div>

<div class="son2"> </div>

</DIV>

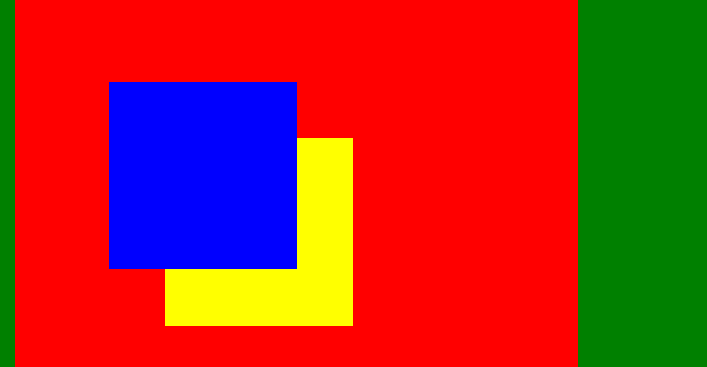
即son1和son2是parent的两个子元素，效果图如下：



这是没有使用z-index，我们发现son2在son1之上，这是因为son2在html中排在了son1之后，所以后来者将前者覆盖，如果我们颠倒一下两者的顺序，就会发现蓝色(son1)在上了。

### 例2：

在son1中加入z-index：1；可以发现效果如下：

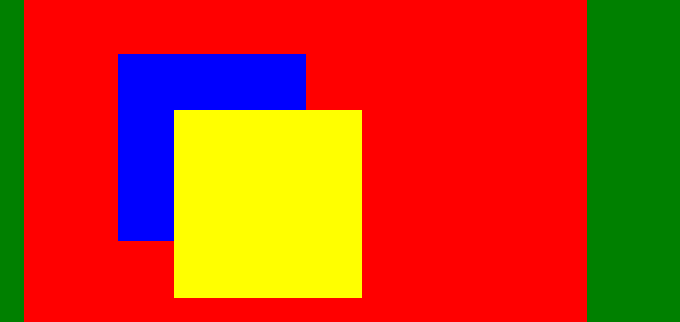


也就是说son2的index值是小于1的。

如果我们给son2也加上z-index:1;呢？结果得到黄色(son2)就在上面了。（因为一旦z-index值相等，情况就和都不设置index值一样了）

### 例3：

在son2中加入z-index:5;可以发现效果如下：



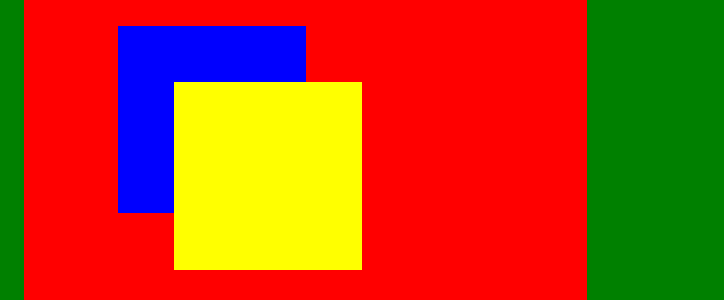
即黄色(son2)又在上面了，这个很简单，不作过多讨论。

### 例4：

在父元素添加z-index:10;

在son1和son2添加z-index:5; 这样理论上父元素就会在上面（黄色覆盖蓝色和黄色）；

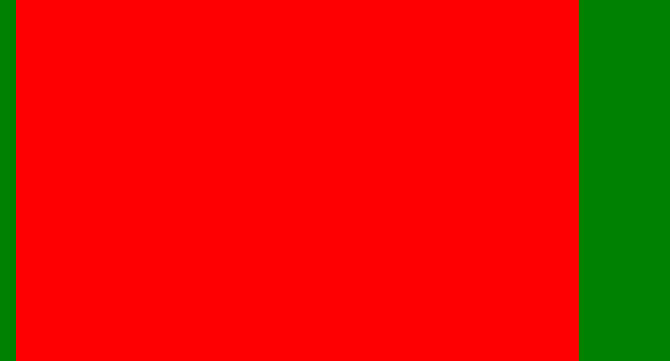
但结果如下：



结果没有变!这就说明父元素和子元素不能做z-index的比较。但这不是绝对的，当设置不同时，显示效果还是会有不同的变化。

### 例5：

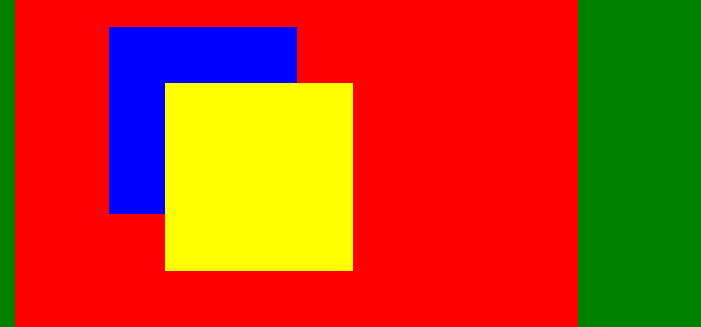
把两个子元素的z-index值同时设置为-5；父元素不设置z-index属性。结果如下：



父元素成功遮挡了子元素。说明在父元素和子元素之间，如果把子元素的z-index值设为负数，就会实现父子元素之间Z-index比较的效果。

### 例6：

我们在例5的基础上再给父元素添加一个z-index:10，讲道理~应该也可以得到和例5相同的结果吧！！

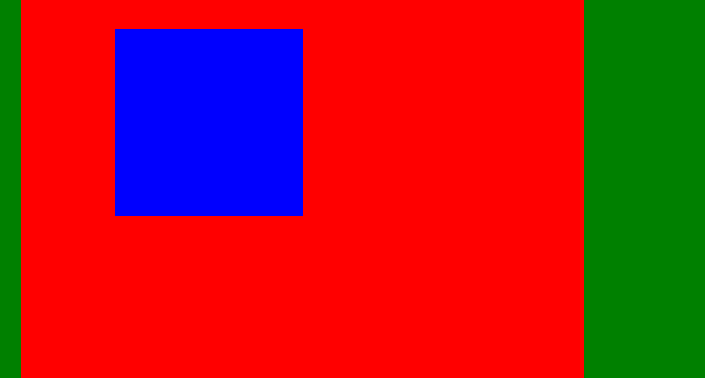


然而.... 看来我们不能设置父元素的z-index值，否则就不能出现我们想要的效果。（可以这样理解：老子（父元素）不容置疑！）

下面再看一个有趣的例子!

### 例7：

我们根据例6的经验不设置父元素的值，现在设置son1(蓝色)的z-index为5，son2（黄色）的z-index为-5，看下面的结果：



即son1在最上面，父元素在中间，son2在最下面。

### 例8：

假设有元素，层次关系如下：

Div(parent1)

Div(son1)

Div(son2)

Div(parent2)

代码如下：

<style type="text/css">

.parent1

{

width:300px;

height:300px;

background-color:#F30;

position:absolute;

left:50px;

z-index:15;

}

.parent2

{

width:300px;

height:300px;

background-color:#F6F;

position:absolute;

left:100px;

z-index:10;

}

.son1

{

position:absolute;

left:100px;

top:50px;

background-color:#06F;

width:100px;

height:100px;

}

.son2

{

position:absolute;

left:100px;

top:50px;

background-color:#FF0;

width:100px;

height:100px;

}

</style>

</head>

<body>

<div class="parent1">

<div class="son1"></div>

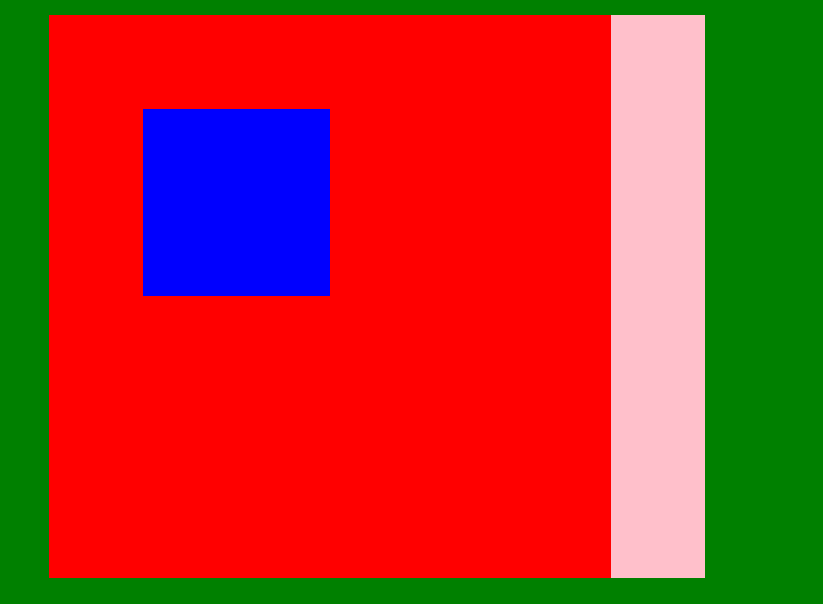
</div>

<div class="parent2">

<div class="son2"></div>

</div>

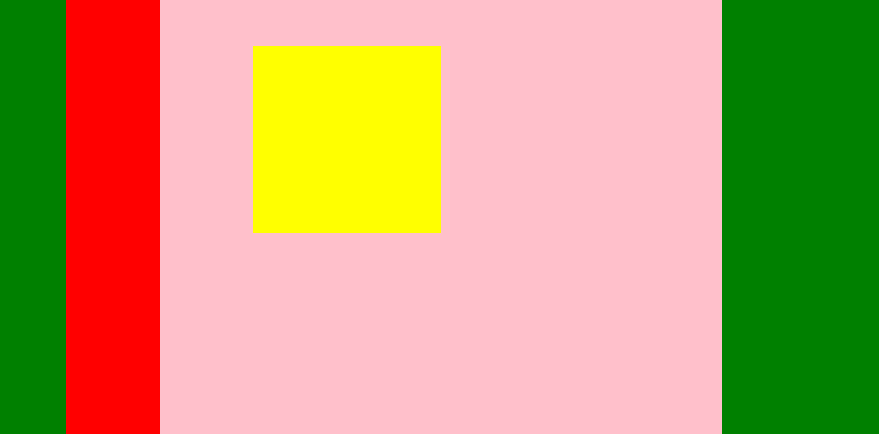
效果如下：



虽然parent1和parent2分别是son1和son2的父元素，按照我们之前的理解，父元素是不可添加z-index值的，否则会导致错误。但是这里parent1和parent2相对于body又是子元素，他俩是同级的，所以就可以进行比较了。且此时parent1的子元素son1（蓝色）在上。

### 例9：

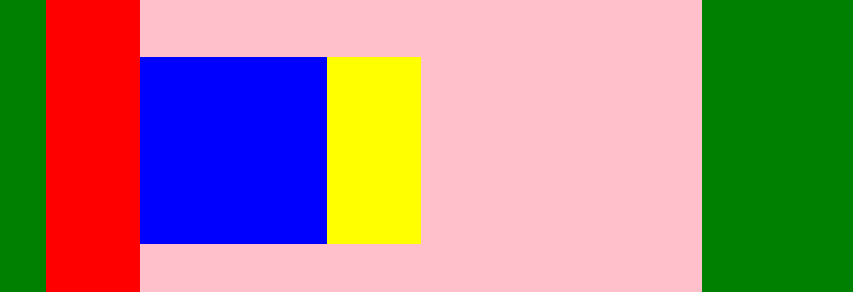
如果我们在例8的基础上，把parent2的z-index值设为20，就会发现如下效果：



因为parent2在上了，因此他的子元素son2也会同时在上。这也就是所谓的“拼爹”了！

### 例10：

同样在例8的基础上，我们不设置parent1和parent2和son2的index值，而只设置son1的z-index值为10，效果如下:

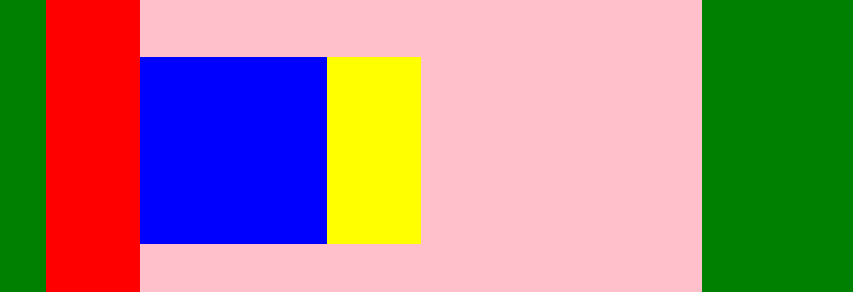


即原本在下面的蓝色son1被提上来了，却没有把父元素（parent1）提上来。唉，翅膀硬了就飞，不孝顺啊！

### 例11：

在例10的基础上，把son2的index值设置的比son1的大，如20，那么son2就会覆盖son1了，并且都在两个父元素只上了。

效果如下图：



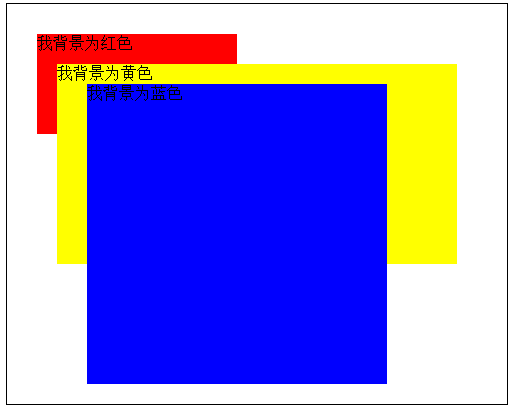
### 例12：

当然，如果我们把两个son的z-index都设置位负数如-5，那么两者就都会被父元素所覆盖：



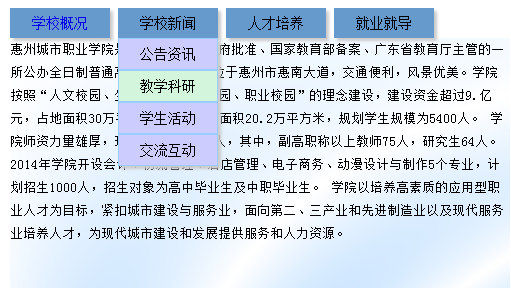
## 课堂练习（一）

新建4个DIV盒子，一个大的DIV盒子，CSS命名为“.div-relative”，三个小DIV盒子放于第一个大DIV对象盒子内，DIV命名分别为“.div-a”、“.div-b”、“.div-c”。请利用所学知识，实现下图所示的重叠效果。



## 课堂练习（二）

利用DIV+css知识，在上一讲《下拉菜单》的基础上，实现以下网页效果（菜单下拉后，显示在网页内容上面）



## 课堂练习（三）

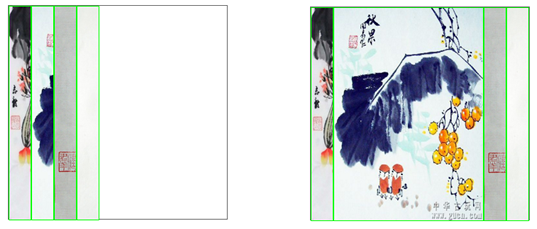
利用DIV+css知识，实现以下网页特效（鼠标移动到相应的图片上时，该图片放大显示）（注：图片放大后300px\*300px）

图1源地址：http://www.sunkinglsx.com/w\_pic/h1.jpg

图2源地址：http://www.sunkinglsx.com/w\_pic/h2.jpg

图3源地址：http://www.sunkinglsx.com/w\_pic/h3.jpg

图4源地址：http://www.sunkinglsx.com/w\_pic/h4.jpg



## 课后作业

使用提供的素材以及所学的DIV+CSS知识，实现下图的网页效果。

