一、布局机制

CSS提供了**3 种机制**来设置盒子的摆放位置，分别是普通流（标准流）、浮动和定位，其中：

普通流（标准流）：

块级元素会独占一行，从上向下顺序排列；

常用元素：div、hr、p、h1~h6、ul、ol、dl、form、table

行内元素会按照顺序，从左到右顺序排列，碰到父元素边缘则自动换行；

常用元素：span、a、em等

浮动

让盒子从普通流中浮起来,主要作用让多个块级盒子一行显示

定位

将盒子定在浏览器的某一个位置——CSS 离不开定位，特别是后面的 js 特效

二、浮动（float）

1、为什么需要浮动?

虽然我们前面学过行内块（inline-block） 但是他却有自己的缺陷：

1.它可以实现多个元素一行显示，但是中间会有空白缝隙

2.盒子垂直对齐问题（一行只能放一个盒子）

因为一些网页布局要求，标准流不能满足我们的需要了，因此我们需要浮动来完成网页布局

2、概念

元素的浮动是指设置了浮动属性的元素会脱离标准普通流的控制

3、作用

让多个盒子(div)水平排列成一行，使得浮动成为布局的重要手段

浮动最早是用来控制图片，实现文字环绕图片的效果

4、语法

```
选择器 {
        float: 属性值;
}
```

属性值	描述
none	元素不浮动（默认值）
left	元素向左浮动
right	元素向右浮动

float属性会让盒子漂浮在标准流的上面，所以第二个标准流的盒子跑到浮动盒子的底下了

注意：浮动的元素互相贴靠一起的，但是如果父级宽度装不下这些浮动的盒子，多出的盒子会另起一行对齐

案例1

```
<head>
    <style>
        .father{
            border: 2px solid blueviolet;
            width: 100%;

        }
        /* 组合选择器 */
        .one,.two,.three,.four{
            float: left;
            width: 200px;
            height: 100px;
        }
        .one{
            background-color: aquamarine;
        }
        .two{
            background-color: blueviolet;
        }
        .three{
            background-color: chartreuse;
        }
        .four{
            background-color: chocolate;
        }
        .five{
            width: 200px;
            height: 200px;
            background-color: cadetblue;
        }
    </style>
</head>
<body>
    <div class="father">
        <div class="one"></div>
        <div class="two"></div>
        <div class="three"></div>
        <div class="four"></div>
    </div>
```

```
    <div class="five"></div>
</body>
```

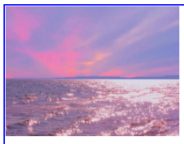
演示

案例2

```
<head>
    <style>
        div{
            border: 2px solid blue;
        }
        img{
            width: 300px;
            height: 300px;
            float: right;
        }
    </style>
</head>
<body>
    <div>
        <img src="../图片/01.jpg" alt="">
        <p>
            如今走过这世间<br>
            万般流连<br>
            翻过岁月不同侧脸<br>
            措不及防闯入你的笑颜<br>
            我曾难自拔于世界之大<br>
            也沉溺于其中梦话<br>
            不得真假  不做挣扎  不惧笑话<br>
            我曾将青春翻涌成她<br>
            也曾指尖弹出盛夏<br>
            心之所动  且就随缘去吧
            </p>
    </div>
</body>
```

演示

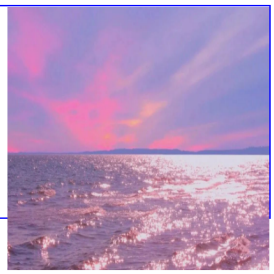
未浮动之前



如今走过这世间
万般流连
翻过岁月不同侧脸
措不及防闯入你的笑颜
我曾难自拔于世界之大
也沉溺于其中梦话
不得真假 不做挣扎 不惧笑话
我曾将青春翻涌成她
也曾指尖弹出盛夏
心之所动 且就随缘去吧

右浮动之后

如今走过这世间
万般流连
翻过岁月不同侧脸
措不及防闯入你的笑颜
我曾难自拔于世界之大
也沉溺于其中梦话
不得真假 不做挣扎 不惧笑话
我曾将青春翻涌成她
也曾指尖弹出盛夏
心之所动 且就随缘去吧



当浮动元素的大小大于父元素时，会发生溢出现象

三、清除浮动

1、为什么要清除浮动?

因为父级盒子很多情况下，不方便给高度，但是子盒子浮动就不占有位置，最后父级盒子高度为0（高度塌陷），就影响了下面的标准流盒子

总结：由于浮动元素不再占用原文档流的位置，所以它会对后面的元素排版产生影响。准确地说，并不是清除浮动，而是清除浮动后造成的影响

2、清除浮动的方法

在CSS中，clear属性用于清除浮动。

（1）方法一

```
语法:
选择器{
    clear:属性值;
}
注意：使用 clear 属性的最常见用法是在元素上使用了 float 属性之后
```

属性值	描述
left	不允许左侧有浮动元素（清除左侧浮动的影响）
right	不允许右侧有浮动元素（清除右侧浮动的影响）
both	同时清除左右两侧浮动的影响（最常用！！）

（2）方法二

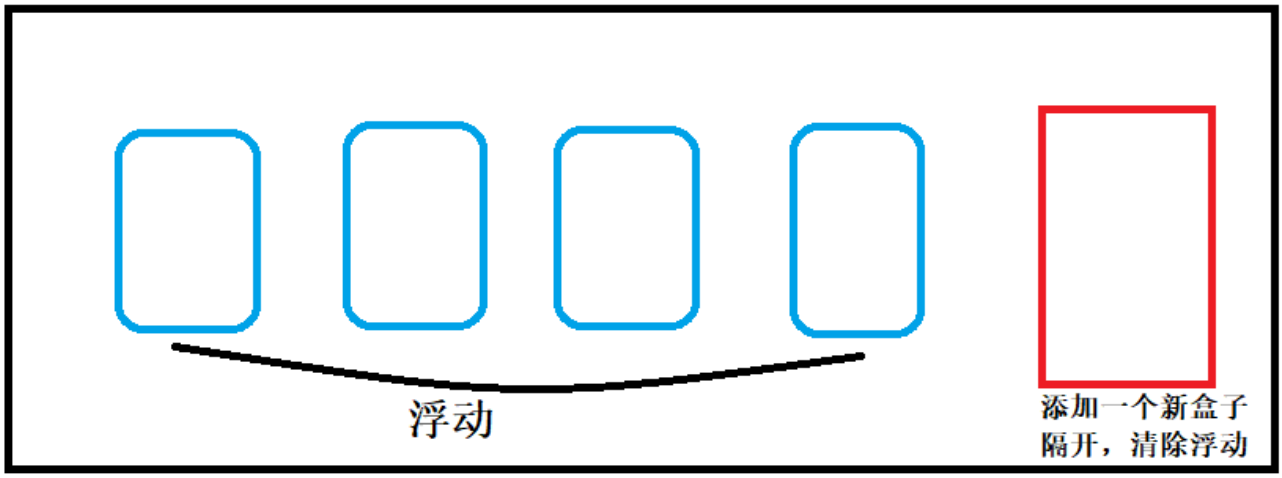
额外标签法（隔墙法）

是W3C推荐的做法是通过在**浮动元素末尾添加一个空的标签**例如

，或者其他标签br等亦可。

优点： 通俗易懂，书写方便

缺点： 添加许多无意义的标签，结构化较差。



（3）方法三（最常用）

父级添加overflow属性方法

给父级添加： overflow为 hidden| auto| scroll 都可以实现。

优点： 代码简洁

缺点： 内容增多时候容易造成不会自动换行导致内容被隐藏掉，无法显示需要溢出的元素。

（4）方法四（了解）

使用after伪元素清除浮动

:after 方式为空元素额外标签法的升级版，好处是不用单独加标签了，**添加在父元素中**，与overflow相同

优点： 符合闭合浮动思想，结构语义化正确（是通过样式实现，没有写在body里面，不显示在页面中）

缺点： 由于IE6-7不支持:after，需要使用 zoom:1触发 hasLayout

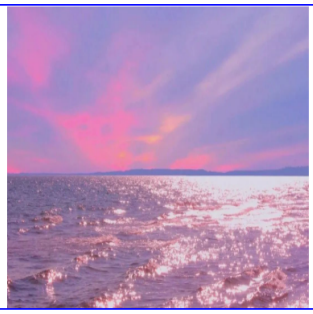
案例

```
<head>
    <style>
         div{
            border: 2px solid blue;
            /* 方式三  给父元素添加overflow */
            /* overflow:hidden; */
        }
```

```
        /* 方法四：给父元素添加伪元素 */
        .one::after{
            /* 固定写法        设置内容为空 */
            content: " ";
            /* 将其转换成块级元素 */
            display: block;
            /* 清楚浮动 */
            clear: both;
        }
        img{
            width: 300px;
            height: 300px;
            float: right;
            padding-right: 20px;
        }
        /* 方式一 */
        /* p{
            clear: both;
        } */

        /* 方式二   浮动元素末尾添加一个空标签 */

    </style>
</head>
<body>
    <div class="one">
        <img src="../图片/01.jpg" alt="">
        <!-- 方式二 -->
        <!-- <div style="clear:both"></div> -->
        <p>
            如今走过这世间<br>
            万般流连<br>
            翻过岁月不同侧脸<br>
            措不及防闯入你的笑颜<br>
            我曾难自拔于世界之大<br>
            也沉溺于其中梦话<br>
            不得真假 不做挣扎 不惧笑话<br>
            我曾将青春翻涌成她<br>
            也曾指尖弹出盛夏<br>
            心之所动 且就随缘去吧,心之所动 且就随缘去吧心之所动 且就随缘去吧心之所动 且就随缘去吧心之所动
且就随缘去吧心之所动 且就随缘去吧心之所动 且就随缘去吧
        </p>
    </div>
</body>
```

演示

如今走过这世间
万般流连
翻过岁月不同侧脸
措不及防闯入你的笑颜
我曾难自拔于世界之大
也沉溺于其中梦话
不得真假 不做挣扎 不惧笑话
我曾将青春翻涌成她
也曾指尖弹出盛夏
心之所动 且就随缘去吧,心之所动 且就随缘去吧心之所动 且就随缘去吧心之所动 且就随缘去吧心之所动 且就随缘去吧心之所动 且就随缘去吧心之所动 且就随缘去吧



3、浮动案例

案例1：

完成一下内容：


双开门大冰箱
￥10488


双开门大冰箱
￥10488


双开门大冰箱
￥10488


双开门大冰箱
￥10488


双开门大冰箱
￥10488

```
<head>
    <style>
        .head{
            overflow: hidden;
        }
        .head>.sp{
            float: left;
            width: 220px;
            text-align: center;
            /* border: 1px solid; */
            margin: 10px 20px;
            padding-left: 20px;
        }

        .head>.sp>.img{
            width: 200px;
            height: 300px;
            padding-top: 10px;
        }
```

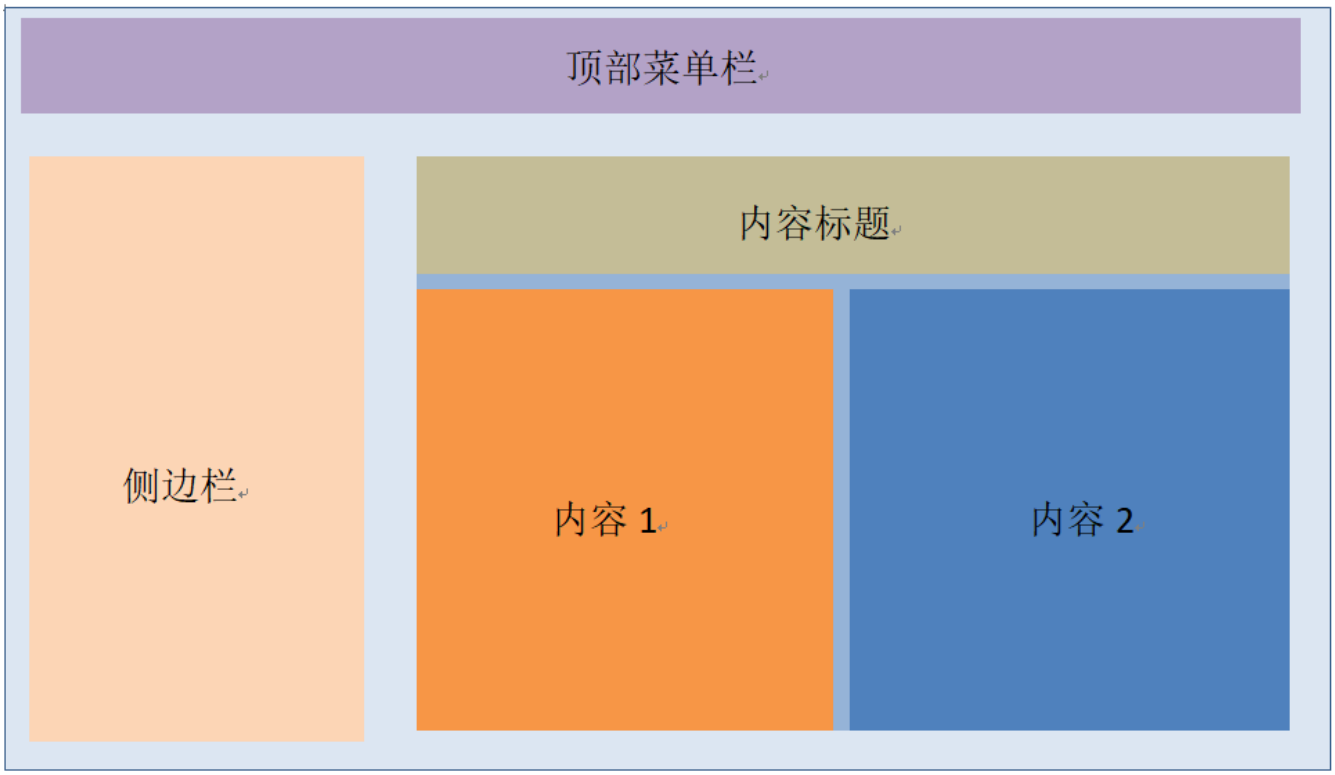
```
        #p2{
            color: red;
        }
    </style>

</head>
<body>
    <div class="head">
        <div class="sp">
            <img src="../../图片/03冰箱.png" alt="" class="img">
            <p id="p1">双开门大冰箱</p>
            <p id="p2">￥10488</p>
        </div>
        <div class="sp">
            <img src="../../图片/03冰箱.png" alt="" class="img">
            <p id="p1">双开门大冰箱</p>
            <p id="p2">￥10488</p>
        </div>
        <div class="sp">
            <img src="../../图片/03冰箱.png" alt="" class="img">
            <p id="p1">双开门大冰箱</p>
            <p id="p2">￥10488</p>
        </div>
        <div class="sp">
            <img src="../../图片/03冰箱.png" alt="" class="img">
            <p id="p1">双开门大冰箱</p>
            <p id="p2">￥10488</p>
        </div>
        <div class="sp">
            <img src="../../图片/03冰箱.png" alt="" class="img">
            <p id="p1">双开门大冰箱</p>
            <p id="p2">￥10488</p>
        </div>
        <div class="sp">
            <img src="../../图片/03冰箱.png" alt="" class="img">
            <p id="p1">双开门大冰箱</p>
            <p id="p2">￥10488</p>
        </div>
        <div class="sp">
            <img src="../../图片/03冰箱.png" alt="" class="img">
            <p id="p1">双开门大冰箱</p>
            <p id="p2">￥10488</p>
        </div>
    </div>

</body>
```

案例2:

```
<head>
    <style>
        .one{
            width: 800px;
            margin:10px auto;
            overflow: hidden;
        }
        .one>li{
            float: left;
            width:42px;
            height: 25px;
             /* 使用高度的时候，字体只能在水平居中，不能垂直居中，这时可以使用line-height */
            margin: 5px 5px;
            /* 去掉列表前面的原点 */
            list-style: none;
            font-size: 15px;
            color: dimgray;
            text-align: center;
        }
        #two{
            background-color: rgb(246, 212, 218);
            color: coral;
            border-radius: 8px;
        }
    </style>
</head>
<body>
    <ul class="one">
        <li id="two">全部</li>
        <li>剧情</li>
        <li>喜剧</li>
        <li>动作</li>
        <li>爱情</li>
        <li>惊悚</li>
        <li>犯罪</li>
        <li>悬疑</li>
        <li>战争</li>
        <li>科幻</li>
        <li>动画</li>
        <li>恐怖</li>
        <li>家庭</li>
        <li>传记</li>
        <li>冒险</li>
        <li>奇幻</li>
        <li>武侠</li>
        <li>历史</li>
        <li>运动</li>
        <li>歌舞</li>
        <li>音乐</li>
        <li>记录</li>
        <li>伦理</li>
    </ul>
</body>
```

作业：

完成此页面布局



代码：

```
<head>
    <style>
        .box{
            width: 710px;
            background-color: aqua;
            text-align: center;
            /* 解决外边距合并 */
            overflow: hidden;
            /* padding-top: 10px; */

        }
        .box1{
            width: 690px;
            height: 30px;
            background-color: darkgreen;
            margin:10px;
        }
        .box2{
            width: 690px;
            height: 360px;
            background-color: aqua;
           margin: 10px 10px;
            overflow: hidden;
        }
        .box3{
```

```
            width: 200px;
            height: 350px;
            background-color: coral;
            float: left;
        }
        .box4{
            width: 480px;
            height: 350px;
            background-color: rgb(162, 198, 248);
            float: left;
            margin-left: 10px;
            overflow: hidden;
        }
        .box5{
            width: 480px;
            height: 30px;
            background-color: chartreuse;
        }
        .box6{
            width: 235px;
            height: 330px;
            background-color: gold;
            float: left;
            margin-top: 10px;
            /* margin-left: 5px; */

        }
        .box7{
            width: 235px;
            height: 330px;
            background-color: rgb(241, 236, 125);
            float: left;
            margin-top: 10px;
            margin-left: 10px;
        }
    </style>
</head>
<body>
    <div class="box">
        <div class="box1">顶部导航栏</div>
        <div class="box2">
            <div class="box3">左边导航栏</div>
            <div class="box4">
                <div class="box5">内容标题</div>
                <div class="box6">内容1</div>
                <div class="box7">内容2</div>
            </div>

        </div>
    </div>

</body>
```

四、浮动扩展

浮动元素与父盒子的关系

子盒子的浮动参照父盒子对齐

不会与父盒子的边框重叠，也不会超过父盒子的内边距

如果父盒子的宽度不够容纳所有的子级盒子，则会换行显示

浮动元素与兄弟盒子的关系

在一个父级盒子中，如果前一个兄弟盒子：

(1)是浮动的，那么当前盒子会与前一个盒子的顶部对齐

(2)是普通流，那么当前盒子会显示在前一个兄弟盒子的下方（浮动只会影响当前的或者后面的盒子，不会影响前面的标准流）

注意

1.如果一个盒子里面有多个子盒子，如果其中一个盒子浮动了，其他兄弟也应该浮动，防止引起问题

2.浮动只会影响当前的或者是后面的标准流盒子，不会影响前面的标准流

五、定位（position）

1、定位介绍

定位也是用来布局的，它有两部分组成：**定位=定位模式+ 边偏移**

注意：**使用定位后，会将元素改为块级元素**

2、边偏移

在 CSS中，通过top、bottom、left和right 属性定义元素的**边偏移**：（方位名词）

边偏移属性	示例	描述
top	top: 80px	**顶端**偏移量，定义元素相对于其父元素**上边线的距离。**
bottom	bottom: 80px	**底部**偏移量，定义元素相对于其父元素**下边线的距离。**
left	left: 80px	**左侧**偏移量，定义元素相对于其父元素**左边线的距离。**
right	right: 80px	**右侧**偏移量，定义元素相对于其父元素**右边线的距离**

定位的盒子有了边偏移才有价值,一般情况下，凡是有定位地方必定有边偏移

3、定位模式(position)

在 CSS 中，通过 position 属性定义元素的**定位模式**，语法如下：

```
选择器{

  position:属性值;

}
```

定位模式是有不同分类的，在不同情况下，我们用到不同的定位模式

取值	特点
static	静态定位是元素的默认定位方式（无定位），指定元素使用**正常的布局**
relative	相对于**自己原来**在标准流中**位置**来进行**移动**的 （**相对定位**）
absolute	相对于最近的非static定位**祖先元素来偏移** （**绝对定位**）
fixed	设置固定定位后 元素不会随滚动条移动而移动 （**固定定位**）

4、定位模式演示

（1）静态定位static

案例

```
<head>
    <style>
        div{
            position: static;
            border: 2px solid blue;
            background-color: rgb(241, 223, 249);
        }
    </style>
```

```
</head>
<body>
    <p>静态定位，始终根据页面的正常流进行定位，无任何特殊定位</p>
    <p>比如：在此标签下放置一个div，设置静态定位</p>
    <div>我设置了静态定位哦</div>
</body>
```

演示

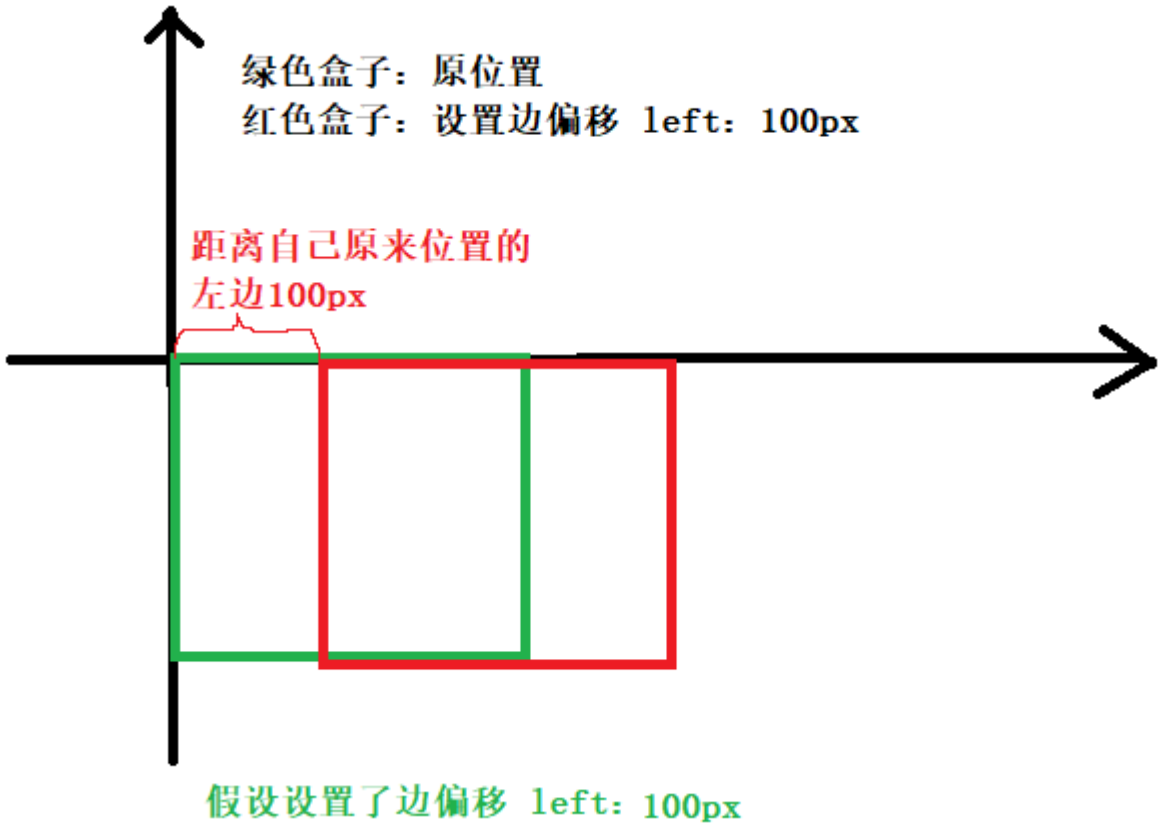
静态定位，始终根据页面的正常流进行定位，无任何特殊定位

比如：在此标签下放置一个div，设置静态定位

我设置了静态定位哦

（2）相对定位（relative）

相对自己原来的位置移动



相对定位的特点：

（1）相对自己原来的位置移动

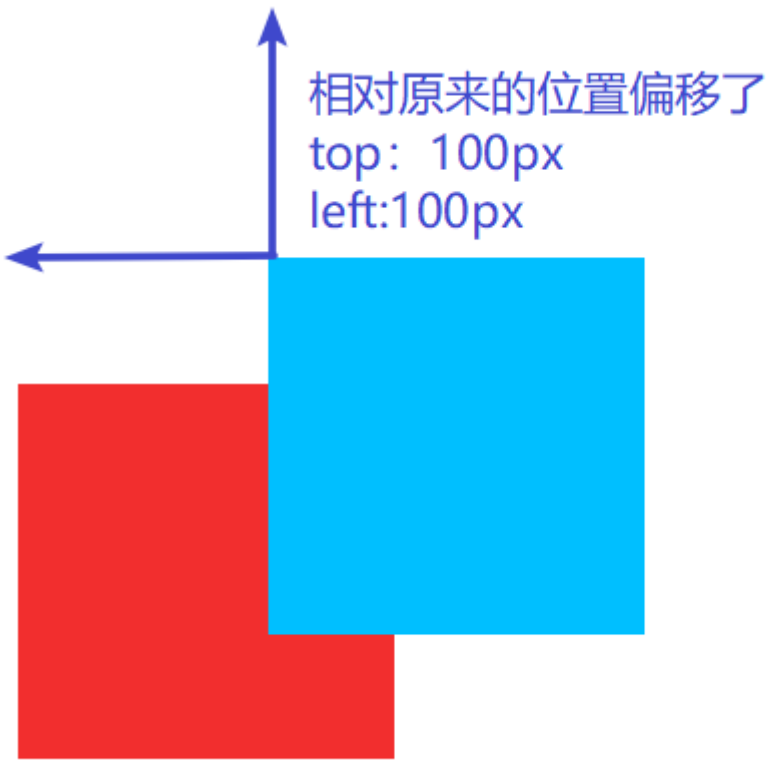
（2）设置相对定位之后（移动了之后），原来的位置依旧保留，如果后面有其他盒子或者标签，依旧以标准流的方式对待它，不会像浮动一样取代原先的位置。

案例：

```
<head>
    <style>
        .box1{
```

```
            width: 150px;
            height: 150px;
            background-color: deepskyblue;

             /* 将第一个盒子设置相对定位 */
            position: relative;
            /* 偏移量  距离上边100px*/
            top: 100px;
            /* 距离左边100px */
            left: 100px;
        }
        .box2{
            width: 150px;
            height: 150px;
            background-color: rgb(242, 46, 46);
        }
    </style>
</head>
<body>
    <div class="box1"></div>
    <div class="box2"></div>
</body>
```

演示



（3）绝对定位（absolute）

根据祖先（父级）元素的位置来移动

绝对定位特点：

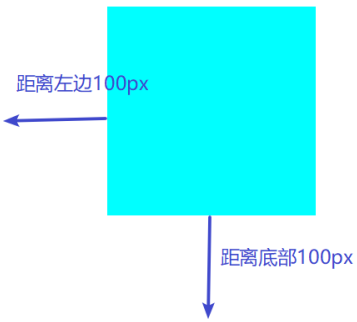
（1）设置绝对定位的盒子如果没有父级或者父级没有设置定位，则相对浏览器页面定位（document文档）

案例

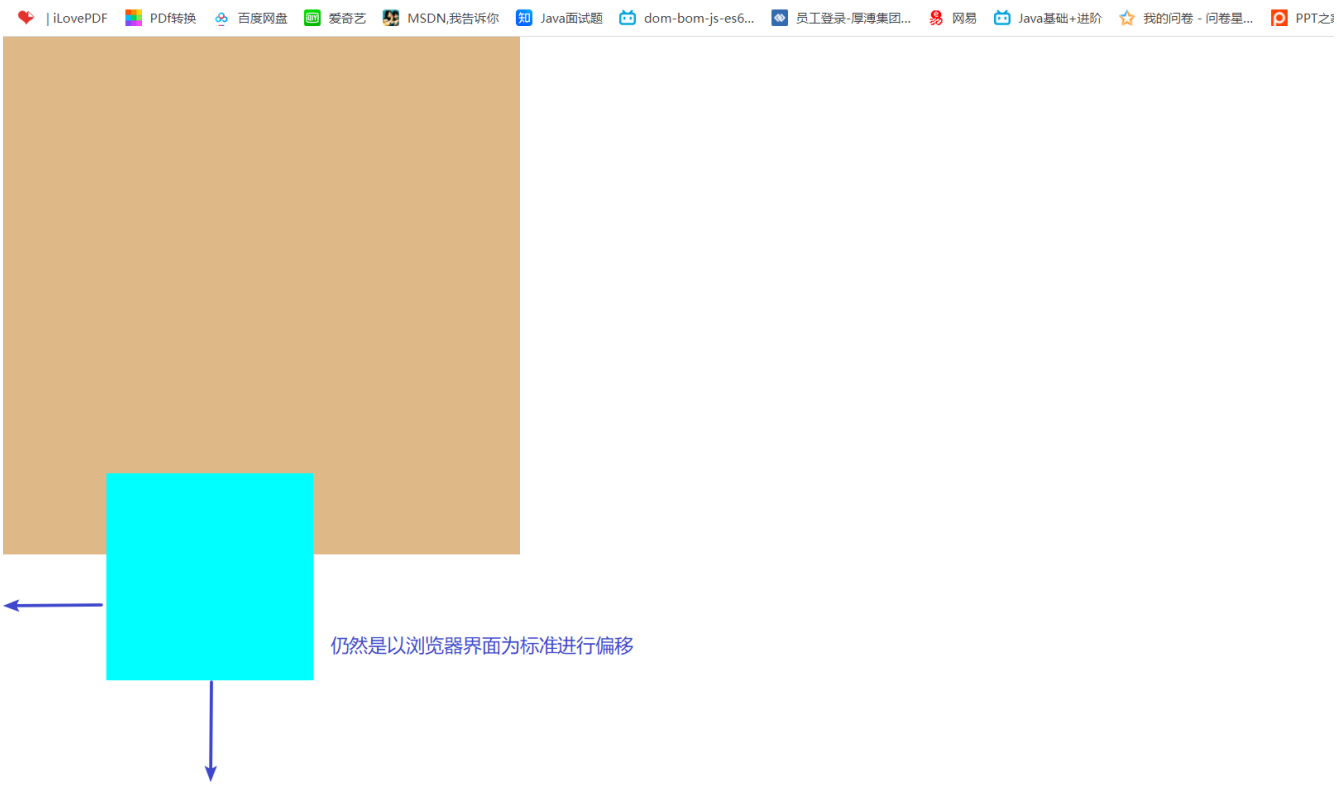
```
<head>
    <style>
        *{
            /* 去除元素自带的内外边距 */
            padding: 0px;
            margin: 0px;
        }
        .son{
            /* 绝对定位 */
            position: absolute;
            bottom: 100px;
            left:100px;
            width: 200px;
            height: 200px;
            background-color: aqua;
        }
    </style>
</head>
<body>
    <div class="son"></div>
</body>
```

演示

（无父元素时）



有父元素但父元素未设置定位时



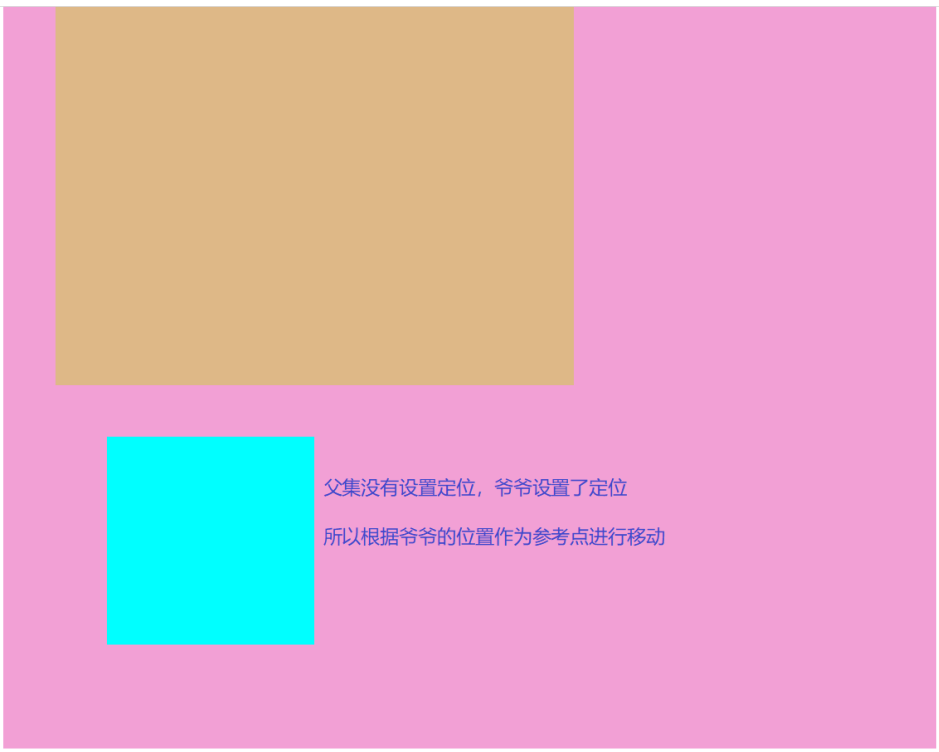
（2）如果祖先元素设置有定位（相对定位，绝对定位，固定定位），则以最近一级带有定位的元素为参考点进行移动

案例

```
<head>
    <style>
        *{
            /* 去除元素自带的内外边距 */
            padding: 0px;
            margin: 0px;
        }
       .gf{
            /* position: relative;
            top: 50px;
            left: 50px; */
            width: 800px;
            height: 800px;
            background-color: rgb(242, 160, 213);
            /* 设置一个内边距，使内容（第二个盒子）距离边框50px*/
            /* padding: 50px; */
        }
        .father{
            /*给父元素设置定位 */
            position: relative;
            top: 50px;
            left: 50px;
```

```
            width: 500px;
            height: 500px;
            background-color: burlywood;
        }
        .son{
            /* 绝对定位 */
            position: absolute;
            bottom:100px;
            left:100px;
            width: 200px;
            height: 200px;
            background-color: aqua;
        }
    </style>
</head>
<body>
    <div class="gf">
        <div class="father">
            <div class="son"></div>
        </div>
    </div>


</body>
```

演示



(3) 绝对定位不再占有之前的位置（脱离标准流）

(4) 子绝父相

子级使用绝对定位的话，父集使用相对定位。

例如：轮播图中左右按钮，以及下方的分页显示

因为这几个按钮是存在于图片之上的，所以是不占用位置的。但是，为什么这三个地方不能使用浮动?

原因：（1）浮动只影响后面的盒子，如果先插入了图片，那么浮动是在图片后面（也就是这几个按钮在图片后面显示，而不是显示在图片上方）

（2）当使用浮动之后，下方的小圆点的盒子会与前面的按钮在同一行显示。若是单独把小圆点浮动，拿到下面来，徒增代码量和工作量。

综上所述：使用“子绝父相”定位方式最简单。

为什么是“子绝父相”而不是其他形式呢?

原因：

（1）子集使用绝对定位，不占用任何位置，就可以放到父集的任何一个地方，而不影响其他的兄弟元素。

（2）父集如果没有定位，则子集的以浏览器为标准进行定位（移动），所以父集必须要有定位。

父集如果也是绝对定位，同样也是不占用位置的，如果父集下面还有其他内容，那下面的内容将会占用父集原先的位置，造成页面混乱。

因此，父集需要使用相对定位。

案例

完成以上轮播图静态界面，忽略小圆点，添加两个按钮即可

```
<head>
    <style>
        .father{
            width: 600px;
            position: relative;
        }
        .father>img{
            width: 600px;
            height: 400px;
        }
        .son1,.son2{
            position: absolute;
            font-size: 50px;
            /* 设置背景颜色，透明度 */
            background-color: rgba(215, 213, 213,0.2);
            /* 设置盒子大小 */
            width: 40px;
            height: 70px;
            border-radius: 10px;
        }
        .son1{
            top: 190px;
            left: 0px;
        }
        .son2{
            top:190px;
            right: 0px;
        }
    </style>
</head>
<body>
    <div class="father">
        <img src="../../图片/04.jpg" alt="">
        <div class="son1"><</div>
        <div class="son2">></div>
    </div>
</body>
```

（5）固定定位

将元素固定在页面的可视区域内，该元素不会随着页面的滚动而滚动。

例如：网站的菜单栏，王者荣耀官网二维码

固定定位的特点

（1）以浏览器的可视窗口为参照点移动元素

- 跟父元素没有关系

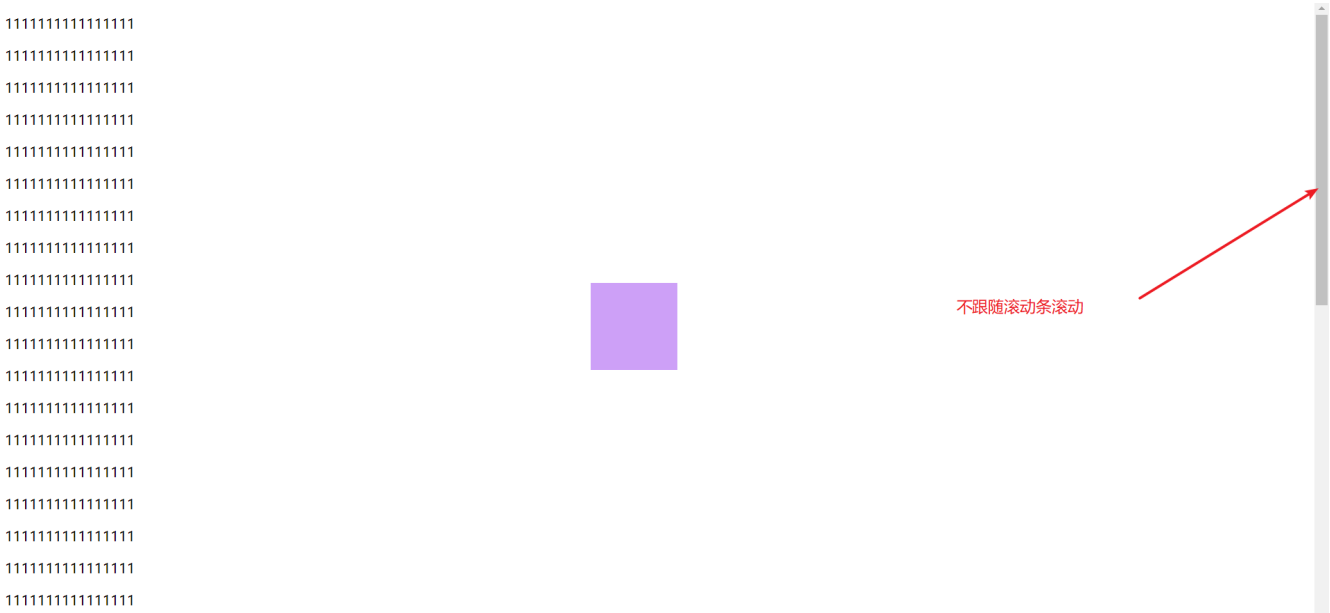
- 不跟随滚动条移动

（2）固定定位不占有原来的位置（是一种特殊的绝对定位，因为当绝对定位没有父元素，或者父元素没有定位的时候，也是以浏览器为参照点移动的，而绝对定位是不占有原来位置的）。

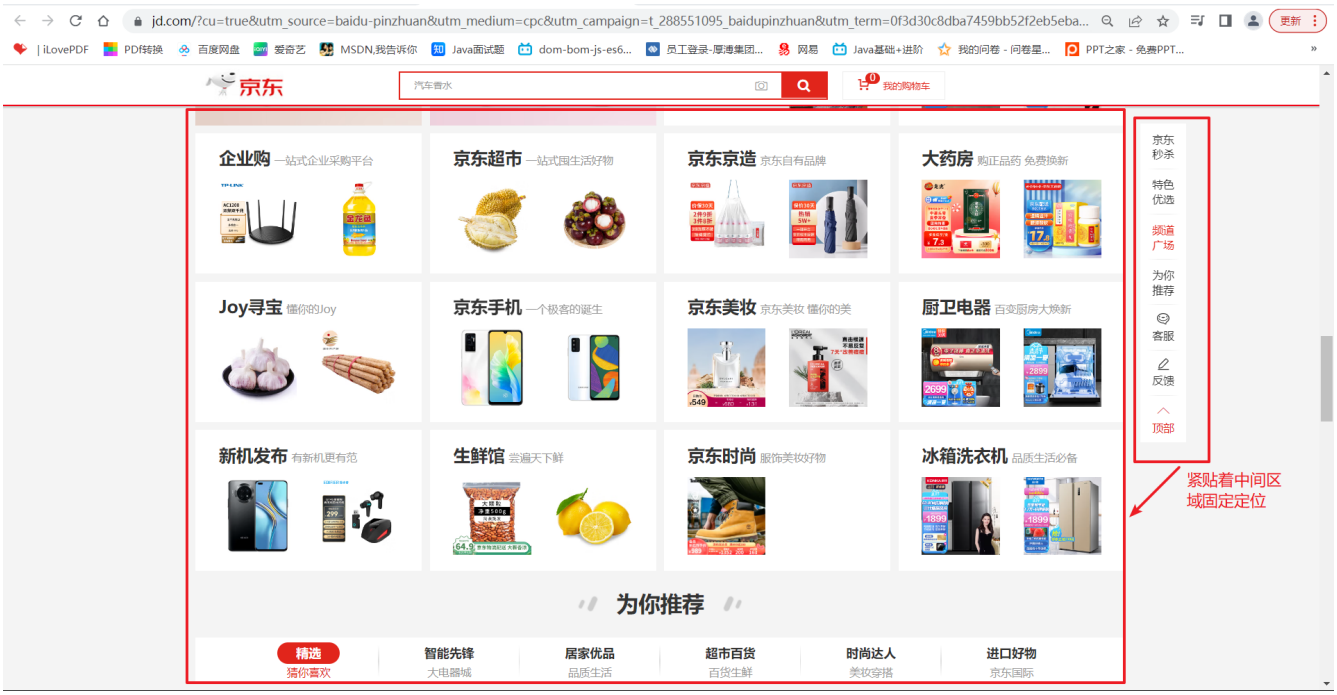
案例

```
<head>
    <style>
        div{
            background-color: rgb(205, 160, 247);
            width: 100px;
            height: 100px;
            position: fixed;
            top: 45%;
            left:45%;

        }
    </style>
</head>
<body>
    <div></div>
    <p>1111111111111111</p>
   ……
</body>
```

演示



（6）固定定位特殊情况

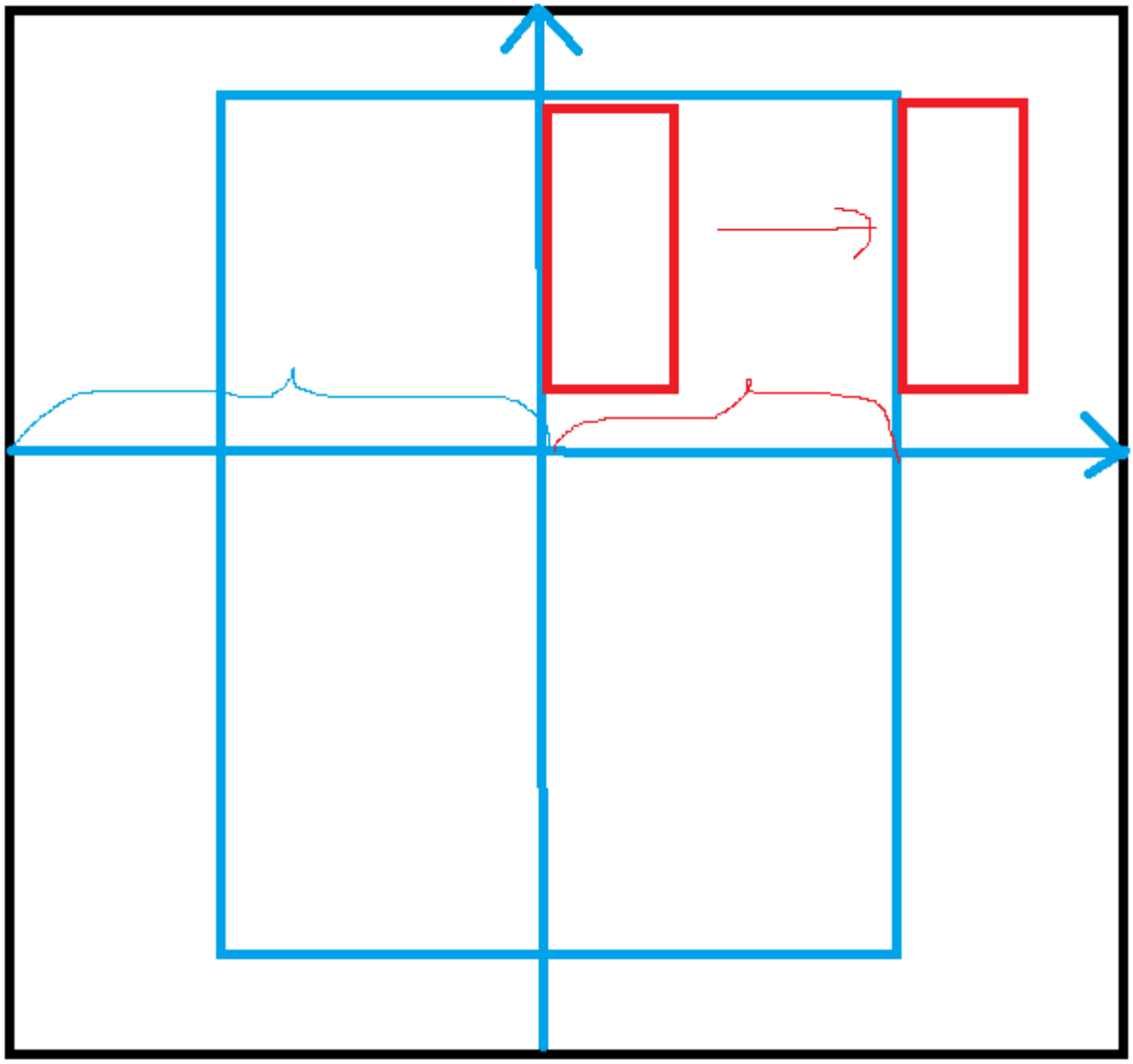


固定定位的盒子紧贴着中间区域的内容（缩放页面可观察）

实现方式：

（1）先将盒子固定到页面中心（left：50%）

（2）再将盒子向右走中心区域的一半距离（margin： px）



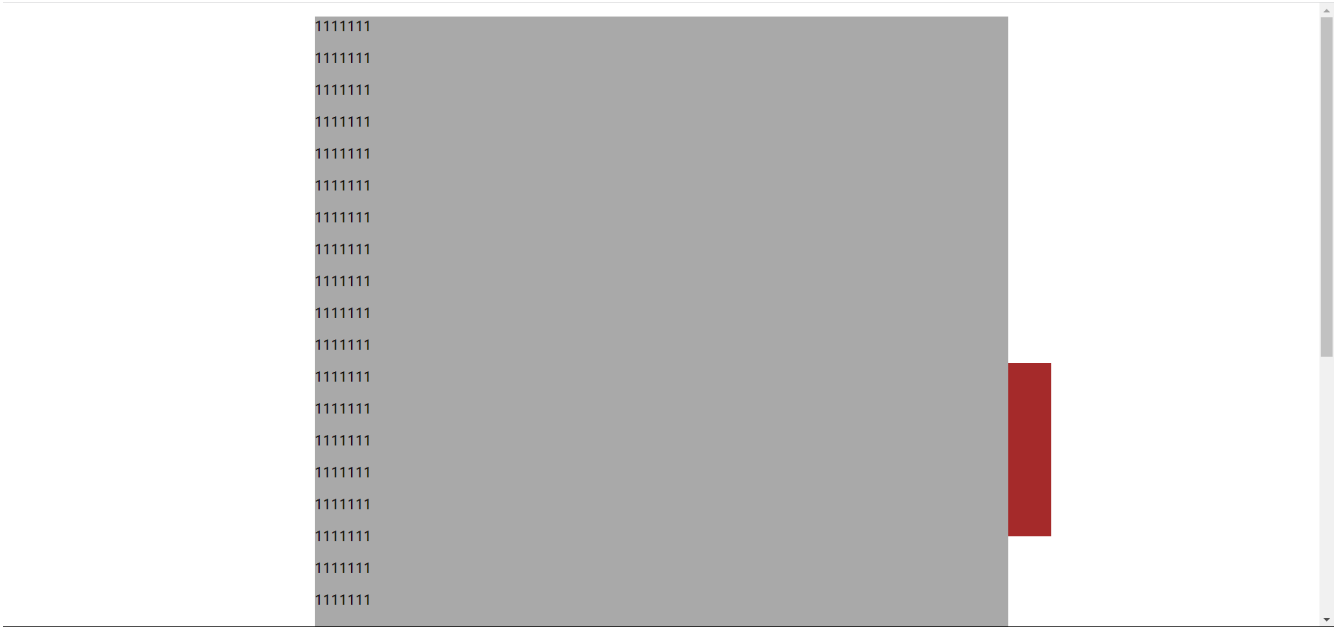
案例

注意：如果中间区域盒子没有加高度,跟随定位的盒子设置了margin之后不能设置定位top和bottom

```
<head>
    <style>
        .one{
            width: 50px;
            height: 200px;
            background-color: brown;
            position: fixed;
            left: 50%;
            /* bottom: 400px; */
            margin: 400px;
        }
        .two{
            width: 800px;
            background-color: darkgray;
            margin: 0 auto;
        }
    </style>
```

```
</head>
<body>
    <div class="one"></div>
    <div class="two">
        <p>1111111</p>
        ……
    </div>
</body>
```

演示



（6）拓展：绝对定位将盒子居中

发现：当盒子设置定位之后，使用margin：0 auto无法实现盒子居中，此时可以通过绝对（固定）定位实现

方法一：

（1）设置绝对定位（固定定位）

（2）先将盒子移动到页面中间（left，top）

（3）再减去盒子一半的宽度，margin为负值（因为设置居中时，是以盒子的左边框和上边框为基准）

方法二：

（1）设置绝对定位（固定定位）

（2）给4个方向都设置为0；用margin自动，实现居中

案例

```
<head>
    <style>
        div{
            width: 200px;
            height: 200px;
            background-color: blue;
```

```
            position: absolute;
            /* 方法一 */
            /* left: 50%;
            margin-left: -100px;
            top: 50%;
            margin-top: -100px; */

            /* 方法二 */
            top: 0;
            left: 0;
            right: 0;
            bottom: 0;
            margin: auto;
        }
    </style>
</head>
<body>
    <div></div>
</body>
```

演示