第5章 网页布局基础

【目标任务】

学习目标	1. 理解 CSS+DIV 布局思路与优势 2. 掌握浮动的原理及应用 3. 掌握清除浮动的方法 4. 掌握 overflow 的应用 5. 掌握相对定位的原理及应用 6. 掌握绝对定位的原理及应用 7. 掌握固定定位的原理及应用 8. 掌握 z-index 层叠的应用
重点知识	1. 浮动的原理 2. 相对定位的原理 3. 绝对定位的原理 4. 固定定位的原理
项目实战	项目 5-1 “国”字型布局网页效果制作 项目 5-2 商品广告版块效果制作
实训作业	实训任务 5-1 新浪微博发言版块制作 实训任务 5-2 制作第 5 章案例作业网站

【知识技能】

5.1 DIV+CSS 布局的思路

网页最开始的呈现就像一张白纸，我们要像排版报纸一样规划网页各个板块的大小、位置等。div 是一个块级元素，相当于一个容器，容器中可以嵌套标题、段落、表格、图片等 HTML 元素。因此，页面布局时，先使用 标签对页面进行板块划分，描述页面整体结构，如建立 id 属性为 top、main(嵌套 left、mid、right)、bottom 的 6 个 div。使用 CSS 盒子模型描述每一个 div 的 width、height、border、padding、margin、background 等属性，并把盒子放到合适的位置。

CSS+DIV 是网页制作中常见的术语之一，掌握基于 CSS 和 DIV 技术的网页布局方式，是实现 Web 标准的基础。DIV 负责网页的结构、CSS 负责网页的样式，实现结构与样式

的分离。结构与样式的分离使得文档结构清晰,代码简洁,有效提高页面浏览速度,缩减带宽成本,易于搜索引擎的搜索;只要修改 CSS 代码即可实现网页的改版,后期维护方便。

5.2 文 档 流

HTML 元素除专门制定,其他所有 HTML 元素都在文档流中定位。块级元素独占一行,它的位置按照元素在 HTML 代码中按先后顺序自上而下排列。行内元素在一行中并排显示,它的位置在每行中按照元素在 HTML 代码中的先后顺序从左到右地顺序排列。如例 5-1 的代码和图 5-1 所示,div 是块级元素,3 个 div 各占一行;span 是行内元素,两个 span 元素不换行。若不进行任何定位与浮动设置,则 div 只能实现“三”字形布局。

【例 5-1】 文档流布局(案例文件:chapter05\example\exp5_1.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>文档流布局</title>
    <style type="text/css">
        body{font-size: 24px; }
        div{padding: 0; margin: 0px; text-align: center; line-height: 50px; }
        #box1{
            width: 50px;
            height: 50px;
            background-color: #666;
        }
        #box2{
            width: 150px;
            height: 50px;
            background-color: #999;
        }
        #box3{
            width: 250px;
            height: 50px;
            background-color: #ccc;
        }
        .span1{
            color: red;
            font-weight: bold;
        }
        .span2{
            color: blue;
```

```
                text-decoration: underline;
        }
</style>
</head>
    <body>
        <div id = "box1" > box1 </div>
        <div id = "box2" > box2 </div>
        <div id = "box3" > box3 </div>
        <p> span 是行内元素,
<span  class = "span1" > span 定义的行内元素 </span>
<span class = "span2" > span 定义的行内元素 </span> </p>
    </body>
</html>
```

例 5-1 的运行效果如图 5-1 所示。

图 5-1　文档流布局

5.3　Float 浮动

由例 5-1 可见,div 是块级元素,每一个 div 默认占一整行,只能实现“三”字型布局。若想使两个 div 在一行并列显示,需要使用浮动属性。元素的浮动是指设置了浮动属性的元素会脱离标准文档流的控制,移动到指定位置的过程。

5.3.1　浮动的基本语法

float 通过 CSS 的 float 属性进行设置,可以设置对象左右浮动,直到其边缘碰到它的父元素或另一个浮动元素的边缘。浮动的语法格式如下。

```
选择器{float: left/right/none/inherit; }
```

在以上的语法中,float 属性的常用值有 4 个,分别表示不同的含义,具体介绍如下。

- left　元素向其父元素的左侧边缘浮动。

- right　元素向其父元素的右侧边缘浮动。
- none　默认值,元素不浮动。
- inherit　从父级元素获取 float 值。

5.3.2　浮动的基本原理

【例 5-2】　三列布局的实现(案例文件:chapter05\example\exp5_2.html)

1. 创建基本 HTML 与 CSS(案例文件:chapter05\example\exp5_2_Step1.html)

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>浮动的原理</title>
        <style type="text/css">
            body{font-size: 24px; }
            div{padding: 0; margin: 0px; text-align: center; }
            #box1{
                width: 50px;
                height: 50px;
                background-color: #666;
            }
            #box2{
                width: 150px;
                height: 50px;
                background-color: #999;
            }
            #box3{
                width: 250px;
                height: 50px;
                background-color: #ccc;
            }
        </style>
    </head>
    <body>
        <div id="box1">box1</div>
        <div id="box2">box2</div>
        <div id="box3">box3</div>
    </body>
</html>
```

代码运行效果如图 5-2 所示。

不发生浮动时,块级元素独占一行,紧随其后的块级元素按照在文档流中的顺序依次在

图 5-2　文档流布局

新行显示。

2. 设置第一个 div 浮动(案例文件：chapter05\example\exp5_2_Step2.html)

修改 box1 的代码,设置 float 属性为 left 对 id="box1"应用左浮动,代码如下。

```
#box1{
……
    float: left;
}
```

修改 box1 代码后的运行效果如图 5-3 所示。

图 5-3　box1 左浮动效果

box1 设置左浮动,它脱离文档流而向左移动,直到其边缘碰到包含框(此处为 body)的边缘为止。因为 box1 脱离文档的普通流(浮在上面),box2 会代替原来 box1 的位置,其左边框与 box1 重合。

3. 设置第 2 个 div 浮动(案例文件：chapter05\example\exp5_2_Step3.html)

修改 box2 的代码,设置 float 属性为 left 对 id="box2"应用左浮动,代码如下。

```
#box2{
    ……
    float: left;
}
```

修改 box2 代码后的运行效果如图 5-4 所示。

图 5-4　box1 和 box2 左浮动效果

box2 设置左浮动,它脱离文档流而向左移动,直到其边缘碰到它前面一个浮动元素(box1)的边缘为止。浮动的元素脱离文档流,box3 会代替原来 box1 的位置,其左边框与 box1 重合。其中,box1 与 box2 所占宽度为 50px+150px=200px,所以 box3 露出了 50px(250px−200px=50px)宽的区域。

4. 设置第 3 个 div 层浮动(案例文件:chapter05\example\exp5_2.html)

若要实现 3 个盒子在一行显示,即“川”字形布局,为 # box3 设置左浮动即可,修改 box3 的代码,设置 float 属性为 left 对 id=“box2”应用左浮动,代码如下。

```
#box3{
    ……
    float: left;
}
```

修改 box3 代码后的运行效果如图 5-5 所示。

图 5-5　“川”字型布局

在 CSS 中,任何元素都可以浮动,浮动元素会生成一个块级框。不论它本身是何种元素,常常通过对 div 元素设置 float 属性进行布局。

- 不发生浮动时,块级元素独占一行,紧随其后的块级元素将在新行显示。
- 设置浮动,浮动框会脱离文档的普通流。后面的元素向前浮动。
- 对象左(右)边没有其他浮动元素时,对象左(右)浮动,直到其边缘碰到它的父元素的边缘。
- 对象左(右)边有其他浮动元素时,对象左(右)浮动,直到其边缘碰到另一个浮动元素的边缘。

5.3.3 清除浮动 clear 属性

例 5-2 中的第三步(案例文件：chapter05\example\exp5_2_Step3.html)中 box1 和 box2 与 box3 重叠，这是浮动元素对未设置浮动的元素产生的影响。由于浮动元素不再占用原文档流的位置，未设置浮动的元素向上移动，所以会对页面中的排版产生影响。在 CSS 中，clear 属性用于清除浮动的影响，其基本语法格式如下。

```
选择器{clear: left/right/both; }
```

clear 属性的常用值有 3 个，分别表示不同的含义，具体如下。

- left。清除左侧浮动的影响。
- right。清除右侧浮动的影响。
- both。清除所有浮动的影响。

【例 5-3】 清除浮动(案例文件：chapter05\example\exp5_3.html)

```
<!DOCTYPE html>
<html lang = "en" >
    <head>
        <meta charset = "UTF-8" >
        <title>浮动的原理</title>
        <style type = "text/css" >
            body{font-size: 24px; }
            div{padding: 0; margin: 0px; text-align: center; }
            #box1{
                width: 50px;
                height: 50px;
                background-color: #666;
                float: left;
            }
            #box2{
                width: 150px;
                height: 50px;
                background-color: #999;
                float: left;
            }
            #box3{
                width: 250px;
                height: 50px;
                background-color: #ccc;
                clear: left;     /*清除浮动*/
            }
        </style>
```

```
    </head>
    <body>
        <div id = "box1" > box1 </div>
        <div id = "box2" > box2 </div>
        <div id = "box3" > box3 </div>
    </body>
</html>
```

例 5-3 的运行效果如图 5-6 所示。

图 5-6　box3 清除浮动效果

5.3.4　子元素浮动,父元素空间不足的情况

1. 子元素浮动,父元素空间不足,子元素被移至下一行

如果父元素的 width 值不足,无法容纳水平排列的多个设置了浮动属性的子元素,即父元素可供浮动的空间小于各个浮动子元素所占空间之和,那么处于后面的浮动子元素会被移至下一行,直到父元素有足够的空间,才会移上去。

```
<!DOCTYPE html>
<html lang = "en" >
    <head>
        <meta charset = "UTF-8" >
        <title> 浮动的原理 - 父盒子宽度不足 </title>
        <style type = "text/css" >
            div{padding: 0; margin: 0; }
            body{font-size: 24px; }
            #page{
                width: 612px;
                height: 100px;
                background-color: #ccc;
            }
            #box1, #box2, #box3 {
                width: 200px;
                height: 50px;
                background-color: #666;
                border: 2px solid red;
```

```
                float: left;
            }
        </style>
    </head>
    <body>
        <div id = "page" >
            <div id = "box1" > box1 </div>
            <div id = "box2" > box2 </div>
            <div id = "box3" > box3 </div>
        </div>
    </body>
</html>
```

【例 5-4】 父元素空间不足(案例文件：chapter05\example\exp5_4.html)

1. 创建基本的 HTML 与 CSS(案例文件：chapter05\example\exp5_4_Step1.html)

代码运行效果如图 5-7 所示。

图 5-7 三列浮动效果

3 个子盒子所占宽度为(200px+2px+2px)×3=612px，当父元素 #page 的宽度大于或等于 3 个子盒子所占宽度时，3 个盒子在一行显示。

修改父盒子 page 的宽度，使父盒子宽度不足以容纳 3 个子元素(案例文件：chapter05\example\exp5_4_Step2.html)，代码如下。

```
#page{
    width: 600px;          /* width 不足 612px */
    ……
}
```

修改代码后的运行效果如图 5-8 所示。

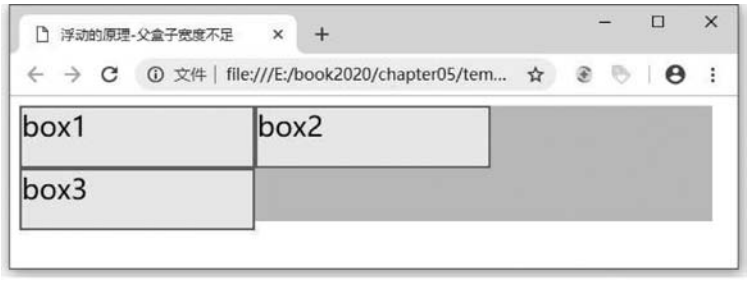

图 5-8 父元素宽度不足

当父元素 page 的宽度小于 3 个盒子所占宽度之和时,最后一个子盒子被移至下一行。

2. 子元素浮动,父元素空间不足,子元素移至下一行时被卡住

在实际项目开发时,我们经常不设置子盒子的高度,而是让内容把子盒子的高度撑开,这样会出现 3 个子盒子高度不一致的情况。如果浮动子元素的高度不同,那么当它向下移动时可能被其前面高度较高的浮动元素"卡住"。

【例 5-5】 被移至下一行的子元素被卡住(案例文件:chapter05\example\exp5_5.html)

修改例 5-4 的 CSS 代码,使 box1 的高度设置为高于 box2 和 box3,代码如下。

```
<style type="text/css">
    div{padding: 0; margin: 0; }
    body{font-size: 24px; }
    #page{
        width: 600px; /*width 不足 612px*/
        height: 100px;
        background-color: #ccc;
    }
    #box1, #box2, #box3 {
        width: 200px;
        height: 50px;
        background-color: #666;
        border: 2px solid red;
        float: left;
    }
    #box1{
        height: 80px;
    }
</style>
```

例 5-5 的代码运行效果如图 5-9 所示。

图 5-9 box1 高度"卡住"了 box3

因 box1 高度较高,box3 在移至下一行时,被 box1 卡住了。

5.3.5 浮动子元素对未设置高度的父元素的影响及解决办法

在制作网页时，经常不设置父元素的高度，而是让子元素把父元素撑开。这时，设置了浮动的子元素会对未设置浮动、未设置高度的父元素产生影响。如例 5-6 所示，父元素变成了一条直线。

【例 5-6】 父元素无高度（案例文件：chapter05\example\exp5_6.html）

代码如下：

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>子元素浮动对父元素的影响</title>
        <style type="text/css">
            #box{
                width: 372px;   /*(100+20+4)*3=372*/
                border: dashed 2px #F00;
                background-color: #ccc;
            }
            #left,#main,#right{
                width: 100px;
                height: 80px;
                text-align: center;
                font-size: 20px;
                border: solid 2px #000;
                margin: 10px;
                background-color: pink;
                float: left;
            }
        </style>
    </head>
    <body>
        <div id="box">
            <div id="left">left    </div>
            <div id="main">main    </div>
            <div id="right">right  </div>
        </div>
        <!-- 对子元素设置浮动时，如果不对其父元素定义高度，则子元素的浮动会对父元素产生影响，如例 5-6 所示父元素变成了一条直线. -->
    </body>
</html>
```

例 5-6 的代码运行效果如图 5-10 所示。

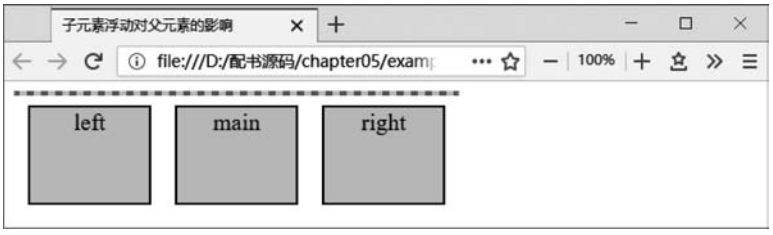

图 5-10　父元素未被子元素撑开

由于父元素没有设置高度、浮动,而子元素设置浮动,脱离文档流,子元素不能把父元素撑开,父元素的背景颜色只显示在一行。

clear 属性只能清除元素左右两侧浮动的影响。子元素的浮动对父元素产生影响,二者属于嵌套关系,clear 解决不了这种浮动影响。常见清除浮动的解决办法有以下几种。

1. 使用空标签清除浮动

在浮动元素之后添加空标签,并对该标签应用"clear: both"样式,可清除子元素浮动、父元素没设置高度所产生的影响,这个空标签可以为 、<p>、<hr />等任何标签。

【例 5-7】 使用空标签清除浮动(案例文件: chapter05\example\exp5_7.html)

将例 5-6 中的 HTML 结构代码修改如下。

```
<div id="box">
    <div id="left">left    </div>
    <div id="main">main    </div>
    <div id="right">right  </div>
    <div style="clear: both; "> </div>  <!--使用空标签清除浮动-->
</div>
```

例 5-7 的代码运行效果如图 5-11 所示。

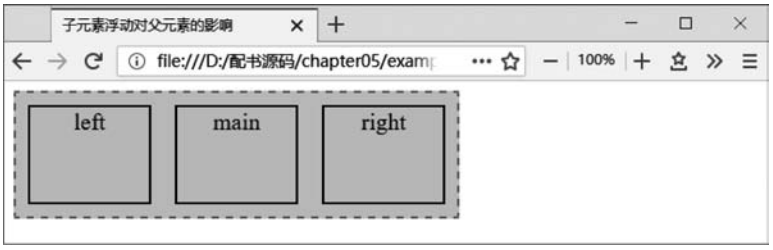

图 5-11　父元素被撑开

图 5-11 中,子元素浮动对父元素的影响已经消除。由于上述方法在无形中增加了毫无意义的 HTML 结构元素(空标签),因此在实际工作中并不建议使用这种改变 HTML 结构的方法,通常我们会采用 CSS 的伪元素解决这个问题。

2. 使用 CSS 的伪元素清除浮动

伪元素(:after)是在某元素后面附加一个元素,早在 CSS2 版本中就提供了伪元素,例

如：before 表示在某元素前面插入新内容，而：after 表示在某元素之后插入新内容，其语法格式如下。

```
: before
{
    content: url(images/a.jpg);
}
: after
{
    content: url(logo.gif);
}
```

现在的主流浏览器都支持伪元素，默认伪元素是行内元素，可以使用 display 属性转换为块级元素。在 CSS3 中，伪元素使用两个冒号“::”表示，即“:: after”以及“:: before”等，以区别伪类选择器，CSS3 中伪类选择器使用一个冒号表示。

使用 CSS 的伪元素(：after/:: after)清除浮动，只需要将 content 属性设置为空，将元素后面新生成的内容设置为不占空间，设置伪元素高度值为 0。以下代码为清除浮动的经典代码。

```
#box: after{                    /* 对父元素应用 after 伪对象样式 */
    content: "";                /* 内容为空 */
    height: 0;
    display: block;             /* 转换为块级元素 */
    clear: both;
    visibility: hidden;         /* 元素不可见 */
}
```

以上代码用于设置了浮动属性的元素的父元素之上，为父元素应用伪元素，设置以上样式代码，以达到清除浮动的目的。

3. 使用 overflow 属性清除浮动

对父元素应用“overflow：hidden;”样式，也可以实现清除子元素浮动对父元素的影响。有时还常见后面会跟一句 zoom：1；这是用来兼容 IE 6 浏览器，不过 IE 6 现在已基本退出历史舞台了，因此这句代码不再是那么重要了。在父元素的样式代码中使用 overflow 属性清除浮动，代码简洁使用方便。如果父元素中使用了定位，则此句代码还有溢出隐藏的效果，其清除浮动功能就会受到影响，因此实践中比较推荐使用伪元素清除浮动的方法。

【例 5-8】 使用 overflow 属性清除浮动(案例文件：chapter05\example\exp5_8.html)

将例 5-6 中的 CSS 代码修改如下。

```
#box{
……
overflow: hidden; /* 使用 overflow 属性清除浮动 */
}
```

代码运行效果如图 5-11 所示，子元素浮动对父元素的影响已经清除，背景正常显示。当父元素浮动时，未设置高度的父元素也能被子元素撑开。

【例 5-9】 父元素也浮动(案例文件：chapter05\example\exp5_9.html)

将例 5-6 中的 CSS 代码修改如下。

```
#box{
    ……
    float: left;
}
```

代码运行效果如图 5-11 所示，子元素浮动对父元素的影响已经清除，背景正常显示。

5.4 overflow 属性

当盒子的内容超出盒子自身的大小时，内容就会溢出。规范溢出内容的显示方式，需要使用 CSS 的 overflow 属性，其基本语法格式如下。

```
选择器{overflow: visible/ hidden/ auto/ scroll; }
```

在上面的语法中，overflow 属性的常用值有 4 个，分别表示不同的含义，具体如下。

- visible：盒子溢出的内容不会被修剪，而呈现在元素框之外。
- hidden：盒子溢出的内容将会被修剪且不可见。
- auto：元素框能够自动适应其内容的多少，在内容溢出时，产生滚动条。
- scroll：盒子溢出的内容将会被修剪，且元素框始终产生滚动条。不论元素是否溢出，元素框中的水平和竖直方向的滚动条都始终存在。

【例 5-10】 overflow 属性(案例文件：chapter05\example\exp5_10.html)

代码如下：

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>overflow-y 属性</title>
        <style type="text/css">
            div{
                width: 180px;
                height: 150px;
                font-size: 12px;
                line-height: 24px;
                padding: 5px;
                margin: 10px;
                float: left;
                background-color: #9f0;
            }
            #box1{overflow: visible; }
```

```
            #box2{overflow: hidden; }
            #box3{overflow: auto; }
            #box4{overflow: scroll; }
            #box5{overflow-y: scroll; }
        </style>
    </head>
    <body>
        <div id="box1">
            <h3>overflow: visible;</h3>
            <p>盒子溢出的内容不会被修剪,而呈现在元素框之外.</p>
            <p>测试文本……</p>
        </div>
        <div id="box2">
            <h3>overflow: hidden;</h3>
            <p>盒子溢出的内容将会被修剪且不可见.</p>
            <p>测试文本……</p>
        </div>
        <div id="box3">
            <h3>overflow: auto;</h3>
            <p>在内容溢出时,产生滚动条,否则,则不会产生滚动条.</p>
            <p>测试文本……</p>
        </div>
        <div id="box3">
            <h3>overflow: auto;</h3>
            <p>在内容不溢出时,不产生滚动条.</p>
        </div>
        <div id="box4">
            <h3>overflow: scroll;</h3>
            <p>盒子溢出的内容将会被修剪,且元素框始终产生滚动条.</p>
            <p>测试文本……</p>
        </div>
        <div id="box5">
            <h3>overflow: scroll;</h3>
            <p>内容不溢出时也产生滚动条.</p>
        </div>
    </body>
</html>
```

例 5-10 的代码运行效果如图 5-12 所示。

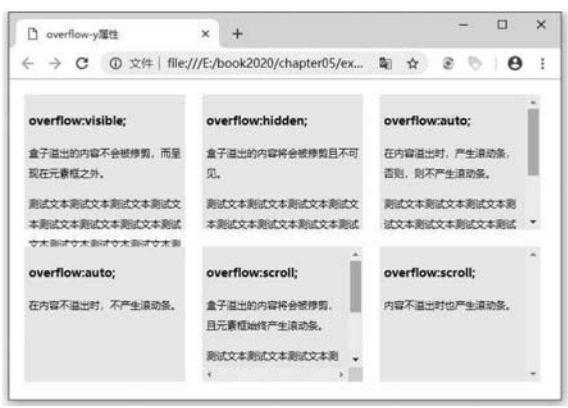

图 5-12　overflow 取值效果

5.5 定位属性

float只能控制元素向左、向右浮动,当需要为元素进行精确定位时,则需要使用定位属性。

5.5.1 定位属性

元素的定位属性包括定位模式和边偏移两部分。

1. 定位模式

在CSS中,position属性用于定义元素的定位模式,其基本语法格式如下。

```
选择器{position: static/ relative/ absolute/ fixed; }
```

在上面的语法中,position属性的常用值有4个,分别表示不同的定位模式,具体如下。

- static:自动定位(默认定位方式)。
- relative:相对定位,相对于其原文档流的位置进行定位。
- absolute:绝对定位,相对于上一个已经定位的父元素进行定位。
- fixed:固定定位,相对于浏览器窗口进行定位。

2. 边偏移

通过边偏移属性top、bottom、left或right,精确定位元素的位置,在本书的阐述中,常用TBRL代表这4个边偏移量。TBRL的取值为不同单位的数值或百分比,对它们的具体解释如下。

- top:顶端偏移量。
- bottom:底部偏移量。
- left:左侧偏移量。
- right:右侧偏移量。

5.5.2 静态定位

静态位置是各个元素在HTML文档流中默认的位置。静态定位是元素的默认定位方式,当position属性的取值为static时,可以将元素定位于静态位置。

任何元素在默认状态下都会以静态定位来确定自身的位置,当没有定义position属性时,并不说明该元素没有自身的位置,元素会遵循默认值显示为静态位置。在静态定位状态下,无法通过边偏移属性(top、bottom、left或right)改变元素的位置。

5.5.3 相对定位

相对定位是将元素相对于它在标准文档流中的正常位置进行定位,当position属性的取值为relative时,可以通过TBRL边偏移属性改变元素的位置,将元素定位于相对位置。对元素设置相对定位后,它在文档流中的位置仍然保留。

【例5-11】 相对定位的原理

1. 创建基本HTML与CSS(案例文件:chapter05\example\exp5_11_Step1.html)

```
<!DOCTYPE html>
<html lang = "en" >
    <head>
        <meta charset = "UTF-8" >
        <title> 相对定位 </title>
        <style type = "text/css" >
            * {padding: 0; margin: 0; }
            body{font-size: 24px; }
            #father-box{
                width: 300px;
                height: 200px;
                background-color: #ccc;
                padding: 50px;
                margin: 0 auto;
            }
            #box1, #box2 {
                width: 100px;
                height: 50px;
                color: #fff;
                line-height: 50px;
                text-align: center;
                background-color: #333;
            }
            #box2{
                margin-top: 50px;
            }
        </style>
    </head>
    <body>
        <div id = "father-box" >
            <div id = "box1" > box1 </div>
            <div id = "box2" > box2 </div>
        </div>
    </body>
</html>
```

代码运行效果如图 5-13 所示。

不设置相对定位时，块级元素独占一行，按照在文档流中的顺序依次向下排列。

2. 为 box1 设置相对定位并设置边偏移量 TL（案例文件：chapter05\example\exp5_11_Step2. html）

```
#box1{
```

```
    position: relative;
    top: 100px;
    left: 150px;
}
```

设置 box1 后的代码运行效果如图 5-14 所示。

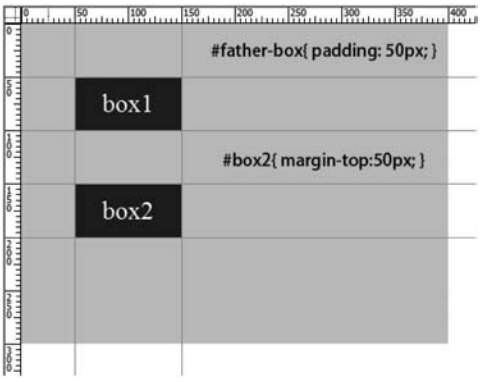

图 5-13　未定位状态

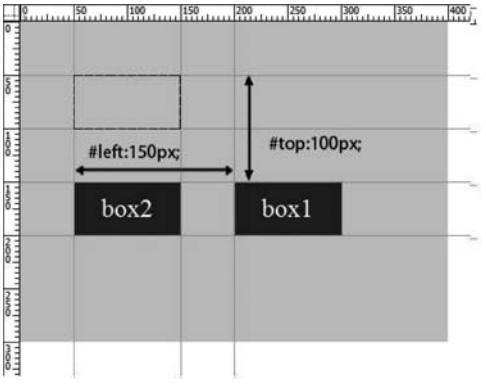

图 5-14　相对定位原理

对 box1 设置相对定位后,它会相对于其自身原来的位置进行偏移,但是它在文档流中的位置仍然保留,虚线框为 box1 原来的位置。语句“top: 100px;”使 box1 向下移动 100px,语句“left: 150px;”使 box1 向右移动 150px。

3. 为 box1 设置相对定位并设置边偏移量 BR(案例文件: chapter05\example\exp5_11_Step3.html)

```
#box1{
    position: relative;
    bottom: 50px;
    right: 50px;
}
```

设置 box1 后的代码运行效果如图 5-15 所示。

虚线框为 box1 原来的位置,语句“bottom: 50px;”使 box1 向上移动 50px,语句“right: 50px;”使 box1 向左移动了 50px。

- top: 顶端偏移量,定义元素相对于原来位置向下移动的距离。
- bottom: 底部偏移量,定义元素相对于原来位置向上移动的距离。
- left: 左侧偏移量,定义元素相对于原来位置向右移动的距离。
- right: 右侧偏移量,定义元素相对于原来位置向左移动的距离。

4. 边偏移量可以取负值(案例文件: chapter05\example\exp5_11_Step4.html)

```
#box1{
    position: relative;
```

```
    top: -20px;
    left: -20px;
}
```

设置 box1 后的代码运行效果如图 5-16 所示。

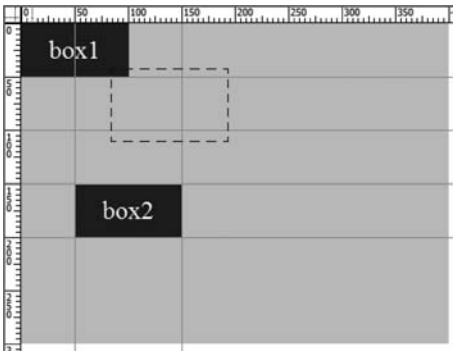

图 5-15　相对定位边偏移量

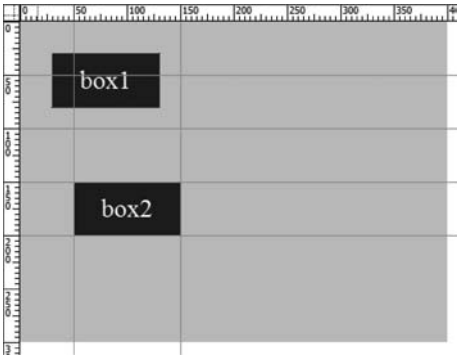

图 5-16　相对定位边偏移量

TBRL 的值可以取负值，此时，“top：－20px；”“left：－20px；”的效果与“bottom：20px；”“right：20px；”相同。

5.5.4　绝对定位

当 position 属性的取值为 absolute 时，可以将元素的定位模式设置为绝对定位。绝对定位是将元素依据最近的已经定位（绝对、固定或相对定位）的祖先元素进行定位，若所有祖先元素都没有定位，则依据 body 根元素（浏览器窗口）进行定位。

【例 5-12】　绝对定位的原理

1. 创建基本 HTML 与 CSS（案例文件：chapter05\example\exp5_12_Step1.html）

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>绝对定位</title>
        <style type="text/css">
            *{padding: 0; margin: 0; }
            body{font-size: 24px; }
            #father-box{
                width: 300px;
                height: 200px;
                background-color: #ccc;
                padding: 50px;
                margin: 0 auto;
            }
```

```
            #box1,#box2 {
                width: 100px;
                height: 50px;
                color: #fff;
                line-height: 50px;
                text-align: center;
                background-color: #333;
            }
            #box2{
                margin-top: 50px;
            }
        </style>
    </head>
<body>
    <div id="father-box">
        <div id="box1">box1</div>
        <div id="box2">box2</div>
    </div>
    </body>
</html>
```

代码运行效果如图 5-17 所示。

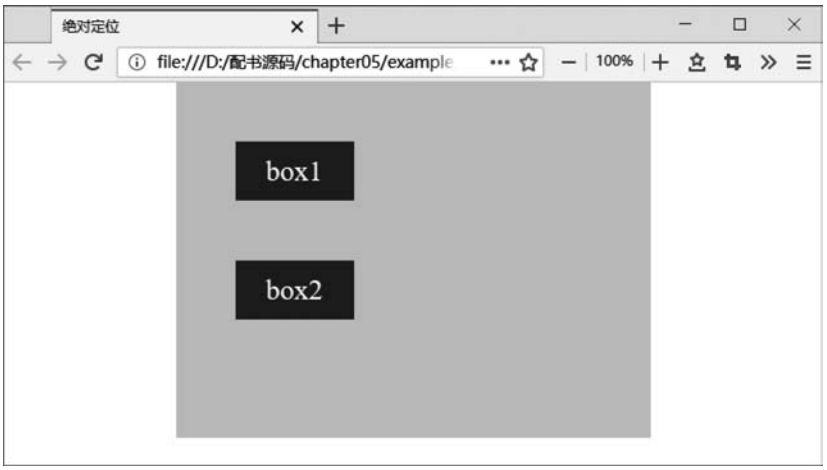

图 5-17　未设置定位属性

2. 设置绝对定位并设置边偏移量 TL

为 box1 设置绝对定位并设置边偏移量 TL(案例文件：chapter05\example\exp5_12_Step2.html)

```
#box1{
    position: absolute;
    top: 50px;
    left: 50px;
}
```

代码运行效果如图 5-18 所示。

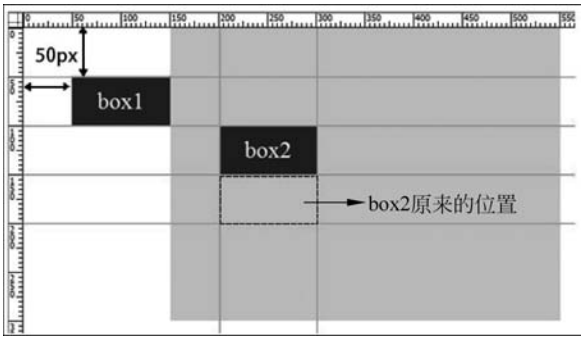

图 5-18　绝对定位原理

当没有给 box1 的祖先元素 father-box 设置定位属性时，box1 设置绝对定位，TL 相对于浏览器窗口进行定位和偏移，top 为相对于浏览器上边的距离，left 为相对于浏览器左边的距离。绝对定位使元素脱离文档流，不占据文档流空间。box1 脱离文档流，浮在页面上，后面的 box2 自动往上移。

改变浏览器的宽度，发现父盒子一直居中，子盒子 box1 一直相对于浏览器窗口进行定位，box1 相对于它的直接父元素的位置发生了变化，如图 5-19 所示。

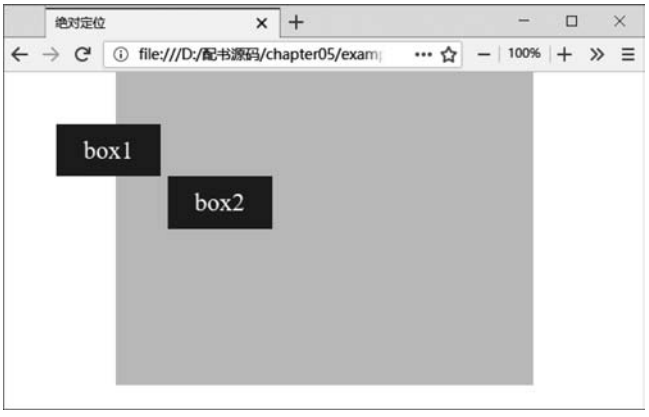

图 5-19　浏览器窗口较小效果

再次改变浏览器的宽度，发现父盒子一直居中，子盒子 box1 一直相对于浏览器窗口进行定位，box1 相对于它的直接父元素的位置发生了变化，如图 5-20 所示。

图 5-20　浏览器窗口变宽效果

3. 设置绝对定位并设置边偏移量 BR

为 box1 设置绝对定位并设置边偏移量 BR(案例文件:chapter05\example\exp5_12_Step3.html)

```
#box1{
    position: absolute;
    bottom: 50px;
    right: 50px;
}
```

代码运行效果如图 5-21 所示。

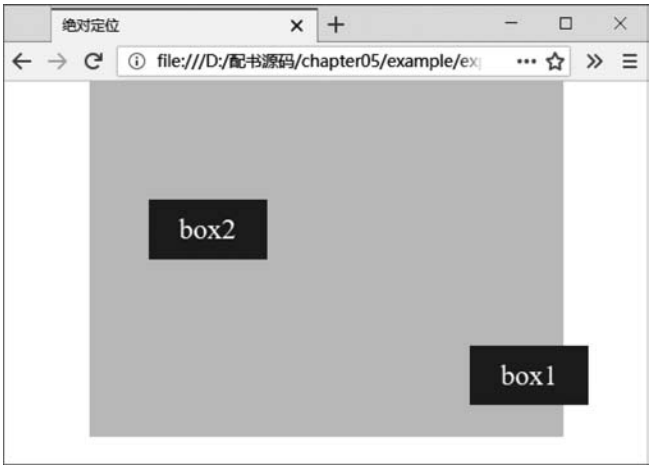

图 5-21 绝对定位边偏移量 BR

子盒子 box1 一直相对于浏览器窗口进行定位,box1 距浏览器窗口底部和右侧各 50px。

4. 父元素相对定位、子元素绝对定位(案例文件:chapter05\example\exp5_12_Step4.html)

从上面的例子中可以看到,改变浏览器的宽度,发现父盒子一直居中,子盒子 box1 一直相对于浏览器窗口进行定位,box1 相对于它的祖先元素的位置发生变化。若想随着浏览器窗口的变化,父盒子一直居中,子盒子一直相对于父盒子而绝对定位,则需要为父元素设置定位属性。

```
#father-box{
    position: relative;        /*父元素相对定位*/
}
#box1{
    position: absolute;        /*子元素绝对定位*/
    top: 150px;
    left: 150px;
}
```

代码运行效果如图 5-22 所示。

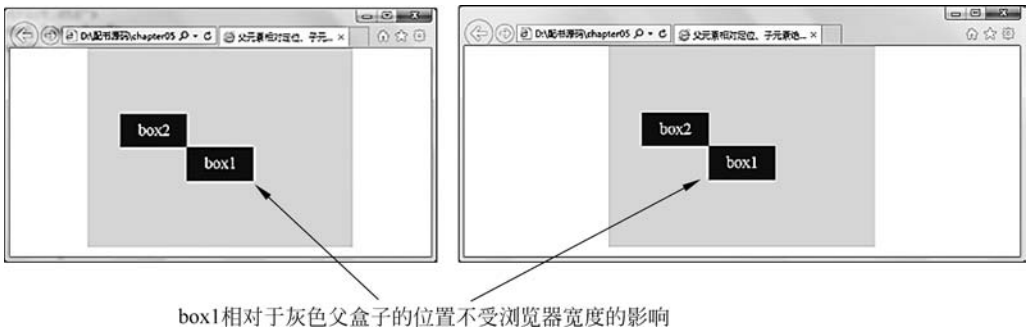

图 5-22　浏览器宽度变化，父子元素相对位置不变

绝对定位是将元素依据最近的已经定位(绝对、固定或相对定位)的祖先元素进行定位。box1 的父元素 father-box 设置定位属性后，box1 设置绝对定位，TL 相对于父元素 father-box 进行定位和偏移，top 为相对于 father-box 上边的距离，left 为相对于 father-box 左边的距离。无论浏览器窗口如何变化，子元素始终相对于其祖先元素进行定位和偏移，如图 5-23 所示。

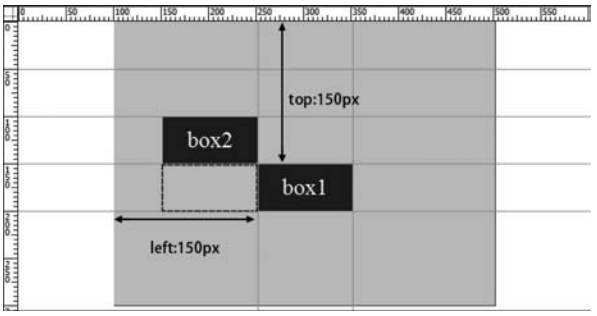

图 5-23　父元素相对定位、子元素绝对定位偏移量

5.5.5　固定定位

固定定位是绝对定位的一种特殊形式，它以浏览器窗口作为参照物来定义网页元素。当 position 属性的取值为 fixed 时，即将元素定位模式设置为固定定位。

当对元素设置固定定位后，它将脱离标准文档流的控制，始终依据浏览器窗口来定义自身的显示位置。不管浏览器滚动条如何滚动，不管浏览器窗口大小如何变化，该元素始终显示在浏览器窗口的固定位置。常见固定定位的应用有网页上返回顶端的按钮、网站侧边栏、顶部或者底部位置不变的导航、广告条等(案例文件：配书源码/fixed)。

【例 5-13】　固定定位的原理(案例文件：chapter05\example\exp5_13.html)

代码如下：

```
<!DOCTYPE html>
<html lang="en">
```

```
    <head>
        <meta charset = "UTF-8" >
        <title> 固定定位的原理 </title>
        <style>
            # page{width: 260px; }
            # go-top{
                width: 58px;
                height: 70px;
                position: fixed; / * 固定定位 * /
                top: 400px;
                left: 280px;
            }
        </style>
    </head>
    <body>
        <div id = "page" >
            < img src = "images/img01.jpg" alt = ""
            < img src = "images/img02.jpg" alt = "" >
            < img src = "images/img03.jpg" alt = "" >
            < img src = "images/img04.jpg" alt = "" >
            < img src = "images/img05.jpg" alt = "" >
            < img src = "images/img06.jpg" alt = "" >
        </div>
        <div id = "go-top" >
            <a href = " # page" > < img src = "images/goTop.png" alt = "" > </a>
        </div>
    </body>
</html>
```

在浏览器中运行,无论如何滚动滚动条,返回顶部的图片一直都在固定的位置,如图 5-24 所示。

5.5.6 z-index 层叠等级属性

当对多个元素同时设置定位时,定位元素之间有可能发生重叠。在 CSS 中,如果调整重叠定位元素的堆叠顺序,可以对定位元素应用 z-index 层叠等级属性,其取值可为正整数、负整数和 0。z-index 的默认属性值是 0,取值越大,定位元素在层叠元素中越居上。

【例 5-14】 z-index 的原理(案例文件:chapter05\example\exp5_14.html)

代码如下:

```
<!DOCTYPE html>
<html lang = "en" >
```

图 5-24　固定定位的“返回顶部”图片

```
<head>
    <meta charset = "UTF-8" >
    <title> z-index </title>
    <style type = "text/css" >
        * {padding: 0; margin: 0; }
        body{font-size: 24px; }
        #box1, #box2, #box3{
            width: 500px;
            border: 2px solid red;
            position: absolute;
        }
        p{
            height: 40px;
            text-align: center;
        }
        #box1{
            background-color: pink;
            top: 20px;
            left: 20px;
        }
        #box2{
            background-color: yellow;
            top: 60px;
            left: 60px;
```

```
            }
                #box3{
                background-color: green;
                top: 100px;
                left: 100px;
            }
        </style>
    </head>

    <body>
        <div id = "box1" >
            <p> box1 </p>
            <p> box1 </p>
        </div>
        <div id = "box2" >
            <p> box2 </p>
            <p> box2 </p>
        </div>
        <div id = "box3" >
            <p> box3 </p>
            <p> box3 </p>
        </div>
    </body>
</html>
```

例 5-14 的代码运行效果如图 5-25 所示。

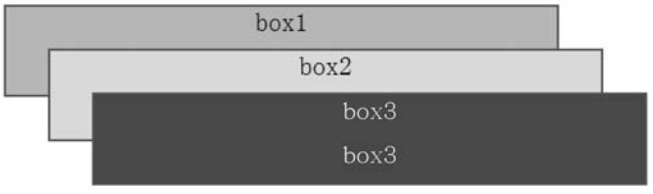

图 5-25　z-index 的原理应用

重新设置例 5-14 中 z-index 属性如下。

```
#box1{z-index: 3; }
#box2{z-index: 2; }
#box3{z-index: 1; }
```

代码运行效果如图 5-26 所示。

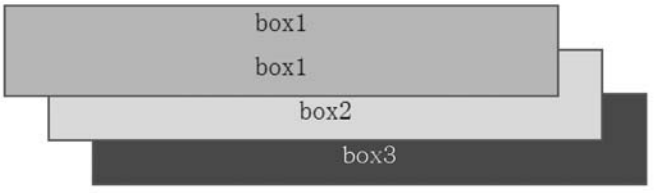

图 5-26　重新设置 z-index 属性后的效果

【项目实战】

项目 5-1 “国”字形布局网页效果制作(案例文件目录：chapter05\demo\demo5_1)

效果图

项目 5-1 的布局效果如图 5-27 所示。

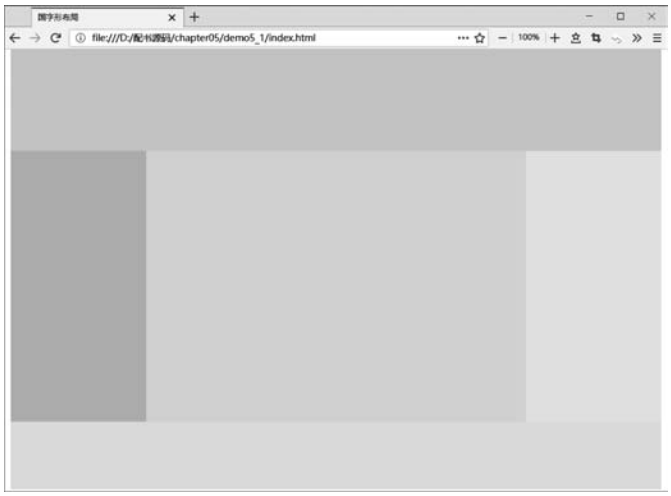

图 5-27 布局效果图

思路分析

“国”字形布局也称“同”字形布局，是许多大型网站中常见的一种网页布局类型。网页的上部常用来放置网站的标题、导航，以及 banner 横幅广告条；下部放置网页的基本信息、联系方式和版权声明等；中部放置网页主体内容，一般分成左、右两部分或者左、中、右三部分。“国”字形布局能充分利用版面组织信息，结构分明。中部分成三部分的“国”字形布局的 HTML 结构如图 5-28 所示。

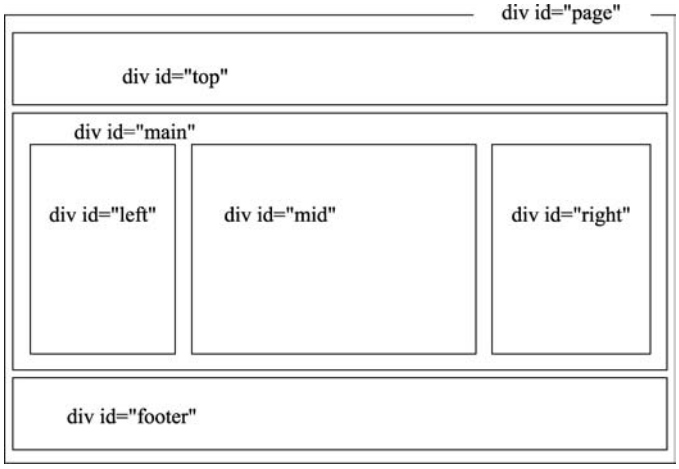

图 5-28 “国”字形布局及 HTML 结构

制作步骤

Step01. 根据布局及 HTML 结构分析搭建页面 HTML 结构,代码如下。

```
<!DOCTYPE html>
<html lang = "en" >
    <head>
        <meta charset = "UTF-8" >
        <title> 国字型布局 </title>
    </head>
    <body>
        <div id = "page" >
            <div id = "top" > </div>
            <div id = "main" >
                <div id = "left" > </div>
                <div id = "mid" > </div>
                <div id = "right" > </div>
            </div>
            <div id = "footer" > </div>
        </div>
    </body>
</html>
```

Step02. 设置 CSS。

为讲解方便,此处使用内联样式。

首先清除浏览器默认样式,再设置最外层 的宽度值以及居中。将 id 值为“page”的 层的宽度值设置为“960px”,将其 margin 值设置为“0 auto”,即上下为 0,左右为自动,实现 层居中。注意:必须设置 的宽度值,否则其宽度将为默认值 100%。 样式代码如下。

```
<style type = "text/css" >
     * {margin: 0; padding: 0; }              /* 清除浏览器默认样式 */
     #page{
         width: 960px;                       /* 设置 id 值为 #page 的 div 层居中 */
         margin: 0 auto;                     /* 设置 id 值为 #page 的 div 层居中 */
     }
</style>
```

Step03. 分别设置 #top 层、#main 层、#footer 层的宽度、高度及背景颜色,在样式代码中追加如下代码。

```
#top{
    width: 960px; /* 宽度设置可省略,默认会继承父级宽度 */
    height: 150px;
```

```
    background-color: #EFCEE8;
}
#main{
    height: 400px;
}
#footer{
    height: 100px;
    background-color: #FF0;
}
```

在浏览器中测试页面的显示效果如图 5-29 所示。可以看到上部 #top 层和底部 #footer 层分别以浅紫色和黄色填充，中间白色区域是未设置背景颜色的 #main 层。页面整体居中呈现上、中、下三段布局显示。

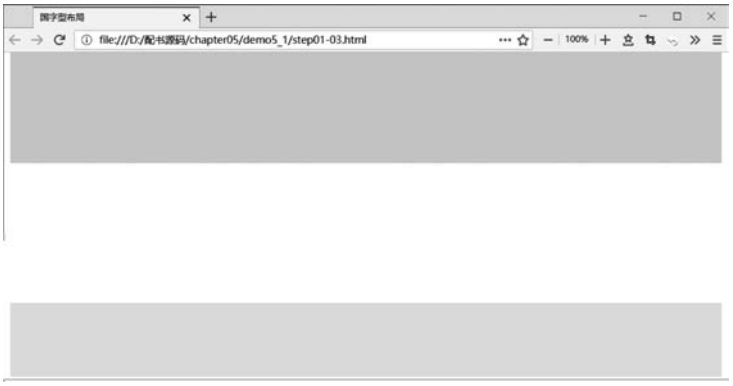

图 5-29　测试页面的显示效果

Step04. 中部 #main 层内部包括 #left 层、#mid 层和 #right 层，分别表示网页主体内容区的左、中、右三部分。首先设置三者共同具有的高度值和左浮动属性，然后分别设置用以区别三者的宽度值和背景色，样式代码如下。

```
#left, #mid, #right{
    height: 400px;
    float: left; /* 左浮动 */
}
#left{
    width: 200px;
    background-color: #E08031;
}
#mid{
    width: 560px;
    background-color: #DCF7A1;
}
#right{
    width: 200px;
    background-color: #E08031; /
}
```

Step05. 浏览器兼容性测试。

“国”字形布局案例网页已制作完毕,在多个不同的浏览器中测试,都可以正常呈现网页。

注意:中部#main层的高度值实际制作时通常不会设置,层高会依靠其内部的内容将图层撑开。另外,#main层内部的#left层、#mid层、#right层都设置了左浮动,使用浮动的同时要考虑清除浮动的操作,因为浮动会影响其他元素布局。清除浮动最为常用的一种方法是在其父元素中加入overflow: hidden,因此#main层的样式代码可以修改为如下所示。

```
#main{
    height: 400px;
    overflow: hidden;          /* 清除此父元素中的子元素浮动对页面的影响 */
}
```

项目5-2　商品广告版块效果制作(案例文件目录:chapter05\demo\demo5_2)

效果图

项目5-2运行效果如图5-30所示。

图5-30　商品广告版块效果

思路分析

从图5-30可以看出,商品广告版块由广告主体图片及优惠信息两部分组成,优惠信息位于半透明矩形背景之上,而且中间数字部分的文字为红色。此版块的布局及HTML结构如图5-31所示。

图5-31　布局及HTML结构

此案例可以运用相对定位和绝对定位的位置属性特点进行布局。注意:图5-31中主图可以使用<img>标签,也可以以 层的背景形式出现,后者能简化HTML结构。

制作步骤

Step01. 根据布局及 HTML 结构分析图，编写 HTML 结构代码如下。

```
<div id = "goods" >
    < img src = "images/logo. png" alt = ""/>
    <a href = " # " > 购物即享 < span> 8. 5 </span> 折优惠 </a>
</div>
```

Step02. 建立 CSS，为讲解方便，此处使用内联样式。

首先清除浏览器默认样式，然后设置最外层 div(# goods)的宽度、高度及背景图，同时设置 overflow 属性值为“hidden”，保证层中溢出的内容不显示，并且设置最外层 # goods 位置属性为相对定位，以更好地定位其内部元素的位置，样式代码如下。

```
* {margin: 0; padding: 0; } /* 清除浏览器默认样式 */
# goods{
    width: 570px;
    height: 273px;
    background-image: url(images/bg. jpg);
    overflow: hidden;          /* 溢出部分不显示 */
    position: relative;        /* 相对定位，# goods 是其内部元素的父元素 */
}
```

Step03. 设置 Logo 图的 CSS。

Logo 图片位于 < div> 层的右上角位置，由于父元素 # goods 的位置属性为相对定位了，因此只要设置 Logo 图片的 < img> 为绝对定位，就能通过坐标确定其位置。Logo 图片的 < img> 标签样式代码如下。

```
# goods img{
    position: absolute;     /* 子元素绝对定位 */
    top: 5px;
    right: 5px;
}
```

Step04. 设置超链接 < a> 的 CSS。

< a> 标签呈半透明黑色矩形状位于图层底部。首先将 < a> 标签转换为块级元素，设置其宽高值。半透明的背景色使用 rgba(红，绿，蓝，alpha)实现，透明度参数 alpha 是介于 0～1 的数字。同样，< a> 标签的定位采用基于父元素的绝对定位，再用坐标定位的方法。< a> 标签的样式代码如下。

```
# goods a{
    text-decoration: none; ;          /* 去除链接默认的下画线样式 */
    display: block;                   /* 将 < a> 标签转换为块级元素，设置其宽度、高度值 */
    width: 550px;
    height: 50px;
    background-color: rgba(0,0,0,.6); /* 背景色半透明 */
    color: # fff;                     /* 设置文字颜色，大小，行高，距左侧边距 */
```

```
        font-size: 18px;
        line-height: 50px;
        padding-left: 20px;
        position: absolute; /* 子元素绝对定位 */
        left: 0;
        bottom: 0px;
    }
```

Step05. 设置<a>标签中黄色文本的 CSS。

设置中的文本颜色为黄色,字号稍大一些,并且与周边文字稍有些间距,以突出文本,标签样式代码如下。

```
    #goods span{
        color: #FF0;
        font-size: 24px;
        padding: 5px;
    }
```

Step06. 兼容性测试。

商品广告版块效果制作完毕,在多个不同的浏览器中测试,都可以正常呈现网页。本案例在 Google Chrome、火狐、Opera、Safari for Windows 主流版本及 IE 11 中测试通过。

这类商品广告式的版块在网页中应用非常广泛,实际制作时常常会再增加一些动态过渡效果,例如底部<a>标签默认状态下不显示,当鼠标移动到 层块上时,<a>标签才逐渐显示,或是鼠标移动到层块时让层的背景图逐渐放大,同时<a>标签逐渐显示,鼠标移开则自动恢复原始状态。这种效果用 CSS3 的 transition 过渡属性较容易实现,例如,如果希望一开始<a>标签默认不出现,首先将<a>标签的样式中 bottom 的坐标值设置成"-50px",即将<a>标签移动到 层的底线下 50px 处,超出层大小位置外溢出内容不显示,就实现了隐藏效果。然后设置 bottom 坐标的过渡属性及过渡时间。最后设置鼠标经过#goods 层时,<a>标签的 bottom 属性为"0",就会出现<a>标签的动态过渡效果,其样式代码修改如下。

```
    #goods a{
        /* bottom: 0px; */
        bottom: -50px;                      /* 距底部-50px,即隐藏至图片区底线之外 */
        transition: bottom 1s;
        -moz-transition: bottom 1s;         /* 兼容 Firefox 4 */
        -webkit-transition: bottom 1s;      /* Safari 和 Chrome */
        -o-transition: bottom 1s;           /* Opera */
    }
    #goods: hover a{bottom: 0; }            /* 鼠标经过#goods 层,<a>标签底部坐标为 0 */
```

若需要制作 层背景图的动态过渡效果,则用到 background-position 属性及 CSS3 新增的 background-size 属性,这两种常用的动态过渡效果案例分别位于本章配书案例目录下的 index_transition1. html 和 index_transition1. html 文档,读者可以参照学习。

【实训作业】

实训任务 5-1　新浪微博发言版块制作

合理使用 HTML 标签、盒子模型、浮动、定位布局等 CSS 技术制作效果图，如图 5-32 所示。具体要求如下。

1. 网站目录结构明晰，有默认图像文件夹（图像文件夹命名为 images 或 img），图像文件命名规范。

2. 首页命名为 index 或 default，首页扩展名为 html 或 htm。

3. 建立外部样式表文件并连接到 index.html 文档。

4. 合理使用所学 HTML 标签组织网页结构，进行布局设计。

5. 合理使用 CSS 选择器，选择器命名规范，并设置 CSS，正确设置盒子模型、浮动、定位等 CSS。

图 5-32　新浪微博发言版块

实训任务 5-2　制作第 5 章案例作业网站

完成第 5 章案例作业网站的制作，具体要求如下。

1. 网站目录结构明晰，有默认图像文件夹（图像文件夹命名为 images 或 img）及样式文件夹（css）。

2. 网站根目录下有首页，首页命名为 index 或 default。

3. 根据章节设定二级频道目录，二级频道目录下要求具有各自的首页及默认图像文件夹。

4. 案例作业网站中包含第 5 章所有案例及项目任务。

5. 每个网页有完整的网页标题、广告条 banner、导航栏、主体内容、页脚和版权声明等信息。