第5章

CSS高级样式设计

5.1 filter 滤镜

5-1 filter 滤镜

CSS3 中的 filter 滤镜属性可以改变图片等元素的可视效果。使用方法：直接给需要设置的图片添加 filter 属性。各种 filter 的功能描述如表 5-1 所示。

filter 属性语法：

```
filter: none | blur() | brightness() | contrast() | drop-shadow() | grayscale() | hue-rotate()
| invert() | opacity() | saturate() | sepia() | url();
```

滤镜的许多属性值通常使用百分比表示，如 75%；也可以使用小数来表示，如 0.75。

表 5-1　各种 filter 的功能描述

filter	描　　述
none	默认值，没有效果
blur(px)	给图像设置高斯模糊。“radius”一值设定高斯函数的标准差，或者是屏幕上以多少像素融在一起，所以值越大越模糊；如果没有设定值，则默认是 0；这个参数可设置 CSS 长度值，但不接受百分比值
brightness(%)	给图片应用一种线性乘法，使其看起来更亮或更暗。如果值是 0，图像会全黑；值是 100%，则图像无变化。其他的值对应线性乘数效果。值超过 100%也是可以的，图像会比原来更亮。如果没有设定值，默认是 100%
contrast(%)	调整图像的对比度。值是 0 时，图像会全黑；值是 100%时，图像不变。值可以超过 100%，意味着会运用更低的对比。若没有设置值，默认是 100%
drop-shadow(h-shadow v-shadow blur spread color)	给图像设置一个阴影效果。阴影是合成在图像下面，可以有模糊度的，可以以特定颜色画出遮罩图的偏移版本。函数接受< shadow >(在 CSS3 背景中定义)类型的值，除了 inset 关键字是不允许的。该函数与已有的 box-shadow 属性很相似；不同之处在于，通过滤镜，一些浏览器为了更好的性能会提供硬件加速。 < shadow >参数如下： < offset-x >、< offset-y >(必需)，设置阴影偏移量的两个 length 值。< offset-x > 设定水平方向距离，负值会使阴影出现在元素左边。< offset-y >设定垂直方向距离，负值会使阴影出现在元素上方。如果两个值都是 0，则阴影出现在元素正后面。如果设置了< blur-radius > 和/或 < spread-radius >，会有模糊效果。

续表

filter	描　　述
drop-shadow (h-shadow v-shadow blur spread color)	<blur-radius>(可选) 这是第三个 length 值,值越大,越模糊,则阴影会变得更大更淡。不允许为负值,若未设定,默认是 0(则阴影的边界很锐利)。 <spread-radius>(可选) 这是第四个 length 值,正值会使阴影扩张和变大,负值会使阴影缩小,若未设定,默认是 0(阴影会与元素一样大小)。 注意:WebKit 以及一些其他浏览器不支持第四个长度,即使加了也不会渲染。 <color>(可选) 若未设定,颜色值基于浏览器。在 Gecko(Firefox),Presto(Opera)和 Trident(Internet Explorer)中,会应用 color 属性的值。另外,如果颜色值省略,WebKit 中阴影是透明的
grayscale(%)	将图像转换为灰度图像。其值定义转换的比例。值为 100%,则完全转为灰度图像;值为 0,图像无变化;值为 0~100%,则是效果的线性乘子。若未设置,值默认是 0
hue-rotate(deg)	给图像应用色相旋转。angle 一值设定图像会被调整的色环角度值。值为 0deg,则图像无变化。若值未设置,默认值是 0deg。该值虽然没有最大值,但若超过 360deg 的值相当于又绕一圈
invert(%)	反转输入图像。值定义转换的比例,100%是完全反转,0 则图像无变化。值为 0~100%,则是效果的线性乘子。若值未设置,默认是 0
opacity(%)	转换图像的透明程度。值定义转换的比例,0%是完全透明,100%则图像无变化。值为 0~100%,则是效果的线性乘子,也相当于图像样本乘以数量。若值未设置,默认是 100%。该函数与已有的 opacity 属性很相似,不同之处在于通过 filter,一些浏览器为了提升性能会提供硬件加速
saturate(%)	转换图像饱和度。值定义转换的比例,0 是完全不饱和,100%则图像无变化。其他值则是效果的线性乘子。超过 100%的值是允许的,则有更高的饱和度。若值未设置,默认是 100%
sepia(%)	将图像转换为深褐色。定义转换的比例,100%为完全是深褐色的,0 为图像无变化。0~100%则是效果的线性乘子。若未设置,默认是 0
url()	URL 函数接受一个 XML 文件,该文件设置了一个 SVG 滤镜,且可以包含一个锚点来指定一个具体的滤镜元素。 例如:filter:url(svg-url#element-id)

【例 5-1】 滤镜样式实例。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Drop Caps with CSS</title>
  <style>
    .blur {
      /* 图片使用高斯模糊效果 */
      -webkit-filter: blur(5.5px);
      /* Chrome, Safari, Opera */
      filter: blur(5.5px);
```

```
}
.grayscale {
  /* 将图像转换为灰度图像 */
  -webkit-filter: grayscale(100%);
  /* Chrome, Safari, Opera */
  filter: grayscale(100%);
}
.brightness {
  /* 使图片变亮 */
  -webkit-filter: brightness(200%);
  /* Chrome, Safari, Opera */
  filter: brightness(200%);
}
.contrast {
  /* 调整图像的对比度 */
  -webkit-filter: contrast(300%);
  /* Chrome, Safari, Opera */
  filter: contrast(300%);
}
.drop-shadow {
  /* 给图像设置一个阴影效果 */
  -webkit-filter: drop-shadow(180px 180px 10px #10beb6);
  /* Chrome, Safari, Opera */
  filter: drop-shadow(-180px 180px 10px red);
}
.hue-rotate {
  /* 给图像应用色相旋转 */
  -webkit-filter: hue-rotate(225deg);
  /* Chrome, Safari, Opera */
  filter: hue-rotate(225deg);
}
.invert {
  /* 反转输入图像 */
  -webkit-filter: invert(100%);
  /* Chrome, Safari, Opera */
  filter: invert(100%);
}
.opacity {
  /* 转换图像的透明程度 */
  -webkit-filter: opacity(30%);
  /* Chrome, Safari, Opera */
  filter: opacity(30%);
}
.saturate {
  /* 转换图像饱和度 */
  -webkit-filter: saturate(800%);
  /* Chrome, Safari, Opera */
  filter: saturate(800%);
```

```
    }
    .sepia {
      /*将图像转换为深褐色*/
      -webkit-filter: sepia(100%);
      /* Chrome, Safari, Opera */
      filter: sepia(100%);
    }
    .many {
      /*综合多种*/
      -webkit-filter: contrast(200%) brightness(150%);
      /* Chrome, Safari, Opera */
      filter: contrast(200%) brightness(150%);
    }
  </style>
</head>
<body>
  <label for="raw">raw</label>
  <img src="fig1.jpg" id="raw" />
  <label for="blur">blur</label>
  <img src="fig1.jpg" id="blur" class="blur" />
  <br>
  <label for="grayscale">grayscale</label>
  <img src="fig1.jpg" id="grayscale" class="grayscale" />
  <label for="brightness">brightness</label>
  <img src="fig1.jpg" id="brightness" class="brightness" />
  <br>
  <label for="contrast">contrast</label>
  <img src="fig1.jpg" id="contrast" class="contrast" />
  <label for="drop-shadow">drop-shadow</label>
  <img src="fig1.jpg" id="drop-shadow" class="drop-shadow" />
  <br>
  <label for="hue-rotate">hue-rotate</label>
  <img src="fig1.jpg" id="hue-rotate" class="hue-rotate" />
  <label for="invert">invert</label>
  <img src="fig1.jpg" id="invert" class="invert" />
  <br>
  <label for="opacity">opacity</label>
  <img src="fig1.jpg" id="opacity" class="opacity" />
  <label for="saturate">saturate</label>
  <img src="fig1.jpg" id="saturate" class="saturate" />
  <br>
  <label for="sepia">sepia</label>
  <img src="fig1.jpg" id="sepia" class="sepia" />
  <label for="many">many</label>
  <img src="fig1.jpg" id="many" class="many" />
</body>
</html>
```

程序运行效果如图 5-1 所示。

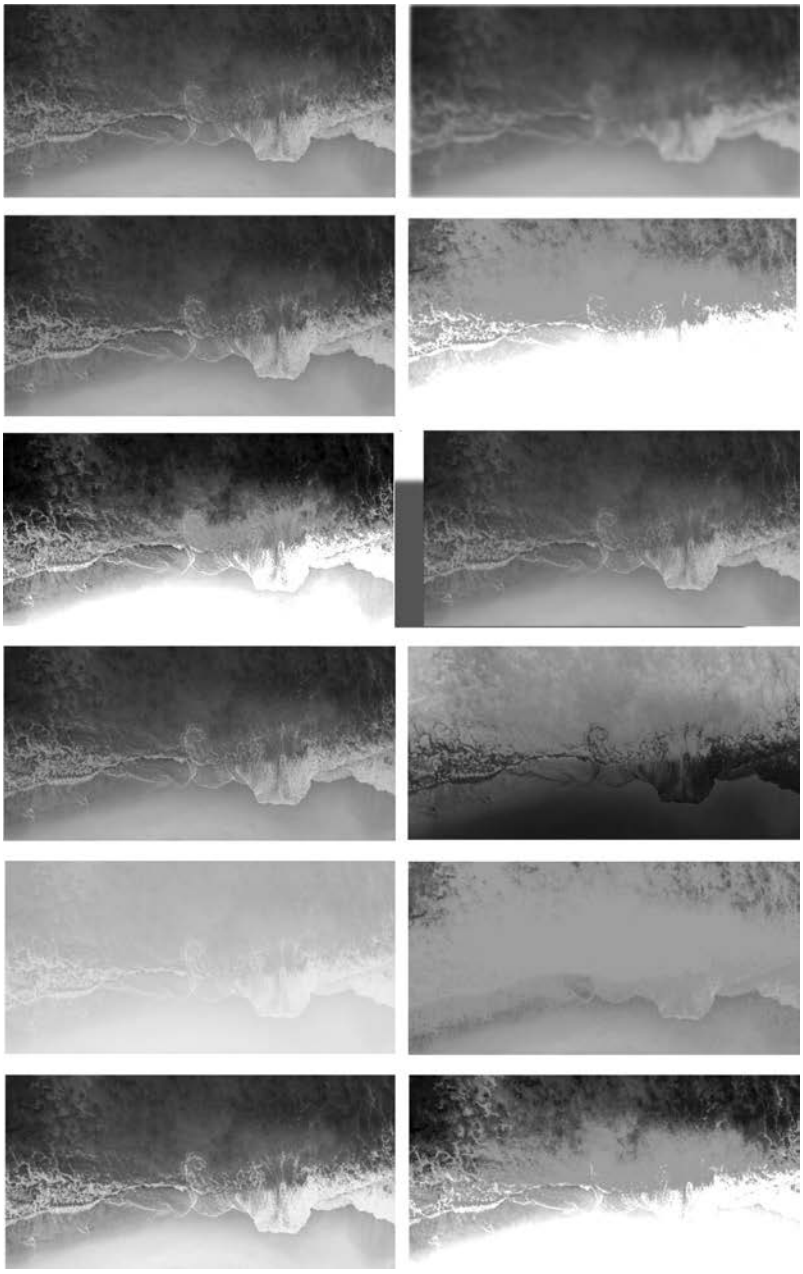

图 5-1　第 1 幅为原图其余分别为 blur、grayscale、brightness、contrast、drop-shadow、hue-rotate、invert、opacity、saturate、sepia 和混合滤镜处理后的效果图

5.2　CSS3 的变形

5-2 transition 过渡

CSS3 的变形能实现移动、旋转、缩放和倾斜等。

5.2.1　translate 平移

transform 的 translate()方法用来实现元素的平移效果。

```
transform: translateX(x);          /* 沿 X 轴方向平移 */
transform: translateY(y);          /* 沿 Y 轴方向平移 */
transform: translate(x, y);        /* 沿 X 轴和 Y 轴方向同时平移 */
```

有 3 种平移：translateX()、translateY()、translate()。参数 x 表示元素在 X 轴方向上的移动距离，参数 y 表示元素在 Y 轴方向上的移动距离，两者的单位可以为 px、em 和百分比等。

【例 5-2】 transform：translate 平移实例。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title></title>
<style type="text/css">
        /* 设置原始元素样式 */
        #before{
            width:200px;
            height:200px;
            border:2px dashed rgb(128, 128, 128);
        }
        /* 设置当前元素样式 */
        #after{
            width:200px;
            height:200px;
            color:white;
            background-color: rgb(30, 250, 129);
            opacity: 0.6;
            transform:translateX(20px);
        }
</style>
</head>
<body>
<div id="before">原来
<div id="after">平移</div>
</div>
</body>
</html>
```

程序运行效果如图 5-2 所示。

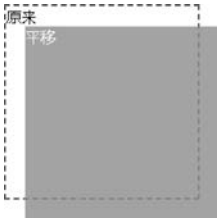

图 5-2 元素平移效果

在实际开发中，单纯对某个元素定义平移是没有太多意义的，变形效果一般都是结合 CSS3 动画一起使用。

5.2.2 scale 缩放

在 CSS3 中可以使用 transform 属性的 scale()方法来实现元素的缩放效果。

【语法】

```
transform: scaleX(x);              /* 沿 X 轴方向缩放 */
transform: scaleY(y);              /* 沿 Y 轴方向缩放 */
transform: scale(x, y);            /* 沿 X 轴和 Y 轴方向同时缩放 */
```

跟 translate()方法类似，缩放也有 3 种情况：scaleX()、scaleY()、scale()。参数 x 表示元

素在 X 轴方向的缩放倍数，参数 y 表示元素在 Y 轴方向的缩放倍数。

【例 5-3】 transform：scale 元素缩放实例。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title></title>
<style type="text/css">
        /*设置原始元素样式*/
        #origin {
            position:absolute;
            top:30px;
            left:20%;
            width:200px;
            height:200px;
            border:2px solid red;
        }
        /*设置当前元素样式*/
        #current {
            width:200px;
            height:200px;
            color:rgb(255, 255, 255);
            background-color: rgb(30, 170, 250);
            opacity: 0.4;
            transform:scaleX(1.5);
        }
</style>
</head>
<body>
<div id="origin">原来
        <div id="current">缩放</div>
    </div>
</body>
</html>
```

程序运行效果如图 5-3 所示。

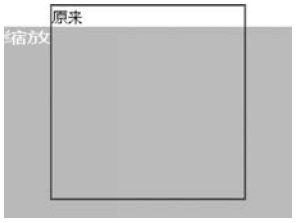

图 5-3 元素缩放效果

5.2.3 rotate 旋转

使用 transform 属性的 rotate()方法来实现元素的旋转效果。

【语法】

```
transform: rotate(angle);
```

参数 angle 表示元素相对于中心原点旋转的度数(deg)。如果度数为正，则表示顺时针旋转；如果度数为负，则表示逆时针旋转。如果没有写单位，如 transform:rotate(50)不会变化。

【例 5-4】 transform:rotate()的元素旋转实例。

```
<!DOCTYPE html>
<html>
<head>
```

```
<meta charset="UTF-8" />
<title></title>
<style type="text/css">
        /*设置原始元素样式*/
        #origin{ position:absolute;
            top:30px;
            left:20%;
            width:200px;
            height:200px;
            border:2px solid red;
        }
        /*设置当前元素样式*/
        #current{
            width:200px;
            height:200px;
            color:rgb(255, 255, 255);
            background-color: rgb(30, 170, 250);
            opacity: 0.4;
            transform:rotate(50deg);
        }
</style>
</head>
<body>
<div id="origin">原来
<div id="current">旋转</div>
</div>
</body>
</html>
```

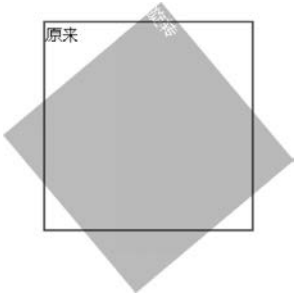

图 5-4　元素旋转效果

程序运行效果如图 5-4 所示。

5.2.4　skew 斜拉

transform 的 skew()方法能实现元素的倾斜效果。

语法：

```
transform: skewX(x);                /*沿 X 轴方向倾斜*/
transform: skewY(y);                /*沿 Y 轴方向倾斜*/
transform: skew(x, y);              /*沿 X 轴和 Y 轴方向同时倾斜*/
```

倾斜也有 3 种情况：skewX()、skewY()、skew()。参数 x 表示元素在 X 轴方向的倾斜度数，单位为 deg。如果度数为正，则表示元素沿 X 轴方向逆时针倾斜；如果度数为负，则表示元素沿 X 轴方向顺时针倾斜。

参数 y 表示元素在 Y 轴方向的倾斜度数，单位为 deg。如果度数为正，则表示元素沿 Y 轴方向顺时针倾斜；如果度数为负，则表示元素沿 Y 轴方向逆时针倾斜。

【例 5-5】 transform：skew()的元素斜拉实例。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
```

```
<title></title>
<style type="text/css">
/*设置原始元素样式*/
        .origin{ position:absolute;
            top:30px; left:20px;
            width:200px;
            height:200px;
            border:2px solid red;

        }
/*设置当前元素样式*/
        .skewX{
             position:absolute;
            top:300px; left:20px;
            width:200px;
            height:200px;
            color:rgb(14, 13, 13);
            background-color: rgb(217, 250, 30);
            opacity: 0.4;
            transform:skewX(30deg);
        }
        .skewY{
            position:absolute;
            top:300px; left:320px;
            width:200px;
            height:200px;
            color:rgb(10, 10, 10);
            background-color: rgb(30, 250, 239);
            opacity: 0.4;
            transform:skewY(40deg);
        }
        .skewXY{
            position:absolute;
            top:300px; left:620px;
            width:200px;
            height:200px;
            color:rgb(20, 18, 18);
            background-color: rgb(250, 30, 48);
            opacity: 0.4;
            transform:skew(50deg,50deg);
        }
</style>
</head>
<body>
<div class="origin">原来</div>
<div class="skewX">skewX</div>
<div class="skewY">skewY</div>
<div class="skewXY">skewXY</div>
</body>
</html>
```

程序运行效果如图 5-5 所示。

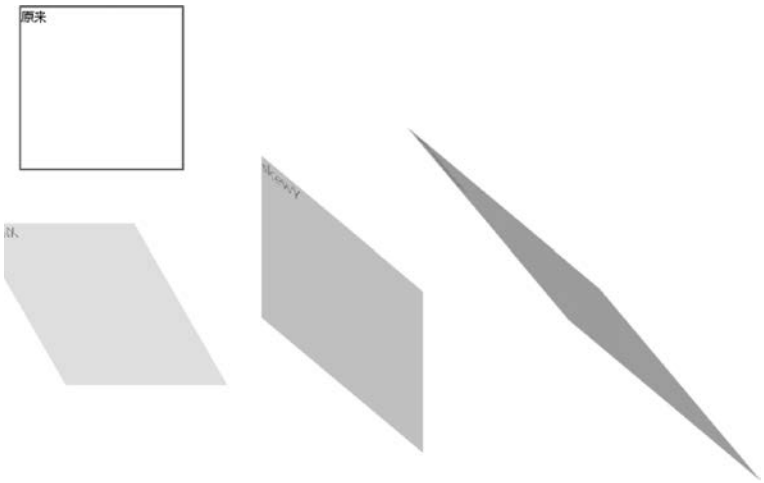

图 5-5　元素斜拉效果

5-3 animation

5.3　transition 过渡

transition 将元素的某一个属性从“一个值”在指定的时间内平滑地过渡到“另一个值”，从而实现动画效果，例如，渐渐显示、渐渐隐藏、动画快慢。

【语法】

```
transition: transition - property transition - duration transition - timing - funciton transition -
delay;
```

其中，transition-property：none|all|property。

none：没有属性会获得过渡效果。

all：所有属性都将获得过渡效果。

property：定义应用过渡效果的 CSS 属性名称列表，也就是 transition 属性可以同时定义两组或两组以上的过渡效果，组与组之间用逗号分隔。具有过渡的 CSS 属性主要有以下几种。

颜色：color，background-color，border-bottom-color，border-top-color，border-left-color，border-right-color，outline-color 等属性。

宽度：width，max-width，border-top-width，border-bottom-width，border-left-width，border-right-width，min-width 等属性。

高度：height，max-height，line-height 等属性。

位置：right，left，bottom，top，background-position 等属性。

边框：padding，padding-left，padding-right，padding-top，padding-bottom，margin，margin-left，margin-right，margin-top，margin-bottom 等属性。

大小：font-size，font-weight，boder-spacing，letter-spacing，text-indent 等属性。

透明度：opacity，visibility，text-shadow 等属性。

transition-duration 属性需要始终设置，否则时长为 0，就不会产生过渡效果。

transition-delay：time；在过渡效果开始前等待时间，单位为 s 或 ms。

transition-timing-function 属性用来描述这个中间值是怎样计算的。通过这个函数建立一条加速度曲线，因此在整个 transition 变化过程中，变化速度可以不断改变。常见的函数如下。

```
transition-timing-function: ease /* 规定慢速开始，然后变快，然后慢速结束的过渡效果(cubic-bezier(0.25,0.1,0.25,1)) */
transition-timing-function: ease-in /* 规定以慢速开始的过渡效果(等于 cubic-bezier(0.42,0,1,1)) */
transition-timing-function: ease-out/* 规定以慢速结束的过渡效果(等于 cubic-bezier(0,0,0.58,1)) */
transition-timing-function: ease-in-out/* 规定以慢速开始和结束的过渡效果(等于 cubic-bezier(0.42,0,0.58,1)) */
transition-timing-function: linear/* 规定以相同速度开始至结束的过渡效果(等于 cubic-bezier(0,0,1,1)) */
transition-timing-function: cubic-bezier(0.1, 0.7, 1.0, 0.1)
```

/* cubic-bezier(n,n,n,n)在 cubic-bezier 函数中定义自己的值。可能的值是 0～1 的数值 */。例如：

```
transition-timing-function: step-start
transition-timing-function: step-end
transition-timing-function: steps(4, end)
transition-timing-function: ease, step-start, cubic-bezier(0.1, 0.7, 1.0, 0.1)
```

cubic-bezier() 函数定义了一个贝塞尔曲线(Cubic Bezier)。贝塞尔曲线由四个点 P_0，P_1，P_2 和 P_3 定义。cubic-bezier 参数范围如图 5-6 所示，P_0 和 P_3 是曲线的起点和终点。P_0 是(0,0)并且表示初始时间和初始状态，P_3 是(1,1)并且表示最终时间和最终状态。P_0 默认值(0,0)，P_1 动态取值(x1,y1)，P_2 动态取值(x2, y2)，P_3 默认值(1, 1)，如图 5-7 所示。需要关注的是 P_1 和 P_2 两点的取值，X 轴的取值范围是 0～1，当取值超出范围时 cubic-bezier 将失效；Y 轴的取值没有规定，当然也无须过大。

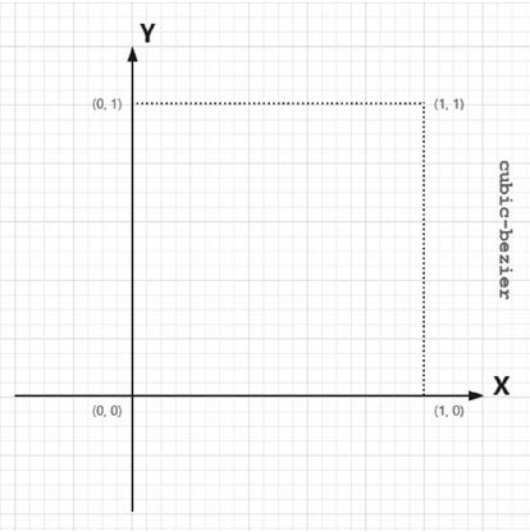

图 5-6 贝塞尔曲线参数范围

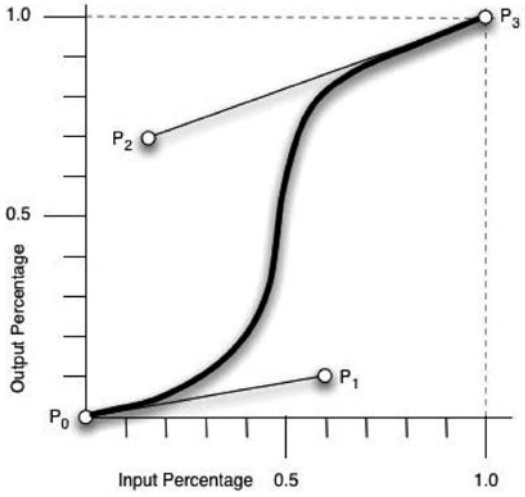

图 5-7 贝塞尔曲线

最直接的理解是,将以一条直线放在范围只有 1 的坐标轴中,并从中间拿出两个点来拉扯(X 轴的取值区间是[0,1],Y 轴任意),最后形成的曲线就是动画的速度曲线。

【例 5-6】 transition 的元素属性过渡实例。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>过渡,动画效果</title>
<style>
div
{width:100px;
    height:100px;
    background-image:url(images/fall.jpg);
transition-timing-function: cubic-bezier(0.1, 0.7, 1.0, 0.1);
    transition-property: width height;
    transition-duration: 8s;
}
div:hover
{
    width:1200px;
    height:1000px;
}
</style>
</head>
<body>
<p><b>注意:</b>该属性不兼容 IE9 以及更早版本的浏览器。</p>
<div></div>
<p>鼠标移动在块上查看动画效果。</p>
</body>
</html>
```

图 5-8 过渡的初始状态

程序运行效果如图 5-8 和图 5-9 所示。

图 5-9 过渡过程的中间状态

5.4 animation 动画

动画属性可以逐渐地从一个值变化到另一个值，如尺寸大小、数量、百分比和颜色。CSS属性是可以有动画效果的，这意味着它们可以用于动画和过渡。

5.4.1 animation 简介

transition 过渡是通过初始和结束两个状态之间的平滑过渡实现动画的。animation 则是通过关键帧@keyframes 来实现更为复杂的动画效果。

对于 transition 属性来说，它只能将元素的某一个属性从一个属性值过渡到另一个属性值，只能实现一次性的动画效果。而对于 animation 属性来说，它可以将元素的某一个属性从第 1 个属性值过渡到第 2 个属性值，然后还可以继续过渡到第 3 个属性值，以此类推，可以实现连续性的动画效果。

animation 也是一个复合属性，包括 8 个子属性，主要使用前 6 个子属性。

【语法】

```
animation: animation - name animation - duration animation - timing - function animation - delay
animation - iteration - count animation - direction animation - play - state animation - fill -
mode;
```

其中，以下几个属性与 transition 过渡的属性类似。

animation-duration 属性取值是一个时间，单位为 s，可以是小数。

animation-timing-function 属性取值共有 5 种，这个跟 CSS3 过渡的 transition-timing-function 是一样的。

animation-delay 属性用来定义动画的延迟时间，单位为 s，可以为小数，其中默认值为 0s。当没有定义 animation-delay 时，动画就没有延迟时间。例如，animation-delay:2s;表示当页面打开后，动画需要延迟 2 秒才会开始执行。

animation-iteration-count 属性取值有两种：一种是“正整数”，另外一种是“infinite”。当取值是 n(正整数)时，表示动画播放 n 次；当取值为 infinite 时，表示动画播放无数次，也就是循环播放。

5.4.2 animation-name 动画名称@keyframes

关键帧的语法是以@keyframes 开头，后面紧跟着动画名称 animation-name。from 等同于 0，to 等同于 100%。百分比跟随的花括号里面的代码，代表此时对应的样式。keyframes 属性如表 5-2 所示。

【语法】

```
@keyframes animationname {
keyframes - selector {
css - styles;
}
}
```

表 5-2 keyframes 的属性

值 属 性	描 述
animationname	必需,定义动画的名称
keyframes-selector	必需,动画时长的百分比。合法的值:0～100%,from(与 0%相同),to(与 100%相同)
css-styles	必需,一个或多个合法的 CSS 样式属性

例如,颜色变化帧:

```
@keyframes mycolor {
            0{background-color:red;}
            30%{background-color:blue;}
            60%{background-color:yellow;}
            100%{background-color:green;}
}
```

animation-name 调用的动画名需要和@keyframes 规则定义的动画名完全一致(区分大小写),如果不一致将不会产生任何动画效果。

【例 5-7】 animation 的动画实例。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>css 实现动画</title>
<style>
div{
    background-image: url("fig1.png");
    width:300px;
    height: 300px;
    position:relative;
    animation:mymove 5s infinite;
    -webkit-animation:mymove 5s infinite; /*Safari and Chrome*/
}
@keyframes mymove{
    from {left:0px;}
    30%{left:30px;}
    60%{left:90px;}
    to {left:200px;}
}
</style>
</head>
<body>
<div>动起来</div>
</body>
</html>
```

上面的动画通过 animation:mymove 5s infinite 语句实现,其中,mymove 为动画名称,5s 为持续时间,infinite 为动画次数不限制。具体效果为:前 30%的时间从 0 移动到 30 个像素,30%～60%的时间为 30～90 像素,60%～100%的时间为 90～200 个像素,因此移动变化的速

度不一样。程序运行效果如图 5-10 和图 5-11 所示。

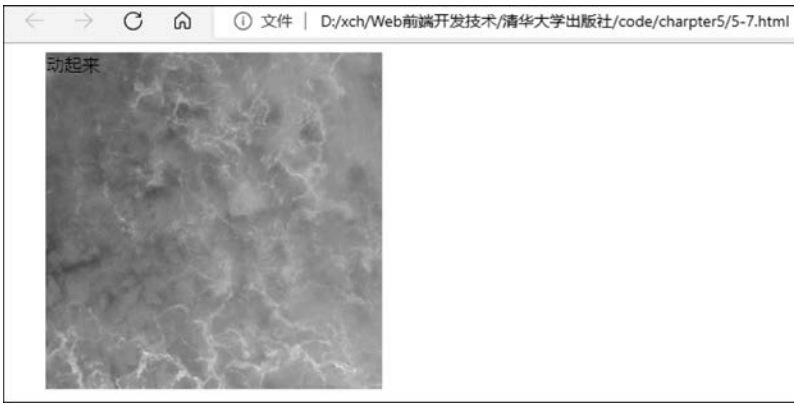

图 5-10　前 30%的某帧

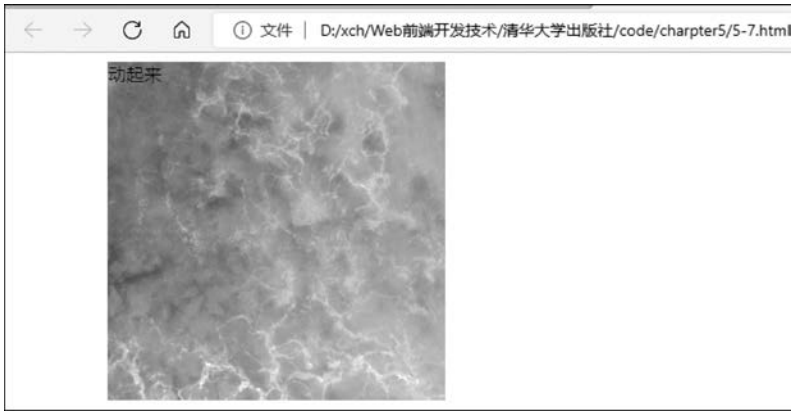

图 5-11　30%～60%的某帧

【例 5-8】　animation 同时有两个属性变换的动画实例。

```
<!DOCTYPEhtml>
<html>
<head>
    <title>Animation 同时改变多个属性的变化</title>
    <style>
        div{
        width:150px;
        height:50px;
        background:#4cff00;
        border:3pxsolid#000000;
        position:absolute;
        left:200px;
        top:300px;
        }
        /*定义一个关键帧*/
        @keyframes xiaolei {
          0 { background-color:#4cff00;
                transform: scale(0);
```

```
            }
        40% {
                background-color:#f00;
                transform: scale(1.5);
                }
        70% {
                background-color:#ffd800;
                transform: scale(2.5);
        }
            100% {
                transform: scale(3.5);
                background-color:#4cff00;
            }
        }
        div:hover{
        animation-name:xiaolei;
        animation-duration:8s;
        animation-timing-function: ease-in;
        }
    </style>
</head>
<body>
    <div>鼠标悬停,开始动画</div>
</body>
</html>
```

初始状态如图 5-12 所示，动画过程有缩放，同时颜色也在变化，如图 5-13 所示。

图 5-12　动画的初始状态

图 5-13　动画的变化过程状态

5.4.3　animation-direction 和 animation-play-state

animation-direction 属性，检索或设置对象动画在循环中是否按照方向运动，常见的播放方向取值见表 5-3。

表 5-3　animation-direction 的取值

属　性　值	描　　述
normal	正方向运动
reverse	反方向运动
alternate	动画先正方向运动再反方向运动，并持续交替运行
alternate-reverse	动画先反方向运动再正方向运动，并持续交替运行
initial	默认值
inherit	继承父元素的属性值

animation-play-state 属性用来定义动画的播放状态。有两个取值：running 为播放(默认值)，paused 为暂停。

【例 5-9】 animation 的播放方向和暂停播放实例。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title></title>
<style type="text/css">
        @keyframes mytranslate{
            0{}
            50%{transform:translateX(160px);}
            100%{}
        }
        #ball{
            width:40px;
            height:40px;
            border-radius:20px;
            background-color:red;
            animation-name:mytranslate;
            animation-timing-function:linear;
            animation-duration:2s;
            animation-iteration-count:infinite;
        }
        #container {
            display:inline-block;
            width:200px;
            border:1px solid silver;
        }
</style>
<script>
        window.onload = function(){
            var oBall = document.getElementById("ball");
            var oBtnPause = document.getElementById("btn_pause");
            var oBtnRun = document.getElementById("btn_run");
            //暂停
            oBtnPause.onclick = function(){
                oBall.style.animationPlayState = "paused";
            };
            //播放
            oBtnRun.onclick = function(){
                oBall.style.animationPlayState = "running";
            };
        }
</script>
</head>
<body>
<div id="container">
<div id="ball"></div>
</div>
<div>
<input id="btn_pause" type="button" value="暂停" />
<input id="btn_run" type="button" value="播放" />
```

```
</div>
</body>
</html>
```

图 5-14　动画播放状态

红球在矩形框内来回移动，单击“暂停”按钮，小球则停止移动；单击“播放”按钮，小球继续移动，效果如图 5-14 所示。

5.5　CSS 页面布局

5.5.1　display 属性

根据 CSS 规范的规定，每一个网页元素都有一个 display 属性，用于确定该元素的类型，每一个元素都有默认的 display 属性值，最常用的显示方式有 block，inline，inline-block。常见元素的默认显示方式如表 5-4 所示，但是可以通过指定其显示方式进行修改。

表 5-4　常见 CSS 元素的默认显示方式

显示方式	默 认 元 素
行内	a，abbr，acronym，b，bdo，big，br，cite，code，dfn，em，font，i，img，input，kbd，label，q，s，samp，select，small，span，strike，strong，sub，sup，textarea，tt，u ，var
块级	header，article，aside，figure，canvas，video，audio，footer，address，blockquote，dir，div，dl，fieldset，form，h1，h2，h3，h4，h5，h6，hr，isindex，menu，noframes，noscript，ol，p，pre，table，ul
可变元素	applet，button，del，iframe，ins，map，object，script

其中 code 元素是计算机代码，在引用源码时使用，dfn 元素为定义字段，kbd 元素为定义键盘文本，samp 元素为定义范例的计算机代码，tt 元素为电传文本，dir 元素为目录列表，isindex 元素为输入提示，menu 为菜单列表。noframes 元素为 frames 的可选内容，对于不支持 frame 的浏览器显示此区块内容。noscript 是可选的脚本内容，对于不支持 script 的浏览器显示此内容。

(1) 块级元素(block)：独占一行，对宽高的属性值生效；如果不给定宽度，块级元素就默认为浏览器的宽度，即 100%宽。

【例 5-10】　block 块级元素样式实例。

```
<!DOCTYPE html>
<HTML>
<head>
<style type="text/CSS">
    div {
    border-width: 4px;
    border-color: red;
    border-style: double;
    }
    #ID1 {
    background-color:rebeccapurple;
    color:#FFF;
    height: 200px;
```

```
    width: 200px;
    }
    #ID2 {
    font: normal 14px/1.5 Verdana, sans-serif;
    height: 200px;
    width: 200px;
    background-color: rosybrown;
    }
</style>
</head>
<body>
<div>
<div id="ID1">ID1</div>
<div id="ID2">ID2</div>
        box1
</div>
</body>
</HTML>
```

程序运行效果如图 5-15 所示。

图 5-15 块级元素的显示方式

(2) 行内元素(inline)：也就是内联，可以多个标签存在一行，完全靠内容撑开宽高，任何的属性调高度都无效。因此，对行内元素设置高度、宽度、内外边距等属性，都是无效的。

【例 5-11】 inline 显示方式实例。

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>CSS display:inline 实现 DIV 并排</title>
<style>
.bbb1,.bbb2,.bbb3,.bbb4{
      border-width: 2px ;
      border-color: red;
      border-style: solid;
      width: 200px;
      height: 200px;
```

```
    display:inline}
/* 4 个 CSS 简写:共用 display:inline 样式 */
</style>
</head>
<body>
<div class="bbb1">我在 bbb1 内</div>
<div class="bbb2">我在 bbb2 内</div>
<div class="bbb3">我在 bbb3 内</div>
<div class="bbb4">我在 bbb4 内</div>
</body>
</html>
```

运行结果如图 5-16 所示。

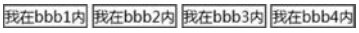

图 5-16　行内元素显示效果

(3) 行内块元素(inline-block)：结合行内元素和块级元素的优点，既可以设置长宽，可以让 padding 和 margin 生效，又可以和其他行内元素并排。

【例 5-12】 inline-block 样式实例。

```
<!DOCTYPE html>
<HTML>
<head>
<style type="text/CSS">
    div {
    border-width: 4px;
    border-color: red;
    border-style: double;
    }
    #ID1 {
    background-color:rebeccapurple;
    color:#FFF;
display: inline-block;
    height: 200px;
    width: 200px;
    }
    #ID2 {
    font: normal 14px/1.5 Verdana, sans-serif;
    height: 200px;
    width: 200px;
    background-color: rosybrown;
display: inline-block;
    }
</style>
</head>
<body>
<div>
<div id="ID1">ID1</div>
<div id="ID2">ID2</div>
        box1
</div>
```

```
</body>
</HTML>
```

程序运行效果如图 5-17 所示。

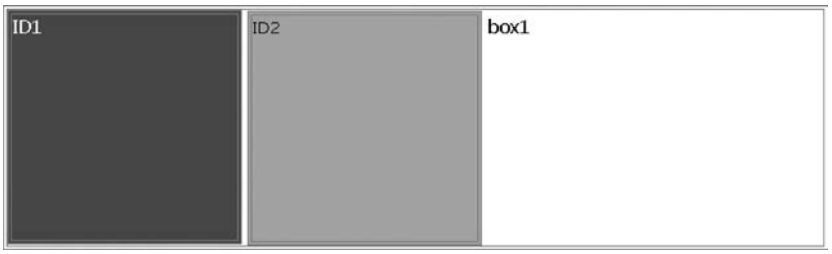

图 5-17 inline-block 行内块元素显示效果

观察图 5-17 可以发现,使用 inline-block 布局两个元素之间会有一个空白间隙。因为现在使用的是 inline-block 元素,为了方便理解,可以将 inline-block 元素看成两个文字,文字与文字之间不可能是连在一起的,肯定是有间隙的。既然知道了是文字的问题,那就将父元素 container 的字体大小设置为 0,可是这个时候会发现 left 和 right 这两个单词也没有了,这是因为 left 和 right 元素继承了父级元素的字体大小,这时候只需要分别设置 left 和 right 元素的字体大小即可。

【例 5-13】 inline-block 样式消除间隙实例。

```
<!DOCTYPE html>
<HTML>
<head>
<style>
<!-- @import "inline-block1.css"; -->
</style>
</head>
<body>
<div class="container">
<div class="left">left</div>
<div class="right">right</div>
</div>
</body>
</HTML>
inline-block1.css 的内容为
.container {
    width: 800px;
    height: 200px;
    font-size: 0; /* 新增 */
  }
  .left {
    font-size: 14px; /* 新增 */
    background-color: red;
    display: inline-block;
    width: 200px;
    height: 200px;
  }
  .right {
```

```
    font-size: 14px; /* 新增 */
    background-color: blue;
    display: inline-block;
    width: 600px;
    height: 200px;
  }
```

程序运行效果如图 5-18 所示。

图 5-18　inline-block 样式消除间隙效果

5.5.2　盒子模型

盒子模型是 HTML 和 CSS 中最核心的内容，结合浮动、定位等技术实现布局。CSS 中所有页面元素都包含在一个矩形框内，这个矩形框就称为盒子。盒子描述了元素及属性在页面布局中所占空间大小，因此盒子可以影响其他元素的位置及大小。掌握盒子模型需要从两个方面理解：一是理解单个盒子的内部结构，二是理解多个盒子之间的关系。

盒子模型用于设置元素的边界(margin)、边界补白(padding)、边框(border)等属性值，使用这一属性的大多是块元素。W3C 组织建议把所有网页上的对象都放在一个盒子(box)中，设计师可以通过创建定义来控制这个盒子的属性，这些对象包括段落、列表、标题、图片以及层。盒子模型主要定义四个区域：内容 (content)、边框距或空白、边界和边距。内容是盒子模型中必需的部分，可以是文字、图片等元素。padding 也称页边距或补白，用来设置内容和边框之间的距离。border 可以设置内容边框线的粗细、颜色和样式等。margin 是外边距，用来设置内容与内容之间的距离。margin，padding，content，border 之间的层次、关系和相互影响的盒子模型如图 5-19 所示。

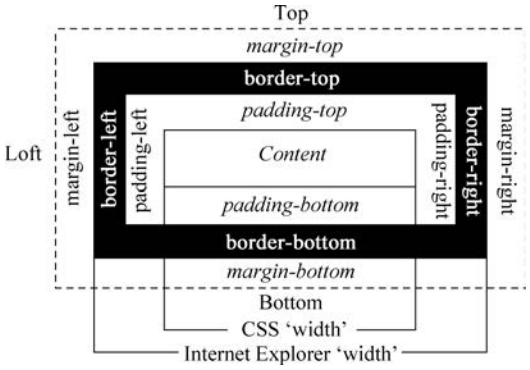

图 5-19　方框盒子模型

margin：包括 margin-top，margin-right，margin-bottom，margin-left，控制块级元素之间的距离，它们是透明不可见的。对于上右下左 margin 值均为 40px，因此代码为：

```
margin-top: 40px;margin-right: 40px;margin-bottom: 40px;margin-left: 40px;
```

根据上、右、下、左的顺时针规则，简写为 margin：40px 40px 40px 40px；。为便于记忆，请参考如图 5-20 所示的顺序书写。

当上下、左右 margin 值分别一致时，可简写为：margin：40px 50px；。

40px 代表上下 margin 值，50px 代表左右 margin 值。

当上下左右 margin 值均一致时，可简写为：margin：40px；。

当上下 margin 值不一致，而左右 margin 值均一致时，可简写为：margin：40px 50px 20px；。

40px，20px 代表上下 margin 值，50px 代表左右 margin 值。

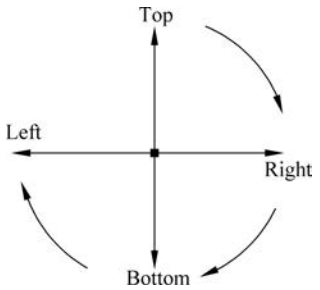

图 5-20 margin 四个位置的书写顺序

如果只提供一个数，将用于全部的四条边；如果提供两个数，第一个用于上下，第二个用于左右；如果提供三个数，第一个用于上，第二个用于左右，第三个用于下；如果提供全部四个参数值，将按上、右、下、左的顺序作用于四边。

margin 不会在绝对元素上折叠。假设有一个 margin-bottom 值为 20px 的段落。在段落后面是一个具有 30px 的 margin-top 的图片。那么段落和图片之间的空间不会是 50px(20px＋30px)而是 30px(30px＞20px)。这就是所谓的 margin-collapse，两个 margin 会合并(折叠)成一个 margin。

绝对定位元素不会像那样进行 margin 的折叠，这会使它们跟预期的不一样。

【例 5-14】 margin 的样式示例。

```
<!DOCTYPE html>
<HTML>
<head>
    <style type="text/CSS">
        #ID1 {
background-color: #333;
color:#FFF;
margin:10px;
}
#ID2 {
font: normal 14px/1.5 Verdana, sans-serif;
margin:30px;
border: 1px solid #F00;
}
</style>
</head>
<body>
    <div id="ID1">
        Hello,world
        <h1 id="ID2">Margins of ID1 and ID2 collapse vertically.<br />
            元素 ID1 与 ID2 的 margins 在垂直方向折叠.</h1>
    </div>
</body>
</HTML>
```

运行程序，结果如图 5-21 所示。

图 5-21　margin 样式结果

padding：包括 padding-top、padding-right、padding-bottom、padding-left，控制块级元素内部 content 与 border 之间的距离。简写请参考 margin 属性的写法，如

```
body {padding: 36px;}                    //对象四边的补丁边距均为 36px
body { padding: 36px 24px; }             //上下两边补丁边距为 36px,左右两边补丁边距为 24px
body {padding: 36px 24px 18px; }         //上下两边的补丁边距分别为 36px、18px,左右两边的补丁边
                                         距均为 24px
body {padding: 36px 24px 18px 12px; }    //上、右、下、左补丁边距分别为 36px、24px、18px、12px
```

border：可以按顺序设置如下属性：border-width，border-style，border-color。如果不设置其中的某个值，也不会出问题，如 border：solid #ff0000；也是允许的。

（1）border-width 简写属性为元素的所有边框设置宽度，或者单独地为各边框设置宽度。只有当边框样式不是 none 时才起作用。如果边框样式是 none，边框宽度实际上会重置为 0。不允许指定负长度值。可能的 border-width 取值及其对应的描述如下。

thin：定义细的边框。

medium：默认值，定义中等的边框。

thick：定义粗的边框。

length：允许自定义边框的宽度。

（2）border-style 属性用于设置元素所有边框的样式，或者单独地为各边设置边框样式。只有当这个值不是 none 时边框才可能出现。可能的 border-style 取值及其对应的描述如下。

none：定义无边框。

hidden：与 none 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。

dotted：定义点状边框。在大多数浏览器中呈现为实线。

dashed：定义虚线。在大多数浏览器中呈现为虚线。

solid：定义实线。

double：定义双线。双线的宽度等于 border-width 的值。

groove：定义 3D 凹槽边框。其效果取决于 border-color 的值。

ridge：定义 3D 垄状边框。其效果取决于 border-color 的值。

inset：定义 3D inset 边框。其效果取决于 border-color 的值。

outset：定义 3D outset 边框。其效果取决于 border-color 的值。

最不可预测的边框样式是 double。它定义为两条线的宽度再加上这两条线之间的空间等于 border-width 值。

（3）border-color 属性设置四条边框的颜色。此属性可设置 1～4 种颜色。border-color 属性是一个简写属性，可设置一个元素的所有边框中可见部分的颜色，或者为 4 个边分别设置不同的颜色。简写请参考 margin 属性的写法，例如：

```
border - color:red green blue pink;
```

表示上边框是红色，右边框是绿色，下边框是蓝色，左边框是粉色。

```
border-color:red green blue;
```

表示上边框是红色，右边框和左边框是绿色，下边框是蓝色。

```
border-color:dotted red green;
```

表示上边框和下边框是红色，右边框和左边框是绿色。

```
border-color:red;
```

表示所有 4 个边框都是红色。

可能的 border-color 取值及其对应的描述如下。

color_name：规定颜色值为颜色名称的边框颜色，如 red。

hex_number：规定颜色值为十六进制值的边框颜色，如 #ff0000。

rgb_number：规定颜色值为 rgb 代码的边框颜色，如 rgb(255,0,0)。

transparent：默认值。边框颜色为透明。

【例 5-15】 padding 和 border 样式实例。

```
<!DOCTYPE html>
<HTML>
<head>
<style type="text/CSS">
    #ID1 {
    background-color: #333;
    color:#FFF;
    margin:10px;
    padding:15px;
    }
    #ID2 {
    font: normal 14px/1.5 Verdana, sans-serif;
    margin:30px;
    padding:15px;
    border-width: 2px ;
    border-style:groove;
    border-color: red blue green;
    }
</style>
</head>
<body>
<div id="ID1">要站在统筹中华民族伟大复兴战略全局和世界百年未有之大变局的高度,统筹国内国际两个大局、发展安全两件大事,</div>
<h1 id="ID2">充分发挥海量数据和丰富应用场景优势,促进数字技术与实体经济深度融合,<br/>赋能传统产业转型升级,催生新产业新业态新模式,不断做强做优做大我国数字经济。</h1>
</body>
</HTML>
```

运行程序，结果如图 5-22 所示。

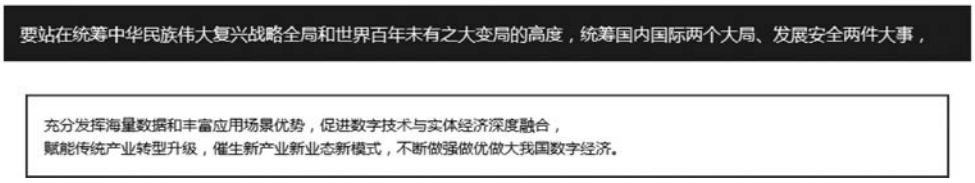

图 5-22　padding 和 border 样式结果

【例 5-16】　计算盒子模型元素的总宽度。

```
<!DOCTYPE html>
<HTML>
<head>
<style type="text/CSS">
    div {
        background-color: #333;
        color:#FFF;
        margin:10px;
        padding:15px;
        border-width: 20px;
        width: 200px;
    }
</style>
</head>
<body>
<div>要站在统筹中华民族伟大复兴战略全局和世界百年未有之大变局的高度,统筹国内国际两个大局、发展安全两件大事。</div>
<div>我的总宽度为多少?</div>
</body>
</HTML>
```

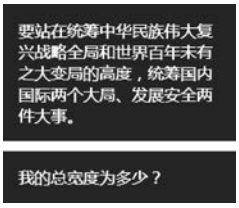

图 5-23　盒子模型的宽度

运行程序，结果如图 5-23 所示，总宽度为 200px(宽)＋10px×2(左右 margin)＋15px×2(左右 padding)＋20px×2(左右边框)＝290px。

5.5.3　标准文档流

标准文档流，指的是元素排版布局过程中，元素会默认自动从左往右，从上往下的流式排列方式。前面内容发生了变化，后面的内容位置也会随着发生变化。HTML 就是一种标准文档流文件。HTML 中的标准文档流的特点通过两种方式体现：微观现象和元素等级。

微观现象有：空白折叠现象(不管有几个空格都会展示一个空格)；文字类的元素如果排在一行会出现一种高低不齐、底边对齐效果；自动换行，元素内一行内容写满元素的 width 时会自动进行换行。

【例 5-17】　微观现象实例。

```
<!DOCTYPE html>
<HTML>
<head>
<meta charset="UTF-8"/>
```

```
</head>
<body>
<section>
<span>有</span><span>无</span>
</section>
<section>
<span>生</span><span>成</span>
</section>
<section>
<span>有</span>
<span>无</span>
</section>
</body>
</HTML>
```

一个空格一个回车和多个空格多个回车都会折叠成一个空格显示在页面上，折叠的空格是当前父元素的文字字体大小。

在标准流中大部分元素是区分等级的，习惯上将元素划分为几种常见的加载级别：块级元素、行内元素、行内块元素等。

标准流中的元素有自己默认的浏览器加载模式，但是加载模式不是一成不变的，后期可以通过 display 属性更改一个标签的显示模式。

display 属性更改的显示模式并没有改变标准流本质性质，页面还是只能从上往下加载，存在空白折叠现象等微观性质。要想实现更多的界面布局效果需要脱离标准流的限制。标签元素脱离标准流的方法包括：浮动、绝对定位、固定定位。

5.5.4 浮动 float 样式

在传统的印刷布局中，文本可以按照需要围绕图片。一般把这种方式称为“文本环绕”。在网页设计中，应用了 CSS 的浮动 float 属性的页面元素就像在印刷布局里面的被文字包围的图片一样。

【语法】

```
{float : none | left |right}
```

参数值：

none：对象不浮动。

left：对象浮在左边，文字下沉。

right：对象浮在右边。

【例 5-18】 float 样式实例。

```
<!DOCTYPE html>
<HTML>
<head>
    <Meta http-equiv="Content-type" content="text/HTML; charset=UTF-8" />
    <link rel="stylesheet" type="text/CSS" href="main.CSS" />
    <title>CSS FLOAT</title>
    <style type="text/CSS">
        .top {
```

```
        width:500px;                    /* div 框的宽度 */
        background:#f1f1f1;             /* div 框的背景色 */
    }
    .img {
        float:left;                     /* 图片向左浮动 */
        margin-right:10px;              /* 图片右侧与文字的边距 */
        margin-bottom:5px;              /* 图片下部与文字的边距 */
        width: 200px;
        height: 150px;
        border:thin dotted red;
    }
</style>
</head>
<body>
    <!-- 环绕的图片及文字,图片的 CSS 类为 img -->
    <h1>滕王阁序</h1>
    <img src="tengwangge.jpg" alt="文字环绕" class="img" />
    <div class="top">豫章故郡,洪都新府。星分翼轸,地接衡庐。襟三江而带五湖,控蛮荆而引瓯越。物华天宝,龙光射牛斗之墟;人杰地灵,徐孺下陈蕃之榻。雄州雾列,俊采星驰。台隍枕夷夏之交,宾主尽东南之美。都督阎公之雅望,棨戟遥临;宇文新州之懿范,襜帷暂驻。十旬休假,胜友如云;千里逢迎,高朋满座。腾蛟起凤,孟学士之词宗;紫电青霜,王将军之武库。家君作宰,路出名区;童子何知,躬逢胜饯。
    </div>
</body>
</HTML>
```

运行程序,结果如图 5-24 所示。

图 5-24　float 样式结果

不能在同一个属性当中同时应用定位属性和浮动。因为对使用什么样的定位方案来说两者的指令是相冲突的。如果两个属性添加到相同的元素上,CSS 会选择后面的属性。

5.5.5　元素定位

定位模式规定了一个盒子在总体的布局上应该处于什么位置以及对周围的盒子会有什么影响。定位模式包括常规文档流、浮动和 position 定位的元素。

【语法】

```
HTML 标签 { position: absolute | relative | fixed | static }
```

static 是 position 默认的属性值。任何应用了 position:static 的元素都处于常规文档流中。它处于什么位置以及它如何影响周边的元素都是由盒子模型所决定的。一个 static 定位

的元素会忽略所有 top，right，bottom，left 以及 z-index 属性所声明的值。不能通过 z-index 进行层次分级。

【例 5-19】 static 位置样式实例。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>position</title>
<style>
            * {
                padding: 0;
                margin: 0;
            }
            .static {
                position: static;
                top: 10px;
                left: 10px;
                width: 100px;
                height: 200px;
                background: yellow;
            }
</style>
</head>
<body>
<div class="static"></div>
</body>
</html>
```

运行程序，结果如图 5-25 所示。

absolute，绝对定位的元素会从常规文档流中脱离。对于包围它的元素而言，它会将该绝对定位元素视为不存在，也就是说，其在正常流中的原有位置不再存在。可以通过 top，right，bottom 和 left 四个属性来设置绝对定位元素的位置。但通常只会设置它们其中的两个：top 或者 bottom，以及 left 或者 right。它们的默认值都为 auto。

假设初始位置如图 5-26 所示，Box1 占据 left 和 top 为(0，0)的位置，下面的 Box2 为 static。

图 5-25 static 位置样式结果

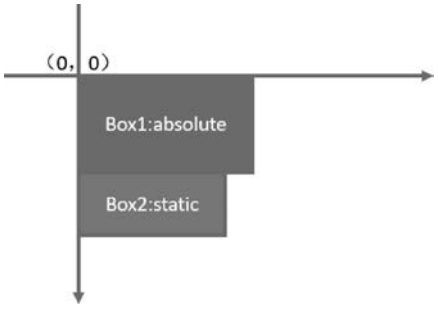

图 5-26 Box1 和 Box2 的初始位置

如果 Box1 的样式设置如下：

```
#box1{
position: absolute ;
top:20px;
left:20px;
}
```

Box1 会往右和往下分别平移 20px，则原来的位置将会被下面的 Box2 占据，变化后的位置如图 5-27 所示。

对于 absolute 定位的层总是相对其最近的定义为 absolute 或 relative 的父元素，也就是非 static 的父元素，而这个父元素不一定是其直接父层。如图 5-28 所示，absolute 的 div 首先查看父元素 div3，如果 div3 为 static，则查 div2。如果 div2 为非 static，则相对 div2 向右平移 20px，向下平移 20px。

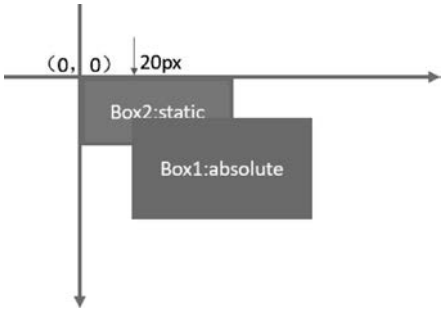

图 5-27　Box1 和 Box2 变换后的位置

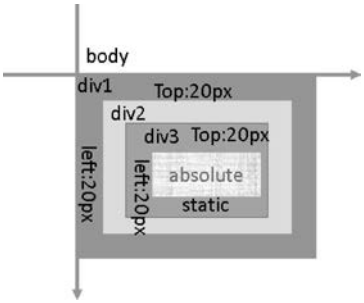

图 5-28　absolute 的相对定位

生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。如果不设置 left，top，right 以及 bottom，会在父元素的适当位置显示。

【例 5-20】 没有设置上下左右位置的固定样式实例。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>position</title>
<style>
            .relative {
                width: 300px;
                height: 300px;
                border: 1px solid;
                position: relative;
                left: 100px;
                top: 100px;
            }
            .absolute {
                position: absolute;
                width: 100px;
                height: 100px;
                border: 1px solid red;
            }
</style>
```

```
</head>
<body>
<div class="relative">
            outer
<div class="absolute"></div>
</div>
</body>
</html>
```

运行程序，结果如图 5-29 所示。

设置了 left 将以父元素的左边界为基准，向右偏移，垂直方向和之前相同。设置了 right 将以父元素的右边界为基准，向左偏移，垂直方向和之前相同。设置了 top 将以父元素的上边界为基准，向下偏移，水平方向和之前相同。设置了 bottom，将以父元素的下边界为基准，向上偏移，水平方向和之前相同。

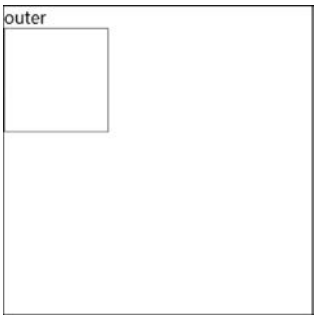

图 5-29 没有设置上下左右位置样式效果

【例 5-21】 设置偏移位置的样式实例。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>position</title>
<style>
            .relative {
                width: 300px;
                height: 300px;
                border: 1px solid;
                position: relative;
                left: 100px;
                top: 100px;
            }
            .absolute1, .absolute2, .absolute3, .absolute4, .absolute5 {
                position: absolute;
                width: 100px;
                height: 100px;
                border: 2px solid red;
            }
            .absolute2 {
                left: 60px;
                border-color: green;
            }
            .absolute3 {
                right: 10px;
                border-color: yellow;
            }
            .absolute4 {
                top: 30px;
                border-color: pink;
```

```
            }
            .absolute5 {
                bottom: 10px;
                border-color: blue;
            }
</style>
</head>
<body>
<div class="relative">
<div class="absolute1">box1</div>
<div class="absolute2">box2</div>
<div class="absolute3">box3</div>
<div class="absolute4">box4</div>
<div class="absolute5">box5</div>
</div>
</body>
</html>
```

运行程序，结果如图 5-30 所示。

relative 方式：以标准流的排版方式为基础，相对定位的元素根据 top、right、bottom 和 left 四个属性相对标准流位置决定自己的位置。相对定位元素离开了正常文档流，但仍然影响着围绕着它的元素，也就是说，在文档中原来的位置还在。那些元素表现得就好像这个相对定位元素仍然在正常文档流当中。假设初始位置如图 5-31 所示，Box1 占据 left 和 top 为(0,0)的位置，下面的 Box2 为 static。

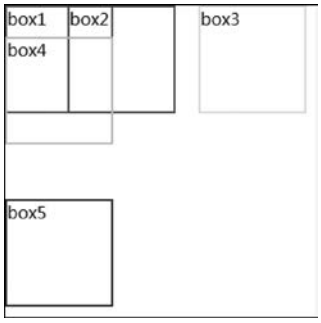

图 5-30　absolute 位置样式效果

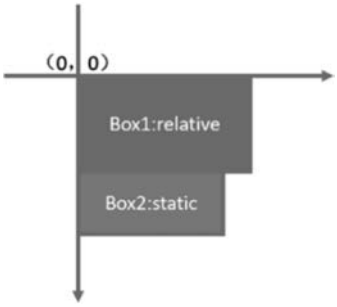

图 5-31　relative 属性元素的初始位置

设置 Box1 的样式：

```
#box1{
width:170px;
height:190px;
position: relative;
top:20px;
left:20px
}
```

Box1 发生了平移后，静态属性的 Box2 没有占领 Box1 的位置，如图 5-32 所示。

relative 定位的层总是相对其直接父元素，无论其父元素是什么定位方式，如图 5-33 所示，relative 的 div 是相对其直接父元素 div3 定位 top 和 left。

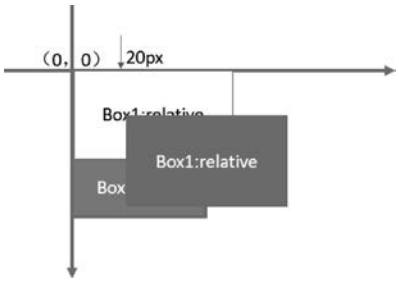

图 5-32　相对位置的原始位置依然有影响

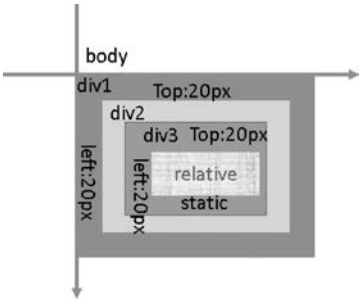

图 5-33　relative 属性定位

生成相对定位的元素，相对于其标准流正常位置进行定位。相对元素正常应该在的位置移动，元素所占的空间位置不变，但是显示的位置发生偏移。left 是向右偏移，right 是向左偏移，top 向下偏移，bottom 向上偏移。

【例 5-22】 设置 relative 的样式实例。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>position</title>
<style>
            * {
                padding: 0;
                margin: 0;
            }
            .div {
                width: 100px;
                height: 100px;
                border: 1px solid;
                float: left;
            }
            .relative1 {
                position: relative;
                left: 10px;
                top: 10px;
                border-color: red;
            }
            .relative2 {
                position: relative;
                right: 10px;
                bottom: 10px;
                border-color: green;
            }
</style>
</head>
<body>
<div class="div">box1</div>
<div class="div relative1">box2</div>
```

```
<div class="div">box3</div>
<div class="div relative2">box4</div>
</body>
</html>
```

程序运行效果如图 5-34 所示。一般父元素设计为 relative，子元素为 absolute，则子元素可以随父元素一起移动。

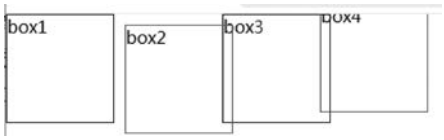

图 5-34　relative 位置样式效果

fixed，相对于浏览器窗口进行定位。元素的位置通过 top，right，bottom 和 left 属性进行规定。只和浏览器窗口有关，与父元素、文档流都无关。固定定位的行为类似于绝对定位，但也有一些不同的地方。第一个不同点，固定定位总是相对于浏览器窗口来进行定位的，并且通过 top，right，bottom 和 left 属性来决定其位置，它抛弃了它的父元素。第二个不同点是继承性，固定定位的元素是固定的。它们并不随着页面的滚动而移动。常见的案例是，左边和右边都是广告，中间为内容，广告设置为 fixed，则无论中间的内容如何变化，广告区域都不会变化，如图 5-35 所示。

图 5-35　fixed 样式效果

【例 5-23】　设置 fixed 的样式实例。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>position</title>
<style>
            * {
                padding: 0;
                margin: 0;
            }
            .fixed-outer {
                position: fixed;
                top: 20px;
                left: 20px;
                width: 50px;
```

```
            height: 50px;
            background: yellow;
            border: groove red;
        }
        .fixed - inner {
            position: fixed;
            top: 40px;
            left: 40px;
            width: 50px;
            height: 50px;
            background: red;
        }
</style>
</head>
<body>
<div class = "fixed - outer"> outer box
<div class = "fixed - inner"> inner box </div>
</div>
</body>
</html>
```

图 5-36　fixed 位置样式效果

运行效果如图 5-36 所示。

综上，4 种位置关系的定位参照物、原来的位置和新位置的关系总结如表 5-5 所示。

表 5-5　position 定位的关系总结

方　　式	定位参照物	新　位　置	原 来 位 置
static	标准文档流	top，right，bottom 和 left 无效	—
absolute	非 static 的父元素	top，right，bottom 和 left	不保留
relative	直接父元素	top，right，bottom 和 left	保留
fixed	浏览器窗口	top，right，bottom 和 left	—

5.5.6　z-index 空间中定位元素

三维空间中定位元素的属性是 z-index，打破了二维平面的约束，具有宽度和高度。z-index 属性设置元素的堆叠顺序。拥有更高堆叠顺序的元素总是会处于堆叠顺序较低的元素的前面。该属性设置一个定位元素沿 Z 轴的位置，Z 轴定义为垂直延伸到显示区的轴。如果为正数，则离用户更近，为负数则表示离用户更远。

【例 5-24】　z-index 样式实例。

```
<!DOCTYPE html>
<HTML>
<head>
    <style type = "text/CSS">
img{
position:absolute;
left:0px;
top:0px;
z - index: - 1;
```

```
}
</style>
</head>
<body>
    <h1 style="color:red">This is a heading</h1>
    <img src="images/c.jpg" />
    <p style="color:red">由于图像的 z-index 是 -1,因此它在文本的后面出现.</p>
</body>
</HTML>
```

程序运行效果如图 5-37 所示。

由图 5-38 可知,z-index 高的位于 z-index 低的上面并朝页面的上方运动。与此相反,一个低的 z-index 在高的 z-index 的下面并朝页面下方运动。所有元素的默认的 z-index 值都为 0,并且可以对 z-index 使用负值。

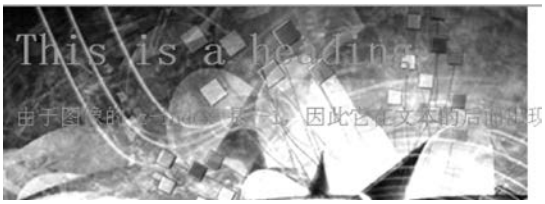

图 5-37　z-index 样式效果

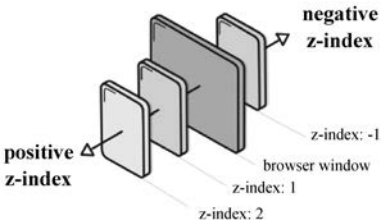

图 5-38　z-index 的上下层关系

假如只是开发简单的弹窗效果,懂得通过 z-index 来调整元素间的层叠关系就够了。但要将多个弹窗间的层叠关系给处理好,那么充分理解 z-index 背后的原理及兼容性问题就是必要的知识储备了。常接触到的 z-index 只是分层显示中的一个属性,而理解 z-index 背后的原理实质上就是要理解分层显示原理。

5.5.7　inline-block 布局和浮动布局对比

inline-block 布局和浮动布局效果类似,但也有区别。如果把 inline-block 布局修改为浮动布局,即 display: inline-block 替换为 float: left,发现 float 就会使得元素脱离文本流,且还有父元素高度坍塌的效果。

【例 5-25】 inline-block 布局修改为浮动布局实例。

```
<!DOCTYPE html>
<HTML>
<head>
    <style type="text/CSS">
    div {
    border-width: 4px;
    border-color: red;
    border-style: double;
    }
    #ID1 {
    background-color:rebeccapurple;
    color:#FFF;
float :left;
```

```
    height: 200px;
    width: 200px;
    }
    #ID2 {
    font: normal 14px/1.5 Verdana, sans-serif;
    height: 200px;
    width: 200px;
    background-color: rosybrown;
float :left;
    }
</style>
</head>
<body>
    <div>
        <div id="ID1">ID1</div>
        <div id="ID2">ID2</div>
        box1
    </div>
</body>
</HTML>
```

程序运行效果如图 5-39 所示。

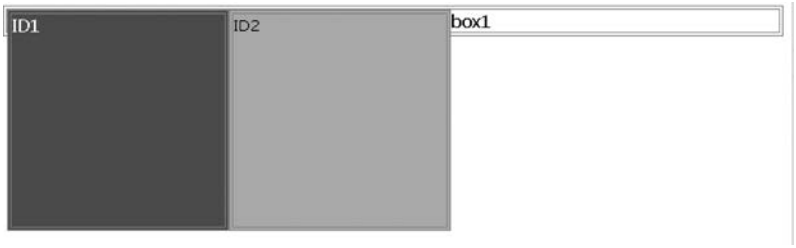

图 5-39 inline-block 布局修改为浮动布局的效果

为了克服坍塌问题,需要闭合浮动,对父元素 box1 设置 overflow:hidden。

【例 5-26】 浮动闭合实例。

```
<!DOCTYPE html>
<HTML>
<head>
    <style type="text/CSS">
        div {
    border-width: 4px;
    border-color: red;
    border-style: double;
overflow: hidden;
    }
    #ID1 {
    background-color:rebeccapurple;
    color:#FFF;
    float :left;
    height: 200px;
    width: 200px;
    }
```

```
    #ID2 {
    font: normal 14px/1.5 Verdana, sans-serif;
    height: 200px;
    width: 200px;
    background-color: rosybrown;
float :left;
    }
</style>
</head>
<body>
    <div>
        <div id="ID1">ID1</div>
        <div id="ID2">ID2</div>
        box1
    </div>
</body>
</HTML>
```

程序运行效果如图 5-40 所示。

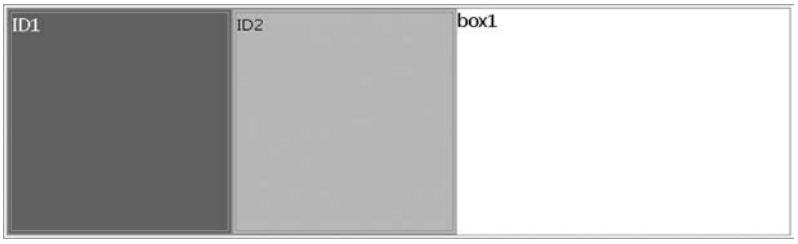

图 5-40　闭合浮动的效果

可见消除了坍塌问题，对比 inline-block 显示方式，消除了缝隙，与设置了字体大小为 0 的效果一样。浮动布局不太好的地方：参差不齐的现象，即使添加了 vertical-align:bottom 也不起作用，也是顶部对齐。

【例 5-27】　多个元素的浮动布局对齐实例。

```
<!DOCTYPE html>
<HTML>
<head>
    <style type="text/CSS">
        div {
    border-width: 4px;
    border-color: red;
    border-style: double;
    overflow: hidden;
    float :left;
    }
    #ID1 {
    background-color:rebeccapurple;
    color:#FFF;
    height: 100px;
    width: 200px;
    }
    #ID2 {
    font: normal 14px/1.5 Verdana, sans-serif;
```

```
    height: 150px;
    width: 200px;
    background-color: rosybrown;
    }
    #ID3 {
    background-color:rebeccapurple;
    color:#FFF;
    height: 200px;
    width: 200px;
    }
    #ID4 {
    font: normal 14px/1.5 Verdana, sans-serif;
    height: 250px;
    width: 200px;
    background-color: rosybrown;
    }
</style>
</head>
<body>
    <div>
        <div id="ID1">ID1</div>
        <div id="ID2">ID2</div>
        <div id="ID3">ID3</div>
        <div id="ID4">ID4</div>
        box1
    </div>
</body>
</HTML>
```

程序运行效果如图 5-41 所示。

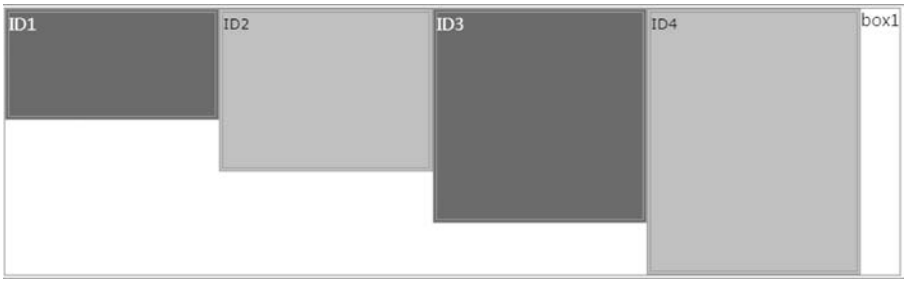

图 5-41　浮动布局的参差不齐效果

对于 inline-block 显示方式默认的情况下也是顶部对齐，效果如图 5-42 所示。

图 5-42　inline-block 的默认顶部对齐效果

但可以通过设置 vertical-align 在顶部、底部或基线对齐。

【例 5-28】 多个元素的浮动布局＋vertical-align 的对齐实例。

```
<!DOCTYPE html>
<HTML>
<head>
<style type="text/CSS">
    div {
    border-width: 4px;
    border-color: red;
    border-style: double;
    vertical-align:bottom;
    }
    #ID1 {
    background-color:rebeccapurple;
    color:#FFF;
    height: 100px;
    width: 200px;
    display: inline-block;
    }
    #ID2 {
    font: normal 14px/1.5 Verdana, sans-serif;
    height: 150px;
    width: 200px;
    background-color: rosybrown;
    display: inline-block;
    }
    #ID3 {
    background-color:rebeccapurple;
    color:#FFF;
    height: 200px;
    width: 200px;
    display: inline-block;
    }
    #ID4 {
    font: normal 14px/1.5 Verdana, sans-serif;
    height: 260px;
    width: 200px;
    background-color: rosybrown;
    display: inline-block;
    }
</style>
</head>
<body>
<div>
    <div id="ID1">ID1</div>
    <div id="ID2">ID2</div>
    <div id="ID3">ID3</div>
    <div id="ID4">ID4</div> box1
</div>
</body>
</HTML>
```

程序运行效果如图 5-43 所示。

图 5-43 inline-block＋vertical-align 的底部对齐效果

综上可见，浮动布局主要用于文字环绕，而水平排列最好使用 inline-block。

5.6 基于中国风的 Web 样式设计——文化和技术的融合

中国风设计不仅是指某个元素，它应该是一个整体，需要有自己的特色，无论设计手法如何变化，只要作品呈现出来，都会有广泛认同的中国风格设计特点。中国风格的网页样式应该是一种建立在中国文化和东方生活方式基础上的，并适应全球经济发展趋势的、有着自身独特魅力和性格的作品。

“中国风”设计的兴起最初并非起源于中国。16 世纪在欧洲产生了一种东方情调的装饰艺术风格，被称为“中国风格”或“中国风”，主要表现在装饰艺术领域，以中国人物或中国动植物、风景为题材，在色彩配置、构图形式上，也部分地借鉴了东方艺术的特色。“中国风”设计虽然血统混杂，不中不西，却具有独特的艺术魅力，它应当是设计师对民族文化理解后的一种自然情感的流露。

1998 年，美国动画大片《花木兰》在全球轰动，是美国制作团队第一次将中国元素传遍世界。

2008 年上映的美国动作喜剧动画电影《功夫熊猫》再次风靡全球，尖端的科技和超凡的想象力把中国元素演绎得出神入化，以中国功夫为主题，以中国古代为背景，其景观、布景、服装以至食物均充满中国元素。

荣膺第 55 届戛纳电影节“最佳导演奖”的韩国电影《醉画仙》，其官方网站的界面设计东方特色浓郁，利用 Web 动画技术把中国传统的水墨画技法体现得淋漓尽致，观者似乎不是在浏览一个网站，而是在欣赏一位水墨大师肆意挥毫，兔起鹘落间笔走龙蛇、墨分五色，或浓或淡、或焦或渴、或干裂秋风、或润含春雨，那种心灵上的震撼与共鸣是难以言表的。创造了一种忆江南的浓浓的中国情调，体现了江南独特的魅力，整个过程由开始的动画入手，给人一种推开一扇沉重的历史大门，然后看里面的情节变化，沧桑风雨，整个网站没有任何现代的东西，全部是中国古文化的结晶，悠悠百年，尽在其中。

中国古籍在装帧设计上形式多样活泼，洋溢着鲜明的中国民族气派，蕴涵了意念的空灵美、淡雅的色彩美、严整的秩序美等中国传统美学精神。

案例网站的界面设计采用了独具东方情调的艺术风格和传统的文化符号，根据网站的内容，采用了独具东方情调的艺术风格和传统的文化符号，以古典的中国画、书法、古建筑物等中国图形元素为主，既有工笔的精雕细刻，又有写意的随性挥洒，或丝丝细线，刚劲而秀妍，层层重彩，色和而调鲜。在视觉呈现上匠心独运，寻求古典与现代最佳的契合点，令观者身心俱醉，

流连忘返，使浏览者感受到中国传统文化的独特魅力。

Web 网页设计应用了大量的高科技方法，增强了交互性，表现形式与表现力远胜于传统艺术形式，能够使审美主体产生更大的沉浸感与愉悦感，但不能形成“唯技术至上”的思维定式。Web 样式设计除了技术之外，通过对优秀网站作品的观摩与学习，更多的是对传统文化的渗透与熏陶，探讨中国元素在网站 UI 设计中的应用。

习题

1. 通过实例，比较亮度 brightness 和对比度 contrast 滤镜的区别。
2. 通过实例，比较透明程度 opacity 和饱和度 saturate 滤镜的区别。
3. 通过实例，比较移动、旋转、缩放和倾斜的区别和联系。
4. 通过实例，比较 transition 过渡和 animation 动画的区别和联系。
5. 通过实例，比较 block，inline，inline-block 的区别和联系。
6. 通过实例，比较盒子模型的边界(margin)、边界补白(padding)、边框(border)的区别和联系。
7. 比较标准文档流、inline-block 和浮动 float 样式的关系。
8. 通过实例，比较 absolute、relative、fixed 和 static 四种元素定位方式的区别和联系。
9. 分析一个经典的具有中国风特色的网站样式设计。