第 5 章

CSS3 新增特性

CSS3 是 CSS(层叠样式表)技术的升级版本。CSS3 完全向后兼容,不必改变现有的设计,浏览器将永远支持 CSS2。CSS3 在 CSS2.1 的基础上增加了很多强大的新功能,以帮助开发人员解决一些实际面临的问题,本章将探索 CSS3 中引入的一些高级功能,例如渐变颜色、阴影效果、2D 和 3D 变换、alpha 透明度,以及创建过渡和动画效果的方法,此外还将探索伸缩布局、滤镜效果、媒体概念查询等。

熟悉基础知识后,我们将进入下一个级别,介绍使用图像精灵在网页上放置元素的方法及相对和关联的概念,如绝对单位、视觉格式模型、显示和可见性、图层、伪类和元素、与媒体相关的样式表等。

本章思维导图如图 5-1 所示。

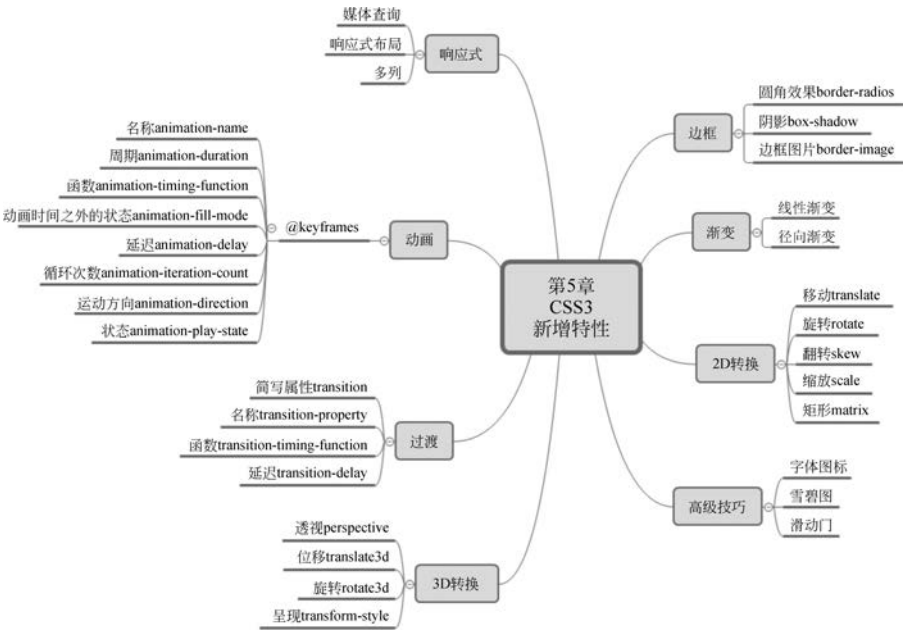


图 5-1　思维导图

5.1 CSS3 边框与渐变

5.1.1 CSS3 特效边框

CSS3 新增了 3 种边框特效，分别是圆角边框、有阴影效果的边框和图片边框，如表 5-1 所示。

表 5-1 新增边框效果

属 性 名	描 述	属 性 名	描 述
border-radius	设置圆角的边框	border-image	设置带背景图片的边框
box-shadow	设置带阴影效果的边框		

1. 圆角边框

传统的圆角生成方案，必须使用多张图片作为背景图案。CSS3 的出现，使我们再也不必浪费时间去制作这些图片了。

CSS3 圆角只需设置一个属性：border-radius(含义是“边框半径”)。

基本语法格式如下：

```
border-radius: [ <length> | <percentage> ]
```

各参数的解释如下。

(1) <length>：用长度值设置对象的圆角半径长度，不允许负值。

(2) <percentage>：用百分比设置对象的圆角半径长度，不允许负值。

border-radius 属性可以接受 1～4 个值，规则如下。

(1) 4 个值：第 1 个值为左上角，第 2 个值为右上角，第 3 个值为右下角，第 4 个值为左下角。

(2) 3 个值：第 1 个值为左上角，第 2 个值为右上角和左下角，第 3 个值为右下角。

(3) 2 个值：第 1 个值为左上角与右下角，第 2 个值为右上角与左下角。

(4) 1 个值：4 个圆角值相同。

border-radius 属性实际上是一种简写形式，也可以单独对每个角进行设置，如表 5-2 所示。

表 5-2 圆角各边框属性

属 性 名	描 述
border-radius	所有 4 条边角 border-*-*-radius 属性的缩写
border-top-left-radius	定义了左上角的弧度
border-top-right-radius	定义了右上角的弧度

续表

属 性 名	描 述
border-bottom-right-radius	定义了右下角的弧度
border-bottom-left-radius	定义了左下角的弧度

使用 border-radius 系列属性为元素设置圆角边框效果，如例 5-1 所示。

【例 5-1】 圆角边框效果

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>圆角边框效果</title>
    <style>
        #rcorners1 {
            border-radius: 25px; /* 4 个角的弧度都是 25px */
            background: #73AD21;
            padding: 20px;
            width: 200px;
            height: 50px;
        }
        #rcorners2 {
            border-radius: 25px;
            border: 2px solid #73AD21;
            padding: 20px;
            width: 200px;
            height: 50px;
        }
        #rcorners3 {
            border-radius: 25px;
            background: url(images/gzh.png);
            background-position: left top;
            background-repeat: repeat;
            padding: 20px;
            width: 190px;
            height: 50px;
        }
    </style>
</head>
<body>
    <p>拥有指定背景颜色的元素的圆角：</p>
    <p id="rcorners1">背景圆角</p>
    <p>带边框元素的圆角：</p>
    <p id="rcorners2">边框圆角</p>
    <p>拥有背景图片的元素的圆角：</p>
    <p id="rcorners3">背景图片圆角 </p>
</body>
</html>
```

在浏览器中的显示效果如图 5-2 所示。

图 5-2　圆角边框效果

2. 盒子阴影

box-shadow 属性可以为边框添加阴影，该属性适用于所有元素。box-shadow 的默认值为 none，表示没有阴影效果。

基本语法格式如下：

```
box-shadow: h-shadow v-shadow blur spread color inset;
```

各参数的解释如下。

(1) h-shadow：必选，水平阴影的位置，允许负值。

(2) v-shadow：必选，垂直阴影的位置，允许负值。

(3) blur：可选，模糊距离。

(4) spread：可选，阴影的尺寸。

(5) color：可选，阴影的颜色，如果不设置，则默认为黑色。

(6) inset：可选，关键字，将外部投影(默认 outset)改为内部投影，inset 阴影在背景之上，在内容之下。

使用 box-shadow 属性为元素设置边框阴影效果，如例 5-2 所示。

【例 5-2】 创建纸质卡片效果

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

```
<head>
    <meta charset="UTF-8">
    <title>创建纸质卡片效果</title>
    <style>
        div.card {
            width: 250px;
            /* 将水平阴影设置为 5px,将垂直阴影设置为 4px,将阴影模糊距离设置为 8px,将阴
            影尺寸设置为 0,阴影颜色为 red */
            box-shadow: 5px 4px 8px 0 red;
            text-align: center;
        }
        div.header {
            background-color: #4CAF50;
            color: white;
            padding: 10px;
            font-size: 40px;
        }
        div.container {
            padding: 10px;
        }
    </style>
</head>
<body>
    <p>box-shadow 属性可用于创建类似纸质的卡片:</p>
    <div class="card">
        <div class="header">
            <h1>1</h1>
        </div>
        <div class="container">
            <p>January 1, 2021</p>
        </div>
    </div>
</body>
</html>
```

在浏览器中的显示效果如图 5-3 所示。

图 5-3 创建纸质卡片效果

box-shadow 可向框添加一个或多个阴影，该属性是由逗号分隔的阴影列表，如例 5-3 所示。

【例 5-3】 多阴影列表

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>多个阴影列表</title>
    <style>
        div{
            width: 100px;height: 100px;
            box-shadow: -10px 0px 10px red,              /*左边阴影*/
            0px -10px 10px #000,                         /*上边阴影*/
            10px 0px 10px green,                         /*右边阴影*/
            0px 10px 10px blue;"                         /*下边阴影*/
        }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

在浏览器中的显示效果如图 5-4 所示。

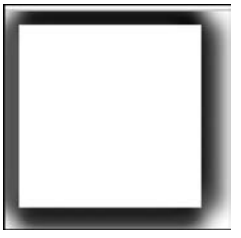


图 5-4 多阴影列表效果

3. 图像边框

border-image 用于给 border(边框)贴上背景图像，该属性适用于所有元素。border-image 属性是一个简写属性，相关属性如表 5-3 所示。

表 5-3 图像边框相关属性

属 性 名	描 述
border-image-source	用在边框的图片的路径
border-image-slice	图片边框向内偏移
border-image-width	图片边框的宽度

续表

属 性 名	描 述
border-image-outset	边框图像区域超出边框的量
border-image-repeat	图像边框是否应平铺(repeated)、铺满(rounded)或拉伸(stretched)
border-image	复合属性

使用 border-image 属性为元素设置图像边框的效果,如例 5-4 所示。

【例 5-4】 图像边框

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>图像边框</title>
    <style>
        div{
            text-align: center;
            border: 15px solid #09f;
            width: 200px;
            border-image: url("images/border.jpg") 15 15 round;
        }
    </style>
</head>
<body>
    <div>图像边框</div>
</body>
</html>
```

在浏览器中的显示效果如图 5-5 所示。

图 5-5 图像边框效果

5.1.2 渐变

CSS3 渐变(gradients)可以让我们在两个或多个指定的颜色之间显示平稳的过渡。以前,必须使用图像实现这些效果,但是,通过 CSS3 渐变,可以减少下载的时间和宽带

的使用。此外，渐变效果的元素在放大时看起来效果更好，因为渐变是由浏览器生成的。

CSS3 定义了以下两种类型的渐变。

(1) 线性渐变(Linear Gradients)：向下、向上、向右、向左、对角方向(to bottom、to top、to right、to left、to bottom right 等)。

(2) 径向渐变(Radial Gradients)：由它们的中心定义。

因为渐变是一个比较新的属性，所以它在浏览器中使用时需要添加它的前缀，如图 5-6 所示是完全支持该属性的第 1 个浏览器版本。

属性					
linear-gradient	10.0	26.0 10.0 -webkit-	16.0 3.6 -moz-	6.1 5.1 -webkit-	12.1 11.1 -o-
radial-gradient	10.0	26.0 10.0 -webkit-	16.0 3.6 -moz-	6.1 5.1 -webkit-	12.1 11.6 -o-
repeating-linear-gradient	10.0	26.0 10.0 -webkit-	16.0 3.6 -moz-	6.1 5.1 -webkit-	12.1 11.1 -o-
repeating-radial-gradient	10.0	26.0 10.0 -webkit-	16.0 3.6 -moz-	6.1 5.1 -webkit-	12.1 11.6 -o-

图 5-6　浏览器对渐变的支持情况

1. CSS3 线性渐变(linear-gradient/repeating-linear-gradient)

为了创建一个线性渐变，必须至少定义两种颜色节点。颜色节点即想要呈现平稳过渡的颜色。同时，也可以设置一个起点和一个方向(或一个角度)。

基本语法格式如下：

```
background - image: linear - gradient(direction, color - stop1, color - stop2, ...);
```

其中，direction 是渐变的方向。color- * 表示渐变颜色，可以有无数种颜色。

1) 线性渐变：从上到下(默认情况下)

下面的实例演示了从顶部开始的线性渐变。起点是红色，慢慢过渡到蓝色，示例代码如下：

```
.divOne {
    width: 100px;
    height: 100px;
    background: linear - gradient(red, blue);
}

< div class = "divOne"></div >
```

示例的应用效果如图 5-7 所示。

2) 线性渐变：从左到右

下面的实例演示了从左边开始的线性渐变。起点是红色，慢慢过渡到蓝色，示例代码如下：

```
.divOne {
    width: 100px;
    height: 100px;
    background: linear-gradient(to right, red, blue);
}
```

示例的应用效果如图 5-8 所示。

3）线性渐变：对角

可以通过指定水平和垂直的起始位置来制作一个对角渐变。

下面的实例演示了从左上角开始（到右下角）的线性渐变。起点是红色，慢慢过渡到蓝色，示例代码如下：

```
.divOne {
    width: 100px;
    height: 100px;
    background: linear-gradient(to bottom right, red, blue);
}
```

示例的应用效果如图 5-9 所示。

图 5-7 从顶向下的线性渐变

图 5-8 从左到右的线性渐变

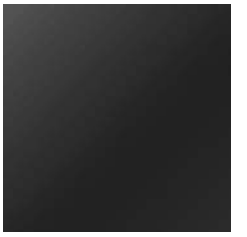


图 5-9 对角线性渐变

4）使用角度

如果想要在渐变的方向上实现更多的控制，则可以定义一个角度，而不用预定义方向（to bottom、to top、to right、to left、to bottom right 等）。

基本语法格式如下：

```
background-image: linear-gradient(angle, color-stop1, color-stop2);
```

其中，angle 表示角度。

角度是指水平线和渐变线之间的角度，逆时针方向计算。换句话说，0deg 将创建一个从下到上的渐变，90deg 将创建一个从左到右的渐变，如图 5-10 所示。

但是，需要注意很多浏览器（Chrome、Safari、Firefox 等）使用了旧的标准，即 0deg 将创建一个从左到右的渐变，90deg 将创建一个从下到上的渐变。换算公式 $90-x=y$，其中 x

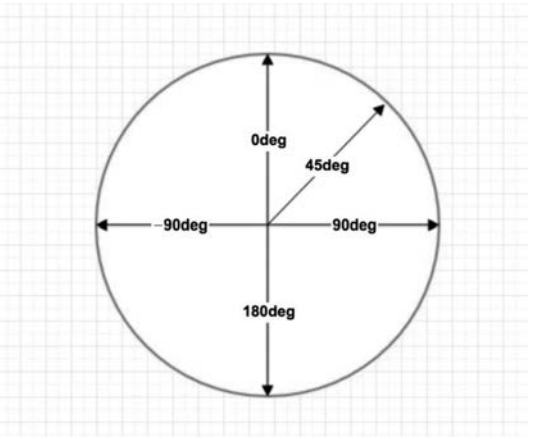


图 5-10　角度

为标准角度，y 为非标准角度。

带有指定角度的线性渐变，示例代码如下：

```
.divOne {
    width: 100px;
    height: 100px;
    background: linear-gradient(30deg, red, blue);
}
```

图 5-11　带有指定角度的线性渐变

示例应用效果如图 5-11 所示。

5）使用透明度（transparent）

CSS3 渐变也支持透明度，可用于创建减弱变淡的效果。

为了添加透明度，可以使用 rgba() 函数来定义颜色节点。rgba() 函数中的最后一个参数可以是 0～1 的值，它定义了颜色的透明度：0 表示完全透明，1 表示完全不透明。

下面的实例演示了从左边开始的线性渐变。起点完全透明，慢慢过渡到完全不透明的红色，代码如下：

```
.divOne {
    width: 100px;
    height: 100px;
    background: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));
}
```

6）重复的线性渐变

repeating-linear-gradient() 函数用于重复线性渐变，代码如下：

```
.divOne {
    width: 100px;
    height: 100px;
    background: linear-gradientrepeating-linear-gradient(red, yellow 10%, green 20%);
}
```

2. CSS3 径向渐变 radial-gradient()/repeating-radial-gradient()

径向渐变由它的中心定义。为了创建一个径向渐变，必须至少定义两种颜色节点。颜色节点即想要呈现平稳过渡的颜色。同时，也可以指定渐变的中心、形状（圆形或椭圆形）、大小。默认情况下，渐变的中心是 center（表示在中心点），渐变的形状是 ellipse（表示椭圆形），渐变的大小是 farthest-corner（表示到最远的角落）。

基本语法格式如下：

```
background: radial-gradient(center, shape ,size, start-color, ..., last-color);
```

各参数的解释如下：

（1）center 表示渐变的中心。

（2）shape 表示径向渐变的形状，有两个可选值：circle（圆）和 ellipse（椭圆形）。

（3）size 表示确定径向渐变的结束形状大小，默认值为 farthest-corner，其他值如下。

closest-side：指定径向渐变的半径长度为从圆心到离圆心最近的边；

closest-corner：指定径向渐变的半径长度为从圆心到离圆心最近的角；

farthest-side：指定径向渐变的半径长度为从圆心到离圆心最远的边；

farthest-corner：指定径向渐变的半径长度为从圆心到离圆心最远的角。

（4）color 表示渐变的起止颜色。

颜色节点不均匀分布的径向渐变，示例代码如下：

```
.divOne {
    width: 100px;
    height: 100px;
    background-image: radial-gradient(red 5%, yellow 15%, green 60%);
}
```

示例的应用效果如图 5-12 所示。

repeating-radial-gradient() 函数用于重复径向渐变，用法同重复线性渐变相同。

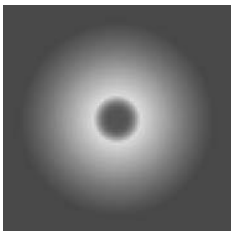


图 5-12　径向渐变

5.2　转换

CSS3 中的转换允许对元素进行旋转、缩放、移动或倾斜。它为分 2D 转换和 3D 转换。

在 CSS2 时代，如果要实现一些图片转换角度，则要依赖于图片、Flash 或 JavaScript 才能完成，但是现在借助 CSS3 就可以轻松倾斜、缩放、移动及翻转元素。通过 CSS 变形，可以让元素生成静态视觉效果，也可以很容易结合 CSS3 的过渡和动画产生一些动画效果。

5.2.1　2D 转换

CSS 2D Transform 表示 2D 转换，目前主流浏览器对 transform 属性的支持情况如图 5-13 所示。

属性					
transform	36.0 4.0 -webkit-	10.0 9.0 -ms-	16.0 3.5 -moz-	3.2 -webkit-	23.0 15.0 -webkit- 12.1 10.5 -o-

图 5-13　浏览器版本对 transform 属性的支持情况

注意：紧跟在 -webkit-、-ms- 或 -moz- 前的数字为支持该前缀属性的第 1 个浏览器版本号。如 Chrome 4.0～35.0 版本支持使用前缀 -webkit-，所以在写成 -webkit-transform 的形式，其他浏览器类似。

通常的属性包含了属性名和属性值，而 transform 的属性值是用函数来定义的。定义 transform 的属性值有 5 种方法，如表 5-4 所示。

表 5-4　transform 属性方法

函　数　名	描　　述
translate(x,y)	元素移动到指定位置，即基于 X 和 Y 坐标重新定位元素
rotate(deg)	元素顺时针旋转指定的角度，负数表示逆时针旋转
scale(x,y)	元素尺寸缩放指定的倍数
skew(xdeg,ydeg)	围绕 X 轴和 Y 轴将元素翻转指定的角度
matrix(n,n,n,n,n,n)	该方法包含了矩阵变换数学函数，根据填入数据的不同可以实现元素的旋转、缩放、移动(平移)和倾斜功能

基本语法格式如下：

```
transform:函数名(x轴值,y轴值);
```

1. 移动 translate()

translate(x,y)方法，根据左(*X* 轴)和顶部(*Y* 轴)位置给定的参数，从当前元素位置移动。

基本语法格式如下：

```
transform: translate(x,y);
```

如 div{transform：translate(50px,100px);}表示元素从左边向右移动 50 像素，并从顶部向下移动 100 像素。

如果省略 *y*，则默认 *y* 轴上的移动距离为 0。

也可以单独使用 translateX()或 translateY()方法指定在水平或垂直方向中的一个方向上移动的距离。

指定在水平方向上平移的语法格式如下：

```
transform: translateX(x);
```

指定在垂直方向上平移的语法格式如下：

```
transform: translateY(y);
```

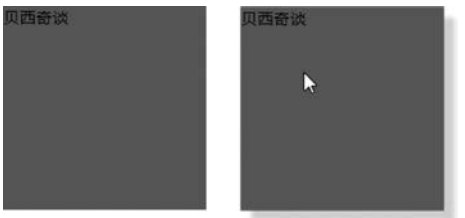
使用 transform 属性的 translate()方法对元素进行 2D 平移，如例 5-5 所示。

【例 5-5】 2D 转换(移动)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>2D 转换(移动)</title>
    <style>
        .box{
           width: 200px;
           height: 200px;
           background: red;
           margin: 30px;
        }
        .box:hover{                                    /*鼠标指针悬停触发*/
           transform: translate(5px, -5px);
```

```
            -ms-transform:translate(5px,-5px);          /* IE 9 */
            -webkit-transform:translate(5px,-5px);      /* Safari andChrome */
            box-shadow: 10px 10px 5px #ddd;             /* 设置盒子阴影 */
        }
    </style>
</head>
<body>
    <div class="box">贝西奇谈</div>
</body>
</html>
```

在浏览器中的显示效果如图 5-14 所示。



(a) 初始加载　　(b) 鼠标悬停

图 5-14　移动变换前后效果

2. 旋转 rotate()

通过 rotate(deg) 方法,元素顺时针旋转给定的角度。允许负值,当为负值时元素将逆时针旋转。它以 deg 为单位,代表旋转的角度。

基本语法格式如下:

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

其中,<angle>表示顺时针旋转指定角度。

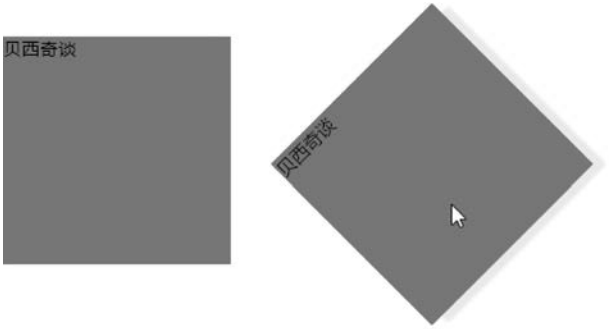
使用 transform 属性的 rotate()方法对元素进行 2D 旋转,如例 5-6 所示。

【例 5-6】 2D 转换(旋转)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>2D 转换(旋转)</title>
    <style>
        .box{
            width: 200px;
            height: 200px;
            background: red;
```

```
            margin: 100px;
        }
        .box:hover{                                    /* 鼠标指针悬停触发 */
            transform: rotate( - 45deg);
            - ms - transform:rotate( - 45deg);         /*  IE 9  */
            - webkit - transform:rotate( - 45deg);     /* Safari and Chrome  */
            box - shadow: 10px 10px 5px #ddd;          /* 设置盒子阴影 */
        }
    </style>
</head>
<body>
    <div class = "box">贝西奇谈</div>
</body>
</html>
```

在浏览器中的显示效果如图 5-15 所示。



(a) 初始加载　　(b) 鼠标悬停

图 5-15　旋转变换前后效果

3. 缩放 scale()

scale(x,y)函数能够缩放元素大小,该函数包含两个参数值,分别用来定义宽和高的缩放比例。

基本语法格式如下:

```
transform: scale(x,y);
```

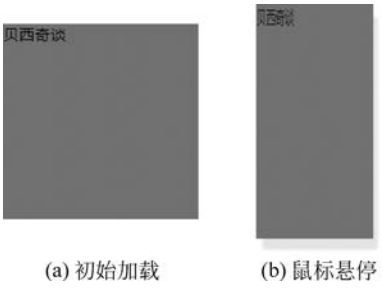
参数值可以是正数、负数和小数。正数值基于指定的宽度和高度将放大元素。负数值不会缩小元素,而是翻转元素(如文字被翻转),然后缩放元素。小数可以缩小元素。如果第 2 个参数省略,则第 2 个参数等于第 1 个参数值。

使用 transform 属性的 scale()方法对元素进行 2D 缩放,如例 5-7 所示。

【例 5-7】 2D 转换(缩放)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>2D 转换(缩放)</title>
    <style>
        .box{
           width: 200px;
           height: 200px;
           background: red;
           margin: 100px;
        }
        .box:hover{                                    /* 鼠标指针悬停触发 */
           transform: scale(0.6,1.2);
           -ms-transform:scale(0.6,1.2);               /* IE 9 */
           -webkit-transform:scale(0.6,1.2);           /* Safari and Chrome */
           box-shadow: 10px 10px 5px #ddd;             /* 设置盒子阴影 */
        }
    </style>
</head>
<body>
    <div class="box">贝西奇谈</div>
</body>
</html>
```

在浏览器中的显示效果如图 5-16 所示。



(a) 初始加载　　(b) 鼠标悬停

图 5-16　缩放变换前后效果

4. 翻转 skew()

skew(xdeg,ydeg)可用于在页面上翻转元素。

基本语法格式如下：

```
transform:skew(<angle>,<angle>);
```

其中，<angle>表示角度值，两个参数值分别表示 X 轴和 Y 轴倾斜的角度，如果第 2 个参数为空，则默认为 0，参数为负表示向相反方向倾斜。

也可以单独使用 skewX()或 skewY()方法指定在水平或垂直方向中的一个方向上的翻转情况。

(1) skewX(<angle>)表示只在 X 轴(水平方向)倾斜。

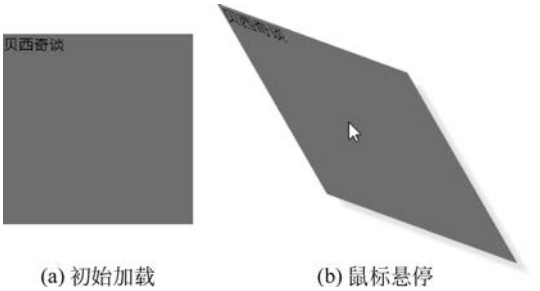
(2) skewY(<angle>)表示只在 Y 轴(垂直方向)倾斜。

使用 transform 属性的 skew()方法对元素进行 2D 翻转，如例 5-8 所示。

【例 5-8】 2D 转换(翻转)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>2D 转换(翻转)</title>
    <style>
        .box{
            width: 200px;
            height: 200px;
            background: red;
            margin: 100px;
        }
        .box:hover{                                    /* 鼠标指针悬停触发 */
            transform: skew(30deg,20deg);
            -ms-transform:skew(30deg,20deg);           /* IE 9 */
            -webkit-transform:skew(30deg,20deg);       /* Safari and Chrome */
            box-shadow: 10px 10px 5px #ddd;            /* 设置盒子阴影 */
        }
    </style>
</head>
<body>
    <div class="box">贝西奇谈</div>
</body>
</html>
```

在浏览器中的显示效果如图 5-17 所示。



(a) 初始加载　　(b) 鼠标悬停

图 5-17　翻转变换前后效果

5. 矩阵变换 matrix()

matrix(n,n,n,n,n,n)是矩阵函数,调用该函数可以非常灵活地实现各种变换效果,如旋转、缩放、移动(平移)和倾斜功能。

基本语法格式如下:

```
transform:matrix(n,n,n,n,n,n);
```

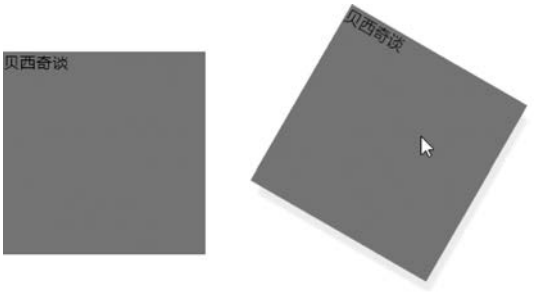
其中,第 1 个参数控制 x 轴缩放,第 2 个参数控制 x 轴倾斜,第 3 个参数控制 y 轴倾斜,第 4 个参数控制 y 轴缩放,第 5 个参数控制 x 轴移动,第 6 个参数控制 y 轴移动。

使用 transform 属性的 matrix()方法对元素进行 2D 矩阵变换,如例 5-9 所示。

【例 5-9】 2D 转换(矩阵变换)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>2D 转换(矩阵变换)</title>
    <style>
        .box{
           width: 200px;
           height: 200px;
           background: red;
           margin: 100px;
        }
        .box:hover{                                          /* 鼠标指针悬停触发 */
           transform:matrix(0.866,0.5,-0.5,0.866,0,0);
           -ms-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* IE 9 */
                 /* Safari and Chrome */
           -webkit-transform:matrix(0.866,0.5,-0.5,0.866,0,0);
              box-shadow: 10px 10px 5px #ddd;              /* 设置盒子阴影 */
        }
    </style>
</head>
<body>
    <div class="box">贝西奇谈</div>
</body>
</html>
```

在浏览器中的显示效果如图 5-18 所示。



(a) 初始加载　　(b) 鼠标悬停

图 5-18　矩阵变换变换前后效果

5.2.2　3D 转换

我们生活的环境是 3D 的，照片就是 3D 物体在 2D 平面呈现的例子。CSS3 中的 3D 坐标系其实就是指立体空间，立体空间是由 3 个轴共同组成的，如图 5-19 所示。

CSS3 的 3D 转换主要包含的函数如下。

(1) 透视：perspective。

(2) 3D 位移：translate3d(x,y,z)。

(3) 3D 旋转：rotate3d(x,y,z)。

(4) 3D 呈现 transform-style。

1. 透视 perspective

如果想要在网页产生 3D 效果，则需要透视(理解成 3D 物体投影在 2D 平面内)。模拟人类的视觉位置，可认为安排一只眼睛去看。距离视觉点越近的物体在计算机平面成像越大，越远的物体成像越小(近大远小视觉效果)，如图 5-20 所示。

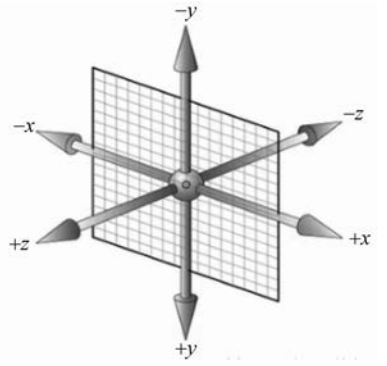


图 5-19　3D 坐标系

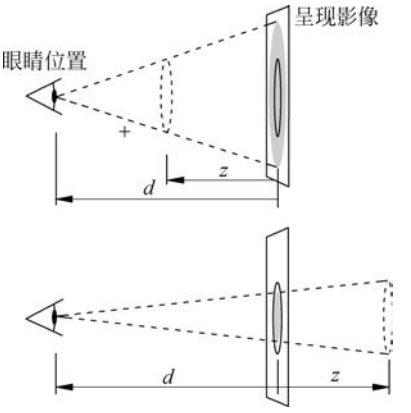


图 5-20　透视效果

2. 3D 移动 translate3d(x,y,z)

3D 移动在 2D 移动的基础上多加了一个可以移动的方向，即 z 轴方向。

基本语法格式如下：

```
translform: translate3d(x,y,z)
```

其中，x，y，z 分别指要移动的轴的方向的距离。

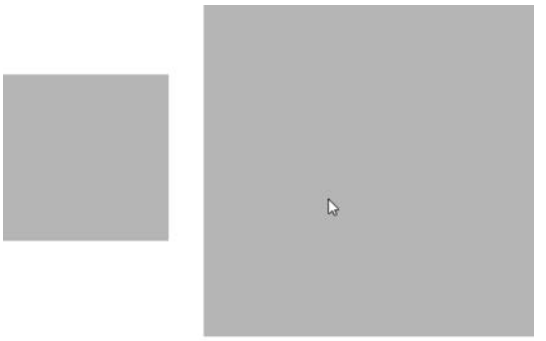
也可以单独使用 translateX()、translateY()、translateZ()函数指定其中一个方向上的移动。

如例 5-10 所示通过比较原图和 3D 位移图，比较移动前后效果。

【例 5-10】 3D 转换(移动)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>3D 转换(移动)</title>
    <style>
        body {
            perspective: 600px;                    /* 给父盒子添加透视效果 */
        }
        div {
            width: 200px;
            height: 200px;
            background-color: pink;
            margin: 100px auto;
        }
        div:hover {                                /* 鼠标悬停 */
            /* transform: translateX(100px); */
            /* transform: translateY(100px); */
            transform: translateZ(300px);
            /* 透视是眼睛到屏幕的距离,透视只是一种展示形式,是有 3D 效果的意思 */
            /* translateZ 是物体到屏幕的距离,Z 是来控制物体近大远小的具体情况 */
            /* translateZ 越大,我们看到的物体越近,物体越大 */
            /* transform: translate3d(x, y, z); x 和 y 可以是 px,可以是 %,但是 z 只能是 px */
        }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

在浏览器中的显示效果如图 5-21 所示。



(a) 初始加载　　(b) 鼠标悬停

图 5-21　3D 转换(移动)前后效果

3. 旋转：rotate3d(x,y,z,deg)

3D 旋转只可以让元素在三维平面内沿着 X 轴、Y 轴、Z 轴或者自定义轴进行旋转。

基本语法格式如下：

```
transform: rotate3d(x,y,z,deg)
```

各参数的解释如下。

(1) x：是 0～1 的数值，描述围绕 X 轴旋转的向量值。

(2) y：是 0～1 的数值，描述围绕 Y 轴旋转的向量值。

(3) z：是 0～1 的数值，描述围绕 Z 轴旋转的向量值。

(4) deg：沿着自定义轴旋转，deg 为角度。

如 transform:rotate3d(1,0,0,45deg)表示沿着 X 轴旋转 45°。

也可以单独使用 rotate3dX()、rotate3dY()、rotate3dZ()函数指定其中一个方向上的旋转。如例 5-11 所示通过比较原图和 3D 旋转图，比较旋转前后的效果。

【例 5-11】 3D 转换(旋转)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>3D 转换(旋转)</title>
    <style>
        div {
            width: 224px;
            height: 224px;
            margin: 100px auto;
            position: relative;
        }
```

```
        div img {
            position: absolute;
            top: 0;
            left: 0;
        }
        div img:first-child {
            z-index: 1;
            backface-visibility: hidden;      /* 如果不是正面对向屏幕,就隐藏 */
        }
        div:hover img {                       /* 鼠标悬停 */
            transform: rotateY(180deg);
        }
    </style>
</head>
<body>
    <div>
        <img src="images/qian.svg" alt=""/>
        <img src="images/hou.svg" alt=""/>
    </div>
</body>
</html>
```

在浏览器中的显示效果如图 5-22 所示。

(a) 初始加载　　(b) 鼠标悬停

图 5-22　3D 转换(旋转)前后效果

4. 呈现 transform-style

控制子元素是否开启三维立体环境。transform-style：flat 表示子元素不开启 3D 立体空间,此为默认值。transf-style：preserve-3d 表示子元素开启立体空间。

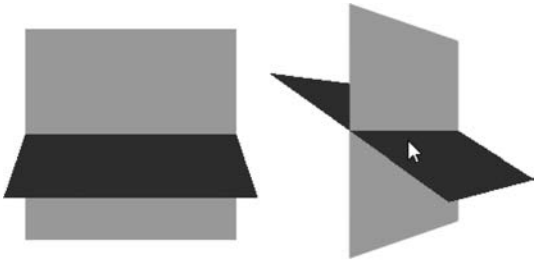
呈现代码写给父级,但是影响的是子盒子,如例 5-12 所示。

【例 5-12】 3D 呈现

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>3D 呈现</title>
```

```
    <style>
        body{
            perspective: 500px;
        }
        .box{
            position: relative;
            width: 200px;
            height: 200px;
            margin: 100px auto;
            transition: all 2s;                          /* 过渡 */
            /* 让子元素保持 3D 立体空间环境 */
            transform-style: preserve-3d;
        }
        .box:hover{                                      /* 鼠标悬停 */
            transform: rotateY(60deg);
        }
        .box div{
            position: absolute;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
            background-color: chartreuse;
        }
        .box div:last-child{
            background-color: blue;
            transform: rotateX(60deg);
        }
    </style>
</head>
<body>
    <div class="box">
        <div></div>
        <div></div>
    </div>
</body>
</html>
```

在浏览器中的显示效果如图 5-23 所示。



(a) 初始加载　　(b) 鼠标悬停

图 5-23　3D 呈现效果

5.3 过渡与动画

5.3.1 过渡

通过过渡 transition 可以在指定时间内将元素从原始样式平滑变化为新的样式，如鼠标悬停后，背景色在 1s 内，由白色平滑地过渡到红色。

目前主流浏览器对 transition 属性的支持情况如图 5-24 所示。

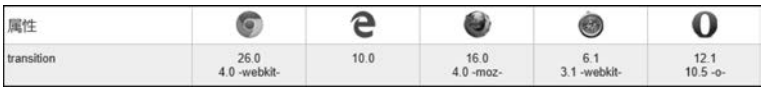


图 5-24 浏览器版本对 transition 属性的支持情况

解释说明：

紧跟在 -webkit-、-ms- 或 -moz- 前的数字为支持该前缀属性的第 1 个浏览器版本号。如 Chrome 4.0～35.0 版本支持使用前缀 -webkit-，应该写成 -webkit-transition 的形式，其他浏览器类似。

过渡 transition 包含 4 种属性，如表 5-5 所示。

表 5-5 transition 过渡属性

属 性 名	描 述
transition-property	规定应用过渡的 CSS 属性的名称
transition-duration	定义过渡效果花费的时间，默认为 0
transition-timing-function	规定过渡效果的时间曲线，默认为 "ease"
transition-delay	规定过渡延迟时间，默认为 0
transition	简写属性，用于在一个属性中设置 4 个过渡属性

1. 过渡属性 transition-property

transition-property 属性用于指定需要过渡发生的 CSS 属性。

基本语法格式如下：

```
transition-property: none|all|property;
```

各参数的解释如下。

(1) none：没有指定任何样式。

(2) all：默认值，表示指定元素所有支持 transition-property 属性的样式。

(3) property：设置过渡的 CSS 属性，如果是多个属性，则用逗号隔开。

同时设置元素的宽度和高度发生过渡，代码如下：

```
p{transition-property: width, height};
```

在实际工作中一般取默认值。

2. 过渡持续时间 transition-duration

transition-duration 属性用于指定过渡持续的时间,为必选赋值属性。

基本语法格式如下:

```
transition-duration: <time>;
```

该属性的单位是秒(s)或毫秒(ms)。

3. 过渡函数 transition-timing-function

transition-timing-function 属性用于设置过渡函数。

基本语法格式如下:

```
transition-timing-function: linear | ease | ease-in | ease-out | ease-in-out | cubic-
bezier;
```

各参数的解释如下。

(1) linear:线性过渡,等同于贝塞尔曲线(0.0, 0.0, 1.0, 1.0)。

(2) ease:平滑过渡,等同于贝塞尔曲线(0.25, 0.1, 0.25, 1.0)。

(3) ease-in:由慢到快,等同于贝塞尔曲线(0.42, 0, 1.0, 1.0)。

(4) ease-out:由快到慢,等同于贝塞尔曲线(0, 0, 0.58, 1.0)。

(5) ease-in-out:由慢到快再到慢,等同于贝塞尔曲线(0.42, 0, 0.58, 1.0)。

(6) cubic-bezier(<number>, <number>, <number>, <number>):特定的贝塞尔曲线类型,4 个数值需要在[0, 1]区间内。

4. 过渡延迟 transition-delay

transition-delay 属性用于指定过渡后多长时间开始执行过渡效果。

基本语法格式如下:

```
transition-delay: <time>;
```

5. 复合属性 transition

transition 属性是一个简写属性,用于设置 4 个过渡属性。

基本语法格式如下:

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

参数之间使用空格隔开，如有未声明的参数取其默认值。

注意：如果只提供一个时间参数，则默认为 transition-duration 属性值。

使用 transition 各个属性实现过渡效果，如例 5-13 所示。

【例 5-13】 过渡

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>过渡</title>
    <style>
        .box{
            width: 200px;
            height: 300px;
            background-color: red;
            margin: 50px;
            /* 等价于 transition:transform 1s, box-shadow 1s,width
                1s,background-color 1s ; */
            /* transition: all 1s linear 1s; */
            transition-property: all;
            transition-duration: 1s;                    /* 持续 1s */
            transition-timing-function:linear;          /* 线性过渡 */
            transition-delay:1s;                        /* 延迟 1s */
        }
        .box:hover{                                     /* 鼠标悬停 */
            transform: translate(0, -10px);
            box-shadow: 0 15px 30px rgba(0,0,0,.3);
            width: 300px;
            background-color: green;
        }
    </style>
</head>
<body>
    <div class="box"></div>
</body>
</html>
```

在浏览器中的显示效果如图 5-25 所示。

(a) 初始加载　　(b) 鼠标悬停

图 5-25　过渡效果

5.3.2 动画

CSS3 使用 animation 属性定义帧动画。动画是 CSS3 中具有颠覆性的特征之一，可通过设置多个关键帧(节点)来精确控制一个或一组动画，常用来实现复杂的动画效果。

与 animation 动画相关的属性如表 5-6 所示。

表 5-6 animation 动画相关属性

属 性 名	描 述
@keyframes	设置自定义动画内容
animation-name	设置需要执行的 @keyframes 动画名称
animation-duration	设置动画完成一个周期所花费的时间(秒或毫秒)，默认为 0
animation-timing-function	设置动画的时间曲线，默认为 "ease"
animation-fill-mode	动画时间之外的状态
animation-delay	规定动画延迟的时间，默认为 0
animation-iteration-count	规定动画循环播放的次数，默认为 1
animation-direction	规定动画的运动方向，默认为 "normal"
animation-play-state	规定动画是否正在运行或暂停，默认为 "running"
animation	所有动画属性的简写属性

1. @keyframes 规则

如需使用 CSS 动画，则必须首先为动画指定一些关键帧。帧(frame)是影像动画中最小单位的单幅影像画面，一帧就相当于一幅静止的画面，连续的帧可以形成动画效果。

如果在 @keyframes 规则中指定了 CSS 样式，则动画将在特定时间逐渐从当前样式更改为新样式，其格式如下：

```
@keyframes 动画名称{
  from {样式}
  to {样式}
}
```

其中，动画名称可以自定义，from 表示起始帧的样式，to 表示最终帧的样式。

要使动画生效，必须将动画绑定到某个元素。

例如，将 "example" 动画绑定到 元素。动画将持续 4s，同时将 元素的背景颜色从 "red" 逐渐改为 "yellow"，代码如下：

```
/* 动画代码 */
@keyframes example {
   from {background-color: red;}
   to {background-color: yellow;}
}
```

```
/* 向此元素应用动画效果 */
div {
   width: 100px;
   height: 100px;
   background-color: red;
   animation-name: example;              /* 动画名称 example */
   animation-duration: 4s;               /* 动画持续时间 4s */
}
```

注意：animation-duration 属性表示需要多长时间才能完成动画。如果未指定 animation-duration 属性，则动画不会发生，因为默认值为 0s。

也可以使用百分比值。通过百分比，可以根据需要添加任意多个样式进行更改。

例如，将在动画完成 25%、完成 50% 及动画完成 100% 时更改 元素的背景颜色，代码如下：

```
/* 动画代码 */
@keyframes example {
   0% {background-color: red;}
   25% {background-color: yellow;}
   50% {background-color: blue;}
   100% {background-color: green;}
}

/* 应用动画的元素 */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

2. 动画名称 animation-name

animation-name 属性专门用于指定需要发生的动画名称，必须与规则@keyframes 配合使用，因为动画名称由@keyframes 定义。

基本语法格式如下：

```
animation-name: none | <identifier>;
```

各参数的解释如下。

(1) none：不引用任何动画名称。

(2) <identifier>：定义一个或多个动画名称(identifier 标识)。

3. 动画持续时间 animation-duration

animation-duration 属性用于指定动画播放的时间。

基本语法格式如下：

```
animation-duration: <time>;
```

其中，<time>用于指定对象动画的持续时间，该属性值的单位为秒或毫秒。

4. 动画速度曲线 animation-timing-function

animation-timing-function 属性用于规定动画的速度曲线。

animation-timing-function 属性可接收以下值。

(1) ease：指定从慢速开始，然后加快，然后缓慢结束的动画(默认)。

(2) linear：规定从开始到结束的速度相同的动画。

(3) ease-in：规定慢速开始的动画。

(4) ease-out：规定慢速结束的动画。

(5) ease-in-out：指定开始和结束较慢的动画。

5. 动画之外状态 animation-fill-mode

animation-fill-mode 属性指在不播放动画时(在开始之前、结束之后或两者都结束时)，animation-fill-mode 属性规定目标元素的样式。

基本语法格式如下：

```
animation-fill-mode: none | forwards | backwards | both ;
```

各参数的解释如下。

(1) none：默认值，动画在执行之前或之后不会对元素应用任何样式。

(2) forwards：元素将保留由最后一个关键帧设置的样式值(依赖 animation-direction 和 animation-iteration-count)。

(3) backwards：元素将获取由第 1 个关键帧设置的样式值(取决于 animation-direction)，并在动画延迟期间保留该值。

(4) both：动画会同时遵循向前和向后的规则，从而在两个方向上扩展动画属性。

6. 动画延迟时间 animation-delay

animation-delay 属性用于规定动画开始的延迟时间。

基本语法格式如下：

```
animation-delay: <time> ;
```

其中，<time>用于指定对象动画延迟的时间，该属性值的单位为秒或毫秒。

7. 动画循环次数 animation-iteration-count

animation-iteration-count 属性用于设置动画的循环播放次数。

基本语法格式如下：

```
animation - iteration - count: infinite | < number >;
```

(1) infinite：无限循环。

(2) < number >：指定对象动画的具体循环次数。

8. 动画运动方向 animation-direction

animation-direction 属性用于指定是向前播放、向后播放还是交替播放动画，基本语法格式如下：

```
animation - direction: normal | reverse | alternate | alternate - reverse;
```

各参数的解释如下。

(1) normal：动画正常播放(向前)，默认值。

(2) reverse：动画以反方向播放(向后)。

(3) alternate：动画先向前播放，然后向后，并持续交替。

(4) alternate-reverse：动画先向后播放，然后向前，并持续交替。

9. 动画行为状态 animation-play-state

animation-play-state 属性用于设置动画的运行状态。

基本语法格式如下：

```
animation - play - state:running | paused ;
```

running：运动。paused：暂停。

10. 动画复合属性 animation

animation 属性用于一次性指定所有的动画设置要求，是一个复合属性。

基本语法顺序如下：

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

参数之间用空格隔开。

注意：若只提供一个时间参数，其值默认赋予 animation-duration 属性。

下面使用 6 种属性动画效果，代码如下：

```
div {
  animation-name: example;
  animation-duration: 5s;
  animation-timing-function: linear;
  animation-delay: 2s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}

@keyframes example {
  0%   {background-color:red; left:0px; top:0px;}
  50%  {background-color:blue; left:200px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
```

使用简写的 animation 属性也可以实现与上例相同的动画效果，代码如下：

```
div {
  animation: example 5s linear 2s infinite alternate;
}
```

使用 animation 动画相关属性实现心跳动画效果，如例 5-14 所示。

【例 5-14】 心跳

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>心跳</title>
    <style>
        img{
           width: 310px;
           height: auto;
          /* animation:动画名称、花费时间、运动曲线 */
           animation: heart 0.5s infinite;
        }
        @keyframes heart{
           0%{
               transform: scale(1);
           }
           50%{
               transform: scale(1.1);
           }
           100%{
               transform: scale(1);
```

```
            }
        }
    </style>
</head>
<body>
    <img src="images/xintiao.png" width="310" alt="loading" />
    <div></div>
</body>
</html>
```

在浏览器中的显示效果如图 5-26 所示。

图 5-26　持续心跳

5.4　响应式

响应式布局可以看作流式布局和自适应布局设计理念的融合。其目标是确保一个页面在所有终端上(各种尺寸的 PC、iPad、手机)都能显示出令人满意的效果,搭配媒体查询技术分别为不同的屏幕分辨率定义布局; 同时,在每个布局中,应用流式布局的理念,即页面元素宽度随着窗口调整而自动适配。

响应式布局,简而言之,就是一个网站能够兼容多个终端——而不是为每个终端开发一个特定的版本,如图 5-27 所示。这个概念是为解决移动互联网浏览而诞生的。响应式布局可以为不同终端的用户提供更加舒适的界面和更好的用户体验,而且随着目前大屏幕移动设备的普及,越来越多的网站采用此技术。

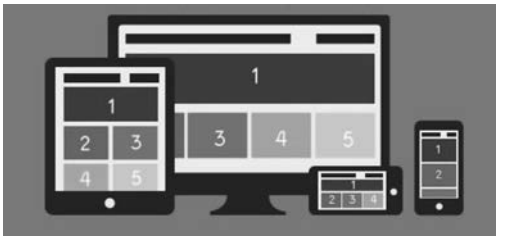


图 5-27　不同终端布局显示效果

5.4.1　媒体查询

媒体查询可以针对不同的屏幕尺寸设置不同的样式，它为每种类型的用户提供了最佳的体验，网站在任何尺寸设置下都能有最佳的显示效果。

以@media 开头来表示这是一条媒体查询语句。@media 后接一个或者多个表达式，如果表达式为真，则应用样式。

基本语法格式如下：

```
@media  媒体类型  逻辑操作符  (媒体功能){
        /* CSS 样式 */
    }
```

1. 逻辑操作符

操作符 and、not 和 only 可以用来构建复杂的媒体查询。

(1) and 操作符用来把多个媒体属性组合起来，合并到同一条媒体查询中。只有当每个属性都为真时，这条查询的结果才为真。

(2) not 操作符用来对一条媒体查询的结果进行取反。

(3) only 操作符表示仅在媒体查询匹配成功的情况下应用指定样式。可以通过它让选中的样式在老式浏览器中不被应用。

注意：若使用了 not 或 only 操作符，则必须明确地指定一个媒体类型。

2. 媒体类型

媒体类型用于描述设备的类别，默认为 all 类型，如表 5-7 所示。

表 5-7　媒体类型

值	描　述
all	用于所有多媒体类型设备
print	用于打印机
screen	用于计算机屏幕、平板电脑、智能手机等
speech	用于屏幕阅读器

所有浏览器都支持值为 screen、print 及 all 的 media 属性。

3. 常用媒体功能

以下仅仅列举了一些可能稍微常用的媒体功能：

(1) height 用于定义输出设备中的页面可见区域高度。

(2) width 用于定义输出设备中的页面可见区域宽度。

(3) max-height 用于定义输出设备中的页面最大可见区域高度。

(4) max-width 用于定义输出设备中的页面最大可见区域宽度。

(5) min-height 用于定义输出设备中的页面最小可见区域高度。

(6) min-width 用于定义输出设备中的页面最小可见区域宽度。

根据屏幕尺寸(屏幕尺寸代表不同终端)自动调整背景颜色，如例 5-15 所示。

【例 5-15】 媒体查询应用

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>媒体查询应用</title>
    <style>
        /* pc 端 */
        @media screen and (min-width: 992px){
            body{
                background-color: red;            /* 当屏幕尺寸大于 992px 时背景为红色 */
            }
        }
        /* ipad 端 */
        @media screen and (max-width: 992px) and (min-width: 768px){
            body{
                background-color: orange;    /* 屏幕尺寸大小在 768～992px 时背景为橘黄色 */
            }
        }
        /* 移动端 */
        @media screen and (max-width: 768px){
            body{
                background-color: blue;           /* 当屏幕尺寸小于 768px 时背景为蓝色 */
            }
        }
    </style>
</head>
<body>
</body>
</html>
```

在浏览器中的显示结果：不断拖拉改变浏览器窗口的大小，背景颜色也会随之发生改变。当屏幕尺寸大于 992px 时背景为红色；屏幕尺寸大小在 768～992px 时背景为橘黄色；

当屏幕尺寸小于768px时背景为蓝色。

5.4.2 响应式布局

响应式布局即根据浏览器或屏幕的大小，调整页面的布局方式。通俗地讲，就是通过一套代码，可以无缝匹配符合计算机、平板电脑、手机预览效果的前端技术，如例5-16所示。

虽然响应式布局应用越来越广泛，但是从零开始去设计一个响应式效果的网站对于程序员来讲是非常复杂的，因为当中包含了大量的逻辑、判断、适配内容，所以如今市面上看见的响应式网站，多数使用了一些开源的代码或者框架，而应用最广泛的是Bootstrap(将在第10章讲解)。

【例5-16】 响应式布局

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>响应式布局</title>
    <style>
        .box{
            margin: 0 auto;
        }
        .box>div{
            width: 229px;
            height: 274px;
            background-color: pink;
            float: left;
            margin-right: 10px;
            margin-bottom: 10px;
        }
        /*PC端*/
        @media screen and (min-width: 992px){
            .box{
                width: 946px;
            }
            .box>div:last-child{
                margin-right: 0;
            }
        }
        /*ipad端*/
        @media screen and (min-width: 768px) and (max-width: 992px){
            .box{
                width: 468px;
            }
            .box>div:nth-child(2),.box>div:nth-child(4){
                margin-right: 0;
            }
```

```
        }
        /* 移动端 */
        @media screen and (max-width: 768px){
            .box{
                width: 307px;
            }
            .box>div{
                width: 307px;
                height: 256px;
                margin-right: 0;
            }
        }
    </style>
</head>
<body>
    <div class="box">
        <div></div>
        <div></div>
        <div></div>
        <div></div>
    </div>
</body>
</html>
```

在浏览器中的显示效果如图 5-28 所示。

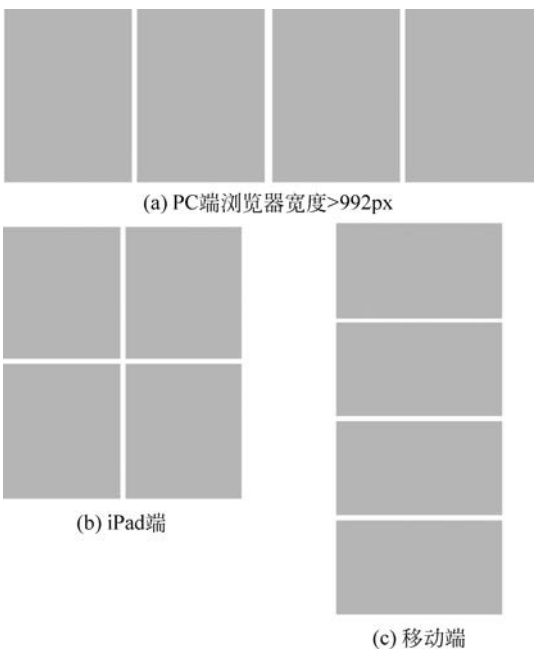


图 5-28　响应式布局在不同终端的显示效果

5.4.3 多列

CSS3 中新出现的多列布局是传统 HTML 网页中块状布局模式的有力扩充。这种新语法能够帮助 Web 开发人员轻松地让文本呈现多列显示的效果。它的显示如同 Word 中的多列，如图 5-29 所示。

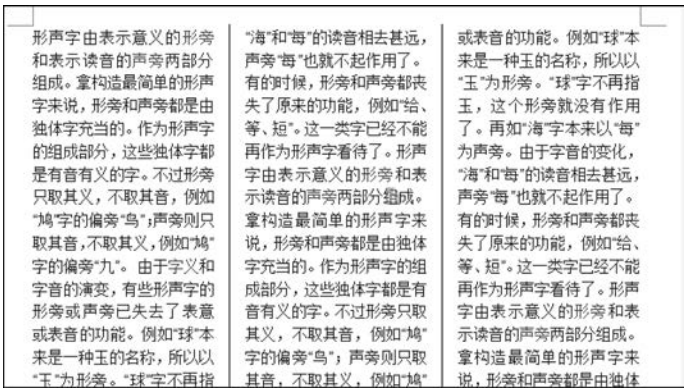

形声字由表示意义的形旁和表示读音的声旁两部分组成。拿构造最简单的形声字来说，形旁和声旁都是由独体字充当的。作为形声字的组成部分，这些独体字都是有音有义的字。不过形旁只取其义，不取其音，例如"鸠"字的偏旁"鸟"；声旁则只取其音，不取其义，例如"鸠"字的偏旁"九"。由于字义和字音的演变，有些形声字的形旁或声旁已失去了表意或表音的功能。例如"球"本来是一种玉的名称，所以以"王"为形旁。"球"字不再指"海"和"每"的读音相去甚远，声旁"每"也就不起作用了。有的时候，形旁和声旁都丧失了原来的功能，例如"给、等、短"。这一类字已经不能再作为形声字看待了。形声字由表示意义的形旁和表示读音的声旁两部分组成。拿构造最简单的形声字来说，形旁和声旁都是由独体字充当的。作为形声字的组成部分，这些独体字都是有音有义的字。不过形旁只取其义，不取其音，例如"鸠"字的偏旁"鸟"；声旁则只取其音，不取其义，例如"鸠"或表音的功能。例如"球"本来是一种玉的名称，所以以"王"为形旁。"球"字不再指王，这个形旁就没有作用了。再如"海"字本来以"每"为声旁。由于字音的变化，"海"和"每"的读音相去甚远，声旁"每"也就不起作用了。有的时候，形旁和声旁都丧失了原来的功能，例如"给、等、短"。这一类字已经不能再作为形声字看待了。形声字由表示意义的形旁和表示读音的声旁两部分组成。拿构造最简单的形声字来说，形旁和声旁都是由独体

图 5-29 多列效果

多列布局的相关属性，如表 5-8 所示。

表 5-8 CSS3 多列相关属性

属 性 名	描 述
columns	设置列数和每列的宽度，复合属性
column-width	设置每列的宽度
column-count	设置列数
column-gap	设置列与列之间的间隙
column-rule	设置列与列之间的边框，复合属性
column-rule-width	设置列与列之间的边框厚度
column-rule-style	设置列与列之间的边框样式
column-rule-color	设置列与列之间的边框颜色
column-span	设置元素是否横跨所有列

1. columns

columns 是一个复合属性，用于设置目标元素的列数和每列的宽度。

基本语法格式如下：

```
columns: <column-width><column-count>
```

(1) column-width：设置每列的宽度。

(2) column-count：设置分割的列数。

这两个属性默认情况下均设置为“自动”,这两个参数也可单独使用。

例如,将文本分为 3 列,每列宽度为 20 像素,代码如下:

```
div{columns: 20px 3;}
```

2. column-gap

column-gap 属性用于规定列与列之间的间隔。

基本语法格式如下:

```
column-gap: <length>| normal
```

其中,length 用长度值来定义列与列之间的间隙,不允许负值。默认值 normal 表示根据 font-size 的值自动分配同等长度值的距离。假设该对象的 font-size 为 16px,则 normal 值为 16px。

例如,指定列之间的间距为 40 像素,代码如下:

```
div { column-gap: 40px;}
```

3. column-rule

column-rule 是复合属性,用于设置列与列之间分割线的宽度、样式和颜色。要启用此功能,必须指定列间隔(column-gap)宽度,而且列间隔宽度与列规则相同或更大。

基本语法格式如下:

```
column-rule: <column-rule-width><column-rule-style><column-rule-color>
```

各参数的解释如下。

(1) column-rule-width:设置列与列之间分割线的宽度。

(2) column-rule-style:设置列与列之间分割线的样式,如实线 solid、虚线 dashed。

(3) column-rule-color:设置列与列之间分割线的颜色。

例如,实现列与列之间分割线为 1 像素宽的红色实线效果,代码如下:

```
div { column-rule: 1px solid red;}
```

4. column-span

column-span 属性表示是否要跨越所有列。

基本语法格式如下:

```
column-span: none | all
```

其中，none 表示不跨列，all 表示横跨所有列。

例如，< h2 > 元素跨所有列，代码如下：

```
h2 { column - span: all;}
```

CSS 多列布局允许我们轻松地定义多列文本，实现报纸、新闻排版效果，如例 5-17 所示。

【例 5-17】 多列布局

```
<!DOCTYPE html >
< html lang = "en">
< head >
    < meta charset = "UTF - 8">
    < title > 多列布局</title >
    < style >
        .box{
           column - count: 3;                          /* 3 列 */
           column - gap: 40px;                         /* 间隔 40px */
           column - rule: 3px dashed lightblue;        /* 分割线设置 */
        }
        h2 {
           column - span: all;                         /* 跨越所有行 */
        }
    </style >
</head >
< body >
    < div class = "box">
    < h2 >用好三千亿 支小再发力</h2 >
        “稳增长、保就业，重在保市场主体特别是量大面广的中小微企业”。为加大对中小微企业的帮扶政策力度，近日召开的国务院常务会议提出，今年再新增 3000 亿元支小再贷款额度，支持地方法人银行向小微企业和个体工商户发放贷款。更振奋人心的是，这新增的 3000 亿元支小再贷款额度将在今年年内发放完毕，并采取“先贷后借”模式，保障资金使用的精准性和直达性。
        当前，我国经济持续恢复增长，发展动力进一步增强，但经济恢复仍然不稳固、不均衡。中小微企业在促进经济增长、增加就业等方面具有重要意义。中小微企业发展仍然面临原材料价格居高不下、应收账款增加、疫情灾情影响等诸多难题。助力中小微企业和困难行业持续恢复，离不开金融活水的浇灌，其中，发挥好再贷款、再贴现和直达实体经济货币政策工具的牵引带动作用，显得尤为必要。
        再贷款是由中央银行贷款给商业银行，再由商业银行贷给普通客户的资金。由于再贷款提供的资金稳定、成本低、期限长，不仅能够降低相关行业企业的资金成本，也能有效分散银行承担的风险，从而激发银行服务企业的积极性，对相关地区地方法人金融机构贷款的撬动效果明显，而“先贷后借”模式，即由商业银行针对符合再贷款使用范围的企业先行发放贷款，事后凭发放清单等资料向人民银行申请再贷款资金。
    </div >
</body >
</html >
```

在浏览器中的显示效果如图 5-30 所示。

图 5-30　多列实现放报纸效果

5.5　CSS3 高级技巧

5.5.1　字体图标

在开发中经常会使用各种图标，为了节省资源，可能不会自己设计需要的图标，这时字体图标(iconfont)是很好的选择。

字体图标的优点如下：

(1) 可以实现跟图片一样的效果，如改变透明度、旋转度等。

(2) 本质上仍是文字，可以很随意地改变颜色、产生阴影、实现透明效果等。

(3) 本身体积更小，但携带的信息并没有削减。

(4) 支持大部分浏览器。

(5) 移动端设备的必备“良药”。

iconfont 字体图标的使用流程如下：

(1) 首先，进入阿里的向量图标库(http://www.iconfont.cn/)，在这个图标库里面可以找到很多图片资源，当然需要登录才能下载或者使用，用 GitHub 账号或者新浪微博账号登录都可以。

(2) 登录以后，找到图标库，搜索一个想要的图标，如“删除”图标，然后添加到购物车，如图 5-31 所示。

(3) 根据自己的需求选择图标，这里选择添加入库(将需要的图标全部添加到购物车)，操作完后可以看到图标已经添加进右上角的购物车里了，如图 5-32 所示。

(4) 在实际工作中，一般将购物车中的图标添加至项目，建立一个自己的图标库，将图标整合在一起，方便后续应用在自己的实际项目中，如图 5-33 所示。

图 5-31 删除字体图标

图 5-32 将图标添加到购物车

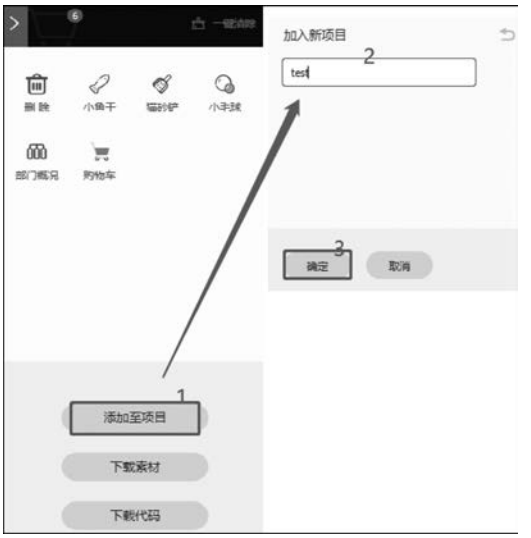


图 5-33 将图标添加至项目

(5) 进入“我的项目”中,将字体文件下载到本地,如图 5-34 所示。

图 5-34 将字体图标下载到本地

(6) 将整个文件夹添加到项目中,在项目中引用文件中的 iconfont.css 文件,如例 5-18 所示。

【例 5-18】 字体图标应用

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <!-- 引入 iconfont.css 字体图标样式 -->
    <link rel="stylesheet" href="iconfont/iconfont.css"/>
    <title>字体图标</title>
</head>
<body>
<ul>
    <!-- iconfont 是默认类型 icon-shanchu 是图标对应的(Font class)类名 -->
    <li><i class="iconfont icon-shanchu"></i>删除</li>
    <li><i class="iconfont">&#xe612;</i>删除</li>
    <li><i class="iconfont">&#xe708;</i>购物车</li>
```

```
    <li><i class="iconfont">&#xe624;</i>小鱼干</li>
</ul>
</body>
</body>
</html>
```

在浏览器中的显示效果如图 5-35 所示。

图 5-35 字体图标应用效果

代码解释：

是字体编码，可在下载的 demo.html 文件中查看，或者可以在阿里向量图标库的网站上查看。

FontClass 是 Unicode 使用方式的一种变种，相比于 Unicode 语意更明确，书写更直观。可以很容易分辨这个图标是什么。当然两种方式的效果是相同的。

5.5.2 雪碧图

CSS 雪碧图，即 CSS Sprite，也有人叫它 CSS 精灵图，是一种图像拼合技术。该技术是将多个小图标和背景图像合并到一张图片上，如图 5-36 所示，然后利用 CSS 的背景定位来显示需要显示的图片部分。

图 5-36 雪碧图

CSS 雪碧图的制作方式如下：

(1) PS 手动拼图。

(2) 使用 Sprite 工具自动生成(CssGaga 或者 CssSprite.exe)。

雪碧图用来减少 HTTP 请求数量，加速内容显示。因为每请求一次，就会和服务器建立一次连接，而建立连接需要额外的时间。同时也解决了命名困扰，只需对一张图片命名，而非对数十张小图片命名。

使用雪碧图之前，需要知道雪碧图中各个图标的位置，如图 5-37 所示。

从上面的图片不难看出雪碧图中各小图标(icon)在整张雪碧图的起始位置，例如第一个图标(裙子)在雪碧图中的起始位置为(0,0)，第 2 个图标(鞋子)在雪碧图中的起始位置为(0,50)，第 3 个图标(足球)在雪碧图中的起始位置为(0,100)，依此类推可以得出各图标相对于雪碧图的起始位置。

以上面的雪碧图为例(实际雪碧图中各个小图标的起始位置和上面的展示图不同)用一个 Demo 来阐述它的使用方法，如例 5-19 所示。

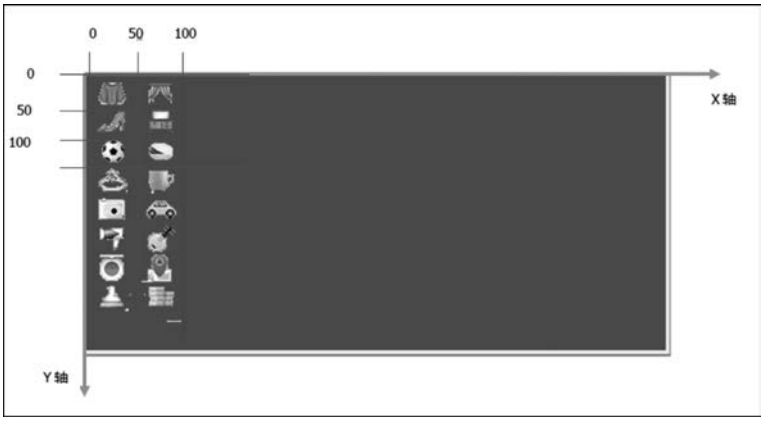


图 5-37　雪碧图坐标

【例 5-19】 雪碧图应用

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>雪碧图</title>
    <style>
        ul li {
            list-style: none;
            margin: 0;
            padding: 0;
        }
        a {
            color: #333;
            text-decoration: none;
        }
        .sidebar {
            width: 150px;
            border: 1px solid #ddd;
            background: #f8f8f8;
            padding: 0 10px;
            margin: 50px auto;
        }
        .sidebar li {
            border-bottom: 1px solid #eee;
            height: 40px;
            line-height: 40px;
            text-align: center;
        }
        .sidebar li a {font-size: 18px;}
```

```
        .sidebar li a:hover {color: #e91e63;}
        .sidebar li .spr-icon {
            display: block;
            float: left;
            height: 24px;
            width: 30px;
            background: url(images/css-sprite.png) no-repeat;        /*雪碧图*/
            margin: 8px 0px;}
        .sidebar li .icon2 { background-position: 0px -24px;}        /*定位雪碧图标*/
        .sidebar li .icon3 { background-position: 0px -48px;}
        .sidebar li .icon4 { background-position: 0px -72px;}
        .sidebar li .icon5 {background-position: 0px -96px;}
        .sidebar li .icon6 {background-position: 0px -120px;}
        .sidebar li .icon7 { background-position: 0px -144px;}
        .sidebar li .icon8 { background-position: 0px -168px;}
    </style>
</head>
<body>
<div>
    <ul class="sidebar">
        <li><a href=""><span class="spr-icon icon1"></span>服装内衣
            </a></li>
        <li><a href=""><span class="spr-icon icon2"></span>鞋包配饰
            </a></li>
        <li><a href=""><span class="spr-icon icon3"></span>运动户外
            </a></li>
        <li><a href=""><span class="spr-icon icon4"></span>珠宝手表
            </a></li>
        <li><a href=""><span class="spr-icon icon5"></span>手机数码
            </a></li>
        <li><a href=""><span class="spr-icon icon6"></span>家电办公
            </a></li>
        <li><a href=""><span class="spr-icon icon7"></span>护肤彩妆
            </a></li>
        <li><a href=""><span class="spr-icon icon8"></span>母婴用品
            </a></li>
    </ul>
</div>
</body>
</html>
```

在浏览器中的显示效果如图 5-38 所示。

上面的例子已经阐述了如何使用雪碧图，只不过初学者可能会对雪碧图中的background-position 属性值为负值有所疑惑。这个问题其实不难回答，细心的人应该很早就发现了使用负数的根源所在。此处用上面的 Demo 为例来分析这个问题。上面的 span 标签是一个 24px×30px 的容器，在使用背景图时，背景图的初始位置会从容器的左上角开

始铺满整个容器，然而容器的大小限制了背景图呈现的大小，超出容器部分被隐藏起来。假如设置 background-position：0 0，那么意味着，背景图像对于容器（span 标签）$X=0$；$Y=0$ 的位置作为背景图的起始位置来显示图片，所以如果需要在容器中显示第 2 个图标，意味着雪碧图 X 轴方向要向左移动，向左移动雪碧图时它的值会被设置为负数，同理 Y 轴方向也一样，如图 5-39 所示。

图 5-38　雪碧图应用效果

图 5-39　雪碧图定位原理

雪碧图的优点如下。

（1）加快网页加载速度：网页上的每幅图片都要向浏览器请求下载图片，而浏览器接受的同时请求数是 10 个，一次能处理的请求数目是两个。

（2）后期维护简单：该工具可以直接通过选择图片进行图片的拼接，当然也可以自己挪动里面的图片，自己去布局雪碧图，更换图片时也只要更改一下图片的位置就可以了。直接生成代码，简单易用。

（3）CSS Sprite 能减少图片的字节，笔者曾经比较过多次，将 3 张图片合并成 1 张图片的字节总是小于这 3 张图片的字节总和。

（4）解决了网页设计师在图片命名上的困扰，只需对一张集合的图片命名就可以了，不需要对每个小元素进行命名，从而提高了网页的制作效率。

（5）更换风格方便，只需要在一张或几张图片上修改图片的颜色或样式，整个网页的风格就可以改变。维护起来更加方便。

雪碧图的缺点如下：

（1）在图片合并时，要把多张图片有序且合理地合并成一张图片，还要留好足够的空间，防止板块内出现不必要的背景；这些还好，最痛苦的是在宽屏且高分辨率的屏幕下的自适应页面，如果图片不够宽，则很容易出现背景断裂。

（2）至于可维护性，这是一把双刃剑。可能有人喜欢，有人不喜欢，因为每次的图片改动都得在图片中删除或添加内容，显得稍微烦琐，而且算图片的位置（尤其是这种上千像素的图）也是一件颇为不爽的事情。当然，在追求性能的口号下，这些都是可以克服的。

（3）由于图片的位置需要固定为某个绝对数值，这就失去了诸如 center 之类的灵活性。

CSS Sprite 一般只能用到固定大小的盒子(box)里,这样才能遮挡住不应该看到的部分。也就是说,在一些需要非单向的平铺背景和需要网页缩放的情况下,CSS Sprite 并不合适。

下面使用雪碧图拼出自己的名字,如例 5-20 所示。

【例 5-20】 拼自己的名字

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>拼自己的名字</title>
    <style>
        span {/*引入雪碧图*/
           background: url(images/abcd.jpg) no-repeat;
           float: left;
        }
        span:first-child {
           width: 108px;
           height: 109px;
           background-position: 0 -9px;
        }
        span:nth-child(2) {
           width: 110px;
           height: 113px;
           background-position: -256px -275px;
        }
        span:nth-child(3) {
           width: 97px;
           height: 108px;
           background-position: -363px -7px;
        }
        span:nth-child(4) {
           width: 110px;
           height: 110px;
           background-position: -366px -556px;
        }
    </style>
</head>
<body>
    <span></span>
    <span></span>
    <span></span>
    <span></span>
</body>
</html>
```

在浏览器中的显示效果如图 5-40 所示。

图 5-40　名字效果

5.5.3　滑动门

美观的工艺，真正灵活的接口组件，并根据文本自适应大小，我们可用两个独立的背景图像来创造它。一个在左边，一个在右边。把这两幅图像想象成两扇可滑动的门，它们滑到一起并交叠，占据一个较窄的空间，或者相互滑开，占据一个较宽的空间，如图 5-41 所示。

图 5-41　推拉门

制作网页时，为了美观，常常需要为网页元素设置特殊形状的背景，例如微信导航栏，有凸起和凹下去的感觉，最大的问题是里面的字数不一样多，如图 5-42 所示。

图 5-42　微信导航栏

为了使各种特殊形状的背景能够自适应元素中文本内容的多少，出现了 CSS 滑动门技术。它从新的角度构建页面，使各种特殊形状的背景能够自由拉伸滑动，以适应元素内部的文本内容，可用性更强。此技术常见于各种导航栏的滑动门。

滑动门的核心技术就是利用 CSS 精灵（主要是背景位置）和盒子 padding 撑开宽度，以便能适应不同字数的导航栏。

一般的经典布局的代码如下：

```
<li>
    <a href="#">
        <span>导航栏内容</span>
    </a>
</li>
```

布局解释：

(1) a 设置背景左侧，a 标签只指定高度，而不指定宽度，padding 撑开合适宽度。

(2) span 设置背景右侧，padding 撑开合适宽度，剩下由文字继续撑开宽度。

(3) a 标签包含 span 标签是因为整个导航都是可以单击跳转的。

滑动门的工作原理如例 5-21 所示。

【例 5-21】 滑动门原理

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>滑动门原理</title>
    <style>
        * {
            margin: 0;
            padding: 0;
        }
        a {
            margin: 10px;
            display: inline-block;
            height: 33px;
            /* 不能设置宽度，我们要推拉门，自由缩放 */
            background: url(images/ao.png) no-repeat;
            padding-left: 15px;
            color: #fff;
            text-decoration: none;
            line-height: 33px;
        }
        a span {
            display: inline-block;
            height: 33px;
            background: url(images/ao.png) no-repeat right;
        /* span 不能给宽度，利用 padding 挤开，要 span 右边的圆角，所以背景位置右对齐 */
            padding-right: 15px;
        }
    </style>
</head>
<body>
    <a href="#">
```

```
        <span>首页</span>
    </a>
    <a href="#">
        <span>公众号</span>
    </a>
    <a href="#">
        <span>贝西奇谈</span>
    </a>
</body>
</html>
```

在浏览器中的显示效果如图 5-43 所示。

图 5-43　滑动门原理效果

自己实现微信官网导航栏效果，如例 5-22 所示。

【例 5-22】 微信导航栏

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>微信导航栏</title>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        ul{
            list-style: none;
        }
        body{
            background: url("images/wrap.jpg") repeat-x;
        }
        .nav{
            height: 75px;
        }
        .nav li a{
            /*1. a 左边放，左圆角，但是文字需要往右移 15px*/
```

```
            display: block;
            background: url("images/to.png") no-repeat;
            padding-left: 15px;
            color: #fff;
            font-size: 14px;
            line-height: 33px;
            text-decoration: none;
        }
        .nav li a span{
            /* 2. span 右边放,右圆角,但是文字需要往左移 15px */
            display: block;
            line-height: 33px;
            background: url("images/to.png") no-repeat right center;
            padding-right: 15px;
        }
        /* 凹下去,第 1 个为左边,第 2 个为右边 */
        .nav li a:hover{
            background-image: url("images/ao.png") ;
        }
        .nav li a:hover span{
            background-image: url("images/ao.png") ;
        }
        .nav li{
            float: left;
            margin: 0 10px;
            padding-top: 21px;
        }
    </style>
</head>
<body>
    <div class="nav">
        <ul>
            <li>
                <a href="#">
                <span>首页</span>
                </a>
            </li><li>
            <a href="#">
                <span>帮助与反馈</span>
            </a>
        </li><li>
            <a href="#">
                <span>公众平台</span>
            </a>
        </li><li>
            <a href="#">
                <span>开放平台</span>
            </a>
        </li><li>
```

```
            <a href="#">
                <span>微信支付</span>
            </a>
        </li><li>
            <a href="#">
                <span>微信网页版</span>
            </a>
        </li><li>
            <a href="#">
                <span>表情开发平台</span>
            </a>
        </li><li>
            <a href="#">
                <span>微信广告</span>
            </a>
        </li>
        </ul>
    </div>
</body>
</html>
```

在浏览器中的显示效果如图 5-44 所示。

图 5-44　微信导航栏效果