第 5 章

CHAPTER 5

CSS 高级应用

5.1 CSS 布局

5.1.1 案例描述

浏览淘宝、京东等购物网站或者新浪、腾讯等新闻网页时，网页中板块分明，内容分割明确，参见图 5-1。本节将利用布局等方式来实现如图 5-1 所示的效果。

图 5-1 案例描述

5.1.2 知识引入

1. CSS 盒模型

1) CSS 盒模型概述

盒模型是 CSS 定位布局的核心内容，它指定元素如何显示以及如何相互交互。页面上的每个元素都被看作为一个矩形框，这个框由元素的内容、内边距、边框和外边距组成。HTML 中大部分的元素（特别是块状元素）都可以看作一个盒子，网页元素的定位实际就是

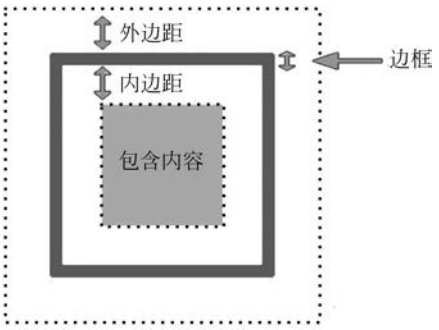


图 5-2　盒模型示意图

这些大大小小的盒子在页面中的定位。这些盒子在页面中是“流动”的，当某个块状元素被 CSS 设置了浮动属性，那么这个盒子就会“流”到上一行。网页布局即关注这些盒子在页面中如何摆放、如何嵌套的问题。这么多盒子摆在一起，最需要关注的是盒子尺寸计算、是否流动等要素，如图 5-2 所示。

2）CSS 内边距

内边距(padding)出现在内容区域的周围。如果在元素上添加背景，那么背景应用于元素的内容和内边距组成的区域。因此可以用内边距在内容周围创建一个隔离带，使内容不与背景混合在一起。当元素的内边距被清除时，所“释放”的区域将会被元素背景颜色填充，取值方式如表 5-1 所示。

表 5-1　内边距属性取值

值	说　明
length	定义一个固定的填充(px、pt、em 等)
%	使用百分比值定义一个填充

在 CSS 中，它可以指定在不同的侧面不同的填充效果：

```
padding-top:25px;
padding-bottom:25px;
padding-right:50px;
padding-left:50px;
```

为了缩短代码，可以在一个属性中指定所有的填充属性。

这就是所谓的缩写属性。所有的内边距属性的缩写属性是"padding"，可以有 1～4 个值。

(1) padding：25px 50px 75px 100px。

- 上填充为 25px。
- 右填充为 50px。
- 下填充为 75px。
- 左填充为 100px。

(2) padding：25px 50px 75px。

- 上填充为 25px。
- 左右填充为 50px。
- 下填充为 75px。

(3) padding：25px 50px。

- 上下填充为 25px。
- 左右填充为 50px。

(4) padding：25px。

所有的填充都是 25px。

示例代码如下所示：

```
<html>
<head>
<title>内边距</title>
<style>
p
{
background-color:yellow;
}
p.padding
{
padding-top:50px;
padding-bottom:50px;
padding-right:50px;
padding-left:50px;
}
p.paddings
{
padding:25px;
}
</style>
</head>
<body>
<p>这是一个没有指定填充边距的段落。</p>
<p class="padding">这是一个指定填充边距的段落。</p>
<p class="paddings">这是一个指定填充边距的段落。</p>
</body>
</html>
```

通过 IE 查看该 HTML，内边距效果如图 5-3 所示。

图 5-3　内边距效果

3) CSS 边框

CSS 边框属性允许指定一个元素边框的样式和颜色。

(1) 边框样式(borde-style)。

边框样式属性指定要显示什么样的边界，边框样式属性用来定义边框的样式。

边框样式属性演示如图 5-4 所示。

(2) 边框宽度(border-width)。

可以通过边框宽度属性为边框指定宽度。

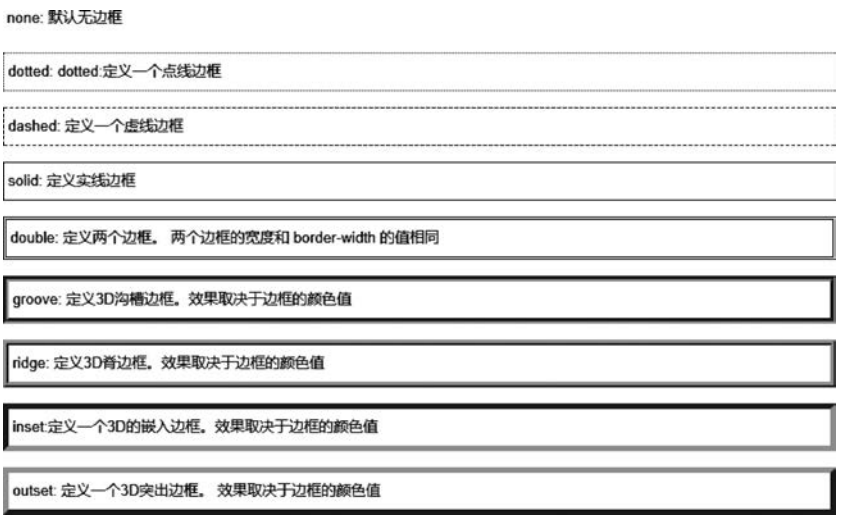


图 5-4　边框样式属性演示

为边框指定宽度有两种方法：可以指定长度值，比如 2px 或 0.1em（单位为 px、pt、cm、em 等）；或者使用 3 个关键字之一，分别是 thick、medium（默认值）和 thin。

注意：CSS 没有定义 3 个关键字的具体宽度，所以一个用户可能把 thick、medium 和 thin 分别设置为等于 5px、3px 和 2px，而另一个用户则分别设置为 3px、2px 和 1px。

（3）边框颜色（border-color）。

边框颜色属性用于设置边框的颜色。设置颜色的方式有以下 4 种：

name——指定颜色的名称，如"red"。

RGB——指定 RGB 值，如"rgb(255,0,0)"。

Hex——指定十六进制颜色值，如"#ff0000"。

还可以设置边框的颜色为"transparent"。

注意：border-color 单独使用是不起作用的，必须先使用 border-style 来设置边框样式。

（4）边框属性。

可以在一个属性中设置边框。

如在"border"属性中设置：

- border-width。
- border-style(required)。
- border-color。

示例代码如下所示：

```
<html>
<head>
<title>CSS 边框</title>
<style>
p
{
border:5px solid red;
text-align:center;
```

```
font-weight: bold;
}
</style>
</head>
<body>
<p>边框演示</p>
</body>
</html>
```

通过 IE 查看该 HTML，结果如图 5-5 所示。

图 5-5　边框属性演示

4）CSS 外边距

外边距(margin)属性定义元素周围的空间。外边距清除周围的元素外边区域。外边距没有背景颜色，是完全透明的外边距可以单独改变元素的上、下、左、右边距，也可以一次改变所有的属性。外边距属性取值方式如表 5-2 所示。

表 5-2　margin 取值

属性	说　明
length	定义一个固定的 margin(使用 px、pt、em 等)
%	定义一个使用百分比的边距
auto	设置浏览器边距。其结果依赖于浏览器

外边距可以使用负值进行页面内容的重叠，在 CSS 中，可以指定不同的侧面不同的边距。

```
margin-top:100px;
margin-bottom:100px;
margin-right:50px;
margin-left:50px;
```

为了缩短代码，可以使用“margin”表示所有外边距属性。这就是所谓的缩写属性。

所有外边距属性的缩写属性是"margin"：margin 属性可以有 1～4 个值。

(1) margin——25px 50px 75px 100px。

- 上边距为 25px。
- 右边距为 50px。
- 下边距为 75px。
- 左边距为 100px。

(2) margin——25px 50px 75px。

- 上边距为 25px。
- 左、右边距为 50px。

- 下边距为 75px。

(3) margin——25px 50px。

- 上下边距为 25px。
- 左右边距为 50px。

(4) margin——25px。

所有的 4 个边距都是 25px。

示例代码如下所示：

```
<html>
<head>
<title>CSS 外边距</title>
<style>
p{
background-color:yellow;
}
p.margin{
margin-top:50px;
margin-bottom:50px;
margin-right:30px;
margin-left:30px;
}
p.margin1{
margin:50px 30px;
}
p.margin2{
margin:auto 0px;
}
</style>
</head>
<body>
<p>没有指定边距大小的段落。</p>
<p class="margin">指定边距大小的段落。</p>
<p class="margin1">指定边距大小的段落。</p>
<p class="margin2">指定边距大小的段落。</p>
</body>
</html>
```

通过 IE 查看该 HTML，结果如图 5-6 所示。

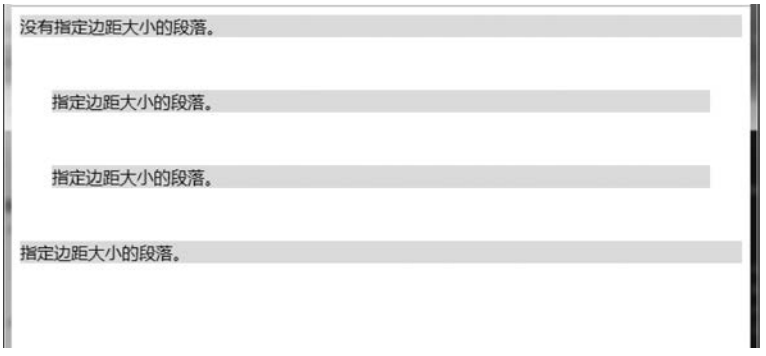


图 5-6　外边距属性示例

5）CSS 轮廓

轮廓(outline)是绘制于元素周围的一条线，可指定元素轮廓的样式、颜色和宽度。轮廓是指边框边缘，可起到突出元素的作用。表 5-3 定义了所有轮廓属性。

表 5-3 轮廓属性

属性	说 明	值
outline	在一个声明中设置所有的轮廓属性	outline-color outline-style outline- width inherit
outline-color	设置轮廓的颜色	color-name hex-number rgb-number invert inherit
outline-style	设置轮廓的样式	none dotted dashed solid double groove ridge inset outset inherit
outline-width	设置轮廓的宽度	thin medium thick length inherit

示例代码如下所示：

```
<html>
<head>
<title>CSS 轮廓</title>
<style>
p
{
border:1px solid red;
outline-style:dotted;
outline-color:#00ff00;
outline-width:3px;
margin-top:50px;
}
```

```
</style>
</head>
<body>
<p><b>注意:</b> 如果只有一个 !DOCTYPE 指定 IE 8 支持 outline 属性。</p>
</body>
</html>
```

通过 IE 查看该 HTML,CSS 轮廓效果如图 5-7 所示。

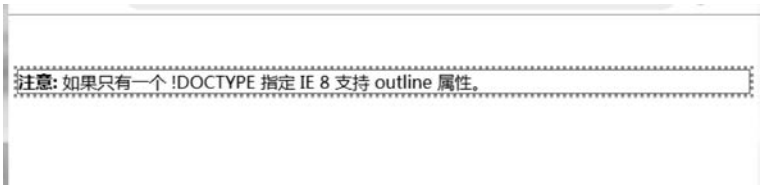


图 5-7　CSS 轮廓属性示例

2. 盒模型布局

1）盒模型显示类型

CSS 中盒模型(box model)分为两种：一种是 W3C 的标准模型,另一种是 IE 的传统模型,它们的相同之处是都是对元素计算尺寸的模型,具体来说,就是对元素的 width、height、padding、border 以及元素实际尺寸的计算；它们的不同之处是两者的计算方法不一致。

(1) W3C 的标准盒模型。

```
/*外盒尺寸计算(元素空间尺寸)*/
Element 空间高度 = content height + padding + border + margin
Element 空间宽度 = content width + padding + border + margin
/*内盒尺寸计算(元素大小)*/
Element Height = content height + padding + border (height 为内容高度)
Element Width = content width + padding + border (width 为内容宽度)
```

(2) IE 的传统盒模型。

```
/*外盒尺寸计算(元素空间尺寸)*/
Element 空间高度 = content height + margin (height 包含了元素内容宽度、边框宽度、内距宽度)
Element 空间宽度 = content width + margin (width 包含了元素内容宽度、边框宽度、内距宽度)
/*内盒尺寸计算(元素大小)*/
Element Height = content height(height 包含了元素内容宽度、边框宽度、内距宽度)
Element Width = content width(width 包含了元素内容宽度、边框宽度、内距宽度)
```

为了帮助大家理解,下面看一个实际的例子。比如现在有一个叫 boxtest 的 Div,它具有下面的属性：

```
.boxtest
{
width: 200px;
  height: 200px;
  border: 20px solid black;
```

```
padding: 50px;
margin: 50px;
}
```

W3C 标准下的盒模型，盒子的总宽度/高度＝width/height＋padding＋border＋margin。

IE 传统模式下的盒模型，盒子的总宽度和高度是包含内边距(padding)和边框(border)宽度在内的，盒子的总宽度/高度＝width/height＋margin＝内容区宽度/高度＋padding＋border＋margin。

也就是说，盒子宽高＝内容区域的宽高＋padding＋border。

可以看出，IE6 以下版本浏览器的宽度包含了元素的 padding 和 border 值，换句话说，在 IE6 以下版本中，内容真正的宽度是 width－padding－boder)。用内外盒来说，W3C 标准浏览器的内盒宽度等于 IE6 以下版本浏览器的外盒宽度。

2) CSS3 伸缩盒布局

CSS3 引入的布局模式伸缩盒布局(Flexbox)，主要思想是让容器有能力改变其子项目的宽度和高度，以最佳方式填充可用空间。伸缩盒布局使用 Flex 项目可以自动放大与收缩，用来填补可用的空闲空间。更重要的是，伸缩盒布局方向不可预知，不像常规的布局(块级从上到下、内联从左到右)，而那些常规的页面布局，对于大型或者复杂的应用程序就缺乏灵活性。

如果常规布局是基于块和内联文本流方向，那么伸缩盒布局就是基于 Flex-flow 方向。先来了解一下伸缩盒布局的一些专用术语。伸缩盒布局模型如图 5-8 所示。

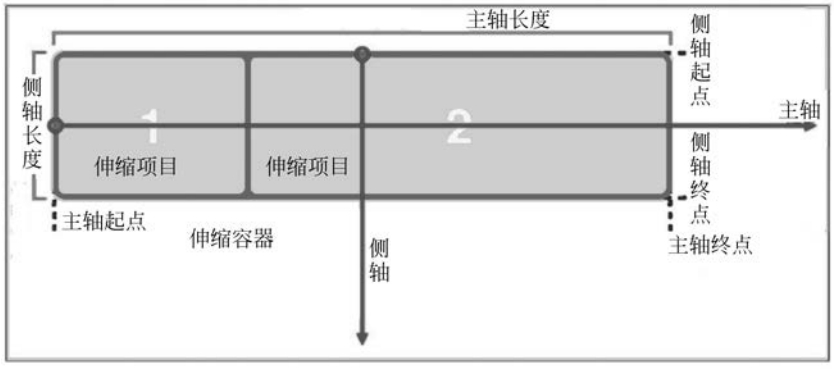


图 5-8　伸缩盒布局模型

主轴：flex 容器的主轴主要用来配置 Flex 项目。它不一定是水平的，这主要取决于 flex-direction 属性。

主轴起点、主轴终点：Flex 项目的配置从容器的主轴起点开始，往主轴终点结束。

主轴长度：Flex 项目在主轴方向的宽度或高度就是项目的主轴长度，Flex 项目的主轴长度属性是 width 或 height 属性，具体由对着主轴方向的那一个决定。

侧轴：与主轴垂直的轴称作侧轴，是侧轴方向的延伸。

侧轴起点、侧轴终点：伸缩行的配置从容器的侧轴起点开始，往侧轴终点结束。

侧轴长度：Flex 项目在侧轴方向的宽度或高度就是项目的侧轴长度，Flex 项目的侧轴

长度属性是 widht 或 height 属性，具体由对着主轴方向的那一个决定。

（1）flex 容器属性。

```
display: flex | inline - flex;
```

定义一个 flex 容器、内联或者根据指定的值，来作用于下面的子类容器：

- box：将对象作为弹性伸缩盒显示。
- inline-box：将对象作为内联块级弹性伸缩盒显示。
- flexbox：将对象作为弹性伸缩盒显示。
- inline-flexbox：将对象作为内联块级弹性伸缩盒显示。
- flex：将对象作为弹性伸缩盒显示。
- inline-flex：将对象作为内联块级弹性伸缩盒显示。

请注意：

① CSS 列（CSS columns）在弹性盒中不起作用。

② float、clear 和 vertical-align 在 Flex 项目中不起作用。

flex-direction（适用于父类容器的元素上）。

定义：设置或检索伸缩盒对象的子元素在父类容器中的位置。

```
flex - direction: row | row - reverse | column | column - reverse
```

- row：横向从左到右排列（左对齐），默认的排列方式。
- row-reverse：反转横向排列（右对齐），从后往前排，最后一项排在最前面。
- column：纵向排列。
- column-reverse：反转纵向排列，从后往前排，最后一项排在最上面。
- 若使 flex 生效，则需定义其父元素 display 为 flex 或 inline-flex（box 或 inline-box，这是旧的方式）。

示例代码如下所示：

```
<html>
<head>
<title>flex - direction 属性</title>
<style>
.box{
display: - webkit - flex;
display:flex;
    margin:0;padding:10px;list - style:none;background - color: # eee;}
.box li{width:100px;height:100px;border:1px solid # aaa;text - align:center;}
# box{
 - webkit - flex - direction:row;
 flex - direction:row;
}
# box2{
 - webkit - flex - direction:row - reverse;
 flex - direction:row - reverse;
}
```

```
#box3{
 height:500px;
  -webkit-flex-direction:column;
 flex-direction:column;
}
#box4{
 height:500px;
  -webkit-flex-direction:column-reverse;
 flex-direction:column-reverse;
}
</style>
</head>
<body>
<h3>flex-direction:row</h3>
<ul id="box" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>flex-direction:row-reverse</h3>
<ul id="box2" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>flex-direction:column</h3>
<ul id="box3" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>flex-direction:column-reverse</h3>
<ul id="box4" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
</body>
</html>
```

flex-direction 各部分效果如图 5-9～图 5-12 所示。

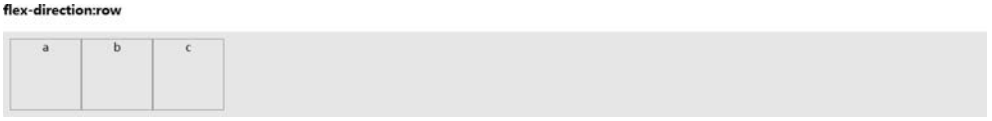


图 5-9 flex-direction：row 部分

(2) flex-wrap(适用于父类容器)。

设置或检索伸缩盒对象的子元素超出父类容器时是否换行。

flex-wrap：nowrap | wrap | wrap-reverse

- nowrap：当子元素溢出父类容器时不换行。

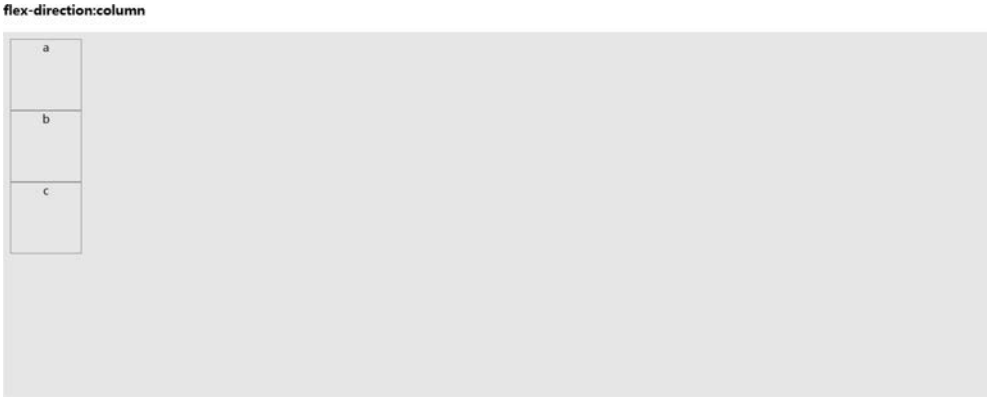


图 5-10　flex-direction：column 部分

图 5-11　flex-direction：row-reverse 部分

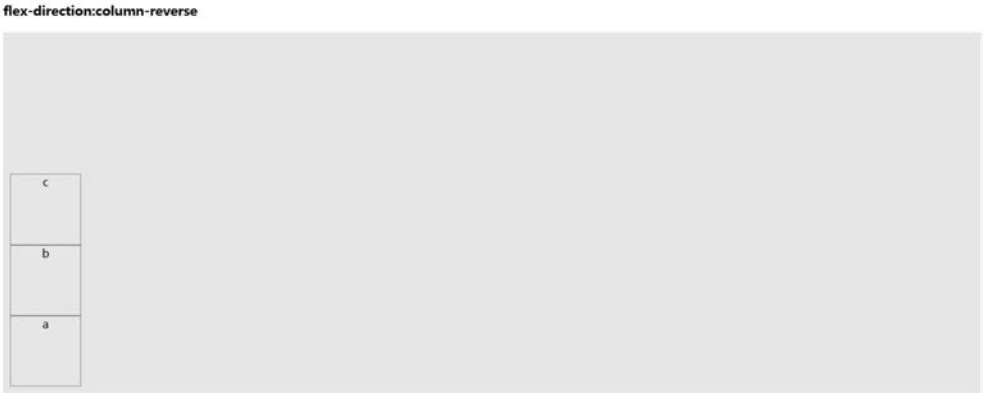


图 5-12　flex-direction：column-reverse 部分

- wrap：当子元素溢出父类容器时自动换行。
- wrap-reverse：当子元素溢出父类容器时自动换行并倒序排列。

示例代码如下所示：

```
<html>
<head>
<title>flex-wrap 使用示例</title>
<style>
.box{
display:-webkit-flex;
display:flex;
    width:220px;margin:0;padding:10px;list-style:none;background-color:red;}
.box li{width:80px;height:80px;border:1px solid #aaa;text-align:center;}
```

```
#box{
-webkit-flex-wrap:nowrap;
flex-wrap:nowrap;
}
#box2{
 -webkit-flex-wrap:wrap;
 flex-wrap:wrap;
}
#box3{
 -webkit-flex-wrap:wrap-reverse;
 flex-wrap:wrap-reverse;
}
</style>
</head>
<body>
<h3>flex-wrap:nowrap</h3>
<ul id="box" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>flex-wrap:wrap</h3>
<ul id="box2" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>flex-wrap:wrap-reverse</h3>
<ul id="box3" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
</body>
</html>
```

通过 Chrome 查看该 HTML，flex-wrap 使用效果如图 5-13 所示。

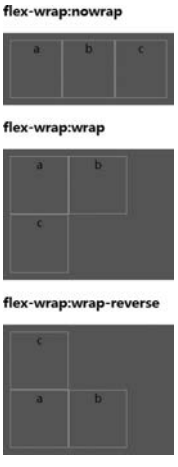


图 5-13 flex-wrap 的显示效果

(3) flex-flow(适用于父类容器)。

flex-flow 是复合属性，设置或检索伸缩盒对象的子元素排列方式。

```
flex-flow: <'flex-direction'> || <'flex-wrap'>
```

- flex-direction：定义弹性盒子元素的排列方向。
- flex-wrap：定义弹性盒子元素溢出父类容器时是否换行。

示例代码如下所示：

```
<html>
<head>
<title>flex-flow 示例</title>
<style>
.box{
display: -webkit-flex;
display:flex;
    width:220px;margin:0;padding:10px;list-style:none;background-color:yellow;}
.box li{width:80px;height:80px;border:1px solid #aaa;text-align:center;}
#box{
 -webkit-flex-flow:row nowrap;
 flex-flow:row nowrap;
}
#box2{
 -webkit-flex-flow:row wrap-reverse;
 flex-flow:row wrap-reverse;
}
#box3{
 height:220px;
 -webkit-flex-flow:column wrap-reverse;
    flex-flow:column wrap-reverse;}
</style>
</head>
<body>
<ul id="box" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>flex-flow:row wrap-reverse</h3>
<ul id="box2" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>flex-flow:column wrap-reverse;</h3>
<ul id="box3" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
</body>
</html>
```

通过 IE 查看该 HTML，flex-flow 效果如图 5-14 所示。

(4) justify-content (适用于父类容器)。

设置或检索弹性盒子元素在主轴(横轴)方向上的对齐方式。

当弹性盒子里一行上的所有子元素都不能伸缩或已经达到其最大值时，这一属性可协助对多余的空间进行分配。当元素溢出某行时，这一属性同样会在对齐格式上进行控制。

```
justify-content: flex-start | flex-end | center | space-between | space-around
```

flex-start：弹性盒子元素将向行起始位置对齐。该行的第一个子元素的主起始位置的边界将与该行的主起始位置的边界对齐，同时所有后续的伸缩盒项目与其前一个项目对齐。

flex-end：弹性盒子元素将向行结束位置对齐。该行的第一个子元素的主结束位置的边界将与该行的主结束位置的边界对齐，同时所有后续的伸缩盒项目与其前一个项目对齐。

center：弹性盒子元素将向行中间位置对齐。该行的子元素将相互对齐并在行中居中对齐，同时第一个元素与行的主起始位置的边距等同于最后一个元素与行的主结束位置的边距（如果剩余空间是负数，则保持两端相等长度的溢出）。

space-between：弹性盒子元素会平均地分布在各行中。如果最左边的剩余空间是负数，或该行只有一个子元素，则该值等效于'flex-start'。在其他情况下，第一个元素的边界与行的主起始位置的边界对齐，同时最后一个元素的边界与行的主结束位置的边距对齐，剩余的伸缩盒项目则平均分布，并确保两两之间的空白空间相等。

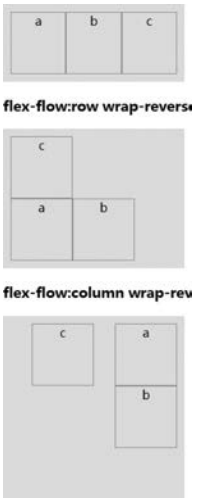


图 5-14　flex-flow 效果

space-around：弹性盒子元素会平均地分布在行中，两端保留子元素与子元素之间间距的一半。如果最左边的剩余空间是负数，或该行只有一个伸缩盒项目，则该值等效于'center'。在其他情况下，伸缩盒项目则平均分布，并确保两两之间的空白空间相等，同时第一个元素前的空间以及最后一个元素后的空间为其他空白空间的一半。

示例代码如下所示：

```
<html>
<head>
<title>justify-content 演示示例</title>
<style>
.box{
display:-webkit-flex;
display:flex;
    width:400px;height:100px;margin:0;padding:0;border-radius:5px;list-style:none;
background-color:yellow;}
.box li{margin:5px;padding:10px;border-radius:5px;background:red;text-align:center;}
#box{
 -webkit-justify-content:flex-start;
    justify-content:flex-start;}
#box2{
 -webkit-justify-content:flex-end;
 justify-content:flex-end;}
    #box3{
 -webkit-justify-content:center;
    justify-content:center;}
#box4{
 -webkit-justify-content:space-between;
    justify-content:space-between;}
#box5{
 -webkit-justify-content:space-around;
    justify-content:space-around;}
```

```
</style>
</head>
<body>
<ul id="box" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>justify-content:flex-end</h3>
<ul id="box2" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>justify-content:center</h3>
<ul id="box3" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>justify-content:space-between</h3>
<ul id="box4" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>justify-content:space-around</h3>
<ul id="box5" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
</body>
</html>
```

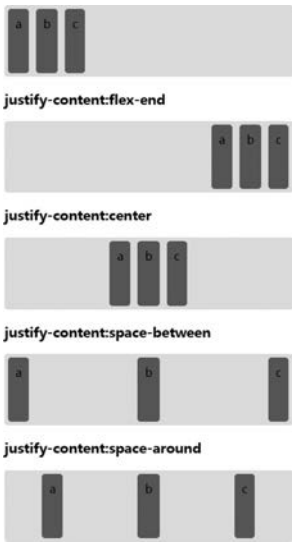


图 5-15　justify-content 效果

通过 IE 查看该 HTML，效果如图 5-15 所示。

(5) align-items（适用于父类容器上）。

设置或检索弹性盒子元素在侧轴（纵轴）方向上的对齐方式。

```
align-items: flex-start | flex-end | center | baseline | stretch
```

- flex-start：弹性盒子元素的侧轴（纵轴）起始位置的边界紧靠该行的侧轴（纵轴）起始边界。
- flex-end：弹性盒子元素的侧轴（纵轴）结束位置的边界紧靠该行的侧轴（纵轴）结束边界。
- center：弹性盒子元素在该行的侧轴（纵轴）上居中放置。（如果该行的尺寸小于弹性盒子元素的尺寸，则会向两个方向溢出相同的长度）。

- baseline：如弹性盒子元素的行内轴与侧轴为同一条，则该值与'flex-start'等效。其他情况下，该值将参与基线对齐。
- stretch：如果指定侧轴大小的属性值为'auto'，则其值会使项目的边距盒的尺寸尽可能接近所在行的尺寸，但同时会遵照'min/max-width/height'属性的限制。

示例代码如下所示：

```
<html>
<head>
<title>align-items</title>
<style>
.box{
display:-webkit-flex;
display:flex;
    width:200px;height:100px;margin:0;padding:0;border-radius:5px;list-style:none;
background-color:yellow;}
   .box li{margin:5px;border-radius:5px;background:red;text-align:center;}
   .box li:nth-child(1){padding:10px;}
   .box li:nth-child(2){padding:15px 10px;}
   .box li:nth-child(3){padding:20px 10px;}
#box{
 -webkit-align-items:flex-start;
 align-items:flex-start;
}
#box2{
 -webkit-align-items:flex-end;
 align-items:flex-end;
}
#box3{
 -webkit-align-items:center;
 align-items:center;
}
#box4{
 -webkit-align-items:baseline;
 align-items:baseline;
}
#box5{
 -webkit-align-items:strecth;
 align-items:strecth;
}
</style>
</head>
<body>
<h3>align-items:flex-start</h3>
<ul id="box" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>align-items:flex-end</h3>
<ul id="box2" class="box">
 <li>a</li>
```

```
<li>b</li>
<li>c</li>
</ul>
<h3>align-items:center</h3>
<ul id="box3" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>align-items:baseline</h3>
<ul id="box4" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
<h3>align-items:strecth</h3>
<ul id="box5" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
</body>
</html>
```

通过 Chrome 查看该 HTML，align-item 各个效果如图 5-16 所示。

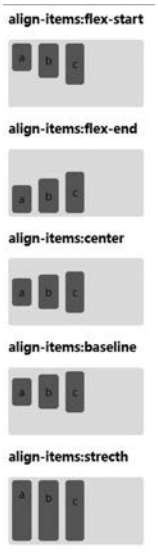


图 5-16　align-items 各个效果

（6）align-content（适用于父类容器）。

设置或检索弹性盒子堆叠伸缩行的对齐方式。

```
align-content: flex-start | flex-end | center | space-between | space-around | stretch
```

- flex-start：各行向弹性盒子容器的起始位置堆叠。弹性盒子容器中第一行的侧轴起始边界紧靠该弹性盒子容器的侧轴起始边界，之后的每一行都紧靠前面一行。
- flex-end：各行向弹性盒子容器的结束位置堆叠。弹性盒子容器中最后一行的侧轴起结束边界紧靠该弹性盒子容器的侧轴结束边界，之后的每一行都紧靠前面一行。
- center：各行向弹性盒子容器的中间位置堆叠。各行两两紧靠，同时在弹性盒子容器中居中对齐，保持弹性盒子容器的侧轴起始内容边界和第一行之间的距离与该容器的侧轴结束内容边界与最后一行之间的距离相等（如果剩下的空间是负数，则各行会向两个方向溢出相等的距离）。
- space-between：各行在弹性盒子容器中平均分布。如果剩余的空间是负数或弹性盒子容器中只有一行，则该值等效于'flex-start'。在其他情况下，第一行的侧轴起始边界紧靠弹性盒子容器的侧轴起始内容边界，最后一行的侧轴结束边界紧靠弹性盒子容器的侧轴结束内容边界，剩余的行则按一定方式在弹性盒子窗口中排列，以保持两两之间的空间相等。

- space-around：各行在弹性盒子容器中平均分布，两端保留子元素与子元素之间间距大小的一半。如果剩余的空间是负数或弹性盒子容器中只有一行，则该值等效于'center'。在其他情况下，各行会按一定方式在弹性盒子容器中排列，以保持两两之间的空间相等，同时第一行前面及最后一行后面的空间是其他空间的一半。
- stretch：各行将会伸展以占用剩余的空间。如果剩余的空间是负数，则该值等效于'flex-start'。在其他情况下，剩余空间被所有行平分，以扩大它们的侧轴尺寸。

示例代码如下所示：

```
<html>
<head>
<title>CSS 轮廓</title>
<style>
.box{
 display:-webkit-flex;
 display:flex;
  -webkit-flex-wrap:wrap;
 flex-direction:wrap;
 width:200px;height:200px;margin:0;padding:0;border-radius:5px;
list-style:none;background-color:#eee;}
.box li{margin:5px;padding:10px;border-radius:5px;background:#aaa;
text-align:center;}
#box{
  -webkit-align-content:flex-start;
 align-content:flex-start;
}
#box2{
  -webkit-align-content:flex-end;
 align-content:flex-end;
}
#box3{
  -webkit-align-content:center;
 align-content:center;
}
#box4{
  -webkit-align-content:space-between;
 align-content:space-between;
}
#box5{
  -webkit-align-content:space-around;
 align-content:space-around;
}
#box6{
  -webkit-align-content:strecth;
 align-content:strecth;
}
</style>
</head>
<body>
<h3>align-content:flex-start</h3>
<ul id="box" class="box">
 <li>a</li>
```

```
 <li>b</li>
 <li>c</li>
 <li>d</li>
 <li>e</li>
 <li>f</li>
</ul>
<h3>align-content:flex-end</h3>
<ul id="box2" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
 <li>d</li>
 <li>e</li>
 <li>f</li>
</ul>
<h3>align-content:center</h3>
<ul id="box3" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
 <li>d</li>
 <li>e</li>
 <li>f</li>
</ul>
<h3>align-content:space-between</h3>
<ul id="box4" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
 <li>d</li>
 <li>e</li>
 <li>f</li>
</ul>
<h3>align-content:space-around</h3>
<ul id="box5" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
 <li>d</li>
 <li>e</li>
 <li>f</li>
</ul>
<h3>align-content:strecth</h3>
<ul id="box6" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
 <li>d</li>
 <li>e</li>
 <li>f</li>
</ul>
</body>
</html>
```

通过 Chrome 查看该 HTML,CSS 轮廓样式效果如图 5-17 和图 5-18 所示。

图 5-17　CSS 轮廓(1)

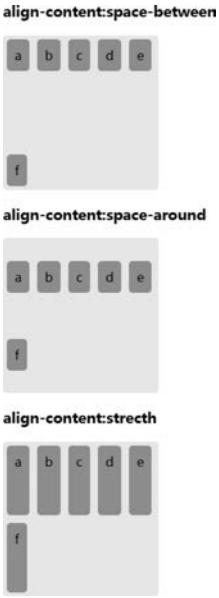


图 5-18　CSS 轮廓(2)

(7) order (适用于弹性盒子模型容器子元素)。

设置或检索弹性盒子模型对象的子元素出现的顺序。

```
order: < integer >
```

< integer >: 用整数值来定义排列顺序,数值小的排在前面。可以为负值。

示例代码如下所示:

```
< html >
< head >
< title > order </title >
< style >
.box{
 display: - webkit - flex;
 display:flex;
      margin:0;padding:10px;list - style:none;background - color:yellow;}
.box li{width:100px;height:100px;border:1px solid red;text - align:center;}
# box li:nth - child(3){
  - webkit - order: - 1;
 order: - 1;
}
</style >
</head >
< body >
< ul id = "box" class = "box">
 < li > a </li >
 < li > b </li >
```

```
 <li>c</li>
 <li>d</li>
 <li>e</li>
</ul>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-19 所示。

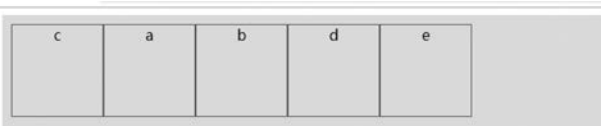


图 5-19　order 效果

(8) flex-grow（适用于弹性盒子模型容器子元素）。

设置或检索弹性盒的扩展比率(根据弹性盒子元素所设置的扩展因子作为比率来分配剩余空间)。

```
flex-grow: <number> (default 0)
```

- <number>：用数值来定义扩展比率。不允许为负值。
- flex-grow 的默认值为 0，如果没有显式定义该属性，则不会拥有分配剩余空间权利。在下面的代码中，b、c 两项都显式地定义了 flex-grow，可以看到，总共将剩余空间分成了 3 份，其中 b 占 1 份，c 占 2 份，即 1∶2。

示例代码如下所示：

```
<html>
<head>
<title>flex-grow</title>
<style>
.box{
 display:-webkit-flex;
 display:flex;
     width:600px;margin:0;padding:10px;list-style:none;background-color:yellow;}
.box li{width:100px;height:100px;border:1px solid red;text-align:center;}
#box li:nth-child(2){
 -webkit-flex-grow:1;
 flex-grow:1;
}
#box li:nth-child(3){
 -webkit-flex-grow:2;
 flex-grow:2;
}
</style>
</head>
<body>
<ul id="box" class="box">
```

```
 <li>a</li>
 <li>b</li>
 <li>c</li>
 <li>d</li>
 <li>e</li>
</ul>
</body>
</html>
```

通过 Chrome 查看该 HTML，结果如图 5-20 所示。

图 5-20 flex-grow 使用示例

(9) flex-shrink（适用于弹性盒子模型容器子元素）。

设置或检索弹性盒子的收缩比率(根据弹性盒子元素所设置的收缩因子作为比率来收缩空间)。

```
flex-shrink: <number> (default 1)
```

- flex-shrink 的默认值为 1，如果没有显式定义该属性，则会自动按照默认值 1 在所有因子相加之后计算比率来进行空间收缩。
- 在下面的例子中显式地定义了 flex-shrink，a、b 没有显式定义，但将根据默认值 1 来计算，可以看到，总共将剩余空间分成了 5 份，其中 a 占 1 份，b 占 1 份，c 占 3 份，即 1∶1∶3。
- 可以看到，父类容器定义为 400px，子项被定义为 200px，相加之后即为 600px，超出父类容器 200px。那么超出的 200px 需要被 a、b、c 分配。
- 按照以上定义 a、b、c 将按照 1∶1∶3 来分配 200px，计算后即可得 40px、40px、120px，换句话说，a、b、c 各需要使用 40px、40px、120px，那么就需要用原来定义的宽度减去这个值，最后得出 a 为 160px，b 为 160px，c 为 80px。

示例代码如下所示：

```
<html>
<head>
<title>flex-shrink</title>
<style>
#flex{display:-webkit-flex;display:flex;width:400px;margin:0;padding:0;
list-style:none;}
#flex li{width:200px;}
#flex li:nth-child(1){background:#888;}
#flex li:nth-child(2){background:red;}
#flex li:nth-child(3){-webkit-flex-shrink:3;flex-shrink:3;
background:yellow;
}
```

```
}
</style>
</head>
<body>
<ul id="flex">
 <li>a</li>
 <li>b</li>
 <li>c</li>
</ul>
</body>
</html>
```

通过 Chrome 查看该 HTML，flex-shrink 效果如图 5-21 所示。

图 5-21　flex-shrink 效果

(10) flex-basis(适用于弹性盒子模型容器子元素)。

设置或检索弹性盒子伸缩基准值。

```
flex-basis: <length> /<percentage>| auto (default auto)
```

- flex-basis：<length> | auto (default auto)。
- auto：无特定宽度值，取决于其他属性值。
- <length>：用长度值来定义宽度。不允许负值。
- <percentage>：用百分比来定义宽度。不允许负值。

示例代码如下所示：

```
<html>
<head>
<title>flex-basis</title>
<style>
.box{
 display:-webkit-flex;
 display:flex;
 width:600px;margin:0;padding:10px;list-style:none;background-color:red;
}
.box li{width:100px;height:100px;border:1px solid yellow;text-align:center;}
#box li:nth-child(3){
 -webkit-flex-basis:600px;
 flex-basis:600px;
}
</style>
</head>
<body>
<ul id="box" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
 <li>d</li>
```

```
<li>e</li>
</ul>
</body>
</html>
```

通过 Chrome 查看该 HTML,结果如图 5-22 所示。

图 5-22 flex-basis 属性

(11) flex(适用于弹性盒子模型子元素)。

flex 为复合属性,用于设置或检索伸缩盒对象的子元素如何分配空间。

如果缩写 flex: 1,则相当于 flex:110。

```
flex:none | [ flex-grow ] || [ flex-shrink ] || [ flex-basis ]
```

- none: none 关键字的写法是 flex: 0 0 auto。
- [flex-grow]: 定义弹性盒子元素的扩展比率。
- [flex-shrink]: 定义弹性盒子元素的收缩比率。
- [flex-basis]: 定义弹性盒子元素的默认基准值。

示例代码如下所示:

```
<html>
<head>
<title>flex</title>
<style>
.box{
 display:-webkit-flex;
 display:flex;
 max-width:400px;height:100px;margin:10px 0 0;padding:0;
border-radius:5px;
list-style:none;background-color:#eee;}
.box li{background:#aaa;text-align:center;}
.box li:nth-child(1){background:red;}
.box li:nth-child(2){background:yellow;}
.box li:nth-child(3){background:#ccc;}
#box li:nth-child(1){-webkit-flex:1;flex:1;}
#box li:nth-child(2){-webkit-flex:1;flex:1;}
#box li:nth-child(3){-webkit-flex:1;flex:1;}
#box2 li:nth-child(1){-webkit-flex:1 0 100px;flex:1 0 100px;}
#box2 li:nth-child(2){-webkit-flex:2 0 100px;flex:2 0 100px;}
#box2 li:nth-child(3){-webkit-flex:3 0 100px;flex:3 0 100px;}
#box3{max-width: 800px;}
#box3 li:nth-child(1){-webkit-flex:1 1 300px;flex:1 1 300px;background:red;}
#box3 li:nth-child(2){-webkit-flex:1 2 500px;flex:1 2 500px;
background:yellow;}
```

```
#box3 li:nth-child(3){-webkit-flex:1 3 600px;flex:1 3 600px;background:#ccc;}
</style>
</head>
<body>
<ul id="box" class="box">
 <li>flex:1;</li>
 <li>flex:1;</li>
 <li>flex:1;</li>
</ul>
<ul id="box2" class="box">
 <li>flex:1 0 100px;</li>
 <li>flex:2 0 100px;</li>
 <li>flex:3 0 100px;</li>
</ul>
<ul id="box3" class="box">
 <li>flex:1 1 400px;</li>
 <li>flex:1 2 400px;</li>
 <li>flex:1 2 400px;</li>
</ul>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-23 所示。

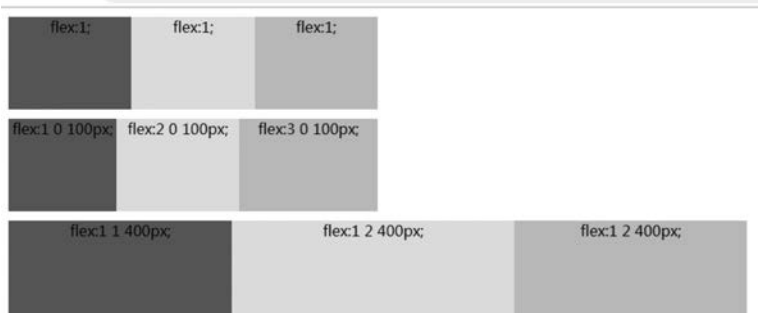


图 5-23　flex 效果

说明：

- 在上面的例子中，定义了父容器宽（即主轴宽）为 800px，由于子元素设置了伸缩基准值 flex-basis，相加为 300＋500＋600＝1400，那么子元素将会溢出 1400－800＝600(px)。
- 由于同时设置了收缩因子，所以加权综合可得 300×1＋500×2＋600×3＝3100(px)。于是可以计算 a、b、c 将被移除的溢出量是多少。

a 被移除溢出量：300×1/3100×600＝3/31，即约等于 58px。

b 被移除溢出量：500×2/3100×600＝10/31，即约等于 194px。

c 被移除溢出量：600×3/3100×600＝18/31，即约等于 348px。

最后 a、b、c 的实际宽度分别为：300－58＝242(px)，500－194＝306(px)，600－348＝252(px)。

(12) align-self（适用于弹性盒子模型子元素）。

设置或检索弹性盒子元素自身在侧轴（纵轴）方向上的对齐方式。

```
align-self: auto | flex-start | flex-end | center | baseline | stretch
```

- auto：如果'align-self'的值为'auto'，则其计算值为元素的父元素的'align-items'值，如果其没有父元素，则计算值为'stretch'。
- flex-start：弹性盒子元素的侧轴(纵轴)起始位置的边界紧靠该行的侧轴起始边界。
- flex-end：弹性盒子元素的侧轴(纵轴)起始位置的边界紧靠该行的侧轴结束边界。
- center：弹性盒子元素在该行的侧轴(纵轴)上居中放置(如果该行的尺寸小于弹性盒子元素的尺寸，则会向两个方向溢出相同的长度)。
- baseline：如弹性盒子元素的行内轴与侧轴为同一条，则该值与'flex-start'等效。在其他情况下，该值将参与基线对齐。
- stretch：如果指定侧轴大小的属性值为'auto'，则其值会使项目的边距盒的尺寸尽可能接近所在行的尺寸，但同时会遵照'min/max-width/height'属性的限制。

示例代码如下所示：

```
<html>
<head>
<title>align-self</title>
<style>
.box{
 display:-webkit-flex;
 display:flex;
  -webkit-align-items:flex-end;
 height:100px;margin:0;padding:10px;border-radius:5px;list-style:none;
background-color:yellow;}
.box li{margin:5px;padding:10px;border-radius:5px;background:red;
text-align:center;}
.box li:nth-child(1){
  -webkit-align-self:flex-end;
 align-self:flex-end;
}
.box li:nth-child(2){
  -webkit-align-self:center;
 align-self:center;
}
.box li:nth-child(3){
  -webkit-align-self:flex-start;
 align-self:flex-start;
}
.box li:nth-child(4){
  -webkit-align-self:baseline;
 align-self:baseline;
 padding:20px 10px;
}
.box li:nth-child(5){
  -webkit-align-self:baseline;
 align-self:baseline;
}
.box li:nth-child(6){
  -webkit-align-self:stretch;
```

```
align-self: stretch;
}
.box li:nth-child(7){
  -webkit-align-self: auto;
 align-self: auto;
}
</style>
</head>
<body>
<ul id="box" class="box">
 <li>a</li>
 <li>b</li>
 <li>c</li>
 <li>d</li>
 <li>e</li>
 <li>f</li>
 <li>g</li>
 <li>h</li>
 <li>i</li>
</ul>
</body>
</html>
```

图 5-24　align-self 效果

通过 Chrome 查看该 HTML，效果如图 5-24 所示。

3）CSS 浮动

float 是 CSS 样式中的定位属性，用于设置标签对象（如< div >标签、< span >标签、< a >标签、< em >标签等）的浮动布局。浮动也就是通常所说的标签对象浮动居左（float：left）和浮动居右（float：right）。浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。由于浮动框不在文档的“普通流”中，所以文档的“普通流”中的块框表现得就像浮动框不存在一样。

首先要知道，div 是块级元素，在页面中独占一行，自上而下排列，也就是通常所说的“流”。CSS 浮动效果示例如图 5-25 所示。

可以看出，即使 div1 的宽度很小，页面中一行可以容下 div1 和 div2，div2 也不会排在 div1 后边，因为 div 元素是独占一行的。注意，以上这些理论，是指标准流中的 div。显然标准流已经无法满足需求，这就要用到浮动。浮动可以理解为让某个 div 元素脱离标准流，漂浮在标准流之上，和标准流不是一个层次。

例如，假设图 5-25 中的 div2 浮动，那么它将脱离标准流，但 div1、div3、div4 仍然在标准流中，所以 div3 会自动向上移动，占据 div2 的位置，重新组成一个流。具体效果如图 5-26 所示。

从图 5-26 中可以看出，由于将 div2 设置为浮动，因此它不再属于标准流，div3 自动上移顶替 div2 的位置，div1、div3、div4 依次排列，成为一个新的流。又因为浮动是漂浮在标准流之上的，因此 div2 挡住了一部分 div3，div3 看起来变“矮”了。

这里 div2 用的是左浮动，可以理解为漂浮起来后靠左排列，右浮动当然就是靠右排列。这里的靠左、靠右是说靠页面的左、右边缘。如果对 div2 使用右浮动，那么具体效果如

图 5-25　CSS 浮动效果(1)

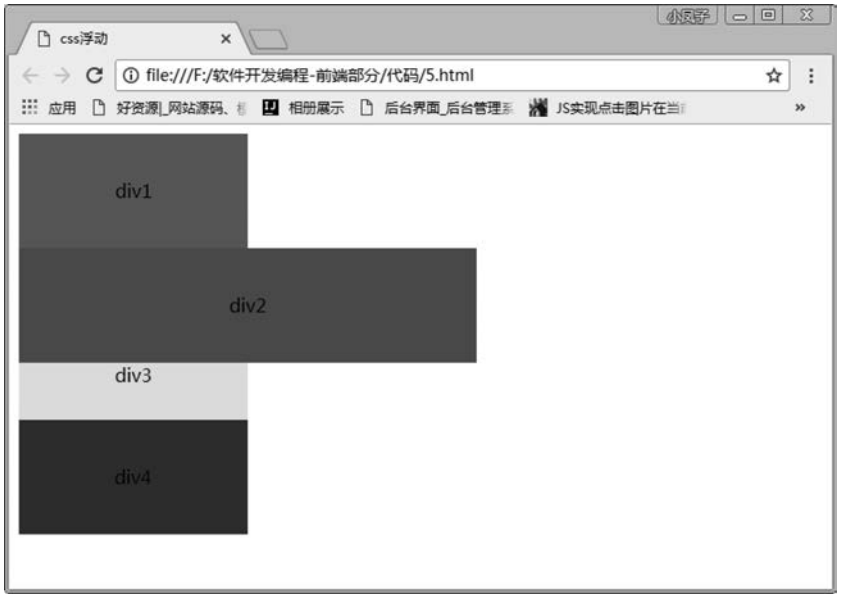


图 5-26　CSS 浮动效果(2)

图 5-27 所示。

此时 div2 靠页面右边缘排列,不再遮挡 div3,读者可以清晰地看到上面所讲的 div1、div3、div4 组成的流。

到目前为止,我们只浮动了一个 div 元素。如果浮动多个 div 元素呢? 下面将 div2 和 div3 都加上左浮动,效果如图 5-28 所示。

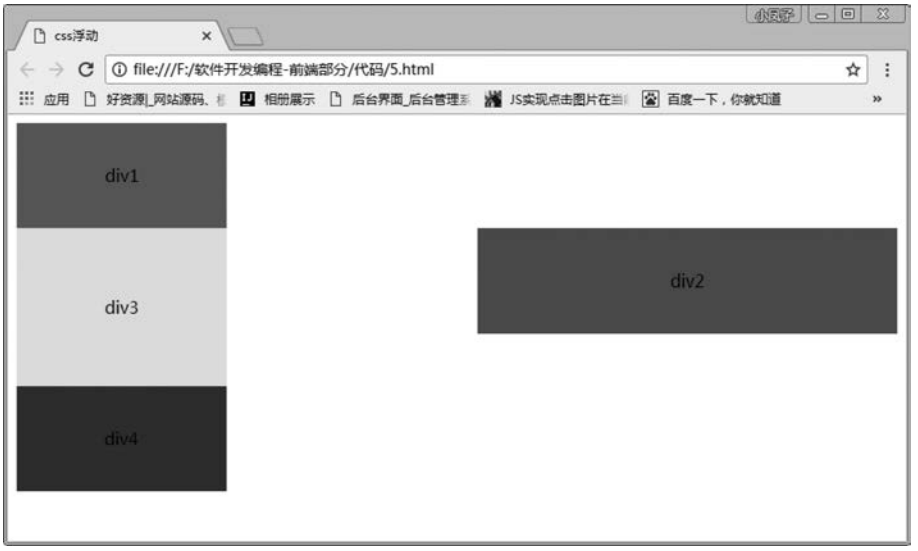


图 5-27　CSS 浮动效果(3)

图 5-28　CSS 浮动效果(4)

同理，由于 div2、div3 浮动，它们不再属于标准流，因此 div4 会自动上移，与 div1 组成一个“新”标准流，而浮动是漂浮在标准流之上，因此 div2 又挡住了 div4。假如某个 div 元素 A 是浮动的，如果 A 元素的上一个元素也是浮动的，那么 A 元素会跟随在上一个元素的后边(如果一行放不下这两个元素，那么 A 元素会被挤到下一行)；如果 A 元素的上一个元素是标准流中的元素，那么 A 的相对垂直位置不会改变，也就是说，A 的顶部总是和上一个元素的底部对齐。div 的顺序是 HTML 代码中< div >标签的顺序决定的。靠近页面边缘的一端是前，远离页面边缘的一端是后。

元素浮动之后，周围的元素会重新排列，为了避免这种情况，可使用 clear 属性。clear

属性指定元素两侧不能出现浮动元素。

清除浮动的关键字是 clear，定义如下：

```
clear : none | left | right | both
```

其中，

none——默认值。允许两边都可以有浮动对象。

left——不允许左边有浮动对象。

right——不允许右边有浮动对象。

both——不允许有浮动对象。

通过 Chrome 查看该 HTML，结果如图 5-29 所示。

图 5-29　CSS 清除浮动示例

4）可见与溢出

（1）设置元素的可见性。在 CSS 中，有两个属性可以控制元素的显示和隐藏，就是 display 和 visibility 属性。display 属性确定一个元素是否应该显示在页面上，以及如何显示。取值有 none、block、inline。

当设置元素的 display 为 none 时，元素在页面隐藏起来，不仅看不见元素，而且元素会退出当前的页面布局，不占用任何空间。

当设置元素的 display 为 block（块级）时，可以强制将 XHTML 中的内嵌元素设置成为块级元素，从而引起后续对象换行。

当设置元素的 display 为 inline（内嵌级）时，CSS 会强制将 XHTML 中的块级元素变成内嵌元素。

visibility 属性控制定位元素是否可见。取值包括 visible（可见）、hidden（隐藏）和

inherit(继承),默认值为 inherit(即继承父级元素的显示属性)。

visibility 属性与 display 属性的不同之处在于：当隐藏元素时,visibility 属性定义的元素仍然为保留原有的显示空间。

示例代码如下所示：

```
<html>
<head>
    <title>设置元素可见性</title>
    <style type="text/CSS">
    .img1{
        display:none;
    }
    </style>
</head>
<body style="background:#fff;">
<img class="img1" src="水果.jpg" alt="clip 示例图片">
<a href="www.baidu.com" title="本机">Welcome</a>
</body>
</html>
```

图 5-30　元素可见性示例

通过 Chrome 查看该 HTML,效果如图 5-30 所示。

(2) 处理溢出：如果一个元素的大小设置得太小,以致不能包含其内容,那么可以用 overflow 来指定其内容不能填充时候该如何处理。

overflow 的取值为 visible、hidden、scroll、auto,其中,visible 是默认值,表示不裁剪内容,也不添加滚动条,强制显示元素内容。

scroll 表示裁剪内容,同时提供滚动条。

hidden 表示裁剪内容,而且不显示内容,而且不显示超出对象尺寸的内容。

auto 表示只有在必要的时候才裁剪内容并添加滚动条。

注意：如果要使用 overflow 属性,那么该元素的 position 属性必须指定为绝对定位(absolute)。

示例代码如下所示：

```
<html>
<head>
    <title>处理溢出效果</title>
    <style type="text/CSS">
    div{
        width:200px;
        height:50px;
        border:solid;
        border-width: 1px;
        border-color: black;
        /*使用 overflow 属性的时候,必须将元素的 position 指定为 absolute.*/
        position: absolute;
        /*可以试试 overflow:visible/hidden/scroll/auto;*/
        overflow: auto;
```

```
    }
    </style>
</head>
<body>
<div>shenjsafeiofjwalkrjeaoisejhiowjklgjewoijglksjfsalfjewiofjlskadfjsjfe
ilsakfjeigjkhfuwfkjhguwejkfhusejkfhuegkjiekjsfiekjiekjig</div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-31 所示。

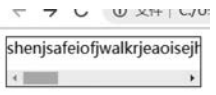


图 5-31　处理溢出效果

(3) 指定裁剪区域。clip 属性可以确定定位对象的裁剪区域，其取值为 rect(top right bottom left)/auto，其中 top、right、bottom 和 left 用于指定上、右、下、左 4 个方向上的裁剪长度，取值为长度值或 auto。如果任意一边使用 auto，则相当于该边没有进行裁剪。

示例代码如下所示：

```
<html>
<head>
<style type="text/CSS">
img
{
position:absolute;
clip:rect(0px 50px 100px 0px)
}
</style>
</head>
<body>
<p>clip 属性剪切了一幅图像:</p>
<p><img border="0" src="水果.jpg" width="120" height="151"></p>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-32 所示。

图 5-32　元素裁剪效果

（4）处理元素重叠。使用 top 和 left 属性可能会造成元素相互重叠在一起，此时可以使用 z-index 属性。z-index 属性用来控制重叠元素的显示顺序，值较大的元素将覆盖值较小的元素。如果值为－1，则表示元素将置于页面默认文本的后边，这对于设置背景图案是很有用的。

注意：z-index 属性在设置了 position 并取值为 absolute 或者 relative 时有用。

示例代码如下所示：

```
<html>
<head>
    <title>元素重叠</title>
    <style type="text/CSS">
    div{
        /*这里 position 要设置为 absolute 或者 relative*/
        position: absolute;
        border:1px solid black;
        height:50px;
        width: 60px;
    }
    div:nth-child(2n){
        background-color: blue;
    }
    #top{
        left:80px;
        top:80px;
        z-index: 3;
        background-color: gray;
    }
    #middle{
        left:60px;
        top:60px;
        z-index: 2;
        background-color: orange;
    }
    #bottom{
        left:40px;
        top:40px;
        z-index: 1;
        background-color: red;
    }
    </style>
</head>
<body>
    <div id="top">1</div>
    <div id="middle">2</div>
    <div id="bottom">3</div>
</body>
</html>
```

通过 IE 查看该 HTML，效果如图 5-33 所示。

3. CSS 定位

CSS 有 3 种基本的定位机制：普通流、浮动和绝对定位。

除非专门指定，否则所有框都在普通流中定位。也就是说，普通流中的元素的位置由元素在 HTML 中的位置决定。

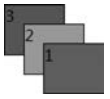


图 5-33 元素重叠效果

块级框从上到下一个接一个地排列，框之间的垂直距离是由框的垂直外边距计算出来的。

行内框在一行中水平布置。可以使用水平内边距、边框和外边距调整它们的间距。但是，垂直内边距、边框和外边距不影响行内框的高度。由一行形成的水平框称为行框(line box)，行框的高度总是足以容纳它包含的所有行内框。不过，设置行高可以增加这个框的高度。

1) position 属性

position 属性规定元素的定位类型。这个属性定义建立元素布局所用的定位机制。任何元素都可以定位，不过绝对或固定元素会生成一个块级框，而不论该元素本身是什么类型。相对定位元素会相对于它在“正常流”中的默认位置偏移。目前几乎所有主流的浏览器都支持 position 属性("inherit"除外)，如表 5-4 所示为 position 属性。

表 5-4 position 属性

值	描 述
absolute	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。元素的位置通过"left"、"top"、"right"以及"bottom"属性进行规定
fixed	生成绝对定位的元素，相对于浏览器窗口进行定位。元素的位置通过"left"、"top"、"right"以及"bottom"属性进行规定
relative	生成相对定位的元素，相对于其正常位置进行定位。因此，"left：20"会向元素的左边位置添加 20 像素
static	默认值。没有定位，元素出现在正常的流中(忽略 top、bottom、left、right 或者 z-index 声明)
inherit	规定应该从父元素继承 position 属性的值

(1) absolute(绝对定位)。

absolute 是生成绝对定位的元素，脱离了文本流(即在文档中已经不占据位置)，参照浏览器的左上角通过 top、right、bottom、left(TRBL) 定位。可以选取具有定位的父级对象(下面将介绍 relative 与 absolute 的结合使用)或者 body 坐标原点进行定位，也可以通过 z-index 进行层次分级。absolute 在没有设定 TRBL 值时是以父级对象的坐标作为原点的，设定 TRBL 值后则以浏览器的左上角作为原点。

示例代码如下所示：

```
<html>
<head>
<title>position:absolute 定位</title>
<style type="text/CSS">
    html,body,div{
                margin:0;
                padding:0;
                list-style:none;
    }
```

```
        .center{
                    margin:30px;
                    border:#999999 solid 10px;
                    width:400px;
                    height:300px;
        }
        .div1{
                width:200px;
                height:200px;
                background:yellow;
                /*设定 TRBL*/
                position:absolute;
                left:0px;
                top:0px;
        }
        .div2{
                width:400px;
                height:300px;
                font-size:30px;
                font-weight:bold;
                color:#fff;
                background:red;
        }
    </style>
    </head>
    <body>
        <div class="center">
            <div class="div1"></div>
            <div class="div2">position:absolute 定位测试</div>
        </div>
    </body>
    </html>
```

图 5-34　绝对定位效果

通过 IE 查看该 HTML，效果如图 5-34 所示。

（2）relative（相对定位）。

relative 是相对的意思，顾名思义，就是相对于元素本身在文档中应该出现的位置来移动这个元素，可以通过 TRBL 来移动元素的位置，实际上该元素依然占据文档中原有的位置，只是视觉上相对原来的位置有移动。

示例代码如下所示：

```
    <head>
    <title>position:relative 定位</title>
    <style type="text/CSS">
        html,body,div{
```

```
                margin:0;
                padding:0;
                list-style:none;
    }
    .center{
            margin:30px;
            border:#999999 solid 10px;
            width:400px;
            height:300px;
            background:#FFFF00;
    }
    .div1{
          width:200px;
          height:150px;
          background:#0099FF;
          position:relative;
          top:-20px;
          left:0px;
    }
    .div2{
          width:400px;
          height:150px;
          font-size:24px;
          font-weight:bold;
          color:#fff;
          background:#FF0000;
    }
</style>
</head>
<body>
    <div class="center">
        <div class="div1"></div>
        <div class="div2">position:relative 定位测试</div>
    </div>
</body>
</html>
```

通过 IE 查看该 HTML，效果如图 5-35 所示。

图 5-35　相对定位

(3) relative 与 absolute 的结合使用。

在网页设计时经常会用到浮动来对页面进行布局，但是浮动所带来的不确定因素很多(例如，IE 浏览器的兼容问题)。相对来说，在有些布局中定位使用会更加简单、快捷、兼容性更好(relative 与 absolute 相结合来使用)。

示例代码如下所示：

```
<html>
<head>
<style type="text/CSS">
html,body,div,ul,li,a{
                margin:0;
```

```
                padding:0;
                list-style:none;
}
a, a:hover{
 color:#000;
 border:0;
 text-decoration:none;
}
   #warp,#head,#main,#foot
{
    width: 962px;
}
/*设置居中*/
#warp{
    margin: 0 auto;
}
#head{
          height:132px;
          position:relative;
}
.logo{
      position:absolute;
      top:17px;
}
.sc a{
          padding-left:20px;
          color:#666;
}
.nav{
       width:1160px;
       height:42px;
       line-height:42px;
       position:absolute;
       bottom:0px;
       background:url(banner1.jpg) no-repeat center;
}
.nav ul{
          float:left;
          padding:0 10px;
}
.nav li{
          float:left;
          padding-right:40px;
          padding-left:20px;
          text-align:center;
          display:inline;
}
.nav li a{
              font-size:14px;
              font-family:Microsoft YaHei !important;
              white-space:nowrap;
}
.nav li a:hover{
```

```
                color:#FBECB7;
}
</style>
<title></title>
</head>
<body>
    <div id="warp">
        <div id="head">
            <div class="logo"><img src="图片 1.png" /></div>
            <div class="nav">
                <ul>
                    <li><a href="">首页</a></li>
                    <li><a href="">关于我们</a></li>
                    <li><a href="">团队文化</a></li>
                    <li><a href="">公司动态</a></li>
                    <li><a href="">资讯参考</a></li>
                    <li><a href="">联系我们</a></li>
                </ul>
            </div>
        </div>
        <div id="main"></div>
        <div id="foot"></div>
    </div>
</body>
</html>
```

通过 IE 查看该 HTML，结果如图 5-36 所示。

首页　　关于我们　　团队文化　　公司动态　　资讯参考　　联系我们

图 5-36　position 定位综合应用

(4) fixed(固定定位)。

其位置永远相对浏览器的位置来计算。即使浏览器内容向下滚动，元素位置也能相对浏览器保持不变。

示例代码如下所示：

```
<html>
    <head>
        <title>position:fixed 定位</title>
        <style type="text/CSS">
            .div1{
                background:#ccc;
                position:fixed;
                bottom:10px;
                right:100px;
            }
        </style>
    </head>
    <body>
        <div class="div1"><a href="#">回到顶部></a></div>
        <p>测试 1</p>
```

```
        <p>测试 2 </p>
        <p>测试 3 </p>
        <p>测试 4 </p>
        <p>测试 5 </p>
        <p>测试 6 </p>
        <p>测试 7 </p>
        <p>测试 8 </p>
        <p>测试 9 </p>
        <p>测试 0 </p>
    </body>
</html>
```

通过 IE 查看该 HTML，效果如图 5-37 所示。

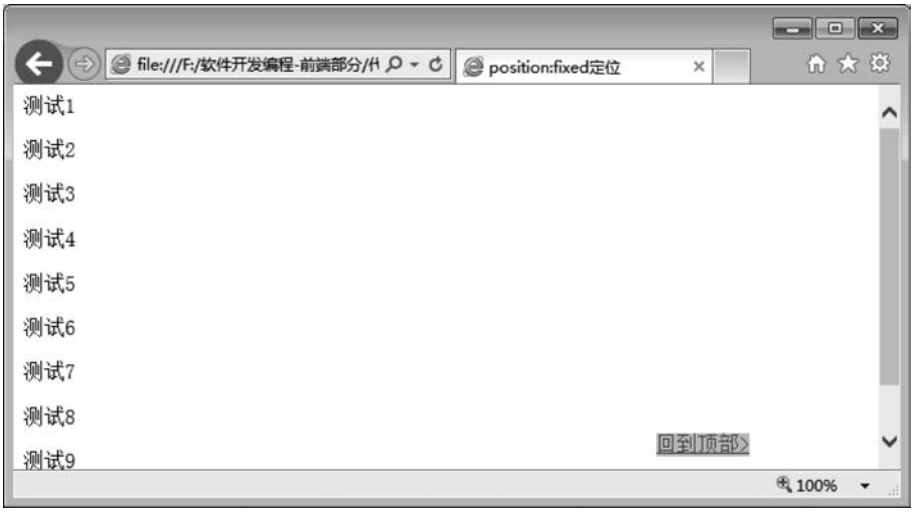


图 5-37　fixed 定位

(5) static(静态定位)。

就是不定位，出现在哪里就显示在哪里，这是默认取值，只有在你想覆盖以前的定义时才需要显示指定。

2) z-index 属性

z-index 属性指定一个元素的堆叠顺序。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。元素可拥有负的 z-index 属性值。CSS 样式表中 z-index 属性的使用方法代码如下所示：

```
.box{position:absolute; left:0px; top:0px;z - index: - 1}
```

需要注意的是：

- z-index 仅对定位元素有效(如 position：relative\absolute\float)。
- z-index 只可比较同级元素。
- z-index 的作用域：假设 A 和 B 两个元素都设置了定位(相对定位，绝对定位或一个相对定位、另一个绝对定位都可以)，且是同级元素，样式为：boxA{z-index:4}、

boxB{z-index:5}。元素 B 的层级要高于元素 A，在此需要指出的是，A 元素下面的子元素的层级也同样都低于 B 元素下面的子元素，即使将 A 元素下面的子元素设为 z-index:9999；同理，元素 B 下面的子元素，即使是设为 z-index:1，它照样比元素 A 的层级要高。

- z-index 属性不会作用于窗口控件，如 select 对象。
- 在父元素的子元素中设置 z-index 的值，可以改变子元素之间的层叠关系。
- 子元素的 z-index 值不管是高于父元素还是低于父元素，只要它们的 z-index 值是大于或等于 0 的数，它们就会显示在父元素之上，即压在父元素上。只要它们的值是小于 0 的数，就显示在父元素之下。

表 5-5 为 z-index 的属性值表。

表 5-5 z-index 的属性值

值	描 述
Auto	默认。堆叠顺序与父元素相等
Number	设置元素的堆叠顺序
Inherit	规定应该从父元素继承 z-index 属性的值

示例代码如下所示：

```
<html>
<head>
<title>z-index 属性</title>
<style>
img
{
 position:absolute;
 left:0px;
 top:0px;
 z-index:-1;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<img src="button3.png" />
<p>因为图像元素设置了 z-index 属性值为 -1，所以它会显示在文字之后.</p>
</body>
</html>
```

通过 IE 查看该 HTML，效果如图 5-38 所示。

4. 页面布局

所谓页面布局，就是将网页中的各个版块有效组织并放置在合适的位置。页面布局一般分为以下几种：

- 表格布局。
- 框架布局。
- DIV+CSS 布局模式。

图 5-38　z-index 属性示例

其中表格布局和 DIV＋CSS 布局是目前最常用的。

1）表格布局

在网页设计中，用表格显示数据只是表格功能的一部分，现在表格在网页中更多是用于网页的布局，其优势在于可以有效地定位网页中不同的元素，结构清晰。为了方便设计者使用表格进行页面布局，Dreamweaver8 提供了“布局”模式。在“布局”模式中，可以使用表格作为基础布局结构来设计网页。

使用表格布局一般要遵循以下原则：

（1）不要把整个网页当成一个大表格，尽可能使用多个表格进行分块。

因为一个大表格的内容要全部加载后才会显示。这样会降低页面的响应速度和效率。此外，单元格在调整时不够方便，在调整局部的单元格时，往往会对其他的单元格产生联动的效果，违背了调整的初衷。最常见的是将网页分为上、中、下 3 部分，上部分用于处理网页的 LOGO、BANNER、MENU 等内容；中部分处理页面的主要内容即展现的信息；下部分用于放置有关的声明、版权信息等。对于这 3 部分的内容可以使用 3 个或 3 个以上的表格进行处理。

（2）使用嵌套表格。

嵌套表格就是在一个单元格内插入另一个表格。放置嵌套表格的单元格，通常设置其垂直对齐方式为“顶端对齐”。嵌套表格作为相对独立的表格，控制十分方便，这也是使用表格布局的常用方法，但是一般不宜超过 3 层，表格嵌套过多会影响浏览器的响应速度，并且不便于后期维护。

（3）表格的边框。

当用表格布局时，表格的边框宽度一般设置为 0。最外层表格宽度一般使用固定的像素值，嵌套的表格的宽度则使用百分比来设定，如果使用像素值，则需要计算结果绝对精确，因此不提倡使用像素值。

示例代码如下所示：

```
<html>
<head>
<title>表格布局</title>
</head>
<body bgcolor = "white" leftmargin = "0" topmargin = "0">
    <!-- 外层表格开始 -->
    <table width = "100 %" border = "0" cellpadding = "0" cellspacing = "0">
        <tbody>
            <tr>
                <td align = "center">
                <!-- 中层表格开始 -->
                 <table width = "800" border = "0" bgcolor = "#FFFFFF" cellpadding = "0"
cellspacing = "0">
                <tbody>
                    <tr align = "center">
                    <!-- 内层表格开始 -->
                    <table width = "600" border = "0" cellpadding = "0" cellspacing = "0">
                    <tbody>
                        <tr>
                            <td colspan = "5"><img src = "纳兰 1.jpg" width = "600" height
= "200"/></td>
                        </tr>
                        <tr height = "50">
                            <td colspan = "3">用户名:_____ 密码:_____</td>
                            <td colspan = "2" align = "right"><a href = "http://www.baidu.
com">帮助</a></td>
                        </tr>
                        <tr height = "2">
                            <td colspan = "5" background = "黑线.png"></td>
                        </tr>
                        <tr valign = "top">
                            <td width = "140">
                                <h3 align = "center">浣溪沙</h3>
                                <ul>
                                    <li>谁念西风独自凉?</li>
                                    <li>萧萧黄叶闭疏窗,</li>
                                    <li>沉思往事立残阳。</li>
                                    <li>被酒莫惊春睡重,</li>
                                    <li>赌书消得泼茶香,</li>
                                    <li>当时只道是寻常。</li>
                                </ul>
                            </td>
                            <td width = "1" background = "黑线.png"></td>
                            <td width = "300">
                                <h3 align = "center">我爱的纳兰容若</h3>
                                <img src = "纳兰 2.jpg" width = "300" height = "300"/>
                            </td>
                            <td width = "1" background = "黑线.png"></td>
                            <td width = "140">
                                <h3 align = "center">采桑子</h3>
                                <ul>
                                    <li>明月多情应笑我,</li>
                                    <li>笑我如今。</li>
```

```
                                    <li>辜负春心,</li>
                                    <li>独自闲行独自吟。</li>
                                    <li>近来怕说当时事,</li>
                                    <li>结遍兰襟。</li>
                                    <li> 月浅灯深,</li>
                                    <li>梦里云归何处寻?</li>
                                </ul>
                            </td>
                        </tr>
                        <tr>
                            <td colspan = "5"><img src = "纳兰 3.jpg" width = "600" height =
"100" /></td>
                        </tr>
                        <tr height = "40">
                            <td colspan = "5" align = "left">版权所有,翻录必究 &copy;1999 -
2014 </td>
                        </tr>
                    </tbody>
                </table>
                <!-- 内层表格结束 -->
                </td>
            </tr>
        </tbody>
        </table>
        <!-- 中层表格结束 -->
        </td>
        </tr>
    </tbody>
</table>
<!-- 外层表格结束 -->
</body>
</html>
```

通过 Chrome 查看该 HTML,效果如图 5-39 所示。

2）框架布局

框架是另一种常用的网页布局排版工具。框架布局就是把浏览器窗口划分为多个区域,每个区域都可以分别显示不同的网页。使用框架最常见的用途就是导航,在使用了框架以后,用户的浏览器不需要为每个页面重新加载与导航相关的图形。而且每个框架可以独立设计,具有独立的功能和作用,可以实现一个浏览器窗口显示多个网页的目的。但是有的浏览器不支持框架,因此,在使用框设计网页要设计 noframes 部分,为那些不能查看框架的用户提供支持。

示例代码如下所示：

```
<html>
  <head>
    <title>框架布局</title>
  </head>
  <frameset rows = "23%, *" frameborder = "no">
        <frame src = "top.html"/>
```

图 5-39　表格布局

```
        <frameset cols="15%,*">
                <frame src="left.html" noresize="noresize" />
<frame src="right.html" noresize="noresize" name="right"/>
        </frameset>
  </frameset>
</html>
```

top.html

```
<html>
  <head>
    <title>top.html</title>
  </head>
  <body>
    <img alt="一只猫" src="button2.png" align="center">
    <font size="7">欢迎来到动感点播台</font>
  </body>
</html>
```

left.html

```
<html>
  <head>
    <title>left.html</title>
  </head>
  <body bgcolor="silver">
   <a href="right.html" target="right">青花瓷</a>
<br/><br/>
   <a href="right.html" target="right">千里之外</a>
<br/><br/>
   <a href="right.html" target="right">迷迭香</a>
<br/><br/>
   <a href="right.html" target="right">爱我别走</a>
<br/><br/>
   <a href="right.html" target="right">可爱女人</a>
<br/><br/>
   <a href="right.html" target="right">迷迭香</a>
<br/><br/>
   <a href="right.html" target="right">爱我别走</a>
<br/><br/>
   <a href="right.html" target="right">可爱女人</a>
<br/><br/>
   <a href="right.html" target="right">迷迭香</a>
<br/><br/>
   <a href="right.html" target="right">爱我别走</a>
<br/><br/>
   <a href="right.html" target="right">可爱女人</a>
<br/><br/>
   <a href="right.html" target="right">迷迭香</a>
<br/><br/>
   <a href="right.html" target="right">爱我别走</a>
<br/><br/>
   <a href="right.html" target="right">可爱女人</a>
<br/><br/>
  </body>
</html>
```

right.html

```
<html>
  <head>
    <title>right.html</title>
  </head>
  <body bgcolor="pink">
   <p>素胚勾勒出青花,笔锋浓转淡</p>
   <p>冉冉檀香透过窗,情似我了然</p>
  </body>
</html>
```

通过 Chrome 查看该 HTML,框架布局效果如图 5-40 所示。

使用框架布局的优点是：支持滚动条,方便导航,节省页面下载时间等；缺点是：兼容

图 5-40　框架布局

性不好，保存时不方便，应用范围有限等。框架布局比较适合应用于小型商业网站、论坛、后台管理系统等方面。

3）DIV＋CSS 布局

一个标准的 Web 网页由结构、外观和行为 3 部分组成，各部分的含义如下所示：

- 结构——用来对网页中的信息进行整理与分类，常用的技术有 HTML、XHTML 和 XML。
- 外观——用于对已经被结构化的信息进行外观上的修饰，包括颜色、字体等，常用技术为 CSS。
- 行为——是指对整个文档内部的一个模型进行定义及交互行为的编写，常用技术为 JavaScript。

网页设计的核心目的：实现网页的结构和外观的分离。

通常网页设计人员在设计网页之前，总是先考虑怎么设计，考虑页面中的图片、字体、颜色甚至是布局方案。然后通过其他的工具（如 PhotoShop）画出来，最后用 HTML 将所有的设计表现在页面上。如果使用 DIV＋CSS 对页面进行布局，那么可先不考虑外观，首要的是将页面内容的语义或结构确定下来。使用 DIV＋CSS 布局，外观不是最重要的，一个结构良好的 HTML 页面可以通过 CSS 以任意外观表现出来。因此引入 CSS 布局的目的就是为了实现真正意义上的结构和外观的分离，这也是 DIV＋CSS 布局最大的特色。

一个完整的网页通常包含以下几个部分：标志和站点名称、主页面内容、站点导航、子菜单、搜索区域、功能区、页脚（版权和法律声明）。有了这些结构就可以根据其在页面中的地位形成一个整体的布局思路：首先将这些结构放置在一个大框内，这个大框作为页面的父框，在其内分成头部、主体、底部 3 个部分，然后将上述的几个部分按作用和地位分配给头、体、脚。CSS 布局的整体思路如图 5-41 所示。

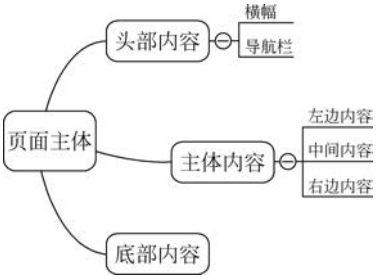


图 5-41　CSS 布局的整体思路

图 5-41 中的模块显示了各部分之间的包含关系，将各个部分都定义成 DIV，很容易就能确定各个 DIV 的嵌套关系。

示例代码如下所示：

```
<html>
<body onkeypress = "">
<center>
<div id = "container">
<div class = "Header">
<div class = "Header_left">来天涯，与 12596246 位天涯人共同演绎你的网络人生</div>
<div class = "Header_right">目前在线：156348
</div>
</div>
<div class = "Main login_gray" id = "pws_yes" style = "display:block;">
    <form method = "post" id = "login" name = "login">
        <ul>
            <li>用户名</li><input id = "text1" name = "vwriter" type = "text" class =
"inp90" size = "18" maxlength = "16" onMouseOut = "this.style.backgroundColor = '#ffffff'"
onmouseover = "this.style.backgroundColor = '#E5F0FF'" onfocus = "this.style.
backgroundColor = '#E5F0FF'"/>
            <li>密码</li>
            <li><input id = "password1" name = "vpassword"
                type = "password" class = "inp90"
                  maxlength = "18" onMouseOut = "this.style.backgroundColor = '#ffffff'"
onmouseover = "this.style.backgroundColor = '#E5F0FF'" onfocus = "this.style.
backgroundColor = '#E5F0FF'"/></li>
            <li style = "cursor:hand;position: relative;">
                <label onMouseOver = "#" onMouseOut = "#"><input type = "checkbox" name =
"rmflag" id = "rmflag" value = "1" onclick = ""/>自动登录</label>
                <div id = "clueto" style = "display:none;">
                    <div class = "clueto_top">
                        <img align = "absbottom" src = "" />
                    </div>
                    <div class = "clueto_main">
                        <p>为了确保你的信息安全，请不要在网吧或者公共机房选择此项!<br />
如果今后要取消此选项，只需单击网站顶部的 "退出"链接即可.</p>
                    </div>
                </div>
            </li>
            <li>
```

```
                    < input id = "button1" name = "button1" value = "登录" type = "button"
    name = "tianya - submit4" class = "tianya_btn50" onclick = ""/>
                    < input value = "免费注册" type = "button" name = "tianya - submit4"
                        class = "tianya_btn" onclick = ""/></li >
                < li class = "login_line" style = "height:58px">
                    < a href = "#" onclick = "">浏览进入</a ></li >
            </ul >
        </form >
    </div >
    < div class = "logining" id = "pws_auto" style = "display:none;">
        < form method = "post" id = "logining" name = "login2">
            < ul >
                < li >用户名</li >
                < li >密码</li >
                < li >
                    < input id = "password1" name = "vpassword" type = "password"class = "inp" size
     = "16" maxlength = "18" onMouseOut = "this. style. backgroundColor = '#ffffff'" onmouseover =
    "this. style. backgroundColor = '#E5F0FF'" onfocus = "this. style. backgroundColor = '#E5F0FF'"
    value = " ********** " style = "color:#666;"/></li >
                < li class = login_auto >
                    < img src = "" align = "absmiddle" />正在自动登录...</li >
                < li >< input value = "取消登录" type = "button" name = "tianya - submit4"
      class = "tianya_btn" onclick = ""/></li >
            </ul >
    </form >
    </div >
    < div class = "clear"></div >
    <!-- AD 开始中央广告位 -- >
    < SPAN id = "adsp_center_banner">
            < img src = "./images/1284862521173. jpg" border = 0 ></SPAN >
    <!-- 结束中央广告位 -- >
    <!-- 开始页面下面部分 -- >
    < div id = "Footer">
    < script type = "text/javascript" charset = "gbk" src = "./JS/tianya_footer. js"></script >
    </div >
    <!-- 结束页面下面部分 -- >
    </div >
    </body >
    </html >
```

通过 IE 查看该 HTML，效果如图 5-42 所示。

通过对天涯网页的 DIV 布局分析可知，DIV 布局的优点是网页代码精简、提高页面下载速度、表现和内容相分离等；缺点是过于灵活，比较难控制。因此 DIV 布局比较适合应用于复杂的不规则页面以及业务种类较多的大型商业网站。

5.1.3　案例实现

1. 案例分析

根据图 5-1 分析得出如图 5-43 所示结果。

图 5-42　CSS＋DIV 布局

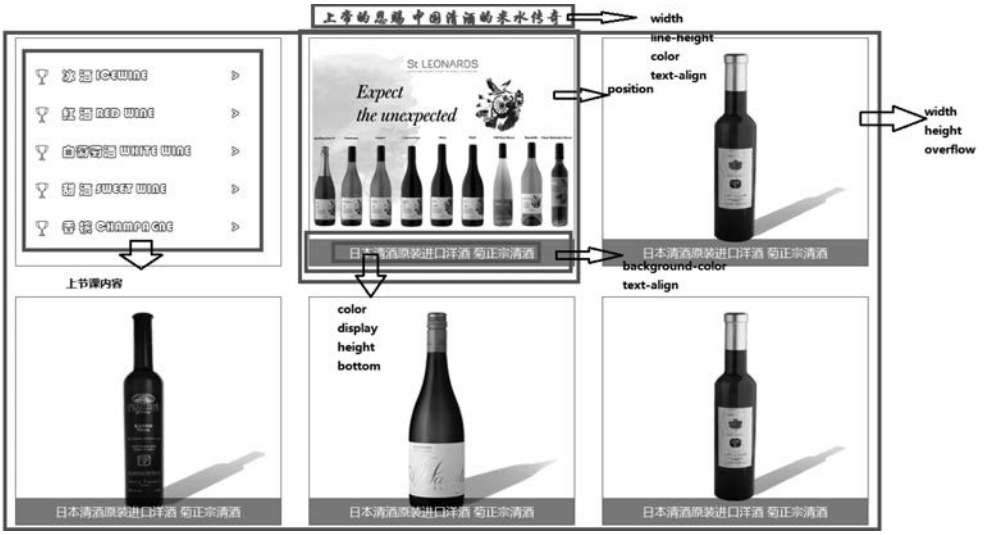


图 5-43　案例分析

2. 代码实现

1）HTML 内容

```
<!-- 产品 -->
<div class = "connent">
    <!-- 选项卡 1 内容 -->
<div class = "con">
    <h3 class = "con - title">上帝的恩赐 中国清酒的米水传奇</h3>
    <div class = "con1">
<ul>
    <li><a href = "chanping.html">冰 酒 ICEWINE <span>></span></a></li>
    <li><a href = "chanping.html">红 酒 RED WINE <span>></span></a></li>
    <li><a href = "chanping.html">白葡萄酒 WHITE WINE <span>></span></a></li>
    <li><a href = "chanping.html">甜 酒 SWEET WINE <span>></span></a></li>
    <li><a href = "chanping.html">香 槟 CHAMPAGNE <span>></span></a></li>
</ul>
    </div>
    <div class = "con1">
        <a href = "spxq.html" >
        <img src = "images/con - cp2.png" class = "con - img" style = "right: 0px;">
        <span class = "con - ht">日本清酒原装进口洋酒 菊正宗清酒</span>
        </a>
    </div>
    <div class = "con1">
        <a href = "spxq.html">
        <img src = "images/con - cp1.png" class = "con - img">
        <span class = "con - ht">日本清酒原装进口洋酒 菊正宗清酒</span>
        </a>
    </div>
    <div class = "con1">
        <a href = "spxq.html">
        <img src = "images/con - cp4.png" class = "con - img" style = " right: 20px;">
        <span class = "con - ht">日本清酒原装进口洋酒 菊正宗清酒</span>
        </a>
    </div>
    <div class = "con1">
        <a href = "spxq.html">
        <img src = "images/con - cp3.png" class = "con - img" style = " right: 20px;">
        <span class = "con - ht">日本清酒原装进口洋酒 菊正宗清酒</span>
        </a>
    </div>
    <div class = "con1">
        <a href = "spxq.html">
        <img src = "images/con - cp1.png" class = "con - img" >
        <span class = "con - ht">日本清酒原装进口洋酒 菊正宗清酒</span>
        </a>
    </div>
</div>
```

2）CSS 页面

```
.con1 {
 float: left;
```

```
 width: 350px;
 height: 300px;
 border: 1px solid #767676;
 margin-left: 36px;
 margin-bottom: 40px;
 position: relative;
}
.con1 > ul{
 width: 300px;
 height: 250px;
 margin-left: 25px;
 margin-top: 25px;
}
.con1 > ul > li{
 width: 100%;
 height: 50px;
 line-height: 50px;
 background-image: url(../images/con-listimg.png);
 background-repeat: no-repeat;
 background-position: left;
 text-indent: 2em;
}
.con1 > ul > li > a{
 position: relative;
 display: block;
 font-family: "华文彩云";
 font-size: 18px;
 color: #000000;
}
.con1 > ul > li > a > span{
 position: absolute;
 right: 30px;
 font-family: "华文彩云";
 font-size: 18px;
}
.con1 > ul > li > a:hover{
 text-decoration: underline;
}
.con-img {
 position: absolute;
 top: 20px;
 right: 30px;
}
.con-ht {
 position: absolute;
 bottom: 0px;
 width: 100%;
 height: 35px;
 background-color:rgba(0,0,0,0.4);
 display: block;
 text-align: center;
 line-height: 35px;
 color: white;
}
```

3. 运行效果

运行结果如图 5-44 所示。

图 5-44　运行效果

5.2　CSS3 动画

5.2.1　案例描述

很多网页也会有一些动画效果，例如淡入淡出、轮播图片进入进出的方式。下面将学习如何定理如下动画效果：该方块向右移动后如此变成黄色，接着下移动变成蓝色，然后向左移动变成绿色，最后向上移动变成红色，如此循环。界面静态效果如图 5-45 所示。

图 5-45　动画案例

5.2.2　知识引入

1. 变形

在 CSS3 中，可以使用 transform 属性来实现文字或图像的各种变形效果，如位移、缩放、旋转、倾斜等，表 5-6 为 transform 属性表。

表 5-6　transform 属性及方法

方法或属性	说　明
translate()	位移
scale()	缩放
rotate()	旋转
skew()	倾斜
transform-origin	中心原点

1）位移 translate()

在 CSS3 中，可以使用 translate()方法将元素沿着水平方向（X 轴）和垂直方向（Y 轴）移动。

位移 translate()方法分为 3 种情况：

- translateX(x)——元素仅在水平方向移动（X 轴移动）。
- translateY(y)——元素仅在垂直方向移动（Y 轴移动）。
- translate(x,y)——元素在水平方向和垂直方向同时移动（X 轴和 Y 轴同时移动）。

（1）translateX(x)方法。

语法如下所示：

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

在 CSS3 中，所有变形方法都属于 transform 属性，因此所有关于变形的方法前面都要加上“transform：”，以表示“变形”处理。这一点大家一定要记住。

x 表示元素在水平方向（X 轴）的移动距离，单位为 px、em 或百分比等。

当 x 为正时，表示元素在水平方向向右移动（X 轴正方向）；当 x 为负时，表示元素在水平方向向左移动（X 轴负方向）。

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 位移 translate()方法</title>
    <style type="text/CSS">
        /*设置原始元素样式*/
        #origin
        {
            margin:100px auto;                    /*水平居中*/
            width:200px;
            height:100px;
            border:1px dashed silver;
        }
        /*设置当前元素样式*/
        #current
        {
            width:200px;
            height:100px;
            color:white;
            background-color: #3EDFF4;
            text-align:center;
            transform:translateX(-20px);
           -webkit-transform:translateX(-20px); /*兼容-webkit-引擎浏览器*/
            -moz-transform:translateX(-20px);   /*兼容-moz-引擎浏览器*/
        }
    </style>
</head>
<body>
```

```
    <div id = "origin">
        <div id = "current"></div>
        <div id = "currents"></div>
    </div>
</body>
</html>
```

通过 IE 查看该 HTML，效果如图 5-46 所示，“transform：translateX(-20px);”表示元素在水平方向向左位移 20px。

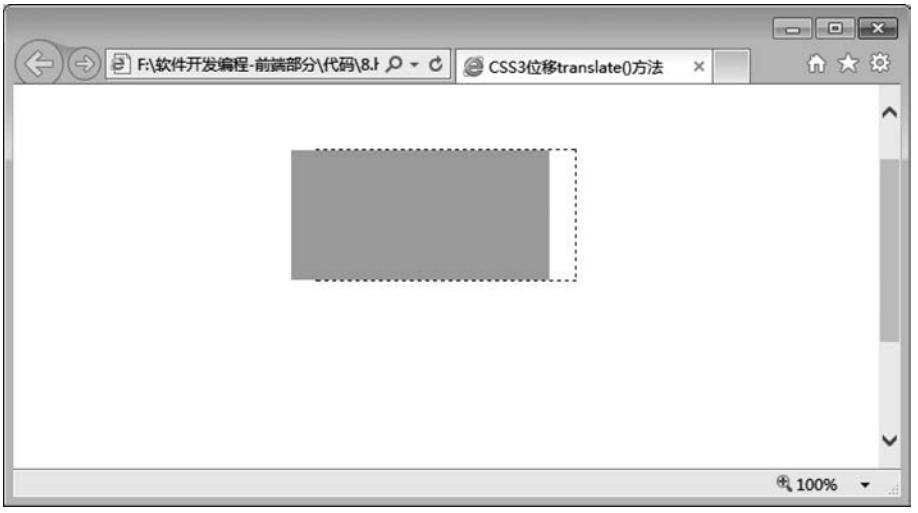


图 5-46　translateX(x)效果

(2) translateY(y)方法。

语法如下所示：

```
tranform:translateY(y)
```

y 表示元素在垂直方向(Y 轴)的移动距离，单位为 px、em 或百分比等。

当 y 为正时，表示元素在垂直方向向下移动；当 y 为负时，表示元素在垂直方向向上移动。

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 位移 translate()方法</title>
    <style type = "text/CSS">
        /* 设置原始元素样式 */
        #origin
        {
            margin:100px auto;              /* 水平居中 */
            width:200px;
            height:100px;
            border:1px dashed #000;
```

```
        }
        /*设置当前元素样式*/
        #current
    {
            width:200px;
            height:100px;
            color:white;
            background-color: #3EDFF4;
            text-align:center;
            transform:translateY(20px);
            -webkit-transform:translateY(20px);      /*兼容-webkit-引擎浏览器*/
            -moz-transform:translateY(20px);         /*兼容-moz-引擎浏览器*/
        }
    </style>
</head>
<body>
    <div id="origin">
        <div id="current"></div>
    </div>
</body>
</html>
```

通过IE查看该HTML，效果如图5-47所示，“transform：translateY(20px);”表示元素在垂直方向向下位移20px。

图5-47　translateY(y)方法

(3) translate(x,y)。

语法如下所示：

```
tranform:translate(x,y)
```

x表示元素在水平方向(X轴)的移动距离，y表示元素在垂直方向(Y轴)的移动距离。注意，y是一个可选参数，如果没有设置y值，则表示元素仅仅沿着X轴正方向移动。

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 位移 translate()方法</title>
    <style type="text/CSS">
        /*设置原始元素样式*/
        #origin
        {
            margin:100px auto;                    /*水平居中*/
            width:200px;
            height:100px;
            border:1px dashed #000;
        }
        /*设置当前元素样式*/
        #current
        {
            width:200px;
            height:100px;
            color:white;
            background-color: #3EDFF4;
            text-align:center;
            transform:translate(20px,40px);
        -webkit-transform:translate(20px,40px);  /*兼容-webkit-引擎浏览器*/
         -moz-transform:translate(20px,40px);    /*兼容-moz-引擎浏览器*/
        }
    </style>
</head>
<body>
    <div id="origin">
        <div id="current"></div>
    </div>
</body>
</html>
```

通过IE查看该HTML，效果如图5-48所示，“transform：translate(20px,40px);”表示元素在垂直方向向下位移40px，在水平方向向右移动20px。

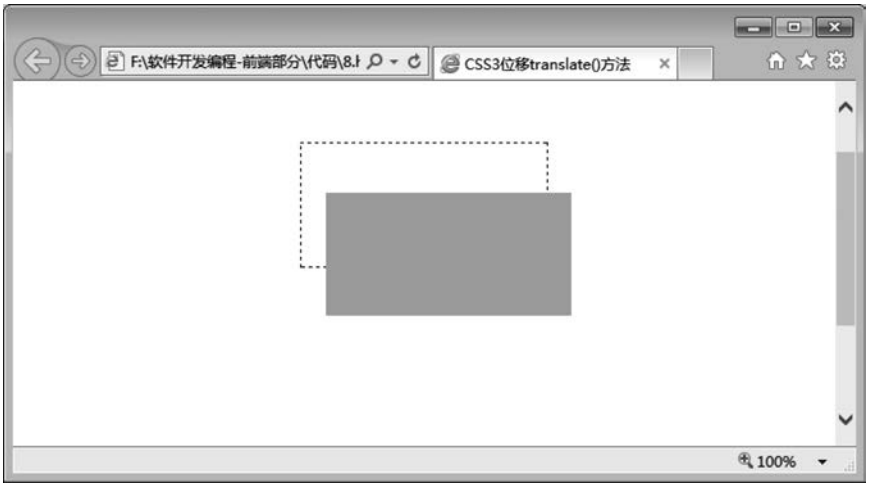


图5-48　translate(x,y)方法

2）缩放 scale()

缩放是指“缩小”和“放大”。在 CSS3 中，可以使用 scale()方法来将元素根据中心原点进行缩放。

与 translate()方法一样，scale()方法也有 3 种情况：

- scaleX(x)——元素仅水平方向缩放(X 轴缩放)。
- scaleY(y)——元素仅垂直方向缩放(Y 轴缩放)。
- scale(x,y)——元素水平方向和垂直方向同时缩放(X 轴和 Y 轴同时缩放)。

(1) scaleX(x)。

语法如下所示：

```
transform:scaleX(x)
```

x 表示元素沿着水平方向(X 轴)缩放的倍数，如果大于 1 则代表放大；如果小于 1 则代表缩小。

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 缩放 scale()方法</title>
    <style type="text/CSS">
        /*设置原始元素样式*/
        #origin
        {
            margin:100px auto;                  /*水平居中*/
            width:200px;
            height:100px;
            border:1px dashed #000;
        }
        /*设置当前元素样式*/
        #current
        {
            width:200px;
            height:100px;
            color:white;
            background-color: #3EDFF4;
            text-align:center;
            transform:scaleX(1.5);
            -webkit-transform:scaleX(1.5);   /*兼容-webkit-引擎浏览器*/
            -moz-transform:scaleX(1.5);      /*兼容-moz-引擎浏览器*/
        }
    </style>
</head>
<body>
    <div id="origin">
        <div id="current"></div>
    </div>
</body>
</html>
```

通过 IE 查看该 HTML，(x)效果如图 5-49 所示，元素沿着 X 轴方向放大了 1.5 倍(两个方向同时延伸，整体放大 1.5 倍)。

图 5-49　scaleX(x)

(2) scaleY(y)。

语法如下所示：

```
transform:scaleY(y)
```

y 表示元素沿着垂直方向(Y 轴)缩放的倍数，如果大于 1 则代表放大；如果小于 1 则代表缩小。

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 缩放 scale()方法</title>
    <style type="text/CSS">
        /*设置原始元素样式*/
        #origin
        {
            margin:100px auto;          /*水平居中*/
            width:200px;
            height:100px;
            border:1px dashed #000;
        }
        /*设置当前元素样式*/
        #current
        {
            width:200px;
            height:100px;
            color:white;
            background-color: #3EDFF4;
            text-align:center;
```

```
            transform:scaleY(2.5);
 -webkit-transform:scaleY(2.5);            /*兼容-webkit-引擎浏览器*/
 -moz-transform:scaleY(2.5);               /*兼容-moz-引擎浏览器*/
        }
    </style>
</head>
<body>
    <div id="origin">
        <div id="current"></div>
    </div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-50 所示，元素沿着 Y 轴方向放大了 1.5 倍（两个方向同时延伸，整体放大 1.5 倍）。

图 5-50　scaleY(y)效果

(3) scale(x,y)。

语法如下所示：

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

x 表示元素沿着水平方向（X 轴）缩放的倍数，y 表示元素沿着垂直方向（Y 轴）缩放的倍数。

注意，y 是一个可选参数，如果没有设置 y 值，则表示在 X 轴、Y 轴两个方向的缩放倍数是一样的（同时放大相同倍数）。

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 缩放 scale()方法</title>
```

```
    <style type="text/CSS">
        /*设置原始元素样式*/
        #origin
        {
            margin:100px auto;                  /*水平居中*/
            width:200px;
            height:100px;
            border:1px dashed #000;
        }
        /*设置当前元素样式*/
        #current
        {
            width:200px;
            height:100px;
            color:white;
            background-color: #3EDFF4;
            text-align:center;
              transform:scale(1.5,2);
-webkit-transform:scale(1.5,2);             /*兼容-webkit-引擎浏览器*/
-moz-transform:scale(1.5,2);                /*兼容-moz-引擎浏览器*/
        }
    </style>
</head>
<body>
    <div id="origin">
        <div id="current"></div>
    </div>
</body>
</html>
```

通过IE查看该HTML,效果如图5-51所示,元素沿着X轴方向放大了1.5倍,在Y轴方向放大了2倍。

图5-51　scale(x,y)效果

3）旋转 rotate()

在 CSS3 中，可以使用 rotate()方法来将元素相对中心原点进行旋转。这里的旋转是二维的，不涉及三维空间的操作。

语法如下所示：

```
transform:rotate(度数);
```

度数是指元素相对中心原点进行旋转的度数，单位为 deg。

如果度数为正，则表示元素相对原点中心顺时针旋转；如果度数为负，则表示元素相对原点中心逆时针旋转。

示例代码如下所示：

```
<head>
    <title>CSS3 旋转 rotate()方法</title>
    <style type="text/CSS">
        /*设置原始元素样式*/
        #origin
        {
            margin:100px auto;                  /*水平居中*/
            width:200px;
            height:100px;
            border:1px dashed gray;
        }
        /*设置当前元素样式*/
        #current
        {
            width:200px;
            height:100px;
            line-height:100px;
            color:white;
            background-color: #007BEE;
            text-align:center;
            transform:rotate(30deg);
            -webkit-transform:rotate(30deg); /*兼容-webkit-引擎浏览器*/
            -moz-transform:rotate(30deg);    /*兼容-moz-引擎浏览器*/
        }
    </style>
</head>
<body>
    <div id="origin">
        <div id="current">顺时针旋转 30°</div>
    </div>
</body>
</html>
```

通过 IE 查看该 HTML，效果如图 5-52 所示，这里虚线框为原始位置，蓝色背景盒子为顺时针旋转 30°后的效果。

4）倾斜 skew()

在 CSS3 中，可以使用 skew()方法将元素倾斜显示。

与 translate()方法、scale()方法一样，skew()方法也有 3 种情况：

- skewX(x)——使元素在水平方向倾斜(X 轴倾斜)。
- skewY(y)——使元素在垂直方向倾斜(Y 轴倾斜)。
- skew(x,y)——使元素在水平方向和垂直方向同时倾斜(X 轴和 Y 轴同时倾斜)，先按照 skewX()方法倾斜，然后按照 skewY()方法倾斜。

图 5-52　rotate()效果

(1) skewX(x)方法。

语法如下所示：

```
transform:skewX(x);
```

x 表示元素在 X 轴倾斜的度数，单位为 deg。

如果度数为正，则表示元素沿水平方向(X 轴)顺时针倾斜；如果度数为负，则表示元素沿水平方向(X 轴)逆时针倾斜。

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 倾斜 skew()方法</title>
    <style type="text/CSS">
        /*设置原始元素样式*/
        #origin
        {
            margin:100px auto;                    /*水平居中*/
            width:200px;
            height:100px;
            border:1px dashed #000;
        }
        /*设置当前元素样式*/
        #current
        {
            width:200px;
            height:100px;
            color:white;
            background-color: red;
            text-align:center;
            transform:skewX(30deg);
            -webkit-transform:skewX(30deg);     /*兼容-webkit-引擎浏览器*/
            -moz-transform:skewX(30deg);        /*兼容-moz-引擎浏览器*/
        }
    </style>
</head>
<body>
    <div id="origin">
```

```
        <div id = "current"></div>
    </div>
</body>
</html>
```

通过 IE 查看该 HTML，效果如图 5-53 所示。

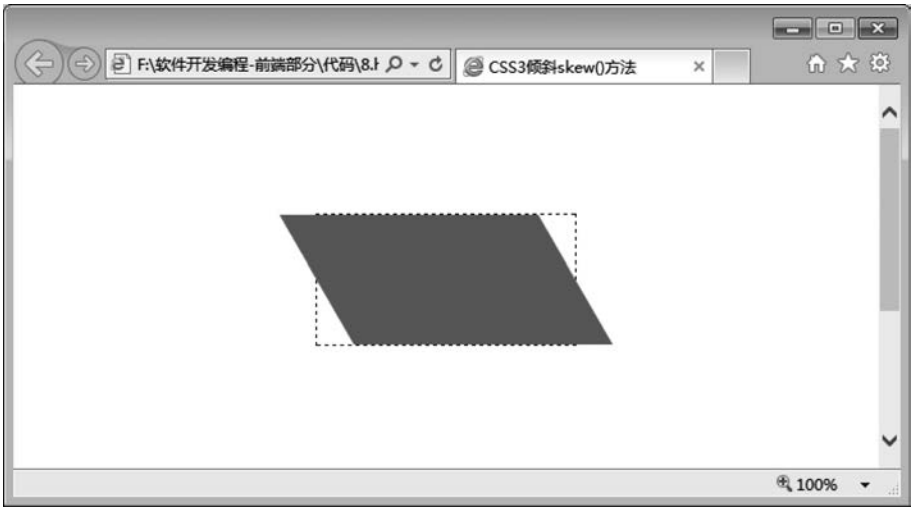


图 5-53　skewX(x)效果

(2) skewY(y)方法。

语法如下所示：

```
transform:skewY(y);
```

y 表示元素在 Y 轴倾斜的度数，单位为 deg。

如果度数为正，则表示元素沿垂直方向(Y 轴)顺时针倾斜；如果度数为负，则表示元素沿垂直方向(Y 轴)逆时针倾斜。

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 倾斜 skew()方法</title>
    <style type = "text/CSS">
        /* 设置原始元素样式 */
        #origin
        {
            margin:100px auto;              /* 水平居中 */
            width:200px;
            height:100px;
            border:1px dashed #000;
        }
        /* 设置当前元素样式 */
        #current
```

```
        {
            width:200px;
            height:100px;
            color:white;
            background-color: red;
            text-align:center;
        transform:skewY(30deg);
-webkit-transform:skewY(30deg);      /*兼容-webkit-引擎浏览器*/
-moz-transform:skewY(30deg);         /*兼容-moz-引擎浏览器*/
        }
    </style>
</head>
<body>
    <div id="origin">
        <div id="current"></div>
    </div>
</body>
</html>
```

通过 IE 查看该 HTML,效果如图 5-54 所示。

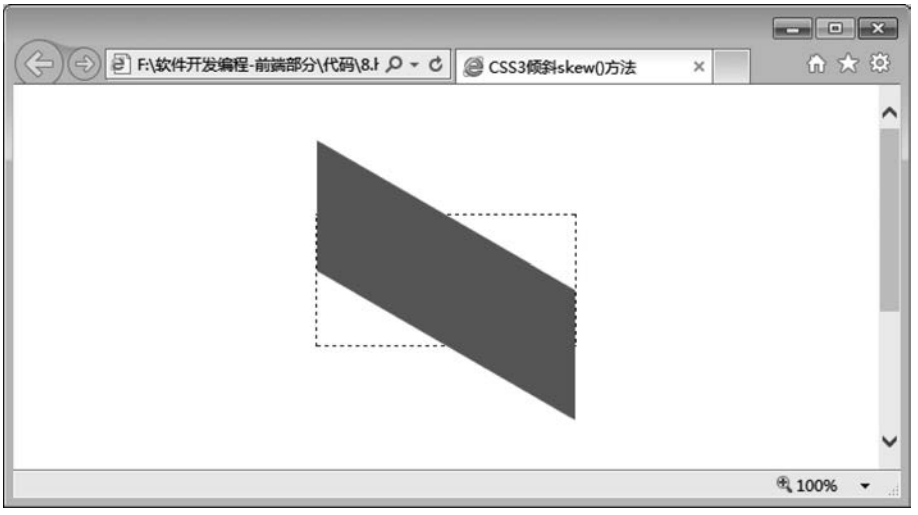


图 5-54　skewY(y)效果

(3) skew(x,y)方法。

语法如下所示:

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

第一个参数对应 X 轴,第二个参数对应 Y 轴。如果第二个参数未提供,则值为 0,也就是 Y 轴方向上无斜切。

示例代码如下所示:

```
<html>
<head>
```

```
    <title>CSS3 倾斜 skew()方法</title>
    <style type="text/CSS">
        /*设置原始元素样式*/
        #origin
        {
            margin:100px auto;                    /*水平居中*/
            width:200px;
            height:100px;
            border:1px dashed #000;
        }
        /*设置当前元素样式*/
        #current
        {
            width:200px;
            height:100px;
            color:white;
            background-color: red;
            text-align:center;
        transform:skew(30deg,30deg);
-webkit-transform:skew(30deg,30deg);              /*兼容-webkit-引擎浏览器*/
-moz-transform:skew(30deg,30deg);                 /*兼容-moz-引擎浏览器*/
        }
    </style>
</head>
<body>
    <div id="origin">
        <div id="current"></div>
    </div>
</body>
</html>
```

通过 IE 查看该 HTML，效果如图 5-55 所示。

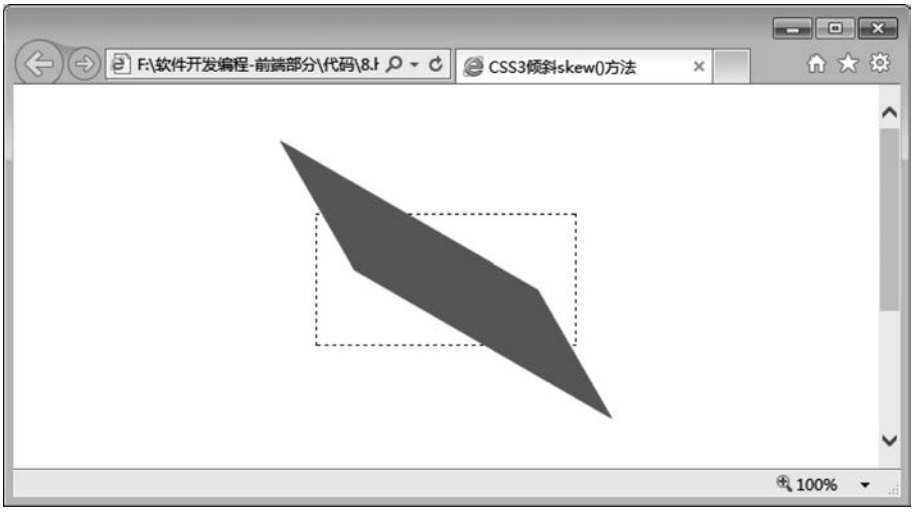


图 5-55　skew(x,y)效果

5）中心原点 transform-origin

任何一个元素都有一个中心原点，默认情况下，元素的中心原点位于 X 轴和 Y 轴的 50%处。

默认情况下，CSS3 变形进行的位移、缩放、旋转、倾斜都是以元素的中心原点进行变形。

假如要使得元素进行位移、缩放、旋转、倾斜这些变形操作的中心原点不是原来元素的中心位置，那该怎么办呢？

在 CSS3 中，可以通过 transform-origin 属性来改变元素变形时的中心原点位置。

语法如下所示：

```
transform - origin:取值;
```

transform-origin 属性取值有 2 种：一种是采用长度值，另外一种是使用关键字。长度值一般使用百分比作为单位，很少使用 px、em 等作为单位。

不管 transform-origin 取值为长度值还是关键字，都需要设置水平方向和垂直方向的值。transform-origin 属性取值跟背景位置 background-position 属性取值相似，表 5-7 为 transform-origin 属性取值。

表 5-7 transform-origin 属性取值

关键字	百分比	说 明
top left	0 0	左上
top center	50% 0	靠上居中
top right	100% 0	右上
left center	0 50%	靠左居中
center center	50% 50%	正中
right center	100% 50%	靠右居中
bottom left	0 100%	左下
bottom center	50% 100%	靠下居中
bottom right	100% 100%	右下

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 中心原点 transform - origin 属性</title>
    <style type = "text/CSS">
        /* 设置原始元素样式 */
        #origin
        {
            margin:100px auto;              /* 水平居中 */
            width:200px;
            height:100px;
            border:1px dashed gray;
        }
        #current
        {
            width:200px;
```

```
            height:100px;
            color:white;
            background-color: yellow;
            text-align:center;
            transform-origin:right center;
            -webkit-transform-origin:right center;    /*兼容-webkit-引擎浏览器*/
            -moz-transform-origin:right center;       /*兼容-moz-引擎浏览器*/
            transform:rotate(30deg);
            -webkit-transform:rotate(30deg);          /*兼容-webkit-引擎浏览器*/
             -moz-transform:rotate(30deg);            /*兼容-moz-引擎浏览器*/
        }
    </style>
</head>
<body>
    <div id="origin">
        <div id="current"></div>
    </div>
</body>
</html>
```

通过 IE 查看该 HTML，效果如图 5-56 所示，“transform-origin：right center;”使得 CSS3 变形的中心原点由“正中”变为“靠右居中”。

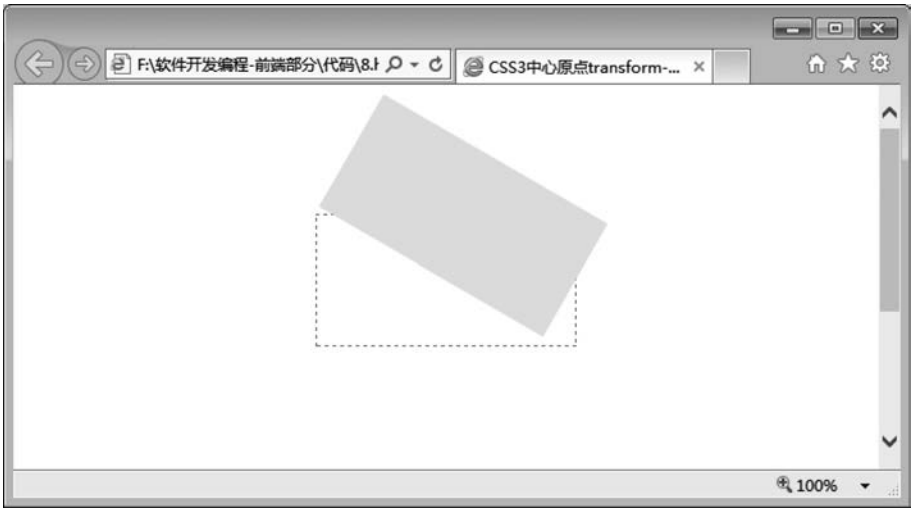


图 5-56　transform-origin 效果

2．过渡

在 CSS3 中，可以使用 transition 属性来将元素的某一个属性从“一个属性值”在指定的时间内平滑地过渡到“另外一个属性值”来实现动画效果。

CSS transform 属性所实现的元素变形，呈现的仅仅是一个“结果”，而 CSS transition 呈现的是一种过渡“过程”，通俗地说，就是一种动画转换过程，如渐显、渐隐、动画快慢等。例如，绿叶学习网中很多地方都用到了 CSS3 过渡，当鼠标指针移动上去的时候，都会有一定的过渡效果。

语法如下所示：

```
transition:属性 持续时间 过渡方法 延迟时间;
```

transition 属性是一个复合属性，主要包含 4 个子属性：

(1) transition-property——对元素的哪一个属性进行操作。

(2) transition-duration——过渡的持续时间。

(3) transition-timing-function——过渡使用的方法(函数)。

(4) transition-delay——可选属性，指定过渡开始出现的延迟时间。

1) 过渡属性 transition-property

可以使用 transition-property 属性单独设定过渡动画所要操作的那个属性。

语法如下所示：

```
transition-property:取值;
```

transition-property 属性的取值是一个“CSS 属性名”。

示例代码如下所示：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>CSS3 transition-property 属性</title>
    <style type="text/CSS">
        div
        {
            display:inline-block;
            width:100px;
            height:50px;
            background-color:#14C7F3;
            transition-property:height;
            transition-duration:0.5s ;
            transition-timing-function:linear;
            transition-delay:0;
        }
        div:hover
        {
            height:100px;
        }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-57 和图 5-58 所示。这里使用 transition-property 属性指定了过渡动画所操作的 CSS 属性是 height。当鼠标指针移动到 div 元素上时，元素的高度会在 0.5s 内从 50px 过渡到 100px。

2) transition-duration 属性

在 CSS3 中，可以使用 transition-duration 属性单独设置过渡持续的时间。

图 5-57　transition-property 属性鼠标放置之前

图 5-58　transition-property 属性过渡之后

语法如下所示：

```
transition-duration:时间;
```

transition-duration 属性取值是一个时间，单位为 s(秒)，可以为小数，如 0.5s。示例代码如下所示：

```
<html>
<head>
    <title>CSS3 过渡</title>
    <style type="text/CSS">
        div
        {
            display:inline-block;
            width:100px;
            height:100px;
            border-radius:0;
```

```
            background-color:#14C7F3;
            transition-property:border-radius;
            transition-duration:0.5s;
            transition-timing-function:linear;
            transition-delay:0;
        }
        div:hover
        {
            border-radius:50px;
        }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-59 和图 5-60 所示的当鼠标指针移动到 div 元素上面时，div 元素的圆角半径在 0.5 秒内从 0 过渡到 50px。

图 5-59　transition-duration 属性鼠标放置之前

图 5-60　transition-duration 过渡之后

3）过渡方式 transition-timing-function

在 CSS3 中可以使用 transition-timing-function 属性来定义过渡方式。“过渡方式”主要用来指定动画在过渡时间内的变化速率。

语法如下所示：

```
transition-function:取值;
```

transition-timing-function 属性取值共有 5 种，具体如图 5-61 所示。

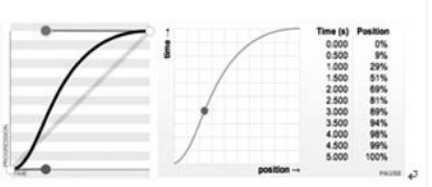
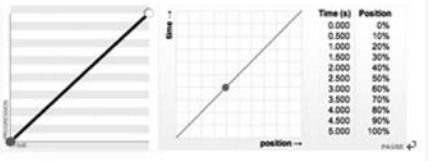
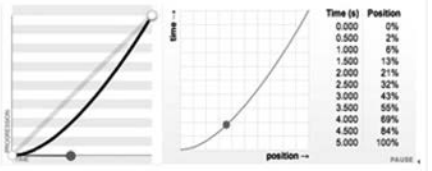
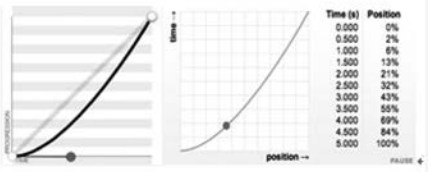
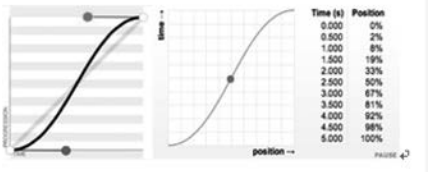
函数	功能描述	图例
ease	默认值，元素样式从初始状态过渡到终止状态时速度由快到慢，逐渐变慢	
linear	元素样式从初始状态过渡到终止状态速度恒定	
ease-in	元素样式从初始状态过渡到终止状态时，速度越来越快，呈一种加速状态。常称这种效果为渐显效果	
ease-out	元素样式从初始状态过渡到终止状态时，速度越来越慢，呈一种减速状态。常称这种效果为渐隐效果	
ease-in-out	元素样式从初始状态到终止状态时，先加速再减速。常称这种效果为渐显渐隐效果	

图 5-61　transition-timing-function 属性取值

示例代码如下所示：

```
<html>
<head>
```

```
    <title>CSS3 transition-timing-function 属性</title>
    <style type="text/CSS">
        div
        {
            width:100px;
            height:50px;
            text-align:center;
            line-height:50px;
            border-radius:0;
            background-color:#14C7F3;
            transition-property:width;
            transition-duration:2s ;
            transition-delay:0;
        }
        div+div
        {
            margin-top:10px;
        }
        #div1{transition-timing-function:linear;}
        #div2{transition-timing-function:ease;}
        #div3{transition-timing-function:ease-in;}
        #div4{transition-timing-function:ease-out;}
        #div5{transition-timing-function:ease-in-out}
        div:hover
        {
            width:300px;
        }
    </style>
</head>
<body>
    <div id="div1">linear</div>
    <div id="div2">ease</div>
    <div id="div3">ease-in</div>
    <div id="div4">ease-out</div>
    <div id="div5">ease-in-out</div>
</body>
</html>
```

通过 Chrome 查看该 HTML,效果如图 5-62 所示。

4) 延迟时间 transition-delay

可以使用 transition-delay 属性来设置动画开始的延迟时间。

语法如下所示:

```
transition-delay:时间;
```

transition-delay 属性取值是一个时间,单位为 s(秒),可以为小数,如 0.5s。

transition-delay 属性默认值为 0,也就是说,当没有设置 transition-delay 属性时,过渡动画就没有延迟时间。

示例代码如下所示:

图 5-62　transition-timing-function 效果

```
<html>
<head>
    <title>CSS3 transition - delay 属性</title>
    <style type = "text/CSS">
        div
        {
            display:inline - block;
            width:100px;
            height:100px;
            border - radius:0;
            background - color:#14C7F3;
            transition - property:border - radius;
            transition - duration:1s ;
            transition - timing - function:linear;
            transition - delay:2s;
        }
        div:hover
        {
            border - radius:50px;
        }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-63 所示。“transition-delay：2s；”表示鼠标指针移动到 div 的那一瞬间开始计时，在计时开始之后还得延迟 2s 才会开始进行过渡动画，这就是所谓的“延迟时间”。然后，从鼠标指针移出 div 的一瞬间开始，过渡动画同样也会延迟 2s 才会开始恢复。

图 5-63　transition-delay 效果

3. 动画

在 CSS3 中，动画效果使用 animation 属性来实现。animation 属性和 transition 属性功能是相同的，都是通过改变元素的“属性值”来实现动画效果的。但是这两者又有很大的区别：transition 属性只能通过指定属性的开始值与结束值，然后在这两个属性值之间进行平滑过渡来实现动画效果，因此只能实现简单的动画效果。animation 属性则通过定义多个关键帧以及定义每个关键帧中元素的属性值来实现复杂的动画效果。

1）@keyframes

使用 animation 属性定义 CSS3 动画需要两步：

（1）定义动画。

（2）调用动画。

在 CSS3 中，在使用动画之前，必须使用@keyframes 规则定义动画。

语法如下所示：

```
@keyframes 动画名
{
    0%
    {
        …
    }
    …
    100%
    {

    }
}
```

0%表示动画的开始，100%表示动画的结束。0%和 100%是必需的，不过在一个@keyframes 规则中可以由多个百分比构成，每一个百分比都可以定义自身的 CSS 样式，从而形成一系列的动画效果。

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 @keyframes</title>
    <style type="text/CSS">
        @-webkit-keyframes mycolor
        {
            0%{background-color:red;}
            30%{background-color:blue;}
            60%{background-color:yellow;}
            100%{background-color:green;}
        }
        div
        {
            width:100px;
            height:100px;
            border-radius:50px;
            background-color:red;
        }
        div:hover
        {
            -webkit-animation-name:mycolor;
            -webkit-animation-duration:5s;
            -webkit-animation-timing-function:linear;
        }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-64 所示。这里使用@keyframes 规则定义了一个名为 mycolor 的动画，刚开始时背景颜色为红色，在 0%～30%部分背景颜色从红色变为蓝色，然后在 30%～60%部分背景颜色从蓝色变为黄色，最后在 60%～100%部分背景颜色从蓝色变为绿色。动画执行完毕，背景颜色回归为红色(初始值)。

图 5-64　@keyframes 定义动画

2）animation-name 属性

在 CSS3 中，使用@keyframes 规则定义的动画并不会自动执行，还需要使用 animation-name 属性来调用动画，之后动画才会生效。

语法如下所示：

```
animation-name:动画名;
```

注意，animation-name 调用的动画名需要和@keyframes 规则定义的动画名称完全一致（区分大小写），如果不一致将不具有任何动画效果。为了浏览器兼容性，针对 Chrome 和 Safari 浏览器需要加上-webkit-前缀，而针对 Firefox 浏览器需要加上-moz-。

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 animation-name 属性</title>
    <style type="text/CSS">
        @-webkit-keyframes mycolor
        {
            0%{background-color:red;}
            30%{background-color:blue;}
            60%{background-color:yellow;}
            100%{background-color:green;}
        }
        @-webkit-keyframes mytransform
        {
            0%{border-radius:0;}
            50%{border-radius:50px; -webkit-transform:translateX(0);}
            100%{border-radius:50px; -webkit-transform:translateX(50px);}
        }
        div
        {
            width:100px;
            height:100px;
            background-color:red;
        }
        div:hover
        {
            -webkit-animation-name:mytransform;
            -webkit-animation-duration:5s;
            -webkit-animation-timing-function:linear;
        }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-65 所示。这里使用@keyframes 规则定义了两个动画：mycolor 和 mytransform。但是我们只使用 animation-name 调用名为

mytransform 的动画。因此，名为 mytransform 的动画会生效，而名为 mycolor 的动画不会生效。

图 5-65　animation-name 调用动画

在 mytransform 动画中，0%～50%的 div 元素 border-radius 属性值实现从 0 变成 50px，然后在 50%～100%的部分保持 border-radius 属性值不变，并且水平向右移动 50px。

3）持续时间 animation-duration

在 CSS3 中，可以使用 animation-duration 属性来设置动画的持续时间，也就是完成从 0%～100%所使用的总时间。animation-duration 属性与 CSS3 过渡中的 transition-duration 属性相似。

语法如下所示：

```
animation-duration:时间;
```

animation-duration 属性取值是一个时间，单位为 s(秒)，可以为小数，如 0.5s。

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 animation-duration 属性</title>
    <style type="text/CSS">
        @-webkit-keyframes mytranslate
        {
            0%{}
            100%{-webkit-transform:translateX(100px);}
        }
        div:not(#container)
        {
            width:40px;
            height:40px;
            border-radius:20px;
            background-color:red;
```

```
            -webkit-animation-name:mytranslate;
            -webkit-animation-timing-function:linear;
        }
        #container
        {
            display:inline-block;
            width:140px;
            border:1px solid silver;
        }
        #div1{-webkit-animation-duration:2s;margin-bottom:10px;}
        #div2{-webkit-animation-duration:4s;}
    </style>
</head>
<body>
    <div id="container">
        <div id="div1"></div>
        <div id="div2"></div>
    </div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-66 所示。这里设置 #div1 的元素动画持续时间为 2s，而设置 #div2 的元素动画持续时间为 4s。

图 5-66　animation-duration 效果

4）播放方式 animation-timing-function

在 CSS3 中，可以使用 animation-timing-function 属性来设置动画的播放方式，“播放方式”主要用来指定动画在播放时间内的变化速率。

语法如下所示：

```
animation-timing-function:取值;
```

animation-timing-function 属性取值跟 transition-timing-function 属性取值一样，共有

5 种，具体如图 5-67 所示。

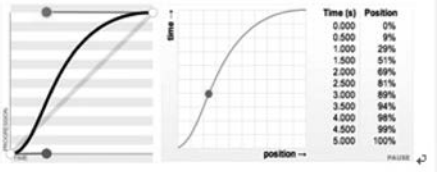
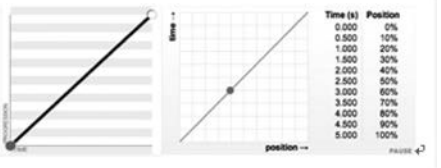
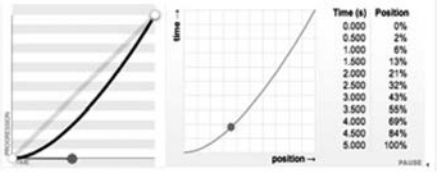
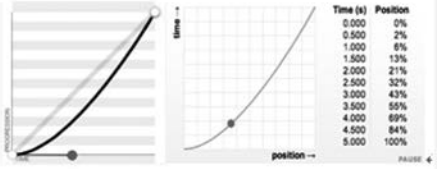
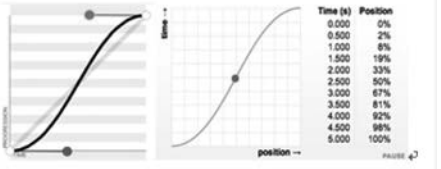
函数	功能描述	图例
ease	默认值，元素样式从初始状态过渡到终止状态时速度由快到慢，逐渐变慢	
linear	元素样式从初始状态过渡到终止状态速度恒定	
ease-in	元素样式从初始状态过渡到终止状态时，速度越来越快，呈一种加速状态。常称这种效果为渐显效果	
ease-out	元素样式从初始状态过渡到终止状态时，速度越来越慢，呈一种减速状态。常称这种效果为渐隐效果	
ease-in-out	元素样式从初始状态到终止状态时，先加速再减速。常称这种效果为渐显渐隐效果	

图 5-67　animation-timing-function 属性取值

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 animation - timing - function 属性</title>
    <style type="text/CSS">
        @ - webkit - keyframes mytransform
        {
            0%{ }
            100%{width:300px;}
        }
        div
```

```
        {
            width:100px;
            height:50px;
            text-align:center;
            line-height:50px;
            border-radius:0;
            background-color:#14C7F3;
            -webkit-animation-name:mytransform;
            -webkit-animation-duration:5s;
            -webkit-animation-timing-function:linear;
        }
        div+div
        {
            margin-top:10px;
        }
        #div1{-webkit-animation-timing-function:linear;}
        #div2{-webkit-animation-timing-function:ease;}
        #div3{-webkit-animation-timing-function:ease-in;}
        #div4{-webkit-animation-timing-function:ease-out;}
        #div5{-webkit-animation-timing-function:ease-in-out}
    </style>
</head>
<body>
    <div id="div1">linear</div>
    <div id="div2">ease</div>
    <div id="div3">ease-in</div>
    <div id="div4">ease-out</div>
    <div id="div5">ease-in-out</div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-68 所示。

图 5-68　animation-timing-function 效果

5）延迟时间 animation-delay

在 CSS3 中，可以使用 animation-delay 属性来定义动画播放的延迟时间。

语法如下所示：

```
animation-delay:时间;
```

animation-delay 属性取值是一个时间，单位为 s(秒)，可以为小数如 0.5s。

animation-delay 属性默认值为 0，也就是说，当我们没有设置 animation-delay 属性时，CSS3 动画就没有延迟时间。

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 animation-delay 属性</title>
    <style type="text/CSS">
        @-webkit-keyframes mytranslate
        {
            0%{}
            100%{-webkit-transform:translateX(100px);}
        }
        #div1
        {
            width:40px;
            height:40px;
            border-radius:20px;
            background-color:red;
            -webkit-animation-name:mytranslate;
            -webkit-animation-timing-function:linear;
            -webkit-animation-duration:2s;
            -webkit-animation-delay:2s; /*设置动画在页面打开之后延迟 2s 开始播放*/
        }
        #container
        {
            display:inline-block;
            width:140px;
            border:1px solid silver;
        }
    </style>
</head>
<body>
    <div id="container">
        <div id="div1"></div>
    </div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-69 所示。这里使用 animation-delay 属性设置动画延迟时间为 2s，也就是说，当页面打开之后延迟 2s 动画才开始播放。

6）播放次数 animation-iteration-count

在 CSS3 中，可以使用 animation-iteration-count 属性来定义动画的播放次数。

图 5-69　animation-delay

语法如下所示：

```
animation - iteration - count:取值;
```

animation-iteration-count 属性取值有两种：

(1) 正整数。

(2) infinity。

animation-iteration-count 属性默认值为 1。也就是说，在默认情况下，动画从开始到结束只播放一次。“animation-iteration-count：n”表示动画播放 n 次，n 为正整数。

当 animation-iteration-count 属性取值为 infinite 时，动画会无限次地循环播放。

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 animation - iteration - count 属性</title>
    <style type = "text/CSS">
        @ - webkit - keyframes mytranslate
        {
            0 % {}
            50 % { - webkit - transform:translateX(100px);}
            100 % {}
        }
        #div1
        {
            width:40px;
            height:40px;
            border - radius:20px;
            background - color:red;
            - webkit - animation - name:mytranslate;
            - webkit - animation - timing - function:linear;
            - webkit - animation - duration:2s;
            - webkit - animation - iteration - count:infinite;
```

```
        }
        #container
        {
            display:inline-block;
            width:140px;
            border:1px solid silver;
        }
    </style>
</head>
<body>
    <div id="container">
        <div id="div1"></div>
    </div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-70 所示。这里设置 animation-iteration-count 属性值为 infinite，然后动画会不断循环播放。

图 5-70　animation-iteration-count 效果

7）播放方向 animation-direction

在 CSS3 中，可以使用 animation-direction 属性定义动画的播放方向。

语法如下所示：

```
animation-direction:取值;
```

animation-direction 属性取值如表 5-8 所示。

表 5-8　animation-direction 属性取值

属性	说　明
normal	每次循环都向正方向播放（默认值）
reverse	每次循环都向反方向播放
alternate	播放次数是奇数时，动画向原方向播放；播放次数是偶数时，动画向反方向播放

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 animation-direction 属性</title>
    <style type="text/CSS">
        @-webkit-keyframes mytranslate{
            0%{}
            100%{-webkit-transform:translateX(100px);}
        }
        #div1{
            width:40px;
            height:40px;
            border-radius:20px;
            background-color:red;
            -webkit-animation-name:mytranslate;
            -webkit-animation-timing-function:linear;
            -webkit-animation-duration:2s;
        }
        #container{
            display:inline-block;
            width:140px;
            border:1px solid silver;
        }
    </style>
</head>
<body>
    <div id="container">
        <div id="div1"></div>
    </div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-71 所示。这里设置动画播放沿着正方向播放（0%～100%），小球从左到右滚动。

图 5-71　animation-direction 效果

8）动画播放状态 animation-play-state

在 CSS3 中，可以使用 animation-play-state 属性来定义动画的播放状态。

语法如下所示：

```
animation-play-state:取值;
```

animation-play-state 属性取值只有两个：running 和 paused。

表 5-9 为 animation-play-state 属性取值。

表 5-9　animation-play-state 属性取值

属性	说　明
running	播放动画(默认值)
paused	暂停动画

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 animation-play-state 属性</title>
    <style type="text/CSS">
        @-webkit-keyframes mytranslat{
            0%{}
            50%{-webkit-transform:translateX(200px);}
            100%{}
        }
        #div1{
            width:40px;
            height:40px;
            border-radius:20px;
            background-color:red;
            -webkit-animation-name:mytranslate;
            -webkit-animation-timing-function:linear;
            -webkit-animation-duration:3s;
            -webkit-animation-iteration-count:infinite;
        }
        #container{
            display:inline-block;
            width:240px;
            border:1px solid silver;
        }
    </style>
    <script src="../App_js/jquery-1.11.3.min.js" type="text/javascript">
</script>
    <script type="text/javascript">
        $(function () {
            $("#btn_pause").click(function () {
                $("#div1").CSS("-webkit-animation-play-state", "paused");
            });
            $("#btn_run").click(function () {
```

```
                $("#div1").CSS("-webkit-animation-play-state", "running");
            });
        })
    </script>
</head>
<body>
    <div id="container">
        <div id="div1"></div>
    </div>
    <div id="control_btn">
        <input id="btn_pause" type="button" value="暂停"/>
        <input id="btn_run" type="button" value="播放"/>
    </div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-72 所示的，当单击“暂停”按钮时，设置动画的 animation-play-state 属性值为"paused"；当单击“播放”按钮时，设置动画的 animation-play-state 属性值为"running"。这种效果就跟视频播放器中的“暂停”和“播放”的效果一样。

图 5-72　animation-play-state 效果

9）时间外属性 animation-fill-mode

在 CSS3 中，可以使用 animation-fill-mode 属性定义在动画开始之前和动画结束之后发生的事情。

语法如下所示：

```
animation-fill-mode:取值;
```

animation-fill-mode 属性取值如表 5-10 所示。

表 5-10　animation-fill-mode 属性取值

属性	说　明
none	动画完成最后一帧时会反转到初始帧处（默认值）
forwards	动画结束之后继续应用最后的关键帧位置

续表

属性	说　明
backwards	会在向元素应用动画样式时迅速应用动画的初始帧
both	元素动画同时具有 forwards 和 backwards 效果

示例代码如下所示：

```
<html>
<head>
    <title>CSS3 animation - fill - mode 属性</title>
    <style type="text/CSS">
        @ - webkit - keyframes mytranslate
        {
            0 % {}
            100 % { - webkit - transform:translateX(100px);}
        }
        div:not( # container)
        {
            width:40px;
            height:40px;
            border - radius:20px;
            background - color:red;
            - webkit - animation - name:mytranslate;
            - webkit - animation - timing - function:linear;
            - webkit - animation - duration:2s;
        }
        # div2
        {
            - webkit - animation - fill - mode:forwards;
        }
        # div3
        {
            - webkit - animation - fill - mode:backwards;
        }
        # div4
        {
            - webkit - animation - fill - mode:both;
        }
        # container
        {
            display:inline - block;
            width:140px;
            border:1px solid silver;
        }
    </style>
</head>
<body>
    <div id = "container">
        <div id = "div1"></div>
        <div id = "div2"></div>
        <div id = "div3"></div>
```

```
        <div id = "div4"></div>
    </div>
</body>
</html>
```

通过 Chrome 查看该 HTML，效果如图 5-73 所示。这里设置 div4 为 -webkit-animation-fill-mode：both，使得其元素动画同时具有 forwards 和 backwards 效果，设置 div2 为“-webkit-animation-fill-mode：forwards；”，使其动画结束之后继续应用最后的关键帧位置，设置 div3 为“-webkit-animation-fill-mode：backwards；”，使其会在向元素应用动画样式时迅速应用动画的初始帧。

图 5-73　animation-fill-mode 效果

5.2.3　案例实现

CSS 动画

1. 案例分析

对图 5-45 进行分析，结果如图 5-74 所示。

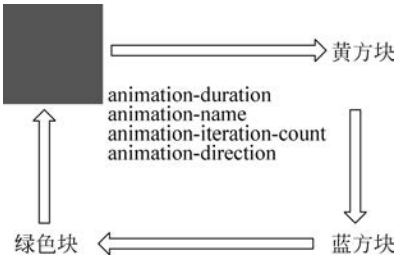


图 5-74　动画分析图

2. 代码实现

```
<style>
div
{
```

```
width:100px;
height:100px;
background:red;
position:relative;

/* Safari and Chrome: */
-webkit-animation-name:myfirst;
-webkit-animation-duration:5s;
-webkit-animation-timing-function:linear;
-webkit-animation-delay:2s;
-webkit-animation-iteration-count:infinite;
-webkit-animation-direction:alternate;
-webkit-animation-play-state:running;

}
@-webkit-keyframes myfirst /* Safari and Chrome */
{
0% {background:red; left:0px; top:0px;}
25% {background:yellow; left:200px; top:0px;}
50% {background:blue; left:200px; top:200px;}
75% {background:green; left:0px; top:200px;}
100% {background:red; left:0px; top:0px;}
}
</style>
```

运行效果需动图，这里无法展示，需输入上述代码自行运行查看。

本章小结

本章的主要知识点如下：

- 盒模型由元素的内容、内边距、边框和外边距组成。
- CSS 中盒模型(Box Model)是分为两种：一种是 W3C 的标准模型，另一种是 IE 的传统模型。
- float 是 CSS 样式中的定位属性，用于设置标签对象的浮动布局。
- display 属性确定一个元素是否应该显示在页面上，以及如何显示。
- overflow 属性来指定其内容不能填充时候该如何处理。
- CSS 有 3 种基本的定位机制：普通流、浮动和绝对定位。
- position 属性规定元素的定位类型。
- 布局一般分为表格布局、框架布局和 DIV+CSS 布局模式。
- 表格布局的优点是：布局容易、快捷而且兼容性好。
- 表格布局的缺点是：改动不方便，彼此之间容易受影响，一旦需要调整工作量会很大。
- 框架由框架和框架集两部分组成。
- 框架是一种常用的网页布局排版工具。框架结构就是把浏览器窗口划分为多个区域，每个区域都可以分别显示不同的网页。
- Web 网页标准构成包括结构、外观和行为 3 部分。

- 用 CSS 布局外观不是最重要的，一个结构良好的 HTML 页面可以通过 CSS 以任何外观表现出来。
- DIV 布局的优点是网页代码精简、提高页面下载速度、表现和内容相分离等；缺点则是过于灵活，比较难控制。
- CSS3 动画效果共 3 个部分：CSS3 变形、CSS3 过渡、CSS3 动画。
- transform 属性用于实现文字或图像的各种变形效果，如位移、缩放、旋转、倾斜等。
- translate()方法将元素沿着水平方向(X 轴)和垂直方向(Y 轴)移动。
- scale()方法用于将元素根据中心原点进行缩放。
- rotate()方法用于将元素相对中心原点进行旋转。
- skew()方法将元素倾斜显示。
- transition 属性用于将元素的某一个属性从“一个属性值”在指定的时间内平滑地过渡到“另外一个属性值”来实现动画效果。
- transition-property 属性用于单独设定过渡动画所要操作的那个属性。
- transition-duration 属性用于单独设置过渡持续的时间。
- transition-timing-function 属性用于定义过渡方式。
- transition-delay 属性用于设置动画开始的延迟时间。
- 动画效果使用 animation 属性来实现。
- 使用 animation 属性定义 CSS3 动画需要两步：定义动画和调用动画。
- 使用@keyframes 规则定义动画。
- animation-name 属性用于调用动画。
- animation-duration 属性用于设置动画的持续时间。
- animation-timing-function 属性用于设置动画的播放方式。
- animation-delay 属性用于定义动画播放的延迟时间。
- animation-iteration-count 属性用于定义动画的播放次数。
- animation-direction 属性用于定义动画的播放方向。
- animation-play-state 属性用于定义动画的播放状态。
- animation-fill-mode 属性用于定义在动画开始之前和动画结束之后发生的事情。