第 3 章　CSS 文字样式设计与速成

文字是一个页面内容的灵魂，因为大多数的页面都是通过文字传达信息的，所以对文字的渲染可以起到对页面更好的修饰作用，而好的文字渲染既可给人带来感官享受，也可使人阅读得更舒服更清晰。

在这一章将开始学习如何控制文字的样式，其中包括 Web 编程的一些主要概念。通过这章的知识点以及实例的分析，使得对 CSS 标签如何控制文字样式有一定的认识，从而达到能够排列出丰富的 CSS 文字效果的目的。

3.1　CSS 3 文字样式

在这一节学习如何为文字添加 CSS 样式，包括 CSS 的一些常用的文字设置。例如，文字的字体、文字的大小、文字的类型、文字的粗细、文字的颜色、链接的样式、文字的布局以及文字的阴影等。另外，还将学习 CSS 3 新增的文字样式的控制方法。

3.1.1　字体

CSS 提供 font-family 属性来控制文字的字体类型，通常使用的汉字字体有宋体、黑体和楷体等，通常使用的外文字体有 Arial 和 Times New Roman 等。

如表 3.1 所示是 Dreamweaver 默认的 font-family 属性的字体集合。

表 3.1　默认的font-family属性的字体集合列表

Dreamweaver默认的font-family属性的字体
宋体
新宋体
Arial, Helvetica, sans-serif
Times New Roman, Times, serif
Courier New, Courier, monospace
Georgia, Times New Roman, Times, serif
Verdana, Arial, Helvetica, sans-serif
Geneva, Arial, Helvetica, sans-serif
黑体
楷体_GB2312
仿宋_GB2312

从上面的表格可以看到，其中有的属性使用了逗号隔开，其意思为：当浏览器没有第一个字体库时，会使用下一个备选的字体库。例如，如果 font-family 属性设置为：Times New

Roman,Times,serif，则浏览器会寻找第一个存在的字体库。

除此以外，Dreamweaver 还提供其他的字体属性，可以通过“编辑字体列表”设置框来增加或者删除默认的字体属性。如图 3.1 所示即为软件“编辑字体列表”的界面。

图 3.1　编辑字体列表

下面是一个设置文字字体的简单示例，通过本示例，可以熟悉常用的字体类型，通常只要知道 5～6 种字体的大概形状就可以了。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312"/>
        <title>
            文字字体设置
        </title>
        <style type="text/css">
            <!--
                div
                {
                    border:#FF0000 2px solid;
                                        /*设置 div 标签的边框颜色和实线大小*/
                    width:300px;        /*设置 div 标签的宽度*/
                    text-align:center;  /*设置 div 标签的文本对齐方式*/
                }
                #font1
                {
                    font-family:"宋体";
                }
                #font2
                {
                    font-family:"黑体";
                }
                #font3
                {
                    font-family:"楷体_GB2312";
                }
                #font4
                {
                    font-family:Arial;
```

```
                }
                #font5
                {
                    font-family:"Times New Roman";
                }
            -->
        </style>
    </head>
    <body>
        <div id="font1">
            宋体
        </div>
        <div id="font2">
            黑体
        </div>
        <div id="font3">
            楷体
        </div>
        <div id="font4">
            Arial
        </div>
        <div id="font5">
            Times New Roman
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

可以发现，HTML 代码部分首先创建了 5 个 DIV 标签元素，然后在 CSS 代码里面根据 ID 号给各个 DIV 设置文字字体。

除了 font-family 可以设置字体外，还可以通过 font 属性来设置字体，因为 font 属性可以包含多个文字属性的设置，包括文字大小、文字类型以及文字粗细等。

如图 3.2 所示为这段代码的运行结果，即对文字字体进行设置。

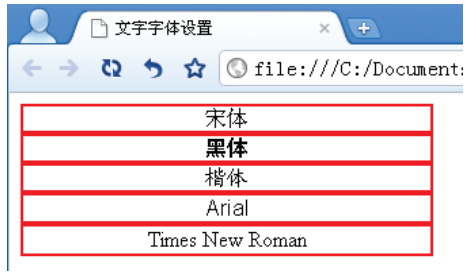


图 3.2　文字字体设置

3.1.2　大小

CSS 提供 font-size 来控制字体的大小，font-size 允许使用百分号形式或者单位形式来进行字体大小的设置。使用单位形式进行设置举例：

```
    font-size:   20px;
```

使用百分号进行设置举例：

```
    font-size:   200%;
```

使用百分号形式时，是相对于父节点的 font-size 的大小进行设置的，一般情况下 body 父节点的字体大小为 16 像素。

除此之外，font-size 还支持以下属性值的设置：xx-small（最小）、x-small（较小）、small（小）、 medium（中等）、large（大）、x-large（较大）以及 xx-large（超大）。

需要强调的是，font-size 还支持相对大小的属性值的设置，即 larger 以及 smaller。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title>
            文字大小设置
        </title>
        <style type="text/css">
            <!--
                body
                {
                    font-size:18px;
                }
                div
                {
                    border:#FF0000 2px solid;
                                        /*设置 div 标签的边框颜色和实线大小*/
                    width:300px;        /*设置 div 标签的宽度*/
                    text-align:center;  /*设置 div 标签的文本对齐方式*/
                }
                #font1
                {
                    font-size:100%;
                }
                #font2
                {
                    font-size:16px;
                }
                #font3
                {
                    font-size:200%;
                }
                #font4
                {
                    font-size:large;
                }
                #font5
                {
                    font-size:larger;
                }
            -->
        </style>
    </head>
    <body>
                 
                 
            BODY  18px
        <div id="font1">
            DIV  100%
        </div>
        <div id="font2">
            DIV  16px
        </div>
        <div id="font3">
            DIV  200%
        </div>
```

```
        <div id="font4">
            DIV  large
        </div>
        <div id="font5">
            DIV  larger
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

（1）首先为 HTML 代码添加 5 个 DIV 标签元素，并依次将其 id 命名为 font1、font2、font3、font4 和 font5。

（2）添加 CSS 代码部分，依次将 body 以及 5 个 DIV 标签的 font-size 设置为 18px、100%、16px、200%、large 和 larger。

因为 body 是 DIV 的直接父节点，所以 font1 的 font-size 的值也为 18px，font3 的 font-size 的值为 36px。其次，large 的效果和 18px 的效果一样。另外，因为父节点的 font-size 为 18px，font5 为 larger，使得其效果和 22px 的效果一样，所以可以看出，使用 larger 后一般增大 4 个像素。在该代码中出现了字符串 ，它表示的是一个空格，所以程序中写了 5 个 ，就表示连着出现了 5 个空格。

如图 3.3 所示即为这段代码的运行结果，即对文字大小进行设置，可以看到文字大小的变化。

图 3.3　文字大小设置

3.1.3　类型

CSS 提供 font-style 来设置字体的类型，主要是针对斜体或者正体字型的设置。CSS 允许 4 种 font-style 属性值的设置，表 3.2 即为 font-style 属性值的设置列表。

表 3.2　font-style属性值的设置列表

值	描　述
normal	默认值。浏览器显示一个标准的字体样式
italic	浏览器会显示一个斜体的字体样式
oblique	浏览器会显示一个倾斜的字体样式
inherit	规定应该从父元素继承字体样式

其中，italic 以及 oblique 都是将文字设置为斜体，它们的样式效果是一样的。以下代码是 font-style 属性的简单使用举例。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title>
            文字类型设置
```

```
        </title>
        <style type="text/css">
            <!--
                body
                {
                    font-size:20px;       /*设置 body 标签的字体大小*/
                    font-style:italic;    /*设置 body 标签的字体风格*/
                }
                div
                {
                    border:#FF0000 2px solid;
                                          /*设置 div 标签的边框颜色和实线大小*/
                    width:300px;          /*设置 div 标签的宽度*/
                    text-align:center;    /*设置 div 标签的文本对齐方式*/
                }
                #font1
                {
                    font-style:normal;
                }
                #font2
                {
                    font-style:italic;
                }
                #font3
                {
                    font-style:oblique;
                }
                #font4
                {
                    font-style:inherit;
                    font-size-adjust:
                }
            -->
        </style>
    </head>
    <body>
                 
                 
            BODY 斜体
        <div id="font1">
            DIV  正常
        </div>
        <div id="font2">
            DIV  斜体 1（italic）
        </div>
        <div id="font3">
            DIV  斜体 2（oblique）
        </div>
        <div id="font4">
            DIV  继承父类（inherit）
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

为 HTML 代码添加 4 个 DIV 标签元素，并依次将其 id 命名为 font1、font2、font3 和 font4。其中因为 body 标签设置为 italic，而 font4 设置为 inherit，所以 font4 的文字类型实际上也为 italic。

如图 3.4 所示为这段代码的运行结果，即对文字类型进行设置，可以看到斜体和正常体等字体。

图 3.4　文字类型设置

3.1.4　粗细

CSS 提供 font-weight 属性来控制文字的粗细，其值可以是指定的数值或者关键字，表 3.3 即为 font-weight 属性值的设置列表。

表 3.3　font-weight属性值的设置列表

值	描　　述
normal	默认值。不加粗
bold	粗体
bolder	比父节点字体粗
lighter	比父节点字体细
100～900（9级粗细字体）	100、200、300、400、500、600、700、800、900

一般的现代浏览器只提供两级粗细字体，所以，从 100～500 都为不加粗字体，从 600～900 都为加粗字体。

以下代码展示了字体粗细的一般设置方式。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title>
            文字粗细设置
        </title>
        <style type="text/css">
            <!--
                body
                {
                    font-size:20px;        /*设置 body 标签的字体大小*/
                    font-weight:600;       /*设置 body 标签的文本粗细*/
                }
                div
                {
                    border:#FF0000 2px solid;
```

```
                                        /*设置 div 标签的边框颜色和实线大小*/
                    width:300px;        /*设置 div 标签的宽度*/
                    text-align:center;  /*设置 div 标签的文本对齐方式*/
                }
                #font1
                {
                    font-weight:normal;
                }
                #font2
                {
                    font-weight:bold;
                }
                #font3
                {
                    font-weight:bolder;
                }
                #font4
                {
                    font-weight:lighter;
                }
                #font5
                {
                    font-weight:100;
                }
                #font6
                {
                    font-weight:900;
                }
            -->
        </style>
    </head>
    <body>
                 
                 
            BODY 600
        <div id="font1">
            DIV  normal
        </div>
        <div id="font2">
            DIV  bold
        </div>
        <div id="font3">
            DIV  bolder
        </div>
        <div id="font4">
            DIV  lighter
        </div>
        <div id="font5">
            DIV  100
        </div>
        <div id="font6">
            DIV  900
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

（1）依次添加 DIV 标签元素，并将它们的 id 分别命名为 font1、font2、font3、font4、font5 和 font6。

（2） 在 CSS 代码部分，首先把 body 标签的 font-weight 设置为 600，然后把 font1、font2、font3、font4、font5 和 font6 的该属性分别设置为 normal（相当于 400）、bold（相当于 700）、bolder、lighter、100 和 900。

（3）按照规定，因为其父节点 body 的 font-weight 属性为 600，且 font3 和 font4 的 font-weight 属性分别为 bolder（实际应该为 700）以及 lighter（实际应该为 500），然而实际运行时只有两种粗体样式。

如图 3.5 所示为这段代码的运行结果，即对文字粗细进行设置。其中只有两种粗细样式，并且设置为 600 和设置为 900 的效果是一样的，所以一般的浏览器只有两级粗细字体。

图 3.5　文字粗细设置

3.1.5　颜色

CSS 提供 color 属性来控制文字的颜色，其值可以是关键字形式、十六进制形式、RGB 形式以及 inherit 形式。另外 HTML 的 font 标签里面也具有 color 属性用于设置文字颜色，然而在 XHTML 1.0 的 Strict DTD 中，不支持 color 属性，需要使用 CSS 的 color 属性来代替。

使用关键字形式进行设置举例：

```
color:  red
```

使用十六进制形式进行设置举例：

```
color:  #ff0000
```

使用 RGB 形式进行设置举例：

```
color:  RGB(255,0,0)
```

以下代码展示了字体颜色的一般设置方式。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title>
            文字颜色设置
        </title>
        <style type="text/css">
            <!--
                body
                {
                    font-size:20px;              /*设置 body 标签的字体大小*/
                    color: yellow;               /*设置 body 标签的颜色*/
                }
                div
```

```
                {
                    border:#FF0000 2px solid; /*设置 div 标签的边框颜色和
                                               实线大小*/
                    width:300px;              /*设置 div 标签的宽度*/
                    text-align:center;        /*设置 div 标签的文本对齐方式*/
                    font-size:20px;
                }
                #font1
                {
                    color:red;
                }
                #font2
                {
                    color:#00ff00;
                }
                #font3
                {
                    color:RGB(0,0,255);
                }
                #font4
                {
                    color:inherit;
                }
            -->
        </style>
    </head>
    <body>
                  
                 
            黄色
        <div id="font1">
            红色
        </div>
        <div id="font2">
            绿色
        </div>
        <div id="font3">
            蓝色
        </div>
        <div id="font4">
            父节点的颜色（黄色）
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

（1）依次添加 4 个 DIV 标签元素，将它们的 id 分别命名为 font1、font2、font3 和 font4。

（2）在 CSS 代码部分，首先把 body 标签的 color 属性设置为 yellow，然后把 font1、font2、font3 和 font4 的 color 属性设置为 red、#00ff00、RGB(0,0,255)以及 inherit。

如图 3.6 所示为这段代码的运行结果，即对文字颜色进行设置。因为 body 为 font4 的父节点且 body 的属性为 yellow，且 font4 的 color 属性为 inherit，

图 3.6　文字颜色设置

所以 body 和 font4 的字体颜色是一样的。

3.1.6　小型大写字体

CSS 提供 font-variant 属性来控制文字的大小写控制，但是该属性较少使用，读者只需了解即可。表 3.4 即为 font-variant 属性值的设置列表。

表 3.4　font-variant属性值的设置列表

值	描　述
normal	默认值。大小写不变
small-caps	英文小型大写字体
inherit	和父节点一致

下面是 font-variant 属性的简单使用举例。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title>
            文字大小写设置
        </title>
        <style type="text/css">
            <!--
                body
                {
                    font-variant:small-caps;   /*设置 body 标签的字体属性*/
                }
                div
                {
                    border:#FF0000 2px solid;  /*设置 div 标签的边框颜色和实线
                                                大小*/
                    width:300px;               /*设置 div 标签的宽度*/
                    text-align:center;         /*设置 div 标签的文本对齐方式*/
                }
                #font1
                {
                    font-variant:normal;
                }
                #font2
                {
                    font-variant:inherit;
                }
            -->
        </style>
    </head>
    <body>
                  
                 
            this is Font style(small-caps)
        <div id="font1">
            this is Font style(normal)
        </div>
```

```
        <div id="font2">
            this is Font style(inherit)
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

（1）依次添加两个 DIV 标签元素，将它们的 id 分别命名为 font1 和 font2。

（2）在 CSS 代码部分，首先把 body 标签的 font-variant 属性设置为 small-caps，然后把 font1 和 font2 的 font-variant 属性设置为 normal 和 inherit。

如图 3.7 所示为这段代码的运行结果，即对文字小型大写字母进行设置。因为 font2 的父节点 body 的 font-variant 属性为 small-caps，且 font2 的 font-variant 属性为 inherit，所以 body 和 font2 都为小型大写字体。

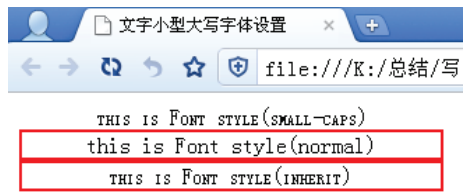


图 3.7　文字小型大写字体设置

3.1.7　链接

CSS 提供 HTML 的链接（瞄和 anchor）标签<a>以及它的伪类选择符的样式。表 3.5 即为瞄标签和伪类选择符的属性值的设置列表。

表 3.5　瞄标签和伪类选择符的属性值的设置列表

值	描　述
link	瞄标签的一般状态下使用该伪类的CSS样式
visited	瞄标签的链接被单击后使用该伪类的CSS样式
active	瞄标签的链接被单击到释放之间使用该伪类的CSS样式
hover	鼠标在瞄标签的链接区域停留时使用该伪类的CSS样式

伪类选择符的使用是如下形式：

```
选择符：　伪类选择符　{　属性：　属性值；　属性：　属性值；　……}
```

其中的选择符可以是标签选择符、ID 选择符或者 CLASS 选择符。以下是一个伪类选择符的使用示例。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
        <title>
            链接伪类的样式设置
        </title>
        <style type="text/css">
            <!--
                a
                {
                    border:#FF0000 2px solid;    /*设置 a 标签的边框颜色和实线
                                                 大小*/
```

```
                    text-align:center;          /*设置 a 标签的文本对齐方式*/
                }
                a:link
                {
                    background-color:#FFFF00;
                }
                a:visited
                {
                    background-color:#FF0000;
                }
                a:hover
                {
                    background-color:#0000FF;
                }
                a:active
                {
                    background-color:#00FF00;
                }
            -->
        </style>
    </head>
    <body>
        <a href="#">
            链接 1
        </a>
        <a href="#">
            链接 2
        </a>
        <a href="#">
            链接 3
        </a>
        <a href="#">
            链接 4
        </a>
    </body>
</html>
```

对上述代码进行剖析如下：

首先，上述代码先创建 4 个瞄点。接着添加所有瞄点标签选择符的公共样式：“border:#FF0000 2px solid;”和“text-align:center;”。这样，该样式将作用于所有<a>标签及其伪类选择符的样式。

然后，为瞄点添加 CSS 伪类选择符 link、visited、hover 和 active 的相应样式。因此，当有如下鼠标动作时，会产生以下的样式效果：

- 标签<a>上没有任何鼠标事件时，将显示 link 伪类的样式效果，即黄色背景。
- 标签<a>上有鼠标移动事件时，将显示 hover 伪类的样式效果，即蓝色背景。
- 标签<a>上有鼠标按下（没放开）事件时，将显示 active 伪类的样式效果，即青色背景。
- 标签<a>上有鼠标按下放开事件时，将显示 visited 伪类的样式效果，即红色背景。

如图 3.8 所示为这段代码的运行结果，即对文字链接伪类的样式进行设置。“链接 4”被鼠标单击过，所以呈现红色；“链接 2”正被鼠标按下，所以呈现青色并带有微小的虚线效果；“链接 1”

图 3.8　链接伪类的样式设置

和“链接 3”没有被鼠标单击，所以呈现黄色。

3.1.8　布局

涉及标签内部内容的布局，需要了解这些样式属性：text-indent、text-align、word-spacing、text-transform、text-decoration 和 white-space。

1．text-indent

text-indent 的功能是控制内部文字第一行的缩进。text-indent 的值可以是单位形式的数值，可以是百分号形式（基于父节点的宽度）的数值，也可以是 inherit。

以下是 text-indent 的简单使用的例子。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title>
            text-indent
        </title>
        <style type="text/css">
            <!--
                div
                {
                    border:#FF0000 2px solid; /*设置 div 标签的边框颜色和实线
                                              大小*/
                    width:400px;              /*设置 div 标签的宽度*/
                }
                #D2
                {
                    text-indent:20px;         /*控制内部#D2 文字第一行的缩进*/
                }
            -->
        </style>
    </head>
    <body>
        <div id="D1">
        1.text-indent 的使用，text-indent 的使用.
        </div>
        <div id="D2">
        2.text-indent 的使用，text-indent 的使用.
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

首先，依次添加两个 DIV 标签元素，将它们的 id 分别命名为 D1 和 D2。

然后，对 D1 不设置 text-indent，对 D2 设置 text-indent 为 20px。D1 和 D2 显示的文本都是相同的。

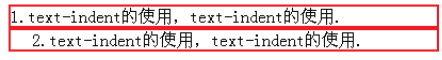


图 3.9　text-indent 的使用效果

如图 3.9 所示为这段代码的运行结果，即使用

text-indent 的效果对比。由于 D1 不设置 text-indent，所以为默认值 0，没有缩进；D2 设置了 text-indent，所以如图 3.9 所示的第二行，存在了一定的缩进。

2．text-align

text-align 的功能是控制内部文字的横向排布。在前面的章节里已经频繁地使用了该属性，text-indent 的值可以是 center、left、right、justify 以及 inherit，表 3.6 即为 text-align 的值及其功能描述的设置列表。

表 3.6　text-align的值及其功能的设置列表

值	描　　述
left	默认值。内容往左边靠
center	瞄标签的链接被单击后使用该伪类的CSS样式
right	内容往右边靠
justify	内容向两边伸展对齐（英文内容效果比中文明显）
inherit	和父节点一致

3．word-spacing

word-spacing 的功能是控制文字间空格的距离。实际上，word-spacing 属性的值是给每个空格添加的增量。word-spacing 属性的值可以是单位形式的数值、normal（默认值）以及 inherit（继承父节点的该属性）。

以下是 word-spacing 简单使用的例子。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title>
            word-spacing
        </title>
        <style type="text/css">
            <!--
                div
                {
                    border:#FF0000 2px solid; /*设置 div 标签的边框颜色和实线
                                               大小*/
                    width:400px;              /*设置 div 标签的宽度*/
                }
                #D1
                {
                    word-spacing:8px;         /*控制#D1 文字间空格的距离*/
                }
                #D2
                {
                    word-spacing:-8px;        /*控制#D2 文字间空格的距离*/
                }
            -->
        </style>
    </head>
    <body>
        <div id="D1">
        This is a paragraphe
        </div>
```

```
        <div id="D2">
        This is a paragraphe
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

首先，依次添加两个 DIV 标签元素，将它们的 id 分别命名为 D1 和 D2。

然后，对 D1 的 word-spacing 属性设置为 8px，对 D2 的 word-spacing 属性设置为-8px。D1 和 D2 显示的文本都是相同的。

如图 3.10 所示为这段代码的运行结果，即使用 word-spacing 的效果对比。由于 D1 设置的 word-spacing 为 8px，所以如图 3.10 所示的第一行，产生更大的空格效果；D2 设置的 word-spacing 为 -8px，所以如图 3.10 所示的第二行，产生没有空格的效果。

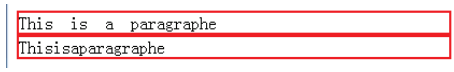


图 3.10　word-spacing 的使用效果

4．text-transform

text-transform 的功能是控制英文的大小写。text-transform 属性的值可以是 none、capitalize、uppercase、lowercase 和 inherit。表 3.7 即为 text-transform 的值及其功能描述的设置列表。

表 3.7　text-transform的值及其功能的设置列表

值	描述
none	默认值。大小写不改变
capitalize	单词首字母必定大写、其他小写
uppercase	全部字母大写
lowercase	全部字母小写
inherit	和父节点一致

以下是 text-transform 的简单使用的例子。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title>
            text-transform
        </title>
        <style type="text/css">
            <!--
                body
                {
                    text-transform:lowercase;    /*设置 body 标签文本中的字母全
                                                 部小写*/
                }
                div
                {
                    border:#FF0000 2px solid;    /*设置 div 标签的边框颜色和实
                                                 线大小*/
```

```
                    width:200px;                    /*设置 div 标签的宽度*/
                }
                #D1
                {
                    text-transform:capitalize;  /*控制英文的大小写*/
                }
                #D2
                {
                    text-transform:uppercase;
                }
                #D3
                {
                    text-transform:inherit;
                }
            -->
        </style>
    </head>
    <body>
        THIS IS A PARAGRAPHE
        <div id="D1">
        THIS is a paragraphe
        </div>
        <div id="D2">
        this is a paragraphe
        </div>
        <div id="D3">
        THIS IS A PARAGRAPHE
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

首先，依次添加 3 个 DIV 标签元素，将它们的 id 分别命名为 D1、D2 和 D3。

然后，对 D1 的 text-transform 属性设置为 capitalize，对 D2 的 text-transform 属性设置为 uppercase，对 D3 的 text-transform 属性设置为 inherit。

如图 3.11 所示为这段代码的运行结果，即使用 text-transform 的效果对比。因为 D3 继承了它的父节点的属性，所以如图 3.11 所示第一行和第四行，body 标签以及第三个 DIV 标签都是小写英文字体；D1 是首字母大写的属性，所以如图 3.11 所示第二行，是首字母大写的英文字体；D2 是全部字母大写的属性，所以如图 3.11 所示第三行，是所有字母大写的英文字体。

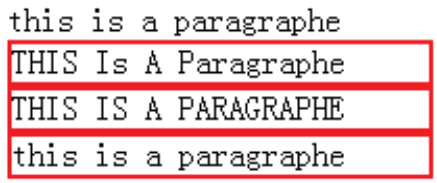


图 3.11　text-transform 的使用

5．text-decoration

text-decoration 的功能是添加额外的文字修饰，例如下划线和闪烁效果。text-decoration 属性的值可以是 none、underline、overline、line-through、blink 以及 inherit。表 3.8 即为 text-decoration 的值及其功能描述的设置列表。

表 3.8　text-decoration的值及其功能的设置列表

值	描　述
none	默认值。没有额外效果

续表

值	描　述
underline	文字底部添加一条横线
overline	文字顶部添加一条横线
line-through	文字中间添加一条横线
blink	文字闪烁效果（有的浏览器不支持该属性值）
inherit	和父节点一致

以下是 text-decoration 的简单使用的例子。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title>
            text-decoration
        </title>
        <style type="text/css">
            <!--
                body
                {
                    text-decoration:line-through; /*文字中间添加一条横线*/
                }
                #DIV1
                {
                    text-decoration:underline;    /*文字底部添加一条横线*/
                }
                #DIV2
                {
                    text-decoration:blink;        /*文字闪烁效果*/
                }
                #DIV3
                {
                    text-decoration:overline;     /*文字顶部添加一条横线*/
                }
            -->
        </style>
    </head>
    <body>
        BODY text-decoration 的使用
        <div id="DIV1">
        DIV1 text-decoration 的使用
        </div>
        <div id="DIV2">
        DIV2 text-decoration 的使用
        </div>
        <div id="DIV3">
        DIV3 text-decoration 的使用
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

该程序依次添加 3 个 DIV 标签元素，即 D1、D2 和 D3。它们的 text-decoration 属性的值依次为：underline、blink 和 overline。

如图 3.12 所示为这段代码的运行结果，即使用 text-decoration 的效果对比。因为第二个 DIV 的 text-decoration 属性的值为 blink，即具有闪烁效果，所以如图 3.12 所示的第三行，该行的文字正好消失；第一个和第三个 DIV 的 text-decoration 属性的值分别为 underline 和 overline，所以图 3.12 所示的第二行和第四行的文字分别有底划线和上划线。另外值得注意的是，body 的 text-decoration 属性值为 line-through，虽然添加的 3 个 DIV 并不是以 inherit 来继承父节点 body 的该属性，但也具有 line-through 值的效果，即文字中间添加了一条横线。

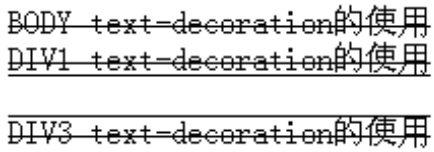


图 3.12　text-decoration 的使用

6．white-space

white-space 的功能是控制文本当中的回车和空格等字符的显示。white-space 属性的值可以是 pre-line、normal、nowrap、pre 以及 pre-wrap。表 3.9 即为 white-space 的值及其功能描述的设置列表。

表 3.9　white-space的值及其功能的设置列表

值	空白符	换行符	是否自动换行
pre-line	合并	保留	是
normal	合并	忽略	是
nowrap	合并	忽略	否
pre	保留	保留	否
pre-wrap	保留	保留	是

以下是 white-space 的简单使用的例子。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title>
            white-space
        </title>
        <style type="text/css">
            <!--
                div
                {
                    border:#FF0000 1px solid;   /*设置 div 标签的边框颜色和实
                                                线大小*/
                    width:300px;                /*设置 div 标签的宽度*/
                }
                #DIV1
                {
                    white-space:nowrap;         /*合并文本当中的回车和空格等
                                                字符的显示*/
                }
                #DIV2
                {
                    white-space:pre-wrap;       /*保留文本当中的回车和空格等
```

```
                                                    字符的显示*/
                }
            -->
        </style>
    </head>
    <body>
        <div id="DIV1">
        DIV1     white-space 的使用     white-space 的使用
        white-space 的使用
        </div>
        <div id="DIV2">
        DIV2     white-space 的使用     white-space 的使用
        white-space 的使用
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

该程序依次添加两个 DIV 标签元素，即 DIV1 和 DIV2。它们包含的文字内容一模一样，不仅文字一模一样，连空格个数和换行符位置也是一模一样。

如图 3.13 所示为这段代码的运行结果，即使用 white-space 的效果对比。因为 DIV1 的 white-space 的属性值是 nowrap，即合并空格、忽略换行符、不执行自动换行；DIV2 的 white-space 属性值是 pre-wrap，即不合并空格、不忽略换行符、执行自动换行。所以，尽管两个 DIV 的 HTML 中的内容一样，通过 CSS 的控制，两个 DIV 的文本却表现出截然不同的格式。

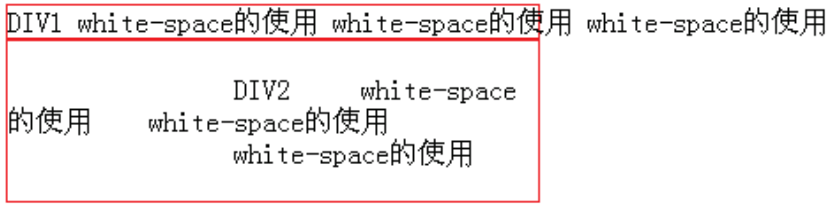


图 3.13　white-space 的使用

3.1.9　文字阴影

CSS 3 提供 text-shadow 来控制文字的阴影，该属性具有 4 个分量，分别表示横坐标偏移位置、纵坐标偏移位置、模糊半径和阴影颜色。

text-shadow 的使用格式如下：

```
text-shadow: 横坐标偏移位置　纵坐标偏移位置　模糊半径　阴影颜色，横坐标偏移位置
纵坐标偏移位置　模糊半径　阴影颜色，……
```

可以在同一个 text-shadow 标识下添加多个阴影样式。

以下是 text-shadow 属性的简单使用的例子。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title>
            文字阴影效果
        </title>
        <style type="text/css">
            <!--
                div
                {
                    border:#FF0000 1px solid;   /*设置 div 标签的边框颜色和实
                                                线大小*/
                    width:600px;                /*设置 div 标签的宽度*/
                    font-family:"黑体";         /*设置 div 标签的字体类型*/
                    font-size:50px;             /*设置 div 标签的字体大小*/
                }
                #DIV1
                {
                    color:#FF0000;
                    text-shadow: 2px 2px 0px #000;
                                                /*只有一个阴影*/
                }
                #DIV2
                {
                    color:#FFFFFF; /*颜色*/
                    text-shadow: 0 0 10px #fff, 0 0 15px #fff, 0 0 40px
                    #ff00de,0 0 40px #ff00de;   /*含有 4 个阴影*/
                }
            -->
        </style>
    </head>
    <body>
        <div id="DIV1">
        CSS3 文字阴影效果 1
        </div>
        <div id="DIV2">
        CSS3 文字阴影效果 2
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

该程序依次添加两个 DIV 标签元素，即 DIV1 和 DIV2。第一个 DIV 标签的 text-shadow 属性设置为“2px 2px 0px #000”，即 DIV1 的纵、横坐标分别偏移 2 个像素，模糊半径为 0（即不模糊），黑色阴影。

第二个 DIV 标签的 text-shadow 属性为“0 0 10px #fff, 0 0 15px #fff, 0 0 40px #ff00de,0 0 40px #ff00de”，即 DIV2 具有 4 个阴影效果，以致产生荧光效果。

如图 3.14 所示为这段代码的运行结果，即使用 text-shadow 的效果对比。由于 DIV1 的 text-shadow 的属性值设置，所以图 3.14 的第一行显示黑色阴影；由于 DIV2 的 text-shadow 的属性值设置，所以图 3.14 的第二行显示淡黄和粉红的荧光效应。

图 3.14　text-shadow 的使用

其实 IE 浏览器不支持 text-shadow 属性，而是提供 shadow 和 dropshadow 两个滤镜来实现阴影效果。

shadow 滤镜的使用格式如下：

```
filter:shadow(Color=颜色值,Direction=和垂直向上方向的夹角 ,Strength=阴影宽度 );
```

dropshadow 滤镜的使用格式如下：

```
filter:dropshadow(color=颜色值, offx=横坐标偏移像素值, offy=纵坐标偏移像素值,
Positive=布尔阴影类型);
```

以下是 shadow 滤镜和 dropshadow 滤镜属性的简单使用的例子。

```
<!DOCTYPE htm  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title>
            阴影滤镜的使用
        </title>
        <style type="text/css">
            <!--
                div
                {
                    border:#FF0000 1px solid;   /*设置 div 标签的边框颜色和实
                                                线大小*/
                    width:400px;                /*设置 div 标签的宽度*/
                    font-family:"黑体";         /*设置 div 标签的字体类型*/
                    font-size:30px;             /*设置 div 标签的字体大小*/
                    color:#FF0000;              /*设置 div 标签的颜色*/
                }
                #DIV1
                {
                    filter:shadow(Color=#000000,Direction=45 ,
                    Strength=5 );               /*shadow 滤镜的使用*/
                }
                #DIV2
                {
                    filter:dropshadow(color=#000000, offx=10,
                    offy=5, Positive=1);        /*dropshadow 滤镜的使用*/
                }
                #DIV3
                {
                    filter:dropshadow(color=#000000, offx=10,
                    offy=5, Positive=0);
                }
            -->
```

```
        </style>
    </head>
    <body>
        <div id="DIV1">
        shadow 滤镜阴影效果
        </div>
        <div id="DIV2">
        dropshadow 滤镜阴影效果 1
        </div>
        <div id="DIV3">
        dropshadow 滤镜阴影效果 2
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

该程序依次添加 3 个 DIV 标签元素，即 DIV1、DIV2 和 DIV3。第一个 DIV 使用 shadow 滤镜，产生一个黑色的、垂直偏移 45° 且 5 个像素宽的连续投影。第二个 DIV 和第三个 DIV 使用 dropshadow 滤镜，它们都是产生一个黑色的且 10 个像素的横坐标偏移量和 5 个像素的纵坐标偏移量的阴影。不过不同点在于，第二个 DIV 是为所有不透明元素建立的阴影；第三个 DIV 是为所有透明元素建立的可见阴影。

如图 3.15 所示为这段代码的运行结果，即使用阴影滤镜的效果对比。由于第二个 DIV 和第三个 DIV 的 Positive 参数值刚好布尔相反，所以它们的阴影效果也恰好是相反的，即如图 3.15 所示的第二行和第三行。此外，还可以从图 3.15 看到，阴影滤镜效果的同时也会产生边框的阴影。

图 3.15　阴影滤镜的使用

3.1.10　文字变换速度控制

CSS 3 提供 transition 属性来控制某些事件发生后标签样式切换的动画效果。对于某些不支持 CSS 3 的浏览器，如 IE 6、IE 5，可以使用 JavaScript 进行控制。

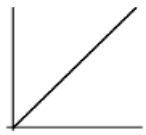
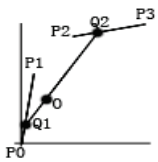
不同的浏览器也可以使用自定义的 transition 属性。例如，Safari 和 Chrome 浏览器使用-webkit-transition 属性，Firefox 浏览器使用-moz-transition 属性，IE9 浏览器使用-ms-transition 属性，Opera 浏览器使用-o-transition 属性。

transition 属性的设置如下：

```
transition: CSS 属性的名字　变化时间　变化类型　等待时间，CSS 属性的名字　变化时间　变化类型　等待时间，…..
```

其中“CSS 属性的名字”（对应于属性 transition-property）填写的是该标签的 CSS 样式属性，如 background-color、border 以及 color 等等；“变化时间”（对应于属性 transition-duration）填写的是变化的用时，如 1s、500ms 分别表示变化用时 1 秒、500 微秒；“等待时间”（对应于属性 transition- delay）填写的是开始变化前的延时，如 1s、500ms 分别表示开始变化前延时 1 秒、500 微秒；“变化类型”（对应于属性 transition-timing-function）填写的是变换的控制函数，表 3.10 即为这些函数的描述列表。

表 3.10　变换的控制函数的描述列表

变化类型	等价效果	输出量×时间 曲线图
ease	cubic-bezier (0.25, 1.0, 0.25, 1.0)	
linear	cubic-bezier (0.0, 0.0, 1.0, 1.0) cubic-bezier (1.0, 1.0, 1.0, 1.0) cubic-bezier (0.0, 0.5, 1.0, 0.5) ……..	
ease-in	cubic-bezier (0.42, 0, 1.0, 1.0)	
ease-out	cubic-bezier (0, 0, 0.58, 1.0)	
ease-in-out	cubic-bezier (0.42, 0, 0.58, 1.0)	
cubic-bezier	cubic-bezier (x1, y1, x2, y2)	

其中，二次贝塞尔曲线（bezier）是这样的曲线：存在两条线段 P0P1 和 P2P3，取 P0P1 和 P2P3 的相等比例的点 Q1 和 Q2，再取 Q1Q2 的相等比例的 O 点，这些所有的 O 点就组成了二次贝塞尔曲线。其中 cubic-bezier 函数的 4 个参数分别是 P1 的 x1、y1 以及 P2 的 x2、y2，而 P0 的坐标为(0,0)，P3 的坐标为(1,1)。

以下是关于文字变换速度控制的一个简单示例。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title>
            文字变换速度控制
        </title>
        <style type="text/css">
            <!--
                div
                {
                    border:#FF0000 1px solid;   /*设置 div 标签的边框颜色和实
                                                线大小*/
                    width:400px;                /*设置 div 标签的宽度*/
                    font-family:"黑体";         /*设置 div 标签的字体类型*/
                    font-size:30px;             /*设置 div 标签的字体大小*/
                }
                #DIV1
                {
                    color:#FF0000;
                    background-color:#FFFFFF;
                    -webkit-transition: background-color 1s ease-in-out 0s,
                    color 1s ease-in-out 0s; /*用于 Safari 和 Chrome 浏览器*/
                    -moz-transition: background-color 1s ease-in-out 0s,
                    color 1s ease-in-out 0s;/*用于 FireFox 浏览器*/
                    -ms-transition: background-color 1s ease-in-out 0s,
                    color 1s ease-in-out 0s;/*用于 IE 9 浏览器*/
                    -o-transition: background-color 1s ease-in-out 0s,
                    color 1s ease-in-out 0s;/* 用于 Opera 浏览器 */
                    transition: background-color 1s ease-in-out 0s, color
                    1s ease-in-out 0s;      /* CSS3 标准 */
                }
                #DIV1:hover
                {
                    color:#00FF00;
                    background-color:#000000;
                }
            -->
        </style>
    </head>
    <body>
        <div id="DIV1">
        文字变换速度控制
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

本代码只含有一个 DIV 标签，DIV1 的 transition 的值为“background-color 1s ease-in-out 0s, color 1s ease-in-out 0s”，表示 background-color 和 color 两个属性无延时（延时 0 秒）且在 1 秒时间内以 ease-in-out 模式进行样式的变换。如果要将所有样式都以 ease-in-out 模式（变化由慢到快再到慢）进行样式变换，则 transition 的值可以写为“all 1s ease-in-out 0s”。

如图 3.16 所示为这段代码的运行结果，即使用 transition 属性的效果。

图 3.16　transition 属性的使用

3.1.11　文字大小变化

CSS 3 提供 transform 属性来控制标签样式的某些变化，例如大小、旋转、角度及位置等要素。transform 支持 rotate、scale、skew、translate 和 matrix 这些变换属性值，表 3.11 即为这些属性的变换举例和功能的描述列表。

表 3.11　transform属性的变换举例和功能的描述列表

变换属性	相关变换属性/举例	功能描述
rotate(角度)	举例：rotate(30deg)	按指定的角度旋转，如果设置的值为正数表示顺时针旋转，如果设置的值为负数，则表示逆时针旋转
scale(数值[,数值])	scaleX(数值) scaleY(数值) 举例：scale(2)	水平方向和垂直方向同时缩放（也就是X轴和Y轴同时缩放）；scaleX仅在水平方向缩放（X轴缩放）；scaleY仅在垂直方向缩放（Y轴缩放）
skew(角度[,角度])	skewX(角度) skewY(角度) 举例：skew(30deg)	元素在水平和垂直方向同时扭曲（X轴和Y轴同时按一定的角度值进行扭曲变形）；skewX仅使元素在水平方向扭曲变形（X轴扭曲变形）；skewY仅使元素在垂直方向扭曲变形（Y轴扭曲变形）
translate(数值[, 数值])	translateX(数值) translateY(数值) 举例： translate (30px)	水平方向和垂直方向同时移动（也就是X轴和Y轴同时移动）；translateX仅在水平方向移动（X轴移动）；translateY仅在垂直方向移动
matrix(数值, 数值, 数值, 数值, 数值, 数值)	举例： matrix(1,0,0,1,100,100); (形状不变，纵向横向各自偏移100px)	进行样式的重新映射，前两个参数是第一个向量的变换系数，中间两个参数是第二个向量的变换系数，最后两个参数是样式的偏移位置

其中，matrix 执行的是这样的变换：一般的样式会在(1,0)和(0,1)两个方向向量上面进行映射，当通过 matrix 变换后会重新映射到新的方向向量上。例如，如果使用 matrix(M11, M12, M21, M22, x, y)，那么新的方向向量为(M11,M12)和(M21,M22)，并且按像素为单位沿(x,y)移动。

以下是关于文字大小变化的一个简单示例。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
    <title>
        文字大小变化
    </title>
    <style type="text/css">
        <!--
            div
```

```
            {
                border:#FF0000 1px solid;/*设置 div 标签的边框颜色和实线大小*/
                width:400px;              /*设置 div 标签的宽度*/
                font-family:"黑体";       /*设置 div 标签的字体类型*/
                font-size:30px;           /*设置 div 标签的字体大小*/
            }
            div:hover
            {
                transform-origin:left top ;              /* CSS3 标准 */
                -webkit-transform-origin:left top ;      /*用于 Safari &
                                                         Chrome 浏览器*/
                -khtml-transform-origin:left top ;       /*用于 Safari &
                                                         Chrome 浏览器*/
                -moz-transform-origin:left top ; /*用于 Firefox 浏览器*/
                -o-transform-origin:left top ;   /*用于 Opera 浏览器 */
                -ms-transform-origin:left top ;  /* IE 9 */
                transform: scale(2);             /* CSS3 标准 */
                -webkit-transform: scale(2);     /*用于 Safari & Chrome 浏
                                                 览器 */
                -khtml-transform: scale(2);      /* 用于 Safari & Chrome 浏
                                                 览器*/
                -moz-transform: scale(2);        /*用于 Firefox 浏览器 */
                -o-transform: scale(2);          /* 用于 Opera 浏览器*/
                -ms-transform: scale(2);         /* IE 9 */
            }
        -->
    </style>
</head>
<body>
    <div>
        文字大小变化
    </div>
</body>
</html>
```

对上述代码进行剖析如下：

本程序只有一个 DIV 标签，在使用 transform 属性之前需要使用 transform-origin 属性来定义 transform 变换的固定位置，因为 transform-origin 属性的默认值为 center，所以 scale 变化会以 center 为参照点进行缩放。而变换后将会有一部分超出了浏览器的显示范围，所以需要将 transform-origin 属性设置为 left top。

另外，transform 属性设置为 scale(2)以达到纵向和横向都放大 2 倍的目的。还要补充的是，不同的浏览器需要对 transform-origin 属性和 transform 属性添加不同的前缀。例如，Safari 和 Chrome 浏览器以-webkit-或-khtml-为前缀，Firefox 浏览器以-moz-为前缀，Opera 浏览器以-o-为前缀，IE 9 浏览器以-ms-为前缀。

如图 3.17 所示为这段代码的运行结果，即使用 transition 属性的 scale 的效果。

图 3.17　transition 属性的 scale 的使用

3.2　文字示例一：新闻摘要

新闻是网页的重要组成部分，传统的网页主要是以发布新闻为主，人们可以在人民网、新浪网和网易新闻网获取大量的新闻。而文字是新闻的重要组成部分，所以这一节我们动手编写一个新闻摘要的页面程序，来进一步学习文字的布局和渲染。

3.2.1　新闻摘要的 DIV 布局设计

下面设计一个简单的新闻 DIV 框架，该 DIV 框架主要包括新闻的标题，发布新闻的报社名称、新闻发布的时间以及新闻的内容摘要等等。

```
    <div id="news">
        <div class="text">
            <a href="#">
                <span><b>新闻标题链接 </b></span>
            </a>
            新闻报社和发布时间
        </div>
        <div>
            新闻内容摘要
        </div>
    </div>
```

本 DIV 框架由 3 个 DIV 标签组成，第一个 DIV 是父节点 DIV，这样设计能够将它和其他相同层次的 DIV 相区别。它的内部还包含两个 DIV 标签，每个 DIV 标签占据一行。

另外，在很多的网页里还会加入新闻的图片快照，这样可以使网页新闻更有吸引力，对新闻内容起到一定的补充作用。达到这些目的只需要加入类似的语句：。

3.2.2　新闻简要的 CSS 样式设计

下面将添加 CSS 样式。只需要添加简单的 CSS 样式就可以呈现出较好的新闻摘要的效果。以下是本程序的完整代码。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
    <title>
        新闻简要
    </title>
    <style type="text/css">
        <!--
            .news
            {
                width:540px;                    /*设置宽度*/
```

```
            }
            .news .text
            {
                font-size:75%;                  /*设置字体的大小*/
                line-height:1.5em;              /*设置行间距离*/
            }
            .news .text span
            {
                font-size:1.167em;              /*设置字体高度*/
            }
            .news a
            {
                color: #0000cc;                 /*设置颜色*/
            }
        -->
    </style>
</head>
<body>
    <div class="news">
        <div class="text">
            <a href="#">
                <span><b>新闻标题链接  </b></span>
            </a>
            <font color=#666666> 新华网 2012-8-9 16:39  </font>
        </div>
        <div>
            <font size="2">【<font color="#CC0000">新闻摘要</font>】新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 </font>
            </div>
    </div>
    <br/>
    <div class="news">
        <div class="text">
            <a href="#">
                <span><b>新闻标题链接  </b></span>
            </a>
            <font color=#666666> 新华网 2012-8-9 16:39  </font>
        </div>
        <div>
            <font size="2">【<font color="#CC0000">新闻摘要</font>】新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 </font>
        </div>
    </div>
    <br/>
    <div class="news">
        <div class="text">
            <a href="#">
                <span><b>新闻标题链接  </b></span>
            </a>
            <font color=#666666> 新华网 2012-8-9 16:39  </font>
        </div>
        <div>
            <font size="2">【<font color="#CC0000">新闻摘要</font>】新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 新闻内容摘要 </font>
        </div>
```

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

对上述代码进行剖析如下：

将第一级的 DIV 的 class 属性设置为 news，并设置其 CSS 的 width 属性为 540px。设置新闻标题的 DIV 标签的 class 属性为 text，并设置其 CSS 的 font-size 和 line-height 属性来控制字体大小以及行间的距离。此处使用了新的单位，即 em。它表示的意义是字体高，针对任意的浏览器，默认的字体高都是 16px，即可以表示为 1em=16px。同理可以推算，10px=0.625em。

不过为了简化 font-size 的计算，一般在标签中会声明 font-size 的百分比。例如，该程序中的 font-size=75%，此时 1em 就等于 16px×75%=12px，所以本程序的 1.5em 就等于 12px×1.5=18px，1.167em 就等于 12px×1.167=14px。

新闻内容的样式还使用了 font 标签来设置，font 标签的使用较为方便，font 标签支持使用 size 和 color 属性来设置文本的大小和颜色。

如图 3.18 所示为这段代码的运行结果，即模拟几个新闻摘要。

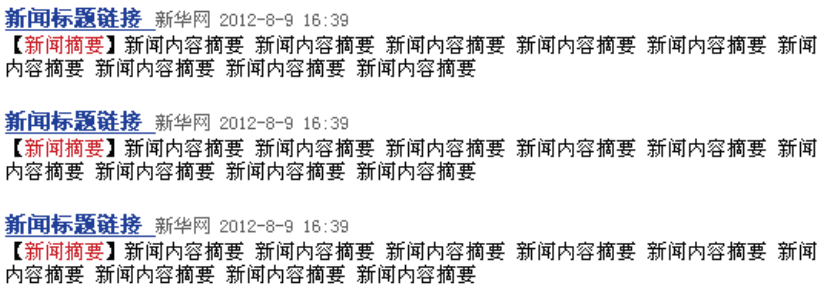


图 3.18　新闻摘要

3.3　文字示例二：个人简介页面布局

当面临毕业时，几乎每个求职学生都免不了要设计一份个人简历，一个好的个人简历可以吸引招聘者眼球并为你赢得一份满意的工作。一方面你可以设计一个 pdf 格式或 doc 格式的简历，另一方面甚至可以设计一个 html 的网页简历。而现在很多的招聘网站，如大街网、51job 等网站里，当你注册并填写个人信息后，系统都会生成相应的网页简历。所以学习编写个人简历的网页也是有它的意义的。在这一节我们动手编写一个个人简历的页面来学习文字的布局。

3.3.1　整体 DIV 布局设计

下面定义一个可扩展的 DIV 框架，以下是本程序的 HTML 部分。

```
    <div id="profile">
```

```
        <h4>个人简介</h4>
        <div class="part">
            <div id="photo">
                <img src="#" />
            </div>
            <div class="title">个人简介</div>
            <div class="content">姓名：张山　　性别：男　　年龄：25
                                 籍贯：陕西西安　　民族：汉族
                                 政治面貌：共青团员　　学历：硕士
                                 地址：陕西西安 XXX 区 XXX 路 XXX 号
                                 E-Mail:xxxxxxx@163.com
                                 邮编：000000　　联系电话：000-0000000</div>
        </div>
        <div class="part">
            <div class="title">教育经验</div>
            <div class="content">2006.9~2010.6　陕西理工大学 本科 软件工程专业
            2010.9~2012.6　陕西理工大学　　研究生　计算机应用技术</div>
        </div>
        <div class="part">
            <div class="title">基本技能</div>
            <div class="content">熟悉 C/C++语言， 8086 汇编， ARM 汇编，Delphi；
            了解 Java，PHP，HTML，CSS，javascript，QT；熟悉 Linux 开发环境。</div>
        </div>
    </div>
```

对上述代码进行剖析如下：

本代码段定义一个 id 属性为 profile 的第一级 div 标签，该 div 包括一个 h4 标签以及若干个 class 属性为 part 的标签。可以通过增加更多的 class 属性为 part 的 div 标签来添加更多的项目。代码中的 img src 表示的是图像文件的来源，即它的值是该图像文件的绝对路径或者相对路径，路径既可以是本地地址，也可以是网址。

3.3.2　添加 CSS 样式

以下是本程序的 CSS 代码部分，所有的 div 标签的 CSS 样式都由本 CSS 代码控制。

```
        #profile
        {
            width:400px;                          /*设置宽度*/
        }
        #profile h4
        {
            text-align:center;                    /*设置文本的对齐方式*/
        }
        #profile .part
        {
            border-top:double #CCCCCC;            /*设置上边框样式和颜色*/
            padding-top:5px;                      /*设置上内边距宽度*/
            margin:10px;                          /*设置外边距*/
        }
        #profile .part .title
        {
            font-size:95%;                        /*设置字体大小*/
            font-weight:bold;                     /*设置字体粗细*/
            background:#cccccc;                   /*设置背景颜色*/
```

```
        width:8em;/*设置宽度*/
    }
    #profile .part .content
    {
        font-size:80%;                          /*设置字体大小*/
        white-space:pre-wrap;                   /*设置如何处理元素内的空白*/
    }
    #profile .part #photo
    {
        float:right;                            /*设置元素的浮动方向*/
        border:#555555 solid 1px;               /*设置边框的颜色和实线的大小*/
        width:100px;                            /*设置宽度*/
        height:110px;                           /*设置高度*/
    }
```

对上述代码进行剖析如下：

其中使用到的关键属性有 text-align、font-size、font-weight、white-space 和 float。其中 text-align:center 使得 h4 标签的内容居中； font-size:95%和 font-size:80%分别控制 class 属性为 title 和 content 的 div 标签的字体的大小；按照 3.2.2 小节的讲解，它们表示的字体高是不同的，其对应的 1em 分别是 16px×95%=15.2px 和 16px×80%=12.8px；font-weight:bold 控制 class 属性为 content 的 div 标签的字体的粗细；white-space:pre-wrap 使得该 div 标签以和 HTML 的文本一致的格式输出。另外 float:right 使得图片浮动到右边。

如图 3.19 所示为这段代码的运行结果，即个人简历。

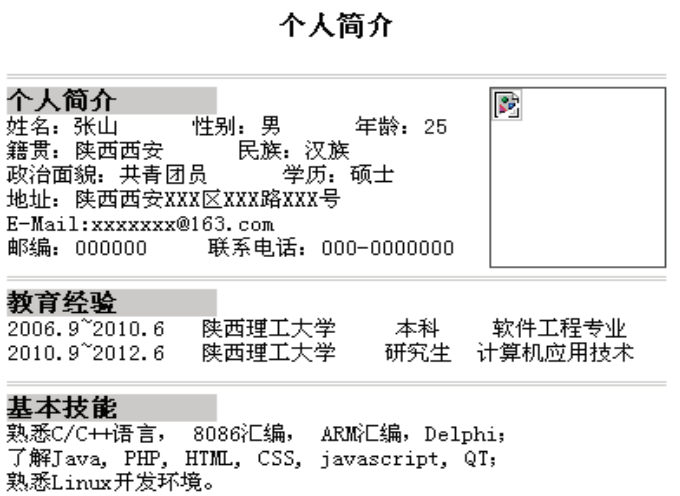


图 3.19　个人简介

3.4　文字示例三：会移动的文字

在一般的网页里面，所见到的文字通常都是静态的，也就是不会移动或者进行其他变换的。而有时我们也需要设计一些文字的动态效果来丰富页面的呈现。

在这一节将介绍两种方法来实现文字的移动效果，第一种方法是使用 JavaScript 的超

时机制来进行控制，第二种方法是使用 marquee 标签来进行控制。

3.4.1　方法 1：使用 JavaScript 控制

本节讲解如何使用 JavaScript 来控制页面中移动文字的动态样式，主要学习使用定时机制来实现该功能，以下是本程序的完整代码。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
    <title>
        文字移动 javascript
    </title>
    <script language="Javascript" type="text/javascript">
        window.setTimeout("Update();",0);       /*设置定时函数*/
        function Update()
        {
            var counter=document.getElementById("mov");
            if(counter.style.left=="4em"||counter.style.left=="")
            {
                counter.style.WebkitTransition="all 1s ease-in-out 0s";
                                /*用于 Safari & Chrome 浏览器*/
                counter.style.MozTransition="all 1s ease-in-out 0s";
                                /*用于 FireFox 浏览器*/
                counter.style.MsTransition="all 1s ease-in-out 0s";
                                /*IE 9*/
                counter.style.OTransition="all 1s ease-in-out 0s";
                                /*用于 Opera 浏览器 */
                counter.style.transition="all 1s ease-in-out 0s";
                                /* CSS 3 标准 */
                counter.style.left="-4em";
                window.setTimeout("Update();",1000); /*设置定时函数*/
            }
            else
            {
                counter.style.WebkitTransition="";
                                /* 用于 Safari & Chrome 浏览器*/
                counter.style.MozTransition=""; /*用于 FireFox 浏览器*/
                counter.style.MsTransition="";  /*用于 IE 9 浏览器*/
                counter.style.OTransition="";   /* 用于 Opera 浏览器 */
                counter.style.transition="";    /* CSS 3 标准 */
                counter.style.left="4em";
                window.setTimeout("Update();",0); /*设置定时函数*/
            }
        }
    </script>
    <style type="text/css">
        <!--
            #frame
            {
                border:#FF0000 solid 1px;       /*设置边框颜色和实线大小*/
                overflow:hidden;     /*定义溢出元素内容区的内容会如何处理*/
                width:4em;
            }
```

```
            #mov
            {
                font-size:1em;                    /*设置字体大小*/
                position:relative;                /*规定元素的定位类型*/
            }
        -->
    </style>
</head>
<body>
    <div id="frame">
        <div id="mov">
            移动文字
        </div>
    </div>
</body>
</html>
```

对上述代码进行剖析如下：

上述代码的 HTML 部分只有两个 div 标签，其中一个的 id 为 frame，另一个的 id 为 mov。因为我们需要移动 id 为 mov 的 div，所以需要设置其 left 属性，即可重新定位该 div 的位置，而这时为了隐蔽该 div 多余的部分，我们给 id 为 frame 的 div 当中的 overflow 属性设置为 hidden。为了更好地控制文字移动的效果，我们将 frame 的 width 属性设置为 4em，并将 mov 的 font-size 属性设置为 1em。和 3.2.2 小节说的一样，这里的 1em 就相当于 16px。

在 JavaScript 部分，通过“window.setTimeout("Update();",0); ”来设置超时函数，第一个参数为超时要调用的函数，第二参数表示延时时间（1000 表示 1 秒）。当超时的时候调用 Update()函数，首先设置 transition 来添加变化效果，再设置 left 属性，这样便可完成文字的移动。如果想了解更多关于 JavaScript 的知识，可以参考本书第 17 章。

如图 3.20 所示为这段代码的运行结果，即使用 JavaScript 移动文字。

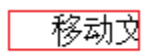


图 3.20　移动文字（使用 JavaScript）

3.4.2　方法 2：使用 marquee 标签控制

另外，HTML 还提供了 marquee 标签用于实现文字的移动。表 3.12 即为 marquee 标签的属性及其描述的列表。

表 3.12　marquee标签的属性及其描述的列表

属性	描　　述
direction	控制移动的方向。向左移动：left；向右移动：right
behavior	控制移动的方式。单方向循环：scroll；单方向不循环：slide；来回方向循环：alternate
loop	循环的次数
scrollamount	控制内容移动的速度
scrolldelay	控制内容每次移动前的延时（单位为微秒）

不仅可以通过 CSS 来控制 marquee 标签的样式，而且 HTML 还提供 marquee 标签的样式控制属性。以下是 marquee 标签的一个简单使用的例子。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
    <title>
        文字移动 marquee
    </title>
    <style type="text/css">
        <!--
            #frame
            {
                border:#FF0000 solid 1px;      /*设置边框的颜色和实线的大小*/
                width:4em;                     /*宽度*/
                font-size:1em;                 /*字体大小*/
            }
        -->
    </style>
</head>
<body>
    <marquee scrollamount=10  id="frame">      /*控制内容移动的速度*/
        移动文字
    </marquee>
</body>
</html>
```

对上述代码进行剖析如下：

在 HTML 代码部分添加 marquee 标签，并设置 scrollamount 属性。然后通过设置 marquee 标签的 CSS 的 border、width 和 font-size 来使得本程序的效果和 3.4.1 小节的结果保持一致。

如图 3.21 所示为这段代码的运行结果，即使用 marquee 标签移动文字。从得到的结果可以发现，使用 marquee 所实现的移动更加流畅。

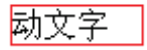


图 3.21　移动文字（使用 marquee 标签）

3.5　文字示例四：文字阴影效果

有时某些文字效果需要编程实现，这里提供了两种设计文字阴影效果方法。首先第一种是使用 CSS 定位的方法，该方法使用两个相同的文本，并通过设置第二个文本的位置，将其放在第一个文本的右下方来制造阴影效果；第二种方法是使用 CSS3 提供的阴影属性来设置阴影。

3.5.1　方法 1：使用 CSS 定位技术

我们借着实现阴影效果来初步了解 CSS 的定位技术，CSS 定位技术在很多情况下都需要使用来实现更加丰富的效果。

以下是完整的本程序的代码。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
```

```
        <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
        <title>
            文字阴影（使用 CSS 定位技术）
        </title>
        <style type="text/css">
            <!--
                #frame
                {
                    border:#FF0000 1px solid;   /*设置边框的颜色和实线大小*/
                    width:600px;                /*设置宽度*/
                    font-family:"黑体";         /*设置字体的类型*/
                    font-size:50px;             /*设置字体的大小*/
                    overflow:hidden;            /*定义溢出元素内容区的内容会
                                                如何处理*/
                }
                #DIV1
                {
                    color:#FF0000;              /*颜色设置*/
                    position:absolute;          /*规定元素的定位类型*/
                }
                #DIV2
                {
                    color:#000000;              /*设置颜色*/
                    position:relative;          /*规定元素的定位类型*/
                    left:5px;
                    top:5px;
                    z-index:-1;
                }
            -->
        </style>
    </head>
    <body>
        <div id="frame">
            <div id="DIV1">
                CSS3 文字阴影效果
            </div>
            <div id="DIV2">
                CSS3 文字阴影效果
            </div>
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

本程序由 3 个 div 标签组成，第一个作为根节点封装了其内部的所有节点，该节点的 CSS 的 overflow 属性设置为 hidden，这样可以隐藏超出本 div 范围的子节点的内容。id 为 DIV1 的 div 标签的 CSS 的 position 属性设置为 absolute，这样可以使该 div 脱离其根 div 节点 frame（frame 节点的 overflow:hidden 属性对 DIV1 不起作用），即 id 为 DIV2 的 div 标签才是 frame 的第一个子节点，所以 DIV1 和 DIV2 的位置实际上会重合到一起。为了使得 DIV2 比其原来的位置有所偏移，我们将 DIV2 的 CSS 的 position 属性设置为 relative，将 left 和 top 属性设置为 5px，这样可以使得其横、纵坐标比原位置偏移 5 个像素，另外将 DIV2 的 CSS 的 z-index 属性设置为-1，以使得 DIV2 被 DIV1 所遮盖，从而实现阴影效果。

如图 3.22 所示为这段代码的运行结果，即使用 CSS 定位技术实现文字阴影的效果。

图 3.22　文字阴影（使用 CSS 定位技术）

3.5.2　方法 2：使用 CSS 3 特有属性

以下是使用 CSS 3 的新增属性来实现文字阴影效果的代码，其实际样式效果和方法 1 的样式效果完全一致。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
        <title>
            文字阴影（使用 CSS3 特有属性）
        </title>
        <style type="text/css">
            <!--
                #DIV1
                {
                    border:#FF0000 1px solid;          /*设置边框的颜色和实线
                                                       的大小*/
                    width:600px;                       /*设置宽度*/
                    font-family:"黑体";                /*设置字体的类型*/
                    font-size:50px;                    /*设置字体的大小*/
                    color:#FF0000;                     /*颜色设置*/
                    text-shadow: 5px 5px 0px #000;     /*给文字添加阴影*/
                }
            -->
        </style>
    </head>
    <body>
        <div id="DIV1">
            CSS3 文字阴影效果
        </div>
    </body>
</html>
```

对上述代码进行剖析如下：

本代码的 HTML 部分只有一个 div 标签。只需要添加 text-shadow，并将其横、纵坐标的偏移设置为 5 像素，将模糊半径设置为 0，并将阴影样式设置为“#000”即可产生和方法 1 相同的样式效果。

如图 3.23 所示为这段代码的运行结果，即使用 CSS 3 特有属性实现文字阴影的效果。

图 3.23　文字阴影（使用 CSS 3 特有属性）