



图 13.7.2 页面的底部列出了你选择的字体（上），这块区域还有三个按钮（下）。Choose（选择）按钮对应你现在所在的状态——浏览字体，选择喜欢的字体。点击 Review（回顾）按钮可以看到更多的例子，以及集合中字体的更多信息。点击 Use 按钮，便可以在页面中使用所选的字体

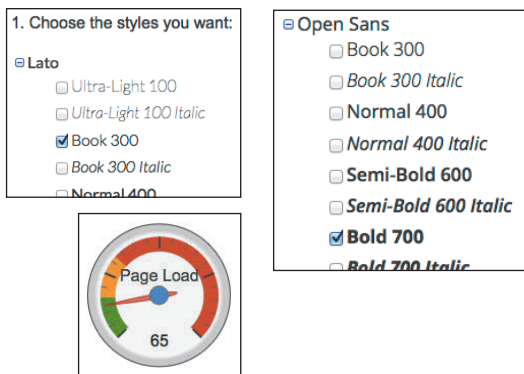


图 13.7.3 我的页面对每种字体均只需要一种粗细版本。Page Load（页面加载）仪表指示了字体文件大小对使用这些字体的页面的加载速度的影响。现在看起来还不错，不过如果添加了其他的粗细版本，页面的加载时间就会变长

(4) 仅选择内容所需的字符（如图 13.7.4 所示），这也会影响文件的大小。

(5) 将 link 元素代码（如图 13.7.5 所示）复制到你的网页的 head 中去（如图 13.7.6 所示）。

(6) 在 font-family 中使用 Google Fonts 提供的名称，便可以对文本应用字体样式。对 font-weight 使用第 (3) 步选择的一种粗细版本（如图 13.7.7 所示）。如果使用了一种斜体的 Web 字体，则在 CSS 样式规则中设置 @font-style: italic;。

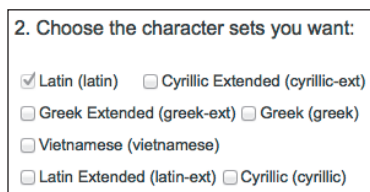


图 13.7.4 你的内容所需的字符对应的字体子集。默认为 Latin

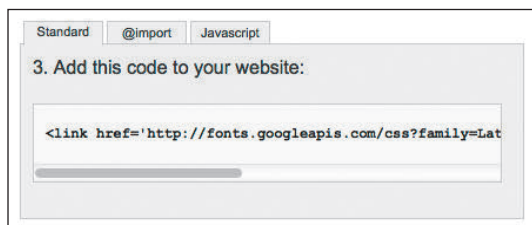


图 13.7.5 复制 Standard（标准）标签下的 link 代码



图 13.7.6 将 Google Fonts 提供的 link 代码粘贴到我自己的样式表 (style.css) 的 link 的前面。注意，Google Fonts 提供的 link 中的 href 值包含了我之前选择的字体名称和粗细版本（参见图 13.7.2 和图 13.7.3）。这样就会加载所有的 Web 字体文件以及使用它们所需的 @font-face 样式规则

(7) 保存 HTML 页面和样式表。

在浏览器中查看页面（如图 13.7.8 所示），可以看到 Web 字体已经生效了。

```
body {
  font: 100%/1.2 Georgia, 'Times New Roman',
  serif;
}

h1 {
  color: #333;
  font-family: 'Lato', sans-serif;
  font-size: 3.25em; /* 52px/16px */
  font-weight: 300;
  letter-spacing: -2px;
  line-height: .975;
  margin-bottom: .4125em;
}

h2 {
  border-bottom: 1px solid #dbdbdb;
  font-family: 'Open Sans',
  sans-serif;
  font-size: .875em; /* 15px/16px */
  font-weight: 700;
  text-transform: uppercase;
  padding-bottom: .75em;
}
```

... 本页面的更多CSS ...

4. Integrate the fonts into your CSS:

The Google Fonts API will generate the necessary browser-specific CSS to use the fonts. All you need to do is add the font name to your CSS styles. For example:

```
font-family: 'Lato', sans-serif;

font-family: 'Open Sans', sans-serif;
```

图 13.7.7 Google Fonts（下）显示了引用字体的方法。我对 h1 元素应用了粗细为 300 的 Lato 字体，对 h2 元素应用了粗细为 700 的 Open Sans 字体。这些 font-weight 数值与 link 元素中的数字（参见图 13.7.6）是对应的，而 link 元素中的数字则与我所选的粗细版本（参见图 13.7.3）对应。这里显示的其他 CSS 并不是让 Web 字体生效的必要代码

Sunny East Garden at the Getty Villa

It is hard to believe, but there are about 300 varieties of plants at the East Garden at the Getty, making the experience truly remarkable. This area is one of the most tranquil spaces at the

POPULAR POSTS

[The City Named After Queen Victoria »](#)

[Heaven on Earth? Let's Have](#)

图 13.7.8 大标题是 Lato 字体，小一些的粗体则是 Open Sans 字体

提示 如果你想对斜体文本应用字体样式，一定要确保在选择 Web 字体样式和粗细版本时（第 (3) 步）选择斜体版本，并在定义 CSS 时（第 (6) 步）指定了正确的 font-weight 值以及 font-style: italic。否则就会显示伪斜体。类似地，如果不引入字体的粗体版本，浏览器就会显示伪粗体。更多示例与讨论参见 13.6 节以及 10.4 节和 10.5 节。

本章内容

- 浏览器兼容性、渐进增强和 polyfill
- 理解厂商前缀
- 为元素创建圆角
- 为文本添加阴影
- 为其他元素添加阴影
- 应用多重背景
- 使用渐变背景
- 为元素设置不透明度
- 生成内容的效果
- 使用 sprite 拼合图像

网站制作者多年来面临的挑战之一就是，使用 CSS 建立丰富布局的选择是很有限的。在大多数情况下，要建立丰富的布局，就需要使用额外的 HTML、CSS 及大量图像。这样做的结果就是，页面变得更为复杂，可访问性降低，浏览器需要花更长的时间下载和显示页面，同时，页面变得更为脆弱，更难维护。

近年来，浏览器快速吸纳了很多新的 CSS3 属性，让上述情况有了改观。如今，仅使用 CSS 创建圆角、渐变和阴影以及调整透明度等已经变成现实。你在本章将看到如何实现这些效果，以及如何应对不支持这些特性的旧浏览器。

这样做的结果是网页可以使用更少的标记和图像，这样加载的速度也会变快。这对

所有的用户来说都有好处，尤其是对使用处理能力较低的设备（如智能手机）的用户。

14.1 浏览器兼容性、渐进增强和 polyfill

由于浏览器发展的脚步在近几年明显地加快了，因此理解这些新的 CSS 属性在什么时候会得到预期的效果就很重要了。这里给出了浏览器对本章将要介绍到的属性开始提供基本支持的时间，如图 14.1.1 所示。

					
border-radius	3.0	9.0	1.0	3.1	10.5
box-shadow	3.5	9.0	1.0	3.1	10.5
text-shadow	3.5	10.0	1.0	3.1	9.5
多重背景	3.6	9.0	1.0	3.1	10.5
渐变	3.6	10.0	2.0	4.0	11.1
透明度	1.0	9.0	1.0	3.1	9.0
before 和 :after	1.0	8.0	1.0	3.1	9.0

图 14.1.1 这个表格仅列出了桌面浏览器，不过也不要忽视移动浏览器，移动浏览器的份额每天都在快速地增长。关于移动浏览器的信息，以及桌面浏览器的更多信息，参见 Alexis Deveria 的 Can I Use 网站（www.caniuse.com），该表中的大部分信息都来自这个网站。还有一份更为简洁的支持性总结位于 <http://fmbip.com/litmus/>

这里列出的大部分浏览器版本都有些年代了，只有 IE8 仍然占据了较高的份额。截至本书写作之际，IE8 在全球的市场份额大约为 8.5%，并处于不断下滑的态势。IE8 的用户无法体验本章列举的大部分特性，不过你将看到，这没关系。

1. 渐进增强

你在前面已经多次听我提到渐进增强了，首次提到这个词是在本书的前言。简单说来，它强调创建所有用户都能访问（无论使用什么样的 Web 浏览器）的基本层面的内容和功能，同时为更强大的浏览器提供增强的体验。更简单一些，渐进增强意味着网站在不同 Web 浏览器中的外观和行为不一样是完全可以接受的，只要内容是可访问的。

Dribbble 网站（<http://dribbble.com>）就是渐进增强的一个实例，如图 14.1.2 所示。通过渐进增强，该网站使用 CSS3 为现代浏览器提供更丰富的体验。在旧的浏览器（如 Internet Explorer 8）中，该网站呈现出稍微不同的视觉体验，但功能并未削减，如图 14.1.3 所示。

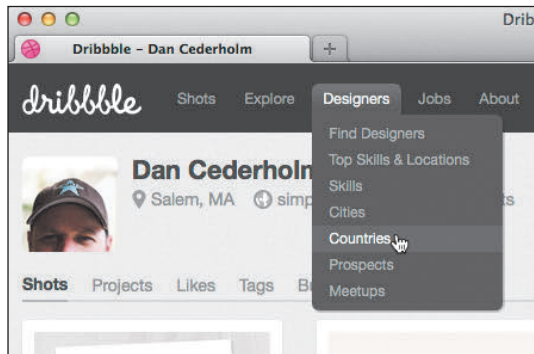


图 14.1.2 Dribbble 网站使用了一些 CSS3 属性（如 `border-radius` 和 `box-shadow`）为使用现代浏览器的用户提供更为丰富的体验，同时，它也记得为功能较弱的浏览器着想，参见图 14.1.3

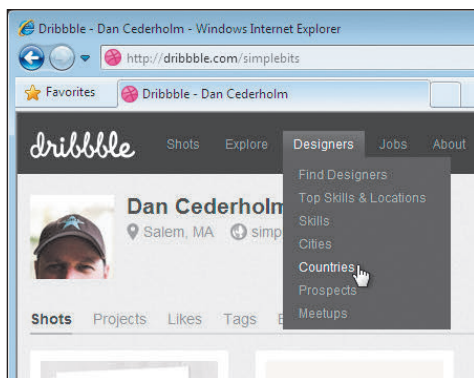


图 14.1.3 在不支持 `border-radius` 或 `box-shadow` 的旧浏览器（如 Internet Explorer 8）中查看该网站，体验就会发生改变。下拉菜单中各项的渐阴影不见了，圆角变成直角了，但功能并未损失，一切照常工作。这是渐进增强在实践中的一个方面

本章稍后为大家介绍如何对 About Me 应用渐变和阴影，如图 14.1.4 所示。IE8 显示的是没有阴影的纯色背景，但内容依旧清晰易读。



图 14.1.4 左图为现代浏览器中显示的效果，右图为 IE8 及其他旧浏览器中呈现的简化版本

2. 为不支持某些属性的浏览器使用 polyfill

如果你想弥合较弱的浏览器与较强的浏览器之间的功能差异，可以使用 polyfill（通常又称垫片，`shim`）。

polyfill 通常使用 JavaScript 实现，它可以为较弱的浏览器提供一定程度的对 HTML5 和 CSS3 的 API 和属性的支持，同时，当浏览

器本身就具有相应的能力时，会不动声色地退而使用官方的支持。需要注意的是，这样做通常会对性能产生一定的负面影响，因为较弱的 Web 浏览器（尤其是旧版本的 Internet Explorer）运行 JavaScript 的速度要慢得多。

关于 IE, Jason Johnston 的 CSS3 PIE (<http://css3pie.com>) 是一种最为常见的 polyfill^①。它为 Internet Explorer 6 ~ 9 提供了本章讨论的大部分 CSS 效果的支持。（其中，IE9 仅需要 PIE 提供对线性渐变的支持；对于其他的效果，IE9 提供原生的支持。）

提示 HTML5 Please (<http://html5please.com>) 是一个不错的资源，使用它可以找出哪些 HTML5 和 CSS3 可以放心使用，以及有哪些好的填补差距的 polyfill。

提示 Modernizr (www.modernizr.com) 是一个 JavaScript 库，它允许你探测浏览器是否支持创建优化的网站体验所需的特定的 HTML5、CSS3 及其他特性。

更多的 CSS3 效果

相比这些页面中应用的效果，CSS3 能够实现的效果要多很多，如变换、过渡、动画等类别。例如，我们可以创建滑过链接时渐变的效果：

```
a { /* 简单起见，省略了前缀 */
  color: #007c21;
  transition: color .4s ease;
}
a:hover { color: #00b3f2; }
```

要想找到学习更多内容的链接，参见 www.htmlcssvqs.com/resources/。

14.2 理解厂商前缀

CSS3 规范要达到 W3C 的推荐标准（即定稿）状态要经过数年。浏览器则通常在 W3C 开发标准的过程中就会提前实现这些特性。这样，标准在最终敲定之前就能知道哪些地方还能进一步改进。

在包含某个特性的初始阶段，浏览器通常会使用厂商前缀实现这类特性，参见图 14.2.1。这样，每个浏览器都可以引入自己的 CSS 属性支持方式，从而可以获取反馈，而且一旦标准发生改变也不会造成影响。

```
div {
  -webkit-border-radius: 10px;
  border-radius: 10px;
}
```

图 14.2.1 使用 border-radius 属性的示例。该属性需要使用 -webkit 前缀支持旧版本的 Android、iOS 和 Safari 浏览器。这些浏览器的一些新版本已经不需要使用带前缀的属性了，而只需使用无前缀的属性就行了（简单写成 border-radius: 10px;）。照例，样式规则中的最后一条声明优先级最高，这也是将无前缀版本放在最后的原因

不过，这种方式会造成一定程度的混乱，因此很多浏览器都逐渐取消了厂商前缀的使用。不过，正如你在本章看到的，还有一些地方需要使用它们，至少在依赖它们的旧浏览器依然占据一定市场份额的时候是这样。

每个主流浏览器都有其自身的前缀：-webkit-（Webkit/Safari/旧版本的 Chrome）、-moz-（Firefox）、-ms-（Internet Explorer）、-o-（Opera）。应该将前缀放在 CSS 属性名的前面，参见图 14.2.1。如今，在多数情况下，我们一般只需要 -webkit 前缀。

^① polyfill 用于填补旧浏览器与新的 Web 技术之间的差异。——译者注

提示 并非所有的 CSS3 属性都需要使用为浏览器准备的前缀，如 text-shadow 和 opacity（本章稍后就会介绍）。

提示 网上有一些创建 CSS3 代码（包括厂商前缀）的工具，参见图 14.2.2。CSS3, Please! 列出了哪些浏览器支持无前缀的语法，哪些需要前缀。这对决定是否使用前缀这一问题来说是非常有用的——你可以自行决定一部分浏览器不显示 CSS3 效果是不是可接受的。

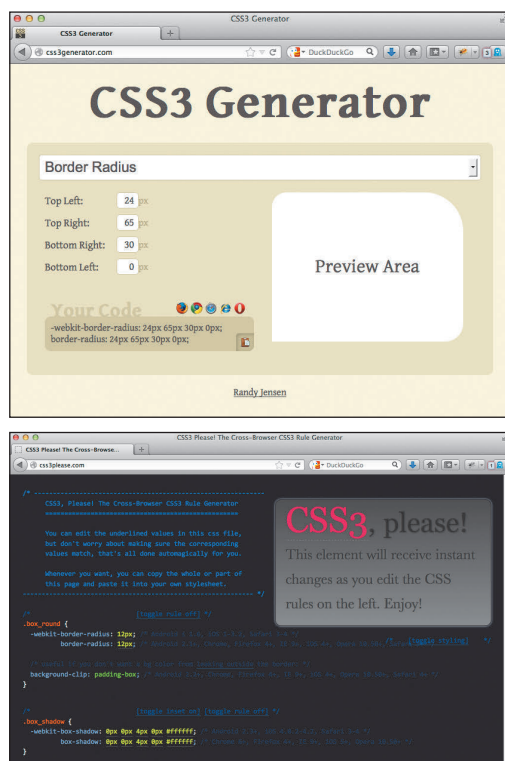


图 14.2.2 Randy Jensen 的 CSS3 Generator（www.css3generator.com），以及 Paul Irish 和 Jonathan Neal 的 CSS3, Please!（css3please.com）可以让你避免书写带前缀和不带前缀的 CSS 属性这种重复劳动。像 LESS、Sass 和 Stylus 这样的 CSS 预处理器处理这个问题会更方便。关于其他厂商前缀的信息，参见 <http://css-tricks.com/how-to-deal-with-vendor-prefixes/>

14.3 为元素创建圆角

使用 CSS3，可以在不引入额外的标记或图像的情况下，为大多数元素（包括表单元素、图像，甚至段落文本）创建圆角，如图 14.3.1 ~ 图 14.3.5 所示。

```
...
<body>
<div class="all-corners"></div>
<div class="one-corner"></div>
<div class="elliptical-corners"></div>
<div class="circle"></div>
</body>
</html>
```

图 14.3.1 这个文档包含带 class 属性的示例 div。每个 div 均用于演示 border-radius 的不同语法，包括创建相同的四个圆角，使用长形式语法创建单独的圆角，创建椭圆形圆角以及创建圆形等图形

```
div {
  background: #999;
  float: left;
  height: 150px;
  margin: 10px;
  width: 150px;
}

.all-corners {
  -webkit-border-radius: 20px;
  border-radius: 20px;
}

.one-corner {
  -webkit-border-top-left-radius:
75px;
  border-top-left-radius: 75px;
}

.elliptical-corners {
  -webkit-border-radius: 50px / 20px;
  border-radius: 50px / 20px;
}

.circle {
  -webkit-border-radius: 50%;
  border-radius: 50%;
}
```

图 14.3.2 使用 CSS 圆角的四个例子，包含了必要的厂商前缀以支持旧版 Android、Mobile Safari 和 Safari 浏览器。对于 .circle，使用 75px 与 50% 的效果是一样的，因为该元素的大小为 150 像素 × 150 像素

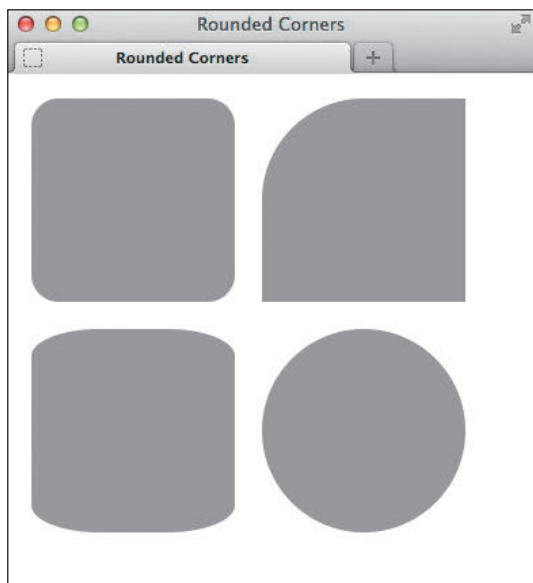


图 14.3.3 支持 border-radius 属性（包括有前缀的和无前缀的）的浏览器会以图中所示的方式将示例呈现出来。注意，不同的实现之间可能存在视觉上的差异，尤其是在旧版本的 Safari 和 Firefox 中，差异更为明显

```
.about {
    background-color: #2b2b2b;
    border-radius: 10px;
    padding: .3125em .625em .625em;
}

.about img {
    border: 5px solid #bebebe;
    border-radius: 15px;
}
```

图 14.3.4 这是一个真实的例子，对本书之前创建的我们熟悉的 About Me 模板应用圆角样式（如图 14.3.5 所示）。注意图像的样式规则中组合使用了 border 和 border-radius

同 border、margin 和 padding 属性一样，border-radius 属性也有长短两种形式的语法，参见图 14.3.6 和图 14.3.7。只有当你想在旧版本的 Android、Mobile Safari 和 Safari 浏览器中显示相应效果的时候才使用 -webkit 前缀。



图 14.3.5 外围的圆角比图像的 15 像素圆角半径更小一些。在旧浏览器中，这两个地方都显示为直角，就像第 11 章和第 12 章中显示的那样

```
div {
    background: #ff6;
    border: 5px solid #326795;
    ...
}

.example-1 {
    /* 将左上方和右下方圆角半径设为10px，右
       上方和左下方圆角半径设为20px */
    border-radius: 10px 20px;
}

.example-2 {
    /* 将左上方圆角半径设为20px，其他圆角半
       径设为0 */
    border-radius: 20px 0 0;
}

.example-3 {
    /* 将左上方圆角半径设为10px，右上方圆角
       半径设为20px，右下方圆角半径设为0，左
       下方圆角半径设为30px */
    border-radius: 10px 20px 0 30px;
}
```

图 14.3.6 更多使用简记法设置一个或多个圆角的例子，使用简记法就不必书写 border-top-left-radius 这样的属性了。简短起见，这里省略了 -webkit- 前缀

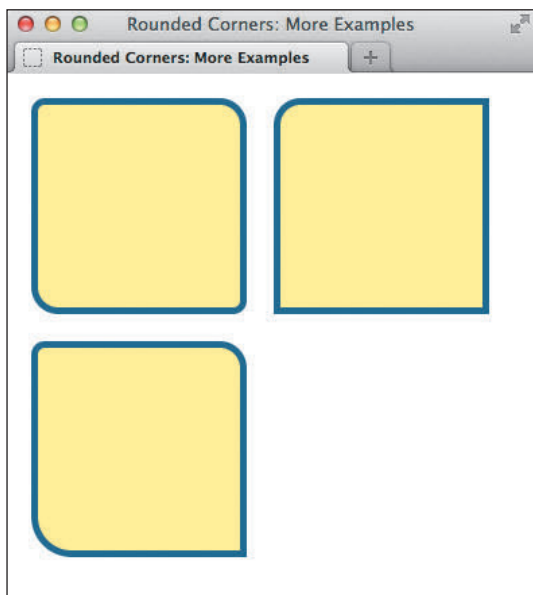


图 14.3.7 使用长语法设置的圆角样式

1. 为元素创建四个相同的圆角

(1) 这一步可选，输入 `-webkit-border-radius: r`，这里的 r 是圆角的半径大小，表示为长度（带单位）。

(2) 输入 `border-radius: r`，这里的 r 是圆角的半径大小，使用与第(1)步中相同的值，参见图 14.3.2。这是该属性的标准短形式语法。

2. 为元素创建一个圆角

(1) 这一步可选，输入 `-webkit-border-top-left-radius: r`，这里的 r 是左上方圆角的半径大小，表示为长度（带单位）。

(2) 输入 `border-top-left-radius: r`，这里的 r 使用与第(1)步中相同的值，参见图 14.3.2。这是该属性的标准长形式语法。

注意这些步骤仅演示了如何创建左上方圆角，此外，还可以单独创建其他方位的圆角，步骤如下。

- 创建右上方圆角：用 `top-right` 替换第(1)步和第(2)步中的 `top-left`。

- 创建右下方圆角：用 `bottom-right` 替换第(1)步和第(2)步中的 `top-left`。

- 创建左下方圆角：用 `bottom-left` 替换第(1)步和第(2)步中的 `top-left`。

3. 创建椭圆形圆角

(1) 这一步可选，输入 `-webkit-border-radius: x/y`，其中 x 是圆角在水平方向上的半径大小， y 是圆角在垂直方向上的半径大小，均表示为长度（带单位）。

(2) 输入 `border-radius: x/y`，其中 x 和 y 跟第(1)步中的值相等，参见图 14.3.2。

4. 使用 border-radius 创建圆形

(1) 输入 `-webkit-border-radius: r`，这里的 r 是元素的半径大小（带长度单位）。要创建圆形，可以使用短形式的语法， r 的值应该等于元素高度或宽度的一半。

(2) 输入 `border-radius: r`，这里的 r 是元素的半径大小（带长度单位），跟第(1)步中的 r 相等，参见图 14.3.3。这是标准的无前缀语法。

提示 不支持 `border-radius` 的旧的浏览器仅会以方角呈现元素。

提示 `border-radius` 仅影响施加该样式的元素的角，不会影响其子元素的角。因此，如果一个子元素有背景，该背景就有可能显示在一个或多个父元素的角的位置，从而影响圆角样式。

提示 有时元素的背景（这里讲的不是子元素的背景）会透过其圆角。为了避免这种情况，可以在元素的 `border-radius` 声明后面增加一条样式规则：`background-clip: padding-box;`。

提示 图 14.3.4、图 14.3.5 和图 14.3.7 展示了如何结合使用 border 和 border-radius。

提示 border-radius 属性不是继承的。

14.4 为文本添加阴影

最早, text-shadow 是 CSS2 规范的一部分, 接着在 CSS2.1 中被移除了, 后来在 CSS3 中又出现了。使用该元素, 可以在不使用图像表示文本的情况下, 为段落、标题等元素中的文本添加动态的阴影效果 (参见图 14.4.1 ~ 图 14.4.4)。

```
...
<body>

<p class="basic">Basic Shadow</p>
<p class="basic-negative">Basic Shadow</p>
<p class="blur">Blur Radius</p>
<p class="blur-inversed">Blur Radius</p>
<p class="multiple">Multiple Text Shadows</p>

</body>
</html>
```

图 14.4.1 演示 text-shadow 用法的段落名和类名

1. 为元素的文本添加阴影

(1) 输入 text-shadow:。

(2) 分别输入表示 x-offset (水平偏移量)、y-offset (垂直偏移量)、blur-radius (模糊半径) 和 color 的值 (前三个值带长度单位, 四个值之间不用逗号分隔), 例如 -2px 3px 7px #999 (关于允许输入的值, 参见提示)。

2. 为元素的文本添加多重阴影

(1) 输入 text-shadow:。

(2) 分别输入 x-offset、y-offset、blur-

radius 和 color 的值 (前三个带长度单位, 四个值之间不用逗号分隔)。blur-radius 的值是可选的。

(3) 输入, (逗号)。

(4) 对四种属性使用不同的值重复第 (2) 步。

```
p {
    color: #222; /* 接近黑色 */
    font-size: 4.5em;
    font-weight: bold;
}

.basic {
    text-shadow: 3px 3px #aaa;
}

.basic-negative { /* 负偏移值 */
    text-shadow: -4px -2px #ccc;
}

.blur {
    text-shadow: 2px 2px 10px grey;
}

.blur-inversed {
    color: white;
    text-shadow: 2px 2px 10px #000;
}

.multiple {
    text-shadow:
        2px 2px white,
        6px 6px rgba(50,50,50,.25);
}
```

图 14.4.2 这些类演示了几种 text-shadow 的效果。第一个、第二个和第五个都省略了模糊半径的值。multiple 类告诉我们, 可以为单个元素添加多个阴影样式, 每组属性之间用逗号分隔。这样, 通过结合使用多个阴影样式, 可以创建出特殊而有趣的效果

3. 将 text-shadow 改回默认值

输入 text-shadow: none;。

提示 text-shadow 属性不需要使用厂商前缀。

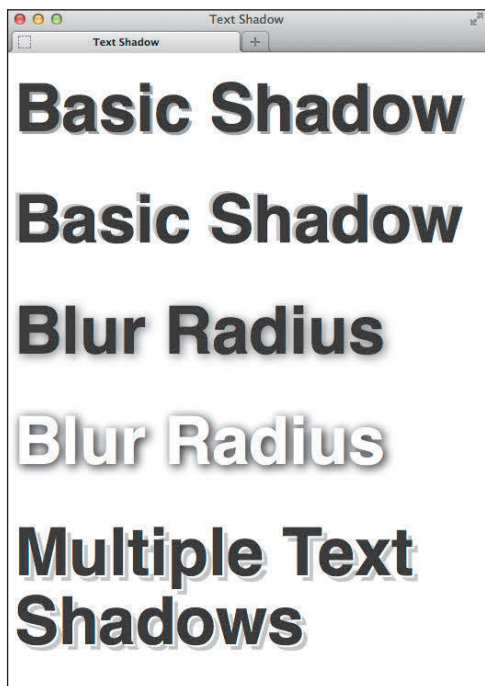


图 14.4.3 第一个段落有正的偏移量，第二个有负的偏移量。（偏移量的两个值并不需要同时为正或同时为负。）这两个段落都没有模糊半径，但第三个和第四个有。第四个有反色的效果，这是将文本颜色设为白色的结果（参见图 14.4.2）。最后一个有两个文本阴影，不过你还可以添加更多的阴影



图 14.4.4 使用了 `text-shadow` 和 `box-shadow`（将在下一节介绍）的标题和第 16 章将要介绍的表单按钮。代码位于 www.htmlcssvqs.com/8ed/16

提示 `text-shadow` 属性接受四个值：带长度单位的 `x-offset`、带长度单位的 `y-offset`、可选的带长度单位的 `blur-radius` 以及 `color` 值。如果不指定 `blur-radius`，将假定其值为 0（图 14.4.2 中有三个这样的例子）。

提示 `x-offset` 和 `y-offset` 值可以是正整数，也可以是负整数，也就是说，`1px` 和 `-1px` 都是有效的。`blur-radius` 值必须是正整数。这三个值都可以为 0。

提示 颜色可以表示为十六进制数、RGB、RGBA 或 HSLA 值（参见 7.5 节的“CSS 颜色”）。

提示 尽管 `text-shadow` 的语法与边框和背景属性的语法是类似的，但它不能像边框和背景那样单独地指定四个属性值。

提示 如果不对 `text-shadow` 的值进行设置，它就会使用初始值 `none`。

提示 `text-shadow` 属性是继承的。

14.5 为其他元素添加阴影

使用 `text-shadow` 属性可以为元素的文本添加阴影，使用 `box-shadow` 属性则可以为元素本身添加阴影，参见图 14.5.1 ~ 图 14.5.6 所示。它们的基础属性集是相同的，不过 `box-shadow` 还允许使用两个可选的属性——`inset` 关键字属性和 `spread` 属性（用于扩张或收缩阴影）。

`box-shadow` 属性与 `text-shadow` 属性的另一个区别是，如果你希望兼容旧版 Android、Mobile Safari 和 Safari 浏览器，那么 `box-shadow` 需要加上 `-webkit-` 厂商前缀。关于这方面内容的最新信息参见 <http://caniuse.com/#search=box-shadow>。开发人员可以自行决定是否省略该前缀。

`box-shadow` 属性接受六个值：带长度单位的 `x-offset` 和 `y-offset`、可选的带长度单

位的 `blur-radius`、可选的 `inset` 关键字、可选的带长度单位的 `spread` 值及 `color` 值。如果不指定 `blur-radius` 和 `spread` 的值，则设为 0。

```
...
<body>
<div class="shadow">
  <p>Shadow with Blur</p>
</div>

<div class="shadow-negative">
  <p>Shadow with Negative Offsets and
  → Blur</p>
</div>

<div class="shadow-spread">
  <p>Shadow with Blur and Spread</p>
</div>

<div class="shadow-offsets-0">
  <p>Shadow with Offsets Zero, Blur, and
  → Spread</p>
</div>

<div class="inset-shadow">
  <p>Inset Shadow</p>
</div>

<div class="multiple">
  <p>Multiple Shadows</p>
</div>
</body>
</html>
```

图 14.5.1 这个文档包含六个 `div`，它们用于演示使用 `box-shadow` 添加一个或多个阴影的效果

```
div {
  background: #fff;
  ...
}

.shadow {
  -webkit-box-shadow: 4px 4px 5px #999;
  box-shadow: 4px 4px 5px #999;
}

.shadow-negative {
  -webkit-box-shadow: -4px -4px 5px
  → #999;
  box-shadow: -4px -4px 5px #999;
}

.shadow-spread {
  -webkit-box-shadow: 4px 4px 5px 3px
  → #999;
  box-shadow: 4px 4px 5px 3px #999;
}

.shadow-offsets-0 {
  -webkit-box-shadow: 0 0 9px 3px #999;
  box-shadow: 0 0 9px 3px #999;
}

.inset-shadow {
  -webkit-box-shadow: 2px 2px 10px #666
  → inset;
  box-shadow: 2px 2px 10px #666
  inset;
}

.multiple {
  -webkit-box-shadow:
    2px 2px 10px rgba(255,0,0,.75),
    5px 5px 20px blue;

  box-shadow:
    2px 2px 10px rgba(255,0,0,.75),
    5px 5px 20px blue;
}
```

图 14.5.2 前五个类各自应用了一个彼此不同的阴影样式。最后一个类应用了两个阴影（还可以应用更多个阴影）。不理解 `box-shadow` 的浏览器会直接忽略这些 CSS 样式规则，呈现没有阴影的页面

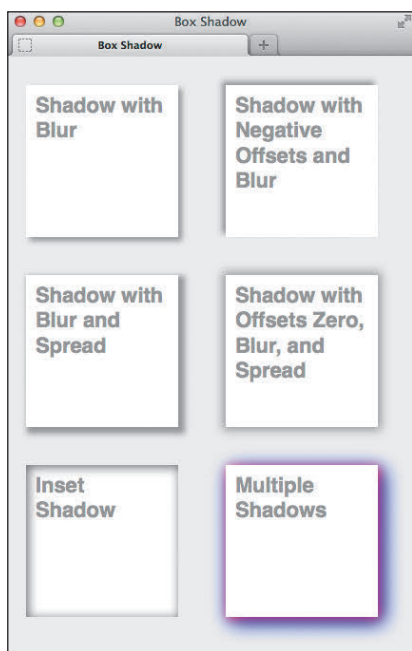


图 14.5.3 六个示例在支持 box-shadow 属性的浏览器中显示的样子

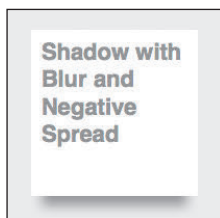


图 14.5.4 这个元素的样式为 `box-shadow: 0 20px 10px -11px #999`;。负的 `spread` 值会让阴影在元素内进行收缩。为 0 的 `x-offset` 值意味着阴影不会向左或向右偏离该元素

```
.about {
  border: 1px solid #d0d0d0;
  -webkit-box-shadow: 0 0 3px #d0d0d0;
  box-shadow: 0 0 3px #d0d0d0;
  padding: 0.313em 0.625em 0.625em;
}
```

图 14.5.5 这是一个真实的例子，即第 11 章和第 12 章的 About Me 模块。我将两个偏移量都设为 0，因此两个微小的阴影会出现在四个边上



图 14.5.6 About Me 模块有了一种新的样式。这时，它有一个 1 像素的边框，四个边上都有一个微小的阴影（深色的背景色被默认的白色取代了）

1. 为元素添加阴影

- (1) 输入 `-webkit-box-shadow:`。
- (2) 分别输入表示 `x-offset`、`y-offset`、`blur-radius`、`spread` 和 `color` 的值（前四个值均带长度单位），例如 `2px 2px 3px 5px #333`。
- (3) 输入 `box-shadow:`，再重复第 (2) 步。

2. 创建内阴影

- (1) 输入 `-webkit-box-shadow:`。
- (2) 分别输入表示 `x-offset`、`y-offset`、`blur-radius`、`spread` 和 `color` 的值（前四个值均带长度单位），例如 `2px 2px 5px #333`。
- (3) 在冒号后输入 `inset`，再输入一个空格（也可以在第 (2) 步之前输入 `inset` 和一个空格）。
- (4) 输入 `box-shadow:`，再重复第 (2) 步和第 (3) 步。

3. 为元素应用多重阴影

- (1) 输入 `-webkit-box-shadow:`。

(2) 分别输入表示 `x-offset`、`y-offset`、`blur-radius`、`spread` 和 `color` 的值（前四个值均带长度单位），例如 `2px 2px 5px #333`。如果有必要可以将 `inset` 关键字包含在内。

(3) 输入，（逗号）。

(4) 对每种属性使用不同的值重复第(2)步。

(5) 输入 `box-shadow:`，再重复第(2)步至第(4)步。

4. 将 `box-shadow` 改回默认值

(1) 输入 `-webkit-box-shadow: none;`。

(2) 输入 `box-shadow: none;`。

提示 `x-offset`、`y-offset` 和 `spread` 值可以是正整数，也可以是负整数，也就是说，`1px` 和 `-1px` 都是有效的。`blur-radius` 值必须是正整数。这三个值都可以为 0。

提示 可以使用负的 `spread` 值，让整个阴影比应用样式的元素小一些（如图 14.5.4 所示）。

提示 颜色可以表示为十六进制数、RGB、RGBA 或 HSLA 值（参见 7.5 节的“CSS 颜色”）。

提示 如果不设置属性值，则使用默认值 `none`。

提示 `inset` 关键字可以让阴影位于元素内部。

提示 `box-shadow` 属性是不继承的。

14.6 应用多重背景

为单个 HTML 元素指定多个背景是 CSS3 引入的一个特性（如图 14.6.1、图 14.6.2 和图 14.6.3 所示）。通过减少对某些元素的需

求（这类元素存在只是为了用 CSS 添加额外的图像背景），指定多重背景便可以简化 HTML 代码，并让它容易理解和维护。多重背景几乎可以应用于任何元素。

```
...
<body>
<div class="night-sky">
  <h1>In the night sky...</h1>
</div>
</body>
</html>
```

图 14.6.1 为 `class="night-sky"` 的单个 `div` 应用多重背景

```
.night-sky {
  background-color: navy; /* 备用 */
  background-image:
    url(ufo.png), url(stars.png),
    url(stars.png), url(sky.png);

  background-position:
    50% 102%, 100% -150px,
    0 -150px, 50% 100%;

  background-repeat:
    no-repeat, no-repeat,
    no-repeat, repeat-x;

  height: 300px;
  margin: 0 auto;
  padding-top: 36px;
  width: 75%;
}
```

图 14.6.2 首先，为使用旧浏览器的用户定义 `background-color`。这一步是可选的，但建议添加，从而保证内容的可访问性。然后定义加载和定位背景图像的属性，说明背景图像如何重复

1. 为单个元素应用多重背景图像

(1) 输入 `background-color: b`，这里的 `b` 是希望为元素应用的备用背景颜色（参见图 14.6.4）。

(2) 输入 `background-image: u`, 这里的 u 是绝对或相对路径图像引用的 URL 列表 (用逗号分隔)。

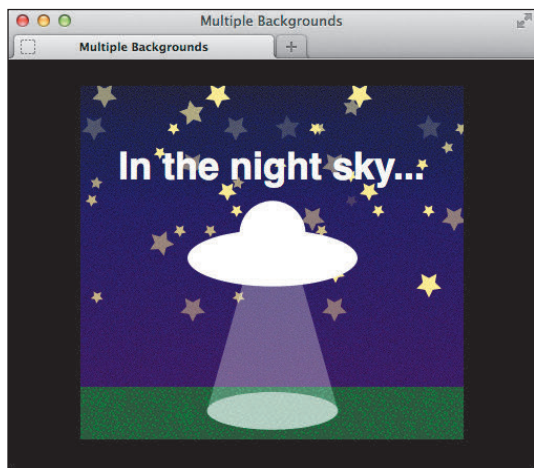


图 14.6.3 支持多重背景的浏览器显示示例的样子。其中, 图像是分层次相互重叠在一起的, 用逗号分隔的列表中的第一个图像位于顶部。另见彩插

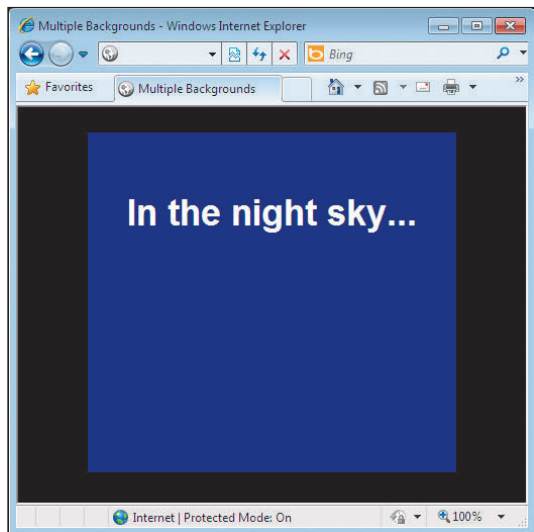


图 14.6.4 IE8 及不支持多重背景图像语法的浏览器会将 `background-color` 属性指定的背景显示为备用背景颜色

(3) 输入 `background-position: p`, 这里的 p 是成对的 `x-offset` 和 `y-offset` (可以是正的, 也可以是负的; 带长度单位或者关键字, 如 `center top`) 的集合, 用逗号分隔。对于第 (2) 步中指定的每个 URL, 都应有一组 `x-offset` 和 `y-offset`。

(4) 输入 `background-repeat: r`, 这里的 r 是 `repeat-x`、`repeat-y` 或 `no-repeat` 值, 用逗号分隔, 第 (2) 步中指定的每个 URL 对应一个值。

提示 关于前面各步中列出的属性的更多信息参见 10.10 节。

提示 对于多重背景图像, 可以使用标准的短形式语法, 即使用逗号分隔每组背景参数, 参见图 14.6.5。这种表示方法的好处是开发者既可以指定备用背景颜色, 也可以为旧浏览器指定图像, 参见图 14.6.6。

提示 指定多重背景不需要使用厂商前缀。

```
.night-sky {
    /* 备用颜色和图像 */
    background: navy url(ufo.png) no-repeat
               center bottom;

    background:
        url(ufo.png) no-repeat 50% 102%,
        url(stars.png) no-repeat 100% -150px,
        url(stars.png) no-repeat 0 -150px,
        url(sky.png) repeat-x 50% 100%;

    ...
}
```

图 14.6.5 这与图 14.6.2 的代码几乎是相同的, 只是使用了简记法, 而且为 IE8 及其他旧浏览器同时提供了背景颜色和背景图像 (如图 14.6.6 所示)。对于支持这些属性的浏览器, 显示效果与图 14.6.6 所示的相同

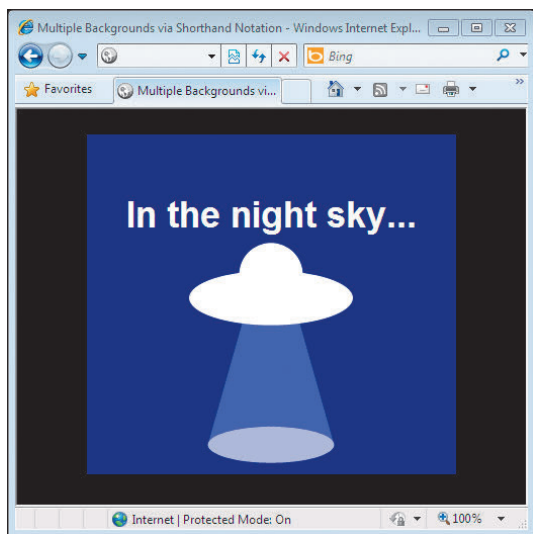


图 14.6.6 旧浏览器现在可以显示比图 14.6.4 更好的效果

14.7 使用渐变背景

渐变背景也是 CSS3 中的新特性，通过它可以在不使用图像的情况下创建从一种颜色到另一种颜色的过渡（参见图 14.7.1 至图 14.7.11）。

```
...
<body>
<div class="vertical-down"><p>default</p>
→ </div>
<div class="vertical-up"><p>to top</p></div>
<div class="horizontal-rt"><p>to right</p>
→ </div>

... 某类div的其余代码 ...
</body>
</html>
```

图 14.7.1 这是所有 CSS 渐变示例（除了图 14.7.10 和图 14.7.11）用到的 HTML。它包含了一系列 div，每个 div 都有一个用于应用渐变样式的类名。（需要说明的是，渐变可以用于大多数元素，并不仅限于 div。）这个页面的完整版本可见本书配套网站

```
.vertical-down { /* 默认 */
background: silver; /* 备用 */
background: linear-gradient(silver, black);
}

.vertical-up {
background: silver;
background: linear-gradient(to top,
→ silver, black);
}
```

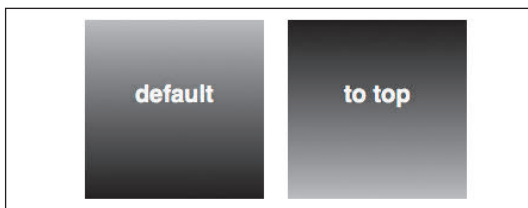


图 14.7.2 CSS 渐变是一个背景图像，因此，用在这里以及其他地方的属性既可以是 background 简记法，也可以是 background-image。应该始终包含一个为旧浏览器准备的基准 background 定义，并放置在渐变背景的定义之前。默认情况下，线性渐变是从上往下渐变的，因此在属性值中不需要指定 to bottom。如果要使用相反的方向，则使用 to top。渐变会依照指定的方向从银灰色渐变为黑色

```
.horizontal-rt {
background: silver;
background: linear-gradient(to right,
→ silver, black);
}

.horizontal-lt {
background: silver;
background: linear-gradient(to left,
→ silver, black);
}
```



图 14.7.3 指定水平渐变为由左向右渐变还是由右向左渐变

```

.diagonal-bot-rt {
  background: aqua;
  background: linear-gradient(to bottom
    → right, aqua, navy);
}

.diagonal-bot-lt {
  background: aqua;
  background: linear-gradient(to bottom
    → left, aqua, navy);
}

.diagonal-top-rt {
  background: aqua;
  background: linear-gradient(to top
    → right, aqua, navy);
}

.diagonal-top-lt {
  background: aqua;
  background: linear-gradient(to top left,
    → aqua, navy);
}

```

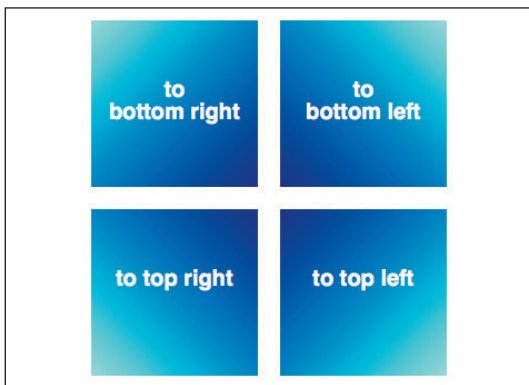


图 14.7.4 有两种指定渐变方向角度的方式。第一种方式就是这里所示的，使用关键字指定渐变方向需要旋转的角度。渐变会从跟指定角相对的角（对角线另一端的角）开始。另见彩插

使用 CSS 创建渐变有两种主要的方式：线性渐变（参见图 14.7.2 至图 14.7.5、图 14.7.10 和图 14.7.11）和径向渐变（参见图 14.7.6 至图 14.7.8），每种方式都有不同的必选参数和可选参数。除非指定一种颜色向另一种颜色过渡的位置，否则浏览器会自行

决定不同颜色之间的过渡（参见图 14.7.9）。

```

.angle-120deg {
  background: aqua;
  background: linear-gradient(120deg,
    → aqua, navy);
}

.angle-290deg {
  background: aqua;
  background: linear-gradient(290deg,
    → aqua, navy);
}

```



图 14.7.5 使用关键字（参见图 14.7.4）只能创建沿对角线方向的渐变。第二种方式是指定渐变角度的度数，如 90deg（90 度）。数值代表的是圆周上的点的位置：0 代表最顶端的点，90 代表最左边的点，180 代表最底端的点，270 代表最右边的点。你列出的值决定的是渐变结束的点的位置。因此，0deg 等价于 to top，90deg 等价于 to right，以此类推。另见彩插

CSS 渐变的语法在相关规范的修订过程中一直在变化。目前的语法是稳定的，但不幸的是仍有几个浏览器使用较旧的语法。因此，为了广泛支持现有的浏览器，还需要使用大量的（甚至可以说是繁琐的）带有厂商前缀的代码（如图 14.7.12 所示）。

了解这一点后，为了降低学习难度，一开始我们将仅演示正确的、无前缀的属性。本节的提示提供了一些额外信息，可供下载的本章代码中包含了完整的示例（包括带厂商前缀的属性）。

根据渐进增强的原则，最好为不支持背景渐变属性的浏览器提供一个备用选项。在 CSS 中，它可以位于背景渐变规则的前面，

参见图 14.7.2 ~ 图 14.7.10。

```
.radial-center { /* 默认 */
  background: red;
  background: radial-gradient(yellow, red);
}
.radial-top {
  background: red;
  background: radial-gradient(at top,
    → yellow, red);
}
```



图 14.7.6 包含额外的可选参数的径向渐变。最简单的例子是默认的风格规则，它使用的参数与线性渐变是一样的。在这个例子中，渐变的原点是元素的中心。可以使用 `at top` 这样的关键字指定中心的位置。跟往常一样，可以在径向渐变声明之前为较旧的浏览器提供一个背景的定义。另见彩插

```
.radial-size-1 {
  background: red;
  background: radial-gradient(100px 50px,
    → yellow, red);
}
.radial-size-2 {
  background: red;
  background: radial-gradient(70% 90% at
    → bottom left, yellow, red);
}
```



图 14.7.7 还可以控制渐变的尺寸。这里展示了几个例子。另见彩插

```
.radial-various-1 {
  background: red;
  background: radial-gradient(closest-side
    → at 70px 60px, yellow, lime, red);
}
.radial-various-2 {
  background: red;
  background: radial-gradient(30px 30px
    → at 65% 70%, yellow, lime, red);
}
```



图 14.7.8 这些例子结合了几个参数。它们都有三个颜色。位于 `at` 后面的值指定的是渐变的中心坐标。在第二个例子中，`30px 30px` 标记了渐变的尺寸。在第一个例子中，`closest-side` 决定了渐变的尺寸，它告诉浏览器从中心（指定为 `at 70px 60px`）向包含该渐变的区域的最近的一边伸展。另见彩插

```
.color-stops-1 {
  background: green;
  background: linear-gradient(yellow 10%,
    → green);
}
.color-stops-2 {
  background: green;
  background: linear-gradient(to top right,
    → yellow, green 70%, blue);
}
```

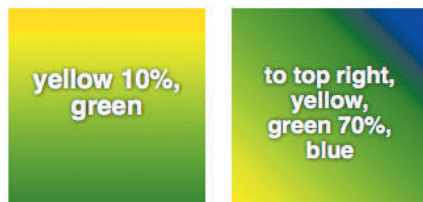


图 14.7.9 可以使用百分数指定一个以上的颜色停止位置。另见彩插

1. 创建备用背景颜色

输入 `background: color` 或者 `background-color: color`, 这里的 `color` 可以是十六进制数、RGB 值以及其他任何支持的颜色名称, 另外也可以使用图像。最好不要将 RGBA、HSL 或 HSLA 值作为备用背景颜色, 因为 IE8 及以前的版本不支持。

2. 定义线性渐变

(1) 输入 `background: linear-gradient(`。

(2) 如果你希望渐变的方向是从上往下 (默认方向), 则跳过这一步。

输入方向后面加一个逗号, 这里的方向指的是 `to top`、`to right`、`to left`、`to bottom right`、`to bottom left`、`to top right` 或 `to top left` 这样的值。

或者输入方向后面加一个逗号, 这里的方向指的是角度值 (如 `45deg`、`107deg`、`180deg` 或 `310deg`)。

(3) 根据后面讲到的“指定颜色”或“指定颜色和颜色的停止位置”, 定义渐变的颜色。

(4) 输入 `)`; 完成渐变。

3. 定义径向渐变

(1) 输入 `background: radial-gradient(`。

(2) 指定渐变的形状。如果你希望渐变的形状根据第 (3) 步中指定的尺寸自行确定, 则跳过这一步。否则, 输入 `circle` 或 `ellipse`。(注意这一声明在某些情况下会被忽略。)

(3) 指定渐变的尺寸。如果你希望渐变的尺寸为自动指定的值 (默认值为 `farthest-corner`, 最远的角), 则跳过这一步。

输入 `size`, 这里的 `size` 可以是同时代表渐变宽度和高度的一个长度值 (如 `200px` 或 `7em`), 也可以是代表宽度和高度的一对值 (如 `390px 175px` 或 `60% 85%`)。注意, 如果只使用一个值, 则这个值不能是百分数。

或者输入 `size`, 这里的 `size` 是 `closest-side`、`farthest-side`、`closest-corner` 或 `farthest-corner`。这些关键字代表相对于渐变的中心, 渐变可以伸展到多大的空间。边界决定了渐变的尺寸。

(4) 指定渐变的位置。如果你希望渐变从元素的中心开始 (默认值), 则跳过这一步。

输入 `pos`, 这里的 `pos` 是 `at top`、`at right`、`at left`、`at bottom right`、`at bottom left`、`at top right` 或 `at top left` 等表示渐变中心位置的值。

或者输入 `pos`, 这里的 `pos` 是表示渐变中心位置的一对坐标, 并以 `at` 开头。例如 `at 200px 43px`、`at 33% 70%`、`at 50% -10px` 等。(仅指定一对。)

(5) 如果指定了第 (2) 步至第 (4) 步中的任何一步, 输入一个逗号。需要说明的是, 如果指定了这些步骤中的多个值, 这些值之间不需要以逗号分隔。

(6) 根据后面讲到的“指定颜色”或“指定颜色和颜色的停止位置”, 定义渐变的颜色。

(7) 输入 `)`; 完成渐变。

4. 指定颜色

输入至少两种颜色, 每个颜色之间用逗号分隔。指定的第一个颜色出现在渐变的开始位置, 最后一个颜色出现在渐变的结束位置。对于径向渐变, 它们分别为最里边的颜色和最外边的颜色。

颜色值可以是十六进制数、RGB、RGBA、HSL 或 HSLA 值的组合。除非指定一种颜色向另一种颜色过渡的位置, 否则浏览器会自行决定不同颜色之间的过渡。

5. 指定颜色和颜色的停止位置

根据“指定颜色”中的说明进行操作。如果需要, 对每个颜色值指定一个百分

数以控制颜色出现在渐变中的位置（参见图 14.7.9）。这个值可以是负数，也可以是大于 100% 的数。

```
.about {
  /* 备用 */
  background-color: #ededed;

  background-image:
    → linear-gradient(#fff, #ededed);
  border: 1px solid #d0d0d0;
  -webkit-box-shadow: 0 0 3px #d0d0d0;
  box-shadow: 0 0 3px #d0d0d0;
  padding: 0.313em 0.625em 0.625em;
}
```

图 14.7.10 我们为曾经用过的 About Me 模块添加了新样式，会是什么样子呢？可以注意到，这里用 background-image 代替了 background 简记法（结果是完全一样的）



图 14.7.11 现在它在先前添加的阴影和边框样式的基础上，又有了一个简单的线性渐变

```
.vertical-down {
  background: silver;

  /* Chrome 10-25, iOS 5+, Safari 5.1+ */
  background: -webkit-linear-gradient
    → (top, silver, black);

  /* Firefox 3.6-15 */
  background: -moz-linear-gradient
    → (top, silver, black);

  /* Opera 11.10-12.00 */
  background: -o-linear-gradient
    → (top, silver, black);

  /* 标准语法，支持浏览器：
    Chrome 26+, Firefox 16+, IE 10+,
    → Opera 12.10+
  */
  background: linear-gradient(silver,
    → black);
}
```

图 14.7.12 噢！这是让图 14.7.2 中的渐变样式规则在各种支持该样式的浏览器里都能正常工作所需要的代码。这里没有包含针对 Safari 4 ~ 5 和 Chrome 10 之前版本的声明，如果加上它们，代码就更长了。大部分这样的代码都可以通过 CSS3, Please! (www.css3please.com) 生成（同时生成关于浏览器支持情况的注释）

6. 为旧版浏览器创建渐变代码

先前我曾指出，要让渐变可以在各种支持它的浏览器上都能正常显示，哪怕是最简单的渐变，都需要一大堆代码。现在你可以看看这些代码（如图 14.7.12 所示）。没有人愿意手写这么多代码，哪怕知道怎么写。幸好有很多工具可以帮我们做这个工作（参见提示）。

好的方面是，大多数需要厂商前缀的浏览器过一段时间（不用太久）就会因为太旧而无需再为其添加额外的代码。

提示 简单起见,在大多数示例中我都使用了颜色关键字。在实践中,你可以根据自己的偏好使用十六进制数或其他的颜色值。

提示 关于对最新浏览器的支持情况,参见 <http://caniuse.com/#search=gradient>。

提示 渐变具有很强的通用性。可以为一个背景定义多个渐变,每个定义之间使用逗号分隔。通过定义多个渐变,可以创造出很多有趣的效果。不妨访问 Lea Verou 的 CSS3 Patterns Gallery (<http://lea.verou.me/css3patterns/>) 看看有哪些有趣的效果。

提示 可以使用 Microsoft 提供的 CSS 渐变背景制作器 (<http://ie.microsoft.com/testdrive/graphics/cssgradientbackgroundmaker/>) 这样的可视化工具替代繁琐的代码编写工作。这个工具还可以帮你创建全部的厂商前缀属性,让渐变获得最大程度的浏览器支持。ColorZilla 的渐变生成器 (<http://colorzilla.com/gradient-editor/>) 也是类似的,但需要注意的是通常它生成的代码要比实际需要的多得多。

提示 可以通过指定 `background-color` 或 `background-image` 为不支持渐变的浏览器提供后备方案,不过要记住的是,无论浏览器是否使用 CSS 中的图像,这些图像都会被下载。

14.8 为元素设置不透明度

使用 `opacity` 属性可以修改元素(包括图像)的透明度,如图 14.8.1 ~ 图 14.8.5 所示。

修改元素不透明度的方法

输入 `opacity: o`, 这里的 `o` 表示元素的不透明程度(两位小数,不带单位)。

```
...
<body>

<div class="box">
  
</div>

</body>
</html>
```

图 14.8.1 这个文档包含一个 `div`, 其中有一个图像

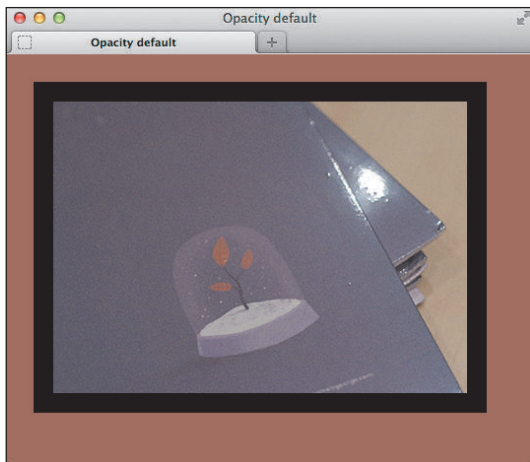


图 14.8.2 这是图 14.8.1 中 `div` 的默认样式。同其他元素一样,浏览器为其设置的默认样式为 `opacity: 1`; (不需要在样式表中指定)。该 `div` 有一个黑色的背景,由于该元素有内边距,因此背景会显示出来。`body` 的背景为褐色。另见彩插

```
.body {
  background: #a2735f; /* brown */
}

.box {
  background: #000; /* black */
  opacity: .5;
  padding: 20px;
  width: 420px;
}
```

图 14.8.3 将 `opacity` 设为小于 1 的值可以让元素及其子元素变成半透明的样子。在这个例子中,不透明度被设为 50% 或 0.5, 小数点前的 0 是可选的

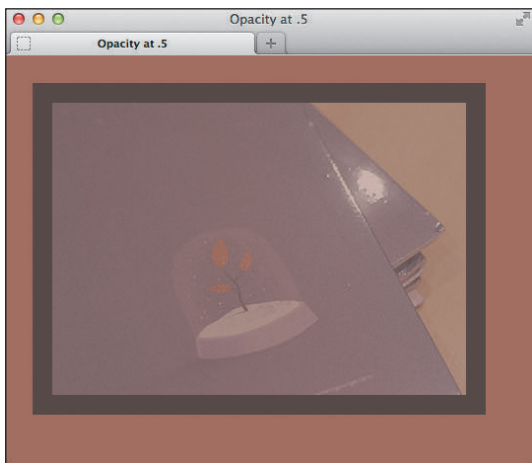


图 14.8.4 这是示例中 `div` 元素的 `opacity` 设为 0.5（即不透明度为 50%）时的样子。注意 `div` 元素的黑色背景现在呈现为深褐色，图像也变成半透明的，且带有一些褐色的色彩。如果将某元素的 `opacity` 设为小于 1 的值，包含它的周边元素就会透出来。在这个例子中，透出来的是 `body` 的褐色背景。如果将 `body` 的背景设为红色，效果会更明显（参见图 14.8.5）。另见彩插

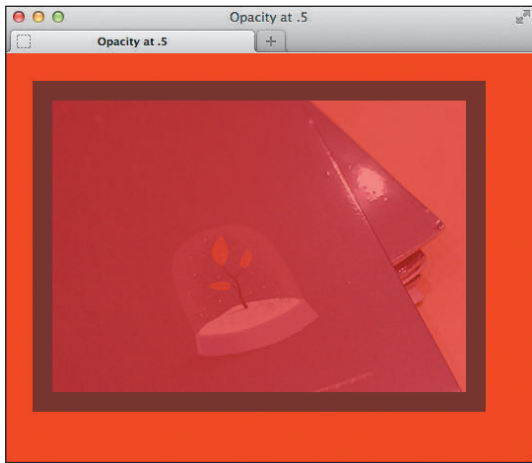


图 14.8.5 示例中 `div` 仍设为 `opacity: .5;`，但 `body` 改为 `background-color: red;`。通透的程度取决于不透明度。如果将不透明度设为 0.35，就会透出更多的红色；如果设为 0.8，则透出的红色就更少。另见彩插

```
div {
  -ms-filter: progid:DXImageTransform.
  → Microsoft.Alpha(opacity=50);
  filter: alpha(opacity=50);
  opacity: .5;
  zoom: 1;
}
```

图 14.8.6 这是针对旧版 IE 应用 IE 专有滤镜设置不透明度的代码。`zoom: 1;` 也是 IE 专有的，该语句也是用来实现效果的一部分。现代浏览器仍会使用 `opacity: .5;` 的设置。注意 IE 滤镜使用的是不同的数字。在这个例子中，使用 50 代替 0.5，表示 50% 的不透明度

提示 `opacity` 的默认值为 1，参见图 14.8.2。该属性值可以使用 0.00（完全透明）至 1.00（完全不透明）之间的两位小数。例如，`opacity: .09;`、`opacity: .2;` 或者 `opacity: .75;`。小数点前后的 0 可有可无，如 0.00 可以写作 0，1.00 可以写作 1。

提示 文本也会受不透明度的影响。如果 `div` 中有白色文本（只是打个比方），图 14.8.4 中的文字会带有一定程度的棕色，图 14.8.5 中的文字会带有一定程度的红色。

提示 通过使用 `opacity` 属性和 `:hover` 伪属性，可以产生一些有趣且实用的效果。例如，`img { opacity: .75; }` 默认情况下可以将图像设置为 75% 的不透明度，`img:hover { opacity: 1; }` 可导致用户鼠标停留在元素上时元素变为不透明。在将缩略图链接到全尺寸版本时经常看到这种效果。对于访问者来说，悬浮可增强图像的动感。

提示 opacity 属性与使用 RGBA 或 HSLA 设置的 alpha 透明背景色是两个容易混淆的概念。如图 14.8.4 和图 14.8.5 所示, opacity 影响的是整个元素(包括其内容), 而 background-color: rgba(128,0,64,.6); 这样的设置仅影响背景的透明度。参见 7.5 节“CSS 颜色”中关于 RGBA 的示例。

提示 Internet Explorer 9 之前的版本并不原生支持 opacity 属性, 但对于它们, 可以使用专有的滤镜效应, 参见图 14.9.6。确保不要在页面中过多使用这类滤镜, 因为它们会反过来影响浏览器的性能。

提示 无论显示成什么样子, opacity 属性并不是继承的。opacity 的值小于 1 的元素的子元素也会受到影响, 但这些子元素的 opacity 值仍为 1。

14.9 生成内容的效果

使用 :before 和 :after 伪元素可以很方便地为页面添加一些令人难以置信的设计效果。它们可以与 content 属性结合使用, 从而创建所谓的生成内容(generated content)。生成内容指的是通过 CSS 创建的内容, 而不是由 HTML 生成的。

不要误解, 生成内容并不是为页面添加段落或标题。这些内容还是应该交给 HTML。使用生成内容可以添加符号(如图 14.9.1 至图 14.9.4 所示), 创建用于添加样式的空内容元素(如图 14.9.5 至图 14.9.8 所示), 还可以做很多其他的事情(参见提示)。

我本可以在第一个例子的 HTML(参见图 14.9.1)中包含箭头, 但这些箭头属于样式, 以后有可能发生改变。如果需要更改样式,

只需要修改 .more 类(参见图 14.9.2)即可, 而不需要改动可能数以百计的 HTML 页面。

```
...
<p>This area is one of the most tranquil
→ spaces at the Villa. As I wandered around,
→ enjoying shade provided by sycamore and
→ laurel trees and serenaded by splashing
→ water from two sculptural fountains,
→ I couldn't help but think &hellip;
→ <a href="victoria.html" class="more">
→ Read More</a></p>
...
```

图 14.9.1 一个简单的段落, 其中有一个带类的链接。注意 Read More 的后面没有两个箭头



图 14.9.2 使用基本 CSS 文本样式的段落。链接后面没有箭头

```
...
.more:after {
    content: " »";
}
```

图 14.9.3 发挥魔力的地方

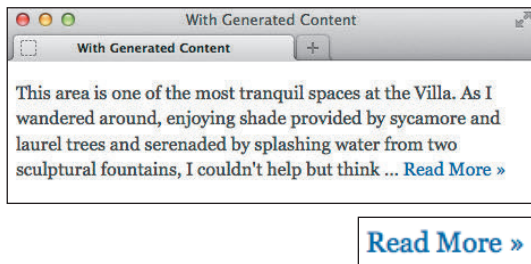


图 14.9.4 现在, 带有 class="more" 的元素会在其后显示一个双箭头

```

...
<div class="travels">
  <h2>My Travels</h2>
  

  <ul class="cities clearfix">
    <li>Victoria</li>
    <li>Los Angeles</li>
    <li>Mexico City</li>
    <li>Buenos Aires</li>
    <li>Paris</li>
    <li>Kampala</li>
    <li>Lagos</li>
    <li>Cairo</li>
    <li>Beijing</li>
  </ul>
</div>
...

```

图 14.9.5 这个城市列表会显示为底部带有箭头标记的方形气泡

第二个例子使用 `content: " "`; 创建了一个空格, 并使用 CSS 边框创建了一个三角形样式, 如图 14.9.6 所示 (参见提示)。可以对生成内容添加背景颜色、背景图像 (包括渐变)、宽度、高度、圆角等样式。

提示 完整的示例代码位于 www.htmlcssvqs.com/8ed/14/gencon。

提示 可以对单个元素同时应用 `:before` 和 `:after`, 从而建立两个可供施加样式的额外容器。例如, 可以应用一种不同于 14.6 节所介绍的多重背景效果。Chris Coyier 演示了使用 `:before` 和 `:after` 的更多神奇用法 (<http://css-tricks.com/pseudo-element-roundup/>)。

提示 Joel Glover 深入讨论了如何使用边框样式创建三角形, 参见 <http://appendto.com/blog/2013/03/pure-css-triangles-explained/>。

```

.travels {
  /* 让位于其中的.cities可以使用绝对定位 */
  position: relative;
}

.map:focus + .cities,
.map:hover + .cities {
  left: 50%; /* 显示 */
}

/* 气泡 */
.cities {
  background: #2B2B2B;
  border-radius: 5px 5px;
  left: -999em; /* 隐藏 */
  margin-left: -111px;
  padding: .5em 0 .9375em .9375em;
  position: absolute;
  top: -75px;
  width: 222px;
}

/* 气泡下的三角形 */
.cities:after {
  border: solid transparent;
  border-top-color: #2b2b2b;
  border-width: 15px;
  content: " "; /* 空格 */
  height: 0;
  left: 50%;
  margin-left: -15px;
  position: absolute;
  top: 99.9%;
  width: 0;
}

...

```

图 14.9.6 这些代码会产生一些效果。最后一条规则使用 `:after` 伪元素创建一个三角形并定位于城市列表的下方。第三条规则让列表在默认情况下位于屏幕之外。第二条规则让列表在访问者与地图交互的时候显示出来 (如图 14.9.7 和图 14.9.8 所示)。它是绝对定位的, 不过是相对于在 HTML 中包含它的 `.travels` div 的 (参见图 14.9.5)

提示 Simon Højberg 的 CSS Arrow Please! (<http://cssarrowplease.com/>) 是一个帮你创建 CSS 三角形代码的工具。

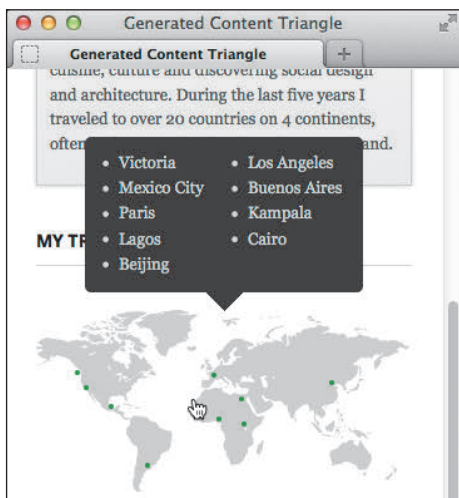


图 14.9.7 气泡底部的箭头是纯 CSS，不是图像！当鼠标移到地图的任何位置，城市气泡就会显示出来。不过，这还不是全部……

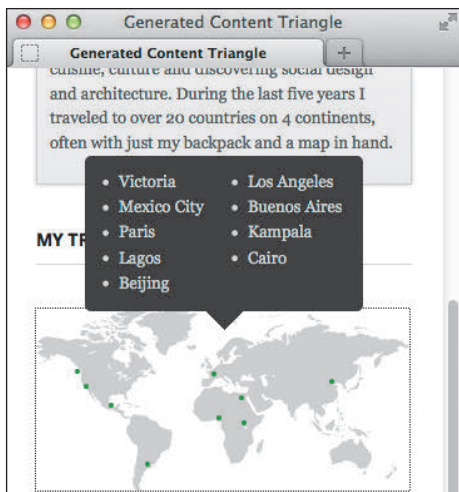


图 14.9.8 `img` 里的 `tabindex="0"` 属性（参见图 14.9.5）让 `:focus` 伪选择器（参见图 14.9.6）得以正常工作，如果访问者使用键盘的 `Tab` 键（而不是通过鼠标）将焦点放到地图上，也可以让城市气泡显示出来。这也体现了可访问性的实现（只有使用 `Tab` 键将焦点放到地图上，才会显示图中所示的虚线）

提示 `:focus` 伪类默认对链接和表单元起作用。对于这些元素，不必添加 `tabindex="0"`。

提示 第 11 章演示的 `.clearfix` 类就使用了生成内容。

14.10 使用 sprite 拼合图像

浏览器通常很快就可以将文本显示出来，但图像往往会减慢页面的加载速度。这一现象在移动设备上尤为明显。为了解决这一问题，可以将多个图像拼合成单个背景图像（sprite），再通过 CSS 控制具体显示图像的哪一部分。其中的秘密就是 `background-position` 属性。

图 14.10.1 显示了一个标准的无序列表（参见第 15 章）。我们的目标是在每个文档链接前面显示一个来自图 14.10.2 所示的 sprite 的图标。

```
...
<ul class="documents">
  <li><a href="expenses.xls" class="icon">
    → Business expenses</a></li>
  <li><a href="user-manual.pdf"
    → class="icon">User Manual</a></li>
  <li><a href="story.docx" class="icon">
    → Short story</a></li>
  <li><a href="brochure.pdf" class="icon">
    → Vacation brochure</a></li>
</ul>
...
```

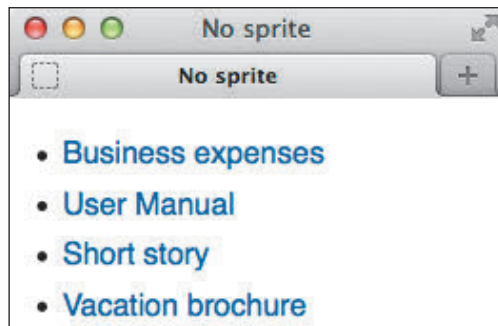


图 14.10.1 可以根据文件扩展名和 `.icon` 类找到对应文档类型的图标。在做这些工作之前，列表前面显示默认的圆点



图 14.10.2 这个 sprite 图像包含三个图标，每个都是 16 像素 × 16 像素大小，相邻两个之间有 2 像素的空隙

可以将 sprite 应用于任意数量的元素。在这个例子中，我使用 `.icon:before` 来实现所需的效果。这样，sprite 就是通过 `content: " "`；生成的空格的背景图像。将其设置为 `display: block;`，从而可以设置与图标大小匹配的高度和宽度。没有这三个属性，图像将不会显示。通过使用 `background-position`（如图 14.10.3 所示），可以将正确的图标放入该位置（如图 14.10.4 所示）。

```
...
.documents { list-style: none; }

.icon {
    display: inline-block;
    min-height: 16px;
    padding-left: 23px;
    position: relative;
    ...
}

.icon:before {
    background-image: url(sprite.png);
    content: " ";
    display: block;
    height: 16px; /* 图标高度 */
    position: absolute;
    width: 16px; /* 图标宽度 */
}

a[href$=".xls"]:before {
    background-position: -17px 0;
}

a[href$=".docx"]:before {
    background-position: -34px 0;
}
```

图 14.10.3 也可以对 HTML 中所有相关的 a 元素使用 `class="pdf"`、`class="xls"` 等（参见图 14.10.1）。不过，根据 href 值中的链接就能确定文档的类型。这样，通过使用 `$=` 就可以根据特定的扩展名找到对应的类型

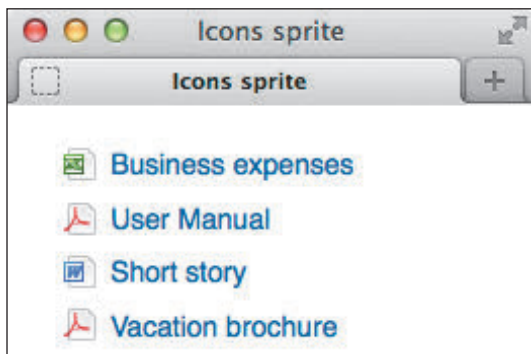


图 14.10.4 每个链接前面都显示出了正确的图标

提示 每个 sprite 都可以包含尺寸不同的图像，这些图像之间的距离也可以不同。可以将它们水平排列，也可以竖直排列。

提示 使用 sprite 并不一定要使用 `:before` 或 `:after`。可以将 sprite 背景直接用于元素本身。

提示 可以在链接或其他元素的 `:hover` 状态中修改 sprite 的 `background-position` 以实现翻转效果。

提示 Project Fondue 提供的 CSS Sprite Generator (<http://spritegen.website-performance.org>) 是一个帮助创建 sprite 和背景定位的工具（有很多类似的工具）。

提示 本节使用的图标来自 Mark James 提供的免费 Silk 图标 (www.famfamfam.com/lab/icons/silk/)。

提示 可以创建为 Retina 显示屏或其他高像素密度显示屏准备的 sprite，参见 12.5 节。

本章内容

- ❑ 创建有序列表和无序列表
- ❑ 选择标记
- ❑ 使用定制的标记
- ❑ 选择列表的起始编号
- ❑ 控制标记的位置
- ❑ 同时设置所有的列表样式属性
- ❑ 设置嵌套列表的样式
- ❑ 创建描述列表

HTML 包含专门用于创建项目列表的元素。你可以创建普通列表、编号列表、符号列表以及描述列表，可以在一个列表中嵌套另外一个或多个列表。

所有的列表都是由父元素和子元素构成的。父元素用于指定要创建的列表的类型，子元素用于指定要创建的列表项目类型。下面列出了三种列表类型以及组成它们的元素。

- ❑ 有序列表，ol 为父元素，li 为列表项。
- ❑ 无序列表，ul 为父元素，li 为列表项。
- ❑ 描述列表，dl 为父元素，dt 和 dd 分别代表 dl 中的术语和描述。描述列表在 HTML5 之前称为定义列表。

在这些类型里面，无序列表是万维网上最为常见的列表类型，它也是对大多数类型的导航进行标记的事实标准（本书有几个这样的例子）。不过，上述三种类型都各有用武之地，本章将对此进行讲解。

15.1 创建有序列表和无序列表

如果列表项的顺序对于列表来说非常关键，那么这种情况有序列表就是恰当的选择。有序列表适于提供完成某一任务的分步说明（如图 15.1.1 和图 15.1.2 所示），或用于创建大型文档的大纲。总之，它适用于任何强调顺序的项目列表。

```
...  
<body>  
  
<h1>Changing a light bulb</h1>  
  
<ol>  
  <li>Make sure you have unplugged the  
    → lamp from the wall socket.</li>  
  <li>Unscrew the old bulb.</li>  
  <li>Get the new bulb out of the  
    → package.</li>  
  <li>Check the wattage to make sure  
    → it's correct.</li>  
  <li>Screw in the new bulb.</li>  
  <li>Plug in the lamp and turn it  
    → on!</li>  
</ol>  
  
</body>  
</html>
```

图 15.1.1 目前还没有对列表标题进行格式化的正式方法。大多数情况下，使用常规的标题或段落即可，就像下面的例子那样。按照惯例（并非必须），可以对列表项目进行缩进，表明它们是嵌套在 ol 里面的（对于 ul 也是这样的）。不过，这些缩进在页面中不会显示出来，页面中的显示是由应用到列表上的 CSS 控制的



图 15.1.2 这个列表使用默认的阿拉伯数字创建带编号的有序列表。可以使用 CSS 对此进行修改。有序列表和无序列表在显示时默认都会进行缩进，无论它们在 HTML 中是否有缩进（参见图 15.1.1）

无序列表恰好相反，且应用更为普遍，如果列表项的顺序不太重要就要使用无序列表，参见图 15.1.3 ~ 图 15.1.7。

两种列表类型都适于标记特定类型的导航，参见第二条提示。

```
...
<body>

<h1>Product Features</h1>
<ul>
  <li>One-click page layout.</li>
  <li>Spell-checker for 327
    languages.
    → </li>
  <li>Image retouching tool.</li>
  <li>Unlimited undos and redos.</li>
</ul>

</body>
</html>
```

图 15.1.3 无序列表中的项目与有序列表中的是相同的，只有 ul 元素是不同的

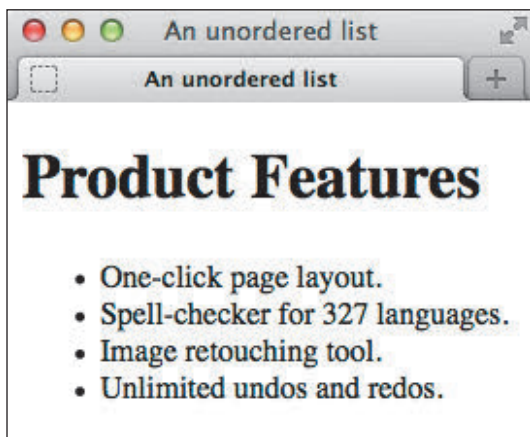


图 15.1.4 在默认情况下，无序列表前面显示实心的圆点。可以通过 CSS 对此进行修改

```
...
<h1 class="hdg">Product Features</h1>
<ul>
  <li>One-click page layout.</li>
  ...
</ul>
...
```

图 15.1.5 正如在图 15.1.4 中所看到的，元素之间的默认间距并不理想。接下来将演示如何对该样式进行调整。这里的 HTML 与图 15.1.3 中的是相同的，只是为 h1 添加了一个类（这也是使用类方便的原因）。添加了图 15.1.6 中的样式之后，就可以为任何刚好位于 ul 或 ol 前面的标题添加 .hdg 类，并得到相同的效果。类似地，可以为列表创建类，从而让缩进和间距的样式规则可以用于 ol 或 ul

创建列表

(1) 输入 （有序列表）或 （无序列表）。对于有序列表，可以包含 start、type 和 reversed 这三个可选的属性。（关于 start，参见 15.4 节；关于 type，参见 15.2 节；关于 reversed，参见最后一条提示。）

(2) 输入 以开始第一个列表项目。对于有序列表，可以包含可选的 value 属性（更多细节参见 15.4 节）。

(3) 添加要包含在列表项目内的内容（如文本、链接、img 元素等）。

(4) 输入 `</li>` 以结束列表项目。

(5) 对于每个新的列表项目，重复第 (2) 步至第 (4) 步。

(6) 输入 `</ol>` 或 `</ul>`（与第 (1) 步中的开始标签对应）以结束列表。

```
body {
    font-family: sans-serif;
}

.hdg {
    font-size: 1.5em;
    margin-bottom: 0;
}

.hdg + ul,
.hdg + ol { /* 用于ul或ol */
    margin-top: .5em; /* 约8px */
}

ul { /* 减少缩进 */
    margin-left: 0; /* 针对IE7及以下版本 */
    padding-left: 18px; /* 约1.125em */
}

ul li {
    /* 列表项之间的空格 */
    margin-top: .4em; /* 约6px */
}
```

图 15.1.6 第二条和第三条规则定义了标题与列表之间的间距，最后一条规则定义了每个列表项目之间的间距。第四条规则减小了缩进，从而让列表标记更靠近左边缘。`padding-left: 18px;` 为列表标记腾出了位置。如果该属性值为 0，则文本会对齐左边缘，而列表标记就看不见了，因为标记在默认情况下是位于列表项目之外的。不一定非要使用 `em` 为单位，如果用 `em` 的话，要记住 `em` 值是相对于父元素的字体大小的

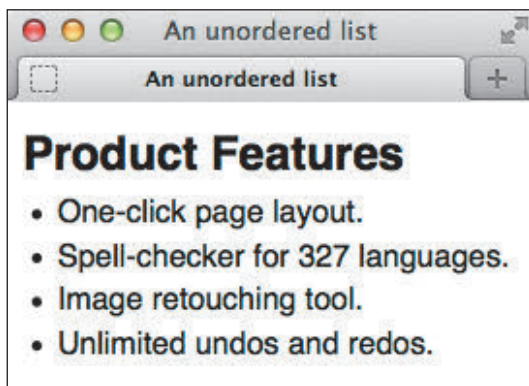


图 15.1.7 页面的呈现效果好了很多

提示 不要根据希望添加在内容旁边的标记样式决定要使用的列表类型，毕竟，这些标记的样式随时都可以通过 CSS 进行修改（甚至可以在有序列表上显示符号）。相反，应该考虑列表的含义——列表会随着其中项目顺序的改变而改变吗？如果答案是肯定的，就使用有序列表进行标记。否则，就使用无序列表。

提示 使用列表标记链接组时，大多数情况下均可以使用无序列表，如主导航链接、指向一组视频或相关报道的链接、页脚中的一组链接等。对于面包屑导航则应使用有序列表，因为这些链接有特定的次序（换句话说，顺序是有意义的）。面包屑导航通常水平地显示在主要内容区域的上方，指示当前页面在网站导航路径中的位置。分页标记是链接的水平列表，看起来像 1 | 2 | 3 | 4，通常会出现搜索结果和产品的列表，允许你在不同的结果页面之间跳转。图 15.7.5 包含了主导航和面包屑的例子。

提示 第 11 章和第 12 章的完整示例网页演示了使用和呈现列表的各种方式。其中有用于导航和其他链接组的无序列表、用于一系列链接到之前的博客条目的有序列表。第 3 章也有在导航中使用 ul 的例子。

提示 除非使用 CSS 另行指定，有序列表中的项目使用阿拉伯数字（1、2、3 等）进行编号（参见图 15.1.2）。

提示 默认情况下，无序列表的项目前面显示实心的圆点（参见图 15.1.4）。你可以选择不同的符号（参见 15.2 节），甚至可以自定义符号（参见 15.3 节）。

提示 列表在默认情况下是从左边缩进的，不过可以通过 CSS（参见图 15.1.6）取消缩进、减小缩进（参见图 15.1.7）或增加缩进。15.3 节也演示了这些操作。列表标记可能会显示在内容的外面，也可能因为超出窗口的边缘而消失（参见图 15.1.6 所示的内边距为零的样式）。

提示 默认情况下，列表有一定的左侧外边距，从而形成缩进。可以使用 CSS 取消（减小）或增大缩进，参见图 15.1.6 和图 15.1.7。15.3 节也会解释这一点。根据减少缩进的量，符号可能处于内容的外边或者消失在窗口的左边缘之外（参见图 15.1.6 中假设内边距为 0 的情况）。

提示 可以在一个列表中创建另一个列表（即嵌套列表，nesting list），甚至可以混合或者搭配使用有序列表和无序列表。不过，要确保用到所有需要的开始标签和结束标签，对每个列表进行正确地嵌套。嵌套有序列表和无序列表的例子见 15.7 节。

提示 只能将列表内容放在 li 元素里。例如，不允许将内容放置在 ol 或 ul 开始标签和第一个 li 元素之间。li 元素里面可以放置多种类型的元素，如各种短语内容元素（em、a、cite 等）。在列表项目中嵌套段落、div 这样的元素也是有效的。

提示 如果将内容方向指定为从右向左（例如在语言为希伯来语的情况下），列表就会通过右侧外边距进行缩进。要指定内容方向，可以在页面的 html 元素中设置 dir 属性，如 `<html dir="rtl" lang="he">`。在这个例子中，lang 被设为表示希伯来语的 he。还可以在 body 里面的元素上设置 dir 和 lang 以覆盖 html 元素上的设置。dir 属性的默认值为 ltr。

提示 截至本书写作之际，浏览器对布尔型的 reversed 属性的支持还不够理想：只有 Chrome 18+、Firefox 18+ 和 Safari 5.2+。该属性的作用是指示降序排列的有序列表（使用 `<ol reversed>` 或 `<ol reversed="reversed">` 均可指定该属性）。支持该属性的浏览器会自动颠倒列表的编号。

15.2 选择标记

创建列表时，无论是有序列表（如图 15.2.1 所示）还是无序列表，都可以选择出现在列表项目左侧的标记的类型。

1. 选择标记

在样式表中，输入 list-style-type: marker，这里的 marker 是以下属性值中的一种。

- ☐ disc（圆点，●）
- ☐ circle（圆圈，○）
- ☐ square（方块，■）

- decimal (数字, 1、2、3……)
- upper-alpha (大写字母, A、B、C……)
- lower-alpha (小写字母, a、b、c……)
- upper-roman (大写罗马数字, I、II、III、IV……), 如图 15.2.2 和图 15.2.3 所示
- lower-roman (小写罗马数字, i、ii、iii、iv……)

```
...
<body>

<h1 class="hdg">The Great American Novel</h1>
<ol>
  <li>Introduction</li>
  <li>Development</li>
  <li>Climax</li>
  <li>Denouement</li>
  <li>Epilogue</li>
</ol>

</body>
</html>
```

图 15.2.1 这是用做示例的有序列表。我们将对其应用大写罗马数字 (upper-roman)。注意, 为了利用图 15.1.6 中的 .hdg 规则, 我对 h1 应用了 class="hdg"

```
li {
  list-style-type: upper-roman;
}

/* 下面的代码的效果一样, 因为li元素继承list-
   style-type属性

ol {
  list-style-type: upper-roman;
}
*/
```

图 15.2.2 可以对 ol (参见注释)、ul 或者列表项本身应用 list-style-type 属性, 如下面的代码所示。虽然这里没有显示, 但我为 body 元素应用了某种字体

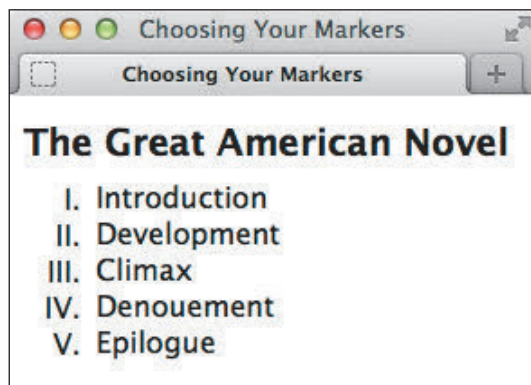


图 15.2.3 现在, 有序列表采用大写罗马数字进行编号。注意, 大多数浏览器会让编号右对齐

15

2. 显示无标记列表

在样式表规则中, 输入 list-style-type:

none。

提示 可以通过 list-style-type 对 ol 和 ul 设置任何标记样式。换句话说, 可以对 ol 使用方块标记, 也可以对 ul 使用数字标记。

提示 还有大量的其他标记类型可用, 不过浏览器对不同标记类型的支持情况不同。

提示 也可以在 HTML 中通过 type 属性为有序列表指定标记类型, 但推荐的做法是尽可能在 CSS 中定义列表样式类型。一种例外情况请参见 html5doctor.com/ol-element-attributes/。type 属性可接受的值包括 A、a、I、i 和 1 (默认值为 1)。例如, <ol type="I"> 表示使用大写罗马数字作为编号。

15.3 使用定制的标记

如果对使用圆圈、方块、圆点以及罗马数字这些标记感到厌倦, 也可以使用图像创建自己定制的标记。要使用定制的标记, 不

必修改 HTML（如图 15.3.1 所示），仅需修改 CSS（参见图 15.3.2 ~ 图 15.3.7）。

```
...
<body>

<h1 class="hdg">Product Features</h1>
<ul>
  <li>One-click page layout.</li>
  <li>Spell-checker for 327 major
    → languages.</li>
  <li>Image retouching tool.</li>
  <li>Unlimited undos and redos.</li>
</ul>

</body>
</html>
```

图 15.3.1 这个列表与任何普通无序列表都是相似的，不过，加上一些 CSS 后，它看起来就不一样了

使用定制的标记

(1) 在目标列表或列表项的样式规则中，输入 `list-style: none;` 以取消常规的标记。

(2) 在目标列表的样式规则中，设置 `margin-left` 和 / 或 `padding-left` 属性，指定列表项目缩进的大小。为了在不同的浏览器上实现相似的效果，通常需要同时设置这两个属性（参见图 15.3.2 和图 15.3.6）。注意，如果为内容设置了 `dir="rtl"`，那么就应该设置 `margin-right` 和 `padding-right` 属性。关于在列表中使用 `dir`、`lang` 及从右至左的语言，参见 15.1 节的提示。

(3) 在目标列表的 `li` 元素的样式规则中，输入 `background: url(image.ext) repeat-type horizontal vertical;`，其中 `image.ext` 是要作为定制标记的图像的路径和文件名，`repeat-type` 是 `no-repeat`、`repeat-x` 和 `repeat-y` 中的一种（通常设为 `no-repeat`），`horizontal` 和 `vertical` 值表示列表项目中背景图像的位置（如图 15.3.2 所示）。

(4) 输入 `padding-left: value;`，这里的 `value` 应不小于背景图像的宽度，以防列表项目的内容覆盖到定制标记的上面。

```
ul {
  /* 取消默认标记 */
  list-style: none;

  /* 删除列表项的缩进 */
  margin-left: 0;
  padding-left: 0;
}

li {
  /* 显示定制的标记 */
  background: url(..img/checkmark.png)
    → no-repeat 0 0;
}
```

图 15.3.2 下面会用三步法展示如何使用定制的标记，这样读者会对各种 CSS 属性如何影响布局一目了然。首先，清除默认的标记（这样就看不到符号和箭头了），如果想删除列表项目的缩进，应该将 `margin-left` 和 `padding-left` 都设为 0（参见倒数第二条提示）。应用于列表项的定制标记的 URL 看起来不一样（参见第一条提示），通常使用 PNG，但 GIF 或 JPEG 也行

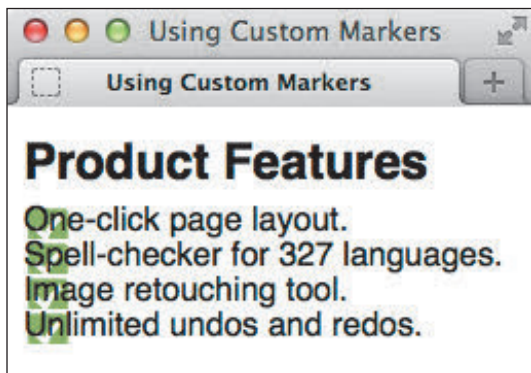


图 15.3.3 默认的圆点被核对图像取代了，列表项之前的缩进被移除了，但是文本叠加到了新的标记上，那么要不要针对列表添加左内边距，从而让标记和列表项之间有点空间呢？让我们来试试，请看图 15.3.4

```
ul {
  list-style: none;
  margin-left: 0;
  padding-left: 30px;
  outline: 2px solid red;
}

li {
  background: url(..img/checkmark.png)
    → no-repeat 0 0;
  outline: 1px solid blue;
}
```

图 15.3.4 将 margin-left 设为 0 后，就可以针对所有浏览器使用 padding-left 控制列表项目的缩进了。我对 ul 和每个 li 都添加了一个临时的边框，从而可以清晰地看到应用内边距的位置（参见图 15.3.5）

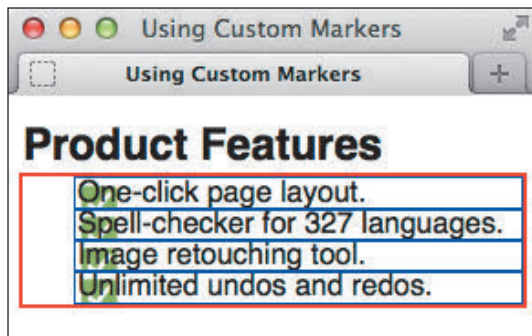


图 15.3.5 这样做让列表项目有了缩进，但文字依然与标记重叠了，这是因为我们仅为列表本身（ul）添加了左内边距，但没有对包含标记的列表项目添加左内边距。下面将修复这个问题（参见图 15.3.6）

提示 注意，相对背景 URL 是相对于样式表的位置，而不是相对于网页的位置（参见图 15.3.2），相关内容参见 10.10 节。url 和前括号之间不应有空格，包围 URL 的引号是可选的。

```
ul {
  list-style: none;
  margin-left: 0;
  padding-left: 0;
}

li {
  /* 让图像在列表项目内显示的位置稍微往
     下一些 */
  background: url(..img/checkmark.png)
    → no-repeat 0 .1em;

  /* 让行高变大，从而可以容纳整个标记 */
  line-height: 1.8;

  /* 将文本推开，为标记腾出位置 */
  padding-left: 1.75em;
}
```

图 15.3.6 将 ul 的左边距重新设为 0，列表会再次向左对齐。为每个 li 设置左内边距可以将所有标记图像显示出来，增大 line-height 则可以保证图像底部不会被截断。此外，这里还让标记向下移动了一些，从而可以与文本更好地对齐（如图 15.3.7 所示）

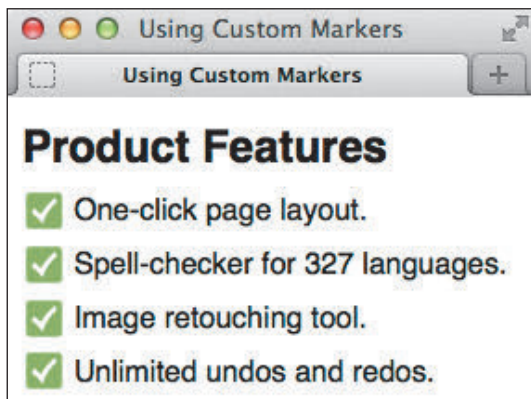


图 15.3.7 好多了

提示 默认标记在默认情况下位于列表项目的外面，而定制标记显示在列表项目的里面，因为定制标记是为列表项目本身应用的背景图像。二者的对比参见 15.1 节的图 15.1.5 ~ 图 15.1.7。那些内容还演示了如何实现本节图 15.3.7 所示的减小标题下面的间距和格式化文本的样式。

提示 如果想为列表中的某一个或几个项目应用定制的标记，可以为其添加一个类，再为这个类定义样式规则。

提示 大多数浏览器是通过 `padding-left` 为列表设置默认缩进的，但旧浏览器（如 IE8 之前版本）则是通过 `margin-left` 设置的。这就是要在使用 `padding-left` 获得统一样式结果之前需要设置 `margin-left: 0;` 的原因。现在，尽管通常不必为 IE7 及更旧的浏览器考虑了（对于大多数国家来说），但添加 `margin-left: 0;` 也没有任何坏处。

提示 另一种显示定制标记的方法是使用 `list-style-image` 属性。例如，`li { list-style-image: url(image.png); }`。不过，使用这种方法很难达到预期的效果，因为不同浏览器对它的显示效果并不一致。并且相比前面展示的背景图像方法，开发者更难控制图像标记的位置。

15.4 选择列表的起始编号

你可能希望列表的编号从默认值 1 以外的某个数字开始，如图 15.4.1 和图 15.4.2 所示。

1. 设置整个列表编号方案的初始值

在 `ol` 开始标签中输入 `start="n"`，这里的 `n` 表示列表的初始值。

2. 修改有序列表中某列表项目的编号

在目标 `li` 项目内输入 `value="n"`，这里的 `n` 代表该列表项目的值。

提示 如果使用 `start` 和 `type`，请始终使用数字。即便决定使用字母或罗马数字对列表进行编号（参见 15.2 节），也应使用数字。通过 CSS 或者 `type` 属性会显示预期的标记。

```
...
<body>

<h1 class="hdg">Changing a light bulb (with a
→ few steps missing)</h1>

<ol start="2">
  <li>Unscrew the old bulb.</li>
  <li value="5">Screw in the new bulb.
  → </li>
  <li>Plug in the lamp and turn it on!
  → </li>
</ol>

</body>
</html>
```

图 15.4.1 在这个例子中，我略去了列表中的一些步骤，不过仍希望余下的步骤保持原有的编号。因此，整个列表从 2 开始编号（使用 `start="2"`），并将第二个项目的值设为 5（使用 `value="5"`）。这两个属性都是可选的，也不一定要像这样同时使用



图 15.4.2 注意，不仅第一个和第二个项目是按我们指定的规则进行编号的，连第三个项目（Plug in the lamp and turn it on!）也受到了影响（我为页面添加了一些样式，但并没有将 CSS 写在这里。我将 `body` 字体设为 Georgia，对 `h1` 应用了 `class="hdg"`，如图 15.4.1 所示，并应用了 15.1 节中图 15.1.6 中的 `.hdg` 样式规则）

提示 `value` 属性的值会覆盖 `start` 属性的值。

提示 使用 `value` 属性对某列表项目的编号进行修改后，后续的列表项目也会相应地重新编号。

提示 如果需要在有序列表中指定两个或两个以上位置相同的项目，使用 `value` 是非常方便的。以表示公路赛前五名选手的列表为例，通常该列表会显示为 1、2、3、4、5，但如果有关并列第 2 名，则可以将第三个项目写成 `<li value="2">`，这时，列表将显示为 1、2、2、3、4。

提示 列表可以包含一个以上的带 `value` 属性的 `li`。

15.5 控制标记的位置

默认情况下，列表会从（其父元素的）左侧进行缩进。标记可以位于整个文本块的外面（这是默认情况），如图 15.5.1 所示，也可以缩在文本块内（称为缩排），如图 15.5.2 和图 15.5.3 所示。

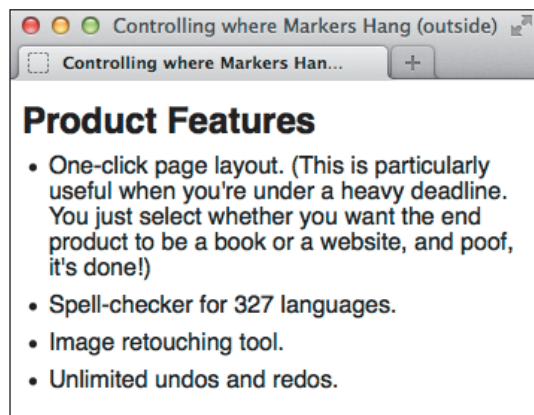


图 15.5.1 这说明了在默认情况下浏览器呈现列表项目文本及标记之间相对位置的方式。标记位于内容的外边

```
ul {
  list-style-position: inside;
}
```

图 15.5.2 将 `list-style-position` 设置为 `inside` 可以改变显示方式

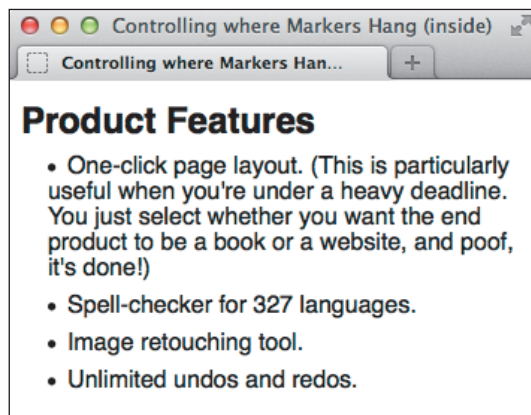


图 15.5.3 每行的标记都是从列表项目的左侧外边距开始的，而不是在整体内容之外

控制标记位置的步骤

(1) 在目标列表或列表项目的样式表规则中，输入 `list-style-position:`。

(2) 输入 `inside` 让标记缩在文本块内（参见图 15.5.2），或者输入 `outside` 让标记显示在列表项目文本的左边（这是默认的设置）。

提示 不需要指定 `list-style-position: outside;`，因为 `outside` 是默认值，除非需要覆盖其他位置设置的 `list-style-position: inside;`。

提示 `list-style-position` 可应用于 `ul`（参见图 15.5.2）、`ol` 或 `li`。结果是一样的，除非针对特殊的 `li`（如某一类的 `li`）设置不同的样式，从而在一个列表中同时拥有内部标记和外部标记。

提示 如果列表项目中的文本显得较为拥挤, 就像图 15.5.1 和图 15.5.3 中的那样, 则可以通过设置 `line-height` 增大文本行间距。例如, 可以设置 `li { line-height: 1.3; }`。不要将行高的设置与列表项目间距 (由对 `li` 设置的 `margin-top` 或 `margin-bottom` 控制) 弄混。例如, `li { margin-bottom: .5em; }`。

提示 如何减少列表项前面的缩进和标题下面的间距, 以及如何格式化文本参见图 15.1.5 至图 15.1.7。

提示 `list-style-position` 属性是继承的。

15.6 同时设置所有的列表样式属性

同 `background`、`border`、`font`、`outline` 等属性一样, CSS 也为 `list-style` 提供了简写形式, 参见图 15.6.1 和图 15.6.2。它将 `list-style-type`、`list-style-position` 和较少使用的 `list-style-image` 放到了一条属性中。

```
ul {
    list-style: circle inside;
}
```

/ 将 `ol` 作为选择器为无序列表添加样式, 有序列表和有序列表都可以使用 `li` */*

图 15.6.1 使用这条样式规则等价于将 `list-style-type` 设为 `circle`, 并将 `list-style-position` 设为 `inside`, 只是这样写更为简洁。如果想在简写属性中指定 `list-style-image`, 这个例子可以写作 `ul { list-style: url(arrow-right.png) circle inside; }`。但第 15.3 节提到过, 最好对 `li` 应用背景图像, 而不是使用 `list-style-image`

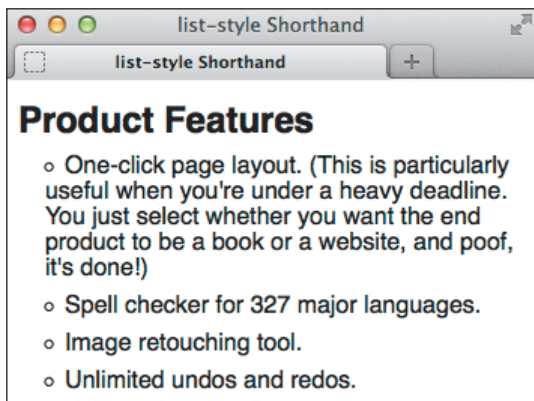


图 15.6.2 代码的实现结果与图 15.5.3 相同, 只是将标记改成了圆圈

同时设置所有的列表样式属性的步骤

- (1) 输入 `list-style:`。
- (2) 如果需要, 指定应出现在列表项目旁边的标记类型 (参见 15.2 节)。
- (3) 如果需要, 指定标记应悬挂在列表段落之外或缩在文本块内 (参见 15.5 节)。
- (4) 如果需要, 指定列表项目所用的定制图像标记 (参见 15.3 节最后一条提示)。

提示 可以指定三个 `list-style` 属性, 也可以指定其中的任意一个, 指定多个属性时顺序任意。图 15.6.1 中指定了两个。任何没有显式地指定的属性都会设为其默认值 (`list-style-type` 的默认值为 `disc`, `list-style-image` 的默认值为 `none`, `list-style-position` 的默认值为 `outside`)。

提示 关于 `list-style`, 或许最为常见的用法是使用 `list-style: none` 取消标记。

提示 跟 `list-style-type`、`list-style-position` 和 `list-style-image` 属性一样, `list-style` 属性是继承的。这也是我们可以将它应用到父元素 `ol` 或 `ul` 上的原因。

15.7 设置嵌套列表的样式

可以在一个列表中插入另一个列表，里面的列表就称为**嵌套列表**。对于有序列表和无序列表均可创建嵌套列表（混合在一起或单独嵌套）。此外，还有另外一种嵌套列表，有关示例参见 15.8 节。

如果要按有序列表的结构创建大纲（可能需要好几级的列表项目，如图 15.7.1、图 15.7.2 和图 15.7.3 所示），或者需要按无序列表的结构创建带子菜单的导航（如图 15.7.4 和图 15.7.5 所示，更多细节参见补充材料“使用嵌套列表创建下拉式导航”），就会发现嵌套列表很有用。可以通过很多方式为嵌套列表设置样式，就像示例所展示的那样。

为嵌套列表设置样式的步骤

(1) 要设置最外层列表的样式，输入 `toplevel li {style_rules}`，其中，`toplevel` 是最外层列表的类型（如 `ol`、`ul`），`style_rules` 是要应用的样式。

(2) 对于第二级列表，输入 `toplevel 2ndlevel {style_rules}`，其中，`toplevel` 对应于第 (1) 步中的 `toplevel`，`2ndlevel` 则是第二级列表的类型，而 `style_rules` 是它需要应用的样式。

(3) 对于第三级列表，输入 `toplevel 2ndlevel 3rdlevel {style_rules}`，其中，`toplevel` 和 `2ndlevel` 对应于第 (1) 步和第 (2) 步中的 `toplevel` 和 `2ndlevel`，`3rdlevel` 则是第三级列表的类型。而 `style_rules` 则是它需要应用的样式。

(4) 对于要设置样式的每个嵌套列表，继续以上述方法进行设置。

可以在每个选择器末尾包含 `li`，从而直接定位列表项目，例如，第 (3) 步中的代码可以写作 `toplevel 2ndlevel 3rdlevel li`

{`style_rules`}，图 15.7.2 中的示例代码中有这两种例子。

```
...
<body>
<h1 class="hdg">The Great American Novel</h1>
<ol>
  <li>Introduction
    <ol>
      <li>Boy's childhood</li>
      <li>Girl's childhood</li>
    </ol>
  </li>
  <li>Development
    <ol>
      <li>Boy meets Girl</li>
      <li>Boy and Girl fall in love
        → </li>
      <li>Boy and Girl have fight
        → </li>
    </ol>
  </li>
  <li>Climax
    <ol>
      <li>Boy gives Girl ultimatum
        <ol>
          <li>Girl can't believe
            → her ears</li>
          <li>Boy is indignant at
            → Girl's indignance</li>
        </ol>
      </li>
      <li>Girl tells Boy to get
        lost
        → </li>
    </ol>
  </li>
  <li>Denouement</li>
  <li>Epilogue</li>
</ol>
</body>
</html>
```

图 15.7.1 注意每个嵌套的 `ol` 都包含在其父元素的开始标签 `<li>` 和结束标签 `</li>` 之间。这里有四个嵌套列表，一个位于 `Introduction` 列表项目内，一个位于 `Development` 列表项目内，一个位于 `Climax` 列表项目内，还有一个突出显示的粗体字列表位于 `Boy gives Girl ultimatum` 列表项目内（该列表项目又位于 `Climax` 列表项目内）

```

ol {
    list-style-type: upper-roman;
}

ol ol {
    list-style-type: upper-alpha;
}

ol ol ol {
    list-style-type: decimal;
}

ol li {
    font-size: .875em;
}

li li {
    font-size: 1em; /* 防止文本不断缩小 */
}

```

图 15.7.2 可以对每一级嵌套列表单独进行格式化。如果要对列表中的文本使用 em 或百分数设置字体大小, 就需要添加 `li li {font-size: 1em;}` (或者将 `1em` 替换为 `100%`), 以防止嵌套列表不断缩小而导致难以辨认 (参见最后一条提示)

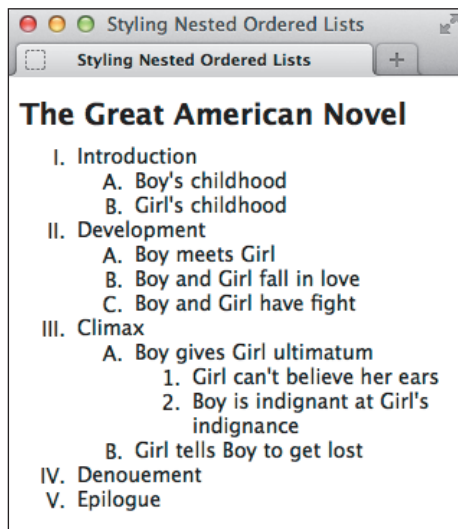


图 15.7.3 第一级列表 (`ol`) 使用大写罗马数字, 第二级列表 (`ol ol`) 使用大写字母, 第三级列表 (`ol ol ol`) 使用阿拉伯数字

```

...
<body>
<nav role="navigation">
  <ul class="nav">
    <li><a href="/">Home</a></li>
    <li><a href="/products/">Products</a>
      <ul class="subnav">
        <li><a href="/products/phones.html">Phones</a></li>
        <li><a href="/products/accessories.html">Accessories</a></li>
      </ul>
    </li>
    <li><a href="/support/">Support</a>
      <ul class="subnav">
        <li><a href="/support/forum/">Community Forum</a></li>
        <li><a href="/support/contact-us.html">Contact Us</a></li>
        <li><a href="/support/how-to-guides.html">How-to Guides</a></li>
      </ul>
    </li>
    <li><a href="/about-us/">About Us</a></li>
  </ul><!-- end .nav -->
</nav>
...
</body>
</html>

```

图 15.7.4 这是另一个嵌套列表的例子。在这个例子中, 使用无序列表构建导航菜单 (带有 `class="nav"`), 同时使用两个嵌套的无序列表构建子菜单 (每个都带有 `class="subnav"`)。通过一些 CSS, 可以让导航水平排列, 同时让子菜单在默认情况下隐藏起来, 并在访问者激活它们时显示出来, 如图 15.7.5 所示

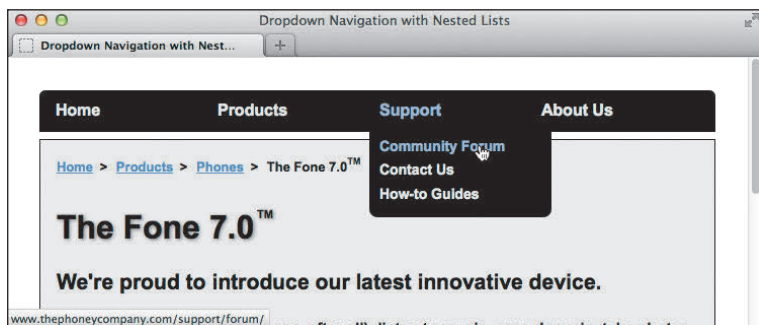


图 15.7.5 Products 和 Support 列表项目都包含嵌套在 ul 中的子菜单，不过，由于应用了一些 CSS，这两个子菜单在默认情况下并不显示出来。在这个例子中，Support 的子菜单显示出来了，这是因为鼠标停留在包含 Support 链接及子菜单嵌套列表（参见图 15.7.4）的 li 上的缘故。这个截图也显示了一个面包屑导航的例子（位于黑色横条的下方）。该面包屑导航使用有序列表进行标记，用于指示当前页面在整个网站层级中的位置。其中，每个列表项目都是一个链接，但最后一个不是链接，因为用户所在的页面正是 The Fone 7.0。完整的 CSS 见本书配套网站

提示 选择器应反映文档中嵌套列表的类型，例如，可能需要使用 ul ul ol 这样的选择器，或者在这个例子中使用 ul ul ol li 直接定位项目列表。

提示 除了上述方法以外，也可以为每个嵌套列表添加类名，再设置对应的样式。图 15.7.2 中设置的方法允许我们可以在不改变 HTML 的情况下控制样式。

提示 无论嵌套的层级是哪一级，有序列表在默认情况下总是使用阿拉伯数字（1、2、3）。可以使用 list-style-type 指定其他的编号方案（参见 15.4 节）。根据 *The Chicago Manual of Style* 一书的建议，正确的列表嵌套次序为 I（罗马数字）、A、1、a（此后交替使用 1 和 a 编号方案）。

提示 默认情况下，对于无序列表，第一级列表使用圆点符号，下一级列表使用空心圆符号，第三级及后续各级列表使用方块符号。类似地，可以使用 list-style-type 指定希望使用的符号类型（参见 15.2 节）。

提示 由于列表项目（li 元素）可以嵌套在其他列表项目内，因此在使用相对值指定字体大小时需要留心。如果使用像 li {font-size: .75em;} 这样的样式规则，那么最外层列表项目的字体大小便是其父元素的 75%，因此如果该父元素的字体大小为默认的 16 像素，那么最外层列表项目就是 12 像素大小，这还算好。但是，第一个嵌套列表项目的字体大小会是其父元素（第一个列表项目，12 像素大小）的 75%，因此只有 9 像素。每一级都会迅速让情况变得更糟糕。一种解决方案是添加 li li {font-size: 1em;}，如图 15.7.2 所示（或者将 1em 替换为 100%）。这样，嵌套列表项目就总是与顶级列表项目一样大，如图 15.7.3 所示。（由 Eric Meyer 提供，www.meyerweb.com。）

使用嵌套列表创建下拉式导航

嵌套列表的一种使用场合是创建下拉式（或飞出式）导航菜单（如图 15.7.4 所示）。通过使用 CSS 为导航添加一些样式，可以让每个子菜单仅在访问者鼠标停留在其父元素的列表项目上时显示出来（如图 15.7.5 所示），并在访问者将鼠标移走时隐藏。

可以通过多种方法实现这一效果，但这些方法总离不开在选择器中使用 `:hover` 伪类以显示子菜单。下面演示了这样一种在默认情况下隐藏嵌套列表，并在访问者鼠标停留时显示该列表的方法：

```
/* 子菜单的默认状态 */
.nav .subnav {
    left: -999em; /* 将子菜单移出屏幕 */
    position: absolute;
    z-index: 1000;
}

/* 当鼠标停留在父元素li上时子菜单的状态 */
.nav li:hover .subnav {
    left: auto; /* 让子菜单回到自然状态 */
}
```

对应的 HTML 如图 15.7.4 所示。要实现水平排布，移除列表项目的符号，以及对外观进行调整，都需要使用更多的 CSS。图 15.7.5 中所显示页面的完整 HTML 和 CSS 代码见本书配套网站 www.htmlcssvqs.com/8ed/15/dropdown-nav。我还在代码中添加了一些注释。

可以使用类似的方法创建带飞出式子菜单的垂直导航（子菜单显示在导航的侧边）。

15.8 创建描述列表

HTML 提供了专门用于描述成组出现的名称（术语）及其值之间关联的列表类型。这种类型在 HTML5 中称为描述列表（description list），而在 HTML 的早期版本中则称为定义列表。

根据 HTML5 规范，“由名称及其值构成的组合可以是术语和定义、元数据主题和值、问题和答案，以及任何其他的名-值

组。”每个列表都包含在 `dl` 中，其中的每个名-值组都有一个或多个 `dt` 元素（名称或术语）以及一个或多个 `dd` 元素（它们的值）。图 15.8.1 展示了一个基本的描述列表示例。除了使用一个简单的样式规则（如图 15.8.2 所示）应用粗体以外，所有的样式都是默认的（如图 15.8.3 所示）。

```

...
<body>
<h1>List of Horror Movie Legends</h1>

<dl>
  <dt>Boris Karloff</dt>
  <dd>Best known for his role in <cite>Frankenstein</cite> and related horror films, this
  → scaremaster's real name was William Henry Pratt.</dd>

  <dt>Christopher Lee</dt>
  <dd>Lee took a bite out of audiences as Dracula in multiple Hammer horror classics.</dd>

  ... [更多恐怖传说] ...
</dl>

</body>
</html>

```

图 15.8.1 这是最基本的描述列表，每个名-值组都由一个 dt 和一个 dd 构成。每个组之间的空行仅仅是为了提高代码的可读性而添加的。这些空行不是必需的，它们不改变内容的含义，也不影响内容的呈现

```

dt {
  font-weight: bold;
}

```

图 15.8.2 你可能想为 dt 元素中的术语添加一些格式，从而让它们突出显示（参见图 15.8.3）



图 15.8.3 在默认情况下，名称（dt）向左对齐，而值（dd）则有缩进。由于图 15.8.2 中设置的简单样式，名称以粗体显示。如果没有设置该样式，它们将以常规文本显示

以下都是 dl 元素内的 dt 和 dd 元素的组

合，这些安排都是有效的。

- 一个 dt 同一个 dd（参见图 15.8.1，同时参见图 15.8.7 中 Cast 下嵌套描述列表中的 Director）。这是最常见的情形。
- 一个 dt 同多个 dd 元素，参见图 15.8.7 中的 Writers。
- 多个 dt 元素同一个 dd，如图 15.8.4 所示。（样式设置的示例如图 15.8.5 和图 15.8.6 所示。）
- 多个 dt 元素同多个 dd 元素。如果图 15.8.4 中的 bogeyman/boogeyman 有多个定义，就是这种情形。

可以使用 dfn 元素包围 dt 中的名称，指出该列表是用于定义术语的，如在术语表中，如图 15.8.4 所示。（关于 dfn，参见 4.9 节。）

可以对描述列表进行嵌套（如图 15.8.7 所示），并通过 CSS 对它们添加所需的样式（如图 15.8.8 所示）。在默认情况下，如果一个 dl 嵌套在另一个 dl 中，它会自动进行缩进，如图 15.8.9 所示（当然也可以通过 CSS 对此进行修改）。

```
...
<body>

<h1>Defining words with multiple spellings</h1>

<dl>
  <dt><dfn>bogeyman</dfn>, n.</dt>
  <dt><dfn>boogeyman</dfn>, n.</dt>
  <dd>A mythical creature that lurks under
  → the beds of small children.</dd>

  <dt><dfn lang="en-gb">aluminium
  → </dfn>, n.</dt>
  <dt><dfn>aluminum</dfn>, n.</dt>
  <dd>...</dd>
</dl>
</body>
</html>
```

图 15.8.4 在这个例子中，每个名-值组都包含多个 dt 和一个 dd，因为这些术语有一个以上的拼写方式，而它们的定义是相同的

```
dd + dt {
  margin-top: 1em;
}
```

图 15.8.5 这会在每个名-值组之间添加比默认设置更多的空间

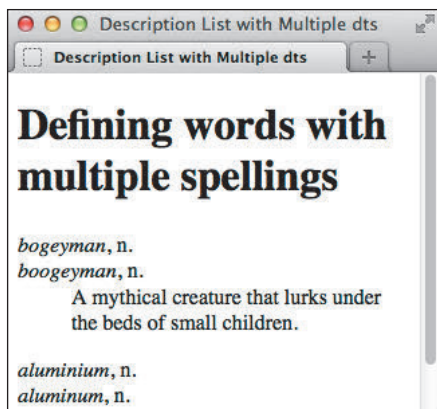


图 15.8.6 现在可以清楚地看到各组描述之间的间隔了。由于 *aluminium, n.* 所在的 dt 紧跟在上一名-值组中的 dd 后面，因此图 15.8.5 中的样式规则发挥了作用

创建描述列表的步骤

- (1) 输入 <dl>。
- (2) 输入 <dt>。
- (3) 输入需要描述或定义的单词或短语，包括任何额外的语义元素（如 dfn）。
- (4) 输入 </dt> 以完成名-值组中的名称部分。
- (5) 如果名-值组中有一个以上的名称或术语，可根据需要重复第 (2) 步至第 (4) 步（参见图 15.8.4）。

```
...
<body>

<h1>Credits for <cite>Amélie</cite></h1>

<dl>
  <dt>Director</dt>
  <dd>Jean-Pierre Jeunet</dd>

  <dt>Writers</dt>
  <dd>Guillaume Laurant (story,
  → screenplay)</dd>
  <dd>Jean-Pierre Jeunet (story)</dd>

  <dt>Cast</dt>
  <dd>
    <!-- 开始嵌套列表 -->
    <dl>
      <dt>Audrey Tautou</dt>
      <dd>Amélie Poulain</dd>

      <dt>Mathieu Kassovitz</dt>
      <dd>Nino Quincampoix</dd>

      ... [rest of Cast] ...
    </dl>
    <!-- 结束嵌套列表 -->
  </dd>

  ... [其他致谢名单] ...
</dl>

</body>
</html>
```

图 15.8.7 这是一个描述列表的例子，它描述了一部电影的导演、编剧和演员表，其中的演员表是由演员姓名及其角色构成的嵌套描述列表。可以根据需要对嵌套的列表设置不同的样式（如图 15.8.8 所示）

- (6) 输入 <dd>。
- (7) 输入第 (3) 步中输入的术语或名称的描述。
- (8) 输入 </dd> 以完成名-值组中的描述 (值) 部分。
- (9) 如果名-值组中有一个以上的定义值, 可根据需要重复第 (6) 步至第 (8) 步 (参见图 15.8.7 中的 Writers)。
- (10) 对每个由术语和描述构成的组, 重复第 (2) 步至第 (9) 步。
- (11) 输入 </dl> 以完成整个描述列表。

```
body {
    font-family: Verdana, Geneva, sans-serif;
}

h1 {
    font-size: 1.75em;
}

dt {
    font-weight: bold;
    text-transform: uppercase;
}

/* 为位于另一个dl中的任意dl的dt设置样式 */
dl dl dt {
    text-transform: none;
}

dd + dt {
    margin-top: 1em;
}
```

图 15.8.8 我想对主要列表中的术语和嵌套列表中的术语进行区分, 因此我对 dt 元素使用了大写字母样式, 再将位于嵌套 dl 中的 dt 元素重新设为常规样式 (使用 text-transform: none; 声明)。不过, 注意所有的术语均以粗体显示 (如图 15.8.9 所示), 这是因为第一条样式规则中的声明适用于所有的 dt 元素, 同时并未在嵌套列表的样式中清除这一样式

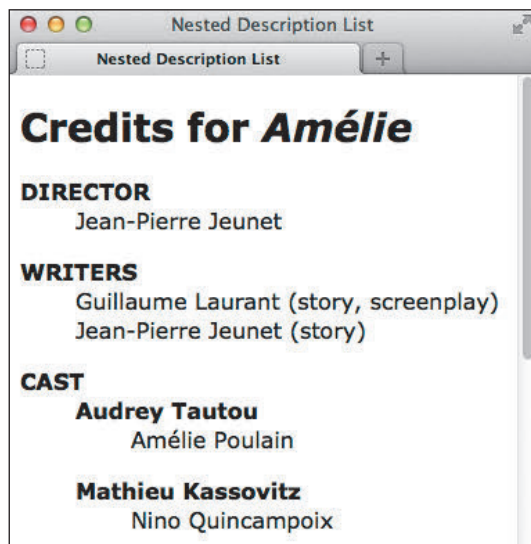


图 15.8.9 在默认情况下, 当一个 dl 嵌套在另一个 dl 中时, 嵌套的列表会自动进行缩进。根据图 15.8.8 中的样式, 第一级 dt 元素使用大写字母, 而嵌套列表中的 dt 元素则使用常规样式。所有的 dt 元素均以粗体显示

提示 对于描述 (值), 浏览器通常会在其术语 (名称) 下面新的一行对其进行缩进 (参见图 15.8.3)。可以通过自定义 dd 元素的 margin-left 值改变缩进。如 dd { margin-left: 0; } 会将描述跟术语左对齐。

提示 你应该已经在示例 (图 15.8.1、图 15.8.4、图 15.8.7) 中看到, 不必 (更确切地说, 不应该) 使用 p 元素对 dd 元素中的单个文本段落进行标记。不过, 如果单个描述是由一个以上的段落构成的, 就应该在一个 dd 元素中使用多个 p 元素对其进行标记, 而不是将每个段落 (不使用 p 元素) 放入单独的 dd。

本章内容

- HTML5 对表单的改进
- 创建表单
- 处理表单
- 对表单元素进行组织
- 创建文本框
- 为表单组件添加标签
- 创建密码框
- 创建电子邮件框、搜索框、电话框和 URL 框
- 创建单选按钮
- 创建复选框
- 创建文本区域
- 创建选择框
- 让访问者上传文件
- 创建隐藏字段
- 创建提交按钮
- 禁用表单元素
- 根据状态为表单设置样式

到目前为止，你学到的所有 HTML 都是用于帮助你将自己的想法告诉访问者的。在本章中，你将学习如何创建表单，与访问者进行交流。

表单有两个基本组成部分：访问者在页面上可以看见并填写的控件、标签和按钮的集合；以及用于获取信息并将其转化为可以读取或计算的格式的处理脚本。本章主要关

注第一部分：创建表单。本书配套网站上提供了许多示例脚本，参见 www.htmlcssvqs.com/8ed/form-scripts。

构造表单的字段和按钮很直观，同创建网页的其他部分是相似的。基本的表单字段类型包括文本框、单选按钮、复选框、下拉菜单、更大的文本区域。如果你曾在网上购过物，曾加入过社交网络，或撰写过基于 Web 的电子邮件，那么你会很熟悉即将在这里学到的表单元素。我还会演示如何使用 CSS 设置表单的样式。

16.1 HTML5 对表单的改进

如果你没有接触过表单，最好先跳过本节，待熟悉本章其余内容之后再回过头来阅读本节。

HTML5 的一个重要的特性就是对表单的改进。过去，我们常常需要花费很多额外的时间，编写 JavaScript 以增强表单行为——例如，要求访问者提交表单之前必须填写某个字段。HTML5 通过引入新的表单元素、输入类型和属性，以及内置的对必填字段、电子邮件地址、URL 以及定制模式的验证，让这一切变得很轻松。这些特性不光帮助了设计人员和开发人员，也让网站访问者的体验有了很大的提升。

更妙的是，不支持这些新特性的旧版浏

览器也不会出太大的问题。它们会直接忽略这些无法理解的属性，表单里的输入框也会正常显示。如果你希望它们拥有 HTML5 行为，可以使用 JavaScript 填补空白，就像过去那样（参见最后一条提示）。

表 16.1.1 和表 16.1.2 总结了 HTML5 大部分与表单相关的特性，并指出了如何获取更多信息。你将看到，我们会将重点放在最常使用的那些特性上面。

表 16.1.1 输入和元素

输入或者元素	简略代码	更多信息
电子邮件框	<code><input type="email"></code>	
搜索框	<code><input type="search"></code>	参见 16.8 节
电话框	<code><input type="tel"></code>	
URL 框	<code><input type="url"></code>	
以下元素得到了部分浏览器的支持		
日期	<code><input type="date"></code>	更多信息参见： www.wufoo.com/html5 浏览器支持： caniuse.com/#feat=input-datetime caniuse.com/#feat=input-number caniuse.com/#feat=input-range
数字	<code><input type="number"></code>	
范围	<code><input type="range"></code>	
	<code><input type="text" name="favfruit" → list="fruit" /></code>	
数据列表	<code><datalist id="fruit"> <option>Grapes</option> <option>Pears</option> <option>Kiwi</option> </datalist></code>	更多信息参见： www.wufoo.com/html5
下面的输入或者元素得到的支持很少，W3C 指出它们在 2014 年定案之时很可能不会列入 HTML5		
颜色	<code><input type="color" /></code>	www.w3.org/html/wg/wiki/HTML5.0AtRiskFeatures
全局日期和时间	<code><input type="datetime" /></code>	
局部日期和时间	<code><input type="datetime-local" /></code>	更多信息参见： www.wufoo.com/html5 浏览器支持： caniuse.com/#feat=input-color caniuse.com/#feat=input-datetime
月	<code><input type="month" /></code>	
时间	<code><input type="time" /></code>	
周	<code><input type="week" /></code>	
输出	<code><output></output></code>	

表 16.1.2 属性

属 性	总 结	更多信息
accept	限制用户可上传文件的类型	www.wufoo.com/html5
autocomplete	如果对 form 元素或特定的字段添加 autocomplete="off", 就会关闭浏览器的对该表单或该字段的自动填写功能。默认值为 on	16.5 节
autofocus	页面加载后将焦点放到该字段	16.5 节
multiple	允许输入多个电子邮件地址, 或者上传多个文件	16.8 节
list	将 datalist 与 input 联系起来	www.wufoo.com/html5
maxlength	指定 textarea 的最大字符数 (在 HTML5 之前的文本框就支持该特性)	16.11 节
pattern	定义一个用户所输入的文本在提交之前必须遵循的模式	16.8 节
placeholder	指定一个出现在文本框中的提示文本, 用户开始输入后该文本消失	16.5 节
required	需要访问者在提交表单之前必须完成该字段	16.5 节
formnovalidate	关闭 HTML5 的自动验证功能。应用于提交按钮	16.15 节
novalidate	关闭 HTML5 的自动验证功能。应用于表单元素	16.2 节

提示 对于浏览器支持信息, caniuse.com 上的信息通常比 www.wufoo.com/html5 上的更新一些, 不过后者仍然是有关 HTML5 表单信息的一个重要资源。

提示 那些有可能不会成为 HTML5 一部分的表单元素, 也有可能被收录到 HTML 5.1 的最终版本 (预计 2016 年发布)。

提示 Ryan Seddon 的 H5F (<https://github.com/ryanseddon/H5F>) 可以为旧的浏览器提供模仿 HTML5 表单行为的 JavaScript 方案。

16.2 创建表单

每个表单都以 form 开始标签开始, 以 form 结束标签结束。两个标签之间是组成表

单的说明标签、控件和按钮, 如图 16.2.1 所示。(注意, 我会交替使用控件 (control) 和字段 (field) 两个词。) 每个控件都有一个 name 属性, 用于在提交表单时对数据进行识别。访问者通过你提供的提交按钮提交表单——触发提交按钮时, 他们填写的数据就会发送至服务器上处理数据的脚本。

了解了表单的全貌之后, 不妨仔细看看其中的细节。form 开始标签可以有一些属性, 其中最重要的就是 action 和 method (如图 16.2.1 所示)。

将 action 属性的值设为访问者提交表单时服务器上对数据进行处理脚本的 URL。例如, action="save-info.php"。

method 属性的值要么是 get, 要么是 post。大多数情况下都可以使用 post, 不过每种方法都有其用途, 了解其用途有助于理解它们。更多细节参见本节的补充材料。

```

...
<body>
<h1>Create a New Account</h1>
<form method="post" action="show-data.php">
  <!-- 各种表单元素 -->
  <fieldset>
    <h2 class="hdr-account">Account</h2>

    <div class="fields">
      <p class="row">
        <label for="first-name">First Name:</label>
        <input type="text" id="first-name" name="first_name" class="field-large" />
      </p>
      <p class="row">
        <label for="last-name">Last Name:</label>
        <input type="text" id="last-name" name="last_name" class="field-large" />
      </p>
      ...
    </div>
  </fieldset>
  ... 更多表单元素 ...

  <!-- 提交按钮 -->
  <input type="submit" value="Create Account" class="btn" />
</form>
</body>
</html>

```

图 16.2.1 每个表单包括 form 元素自身、访问者输入信息的其他表单元素以及用于将收集到的信息发送给服务器的提交按钮

创建表单的步骤

- (1) 输入 `<form method="formmethod">`，这里的 formmethod 是 get 或者 post。
- (2) 输入 `action="script.url">`，这里的 script.url 是提交表单时要运行的脚本在服务器上的位置。
- (3) 根据从 16.5 节开始讲解的知识，创建表单的内容（包括一个提交按钮）。
- (4) 输入 `</form>` 以结束表单。

提示 可以使用 CSS 布局表单元素，参见图 16.2.2。此处演示的这个贯穿本章的表单示例显示在浏览器中如图 16.2.3 所示。

提示 图 16.2.1 和图 16.2.2 中完整的 HTML 和 CSS 代码可以从本书配套网站下载（www.htmlcssvqs.com/8ed/16），其中部分代码会贯穿本章。此外，配套网站上还有表单在 action 属性中引用的 show-data.php 脚本。学习本章过程中，你可以随意使用它来测试你的表单。需要注意的是，除非你的服务器安装了 PHP，否则这个脚本不工作。

提示 可以对 form 元素应用 novalidate 属性，关闭表单的 HTML5 验证特性（本章后面会演示这一特性）。例如 `<form method="post" action="show-data.php" novalidate>`。

```
fieldset {
    background-color: #f1f1f1;
    border: none;
    border-radius: 2px;
    margin-bottom: 12px;
    overflow: hidden;
    padding: 0 .625em; /* 10px */
}

.fields {
    background-color: #fff;
    border: 1px solid #eaeaea;
    margin: .75em; /* 12px */
    padding: .75em;
}

.fields .row {
    margin: 0.5em 0;
}

label {
    cursor: pointer;
    display: inline-block;
    padding: 3px 6px;
    text-align: right;
    width: 150px;
    vertical-align: top;
}

input, select, button {
    font-size: inherit;
}

/* 各种表单字段的宽度 */
.field-small {
    width: 75px;
}

.field-medium {
    width: 150px;
}

.field-large {
    width: 250px;
}
```

图 16.2.2 这是用于对表单进行格式化的样式表的一部分。完整的样式表见本书的配套网站

Create a New Account

ACCOUNT

First Name:

Last Name:

Email:

yourname@example.com

Password:

Re-enter Password:

ADDRESS

Street Address:

City:

State:

Alabama

ZIP Code:

PUBLIC PROFILE

Picture:

Browse...

Maximum size of 700k. JPG, GIF or PNG.

Screen Name:

Website URL:

http://www.example.com

Have a homepage or a blog? Put the address here, beginning with http:// or https://.

Bio:

Gender:

Male

Female

EMAILS

☐ It is okay to email me with messages from other users.

☐ It is okay to email me with occasional promotions about our other products.

Create Account

图 16.2.3 这是本章讨论的完整的 New Account 表单

method="get" 与 method="post" 的区别

前面提到, form 元素的 method 属性要么是 get, 要么是 post。

如果对表单使用 method="get", 那么表单提交后, 表单中的数据会显示在浏览器的地址栏里。通常, 如果你希望表单提交后从服务器得到信息, 就使用 get。例如, 大多数搜索引擎都会在搜索表单中使用 get——输入 Kermit meets Yoda, 提交表单, 搜索引擎会得到搜索结果。由于数据出现在 URL 中, 因此用户可以保存搜索查询, 或者将查询发给朋友。

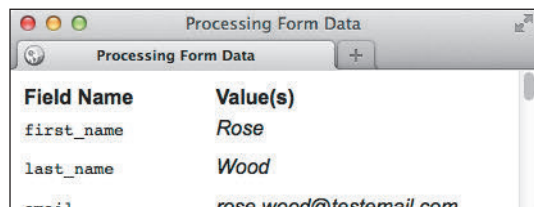
如果对表单使用 method="post", 那么提交表单后, 表单中的数据不会显示在浏览器的地址栏里, 这样更为安全。同时, 比起 get, 使用 post 可以向服务器发送更多的数据。通常, post 用于向服务器存入数据, 而非获取数据。因此, 如果需要在数据库中保存、添加和删除数据, 就应选择 post。例如, 电子商务网站使用 post 保存信用卡、邮件地址以及其他用户输入的信息。

通常, 如果不确定使用哪一种, 就使用 post, 这样数据不会暴露在 URL 中。

16.3 处理表单

表单从访问者那里收集信息, 脚本则对这些信息进行处理。脚本可以将信息记录到服务器上的数据库里, 通过电子邮件发送信息, 或者执行很多其他的功能。

有很多语言都可以用于编写表单处理脚本。对于刚起步的新手, PHP 是个不错的选择, 因为用它处理一些常见任务很简单。而且, 有很多关于 PHP 的书籍、在线教程、论坛帮你学习。关于 PHP 的内容超出了本书的范围, 不过我在 www.htmlcssvqs.com/8ed/form-scripts 提供了两个基本的示例脚本(如图 16.3.1 所示)。



Field Name	Value(s)
first_name	Rose
last_name	Wood
email	rose.wood@testemail.com

图 16.3.1 本书配套网站演示的脚本之一是 show-data.php。这里是它的一部分, 提交表单后, 它会将用户填写的所有字段的名和值显示出来。另一个脚本(email-data.php)会将表单数据发送至脚本中指定的电子邮件地址

除了 PHP, 我们还可以选择其他语言, 如 Django (一个使用 Python 的框架)、Ruby on Rails、ASP.NET、JSP(JavaServer Pages)等。

表单安全性

从服务器接收数据尤其需要注意安全性。不要对数据作任何假设, 因为即便对表单建立了安全措施, 坏人也可以创建他们自己的表单, 调用你的脚本并发送无数的垃圾信息。他们还可能提交恶意文本, 损坏服务器上的数据。保护表单是个前沿主题, 我在 www.htmlcssvqs.com/8ed/form-security 提供了一些链接。

表单验证

表单验证指的是提交表单时, 对用户输入的每个字段的内容进行检查, 看是否符合预期的格式(例如, 对于电子邮件字段, 检查输入是否为正确的电子邮件地址格式)。先前提到, 有的表单元素有内置的验证功能。有的网站使用 JavaScript 进行验证(网

上有相关的教程和代码)。这些都不能完全替代服务器端验证,因为旧的浏览器或禁用 JavaScript 的浏览器不会执行客户端验证。进行服务器端验证也是出于安全性考虑(参见前面的补充材料“表单安全性”)。总之,必须考虑将服务器端的验证加入表单处理脚本。

服务器端与客户端

PHP 是一种服务器端(server-side)语言,这意味着它运行于为网页服务的计算机(称为服务器),而不是查看网页的访问者的计算机。脚本必须上传到服务器上才能发挥作用,这通常跟你托管网页、图像等用的是同一个服务器(相关内容参见 21.2 节)。此外,服务器必须已经安装 PHP 才能对脚本进行解释。大多数 Web 主机都为你安装了 PHP,因此很容易找到支持 PHP 的服务器。专业网站的很多功能(如存储数据、发送电子邮件等)都需要服务器端语言。

客户端(client-side)语言(如 HTML 和 CSS)是在浏览器中运行的。JavaScript 是另一种客户端语言(也可以用在服务器端)。它们可以在完全不与服务器交互的情况下执行很多任务。例如,可以使用 JavaScript 在提交表单前检查是否所有数据都已填写,以及其他不需要服务器(或在涉及服务器之前)的任务。

16.4 对表单元素进行组织

如果表单上有很多信息需要填写,可以使用 `fieldset` 元素将相关的元素组合在一起,使表单更容易理解。表单越容易让访问者理

解,访问者就越有可能正确地填写表单。还可以使用 `legend` 元素为每个 `fieldset` 提供一个标题(caption),用于描述每个组的目的,有时这些描述还可以使用 `h1 ~ h6` 标题,如图 16.4.1 所示。对于一组单选按钮(参见 16.9 节),`legend` 元素尤其重要,因为通常如果不配合使用 `legend`,单选按钮(参见 16.9 节)就没有明显的上下文。

即便不添加任何 CSS,浏览器也会让哪些控件属于哪个 `fieldset` 显得相当清晰(如图 16.9.2 所示)。当然,你可以自己为 `fieldset` 和 `legend` 添加样式,从而让表单更吸引人,更便于使用(如图 16.9.3 和图 16.9.4 所示)。

对表单元素进行组织的步骤

(1) 在 `form` 开始标签的下面、任何希望包含在第一个组的表单元素的上面,输入 `<fieldset>`。

(2) 如果需要,输入 `<legend>`。(如果需要包含 `legend`,它必须是 `fieldset` 里的第一个元素。)

(3) 输入标签的文本。

(4) 输入 `</legend>` 以完成标签。

(5) 如果不使用 `legend`,则创建一个标题从而可以识别属于该 `fieldset` 的一组控件。(参见补充材料“`legend` 元素、标题、屏幕阅读器和样式”。)

(6) 创建属于第一组的表单元素,更多信息参见从 16.5 节开始讲解的知识。

(7) 输入 `</fieldset>` 结束第一组表单元素。

(8) 为每一组表单元素重复第(1)步至第(7)步。

```

...
<h1>Create a New Account</h1>
<form method="post" action="show-data.php">
  <fieldset>
    <h2 class="hdr-account">Account</h2>
    ... Account字段 ...
  </fieldset>

  <fieldset>
    <h2 class="hdr-address">Address</h2>
    ... Address字段 ...
  </fieldset>

  <fieldset>
    <h2 class="hdr-public-profile">Public Profile</h2>
    ... Public Profile字段 ...

    <div class="row">
      <fieldset class="radios">
        <legend>Gender:</legend>
        <input type="radio" id="gender-male" name="gender" value="male" />
        <label for="gender-male">Male</label>

        <input type="radio" id="gender-female" name="gender" value="female" />
        <label for="gender-female">Female</label>
      </fieldset>
    </div>
  </fieldset>

  <fieldset>
    <h2 class="hdr-emails">Emails</h2>
    ... Emails字段 ...
  </fieldset>

  <input type="submit" value="Create Account" class="btn" />
</form>
...

```

图 16.4.1 我对四个表单部分分别使用了 fieldset，并将 Public Profile（公开资料）部分的 Gender（性别）单选按钮使用一个 fieldset 包围起来。注意，我为被嵌套的 fieldset 添加了 radios 类，方便为其添加特定的样式，同时，还在其中添加了一个 legend 元素，用于描述单选按钮（为确保尽量简单，这段代码中移除了一些 div。这段被嵌套的 fieldset 的完整代码见 16.9 节）

提示 使用 fieldset 元素对表单进行组织是可选的，使用 legend 也是可选的（使用 legend 则必须要有 fieldset）。不过我强烈推荐使用 fieldset 和 legend 对相关的单选按钮进行分组。

提示 CSS 对 legend 元素的样式修改能力有限，尤其是关于定位的样式。如果遇到麻烦，可以上网搜索相关的解决办法。

图 16.4.2 没有添加任何 CSS 的情况下，你可以看到，浏览器默认为每个 fieldset 添加了一个边框，Public Profile 里面嵌套的 Gender 也有边框

```

fieldset {
    background-color: #f1f1f1;
    border: none;
    border-radius: 2px;
    margin-bottom: 12px;
    overflow: hidden;
    padding: 0 .625em;
}

.radios { /* nested fieldset */
    background-color: transparent;
    position: relative;
    margin-bottom: 0;
}

h2 {
    background-color: #dedede;
    border-bottom: 1px solid #d4d4d4;
    border-top: 1px solid #d4d4d4;
    border-radius: 5px;
    box-shadow: 3px 3px 3px #ccc;
    color: #fff;
    font-size: 1.1em;
    margin: 12px;
    padding: 0.3em 1em;
    text-shadow: #9FBEB9 1px 1px 1px;
    text-transform: uppercase;
}

.hdr-account { background-color: #0b5586; }
.hdr-address { background-color: #4494c9; }
.hdr-public-profile { background-color:
→ #377d87; }
.hdr-emails { background-color: #717f88; }

```

图 16.4.3 我为所有的 fieldset 元素添加了外边距、背景颜色和内边距，并为每个标题添加了特殊的背景颜色

legend 元素、标题、屏幕阅读器和样式

legend 元素可以提高表单的可访问性。对于每个表单字段，屏幕阅读器都会将与之关联的 legend 文本念出来，从而让访问者了解字段的上下文。这种行为在不同的屏幕阅读器和浏览器上并不完全一样，不同的模式下也不一样（屏幕阅读器的用户有多种听读和导航的模式）。

很多情况下，legend 文本会如你所预期的那样念出来。有时，文本不会念出来；有时，屏幕阅读器又会走向另一个极端，针对每个控件都将 legend 文本念一遍。此外，有的浏览器会限制可以对 legend 添加的样式，或者让添加样式变得更难。

考虑到以上这些问题,使用 h1 ~ h6 标题代替 legend 来识别一些 (并非所有) fieldset,就像我之前所做的那样,参见图 16.4.1,就是一个明智的选择。JAWS (用户数量最多的屏幕阅读器) 会将标题与相关联的表单字段连在一起阅读,因此可访问性不会受到影响。而且,屏幕阅读器允许用户通过标题在页面内切换内容。此外,还可以很轻松地对标题添加样式,这也是一个额外的好处。

并非所有的表单都是相似的,因此你所使用的方法可以有一些差异。无论如何,对于单选按钮,最好总是使用 fieldset 和 legend。

图 16.4.4 新添加的 CSS 将不同分组的表单字段清晰地隔开了

16.5 创建文本框

文本框可以包含一行无格式的文本,它可以是访问者想输入的任何内容,通常用于姓名、地址等信息。

每个文本框都是通过带有 type="text" 的 input 标签表现出来的。除了 type 之外,还有一些其他可用的属性,其中最重要的就是 name (参见图 16.5.1)。服务器端的脚本使用你指定的 name 获取访问者在文本框中输入的值或预设的值 (即 value 属性值)。事实上, name

和 value 对其他的表单字段类型来说,也是很重要的,你将在本章余下的部分看到这一点。

```
...
<form method="post" action="show-data.php">
<fieldset>
  <h2 class="account">Account</h2>
  <div class="fields">
    <p class="row">
      <label for="first-name">First Name:
      → </label>
      <input type="text" id="first-
      name" name="first_name"
      → class="field-large" required
      → aria-required="true" />
    </p>
    <p class="row">
      <label for="last-name">Last Name:
      → </label>
      <input type="text" id="last-
      name" name="last_name"
      → class="field-large" />
    </p>
    <p class="row">
      <label for="email">Email:</label>
      <input type="email" id="email"
      → name="email" placeholder=
      → "yourname@example.com"
      → class="field-large" />
    </p>
    ... 更多字段 ...
  </div>
...

```

图 16.5.1 必须为每个文本框设置 name 属性,只有在希望为文本框添加默认值的情况下才需要设置 value 属性。第三个文本框使用了 placeholder 属性 (参见图 16.5.2), 第一个则使用了 required 属性 (参见图 16.5.3)。注意,我还设置了 aria-required="true" (参见提示)。这个例子还演示了 name 可以与 for 和 id 不一样,也可以与它们一样 (参见 16.6 节的补充材料)

创建文本框的步骤

(1) 如果需要, 输入用于让访问者识别文本框的标签, 例如 `<label for="idlabel">Last Name:</label>`, 其中 `idlabel` 匹配第 (4) 步中的标签, 关于这一点, 下一节会给出更多解释。

(2) 输入 `<input type="text"`。

(3) 输入 `name="dataname"`, 这里的 `dataname` 是用于让服务器 (和脚本) 识别输入数据的文本。

(4) 如果在第 (1) 步中创建了 `label`, 输入 `id="idlabel"`, 这里的 `idlabel` 跟第 (1) 步中 `for` 属性的文本一样。这跟具有显式文本框的 `label` 元素关联。很多程序员将 `id` 和 `name` 设为同一个值, 尽管这并不是必需的 (图 16.5.1 展示了这两种方法)。

(5) 如果需要, 输入 `value="default"`, 这里的 `default` 是这个字段中最初显示的数据, 如果访问者没有输入别的内容的话, 这一数据将被发送到服务器。

(6) 如果需要, 输入 `placeholder="hinttext"`, 这里的 `hinttext` 是这个字段中最初显示的数据, 用于指导用户的输入, 如图 16.5.2 所示。当 `input` 元素获得焦点时, 这些文本将会消失, 让用户输入内容。

(7) 如果需要, 输入 `required="required"`, 表示仅在这个字段有值的情况下才能提交表单, 如图 16.5.3 所示。

(8) 如果需要, 输入 `autofocus` 或 `autofocus="autofocus"` (在 HTML5 中这两种方法均可), 如图 16.5.4 所示。如果这是第一个拥有此属性的表单控件, `input` 元素在页面加载时会默认获得焦点。

(9) 如果需要, 通过输入 `size="n"` 定义文本框的大小, 这里的 `n` 是需要设置的文本框宽度, 以字符为单位。也可以使用 CSS 设置输入框的宽度, 参见图 16.5.5。

(10) 如果需要, 输入 `maxlength="n"`, 这里的 `n` 是该文本框允许输入的最大字符数。

(11) 最后, 输入 `>` 或 `/>`, 结束文本框 (参见最后一条提示)。

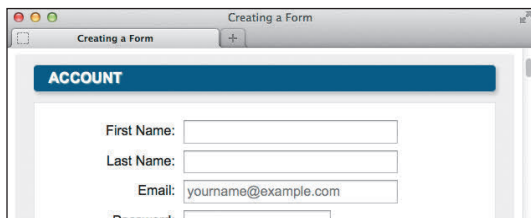


图 16.5.2 占位符是为用户填写表单提供提示和额外指导的好方法。placeholder 属性中的文本会以浅灰色出现在文本框中。当用户开始在字段中输入文本时, 浅灰色文本就会消失。(一种例外情况是: 当字段获得焦点, 而非用户开始输入的时候, IE10 会隐藏占位符文本。)如果用户在没有输入任何信息的情况下将焦点移开, 浅灰色文本又会再次出现。这是 HTML5 的另一项新特性, 旧浏览器会忽略它

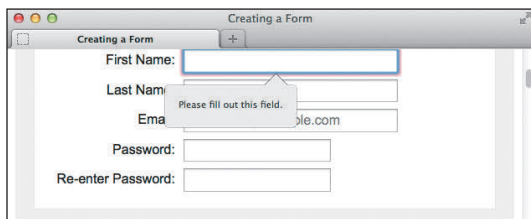


图 16.5.3 如果用户在提交表单时有一个带 required 属性的字段没有填写, 浏览器会显示一个类似的提示。这个消息的样式在不同的浏览器中会有差异。这个特性是 HTML5 专有的, 因此旧的浏览器会忽略 required (不过表单仍可正常工作)。在 16.3 节中提到, 针对旧的浏览器, 可以使用 JavaScript 对表单进行验证, 此外, 无论用户使用哪种浏览器, 都应该在服务器端验证表单

```
<input type="text" id="first-name"
→ name="first_name" class="field-large"
→ required aria-required="true" autofocus />
```

图 16.5.4 页面加载时, 最好有一个字段自动获取焦点, 这样用户就可以立即开始输入。要实现这一功能, 可以使用 autofocus 属性

图 16.5.5 针对不同的字段类型，可以使用不同尺寸的文本框。在我们的例子中，我们使用 CSS 通过类设置宽度

图 16.5.6 输入 Ma 后，Firefox 根据先前的表单输入给出了一些包含同样字母的建议文本。我可以从提示文本列表中进行选择，也可以继续输入。如果整个表单或者这个特定的 input 设置了 `autocomplete="off"`，就不会显示这些提示，浏览器也不会保存用户所输入的内容

分隔表单元素

对各个表单元素进行分隔有很多方法。在这些例子中，我们大多数情况下使用 `p` 元素，还有一些情况下使用 `div` 元素（譬如嵌套 `p` 的情况），参见图 16.5.1。所有的示例中都使用了 `class="row"` 作为定义样式的钩子。此外，有的人使用 `ol` 或 `ul` 来组织表单的元素。通常，列表对屏幕阅读器用户来说很有帮助。

不过有的屏幕阅读器在朗读放在列表中的表单元素时会有一些问题，并让用户感到疑惑。因此，我选择 `p` 和 `div`。

提示 尽管 `label` 是可选的，但我强烈推荐使用它们。它们对提升表单的可访问性和可用性有很重要的作用。

提示 如果访问者跳过了一个字段，而且你没有为该字段设置默认值，那么当用户提交表单时，发送到服务器的 `name` 属性就是未定义的空值。

提示 不要将 `placeholder` 属性（图 16.5.1 和图 16.5.2 中都有例子）同 `value` 属性弄混。它们都会让文本框默认出现一些文本，但 `placeholder` 文本会自动消失，且不会被发送到服务器，而 `value` 在输入框获得焦点时不会消失，且这些内容会被发送到服务器。由此看来，如果一个文本框同时具有 `placeholder` 和非空 `value`，后者会显示在文本框中。

提示 默认情况下，大多数浏览器会保存用户输入的文本，这样就可以在日后节省用户输入的时间（参见图 16.5.6）。可以通过对 `input` 添加 `autocomplete="off"` 来关闭这一特性。这对于要求输入敏感信息（如信用卡号）的字段来说是很有用的。如果将它用于 `form` 元素（例如，`<form method="post" action="process.php" autocomplete="off">`），那么其中的每个字段都会这样。

提示 我们所说的“获取焦点”或“拥有焦点”，指的是光标位于文本框内，访问者可以随时输入值的状态。

提示 文本框大小的默认值为 20。不过，访问者可以输入更多的字符，直到达到 `maxlength` 属性规定的限制。如果需要更大的可以容纳好几行文本的字段，最好使用 `textarea` 元素（参见 16.11 节）。

提示 就像其他自关闭的元素（如 `img`）一样，使用 `>` 或 `/>` 结尾在 HTML5 中都是有效的。无论你选择哪种方式，一定要保持一致。

16.6 为表单组件添加说明标签

标签（`label`）是描述表单字段用途的文本。例如，在访问者应该输入其姓名中名的文本字段之前，可能有“First Name:”的字样。可以使用 `label` 元素（有点小意外吧）标记这些文字说明标签。你或许已经注意到前面的示例中已经使用过 `label` 元素了。

`label` 元素有一个特殊的属性：`for`。如果 `for` 的值与一个表单字段的 `id` 的值相同，该 `label` 就与该字段显式地关联起来了（如图 16.6.1 所示）。这对提升表单的可用性和可访问性都有帮助。例如，如果访问者与标签有交互（如使用鼠标点击了标签），与之对应的表单字段就会获得焦点（如图 16.6.2 所示）。这种关联还可以让屏幕阅读器将文本标签与相应的字段一起念出来。设想，这对不了解表单字段含义的视障用户来说是多么重要。出于这些原因，我强烈建议你在 `label` 元素中包含 `for` 属性。

```
...
<fieldset>
  <h2 class="account">Account</h2>
  <div class="fields">
    <p class="row">
      <label for="first-name">
        → First Name:</label>
      <input type="text"
        → id="first-name"
        → name="first_name"
        → class="field-large" />
    </p>
    <p class="row">
      <label for="last-name">
        → Last Name:</label>
      <input type="text"
        → id="last-name"
        → name="last_name"
        → class="field-large" />
    </p>
    ... 更多字段 ...
  </div>
</fieldset>
...
```

图 16.6.1 每个 `label` 的 `for` 属性与该标签对应的表单字段的 `id` 是匹配的，这样可以将该标签与该字段显式关联起来

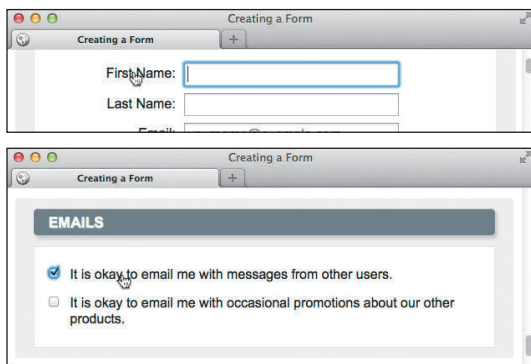


图 16.6.2 如果访问者与文本框的标签有交互，光标就会位于该文本框内，方便用户输入。同时，复选框和单选按钮的标签则让用户可以通过点击标签来修改状态（就像点击表单控件一样）。这里，选了一个复选框

为表单组件添加正式标签

- (1) 输入 `<label>`。
- (2) 如果需要, 输入 `for="idlabel">`, 这里的 *idlabel* 是对应表单元素的 `id` 属性值。
- (3) 输入标签的内容。
- (4) 输入 `</label>`。
- (5) 如果创建的是表单元素, 确保包括 `id` 属性, 对应于第 (2) 步中指定的 `for` 属性。

提示 可以使用 CSS 对标签添加样式。图 16.6.3 的示例让每个标签都与它旁边的字段对齐 (参见图 16.6.2)。

提示 `for`、`id` 和 `name` 属性都可以拥有任意值, 只要没有空格。相关信息参见本页补充材料中的介绍。

提示 还可以将一个表单字段放在一个包含标签文本的 `label` 内, 例如, `<label>First Name: <input type="text" name="first_name" /></label>`。(注意在这种情况下, 就不需要使用 `for` 和 `id` 了。)不过, 将标签与字段分开是更常见的做法 (参见图 16.6.1), 原因之一是这更容易添加样式。

提示 有时, `placeholder` 会被误作 `label` 的替代属性。一定要确保 `placeholder` 只能用作一种提示。

```
/* 表单字段前面的标签 */
label {
    cursor: pointer;
    display: inline-block;
    padding: 3px 6px;
    text-align: right;
    width: 150px;
    vertical-align: top;
}

/* 复选框后面的标签 */
.checkboxes label {
    text-align: left;
    width: 475px;
}
```

图 16.6.3 对标签设置样式, 是让表单变得更方便使用和更好看的重要方面。如果为 `label` 指定了 `cursor: pointer`, 当访问者指向标签时, 就会显示为手形, 而不是默认的箭头。显示为手形就能提示用户这是一个可以操作的元素。使用 `vertical-align: top`; 则会让标签与相关的表单字段对齐

id、for 和 name 属性的命名习惯

正如先前提到的, 让 `for`、`id` 和 `name` 属性值都一样是一种并非必需但很常见的做法。(单选按钮和复选框是例外, 因为它们来说, 有一组 `input` 会使用同一个 `name`, 而 `id` 要求对每个 `input` 来说都是唯一的。)本章的例子中使用的值都是一个单词, 例如 `for="email"`、`id="email"` 和 `name="email"`。

对于包含多个单词的值, 我们在 `for` 和 `id` 中使用连字符 (-) 分隔各个单词, 在 `name` 中使用下划线 (_), 参见图 16.6.1。例如, `for="first-name"`、`id="first-name"` 和 `name="first_name"`。对 `name` 使用不同的形式, 表现出该属性的值可以与另外两个不同, 实际上对需要传入服务器端脚本的 `name` 值使用下划线分隔多个单词也是一种常见的做法。

无论使用哪种方法, 一定要保证 `for` 和 `id` 的值是一样的。

16.7 创建密码框

密码框与文本框的唯一区别是，密码框中输入的文本会使用圆点或星号进行隐藏，如图 16.7.1 和图 16.7.2 所示。

```
...
<p class="row">
  <label for="password">Password:</label>
  <input type="password" id="password"
    → name="password" />
</p>
<p class="row">
  <label for="password2">Re-enter
    → Password:</label>
  <input type="password" id="password2"
    → name="password2" />
</p>
...
```

图 16.7.1 使用 type="password" 创建密码框，而不要用 type="text"

图 16.7.2 当访问者在表单中输入密码时，密码用圆点或星号隐藏起来了。但提交表单后访问者输入的真实值会被发送到服务器。信息在发送过程中没有加密

创建密码框的步骤

- (1) 输入用于让访问者识别密码框的标签，跟 16.6 节中的一样。
- (2) 输入 `<input type="password">`。
- (3) 输入 `id="idlabel"`，这里的 `idlabel` 跟第 (1) 步中 `label` 的 `for` 属性值一样。

(4) 输入 `name="dataname"`，这里的 `dataname` 是用于让服务器识别输入数据的文本。

(5) 通过输入 `size="n"` 定义密码框的大小，这里的 `n` 是需要设置的密码框宽度，以字符为单位。

(6) 如果需要，输入 `maxlength="n"`，这里的 `n` 是该密码框允许输入的最大字符数。

(7) 如果需要，输入 `required` 或者 `required="required"`（参见 16.5 节）。

(8) 如果需要，输入 `autofocus` 或者 `autofocus="autofocus"`（参见 16.5 节）。

(9) 输入 `>` 或 `/>`，结束密码框。

提示 即便密码框中没有输入任何内容，`name` 属性仍将被发送到服务器（使用未定义的 `value`）。

提示 密码框提供的唯一保护措施就是防止其他人看到用户输入的密码。如果要真正地保护密码，可以使用安全服务器（<https://>）。

16.8 创建电子邮件框、搜索框、电话框和 URL 框

电子邮件、电话和 URL 这几种输入类型是 HTML5 中新增的，如图 16.8.1 所示。它们看起来同文本框很相似，但却有一些小而有用的特性，用于帮助验证和输入内容，如图 16.8.2 ~ 图 16.8.4 所示。过去，我们必须依赖 JavaScript 在浏览器中创建这些功能。

搜索框也是 HTML5 中新增的输入类型（如图 16.8.5 所示）。搜索框跟文本框很像，只是很多浏览器将搜索框显示得像操作系统中默认搜索框，参见图 16.8.6。

```

...
<p class="row">
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" class="field-large" />
</p>
<div class="row">
  <label for="website">Website URL:</label>
  <input type="url" id="website" name="website" class="field-large"
  → placeholder="http://www.example.com" />
  <p class="instructions">Have a site or a blog? Put the address here, beginning with
  → <kbd>http://</kbd> or <kbd>https://</kbd>.</p>
</div>
<p class="row">
  <label for="phone">Phone:</label>
  <input type="tel" id="phone" name="phone" class="field-large" placeholder="xxx-xxx-xxxx"
  → pattern="\d{3}-\d{3}-\d{4}" />
</p>
...

```

图 16.8.1 合适的 type 属性值分别指定电子邮件框、URL 框和电话框。pattern 属性用于定制验证规则。它使用正则表达式对用户框中输入的内容进行限制。对于这个例子中的电话输入框，pattern 属性要求“只接受以下格式的输入：xxx-xxx-xxxx，其中 x 均为数字”（许多电话号码都是这样的）。不必对那些不同寻常的正则表达式语法感到担心，可以在 <http://html5pattern.com> 找到一些常用的正则表达式，并将它们复制粘贴到自己的 pattern 属性中

16

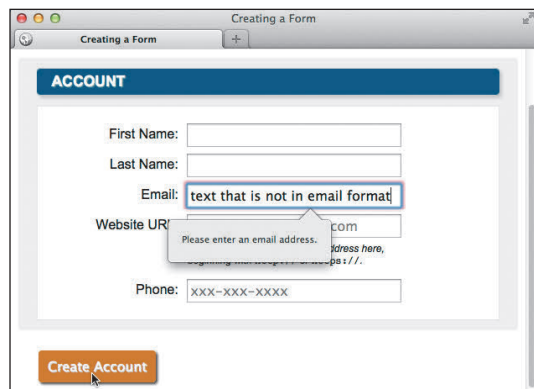


图 16.8.2 当访问者在现代浏览器中提交表单时，浏览器会对 Email 字段的输入文本进行检查，确保其符合电子邮件地址的格式。如果格式不合法，会显示如图所示的错误信息，且鼠标重置到该字段，以便访问者修改文本

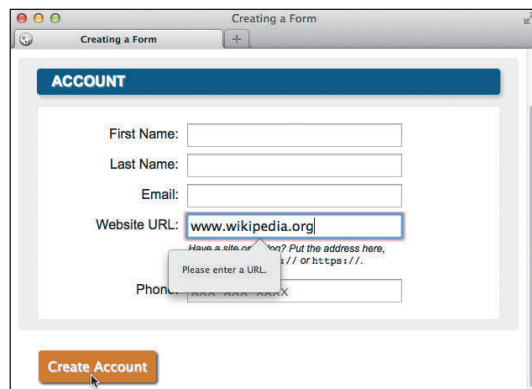


图 16.8.3 当访问者在现代浏览器中提交表单时，浏览器会对 Website URL 字段的输入文本进行检查，确保其符合 URL 格式。注意，www.wikipedia.org 并不是有效的 URL，因为 URL 必须以 http:// 或 https:// 开头。这里最好使用占位符提示访问者。另外，我还在该字段下面的解释文本中指出了合法的格式。从图 16.8.4 可以看得很清楚

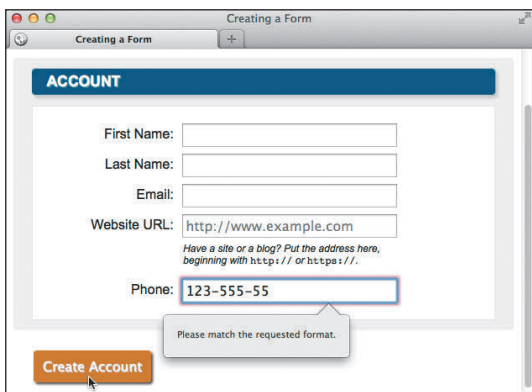


图 16.8.4 当访问者在现代浏览器中提交表单时，浏览器会对 Phone 字段的输入文本进行检查，确保其符合 pattern 属性中指定的电话号码格式。同时，这种字段对于使用 iOS（iPhone 和 iPad）上的 Safari 的用户来说也很方便，因为它会启动数字键盘而非常见的标准键盘

```
<form method="get" action="search-results.php"
→ role="search">
  <label for="search">Search:</label>
  <input type="search" id="search"
  → name="search" size="30"
  → placeholder="e.g., a book or
  → magazine" />
  <input type="submit" value="Find It!" />
</form>
```

图 16.8.5 搜索框是应用 placeholder 的天然场合。同时，注意这里的 form 用的是 method="get" 而不是 method="post"。这是搜索字段的常规做法（无论是 type="search" 还是 type="text"）。关于 role="search"，参见提示

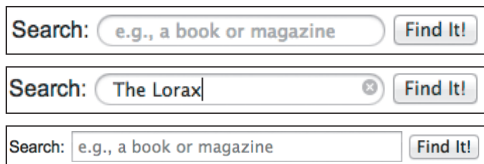


图 16.8.6 OS X 上的 Chrome（如上面的两幅图）、Safari 以及 iOS 上的 Mobile Safari 会让搜索框显示为圆角边框。当用户开始输入，字段右侧会出现一个 x 按钮，用于清除输入的内容。在其他浏览器中，它显示为常规文本框的样子，如最下面一张图（更多关于样式的讨论，参见提示）

创建电子邮件框、搜索框、电话框和 URL 框的步骤

(1) 输入用于让访问者识别输入框的标签，参见 16.6 节。

(2) 对于电子邮件框，输入 `<input type="email">`；对于搜索框，输入 `<input type="search">`；对于电话框，输入 `<input type="tel">`；对于 URL 框，输入 `<input type="url">`。

(3) 输入 `id="idlabel"`，这里的 `idlabel` 跟第 (1) 步中 `label` 的 `for` 属性相同。

(4) 输入 `name="dataname"`，这里的 `dataname` 是用于让服务器识别输入数据的文本。

(5) 如果需要，输入 `value="default"`，这里的 `default` 是最初显示在字段里的数据，也是访问者什么都不输入时将要发送给服务器的数据。

(6) 如果需要，输入 `placeholder="hinttext"`，这里的 `hinttext` 是最初显示在字段里作为对用户输入内容的提示的数据（参见 16.5 节）。

(7) 如果需要，输入 `required` 或者 `required="required"`（参见 16.5 节）。

(8) 如果需要，输入 `autofocus` 或者 `autofocus="autofocus"`（参见 16.5 节）。

(9) 如果需要，通过输入 `size="n"` 定义框的大小，这里的 `n` 是需要设置的框的宽度，以字符为单位。也可以使用 CSS 设置输入框的宽度（参见 16.5 节）。

(10) 如果需要，输入 `maxlength="n"`，这里的 `n` 是该框允许输入的最大字符数。

(11) 最后，输入 `>` 或者 `/>`，结束输入框（两种方法在 HTML5 中均可）。

提示 截至本书写作之际，Chrome、Firefox、IE10 和 Opera 10+ 都提供了对电子邮件框和 URL 框的验证功能。注意这些功能只是为网站创立者和访问者提供了一些便利，但服务器端的验证还是很有必要的（参见 16.3 节）。即便你使用像之前提到过的 H5F（<https://github.com/ryanseddon/H5F>）这样的 JavaScript 来为旧浏览器提供 HTML5 功能，也必须建立服务器端验证。

提示 空的电子邮件框、电话框或 URL 框也会通过验证，除非有 `required` 属性。不过，如果对电话框添加了 `required`，但没有为 `pattern` 属性提供内容，浏览器仍会要求用户填写内容，但任何文本（包括数字、字母、字符等）都会通过验证。

提示 电子邮件框也可以使用 `multiple` 属性，这样就可以输入多个电子邮件地址（以逗号分隔不同的地址）。

提示 浏览器不会检查输入的电子邮件地址或 URL 是否真实存在，它只检查格式是否正确。

提示 这些输入类型也支持 `autocomplete` 属性，更多信息参见 16.5 节的提示。

提示 默认情况下，为 Chrome、Safari 和 Mobile Safari 等浏览器中的搜索框设置样式是受到限制的（参见图 16.8.6）。如果要消除这一约束，重新获得 CSS 的控制权，可以使用专有的 `-webkit-appearance: none;` 声明，例如 `input[type="search"] { -webkit-appearance: none; }`。更多信息（包括对 Firefox 的支持）参见 <http://css-tricks.com/almanac/properties/a/appearance/>。不过要注意，`appearance` 属性并不是官方的 CSS，因此不同浏览器的行为有可能不一样。

提示 如果仔细查看图 16.8.5 中的代码，你会发现我为 form 添加了 ARIA 地标角色，`role="search"`。这会让屏幕阅读器指出网页上有搜索区域，从而提高页面的可访问性。如果你的表单有多个控件而非只有搜索，就将与搜索相关的控件放在一个 `fieldset` 或 `div` 中并为其添加 `role="search"`，而不是将 `role="search"` 添加到 form 元素本身。

提示 WebKit 浏览器支持两个非官方 HTML5 属性：`autosave` 和 `results`，它们为搜索框提供了额外的行为和视觉效果。更多信息参见 www.wufoo.com/html5/types/5-search.html。

提示 如果包含了 `pattern` 属性，一定要明确告诉访问者需要遵循的模式。稍不注意，访问者就有可能放弃提交表单。

提示 正则表达式超出了本书的范围，不过，网上有很多在线资源（搜索 `regex tutorial`）。<http://html5pattern.com> 上有很多有用的模式^①。

① 模式指的是用于描述一系列用来匹配规则的字符串（表达式）。——译者注

16.9 创建单选按钮

还记得老式汽车收音机上那些大大的黑色塑料按钮吗？按下其中的一个可以收听 WFCR；按下另一个可以收听 WRNX。不过，不可以同时按下两个按钮。单选按钮也遵循同样的工作方式。对 `input` 元素设置 `type="radio"` 即可创建单选按钮，参见图 16.9.1 和图 16.9.2。

```
...
<fieldset class="radios">
  <legend>Gender:</legend>

  <p class="row">
    <input type="radio"
      → id="gender-male" name="gender"
      → value="male" />
    <label for="gender-male">Male</label>
  </p>
  <p class="row">
    <input type="radio"
      → id="gender-female" name="gender"
      → value="female" />
    <label for="gender-female">Female
      → </label>
  </p>
</fieldset>
...
```

图 16.9.1 同一组单选按钮的 `name` 属性值必须相同，这样在同一时间只有其中一个能被选中。`value` 属性也很重要，因为对于单选按钮访问者无法输入值

创建单选按钮的步骤

- (1) 如果需要，输入单选按钮的介绍文本。例如，可以使用“选择下列选项中的一个”。
- (2) 输入 `<input type="radio">`。
- (3) 输入 `name="radioset"`，这里的 *`radioset`* 用于识别发送至服务器的数据，同时用于将多个单选按钮联系在一起，确保同一组中最多只有一个被选中。

```
.radios {
  background-color: transparent;
  position: relative;
  margin-bottom: 0;
}

.radios .row {
  margin: 0 0 0 150px;
}

.radios legend {
  left: 0;
  padding: 0 6px;
  position: absolute;
  text-align: right;
  top: 2px;
  width: 148px;
}

.radios label {
  padding-left: 2px;
  margin-right: 5px;
  vertical-align: middle;
  width: auto;
}
```

图 16.9.2 这里让 `legend` 在 `.radios` `div` 中绝对定位，`.radios` `div` 已设为 `position: relative`（对 `legend` 来说，`position` 比 `margin` 更有效）。为 `.row` 类添加一个大的 `margin-left` 值，让它们位于 `Gender` 的 `legend` 的右侧。对 `label` 元素设置的 `vertical-align: middle`；则让标签文本与单选按钮在竖直方向上对齐

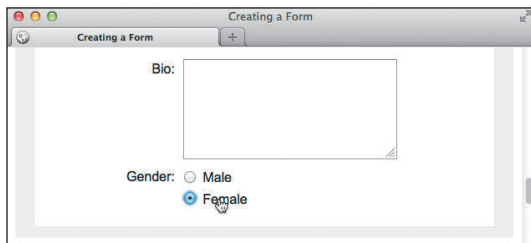


图 16.9.3 由于标签（`Male` 和 `Female`）是 `label` 元素，与标签交互则会选中它们对应的单选按钮

- (4) 输入 `id="idlabel"`，这里的 *`idlabel`* 对应你在第 (8) 步中创建的 `for` 属性值。跟 `name` 值不同（同一组单选按钮的 `name` 值都是

相同的)，同一页面中每个元素的 `id` 必须是唯一的。

(5) 输入 `value="data"`，这里的 *data* 是该单选按钮被选中（无论是被默认选中还是被访问者选中）时要发送给服务器的文本（如图 16.9.3 所示）。

(6) 如果需要，输入 `checked` 或者 `checked="checked"` 让该单选按钮在页面打开时默认处于激活状态（两种方法在 HTML5 中均可）。在一组单选按钮中，只能对一个按钮添加这一属性。

(7) 输入 `>` 或者 `/>`。

(8) 输入 `<label for="idlabel">radio label</label>`，其中，*idlabel* 与第 (4) 步中单选按钮的 `id` 值相同，*radio label* 则用于让访

问者识别该单选按钮。*radio label* 的值通常与 `value` 的值相同，但这并不是必需的。

(9) 对同一组内的所有单选按钮，重复第 (2) 步至第 (8) 步。

提示 我推荐使用 `fieldset` 嵌套每组单选按钮，并用 `legend` 进行描述（如图 16.9.1 所示）。更多细节参见 16.4 节。

16.10 创建复选框

在一组单选按钮中，只允许选择一个答案；但在一组复选按钮中，访问者可以选择任意数量的答案。同单选按钮一样，复选框也与 `name` 属性的值联系在一起，参见图 16.10.1。

```
<div class="fields checkboxes">
  <p class="row">
    <input type="checkbox" id="email-ok-msg-from-users" name="email_signup[]"
      → value="user-emails" />
    <label for="email-ok-msg-from-users">It is okay to email me with messages from other users.
      → </label>
  </p>
  <p class="row">
    <input type="checkbox" id="email-ok-occasional-updates" name="email_signup[]"
      → value="occasional-updates" />
    <label for="email-ok-occasional-updates">It is okay to email me with occasional promotions
      → about our other products.</label>
  </p>
</div>
```

图 16.10.1 注意，标签文本（未突出显示）不需要与 `value` 属性一致。这是因为标签文本用于在浏览器中向访问者标识复选框，而 `value` 则是发送到服务器端脚本的数据的一部分。空的方括号是为 PHP 脚本的 `name` 准备的（参见提示）。我创建了一个 `.checkboxes` 类，以限制为复选框添加样式的 `label`，参见图 16.10.2

```
.checkboxes label {
  text-align: left;
  width: 475px;
}
```

图 16.10.2 对于复选框，通常需要为标签设置不同的样式，因为这些标签位于输入元素之后，参见图 16.10.3

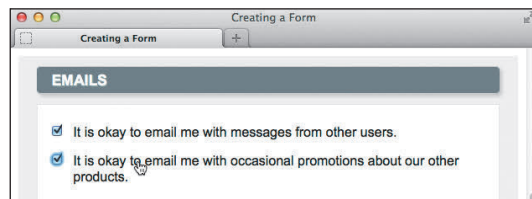


图 16.10.3 访问者可以根据需要选择任意数量的框，每个框对应的值及复选框组的名称都会被发送到脚本

创建复选框的步骤

(1) 如果需要，输入复选框的介绍文本。例如，可以使用“选择下列选项中的一个或多个”。

(2) 输入 `<input type="checkbox">`。

(3) 输入 `name="boxset"`，这里的 *boxset* 用于识别发送至服务器的数据，同时用于将多个复选框联系在一起（对于所有复选框使用同一个 `name` 值）。

(4) 输入 `id="idlabel"`，这里的 *idlabel* 对应于第 (8) 步中 `label` 的 `for` 属性值。

(5) 输入 `value="data"`，这里的 *data* 是该复选框被选中（无论是被访问者选中还是被建站者选中，如图 16.10.1 所示）时要发送给服务器的文本。

(6) 输入 `checked` 或 `checked="checked"` 让该复选框在页面打开时默认处于选中状态（在 HTML5 中两种写法均可），建站者或访问者可能会勾选默认的选项。

(7) 输入 `>` 或者 `/>` 以结束复选框。

(8) 输入 `<label for="idlabel"> checkbox label</label>`，其中，*idlabel* 与第 (4) 步复选框元素中的 `id` 值相同，*checkbox label* 则用于让访问者识别该复选框。

(9) 对同一组内的所有复选框，重复第 (2) 步至第 (8) 步。

提示 如果使用 PHP 处理表单，在图 16.10.1 中使用 `name="boxset[]"`（这里的 *boxset* 用于标识发送给脚本的数据）就会自动地创建一个包含复选框值的数组（名为 `$_POST['boxset']`）。

16.11 创建文本区域

如果希望给访问者填写问题或评论的空

间，可以使用文本区域。参见图 16.11.1。

```
<label for="bio">Bio:</label>
<textarea id="bio" name="bio" cols="40"
→ rows="5" class="field-large"></
textarea>
```

图 16.11.1 `rows` 和 `cols` 属性分别控制文本区域的高度和宽度，除非使用 CSS 覆盖其中之一（参见图 16.11.2）或者全部。即便在 CSS 中设置了尺寸，`rows` 和 `cols` 还是有用的，因为用户的浏览器有可能会关闭 CSS

```
textarea {
    font: inherit;
    padding: 2px;
}

.field-large {
    width: 250px;
}
```

图 16.11.2 默认情况下不会继承 `textarea`、`font` 属性，因此必须显式地设置该属性。如果需要，可以为与其他文本框、URL 框等输入框具有一个类（这里为 *.field-large*）的文本区域设置宽度。由于没有设置高度，因此文本区域的高度（如图 16.11.3 所示）由 HTML 中的 `rows` 属性决定（参见图 16.11.1）

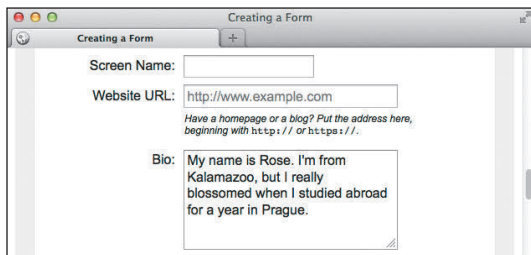


图 16.11.3 如果没有使用 `maxlength` 属性限制文本区域的最大字符数量，访问者可以输入多达 32 700 个字符。必要时，文本区域内会显示一个滚动条（这里没有显示）。访问者可以通过拖曳文本区域右下角的斜线改变文本区域的大小。如果设置了 `textarea { resize: none; }`，那么访问者就无法这样操作了

创建文本区域的步骤

(1) 如果需要, 输入用于标识文本区域的解释性文本。

(2) 输入 `<textarea>`。

(3) 输入 `id="idlabel"`, 这里的 `idlabel` 跟第 (1) 步中 `label` 的 `for` 属性值相同。

(4) 输入 `name="dataname"`, 这里的 `dataname` 是用于让服务器 (和脚本) 识别输入数据的文本。

(5) 如果需要, 输入 `maxlength="n"`, 这里的 `n` 是可以输入的最大字符数。

(6) 输入 `cols="n"`, 这里的 `n` 是文本区域的宽度 (以字符为单位)。

(7) 输入 `rows="n"`, 这里的 `n` 是文本区域的高度 (以行为单位)。

(8) 输入 `>`。

(9) 输入文本区域的默认文本 (如果有的话), 这些文本会显示在文本区域中。

(10) 输入 `</textarea>` 以完成文本区域。

提示 要为文本区域添加预设值, 就在 `textarea` 开始标签和结束标签之间添加文本 (这里没有 `value` 属性)。通常, 可以包含 `placeholder` 属性定义用于占位的文本。

提示 `maxlength` 是 HTML5 中为文本区域新增的属性, 因此它的行为在不同浏览器中可能不一样 (参见 www.wufoo.com/html5/attributes/03-maxlength.html)。旧浏览器会直接忽略该属性。

提示 通过 CSS 可以更好地控制文本区域的尺寸。

16.12 创建选择框

选择框非常适合向访问者提供一组选项, 从而允许他们从中选取, 参见图 16.12.1。它们通常呈现为下拉菜单的样式, 参见图 16.12.3。如果允许用户选择多个选项, 选择框就会呈现为一个带滚动条的项目框。

```
<label for="state">State:</label>
<select id="state" name="state">
  <option value="AL">Alabama</option>
  <option value="AK">Alaska</option>
  ...
</select>
```

图 16.12.1 选择框由两种 HTML 元素构成: `select` 和 `option`。通常, 在 `select` 元素里设置 `name` 属性, 在每个 `option` 元素里设置 `value` 属性。我们可以为 `select` (如图 16.12.2 所示) 和 `option` 元素添加样式, 但有一定的限制

```
select {
  font-size: inherit;
}
```

图 16.12.2 CSS 规则要求菜单文本跟其父元素字号大小相同, 否则默认情况下它看上去会小很多。可以使用 CSS 对 `width`、`color` 和其他的属性进行调整, 不过, 不同的浏览器呈现下拉菜单列表的方式略有差异

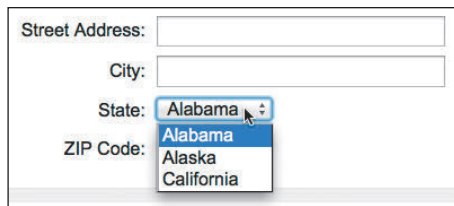


图 16.12.3 默认的选择是菜单中的第一个选项, 或者是在 HTML 中指定了 `selected` 的选项 (需要注意的一点是, 除非设置了 `size` 属性, 否则访问者就必须选择菜单中的某个选项)

1. 创建选择框的步骤

(1) 如果需要, 输入描述菜单的文本, 参见 16.6 节。

(2) 输入 `<select>`。

(3) 输入 `id="idlabel"`, 这里的 `idlabel` 跟第 (1) 步中 `label` 的 `for` 属性值一样。

(4) 输入 `name="dataname"`, 这里的 `dataname` 用于在收集的数据发送至服务器时对数据进行识别。

(5) 如果需要, 输入 `size="n"`, 这里的 `n` 代表选择框的高度 (以行为单位)。

(6) 如果需要, 输入 `multiple` 或者 `multiple="multiple"` (两种方法在 HTML5 中均可), 从而允许访问者选择一个以上的菜单选项 (选择的时候需按住 Control 键或 Command 键)。

(7) 输入 `>`。

(8) 输入 `<option>`。

(9) 输入 `value="optiondata"`, 这里的 `optiondata` 是选项选中后要发送给服务器的数据 (如果省略 `value`, 你在第 (12) 步中输入的文本就是选项的值)。

(10) 如果需要, 输入 `selected` 或者 `selected="selected"` (在 HTML5 中两种方式均可), 指定该选项被默认选中。

(11) 输入 `>`。

(12) 输入希望出现在菜单中的选项名称。

(13) 输入 `</option>`。

(14) 对每个选项重复第 (8) 步至第 (13) 步。

(15) 输入 `</select>`。

如果有一个选项很多的特别大的菜单, 可能需要对这些选项进行分组, 分别放入不同的类别, 参见图 16.12.4 和图 16.12.5。

2. 对选择框选项进行分组

(1) 根据“创建选择框的步骤”, 创建所需的选项。

(2) 在希望放在同一子菜单中的第一组选项中的第一个 `option` 元素 (参见前面的第 (8) 步) 之前, 输入 `<optgroup>`。

(3) 输入 `label="submenuitle">`, 这里的 `submenuitle` 是子菜单的标题。

(4) 在该组的最后一个 `option` 元素之后, 输入 `</optgroup>`。

(5) 对每个子菜单重复第 (2) 步至第 (4) 步。

```
<label for="referral">Where did you find
→ about us?</label>
<select id="referral" name="referral">
  <optgroup label="Online">
    <option value="social_network">Social
    → Network</option>
    <option value="search_engine">Search
    → Engine</option>
  </optgroup>
  <optgroup label="Offline">
    <option value="postcard">Postcard
    → </option>
    <option value="word_of_mouth">Word of
    → Mouth</option>
  </optgroup>
</select>
```

图 16.12.4 每个子菜单都有一个标题 (在 `optgroup` 开始标签的 `label` 属性中指定) 和一系列选项 (使用 `option` 元素和常规文本定义)

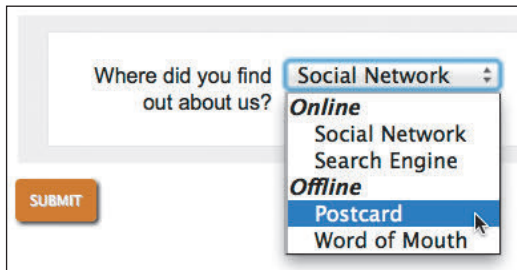


图 16.12.5 浏览器通常会对 `optgroup` 中的 `option` 缩进, 从而将它们和 `optgroup label` 属性文本区别开

提示 如果添加了 `size` 属性, 那么选择框看起来会更像一个列表, 且没有自动选中的选项, 参见图 16.12.6 (除非设置了 `selected`)。

提示 如果 `size` 大于选项的数量，访问者就可以通过点击空白区域让所有的选项处于未选中状态。

提示 可以对 `option` 元素添加 `label` 属性，该属性用于指定需要显示在菜单中的文本（替代了 `option` 标签之间的文本），参见 16.10 节。不过，Firefox 并不支持这一属性，因此最好不要用它。

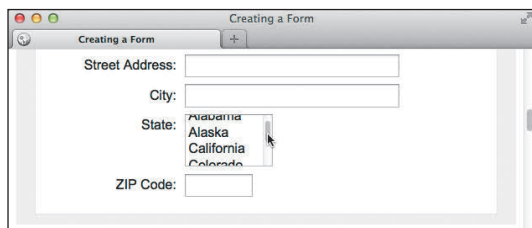


图 16.12.6 由于设置了 `size` 属性，菜单显示为一个有滚动条的列表，默认情况下没有选中任何选项。这个示例的代码为 `<select id="state" name="state" size="3">`，这让菜单的高度为三行

16.13 让访问者上传文件

有时需要让网站的用户向服务器上传文件（如照片、简历等），参见图 16.13.1。

```
<form method="post" action="show-data.php"
  enctype="multipart/form-data">
  ...
  <label for="picture">Picture:</label>
  <input type="file" id="picture"
    → name="picture" />
  <p class="instructions">Maximum size of
    → 700k. JPG, GIF or PNG.</p>
  ...
</form>
```

图 16.13.1 要让访问者能够上传文件，必须正确地设置 `enctype` 属性，创建 `input type="file"` 元素。对 `input` 使用 `multiple` 属性可以允许上传多个文件（这里并没有包含该属性）。这是 HTML5 中新增的内容，它也得到了浏览器的广泛支持，不过，移动端浏览器和 IE 会直接忽略它（仅 IE10 支持）

处理文件上传

处理文件上传需要一些特殊的代码。可以在网上搜索“file upload script”（文件上传脚本）查看相关的资源。同时，服务器需要配置正确才能存储文件。如需帮助，可以联系网站托管商。

让访问者上传文件的步骤

(1) 输入 `<form method="post" enctype="multipart/form-data">`。 `enctype` 属性可以确保文件采用正确的格式上传。

(2) 接下来，输入 `action="upload.url">`，其中的 `upload.url` 是处理上传文件的脚本的 URL。

(3) 为文件上传区域输入标签，以便访问者知道上传什么文件（参见 16.6 节）。

(4) 输入 `<input type="file"`，创建一个文件上传框和一个 Browse（浏览）按钮，如图 16.13.2 所示。

(5) 输入 `id="idlabel"`，这里的 `idlabel` 跟第 (3) 步中 `label` 的 `for` 属性值相同。

(6) 输入 `name="dataname"`，这里的 `dataname` 用于识别将要上传的文件。

(7) 如果需要，输入 `size="n"`，这里的 `n` 是访问者可以输入路径和文件名的字段的宽度。

(8) 如果需要，输入 `multiple` 或者 `multiple="multiple"`（HTML5 中这两种方式均可），这表示访问者可以上传一个以上的文件，参见图 16.13.1 的标题说明。

(9) 输入 `>` 或者 `/>`。

(10) 跟平常一样结束表单，包括创建提交按钮，输入 `</form>` 结束标签。

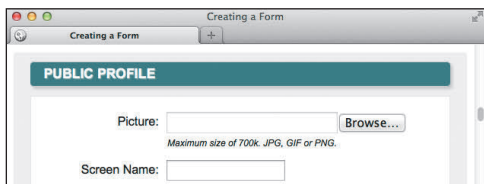


图 16.13.2 文件上传区域为用户提供了从其系统中选择文件的方式。对于 `type="file"` 的 `input` 元素，浏览器会自动创建 Browse（浏览）按钮。Chrome 和 Safari 不会创建框，它们只显示按钮。浏览器通常不允许像对其他表单元素那样对此类 `input` 设置样式

提示 对于允许上传的表单，不能使用 `get` 方法。

16.14 创建隐藏字段

隐藏字段可以用于存储表单中的数据，但它不会显示给访问者（参见图 16.14.1 和图 16.14.2）。可以认为它们是不可见的文本框。它们通常用于存储先前的表单收集的信息，以便将这些信息同当前表单的数据一起交给脚本进行处理，如图 16.14.2 所示。

```
<form method="post" action="your-script.php">
  <input type="hidden" name="step"
    → value="6" />

  ... 其他表单字段 ...

  <input type="submit"
    → value="Submit Form" />
</form>
```

图 16.14.1 访问者不会看到这个输入框，但他们提交表单的时候，名“step”和值“6”会随着表单中从访问者输入获取的数据一起传送给服务器

创建隐藏字段的步骤

- (1) 输入 `<input type="hidden">`。
- (2) 输入 `name="dataname"`，这里的 `dataname`

确定要提交给服务器的信息。

(3) 输入 `value="data"`，这里的 `data` 是要提交的信息本身。它通常是表单处理脚本中的一个变量（参见图 16.14.1）。

(4) 输入 `>` 或者 `/>`（在 HTML5 中两种方式均可）。

```
<form method="post" action="your-script.php">
  <input type="hidden" name="email"
    → value="<?= $email ?>" />

  ... 其他表单字段 ...

  <input type="submit"
    → value="Submit Form" />
</form>
```

图 16.14.2 创建隐藏字段时，可以使用脚本中的变量将字段的值设置为访问者原来输入的值（这个示例使用了 PHP 语法）

什么时候使用隐藏字段

这里有个隐藏字段非常方便的例子。假设你有一个表单，希望让访问者在提交表单之前有机会检查他们输入的内容。处理表单的脚本可以向访问者显示提交的数据，同时创建一个表单，其中有包含同样数据的隐藏字段。如果访问者希望编辑数据，他们只需后退就可以了。如果他们想提交表单，由于隐藏字段已经将数据填好了，因此他们就不需要再次输入数据了。

提示 隐藏字段出现在表单标记中的位置并不重要，因为它们在浏览器中是不可见的。

提示 不要将密码、信用卡号等敏感信息放到隐藏字段中。即便它们不会显示到网页中，访问者也可以通过查看 HTML 源代码看到它们（参见 2.8 节）。

提示 要创建访问者可见但不可修改的表元素，有两种方法。一种是使用 `disabled`（禁用）属性（参见 16.16 节）。另一种是使用 `readonly`（只读）属性。与禁用字段不同，只读字段可以获得焦点，访问者可以选择和复制里面的文本，但不能修改这些文本。它只能应用于文本输入框和文本区域，例如，`<input type="text" id="coupon" name="coupon" value="FREE" readonly />`。还可以使用 `readonly="readonly"` 这样的形式，结果是一样的。

16.15 创建提交按钮

访问者输入的信息如果不发送到服务器，就没什么用。应该总是为表单创建提交按钮，让访问者可以将信息交给你。提交按钮可能呈现为文本（参见图 16.15.1 ~ 图 16.15.3）、可能是图像（参见图 16.15.4 和图 16.15.5），也可能是两者的结合（参见图 16.15.6 和图 16.15.7）。

```
<input type="submit" value="Create Profile"
→ class="btn" />
```

图 16.15.1 如果不填写 `name` 属性，则提交按钮的名-值对就不会传给脚本。由于通常不需要这一信息，因此这里的做法是有益的

```
.btn {
  background-color: #da820a;
  border: none;
  border-radius: 4px;
  box-shadow: 2px 2px 2px #333;
  color: #fff;
  margin: 12px 0 0 26px;
  padding: 8px;
  text-shadow: 1px 1px 0px #777;
}
```

图 16.15.2 通过使用类，我为提交按钮应用了背景、字体格式和一些 CSS3 特性。因为是类，所以可以将该样式重用到其他按钮上

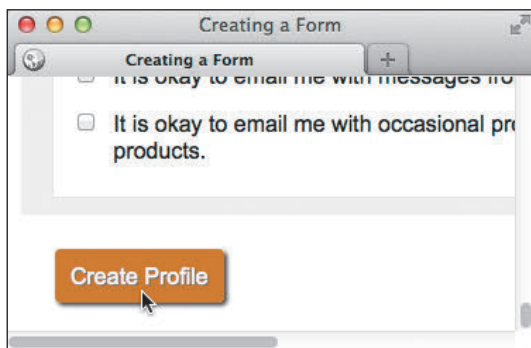


图 16.15.3 这里将按钮的 `value` 设为 `Create Profile`，比起默认值，这样的值对访问者来说含义更丰富（参见第一条提示）。如果激活提交按钮，就会将表单数据发给服务器上的脚本，从而可以利用这些信息

1. 创建提交按钮的步骤

- (1) 输入 `<input type="submit">`。
- (2) 如果需要，输入 `value="submit message"`，这里的 `submit message` 是将要出现在按钮上的文本。
- (3) 最后输入 `>` 或者 `/>`。

2. 创建带图像的提交按钮

有时设计师会创建一些超出 CSS3 能力（如渐变、阴影、圆角等）的按钮样式。这时，可以使用仅包含图像的 `input` 元素来提交表单（参见图 16.15.4 和图 16.15.5）。

- (1) 创建 PNG、GIF 或者 JPEG 图像，通常最好是 PNG，因为文件尺寸小。
- (2) 输入 `<input type="image">`。
- (3) 输入 `src="image.url"`，其中的 `image.url` 是图像在服务器上的位置。
- (4) 输入 `alt="description"`，这里的 `description` 是当图像无法显示时需要出现的文本。
- (5) 输入 `>` 或者 `/>` 以结束图像提交按钮。

```
<input type="image" src="button-submit.png"
→ width="188" height="95" alt="Create Profile"
→ />
```

图 16.15.4 利用 `type="image"` 可以创建图像提交按钮, `width` 和 `height` 属性是可选的

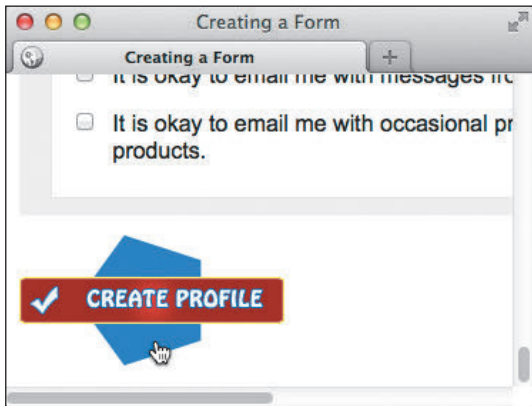


图 16.15.5 当用户将鼠标悬停在图像提交按钮上时, 浏览器会将箭头替换为手形图标

3. 创建结合文本和图像的提交按钮

使用 `button` 元素可以创建包含其他 HTML 元素 (而并非仅包含简单的文本值或图像) 的按钮, 参见图 16.15.6 和图 16.15.7。(如果要使用 `button` 元素, 注意考虑 IE8 之前的 IE 的兼容性问题。更多细节可以在网上搜到。)

(1) 输入 `<button type="submit">`。

(2) 输入将要出现在按钮中图像左侧的文本 (如果有的话)。

(3) 输入 `"image.url"`, 这里的 `image.url` 是出现在按钮上的图像名称。

(4) 输入 `alt="alternate text"`, 这里的 `alternate text` 是当图像无法显示时需要出现的文本。

(5) 如果需要, 添加其他图像属性。

(6) 输入 `>` 或者 `/>` 以结束图像。

(7) 输入将要出现在按钮中图像右侧的文本 (如果有的话)。

(8) 输入 `</button>`。

```
<button type="submit" class="btn">
→  Create Profile</button>
```

图 16.15.6 `button` 元素可以为提交按钮的内容提供更大的灵活性。这个元素既包含图像, 也包含文本。`button` 元素还可以包含其他的 HTML 元素

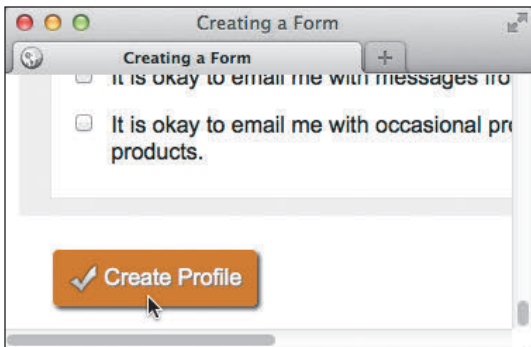


图 16.15.7 除去那个对勾的图像, 这个按钮与图 16.15.3 中的按钮很像, 因为它们都有 `.btn` 类 (参见图 16.15.2)。当鼠标移到这个按钮上时, 指针会显示为一个箭头 (参见图 16.15.3)

提示 如果省略 `value` 属性, 那么根据不同的浏览器, 提交按钮就会显示默认的 `Submit` 或者 `Submit Query`。

提示 如果有多个提交按钮, 可以为每个按钮设置 `name` 属性和 `value` 属性, 从而让脚本知道用户按下的是哪个按钮。否则, 最好省略 `name` 属性。

提示 可以使用 `button` 元素创建不包含图像的提交按钮。任何时候, 如果你的表单需要一个以上的提交按钮, 就应避免使用 `button` 元素, 因为针对这种情况, 不同浏览器的行为并不完全一致。

提示 表单还可以有重置按钮，用于将表单数据还原为页面加载时（在访问者填写表单之前）的样子。创建重置按钮可以使用 `<input type="reset" />` 或 `<button type="reset">Reset</button>`。我们也可以为重置按钮添加样式。

提示 HTML5 对 `type="email"` 和 `type="URL"` 的 `input` 添加了自动验证功能。对提交按钮使用 `formnovalidate` 属性可以关闭该功能，如 `<input type="submit" formnovalidate />`。

16.16 禁用表单元素

在某些情况下，你可能不想让访问者使用表单中的某些部分。例如，你可能希望在所有必填字段完成之前禁用提交按钮，如图 16.16.1 所示。

禁用表单元素的值不会发送到服务器，如果使用键盘在页面中导航，它也会被跳过。

```
...
<div id="choices">
  <p>
    <input type="radio" name="how" value="advertisement" id="advertisement" />
    <label for="advertisement">Advertisement</label>
  </p>
  ...
  <p>
    <input type="radio" name="how" value="other" id="other" />
    <label for="other">Other</label>
  </p>
  <p>
    <textarea id="other-description" cols="35" rows="5" placeholder="TV, school, bingo game,
    → etc." title="Please describe how you heard about us." disabled="disabled"></textarea>
  </p>

  <input type="submit" value="Submit" class="btn" />
</div>
...
</form>

<!-- 这块要放到最后，刚好在</body>之前 -->
<script src="js/toggle-textarea.js"></script>
</body>
</html>
```

16

图 16.16.1 这里对 `textarea` 元素添加了 `disabled` 属性，并在页面最底部引入了一个 JavaScript 文件。该文件包含了一段脚本，其作用是当用户选择 Other（其他）单选按钮时，让 `textarea` 变为可用的（如图 16.16.2 和图 16.16.3 所示）。选择其余两个单选按钮中的任意一个，则会禁用 `textarea`（注意，在实践中，我会使用 JavaScript 添加 `disabled` 属性，而不是像这样写在 HTML 中。这样，禁用了 JavaScript 的访问者仍可填写该文本区域。第 19 章的 `toggle-textarea.js` 就实现了这样的功能）

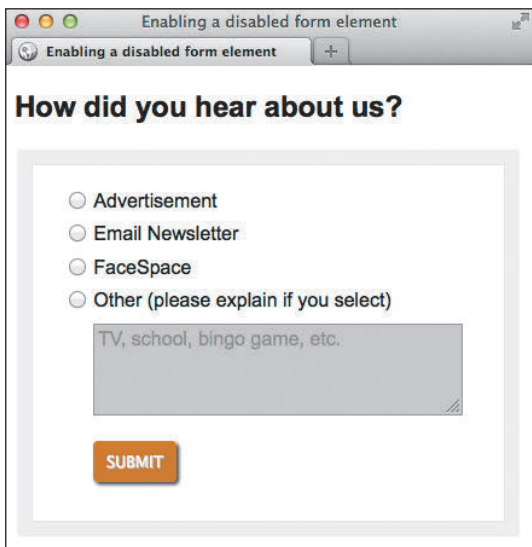


图 16.16.2 当 Other 单选按钮未选中时，文本区域是灰色的、被禁用的，用户无法选中该框并输入内容

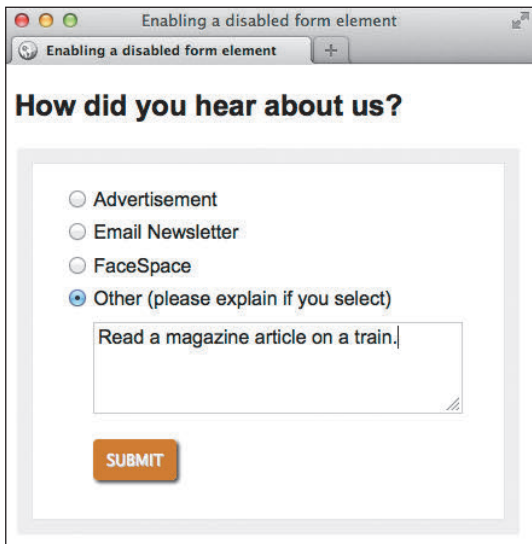


图 16.16.3 由于使用了 JavaScript，当访问者选择 Other 单选按钮以后，文本区域变成了白色，用户可以输入文本以提交到服务器

禁用表单元素的方法

在表单元素的开始标签后输入 `disabled` 或者 `disabled="disabled"`（两种方法在 HTML5 中均可以）。

提示 可以使用 JavaScript 将表单元素的状态由禁用改为可用（也可以反过来）。对 JavaScript 的讲解超出了本书的范围，但我在第 19 章提供了图 16.16.1 引用的 `toggle-textarea.js` 文件（参见 19.1 节）。该文件中有一些用于解释其工作原理的注释。要学习 JavaScript，eloquentjavascript.net 是个不错的资源。

提示 更多关于 `script` 元素的信息参见第 19 章。

提示 要理解 `disabled` 和 `readonly` 属性的区别，参见 16.14 节最后一条提示。

16.17 根据状态为表单设置样式

有时，你可能需要根据表单的状态或是否必须包含某个属性的情况设置不同的样式。例如，你可能需要将必填字段与其他字段在样式上区分开。

你可能会想起本书其他地方（如 9.7 节）介绍过的伪类。不过，CSS 提供了其他一些根据表单元素状态设置其样式的伪类。其中的大多数都是 CSS3 新增的。我总结了其中得到浏览器广泛支持的一些，如表 16.17.1 所示。其他的参见 Peter-Paul Koch 的列表——打开 www.quirksmode.org/css/selectors/ 并拖到 UI state pseudo-classes（UI 状态伪类）部分。

表 16.17.1 伪类

选 择 器	应 用	浏览器支持情况
:focus	获得焦点的字段（参见图 16.17.1 和图 16.17.2）	IE8+ 及其他
:checked	选中的单选按钮或复选框（参见图 16.17.3 和图 16.17.4）	IE9+ 及其他
:disabled	具有 disabled 属性的字段（参见图 16.17.5）	IE9+ 及其他
:enable	与 :disabled 相反	IE9+ 及其他
:required	具有 required 属性的字段（参见图 16.17.6 和图 16.17.7）	IE10+、Safari 5+ 及其他
:optional	与 :required 相反	IE10+、Safari 5+ 及其他
:invalid	其值与 pattern 属性给出的模式不匹配的字段；或值不是有效电子邮件格式的电子邮件框，值不是有效 URL 格式的 URL 框，以及任何标记为 required 但值为空的字段（参见提示），如图 16.17.8 和图 16.17.9 所示	IE10+、Safari 5+ 及其他
:valid	与 :invalid 相反	IE10+、Safari 5+ 及其他

注意：“IE9+ 及其他”指的是 Internet Explorer 9 及之后的版本支持该选择器，其他现代浏览器也支持该选择器。其他的说明也是类似的

```
input:focus,
textarea:focus {
    background-color: greenyellow;
}
```

图 16.17.1 这条规则为任意获得焦点的 input（包括提交按钮）或 textarea 添加了背景色。要定位特定的 input 类型，可以包含一个属性选择器，如 input[type="submit"]:focus { backgroundcolor: #ff8c00; } 就仅为获得焦点的提交按钮设置样式

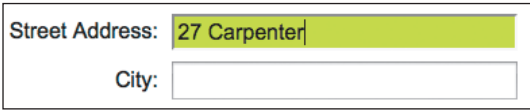


图 16.17.2 第一个字段获得了焦点，因此它的背景是浅绿色的（虽然并不是最舒服的颜色，但能说明问题）

```
input:checked + label {
    color: green;
}
```

图 16.17.3 这条规则为选中的单选按钮或复选框（或指定了 checked 属性的单选按钮或复选框）添加样式，效果如图 16.17.4 所示

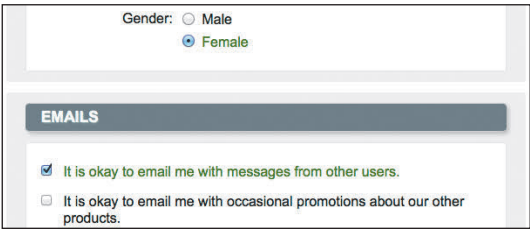


图 16.17.4 只有选中的单选按钮和复选框是绿色的

```
textarea:disabled {
    background-color: #ccc;
    border-color: #999;
    color: #666;
}
```

图 16.17.5 这是用于图 16.16.2 中代码的样式规则

```
input:required,
textarea:required {
    border: 2px solid #000;
}
```

图 16.17.6 所有必填的 input 和 textarea 元素会有一个更醒目的边框

针对特定状态设置表单元素样式的步骤
(1) 输入 selector，这里的 selector 包含

表 16.17.1 中显示的一种或多种状态，接着输入 { 开始样式规则的声明。

(2) 输入你想为该状态设置的任何 `property: value;` 声明。关于 CSS 属性的讲解是从第 8 章开始的。

(3) 输入 } 结束样式规则的声明。

图 16.17.7 如你所见，只有 First Name（名）字段是必填的。红色的星号并不是图 16.17.6 中的 CSS 产生的效果。对于必填的字段，在其标签中添加星号或“必填”字样是一种常见的做法，因此这里添加了星号。这样，不支持 `:required` 的浏览器也会表现出这是一个必填字段的樣子

```
input[type="email"]:invalid {
    color: red;
}

input[type="email"]:valid {
    color: black;
}
```

图 16.17.8 电子邮件框中的值如果不是有效的电子邮件地址，该框中的字就会变成红色的

提示 页面一开始加载就会应用 `:invalid` 状态，因此，根据你所设置的状态，有可能会出现一些意外的结果。例如，如果规则是 `input:invalid { background-color: pink; }`，那么必填的 `input` 字段在用户填写任何内容之前就会显示粉红色的背景。（任何标记为 `required` 但值为空的字段会被认为处于无效状态。）为了将必填字段排除在该规则之外，可以使用 `:not` 伪类，如 `input:invalid:not(:required) { border: 2px solid red; }`。除了 IE9 之前的版本，所有浏览器都支持 `:not` 伪类。

图 16.17.9 输入的内容都是红色的，直到输完一个有效的电子邮件地址（此时输入的内容会变成黑色的）

提示 上一条提示讲到的情境还有一种应对方法，即提交表单时，使用 JavaScript 为 `form` 元素添加一个类，并在样式表中为该创建无效样式，例如，`.submitted input:invalid { background-color: red; }`。示例代码及其他提示参见 Peter Gasston 的文章（位于 html5doctor.com/css3-pseudo-classes-and-html5-forms）。

提示 对于浏览器无法识别的伪类，浏览器会直接忽略其中的样式规则。对于 Web 设计人员和开发人员来说，这通常是可以接受的——使用功能较强的浏览器的访问者获得增强的体验。Keith Clark 的 Selectivizr（<http://selectivizr.com/>）是一个用于让旧浏览器理解这些选择器的 JavaScript 文件。

使用属性选择器为表单设置样式

别忘了，还可以使用属性选择器定位拥有特定属性的表单字段，例如：

- `[autocomplete]`
- `[autofocus]`
- `[multiple]`（仅限于电子邮件框和文件上传框）
- `[placeholder]`
- `[type="email"]`（参见图 16.17.8）、`[type="url"]` 等（其他的输入框类型都可以像这样使用）

本章内容

- 第三方插件与步入原生
- 视频文件格式
- 在网页中添加单个视频
- 为视频添加控件和自动播放
- 为视频指定循环播放和海报图像
- 阻止预加载视频
- 使用多种来源的视频和备用文本
- 提供可访问性
- 音频文件格式
- 为网页添加带控件的音频文件
- 自动播放、循环和预加载音频
- 提供带有备用设置的多种音频来源
- 添加具有备用 Flash 的视频和音频
- 高级多媒体
- 更多资源

在页面上添加电影、声音、图像和动画可以增强访问者的体验。在 HTML5 出现之前，为网页添加多媒体的唯一办法就是使用第三方的插件（如 Adobe Flash Player 和苹果的 QuickTime）。通过引入原生的多媒体（浏览器负责一切），HTML5 改变了这一状况。

并非所有支持 HTML5 的浏览器都支持相同格式的视频和音频格式。你将在本章学到如何提供多种格式（以及为不支持任何一种格式的浏览器提供备用 Flash），让不同的访问者都能正常使用。

注意，本章的目的是介绍多媒体 Web 文件，重点放在了所需的 HTML5 代码上。这里不会讲解如何创建多媒体内容，只会讲解如何让访问者可以浏览它们。

17.1 第三方插件和步入原生

如上文所述，在 HTML5 出现之前，可以通过第三方插件为网页添加音频和视频，但这样做有一些问题。在某个浏览器中嵌入 Flash 视频的代码在另一个浏览器中可能不起作用，也没有包围它的优雅方式。而且，这样做会影响用户访问网站的体验，因为像 Flash 这样的插件会占用大量的计算资源。有时，浏览器会变慢，甚至崩溃。

考虑到这些问题，HTML5 规范中添加了原生的多媒体。这样做有很多好处：速度更快（任何浏览器原生的功能势必比插件要快一些），媒体播放按钮和其他控件内置到浏览器，对插件的依赖极大地降低（但也并非完全不依赖，这在后面将会看到）。

对于任何一套标准，都有关于 HTML5 原生多媒体及其支持的文件格式的问题。最初，HTML5 规范指定了两种兼容 HTML5 的浏览器必须支持的媒体格式（分别对应音频和视频）。这本来是很好的，但并非所有的厂商都愿意遵循。这意味着你需要为你的媒体提供一种以上的格式才能让它在兼容 HTML5 的浏览

器中播放。后面我们将详细地讨论这些问题。

当苹果宣布不再在其移动设备（包括 iPhone 和 iPad）上支持 Flash 时，HTML5 及原生媒体就变得更为实用了。随着这些设备的日渐普及，过去播放媒体文件对 Flash 的依赖正在迅速削减，因此迫切需要一种新的解决方案。于是，HTML5 原生多媒体进入了这一领域并展现出了强大的力量，因为苹果移动设备的浏览器是支持 HTML5 的。其他移动设备也迅速跟进。

数字版权管理（DRM）

在嵌入音频和视频文件的过程中，你一定注意到了这样一个事实——所有人都看得见指向源文件的 URL，从而利用它下载并“盗取”你的内容，就像嵌入的图像以及 HTML、JavaScript 和 CSS 源文件一样。对此我们毫无办法。

尽管已经有了一些相关的讨论，但 HTML5 并没有提供任何保护媒体内容的方法。因此，如果你很在意对媒体文件的保护，那么暂时不要使用 HTML5 原生多媒体。

17.2 视频文件格式

HTML5 支持三种视频文件格式（即编解码器）。

- Ogg Theora 使用的文件扩展名为 .ogg 或 .ogv，支持它的浏览器包括 Firefox 3.5+、Chrome 4+、Opera 10.5+ 以及 Android 版 Firefox。
- MP4（H.264）使用的文件扩展名为 .mp4 或 .m4v，支持它的浏览器包括 Safari 3.2+、Chrome 4+（参见提示）、Internet Explorer 9+、iOS（Mobile Safari）和 Android 2.1+、Android 版 Chrome、Android 版 Firefox 和 Opera

Mobile 11+。

- WebM 使用的文件扩展名为 .webm，支持它的浏览器包括 Firefox 4+、Chrome 6+、Opera 10.6+、Android 2.3+、Android 版 Chrome、Android 版 Firefox 和 Opera Mobile 14。

什么是编解码器

编解码器是使用压缩算法对数据的数字流进行编码和解码，使之更适合播放的计算机程序。

编解码器的目标通常是在保证音频和视频所能达到的最高质量的情况下减小文件尺寸。

当然，不同编解码器的表现是不一致的。

设置 MIME 类型

在有的浏览器中，如果没有设置正确的 MIME 类型，媒体文件就不会播放。如果你的网站运行在 Apache Web 服务器上（很可能是它），可以通过 .htaccess 文件设置 MIME 类型。它是一个文本文件，通常与主页一起位于网站的根目录。

这是可以在 .htaccess 文件中添加的 MIME 类型（可以使用文本编辑器编辑该文件）。

```
AddType video/ogg .ogg
AddType video/mp4 .mp4
AddType video/webm .webm
AddType audio/ogg .ogg
AddType audio/mp3 .mp3
```

如果你的网站已经拥有该文件，将其命名为 a.htaccess，从 Web 服务器上下载下来，添加上述 MIME 类型，再上传到服务器上，重新命名为 .htaccess。如果你的网站没有该文件，可以重新创建一个。

对于更新 .htaccess，如有问题，请联系 Web 托管商。

转换文件格式

如果你已经拥有一个视频资源并希望将它转换为其他的文件格式，有大量的工具可以帮你完成这一任务。下面就是两个这样的工具：

- ❑ Miro Video Converter (www.mirovideo-converter.com)
- ❑ HandBrake (<http://handbrake.fr>)

提示 开发者至少需要为视频提供两种格式（MP4 和 WebM），才能确保获得所有兼容 HTML5 的浏览器的支持。

提示 Google 曾经说过在 Chrome 中放弃对 MP4 的支持，不过他们没这么做。Firefox 正在逐步提供对 MP4 的支持。在桌面端，Windows 7+ 首次提供了对它的支持，不过需要用用户在机器上安装相应的编解码器。

提示 如果用户的机器上安装了 WebM，那么 WebM 在 IE 9+ 或 Safari 中都可以正常运行。

17.3 在网页中添加单个视频

要在 HTML5 网页中添加视频，需要使用新的 `video` 元素。这一过程再简单不过了，如图 17.3.1 所示。浏览器会计算视频的尺寸，并在加载文件后以计算的尺寸显示视频，参见图 17.3.2。开发人员也可以自行设定视频尺寸，如图 17.3.3 所示。

如果浏览器不能识别你指定的视频格式，就无法显示该视频，参见图 17.3.4。

```
...
<body>
  <video src="paddle-steamer.webm"></video>
</body>
</html>
```

图 17.3.1 指定单个 WebM 视频（不含控件）

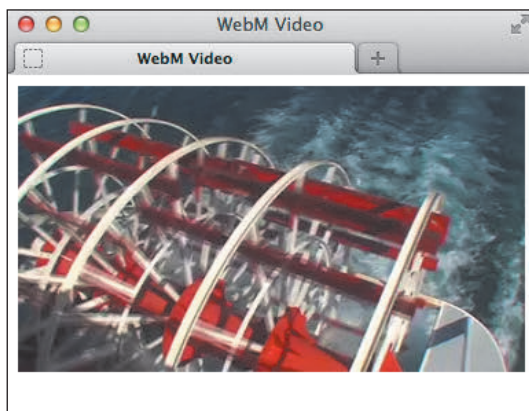


图 17.3.2 默认情况下，视频会暂停在第一帧，目前我们得到的视频没有播放按钮，也就是不能观看！下一节会纠正这个问题

```
...
<body>
  <video src="paddle-steamer.webm"
    → width="369" height="208"></video>
</body>
</html>
```

图 17.3.3 这里为视频设置的尺寸符合其正常大小，因此其效果与图 17.3.2 相同。不过，同图像一样，可以使用其他大小的 `width` 和 `height` 值，浏览器会对视频进行缩放，并尽可能地保证质量

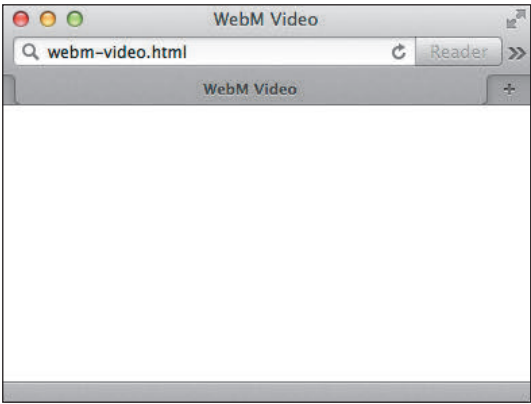


图 17.3.4 像 Safari 这样不支持 WebM 的浏览器会显示一片空白。这并不好！后面会介绍如何指定多种视频格式，从而支持各种的浏览器

1. 在网页中添加单个视频的步骤

(1) 获取视频资源。

(2) 输入 `<video src="my-video.ext"></video>`，这里的 *my-video.ext* 是视频文件的位置、名称和扩展名。就这么简单（参见图 17.3.2 和图 17.3.4）！

2. video 属性

除了 `src`，还有哪些属性可以用在 `video` 元素上呢？表 17.3.1 列出了这些属性。其中的某些属性为视频提供了很高的灵活性。

表 17.3.1 video 属性

属 性	描 述
<code>src</code> （源）	指定视频文件的 URL
<code>autoplay</code> （自动播放）	当视频可以播放时立即开始播放
<code>controls</code> （控件）	添加浏览器为视频设置的默认控件
<code>muted</code> （静音）	让视频静音
<code>loop</code> （循环）	让视频循环播放
<code>poster</code> （海报）	指定视频加载时要显示的图像（而不显示视频的第一帧）。接受所需图像文件的 URL
<code>width</code> （宽度）	视频的宽度（以像素为单位），通常默认为 300
<code>height</code> （高度）	视频的高度（以像素为单位），通常默认为 150
<code>preload</code> （预加载）	告诉浏览器要加载的视频内容的多少。可以是以下三个值： ☐ <code>none</code> 表示不加载任何视频 ☐ <code>metadata</code> 表示仅加载视频的元数据（如长度、尺寸等） ☐ <code>auto</code> 表示让浏览器决定怎样做（这是默认的设置）

17.4 为视频添加控件和自动播放

目前仅向你展示了在网页中添加视频的最简单的方法，而示例中的视频甚至不会自动开始播放，因为我们没有让它这样做。此外，访问者也不能开始播放视频，因为播放器没有显示任何控件。

很容易就可以通过图 17.4.1 中的代码改变这种情况。`controls` 属性会告诉浏览器添

加一套用于控制视频播放的控件。每个浏览器都有自己默认的控件，它们看起来都不一样（参见图 17.4.2 ~ 图 17.4.6）。

通常，只有用户点击播放按钮后，视频才会开始播放。可以添加 `autoplay` 属性，让视频自动播放（如图 17.4.7 所示）。

```
...
<body>
  <video src="paddle-steamer.webm"
    → width="369" height="208" controls>
    → </video>
</body>
</html>
```

图 17.4.1 添加单个 WebM 视频（这次是包含控件的）



图 17.4.2 Firefox 中的视频控件。从这里可以看到 Firefox 中的视频比其他浏览器中的视频长 1 秒

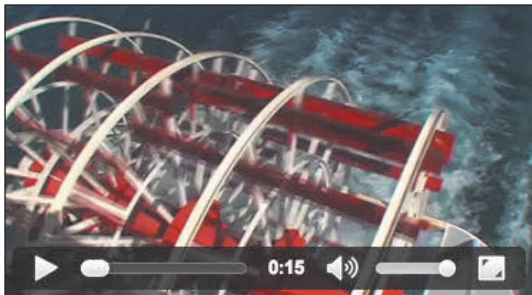


图 17.4.3 Chrome 中的视频控件



图 17.4.4 Internet Explorer 10 中的视频控件



图 17.4.5 Opera 中的视频控件



图 17.4.6 Safari 中的视频控件

```
...
<body>
  <video src="paddle-steamer.webm"
    → width="369" height="208" autoplay
    → controls></video>
</body>
</html>
```

图 17.4.7 现在，video 元素有了表 17.3.1 中的三个属性。由于有 autoplay，因此视频会自动播放。由于有 controls，访问者有了暂停按钮。这些属性出现的顺序无关紧要

1. 为视频添加控件

输入 `<video src="my-video.ext" controls>`
`</video>`，其中 `my-video.ext` 指向你的视频文件。

2. 为视频添加自动播放

输入 `<video src="my-video.ext" autoplay controls>``</video>`，这里的 `my-video.ext` 指向你的视频文件。

如果浏览器不支持某种视频格式

如果查看代码示例（参见图 17.4.1 和图 17.4.7）的浏览器不支持你使用的视频文件格式，它会显示视频的控制条，或者显示一个空的白色矩形区域（大多数情况下是这样），或者显示海报图像（前提是通过 `poster` 属性指定了海报）。

例如，Internet Explorer 和 Safari 不支持 WebM，但支持 MP4。为了在图 17.4.4 和图 17.4.6 中演示它们的播放控件，我创建了同时使用 MP4 和 WebM 格式的页面。关于实现方法，参见 17.7 节。

如果视频没有通过 `width` 和 `height` 指定尺寸，空的矩形通常为 300 像素 × 150 像素大小。如果包含了 `controls`，但使用了一种 IE10 不支持的格式，它会显示一个空白的矩形，以及 Invalid Source(无效来源)的字样。

布尔类型的属性

回想第 1 章，你应该能记得，布尔类型的属性（如 `controls` 和 `autoplay`）不需要指定值，因为它们是在几个值之间进行选择。无论采取哪种方式，它们出现在 `video` 和 `audio` 元素中产生的效果是一样的。

本书的示例都没有为这些布尔类型的属性指定值，不过，图 17.4.7 中的代码也可以写作：

```
<video src="paddle-steamer.webm"
controls="controls" autoplay="autoplay">
</video>.
```

17.5 为视频指定循环播放和海报图像

不仅可以将视频设为自动播放，还可以将它设为持续播放，直到停止，如图 17.5.1 所示。（不过这一做法并不推荐，考虑一下用户的感受就知道原因了。）要实现循环播放，只需要使用 `autoplay` 和 `loop` 属性。

如果不设置 `autoplay` 属性，通常浏览器会在视频加载时显示视频的第一帧。你可能想对此作出修改，指定你自己的图像。这可以通过海报图像实现，参见图 17.5.2 和图 17.5.3。

```
...
<body>
  <video src="paddle-steamer.webm"
    → width="369" height="208" autoplay
    → loop></video>
</body>
</html>
```

图 17.5.1 设置了自动播放和循环播放的单个 WebM 视频。如果这里不设置 `controls`，访问者就无法停止视频！因此，如果将视频指定为循环，最好包含 `controls`。即便如此，自动循环播放的视频也可能让访问者非常不爽

```
...
<body>
  <video src="paddle-steamer.webm"
    → width="369" height="208"
    → poster="paddle-steamer-poster.jpg"
    → controls></video>
</body>
</html>
```

图 17.5.2 指定了海报图像（当页面加载并显示视频时显示该图像）的单个 WebM 视频（含控件）

1. 为视频添加自动播放和循环播放

输入 `<video src="my-video.ext" autoplay loop></video>`，这里的 `my-video.ext` 指向你的视频文件。

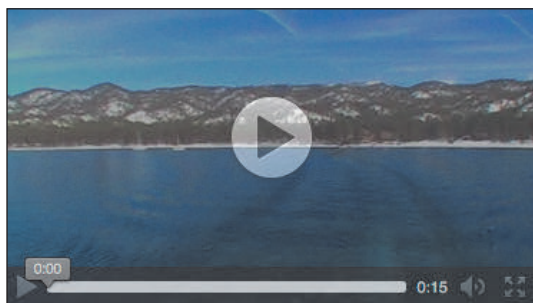


图 17.5.3 显示海报图像的视频。在这个例子中，图像是取自视频本身的一幅屏幕截图

2. 为视频指定海报图像

输入 `<video src="my-video.ext" controls poster="my-poster.jpg"></video>`，其中，*my-video.ext* 指向你的视频文件，*my-poster.jpg* 是想用做海报图像的图像。

17.6 阻止视频预加载

如果认为用户观看视频的可能性较低（如该视频并不是页面的主要内容），那么可以告诉浏览器不要预先加载该视频，参见图 17.6.1。这样能节省带宽，尤其对于移动设备用户来说好处多多。

对于设置了 `preload="none"` 的视频，在初始化视频之前，浏览器显示视频的方式并不一样（参见图 17.6.2 和图 17.6.3）。

```
...
<body>
  <video src="paddle-steamer.webm"
    → preload="none" controls></video>
</body>
</html>
```

图 17.6.1 页面完全加载时也不会加载的单个 WebM 视频。仅在用户试着播放该视频时才会加载它。注意我省略了 `width` 和 `height` 属性

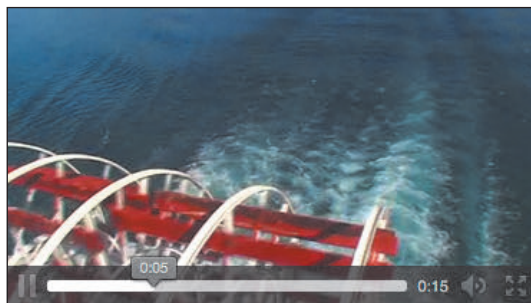
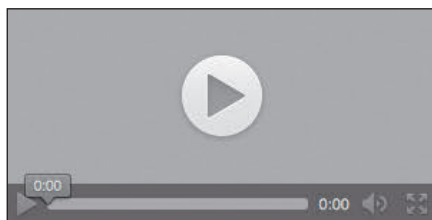


图 17.6.2 在 Firefox 中将 `preload` 设为 `none` 的视频。如你所见，什么也不会显示，因为浏览器没有得到关于该视频的任何信息（连尺寸都不知道），也没有指定海报图像。如果用户播放视频，浏览器会获取视频的尺寸，并调整视频大小

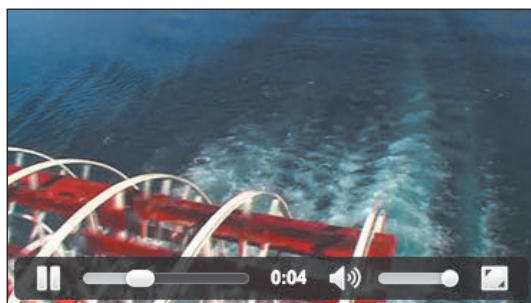


图 17.6.3 Chrome（上）在控制组件上面显示一个空白的矩形。这时，控制组件的大小比访问者播放视频时（下）显示的组件要窄一些

告诉浏览器不预先加载视频的步骤

输入 `<video src="my-video.ext" preload="none" controls></video>`，这里的 `my-video.ext` 指向你的视频文件。

其他预加载设置

`preload` 的默认值是 `auto`。这会让浏览器具有用户将要播放该视频的预期，从而做好准备，让视频可以很快进入播放状态。浏览器会预先加载大部分视频甚至整个视频。因此，在视频播放的过程中对其进行多次开始、暂停的操作会变得更不容易，因为浏览器总是试着下载较多的数据让访问者观看。

在 `none` 和 `auto` 之间有一个不错的中间值，即 `preload="metadata"`。这样做会让浏览器仅获取视频的基本信息，如尺寸、时长甚至一些关键的帧。在开始播放之前，浏览器不会显示白色的矩形，而且视频的尺寸也会与实际尺寸一致。

使用 `metadata` 会告诉浏览器，用户的连接速度并不快，因此需要在不妨碍播放的情况下尽可能地保留带宽资源。

17.7 使用多种来源的视频和备用文本

一切看起来都很棒，不过你也应该注意到了，前面所有的例子都只用了一个视频文件（同时意味着只有一种格式）。你已经知道了，要获得所有兼容 HTML5 的浏览器的支持，至少需要提供两种格式的视频：MP4 和 WebM。

如何做到呢？这时就要用到 HTML5 的 `source` 元素了。通常，`source` 元素用于定义一个以上的媒体元素（在这个例子中为 `video`）的来源。

一个 `video` 元素中可以包含任意数量的 `source` 元素，因此为我们的视频定义两种不同的格式是相当容易的，如图 17.7.1 所示。浏览器会加载第一个它支持的 `source` 元素引用的文件格式，并忽略其他的来源（如图 17.7.2 和图 17.7.3 所示）。无法播放 HTML5 视频的浏览器则会显示你提供的消息中的备用链接（如图 17.7.1 所示）。

```
...
<body>
  <video width="369" height="208" controls>
    <source src="paddle-steamer.mp4"
      → type="video/mp4">
    <source src="paddle-steamer.webm"
      → type="video/webm">
    <p><a href="paddle-steamer.mp4">
      → Download the video</a></p>
  </video>
</body>
</html>
```

图 17.7.1 这里为视频定义了两个源：一个 MP4 文件和一个 WebM 文件。（注意 `video` 开始标签中没有像之前指定单个视频来源时使用的 `src` 属性。）旧的浏览器则只会显示 `p` 元素中的文本（参见图 17.7.4）。要确保备用链接或文本位于 `video` 里面，不然所有的浏览器都会显示它



图 17.7.2 IE10 等支持 MP4 的文件会加载 `paddle-steamer.mp4`

1. 指定两种不同的视频来源（带有备用信息）

(1) 获取视频资源（这次需要两个）。

- (2) 输入 <video controls> 开始 video 元素（含默认控件集）。
- (3) 输入 <source src="my-video.mp4" type="video/mp4">，这里的 my-video.mp4 是 MP4 视频源文件的名称。
- (4) 输入 <source src="my-video.webm" type="video/webm">，这里的 my-video.webm 是 WebM 视频源文件的名称。
- (5) 为旧浏览器输入备用链接或者备用文本信息，参见图 17.7.4。
- (6) 输入 </video> 结束 video 元素。

2. 多个媒体源的更多信息

接下来我们将学习 source 元素的各种可用的属性，不过在此之前，不妨先简单地了解一下为同一媒体指定多个来源的工作原理。

当浏览器发现 video 元素时，首先会查看该元素本身是否定义了 src。如果没有，就会检查 source 元素。浏览器会逐个查看这些来源，直到找到它可以播放的一个来源。一旦找到这样一个，就会播放它并忽略其他的来源。

在前一个例子中，Safari 会播放 MP4 文件（参见图 17.7.2）并完全忽略 WebM 文件，而 Firefox 则无法播放 MP4 文件，从而转向它能播放的 WebM 文件，如图 17.7.3 所示。（Firefox 正致力于将对 MP4 的支持添加到所有版本，因此当你读到本书的时候情况可能有所改变。）



图 17.7.3 不支持 MP4 但支持 WebM 的浏览器会加载 paddle-steamer.webm

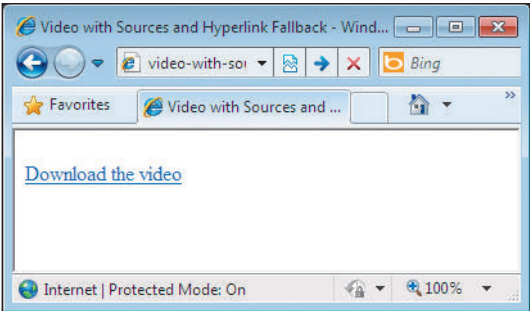


图 17.7.4 IE8 会忽略 video 和 source 元素，仅显示下载链接。这里我选择链接到视频的 MP4 版本（参见图 17.7.1），我也可以选择 WebM 版本，或者提供两个版本的链接

既不识别 video 元素又不识别 source 元素的浏览器（即不支持 HTML5 的浏览器）在解析文档时会完全忽略这些标签，它只会显示在 video 元素结束标签之前输入的文本。

表 17.7.1 列出了 source 元素的属性。

表 17.7.1 source 的属性

名 称	描 述
src	视频来源的 URL
type	用于指定视频的类型，帮助浏览器决定它是否能播放该视频。如图 17.7.1 所示，该属性的值反映的是视频的格式，即编解码器（如 video/mp4、video/webm、video/ogg）
media	用于为视频来源指定 CSS3 媒体查询，从而可以为具有不同屏幕尺寸的设备指定不同的（如更小的）视频

提示 图 17.7.1 中提供的备用内容除了可以放置链接，还可以放置视频的截图或其他内容，例如，`<p>Sorry, your browser doesn't support HTML5 video.</p>`，然后再提供一个链接。

提示 可以在 www.bigbuckbunny.org/index.php/download/ 找到一些免费的视频，用于试验 video 和 source 元素。该网站没有 WebM 格式的视频，不过你可以通过 17.2 节提供的工具进行格式转换。

17.8 提供可访问性

利用现代浏览器提供的原生可访问性支持，原生多媒体可以更好地使用键盘进行控制，这是原生多媒体的另一个好处。

HTML5 视频和音频的键盘可访问性支持在 Firefox、Internet Explorer 和 Opera 中表现良好。不过对于 Chrome 和 Safari，截至本书写作之际，实现键盘可访问性的唯一办法是自制播放控件。为此，需要使用 JavaScript Media API（这也是 HTML5 的一部分），不过这已经超出了本章的讨论范围。

HTML5 还指定了一种新的文件格式 WebVTT（Web Video Text Track，Web 视频文本轨道）用于包含文本字幕、标题、描述、篇章等视频内容。关于 WebVTT 和字幕的进一步讨论已经超出了本章的范围，不过更多信息可以参见 www.iandevlin.com/blog/2011/05/html5/webvtt-and-video-subtitles，其中包括为了对接规范修改在 2012 年进行的更新。

提示 Ian Devlin 的 *HTML5 Multimedia: Develop and Design*（Peachpit Press，2011）一书中有专门演示如何创建个人的可访问控件集和使用 WebVTT 的章节。<http://net.tutsplus.com/tutorials/html-csstechiques/an-in-depth-overview-of-html5-multimedia-and-accessibility/> 摘录了本书部分章节。

提示 Terrill Thompson 比较了不同浏览器对 HTML5 视频可访问性的支持情况，见 <http://terrillthompson.com/blog/366>。

提示 WebVTT 规范还在制订之中，因此通过它设置标题等内容的方法未来有可能会变化。

17.9 音频文件格式

学会了使用 HTML5 原生媒体在网页中添加视频之后，下面看看如何添加音频。同 HTML5 视频一样，HTML5 也支持大量不同的音频文件格式（编解码器）。

- ❑ Ogg Vorbis 使用的文件扩展名为 .ogg，支持它的浏览器包括 Firefox 3.5+、Chrome 5+ 和 Opera 10.5+。
- ❑ MP3 使用的文件扩展名为 .mp3，支持它的浏览器包括 Safari 5+、Chrome 6+、Internet Explorer 9+ 和 iOS。
- ❑ WAV 使用的文件扩展名为 .wav，支持它的浏览器包括 Firefox 3.6+、Safari 5+、Chrome 8+ 和 Opera 10.5+。
- ❑ AAC 使用的文件扩展名为 .aac，支持它的浏览器包括 Safari 3+、Internet Explorer 9+、iOS 3+ 和 Android 2+。
- ❑ MP4 使用的文件扩展名为 .mp4，支持它的浏览器包括 Safari 3+、Chrome 5+、Internet Explorer 9+、iOS 3+ 和

Android 2+。

- ❑ Opus 使用的文件扩展名是 .opus。这是一种新的音频文件格式，在本书写作之时只有 Firefox 支持。

你应该还记得，MP4 是一种视频编解码器，不过它也可以仅对音频数据进行编码。

提示 对于视频，需要使用两种不同的文件格式才能确保获得所有兼容 HTML5 的浏览器的支持。对于音频，最好的两种格式是 Ogg Vorbis 和 MP3。

提示 在 17.2 节的补充材料“转换文件格式”中提到的 Miro Video Converter 应用程序也可以用来转换音频。

17.10 在网页中添加带控件的单个音频文件

让我们来实际操作一下向网页中添加音频文件。过程跟添加视频文件非常相似，只不过是使用 `audio` 元素，参见图 17.10.1。当然，同视频控件一样，每个浏览器都有处理音频控件外观的独特方式（参见图 17.10.2 ~ 图 17.10.6）。

```
...
<body>
  <audio src="piano.ogg" controls></audio>
</body>
</html>
```

图 17.10.1 单个 Ogg 编码的音频（包含默认控件集）。可以省略 `controls` 属性，但因为音频文件不可见，所以什么都不会显示



图 17.10.2 Firefox 中的音频控件。跟视频控件一样，Firefox（和 IE10）标记的音频文件要比其他浏览器标记的长 1 秒



图 17.10.3 Chrome 中的音频控件



图 17.10.4 IE10 中的音频控件



图 17.10.5 Opera 中的音频控件



图 17.10.6 Safari 中的音频控件

如果浏览器不支持某种音频文件格式

浏览器面对不支持的音频格式时显示的消息并不一致。例如，如果仅指定 Ogg 文件（参见图 17.10.1），IE10 会显示如图 17.10.7 所示的消息，而 Safari 则会显示播放控件和 Loading... 字样。图 17.10.4 和图 17.10.6 显示了它们支持某文件格式（如 MP3）时显示的播放控件的样子。

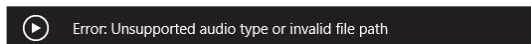


图 17.10.7 IE10 不支持你在 `src` 中指定的音频文件格式的情况

可用于 `audio` 元素的属性参见表 17.10.1。

表 17.10.1 音频属性

名 称	描 述
<code>src</code> (源)	指定音频文件的 URL
<code>autoplay</code> (自动播放)	当音频可以播放时立即开始播放
<code>controls</code> (控件)	添加浏览器为音频设置的默认控件
<code>muted</code> (静音)	让音频静音
<code>loop</code> (循环)	让音频循环播放
<code>preload</code> (预加载)	告诉浏览器要加载的音频内容的多少。可以是以下三个值： <code>none</code> 表示不加载任何音频； <code>metadata</code> 表示仅加载音频的元数据（如长度）； <code>auto</code> 表示让浏览器决定怎样做（这是默认的设置）

在网页中添加带控件的单个音频文件的步骤

- ❑ 获取音频文件。
- ❑ 输入 `<audio src="my-audio.ext" controls>`
`</audio>`，其中的 `my-audio.ext` 是音频文件的位置、名称和扩展名。

17.11 自动播放、循环和预加载音频

这一节介绍的属性与 `video` 对应的属性的原理是一样的。使用 `autoplay` 属性会让音频文件在页面加载时就自动播放（如图 17.11.1 和图 17.11.2 所示）。使用 `loop` 属性可以让音频文件循环播放（如图 17.11.3 所示）。对 `preload` 属性使用表 17.10.1 中的值，可以指定浏览器预加载音频文件的方式（如图 17.11.4 所示）。

```
...
<body>
  <audio src="piano.ogg" autoplay
    → controls></audio>
</body>
</html>
```

图 17.11.1 当页面加载时会自动播放的 Ogg 音频文件（含默认控件集）



图 17.11.2 加载时会自动开始播放的音频文件（含控件）

```
...
<body>
  <audio src="piano.ogg" loop controls>
    → </audio>
</body>
</html>
```

图 17.11.3 会循环播放的 Ogg 音频文件（含默认控件集）

```
...
<body>
  <audio src="piano.ogg" preload=
    → "metadata" controls></audio>
</body>
</html>
```

图 17.11.4 页面加载时，Ogg 音频文件只加载元数据（如长度）

1. 为音频文件添加控件并让其自动播放的步骤

输入 `<audio src="my-audio.ext" autoplay controls></audio>`，这里的 `my-audio.ext` 指

向你的音频文件。如果省略 `controls` 属性，音频文件会自动播放，但不会显示在浏览器页面上。

2. 让音频文件循环播放的步骤

输入 `<audio src="my-audio.ext" loop controls></audio>`，这里的 `my-audio.ext` 指向你的音频文件。

3. 让浏览器仅预加载音频元数据

输入 `<audio src="my-audio.ext" preload="metadata" controls></audio>`，这里的 `my-audio.ext` 指向你的音频文件。

提示 可以对 `audio` 元素包含 `autoplay`、`loop` 和 `preload` 属性的任意组合。注意，包含 `autoplay` 属性会覆盖 `preload` 属性的设置，因为只有加载音频文件才能播放。

提示 17.6 节介绍的 `preload` 属性的 `auto`、`none` 和 `metadata` 值的用法说明同样适用于 `audio` 元素。（关于如何确定尺寸的内容并不适用于 `audio`。）记住，`preload` 属性的值不会影响浏览器的行为，它只影响请求。

17.12 提供带备用内容的多个视频源

为了获得所有兼容 HTML5 的浏览器的支持，至少需要为音频提供两种格式。实现这一目标的方法同 `video` 元素也是一样的，即使用 `source` 元素。备用方法跟 `audio` 元素也一样，如图 17.12.1 所示。你可能猜到了，浏览器会忽略它不支持的音频格式，而显示它支持的音频格式（参见图 17.12.2 和图 17.12.3）。不支持 `audio` 元素的浏览器显示备用内容，参见图 17.12.4。

```
...
<body>
  <audio controls>
    <source src="piano.ogg"
      → type="audio/ogg">
    <source src="piano.mp3"
      → type="audio/mp3">
    <p>Your browser doesn't support
      → HTML5 audio, but you can
      → <a href="piano.mp3">download the
      → audio file</a> (MP3, 1.3 MB)</p>
  </audio>
</body>
</html>
```

图 17.12.1 这个 `audio` 元素（含默认控件集）定义了两个音频源文件，一个编码为 Ogg，另一个为 MP3。完整的过程同指定多个视频源文件的过程是一样的。浏览器会忽略它不能播放的，仅播放它能播放的



图 17.12.2 支持 Ogg 的浏览器（如 Firefox）会加载 `piano.ogg`。Chrome（这里并未显示）同时理解 Ogg 和 MP3，但是会加载 Ogg 文件，因为在 `audio` 元素的代码中，Ogg 文件位于 MP3 文件之前



图 17.12.3 不支持 Ogg 格式，但支持 MP3 格式的浏览器（IE10）会加载 `piano.mp3`

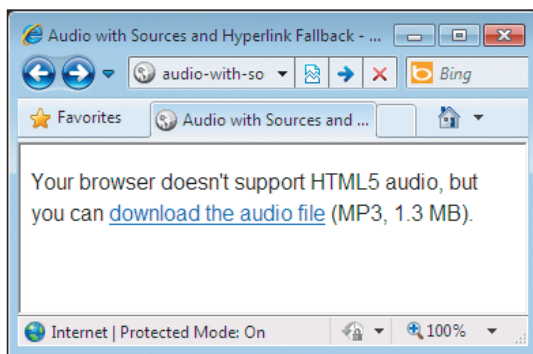


图 17.12.4 旧浏览器（如 IE8）会显示备用信息

指定两种不同的音频来源的步骤

(1) 获取音频文件。

(2) 输入 `<audio controls>` 开始 `audio` 元素（含默认控件集）。

(3) 输入 `<source src="my-audio.ogg" type="audio/ogg">`，这里的 `my-audio.ogg` 是 Ogg Vorbis 指向你的音频文件。

(4) 输入 `<source src="my-audio.mp3" type="audio/mp3">`，这里的 `my-audio.mp3` 指向你的 MP3 音频文件。

(5) 可选步骤（但推荐带着）。对于不支持 HTML5 音频元素的浏览器创建备用内容或音频下载链接。

(6) 输入 `</audio>` 结束 `audio` 元素。

type 属性

`type` 属性可以帮助浏览器判断它是否能播放某个文件。对音频文件来说，其值总是 `audio/` 加上格式本身，包括 `audio/ogg`、`audio/mp3`、`audio/aac`、`audio/wav` 和 `audio/mp4`。

17.13 添加具有备用 Flash 的视频和音频

除了可以提供下载链接作为备用方案，还可以（或许可以说应该）嵌入一个能播放 MP4 视频文件的 Flash 备用播放器。是的，尽管 HTML5 和原生多媒体非常强大，但为了照顾那些无法处理这些技术的旧浏览器（比如 IE9 之前的版本），恐怕还得求助于嵌入 Flash 内容的方法。也就是说，如果希望获得尽可能多的用户，至少还有一种选择！

我会使用 John Dyer 的 MediaElement.js（<http://mediaelementjs.com>）演示如何嵌入

Flash。这并不是唯一的解决方案，但它做得不错，而且被 WordPress（一个广泛使用的博客和 CMS^① 软件）这样的系统采用了。我在提示里面还列举了一些其他的替代方案。

要让 MediaElement.js 与原生视频和音频一起运行，需要做一些额外的工作。首先，需要获取 MediaElement.js 文件，将它们放到网站文件夹里（如图 17.13.1 至图 17.13.3 所示）。然后将其中的一些文件放到网页里（如图 17.13.4 和图 17.13.5 所示）。这些步骤对视频和音频来说都是必要的。

然后，你就可以在页面中插入视频（如图 17.13.6 和图 17.13.7 所示）和音频（如图 17.13.8 和图 17.13.9 所示）了。当 MediaElement.js 由图 17.13.6 和图 17.13.8 中的脚本初始化以后，它会自动决定让浏览器使用 HTML5 原生播放系统还是 Flash 播放器。

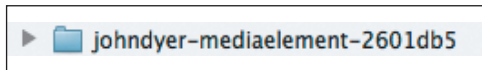


图 17.13.1 随着 MediaElement.js 的升级，ZIP 文件解压后的文件夹的名称可能与这里不一样

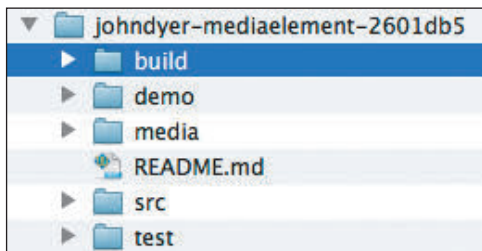


图 17.13.2 要让 MediaElement.js 运行在你的网站上，只需要 `build` 这一个文件夹里的内容

1. 获取 MediaElement.js 文件，并将文件包括在网站目录中

(1) 访问 <http://mediaelementjs.com>。点击 Download Latest 按钮，下载 ZIP 包。

① CMS 是内容管理系统的简称。——译者注

(2) 在你的电脑上找到下载下来的 ZIP 文件。该文件通常位于 Downloads 文件夹（除非你指定了别的存放路径）。解压该文件（通常是双击 ZIP 文件名），会看到一个类似于图 17.13.1 中所示的文件夹。

(3) 打开文件夹，呈现出子文件夹。复制 build 文件夹（如图 17.13.2 所示），再粘贴到网站文件夹（如图 17.13.3 所示）。

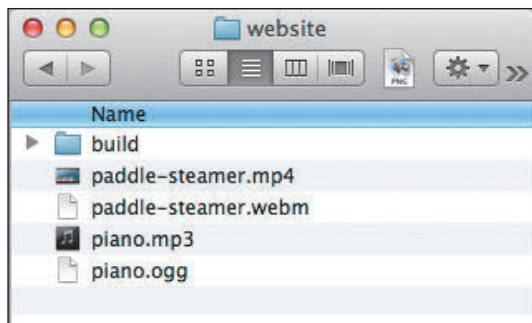


图 17.13.3 将 build 文件夹粘贴到网站的文件夹（你可能使用的是其他的文件夹名称）。为了简单，我将 build 直接放到了媒体文件所在的目录。通常，我会将媒体文件统一放到名为 media 或类似名称的文件夹里

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>HTML5 Media with Fallback Flash
    → Player</title>
  <script src="build/jquery.js">
    → </script>
  <script src="build/mediaelement-and-
    → player.min.js"></script>
  <link rel="stylesheet" href="build/
    → mediaelementplayer.min.css" />
</head>
<body>

</body>
</html>
```

图 17.13.4 这些文件包含了让媒体播放器得以工作的文件，还包括了统一的外观样式文件

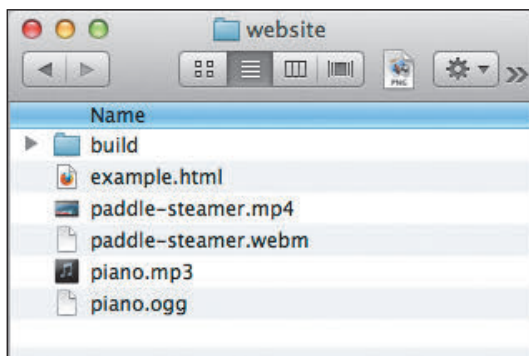


图 17.13.5 现在，有了 build 文件夹、你的网页和你的媒体文件

2. 为网页添加 MediaElement.js

页面需要加载一套特殊的文件，无论是为视频还是为音频设置备用播放器。

(1) 创建一个新的 HTML 页面，或者打开一个现有的页面。

(2) 将图 17.13.4 中突出显示的代码添加到页面中，从而让页面加载必要的样式表和 JavaScript 文件。

(3) 将页面保存到放置 build 文件夹的目录（如图 17.13.5 所示）。

3. 为视频添加备用 Flash 的步骤

(1) 获取视频资源。

(2) 输入 <video controls> 开始 video 元素（含默认控件集）。如有需要，在此处指定 width、height、poster 等其他属性。

(3) 输入 <source src="my-video.mp4" type="type/mp4">，这里的 myVideo.mp4 是 MP4 视频源文件的名称。

(4) 输入 <source src="myVideo.webm" type="video/webm">，这里的 my-video.webm 是 WebM 视频源文件的名称。

(5) 如有需要，为不支持 HTML5 视频和 Flash 的浏览器提供备用信息和链接。

(6) 输入 `</video>` 结束 video 元素。

(7) 将图 17.13.6 中突出显示的代码添加到页面中，从而初始化一个视频播放器。将 `script` 放到 `</body>` 的前面，哪怕页面拥有视频以外的其他内容。

现在，你的视频拥有了尽可能广泛的受众，参见图 17.13.7。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>HTML5 Media with Fallback Flash
    → Player</title>
  <script src="build/jquery.js"></script>
  <script src="build/mediaelement-and-
    → player.min.js"></script>
  <link rel="stylesheet" href="build/
    → mediaelementplayer.min.css" />
</head>
<body>
<video width="369" height="208"
→ preload="metadata" controls>
  <source src="paddle-steamer.mp4"
    → type="video/mp4">
  <source src="paddle-steamer.webm"
    → type="video/webm">
  <p>Your browser doesn't support HTML5
    → video or Flash, but you can <a href=
    → "paddle-steamer.mp4">download the
    → video</a> (MP4, 2.4 MB)</p>
</video>

<!-- 以下代码要放在所有内容下面 -->
<script>
$('video').mediaelementplayer();
</script>
</body>
</html>
```

图 17.13.6 这段视频代码看起来很熟悉。事实上，它与先前引用多个来源的代码是一样的。根据需要，可以添加其他的属性或删除一部分属性。底部的脚本告诉 `MediaElement.js` 施展它的魔法



图 17.13.7 无论使用哪种浏览器，无论是否针对不支持 HTML5 视频的浏览器使用 Flash 版本，视频的播放控件看起来都是一样的

... 页面顶端跟图 17.13.4 和图 17.13.6 一样 ...

```
<body>
<audio controls>
  <source src="piano.ogg" type="audio/ogg">
  <source src="piano.mp3" type="audio/mp3">
  <p>Your browser doesn't support HTML5
    → audio or Flash, but you can <a href=
    → "piano.mp3">download the audio file
    → </a> (MP3, 1.3 MB)</p>
</audio>

<!-- 下面的代码放在所有内容之后 -->
<script>
$('audio').mediaelementplayer();
</script>
</body>
</html>
```

图 17.13.8 这段代码与图 17.13.6 中代码的唯一区别就是这里指定的是 `audio` 而非 `video`。`audio` 元素看起来也相当熟悉



图 17.13.9 同视频一样，无论在哪个浏览器，无论是否使用 Flash 版本，音频的播放控件看起来都是一样的

```
...
<script>
$('audio,video').mediaelementplayer();
</script>
</body>
</html>
```

图 17.13.10 以下代码会初始化页面中所有的音频和视频

4. 为音频添加备用 Flash 的步骤

(1) 获取音频文件。

(2) 输入 `<audio controls>` 开始 audio 元素（含默认控件集）。

(3) 输入 `<source src="my-audio.ogg" type="audio/ogg">`，这里的 *my-audio.ogg* 指向你的 Ogg Vorbis 音频文件。

(4) 输入 `<source src="my-audio.mp3" type="audio/mp3">`，这里的 *my-audio.mp3* 指向你的 MP3 音频文件。

(5) 如果需要，为不支持 HTML5 音频或者 Flash 的旧浏览器创建信息和链接。

(6) 输入 `</audio>` 结束 audio 元素。

(7) 将图 17.13.8 中突出显示的代码添加到页面中去，从而初始化音频播放器。将 `script` 放到 `</body>` 的前面，哪怕页面拥有音频以外的其他内容。

现在，你的音频拥有了尽可能广泛的受众，参见图 17.13.9。一天便有了两个成就！

提示 如果页面同时包含了视频和音频，那么需要修改页面询问的脚本，如图 17.13.10 所示。

提示 Internet Explorer 8 等浏览器会忽略 `audio` 和 `source` 元素，直接使用备用 Flash 播放器。只要用户安装了 Flash，就能播放视频或音频内容。

提示 如果没有为视频指定 `preload="metadata"`（就像图 17.13.6 所示的那样），也没有指定海报图像，那么视频的 Flash 版本会显示一个黑色的矩形而不是视频的一帧。不过，当视频开始播放时，黑色的矩形就会被视频本身替代。

提示 如果在 `video` 开始标签中指定了 `width` 和 `height`，那么视频的 Flash 版本就有可能显示黑边。不过，如果不指定尺寸，视频会在一开始呈现一个比正常尺寸要大的区域，然后再缩小到正常的大小。

提示 其他的 Flash 播放解决方案包括 Video.js (www.videojs.com)、JW Player (www.longtailvideo.com/jw-player/)、Flowplayer (<http://flowplayer.org>) 等。JW Player 和 Flowplayer 的免费版本会在媒体播放器上显示它们的标识。

当 Flash 播放器不起作用时

由于 Flash 的安全设置，当你在自己的电脑上测试 Flash 播放器（即所有的文件都位于自己的电脑上，而不是在服务器上）时，有可能无法播放媒体文件。

一种解决办法是将你的 MP3 和 MP4 文件上传到 Web 服务器，并在 HTML 中使用绝对路径引用它们。

例如，将图 17.13.6 中的文件上传到服务器的 `media` 文件夹中。可以将第一个 `src` 值修改如下（将 `www.yourdomain.com` 替换为真实域名）：

```
<source src="http://www.yourdomain.com/
→ media/paddle-steamer.mp4" type="video/
→ mp4">
```

然后保存 HTML 页面，再在你的电脑上测试（HTML 页面不必位于服务器上）。你还可以修改 Flash 安全设置，允许本地目录播放 Flash。参见 <http://mediaelementjs.com/#installation>。

如果还是有问题，参见 17.2 节的补充材料“设置 MIME 类型”。

17.14 高级多媒体

使用 HTML5 原生多媒体的另一个好处是可以利用很多来自 HTML5 或与 HTML5 相关的新特性和新功能。本节主要讨论其中的两个：canvas 元素和 SVG。

1. 通过 canvas 操作视频

使用 canvas 元素及相应的 JavaScript API 可以在网页上描制并创建动画。

可以对 HTML5 视频应用这些 API，因为 video 元素可以同其他 HTML 元素一样进行处理，因此它也可以被 canvas 访问和获取。

通过 JavaScript API，可以从播放的视频中抓取图像，并在 canvas 元素中重新绘制该图像，从而创建视频的截图。

通过 API 可以对单个图像像素进行操作，同时由于可以根据视频在 canvas 中创建图像，因而可以调整视频的像素。例如，可以将视频转化为灰度模式。

这只是让你对通过 canvas 操作 video 元素建立一些简单的概念，对这一主题的深入探讨已经超出了本书的范围。

2. 联合使用 SVG 和视频

人们开始关注的另一项与 HTML5 有关的

技术是 SVG（Scalable Vector Graphic，可缩放矢量图形）。

SVG 已经存在相当一段时间了（它诞生自 1999 年），但直到 HTML5 才有了 svg 元素。通过该元素可以在网页本身嵌入 SVG 定义。

SVG 使用 XML 定义图形和图像，浏览器则对其进行解释和使用，从而描绘出真正的图形。SVG 定义所包含的全部内容就是如何绘制和绘制什么的说明。

使用 SVG 创建的图像也是基于矢量而不是基于光栅的。这意味着它们可以很好地适应缩放，因为浏览器只是简单地依照绘制说明，根据所需的尺寸，将图形绘制出来。相较而言，GIF、PNG 和 JPEG 文件等光栅光栅图像包含的是像素数据，如果要以远大于原始图像的尺寸重新绘制图像，就会因为缺少足够的像素数据导致图像质量受损。

关于 SVG 的完整讨论超出了本章的范围，这里只是想让你知道视频可以同 SVG 定义联合使用。通过 SVG 创建的图形可以用于对视频进行遮罩，也即只显示能透过该图形（如圆圈）的底层视频。还可以创建可调整为任意大小的自定义视频控件。

此外，还有一些 SVG 滤镜可以应用于 HTML5 视频，如黑白转换、高斯模糊、色彩饱和度等。

17.15 更多资源

本章仅涵盖了 HTML5 多媒体的基础知识。关于这一主题，还有很多值得学习的东西。这里有大量有关资源供你研究。

1. 在线资源

- ❑ “Video on the Web”
(<http://diveinto.html5doctor.com/video.html>)
- ❑ “WebVTT and Video Subtitles”
(www.iandevlin.com/blog/2011/05/html5/webvtt-and-video-subtitles)
- ❑ “HTML5 Canvas: The Basics”
(<http://dev.opera.com/articles/view/html-5-canvas-the-basics>)
- ❑ “Learning SVG”
(<http://my.opera.com/tagawa/blog/learning-svg>)

2. 图书

- ❑ Ian Devlin 所著的 *HTML5 Multimedia: Develop and Design* (Peachpit Press, 2011), 参见 <http://html5multimedia.com>
- ❑ Shelley Powers 所著的 *HTML5 Media* (O'Reilly Media, 2011)
- ❑ Silvia Pfeiffer 所著的 *The Definitive Guide to HTML5 Video* (Apress, 2010)

本章内容

- 结构化表格
- 让单元格跨越多列或多行

在日常生活中，我们对表格式数据已经很熟悉了。这种数据有多种形式，如财务数据、调查数据、事件日历、公交车时刻表、电视节目表等。在大多数情况下，这类信息都由列标题或行标题加上数据本身构成。

本章将对 `table` 元素及其子元素进行讲解，重点是基本的 `table` 结构和样式。HTML 表格可以很复杂，不过很少需要实现特别复杂的表格（除非是具有丰富数据的网站）。对于一些比较复杂的示例，参见以下 URL。

- Roger Johansson 的 “Bring On the Tables”
(www.456bereastreet.com/archive/200410/bring_on_the_tables/)
- Roger Hudson 的 “Accessible Data Tables”
(www.usability.com.au/resources/tables.cfm)
- Stephen Ferg 的 “Techniques for Accessible HTML Tables”
(<http://accessiblehtml.sourceforge.net/>)

18.1 结构化表格

放在电子表格中的信息通常很适合用

HTML 表格呈现。

从基本层面看，`table` 元素是由行组成的，行又是由单元格组成的。每个行 (`tr`) 都包含标题单元格 (`th`) 或数据单元格 (`td`)，或者同时包含这两种单元格。如果认为整个表格添加一个标题有助于访问者理解该表格，可以提供 `caption`。在浏览器中，标题通常显示在表格上方，其用途显而易见。此外，`scope` 属性（也是可选的，不过推荐使用）可以告诉屏幕阅读器和其他辅助设备当前的 `th` 是列的标题单元格（使用 `scope="col"`）还是行的标题单元格（使用 `scope="row"`），抑或是用于其他目的的单元格（参见最后一条提示）。表格元素的基本实现代码如图 18.1.1 所示。

在默认情况下，表格在浏览器中呈现的宽度是其中的信息在页面可用空间里所需要的最小宽度，如图 18.1.2 所示。很容易就会想到可以通过 CSS 改变表格的格式，后面很快就会讲到。

不过，图 18.1.1 中的表格似乎缺少一些东西。如何知道每行数据表示的是什么呢？如果每个行也有标题单元格，就很容易理解了。添加这些单元格只需要在每行开头添加一个 `th` 元素就可以了。列标题应设置 `scope="col"`，而每个行的 `th`（位于 `td` 之前）则应设置 `scope="row"`，如图 18.1.3 所示。

```
...
<body>

<table>
  <caption>Quarterly Financials for
  → 1962-1964 (in Thousands)</caption>
  <tr>
    <th scope="col">1962</th>
    <th scope="col">1963</th>
    <th scope="col">1964</th>
  </tr>
  <tr>
    <td>$145</td>
    <td>$167</td>
    <td>$161</td>
  </tr>
  <tr>
    <td>$140</td>
    <td>$159</td>
    <td>$164</td>
  </tr>
  <tr>
    <td>$153</td>
    <td>$162</td>
    <td>$168</td>
  </tr>
  <tr>
    <td>$157</td>
    <td>$160</td>
    <td>$171</td>
  </tr>
</table>

</body>
</html>
```

图 18.1.1 每个行都是由 tr 元素标记的。这是个很简单的表格，它只有一个包含标题单元格（th 元素）的行和三个包含数据单元格（td 元素）的行。本例中也包含了 caption 元素，不过它是可选的，参见第一条提示

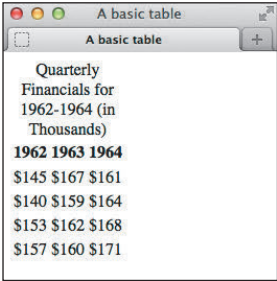


图 18.1.2 在默认情况下，th 文本是以粗体显示的，th 和 caption 文本都是居中对齐的，表格的宽度就是内容所需的宽度

```
...
<body>

<table>
  <caption>Quarterly Financials for
  → 1962-1964 (in Thousands)</caption>
  <thead> <!-- 表格头部 -->
    <tr>
      <th scope="col">Quarter</th>
      <th scope="col">1962</th>
      <th scope="col">1963</th>
      <th scope="col">1964</th>
    </tr>
  </thead>
  <tbody> <!-- 表格主体 -->
    <tr>
      <th scope="row">Q1</th>
      <td>$145</td>
      <td>$167</td>
      <td>$161</td>
    </tr>
    <tr>
      <th scope="row">Q2</th>
      <td>$140</td>
      <td>$159</td>
      <td>$164</td>
    </tr>
    ... [Q3 and Q4 rows] ...
  </tbody>
  <tfoot> <!-- 表格尾部 -->
    <tr>
      <th scope="row">TOTAL</th>
      <td>$595</td>
      <td>$648</td>
      <td>$664</td>
    </tr>
  </tfoot>
</table>

</body>
</html>
```

图 18.1.3 通过指定 thead、tbody 和 tfoot 显式地定义了表格的不同部分。接着，在每行的开头添加了 th 元素。tbody 和 tfoot 中的 th 设置了 scope="row"，表明它们是行标题。表格在浏览器中的显示效果如图 18.1.4 所示

Quarter	1962	1963	1964
Q1	\$145	\$167	\$161
Q2	\$140	\$159	\$164
Q3	\$153	\$162	\$168
Q4	\$157	\$160	\$171
TOTAL	\$595	\$648	\$664

图 18.1.4 这个表既有列标题，也有行标题。由于添加了新列，表格变宽了，因此整个表格的标题也比图 18.1.2 中的更宽一些

图 18.1.3 中还特意引入了其他一些专门用于定义表格的元素，即 `thead`、`tbody` 和 `tfoot`。`thead` 元素可以显式地将一行或多行标题标记为表格的头部。`tbody` 元素用于包围所有的数据行。`tfoot` 元素可以显式地将一行或多行标记为表格的尾部。可以使用 `tfoot` 包围对列的计算值（就像图 18.1.3 中所示的那样），也可以在长表格（如列车时刻表）中使用 `tfoot` 重复 `thead` 中的内容（如果表格在打印时超过一页，有的浏览器会在每一页都打印 `tfoot` 和 `thead` 元素的内容）。`thead`、`tfoot` 和 `tbody` 元素不会影响表格的布局，也不是必需的（不过推荐使用它们）。如果包含了 `thead` 或 `tfoot`，则必须同时包含 `tbody`。此外，还可以对它们添加样式。

创建表格结构的步骤

(1) 输入 `<table>`。

(2) 如果需要，输入 `<caption>caption content</caption>`，其中的 `caption content` 是对表格的描述。

(3) 如果需要，在要创建的表格部分的第一个 `tr` 元素之前，输入 `<thead>`、`<tbody>` 或 `<tfoot>`（需要说明的是，`<tbody>` 不能在

`<thead>` 之前）。

(4) 输入 `<tr>` 定义行的开始。

(5) 输入 `<th scope="scopetype">` 开始标题单元格（其中的 `scopetype` 可以是 `col`、`row`、`colgroup` 或 `rowgroup`），或者输入 `<td>` 定义数据单元格的开始。

(6) 输入单元格的内容。

(7) 输入 `</th>` 结束标题单元格，或者输入 `</td>` 结束数据单元格。

(8) 对行内的每个单元格重复第 (5) 步至第 (7) 步。

(9) 输入 `</tr>` 结束行。

(10) 为所在表格部分内的每个行重复第 (4) 步至第 (9) 步。

(11) 如果在第 (3) 步中开始了一个表格部分，就根据需要使用 `</thead>`、`</tbody>` 或 `</tfoot>` 结束这个部分。

(12) 为每个表格部分重复第 (3) 步至第 (11) 步。注意，一个表格只能有一个 `thead` 和 `tfoot`，但可以有多多个 `tbody` 元素。

(13) 输入 `</table>` 以结束表格。

如图 18.1.4 所示，在默认情况下，表格被压得很扁。通过应用一些 CSS（如图 18.1.5 所示），可以为单元格添加一些空间让它们扩大一些（使用 `padding`），添加边框以指示单元格的边界（使用 `border`），还可以对文本进行格式化。这些样式都有助于更好地理解表格，如图 18.1.6 所示。

提示 如果包含了 `caption` 元素，那么它必须是 `table` 中的第一个元素，如图 18.1.1 所示。（`caption` 元素可以包含 `p` 和其他文本元素。）

提示 如果包含了 `thead` 或 `tfoot`，则必须包含 `tbody`。`tbody` 不能位于 `thead` 之前。一个 `table` 只能拥有一个 `thead` 和一个 `tfoot`，但可以有多多个 `tbody` 元素。

```

body {
    font: 100% "Courier New", Courier,
        → monospace;
}

table {
    border-collapse: collapse;
}

caption {
    font-size: .8125em;
    font-weight: bold;
    margin-bottom: .5em;
}

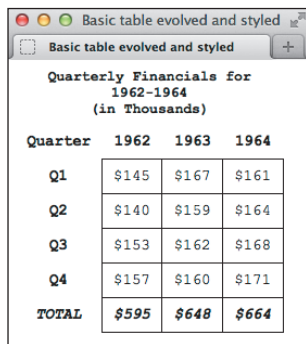
th,
td {
    font-size: .875em;
    padding: .5em .75em;
}

td {
    border: 1px solid #000;
}

tfoot {
    font-style: italic;
    font-weight: bold;
}

```

图 18.1.5 这个简单的样式表为每个数据单元格添加了 border，为标题单元格和数据单元格都添加了 padding，还对表格 caption 和内容进行了格式化。如果不对 table 定义 border-collapse: collapse，每个 td 的边框和相邻 td 的边框之间就会出现一些空隙（默认值为 border-collapse: separate;）。还可以像 18.2 节所示的那样，对 th 元素应用边框



Quarter	1962	1963	1964
Q1	\$145	\$167	\$161
Q2	\$140	\$159	\$164
Q3	\$153	\$162	\$168
Q4	\$157	\$160	\$171
TOTAL	\$595	\$648	\$664

图 18.1.6 现在，表格既有列标题，也有行标题，还有一个显示各列数据之和的行（位于 tfoot 元素中）。HTML 还没有聪明到为你计算这些东西的数量，因此确保行列数都正确

提示 如果 table 是嵌套在 figure 元素内除 figcaption 以外的唯一元素，则可以省略 caption，使用 figcaption 对表格进行描述（参见 4.4 节）。注意，不要在 table 中嵌套 figcaption，而应跟往常一样将 figcaption 放在 figure 中。

提示 还可以在样式表中对 table、td 或 th 元素设置 background、width 等属性，尽管没有在 CSS 示例中进行演示，如图 18.1.4 所示。总之，大多数用于其他 HTML 元素的文本格式和其他格式也可以应用于表格（参见 18.2 节的另一个例子）。你可能会发现，不同的浏览器显示的样式稍有差异，尤其是 Internet Explorer。

提示 可以通过 scope 属性指定 th 为一组列的标题（使用 scope="colgroup"），或者为一组行的标题（使用 scope="rowgroup"）。对于后者，参见下一节的示例。

18.2 让单元格跨越多列或多行

可以通过 colspan 和 rowspan 属性让 th 或 td 跨越一个以上的列或行。对该属性指定的数值表示的是跨越的单元格的数量，如图 18.2.1 和图 18.2.2 所示。

1. 让单元格跨越两个或两个以上列的步骤

(1) 在需要定义跨越一个以上的列的单元格的地方，如果为标题单元格，输入 <th 后加一个空格，否则输入 <td 后加一个空格。

(2) 输入 colspan="n">，这里的 n 是单元格要跨越的列数。

(3) 输入单元格的内容。

(4) 根据前面的内容，输入 </th> 或者 </td>。

(5) 根据 18.1 节中的介绍, 完成表格的其余部分。如果创建了一个跨越两列的单元格, 在该行就应该少定义一个单元格; 如果创建了一个跨越三列的单元格, 在该行就应该少定义两个单元格, 以此类推。

```
...
<body>

<table>
  <caption>TV Schedule</caption>
  <thead> <!-- 表格头部 -->
    <tr>
      <th scope="rowgroup">Time</th>
      <th scope="col">Mon</th>
      <th scope="col">Tue</th>
      <th scope="col">Wed</th>
    </tr>
  </thead>
  <tbody> <!-- 表格主体 -->
    <tr>
      <th scope="row">8 pm</th>
      <td>Staring Contest</td>
      <td colspan="2">Celebrity Hoedown</td>
    </tr>
    <tr>
      <th scope="row">9 pm</th>
      <td>Hardy, Har, Har</td>
      <td>What's for Lunch?</td>
      <td rowspan="2">Screamfest Movie of the Weak</td>
    </tr>
    <tr>
      <th scope="row">10 pm</th>
      <td>Healers, Wheelers & Dealers</td>
      <td>It's a Crime</td>
    </tr>
  </tbody>
</table>

</body>
</html>
```

图 18.2.1 通过在包含 *Celebrity Hoedown* 演出的 td 上应用 `colspan="2"`, 指示该演出的上映时间包含周二和周三晚上 8 点这两个时间。类似地, 在包含 *Screamfest Movie of the Weak* 的 td 上添加了 `rowspan="2"`, 因为该演出持续两个小时。同时, 注意 Time 这个 th 设置了 `scope="rowgroup"`, 因为它是它正下方的每个行标题组的标题

TIME	MON	TUE	WED
8 pm	Staring Contest	Celebrity Hoedown	
9 pm	Hardy, Har, Har	What's for Lunch?	Screamfest Movie of the Weak
10 pm	Healers, Wheelers & Dealers	It's a Crime	

图 18.2.2 仅仅通过查看代码, 很难判断 `colspan` 和 `rowspan` 对表格的影响, 但在浏览器中查看就清楚多了。这个表格还用 CSS 添加了一些样式, 完整的样式表见本书的配套网站 www.htmlcssvqs.com/8ed/18

2. 让单元格跨越两个或两个以上行的步骤

(1) 在需要定义跨越一个以上的行的单元格的地方, 如果为标题单元格, 输入 `<th>` 后加一个空格, 否则输入 `<td>` 后加一个空格。

(2) 输入 `rowspan="n">`, 这里的 *n* 是单元格要跨越的行数。

(3) 输入单元格的内容。

(4) 根据前面的内容, 输入 `</th>` 或者 `</td>`。

(5) 根据 18.1 节中的介绍, 完成表格的其余部分。如果创建了一个 `rowspan` 等于 2 的单元格, 就不需要定义下一行中该单元格对应的单元格了; 如果创建了一个 `rowspan` 等于 3 的单元格, 就不需要定义下面两行中该单元格对应的单元格了, 以此类推。

提示 表格中的每一行都应具有相同的单元格数量。跨越多列的单元格应算做多个单元格, 它的 `colspan` 属性值为多少, 就算做多少个单元格。

提示 表格中的每一列都应具有相同的单元格数量。跨越多行的单元格应算做多个单元格, 它的 `rowspan` 属性值为多少, 就算做多少个单元格。

本章内容

- ❑ 加载外部脚本
- ❑ 添加嵌入脚本
- ❑ JavaScript 事件

HTML 定义网页的内容，CSS 定义网页的表现，JavaScript 则定义特殊的行为。建立网站不可能脱离 HTML（如果能让网站看起来很吸引人，则离不开 CSS），但 JavaScript 并不是必需的。在大多数情况下，JavaScript 的特性都是用于增强访问者体验的——它们在由 HTML 和 CSS 构建的核心体验的基础上进行增强。（参见本书前言部分“渐进增强：一种最佳实践”）。

通过编写简单的 JavaScript 程序，可以显示和隐藏内容；通过编写复杂一些的程序，可以加载数据并动态地更新页面。可以操作定制的 HTML5 `audio` 和 `video` 元素控件，使用 HTML5 的 `canvas` 元素创建游戏。可以利用地理定位，根据访问者所在的位置定制其体验；可以让访问者通过将文件拖放到浏览器窗口进行上传（Dropbox 的网站就是一个这样做的例子）。可以利用 JavaScript 和一些强大的 HTML5 特性及相关技术构建成熟的 Web 应用程序（这些内容都是高级特性，因此不在本书讲解范围之内）。

如你所见，JavaScript 的功能非常强大，

而它的使用也呈现了爆炸式的增长。jQuery（jquery.com）等 JavaScript 库确保了为页面添加简单交互和复杂行为的过程变得容易了许多。还有很多跟 jQuery 功能类似的库，但在这些库中，jQuery 是用得最多的一个，这主要是因为初学者很容易上手，同时它有很好的在线文档和大型社区支持。除了 jQuery 以外，还有各式各样的 JavaScript 可用于帮你构建大型 Web 应用。使用 Node.js（<http://nodejs.org>）甚至可以用 JavaScript 构建 Web 服务器。

浏览器厂商花费了大量的精力确保其浏览器处理 JavaScript 的速度较几年前的版本有了显著提升。JavaScript 也可以在平板电脑和现代移动浏览器中运行，不过出于性能方面的原因，可能需要为这些设备考虑在页面中加载的脚本的大小。

不过 JavaScript 本身是一个独立、庞大的主题，因此我们不会在本书讲解这门语言。我仍会解释如何将创建好的脚本插入到 HTML 文档中去，同时给出一些关于如何在插入脚本时尽量降低其对页面影响的建议，此外还会提供对事件处理程序的概览。

当你熟悉 HTML 和 CSS 后，我建议 you 开始学习 JavaScript（以及 jQuery）。Marijn Haverbeke 的 *Eloquent JavaScript*（中文版《JavaScript 编程精解》，机械工业出版社，

2012) 是开始学习 JavaScript 的好资料。该书的原始版本是免费的, 位于 <http://eloquentjavascript.net>, 此外还有修订后的纸质版。当你对 JavaScript 语言有一定了解后, 可以看看 Ivo Wetzel 和 Zhang Yi Jiang 的 JavaScript Garden (<http://bonsaiden.github.io/JavaScript-Garden/>), 这是个学习 JavaScript 语言精华和怪异特性的免费、简洁的资源。

JavaScript 要比 HTML 和 CSS 更复杂一些, 因此, 如果你发现自己要花费较长的时间学习, 也不要感到气馁。对任何事物来说, 你操作越多, 就越熟练。

19.1 加载外部脚本

脚本主要有两种类型, 一种是从外部文件 (使用纯文本格式) 加载的脚本, 另一种是嵌入在页面中的脚本 (这一种将在下一节讲解)。这同外部样式表和嵌入样式表的概念是类似的。

同为页面添加样式表一样, 从外部文件加载脚本通常比在 HTML 中嵌入脚本要好一些 (如图 19.1.1 所示)。这样做的好处也是类似的, 即可以在需要某一脚本的每个页面加载同一个 JavaScript 文件。需要对脚本进行修改时, 就可以仅编辑一个脚本, 而不是在各个单独的 HTML 页面更新相似的脚本。

无论是加载外部脚本还是嵌入脚本, 均使用 `script` (脚本) 元素。

加载外部脚本的方法

输入 `<script src="script.js"></script>`, 这里的 `script.js` 是外部脚本在服务器上的位置及文件名。应该尽可能地将脚本元素放在 `</body>` 结束标签之前 (参见图 19.1.1), 而不是放在文档的 `head` 元素里 (参见图 19.1.2)。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>Loading an External Script</title>
  <link rel="stylesheet"
    → href="css/global.css" />
</head>
<body>
... 所有的HTML内容写在这里 ...

<script src="behavior.js"></script>
</body>
</html>
```

图 19.1.1 `script` 元素的 `src` 属性引用脚本的 URL。在大多数情况下, 最好在页面的最末尾加载脚本, 即 `</body>` 结束标签的前面。也可以在页面的 `head` 元素中加载脚本, 如图 19.1.2 所示, 但这样做会影响页面显示的速率。更多信息参见补充材料 “脚本和性能的最佳实践”

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>Loading an External Script</title>
  <!-- 在加载任何JS文件之前加载样式表 -->
  <link rel="stylesheet" href="css/
    → global.css" />
  <script src="behavior.js"></script>
</head>
<body>
... 所有的HTML内容写在这里 ...
</body>
</html>
```

图 19.1.2 在这个例子中, 脚本是在 `head` 中加载的。它位于 `link` 元素的后面, 因此不会影响 CSS 文件先于脚本开始加载。关于为什么要尽可能地避免在 `head` 中加载脚本, 参见 “脚本和性能的最佳实践”

提示 为了保持组织文件良好，通常将 JavaScript 文件放在一个子文件夹中（常用的名称包括 js、scripts 等），参见 2.6 节。src 属性需要反映这一点，就像指向某一资源的其他 URL 一样。例如，如果图 19.1.1 中的文件位于名为 assets/js/ 的文件夹内，则应输入 `<script src="assets/js/behavior.js"></script>`。（这只是个示例，还有其他表示 URL 的方式，参见 1.7 节。）

提示 图 19.1.3 显示了一段 JavaScript 示例。由于 JavaScript 是纯文本，因此你可以在创建 HTML 和 CSS 的同一编辑器中编写 JavaScript。如果这个例子保存为 behavior.js，它就会在图 19.1.1 和图 19.1.2 所示的页面中加载。

```
/*
  当用户选择 Other 单选按钮时，textarea 会变为可用的，光标也会显示在其中，用户可以马上输入。选择其余
  两个单选按钮中的任意一个，则会禁用 textarea
*/

(function (window, document) {
    'use strict';

    var choices = document.getElementById('choices'),
        textarea = document.getElementById('other-description');

    if (!choices || !textarea) {
        return;
    }

    // 默认禁用textarea
    textarea.disabled = true;

    // 为单选按钮添加行为
    choices.onclick = function(e) {
        var target,
            e;

        if (!e) {
            e = window.event;
        }

        target = e.target || e.srcElement;

        // 根据所选的单选按钮，切换textarea的状态
        if (target.getAttribute('type') === 'radio') {
            if (target.id !== 'other') {
                textarea.disabled = true;
            } else {
                textarea.disabled = false;
                textarea.focus();
            }
        }
    };
})(window, document);
```

图 19.1.3 如你所见，JavaScript 与 HTML 和 CSS 相当不一样！这段脚本与 16.16 节引用的 toggle-textarea.js 几乎是一样的。本书配套网站上的例子包含了额外的注释，解释了代码的工作原理（参见 www.htmlcssvqs.com/8ed/19）。即便如此，我也不会具体地讲解代码的方方面面，如果你刚开始接触 JavaScript，你会发现不懂的地方很多。我提供这个例子，主要是为了让你对这门语言有个大致的感受