

提示 在 HTML5 之前有 acronym (首字母缩写词) 元素, 但设计和开发人员常常分不清楚缩写词和首字母缩写词, 因此 HTML5 废除了 acronym 元素, 让 abbr 适用于所有的场合。

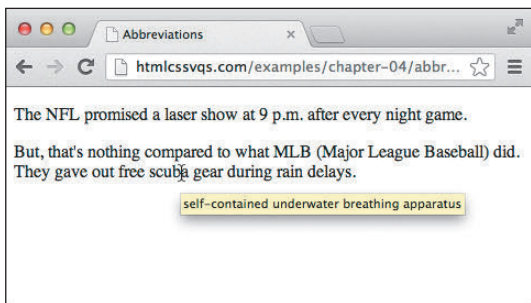


图 4.8.3 当访问者将鼠标移至 abbr 上, 该元素 title 属性的内容就会显示在一个提示框里 (这里还显示, 在默认情况下, Chrome 等一些浏览器不会让带有 title 属性的缩写词跟普通文本有任何显示上的差别)

4.9 定义术语

在印刷物中, 首次定义术语通常会对其添加区别于其他文本的格式 (英文通常为斜体, 汉语通常为黑体或者楷体), 在其后提到术语时则不再使用特殊格式。

在 HTML 中定义术语时, 可以使用 dfn 元素对其作语义上的区分。仅用 dfn 包围要定义的术语, 而不是包围定义, 如图 4.9.1 所示。同印刷物的惯例一样, 后续使用术语时不再需要使用 dfn 对其进行标记, 因为不再需要对其进行定义。(在 HTML 中, 定义术语的地方称为“术语的定义实例”。)

标记术语的定义实例

- (1) 输入 <dfn>。
- (2) 输入要定义的术语。
- (3) 输入 </dfn>。

```
...
<body>

<p>The contestant was asked to spell
→ "pleonasm." She requested the definition
→ and was told that <dfn>pleonasm</dfn>
→ means "a redundant word or expression"
→ (Ref: <cite><a href="http://dictionary.
→ reference.com/browse/pleonasm" rel=
→ "external">dictionary.com</a></cite>).</p>

</body>
</html>
```

图 4.9.1 注意, 在这个例子中, 尽管 pleonasm 出现了两次, 但只对第二个用 dfn 进行了标记, 因为那时才定义这个术语 (即定义实例)。类似地, 如果我在文档后面用到 pleonasm, 也不会用 dfn 标记, 因为我已经定义过它了。尽管浏览器在默认情况下会为 dfn 文本加上与普通文本不同的样式 (参见图 4.9.2), 但重要的是术语使用了特殊的标记。同时, 不必在每次使用 dfn 时都使用 cite 元素, 该元素只在引用参考源的时候使用

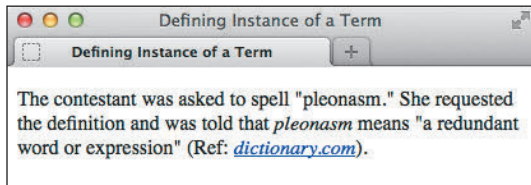


图 4.9.2 通常, dfn 元素默认以斜体显示, 与 cite 一样

术语及其定义应位置相近

由 dfn 标记的术语与其定义的距离远近相当重要。如 HTML5 规范所述: “如果一个段落、描述列表或区块是某 dfn 元素距离最近的祖先, 那么该段落、描述列表或区块必须包含该术语的定义。”简言之, dfn 元素及其定义必须挨在一起, 否则便是错误的用法。图 4.9.1 和第四条提示中的例子都遵循这一规则——dfn 及其定义位于同一个段落。

提示 还可以在描述列表（dl 元素）中使用 dfn，参见第 15 章。

提示 仅在定义术语时使用 dfn，而不能为了让文字以斜体显示就使用该元素。使用 CSS 可以将任何文字变为斜体（参见 10.4 节）。

提示 dfn 可以在适当的情况下包住其他的短语元素，如 abbr。例如，`<p>A <dfn><abbr title="Junior">Jr.</abbr></dfn> is a son with the same full name as his father.</p>`。

提示 如果在 dfn 中添加可选的 title 属性，其值应与 dfn 术语一致。正如在上一条提示中看到的，如果只在 dfn 里嵌套一个单独的 abbr，dfn 本身没有文本，那么可选的 title 只能出现在 abbr 里。

4.10 创建上标和下标

比主体文本稍高或稍低的字母或数字分别称为上标和下标（参见图 4.10.1）。HTML 包含用来定义这两种文本的元素。常见的上标包括商标符号、指数和脚注编号等（参见图 4.10.2）；常见的下标包括化学符号等。

创建上标和下标的步骤

- (1) 输入 `<sub>` 创建下标，或输入 `<sup>` 创建上标。
- (2) 输入要出现在下标或上标里的字符或符号。
- (3) 根据第 (1) 步中使用的元素，输入 `</sub>` 或 `</sup>` 结束该元素。

提示 大多数浏览器会自动将上标或下标文字的字号减少几磅。

```
...
<body>

<article>
  <h1>Famous Catalans</h1>
  <p>... Actually, Pablo Casals' real
    → name was <i>Pau</i> Casals, Pau
    → being the Catalan equivalent of Pablo
    → <a href="#footnote-1" title="Read
    → footnote 1"><sup>1</sup></a>.</p>

  <p>... Pau Casals is remembered in this
    → country for his empassioned speech
    → against nuclear proliferation at the
    → United Nations <a href="#footnote-2"
    → title="Read footnote 2"><sup>2</sup></a>.</p>

  <footer>
    <p id="footnote-1"><sup>1</sup></p>It
    → means Paul in English.</p>
    <p id="footnote-2"><sup>2</sup></p>In
    → 1963, I believe.</p>
  </footer>
</article>

</body>
</html>
```

图 4.10.1 sup 元素的一种用法就是表示脚注编号。根据从属关系，我将脚注放在 article 的 footer 里，而不是整个页面的 footer 里。我还为文章中每个脚注编号创建了链接，指向 footer 内对应的脚注，从而让访问者更容易找到它们。同时，注意链接中的 title 属性也提供了一些提示

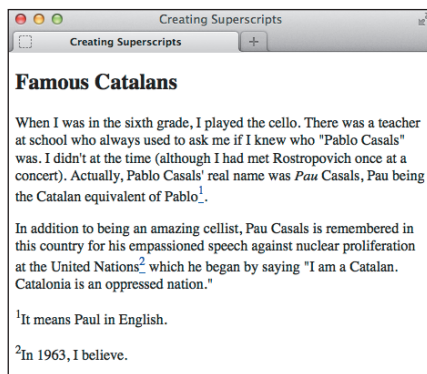


图 4.10.2 sup 元素的位置比同一行的文本要高。不过，不好的一点是，它们会扰乱行间距（参见最后一条提示）

提示 上标是对某些外语缩写词进行格式化的理想方式，例如，法语中用 M^{lle} 表示 Mademoiselle（小姐），西班牙语中用 3^a 表示 tercera（第三）。此外，一些数字形式也要用到上标，如 2nd、5th。

提示 下标适用于化学分子式（如 H₂O）。例如，`<p>I'm parched. Could I please have a glass of H<sub>2</sub>O?</p>.`

提示 上标和下标字符会轻微地扰乱行与行之间均匀的间距。例如，在图 4.10.2 中，第一段第 4 行和第 5 行之间的距离以及第二段第 2 行和第 3 行之间的距离比其他行间距要大一些。不过，可以使用一些 CSS 解决这个问题。解决的办法参见“修复使用 sub 和 sup 时的行间距问题”。

修复使用 sub 和 sup 时的行间距问题

sub 和 sup 元素会轻微地增大行高。幸好，用一点 CSS 就可以修复这个问题。下面的代码来自 Nicolas Gallagher 和 Jonathan Neal 出色的 normalize.css（<http://necolas.github.com/normalize.css/>）。下面的方法并不是他们发明的，他们借用了 <https://gist.github.com/413930> 里的代码并去掉了注释。上面这个 GitHub 的链接里包含了对这一段 CSS 的详细解释，建议你去看一看。

同时，你可以在自己的项目里使用 normalize.css，推荐你下载这个文件。该文件可以帮你建立一个跨浏览器的统一基准样式表，其文档也很详尽（参见 11.4 节）。

```
/*
 * 在所有浏览器中防止sub和sup影响line-height
 * gist.github.com/413930
 */
sub,
sup {
  font-size: 75%;
  line-height: 0;
  position: relative;
  vertical-align: baseline;
}
sup {
  top: -0.5em;
}
sub {
  bottom: -0.25em;
}
```

你可能还需要根据内容的字号对这个 CSS 做一些调整，使各行行高保持一致，但上面的代码至少已经为你建立了很好的起点。关于如何创建样式表并将 CSS 添加到网站，参见第 8 章。

4.11 添加作者联系信息

你或许以为 `address` 元素是用于标记通讯地址的, 但其实并非如此(有一种例外的情况, 参见第一条提示)。实际上, 没有专门用于标记通讯地址的 HTML 元素。

实际上, `address` 元素是用以定义与 HTML 页面或页面一部分(如一篇报告或新文章)有关的作者、相关人士或组织的联系信息, 通常位于页面底部或相关部分内(参见图 4.11.1 和图 4.11.2)。至于 `address` 具体表示的是哪一种信息, 取决于该元素出现的位置。下面的第 (1) 步中对每一种场景都做了描述。

提供作者联系信息

(1) 如果要为一个 `article` 提供作者联系信息, 就将光标放在该元素内(参见图 4.11.1 中第一个例子)。如果要提供整个页面的作者联系信息, 就将光标放在 `body` 中(更常见的做法是放在页面级的 `footer` 里), 参见图 4.11.1 中第二个例子。

(2) 输入 `<address>`。

(3) 输入作者的电子邮件地址、指向联系信息页的链接等。

(4) 输入 `</address>`。

提示 大多数时候, 联系信息的形式是作者的电子邮件地址或指向联系信息页的链接。联系信息也有可能是作者的通讯地址, 这时将地址用 `address` 标记就是有效的。但是用 `address` 标记公司网站“联系我们”页面中的办公地点, 则是错误的用法。4.16 节中提供了一种标记通讯地址(街道地址)的方法。

```
...
<body>
<main role="main">
<article>
  <h1>Museum Opens on the Waterfront</h1>
  <p>The new art museum not only
  → introduces a range of contemporary
  → works to the city, it's part of
  → larger development effort on the
  → waterfront.</p>

  ... [故事内容的其余部分] ...

  <!-- 文章的页脚, 其中包含文章的地址信息 -->
  <footer>
    <p>Tracey Wong has written for
    → <cite>The Paper of Papers</cite>
    → since receiving her MFA in art
    → history three years ago.</p>
    <address>
      Email her at <a href="mailto:
      → traceyw@thepaperofpapers.com">
      → traceyw@thepaperofpapers.com
      → </a>.
    </address>
  </footer>
</article>
</main>

<!-- 页面的页脚, 其中包含整个页面的地址信息 -->
<footer role="contentinfo">
  <p><small>&copy; 2014 The Paper of
  → Papers, Inc.</small></p>
  <address>
    Have a question or comment about the
    → site? <a href="site-feedback.html">
    → Contact our web team</a>.
  </address>
</footer>
</body>
</html>
```

图 4.11.1 这个页面有两个 `address` 元素: 一个用于 `article` 的作者, 另一个位于页面级的 `footer` 里, 用于整个页面的维护者。注意 `article` 的 `address` 只包含联系信息。尽管 `article` 的 `footer` 里也有关于 Tracey Wong 的背景信息, 但这些信息是位于 `address` 元素外面的



图 4.11.2 address 元素中的文字默认以斜体显示（“The Paper of Papers”也是斜体，不过这是由于它包含在 cite 元素中，参见本章 4.5 节）

提示 如果 address 嵌套在 article 里，则属于其所在的最近的 article 元素；否则属于页面的 body。说明整个页面的作者的联系信息时，通常将 address 放在 footer 元素里，如图 4.11.1 中第二个 address 的实例。

提示 article 里的 address 提供的是该 article 作者的联系信息（图 4.11.1），而不是嵌套在该 article 里的其他任何 article（如用户评论）的作者的联系信息。

提示 address 只能包含作者的联系信息，不能包括其他内容，如文档或文章的最后修改时间（图 4.11.1）。此外，HTML5 禁止在 address 里包含以下元素：h1 ~ h6、article、address、aside、footer、header、hgroup、nav 和 section。

提示 关于 article 的 footer 元素，参见第 3 章。

4.12 标注编辑和不再准确的文本

有时可能需要将在前一个版本之后对页面内容的编辑标出来，或者对不再准确、不再相关的文本进行标记。有两种用于标注编辑的元素：代表添加内容的 ins 元素和标记已删除内容的 del 元素（参见图 4.12.1 至图 4.12.4）。这两个元素既可以单独使用，也可以一起使用。

```
...
<body>

<h1>Charitable Gifts Wishlist</h1>

<p>Please consider donating one or more
→ of the following items to the village's
→ community center:</p>

<ul>
  <li><del>2 desks</del></li>
  <li>1 chalkboard</li>
  <li><del>4 solar-powered tablets
  → </del></li>
  <li><ins>1 bicycle</ins></li>
</ul>

</body>
</html>
```

图 4.12.1 在这个礼品清单上一次发布之后，又增加了一个条目 (bicycle)，同时根据 del 元素的标注，移除了一些已购买的条目。使用 ins 的时候不一定要使用 del，反之亦然。浏览器通常会让它们看起来与普通文本不一样，参见图 4.12.2

同时，s 元素用以标注不再准确或不再相关的内容（一般不用于标注编辑内容），如图 4.12.5 和图 4.12.6 所示。



图 4.12.2 浏览器通常对已删除的文本加上删除线，对插入的文本加上下划线。可以用 CSS 修改这些样式



图 4.12.4 与前面类似，浏览器会将删除或插入的内容标出

```
...
<body>

<h1>Charitable Gifts Wishlist</h1>
<del>
  <p>Please consider donating one or more
    → of the following items to the village's
    → community center:</p>
</del>

<ins>
  <p>Please note that all gifts have been
    → purchased.</p>
  <p>Thank you <em>so much</em> for your
    → generous donations!</p>
</ins>

<del>
  <ul>
    <li><del>2 desks</del></li>
    <li>1 chalkboard</li>
    <li><del>4 solar-powered tablets
      → </del></li>
    <li><ins>1 bicycle</ins></li>
  </ul>
</del>

</body>
</html>
```

图 4.12.3 del 和 ins 是少有的既可以包围短语内容（HTML5 之前称“行内元素”）又可以包围块级内容的元素，如下面的代码所示

```
...
<body>

<h1>Today's Showtimes</h1>
<p>Tickets are available for the following
  → times today:</p>

<ol>
  <li><ins>2 p.m. (this show just added!)
    → </ins></li>
  <li><s>5 p.m.</s> SOLD OUT</li>
  <li><s>8:30 p.m.</s> SOLD OUT</li>
</ol>

</body>
</html>
```

图 4.12.5 这个例子展示了一个关于演出时间的有序列表（ol 元素）。与剩余票数不再相关的时段都用 s 元素进行了标记。可以对任何短语使用 s，而不仅限于列表项（li 元素）里的文本。不过，不要像 del 和 ins 那样用 s 标记整个段落或其他块级元素

1. 标记新插入文本

- (1) 输入 <ins>。
- (2) 输入新内容。
- (3) 输入 </ins>。

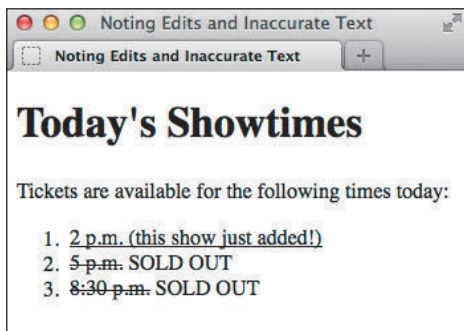


图 4.12.6 默认情况下，浏览器会对 s 元素添加删除线

2. 标记已删除文本

(1) 将光标放在待标记为已删除的文本或元素之前。

(2) 输入 `<del>`。

(3) 将光标放在待标记为已删除的文本或元素之后。

(4) 输入 `</del>`。

3. 标记不再准确或不再相关的文本

(1) 将光标放在希望标记为不再准确或不再相关的文本的前面。

(2) 输入 `<s>`。

(3) 将光标放在希望标记的文本的后面。

(4) 输入 `</s>`。

提示 `del` 和 `ins` 都支持两个属性：`cite` 和 `datetime`。`cite` 属性（区别于 `cite` 元素）用于提供一个 URL，指向说明编辑原因的页面。例如，`<ins cite="http://www.movienews.com/ticket-demand-high.html">2 p.m. (this show just added!)</ins>`。`datetime` 属性提供编辑的时间。（关于 `datetime` 可接受的格式，参见 4.7 节。）浏览器不会将这两个属性的值显示出来，因此它们的使用并不广泛。不过，应该尽量包含它们，从而为内容提供一些背景信息。它们的值可以通过 JavaScript 或分析页面的程序提取出来。

提示 如果需要向访问者展示内容变化情况，就可以使用 `del` 和 `ins`。例如，经常可以看见一些 Web 设计与开发教程使用它们表示初次发布以后新更新的信息，这样可以保持原始信息的完整性。博客、新闻网站等也可以这样做。

提示 用 `ins` 标记的文本通常会显示一条下划线（图 4.12.2）。由于链接通常也以下划线表示（可能你的网站不是这样，但其他很多网站都是这样），这可能会让访问者感到困惑。可以使用样式表改变插入的段落（或链接）的外观（参见第 10 章）。

提示 用 `del` 标记的文本通常会显示一条删除线（图 4.12.2）。为什么不直接清除这些文字呢？这取决于你认为是否需要将删除的内容显示出来。加上删除线以后，用户就很容易看出修改了什么。（同时，屏幕阅读器可以读出被删除的内容，不过目前对这一特性的支持还不充分。）

提示 仅在具有语义价值的时候使用 `del`、`ins` 和 `s`。如果只是出于装饰的原因要给文字添加下划线或删除线，可以用 CSS 实现这些效果（参见 10.6 节）。

提示 HTML5 指出：“s 元素不适用于指示文档的编辑，要标记文档中一块已移除的文本，应使用 `del` 元素。”有时，这之间的差异是很微妙的，只能由你决定哪种选择更符合内容的语义。

4.13 标记代码

如果你的内容包含代码示例或文件名，就可以使用 `code` 元素（参见图 4.13.1 和图 4.13.2）。

```
...
<body>

<p>The showPhoto() function
→ displays the full-size photo of the
→ thumbnail in our &lt;ul id=
→ "thumbnail"&gt; carousel list.</p>

<p>This CSS shorthand example applies a
→ margin to all sides of paragraphs: <code>p
→ { margin: 1.25em; }</code>. Take a look
→ at <code>base.css</code> to see more
→ examples.</p>

</body>
</html>
```

图 4.13.1 `code` 元素表示其中的文本是代码或文件名。如果你的代码需要显示 `<` 或 `>`，应分别使用 `<` 和 `>`。这里第二个 `code` 示例演示了这一点。如果真的用了 `<` 和 `>`，浏览器会将这些代码当做 HTML 元素处理，而不是当做文本处理

示例展示了句子中的代码。要显示单独的一块代码（位于句子以外），可以用 `pre`

元素包住 `code` 元素以维持其格式（具体的示例参见 4.14 节）。

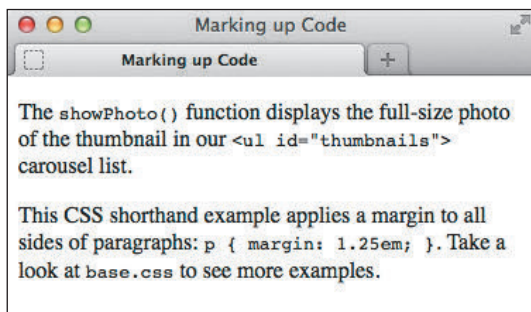


图 4.13.2 `code` 元素的文本看起来像代码，因为其默认字体为等宽字体

标记代码或文件名的步骤

- (1) 输入 `<code>`。
- (2) 输入代码或文件名。
- (3) 输入 `</code>`。

提示 可以通过 CSS 改变 `code` 默认的等宽字体样式（图 4.13.2），参见第 10 章。

提示 关于字符实体（图 4.13.1），参见 1.4 节。

其他计算机相关元素：`kbd`、`samp` 和 `var`

`kbd`、`samp` 和 `var` 元素极少使用，不过你可能会在内容中用到它们。下面对它们作简要说明。

● `kbd` 元素

使用 `kbd` 标记用户输入指示。

```
<p>To log into the demo:</p>
<ol>
  <li>Type <kbd>tryDemo</kbd> in the User Name field</li>
  <li><code><kbd>TAB</kbd> to the Password field and type <kbd>demoPass</kbd></li>
  <li>Hit <kbd>RETURN</kbd> or <kbd>ENTER</kbd></li>
</ol>
```

与 `code` 一样，`kbd` 默认以等宽字体显示。

◎ samp 元素

samp 元素用于指示程序或系统的示例输出。

```
<p>Once the payment went through, the site returned a message reading,<samp>Thanks  
→ for your order!</samp></p>
```

samp 也默认以等宽字体显示。

◎ var 元素

var 元素表示变量或占位符的值。

```
<p>Einstein is best known for <var>E</var>=<var>m</var><var>c</var><sup>2</sup></p>
```

var 也可以作为内容中占位符的值,例如,在填词游戏的答题纸上可以放入 <var>adjective</var>, <var>verb</var>。

var 默认以斜体显示。

需要注意的是,可以在 HTML5 页面中使用 math 等 MathML 元素表示高级的数学相关的标记。更多信息参见 <http://dev.w3.org/html5/spec-author-view/mathml.html>。

4.14 使用预格式化的文本

通常,浏览器会将所有额外的回车和空格压缩,并根据窗口的大小自动换行。预格式化的文本可以保持文本固有的换行和空格。它是计算机代码示例的理想元素,如图 4.14.1 所示,不过你也可以将它用于文本(比如,ASCII 艺术^①)。

使用预格式化文本的步骤

- (1) 输入 <pre>。
- (2) 输入或复制希望以原样显示的文本,包括所需要的空格、回车和换行。除非是代码,不要用任何 HTML(如 p 元素)标记这些文本。
- (3) 输入 </pre>。

```
...
<body>

<p>Add this to your style sheet if you want
→ to display a dotted border underneath the
→ <code>abbr</code> element whenever it has
→ a <code>title</code> attribute.</p>

<pre>
  <code>
    abbr[title] {
      border-bottom: 1px dotted #000;
    }
  </code>
</pre>

</body>
</html>
```

图 4.14.1 对于包含重要的空格和换行的文本(如这里显示的 CSS 代码),pre 元素是非常适合的。同时要注意 code 元素的使用,该元素可以标记 pre 外面的代码块或与代码有关的文本(更多细节参见 4.13 节)

① ASCII 艺术指的是用计算机字符(主要是 ASCII 字符)表示图片的一种艺术形式,通常要求使用等宽字体显示。

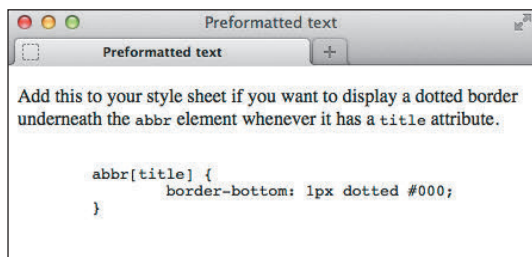


图 4.14.2 注意 pre 内容里的缩进和换行都被保留了

提示 预格式化的文本通常以等宽字体（如 Courier、Courier New 等）显示，参见图 4.14.2。可以使用 CSS 改变字体样式（参见第 10 章）。

提示 如果要显示包含 HTML 元素的内容（如教程中的代码示例），应将包围元素名称的 < 和 > 分别改为其对应的字符实体 < 和 >（有关示例参见 4.13 节）。否则，浏览器就会试着显示这些元素。

提示 一定要对页面进行验证，检查是否在 pre 中嵌套了 HTML 元素（参见 20.1 节）。

提示 不要将 pre 作为逃避以合适的语义标记内容和用 CSS 控制样式的快捷方式。例如，如果你想发布一篇在字处理软件中写好的文章，不要为了保留原来的格式，简单地将它复制、粘贴到 pre 里。相反，应该使用 p（以及其他相关的文本元素）标记内容，编写 CSS 控制页面的布局。

提示 同段落一样，pre 默认从新一行开始显示（图 4.14.2）。

pre 在表现方面的注意事项

注意，浏览器通常会对 pre 里面的内容关闭自动换行，因此，如果这些内容很宽，就会影响页面的布局，或产生横向滚动条。下面的 CSS 规则可以对 pre 的内容打开自动换行，但在 Internet Explorer 7 及以下版本中并不适用。（随着浏览器市场占有率的改变，在大多数情况下，这些情况都不再需要考虑了。）

```
pre {
  white-space: pre-wrap;
}
```

与之相关的一点提示是，在大多数情况下不推荐对 div 等元素使用 white-space: pre 以代替 pre，因为空格可能对这些内容（尤其是代码）的语义非常重要，而只有 pre 才能始终保留这些空格。（同时，如果用户在其浏览器中关闭了 CSS，格式就丢失了。）

我们将从第 7 章开始讲解 CSS，在第 10 章中讨论文本的格式化。

4.15 突出显示文本

我们都在某些时候用过荧光笔——也许是在复习考试，抑或是在审查合同。无论是哪种情况，荧光笔标记的都是一些与任务有关的关键词。

HTML5 使用新的 mark 元素实现同样的目的。试着将 mark 想象成荧光笔的语义化对照物。换句话说，重要的是对特定的词语进行标注，与它们如何显示无关。可以用 CSS 对 mark 元素里的文字应用样式（不应用样式也可以），但应仅在合适的情况下使用该元素。

无论何时使用 mark，该元素总是用于提起读者对特定文本片段的注意。下面是一些应用的例子。

- ❑ 对搜索结果页面或文章中的搜索词进行突出显示。当我们谈到 `mark` 时，这是最常见的应用场景。假设使用网站的搜索功能查找“solar panels”。搜索结果或每个结果文章可以使用 `<mark>solar panels</mark>`，从而在整篇文字中突出显示该词。
- ❑ 对于某段引述，如果作者在原来的格式中没有对其进行突出显示，可以用 `mark` 引起对它的注意（参见图 4.15.1 和图 4.15.2）。这与现实世界中的做法是类似的。
- ❑ 引起对代码片段的注意（参见图 4.15.3 和图 4.15.4）。

```
...
<body>

<p>So, I went back and read the instructions
→ myself to see what I'd done wrong. They
→ said:</p>

<blockquote>
  <p>Remove the tray from the box. Pierce
  → the overwrap several times with a
  → fork and cook on High for <mark>15
  → minutes</mark>, rotating it half way
  → through.</p>
</blockquote>

<p>I thought he'd told me <em>fifty</em>. No
→ wonder it exploded in my microwave.</p>

</body>
</html>
```

图 4.15.1 `mark` 最常见的使用场合是在搜索结果页面，不过，这里给出了另一种合理的用法。包装上的说明中没有突出显示“15 minutes”一词，但这个 HTML 的作者却使用 `mark` 突出显示了这个词

突出显示文本的步骤

- (1) 输入 `<mark>`。
- (2) 输入希望引起注意的字词。
- (3) 输入 `</mark>`。

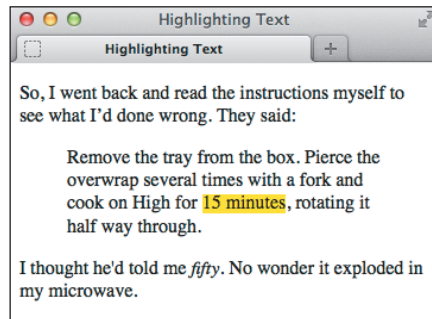


图 4.15.2 对 `mark` 原生支持的浏览器将对该元素的文字默认加上黄色背景。旧的浏览器不会这样做，但通过一条简单的样式表规则就可以让它们实现这种效果（参见提示）

```
...
<body>

<p>It's usually bad practice to use a class
→ name that explicitly describes how an
→ element should look, such as the
→ highlighted portion of CSS below:</p>

<pre>
  <code>
    <mark>.greenText</mark> {
      color: green;
    }
  </code>
</pre>

</body>
</html>
```

图 4.15.3 这个例子使用 `mark` 引起对代码片段的注意

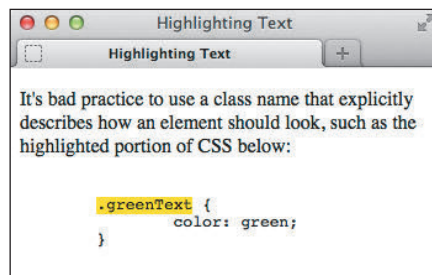


图 4.15.4 浏览器突出显示了由 `mark` 标记的代码

提示 mark 元素与 em（表示强调）或 strong（表示重要）不同，这两个元素在本章前面讲过了。

提示 由于 mark 是 HTML5 的新元素，因此旧的浏览器不会默认加上黄色背景。可以在样式表中加上 mark { background-color: yellow; } 让这些浏览器实现同样的效果。

提示 不要仅仅因为要给文字加上背景颜色或其他视觉上的考虑而使用 mark。如果只是要给一块文字添加样式，又没有合适的语义化元素，可以使用 span 元素（本章后面将讲到），并用 CSS 为其添加样式（span 元素可能需要添加一个 class）。

4.16 创建换行

浏览器会根据包含内容的块或窗口的宽度让文本自动换行。大多数情况下，让内容像这样充满整行是很合适的，但有时你希望手动地强制文字进行换行。可以使用 br 元素实现这一要求。

要确保使用 br 是最后的选择，因为该元素将表现样式带入了 HTML，而不是让所有的呈现样式都交由 CSS 控制。例如，不要使用 br 模拟段落之间的距离。相反，应该用 p 标记两个段落并通过 CSS 的 margin 属性规定两段之间的距离。

那么，什么时候该用 br 呢？实际上，对于诗歌、街道地址（图 4.16.1 和图 4.16.2）等应该紧挨着出现的短行，都适合用 br 元素。

```
...
<body>

<p>53 North Railway Street<br />
Okotoks, Alberta<br />
Canada T1Q 4H5</p>

<p>53 North Railway Street <br />Okotoks,
→ Alberta <br />Canada T1Q 4H5</p>

</body>
</html>
```

图 4.16.1 同样的地址出现了两次，不过出于演示的目的，对它们的编码有所不同。记住，浏览器总是会忽略代码里的回车，因此两个段落的显示效果是一样的（图 4.16.2）

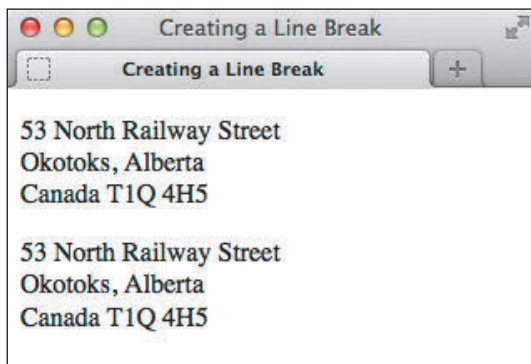


图 4.16.2 每个 br 元素强制让接下来的内容在新的一行显示。如果没有 br 元素，整个地址都会显示在同一行（除非浏览器窗口太窄导致内容换行）

插入换行的方法

在需要换行的地方输入
（或
）。没有单独的 br 结束标签，因为它是所谓的空元素，没有任何内容。

提示 在 HTML5 中，输入
 或
 都是有效的。

提示 可以使用 CSS 控制段落中的行间距（参见 10.7 节）以及段落之间的距离（参见 11.8 节）。

提示 hCard 微格式（<http://microformats.org/wiki/hcard>）是用于表示人、公司、组织和地点的人类和机器都可读的语义形式。可以使用微格式替代图 4.16.1 中表示街道地址的方式。

4.17 创建 span

同 div 一样, span 元素是没有任何语义的。不同的是, span 只适合包围字词或短语内容, 而 div 适合包含块级内容（参见 3.12 节）。

如果你想将下面列出的项目应用到某一小块内容, 而 HTML 又没有提供合适的语义化元素, 就可以使用 span。

- 属性, 如 class、dir、id、lang、title 等（图 4.17.1 和图 4.17.2）。

```
...
<body>

<h1 lang="es">La Casa Milà</h1>

<p>Gaudi's work was essentially useful.
→ <span lang="es">La Casa Milà</span> is
→ an apartment building and <em>real people
→ </em> live there.</p>

</body>
</html>
```

图 4.17.1 在这个例子中, 我想对一小块文字指定不同的语言, 但从句子的上下文看, 没有一个语义上适合“La Casa Milà”的 HTML 元素。段落之前包含“La Casa Milà”的 h1 在语义上是合适的, 因为这些文字就是后面内容的标题。因此对标题来说, 直接在 h1 中添加 lang 属性就可以了, 不必为了指定语言而将标题包在一个 span 中（lang 属性用于对元素的文本指定语言）

□ CSS 样式。

□ JavaScript 行为。

由于 span 没有任何语义, 因此应将它作为最后的选择, 仅在没有其他合适的元素时才使用它。



图 4.17.2 span 元素没有任何默认样式

添加 span 的步骤

- (1) 输入 。
- (2) 如果需要, 输入 id="name", 其中 name 用于唯一地标识 span 包含的内容。
- (3) 如果需要, 输入 class="name", 其中 name 是 span 所属类的名称。
- (4) 如果需要, 输入其他的属性（如 dir、lang、title 等）及其值。
- (5) 输入 > 结束开始标签。
- (6) 创建希望包含在 span 里的内容。
- (7) 输入 。

提示 span 没有任何默认格式（参见图 4.17.2），但就像其他 HTML 元素一样, 可以用 CSS 添加你自己的样式。

提示 可以对一个 span 元素同时添加 class 和 id 属性, 但通常只应用这两个中的一个（如果真要添加的话）。主要区别在于, class 用于一组元素, 而 id 用于标识页面中单独的、唯一的元素。

提示 在 HTML 没有提供合适的语义化元素时，微格式经常使用 `span` 为内容添加语义化类名，以填补语义上的空白。要了解更多信息，参见 <http://microformats.org>。

4.18 其他元素

本节讲的是 HTML 文本中可能用到的其他元素，但它们通常只在极少数情况下才会用到，或者浏览器对它们的支持还不完善（或兼而有之）。

1. u 元素

同 `b`、`i`、`s` 和 `small` 一样，HTML5 重新定义了 `u` 元素，使之不再是无语义的、用于表现的元素。以前，`u` 元素用来为文本添加下划线。现在，`u` 元素用于非文本注解（听起来确实有些迷惑）。HTML5 对它的定义为：

`u` 元素为一块文字添加明显的非文本注解，比如在中文中将文本标为专有名词（即中文的专名号^①），或者标明文本拼写有误。

下面是使用 `u` 标注拼错的词的例子：

```
<p>When they <u class="spelling">
→ recieved</u> the package, they put
→ it with <u class="spelling">there
→ </u> other ones with the intention
→ of opening them all later.</p>
```

`class` 完全是可选的，它的值（可以是你想填的任何内容）也不会内容中明显指出这是个拼写错误。不过，可以用它对拼错的词添加不同于普通文本的样式（`u` 默认仍以以下划线显示）。通过 `title` 属性可以为该元素包含的内容添加注释，如 “[sic]”（在一些语言中用于指示拼写错误的惯用符号）。

仅在 `cite`、`em`、`mark` 等其他元素语义上不适合的情况下使用 `u` 元素。同时，最好改变 `u` 文本的样式，以免与同样默认添加下划线的链接文本弄混（图 4.18.1）。

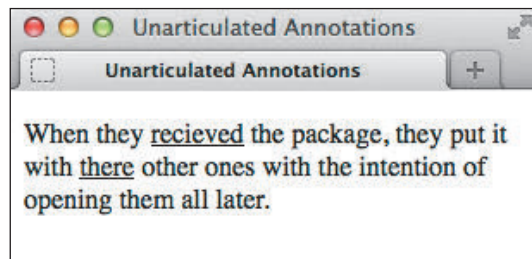


图 4.18.1 同链接一样，浏览器默认为 `u` 元素添加下划线。除非用 CSS 改变它们的样式，否则容易引起混淆

2. wbr 元素

HTML5 为 `br` 引入了一个相近的元素，称为 `wbr`。它代表“一个可换行处”。可以在一个较长的无间断短语（如 URL）中使用该元素，表示此处可以在必要的时候进行换行，从而让文本在有限的空间内更具可读性。因此，与 `br` 不同，`wbr` 不会强制换行，而是让浏览器知道哪里可以根据需要进行换行。

下面是一些例子：

```
<p>They liked to say, "FriendlyFleas
→ andFireFlies<wbr /> FriendlyFleas
→ ndFireFlies<wbr /> FriendlyFleasand
→ FireFlies<wbr />" as fast as they
→ could over and over.</p>

<p>His favorite site is this<wbr />
→ is<wbr />a<wbr />really<wbr />
→ really<wbr />longurl.com.</p>
```

输入 `wbr` 时，既可以用 `<wbr />`，也可以用 `<wbr>`。或许你已经猜到了，用到 `wbr` 的场合并不多。而且，截至本书写作之际，浏览器对它的支持并不一致。尽管在当前版

① 专名号用于表示人名、地名、朝代名等专名。——译者注

本的 Chrome 和 Firefox 中 wbr 是有效的，但 Internet Explorer 和 Opera 会忽略它。

3. ruby、rp 和 rt 元素

旁注标记(ruby annotation)是东亚语言(如中文和日文)中一种惯用符号，通常用于表示生僻字的发音。这些小的注解字符出现在它们标注的字符的上方或右方。它们常简称为旁注(ruby 或 rubi)。日语中的旁注字符称为振假名(furigana)。

ruby 元素以及它们的子元素 rt 和 rp 是 HTML5 中为内容添加旁注标记的机制。rt 指明对基准字符进行注解的旁注字符。可选的 rp 元素用于在不支持 ruby 的浏览器中的旁注文本周围显示括号。

下面的例子用英文占位符展示了这种结构，帮助你理解代码中以及支持和不支持它们的浏览器中对各类信息的安排方式(支持的浏览器效果如图 4.18.2，不支持的效果如图 4.18.3)。浏览器将旁注文本进行了突出显示：

```
<ruby>
  base <rp></rp><rt>ruby chars
  → </rt><rp></rp>
  base <rp></rp><rt>ruby chars
  → </rt><rp></rp>
</ruby>
```

现在，展示一个真实例子。例子中有两个表示“Beijing”的中文基准字符，以及伴随它们的旁注字符(图 4.18.4)：

```
<ruby>
  北 <rp></rp><rt>ㄅㄟ̌</rt><rp></rp>
  京 <rp></rp><rt>ㄐㄩㄥ</rt><rp></rp>
</ruby>
```

可以看到在不支持 ruby 的浏览器中括号的重要性(参见图 4.18.5)。没有它们，基准字符和旁注文本就会显示在一起，让内容变得混乱。



图 4.18.2 支持旁注标记的浏览器会将旁注文本显示在基准字符的上方(也可能在旁边)，不显示括号

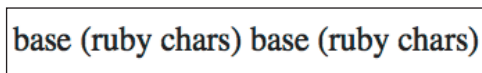


图 4.18.3 不支持旁注标记的浏览器会将旁注文本显示在括号里，就像普通的文本一样

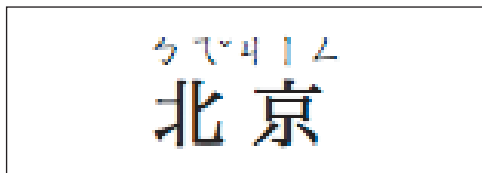


图 4.18.4 支持旁注的浏览器中显示的“Beijing”的旁注标记

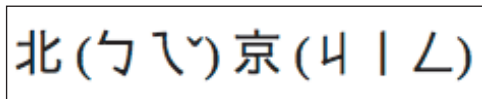


图 4.18.5 在不支持旁注的浏览器中，如果没有括号，内容就会变得难以理解

提示 在本书写作之际，Firefox 和 Opera 还缺少基本的 ruby 支持(这也是使用 rp 的原因)。不过，Firefox 插件 HTML Ruby (<https://addons.mozilla.org/en-US/firefox/addon/html-ruby/>) 为 Firefox 提供了对旁注的支持。

提示 关于旁注字符的更多信息，参见 http://en.wikipedia.org/wiki/Ruby_character。

4. bdi 和 bdo 元素

如果你的 HTML 页面中混合了从左至右书写的字符（如大多数语言所用的拉丁字符）和从右至左书写的字符（如阿拉伯语或希伯来语字符），就可能要用到 **bdi** 和 **bdo** 元素。

不过，首先要讲一点背景知识。除非在 **html** 元素中添加 **dir** 属性并将属性值设为 **rtl**，否则内容的基准方向都默认为从左至右。例如，`<html dir="rtl" lang="he">` 指明内容的方向为从右至左，且基准语言为希伯来语。

就像我在全书多个示例中对 **lang** 的处理方式一样，对于页面内的元素，如果其内容与页面基准设置不一致，也可以对其设置 **dir** 属性。因此，如果基准语言设为英语（`<html lang="en">`），又要包含一段希伯来语，就可以标记为 `<p dir="rtl" lang="he">...</p>`。

单独进行了设置的内容一般将按照希望的方向显示。这是由 Unicode 的双向（**bidirectional**）算法控制的。

如果上述算法不按设想的方式显示文本，可以使用 **bdo**（**bidirectional override**，双向重载）元素对内容进行重载。通常，这代表 HTML 源代码中的内容是视觉顺序而非逻辑顺序的情况。

视觉顺序（**visual order**）就是内容看上去的顺序——HTML 源代码内容与你希望显示的顺序相同。**逻辑顺序**（**logical order**）则与之相反，可用于希伯来语这样从右至左书写的语言。先输入从右至左的第一个字符，然后是第二个（也就是第一个字符左边的字符），以此类推。

根据最佳实践，Unicode 希望双向文本都以逻辑顺序呈现。因此，如果文本是以视觉顺序呈现的，算法仍会让字符反向，显示出与预期相反的顺序。如果你无法将 HTML 源代码中的文本转换为逻辑顺序（例如，这

些文本可能来自数据库或源），就只能依靠 **bdo** 了。

要使用 **bdo**，必须包含 **dir** 属性并将属性值设为 **ltr**（由左至右）或 **rtl**（由右至左），指定你希望呈现的方向。继续先前在英文页面中包含希伯来语段落的例子，应输入 `<p lang="he"><bdo dir="rtl">...</bdo></p>`。**bdo** 适用于段落里的短语或句子。不能用它包围多个段落。

bdi 元素是 HTML5 中新加的元素，用于内容的方向未知的情况。不必包含 **dir** 属性，因为默认已设为自动判断。下面的例子来自 HTML5 规范（但有少许修改）：

这个元素特别适用于包围用户生成的方向未知的内容。

在这个例子中，用户名与用户提交的帖子的数量显示在一起。如果不使用 **bdi** 元素，阿拉伯用户的用户名将让文本变得难以理解（双向算法会把冒号和数字“3”置于“User”一词旁边，而不是在“posts”一词旁边）。

```
<ul>
  <li>User <bdi>jcranmer</bdi>:
    → 12 posts.</li>
  <li>User <bdi>hober</bdi>:
    → 5 posts.</li>
  <li>User <bdi>ناني</bdi>:
    → 3 posts.</li>
</ul>
```

提示 要了解关于由右至左语言的更多信息，推荐阅读 W3C 的文章“Creating HTML Pages in Arabic, Hebrew, and Other Right-to-Left Scripts”（www.w3.org/International/tutorials/bidi-xhtml/）。

5. meter 元素

meter 元素也是 HTML5 的新元素。乍看起来，它很像 **progress** 元素（接下来会讲到，

根据规范，它表示“任务的完成进度”）。

可以用 `meter` 元素表示分数的值或已知范围的测量结果。简单地说，它代表的是投票结果（如“30% Smith, 37% Garcia, 33% Hawkins”）、已售票数（如“共 850 张，已售 811 张”）、考试分数（如“百分制的 90 分”）、磁盘使用量（如“256 GB 中的 74 GB”）等测量数据。

HTML5 建议（并非强制）浏览器在呈现 `meter` 时，在旁边显示一个类似温度计的图形——一个表示测量值的横条，测量值的颜色与最大值的颜色有所区别（相等除外）。作为当前少数几个支持 `meter` 的浏览器，Firefox 正是这样显示的（参见图 4.18.6）。对于不支持 `meter` 的浏览器，可以通过 CSS 对 `meter` 添加一些额外的样式，或用 JavaScript 进行改进。

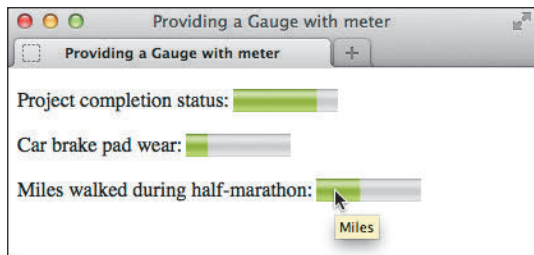


图 4.18.6 支持 `meter` 的浏览器（如 Firefox）会自动显示测量值，并根据属性值进行着色。`<meter>` 和 `</meter>` 之间的文字不会显示出来。如最后一个例子所示，如果包含 `title` 文本，就会在鼠标悬停在横条上时显示出来

虽然并非必需，但最好在 `meter` 里包含一些反映当前测量值的文本，供不支持 `meter` 的浏览器显示，如图 4.18.7 所示。

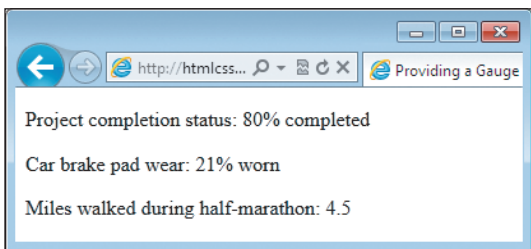


图 4.18.7 IE9 不支持 `meter`，它会将 `meter` 元素里的文本内容显示出来，而不是显示一个彩色的横条。可以通过 CSS 改变其外观

下面是一些 `meter` 的例子（显示结果如图 4.18.6 和图 4.18.7 所示）：

```
<p>Project completion status: <meter
→ value="0.80">80% completed</meter>
→ </p>

<p>Car brake pad wear: <meter low=
→ "0.25" high="0.75" optimum="0"
→ value="0.21">21% worn</meter></p>

<p>Miles walked during half-marathon:
→ <meter min="0" max="13.1" value="5.5"
→ title="Miles">4.5</meter></p>
```

`meter` 不提供定义好的单位，但可以使用 `title` 属性指定单位，如最后一个例子所示。通常，浏览器会以提示框的形式显示 `title` 文本（图 4.18.7）。

提示 `meter` 支持好几个属性。`value` 是唯一必需包含的属性。如果不指定 `min`（最小值）和 `max`（最大值），则默认将它们分别设为 0 和 1.0。`low`、`high` 和 `optimum` 属性通常共同作用，它们将范围划分为低、中、高三个区间。`optimum` 代表范围内的最优位置，如上文例子中的“0 brake pad wear”。如果 `low` 和 `high` 的值均不是最优的，可以将 `optimum` 设为它们之间的值。

提示 截至本书写作之际，浏览器对 `meter` 的支持情况还在变化：Internet Explorer、移动 Safari（用于 iOS 设备）和 Android 浏览器还不支持该元素。这大概也是现实中很少见到它的原因。随意使用它，只是需要了解大多数浏览器默认只显示 `meter` 文本而不显示图形（图 4.18.7）。关于最新的浏览器支持情况，参见 <http://caniuse.com/#feat=progressbar>。

提示 支持 `meter` 的不同浏览器显示测量值图形的样式可能有差异。

提示 已经有人试过针对支持 `meter` 的浏览器和不支持的浏览器统一编写 `meter` 的 CSS。在网上搜索“style HTML5 meter with CSS”（用 CSS 为 HTML5 的 `meter` 添加样式），就可以找到一些解决方案（注意其中的一些用到了 JavaScript）。

提示 `meter` 并不用于标记没有范围的普通测量值，如高度、宽度、距离、周长等。例如，这种写法是不正确的：`<p>I walked <meter value="4.5">4.5</meter> miles yesterday.</p>`。

提示 一定不要将 `meter` 和 `progress` 元素混在一起使用。

6. progress 元素

`progress` 元素也是 HTML5 的新元素。前面说过，它指示某项任务的完成进度。可以用它表示一个进度条，就像在 Web 应用中看到的指示保存或加载大量数据操作进度的那种组件。

就像 `meter` 一样，支持 `progress` 的浏览器会根据属性值自动显示一个进度条（参

见图 4.18.8）。同时，和 `meter` 一样，为了让旧的浏览器也能表现进度，尽管并非必需，但最好在 `progress` 中包含反映当前进度的文本（如示例中的“0% saved”，参见图 4.18.9）。

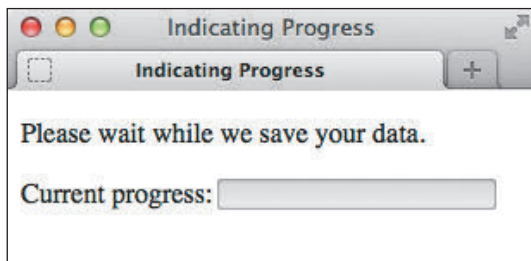


图 4.18.8 支持 `progress` 的浏览器（如 Firefox）会自动显示进度条，并根据值对其进行着色。`<progress>` 和 `</progress>` 之间的文本不会显示出来。在这个例子中 `value` 属性设成了 0，因此整个横条都是同样的颜色

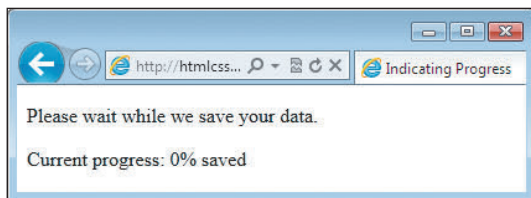


图 4.18.9 IE9 不支持 `progress`，因此不会显示有颜色的横条，而是显示该元素里面的文本。可以通过 CSS 改变其外观

下面是一个例子：

```
<p>Please wait while we save your  
→ data.</p>  
  
<p>Current progress: <progress  
→ max="100" value="0">0% saved  
→ </progress></p>
```

对 `progress` 的完整说明已经超出了本书的范围。通常，你只能通过 JavaScript 动态地更新 `value` 属性值和元素里面的文本以指示任务进程（例如，指示已完成 37%）。使用 JavaScript 与在 HTML 中硬编码从视觉上看是

一样的，即

```
<progress max="100" value="37">37%
saved</progress>
```

效果如图 4.18.10 所示。当然，不支持的浏览器会显示为类似于图 4.18.9 所示的样子。

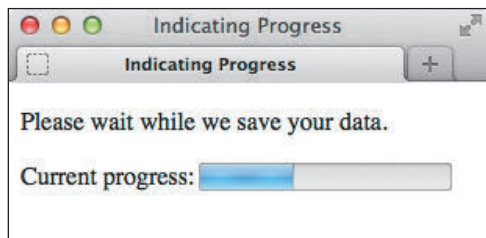


图 4.18.10 Firefox 中显示的进度条，通过 JavaScript（或直接在 HTML 中）将 value 属性设为 37（假定 max="100"）

提示 progress 元素支持三个属性：max、value 和 form。它们都是可选的，max 属性指定任务的总工作量，其值必须大于 0。value 是任务已完成的量。如果 progress 没有嵌套在 form 元素里面，又需要将它们联系起来，可以添加 form 属性并将其值设为该 form 的 id。

提示 下面简单地看看如何用 JavaScript 修改 progress 元素。假定进度条已经设定了一个 id，如：

```
<progress max="100" value="0" id=
→ "progressBar">0% saved</progress>
```

下面的 JavaScript 可以让你获取该元素：

```
var bar = document.getElementById
→ ('progressBar');
```

然后就可以通过 bar.value 获取或设定 value 的值。例如，**bar.value=37;** 可以将值设为 37。

提示 在本书写作之际，大多数桌面浏览器的最新版本都支持 progress 元素。IE9 及之前的版本、移动 Safari 和 Android 浏览器还不支持该元素。关于最新的浏览器支持情况，参见 <http://caniuse.com/#feat=progressmeter>。

提示 不同浏览器显示进度条的样式可能不同，不过可以通过 CSS 为它添加样式。

本章内容

- 关于 Web 图像
- 获取图像
- 选择图像编辑器
- 保存图像
- 在页面中插入图像
- 提供替代文本
- 指定图像尺寸
- 在浏览器中改变图像的尺寸
- 在图像编辑器中改变图像的尺寸
- 为网站添加图标

为 Web 创建图像与为在纸上输出而创建图像有所不同。尽管 Web 图像和可打印图像的基本性质是相同的，但它们在格式、下载速度、颜色、大小（尺寸）、透明度和动画等六个主要方面有一些区别。

本章将阐释这六个要素的重要方面，以及如何使用这些知识为网站创建有效的图像。另外，还将学习如何在网页中插入图像。

5.1 关于 Web 图像

让我们来看看创建 Web 图像时应记住的六个要素。如果你不愿意花时间了解具体的细节，可以直接跳至节末的“小结”。

1. 格式与下载速度

在纸上打印图像的人不必担心读者将使

用什么查看图像，读者在打开杂志或报纸的页面时也不必等待图像的出现。不过，在万维网上，情况就不一样了。

每天，人们通过数以万计的 Mac、基于 Windows 的 PC、Linux 机器、手机、平板电脑等各种各样的设备访问万维网。网页中的图像应该采用这些设备的操作系统都能识别的格式。当前，Web 上用的最广泛的三种格式是 GIF、PNG 和 JPEG。我们的目标是选择质量最高，同时文件最小的格式。

◎ JPEG

JPEG 格式适用于彩色照片，因为它包含大量的颜色并进行了合理的压缩，采用这种格式保存的文件相对较小（参见图 5.1.1）。图像的文件变小（且不论是何种格式），下载速度就会变快，访问者就不必等待较长时间才能看到图像。



图 5.1.1 全彩的照片通常保存为 JPEG 格式。PNG-24 格式效果也很好，但通常其文件大小比 JPEG 要大得多

不过，JPEG 是一种有损的格式，因此将在图像保存为 JPEG 时会丢失一部分原始信息，但通常有必要这样做，因为我们可以将图像质量的损失控制在用户不易觉察的范围内，却能显著改善图像的加载速度。

注意，对 JPEG 进行解压缩无法还原图像先前丢失的细节。因此，如果未来还有可能对图像进行编辑，就应该保存一份使用无损压缩格式（如 PSD 或 TIFF）的图像副本，只有在确保不再对图像进行修改的情况下才能只保存 JPEG 格式的图像。

◎ PNG 和 GIF

PNG 和 GIF 是无损的格式，因此采用这两种格式对图像进行压缩时不会造成品质的损失。GIF 只有 256 种颜色，但 PNG 却支持几百万种颜色。与 JPEG 不同，PNG 和 GIF

均支持透明，它们更适用于保存非照片类的图像。通常，拥有大片纯色的图像，如标识、重复的图案、插图以及图像文字等都适合使用这两种格式。

可以使用 PNG 保存照片，但由于无损图像质量，文件尺寸会比 JPEG 大得多。因此，只有在压缩造成的质量损失不可忽略的情况下才使用 PNG 保存照片。

PNG 有几种不同的分支：PNG-8、PNG-24 和 PNG-32。一般来说，对于 PNG 和 GIF，应优先选择 PNG，因为它对透明度的支持更好，压缩算法也更好，产生的文件更小。

通过表 5.1.1 可以对比不同的图像格式（包括 PNG 的不同分支）。5.4 节包含了对图像格式和压缩情况进行设置的例子。

表 5.1.1 图像格式对比

格式	用 法	颜 色	索引色（基本）透明	alpha 透明
JPEG	适用于大多数照片，以及其他颜色较多且可接受一些质量损失的图像	1600 万以上	—	—
PNG-8	适用于标识、重复的图案以及其他颜色较少的图像或具有连续颜色的图像	256	支持	支持
PNG-24	与 PNG-8 相似，不过支持颜色更多的图像。适用于颜色丰富且质量要求高的照片	1600 万以上	支持	—
PNG-32	与 PNG-24 相似，不过支持具有 alpha 透明的图像	1600 万以上	—	支持
GIF	用法与 PNG-8 相似，在大多数情况下应使用 PNG-8	256	支持	—

WebP 图像格式

谷歌建立了另一种图像格式，名为 WebP。这种格式既支持有损压缩也支持无损压缩，它产生的文件大小也远小于 JPEG 和 PNG。跟 PNG 一样，它还支持 alpha 透明。

WebP 还在发展之中，截至本书写作之际，完全支持这种格式的浏览器还仅限于 Chrome、Opera 12+ 和一些 Android 浏览器，至于其他浏览器是否会支持这种格式尚不可知。不过，有必要关注一下这种格式的进展情况。在很长一段时间内，还没有办法做到让支持它的浏览器显示这种格式的图像，而其他浏览器使用备用格式的图像。要了解 WebP 的详细信息，参见 <https://developers.google.com/speed/webp/>。最新的浏览器支持情况参见 <http://caniuse.com/#search=webp>。

2. 颜色

大多数计算机显示器可以显示数以百万计的颜色，但也有例外的情况。有的图像格式的调色板是有限的。GIF 和 PNG-8 图像只有 256 种颜色，对标志和图标来说通常这已经足够了（参见图 5.1.2）。



图 5.1.2 标识和其他颜色较少的图像通常保存为 PNG-8 格式（也可以保存为 GIF 格式，不过相较之下 PNG 更好一些）。另见彩插

JPEG、PNG-24 和 PNG-32 均支持超过 1600 万种的颜色，因此照片和复杂的插图应选择这些格式。不过，需要指出的是，对于这些图像，大多数情况下应使用 JPEG。

3. 大小（尺寸）

你曾经收到过一些在屏幕上看起来特别大的照片吗？这样的照片通常是由数码相机拍摄的，而发送照片给你的人又没有事先编辑文件以缩减尺寸。那么为什么这些照片起初这么大呢？如果将它们放到网站上，应该将文件大小控制在多少呢？

数字图像以像素为单位进行度量。如今，超过 800 万像素的数码相机已经很常见了，不过我们不妨以 300 万像素的数码相机举例说明。300 万像素的数码相机可以照出 2048 像素宽、1536 像素高的照片。这在浏览器里有多大呢？视情况而定。不过一般而言，这样的尺寸太大了（参见图 5.1.3），你在邮件中查看这么大的图像就是这个样子。

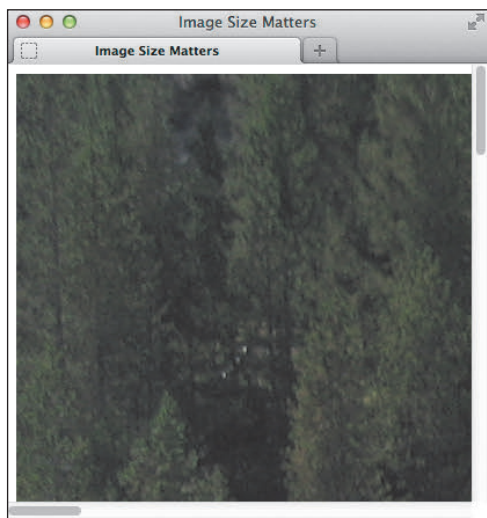


图 5.1.3 这个图像为 2048 像素宽，1536 像素高，这意味着要看到照片的其他部分，既需要纵向滚动条，又需要横向滚动条，更别提文件尺寸了，需要很长时间才能将图像显示完整。这可谓只见树木、不见森林啊（真是令人崩溃）

那么，对于网站上的图像，多大合适呢？简单地说，通常应控制在几百个像素宽（参见图 5.1.4）。图像尺寸越大，文件就越大，网页加载它的时间就越长。选择图像尺寸是一个取舍的过程。此外，你已看到，如果图像尺寸过大，访问者就需要使用横向滚动条来查看（参见图 5.1.3）。

当然，图像的尺寸也取决于其用途。图标通常很小，标识稍大一些，照片则大得多。有时，为了产生视觉冲击，我们可能需要使用一张达到或超过整个网站内容宽度的大图。这样的图像通常不超过 960 像素宽。

打印出来的图像通常比显示器上看到的小一些。打印机的每英寸点数（dpi）通常比显示器的每英寸像素数（ppi）要多。屏幕分辨率变大加剧了这一情形。这就是为什么相同的图像在显示器或笔记本上看要比在纸上看要大得多。

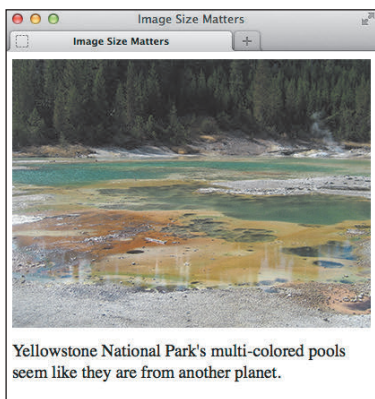


图 5.1.4 看，原来这张图还有这么多内容！我减小尺寸（参见 5.9 节）后很容易就能看到整个图像。这个版本有 400 像素宽，300 像素高，维持了原图 4 : 3 的宽高比，保存了图像的基本形状。我们在第 12 章会了解到，可以使用 CSS 控制图像的宽度和高度，让图像放大或缩小，使之在所有的屏幕（从手机到大的显示器）上都能以最好的效果显示

可缩放矢量图形（SVG）

对于使用 SVG 图像语言创建的图像，无论放大还是缩小都不会影响其质量（以及其他的一些参数）。而且，对于某个 SVG 图像来说，无论它在页面中显示的尺寸是多大，其文件大小总是恒定的。目前，几乎所有的现代浏览器都提供基本的 SVG 支持，因此你可以在网页中使用 SVG。不过，IE8 还不支持 SVG。可以使用 SVG Web（<http://code.google.com/p/svgweb/>）或 Raphaël（<http://dmitrybaranovskiy.github.io/raphael/>）等 JavaScript 库在 IE8 中实现类似的效果。更多关于 SVG 的信息（尤其是结合使用 HTML 视频和 SVG），参见 17.14 节。

4. 透明度

可以利用透明度为图像创建非矩形的边缘，在图像的下面设置背景色或图案。PNG 和 GIF 都支持透明度，JPEG 则不支持。

在 GIF 格式中，一个像素要么是完全透明的，要么是不透明的。这称作索引色透明（index transparency）。而 PNG 则既支持索引色透明，又支持 alpha 透明（alpha transparency）。alpha 透明可以控制一个像素透明的程度。也就是说，一个像素可以部分透明，而非要么透明要么不透明。这意味着具有复杂透明背景的图像使用 PNG 的效果（参见图 5.1.5）要好于使用 GIF 的效果，因为使用 PNG 可以让边缘变得平滑，避免产生锯齿。

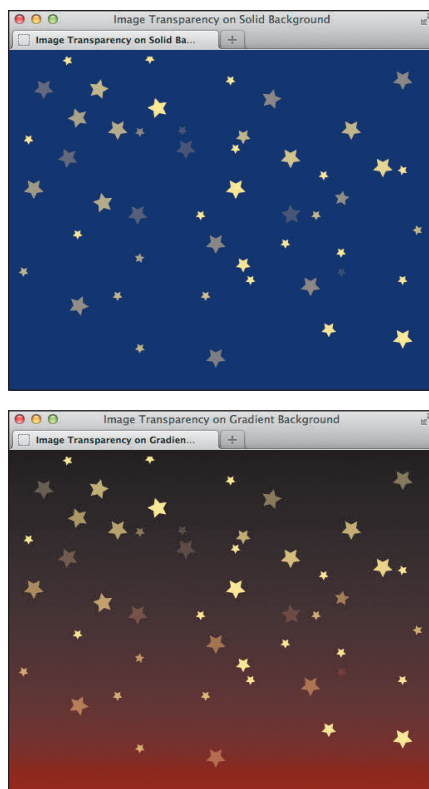


图 5.1.5 纯蓝色背景和从黑到红渐变色背景都不是星星 PNG 图的一部分，它们只是用 CSS 向页面的 body 添加的颜色。星星图具有 alpha 透明度，这样，无论背景是纯色、渐变色，还是另一张图像，就都能穿透，且看起来相当“干净”。JPEG 无法实现这一点，GIF 仅支持一种简单的透明度，效果并不太好（除非背景是纯色的）。另见彩插

PNG-8 既支持索引色透明, 也支持 alpha 透明, 但需要使用 Fireworks 这样的程序才能将图像保存为 PNG-8 格式。Photoshop 不支持 alpha 透明的 PNG-8, 但支持 alpha 透明的 PNG-32。(参见补充材料“Photoshop、PNG-24 与 PNG-32”。)这也是万维网上大多数透明 PNG 都是 PNG-32 的原因。

总之, 对于透明图像, 应使用 PNG-8 或 PNG-32。后者允许使用超过 256 种颜色。

Photoshop、PNG-24 与 PNG-32

PNG-24 与 PNG-32 几乎是一样的, 只是后者支持 alpha 透明。奇怪的是, Photoshop 将 PNG-24 和 PNG-32 都称为 PNG-24, 因此通常容易认为 PNG-24 是支持 alpha 透明的。实际上, 在 Photoshop 里, 如果为 PNG-24 选中透明度选项, 它会自动创建 PNG-32 图像。

其实, 大多数人都不清楚这一点。很多人甚至没有听说过 PNG-32。因此, “PNG-24”也常常用于指代带有 alpha 透明的图像, 不过, 从技术上看这种说法并不准确。

5. 动画

动画可以保存为 GIF, 但不能是 JPEG 或 PNG。实际上, 使用图像表现动画已经用得越来越少了。(一种例外的情况是那些好玩的 GIF 动画, 它们在 Tumblr 这类网站相当流行。)我们通常使用 CSS 动画、JavaScript、HTML5 Canvas、SVG(可谓是一匹黑马)和 Flash 创建动画。近几年, 使用 Flash 创建的动画越来越少了。这主要是由于 iOS 不支持 Flash, 且其他标准的 Web 技术的能力和浏览器支持程度都提升了很多。

6. 小结

让我们回顾一下有关 Web 图像的要点。

- ❑ 大多数照片都应保存为 JPEG 格式, 图标、标识等颜色较少的图像应保存为 PNG 格式。
- ❑ 应使用 PNG-8 或 PNG-32(常误作 PNG-24)创建具有 alpha 透明度的图像。
- ❑ 合理的图像大小(尺寸)会让图像文件变得更小。应尽可能地减小图像尺寸, 从而让页面加载得更快。

5.2 获取图像

如何获取在网页中使用的图像呢? 有几种方式。可以购买或下载现成的图像, 通过扫描仪将照片或手绘图像数字化, 使用数码相机拍摄照片, 或使用 Adobe Photoshop 这样的图像编辑软件从头绘制图像。在有了图像以后, 就可以将它们保存为上文提到的某种格式并在万维网上使用它们了。

获取图像的手段

- ❑ 可以使用搜索引擎寻找万维网上的图像, 方法是点击搜索框上方的“图像”(Images)链接, 再跟平常一样输入搜索词。注意, 一般来说, 对于在万维网上找到的图像, 即便是免费的, 也会受到某种形式的版权限制(参见后面的“知识共享许可协议”)。购买的图像通常可以随意使用, 但一般不能对图像本身再次进行销售。请仔细阅读声明或许可协议。
- ❑ 很多公司以较低的价格销售库存的照片和图像。通常, 每张图像都有几个版本, 以满足不同的用途和分辨率要求。
- ❑ 数码相机(包括智能手机的相机)可能是创建个人图像最为常用的方法。

知识共享许可协议

知识共享 (Creative Commons, www.creativecommons.org) 是一个非营利组织, 它开发了一个版权模板体系, 让艺术家可以按照其指定的方式分享他们的作品, 同时无需放弃对作品的所有权利。网站设计师、音乐家和摄影师使用知识共享许可协议将他们的作品放入市场, 而无需担心别人以他们不允许的方式利用这些作品。

Flickr 是流行的图片分享 Web 应用 (www.flickr.com)。它要求用户为他们上传的每张图片指定一种知识共享许可协议, 然后访问者可以根据许可协议的类型搜索图片。它是为网站寻找图片的好地方。

此外, 使用 Google 搜索引擎可以根据使用权限对搜索进行限定 (进入 www.google.com/advanced_search, 再在底部“使用权限” (Usage Rights) 下拉菜单中选择需要的选项)。

5.3 选择图像编辑器

有很多不同的软件可以用来创建和保存 Web 图像。大多数现代图像编辑器都提供了创建 Web 图像的专用工具。这些工具考虑了本章前面讨论的几种要素。

毫无疑问, 行业标准是 Adobe Photoshop (www.adobe.com)。跟它功能相近的 Adobe Fireworks 也非常强大。它们都既有 OS X 版也有 Windows 版。在 2013 年春, Adobe 宣称尽管它们还将继续销售 Fireworks, 但不会再添加新的功能。这是在购买该软件时需要考虑的一点。还有很多软件可以作为 Photoshop 和 Fireworks 的替代品 (通常也要便宜得多), 参见补充材料。

对于为网站准备的图像, 这些软件都可

以对其进行最基本的处理, 如缩放、剪裁、调整颜色、应用效果、优化等。对于收费软件, 通常都有免费的试用版。OS X 自带一个非常基础的图像编辑器, 名为 Preview。还可以在网搜索 “image editors” (图像编辑器), 找到更多的选择。

不过, 需要强调一点, 无论使用哪种软件, 优化 Web 图像的基本策略始终不会改变。它们所用的命令名称可能有所不同, 步骤也有多有少, 但思路是一样的。

Photoshop 和 Fireworks 的替代品

创建图像并不仅限于 Photoshop 和 Fireworks。以下列举了一些可以替代它们的软件。

❑ Gimp

(Linux 或 OS X: www.gimp.org Windows: <http://gimp-win.sourceforge.net/stable.html>), 免费

❑ Acorn

(OS X: <http://flyingmeat.com/acorn/>)

❑ Pixelmator

(OS X: www.pixelmator.com)

❑ Paint.NET

(Windows: www.getpaint.net), 免费

❑ PaintShop Pro

(Windows: www.corel.com)

每款程序都有其自身的使用方式, 因此你的选择取决于你的使用偏好。

5.4 保存图像

创建图像后需要保存图像, 这一过程是在图像的视觉质量与文件大小之间寻找平衡的艺术。

如果你的计算机上没有安装 Photoshop，可以先使用试用版。

1. Adobe Photoshop

Photoshop 在 File 菜单中提供了 Save for Web (存储为 Web) 命令。它让用户可以从视觉上对比原始图像和一至三个优化后的版本，还可以看到对文件大小和下载时间的影响。

2. 使用 Photoshop 的 Save for Web 命令的步骤

(1) 打开 Photoshop，创建图像。或者打开现有的图像，并通过剪裁、调整大小和编辑，为发布做好准备。

(2) 选择 File → Save for Web (在之前的版本中称为 Save for Web & Devices)。出现 Save For Web 对话框，参见图 5.4.1。

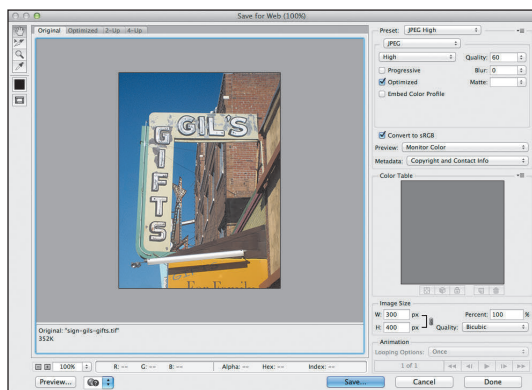


图 5.4.1 Save for Web 对话框默认对应 Original 选项卡，显示原始图像

(3) 点击 2-Up (2 联) 选项卡查看一个优化后的版本，或点击 4-Up (4 联) 选项卡查看三个优化后的版本，参见图 5.4.2。

(4) 根据需要，点击一个优化后的版本，参见图 5.4.2。



图 5.4.2 选择 4-Up 选项卡 (上)，可以比较原始图像与你所选择的三个优化方案。这里选择了左下方的图，从而可以修改其设置 (图 5.4.3)

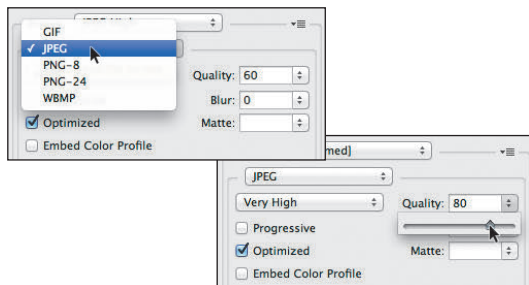


图 5.4.3 Save for Web 对话框的右侧是改变优化设置的控件。可以从下拉菜单 (上) 中选择图像格式，然后通过直接输入一个数值或者拖动滑动条 (下) 指定图像的质量。数值越大，质量就越高，而文件也越大。此外，还可以在 Quality 左侧的下拉菜单中选择预设值，如“Very High” (非常高)。尽管我们通常认为 JPEG 是保存照片的最佳格式，但如果质量设的太低，大块的颜色 (如本图中的蓝天) 就会变得有些模糊不清。

(5) 选择希望采用的格式，参见图 5.4.3。

通常，在计算机上创建的图像（包括标志、图表、图形、线条艺术，以及任何包含由单色和锐利细节构成的大块区域的图像）应保存为 PNG-8 格式，参见图 5.4.1。

具有连续色调的图像（如照片）应保存为 JPEG 或 PNG-24 格式。

(6) 调整其他设置选项，直到在图像质量可接受的条件下获得最小的文件，参见图 5.4.4。

(7) 点击 Save。选择一个目录，并对新文件进行命名。它将自动包含所选格式的扩展名（而且，这样通常就不会覆盖原始图像）。

输出 PNG 图像的步骤相同（参见图 5.4.5 和图 5.4.6）。



图 5.4.4 这里的修改反映的是左下方的区块。如果保存它，其大小就是 62.39K，尽管在图 5.4.2 中它只有 19.44K，但现在它的边缘更锐利了，这种质量上的提升是值得的。这里还修改了其他两个区块的设置。PNG-8（右上）的压缩让图像的像素化程度加深了，而且文件大小（70.56K）也比 JPEG 要大。PNG-24（右下）保持了很高的质量，但文件大小（224.3K）也大得多。显然，JPEG 是保存照片的最佳选择。另见彩插

提示 应该使用 RGB^①模式创建图像，而不是 CMYK^②模式（用于印刷）。

提示 在优化和输出图像时，没有所谓的唯一正确或者错误的设置，记住你的主要目标是在保留可以接受的图像品质的前提下让图像尺寸尽可能小。

① R、G、B 分别表示 Red（红）、Green（绿）、Blue（蓝）。RGB 模式又称三原色光模式。——译者注

② C、M、Y、K 分别表示 Cyan（青）、Magenta（品红）、Yellow（黄）、Black（黑）。CMYK 模式又称印刷四分色模式。——译者注

提示 压缩图像文件大小的工具有很多（甚至可以对已经在图像编辑器中优化过的图像作进一步压缩）。其中有的运行在你的机器上，如 ImageOptim（<http://imageoptim.com>，只有 OS X 版）和 JPEGmini（www.jpegmini.com），有的则是基于 Web 的，如 www.smushit.com。可以在网上搜索“image optimization tools”（图像优化工具），找到更多这样的工具。推荐使用其中的某个工具。

提示 如果不确定应该选择哪种格式，可以比较两种优化结果，看看哪种格式的压缩效果更好。另外，不同的编辑器生成的优化图像的尺寸可能不同。



图 5.4.5 这个图像（已被放大）有很多平铺的颜色及文字，它们需要保持锐利。可以看到，PNG-8 格式（左下）对图像的压缩是最好的，文件大小只有 5.5K。如果减少颜色的数量，其大小还将进一步减小。PNG-24 可以提供更多的颜色，使用这种格式的话，文件大小为 11.17K。使用最高质量的 JPEG 则为 19.76K。如果将 JPEG 的质量降为 50（这里没有显示），文件大小可比 PNG-8 更小一点，但其效果看上去却很可怕。另见彩插

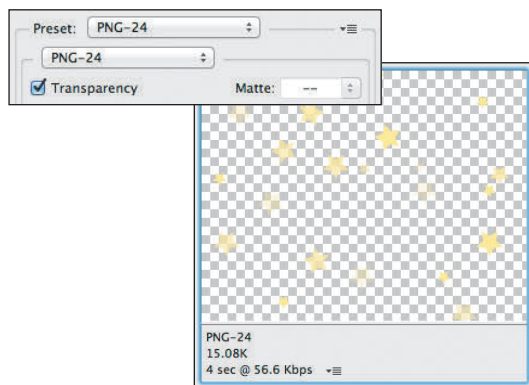


图 5.4.6 对于具有 alpha 透明的图像，如这里的星星图（下），选择 PNG-24 并选中 Transparency（上）。其他的图像编辑器可能将这个选项称为 PNG-32（注意，要在打开 Save for Web 对话框之前设置好图像的透明区域）

提示 Save for Web 命令会创建新的图像，并保持原始图像不变（除非你在旧图像文件的同一个文件夹使用相同的文件名和扩展名保存新的图像）。

提示 在 PSD（Photoshop 文档）中，可以让图像的不同部分位于不同的图层，从而可以对它们进行显示和隐藏。通常，在优化后的版本中，只保存了图像的可视图层。Fireworks 可以让 PNG 拥有额外的信息，从而可以保存图层。注意，Photoshop 无法显示这些图层。

5.5 在页面中插入图像

可以在网页中放置各种各样的图像，从标志到照片都可以。当访问者浏览网页时，浏览器会自动加载像图 5.5.1 这样在 HTML 文档中描述的图像，结果参见图 5.5.2。不过，图像加载时间跟访问者的网络连接强度、图像尺寸，以及页面中包含的图像个数相关。

```

...
<body>
<h1>Barcelona's Market</h1>



<p>This first picture shows one of the fruit
→ stands in the <span lang="es">Mercat de la
→ Boqueria</span>, the central market that
→ is just off the Rambles. It's an incredible
→ place, full of every kind of food you
→ might happen to need. It took me a long
→ time to get up the nerve to actually take
→ a picture there. You might say I'm kind
→ of a chicken, but since I lived there,
→ it was just sort of strange. Do you take
→ pictures of your supermarket?</p>
</body>
</html>

```

图 5.5.1 这个图像的 URL 只包含文件名，没有路径，因此代表该图像位于与此网页相同的文件夹。关于如何引用不同文件夹中的图像参见第一条提示

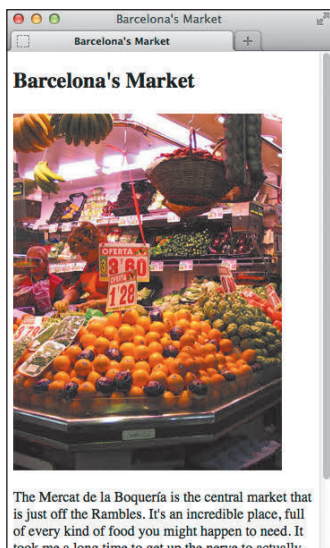


图 5.5.2 图像贴向页面的左侧，与文本的对齐方式一致。使用 float（参见 11.11 节）等 CSS 属性可以改变对齐方式或让文字环绕图像

在页面中插入图像的步骤

(1) 在 HTML 代码中，将光标放在希望图

像出现的位置。

(2) 输入 ``，其中 `image.url` 指示图像文件在服务器上的位置。

(3) 输入一个空格和 `/>`（或者 `>`，在 HTML5 中两者均有效）。

提示 图 5.5.1 中的例子显示了最简单的图像路径形式，即只有文件名。不过，在实践中，为了保持良好的文件组织结构，通常将图像保存在单独的文件夹中。img 标签的 src 属性中的 URL 也应该反映这一路径。假设图 5.5.1 中的页面所在的文件夹还包含一个名为 images 的文件夹，且图像位于 images 文件夹中，则显示该图的 HTML 应为 ``。更多关于如何引用文件的信息，参见 1.7 节。

提示 图像必须先上传到服务器上，访问者才有可能看到它们。上传图像跟上传 HTML、CSS、JavaScript 以及其他文件一样，参见 21.3 节。

提示 不要期望你的访问者会长时间等待页面加载和显示。可以对页面进行测试（别忘了你的连接速度可能比访问者的快）。如果你都等不下去，那么访问者也一样等不下去。另一种办法是为大图创建缩略图，让访问者可以通过链接选择查看大图，查看 6.3 节学习如何创建这种链接。Charles（www.charlesproxy.com）和 Fiddler（<http://fiddler2.com>）是模拟慢连接的两个不错的工具。

提示 可以使用 CSS 的 border 简写属性（及相关属性）为图像应用边框样式，这将在第 11 章讲到。旧的浏览器会对带链接的图像自动加上一个边框，可以在 CSS 中指定 `img { border: none; }` 移除该边框。

5.6 提供替代文本

使用 alt 属性，可以为图像添加一段描述性文本，当图像出于某种原因不显示的时候，就将这段文字显示出来。屏幕阅读器可以朗读这些文本，帮助视障访问者理解图像的内容。HTML5 规范推荐将 alt 文本理解为图像的替代性描述：“一般来说，替代文本是考虑图像未能正常加载的情况下需要呈现的文字。”通常，这意味着 alt 文本可以插入到图像两侧的文本流中，在大多数情况下，它不应是对图像的描述。

提供图像无法显示时的替代文本的步骤

- (1) 在 img 标签内，在 src 属性及其值的后面，输入 alt="”。
- (2) 输入图像出于某种原因没有显示时应该出现的文本（参见图 5.6.1 和图 5.6.2）。
- (3) 输入 "。

```
...
<body>
<h1>Barcelona's Market</h1>



<p>The <span lang="es">Mercat de la Boqueria
→ </span> is the central market that is just
→ off the Rambles. It's an incredible place,
→ full of every kind of food you might
→ happen to need...</p>
</body>
</html>
```

图 5.6.1 我引用了一个不在我的网站的图像（market.jpg），以显示 alt 文本的效果



图 5.6.2 在 Internet Explorer 10 中，替代文本出现在一个带叉的小方块旁边，且两者由一个方框包围。在 Firefox 和 Opera 等其他浏览器中，替代文本是单独出现的。Chrome 和 Safari 不会显示 alt 文本，而是显示缺失图像的图标

提示 HTML5 规范包含了对在不同场景中如何有效使用 alt 的大量讨论（www.w3.org/TR/html5/embedded-content-0.html#alt），建议读者去看一看。

提示 如果图像对内容的价值较小，对视障用户来说不太重要，则可以提供空的替代文本，即 alt=""。如果图像与邻近的文本表达的信息相似，也可以将 alt 属性留空。（注意，图 5.6.1 中的例子便符合这一标准，尤其是段落中提到了水果类型的情况。）

提示 不要用 alt 文本代替图像的 caption。在这种情况下，应考虑将 img 放入一个 figure 元素，并添加一个 figcaption 元素。4.4 节有一个这样的例子。

提示 如果图像是页面设计的一部分，而不是内容的一部分，则应使用 CSS background-image 属性引入该图像，而不是使用 img 标记。参见 10.10 节。

为什么图像无法显示

图像无法显示的原因有很多种。也许是因为你在 src 属性中填写的 URL（参见图 5.6.1）是错误的，也许是因为你忘了将图像上传到服务器。其他原因则超出了你的控制范围，如访问者的网络连接状况可能很糟糕。此外，你知道可以让浏览器不加载图像吗？大多数浏览器都可以在偏好设置中设定是否加载图像。通常，为了加快页面加载速度，用户可能选择这一选项。使用手机或平板电脑的用户，为了防止侵吞太多流量，也有可能选择这一选项。

5.7 指定图像尺寸

有时，加载网页会先看到文本，等一小段时间以后图像开始加载时，文本跳到图像周围，留出可容纳图像的空间。出现这种现象，是因为没有在 HTML 中指定图像的尺寸。如果用户使用旧浏览器或者网络连接速度慢，就容易出现这种情况。

如果指定图像的尺寸，浏览器就可以预留空间，在图像加载的同时让文本显示在周围，保持布局的稳定。

可以通过浏览器或图像编辑软件获取图像的精确尺寸。

1. 在浏览器中查看图像尺寸

(1) 右击图像，出现背景弹出菜单，参见图 5.7.1。

(2) 选择 View Image Info（查看图像信息，具体选项取决于所使用的浏览器），出现的框中会以像素为单位显示图像的尺寸，参见图 5.7.2。

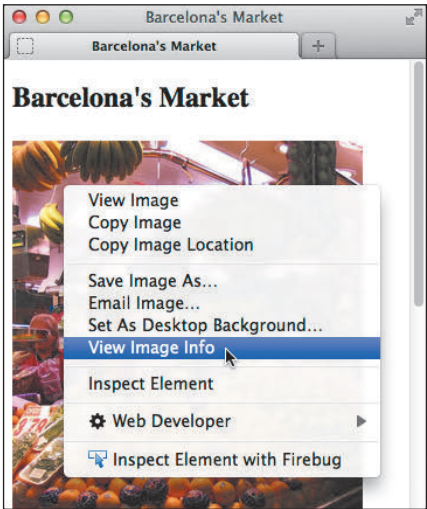


图 5.7.1 在浏览器中右击图像，出现背景弹出菜单。浏览器会提供检查图像、显示属性或获取尺寸的方式

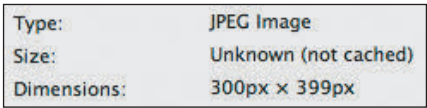


图 5.7.2 这个框（它的外观取决于所用的浏览器）以像素为单位显示了图像的尺寸

2. 在 Photoshop 中查看图像尺寸

(1) 在 Photoshop 中打开图像。

(2) 选择 Image → Image Size（如图 5.7.3 所示）。出现 Image Size 对话框（参见图 5.7.4）。

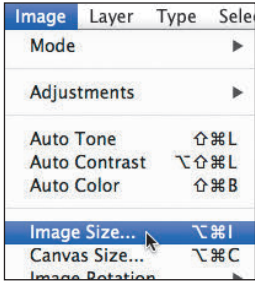


图 5.7.3 在 Photoshop 中，从 Image 下拉菜单中选择 Image Size

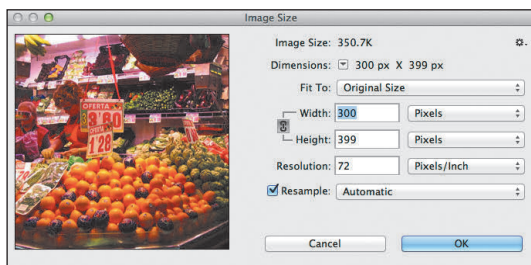


图 5.7.4 Image Size 对话框显示图像为 300 像素 × 399 像素（可以改变这个对话框的大小，确保在 Width 或 Height 旁边的下拉菜单中选择的单位为 Pixels，再输入新的 Width 值，Height 值会根据图像的比例自动计算出来。更多信息参见最后一条提示）

3. 在 HTML 中指定图像尺寸

(1) 使用“在浏览器中查看图像尺寸”或“在 Photoshop 中查看图像尺寸”中介绍的方法，确定图像的尺寸。

(2) 在 `img` 标签中，`src` 属性的后面，输入 `width="x" height="y"`，采用第 (1) 步中确定的值，以像素为单位指定 `x` 和 `y`（分别代表图像的宽度和高度）的值，参见图 5.7.5。

```
...
<body>
<h1>Barcelona's Market</h1>



<p>The <span lang="es">Mercat de la Boqueria
→ </span> is the central market that is just
→ off the Rambles. It's an incredible place
→ ...</p>

</body>
</html>
```

图 5.7.5 通常我们会在 HTML 中明确指定图像的高度和宽度，这样浏览器就不必花时间来判断图像的尺寸，从而更快地将图像显示出来。但有一种情况我们应该忽略 `width` 和 `height` 属性，即显示响应式图像的情况，我们会在 12.2 节讨论这个问题

提示 `width` 和 `height` 属性不一定要反映图像的实际尺寸。

提示 如果有几个尺寸相同的图像，可以通过样式表同时设置它们的高度和宽度。当然，单个图像也能使用这种方法。

提示 在浏览器中，如果在单独的窗口中打开图像，就可以看到图像的尺寸，参见图 5.7.6。

提示 在 Photoshop 中可以选中整个图像，再在 Info（信息）面板中查看图像的尺寸。

提示 如果使用如图 5.7.4 所示的 Image Size 对话框修改图像的尺寸，要确保 Resample（重新取样）选项为选中状态。Resolution（分辨率）选项与 Web 图像无关。5.9 节介绍了另外一种在 Photoshop 中改变图像尺寸的方法。

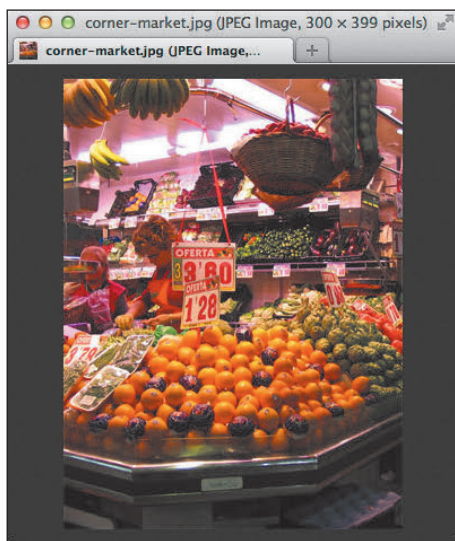


图 5.7.6 如果在单独的窗口中打开图像，或者将图像拖入单独的窗口，标题栏里就会显示图像的尺寸

5.8 在浏览器中改变图像的尺寸

通过为图像指定新的高度和宽度（以像素为单位），可以改变图像显示的尺寸，参见图 5.8.1、图 5.8.2 和图 5.8.3。这样做可以在保持所有显示屏上图像尺寸相同的情况下，让使用 Retina 显示屏的用户看到锐利的图像。具体参见补充材料“创建和缩放为 Retina 显示屏准备的图像”。

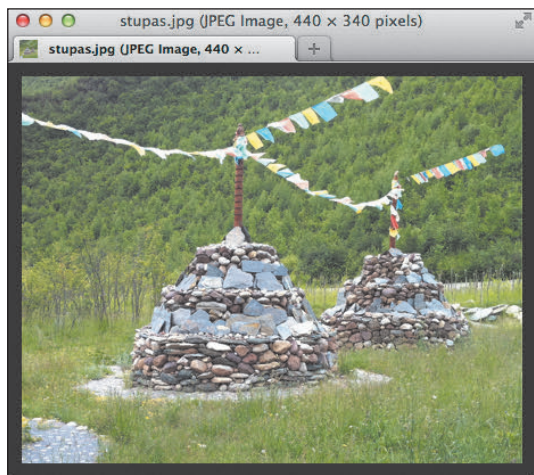


图 5.8.1 这张图在创建时使用的是双倍尺寸（440 像素 × 340 像素），但在页面中显示的尺寸只有实际尺寸的一半（即 220 像素 × 170 像素），这样就能提升 Retina 显示屏及其他高像素密度显示屏上图像的锐利性（黑色的轮廓线来自于浏览器，而不是图像本身）

在浏览器中改变图像尺寸的步骤

(1) 输入 ``，其中 `image.url` 是图像在服务器上的位置。

(2) 输入 `width="x" height="y"`，其中 `x` 和 `y` 分别是希望设定的宽度和高度（以像素为单位）。

(3) 根据需要，添加任何其他的图像属性，最后输入 `</img>`。

```
...
<figure>
  

  <figcaption>These stupas in Yunnan,
  → China, are Buddhist monuments used as
  → a place for worship.</figcaption>
</figure>
...
```

图 5.8.2 调整 `height` 和 `width` 属性均为原来的一半，由于图像的高度和宽度比例保持不变，图像就不会失真

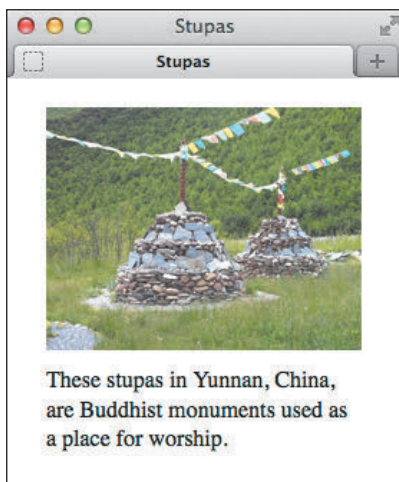


图 5.8.3 显示的图像只有原图的一半大小。不过要注意，加载它需要花的时间跟以前是一样的。毕竟，用的是同一个文件

提示 除了为 Retina 显示屏准备的图像，使用 `width` 和 `height` 属性改变图像显示在网页中的大小是一种快捷但有些丑陋的做法。由于文件本身并未改变，因此访问者容易有受骗的感觉——通常用这种方法缩小的图像总是比实际这个尺寸的图像加载得更慢。较好的做法是使用图像编辑器改变图像的尺寸。

提示 Thomas Fuchs (<http://retinafy.me>) 和 Daan Jobsis (<http://blog.netvlies.nl/designinteractie/retina-revolution/>) 讨论过一种有趣的技术,通过这项技术可以创建为 Retina 显示屏准备的图像,同时保持文件大小控制在一个合理的范围之内。其做法为:让图像尺寸变为两倍,但使用高压缩设置(在 Photoshop 中,就是选择低的图像质量)。当以一半大小显示图像时,不容易察觉到高压压缩产生的失真效果的影响。实际的效果取决于图像的内容。

创建和缩放为 Retina 显示屏准备的图像

你可能在有些人提到苹果的 iPhone、iPad 和 MacBook 时听说过 Retina 显示屏。Retina 显示屏是什么呢?

设想你仅使用点来作画,就像克洛德·莫奈或乔治·修拉^①的画。然后再设想在同样大小的画布上描绘相同的画,只是每一个点都用四个点来代替。第二张画能呈现更多的细节,同时也更难看清单独的点。

苹果的 Retina 显示屏就像第二张画。在相同的空间里,它拥有的像素数量是普通显示屏的像素数量的四倍(如图 5.8.4 所示),因此图像会更锐利。专业的说法是,它的每英寸点数(PPI)更多,或者说像素密度更大。除了苹果,还有其他一些公司的设备也使用了高像素密度显示屏。

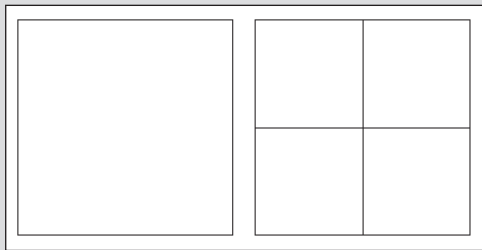


图 5.8.4 这张图有助于从视觉上理解 Retina 显示屏(右)是如何使用四个像素取代大多数显示屏(左)的一个像素的。这并非缩放,实际的像素要小得多,我想你应该能理解

需要说明的是,有时需要考虑 Retina 显示屏或类似的其他显示屏,否则图像在浏览器里看起来就会显得模糊。如果你不关心其中的原因,至少应记住,让图像的尺寸扩大为原先的两倍,但仅以一半的尺寸显示它们。

例如,如果你想让图像在所有的显示屏(并非只是 Retina 显示屏)上都是 40×30 的尺寸(此处及以下尺寸单位均为像素),就应该创建 80×60 大小的图像,而代码写作 ``。浏览器会将 80×60 的图像缩小,以 40×30 的尺寸显示(图 5.8.1 至图 5.8.3 展示了这种方法)。

^① 克洛德·莫奈(Claude Monet)和乔治·修拉(George Seurat)都是印象派画家,他们的作品往往充满了细腻缤纷的小点。——译者注

其中的原理是这样的，在 80×60 的图像中，像素总数为 4800，是原先的四倍（ 40×30 为 1200 像素）。这样，Retina 显示屏就可以显示这些额外的像素，让图像看起来更锐利。如果使用 40×30 的图像，Retina 显示屏就会拉伸这些像素以填充其对应的空间，导致锐利程度降低。每个图像的情况都不一样。

代码编写人员和设计师通常不会让每个图像的分辨率都扩大为原先的两倍，具体做法取决于开发人员自己。如果图像特别重要（如摄影师的作品集网站），强烈推荐这种做法。这样图像的大小可能会变大（要了解如何避免这种情况，参见第二条提示），而且双倍分辨率的图像会消耗设备更多的内存（这主要是移动设备的问题）。建议至少在一个移动设备上做好测试。

◎ 图标字体与 SVG

图标字体与 SVG 在缩放时都不会导致失真。对于单色的图标，建议尽可能地使用图标字体，而非使用图像（参见 13.2 节）。对标识和其他非照片类图像，可以考虑使用 SVG（关于 SVG 的说明，参见 5.1 节的补充材料）。

5.9 在图像编辑器中改变图像的尺寸

大多数图像对网页来说都太大了。一幅用于打印的图像可能有 1800 像素宽（以 300 dpi 打印，有 6 英寸宽，约 15 厘米），而用于网页的图像很少需要超过 600 像素，通常还要小得多。

使用图像编辑器既可放大图像，也可缩小图像。但是图像放大后质量通常会变差，看起来非常明显。此外，放大图像还会增加图像文件的大小，导致页面加载时间变长。

用 Photoshop 改变图像尺寸

(1) 在 Save for Web 窗口右下角的 Image Size 部分，点击 W（宽度）框或 H（高度）框，如图 5.9.1 所示。

(2) 以像素为单位输入宽度或高度，或输入百分数，再按一下 Tab 键，改变图像的尺寸，如图 5.9.2 所示。

(3) 可以继续调大或调小，直到满意为止。在按下 Save 之前，图像不会重新取样。

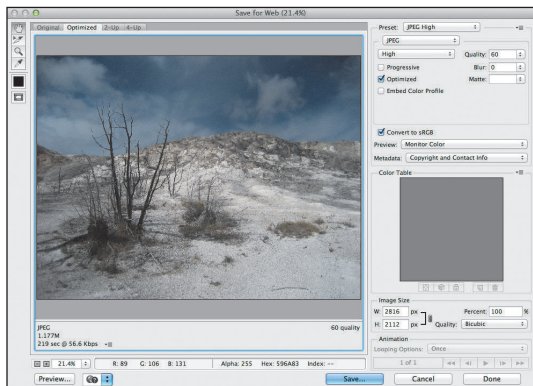


图 5.9.1 原始图像是用笔者的数码相机在默认设置下拍摄的，其尺寸为 2816 像素 × 2112 像素，对于常见的网页来说太大了，将图像压缩为高质量 JPEG 后仍然有 1.177MB。要是你在自己的网站上使用这样一张图像，肯定会让你的访问者惟恐避之不及

提示 在进入 Save for Web 之前，也可以通过 Image（图像）菜单下的 Image Size（图像大小）命令改变图像的尺寸，参见图 5.7.3 和图 5.7.4。

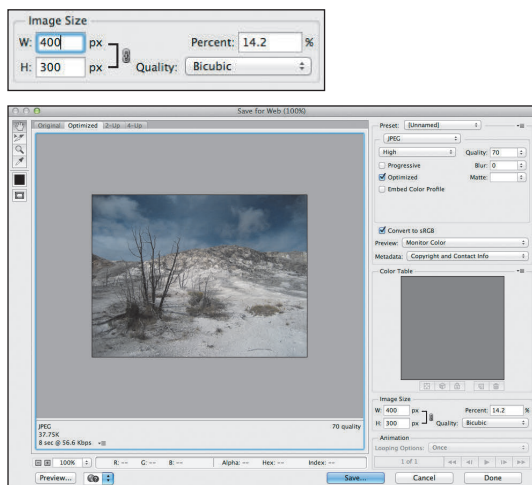


图 5.9.2 在 W 字段中输入新的宽度（这里是 400 像素），H 字段中的高度值会按照原始比例做相应改变。缩小后的图像（下图）在页面中很合适，通过改变压缩设置，文件大小降到了 37.75K

提示 减小图像尺寸的另一个好办法是将图像上不需要的区域裁剪掉。

5.10 为网站添加图标

我们在浏览器选项卡（参见图 5.10.1）、历史记录、书签页、收藏夹和地址栏中看到的（与网站相关）的小图标称为 favicon，这个词是 favorites icon（收藏夹图标）的简称。我们创建的图标至少应该为 16×16（所有图标大小单位均为像素），不过，无论如何，浏览器都会尝试尽力加载。

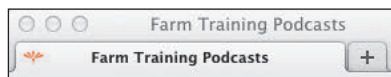


图 5.10.1 相较于浏览器的其他位置，favicon 更多地显示在标签页上（如本图所示）。Internet Explorer 是唯一将它显示在地址栏上 URL 之前的浏览器。由于浏览器通常将该图标显示在灰色或其他颜色的背景上，因此有必要将图标的背景设为透明的（如图 5.10.2 所示）

此外，还可以创建一个或多个触屏图标，即在苹果设备及其他触屏设备上将网站添加到主屏幕上时要显示的图标。苹果规定，iPhone 和 iPod touch 的图标大小为 57×57 或 114×114（对于 Retina 显示屏），iPad 的图标大小为 72×72 或 144×144（对于 Retina 显示屏）。Android 操作系统也支持这些图标。

为网站添加图标

(1) 创建一个 16×16 的图像，并保存为 ICO 格式，文件名为 favicon.ico，如图 5.10.2 所示。作为可选步骤，为 Retina 显示屏创建一个 32×32 的图像。ICO 文件允许在同一个文件中包含多个不同尺寸的同名文件。

(2)（推荐）为触屏设备至少创建一个图像（如图 5.10.3 所示），并保存为 PNG 格式。如果只创建了一个，将其命名为 apple-touch-icon.png。如有需要，还可以创建其他的触屏图标。

(3) 将图标图像放在网站的根目录里。浏览器会自动在根目录寻找这些特定的文件名，找到后就将图标显示出来。

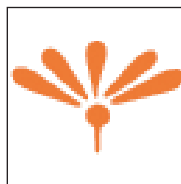


图 5.10.2 在现实环境中，favicon 比这里显示的要小，它们只有 16 像素 × 16 像素



图 5.10.3 将网站从 Safari 添加到 iOS 设备的主屏幕上会用到 apple-touch-icon

提示 有很多 ICO 格式图像编辑器，如 X-Icon Editor (<http://xiconeditor.com>)。此外，还有一些 Photoshop 插件（可以在网上搜索）。

提示 也可以将图标文件放在根目录以外的其他文件夹，不过这样的话，就需要在 HTML 中添加 link 元素，从而让浏览器得以找到这些图标（参见补充材料）。

提示 HTML5 Boilerplate (h5bp.com) 里提供了各种尺寸的触屏图标的示例。

关于 favicon 的更多信息

关于创建 favicon 的更多信息，参见 www.netmagazine.com/features/create-perfect-favicon。（注意，其中的一些信息已经过时了）。此外，Thomas Fuchs 有一篇关于 favicon 的文章，见 <http://davidwalsh.name/retina-favicons>。

本章内容

- 创建指向另一个网页的链接
- 创建锚并链接到特定的锚
- 创建其他类型的链接

链接是万维网的命脉。没有它们，每个页面都只能独立存在，同其他所有页面完全地分开。

链接有两个主要的部分：目标和标签。使用目标（destination）可以指定访问者点击链接时会发生什么。可以创建链接进入另一个页面，在页面内跳转，显示图像，下载文件，呼叫电话，等等。不过，最常见的是连接到其他网页的链接，其次是连接到其他网页特定位置（称为锚，anchor）的链接。目标是通过编写 URL 定义的，通常只能在（桌面）浏览器的状态栏中看到。

链接的第二个部分是标签（label），即访问者在浏览器中看到或在屏幕阅读器中听到的部分。激活标签就可以到达链接的目标。例如，航空公司网站上可能有这样的链接标签：预订航班。标签可以是文本、图像或二者兼有。浏览器通常会将标签文本默认显示为带下划线的蓝色文字。通过 CSS 可以很容易地改变这一样式。

6.1 创建指向另一个网页的链接

a 元素是创建链接的关键，如图 6.1.1 所示。如果你有多个网页，那么很可能希望创建从一个页面到另一个页面（以及返回）的链接（参见图 6.1.2 ~ 图 6.1.4）。还可以链接到其他网站的页面，可能是你自己制作的，也可能是他人创建的（参见图 6.1.5 ~ 图 6.1.8）。

1. 创建指向另一个网页的链接

(1) 输入 `<a href="page.html">`，其中 `page.html` 是目标网页的 URL，如图 6.1.1 所示。

(2) 输入标签文本，也就是默认突出显示的文本，如图 6.1.3 所示，访问者激活它时，就会转到第 (1) 步中所指向的页面。也可以添加一个 `img` 元素替代文本（或同文本一起）作为标签。（参见 6.3 节及补充材料“将缩略图链接到图像”。）

(3) 输入 `</a>` 结束链接定义。

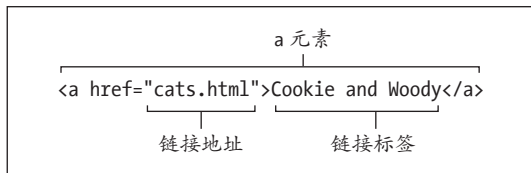


图 6.1.1 每个链接都有一个目标 URL 地址（由 href 属性指定）和一个标签。这个例子中的标签是文本，不过标签也可能是图像或者文本和图像的组合

```

...
<body>

<h1>Cookie and Woody</h1>




<p>Generally considered the sweetest and yet most independent cats in the <a href="pioneer-valley.
→ html">Pioneer Valley</a>, Cookie and Woody are consistently underestimated by their humble
→ humans.</p>

</body>
</html>

```

图 6.1.2 由于在 href 属性中只有文件名（没有路径），文件 pioneer-valley.html 必须与包含这个链接的网页位于同一个目录。否则，当用户激活该链接时，浏览器将找不到 pioneer-valley.html

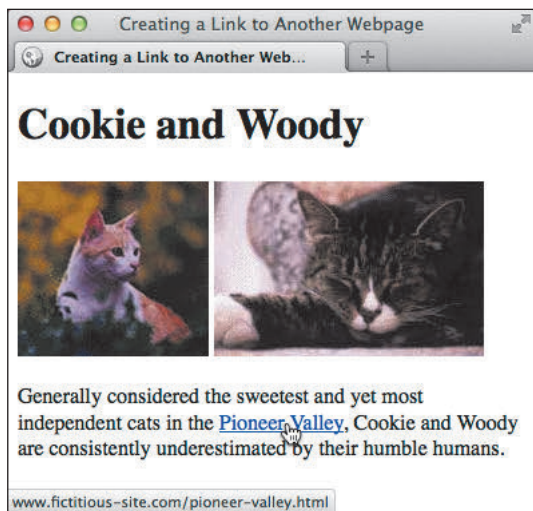


图 6.1.3 当访问者指向链接，在大多数浏览器中，目标 URL 会显示在状态栏中。（图 6.1.2 中的示例假设两个网页都位于 www.fictitious-site.com）如果用户激活该链接……

可以创建指向另一个网站的页面的链接，例如：`<a href="http://www.site.com/directory/page.html" rel="external">Label text</a>`（参见图 6.1.5 ~ 图 6.1.8）。将 href 的值替换为目标 URL 地址，rel 属性是可选的，

即便没有它，链接也能照常工作。但对于指向另一网站的链接，推荐包含这个值。它描述包含链接的页面和链接指向的页面之间的关系。它也是另一种提升 HTML 语义化程度的方式。搜索引擎也会利用这些信息。此外，还可以对带有 `rel="external"` 的链接添加不同的样式，从而告知访问者这是一个指向外部网站的链接。（参见 9.8 节。）

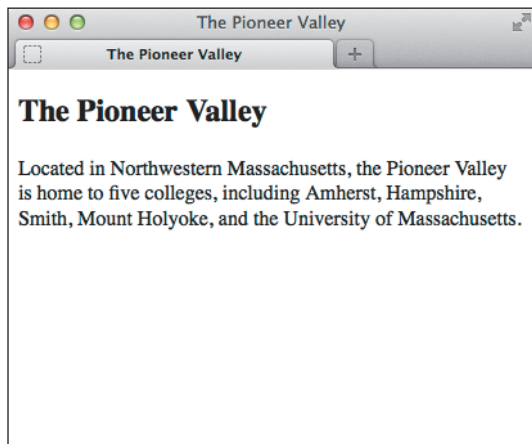


图 6.1.4 ……与这个目标 URL 相关联的页面就会显示在用户的浏览器中

通过键盘进行导航

可以通过键盘对网页进行导航。实际上，有的人由于有一些肢体障碍，无法使用鼠标这样的设备，因此不得不通过键盘对页面进行导航。

每按一次 Tab 键，焦点就会转移到 HTML 代码中出现的下一个链接、表单控件或图像映射（每按一次 Shift+Tab，焦点就会向前转移）。这个顺序不一定与屏幕上出现的顺序一致，因为页面的 CSS 布局可能不同。通过使用 HTML 的 tabindex 属性，可以改变使用 Tab 键的顺序，但不建议使用，因为在大多数情况下没必要这么做，而且这种方法已经过时了。这种方法可能导致屏幕阅读器用户迷失方向。（14.9 节演示了 tabindex 的一个有帮助的用法。）

对内容添加标记的时候，应注意保证转移焦点的顺序是符合逻辑的。应对自己的页面使用 Tab 键进行测试，就像用户的操作一样，再对 HTML 作相应的调整。（注意，如果你使用的是 Mac，这一功能可能被禁用了。如果要开启这一功能，可以在网上搜索“Enabling keyboard navigation in OS X browsers”（在 OS X 浏览器中激活键盘导航）。）

```
...
<body>

<h1>The Glory of Cats</h1>

<p><a href="http://en.wikipedia.org/wiki/
→ Cat" rel="external" title="Cat entry
→ on Wikipedia">Cats</a> are wonderful
→ companions. Whether it's a bottle cap,
→ long string, or your legs, they always
→ find something to chase around.</p>

<p>In fact, cats are so great they even have
→ <a href="http://www.catsthemusical.com"
→ rel="external" title="Official site of
→ Andrew Lloyd Webber's musical">their own
→ musical</a>. It was inspired by T.S. Eliot's
→ <cite>Old Possum's Book of Practical Cats
→ </cite>.</p>

</body>
</html>
```

图 6.1.5 如果要创建的链接是指向其他网站的，应使用绝对 URL。绝对 URL 由 http://、主机及完整路径（如果需要的话）组成。同 rel 一样，title 属性也是可选的（关于 cite 元素的用法，参见第 4 章）

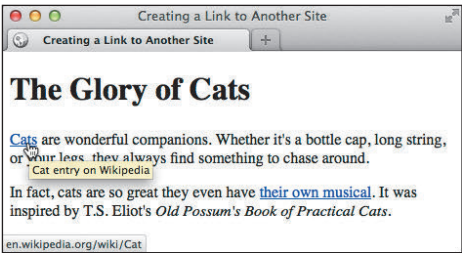


图 6.1.6 同指向站内页面的链接一样，访问者将鼠标移到指向其他网站的链接上时，目标 URL 会出现在状态栏里，title 文字（如果指定的话）也会显示在链接旁边。当访问者点击链接时……



图 6.1.7 ……目标 URL 指示的页面会显示在访问者的浏览器中

2. HTML 块级链接

目前，你已看过一些将某一元素内的一小部分文本作为链接的例子。实际上，HTML5 几乎允许在链接内包含任何类型的元素或元素组，参见图 6.1.8。例如段落、列表、整篇文章和区块——几乎任何元素都行（但其他链接、音频、视频、表单元素、`iframe` 等交互式内容除外），这些元素大部分为块级元素。使用 HTML 验证器对页面进行测试（参见 20.5 节）可以防止链接中出现不允许包含的元素。

```
...
<a href="giraffe-escapes.html">
  <p>A giraffe escaped from the zoo today,
  → and animals rejoiced worldwide.</p>
  <p>Read more</p>
</a>

... [更多标题] ...
```

图 6.1.8 恐怕大家都访问过这样的网站，以文章的一小段内容为链接，指向完整的文章。如果想让这一小段内容和提示（如“阅读全文”）都形成指向完整文章页面的链接，就应使用块级链接。可以通过 CSS 让部分文字显示下划线，或者所有的文字都不会显示下划线

这些块级链接（block-level link）是 HTML5 同 HTML 早期版本有巨大差异的地方。在以前的 HTML 中，链接中只能包含图像、文本短语，以及标记文本短语的元素（如 `em`、`strong`、`cite` 等）。

有趣的是，尽管在以前的 HTML 规范中块级链接是不允许的，但浏览器都支持。这意味着你现在就可以使用它们，而且它们在旧的浏览器和现代浏览器中均能正常工作。不过，使用它们的时候也要小心（参见图 6.1.8 和图 6.1.9）。

有一些可访问性方面的注意事项，特别是涉及不同的屏幕阅读器如何处理块级链接

的问题。无障碍访问专家 Derek Featherstone 和 Steve Faulkner 的文章深入探讨了这个问题，参见 <http://simplyaccessible.com/article/html5-block-links/> 和 www.paciellogroup.com/blog/2011/06/html5-accessibilitychops-block-links/。他们建议将最相关的内容放在链接的开头，而且不要在一个链接中放入过多内容。Featherstone 指出，随着屏幕阅读器和浏览器逐渐开始官方支持块级链接，可访问性问题可能只是暂时的。

```
...
<body>
  <a href="pioneer-valley.html">
    <h1>Cookie and Woody</h1>
    

    <p>Generally considered the sweetest
    → and yet most independent cats in the
    → Pioneer Valley, Cookie and Woody are
    → consistently underestimated by their
    → humble humans.</p>
  </a>
  ...
</body>
</html>
```

图 6.1.9 不要过度使用块级链接。应该避免这里演示的情况——将一大段内容使用一个链接包起来。尽管这样的链接是有效的 HTML5，但屏幕阅读器有可能将所有这些内容多朗读一次，多读的这些内容可能比访问者本希望听到的链接信息要多得多。因此，最好仅将与链接的含义密切相关的内容放在链接里

一般来说，用的最多的还是第一个例子（参见图 6.1.1）那样简单、传统的链接样式，不过也要知道，也用得上制作精巧的块级链接。

提示 可以改变标签文本的默认样式（参见第 10 章），甚至可以用图像作为标签（参见 6.5 节）。

提示 href 指 hypertext reference（超文本引用）。通常，对指向站内网页的链接使用相对 URL，对指向其他网站页面的链接使用绝对 URL。更多信息参见 1.7 节。

提示 仅指定路径，省略文件名，就可以创建指向对应目录下默认文件（通常为 index.html）的链接：www.site.com/directory/。如果连路径也省略，就指向网站的默认（首）页：www.site.com。

提示 在 URL 中应该全部使用小写字母，除非指向的页面或目录名称中含有大写字母。对于你自己的网站，应使用小写字母命名所有的文件夹和文件，并与链接 URL 对应，相关内容参见 1.6 节。

提示 不要让链接的标签太长。如果链接是句子的一部分，应在链接的标签定义中仅保留关键字。

提示 不管怎样，应避免使用“点击此处”作为标签。这种类型的链接文本在万维网上实在是太常见了，它会破坏网站的可用性和可访问性，从而对网站的拥有者产生不利的影响。当用户快速扫过页面上的链接（无论是通过屏幕还是通过屏幕阅读器）时，会发现“点击此处”缺乏上下文（“点击此处？为什么？”）。它对激活链接几乎不会产生激励，而且依赖于访问者阅读链接周围的文字，并寄希望于这些文字可以解释链接的目的。不难理解，访问者通常更倾向于跳过这样的链接。此外，正如本章开头提到的，“点击”一词并不适用于用户触发链接的所有方式。相反，应该使用文本中已经存在的关键字对链接进行标识。例如，应使用“了解我们的销售情况”，而不是“点击此处了解我们的销售情况”。

提示 如果你想为块级链接添加背景、边框、外边距、内边距等样式，需要在样式表中为链接设置 `display: block`（诚然，这些内容有些超前了）。关于 CSS 中的 `display` 属性，参见 11.6 节。

提示 尽管这里不会讲解图像映射的用法，但你应知道它们是用来为单张图像的一个或多个区域添加链接的。可以在网上搜索“HTML image maps”（HTML 图像映射）了解有关图像映射的更多信息。

提示 维基百科上有一份 rel 值的列表（<http://microformats.org/wiki/existing-rel-values>），该列表还在持续更新中。

提示 一定要在网站的所有页面包含指向网站各主要板块（包括首页）的导航。这可以让访问者自由地浏览网站，不管他们是直接访问网站还是通过其他网站的链接访问。你无法知道访问者会从哪里进入你的网站，可能是通过指向网站内页的“深度”链接，因此你应该让他们可以从那里开始访问网站的其余部分。

如何使用 target 属性，以及为何不要在大多数情况下使用该属性

可以将链接设置为在新的窗口或标签页（取决于所用的浏览器）中打开，但我们将其看做一种不好的实践，不推荐使用。这样说是合理的。

首先，应该让用户决定是否在不同的窗口或标签页打开链接，而不是让 HTML 开发人员决定。否则，我们就支配了用户的浏览行为。

也有一些可用性和可访问性方面的考虑。缺乏经验的用户在激活一个链接却没有在当前窗口看到结果时可能会有一些疑惑。使用浏览器并非对所有人都是一件容易的事。我曾向不同年龄段的人展示标签页，他们之前并不知道可以同时打开多个页面。类似地，屏幕阅读器等辅助设备的用户将不得不花费精力去寻找新的窗口或标签页，前提还是他们明确知道是哪个窗口或标签页加载了新的内容。

如果所有这些并未说服你避免将链接设置为在新窗口和标签页打开，或者你的老板或客户不赞同你不使用它们的理由，可以看看如何创建这种链接：在链接定义中输入 `target="window"`，其中 *window* 是应该显示相应页面的窗口的名称（由你自己选择）。

例如，`<a href="some-page.html" target="doodad">Some page</a>` 会在名为 *doodad* 的新窗口或标签页中打开 *some-page.html*。

如果让多个链接指向同一个窗口（即使用同一个名称），链接将都在同一个窗口打开。或者，如果你希望链接总是在不同的窗口或标签页打开（即使多次激活同一个链接），就使用 HTML 预定义的名称 `_blank`（`target="_blank"`）。

不过我还是不推荐你这样做，能免则免。

`target` 还有一种用法，就是在 `iframe` 中打开链接。可以用同样的方法编写 `target`，只是其值应与 `iframe` 的 `id` 对应。你很少有机会用到这个，特别是在通常不推荐使用 `iframe` 的情况下（不过它们偶尔会有用武之地）。关于 `iframe` 元素的更多信息，参见 <https://developer.mozilla.org/en/HTML/Element/iframe>。

6.2 创建锚并链接到锚

通常，激活一个链接会将用户带到对应网页的顶端。如果要想用户跳至网页的特定区域，可以创建一个锚，并在链接中引用该锚，参见图 6.2.1、图 6.2.2 和图 6.2.3。FAQ 页面可能是最常使用锚链接的页面。

1. 创建锚的步骤

(1) 将光标放在希望用户跳转至的元素的开始标签里。

(2) 输入 `id="anchor-name"`，其中 *anchor-*

name 是在内部用来标识网页中这部分内容的文字。一定要在元素名称和 `id` 之间保留一个空格，例如 `<h2 id="features">`。

2. 创建链接到特定锚的链接

(1) 输入 `<a href="#anchor-name">`，其中 *anchor-name* 是目标的 `id` 属性值（参见“创建锚的步骤”中的第 (2) 步，例如 `<a href="#features">`）。

(2) 输入标签文本，即突出显示的文本（通常默认显示为带下划线的蓝色文字），也即用户激活它时将用户带到第 (1) 步中引用的区

域的文本文本。跟通常一样，也可以将图像用做标签。

(3) 输入 `</a>` 结束对链接的定义。

```
...
<body>
<article>
  <header>
    <h1>Frequently Asked Questions (FAQ)</h1>
    <nav>
      <ul>
        <li><a href="#question-01">Can an id have more than word</a></li>
        <li><a href="#question-02">Can visitors bookmark anchor links?</a></li>
        <li><a href="#question-03">My anchor link isn't working. What am I doing wrong?</a></li>
        ...
      </ul>
    </nav>
  </header>

  <h2 id="question-01">Can an id have more than word?</h2>
  <p>Yes, your ids can have more than one word as long as there are no spaces. Separate each word
  → with a dash instead.</p>

  <h2 id="question-02">Can visitors bookmark anchor links?</h2>
  <p>Yes, they can! And when they visit that link, the browser will jump down to the anchor as
  → expected. Visitors can share the link with others, too, so all the more reason to choose
  → meaningful anchor names.</p>

  <h2 id="question-03">My anchor link isn't working. What am I doing wrong?</h2>
  <p>The problem could be a few things. First, double-check that you added an id (without "#") to
  → the element your link should point to. Also, be sure that the anchor in your link <em>is</em>
  → preceded by "#" and that it matches the anchor id.</p>

  ...
</article>
</body>
</html>
```

图 6.2.1 每个以 # 开头的链接 href 值都指向拥有相应 id (不含 #) 的元素。例如，`<a href="#question-03">...</a>` 指向 `<h2 id="question-03">...</h2>`。可以为任何元素指定 id，只要任何给定的 id 在一个页面中只存在一次（参见 3.14 节）。这个例子还让你提前看到了无序列表（ul）的应用。无序列表是目前万维网上使用频率最高的列表类型（第 15 章将深入讨论列表）

提示 为每一个锚 id 赋一个有意义的名称可以增强 HTML 文档的语义丰富度。换句话说，避免使用像 `anchor3` 这样的 id。

提示 如果锚位于另一个文档，就使用 `<a href="page.html#anchor-name">` 引用该区域。（在 URL 和 # 之间没有空格。）如果锚位于另一台服务器上的页面，则需输入 `<a href="http://www.site.com/directory/page.html#anchor-name">`（没有空格）。

提示 如果锚位于页面的底部，且它下面的内容的高度小于浏览器的可视区域的高度，那么它可能不会显示在窗口的顶部，而是显示在中间。

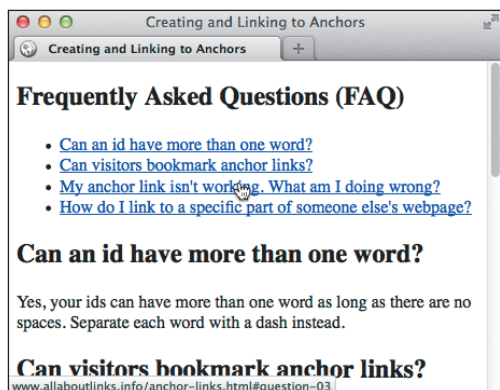


图 6.2.2 在桌面浏览器中，当访问者将鼠标指向引用锚的链接时，URL 和锚名称会显示在状态栏中（在窗口的左下角）

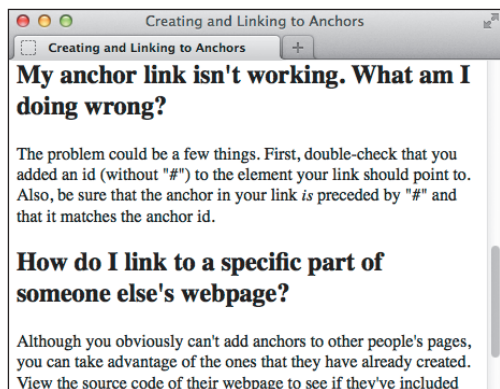


图 6.2.3 当访问者激活该链接时，锚引用的页面特定部分就会显示在浏览器窗口的顶部。这个页面比较搞笑，它链接到了打不开锚链接的情况下应该如何处理的信息

6.3 创建其他类型的链接

并非只能创建指向其他网页的链接，其

实可以创建指向任何 URL 的链接——RSS 源、图像、希望访问者可以下载的文件、电子邮件地址、电话号码等（参见图 6.3.1）。

```
...
<h1>Other Types of Links</h1>

<p>There are lots of different kinds of
→ links that you can create on a webpage.
→ Following are some examples.</p>

<h2>Images</h2>
<p>You can link directly to <a href=
→ "img/blueflax.jpg">a photo</a> or
→ even make links out of photos. For
→ example, the following image is linked
→ to a flowers photo gallery page.
→ <a href="gallery-flowers.html" title=
→ "More flower images"></a></p>

<h2>Other Assets</h2>
<p><a href="media/piano.mp3">Listen to
→ tickling of the ivories</a> (MP3, 1.3 MB)
→ or <a href="media/paddle-steamer.mp4">
→ watch a paddle steamer</a> (MP4, 2.4 MB).
→ These link directly to the files (handy for
→ downloading).</p>

<h2>Email Addresses</h2>
<p>Send feedback to <a href="mailto:
→ someone@somedomain.com">someone@
→ somedomain.com</a>.</p>

<h2>Phone Numbers</h2>
<p>Call now for free things!
→ <a href="tel:+18001234567">1
→ (800) 123-4567</a></p>
...
```

图 6.3.1 可以创建指向各种类型 URL 的链接。这个页面包含六个链接，但有的浏览器可能无法显示包围图像的链接（参见图 6.3.2）

创建其他类型链接的步骤

- (1) 输入 `<a href=`。
- (2) 输入 URL。

对于指向万维网上任何文件（包括图像、ZIP 文件、程序、PDF 及其他等）的链接，输入 `http://www.site.com/dir/file.ext`，其中 `www.site.com` 是主机名称，`dir/file.ext` 是目标文件的路径。后者包括了文件目录和文件名（以及扩展名）。

对于电子邮件地址，输入 `mailto:name@domain.com`（不以 `http://` 开头），其中 `name@domain.com` 是电子邮件地址（不过，应避免链接到电子邮件地址，参见提示）。

对于电话号码，输入 `tel:+`（不以 `http://` 开头）并紧跟着国家代码和电话号码（所有的号码中都不必包含短横线）。号码的长度在每个国家都不一样。例如，美国的国家代码是 1，因此美国的电话号码形如 `tel:+18889995555`（英国的国家代码为 44，肯尼亚为 254，依次类推）。

(3) 输入 `>`。

(4) 输入链接的标签。标签是默认以下划线和其他颜色突出显示的文本，当用户激活它时，会将访问者带到第 (2) 步中引用的 URL。也可以添加一个 `img` 元素替代文本作为标签（参见图 6.3.1 以及补充材料“将缩略图链接到图像”）。

(5) 输入 `</a>`。

提示 如果链接指向的文件是浏览器不知道如何处理类型（例如 Excel 文件），浏览器将试着打开一个辅助程序来查看这个文件，或试着将它下载到访问者的磁盘上。

提示 建议不要使用指向电子邮件地址的链接，因为垃圾邮件机器人会从网页上搜集这些地址并向其发送垃圾邮件。最好使用描述性的文字表示电子邮件地址，如“someone at some domain”，不过这样做也并非万无一失。

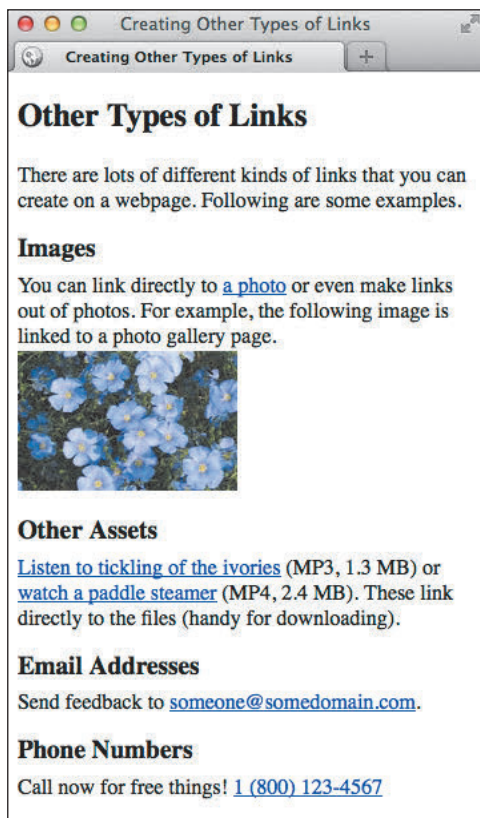


图 6.3.2 无论链接通向哪里，它们在浏览器中的默认样式都是一样的，除非其中包含图像（有的浏览器在图像链接周围添加边框，有的则不会）。注意，我尽量创建与文本主体混在一起的标签，而不是使用“点击此处”（Click me）作为标签

提示 在理解 `tel:` 的智能手机里，如果点击这样的链接，手机就会询问用户是否需要拨打该号码。对于一些不是电话的设备（如 iPad），则会询问用户是否需要将该号码添加到通讯录里。此外，有的桌面浏览器在这种情况下会启动 Google Voice 或 Skype，而其他的浏览器则不知道该如何处理这种情况。

提示 尽管可以链接到 PDF 和其他非 HTML 文档（Word、Excel 等），但应尽量避免这样做。相反，应链接到包含有关信息的 HTML 页面。PDF 可能要花更长的时间加载，而且有的浏览器在尝试显示它们的时候会变得很慢。如果 PDF 是唯一的选择，要让用户知道链接指向的是 PDF 而不是另一个 HTML 页面，以免他们感到意外（如果让用户进入耗时的下载，他们是不会领情的）。这条建议对其他非 HTML 文档也适用。显示给用户的信息可以是简单地用括号包围的文件类型和大小，也可以显示一个图标。下面是一个不包含图标的例子：`<a href="q2-sales-report.pdf">Q2 Sales Report (PDF, 725kb)</a>`。还可以在链接中包含 title 属性（如 `title="Opens a PDF"`）。

提示 对供访问者下载的大文件和文件组进行压缩是个好办法。例如，一套保存为 PSD 文件的 Photoshop 模板。在网上搜索“ZIP and RAR”，查找创建和打开使用这些流行的压缩格式的文件存档的工具。

提示 如果要“创建指向 iTunes Store、App Store、iBookstore 和 Mac App Store 上内容的链接”（链接如下），可以使用苹果公司的 Link Maker（<http://itunes.apple.com/linkmaker>）生成 URL 放在 HTML 里。如果你是联盟成员（www.apple.com/itunes/affiliates/），人们通过你的链接购买物品时，苹果公司会为你支付佣金。

将缩略图链接到图像

你肯定访问过显示几个缩略图（图像的微缩版本）的相册页面，其中缩略图链接到大一些的图像。这样就可以一次看到很多图像，让访问者选择要查看全尺寸版本的图像。

实现其基本版本类似于将鲜花图像链接到其他页面的示例代码（参见图 6.3.1）。这些页面中的每一个都包含一个全尺寸照片。（也可以在单独一个动态页面中实现，但这是超出 HTML 能力的高级用法。）

注意不要让任何给定页面上的缩略图数量太多。它们可能很小，但每个缩略图都会生成对 Web 服务器的独立请求，合在一起就会让页面变慢。没有规则规定一个页面放多少缩略图是合适的。这部分取决于页面加载的其他资源的数量和大小，也取决于网站的目标受众。例如，移动设备加载资源通常要慢一些。

因此，如果你有很多缩略图，就要考虑将它们分入多个页面。可以对页面进行测试，再确定最合适的数量。

最后，推荐使用无序列表（ul，将在第 15 章讲到）对缩略图列表进行标记。

本章内容

- 构造样式规则
- 为样式规则添加注释
- 理解继承
- 层叠：当样式发生冲突时
- 属性的值

不得不说，到目前为止，本书演示的绝大多数网页在视觉上都没有什么吸引力。这是因为我们一直将重点放在用 HTML 定义内容上，而不是放在用 CSS（层叠样式表）定义样式上。不过，这一切行将改变。在接下来的几章里，你将学会如何为文本和背景添加样式，实现多栏布局，建立起适应各种设备（从手机到台式计算机甚至屏幕更大的设备）的布局，等等。在这一章，我们将从基本的 CSS 概念讲起，为后面的内容奠定基础。

样式表不过是一种文本文件，其中包含一个或多个（通过属性和值）决定网页某特定元素如何显示的规则。CSS 里有控制基本格式的属性（如 `font-size` 和 `color`），有控制布局的属性（如 `position` 和 `float`），还有决定访问者打印时在哪里换页的打印控制元素。CSS 还有很多控制项目显示或消失的动态属性，可以用于创建下拉列表和其他交互性组件。

CSS2 是新旧浏览器支持最为广泛的版

本，因此本书将大量讨论这一版本的内容。CSS3 目前还没有成为规范，它以 CSS2 为基础，提供了大量设计人员和开发人员长期期待的功能。这些功能包括圆角、阴影效果、文字阴影、自定义字体、旋转文本、半透明背景颜色、多图像背景、渐变以及其他很多功能。值得庆幸的是现代浏览器已经实现了一些 CSS3 的组件（且即将实现更多），因此从现在起你就可以使用它们了。

实际上，CSS 在近几年又添加了很多新的特性，以致于我们无法将所有的特性尽数收入本书。哎呀，连 CSS4 都已经提上日程了。不过，本书仍会介绍一些最为有用的 CSS3 特性，并会提供用以学习本书没有提到的一些特性的资源的链接。

CSS 很棒的一点在于开发人员可以在 HTML 页面之外创建 CSS 文件，再将它应用于网站上所有的页面。这在构建网页之初及随后对其进行修改时都极大地简化了样式设置工作。一段时间后，如果需要重新设计网站，而内容和结构保持不变，就可以在 HTML 不发生变动的情况下，为网页提供一套全新的外观。

为充分利用 CSS 的优势，必须一以贯之地依照 HTML 相关章节的推荐做法标记网页。

7.1 构造样式规则

样式表中包含了定义网页外观的规则。

样式表中的每条规则都有两个主要部分：**选择器**（selector）和**声明块**（declaration block）。选择器决定哪些元素受到影响；声明块由一个或多个属性-值对（每个属性-值对构成一条**声明**，declaration）组成，它们指定应该做什么（参见图 7.1.1 ~ 图 7.1.4）。

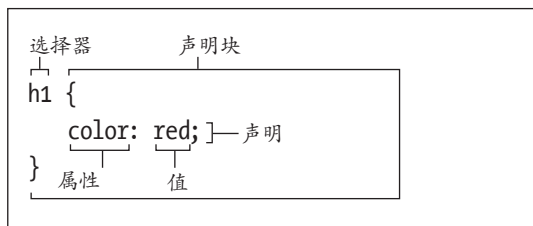


图 7.1.1 样式规则由选择器（表示将对哪些元素进行格式化）和声明块（描述要执行的格式化）组成。声明块内的每条声明都是一个由冒号隔开、以分号结尾的属性-值对。声明块以前花括号开始，以后花括号结束

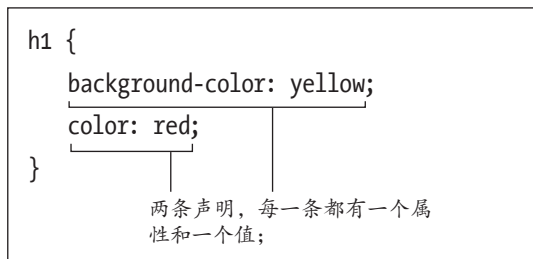


图 7.1.2 声明的顺序并不重要，除非对相同的属性定义了两次。在这个例子中，color: red 也可以放在 background-color: yellow 前面，效果是一样的。注意额外的空格和缩进（可选，但推荐包含）提高了可读性

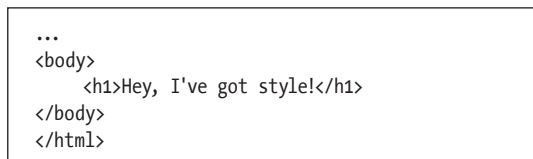


图 7.1.3 如果将图 7.1.2 中的样式规则应用到下面这一小段 HTML 文档中……



图 7.1.4 标题从浏览器默认的白色背景、黑色文字样式变为黄色背景、红色文字（另见彩插）

构造样式规则的步骤

(1) 输入 *selector*，这里的 *selector* 表示希望进行格式化的元素。第 9 章将讲解如何创建各种类型的选择器。

(2) 输入 {（前花括号）开始声明块。

(3) 输入 *property:value;*，其中 *property* 是 CSS 属性的名称，描述要应用哪种格式；*value* 是该属性允许的选项之一。本书从第 10 章开始讲解 CSS 属性和值。

(4) 根据需要，重复第 (3) 步。通常一行输入一个 *property: value*（一条声明），如图 7.1.2 所示的那样，但这并非强制要求。

(5) 输入 }，结束声明块和样式规则。

提示 在样式规则中可以添加额外的空格、制表符或回车，从而提高样式表的可读性（参见图 7.1.2）。示例中的格式或许是编码人员中最为常见的一种格式。

提示 从技术上讲，可以省略样式规则中最后一条声明后面的分号，不过最好加上，这是一种好习惯。

7.2 为样式规则添加注释

在 CSS 中添加注释是个好主意，这样就可以标注样式表的主要区域，或者只是针对某条规则或声明进行解释。注释不仅对开发人员有用，对阅读代码的其他人也有好处。

即便你是网站的唯一开发人员，当你过几个月再看自己写的代码时，会庆幸留下了这些注释。

为样式规则添加注释

- (1) 在样式表中，输入 `/*` 开始注释。
- (2) 输入注释。
- (3) 输入 `*/` 结束注释。

提示 注释可以包含回车，因此可以跨越多行。类似地，`/*` 和 `*/` 既可以单独成行，也可以跟文本同处一行，图 7.2.1 中这两种例子都有。另外，注释还可以放到声明块内部，或者样式规则后面，参见图 7.2.2。

```
/*
This is a CSS comment. It can be one line
→ long or span several lines. This one
→ is much longer than most. Regardless, a
→ CSS comment never displays in the
→ browser with your site's HTML content.

Of course, you wouldn't really write a
→ silly comment like this that merely
→ talks about comments. The next comment
→ is more in line with a comment's
→ typical use.
*/

/* Set default rendering of certain HTML5
→ elements for older browsers. */
article,
aside,
figcaption,
figure,
footer,
header,
main,
nav,
section {
    display: block;
}
```

图 7.2.1 注释可长可短，但它们通常较短。可以根据你的想法，用它们描述样式规则或一组相关的规则的目的。注释做得好可以让样式表更易于维护

```
.byline {
    color: green;
    font-size: .875em;
    text-shadow: 2px 1px 5px orange; /* IE9
→ and earlier don't support */
} /* You can put comments here, too! */
```

图 7.2.2 我们也可以在声明块内部或者样式规则后面插入注释

提示 不能将一个注释嵌套在另一个注释内部，下面的做法是不正确的：

`/*` 做法错误，因为 `/*` 这条注释 `*/` 嵌套在外面这条注释内部 `*/`。

提示 注释是很有用的组织工具。样式表很快就会变得很长，因此，组织好样式表对于保持 CSS 易于维护至关重要。通常，将相关的规则放在一起，形成分组，并在每组前面放置一段描述性的注释，参见图 7.2.3。

```
/* GLOBAL NAVIGATION
----- */
... rules for global nav ...

/* MAIN CONTENT
----- */
... rules for main content ...

/* SIGN-UP FORM
----- */
... rules for sign-up form ...

/* PAGE FOOTER
----- */
... rules for page footer ...
```

图 7.2.3 对样式表中的主要区域添加注释，就可以保持样式表井然有序。下面的格式（使用大写字母和一条下划线）可以很清楚地标识分组的开始位置

提示 无论采用哪种注释格式，推荐你定下一种并坚持使用，特别是在团队协作的情况下。

提示 可以将注释放在样式规则内部（参见图 7.2.4）或者周围（参见图 7.2.5），从而对浏览器隐藏样式规则。这是对样式表进行测试的一种好方法，不必永久性地删除注释部分，直到你认为可以删除它们。这也是很有用的调试工具，可以将你认为可能引起问题的地方“注释掉”，再在浏览器中刷新页面，查看问题是不是解决了。

```
img {
  border: 4px solid red;
  /* margin-right: 12px; */
}
```

图 7.2.4 可以将你不希望对页面产生影响的声明“注释掉”。这里，所有的图像都会拥有一条 4 像素宽的红色实线边框，但不会有右侧外边距，因为 `margin-right: 12px;` 位于一条注释里面

```
p {
  line-height: 1.2;
}

/*
.byline {
  color: black;
  font-size: .875em;
  text-shadow: 2px 1px 5px orange;
}

img {
  border: 4px solid red;
  margin-right: 12px;
}
*/
```

图 7.2.5 注释也可以包围整条规则或者多条规则。在下面这条样式规则中，只应用了段落的 `line-height`，`.byline` 和 `img` 规则被注释掉了。注意，我删除了这两个样式原来的注释（在图 7.2.2 和图 7.2.4 中显示了），这样再在它们周围添加注释就不会导致错误了

提示 出于演示的目的，这些例子都使用了很多注释，但不要因此认为要对所有内容添加注释。如果样式表里的注释过多，反而会难以阅读。要根据需要混合使用组织性注释和描述性注释。要针对你和团队里的其他成员，找到一个平衡点。

7.3 理解继承

继承（inheritance）是 CSS 里一个很重要的概念。看看图 7.3.1 所示的网页。浏览器会将这份 HTML 理解为图 7.3.2 所示的文档树。文档树有助于你理解 CSS。原因是，当你为某个元素应用 CSS 属性时，这些属性不仅会影响该元素，还会影响其下的分支元素。也就是说，这些下层的元素继承了其祖先元素的属性。只不过，它们继承的可不是冷冰冰的钞票，而是颜色、字体大小这样的东西。

```
...
<body>
<div>
  <h1>The Ephemeral Blue Flax</h1>

  <p>I am continually <em>amazed</em>
  → at the beautiful, delicate Blue Flax
  → that somehow took hold in my garden.
  → They are awash in color every morning,
  → yet not a single flower remains
  → by the afternoon. They are the very
  → definition of ephemeral.</p>

  <p><small>&copy; Blue Flax Society.
  → </small></p>

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

图 7.3.1 所有的内容元素都是 `body` 元素的后代。在这个例子中，用一个 `div` 包住了所有的内容。进一步，`em` 和 `small` 元素都包含在一个 `p` 元素里，因此它们是 `p` 的后代（同时也是 `div` 和 `body` 的后代），如图 7.3.2 所示

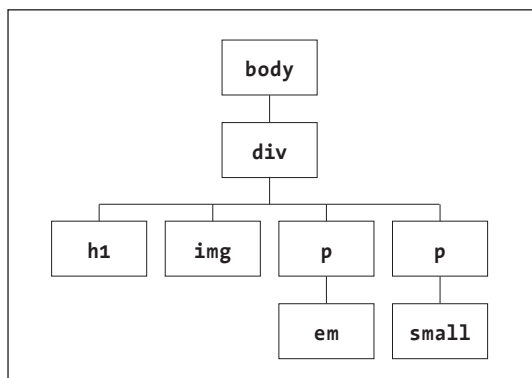


图 7.3.2 将 HTML 页面转换为树形结构后，很容易看出哪些元素是哪些元素的后代。任何一个直接包含在另一个元素中的元素（例如在图 7.3.1 中，img 元素由 div 直接包含），都是其父元素的分支

```

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

div {
  border: 1px solid #000;
  overflow: hidden;
  padding: 0 1em .25em;
}

p {
  color: #36c; /* a blue color */
  font-weight: bold;
}

img {
  float: left; /* makes text wrap it */
  margin-right: 1em;
}
  
```

图 7.3.3 这段应用于图 7.3.1 中 HTML 的样式表在为页面设置样式时体现了继承的特性。如果现在不理解其中的细节没有关系，只需注意这里只有 body、div 和 p 元素的样式规则，而没有 h1、em 和 small 元素的规则

图 7.3.1、图 7.3.3 和图 7.3.4 演示了继承的实际效果。例如，font-family 会被继承，而 border 和 padding 则不会继承。这也是为什么么页面上的文字使用 Verdana 字体，而只有

div 拥有边框（黑色的细线方框）和内边距（边框和 div 中内容之间的空间），它的后代则没有这些样式，如图 7.3.4 所示。此外，color 和 font-weight 属性也是继承的，因此 em 和 small 元素里的文本也像其他段落文本一样显示为蓝色粗体文字，而不是浏览器默认的黑色非粗体文字。补充材料“哪些属性会被继承”列出了其他一些会被继承的属性。

可见，继承可以简化样式表。设想你要为页面中的每个元素单独设置字体，多么恐怖！在编写 CSS 时，要牢记这一规则，并有效地利用继承。

此外，对大多数属性来说，还可以使用 inherit 值强制进行继承（参见 7.5 节）。



图 7.3.4 这里展示了继承的实际效果（以及未继承的效果）。h1 从 body 中继承了 Verdana 字体样式（由 font-family 设置）。它在图 7.3.3 中没有自己的样式，因此它还会根据浏览器的默认样式进行显示，即显示为黑色粗体的标题样式。类似地，em 和 small 元素也没有指定具体的样式规则，但它们也从 body 继承了 Verdana 字体样式，从 p 继承了 font-weight 和 color 的样式。用 small 标记的文字（法律声明）比周围的文字小一些，这也是浏览器的默认样式。最后，包含所有内容的 div 有一个细线边框和内边距，但其中的其他元素没有这样的样式，因此这些属性不会被继承

哪些属性会被继承

以下是会被继承的 CSS 属性，我们按照类型对其进行了分组。这些属性中的大多数都将在本书后续章节进行讲解，不过你可以根据它们的名称猜出其作用。

◎ 文本

- ☐ color (颜色, a 元素除外)
- ☐ direction (方向)
- ☐ font (字体)
- ☐ font-family (字体系列)
- ☐ font-size (字体大小)
- ☐ font-style (用于设置斜体)
- ☐ font-variant (用于设置小型大写字母)
- ☐ font-weight (用于设置粗体)
- ☐ letter-spacing (字母间距)
- ☐ line-height (行高)
- ☐ text-align (用于设置对齐方式)
- ☐ text-indent (用于设置首行缩进)
- ☐ text-transform (用于修改大小写)
- ☐ visibility (可见性)
- ☐ white-space (用于指定如何处理空格)
- ☐ word-spacing (字间距)

◎ 列表

- ☐ list-style (列表样式)
- ☐ list-style-image (用于为列表指定定制的标记)
- ☐ list-style-position (用于确定列表标记的位置)
- ☐ list-style-type (用于设置列表的标记)

◎ 表格

- ☐ border-collapse (用于控制表格相邻单元格的边框是否合并为单一边框)
- ☐ border-spacing (用于指定表格边框之间的空隙大小)
- ☐ caption-side (用于设置表格标题的位置)
- ☐ empty-cells (用于设置是否显示表格中的空单元格)

◎ 页面设置 (对于印刷物)

- ☐ orphans (用于设置当元素内部发生分页时在页面底部需要保留的最少行数)
- ☐ page-break-inside (用于设置元素内部的分页方式)
- ☐ widows (用于设置当元素内部发生分页时在页面顶部需要保留的最少行数)

◎ 其他

- cursor (鼠标指针)
- quotes (用于指定引号样式)

7.4 层叠：当规则发生冲突时

样式的来源很多。正如你在第 1 章中学到的，每个浏览器都有其默认样式。不过，你可以用自己的样式覆盖它们或对它们进行补充。应用样式有三种方式：从一个或多个外部文件导入（推荐），如图 7.4.1 中的代码所示，插入到 HTML 文档的顶部，或直接应用于代码中特定的 HTML 元素上（不过，应该尽可能地避免这样做）。下一章将具体讲解三种方式。

你可能会问，对某一给定元素应用多条样式规则时，会发生什么情况？比如一条规则定义了一个元素的颜色，而另一条规则定义了它的宽度，这两条规则会有效结合，一起应用到元素上。

```
p {
    color: red;
}

.example {
    color: blue;
}

.example.example-2 {
    color: magenta;
    /* 会被下一条规则覆盖 */
}

.example.example-2 {
    color: green;
}
```

图 7.4.1 在这个例子中，有四条特殊性不尽相同的样式规则。第一条影响所有 p 元素，第二条仅影响图 7.4.2 的 HTML 中类名为 example 的元素，第三条和第四条则影响类名同时包含 example 和 example2 的元素。这些规则的顺序并不重要（除了最后两条，因为它们的选择器完全相同）

```
...
<link rel="stylesheet" href="style.css" />
</head>
<body>

<p>Here's a generic <code>p</code> element. It will be red.</p>

<p class="example">Here's a <code>p</code> element with a <code>class</code> of <code>example
→ </code>. There are two rules that could apply, but since the <code>.example</code> selector is
→ more specific, this paragraph will be blue.</p>

<p class="example example-2">Here's a <code>p</code> element with two classes: <code>example
→ </code> and <code>example-2</code>. There are four rules that could apply to this paragraph. The
→ first two are overruled by the more specific last two. However, because the last two have the
→ same selector, the order breaks the tie between them: the one that appears later wins, and thus
→ this paragraph will be green instead of magenta.</p>

</body>
</html>
```

图 7.4.2 这里有三个段落：一个普通的，一个带有一个类名，一个带有两个类名。注意，如果一个元素需要包含一个以上的类名，在 HTML 中应使用空格分隔不同的类名，但在 CSS 中不能像这样定位元素。在 CSS 中，类名的前面使用点号（.）。图 7.4.1 中的样式表文件名为 style.css，第二行中的 link 标签会将该文件加载到页面中，这些内容将在下一章进行讲解

然而，有时候多条规则会定义元素的同一个属性（参见图 7.4.1），这时该怎么办呢？CSS 用层叠的原则来考虑样式声明，从而判断相互冲突的规则中哪个规则应该起作用。首先，你编写的样式如果与浏览器的默认样式冲突，均以你编写的样式为准。在此基础上，CSS 用层叠的原则来考虑特殊性（specificity）、顺序（order）和重要性（importance），从而判断相互冲突的规则中哪个规则应该起作用。不要受这些术语的影响，你只要去试，就能明白 CSS 决定该应用哪些样式以及何时应用这些样式的方式。

接下来，我们详细介绍特殊性、顺序和重要性。

1. 特殊性

特殊性规则指定选择器的具体程度。选择器越特殊，规则就越强。遇到冲突时，优先应用特殊性强的规则，如图 7.4.3 和图 7.4.4 所示。讲得通，对吧？

我们不限于使用示例中显示的基本选择器来定义样式。第 9 章会涉及特殊性权重各

不相同的其他类型的选择器。我们认为 id 选择器是最特殊的（因为它们在一个文件中必须是唯一的），如图 7.4.4 所示。另一方面，除了浏览器默认样式，我们认为继承的样式是最一般的，可以被任何其他规则覆盖。

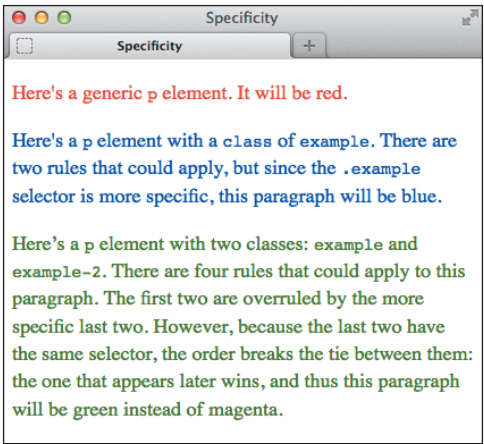


图 7.4.3 第二个段落是蓝色的，因为它的类名包括 example。对于第三个段落，由于第三条和第四条规则的特殊性相同，因此它们的顺序变为决定性因素——第四条规则后出现，因此它起作用

选择器	对应的 HTML
p { ... }	...
.someClass { ... }	<p class="someClass">...</p>
.someClass.someOtherClass { ... }	<p class="someClass someOtherClass">...</p>
#someID { ... }	<p id="someID">...</p> <p id="someID" class="someClass">...</p> <p id="someID" class="someOtherClass">...</p> <p id="someID" class="someClass someOtherClass">...</p>

图 7.4.4 这里列出了几个选择器，按照特殊性由低向高排列。其中，特殊性最低的是元素名本身（最上面），特殊性最高的是 ID（最下面）。图 7.4.1 ~ 图 7.4.3 反映了这一行为规则

ID 选择器将在第 9 章进行讲解（参见 9.3 节），不过现在要提一句，建议在样式表中多使用类选择器，避免使用 ID 选择器。使用 ID 选择器通常会矫枉过正，降低了灵活性。

2. 顺序

有时候，特殊性还不足以判断在相互冲突的规则中应该优先应用哪一个。在这种情况下，规则的顺序就可以起到决定作用：晚出现的优先级高（参见图 7.4.1、图 7.4.2 和图 7.4.3）。例如，直接应用在 HTML 元素上的规则（不推荐使用这种做法）被认为比外部样式表中或插在 HTML 文档顶部的特殊性相同的规则出现得更晚（因此优先级更高）。

3. 重要性

如果这还不够，可以声明一条特殊的规则覆盖整个系统中的规则，这条规则的重要程度要比其他所有规则高。也可以在某条声明的末尾加上 `!important`，比如 `p{ color: orange !important; }`（除非是在特殊情况下，否则不推荐使用这种方法）。

相关细节参见 8.5 节。

4. 小结

你编写的样式会覆盖浏览器的默认样式。当两个或两个以上的样式发生冲突时，会应用特殊性高的样式声明，不管它位于样式表中的哪个位置。如果两个或两个以上的规则拥有相同的特殊性，则使用后出现的规则，除非其中某条规则标记了 `!important`。

如果某元素没有直接指定某条规则，则使用继承的值（如果有的话）。

如果这些内容看起来有些令人困惑，现在也不必担心。当你试着编写 CSS，开始使用一些选择器后，你会发现，在大多数情况下，层叠的运行规则跟你设想的是一样的。

提示 如果你需要复习 HTML 中的类和 ID 名称，参见 3.14 节。

提示 如果你对浏览器计算特殊性权重的具体细节感兴趣，可以查看 CSS 规范的第 9 部分（www.w3.org/TR/selectors/#specificity）。注意，在学习本书第 9 章之前，你可能会对规范中的内容感到陌生。其实，编写 CSS 不必死记硬背公式化的内容。几乎没有多少代码编写人员能背下这些特殊性规则，即便那些具有多年经验的程序员也是如此。他们（以及你）不必死记硬背的原因是 CSS 特殊性规则实际上相当直观。如果一条样式并未按照你设想的方式呈现，很有可能是其特殊性不够高，这样，你就可以从编码的过程中得到经验。

提示 先前提到，样式表可能来自于浏览器和代码编写人员。实际上还有第三种来源，即访问者（尽管这种来源并不常见）。有的浏览器允许用户创建自己的样式表并将其应用到他们所访问的页面（包括代码编写人员的页面），从而根据自己的喜好定制体验。例如，一个有视觉障碍的用户可能希望文本和背景颜色的对比度更大一些。实际上，没有多少用户知道浏览器的这一特性，使用它的用户就更少了。这些样式确实有可能影响层叠。它们的优先级高于浏览器的默认样式，但低于你为网站创建的 CSS（除非用户的样式表声明中标记了 `!important`）。提示这一点主要是为了让你了解这种用户行为，不必因此对设置网页样式的方法作出任何改变。

7.5 属性的值

每个 CSS 属性对于它可以接受哪些值都有不同的规定。有的属性只能接受预定义的值。有的属性接受数字、整数、相对值、百分数、URL 或者颜色。有的属性可以接受多种类型的值。每个属性可接受的值将在描述该属性的章节（大多在第 10 章、第 11 章和第 14 章）列出来，不过这里将讨论基本体系。

1. inherit

对于任何属性，如果希望显式地指出该属性的值与对应元素的父元素对该属性设定的值相同，就可以使用 `inherit` 值。例如，假设有一个 `article` 元素，其中包含几个段落。`article` 元素设置了一个边框。边框通常不会被继承，因此 `p { border: inherit; }` 这条规则可以让这些段落获得相同的边框样式。（在 IE8 之前的 Internet Explorer 版本中，大部分属性都不支持 `inherit` 值，不过你大概也不需要支持那些大多数网站都不再支持的旧浏览器。）

2. 预定义的值

大多数 CSS 属性都有一些可供使用的预定义值。例如，`float` 属性可被设为 `left`、`right` 或 `none`。与 HTML 不同，不需要（也不能）将预定义的值放在引号里，参见图 7.5.1。实际上，大多数 CSS 值，无论是否为预定义的值，都不需要加引号。（也有一些例外，如超过一个单词的 `font-family` 名称，以及要生成的内容。）



图 7.5.1 很多 CSS 属性只接受预定义的值。要准确地输入这些值，并且不要加上引号

3. 长度和百分数

很多 CSS 属性的值是长度。所有长度都必须包含数字和单位，并且它们之间没有空格。例如 `3em`、`10px`，参见图 7.5.2。唯一的例外是 `0`，它可以带单位也可以不带。效果是一样的，因此 `0` 一般不带单位。

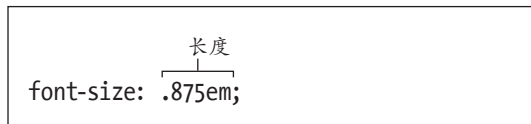


图 7.5.2 必须显式指出长度的单位。在单位和数值之间应该没有空格

有的长度是相对于其他值的。一个 `em` 的长度大约与对应元素的字号相等。例如，对元素设置 `margin-left: 2em` 就代表将元素的左外边距设为元素字号的两倍。（当 `em` 用于设置元素的 `font-size` 属性本身时，它的值继承自对应元素的父元素的字号，第 10 章会讨论这一点。）`em` 的这种相对性对如今的网站建设工作来说是至关重要的，尤其是对那些需要适应不同屏幕尺寸的网站来说（这种做法被称为响应式 Web 设计，将在第 12 章介绍）。此外，`rem` 单位是相对于 `html` 元素的字体大小的，参见 10.6 节。

如果用过 Adobe Photoshop 这样的软件，你一定很熟悉像素的概念。跟 `em` 不同，像素（`px`）并不是相对于其他样式规则的。因此以 `px` 为单位的值不会像以 `em` 为单位的值那样受 `font-size` 设置的影响。不过，在如今设备种类繁多的情况下，已经很难再准确度量一个像素的实际大小了。某种设备上 1 个像素的大小不一定与另一种设备上的完全相等。（参见 Peter-Paul Koch 对此问题的详细说明，www.quirksmode.org/blog/archives/2010/04/a_pixel_is_not.html。警告：有些内容可能不好懂。）有时候我们需要使用 `px`，只是没那么

频繁。

还有一些无需说明的绝对单位，如磅（pt），应该在为打印准备的样式表中保留这个单位。一般来说，应该只在输出尺寸已知的情况下使用绝对长度（如在打印的页面中使用 pt）。

百分数（如 65%）的工作方式很像 em，它们都是相对于其他值的值，参见图 7.5.3。出于这个原因，百分数是创建我们很快就要学习的响应式网站的一个强大工具。

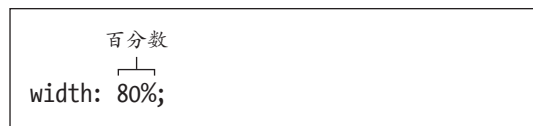


图 7.5.3 百分数通常是相对于父元素的。因此，在这个例子中，字号将被设为父元素字号的 80%

在上述单位中，最常使用的是 em、百分数和像素，rem 的使用情况越来越多。你的样式表可以自由组合使用上述单位，即使是在同一条样式规则中，也可以组合使用不同的单位。还有其他一些单位，不过鲜有使用。

4. 纯数字

只有极少数的 CSS 属性接受不带单位的数字，如 3、0.65。其中最常见的是 line-height（参见图 7.5.4）、z-index 和 opacity（分别参见 10.7 节、11.15 节和 14.8 节）。

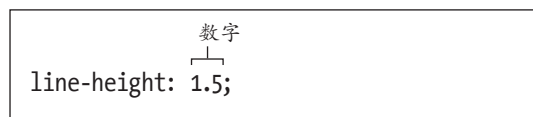


图 7.5.4 不要将数字、整数和长度弄混。数字或整数没有单位（如 px）。这个例子中的值将与字号相乘，得到行高

5. URL

有的 CSS 属性允许开发人员指定另一个

文件（尤其是图像）的 URL，background-image 就是这样一个属性。在这种情况下，使用 url(file.ext)，其中 file.ext 是目标资源的路径和文件名，参见图 7.5.5。注意，规范指出，相对 URL 应该相对于样式表的位置而不是相对于 HTML 文档的位置。

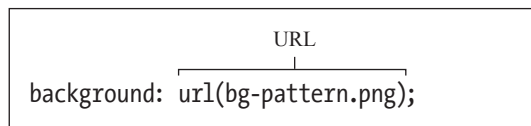


图 7.5.5 CSS 属性中的 URL 无需使用引号包围

可以在文件名上加上引号，但这不是必需的。此外，在单词 url 和前括号之间不应该有空格。括号和地址之间允许有空格，但这不是必需的（通常也不这样做）。

关于编写 URL 本身的信息，参见 1.7 节。

6. CSS 颜色

我们可以使用预定义颜色关键字或以十六进制（通常称为 hex）、RGB、HSL、RGBA、HSLA 等格式表示的值中为 CSS 属性指定颜色。最后两种格式可以指定具有一定程度 alpha 透明度的颜色。HSL、RGBA 和 HSLA 格式都是在 CSS3 中引入的。

CSS3 指定了 CSS2.1 本来就有的 16 个基本的名称（参见图 7.5.6），另外又增加了 131 个。完整的列表见 www.w3.org/TR/css3-color/#svg-color。

当然，除了几样最简单的名称，没有人记得住这些颜色名。可以使用 Adobe Photoshop 等工具进行取色，但它们并不使用 CSS 颜色名，你获取的是 RGB 或十六进制值。此外，颜色关键字也只是你使用的颜色中很小的一部分，因此在实践中，定义 CSS 颜色更常规的方法是使用十六进制格式（这是目前为止最为常见的方式）或 RGB 格式。当然，

要指定 alpha 透明度的情况除外，这时应该用 RGBA 和 HSLA 格式。

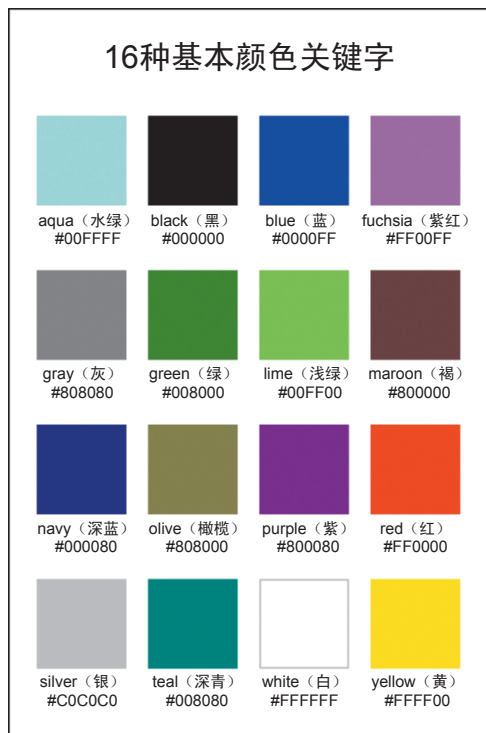


图 7.5.6 16 种基本颜色关键字，以及它们对应的十六进制数字值。CSS3 新增了 131 个关键字，不过通过十六进制、RGB 或 HSL 等格式可以定义多得多的颜色。另见彩插

7. RGB

可以通过指定红、绿、蓝（这也是 RGB 这一名称的由来）的量来构建自己的颜色。可以使用百分数、0 ~ 255 之间的数字来指定这三种颜色的值。例如，如果创建一种深紫色，可以使用 89 份红、127 份蓝、0 份绿。这个颜色可以写做 `rgb(89, 0, 127)`，如图 7.5.7 所示。

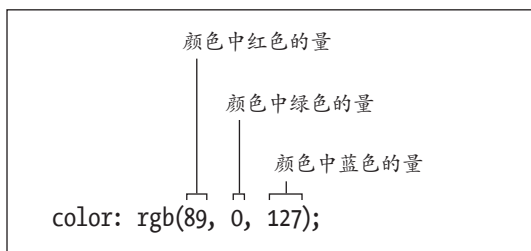


图 7.5.7 表示 CSS 颜色的另一种方式是用 0 ~ 255 之间的数字指示 RGB 数值。首先定义红色，然后是绿色，最后是蓝色

此外，也可以将每个值表示为百分数，不过很少用到这种做法，可能因为 Photoshop 等图像编辑器通常用数字表示 RGB 值。如果你想使用百分数，可以将上面的颜色写作 `rgb(35%, 0%, 50%)`，因为 89 是 255 的 35%，127 是 255 的 50%。

8. 十六进制数

我把最常用的方法放在最后讲解，参见图 7.5.8。将这些数字值转化为十六进制，然后将它们合并到一起，再在前面加一个 #，就像 `#59007F` 这样。你心里可能想：“为什么不用十进制呢？”别慌，类似于 Photoshop 这样的工具在选择颜色时可以显示颜色的 RGB 值，以及对应的十六进制数。此外你也可以上网搜索免费的转换工具（如输入 `rgb to hex converter`）。

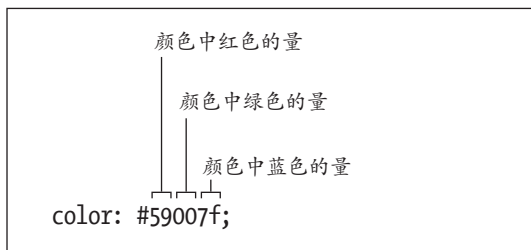


图 7.5.8 CSS 中定义颜色最常用的方式是用十六进制数指定颜色所包含的红、绿、蓝的量

对于 #59007F，十六进制的 59、00、7F 分别等于 89、0、127。7F 和 7f 都是允许的写法（我倾向于使用后一种写法，不过很多开发人员和设计人员都选择前一种）。

如果一个十六进制颜色是由三对重复的数字组成的，如 #ff3344，可以缩写为 #f34。这种做法也是一种最佳实践，因为没有理由让代码无谓地变长。

9. 更多 CSS3 提供的指定颜色的方式：RGBA、HSLA 和 HSL

CSS3 引入了另一种指定颜色的方式——HSL，以及通过 RGBA 和 HSLA 设置 alpha 透明度的能力。（使用十六进制记法无法指示 alpha 透明度。）

◎ RGBA

RGBA 在 RGB 的基础上加了一个代表 alpha 透明度（alpha transparency）的 A。alpha 透明度的一种常见使用场景是将其用在对元素设置 background-color 或 background 的情况（均用于设置背景），参见图 7.5.9 和图 7.5.10，因为 alpha 透明度允许元素下面的任何东西（如图像、其他颜色、文本等）透过来并与元素混合在一起（如图 7.5.11 所示）。也可以对其他基于颜色的属性使用 alpha 透明度，如 color、border、border-color、box-shadow、text-shadow 等。所有现代浏览器都支持 alpha 透明度，但 IE9 之前的 Internet Explorer 不支持（如图 7.5.12 所示）。

可以在红、绿、蓝数值后面加上一个用以指定透明度的 0 到 1 之间的小数。语法如下：

```
property: rgba(red, green, blue, alpha  
→ transparency);
```

alpha 设置越接近 0，颜色就越透明。如果设为 0，就是完全透明的，就像没有设置任何颜色。类似地，1 表示完全不透明。

为了解释上面的说法，下面给出了一些声明示例：

```
/* 不透明，和 rgb(89, 0, 127); 效果相同 */  
background-color: rgba(89,0,127,1);  
/* 完全透明 */  
background-color: rgba(89,0,127,0);  
/* 25% 透明 */  
background-color: rgba(89,0,127,0.75);  
/* 60% 透明 */  
background-color: rgba(89,0,127,0.4);
```

当然，为了让这些记法正常运行，需要将它们放入一个或多个规则中（参见图 7.5.9）。可以看到，尽管 RGB 颜色值是一样的，但由于设置了不同的透明度，颜色在浏览器中显示的样子就不一样了，参见图 7.5.11。

◎ HSL 和 HSLA

HSL 和 HSLA 也是 CSS3 中新增的。HSLA 是除 RGBA 以外为颜色设置 alpha 透明度的另一种方式。用 HSLA 指定 alpha 透明度的方法与 RGBA 是一致的。待会儿再看 HSLA，先看看 HSL 的工作原理。

HSL 代表色相（hue）、饱和度（saturation）和亮度（lightness），其中色相的取值范围为 0 ~ 360，饱和度和亮度的取值均为百分数，范围为 0 ~ 100%，如图 7.5.13 所示，0 和 360 在顶部汇合，意思是 0 和 360 代表同一种颜色：红色。饱和度和亮度设置会直接应用到颜色上（不要将 HSL 和 HSB、HSV 混淆。它们很相似，但不是一回事，CSS 不允许用 HSB 和 HSV 表示颜色）。

```

/* 设置重复页面背景图像和默认文字颜色、字体、
   字号 */
body {
    background: url(..img/bg-pattern.png);

    /* 由于继承原因，这些文字样式会应用到所
       有的文字，除非定义了其他样式覆盖前面
       的样式 */
    color: #fff0; /* 十六进制黄色 */
    font-family: arial, helvetica,
    → sans-serif;
    font-size: 100%;
}

/* 所有段落都会显示纯色背景，除非指定下列类别
   之一。IE9之前的版本不支持透明色，无论如何都
   会显示背景色 */
p {
    background-color: rgb(60,143,0);
    font-size: 1.75em;
    padding: .5em;
}

/* 25% 透明度 */
.test-1 {
    background-color: rgba(60,143,0,.75);
}

/* 60% 透明度 */
.test-2 {
    background-color: rgba(60,143,0,.4);
}

```

图 7.5.9 这份简单的样式表对整个页面应用了一个重复显示的背景图，设置了默认文字颜色，并依据类别的不同对各个段落添加了不同的背景，参见图 7.5.10。现代浏览器显示的结果如图 7.5.11 所示。IE9 之前的版本不支持 RGBA，因此它们会忽略对 .test-1 和 .test-2 的声明，直接显示为 p 应用的 RGB 颜色样式，参见图 7.5.12

在 CSS 中，HSL 语法为（具体参见图 7.5.14）：

property: hsl(hue, saturation, lightness);

HSLA 的格式为：

property: hsla(hue, saturation,
→ lightness, alpha transparency);

看一下更新后的样式表（其中省略了部

分相同的代码），如图 7.5.15 所示。这张样式表使用 HSL 和 HSLA 定义了与前例中使用 RGB 和 RGBA 定义的同样的绿色。

```

...
<body>

<p class="test-1">This background is 25%
→ transparent.</p>

<p class="test-2">This background is 60%
→ transparent.</p>

<p>This has the same background color as
→ the other two, but doesn't have an alpha
→ transparency setting.</p>

</body>
</html>

```

图 7.5.10 第一个段落有 test-1 的类名，因此根据样式表，它有 25% 透明的背景色。第二个段落有 test-2 的类名，因此有 60% 透明的背景色。最后一个段落没有任何类名，因此它的背景色是纯色的，由图 7.5.9 中对所有 p 元素定义的样式（第二条规则）所指定

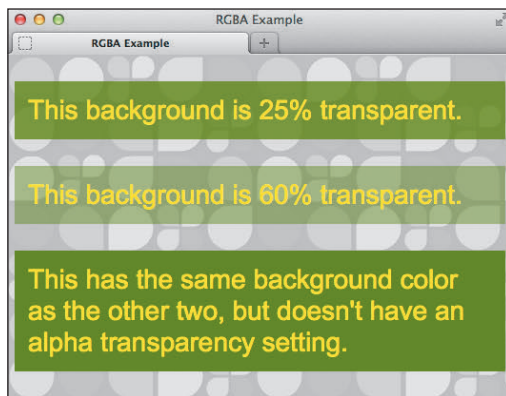


图 7.5.11 我们可以看到页面背景图透过前两个段落的背景，但没有透过最后一个段落的背景。这三个段落的背景色都是一样的，但由于 alpha 透明度不同，它们看起来就像是深浅不同的三种绿色（参见图 7.5.9 中 body 规则的 CSS 注释，了解如何实现这里的文本样式）

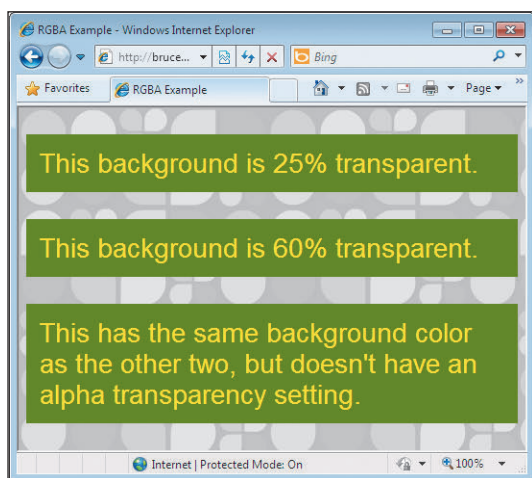


图 7.5.12 IE 直到 IE9 才支持 alpha 透明度 (RGBA 或 HSLA)。这是 IE8 中的截图。IE8 忽略了样式表中的 GRBA 声明，因为它并不理解这些规则。因此所有的段落都是纯色背景（由对 p 元素进行定义的规则指定）。如果没有这条规则，则段落不会有任何背景色（段落中的文字现在看起来已经不正确了，这是因为现在没有透明效果）

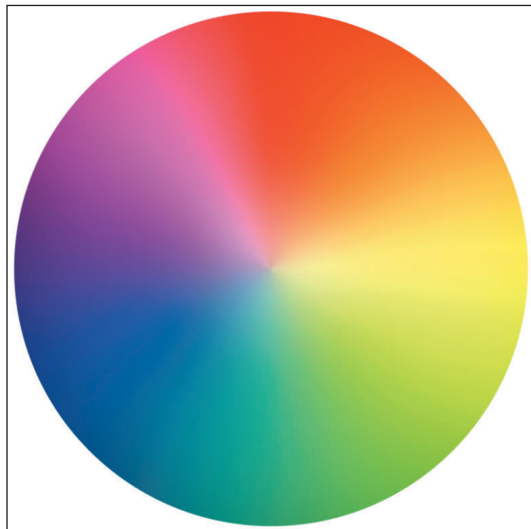


图 7.5.13 HSL 色相值沿顺时针改变。最右边的点为 90，最底部的点为 180，最左边的点为 270，0 和 360 在顶端重合。关于标准颜色及色相值参见补充材料“如何构想 HSL”

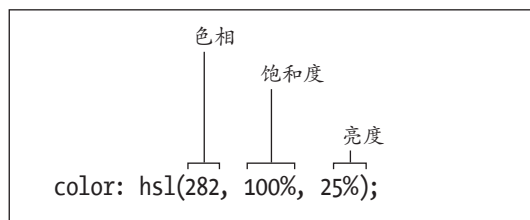


图 7.5.14 对 HSL 格式的分解

```
body {
    ... 与前例相同的样式 ...
}

/* 所有段落都会显示纯色背景，除非指定下列类别之一。IE9之前的版本不支持透明色，只显示十六进制颜色 */
p {
    background-color: #3c8f00; /* for IE
    → before IE9 */
    background-color: hsl(95,100%,28%);
    ...
}

/* 25% 透明度 */
.test-1 {
    background-color:
    → hsla(95,100%,28%,.75);
}

/* 60% 透明度 */
.test-2 {
    background-color: hsla(95,100%,28%,.4);
}
```

图 7.5.15 同样的样式表，但使用 HSL 和 HSLA 表示。此外，我们还针对旧浏览器为段落定义了备选的背景色。p 中声明的顺序是有要求的。旧版本的 IE 使用第一行，因为它能理解这一行，而现代浏览器对这两行都理解，它会应用第二行的样式，因为它出现在后面

如果你仔细看看这些定义 p 元素样式的规则，你会发现一些有趣的事情。这里使用十六进制数定义了 background-color，紧接着又使用 HSL 格式重新定义了一遍。这样做的原因是 Internet Explorer 的所有版本都理解十六进制值，但 IE9 之前的版本无法理解

HSL 或 HSLA。这些旧的浏览器会忽略 HSL 和 HSLA 样式，就像忽略 RGBA 一样，而其他的浏览器则会应用这些定义。这样做的结果就是新的样式表在现代浏览器中保持了一致的效果（如图 7.5.11 所示），而 IE8 也没有变化（如图 7.5.12 所示）。

并非所有的图像编辑器都可以在对话框中指定 HSL（可以为 Photoshop 安装一个插件）。不过，通过 Brandon Mathis 强大的免费在线工具 HSL Color Picker (hslpicker.com) 可以选取颜色，获取其 HSL 值、十六进制数值和 RGB 值。还可以输入这些格式的颜色值，查看颜色的变化。通过在线搜索还可以找到更多其他的颜色工具。

提示 前面已经提到，IE 直到 IE9 才能理解 RGBA、HSL 和 HSLA。可以为这些旧浏览器单独定义备选纯色（参见图 7.5.9），也可以在同一条规则中提供备选的纯色（参见图 7.5.15）。

提示 如果确实想在旧版本的 IE 中使用 alpha 透明度，也可以找到一些方法。一种方法是使用一个 1 像素 × 1 像素的半透明 PNG 作为重复的背景（参见 10.10 节），在使用 RGBA 或 HSLA 的属性之前使用该 PNG 定义备选背景（类似于图 7.5.15）。可以在 rgbapng.com 网站上创建所需的图像，只是需要注意不要在文件名中使用 .part 的后缀（这是它默认添加的）。另一种方法是使用 IE 专有的 Gradient 滤镜。它的语法很令人崩溃，不过好在有专门创建这些代码的工具，如 Michael Bester 的 RGBA & HSLA CSS Generator for Internet Explorer (<http://kimili.com/journal/rgba-hsla-css-generator-for-internet-explorer>)。注意，过多地使用滤镜可能会减慢页面在 IE 中的加载速度。

如何构想 HSL

学习 HSL 的逻辑需要花些时间，但是一旦找到感觉，你会发现它比其他的格式更容易使用。Brandon Mathis 在其 HSL Color Picker 网站的“Why?”（为何使用 HSL）部分对这种格式有一段精彩的解释。他写道：

“选择一个 0 到 360 之间的色相，并将饱和度设为 100，亮度设为 50，就会得到这种颜色最纯的形式。降低饱和度，颜色就会向灰色变化。增加亮度，颜色就会向白色变化；减少亮度，颜色就会向黑色变化。”

例如，沿着圆环移动的过程中可以得到下面这些比较重要的颜色：

- ❑ 红色为 `hsl(0,100%,50%)`;
- ❑ 黄色为 `hsl(60,100%,50%)`;
- ❑ 绿色为 `hsl(120,100%,50%)`;
- ❑ 青色为 `hsl(180,100%,50%)`;
- ❑ 蓝色为 `hsl(240,100%,50%)`;
- ❑ 紫红色为 `hsl(300,100%,50%)`;

本章内容

- ❑ 创建外部样式表
- ❑ 链接到外部样式表
- ❑ 创建嵌入式样式表
- ❑ 应用内联样式
- ❑ 样式表的层叠和顺序
- ❑ 使用与媒体相关的样式表
- ❑ 借鉴其他人的灵感：CSS

在开始定义样式表之前，要知道如何创建和使用包含这些样式表的文件。在本章中，你将学习如何创建样式表文件，如何将 CSS 应用到多个网页（包括整个网站）、单个页面或单独的 HTML 元素。我们会通过三种方法实现这些应用：外部样式表（首选方法）、嵌入样式表和内联样式（最不可取的方法）。

在随后的几章，我们将学习如何创建样式表中的内容。

8.1 创建外部样式表

外部样式表非常适合给网站上的大多数页面或者所有页面设置一致的外观。可以在一个外部样式表中定义全部样式，然后让网站上的每个页面加载这个外部样式表，从而确保每个页面都有相同的设置。尽管后面你将学到嵌入样式表和内联样式表，但从外部样式表为页面添加样式才是最佳实践，推荐

使用这种方法（允许偶尔的例外）。

创建外部样式表的步骤

(1) 在你选择的文本编辑器（参见第一条提示）中创建一个新文档。大多数人使用同一个编辑器创建 HTML 和 CSS 文档。

(2) 为网页定义样式规则，同时，根据需要在 CSS 中添加注释（参见 7.1 节和 7.2 节），如图 8.1.1 所示。

(3) 将文档以纯文本格式保存在希望放置的目录中。尽管任何名称都是允许的，但应该为该文件添加 .css 的扩展名，表明这是一个层叠样式表，如图 8.1.2 所示。

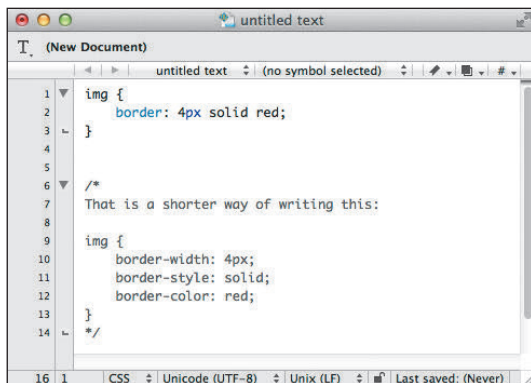


图 8.1.1 使用任何你喜欢的文本编辑器创建样式表。这是 Mac 上的 TextWrangler。我在文档底部包含了一条 CSS 注释，以表明 border 属性的工作原理，这跟 8.5 节中的内容相关。这里绝对不是建议你为样式表中的每一条规则添加注释

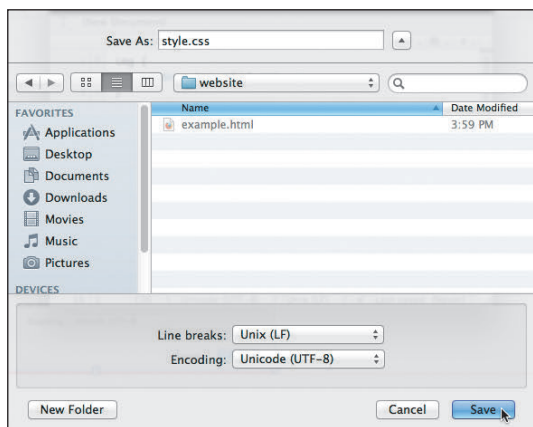


图 8.1.2 一定要将 CSS 文件保存为以 .css 为扩展名、用 UTF-8 编码的文件。有的编辑器需要用户指定将文件保存为纯文本格式（编辑器可能称其为 Text Document（文本文档）、Plain Text（纯文本）、ASCII 等），但 TextWrangler 并不需要。Notepad 会要求这样做，更多信息参见 2.3 节。注意这里将 style.css 与将会用到它的 HTML 页面（example.html）放在同一个目录。这样做可以让这个例子变得相当简单，但在实践中，最好将样式表文件放在单独的目录，从而让网站具有良好的组织结构

在下一节，你将学到如何将样式表加载到 HTML 页面里，从而让样式规则应用到页面上。

提示 关于可用于编写 CSS 的文本编辑器的信息，参见 2.2 节。

提示 如果你现在对 CSS 本身还不太理解，不必担心。在接下来的几章，你将学会如何创建这样的样式规则。不过通常都可以猜出效果，例如，本节例子中规则的意思是“为页面上所有 img 元素添加 4 像素宽的红色实线边框”。

提示 可以以任何名称为样式表文件命名。因此，如果网站只有一个样式表文件，通常会命名为 style.css 或 styles.css。更大的网站则通常会拥有多个样式表文件。在那种情况下，base.css、global.css 和 main.css 是常见的样式表名称，它们通常包含应用于网站大多数页面的样式规则。网站制作者通常创建一些为某些区块所特有的附加 CSS 文件（它们都会被加载到相关页面），作为对基本样式的补充。例如，对于一个商业网站，products.css 包含的可能是为产品相关页面准备的样式规则。无论选择什么文件名，一定不要包含空格。

提示 外部样式表要么是通过链接引用的（参见 8.2 节），要么是导入的（通过 @import），不过不推荐导入。@import 指令会影响页面的下载速度和呈现速度，在 Internet Explorer 中影响更为明显。Web 性能专家 Steve Souders 对此问题进行了讨论，参见 www.stevesouders.com/blog/2009/04/09/dont-use-import/。如果读者对 @import 好奇，请查阅 <http://reference.sitepoint.com/css/at-import> 了解基础知识。

提示 CSS 注释（参见图 8.1.1）既不会影响页面的呈现，也不会显示在网页上，但如果访问者查看源代码，就能看到它们（参见本章后面的 8.7 节）。

8.2 链接到外部样式表

创建了样式表（参见图 8.2.1）之后，需要将它加载到 HTML 页面中去，从而为内容应用这些样式规则。要做到这一点，最好的方式是链接到样式表（参见图 8.2.2）。

```
img {
    border: 4px solid red;
}
```

图 8.2.1 这是本章前面创建的外部样式表 style.css，简洁起见，省略了 CSS 注释

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>El Palau de la Música</title>
    <link rel="stylesheet"
        → href="style.css" />
</head>
<body>
<article>
    <h1>El Palau de la Música</h1>

    <p>I love the <span lang="es">Palau de la
        → Música</span>. It is ornate and gaudy
        → and everything that was wonderful
        → about modernism. It's also the home
        → of the <span lang="es">Orfeó Català
        → </span>, where I learned the benefits
        → of Moscatell.</p>
</article>
</body>
</html>
```

图 8.2.2 link 元素位于 HTML 文档的 head 部分。页面可以包含一个以上的 link 元素，但使用它的次数最好尽可能地少，让页面得以更快地加载

链接到外部样式表的步骤

- (1) 在每个希望使用样式表的 HTML 页面的 head 部分，输入 `<link rel="stylesheet">`。
- (2) 输入一个空格，然后输入 `href="url.css"`，其中 `url.css` 是 CSS 样式表的位置和名称。
- (3) 输入一个空格和 `/>`。（也可以不输入空格，直接输入 `>`。这两种形式都是 HTML5 所允许的，它们的效果也完全一样。）

提示 对外部样式表进行修改时，所有引用它的页面也会自动更新（参见图 8.2.3 和图 8.2.4）。这是外部样式表的神奇力量！



图 8.2.3 将这个样式规则（4 像素宽的红色实线边框）应用于每个 img 元素

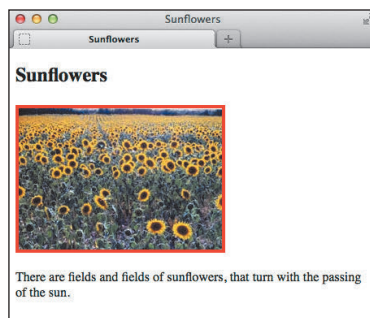


图 8.2.4 其他文档也可以链接到同一个样式表，并拥有同样的样式

提示 外部样式表的另一个好处是，一旦浏览器在某个页面加载了它，在随后浏览引用它的页面时，通常无需再向 Web 服务器请求该文件。浏览器会将它保存到缓存里，也就是保存到用户的计算机里，并使用这个版本的文件。这样做可以加快对页面的加载。不过，不必担心。如果随后对样式表作了修改，再将它传到 Web 服务器，浏览器就会下载更新后的文件，而不是使用缓存的文件（从技术上讲也有例外，但通常不会遇到这种情况）。

提示 出于简化的目的，这个例子中的链接假定 HTML 页面与 style.css 位于同一个目录（参见图 8.2.2）。不过，实践中最好将样式表组织在子文件夹里，而不是与 HTML 页面混在一起。常见的样式表文件夹名称包括 css、styles 等，你也可以按照自己的意愿对其进行命名，只要你认为该名称放在链接的 href 值里是合适的。例如，如果 style.css 位于名为 css 的文件夹里，而 HTML 位于该文件夹的上一级（参见图 8.2.5），那么 link 元素就应该写作图 8.2.6 中那样。

Name	Size	Kind
css	--	Folder
img	--	Folder
example.html	647 bytes	HTML Document

Name	Size	Kind
css	--	Folder
style.css	171 bytes	CSS
img	--	Folder
palau.jpg	23 KB	JPEG image
sunflowers.jpg	16 KB	JPEG image
tickets.jpg	4 KB	JPEG image
example.html	647 bytes	HTML Document

图 8.2.5 这些截图显示了组织网站文件的常见方式。第一个图是文件组织结构的鸟瞰图，接下来是将所有文件夹展开后的视图

```
...
<head>
  <meta charset="UTF-8" />
  <title>El Palau de la Música</title>
  <link rel="stylesheet"
    → href="css/style.css" />
</head>
...
```

图 8.2.6 由于 style.css 位于名为 css 的子文件夹中，因此 example.html 中指向样式表的路径应做相应的修改

提示 外部样式表里的 URL 是相对于服务器上该样式表文件的位置的，而不是相对于 HTML 页面的位置的。你将在 10.10 节学习 CSS 背景图像时看到有关的实例。

提示 可以在页面中使用多个 link 元素，从而加载多个样式表文件。如果在不同的文件中有显示声明产生冲突，则后面文件中的规则的优先级更高。

提示 外部样式表中的规则可能被 HTML 文档内的样式覆盖。8.5 节中总结了以不同方式应用样式的相对影响力。

提示 可以通过设置 media 属性将样式表限制在特定类型的输出上，详细信息参见 8.6 节。

提示 HTML 早期版本要求在 link 元素定义中包含 type="text/css"，但 HTML5 不要这样做，因此可以像代码示例那样忽略它。

8.3 创建嵌入样式表

嵌入样式表是在页面中应用 CSS 的第二种方式。我们在 HTML 页面的 head 部分创建一个 style 元素，其中包含了我们的样式表，如图 8.3.1 所示。由于这些样式只在一个网页里存在，因此不会像外部样式表中的规则那样应用到其他的页面，同时，缓存的好处也不存在了。如上文所述，对于大多数情况，我们推荐使用外部样式表，但理解其他的选择以备不时之需也是很重要的。

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>El Palau de la Música</title>
  <style>
    img {
      border: 4px solid red;
    }
  </style>
</head>
<body>
<article>
  <h1>El Palau de la Música</h1>

  <p>I love the <span lang="es">Palau de
    → la Música</span>. It is ornate and
    → gaudy and everything that was
    → wonderful about modernism. It's also
    → the home of the <span lang="es">Orfeó
    → Català </span>, where I learned the
    → benefits of Moscatell.</p>
</article>
</body>
</html>

```

图 8.3.1 使用嵌入样式表时，style 元素及其包围的样式规则通常位于文档的 head 部分。浏览器对页面的呈现方式与使用外部样式表时是一样的，参见图 8.3.2

创建嵌入样式表的步骤

- (1) 在 HTML 文档的 head 部分输入 <style>。
- (2) 根据需要，定义任意数量的样式规则，有必要的话添加适量 CSS 注释（参见 7.1 节和 7.2 节）。
- (3) 输入 </style> 结束嵌入样式表（参见图 8.3.1）。



图 8.3.2 结果与链接到外部样式表完全一样。区别在于，其他网页无法利用这个页面中定义的样式

提示 当且仅当 style 元素出现在 link 元素后面的时候，嵌入样式表中的样式才会覆盖外部样式表中的样式。详细信息参见 8.5 节。

提示 嵌入样式表是为页面添加 CSS 的次选方式。（也有例外的情况，如特定条件下拥有极大流量的网站。）

提示 HTML5 之前的版本要求在 style 开始标签中包含属性 type="text/css"，不过 HTML5 没有要求。因此你可以省略，本书的示例代码中就都没有写。

8.4 应用内联样式

内联样式是在 HTML 中应用 CSS 的第三种方式。不过，应当最后考虑这种方式，因为它将内容（HTML）和表现（CSS）混在了一起，严重地违背了最佳实践，如图 8.4.1 所示。内联样式只影响一个元素，如图 8.4.2 所示，因此使用它将失去外部样式表的关键好处：一次编写，到处可见。设想要对大量 HTML 做简单的文字颜色的改变，就需要对这些页

面逐一进行检查和修改，这也是大家不会经常使用内联样式的原因。

不过，如果你想快速地测试某种样式，以便随后将它从 HTML 中搬到更易于长期维护的外部样式表中（假定你对结果满意），内联样式就能派上用场了。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>El Palau de la Música</title>
</head>
<body>
...
  

  
...
</body>
</html>
```

图 8.4.1 内联样式规则只影响单个元素（在本例中为第一个 img 元素）

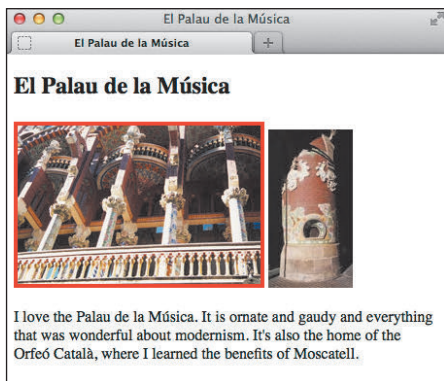


图 8.4.2 只有第一个图像拥有边框。如果要让其他元素也有这种效果，就要在每个 img 元素中单独加上 style="border: 4px solid red"。如你所见，内联样式的效率很低，对全站的内联样式统一进行更新是一件很头疼的事。更好的做法请参见图 8.4.3 和图 8.4.4

```
...
<body>
...
  

  
...
</body>
</html>
```

图 8.4.3 这是实现图 8.4.2 所示效果的一种较好的方法。为第一个 img 添加一个类名，而不是添加一个内联样式。然后，在外部样式表中为该创建一条规则（如图 8.4.4 所示）。如果你希望其他的元素拥有这种样式，只需要为其添加 class="frame" 即可。用任何类名都行，frame 只是一个例子

```
.frame {
  border: 4px solid red;
}
```

图 8.4.4 通过使用类选择器，这条规则可以为任何在 HTML 中拥有 class="frame" 的元素（不仅是 img）添加红色的边框。将它包含在外部样式表中，这样就可以很方便地在全站应用这种样式。下一章将介绍选择器的相关知识

应用内联样式的步骤

(1) 在希望进行格式化的 HTML 元素的开始标签中，输入 style="（或者直接在 img 等虚元素的标签内输入，虚元素不需要结束标签）。

(2) 创建一个样式规则，但不要包含花括号和选择器。不需要选择器是因为直接将样式放入目标元素中了。如果你的样式规则包含多条声明，使用分号将不同的声明隔开。

(3) 输入后引号 "。

提示 注意，不要混淆等号与冒号。由于它们都表示赋值，因而很容易不小心用错。

提示 不要忘记用直引号包围样式定义，以及用分号分隔多条属性定义。

提示 内联样式的优先级高于其他所有样式，除非其他地方与之冲突的样式标记了 `!important`（参见 8.5 节）。

提示 如果要在内联样式声明中指定字体（参见第 10 章），对于有多个单词的字体名称，要用单引号包围，以避免与 `style` 元素的双引号发生冲突。不能在这两个地方使用相同类型的引号。

提示 或许内联样式最为常见的使用场景是在 JavaScript 函数中为元素应用内联样式，从而为页面某个部分添加动态行为。当你关注页面来源的时候可能会注意到这些生成的内联样式，如 Firebug 或 Chrome 的开发者工具。在大多数情况下，应用这些样式的 JavaScript 同 HTML 是分离的，因而仍然保持了内容（HTML）、表现（CSS）和行为（JavaScript）分离的原则。

8.5 样式的层叠和顺序

将多个样式规则应用于同一元素的情况并不少见，大型网站更是如此，他们需要更多精力维护 CSS。有时候这些规则针对的是同一条属性。正如 7.4 节中提到的，在规则的特殊性相同的情况下，样式的顺序就成为关键。基本规则是：在其他条件相同的情况下，越晚出现的样式优先级越高（参见图 8.5.1 至图 8.5.5）。以下几条解释了顺序如何决定样式规则的优先级。

- 嵌入样式表（位于 `style` 元素内）与任何链接的外部样式表之间的关系取决于

它们在 HTML 中的相对位置。在两者发生冲突时，如果 `link` 元素在 HTML 代码中出现得早（图 8.5.2），`style` 元素就会覆盖链接的样式表（图 8.5.3）；如果 `link` 元素出现得晚（图 8.5.4），其中的样式及其包含的任何导入样式表就会覆盖 `style` 元素的规则（图 8.5.5）。

- 内联样式（实际上直接应用于元素）在外部样式表和嵌入样式表之后。由于顺序最靠后，其优先级是最高的。一旦应用到任何地方，都会覆盖与之冲突的其他样式。
- 关于相互冲突的样式的顺序对优先级的影响，有一种例外情况，就是标记 `!important` 的样式总是具有最高的优先级，无论它出现在最前、最后，还是中间。例如，`p { margin-top: 1em !important; }`。不过，要尽量避免这种用法，极端情况除外。几乎在全局情况下，使用选择器都能达到同样的效果。此外，`!important` 让声明变得太强，如果要覆盖这样的样式，就不得不借助于更长的规则。

```
img {
    border: 4px solid red;
}

/*
上面是对以下编写方法的简化：

img {
    border-width: 4px;
    border-style: solid;
    border-color: red;
}
*/
```

图 8.5.1 这是我们熟悉的外部样式表的例子 `style.css`，是从前面搬过来的

```

...
<head>
  <title>El Palau de la Música</title>

  <link rel="stylesheet" href="style.
css" />

  <style>
    img {
      border-style: dashed;
    }
  </style>
</head>
...

```

图 8.5.2 在这个例子中，嵌入样式表出现在最后，因此，它的样式的优先级高于 style.css 里的样式（如果冲突的样式规则具有相同的继承性和特殊性的话）。嵌入样式表会覆盖 style.css 中的边框样式，将实线改为虚线，但不会影响宽度和颜色（参见图 8.5.3）



图 8.5.3 style 元素中定义的虚线边框优先于链接的 style.css 中定义的实线边框，不过保留了 style.css 中指定的边框宽度和颜色

明白了吗？不过，由于你只会使用一个外部样式表，且会避免使用 !important（是吧），因此好的一面是，你不必担心我在上面讲解的这些规则。不过，如果你将来看到其他人的代码并不是像你这样编写的，了解上述知识就很有用了。

对你来说，顺序规则影响最大的情形是：如果网页包括一个以上的 link 元素，不同的

外部样式表中的规则如有冲突，后出现的样式表中的规则优先。

```

...
<head>
  <title>El Palau de la Música</title>
  <style>
    img {
      border-style: dashed;
    }
  </style>
  <link rel="stylesheet"
    href="style.css" />
</head>
...

```

图 8.5.4 这里，链接的样式表最后出现，其优先级高于 style 元素中的规则（其他情况都相同）



图 8.5.5 style.css 样式表中定义的实线边框优先于内部 style 元素中定义的虚线边框

提示 建议不要使用 @import，这主要是出于性能上的考虑。如果用了 @import，当引入的样式后面有与之冲突的样式，则后出现的样式会覆盖这些引入的样式，与预期一致。

提示 有一种使用 !important 的合理情形。有时网页会包含一些你无法修改的 HTML，例如来自第三方服务的新闻源。如果这些 HTML 中有一些内联样式与你的设计不符，你就可以在自己的样式表中使用 !important 覆盖这些样式。

8.6 使用与媒体相关的样式表

可以指定一个只用于特定输出的样式表，如只用于打印，或只用于在浏览器中查看屏幕。例如，可以创建一个具有打印和屏幕版本共有特征的通用样式表，再创建单独的打印样式表和屏幕样式表，分别包含只用于打印的属性和只用于屏幕显示的属性。

指定与媒体相关的样式表的步骤

(1) 在 `link` 或 `style` 元素的开始标签中添加 `media="output"`，其中 `output` 可以是 `print`、`screen` 或 `all`（尽管还有其他一些选项，但这些都是最常见的），参见图 8.6.1。多个值之间用逗号分隔。

(2) 也可以在样式表中使用 `@media` 规则，参见图 8.6.2。这种方法不需要在 `link` 元素中指定媒体类型。

提示 有 9 种可能的输出类型：`all`、`aural`、`braille`、`handheld`、`print`、`projection`、`screen`、`tty` 和 `tv`。浏览器对它们的支持程度各不相同（大多只有少量的支持）。实际上，用得上的只有 `screen` 和 `print`（或许还应加上 `all`），浏览器对它们的支持情况都很好。另一方面（可以这么说），设备对 `handheld` 的支持程度并不高，因此在为移动设备设计时，通常使用 `screen` 而不是 `handheld`（参见第 12 章）。`projection` 类型是为投影仪和其他类似的设备准备的，Opera 的投影模式 Opera Show 支持这种类型。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>El Palau de la Música</title>
  <link rel="stylesheet" href="style.css"
    → media="screen" />
  <link rel="stylesheet" href="print.css"
    → media="print" />
</head>
<body>
<article>
  <h1>El Palau de la Música</h1>

  <p>I love the <span lang="es">Palau de
    → la Música</span>. It is ornate and
    → gaudy and everything that was
    → wonderful about modernism. It's also
    → the home of the <span lang="es">Orfeó
    → Català</span>, where I learned the
    → benefits of Moscatell.</p>
</article>
</body>
</html>
```

图 8.6.1 通过对 `link` 元素添加 `media` 属性，可以将样式表限于特定输出。在这个例子中，使用浏览器查看页面时，`style.css` 会起作用（由于使用了 `media="screen"`）；打印页面时，`print.css` 会起作用（由于使用了 `media="print"`）。如果第一个 `link` 元素包含了 `media="all"` 的设置，或者没有 `media` 属性，`style.css` 中的规则也会应用到打印页面

```

/* 针对所有媒体的样式 */
img {
    border: 4px solid red;
}

p {
    color: orange;
    font-style: italic;
}

/* 打印样式表 */
@media print {

    body {
        /* 加大字号 */
        font-size: 25pt;
    }

    p {
        /* 黑色的十六进制值 */
        color: #000;
    }

    img {
        /* 不显示图像 */
        display: none;
    }
}

```

图 8.6.2 样式表中的 `@media` 规则是指定其他媒体类型的另一种方式（参见第 12 章）。在这个例子中，上面是影响所有媒体类型（包括打印）的样式，下面是专门用于打印的样式，样式规则放在 `@media print { }` 中。在浏览器中查看这个页面，效果跟本章其他图一样，只是文字是橙色和倾斜的。这个页面的打印版如图 8.6.3 所示

提示 联合使用 CSS3 引入的媒体查询和这里介绍的媒体输出类型可以形成强大的威力。通过这些技术，可以根据输出设备的属性确定需要应用到页面的样式。例如，可以让页面在较窄的屏幕（如智能手机屏幕）上呈现为一种样式，在较宽的屏幕（如笔记本屏幕）上呈现为另一种样式。现在，设备的种类越来越多，媒体查询已经变成代码编写人员的一种重要工具。第 12 章将向你介绍如何构建可以在各种设备上正常运行的网站。关于媒体查询的内容是从 12.4 节开始介绍的。

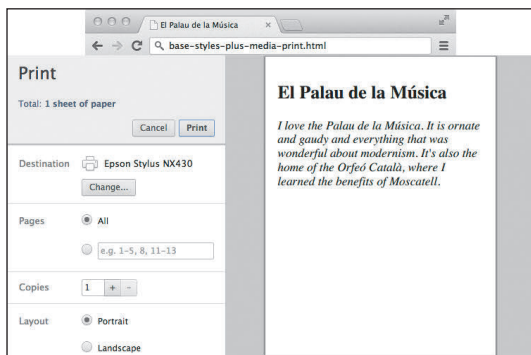


图 8.6.3 Chrome 浏览器的打印预览功能可以显示图 8.6.1 中页面打印出来的样子。文本变大了，不会显示图像，段落文本是加粗的斜体，而不是橙色的。`font-style: italic` 这条声明也应用到了打印模式，因为打印样式表没有指定其他的 `font-style`

8.7 借鉴他人的灵感

我们在第 2 章中学习了如何查看网页的源代码。查看他人的 CSS 也不难。

查看其他设计人员的 CSS 代码

(1) 首先查看页面的 HTML 代码（如图 8.7.1 所示）。关于查看 HTML 源代码的更多信息，参见 2.8 节。

如果 CSS 代码是嵌入样式表，就已经能看到了。

(2) 如果 CSS 是外部样式表，就在 HTML 中找到对它的引用，再点击文件名（参见图 8.7.1）。样式表就显示在浏览器窗口中了（如图 8.7.2 所示）。如果愿意，可以复制这些代码，再粘贴到文本编辑器中。



图 8.7.1 查看包含你感兴趣的样式表的 HTML 页面源代码，再点击样式表的文件名

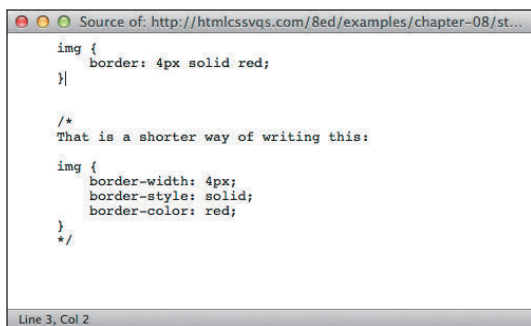


图 8.7.2 样式表显示在浏览器窗口中，img 规则下面包含了 CSS 注释（因此确保注释整洁）

提示 同 HTML 一样，可以从其他设计人员的代码中借鉴灵感，然后编写自己的样式表。不过，查看其他人的代码时也要留心。这些代码放在万维网上并不意味着它们就是编写某一效果的代码的最佳方式（尽管其作者希望如此）。

提示 现代浏览器允许像图中显示的那样直接在 HTML 代码中点击样式表的名称。要在旧的浏览器中查看样式表，可能需要复制 link 元素中显示的 URL，再粘贴到浏览器的地址栏（替换 HTML 文件名），然后按回车键。如果样式表的 URL 是相对地址（参见 1.7 节），就需要将网页的 URL 和样式表的相对 URL 结合在一起，形成样式表的完整 URL。

提示 使用现代浏览器提供的开发者工具，可以快速查看页面的 CSS。大多数浏览器都捆绑了这类工具。Firefox 中有一个实现此功能的扩展，名为 Firebug（参见第 20 章）。



本章内容

- ❑ 构造选择器
- ❑ 按名称选择元素
- ❑ 按类或 ID 选择元素
- ❑ 按上下文选择元素
- ❑ 选择第一个或者最后一个子元素
- ❑ 选择元素的第一个字母或者第一行
- ❑ 按状态选择链接元素
- ❑ 按属性选择元素
- ❑ 指定元素组
- ❑ 组合使用选择器

正如 7.1 节中看到的，CSS 样式规则有两个主要部分。选择器决定将格式化应用到哪些元素，而声明则定义要应用的格式化。在本章中，我们将学习如何定义 CSS 选择器。

最简单的选择器可以对给定类型的所有元素（如所有的 h2 标题）进行样式化，有的选择器允许我们根据元素的类、上下文、状态等来应用格式化规则。随着你对本章内容的学习，你会了解更多类型的选择器。请记住，我们可以在一个选择器中组合使用多个选择器，例如，编写类选择器和属性选择器的复合选择器。

定义选择器以后，可以根据第 10 章 ~ 第 16 章的讲解创建声明（包括实际的属性和值）。在此之前，我们将在例子中使用非常简单且意义明确的 {color: red;}。

9.1 构造选择器

选择器决定样式规则应用于哪些元素。例如，如果要对所有的 p 元素添加 Georgia 字体、12 像素高的格式，就需要创建一个只识别 p 元素而不影响代码中其他元素的选择器。如果要对每个区域中的第一个 p 设置特殊的缩进格式，就需要创建一个稍微复杂一些的选择器，它只识别作为页面中每个区域的第一个元素的 p 元素。

选择器可以定义五个不同的标准来选择要进行格式化的元素。

- ❑ 元素的类型或名称，如图 9.1.1 所示。
- ❑ 元素所在的上下文，如图 9.1.2 所示。
- ❑ 元素的类或 ID，如图 9.1.3 和图 9.1.4 所示。
- ❑ 元素的伪类或伪元素，如图 9.1.5 所示（放心，我将在后面解释伪类和伪元素）。
- ❑ 元素是否有某些属性和值，如图 9.1.6 所示。

为了指出目标元素，选择器可以使用这五个标准的任意组合。在大多数情况下，只使用一个或两个标准即可。另外，如果要对几组不同的元素应用相同的样式规则，可以将相同的声明同时应用于几个选择器（参见 9.9 节）。

本章其余部分将详细解释如何定义选择器。

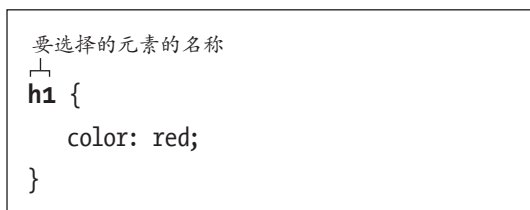


图 9.1.1 最简单的选择器类型就是要格式化的元素的类型的名称（在这个例子中为 h1）

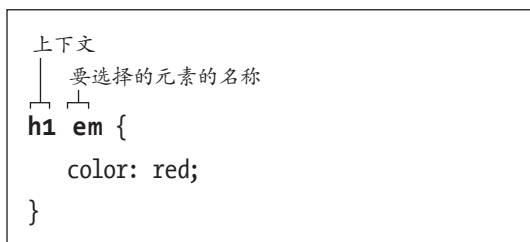


图 9.1.2 这个选择器使用上下文。这个样式只应用于 h1 元素中的 em 元素。其他地方的 em 元素不受影响

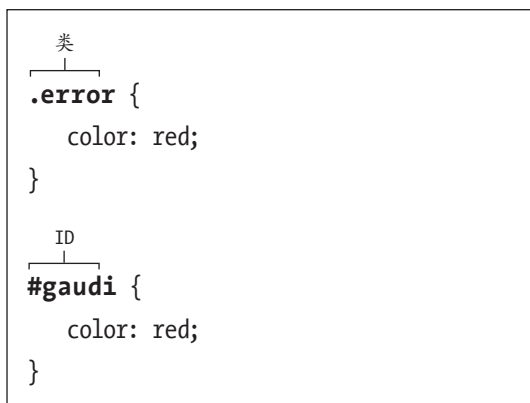


图 9.1.3 第一个选择器选择所有属于 error 类的元素。换句话说，就是所有在开始标签中包含 class="error" 的元素。第二个选择器选择 id 为 gaudi 的那个元素，也就是在开始标签中指定 id="gaudi" 的元素。回忆一下，一个 id 在每个页面中只能出现一次，而一个 class 可以出现任意次数。这是我们推荐 class 选择器而不推荐 ID 选择器的主要原因，你可以针对尽可能多的元素重用 class 选择器的样式

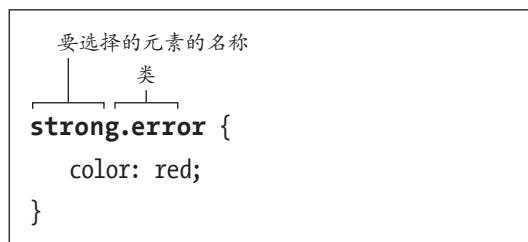


图 9.1.4 通过在 class 或 id 选择器前面添加目标元素的名称，可以增强选择器的特殊性。在这个例子中，第一个选择器仅选择属于 error 类的 strong 元素，而不是属于 error 类的任何元素。除非确有必要，一般不要使用这种方法。图 9.1.3 中的选择器是更好的方法，因为它比较灵活

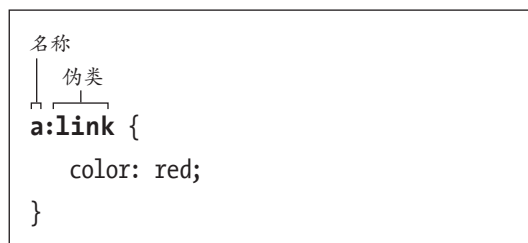


图 9.1.5 在这个例子中，选择器选择属于 link 伪类的 a 元素（即页面上还未访问的链接）

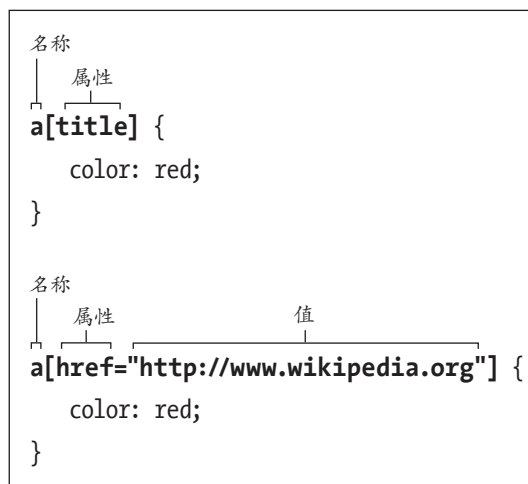


图 9.1.6 可以在选择器中使用方括号添加关于目标元素的属性或值的信息，第一个例子针对的是所有具有 title 属性的 a 元素，第二个例子针对的只是指向维基百科页面的 a 元素

提示 我们建议你尽量不要使用 ID 选择器，我会在 9.3 节详细解释原因。

提示 我们会在 9.8 节了解到，选择器不仅限于针对一个属性的确切值，图 9.1.6 就是这种情况。

提示 编写 CSS 的一个重要目标就是让选择器尽可能的简单，仅保持必要的特殊性。应该充分利用样式的层叠特性，即元素的后代会继承祖先元素的样式。同时，应该将页面中的通用设计元素挑选出来，编写一个选择器（如类名），从而可以在全站不同的元素上共享该样式。这样，样式表通常会变得更小，也更容易维护。随着你构建页面经验的积累，你会越来越深入地体会到这些技巧，不过现在先提出来，有助于日后的理解。

9.2 按名称选择元素

选择要格式化的元素（参见图 9.2.1），最常用的标准可能是元素的名称（也就是大家平常说的类型选择器，因为你指定了应用样式的元素的类型）。例如，可能要让所有的 h1 元素以大字号、粗体显示，让所有的 p 元素以无衬线字体显示。

按照类型选择要格式化的元素

(1) 输入 *selector*，其中 *selector* 是目标元素的类型名称（不含任何属性），如图 9.2.2 所示。

(2) 输入 {。

(3) 输入你希望应用到选中元素的样式，这要用属性：值对的形式表示。我们从下一章才开始讲解 CSS 属性和值。

(4) 输入 } 结束样式规则。

```
<!DOCTYPE html>
<html lang="en">
<head>
...
</head>
<body>
...
<article class="architect">
  <h1>Antoni Gaudí</h1>
  <p>Many tourists are drawn to Barcelona to see Antoni Gaudí's incredible architecture.</p>

  <p>Barcelona <a href="http://www.gaudi2002.bcn.es/english/" rel="external">celebrated the 150th
  → anniversary</a> of Gaudí's birth in 2002.</p>

  <h2 lang="es">La Casa Milà</h2>
  <p>Gaudí's work was essentially useful. <span lang="es">La Casa Milà</span> is an apartment
  → building and <em>real people</em> live there.</p>

  <h2 lang="es">La Sagrada Família</h2>
  <p>The complicatedly named and curiously unfinished Expiatory Temple of the Sacred Family is
  → the <em>most visited</em> building in Barcelona.</p>
</article>
...
```

图 9.2.1 这段 HTML 代码包含两个 h2 元素（或许你对 lang 属性感到疑惑。它指明内容使用的是页面默认语言以外的另一种语言。页面默认语言是在每页开头处紧跟在 DOCTYPE 后面的 html 元素中指定的。在这个例子中，每个 h2 上的 lang="es" 表示其内容为西班牙语）

注意，在本章后面的章节中，再谈到构造选择器的步骤时我不会重述第 (2) ~ (4) 步的内容了，不过对于创建完整的样式规则，这些步骤都是必需的。更多信息请参见 7.1 节。

```
h2 {
    color: red;
}
```

图 9.2.2 这个选择器会选择文档中所有的 h2 元素，并让它们显示为红色，参见图 9.2.3

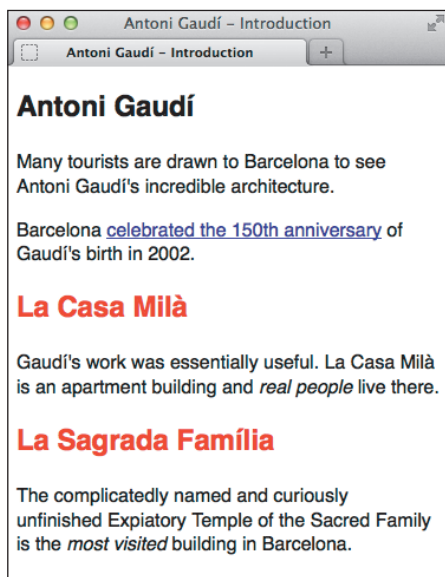


图 9.2.3 所有的 h2 元素都显示为红色

提示 除非指定其他情况（使用本章其余部分介绍的技术），指定类型的所有元素都被格式化，无论它们出现在文档的什么位置。

提示 并非所有的选择器都需要指定元素的名称。如果对某一类的元素进行格式化，而不管属于这个类的元素的类型，就可以从选择器中去掉元素名称。下一节将解释具体怎么做。

提示 通配符 *（星号）匹配代码中的任何元素名称。例如，使用 `* { border: 2px solid green; }` 会让每个元素都有一个 2 像素宽的绿色实线边框！因为匹配范围太广，会让浏览器加载页面变慢，因此应该谨慎使用通配符。实际适合使用通配符的情况比较少。

提示 可以在一个选择器中使用一组元素名称，名称之间用逗号分隔。更多细节参见 9.9 节。

9.3 按类或 ID 选择元素

如果已经在元素中标识了 class（参见图 9.3.1）或 id（参见最后一条提示），就可以在选择器中使用这个标准，从而只对已标识的元素进行格式化（参见图 9.3.2）。不过推荐使用类选择器，一会儿我会解释理由。

1. 按类选择要格式化的元素

(1) 输入 .（点号）。

(2) 不加空格，直接输入 *classname*，这里的 *classname* 标识希望应用样式的类。

2. 按 id 选择要格式化的元素

(1) 输入 #（井号）。

(2) 不加空格，直接输入 *id*，这里的 *id* 唯一标识希望应用样式的元素。

提示 可以单独使用 class 和 id，也可以同其他选择器标准混在一起使用。例如，`.news { color: red; }` 会影响所有属于 news 类的元素，而 `h1.news { color: red; }` 只会影响属于 news 类的 h1 元素。除非必须特别针对目标元素，最好不要在 id 或 class 选择器中添加元素名称。

```

...
<article id="gaudi" class="architect">
  <h1>Antoni Gaudí</h1>
  <p>Many tourists are drawn to Barcelona to see Antoni Gaudí's incredible architecture.</p>

  <p>Barcelona <a href="http://www.gaudi2002.bcn.es/english/" rel="external">celebrated the 150th
  → anniversary</a> of Gaudí's birth in 2002.</p>

  <h2 lang="es">La Casa Milà</h2>
  <p>Gaudí's work was essentially useful. <span lang="es">La Casa Milà</span> is an apartment
  → building and <em>real people</em> live there.</p>
  ...
</article>

<p>This paragraph doesn't have <code>class="architect"</code>, so it isn't red when the CSS is
→ applied.</p>

<article class="architect">
  <h1>Lluís Domènech i Montaner</h1>
  <p>Lluís Domènech i Montaner was a contemporary of Gaudí.</p>
  ...
</article>
...

```

图 9.3.1 有两个 class 为 architect 的 article 元素。它们之间还有一个不含 class 的短段落

```

.architect {
  color: red;
}

```

图 9.3.2 这个选择器会选择 class 为 architect 的元素。在这个例子中，它们都是 article 元素，不过，可以将类应用于任何元素。如果只想对这个类的 article 元素应用这种样式，可以将选择器写为 article.architect。不过，这样做通常会使特殊性高于你所需要的程度

提示 注意，在图 9.3.1 和图 9.3.2 中使用了表达内容含义的 class 名称 (architect)，而不是命名为 red。尽管也有极端情况，但最好避免创建描述事物如何显示的 class 名称（一般称为描述性类名），因为你可能在未来改变样式（例如，在本节示例中让文本显示为绿色）。

提示 如果要定位的元素有多个类名，可以在选择器中将它们连在一起，就像 .architect.bio { color: blue; } 这样。任何 .architect 或 .bio 选择器的规则仍会应用于该元素，但 .architect.bio 的规则的特殊性更高，因此如果有样式冲突，.architect.bio 的规则的优先级更高。注意类名之间没有空格。如果有空格，就会针对任何 architect 类的元素嵌套的 bio 类元素设置样式（参见 9.4 节）。

提示 如果图 9.3.2 中写的是 #gaudi { color: red; }，就只有第一个 article 的文本会显示为红色，因为它是唯一包含 id="gaudi" 的元素。每个 id 都必须是唯一的，因此，不能在关于 Lluís Domènech i Montaner 的 article 上再用这个 id。

提示 关于在 HTML 代码中为元素指定类的信息，参见 3.14 节。



图 9.3.3 属于 architect 类的 article 显示为红色，而两个 article 元素之间的 p 元素没有显示为红色（或许你对链接显示为蓝色感到疑惑。链接显示为蓝色是浏览器的默认样式，可以编写自己的样式覆盖它）

类选择器与 ID 选择器的比较

要在 class 选择器和 id 选择器之间作出选择的时候，建议尽可能地使用 class 选

择器。这主要是因为我们可以复用 class 选择器。有人提议完全不使用 id 选择器，我赞同这种观点，并且在自己的工作中始终遵循这一点，但最终的决定权掌握在网站开发人员手中。id 选择器会引入下面两个问题。

- 与它们关联的样式不能在其他元素上复用（记住，在一个页面中，一个 id 只能出现在一个元素上）。这会导致在其他元素上重复样式，而不是通过 class 共享样式。
- 它们的特殊性比 class 选择器要强得多。这意味着如果要覆盖使用 id 选择器定义的样式，就要编写特殊性更强的 CSS 规则。如果数量不多，可能还不难管理。如果处理规模较大的网站，其 CSS 就会变得比实际所需的更长、更复杂。

随着你处理的 CSS 越来越多，你对这两点的理解也会更加清晰。（另一方面，有人喜欢使用 id 选择器的原因之一就是使用它们一眼就能看出元素是唯一的。不过，依照个人经验，这点好处还不足一均衡它带来的弊端。另外，万一你哪天改变了网站设计，原先唯一的元素不再唯一，id 选择器就不合适了。）

因此，推荐的做法是寻找能将共享样式结合进一个或多个 class 的机会，从而可以对它们进行复用，同时，如果确实要使用 id 选择器，也应该尽量少用。这样，你的样式表会比较短，更易于管理。

需要说明的是，id 选择器在 HTML 中仍具有重要的作用。通过它们可以在页面中定义锚（参见 6.2 节），在编写 JavaScript 为特定的页面元素应用特殊行为时它们尤其具有重要的价值（JavaScript 是一个单独的主题，本书不会作过多介绍）。

9.4 按上下文选择元素

在 CSS 中，可以根据元素的祖先、父元素或同胞元素来定位它们（参见 1.3 节中的“父元素和子元素”），如图 9.4.1 ~ 图 9.4.4 所示。

```
...
<article class="architect">
  <h1>Antoni Gaudí</h1>
  <p>Many tourists ... </p>
  <p>Barcelona ... </p>

  <section>
    <h2 lang="es">La Casa Milà</h2>
    <p>Gaudí's work ... </p>
  </section>

  <section>
    <h2 lang="es">La Sagrada Família</h2>
    ...
  </section>
</article>
...
```

图 9.4.1 这里对 `article` 的一部分使用了 `section` 元素，从而可以演示几代元素的关系。同时，对文字内容进行了压缩，这样可以更容易看清元素之间的关系。注意，在这个代码片段中，有两个第二代 `p` 元素，它们由类名为 `architect` 的 `article` 直接包含，另外还有一个第三代 `p` 元素，位于第一个 `section`（包含在 `article` 里）。在完整的代码中还有一个第三代 `p` 元素，不过在这里没有显示出来。这里，所有 `h2` 的实例也是第三代元素

```
.architect p {
  color: red;
}
```

图 9.4.2 这里组合使用了类选择器和类型选择器。`.architect` 和 `p` 之间的空格表示这个选择器会寻找任何作为 `architect` 类元素后代（无论是第几代）的 `p` 元素。实际效果参见图 9.4.4，此外，图 9.4.3 提供了几种实现同样效果的其他方式

```
/* 得到相同效果的其他方式
----- */
/* 是任意article祖先的所有p元素，
   这是三个中特殊性最低的一个 */
article p {
  color: red;
}

/* 属于architect类article元素的祖先的任意p元
   素，是三个中特殊性最高的一个 */
article.architect p {
  color: red;
}
```

图 9.4.3 第一个例子中的选择器（`article p { }`）比图 9.4.2 中的选择器和它后面的选择器（`article.architect p { }`）的特殊性都要低。这里演示的第二个例子比其他两个例子的特殊性都要高，不过在实践中，在类名（尤其是 ID 名）之前添加元素名通常会让特殊性比实际需要的要高

祖先（ancestor）是包含目标元素（后代，descendant）的任何元素，不管它们之间隔了多少代。（父元素是直接包含另一个元素的元素，它们之间只隔一代，被包含的元素称为子元素。）通常对元素的子元素增加缩进，从而可以清晰地看到它们之间的关系（如图 9.4.2 所示）。缩进对页面的显示没有任何影响。

要实现某种效果，通常有多种构建选择器的方式可供选择。具体取决于你需要多大的特殊性（如图 9.4.3 所示）。

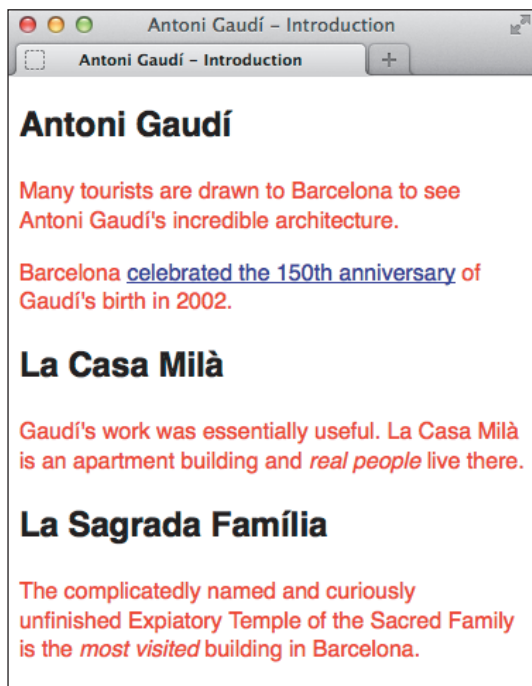


图 9.4.4 所有包含在属于 `architect` 类的元素里的 `p` 元素都是红色的，即使它们同时也包含在其他属于 `architect` 类的元素。图 9.4.2 和图 9.4.3 中每条样式规则的结果都显示在这里

1. 按祖先元素选择要格式化的元素

(1) 输入 `ancestor`，这里的 `ancestor` 是希望格式化的元素的祖先元素的选择器。

(2) 输入一个空格（必不可少）。

(3) 如果需要，对后续每个祖先元素重复第 (1) 步和第 (2) 步。

(4) 输入 `descendant`，这里的 `descendant` 是希望格式化的元素的选择器。

提示 我们通常将基于元素祖先的选择器称为后代选择器，不过 CSS3 将其重新命名为后代结合符。（有些人仍然使用“选择器”这一名称。）

提示 不要对图 9.4.3 中的 `article.architect` 部分感到疑惑。记住这只代表“`class` 等于 `architect` 的 `article`”。因此 `article.architect p` 表示“包含在 `class` 等于 `architect` 的 `article` 元素里的任何 `p` 元素”。相比之下，特殊性低一些的 `.architect p` 表示“包含在 `class` 等于 `architect` 的任意元素里的所有 `p` 元素”，参见图 9.4.3。

2. 按父元素选择要格式化的元素

上面的例子展示了后代结合符。CSS 也有子结合符，从而可以为父元素的直接后代（即子元素）定义样式规则。在 CSS3 之前，它们称为子选择器。

(1) 输入 `parent`，这里的 `parent` 是直接包含待格式化元素的元素的选择器。

(2) 输入 `>`（大于号），参见图 9.4.5。

(3) 如果需要，对后续每代父元素重复第 (1) 步和第 (2) 步。

(4) 输入 `child`，这里的 `child` 是要格式化的元素的选择器。

```
.architect > p {
  color: red;
}
```

图 9.4.5 这个选择器仅选择 `architect` 类元素的子元素（而非子子元素、子子子元素等）的 `p` 元素。包含于任何其他元素的 `p` 元素均不会被选中，实现参见图 9.4.6

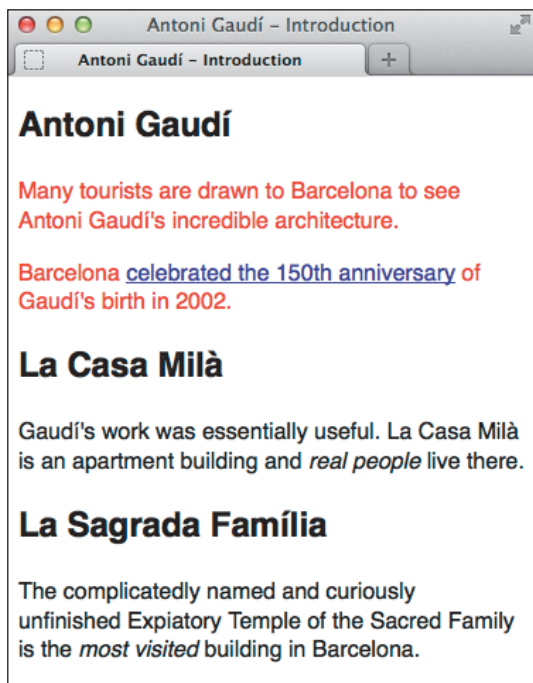


图 9.4.6 只有头两个 p 元素是 architect 类元素的子元素。另两个 p 元素是 article 里 section 元素的子元素。这个例子使用的 HTML 代码见图 9.4.1

提示 正如你在后代结合符中看到的，可以忽略 class 前面的元素名称。例如，在上面的例子中使用 `article.architect > p { color: red; }` 会产生同样的效果，但这个选择器的特殊性更高。要想比以上两种方法的特殊性更低（图 9.4.5 就是个例子），应该忽略整个类别，就像图 9.4.5 中的 `article > p { color: red; }`。本章剩余部分的一些例子也可以按类似的方式进行简化。你已经了解了如何进行简化，因此不必再一一列举这些替代方案，但应记住，通常最好保持较低的特殊性，让样式更易于复用。

提示 也可以在子结合符中使用 id 选择器，但推荐尽量使用特殊性更低的选择器（如类选择器）。

提示 学习本章内容同时可参考 1.3 节中的“父元素和子元素”。

3. 按相邻同胞元素选择要格式化的元素

下面继续讲我们的“家族”主题。同胞（sibling）元素是拥有同一父元素的任何类型的子元素。相邻同胞元素（adjacent sibling）是直接相互毗邻的元素，即它们之间没有其他的同胞元素。在下面这个简略的例子中，h1 和 p 是相邻同胞元素，p 和 h2 是相邻同胞元素，而 h1 和 h2 则不是相邻同胞元素。不过，它们都是同胞元素（也是 body 元素的子元素）。

```
...
<body>
    <h1>...</h1>
    <p>...</p>
    <h2>...</h2>
</body>
</html>
```

CSS 相邻同胞结合符（adjacent sibling combinator）可以选择直接跟在指定的同胞元素后面的同胞元素，如图 9.4.7 和图 9.4.8 所示。关于 CSS3 中新出现的普通同胞结合符（general sibling combinator），参见最后一条提示。

(1) 输入 *sibling*，这里的 *sibling* 是包含在同一父元素中的、直接出现在目标元素前面的元素的选择器。（它们不必是同一种元素类型，只要它们彼此直接相邻就行。）

(2) 输入 +（加号）。

(3) 如有需要, 对每个后续的同胞元素重复第 (1) 步和第 (2) 步。

(4) 输入 `element`, 这里的 `element` 是要格式化的元素的选择器。

```
.architect p+p {
  color: red;
}
```

图 9.4.7 相邻同胞结合符只选择直接跟在同胞 `p` 元素之后的元素

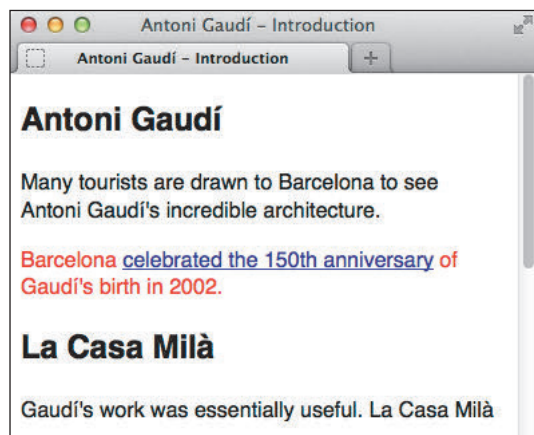


图 9.4.8 只有直接跟在同胞 `p` 元素后面的 `p` 元素显示为红色。如果后面还有第三个、第四个以及更多的段落, 它们也将显示为红色。例如, 如果要对除第一个段落以外的所有段落进行缩进, 相邻同胞结合符就很有用

提示 我们可能会用到普通同胞结合符, 通过它可以选那些并非直接出现在另一同胞元素后面的同胞元素。它与相邻同胞结合符的唯一区别是使用 `~` (波浪号) 代替 `+` 分隔同胞元素。例如, `h1 ~ h2 { color: red; }` 会让任何属于同一父元素的同胞 `h1` 后面的 `h2` 元素显示为红色 (它们可以直接相邻, 也可以不直接相邻)。

9.5 选择第一个或最后一个子元素

上一节解释了如何选择作为另一个元素的子元素的元素, 举的例子是 `.architect > p`, 选择所有作为 `architect` 类元素的子元素的段落。不过, 有时需要选择仅作为某元素第一个或最后一个子元素的元素。这时, 就要用到 `:first-child` 和 `:last-child` 伪类 (前者参见图 9.5.1 ~ 图 9.5.3, 后者参见图 9.5.1、图 9.5.4 和图 9.5.5)。

```
...
<p>A partial list of Gaudí's projects
→ follows:</p>

<ul>
  <li lang="es">La Casa Milà</li>
  <li lang="es">La Sagrada Família</li>
  <li>College of the Teresians <span lang=
→ "es">(Colegio Teresiano)</span></li>
  <li>Park Güell</li>
</ul>
...
```

图 9.5.1 我们将在第 15 章讲解列表, 不过在这里你只需要知道, 这是一个无序列表 (`ul`), 每个列表项目 (`li`) 都是其子元素。这是使用 `:first-child` 和 `:last-child` 规定样式的常见情形, 不过通常是用于添加一个边框, 而不是简单地改变一下文字颜色

```
li:first-child {
  color: red;
}
```

图 9.5.2 这个选择器会选择作为父元素的第一个子元素的 `li` 元素

A partial list of Gaudí's projects follows:

- La Casa Milà
- La Sagrada Família
- College of the Teresians (Colegio Teresiano)
- Park Güell

图 9.5.3 因为第一个 li 元素是 ul 元素的第一个子元素，所以显示为红色。缩进和项目符号是浏览器针对无序列表的默认样式

```
li:last-child {
    color: red;
}
```

图 9.5.4 这个选择器只选择作为父元素的最后一个子元素的 li 元素

A partial list of Gaudí's projects follows:

- La Casa Milà
- La Sagrada Família
- College of the Teresians (Colegio Teresiano)
- Park Güell

图 9.5.5 这种情况下只有最后一个 li 元素显示为红色

有的人在一开始没有真正弄清楚这些伪类的工作原理。他们以为 `li:first-child` 这样的选择器会选择 li 元素的第一个子元素，而 `li:last-child` 会选择 li 元素的最后一个子元素。如果是这样的话，“(Colegio Teresiano)”在两个例子中都是红色的（参见图 9.5.3 和图 9.5.5），因为它们所在的 span 既是某个 li 的第一个元素，也是其最后一个元素（参见图 9.5.1）。实际情况是，这些伪类选择的是作为第一个子元素或最后一个子元素的元素（在这个例子中就是 li）。

选择某元素的第一个或最后一个子元素进行格式化

(1) 这一步可选，输入代表我们想应用样式的第一个或者最后一个子元素（如 p 或 .news），参见最后一条提示。接下来千万不要有空格。

(2) 如果选择的是第一个子元素就输入 `:first-child`（参见图 9.5.2），如果选择的是最后一个子元素，就输入 `:last-child`（参见图 9.5.4）。

提示 在下一节的补充材料中我们会更详细地探讨伪类。

提示 `:last-child` 伪类被添加至 CSS 的时间要晚于 `:first-child`。因此，IE9 之前的 Internet Explorer 并不支持 `:last-child`。好在通常你可以绕过这一限制，因为 IE8 及之后的版本都支持 `:first-child`。这里介绍一种常见的场景（其中提到的外边距在第 11 章才会讲到）。假设你希望每个 li 的下边都有 20 像素的外边距，但最后一个 li 除外。你可以设置 `li { margin-bottom: 0; margin-top: 20px; }`，让每个列表项目都拥有 20 像素的上外边距，没有下外边距。然后使用 `li:first-child { margin-top: 0; }` 取消第一个元素的上外边距。这相当于反过来编写代码，但可以达到完全一样的效果。

提示 在伪类之前添加其他选择器可以使其特殊性更高。例如 `.architect h1:first-child { color: red; }` 仅对作为 architect 类元素的第一个子元素的 h1 应用样式。

提示 尽管第(1)步中指定选择符这一做法是很常见的,但这并非必需。例如,仅仅使用 `:first-child { color: red; }`,可以为任何作为另一个元素的第一个子元素的元素应用样式。在图 9.5.1 所示的例子中,这样做会让段落(`body`的第一个子元素)、第一个 `li`(`ul`的第一个子元素)和“(Colegio Teresiano)”(它所在的 `span` 是一个 `li` 的第一个元素)变成红色。

9.6 选择元素的第一个字母或者第一行

我们可以分别使用 `:first-letter` 和 `:first-line` 伪元素只选择元素的第一个字母(参见图 9.6.1、图 9.6.2 和图 9.6.3)或第一行(参见图 9.6.1、图 9.6.4 和图 9.6.5)。

```
...
<article class="architect">
  <h1>Antoni Gaudí</h1>
  <p>Many tourists are drawn to Barcelona
  → to see Antoni Gaudí's incredible
  → architecture.</p>
  <p>Barcelona <a href="http://
  → www.gaudi2002.bcn.es/english/"
  → rel="external">celebrated the 150th
  → anniversary</a> of Gaudí's birth in
  → 2002.</p>

  <h2 lang="es">La Casa Milà</h2>
  <p>Gaudí's work was essentially
  → useful...</p>
...
</article>
...
```

图 9.6.1 很容易判断 `:first-letter` 定位的第一个字母,但 `:first-line` 会影响哪些单词只有在浏览器里查看页面,看到内容的排布情况时才能知道(如图 9.6.5 所示)。HTML 本身无法定义单词位于哪一行

```
p:first-letter {
  color: red;
  font-size: 1.4em; /* make letter
  → larger */
  font-weight: bold;
}
```

图 9.6.2 下面的选择器只选择每个 `p` 元素的第一个字母

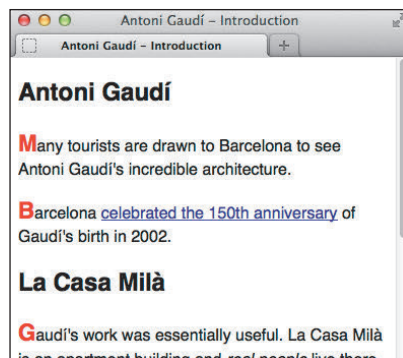


图 9.6.3 `first-letter` 选择器可用来实现每段首字母大写的效果

```
p:first-line {
  color: red;
}
```

图 9.6.4 下面的选择器只选择每个 `p` 元素的第一行

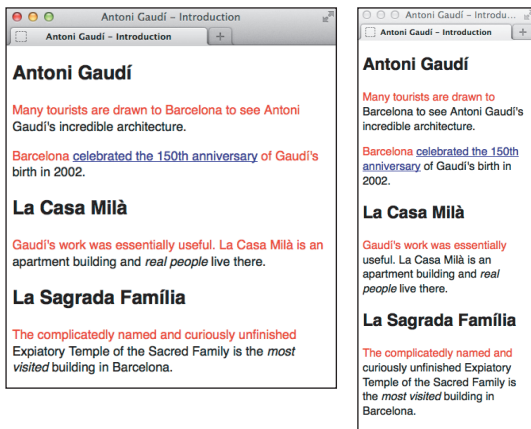


图 9.6.5 调整浏览器窗口会改变每段第一行的具体内容,但选择器选中的始终是第一行

1. 选择元素的第一个字母

(1) 输入 `element`，这里的 `element` 是要对其第一个字母进行格式化的元素的选择器。

(2) 输入 `:first-letter` 以选择第 (1) 步中引用的元素的第一个字母。

2. 选择元素的第一行

(1) 输入 `element`，这里的 `element` 是要对其第一行进行格式化的元素的选择器。

(2) 输入 `:first-line` 以选择第 (1) 步中引用的元素的第一行。

提示 可以将 `:first-letter` 和 `:first-line` 伪元素与比这个例子中的选择器更复杂的选择器结合使用。例如，如果要选择包含在 `project` 类元素中的每个段落的第一个字母，可以使用 `.project p:first-letter`。

提示 只有某些特定的 CSS 属性可以应用于 `:first-letter` 伪元素，包括 `font`、`color`、`background`、`text-decoration`、`vertical-align`（只要 `:first-letter` 不是浮动的）、`text-transform`、`line-height`、`margin`、`padding`、`border`、`float` 和 `clear`。我们将在第 10 章和第 11 章讲到这些属性。

提示 第一个字母前面的标点符号（如引号）会被当做第一个字母的一部分，一同被格式化。现代浏览器都支持这一特性，但 IE8 之前的版本并不是这样做的，它将标点符号本身当做第一个字母。

伪元素、伪类及 CSS3 语法

在 CSS3 中，`:first-line` 的语法为 `::first-line`，`:first-letter` 的语法为 `::first-letter`。注意，它们用两个冒号代替了单个冒号。这样修改的目的是将伪元素（有四个，包括 `::first-line`、`::first-letter`、`::before` 和 `::after`）与伪类（如 `:first-child`、`:link`、`:hover` 等）区分开。

伪元素（pseudo-element）是 HTML 中并不存在的元素。例如，定义第一个字母或第一行文字时，并未在 HTML 中作相应的标记。它们是另一个元素（在本例中为 `p` 元素）的部分内容。

相反，伪类（pseudo-class）则应用于一组 HTML 元素，而你无需在 HTML 代码中用类标记它们。例如，使用 `:first-child` 可以选择某元素的第一个子元素，你就不用写成 `class="first-child"`。更多关于伪类的内容，请参见下一节。

未来，`::first-line` 和 `::first-letter` 这样的双冒号语法是推荐的方式，现代浏览器也支持它们。原始的单冒号语法则被废弃了，但浏览器出于向后兼容的目的，仍然支持它们。不过，IE9 之前的 Internet Explorer 版本均不支持双冒号。因此，你可以选择继续使用单冒号语法，除非你为 IE8 及以下版本设置了单独的 CSS（参见 <http://reference.sitepoint.com/css/conditionalcomments>）。

9.7 按状态选择链接元素

CSS 允许根据链接的当前状态对它们进行格式化，参见图 9.7.1。链接的状态包括访问者是否将鼠标停留在链接上，链接是否被访问过，等等。可以通过一系列伪类实现这一特性。

```
...
<p>Many tourists are drawn to Barcelona
→ to see Antoni Gaudí's incredible
→ architecture.</p>

<p>Barcelona <a href="http://www.
→ gaudi2002.bcn.es/english/">
→ celebrated</a> the 150th anniversary
→ of Gaudí's birth in 2002.</p>
...
```

图 9.7.1 无法在代码中指定链接的状态。链接的状态是由访问者控制的。伪类让你可以获取链接的状态，以改变链接在该状态下显示的效果

按状态选择要格式化的链接元素的步骤

- (1) 输入 a (a 是链接元素的名称)。
- (2) 输入 : (冒号)，前后都没有空格。
- (3) 完成第 (2) 步以后，执行下列操作之一以表明你希望影响的链接状态 (参见图 9.7.2)：
 - ❑ 输入 link 以设置从未被激活或指向，当前也没有被激活或指向的链接的外观 (如图 9.7.3 所示)；
 - ❑ 输入 visited 以设置访问者已激活过的链接的外观 (如图 9.7.4 所示)；
 - ❑ 输入 focus，前提是链接是通过键盘选择并已准备好激活的，如图 9.7.5 所示 (注意：如果链接处于活跃状态也会获得焦点)；
 - ❑ 输入 hover 以设置光标指向链接时链接的外观 (如图 9.7.6 所示)；
 - ❑ 输入 active 以设置激活过的链接的外观 (如图 9.7.7 所示)。

```
a:link {
    color: red;
}

a:visited {
    color: orange;
}

a:focus {
    color: purple;
}

a:hover {
    color: green;
}

a:active {
    color: blue;
}
```

图 9.7.2 应该始终按照这种顺序定义链接的样式，以避免链接处于多种状态 (如已访问和鼠标停留) 时覆盖属性

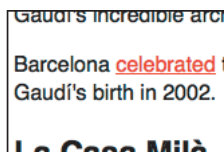


图 9.7.3 新的、未访问的链接显示为红色。另见彩插

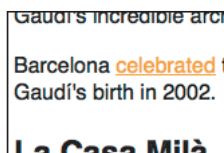


图 9.7.4 访问过的链接变为橙色。另见彩插

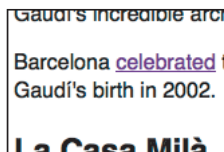


图 9.7.5 链接获得焦点 (如通过 Tab 键) 时显示为紫色。另见彩插

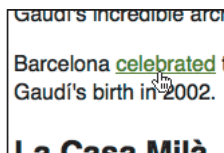


图 9.7.6 当访问者将鼠标指针停留在链接上时，它显示为绿色。另见彩插

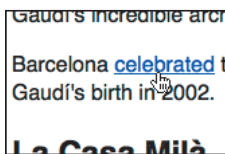


图 9.7.7 当访问者激活链接时，它变为蓝色。另见彩插

提示 要对链接指定样式，不一定要指定伪类（因此第(2)步和第(3)步都是可选的步骤）。例如，使用 `a { color: red; }` 会让链接的所有状态显示为同一种样式。不过，最好使用伪类区分不同状态的样式，这样做可以方便访问者。

提示 也可以对其他类型的元素使用 `:active` 和 `:hover` 伪类。例如，设置 `p:hover { color: red; }` 以后，鼠标停留在任何段落上段落都会显示为红色。

提示 由于链接可能同时处于多种状态（如同时处于激活和鼠标停留状态），且晚出现的规则会覆盖前面出现的规则，所以，一定要按照下面的顺序定义规则：`link`、`visited`、`focus`、`hover`、`active`（缩写为 LVFHA）。一种助记口诀为“Lord Vader’s Former Handle Anakin”（达斯·维达的原名叫安纳金）。有人提议使用 LVHFA 的顺序，这也是可行的。

提示 触屏设备（如智能手机和平板电脑）的浏览器没有桌面浏览器所具有的“鼠标悬停”（即 `hover`）状态。这是因为，像 iPad 这样的设备无法判断手指停留在一个链接上的时刻，只能判断手指轻触该链接以激活它的时刻。不过，在 iPhone 和 iPad 上，访问者激活链接时，确实会显示通过 `:hover` 指定的样式。其他设备的行为则不一而定。

9.8 按属性选择元素

可以对具有给定属性或属性值的元素进

行格式化，参见图 9.8.1。CSS 提供了多种方式匹配属性和属性值，包括只检查属性名，检查全部或者部分属性值（更多信息请参见表 9.8.1）。如果在选择器中忽略属性值（参见图 9.8.2），就可以为具有给定属性的元素应用样式，而不管具体的属性值是什么（参见图 9.8.3）。

```
...
<article class="architect">
  <h1>Antoni Gaudí</h1>

  <p class="intro">Many tourists are
  → drawn to Barcelona to see Antoni
  → Gaudí's incredible architecture.</p>

  <p>Barcelona <a href="http://www.
  → gaudi2002.bcn.es/english/" rel=
  → "external">celebrated the 150th
  → anniversary</a> of Gaudí's birth
  → in 2002.</p>

  <h2 lang="es">La Casa Milà</h2>
  <p class="highlight">Gaudí's work was
  → essentially useful. <span lang="es">La
  → Casa Milà</span> is an apartment
  → building and <em>real people</em> live
  → there.</p>

  <h2 lang="es">La Sagrada Família</h2>
  <p>The complicatedly named and curiously
  → unfinished Expiatory Temple of the
  → Sacred Family is the <em>most visited
  → </em> building in Barcelona.</p>
</article>
...
```

图 9.8.1 出于演示目的，我为两个段落都添加了 `class` 属性

```
p[class] {
  color: red;
}
```

图 9.8.2 方括号包围目标属性和目标属性值。这个例子中没有属性值，它选择的是所有具有 `class` 属性的段落

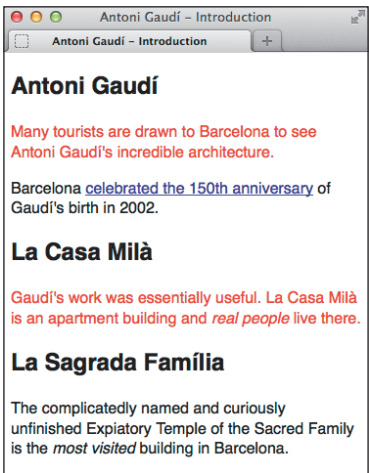


图 9.8.3 每个包含 class 属性的 p 元素（无论其 class 值是什么）都显示为红色。如果选择器是 p[class="intro"], 只有第一个段落会显示为红色。如果选择器是 p[class^="intro"], 则第一个段落以及 class="introduction" 和 class="introductory" 的段落也显示为红色

按属性选择要格式化的元素

- (1) 输入 *element*，这里的 *element* 是要考察其属性的元素的选择器。
- (2) 输入 *attribute*，这里的 *attribute* 是选择元素需要考察的属性的名称。
- (3) 以下步骤可选，根据需要输入：
 - ❑ 输入 `= "value"`，表示属性值等于这里的 *value* 的元素将被选中。
 - ❑ 输入 `~ = "value"`，表示属性值包含这里

的 *value* 的元素将被选中（属性值还可以包含其他内容，不同的属性值之间用空格分隔）。它必须匹配完整的单词，而不是单词的一部分。

- ❑ 输入 `| = "value"`（前面是管道符号，而不是数字 1 或小写字母 l），表示属性值等于这里的 *value* 或以 *value-* 开头的元素将被选中。不要输入连字符，浏览器知道搜索连字符（这常常用以搜索包含 lang 属性的元素，如在 HTML 中，`[lang|= "en"]` 会同时匹配 `lang= "en"` 和 `lang= "en-US"`）。
- ❑ 输入 `^ = "value"`，表明属性值以这里的 *value* 开头（作为完整的单词，或单词的一部分）的元素将被选中（这是 CSS3 中新增的特性，参见本节提示）。
- ❑ 输入 `$ = "value"`，表明属性值以这里的 *value* 结尾（作为完整的单词，或单词的一部分）的元素将被选中（这是 CSS3 中新增的特性，参见本节提示）。
- ❑ 输入 `* = "value"`，表明属性值至少包含这里的 *value* 一次的元素将被选中。也就是说，*value* 不必是属性值中的完整单词（这是 CSS3 中新增的特性，参见本节提示）。
- (4) 输入 `]`。如果你想为元素指定其他属性或者属性值，可重复第 (2) ~ (4) 步。

表 9.8.1 属性选择器参考表

选 择 器	属 性 值
[attribute]	匹配指定属性，不论具体值是什么
[attribute="value"]	完全匹配指定属性值
[attribute~="value"]	属性值是以空格分隔的多个单词，其中有一个完全匹配指定值
[attribute = "value"]	属性值以 value- 打头
[attribute^="value"]	属性值以 value 开头，value 为完整的单词或单词的一部分
[attribute\$="value"]	属性值以 value 结尾，value 为完整的单词或单词的一部分
[attribute*="value"]	属性值为指定值的子字符串

提示 当前所有主流浏览器均支持按元素包含的属性（和属性值）选择元素。对于第(3)步中提到的 CSS3 中新增的属性选择器，IE7 和 IE8 有一些异常。更多信息参见 <http://reference.sitepoint.com/css/css3attributeselectors>。

更多属性选择器示例

属性选择器看起来都长得差不多（参见表 9.8.1），但是它们使用起来非常灵活，可以以多种方式匹配元素的属性及其值。这里给出更多的例子，演示属性选择器几种不同的使用方式。这些例子在实践中也用得上。

- 这个选择器选择任何 `rel` 属性值等于 `external`（必须完全匹配）的 `a` 元素。对任何链接到站外页面的 `a` 元素（即外部链接）添加 `rel="external"` 是非常好的习惯。使用下面的样式规则，可为外部链接添加独特的样式，从而告诉访问者点击该链接将离开你的网站。

```
a[rel="external"] {  
    color: red;  
}
```

- 假设某 `article` 元素有两个类，如 `<article class="project barcelona">`，另一个 `article` 有一个类，如 `<article class="barcelona">`。`~` 语法可以测试单词的部分匹配，即匹配以空格相隔的多个单词中的一个。在这个例子中，两个元素都将显示为红色。

```
article[class~="barcelona"] {  
    color: red;  
}
```

/ 这个选择器也能匹配，因为这个选择器匹配部分字符串（不需要完整的单词） */*

```
article[class*="barc"] {  
    color: red;  
}
```

/ 这个选择器无法匹配，因为barc并不是空格分隔的单词列表中的某个完整的单词 */*

```
article[class~="barc"] {  
    color: red;  
}
```

- 这个选择器选择任何带有 `lang` 属性且属性值以 `es` 开头的 `h2`（如 `es-ES` 和 `es-PE`，前者代表西班牙西班牙语，后者代表秘鲁西班牙语）。在 HTML 代码示例（参见图 9.8.1）中有两个这样的实例。

```
h2[lang="es"] {
    color: red;
}
```

- 通过使用通用选择器，这个选择器选择任何带有 lang 属性且属性值以 es 开头的元素。在 HTML 代码示例（参见图 9.8.1）中有三个这样的实例。

```
*[lang="es"] {
    color: red;
}
```

- 通过联合使用多种方法，这个选择器选择所有既有任意 href 属性，又有任意属性值包含单词 howdy 的 title 属性的 a 元素。

```
a[href][title~="howdy"] {
    color: red;
}
```

- 作为上一选择器的精确度低一些的变体，这个选择器选择所有既有任意 href 属性，又有任意属性值包含 how（作为完整的单词或单词的一部分，它匹配 how、howdy、show 等，无论 how 出现在属性值的什么位置）的 title 属性的 a 元素。

```
a[href][title*="how"] {
    color: red;
}
```

- 这个选择器匹配任何 href 属性值以 http:// 开头的 a 元素。

```
a[href^="http://"] {
    color: orange;
}
```

- 这个选择器匹配任何 src 属性值完全等于 logo.png 的 img 元素。

```
img[src="logo.png"] {
    border: 1px solid green;
}
```

- 这个选择器的精确度比前一个低一些，它匹配任何 src 属性值以 .png 结尾的 img 元素。

```
img[src$=".png"] {
    border: 1px solid green;
}
```

只用上面这些例子很难全面概括使用选择器能够实现的效果，不过，希望它们可以激发你继续探索。

9.9 指定元素组

我们经常需要将相同的样式规则应用于多个元素。可以为每个元素重复地设置样式规则，也可以组合选择器，一次性地设置样式规则（参见图 9.9.1 至图 9.9.3）。当然，后一种方法效率更高，通常也会让样式表更易于维护。

```
...
<article class="architect">
  <h1>Antoni Gaudí</h1>
  <p>Many tourists are drawn ...</p>
  <p>Barcelona ...</p>

  <h2 lang="es">La Casa Milà</h2>
  <p>Gaudí's work was ...</p>

  <h2 lang="es">La Sagrada Família</h2>
  <p>The complicatedly named ...</p>
</article>
...
```

图 9.9.1 这段代码包含一个 h1 和两个 h2

```
h1,
h2 {
  color: red;
}
```

图 9.9.2 可以列出任意数量的单独的选择器（无论它们包含的是元素名称、类还是伪元素），只需逗号分隔它们即可

将样式应用于元素组的步骤

- (1) 输入 *selector1*，这里的 *selector1* 是受样式规则影响的第一个元素的名称。
- (2) 输入，（逗号）。
- (3) 输入 *selector2*，这里的 *selector2* 是受样式规则影响的下一个元素的名称。
- (4) 对其他每个元素重复第 (2) 步和第 (3) 步。

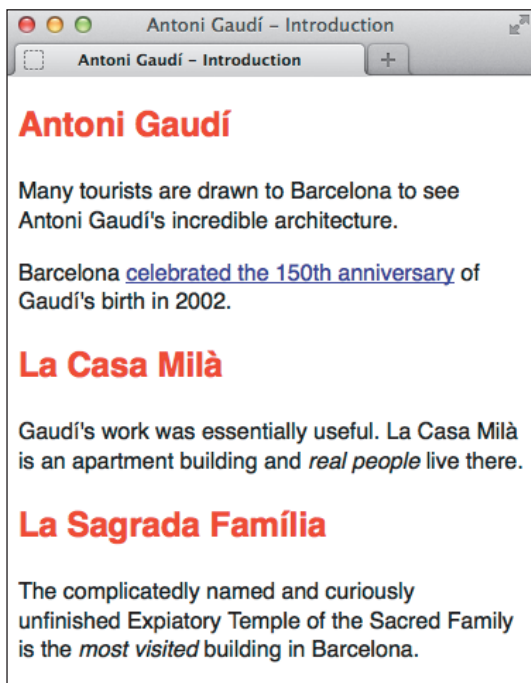


图 9.9.3 通过同一条样式规则，可以让 h1 和 h2 元素都显示为红色

提示 通过元素组添加样式只是一种简写方式。图 9.9.2 中的规则跟 `h1 { color: red; }` 和 `h2 { color: red; }` 这两个规则的效果完全一样。

提示 可以组合使用任何类型的选择器，从最简单的（如图 9.9.2 所示）到最复杂的都可以。例如，可以使用 `h2, .project p:first-letter` 来选择二级标题以及包含在 `class` 等于 `project` 的所有元素中的 `p` 元素的第一个字母。

提示 不同的选择器不一定非得单独成行（图 9.9.2 采用的就是这种做法），但很多编码人员都遵循这种惯例，易于阅读！

提示 有时，为应用于多个选择器的共同样式创建一个样式规则，再为没有共同点的样式分别创建单独的样式规则是很有用的。要记住，在样式表中后指定的规则会覆盖先指定的规则。

9.10 组合使用选择器

本章的示例都尽量保持简单，以便你全面了解各种类型的选择器。不过，现实中常常需要组合使用这些技术，才能找到要格式化的元素，但这也是选择器的功能强大之所在。

图 9.10.1（实现结果见图 9.10.2）通过一个极端的例子展示了如何组合使用选择器，不过此处我们不推荐采用这种方法。这里展示了可实现相同效果的几种方式，它们是按特殊性由低向高排列的。

```
em {
    color: red;
}

.project em {
    color: red;
}

.architect .project em {
    color: red;
}
```

没有必要搞一大堆吓唬人的选择器来实现网页中的大部分设计，不管这些设计在浏览器中看起来多么错综复杂。只在必要时组合使用选择器，并且，最好将特殊性控制在刚好需要的程度。例如，如果你只想选择包含在 `class="project"` 的元素内的 `em`

元素，可以使用 `.project em { color: red; }`。尽管在 HTML 中 `em` 元素是嵌套在 `p` 元素中的，但没有必要写成 `.project p em { color: red; }`，除非你不想为段落以外的 `em` 元素应用该样式。总之，从最简单的开始，按需增加特殊性。

```
.project h2[lang="es"] + p em {
    color: red;
}
```

图 9.10.1 这对你来说可能是个挑战。从右向左看，它表明“仅选择 `em` 元素，它们包含在 `p` 元素中，这样的 `p` 元素是 `lang` 属性值以 `es` 开头的 `h2` 元素的直接相邻同胞元素，且是 `class` 等于 `project` 的任何元素的子元素”。弄明白了吗？实际上，很少需要编写这么复杂的选择器，但至少这样做是可行的

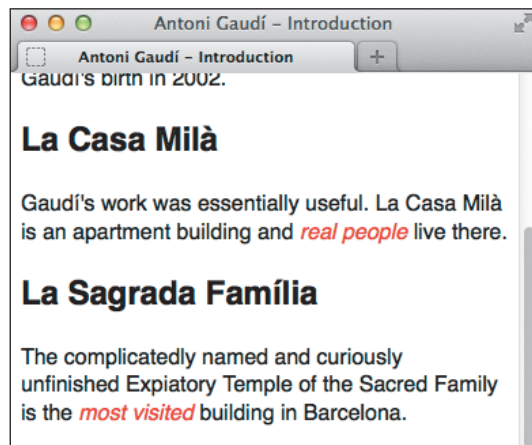


图 9.10.2 图 9.10.1 中所有那些东西只是让 `em` 元素显示为红色吗？如果你认为简单地写成 `.em { color: red; }` 或者 `.architect em { color: red; }` 要好得多（也更易于维护），你肯定是对的。除非需要更高的特殊性，否则越简单越好

更多 CSS3 选择器

CSS3 为你的工具箱增加了不少新的选择器。你已经在本章中见到了其中的一些。其他的新选择器大多为伪类，其中的一些还相当复杂，不过也非常强大。所有 CSS3 选择器及其完整描述见 www.w3.org/TR/css3-selectors/#selectors，简介与示例参见 www.w3.org/wiki/CSS/Selectors。

除了 Internet Explorer 以外，其他浏览器对选择器的支持程度都很好。Internet Explorer 直到 IE9 才开始支持大多数 CSS3 中新增的选择器（尤其是伪类和伪元素）。

在很多情况下，如果你的网站在旧的浏览器里的样式与在现代浏览器里的样式稍有差别也是没有问题的。不过，如果你确实希望 IE8 甚至更旧的浏览器也能使用这些选择器，可以考虑使用 Keith Clark 的 Selectivizr（<http://selectivizr.com>）。根据他的介绍，这是“一个在 Internet Explorer 6 ~ 8 中模拟 CSS3 伪类和属性选择器的 JavaScript 工具”。根据他的提示，Selectivizr 有一些限制。我的建议是尽可能不使用该工具（我无意冒犯该工具本身）。页面上使用的 JavaScript 越多，页面的加载时间就越长（参见第 19 章），何况旧的浏览器（如 IE8）处理 JavaScript 的效率要低得多。

本章内容

- 本章之前与本章之后
- 选择字体系列
- 指定替代字体
- 创建斜体
- 应用粗体格式
- 设置字体大小
- 设置行高
- 同时设置所有字体值
- 设置颜色
- 设置背景
- 控制间距
- 增加缩进
- 对齐文本
- 修改文本的大小写
- 使用小型大写字母
- 装饰文本
- 设置空白属性

使用 CSS 可以修改文本的字体、大小、粗细、倾斜、行高、前景和背景颜色、间距和对齐方式，可以决定是否为文本添加下划线或删除线，可以将文本转化为全部使用大写字母、全部使用小写字母或使用小型大写

字母。而且，通过短短几行代码就可以让这些样式应用于整篇文档或整个网站。在本章中，你将学习如何做到这些。

本章讨论的很多属性只是应用到了文本，10.10 节中讨论的属性也应用到了非文本元素。另外其他章节也涉及一些文本相关主题：第 13 章讲到了 Web 字体，第 14 章讲到了为文本添加阴影。

所有这些 CSS 特性都是设计网页不可或缺的部分。在第 11 章讲解 CSS 布局时我们还会继续关注 CSS 特性。

10.1 本章之前与本章之后

浏览器会给页面添加极少量的默认样式（如图 10.1.1 所示）。在本章中，我们将尝试对文本和背景应用我们自己的 CSS，从而改善页面的外观（如图 10.1.2 所示）。在学习本章的过程中，你可能需要用到图 10.1.3 所示的简化过的 HTML（尤其要注意其中的类名是如何在 CSS 中使用的）。完整的 HTML 及本章的所有示例都可以在本书配套网站上查找，见 www.htmlcssvqs.com/8ed/10。



图 10.1.1 页面的默认样式在所有的浏览器中都是与此类似的（但标题的字号大小在不同的浏览器中可能略有差异）



图 10.1.2 你即将了解到，对页面施加一系列样式可以改变页面的外观。可以通过 www.htmlcssvqs.com/8ed/text-final 访问这个页面

```
...
<body>
<article class="architect">
  <div class="intro">
    <h1>Barcelona's Architect</h1>

    <p class="subhead">Antoni Gaudí's
    → incredible buildings bring...</p>

    <p>Gaudí's...search for simplicity...is
    → quite apparent in his work, from the
    → <a href="#park-guell">Park Guell</a>
    → ... to the Church of the <a href=
    → "#sagrada-familia">Sacred Family
    → </a> ...</p>
  </div>

  <section class="project family">
    <h2 id="sagrada-familia">La Sagrada
    → Família</h2>

    <div class="photos">
      
      ... 6 more images ...
    </div>

    <p>The complicatedly named...</p>
    <p>The Sagrada Família attracts...</p>
  </section>

  <section class="project guell">
    <h2 id="park-guell">Park Guell</h2>

    <div class="photos">
      ... 5 images ...
    </div>

    <p>The Park Guell always reminds me
    → of ... Howard Roark in ... <a href=
    → "http://..."><cite>The Fountainhead
    → </cite></a>...</p>
    <p>...now we <em>all</em> get to enjoy
    → it...</p>
  </section>
</article>
</body>
</html>
```

图 10.1.3 HTML 中包含了很多类名，方便 CSS 找到这些特殊的区域

10.2 选择字体系列

对于自己的网站，一大关键问题就是选择标题和主体文本所用的字体。你将了解到，并非所有的系统都支持相同的默认字体，因此，应该定义替代字体作为备选。不过，让我们首先看看如何定义单个字体（如图 10.2.1 和图 10.2.2 所示），以及未提供替代字体的影响（如图 10.2.3 所示）。

设置字体的方法

在样式表中需要的选择器之后，输入 `font-family: name`，这里的 *name* 是首选字体的名称，参见图 10.2.1。

```
body {
    font-family: Geneva;
}

h1, h2 {
    font-family: "Gill Sans";
}
```

图 10.2.1 由于 `font-family` 是继承属性，而我们对 `body` 元素设置了 Geneva 字体，该样式会应用到其他元素。通过对 `h1` 和 `h2` 元素设置 Gill Sans 字体，覆盖了继承属性

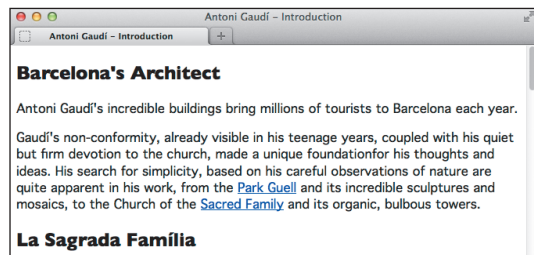


图 10.2.2 在 OS X 系统中，Geneva 和 Gill Sans 很常见，因此可以正常显示。`p` 元素和 `a` 元素继承了 `body` 的 `font-family` 设置，除了标题之外的文本都显示为 Geneva。如果没有为 `h1` 和 `h2` 指定 Gill Sans 字体，它们也将显示为 Geneva 字体

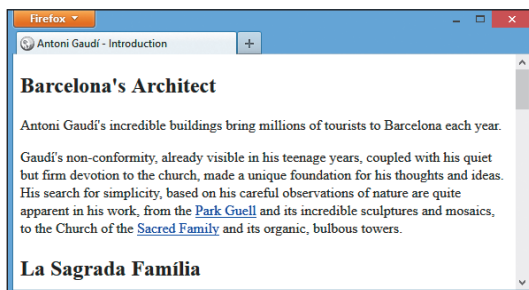


图 10.2.3 一些 Windows 系统中并未安装 Geneva 和 Gill Sans 字体。如果选择了访问者系统中没有的字体，他们的浏览器中就会显示默认字体（如图所示，Windows 中的默认字体是 Times New Roman）

提示 对于包含多个单词的字体名称，应该用引号（单引号或双引号）包围起来。

提示 可以使用小写字母指定字体名称，如 `font-family: geneva;`。

提示 开发人员可以指定自己想要设置的任何字体，不过访问者只会看到他们的系统里安装的字体（一种特殊情况是加载 Web 字体，参见第 13 章）。关于 Windows 和 Mac OS X 共享的标准字体的更多信息参见下一节。

提示 虽然 `font-family` 属性是继承的，但有几个元素不会继承父元素的字体设置，其中有表单的 `select`、`textarea` 和 `input` 元素（参见第 16 章）。不过，可以强制它们继承父元素的字体设置，代码为 `input, select, textarea { font-family: inherit; }`。

提示 使用通用的 `font` 属性可以一次性定义字体、大小和行高。参见 10.8 节。

10.3 指定替代字体

尽管你可以指定自己想要的任何字体，但这些字体不一定在每个系统上都能显示出来（如上一节所示）。为此，可以在 `font-family` 属性中列举一种以上的字体（如图 10.3.1 所示）。浏览器会使用列表中第一个已安装在访问者系统内的字体（如图 10.3.2 所示和图 10.3.3 所示）或 Web 字体（参见第 13 章）。

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

h1,
h2 {
    font-family: "Gill Sans", "Gill Sans MT",
        → Calibri, sans-serif;
}
```

图 10.3.1 第一字体栈告诉浏览器如果系统上没有 Geneva 就用 Tahoma，如果这两个都没有就用标准的 sans-serif。标题的字体栈提供了三种备选字体，替换字体可能无法完全匹配首先字体，不过，我们的目标是指定尽可能接近的字体

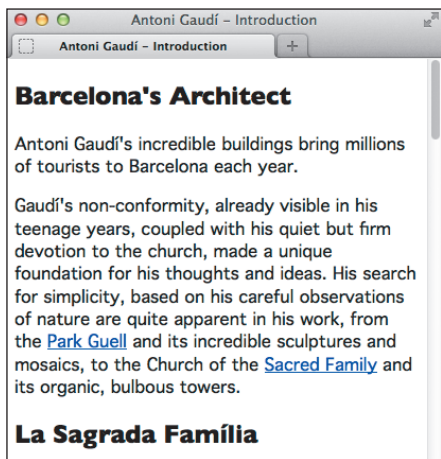


图 10.3.2 安装了 Geneva 和 Gill Sans 的系统（如这里的 OS X）会继续使用该字体。因此，这里显示的字体与图 10.2.2 是完全一样的（截图看起来不一样，这是因为本图里的浏览器宽度更窄）

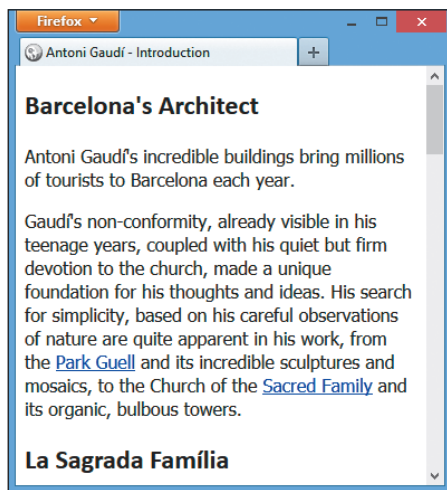


图 10.3.3 没有 Geneva 字体的系统会使用 Tahoma 字体（对于这种情况，大多数 Windows 系统都会使用后一种字体）。如果 Tahoma 也没有，浏览器会使用 sans-serif 字体。标题字体也用同样的方法处理。Windows 机器既没有 Gill Sans，也没有 Gill Sans MT，标题是使用第三种替换字体 Calibri 显示的

字体列表称为字体栈（font stack）。通常，字体栈至少包含三个字体：希望使用的字体、一个或几个替代字体，以及一个表示类属的标准字体，表示“如果其他的字体都不可用，就用这个”（参见第一条提示）。

补充材料“OS X 和 Windows 上默认共有的字体”讨论了两个系统共享的几种字体。使用两种系统上的共享字体或者 Web 字体的情况很常见，因此通常访问者看到的字体都是一样。

指定替代字体

(1) 输入 `font-family: name`，这里的 `name` 是首选字体的名称。

(2) 输入，`name2`，这里的 `name2` 是第二字体的名称。用一个逗号和空格分隔每个字体。

(3) 根据需要，重复第 (2) 步，最后以一个表示类属的标准字体（`serif`、`sans-`

serif、cursive、fantasy 或 monospace^①，它们代表首选字体最为接近的风格）结束字体列表。

OS X 和 Windows 上默认共有的字体

OS X 和 Windows 上默认都有的字体是非常有限的，它们仅包括 Arial、Comic Sans MS、Courier New、Georgia、Impact、Trebuchet MS、Times New Roman 和 Verdana。结果，绝大多数网站使用的字体都无外乎这些。尽管它们在 Mac OS 和 Windows 上的浏览器中显示的效果也并不完全一致，但至少你可以肯定它们都能显示出来。

除此之外，还有别的选择（如图 10.3.1 所示）。OS X 和 Windows 都包含了更多（却不相同的）可用于字体栈的系统字体。在网上搜索“font stacks”（字体栈），就可以看到很多 font-family 声明。可以将这些声明复制到你的样式表里，从而为不同的访问者提供相似的字体。

此外，还可以在网页中加载系统默认没有的字体。Web 设计人员多年来都痴迷于这种想法，近年来终于成为现实。Web 字体变得越来越普遍，第 13 章将讲解如何使用 Web 字体。

在 Windows 上显示 Arial，在 OS X 上显示 Helvetica

这里将演示一个小技巧。不过在此之前，先介绍一些背景知识。

几乎可以确定 Arial 是万维网上用的最多的字体，因为 font-family: arial, helvetica, sans-serif；这条声明可谓随处可见。问题是，设计师通常更愿意在能使用 Helvetica 的情况下使用 Helvetica，但大多数 OS X 机器也安装有 Arial，因此在这个字体栈中，Helvetica 就被忽略了。

所有的 OS X 机器都有 Helvetica，但 Windows 上一般没有这个字体。即便哪个用户在 Windows 上安装了这个字体，其显示效果也不太如意。这是字体栈中不会将 Helvetica 列在 Arial 前面的原因。

怎么办呢？解决方案很简单，就是使用 font-family: sans-serif；。这种方法会让所有的操作系统使用无衬线字体——Windows 的浏览器显示 Arial，而 OS X 的浏览器则显示 Helvetica。

提示 系统通常都具有下列表示类属的字体名称对应的字体：serif、sans-serif、cursive、fantasy 和 monospace。因此，标准的做法是在字体栈的末尾指定上述字体名称中的一种。到目前为止，serif 和 sans-serif 是用得最多的，因为它们分别对应于最为常见的两类字体。serif 通常对应 Windows 上的 Times New Roman 和 OS X 上的 Times。sans-serif 通常对应 Windows 上的 Arial 和 OS X 上的 Helvetica。

^① cursive、fantasy、monospace 分别表示手写字体、装饰字体和等宽字体。——译者注