第6章　链接:文本、按钮、图标及图形

Web导航是通过选择与各个页面相联系的链接方式来完成的，尽管 Web遵循一种相对简单的链接模型，但网页间的链接方式极为丰富，从简单的 HTML文本链接变换到复杂的图形按钮，设计者应该明白，每一种链接方式都有自己的优缺点。特别应注意的是，要让用户很容易理解制作的链接，糟糕的链接一定通向一个糟糕的站点。链接的正确使用和维护可能是困难的，链接问题很容易出现。

6.1　基本的Web链接模型

Web的超文本链接相对较为简单。链接是传统上的单向链接，不创建任何特殊的程序，只会装载一个单页。在 Web上加载链接的主要方法是使用 HTML的<A>或“ 锚定 ”元素。使用 <A>元素包装能够触发链接的内容，并用如下所示的 HREF属性指定链接的目标：

```
<A HREF="URL of document to load">linked content</A>
```

当链接被激活的时候，HTLM属性将被设置到将装载页面的 URL中。

将链接分类的一般方法是根据要装载文件的地址或 URL。我们描述一个链接是内部的，就意味着它们能够和其他页面或站点的 URL相链接。内部链接也包括页面内的链接。有的链接使用户绕一个文件来回跳跃。一个页面内部链接的例子是常用的“ 回到顶部 ”链接，它经常在长页面上找到。

6.1.1　结构和非结构链接的比较

在Web站点内部，大致有两种形式的链接：不变的链接和变化的链接。例如，考虑导航条上的链接，这些链接在位置、形式以及页面间的指向位置上都趋于相当固定。我们称这些链接为结构链接，因为它们的使用和大多数 Web站点通用的层次结构很近似。结构性链接对带有计划任务如寻找什么东西和完成一项特殊任务的用户是很有利的。

另一方面，非结构性链接是那些在用户看来有些随意的链接。例如，文本内的文本性链接可认为是非结构性链接。如果在句子之中突然被建议可在 http://www.democompany/products/butler.htm看到机器人管家，你就会得到第一手例子，在一个网页上遇到这种情形时 ,一个非结构性链接会显得多么令人沮丧。为什么我在这个时刻给你一个到机器人管家的链接，并邀请你了解我的论证的思路？非结构性链接不需遵循站点的结构，任何时刻都可能在一个站点内跳跃或退出站点。但是，不要认为非结构性站点永远不被使用。考虑在你的页面上加载一些文本式跳跃和浏览链接——主要对文本中重要的字眼和段落。这些链接也许只是鼓励参观者停留或参观。并不是每一个人都有一个确切的任务；让用户闲逛一下也是有用的。

建议：偶尔在文件文本中提供一些非结构性的链接，以促进探索和思考。

然而，使用非结构性链接要很小心，如前一章讨论的那样，站点中的逻辑导航是可用性的核心。在用户看来随意的链接和许多交叉的链接在让用户高兴的同时，也容易使他们感到困惑。

6.1.2 静态和动态链接

在一个Web站点中，为链接分类的另一种方法与它们是如何创建的有关。链接是永远指向同一内容，还是根据内容动态地创建？分别称以上两种链接为静态链接和动态链接。

定义：静态链接中，目标文件被文档作者通过编码的形式确定。

大多数站点主要应用静态链接。这些链接形式的不足之处是链接的含义可以根据用户的意愿和上下文改变。一个可以根据环境变化的动态链接也许更好地适应交互式的 Web。

定义：动态链接没有一个固定的目标。目标文件是在浏览页的时候根据环境和浏览者的需要确定的。

动态链接对比静态链接有两个显著的优点。首先，动态链接根据用户的条件做出反映，因此他们根据用户的技巧、浏览器能力、用户偏好以及另外一些条件提供不同的目标。第二，动态链接提高了可维护性。Web站点的一个普遍问题是遇到无数断开的链接。一个可以动态决定链接的站点能够避免这些问题，因为当页面被加入和移走时，链接可以自动地重新确定。如本章随后的讨论，动态链接能够减少 Web设计者维护站点的负担。

6.2 链接形式的分类

Web的链接呈现出许多种形式，从基本的 HTML文本链接到复杂的带有不规则热区的图像（称图像映射）。每种形式的链接都有自己的用途，我们将逐一考察。考虑到链接的多种形式，设计者应特别小心确保链接的内容十分明显。记住：满屏滑动鼠标试图找到活动的点击区域，一般来讲对用户是一个枯燥甚至烦恼的事。在各种形式的链接讨论之后，将给出确保可用性的链接技巧。

6.2.1 文本链接

网页上最基本的链接方式是文本链接，它在 <A>标签内用文本来指明，如下所示。

```
<A HREF=http://www.democompany.com>Visit Demo Company -
                    Makers of the Robot Butler</A>
```

这些链接形式是多用途的；它们可用作基本的导航链接和大量文本中前后关系的文本链接。图6-1显示了链接的两种形式。

站点中文本链接的一般位置在页的底部，如图 6-2所示。这些备份链接经常用来模仿页面顶部的链接，或者提供代替图形链接形式如图像映射的链接方式。许多高级用户本能地滚动到缓慢装载的页面的底部，以寻找可以替代的文本链接。

建议：当使用长页或带有图形按钮的页时，要在页面的底部提供文本链接。

文本链接是很有用的，因为它们是很轻便的；它们的下载时间很少。更新它们或使它们完

全动态化是很容易的。

文本链接的主要不足是，它们经常不能被容易地识别出来，特别是设计者改变了链接的暗示如颜色和下划线时（本章随后讨论）。尽管固定导航链接也许相对容易识别，当链接装饰被动态地修改时，置身在内容中的文本链接变得几乎看不见。链接的颜色和装饰的可用性在本章的后面讨论。

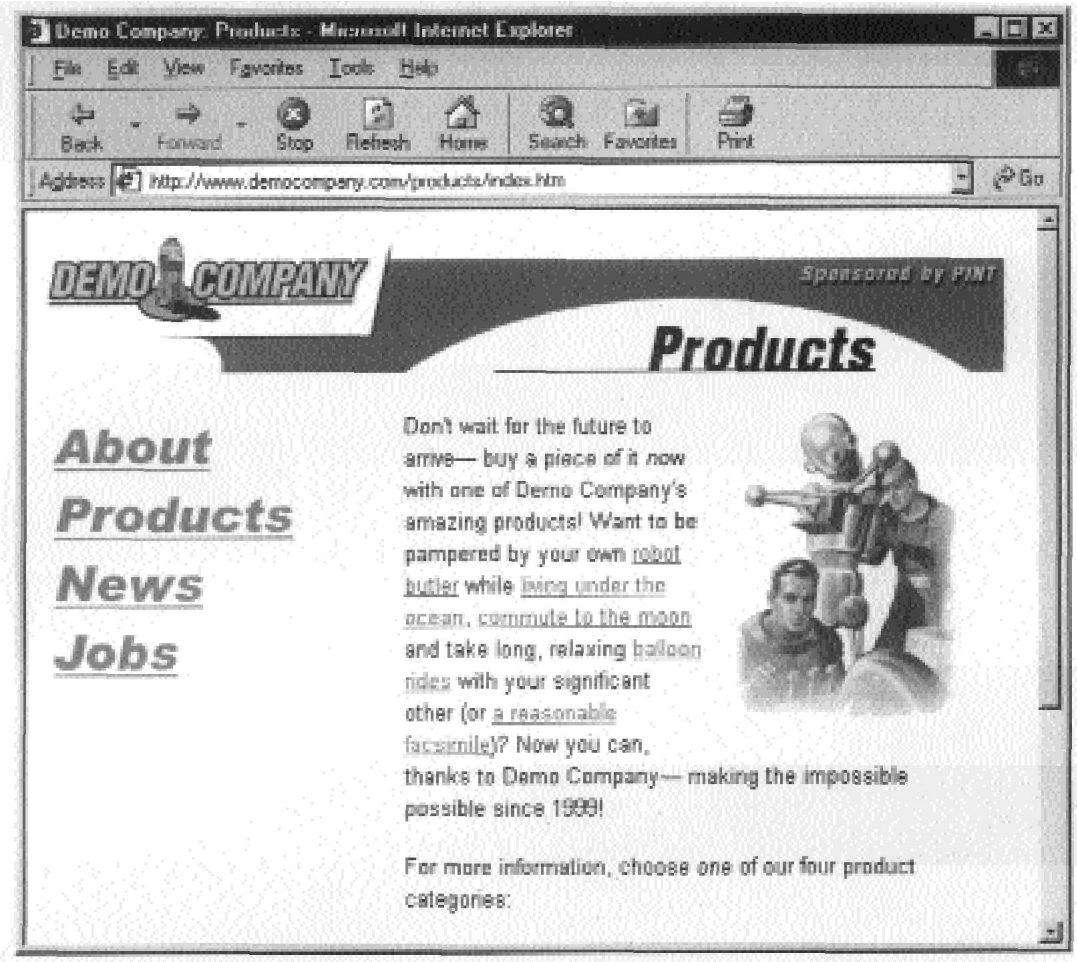


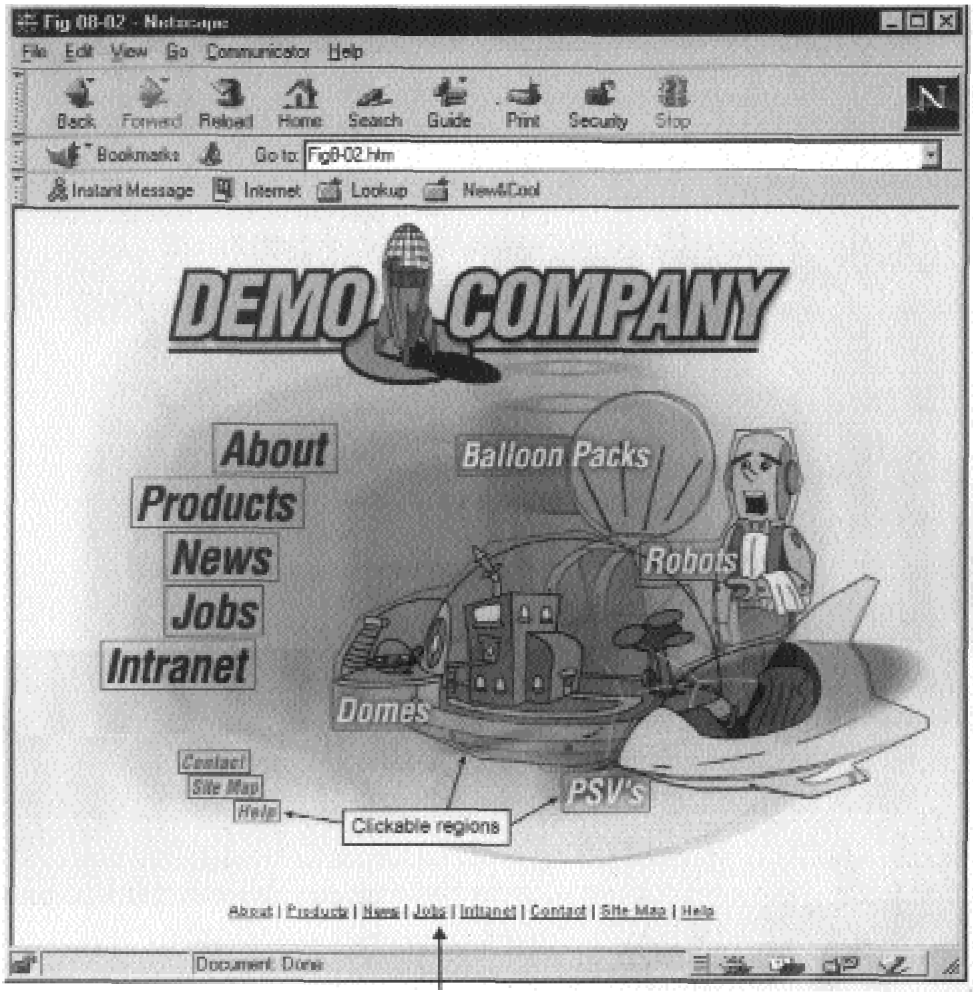
图6-1 使用文本链接

6.2.2 图形文本链接

由于文本链接相当简单，可能不支持Web站点的市场目标，一些设计者倾向于用图形文本链接。直到浏览器的第四版本为止，精确控制Web文本外观的唯一方法是在图形中做一幅图像。对有意义的导航链接，设计者经常创建图形文本按钮。

为了确保这些图形文本项看上去是可点击的，设计者经常为按钮选择不同的字体，改变它

们的颜色，增加它们的尺寸，使文本远离其他内容或采用阴影等特殊效果。图 6-3列出了一些制作图形文本时处理文本的例子。



文本链接是图像映射的备份

图6-2 作为备份导航的文本链接

图形文本按钮的一个主要的不足是，即使得到优化，它们仍需要大量的下载时间，特别是在与滚动状态结合的时候。不管做了多少优化，在简单文本中的“ About ”将始终比包含它的GIF图像要小得多。况且，在许多情况下图像按钮也许会限制可访问性。没有可选择的文本，它们就不能被转换到不支持图形的环境；即使在图形环境下，也与相应的分辨率不符。

1. 用Fireworks创建时尚的文本链接

作为创建图形按钮的一个短小的示例，我们将用流行的图片编辑程序 Macromedia Fireworks 创建一个小的导航条（www.macromedia.com/fireworks）。

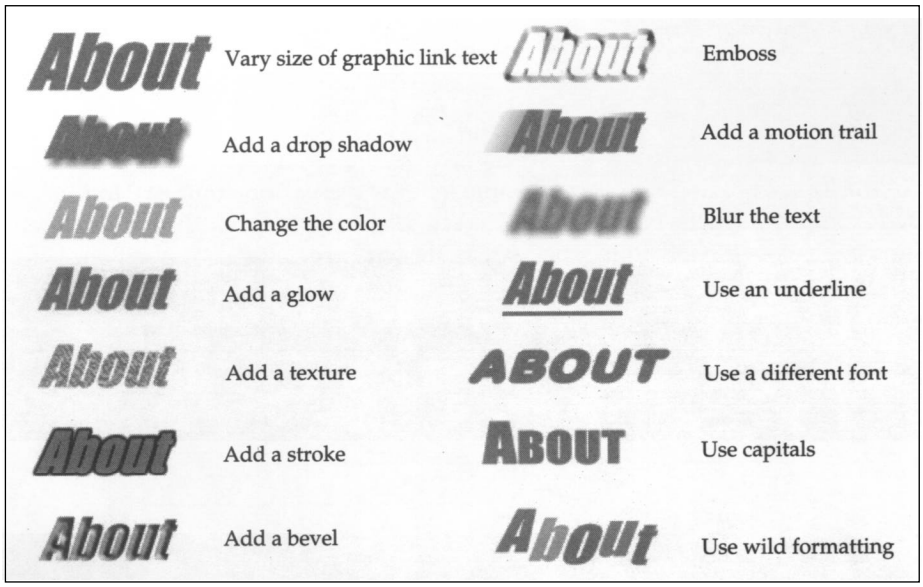


图6-3 图形文本按钮的例子

在职业图形设计者中，AdobePhotoshop（www.adobe.com）可能是最好的图片编辑程序，但是使用Fireworks 3能很容易地创建按钮和与Web站点图形有关的事宜。这个事例的目的是展示人们在创建图形按钮时的基本方法。

应用Fireworks 3（或后来的版本），创建一个大约250像素高200像素宽的小文档。

从左侧的工具调色板上选择与下面图标一样的文本工具。

在文本工具对话框里，给不同的按钮加上一些文本。在这个例子中，加上 About、Products、News和Jobs。我们为文本选择一个36点Impact字体和红橘色的颜色，如图6-4所示。

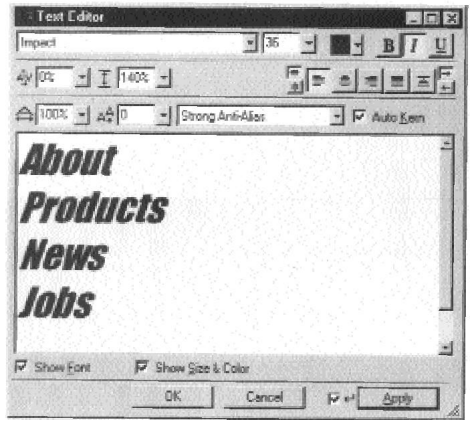


图6-4 选择颜色

当把文本插入到文档后，用箭头选择Fireworks文本框并从窗口菜单中选择效果。

从Effect窗口的下拉菜单中选择Shadow和Glow，然后选择Drop shadow。调整阴影效果的程度使之看起来像坐在页面上。根据这一要求，将使得文本比页面上的正常文本显得更突出，看上去可以点击。实现这一效果的最后对话框如图6-5所示。

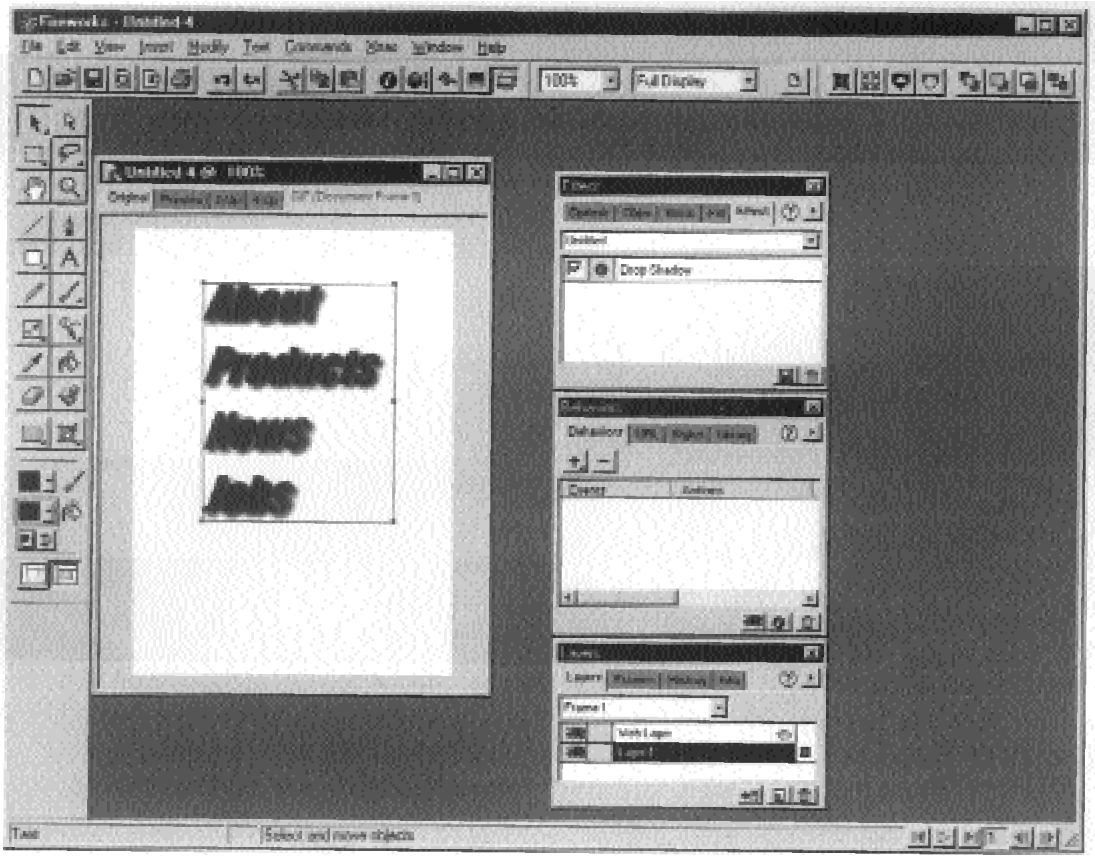


图6-5 在Fireworks中创建文本阴影

这时文本图片可以直接输出，作为一个单独的图像或单个的片段。使用 Fireworks，有可能在按钮中加入图形文本。

首先，使用箭头按钮选择文档中已存在的文本框。

从插入菜单中，选择Hotspot，在文本框的顶部加入一个热点。

调整热点的尺寸，热点区域显示浅蓝色，在文本之上激活它，然后如图 6-6所示设置目标URL和ALT文本。

注意，使用带有热点区域的单个文本将产生一个图片映射。有可能分别创建各个按钮。过程是类似的，在本章后面会以同样的方式在 Fireworks中制作按钮。

无论采用什么样的图形工具，在制作图形文本按钮时，一定要把按钮制作得和文本整体或站点上使用的页标签有显著的不同。使用不同的颜色，较大的尺寸，不同的形式如斜体或阴影

效果，将有助于用户理解图形文本元素是可点击的。

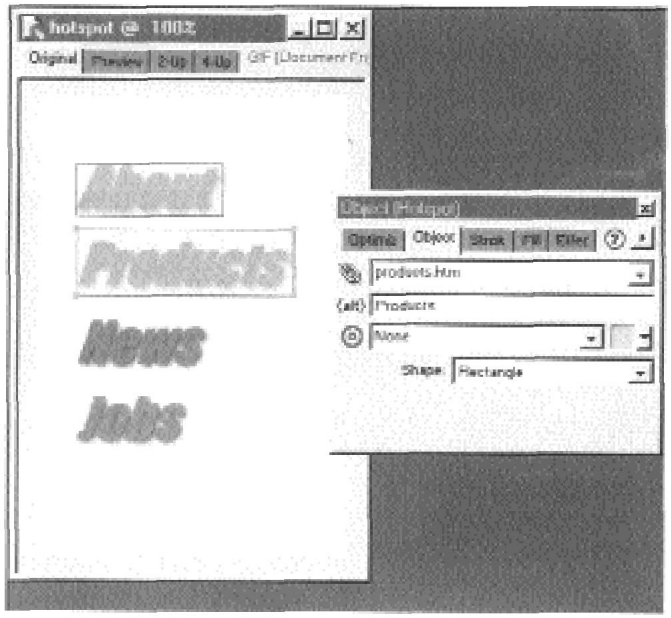


图6-6 设置VRL 和ALT 文本

2. 在CSS中创建时尚的文本按钮

图形按钮的一个主要缺点是下载的时间较长。随着层叠样式单的兴起，现在有可能不使用图形编辑器而创建令人满意的文本链接。如前一节创建的阴影链接，创建稍微不同颜色的两个字并把它们彼此偏移开来。以下所示 HTML的标记范例显示了这一过程：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>CSS Styled Text Links</TITLE>
<SCRIPT>
<!--
(document.layers || document.all) ? css=true : css=false;
//-->
</SCRIPT>
<STYLE>
<!--
.dropshadow {position: relative;
left: 50px;
   font-size: 30px;
   font-family: Impact;
   font-style: oblique;
   color: black;
   font-weight: bold;
   letter-spacing: 2px;}
```

```
 .buttontext {position: relative;
top: -35px;
left: 46px;
      font-size: 30px;
      font-family: Impact;
      font-style: oblique;
      color: #CC3300;
      font-weight: bold;
      letter-spacing: 2px;}
A {text-decoration: none;}
A:hover {text-decoration: underline;}
-->
</STYLE>
</HEAD>
<BODY>
<A HREF="about.htm">
<SCRIPT>
<!--
if (css)
  document.write('<DIV CLASS="dropshadow"> About </DIV>');
//-->
</SCRIPT>
<DIV CLASS="buttontext"> About </DIV>
</A>
</BODY>
</HTML>
```

注意 Web链接：能够在线寻找到这些代码：

`http://www.Webdesignref.com/chapter6/csstextlinks.htm`。

注意，这些代码主要采用 CSS。一个不支持 CSS的浏览器将会输出文本链接“ About ”两次，所以在加载第二个文本链接时，需要用 JavaScript为4.0版本或更好的浏览器做检测。即使在支持CSS的浏览器中，结果也是不定的。图 6-7展示了IE4.0、Netscape4、Netsape3窗口下的显示。

CSS2定义了可用于创建阴影的 text-shadow属性。为了创建一个特殊的阴影类，可以如下所示定义一条CSS规则：

```
.dropshadow   {text-shadow: black 0.2em 0.2em}
```

这将在一个对象的右下侧放置一块小的黑色阴影。设置负的值能够改变出现在对象的左侧或上部的阴影。也可以设置一个模糊值。

```
H1 { text-shadow: red 4px 4px 5px}
```

这就指定了所有在 <H1>中的元素在右下部有 4个像素的阴影，半径为 5个像素。尽管在CSS中文本阴影的确简化了阴影的创建，但 4.X和早期的5.X版本浏览器不支持它。

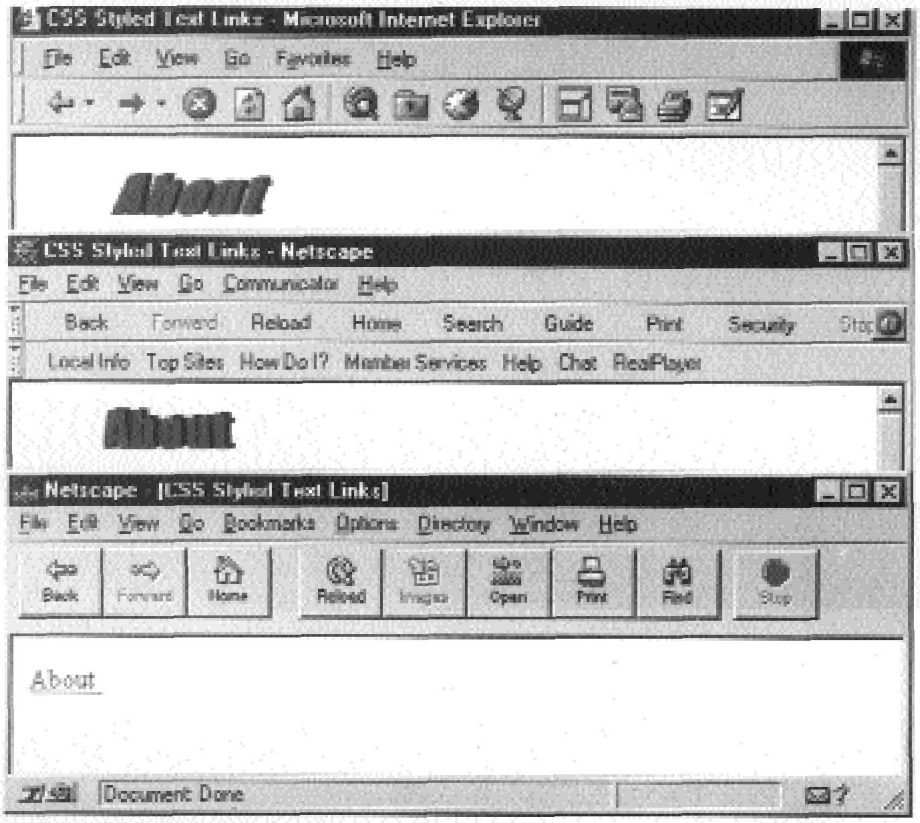


图6-7 CSS文本链接显示

6.2.3 按钮

将链接创建得像一个按钮，是提高站点链接可用性的一个好办法。制作带有图形的自定义按钮和使用HTML形式的按钮是可能的。如图6-8所示，有许多可以使用的按钮形式。

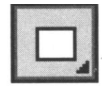
1. 在Fireworks内创建按钮

用photoshop或fireworks工具创建图形按钮是相当直接的。

以下的例子将再次使用Macromedia的Fireworks工具。

从范例文档中，从Insert菜单中选择New Button命令。

在Button对话框中，用主Tool调色板上的Rectangle工具画一个矩形。“矩形工具”图标如下所示。



双击工具打开选择项，画一个圆角的矩形。在这个例子中，画一个矩形，各个角的值为60。

把按钮的背景颜色设置成你喜欢的任意颜色。在这个例子中，我们使用#FFCC00。

现在，从Window菜单中选择Effect使按钮看起来是可按下的。

从Effect对话框中，从下拉菜单中选择“斜角”和“浮雕”。现在将值调整为菜单创建斜角的尺寸。

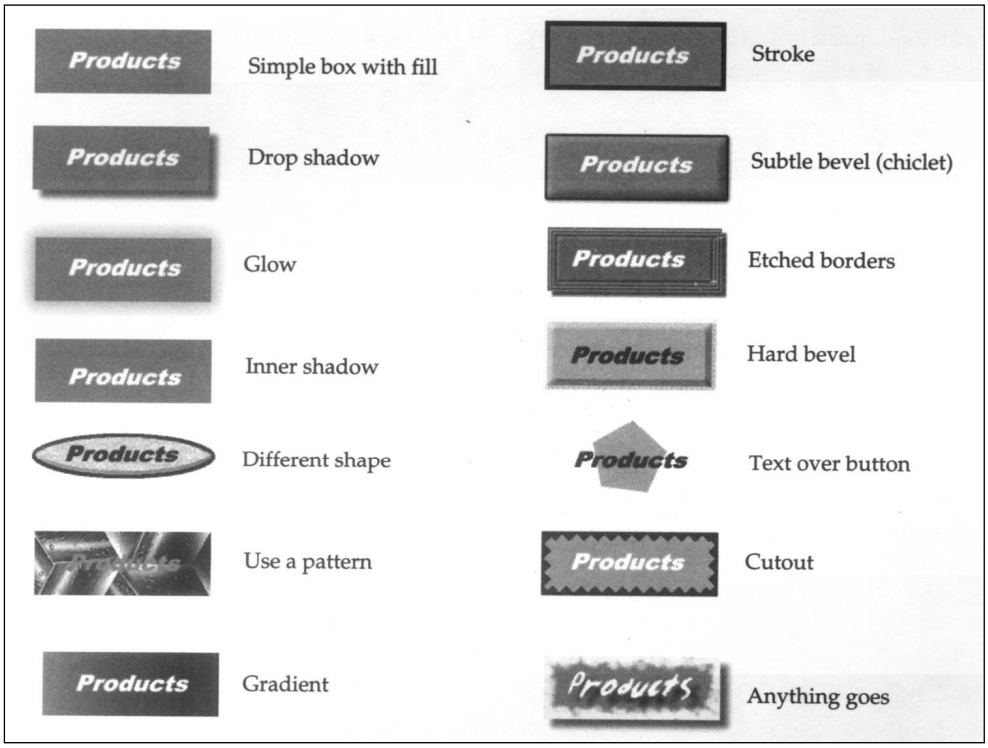


图6-8 按钮样式的例子

一旦按钮制作完成，加一些白色的文本作为按钮的标签。就会在屏幕上得到类似于图 6-9所示的东西。

一旦完成了一个按钮，就复制并改变文本标签。 Fireworks支持可用于按钮的许多效果，甚至允许轻松地创建称为“滚动按钮”的多状态按钮。本章将随后讨论这一点。

2. 用html创建按钮

像图形文本链接一样，图形按钮确实导致下载困难和更新困难。其优点是它们经常比标准链接更接近可视化的要求。在更高级的 HTML和CSS中，创建自己的轻便按钮是可能的。

在HTML中创建按钮的最简便的方法是使用颜色表格，把单元表格设置为一种背景颜色，并把边界的状态置为“开”，能够创建十分漂亮的文本模式的按钮，如下所示。

Products

尽管这看起来像一个按钮，但实际上不是，因为只有文本是可点击的。这是设置下划线的另一个原因。使用以下的标记，创建所示的按钮：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>HTML Table Buttons</TITLE>
</HEAD>
<BODY LINK="black" VLINK="black" ALINK="white">
<TABLE BORDER="1" BORDERCOLOR="black">
<TR>
<TD BGCOLOR="#CC3300">
 
<FONT FACE="Verdana,Helvetica,sans-serif" SIZE="+2">
<A HREF="products.htm">Products</A>
</FONT>
 
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

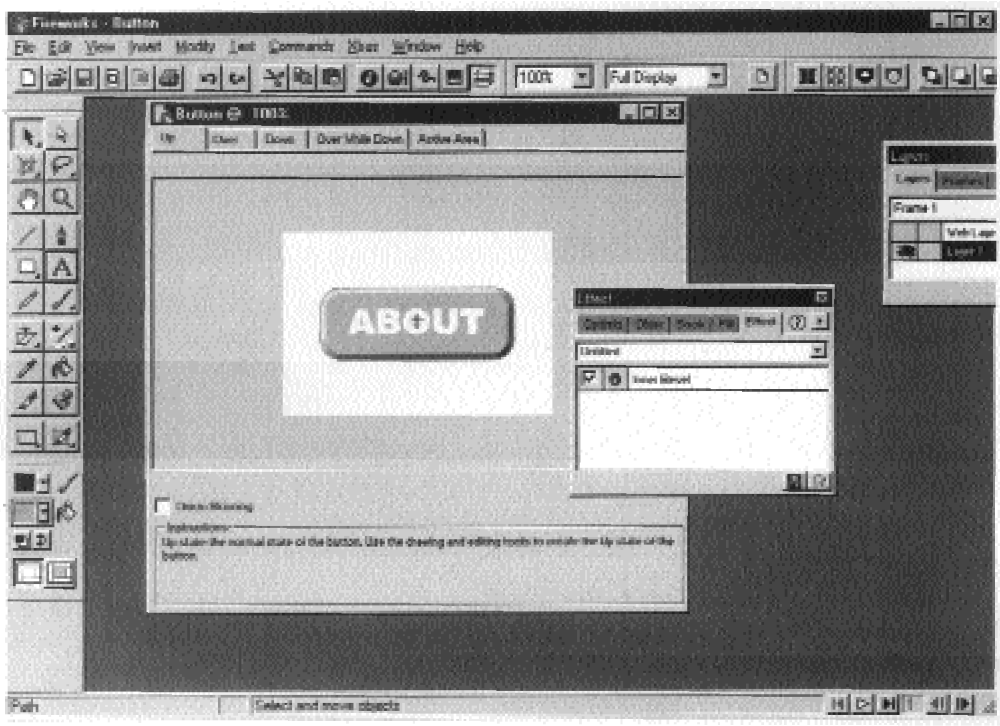


图6-9 在Fireworks中创建按钮

注意 Weblink:在http://www.Webdesignref.com/chapter6/htmltablebuttons.html上可以在线获得代码。

注意 一些早期的浏览器允许设置文本的颜色，但不能设置表格的颜色。如果使用这些浏览器，当在有颜色的表格上使用白色文本时，由于页面本身的背景是白色的，按钮文本将无法显示。

使用<INPUT>元素也可创建简单的按钮：

```
<FORM>
  <INPUT TYPE="BUTTON" VALUE="Yahoo!"
onClick="location='http://www.yahoo.com'">
</FORM>
```

遗憾的是，当程序打开的时候，这个特殊的例子只在支持 JavaScript的浏览器上奏效。使用Submit按钮，有可能处理各种浏览器类型：

```
<FORM METHOD="GET"
              ACTION="http://www.yahoo.com">
<INPUT TYPE="SUBMIT" VALUE="Yahoo!">
</FORM>
```

注意 这可能奏效，但有一个缺点。正如在提交窗体时，代码将在试图访问的 URL上附加一个问号。不像实际的窗体提交，没有传递参数。因此问号后面没有跟随字符串或符号。URL不寻常的外观也许会困惑用户。

给出了窗体按钮的外观，并且用户相信它们与一些窗体如次序窗体有关，设计者就被警告不能使用 —— 窗体按钮触发链接。

3. 用CSS创建按钮

应用CSS，有可能创建所有形式的简单按钮。在后面我们将会看到与 javaScript结合起来，甚至可以使按钮真的动作起来。这里的代码创建了一个简单的带有阴影的按钮：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>CSS Buttons</TITLE>
<STYLE>
<!--
.dropshadow {position: relative;
             left: 52px;
             top: 5px;
             font-size: 30px;
             font-family: Impact;
             font-style: oblique;
             background-color: black;
             font-weight: bold;
             letter-spacing: 2px;
             height: 30px;
             width: 100px;}
```

```
.buttontext {position: relative;
             top: -35px;
             left: 46px;
             font-size: 30px;
             font-family: Impact;
             font-style: oblique;
             background-color: #CC3300;
             color: white;
             font-weight: bold;
             letter-spacing: 2px;
             height: 30px;
             width: 100px;}
A {text-decoration: none; cursor: hand;}
-->
</STYLE>
</HEAD>
<BODY>
<BR><BR>
<A HREF="about.htm">
<DIV CLASS="dropshadow">   </DIV>
<DIV CLASS="buttontext">  About  </DIV>
</A>
</BODY>
</HTML>
```

注意 Weblink: 可以在http://www.Webdesignref.com/chapter6/cssbuttons.html在线获得代码。

使用CSS的边界属性。可以很容易地创建所有形式的带有对角、颜色，甚至图案背景的可点击区域，如下例所示：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//E
<HTML>
<HEAD>
<TITLE>CSS Buttons 2</TITLE>
<STYLE>
<!--
.button   {border-style: inset;
           border-color: #0066FF;
           background-color: #CC3300;
           background-image: url("blueswirls.gif");
           width: 80px;
           text-align: center;}

A.buttontext {color: white;
              text-decoration: none;
              font: bold 12pt Verdana;
              cursor: hand;}
-->
```

```
</STYLE>
</HEAD>
<BODY>
<A HREF="about.htm" CLASS="buttontext">
<DIV CLASS="button">About</DIV></A>
</BODY>
</HTML>
```

注意 Weblink：可以在http: //www.webdesignref.com/chapter 6/cssbuttons2.htm在线获得代码。

在IE浏览器中，先前的代码产生如下形式的按钮：

遗憾的是，由于浏览器的CSS实现的错误，这个例子在Netscape中并不能很好地显示。即使在IE浏览器中，也不允许轻易地改变文本比例，因此明显地设置宽度并不是最好的方法。然而，这个演示说明在不久的将来会创建时尚的按钮，再也不需要处理图形。

6.2.4 图标

一个图标是一幅小画，用来在屏幕上表示某些动作和内容，图标可以单独使用或与文字联用。通过使用图标，可以节省空间。一个可视图标经常被认为与几个单字所占的像素差不多。考虑下面的图标和与它们等量的文字：

一个图标也很容易地像古代埃及的象形文字一样易于识别。考虑下面常用系统图标的意思：

不带有标签的图标	对应的含义	
	Internet options	World time
	Add/Remove programs	Tools
	Regional settings	Directory assistance
	ODBC data sources (32 bit)	

如果没有标签，除非意思很简单，否则描述图标的意义和内容是很困难的。没有与之关联的文本标签，理解某些图标也是有困难的。地球的图标也许很容易理解，但它在页面文本中的意义可以是一些跨国公司的全球主页的入口，或者有关地理方面链接的变化。文本可以澄清图标的意思。考虑带有标签的同样的图标，它们是相当容易理解的。

遗憾的是，如果用标签表示，图标就会失去它们节省空间的优点。使用 TITLE属性，如本章随后讨论的一样，就有可能隐藏图标，直到用户把指针放在图标上。但是，站点设计者应尽量在能显示图标的地方显示图标，以避免用户不得不点击按钮来判断它的意思。

撇开标签的用途不谈，图标比文本具有某些优势。图标在一般情况下比文字更容易被用户识别。即使人们很难描述图标，但在一段时间过后与文字相比，人们更容易回忆起图标的意思。想像一下与姓名相比，人们是否更容易记住一个人的面孔？想一想在许多桌面应用程序上使用的粘贴图标。

剪贴板真的需要声明是“粘贴”吗？你什么时候考虑过它？你能告诉说这是一个小尺寸的剪贴板吗？一段时间过后，图标代表什么不再是问题了，因为用户知道当他们按下文件放大镜的图片按钮时，将得到一个打印预览。从这种意义上说，许多常用的图标对用户来说已经变得很习惯了。随着Web的发展，一些按钮必然变得很普遍，并且它们的意思相当容易理解。对桌面应用程序来说，绝大多数按钮具有某些继承性，但另一些也许是新的。表 6-1给出了一些应用于Web的按钮以及它们的典型意义。

注意 采用特殊样式的图标是不合理的。记住，形状是图标的首要关注对象。

这并不是一个完整的图标列表，所有的 Web用户并不能始终理解这些按钮，尽管大多数人感觉它们很熟悉，特别是和文本标签连用时。如果你有意创建自己的按钮，就有可能应用任何图像处理程序。但按钮生成工具如 AX-Icons也许使程序更简单。

小心不要落入这样的陷阱：认为不需要用户真的阅读什么，图标就能够提供完美的、透明的Web导航。尽管用户不需要必须去理解一个图标，但考虑一下随后会发生什么。如果一个文盲用户或一个非母语用户能够理解一个图标的意思，但因为缺乏语言技巧，他们也许在不能理解内容的情况下结束浏览。总而言之，不要认为图标能够解决所有的本土化问题。正如第 3章讨论的，可用性赋予界面设计某些诀窍。

表6-1 Web站点上可以找到的通用图标

任务图标	
（发送Email或联系）	
打印	

（续）

任务图标	
发送（讨论面板上的消息）	
讨论或谈话	
评论	
联系	
下载或保存	
书签	
访问购物卡	
编辑	
删除	
关闭	
附加	
导航图标	
主页	
返回	
向前	
向上（页顶或一个层次的返回）	
帮助	
搜索	
（同一个层次）其他事情	
内容图标	
音频	
视频	
图形附加	
图片放大	
Acrobat文件	
新信息	
“酷”信息	

6.2.5 图像映射

许多可视化Web界面应用了图像映射或带有可点击区域的称为热点的大图像。图像映射是很流行的，因为它们为设计者提供了创建任意形状按钮的能力。在按钮和图标的情形下，点击区域是正方形或矩形。使用图像映射，矩形、圆形和任意形状的多边形都是可能的。

HTML定义的图像映射有两种形式：客户端和服务器端。服务器端的图像映射是通过包括ISMAP属性和指向一个远程服务器的映射文件链接来定义的。如下面这个简单的例子所示：

```
<A HREF="shapes.map">
<IMG SRC="shapes.gif" ISMAP BORDER="0" WIDTH="400" HEIGHT="200">
</A>
```

这个映射文件包含暗示热点的坐标，还有当热点被选中时指向的 URL。一个简单的映射文件看起来和下面的文件类似：

```
rect rectangle.htm 6,50 140,143
circle circle.htm 195,100 144,86
poly polygon.htm 256,120 306,52 333,58 336,0 386,
73 372,119 322,172 256,120 256,119 256,119 258,118

default defaultreg.htm
```

服务器端的图像映射问题是双重的。首先，依据他们在哪里点击申请对服务器的访问，以及服务器对映射文件的解释以确定用户应该去哪里。Web路由的兜圈可能减慢用户的速度。其次，当用户的鼠标在图像的不同部位移动时，坐标——不仅仅是URL——将显示在状态条上（见图6-10）。

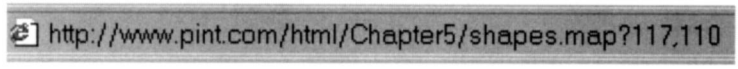


图6-10 状态条

指针的坐标对那些经常查阅 URL以获得链接目标信息的用户来说并非是最理想的反馈。幸运的是，服务器端的图像映射基本上已经过时了。所有的现代浏览器支持客户端的图像映射，一个客户端的图像映射用 <IMG>元素的USEMAP属性来定义。USEMAP用于设定指向文件某处的<MAP>单元，从而显示图像中的热点区域。下面的简单例子显示一个基于客户端的图像映射所需要的HTML代码。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Client-side Imagemap</TITLE>
</HEAD>
<BODY>
<H1 ALIGN="CENTER">Client-side Imagemap Test</H1>
<DIV ALIGN="CENTER">
<IMG SRC="shapes.gif" USEMAP="#shapes" BORDER="0" WIDTH="400"
    HEIGHT="200">
</DIV>

<MAP NAME="shapes">
<AREA SHAPE="RECT" COORDS="6,50,140,143"
      HREF="rectangle.htm" ALT="Rectangle">
<AREA SHAPE="CIRCLE" COORDS="195,100,50"
      HREF="circle.htm" ALT="Circle">
<AREA SHAPE="POLY"
     COORDS="255,122,306,53,334,62,338,0,388,
             77,374,116,323,171,255,122"
```

```
     HREF="polygon.htm" ALT="Polygon">
<AREA SHAPE="DEFAULT" HREF="defaultreg.htm">
</MAP>

</BODY>
</HTML>
```

注意 Weblink: 可以在www.webdesignref.com/chatier 6/clientsidemap.htm获得在线代码。

尽管在一个图像中指定活动区域需要令人生畏的代码，创建一个图像映射并不是很困难的。许多图像映射工具可用来创建图片。流行的编辑器如 Macromedia的Dreamweaver提供了简单的图像映射编辑工具，如图6-11所示。在定义一个图像映射以前，要确保不再需要修改这幅图，如果要调整图像尺寸或移动什么，热点不能相应调整，或许它们不再相关。制作图像映射一般来说是布局的最后一部分。

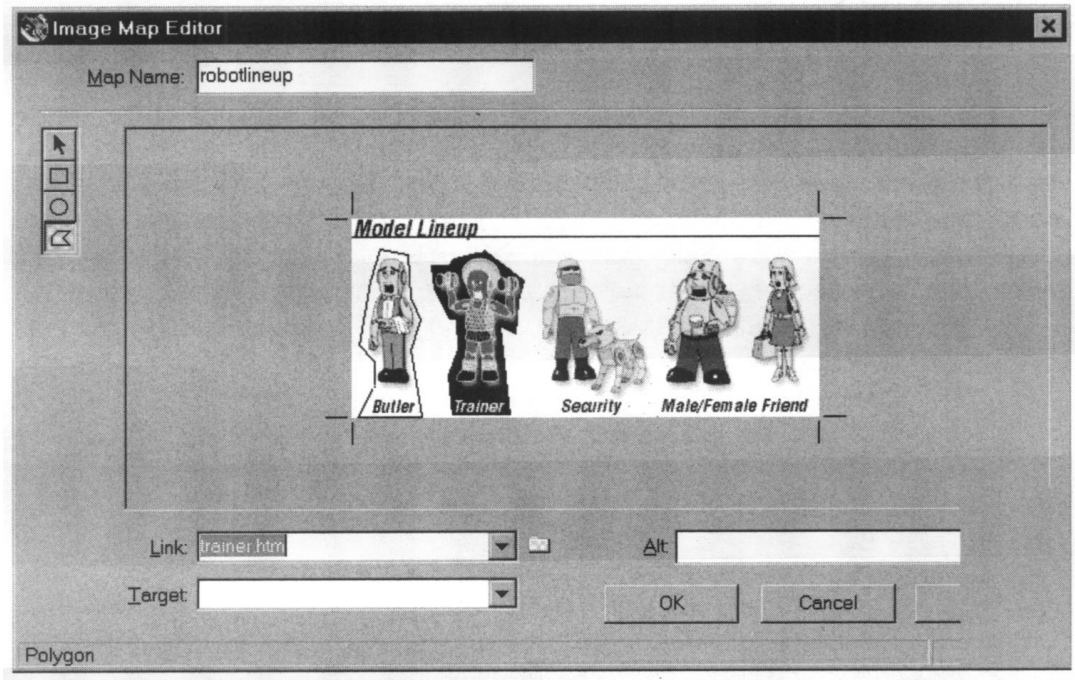


图6-11 用Dreamweaver绘制图像映射

图像映射一个主要的好处是：它们允许不规则的热点，它能用来创造有意义的界面如图 6-12所示。在这个例子中，在机器人不同的部位滚动，就会显示相应的特征信息。图像映射的第二个好处是减少了向一个服务器请求服务的次数，设计者经常剪下菜单条放入多重图片中，特别是试图创建滚动图像时，第二代使用图像映射的滚动代码在本章之后讨论。始终记住文件尺寸并非一切；装载有许多小变化的页面时，和服务器链接的次数也许会显著地减慢装载页面的速度。因为一个图像映射也许包含许多热点，一个有多幅图像请求的大的导航条能够减小为单个请求 —— 尽管这将导致图像映射相当大。

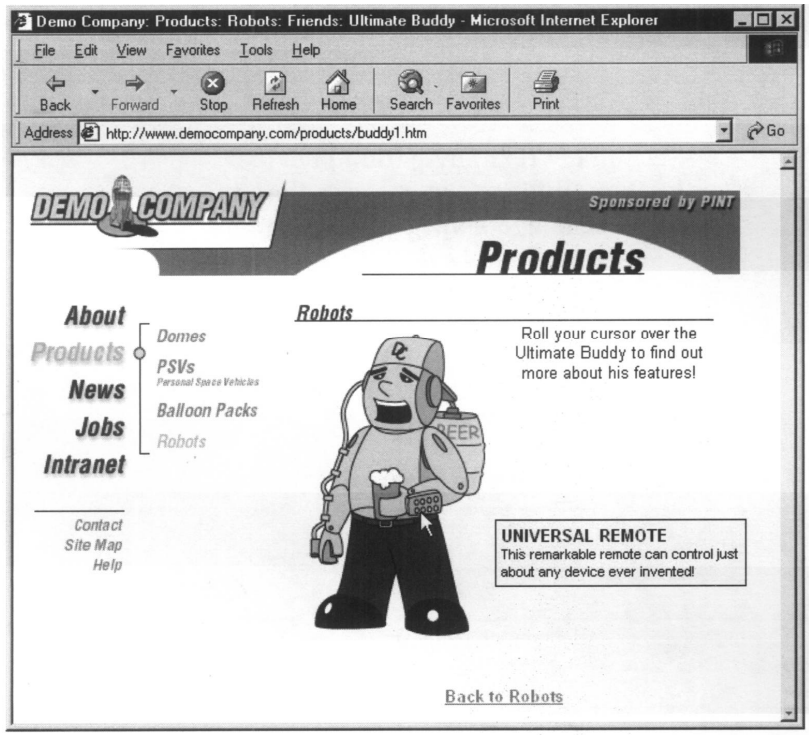


图6-12 使用图像映射的交互界面

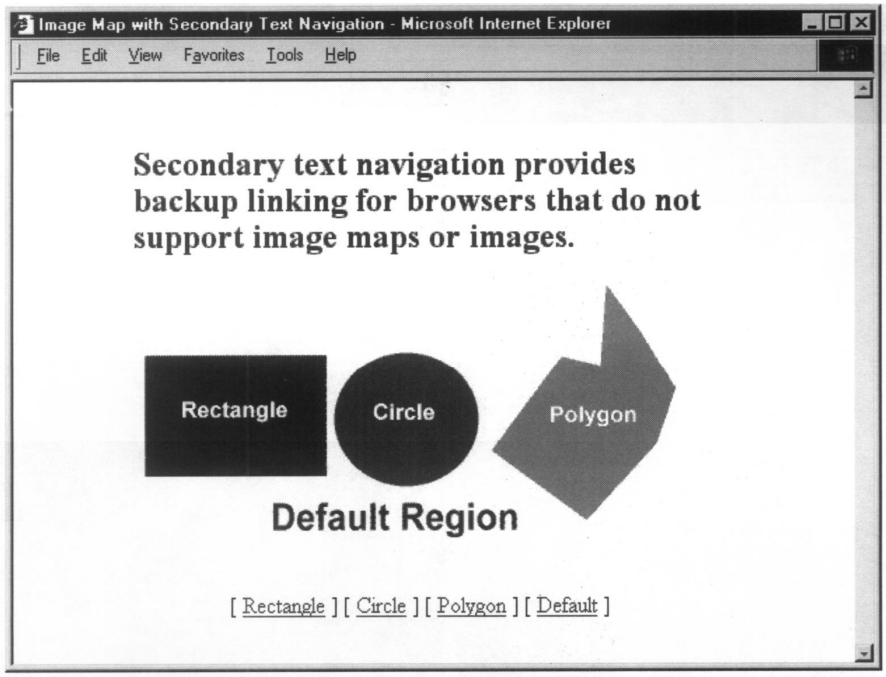


图6-13 文本链接提供了图像映射的替换方案

图像映射的主要问题是它们通常使用十分奢华的布局，这也许会显著延长下载的时间。进一步说，图像映射趋于比标准的链接方式更难以访问，特别对不支持图形显示的环境来说。正因为如此，设计者始终被鼓励为图像映射提供辅助的导航链接，如图 6-13所示。

建议：当使用图像映射时，应始终提供辅助的导航形式，如文本链接。

6.2.6 其他链接方式

除了文本链接、按钮和图像映射外，还有许多其他对象可用来触发页面装载或其他动作，最一般的是特殊形式的称为标题广告的广告按钮和修改的窗体元素。但是，随着 HTML 4.0的兴起，有可能使用其他的对象触发页面装载。

1. 标题广告

考虑到许多站点的商业性质，会经常发现标题广告。有多种尺寸的标题广告，并且经常是动画的。点击广告一般将带用户到广告站点。广告的有效率用它的点击率来衡量，意思是指看到这个广告的人数和实际点击它的人数的百分比。遗憾的是，经过了几年，标题广告的点击率直线下降。一些专家把这称为“标题广告盲点”的现象。基本上，对标题广告视而不见意味着用户对标题广告的尺寸、形状、放置都很熟悉，以至他们能对其视而不见。正因为如此，设计者应尽量避免把非广告性链接制作成这种样式。

技巧 避免按钮在尺寸和风格上与标题广告类似。

不管在标题广告上所附加的特殊用途如何，它们是一种普通的链接形式，理解它们的规则是很重要的。因特网广告局指定了标准标题广告的尺寸，如表 6-2所示。

表6-2 通用的标题广告尺寸

标题类型	像素尺寸
全标题	468 x 60
带有垂直导航条的标题	392 x 72
半标题	334 x 60
正方形标题	125 x 125
垂直标题	120 x 240
#1按钮	120 x 90
#2按钮	120 x 60
微型按钮	88 x 31

不同于表中所示的标题广告形式也可以见到，这依赖于具体的站点。进一步说，千字节大

小的标题广告以及动画的可能性在不同的站点是不同的。比如，在写本书之际，标题广告 Web LinkExchange把标题广告限制在 10kb、7s的动画时间并且没有循环。在创建标题广告时确保检查了标题广告的规范。

2. 为触发链接使用GUI小件

许多站点把各种形式的小件应用于导航。可能用于触发链接的最一般的形式是下拉式菜单。许多站点应用下拉式菜单实现在页面间移动的迅速跳转功能。

下拉式菜单变得很普通，对用户来说显得很容易理解，其他用于导航的元素应避免使用。特别地，单选按钮和复选框都不能以这种方式应用。但是，对一般形式按钮的使用尽管不鼓励，也是可以接受的，但要明白这也许是在向用户暗示某个功能不是页面装载。对代码和使用 GUI小件的命令问题的全面讨论将在第 12章进行。

3. 热点到处都有吗？

随着 HTML 4或更高版本的引入，通过使用 onClick核心事件处理程序，并把它附加在 JavaScript上，每一个HTML元素几乎都是可以点击的。例如，把一小段 JavaScrpt附加在onClick 的属性上，试一下下面的HTML标记：

```
<P onClick="location='http://www.yahoo.com'">
Is this a link? Maybe or maybe not.</P>
```

如果使用IE4或更高版本的浏览器，点击段落可能会加载一个新的页面。但是，如果关闭了对脚本的支持或在使用一个老版本的浏览器，就什么也不会发生。忽略浏览器的支持，问题的潜在性是显而易见的。如果任何元素都可以点击，困扰用户的可能性就变大了。下一节将解决此问题，让用户知道页面上哪些是可点击的，哪些不是。

6.3 链接的实现问题：可用性、反馈和支持

不管站点使用了什么形式的链接或链接的组合，当涉及到站点链接的实际实现问题时，将遵从一定的界面设计考虑：链接应增强而不是减弱一个站点的可用性。他们应该向用户提供足够多的反馈，应该提供支持站点的特性，而不是让自身承担整个站点的导航任务。下一节通过一系列与之相关的命题，讲述了链接的一般性实现问题：链接的可用性，滚动，用户的期望，范围注释的使用，最后是对链接的键盘支持。

6.3.1 可用的链接

不管喜欢与否，用户都期望链接是蓝色的并带有下划线。进一步说，他们希望按钮在字面上看起来是可按下的，这就意味着使用常见的效果如倾斜或阴影。遗憾的是，一些设计者并没有使用这些惯例来设计符合要求的布局，经常是彻底地忽略了它们。在为了样式而违反规则之前，重要的是理解链接的可用性。始终记住：如果用户不能很好地工作，再好看的站点也无济于事。

1. 链接的颜色

Web上对没有访问过的链接的缺省颜色是蓝色，被激活（按下）的链接的颜色是红色，对访问过的站点是紫色。在 CSS2规则下，当一个指针盘旋在链接上并即将选择一个链接时，改变链

接的颜色是可以的。

使用HTML或CSS，链接的颜色很容易被忽视。在 HTML中修改链接的颜色要求为<BODY>元素修改属性，而CSS依赖<A>元素的伪类规则。表6-3显示了链接的类型、颜色和修改句法。

注意 在链接颜色的确定上一定要对CSS予以认真的考虑。在一个统一的链接样式单上定义的次序是：没有访问的，访问过的，盘旋的，最后是激活状态。考虑到重叠将引起潜在的破坏，任何其他次序将产生不正确的结果。

表6-3 链接类型和颜色

链接的类型	标准颜色	HTML标签	CSS 伪类规则
没访问过的	蓝色	<BODY LINK="colorvalue">	A:link{color: colorvalue}
访问过的	紫色	<BODY LINK="colorvalue">	A:visted{color: colorvalue}
盘旋的	N/A	N/A	A:hover{color: colorvalue}
处于激活状态的	红色	<BODY LINK="colorvalue">	A:active{color: colorvalue}

显著地改变蓝色/红色和紫色等链接颜色是危险的。不管是好还是坏，用户已经认为蓝色是链接的颜色，紫色是他们已经点击过的链接的颜色。改变颜色是重要的，因为用户可以据此知道他们曾到过哪里。在Hansel和Gretel神话中被称为“面包屑法”，童话中小孩们通过丢面包屑找到走出森林的路，遗憾的是，站点设计者经常因为市场原因破坏这些规则。如采用蒙蔽的手段，修改访问过的链接的颜色，从而误导用户重新访问该页。尽管这乍看起来似乎是鼓励多浏览页面的一个好办法，但用户在重复浏览后会感到沮丧。

规则：不要彻底改变已访问过的链接的表示。

另一个改变链接颜色的原因也许是因为美的缘故。蓝色和紫色的连用也许不是合适的合成色。当然，一旦改变，已访问过的链接的颜色看起来更好，但最终用户也许认不出链接或知道哪些链接他们已访问过。

规则：避免改变链接的颜色。

如果链接的颜色由于某种原因必须改变，一定要确保访问过的和没有访问过的链接有明显的不同。同样，要使链接的颜色和背景颜色的区别较大，以便能够很容易地识别。糟糕的颜色选择使得站点对那些有视觉缺陷的人来说是不可行的。关于颜色在Web中的使用的深入讨论见第11章。

2. 链接装饰

在一般的浏览器中，链接经常不仅通过颜色而且通过下划线表示。当用户对颜色的改变不太敏感时，第二种反馈形式是特别有用的。正因为如此，设计者应在设计中对下划线的使用很敏感。HTML＜U>标志和CSS的文本装饰特性都能用于创建下划线，如下所示：

```
<U>This looks like a link</U><BR>
<SPAN STYLE="text-decoration: underline">
This also looks like a link</SPAN>
```

遗憾的是，当用户点击它并认为它是一个链接时，这种形式的文本能困惑用户。这引出了下面的规则：

规则：避免在Web文件中对非链接文本加下划线——对这种情况应采用斜体或粗体。

尽管下划线提供了超越颜色的第二种形式的反馈，一个充满下划线的页面看上去不太令人舒服。正因为如此，在浏览器的偏好设置中许多人倾向于关掉下划线。这样做的用户不应过分被关注，但他们确实提供了一种额外的不应显著改变颜色的原因。但是，对 CSS，现在可以用text-decorarion属性关掉文本链接装饰，如下面的规则所示：

```
A   {text-decoration: none}
```

当然，这使得用户决定文本链接更加困难。另一种反馈形式应加到文本链接中，比如改变其字体类属、尺寸、风格或背景颜色。例如，当关掉下划线时，链接能用大一点的文本来表示，如粗体、斜体、以及变化的文本样式如小写、改变背景颜色、改变使用的字体类属。如第 10章的讨论，当使用文本时，应建立一个清晰的类型层次，在字体的尺寸、形式、类属中为用户提供充分的差异，并明显地区分。当这个变化较小时，文本链接看上去和正常文本太相似，这会困扰用户。表6-4反映了当下划线被关掉并用格式规则来实现它们时区别链接的方法。

建议：避免自动关掉下划线。如果这样做了，就加入另一种链接表示形式。

3. 链接反馈：光标

通常，浏览器通过光标变化来显示什么是一个链接或链接是可按下的。在大多数的 GUI系统中，典型的光标是，当可点击时，一个箭头或指针变成一个手，当可键入时，变为一个 I梁。CSS2和CSS3通过使用cursor属性，来引入修改光标的能力。

表6-4 使用CSS表示其他形式的链接

链接表示	CSS 规则
大的文本	A{text-decoration:none;font-size;larger;}
黑体字	A{text-decoration:none;font-weight; bold;}
斜体字	A{text-decoration:none;font-style;italic;}
小写	A{text-decoration:none;font-variant;small-caps;}
背景色	A{text-decoration:none;background-color;yellow;}
不同的字体	A{text-decoration:none;font-family;cursive;}

例如，当用户在任何＜B＞标志上移动时，放置一个光标使其看上去是可按下的，使用如下样式：

```
<B STYLE="cursor: hand">Can you press me?</B>
```

CSS2指定了许多形式的光标，如表 6-5所示。

CSS3规则建议了许多新的光标特性值，如表 6-6所示。

注意 对光标值实行典型的着色是不可能的，因为在写本书之际，还没有浏览器支持这些值。

4. 自定义光标

在理论上，CSS2定义了自定义光标的能力。在Windows系统中，光标用一个.cur文件来定义，那是一个32×32或更小一点的位图——一般为16色。光标偶尔也被激活，在这种情形下也许有.ani形式的扩展名。按照CSS2的规则，一个浏览器应从一个特殊的URL获取一个光标文件——类似于获取一个字体。这个特性包括一系列用逗号分开的光标值，因此当用户的鼠标掠过 ID属性设置为specialcursor的元素时，CSS2规则：

```
#specialcursor   {cursor: url("robot.cur"),
                  url("robot.csr"), default;}
```

将把光标指定设置为robot.cur，robot.csr或缺省样式。

注意 在写本书的时候，没有浏览器支持可下载的光标。但是，Comet系统（www.cometsystems.com）支持提供类似功能的技术，既支持 active控制，也支持Netscape插入件，包括对动画光标的支持。

Javascript中的光标跟踪

尽管在许多浏览器中设置光标的形状是不可能的。但用 Javascript和样式单附加一个小的GIF或JPEG图像，并使其在屏幕上跟随光标则是很容易的。不同的效果，包括在字面上追踪光标激活的图像，是可能的。这里给出的 Javascript展示了一个如何在因特网浏览器 4.X或更高版本中制作自定义光标的例子：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>JavaScript Cursor Demo</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function MainMouseEvent(e) {
   customcursor.left=event.clientX + document.body.scrollLeft;
   customcursor.top=event.clientY + document.body.scrollTop; }

function Initialize() {
   customcursor=document.all.cursorlayer.style;
   document.onmousemove = MainMouseEvent; }

var IE = (document.all) ? true:false;  // make sure IE
if (IE) {
  document.writeln('<DIV ID="cursorlayer"
  STYLE="position:absolute; ');
  document.writeln('top:-32; left:-32; width:32; height:32">');
  document.writeln('<IMG NAME="cursorimg" SRC="yourcursor.gif"');
  document.writeln(' border=0>');
  document.writeln('</DIV>');

  window.onload = Initialize;}
```

```
//-->
</SCRIPT>

<BR><BR>Insert your page content here.<BR><BR>

</BODY>
</HTML>
```

注意 Weblink：代码可以在http://www.webdesignref.com/chapter6/customcusor.html在线获得。

为使用代码，简单地创建一个 32像素乘32像素的光标，以 GIF图像存储，并且取代“yourcursor.gif”的文本。当在屏幕上移动时，图像会跟踪光标。

尽管创建这种光标的与浏览器版本无关的代码是可能的，但 Netscape 4.X版本浏览器错误太多，甚至用正确的书写代码来实现自定义光标也会引起浏览器锁定。这不是天意，这些代码在Netscape浏览器下，品质会下降，而无法显示自定义的光标。

表6-5 CSS2 光标和典型的显示

CSS 光标属性值	描 述	典型显示
自动	浏览器根据当前的上下文确定光标的显示	N/A
十字丝	简单的十字丝通常模仿一个加号	
缺省	浏监器的缺省光标通常是个箭头	
移动	这显示某物是可以移动的；通常显示为四个在一块的箭头	
向东调整	这会显示指向东的箭头（指向右边）	
向东北调整	这会显示指向东北的箭头（指向顶部右侧）	
向西北调整	这会显示指向的箭头（指向顶部左侧）	
向北调整	北（向上）	
向东南调整	东南（右下方）	
向西南调整	西南（左下侧）	
向南调整	南（下）	
向西调整	西（左）	
文本	这显示文本可以选择或输入；通常显示为一个工条	
等待	这显示页正忙；通常显示为一个手表或沙漏	
帮助	这显示可以获得帮助；光标通常显示为问号或气球	

表6-6 CSS3光标属性

CSS 3 光标属性	含 义
拷贝	显示可以拷贝。可以显示为与加号相连的箭头
别名	显示某事物的快捷方式或别名，经常显示为与小的曲线箭头相连的箭头
上下文菜单	该光标显示一个上下文菜单，通常与某个对象可以获得的辅助鼠标按钮一起被选择。经常显示为与小的菜单图形相连的箭头

（续）

CSS 3 光标属性	含　义
单元	显示单元或一系列单元可以被选择，应该被显示为一个很宽的加号
抓取	显示某个对象可以被抓取。应该被显示为一个握紧的手
旋转	显示程序正在完成某个任务。与 wait属性相类似，但用户仍能够与程序交互。有很多显示方式，包括旋转的皮球
正向计数	显示程序在正向计数，可以显示为用手指计数
倒计数	显示程序在倒计数，像正向计数一样，可以显示为一个手指
正反向混合计数	显示程序在正向与反向之间交替计数

要一直记住，除非能增加效应，否则不要修改光标。微小的改动是很好的；一般的改变是使用样式规则，把手指指针改变为反方向的箭头，如以下的短例子所示：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Subtle Cursor Change</TITLE>
<STYLE TYPE="text/css">
<!--
A:hover {cursor:ne-resize;}
-->
</STYLE>
</HEAD>
<BODY>
<A HREF="http://www.yahoo.com">Yahoo</A>
</BODY>
</HTML>
```

但是要明白，像改变链接的颜色一样，改变光标也许使一些用户对什么是链接、什么不是链接感到困惑。

5. 链接和省略号

在图形界面中，当用户选择了一个特殊的命令时，省略号（...）被用来表示还有很多事情要发生——特别是需要更多的输入。但是，这个思想没有很好地传递给 Web。考虑到几乎所有其的链接都有其随后的事情，用户期望是这样。可能应用省略号的唯一机会是，一个页只是简单地打开另一个包含大量选择和少量内容的页时。现在，没有站点使用省略号，除了偶尔使用难题精选引出更多的信息时，如下所示：

Gravity Defeated

Today the DemoCorporation announced that gravity has been defeated.

Dr. R. Smart boasted that his anti-Newton drive would turn the

Personal space craft industry on its head. [More...]

规则可以改变，但应当避免使用省略号。

建议 避免在链接中使用省略号，因为它们通常是多余的。

6.3.2 滚动

一个普通的链接反馈机制称为滚动。滚动链接是流行的链接方式，当用户点击它时，通常会发生颜色或形状的改变。当滚动的作用在Web中变得如此平常时，它们已经成了陈词滥调，它们可以为用户提供更多的反馈，也可作为页面的片段，并且在制作良好的情况下可提供更多的关于链接目标的信息。

制作滚动链接的最简单方法是用CSS2在<STYLE>标签中，使用A:hover伪类激活一个文本链接，如下所示：

```
<STYLE>
<!--
A:hover {color: #FF0000}
-->
</STYLE>
```

在这个例子中，当用户将他们的鼠标放在链接的文本之上时，文本将变成红色的。使用样式单规则，可以改变链接来显示变化的多样性，比如文本的尺寸或风格。设计者应该小心避免滚动效果的剧烈变化，当用户滚动这些链接时，浏览器将不得不以一种明显的方式刷新页面。

也可以用JavaScript为图形按钮创建基本的文本滚动。这种风格的滚动基本上这样工作。创建一个规则的图形按钮，然后创建这个按钮的相同尺寸的激活后的版本。下一步，加一段JavaScript，用户的鼠标经过时，用它的激活后的图像代替正常的图像按钮，当用户按过按钮后，将其恢复到它的正常状态下的按钮。JavaScript写起来相对较简单，且仅要求先装载图像并支持确定的图像对象。以下的代码演示了基本的滚动：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
<HTML>
<HEAD>
<TITLE>Simple Rollover</TITLE>
<SCRIPT LANGUAGE ="JavaScript">
<!--
/* Preload the images */

  if (document.images)          {
    abouton = new Image(85, 48);
    abouton.src = "images/abouton.gif";
    aboutoff = new Image(85, 48);
    aboutoff.src = "images/about.gif";
  }

function On(imgName) {
        if (document.images) {
```

```
        imgOn = eval(imgName + "on.src");
        document [imgName].src = imgOn;
        }
}

function Off(imgName) {
        if (document.images) {
        imgOff = eval(imgName + "off.src");
        document [imgName].src = imgOff;
        }
}
// -->
</SCRIPT>
</HEAD>
<BODY>

<A HREF="About/index.htm"
   onMouseover="On('about')"
   onMouseout="Off('about')">
<IMG SRC="images/about.gif"
    WIDTH="85" HEIGHT="48" BORDER="0" ALT="About" NAME="about"> </A>
<BR>

</BODY>
</HTML>
```

提示 Weblink:代码可以在http://www.webdesignref.com/chapter6/rollover.htm上在线获得。

要使用代码，简单地加入适当 HEIGHT和WIDTH属性的一个新的 <IMG>标签。一定要为<IMG>元素命名，然后加入预装载代码去装载当前状态下的图形。要为 Products按钮装载另一个按钮，可在JavaScript的预装载部分加入以下代码：

```
producton = new Image(85, 48);
producton.src = "images/producton.gif";
productoff = new Image(85, 48);
productoff.src = "images/product.gif";
```

在<BODY>内，如下所示把另一个链接加入到图像：

```
<A HREF="products/index.htm"
   onMouseover="On('products')"
   onMouseout="Off('products')">
<IMG SRC="images/product.gif"
     WIDTH="85" HEIGHT="48" BORDER="0" ALT="Products"
NAME="products"></A><BR>
```

这段程序适用于Netscape 3.0版本和更高版本，以及IE浏览器4.0版本或更高版本。使用脚本的唯

一问题是要确保为图像正确命名。如果你对把这种程序手工加入到文档中不感兴趣，许多Web编辑器，包括macromedia Dreamweaver，都支持加入这种代码，那只是运行一个如“插入滚动图像”的命令,并选择合适的图形状态。图6-14显示了使用和前一个例子相同的数据时Dreamweaver的对话框。

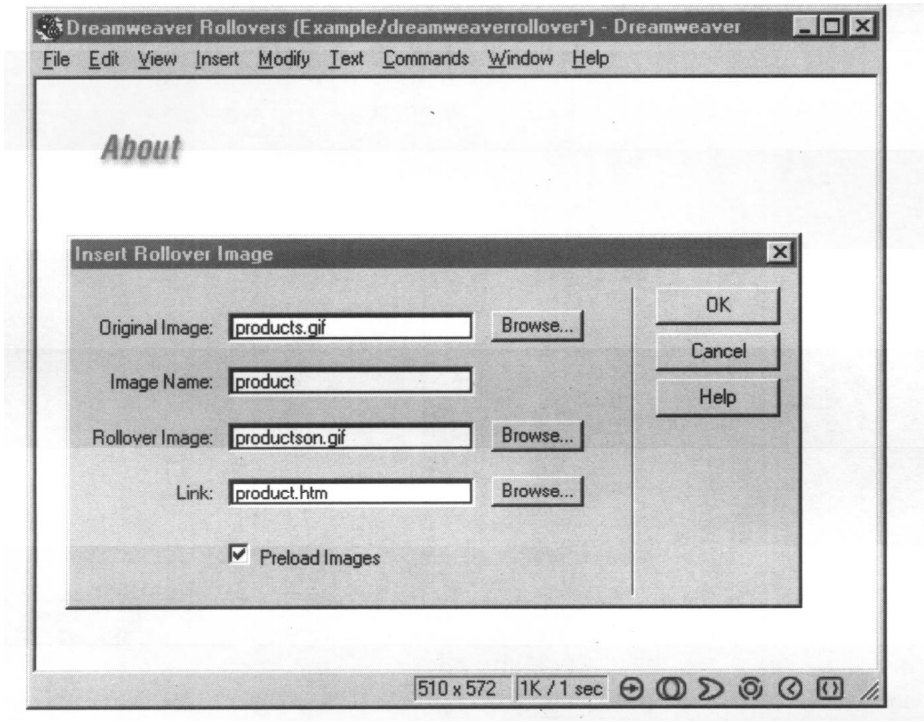


图6-14 DreamWeaver滚动对话框

1. 在Fireworks中创建滚动

使用Fireworks甚至可以创建各种各样的滚动图像，只须遵循下列步骤：

1) 打开文档，拉出Insert菜单并选择New Button。

2) 现在制作一个简单的图形按钮，类似于先前本章的“在 Fireworks中制作按钮”一节讨论的制作过程。一旦完成了，对话框也许看起来像图 6-15 所示。

3) 一旦完成了按钮的制作，在当前对话框中选择标有 Dver的标签。这将允许你指定当一个鼠标经过按钮时可以应用的图像。

4) 按下标有Copy Up Graphic的按钮。这将在窗口中得到正常按钮的一个复制品。

5) 为结束状态修改按钮的颜色和风格，将有一个看起来和图 6-16 所示类似的对话框。

6) 在对话框的右上角选择Close框,关闭New Button对话框。

7) 返回主窗口，选择新按钮，你现在应该停在 Object调色板上，如果没有，请从 Window菜单中访问Object调色板。

8) 从Object调色板，选择标有Link Wizard的按钮。

9) 在Link Wizard内，选择标有Link的Link Wixard，指定按钮应链接的URL和任意的ALT文本。按下OK, 完成操作。

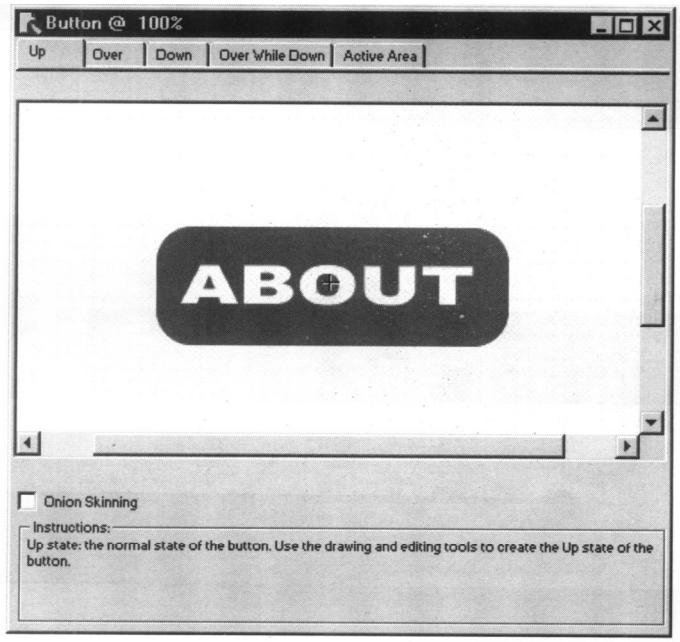


图6-15 Button 对话框

10) 从File菜单中通过选择浏览器中的Preview，观察滚动效果。

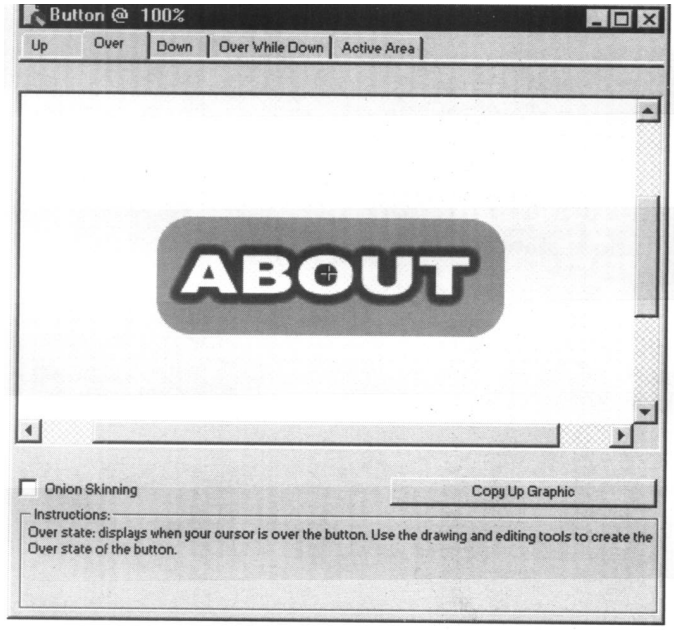


图6-16 Over 选项卡

一旦掌握了在Fireworks中制作一个单独的滚动，就很容易复制这个滚动，改变它的文本，应用“链接向导”重新设置链接。一旦完成了，就可以输出整个布局——包括 HTML和

javascript—— 拥有一个充满无数滚动的页。

图形滚动按钮的一个主要不足是，对每一个滚动按钮它们需要两个或两个以上的分开图像。因此，假设一个带有8个按钮的导航条，一共需要装载16个图像。考虑与服务器的多重链接，滚动当然引起较大的下载延迟。幸运的是，随着样式单的兴起，有可能应用 hover属性和定位来制作轻便的滚动。

2. 第二代图形滚动

使用CSS定位，有可能仅使用两幅图像创建一个图形滚动按钮菜单。下面创建开启状态下所有按钮的大图像，以及所有按钮关闭状态下的大图像，如图 6-17所示。

图6-17 不同状态下的大图像

在这个例子中，将它们命名为 all.gif和allon.gif。下一步，为图像创建一个图像映射。现在可以使用CSS定位属性，把两幅图像放在各自文档的上部 —— 除了allon.gif将会隐藏。下面将写一段代码以便用户经过图形时， allon.gif的一部分会显示出来。关键是应用剪切路径去剪下想显示的区域。因为在图像的顶部使用了图像映射，我们已经有了剪切路径。下面的代码和标记说明了这个思想。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Rollovers Generation 2</TITLE>
<STYLE TYPE="text/css">
<!--
 #menu {position: relative }
 #menuoff {position: absolute;
           top: 0; left: 0; }
 #menuon {position: absolute;
          top: 0; left: 0;
          visibility: hidden;}
-->
</STYLE>
```

```
<SCRIPT>
<!--
IE4 = (document.all) ? 1 : 0;
NS4 = (document.layers) ? 1 : 0;
ver4 = (IE4 || NS4) ? 1 : 0;


function clipRegion(left,top,right,bottom)
 {
   this.left = left;
   this.top = top;
   this.right = right;
   this.bottom = bottom;
}


if (ver4) {
     /* create clipping regions */

               var cliparray = new Array();

    cliparray[0] = new clipRegion(5,138,83,164);
    cliparray[1] = new clipRegion(4,107,83,133);
    cliparray[2] = new clipRegion(3,77,83,102);
    cliparray[3] = new clipRegion(2,49,83,73);
    cliparray[4] = new clipRegion(23,19,84,43);

    }


function rollover(region,turnon)
 {
   if (!ver4)
     return;

   if (IE4)
     currentbutton = document.all.menuon.style;
   else
     currentbutton = document.menu.document.menuon;

   if (!turnon)
     {
  currentbutton.visibility = "hidden";
  return;
 }


if (NS4)
```

```
        {
                currentbutton.clip.top = cliparray[region].top;
                currentbutton.clip.right = cliparray[region].right;
                currentbutton.clip.bottom = cliparray[region].bottom;
                currentbutton.clip.left = cliparray[region].left;
            }
    else
      currentbutton.clip = "rect(" + cliparray[region].top + " " +
cliparray[region].right + " " + cliparray[region].bottom + " " +
cliparray[region].left + ")";
      currentbutton.visibility = "visible"
     }

//-->
</SCRIPT>
</HEAD>

<BODY BGCOLOR="#FFFFFF">

<DIV ID="menu">
 <DIV ID="menuoff">
  <IMG SRC="images/all.gif" WIDTH="85" HEIGHT="168"
       BORDER="0" USEMAP="#buttons">
</DIV>
<DIV ID="menuon">
   <SCRIPT>
   <!--
    document.write('<img src="images/allon.gif" width="85"
              height="168" border="0" usemap="#buttons">');
   //-->
   </SCRIPT>
</DIV>
</DIV>

<MAP NAME="buttons">

  <AREA SHAPE="rect" COORDS="5,138,83,164" HREF="intranet.htm"
ALT="Intranet"
 onMouseover="rollover(0,true)" onMouseout="rollover(0,false)">
<AREA SHAPE="rect" COORDS="4,107,83,133" HREF="jobs.htm" ALT="Jobs"
onMouseover="rollover(1,true)" onMouseout="rollover(1,false)">
<AREA SHAPE="rect" COORDS="3,77,83,102" HREF="news.htm" ALT="News"
onMouseover="rollover(2,true)" onMouseout="rollover(2,false)">
<AREA SHAPE="rect" COORDS="2,49,83,73" HREF="products.htm"
        ALT="Products"
  onMouseover="rollover(3,true)" onMouseout="rollover(3,false)">
  <AREA SHAPE="rect" COORDS="23,19,84,43" HREF="about.htm"
        ALT="About"
```

```
  onMouseover="rollover(4,true)" onMouseout="rollover(4,false)">
</MAP>
</BODY>
</HTML>
```

提示 Weblink:在http://www.Webdesignref.com/chapter6/rollover2.htm上可以在线获得代码。

为站点调整这段代码是相对容易的。首先，制作这两幅图像；然后，建立图像映射。下一步，用<DIV>元素加上定位和提供的风格属性。如果这时浏览这个页面，你将仅看到图形的关闭状态。现在，加入javascript，仅有的变化是设置了不同的剪切区域，代码如下：

```
cliparray[0] = new clipRegion(5,138,83,164);
    cliparray[1] = new clipRegion(4,107,83,133);
    cliparray[2] = new clipRegion(3,77,83,102);
    cliparray[3] = new clipRegion(2,49,83,73);
    cliparray[4] = new clipRegion(23,19,84,43);
```

只要改变右侧的坐标与图像映射的坐标匹配，并且加入更多的 cliparray[]项，如下所示：

```
cliparray[5] = new clipRegion(28,19,84,20);
```

现在，在图像映射中，只要改变 <AREA>元素就可以了。

```
onMouseover="rollover(5,true)" onMouseout="rollover(5,false)">
```

这段代码的唯一不足是仅用于4.X及更好的浏览器。但是，它会在老的浏览器中优雅地降级，只是看不到滚动效果。

注意 在写这本书时，还没有WYSIWYG编辑器支持第二代滚动代码的随意产生。希望在不久的将来情况会改变。

3. 仅使用CSS的滚动

当使用CSS风格的按钮时，依然能够创建滚动效果。 A:hover伪类也许对创建简单的滚动如改变颜色是有用的，如下面的格式规则所示：

```
A       {color: blue;}
A:hover {color: red;}
```

另一种普通的样式是仅当用户的鼠标经过时加下划线，这可以用两个基本的 CSS规则完成：

```
A       {text-decoration: none;}
A:hover {text-decoration: underline;}
```

但是，对于复杂的CSS按钮，这个A:hover伪类也许并不够。使用 Javascript，有可能动态地修改区域的风格。这里的短例子展示了在 CSS中如何用javaScript制作简单的滚动。注意，这段程序仅在IE浏览器中有效。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 TranSitional//EN"
<HTML>
```

```
<HEAD>
<TITLE>CSS Rollover Buttons</TITLE>
<STYLE>
<!--
 #mybutton    {border-style: inset;
              border-color: #ff6633;
              background-color: #CC3300;
              width: 80px;
              text-align: center;}
  A.buttontext {color: white;
text-decoration: none;
      font: bold 12pt Verdana;
      cursor: hand;}
  .buttonover  {color: yellow;
      text-decoration: underline;
      font: bold 12pt Verdana;
      cursor: hand;}
-->
</STYLE>
</HEAD>
<BODY>
<A HREF="about.htm" CLASS="buttontext"
onMouseover="this.className='buttonover';
      mybutton.style.background='#FF6633'"
onMouseout="this.className='buttontext;
      mybutton.style.background='#CC3300'">
<DIV ID="mybutton">
About
</DIV></A>
</BODY>
</HTML>
```

提示 Weblink:代码可以在http://www.Webdesignref.com/chapter6/cssrollover.htm上在线获得。

4. 使用滚动

不管形式如何，设计者应该力图正确使用滚动。所犯的最通常的错误是不考虑一个滚动按钮应该包含多少个状态。第一个状态是正常状态或未选状态。一些人称之为“悬挂”状态。这种状态下的按钮还没有被按下，但能被按下。第二个状态是选过状态。在这种状态下，按钮不能被按下，因为它不适用于当前内容或目前已被选中。一般来讲，选过状态下的按钮变成灰色或外表上被改变。第三个状态是经过状态，在术语上称为激活状态。这种状态是当用户在按钮上经过，还没有决定选择它时的状态。第四个可能的状态是按下状态或鼠标点击状态。当用户实际按下了按钮时这个状态才发生。关于一个按钮的所有状态的例子显示在图 6-18中。

注意到这些状态和链接状态是类似的—未访问、激活、访问、新状态和盘旋。尽管这样，许多站点的按钮缺乏所有的状态。这样做的主要原因是对于图形按钮，多一个状态会有更多的

图像被下载。由此考虑下面的设计建议。

建议：在未选中和选中状态下图形按钮应该最小。经过和按下状态可考虑为选择项。

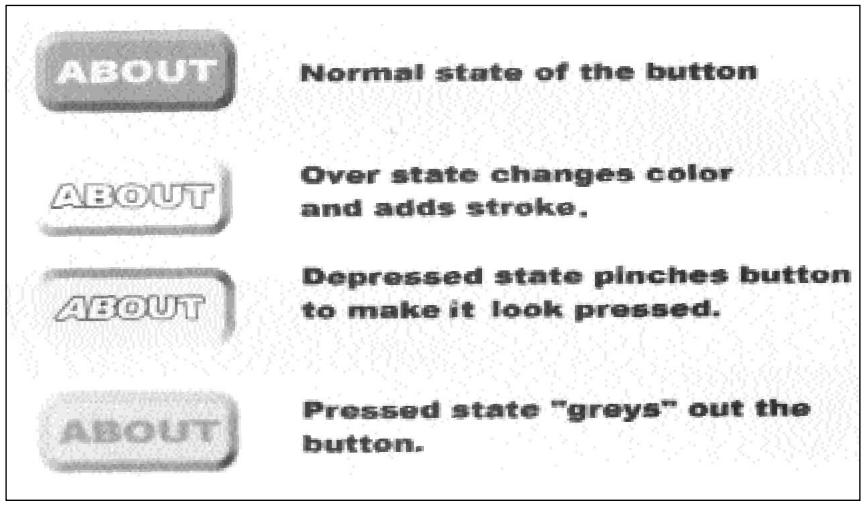


图6-18 滚动状态

当使用基于CSS的按钮或带宽足够不再考虑下载时，一般包括按钮的所有状态。

当使用正确时，滚动是有用的，因为它们进一步让用户知道对象是活动的。多数情况下，滚动是有某些技巧的——使文本发光或按钮改变形状。但是，在某些情形下，滚动通过显示描述性文本使用户知道将要装载的内容。当使用这种方式时，滚动的效果确实使按钮更加实用。

6.3.3 理解用户的期望

对用户来说，在Web站点的使用中最烦人的方面是选择了一个页面但没有得到期望的信息，这甚至超过了页面的缓慢装载。从用户的观点来看，每个链接代表一个门，链接标签用来指示每个门内有什么。当用户不确定链接后面有什么时，他们力图尝试这个按钮——如果那里没有他们所寻找的东西，他们就会返回。乍看起来这也许很有意思——就像探险——但过一会就会变得很沮丧。

站点的设计者应始终力图让用户在点击了一个链接的时候，就知道他们将看到什么。面对一个链接，用户也许会问下列问题，下面给出了一些如何处理这些问题的建议。

- 这些链接将装载什么样的内容？

建议：提供表明内容形式的好的标签。考虑用图标显示内容的类型。

Datasheet – robot235.pdf (Adobe Acrobat Format)

Robot Datasheet

• 链接将带他们到哪里去？

建议：确切地显示链接是否带他们在一个页面内跳转，在一个站点内跳转或者跳到一个外部的站点内。不要隐藏URL，以免用户自己推论。

使用上下箭头表示页面内的跳转。

^ Back to top

v Spec Sheet

用下面的图标表示一个外部链接：

DemoCompany Partners (outside link)

DemoCompany

建议：通过显示URL或用一个图标表示一个外部链接。如果启动一个下载，要标明文件尺寸。

把其余的链接单独放置，这样用户就会假定它们是正常的内部站点链接。

About Demo Company

• 它意味着一个长下载吗？

Specification.pdf (854K)

• 它费钱吗？

建议：用一个图标或记号，或在链接之前弹出一个警告对话框。

$ sign-up today $

Specification **(payment required)**

• 链接的内容新鲜吗？

建议：使用一个新图标或在需要的地方加入最新修改的日期。

Printer drivers (1/5/99)

NEW Specifications

• 内容有潜在的攻击性吗？

建议：如果内容有潜在的攻击性，应使用警告或用明显的标签警告。

Red light district (21 and up only)

绝大多数问题的关键是正确地表明一个链接。然而当链接本身不得已很短时 —— 使用TITLE属性 —— 把一个滚动或额外的范围注释信息提供给用户。状态条也给出了一个通知用户链接目标的地点。后面将依次讨论这些方法。

6.4 使用范围注释

除了好的标签，让用户知道链接含义的最好方法是使用范围注释。范围注释提供了链接的含义，以及其他文本信息的描述。考虑一个链接标签如 About。我们能够加上范围注释来解释标签的含义。如下所示：

```
About
Information about DemoCompany including corporate history,
press releases, and self congratulating biographies.
```

确保把范围标签的字体设置得小一些，或用不同的风格，以免盖过基本的链接。

通过提供嵌在描述文本中的向前跳跃链接，范围注释可能提供如下的好处：

```
About

  Information about DemoCompany including corporate history,
  press releases, and self congratulating biographies.
```

范围注释唯一的主要不足是它们也许会打乱布局或使注意力偏离一个页面的重要项。从某种意义上说，范围注释像帮助信息。对那些寻找更多信息的人来说，它们真的是最有用处的。由于这些潜在的缺点，许多设计者决定隐藏范围注释，只有用户经过或激活按钮时才显示出来。

一般来说，应该避免把向前跳跃链接放入到会显示出来的范围注释中。当用户移动以点击新出现的链接时，要确保他们的鼠标不使范围注释项消失，这是十分恼人的事。因此，如果在滚动中包括向前跳跃，一定要确保一旦鼠标经过它时，范围注释要出现。

6.4.1 TITLE属性

显示范围注释的最简单的形式是用 TITLE属性提供的工具提示信息。对于想要的文本链接，设置TITLE属性如下所示：

```
<A HREF="about.htm" TITLE="Information about DemoCompany including
corporate history, press releases, and self congratulating
biographies.">About</A>
```

当用户用鼠标经过链接时，将出现附加的文本信息。链接标题应该提供关于链接将做些什么的更多信息，但不要显得太冗长而被忽略。力图把链接标题做得简短，易于扫描 —— 或许10~15个字，或最大60~80个字母。

注意 对长的链接标题要小心，当一些老的浏览器不能转换标题信息时，范围信息可能被删除。

当使用图形文本按钮时，存在一些问题如ALT或TITLE属性文本是否会出现。你将会发现结果会随浏览器而变化。如果有必要，考虑把ALT和TITLE属性做成一样的。

6.4.2 滚动信息

当用户的鼠标经过一个链接时，用滚动在屏幕的其他某些地方显示文本或图形是可能的。可以编写一段程序来显示一个范围注释和改动链接的状态。以下的标记和JavaScript显示使用图形时它是如何起作用的。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Targeted Rollovers</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--

if (document.images)
  {
      abouton = new Image(147, 29);
      abouton.src = "abouton.gif"
      aboutoff = new Image(147, 29);
      aboutoff.src = "about.gif"

      blank = new Image(130, 127);
      blank.src = "blank.gif"

      description1 = new Image(130, 127);
      description1.src = "description.gif"
  }

function On(imgName,description) {
        if (document.images) {
        imgOn = eval(imgName + "on.src");
        document.images[imgName].src = imgOn;
        document.images.descriptionregion.src= description.src;
        }
}

function Off(imgName) {
if (document.images) {
        imgOff = eval(imgName + "off.src");
        document.images[imgName].src = imgOff;
        document.images.descriptionregion.src= "blank.gif";
        }
```

```
}

// -->
</SCRIPT>
</HEAD>

<BODY>

<A HREF="about.htm"
onMouseover="On('about',description1);
window.status='Company'; return true"
onMouseout="Off('about');">
<IMG SRC="about.gif" BORDER="0" ALT="About" NAME="about"
WIDTH="159" HEIGHT="57"></A>
   
<A HREF="">
<IMG SRC="blank.gif" NAME="descriptionregion" WIDTH="328"
HEIGHT="84" BORDER="0" ALT=""></A>

</BODY>
</HTML>
```

提示 Weblink: 在http://www.webdesignref.com/chapter 6/targetedrollovers.htm上可在线获得代码。

图6-9显示了运行中的滚动代码。

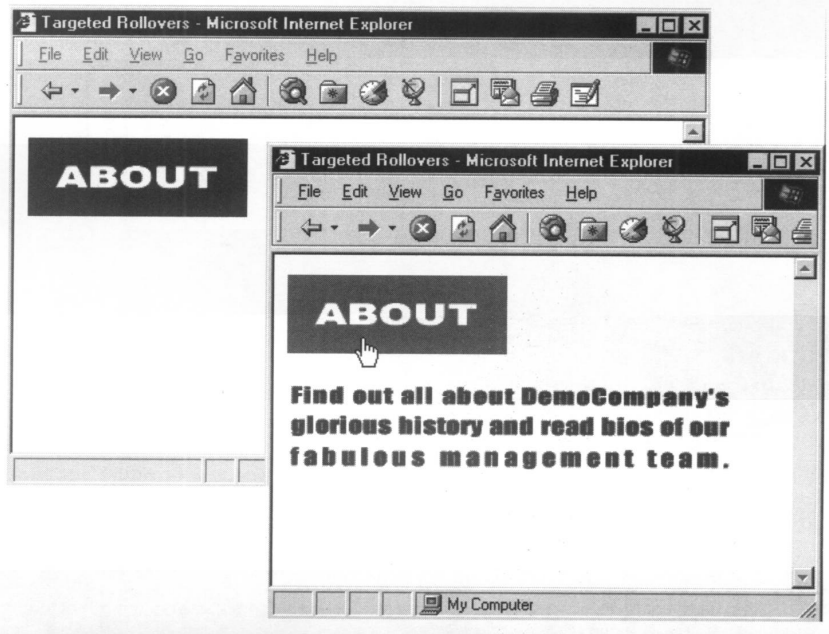


图6-9 滚动可以显示描述性的文本

一般来讲，设计者被鼓励使用显示附加信息的滚动，但一定要记住，像多状态滚动一样，滚动信息是很讨厌的，因为它们需要很多图像。幸运的是，使用样式单，可以建立轻便的滚动信息。下面的代码显示了这一点是如何实现的：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>CSS Target Rollovers</TITLE>
<STYLE>
<!--
#buttons   {position: absolute;
           top: 10px;
           background-color: yellow;
           width: 20%;}
#blankdescription {position: absolute;
           top: 10px;
           left: 40%;
           background-color: orange;}

#aboutdescription {position: absolute;
           top: 10px;
           left: 40%;
           background-color: orange;
           visibility: hidden;}

#productsdescription {position: absolute;
           top: 10px;
           left: 40%;
           background-color: orange;
           visibility: hidden;}
-->
</STYLE>
<SCRIPT>
<!--
var IE4 = (document.all) ? true : false;
var NS4 = (document.layers) ? true : false;

function changeVisibility(id, showflag)
 {
   if (NS4)   {
      var str = (showflag) ? 'show' : 'hide';
      eval("document." + id).visibility = str;
    }
    else   {
       var str = (showflag) ? 'visible' : 'hidden';
       eval("document.all." + id).style.visibility = str;
     }
```

```
}

//-->
</SCRIPT>
</HEAD>

<BODY>

<DIV ID="buttons">
<A HREF="about.htm"
onMouseover="changeVisibility('blankdescription', false);
             changeVisibility('aboutdescription',true)"
onMouseout="changeVisibility('blankdescription', true);
            changeVisibility('aboutdescription',false)">About</A>
<BR>
<A HREF="products.htm"
onMouseover="changeVisibility('blankdescription', false);
             changeVisibility('productsdescription',true)"
onMouseout="changeVisibility('blankdescription', true);
       changeVisibility('productsdescription',false)">Products</A>

</DIV>

<!-- Description text follows -->
<DIV ID="blankdescription">

</DIV>

<DIV ID="aboutdescription">
Discover the history and management behind the DemoCompany.
</DIV>

<DIV ID="productsdescription">
If you like our domes, you'll love our robots!
</DIV>
</BODY>
</HTML>
```

提示 Weblink:在http://www.webdesignref.com/chapter6/csstargetedrollovers.htm上可以在线获得代码。

注意 这段代码只在4.X版本的浏览器下有效，因为它不仅依赖JavaScript，也依赖于样式单。

6.4.3 状态条信息

在浏览器窗口底部的状态条上显示链接的结果是有可能的。一般来讲，一个浏览器将在状

态条上显示目标URL，但可以应用简短的JavaScript来定制。在JavaScript的所有版本中，可以利用窗口对象的状态特性，它定义了整个浏览器窗口。当用户鼠标经过链接时，使用核心 HTML 4.0的onMouseover属性，可以触发一个信息。实现这些的程序如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Status Bar Link Messages</TITLE>
</HEAD>
<BODY>
<H2>Status Bar Link Messages</H2>
<A HREF="http://www.democompany.com/"
   onMouseover="window.status='Visit DemoCompany home of the
 Robot Butler! '; return true;"
   onMouseout="window.status='';return true;">Demo Company</A><BR>

<A HREF="http://www.yahoo.com/"
   onMouseover="window.status='Have you been to Yahoo! today?';
return true;"
   onMouseout="window.status='';return true;">Yahoo!</A><BR>
</BODY>
</HTML>
```

提示 Weblink:要参阅代码，请在线访问http://www.Webdesignref.com/chapter6 /statusbar .htm。

注意 onMouseover代码必须返回true值，否则不会显示信息。

在前一个例子中，只需改变显示文本引用的字符串。

考虑为用户在状态条上提供信息的潜在的不足。首先，用户也许不看这个位置。其次，如果用户不看这里，为了确定链接的目标，他们期望 URL信息。更经常的是，这里显示的状态信息重复基本的文本链接信息。对于在点击链接之前想知道 URL外部链接信息的用户是特别麻烦的，对其他内容形式的链接也一样。对于外部链接，考虑使用如下所示的状态信息格式：

```
<A HREF="http://www.yahoo.com/"
   onMouseover="window.status='Have you been to Yahoo! (www.yahoo.com)
today? '; return true;"
   onMouseout="window.status='';return true;">Yahoo!</A>
```

当链接到另一种内容形式时，状态条也许用于让用户知道内容的格式，如下面的例子所示：

```
<A HREF="robotdatasheet.pdf"
   onMouseover="window.status='Adobe Acrobat Format'; return true;"
   onMouseout="window.status='';return true;">Robot Butler Datasheet</A>
```

在这个例子中，一个图标甚至一个链接内容为 Acrobat的标签表示要比单独依赖状态条好。

建议：使用状态条信息，当外部链接时，考虑提供文本的 URL信息。

在许多方面，提供状态条信息是!!!嗦的。考虑到同样的信息被TITLE属性提供，如下所示：

```
<A HREF="http://www.yahoo.com/"
   TITLE="Have you been to Yahoo! (www.yahoo.com) today?">Yahoo!</A>
```

在这个特例中，“工具提示”甚至直接在用户鼠标所在处显示目标 URL，以及用户的状态信息。使用状态条信息的唯一真正的好处是它适用于老的支持 JavaScript的浏览器，而TITLE属性仅在HTML 4.0相应的浏览器中使用。

6.5 链接的键盘支持

设计者应该力图使站点对所有用户是有用的和可访问的。考虑到用户也许发现难于使用鼠标，而宁愿使用键盘。使用键盘命令也应容易激活链接。绝大多数浏览器支持链接的 Tab键，一些已经支持加速键。

HTML 4.0规则为<A>标记加入了 ACCESSKEY属性，还有各种窗体元素。使用这一属性，就可能设置一个键，激活一个锚定而不需选择链接的指针工具。通过联合使用加速键，通常是ALT键，由属性指定的键来激活链接。因此，制作一个到 Yahoo！的链接可以通过ALT-Y来激活。到目前为止，只有IE 4.X和以上版本支持这种链接访问的升级。

```
<A HREF="http://www.yahoo.com" ACCESSKEY="Y">Yahoo!</A>
```

当加入键盘访问到一个页面上时,将被认为是一个显著的改进。 HTML作者应注意，访问键是和浏览环境绑定在一起的。假定主要的浏览器支持 ACCESSKEY属性，作者应小心不要使用表6-7列出的加速键。

加速键的另一个问题是如何在页面上显示它们。一般在软件中，加速键的字母用下划线表示出来。当然，浏览器中链接一般加下划线。因此这种方法不可行。用样式单改变链接的方向是可能的，因此为第一个字母加下划线是可能的，但随后用户就可能被迷惑，因为他们期望链接是全部加下划线的。另一个显示加速键的方法是用粗体或稍大一点的尺寸设置一个文本链接的访问键。设计者被鼓励采用在 Web页面中已成为标准的标记。

使用<A>标签的 TABINDEX属性来定义支持键盘导航的浏览器中的 tab次序。典型的TABINDEX是个正值。随着TABINDEX值的增加，浏览器会逐一扫描每个链接，但会忽略那些负值。这样 <A HREF=“about.htm”TABINDEX=“1”>会设置第一个 Tab目标。如果TABINDEX没有定义，那么浏览器会根据文档中发现的次序依次 Tab每一个链接。

表6-7 保留的加速键

键	描 述
F	File（文件）菜单
E	Edit（编辑）菜单
C	Communicator（通信）菜单（Netscape仅有）

（续）

键	描 述
V	View（浏览）菜单
G	Go菜单
A	Favorites（偏好）菜单(IE仅有)
H	Help（帮助）

6.6 高级的Web链接模型

现在，Web显示了一种简单的链接模型；然而，将来也许会改变。 HTML引入了<LINK>元素，可以用来定义文本的链接关系。使用 <Link>的最常见方式是与样式单一起使用，如下例所示：

```
<LINK REL="STYLESHEET" HREF="corporate.css">
```

一个这样的<LINK>元素可以在HTML文档的<HEAD>中发现。

然而，用REL属性确定任意的关系是可能的。例如，使用 <LINK>可以定义哪一个文档下一步将被点击，像这样：

```
<LINK REL="NEXT" HREF="nextpage.htm">
```

一些浏览器如WebTV可能使用它来暗示预加载页面。所有的种类，如主页、版权、对拥有者的链接等等，都可能被使用。然而，除非浏览器支持更多的类型，否则简单的链接模型将继续被采用。

6.7 链接的维护

即便一个链接在站点中使用得很好，并且用户对每一个链接都理解得很好，链接仍需维护。一个方法是首先不让断开的链接进入站点。使用动态链接的站点能够避免断线链接，因为当加入页面时链接被动态调整。但是，绝大多数站点没有使用动态链接，因此不可避免地，经过一段时间后，内容变化了的链接就会断开。更一般的是，当其他的站点移动了页面，又没有考虑外部的链接时，对外部站点的链接将会断开。搜索出一个站点的断开链接可能是枯燥的，但这是唯一提前能做的。一个断开的链接是一个严重的问题。用户点击一个断开的链接，哪里也不能到达，最终得到最近较出名的“ 404没发现 ”的信息或类似的信息。如果一个应用软件的菜单上引出一个信息说“ 对不起，拼写检查没找到 ”，这样的疏忽在软件中是不能忍受的，在站点中也应该被认为是同一个层次的问题。

规则：断开链接被认为是灾难性的失败。

幸运的是，识别和修复断开链接并不是太困难，有如LinkBot或Coast WebMaster这样的工具，发现断开链接是一个简单的事情。但是，考虑到站点中有外部链接，甚至经常性的监视也不能始终排除站点中的断开链接，则考虑安装一个定制的 404页面来说明不可访问的链接。把站点映像或站点管理员联系的方法放在定制的错误页面中。定制的 404页的例子如图6-20。

注意 定制的404错误页的安装依赖于使用的服务器。

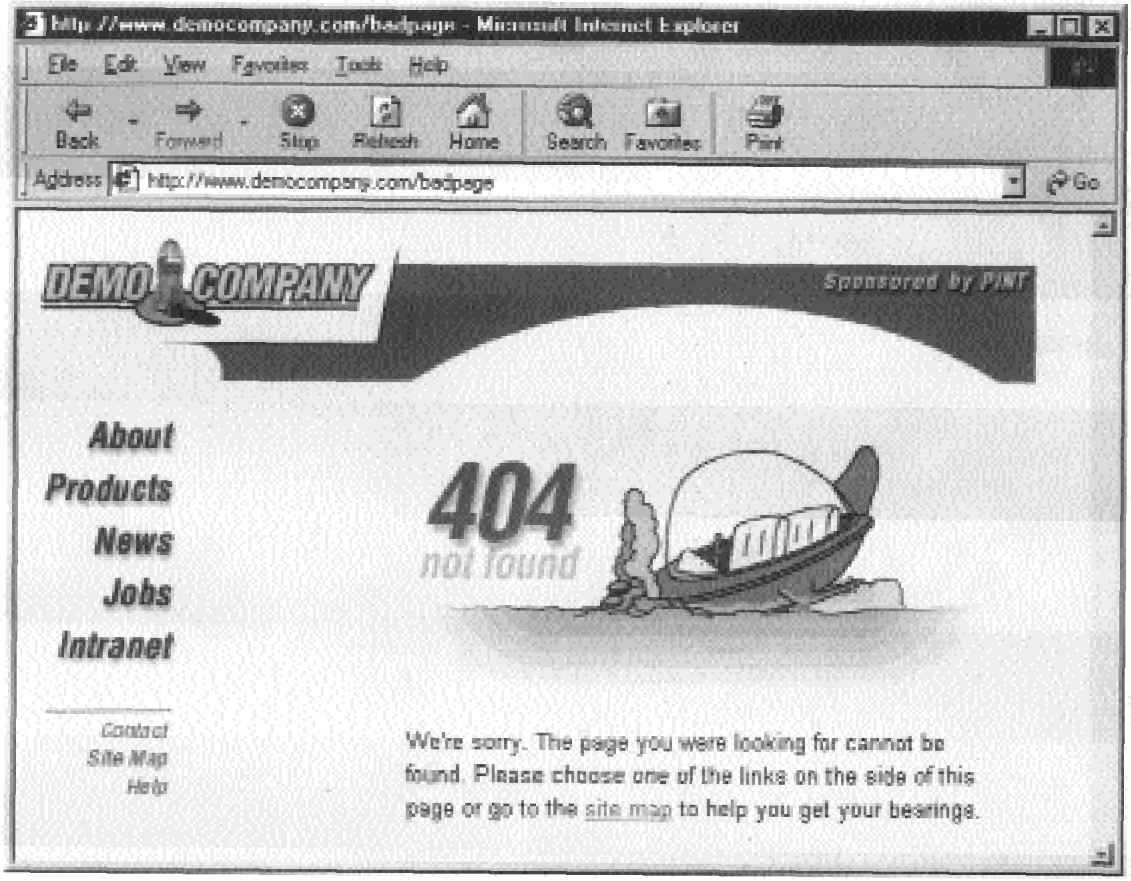


图6-20 定制的404页适合于站点设计

重定向页

即便显示错误，许多站点宁愿把用户重定向到一个新页。如果在一个 URL的内容如http://www.democompany.com/movedon.html已经转向一个新的位置，最好安装一个页面把人们指向新页或更快地重定向到这里。先前给定的 URL只能做到这些。某些站点的维护者宁愿把人们直接指向一个新页，尽管其他人通过安装一个临时页通知访问者页面的变化，如图 6-21所示的例子。

把人们直接指向一个新页或许是很连贯的。但确实减少了用户的控制。例如，如果用户期望打开一个特殊页，比如说机器人狗，一个重定向将他们指向不同的页，他们将变得很沮丧。应始终确保新页与移动页有联系。

如果用户请求一个已不存在的页，一些站点宁愿把用户直接指向站点的主页。因为这会使用户感到困惑，所以本书不推荐采用这种方法。错误是不可避免的，用户也会犯错误。一个设

计者的目标应该是降低困难，帮助用户少犯错误——但也不要让他们失去控制权。进一步说，对坏页面的迅速重定向要求将不鼓励站点的维护者解释错误背后的原因。

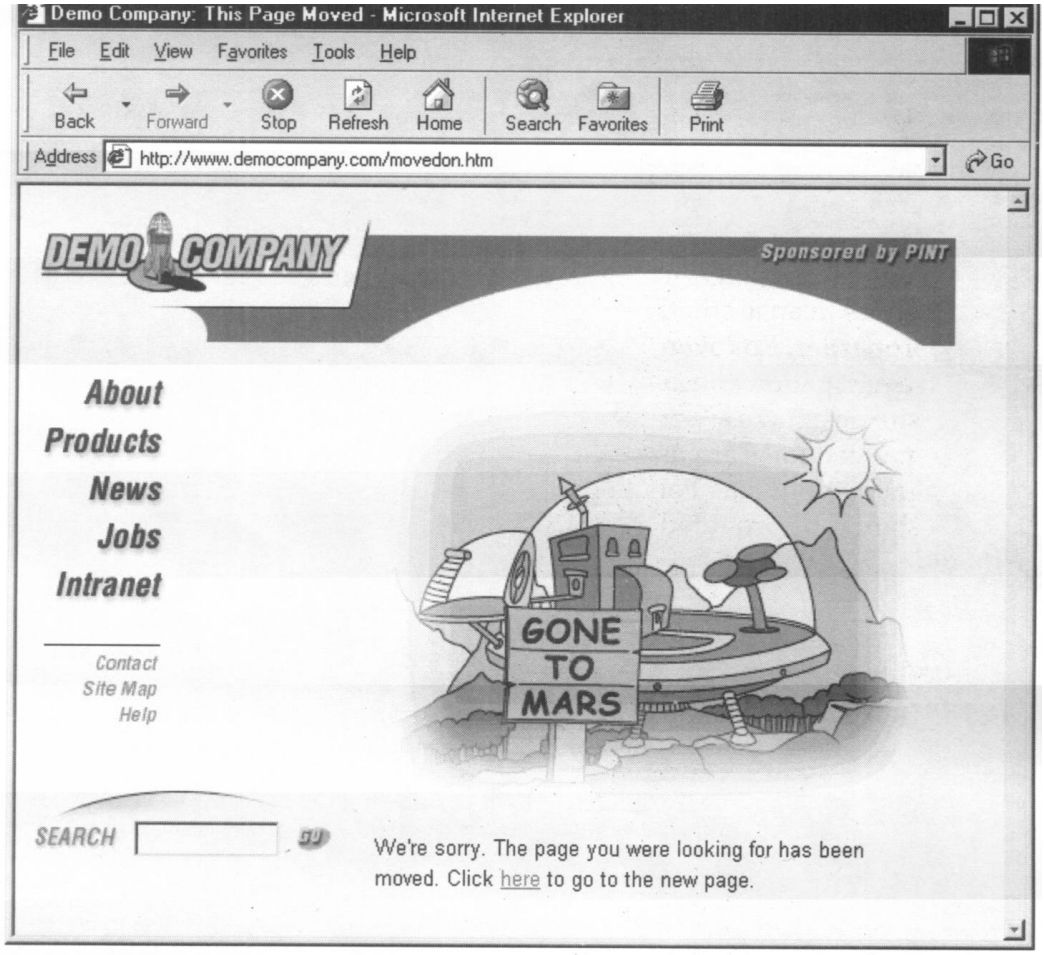


图6-21 “页移动”的示例

建议：避免因为404错误而自动重定向。

维修站点可能需要大量的工作。定制错误页面和重定向页有帮助作用，但 Web维护者不得不警惕对Web链接的监视。好的Web站点维护应该经常参考站点日志文件。进一步说，考虑访问一个搜索引擎，并做反面的搜索。特别地，搜索链接到你的站点的站点并且确保对任何显著的站点变化进行更新。确保和你链接的站点正确，也许是一个工作量很大的任务，但这是成为一个好的Web站点的必要部分。关于链接和站点维护的更多内容见第 14章。

6.8 小结

确保用户理解链接的用途是开发可用站点的一个完整的部分。在一个站点中加入链接的方

法很多，包括文本链接、按钮、图像映射，甚至任意的热点。设计者应该遵循一般的链接惯例如颜色、下划线、URL反馈。但是，可以应用图像或最近出现的CCS属性，从审美的角度来改变链接的风格。滚动和其他的动态功能可以加入到链接中，以进一步提高导航并创建动态站点。维持一个链接是困难的，设计者被鼓励去解决断开的链接并移动页面。即使链接被正确地使用，也一定有用户在站点中遇到导航困难。对这些用户，应为他们提供特殊的设施如搜寻引擎和站点映像，这些在随后的两章中讨论。