由图10-29可以发现，<h4>和<ul>的高度不能固定，因此要求背景图片有一定的延展能力，因此可以如图10-30所示来处理背景。

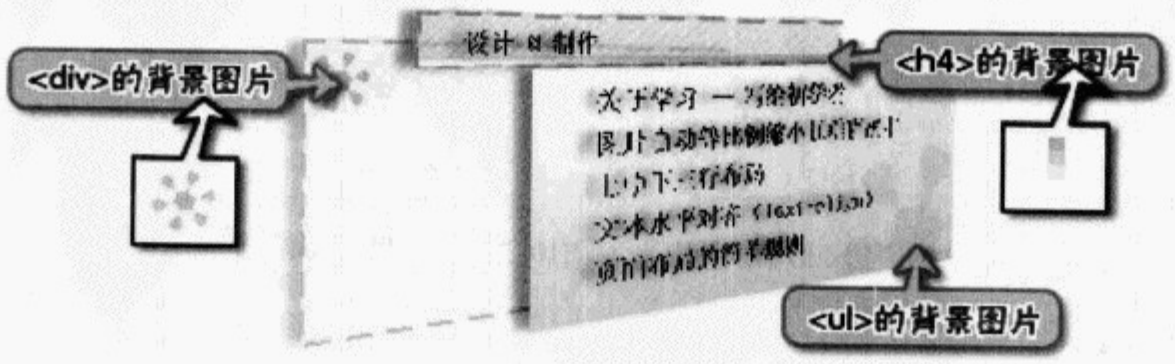


图10-30　设计图背景分析

对于<ul>的背景图片，由于是一张不重复的图片，因此要考虑到在水平和垂直方向一定的扩展能力，因此可以制作得比需要的大一些，如图10-31所示。

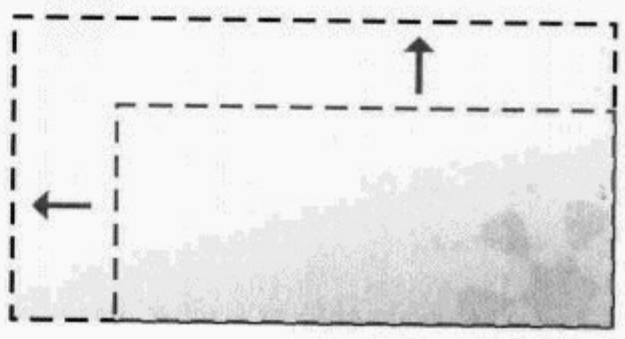


图10-31　背景图片的制作要考虑有一定的扩展能力

因此其CSS如下：

```
#design {
……
background:#fff url(sample_img/design_bg1.png) no-repeat;
}
#design h4 {
……
background:url(sample_img/design_bg2.png) repeat-x bottom;
}
#design ul {
background:url(sample_img/design_bg3.png) no-repeat right bottom;
……
}
#design ul a {
……
}
```

如果访问者调整了字体大小，在一定范围内也不会出现错乱的情况，如图10-32所示。

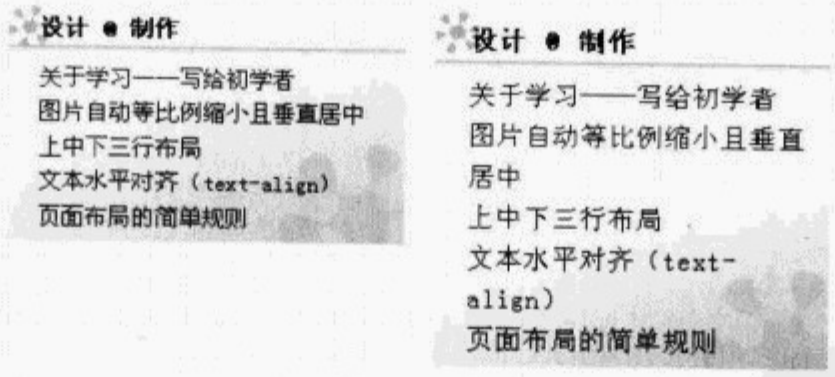


图10-32　访问者调整字体对于页面的影响

10.4.2 模拟边框

由于元素框是矩形的，因此其边框（border）也是矩形，而很多时候，设计师希望通过圆角、阴影、渐变等来划分栏目，因此也需要使用背景图片来完成。而在制作矩形区域时，一般可分为以下3种情况。

1. 固定宽度和高度

这样的情况实现方法很简单，即制作一个完整的背景图片即可，如图10-33所示的设计稿，根据设计其宽度和高度是固定的，其XHTML代码如下，背景图片如图10-34所示。

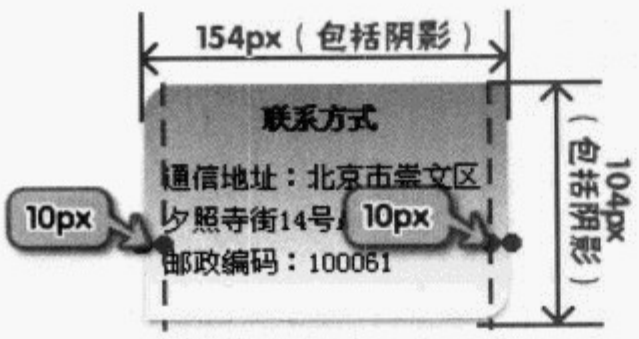


图10-33 固定宽度和高度的圆角矩形的设计图

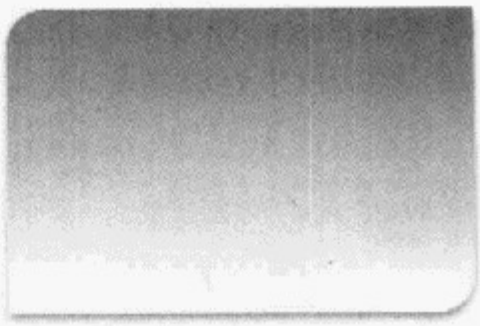


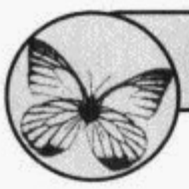
图10-34 固定宽度和高度的圆角矩形的背景图片

```
<div id="contact">
 <h5>联系方式</h5>
 <ul>
  <li>通信地址：北京市崇文区夕照寺街14号A座</li>
  <li>邮政编码：100061</li>
 </ul>
</div>
```

其CSS如下：

```
#contact {
……
background : #FFF url(sample_img/contact_bg.png) no-repeat center top;
}
……
```

由于背景图片是整体的，因此设置“no-repeat（不重复）”，这种方法的局限性很明显，如果内容超出设计的高度，就会造成溢出。



提示：在不影响大局的情况下，文字的行距与设计图稍有差异时可以忽略。

2. 宽度固定高度不定

由于浏览器的宽度是一定的，而高度可以随内容而改变，因此在设计中往往会根据一个分辨率（如800×600）来设计页面，而高度会适应内容的高度。所以，在很多情况下，板块的宽度是固定的，而高度不定，如图10-35所示。

本例的XHTML如下所示：

```
<div id="propertyList">
 <div class="property">
  <h5>background-image</h5>
  <dl>
   <dt>值：&lt;uri&gt; | none | inherit</dt>
   <dd>uri：图片的链接地址。</dd>
   <dd>none：无图片。</dd>
  </dl>
 </div>
 <div class="property">
  <h5>background-repeat</h5>
  <dl>
   <dt>值：repeat | repeat-x | repeat-y | no-repeat | inherit</dt>
   <dd>repeat：背景图片在水平和垂直方向都重复显示。</dd>
   <dd>repeat-x：背景图片只在水平方向重复显示。</dd>
   <dd>repeat-y：背景图片只在垂直方向重复显示。</dd>
   <dd>no-repeat：背景图片不重复，只显示1次。</dd>
  </dl>
 </div>
</div>
```

```
background-image
值：<uri> | none | inherit
uri：图片的链接地址。
none：无图片。
```

```
background-repeat
值：repeat | repeat-x | repeat-y | no-repeat | inherit
repeat：背景图片在水平和垂直方向都重复显示。
repeat-x：背景图片只在水平方向重复显示。
repeat-y：背景图片只在垂直方向重复显示。
no-repeat：背景图片不重复，只显示1次。
```

图10-35 宽度固定高度不定的圆角矩形

对于这种情况，根据实际情况，一般有两种实现方法。

（1）如果矩形框的高度比较小而且高度变化也不会太大的情况下，可以采取如图10-36所示的分割方法。

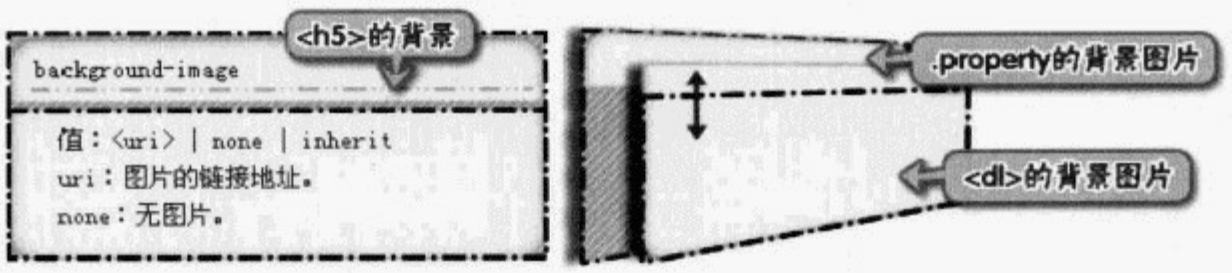


图10-36　矩形框的高度比较小而且高度变化也不会太大的情况

其中， 元素“property”和<dl>的背景图片的高度要比实际设计图的高度高，如图10-37所示。这样当高度有一定的变化时，背景图片也可以保证视觉上边框的完整性。

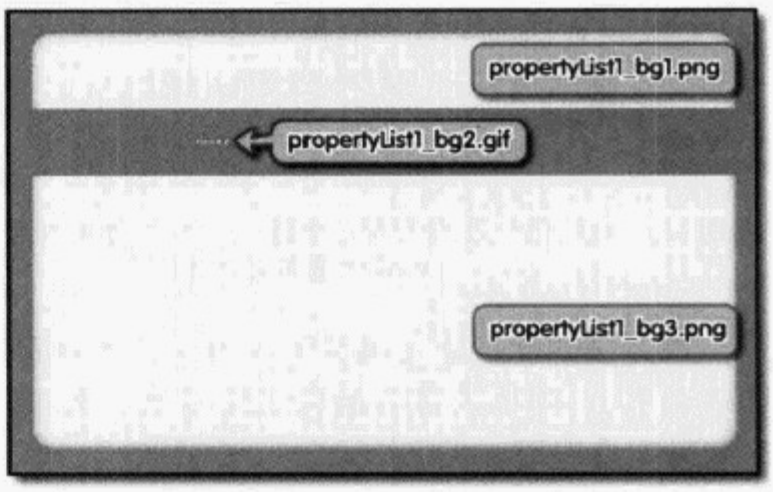


图10-37　“property”和<dl>的背景图片需要比设计稿高一些

其CSS规则如下：

```
.property {
……
    background:#ffc url(sample_img/property
List1_bg1.png) no-repeat;
}
.property h5 {
……
height:25px;
background:url(sample_img/propertyList1_bg2.gif)
repeat-x left bottom;
}
.property dl {
padding:8px 24px; /* 下补白可以保证边框与最后一行文字有一定的距离。 */
background:url(sample_img/propertyList1_bg3.png) no-repeat left bottom;
}
```

本方法的优点在于，背景图片比较少，但是如果内容过多，则可能出现背景图片高度不够而导致边框断开的现象。

（2）如果无法确定矩形的高度，则可以采用如图10-38所示的分割方法。

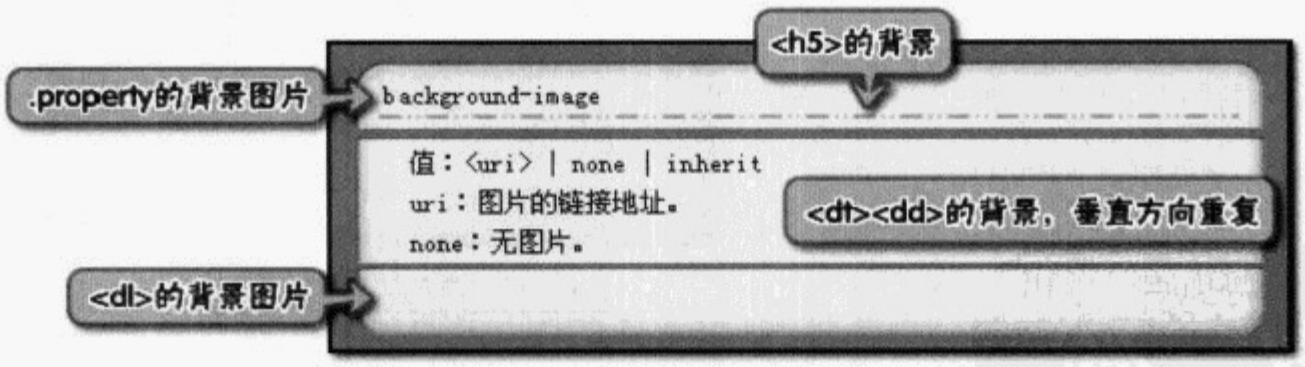


图10-38　无法确定矩形高度的情况

其中， “property”和<h5>的背景维持不变，而<dt>、<dd>和<dl>的背景图片如图10-39所示。<dt>和<dd>的背景图片在垂直方向重复，因此矩形的中间部分根据<dt>、<dd>的多少来伸缩，而头尾的圆角保持不变。

图10-39　<dt>、<dd>和<dl>的背景图片

因此对上例中的CSS规则修改如下：

```
dl {
padding: 8px 0 10px; /* 下补白要比圆角高一些。 */
background: url(sample_img/propertyList2_bg4.png) no-repeat left bottom;
}
dt,
dd {
padding:0 24px;
background: url(sample_img/propertyList2_bg3.png) repeat-y left top;
}
```

此处需要注意<dl>的下补白，如果设置过小，可能造成<dd>的背景图遮盖住<dl>的背景图致使圆角显示不完全，如图10-40所示。

因此<dl>的下补白一定要大于等于圆角的高度，如图10-41所示。

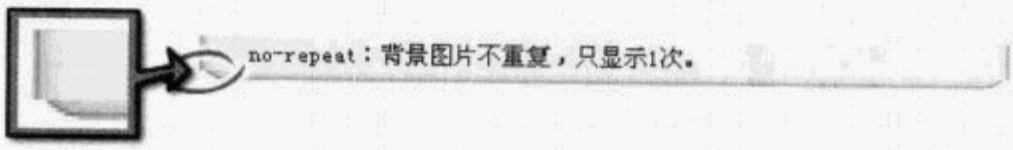


图10-40 <dl>的下补白设置过小造成<dd>的背景图遮盖住<dl>的背景图致使圆角显示不完全

图10-41 <dl>的下补白一定要大于等于圆角的高度

本方法的优点在于高度可以随内容无限延伸，缺点是，由于背景图片由3部分组成（需要3个块级元素），如果内容的结构比较简单（例如只有2个元素），可能需要添加额外的与结构无关的元素来完成表现。

3. 宽度和高度都不固定

当以百分比或者em、ex等作为元素宽度的时候，元素的宽度也是可变的，因此在水平方向和垂直方向都需要一定的延展能力，例如上例中圆角边框，要求宽度也可以变化，则可以如图10-42所示分割背景。

由图10-42可以发现，图片的分割复杂了很多，因此现有的结构已经不能实现其表现，因此需要增加与内容无关的的元素来辅助实现，其XHTML代码如下：

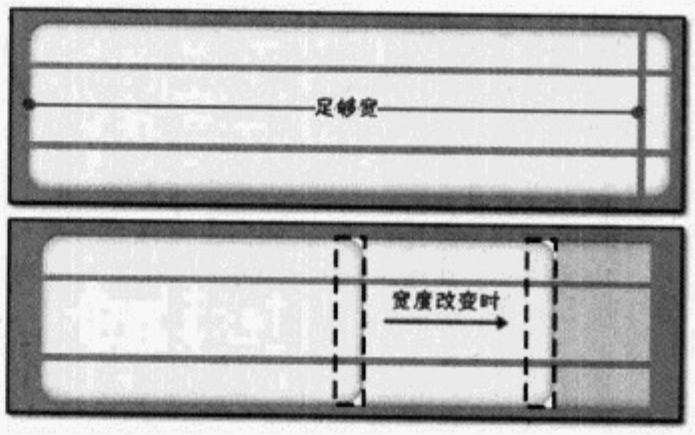


图10-42 水平方向也可以伸缩的矩形

```
<div class="box">
  <div class="box2">
    <div class="box3">
     <div class="box4">
       <h5>background-image</h5>
       <dl>
            <dt>值：&lt;uri&gt; | none | inherit</dt>
        <dd>uri：图片的链接地址。</dd>
        <dd>none：无图片。</dd>
      </dl>
     </div>
    </div>
  </div>
</div>
```

而背景的分配如图10-43所示。

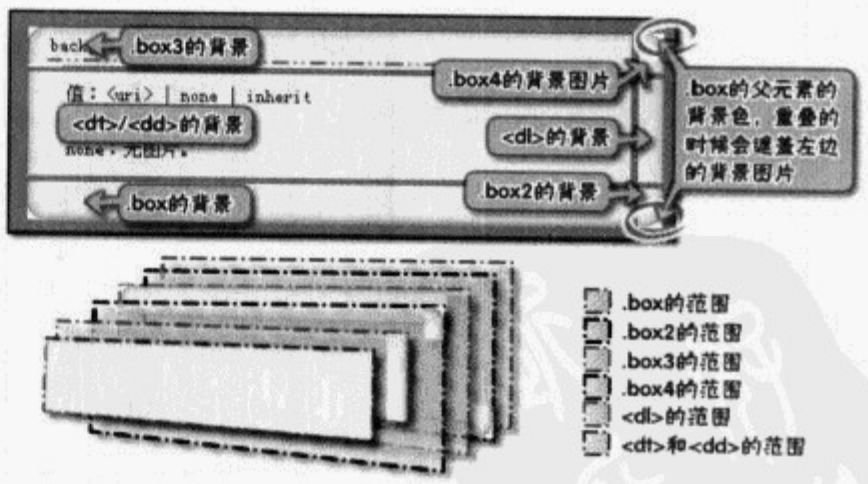
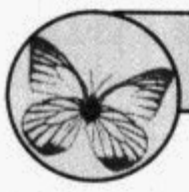


图10-43 各元素背景对应的边框部分



提示：各个背景图片都要考虑到扩展性，因此在宽度上要比设计的宽一些。

其CSS规则如下：

```
.box {
……
width:80%;
background:#ffc url(sample_img/propertyList3_left_bottom.png) no-repeat left bottom;
}
.box2 {
padding-bottom:10px; /* 下补白要比圆角高一些。 */
background:url(sample_img/propertyList3_right_bottom.png) no-repeat right bottom;
}
.box3 {
background:url(sample_img/propertyList3_left_top.png) no-repeat;
}
.box4 {
background:url(sample_img/propertyList3_right_top.png) no-repeat right top;
}
.box4 h5 {
……
margin:0 12px 5px; /* 右边距要大于等于圆角的宽度。 */
background:url(sample_img/propertyList1_bg2.gif) repeat-x left bottom;
}
.box4 dl {
background: url(sample_img/propertyList3_right_center.png) repeat-y right top;
}
.box4 dt,
.box4 dd {
margin-right:10px; /* 右边距要大于等于圆角的宽度。 */
padding:0 14px 0 24px;
background: url(sample_img/propertyList3_left_center.png) repeat-y left top;
}
```

其在浏览器内显示如图10-44所示。

图10-44　宽度和高度都可变的圆角矩形

此方法虽然可以实现比较灵活的圆角矩形，但是也添加了很多无语义的额外元素，使结构变得复杂。

10.4.3 简单的链接背景替换

在前面的章节内，多次介绍过利用链接和动态伪类实现不同的效果，而链接本身，特别是导航条内的链接，往往需要进行美化，变化背景图片来实现特殊效果。如图10-45所示，为一导航条的设计效果图，其中鼠标指向时前景和背景都改变颜色。

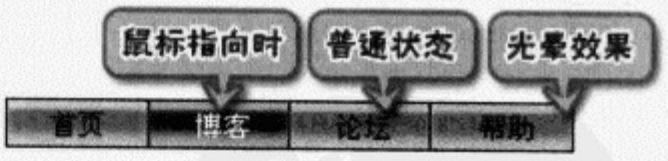
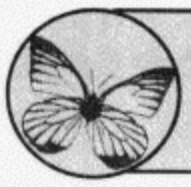


图10-45　导航条设计效果图



提示：本例中，导航的左右边框处有光晕效果，因此不能使用水平方向背景重复的方法来实现。如果左右边框处没有特殊效果，则只需设定背景的水平重复即可。

导航的XHTML代码如下：

```
<div id="nav1">
 <ul>
  <li><a href="#" title="回到首页">首页</a></li>
  <li><a href="#" title="访问博客">博客</a></li>
```

```
    <li><a href="#" title="访问论坛">论坛</a></li>
    <li><a href="#" title="在线帮助">帮助</a></li>
  </ul>
</div>
```

对设计效果图详细分析，如图10-46所示。

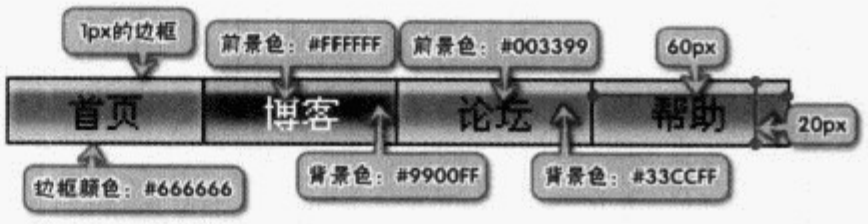


图10-46 分析设计效果图

对设计图主要注意以下几点。

● 需要左浮动（float：left），以实现横排显示；

● <a>元素为行内元素，行内元素不能设定高度和宽度，因此为了完整显示背景图片，需要设定其display属性为“block”；

● 边框的实现方法比较多，例如可以通过设定<li>的边框来实现，也可以通过<ul>的边框来实现，需要根据实际的应用情况来定，在本例中只采用最简单的方法。

CSS规则定义如下，其显示如图10-47所示。

```
#nav1 {
margin : 10px;
}
#nav1 ul {
float : left; /* 浮动元素高度会包含其内浮动的子元素 */
border-left : 1px solid #666;
list-style:none;
}
#nav1 li {
float : left; /* li浮动可能会导致其父元素无高度 */
border : 1px solid #666;
border-left : 0; /* 去掉左边框 */
}
#nav1 li a {
display : block;
text-align : center;
line-height : 20px;
height : 20px;
width : 60px;
color : #039;
background : #3CF;
text-decoration : none;
}
#nav1 li a:hover {
color : #FFF;
background-color : #90F;
}
```

图10-47 初步完成CSS后浏览器内显示

图10-48 背景图替换实现原理

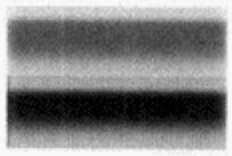


图10-49 <a>元素的背景图片

由于普通链接的背景图和鼠标指向时的不同，因此需要分别指定，但是如图将普通状态和鼠标指向的图片分别保存为2个文件，然后通过background-image属性指定，在鼠标指向的时候，由于浏览器从服务器读入图片需要时间（视网络情况而定），因此可能会出现背景图片的延迟显示，为了解决这个问题，可以将这2个状态的图片放在同一个图片文件内，然后通过设置背景图片的定位来指定显示图片的哪一部分，这样，两种状态的背景图片在页面加载的时候就会被浏览器载入，因此在鼠标指向的时候，也不会出现读取图片的延迟，其原理如图10-48所示。

因此<a>元素的背景图片应该如图10-49所示。

修改<a>元素的background属性，如下所示：

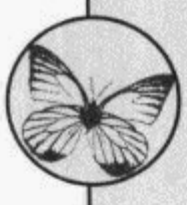
```
#nav1 li a {
……
background : #3CF url(sample_img/nav1_a.png) no-repeat;
}
```

同时，为“a:hover”增加背景图片定位，如下所示：

```
#nav1 li a:hover {
color : #FFF;
background-color : #90F;
background-position : left bottom;
}
```

图10-50 完成效果

此处的背景属性不能缩写，因为缩写会覆盖掉前面的定义。此时在浏览器内显示如图10-50所示。



提示：根据实际情况，还可以定义“:visited”等规则。利用此原理可以实现其他元素的“:hover”伪类背景图片替换效果，而对于不支持除<a>元素以外的“:hover”的浏览器（例如IE 6.0）则需要使用JavaScript来辅助实现，读者可以参考http://www.ddcat.net/bbs2007/showthread.php?t=656。

10.4.4 导航菜单的滑动门效果

在上一小节内介绍了基本的背景图片替换的导航菜单，这种实现方法存在一个问题，即<a>元素的宽度和高度是固定的，即所有的<a>元素都是相同的宽度，而在实际情况中，导航文字的数量可能不尽相同，同时，为了提高可访问性，浏览器允许访问者自己设定文字大小，当访问者设定了比较大的字号的时候，可能会打乱布局，如图10-51所示。

图10-51 简单的链接背景图片替换效果存在的问题

一种被称为“滑动门”的技巧，可以解决这个问题。滑动门的实现思路，和［10.3.2 边框］一节中宽度不固定的情况类似，即将边框分为左右2部分，左边的背景图片制作的比较宽，以适应不同的元素宽度，这就如同2个可以滑动的门，它们滑到一起并交叠，占据一个较窄的空间，或者相互滑开，占据一个较宽的空间，如图10-52所示。

图10-52 滑动门的原理

而在实际的应用中，2个门的宽度并不是平分的，而是如图10-53所示。

图10-53 2个门的宽度不等

左边的门是固定不动的，而右边的门会根据实际的宽度滑动。如图10-54所示的设计效果图，其XHTML代码如下。

图10-54 设计效果图

```
<div id="nav2">
 <ul>
  <li><a href="#" title="回到首页">首页</a></li>
  <li><a href="#" title="源代码下载">源代码下载</a></li>
```

```
    <li><a href="#" title="关于我们">关于我们</a></li>
    <li><a href="#" title="网站地图">网站地图</a></li>
   </ul>
  </div>
```

其背景划分如图10-55所示。

图10-55 背景图片的分割

由于此处的链接也要采用和上例中鼠标普通状态与指向状态背景图的处理方法，同时，由于实现滑动门需要2个元素，而IE 6.0不支持非<a>元素的“:hover”伪类，因此需要修改XHTML代码，在<a>元素内添加额外的元素来实现效果，相应代码如下，实际背景图的制作如图10-56所示。

```
<div id="nav2">
 <ul>
  <li><a href="#" title="回到首页"><strong>首页</strong></a></li>
  <li><a href="#" title="源代码下载"><strong>源代码下载</strong></a></li>
  <li><a href="#" title="关于我们"><strong>关于我们</strong></a></li>
  <li><a href="#" title="网站地图"><strong>网站地图</strong></a></li>
 </ul>
</div>
```

在本例中，也需要将<li>元素左浮动，而<a>和元素需要设定display属性为“block”，同时还要注意，由于<li>元素的浮动，将导致<ul>乃至 都没有高度，可以选择以下方法之一解决此问题：

- 为<ul>和 设定一个height属性值；
- 将 和<ul>元素浮动；
- 在最后一个浮动的<li>元素后清除浮动。

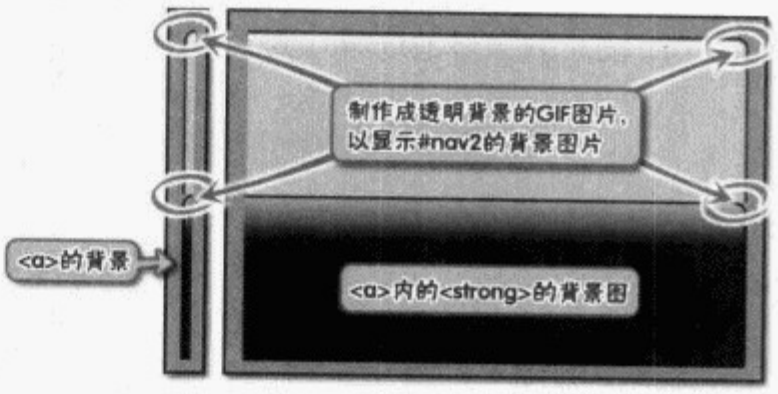


图10-56 背景图片的制作

首先制作链接的效果，链接部分的细化分析如图10-57所示，由图10-57可以发现，为了使文字的大小发生改变时，导航栏在一定的范围内也可以具有“弹性”，因此不直接设定元素的width和heigth属性，而是利用文字本身的字体尺寸和元素的补白来控制文字的位置。

由图10-57的分析可以设定CSS规则如下：

```
#nav2 li {
float : left;
list-style : none;
}
#nav2 li a {
color : #666;
text-decoration : none;
display : block;
padding-left : 5px;
background : url(sample_img/nav2_left.gif) no-repeat;
}
#nav2 li a strong {
font-weight : normal;
display : block;
padding : 8px 10px 5px 5px;
background : url(sample_img/nav2_right.gif) no-repeat right top;
}
#nav2 li a:hover {
color : #fff;
background-position:0 -100px;
```

```
}
#nav2 li a:hover strong {
background-position : 100% -100px;
}
```

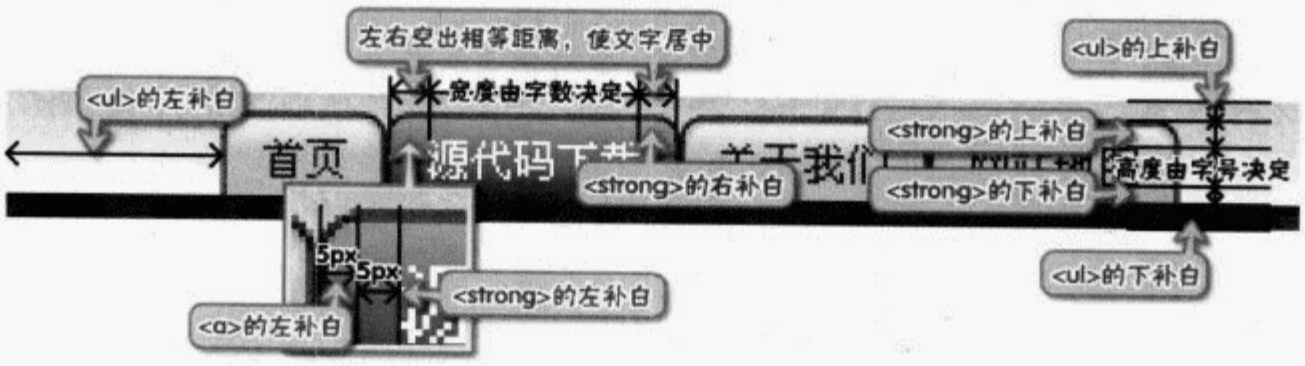


图10-57　链接部分的细节分析

此时的显示如图10-58所示。此时如果改变文字的字号大小，其背景显示也是完整的，如图10-59所示。此时，由于<li>元素左浮动，因此链接全部靠左显示，因此需要设置<ul>的左补白如下：

图10-58　设定好CSS规则的链接元素的显示

图10-59　改变字号大小背景也可完整显示

```
#nav2 ul {
padding : 5px 0 7px 60px;
}
```

其显示如图10-60所示。

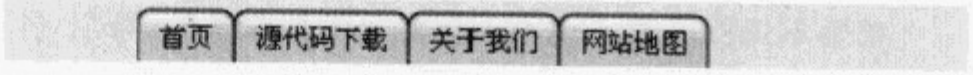


图10-60　设置<ul>的补白

和 元素的背景图片分析如图10-61所示。

图10-61　和<ul>元素的细节分析

由于<li>全部浮动，<ul>元素（有补白部分）和 元素没有高度，此时设置它们的背景图片也将无法正常显示，有以下几种方法可以解决这个问题，但是在应用中还要结合实际情况来决定使用哪种方法。

1. 设置 和<ul>浮动

浮动的元素会适应子元素的高度，但是浮动的元素如果不设定宽度，则会被压缩，因此如果 和具有确定的宽度（例如600px），则可以如下定义CSS：

```
#nav2 {
font : 12px/1em "宋体", serif;
background : #fff url(sample_img/nav2_bg1.png) repeat-x;
width : 600px;
float : left;
}
#nav2 ul {
width : 540px;  /* 600px - 60px*/
background : url(sample_img/nav2_bg2.gif) repeat-x left bottom;
float : left;
padding : 5px 0 7px 60px;
}
```

但是，如果 以百分比或者em等为单位，则无法准确计算出<ul>的宽度，因此无法实现<ul>撑满父元素的效果。同时， 的浮动有可能会造成 旁边的元素的错位或者其他问题。

2. 清除浮动

为浮动元素后面的非浮动元素设定clear属性可以清除浮动造成的高度问题，但是在本例中，浮动的是<li>元素，而其后没有其他元素，因此如果采用本方法，需要添加一个空元素，相应的代码如下：

```
#nav2 .clearFloat {
float : none;
clear : both;
}
<div id="nav2">
 <ul>
  ……
  <li><a href="#" title="网站地图"><strong>网站地图</strong></a></li>
  <li class="clearFloat"></li>
 </ul>
</div>
```

这个额外的元素在IE 内会产生空白，如图10-62所示，同时，这个元素也是多余的。因此一般不建议使用此方法来清除浮动。

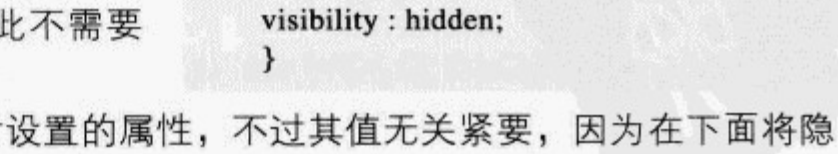


图10-62 空元素在IE内产生空白

3. 只针对非IE浏览器清除浮动

如果不浮动元素，也不使用额外的元素清除浮动，则在IE内预览页面，可以发现 会自动扩展高度以包含子孙元素，只是<ul>元素的高度不合要求，因此如图10-63所示。

```
ul {
……
height : 100%;
}
```

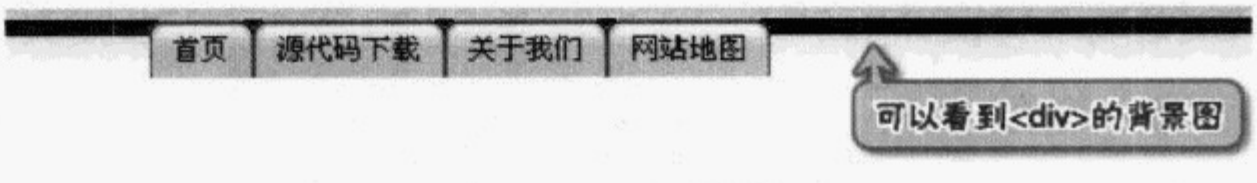


图10-63 IE的浏览效果

而此时如果为<ul>添加CSS规则如下，其显示如图10-64所示。

图10-64 为<ul>元素添加height属性后的显示

此时在IE内已经达到设计要求，而在其他的浏览器内显示如图10-65所示。

图10-65 在非IE浏览器内的显示

因此只要针对非IE浏览器清除浮动即可。由于clear属性是清除本元素前面的浮动元素，因此，可以利用:after伪元素来实现清除。

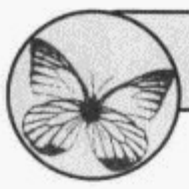
本书［4.3.2 伪元素（Pseudo-Elements）］一节内简要介绍过:after伪元素，它用来在一个元素的内容之后插入生成的内容，因此可以利用它来清除浮动，而这个伪元素并不存在于(X)HTML文档内，因此不需要修改(X)HTML代码，其CSS规则如右：

```
#nav4 ul:after {
content : ".";
display : block;
height : 0;
clear : both;
visibility : hidden;
}
```

其中content属性是必须与:after伪类同时设置的属性，不过其值无关紧要，因为在下面将隐藏这个伪元素，因为它的作用只是用来清除浮动。由于IE不支持:after伪元素，因此会忽略，而

支持:after伪元素的浏览器也能正确显示，如图10-66所示。

图10-66　清除浮动后非IE浏览器内的显示



提示：此方法可以应用在其他需要清除浮动的情况。

至此，效果已经基本完成，但是鼠标指向链接的时候还有一个细节，如图10-67所示。

由图10-67可以发现，鼠标指向时链接元素比普通状态高2px，盖住了横线，因此可以增加元素的:hover伪类的下补白2px，如下：

```
#nav2 li a:hover strong {
padding-bottom:7px; /* 5px + 2px */
background-position:100% -100px;
}
```

图10-67　鼠标指向链接时的细节

但是，如果只增加下补白，则会产生空白，如图10-68所示。因此需要同时设定元素的:hover伪类的下边距为-2px，如下：

```
#nav2 li a:hover strong {
……
margin-bottom : -2px;
}
```

图10-68　增加下补白造成了空白

此时在Firefox 2.0、Opera 9.2等浏览器内预览如图10-69所示。但是在IE浏览器的显示如图10-70所示。

图10-69　在Firefox 2.0、Opera 9.2等浏览器内的显示

图10-70　IE浏览器的显示

这是由于<li>元素限制了其子孙元素的高度，因此需要修改CSS，去掉先前设定的<ul>元素的下补白，而将其应用在<li>元素上，这样就扩大了<li>元素的高度，使能正常遮盖，CSS规则如下，其显示如图10-71所示。

```
#nav2 ul {
……
padding : 5px 0 0 60px;
}
#nav2 li {
……
padding-bottom : 7px;
}
```

图10-71　修改<li>和<ul>元素的下补白后IE浏览器的显示

至此，滑动门效果的导航条制作完毕。

11

Chapter

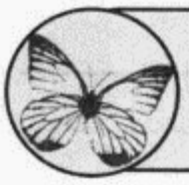
第11章 表格

表格曾经是用来布局的利器，直到现在很多网站依然在使用相互嵌套的表格来排布页面，但是，表格的实际“语义”却只是用来放置表格类的数据的，例如班级的学生成绩单或者课程表。

在本章将介绍CSS中关于表格的视觉格式化内容以及行、行组、列、列组、单元格等表格类元素之间的关系，以及可以应用的CSS属性。

11.1 表格的标签与属性

在(X)HTML中，用来构建表格的是一系列的标签，在本书［2.3.2 常用的XHTML标签和属性］一节内已经简要介绍过表格常用的标签及属性，本节将进一步介绍这些标签和属性。



提示：本节示例代码，读者可参见下载文件包内［/第2部分/第11章：表格/table.html］文件。

11.1.1 标签概览

与表格有关的标签有以下几种，其中很多标签的属性（Attribute）可以用CSS的属性（Property）来代替，不过其中有些属性可能存在浏览器支持的问题。

1. <table>

表格以<table>标签为开始，</table>标签为结束，其他的标签都要按照一定的顺序包含在<table>标签之内。

<table>	表格
说明	成对出现，以<table>开始，以</table>结束
属性	Common：一般属性。 summary：代表表格的摘要说明。 width：代表表格的宽度（可通过CSS的width属性实现）。 border：代表表格边框（可通过CSS的border属性实现）。 cellspacing：代表表格边框与表格内容填充的距离,也是内容填充之间的距离。 cellpadding：代表内容填充的宽度（可通过CSS的padding属性实现）

有些属性在很多标签上的意义都是一样的，它们可以分成几组集合(Attribute Collections)。XHTML 1.1的属性集合如下。

- 原属性（Core）：xml:space、class、id和title。
- I18N属性（Internationalization）：xml:lang。
- 事件属性（UI events）：onclick、ondblclick、onmousedown、onmouseup、onmouseover、onmousemove、onmouseout、onkeypress、onkeydown、onkeyup。
- 样式属性（Style）：style。
- 一般属性（Common）：上面4组属性合在一起被称为一般属性（Common），即Common = Core + Events + I18N + Style。

大多数标签都具有“一般属性（Common）”，有一些标签还具有自己的“私有属性”，比如，链接或表单元素等的accesskey属性、链接的href属性等。

表格不仅可以有标题<caption>，还可以有一个摘要说明“summary”，摘要是不会显示出来的，通常是给一些其他媒体使用的，比如盲人阅读器等。由于摘要不会显示在浏览器中，所以可以尽可能地使摘要描述足够详细，这样更有利于那些通过“听”的浏览者了解你的表格内容。

2. <caption>

<caption>用来定义表格的标题，为表格提供一个描述。默认情况下，浏览器将标题显示在

表格的上方。CSS里的caption-side属性用来控制表格标题显示的位置。

<caption >	表格的标题
说明	成对出现，以< caption>开始，以</caption>结束
属性	Common：一般属性

3. <tr>

<tr>是“table row”的缩写，即表格行，<table>里面可以有很多行，每一行使用<tr>表示，每个行<tr>里面又可以有很多列，每一列使用<td>表示。

<tr>	表格的1行
说明	成对出现，以<tr>开始，以</tr>结束
属性	Common：一般属性。 align：行的水平对齐方式，left | center |right | justify（可通过CSS的text-align来实现）。 valign：代表行的垂直对齐方式，top | middle | bottom | baseline（可通过CSS的vertical-align属性实现）

4. <th>

<th>代表表格中的表头，其本质也是单元格。

<th>	表格的表头单元格
说明	成对出现，以<th>开始，以</th>结束
属性	Common：一般属性。 abbr：代表表头的简写。 axis：对单元格在概念上分类。 colspan：一行跨越多列。 headers：连接表格的数据与表头，值可以是一系列以空格分开的标签的id。 rowspan：一列跨越多行。 scope：定义行或列的表头。 align：行的水平对齐方式，left | center |right | justify（可通过CSS的text-align来实现）。 valign：代表行的垂直对齐方式，top | middle | bottom | baseline（可通过CSS的vertical-align属性实现）

5. <td>

<td>是“table data cell”的缩写，代表表格中的单元格。

<td>	表格的单元格
说明	成对出现，以<td>开始，以</td>结束
属性	Common：一般属性。 abbr：代表表头的简写。 axis：对单元格在概念上分类。 colspan：一行跨越多列。 headers：连接表格的数据与表头，值可以是一系列以空格分开的标签的id。 rowspan：一列跨越多行。 scope：定义行或列的表头。 align：行的水平对齐方式，left | center |right | justify（可通过CSS的text-align来实现）。

续表

	valign：代表行的垂直对齐方式，top | middle | bottom | baseline（可通过CSS的vertical-align属性实现）

一个表格可以有若干<tr>元素，<tr>内又可以包含若干个<th>或者<td>，例如下列代码，其显示如图11-1所示。

```
table {
text-align : left;
width : 300px;
border : 1px solid #06C;
}
th,
td {
border : 1px dotted #3CF;
}
<table summary="样例表格，标示表格组成部分，本表格为4行3列表格。">
 <caption>
 样例表格
 </caption>
 <tr>
  <th scope="col">编号</th>
  <th scope="col">姓名</th>
  <th scope="col">年龄</th>
 </tr>
 <tr>
  <td>1</td>
  <td>张三</td>
  <td>15</td>
 </tr>
 <tr>
  <td>2</td>
  <td>李四</td>
  <td>16</td>
 </tr>
 <tr>
  <td>3</td>
  <td>王五</td>
  <td>14</td>
 </tr>
</table>
```

图11-1　表格的各组成元素

6. <colgroup>和<col>

<colgroup>是对(X)HTML表格进行结构化的分区，在此分区中可以通过使用col定义每列表格的样式。

<colgroup>	表格结构化的分区
说明	成对出现，以<colgroup>开始，以</colgroup>结束
属性	Common：一般属性。 span：定义一个colgroup跨越的列数，默认值为1

<col>可以对(X)HTML表格结构化分区后的一个或几个区域使用同样的样式行。

<col>	可以对表格结构化分区后的一个或几个区域使用同样的样式行
说明	成对出现，以<col>开始，以</col>结束
属性	Common：一般属性。 span：定义一个col跨越的列数，默认值为1

一般情况下，可以不定义列和列组，但是如果要针对列设定样式，则可以使用，例如下列代码，其显示如图11-2所示。

```
.....
.style1 {
background:#CFC;
}
.style2 {
background: #FF9;
}
<table summary="样例表格2，列组和列。" id="sample3">
 <caption>
 样例表格2：列组和列
 </caption>
 <colgroup span="2" class="style1" />
 <colgroup>
     <col span="2" class="style2" />
     <col />
 </colgroup>
 <tr>
  <th scope="col">编号</th>
  <th scope="col">姓名</th>
  <th scope="col">年龄</th>
  <th scope="col">籍贯</th>
  <th scope="col">性别</th>
 </tr>
 <tr>
  <td>1</td>
  <td>张三</td>
  <td>15</td>
  <td>山东</td>
  <td>男</td>
 </tr>
 <tr>
  <td>2</td>
  <td>李四</td>
  <td>16</td>
  <td>山东</td>
  <td>女</td>
 </tr>
</table>
```

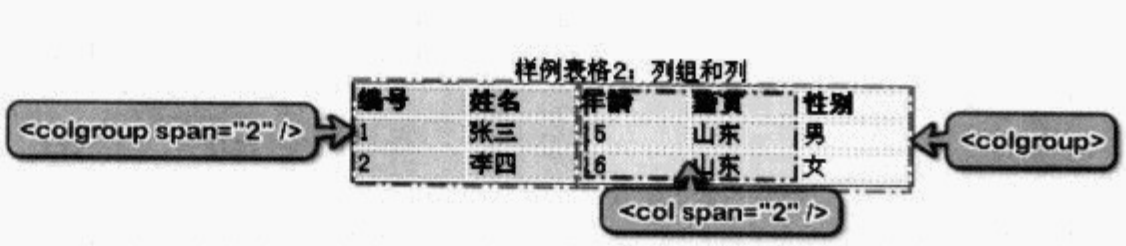


图11-2 列组和列

7. <thead>、<tbody>和<tfooter>

这3个标签必须同时出现在<table>内，浏览器对其支持可能不一致。

● <thead>标签为表格头部，其内必须有<tr>标签。可以使用单独的样式定义表头，并且在打印时可以在分页的上部打印表头。

● <tbody>标签为表格主体内容，其内必须有<tr>标签。浏览器显示表格时，通常是完全下载表格后，再全部显示，所以当表格很长时，可以使用多个<tbody>分段显示。

● <tfoot>标签为表格注脚，其内必须有<tr>标签。可以使用单独的样式定义注脚，并且在打印时可以在分页的上部打印注脚。

11.1.2 (X)HTML属性

上节中介绍的表格标签还可以具有以下几种属性。

1. rowspan属性

rowspan属性可以实现单元格跨越多行，例如下列代码，其显示如图11-3所示。

```
<table summary="样例表格3，rowspan属性。" id="sample3">
 <caption>
  样例表格3：rowspan属性
 </caption>
 <tr>
  <th scope="row">属性</th>
  <th scope="col">float</th>
  <th scope="col">text-aling</th>
 </tr>
 <tr>
  <th scope="col" rowspan="4">值</th>
  <td>left</td>
  <td>left</td>
 </tr>
 <tr>
  <td>right</td>
  <td>center</td>
 </tr>
 <tr>
  <td>none</td>
  <td>right</td>
 </tr>
 <tr>
  <td> </td>
    <td>justify</td>
 </tr>
</table>
```

样例表格3：rowspan属性

属性	float	text-aling
值	left	left
	right	center
	none	right
		justify

rowspan="4"

图11-3 rowspan属性的表现

2. colspan属性

colspan属性和rowspan属性类似，实现单元格跨越多列，例如下列代码，其显示如图11-4所示。

```
<table summary="样例表格4，colspan属性。" id="sample4">
 <caption>
  样例表格4：colspan属性
 </caption>
 <tr>
  <th scope="col">属性</th>
  <th scope="col" colspan="4">值</th>
 </tr>
 <tr>
  <th scope="col">float</th>
  <td>left</td>
  <td>right</td>
  <td>none</td>
  <td> </td>
 </tr>
 <tr>
  <th scope="col">text-aling</th>
  <td>left</td>
  <td>center</td>
  <td>right</td>
    <td>justify</td>
 </tr>
</table>
```

样例表格4：colspan属性

属性	值			
float	left	right	none	
text-aling	left	center	right	justify

colspan="4"

图11-4 colspan属性的表现

3. abbr属性

表格元素的abbr属性代表表头缩写，通常是提供给某些网页阅读器使用的。例如：

```
<table summary="样例表格，abbr属性。" id="sample5">
 <caption>
 样例表格5：abbr属性
```

```
  </caption>
  <tr>
    <th scope="col">日期</th>
    <th scope="col">数量</th>
    <th scope="col">价格</th>
  </tr>
  <tr>
    <td abbr="200801">2008年1月</td>
    <td>1000</td>
    <td>1500</td>
  </tr>
  <tr>
    <td abbr="200802">2008年2月</td>
    <td>2000</td>
    <td>3000</td>
  </tr>
</table>
```

4. headers属性

headers属性定义了与表格单元格相关联的一系列表头，例如：

```
<table summary="样例表格，abbr属性。" id="sample6">
  <caption>
  样例表格6：headers属性
  </caption>
  <tr>
    <th rowspan="2" scope="col">日期</th>
    <th colspan="2" scope="col" id="cn">国内</th>
    <th colspan="2" scope="col" id="int">国际</th>
  </tr>
  <tr>
    <th scope="col" id="cn_type">类型</th>
    <th scope="col" id="cn_amount">数量</th>
    <th scope="col" id="int_type">类型</th>
    <th scope="col" id="int_amount">数量</th>
  </tr>
  <tr>
    <td abbr="200801" id="y200801">2008年1月</td>
    <td headers="cn cn_type y200801">手提包</td>
    <td headers="cn cn_amount y200801">1500</td>
    <td headers="int int_type y200801">手提包</td>
    <td headers="int int_amount y200801">1500</td>
  </tr>
  <tr>
    <td abbr="200802" id="y200802">2008年2月</td>
    <td headers="cn cn_type y200802">旅行箱</td>
    <td headers="cn cn_amount y200802">1000</td>
    <td headers="int int_type y200802">旅行箱</td>
    <td headers="int int_amount y200802">2500</td>
  </tr>
</table>
```

5. scope属性

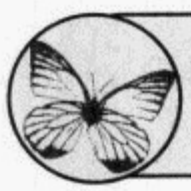
scope属性定义了行或列的表头，其取值可以如下。

- col：定义列表头。
- row：定义行表头。
- colgroup：定义列组的表头信息，是column group的缩写。
- rowgroup：定义行组的表头信息，是row group的缩写。

提示：本属性的例子，请参见本章［11.1.1.4 <th>］一节。

11.2 CSS的表格模型

在CSS中除了前面章节介绍的常规流向、浮动和定位，还有一种“表格布局”，此处所指的布局也同设计中的版面布局无关，而是指用户端如何来呈现表格类型的元素。



提示：本节示例代码，可以参见下载文件包内［/第2部分/第11章：表格/table_model.html］文件。

11.2.1 表格模型概述

在CSS 2.1中，同一行的单元格总是具有相同的高度，无论此行内有多少单元格，或者单元格内的内容有多少，而同一列中的单元格则具有相同的宽度，这种不同元素之间的互相影响是表格都有的特征。也正是如此，有很多CSS属性不能应用于表格元素中。

CSS的表格模式基于HTML 4.0的表格模式，表格的结构和其视觉布局很接近。在这个模型中，表格由1个可选择的标题（<caption>）和任意行及其内的若干单元格组成。表格模型通常称为“行优先”，因为制作者在文档语言中显式地指定行而不是列。一旦所有的行被指定，列就可以被派生出来：每行中第1个单元格属于第1列，第2个属于第2列，依次类推。例如下列代码：

```
<table summary="样例表格，标示表格组成部分，本表格为4行3列表格。" id="sample1">
 <caption>
 样例表格
 </caption>
 <tr>
  <th scope="col">编号</th>
  <th scope="col">姓名</th>
  <th scope="col">年龄</th>
 </tr>
 <tr>
  <td>1</td>
  <td>张三</td>
  <td>15</td>
 </tr>
 <tr>
  <td>2</td>
  <td>李四</td>
  <td>16</td>
 </tr>
</table>
```

行的数量由<tr>的数量来决定，而列的数量则由<td>来决定。行与列可以在结构上被分组，并反应到呈现中（例如，在一组行的周围可能画上一个边框）。因此，表格模型包含了表格、标题、行、行组、列、列组以及单元格。

11.2.2 display属性

CSS模型并不要求文档语言包含对应这些元素中每一个的元素。对于某些文档语言（如XML），

并没有预定义的表格元素，因此作者必须建立文档语言元素到表格元素的映射。这是通过display属性完成的。

在本书［9.2 显示类型：display属性］一节曾经介绍过display属性，而其属性值中与表格模型相关的值如下。

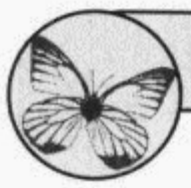
- table（HTML：<table>）：指定了一个元素，定义了一个块级表格，它是一个长方形的块，并参与块格式化内容。
- inline-table（HTML：<table>）：指定了一个元素，定义了一个行内表格，它是一个长方形的块，并参与行内格式化内容。
- table-row（HTML：<tr>）：指定一个元素为单元格组成的行。
- table-row-group（HTML：<tbody>）：指定一个元素将一行或多行分组。
- table-header-group（HTML：<thead>）：类似“table-row-group”，但是对于视觉格式化而言，头部行组总在所有其他行、其他行组前显示，但在任何顶部的标题之后显示。用户端在打印时可能在表格延展的每页都重复打印头部行组。
- table-footer-group（HTML：<tfoot>）：类似“table-row-group”，但是对于视觉格式化而言，注脚行组总在所有其他行、其他行组之后显示，但在任何底部的标题之前显示。用户端在打印时可能在表格延展的每页都重复打印脚注行组。
- table-column（HTML：<col>）：指定一个元素描述了单元格列。
- table-column-group（HTML：<colgroup>）：指定一个元素将一列或多列分组。
- table-cell（HTML：<td><th>）：指定一个元素代表了一个单元格。
- table-caption（HTML：<caption>）：指定了表格的标题。

使用display的相关值，可以使元素呈现表格的外观，例如下列代码，其显示如图11-5所示。

```
#sample2 {
display:table;
……
}
#sample2 ul {
display:table-row;
}
#sample2 li {
display:table-cell;
border: 1px dotted #fff;
}
<div id="sample2">
 <ul>
  <li>第1项</li>
  <li>第2项</li>
  <li>第3项</li>
 </ul>
</div>
```

图11-5 利用display属性来模拟表格

由图11-5可以发现，<li>元素呈现了单元格的特征。



提示：IE 7.0及更早的版本不支持display的表格类值。

如果一个元素的display设置为“table-column”或“table-column-group”，它不会被渲染（就像被设置为“display:none”一样），但是它们还是有用的，因为它们的有些属性可能会给它们所代表的列引入某些特定的样式。

在HTML 4.0默认样式表中表格元素的display属性如下：

```
table { display: table }
tr { display: table-row }
thead { display: table-header-group }
tbody { display: table-row-group }
tfoot { display: table-footer-group }
col { display: table-column }
colgroup { display: table-column-group }
td, th { display: table-cell }
caption { display: table-caption }
```

用户端可能会忽略HTML表格元素的display属性值，因为HTML表格可能会使用向后兼容的算法来呈现。

11.2.3 匿名表格对象

非HTML的文档语言可能不包含CSS 2.1表格模型中所有的元素。在这种情况下，必须假定没有“缺失”元素，以便表格模型可以工作。

表格元素会在自己周围自动地生成必需的匿名表格元素，最少由3层嵌套对象组成，包括“table/inline-table”元素、“table-row”元素和“table-cell”元素。

例如右边的XHTML代码：

```
<table>
 <td>姓名：</td>
 <td><input type="text" /></td>
</table>
```

制作者想定义1行2列的表格，但是忘记了<tr>标签，而根据CSS的规定会产生一个匿名的对象，来包含2个单元格以使表格结构完整，类似如下代码，其在浏览器内的显示如图11-6所示。

```
<table>
[开始匿名表格行对象]
 <td>姓名：</td>
 <td><input type="text" /></td>
[结束匿名表格行对象]
</table>
```

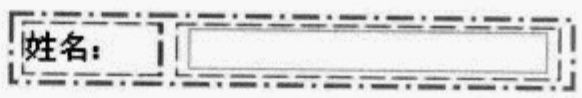
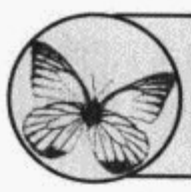


图11-6 匿名表格对象



提示：关于生成匿名表格对象的具体规则，可以参见http://www.w3.org/TR/CSS21/tables.html#anonymous-boxes（英文）。

11.2.4 列

表格的单元格可能受两方面的影响：行和列。单元格是由表格行派生的，而不是列，并不是所有的CSS属性都可以应用在列或者列组元素，而只有下面的属性适用于列和列组元素：

- border：只有当表格元素的border-collapse属性设置为“collapse”时，才适用列的各个边框属性。在这时，设置在列以及列组上的边框会通过解决边框冲突算法从而最终选定每个单元格边界的边框。
- background：该属性设置了列中单元格的背景，但是只有单元格和行设置了透明背景时适用。
- width：该属性给出了列的最小宽度。
- visibility：如果设置一个列的visibility属性为“collapse”，那么该列中所有的单元格都不会被渲染，而延伸到其他列的单元格将被剪裁。另外，表格的宽度也会相应减少该列本应占据的宽度。其他visibility属性的取值没有效果。

11.3 表格的视觉格式化

在视觉格式化模型中，表格可以表现为块级（display: table）或者行内级（display: inline-table）元素。在这两种情况中，表格框都会生成一个匿名框按照文档顺序来包含表格框自身和标题框（< caption>元素）。

11.3.1 匿名框、标题框与表格框

标题框是块级框，它保留自己的内容、补白、边距和边框区域，并且在匿名块内以常规块呈现。如果表格是块级元素，则匿名框是块框，如果表格是行内级元素，则匿名框就是行内块框。在CSS 2.1中，匿名框的宽度由其内部的表格框的边框边决定。表格的width和height属性的百分比值是基于匿名框的包含块，而不是匿名框本身。

匿名框、标题<caption>的框和表格<table>的框之间关系如图11-7所示。

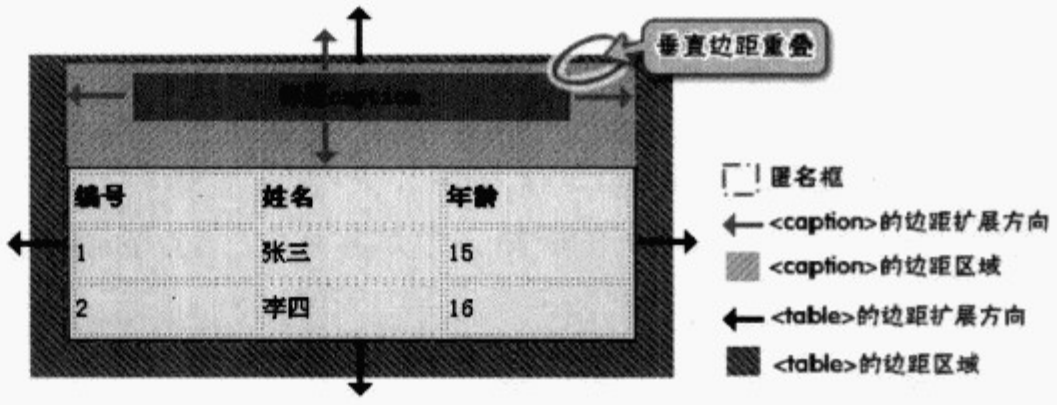


图11-7 CSS 2.1中标题<caption>在上方的标题框、匿名框和表格框的关系

提示： 读者也可参见下载文件包内［/第2部分/第11章：表格/table_model.html］文件。

而在CSS 2中规定，表格和标题框包含自身的内容、边白、边距和边框区域。而匿名框的尺寸是能包含两者的最小尺寸。垂直边距在表格框和标题框相交处重合，如图11-8所示。

而目前比较流行的几款浏览器对于标题框和表格框之间的关系理解各不相同，Opera 9.2遵循CSS 2.1的规定，Firefox 2.0则遵循CSS 2的规定，而IE 7.0及更早的版本不支持<caption>元素的margin属性，因此其显示可能如图11-9所示。

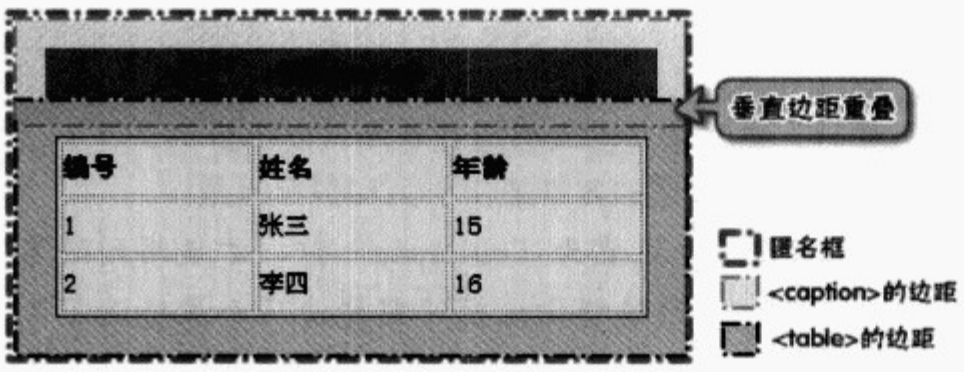


图11-8 CSS 2中标题<caption>在上方的标题框、匿名框和表格框的关系

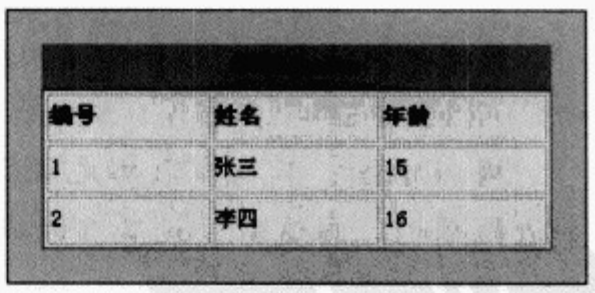


图11-9 IE 7.0及更早的版本不支持<caption>元素的margin属性

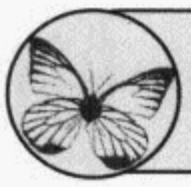
11.3.2 标题<caption>的定位：caption-side属性

无论在文档内标题框放置在表格框的前面还是后面，其位置都是由caption-side属性来决定的。具体定义列表如下：

语法	caption-side : top | bottom | inherit
说明	设置表格中标题<caption>元素的位置
值	top：标题在表格的上方。 bottom：标题在表格的下方
初始值	top
继承性	继承
适用于	“table-caption”类型元素
媒体	视觉
计算值	同指定值

提示：在CSS 2中，caption-side属性还有“left ”和“right”值，表示标题在表格的左边或右边，但是在CSS 2.1中删除了这两个值，在CSS 3中，将以“top-outside”和“bottom-outside”这两个值来替代“left ”和“right”的表现。IE 7.0及更早的版本不支持此属性。

<caption>元素的位置不是由其在文档内的书写顺序决定，而是由caption-side属性来决定，例如下列代码：



提示：本节示例代码，可以参见下载文件包内［/第2部分/第11章：表格/table_model.html］文件。

```
<table summary="样例表格，演示标题的定位。" id="sample4">
 <tr>
 ……
 </tr>
 <caption>
 标题caption
 </caption>
</table>
```

caption

编号	姓名	年龄
1	张三	15
2	李四	16

图11-10　<caption>元素的位置与其书写顺序无关

虽然<caption>元素在表格的尾部，但是其在浏览器内的显示如图11-10所示。

由图11-10可以发现，标题仍然在表格内容之前显示，这是由于caption-side属性的初始值为“top”即上方，如果修改<caption>元素的caption-side属性如下，则其显示如图11-11所示。

```
#sample4 caption {
caption-side : bottom;
……
}
```

caption

编号	姓名	年龄
1	张三	15
2	李四	16

图11-11　修改<caption>元素的caption-side属性为“bottom”后标题在表格下方显示

<caption>元素水平方向的对齐则由text-align属性来决定，例如：

```
caption { text-align : right; }
```

11.3.3 表格内容的视觉布局

表格内的元素会产生具有内容和边框的矩形框，单元格具有补白（padding），但是表格内的元素没有边距（margin）。

读者可以将表格内想象为一些排列整齐的格子，而表格内的元素（行、列、单元格等）生成

的框受这些行列栅格的控制，这些元素生成的框占据1个或者多个栅格，如图11-12所示。

而框所占的栅格数量是由如下规则确定的。

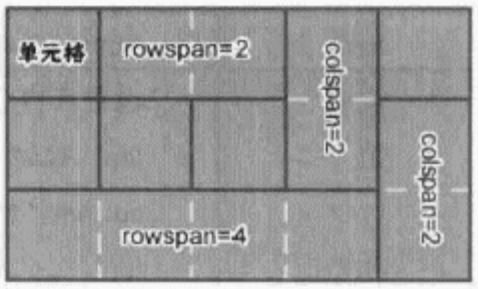
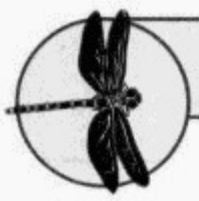


图11-12 表格内的元素与栅格



注意：这些规则不适用于HTML 4.0或更早的版本；HTML有它本身关于行列扩展的限制。

（1）每个行框占据1行的栅格单元格。这些行框在一起以它们在源文档中出现的顺序自顶至底填充表格（也就是说，表格占据了和行元素数量一样多的栅格行）。

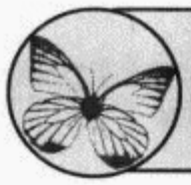
（2）行组占据它包含的行所占据的栅格单元格。

（3）列框占据一个或多个栅格单元格列。列框按照它们出现的顺序挨个排列。第一列可能是在左边也可能是在右边，这取决于表格的direction属性。

（4）列组占据它包含的列所占据的栅格单元格。

（5）单元格可以扩展到若干行或列。每个单元格是一个矩形框，宽度和高度为一个或多个栅格单元格。该矩形的顶行是该单元格父元素指定的行。该矩形必须尽可能靠左（direction属性为“ltr”），但是不可以覆盖任何其他单元格框，并在源文档早于它出现的同行中所有其他单元格之右。

（6）单元格框不可以延伸超出表格或行组的最后一个行框；用户端必须将其缩短直到它合适为止。



提示：对单元格的定位和浮动会导致它脱离表格，因此不建议使用。对于单元格在行和列方向的扩展会造成交叠的情况，CSS并未给出规定。因此用户端可能会自行处理。

11.3.4 表格的层和透明性

为了确定每个单元格的背景，不同的表格元素可以被理解为处在一个6个层次的层面上。在一个层上设置的元素的背景只有当它上面的层都是透明背景时才可见，如图11-13所示。

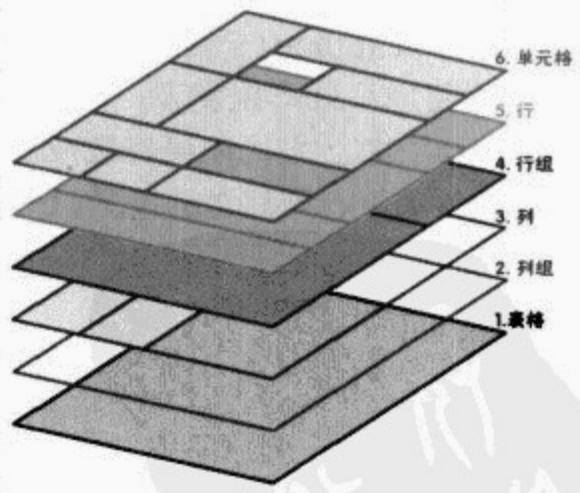


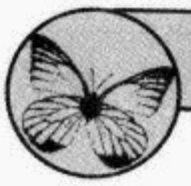
图11-13 表格的层规划

- 最底层是一个单一平面，代表表格框本身。与所有框类似，它可以是透明的。
- 向上一层包含列组。列组与表格同高，水平方向不一定覆盖整个表格。每个列组由它包含的最顶端行的单元格的顶端延伸到最底端行的单元格的底端，左边缘始自其最左边的列，右边缘至最右边的列。背景区域包括整个列组内的原始单元格，即使单元格延展到列组之外，不过此面积的差异不会影响到背景图片的定位。
- 列组之上一层是矩形的列框区域。列与列组同高，而宽度是其包含的普通单元格的宽度。背景区域是列所包含的全部原始单元格的区域，即使单元格延展到列之外，不过此面积的差异不会影响到背景图片的定位。

● 再上一层包含的是行组。每个行组起始于第一列最上面单元格的左上角，结束于最后一列最下面单元格的右下角。

● 再上一层包含行。每行的宽度与行组相同，而高度为其内包含的普通单元格的高度。与列类似，其背景区域包括整个行内的原始单元格，即使单元格延展到行之外，不过此面积的差异不会影响到背景图片的定位。

● 最上层包含单元格。尽管所有的行包含同样数量的单元格，但不是每个单元格都有特定的内容。在CSS 2.1中规定，如果border-collapse属性的值为“separate”，而empty-cells属性为“hide”，则这些“空”的单元格是透明的，可以透过单元格、行、行组，列和列组的背景，让表格的背景显示。而在CSS 2中，空单元格会透出下面一层的背景（如果不是透明的话）。

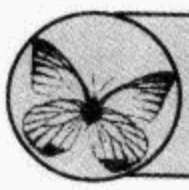


提示：关于表格边框的模式（border-collapse）将在后面的小节内介绍。

在实际情况中，可能存在“缺失的单元格”，它也是一个行及列中的栅格，但是不占用一个元素或伪元素。缺失的单元格犹如一个匿名的单元格框占据自己的栅格位置。

例如下列代码：

```
#sample6 {
background: #FF9;
border: solid black 1px;
empty-cells: hide;
}
#sample6 .row1 {
background: #FC3;
}
#sample6 .row2 {
background: #9CF;
}
#sample6 td {
border: solid black 1px;
}
<table summary="样例表格，表格的层和透明性。" id="sample6">
 <tr class="row1">
  <td>1</td>
  <td rowspan="2"> 2</td>
  <td>3</td>
  <td>4</td>
 </tr>
 <tr class="row2">
  <td>5</td>
  <td></td>
 </tr>
</table>
```



提示：本小节示例代码，读者可参见下载文件包内［/第2部分/第11章：表格/table_model.html］文件。

表格的第1行有4个非空的单元格，而第2行有1个非空的单元格、1个空单元格，同时第1行的单元格延展到了第2行，透过空单元格表格的背景会显示出来，如图11-14所示。

而在IE 6.0/7.0以及Opera 9.2中，其显示如图11-15所示。

空单元格透出的是行（row2）的背景色，而不是表格的背景色。

图11-14 空单元格与缺失的单元格

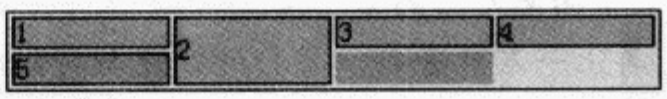


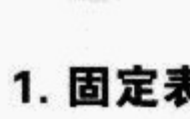
图11-15 IE 6.0/7.0以及Opera 9.2中空单元格的背景

11.3.5 表格宽度算法：table-layout属性

表格的计算宽度与本书［第8章 框模型］中所介绍的有所不同，例如表格的边距为0而width属性为“auto”时，表格不会自动充满包含块的宽度。

CSS中表格的计算有2种方法：固定算法和自动算法，制作者可以通过table-layout属性来定义采用何种布局算法来计算表格的尺寸。table-layout属性具体定义列表如下：

语法	table-layout : auto | fixed | inherit
说明	设置布局表格单元格、行、列的算法
值	fixed：适用固定表格布局算法。 auto：适用自动表格布局算法
初始值	auto
继承性	继承
适用于	“table”和“inline-table”类型元素
媒体	视觉
计算值	同指定值



提示：本小节示例代码，读者可参见下载文件包内［/第2部分/第11章：表格/table-layout.html］文件。

1. 固定表格布局

固定布局算法，表格的水平布局不依赖于单元格的内容；而只依赖于表格的宽度，列的宽度以及边框或单元格的间隔，因此其布局的速度最快。

在固定布局算法中，列的宽度由下列规则决定：

（1）如果列的width属性值不是“auto”，则其宽度为属性设定的值。

（2）否则，列中第一行的单元格的宽度值如果不是“auto”，则列宽等于该值。如果单元格跨越若干列，则这几个列拆分该宽度。

（3）剩下的列平分表格水平方向剩余的空间（减去边框或单元格间隔）。

因此，表格的宽度将是表格元素width属性值和列宽（加上单元格间隔或边框）的较大值。用户端在收到整个第一行后就可以开始表格布局，后续行的单元格不影响列宽。如果单元格内容有溢出，则根据overflow属性确定是否剪裁溢出的内容。如果表格比列宽，多余的空间将分配到所有列中。例如下列代码，其显示如图11-16所示。

```
table { width : 300px; }
td { background : #FF9; }
#fixed1 { table-layout : fixed; }
.col1 { width : 120px; }
.cell1 { width : 30px; }
<table summary="样例表格，table-layout : fixed，列宽的计算。" id="fixed1">
 <caption>
 固定布局算法1
 </caption>
 <col class="col1" />
```

```
  <tr>
   <td>1-1</td>
   <td>1-2</td>
   <td>1-3</td>
   <td class="cell1">1-4</td>
  </tr>
  <tr>
   <td colspan="2">2-1</td>
   <td class="cell2">2-2</td>
   <td>2-3</td>
  </tr>
</table>
```

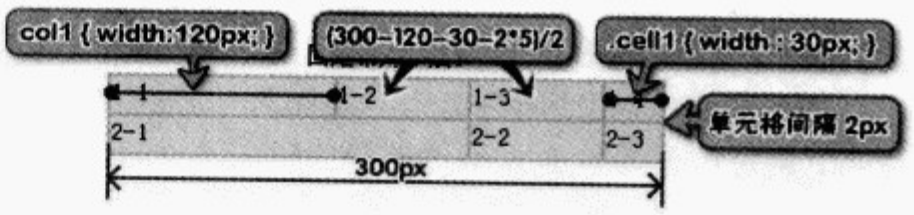


图11-16 固定布局算法的列宽度计算

由图11-16可以发现：

- 第1列定义了宽度120px；
- 最后1列虽然没有定义宽度，但是它包含的第1行的单元格定义了宽度为30px，因此此列宽为30px；
- 虽然第2行第3列的单元格定义了宽度为50px，但是由于第1行的单元格决定列宽度，因此此值被忽略；
- 没有定义宽度的第2和第3列均分表格剩余的宽度（300px-120px-30px-2px*5）/2=70px。

如果所有列宽度总和比表格的width属性值大，则内容会溢出，例如下列代码，其显示如图11-17所示。

```
#fixed2 {
width:200px;
table-layout:fixed;
}
.col1 {
width : 120px;
}
<table summary="样例表格，table-layout : fixed，列宽总合大于表格宽度。" id="fixed2">
 <caption>
 固定布局算法2
 </caption>
 <col class="col1" />
 <col class="col1" />
 <tr>
  <td>1-1</td>
  <td>1-2</td>
  <td>1-3</td>
 </tr>
 <tr>
  <td>2-1</td>
  <td>2-2</td>
  <td>2-3</td>
 </tr>
</table>
```

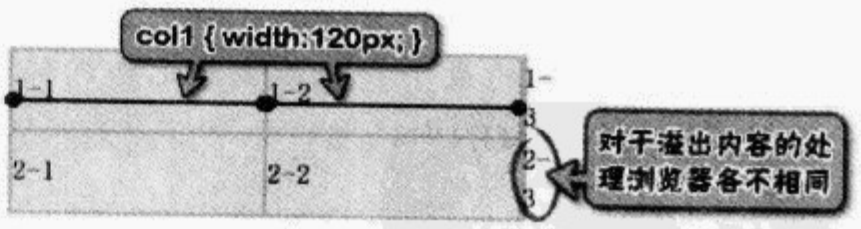
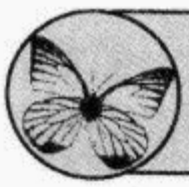


图11-17 列宽总和大于表格的宽度

由图11-17可以发现，虽然表格宽度为200px，但是定义了宽度的列仍然以定义的宽度显示，而由于前2列的宽度和大于200px，因此第3列发生了溢出。



提示： 浏览器在默认状态下对于溢出部分的处理各不相同，读者可以自行在浏览器内测试。

2. 自动表格布局

在本算法中（一般要求不超过两次解析），表格的宽度由列宽（以及其间的边框）给出。与此算法本质相同的HTML表格模式已使用了很多年。

如果表格的width属性值为“auto”，则对目前大多数用户端将触发这个模式，而无论table-layout属性的值是什么。不过如果table-layout为“auto（初始值）”，用户端并不被要求实施该算法来确定表格布局，它们可以使用任何其他算法。

这种算法可能没有固定表格布局那样有效率，因为它要求用户端在确定最终布局前获得所有的表格内容，并可能要求多于一次的解析。也就是说，自动布局要求用户端每得到一个新的单元格就要重新解析整个表格。这通常要求用户端执行某些计算后要再回头对表格进行第二次计算。单元格的内容将被仔细地检查，因为作为HTML表格，表格布局依赖于它。

确定列的宽度可以按以下步骤进行。

（1）计算列中的每个单元格，并得出单元格宽度的最大值和最小值。

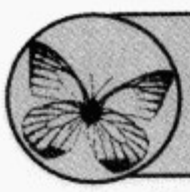
● 确定用以显示内容的最小宽度。内容可能会分布在若干行内，但是不会溢出单元格的框外；如果单元格设定了width属性值，且比这个最小可能宽度大，则单元格的最小宽度等于width属性的值。

● 最大宽度，是内容完全在1行内（除非有显式的分行，例如
）显示时的宽度；如果单元格设定了width属性值，且比这个最大可能宽度大，则最大宽度等于width属性值。

（2）计算每一列宽度的最大值和最小值。

● 列的最小宽度由列中包含的单元格最小宽度值中大的值确定；如果列设定了width属性值且大于这个最小宽度，则最小宽度等于width属性值。

● 列中单元格最大宽度值中大的值决定了列的最大宽度；如果列设定了width属性值且大于这个最大宽度，则最大宽度等于width属性值。



提示：第（2）步中的行为再现了传统的HTML表格中强制扩展列宽以适应其内单元格宽度的行为。

（3）如果单元格跨越了不止1列，则其跨越的这些列的最小宽度的总和，必须等于这个单元格的最小宽度。同样地，这些列的最大宽度和也必须等于这个单元格的最大宽度。在可能的情况下，这些列的值应该是平分的。

这样对于每个列就得到了最大和最小宽度。列宽将影响最终表格的宽度。

● 如果“table”或“inline-table”元素的width属性的计算值（W）不是“auto”，计算出所有列要求的最小宽度加上单元格间隔或边框的总和（MIN），元素width属性的计算值应该是W和MIN中较大的那个值。如果W大于MIN，那么多余的宽度将分配到各列中去。

● 如果“table”或“inline-table”元素width属性值（W）是“auto”，那么计算得到的表格宽度应为MIN和表格包含块的宽度中较大的值。然而，如果列加上单元格间隔或边框要求的最大宽度（MAX）小于包含块的宽度，使用MAX。

例如下列代码，其显示如图11-18所示。

```
.....
.col2 { width:30px; }
.cell1 { width : 30px; }
.cell2 { width : 50px; }
.cell3 { width:100px; }
.cell4 { width:40px; }
```

```
<body>
<table summary="样例表格，table-layout : auto，列宽的计算。">
 <caption>
 自动布局算法1
 </caption>
 <col class="col2" />
 <tr>
  <td>行1列1</td>
  <td>行1列2</td>
  <td>行1列3</td>
  <td class="cell1">行1列4</td>
  <td>行1列5</td>
 </tr>
 <tr>
  <td colspan="2" class="cell3">行2列1</td>
  <td>行2列3，内容</td>
  <td class="cell2">行2列4</td>
  <td class="cell4">行2列5<img src="../../img/ddcat_ad.gif" alt="样例" width="180" height="60" /></td>
 </tr>
</table>
</body>
```

由图11-18可以发现：

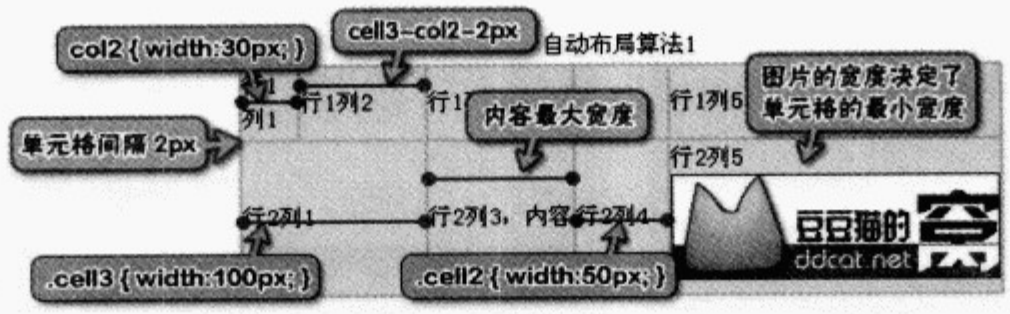
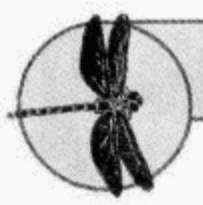


图11-18　表格的自动布局

- width属性和table-layout属性的初始值都是“auto”，因此不用显式设定；
- 第1列定义了宽度值30px，第1行第1列的内容回行；
- 第1列第2行的单元格跨越了第1和第2两个列，其width属性值为100px，因此第2列第1行的单元格宽度为第1列第2行的单元格宽度减去第1列的宽度再减去单元格间隔；
- 第3列第2行的单元格的最大宽度决定了第3列的宽度；
- 第3列中，第2行的单元格定义的width值大于第1行定义的width值，因此第2行的单元格宽度决定列宽即50px；
- 第5列第2行的内容含有图片，图片的内在尺寸大于单元格设定的width属性值，因此单元格的宽度为图片的内在宽度值；
- 表格的宽度为各列的宽度与单元格间隔的总和。

需要注意的是，此时表格的父元素<body>的宽度足够大，也就是表格的包含块的宽度比表格的总宽度宽。如果缩小浏览器窗口的宽度小于表格各列宽度总和的时候，则列的宽度将被压缩，表格可能如图11-19所示。

由图11-19可以发现，首先被压缩的列是宽度为“auto”（第3列），然后是包含宽度为“auto”的单元格的列（第2列），如果宽度仍大于包含块宽度，设定了width值的列也可能被压缩，直至表格所有单元格都被压缩到最小宽度为止，如图11-20所示。



注意：含有图片的单元格其最小宽度就是图片的宽度。

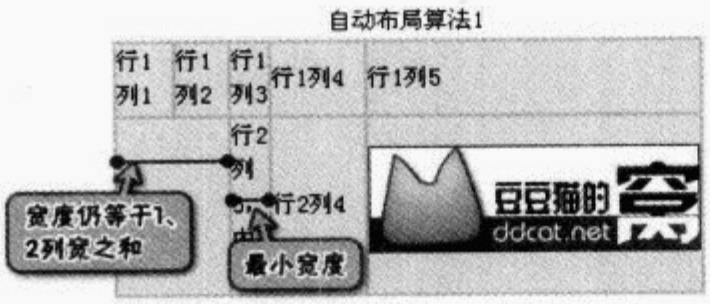


图11-19 包含块宽度小于表格列宽度和时将压缩列的宽度

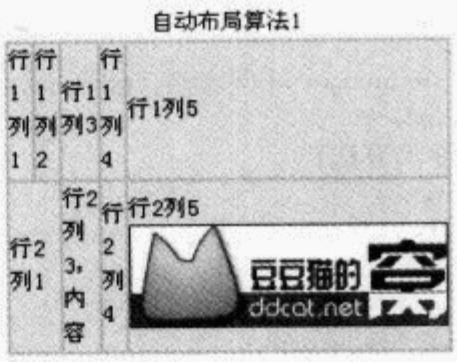


图11-20 表格宽度被压缩到最小

如果设定列宽为百分比值，则相对于表格的宽度。如果表格设置了“width : auto”，那么百分比表示了列宽的约束，用户端应该尝试满足该约束（当然并不是总能满足：如果列宽是110%，那么该约束是无法满足的）。用户端需要储存百分比值并在算法的未来部分利用它，此时，用户端将决定每列的宽窄。例如下边代码：

```
<div class="wrap">
<table summary="样例表格，table-layout : auto，列宽的计算。">
 <col class="col3" />
 <tr>
  <td>行1列1</td>
  <td>行1列2</td>
  <td>行1列3，内容</td>
  <td>行1列4</td>
  <td>行1列5</td>
 </tr>
 <tr>
  <td colspan="2">行2列1</td>
  <td>行2列3</td>
  <td>行2列4</td>
  <td>行2列5</td>
 </tr>
</table>
</div>
```

如果设置表格的width如下，则其显示如图11-21所示。

```
.wrap { width : 500px; }
.col3 { width : 20%; }
table { width : auto; }
```

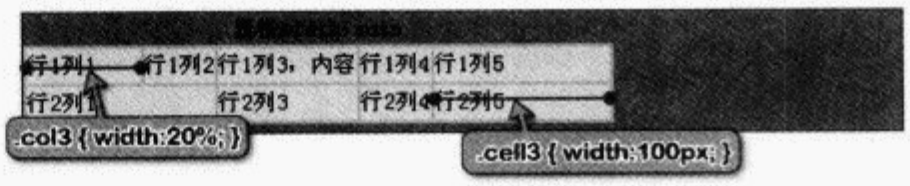


图11-21 宽度为“auto”时百分比列宽的显示

由于表格的宽度为“auto”，其列宽度如果不指定特定的width值，则按内容宽度计算，因此可以计算出表格的宽度和百分比列的宽度（即2~5列的总宽度+单元格间隔=80%表格宽度）。如果修改表格的宽度如下，则其显示如图11-22所示。而如果为表格设定一个过小的宽度如下，则其显示如图11-23所示。

```
table { width : 90%; }
```

由图11-23可以发现，表格的列宽都被压缩到最小宽度，包括百分比宽度的列。

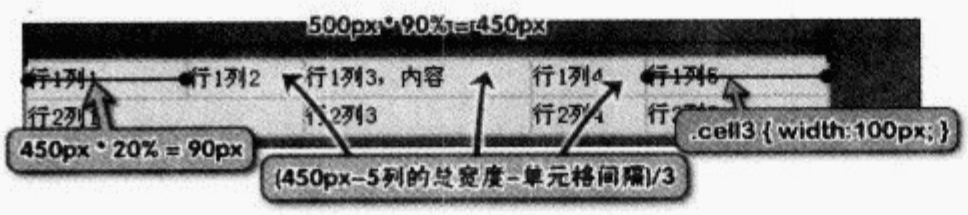


图11-22 表格width值大于表格各列宽度和时表格宽度为设定值

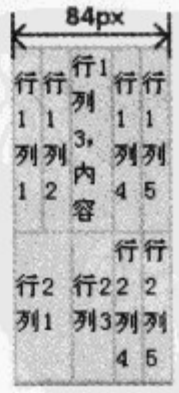
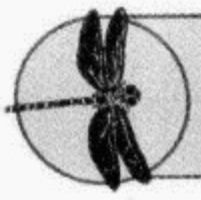


图11-23 过小的width值使表格被压缩



注意：在本算法中，行（及行组）和列（及列组）都约束了也受约束于它们包含的单元格的尺寸，设置列宽可能间接影响到表格行的高度，反之亦然。

11.3.6 表格高度

表格的高度由“table”或“inline-table”元素的height属性给出。如果取值为“auto”则意味着表格的高度为所有表格行的高度加上单元格间隔和边框宽的总和。

如果设定了表格的高度值，也意味着，一个表格的height属性值可能比其内包含的行高度的总和要高或者少。虽然CSS 2.1规定了，heigth属性任何非“auto”的值都将被视为最小高度，但是CSS 2.1并没有明确定义这种情况下该如何处理，用户端可以扩大或缩小一个行，以配合其高度，或在表格框内留下空位，或是完全不同，这由每个用户端来决定。

确定高度的过程和确定宽度的过程其实很相似。为每个单元格确定最大高度和最小高度。在CSS 2.1中，单元格框的高度是单元格的height属性值和内容要求的最小高度中大的那个值，如果height属性值为“auto”，则将使用内容的最小高度值布局。

表格行的高度为本行内设定的height属性值和单元格最小高度（MIN）中最大的值。如果表格行元素height属性为“auto”，则使用MIN值来确定其布局高度。MIN依赖单元格框的高度和垂直对齐方式（很类似于行框的高度计算）。

提示：关于垂直对齐方式对高度的影响将在下一小节内介绍。

最后根据各行的高度、单元格间隔和边框宽度计算表格的高度。除了上面介绍的计算方法，对于以下几点，CSS 2.1中并未给出解决方法：

- 单元格的百分比高度的效果；
- 表格行和行组的百分比高度的效果；
- 当单元格跨越多行时如何影响行的高度的计算（但是要求相关行高度的和必须足以涵盖跨行的单元格）。

由此可见，表格高度的计算主要是由用户端来决定，而客户端处理的方式可能存在着差异，因此在制作表格的过程中，应该尽量避免设置height值。例如下列代码：

```
#height1 {
width:300px;
height:100px;
}
.row1 {
height:30px;
}
<table summary="样例表格，表格高度的计算。" id="height1">
 <caption>
 表格高度
 </caption>
 <tr class="row1">
  <td>行1列1，内容内容内容内容</td>
  <td>行1列2</td>
  <td>行1列3</td>
 </tr>
 <tr>
  <td>行2列1</td>
  <td>行2列2</td>
  <td>行2列3</td>
 </tr>
</table>
```

Windows IE 7
表格高度

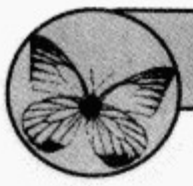
行1列1，内容内容内容内容	行1列2	行1列3
行2列1	行2列2	行2列3

Firefox 2.0 / Opera 9.2
表格高度

行1列1，内容内容内容内容	行1列2	行1列3
行2列1	行2列2	行2列3

图11-24　不同浏览器对于表格高度的不同处理方式

表格设置了高度100px，而第1行设置了高度30px，在不同浏览器内的显示如图11-24所示。



提示： 在将来版本的CSS可能会指定这些问题的解决方案。

11.3.7 单元格内容的对齐

单元格内容的对齐方式包括水平方向和垂直方向。本节示例代码，读者可参见下载文件包内［/第2部分/第1 1章：表格/table_align.html］文件。

1. 水平方向

在CSS 2.1中，单元格被当作块级框，其内容的水平方向对齐方式由text-align属性决定。关于text-align属性，读者可参见本书［7.1文本水平对齐：text-align属性］一节。

在CSS 2中，除了text-align属性允许的值以外，单元格的内容还可以根据某个字符对齐，例如以下代码，不过在CSS 2.1中已经取消了这一设定。

```
td { text-align : "."; } /* 以英文"."对齐 */
```

2. 垂直方向

单元格的垂直对齐方式也是由vertical-align属性设定，在前一小节对表格的高度的计算中，曾经提到单元格垂直方向的对齐方式，可能会影响到表格行的高度。关于vertical-align属性，读者可以参见本书［7.4 垂直对齐：vertical-align属性］一节。

每个单元格的内容都具有基线、顶线、中线和底线，而其又与单元格所在行的基线、顶线、中线和底线有关。

- “vertical-align : top”：单元格内容的顶线和它所在的表格行的顶线对齐，如果单元格跨越多行，则与其所跨越的行中第1行的顶线对齐。
- “vertical-align : bottom”：单元格内容的底线与所在表格行的底线对齐，如果跨越多行，则与所跨越行的最后一行的底线对齐。
- “vertical-align : middle”：初始值，单元格内容的中线与所在的表格行的中线对齐；如果单元格跨越多行，则以所有行的总和的中线对齐。
- “vertical-align : baseline”：单元格的基线与所在表格行的基线对齐。

“sub”、“super”、“text-top”和“text-bottom”这些值对单元格不适用，单元格采用基线对齐。单元格的基线由其包含的在流内（常规流向）的第一个行框或者表格行决定，如果没有行框或者表格行则基线为其内容边的底边；而同一表格行内所有设定了基线对齐的单元格中单元格框顶到基线的最大距离将用来设置本表格行的基线。

单元格的对齐按照如下顺序进行。

（1）首先定位基于基线对齐的单元格，这样就可以得到该行的基线。接着，定位“vertical-align: top”的单元格。

（2）此时，表格行具有了顶线，可能还具有基线（要看是否有基于基线对齐的单元格），同时还有一个临时的高度——是当前定位的单元格顶到最低的底的距离。

（3）如果剩余的单元格中，有对齐方式是底线或中线，而高度比目前的表格行的高度要大的单元格，则此表格行的高度将通过降低底线的方式来达到这些单元格中最大的高度。

（4）最后，定位这些剩余的单元格。

单元格框高度如果小于行的高度，会自动加入上下补白，以保证每行的单元格高度是一致的。例如下列代码，其显示如图11-25所示。

```
.align1 { line-height : 2; font-size : 12px; }
.align1 td { width : 60px; }
.align1 td span { display : block; background : #CF6; }
```

```
.a1_r1c1, .a1_r1c3 { vertical-align : baseline; }
.a1_r1c2 { vertical-align : baseline; font-size : 20px; }
.a1_r1c4 { vertical-align : top; }
.a1_r1c5 { vertical-align : middle; }
.a1_r1c6 { vertical-align : bottom; }
<table summary="样例表格，表格的垂直对齐" id="align1" class="sample1">
 <caption>
 表格单元格的基线与行的基线
 </caption>
 <tr>
  <td class="a1_r1c1">baseline, 行高12px*2=24px</td>
  <td class ="a1_r1c2">baseline, 行高20px*2=40px</td>
  <td class ="a1_r1c3"><img src="../../img/ddcat_ad2.gif" alt="样例" width="60" height="60" /></td>
  <td class ="a1_r1c4"><span>垂直对齐, top</span></td>
  <td class ="a1_r1c5"><span>垂直对齐, bottom</span></td>
  <td class ="a1_r1c6"><span>垂直对齐, middle</span></td>
 </tr>
</table>
```

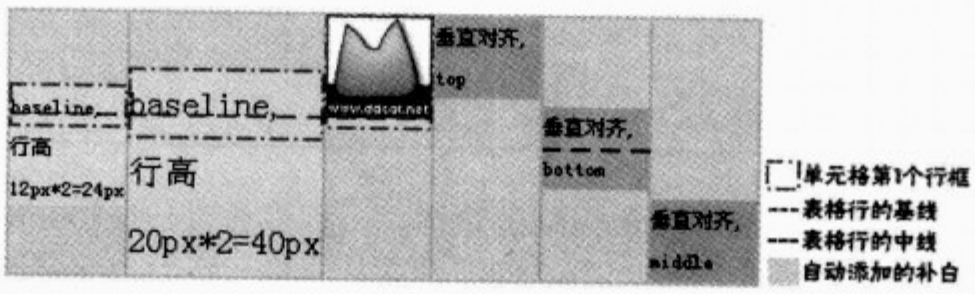
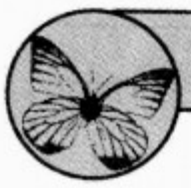


图11-25　表格行和单元格基线的关系



提示：关于行框、基线与行高，参见本书［7.3 行高：line-height属性］一节。

由图11-25可以知道：

● 表格的行高为缩放因子，单元格继承的行高会根据字体大小来计算，因此第1个单元格和第2个单元格的行高实际为24px和40px；

● 第3个单元格内的图片将本单元格的第1个行框高度撑开，因此其基线到单元格顶线的距离最大，第1行的基线以这个距离为准；

● 第4至6个单元格不是基线对齐，因此对表格行的基线无影响。

图片本身也有垂直对齐（初始值为“baseline”），如果改变图片的垂直对齐方式，也会影响到表格行的基线位置，例如为第3个单元格的图片添加CSS规则如右，其显示如图11-26所示。

```
.a1_r1c3 img { vertical-align : top; }
```

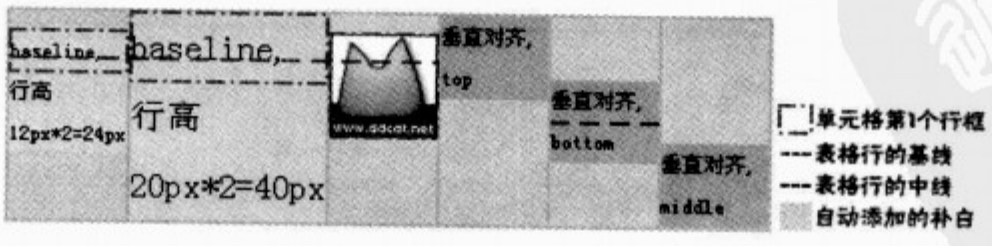


图11-26　改变图片的对齐方式对表格行基线的影响

由图11-26可以发现，图片与行框顶对齐，而行高为24px，因此本行基线的高度改由第2个单元格的基线到单元格顶部的距离决定。同时，表格行和整个表格的高度都发生了变化。

在IE 7.0及更早的版本中，对于表格高度的处理稍有不同，其显示如图11-27所示。IE 6.0及更早的版本存在行高失效的问题，请参见本书［7.3.4 浏览器的差别与错误］一节。

由图11-27可以发现，表格行的高度由内容最高的第2个单元格决定，而基线对齐并没有对该单元格生效。如果增加单元格的高度如下，则在IE内显示如图11-28所示。由图11-28可以发现，IE对于行内图片的对齐并没有按CSS规范处理。

```
.sample1 td { height : 160px; }
```

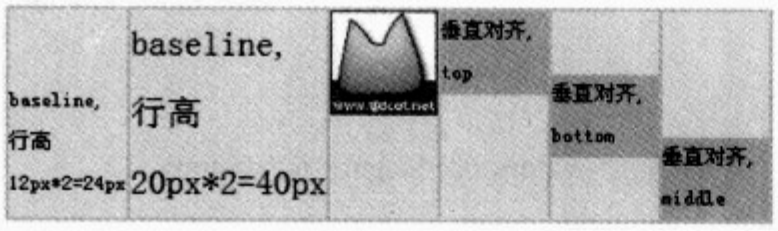


图11-27 IE 7.0对表格高度的处理

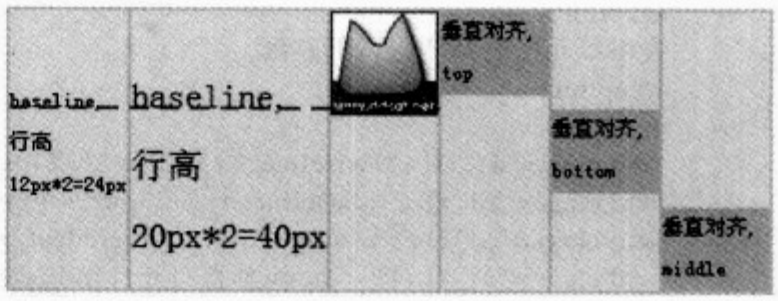


图11-28 增加表格的高度在IE 7.0内的基线对齐

11.4 单元格边框：border-collapse属性

CSS中有两种完全不同的方法设置单元格的边框。一种最适合于在独立的单元格中设置分离的边框（即单元格的边框之间有空隙），另一种则设置从表格一端到另一端的连续边框。很多边框样式在两个方法中都可以实现。

border-collapse属性设定按哪种方式显示边框，其具体定义列表如下：

语法	border-collapse : collapse | separate | inherit
说明	设置边框显示的方式
值	collapse：重合的边框模型。 separate：分离的边框模型
初始值	separate
继承性	继承
适用于	“table”和“inline-table”类型元素
媒体	视觉
计算值	同指定值

提示： 本节示例代码，可以参见下载文件包内［/第2部分/第11章：表格/table_border.html］文件。

为<table>元素设定边框，则只会在整个表格的四周形成边框，而不会影响到单元格，例如下列代码，其显示如图11-29所示。

```
. sample1 {
border:2px solid #F60;
……
}
. sample1 td {
border: 2px solid #39F;
……
}
<table summary="样例表格，表格元素的边框与单元格的边框" class="sample1">
```

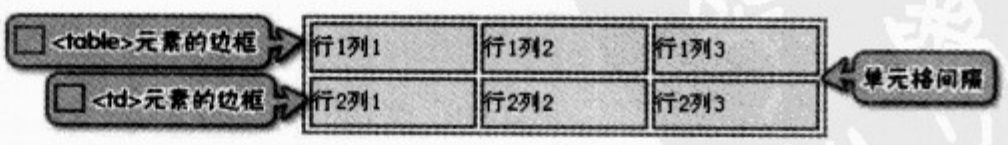


图11-29 表格元素与单元格元素的border属性彼此无关

```
    <tr>
     ……
    </tr>
    <tr>
     ……
    </tr>
  </table>
```

由图11-29可以发现，表格边框与单元格边框、单元格与单元格边框间存在着间隙，这就是“separate（分离）”的边框模式。而如果修改<table>元素的CSS规则如下，其显示如图11-30所示。

行1列1	行1列2	行1列3
行2列1	行2列2	行2列3

图11-30　“border-collapse:collapse”的单元格边框

```
.sample1 { border-collapse: collapse; }
```

由图11-30可以发现，单元格边框之间的间隙消失了，同时表格的边框也不见了，造成这种现象的原因，将在下面的［11.4.2 重合的边框模型］一节内详细介绍。

11.4.1 分离的边框模型

border-collapse属性的初始值就是“separate”，因此如果不特别设定，表格就是以分离的边框模型布局，这种模型中，每个单元格有一个独立的边框。

1. 单元格的间隙：border-spacing属性

border-spacing属性指定了相邻单元格边框间的距离，这个空间由表格元素的背景填充。border-spacing属性具体定义列表如下：

语法	border-spacing : <长度> <长度>? | inherit
说明	设置分隔相邻的单元格之间的距离
值	长度：如果只有1个长度，就指定了水平和垂直的间距。如果给出2个长度，第一个指定水平间距，而第二个指定垂直间距。长度不可以是负数，不能为百分比值
初始值	0
继承性	继承
适用于	“table”和“inline-table”类型元素
媒体	视觉
计算值	2个绝对长度值

border-spacing属性的值可以是1个，也可以是2个，例如：

```
table { border-spacing : 10px; } /* 相邻的单元格水平和垂直方向的间距都是10px。 */
table { border-spacing : 20px 10px; } /* 相邻的单元格水平方向间距20px，垂直方向间距10px。 */
```

在表格<table>的边框与单元格边框之间的距离，是单元格间隙与表格的补白的和，表格的宽度是表格左补白的内边缘到右补白的内边缘之间的距离，包括单元格间隙。然而，在HTML和XHTML1中<table>元素的宽度是左边框边到右边框边的距离。例如下列代码，其显示如图11-31所示。

```
#border3 {
width : 240px;
padding : 10px;
background : #FC3;
border : 5px solid #930;
border-spacing : 10px;
}
#border3 td{
height : 50px;
border : 5px solid #6CF;
```

```
}
<table summary="样例表格，分离的边框模型，border-collapse : separate，表格的宽度" id="border3">
 <caption>
 表格的宽度
 </caption>
 <tr>
  <td>行1列1</td>
  <td>行1列2</td>
 </tr>
 <tr>
  <td>行2列1</td>
  <td>行2列2</td>
 </tr>
</table>
```

单元格边框之间的空间，表格行、列、行组和列组的背景是不可见的，而是显示出表格的背景。行、列、行组和列组不具有边框，即用户端会忽略这些元素的边框属性。

但是IE 7.0及更早的版本并不支持表格元素的padding和border-spacing属性，其单元格的间隙是固定的2px，如图11-32所示。

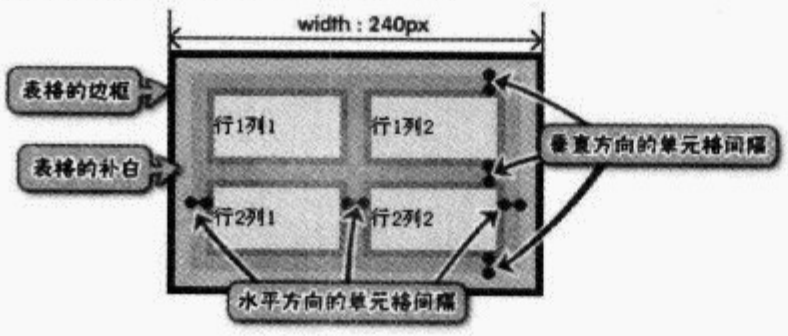


图11-31 HTML和XHTML1中<table>元素的宽度

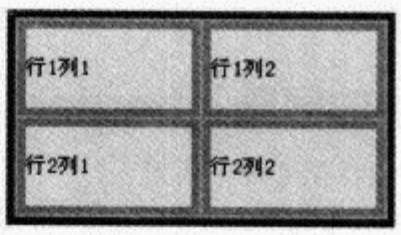


图11-32 IE 7.0及更早的版本不支持表格的padding和border-spacing属性

2. 空单元格的边框和背景：empty-cells属性

empty-cells属性具体定义列表如下：

语法	empty-cells : show | hide | inherit
说明	设置分隔相邻的单元格之间的距离
值	show：显示空单元格的边框和背景。 hide：不显示空单元格的边框和背景
初始值	show
继承性	继承
适用于	“table-cell”元素
媒体	视觉
计算值	同指定值

在分离的边框模型中，empty-cells属性控制没有可视内容的单元格周围的边框和背景的绘制，空的单元格和visibility属性为“hidden”的单元格属于没有可视内容。除非含有一个或一个以上的下列内容，否则单元格就是空的：

- 浮动的内容（包括空元素）；
- 在流内的内容（包括空元素），除了已经按照white-space属性压缩处理的空格。

如果empty-cells属性值为“show（初始值）”，则空白单元格周围会画出边框和背景（和一般的单元格一样）。如果empty-cells属性值为“hide”，则空单元格将没有边框和背景。

例如下列代码，其显示如图11-33所示。

```
#border4 {
border : 2px solid #F90;
```

```
background : #CFC;
}
#border4 td{
background : transparent; /* 单元格透明，可以看到下面的背景 */
border : 3px solid #06F;
height : 30px;
width : 100px;
}
#border4 .spl1 {
background : #F9C;
}
#border4 .spl2 {
empty-cells : hide;
}
<table summary="样例表格，分离的边框模型，empty-cells属性"id="border4">
 <tr>
  <td>行1列1</td>
  <td> </td> <!--  表示半角空格 -->
 </tr>
 <tr class="spl1">
  <td></td>
  <td class="spl2"> </td> <!--空格将被压缩，因此此单元格为空 -->
 </tr>
</table>
```

由图11-33可以发现，第2行的两个单元格都是空单元格，第1个单元格为“show（初始值，不用显式设定）”，因此有边框，并且由于单元格背景透明，因此可以看到其所在行<tr>的背景；第2个单元格为“hide”，则它不仅没有边框，其所在行的背景也不显示，而是显示表格<table>元素的背景。如果表格行内所有的单元格都是空单元格，而且empty-cells设定了“hide”，则此行会被当作如同设置了“display:none”一样处理。

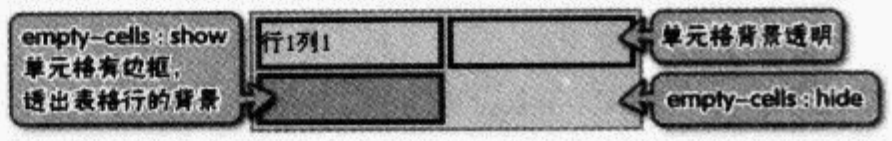
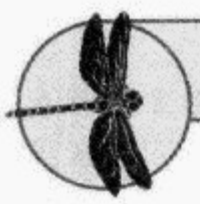


图11-33 empty-cells属性的表现



注意：虽然CSS是如此规定，但是浏览器可能不会如此处理，而只是不显示边框和背景。

不同的浏览器的显示可能不仅相同，例如Opera将显示行的背景（CSS 2规范），如图11-34所示。而IE 7.0及更早的版本不支持此属性，其显示如图11-35所示。

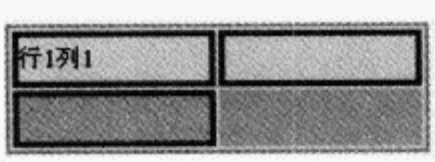


图11-34 Opera 9.2中对“empty-cells:hide”的显示

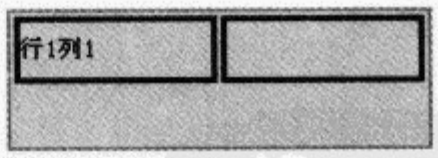


图11-35 IE 7.0及更早的版本不支持empty-cells属性

右边的规则可以设定所有的单元格都有边框和背景：

```
table { empty-cells: show }
```

11.4.2 重合的边框模型

设置表格元素的border-collapse属性为“collapse”时，表示使用重合的边框模型，在此模型中，单元格之间没有间隙，因此情况也变得比分离边框模型复杂。

在本模型中，有以下的几点需要注意：

- display属性为“table”或“inline-table”的元素不具有补白，但是可以有边距；

● 边框可以应用于单元格、行、行组、列、列组以及表格；

● 单元格的边框之间没有间隙，因为边框之间发生重叠，因此只有一个边框会被采用。这点类似于边距的重叠，结果是大的边距；

● 边距的重叠，居中于假定的栅格线。

1. 重合边框布局

在重合边框模型中，单元格边框居中于单元格之间的栅格线。图11-36显示了表格宽度、边框宽度、补白和单元格宽度的关系。

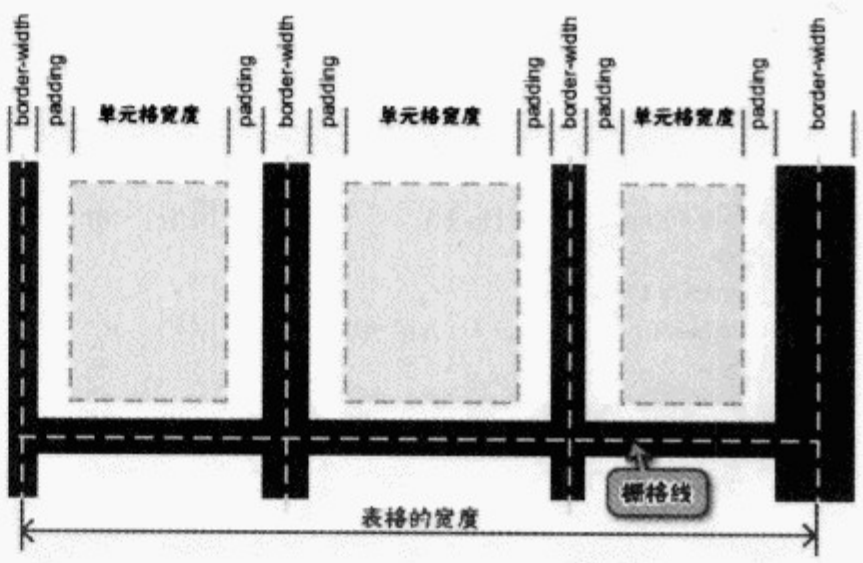


图11-36 单元格、边框以及单元格边白的关系

相邻的单元格之间，只能存在1个边框，而不是2条边框的总和或者一边一半，例如左右相邻的2个单元格中间的边框，要不是左边单元格的右边框，要不然就是右边单元格的左边框，要看哪边胜出。

由图11-37可以发现，表格元素的边框将取1/2的宽度计算在表格宽度内，而另外1/2则属于表格边距的范围。表格行宽度的计算公式如下：

$$\text{表格行宽} = (0.5 \times \text{border-width}_0) + \text{padding-left}_1 + \text{width}_1 + \text{padding-right}_1 + \text{border-width}_1 + \text{padding-left}_2 + \ldots + \text{padding-right}_n + (0.5 \times \text{border-width}_n)$$

其中n为表格行内单元格的数量，padding-lefti和padding-righti指第［i］个单元格的左右补白，border-widthi指第［i］个单元格与第［i+1］个单元格之间的边框宽度。

当开始重合边框布局的时候，用户端会为表格的左右边框计算一个初始值，取第1行的第1个单元格的左边框宽度的一半作为表格左边框的初始值，取第1行最后1个单元格的右边框的一半为表格右边框的初始值。其后的行的左右边框如果比这个初始值宽，则扩展到表格的边距区域内。

计算所有和表格上边框发生重合的单元格的上边框宽度，其中最大值的一半为表格上边框的宽度初始值。同理，表格下边框宽度的初始值是所有和表格下边框发生重合的单元格的下边框宽度最大值的一半。用户端（如屏幕像素点、打印机点阵等）会用某种规则来取整处理间距为奇数单位的情况。

2. 边框的重合

由于单元格、行、行组、列、列组以及表格本身都可以设定边框属性，而表格、行、列、单元格之间又是嵌套的关系，因此一个单元格的各个边上的发生重叠的这些元素的边框有可能会被设定不同的宽度、样式和颜色。

在前面介绍过，2个相邻单元格之间边框的重叠只能显示1个边框，而不是几个边框的综合，那么究竟该显示哪条边框？准则就是，在每条边上最“吸引眼球”的边框样式被选中，除非有任何一个“hidden”样式而无条件地将边框关闭。具体判断规则如下。

（1）如果边框的border-style属性是“hidden”，则它将获得最高优先级，而此位置的边框将隐藏。

（2）样式为“none”的边框的优先级最低，只有所有参与重叠的元素边框的样式都是“none”，边框才会被省略（不过要注意“none”是边框样式的默认值）。

（3）如果重合的边框中，没有样式是“hidden” 的边框，而且，至少有1个边框的样式不是“none”，那么相对较窄的边框会被忽略。如果若干个边框的宽度一样，那么样式的优先次序如下：“double”、“solid”、“dashed”、“dotted”、“ridge”、“outset”、“groove”和最低的“inset”。

（4）如果边框样式只有颜色上的区别，那么单元格的样式优先，然后是行的、行组的、列的、列组的，最后是表格的。

（5）如果2个元素的类型也相同，则文档位置在后面的元素的左（上）边框优先。

例如下列代码，其显示如图11-37所示。

```
.sample2 {
border-collapse:collapse;
width:340px;
border:4px solid #F90;
margin:20px;
}
.sample2 td{
border-style:solid;
border-width:10px;
border-color:#06F #390 #939 #C90;
padding:15px;
}
```

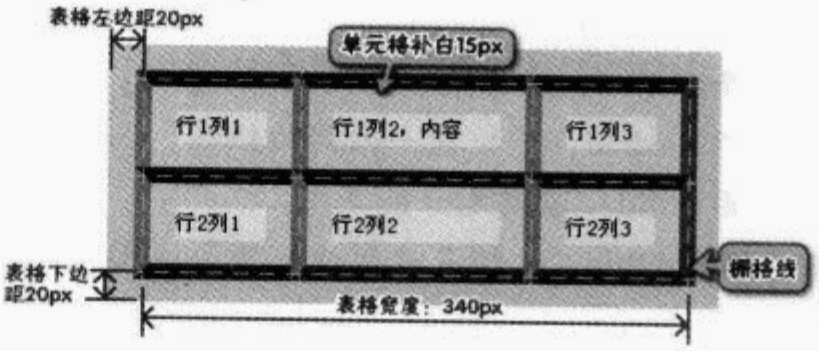


图11-37　重合的边框只有颜色不同时的选择

```
<table summary="样例表格，重合的边框模型，边框重合原理" class="sample2">
 <tr>
  <td>行1列1</td>
  <td>行1列2，内容</td>
  <td>行1列3</td>
 </tr>
 <tr>
  <td>行2列1</td>
  <td class="spl4">行2列2</td>
  <td>行2列3</td>
 </tr>
</table>
```

由图11-37可以发现，由于单元格边框宽度和样式一样，因此左右相邻的2个单元格中右边单元格的左边框优先于左边单元格的右边框显示，上下相邻的单元格，下面单元格的上边框优先于上面单元格的下边框显示。

如果调整边框的宽度值如下，则其显示如图11-38所示。

```
.sample2 td { border-width: 10px 16px 16px 10px; }
```


图11-38　改变边框宽度后边框优先权改变

由图11-38可以发现，由于单元格右边框的宽度为16px比左边框10px宽，因此左右相邻的单元格显示的是左边单元格的右边框，上下相邻的单元格，显示上面单元格的下边框。

而如果再修改CSS规则如下，其显示如图11-39所示。

```
.sample2 { border-style:hidden; }
```


图11-39　设定了“border-style:hidden”的边框优先级最高

在前2个例子中，由于表格元素的边框宽度比单元格小，因此不显示，而当设定了表格的边框样式“border-style : hidden”，则它具有了最高级，因此与表格元素边框重合的单元格的边框全部隐藏。

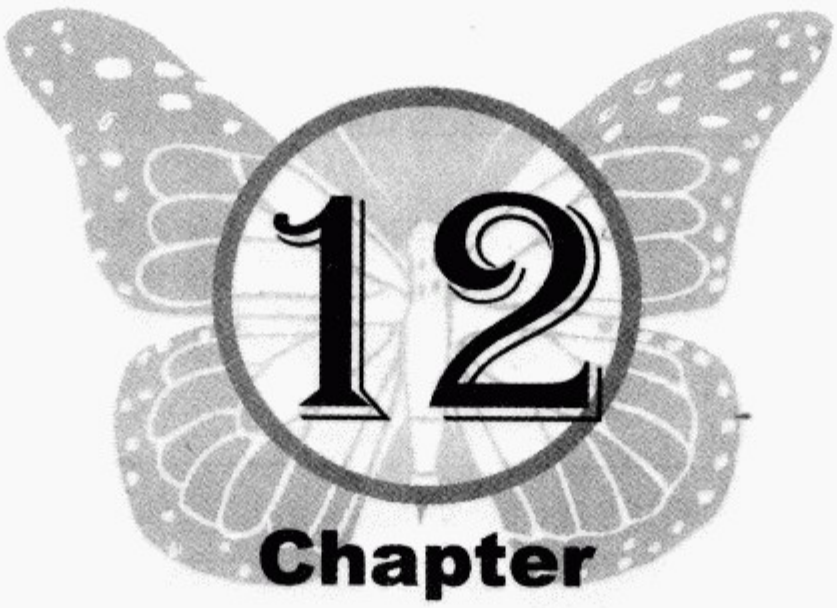
注意：IE 7.0及更早的版本并没有按照此规则来显示重合的边框，读者可以自行测试以观察其表现。

11.4.3 边框样式

某些border-style的取值在表格中的含义和在其他元素中的含义不同，以下将用粗体标识。

- none：没有边框。
- **hidden**：等于“none”，但是在重合边框模型中会同时抑制其他边框。
- dotted：边框是一系列的点。
- dashed：边框是一系列的短线。
- solid：边框是一条实线。
- double：边框是双实线，两条线及只见空白的距离之和等于'border-width'的值。
- groove：边框看起来像是嵌入绘画平面。
- ridge：和“grove”相反，边框看起来像是突出于绘画平面。
- **inset**：在分离边框模型中，边框使整个框看起来像是嵌入绘画平面。在重合边框模型中，等于“groove”。
- **outset**：在分离边框模型中，边框使整个框看起来像是突出于绘画平面。在重合边框模型中，等于“ridge”。

说明：关于border-style属性的详细介绍，可以参见本书［8.8.3 边框样式］一节。



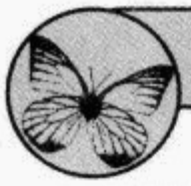
第12章
列表和生成的内容

列表（<ul>或者<ol>）是经常会用到的元素，而<ol>前面的数字编号或者<ul>前面的圆点就是用户端生成的内容，而不需要制作者手工输入这些内容。同时，利用:before及:after伪元素还可以在其他元素之前或者之后自动生成制作者指定的某些内容。

12.1 列表

从某种意义来讲，很多内容都可以归类于某种列表，比如导航栏、每日更新内容标题列表、排行榜、某个栏目内文章列表、注册用户列表等。CSS 2.1提供的是列表的基本视觉格式化模型。

display属性为“list-item”的元素其内容生成一个主块框和一个可选择的标记框，以在视觉上表明这个元素是列表项。



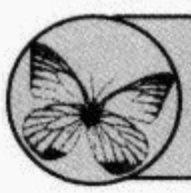
提示： 关于“主块框”，请参见本书［9.1.1 块框的生成（block box）］一节。

通过设定CSS的列表属性，可以指定列表的标记类型（图片、圆点或者数字等）和标记在主块框内的定位等（在主块框外还是在内）。但是不允许作者设定标记的风格（颜色、字体、对齐方式等），或者调整标记在主块框内的具体位置等。背景属性只作用于主块框，而在主块框外生成的标记框将是透明的。

12.1.1 列表样式类型：list-style-type属性

列表样式类型（list-style-type）属性设定列表项的标记的样式，CSS 2.1和CSS 2中的值有所不同，list-style-type属性具体定义列表如下：

语法	list-style-type : <关键字（见下）> | none | inherit
说明	设置列表项的标记样式类型
值	CSS2.1 : disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | none | inherit CSS2: disc | circle | square | decimal | decimal-leading-zero | upper-alpha | lower-alpha | upper-roman | lower-roman | lower-greek | hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana | hiragana-iroha | none | inherit
初始值	disc
继承性	继承
适用于	display属性为“list-item”的元素
媒体	视觉
计算值	同指定值



提示： 本小节示例代码，读者可以参见下载文件包内［/第2部分/第12章：列表和生成的内容/list-style-type.html］文件。

list-style-type属性指定的列表项的标记只有在list-style-image属性（列表样式图片）值为“none（无图片）”或者指定的图片的URI错误而无法显示的时候才会显示。关于list-style-image属性，请参见本章［12.1.2 列表样式图片：list-style-image属性］一节。

而列表项标记的样式可以分为3类：符号、数字编号和字母编号。“none”表示不使用标记。

CSS 2.1的符号类关键字如下：

关键字	效果	说明
disc	实心的点	符号“disc”、“circle”和“square”的具体外观依赖于用户端
circle	空心的点	
square	实心或空心的方块	

CSS 2.1的数字编号类关键字如下：

关键字	效果
decimal	从1开始的整数（1, 2, 3, 4, 5, ...）
decimal-leading-zero	整数（01, 02, 03, ..., 98, 99）
lower-roman	小写罗马数字（i, ii, iii, iv, v, ...）
upper-roman	大写罗马数字（I, II, III, IV, V, ...）
georgian	传统格鲁吉亚数字
armenian	传统的亚美尼亚数字

CSS 2.1的字母编号类关键字如下：

关键字	效果
lower-alpha、lower-latin	小写ASCII字母（a, b, c, d, e, ...z）
upper-alpha、upper-latin	大写ASCII字母（A, B, C, D, E, ...Z）
lower-greek	小写希腊古典符号（, , , ...）

CSS规范没有定义字母系统在字母表之后如何回绕。例如，26个列表项之后，“lower-latin”的渲染未被定义。因此，对于长的列表，推荐使用数字编号。

注意： IE 7.0及更早的版本并不完全支持这些关键字，不支持的有：decimal-leading-zero、georgian、armenian、lower-latin、upper-latin和lower-greek。

而在CSS 2规范中，还包含有下表所列的关键字，但是这些关键字并未包含在CSS 2.1规范内。

关键字	效果
hebrew	传统的希伯来数字
cjk-ideographic	表意文字数字
katakana	日本数字编号（A、I、U、E、O、KA、KI…）
katakana-iroha	日本数字编号（I、RO、HA、NI、HO...）
hiragana	日本数字编号（a、i、u、e、o...）
hiragana-iroha	日本数字编号（i、ro、ha、ni、ho...）

其显示如图12-1所示。

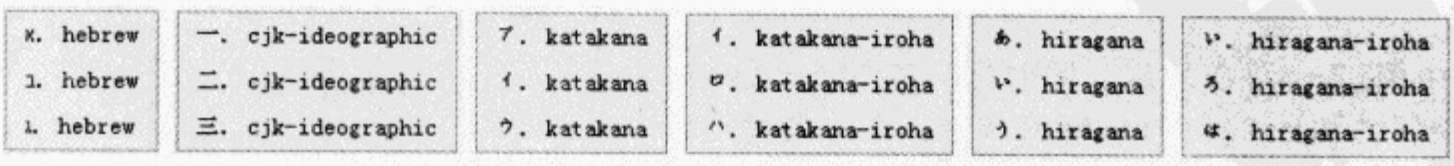


图12-1　CSS 2.1未包含的CSS 2关键字的显示

标记的具体显示细节，还依赖于客户端，例如下列代码，其在不同浏览器的显示如图12-2所示。

```
ul {
……
list-style-type : circle;
}
<ul>
 <li>circle</li>
 <li>circle</li>
 <li>circle</li>
</ul>
```

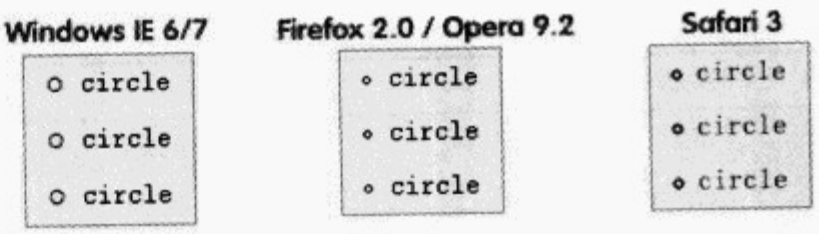
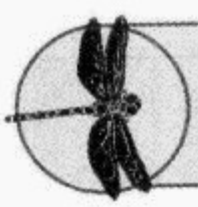


图12-2 不同浏览器对列表标记的渲染不同



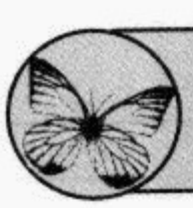
注意：由于list-style属性是可继承的，因此对列表元素（<ul>或者<ol>）设定该属性或者直接对设定<li>元素都是可以生效的。

12.1.2 列表样式图片：list-style-image属性

系统生成的列表标记往往难以满足设计人员的要求，而通过list-style-image属性指定列表项的标记图片可以使标记变得更加丰富。list-style-image属性具体定义列表如下：

语法	list-style-image : <uri> | none | inherit
说明	设置列表项标记的图片
值	uri：图片的链接地址，请参见本书［5.7 URL + URN = URI］一节。 none：无图片
初始值	none
继承性	继承
适用于	display属性为“list-item”的元素
媒体	视觉
计算值	绝对的URI或者为“none”

设定list-style-image属性则会替代list-style-type属性的设定，除非图片URI无效或者无法显示。例如下列代码，其显示如图12-3所示。



提示：本小节示例代码，读者可以参见下载文件包内［/第2部分/第12章：列表和生成的内容/list-style-image.html］文件。

```
ul {
list-style-image : url(../../img/btn1.gif);
list-style-type : circle;
……
}
<ul>
 <li>list-style-image</li>
 <li>list-style-image</li>
 <li>list-style-image</li>
</ul>
```

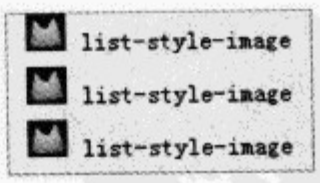


图12-3 使用list-style-image属性设定列表项的标记图片

由图12-3可以发现，同时定义的list-style-image属性和list-style-type属性，图片会优先显示，但是如果图片不能正常加载，则显示如图12-4所示。作为标记的图片的尺寸并没有要求，但是浏览器对其的表现可能不同，如图12-5所示。

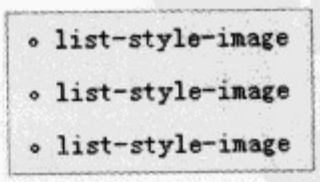


图12-4 标记图片无法显示时则显示list-style-type属性设定的标记

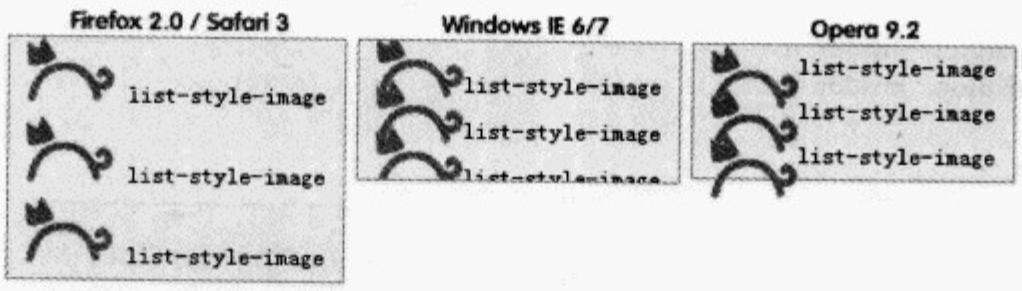
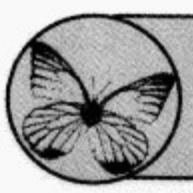


图12-5　不同浏览器对标记图片的显示方式不同



提示：由于标记图片不能控制其显示的方式和位置，因此一般情况推荐使用设定<li>的背景图片来实现。

12.1.3 列表样式定位：list-style-position属性

通过列表样式定位（list-style-position）属性可以简单地控制标记相对于主块框的位置。list-style-position属性具体定义列表如下：

语法	list-style-position : inside | outside | inherit
说明	设置列表项标记的位置
值	inside：标记框是主块框的第一个行内框，其后跟随的是元素的内容。 outside：标记框在主块框内外
初始值	outside
继承性	继承
适用于	display属性为“list-item”的元素
媒体	视觉
计算值	同指定值

CSS 2.1没有指明确切位置的标记框，因此其表现还是由用户端来决定。同时需要注意的是，即使是对<ul>或者<ol>元素指定属性，而实际还是应用在<li>元素上。例如下列代码，其显示如图12-6所示。

```
.list2 {
margin:5px 20px;
padding:0;
border:2px solid #F90;
background: #FFC;
width:300px;
}
.list2 li {
margin : 5px 0 5px 20px;
border : 3px solid #6C0;
background : #CFC;
}
#position1 {
list-style-position : outside;
}
#position2 {
list-style-position : inside;
}
<ul id="position1" class="list2">
  <li>list-style-position : outside</li>
  <li>list-style-position : outside</li>
</ul>
<ul id="position2" class="list2">
```

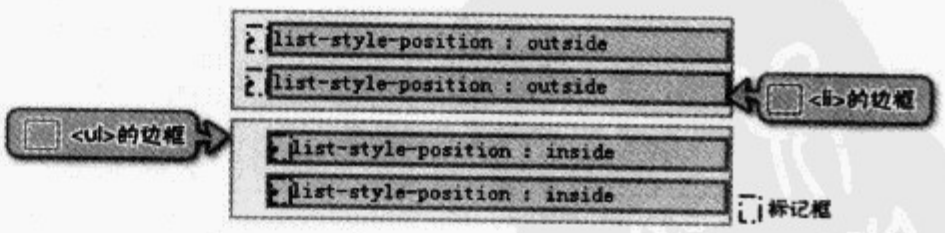
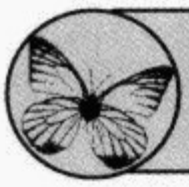


图12-6　列表项标记的定位

```
 <li>list-style-position : inside</li>
 <li>list-style-position : inside</li>
</ul>
```



提示： 本小节示例代码，读者可以参见下载文件包内［/第2部分/第12章：列表和生成的内容/list-style-position.html］文件。

由图12-6可以发现，标记框相对于<li>元素的主块框，而不是<ul>元素的。而且在从右到左的文本中，标记会在主块框的右边。

12.1.4 列表样式缩写：list-style属性

同background和font属性类似，列表样式也有缩写属性list-style，具体定义列表如下：

语法	list-style : [<'list-style-type'> || <'list-style-position'> || <'list-style-image'>] | inherit
说明	设置列表项标记的样式
值	list-style-type、list-style-position和list-style-image属性的值
初始值	视各属性而定
继承性	继承
适用于	display属性为“list-item”的元素
媒体	视觉
计算值	视各属性而定

list-style-type、list-style-position和list-style-image属性值可以按任意顺序排列，而且也可以省略不写，例如下列代码都是正确的。

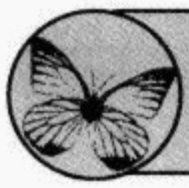
```
li { list-style : none; }
ul { list-style : inside url(../../img/btn1.gif); }
ol { list-style : url(../../img/btn1.gif) outside square; }
```

12.1.5 浏览器对列表的表现与样式的继承

由于CSS对于一些细则没有规定，因此浏览器对于列表的呈现存在着一定的差异。

1. 浏览器的默认样式与列表的表现

在［2.3.1 (X)HTML与浏览器默认样式］一节介绍过浏览器有自己的默认样式，而对相同元素的默认样式，浏览器之间可能存在着差异。例如下列代码，其在不同浏览器内显示如图12-7所示。



提示： 本小节示例代码，读者可以参见下载文件包内［/第2部分/第12章：列表和生成的内容/ list_browser.html］文件。

```
ul {
background : #FC6;
border : 3px solid #F90;
}
li {
background : #FF9;
border : 2px solid #6C0;
border-left-width : 15px;
}
```

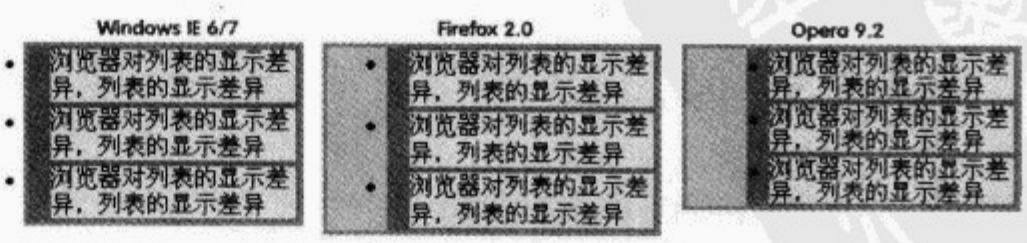
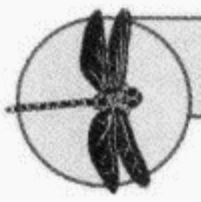


图12-7 浏览器默认样式的差别

```
<ul id="list1">
 <li>浏览器对列表的显示差异，列表的显示差异</li>
 <li>浏览器对列表的显示差异，列表的显示差异</li>
 <li>浏览器对列表的显示差异，列表的显示差异</li>
</ul>
```



注意： 除了为<ul>和<li>设置背景色和边框，没有设定任何其他的CSS规则。

如果修改<ul>和<li>的CSS规则如下，则在各浏览器内显示如图12-8所示。

```
ul {
padding:0;
}
li {
margin : 0 0 0 20px;
border-width : 2px;
padding-left : 20px;
}
```

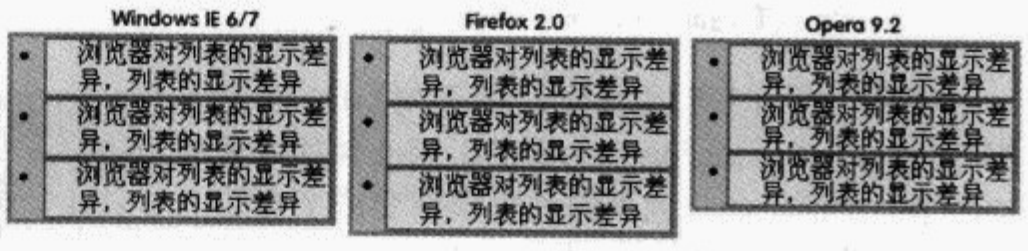


图12-8　修改<ul>元素的padding属性后的显示差别

对比图12-7和图12-8可以发现，浏览器的差异主要表现在：

- IE 6.0/7.0中列表（<ul>或<ol>）元素没有左补白，而Firefox和Opera有左补白；
- 列表默认标记的位置在<li>元素之外，IE 6.0/7.0和Firefox是从<li>元素左边框的外边缘向左一定距离，而Opera则是从<li>元素的左补白边开始向左一定距离，这个距离CSS无法控制，由浏览器决定；
- 列表标记显示在边距、补白和边框之上；
- Firefox 2.0中标记默认样式“disc”是菱形点而不是圆点。

如果修改<ul>的样式定位为“inside”如下，则在各浏览器内显示如图12-9所示。

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

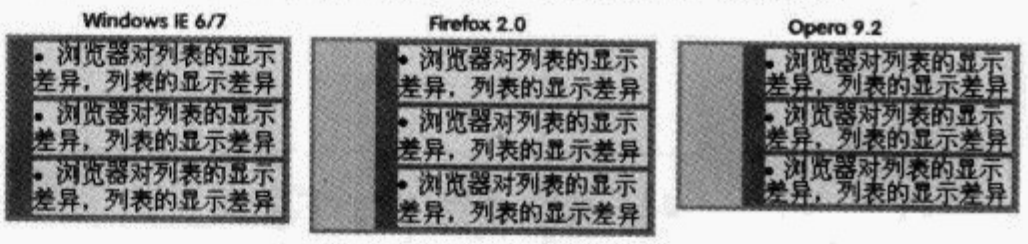


图12-9　“list-style:inside”时浏览器的不同表现

如果将<ul>的补白设为0，CSS规则如下，则其显示如图12-10所示。

```
ul { padding : 0; }
li { border-width : 2px; }
```

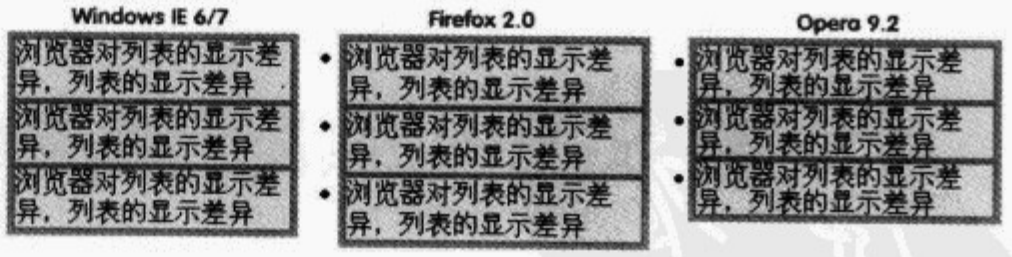


图12-10　将<ul>元素补白设定为0时的显示

正是由于列表样式与浏览器各自的“理解”关系密切，因此在实际应用中，如果需要精确控制列表项的标记，则一般不使用list-style属性，而是采取对<li>或者<ul>（<ol>）元素设置背景图片的方式。

在CSS 2规范中，提供了marker-offset属性和display属性的“marker”关键字来控制列表的标记，但是在实际使用中，却并没有收到好的效果，因此在CSS 2.1中又被删除了。

目前CSS 3的草案中定义了一个新的和更紧凑的方式来影响标记，即::marker伪元素。如果这个模块不会在正式规范发布时改变，那么制作者可以通过设定类似以下代码来控制标记：

```
li::marker { margin-right : 0.2em; }
```

2. 列表的嵌套与样式的继承

在使用列表样式的时候还需要注意：列表样式的各属性是可继承的，因此，如果存在嵌套的列表，则可能都将应用样式，例如下列代码，其显示如图12-11所示。

```
ul {
padding-left : 2em;
background : #FFC;
list-style : inside square;
}
<ul>
 <li>内容文字</li>
 <li>嵌套ol
  <ol>
      <li>嵌套的ol的li</li>
      <li>嵌套的ol的li</li>
      <li>嵌套的ol的li</li>
    </ol>
 </li>
 <li>嵌套ul
  <ul>
      <li>嵌套的ul的li</li>
      <li>嵌套的ul的li</li>
      <li>嵌套的ul的li</li>
    </ul>
 </li>
</ul>
```

由图12-11可以发现，由于对<ul>设定了"list-style : inside square"，因此包括嵌套的<ul>在内的2个<ul>列表的列表项都变成了实心方块，而<ol>内的列表项为默认的数字序号。而如果设定CSS如下，则其显示如图12-12所示。

```
li{ list-style-type:square; }
```

由图12-12可以发现，包括<ol>在内的所有<li>都应用了样式。而如果设定CSS如下，则其显示如图12-13所示。

```
ul ol li { list-style:lower-roman; }
```

因此在设定嵌套的列表项样式的时候，需要特别注意。

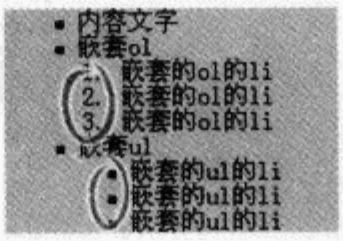


图12-11 列表样式的继承

图12-12 对列表项设定CSS将可能作用在所有列表项上

图12-13 利用包含选择器设定列表项的CSS

12.2 生成的内容

列表的标记或者编号属于最简单的生成内容，而CSS还可以控制用户端生成更加复杂的内容。

12.2.1 :before和:after伪元素

在本书［4.3.2 伪元素（Pseudo-Elements）］一节曾经介绍过，:before和:after伪元素用来在一个元素的内容之前（:before）或之后（:after）插入生成的内容，这个内容在(X)HTML文档内并不存在，而是通过CSS的content属性设定的。例如下列代码，其显示如图12-14所示。

提示： 本小节示例代码，读者可以参见下载文件包内［/第2部分/第12章：列表和生成的内容/ content.html］文件。由于IE 7.0及更早的版本不支持伪元素，因此请使用Firefox或者Opera等浏览器测试。

```
p.note {
color : #C00;
border : 1px solid #F90;
……
}
p.note:before {
content : "提示： ";
font-weight : bold;
}
<p class="note">:before和:after必须和CSS的content属性配合使用。</p>
```

提示：:before和:after必须和CSS的content属性配合使用。

图12-14　利用伪元素插入内容

一个元素生成的格式化对象（如框）包含生成的内容，因此生成的文字“提示：”包含在<p>的边框之内。

:before和:after伪元素继承它们在文档树相联的元素的所有可继承的属性（例如本例中的color属性），而非继承的属性取它们的初始值，因此，由于display属性的初始值为“inline”，所以本例中伪元素作为一个行内框插入（即和元素的初始文本内容保持在同一行）。

而如果设定伪元素为块级元素，例如下列代码，其显示如图12-15所示。

```
p.end {
……
}
p.end:after {
content : "-=本章结束=-";
display : block;
text-align : right;
}
<p class="end">本章介绍了列表和生成的内容，下一章将介绍与用户界面相关的CSS属性。</p>
```

本章介绍了列表和生成的内容，下一章将介绍与用户界面相关的CSS属性。

-=本章结束=-

图12-15　设置伪元素为块级元素

由图12-15可以发现，插入的内容独占一行并且右对齐显示。

提示：利用伪元素，可以针对不同的媒体生成相应的内容，例如针对屏幕阅读器加入表示段落开始或者结束的语音提示，针对打印机加入页眉和页脚等。可以将content声明放置在@media规则中。例如，文本可以用于任何媒体组，而图形仅适用于图形和点阵图形媒体组，而声音文件只适用于音频媒体组。

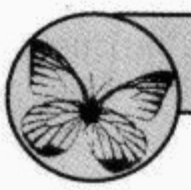
12.2.2　生成内容：content属性

:before和:after伪元素必须和content属性一起使用，否则无法生效，content属性的具体定义列表如下：

语法	content：normal | none | [<字符串> | <uri> | <计数器> | attr(<标识符>) | open-quote | close-quote | no-open-quote | no-close-quote]+ | inherit
说明	设置生成的内容
值	none：不生成伪元素。 normal：:before和:after伪元素的计算值为“none”。 <字符串>：文本内容。 <uri>：该值为指定了一个外部的资源URI（例如1个图片）。如果用户端的媒体类型的限制而不支持该资源，将忽略该资源。 <计数器>：计数其可以通过counter()或者counters()两个函数指定。 open-quote和close-quote：该值由quotes属性中合适的字符串代替。 no-open-quote 和no-close-quote：不插入任何内容（空字符串），但是增加引号的嵌套层次。

续表

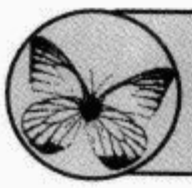
	attr(X)：该函数以字符串形式返回选择器主体的X属性的值
初始值	normal
继承性	不继承
适用于	:before和:after伪元素
媒体	全部
计算值	对于元素，总是“normal”。对于before和:after，如果设定为“normal”，则计算值为“none”。否则，对于URI值，计算值为绝对的URI；对于attr()值，生成的字符串；对于其他关键字同指定值



提示：关于字符串、URI、计数器等值，可以参见本书［第5章 单位和值］。

1. 字符串

字符串会照原样输出，除非之内含有特殊性质的转义字符，例如下列代码其显示如图12-16所示。



提示：本小节示例代码，读者可以参见下载文件包内［/第2部分/第12章：列表和生成的内容/ content.html］文件。

```
#content1 { …… }
#content1 h4 { …… }
#content1 p { …… }
#content1 p { …… }
#content1 h4:before {
color : #FFF;
content : "<span>content:</span>";
}
#content1 p:before {
display : block;
text-align : center;
white-space : pre;
color : #999;
content : "样例段落\A **********"
}
<div id="content1">
 <h4>字符串值</h4>
 <p>内容文字。内容文字。内容文字。内容文字。内容文字。</p>
</div>
```

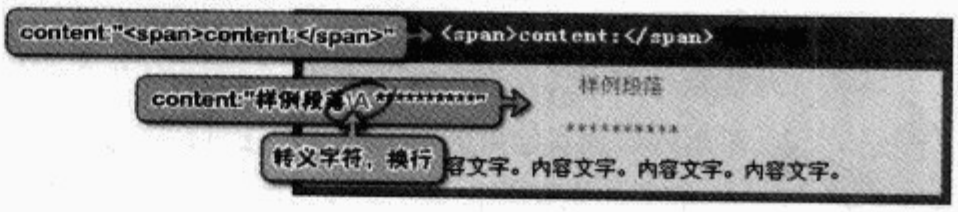


图12-16 字符串与转义字符

由图12-16可以发现，h4:before中content的值被原样显示出来，而不是生成元素，而p:before的content值则由于有转义字符“\A”而换行。

2. URI

如下代码将在段落文字前加入1个图片，如图12-17所示。

```
#content2 p:before { content : url(../../img/btn1.gif); }
<div id="content2">
 <h4>URI</h4>
 <p>内容文字。内容文字。内容文字。内容文字。内容文字。</p>
</div>
```

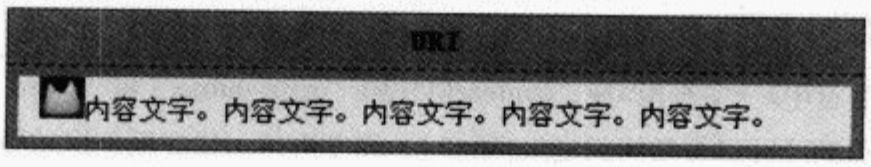


图12-17　content属性的URI值

可以同时设定多个content属性值，多个值之间用英文空格分隔。例如右边代码，其显示如图12-18所示。

```
#content2 p:before { content:url(../../img/btn1.gif)
"提示：";}
```

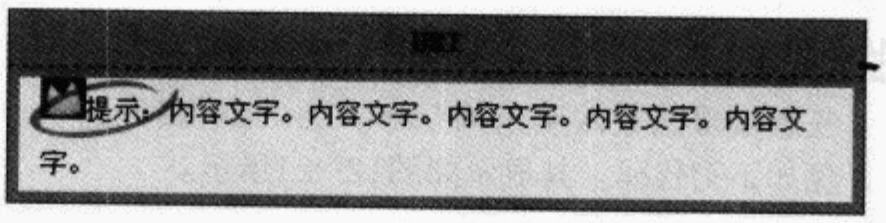


图12-18　同时设定多个content属性值

3. 属性值

有时，制作者可能需要提取元素属性值并作为插入内容的一部分来显示。例如下列代码，可以将链接的href属性值显示出来：

```
#content3 a:after { content: "[" attr(href) "]"; }
<div id="content3">
 <h4>属性值</h4>
    <p>您可以：<a href="http://www.ddcat.net/" title="访问猫窝">访问猫窝</a>，或者<a
href="http://bbs.ddcat.net/" title="去猫沙盆">去猫沙盆</a>淘宝。</p>
</div>
```

CSS处理器不解析attr(X)函数中的字符串X。如果选择器主题没有X属性，返回一个空字符串。属性名的大小写敏感性取决于文档语言。任何的属性值都可以插入到内容中，如class属性值或者id属性值。以下代码将在 后面插入该 的id属性值：

图12-19　利用content的attr(X)函数显示链接的href属性的值

```
div:after { content: "[id: " attr(id) "]"; }
<div>
 没有id的div。
</div>
```

如果 没有id属性，则返回空串，如图12-20所示。

但是除了attr(id)以外的content属性值（如“[id:”）还会正常显示。

图12-20　当未取到属性值时返回空字符串

4. 引用标记

生成内容的一个特殊性的形式就是引用标记，而CSS 2.x提供了强大的方式来管理引用和嵌套行为。而实现这个需要content属性的类似“open-quote”值和quotes属性搭配使用。

quotes属性的具体定义列表如下：

语法	quotes : [<字符串> <字符串>]+ | none | inherit
说明	设置生成内容的引号
值	none：content属性的“open-quote”和“close-quote”值不产生引用标记。

续表

	[<字符串> <字符串>]+：content属性的“open-quote”和“close-quote”值从该引号对列表中得到（开引号和闭引号）。第1对（最左边的）代表最外层的引用，第2对引号代表第1层的嵌套，以此类推。用户端必须根据嵌套的层次使用相应的引号对
初始值	由用户端决定
继承性	继承
适用于	所有元素
媒体	视觉
计算值	同指定值

quotes属性指定了任意数量的嵌入引用的引号。content属性的“open-quote / close-quote”值会生成引号，例如下列代码，其显示如图12-21所示。

```
span span { color:#F00; }
span:before { content:open-quote; }
span:after { content:close-quote; }
<p>嵌套：<span>span中的<span>span</span>。</span></p>
```

由图12-21可以发现，浏览器生成了英文双引号作为引用标记，而2层嵌套的的标记符号是相同的。如果增加CSS如下，则其显示如图12-22所示。

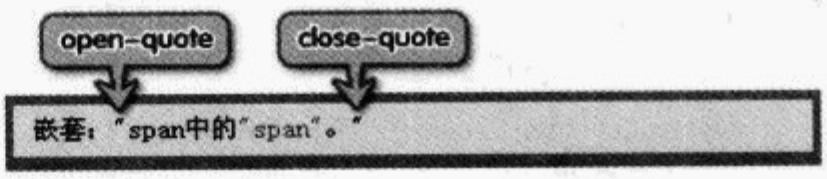
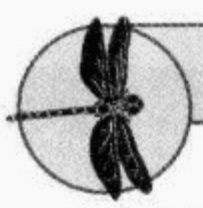


图12-21 content属性的“open-quote / close-quote”值生成的引用标记

```
span { quotes:"[" "]" "(" ")"; }
```



注意： quotes属性在相应选择器内定义，而不是在伪元素内定义。

由图12-22可以发现，嵌套的按照顺序使用了quotes属性定义的2对引号。但是如果将XHTML修改如下：

```
<p>嵌套：<span>span中的<em>em</em>。</span></p>
```

则其显示如图12-23所示。

这是因为quotes属性必须和content属性一起定义，即设定选择器quotes属性的同时，还要定义该选择器的:before和:after伪元素的content属性值为“open-quote”和“close-quote”。可以理解为，content属性在元素前后插入标记，而quotes属性用来定义标记的符号。

因此，增加的CSS如下定义，其显示如图12-24所示。

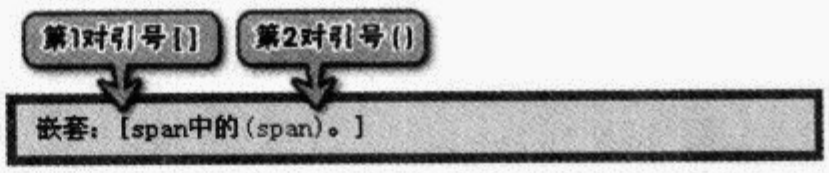


图12-22 使用quotes属性定义嵌套的引用标记符号

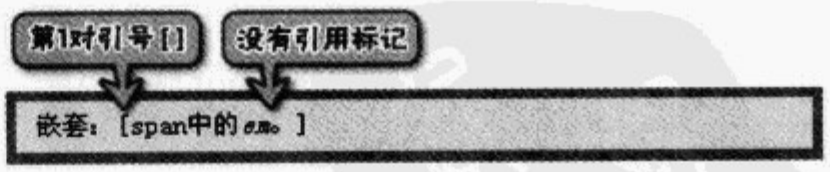


图12-23 嵌套只限于对应的选择器

```
em:before { content:open-quote; }
em:after { content:close-quote; }
```

图12-24 quotes属性必须和content属性一起定义

引用标记的嵌套是根据content属性来决定的，例如下列代码：

```
……
span { quotes:"{" "}" "[" "]" "(" ")"; }
span:before, em:before { content:open-quote; }
```

```
span:after, em:after { content:close-quote; }
<p>嵌套：<span>span中的<strong>strong中<em>em</em></strong></span>。</span></p>
```

由图12-25可以发现，虽然也在元素内，但是由于未设定content属性，因此不计算在嵌套内，因此使用第2对引号，如果增加元素的CSS定义如下，则其显示如图12-26所示。

图12-25　content属性决定了标记的嵌套

```
strong:before { content : open-quote; }
strong:after { content : close-quote; }
```

图12-26　增加元素的content设置后的嵌套

由此可见，嵌套深度与源文档的嵌套或格式化结构无关。content属性还有2个值“no-open-quote / no-close-quote”，该值表示不生成标记符号，但是计算嵌套的层，例如修改上例代码如下，则其显示如图12-27所示。

```
strong:before { content : no-open-quote; }
strong:after { content : no-close-quote; }
```

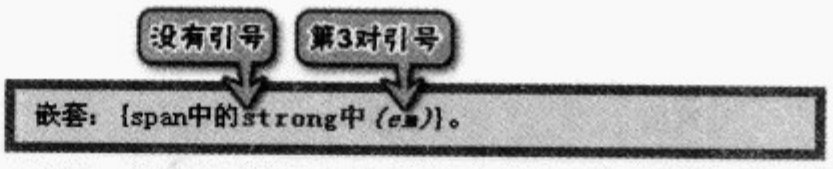


图12-27　“content : no-close-quote”的表现

由图12-27可以发现，虽然元素前后没有引号，但是嵌套的关系还存在，因此使用的是第3对引号。如果嵌套的层次大于定义的引号对数，则最后一对引号会重复使用。也可以使用ISO 10646字符来设置quotes属性的符号，下表列出了一些ISO 10646的引号字符：

字符	大致外观	ISO 10646码（十六进制）	描述
"	"	0022	引号（ASCII双引号）
'	'	0027	撇号（ASCII单引号）
‹	‹	2039	左尖括号
›	›	203A	右尖括号
«	«	00AB	左双尖括号
»	»	00BB	右双尖括号
‘	`	2018	左单引号［single high-6］
’	'	2019	右单引号［single high-9］
“	``	201C	左双引号［double high-6］
”	"	201D	右双引号［double high-9］
„	,,	201E	低双LOW-9引号［double low-9］

这些字符需要使用“\”来转义，例如：

```
quote {quotes: '\201C' '\201D' '\2018' '\2019';}
```

12.2.3 自动记数和编号

在CSS 1中，没有提供编号内容，而(X)HTML的有序列表<ol>中的列表项可以自动编号，这可以看做是一种计数器。但是在XML中，并没有类似<ol>这样的有序列表元素，因此需要CSS提供一个可以定义如何计数的方法。

CSS 2中的计数器并不是像(X)HTML中的<ol>那样只是简单地计数，通过2个属性和2个content属性的值的配合，可以设定样式复杂的计算，例如像本书目录这样的计数“12.2”、“12.2.1”。

1. 复位和增量

计数器最基本的功能就是开始一个计数，并且设定其增量。

counter-reset属性可以使计数器复位，重新开始计数，其具体定义列表如下：

语法	counter-reset : [<标识符> <整数>?]+ | none | inherit
说明	将指定选择器的计数器复位
值	none：不复位。 <标识符>：计数器名称（标识）。 <整数>：指明该元素每次出现时记数器被设置的值，默认为0。 [<标识符> <整数>?]+：表示<整数>为可选值，而且［ <标识符> <整数>? ］可重复出现
初始值	none
继承性	不继承
适用于	所有元素
媒体	全部
计算值	同指定值

而counter-increment属性可以设定计数器的增量，其具体定义列表如下：

语法	counter-increment : [<标识符> <整数>?]+ | none | inherit
说明	元素每次出现时，计数器按照指定的增量计数
值	none：不增加。 <标识符>：要计数的计数器名称（标识）。 <整数>：指明记数器的增量值，默认为1，可以为0或者负整数。 ［ <标识符> <整数>? ］+：表示<整数>为可选值，而且［ <标识符> <整数>? ］可重复出现
初始值	none
继承性	不继承
适用于	所有元素
媒体	全部
计算值	同指定值

counter-reset属性可以定义1个或者多个计数器，<标识符>是制作者指定的计数器名称，后面可以跟一个表示每次元素出现时计数器开始的值，而counter-increment属性为指定的计数器设定一个增量。例如下列代码，

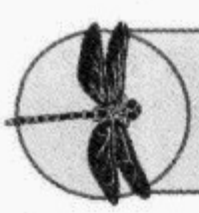
```
h1#ch3 { counter-reset : chapter 3; } /* id 为#ch3 的h1的计数器 chapter 从3开始 */
 h2 { counter-reset: section subsec; } /* 设置 section 和 subsec 为 0，即section和subsec在每个h2出现后开始重新计数 */
h3 { counter-increment: section; } /* 每个h3都会使 section 加 1 */
```

counter-reset中计数器的值允许为负值，例如：

```
h1 {counter-reset: chapter 3 section -3 subsec ;}
```

增量同样允许为负值，例如下列规则，section的值将递减。

```
h3 { counter-increment: section -1;}
```



注意：CSS规范没有规定用户端如何处理负值在非数字型计数器样式中的显示，例如计数器的值为-8，而计数器的样式为“upper-alpha”。

counter-reset属性遵循层叠规则。因此，基于层叠，右边的样式将只重置“imagenum”。

```
h1 { counter-reset : section -1 ; }
h1 { counter-reset : imagenum 99 ; }
```

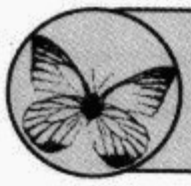
如果要重置两个记数器，它们必须一起指定，如右边代码：

```
h1 { counter-reset: section -1 imagenum 99 }
```

2. 嵌套和作用范围

记数器是“自我嵌套”的，即如果在一个子元素中重复用一个记数器，将自动生成该记数器的一个新的实例。这对于某些情况（如HTML中的列表）很重要，因为元素可以在它们之内嵌套到任意深度，而不必为每一个层次定义一个独立命名的记数器。

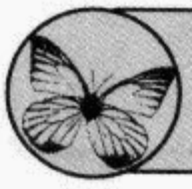
因此，如下的CSS对于数字编码的嵌套列表项来说就足够了，其结果和设置<li>元素的“ist-style: inside”类似，如图12-28所示。



提示：本小节示例代码，读者可以参见下载文件包内［/第2部分/第12章：列表和生成的内容/ counter.html］文件。

```
ol {
……
counter-reset : item;
}
li { list-style : none; }
li:before {
content : counter(item) ". ";
counter-increment : item;
}
<ol>                        <!-- { item[0]=0       -->
 <li>列表项</li>            <!-- item[0]++ (=1)   -->
 <li>列表项                 <!-- item[0]++ (=2)   -->
  <ol>                      <!-- { item[1]=0       -->
   <li>列表项</li>          <!-- item[1]++ (=1)   -->
   <li>列表项</li>          <!-- item[1]++ (=2)   -->
   <li>列表项               <!-- item[1]++ (=3)   -->
    <ol>                    <!-- { item[2]=0       -->
     <li>列表项</li>        <!-- item[2]++ (=1)   -->
    </ol>                   <!--                  -->
    <ol>                    <!-- }{ item[2]=0      -->
     <li>列表项</li>        <!-- item[2]++ (=1)   -->
    </ol>                   <!--                  -->
   </li>                    <!-- }                 -->
   <li>列表项</li>          <!-- item[1]++ (=4)   -->
  </ol>                     <!--                  -->
 </li>                      <!-- }                 -->
 <li>列表项</li>            <!-- item[0]++ (=3)   -->
 <li>列表项</li>            <!-- item[0]++ (=4)   -->
</ol>                       <!--                  -->
<ol>                        <!-- }{ item[0]=0      -->
 <li>列表项</li>            <!-- item[0]++ (=1)   -->
 <li>列表项</li>            <!-- item[0]++ (=2)   -->
</ol>
```

图12-28　计数器的嵌套



提示：item［n］表示计数器“item”的第n个实例，而“{”和“}”表示该计数器范围的开始和结束。

由图12-28可以发现，由于设定了“ol { counter-reset : item; }”，因此在每个<ol>出现的时候，都将重新开始计数，而不需要对每个<ol>元素设定不同的计数器。

3. 使用计数器

设置了计数器以后，需要使用content属性值中的counter()或者counters()函数取得计数器的值，并且将其显示出来。

（1）counter()函数。

counter()函数有2种形式：counter(名称)或counter(名称，样式)。

- counter(名称)：样式是默认情况（默认为“decimal”），记数器是以十进制数格式化的；
- counter(名称，样式)：可以指定样式，所有适用于list-style-type属性的样式也同样适用于记数器。

例如下列代码，其显示如图12-29所示。

```
.....
div { counter-reset : pNum; }
div p:before { content : "段落" counter(pNum) "： "; }
div p { counter-increment : pNum; }
<div id="counter1">
  <p>段落文字。</p>
  <p>段落文字。</p>
  <p>段落文字。</p>
</div>
```

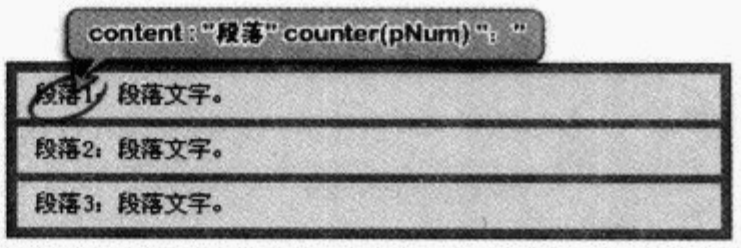
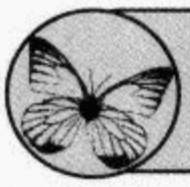


图12-29 将计数器的值插入到元素前



提示：本小节示例代码，读者可参见下载文件包内［/第2部分/第12章：列表和生成的内容/counter.html］文件。

由图12-29可以发现：

- “#counter1 { counter-reset: pNum; }”将计数器“pNum”重置为0（默认值）；
- “#counter1 p { counter-increment: pNum; }”控制每个<p>出现时，pNum都会加1（默认值），如果不设定此属性，则pNum为counter-reset属性中设定的值而不会变；
- “#counter1 p:before { content: "段落" counter(pNum) "： "; }”则是将计数器pNum的值插入到<p>元素之前，其格式是“段落n：”。

“counter(pNum)”函数计算出pNum的当前值，虽然“p:before”表示在<p>元素之前，但是计数器是先增加（或者重置）后使用的。如果修改上例的CSS规则如下，其显示如图12-30所示。

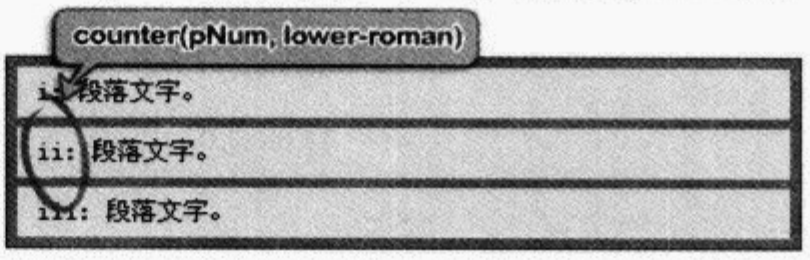


图12-30 为计数器设定样式

```
div p:before { content : counter(pNum, lower-roman) ":"; }
```

（2）counters()函数。

counters()函数也有两种形式：counters(名称，字符串)或counters(名称，字符串，样式)。

本函数生成的文本是指定选择器格式化结构中作用范围内所有该命名的记数器的值，并以指定的字符串分割。记数器以指定的样式（默认为“decimal”）渲染。

例如本章［12.3.4.2 嵌套和作用范围］小节内嵌套的<ol>一例，如果按如下所示修改CSS，则会显示类似“1-2-2”的编号，如图12-31所示。

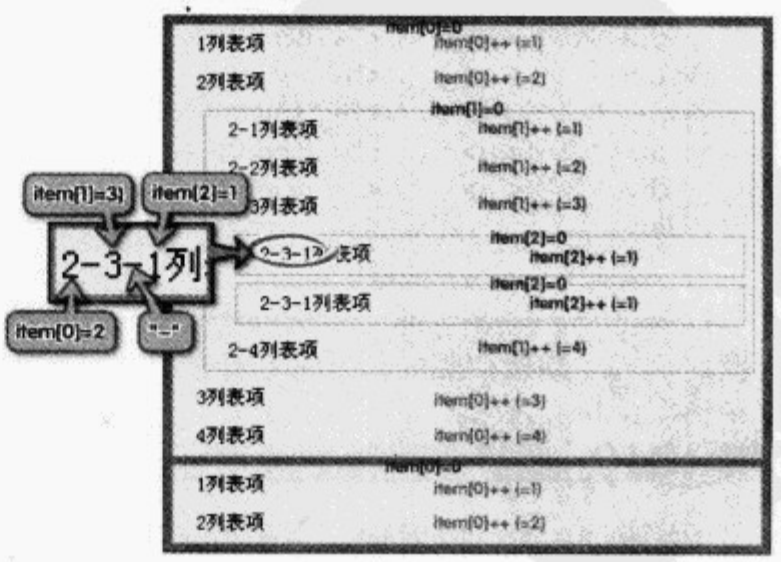


图12-31 counters()函数的使用

由图12-31可以发现，counters()函数分别取得选择器“li:before”内所有计数器“item”的实例的值，并且中间使用“-”分割，因此可以分层显示出编号，而不需要分别设置。

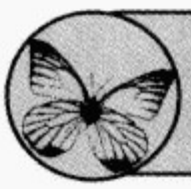
```
li:before {
content: counters(item, "-");
counter-increment: item;
}
```

（3）多个计数器的使用。

可以同时设置多个计数器，针对不同的选择器分别进行计数，例如下列代码，将生成1个书籍的目录，其显示如图12-32所示。

```
h1:before { content: "第" counter(chapter) "章 "; } /* 在h1前面添加文字 */
h1 {
……
counter-increment: chapter;  /* chapter 加 1 */
counter-reset: section;  /* 设置 section 为 0 ，节在每章开始应该重新计数 */
}
h2:before {
/* 在h2前面加入文字：目前 chapter 的值 + "."+ section的值 + 空格*/
content: counter(chapter) "." counter(section) " ";
}
h2 {
……
counter-increment: section;  /* section 加 1 */
counter-reset: subsec;  /* 设置 subsec 为 0 ，小节在每节开始重新计数 */
}
h3:before {
……
/* 在h3前面加入文字：目前 chapter 的值 + "."+ section的值 + "." + subsec的值 + 空格 */
content: counter(chapter) "." counter(section) "." counter(subsec) " "; }
h3 {
……
counter-increment: subsec;  /* subsec 加 1 */
}
<h1>WEB标准概述</h1>
<h2>WEB标准概述</h2>
<h2>表现与结构的分离</h2>
<h2>可访问性</h2>
<h2>难点所在</h2>
<h3>DIV+CSS不等于WEB标准</h3>
<h3>正确使用XHTML标签</h3>
<h2>SEO简介</h2>
<h1>结构与XHTML</h1>
<h2>理解结构与表现</h2>
<h3>内容</h3>
<h3>结构（Structure）</h3>
<h3>表现（Presentation）</h3>
<h3>行为（Behavior）</h3>
<h2>从HTML到XHTML</h2>
<h2>理解(X)HTML标签的语义</h2>
```

图12-32　多个计数器的使用



提示：读者可以参见下载文件包内［/第2部分/第12章：列表和生成的内容/counter2.html］文件。

在本例中，虽然没有针对设定“counter-reset: chapter”，但是“chapter”依然从0开始，这是因为如果counter-increment引用了一个记数器，而它不在任何counter-reset的范围中，该记数器被认为由根元素设置为0。

因此第1个<h1>（计数器先+1）前的计数值为1，而第2个<h1>（再+1）的计数值为2。而在每个h1选择器内需要将<h2>的计数器“section”复位，否则，所有的<h2>会顺序计数而不分

章节，同理，在每个h2选择器内将<h3>的计数器“subsec”复位。

4. 设定了“display: none”的元素的计数

如果一个元素设定了“display: none”，那么它将不会增加计数或者重置计数器。例如下列代码，段落“secret”将不显示，也不参加计数，而第3个段落将被计算为“2”，如图12-33所示。

```
#counter5 { counter-reset: pNum2; }
#counter5 p { counter-increment: pNum2; }
#counter5 p.secret {display: none;}
#counter5 p:before {content : "Num" counter(pNum2) "=";}
<div id="counter5">
 <p>普通的段落1。</p>
 <p class="secret">普通的段落2。</p>
 <p>普通的段落3。</p>
</div>
```

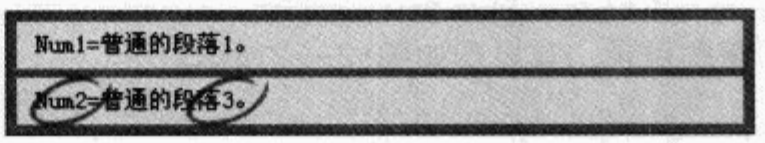


图12-33 设定“display : none”的段落不参见计数

设定“display : none”的伪元素也同样不会增加或者重置计数器。但是，设定“visibility : hidden”的元素会计数，例如修改上例的CSS规则如下，其显示如图12-34所示。

```
#counter6 p.secret { visibility:hidden; }
```

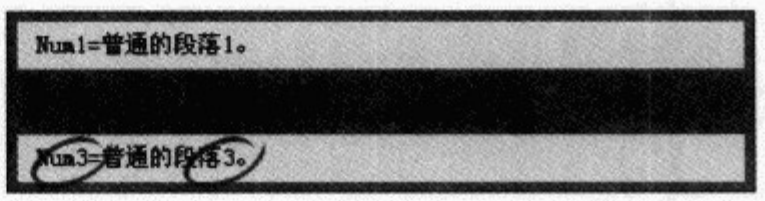
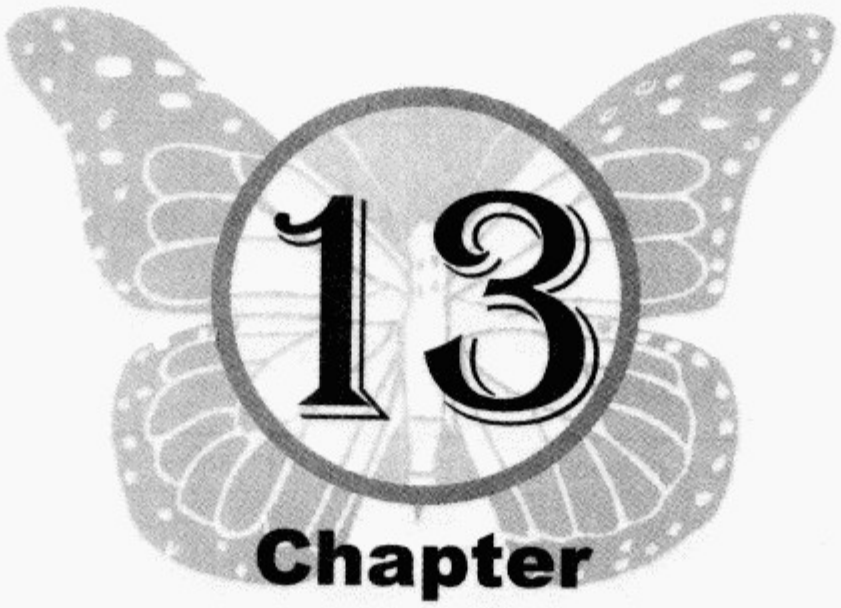


图12-34 设定“visibility : hidden”的段落参见计数



第13章 用户界面

用户界面（User Interface，简称UI）是用户与用户端交互的界面，一个优秀的用户界面，是以用户为中心的，用户界面设计的三大原则是：置界面于用户的控制之下，减少用户的记忆负担，保持界面的一致性。

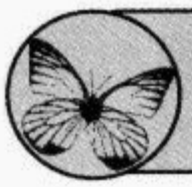
而在CSS中可以设定光标的样式、焦点的外廓等，从而使用户能更加方便和愉快地访问网站，使用网站提供的各项功能。但是，如果设定不当，也可能造成用户的困惑。

13.1 鼠标指针：cursor属性

鼠标指针在用户界面中的作用很大，不同形状的指针可以提示用户，哪里是可以点击的、哪里是可以选择的、哪里是可以输入文字的等。

CSS允许制作者改变指针图标，通过cursor属性可以设定指针使用某个图标，或者是制作者指定的某个图标文件。cursor属性具体定义列表如下：

语法	cursor : [[<uri>,]*[auto | <关键字（见下）>]] | inherit
说明	设置鼠标光标的外观
值	<uri>：指针图标文件的URI（[<uri>,]*表示可以出现0或者多次）。 auto：用户端基于上下文决定应该显示什么光标。 关键字：crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize | text | wait | help | progress
初始值	auto
继承性	继承
适用于	所有元素
媒体	视觉、交互（参见［3.5.2 媒体组］一节）
计算值	<uri>为绝对URI值，否则同指定值



提示：本节示例代码，读者可参见下载文件包内［/第2部分/第13章：用户界面/cursor.html］文件。

13.1.1 关键字

除了“auto”以外的关键字使用的都是系统图标，不同的操作系统、不同的用户端具体的显示可能不同。

- crosshair：十字线（例如，短线组成一个类似“+”的符号）。
- default：基于平台的默认光标。通常渲染为一个箭头。
- pointer：指针光标，表示一个连接。
- move：表示正在移动某些东西。
- e-resize, ne-resize, nw-resize, n-resize, se-resize, sw-resize, s-resize, w-resize：表示正在移动某个边。例如，'se-resize'光标用来表示框的移动开始于东南角。
- text：表示可以选择文本。通常渲染为I形光标。
- wait：表示程序正忙，需要用户等待。通常渲染为手表或沙漏。
- help：光标下的对象有帮助内容。通常渲染为一个问号或一个气球。
- progress：进度指针。程序正在执行一些处理，但是与“wait（等待）”不同，用户可继续与程序互动。通常渲染为一个箭头与手表或沙漏。

如图13-1所示为Windows IE 6.0的指针图标样式。在这些鼠标指针样式中，最常见的就是“pointer”和“text”，前者是链接默认的鼠标指针，后者则是文本或者输入框的鼠标指针。

而有些情况下，比较常见的是使用JavaScript在页面内创建了动态效果，则需要更改鼠标的

样式以便可以更好地和访问者进行交互。例如，使用JavaScipt设定了一个可以点击的区域，为了提示访问者这个区域可以点击，则可以将其鼠标指针样式设置为“pointer”；或者页面内某个元素可以拖动的时候，则可以将其鼠标指针设定为“move”等。

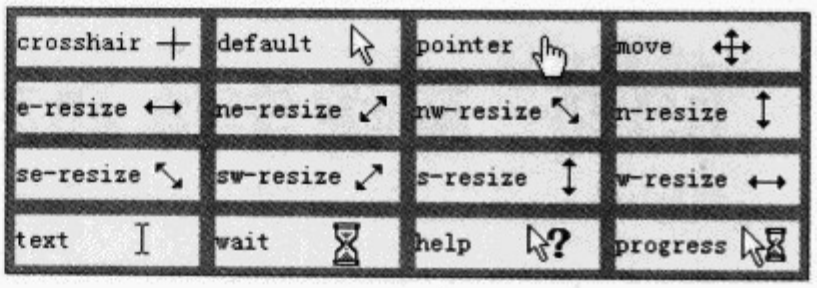
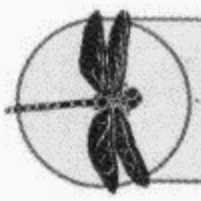


图13-1　Windows IE 6.0的指针图标样式



注意：由于访问者可能已经习惯了这些默认的鼠标指针样式，因此设定样式的时候，还是要遵循图标所对应的含义，而不要随意更改。

13.1.2 图片鼠标指针

鼠标指针文件有【.cur】和【.ani】这两种格式，它们是Windows鼠标指针的标准格式，前者为静态光标，后者为动态光标。可以为cursor属性指定1个或者多个图片地址，其格式如下：

```
cursor: [<url>,]* keyword;
```

“*”的意思是0个或者更多的URL可能被定义（使用英文逗号隔开），但是在最后必须包含一个CSS规范中定义的关键字，如“auto”或“pointer”。

用户端从URI指定的资源处获得指针文件。如果用户端无法处理一系列光标中的第1个，那么它应尝试处理第2个、第3个等。如果用户端无法处理任何用户定义的光标，它必须使用列表最后的通用光标。例如下列代码，其在Windows IE 6.0中显示如图13-2所示。

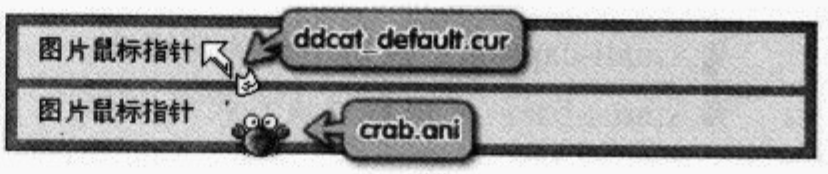
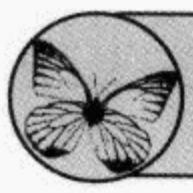


图13-2　Windows IE 6.0中显示的静态和动态图片指针

```
.c17 { cursor: url(../../img/ddcat_default.cur), auto; }
.c18 { cursor: url(../../img/crab.ani), url(http://www.ddcat.net/images/ddcat_link.cur), pointer; }
<div>
 <p class="c17">图片鼠标指针</p>
 <p class="c18">图片鼠标指针</p>
</div>
```



提示：Opera 9.2及更早的版本不支持图片指针，Firefox 2.0及Safari 3.0支持静态图片指针（.cur格式）不支持动态图片指针（.ani格式），而Windows IE 6及以后的版本则两者都支持。

而由于Firefox 2.0不支持动态图片指针，因此将读取第2个URI，其显示如图13-3所示。

图13-3　Firefox 2.0内显示的静态图片指针

提示：CSS还支持另一种指针格式：SVG格式。SVG（Scalable Vector Graphics，可伸缩向量图形）基于xml语言，用来定义面向Web的向量图形。当伸缩或重新设置图形大小时，SVG图形不损失质量。SVG里的每一个元素和属性都可以自定义，它同时也是W3C推荐标准，单是目前缺少浏览器的支持。详细信息请参阅http://www.w3.org/TR/SVG/interact.html#CursorElement（英文）。

13.2 系统字体和颜色

随着基于Web的应用越来越广泛，有时制作者希望网页有着和系统软件类似的界面，虽然CSS 2不可能让文档完全呈现操作系统的外观，但是可以通过选择颜色和菜单字体来使它们尽可能地相近。

提示： 本小节示例代码，读者可以参见下载文件包内［/第2部分/第13章：用户界面/system.html］文件。

13.2.1 系统字体

在本书［6.6.3 系统字体］一节内介绍了使用font属性定义系统字体，有以下的关键字可以定义。

- caption：使用有标题的系统控件的文本字体（如按钮、菜单等）。
- icon：使用图标标签的字体。
- menu：使用菜单的字体。
- message-box：使用信息对话框的文本字体。
- small-caption：使用小控件的字体。
- status-bar：使用窗口状态栏的字体。

而具体的样式，会与访问者所使用的操作系统（同时访问者也可能应用了桌面主题、或者浏览器主题）保持一致。例如下边代码：

```
#font1 p { font : menu; }
<div id="font1">
 <p>系统菜单字体样式</p>
</div>
```

当操作系统菜单设置为“宋体，9pt”时，如图13-4所示，其显示如图13-5所示。而如果调整系统的菜单设置，如图13-6所示，则其显示如图13-7所示。

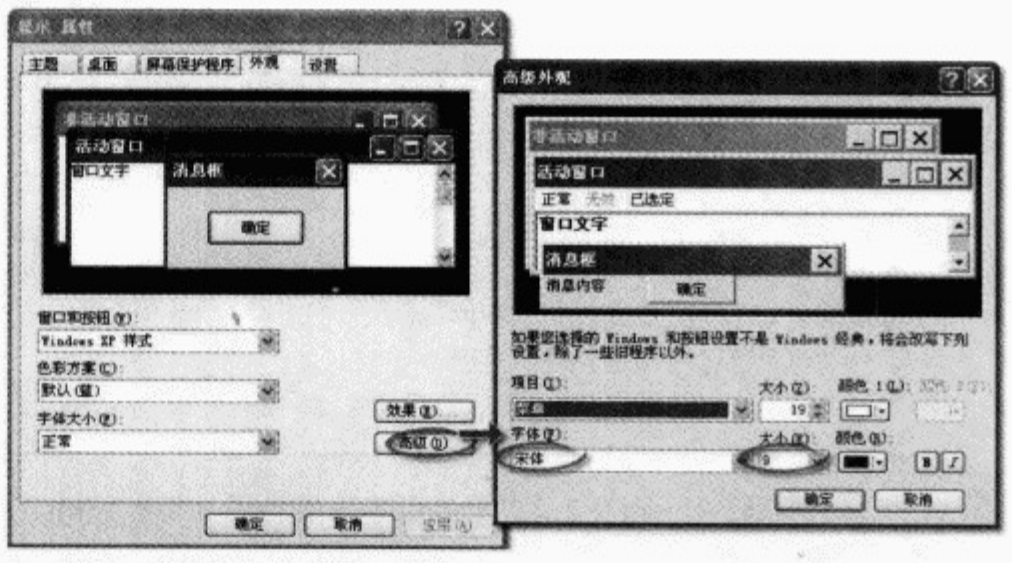


图13-4 系统菜单设置为“宋体，9pt”

图13-5 “p { font : menu; }”的显示效果与菜单一致

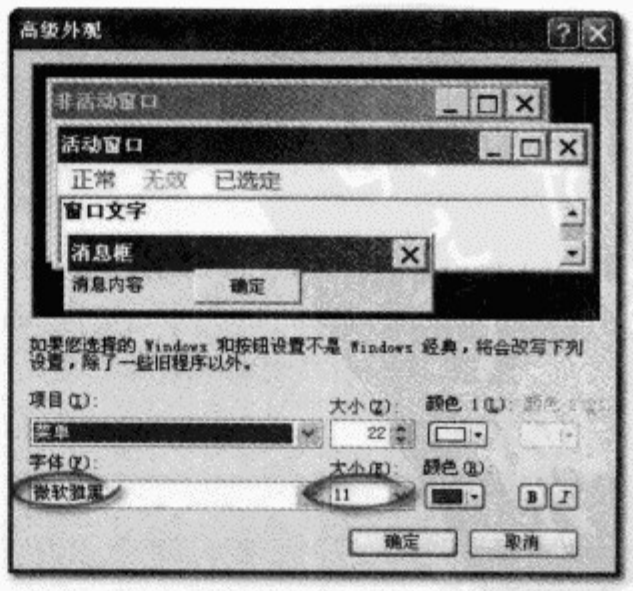


图13-6 修改系统菜单的字体和大小

图13-7　“p { font : menu; }”的显示效果与菜单一致

13.2.2 系统颜色

注意：在CSS 3的颜色模块中并不赞成使用系统颜色关键字，而是用新的属性来代替，因此不推荐使用系统颜色，因为它们可能会从CSS 3中删除。关于CSS 3的颜色模块，读者可以参考http://www.w3.org/TR/2003/CR-css3-color-20030514/（英文）。

CSS 2定义了一系列的系统颜色关键字，它们可以指定给任何可以设定<颜色>值的属性，使制作者可以指定某种和访问者系统相一致的颜色风格。例如：

```
p { background-color: Background; } /* 背景色和访问者操作系统的桌面背景一样的颜色。 */
body {color: WindowText; } /* 文字颜色与操作系统文字颜色一样。 */
```

这样做的好处是，用户可以通过定制自己的桌面而使文档更加便于阅读。CSS 2共定义了28个关键字来调用系统颜色，说明如下。

- ActiveBorder：活动窗口边框。
- ActiveCaption：活动窗口标题。
- AppWorkspace：MDI（Multi Document Interface 多文档界面）窗口的背景颜色。
- Background：桌面背景。
- ButtonFace：三维显示元素的外观颜色。
- ButtonHighlight：三维显示元素的深色阴影（适用于背着光源的边）。
- ButtonShadow：三维显示元素的阴影。
- ButtonText：按钮上的文本颜色。
- CaptionText：标题、尺寸框以及滚动条箭头框中的文本。
- GrayText：灰色（禁用）文本。如果当前显示驱动不支持灰色，将被设置为#000。
- Highlight：控件中被选择的项目。
- HighlightText：控件中被选择的文本。
- InactiveBorder：不活动的窗口边框。
- InactiveCaption：不活动的窗口标题。
- InactiveCaptionText：不活动窗口标题文本颜色。
- InfoBackground：控件提示的背景色。
- InfoText：控件提示的文本颜色。
- Menu：菜单背景。
- MenuText：菜单文本。
- Scrollbar：滚动条灰色区域。
- ThreeDDarkShadow：三维显示元素的深色阴影。

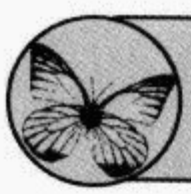
- ThreeDFace：三维显示元素的表面颜色。
- ThreeDHighlight：三维显示元素的高亮颜色。
- ThreeDLightShadow：三维显示元素的浅色阴影（适用于面向光源的那一边）。
- ThreeDShadow：三维显示元素的深色阴影。
- Window：窗口背景。
- WindowFrame：窗口框架。
- WindowText：窗口中文本。

例如，在以下CSS规则中，<p>的背景将与访问者桌面背景颜色一致，而文字则同活动窗口的标题颜色一致。

```
p {
background: Background;
color: ActiveCaption;
}
```

13.3 动态的外廓：outline属性

有些时候，样式表制作者可能想在可视对象（如按钮、活动窗体域、图形地图等）周围创建外廓使它们突出显示。



提示： Windows IE 7.0及更早的版本并不支持此属性。本小节示例代码，读者可以参见下载文件包内［/第2部分/第13章：用户界面/ outline.html］文件。

13.3.1 外廓与边框的区别

虽然外廓看起来和边框很类似，但是它与边框有两点本质区别。

1. 外廓不占用空间

外廓位于元素框之上，它总是会显示在最上面，它不会影响本元素框以及其他元素框的定位和尺寸。也就是说，外廓不参与文档布局，也不会在其出现和消失的转换中触发文档重绘。例如，某个元素设定了30px的外廓，那么外廓很可能会覆盖在其他元素之上。

2. 外廓不一定是矩形

不过这也不意味着可以创建圆形或者三角形的外廓。外廓可能会产生类似分布到若干行的行内元素边框一样效果，多个块融合在一起形成连续的非矩形外廓。

例如下列代码，其显示如图13-8所示。请使用Safari 3.0测试本例代码。

```
#outline1 em { outline : 15px solid #FC0; }
#outline1 strong { outline : dotted 5px #36C; }
<div id="outline1">
  <p>第1个特征：<em>外廓不占空间。</em></p>
  <p>第2个特征：<strong>外廓不一定是矩形，有可能随元素内容分布在几行内，例如本例中的strong元素。
</strong></p>
</div>
```

但是CSS 2.1并没有要求用户端都这样来生成外廓，例如，Firefox 2.0 和Opera 9.2会在各行分开生成矩形的外廓，如图13-9所示。

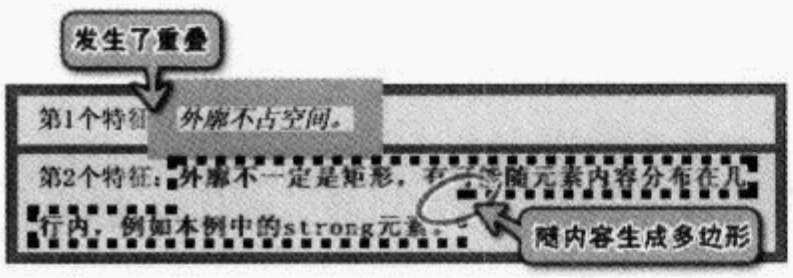


图13-8　外廓与边框的区别

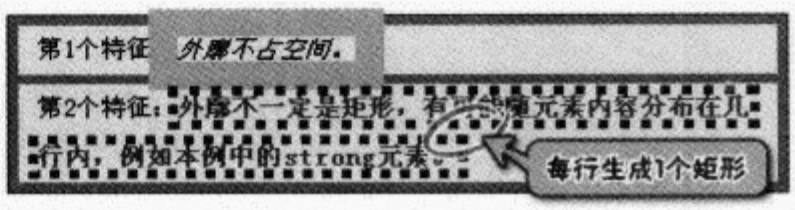
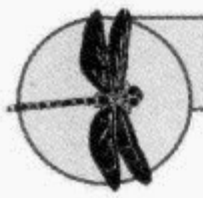


图13-9　Firefox 2.0及Opera 9.2对外廓的显示

13.3.2 外廓宽度：outline-width属性

与边框类似，外廓也可以通过outline-width、outline-style和outline-color属性分别定义宽度、样式和颜色，outline则是缩写属性。除了外廓不能单独定义4个边以外，其属性和边框的相关属性非常相近。outline-width属性的具体定义列表如下：

语法	outline-width: thin | medium | thick | <长度> | inherit
说明	设置外廓的宽度
值	thin：一个窄的外廓。 medium：一个中等的外廓。 thick：一个厚外廓。 <长度>：长度值，不可为负值
初始值	medium
继承性	不继承
适用于	所有元素
媒体	视觉、交互（参见［3.5.2 媒体组］一节）
计算值	为绝对长度值，如果样式为“none”，则值为0



注意：由于外廓是在元素框之上的，因此过宽的宽度会使其覆盖其他元素。

outline-width属性的关键字和border-width属性的关键字一样，其表现也一样。

13.3.3 外廓样式：outline-style属性

使用outline-style属性可以为外廓定义边框的样式，outline-style属性的具体定义列表如下：

语法	outline-style: none | <边框样式> | inherit
说明	设置外廓的样式
值	边框样式：dotted | dashed | solid | double | groove | ridge | inset | outset，同border-style属性，参见本书［8.8.3 边框样式］一节
初始值	none
继承性	不继承
适用于	所有元素
媒体	视觉、交互（参见［3.5.2 媒体组］一节）
计算值	同指定值

外廓样式的关键字和边框样式对应的关键字的表现相同，只是没有“hidden”，用户端会将其按“none”来处理。

outline-style的初始值为“none”。设定outline-style属性值为“none”会强制将outline-width属性值置为0，因此如果不显示地设定outline-style则外廓不会显示。

13.3.4 外廓颜色：outline-color属性

通过outline-color属性可以指定外廓的颜色，具体定义列表如下：

语法	outline-color: <颜色> | invert | inherit
说明	设置外廓的颜色
值	颜色：合法的颜色值。 invert：CSS 2.1，反转屏幕颜色
初始值	invert
继承性	不继承
适用于	所有元素
媒体	视觉、交互（参见［3.5.2 媒体组］一节）
计算值	同指定值

外廓颜色的“invert”值是CSS 2.1增加的关键字，外廓的颜色由其所处的背景颜色决定，将背景颜色“翻转”过来，即取背景的补色，这样保证了外廓在任何背景颜色中都是可见的。

例如下列代码，其显示如图13-10所示。

```
p strong{
outline-width : 3px;
outline-color : invert;
outline-style : dotted;
}
.p1 { background : #C00; }
.p2 { background : #FC3; }
.p3 { background : #000; }
.p4 { background : #fff url(../../img/ddcat2.gif); }
<p class="p1"><strong>invert，翻转颜色</strong></p>
<p class="p2"><strong>invert，翻转颜色</strong></p>
<p class="p3"><strong>invert，翻转颜色</strong></p>
<p class="p4"><strong>invert，翻转颜色</strong></p>
```

图13-10 outline-color的“invert”值取背景色的补色

由图13-10可以发现，不同的背景色上，外廓的颜色不同，都是取背景的补色，包括背景图片部分，也是对应图片相应位置的颜色的补色，因此背景图片上的外廓可能颜色不一样。

同时，也可以使用允许的颜色值，例如：

```
outline-color: red;
outline-color: #000;
outline-color: rgb(50%,50%,50%);
```

13.3.5 缩写：outline属性

外廓的宽度、样式和颜色也可以使用缩写属性outline同时定义，具体定义列表如下：

语法	outline: [<外廓颜色> || <外廓样式> || <外廓宽度>] | inherit
说明	设置外廓的颜色样式和宽度
值	同各属性
初始值	视各属性而定
继承性	不继承
适用于	所有元素
媒体	视觉、交互（参见［3.5.2 媒体组］一节）
计算值	视各属性而定

颜色、样式和宽度值的顺序可以任意调换，例如右上代码：

```
outline : dotted 5px #36C;
outline : 1px #36C solid;
outline : #ff0 dashed 3px;
```

颜色值如果省略，则取其初始值“invert”，例如右右代码：

```
outline: 3px solid;
```

元素可以同时有外廓和边框，CSS 2.1中没有特别严格地规定外廓和边框的位置关系，只是笼统地指出，外廓可以从边框的外边沿开始，而不是“必须”。

因此，浏览器可以认为外廓在边框之外显示，例如下列代码，其显示如图13-11所示。

```
p {
border:5px solid #3CF;
outline: solid 10px #fc0;
}
<p>外廓与边框</p>
```

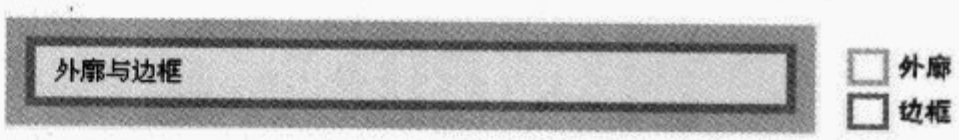


图13-11　外廓在边框外显示

13.3.6 外廓与焦点

外廓一般用于突出元素，因此经常和:focus伪类搭配使用以显示当前焦点所在的元素，例如下列代码，光标所在的输入框会有蓝色的外廓，其显示如图13-12所示。

```
#test_form input[type="text"] {  /* 属性选择器 */
border : 1px solid #999;
}
#test_form input:focus {
border : 1px solid #09F;
outline : 2px solid #6CF;
}
<form id="test_form" method="post" action="#">
 <fieldset>
  <legend>用户基本信息</legend>
  <p><label for="nikename">昵称：</label>
   <input name="nikename" id="nikename" type="text" size="16" maxlength="30" /></p>
   <p><label>地址：</label>
    <input name="address" id="address" type="text" value="" size="30" /></p>
  <p><input type="radio" name="sex" id="sex_male" value="male" />
   <label for="sex_male">男</label>
   <input type="radio" name="sex" id="sex_female" value="female" />
   <label for="sex_female">女</label></p>
  <p><input type="submit" name="btn_submit" value="提交" />
   <input type="reset" name="btn_reset" value="重置" /></p>
 </fieldset>
</form>
```

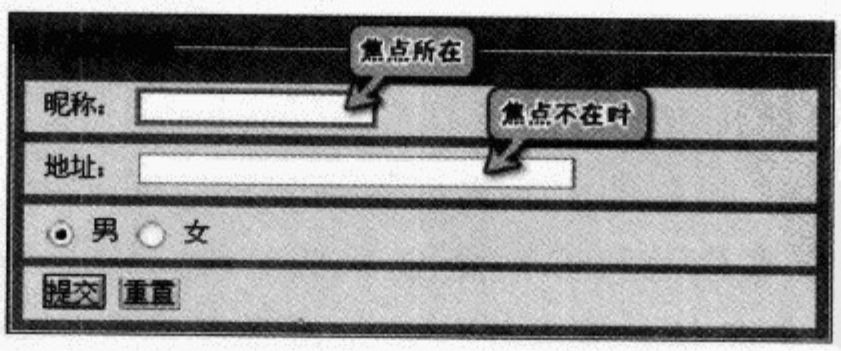
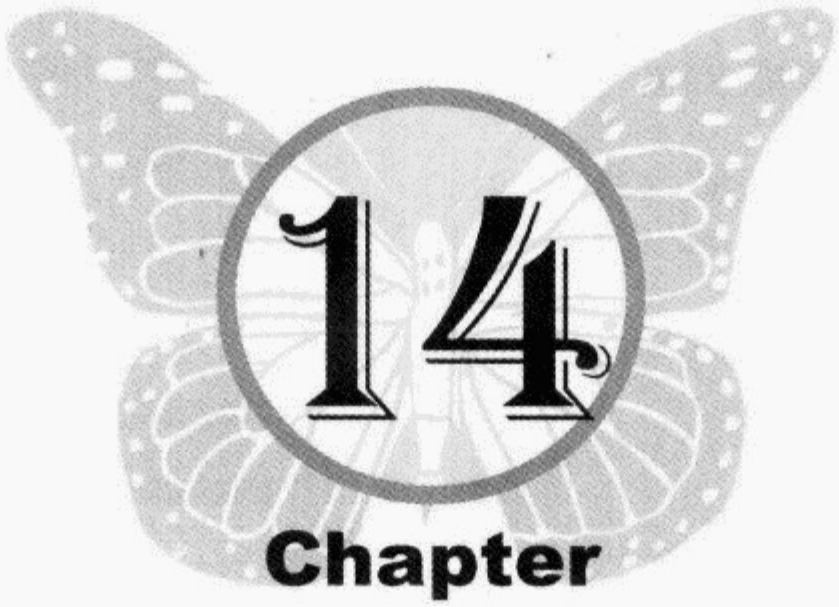


图13-12　利用:focus伪类和outline突出显示焦点所在的元素

本例中使用了属性选择器，外廓将只匹配“type"属性为“text”的<input>元素，而对于单选框和按钮都没有作用。



第14章 页面媒体

本书的第3章已经介绍过CSS支持的媒体类型以及媒体的分组，其中2种分组方式是：

- 连续媒体（continuous，例如显示器）或页面媒体（paged，例如打印机）；
- 视觉（visual）、音频（audio）、语音（speech）或触觉（tactile）。

之前介绍的绝大部分CSS属性都是适用于“视觉”类媒体的，大部分页面媒体也属于视觉媒体（例如打印机和投影仪），同时页面类媒体又有其特殊性。

连续媒体，顾名思义其内容是连续的，可以从头到尾滚动浏览，而页面媒体的内容则是分页显示的，例如图书和杂志。因此，CSS制定了针对页面媒体的一些样式属性。

注意： CSS 2中包含的一些属性，在CSS 2.1中又被删除了，如size属性。

14.1 页面媒体简介

页面媒体（如纸张、透明胶片、在显示器内显示的页等）与连续媒体不同，主要是文档的内容被分割为一个或多个独立的页，为了处理分页，CSS 2.1规则定了如何设置页框（page box）、页边距和如何分页等。

用户端负责将文档的页框转换为实际介质（如纸张或者透明胶片）上的最终效果，其转换可能包括以下内容：

- 将一个页框转换为一页（sheet，如单面打印）；
- 将两个页框转换为同一页的两面（如双面打印）；
- 将N个（小）页框转换到一页（称为“n-up，多页打印在一张纸上”）；
- 将一个（大）页框转换到N * M个页（称为“拼贴”）；
- 创建折标（折标是印在一本书每一印张的第一页底部的一个字母、数字或记号，用于在装订时作为正确次序的指示）；
- 将一个文档打印到若干个输出托盘；
- 将一个文档输出为一个文件。

14.2 指定媒体类型

本书［3.2.3.4 media属性］和［3.5 媒体类型］内介绍过如何指定样式表适用的媒体类型，一般有以下几种方法。

1. media属性

在(X)HTML文档中，media属性可以在<link>和<style>标签内使用，其含义是相同的，即指定样式表的媒体类型，例如：

```
<link rel="stylesheet" type="text/css" href="print.css" media="print">
<link rel="stylesheet" type="text/css" href="page.css" media="print, projection">
<style type="text/css" media="projection">
body {font-family: sans-serif;}
</style>
```

2. @media规则

在样式表（外部或嵌入样式表）中通过“@media”规则或“@import”规则来指定目标媒体。例如：

```
@import url(basic.css) print, screen;
@media print {
 body { font-size: 10pt; }
}
```

或者：

```
<style type="text/css">
body { background: white; }
@media screen {  /* 针对显示器的CSS */
 body { font-family: sans-serif; }
```

```
 h1 { margin-top: 1em; }
}
@media print {  /* 针对打印设备的CSS */
 body { font-family: serif; }
 h1 { margin-top: 2em; }
}
</style>
```

@media结构允许不同媒体的样式表规则存在于同一样式表中。如果不指定媒体类型，则CSS将适用于所有媒体。

提示： 在浏览器的【打印预览】功能中，可以查看媒体类型为“print”的样式效果。而使用Opera在全屏模式下（快捷键［F11］）可以查看媒体类型为“projection”的样式效果。

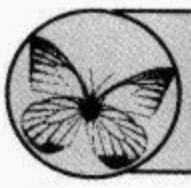
14.3 页框：@page规则

页框包括以下两个区域。

● 页面区域：包括布置在该页面上的框，第1个页面区域的边缘确定文档初始包含块的矩形范围。

● 边距区域：包围着页面区域。

在CSS 2.1中不能设定页框的大小，只能设定边距区域的尺寸。而在CSS 2中，除了可以设定边距的大小，还可以通过size属性设定页框大小。size属性没有包含在CSS 2.1中，因为浏览器对其支持得不好。因此，在CSS 2.1中只有通过边距来控制内容的尺寸。

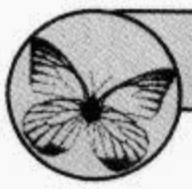


提示： 本小节示例代码，读者可参见下载文件包内［/第2部分/第14章：页面媒体/print.html］文件。读者可以使用Opera 9.2的打印预览来测试效果。

注意： 打印预览的效果还同浏览器内打印相关选项的设置有关，例如纸张大小、方向等。IE 6.0/7.0、Firefox 2.0以及Safari 3.0都不支持@page规则，这些浏览器中使用系统的打印设定来控制边距。

14.3.1 页边距

在CSS 2.1中，只有margin属性（包括4个方向和缩写属性）可以应用在页面上下文（page context）中。页、页框和页边距的关系如图14-1所示。



提示： 关于margin属性，请参见本书［8.9 边距：margin属性］一节。

可以使用@page规则设定页框的边距。@page规则以“@page”关键字开始，后面是页面选择

器（可选），然后是声明块。页面选择器指定了声明所适用的页面，在CSS 2.1中，通过页面选择器可以设置第一页、所有左边的页或者所有下边的页。例如：

```
@page { margin: 3cm; }
```

值得注意的是，页框（或者说页面上下文）和字体没有关系，因此不能使用“em”和“ex”单位来定义边距区域或者页面区域。百分比值的计算则是基于页框的尺寸，左右边距以页框的宽度为基数，而上下边距以页框的高度为基数。当然，还可以使用“in”、“cm”等绝对单位。关于绝对单位，请参见本书［5.3.2 长度单位］一节。

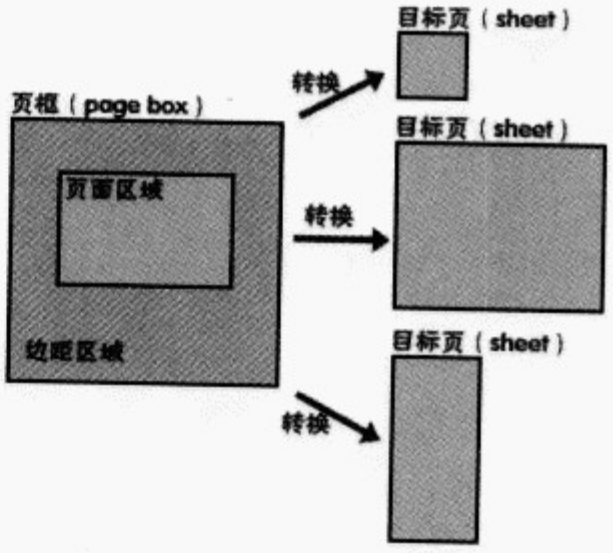


图14-1　纸张、页框和页边距的关系

由于负的边距值（页框或者元素的），或绝对定位的缘故，内容可能会在页框之外，但是这样的内容会被“剪切”掉—用户端、打印机或者最后由裁纸机来完成。

如果一个页框不匹配目标页的尺寸，用户端可以选择：

- 将页框旋转90度，如果这样可以使页框和页面匹配；
- 放缩页面使之与目标匹配。

如果页框小于目标页尺寸，用户端可以自由地在页上放置页框。不过，推荐的方法是将页框放置在页的中央，因为这样做有利于对齐双面的页，也避免了错误丢失那些打印在靠近页边的信息。

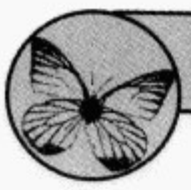
在页面模式中格式化内容时，有些内容可能最终会出现在页框之外。例如某个元素设定了“white-space : pre”而导致元素框突出到页框外；或者由于绝对定位而使元素在不合适的位置。对于这个问题，CSS建议制作者和用户端遵循如下规则。

- 应该允许内容在页框稍微外边一点的地方出现，而允许页面“撑出”。
- 用户端应该避免为了遵守元素的定位而产生大量的空白页框。不过为了遵守page-break-before属性和page-break-after属性的“left”和“right”值，少量的空白页框还是必须产生的。
- 制作者不应该将元素放在一个不合适的位置而避免渲染它们。
- 用户端可以用几种方法处理定位在页框之外的框，包括忽视它们或在文档结尾处为它们创建页框。

对于第3点，在本书［9.3.6 应用：显示提示内容］一节内介绍过利用绝对定位设定left属性的负值来隐藏元素的方法，该元素被定位在显示器的显示范围之外，在这种情况下，应该针对页面媒体特别设定CSS规则，例如设定该元素的display属性为“none”。

14.3.2 页面选择器

当文档需要双面打印的时候，左页和右页的页框可能会不同。例如要留出装订线的位置，因此右页的左边距需要宽一些，对应到左页，则应该是右边距宽一些。通过CSS的页面选择器中的:left和:right伪类，可以完成这个需求。



提示： 书籍的第1页是从右页开始的。

所有的页都自动地被用户端归为:left或:right伪类。例如以下代码，其效果如图14-2所示。

```
<style type="text/css" media="print">
<!--
@page :right{
```

```
margin: 2cm 1.5cm 2cm 2.5cm;
}
@page :left{
margin: 2cm 2.5cm 2cm 1.5cm;
}
.....
-->
</style>
<body>
.....
</body>
```

同时，还可以通过:first伪类指定文档第1页的样式。例如上例中增加如下CSS规则，其效果如图14-3所示。

```
@page :first { margin-top: 10cm; }
```

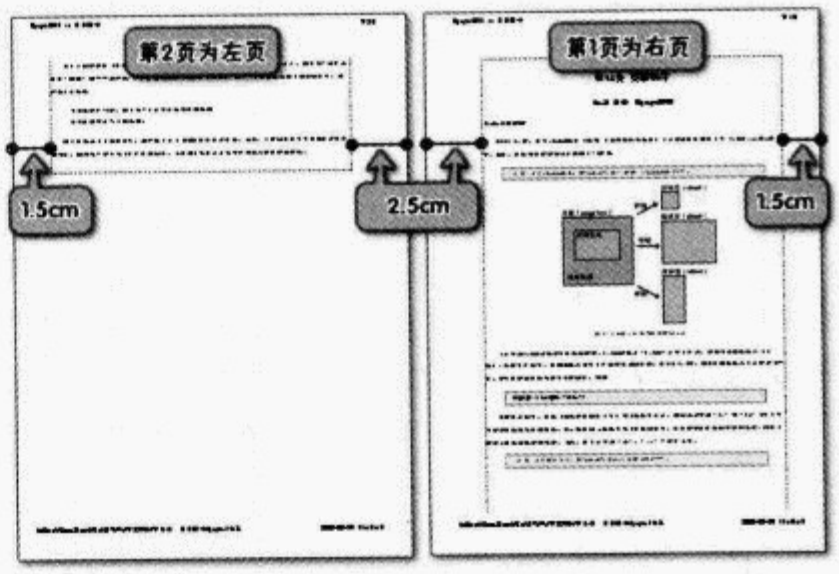


图14-2 利用:left和:right伪类为设置不同的边距

带有伪类的@page规则的优先级高于没有伪类的@page规则，“@page:first”定义的属性高于“@page:left”和“@page:right”。

同时对左页、右页和第1页设定边距可能会使页面区域具有不同的宽度。用户端可能会简化而只使用第1页的设定。

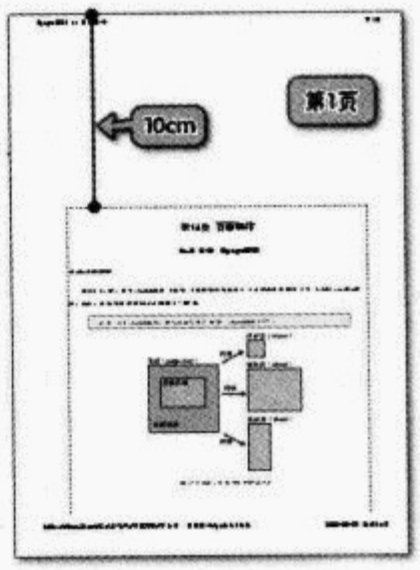


图14-3 :first伪类匹配文档的第1页

14.4 分页

页面媒体的特点就是内容可能需要分页来呈现，分页显示页面可以使内容更具条理性，例如一本书的每章的开始部分，都是从新的一页开始的。

在CSS中，有5个属性规定了用户端在哪里可以或应该分页，在哪一页（左页或右页）上接续后继的内容。每一个分页终止了当前页框的布局，并把文档中剩余的部分在一个新的页框中进行布局。

14.4.1 元素前后分页：page-break-before和page-break-after属性

page-break-before和page-break-after属性控制在元素的前面（before）还是后面分页（after），这2个属性的值是相同的，具体定义列表如下：

语法	page-break-before: auto | always | avoid | left | right | inherit page-break-after: auto | always | avoid | left | right | inherit
说明	设置在元素的前面（before）还是后面分页（after）

续表

值	auto：不强迫也不禁止在生成框之前（之后）分页。 always：总是强迫在生成框之前（之后）分页。 avoid：避免在生成框之前（之后）分页。 left：在生成框之前（之后）分1个或2个页，以使下一页成为一个左页。 right：在生成框之前（之后）分1个或2个页，以使下一页成为一个右页
初始值	auto
继承性	不继承
适用于	块级元素
媒体	图形、页面
计算值	同指定值

例如上例中，部分XHTML代码如下：

```
<body id="c_page">
<div id="wrap">
 <h1>第14章 页面媒体</h1>
 <h2>14.3 页框：@page规则</h2>
 <div id="c_14_3_1">
  <h3>14.3.1 页边距</h3>
  ……
 </div>
 <div id="c_14_3_2">
  <h3>14.3.2 页面选择器</h3>
  ……
 </div>
</div>
</body>
```

如果想让“ ”从新的1页开始打印，则可以添加如下CSS：

```
#c_14_3_2 { page-break-before: always; }
```

其效果如图14-4所示。

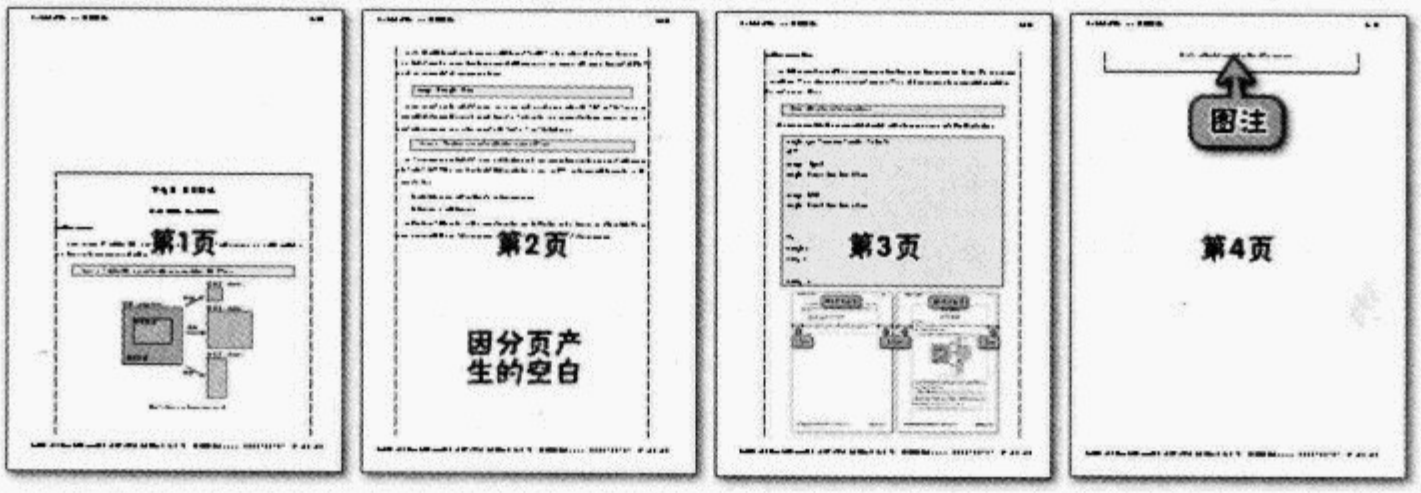


图14-4　分页属性的使用

由图14-4可以发现，虽然第2页仍有空间，但是依旧从“c_14_3_2”元素处重新开始1页。同时，也可发现，在第4页的开始部分是图注，而图注应该是跟在图片后面而不能作为页面的开始，因此可以增加如下CSS，其效果如图14-5所示。

```
.image em {
display : block;
……
page-break-before: avoid;
}
 <p class="image"><img src="img/1402.gif" alt="图14-2" title="图14-2 利用:left和:right伪类为设置不同的边距" width="502" height="347" /><em>图14-2 利用:left和:right伪类为设置不同的边距</em></p>
```

由图14-5可以发现，用户端从图片处开始分页。针对本例这种情况，还有1种解决方法，将在下一小节介绍。

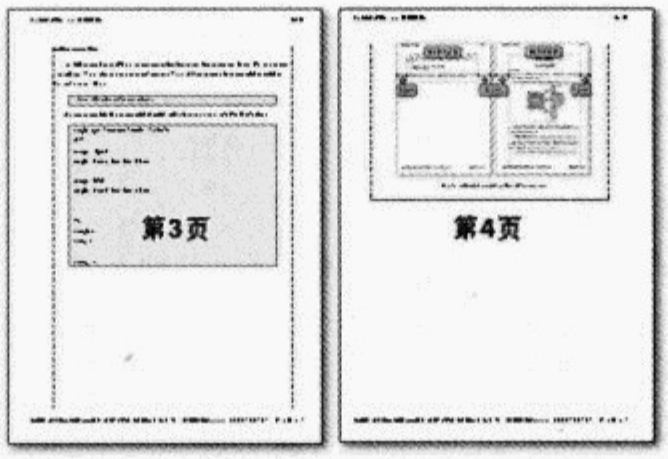


图14-5 不允许元素前分页

14.4.2 元素内部分页：page-break-inside属性

page-break-inside属性控制本元素的内部是否允许分页，具体定义列表如下：

语法	page-break-inside: avoid | auto | inherit
说明	设置在元素的内部是否允许分页
值	auto：不强迫也不禁止在生成框内部分页。 avoid：避免在生成框内部分页
初始值	auto
继承性	继承
适用于	块级元素
媒体	图形、页面
计算值	同指定值

对于上例的情况，在“<p class="image">”内不希望出现分页的情况，因此可以如下设定CSS，其效果如图14-6所示。

```
.image { page-break-inside: avoid; }
```

page-break-inside是可继承的。一个可能的分页位置一般要受到其父元素的page-break-inside属性、前继元素的page-break-after属性以及后续元素的page-break-before属性的影响。如果这些属性值不为“auto”，则“always”、“left”和“right”将优先于“avoid”。

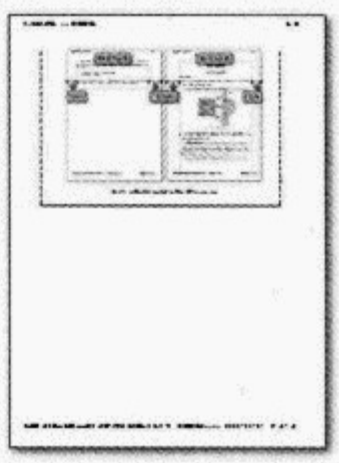


图14-6 通过page-break-inside属性控制元素内部分页

14.4.3 元素内的分割：orphans和widows属性

orphans属性规定了一个段落中必须保留在一个页面底部的行数的最小数目，具体定义列表如下：

语法	orphans: <整数> | inherit
说明	段落中必须遗留在一个页面底部的行数的最小数目
值	整数：最少的行数
初始值	2
继承性	继承.

续表

适用于	块级元素
媒体	图形、页面
计算值	同指定值

widows属性规定了一个元素中必须保留在一个页面顶部的行数的最小数目，具体定义列表如下：

语法	widows: <整数> | inherit
说明	设置元素必须遗留在一个页面顶部的行数的最小数目
值	整数：最少的行数
初始值	2
继承性	继承
适用于	块级元素
媒体	图形、页面
计算值	同指定值

提示：关于内容的格式化，请参见本书［7.3.2 内容区域、行内框和行框］一节。本小节示例代码，读者可参见下载文件包内［/第2部分/第14章：页面媒体/page_break.html］文件。读者可以使用Opera .2的打印预览来测试效果。

假设有某个<p>元素如图14-7所示发生分页：

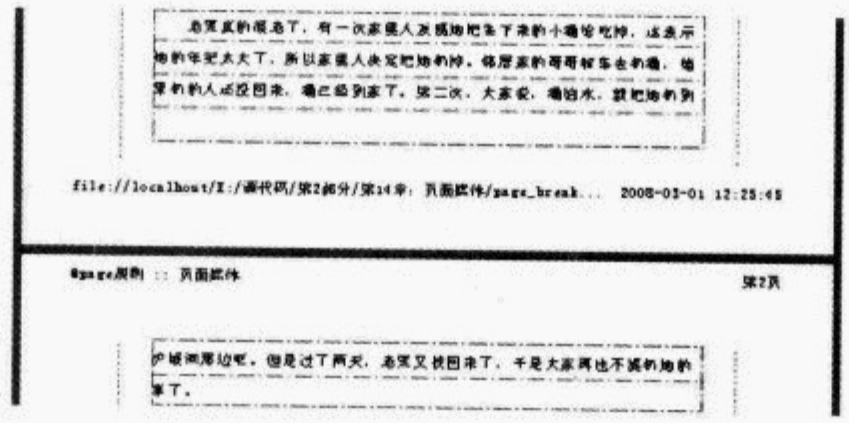


图14-7　orphans和widows属性的初始值效果

由图14-7可以发现，由于orphans和widows属性的初始值都为“2”，因此在段落分页处，虽然第1页最后仍有空间，但是由于元素保留至少2行在第2页的开头，而在第1页最后出现了空行。如果增加CSS规则如下，则其显示如图14-8所示。

```
p { widows : 0; }
```

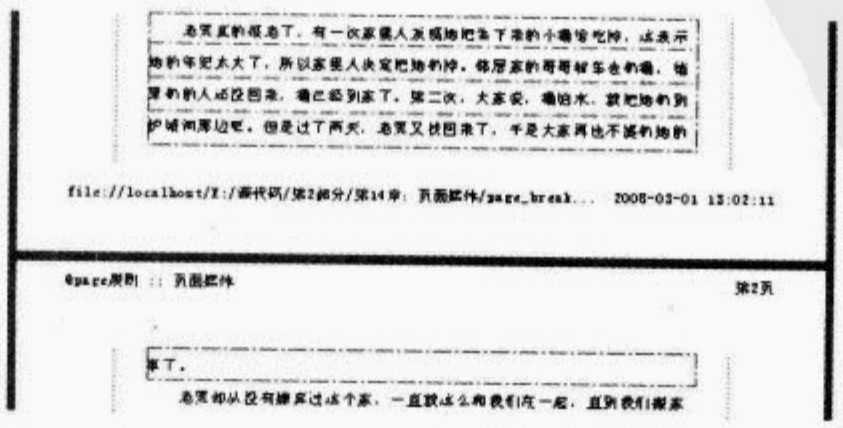


图14-8　设定“p { widows:0; }”后的效果

而如果设置CSS规则如下，则其显示如图14-9所示。

```
p { orphans : 4; }
```

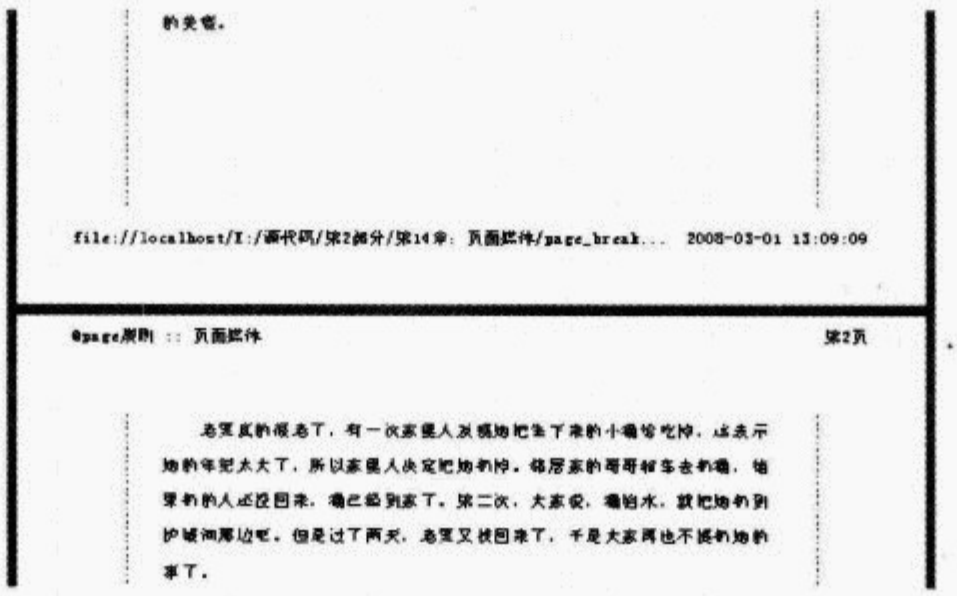


图14-9 设定“p { orphans:4; }”的效果

由于“p { orphans : 4; }”要求在页面底部至少保留4行，而默认的widows属性为2，即在页面顶部至少要保留2行，因此整个段落被下移到第2页显示。

14.4.4 分页的规则

在常规流向中，分页可能发生在下述两种情况中：一种情况是，在块级元素间的垂直方向的边距处产生了分页，则相关的margin-top和margin-bottom属性设置为0；第二种情况是在一个块框中的2个行框之间。

1. 分页的普遍规则

分页需要遵守下列的几条规则。

（1）允许在第一种情况下发生分页的条件是：只有当所有相关元素的page-break-after和page-break-before属性都允许分页的时候，即它们中至少有一个设置了“always”、“left”或“right”，或者它们都是“auto”。

（2）然而，如果它们都是“auto”且所有这些元素的最近的共同前辈设置了“page-break-inside: avoid”，则该处的分页将被禁止。

（3）允许分页在第二种情况下发生的条件是：分页处和包含它们的元素块框起始处之间的行框数量等于或大于orphans属性的值，且分页处和框结尾之间的行框的数量等于或大于widows的值。

（4）另外，在第二种情况下发生分页只有在page-break-inside属性设置为“auto”时才被允许。

如果上述规则无法提供足够的分页点来防止内容溢出页框，则放弃规则（1）、（2）和（4），以寻找另外的分页点。

如果还不能找到足够的分页点，规则（3）将被放弃，以进一步找出更多的分页点。

2. 强制分页

分页必须出现在情况(1)的条件是：如果所有与此处边距相关的元素的生成框的page-break-after和page-break-before属性中，至少有一个设置为“always”、“left”或“right”。

3. “最佳”分页

CSS 2.1没有规定必须使用哪种分页方法，也没有禁止用户端在某个可用的分页点分页，更

没有禁止它不分页。但是CSS 2.1推荐用户端遵循如下建议（由于它们有时互相矛盾，所以称为建议而不是规则）：

- 尽可能少地分页；
- 所有不以强制分页结束的页面的高度要差不多；
- 避免在有边框（border）的块内分页；
- 避免在表格内分页；
- 避免在浮动元素内分页。

例如，有CSS规则如右，且目前距当前页面的底部有20行（行框）。

```
p {
orphans : 4;
widows : 2;
}
```

- 如果<p>元素内容共有20行或更少，则此元素将置于在当前页内。
- 如果这个<p>元素有21或22行，则此元素内会发生分页，但是产生的第2部分不能违反widows属性的定义，即第2部分最少有2行。
- 如果这个<p>元素有23行或者更多，则第1部分为20行，第2部分包含剩余的行。

再例如，有CSS规则如右，而目前距当前页面的底部有8行。

```
p {
orphans : 10;
widows : 20;
}
```

- 如果当前页底部的1个<p>元素内容有8行或更少，则此元素将置于在当前页内。
- 如果这个<p>元素有9行或更多，则它不能被拆分，因为不能违反orphan属性值，因此它将作为1个块整体移到下一页，类似于图14-9所示的情况。

14.5 CSS 2中的属性

有几个包含在CSS 2中的属性，在CSS 2.1中又被删除了，在本节只做简要的介绍。

14.5.1 页框尺寸：size属性

size属性可以定义页框的尺寸和方向，具体定义列表如下：

语法	size: <长度>{1,2} | auto | portrait | landscape | inherit
说明	设置页框的尺寸和方向
值	auto：页框将根据目标页面的尺寸和方向设置。 landscape：超越目标的方向。页框和目标的尺寸一样，较长的边为水平边。 portrait：超越目标的方向。页框和目标的尺寸一样，较短的边为水平边。 <长度>{1,2}：创建一个绝对页框。如果只有1个长度值，则它同时设置了该页框宽度和高度。如果有2个值，则第1个值为宽度，第2个值为高度
初始值	auto
继承性	不继承
适用于	页面上下文
媒体	图形、页面

页框的尺寸可以是“绝对尺寸”（固定尺寸）或“相对尺寸”（可放缩的，即适应可用的页面尺寸）。相对值的页框允许用户端缩放一个文档以更好地利用目标尺寸。

14.5.2 裁切标记：marks属性

在高质量打印中，经常在页框之外加入标记。marks属性指定了十字线或裁切标记或两者是否应该在页框边的外面得到渲染。其具体定义列表如下：

语法	marks: [crop || cross] | none | inherit
说明	设置页框的尺寸和方向
值	auto：页框将根据目标页面的尺寸和方向设置。 none：不生成标记。 crop：生成裁切记号。 cross：生成十字线
初始值	none
继承性	不继承
适用于	页面上下文
媒体	图形、页面

裁剪标记标明页面应该在哪里被裁剪。十字线标记（印刷中的定位标记）用来对齐各个页面。标记只在绝对页框中可见（参见size属性）。相对页框中，页框对齐目标页面而标记将在可打印的区域之外。十字线标记的尺寸，样式和位置取决于用户端。

14.5.3 使用命名的页：page属性

page属性用来指定一个元素应该显示在其上的特定的页面类型。其具体定义列表如下：

语法	page: <标识符> | auto
说明	指定一个元素应该显示在其上的特定的页面类型
值	auto：页框将根据目标页面的尺寸和方向设置。 标识符：页的名称
初始值	auto
继承性	继承
适用于	块级元素
媒体	图形、面

例如某个文档内含有若干表格，而表格比内容宽，作者希望这些表格都能单独分页横向打印，则可以如右定义CSS：

```
table {
page : tables; /* 将表格定义为"tables"页 */
page-break-before : always;
page-break-after: always;
}
@page tables {
size : landscape; /* 横向 */
}
```

14.6 显示器、打印机和投影

显示器、打印机和投影仪都是平时常见并且使用率比较高的设备，在设计样式表的时候，要

注意其各自的特点，以使内容清晰美观。

14.6.1 设备特点

计算机的显示器属于连续视觉类媒体，其特点是色彩丰富，可以调节分辨率，网页的内容可以连续显示。

打印机，包括黑白和彩色，其打印原理也有针式、喷墨、激光以及热感应等，不同类型的打印机打印的精度可能相差很大。同时，打印的内容外观还受纸张大小、颜色的限制。

投影仪一般用于会议和课堂等，由于需要像很多人展示内容，由于受投影仪分辨率和面积的限制，其可供显示的区域可能不会很大。

14.6.2 设计要点

针对设备的不同特点，在设置CSS的时候需要特别注意以下几点。

1. 字体

虽然字体的设定受到操作系统的限制，但是，有几个基本的字体是各系统普遍会提供的，如“宋体”、“黑体”、“Times New Roman”等。关于字体，请参见本书［6.1字体集：font-family属性］一节。

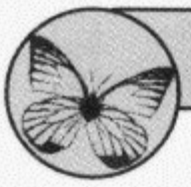
对于显示器，中文的宋体可以显示得很清晰，而英文的衬线系列字体也易于辨认。但是对于投影仪，由于需要照顾比较远的观众，因此比较粗的黑体和无衬线系字体，也许更醒目。

2. 字体尺寸

对于文字内容，要特别注意的就是字体的尺寸。12px对于显示器可能已经足够清晰，但是对于打印机就可能比较细小，而对于投影仪，则远处的观众也许根本辨认不清。因此针对不同的设备，设置不同字号是一个明智的选择。

3. 单位

px是相对单位，其真实的尺寸和设备的dpi相关。dpi指每平方英寸上，所印刷的网点数（Dot Per Inch），因此“实际尺寸 = 像素尺寸÷设备的dpi”，因此相同的100px宽的图片，在不同的打印机上实际大小可能会相差很多。因此针对打印机，最好使用绝对单位。



提示：关于相对单位和绝对单位，请参见本书［5.3.2 长度单位］一节。

4. 前景和背景

在本书的［10.1 颜色基础］一节介绍过：“色相”、“明度”和“纯度”称为色彩的三属性，明度和色相合并为二线的色状态，称为色调。

通过背景可以美化网页，不同的文字颜色则更可以区分内容，但是，如果前景色和背景色的明度相同而色相不同，在彩色设备上看也许是没有问题的，但是在黑白的设备上，可能会变成同一种颜色，如图14-10所示。

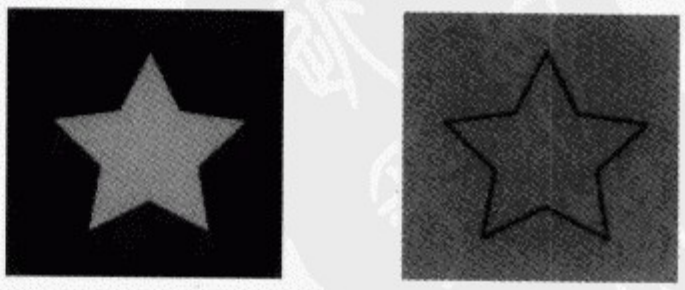


图14-10　不同色相的颜色在无彩色的情况下可能会变成同一种颜色

同时，漂亮的图片背景，对于打印也许不适合，由于打印机精度的限制，精度比较低的打印机，有可能会将背景与文字混成一片，因此很多浏览器默认的打印设置里面是不打印背景图片和颜色的，此时，对于屏幕上深色背景上的浅色文字来说，可能打印以后很难辨认。

因此针对打印机不设置背景比较明智。同时，可以如右定义前景色和背景色，这样会将所有元素的背景设定为白色，而文字为黑色。

```
* {
color: black !important;
background: white !important;
}
```

提示：CSS 2.1中不能为不同类型的打印机设定不同的样式，因此要考虑到比较常见的情况，而在CSS 3的“媒体询问模块”中，可以为彩色和黑白的打印机分别设定样式，但是本书完稿之日止，还没有实现。

但是对于投影仪，其灯泡的亮度、荧幕的质量等，都会影响到观看效果，因此高对比度的前景和背景，也许更加容易阅读。

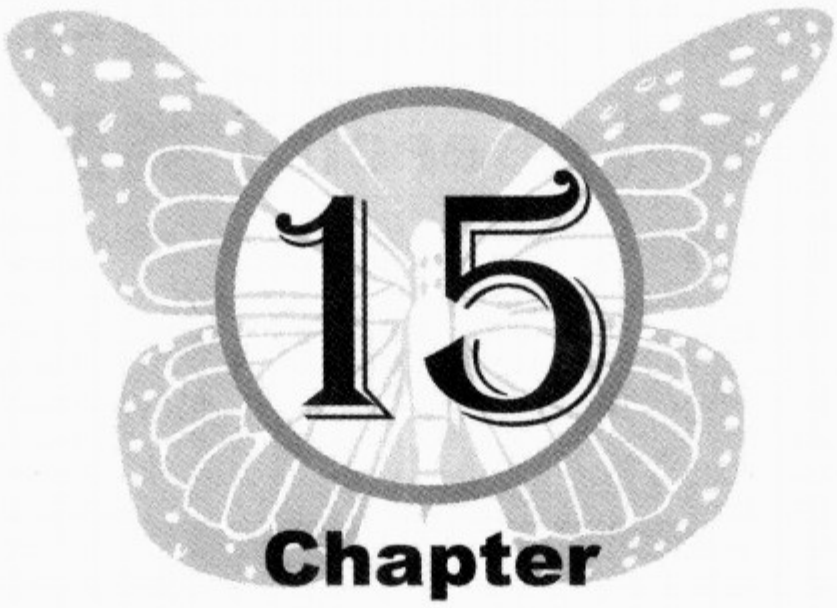
5. 定位和浮动的元素

在前面的章节曾经介绍过，绝对定位的元素可能会在页面区域以外，因此可能会被裁切或者在最后打印，这可能会导致纸张的浪费或者内容的不连贯，因此，如果不需要打印定位的元素，最好为其设定“display：none”。

通过浮动而并排显示的元素，可能由于内容区域内的宽度与显示器的分辨率不同而变成了上下关系，因此，在打印样式中最好不要使用浮动。

6. 内容

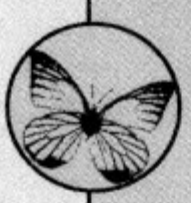
使用过Microsoft公司出品的PowerPoint软件的读者应该知道，幻灯片式的文档其每页的内容区域非常有限，同时，为了保证内容的易读，字体一般设置得都比较大，因此幻灯片一般用来展示提纲或者图片。



第15章 听觉样式表

听觉样式主表要是针对可以发音的用户端（例如屏幕阅读机），这主要为视力有障碍的用户或者无法使用视觉媒体的用户提供服务。

听觉样式表是CSS 2规范的1个正式章节，但是在CSS 2.1中，它被移动到了文档附录中，作为资料提供，CSS 2.1不要求用户端遵守这些规定以求符合CSS 2.1。CSS 2.1虽然指出在将来的版本中会更好地支持“语音（speech）设备”，却语焉不详。因此在本章中，将只对相关属性作简要的介绍。



提示：屏幕阅读器是一种结合扬声器或点字显示器，通过播放或摸读的方式来读取电脑屏幕里包括图示（图片的“alt”属性）、功能表、文字、符号和控制按钮等所有资讯的软件。比较著名的是Jaws屏幕阅读器（Jaws Screen Reader Opens a new browser window）和Emacspeak（Linux版）。

15.1 听觉（aural）类型与语音（speech）类型

在本书［3.2.3. 4 media属性］一节内介绍的“speech”类型媒体，就是语音设备。而在CSS 2中没有“speech”类媒体，而是使用“aural（听觉）”类媒体。

在CSS 2.1中只是预留了“speech”媒体类型，却没有指定哪些属性可与其相配合使用。因此在本章后续内容的属性定义列表中，“适用于”一项，仍沿用CSS 2中的“听觉”类媒体的说法。

15.1.1 链接听觉样式的特点

使用听觉属性的时候，实际包含了一个三维的物理空间（声音的环绕）和一个时间维度（可以指定一个声音在另一个声音之前、之中、之后）。CSS属性同样允许作者改变合成语音的质量（嗓音类型、频率、变形等）。例如右列代码：

表示使用一种叫作“Paul”的嗓音（可以称为某种“音频字体”）来朗读页面内的文字。

```
@media speech {
 body { voice-family: Paul }
}
```

15.1.2 与听觉属性相关的值

在本书［5.8 其他值］一节内介绍的3种属性值与听觉属性相关：角度、时间和频率，不熟悉的读者可以参考这3种属性的说明。

15.2 音量属性：volume属性

可以使用volume属性来控制语音的音量，具体定义列表如下：

语法	volume: <实数> | <百分比> | silent | x-soft | soft | medium | loud | x-loud | inherit
说明	设置语音的音量
值	实数：任何在0~100的数值，0表示最小可闻音量而100对应最大可适音量。 百分比：百分比值相对继承值进行计算，然后剪裁到0~100的区间。 silent：没有声音，取值0并不意味着“无声”。 x-soft：与0一样。 soft：与25一样。 medium：与50一样。 loud：与75一样。 x-loud：与100一样
初始值	medium
继承性	继承
适用于	所有元素
媒体	听觉
计算值	实数

音量参照于波形的中值音量。换句话说，一个高度扭曲的嗓音的音量为50，但实际可能远远

超过这个数值。总体数值可以由人工调节到舒服的层次，例如通过物理的音量调节（它会同时等比提高0和100的值），该属性所做的只是调节动态范围。

用户端应该允许对应于0和100的值由听众来设置。没有一个设置是放诸四海而皆准的；合适的值取决于适用的设备（扬声器或者耳机）、环境（车内、家庭影院或者图书馆）以及个人爱好。可以举一些例子。

- 作为车内适用的浏览器的设置需要考虑到有很多背景噪音。0会映射到一个相对较高而100会映射到很高的音量级别。
- 另一个语音浏览器用在夜深人静的公寓中，或公共阅读室等安静的环境中。0会被设置为非常轻而100设置为相对较轻的级别。
- 在一个安静并隔音的房子中，高档家庭影院的设置，应该是0比较低，而100比较高，动态范围很宽。

所有的情形下，可以应用相同的作者样式表，只要简单的将0和100的值映射到适合客户端的状态就可以了。例如可以如右定义：

```
body { volume : 40; }
div {volume: 80%;}
```

15.3 发音：speak属性

可以使用speak属性来控制元素的内容是否需要读出来，具体定义列表如下：

语法	speak: normal | none | spell-out | inherit
说明	设置元素的内容是否需要读出来
值	none：禁止音频渲染，从而该元素不需要时间渲染。 normal：使用与语言有关的发音规则渲染一个元素和它的子元素。 spell-out：将文本一个字母一个字母的拼出（对缩写和简写词很有用）
初始值	normal
继承性	继承
适用于	所有元素
媒体	听觉
计算值	同指定值

元素的volume属性为“silent”以及元素的speak属性为“none”之间的区别是：前者和讲出它（包括任何之前之后的停顿）需要一样的时间，只是不出声。后者不需要任何时间也不会被渲染（尽管它的后代可能会）。

speak属性为“none”的元素的后代元素可能超越该值而被读出（要确保一个元素及其后代不被渲染，使用display属性）。例如可以如下定义，文字“跳转到”不会被读出，而“首页”则会正常读出。

```
#nav { speak : none; }
#nav_home { speak : normal; }
<div id="nav">
跳转到：<a href="home.html" id="nav_home">首页</a>
</div>
```

但是如果修改CSS如下，则整个“nav”层都会被忽略。

```
#nav { display : none; }
```

15.4 暂停：pause-before、pause-after和pause属性

可以使用pause-before属性来使语音在元素前暂停，具体定义列表如下：

语法	pause-before: <时间> | <百分比> | inherit
说明	设置语音在元素前暂停
值	时间：以绝对时间单元（秒及毫秒）表示暂停。 百分比：相对于speech-rate属性值的倒数。例如，如果说话速度是120字每分钟（也就是，一个字要半秒钟，或500ms），那么pause-before为100%意味着暂停500 ms而pause-before为20%就是100ms
初始值	0
继承性	不继承
适用于	所有元素
媒体	听觉
计算值	时间值

可以使用pause-after属性来使语音在元素后暂停，具体定义列表如下：

语法	pause-after: <时间> | <百分比> | inherit
说明	设置语音在元素后暂停
值	时间：以绝对时间单元（秒及毫秒）表示暂停。 百分比：相对于speech-rate属性值的倒数。例如，如果说话速度是120字每分钟（也就是，一个字要半秒钟，或500ms），那么pause-before为100%意味着暂停500 ms而pause-before为20%就是100ms
初始值	0
继承性	不继承
适用于	所有元素
媒体	听觉
计算值	时间值

暂停插入在元素内容和任何cue-before或cue-after内容之间。推荐使用相对单位来创建更健壮的样式表，因为说话速度可能有很大变化。pause属性是缩写属性，即将前面2个暂停属性缩写，具体定义列表如下：

语法	pause: [[<时间> | <百分比>]{1,2}] | inherit
说明	设置语音的暂停
值	时间：以绝对时间单元（秒及毫秒）表示暂停。 百分比：相对于speech-rate属性值的倒数
初始值	同各属性
继承性	不继承
适用于	所有元素
媒体	听觉
计算值	同各属性

如果pause属性有2个值，则第1个值为pause-before的值，而第2个值为pause-after的值。如果只有1个值，则视为2个属性的值相同。例如：

```
h1 { pause: 20ms } /* pause-before: 20ms; pause-after: 20ms */
h2 { pause: 30ms 40ms } /* pause-before: 30ms; pause-after: 40ms */
h3 { pause-after: 10ms } /* pause-before unspecified; pause-after: 10ms */
```

15.5 提示：cue-before、cue-after和cue属性

使用音频图标是区别语义要素的一种方式。cue-before属性是在元素前面插入音频标记，具体定义列表如下：

语法	cue-before: <uri> | none | inherit
说明	设置元素前面插入的音频标记
值	uri：URI必须指向一个音频标记资源。如果URI解析指向非音频文件，如图形，那么该资源将被忽略而且处理该属性时就如同其取值为“none”。 none：未指定音频标记
初始值	none
继承性	不继承
适用于	所有元素
媒体	听觉
计算值	绝对的URI或者“none”

cue- after属性是在元素后面插入音频标记，具体定义列表如下：

语法	cue-after: <uri> | none | inherit
说明	设置元素后面插入的音频标记
值	uri：URI必须指向一个音频标记资源。如果URI解析指向非音频文件，如图形，那么该资源将被忽略而且处理该属性时就如同其取值为“none”。 none：未指定音频标记
初始值	none
继承性	不继承
适用于	所有元素
媒体	听觉
计算值	绝对的URI或者“none”

在元素之前和（或）之后加入声音标记，这样可以区分不同的元素。例如：

在链接和<h1>元素开始和结束的时候，都会播放相应的声音。这2个属性也有缩写属性cue，具体定义列表如下：

```
a { cue-before: url("bell.aiff"); cue-after: url("dong.wav") }
h1 {cue-before: url("pop.au"); cue-after: url("pop.au") }
```

语法	cue: [<cue-before> || <cue-after>] | inherit
说明	设置元素的音频标记
值	见各属性
初始值	见各属性

续表

继承性	不继承
适用于	所有元素
媒体	听觉
计算值	同各属性

cue属性如果有2个值，则第一个值是cue-before的值，第2个值是cue-after的值；如果只有1个值，则等于为2个属性设定相同值。例如右边声明是等价的：

```
h1 {cue-before: url("pop.au"); cue-after: url("pop.au") }
h1 {cue: url("pop.au") }
```

15.6 混音：play-during属性

和cue-before和cue-after属性类似，play-during属性指定了当某元素的内容被读出时播放的声音，具体定义列表如下：

语法	play-during: <uri> [mix || repeat]? | auto | none | inherit
说明	设置元素的内容被读出时播放的声音
值	uri：在说出元素内容时，播放<uri>指定的声音。 mix：如果指定该值，意味着从父元素的play-during属性继承来的声音继续播放，而<uri>指定的声音会与之混合。如果不指定“mix”，该元素的背景声将替换父元素的。 repeat：如果指定该值，如果声音长度不足以填充该元素的持续时间，则重复播放；否则，声音仅播放一次而停止（和background-repeat属性类似）；如果声音对于元素而言过长，那么一旦元素内容被讲完，它会被剪裁。 auto：父元素的声音继续播放（但不是重新开始，如果属性被继承，则是重新开始播放）。 none：父元素的声音（如果有的话）在该元素期间无声，在当前元素之后继续
初始值	auto
继承性	不继承
适用于	所有元素
媒体	听觉
计算值	绝对的URI，其余的同指定值

play-during属性可以设置类似“背景”的效果，还可设定元素与其父与元素同时存在声音时，如何处理。例如：

```
blockquote.sad { play-during: url("violins.aiff") } /* 类为"sad"的<blockquote>元素播放"violins.aiff"文件 */
blockquote q { play-during: url("harp.wav") mix; } /* <q>播放"harp.wav"，同时从<blockquote>继承的声音也继续播放 */
span.quiet { play-during: none } /* 类为"quiet"的<span>元素不播放父元素的声音。 */
```

15.7 空间：azimuth和elevation属性

空间音频是音频呈现的一个重要的样式属性。它提供了一种自然的方式来区别各个声音，就

像在生活中那样（人们很少会都呆在房间里的同一个地方）。立体声喇叭可以产生一个横向的声音平台。双耳耳机或者越来越流行的5喇叭家庭影院设置可以生成完全的环绕声音。多喇叭设置可以创造一个真实的3D声音平台。

azimuth属性指定了声音在播放时所处水平空间中的角度，具体定义列表如下：

语法	azimuth: <角度> | [[left-side | far-left | left | center-left | center | center-right | right | far-right | right-side] || behind] | leftwards | rightwards | inherit
说明	设置音频在空间内的角度
值	角度：定位以角度表示，范围为-360deg~360deg。0deg表示正对声音平台正中的前方。90deg是右，180deg在后，而270deg（或者-90deg）是左。 left-side：等于270deg。和behind连用时为270deg。 far-left：等于300deg。和behind连用时为240deg。 left：等于320deg。和behind连用时为220deg。 center-left：等于340deg。和behind连用时为200deg。 center：等于0deg。和behind连用时为180deg。 center-right：等于20deg。和behind连用时为160deg。 right：等于40deg。和behind连用时为140deg。 far-right：等于60deg。和behind连用时为120deg。 right-side：等于90deg。和behind连用时为90deg。 leftwards：将声音相对当前角度左移，更精确地说，是减20度（以360度为模）。“leftwards”是“逆时针”，因为它总是减20度，即使继承的角度已经在听众背后，也就是声音实际上是向右边移。 rightwards：将声音相对当前角度右移，更精确地说，是加20度（参照“leftwards”）。 behind：表示与设定角度相对应在听众后面的角度
初始值	center
继承性	继承
适用于	所有元素
媒体	听觉
计算值	正常的角度值

各关键字对应的角度如图15-1所示。

要实现该属性很可能是通过以不同音量在不同声道中混合同一个信号而实现的。也可以通过相位平移，数字延迟以及其它技术来提供声音平台的假象。达到这个效果的确切方法以及使用的音箱数量取决于用户端；该属性很少指明需要的终端效果。例如：

```
h1  { azimuth: 30deg }
td.a { azimuth: far-right }        /* 60deg */
#12  { azimuth: behind far-right } /* 120deg */
p.comment { azimuth: behind }     /* 180deg */
```

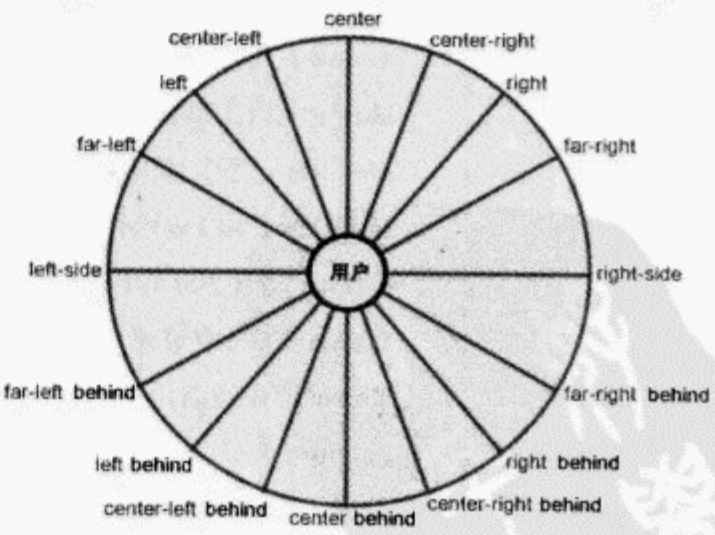


图15-1　azimuth属性关键字值对应的角度位置

elevation属性和azimuth属性很像，只是它指定的是垂直方向的空间角度，具体定义列表如下：

语法	elevation: <角度> | below | level | above | higher | lower | inherit
说明	设置语音在垂直方向的角度

续表

值	角度：以角度指定抬升位置，取值在-90deg和90deg之间。0deg意味着在前方水平面，大致就是和听众水平的平面。90deg意味着在头顶，而-90deg意味着在脚底。 below：等于-90deg。 level：等于0deg。 above：等于90deg。 higher：在当前角度加10度。 lower：在当前角度减10度
初始值	level
继承性	继承
适用于	所有元素
媒体	听觉
计算值	正常的角度值

CSS中并未定义实现这些效果的确切方法和使用多少扬声器。该属性只是指出了需要的终端效果。

15.8 语音特征属性

CSS可以控制语音特征的，包括以下几个属性：speech-rate、voice-family、pitch、pitch-range、stress和richness。

1. 速率：speech-rate

speech-rate属性可以设置语音的速率，即每分钟读多少个字，具体定义列表如下：

语法	speech-rate: <数字> | x-slow | slow | medium | fast | x-fast | faster | slower | inherit
说明	设置语音发音的速率
值	数字：以每分钟字数（WPM）来指定说话速率。语种不同，具体速率也不一样。但是不管任何语音合成器都广泛支持。 x-slow：等于80 WPM。 slow：等于120 WPM。 medium：等于180~200 WPM。 fast：等于300 WPM。 x-fast：等于500 WPM。 faster：在当前语速上加40 WPM。 slower：在当前语速上减40 WPM
初始值	medium
继承性	继承
适用于	所有元素
媒体	听觉
计算值	数字

该属性指定了说话速率，可以使用绝对和相对关键字值（和font-size比较类似）。

2. 嗓音：voice-family

和为文字指定字体集font-family属性类似，声音也可以通过voice-family属性来设置用什么样的嗓音来朗读。具体定义列表如下：

语法	voice-family: [[<特殊语音>	<普通语音>],]* [<特殊语音>	<普通语音>]	inherit
说明	设置语音的嗓音
值	普通语音：取值为语音家族，可能的值有“male”、“female”及“child”。 特殊语音：取值为特定的实例（例如，喜剧演员，Trinoids、Carlos、Lani）
初始值	取决于用户端
继承性	继承
适用于	所有元素
媒体	听觉
计算值	同指定值

voice-family和font-family在某些地方非常相似，例如其值可以为多个以英文逗号“,”分隔的值和优先排列的语音家族列表。例如以下代码：

```
h1 { voice-family: announcer, male }
p.romeo { voice-family: romeo, male }
p.juliet { voice-family: juliet, female }
```

特定嗓音的名字可以加引号。如果构成名称的任何单字不符合表征化语法规则的话，则必须加引号。如果一个名称包含超过一个单字，也推荐用引号分割。如果不加引号，在嗓音前后的任何空白字符都被忽略，而且嗓音名称内任何序列的空白都会转化为单一空格。

3. 音高：pitch

声音是一种振动波，具有振动频率，也就是常说的音调、音高，频率高音调就高亢，频率低，音调就低沉。pitch属性可以设定发音的音高。pitch属性具体定义列表如下：

语法	pitch: <频率>	x-low	low	medium	high	x-high	inherit
说明	设置嗓音的频率
值	频率：以赫兹（Hz）指定说话嗓音的频率。 x-low, low, medium, high, x-high：这些值并不映射到绝对的频率，因为这取决于语音家族。用户端根据语音家族和用户环境将这些值映射到合适的频率
初始值	取决于用户端
继承性	继承
适用于	所有元素
媒体	听觉
计算值	频率值

频率高低主要取决于语音家族，例如，标准男声是120Hz左右，而女声大约是210Hz。例如：

```
h1 {
voice-family: Jethro, Susie;
pitch: 100Hz;
}
```

4. 音高范围：pitch-range

人在说话时往往带有高低可变的语调，以强调重点或者表示感情，pitch-range属性可以设置音高变化的动态范围，具体定义列表如下：

语法	pitch-range: <数字>	inherit
说明	设置音高的变化范围

续表

值	数字：介于0~100之间的数字
初始值	50
继承性	继承
适用于	所有元素
媒体	听觉
计算值	同指定值

一种活泼的声音，也就是高度变形的声音，会呈现高的音高范围。pitch-range属性指定了这些变化的范围。设定pitch-range属性值为0，会产生一种平直而单调的声音；设定为50生成通常的变形；超过50会生成活泼的嗓音。

5. 重音：stress

stress属性用以设置声音波形的最高峰值，具体定义列表如下：

语法	pitch-range: <数字> | inherit
说明	设置声音波形的最高峰值
值	数字：介于0~100之间的数字
初始值	50
继承性	继承
适用于	所有元素
媒体	听觉
计算值	同指定值

stress属性指定了语音的语调轮廓的“局部高峰”的高度。例如，英语是一个有强调的语种，一句话中的不同部分有首要，次要，再次的强调之分。stress的值控制着基于这些强调标记的变形数量的结果。该属性是pitch-range属性的伴随属性，用来允许开发者开发高端音频渲染。

数值的含义取决于发音的语种。例如，数值为50对于标准的英语男声（平均锐度 = 122 Hz），以正常语调和强调说话而言，和意大利语音中取值50的含义就不一样。

6. 音色：richness

在嘈杂的环境，圆润洪亮的声音也可以让很多人听得很清晰，而柔和的声音可能就听不清。richness属性可以设定音色，具体定义列表如下：

语法	richness: <数字> | inherit
说明	设置音色
值	数字：介于0~100之间的数字
初始值	50
继承性	继承
适用于	所有元素
媒体	听觉
计算值	同指定值

属性的取值介于0~100之间。取值越高，声音会越洪亮。较低的值产生柔和甜蜜的声音。

15.9 语音：speak-punctuation和speak-numeral属性

speak-punctuation属性设置是否读出标点符号，具体定义列表如下：

语法	speak-punctuation: code | none | inherit
说明	设置是否读出标点符号
值	code：诸如分号、花括号等都按原样读出。 none：不读出标点符号，但是会渲染成自然的停顿
初始值	none
继承性	继承
适用于	所有元素
媒体	听觉
计算值	同指定值

speak-numeral属性设置如何读出数字，具体定义列表如下：

语法	speak-numeral: digits | continuous | inherit
说明	设置如何读出数字
值	digits：数字被读成独立的各个位，因此237读成“2、3、7”。 continuous：将数字读成一个数。因此，237读成“二百三十七”。用什么单字读出取决于语种
初始值	continuous
继承性	继承
适用于	所有元素
媒体	听觉
计算值	同指定值

15.10 叙述表头：speak-header属性

当语音设备遇到表格的时候，表头单元格<th>和数据单元格<td>的关系必须以不同于水平和垂直方向对齐方式不同的方式读出。

有些语音浏览器可能允许用户以二维的方式在表格中移动，因而给予它们对应以空间表示的关系的可能。如果这是可能的，样式表必须指出表头要在哪个点上读出。

speak-header属性设置表格头是否在每个单元格前说出，或只是在某个单元格关联的头与前一个不同时说出，具体定义列表如下：

语法	speak-header: once | always | inherit
说明	设置如何读出表头与单元格的联系
值	once：表头只在一系列单元格之前读出1次。 always：表头在相关的单元格前读出

续表

初始值	once
继承性	继承
适用于	具有表头信息的元素
媒体	听觉
计算值	同指定值

每个文档语言可能有不同的机制允许作者指定头。例如，在HTML 4.0中，可以有3种不同的属性来指定头信息——headers、scope以及axis。如果这些属性没有被指定，本规范提供了一种算法来确定头信息。例如有如下表格代码，其显示如图15-2所示。

```
<table summary="样例表格，abbr属性。" id="sample5">
 <caption>
 样例表格5：abbr属性
 </caption>
 <tr>
  <th scope="col">年</th>
  <th scope="col">月</th>
  <th scope="col">数量</th>
  <th scope="col">价格</th>
 </tr>
 <tr>
  <th rowspan="2" axis="2007年">2007年</th>
  <th headers="2007年">1月</th>
  <td>1000</td>
  <td>1500</td>
 </tr>
 <tr>
  <th headers="2007年">2月</th>
  <td>2000</td>
  <td>3000</td>
 </tr>
 <tr>
  <th rowspan="2" axis="2008年">2008年</th>
  <th headers="2008年">1月</th>
  <td>1200</td>
  <td>1560</td>
 </tr>
 <tr>
  <th headers="2008年">2月</th>
  <td>24000</td>
  <td>3600</td>
 </tr>
</table>
```

价目表

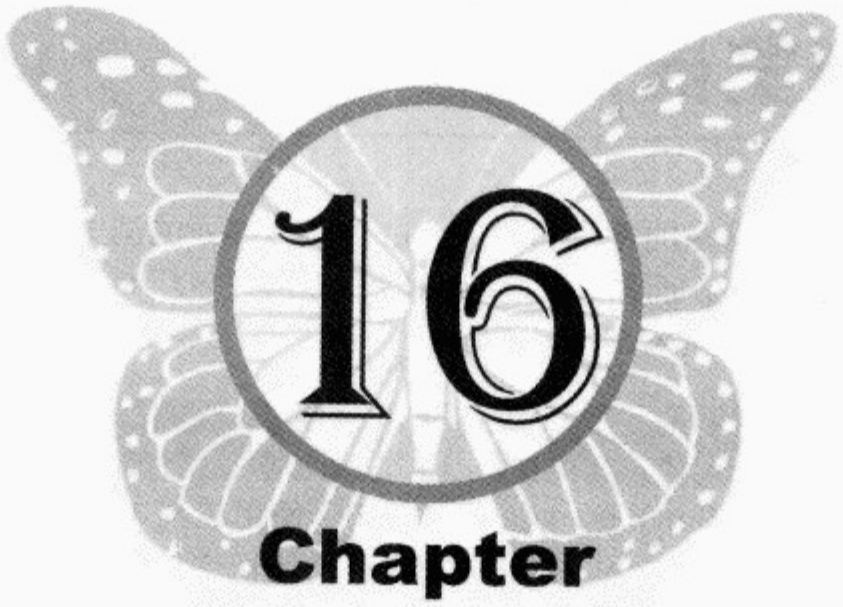
年	月	数量	价格
2007年	1月	1000	1500
	2月	2000	3000
2008年	1月	1200	1560
	2月	24000	3600

图15-2 表格的示意图

以这种方式提供数据模型，就可以通过CSS使语音设备可以用非常丰富的方法来浏览这个表格，例如以下左边表格，每个单元格可以作为列表说出，在每个数据单元格之前重复适用的头，或者也可以采用以下右边的形式。

```
2007年，1月，数量：1000
2007年，1月，价格：1500
2007年，2月，数量：2000
2007年，2月，价格：3000
```

```
2007年，1月，数量：1000
          价格：1500
      2月，数量：2000
          价格：3000
```



第16章 浏览器与Hack

CSS虽然有很强的能力，但是所有效果都是基于用户端（例如浏览器）的支持。目前常用的浏览器对于CSS 2.1的支持并不完全，特别是用户最为广泛的Windows IE 6.0（它甚至连CSS 2都没有完全支持），这无疑阻挡了CSS的能力发挥，同时，对于CSS规范理解的不同、软件本身编写时存在的错误等，都会导致相同的CSS在不同的浏览器内呈现截然不同的表现，这点在前面章节的例子中，读者已有所了解。本章将介绍浏览器常见的问题和解决方法。

16.1 浏览器简介

可以说，浏览器推动了网络的普及和发展。由于图形界面的浏览器的广泛应用，使得越来越多的普通用户也能够使用网络，而作为软件产品的一种，很多厂商也在这个领域里争夺用户，目前竞争最为激烈的，就是IE和Firefox。

16.1.1 浏览器的发展

蒂姆·伯纳斯-李（Tim Berners-Lee）是第一个使用超文本来分享资讯的人。他于1990年发明了首个网页浏览器WorldWideWeb。在1991年3月，他把这发明介绍给了给他在CERN工作的朋友。从那时起，浏览器的发展就和网络的发展联系在了一起。

NCSA Mosaic使互联网得以迅速发展。它最初是一个只在UNIX运行的图像浏览器；很快便发展到在Apple Macintosh和Microsoft Windows亦能运行。

NCSA中Mosaic项目的负责人Marc Andreesen辞职并建立了网景通讯公司。网景公司在1994年10月发布了他们的旗舰产品网景导航者，但第二年Netscape的优势就被削弱了。错失了互联网浪潮的微软公司在这个时候匆促购入了Spyglass公司的技术，改成Internet Explorer，掀起了软件巨头微软和网景之间的浏览器大战。

1998年，网景公司承认其市场占有率已无法挽回，微软能取胜的其中一个因素是它把浏览器与其操作系统一并出售（OEM，原始设备制造），这亦使它面对反垄断诉讼。网景公司以开放源代码迎战，创造了Mozilla。Mozilla 1.0的出现被视为其里程碑。同年，衍生出Phoenix（后改名Firebird，最后又改为Firefox）。Firefox 1.0于2004年发表。

Opera发布于1996年，是一个小巧的浏览器，Windows版的Opera安装后只占不到10MB的空间，但是却功能丰富。正是由于它的“体积”很小，使它在手持电脑以及手机等存储容量有限制的设备上十分流行，而在个人电脑网络浏览器市场上的占有率则很小。

Lynx浏览器仍然是Linux市场上十分流行的浏览器。它是全文字模式的浏览器，视觉上并不讨好。还有一些有着进阶功能的同类型浏览器，例如Links和它的分支ELinks。

Konqueror是一个由KDE开发的浏览器，这个浏览器使用了自家开发的解释引擎KHTML，由于Konqueror只常见于Unix-like下的KDE桌面环境，所以并未普及。

纵然Macintosh的浏览器市场现在亦同样被Internet Explorer和Firefox占据，但未来有可能会是苹果电脑自行推出的Safari的世界。Safari是基于Konqueror这个开放源代码浏览器的KHTML解释引擎而制成的，它是Mac OS X的默认浏览器，目前也推出了Windows版。

16.1.2 浏览器的解释引擎

浏览器可以简单地看作由2个部分组成：解释引擎（Rendering Engine）和用户界面（User Interface）。

解释引擎也被通俗地称为浏览器“内核”，是浏览器的关键所在，它决定了如何将(X)HTML、CSS和JavaScript等转换成访问者看到的样子。而用户界面负责同用户进行交互。

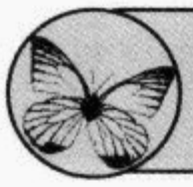
不同的解释引擎造成了相同网页的不同表现，而使用相同引擎的浏览器，也许它们的用户界面千差万别，但是在网页的表现上总是类似的。比较常见的引擎有以下几种。

1. Trident

Trident（又称为MSHTML），是微软开发的一种解释引擎。它的第1个版本随着1997年10月Windows Internet Explorer第4版的发布而发布。随后，Trident不断地被更新和完善。

不光是IE，Windows中的许多地方也使用了Trident的技术，包括Windows Explorer、Windows 98及其后续所有版本的视窗操作系统内的Windows Help程序。此外，像Windows Media Player、Windows Live Messenger、Outlook Express等也使用了Trident技术。也正因为如此，Trident无法从Windows系统中彻底地卸载掉，就算是彻底卸载了，Windows的许多功能也会出现问题。

Windows Internet Explorer、Maxthon（傲游，双内核）、Tencent Traveler（腾讯公司）都是基于这个引擎。



提示：运行在苹果操作系统Mac OS上的Mac版IE使用的是与Windows版不同的Tasman内核，因此其表现也与Windows不尽相同。

2. Gecko

Gecko是套开放源代码的、以C++编写的网页解释引擎。这软件原本是由网景通讯公司开发的，现在则由Mozilla基金会维护。

由于Gecko是开源的，所以使用Gecko的网页浏览器也很多，其中的领军人物便是Mozilla Firefox。此外还有AOL for Mac OS X、Camino、Epiphany、Flock、Galeon、Netscape Browser、Sleipnir、Maxthon（傲游，双内核）等。

3. WebCore / KHTML

WebCore是苹果公司开发的解释引擎，它是在另外一个解释引擎“KHTML”的基础上而来的。苹果电脑于2002年采纳了KHTML，作为开发Safari浏览器之用，并发布所修改的最新及过去版本源代码。后来发表了开放源代码的WebCore及WebKit引擎，它们均是KHTML的衍生产品。使用WebCore的主要有Safari，此外还有OmniWeb、Shiira、Swift等。

4. Presto

Presto是一个由Opera Software开发的浏览器解释引擎，供Opera 7.0及以上使用。此外，Macromedia Dreamweaver（MX版本及以上）和Adobe Creative Suite 2也使用了Presto的内核。

16.1.3 浏览器的工作模式

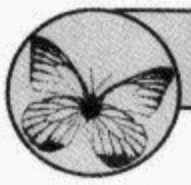
当浏览器厂商开始创建与标准兼容的浏览器时，他们希望确保向后兼容性。为了实现这一点，他们创建了两种表现模式：标准模式（Standards Mode或Strict Mode））和怪异模式（Quirks Mode）。

在标准模式中，浏览器根据W3C规范表现页面；在怪异模式中，页面以一种比较宽松的向后兼容的方式显示。怪异模式通常模拟老式浏览器（比如Microsoft IE 4和Netscape Navigator 4）的行为以防止老站点无法工作。

对于这两种模式之间的差异，最显著的例子涉及Windows上IE专有的框模型。在IE 6出现时，在标准模式中使用正确的框模型，在怪异模式中使用老式的专有框模型。为了维持对IE 5和更低版本的向后兼容性，Opera 7和更高版本也在怪异模式中使用有缺点的IE框模型。

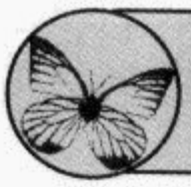
提示：关于IE错误的框模型，请参见本书［8.1 框模型（Box Model）］一节。到本书完稿之日，IE 8.0版还在测试中，相对于以前的版本，IE 8.0又增加了一种模式，以求更好地符合Web标准。

Opera 浏览器（Opera 7~8）支持与IE相同的两种呈现模式：怪异模式和标准模式，但是Opera 9的怪异模式又与之前的怪异模式不太一样，比如不再兼容IE 5.0那种盒模式。



提示： 有关详细信息，可以参阅http://www.opera.com/docs/specs/doctype/（英文）。

Mozilla Firefox和Safari支持3种呈现模式：怪异模式、几乎标准的模式（Almost Standards Mode）和标准模式。Firefox的几乎标准模式对应于IE和Opera的Standards模式。

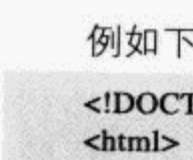


提示： 有关详细信息，请参阅http://www.mozilla.org/docs/web-developer/quirks/doctypes.html（英文）。

浏览器根据DOCTYPE是否存在以及使用的DTD来选择要使用的表现方法。如果XHTML文档包含形式完整DOCTYPE，那么它一般以标准模式表现。对于HTML 4.01文档，包含严格DTD的DOCTYPE常常导致页面以标准模式表现。包含过渡DTD和URI的DOCTYPE也导致页面以标准模式表现，但是有过渡DTD而没有URI会导致页面以怪异模式表现。关于DOCTYPE，读者可参见本书［2.2.3.1 选择DTD定义文档的类型］一节。

DOCTYPE切换是浏览器用来区分遗留文档和符合标准的文档的手段。无论是否编写了有效的CSS，如果选择了错误的DOCTYPE，那么页面就将以怪异模式表现，其表现就可能会有错误或不可预测。因此，一定要在站点的每个页面上包含形式完整的DOCTYPE声明，并且如果使用HTML，则建议选择严格的DTD，如下所示：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/ TR/html4/strict.dtd">
```



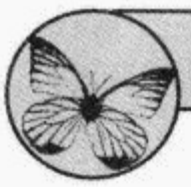
提示： 关于DOCTYPE对浏览器的影响，读者可以参阅http://meyerweb.com/eric/dom/dtype/dtype-grid.html（英文），这个表格说明了在不同DOCTYPE下浏览器将采用何种模式来显示网页。

例如下列代码，其显示如图16-1所示。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/ TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>完整的DTD :: 浏览器的模式 :: 浏览器与Hack</title>
<style type="text/css">
div{
width:200px;
border:30px #F93 solid;
padding:30px;
background: #FFC;
}
p {
background: #9CF;
}
</style>
</head>

<body>
 <div>
   <p>HTML</p>
 </div>
</body>
</html>
```

图16-1 带有完整DTD的HTML文档在Windows IE 6内的显示



提示：读者可以参见下载文件包内［/第2部分/第16章：浏览器与Hack/dtd1.html］文件。

如果去掉DOCTYPE，则其显示如图16-2所示。在图16-1中， 元素的宽度是按照CSS规范中规定的框模型显示的，而在图16-2中，由于触发了浏览器的怪异模式，因此框宽度的计算沿用了Windows IE 5.0的错误模式。因此，正确的DOCTYPE是确保CSS能被浏览器正确解释的前提。

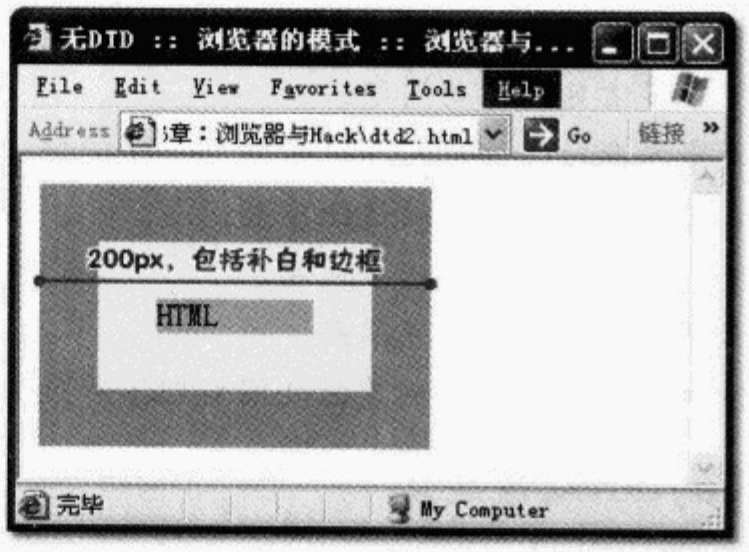
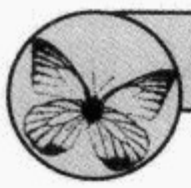


图16-2　没有DTD的HTML文档在Windows IE 6.0内的显示



提示：读者可以参见下载文件包内［/第2部分/第16章：浏览器与Hack/dtd2.html］文件。

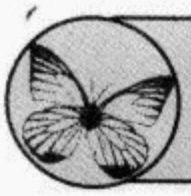
16.2 Windows IE

Windows IE是市场占有率最高的浏览器，同时也是被报告错误最多的浏览器，而它的5.0版、5.5版、6.0版和7.0版（包括还在测试中的8.0版）都有很大的不同，因此也是造成制作者最多困扰的浏览器。

16.2.1 hasLayout属性

“Layout”是一个IE的私有概念，它决定了一个元素如何显示以及约束其包含的内容、如何与其他元素交互和建立联系、如何响应和传递应用程序事件、用户事件等，这有点类似于一个窗体的概念。微软的开发者们认为框类元素（box-type elements）应该具有一个属性（property），于是他们便使用了hasLayout属性。

注意：苹果公司的操作系统Mac OS中的Mac IE和Win IE是完全不同的，它们各自拥有自己的解释引擎。Mac IE就全然不知“hasLayout”所谓何物。相比之下Mac IE的解释引擎要更标准兼容一点。因此针对hasLayout的解决方法（特别是通过使用height或width属性的）往往对Mac IE来说是有害的，所以需要对其隐藏。更多关于Mac IE的相关问题请参阅http://www.l-c-n.com/IE5tests/（英文）和http://realazy.org/blog/ 2006/07/29/working-with-buggy -browsers-1/（中文）。



提示：本小节示例，读者可以参见下载文件包内［/第2部分/第8章：框模型/ie_hasLayout.html］文件。

hasLayout并不是CSS或者(X)HTML的属性，更不是一个行为，而是IE的解释引擎的一个解释概念，在这个概念下解释的元素将具有一种特性。实际上这种特性在有些(X)HTML元素中与身俱来，而在另外一些元素中也可以通过一些CSS属性将其触发为“true（真）”，且一旦触发将不可逆转。

Layout在显示元素框时有着不同寻常而且难以预料的效果，而且有时甚至会牵连到它们的后代元素。一个元素是否具有Layout可能会引发以下一些问题。

- IE很多常见的浮动问题。
- 元素本身对一些基本属性的异常处理问题。
- 容器和其后代之间的边距重叠问题。
- 使用列表时遇到的诸多问题。
- 背景图像的定位偏差问题。
- 使用脚本时遇到的浏览器之间处理不一致的问题等。

1. hasLayout属性定义

hasLayout是IE 5.5中第一次出现，它是一个只读属性，也就是说，不可以通过任何方法来设置“hasLayout = true”或者“hasLayout = false”。

注意：在以下的内容中，如果指出一个元素“拥Layout”或“得到Layout”，或者说一个元素“has Layout”的时候，意思是指它的微软专有属性hasLayout被设为了“true”。一个“Layout元素”可以是一个默认就拥有Layout的元素或者是一个通过设置某些CSS属性得到Layout的元素。

语法	HTML：不支持 Scripting：[sHasLayout =] object.currentStyle.hasLayout
说明	IE专有属性
值	false：对象不具有Layout。 true：对象具有Layout

有些(X)HTML元素默认是具有Layout的，如：

- <html>、<body>；
- <table>、<tr>、<th>、<td>；
- <img>；
- <hr>；
- <input>、<select>、<textarea>、<button>；
- <iframe>、<embed>、<object>、<applet>；
- <marquee>。

2. 可触发Layout的CSS属性

有些CSS属性可以触发元素的hasLayout属性，由于此属性只读，因此一旦某个元素触发了“hasLayout = true”，则无法再重新设置为“false”，除非撤销触发的CSS规则。

在IE 6.0中可以触发Layout的CSS属性如下表所示。

CSS属性	值	是否符合标准
display	inline-block	是
width	任意值	是

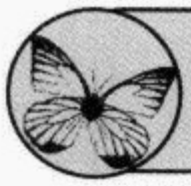
续表

height	任意值	是
float	left | right	是
position	absolute	是
writing-mode	tb-rl	否，IE私有属性
zoom	任意值	否，IE私有属性

在IE 7.0中，又增加了如下表所示的可以触发Layout的CSS属性。

CSS属性	值	是否符合标准
overflow	hidden | scroll | auto	是
overflow-x	hidden | scroll | auto	是，CSS 3属性
overflow-y	hidden | scroll | auto	是，CSS 3属性
min -width	任意值（包括0）	是
max-width	除“none”外的任意值	是
min-height	任意值（包括0）	是
max-height	除“none”外的任意值	是
position	fixed	是

对于行内元素（display属性为“inline”的元素），width和height属性只在IE 5.5以及IE 6.0、IE 7.0的怪异模式下触发hasLayout。因为如果浏览器运行于标准模式，width和height属性只能应用于块级元素，所以对行内元素设置width或height属性不能在此种情况下令该元素具有Layout。zoom属性总是可以触发hasLayout，但它是IE私有属性，不符合CSS规范，无法通过CSS校验。



提示： 微软公司关于hasLayout属性的说明，可以参见相关文档http://msdn2.microsoft.com/zh-cn/library/ms533776(en-us).aspx（英文）。

具有Layout的行内元素，其行为就和CSS的“inline-block”元素很类似了：水平方向可以并排连续排列，受 vertical-align影响，并且大小可以根据内容自适应调整。但这只是类似，并不是说IE支持“display : inline-block;”。

这也可以解释为什么单单在IE中内联元素可以包含块级元素而少出问题，因为在别的浏览器中设定了“display : inline”的元素就是行内元素，不像IE一旦行内元素拥有Layout还会变成“inline-block”类元素。

3. 重置 hasLayout

在另一条规则中重设以下属性为默认值将重置（或撤销）hasLayout（如果没有其他属性再添加hasLayout的话）。

- width、height：设为“auto”。
- max-width、max-height：设为“none”（在IE 7.0中）。
- position：设为“static”。
- float：设为“none”。
- overflow：设为“visible”（在IE 7.0中）。
- zoom：设为“normal”。
- writing-mode：从“tb-rl”设为“lr-t”。

在重置这些属性时要小心，重置可能导致渲染出现意外的状况，甚至导致IE 6.0程序不稳定。display属性与众不同，当用“inline-block”设置了“hasLayout = true”时，就算在一条独立的规则中覆盖这个属性为“block”或“inline”，hasLayout这个标志位也不会被重置为false。

把mid-width、mid-height设为它们的默认值0仍然会触发Layout，但是IE 7.0却可以接受一个不合法的属性值“auto”来重置hasLayout。

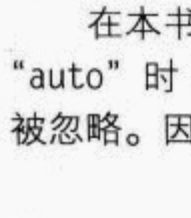
4. 脚本属性hasLayout

可以将hasLayout理解为一个只读的“脚本属性”，这样可以和CSS属性相区别。没有办法直接设置或重置一个元素的脚本属性 hasLayout。用JavaScript的hasLayout属性可以用来检测一个元素是否拥有Layout，例如上边代码：

```
<div id="test">测试</div>
```

那么只要在IE 5.5以上版本的地址栏里输入下边代码，即可检测它的状态。

```
javascript: alert(test.currentStyle.hasLayout)
```



提示： IE中的“Developer Toolbar”插件可以实时检查一个元素的当前样式，如果hasLayout是“true”，那么它的值显示为“-1”。可以通过实时修改一个元素的属性将IE私有的CSS属性zoom设置为1来触发hasLayout以便调试（但不建议在正式发布时使用，因为它通不过W3C的CSS校验）。“Developer Toolbar”插件的官方网址为http://www.microsoft.com/downloads/ details.aspx?familyid=E59C3964-672D-4511-BB3E-2D5E1DB91038&displaylang=en。

另外一个需要注意的是，Layout会影响JavaScript编程。如果一个元素没有Layout，那么clientWidth和clientHeight总是返回0。而且这和 Mozilla 浏览器的处理方式也不一样。

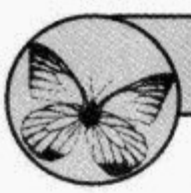
5. 具有Layout的元素的表现

具有Layout的元素有以下基本特征：

- 如果一个Layout元素中有内容，内容的排版布局将由它的边界矩形框决定；
- 拥有Layout的意思基本上就是表示某元素是一个矩形；
- 从内部来说，拥有Layout的元素将自己负责绘制其内部内容。

一般来说，在IE的DHTML引擎中，元素是不对自己的位置安排负责的。虽然每个元素（如 或者元素）都在源文档中有一个位置，在文档流也有一个位置，但是它们的内容却是由它们最近的一个Layout祖先（经常是<body>）控制安排的。这些元素依赖它们祖先的Layout来为他们处理诸如决定大小尺寸和测量信息等工作。

在实际的应用中，具有Layout的元素比较常见的问题如下。



提示： 下面所述的现象，一般都是在一定的条件才会出现。

（1）自动扩展宽度和高度。

浮动元素会被具有Layout的祖先元素自动包含。其最典型的例子就是，IE 6.0会自动扩展以包含溢出的子元素。例如有如下的代码：

在本书［8.10.3.2 height属性与auto值］一节内介绍过：常规流向中的元素，height属性为“auto”时（初始值），高度取决于该元素是否有任何块类子元素，子元素中浮动框和绝对定位框被忽略。因此，上述代码的正确显示如图16-3所示。

```
#float1 {
width:400px;
……
}
.sample1 {
width:200px;
height:60px;
float:left;
background:#FC6;
+display:inline; /* 解决 IE 6.0双倍边距Bug */
}
<div id="float1">
  <p class="sample1">浮动的p</p>
  <p>不浮动的p</p>
</div>
```

图16-3 height属性为“auto”时（初始值）元素高度不包含浮动的子元素

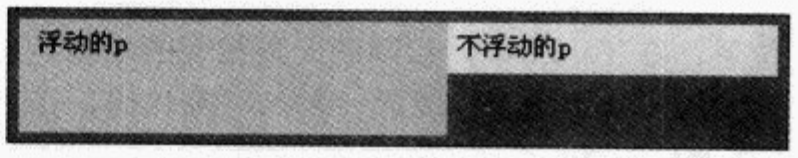
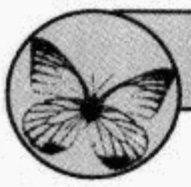


图16-4 具有Layout的元素的高度会包含浮动的子元素

但是由于 设定了width属性，这使得它具有了Layout，因此在IE 6.0/7.0中显示如图16-4所示。



提示：关于IE 6.0浮动元素双倍边距的问题，将在下面的小节内讨论。

这种情况在使用float属性对网页进行排版布局的时候经常会遇到，不过，情况往往是，制作者希望其他的浏览器能够像IE这样来自动包含浮动的子元素。但是，无论制作者觉得哪个更好，其实际情况是：IE这样的显示不符合CSS规范。

在水平方向，IE 6.0也会扩展以包含溢出的元素，但是IE 7.0却不会，例如下列代码，在6.0和I7.0内的显示如图16-5所示。

```
div { width : 400px; }
p { width : 440px; }
<div>
  <p> width : 440px </p>
</div>
```

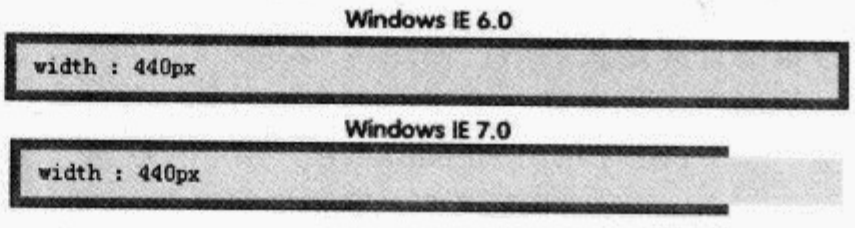


图16-5 IE 6.0和IE 7.0在水平方向的处理不相同

（2）浮动元素旁边的元素。

在本书［9.4.2 浮动元素的视觉格式化内容］一节内介绍过，浮动元素旁的块级元素还是独占1行，只是其内的行框缩短以容纳浮动元素，因此，其文字内容如果比浮动元素高的话，高出不分的文字会在浮动元素下显示。例如右列代码，其正确显示如图16-6所示。

“sample2”内的“height : 100%”实际上并没有实际效果，因为百分比的高度，基于包含块的高度，而包含块高度不定的时候，这个值无效，但是对于IE却足够触发Layout，因此在IE内显示如图16-7所示。

由图16-7可以发现，不浮动的<p>元素框整体右移，就好像它自己也是一个浮动元素一样，因此其中的文字就不再环绕左浮动元素了（而会形成一个矩形区域，保持在它的右边）。

```
.sample1 {
float : left;
……
}
.sample2 {
height : 100%;
……
}
<div id="float3">
  <p class="sample1">浮动的p</p>
  <p class="sample2">……</p>
</div>
```

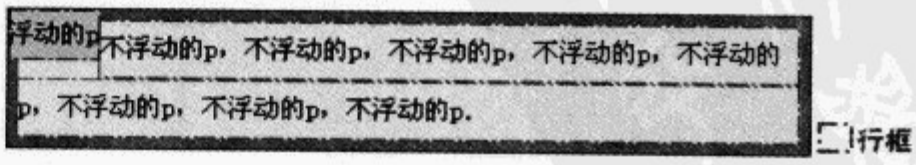


图16-6 不浮动的元素仍然会独占1行

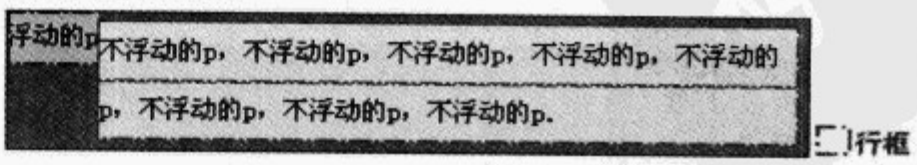


图16-7 IE内会将不浮动元素的框压缩

注意：在实际应用中，制作者也许正需要这种效果，这样可以完成左右2列的布局，而且，不指定宽度，右边的列可以自适应父元素的宽度，但是在其他浏览器内的兼容问题就会出现。更多的例子可以参见http://dev.l-c-n.com/IEW2-bugs/float-layout.php（英文）和http://dev.l-c-n.com/IEW2-bugs/float-adjecant.php（英文）。

同样会受到影响的，还有相对定位的元素。相对定位元素，它的偏移量应该参照的是其静态位置，在IE 6.0中，和浮动元素相邻的相对定位元素，由于静态位置的改变，造成偏移是从浮动元素的右边距边开始算起，例如下列代码，其正确显示如图16-8所示。

```
.sample1 {
float : left;
……
}
.sample3 {
position : relative;
left : 20px;
}
<div id="clear1">
  <p class="sample1">浮动的p</p>
  <p class="sample2">不浮动的p，<strong>清除浮动的子元素</strong></p>
</div>
```

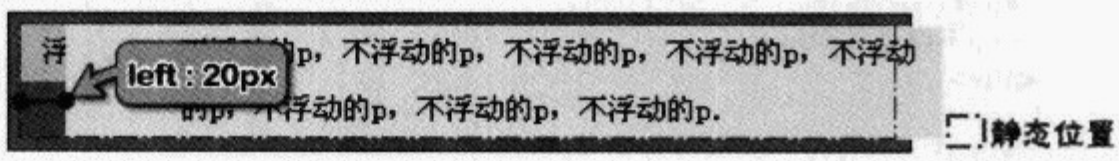


图16-8 浮动元素旁相对定位元素的偏移量仍以静态位置为参考

但是在IE 6.0和IE 7.0的内显示如图16-9所示。

由图16-9可以发现，在IE 7.0内，虽然偏移量的计算是正确的，但是IE 6.0和IE 7.0同时都出现了元素背景绘制不完全的现象，这是由于“position: relative”并不触发hasLayout，所以很多诸如内容消失或错位的渲染错误就会因此而起。这些现象可能会在刷新页面、调整窗口大小、滚动页面、选中内容等情况下出现。

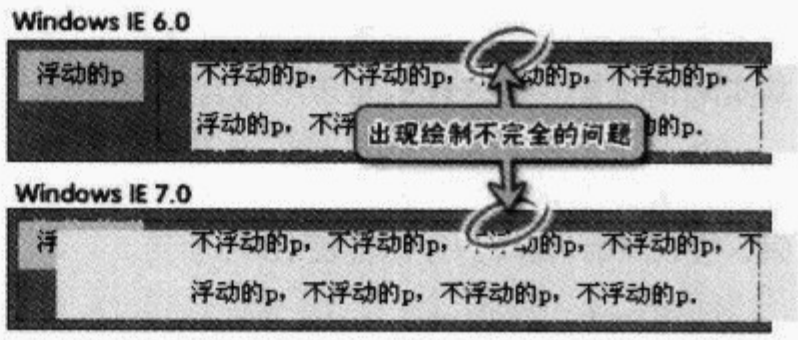


图16-9 浮动元素旁相对定位元素在IE 6和IE 7中的显示。

而如果为不浮动的<p>元素添加一个可以触发hasLayout的CSS规则：

```
.sample4 { height : 100%; }
```

此时IE 6.0和IE 7.0的效果将统一，如图16-10所示。可以发现，触发hasLayout属性可以解决背景问题，但是也使得定位的位置发生错误。

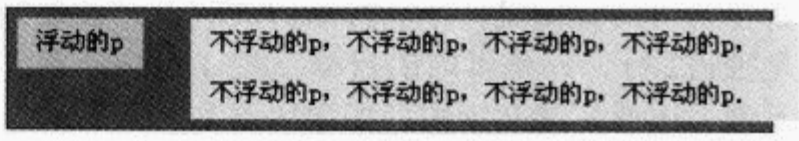


图16-10 触发hasLayout属性后IE内的显示

（3）未定义宽度的浮动元素。

如果一个浮动元素没有设定宽度（即“width : auto”），那么这个元素宽度会被压缩，宽度由其包含的内容来确定。但是，对于IE，如果子元素具有Layout，则元素会被撑到可以容纳它的最大宽度。例如下列代码，其正确的显示如图16-11所示。

```
.sample1 {
float : left;
……
}
.sample1 strong{
display : block;
height : 100%;
}
<div>
  <p class="sample1">浮动的p</p>
```

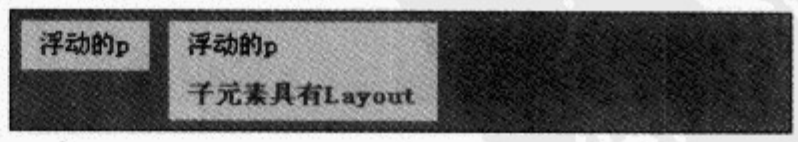


图16-11 未设定宽度的浮动元素宽度被压缩

```
  <p class="sample1">浮动的p<strong>子元素具有Layout</strong></p>
</div>
```

而在IE 6.0内显示如图16-12所示。由图16-12可以发现，第2个浮动的<p>元素占据了 内剩余的宽度。

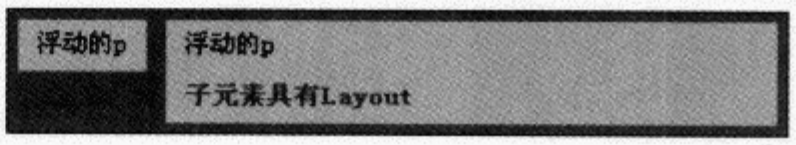


图16-12　IE 6.0内具有Layout的子元素会将浮动元素撑开

（4）清除浮动。

在本书［9.4.3 清除浮动：clear属性］一节内介绍过，设置“clear:left（right）”属性的元素的左边（右边）允许有左（右）浮动框。但是在IE中，clear属性无法影响其Layout包含容器之外的浮动元素。例如下列代码，其正确的显示如图16-13所示。

```
.sample1 {
float : left;
……
}
.sample2 strong {
clear:left;
display:block;
}
<div>
  <p class="sample1">浮动的p</p>
  <p class="sample2"><strong>子元素1</strong><strong>子元素2</strong></p>
</div>
```

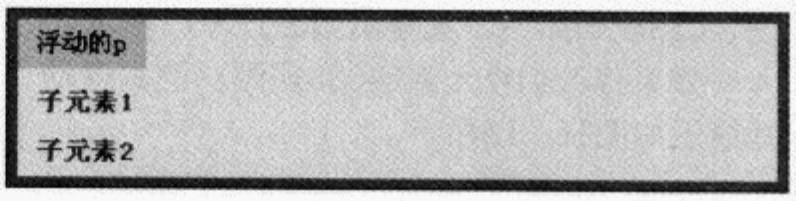


图16-13　清除浮动的正确效果

此时“sample2”没有Layout，因此其在IE内的显示如图16-14所示。

而如果对“sample2”增加如下CSS触发其hasLayout属性，则显示如图16-15所示。

```
.sample2 { height : 100%; }
```

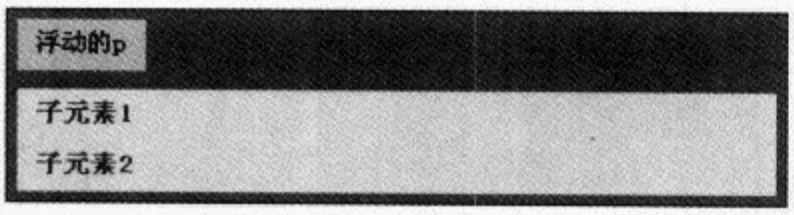
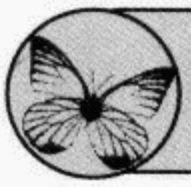


图16-14　IE内清除浮动的效果

图16-15　IE内具有Layout的元素内部子元素清除浮动的效果

由图16-15可以发现，元素无法清除左边的浮动元素，因为被限制在具有Layout的父元素中。这点在使用float和clear属性进行页面排版布局的时候需要特别注意，因为在不同浏览器内会发生显示不一致的情况。



提示：关于清除浮动更多的内容，可以参阅http://www.satzansatz.de/cssd/ acidicfloat.html（英文）。

（5）列表。

无论是<ol>、<ul>还是<li>元素，拥有layout后都会影响列表的表现。不同版本IE的表现又有不同，最明显的效果就体现在列表标记上。具体内容读者可参见本书［12.1.5 浏览器对列表的表现与样式的继承］一节。

由于系统列表标记是通过“内部机制”添加的，因此无法去控制它们。例如下列代码，在正常情况下，其显示如图16-16所示。

```
ol {
margin : 5px;
border : 2px solid #FC3;
```

```
}
li {
background : #CFF;
margin : 5px 5px 5px 30px;
}
<div>
 <ol>
  <li><a href="#">列表项1</a></li>
  <li><a href="#">列表项2</a></li>
  <li><a href="#">列表项3</a></li>
 </ol>
</div>
```

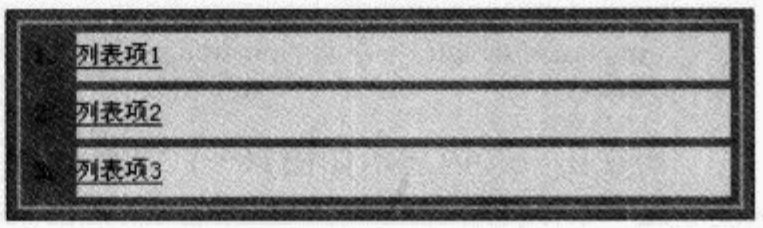


图16-16 正常的列表显示

如果增加CSS规则以触发<li>的hasLayout属性，CSS规则如下，其显示如图16-17所示。

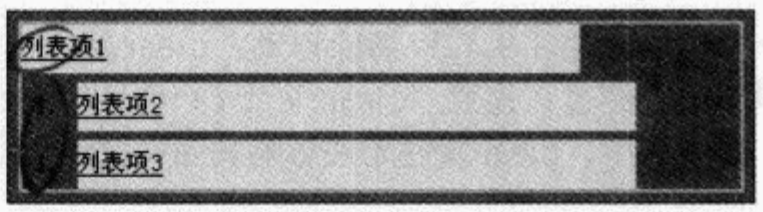


图16-17 触发了hasLayout属性后<li>的错误表现

如果修改<li>元素的CSS规则，使其浮动，则其显示如图16-18所示。

```
li {
width:300px;
float:left;
}
```

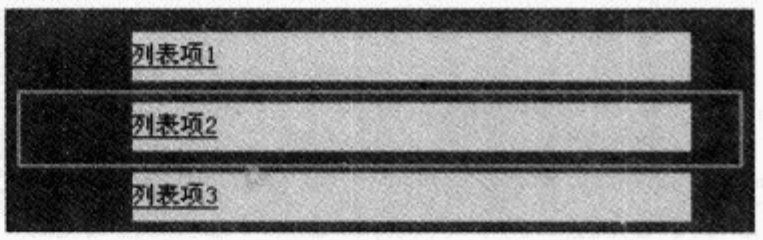


图16-18 浮动<li>元素后IE的显示

由图16-18可以发现，<li>的列表标记消失了，同时<ol>的高度显示错误。

以上某些问题还是无法解决的，所以如果需要列表符号的时候最好避免让列表和列表元素获得Layout。如果需要限定尺寸，最好给其他元素设定尺寸，比如给整个列表外面套一个元素并设定它的宽度，又或者给每个列表元素中的内容设定高度等。

（6）表格。

表格总是具有Layout，它总表现为一个已定义宽度的对象。在IE 6.0中，“table-layout: fixed”的表现通常和一个宽度设100%的表格相同。例如下列代码，其正确显示如图16-19所示。

```
#table1 {
……
}
#table1 table {
……
table-layout : fixed;
}
#table1 td {
background : #FC6;
}
<div id="table1">
<table summary="样例表格，table-layout : fixed，没有设定宽度的表格。">
 <caption>
 固定布局
 </caption>
 <tr>
  <td>1-1</td>
  <td>1-2</td>
  <td>1-3</td>
 </tr>
 <tr>
  <td>2-1</td>
  <td>2-2</td>
  <td>2-3</td>
 </tr>
</table>
</div>
```

图16-19 固定布局的表格宽度由第1行内的单元格决定

而在IE 6.0/7.0内的显示如图16-20所示。

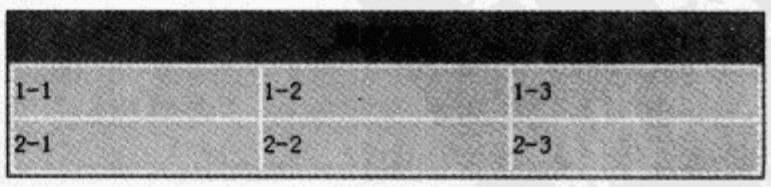


图16-20 IE内固定布局表格宽度等于父元素的内容宽度

（7）绝对定位。

在本书［9.3.4 绝对定位］一节内介绍过，IE 6.0内绝对定位计算上的错误，造成这个错误的原因就是该元素没有Layout，因此设定height属性或者其它可以触发Layout的CSS属性，可以纠正这个错误。

在IE中绝对定位只有当其包含元素拥有Layout时才会计算正确，而且绝对定位元素的百分比宽度参考也存在问题。尽可能地让绝对元素的包含块拥有Layout，而且尽量让其就是绝对定位元素的父元素，可以最大限度地避免定位错误的发生。

（8）背景定位。

在本书［10.3.3 背景图片定位：background-position］一节内介绍过，背景图片的定位，以元素补白边为起点，而在 IE/Win 下，如果 "hasLayout = false" 则指的是 "边框边"，当 "hasLayout=true" 时指的才是补白边。

（9）边距重叠。

hasLayout会影响一个盒子和其后代元素的边距重叠。在本书［8.9.2.2 边距的重叠］一节内介绍过：相邻元素垂直方向的边距重叠，包括元素（当元素没有边框和补白的时候）和其后代元素之间。例如右边代码，其正确显示如图16-21所示。

```
#margin1 p {
margin : 10px;
height : 60px;
padding : 0;
}
#margin1 p strong {
display : block;
background : #FC6;
margin : 30px;
}
<div>
  <p><strong>p内的子元素。</strong>
</p>
</div>
```

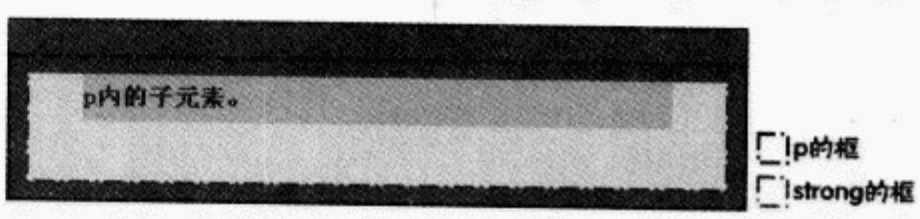


图16-21 元素垂直边距的重叠

但是由于height属性使元素具有了Layout，因此在IE内的显示如图16-22所示。可以发现，IE 6.0和IE 7.0对于边距重叠的表现也不相同。因此，在实际应用中，应尽量避免给触发了hasLayout属性的子元素设置垂直边距。

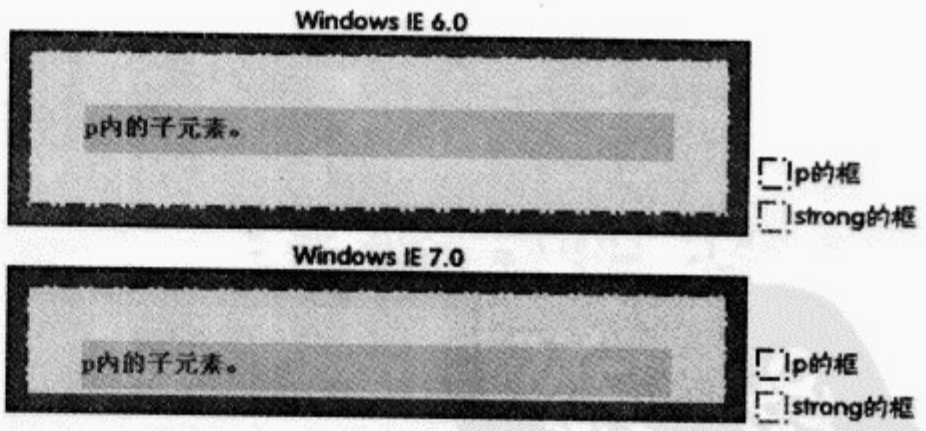


图16-22 hasLayout属性对于垂直边距压缩的影响

（10）对已渲染元素的重排（re-flow）。

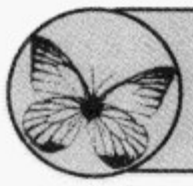
当所有元素都已渲染完成时，如果有一个因鼠标经过而引起的变化产生(比如某个链接的 background 有变化)，IE会对其 layout 包含区块进行重排。有时一些元素就会因此被排到了新的位置，因为当这个鼠标经过发生时，IE已经知道了所有相关元素的宽度、偏移量等数据了。这在文档首次载入时则不会发生，那时由于自动扩张的特性，宽度还无法确定。这种情况会导致在鼠标经过时页面出现跳变。

（11）边缘裁切。

当一个元素框包含了诸如伸出其边缘的内容这种更复杂的结构时，这个容器就经常需要hasLayout属性来避免一些渲染错误。但使用这种常用方法又会在边界处理时左右为难，因为一个获得Layout的元素会变成某种自封闭的矩形框。

例如，在本书［8.9.4 负值边距］一节中的图8-38中，由于外层 设定了width属性，因此其内设定了负边距的<p>元素向外移动溢出部分被裁切。

以上介绍的，只是hasLayout属性比较常见的现象和问题，有时hasLayout属性可以帮助解决某些问题，但是有时又可能引发其他的问题，因此需要根据实际情况来决定采取何种方式解决问题。



提示：有很多IE中的Bug，需要触发hasLayout属性来解决，但是也有很多问题也是由于hasLayout属性引起，在实际应用中需要灵活运用。

16.2.2 条件注释

在(X)HTML内添加注释可以方便阅读和分析代码，在注释标签内的内容不会被浏览器显示。注释的语法如右：

```
<!--注释内容-->
```

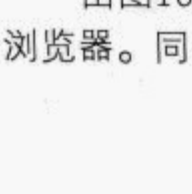
而从IE 5.0开始，可以使用一种“条件注释”，IE会据注释中的条件判断是否解释注释中的内容，其语法如右：

```
<!--[if 条件]>
这里是正常的(X)HTML代码
<![endif]-->
```

（1）条件注释的基本结构和(X)HTML的注释（<!-- -->）是一样的，因此IE以外的浏览器将会把它们看作是普通的注释而完全忽略它们。

（2）IE将会根据“if 条件”来判断是否如解析普通的页面内容一样解析条件注释里的内容。

（3）条件注释使用的是(X)HTML的注释标签，因此它们只能使用在(X)HTML文件里，而不能在CSS文件中使用。



提示：本小节示例，读者可以参见下载文件包内［/第2部分/第8章：框模型/ie_if.html］文件。

可使用如下代码检测当前IE浏览器的版本，其在浏览器内显示如图16-23所示。

```
<body>
 <!--[if IE]>
  <h1>您正在使用IE浏览器</h1>
  <!--[if IE 5]>
   <h2>版本 5</h2>
  <![endif]-->
  <!--[if IE 5.0]>
   <h2>版本 5.0</h2>
  <![endif]-->
  <!--[if IE 5.5]>
   <h2>版本 5.5</h2>
  <![endif]-->
  <!--[if IE 6]>
   <h2>版本 6</h2>
  <![endif]-->
  <!--[if IE 7]>
   <h2>版本 7</h2>
  <![endif]-->
 <![endif]-->
 <!--[if !IE]><!-->
  <h1>您使用不是 Internet Explorer</h1>
 <!--<![endif]-->
</body>
```

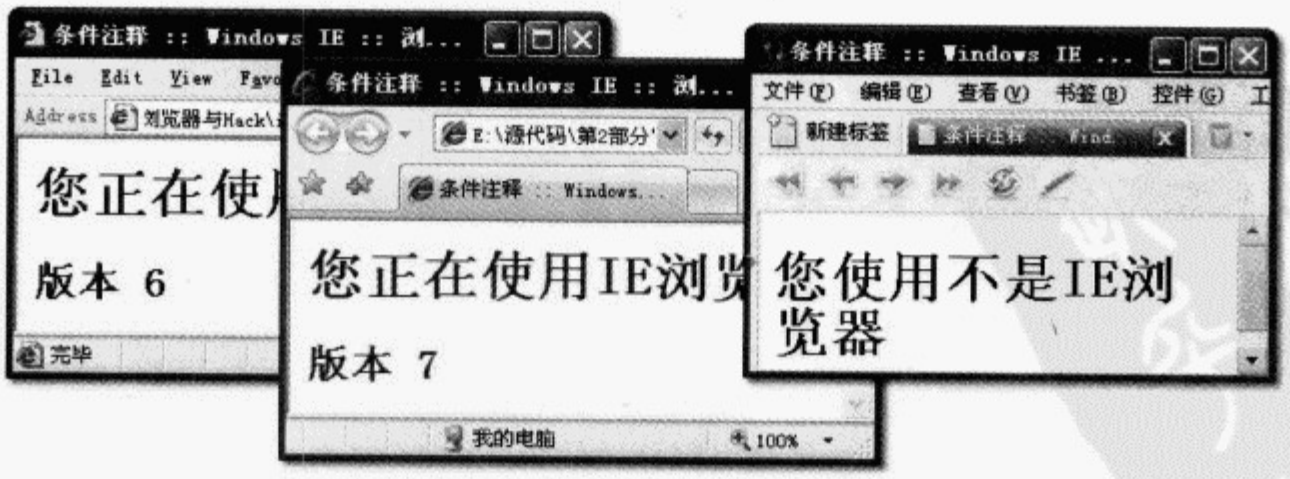


图16-23 条件注释在不同浏览器内的显示

由图16-23可以发现，在IE内根据不同的版本显示了不同的文字，而在Opera内，显示不是IE浏览器。同时条件注释还可以添加以下逻辑判断的参数。

- lte：Less than or equal to，小于或等于。
- lt ：Less than，小于。
- gte：Greater than or equal to，大于或等于。
- gt ：Greater than，大于。
- ！：不等于，非。

例如右边代码：

```
<!--[if gt IE 5.5]>  （如果IE版本大于5.5）
<!--[if lte IE 6]>   （如果IE版本小于等于6）
<!--[if !IE]>        （如果浏览器不是IE）
```

条件注释可以在(X)HTML文档内使用，因此可以在<head>区内，针对不同版本的IE来调用不同的CSS文件，例如以下代码：

```
<!-- 默认先调用basic.css样式表 -->
<link rel="stylesheet" type="text/css" href="basic.css" />
<!--[if lte IE 5.5]>
<!-- 如果IE浏览器版本小于或等于5.5，调用ie5.css样式表 -->
<link rel="stylesheet" type="text/css" href="ie5.css" />
<!--[if IE 6]>
<!-- 如果IE浏览器版本为6，调用ie6.css样式表 -->
<link rel="stylesheet" type="text/css" href="ie6.css" />
<![endif]-->
```

也可以如下使用：

```
<!--[if lte IE 5.5]>
<!-- 如果IE浏览器版本小于或等于5.5-->
<style type="text/css" >
 /* 针对IE 5.5及以前版本的CSS */
</style>
<![endif]-->
```

条件注释，对于非IE浏览器，等同于注释，因此可以通过W3C的校验。

16.3 常用的CSS Hack

由于不同的浏览器对CSS的“理解”可能不一样，因此会导致生成的网页的显示效果不一样，更严重的，可能由于浏览器本身的错误而出现网页显示不完全或者错位等问题。

这个时候就需要针对不同的浏览器去写不同的CSS，以使网页在不同的浏览器内的效果基本一致。由于网页的重点在于“易用”和“友好”，因此不必追求在不同浏览器内显示完全一致。

针对不同的浏览器写不同的CSS规则，这就叫CSS Hack。其实这更像一种过滤手法，利用一些方法，过滤掉某些浏览器。

16.3.1 CSS Hack原理

CSS本身具有很强的向前兼容性，如果浏览器不理解某个选择器，那么它会忽略整个规则，同样，如果它不理解某个属性或值，也会忽略整个声明，这个特性意味着添加新的选择器、属性和值，一般不会对老式浏览器产生严重的影响。

因此，可以利用这一特性，针对比较高级的浏览器应用高级的规则和声明，老式浏览器会忽略这些规则。同时，利用浏览器对于特殊字符的识别能力和CSS的层叠特性，也可以达到针对浏览器屏蔽某些规则的目的。

有时候也可以利用浏览器的特性（例如IE专有的hasLayout属性）来触发浏览器的特殊模式，以达到效果。但是这些方法并不是万全之策，因为在新版本的浏览器内，可能会做相应的修改。

16.3.2 CSS Hack不是必须的

会使用CSS Hack并不是高水平的表现。当某些效果在某个浏览器内不起作用，或者几个浏览器内效果不统一（特别是其他浏览器与IE浏览器不统一）时，新手们往往会去求助于CSS Hack，而从前面章节的介绍，读者应该已经了解，应用最广泛的IE却恰恰是最不“标准”的浏览器。

并非所有的CSS问题都是浏览器的错，很多情况下，往往是制作者对于CSS规范理解的不够，或者代码中的错误引起的。同时，有些错误也许是在允许范围之内的，例如IE中3px的偏差，对于内容的显示也许没有什么影响，因此也不必为此添加CSS Hack来纠正。

在实际应用中，针对不同元素设定CSS，可以完成一样的效果，因此有很多情况可以通过修改CSS的方案来避免使用CSS Hack。

16.3.3 常用的CSS Hack

如果只有少数的地方使用了CSS Hack，可以直接写在样式表中，而如果有比较多而且复杂的CSS Hack，则最好写在单独的CSS文件中，然后链接到页面内。

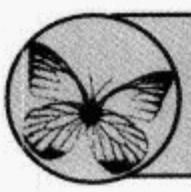
例如在本书［3.2.4 导入式样式表］一节内介绍的使用@import规则导入样式表文件，对于老式的不支持@import规则的浏览器会忽略样式表。读者也可参见下载文件包内［/第2部分/第16章：浏览器与Hack/ css_hack.html］文件。

1. IE条件注释

在本章［16.2.2 条件注释］一节内介绍的IE专有的条件注释是相对安全的过滤方法。

2. 子选择器

最安全的过滤方法依赖于未实现的CSS而不是浏览器的Bug，因为它们是使用有效的CSS选择器来应用有效的声明，所以严格地说，它们不属于Hack手段。



提示：Bug指程序缺陷、臭虫，电脑系统或者程序中存在的任何一种破坏正常运转能力的问题或者缺陷，都可以叫作“Bug”。

由于IE 6.0和更早的版本不支持子选择器，因此可以使用它对这些浏览器隐藏规则。为了让这种过滤方法起作用，必须确保在子选择器前后没有空格。关于子选择器，请参见本书［4.2.6 子元素选择器（Child Selectors）］一节。

例如下列代码：

```
#hack1 {
border : 1px solid #36C;
line-height : 50px;
background : #9CF url(../../img/bg1.gif) repeat-x;
}
body>#hack1 {
background-image : url(../../img/bg1.png);
}
<body>
<div id="hack1">
IE 6及以前的版本不支持透明的PNG图片
</div>
</body>
```

由于IE 6.0及更早版本不支持透明的PNG图片，因此首先对“hack1”层设定了“bg1.gif”图片为背景图片，而对于支持子选择器的浏览器将使用“body>#hack1”（特殊性高于“#hack1”）内定义的“bg1.png”图片为背景，其在浏览器内显示如图16-24所示。

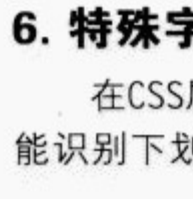
图16-24 使用子选择器过滤浏览器

此时需注意的是，要保证“body>#hack1”的特殊性高于前面的选择器，请参见本书［4.6层叠］一节。

3. 属性选择器

和子选择器类似，IE 6.0及更早的版本也不支持属性选择器，因此可以使用属性选择器在比较高级的浏览器内对类和ID应用样式。例如右边代码：

```
div[id="nav"] {
background-image: url(../../img/bg1.png);
}
```



提示：关于属性选择器，请参见本书［4.2.8 属性选择器（Attribute Selectors）］一节。

4. “* html”

在本书［4.2.1 通配选择器（Universal Selector）］内介绍过，星号“*”为通配符，表示所有的元素，而<html>元素为根元素。

但是IE 6.0及更早的版本有一个匿名的根元素，它包围着<html>元素，因此可以使用通配选择器制定包围在<html>元素外的这个匿名元素，从而只针对IE 6.0及更早的版本设定CSS规则，其他的浏览器将忽视这些规则。例如：

```
* html { font-size : x-small; }
* html div{ color : red; }
```

5. !important

这个仍旧是针对IE 6.0及更早的浏览器的Hack，因为它们不支持“!important”。关于“!important”，请参见本书［4.6.4 重要性］一节。

例如下列代码：

```
#hack2 {
……
background:#9CF url(../../img/bg1.png) repeat-x !important;
background-image:url(../../img/bg1.gif);
}
<div id="hack1">
IE 6及以前的版本不支持透明的PNG图片
</div>
```

对于IE 7.0及Firefox、Opera等浏览器，将使用“!important”声明的“bg1.png”图片，而由于IE 6.0及更早版本不识别“!important”，因此根据层叠的规定，使用“bg1.gif”图片替代前面定义的“bg1.png”图片。因此，特别需要注意2条规则的书写顺序，“!important”声明在前面，而希望IE 6.0接受的规则放在后面。

6. 特殊字符

在CSS属性的前面或者中间添加特殊的字符也可以达到过滤某些浏览器的目的，例如，IE 6.0能识别下划线“_”和星号“*”，IE7.0能识别星号“*”，但不能识别下划线“_”，而Firefox、

Opera和Safari则都不能认识。因此可以利用这些特殊字符来针对不同的浏览器设定不同的CSS规则，例如下列代码：

```
#hack3 {
background : #9CF;
*background : #CFC;
_background : #FC6;
}
<div id="hack3">
 <p>background:#9CF -- Firefox、Opera和Safari都不识别。</p>
 <p>*background:#CFC -- IE 7识别"*"不识别"_"</p>
 <p>_background: #FC6 -- IE6 识别"_"和"*"</p>
</div>
```

由于Firefox、Opera和Safari不能识别“*”和“_”开头的声明，因此会忽略，只接受第1条声明；IE 7.0不能识别“_”开头的声明，但是可以识别“*”开头的声明，因此依据层叠的规定，采用第2条声明；而IE 6.0可以识别“_”开头的声明，因此接受第3条声明，在这几个浏览器内的显示如图16-25所示。

图16-25 不同的浏览器对特殊字符的识别不同

如果只想针对IE设定某些声明，则可以只使用“*”，例如：

```
div {
……
*width : 100%;
}
```

如果只想针对IE 6.0设定某些声明，则可以只使用“_”，例如：

```
div {
……
_height : 100%; /* 可以用来触发hasLayout属性 */
}
```

7. CSS注释/**/

如果只想针对IE 6.0隐藏某些声明，可以使用空格加CSS注释的方法来完成，例如：

```
#hack4 {
……
background /**/ : #FC6; /* 在注释和属性中间有空格 */
}
<div id="hack4">只有IE 6没有背景颜色。</div>
```

在属性和注释/**/之间有一个空格，只有IE 6.0无法识别此条声明，IE 7.0乃至IE 5.5都可以识别，因此其显示如图16-26所示。

图16-26 IE 6.0不能识别空格加注释

8. 针对Mac IE 5隐藏

由于Mac版的IE和Windows版的IE使用不同的解释引擎，因此很多Windows IE内会出现的Bug在Mac IE中不会出现，因此很多修复Bug的Hack要针对Mac IE隐藏。

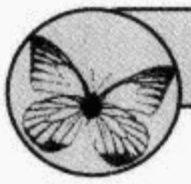
可以利用注释反斜线的Hack来对Mac IE 5隐藏CSS声明。例如以下右边代码：

在前面的章节曾经介绍过，“\”表示转义，因此Mac IE 5错误地将注释内容中的“\”也当作转义符号来看待，因此“*/”被转义，Mac IE 5会认为注释没有结束，因此其后的内容都被当作了注释，直到遇到下一个正常的“*/”。

```
/* 对Mac IE 5隐藏 \*/
div { height:1%;}
/* 结束隐藏 */
```

因此如果只想对Windows IE 6.0及更早版本应用规则，可以如右设定：

```
/* 对Mac IE 5隐藏 \*/
* html div { height:1%;}
/* 结束隐藏 */
```



提示：更多的过滤方法，可以参见http://centricle.com/ref/css/filters/（英文）。

16.4 发现与解决问题

虽然浏览器可能存在着很多Bug，但是并不是与自己感觉不一样的时候就是出现了Bug。产生问题的原因可能有很多种，因此制作者需要一定的方法来排查问题。

16.4.1 排查问题

选择一个先进的浏览器进行测试是明智的做法，例如对CSS 2.1支持比较好的Opera 9.2、Firefox 2.0或者Safari 3.0，而不要使用IE 6.0作为唯一的测试浏览器，因为IE 6.0的问题是最多的，这非常不利于制作者（特别是初学者）理解CSS 2.1的规范与判断浏览器的表现是否正确。如果在先进的浏览器内测试是正确的，而在IE内有问题，那基本上可以判定是IE的问题。如果在先进的浏览器内的显示不正确，那么可以先进行如下初步判断。

1. 拼写是否正确

可以使用W3C的校验，或者网页编辑软件的校验功能，来检查(X)HTML文档内的标签是否配套、嵌套顺序是否正确、空标签是否闭合，CSS拼写是否正确。不正确的嵌套、错误的拼写是很常见的错误。

提示：现在有很多编辑软件都可以提供(X)HTML和CSS的校验功能，包括浏览器对CSS属性是否支持等，如Dreamweaver 8以上版本、TopStyle等软件。Firefox中的附加软件“Firebug”是一个非常好用的工具，它不仅可以检查(X)HTML、CSS和JavaScript是否正确，还可以动态显示页面内元素的框和位置，是调试网页很好的辅助插件。读者可以访问它的官方网站下载http://www.getfirebug.com/（英文）。

2. 是否有合适的DTD

在本书的其他章节里，曾经不止一次地强调过DOCTYPE的重要性，不同的DOCTYPE直接影响浏览器对于(X)HTML和CSS的解释。

3. CSS属性浏览器是否支持

虽然现代浏览器支持绝大部分的CSS 2.1规范和部分的CSS 3规范，但是在前面的章节也介绍过，有一些CSS属性还没有被浏览器广泛支持，因此在某个属性没有生效的时候，请确定浏览器是否支持。

4. 隔离问题

将有问题的地方突出出来，例如给元素加一个醒目的边框或者背景颜色。如果增加了边框就可以解决问题，那么就是边距重叠的问题。如果增加了背景，但是背景不显示，那么有可能是特殊性或者浮动元素没有闭合。尝试修改一些属性，特别是会触发IE的hasLayout的属性，判断是否是IE常见的Bug，读者可以参见［16.4.3 Windows IE常见Bug］一节。

5. 建立基本测试

如果还不能解决问题，则可以复制问题文件，然后删除多余的(X)HTML，只留下有问题的部分。删除(X)HTML内的注释问题，看问题是否会消失。删除元素间的空格，看问题是否会消失。然后分块注释掉样式表，直到问题消失，则刚注释掉的样式即为问题所在。

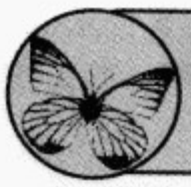
6. 解决问题而不是解决现象

找到问题根源的所在并解决它是最终目，而不是为了迁就表现而使用复杂的Hack来让网页“看上去很美”。不从根源上解决问题，当浏览器升级以后，可能会遇到更多的问题。同时，Hack的时候可能会造成新的问题的出现，特别是触发或者避免触发IE的hasLayout属性。

换一种思路也许也可以避免问题的出现，例如将元素的margin属性取消，改为设置其父元素的padding属性。只有实在无法解决的时候，再使用Hack。

16.4.2 常见的非Bug问题

有一些表现上的不一致并不是浏览器的错误，而是制作者对CSS理解不透彻而造成的。以下是比较常见的容易引起误解的地方。



提示： 读者可参见下载文件包内［/第2部分/第16章：浏览器与Hack/ css_inspect.html］文件。

1. 层叠与特殊性

CSS的层叠与特殊性也是经常困扰初学者的问题，当定义了某个属性而没有起作用的时候，请检查是否在其他地方也定义了该属性，并且比较看哪个定义具有更高特殊性。

例如有以下左边的XHTML代码，如果定义为右上代码，则“<li class="first">”并不会显示蓝色，而仍是红色，这是因为“#nav li”的特殊性高于“.first”，因此需要如右下定义，才会使“.first”生效：

```
<div id="nav">
 <ul>
  <li class="first">列表项1</li>
  <li>列表项2</li>
 </ul>
</div>
```

```
#nav li { color : red; }
.first { color : blue; }
```

```
#nav .first { color : blue; }
```

2. 相邻垂直边距重叠

相邻元素之间垂直边距的压缩是经常会造成困扰的问题之一，特别是当IE内元素被触发了hasLayout属性而垂直边距不重叠时，很容易让初学者认为只有IE是正确的。

例如下列代码：

```
#hack5 {
background : #9CF;
margin : 5px;
width : 400px;
}
#hack5 p {
background : #FFC;
margin : 10px;
}
<div id="hack5">
 <p>垂直边距的重叠</p>
</div>
```

图16-27　在IE内触发了hasLayout属性的元素边距不重叠

由于“hack5”设定了width属性，因此触发了IE的hasLayout属性，因此在IE内的显示如图16-27所示，这往往是制作者希望的效果。但是，按照CSS规范，其显示应该如图16-28所示。

可以通过为“hack5”增加1px的补白或者边框来解决这个问题，如：

```
#hack5 {
background : #9CF;
margin : 5px;
padding : 1px;
width : 400px;
}
#hack5 p {
background : #FFC;
margin : 9px;
}
```

图16-28　CSS规范中垂直边距的重叠

3. 高度不适应（闭合浮动元素）

当使用float属性进行布局的时候，往往会遇到父元素无法包含浮动的子元素的问题，最严重的现象是，当元素的子元素全部浮动的时候，本元素将没有高度，例如下列代码：

```
#wrap {
border:1px solid #F60;
width:500px;
}
#header {
background:#9CF;
}
#content {
width:370px;
background:#FFC;
float:left;
}
#menu {
width:130px;
background:#CFC;
float:right;
}
#footer {
background:#9CF;
}
<div id="wrap">
 <div id="header">header</div>
 <div id="content">content</div>
 <div id="menu">menu</div>
 <div id="footer">footer</div>
</div>
```

图16-29　浮动造成的高度问题

这是1个常见的3行2列的布局样式，但是，由于中间的2列浮动，因此其显示如图16-29所示。

此时需要对 预元素“footer”进行清除浮动才可以使其正常显示。其CSS规则如右，显示如图16-30所示。

```
#footer {
……
clear:both;
}
```

但是有的时候，在浮动元素后没有可以用来清除浮动的元素，例如下列代码：

```
ul {
background : #FF9;
border : 1px solid #F60;
width : 300px;
padding : 2px;
}
#inspect3 li {
float:left;
……
}
<div>
 <ul>
  <li>列表项1</li>
  <li>列表项2</li>
  <li>列表项3</li>
 </ul>
</div>
```

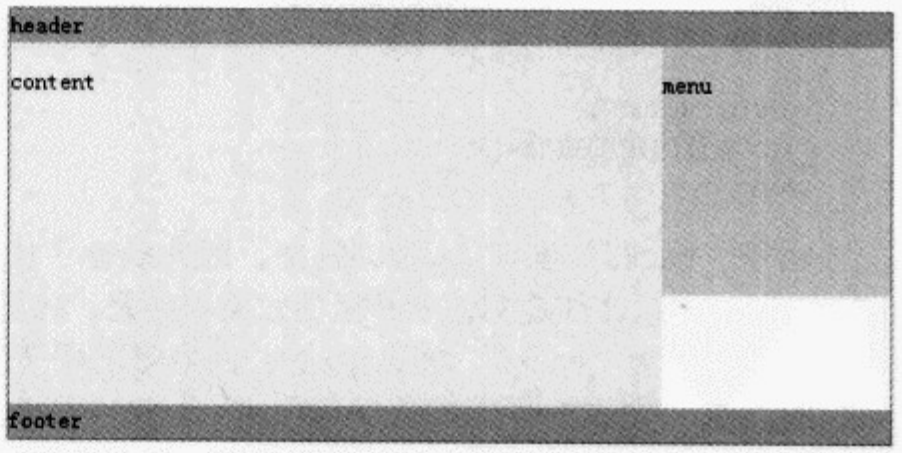


图16-30 清除浮动后高度问题解决

对于标准浏览器，由于<li>全部浮动，因此<ul>将失去高度，其显示如图16-31所示。而在IE内，由于width属性触发了<ul>元素的hasLayout，因此其显示如图16-32所示。

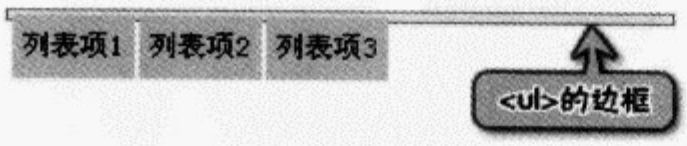


图16-31 子元素全部浮动导致元素失去高度

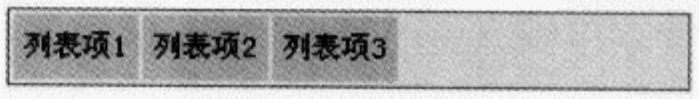


图16-32 IE内的显示

由于<ul>内没有可以设定清除浮动的元素，因此需要利用ul:after伪元素来在<ul>最后生成一个元素，然后利用这个生成的元素来清除浮动。其CSS规则如下，显示如图16-33所示。

```
ul:after {
content: ".";  /* :after必须和content属性一起使用，具体内容无关紧要 */
display: block;  /* 块级元素才可以清除浮动 */
height: 0; /* 高度为0，以隐藏生成的内容 */
clear: both; /* 清除浮动 */
visibility: hidden; /* 隐藏生成的内容 */
}
```

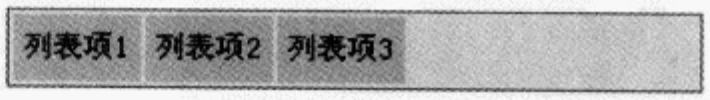
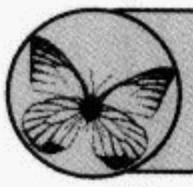


图16-33 清除浮动后的<ul>元素高度包含子元素



提示：这些CSS声明是固定的，读者可以将其复制到需要的地方，而不需要修改里面的内容。

有时候，制作者也许不确定是否触发了IE的hasLayout属性，因此还有一套通用的闭合浮动元素的Hack：

```
/* Clear Fix */
.clearfix:after {
content : ".";
display : block;
height : 0;
clear : both;
visibility : hidden;
}
.clearfix {
display: inline-block; /* 触发hasLayout */
}
/* Hides from IE-mac \*/
* html .clearfix {
height: 1%; /* 触发IE 5-6 的hasLayout */
}
/* End hide from IE-mac */
.clearfix {
display: block; /* 恢复display属性 */
}
/* end of clearfix */
```

只要为需要闭合的元素添加class即可解决问题，例如本例中，应该为<ul>元素添加class属性如右：

```
<ul class="clearfix">
```

如果上面的方法都不可行，那么也可以通过添加1个额外的空元素来清除浮动，例如右边代码：

```
.clearFloat { clear : both; }
<div id="wrap">
  <div id="header">header</div>
  <div id="content">content</div>
  <div id="menu">menu</div>
  <div class="clearFloat"></div>
</div>
```

空元素没有高度，因此不会被显示，但是需要注意此元素不能有padding、margin和border等属性，因此一般使用 ，因为没有浏览器默认样式。这个方法虽然可以有效地清除浮动，闭合元素，但是它不是网页内容的一部分，而且，如果设计方案改变，此处不需要清除浮动时，还需要删除此代码，因此灵活性较差。清除浮动的方法需要根据实际情况来决定。

4. 尺寸计算是否正确

尺寸（特别是宽度）的计算也经常是造成错误显示的原因，例如忽略了元素框的宽度为widht、padding、border和margin属性的和，因此若干并列浮动显示的子元素宽度的总和大于父元素的宽度而造成回行。

IE对于百分比宽度的计算存在误差，有时候1行内的几个使用百分比宽度的元素的总宽度等于100%，在IE内实际计算值会大于100%，从而造成回行或者溢出。例如下边代码：

```
#width {
border:1px solid #F90;
background:#FFC;
}
#width li {
list-style:none;
float:left;
width:25%;
background:#9CF;
}
<ul id="width" class="clearfix">
  <li>列表项</li>
  <li>列表项</li>
  ……
</ul>
```

按照设定，应该是每行4个列表项，但是在IE内有时候会如图16-34所示显示。

这是对小数点取整造成的误差导致的。因此可能需要针对IE设定一个比100%稍小的宽度总和，例如：

```
#width li {
width:24.9%;
}
```

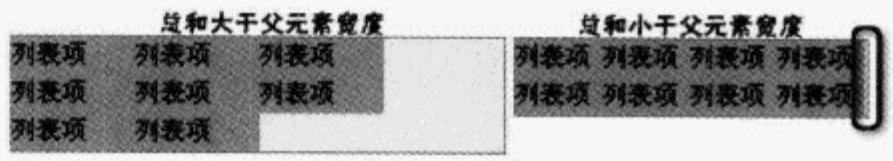


图16-34　IE对于百分比宽度计算的误差

5. 浏览器的默认设置是否清除

浏览器的默认设置有时候很容易被忽略，特别是在使用列表的时候，因此建议读者在样式表的最开始使用通配符来定义margin和padding属性，例如右上代码：

```
* {
margin : 0;
padding : 0;
}
```

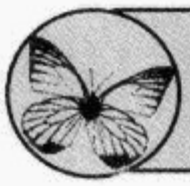
但是，除了这2个属性以外，建议不要再定义其他属性。为了消除带链接的图片的边框，可以再增加右下代码：

```
img {
border : 0;
}
```

16.4.3 Windows IE常见Bug

由于Windows IE的用户群庞大，同时，Windows IE6.0已经很多年没有更新，因此针对它的Bug报告层出不穷。而其他浏览器都和乐于修正Bug，进行升级，因此基本上没有特别重大的Bug存在。

浏览器的Bug有些是会经常碰到的，例如IE浮动双倍边距的Bug，有些需要特殊的条件才有可能触发，例如“断头台”Bug，有些可能需要很复杂的条件才会出现，本章只针对常见的Bug进行说明。



提示： 下列Bug大部分是IE 6.0及更早版本中的，IE 7.0修复了绝大部分Bug。本小节示例代码，读者可参见下载文件包内［/第2部分/第16章：浏览器与Hack/ bug.html］文件。

1. 浮动双倍边距（Doubled Float-Margin Bug）

在IE 6.0及更早的版本中，当一个左（右）浮动的元素同时有左（右）边距的时候，边距可能会被显示为双倍，这个Bug只在浮动元素是本行第1个浮动元素时发生。例如下边代码，其正确显示如图16-35所示。

而在IE 6.0内，其显示如图16-36所示。

```
div { …… }
.sample1{
float : left;
width : 100px;
padding : 0;
margin : 5px 10px 5px 30px;
}
.sample2{
float : right;
width : 100px;
padding : 0;
margin : 5px 30px 5px 10px ;
}
<div>
 <p class="sample1">左浮动1</p>
 <p class="sample1">左浮动2</p>
 <span class="clearFloat"></span>
 <p class="sample2">右浮动1</p>
 <p class="sample2">右浮动2</p>
 <span class="clearFloat"></span>
</div>
```

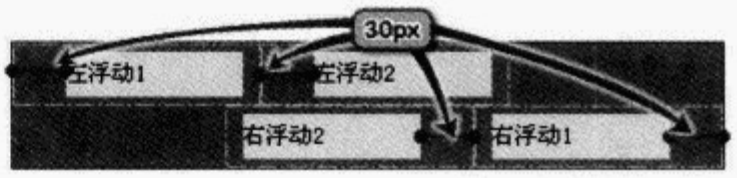


图16-35 浮动的元素的左边距正确显示

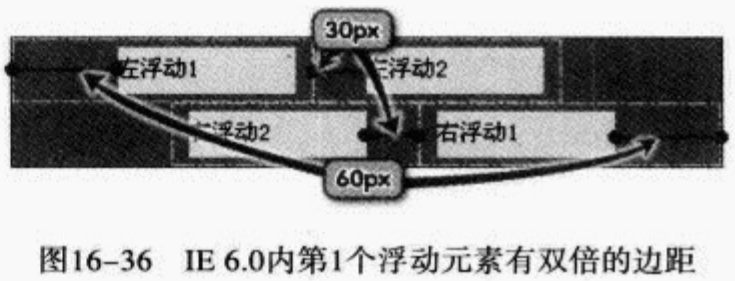


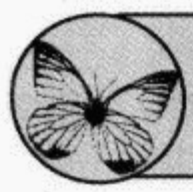
图16-36 IE 6.0内第1个浮动元素有双倍的边距

由图16-36可以发现：

- 只有与浮动方向相同的边距会加倍；
- 只有本行第1个浮动元素的边距会加倍。

这个问题很常见，解决的方法也很简单，为浮动元素添加CSS属性：“display : inline”，由于浮动的元素其display属性强制为“block”，因此设定“display : inline”不会产生影响，但是却可以解决IE 6.0双倍边距的问题。因此修改本例的CSS如右，则在IE 6.0内可恢复正常显示。

```
.sample1{
……
display : inline;
}
. sample2 {
……
display : inline;
}
```



提示： 关于这个Bug的英文资料请参阅http://www.positioniseverything.net/explorer/doubled-margin.html。

2. 3px文本偏移（Three Pixel Text-Jog Bug）

当一段文字与一个浮动元素相邻的时候，如果不希望文字环绕浮动元素，一般可以给包含文字一个同浮动元素等宽（或略宽）的边距，例如下列代码，其正确显示如图16-37所示。

```
.sample3 {
……
float:left;
width:100px;
}
.sample4 {
……
margin:0 0 0 100px;
}
<div>
 <p class="sample3">左浮动</p>
   <p class="sample4">不浮动的文字。不浮动的文字。不浮动的文字。不浮动的文字。不浮动的文字。不浮动的文字。不浮动的文字。不浮动的文字。不浮动的文字。不浮动的文字。</p>
</div>
```

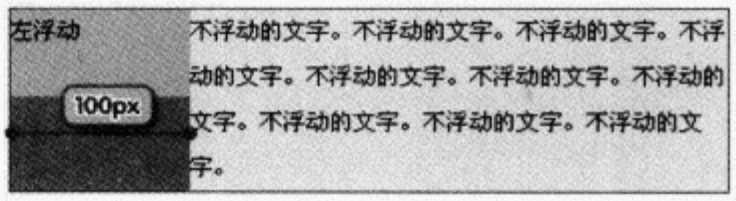


图16-37　利用左边距避免文字环绕

这也是左右2列布局经常用到的方式。但是，在IE 5.0/6.0中，位于浮动元素旁的文字会产生3px的偏移，如图16-38所示。

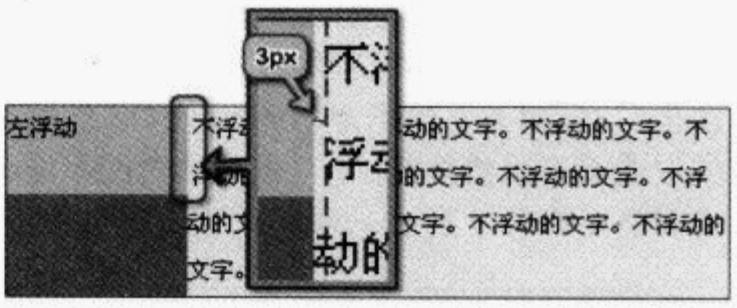
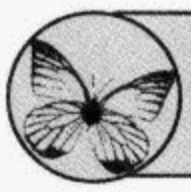


图16-38　IE 5-6中产生3px文字偏移

这个Bug也经常会出现，而要修复它比较复杂。首先，要使不浮动的段落具有Layout。

为其添加“height : 1%”，这是一个常用的触发Layout的CSS声明，在前面的例子里，使用过和它类似的“height : 100%”，但是，如果元素的父元素具有一个高度，“100%”会改变元素的高度，而由于IE 6及更早的版本，当实际的高度超过height的值的时候，会自动扩展高度，因此可以认为：IE 6.0将height作为min-height来看待，因此为元素设定“1%”的高度不会影响实际的显示。当然，对于其他的浏览器则会有影响，因此需要使用Hack来只让IE 6.0及更早的版本识别这条声明，具体代码如右：

```
* html .sample4 {
height : 1%;
margin-left : 0;
}
```



提示：针对IE 6.0设定“height : 1%”的方法被称为“Holly Hack”。“height : 1%”会对Mac IE 5.0产生影响，因此在实际应用中，还需要针对Mac IE 5.0隐藏此规则。

此时在IE 6.0内显示如图16-39所示。此时需要针对浮动元素设定-3px的右边距来消除这个间隙，CSS如下：

```
* html .sample3 {
margin-right : -3px;
}
```

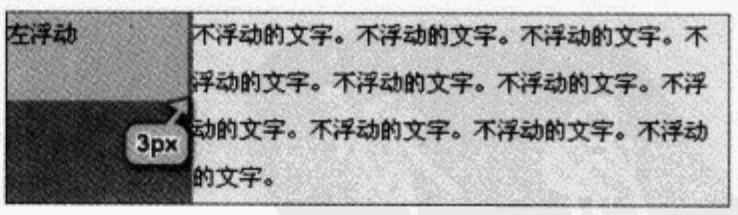
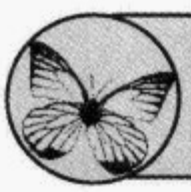


图16-39　触发Layout后不浮动元素与浮动元素间还有3px的间隙



提示：其实在很多情况下，3px的误差是可以忍受的。关于该Bug的英文资料可参阅http://www.positioniseverything.net/explorer/ threepxtest.html。

3. 复制文字Bug（Duplicate Characters Bug）

这个也是涉及浮动元素的Bug，但是它只出现在IE 6.0中，IE 5.5和IE 7.0中都没有此问题。

当一系列浮动元素的第1个和最后1个元素之间有多个注释的时候，同时，浮动元素的宽度会占满父元素宽度时，就会出现这个Bug。例如下列代码，在IE 6.0内显示如图16-40所示。

```
p{
margin : 0;
padding : 0;
float : left;
width : 100%;
}
<div>
 <p>浮动的p1</p>
 <p>浮动的p2</p>
 <!-- 注释文字1 -->
 <!-- 注释文字2 -->
 <p>浮动的p3 </p>
</div>
```

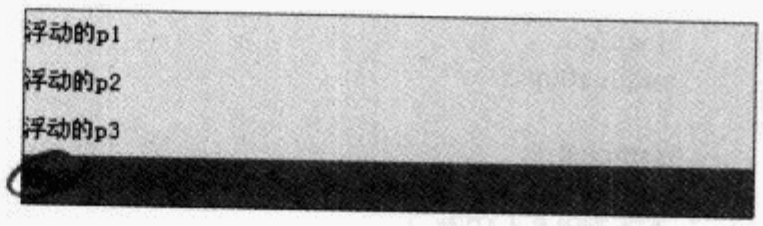
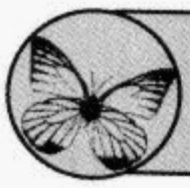


图16-40 IE 6.0内复制文字Bug的表现

被复制的文字的字数与注释的数量有一定的关系，增加注释的数量，则被复制的文字增多，减少注释的数量，则减少。

如果浮动元素的宽度没有撑满父元素，如“width : 99%”，则不会出现这个Bug。其实这个问题也与3px相关，由于浮动元素旁边多出的3px距离导致这个问题。因此可以为浮动元素设定-3px的右边距可以修复此Bug，但是负边距可能会导致其他问题，因此最稳妥的方式，就是删除注释，或者将注释移动到这一系列浮动元素的前面或者后面。



提示： 关于该Bug的英文资料可参阅http://www.positioniseverything.net/explorer/ dup-characters.html。

4. 躲躲猫Bug（Peekaboo Bug）

这个Bug的名字源于它的表现，在某些条件下，文本会在IE 6.0内消失，其条件如下：

- 一个浮动元素后面跟着一些非浮动元素，然后是一个清除元素；
- 所有这些元素都包含在一个设置了背景颜色或者图片的父元素中，且这个父元素没有设定宽度或高度（即没有触发Layout）；
- 清除元素碰到浮动元素。

当满足以上条件时，中间的非浮动元素看起来会消失，实际上是隐藏到了父元素的背景之后，只有在刷新页面（或者调整窗口大小使元素重排）时才会重新出现。例如下列代码，其正常显示如图16-41所示，而在IE 6.0内，显示如图16-42所示。

```
div {
background:#9CF;
border:2px solid #06F;
}
p {
margin:0;
padding:0;
background:#FFC;
border:1px solid  #C60;
}
.sample5 {
background:#FC3;
float:left;
width:100px;
height:100px;
}
<div>
```

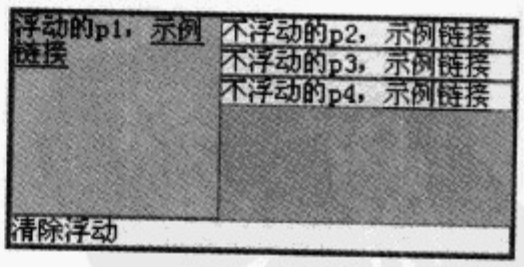


图16-41 非浮动元素正常显示

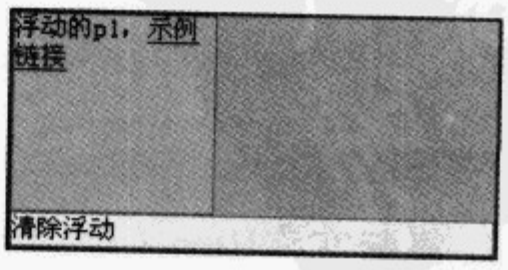
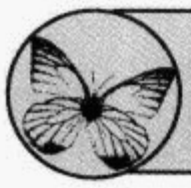


图16-42 IE 6.0内非浮动元素失踪

```
<p class="sample5">浮动的p1，<a href="#">示例链接</a></p>
<p>不浮动的p2，<a href="#">示例链接</a></p>
<p>不浮动的p3，<a href="#">示例链接</a></p>
<p>不浮动的p4，<a href="#">示例链接</a></p>
<p class="clearFloat">清除浮动</p>
</div>
```



提示： 读者可以参见下载文件包内［/第2部分/第16章：浏览器与Hack/ bug_peekaboo.html］文件。

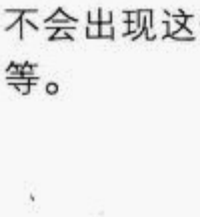
解决这个问题的方法有很多，最简单的方法就是给父元素设定行高，甚至为<body>设定一个行高都可以避免这个问题。例如：

```
div { line-height : 1; }
```

或者去除父元素的背景，也可以解决这个问题，但是这往往是不可行的。如果父容器有特定的尺寸，也可以避免这个Bug。例如：

```
div { _height : 1%; }
```

避免清除元素与浮动元素接触，也可以解决这个问题。将清除元素和浮动元素的position属性设为“relative”也可以缓解这一症状。



提示： 关于该Bug的英文资料可参阅http://www.positioniseverything.net/explorer/peekaboo.html。

5. 断头台Bug（Guillotine Bug）

断头台的名称也很形象，就如同断头台一样，在某种条件下，元素会被切断，不过切掉的不是头部而是底部。例如下列代码：

```
.sample5 {
background:#FFC;
border:2px solid #FC0;
width:150px;
margin:0;
float:left;
}
#guillotine a {
display:block;
}
#guillotine a:hover {
background:#FFF;
}
<div id="guillotine">
  <p class="sample5">断头台的名称也很形象，就如同断头台一样，某种条件下，元素会被切断，不过切掉的不是头部而是底部。</p>
  <a href="#" title="示例链接">示例链接1</a>
  <a href="#" title="示例链接">示例链接2</a>
  <a href="#" title="示例链接">示例链接3</a>
</div>
```

这段代码在IE 6.0及更早的版本中显示如图16-43所示。

由图16-43可以发现，当鼠标指向右边的第2个（或第3个）链接的时候，左边浮动元素的下半部分被砍掉，只剩下同右边3行链接文字的高度，而当指向第1个链接的时候，将恢复正常状态。引发这个问题的原因在于“a:hover”的设定，如果不设定“a:hover”的背景色，则不会出现这个问题，与背景类似可以触发这个问题的属性还有padding、border、加粗、斜体等。

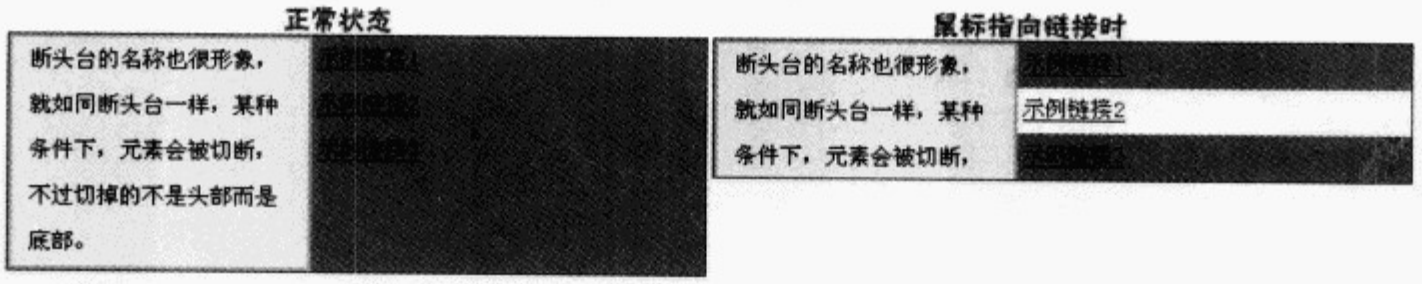


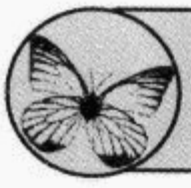
图16-43 断头台Bug的表现

由于左边是浮动元素，右边是非浮动元素，因此IE在计算 高度的时候，会将这些元素都计算在内，但是当发生了“a:hover”，IE认为这些属性会改变 的高度，因此要重新计算，但是又没能计算正确，只是把非浮动元素的高度赋予了 （这点倒是符合CSS关于高度的规定），因此造成了此现象。

因此解决方法有：

- 用1个新的浮动元素将非浮动元素包含起来；
- 在最下面增加1个清除浮动的元素，这样也可以解决非IE浏览器内高度自适应的问题。

这个例子和“3px文本偏移”的例子很相近，这说明IE在处理浮动的时候经常会出现问题。



提示： 关于该Bug的英文资料可参阅http://www.positioniseverything.net/explorer/guillotine.html。

6. 浮动元素下边距无效

虽然IE会自动包含浮动元素，但是也不是完美的，例如右列代码，其在IE内显示如图16-44所示。

这个问题出现在IE 6.0/7.0中，其原因就是没有闭合浮动元素造成的，这个问题即使触发了元素的hasLayout属性也无法解决，因为它同断头台Bug类似，是对元素高度计算的错误造成的，因此解决方法，可以在浮动元素后面添加清除元素，或者不设定浮动元素的margin-bottom属性，而设定 的padding-bottom来解决。例如：

```
div { padding-bottom : 30px; }
div p { margin : 5px 5px 0; }
```

```
div { margin : 0; }
p {
float : left;
width : 150px;
margin : 5px 5px 30px; /* 缩写：上 左右 下 */
display : inline;
}
<div>
 <p>浮动元素1</p>
 <p>浮动元素2</p>
</div>
<div>后面的div</div>
```

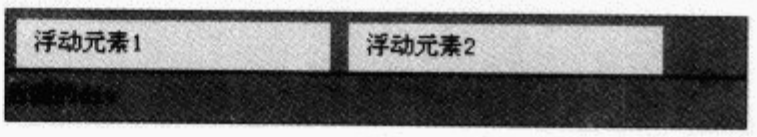


图16-44 浮动元素的下边距无效

7. “display : block”多出的空白

这是一个IE中列表常见问题。当<li>中的内容是一个<a>元素，且<a>元素设置了“display : block”，此时IE 6.0及更早的版本中，将不会忽略(X)HTML文档中列表标签之间的空格，而且通常会显示成额外的一行夹在<li>之间。例如下列代码，其显示如图16-45所示。

```
li a {
display : block;
background : #FF9;
}
<div>
 <ul>
  <li><a href="#" title="示例链接">示例链接1</a></li>
  <li><a href="#" title="示例链接">示例链接2</a></li>
  <li><a href="#" title="示例链接">示例链接3</a></li>
```

图16-45 空格没有被压缩而是显示出来

```
  </ul>
</div>
```

一种避免这种竖直方向多余空白的解决方法是赋予这些锚点Layout，这样还有一个好处就是可以让整个锚点的矩形区域都可以响应鼠标点击。例如：

```
li a { *height:1%; }
```

删除<ul>内的所有空格也可以解决这个问题。例如：

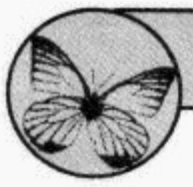
```
<ul><li><a href="#" title="示例链接">示例链接1</a></li><li><a href="#" title="示例链接">示例链接
2</a></li><li><a href="#" title="示例链接">示例链接3</a></li></ul>
```

8. 图片下的空隙

当一个元素内只含有图片，图片和元素的底部会有空隙，这是由于行高引起的，因为图片默认是基线对齐。例如下列代码：

```
<div id="image">
 <img src="../../img/ddcat_anim.gif" alt="示例" width="88" height="31" />
</div>
```

删除 和<img>标签间的空格可以解决这个问题，或者设置图片的vertical-align属性为“top”、“bottom”或者“middle”都可以解决这个问题。



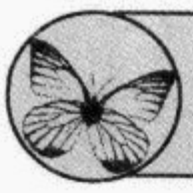
提示：严格来讲，这不算Bug，这个问题在其他浏览器内也存在。

9. 相对定位元素中的绝对定位

相对定位的元素会为其后代元素生成包含块，但是IE 6.0及更早的版本在计算偏移量的时候往往会出现问题。由于相对定位并不能触发元素的hasLayout属性，因此，造成了一些奇怪的现象。例如下列代码，其正确显示如图16-46所示。

```
#wrap {
position : relative;
……
}
#menu {
position : absolute;
right:0;
bottom:0;
……
}
<body>
<div id="wrap">
 <div id="menu">绝对定位的div</div>
 <p>静态的p</p>
 ……
</div>
<p>静态的p</p>
……
</body>
```

图16-46　相对定位元素内的绝对定位元素的正确位置



提示：读者可参见下载文件包内［/第2部分/第16章：浏览器与Hack/ bug_position. html］文件。

但是在IE 6.0内显示如图16-47所示。为了解决这个问题，需要针对IE 6.0触发相对定位元素的hasLayout属性，CSS规则如下：

```
* html #wrap { height : 1%; }
```

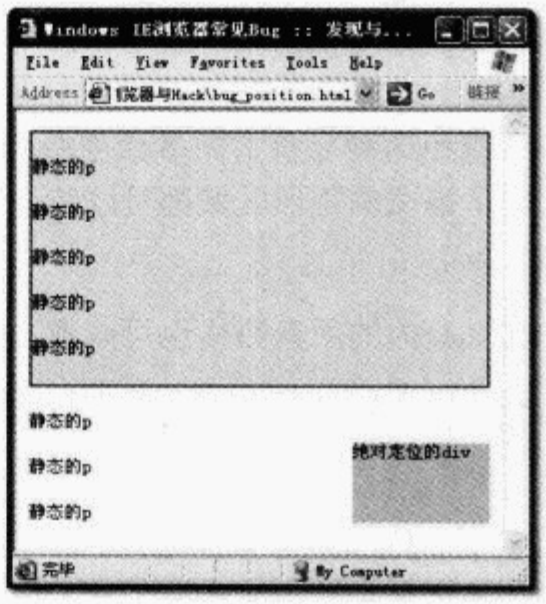


图16-47 IE 6.0对于绝对定位元素偏移量计算不正确

10. IE 6.0行高失效

在本书［7.3.4 浏览器的差别与错误］一节内介绍过，含有替换元素的行，在IE 6.0内行高会失效，因此当利用行高来使替换元素（如图片）垂直居中时会遇到问题。例如下列代码，其正确显示如图16-48所示。

```
#lineHeight { line-height : 50px; }
#lineHeight img { vertical-align : middle; }
<div id="lineHeight">
 <img src="../../img/ddcat_anim.gif" alt="示例" width="88" height="31" />
</div>
```

而在IE 6.0内显示如图16-49所示。

图16-48 利用行高使图片垂直居中

图16-49 IE 6.0行高失效

解决这个问题的方法是针对IE 6.0设定元素的上下补白，CSS规则如左边代码，或者采用右边代码：

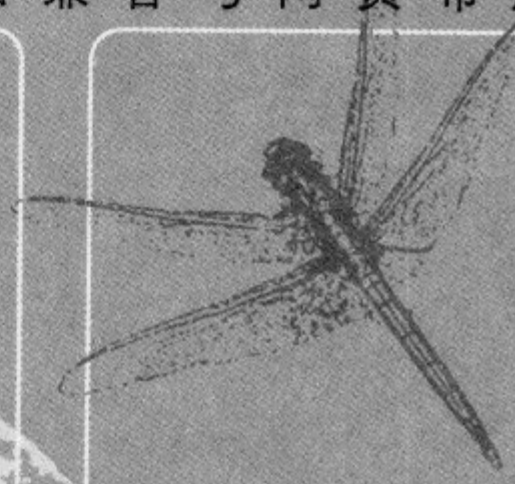
```
* html #lineHeight { padding : 10px 0; }
```

```
#lineHeight { _padding:10px 0; }
```

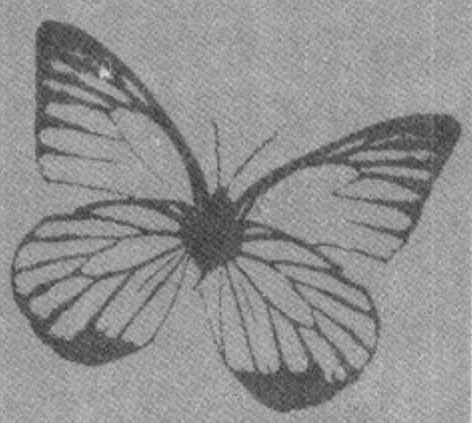
以上列举的只是IE内比较常见、影响比较大的Bug，还有很多Bug触发的条件很复杂，因此很难重现，而有些问题可能还没有完美的解决方案。因此，在很多情况下，可能需要根据实际情况调整设计方案及元素的关系、CSS的设定等综合解决问题。

不过由于IE 7.0修复了绝大部分的Bug，因此在IE 7.0普及时，这些问题将成为过去。

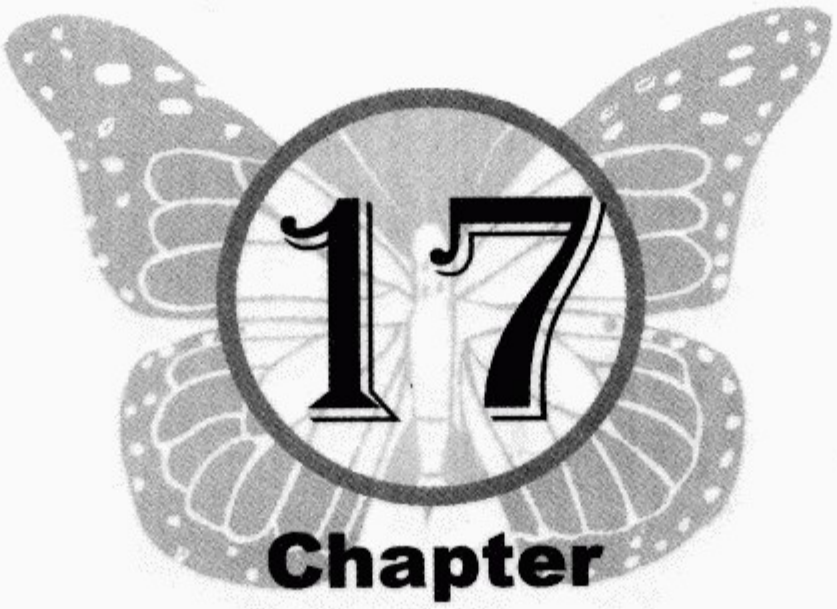
CSS 属性、浏览器兼容与网页布局



第3部分 结构化实例



别具光芒 CSS

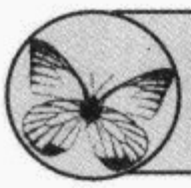


第17章
旅游网站

在本书［2.4.2 静态页面制作］一节内介绍过静态页面制作的基本步骤：构建内容的结构→提取尺寸及图片→样式表美化→细节处理→优化样式表。在本章将实际制作一个旅游网站的首页页面。

17.1 结构化

虽然页面的内容是一定的，但是内容的结构却不是唯一的。不同的制作者对于相同的内容可能会采用不同的(X)HTML标签来构建，但是，在大部分情况下，如何结构化还是有章可循的。



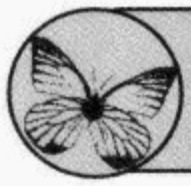
提示：在本书下面的制作过程中所介绍的结构化方法，只是常用的方法之一，而不是唯一的方法，读者也可以自己尝试使用其他的方法。

结构是以内容为基础的，而不是以外观表现为基础。初学者往往会执着于设计图的视觉效果，而造成结构的不合理、标签的多层嵌套，以及装饰性图片与内容的混杂等问题。因此在结构化内容的时候，暂时要忘记设计图是如何精美的，只是关注实际的内容是什么，不要考虑类似“这个圆角怎么做”、“这条虚线怎么做”这样的问题。

当把结构代码完成以后，再根据实际情况考虑是否必需添加一些额外的标签以完成实现外观表现的要求。

17.1.1 分析内容结构

虽然大部分的页面制作不是从内容策划开始而是从设计图开始，不过，由设计图也可以总结出页面的基本内容，从而确定其结构。本例网站首页的设计如图17-1所示。



提示：读者可以参见下载文件包内［/第3部分/第17章：旅游网站/设计原稿/首页.png］文件。

由图17-1可以发现，网页基本分为上、中、下3部分：

- 页首部分是网站Logo、主导航条、用户登录和登录用户功能菜单4个模块，如图17-2所示；
- 中部是主要内容区，其中又分为若干不同的栏目模块，同时各板块内容的重要程度也不相同，如图17-3所示；
- 页脚部分是网站副导航条和版权信息，如图17-4所示。

图17-1　首页设计图

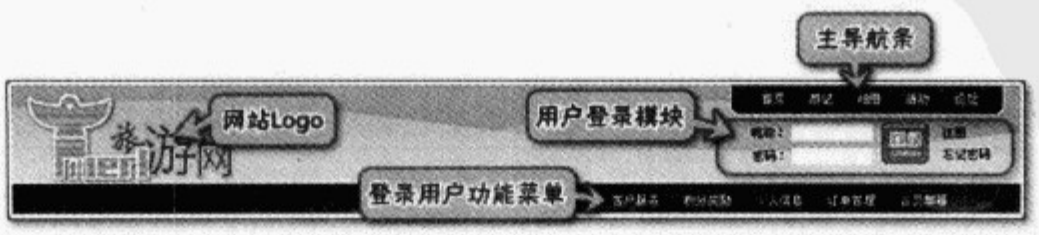


图17-2　网页页首部分内容分析

图17-3 网页中部内容分析

图17-4 网页页脚部分内容分析

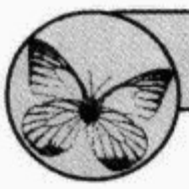
17.1.2 基本结构

编写(X)HTML文档结构的过程，就如同给一篇文章排版，主要需要注意以下几点：

- 选择适合的DTD类型；
- 设置正确的<title>元素和页面字符编码；
- 为页面添加适当的说明（description）和关键字（keywords）；
- 用(X)HTML标签将1级、2级、3级标题、列表、正文、引用文字等标记出来；
- 为了使结构清晰，需要使用一些标签把相关的内容“包含”起来，例如用 将页首、页中和页脚内容分开，而各栏目内容也可用 包含；
- 给关键的标签设定ID，如包含“游记精选”栏目的 。

本例文档<head>部分XHTML代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="zh-CN" lang="zh-CN">
<head>
<title>首页 :: Imerl旅游网</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<meta name="description" content="Imerl旅游网站，服务项目：自助游，旅行社，预订车票、旅店、宾馆，旅游景点折扣。" />
<meta name="keywords" content="旅游,自助游,旅行社,预订车票,旅店,宾馆,折扣票价" />
</head>
```



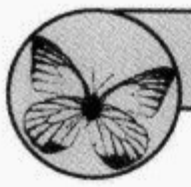
提示：读者可参见下载文件包内［/第3部分/第17章：旅游网站/site/index.html］文件。

说明（description）和关键字（keywords）不同，说明是一段叙述性的描述网站内容的文字，

而关键字（keywords）则是提供给搜索程序用的，因此应该是一些用英文逗号分隔的关键文字。同时，关键字的设置要实事求是，不要包含与本站内容无关的词汇。

本例中，可以使用3个 将页头、页中和页脚部分分开，XHTML如右：

```
<body>
 <div id="header"></div>
 <div id="main"></div>
 <div id="footer"></div>
</body>
</html>
```



提示：关于选择器的命名规范，读者可参考本书［4.8 命名规范］一节。

有理论认为，在编写CSS的时候，尽量使用类选择器而不要使用ID选择器，即在(X)HTML内使用class属性指定类选择器而不用id选择器，例如： ，而ID应该供JavaScript或者其他程序使用，以避免程序人员修改ID或者其他操作而造成CSS失效。

不过这个问题也可以通过制作人员和程序人员之间的交流协调解决，因此这两种选择器都可以使用。在本例中将使用id属性。

17.1.3 页首部分的结构化

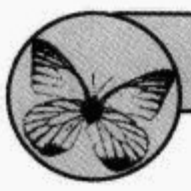
由图17-2可以发现页首部分的结构比较简单，包括以下4个部分。

1. 网站Logo

虽然设计图上只有一个Logo，但是实际的XHTML中应该包括网站的名称，这样不仅利于SEO，同时当用户端不支持（或者屏蔽了）CSS及不显示图片的时候，访问者也能知道访问的是什么网站。

对于网页来讲，网站的名称可以认为是其顶级的标题，因此使用<h1>来放置网站名称是比较合适的选择：

```
<h1><a href="/index.html" title="前往[Imerl旅游网]首页">Imerl旅游网</a></h1>
```



提示：理论上讲，1级标题<h1>在网页内只出现一次。

对于Logo图片，使用<img>标签将图片直接插入到XHTML内：

```
<h1><a href="/index.html" title="前往[Imerl旅游网]首页"><img src="img/logo.gif" alt="网站Logo"
width="209" height="100" />Imerl旅游网</a></h1>
```

2. 主导航条

导航条一般使用无序列表<ul>制作，这样在没有CSS的时候也可以清晰地显示：

```
<ul id="mainNav">
 <li><a href="/index.html" title="网站首页">首页</a></li>
 <li><a href="/blog/index.html" title="[游记]栏目首页">游记</a></li>
 <li><a href="/photos/index.html" title="[相册]栏目首页">相册</a></li>
 <li><a href="/activities/index.html" title="[活动]栏目首页">活动</a></li>
 <li><a href="/forum/index.html" title="[论坛]首页">论坛</a></li>
</ul>
```

3. 用户登录模块

用户登录模块部分，需要使用表单，而由于表单元素大部分为行内替换元素，因此一般使用<ul>和<li>标签（也可以使用<p>标签）等来格式化，例如：

```
<form id="formLogin" action="#">
 <fieldset>
 <legend>用户登录</legend>
 <ul class="loginInfo">
  <li>
   <label for="loginNickname">昵称：</label>
   <input type="text" id="loginNickname" maxlength="20" />  </li>
  <li>
   <label for="loginPassword">密码：</label>
   <input type="password" id="loginPassword" maxlength="20" />  </li>
  <li>
   <input type="submit" value="登录" id="btnLoginSubmit" title="单击登录" />
  </li>
 </ul>
 <ul class="reg">
  <li><a href="/member/reg.html" title="前往注册页面">注册</a></li>
  <li><a href="/member/password.html" title="前往找回密码页面">忘记密码</a></li>
 </ul>
 </fieldset>
</form>>
```

在此将登录信息放在一个<ul>内，而注册和找回密码放在一个<ul>内，分别定义class属性以区分其内容。

4. 登录用户功能菜单

这一部分稍微复杂一些，本菜单是一个2级菜单，当鼠标指向菜单文字的时候会显示2级菜单内容，如图17-5所示，其中包括了用户登录后可以进行的操作。

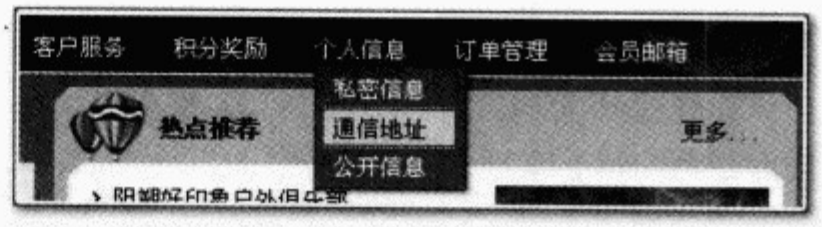
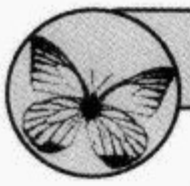


图17-5 用户功能菜单的2级菜单设计图

1级和2级菜单都可以使用<ul>来构建，即嵌套的<ul>，XHTML如下：

```
<ul id="controlMenu">
 <li><a href="#" title="客户服务项目查询">客户服务</a>
  <ul>
          <li><a href="#" title="预定国内飞机、火车、船票">定票</a></li>
          <li><a href="#" title="预定国内各地酒店">定酒店</a></li>
          <li><a href="#" title="预定国内旅行社行程">定行程</a></li>
         </ul>
 </li>
 <li><a href="#" title="积分奖励内容查询">积分奖励</a>
  <ul>
          <li><a href="#" title="查询积分情况">积分查询</a></li>
          <li><a href="#" title="使用积分换取礼品">积分换礼</a></li>
         </ul>
 </li>
 <li><a href="#" title="修改个人信息">个人信息</a>
  <ul>
          <li><a href="#" title="修改不公开的私密信息">私密信息</a></li>
          <li><a href="#" title="修改通信地址">通信地址</a></li>
          <li><a href="#" title="修改对其他用户公开的信息">公开信息</a></li>
         </ul>
 </li>
 <li><a href="#" title="会员订单查询管理">订单管理</a>
  <ul>
          <li><a href="#" title="目前正在执行中的订单">订单查询</a></li>
          <li><a href="#" title="已结订单">历史记录</a></li>
         </ul>
 </li>
 <li><a href="#" title="查看会员邮箱">会员邮箱</a></li>
</ul>
```



提示： 和<li>嵌套需要特别注意<li>标签要成对。

此时页首部分的总体结构如下：

```
<div id="header">
  <h1><a href="/index.html" title="前往网站首页"><img src="img/logo.gif" alt="网站Logo" width="209"
height="100" />Imerl旅游网</a></h1>
  <ul id="mainNav">
    ……
  </ul>
  <form id="formLogin" action="#">
   ……
  </form>
  <ul id="controlMenu">
    ……
  </ul>
</div>
```

不过，为了使不同的内容模块之间的界限更加清晰，通常（但不是必需）会增加 标签分别将这4部分内容包裹起来，XHTML修改如下：

```
<div id="header">
 <div id="logo">
  <h1>……</h1>
 </div>
 <div id="mainNav">
  <ul>
   ……
  </ul>
 </div>
 <div id="login">
  <form id="formLogin" action="#">
  ……
  </form>
 </div>
 <div id="controlMenu">
  <ul>
  ……
  </ul>
 </div>
</div>
```

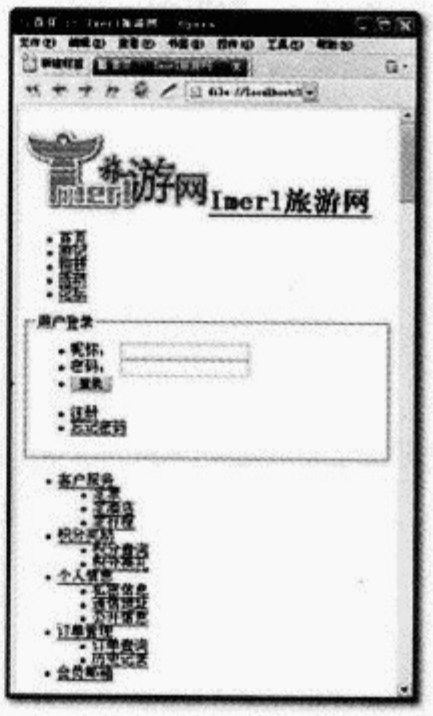


图17-6　页首部分在浏览器内的显示

至此，页首部分结构化完成，其在浏览器内显示如图17-6所示。

由图17-6可以发现，虽然没有CSS的美化，各项内容很清晰易辨这就是结构化所要达到的目的。

17.1.4 中间部分的结构化

中间是网站的真正内容部分，一共可分为9个内容模块，如图17-3所示。

由图17-7可以发现，网站整体看来是左右2列版式，可以使用2个层将内容分开，例如：

```
<div id="col1">
 1、5、6模块内容
</div>
<div id="col2">
 2、3、4、7、8、9模块内容
</div>
```

这样设定的好处是，只要浮动col1和col2层，就可以分成左右2列显示。

但是实际情况是，由于计算机屏幕的限制，访问者首先看到的是上半部分的内容，即1~4的内容，因此一般会将网站比较重要的、希望访问者最先看到的内容放置在这些区域，而5~9的内容在第2屏内，重要性次之。因此当页面不加载CSS，而让内容顺序显示时，其显示顺序应该如图17-7的序号所示。

图17-7 中间部分内容模块的书写顺序

1. 游记精选栏目

栏目的内容如图17-8和图17-9所示。

图17-8 游记精选栏目内容

图17-9 文章列表的具体内容

（1）栏目标题和小导航。

栏目标题和栏目的小导航条的结构比较简单：

```
<h2>游记精选</h2>
<ul class="subjectNav">
 <li><a href="#" title="发表新的游记">发表游记</a></li>
 <li><a href="#" title="查看更多的游记">更多游记</a></li>
</ul>
```

对于栏目小导航条设定了class属性为“subjectNav”，而没有使用ID，因为其重要性比较低，同时，小导航在1个栏目内可以重复出现，因此使用了类选择器。

（2）文章列表具体内容。

对于游记文章的列表部分，可以使用<ol>（<ul>）或者 来构建，在此首先讨论文章列表的具体内容的格式化。

文章列表的具体内容如图17-9所示，包括文章标题、作者、发表日期等5部分内容。文章标题使用<h3>来构建，此时需注意，标题前面有前置文字，如“推荐”、“国内”、“国外”等（根据后台设置），理论上，这个文字可以利用:before伪元素和content属性来生成，例如：

```
. commend:before { content : "［推荐］"; }
. domestic:before { content : "［国内］"; }
. overseas:before { content : "［国外］"; }
<h2 class=" commend">文章的标题文字</h2>
<h2 class=" domestic">文章的标题文字</h2>
<h2 class=" overseas">文章的标题文字</h2>
```

但是由于IE不支持:before伪元素，因此只能依靠后台程序生成具体的文字。由于“推荐”和“国内（外）”的重要程度不同，因此可以分别使用和来结构化：

```
<h3><strong>[推荐]</strong><a href="#" title="文章标题">这里是文章的标题</a></h3>
<h3><em>[国内]</em><a href="#" title="文章标题">这里是文章的标题</a></h3>
```

作者的名字，可以使用<h4>来结构化：

```
<h4>作者：<a href="#" title="进入[豆豆猫]的Blog">豆豆猫</a></h4>
```

发表日期按照其重要程度，可以使用<h5>来结构化：

```
<h5>发表日期：2008-02-28</h5>
```

图片和摘要文字都是具体的内容，可以使用<p>来结构化：

```
<p class="pic"><img src="pic/blog_01.jpg" title="文章的标题" alt="插图" width="128" height="88" /></p>
<p>文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的摘要内容。</p>
```

此处为放置图片的<p>元素定义class属性为“pic”，表明其内容是照片。

至此，文章列表的具体内容结构如下：

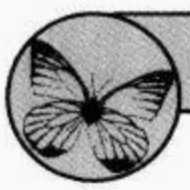
```
<h3><strong>[推荐]</strong><a href="#" title="文章标题">这里是文章的标题</a></h3>
<h4>作者：<a href="#" title="进入[豆豆猫]的Blog">豆豆猫</a></h4>
<h5>发表日期：2008-02-28</h5>
<p class="pic"><a href="/blog/" title="文章标题"><img src="pic/blog_01.jpg" title="文章的标题" alt="插图" width="128" height="88" /></a></p>
<p>文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的摘要内容。</p>
```

（3）文章列表的总结构。

文章列表的总结构可以使用<ol>（<ul>）：

```
<ol>
 <li>
  <h3><strong>[推荐]</strong><a href="#" title="文章标题">这里是文章的标题</a></h3>
  <h4>作者：<a href="#" title="进入[豆豆猫]的Blog">豆豆猫</a></h4>
  ……
 </li>
 <li>
  <h3><strong>推荐</strong><a href="#" title="文章标题">这里是文章的标题</a></h3>
  ……
 </li>
 ……
</ol>
```

也可以使用 来结构化：



提示：读者可以参见下载文件包内［/第3部分/第17章：旅游网站/site/travels. html］文件。

```
<div>
 <h3><strong>[推荐]</strong><a href="#" title="文章标题">这里是文章的标题</a></h3>
 <h4>作者：<a href="#" title="进入[豆豆猫]的Blog">豆豆猫</a></h4>
 ……
</div>
<div>
 <h3><strong>推荐</strong><a href="#" title="文章标题">这里是文章的标题</a></h3>
 ……
</div>
```

这两种结构在浏览器内显示如图17-10所示。使用<ol>无疑可以使结构更清晰，同时也更合

理，因为文章的推荐本身更应该是列表。至此，游记精选栏目的结构如下：

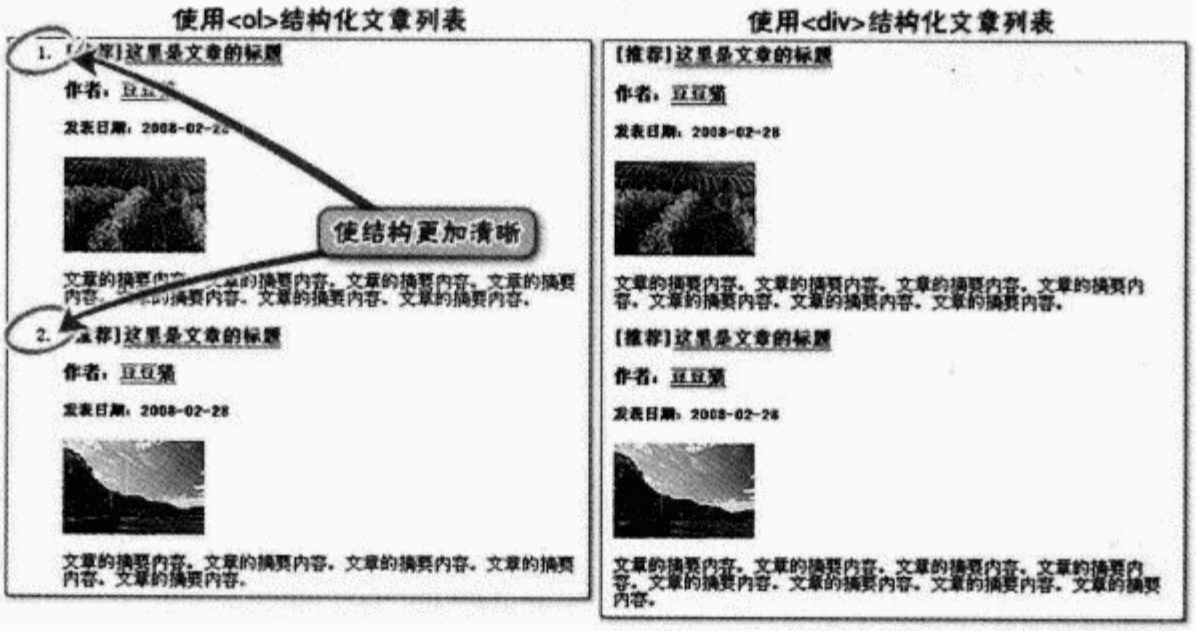


图17-10 使用<ol>和 结构化的区别

```
<div id="travels">
 <h2>游记精选</h2>
 <ul class="subjectNav">
  ……
 </ul>
 <ol class=" travelsList">
  <li>
    <h3><strong>推荐</strong><a href="#" title="文章标题">这里是文章的标题</a></h3>
    <h4>作者：<a href="#" title="进入[豆豆猫]的Blog">豆豆猫</a></h4>
    <h5>发表日期：2008-02-28</h5>
        <p class="pic"><img src="pic/blog_01.jpg" title="文章的标题" alt="插图" width="128" height="88"
/></p>
        <p>文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的
摘要内容。文章的摘要内容。</p>
      </li>
      ……
    </ol>
  </div>
```

2. 热点推荐栏目

热点推荐栏目的内容分析如图17-11所示。

栏目内容相对来说比较简单，其总体结构如下：

```
<div id="hot">
 <h2>热点推荐</h2>
 <ul class="subjectNav">
  <li><a href="#" title="查看更多的游记">更多...</a></li>
 </ul>
 <ol class="hotArticle">
  <li><a href="#" title="这里是热点推荐的标题">这里是热点推荐的标题</a></li>
  ……
 </ol>
 <ol class="hotPicture">
     <li><a href="#" title="标题文字"><img src="pic/ad_01.jpg" alt="热点图片" width="140" height="60"
/></a></li>
    ……
 </ol>
</div>
```

在此也是利用class属性表明<ol>元素的内容。

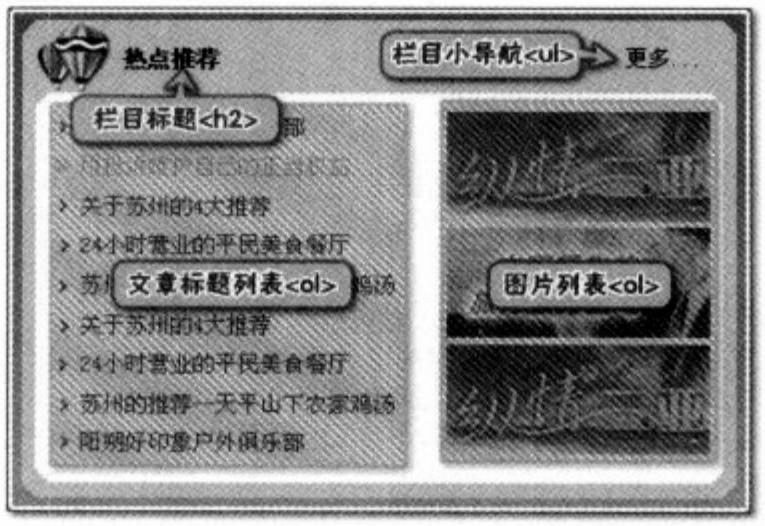


图17-11　热点推荐栏目内容分析

3. 广告栏目

广告栏目的内容很简单，只有1个<ul>列表如下：

```
<div id="ad1">
 <ul>
 <li><a href="#" title="这里是广告的标题1">这里是广告的标题1</a></li>
  ……
 </ul>
</div>>
```

4. 照片精选栏目

这个栏目内容和“热点推荐”内容类似，如图17-12所示。

栏目的总体结构如下：

```
<div id="photos">
  <h2>照片精选</h2>
  <ul class="subjectNav">
   <li><a href="#" title="发表新的游记">发表照片</a></li>
   <li><a href="#" title="查看更多的游记">更多...</a></li>
  </ul>
  <ul class="photoList">
    <li><a href="#" title="图片1的标题文字"><img src="pic/photo_01.jpg" width="100" height="80" alt="照片
精选1" /></a>
     <strong><a href="#" title="图片1的标题文字">图片的标题文字</a></strong>
    </li>
    ……
   </li>
  </ul>
 </div>
```

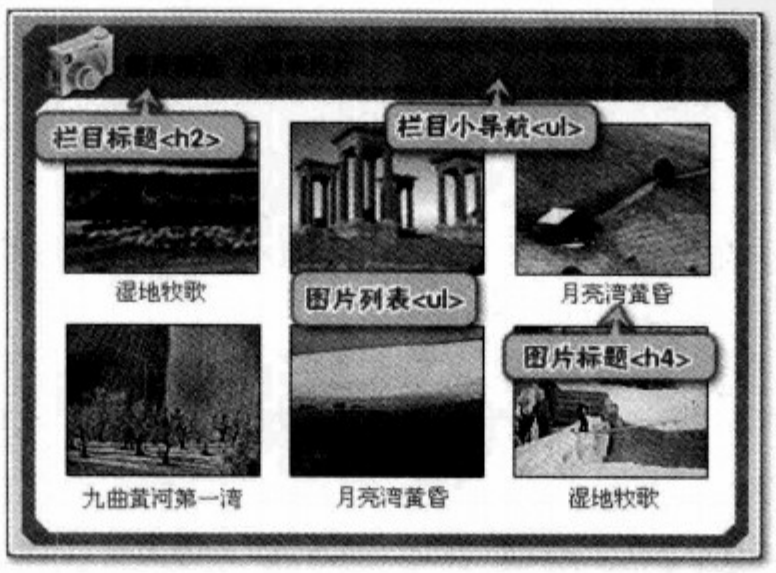


图17-12　照片精选栏目内容分析

对于照片的说明，使用了元素，以起到“特别强调”的作用。

5. 论坛导航

论坛导航部分的内容比较简单，一个论坛的链接列表：

```
<div id=" forumList">
 <ul>
  <li><a href="#" title="进入[宾馆点评]分论坛">宾馆点评</a></li>
  <li><a href="#" title="进入[餐馆点评]分论坛">餐馆点评</a></li>
  <li><a href="#" title="进入[景点点评]分论坛">景点点评</a></li>
  <li><a href="#" title="进入[娱乐场所点评]分论坛">娱乐场所点评</a></li>
 </ul>
</div>
```

6. 论坛热贴栏目

这个栏目内包括4个论坛的帖子内容，但是这几部分的结构基本是相同的，如图17-13所示。对于“自助线路”和“七嘴八舌”，帖子列表可以使用以下的结构：

```
<ul>
  <li><a href="#" title="这里是帖子的标题1" class="postTitle">这里是帖子的标题</a> <a href="#" title="查看[豆豆猫]的个人资料" class="postMember">豆豆猫</a></li>
  ……
</ul>
```

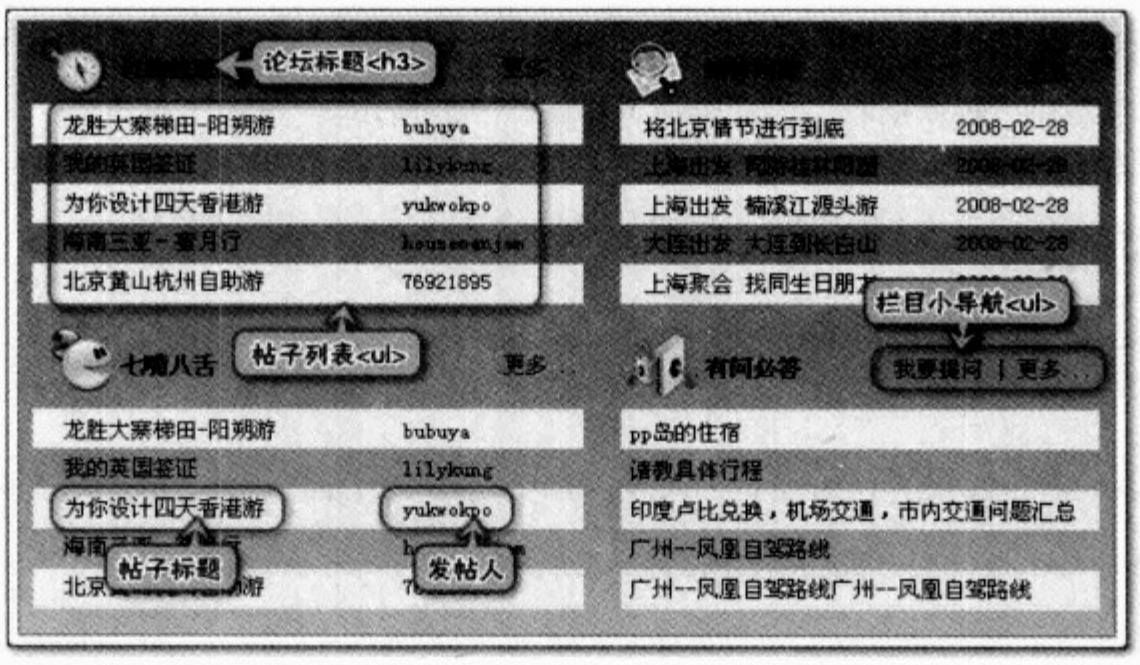


图17-13 论坛热贴栏目内容分析

而“结伴同游”中帖子的标题后面是发帖的时间，因此需要稍加修改：

```
<ul class="hotList">
   <li><a href="#" title="这里是帖子的标题" class="postTitle">这里是帖子标题</a>  <em>2008-02-28</em></li>
  ……
</ul>
```

由于“结伴同游”具有时效性，因此在此使用标签来强调日期。“有问必答”栏目内只有帖子的标题。“论坛热贴”栏目最后的结构如下：

```
<div id="forumHot">
 <div id="budgetLine">
  <h3>自助线路</h3>
  <ul class="subjectNav">……</ul>
  <ul class="hotList">
   <li><a href="#" title="这里是帖子的标题" class="postTitle">这里是帖子标题</a> <a href="#" title="查看[豆豆猫]的个人资料" class="postMember">豆豆猫</a></li>
```

```
        ……
      </ul>
    </div>
    <div id="company">
      <h3>结伴同游</h3>
      <ul class="subjectNav">……</ul>
      <ul class="hotList">
          <li><a href="#" title="这里是帖子的标题" class="postTitle">这里是帖子标题</a> <em>2008-02-
28</em></li>
        ……
      </ul>
    </div>
    <div id="colloquy">
      <h3>七嘴八舌</h3>
      <ul class="subjectNav">……</ul>
      <ul class="hotList">
        <li><a href="#" title="这里是帖子的标题" class="postTitle">这里是帖子标题</a> <a href="#" title="查看
[豆豆猫]的个人资料" class="postMember">豆豆猫</a></li>
        ……
      </ul>
    </div>
    <div id="question">
      <h3>有问必答</h3>
      <ul class="subjectNav">……</ul>
      <ul class="hotList">
        <li><a href="#" title="这里是帖子的标题" class="postTitle">这里是帖子的标题的标题</a></li>
        ……
      </ul>
    </div>
  </div>
```

7. 俱乐部栏目

本栏目内容分析如图17-14所示。

```
  <div id="club">
    <h2>俱乐部专区</h2>
    <ul class="subjectNav">……</ul>
    <ul class="clubList">
        <li><a href="#" title="查看[Imerl自由行俱乐部]详情"><img src="pic/clubLogo_ 1.gif" width="65"
height="48" alt="Imerl自由行俱乐部Logo" />《Imerl自由行》俱乐部</a></li>
      ……
    </ul>
    <ul class="activitiesList">
      <li><a href="#" title="这里是帖子的标题">这里是帖子的标题的标题</a></li>
      ……
    </ul>
  </div>
```

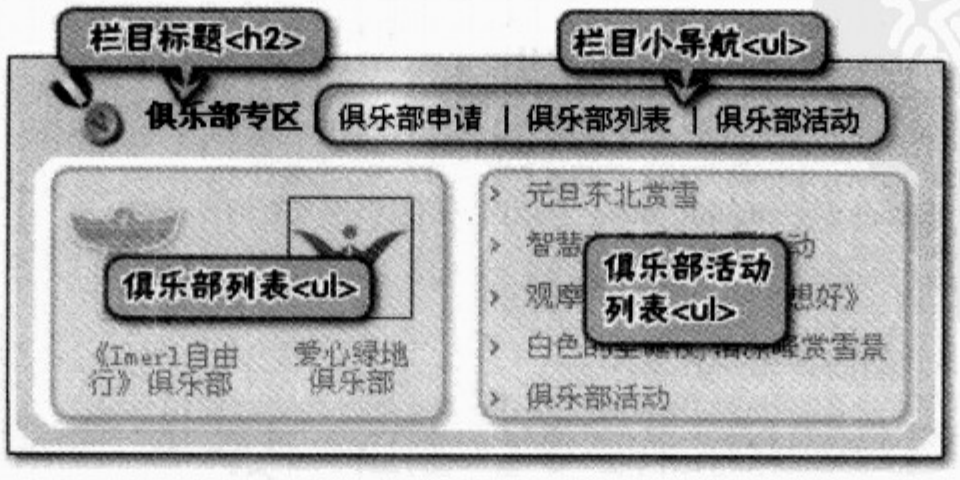


图17-14　俱乐部栏目内容分析

8. 投票栏目

本栏目是一个投票表单，对于使用单选按钮的5个选项，可以使用<label>标签来结构化：

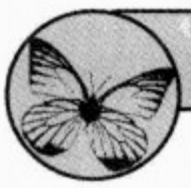
```
<label><input type="radio" name="choice" value="1" />北上赏雪，领略北国边疆飘洒的漫天飞雪</label>
<label><input type="radio" name="choice" value="2" />南下避寒，享受南国海滨温暖的冬日阳光</label>
……
```

但是这样在IE 6.0内无法通过点击文字选中选项，不方便用户选择，因此需要修改如下：

```
<input type="radio" name="choice" id="choice1" value="1" /><label for="choice1">北上赏雪，领略北国边疆飘洒的漫天飞雪</label>
```

同时，可以使用<ul>来结构化：

```
<div id="vote">
 <form id="voteForm" action="#">
  <fieldset>
  <legend>热点投票</legend>
  <p>今年冬季，你选择以哪种方式来度过这个寒冷的冬天？</p>
  <ul>
   <li>
    <input type="radio" name="choice" id="choice1" value="1" />
    <label for="choice1">北上赏雪，领略北国边疆飘洒的漫天飞雪</label>
   </li>
   <li>
    <input type="radio" name="choice" id="choice2" value="2" />
    <label for="choice2">南下避寒，享受南国海滨温暖的冬日阳光</label>
   </li>
   ……
  </ul>
    <p class="button"><input type="submit" id="btnVoteSubmit" value="投票" /><input type="submit" id="btnShowResults" value="查看结果" /></p>
  </fieldset>
 </form>
</div>
```



提示：在此使用<p>来放置按钮，也可以使用<ul>或者 。

9. 社区服务栏目

本栏目内容比较简单，其结构如下：

```
<div id="community">
 <h2>社区服务台</h2>
 <ul>
 <li><a href="/community/suggestion.html" title="为社区建设提意见建议">社区意见箱</a></li>
  <li><a href="/community/manage.html" title="社区管理">社区管理</a></li>
  <li><a href="/community/info.html" title="为社区提供信息">信息征集</a></li>
  <li><a href="/community/errata.html" title="为社区提供勘误信息">网络查错</a></li>
  <li><a href="/community/faq.html" title="社区使用说明">社区说明</a></li>
  <li><a href="/community/serch.html" title="查询网友信息">网友查询</a></li>
 </ul>
</div>
```

至此页面中间部分的内容结构化完毕。

17.1.5 页脚部分的结构化

页脚部分的内容比较简单，包括一个副导航条<ul>和版权信息<p>，如图17-4所示，其结构如下：

```
<div id="footer">
 <ul>
  <li><a href="/member/reg.html" title="前往注册页面">免费注册</a></li>
  <li><a href="/sitemap.html" title="查看网站地图">网站导航</a></li>
  ……
 </ul>
 <p>Copyright &copy; Imerl.com .All Rights Reserved.</p>
</div>
```

至此XHTML结构化完成，此时，页面虽然没有漂亮的外观，但是访问者依然能够清晰地浏览每部分的内容，这就是结构与表现分离的优点。

17.2 图片格式与网络基础知识

除了前面章节介绍的(X)HTML、CSS、浏览器等知识以外，网页制作往往还涉及很多其他方面的基础知识，其中同制作关系最密切的就是图片的格式与优化。

17.2.1 图片格式

图片是页面内最常见的元素之一。图片可以插在HTML代码中，也可以使用CSS设置成元素的背景图片，而根据图片的格式不同，其适用的地方也不太相同。除去内容图片，装饰图片应该通过设定元素的背景来实现。目前浏览器支持的图片类型有以下几种。

1. JPEG图片

JPEG文件是按Joint Photographic Experts Group（联合图片专家组）制定的压缩标准产生的压缩格式，属J—PEG File Interchange Format，可以用不同的压缩比例对文件压缩，这是到目前为止比较好的图像压缩技术，不过属于有损压缩。

JPEG图片支持真彩色，对于大部分的照片和颜色丰富的图片，一般使用这种文件格式。JPEG文件的后缀名通常为.jpg，而且浏览器都支持这种格式的图片文件。

为了减少访问者的等待时间、节约服务器空间以及带宽流量，牺牲一些质量也是可以忍受的。而且对于最高质量的压缩比，是几乎看不出损失的，因此JPEG格式文件是网页上最常出现的图像格式之一。

2. 静态GIF图片

Graphics Interchange Format，图形交换格式，这种格式是由CompuServe公司设计的，分为87a及89a两种版本，存储格式由1位到8位。这是专用于网络传输的文件格式，许多平台都支持GIF。

GIF支持24位彩色，“索引色”格式文件，由一个最多256种颜色的调色板实现，图像大小最多是64K×64K个像素点。GIF文件的后缀名通常为.gif，动画和静态图片的后缀是一样的。

GIF的特点有以下几项。

- LZW压缩：此种压缩为无损压缩，对图像没有损失。
- 多图像的定序或覆盖：即可实现动画效果。
- 交错屏幕绘图：显示的图象从网络下载时，先显示一个粗略的图像，然后逐步增加精度显示，但这种格式会增加文件大小。这种格式不能存储Alpha通道。
- 支持透明通道：GIF89a格式支持一个Alpha通道，因此可以制作背景透明的图像，以方便放置在不同的位置。

3. GIF动画

上文中提到，GIF文件支持多图像的定序或覆盖，并分为87a及89a两种版本，其中89a是制作2D动画软件Animator早期支持的文件格式，所以该格式被广泛使用来制作动画。

GIF动画的原理就是在一个文件内存储多帧（Frame）图像，然后按顺序显示，同时还可以设置每帧的延时时间。GIF动画目前仍被广泛应用，因为它不需要安装播放插件就可以在各种浏览器内显示。

4. PNG图片

PNG（Portable Network Graphic）图片结合了GIF和JPEG的优点，具有存贮形式丰富的特点。PNG最大色深为48bit，采用无损压缩方案存储，而且能够显示带透明度的Alpha通道，如图17-15所示。

由图17-15可以发现，PNG图片的阴影部分在不同的背景色中都能呈现半透明状态。但是目前并不是所有浏览器都支持Alpha透明度的PNG文件，例如，用IE 6.0浏览带Alpha透明的PNG图片，其效果如图17-16所示。

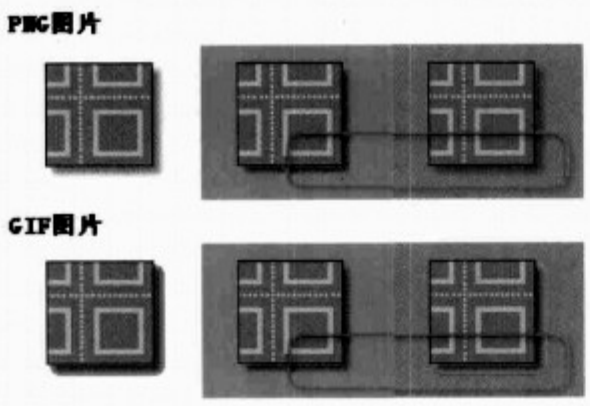


图17-15 Alpha透明度的PNG图片

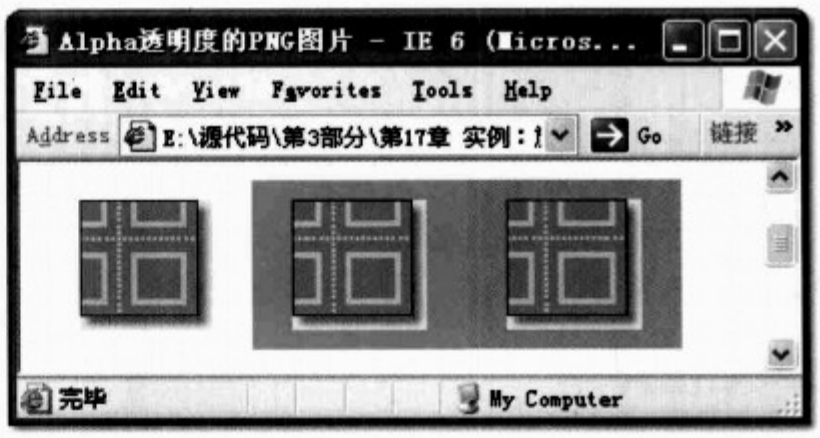


图17-16 用IE 6.0浏览Alpha透明度的PNG图片

17.2.2 图片与优化

网页文件存放在服务器上，访问者使用浏览器来访问这些文件。当一个网页被浏览，服务器就会和访问者的浏览器建立连接，每个连接表示一个并发。

当页面包含很多图片，图片并不是一个一个显示的，服务器会产生出多个连接同时发送文字和图片以提高浏览速度，如果页面中的图片越多那么服务器的并发连接数量就越多。当图片或页面被服务发送后服务器就关闭连接用于和其他请求者建立连接。

在网络速度比较低的时代，一般会把大的图片切分成小图片，也就是同时建立多个连接，从而提高图片的显示速度。而对于背景图片，则尽量使用小图片重复显示。

这样虽然可以提高图片的显示速度，但是对于服务器（或者防火墙）来讲，同一时间内可接受的连接数是有限度的。一个网页内并发连接数越多，服务器在同一时间内可接受的用户数则越低。例如，某个服务器最大可接受的并发连接数为50，而某个网页中的连接数为10，那么，同一时间服务器可以接受的用户数为5。

同时，并发连接数的增大还会影响服务器系统内存的消耗、CPU的负载等很多方面。因此，在网络速度已经大幅提升的今天，已经不推荐上面所说的切分图片的处理方法，而是要降低网页内的并发连接数。

目前比较流行的制作方法之一，就是将不重复且颜色数比较少的背景图片全部放置在一个图片文件内，然后通过设定background-position属性控制图片的位置来显示相应背景。

由于IE 6.0不支持Alpha透明的PNG图片，因此如果背景图片内有需要透明显示的部分，则只能采取GIF格式，所以不适用于色彩丰富的背景图。

但是，减少并发连接数的同时，还要注意服务器的流量问题，在一定程度上，还是要减少图

片的字节数，以节约流量。因此，也不是说把所有图片都放在一个文件内就是最佳的方案，还是要结合实际情况来定。

17.3 CSS美化

结构化完成后，就可以进行CSS美化，在编写CSS之前，要先对设计图进行细节的分析，这样可以对如何实现“表现层”有一个比较清晰的思路，同时还要将美化过程需要的元素的尺寸、背景图片、前景颜色值、链接的样式等都提取出来。有时设计图没有的细节，也要考虑周全。

17.3.1 整体分析

由图17-1可以发现，本例主色调为#3399FF，突出色为#FF9900（大部分链接文字鼠标指向时的颜色），辅助色为#0066CC和#DDDDDD，主要的文字颜色为#666666，如图17-17所示。

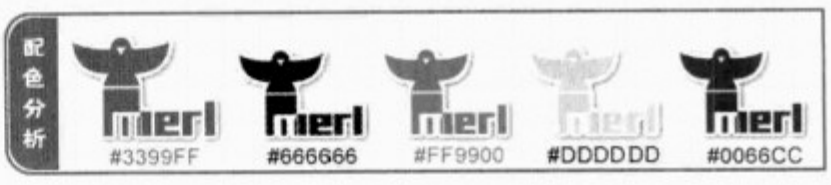


图17-17　配色分析

网页的整体布局是居中的，宽度为980px，同时页面的背景图片是满屏的，这样当显示器的水平方向分辨率大于1024px的时候，也可以有背景图片显示，如图17-18所示。

图17-18　整体设计分析

大型的网站一般在首页使用嵌入式样式，这样可以减少并发连接数，同时也可以避免CSS文件载入出错或者不及时而造成的首页无样式的问题。

在本例中，将仍使用外部样式表：

```
<head>
<title>首页 :: Imerl旅游网</title>
……
<link href="style/index.css" rel="stylesheet"
type="text/css" />
</head>
```

提示：关于嵌入式样式表，请参见本书［3.2.2 嵌入式样式表（embedded style sheets）］一节。在制作过程中，使用Opera 9.2或Firefox 2.0等符合标准的显示器测试页面，然后在IE 7.0内测试，最后再在IE 6.0内测试。

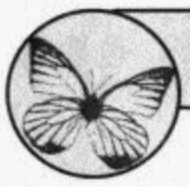
由图17-18可以发现，栏目的标题（<h2>、<h3>等）都包含有色彩丰富且需要背景透明的图标，这些图片需要制作为GIF格式的图片，但是由于图标的色调不同，因此如果放置在同一个GIF文件内，总颜色数限制为256色，则会影响到图片的质量，因此还是需要保存为单独的文件。

mainNav层、hot层、photos层、club层、vote层和community层的背景颜色数较少，且尺寸固定，因此可以保存在1个GIF文件内。而travels层内的文章列表和forumHot层的背景为渐变色，因此也需要单独保存。

1. 通用CSS属性设定

在CSS的开头部分，应该首先利用通配选择器清除掉浏览器的默认边距和补白，然后对<body>设定font属性、前景色和背景色。

```
* {
margin : 0;
padding : 0;
}
body {
font : small/1 "宋体", serif;
background : #fff;
color : #000;
}
```



提示： 关于通配选择器，请参见本书［4.2.1 通配选择器（Universal Selector）］一节。

设计稿内几乎所有的<h2>和<h3>的样式都是一样的，因此可以统一设定如右：

```
h2,
h3 {
font-size: 1em;
color:#039;
}
```

为了防止带链接的图片出现边框，可以设定边框为0，而表单内的<fieldset>元素默认具有边框，这是设计中不需要的，因此也可以设为0，CSS规则如下：

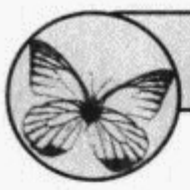
```
img,
fieldset {
border : 0; /* 此规则与 border : none 等效*/
}
```

去掉<li>的列表符号：

```
li {
list-style:none outside; /* 设定outside，防止浏览器仍会在<li>内部保留列表符号的位置 */
}
```

接下来设定链接的通用规则：

```
a {
text-decoration:none;
}
a:link {
color:#666;
}
a:visited {
color:#999;
}
a:hover {
color:#F90;
text-decoration:underline;
}
```



提示： 注意伪类的书写顺序。

当以后的栏目中有不同的设计时，可以利用层叠的原理覆盖掉这里的设定。

2. 整体居中

在本书［8.9.5 应用：元素水平居中］一节内介绍过，通过设定“margin: 0 auto”使页面整体居中，在此，可以在页面所有元素外增加1个 来实现，代码如下：

```
.wrap {
margin : 0 auto;
width : 980px;
}
<body>
  <div class="wrap">
    <div id="header">页头内容</header>
    <div id="main">中部内容</header>
    <div id="footer">页脚内容</header>
  </div>
</body>
```

也可以对header层、main层和footer层分别设定居中。

```
#header,
#main,
#footer {
margin : 0 auto;
width : 980px;
clear:both; /* 3个层都需要独占1行，不与其他内容并列 */
}
```

在本例中将采用第2种方法，这样就不必再添加额外的标签。

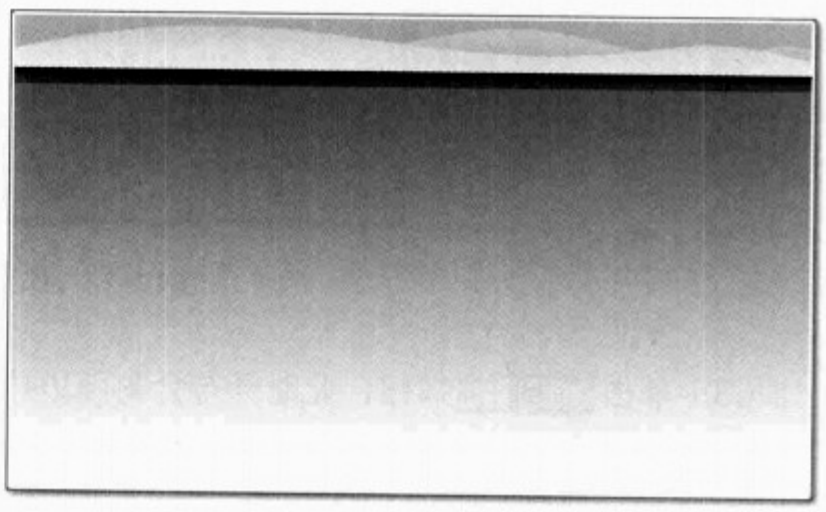


图17-19　<body>的背景

3. body的背景

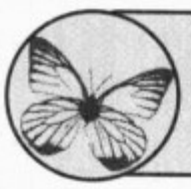
整个背景为1个图片，设定为<body>的背景，如图17-19所示。

由于颜色数不是很多，同时也不需要设置透明通道，因此可以采用PNG格式，其文件字节数为16KB，也在可以接受的范围之内。为<body>添加CSS如下：

```
body {
……
background : #fff url(../img/index_body_ bg.png) repeat-x top center;
}
```

图片的宽度为1500px，对于常见的屏幕分辨率已经足够宽，不过为了防止有更宽的分辨率，因此设定水平方向重复。但是此时背景图片不是无缝拼接，会有明显的接痕，如图17-20所示。

图17-20　背景图片重复显示时的拼接不整齐



提示：执行Opera 9.2的菜单栏【查看】|【缩放】（或按【Alt】+鼠标滚轮）可以等比例缩放网页内容。

因此需要修改背景图片，使其能够无缝拼接。

（1）使用【Fireworks】打开下载文件包内【/第3部分/第17章：旅游网站/设计原稿/body背景.png】文件，如图17-21所示。由图17-21可以发现，只需要对上面进行处理，下面的渐变部分本身就是在水平方向重复显示的。

（2）执行菜单栏【修改】|【画布】|【画布大小】打开【画布大小】对话框，在【新尺寸】的【宽↔】对话框内输入3000，单位为【像素】，【锚定】选择【✣】，即从顶端的中心点向四周扩充，如图17-22所示。

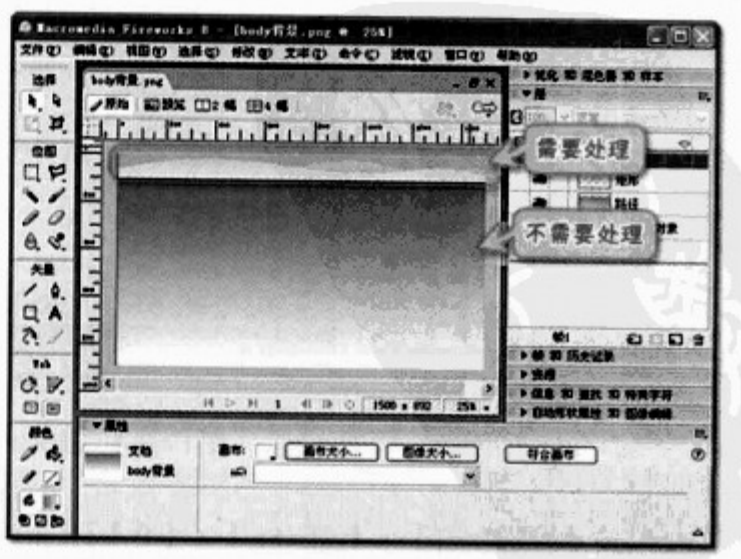


图17-21　使用【Fireworks】打开首页设计稿

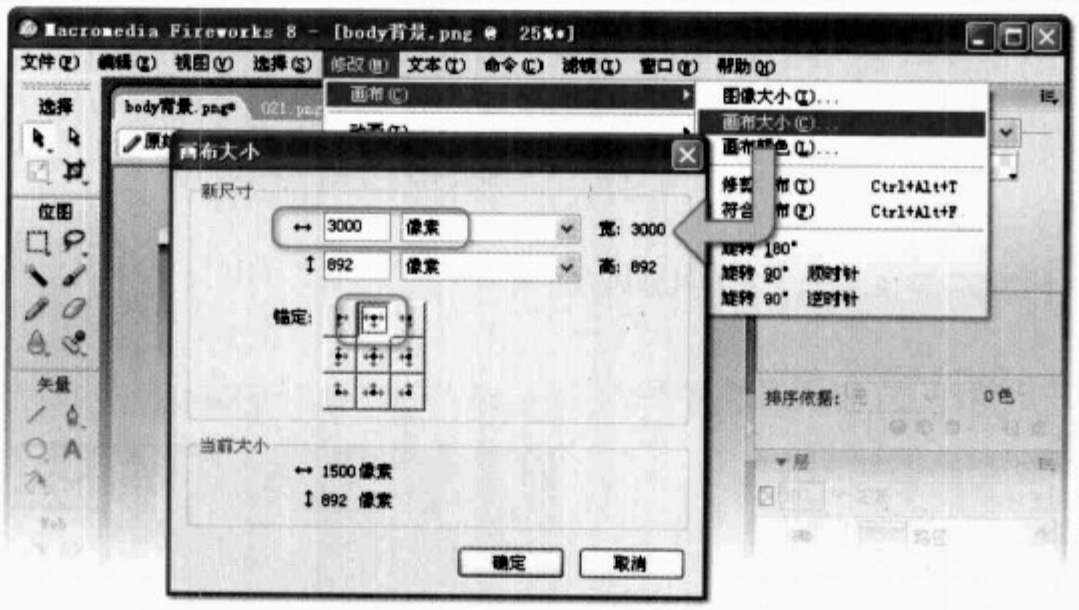


图17-22 扩充画布尺寸

（3）单击 确定 按钮，此时画布会向两边扩充，如图17-23所示。

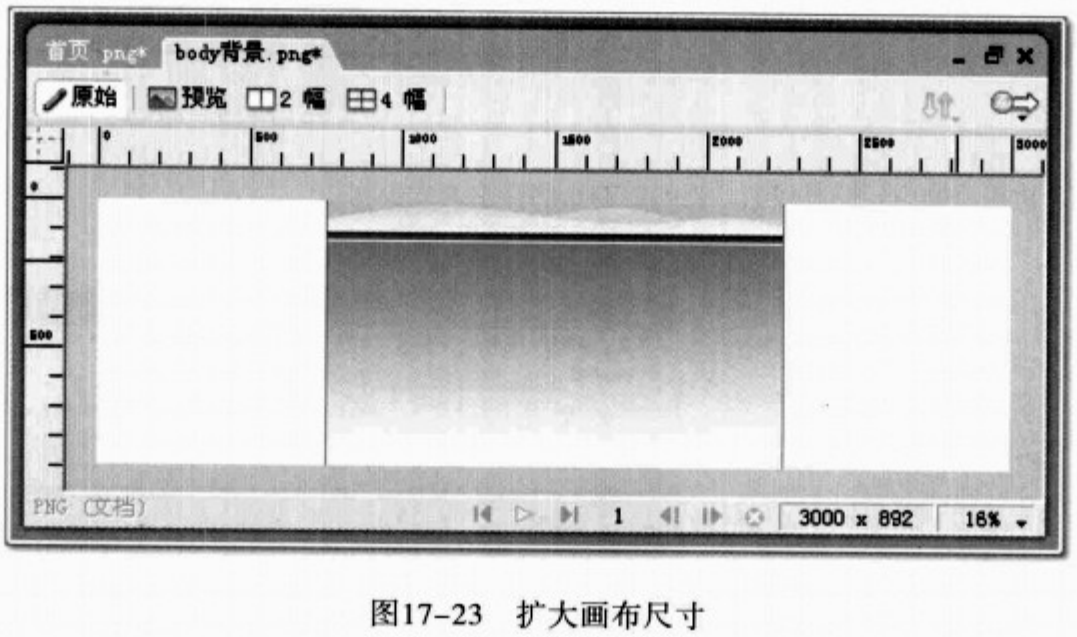


图17-23 扩大画布尺寸

（4）选择头部的曲线对象和背景矩形，如图17-24所示。

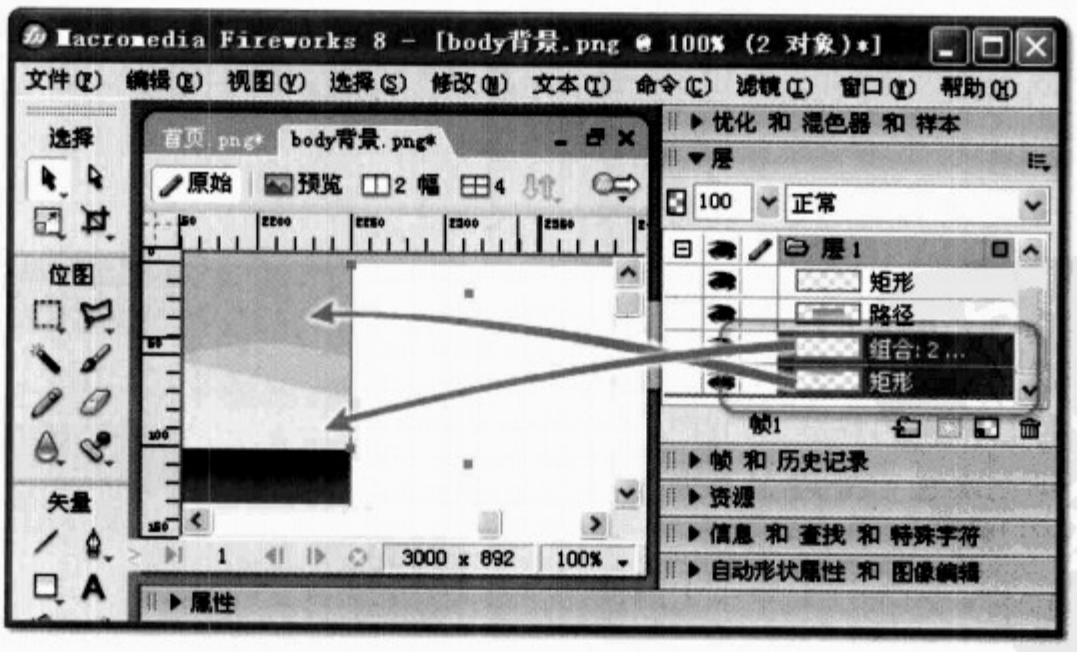


图17-24 选取头部的曲线对象和矩形背景

（5）按【Ctrl】+【Shift】+【D】组合键克隆选中的对象，向左移动克隆的对象至如图17-25所示的位置。

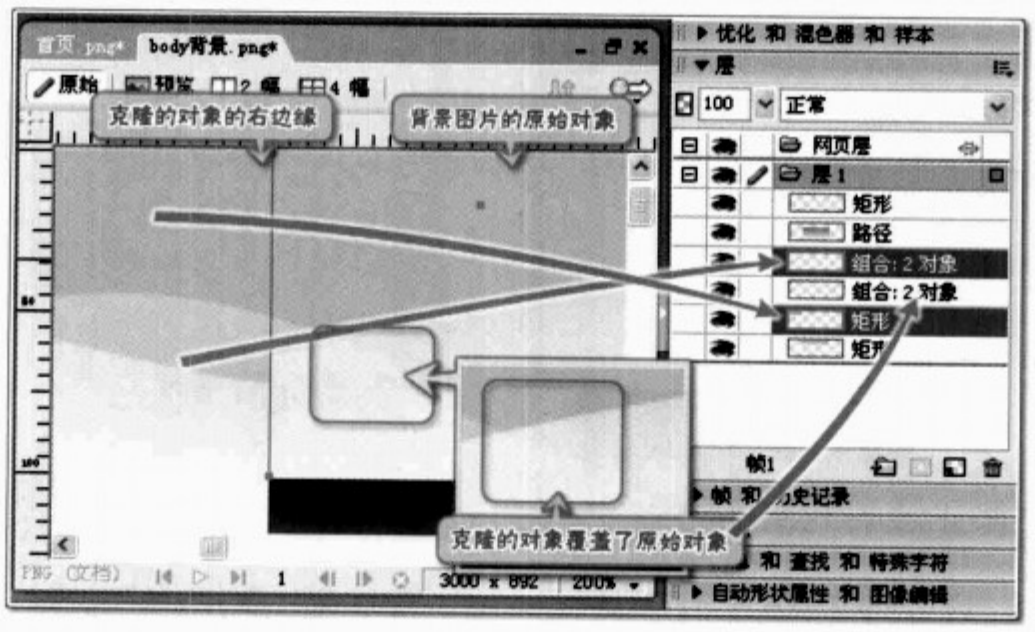


图17-25　克隆原始对象并且移动位置

（6）由图17-25可以发现克隆的曲线对象覆盖了原始对象，因此需要将其置于原始对象之下。在【层】面板内选择克隆的对象，将其移动到原始对象下面，如图17-26所示。

图17-26　将克隆对象移动到原始对象之下

（7）使用【部分选定工具】（快捷键【1】或【A】）选择原始曲线对象左上角的节点，按【↓】键向下移动到如图17-27所示的位置。

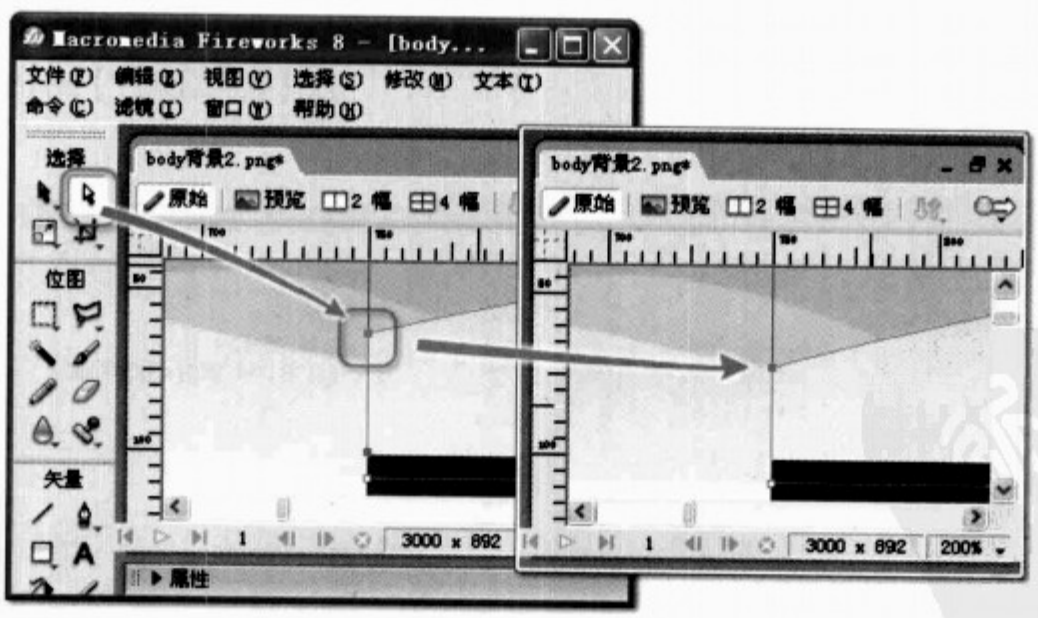


图17-27　选取节点并移动位置

（8）保持节点的选择状态，按住【Alt】键，在节点上单击并按住鼠标左键，向右拖动，此时将有曲柄出现，可以调节曲线，如图17-28所示。

（9）重复步骤（4）和步骤（5），并将克隆的对象移动到原始对象的右边，如图17-29所示。由图17-29可以发现，拼接部分已经能够平滑过渡。

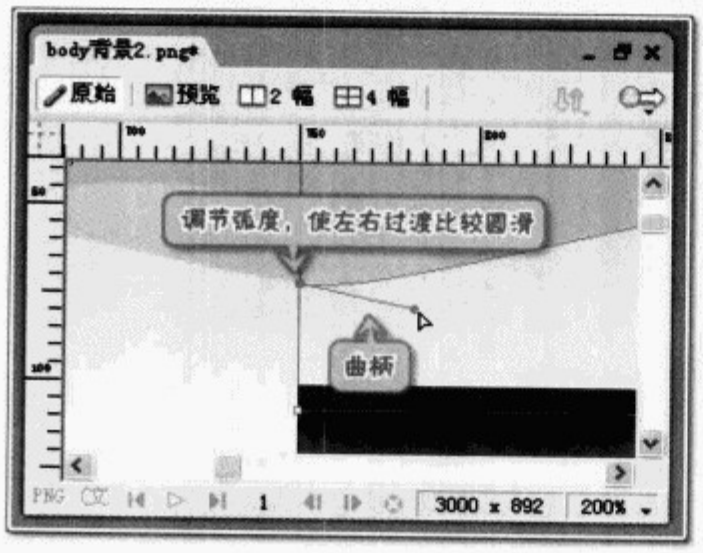


图17-28 调节曲线的弧度使过渡自然

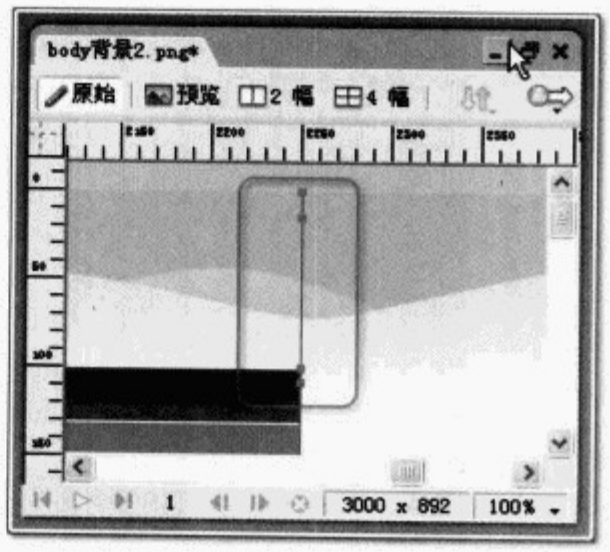


图17-29 再次克隆对象并移动到右边

（10）执行菜单栏中的【修改】|【画布】|【画布大小】命令，打开【画布大小】对话框，在【新尺寸】的【宽 ↔ 】对话框内输入1500，单位为【像素】，【锚定】选择【 ✣ 】，单击 确定 按钮，如图17-30所示。

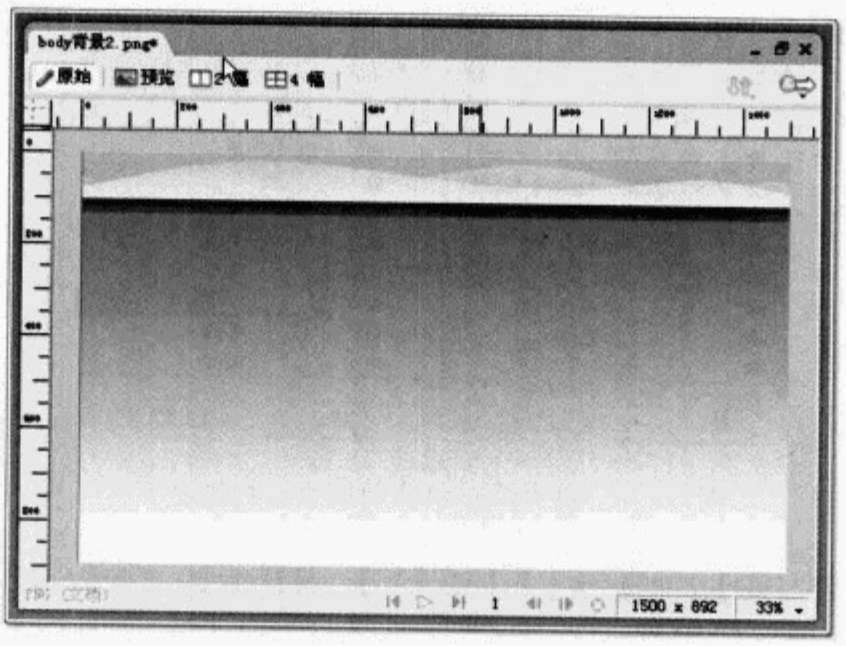


图17-30 处理完毕的背景图片

（11）执行菜单栏中的【文件】|【图像预览】命令，打开【图像预览】对话框，如图17-31所示设定导出选项。

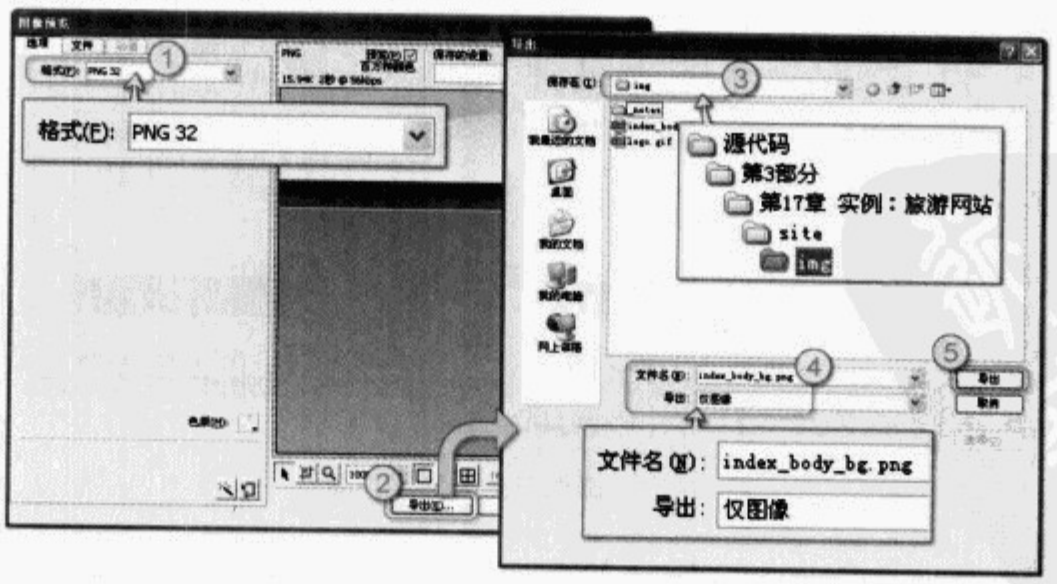
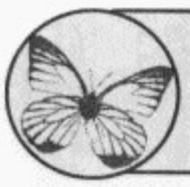


图17-31 导出背景图片

（12）单击 导出(E)... 按钮，打开【导出】对话框，如图17-31所示设定文件夹和文件名称，单击 导出 按钮保存文件。



提示：修改后的背景图片文件，读者也可参见［/第3部分/第17章：旅游网站/设计原稿/body背景2.png］文件。

此时在浏览器内浏览【index.html】文件如图17-32所示。

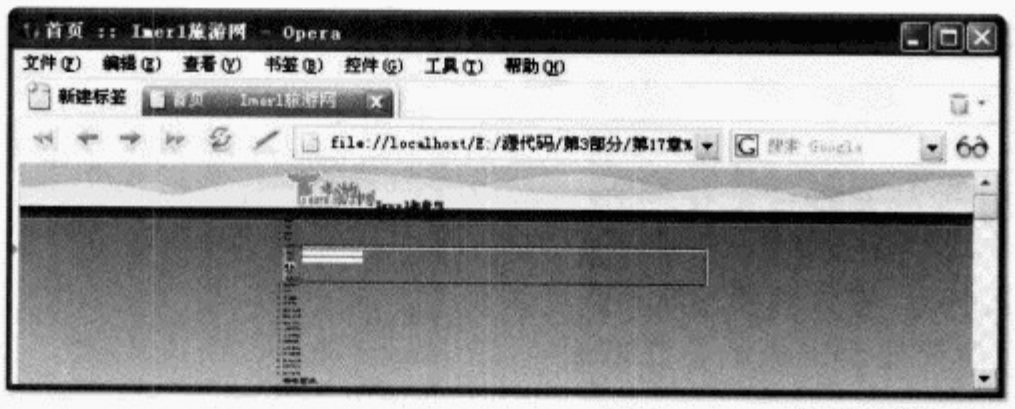


图17-32　在Opera 9.2内缩小网页观察背景图片

17.3.2 header层

header层的内容相对较少，不过用户功能菜单部分比较复杂，涉及到2级栏目的隐藏和显示，如图17-5所示。

1. 拆分尺寸和图片

header层内各部分内容的尺寸如图17-33所示。

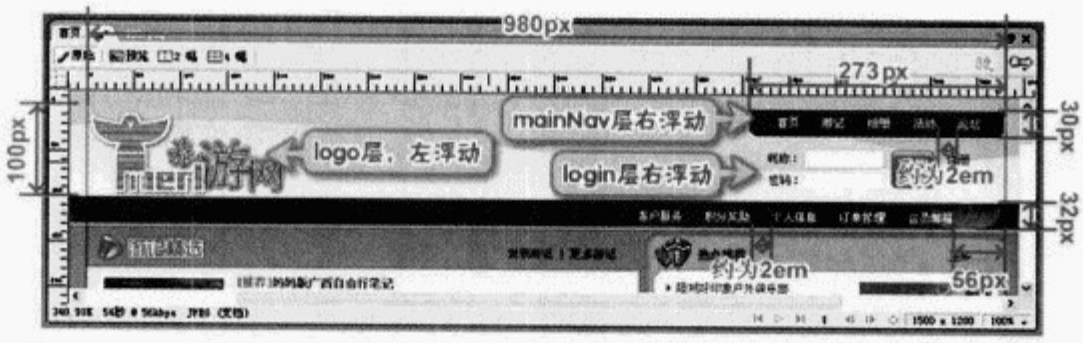
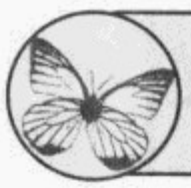


图17-33　header层细节分析

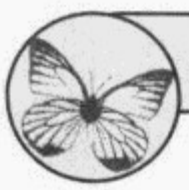


提示：读者可用Fireworks软件打开下载文件包内【/第3部分/第17章：旅游网站/设计原稿/首页.png】文件，利用软件的标尺和辅助线等工具来测量尺寸。

由于<body>背景图高度是一定的，因此页头部各层的高度和宽度必须是固定的。

2. header层CSS

header层在［17.3.1　整体分析］一节内已经设定了宽度和居中，同时设定了body的字号为"small"，而对于header层内的文字大小为12px，对于大部分浏览器，默认的字号为16px，CSS 2.1归定的关键字所放因子为1.2，因此small=16px/1.2≈13px，0.9em=small*0.9≈12px。



提示：关于字体尺寸的关键字，请参见本书［6.2.2 绝对尺寸］一节。

在此需增加对字号的设定，同时，header层的总高度为132px：

```
#header {
font-size: 0.9em; /* header层内的子元素会继承这个值 */
height : 132px;
}
```

17.3.3 logo层

logo层的XHTML如下：

```
<div id="logo">
   <h1><a href="/index.html" title="前往网站首页"><img src="img/logo.gif" alt="网站Logo" width="209"
height="100" />Imerl旅游网</a></h1>
</div>
```

logo层高度为100px，整体需要左浮动：

```
#logo {
float : left;
height : 100px;
overflow : hidden; /* 如果内容超高，会隐藏 */
}
```

<h1>元素是不需要显示出来的，在前面的章节介绍过几种隐藏内容的方法：

- 设定“text-indent : -9999em”；
- 设定“display : none”；
- 设定“visibility : hidden”。

此处需注意的是，设定“display : none”的元素的内容可能会被搜索引擎忽略，而如果设定“text-indent : -9999em”则图片也会被隐藏，因此在此选择使用设定“visibility : hidden”的方法。不过，由于图片和文字混在一起，因此需要修改XHTML代码，将文字用标签嵌套起来：

```
<h1><a href="/index.html" title="前往网站首页"><img src="img/logo.gif" alt="网站Logo" width="209"
height="100" /><span>Imerl旅游网</span></a></h1>
```

增加CSS如下：

```
#logo h1 span {
visibility : hidden;
}
```

此时文档在浏览器内显示如图17-34所示。

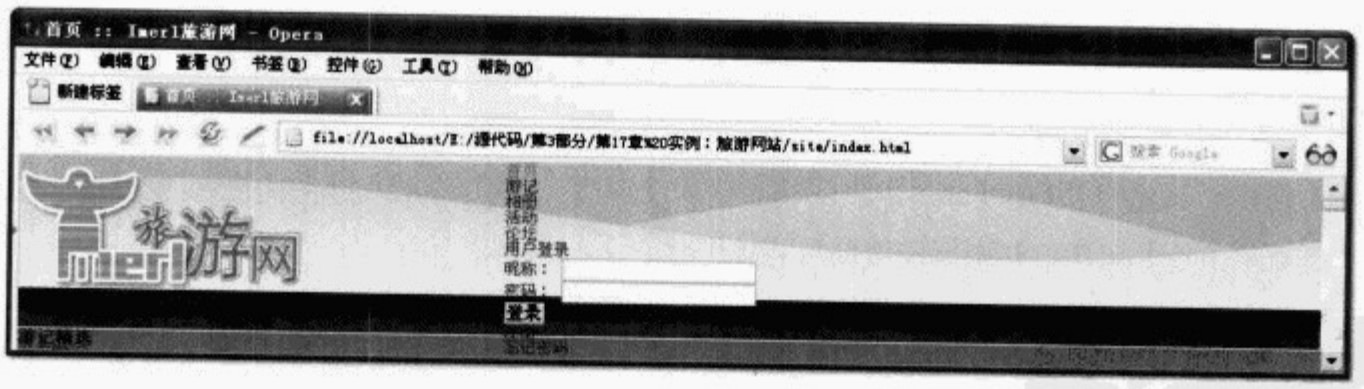


图17-34 logo层在浏览器内的显示

17.3.4 mainNav层

mainNav层的XHTML代码如下：

```
<div id="mainNav">
 <ul>
  <li><a href="/index.html" title="网站首页">首页</a></li>
  ……
 </ul>
</div>
```

由图17-33可以发现，mainNav层需要右浮动，宽度为273px，高度为30px，同时，mainNav层需要设定背景图片，背景图片需要设置透明背景，如图17-35所示。层内的列表项<li>需要横排显示。

图17-35 mainNav层背景的细节

1. 制作背景图片

在［17.2.2 图片与优化］一节内介绍过，可以将不重复的背景图片放在一个文件内，利用background-position属性控制图片的位置，mainNav层的背景图片即可如此处理。

（1）在【Fireworks】内执行菜单栏中的【文件】|【新建】命令（或者按【Ctrl】+【N】组合键）打开【新建文档】对话框，如图17-36所示建立新文档。

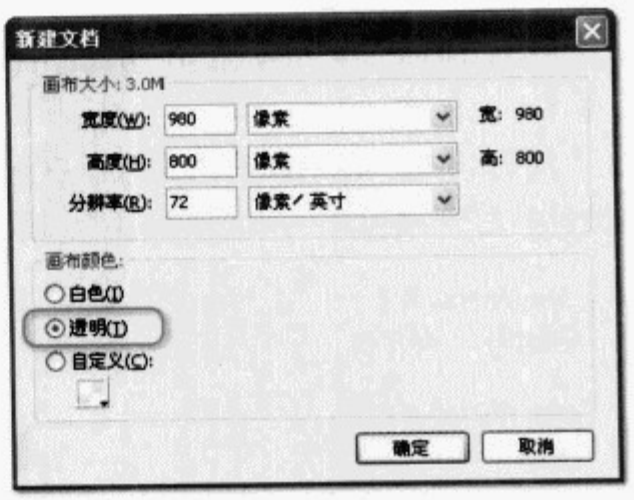
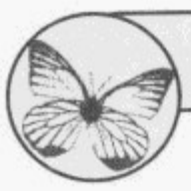


图17-36 建立新文档



提示：文件的大小只是估计的尺寸，可以随时调整。

（2）选择【首页.png】内mainNav层的背景对象，将其复制到新建文档内，如图17-37所示。

（3）按【Ctrl】+【S】组合键打开【另存为】对话框，将文件保存到设计原稿的文件夹内，文件名为【背景.png】，以备以后修改。

（4）在【优化】面板（快捷键【F6】）内如图17-38所示设定优化选项。

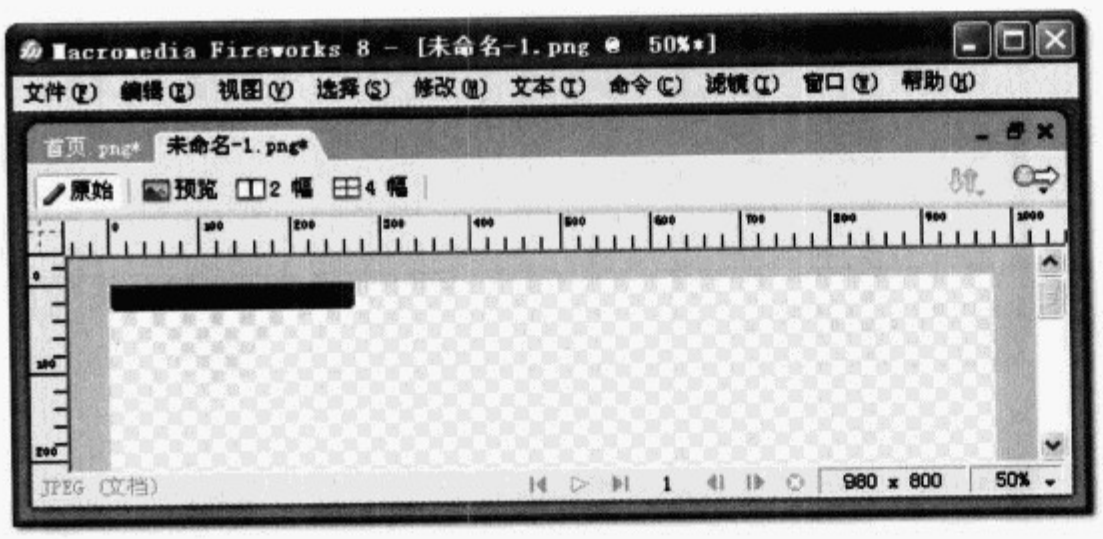


图17-37 将mainNav层的背景复制到新建文档内

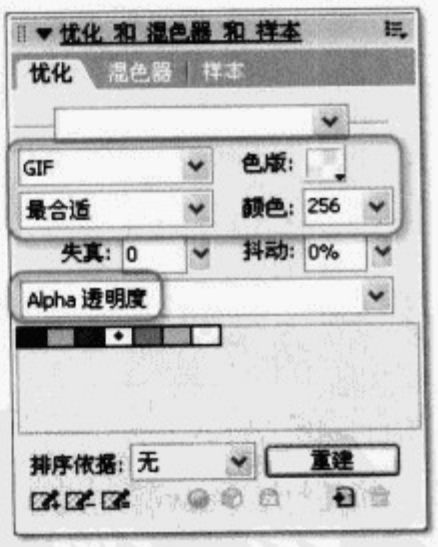
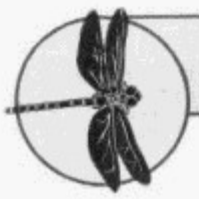


图17-38 优化设定

（5）执行菜单栏中的【文件】|【导出】命令（或者按【Ctrl】+【Shift】+【R】组合键），将图片导出到与【index_body_bg.png】文件相同的文件夹内，文件名称为【bg_01.gif】。



注意：以后每次修改这个背景文件，都需要重新导出GIF图片。

2. 设定CSS

设定maniNav层的CSS如下：

```
#mainNav {
float : right;
width : 273px;
height : 30px;
text-align : center; /* 文字水平居中 */
line-height : 30px; /* 使文字垂直居中 */
overflow : hidden;
background : url(../img/bg_01.gif) no-repeat; /* 不设定偏移量，则默认为0 0 */
}
```

同时，为了使<li>的内容可以在一行内显示，需要改变<li>的display属性为“inline”，此时mainNav层在浏览器内显示如图17-39所示。

```
#mainNav li {
display : inline;
}
#mainNav a {
color : #fff;
}
#mainNav a:hover {
color : #F90;
}
```

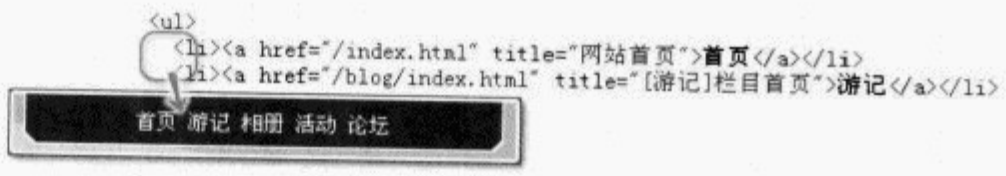


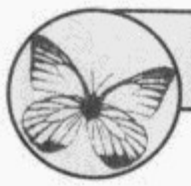
图17-39 元素设定“display : inline”后空格会显示出来

由图17-38可以发现，由于设定了<li>为行内元素，其标签之间的空格则会显示出来，删除<li>标签之间的回行和空格可以消除这个问题：

```
<div id="mainNav">
  <ul><li><a href="/index.html" title="网站首页">首页</a></li><li><a href="/blog/index.html" title="[游记]栏目首页">游记</a></li><li><a href="/photos/index.html" title="[相册]栏目首页">相册</a></li><li><a href="/activities/index.html" title="[活动]栏目首页">活动</a></li><li><a href="/forum/index.html" title="[论坛]首页">论坛</a></li></ul>
</div>
```

为<li>添加左右补白以增加链接文字间的距离：

```
#mainNav li {
display : inline;
padding : 0 1em;
}
```



提示： 如果不删除空格，则可以将补白值缩小。

此时mainNav层的显示如图17-40所示。

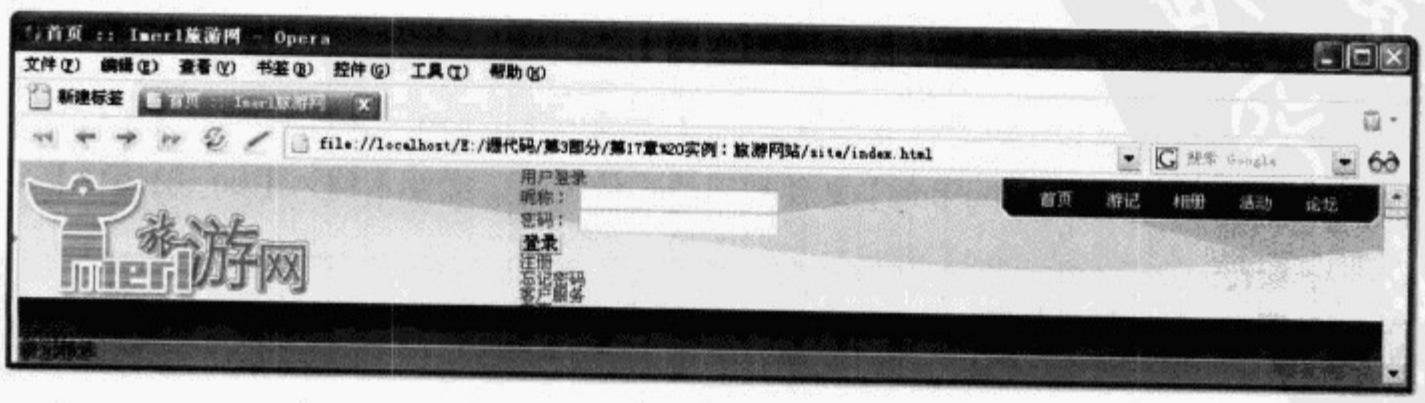
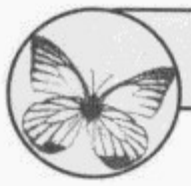


图17-40 设定好CSS后mainNav层的显示效果

17.3.5 login层

login层内包含的是登录表单，由于部分浏览器对表单元素的CSS支持不好，因此表单的排版布局也是比较难掌握的。login层的XHTML代码如下：

```
<div id="login">
 <form id="formLogin" action="#">
  <fieldset>
  <legend>用户登录</legend>
  <ul class="loginInfo">……</ul>
  <ul class="reg">……</ul>
  </fieldset>
 </form>
</div>
```



提示：某些时候，也可以考虑使用表格来布局表单元素，要注意灵活应用。

首先设定login层右浮动：

```
#login {
width : 273px;
float : right;
}
```

此时login层在浏览器内位置如图17-41所示。

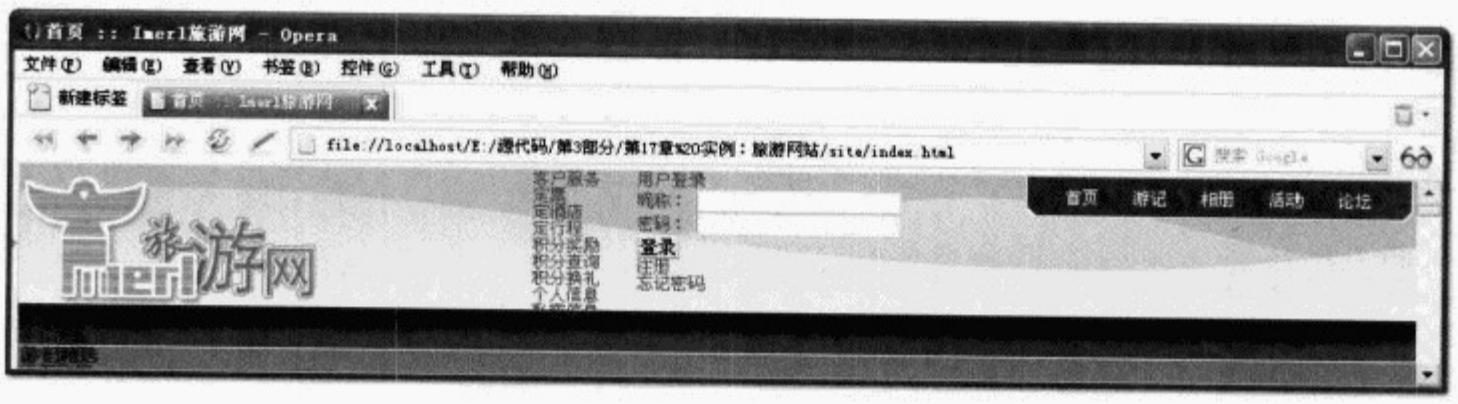
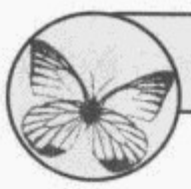


图17-41　login层浮动后在浏览器内的显示

由图17-41可以发现，mainNav层浮动在login层的右侧，因为mainNav层的文档位置在login层之前，因此它向右浮动到其右边接触包含块的右边缘，而login层向右浮动，到右边接触到右浮动的mainNav层的左边。而设计稿中，login层右边没有右浮动的元素，因此需要为login层增加clear属性清除右浮动元素，如右边代码：

```
#login {
……
clear : right;
}
```



提示：关于float和clear属性，读者可参阅本书［9.4 浮动与清除］一节。

此时显示如图17-42所示。

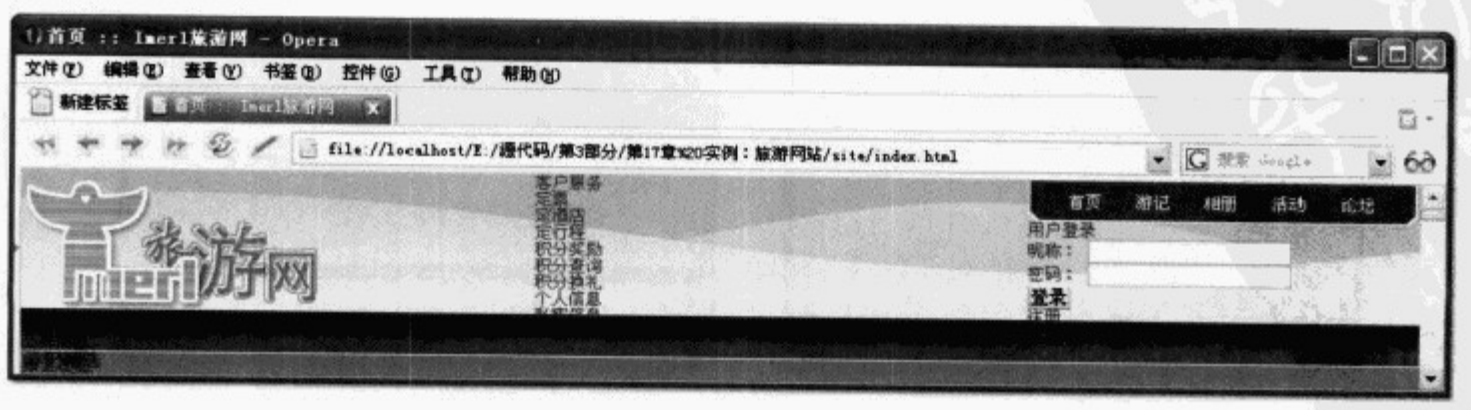


图17-42　清除右浮动后login层的位置显示

<legend>元素是不需要显示的，因此可以设定display属性将其隐藏，如右边代码：

```
#login legend {
display : none;
}
```

表单由2个<ul>构成“loginInfo”和“reg”，这两个<ul>元素需要左浮动以使其在一行内显示，登录按钮需要绝对定位到设计的位置，因此“loginInfo”需要相对定位以为按钮创建包含块。各元素尺寸细节如图17-43所示。

图17-43 login层细节分析

```
#login ul {
float : left;
color : #039;
display : inline; /* 消除IE 6.0内浮动双边距的Bug */
}
#login li {
line-height : 2; /* 行高为估计值，可调整 */
}
#login a:link {
color : #039;
}
#login .loginInfo {
width : 192px;
margin-left : 20px;
position : relative; /* 生成包含块*/
}
#login input {
font-size : 1em; /* 某些浏览器不同type属性的<input>元素默认字体大小可能不一样 */
width : 81px;
height : 17px;
border : 0;
}
```

此时在Opera 9.2内login层显示如图17-44所示，而在IE 6.0内由于含有替换元素的行高失效问题，其显示如图17-45所示。

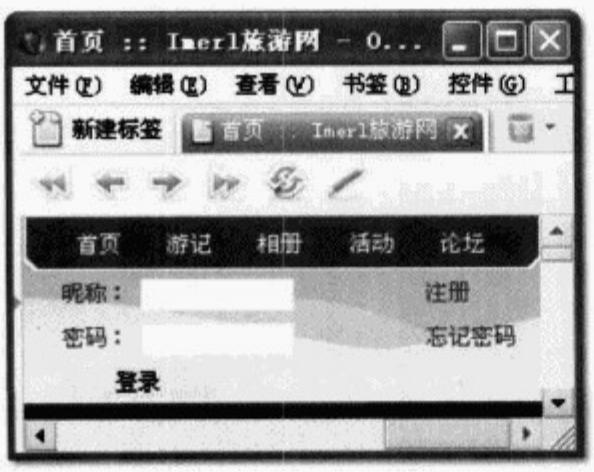


图17-44 login层在Opera 9.2内的显示

图17-45 login层在IE 6.0内的显示

因此需要运用Hack针对IE 6.0设定<li>元素的上下补白来解决垂直方向的距离：

```
#login .loginInfo li{
_padding : 3px 0; /* _只有IE 6.0识别，(12px *2em - 17px)/2 ≈ 3px */
}
```

同时，由图17-44可以发现，内容整体偏上，因此可以为login层添加上补白：

```
#login {
……
padding-top : 10px;
}
```

图17-46　调整后IE 6.0内login层的显示

此时在浏览器内显示如图17-46所示。对于“登录”按钮，有以下几种方法可以实现。

● **第一种方法，为按钮设定背景图片。**

具体的XHTML如下：

```
<input type="submit" value="登录" id="btnLoginSubmit" title="单击登录" />
```

背景图片也属于颜色较少且不重复的图片，因此可以放置在大背景图片内。

（1）使用Fireworks打开下载文件包内【/第3部分/第17章：旅游网站/设计原稿/首页.png】文件和前面操作中保存的【背景.png】文件。

（2）将从【首页.png】中将登录按钮复制到【背景.png】文件内，并调整对象位置，如图17-47所示。图17-47中“X/Y”的坐标值，为背景图片设定CSS定位时的偏移量。

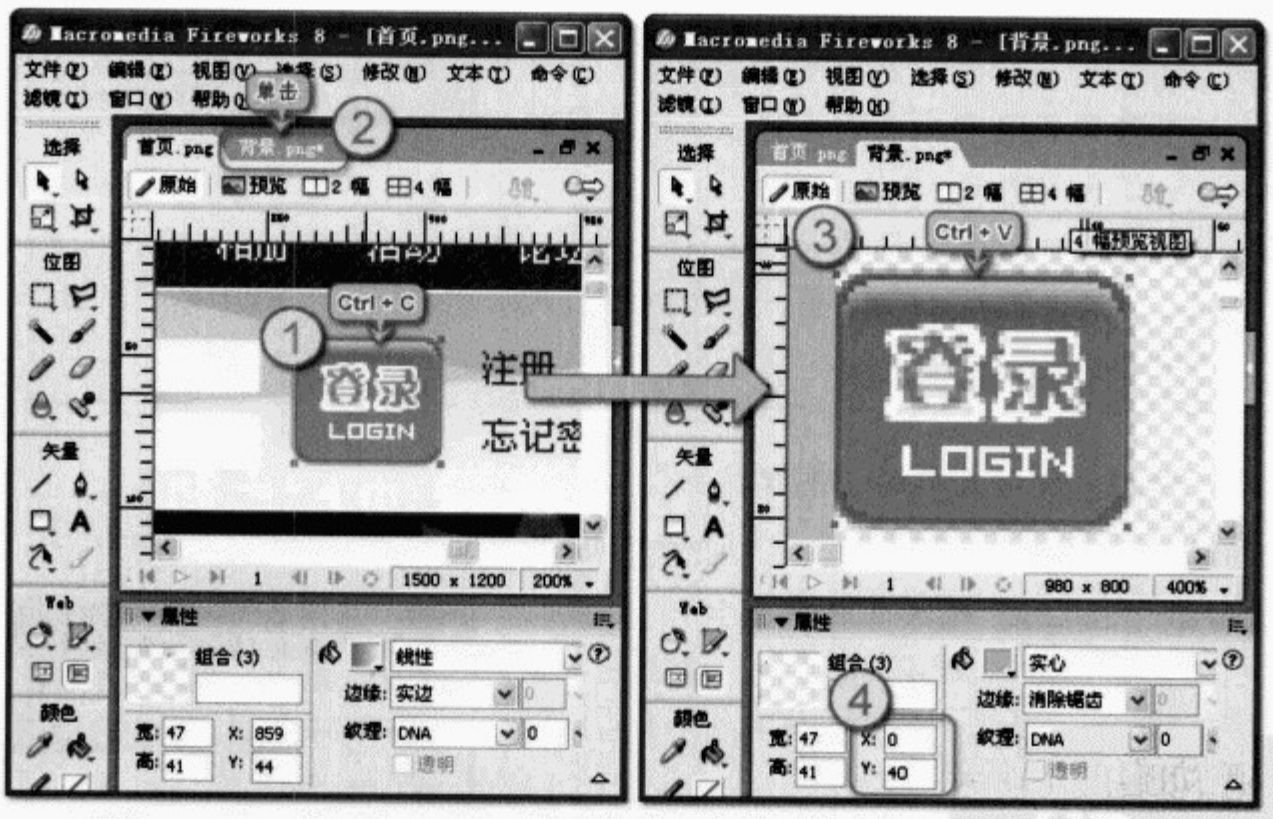


图17-47　将登录按钮复制到背景文件中

（3）按【Ctrl】+【Shift】+【R】组合键重新导出背景图片文件。按钮的CSS设定如下：

```
#login #btnLoginSubmit {
width : 47px; /* 见图17-47中对象的宽和高 */
height : 41px;
position : absolute;
right : 12px; /* 据包含块（<ul>）的右边缘12px，如图17-43所示。*/
top : 3px;
background : url(../img/bg_01.gif) no-repeat 0 -40px; /* -40px表示图片整体向上偏移40px */
}
```

按钮的背景图片部分在整个图片内的坐标为（0，40），因此，如果要让其显示出来，就需要将图片整体向上移动40px，如图17-48所示。此时在浏览器内的显示如图17-49所示。

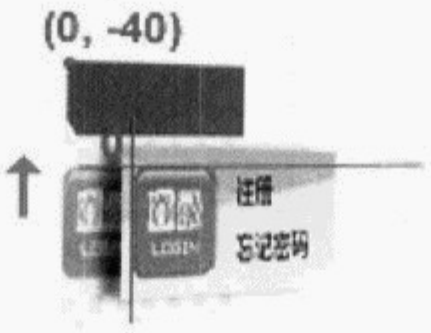


图17-48 背景图片定位示意

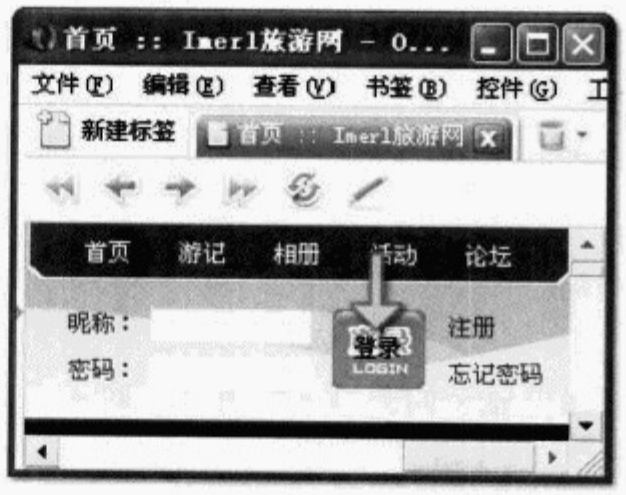


图17-49 登录按钮在浏览器内的显示

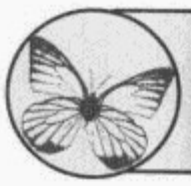
<input>元素的value属性值“登录”显示在背景之上，因此需要将其隐藏：

```
#login #btnLoginSubmit {
……
text-indent : -9999em;
}
```

但是由于Opera不支持按钮文字的缩进，因此此方法在Opera内无效。而其他浏览器则可以实现设计效果，如图17-50所示。

由于Opera的效果无法完成，因此可以采取以下的方法来实现。

图17-50 按钮设定完成



提示：不建议删除value属性值，这样做在无样式或者不显示图片的时候，会显示一个没有文字的空按钮。

● 第二种方法，使用表单的“图像按钮”。

```
<input type="image" src="img/btn_login.gif" name="btnLoginSubmit" id="btnLogin - Submit" />
```

type属性的“image”值表示使用图片作为按钮，src指向图片的URI。其效果类似于插入了一个图片，单击图片可以提交表单，因此不需要设定背景，CSS规则如下，其显示如图17-51所示。

```
#login #btnLoginSubmit {
width : 47px;
height : 41px;
position : absolute;
right : 12px;
top : 3px;
}
```

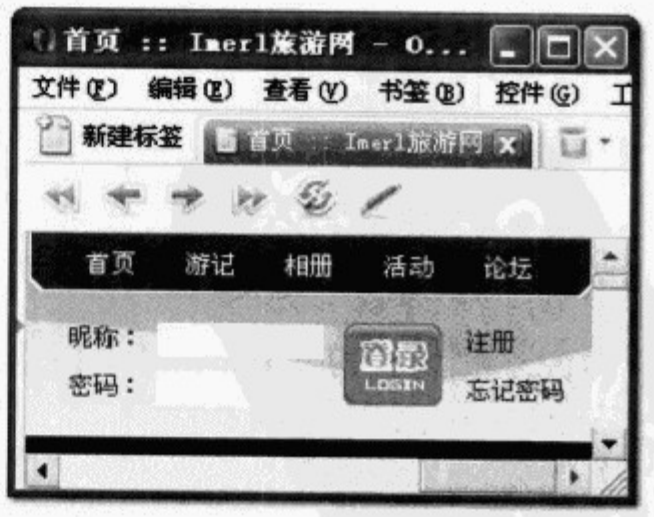


图17-51 图像按钮的显示

这样制作的问题在于，当访问者选择不显示图片（或者图片由于种种原因不能正常显示）的时候，浏览器的显示可能如图17-52所示。

这无疑会使访问者迷惑，不知道这个图像是一个按钮。

图17-52　屏蔽图片后图像按钮的显示

● 第三种方法，改用<button>标签。

<button>标签定义一个按钮。在<button>内部可以放置内容，比如文本或图像。这是该元素与使用<input>创建的按钮之间的不同之处。<button>与<input type="button">相比，提供了更为强大的功能和更丰富的内容。<button> 与 </button> 标签之间的所有内容都是按钮的内容，其中包括任何可接受的正文内容。

因此可以修改XHTML如下：

```
<button type="submit" id="btnLoginSubmit">登录</button>
```


提示： 关于<button>标签更多内容读者可参见http://www.w3school.com.cn/tags/tag_button.asp（中文）。

可以将第一种方法内的CSS完全应用在<button>元素上：

```
#login #btnLoginSubmit {
……
background : url(../img/bg_01.gif) no-repeat 0 -40px;
text-indent : -9999em;
border : 0; /* 去掉<button>默认的边框*/
cursor : pointer; /* 鼠标指向按钮的时候显示可点击的手型光标 */
}
```

由此可见，第3种方法是比较好的解决方案。

图17-53　<button>元素的显示

17.3.6 controlMenu层

controlMenu层在logo层和login层之下，其内容包含共下拉菜单，如图17-5所示。controlMenu层的XHTML如图17-54所示。

图17-54 controlMenu层的XHTML代码示意

controlMenu层的细节分析如图17-55和图17-56所示。

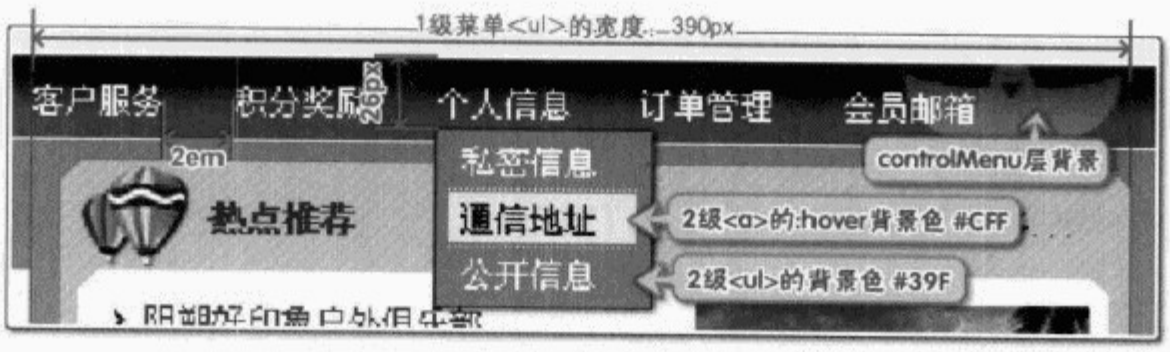


图17-55 controlMenu层细节分析

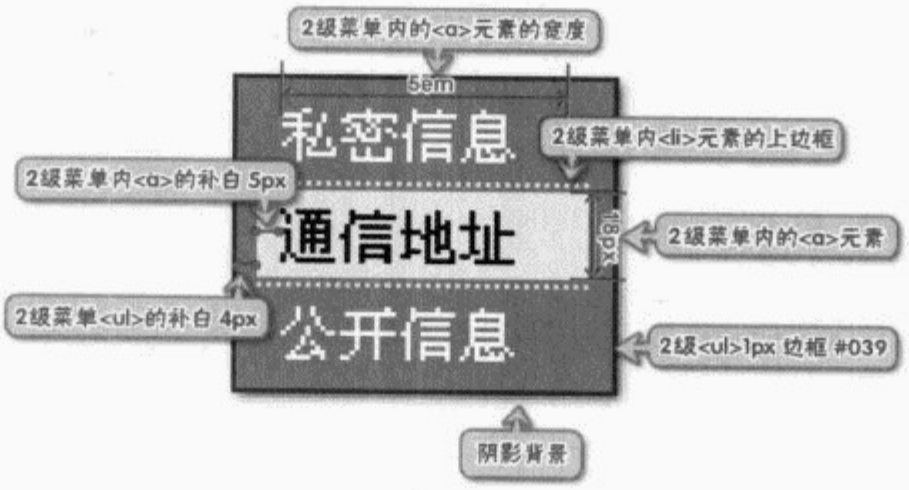


图17-56 controlMenu层内2级菜单的细节分析

1. controlMenu层的背景图片

controlMenu层背景图片为不重复的背景，可以放在大背景图片中。同时需要注意，controlMenu层的背景是叠加在<body>背景之上的，如图17-57所示。

（1）使用Fireworks打开下载文件包内【/第3部分/第17章：旅游网站/设计原稿/首页.png】文件和【背景.png】文件。

（2）如图17-58所示选择【首页.png】中的背景对象。

（3）按【Ctrl】+【C】组合键复制选择的对象，再按【Ctrl】+【N】组合键新建文档。

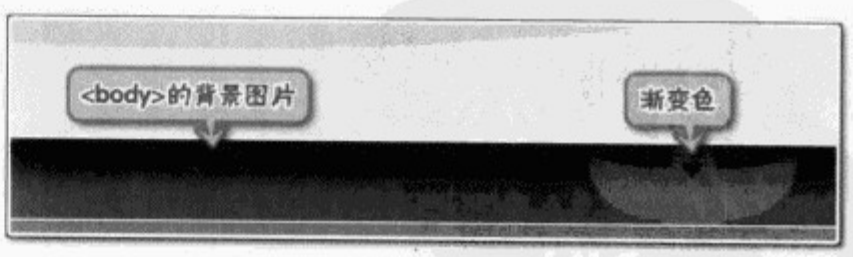


图17-57 1级<ul>背景分析

图17-58 选取背景对象

（4）在新建文档内按【Ctrl】+【V】，粘贴复制的对象，按【Ctrl】+【D】取消选取状态。

（5）选取合成的路径，如图17-59所示，按【Ctrl】+【C】组合键复制对象。

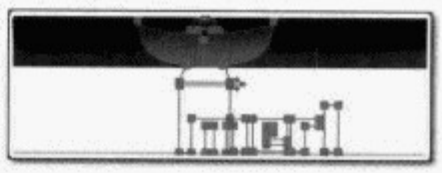


图17-59　选取合成的路径

（6）再次选取2个对象，按【Ctrl】+【G】组合键组合对象。

（7）执行菜单栏中的【编辑】|【粘贴为蒙版】命令，此时对象如图17-60所示。

（8）在合成对象上单击鼠标右键，在弹出的菜单内选择【平面化所选】命令，将路径转化为位图。

（9）复制此位图，并粘贴到【背景.png】文件内，如图17-61所示设定位图的偏移量。

图17-60　制作<ul>的背景图片

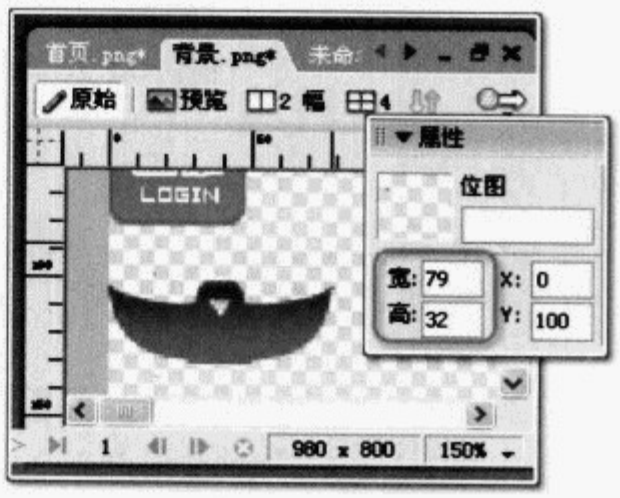
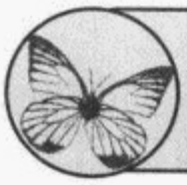


图17-61　偏移量

背景图片的宽度为79px，Y为100px，controlMenu层的宽度为980px，因此背景图片水平方向的偏移量为980px-79px=901px；垂直偏移量为-100px。

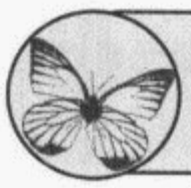


提示：不能设定偏移量为“right”，因为“right”为整个大背景图片的右边对齐。新建的文档不需要保存。

2. controlMenu层定位

controlMenu层不浮动，同时也不允许两旁出现浮动元素：

```
#controlMenu {
clear:both;
height:32px;
line-height:32px;
background:url(../img/bg_01.gif) no-repeat 901px -100px;
}
#controlMenu a:link{
color:#fff;
}
#controlMenu a:hover {
text-decoration:none;
}
```



提示：常规流向的元素宽度为“auto”时，会自动占满父元素的内容宽度，因此不需要为controlMenu层定义widht属性。

此时在浏览器内显示如图17-62所示。

由图17-62可以发现，在IE内controlMenu层位置偏下，这是由于login层内的“登录”按钮

绝对定位，而按钮的父元素<li>内没有常规流向的内容，因此应没有高度。而在IE内<li>仍保留了高度，这是IE对空格处理的问题，由图17-62读者可以发现，<li>内有空格存在，IE内login层的高度比其他浏览器内高。因此需要调整login层的CSS如下：

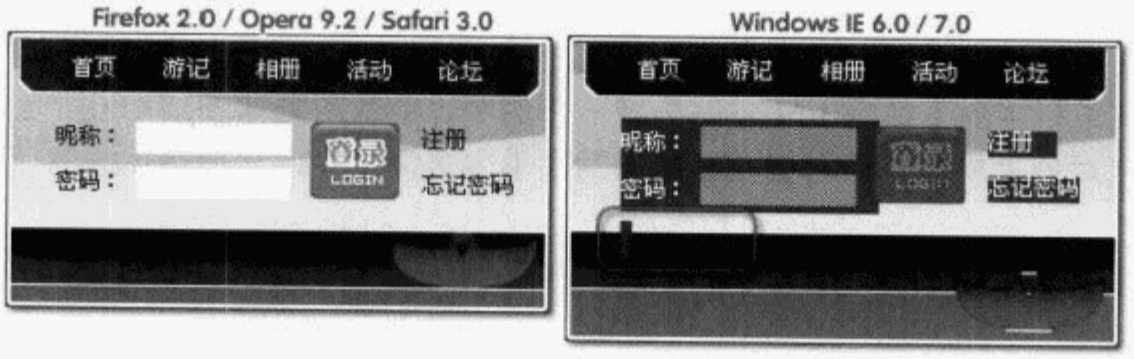


图17-62 controlMenu层在浏览器内的显示

```
#login {
……
padding-top : 10px;
height : 60px; /* mainNav层和login层总高100px ®C mainNav层高30px ®C 上补白10px */
overflow : hidden; /* 隐藏溢出的内容 */
}
```

此时在IE内controlMenu层位置恢复正常。

3. 下拉菜单背景实现原理

2级菜单的显示原理，与［9.7.2 应用：显示及隐藏元素］所介绍的原理相同，利用visibility属性与:hover伪类实现。但是由于IE 6.0不支持除<a>以外的元素的:hover伪类因此需要针对IE 6.0进行特别处理。同时2级菜单有透明的阴影背景，可以使用透明PNG图片实现，但是对于IE 6.0也需要另行处理。

由于2级<ul>还有背景颜色和边框，因此有两种方法实现阴影背景。

● 制作固定大小的背景图片。

将<ul>的阴影背景和边框做成1个没有上边框的图片，而上边框由<ul>的第一个<li>子元素来实现，其原理如图17-63所示。

其原理同［10.3.2 模拟边框］一节内介绍的宽度固定高度不定的圆角边框的实现原理相同，只是第一个<li>的CSS设定复杂一些。

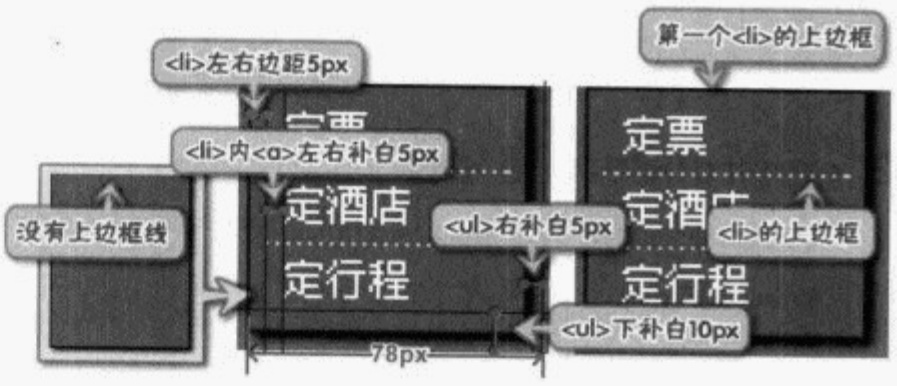
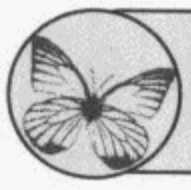


图17-63 背景实现原理



提示：为了节约篇幅，在此省略CSS设定过程，读者可参见下载文件包内［/第3部分/第17章：旅游网站/site/ controlMenu_drop.html］文件。

● 不固定宽度的阴影背景。

由于上面所介绍的方法背景图片宽度固定，灵活性较差，因此还可以采取另一种方法来实现阴影背景：即2级菜单的<ul>元素设定背景色和边框，另外需要增加1个元素来包裹<ul>元素以实现阴影效果。

```
<div id="controlMenu">
 <ul>
  <li><a href="/service/index.html" title="客户服务项目查询">客户服务</a>
    <div>
```

```
      <ul>……</ul>
    </div>
  </li>
  <li><a href="/member/score/" title="积分奖励内容查询">积分奖励</a>
    <div>
      <ul>……</ul>
    </div>
  </li>
  ……
 </ul>
</div>
```

阴影图片的制作方法如图17-64所示，读者可参见下载文件包内［/第3部分/第17章：旅游网站/设计原稿/阴影.png］文件。将文件导出到网站图片文件夹内，命名为【shadow.png】。

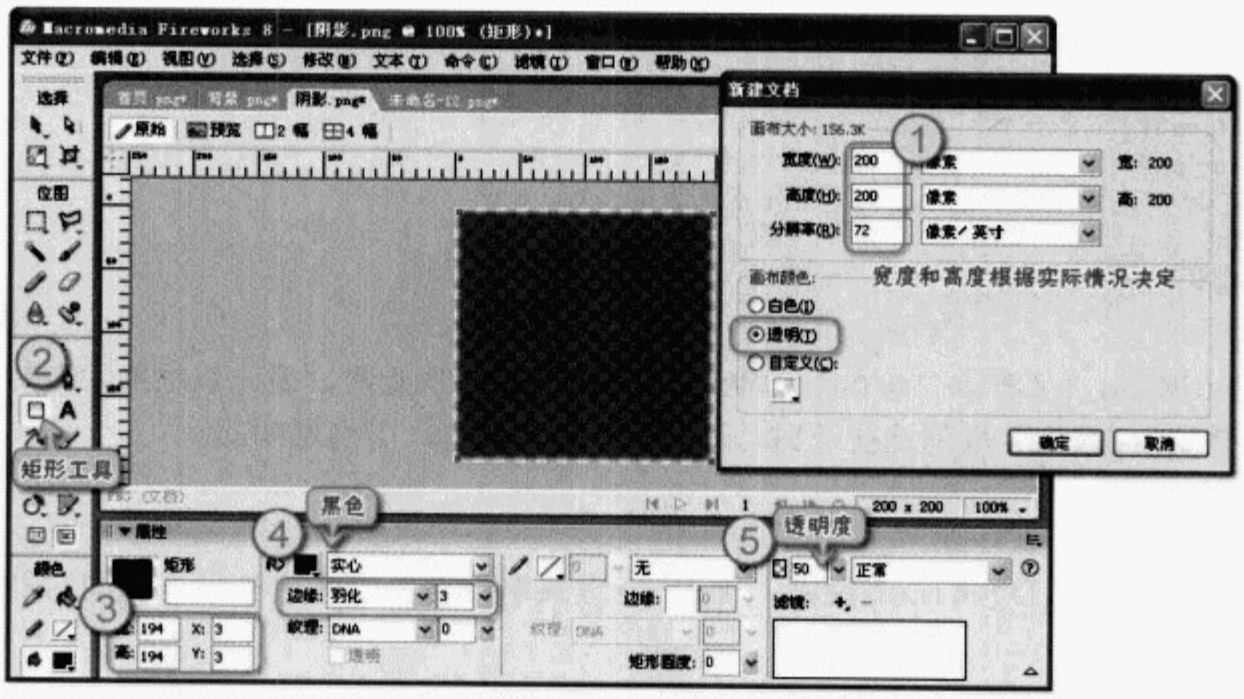


图17-64　阴影图片的制作

4. 1级菜单的定位

由于修改了XHTML代码，因此controlMenu层内文档结构树如图17-65所示。

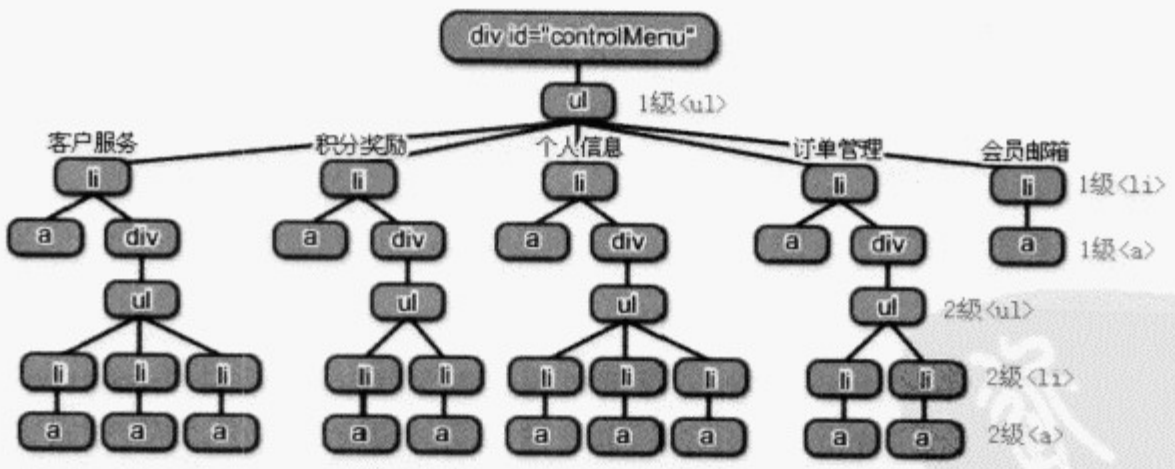


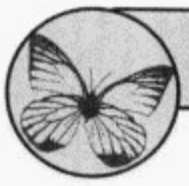
图17-65　controlMenu层文档结构树

1级菜单<ul>在层内居右，因此需要将其右浮动，1级的<li>需要横排显示，同时由于其包含2级菜单，因此不能设定“display:inline”，而需要将其左浮动。

对于支持CSS 2.1子元素选择器的浏览器，应如下定义CSS：

```
#controlMenu > ul {
width : 390px;
float : right;
}
```

```
#controlMenu > ul > li {
position : relative; /* 为下拉菜单生成包含块 */
float : left;
margin-right : 2em;
}
```



提示：关于子元素选择器，请参见本书［4.2.6 子元素选择器（Child Selectors）］一节。

但是由于IE 6.0不支持子元素选择器，因此需要如下定义CSS：

```
#controlMenu ul {
width : 390px;
float : right;
}
#controlMenu li {
position : relative; /* 为下拉菜单生成包含块 */
z-index:100; /* 防止后面可能出现的定位元素遮盖菜单 */
float : left;
margin-right : 2em;
}
```

“#controlMenu ul”表示“#controlMenu”内所有的<ul>元素，因此包括2级菜单的<ul>，所以在后面还需要对2级<ul>取消这些CSS规则，“#controlMenu li”也是如此。

5. 2级菜单的定位

由于前面定义了<ul>和<li>的规则，而对于2级菜单则需要去除这些规则，同时2级菜单有自己的CSS：

```
#controlMenu ul ul {
width : auto; /* 清除前面的定义 */
float:left; /* 浮动的元素会压缩宽度，使ul的宽度适用其内容宽度 */
background : #39F;
border : 1px solid #039;
padding : 4px;
}
#controlMenu ul ul li {
float : none; /* 清除前面的定义 */
position : static; /* 清除前面的定义 */
border-top : 1px dotted #CFF;
padding : 1px 0;
margin : 0; /* 清除前面的定义 */
}
```

“#controlMenu ul ul”表示“#controlMenu”内<ul>内的<ul>，由于本代码中只有2层<ul>，因此此选择器匹配2级菜单的<ul>，而如果2级菜单内还包含3级<ul>，则“#controlMenu ul ul”也包括3级的<ul>。“#controlMenu ul ul li”也同理。同时设定2级 的CSS如下：

```
#controlMenu div {
background:url(../img/shadow.png) no-repeat right bottom; /* 阴影背景图片 */
padding:0 4px 4px 0; /* 使阴影显示出来*/
position:absolute;
left:0;
top:26px;
}
```

由于绝对定位的元素如果width值为“auto”会被压缩宽度，因此产生阴影背景的 会同<ul>等宽，则背景无法显示出来，因此要给 右、下方增加补白，使背景的阴影显示。

6. 2级菜单内<a>元素的设定

2级菜单内的<a>元素需要独占1行，且在鼠标指向时有背景色，因此需要将其设定为块级元素：

```
#controlMenu ul ul a {
display : block;
width : 5em;
line-height : 18px;
padding : 0 5px;
}
#controlMenu ul ul a:hover {
background : #CFF;
color : #039;
text-decoration : none;
}
```

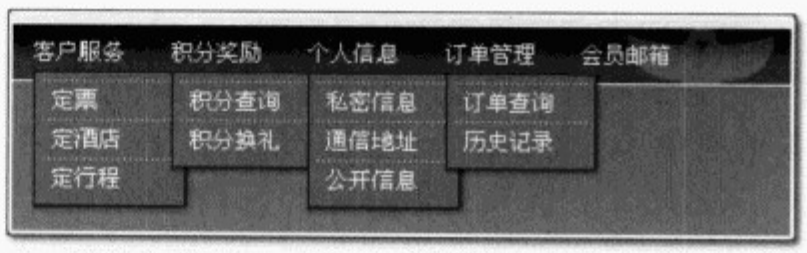
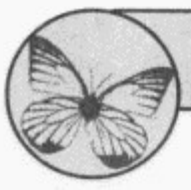


图17-66　2级菜单效果

此时在Opera 9.2内的效果如图17-66所示。

由图17-66可以发现，2级第一个元素<li>的上边框是不需要的，因此增加CSS如右：

```
#controlMenu ul ul li:first-child {
border : 0;
}
```



提示： 关于:first-child伪类，可参见本书［4.3.1 伪类（Pseudo-Classes）］一节。

此时的显示效果如图17-67所示。

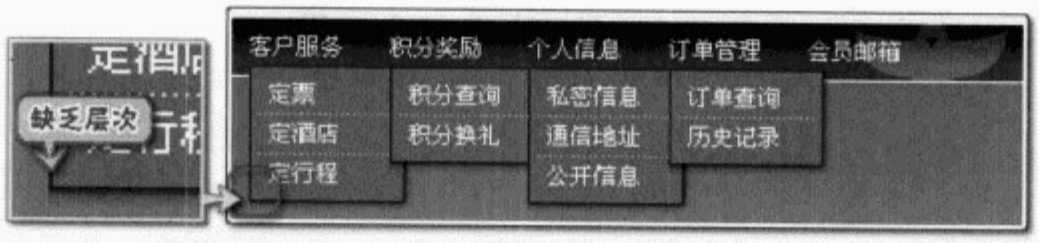


图17-67　去除第一个<li>的边框

由于阴影是向斜下方的，因此需要调整2级 和<ul>的设定：

```
#controlMenu ul ul {
……
padding : 4px;
margin : -4px 0 0 -4px; /* 负边距使元素向上/左移动 */
}
#controlMenu div {
……
left : 4px; /* 0 + 4px */
top : 30px; /* 26px +4px */
}
```

此时2级菜单显示效果如图17-68所示。此时可以设定菜单的visibility属性以隐藏2级菜单，同时增加对1级菜单鼠标指向时的CSS：

```
#controlMenu div {
……
left : 4px;
top : 30px;
visibility : hidden; /* 隐藏菜单 */
}
#controlMenu li:hover div {
visibility : visible;
}
#controlMenu li a:hover {
border:0; /* 解决IE中a:hover不响应的Bug，参见本书［9.2.2 应用：显示或隐藏元素］一节 */
}
```

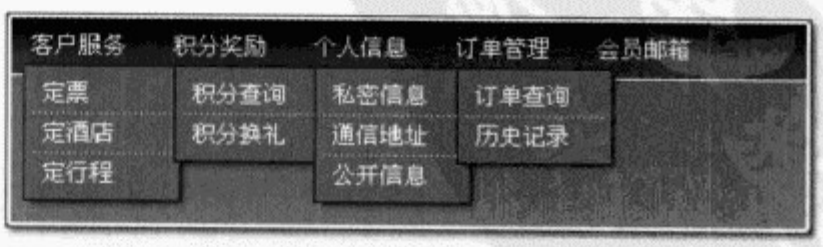


图17-68　调整边距后阴影的显示

此时2级菜单的效果如图17-69所示。

图17-69 2级下拉菜单的完成效果

7. 针对IE 6.0的调整

由于IE 6.0不支持非<a>元素的:hover伪类，同时也不支持:first-child伪类，因此有2种方法解决这个问题：使用JavaScript控制菜单的显示，或者增加一些额外的(X)HTML代码并修改CSS。

JavaScript控制菜单的显示不在本书的讨论范围之内，因此本例将采用增加额外(X)HTML代码的方法来实现。

提示： 关于通过使用JavaScript使IE 6.0支持:hover、透明PNG图片等的方法，读者可以参阅http://www.ddcat.net/bbs2007/showthread.php?t=578（中文）及http://dean.edwards.name/weblog/2008/01/ie7-2/（英文）。

通过增加额外的(X)HTML代码可以使IE 6.0实现纯CSS的下拉菜单，需要为每个1级<li>添加如下代码：

```
<li><a href="/service/index.html" title="客户服务项目查询">客户服务<!--[if IE 7]><!--></a><!--<![endif]-->
 <!--[if lte IE 6]><table><tr><td><![endif]-->
 <div>
  <ul>
   <li class="first-child"><a href="/service/ticket.html" title="预定国内飞机、火车、船票">定票</a></li>
   <li><a href="/service/hotel.html" title="预定国内各地酒店">定酒店</a></li>
   <li><a href="/service/journey.html" title="预定国内旅行社行程">定行程</a></li>
  </ul>
 </div>
 <!--[if lte IE 6]></td></tr></table></a><![endif]-->
</li>
```

上述代码主要是利用IE的条件注释插入只有IE 6.0及更早版本会识别的标签：

```
<!--[if lte IE 6]><table><tr><td><![endif]-->
<!--[if lte IE 6]></td></tr></table></a><![endif]-->
```

因此，对于IE 7.0及非IE浏览器，实际有效的XHTML代码如下：

```
<li><a href="/service/index.html" title="客户服务项目查询">客户服务</a>
 <div>
  <ul>
   <li class="first-child"><a href="/service/ticket.html" title="预定国内飞机、火车、船票">定票</a></li>
   <li><a href="/service/hotel.html" title="预定国内各地酒店">定酒店</a></li>
   <li><a href="/service/journey.html" title="预定国内旅行社行程">定行程</a></li>
  </ul>
 </div>
</li>
```

而对于IE 6.0及更早的版本，实际有效的XHTML代码如下：

```
<li><a href="/service/index.html" title="客户服务项目查询">客户服务
 <table><tr><td>
 <div>
  <ul>
   <li class="first-child"><a href="/service/ticket.html" title="预定国内飞机、火车、船票">定票</a></li>
   <li><a href="/service/hotel.html" title="预定国内各地酒店">定酒店</a></li>
   <li><a href="/service/journey.html" title="预定国内旅行社行程">定行程</a></li>
  </ul>
 </div>
 </td></tr></table>
 </a>
</li>
```

利用<a>元素包裹一个<table>元素及2级菜单。这个方法来自网站CSSplay（http://www.cssplay.co.uk），笔者试验了各种标签，最后确认只有在<table>的效果最理想。可以认为，IE对于<table>内的元素有很多特殊的理解。

同时需要调整CSS：

```
#controlMenu .first-child {
border : 0;  /* <li class="first-child">去掉2级菜单第1个<li>的上边框 */
}
* html #controlMenu a:hover div {
visibility : visible;  /* 利用<a>元素的:hover伪类显示菜单 */
}
* html #controlMenu a {
display : block; /* 针对1级菜单的<a>元素，增大:hover的响应范围，否则鼠标无法移到2级菜单 */
}
```

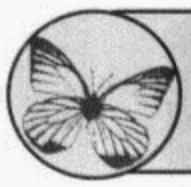
IE 6.0不支持透明PNG图片，可以通过使用IE 5.5及更新版本开始支持的私有滤镜属性来解决。

```
* html #controlMenu div {
background : none;
padding : 0;
left : 0;
top : 26px;
filter : progid:DXImageTransform.Microsoft.Shadow(color=#666666, Direction=135, Strength=4);
}
* html #controlMenu div ul {
margin : 0;
}
```

Shadow滤镜可以在指定的方向建立物体的投影，其语法如下：

语法	filter : progid:DXImageTransform.Microsoft.Shadow (enabled=bEnabled , color=sColor , direction=iOffset , strength=iDistance)
说明	在指定的方向建立物体的投影
属性	enabled：可选项，布尔值（Boolean：true | false），设置滤镜是否激活。其中true是默认值，表示滤镜激活，而false表示滤镜被禁止。 color：可选项，字符串（String），此滤镜作用的颜色值，其格式为#RRGGBB。 direction：可选项，整数值（Integer），设置滤镜效果的运动偏移方向，默认单位为角度。 strength：可选项，整数值（Integer），单位为像素（px），设置在运动方向上的向外扩散距离，其取值范围为>=0，默认值为5

其中direction属性值小于0或大于360的值会自动转换为0~360之间的值。例如，-45会转换为315。此外，0＝上，45＝右上，90＝右，135＝右下，180＝下，225（默认值）＝左下，270＝左，315＝左上。



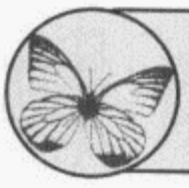
提示：应该将针对IE 6.0的CSS规则写入单独的CSS文件，然后通过IE条件注释链接到XHTML文件内。

注意：非IE浏览器不支持IE的私有滤镜，同时滤镜虽然提供了丰富的效果，但是也会占用系统资源，有时候可能会导致IE无法响应，因此在使用的时候需要特别注意。关于滤镜更多的介绍读者可参阅http://www.linux-cn.com/html/website/css/20070413/ 7198.html（中文）。

此时网页在IE 6.0内的效果如图17-70所示。

图17-70 修改后的代码在IE 6.0内的显示效果

至此header层制作完毕，下面将制作main层的内容。



提示： 由图17-70读者可以发现，IE 6.0内边框的“dotted”样式为虚线，而不是点线，不过这样对于效果没有太大影响，因此可以接受。

17.3.7 main层

main层内是网页的主要内容，内容各层主要是左右2列布局版式，这种版式有多种实现方法，例如本书［8.10.2 应用：宽度自适应的布局］所介绍的使用左浮动和下边距实现2列布局：

```
#travels,
#forumList,
#forumHot {
float : left;
clear : left;
width : 590px;
}
#hot,
#ad1,
#photos,
#club {
margin-left : 600px;
}
```

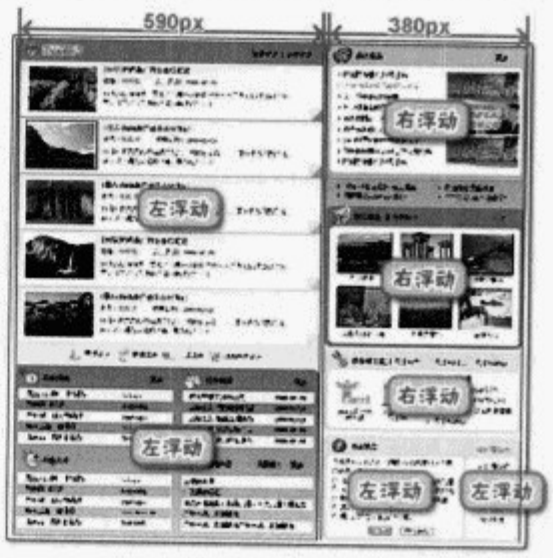
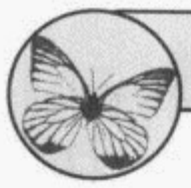


图17-71 main层内容版式分析



提示： vote层和community层并列显示，因此需要另行设定。

由于main层及内部各层的宽度是固定的，因此也可以全部使用浮动（float）来实现左右布局的版式，如图17-71所示。

在本例中将采用全部浮动的布局方式，CSS规则如下：

```
#travels,
#forumList,
#forumHot {
float : left;
width : 590px;
margin-top : 10px;
clear : left;
}
#hot,
#ad1,
#photos,
#club {
float : right;
clear:right;
width : 380px;
}
#vote {
```

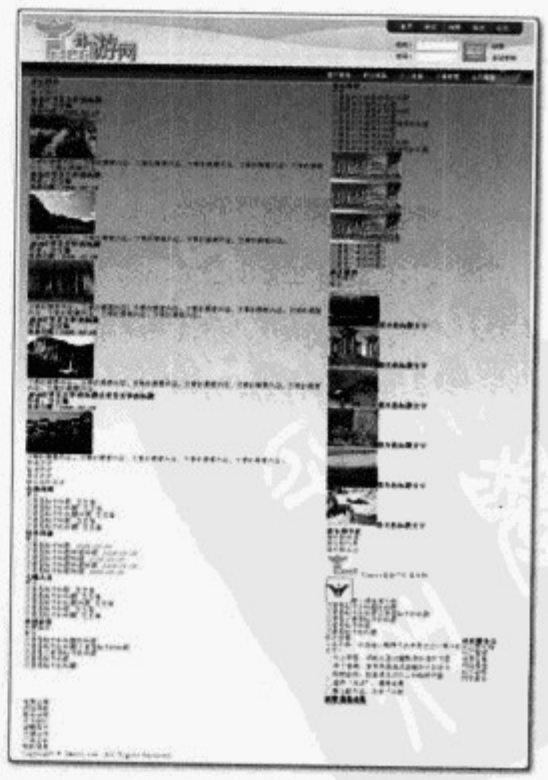
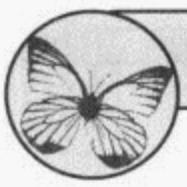


图17-72 总体2列版式布局效果

```
float : left;
margin-left : 10px;
display : inline;
width : 268px;
}
```

```
#community {
width : 112px;
float : left;
}
```

此时在浏览器内效果如图17-72所示。由图17-72可以发现，main层内各栏目的标题和subjectNav列表并排显示且样式基本是一样的，只是位置不同，因此可以统一设定相同的部分：



提示：main层的宽度和居中已经在前面设定，在此不需要再设定CSS。

```
#main h2,
#forumHot h3 {
float : left;
display : inline;
height : 35px;
line-height : 35px;
}
.subjectNav {
float : right;
display : inline;
```

```
font-size : 0.9em;
}
.subjectNav a:link {
color : #039;
}
.subjectNav a:hover {
color : #F90;
text-decoration : none;
}
```

17.3.8 travels层

travels层的细节分析如图17-72所示，其中<h2>和文章列表<ul>需要制作背景图片。

1. 栏目标题<h2>

<h2>的实现方法同［7.2.4 应用：隐藏单行文字］一样，将特殊字体的文字制作为背景图片，同时利用text-indent属性隐藏<h2>内的文字。

由图17-73读者可以发现，<h2>的背景图片位于蓝色背景之上，同时图标部分有阴影，如果直接建立一个背景透明的文件，导出GIF图片时阴影部分可能会有杂点，如图17-74所示，因此需要特别处理。

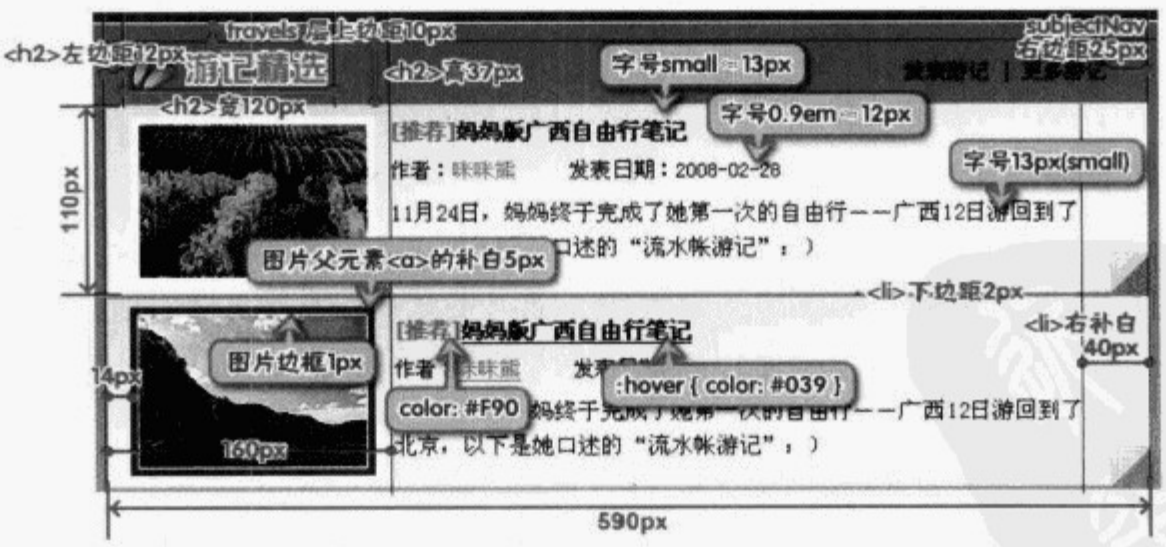
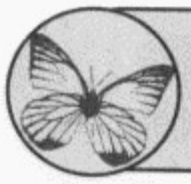


图17-73　travels层的细节分析

图17-74　<h2>的背景图片细节分析

（1）复制<h2>的标题对象，按【Ctrl】+【N】组合键打开【新建文档】对话框，如图17-75所示建立新文档。



提示：画布的颜色取设计稿内<h2>标题附近的背景颜色，如果为过渡色，则取相近的颜色即可。

（2）按【Ctrl】+【V】组合键将复制的标题对象粘贴到新建文档中。

（3）打开【优化】面板（快捷键【F6】），如图17-76所示设置优化选项，按 重建 按钮重建颜色库。

图17-75 修改画布颜色

图17-76 <h2>背景图片的优化选项设定

（4）按【Ctrl】+【Shift】+【R】组合键将文件导出到【bg_01.gif】文件相同的文件夹，文件名为【travels_h2_bg.gif】。<h2>的CSS设定如下：

```
#travels {
padding-top : 10px;
}
#travels h2{
width : 120px;
height : 37px;
margin-left : 12px;
text-indent : -9999em;
background : url(../img/ travels_h2_bg.gif) no-repeat;
}
```

2. 栏目内小导航subjectNav

subjectNav列表右浮动与<h2>并排显示，同时，其内的列表项需要左浮动以横排显示（也可以使用“display : inline”），列表项之间的“|”通过设定<li>的左边框实现，与controlMenu层的2级菜单的情况类似，第一个列表项不需要边框，同时为了兼容IE 6.0，因此需要修改XHTML代码如下：

```
<ul class="subjectNav">
 <li class="first-child"><a href="/blog/" title="发表新的游记">发表游记</a></li>
 <li><a href="/blog/" title="查看更多的游记">更多游记</a></li>
</ul>
```

相应的CSS如下：

```
#travels .subjectNav {
margin : 10px 25px 0 0;
}
#travels .subjectNav li {
```

图17-77 <h2>和小导航完成效果

```
float : left;
line-height : 1.2;
padding : 0 0.5em;
display : inline;
border-left : 1px solid #039;
}
#travels .subjectNav li.first-child {
border : 0;
}
```

此时页面在浏览器内的效果如图17-77所示。

3. 文章列表travelsList

文章列表的背景图片为渐变图片，颜色比较多，因此需要保存为单独的图片。

（1）将文章列表的背景复制到新的文档内，如图17-78所示。

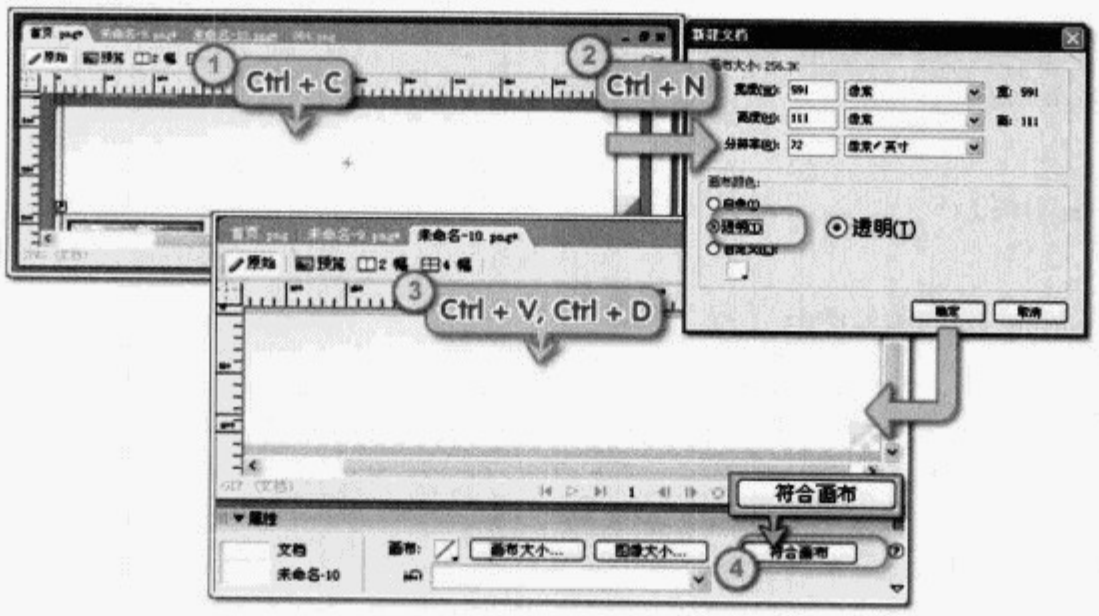


图17-78　将travelsList的背景复制到新建文档

（2）按【Ctrl】+【Shift】+【S】组合键打开【另存为】对话框，如图17-79所示设定并保存文件。

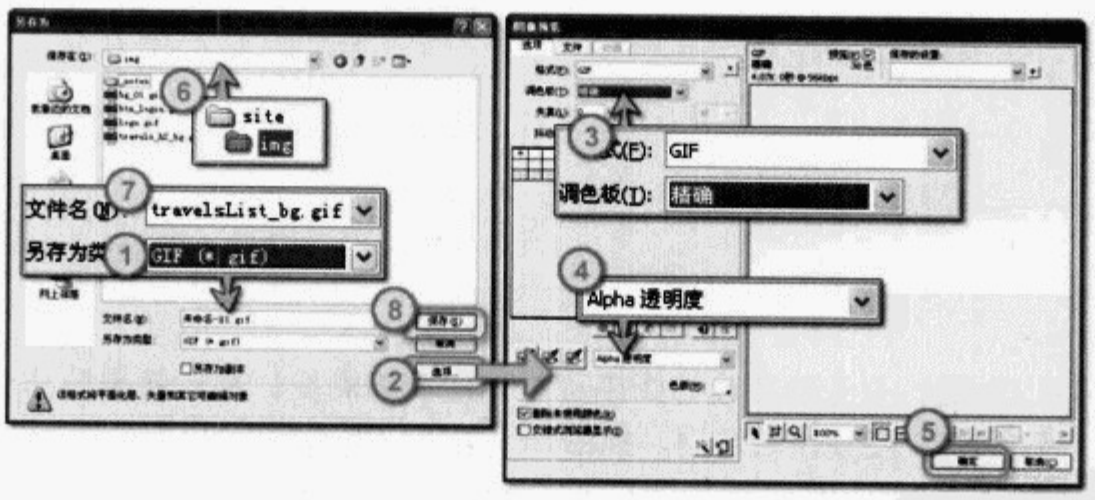


图17-79　保存背景图片

文章列表的结构如下：

```
<ol class="travelsList">
 <li>
  <h3><strong>推荐</strong><a href="/blog/" title="文章标题">这里是文章的标题</a></h3>
  <h4>作者：<a href="/blog/ddcat/" title="进入[豆豆猫]的Blog">豆豆猫</a></h4>
  <h5>发表日期：2008-02-28</h5>
  <p class="pic"><a href="/blog/" title="文章标题"><img src="pic/blog_01.jpg" title="文章的标题" alt="插图
" width="128" height="88" /></a></p>
  <p>文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的摘要内容。文章的摘
要内容。文章的摘要内容。</p>
 </li>
 ……
</ol>
```

由图17-73可以发现，文章列表也是一个2列的版式，文章插图需要居左，其他内容为右列，而<h4>和<h5>并排显示。在此介绍另一种2列布局的方式，即使用绝对定位来定位文章插图。首先设定其他内容的CSS：

```
.travelsList {
clear : both; /* 清除标题和小导航的浮动 */
}
.travelsList li {
height : 110px;
position : relative; /* 生成包含块 */
line-height : 2; /* 估计值，可根据实际效果调节 */
padding : 0 40px 0 160px;
margin-bottom : 2px;
background : url(../img/travelsList_bg.gif) no-repeat;
}
.travelsList h3 a:hover {
color : #039;
}
.travelsList h3 a:visited {
color : #999;
}
.travelsList h3 strong { /* [推荐]的样式 */
color : #f60;
}
.travelsList h3 em { /* [国内][国外]的样式 */
font-style : normal;
color : #666;
}
.travelsList h4,
.travelsList h5 {
float : left;
font-size : 0.9em;
font-weight : normal;
}
.travelsList h4 {
margin-right : 3em; /* 估计值，使其和后面的<h5>保持一定距离 */
}
.travelsList h4 a:link {
color:#F90;
}
.travelsList p {
clear : both;
line-height : 1.7; /* 估计值，可根据实际效果调节 */
}
```

由上述代码可以发现，有些值在设计图中可能无法准确测量，那么就需要估计一个值，并且根据实际效果调节这个值，使其比较合理。插图部分的CSS规则如下：

```
.travelsList .pic {
position : absolute;
top : 5px; /* （总高110px -图片高88px -上下边框2px -上下补白5px）÷2 */
left : 14px;
}
.travelsList .pic a {
display : block;
background : #fff;
padding : 5px;
}
.travelsList .pic a:hover {
background : #039;
}
.travelsList .pic a img{
width:128px; /* 统一图片的尺寸防止有不合尺寸的图片破坏布局 */
height:88px;
```

```
vertical-align : top; /* 防止图片下方出现空隙 */
border : 1px solid #fff;
}
```

此时页面在浏览器内显示如图17-80所示。至此，travels层制作完毕。

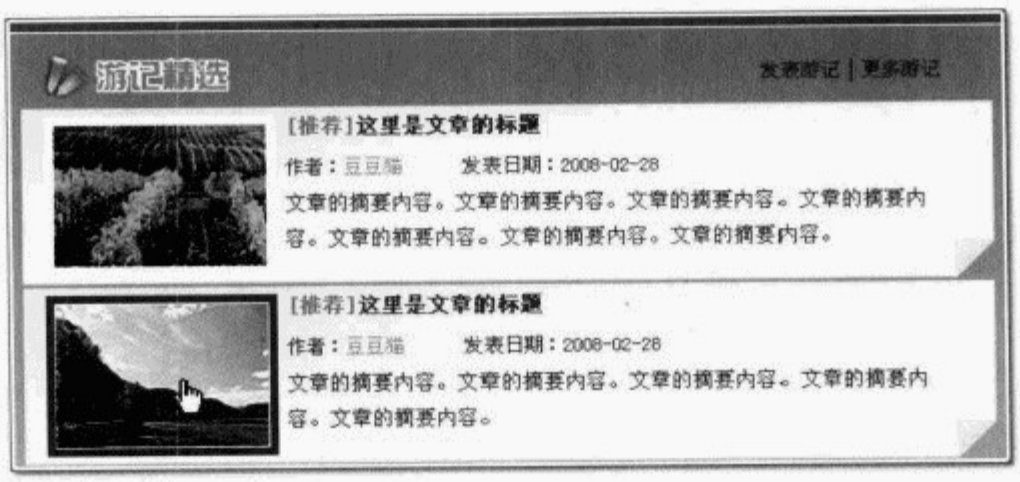


图17-80　travels层完成效果

17.3.9 hot层

hot层结构如下：

```
<div id="hot">
 <h2>热点推荐</h2>
 <ul class="subjectNav">
  <li><a href="#" title="查看更多的游记">更多...</a></li>
 </ul>
 <ol class="hotArticle">
  <li><a href="#" title="这里是热点推荐的标题">这里是热点推荐的标题</a></li>
  ……
 </ol>
 <ol class="hotPicture">
    <li><a href="#" title="标题文字"><img src="pic/ad_01.jpg" alt="热点图片" width="140" height="60"
/></a></li>
  ……
 </ol>
</div>
```

hot层内主要内容为文字列表和图片列表，也是左右2列的版式，因此可以采取左右浮动的方法来布局。同时，文字链接前面的小图标需要设置背景图片，同时其不同状态的小图标不同，如图17-81所示。

hot层的高度和宽度都是固定的，因此可以使用1个背景图片，而且这个背景图片颜色很少，因此可以放置在大的背景图片内，如图17-82所示。

图片的高即为hot层的高度，Y坐标为背景偏移量。<h2>前面的图标图片颜色比较丰富，因此同travels层<h2>背景的处理方法相同，单独保存为GIF图片【hot_h2_bg.gif】。

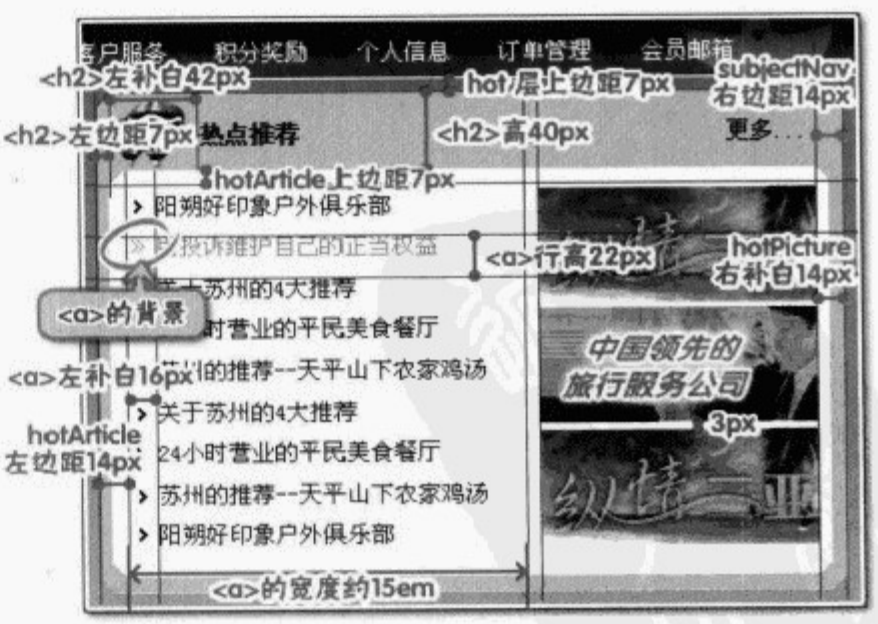


图17-81　hot层细节分析

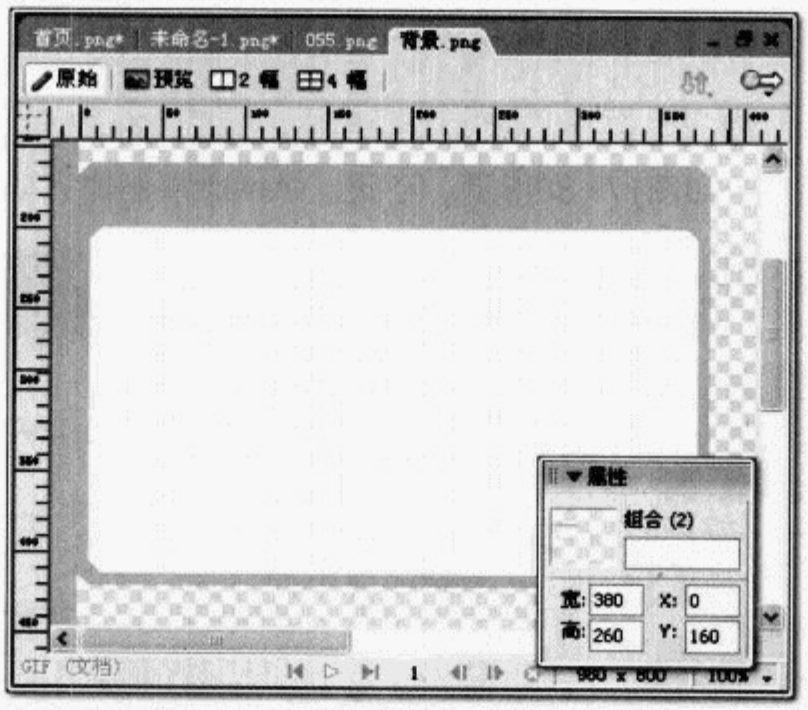


图17-82 hot层背景图片属性

```
#hot {
margin-top : 7px;
height : 260px;
overflow : hidden;
background : url(../img/bg_01.gif) no-repeat 0 -160px;
}
#hot h2 {
margin-left : 7px;
padding-left : 42px;
height : 40px;
line-height : 40px;
background : url(../img/hot_h2_bg.gif) no-repeat left center;
}
#hot .subjectNav {
line-height : 40px;
margin-right : 7px;
}
#hot .hotArticle {
float : left;
clear : both; /* 清除前面的浮动元素，不会影响到后面的浮动元素 */
margin :7px 0 0 14px;
}
#hot .hotPicture {
clear : right;
float : right;
padding : 7px 14px 0 0;
}
#hot .hotPicture li {
margin-bottom : 3px;
}
#hot .hotPicture img {
width:140px;
height:60px;
vertical-align:top;
}
```

对于hotArticle内的链接元素，文字前面有小图标，分为3种状态如图17-83所示。

其实现原理同［10.3.3 简单的链接背景替换］一节介绍的原理相同，即通过对:link、:hover和:visited设定不同的背景偏移量实现，这3种状态的小图标也可以放置在大背景图片内，3个图片垂直间距为<a>元素的行高22px，如图17-84所示。

链接文字需要在一行内显示不回行，最多显示宽度约为15em。

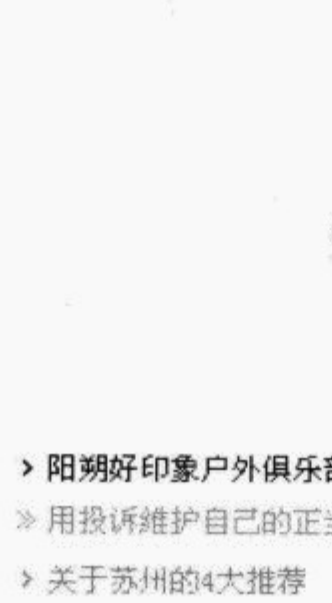


图17-83　hotArticle内的链接的不同状态

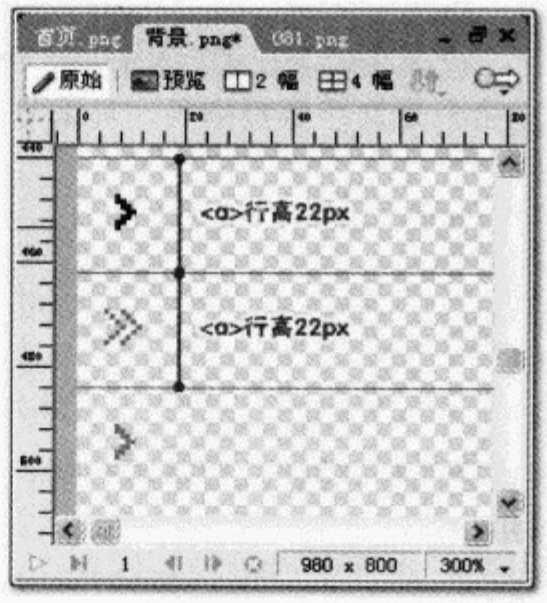


图17-84　hotArticle内的链接的背景图片

```
#hot .hotArticle a {
font-size : 0.9em;
padding-left : 16px;
color : #039;
background : url(../img/bg_01.gif) no-repeat 0 -440px;
display : block; /* 块级元素才可以设定宽度 */
width : 15em;
white-space : nowrap;
overflow : hidden;
text-overflow : ellipsis;
-o-text-overflow : ellipsis;
text-decoration : none;
}
#hot .hotArticle a:visited {
color:#999;
background-position: 0 -484px;
}
#hot .hotArticle a:hover {
color:#f90;
background-position: 0 -462px;
}
```

“white-space : nowrap”表示不回行，但是必须同时设定宽度才可生效。“text-overflow”属性为CSS 3的属性，表示当文字溢出时如何处理，属性值“ellipsis”表示在被截断的文字后面加“…”表示，IE 6.0、IE 7.0和Safari 3.0都支持这个属性，而“-o-text-overflow”是Opera的私有属性，其功能与“text-overflow”相同，如图17-85所示。

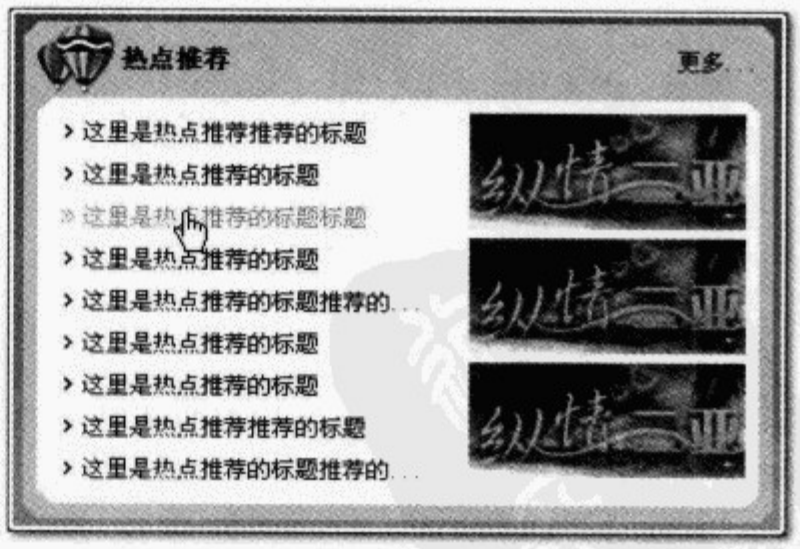
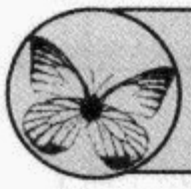


图17-85　“text-overflow”属性的效果



提示：遗憾的是，Firefox 2.0不支持该属性，不过可以通过JavaScript实现截字的效果，读者可参阅http://realazy.org/blog/2007/05/17/avoid-cutting-half-character/（中文）。

17.3.10 ad1层

ad1层的内容只是文字链接列表，其细节分析如图17-86所示。<li>需要浮动以并排显示。

```
#ad1 ul {
margin : 11px 14px;
}
#ad1 li {
float : left;
width : 50%;
}
```

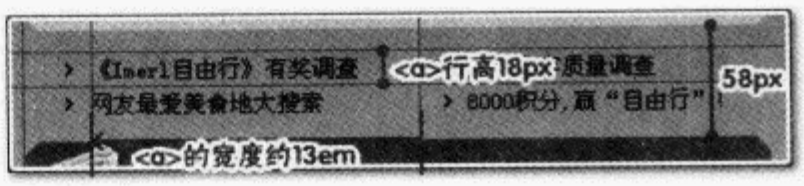


图17-86 ad1层细节分析

本层文字链接的样式同hot层基本一样，只是<a>元素的行高和宽度不同，因此可以和hot层的链接样式和并：

```
#hot .hotArticle a,
#ad1 a {
……
}
#hot .hotArticle a:visited,
#ad1 a:visited {
……
}
#hot .hotArticle a:hover,
#ad1 a:hover {
……
}
```

然后再利用层叠重新定义不同的属性：

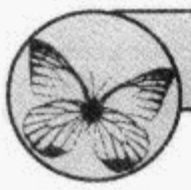
```
#ad1 a {
line-height : 18px;
width : 13em;
background-position:0 -442px; /* hot行高22px ， ad1行高18px，上下相差2px */
}
#ad1 a:visited {
background-position: 0 -486px;
}
#ad1 a:hover {
background-position: 0 -464px;
}
```

17.3.11 photos层

photos层的外观和hot层很接近，也是定宽定高，<h2>的表现形式也一样，如图17-87所示。

```
#photos {
height : 282px;
background : url(../img/bg_01.gif) no-repeat 0 -530px;
}
#photos h2 {
margin-left : 7px;
padding-left : 42px;
height : 40px;
line-height : 40px;
padding-right : 11px;
background : url(../img/photos_h2_bg.gif) no-repeat left center;
}
```

图17-87 photos层细节分析



提示：为了节约篇幅，photos层的背景和<h2>背景的制作方法不再赘述。

其subjectNav的第1个<li>居左，而第2个<li>居右，为了兼容IE 6.0，也需要增加代码如下：

```
<div id="photos">
 <h2>照片精选</h2>
 <ul class="subjectNav">
  <li class="first-child"><a href="/photo/" title="发表新的游记">发表照片</a></li>
  <li><a href="/photo/" title="查看更多的游记">更多...</a></li>
 </ul>
 <ul class="photoList">
  <li><a href="/photo/" title="图片1的标题文字"><img src="pic/photo_01.jpg" width="100" height="80" alt="
照片精选1" /></a><strong><a href="/photo/" title="图片1的标题文字">图片的标题文字</a></strong></li>
  ……
 </ul>
</div>
```

由图17-87可以发现，subjectNav整体是居左的，而不是居右，因此设定CSS如下：

```
#photos .subjectNav {
float : left;
line-height : 14px;
margin : 13px 0;  /* (高40px – 行高14px) ÷ 2 */
width : 250px;  /* 不设定宽度则浮动元素宽度会被压缩 */
}
#photos .subjectNav li{
float : right;
padding-left : 11px;
}
#photos .subjectNav li.first-child {
float : left;
border-left : 2px solid #039;
}
```

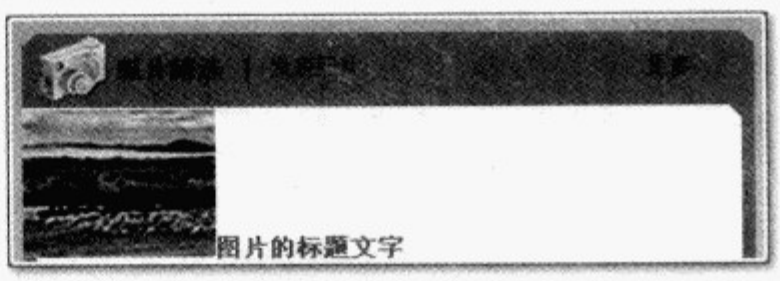


图17-88　photos层内subjectNav的效果

此时页面在浏览器内的效果如图17-88所示。

图片列表是2行3列的版式，同时内容居中，CSS设定如下：

```
#photos .photoList {
padding : 12px 14px 0 14px;
clear : both;
}
#photos .photoList li {
width : 33.3%;
float : left;
text-align : center;
line-height : 2em;  /* 估计值 */
}
#photos .photoList img {
width:100px;
height:80px;
vertical-align : top;
border : 1px solid #039;
}
#photos .photoList strong {
font-weight : normal;
display : block;  /* 使文字独占1行 */
}
```

图17-89　photos层在Opera 9.2内的显示

此时在Opera 9.2内浏览页面如图17-89所示，而在IE内的效果却如图17-90所示。

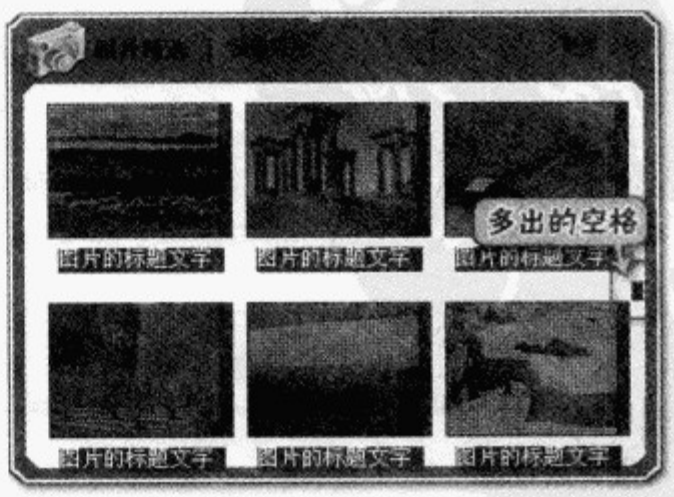


图17-90　photos层在IE内的显示

IE对于空格处理的问题使得两行之间空隙加大，因此需要通过Hack触发<ul>的hasLayout属性解决，修改photoList的CSS规则如下，此时在IE内显示将恢复正常。

```
#photos .photoList {
padding:12px 14px 0 14px;
```

```
clear:both;
*min-height:1%; /* 针对IE 7.0 */
_height:1%; /* 针对IE 6.0*/
}
```

17.3.12 forumList层

forumList层包括4个链接，每个链接前都有图标，实现其表现有两种方法：

- 为每个链接定义ID或者class，然后分别设置背景图片；
- 将所有图标制作为1个<ul>背景图片，而每个<li>左浮动，固定字体大小、左补白和右补白，而由字数多少决定<li>的宽度，如图17-91所示。

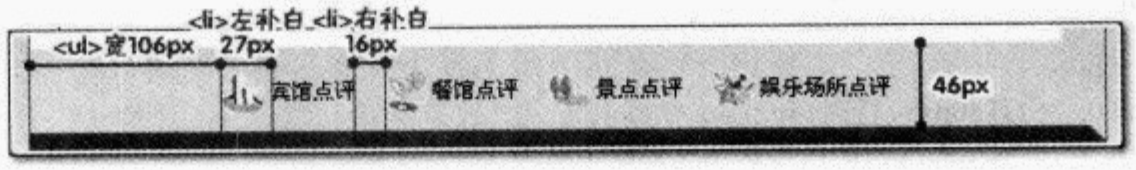


图17-91 forumList层图标的实现方法2

第2种方法的缺点在于，由于背景的大小是固定的，如果访问者调整字号大小，<li>的宽度变化，则文字与背景会发生错位，因此本例中将采取第1种方法。

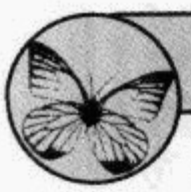
首先需要修改XHTML代码如下：

```
<div id="forumList">
 <ul>
  <li class="forumList1"><a href="/forum/" title="进入[宾馆点评]分论坛">宾馆点评</a></li>
  <li class="forumList2"><a href="/forum/" title="进入[餐馆点评]分论坛">餐馆点评</a></li>
  <li class="forumList3"><a href="/forum/" title="进入[景点点评]分论坛">景点点评</a></li>
  <li class="forumList4"><a href="/forum/" title="进入[娱乐场所点评]分论坛">娱乐场所点评</a></li>
 </ul>
</div>
```

如图17-91所示4个小图标颜色都比较少，因此可以放入大背景图片中，制作方法不再赘述，相应CSS规则如下：

```
#forumList {
height : 46px;
line-height : 46px;
text-align : center;
font-size : 0.9em;
}
#forumList a:link{
color : #039;
}
#forumList li {
display : inline;
padding : 6px 10px 6px 27px;
    background : url(../img/bg_01.gif) no-repeat 0 -840px;
    }
    #forumList .forumList2 {
    background-position : 0 -890px;
    }
    #forumList .forumList3 {
    background-position : 0 -940px;
    }
    #forumList .forumList4 {
    background-position : 0 -990px;
    }
```

设定<li>的“display : inline”使其并排显示，并且水平居中，同时对于行内元素，其背景区域为其内容区域，而图标高度大于文字的高度，因此需要增加上下补白以扩大背景的显示区域。forumList层在浏览器内的显示如图17-92所示。



提示：制作背景图片的时候，图标之间的垂直间隔应大于行距。

至此，forumList层制作完成。

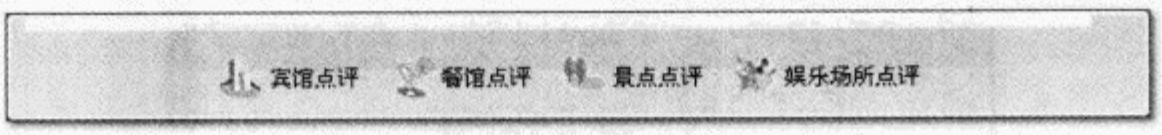


图17-92 forumList层在Opera 9.2内的显示效果

17.3.13 forumHot层

forumHot层包含4个内容层，而这4个层的外观基本相同，如图17-93所示。

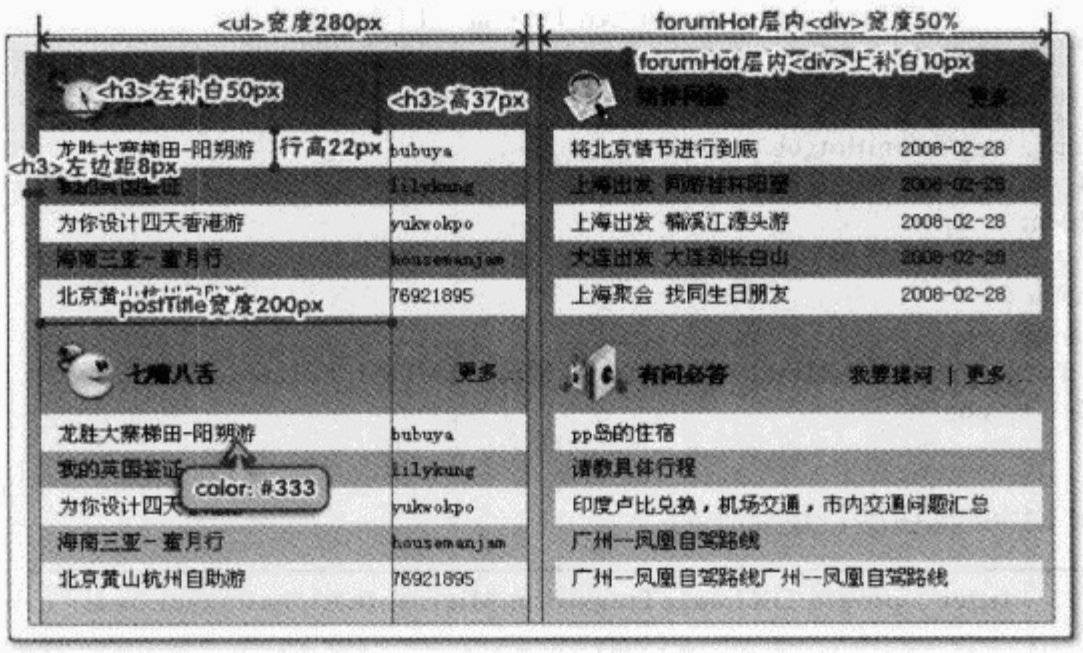


图17-93 forumHot层细节分析

1. forumHot层的背景

forumHot层的背景是渐变色，因此需要单独保存，但是如果输出为GIF文件，则会丢失颜色，如图17-94所示。

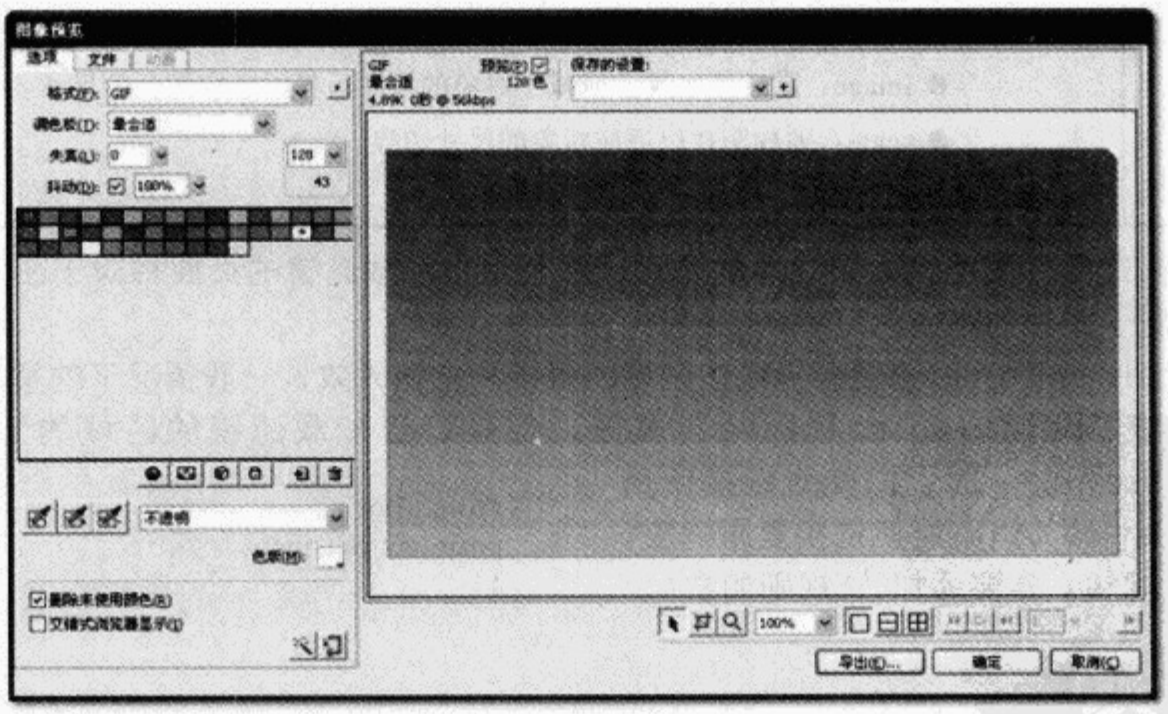


图17-94 受GIF色彩数限制而导致颜色丢失

而由于该背景的右上角需要透明，如果保存为JPEG文件，则无法设置透明通道显示<body>的背景色，从而可能出现颜色的不协调，如图17-95所示。

因此背景图片需要输出为透明的PNG图片，对于IE 6.0使用IE专用的滤镜属性来加载PNG图片。CSS规则如下：

图17-95 JPEG图片不能使用透明通道

```
#forumHot {
background : url(../img/forumHot_bg.png) no-repeat;
height : 340px;
overflow : hidden;
}
* html #forumHot {
background : none;
            filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(sizingMethod=scale,
src="img/forumHot_bg.png");
}
```

AlphaImageLoader滤镜的说明如下：

语法	filter : progid:DXImageTransform.Microsoft.AlphaImageLoader (enabled=bEnabled , sizingMethod=sSize , src=sURL)
说明	在元素的背景和内容之间显示一张图片，并提供对此图片的剪切和改变尺寸的操作。如果载入的图片是PNG格式，则0%~100%的透明度被支持
属性	enabled：可选项，布尔值（Boolean：true | false），设置滤镜是否激活，其中true是默认值，表示滤镜激活，而false表示滤镜被禁止。 sizingMethod：可选项，字符串（String）。设置滤镜作用的图片在元素内的显示方式。 ● cro：剪切图片以适应对象尺寸。 ● image：默认值，增大或减小对象的尺寸边界以适应图片的尺寸。 ● scale：缩放图片以适应对象的尺寸边界。 src：必选项，URI

src属性指定图片的路径，要注意的是这个路径是指加载滤镜的页面相对于图片的路径而不是CSS文件相对于图片的路径。

但是AlphaImageLoader滤镜会导致该区域的链接和按钮无效，一般情况下的解决办法是为链接或按钮添加“position:relative”属性，但是，当加载滤镜的区域的父层绝对定位“position:absolute”时，即使对链接使用“position:relative”也不能使链接复原响应，所以要慎用此滤镜。需要添加CSS规则如右：

```
#forumHot a {
position : relative;
}
```

2. 内容层的标题

由于内容4个层的样式基本相同，因此可以统一设定。4个层的标题<h3>的图标文件色彩都比较丰富，因此需要单独保存为透明GIF文件，制作过程同travels层的<h2>背景制作相同，在此不再赘述。标题的CSS规则如下：

```
#forumHot div{
width : 50%;
float : left;
padding-top : 10px;
```

```
}
#forumHot h3 {
height : 37px;
line-height : 37px;
padding-left : 50px;
margin-left : 8px;
}
#budgetLine h3 {
background : url(../img/forumHot_budgetLine_h3.gif) no-repeat 3px 0;
}
#company h3 {
background : url(../img/forumHot_company_h3.gif) no-repeat 3px 0;
}
#colloquy h3 {
background : url(../img/forumHot_colloquy_h3.gif) no-repeat 5px 0;
}
#question h3 {
background : url(../img/forumHot_question_h3.gif) no-repeat 3px 0;
}
```

由于标题<h3>前图标的宽度不相同，因此背景偏移位置需要根据实际效果进行调节。

3. 小导航subjectNav

4个层内的subjectNav导航只有question层不同，而question层subjectNav的样式同travels层的subjectNav样式类似，列表的第1个<li>的样式不同，因此也需要修改相应的XHTML代码如下：

```
<div id="question">
 <h3>有问必答</h3>
 <ul class="subjectNav">
  <li class="first-child"><a href="/forum/" title="到[有问必答]发帖提问 ">我要提问</a></li>
  <li><a href="/forum/" title="查看[有问必答]更多的帖子">更多...</a></li>
 </ul>
 <ul class="hotList">
  ……
 </ul>
</div>
```

CSS规则如下：

```
#forumHot .subjectNav {
line-height : 37px;
}
#question .subjectNav li {
float:left;
line-height : 1.2em;
margin-top : 10px;
padding : 0 6px;
border-left : 1px solid #039;
}
#question .subjectNav li.first-child {
border : 0;
}
```

4. 内容列表hotList

由图17-93可以发现，列表hotList宽度为280px，在其父层内居中显示，同时，列表项有交替的背景色，即单数行为灰色背景，双数行无背景色，由于目前还没有浏览器支持CSS 3的结构伪类，因此实现交替的背景色一般有3种方法：

- 为每行设定不同的class；
- 使用JavaScript设定背景色；
- 使用双色的背景图片。

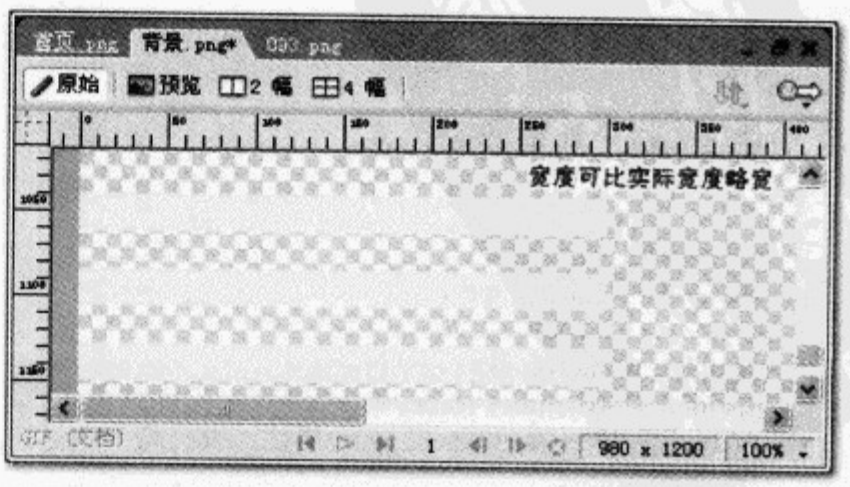


图17-96　hotList的背景图片

由于<li>的高度（行高）固定，因此可以采用背景图片的方法，同时由于每个列表固定高度为5行，因此可以制作1个整体的背景图片，并且可放置在大背景图片之内，如图17-96所示。

设定CSS规则如下：

```
#forumHot .hotList {
font-size : 0.9em;
clear : both;
width : 280px;
margin : auto;
line-height : 22px;
background : url(../img/bg_01.gif) no-repeat 0 -1040px;
}
#forumHot .hotList li {
height : 22px;
overflow : hidden;
padding-left : 10px;
}
```

此时forumHot层显示如图17-97所示。

图17-97 forumHot层在浏览器内的效果

5. 内容

由图17-93可以发现，除了question层只有标题链接以外，其他3个层的<li>内都有2项内容：标题链接和发帖人姓名（或者日期），其结构如下。

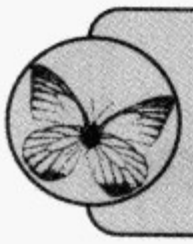
```
<ul class="hotList">
    <li><a href="/forum/" title="这里是帖子的标题" class="postTitle">这里是帖子标题</a> <a
href="/blog/ddcat/" title="查看[豆豆猫]的个人资料" class="postMember">豆豆猫</a></li>
    ……
</ul>
```

由于<a>元素是行内元素，因此其width属性无效，因此需要设定其display属性为“inline-block”，使其可在一行内显示，同时又具有块级元素的属性，CSS规则如下：

```
#forumHot .hotList a,
#forumHot .hotList em {
font-style : normal;
display : inline-block;
display : -moz-inline-stack; /* Firefox 私有属性值，也可以用display: -moz-inline-box */
*zoom:1; /* 触发IE的hasLayout属性，说明见下 */
*display:inline;
```

```
}
#forumHot .hotList a.postTitle {
width : 200px;
white-space : nowrap;
overflow : hidden;
text-overflow : ellipsis;
-o-text-overflow : ellipsis;
text-decoration : none;
}
#forumHot .hotList a:link {
color : #333;
}
#forumHot .hotList a:hover {
color : #F90;
}
#question .hotList a.postTitle {
width : 260px;
}
```

支持“display:inline-block”的浏览器有Opera和Safari等，而Firefox的私有属性值“-moz-inline-stack”和“-moz-inline-box”可以实现“inline-block”的效果，但是这2个属性都存在问题，例如，“-moz-inline-box”会存在元素间的对齐等问题，而如果一个“display:-moz-inline-stack”的元素父元素是“display : inline”的元素，则即会使Firefox中包含的链接不可单击，因此对于私有属性要慎重使用。



提示： Firefox3 beta、IE 8.0 beta支持“display:inline-block”。更多关于inline-block元素的资料，读者可参见http://www.planabc.net/ 2008/04/08/ cross_browser_support_for_inline-block_styling/（中文）。

而对于IE，虽然也不支持“inline-block”值，但是通过触发元素的hasLayout属性，则可以使行内元素具有块级元素的特征，相关CSS规则如下：

```
*zoom:1; /* 触发IE的hasLayout属性 */
*display:inline; /* 将类型恢复为inline */
```

由于“zoom : 1”触发了<a>元素的hasLayout属性，因此即使其类型为“display:inline”也可以设定width属性。此时页面在浏览器内的显示如图17-98所示。

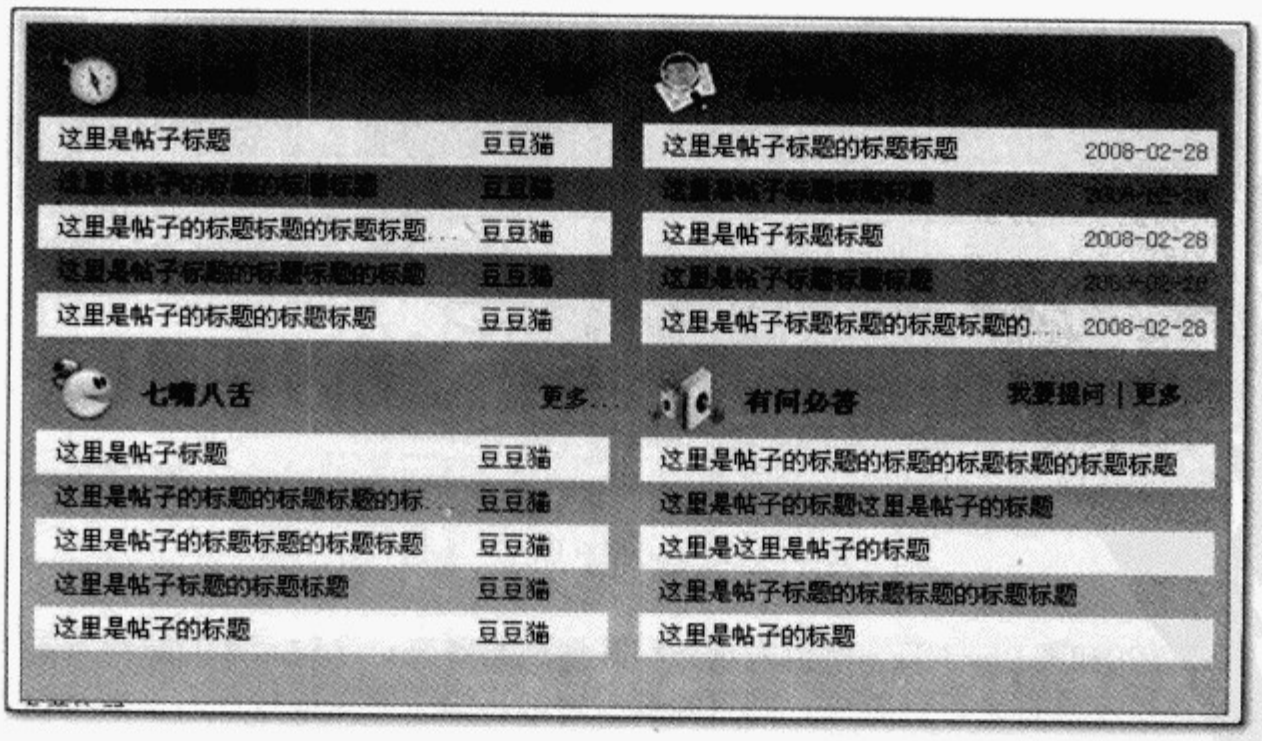


图17-98　forumHot层的完成效果

17.3.14 club层

club层的形式和hot层类似，层背景和<h2>的背景的制作方法不再赘述，其细节分析如图17-99所示。

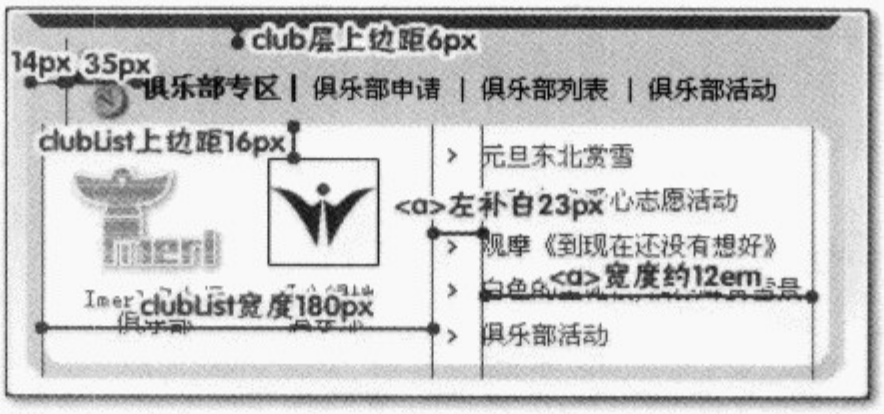


图17-99 club层细节分析

1. club层的上边距

为club层设定CSS规则如下：

```
#club {
margin-top : 6px;
height : 282px;
background : url(../img/bg_01.gif) no-repeat 0 -1180px;
}
```

其在浏览器内显示如图17-100所示。

图17-100 club层在浏览器内的显示

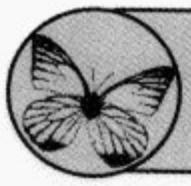
由图17-100可以发现，club层没有同forumList层同行，而是和forumHot层在同1行内浮动，因此造成其顶端空出forumList层的高度。而在IE内显示如图17-101所示。

图17-101 club层在IE内的显示

对比图17-100和图17-101，可以发现IE和其他浏览器在对浮动元素的解释方面有所不同。

由于IE的表现已经符合设计要求，则只需要针对非IE浏览器采取校正方法即可，例如设定负的上边距使元素上移：

```
#club {
margin-top : -42px; /* club层上边距6px -fourmList层高48px */
*margin-top : 6px; /* IE会接受这个值 */
height : 282px;
background : url(../img/bg_01.gif) no-repeat 0 -1180px;
overflow : hidden;
}
```



提示：也可以调整文档的结构，将forumList层包含到forumHot层内，同时调整CSS设定，读者可以自行测试。

club层的<h2>CSS规则如下：

```
#club h2 {
margin-left : 14px;
padding-left : 35px;
height : 40px;
line-height : 40px;
padding-right : 11px;
background : url(../img/club_h2_bg.gif) no-repeat left center;
}
```

2. 小导航subjectNav

club层的subjectNav同photos层类似，不过其<li>全部左浮动即可，同时需要为第1个<li>增加class="first-child"属性。其CSS规则如下：

```
#club .subjectNav {
float : left;
line-height : 14px;
margin : 13px 0;
width : 250px;
}
#club .subjectNav li{
float:left;
border-left : 1px solid #039;
padding : 0 6px;
}
#club .subjectNav li.first-child {
border-left : 2px solid #039;
}
```

3. 俱乐部列表clubList

俱乐部列表clubList的<li>左浮动，宽度为50%，同时其内的<img>设定为块级元素使其独占1行，CSS规则如下：

```
#club .clubList {
width : 180px;
float : left;
display : inline;
margin : 16px 0 0 7px;
font-size : 0.9em;
line-height : 1.5em;
}
#club .clubList li {
float : left;
width : 50%;
text-align : center;
}
#club .clubList li img {
display : block;
margin : auto; /* 水平居中 */
}
```

4. 活动列表activitiesList

activitiesList内的链接样式同hot层的链接样式类似，其设定如下：

```
#club .activitiesList {
float : left;
line-height : 22px;
}
#club .activitiesList a {
display : block;
padding-left : 23px;
background : url(../img/bg_01.gif) no-repeat 0 -440px;
width : 12em;
white-space : nowrap;
overflow : hidden;
```

```
text-overflow : ellipsis;
-o-text-overflow : ellipsis;
text-decoration : none;
}
#club .activitiesList a:visited {
color : #999;
background-position : 0 -484px;
}
#club .activitiesList a:hover {
color : #f90;
background-position : 0 -462px;
}
```

至此club层设定完毕。

17.3.15 vote层和community层

在本书［17.3.7 main层］一节内已经设定了vote层和community层的宽度，而这2个层的细节分析如图17-102所示。

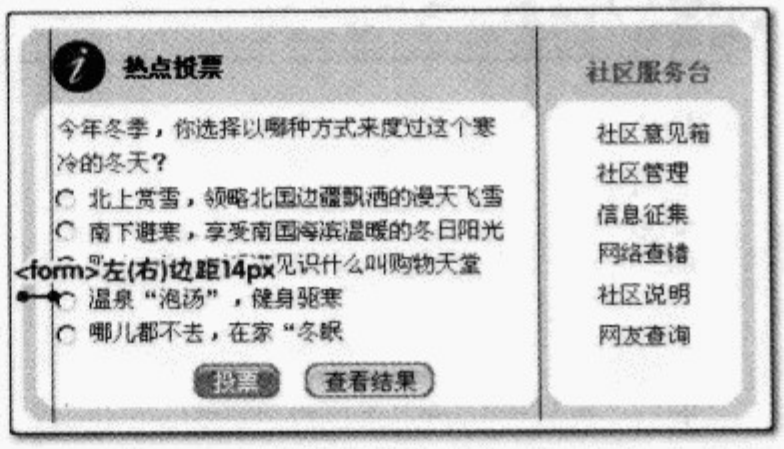


图17-102 vote层和community层细节分析

1. 背景和标题

vote层和community层可以使用同一个背景图片如图17-103所示，设定不同的偏移量即可，在前面已经设定了这2个层的宽度，因此在此不再设定，CSS规则如下：

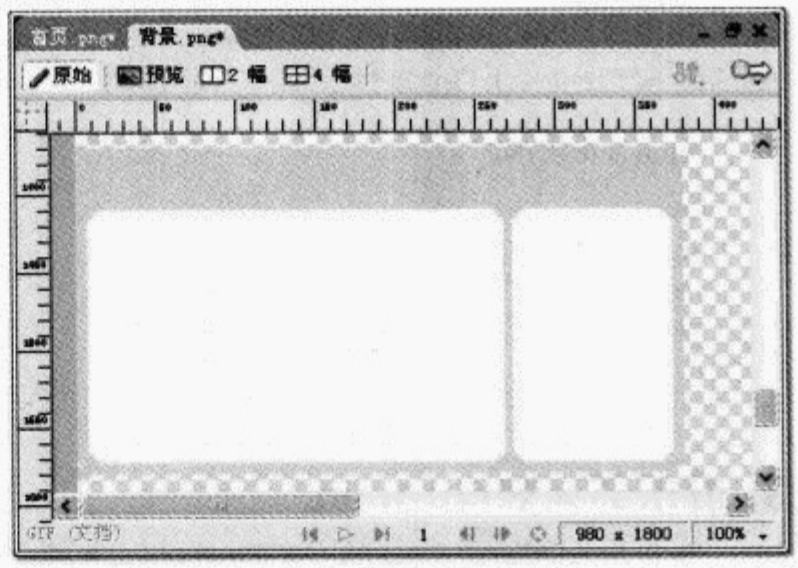


图17-103 vote层和community层的背景图片

```
#vote,
#community {
margin-top : 10px;
 background : url(../img/bg_01.gif) no-repeat 0 -1370px;
height : 210px;
}
#community {
background-position : -268px -1370px; /* -268px为vote层宽度*/
}
#vote form {
font-size:0.9em;
margin:0 14px;
line-height:1.4em;
}
```

2. vote层

vote层没有<h2>标题，但是<form>内的<legend>的作用相同，其CSS规则如下：

```
#vote legend {
height : 40px;
line-height : 40px;
padding-left : 35px;
font-size : 1em;
font-weight : bold;
color : #039;
 background : url(../img/vote_legend_bg.gif) no-repeat left center;
}
#community h2 {
color : #F90;
text-align : center;
display : block;
width : auto;
float : none;
}
```

vote层内的<form>表单包括段落文字、选项列表和按钮3部分内容，其CSS规则如下：

```
#vote p {
margin-top : 5px;
}
#vote p.button {
text-align : center;
}
#vote p.button input {
height : 18px;
border : 0;
margin : 0 5px;
}
```

```
background : url(../img/bg_01.gif) no-repeat;
cursor : pointer;
}
#vote #btnVoteSubmit {
color : #fff;
width : 44px;
background-position : 0 -1630px;
}
#vote #btnShowResults {
width : 66px;
background-position : 0 -1670px;
}
```

3. community层

community层包括标题和列表，列表CSS规则如下：

```
#community h2 {
color : #F90;
text-align : center;
display : block;
width : auto;
float : none;
}
#community ul {
width : 5em;
margin : 14px auto;
line-height : 1.5; /* 估计值 */
}
```

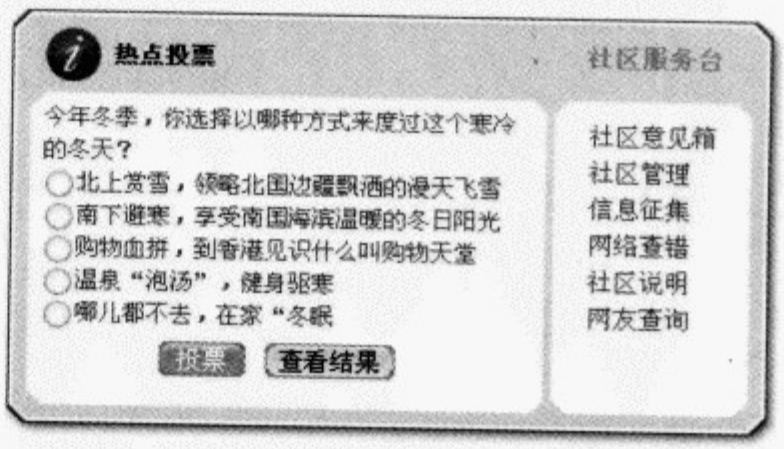


图17-104 vote层和community层设定完成效果

至此vote层和community层设定完成，其显示如图17-104所示。

17.3.16 footer层

footer层相对比较简单，没有背景图片，如图17-4所示。其中副导航条的“|”实现方法同travels层相同，因此需要修改XHTML代码如下：

```
<div id="footer">
 <ul>
  <li class="first-child"><a href="/member/reg.html" title="前往注册页面">免费注册</a></li>
  ……
 </ul>
 <p>Copyright &copy; Imerl.com .All Rights Reserved.</p>
</div>
```

设定副导航条的<li>的display属性为“inline”实现横排显示，CSS规则如下：

```
#footer {
padding : 10px;
text-align : center;
line-height : 2;
}
#footer li {
display : inline;
font-size : 0.9em;
border-left : 1px solid;
padding : 0 0.5em 0 1em;
}
#footer li.first-child {
border : 0;
}
```

在制作mainNav层时介绍过，由于“display:inline”，在每个<li>后面会多一个空格（代码中的空格被压缩为1个），因此设定补白的时候，右补白为0.5em（空格为0.5em），也可以同mainNav层一样删除代码间的表格。此时footer层在Opera 9.2内的效果如图17-105所示。

图17-105 footer层在Opera 9.2内的显示效果

而在IE 6.0内的显示效果却如图17-106所示。

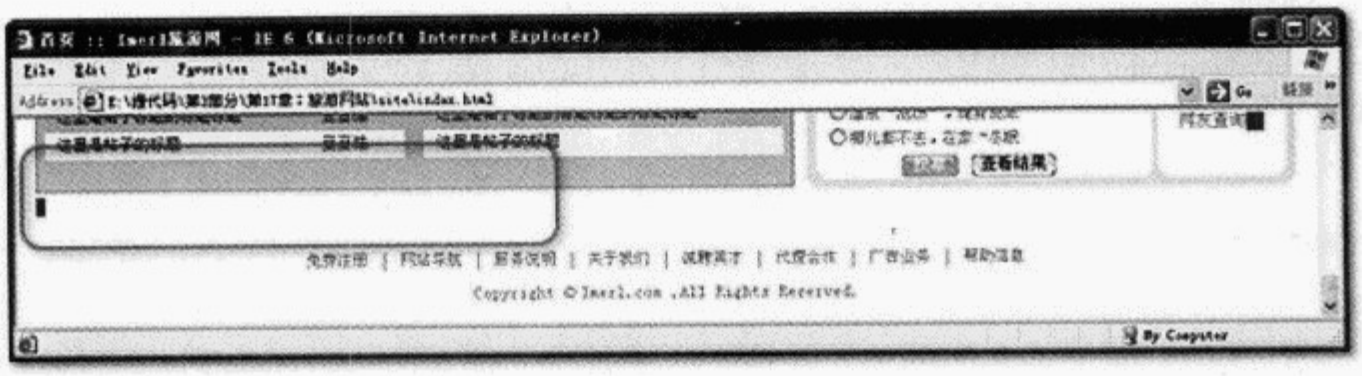


图17-106 IE 6.0内空格造成的空隙

这又是由于IE 6.0对代码中的空格处理不当所引起的问题，最简单的解决方法就是删除代码间的空格：

```
<div id="community">
  <h2>社区服务台</h2>
    <ul><li><a href="/community/suggestion.html" title="为社区建设提意见建议">社区意见箱</a></li><li><a
href="/community/manage.html" title="社区管理">社区管理</a></li><li><a href="/community/info.html" title="
为社区提供信息">信息征集</a></li><li><a href="/community/errata.html" title="为社区提供勘误信息">网络查
错</a></li><li><a href="/community/faq.html" title="社区使用说明">社区说明</a></li><li><a
href="/community/serch.html" title="查询网友信息">网友查询</a></li></ul>
  </div>
```

至此index页面制作完毕，在浏览器内的显示效果如图17-107所示。

图17-107 index页面完成效果

提示：在几个浏览器内查看效果，如果有不合适的地方，可再进行调节，不过也不必过分追求浏览器之间毫厘不差。页面全部制作完毕之后，应该整理CSS文档，将可分组的规则分组书写，某些可以缩写的属性也要缩写。下载文件中为了让读者能够清晰地学习，没有缩写CSS文档。

17.4 版式与结构

本例中的内容模块基本都是限定了高度的，因此可以不增加额外的元素，只利用浮动就可以完成左右2列的版式布局。在此也是为了向读者介绍，如何用最简洁的结构完成版式布局。

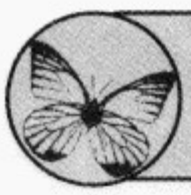
在本章［17.1.4　中间部分的结构化］一节内曾经介绍过，可以利用2个 来实现左右2列的布局，这样的写法相对简单，对于初学者比较好掌握。

而本例中所应用的这种方法的优点在于，各功能模块之间的关系是平级的兄弟关系，因此可以利用CSS的浮动、定位等方法调整版式，例如修改代码如下：

```
#travels {
float : right;
clear : right;
width : 590px;
}
#hot,
#ad1,
#photos {
float : left;
clear : left;
width : 380px;
}
```

其显示如图17-108所示。

图17-108　只修改CSS文件即可实现版式的改变



提示：读者可以参见下载文件包内［/第3部分/第17章：旅游网站/site/index_ 2.html］文件。

由图17-108可以发现，只修改了很少的CSS规则，就可以改变版式布局的样式，这正是样式和结构分离的优点所在。不过，这样的方法也存在局限性，即当内容高度不确定的时候，可能会出现一些问题，如图17-109所示。

由图17-108可以发现，由于“游记精选”板块的内容增加，造成右侧photos层与club层的内容中间产生空白。而如果采取用 分为左右2列的方法，则不会出现这种情况。因此，如何结构化还要根据实际情况出发，灵活掌握。

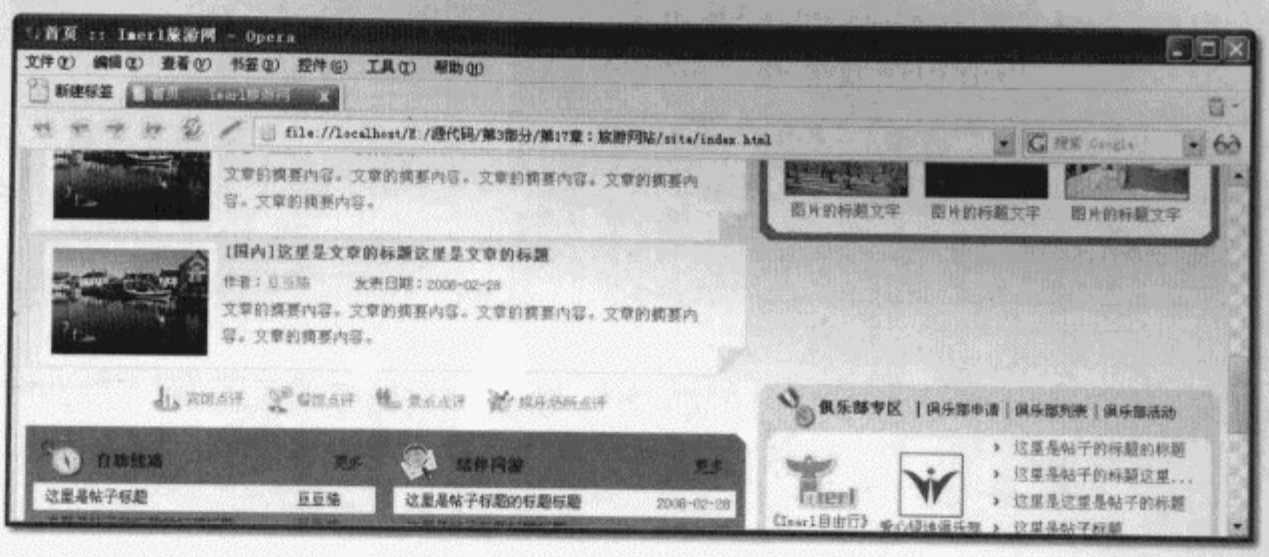
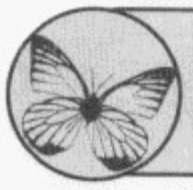


图17-109 内容高度变化造成空白



提示：关于版式布局，读者可参阅http://blog.html.it/layoutgala/（相同结构的40种CSS布局，英文）。

17.5 小结

结构要以内容为基础，而不是以表现为基础，虽然有时候为了实现表现需要添加一些额外的标签，但那也是权宜之策，当CSS 3制定成熟并被广泛支持之时，很多现在看来复杂的效果都可以简单地实现，例如圆角矩形、双色行背景等。同时，相同的内容结构化的结果也不是唯一的，对于不同的制作者，其考虑问题的角度不同，则结构就会不同。

网页的结构化和美化不是一次性完成的过程，中间可能要反复修改。也许一开始制定的结构有不合理的地方，会给后期制作造成困难，那么就需要修改结构。有时候很难实现的设计效果，也许可以换作简单些的效果，毕竟内容是访问者最关心的，而不是花哨的渐变背景或者阴影。只有多做多实践才能积累经验，少走弯路。

[General Information]
书名=别具光芒　　CSS 属性、浏览器兼容与网页布局
作者=
页数=416
出版社=
出版日期=
SS号=12085165
DX号=
URL=http://book.szdnet.org.cn/bookDetail.jsp?dxNumber=&d=20401835 0E153ABB9B73FEBC82B567 15

封面
书名
版权
前言
目录
第1部分　Ｗｅｂ标准
第1章　Ｗｅｂ标准概述
１．１　Ｗｅｂ标准概述
１．2表现与结构的分离
１．3易用性
１．4难点所在
１．４．１　ＤＩＶ＋ＣＳＳ不等于Ｗｅｂ标准
１．４．2正确使用ＸＨＴＭＬ标签
１．４．3表格本身并没有被抛弃
１．４．4善于利用ＣＳＳ
１．４．5不要滥用ｃｌａｓｓ
１．４．6应对浏览器
１．４．７．通过验证”并不是最终目的
１．5　ＳＥＯ简介
第2章　结构ＸＨＴＭＬ
２．1理解结构与表现
２．１．1内容
２．１．2结构（Ｓｔｒｕｃｔｕｒｅ）
２．１．3表现（Ｐｒｅｓｅｎｔａｔｉｏｎ）
２．１．4行为（Ｂｅｈａｖｉｏｒ）
２．2从ＨＴＭＬ到ＸＨＴＭＬ
２．２．１　ＨＴＭＬ简史
２．２．２　ＨＴＭＬ的缺点
２．２．3从ＨＴＭＬ到ＸＨＴＭＬ
２．3理解（Ｘ）ＨＴＭＬ标签的语义
２．３．1（Ｘ）ＨＴＭＬ与浏览器默认样式
２．３．2常用的ＸＨＴＭＬ标签和属性
２．３．3（Ｘ）ＨＴＭＬ各个元素对搜索引擎的权重比例
２．4网站整体制作基本流程
２．４．1总体流程与分工
２．４．2静态页面制作
第2部分　层叠样式表ＣＳＳ
第3章　ＣＳＳ入门
３．1　ＣＳＳ简介
３．１．1起源
３．１．2神奇的ＣＳＳ
３．１．3　ＣＳＳ与ＨＴＭＬ
３．１４　ＣＳＳ与浏览器
３．１５　ＣＳＳ　２．1与ＣＳＳ2
３．2　ＣＳＳ的使用方法
３．２．1行内式样式（ｌｉｎｌｉｎ　ｓｔｙｌｅ）
３．２．2嵌入式样式表（Ｅｍｂｅｄｄｅｄ　ｓｔｙｌｅ　ｓｈｅｅｔｓ）
３．２．3外部样式表（Ｌｉｎｋ　ｓｔｙｌｅ　ｓｈｅｅｔｓ）
３．２．4导入式样式表
３．２．5应用
３．２．6维护和组织样式表
３．3基本样式规则
３．３．1基本语法
３．３．2继承与层叠

3．3．3分组
3．3．4注释
3．3．5缩写
3．3．6注意事项
3．4元素类型
3．4．1替换和不可替换元素
3．4．2显示元素
3．5媒体类型
3．5．1指定媒体相关的样式表
3．5．2媒体组
第4章　文档结构与选择器
4．1文档结构
4．2　CSS选择器
4．21通配选择器（Universal selector）
4．2．2类型选择器（Type selectors
4．2．3　ID选择器（ID Selectors）
4．2．4类选择器（Class selectors）
4．2．5包含选择器（Descendant selectors）
4．2．6子元素选择器（Child selectors）
4．2．7相邻兄弟选择器（Adjacent sibling selectors）
4．2．8属性选择器（Attribute selectors）
4．3伪类与伪元素
4．3．1伪类（Pseudo-Classes）
4．3．2伪元素（Pseudo-Elements）
4．3．3注意
4．4指定值、计算值和实际值
4．5继承
4．5．1值的继承
4．5．2　“inherit”值
4．5．3继承的局限性
4．6层叠
4．6．1层叠的顺序
4．6．2特殊性的计算
4．6．3继承和特殊性
4．6．4重要性
4．6．5非CSS的表现类内容
4．7　CSS　3新增选择器前瞻
4．7．1更多的属性选择器
4．7．2普通兄弟选择器
4．7．3结构伪类（Structural　Pseudo-Classes）
4．7．4　UI元素伪类和伪元素
4．7．5其他伪类
4．8命名规范
4．9选择器综合运用
第5章　单位和值
5．1颜色＜Color＞
5．1．1颜色关键字
5．1．2　RGB颜色
5．1．3关键字transparent
5．1．4网页安全色（Web-safe　Colors　）
5．2整数值＜integer＞和实数值＜number＞

5．3长度＜length＞
5．3．1格式
5．3．2长度单位
5．3．3应用
5．4百分比＜percentage＞
5．5关键字
5．6字符串＜string＞
5．7　URL＋URN＝URI
5．8其他值
5．8．1记数器＜counter＞
5．8．2角度＜angle＞
5．8．3时间＜time＞
5．8．4频率＜frequency＞
5．9不支持的值的处理
第6章　字体
6．1字体集：font-family属性
6．1．1语法
6．1．2常用字体系列
6．2字体尺寸：font-size属性
6．2．1语法
6．2．2绝对尺寸
6．2．3相对尺寸
6．2．4百分比和em
6．2．5尺寸的继承与浏览器的显示
6．2．6分辨率与弹性设计
6．3字体磅值：font-weight属性
6．3．1语法
6．3．2继承
6．3．3浏览器显示原理
6．4字体样式：font-style属性
6．5字体变形：font-variant属性
6．6缩写的字体属性：font属性
6．6．1语法
6．6．2注意
6．6．3系统字体
6．7调整与拉伸
6．7．1字体调整：font-size-adiust属性
6．7．2字体伸展：font-stretch属性
6．8字体匹配原理
6．8．1字体的匹配步骤
6．8．2设定字体集的注意事项
6．8．3字体的选择
6．8．4@font-face规则
第7章　文本
7．1文本水平对齐：text-align属性
7．1．1语法
7．1．2适用于：块级元素
7．1．3继承
7．1．4应用：整体居中
7．2文本缩进：text-indent属性
7．2．1　语法
7．2．2　正值缩进
7．2．3　负值缩进
7．2．4　应用：隐藏单行文字
7．3　行高line-heightl属性

7．3．1语法
7．3．2内容区域、行内框和行框
7．3．3行高的计算与继承
7．3．4浏览器的差别与错误
7．3．5应用：单行文字在垂直方向居中
7．4垂直对齐：vertical-align属性
7．4．1语法
7．4．2属性值详解
7．4．3奇怪的IE
7．4．4文档类型与纯图片内容的垂直对齐
7．4．5单元格的垂直对齐
7．5单词间隔（word-spacing）和字母间隔（letter-spacing）
7．5．1单词间隔：word-spacing属性
7．5．2字母间隔：letter-spacing属性
7．5．3水平对齐的影响和继承
7．6文本转换：text-transform属性
7．7文本装饰：text-decoration属性
7．8空白：white-space属性
7．8．1语法
7．8．2属性值详解
7．8．3应用：显示不回行文本
7．9文本阴影：text-shadow属性
7．10文字方向direction和编码方式unicode-bidi
第8章 框模型
8．1 框模型（Box Model）
8．2包含块（Containing Block）
8．2．1视口（viewport）
8．2．2包含块
8．3宽度：width属性
8．3．1语法
8．3．2行内元素的宽度
8．3．3长度和百分比
8．4最大宽度（max-width）和最小宽度（min-width）
8．5高度：height属性
8．5．1语法
8．5．2行内元素的高度
8．6最大高度 （max-height）和最小高度（min-height）
8．7补白：padding属性
8．7．1 缩写属性： padding
8．7．2 补白宽度和高度
8．7．3 百分比值补白
8．8边框：border属性
8．8．1边框颜色
8．8．2边框宽度
8．8．3边框样式
8．8．4不同方向的边框属性缩写
8．8．5缩写属性border
8．8．6行内元素的边框
8．8．7应用：文字链接的装饰
8．9边距：margin属性
8．9．1水平方向的边距：margin-left属性和margin-right属性
8．9．2垂直方向的边距：margin-top属性和mar

ｇｉｎ-ｂｏｔｔｏm属性
８．９．３百分比值边距
８．９．４负值边距
８．９．５应用：元素水平居中
８．１０常规流向中的视觉格式化
８．１０．１块级元素的水平格式化
８．１０．２应用：宽度自适应的布局
８．１０．３块级元素的垂直格式化
８．１０．４应用：高度自适应浏览器窗口
８．１０．５行内元素的格式化
第９章　浮动、定位与视觉格式化模型
９．１视觉格式化模型控制框的生成
９．１．１块框的生成（ｂｌｏｃｋ　ｂｏｘ）
９．１．２行内框（ｉｎｌｉｎｅ　ｂｏｘ）
９．１．３插入框（ｒｕｎ-ｉｎ　ｂｏｘ）
９．２显示类型：ｄｉｓｐｌａｙ属性
９．２．１语法
９．２．２应用：显示或隐藏元素
９．３定位
９．３．１选择定位方式：ｐｏｓｉｔｉｏn属性
９．３．２设定框偏移：ｔｏp、ｒｉｇｈｔ、ｂｏｔｔｏm、ｌｅｆｔ属性
９．３．３相对定位
９．３．４绝对定位
９．３．５堆叠顺序：ｚ-ｉｎｄｅｘ属性
９．３．６　ＩＥ中的ｐｏｓｉｔｉｏｎ
９．３．７应用：显示提示内容
９．４浮动与清除
９．４．１设定浮动：ｆｌｏａｔ属性
９．４．２浮动元素的视觉格式化内容
９．４．３清除浮动：ｃｌｅａｒ属性
９．４．４　应用：３行３列布局设计
９．５　ｄｉｓｐｌａｙ、ｆｌｏａｔ和ｐｏｓｉｔｉｏｎ
９．６　溢出和剪切
９．６．１　溢出：Ｏｖｅｒｆｌｏｗ属性
９．６．２剪切：ｃｌｉｐ属性
９．６．３ｃｌｉｐ　与ｏｖｅｒｆｌｏｗ属性的关系
９．７可视性：ｖｉｓｉｂｉｌｉｔｙ属性
９．７．１属性值详解
９．７．２应用：显示及隐藏元素
第１０章　颜色与背景
１０．１颜色基础
１０．２前景色：ｃｏｌｏｒ属性
１０．２．１链接
１０．２．２边框
１０．２．３表单元素
１０．３背景
１０．３．１背景颜色：ｂａｃｋｇｒｏｕｎｄ-ｃｏｌｏｒ属性
１０．３．２背景图片：ｂａｃｋｇｒｏｕｎｄ-ｉｍａｇｅ属性
１０．３．３背景图片重复：ｂａｃｋｇｒｏｕｎｄ-ｒｅｐｅａｔ属性
１０．３．４背景图片附属：ｂａｃｋｇｒｏｕｎｄ-ａｔｔａｃｈｍｅｎｔ属性
１０．３．５背景图片定位：ｂａｃｋｇｒｏｕｎｄ-ｐｏｓｉｔｉｏｎ属性

10.3.6 缩写属性：background
10.3.7 <html>元素的背景
10.4 应用
10.4.1 灵活使用背景
10.4.2 模拟边框
10.4.3 简单的链接背景替换
10.4.4 导航菜单的滑动门效果
第11章 表格
11.1 表格的标签与属性
11.1.1 标签概览
11.1.2 （X）HTML属性
11.2 CSS的表格模型
11.2.1 表格模型概述
11.2.2 display属性
11.2.3 匿名表格对象
11.2.4 列
11.3 表格的视觉格式化
11.3.1 匿名框、标题框与表格框
11.3.2 标题<caption>的定位：caption-side属性
11.3.3 表格内容的视觉布局
11.3.4 表格的层和透明性
11.3.5 表格宽度算法：table-layout属性
11.3.6 表格高度
11.3.7 单元格内要容的对齐
11.4 单元格边框：border-collapse属性
11.4.1 分离的边框模型
11.4.2 重台的边框模型
11.4.3 边框样式
第12章 列表和生成的内容
12.1 列表
12.1.1 列表样式类型：list-style-type属性
12.1.2 列表样式图片：list-style-image属性
12.1.3 列表样式定位：list-style-position属性
12.1.4 列表样式缩写：list-style属性
12.1.5 浏览器对列表的表现与样式的继承
12.2 生成的内容
12.2.1 :before和:after伪元素
12.2.2 生成内容：content属性
12.2.3 自动记数和编号
第13章 用户界面
13.1 鼠标指针：cursor属性
13.1.1 关键字
13.1.2 图片鼠标指针
13.2 系统字体和颜色
13.2.1 系统字体
13.2.2 系统颜色
13.3 动态的外廓：outline属性
13.3.1 外廓与边框的区别
13.3.2 外廓宽度：outline-width属性
13.3.3 外廓样式：outline-style属性
13.3.4 外廓颜色：outline-color属性

13.3.5缩写：outline属性
13.3.6外廓与焦点
第14章 页面媒体
14.1页面媒体简介
14.2指定媒体类型
14.3页框：@page规则
14.3.1页边柜
14.3.2页面选择器
14.4分页
14.4.1元素前后分页：page-break-berfore和page-break-after属性
14.4.2元素内部分页：Page-break-inside属性
14.4.3元素内的分割：Orphans和windows属性
14.4.4分页的规则
14.5 CSS2中的属性
14.5.1页框尺寸：size属性
14.5.2裁切标记：marks属性
14.5.3使用命名的页：page属性
14.6显示器、打印机和投影
14.6.1设备特点
14.6.2设计要点
第15章 听觉样式表
15.1听觉（aural）类型与语音（speech）类型
15.1.1链接听觉样式的特点
15.1.2与听觉属性相关的值
15.2音量属性：volume属性
15.3发音：speak属性
15.4暂停：pause-before、pause-after和pause属性
15.5提示：cue-before、cue-after和cue属性
15.6混音：play-during属性
15.7空间：azimuth和elevation属性
15.8语音特征属性
15.9语音：speak-punctuation和speak-numeral属性
15.10叙述表头：speak-header属性
第16章 济览器与Hack
16.1浏览器简介
16.1.1浏览器的发展
16.1.2浏览器的解释引擎
16.1.3浏览器的工作模式
16.2 Windows IE
16.2.1 hasLayout属性
16.2.2条件注释
16.3常用的CSS Hack
16.3.1 CSSHack原理
16.3.2 CSS Hack不是必须的
16.3.3常用的CSSHack
16.4发现与解决问题
16.4.1 排查问题
16.4.2 常见的非Bug问题
16.4.3 Windows IE常见Bug
第3部分 结构化实例

第１７章　旅游网站
１７．１结构化
１７．１．１分析内容结构
１７．１．２基本结构
１７．１．３页首部分的结构化
１７．１．４中间部分的结构化
１７．１．５页脚部分的结构化
１７．２图片格式与网络基础知识
１７．２．１图片格式
１７．２．２图片与优化
１７．３　ＣＳＳ美化
１７．３．１整体分析
１７．３．２　ｈｅａｄｅｒ层
１７．３．３　ｌｏｇｏ层
１７．３．４　ｍａｉｎＮａｖ层
１７．３．５　ｌｏｇｉｎ层
１７．３．６　ｃｏｎｔｒｏｌＭｅｎｕ层
１７．３．７　ｍａｉｎ层
１７．３．８　ｔｒａｖｅｌｓ层
１７．３．９　ｈｏｔ层
１７．３．１０　ａｄｌ层
１７．３．１１　ｐｈｏｔｏｓ层
１７．３．１２　ｆｏｒｕｍＬｉｓｔ层
１７．３．１３　ｆｏｒｕｍＨｏｔ层
１７．３．１４　ｃｌｕｂ层
１７．３．１５　ｖｏｔｅ层和ｃｏｍｍｕｎｉｔｙ层
１７．３．１６　ｆｏｏｔｅｒ层
１７．４版式与结构
１７．５小结