

HTML 教程

第一章 HTML概述与基本结构	2
一、HTML的概述.....	2
二、 HTML的基本结构:	2
三、 HTML的标签与属性:	3
第二章 HTML主体标签及属性、颜色的设定.....	4
一、 html的主体标签<body>	4
二、 颜色的设定.....	5
第三章 文字版面的编辑.....	6
第四章 建立列表	15
第五章 图像的处理.....	21
第六章 建立超链接.....	26
第七章 TABLE表格	35
7-1 定义表格的基本语法	35
7-2 表格<table>标签属性	36
7-2 (1) 表格<table>标签的常用属性	36
7-2 (2) 设置分隔线的显示状态 rules.....	37
7-2 (3) 表格的边框显示状态 frame	38
7-3 表格行的设定	39
7-4 单元格的设定	40
7-5 设定跨多行多列单元格	42
7-6 表格的分组	43
7-6-1 表格的行分组<thead>/<tbody>/<tfoot>	43
7-6-2 表格按列分组<colgroup>	44
7-6-3 表格的行列分组	45
7-7 表格的标题标签<caption>	45
7-8 表格的嵌套	46
第八章网页的动态、多媒体效果.....	49
第九章 多视窗口框架	53
第十章表单的设计.....	59

第一章 HTML概述与基本结构

一、HTML 的概述

HTML的英语意思是: Hypertext Marked Language, 即超文本[标记语言](#), 是一种用来制作超本文档的简单标记语言。超文本传输协议规定了浏览器在运行 HTML 文档时所遵循的规则和进行的操作。HTTP 协议的制定使浏览器在运行超文本时有了统一的规则 and 标准。用HTML编写的超本文档称为HTML文档, 它能独立于各种操作系统平台, 自 1990 年以来HTML就一直被用作 W W W (是World Wide Web的缩写, 也可简写WEB、中文叫做万维网) 的信息表示语言, 使用HTML语言描述的文件, 需要通过WEB浏览器[显示](#)出效果。

所谓超文本, 是因为它可以加入图片、声音、动画、影视等内容, 事实上每一个 HTML 文档都是一种静态的网页文件, 这个文件里面包含了 HTML 指令代码, 这些指令代码并不是一种程序语言, 它只是一种排版网页中资料显示位置的标记结构语言, 易学易懂, 非常简单。HTML 的普遍应用就是带来了超文本的技术----通过单击鼠标从一个主题跳转到另一个主题, 从一个页面跳转到另一个页面与世界各地主机的文件链接。直接获取相关的主题。如下所示:

通过 HTML 可以表现出丰富多彩的设计风格:

图片调用:

文字格式: 文字

通过 HTML 可以实现页面之间的跳转:

页面跳转:

通过 HTML 可以展现多媒体的效果:

音频: <EMBED SRC="音乐地址" AUTOSTART=true>

视频: <EMBED SRC="视频地址" AUTOSTART=true>

从上面我们可以看到HTML超文本文件时需要用到的一些[标签](#)。在HTML中每个用来作标签的符号都是一条命令、它告诉浏览器如何显示文本。这些标签均由"<"和">"符号以及一个字符串组成。而浏览器的功能是对这些标记进行解释, 显示出文字、图像、动画、播放声音。这些标签符号用"<标签名字 属性>"来表示。

HTML 只是一个纯文本文件。创建一个 HTML 文档, 只需要两个工具, 一个是 HTML 编辑器, 一个 WEB 浏览器。HTML 编辑器是用于生成和保存 HTML 文档的应用程序。WEB 浏览器是用来打开 WEB 网页文件, 提供给我们查看 WEB 资源的客户端程序。

二、HTML 的基本结构:

一个 HTML 文档是由一系列的元素和标签组成。元素名不区分大小写。HTML 用标签来规定元素的属性和它在文件中的位置,

HTML 超本文档分[文档头](#)和[文档体](#)两部分, 在文档头里, 对这个文档进行了一些必要的定义, 文档体中才是要显示的各种文档信息。

下面是一个最基本的html文档的代码: [1-1.html](#)

<HTML> ----- 开始标签

<HEAD> -----| 头部标签

```

<TITLE> 一个简单的 HTML 示例 </TITLE>
</HEAD> -----

<BODY> ----- | 文件主体
<CENTER>          |
<H1>欢迎光临我的主页</H1>      |
<BR>                |
<HR>
<FONT SIZE= 7 COLOR= red>      |
这是我第一次做主页            |
</FONT>                |
</CENTER>              |
</BODY> -----

</HTML> -----  结尾标签

```

<HTML></HTML>在文档的最外层， 文档中的所有文本和 html 标签都包含在其中， 它表示该文档是以超文本标识语言（HTML）编写的。事实上，现在常用的 Web 浏览器都可以自动识别 HTML 文档，并不要求有 <html>标签，也不对该标签进行任何操作，但是为了使 HTML 文档能够适应不断变化的 Web 浏览器，还是应该养成不省略这对标签的良好习惯。

<HEAD></HEAD>是HTML文档的头部标签，在浏览器窗口中，头部信息是不被显示在正文中的,在此标签中可以插入其它标记，用以说明文件的[标题](#)和整个文件的一些公共属性。若不需头部信息则可省略此标记，良好的习惯是不省略。

<title>和</title>是嵌套在<HEAD>头部标签中的，标签之间的文本是文档标题，它被显示在浏览器窗口的标题栏。

<BODY> </BODY>标记一般不省略，标签之间的文本是正文，是在浏览器要显示的页面内容。

上面的这几对标签在文档中都是唯一的，HEAD 标签和 BODY 标签是嵌套在 HTML 标签中的。

三、 HTML的标签与属性：

对于刚刚接触超文本的朋友，遇到的最大的障碍就是一些用“<”和 “>”括起来的句子，我们称它为标签，是用来分割和标签文本的元素，以形成文本的布局、文字的格式及五彩缤纷的画面。标签通过指定某块信息为段落或标题等来标识文档某个部件。属性是标志里的参数的选项，

HTML 的标签分单标签和成对标签两种。成对标签是由首标签<标签名> 和尾标签</标签名>组成的，成对标签的作用域只作用于这对标签中 的文档。单独标签的格式<标签名>，单独标签在相应的位置插入元素就可以了，大多数标签都有自己的一些属性，属性要写在始标签内，属性用于进一步改变显示的效果，各属性之间无先后次序，属性是可选的，属性也可以省略而采用默认值;其格式如下：

```
<标签名字 属性 1 属性 2 属性 3 ... >内容</标签名字>
```

作为一般的原则，大多数属性值不用加双引号。但是包括空格、%号，# 号等特殊字符的属性值必须加入双引号。为了更好的习惯，提倡全部对属性值加双引号。如：

```
<font color="ff00ff" face="宋体" size="30">字体设置</font>
```

注意事项：输入始标签时，一定不要在“<”与标签名之间输入多余的空格，也不能在中文输入法状态下输入这些标签及属性，否则浏览器将不能正确的识别括号中的标志命令，从而无法正确的显示你的信息。

第二章 HTML主体标签及属性、颜色的设定

为了使你的网页绚丽多姿，吸引更多的浏览者阅读，可以给网页添加更多的标签及属性，这是为了对网页中的元素进行修饰、布局。下面就来逐一介绍这些标签。

一、html 的主体标签<body>

在<body>和</body>中放置的是页面中所有的内容，如图片、文字、表格、表单、超链接等设置。

<body>标签有自己的属性，设置 <body>标签内的属性，可控制整个页面的显示方式。

<body>标签的属性

属 性	描 述
link	设定页面默认的连接颜色
alink	设定鼠标正在单击时的连接颜色
vlink	设定访问后连接文字的颜色
background	设定页面背景图像
bgcolor	设定页面背景颜色
leftmargin	设定页面的左边距
topmargin	设定页面的上边距
bgproperties	设定页面背景图像为固定，不随页面的滚动而滚动
text	设定页面文字的颜色

格式：

```
<body text="#000000" link="#000000" alink="#000000" vlink="#000000"
background="gifnam.gif" bgcolor="#000000" leftmargin=3 topmargin=2
bgproperties="fixed">
```

实例：[2-1.html](#)

```
<html>
<head>
<title>bady 的属性实例</title>
</head>
<body bgcolor="#FFFE7" text="#ff0000" link="#3300FF" alink="#FF00FF"
vlink="#9900FF">
<center>
<h2>设定不同的连接颜色</h2>
测试 body 标签<p>
<a href="http://www.baidu.com/">默认的连接颜色</a>
<p>
<a href="http://www.sina.com.cn">正在按下的连接颜色,</a>
<p>
<a href="http://www.sohu.com/">访问过后的连接颜色,</a>
```

```
<P>
<a href="#" onClick="window.history.back()">返回</a>
</conter>
</body>
</html>
```

实例说明：

<body>的属性设定了页面的背景颜色，文字的颜色，链接的颜色为#3300ff，单击的连接颜色为#ff00ff，单击过后的颜色为#9900ff。

Body 里面的是页面中的链接标签，对于属性可根据页面的效果来定，用那个属性就设定那个属性。对于上面的属性在后面的章节中还会介绍，在这里就不逐一引用了，我们的学习目的主要是掌握标签及属性的使用方法。

二、 颜色的设定

颜色值是一个关键字或一个 RGB 格式的数字。在网页中用得很多，在此就先介绍一下。

颜色是由 "red" "green" "blue " 三原色组合而成的，在 HTML 中对颜色的定义是用十六进位的，对于三原色 HTML 分别给予两个十六进位去定义，也就是每个原色可有 256 种彩度，故此三原色可混合成 16777216 种颜色。

例如： 白色的组成是 red=ff, green=ff, blue= ff,RGB 值即为 ffffff
红色的组成是 red=ff, green= 00, blue= 00, RGB 值即为 ff0000
绿色的组成是 red=00, green=ff, blue= 00, RGB 值即为 00ff00
蓝色的组成是 red=00, green= 00, blue= ff, RGB 值即为 0000ff
黑色的组成是 red=00, green=00, blue=00, RGB 值即为 000000

应用时常在每个 RGB 值之前加上" #" 符号，如：bgcolor="#336699" 用英文名字表示颜色时直接写名字。如 bgcolor=green

RGB 颜色可以有四种表达形式：

- **#rrggb** (如, #00cc00)
- **#rgb** (如, #0c0)
- **rgb(x,x,x)** x 是一个介乎 0 到 255 之间的整数 (如, rgb(0,204,0))
- **rgb(y%,y%,y%)** y 是一个介乎 0.0 到 100.0 之间的整数 (如, rgb(0%,80%,0%))

Windows VGA(视频图像阵列)形成了 16 各关键字: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, , and yellow

下面是部分颜色的 RGB 代码图表:

第三章 文字版面的编辑

3-1 换行标签

换行标签是个单标签，也叫空标签，不包含任何内容，在html文件中的任何位置只要使用了
标签，当文件显示在浏览器中时，该标签之后的内容将显示下一行。

请看下面的例子：[3-1.html](#)

```
<html>
```

```
<head>
```

```
<title>无换行示例</title>
```

```
</head>
```

```
<body>
```

无换行标记：登鹳雀楼 白日依山尽，黄河入海流。欲穷千里目，更上一层楼。

有换行标记：
登鹳雀楼
白日依山尽，
黄河入海流。
欲穷千里目，
更上一层楼。

```
</body>
```

```
</html>
```

3-2 换段落标签<p>及属性：

由<p>标签所标识的文字，代表同一个段落的文字。不同段落间的间距等于连续加了两个换行符，也就是要隔一行空白行，用以区别文字的不同段落。它可以单独使用，也可以成对使用。单独使用时，下一个<P>的开始就意味着上一个<P>的结束。良好的习惯是成对使用。

格式：

```
<P>
```

```
<P ALIGN= 参数>
```

其中,ALIGN 是<p>标签的属性,属性有三个参数 left,center,right.这三个参数设置段落文字的左,中,右位置的对齐方式。

实例：[3-2.html](#)

```
<html>
```

```
<head>
```

```
<title>测试分段控制标签</title>
```

```
</head>
```

```
<body>
```

<p>花儿什么也没有。它们只有凋谢在风中的轻微、凄楚而又无奈的吟怨，就像那受到了致命伤害的秋雁，悲哀无助地发出一声声垂死的鸣叫。</p>

<p align="right">或许，这便是花儿那短暂一生最凄凉、最伤感的归宿。</p>

<p align=center>而美丽苦短的花期</p>

<p align="left">却是那最后悲伤的秋风挽歌中的瞬间插曲。</p>

```
</body>
```

```
</html>
```

3-3 原样显示文字标签<pre>

要保留原始文字排版的格式,就可以通过<pre>标签来实现,方法是把制作好的文字排版内容前后分别加上始标签<pre>和尾标签</pre>.

实例: [3-3html](#)

```
<HTML>
<HEAD>
<TITLE>原样显示文字标签</TITLE>
</HEAD>
<BODY>
<PRE>
    白日依山尽，
        黄河入海流。
            欲穷千里目，
                更上一层楼。
</PRE>
</BODY>
</HTML>
```

3-4 居中对齐标签<center>

文本在页面中使用<center>标签进行居中显示,<center>是成对标签,在需要居中的内容部分开头处加<center>,结尾处加</center>

实例: [3-4html](#)

```
<HTML>
<HEAD>
<TITLE>居中对齐标签</TITLE>
</HEAD>
<BODY>
<CENTER>
    《送孟浩然之廣陵》<p>
    故人西辭黃鶴樓，
    煙花三月下揚州。
    孤帆遠影碧山盡，
    惟見長江天際流。
</CENTER>
</BODY>
</HTML>
```

3-5 引文标签（缩排标签）<blockquote>

<blockquote>标签可以用来建立一个引文,他特别适合较长文本的引用,引文显示时将会自动右移,左边空出几个格,加以区别。

实例: [3-5html](#)

```
<HTML>
<HEAD>
```

<TITLE>引文标签</TITLE>

</HEAD>

<BODY>

春

<PRE>

<BLOCKQUOTE>

盼望着，盼望着，东风来了，春天脚步近了。

一切都像刚睡醒的样子，欣欣然张开了眼。山朗润起来了，水涨起来了，太阳的脸红起来了。

小草偷偷地从土里钻出来，嫩嫩的，绿绿的。园子里，田野里，瞧去一大片一大片满是的。坐着，躺着，打两个滚，踢几脚球，赛几趟跑，捉几

回迷藏。风轻悄悄的，草软绵绵的。

</BLOCKQUOTE>

<BLOCKQUOTE><BLOCKQUOTE>

桃树、杏树、梨树，你不让我，我不让你，都开满了花赶趟儿。红的像火，粉的像霞，白的像雪。花里带着甜味儿；闭了眼，村上仿佛已

经满是桃儿、杏儿、梨儿。花下成千成百的蜜蜂嗡嗡地闹着，大小的蝴蝶飞来飞去。野花遍地是：杂样儿，有名字的，没名字的，散在草

丛里像眼睛，像星星，还眨呀眨的。

“吹面不寒杨柳风”，不错的，像母亲的手抚摸着你。风里带来些新翻的泥土的气息，混着青草味儿，还有各种花的香，都在微微润湿的

空气里酝酿。鸟儿将巢安在繁花嫩叶当中，高兴起来了，呼朋引伴地卖弄清脆的喉咙，唱出宛转的曲子，跟轻风流水应和着。牛背上牧童

的短笛，这时候也成天嘹亮地响着。

雨是最寻常的，一下就是三两天。可别恼。看，像牛毛，像花针，像细丝，密密地斜织着，人家屋顶上全笼着一层薄烟。树叶儿却绿得发

亮。小草儿也青得逼你的眼。傍晚时候，上灯了，一点点黄晕的光，烘托出一片安静而和平的夜。在乡下，小路上，石桥边，有撑着伞慢

慢走着的人；地里还有工作的农民，披着蓑戴着笠。他们的房屋，稀稀疏疏的，在雨里静默着。

</BLOCKQUOTE></BLOCKQUOTE>

<BLOCKQUOTE><BLOCKQUOTE><BLOCKQUOTE>

天上风筝渐渐多了，地上孩子也多了。城里乡下，家家户户，老老小小，也都赶趟儿似的，一个个都出来了。舒活舒活筋骨，抖擞

抖擞精神，各做各的一份事儿去。

“一年之计在于春”，刚起头儿，有的是工夫，有的是希望。

春天像刚落地的娃娃，从头到脚都是新的，它生长着。

春天像小姑娘，花枝招展的，笑着，走着。

春天像健壮的青年，有铁一般的胳膊和腰脚，他领着我们上前去。

</BLOCKQUOTE></BLOCKQUOTE></BLOCKQUOTE>

</PRE>

</BODY>

</HTML>

3-6 水平分隔线标签<hr>

<hr>标签是单独使用的标签，是水平线标签，用于段落与段落之间的分隔，使文档结构清晰明了，使

文字的编排更整齐。通过设置<hr>标签的属性值，可以控制水平分隔线的样式。

<hr>标签的属性

属性	参数	功能	单位	默认值
size		设置水平分隔线的粗细	pixel(像素)	2
width		设置水平分隔线的宽度	pixel(像素)、%	100%
align	left center right	设置水平分隔线的对齐方式		center
color		设置水平分隔线的颜色		black
noshade		取消水平分隔线的 3d 阴影		

实例：[3-6html](#)

```
<HTML>
<HEAD>
<TITLE>测试水平分隔线标签</TITLE>
</HEAD>
<BODY>
<CENTER>
春 晓
<HR >
春眠不觉晓，
<hr size="6">
处处闻啼鸟。
<hr width="40%">
夜来风雨声，
<hr width="60" align="left">
花落知多少？
<hr size="6" width="30%" align="center" noshade color=red >
</CENTER>
</BODY>
</HTML>
```

3-7 署名标签<address>

<address>署名标签一般用于说明这个网页是由谁或是由哪个公司编写的，以及其它相关信息。在<address></address>标签之间的文字显示效果是斜体字。

实例：[3-7html](#)

```
<HTML>
<HEAD>
<TITLE>署名标签</TITLE>
</HEAD>
<BODY>
<CENTER>
乐 游 原
<HR WIDTH="50%" SIZE="5" COLOR="FFCC00" ALIGN=CENTER>
<PRE>
向晚意不适，
```

```
驱车登古原。  
夕阳无限好，  
只是近黄昏。  
</PRE>  
<HR WIDTH="50%" SIZE="5" COLOR="FFCC00" ALIGN=Center>  
<ADDRESS>  
[唐] 李商隐  
</ADDRESS>  
</BODY>  
</HTML>
```

3-8 特殊字符

在 HTML 文档中，有些字符没办法直接显示出来，例如?。使用特殊字符可以将键盘上没有的字符表达出来，而有些 HTML 文档的特殊字符在键盘上虽然可以得到，但浏览器在解析 HTML 文当时会报错，例如 "<"等，为防止代码混淆，必须用一些代码来表示它们。

HTML 几种常见特殊字符及其代码表

特殊或专用字符	字 符 代 码	特殊或专用字符	字 符 代 码
<	<	?	©
>	>	×	×
&	&	?	®
"	"	空格	

实例：[3-8.html](#)

```
<HTML>  
<HEAD>  
<TITLE>特殊字符</TITLE>  
</HEAD>  
<BODY>  
<CENTER>  
&lt;赋得古原草送别&gt;  
<HR WIDTH="49%" SIZE="5" ALIGN=Center NOSHADE>  
<PRE>  
离离原上草，  
一岁一枯荣。  
野火烧不尽，  
春风吹又生。  
</PRE>
```

```
<HR WIDTH="49%" SIZE="5" ALIGN=CENTER NOSHADE>
<ADDRESS>
白居易 &copy;
</ADDRESS>
</BODY>
</HTML>
```

3-9 注释标签

在 HTML 文档中可以加入相关的注释标记，便于查找和记忆有关的文件内容和标识，这些注释内容并不会在浏览器中显示出来。

注释标签的格式有如下：

```
<!--注释的内容-->
```

实例: [3-9html](#)

```
<HTML>
<HEAD>
<TITLE>注释标签</TITLE>
</HEAD>
<BODY>
<!--body 标签是主体内容-->
<CENTER>
&lt;赋得古原草送别&gt;
<HR WIDTH="49%" SIZE="5" ALIGN=CENTER NOSHADE>
<PRE>
<! pre 代表原样显示排版格式>
离离原上草，
一岁一枯荣。
野火烧不尽，
春风吹又生。
</PRE>
<HR WIDTH="49%" SIZE="5" ALIGN=CENTER NOSHADE>
<ADDRESS>
白居易 &copy;
</ADDRESS>
</BODY>
</HTML>
```

3-10 字体属性

3-10-1 标题文字标签<h1>

<h1>标签用于设置网页中的标题文字，被设置的文字将以黑体或粗体的方式显示在网页中。

标题标签的格式：<h1 align=参数>标题内容</h1>

说明：<h1>标签是成对出现的，<h1>标签共分为六级，在<h1>...</h1>之间的文字就是第一级

标题，是最大最粗的标题；<h6>...</h6>之间的文字是最后一级，是最小最细的标题文字。align 属性用于设置标题的对齐方式，其参数为 left（左），enter（中），right（右）。<hn>标签本身具有换行的作用，标题总是从新的一行开始。

实例 [3-10-1html](#)

```
<HTML>
<HEAD>
<TITLE>设定各级标题</TITLE>
</HEAD>
<BODY>
<H1 ALIGN="CENTER">一级标题。</H1>
<H2>二级标题。</H2>
<H3>三级标题。</H3>
<H4>四级标题。</H4>
<H5 ALIGN="RIGHT">五级标题。</H5>
<H6 ALIGN="LEFT">六级标题。</H6>
</BODY>
</HTML>
```

3-10-2 文字格式控制标签

标签用于控制文字的字体，大小和颜色。控制方式是利用属性设置得以实现的。

FONT 标签的属性

属性	使用功能	默认值
face	设置文字使用的字体	宋体
size	设置文字的大小	3
color	设置文字的颜色	黑色

格式：文字

说明：如果用户的系统中没有 face 属性所指的字体，则将使用默认字体。size 属性的取值为 1~7。也可以用"+"或"-"来设定字号的相对值。color 属性的值为：rgb 颜色"#nnnnnn"或颜色的名称

实例： [3-10-2html](#)

```
<html>
<head>
<title>控制文字的格式</title>
</head>
<body>
<center>
<font face=黑体 size=6 color="red" >盼望着，盼望着，东风来了，春天脚步近了。</font> <p>
<font face=隶书 size=+3 color="green">
一切都像刚睡醒的样子，欣然张开了眼。<p>山朗润起来了，水涨起来了，太阳的脸红起来了。
</font><p>
<font face=楷体 size=4 color="#ff00ff">
小草偷偷地从土里钻出来，嫩嫩的，绿绿的。<p>园子里，田野里，瞧去一大片一大片满是的。<p>坐着，
躺着，打两个滚，踢几脚球，赛几趟跑，捉几回迷藏。<p>风轻悄悄的，草软绵绵的。
</font>
</center>
```

</body>

</html>

3-10-3 特定文字样式标签

在有关文字的显示中，常常会使用一些特殊的字形或字体来强调、突出、区别以达到提示的效果。在html中用于这种功能的标签可以分为两类，物理类型和逻辑类型。

1、物理类型

(1) 粗体标签

放在与标签之间的文字将以粗体方式显示。

(2) 斜体标签<i>

放在<i>与</i>标签之间的文字将以斜体方式显示。

(3) 下划线标签<u>

放在<u>与</u>标签之间的文字将以下划线方式显示。

实例[3-10-31html](#)

```
<html>
```

```
<head>
```

```
<title>字体的物理类型</title>
```

```
</head>
```

```
<body>
```

```
<center>
```

```
<font color="#FF0000" size="+2"><b>这些文字是粗体的</b></font><br><br>
```

```
<i>这些文字是斜体的</i> <br><br>
```

```
<u>这些文字带有下划线</u>
```

```
</center>
```

```
</body>
```

```
</html>
```

2、逻辑类型

逻辑类型是使用一些标签来改变字体的形态和式样，以便产生一些浏览者习惯的或约定的显示效果，常用的逻辑类型标签有八种，放在标签之间的文字受其控制。下面请看常用逻辑标签的实例

实例[3-10-32html](#)

```
<html>
```

```
<head>
```

```
<title>字体的逻辑类型</title>
```

```
</head>
```

```
<body>
```

```
<pre>
```

em 标签: 用于强调的文本，一般显示为斜体字

strong 标签: 用于特别强调的文本，显示为粗体字

cite 标签: <cite>用于引证和举例，通常是斜体字</cite>

code 标签: <code>用来指出这是一组代码</code>

small 标签: <small>规定文本以小号字显示</small>

big 标签: <big>规定文本以大号字显示</big>

samp 标签: <samp>显示一段计算机常用的字体, 即宽度相等的字体</samp>

kbd 标签: <kbd>由用户输入文本, 通常显示为较粗的宽体字</kbd>

var 标签: <var>用来表示变量, 通常显示为斜体字</var>

dfn 标签: <dfn>表示一个定义或说明, 通常显示为斜体字</dfn>

sup 标签: 12²=144

sub 标签: 硫酸亚铁分子式是 Fe₂SO₄

</pre>

</body>

</html>

第四章 建立列表

在 html 页面中，合理的使用列表标签可以起到提纲和格式排序文件的作用。

列表分为两类，一是无序列表，一是有序列表，无序列表就是项目各条列间并无顺序关系，纯粹只是利用条列来呈现资料而已，此种无序标签，在各条列前面均有一符号以示区隔。而有序条列就是指各条列之间是有顺序的，比如从 1、2、3...一直延伸下去。

列表的主要标签

标 签	描 述
	无序列表
	有序列表
<DIR>	目录列表
<DL>	定义列表
<MENU>	菜单列表
<DL>/<DT>/<DD>	定义列表的标记
	列表项目的标记

4-1 无序列表

无序列表使用的一对标签是 ，无序列表指没有进行编号的列表，每一个列表项前使用。的属性 type 有三个选项,这三个选项都必须小写：

disc 实心圆
circle 空心圆
square 小方块

如果不使用其项目的属性值,即默认情况下的会加"实心圆"。

格式 1:

```
<UL>
    <LI>第一项
    <LI>第二项
    <LI>第三项
</UL>
```

格式 2

```
<ul>
    <li type=disc>第一项
    <li type=circle>第二项
    <li type=square>第三项
</ul>
```

示例 [5-1.html](#)

```
<html>
<head>
```

```
<title>无序列表</title>
</head>
<body>
<ul>
    <li>默认的无序列表加"实心圆"
    <li>默认的无序列表"实心圆"
    <li>默认的无序列表"实心圆"
</ul>
<ul>
    <li type=square>无序列表 square 加方块
    <li type=square>无序列表 square 加方块
    <li type=square>无序列表 square 加方块
</ul>
<ul>
    <li type=circle>无序列表 circle 加空心圆
    <li type=circle>无序列表 circle 加空心圆
    <li type=circle>无序列表 circle 加空心圆
</ul>
</body>
</html>
```

4-2 有序列表

有序列表和无序列表的使用格式基本相同，它使用标签，每一个列表项前使用。列表的结果是带有前后顺序之分的编号。如果插入和删除一个列表项，编号会自动调整。

顺序编号的设置是由的两个属性 type 和 start 来完成的。start=编号开始的数字，如 start=2 则编号从 2 开始，如果从 1 开始可以省略，或是在标签中设定 value="n"改变列表行项目的特定编号，例如<li value="7">。type=用于编号的数字,字母等的类型，如 type=a，则编号用英文字母。为了使用这些属性，把它们放在或的的初始标签中。

有序列表 type 的属性

type 类 型	描 述
type=1	表示列表项目用数字标号（1,2,3...）
type=A	表示列表项目用大写字母标号（A,B,C...）
type=a	表示列表项目用小写字母标号（a,b,c...）
type=I	表示列表项目用大写罗马数字标号（I,II,III...）
type=i	表示列表项目用小写罗马数字标号（i,ii,iii...）

格式 1

```
<ol type=编号类型 start=value>
    <li>第 1 项
    <li>第 2 项
    <li>第 3 项
</ol>
```


格式 2

```
<ol>
  <li>第 1 项
  <li>第 2 项
  <li>第 3 项
</ol>
```

实例: [5-2.html](#)

```
<html>
<head>
<title>有序列表</title>
</head>
<body>
  <ol>
    <li>默认的有序列表
    <li>默认的有序列表
    <li>默认的有序列表
  </ol>
  <ol type=a start=5>
    <li>第 1 项
    <li>第 2 项
    <li>第 3 项
    <li value= 20>第 4 项
  </ol>
  <ol type= I start=2>
    <li>第 1 项
    <li>第 2 项
    <li>第 3 项
  </ol>
</body>
</html>
```

4-3 嵌套列表

将一个列表嵌入到另一个列表中，作为另一个列表的一部分，叫嵌套列表。无论是有序列表和无序列表的嵌套，浏览器都可以自动地分成排列。

实例: [5-3.html](#)

```
<HTML>
<HEAD>
<TITLE>嵌套列表</TITLE>
</HEAD>
<BODY>
<h3>目录</h3>
<ol type=a>
  <li>这是以序号类型 a 开头第一行内容</li>
```

```

<li>这是以序号类型 a 开头第二行内容</li>
  <ol type=A start=3>
    <li>这是以序号 A 类型 C 开头第一行内容</li>
    <li>这是以序号 A 类型 C 开头第二行内容</li>
      <ol type=1 start=51>
        <li>这是以序号数字 51 开头第一行内容</li>
        <li>这是以序号数字 51 开头第二行内容</li>
        <li>这是以序号数字 51 开头第三行内容</li>
      </ol>
    <li>这是以序号 A 类型 C 开头第三行内容</li>
  </ol>
<li>这是以序号类型 a 开头第三行内容</li>
</ol>
<HR>
</BODY>
</HTML>

```

4-4 定义列表的标记<DL>/<DT>/<DD>

定义列表的标记也叫描述性列表，定义列表默认为两个层次，第一层为列表项标签<DT>，第二层为注释项标签<DD>。<DT>和<DD>标签通常是成对使用的。也可以一个列表项对应于几个解释项，这种方式很少用。<DD>默认的注释是显示在另一行中，当使用<dl compact="compact">后，注释项和列表项将显示在同一行。其格式为：

```

<dl>
  <dt>第 1 项 <dd>注释 1
  <dt>第 2 项 <dd>注释 2
  <dt>第 3 项 <dd>注释 3
</dl>

```

实例：[5-4.html](#)

```

<html>
<head>
<title>定义性列表</title>
</head>
<body>
定义性列表<P>
<dl>
  <dt> WWW: <dd> WWW 是(World wide web)的缩写，也可简写 web;
  <dt> WWW: <dd> WWW 又叫万维网;
  <dt> WWW: <dd> internet 提供的最常用的服务是 WWW;
</dl>
</body>
</html>

```

4-5 目录列表<DIR>和菜单列表<MENU>

<dir>为目录列表标签,<menu>为菜单列表标签,它们的格式和无序列表是一样的,例如:

格式 1:

```
<dir>
  <li>第一项
  <li>第二项
  <li>第三项
</dir>
```

格式 2

```
<menu>
  <li type=disc>第一项
  <li type=circle>第二项
  <li type=square>第三项
</menu>
```

实例: [5-5.HTML](#)

```
<html>
<head>
<title>无序列表</title>
</head>
<body>
<ul>
  <li>默认的无序列表加"实心圆"
  <li>默认的无序列表"实心圆"
  <li>默认的无序列表"实心圆"
</ul>
<dir>
  <li>默认的目录列表加"实心圆"
  <li>默认的目录列表"实心圆"
  <li>默认的目录列表"实心圆"
</dir>
<menu>
  <li>默认的菜单列表加"实心圆"
  <li>默认的菜单列表"实心圆"
  <li>默认的菜单列表"实心圆"
</menu>
<dir>
  <li type=square>目录列表 square 加方块
  <li type=square>目录列表 square 加方块
  <li type=square>目录列表 square 加方块
</dir>
<menu>
  <li type=circle>菜单列表 circle 加空心圆
  <li type=circle>菜单列表 circle 加空心圆
```

```
    <li type=circle>菜单列表 circle 加空心园
</menu>
</body>
</html>
```

第五章 图像的处理

图像可以使 html 页面美观生动且富有生机。浏览器可以显示的图像格式有 jpeg, bmp, gif。其中 bmp 文件存储空间大，传输慢，不提倡用，常用的 jpeg 和 gif 格式的图像相比较，jpeg 图像支持数百万种颜色，即使在传输过程中丢失数据，也不会质量上有明显的不同，占位空间比 gif 大，gif 图像仅包括 265 色彩，虽然质量上没有 jpeg 图像高，但占位储存空间小，下载速度最快、支持动画效果及背景色透明等特点。因此使用图像美画页面可视情况而决定使用那种格式。

5-1 背景图像的设置

在网页中除了可以用单一的颜色做背景外，还可用图像设置背景。

设置背景图像的格式：

```
<body background= "image-url">
```

其中 "image-url" 指图像的位置。

实例：[6-1.html](#)

```
<html>
<head>
<title>设置背景图像</title>
</head>
<body background="imge/11.gif">
<center>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p><font color="#006600" size="+6">盼望着，盼望着，东风来了，春天脚步近了.</font>
</p>
</center>
</body>
</html>
```

5-2 网页中插入图片[标签](#)

网页中插入图片用单标签，当浏览器读取到标签时，就会显示此标签所设定的图像。如果要对插入的图片进行修饰时，仅仅用这一个属性是不够的，还要配合其它属性来完成。

插入图片标签的属性

属 性	描 述
src	图像的 url 的路径
alt	提示文字
width	宽度 通常只设为图片的真实大小以免失真,改变图片大小最好用图像工具.
height	高度 通常只设为图片的真实大小以免失真,改变图片大小最好用图像工具.
dynsrc	avi 文件的 url 的路径
loop	设定 avi 文件循环播放的次数

loopdelay	设定 avi 文件循环播放延迟
start	设定 avi 文件的播放方式
lowsrc	设定低分辨率图片,若所加入的是一张很大的图片,可先显示图片。
usemap	映像 地图
align	图像和文字之间的排列属性
border	边框
hspace	水平间距
valign	垂直间距

 的格式及一般属性设定:

```

```

5-2-1 插入普通图片

实例: [5-2-1.html](#)

```
<html>
<head>
<title>普通插入图片</title>
</head>
<body>
<BODY>
<CENTER>
<H2>爱在深秋</H2>
<IMG src="../../image/6-5.gif">
</CENTER>
<WBR>
<p>秋雨无声无息地下着。<BR>
飒飒的秋风不可一世地横行在萧条的郊外。无力与秋风抗争的枯叶，只能带着丝丝牵挂，无可奈何地飘离
留恋的枝头。秋蝉衰弱的残声逐渐地少了，地上落叶多了.....<BR>
黄昏，我漫步在郊外的林间，想细细地品味秋雨的凄冷。<BR>
然而，“雨到深秋易作霖，萧萧难会此时心”，此时，又有谁能听我诉说心中的那份情怀呢？<BR>
</p>
</WBR>
</body>
</html>
```

5-2-2 设定上下左右空白位置 hspace/vspace

实例: [5-2-2.html](#)

```
<html>
<head>
<title>设定图像与文本之间的距离</title>
```

```
<body>

<font size="+3">秋雨无声无息地下着。<BR>
飒飒的秋风不可一世地横行在萧条的郊外。无力与秋风抗争的枯叶，只能带着丝丝牵挂，无可奈何地飘离
留恋的枝头。秋蝉衰弱的残声逐渐地少了，地上落叶多了.....<BR>
黄昏，我漫步在郊外的林间，想细细地品味秋雨的凄冷。
然而，“雨到深秋易作霖，萧萧难会此时心”，此时，又有谁能听我诉说心中的那份情怀呢？</font><BR>
</body>
</html>
```

5-2-3 设定字画对其方式

图像的对齐：

- 1) 单独占一行时，放在<p>...</p>中，用<p>的对齐属性进行设置。
- 2) 与文本在同一行时，用其自身的 align 属性（top,middle,bottom）设置图像与文本的垂直对齐。其中：bottom 为默认值。
- 3) 图文混排时，可实现图像的左、右环绕文本，用 align 属性（left 图像在左、文本在右,right）。

实例：[5-2-3.html](#)

```
<html>
<head>
<title>控制图像相对于文字基准线的水平对齐方式</title>
</head>
<body>

```

此图像相对于文字基准线为靠上对齐的多行文字
试想在圆月朦胧之夜，海棠是这样的妩媚而嫣润；枝头的好鸟为什么却双栖而各梦呢？在这夜深人静的当儿，那高踞着的一只八哥儿，又为何尽撑着眼皮儿不肯睡去呢？他到底等什么来着？舍不得那淡淡的月儿么？舍不得那稀疏的帘儿么？不，不，不，您得到帘下去找，您得向帘中去找——您该找着那卷帘人了？他的情韵风怀，原是这样这样的哟！朦胧的岂独月呢；岂独鸟呢？但是，咫尺天涯，教我如何耐得？
我拚着千呼万唤；你能够出来么？
这页画布局那样经济，设色那样柔活，故精采足以动人。虽是区区尺幅，而情韵之厚，已足沦肌浃髓而有余。我看了这画。瞿然而惊：留恋之怀，不能自己。故将所感受的印象细细写出，以志这一段因缘。但我于中西的画都是门外汉，所说的话不免为内行所笑。——那也只好由他了。

```
<p>
<hr color="#00ff00">

```

此图像相对于文字基准线为靠上的多行文字对齐
试想在圆月朦胧之夜，海棠是这样的妩媚而嫣润；枝头的好鸟为什么却双栖而各梦呢？在这夜深人静的当儿，那高踞着的一只八哥儿，又为何尽撑着眼皮儿不肯睡去呢？他到底等什么来着？舍不得那淡淡的月儿么？舍不得那稀疏的帘儿么？不，不，不，您得到帘下去找，您得向帘中去找——您该找着那卷帘人了？他的情韵风怀，原是这样这样的哟！朦胧的岂独月呢；岂独鸟呢？但是，咫尺天涯，教我如何耐得？
我拚着千呼万唤；你能够出来么？
这页画布局那样经济，设色那样柔活，故精采足以动人。虽是区区尺幅，而情韵之厚，已足沦肌浃髓而有余。我看了这画。瞿然而惊：留恋之怀，不能自己。故将所感受的印象细细写出，以志这一段因缘。但我于中西的画都是门外汉，所说的话不免为内行所笑。——那也只好由他了。

```
<p>
```

```

<hr color="#00ff00">

此图像相对于文字基准线为顶部单行对齐<p>
<hr color="#00ff00">

此图像相对于文字基准线为底线单行对齐</p>
<p>
<hr color="#00ff00">
<p>
此图像相对于文字基准线为置中单行对齐</p>
<p>&nbsp;</p>
<p><a href="#" onClick="window.history.back()">返回</a></p>
<p>
</body>
</html>

```

5-2-4 图片大小设定

实例：[5-2-4.html](#)

```

<html>
<head>
<title>图像大小的设定</title>
</head>
<body>
<center>
<p>
缩小图像
<p>
<p>原图显示
<p>

<p>放大图像
<p>

</p>
</center>
</body>
</html>

```

5-2-5 图像边框的设定

实例：[5-2-5.html](#)

```

<html>
<head>
<title>设定图像的边框</title>
</head>

```



```

<body>
<center>
<div align="center">
<pre></pre>
</div>
</body>
</html>

```

5-3 用标签插入 avi 文件

格式：

标签插入 avi 文件的属性

属 性	描 述
dynsrc	指定 avi 文件所在路径
loop	设定 avi 文件循环次数
loopdelay	设定 avi 文件循环延迟
start	设定文件播放方式 fileopen/mouseover(网页打开时即播放/ 当鼠标滑到 avi 文件时播放)

实例：[6-4.html](#)

```

<html>
<head>
<title>设定图像的边框</title>
</head>
<body>
<center>
<p align="center">

</p>
</center>
</body>
</html>

```

第六章 建立超链接

HTML 文件中最重要的应用之一就是超链接，超链接是一个网站的灵魂，web 上的网页是互相链接的，单击被称为超链接的文本或图形就可以链接到其它页面。超文本具有的链接能力，可层层链接相关文件，这种具有超级链能力的操作，即称为超级链接。超级链接除了可链接文本外，也可链接各种媒体，如声音、图象、动画，通过它们我们可享受丰富多采的多媒体世界。

建立超链接的标签为<A>和

格式为：超链接名称

说明：标签<A>表示一个链接的开始，表示链接的结束；

属性“HREF”定义了这个链接所指的目标地址；目标地址是最重要的，一旦路径上出现差错，该资源就无法访问

TARGET: 该属性用于指定打开链接的目标窗口，其默认方式是原窗口。

建立目标窗口的属性

属性值	描 述
_parent	在上一级窗口中打开，一般使用分帧的框架页会经常使用
_blank	在新窗口打开
_self	在同一个帧或窗口中打开，这项一般不用设置
_top	在浏览器的整个窗口中打开，忽略任何框架

TITLE: 该属性用于指定指向链接时所显示的标题文字。

“超链接名称”是要单击到链接的元素，元素可以包含文本，也可以包含图像。文本带下划线且与其它文字颜色不同，图形链接通常带有边框显示。用图形做链接时只要把显示图像的标志嵌套在之间就能实现图像链接的效果。当鼠标指向“超链接名称”处时会变成手状，单击这个元素可以访问指定的目标文件。

6-1 链接路径

每一个文件都有自己的存放位置和路径，理解一个文件到要链接的那个文件之间的路径关系是创建链接的根本。

URL(Uniform Resourc Locator)中文名字为“统一资源定位器”。指的就是每一个网站都具有的地址。同一个网站下的每一个网页都属于同一个地址之下，在创建一个网站的网页时，不需要为每一个连接都输入完全的地址，我们只需要确定当前文档同站点根目录之间的相对路径关系就可以了。因此，链接可以分为以下三种：

- 绝对路径
- 如：http://www.sina.com.cn
- 相对路径
- 如：news/index.html
- 根路径
- 如：d | /web/news/index.html

1、绝对路径

绝对路径：包含了标识 INTERNET 上的文件所需要的所有信息。文件的链接是相对原文档而定的。包括完整的协议名称，主机名称，文件夹名称和文件名称

其格式为： 通讯协议：//服务器地址：通讯端口/文件位置...../文件名。

Internet 遵循一个重要的协议及 HTTP(Hypertext Transfer Protocol)超文本传输协议, http 是用于传输 Web 页的客户端/服务器协议。当浏览器发出 Web 页请求时,此协议将建立一个与服务器的链接.当链接畅通后,服务器将找到请求页,并将它发送给客户端,信息发送到客户端后,http 将释放此链接.这使得此协议可以接受并服务大量的客户端请求。

Web 应用程序是指 Web 服务器上包含的许多静态的和动态的资源集合.Web 服务器承担着为浏览器提供服务的责任。

www.sina.com.cn 就是资源所在的主机名为: 通常情况下使用默认的端口号 80, 资源在 WWW 服务器主机 web 文件夹下, 资源的名称为: index.html。

例 : http://www.163.net/myweb/book.htm (此网址为假设)

表明采用 http 从名为 www.163.net 的服务器上的目录 myweb 中获得文件 book.htm

2、相对路径

相对路径是以当前文件所在路径为起点, 进行相对文件的查找。一个相对的 URL 不包括协议和主机地址信息, 表示它的路径与当前文档的访问协议和主机名相同, 甚至有相同的目录路径。通常只包含文件夹名和文件名。甚至只有文件名。可以用相对 URL 指向与源文档位于同一服务器或同文件夹中的文件。此时, 浏览器链接的目标文档处在同一服务器或同一文件夹下。

- 如果链接到同一目录下, 则只需输入要链接文件的名称。
- 要链接到下级目录中的文件。只需先输入目录名, 然后加"/", 再输入文件名。
- 要链接到上一级目录中文件, 则先输入"..", 再输入文件名。

相对路径的用法

相对路径名	含 义
herf="shouey.html"	shouey.html 是本地当前路径下的文件
herf="web/shouey.html"	shouey.html 是本地当前路径下称做"web"子目录下的文件
herf="../shouey.html"	shouey.html 是本地当前目录的上一级子目录下的文件
herf="../../shouey.html"	shouey.html 是本地当前目录的上两级子目录下的文件

3、根路径

根路径目录地址同样可用于创建内部链接, 但大多数情况下, 不建议使用此种链接形式。

根路径目录地址的书写也很简单, 首先以一个斜杠开头, 代表根目录, 然后书写文件夹名, 最后书写文件名。

如果根目录要写盘符, 就在盘符后使用"|", 而不用":", 这点与 DOS 的写法不同。

如: /web/highlight/shouey.html

d | /web/highlight/shouey.html

也许读者会问, 链接本地机器上的文件时, 应该使用相对路径还是根路径? 在绝大多数情况下使用相对路径比较好, 例如, 用绝对路径定义了链接, 当把文件夹改名或者移动之后, 那么所有的链接都要失败, 这样就必须对你的所有 html 文件的链接进行重新编排, 而一旦将此文件夹移到网络服务器上时, 需要重新改动的地方就更多了, 那是一件很麻烦的事情。而使用相对路径, 不仅在本本地机器环境下适合, 就是上传到网络或其他系统下也不需要多少更改就能准确链接。

6-2 超链接的应用

6-2-1 书签链接

链接文档中的特定位置也叫书签链接，在浏览页面时如果页面很长，要不断的拖动滚动条给浏览带来不便，要是浏览者可以从上头阅读到尾，又可以选择自己感兴趣的部分阅读，这种效果就可以通过书签链接来实现，方法是选者一个目标定位点，用来创建一个定位标记，用<a>标签的属性 name 的值来确定定位标记名 <a>。然后在网页的任何地方建立对这个目标标记的链接"标题"，在标题上建立的链接地址的名字要和定位标记名相同，前面还要加上"#"号，。单击标题就跳到要访问的内容。

书签链接可以在同一页面中链接，也可以在不同页面中链接，在不同页面中链接的前提是需要指定好链接的页面地址和链接的书签位置

格式：

- 在同一页面要使用链接的地址：

超链接标题名称

- 在不同页面要使用链接的地址：

超链接标题名称

- 链接到的目的地址：

目标超链接名称

name 的属性值为该目标定位点的定位标记点名称，是给特定位置点(这个位置点也叫锚点)起个名称。

实例[4-2-11html](#)

<HTML>

<HEAD>

<TITLE>唐诗欣赏</TITLE>

</HEAD>

<BODY>

<a ><H2>唐诗欣赏</H2>

<a ><H2>李白</H2>

清平調三首

長干行

月下獨酌

<HR>

<H3><A >清平調三首 </H3>

雲想衣裳花想容，
春風拂檻露華濃。
若非群玉山頭見，
會向瑤台月下逢。
一枝紅艷露凝香，
雲雨巫山枉斷腸。
借問漢宮誰得似？
可憐飛燕倚新妝。
名花傾國兩相歡，
常得君王帶笑看。
解釋春風無限恨，
沈香亭北倚闌干。

<H3><A >長干行 </H3>

<PRE>

妾髮初覆額，
折花門前劇。
郎騎竹馬來，
遶床弄青梅。
同居長干里，
兩小無嫌猜。
十四為君婦，
羞顏未嘗開。
低頭向暗壁，
千喚不一回。
十五始展眉，
願同塵與灰。
常存抱柱信，
豈上望夫臺。
十六君遠行，
瞿塘滪澗堆。
五月不可觸，
猿鳴天上哀。
門前逕行跡，
一一生綠苔。
苔深不能掃，
落葉秋風早。
八月蝴蝶來，
雙飛西園草。
感此傷妾心，
坐愁紅顏老。
早晚下三巴，
預將書報家。
相迎不道遠，
直至長風沙。

```

</PRE>
<A href="#top">唐诗欣赏</a>
<BR><BR>
<H3><A >月下獨酌</A> </H3>
<PRE>
花間一壺酒，<br>獨酌無相親。<br>舉杯邀明月，<br>對影成三人。<br>月既不解飲，<br>影徒隨我身。
<br>暫伴月將影，<br>行樂須及春。<br>我歌月裴回，<br>我舞影零亂。<br>醒時同交歡，<br>醉後各分散。
<br>永結無情遊，<br>相期邈雲漢。
</PRE>
<p><a href="4-2-12.html#libai">李白蔑視權貴</a></p>
</BODY>
</HTML>

```

实例[4-2-12html](#)

```

<html>
<head>
<title>李白</title>
</head>
<body>
<center>
<a href="4-2-11.html#lb"><h1 align="center"><font color="#339933">李白</font></h1></a>
<font color="#339933" size="+2">李白（701~762），<br>字太白，号青莲居士。<br>祖籍陇西成纪（今
甘肃省天水市附近的秦安县），<br>隋朝末年其先祖因罪住在中亚细亚。<br>李白的家世和出生地至今还
是个谜，<br>学术界说法不一。<br>一说李白就诞生在安西都护府所辖的碎叶城，<br>五岁时随父迁到绵
州昌隆县青莲乡。<br>
<p>李白性情豪放，<br>喜爱纵横家的作风，<br>爱好任侠之事，轻视财货。<br>早年在蜀中度过。他的
父亲是个富商。<br>李白二十五岁开始漫游全国，<br>走过湖北，江西，河南，山东等地。<br></p>
<p><A >李白蔑视权贵</A>，<br>传说他喝醉酒，<br>曾在玄宗面前使高力士给他脱靴。<br>高力士认为
这是很大的耻辱，<br>就摘取李白诗句激怒杨贵妃。<br>玄宗每次让李白做官，杨贵妃就加以阻止。<br>
李白知道玄宗的亲信对他有意见，<br>于是恳求还家。<br>玄宗赐给他财物，放他开。<br></p>
<p>李白是我国唐代伟大的浪漫主义诗人，<br>被誉为“诗仙”。<br>他的诗豪迈瑰丽，诗里有突破现实的幻
想。<br>也有对当时民生疾苦的反映和对政治黑暗的抨击。<br>他的散文具有清新明朗，<br>奔放流畅的
特点。</p></font></center>
</body>
</html>

```

6-2-2 在站点内部建立链接

所谓内部链接，指的是在同一个网站内部，不同的 html 页面之间的链接关系，在建立网站内部链接的时候，要明确哪个是主链接文件（即当前页），哪个是被链接文件。在前面介绍链接路径时，已经给大家介绍了内部链接概念，内部链接一般采用相对路径链接比较好。下面我们根据如图示，来看看相对路径的具体的链接方法：

在站点内部建立链接

| 当前页面 | 被链接页面 | 超链接代码 |
|----------|----------|--------------------------------------|
| 2-1.html | 3-1.html | 超链接元素 |
| 3-1.html | 1-1.html | 超链接元素 |
| sy.html | 1-1.html | 超链接元素 |
| 2-1.html | sy.html | 超链接元素 |
| 1-1.html | sy.html | 超链接元素 |
| sy.html | 2-1.html | 超链接元素 |

6-2-3 外部链接

所谓外部链接，指的是跳转到当前网站外部，与其它网站中页面或其它元素之间的链接关系。这种链接的 URL 地址一般要用绝对路径，要有完整的 URL 地址，包括协议名，主机名，文件所在主机上的位置的路径以及文件名。

最常用的外部链接格式是：，还有其他的格式如下表所示：

URL 外链部接格式

| 服 务 | URL 格式 | 描 述 |
|--------|-------------|-----------------|
| WWW | http://"地址" | 进入万维网站点 |
| Ftp | ftp://" | 进入文件传输协议 |
| Telnet | telnet://" | 启动 Telnet 方式 |
| Gopher | gopher://" | 访问一个 gopher 服务器 |
| News | news://" | 启动新闻讨论组 |
| Email | email://" | 启动邮件 |

6-2-31 链接其它站点：

站点之间的页面和元素的链接是万维网交流信息的关键，这种链接用 HTTP 协议来建立网站之间的超链接。

格式：

实例 6-2-31.html

```

<HTML>
<HEAD>
<TITLE>链接到 http://</TITLE>
</HEAD>
<BODY>
<CENTER>
<h2>绝对路径链接</h2>
<HR>
<A HREF="http://www.sina.com.cn" title="打开新浪首页"></A><BR><BR>
<A HREF="http://www.tsinghua.edu.cn">清 华 大 学</A><BR><BR>
<A HREF="http://www.pku.edu.cn">北 京 大 学</A><BR><BR>
<A HREF="http://www.shisu.edu.cn" target=_blank>上海外国语学院</A><BR>
</CENTER>

```

```
</BODY>
</HTML>
```

6-2-32 发送 E-mail

在 HTML 页面中，可以建立 E-mail 链接。当浏览者单击链接后，系统会启动默认的本地邮件服务系统发送邮件。

基本语法：描述文字
在实际应用中，用户还可以加入另外的两个参数"?cc="E-mail 地址"和"&body=?",这分别表示在发送邮件的同时把邮件抄送给第三者和设定邮件主题内容。

邮件的参数

参 数	描 述
subject	电子邮件主题
cc	抄送收件人
body	主题内容
bcc	暗送收件人

如果希望同时写下多个参数，则参数之间使用"&"分隔符。

实例 6-2-32.html

```
<html>
<head>
<title>发送邮件</title>
</head>
<body>
<p>
<a href="mailto:zhoujr3071@yahoo.com.cn?cc=benbenmao@163.com&Subject=早安&bcc=a@b.c
&Body= 早安,笨笨猫祝你度过一个快乐的早晨！">
向笨笨猫的好友发送邮件(抄送 笨笨猫)
</a>
</p>
</body>
</html>
```

单击"向笨笨猫的好友发送邮件(抄送 笨笨猫)"后打开了系统的邮件服务软件 Outlook Exqress 发送邮件。

6-2-33 链接 FTP

Internet 上资源丰富，通过 ftp 文件传输协议，就可以获得各种免费软件和其他文件，ftp 协议是使计算机与计算机之间能够相互通信的语言，ftp 使文件和文件夹能够在 Internet 上公开传输，通过 ftp 可以访问某个网络或服务器，而不需要该计算机的账户和授权的密码就可通过 ftp 公开获得数据

语法格式：

```
<a href="ftp://ftp 主机地址">文字链接</a>
```

实例：6-2-33.html

```
<html>
<head>
<title>链接 ftp 主机</title>
</head>
<body>
```

```
<p>
<a href=ftp://ftp.pku.edu.cn>北京大学 ftp 站点</a>
</p>
</body>
</html>
```

6-2-4 图像的超链接

6-2-41 图像的超链接

图像的链接和文字的链接方法是一样的，都是用<a>标签来完成，只要将标签放在<a>和之间就可以了。用图像链接的图片的上有蓝色的边框，这个边框颜色也可以在<body>标签中设定。

实例：[6-2-41.html](#)

```
<html>
<head>
<title>使用图像为选取的对象</title>
</head>
<body>
<p align="center">&nbsp;</p>
<h1 align="center">图片的超链接</h1>
<P>
<center>
<a href="http://www.sohu.com/" target="_blank"></a><p>
<a href="http://www.baidu.com/"></a><p>
<a href="http://www.sina.com.cn"></a>
</center>
</body>
</html>
```

6-2-42 图像的影像地图超链接

在 HTML 中还可以把图片划分成多个热点区域，每一个热点域链接到不同网页的资源。这种效果的实质是把一幅图片划分为不同的热点区域，再让不同的区域进行超链接。这就是影像地图。要完成地图区域超链接要用到三种标签：<map><area>下面分别介绍这些标签的用法：

影像地图（Image Map）标签的使用格式：

```

```

<!-- 插入图片时要在标记中设置参数 usemap="#图的名称" ismap, 以表示对图像地图(图的名称)的引用; -->

```
<map name="图的名称">
```

```
<!--用<map>标记设定图像地图的作用区域，并用 name 属性为图像起一个名字-->
```


<area shape=形状 coords=区域坐标列表 href="URL 资源地址">

.....可根据需要定义多少个热点区域

<area shape=形状 coords=区域坐标列表 href="URL 资源地址">

</map>

【1】shape -- 定义热点形状

shape=rect: 矩形

shape=circle: 圆形

shape=poly: 多边形

【2】coords -- 定义区域点的坐标

a.矩形：必须使用四个数字，前两个数字为左上角坐标，后两个数字为右下角坐标

例：<area shape=rect coords=100,50,200,75 href="URL">

b.圆形：必须使用三个数字，前两个数字为圆心的坐标，最后一个数字为半径长度

例：<area shape=circle coords=85,155,30 href="URL">

c.任意图形（多边形）：将图形之每一转折点坐标依序填入

例：<area shape=poly coords=232,70,285,70,300,90,250,90,200,78 href="URL">

在制作本文介绍的效果时应注意的几点：

1、在标记不要忘记设置 usemap、ismap 参数，且 usemap 的参数值必须与<map>标记中的 name 参数值相同，也就是说，“图像地图名称”要一致；

2、同一“图像地图”中的所有热点区域都要在图像地图的范围内，即所有<area>标记均要在<map>与</map>之间；

3、在<area>标记中的 coords 参数设定的坐标格式要与 shape 参数设定的作用区域形状配套，避免出现 shape 参数设置的矩形作用区域，而在 coords 中设置的却是多边形区域顶点坐标的现象出现。

实例 [6-2-42.html](#)

<html>

<head>

<title>影像地图</title>

</head>

<body>

<map name="yxdt">

<area shape="rect" coords="80,69,180,120" href="http://www.baidu.com/" target="_blank" alt="点击链接到百度搜索">

<area shape="circle" coords="283,95,45" href="http://www.sina.com.cn" target="_blank" alt="点击链接到新浪网站">

</map>

<p> </p>

<p> </p>

<p> </p>

<p> </p>

<p> </p>

<p> </p>

<p>返回</p>

</body>

</html>

第七章 TABLE表格

表格在网站应用中非常广泛,可以方便灵活的排版,很多动态大型网站也都是借助表格排版,表格可以把相互关联的信息元素集中定位,使浏览页面的人一目了然.所以说要制作好网页,就要学好表格

7-1 定义表格的基本语法

在 html 文档中, 表格是通过<table>,<th>,<tr>,<td>标签来完成的, 如下表所示:

表格标记

标 签	描 述
<table>...</table>	用于定义一个表格开始和结束
<caption> ...</caption>	定义表格的标题。在表格中也可以不用此标签。
<th>...</th>	定义表头单元格。表格中的文字将以粗体显示, 在表格中也可以不用此标签, <th>标签必须放在<tr>标签内
<tr>...</tr>	定义一行标签, 一组行标签内可以建立多组由<td>或<th>标签所定义的单元格
<td>...</td>	定义单元格标签, 一组<td>标签将建立一个单元格, <td>标签必须放在<tr>标签内

注: 在一个最基本的表格中, 必须包含一组<table>标签, 一组标签<tr>和一组<td>标签或<th>。

实例: [7-1.html](#)

```
<HEAD>
<TITLE>一个简单的表格</TITLE>
</HEAD>
<BODY>
<center>
  <table>
    <caption>表格标题</caption>
    <tr>
      <td>第 1 行中的第 1 列</td>
      <td>第 1 行中的第 2 列</td>
      <td>第 1 行中的第 3 列</td>
    </tr>
    <tr>
      <td>第 2 行中的第 1 列</td>
      <td>第 2 行中的第 2 列</td>
      <td>第 2 行中的第 3 列</td>
    </tr>
  </table>
</center>
</BODY>
```

</HTML>

7-2 表格<table>标签属性

7-2（1） 表格<table>标签的常用属性

表格标签<table>有很多属性，最常用的属性有：

<table>标签的属性

属 性	描 述	说明
width	表格的宽度	
height	表格的高度	
align	表格在页面的水平摆放位置	
background	表格的背景图片	
bgcolor	表格的背景颜色	
border	表格边框的宽度（以像素为单位）	
bordercolor	表格边框颜色	当 border>=1 时起作用
bordercolorlight	表格边框明亮部分的颜色	当 border>=1 时起作用
bordercolordark	表格边框昏暗部分的颜色	当 border>=1 时起作用
cellspacing	单元格之间的间距	
cellpadding	单元格内容与单元格边界之间的空白距离的大小	

实例：[7-2-1.html](#)

```
<html>
<head>
<title>无标题文档</title>
</head>
<body>
<table border=10 bordercolor="#006803" align="center" bgcolor="#DDFFDD" width=500
height="200"bordercolorlight="#FFFFCC" bordercolordark="#660000"
background="../../img/b0024.gif" cellspacing="2" cellpadding="8">

<tr>
<td>第 1 行中的第 1 列</td>
<td>第 1 行中的第 2 列</td>
<td>第 1 行中的第 3 列</td>
</tr>
<tr>
<td>第 2 行中的第 1 列</td>
<td>第 2 行中的第 2 列</td>
<td>第 2 行中的第 3 列</td>
```

```
</tr>
</table>
</body>
</html>
```

7-2（2） 设置分隔线的显示状态 rules

语法格式：<table rules="值">

分隔线的显示状态 rules 的值的设定

rules 的值	描 述
all	显示所有分隔线
groups	只显示组与组的分隔线
rows	只显示行与行的分隔线
cols	只显示列与列的分隔线
none	所有分隔线都不显示

实例: [7-2-3.html](#)

```
<html>
<head>
<title>无标题文档</title>
</head>
<body>
<TABLE border=6 bgcolor="#FFFFCC" rules="cols" bordercolor="#9900FF" width="400"
height="160" align="center">
<TR>
<TH>姓名</TH>
<TH>性别</TH>
<TH>年龄</TH>
<TH>专业</TH>
</TR>
<TR>
<TD>笨笨猫</TD>
<TD>女</TD>
<TD>99</TD>
<TD>计算机</TD>
</TR>
</TABLE><p>
<TABLE border=6 bgcolor="#FFFFCC" rules="rows" bordercolor="#9900FF" width="400"
height="160" align="center">
<TR>
<TH>姓名</TH>
<TH>性别</TH>
<TH>年龄</TH>
<TH>专业</TH>
```

```
</TR>
<TR>
<TD>笨笨猫</TD>
<TD>女</TD>
<TD>99</TD>
<TD>计算机</TD>
</TR>
</TABLE>
</body>
</html>
```

7-2（3） 表格的边框显示状态 frame

表格的边框分别有上边框、下边框、左边框、右边框。这四个边框都可以设置为显示或隐藏状态
语法格式：<table frame="边框显示值">

表格边框显示状态 frame 的值的设定

frame 的值	描 述
box	显示整个表格边框
void	不显示表格边框
hsides	只显示表格的上下边框
vsides	只显示表格的左右边框
above	只显示表格的上边框
below	只显示表格的下边框
lhs	只显示表格的左边框
rhs	只显示表格的右边框

实例：[7-2-2.html](#)

```
<HTML>
<HEAD>
<TITLE>表格边框的显示状态</TITLE>
</HEAD>
<BODY >
<TABLE border=6 bgcolor="#FFFFCC" frame="hsides" bordercolor="#9900FF"
width="400" height="160">
<TR>
<TH>姓名</TH>
<TH>性别</TH>
<TH>年龄</TH>
<TH>专业</TH>
</TR>
<TR>
<TD>卡卡</TD>
<TD>男</TD>
```

```
<TD>50</TD>
<TD>计算机</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

7-3 表格行的设定

表格是按行和列（单元格）组成的，一个表格有几行组成就要有几个行标签<tr>，行标签用它的属性值来修饰，属性都是可选的。

<tr>标签的属性

属 性	描 述
height	行高
align	行内容的水平对齐
valign	行内容的垂直对齐
bgcolor	行的背景颜色
bordercolor	行的边框颜色
bordercolorlight	行的亮边框颜色
bordercolordark	行的暗边框颜色

```
<TR> 的参数设定（常用）：
<tr align="RIGHT" valign="MIDDLE" bgcolor="#0000FF" bordercolor="#FF00FF"
bordercolorlight="#808080" bordercolordark="#FF0000">
```

实例：[7-3.html](#)

```
<HTML>
<HEAD>
<TITLE>表格行的控制</TITLE>
</HEAD>
<BODY>
<TABLE border=1 align="center" width="80%" height="150">
<TR ALIGN="CENTER">
<TH>姓 名</TH>
<TH>性 别</TH>
<TH>年 龄</TH>
<TH>专 业</TH>
</TR>
<TR ALIGN=CENTER bordercolor="#336600" bgcolor="#C1FFC1">
<TD>咚 咚</TD>
<TD>男</TD>
<TD>18</TD>
<TD>学 生</TD>
</tr>
```

```
<tr align=center height=50 bordercolor=navy bgcolor="#86B8E1" valign=bottom
bordercolorlight="#E1F0FD" bordercolordark="#002346">
<TD>楠 楠</TD>
<TD>女</TD>
<TD>17</TD>
<TD>学 生</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

7-4 单元格的设定

<th>和<td>都是插入单元格的标签，这两个标签必须嵌套在<tr>标签内。是成对出现的。<th>用于表头标签，表头标签一般位于首行或首列，标签之间的内容就是位于该单元格内的标题内容，其中的文字以粗体居中显示。数据标签<td>就是该单元格中的具体数据内容,<th>和<td>标签的属性都是一样的，属性设定如下：

<th>和<td>的属性	
属 性	描 述
width/height	单元格的宽和高，接受绝对值（如 80）及相对值（如 80%）。
colspan	单元格向右打通的栏数
rowspan	单元格向下打通的列数
align	单元格内字画等的摆放贴，位置（水平），可选值为： left, center, right。
valign	单元格内字画等的摆放贴 位置（垂直），可选值为： top, middle, bottom。
bgcolor	单元格的底色
bordercolor	单元格边框颜色
bordercolorlight	单元格边框向光部分的颜色
bordercolordark	单元格边框背光部分的颜色
background	单元格背景图片
nowrap	<td nowrap>禁止单元格内容自动换行。 TD 元素 nowrap 属性的行为与 TD 元素的 width 属性有关，在设置了 table 元素的 width 后： ◆如未设置 TD 宽度，则 nowrap 属性起作用； ◆如设置了 TD 宽度,则 nowrap 属性不起作用。 注：如果不设置 table 元素的 width 属性，nowrap 不起作用。

<TD> 的参数设定格式：

例如: <td width="48%" height="400" colspan="5" rowspan="4" align="RIGHT" valign="BOTTOM" bgcolor="#FF00FF" bordercolor="#808080" bordercolorlight="#FF0000" bordercolordark="#00FF00" background="myweb.gif">

实例: [7-4 \(1\) .html](#)

```
<HTML>
<HEAD>
<TITLE>单元格的设定</TITLE>
</HEAD>
<BODY>
<TABLE border=1 align="center" height="150" width="80%">
<TR>
<TH width=70 bgcolor="#FFCC00">姓 名</TH>
<TH bgcolor="#FFCCFF">性 别</TH>
<TH background=".../imge/b0024.gif">年 龄</TH>
<TH background=".../imge/aki-5.gif">专 业</TH>
</TR>
<TR>
<TD bordercolor=red align="left">笨笨猫</TD>
<TD bordercolorlight="#FFCCFF" bordercolordark="#FF0000" align="center">女</TD>
<TD bgcolor="#FFFFCC" valign="bottom" align="center">18</TD>
<TD bgcolor="#CCFFFF" align="right">学生</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

实例: [7-4 \(2\) .html](#)

```
<html>
<head>
<title>wrap 属性研究</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<p>测试字符串: </p>
<p>我终于明白,我其实有一条韧性十足的命,它远比我想象中的那条命结实得多、耐磨的多……</p>
<p>单元格未设置 nowrap 属性的空表: </p>
<table width="100" border="1" cellspacing="0" cellpadding="0">
<tr>
<td>&nbsp;</td>
</tr>
</table>
<p>加入测试字符串: </p>
<table width="100" border="1" cellspacing="0" cellpadding="0">
```

```

<tr>
<td>我终于明白，我其实有一条韧性十足的命，它远比我想象中的那条命结实得多、耐磨的多……</td>
</tr>
</table>
<p>单元格设置了 nowrap 属性，未设置 width 属性：</p>
<table width="100" border="1" cellspacing="0" cellpadding="0">
<tr>
<td nowrap>我终于明白，我其实有一条韧性十足的命，它远比我想象中的那条命结实得多、耐磨的多……
</td>
</tr>
</table>
<p>单元格设置了 nowrap 属性，也设置了 width 属性：</p>
<table width="200" border="1" cellspacing="0" cellpadding="0">
<tr>
<td width="120" nowrap>我终于明白，我其实有一条韧性十足的命，它远比我想象中的那条命结实得多、耐磨的多……</td>
<td>&nbsp;   </td>
</tr>
</table>
</body>
</html>

```

7-5 设定跨多行多列单元格

要创建跨多行、多列的单元格，只需在<TH>或<TD>中加入 ROWSPAN 或 COLSPAN 属性的属性值，默认值为 1。表明了表格中要跨越的行或列的个数。

跨多列的语法： <th colspan=#> <td colspan=#>

colspan 表示跨越的列数，例如 colspan=2 表示这一格的宽度为两个列的宽度。

跨多行的语法： <th rowspan=#> <td rowspan=#>

rowspan 所要表示的意义是指跨越的行数，例如 rowspan=2 就表示这一格跨越表格两个行的高度。

实例： [7-5.html](#)

```

<html>
<head>
<title>跨多行跨多列的单元格</title>
</head>
<body>
<center>
<table border=10 width=80% align="center" height="150"
background="../img/b0024.gif" bordercolorlight="#9999FF"
bordercolordark="#9900CC">
<TR ALIGN=center>
<TH colspan=3> 学生基本信息</TH>

```

```

<TH colspan=2>成 绩</TH>
</TR>
<TR ALIGN=center>
<TH>姓 名</TH>
<TH>性 别</TH>
<TH>专 业</TH>
<TH>课 程</TH>
<TH>分 数</TH>
</TR>
<TR ALIGN=center>
<TD>咚 咚</TD>
<TD>男</TD>
<TD rowspan=2>计算机</TD><TD rowspan=2>程序设计</TD>
<TD>68</TD>
</TR>
<TR ALIGN=center>
<TD>喃 喃</TD>
<TD>女</TD>
<TD>89</TD>
</TR>
</table>
</body>
</html>

```

7-6 表格的分组

7-6-1 表格的行分组<thead>/<tbody>/<tfoot>

html 文档的表格按行分组是由表头标签<thead>、表格主体标签<tbody>和尾注标签<tfoot>三个部分组成的。其中尾注标签很少用。<thead>标签是成对标签，其标签内是由表头标签标识的表头元素。

<tbody>标签用于规定表格主体部分的元素，可出现多次。<thead>和<tbody>标签的属性和<th><td>标签是一样的。

实例：[7-6-1.html](#)

```

<html>
<head>
<title>表格按行分组</title>
</head>
<body>
<center>
<table border=3 width=60% align="center" height="150">
<thead bgcolor="#CCFFCC">
<TR ALIGN=center>
<TH>姓 名</TH>

```

```

<TH>性 别</TH>
<TH>专 业</TH>
</TR>
</thead>
<tbody bgcolor="#448FBD">
<TR ALIGN=center>
<TD>咚 咚</TD><TD>男</TD><TD>计算机</TD>
</TR>
<TR ALIGN=center>
<TD>喃 喃</TD><TD>女</TD><TD>园 林</TD>
</TR>
</tbody>
</table>
</body>
</html>

```

7-6-2 表格按列分组<colgroup>

html 使用<colgroup>标签来将表格分组。

语法格式: **<colgroup span="数值" align="参数">**

说明: <colgroup>标签有两个属性, span 和 align, 他们都是可选的。span 属性的值是数字, 表示该组包含的列数, 默认值为 1。align 属性用以规定该组所跨列中所有单元格中内容在水平方向上的位置。该属性的值为 left,center,right 之一。它们表示单元格的内容是左对齐, 水平居中还是右对齐。

实例: [7-6-2.html](#)

```

<html>
<head>
<title>表格按列分组</title>
</head>
<center>
<table border=10 width=80% height="160" align="center" bordercolorlight="#CCCCFF"
bordercolordark="#9900FF">
<tr>
<th>姓名</th><th>性别</th><th>专业</th><th>分数</th>
</tr>
<colgroup span=2 align=center>
<colgroup align =left>
<colgroup align=right>
<tr>
<td>咚 咚</td><td>男</td><td>计算机</td><td>79</td>
</tr>
<tr>
<td>喃 喃</td><td>女</td><td>园 林</td><td>90</td>
</tr>
<tr>

```

```

<td>依 依</td><td>女</td><td>微波通信</td><td>76<td>
</tr>
</table>
</body>
</html>

```

7-6-3 表格的行列分组

实例: [7-6-3.html](#)

```

<HTML>
<HEAD>
<TITLE>同时进行行列分组</TITLE>
</HEAD>
<BODY>
<CENTER>
<TABLE BORDER=10 WIDTH=80% height="200" align="center" bordercolor="#9900FF">
<COLGROUP SPAN=2 ALIGN=CENTER>
<COLGROUP ALIGN=LEFT>
<COLGROUP ALIGN=RIGHT>
<THEAD bgcolor="#FFFFCC">
<TR><TH>姓名</TH><TH>性别</TH><TH>专业</TH><TH>分数</TH>
<TR>
</THEAD>
<TBODY bgcolor="#FFCCFF">
<TR>
<TD>咚 咚</TD><TD>男</TD><TD>计算机</TD><TD>85</TD>
</TR>
<TR>
<TD>喃 喃</TD><TD>女</TD><TD>园 林</TD><TD>94</TD>
</TR>
<TR>
<TD>依 依</TD><TD>女</TD><TD>微波通信</TD><TD>87</TD>
</TR>
</TBODY>
</TABLE>
</BODY>
</HTML>

```

7-7 表格的标题标签<caption>

表格标题的位置,可由 **ALIGN** 属性和 **VALIGN** 属性来设置, **ALIGN** 属性设置标题位于文档的左,中,右。
VALIGN 属性设置标题位于表格的上方和表格的下方。下面为表格标题位置的设置格式。

语法格式:

<caption align="值" valign="值" >表哥标题</caption>

<caption>应放在<table>标签内，在表格行标签<tr>标签之前。

<caption>标签的默认属性是标题位于表格的上方中间位置。

实例: [7-7.html](#)

```
<html>
<head>
<title>表格的标题标签</title>
</head>
<body>
<center>
<table border=10 width=80% align="center" height="150"
background="../../image/b0024.gif" bordercolorlight="#9999FF"
bordercolordark="#9900CC">
<caption>学生信息表</caption>
<TR ALIGN=center>
<TH colspan=3> 学生基本信息</TH>
<TH colspan=2>成 绩</TH>
</TR>
<TR ALIGN=center>
<TH>姓 名</TH>
<TH>性 别</TH>
<TH>专 业</TH>
<TH>课 程</TH>
<TH>分 数</TH>
</TR>
<TR ALIGN=center>
<TD>咚 咚</TD>
<TD>男</TD>
<TD rowspan=2>计算机</TD><TD rowspan=2>程序设计</TD>
<TD>68</TD>
</TR>
<TR ALIGN=center>
<TD>喃 喃</TD>
<TD>女</TD>
<TD>89</TD>
</TR>
</body>
</html>
```

7-8 表格的嵌套

在html页面中，使用表格排版是通过嵌套来完成的，即一个表格内部可以嵌套另一个表格，用表格来排版页面的思路是：由总表格规划整体的结构，由嵌套的表格负责各个子栏目的排版，并插入到表格的相

应位置,这样就可以使页面的各个部分有条不紊,互不冲突,看上去清晰整洁。在实际做网页时一般不显示边框,边框的显示可根据自己的爱好来设定。在实例中为了让大家能够看清楚,都设置了有边框。

实例[7-8.html](#)

[illegible]

```
<tr>
  <td height="90">内容六</td>
</tr>
</table>
</td>
<td width="275">
  <table width="275" height="325" border="3" cellpadding="1" cellspacing="1">
    <tr>
      <td width="263">内容一</td>
    </tr>
    <tr>
      <td>内容二</td>
    </tr>
  </table>
</td>
<td width="163">
  <table width="157" height="320" border="3" cellpadding="1" cellspacing="1"
align="center">
    <tr>
      <td width="136" height="94">内容三</td>
    </tr>
    <tr>
      <td height="62">内容四</td>
    </tr>
    <tr>
      <td height="160">内容五</td>
    </tr>
  </table>
</td>
</tr>
</table>
</body>
</html>
```


第八章网页的动态、多媒体效果

在网页的设计过程中，动态效果的插入，会使网页更加生动灵活、丰富多彩。

8-1 滚动字幕<marquee>

<marquee>标签可以实现元素在网页中移动的效果，以达到动感十足的视觉效果。<marquee>标签是一个成对的标签。应用格式为：

```
<marquee>...</marquee>
```

<marquee>标签有很多属性，用来定义元素的移动方式：

<marquee>的属性

属 性	描 述
align	指定对齐方式 top,middle,bottom
bgcolor	设定文字卷动范围的背景颜色
loop	设定文字卷动次数，其值可以是正整数或infinite 表示无限次 默认为无限循环
height	设定字幕高度
width	设定字幕宽度
scrollamount	指定每次移动的速度,数值越大速度越快
scrolldelay	文字每一次滚动的停顿时间，单位是毫秒。时间越短滚动越快
hspace	指定字幕左右空白区域的大小
vspace	指定字幕上下空白区域的大小
direction	设定文字的卷动方向,left 表示向左,right 表示向右,up 表示往上滚动
behavior	指定移动方式，scroll 表示滚动播出，slibe 表示滚动到一方后停止，alternate 表示滚动到一方后向相反方向滚动。

[8-1 实例h8-1.html](#)

```
<html>
<body>
<center>
<font face="字体 2" size=6 color="#ff0000">
滚动字幕
</font><br>
<marquee>啦啦啦~~~我会跑了</marquee>
<p>
<marquee height="200" direction="up" hspace="200">啦啦啦~~~我会往上跑了<br>啦啦啦
~~~我会往上跑了</marquee>
<p>
<marquee direction="right">啦啦啦~~~我会往右跑了</marquee>
```

```

<p>
<marquee height="200" direction="down"><center>啦啦啦~~~我会往下跑了
</center></marquee>
<p>
<marquee width="500" behavior="alternate">啦啦啦~~~我来回的跑</marquee>
<p>
<marquee behavior="slide">啦啦啦~~~我跑到目的地就该休息了</marquee>
<P>
<marquee scrollamount="2">啦啦啦~~~我累了，要慢慢的溜达</marquee>
<P>
<marquee scrolldelay="300">啦啦啦~~~我累了，我要走走停停</marquee>
<p>
<marquee scrollamount="20">哈哈~都没有我跑得快</marquee>
<p>
<marquee>啦啦啦~~图片也可以啊</marquee>
<p>
<marquee bgcolor="#FFFFCC" width="700" vspace="30"><font size="+3"
color="#FF0000">啦啦啦~~滚动文字有背景了</font></marquee>
</center>
</body>
</html>

```

8-2 插入多媒体文件

在网页中可以用<embed>标签将多媒体文件插入，比如可以插入音乐和视频等。用浏览器可以播放的音乐格式有：MIDI 音乐、WAV 音乐、mp3、AIFF、AU 格式等。另外在利用网络下载的各种音乐格式中，MP3 是压缩率最高，音质最好的文件格式。但要说明一点，虽然我们代码标签插入了多媒体文件，但 IE 浏览器通常能自动播放某些格式的声音与影像，但具体能播放什么样格式的文件，取决于所用计算机的类型以及浏览器的配置。浏览器使用 Hn7 或 n7 协议获取多媒体文件，(可能得花很长时间，因为多媒体文件通常比较大)，浏览器既就决定如何播放。通常是调用称为插件的内置程序来播放的。事实上，浏览器仅仅能显示几种文件格式。是插件扩展了浏览器的能力。有许多种不同的插件程序，每种都能赋予浏览器一种新的能力。有时，不得不分别下载每个浏览器的多媒体插件程序。系统最小化的安装一般不包括声音与影像播放器。另外在播放影音文件时，若是使用一小部分窗口播放，大多数的计算机还比较快。若是全屏幕播放，就需要专用的硬件或是性能好的计算机。对于 IE，若无预先安装好的插件程序，它会提示你或是打开文件或是保存文件或是取消下载。若打开未知类型的文件，浏览器会试图使用外部的应用程序显示此文件。但这要取决于操作系统的配置。

<embed>标签的使用格式：

<EMBED SRC="音乐文件地址">

常用属性如下：

SRC="FILENAME"	设定音乐文件的路径
AUTOSTART=TRUE/FALSE	是否要音乐文件传送完就自动播放，TRUE 是要，FALSE 是不要，默认为 FALSE
LOOP=TRUE/FALSE	设定播放重复次数，LOOP=6 表示

	重复 6 次, TRUE 表示无限次播放, FALSE 播放一次即停止。
STARTIME="分:秒"	设定乐曲的开始播放时间, 如 20 秒后播放写为 STARTIME=00:20
VOLUME=0-100	设定音量的大小。如果没设定的话, 就用系统的音量。
WIDTH HEIGHT	设定播放控件面板的大小
HIDDEN=TRUE	隐藏播放控件面板
CONTROLS=CONSOLE/SMALLCONSOLE	设定播放控件面板的样子

[实例 h8-2-1.html](#)

```
<HTML>
<HEAD>
<TITLE>插入多媒体文件</TITLE>
</HEAD>
<body>
<H2 ALIGN="CENTER">网页中的多媒体</H2>
<HR>
<center>
<embed src="../../image/一千零一夜.mid" height=150 width=400 loop="false">
</center>
</BODY>
</HTML>
```

[实例 h8-2-2.html](#)

```
<HTML>
<HEAD>
<TITLE>插入多媒体文件</TITLE>
</HEAD>
<body>
<H2 ALIGN="CENTER">网页中的多媒体</H2>
<HR>
<center>
<embed src="../../image/mmm.avi" width="400" height="500" loop="false">
</center>
</BODY>
</HTML>
```

[实例 h8-2-3.html](#)

```
<HTML>
<HEAD>
<TITLE>插入 flash</TITLE>
</HEAD>
<body>
<H2 ALIGN="CENTER">网页中的多媒体</H2>
```

```

<HR>
<center>
<embed src="../../img/023.swf" height="500" width="550"><!--插入 flash-->
<embed src="../../img/023.swf" height="500" width="550" wmode="transparent"><!--
插入透明 flash-->
</center>
</BODY>
</HTML>

```

[实例 h8-2-4.html](#)

```

<html>
<head>
<title>在图片上插入透明 flash</title>
</head>
<body bgColor=lightblue>
<br><center>在图片上插入透明 Flash 动画</center><br>
<center><TABLE cellSpacing=2 cellPadding=2 border=10 bordercolorlight="#00CC99"
bordercolordark="#009933" background="../../img/hai.jpg"><TBODY><TD><EMBED
src="../../img/016.swf" width=630 height=480 type=application/x-shockwave-flash
quality="high" wmode="transparent"></embed></TD></TBODY></table></center>
</body>
</html>

```

注意 长和宽的设定要根据图片的实际大小来设定

8-3 嵌入多媒体文件

除了可以使用上述方法插入多媒体文件外，还可以在网页中嵌入多媒体文件，这种方式将不调用媒体播放器

8-3-1 嵌入背景音乐

<bgsound>标签用来设置网页的背景音乐。但只适用于 IE，其参数设定不多。格式如下：

```
<BGSOUND src="your.mid" autostart=true loop=infinite>
```

- src="your.mid"
- 设定 midi 档案及路径，可以是相对或绝对。声音文件可以是 wav, midi, mp3 等类型的文件
- autostart=true
- 是否在音乐档传完之後，就自动播放音乐。true 是，false 否（内定值）。
- loop=infinite
- 是否自动反覆播放。LOOP=2 表示重复两次，Infinite 表示重复多次。直到网页关闭为止。

[实例 8-3-1.html](#)

```

<HTML>
<HEAD>
<TITLE>背景声音</TITLE>
</HEAD>
<body>
<H4 ALIGN="CENTER" >网页的背景声音<H4>
<HR>

```

```
<bgsound src="../../image/小三和弦.mp3" LOOP="3">
```

```
</BODY>
```

```
</HTML>
```

背景音乐可以放在<body></body>或<head></head>之间

8-3-2 在网页中嵌入视屏文件使用前面学过的标签,在此就不重述了。

8-4 点播音乐

将音乐做成一个链接，只需用鼠标在上面单击，就可以听到动人的音乐了，这样做的方法很简单：

```
<A HREF="音乐地址">乐曲名</A>
```

例如：播放一段 MIDI 音乐：

```
<A HREF="image/一千零一夜.mid">MIDI 音乐</A>     MIDI 音乐
```

播放一段 AU 格式音乐：

```
<A HREF="image/小三和弦.mp3">小三和弦-mp3 音乐</A>     小三和弦-mp3 音乐
```

把我们喜欢的音乐收集起来，作成一个音乐库，随时都可以让自己和别人徜徉在音乐的海洋中，已经有人这样做了，你一定已经遇到很多，而且自己也可以试着去做了！

第九章 多视窗口框架

9-1 框架的含义和基本构成

框架就是把一个浏览器窗口划分为若干个小窗口，每个窗口可以显示不同的 URL 网页。使用框架可以非常方便的在浏览器中同时浏览不同的页面效果，也可以非常方便的完成导航工作。

而所有的框架标记 要放在一个 html 文档中。html 页面的文档体标签<body>被框架集标签<frameset>所取代，然后通过<frameset>的子窗口标签<frame>定义每一个子窗口和子窗口的页面属性。

语法格式：

```
<html>
<head>
</head>
<frameset>
    <frame src="url 地址 1">
    <frame src="url 地址 2">
    .....
</frameset>
</html>
```

Frame 子框架的 src 属性的每个 URL 值指定了一个 html 文件（这个文件必须事先做好）地址，地址路径可使用绝对路径或相对路径，这个文件将载入相应的窗口中。

框架结构可以根据框架集标签<frameset>的分割属性分为 3 种：

1. 左右分割窗口
2. 上下分割窗口
3. 嵌套分割窗口

9-2 框架集<frameset>控制

<frameset>的属性

属 性	描 述
border	设置边框粗细,默认是 5 像素.
bordercolor	设置边框颜色
frameborder	指定是否显示边框 : "0"代表不显示边框, "1"代表显示边框
cols	用"像素数" 和 "%"分割左右窗口,"*"表示剩余部分
rows	用"像素数" 和 "%"分割上下窗口,"*"表示剩余部分
framespacing="5"	表示框架与框架间的保留空白的距离
noresize	设定框架不能够调节, 只要设定了前面的, 后面的将继承

1. 左右分割窗口属性: cols

如果想要在水平方向将浏览器分割多个窗口, 这需要使用到框架集的左右分割窗口属性 cols.分割几个窗口其 cols 的值就有几个, 值的定义为宽度, 可以是数字(单位为像素), 也可以是百分比和剩余值. 各值之间用逗号分开. 其中剩余值用"*"号表示, 剩余值表示所有窗口设定之后的剩余部分, 当"*"只出现一次时, 表示该子窗口的大小将根据浏览器窗口的大小自动调整, 当"*"出现一次以上时, 表示按比例分割剩余的窗口空间. cols 的默认值为一个窗口

如: <frameset cols="40%,2*,*"> 将窗口分为 40%, 40%, 20%
 <frameset cols="100,200,*">
 <frameset cols="100,*,*"> 将 100 像素以外的窗口平均分配
 <frameset cols="*,*,*"> 将窗口分为三等份

2. 上下分割窗口属性: rows

上下分割窗口的属性设置和左右窗口的属性设定是一样的, 参照上面所述就可以了。

9-3 子窗口<frame>标签的设定

<frame>是个单标签,<frame>标签要放在框架集 frameset 中,<frameset>设置了几个子窗口就必须对应几个<frame>标签, 而且每一个<frame>标签内还必须设定一个网页文件 (src="*.html",其常用属性有:

<frame>常用属性

属 性	描 述
src	指示加载的 url 文件的地址
bordercolor	设置边框颜色
frameborder	指示是否要边框,1 显示边框, 0 不显示 (不提倡用 yes 或 no)
border	设置边框粗细
name	指示框架名称,是连结标记的 target 所要的参数
noresize	指示不能调整窗口的大小,省略此项时就可调整,
scrolling	指示是否要滚动条,auto 根据需要自动出现,Yes 有,No 无
marginwidth	设置内容与窗口左右边缘的距离, 默认为 1
marginheight	设置内容与窗口上下边缘的边距, 默认为 1
width	框窗的宽及高 默认为 width="100" height="100"
align	可选值为 left, right, top, middle, bottom

子窗口的排列遵循从左到右，从上到下的次序规则。

一、窗口的左右设定：

[实例一 sl9-1.html](#)

首先我们新建一个文件夹，然后做四个要放到子窗口中的页面，sl1.html、sl2.html、sl3.html、sl4.html。

```
<html>
<head>
</head>
<frameset rows="20%,2*,*" framespacing="1" frameborder="yes" border="1"
bordercolor="#FF00FF">
<frame src="sl1.html">
<frame src="sl2.html">
<frame src="sl3.html">
</frameset><noframes></noframes>
</html>
```

二、窗口的上下设定

[实例二 sl9-2.html](#)

```
<html>
<head>
</head>
<frameset rows="20%,*,200" framespacing="1" frameborder="yes" border="1"
bordercolor="#FF00FF">
<frame src="sl1.html">
<frame src="sl2.html">
<frame src="sl3.html" noresize="noresize">
</frameset><noframes></noframes>
</html>
```

三、窗口的嵌套设定

[实例三 sl9-3.html](#)

```
<html>
<head>
</head>
<frameset cols="20%,*" framespacing="1" frameborder="yes" border="1"
bordercolor="#FF00FF">
<frame src="sl1.html">
<frameset rows="300,500"framespacing="1" frameborder="yes" border="1"
bordercolor="#FF00FF">
<frame src="sl2.html">
<frame src="sl3.html">
</frameset>
</frameset><noframes></noframes>
</html>
```

[实例四 sl9-4.html](#)

```
<html>
<head>
```

```

</head>
<frameset rows="20%,*" framespacing="1" frameborder="yes" border="1"
bordercolor="#FF00FF">
<frame src="sl1.html">
<frameset cols="20%,*"framespacing="1" frameborder="yes" border="1"
bordercolor="#FF00FF">
<frame src="sl2.html">
<frame src="sl3.html">
</frameset>
</frameset><noframes></noframes>
</html>

```

实例五 [sl9-5.html](#)

```

<html>
<head>
</head>
<frameset rows="20%,*,15%" framespacing="1" frameborder="yes" border="1"
bordercolor="#FF00FF">
<frame src="sl1.html">
<frameset cols="20%,*"framespacing="1" frameborder="yes" border="1"
bordercolor="#FF00FF">
<frame src="sl2.html">
<frame src="sl3.html">
</frameset>
<frame src="sl4.html">
</frameset><noframes></noframes>
</html>

```

大家看到上面的文件中还有一对<noframes></noframes>标签，即使在做框架集网页时没有这对标签，文件在很多浏览器解析时也会自动生成<noframes>标签，这对标签的作用是当浏览者使用的浏览器太旧，不支援框架这个功能时，他看到的将会是一片空白。为了避免这种情况，可使用 <NOFRAMES>这个标记，当使用的浏览器看不到框架时，他就会看到 <NOFRAMES>你的浏览器不支持框架网页</NOFRAMES>之间的内容，而不是一片空白。这些内容可以是提醒 浏览转用新的浏览器的字句，甚至是一个没有框架的网页或能自动切换至没有框架的版本。

9-4 窗口的名称和链接

如果在窗口中要做链接，就必须对每一个子窗口命名，以便于被用于窗口间的链接，窗口命名要有一定的规则，名称必须是单个英文单词，允许使用下滑线，但不允许使用"—","句点"和空格等，名称必须以字母开头，不能使用数字，还不能使用网页脚本中保留的关键字，在窗口的链接中还要用到一个新的属性"targe",用这个属性就可以将被链接的内容放置到想要放置的窗口内。

下面的实例就是窗口内的命名和链接方法

文件一中的片断代码格式为：

```

<a href="sl1.html" target="aa2">爱在深秋</a>
<a href="sl2.html" target="aa3">图像对文字的水平居中</a>
<a href="sl3.html" target="aa2">图像与文本之间的距离</a>
<a href="sl4.html" target="aa3">图像大小的设定</a>
<a href="http://www.sina.com.cn" target="aa3">新浪网站</a>

```


文件二的片断代码格式为:

```
<frame src="ml.html" name="aa1">
<frame src="sl1.html" name="aa3">
<frame src="sl2.html" name="aa2" noresize="noresize">
```

文件一:

[ml.html](#)

```
<html>
<head>
<title>无标题文档</title>
</head>
<body>
<center>
<h2>目录</h2>
<hr>
<p><a href="sl1.html" target="aa2">爱在深秋</a></p>
<p><a href="sl2.html" target="aa3">图像对文字的水平居中</a></p>
<p><a href="sl3.html" target="aa2">图像与文本之间的距离</a></p>
<p><a href="sl4.html" target="aa3">图像大小的设定</a></p>
<p><a href="http://www.sina.com.cn" target="aa3">新浪网站</a></p>
</center>
</body>
</html>
```

文件二:

[sl94.html](#)

```
<html>
<head>
</head>
<frameset cols="20%,*,200" framespacing="1" frameborder="yes" border="1"
bordercolor="#99CCFF">
<frame src="ml.html" name="aa1">
<frame src="sl1.html" name="aa3">
<frame src="sl2.html" name="aa2" noresize="noresize">
</frameset><noframes></noframes>
</html>
```

9-5 浮动窗口<iframe>

<iframe>这标记只适用于 IE 浏览器。它的作用是在浏览器窗口中可以嵌入一个框窗以显示另一个文件。它是一个围堵标记,但围著的字句只有在浏览器不支援 iframe 标记时才会显示,如<noframes> 一样,可以放些提醒字句之类。通常 iframe 配合一个辨认浏览器的 JavaScript 会较好,若 JavaScript 认出该浏览器并非 Internet Explorer 便会切换至另一版本。

<iframe> 的参数设定格式:

```
<iframe src="iframe.html" name="test" align="MIDDLE" width="300" height="100"
marginwidth="1" marginheight="1" frameborder="1" scrolling="Yes">
```

<iframe>属性的含义

属 性	含 义
-----	-----

src	浮动窗框中的要显示的页面文件的路径,可以是相对或绝对。
name	此框窗名称,这是连结标记的 target 参数所 要的,
align	可选值为 left, right, top, middle, bottom, 作用不大
height	框窗的高,以 pixels 为单位。
width	框窗的宽,以 pixels 为单位。
marginwidth	该插入的文件与框边所保留的空间。
marginheight	该插入的文件与框边所保留的空间。
frameborder	使用 1 表示显示边框, 0 则不显示。(可以是 yes 或 no)
scrolling	使用 Yes 表示容许卷动(内定), No 则不容许卷动。

[sl95.html](#)

```
<html>
<head>
<title>浮动框架</title>
</head>
<body bgcolor="#E1FFE1">
<center>
<iframe src="ml1.html" name="aa" width="600" height="400" marginwidth="30"
marginheight="20" align="middle" allowtransparency="true">
这是一个浮动窗口
</iframe>
<p><a href="sl1.html" target="aa">爱在深秋</a></p>
<p><a href="sl2.html" target="aa">图像对文字的水平居中</a></p>
<p><a href="sl3.html" target="aa">图像与文本之间的距离</a></p>
<p><a href="sl4.html" target="aa">图像大小的设定</a></p>
<p><a href="http://www.sina.com.cn" target="aa3">新浪网站</a></p>
</center>
</body>
</html>
```

注意: Internet Explorer 5.5 支持浮动框架的内容透明。如果想要为浮动框架定义透明内容,则必须满足下列条件。

与 IFRAME 元素一起使用的 ALLOWTRANSPARENCY 标签属性必须设置为 true。

在 IFRAME 内容源文档, background-color 或 BODY 元素的 BGCOLOR 标签属性必须设置为 transparent。

第十章表单的设计

10-1 表单标记<form>

表单在 Web 网页中用来给访问者填写信息,从而能采集客户端信息,使网页具有交互的功能。一般是将表单设计在一个 Html 文档中,当用户填写完信息后做提交(submit)操作,于是表单的内容就从客户端的浏览器传送到服务器上,经过服务器上的 ASP 或 CGI 等处理程序处理后,再将用户所需信息传送回客户端的浏览器上,这样网页就具有了交互性。这里我们只讲怎样使用 Html 标志来设计表单。

表单是由窗体和控件组成的,一个表单一般应该包含用户填写信息的输入框,提交和按钮等,这些输入框,按钮叫做控件,表单很像容器,它能够容纳各种各样的控件。

一个表单用<form></form>标志来创建。也即定义表单的开始和结束位置,在开始和结束标志之间的一切定义都属于表单的内容。<form>标志具有 action、method 和 target 属性。action 的值是处理程序的程序名(包括 网络路径:网址或相对路径),如: <form action="用来接收表单信息的 url">,如果这个属性是空值("")则当前文档的 url 将被使用.当用户提交表单时,服务器将执行网址里面的程序(一般是 CGI 程序)。method 属性用来定义处理程序从表单中获得信息的方式,可取值为 GET 和 POST 的其中一个。GET 方式是处理程序从当前 Html 文档中获取数据,然而这种方式传送的数据量是有所限制的,一般限制在 1KB (255 个字节) 以下。POST 方式与 GET 方式相反,它是当前的 Html 文档把数据传送给处理程序,传送的数据量要比使用 GET 方式的大的多。target 属性用来指定目标窗口或目标帧。可选当前窗口_self,父级窗口_parent,顶层窗口_top,空白窗口_blank。

表单标签的格式: <FORM action="url" method=get|post name="myform" target="_blank">...</FORM>

10-2 写入标记<input>

在 html 语言中,标记<input>具有重要的地位,它能够将浏览器中的控件加载到 html 文档中,该标记是单个标记,没有结束标记。<input type="">标志用来定义一个用户输入区,用户可在其中输入信息。此标志必须放在 <form></form>标志对之间。<input type="">标志中共提供了九种类型的输入区域,具体是哪一种类型由 type 属性来决定。请看下边列表:

type 属性值定义

type 属性取值	输入区域类型	控件的属性及说明
<input type="TEXT" size="" maxlength="">	单行的文本输入区域,size 与 maxlength 属性用来定义此种输入区域显示的尺寸大小与输入的最大字符数	(1)name 定义控件名称 (2)value 指定控件初始值,该值就是浏览器被打开时在文本框中的内容 (3)size 指定控件宽度,表示该文本输入框所能显示的最大字符数。 (4)maxlength 表示该文本输入框允许用户输入的最大字符数。 (5)onchang 当文本改变时要执行的函数 (6)onselect 当控件被选中时要执行的函数 (7)onfocus 当文本接受焦点时要执行的函数
<input type="button">	普通按钮,当这个按钮被点击时,就会调用属性 onclick 指定的函数;在使用这个按钮时,一般配合使用 value 指定在它上面显示的文字,用 onclick 指定一个函数,一般为 JavaScript 的一个事件。	这三个按钮有下面共同的属性: (1)name 指定按钮名称 (2)value 指定按钮表面显示的文字 (3)onclick 指定单击按钮后要调用的函数
<input type="SUBMIT">	提交到服务器的按钮,当这个按钮被点击时,就会连接到表单 form 属性 action 指定的 url 地址。	(4)onfocus 指定按钮接受焦点时要调用的函数

<input type="RESET">	重置按钮,单击该按钮可将表单内容全部清除,重新输入数据。	
<input type="CHECKBOX" checked>	一个复选框,checked 属性用来设置该复选框缺省时是否被选中,右边示例中使用了三个复选框	checkbox 用于多选,有以下属性: (1) name 定义控件名称 (2) value 定义控件的值 (3) checked 设定控件初始状态是被选中的 (4) onclick 定义控件被选中时要执行的函数 (5) onfocus 定义控件为焦点时要执行的函数
<input type="HIDDEN">	隐藏区域,用户不能在其中输入,用来预设某些要传送的信息	hidden 隐藏控件,用于传递数据,对用户来说是不可见的;属性有: (1)name 控件名称, (2)value 控件默认值 (3)hidden 隐藏控件的默认值会随表单一起发送给服务器,例如: <input type="Hidden" name="ss" value="688"> 控件的名称设置为 ss,设置其数据为"688",当表单发送给服务器后,服务器就可以根据 hidden 的名称 ss,读取 value 的值 688;
<input type="IMAGE" src="URL">	使用图像来代替 Submit 按钮,图像的源文件名由 src 属性指定,用户点击后,表单中的信息和点击位置的 X、Y 坐标一起传送给服务器	(1)name 指定图像按钮名称 (2)src 指定图像的 url 地址
<input type="PASSWORD">	输入密码的区域,当用户输入密码时,区域内将会显示 "*"号	password 口令控件 表示该输入项的输入信息是密码,在文本输入框中显示 "*",属性有: (1)name 定义控件名称 (2)value 指定控件初始值,该值就是浏览器被打开时在文本框中的内容 (3)size 指定控件宽度,表示该文本输入框所能显示的最大字符数。 (4)maxlegnth 表示该文本输入框允许用户输入的最大字符数。
<input type="RADIO">	单选按钮类型,checked 属性用来设置该单选框缺省时是否被选中,右边示例中使用了三个单选框	radio 用于单选, 有以下属性: (1) name 定义控件名称 (2) value 定义控件的值 (3) checked 设定控件初始状态是被选中的 (4) onclick 定义控件被选中时要执行的函数 (5) onfocus 定义控件为焦点时要执行的函数 当为单项时,所有按钮的 name 属性必需相同,如:都设置为 my_radio。

以上类型的输入区域有一个公共的属性 **name**, 此属性给每一个输入区域一个名字。这个名字与输入区域是一一对应的, 即一个输入区域对应一个名字。服务器就是通过调用某一输入区域的名字的 **value** 值来获得该区域的数据的。而 **value** 属性是另一个公共属性, 它可用来指定输入区域的缺省值。

应用格式

<input 属性 1 属性 2.....>

常用属性:

1 name 控件名称

2 type 控件类型 如: **button** 普通按钮, **text** 文本框等

3 align 指定对齐方式,可取 top, bottom, middle

4 size 指定控件的宽度

5 value 用于设定输入默认值

6 maxlength 在单行文本的时候允许输入的最大字符数

7 src 插入图像的地址

8 event 指定激发的事件

实例: [10-2.html](#)

```
<html>
```

```
<head>
```

```
<title>&lt;input&gt; 的控件</title>
```

```
</head>
```

```
<body>
```

```
<center>
```

```
<h2><font color="#339933">&lt;input&gt; 控件的使用</font></h2>
```

```
</center>
```

```
<pre>
```

```
<form action="" method="post" target="_parent">
```

单行的文本输入区域: <INPUT class="text" name=T1>

普通按钮: <INPUT class="text" name=B1 type=submit value=Submit>

提交按钮: <INPUT class="text" name=B1 type=submit value=Submit>

重置按钮: <INPUT name=B1 type=reset value=Reset>

复选框: 你喜欢哪些教程: <INPUT name=C1 type=checkbox value=ON> Html 入门 <INPUT

CHECKED name=C2 type=checkbox value=ON> 动态 Html <INPUT name=C3

type=checkbox value=ON> ASP

图像来代替 Submit 按钮: <INPUT border=0 height=20 name=I2 src="../../image/nnn.gif"

type=image width=65>

密码的区域: <INPUT class="text" name=p1

type=password> </P>

单选按钮: 你的休闲爱好是什么: <INPUT CHECKED name=R1 type=radio value=V1> 音乐

<INPUT name=R1 type=radio value=V2> 体育 <INPUT name=R1 type=radio value=V3>

旅游

```
</form>
```

```
</pre>
```

```
<a href="#" onClick="window.history.back()"><FONT size=4>返回</FONT></A></SUB>
```

```
</pre>
```

```
</body>
```

```
</html>
```

10-3 菜单下拉列表框<select></select><option>

`<select></select>` 标志对用来创建一个菜单下拉列表框。此标志对用于 `<form></form>` 标志对之间。`<select>` 具有 `multiple`、`name` 和 `size` 属性。`multiple` 属性不用赋值，直接加入标志中即可使用，加入了此属性后列表框就成了可多选的了；`name` 是此列表框的名字，它与上边讲的 `name` 属性作用是一样的；`size` 属性用来设置列表的高度，缺省时值为 1，若没有设置(加入)`multiple` 属性，显示的将是一个弹出式的列表框。

`<option>` 标志用来指定列表框中的一个选项，它放在 `<select></select>` 标志对之间。此标志具有 `selected` 和 `value` 属性，`selected` 用来指定默认的选项，`value` 属性用来给 `<option>` 指定的那一个选项赋值，这个值是要传送到服务器上的，服务器正是通过调用 `<select>` 区域的名字的 `value` 属性来获得该区域选中的数据项的。

实例：[10-3-1.html](#)

```
<html>
<head>
<title>下拉列表框</title>
</head>
<body>
<form action="" method="post">
<p>请选择最喜欢的男歌星:
<select name="gx1" size="1">
<option value="ldh">刘德华
<option value="zhxy" selected>张学友
<option value="gfch">郭富城
<option value="lm">黎明
</select>
</form>
</body>
</html>
```

实例：[10-3-2.html](#)

```
<html>
<head>
<title>复选列表框</title>
</head>
<body>
<form action="" method="post">
<p>请选择最喜欢的女歌星:
<select name="gx2" multiple size="4">
<option value="zhmy">张曼玉
<option value="wf" selected>王菲
<option value="tzh">田震
<option value="ny">那英
</select>
</form>
</body>
</html>
```

10-4 多行的文本框.`<textarea></textarea>`

`<textarea></textarea>`用来创建一个可以输入多行的文本框，此标志对用于`<form></form>`标志对之间。`<textarea>`具有以下属性：

(1)`onchange` 指定控件改变时要调用的函数

(2)`onfocus` 当控件接受焦点时要执行的函数

(3)`onblur` 当控件失去焦点时要执行的函数

(4)`onselect` 当控件内容被选中时要执行的函数

(5)`name` 这文字区块的名称，作识别之用，将会传及 CGI。

(6)`cols` 这文字区块的宽度。

(7)`rows` 这文字区块的列数，即其高度。

(8)`wrap` 属性 定义输入内容大于文本域时显示的方式，可选值如下：

*默认值是文本自动换行；当输入内容超过文本域的右边界时会自动转到下一行，而数据在被提交处理时自动换行的地方不会有换行符出现；

*`Off`，用来避免文本换行，当输入的内容超过文本域右边界时，文本将向左滚动；

*`Virtual`，允许文本自动换行。当输入内容超过文本域的右边界时会自动转到下一行，而数据在被提交处理时自动换行的地方不会有换行符出现；

*`Physical`，让文本换行，当数据被提交处理时换行符也将被一起提交处理。

这里列与行是以字符数为单位的。

实例：[10-4.html](#)

```
<html>
<head>
<title>多行的文本框</title>
</head>
<body>
<form action="" method="post">
<p>您的意见对我很重要:
<textarea name="yj" cols="20" rows="5">
请将意见输入此区域
</textarea>
</form>
</body>
</html>
```

序言

第一章 HTML 基本结构及实体

本章目标: 了解 HTML 文档的基本结构
掌握 HTML 结构标签<html><head><title><body>
掌握 HTML 字符实体
本章重点: 了解 HTML 文档的基本结构
本章难点: HTML 字符实体的使用

一、HTML 文档的基本结构

HTML 文件是什么?

- ☐ HTML 表示超文本标记语言 (Hyper Text Markup Language)。
- ☐ HTML 文件是一个包含标记的文本文件。
- ☐ 这些标记告诉浏览器怎样显示这个页面。
- ☐ HTML 文件必须有 htm 或者 html 扩展名。
- ☐ HTML 文件可以用一个简单的文本编辑器创建。

想不想尝试一下?

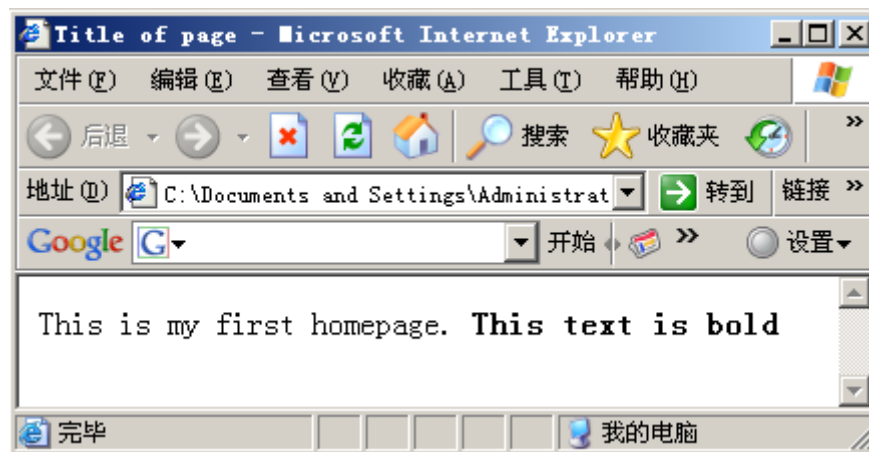
假如你运行的是 windows 系统, 打开记事本, 在其中输入以下文本:

```
<html>
<head>
  <title>Title of page</title>
</head>
<body>
  This is my first homepage.
  <b>This text is bold</b>
</body>
</html>
```

将此文件保存为 “mypage.htm”。

启动浏览器。在文件菜单中选择“打开”(或者“打开页面”), 这时将出现一个对话框。选择“浏览”(或者“选择文件”), 定位到你刚才创建的 HTML 文件——“mypage.htm”, 选择它, 单击“打开”。然后在对话框中, 你将看到这个文件的地址, 比如说: “C:\MyDocuments\mypage.htm”。单击“确定”, 浏览器将显示此页

面。



例子解释:

HTML 文档中, 第一个标签是<html>。这个标签告诉浏览器这是 HTML 文档的开始。HTML 文档的最后一个标签是</html>, 这个标签告诉浏览器这是 HTML 文档的终止。

在<head>和</head>标签之间文本的是头信息。在浏览器窗口中, 头信息是不被显示的。

在<title>和</title>标签之间的文本是文档标题, 它被显示在浏览器窗口的标题栏。

在<body>和</body>标签之间的文本是正文, 会被显示在浏览器中。

在和标签之间的文本会以加粗字体显示。

关于 HTML 编辑器:

用一些所见即所得的编辑器, 比如 frontpage, dreamwaver, 你可以很容易创建一个页面, 而不需要在纯文本中编写代码。

但是假如你想成为一名熟练的网络开发者, 我们强烈推荐你用纯文本编辑器编写代码, 这有助于学习 HTML 基础。

常见问题:

问: 我编写完了 HTML 文件, 但是不能在浏览器中看见结果, 为什么?

答: 请确认你保存了文件, 并且使用了正确的文件名和扩展名, 例如: “c:\mypage.htm”, 并且确认你用浏览器打开时使用同样的文件名。

问: 我编辑了 HTML 文件, 但是修改结果并没有在浏览器中显示, 为什么?

答: 浏览器缓存了你的页面, 所以它不需要两次读取同样的页面。你修改了这个页面, 浏览器并不知道。请使用“刷新/重载”按钮来强迫浏览器读取编辑过的页面。

HTML 元素:

HTML 文档是由 HTML 元素组成的文本文件。

HTML 元素是预定义的正在使用的 HTML 标签。

HTML 标签:

HTML 标签用来组成 HTML 元素。

HTML 标签两端有两个包括字符:“<”和“>”,这两个包括字符被称为角括号。

HTML 标签通常成对出现,比如和。一对标签的前面一个是开始标签,第二个是结束标签,在开始和结束标签之间的文本是元素内容。

HTML 标签是大小写无关的,跟表示的意思是一样的。

HTML 元素:

回忆一下上面的 HTML 例子:

```
<html>
<head>
  <title>Title of page</title>
</head>
<body>
  This is my first homepage.
  <b>This text is bold</b>
</body>
</html>
```

下面是一个 HTML 元素:

```
<b>This text is bold</b>
```

此 HTML 元素以开始标签起始, 内容是: This text is bold, 以结束标签中止。
标签的目的是定义一个需要被显示成粗体的 HTML 元素。

下面也是一个 HTML 元素:

```
<body>
This is my first homepage.
<b>This text is bold</b>
</body>
```

此 HTML 标签以开始标签<body>起始, 终止于结束标签</body>。<body>标签的目的是定义一个 HTML 元素, 使其包含 HTML 文档的主体。

为什么使用小写标签?

我们刚说过, HTML 标签是大小写无关的: 跟含义相同。当你上网的时候, 你会注意到多数教程在示例中使用大写的 HTML 标签, 我们总是使用小写标签。为什么?

假如你想投入到下一代 HTML 中, 你应该开始使用小写标签。W3C 在他们的 HTML4 建议中提倡使用小写标签, XHTML (下一代 HTML) 也需要小写标签。

标签属性:

标签可以拥有属性。属性能够为页面上的 HTML 元素提供附加信息。

标签<body>定义了 HTML 页面的主体元素。使用一个附加的 bgcolor 属性, 你可以告诉浏览器: 你页面的背景色是红色的, 就像这样:

```
<body bgcolor="red">
```

标签<table>定义了一个 HTML 表格。使用一个附加的 border 属性, 你可以告诉浏览器: 这个表格是没有边框的, 代码是:

```
<table border="0">
```

属性通常由属性名和值成对出现, 就像这样: name="value"。属性通常是附加给 HTML 元素的开始标签的。

引号样式:

属性值应该被包含在引号中。双引号是最常用的, 但是单引号也可以使用。

在很少情况下, 比如说属性值本身包含引号, 使用单引号就很必要了。

比如: name='John "ShotGun" Nelson'。

注意: 中文引号跟英文引号是不一样的。上面所指的引号都是英文状态下的引号。

二、 HTML 实体

有些字符, 比如说“<”字符, 在 HTML 中有特殊的含义, 因此不能在文本中使用。

想要在 HTML 中显示一个小于号“<”, 需要用到字符实体。

字符实体:

在 HTML 中,有些字符拥有特殊含义,比如小于号“<”定义为一个 HTML 标签的开始。假如我们想要浏览器显示这些字符的话,必须在 HTML 代码中插入字符实体。

一个字符实体拥有三个部分:一个 and 符号 (&),一个实体名或者一个实体号,最后是一个分号 (;)

想要在 HTML 文档中显示一个小于号,我们必须这样写: <或者<

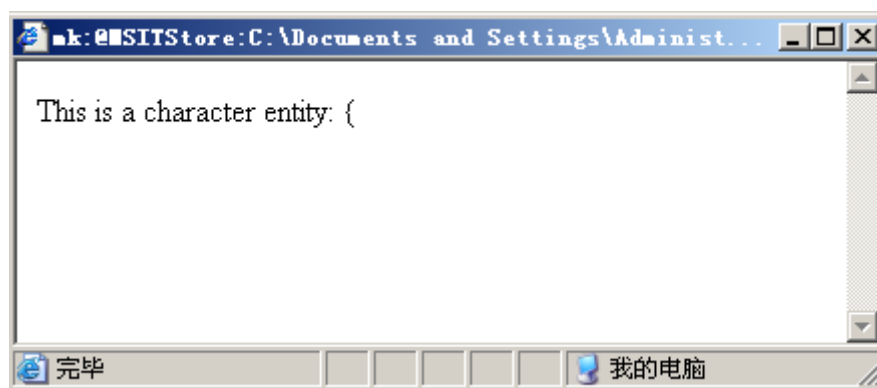
使用名字相对于使用数字的优点是容易记忆,缺点是并非所有的浏览器都支持最新的实体名,但是几乎所有的浏览器都能很好地支持实体号。

注意:实体名是大小写敏感的。

下面这个例子能够让你针对 HTML 实体实践一下。

```
<html>
<body>
  <p>This is a character entity: &#123;</p>
</body>
</html>
```

运行代码,结果如下:



不可拆分的空格

在 HTML 中,最常见的字符实体就是不可拆分空格。

通常,HTML 会合并你文档中的空格。假如在你的 HTML 文本中连续写了 10 个空格,其中 9 个会被去掉。想要在 HTML 中插入空格,可以使用实体:

最常用的字符实体:

显示结果	描述	实体名	实体号
	不可拆分的空格	 	
<	小于	<	<
>	大于	>	>
&	and符号	&	&
"	引号	"	"
'	单引号		'

其他一些常用的字符实体:

显示结果	描述	实体名	实体号
¢	分	¢	¢
£	英镑	£	£
¥	人民币元	¥	¥
§	章节	§	§
©	版权	©	©
®	注册	®	®
×	乘号	×	×
÷	除号	÷	÷

第二章 HTML基本元素的运用

本章目标: 掌握下列标签:

段落相关标签<p>
<hr>

格式化相关标签<small><sub><sup><pre>

列表相关标签

图片相关标签

超链相关标签<a>

本章重点: 段落相关标签, 超链标签

本章难点: 超链相关标签<a>

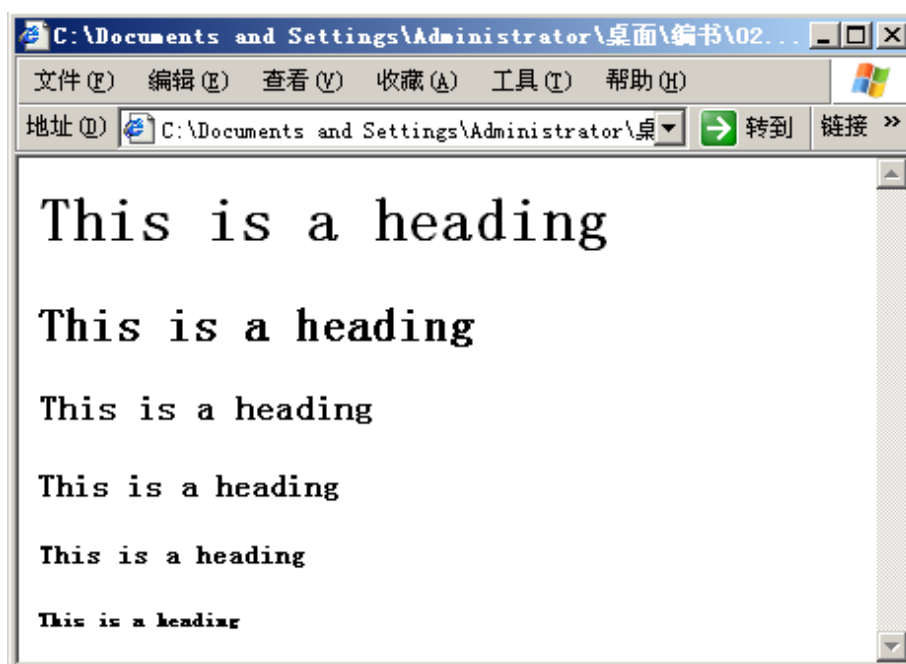
一、 段落相关标签

标题元素:

标题元素由标签<h1>到<h6>定义。<h1>定义了最大的标题元素, <h6>定义了最小的。

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
<h4>This is a heading</h4>
<h5>This is a heading</h5>
<h6>This is a heading</h6>
```

运行代码, 结果如下:



HTML 自动在一个标题元素前后各添加一个空行。

段落:

段落是用<p>标签定义的。

```
<p>This is another paragraph</p>
```

运行代码, 结果如下:



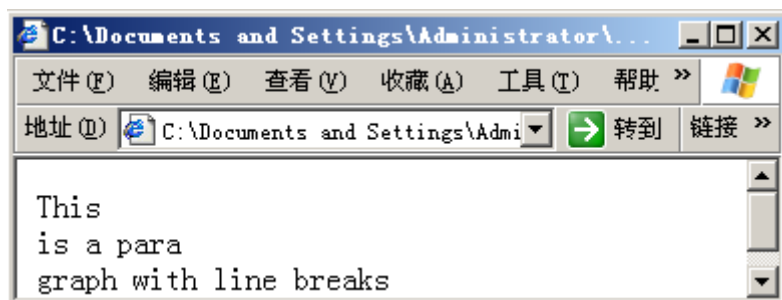
HTML 自动在一个段落前后各添加一个空行。

换行:

当需要结束一行, 并且不想开始新段落时, 使用
标签。
标签不管放在什么位置, 都能够强制换行。

```
<p>This <br> is a para<br>graph with line breaks</p>
```

运行代码, 结果如下:



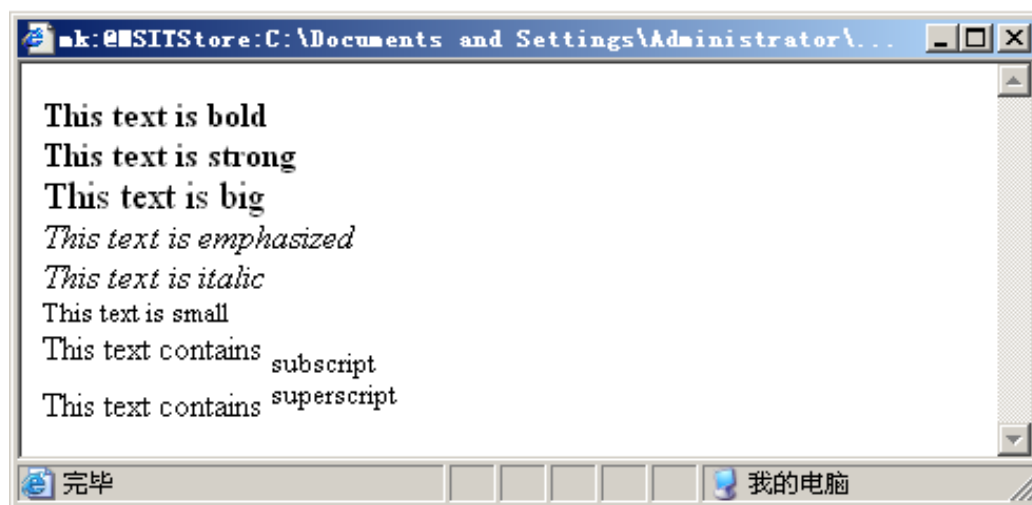
标签是一个空标签，它没有结束标记。

二、 格式化相关标签

格式化文字:

```
<html>
<body>
  <b>This text is bold</b><br>
  <strong>
    This text is strong
  </strong><br>
  <big>
    This text is big
  </big><br>
  <em>
    This text is emphasized
  </em><br>
  <i>
    This text is italic
  </i><br>
  <small>
    This text is small
  </small><br>
  This text contains
  <sub>
    subscript
  </sub><br>
  This text contains
  <sup>
    superscript
  </sup>
</body>
</html>
```


运行代码, 结果如下:



这个例子说明了在 HTML 里面可以怎样格式化文本。

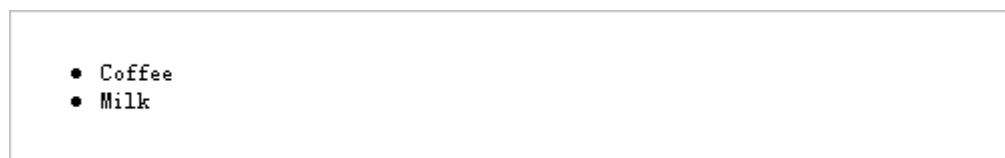
三、 列表相关标签

无序列表:

无序列表是一个项目的序列。各项目前加有标记 (通常是黑色的实心小圆圈)。无序列表以标签开始。每个列表项目以开始。

```
<ul>
  <li>Coffee</li>
  <li>Milk</li>
</ul>
```

运行代码, 结果如下:



无序列表的项目中可以加入段落、换行、图像、链接, 其他的列表等等。

有序列表:

有序列表也是一个项目的序列。各项目前加有数字作标记。有序列表以标签开始。每个列表项目以开始。

```
<ol>
```

```
<li>Coffee</li>
<li>Milk</li>
</ol>
```

运行代码, 结果如下:

```
1. Coffee
2. Milk
```

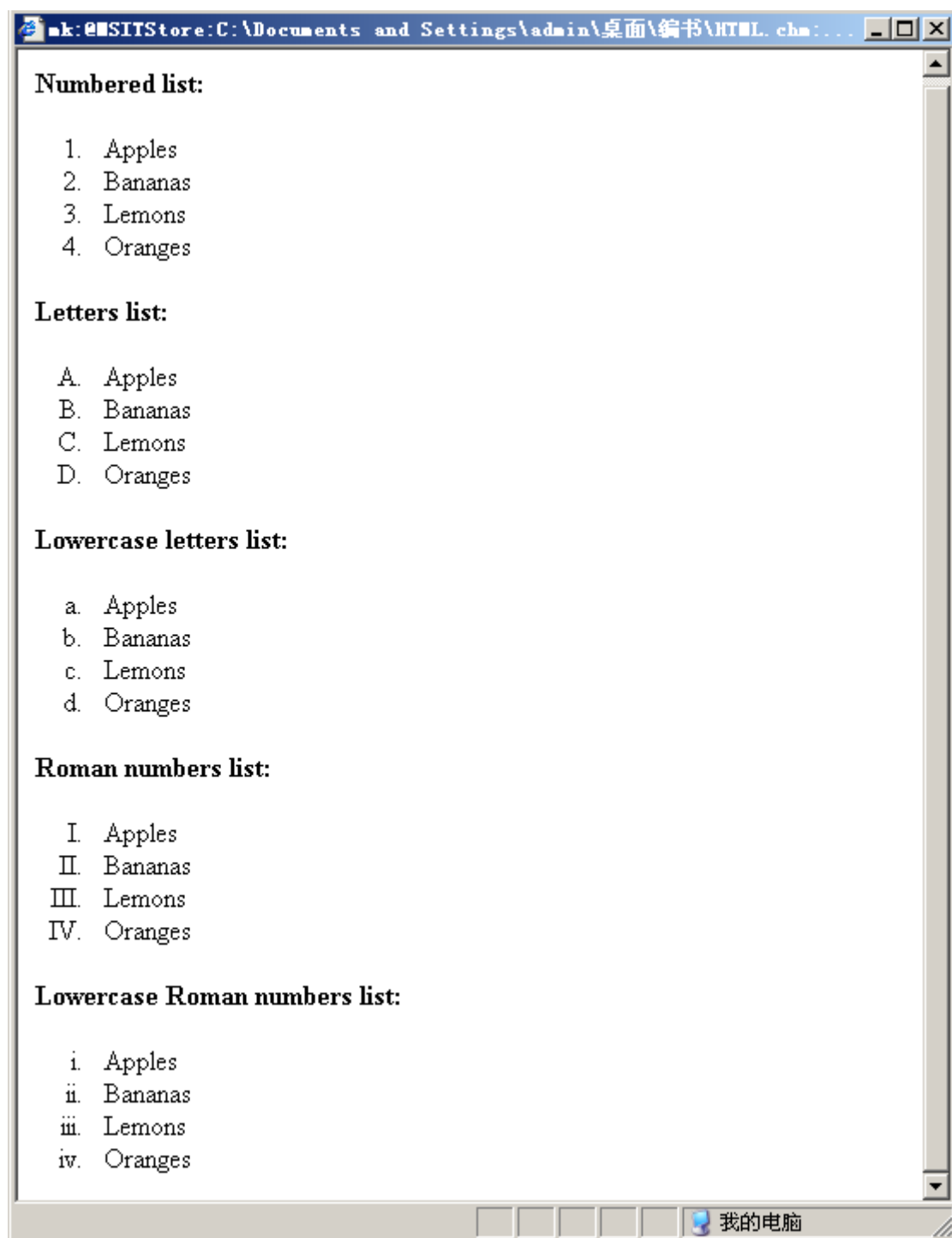
更多示例:

有序列表的不同类型:

```
<html>
<body>
<h4>Numbered list:</h4>
<ol>
  <li>Apples</li>
  <li>Bananas</li>
  <li>Lemons</li>
  <li>Oranges</li>
</ol>
<h4>Letters list:</h4>
<ol type="A">
  <li>Apples</li>
  <li>Bananas</li>
  <li>Lemons</li>
  <li>Oranges</li>
</ol>
<h4>Lowercase letters list:</h4>
<ol type="a">
  <li>Apples</li>
  <li>Bananas</li>
  <li>Lemons</li>
  <li>Oranges</li>
</ol>
<h4>Roman numbers list:</h4>
<ol type="I">
  <li>Apples</li>
```

```
<li>Bananas</li>
<li>Lemons</li>
<li>Oranges</li>
</ol>
<h4>Lowercase Roman numbers list:</h4>
<ol type="i">
  <li>Apples</li>
  <li>Bananas</li>
  <li>Lemons</li>
<li>Oranges</li>
</ol>
</body>
</html>
```

运行代码，结果如下：



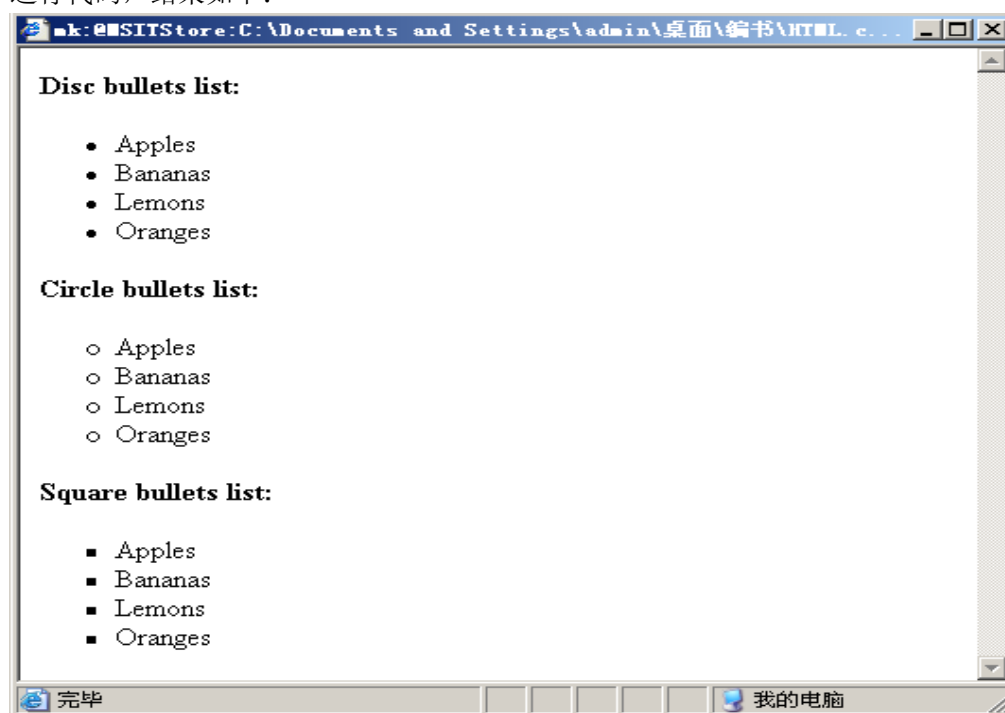
这个例子显示了有序列表的不同类型。

无序列表的不同类型:

```
<html>
```

```
<body>
<h4>Disc bullets list:</h4>
<ul type="disc">
  <li>Apples</li>
  <li>Bananas</li>
  <li>Lemons</li>
  <li>Oranges</li>
</ul>
<h4>Circle bullets list:</h4>
<ul type="circle">
  <li>Apples</li>
  <li>Bananas</li>
  <li>Lemons</li>
  <li>Oranges</li>
</ul>
<h4>Square bullets list:</h4>
<ul type="square">
  <li>Apples</li>
  <li>Bananas</li>
  <li>Lemons</li>
  <li>Oranges</li>
</ul>
</body>
</html>
```

运行代码, 结果如下:



这个例子显示了无序列表的不同类型。

四、 图片相关标签

Img 标签和 src 属性:

在 HTML 里面, 图像是由标签定义的。

是空标签, 意思是说, 它只拥有属性, 而没有结束标签。

想要在页面上显示一个图像, 需要使用 src 属性。“src”表示“源”的意思。“src”属性的值是所要显示图像的 URL。

插入图像的语法:

```

```

URL 指向图像存储的地址。网站“www.w3schools.com”子目录“images”中的图像“boat.gif”的 URL 如下: “http://www.w3schools.com/images/boat.gif”。

当浏览器在文档中遇到 img 标签时, 就放置一个图像。如果把 img 标签放在两个段落之间, 就会先显示一个段落, 然后是这个图像, 最后是另外一个段落。

alt 属性:

alt 属性用来给图像显示一个“交互文本”。alt 属性的值是由用户定义的。

```

```

“alt”属性在浏览器装载图像失败的时候告诉用户所丢失的信息, 此时, 浏览器显示这个“交互文本”来代替图像。给页面上的图像都加上 alt 属性是一个好习惯, 它有助于更好地显示信息, 而且, 对纯文本浏览器很有用。

基本注意点——有用的技巧:

如果一个 HTML 文档包含 10 个图像, 那么为了正确显示这个页面, 需要加载 11 个文件。加载图像是需要时间的, 所以请谨慎使用图像。

更多示例:

调整图像大小:

```
<html>
<body>
  <p>
```

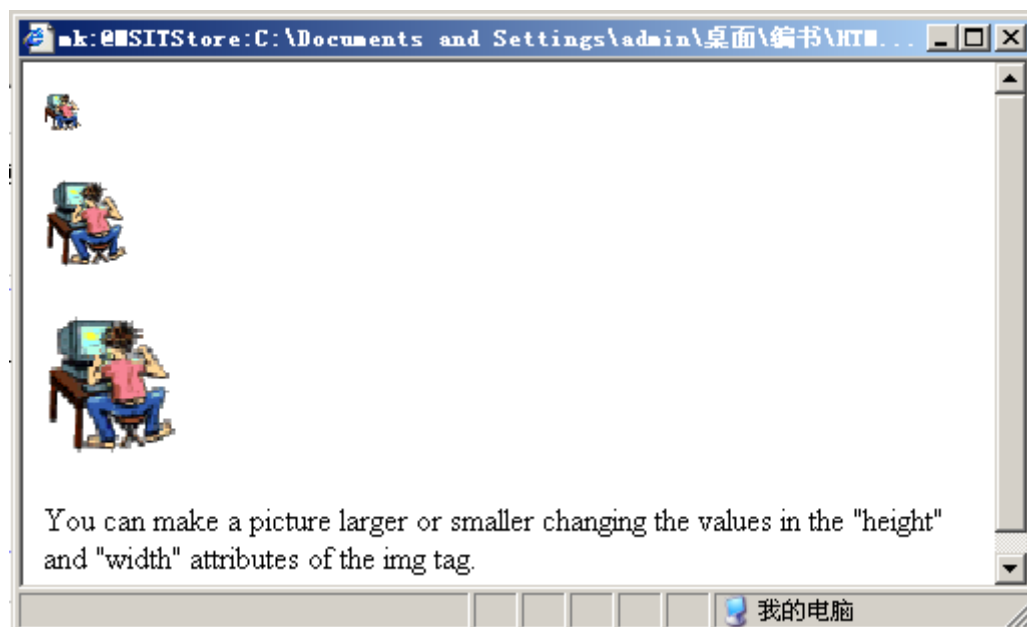
```

</p>
<p>

</p>
<p>

</p>
<p>
  You can make a picture larger or smaller changing the values in the "height"
  and "width" attributes of the img tag.
</p>
</body>
</html>
```

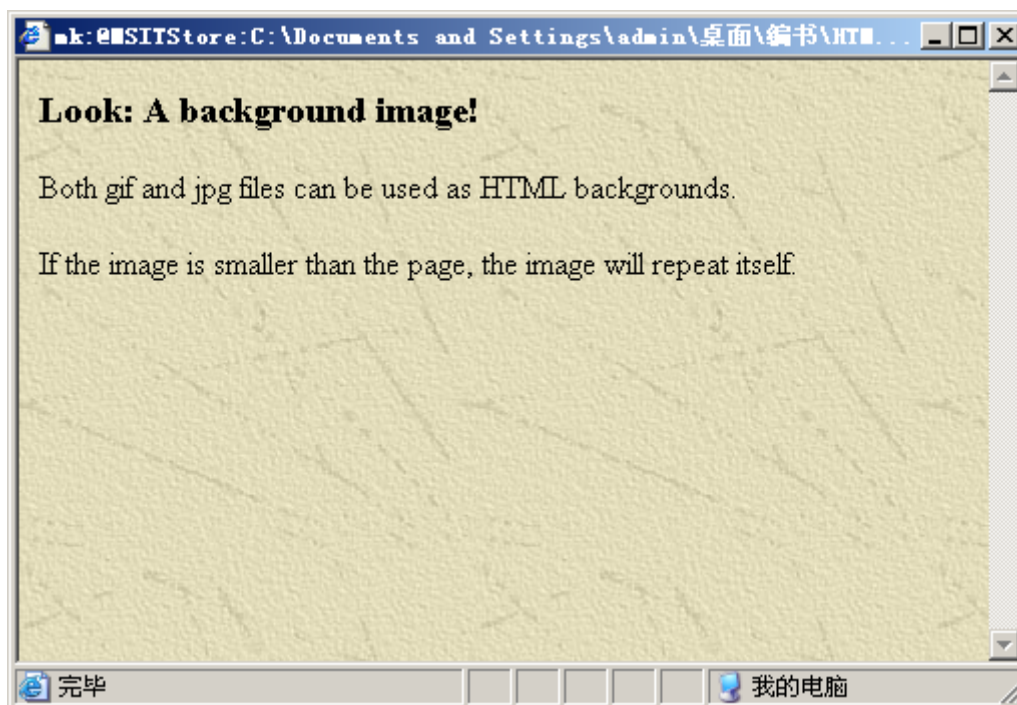
运行代码, 结果如下:



背景图像:

```
<html>
<body background="./images/background.jpg">
  <h3>Look: A background image!</h3>
  <p>Both gif and jpg files can be used as HTML backgrounds.</p>
  <p>If the image is smaller than the page, the image will repeat itself.</p>
</body>
</html>
```

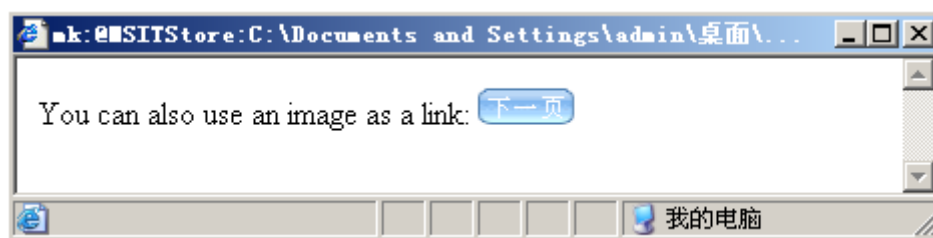
运行代码, 结果如下:



图像链接:

```
<html>
<body>
  <p>
    You can also use an image as a link:
    <a href="back.htm">
      
    </a>
  </p>
</body>
</html>
```

运行代码, 结果如下:



五、 超链相关标签

锚标签和 href 属性:

HTML 使用锚标签 (<a>) 来创建一个连接到其他文件的链接。锚可以指向网络上的任何资源: HTML 页面, 图像, 声音, 影片等等。

创建一个锚的语法:

```
<a href="url">Text to be displayed</a>
```

锚可以指向网络上的任何资源: HTML 页面, 图像, 声音, 影片等等。

标签<a>被用来创建一个链接指向的锚, href 属性用来指定连接到的地址, 在锚的起始标签<a>和结束标签中间的部分将被显示为超级链接。

这个锚定义了一个到 W3Schools 的链接:

```
<a href="http://www.w3schools.com/">Visit W3Schools!</a>
```

上面这段代码在浏览器中显示的效果如下:

```
Visit W3Schools!
```

target 属性:

使用 target 属性, 你可以定义从什么地方打开链接地址。

下面这段代码打开一个新的浏览器窗口来打开链接:

```
<a href="http://www.w3schools.com/" target="_blank">Visit W3Schools!</a>
```

锚标签和 name 属性

name 属性用来创建一个命名的锚。使用命名锚以后, 可以让链接直接跳转到一个页面的某一章节, 而不用用户打开那一页, 再从上到下慢慢找。

下面是命名锚的语法:

```
<a name="label">Text to be displayed</a>
```

你可以为锚随意指定名字, 只要你愿意。下面这行代码定义了一个命名锚:

```
<a name="tips">Useful Tips Section</a>
```

你应该注意到了: 命名锚的显示方式并没有什么与众不同的。

想要直接链接到“tips”章节的话, 在 URL 地址的后面加一个“#”和这个锚的名字, 就像这样:

```
<a name="http://www.w3schools.com/html_links.asp#tips">Jump to the Useful Tips Section</a>
```

一个链接到本页面中某章节的命名锚是这样写的:

```
<a name="#tips">Jump to the Useful Tips Section</a>
```

基本注意点——有用的技巧:

尽量在子目录路径后面加一个左斜杠。假如你像下面这样写:
`href="http://www.w3schools.com/html"`, 将会产生向服务器产生两个 HTTP 请求, 因为服务器会在后面追加一个左斜杠, 产生一个新的请求, 就像这样:
`href="http://www.w3schools.com/html/"`。

命名锚通常用来在大型文档的开头创建章节表。这个页面的每个章节被加上一个命名锚, 到这些锚的链接被放在页面的顶端。

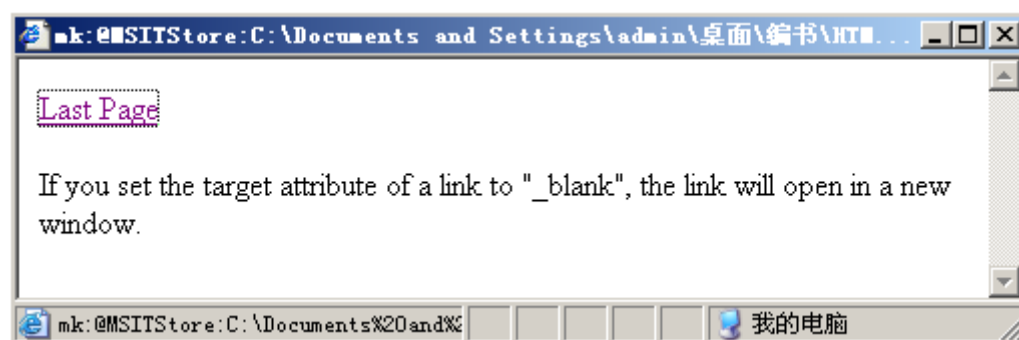
如果浏览器无法找到指定的命名锚, 它将转到这个页面的顶部, 而不显示任何错误提示。

更多示例:

在新浏览器窗口中打开链接:

```
<html>
<body>
<a href="lastpage.htm" target="_blank">Last Page</a>
<p>
If you set the target attribute of a link to "_blank",
the link will open in a new window.
</p>
</body>
</html>
```

运行代码, 结果如下:



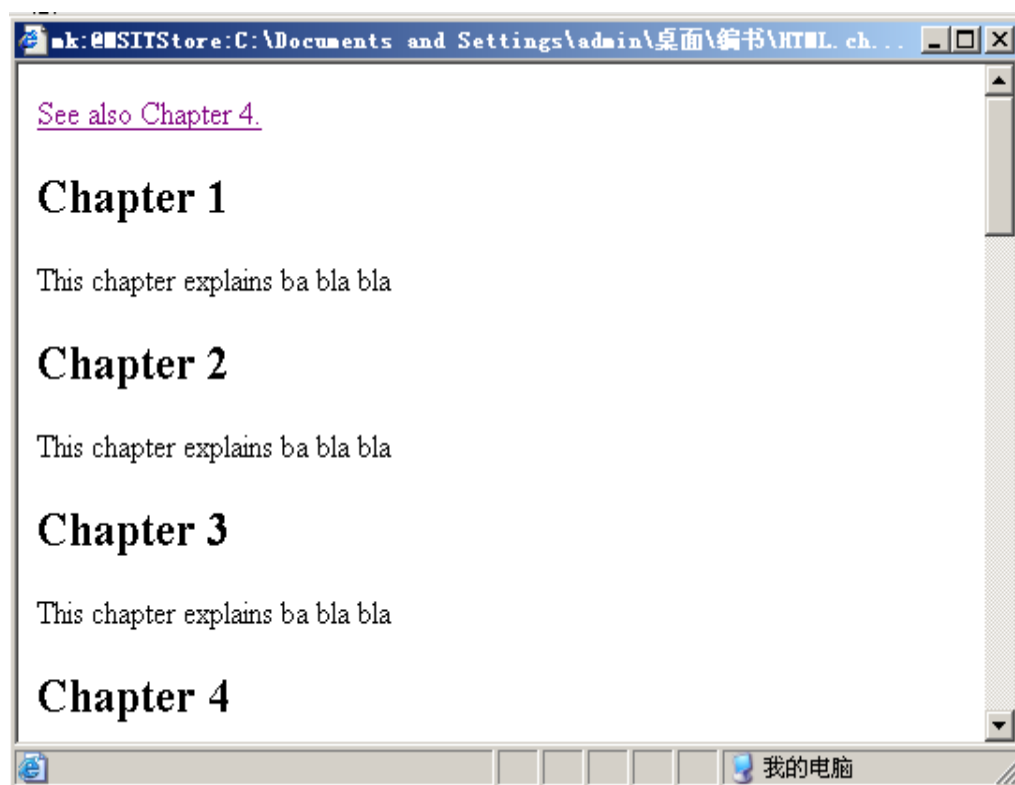
单击超连接, 打开一个新窗口:



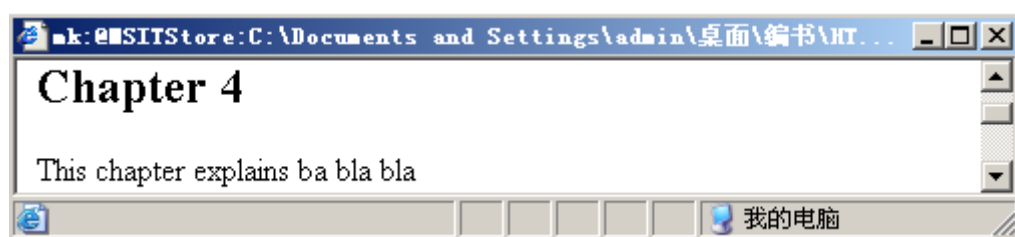
链接到本页面的某个位置:

```
<html>
<body>
  <p>
    <a href="#C4">
      See also Chapter 4.
    </a>
  </p>
  <p>
    <h2>Chapter 1</h2>
    <p>This chapter explains ba bla bla</p>
    <h2>Chapter 2</h2>
    <p>This chapter explains ba bla bla</p>
    <h2>Chapter 3</h2>
    <p>This chapter explains ba bla bla</p>
    <a name="C4"><h2>Chapter 4</h2></a>
    <p>This chapter explains ba bla bla</p>
  </body>
</html>
```

运行代码, 结果如下:



单击超连接,



第三章 用HTML创建表格

本章目标: 了解掌握表格的基本结构<table><tr><th><td>
掌握跨行、跨列属性 colspan rowspan
掌握表格相关修饰属性 border width height bgcolor
background height cellpadding cellspacing
本章重点: 掌握表格的基本结构及相关属性
本章难点: 掌握跨行、跨列属性 colspan rowspan

一、HTML 表格

表格:

表格是用<table>标签定义的。表格被划分为行(使用<tr>标签), 每行又被划分为数据单元格(使用<td>标签)。td 表示“表格数据”(Table Data), 即数据单元格的内容。数据单元格可以包含文本, 图像, 列表, 段落, 表单, 水平线, 表格等等。想不想尝试一下?

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

在浏览器中显示如下:

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

表格和 border 属性:

```
<table border="1">
<tr>
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td>
</tr>
</table>
```

如果不指定 border 属性, 表格将不显示边框。有时候这很有用, 但是多数时候我们希望

显示边框。

表格头:

表格头使用<th>标签指定。

```
<table border="1">
<tr>
<th>Heading</th>
<th>Another Heading</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

在浏览器中显示如下:

Heading	Another Heading
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

表格中的空单元格

在多数浏览器中, 没有内容的单元格显示得不太好。

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td></td>
</tr>
</table>
```

在浏览器中显示如下:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

注意一下空单元格的边框没有显示出来。为了避免这个，可以在空单元格里加入不可分空格来占位，这样边框能正常显示。

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>&nbsp;</td>
</tr>
</table>
```

在浏览器中显示如下:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

基本注意点——有用的技巧

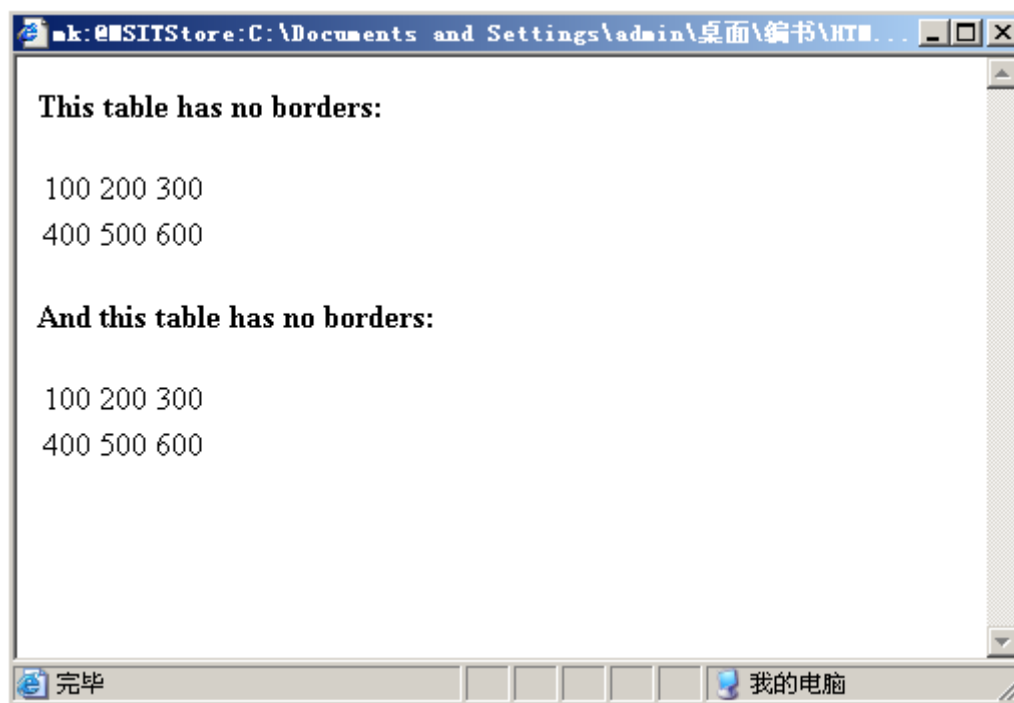
通常很少使用<thead>, <tbody>, <tfoot>标签, 因为浏览器对它们的支持不好。希望这个在 XHTML 的未来版本中得到改变。

更多示例:

没有边框的表格:

```
<html>
<body>
<h4>This table has no borders:</h4>
<table>
<tr>
<td>100</td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td>400</td>
<td>500</td>
<td>600</td>
</tr>
</table>
<h4>And this table has no borders:</h4>
<table border="0">
```

```
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>
</body>
</html>
```



表格头:

```
<html>
<body>
<h4>Table headers:</h4>
<table border="1">
<tr>
  <th>Name</th>
  <th>Telephone</th>
  <th>Telephone</th>
</tr>
<tr>
```

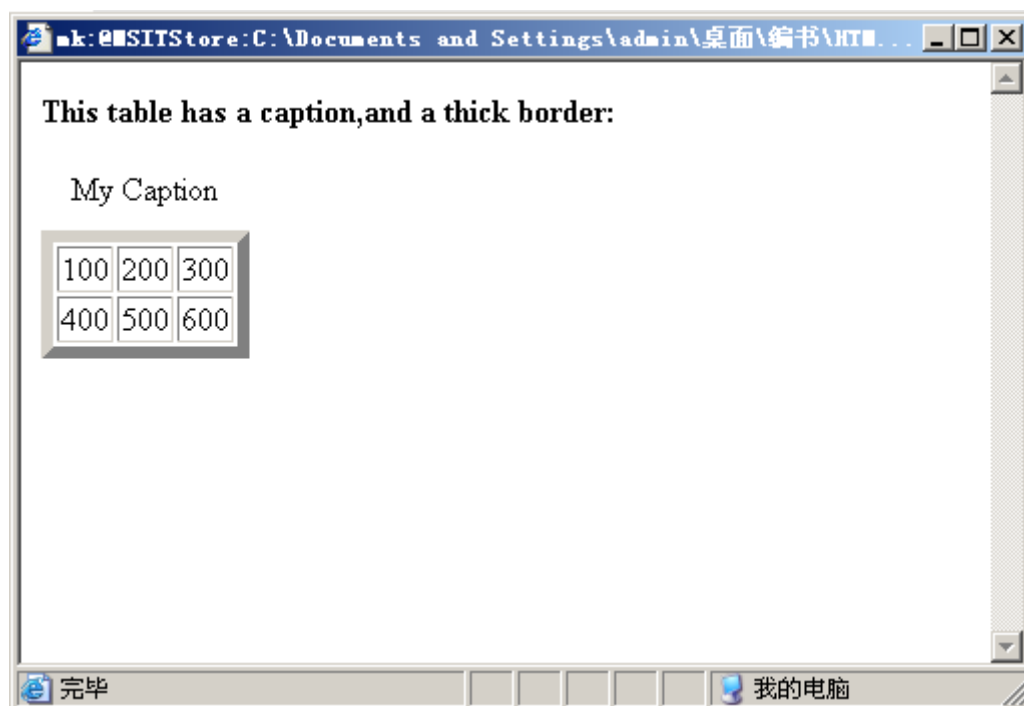


```
<td>Bill Gates</td>
<td>555 77 854</td>
<td>555 77 855</td>
</tr>
</table>
<h4>Vertical headers:</h4>
<table border="1">
<tr>
<th>First Name:</th>
<td>Bill Gates</td>
</tr>
<tr>
<th>Telephone:</th>
<td>555 77 854</td>
</tr>
<tr>
<th>Telephone:</th>
<td>555 77 855</td>
</tr>
</table>
</body>
</html>
```



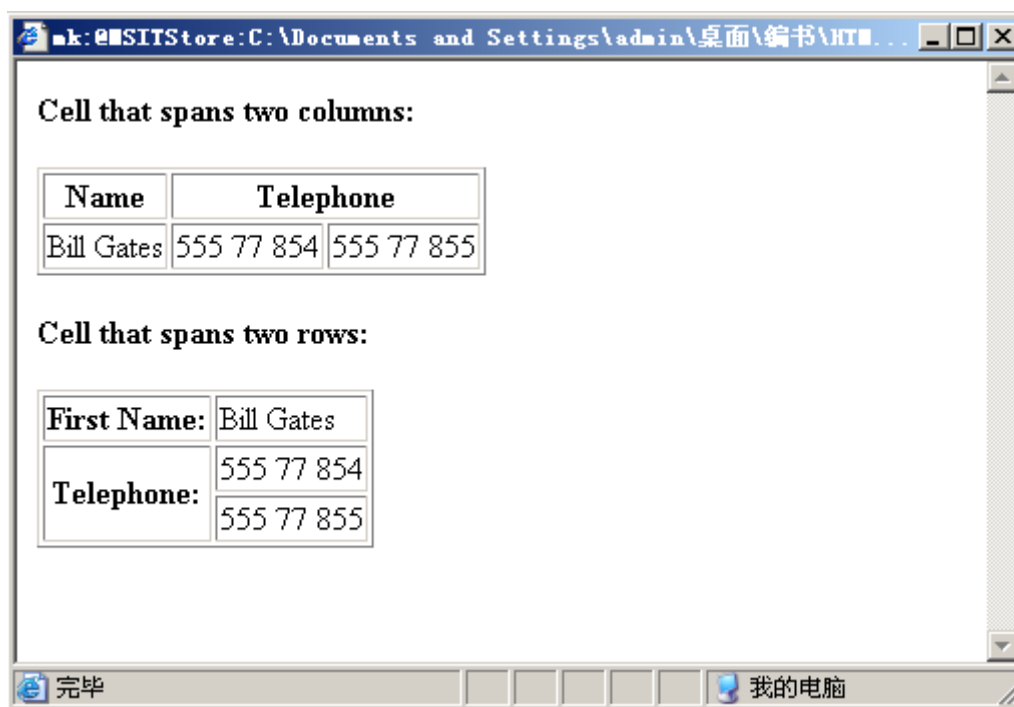
有标题的表格:

```
<html>
<body>
<h4>
This table has a caption,and a thick border:
</h4>
<table border="6">
<caption>My Caption</caption>
<tr>
    <td>100</td>
    <td>200</td>
    <td>300</td>
</tr>
<tr>
    <td>400</td>
    <td>500</td>
    <td>600</td>
</tr>
</table>
</body>
</html>
```



单元格跨行（列）的表格:

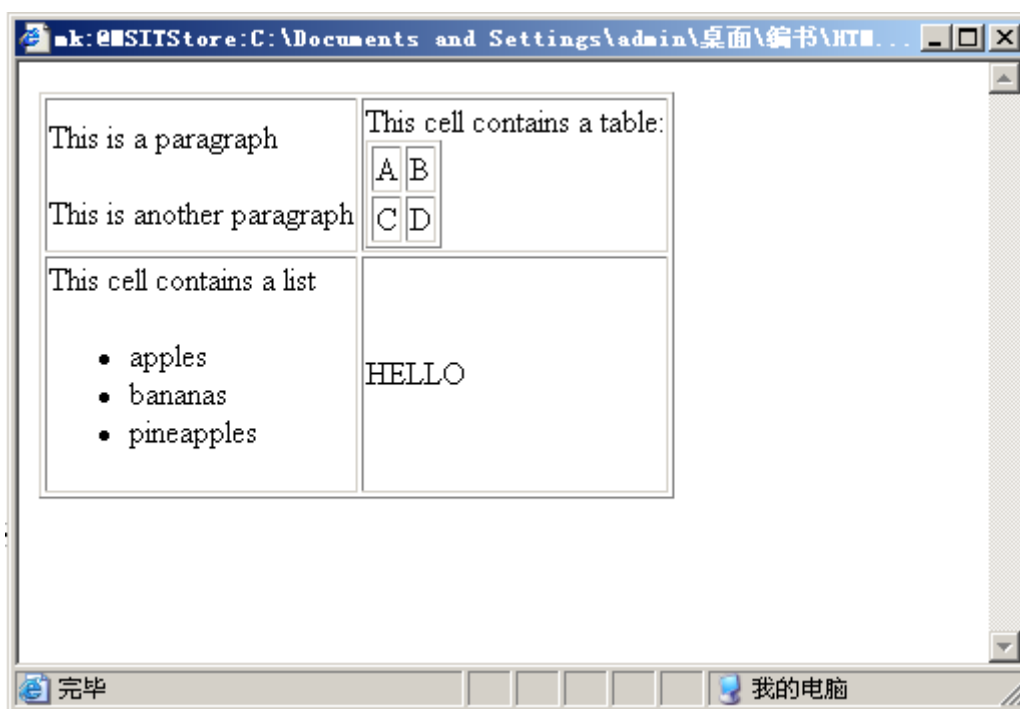
```
<html>
<body>
<h4>Cell that spans two columns:</h4>
<table border="1">
<tr>
    <th>Name</th>
    <th colspan="2">Telephone</th>
</tr>
<tr>
    <td>Bill Gates</td>
    <td>555 77 854</td>
    <td>555 77 855</td>
</tr>
</table>
<h4>Cell that spans two rows:</h4>
<table border="1">
<tr>
    <th>First Name:</th>
    <td>Bill Gates</td>
</tr>
<tr>
    <th rowspan="2">Telephone:</th>
    <td>555 77 854</td>
</tr>
<tr>
    <td>555 77 855</td>
</tr>
</table>
</body>
</html>
```



表格内的其他标签:

```
<html>
<body>
<table border="1">
<tr>
  <td>
    <p>This is a paragraph</p>
    <p>This is another paragraph</p>
  </td>
  <td>This cell contains a table:
    <table border="1">
      <tr>
        <td>A</td>
        <td>B</td>
      </tr>
      <tr>
        <td>C</td>
        <td>D</td>
      </tr>
    </table>
  </td>
</tr>
<tr>
  <td>This cell contains a list
    <ul>
```

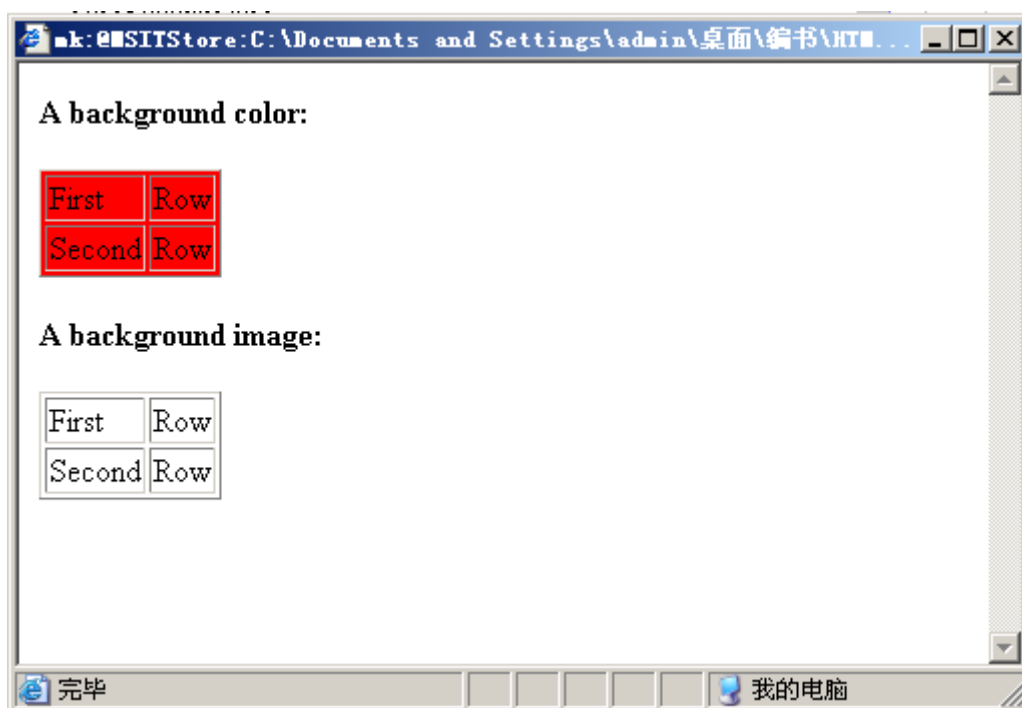
```
<li>apples</li>
<li>bananas</li>
<li>pineapples</li>
</ul>
</td>
<td>HELLO</td>
</tr>
</table>
</body>
</html>
```



给表格增加背景色或者背景图像:

```
<html>
<body>
<h4>A background color:</h4>
<table border="1" bgcolor="red">
<tr>
<td>First</td>
<td>Row</td>
</tr>
<tr>
<td>Second</td>
<td>Row</td>
</tr>
</table>
```

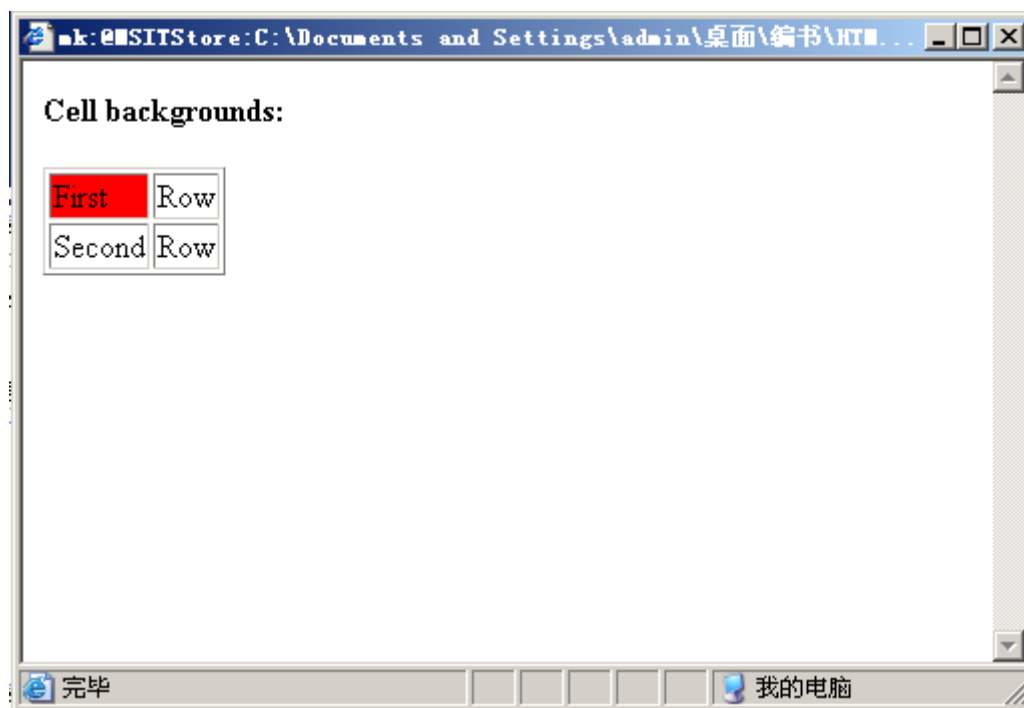
```
<h4>A background image:</h4>
<table border="1" background="/images/bgdesert.jpg">
<tr>
    <td>First</td>
    <td>Row</td>
</tr>
<tr>
    <td>Second</td>
    <td>Row</td>
</tr>
</table>
</body>
</html>
```



这个例子说明了如何给表格增加背景。

```
<html>
<body>
<h4>Cell backgrounds:</h4>
<table border="1">
<tr>
    <td bgcolor="red">First</td>
    <td>Row</td>
</tr>
<tr>
    <td background="/images/bgdesert.jpg">Second</td>
```

```
<td>Row</td>
</tr>
</table>
</body>
</html>
```

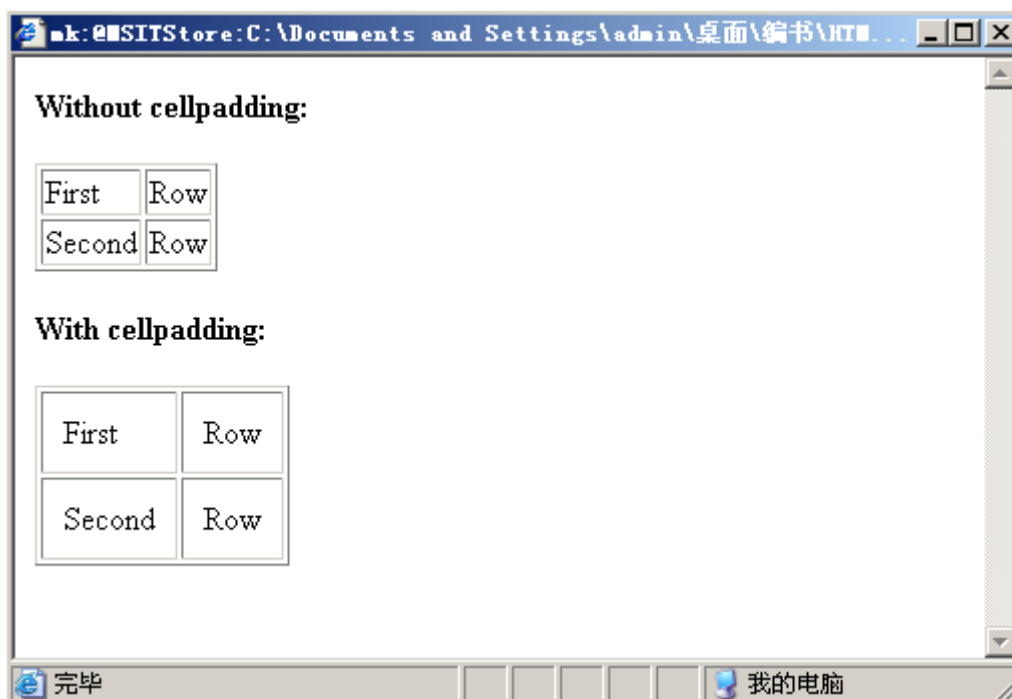


这个例子说明了如何给一个或多个单元格增加背景。

cellpadding 属性:

```
<html>
<body>
<h4>Without cellpadding:</h4>
<table border="1">
<tr>
<td>First</td>
<td>Row</td>
</tr>
<tr>
<td>Second</td>
<td>Row</td>
</tr>
</table>
<h4>With cellpadding:</h4>
<table border="1" cellpadding="10">
<tr>
```

```
<td>First</td>
<td>Row</td>
</tr>
<tr>
<td>Second</td>
<td>Row</td>
</tr>
</table>
</body>
</html>
```



这个例子说明了如何使用 cellpadding 属性在表格内容和边框之间留出更多空白。

cellspacing 属性:

```
<html>
<body>
<h4>Without cellspacing:</h4>
<table border="1">
<tr>
<td>First</td>
<td>Row</td>
</tr>
<tr>
<td>Second</td>
<td>Row</td>
```



```
</tr>
</table>
<h4>With cellspacing:</h4>
<table border="1" cellspacing="10">
<tr>
    <td>First</td>
    <td>Row</td>
</tr>
<tr>
    <td>Second</td>
    <td>Row</td>
</tr>
</table>
</body>
</html>
```

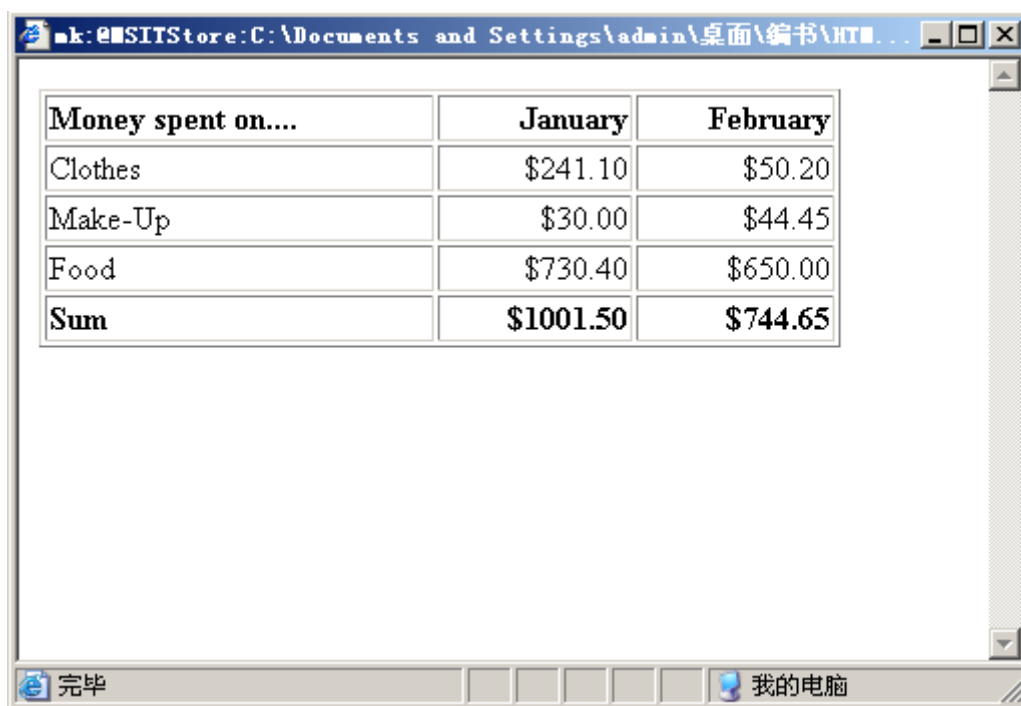


这个例子说明了如何使用 `cellspacing` 属性来增加单元格间距。

给单元格内容设置对齐方式:

```
<html>
<body>
<table width="400" border="1">
<tr>
    <th align="left">Money spent on....</th>
    <th align="right">January</th>
```

```
<th align="right">February</th>
</tr>
<tr>
<td align="left">Clothes</td>
<td align="right">$241.10</td>
<td align="right">$50.20</td>
</tr>
<tr>
<td align="left">Make-Up</td>
<td align="right">$30.00</td>
<td align="right">$44.45</td>
</tr>
<tr>
<td align="left">Food</td>
<td align="right">$730.40</td>
<td align="right">$650.00</td>
</tr>
<tr>
<th align="left">Sum</th>
<th align="right">$1001.50</th>
<th align="right">$744.65</th>
</tr>
</table>
</body>
</html>
```



The screenshot shows a web browser window with the title bar 'mk: @SITStore: C:\Documents and Settings\admin\桌面\编书\HTM...'. The browser displays the rendered HTML table. The table has three columns: 'Money spent on....', 'January', and 'February'. The rows are: 'Clothes' (\$241.10, \$50.20), 'Make-Up' (\$30.00, \$44.45), 'Food' (\$730.40, \$650.00), and 'Sum' (\$1001.50, \$744.65). The 'Sum' row is bolded. The browser's status bar at the bottom shows '完毕' and '我的电脑'.

Money spent on....	January	February
Clothes	\$241.10	\$50.20
Make-Up	\$30.00	\$44.45
Food	\$730.40	\$650.00
Sum	\$1001.50	\$744.65

这个例子说明了如何使用“align”属性来设置单元格的对齐方式，让表格好看一些。

第四章 HTML表单页面的运用

本章目标: 掌握表单基本结构<form>
掌握各种表单元素
能理解 post 和 get 两种提交方式的区别
本章重点: 掌握各种表单元素
本章难点: post 和 get 两种提交方式的区别

一、HTML表单

表单:

表单是一个能够包含表单元素的区域。

表单元素是能够让用户在表单中输入信息的元素(比如文本框, 密码框, 下拉菜单, 单选框, 复选框等等)。

表单是用<form>元素定义的:

```
<form>
<input>
<input>
</form>
```

Input:

最常用的表单标签是<input>标签。Input 的类型用 type 属性指定。最常用的 input 类型解释如下:

文本框: 在表单中, 文本框用来让用户输入字母、数字等等。

```
<form>
First name:
<input type="text" name="firstname">
<br>
Last name:
<input type="text" name="lastname">
</form>
```

在浏览器中显示如下:

First name:
Last name:

单选按钮: 当需要用户从有限个选项中选择一个时, 使用单选按钮。

```
<form>  
<input type="radio" name="sex" value="male">Male  
<br>  
<input type="radio" name="sex" value="female">Female  
</form>
```

在浏览器中显示如下:

☐ Male
☐ Female

注意, 各选项中只能选取一个。

复选框: 当需要用户从有限个选项中选择一个或多个时, 使用复选框。

```
<form>  
<input type="checkbox" name="bike">  
I have a bike  
<br>  
<input type="checkbox" name="car">  
I have a car  
</form>
```

在浏览器中显示如下:

☐ I have a bike
☐ I have a car

表单的 action 属性和提交按钮: 当用户点击提交按钮的时候, 表单的内容会被提交到其他文件。表单的 action 属性定义了所要提交到的目的文件, 该目的文件收到信息后通常进行相关的处理。

```
<form name="input" action="html_form_action.asp" method="get">  
Username:  
<input type="text" name="user">  
<input type="submit" value="Submit">  
</form>
```

在浏览器中显示如下:

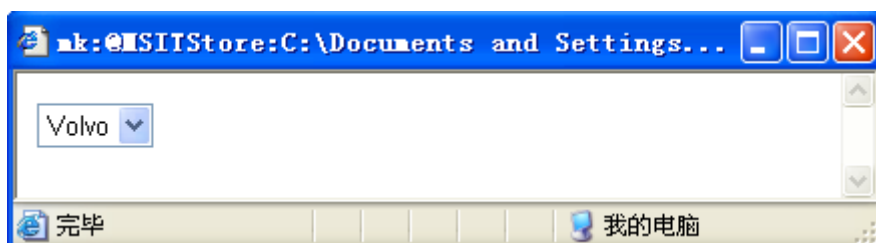
Username:

如果在上面这个文本框中输入一些字符,按下提交按钮以后,输入的字符将被提交到页面“action.asp”。

更多示例:

简单的下拉列表:

```
<html>
<body>
  <form>
    <select name="cars">
      <option value="volvo">Volvo
      <option value="saab">Saab
      <option value="fiat">Fiat
      <option value="audi">Audi
    </select>
  </form>
</body>
</html>
```

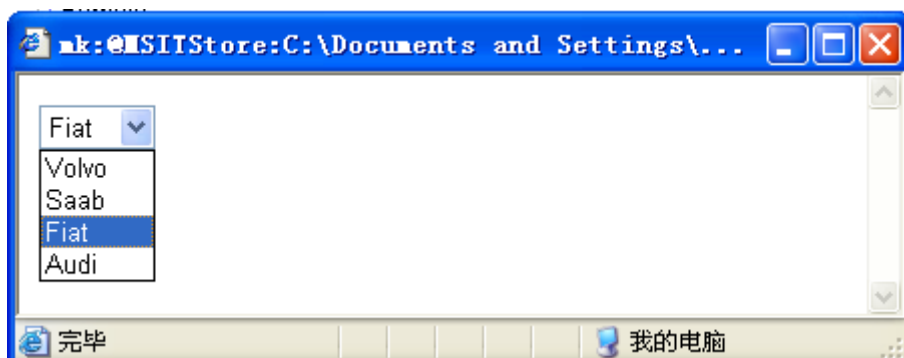


这个例子说明了在 HTML 页面如何创建下拉列表。下拉列表是可以选择的列表。

预选的下拉列表:

```
<html>
<body>
  <form>
    <select name="cars">
      <option value="volvo">Volvo
      <option value="saab">Saab
      <option value="fiat" selected>Fiat
      <option value="audi">Audi
    </select>
  </form>
</body>
```

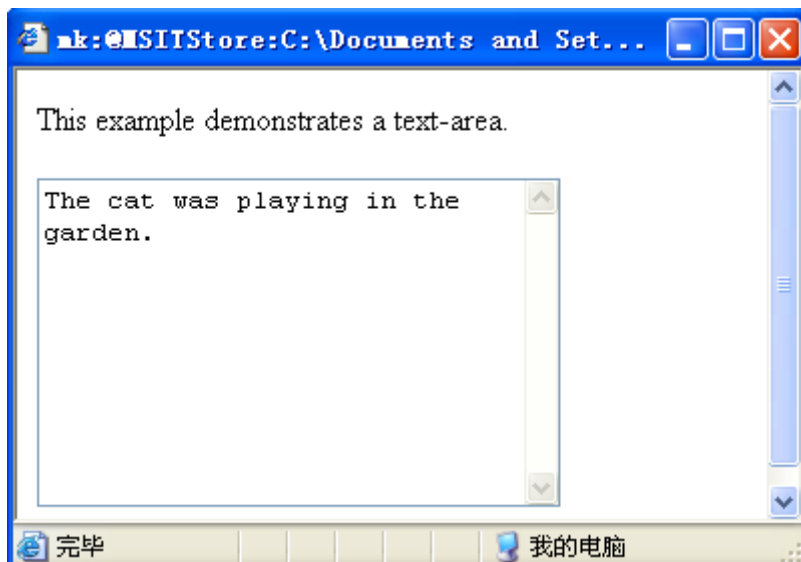
```
</html>
```



这个例子说明了如何创建一个含有预先选定元素的下拉列表。

文本域:

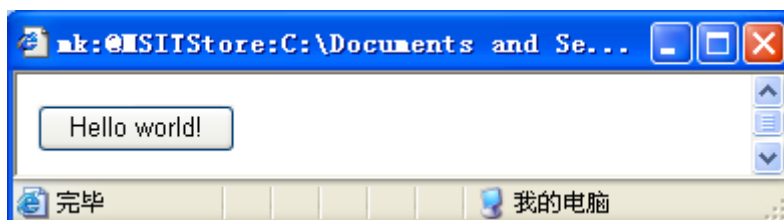
```
<html>
  <body>
    <p>
      This example demonstrates a text-area.
    </p>
    <textarea rows="10" cols="30">
      The cat was playing in the garden.
    </textarea>
  </body>
</html>
```



这个例子说明了如何创建文本域（多行文本），用户可以在其中输入文本。在文本域中，字符个数不受限制。

创建按钮:

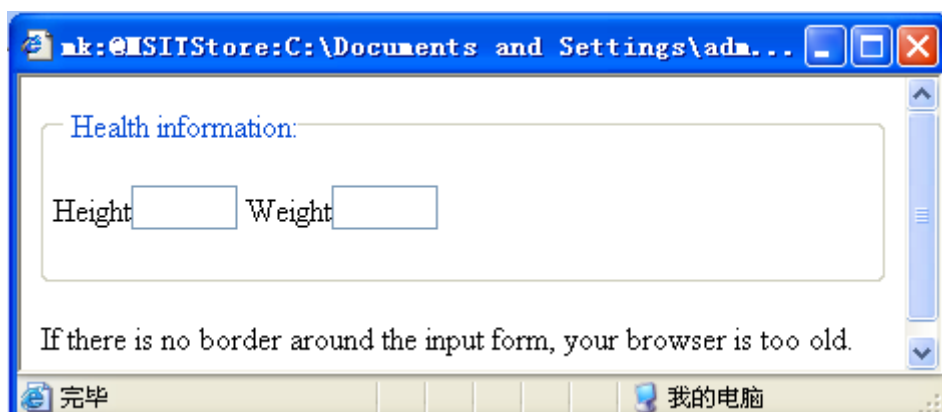
```
<html>
  <body>
    <form>
      <input type="button" value="Hello world!">
    </form>
  </body>
</html>
```



这个例子说明了如何创建按钮。按钮上的文字可以自己定义。

数据周围的标题边框:

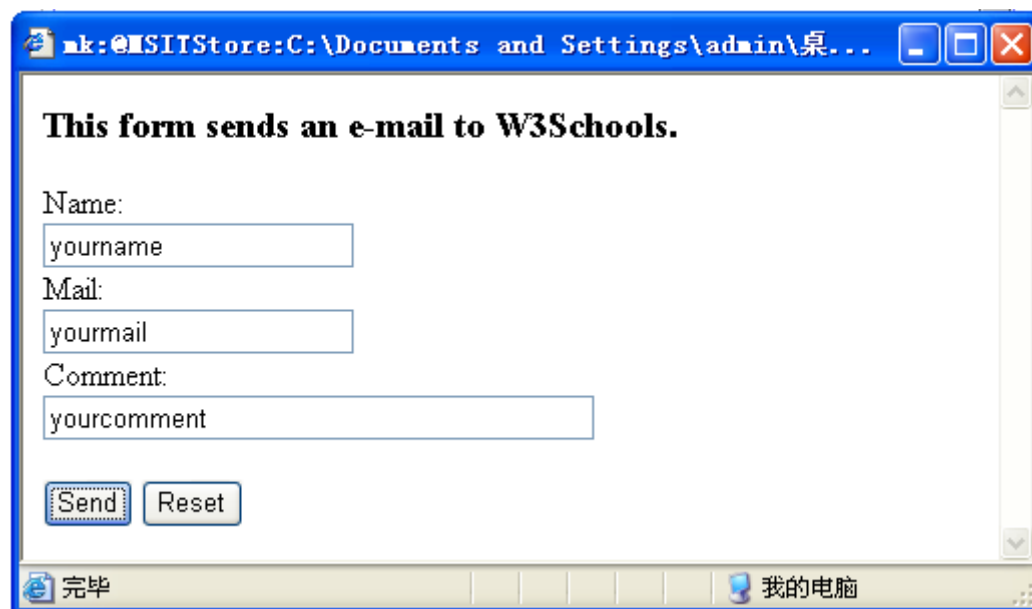
```
<html>
  <body>
    <fieldset>
      <legend>
        Health information:
      </legend>
      <form>
        Height<input type="text" size="3">
        Weight<input type="text" size="3">
      </form>
    </fieldset>
    <p>
      If there is no border around the input form, your browser is too old.
    </p>
  </body>
</html>
```



这个例子说明了如何在数据周围画带有标题的边框。

从表单发送电子邮件:

```
<html>
<body>
  <form action="MAILTO:someone@w3schools.com" method="post"
  enctype="text/plain">
    <h3>This form sends an e-mail to W3Schools.</h3>
    Name:<br>
    <input type="text" name="name" value="yourname" size="20">
    <br>
    Mail:<br>
    <input type="text" name="mail" value="yourmail" size="20">
    <br>
    Comment:<br>
    <input type="text" name="comment" value="yourcomment" size="40">
    <br><br>
    <input type="submit" value="Send">
    <input type="reset" value="Reset">
  </form>
</body>
</html>
```



这个例子说明了如何从一个表单发送电子邮件

第五章 使用样式表美化页面 1

本章目标: 掌握在网页中使用 CSS 的方法

熟悉 CSS 的不同选择器的使用方法

熟悉字体属性: font-family, font-size, font-style,
font-weight

熟悉文本属性: text-align, text-indent, text-decoration,
text-transform, vertical-align, word-spacing,
letter-spacing

本章重点: 掌握在网页中使用 CSS 的方法

本章难点: 文本属性, 字体属性

一、CSS的工作原理

在这一节, 你将学习如何制作自己的第一个样式表。你将了解基本的 CSS 模型, 以及在 HTML 文档里使用 CSS 所必需的代码。

级联样式表 (CSS) 里用到的许多 CSS 属性都与 HTML 属性相似, 所以, 假如你熟悉采用 HTML 进行布局的话, 那么这里的许多代码你都不会感到陌生。我们先来看一个具体的例子。

基本的 CSS 语法:

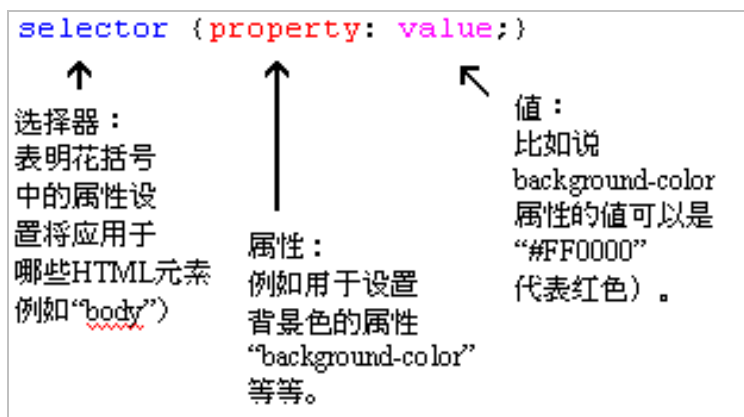
比方说, 我们要用红色作为网页的背景色:
用 HTML 的话, 我们可以这样:

```
<body bgcolor="#FF0000">
```

用 CSS 的话, 我们可以这样获得同样的效果:

```
body {background-color: #FF0000;}
```

你会注意到, HTML 和 CSS 的代码颇有几分相似。上例也向你展示了基本的 CSS 模型:



但是把 CSS 代码放在哪里呢？这正是我们下面要讲的

为一个 HTML 文档应用 CSS：

为 HTML 文档应用 CSS，有三种方法可供选择。下面对这三种方法进行了概括。我们建议你第三种方法（即外部样式表）予以关注。

方法 1：行内样式表（style 属性）

为 HTML 应用 CSS 的一种方法是使用 HTML 属性 style。我们在上例的基础之上，通过行内样式表将页面背景设为红色：

```
<html>
  <head>
    <title>例子</title>
  </head>
  <body style="background-color: #FF0000;">
    <p>这个页面是红色的</p>
  </body>
</html>
```

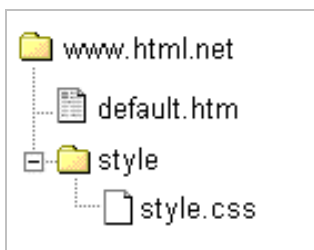
方法 2：内部样式表（style 元素）

为 HTML 应用 CSS 的另一种方法是采用 HTML 元素 style。比如像这样：

```
<html>
  <head>
    <title>例子</title>
    <style type="text/css">
      body {background-color: #FF0000;}
    </style>
  </head>
  <body>
    <p>这个页面是红色的</p>
  </body>
</html>
```

方法 3: 外部样式表 (引用一个样式表文件)

我们推荐采用这种引用外部样式表的方法。在本教程之后的例子中, 我们将全部采用该方法。外部样式表就是一个扩展名为 **css** 的文本文件。跟其他文件一样, 你可以把样式表文件放在 **Web 服务器** 上或者本地硬盘上。例如, 比方说你的样式表文件名为 **style.css**, 它通常被存放于名为 **style** 的目录中。就像下面这样:



现在的问题是: 如何在一个 **HTML** 文档里引用一个外部样式表文件 (**style.css**) 呢? 答案是: 在 **HTML** 文档里创建一个指向外部样式表文件的链接 (**link**) 即可, 就像下面这样:

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

注意要在 href 属性里给出样式表文件的地址

这行代码必须被插入 **HTML 代码的头部 (header)**, 即放在标签 **<head>** 和标签 **</head>** 之间。就像这样:

```
<html>
  <head>
    <title>我的文档</title>
    <link rel="stylesheet" type="text/css" href="style/style.css" />
  </head>
  <body>
    ...
```

这个链接告诉浏览器: 在显示该 **HTML** 文件时, 应使用给出的 **CSS** 文件进行布局。

这种方法的优越之处在于: 多个 **HTML** 文档可以同时引用一个样式表。换句话说, 可以用一个 **CSS** 文件来控制多个 **HTML** 文档的布局。



这一方法可以令你省去许多工作。例如, 假设你要修改某网站的所有网页(比方说有 100 个网页)的背景颜色, 采用外部样式表可以避免你手工一一修改这 100 个 HTML 文档的工作。采用外部样式表, 这样的修改只需几秒钟即可搞定——修改外部样式表文件里的代码即可。

让我们来实践刚刚所学到的知识, 自己试试看:

打开记事本(或其他文本编辑器), 创建两个文件——一个 HTML 文件, 一个 CSS 文件——它们的内容如下:

default.htm:

```
<html>
  <head>
    <title>我的文档</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
  </head>
  <body>
    <h1>我的第一个样式表</h1>
  </body>
</html>
```

style.css:

```
body {
  background-color: #FF0000;
}
```

然后, 把这两个文件放在同一目录下。记得在保存文件时使用正确的扩展名(分别为“htm”和“css”)。用浏览器打开 **default.htm**, 你所看到的页面应该具有红色背景。恭喜! 你已经完成了自己的第一个样式表!

二、 元素的分类与标识 (class和id)

有时, 你希望对特定元素或者特定一组元素应用特殊的样式。在这一节, 我们将深入学习如何利用 class 和 id 来为所选元素指定属性。

如何实现为网站上许多标题中的某一个单独应用颜色? 如何实现把网站上的链接分为不同的类, 并对各类链接分别应用不同的样式? 这只是本节将解决的诸多问题中的最具代表性的两个问题。

用 class 对元素进行分类:

比方说, 我们有两个由链接组成的列表, 它们分别是用于制造白葡萄酒和红葡萄酒的葡萄。其 HTML 代码如下:

```
<p>制造白葡萄酒的葡萄: </p>
<ul>
<li><a href="ri.htm">雷司令 (Riesling) </a></li>
<li><a href="ch.htm">夏敦埃 (Chardonnay) </a></li>
<li><a href="pb.htm">白比诺 (Pinot Blanc) </a></li>
</ul>

<p>制造红葡萄酒的葡萄: </p>
<ul>
<li><a href="cs.htm">卡百内索维农 (Cabernet Sauvignon) </a></li>
<li><a href="me.htm">墨尔乐 (Merlot) </a></li>
<li><a href="pn.htm">黑比诺 (Pinot Noir) </a></li>
</ul>
```



现在,我们希望白葡萄酒的链接全部显示为黄色,红葡萄酒的链接全部显示为红色,其余的链接显示为缺省的兰色。为了实现这一要求,我们将链接分为两类。对链接的分类是通过为链接设置 HTML 属性 class 实现的。

参加如下代码:

```
<p>制造白葡萄酒的葡萄: </p>
<ul>
<li><a href="ri.htm" class="whitevine">雷司令 (Riesling) </a></li>
<li><a href="ch.htm" class="whitevine">夏敦埃 (Chardonnay) </a></li>
<li><a href="pb.htm" class="whitevine">白比诺 (Pinot Blanc) </a></li>
</ul>

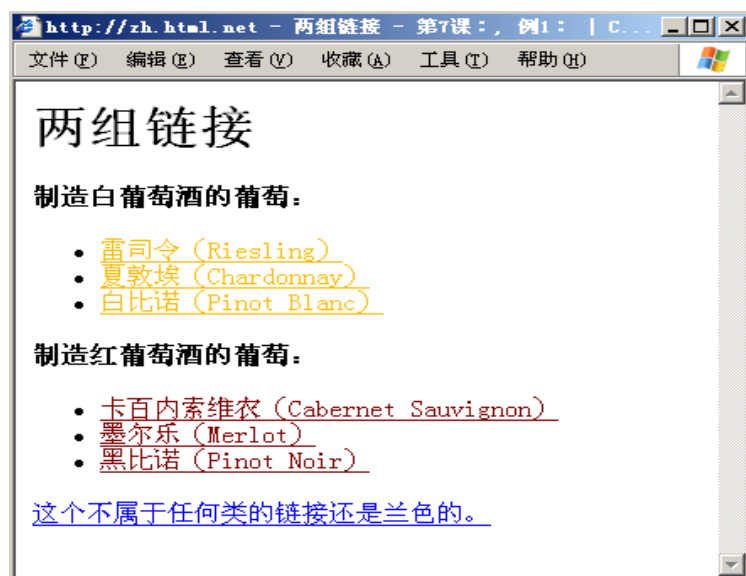
<p>制造红葡萄酒的葡萄: </p>
<ul>
<li><a href="cs.htm" class="redwine">卡百内索维农 (Cabernet Sauvignon) </a></li>
<li><a href="me.htm" class="redwine">墨尔乐 (Merlot) </a></li>
<li><a href="pn.htm" class="redwine">黑比诺 (Pinot Noir) </a></li>
</ul>
```

然后,我们就可以为白葡萄酒和红葡萄酒的链接分别应用不同的风格了。

```
a {
    color: blue;
}

a.whitevine {
    color: #FFBB00;
}

a.redwine {
    color: #800000;
}
```



如上例所示,你可以通过在样式表里利用**.classname** 来为属于某一类的元素定义 CSS 属性。

利用 id 标识元素:

除了可以对元素进行分类以外,你还可以标识单个元素。这是通过 HTML 属性 id 实现的。HTML 属性 id 的特别之处在于,在同一 HTML 文档中不能有两个具有相同 id 值的元素。文档中的每个 id 值都必须是唯一的。在其他情况下,你应该使用 class 属性。下面,我们来看一个使用 id 属性的例子:

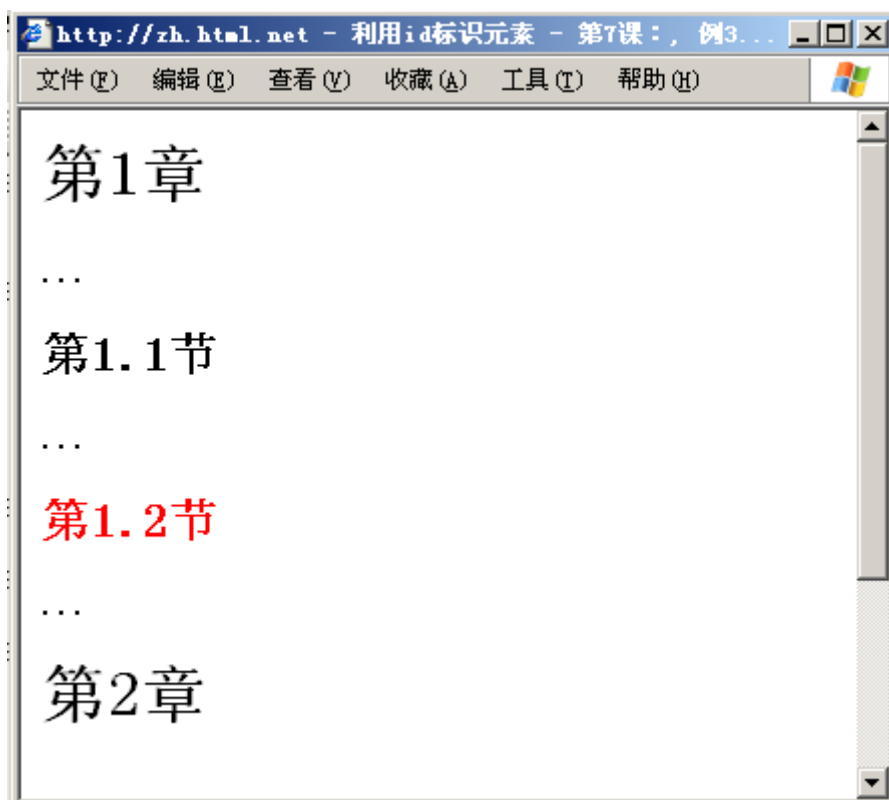
```
<h1>第1章</h1>
...
<h2>第1.1节</h2>
...
<h2>第1.2节</h2>
...
<h1>第2章</h1>
...
<h2>第2.1节</h2>
...
<h2>第2.1.1小节</h2>
...
```

上例是一个文章的各章节的标题。我们自然可以为其中每一章节指定一个 id (如下):

```
<h1 id='c1'>第1章</h1>
...
<h2 id='c1-1'>第1.1节</h2>
...
<h2 id='c1-2'>第1.2节</h2>
...
<h1 id='c2'>第2章</h1>
...
<h2 id='c2-1'>第2.1节</h2>
...
<h3 id='c2-1-2'>第2.1.1节</h3>
...
```

假如我们要求第 1.2 节显示为红色,那么 CSS 可以这样写:

```
#c1-2 {
    color: red;
}
```

如上例所示, 你可以在样式表里通过#id 为特定元素定义 CSS 属性

三、 字体属性

这一节, 你将学习字体以及如何用 CSS 来设置字体。我们还会考虑如何解决“网站所选的字体仅当访问者的 PC 上安装有该字体时才会被显示”这一难题。本节将对下列 CSS 属性进行讲解:

- [font-family](#)
- [font-style](#)
- [font-variant](#)
- [font-weight](#)
- [font-size](#)
- [font](#)

字体族[font-family]:

CSS 属性 font-family 的作用是设置一组按优先级排序的字体列表, 如果该列表中的第一个字体未在访问者计算机上安装, 那么就尝试列表中的下一个字体, 依此类推, 直到列表中的某个字体是已安装的。

有两种类型的名称可用于分类字体: 字体族名称 (family-name) 和族类名称 (generic family)。下面来解释这两个术语。

字体族名称 (family-name):

字体族名称 (就是我们通常所说的“字体”) 的例子包括 “Arial”、“Times New Roman”、“宋体”、“黑体”等等。

族类 (generic family):

一个族类是一组具有统一外观的字体族。sans-serif 就是一例, 它代表一组没有“脚”的字体。

下面我们通过三个族类的例子来进行解释:

Times New Roman Garamond Georgia	这三个字体族属于 serif 族类。 它们的共同特点是, 笔画两端有“脚”。
Trebuchet Arial Verdana	这三个字体族属于 sans-serif 族类。 它们的共同特点是, 笔画两端没有“脚”。
Courier Courier New Andale Mono	这三个字体族属于 monospace 族类。 它们的共同特点是, 所有字符的宽度都一样。

你在给出字体列表时, 自然应把首选字体放在前面、把候选字体放在后面。建议你在列表的最后给出一个族类 (generic family), 这样, 当没有一个指定字体可用时, 页面至少可以采用一个相同族类的字体来显示。

下面是一个按优先级排列的字体列表的例子:

```
<body>  
  <h1>大标题是 Arial 字体</h1>  
  <h2>而次标题是 Times New Roman 字体</h2>  
</body>
```



h1 标题将采用 Arial 字体显示。如果访问者的计算机未安装 Arial, 那么就使用 Verdana 字体。假如 Verdana 字体也没安装的话, 那么将采用一个属于 **sans-serif** 族类的字体来显

示这个 h1 标题。注意我们为“Times New Roman”采用的写法: 因为其中包含空格, 所以我们用引号将它括起来。

字体样式[font-style]:

CSS 属性 font-style 定义所选字体的显示样式: **normal**(正常) **italic**(斜体)或 **oblique**(倾斜)。在下例中, 所有 h2 标题都将显示为斜体。

```
h1 {font-family: arial, verdana, sans-serif;}
h2 {font-family: "Times New Roman", serif; font-style: italic;}
```



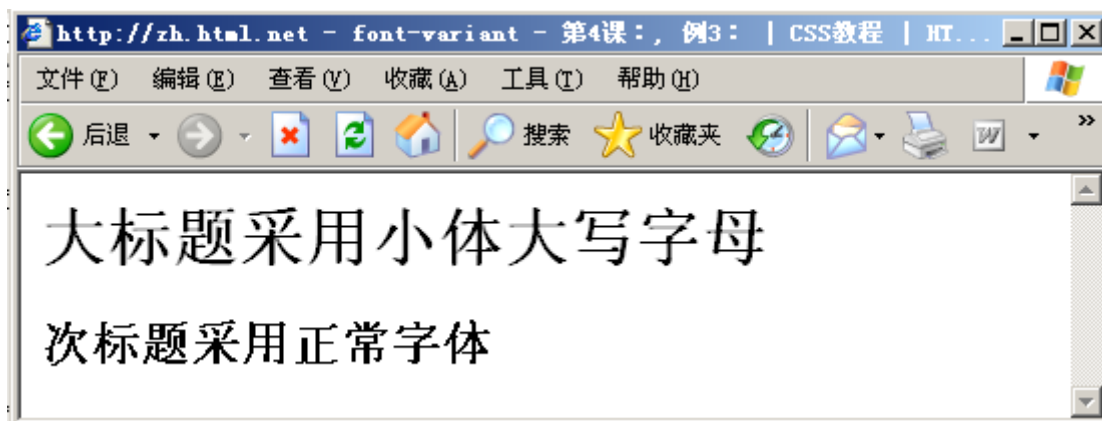
字体变化[font-variant]:

CSS 属性 font-variant 的值可以是: **normal** (正常) 或 **small-caps** (小体大写字母)。**small-caps** 字体是一种以小尺寸显示的大写字母来代替小写字母的字体。不太明白? 我们来看几个例子:

Sans Book SC	Sans Bold SC	Serif Book SC	Serif Bold SC
ABCABC	ABCABC	ABCABC	ABCABC

如果 font-variant 属性被设置为 **small-caps**, 而没有可用的支持小体大写字母的字体, 那么浏览器多半会将文字显示为正常尺寸 (而不是小尺寸) 的大写字母。

```
h1 {font-variant: small-caps;}
h2 {font-variant: normal;}
```



字体浓淡[font-weight]:

CSS 属性 font-weight 指定字体显示的浓淡程度。其值可以是 **normal** (正常) 或 **bold** (加粗)。有些浏览器甚至支持采用 100 到 900 之间的数字 (以百为单位) 来衡量字体的浓淡。

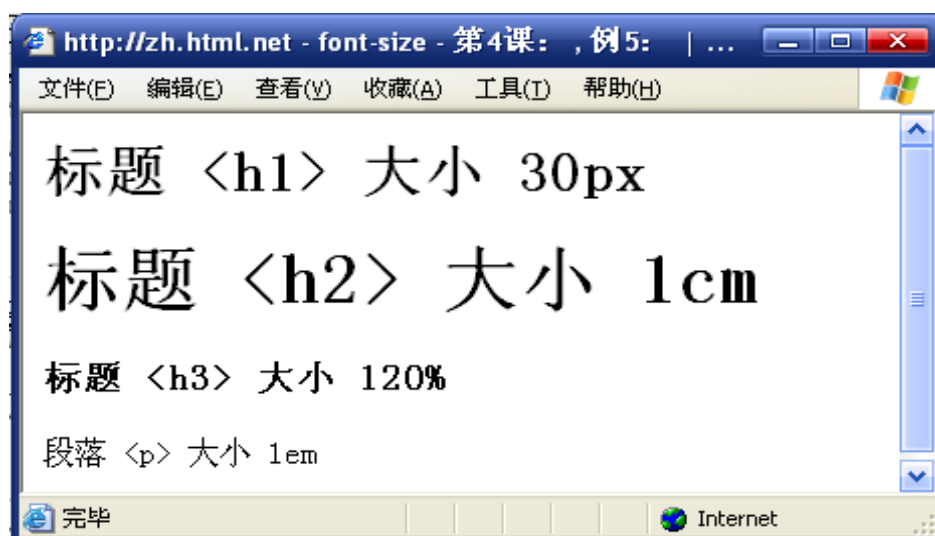
```
p {font-family: arial, verdana, sans-serif;}
td {font-family: arial, verdana, sans-serif; font-weight: bold;}
```

字体大小[font-size]:

字体的大小用 CSS 属性 font-size 来设置。

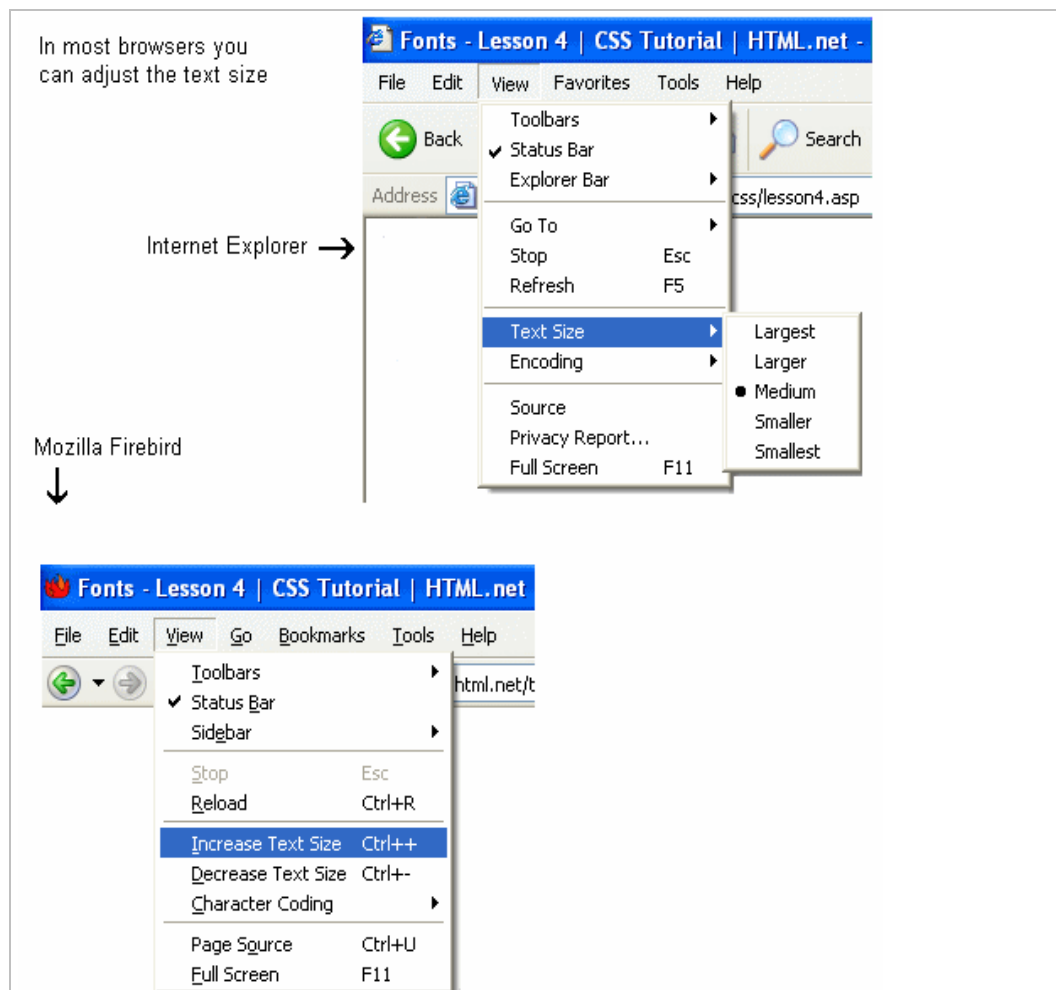
字体大小可通过多种不同单位 (比如像素或百分比等) 来设置。在本教程中, 我们将关注于最常用和最合适的单位。比如:

```
h1 {font-size: 30px;}
h2 {font-size: 12pt;}
h3 {font-size: 120%;}
p {font-size: 1em;}
```



上面四种单位有着本质的区别。‘**px**’和‘**pt**’将字体设置为固定大小，而‘**%**’和‘**em**’允许页面浏览者自行调整字体的显示尺寸。有些页面浏览者可能是残疾者、年长者、视力不佳者，或者他所使用的电脑显示屏显示质量差。为了令你的网站对所有人都具有良好的可用性（accessibility），你应采用像‘**%**’或‘**em**’这种允许用户调节字体显示大小的单位。

下面你能看到我们展示如何在 Mozilla Firefox 和 Internet Explorer 里调整字体大小。自己试试看！这个功能很不错吧？



缩写[font]:

CSS 属性 font 是上述各有关字体的 CSS 属性的缩写用法。

比如说下面四行应用于 p 元素的代码：

```
p {
    font-style: italic;
    font-weight: bold;
    font-size: 30px;
    font-family: arial, sans-serif;
}
```

如果用 font 属性的话，上述四行代码可简化为：

```
p {  
    font: italic bold 30px arial, sans-serif;  
}
```

font 属性的值应按以下次序书写:

font-style|font-variant|font-weight|font-size|font-family

小结:

在这一节, 你学习了有关字体设置的用法。记住: CSS的一个主要优势就是可以在任何时候设置字体, 你花几分钟就可以改变整个网站的字体。CSS节省时间, 而且把事情简化。在下一节, 我们将学习文本 (text)。

四、 文本属性

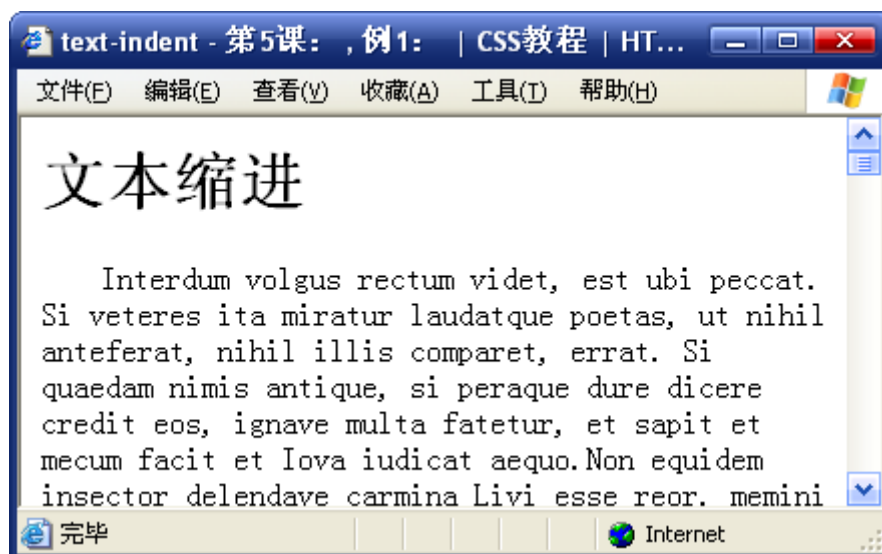
文本的显示格式与式样对于网页设计师来说是一个重要问题。这一节将向你介绍 CSS 在文本布局方面令人激动的特性。本节将对下列 CSS 属性进行讲解:

- [text-indent](#)
- [text-align](#)
- [text-decoration](#)
- [letter-spacing](#)
- [text-transform](#)

文本缩进[text-indent]:

CSS 属性 text-indent 用于为段落设置首行缩进, 以令其具有美观的格式。在下例中, 我们为采用 p 元素的段落应用了 30 像素的首行缩进。

```
p {  
    text-indent: 30px;  
}
```



文本对齐[text-align]:

CSS 属性 text-align 与 HTML 属性 align 的功能相同。该属性的值可以是: **left** (左对齐)、**right** (右对齐) 或者 **center** (居中)。除了上面三种选择以外, 你还可以将该属性的值设为 **justify** (两端对齐), 即伸缩行中的文字以左右靠齐。报刊杂志经常采用这种布局。

在下例中, 标题 (th) 中的文字被设置为右对齐, 而表中数据 (td) 被设置为居中。正常的文本段落被设置为两端对齐。

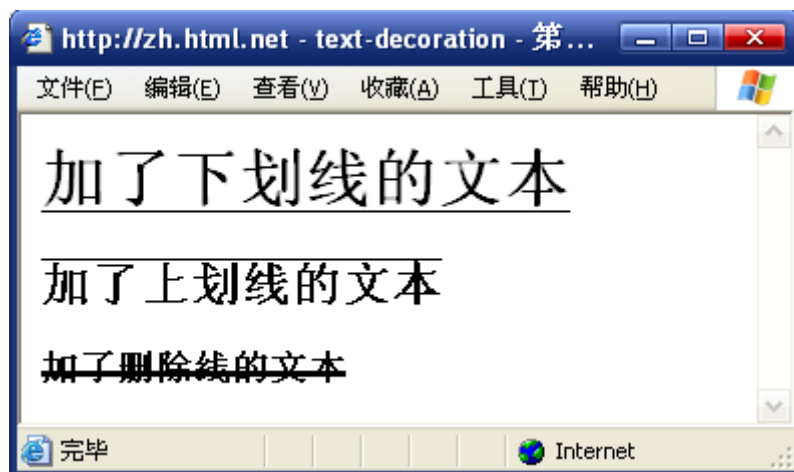
```
th {  
    text-align: right;  
}  
  
td {  
    text-align: center;  
}  
  
p {  
    text-align: justify;  
}
```



文本装饰[text-decoration]:

CSS 属性 text-decoration 令我们可以为文本增添不同的“装饰”或“效果”。例如, 你可以为文本增添下划线、删除线、上划线等等。在接下来的例子中, 我们为 h1 标题增添了下划线, 为 h2 标题增添了上划线, 为 h3 标题增添了删除线。

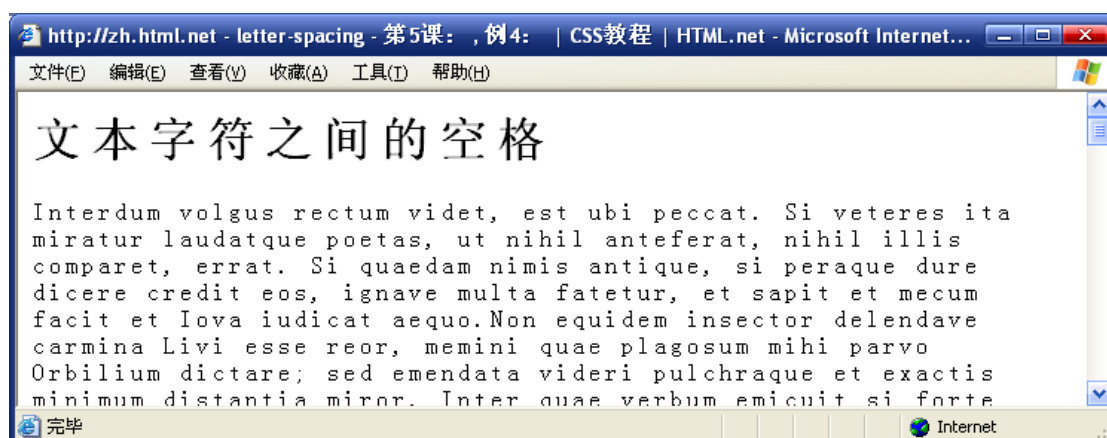
```
h1 {  
    text-decoration: underline;  
}  
  
h2 {  
    text-decoration: overline;  
}  
  
h3 {  
    text-decoration: line-through;  
}
```

字符间距[letter-spacing]:

CSS 属性 letter-spacing 用于设置文本的水平字间距。我们可以把期望的字间距宽度作为这个属性的值。例如, 假如你希望 p 元素里的文本段落的字间距为 3 个像素, 而 h1 标题的字间距为 6 个像素, 代码可以这样写:

```
h1 {  
    letter-spacing: 6px;  
}  
  
p {  
    letter-spacing: 3px;  
}
```



文本转换[text-transform]:

CSS 属性 text-transform 用于控制文本的大小写。无论字母本来的大小写, 你可以通过该属性令它首字母大写(capitalize)、全部大写(uppercase)或者全部小写(lowercase)。比如, 单词“headline”在展现给网页浏览者时, 可以是“HEADLINE”或者“Headline”。text-transform 属性有四个可选值:

Capitalize: 将每个单词的首字母转换为大写。例如: “john doe” 将被转换为 “John

Doe”。

Uppercase: 所有字母都转换为大写。例如: “john doe” 将被转换为 “JOHN DOE”。

Lowercase: 所有字母都转换为小写。例如: “JOHN DOE” 将被转换为 “john doe”。

None: 不作任何转换——文本怎么写的就怎么显示。

来举个例子, 我们将使用一个姓名列表。所有姓名都用 (列表项) 标签来标记。我们希望对姓名采用首字母大写的方式, 而对标题采用全部大写的方式。

查看过该例的 HTML 代码后你会发现, 其实在 HTML 代码里我们写的姓名和标题全部都是小写。

```
h1 {  
    text-transform: uppercase;  
}  
  
li {  
    text-transform: capitalize;  
}
```

```
<body>  
  <h1>这个标题采用大写字母</h1>  
  <ul>  
    <li>peter hanson</li>  
    <li>max larsen</li>  
    <li>joe doe</li>  
    <li>paula jones</li>  
    <li>monica lewinsky</li>  
    <li>donald duck</li>  
  </ul>  
  <p>注意, 我们用 CSS 实现了令所有人名的首字母大写。</p>  
</body>
```



第六章 使用样式表美化页面 2

本章目标: 熟悉显示属性: `display`
熟悉边框属性: `Border, border-style` 等
熟悉定位属性: `top, Width, Height, Left`
本章重点: 如何使用 CSS 样式表进行布局修饰
本章难点: 边框属性

一、显示属性

显示属性允许使用四个值中的一个来定义一个元素:

block : 在元素的前和后都会有换行

inline : 在元素的前和后都不会有换行

list-item : 与 **block** 相同, 但增加了目录项标记

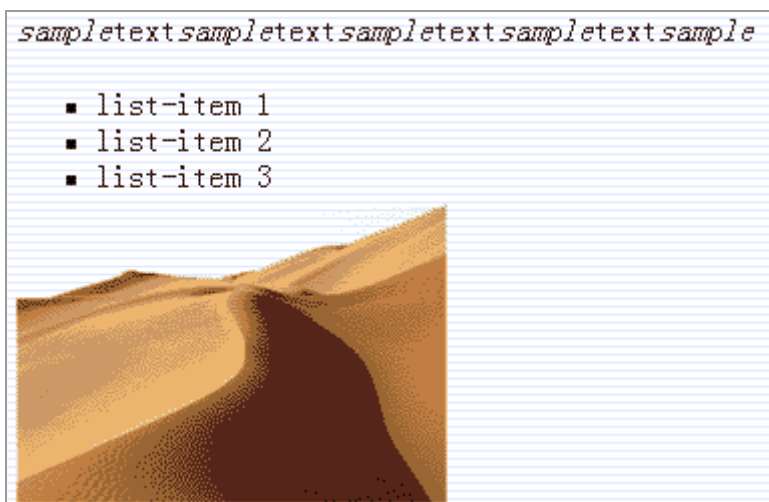
none : 没有显示

下面我们来看一个分级属性的例子, 代码如下所示:

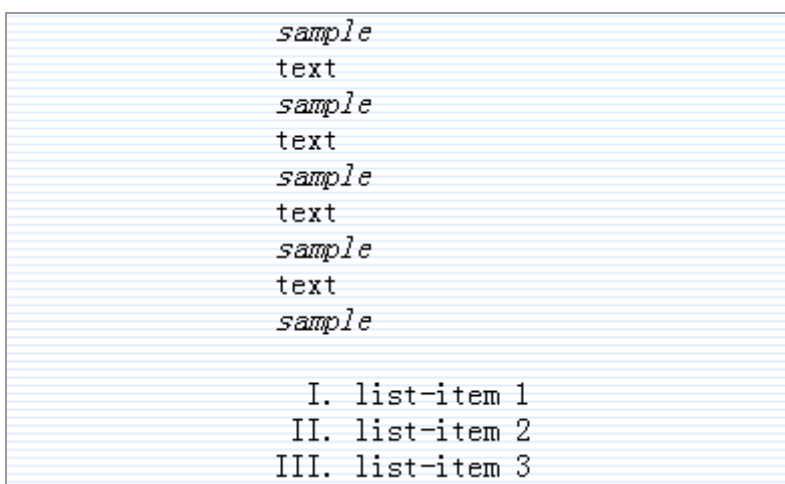
```
p{display: block; white-space: normal} ↓  
em{display: inline} ↓  
li{display: list-item; list-style: square} ↓  
img{display: block} ↗
```

```
<body>  
  <p>  
    <em>sample</em>text<em>sample</em>text<em>sample</em>  
    text<em>sample</em> text<em>sample</em>  
  </p>  
  <ul>  
    <li>list-item 1</li>  
    <li>list-item 2</li>  
    <li>list-item 3</li>  
  </ul>  
  <p><img src= "ss01068. jpg" width= "280" height= "185"  
    alt= "invisible" >  
  </p>  
</body>
```

上段代码的显示效果如下图:



我们看到由于定义了<p>的属性为 Block, 所以文本、列表、图片都在不同的位置上打开, Inline 属性使文本不折行, list-style-type 的属性值为 square 使列表项前的符号为方块; 如果我们在上面的代码中做一些改动, 则将以另一种效果显示, 我们在<head>中把“EM”的 display 属性值改为 block, 使其都在新的位置打开; li 的“list-style”属性值改为“Upper-roman” (大写罗马符号), img 的“display”属性值改为“none” (让图片不显示)。修改后的显示效果如下图:



二、 边框属性

边框 (border) 可以有多种用途, 比如作为装饰元素或者作为划分两物的分界线。在设置边框方面, CSS 为你提供了无尽选择。

- [border-width](#)
- [border-color](#)
- [border-style](#)

边框宽度[border-width]:

边框宽度由 CSS 属性 border-width 定义, 其值可以是 “thin” (薄)、 “medium” (普通) 或 “thick” (厚) 等, 也可以是像素值。如下图所示:



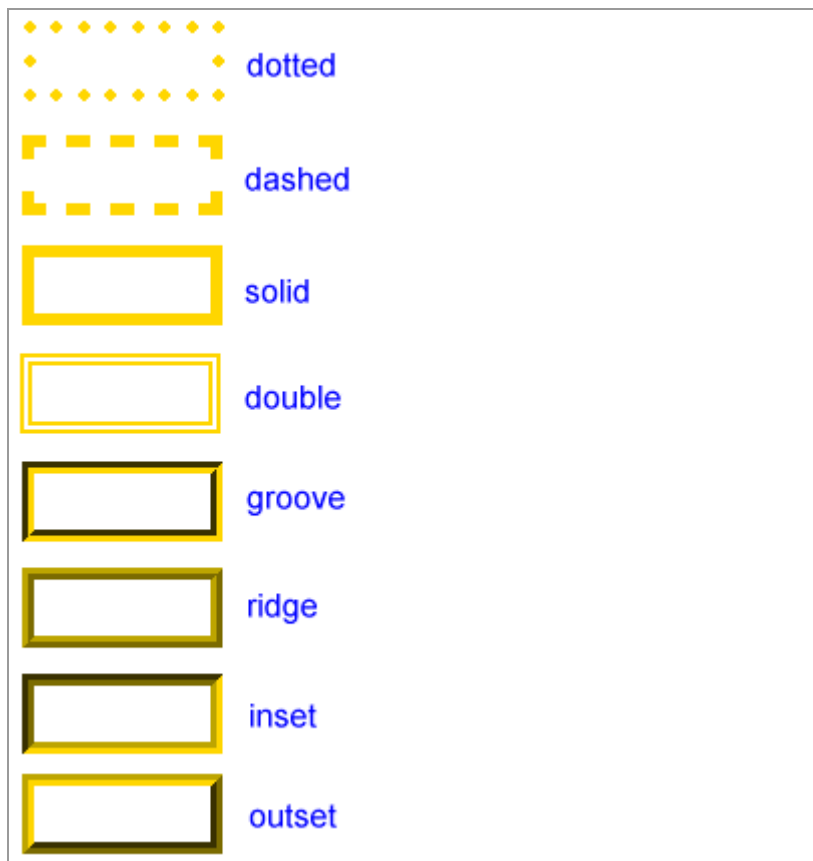
边框颜色[border-color]:



CSS 属性 border-color 用于定义边框的颜色。其值就是正常的颜色值, 例如: “#123456”、 “rgb(123,123,123)”、 “yellow” 等。

边框样式[border-style]:

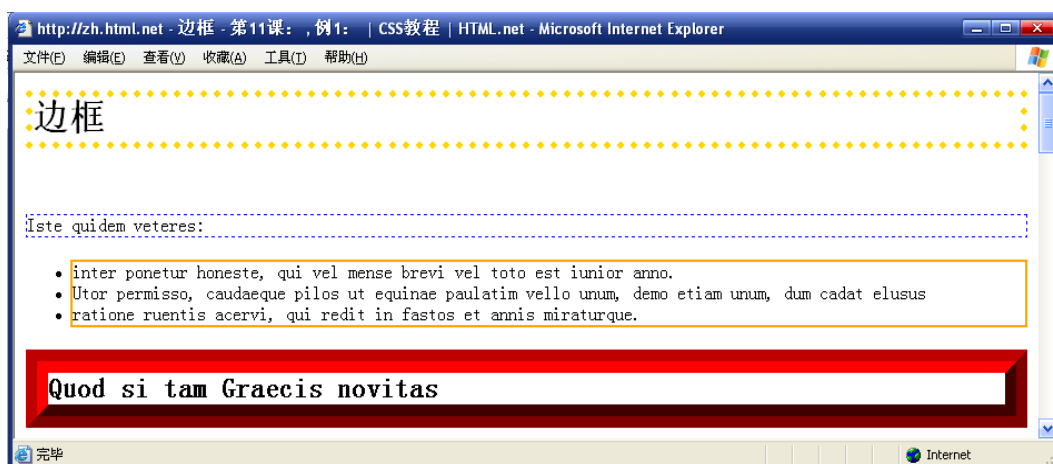
边框样式有多种可供选择。下图显示了 8 种不同样式的边框在 Internet Explorer 5.5 里的实际显示效果。在这个例子里, 我们为这 8 种边框都选择了“金色 (gold)”作为边框颜色、“厚(thick)”作为边框宽度。当然, 这只是个例子, 你可以为边框设置别的颜色和厚度。如果你不想有任何边框, 可以为它取值为 “none” 或者 “hidden”。



一些示例:

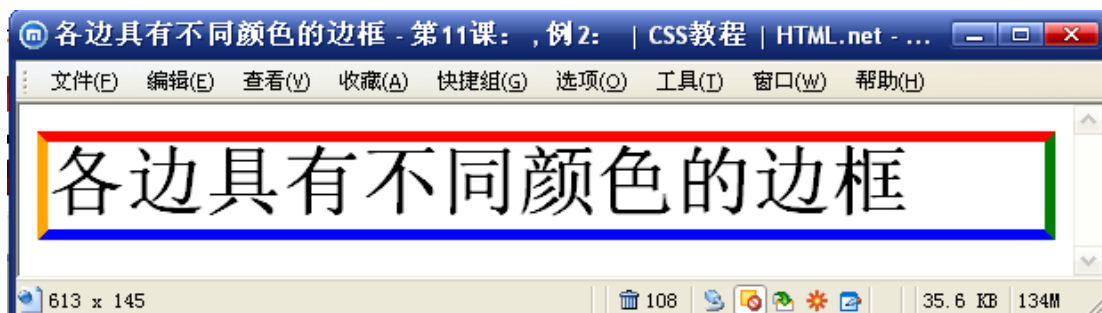
我们可以将上面三个有关边框的 CSS 属性组合起来使用, 从而制造出多种多样的变化。来举个例子, 我们要为一个文档中的 h1、h2、u1 和 p 等元素分别定义不同的边框。尽管其显示效果也许并不那么美观, 但是它很好地向你展示了多种变化的可能:

```
h1 {  
    border-width: thick;  
    border-style: dotted;  
    border-color: gold;  
}  
  
h2 {  
    border-width: 20px;  
    border-style: outset;  
    border-color: red;  
}  
  
p {  
    border-width: 1px;  
    border-style: dashed;  
    border-color: blue;  
}  
  
ul {  
    border-width: thin;  
    border-style: solid;  
    border-color: orange;  
}
```



我们也可以为上边框、下边框、右边框、左边框分别指定特定的 CSS 属性。具体做法如下例所示:

```
h1 {  
    border-top-width: thick;  
    border-top-style: solid;  
    border-top-color: red;  
  
    border-bottom-width: thick;  
    border-bottom-style: solid;  
    border-bottom-color: blue;  
  
    border-right-width: thick;  
    border-right-style: solid;  
    border-right-color: green;  
  
    border-left-width: thick;  
    border-left-style: solid;  
    border-left-color: orange;  
}
```



缩写[border]:

跟许多其他属性一样,你也可以将有关边框的 CSS 属性缩写为一个 border 属性。让我们看一个例子:

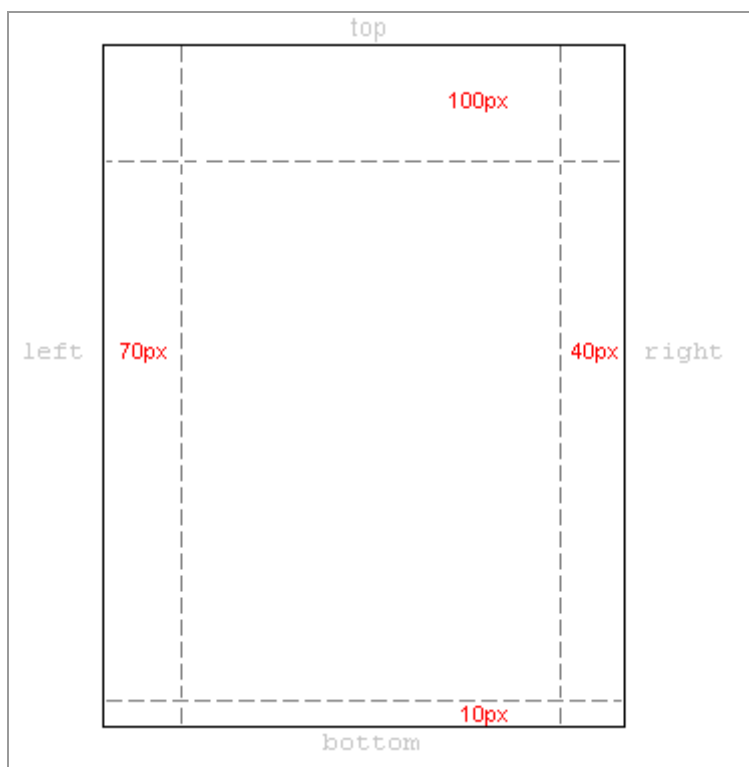
```
p {  
    border-width: 1px;  
    border-style: solid;  
    border-color: blue;  
}
```

可被缩写为:

```
p {  
    border: 1px solid blue;  
}
```

外边距和内边距:

一个元素有上 (top)、下 (bottom)、左 (left)、右 (right) 四个边。外边距 (margin) 表示从一个元素的边到相邻元素(或者文档边界)之间的距离。在下面这个例子中, 我们将了解如何为文档本身 (即 body 元素) 定义外边距。下图显示了我们对外边距的要求。

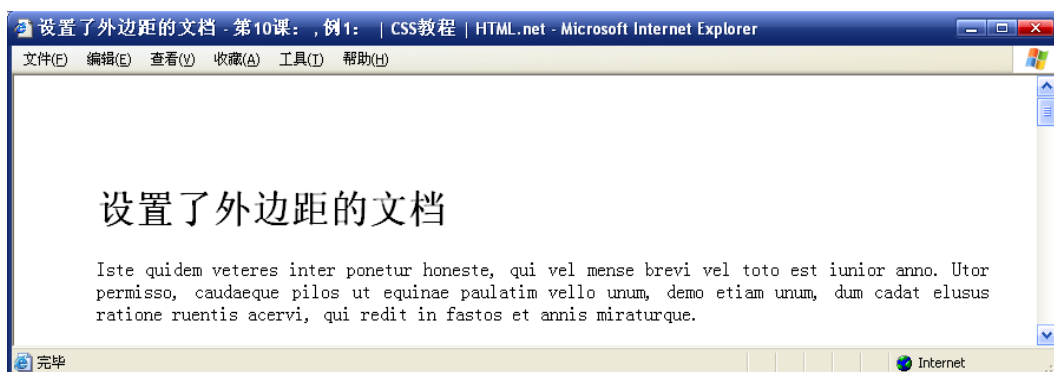


满足上述要求的 CSS 代码如下:

```
body {  
    margin-top:100px;  
    margin-right:40px;  
    margin-bottom:10px;  
    margin-left:70px;  
}
```

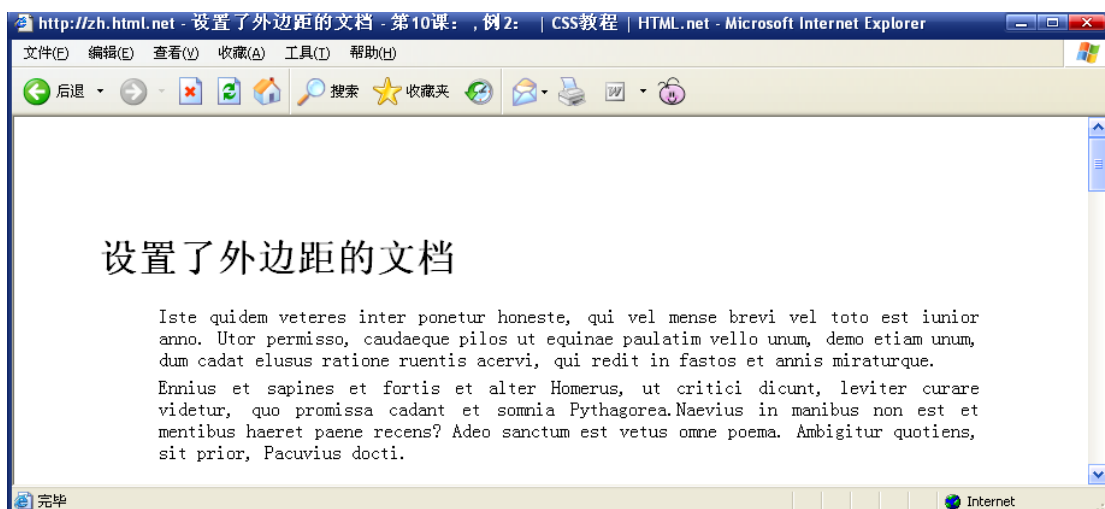
或者你也可以采用一种较优雅的缩写形式:


```
body {  
    margin: 100px 40px 10px 70px;  
}
```



几乎所有元素都可以采用跟上面一样的方法来设置外边距。例如, 我们可以为所有用<p>标记的文本段落定义外边距:

```
body {  
    margin: 100px 40px 10px 70px;  
}  
  
p {  
    margin: 5px 50px 5px 50px;  
}
```

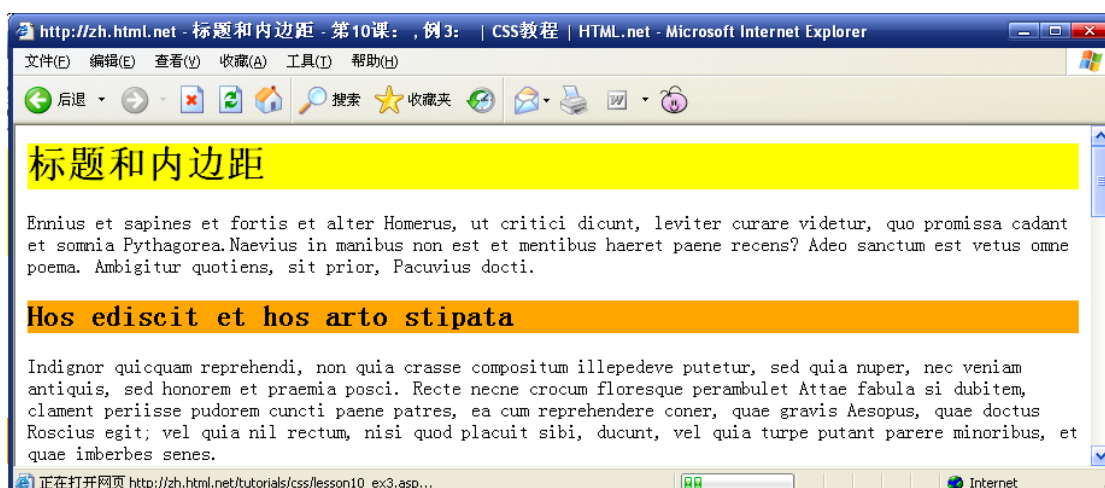


为元素设置内边距:

内边距 (padding) 也可以被理解成“填充物”。这样理解是合理的, 因为内边距并不影响元素间的距离, 它只定义元素的内容与元素边框之间的距离。

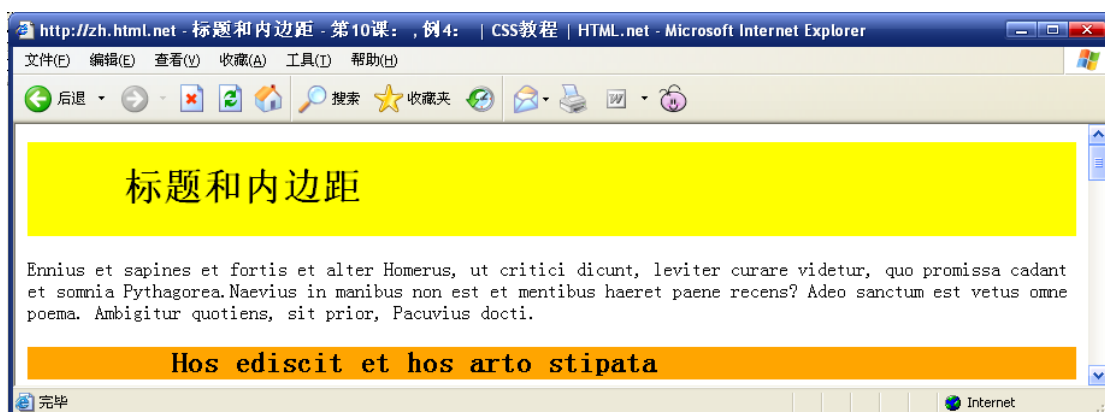
下面我们通过一个简单的例子来说明内边距的用法。在这个例子中, 所有标题都具有背景色:

```
h1 {  
    background: yellow;  
}  
  
h2 {  
    background: orange;  
}
```



通过为标题设置内边距, 你可以控制在标题文本周围填充多少空白:

```
h1 {  
    background: yellow;  
    padding: 20px 20px 20px 80px;  
}  
  
h2 {  
    background: orange;  
    padding-left: 120px;  
}
```

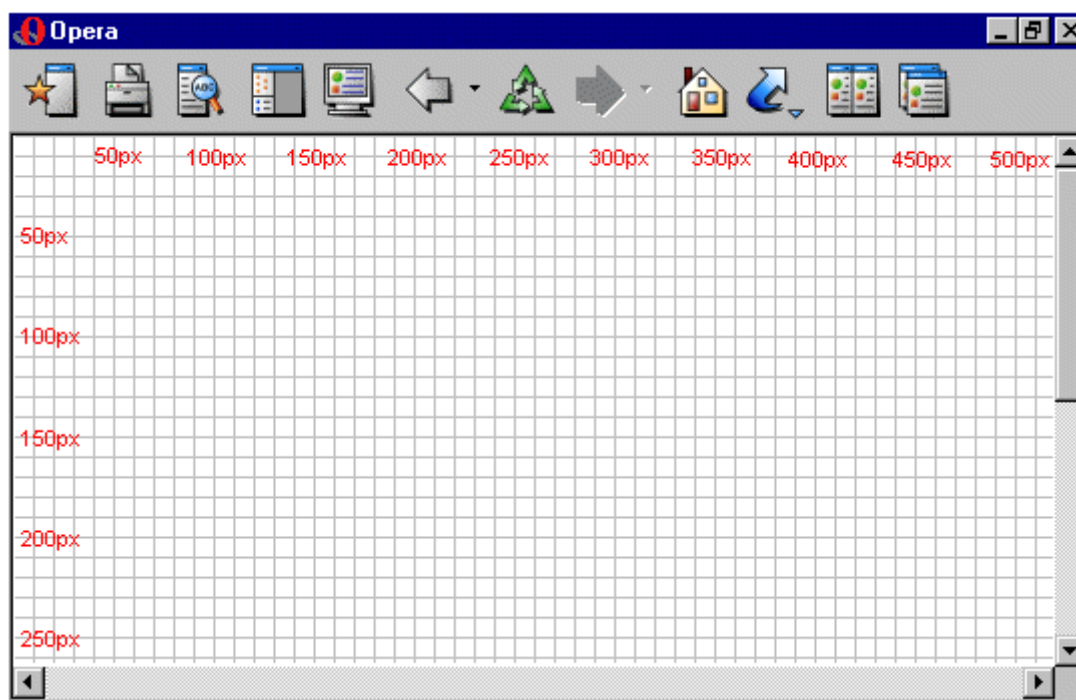


三、 定位属性

CSS 定位令你可以将一个元素精确地放在页面上你所指定的地方。

CSS 定位的原理:

把浏览器窗口想象成一个坐标系统:



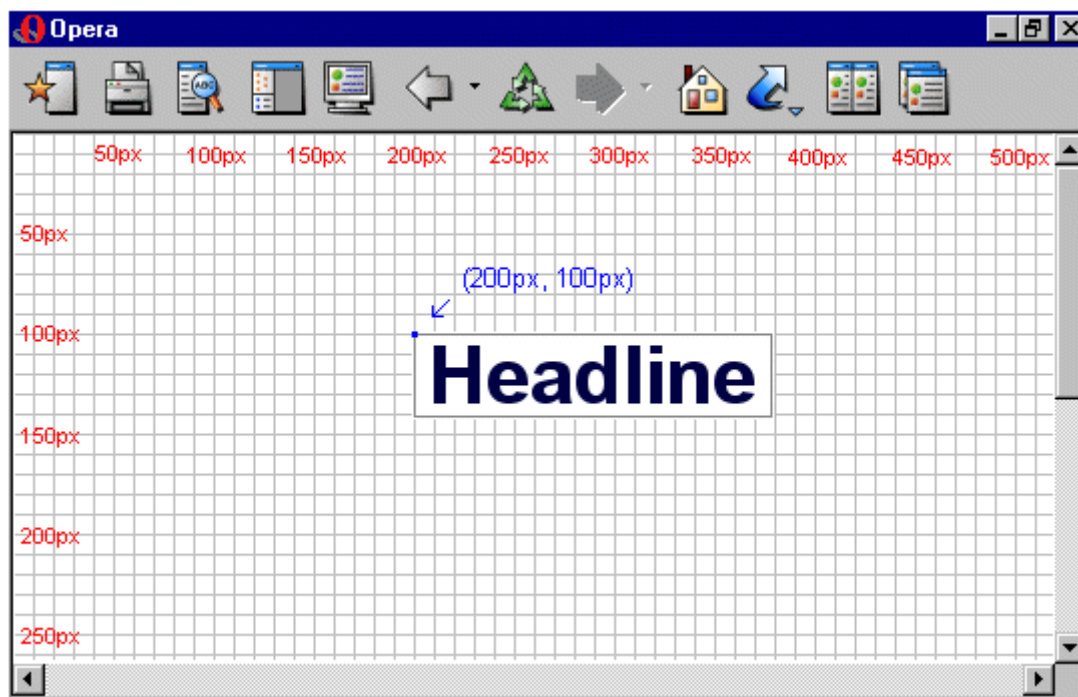
假设我们要放置一个标题:



如果我们要把这个标题放置在距文档顶部 100 像素、左边 200 像素的地方, 我们可以在 CSS 中输入以下代码:

```
h1 {  
    position: absolute;  
    top: 100px;  
    left: 200px;  
}
```

得到的显示效果如下:



正如你所看到的, 采用 CSS 定位技术来放置元素是非常精确的。相对于使用表格、透明图像或其他方法而言, CSS 定位要简单得多。

绝对定位:

一个采用绝对定位的元素不获得任何空间。这意味着: 该元素在被定位后不会留下空位。要对元素进行绝对定位, 应将 `position` 属性的值设为 **absolute**。接着, 你可以通过属性 **left**、**right**、**top** 和 **bottom** 来设定将盒子放置在哪里。

举个绝对定位的例子, 假如我们要在文档的四个角落各放置一个盒子:

```
#box1 {  
    position: absolute;  
    top: 50px;  
    left: 50px;  
}  
  
#box2 {  
    position: absolute;  
    top: 50px;  
    right: 50px;  
}  
  
#box3 {  
    position: absolute;  
    bottom: 50px;  
    right: 50px;  
}  
  
#box4 {  
    position: absolute;  
    bottom: 50px;  
    left: 50px;  
}
```



相对定位:

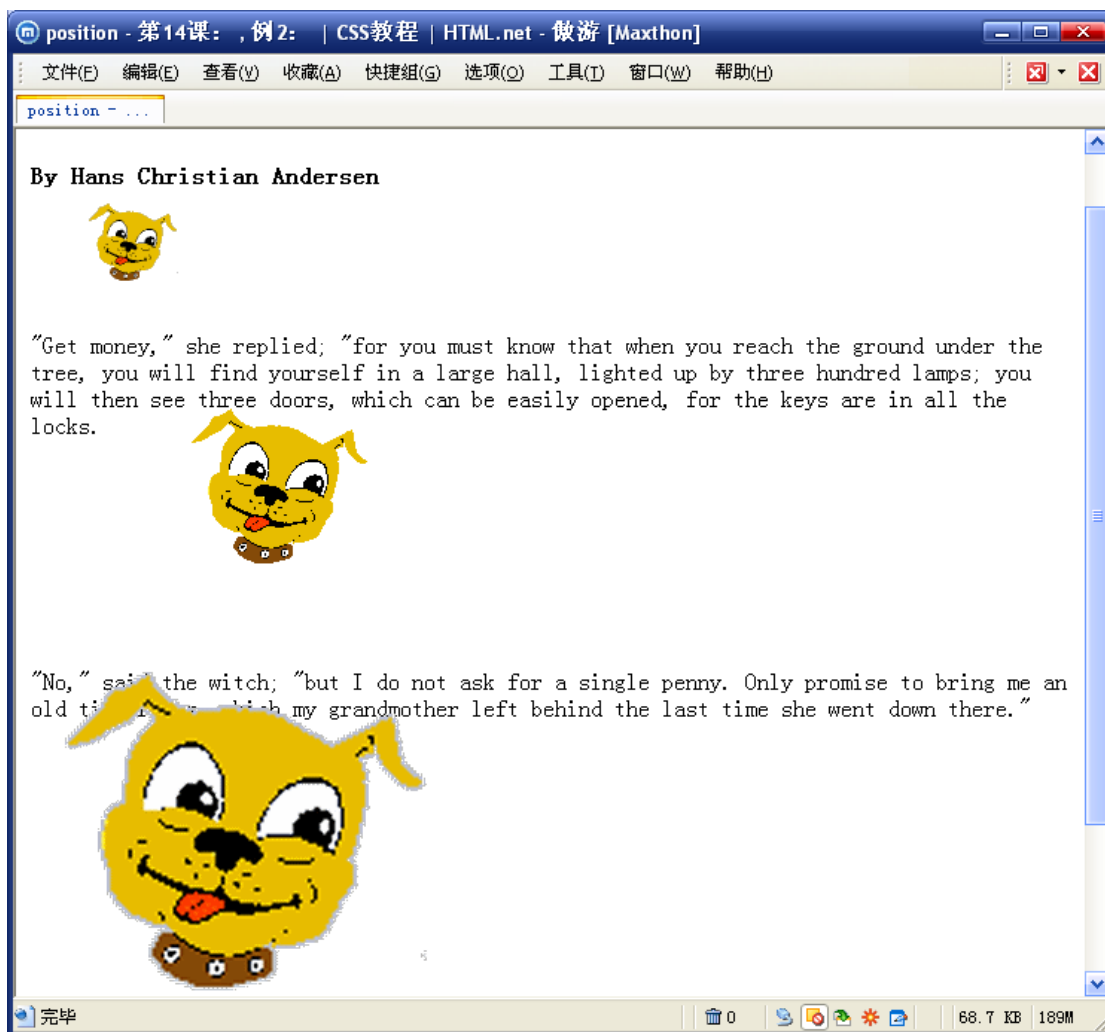
要对元素进行相对定位, 应将 `position` 属性的值设为 `relative`。绝对定位与相对定位的区别在于计算位置的方式。

采用相对定位的元素, 其位置是相对于它在文档中的原始位置计算而来的。这意味着, 相对定位是通过将元素从原来的位置向右、向左、向上或向下移动来定位的。采用相对定位的元素会获得相应的空间。

举个相对定位的例子, 我们可以相对于三张图片在页面上的原始位置来对它们进行相对定位。注意这些图片将在文档中各自的原始位置处留下空位。

```
#dog1 {  
    LEFT: 35px; BOTTOM: 15px; POSITION:  
relative  
}  
#dog2 {  
    LEFT: 100px; BOTTOM: 50px; POSITION:  
relative  
}  
#dog3 {  
    LEFT: 10px; BOTTOM: 70px; POSITION:  
relative  
}
```

```
<BODY>  
  <H1>The Tinder-Box</H1>  
  <P><STRONG>By Hans Christian Andersen</STRONG></P>  
  <DIV id=dog1>  
    <IMG src="dog1.gif"></DIV>  
  <P>"Get money," she replied; "for you must know that when you reach the ground  
  under the tree, you will find yourself in a large hall, lighted up by three  
  hundred lamps; you will then see three doors, which can be easily opened,  
  for the keys are in all the locks.  
  </P>  
  <DIV id=dog2><IMG src="dog2.gif"></DIV>  
  <P>"No," said the witch; "but I do not ask for a single penny. Only promise  
  to bring me an old tinder-box, which my grandmother left behind the last time  
  she went down there." </P>  
  <DIV id=dog3><IMG src=" dog3.gif"></DIV>  
  <P>Then he went into the third room, and there the dog was really hideous;  
  his eyes were, truly, as big as towers, and they turned round and round in  
  his head  
  like wheels...</P>  
</BODY>
```



第七章 HTML 中框架、层的运用

本章目标: 掌握框架结构<frameset><frame><iframe>

掌握组织元素: span 和 div

本章重点: 框架结构<frameset><frame><iframe>

本章难点: 框架的搭建

一、 框架

使用框架,可以在一个浏览器窗口中显示不止一个 HTML 文档。这样的 HTML 文档被称为框架页面,它们是相互独立的。:

使用框架的不利因素有:

- 网站开发者需要关心更多 HTML 文档的情况。
- 打印整个页面变得困难。

frameset 标签:

- <frameset>标签定义了如何将窗口拆分成框架。
- 每个 frameset 标签定义了一组行和列。
- 行/列的值指明了每个行/列在屏幕上所占的大小

frame 标签:

- <frame>标签定义了每个框架中放入什么文件。

下面这个例子中,有一个两列的分栏。第一个被设置成窗口宽度的 25%,第二个被设置成窗口宽度的 75%。页面“frame_a.htm”被放在第一个分栏中,“frame_b.htm”被放在第二个分栏中。

```
<frameset cols="25%,75%">
  <frame src="frame_a.htm">
  <frame src="frame_b.htm">
</frameset>
```

基本注意点——有用的技巧:

假如一个框架有可见边框,用户可以拖动边框来改变它的大小。如果不想让用户改变大小,可以在<frame>标签中加入: noresize="noresize"。

给不支持框架的浏览器写上<noframes>标签。

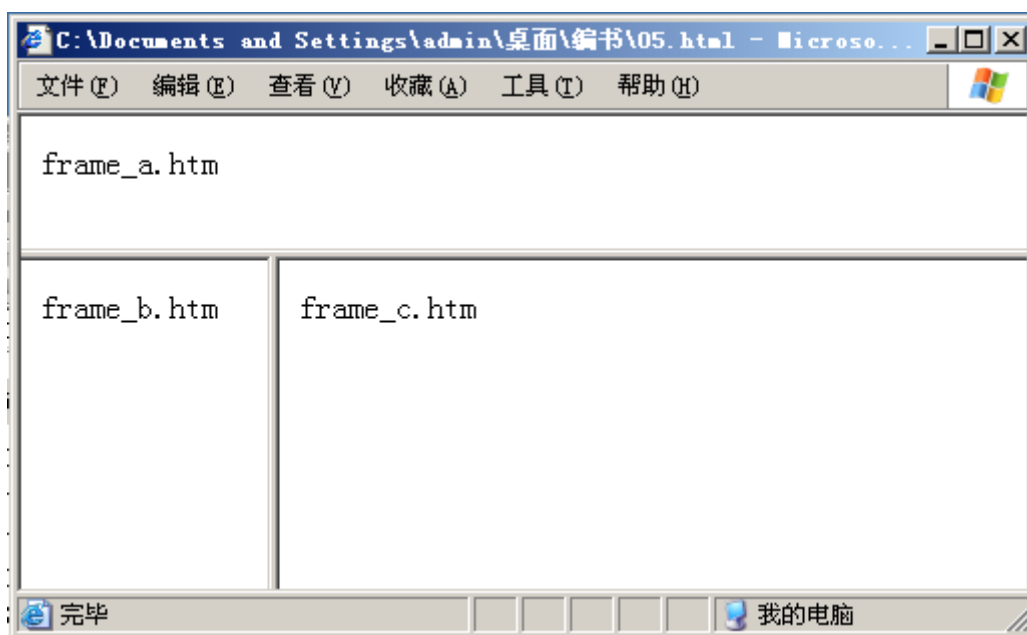
更多示例:

混合框架:

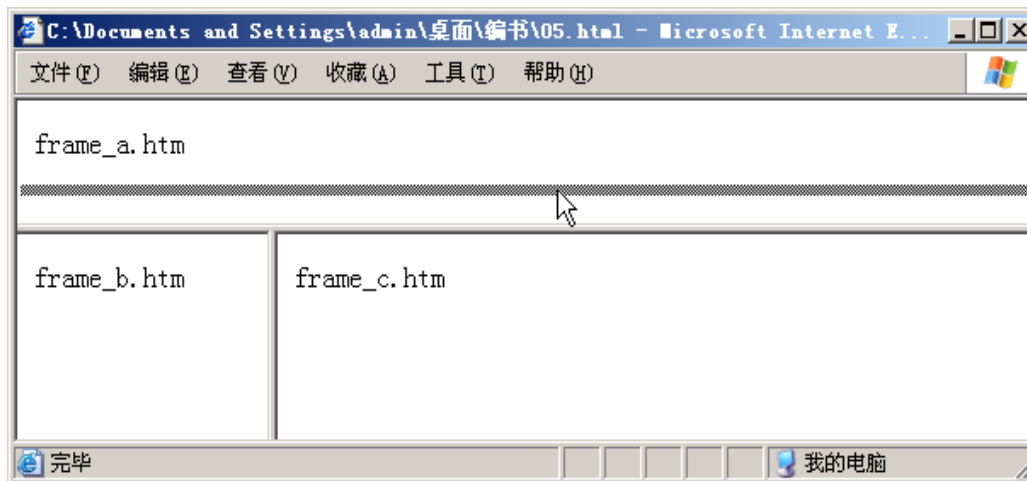
```
<html>
```



```
<frameset rows="50%, 50%">
  <frame src="frame_a.htm">
  <frameset cols="25%, 75%">
    <frame src="frame_b.htm">
    <frame src="frame_c.htm">
  </frameset>
</frameset>
</html>
```



这个例子说明了怎样把三个页面以行列混合的方式放在框架中。

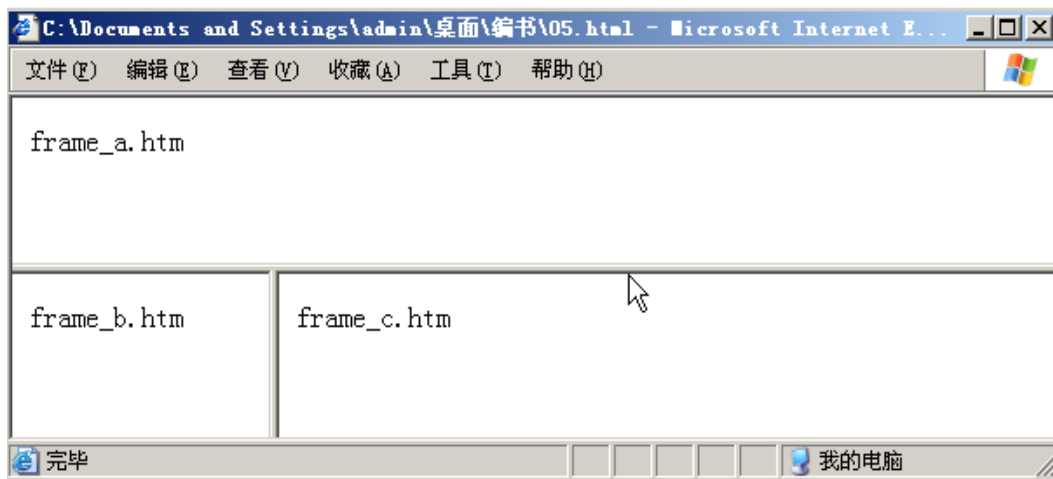


如上图所示: 把鼠标移动到框架边界上, 你会发现可以调整框架大小。

使用了 `noresize="noresize"` 的框架:

```
<html>
  <frameset rows="50%, 50%">
    <frame noresize="noresize" src="frame_a.htm">
    <frameset cols="25%, 75%">
```

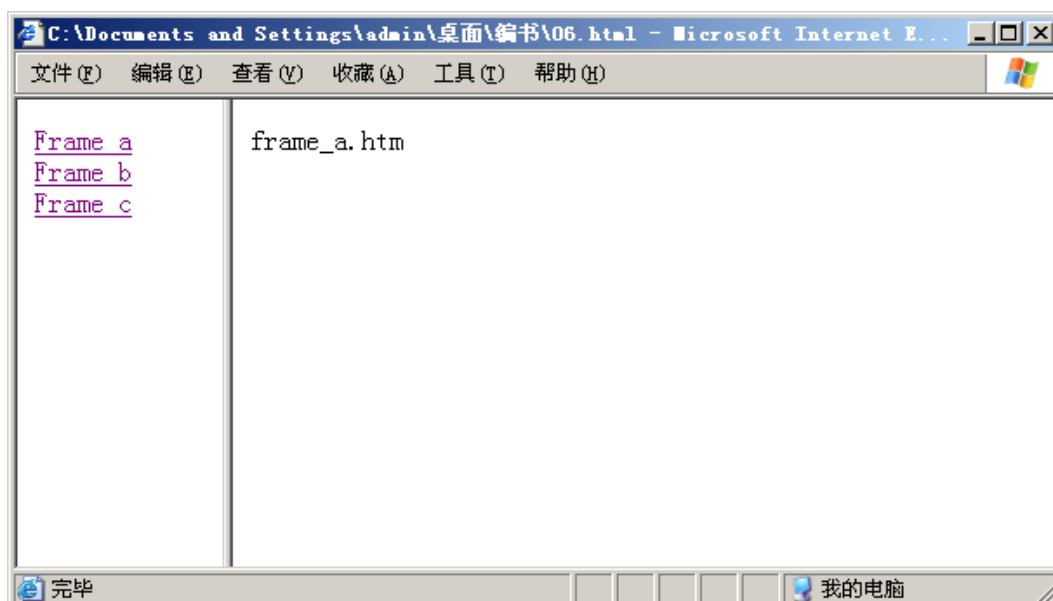
```
<frame noresize="noresize" src="frame_b.htm">
<frame noresize="noresize" src="frame_c.htm">
</frameset>
</frameset>
</html>
```



这个例子说明了 noresize 属性。这个框架是不可改变大小的，把鼠标移动到框架边界上，你会发现无法调整大小。

导航框架：

```
<html>
<frameset cols="120,*">
  <frame src="frame_link.htm">
  <frame src="frame_a.htm" name="showframe">
</frameset>
</html>
```



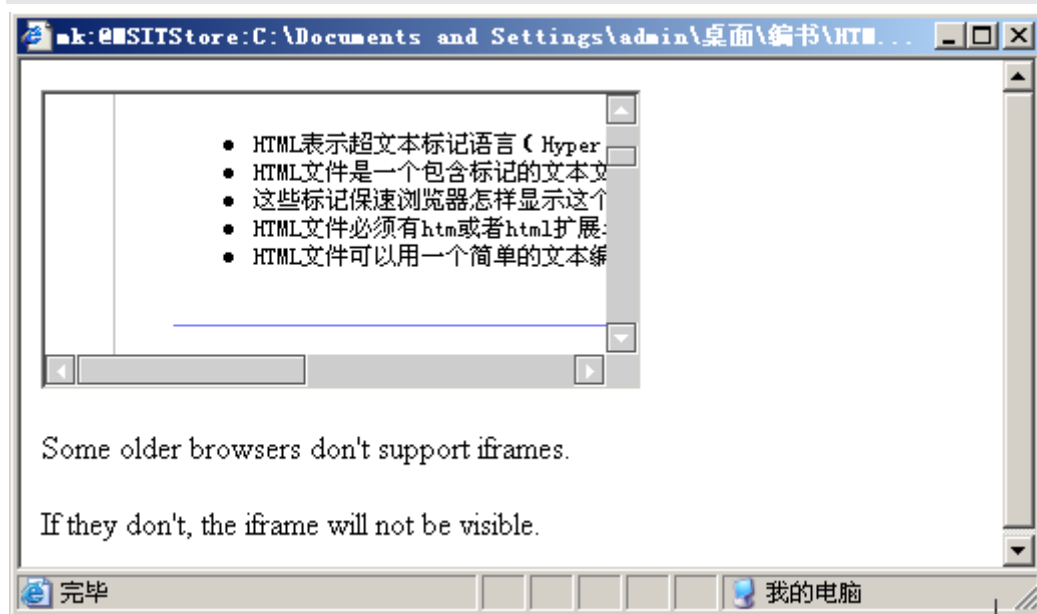
这个例子说明了如何创建一个导航框架。导航框架包含了一系列链接，它们的目标页面在第二个框架中。文件“frame_links.htm”包含了三个链接，链接的代码如下：

```
<a href = "frame_a.htm" target = "showframe">Frame a</a>
<a href = "frame_b.htm" target = "showframe">Frame b</a>
<a href = "frame_c.htm" target = "showframe">Frame c</a>
```

第二个框架将显示链接到的页面。

内联框架:

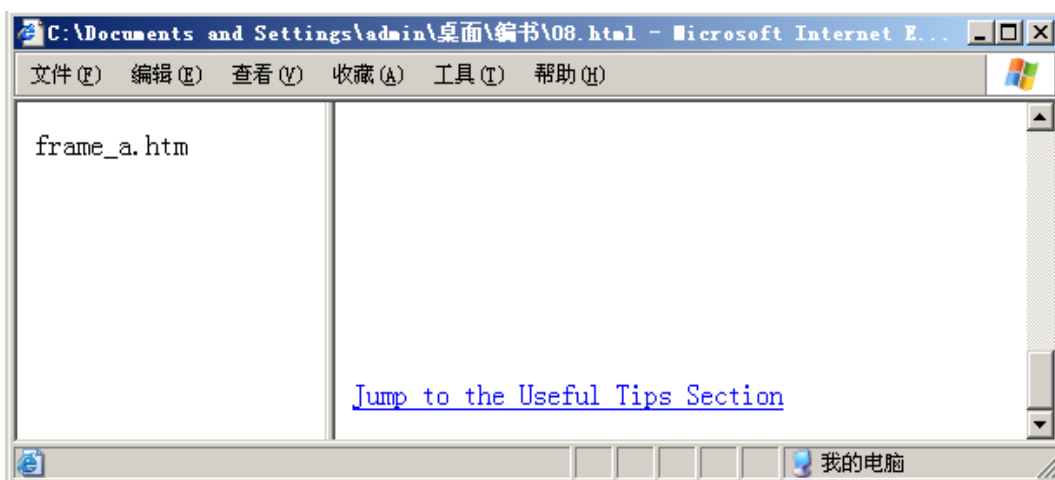
```
<html>
  <body>
    <iframe src="intro.htm"></iframe>
    <p>Some older browsers don't support iframes.</p>
    <p>If they don't, the iframe will not be visible.</p>
  </body>
</html>
```



这个例子说明了如何创建一个内联框架（包含在 HTML 页面里的框架）。

在框架内跳转到指定章节:

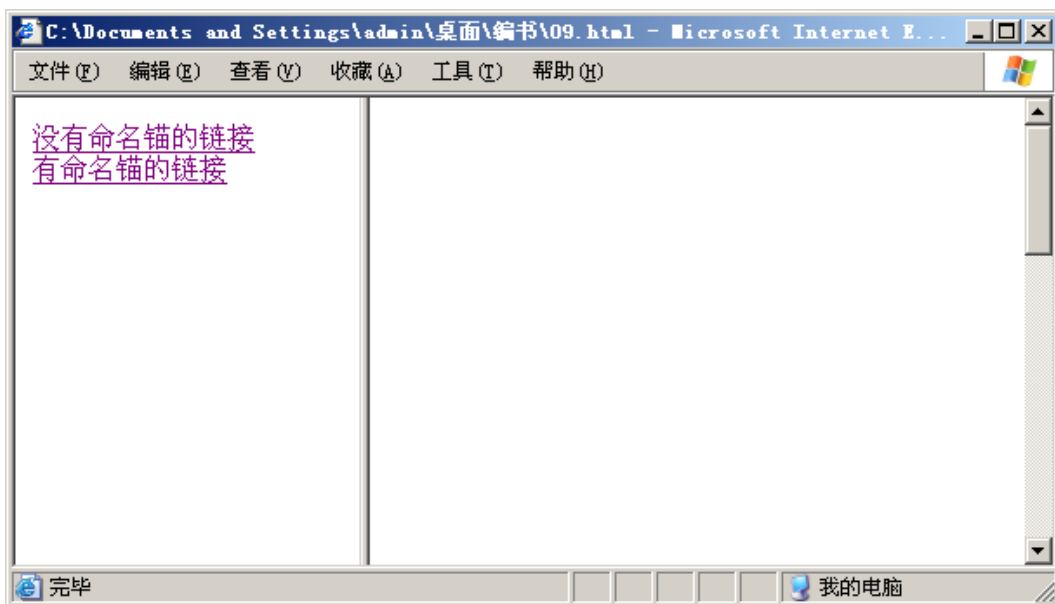
```
<html>
  <frameset cols="30%, 70%">
    <frame src="frame_a.htm">
    <frame src="frame_section.htm#C10">
  </frameset>
</html>
```



这个例子显示了两个框架页。其中一个的源是一个文件的指定章节，该章节在文件“frame_section.htm”中使用代码[](#)指定。

使用导航框架跳转到指定章节：

```
<html>
<frameset cols="200,*">
  <frame src="frame_linksection.htm">
  <frame src="frame_section.htm" name="showframe">
</frameset>
</html>
```



这个例子显示了两个框架页。左边的导航框架包含了一系列以第二个框架为目标的链接（“frame_linksection.htm”），第二个框架显示链接文件（“frame_section.htm”）。导航框架中的一个链接指向目标文件中的指定章节。文件“frame_link”中的 HTML 代码是像这样的：

```
<a href = "frame_section.htm" target = "showframe">没有命名锚的链接</a><br>
<a href = "frame_section.htm#C10" target = "showframe">有命名锚的链接</a>
```

二、 组织元素: span和div

span 和 div 元素用于组织和结构化文档, 并经常联合 class 和 id 属性一起使用。
用 span 组织元素:

span 元素可以说是一种中性元素, 因为它不对文档本身添加任何东西。但是与 CSS 结合使用的话, span 可以对文档中的部分文本增添视觉效果。

让我们用一句本杰明·弗兰克林 (Benjamin Franklin) 的名言来举个例子:

```
<p>早睡早起  
使人健康、富裕又聪颖。</p>
```

假设我们想用红色来强调弗兰克林先生所认为的“不要在睡眠中度过一天”的好处, 我们可以用标签来标记上述每一点好处。然后, 我们将这几个 span 设置为相同的 class。这样, 我们稍后就可以在样式表里针对这个 class 定义特定的样式。

```
<p>早睡早起  
使人<span class="benefit">健康</span>、  
<span class="benefit">富裕</span>  
和<span class="benefit">聪颖</span>。</p>
```

相应的 CSS 代码如下:

```
span.benefit {  
    color:red;  
}
```

用 div 组织元素:

如前面例子所示, span 的使用局限在一个块元素内, 而 div 可以被用来组织一个或多个块元素。

除去这点区别, div 和 span 在组织元素方面相差无几。让我们来看一个例子。我们将历届美国总统按其所属的政营分别组织为两个列表:

```
<div id="democrats">
<ul>
<li>富兰克林·D·罗斯福</li>
<li>哈利·S·杜鲁门</li>
<li>约翰·F·肯尼迪</li>
<li>林登·B·约翰逊</li>
<li>吉米·卡特</li>
<li>比尔·克林顿</li>
</ul>
</div>

<div id="republicans">
<ul>
<li>德怀特·D·艾森豪威尔</li>
<li>理查德·尼克松</li>
<li>杰拉尔德·福特</li>
<li>罗纳德·里根</li>
<li>乔治·布什</li>
<li>乔治·W·布什</li>
</ul>
</div>
```

在这里,我们可以采用跟上例同样的方法来处理样式表:

```
#democrats {
    background:blue;
}

#republicans {
    background:red;
}
```

第九章 XML 基本知识

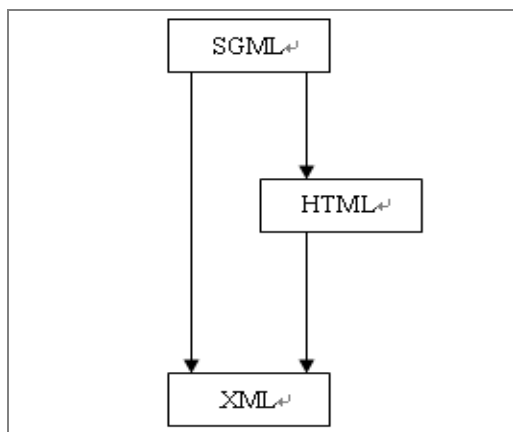
本章目标: 了解 XML 的应用范围
了解 XML 的文档结构
理解格式正规的 XML 文档的特点
熟悉有效的 XML 文档的编写规则
理解命名空间
本章重点: 熟悉有效的 XML 文档的编写规则
本章难点: 理解命名空间

一、XML的应用范围

人类一直在不断地尝试改进自己的发明, 其中也包括人类最伟大的发明——文字的构成。第一个文本处理系统是用纸笔记录文字。现在, 计算机文本处理器已经取代了手工处理, 它不仅包含原始文档, 还负责设置格式、出版和管理。在这些方便的功能整合到字处理之前, 是由排字工人遵循书面标记说明来完成所有格式编排的。正是利益于这种实践, 人们将“标记”这个词加入到 HTML 和 XML。顾名思义, 标记是指加上记号。文本处理环境(如 XML)中使用了相同的标记过程。本意讲述标记语言的历史和创建 XML 文档的方法。

使用脚本语言或 DHTML 能够以各种方式显示信息。这就要求必须为相同的输出编写不同的代码以供不同的浏览器使用, 因为这些语言不能跨浏览器兼容。

XML (eXtensible Markup Language, 可扩展标记语言) 克服了这些缺点。顾名思义, XML 是可扩展的, 即开发人员可以定义自己的一组标签, 并使其他的人或程序能够理解这些标签。HTML 是单标记语言, 为特定应用设计, 而 XML 则是一系列的标记语言。因此, XML 比 HTML 灵活得多。实际上, 由于 XML 标签表示了数据的逻辑结构, 不同的应用可以通过不同的方式来解释和使用这些标签。Web 上的数据大多是继承的, XML 继承了 SGML 和 HTML 的优点。也就是说, 它不仅继承了 SGML 的特色, 还结合了 HTML 的特色。它采用了 SGML 的主要框架, 有时, 人们也将 XML 称为 SGML 的子集。因此, HTML 是 SGML 的应用, 而 XML 是 SGML 的子集。下图显示了标记语言的层次结构。



使用标签对文档进行标记以提供有关内容的信息, 不仅能加快搜索速度, 而且还能降低网络流量。XML 是由 SGML 修整并改造而来, 它是一种元语言, 用于描述其他语言。

我们可以使用 **XML** 为特定目的创建自己的标记语言（如化学标记语言）。**XML** 是基于文本的格式，允许开发人员描述结构化数据并在各种应用之间发送和交换这些数据，这样客户端就可以显示并自定义数据。

XML 还有助于在服务器之间传输结构化数据。有许多信息是分布在不同的和不匹配的数据库中。如有必要，**XML** 允许通过使用自定义格式来标识、交换和处理这些数据库可以理解的数据。

XML 和 **HTML** 有许多相同点和不同点。**XML** 描述数据，如城市名称、温度和气压；**HTML** 定义描述数据显示方式的标签，如使用项目符号列表或表格。但 **XML** 允许开发人员定义任意数量的标签集，使用开发人员有很大的灵活性来决定要使用哪些数据，并确定数据的适用标准或自定义标签。

XML 应用范围：

对于 **Internet** 和大型企业 **Intranet** 环境，**XML** 都是十分有价值的。这是因为它通过灵活、开放及基于标准的格式、访问遗留数据库及将数据发送至 **Web** 客户端的新方式，来提供了协同工作能力。不仅可以更快地构建应用，而且更易于维护，还可以通过不同的样式表提供多个结构化数据的视图，这将在后面的章节中介绍。

下面使用两个救命来说明使用 **XML** 会给个人、公司和组织带来什么好处。

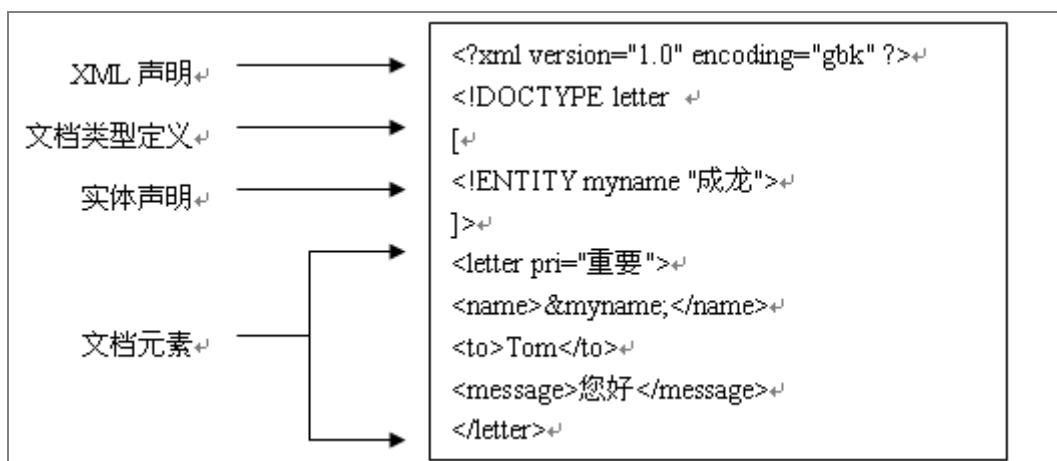
SABRE：**SABRE** 集团是主要的国际旅游服务商之一，通过旅行社和 **Web** 提供电子旅游预约。他们使用 **Java** 应用程序将旅游信息转换为 **XML**，这样全球的移动电话用户都可以通过移动电话查找、预订和购买机票和门票。**XML** 自动转化为 **Wireless Markup Language**（无线标记语言，**WML**），这是在移动电话上构建应用的标准语言。对于此应用来说，**XML** 的优势在于其可扩展性、开发速度、能够使用 **XML** 构建标准资料库以及将它转换到特定的环境中，本例中为移动电话。

化学标记语言：化合物和化学分子是原子的复杂组合。大概有两千万个已知分子，直到最近才有机器可读的表示分子的标准方式。诺丁汉大学的 **Peter Murray Rust** 教授和伦敦大学帝国理工学院的 **Henry Rezza** 博士这两们英国科学家开发出了用 **XML** 描述分子的标准方式。**Chemical Markup Language**（化学标记语言，**CML**）预期能够为化学行业节省大量人力物力，还有助于化学家之间以及其他相关学科（如生物和医药）之间的交流。在这种应用中使用 **XML** 的关键优势之一在于 **XML** 提供大量的工具，有助于迅速高效地创建 **CML** 应用。

二、 XML 的文档结构

XML 文档是由一组使用唯一名称标识的实体组成。所有文档都以根或文档实体开始，而且所有实体都是可选的。实体可以被视为更复杂功能的别名。单个实体名称可以代替许多文本。在别名方案中，每当需要引用某个文本时，只需要使用别名，处理器会展开别名的内容。

XML 文档也有一种逻辑结构。逻辑上，文档的组成包括声明、元素、注释、字符引用和处理指令（在文档中使用显式标记表示），如下图所示。



XML 文档始终以一个声明开始, 这个声明指定该文档遵循 XML1.0 规范。

XML 声明:

XML 声明的语法如下所示。

```
<?xml version="1.0" ?>
```

XML 声明是可选的, XML1.0 版本是默认值。W3C 规范建议使用 XML 声明, 这样可以为文档匹配合适的解析器。发布更新版本时, 需要提供相应的 XML 版本号。

XML 声明是处理指令, 告知处理代理该文档已经标记为 XML 文档, 它还告诉解析器和其他应用程序应如何处理文件中的数据。包括 XML 声明在内的所有处理指令都以 “<?” 开始, 以 “>” 结束。“<?” 后面是处理指令的名称, 即 “xml”。XML 处理指令要求指定一个 version 属性, 并允许指定可选的 standalone 和 encoding 属性。XML 声明至少应有保留名称 xml 以及一个版本号, 只有版本号是必需的。encoding 详细信息和 standalone 声明可以跟在版本号后面, 如下所示。

```
<?xml version="1.0" standalone="no" encoding="UTF-8" ?>
```

如果包括了可选属性, 则必须先指定版本。

standalone 属性可以设置为 yes 或 no。yes 指定不使用外部声明, 而 no 则表示将引用外部声明。

所有 XML 解析器都必须支持与 ASCII 相应的 8 位或 16 位 Unicode 编码。“encoding="UTF-8"”指定作者使用的字符编码。UTF8 与 8 位 ASCII 字符相对应。“GB2312”或“GBK”与中文字符集相对应。

根元素:

根元素只能有一个, 用于描述文档的功能。每个 XML 文档都有一个根元素。<HTML> 是 HTML 的根元素。在 XML 中, 可以自定义根元素。例如, 使用<BOOK>作者为根元素, 如下所示:

```
<?xml version="1.0" standalone="no" encoding="UTF-8" ?>
<BOOK>
</BOOK>
```

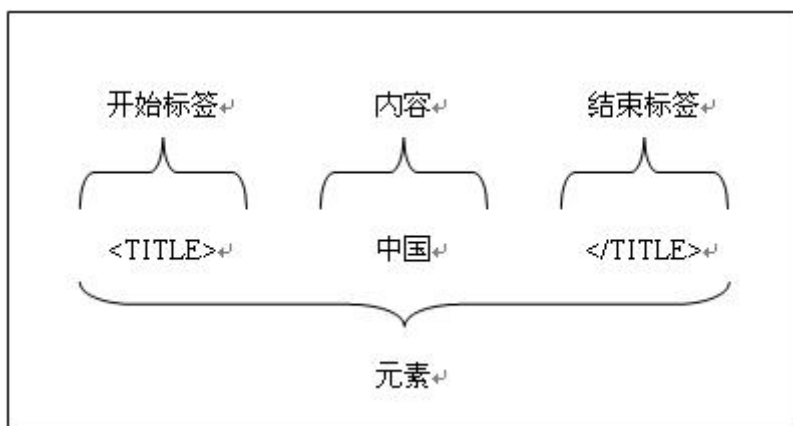
XML 代码:

根据应用需要创建自定义元素和属性。元素涉及标签及其内容。例如:

```
<B>BOLD TEXT</B>
```

标签包括尖括号以及尖括号中的文本。例如, <P>、<I>和</I>是在 HTML 中使用的一些标签。标签告诉用户代理(浏览器)处理结束标签之间的内容。

元素是 XML 内容的基本单元。开始标签, 如<element_name>, 标志着元素的开始, 而结束标签, 如</element_name>, 标志着元素的结束。图 1.5 举例说明了元素的组成部分。



标签组成了 XML 标记的主要部分。XML 标签与 HTML 标签非常相似。

数据与标记:

XML 文档由数据以及描述该数据的标记组成。数据通常是字符数据, 但也可以是二进制数据。标记包括标签、注释、处理指令、DTD 和引用等。

以下是一个字符数据和标记的简单示例。

```
<NAME>成龙</NAME>
```

在本例中, <NAME>和</NAME>是标记, “成龙”是字符数据。

注释:

有时, 需要在 XML 文档中包含某些标签, 而 XML 解析器 (XML 解析器帮助计算机解释 XML 文件) 应该忽视这些标签, 这种类型的文本称为注释文本。在 HTML 中, 使用 <!--和-->语法指定注释。在 XML 中也以相同的方式指定注释。注释的语法如下所示。

```
<!-- 在此处写注释 -->
```

使用注释时要遵循以下规则:

注释文本中不应包含“-”或“--”, 因为可能会使 XML 解析器产生混淆。

注释绝对不能放在标签中。因此, 以下代码是错误的。

```
<NAME <!-- 姓名 -->>汤姆·克鲁斯</NAME>
```

注释不能放在实体声明中, 也不能放在 XML 声明之前。XML 声明必须始终在任何 XML 文档的第一行。

注释可用于注释标签集。因此, 在以下的代码中, 除汤姆·克鲁斯以外的所有名字都会被忽略。

```
.....
<!-- 不显示
<NAME>凯特·温斯莱特</NAME>
<NAME>妮可基曼</NAME>
<NAME>阿诺德</NAME>
-->
<NAME>汤姆·克鲁斯</NAME>
.....
```

注释不能嵌套

处理指令:

处理指令是为使用该 XML 文档的应用提供的一则信息。该指令直接传递到使用该解析器的应用,应用可以将信息传递到另一个应用,或自行对信息进行解释。XML 声明也是一个处理指令。

处理指令的格式相同,如下所示。

```
<?xml :stylesheet type="text/xml" ?>
      ^           ^           ^
      应用的名称  指令信息
```

所有处理指令都必须以<? 开始,以 ?> 结束。

标签间的字符数据的分类:

开始标签和结束标签之间的文本被定义为字符数据。字符数据可以是“<”以外的任何合法(Unicode)字符。“<”字符预留作标签的开始字符。

Unicode 定义一个完全国际化的字符集,可以表示人类语言中的所有字符。它统一了许多字符集,如拉丁字符、希腊字母、阿拉伯文等。

字符数据可以分为以下两类:PCDATA,CDATA。

具体叙述如下:

PCDATA: PCDATA 表示已解析的字符数据。字符数据可被视为 XML 元素的开始标签和结束标签之间的文本。PCDATA 是将来通过解析器进行解析的文本。文本中包含的标签将被视为标记,实体将会扩展。

CDATA: CDATA 指字符数据。CDATA 是不通过解析器进行解析的文本,文本中包含的标签将不被视为标记,实体不会扩展。在 CDATA 块中,XML 解析器会忽略所有标签和实体引用。为了便于包含大量的特殊字符,提供了 CDATA 块。下面来看以下一段代码。

```
.....
<SAMPLE>
<![CDATA[<DOCUMENT>
<NAME>成龙</NAME>
<EMAIL>jackie@usa.com</EMAIL>
</DOCUMENT>]]>
</SAMPLE>
.....
```

在以上代码中,不允许在 CDATA 块之内使用字符串“]]>”,因为它表示 CDATA 块的结束。

实体:

实体是 XML 的存储单元。实体可以包含常用的短语、键盘字符、文件、数据库刻录或任何包含数据的项。

在文档中使用实体可以避免在文档中重复键入长段的文本。可以将一个实体名和文本关联,然后每当需要在文档中放入该文本时,就使用此实体名。处理文档时,该实体名将被替换为指定的文本。

在 XML 中,有些字符(如<、>或&)可以包括在文本中,但不能以字面格式存在,否则解析器会生成错误。

XML 规范定义了一个预定义字符实体集,可以用于取代字符的字面格式。一共有 5 个这种表示字符的预定义实体,这些字符可能会与 XML 标记代码混淆,见下表。

<	<
>	>
&	&
"	"
'	'

使用实体引用将实体插入 XML 文档。解析器遇到实体引用时,会将引用替换为实体的内容。例如,可以在标记中使用的实体引用,如:

```
<ORDER VALUE="He said, "Don't jump out of the window!"">
```

应写成:

```
<ORDER VALUE="He said, &quot;Don&apos;t jump out of the window!&quot;">
```

基本上,实体是用于定义常见文本的快捷方式的变量。实体分为两类:一般实体、参数实体。

一般实体:

可以在 XML 文档中的任何位置出现的实体称为一般实体,实体可以声明为内部实体或外部实体。内部实体仅存在于声明它们的文档中,外部实体则指文档外的存储单元。

一般实体命名如下:

```
<!ENTITY address "要以实体表示的文本">
```

替换文本后,该示例会变成:

```
<!ENTITY address "我的地址: 美国洛杉矶 第 10 大街 12 号 12 套房">
```

上面指定的实体是一个内部实体。

外部实体使用一个标识符指向文档外的存储单元。外部实体标识符分为两种类型，**SYSTEM**（系统）和 **PUBLIC**（公共）。前者用于引用本地计算机（或网络），后者用于引用公共计算机（或网络）。外部实体示例如下：

```
<!ENTITY greeting SYSTEM "test.txt">
```

在本例中，XML 处理器会将实体引用替换为 URI “test.txt” 指定的文档内容。“test.txt” 文件包含引用实体时要放入的文本。**SYSTEM** 关键字指引解析器在指定 URI 查找文件。使用实体引用将实体插入 XML 文档。实体引用指解锁实体的密钥，已经在实体声明中声明。

语法如下：

```
&ENTITY_NAME
```

如 &address;。

假设将一个地址作为实体保存在共享文件中。每当在 XML 中写入此地址时，将执行与以下代码相似的操作。

```
<LETTER>
  &address;
  <TO>成龙</TO>
  <BODY>嗨!您好!</BODY>
  <FROM>克里斯</FROM>
</LETTER>
```

地址将扩展为：

我的地址：美国洛杉矶第 10 大街 12 号 12 套房

管理实体引用的规则包括：

引用实体前，必须先在 XML 文档中声明该实体。

实体引用不应含有任何空格。例如，“& address;”或“&address ;”将导致错误。

实体引用的文本必须是格式良好的 XML 文档。

实体引用可以替代常规的字符数据，还可以在标签属性中使用实体引用。例如：

```
<CLIENT="&APTECH;" PRODUCT="&PRODUCT_ID;" QUANTITY="15">
```

参数实体：

当实体和实体引用都只需在 DTD 中出现时，则使用参数实体。参数实体，无论是内部还是外部，都只在 DTD 中使用。它们不能在文档内容中使用，因为处理器无法识别。

格式良好的参数实体看上去与一般实体相似，区别仅在于前者使用“%”说明符。假设以下示例。

```
<!ENTITY% ADDRESS "实体要表示的文本">
```

参数实体引用与一般实体引用相似。在本例中，使用“%”而不是“&”。

```
%PARAMETER_ENTITY_NAME;
```

稍后会在 DTD 一节中给出实例。

DOCTYPE 声明:

在 XML 文档中, <!DOCTYPE [...]> 声明跟在 XML 声明的后面。实体必须在文档 DOCTYPE 声明中声明。

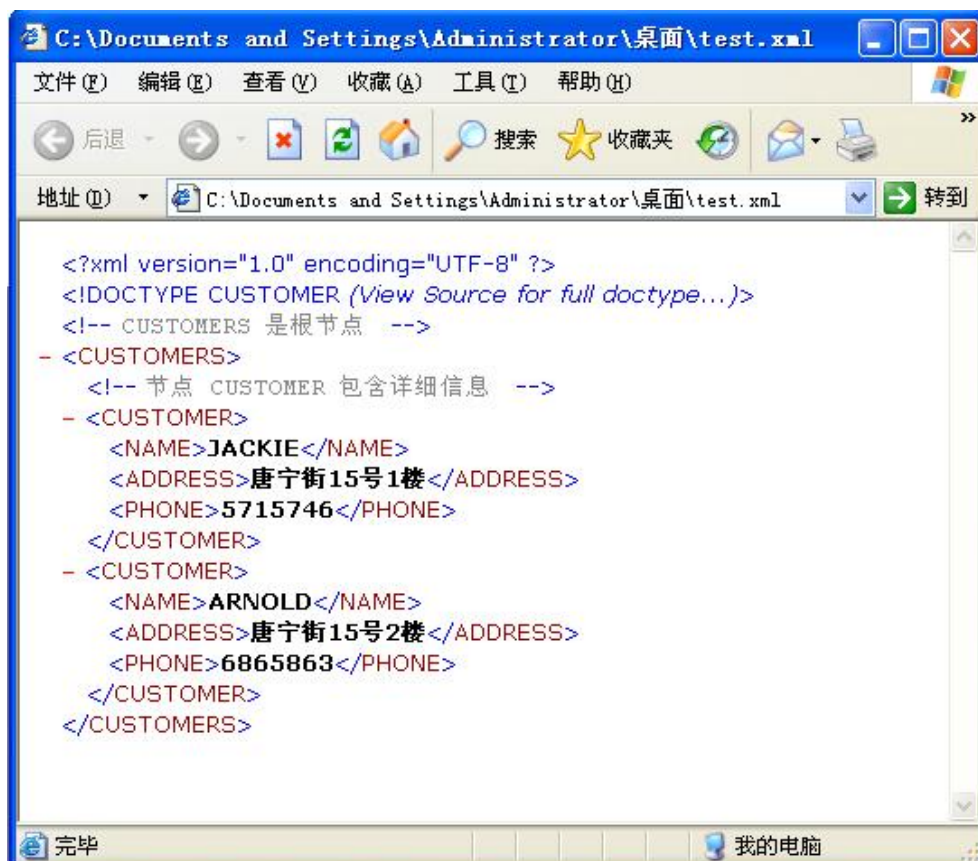
语法如下所示:

```
<?xml version="1.0" ?>
<!DOCTYPE myDoc [
...在此处声明实体...
<myDoc>
...文档正文...
</myDoc>
```

对 XML DOCTYPE 声明进行编码有助于创建文档(如例 2 所示)。使用实体时要考虑更改地址的方便性。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CUSTOMER [
<!ENTITY FIRSTFLOOR "唐宁街15号1楼">
<!ENTITY SECONDFLOOR "唐宁街15号2楼">
]>
<!--CUSTOMERS 是根节点-->
<CUSTOMERS>
  <!--节点 CUSTOMER 包含详细信息-->
  <CUSTOMER>
    <NAME>JACKIE</NAME>
    <ADDRESS>&FIRSTFLOOR;</ADDRESS>
    <PHONE>5715746</PHONE>
  </CUSTOMER>
  <CUSTOMER>
    <NAME>ARNOLD</NAME>
    <ADDRESS>&SECONDFLOOR;</ADDRESS>
    <PHONE>6865863</PHONE>
  </CUSTOMER>
</CUSTOMERS>
```

例 2 的输出结果如图 1.6 所示:



在例 2 中, 使用了 FIRSTFLOOR 和 SECONDFLOOR 这两个实体引用。实体引用替换常规字符数据, 使用以下语句引用该常规字符数据。

```
<ADDRESS>&FIRSTFLOOR;</ADDRESS>
```

遇到此行时, 早先在 XML 代码的开头声明的整个字符数据都会被替换为字符。

三、 格式良好和有效的XML文档

如果一个 XML 文档满足了最低的要求集（在定义 XML 语法的 XML 1.0 规范中定义），则该文档被视为格式良好。这些要求确保以正确的方式使用正确的词语（在 XML 规范中定义）。如果文档不满足任何一个良好格式的要求，则将发生致命错误。

有效的 XML 文档是格式良好的 XML 文档，符合 Document Type Definition（文档类型定义，DTD）的规则。DTD 定义了文档中的标记必须遵循的规则，还包含指定文档总体结构的定义以及可以接受的数据内容值的类型。有效的 XML 文档还符合 SGML 文档的标准。

至少需要一个元素：

所有格式良好的 XML 文档都必须至少有一个元素。

XML 标签区分大小写：

必须注意确保在标签集使用正确的大小写，也就是说，尽管它们在 HTML 中表示相同的意思，但<HELLO>和<hello>标签是不一致的。这是因为 XML 区分大小写。

应正确使用结束标签：

除了拼写和大小写与开始标签相同，结束标签应该在前面有一个斜杠“/”。因此，在大多数情况下，以<HELLO>作为开始标签，就应该以</HELLO>作为结束标签。在某些时候，结束标签可以省略。尤其是如果需要使用不带内容的标签，则使用带有尾随斜杠的单个开始标签，如<HR/>。

正确嵌套标签：

请注意，XML 元素可以包含其他元素，但元素的嵌套必须正确。以下代码中的元素嵌套是错误的。

```
.....  
<CONTACT>  
<NAME>成龙</NAME>  
<EMAIL>jackie@china.com>  
</CONTACT>  
</NAME>  
</EMAIL>  
.....
```

它应该是：

```
<CONTACT>  
<NAME>成龙</NAME>  
<EMAIL>jackie@china.com</EMAIL>  
</CONTACT>
```

应使用合法标签：

标签必须以一个字母、下划线（_）或冒号（:）开始，然后是字母、数字、句号（.）、冒号、下划线或连字符（-）的组合，但不能有空格。标签不应以“xml”开头，因为它是保留字。最好不要将冒号作为标签名称的第一个字符（即使这是合法的），因为它会

引起混淆。

标记名称的长度:

尽管 XML 1.0 标准中规定可以使用任何长度的名称, 但实际的 XML 处理器可能会限制标记名称的长度。XML 标签名称的长度取决于处理器。

应定义有效的属性:

标签可以指定许多支持属性。一个标签中的属性不能重复。指定一个名称和值对, 以等号 (=) 分隔, 其中, 值使用引号分隔。

```
<CAR MODEL="MARUTI 800" COLOR="WHITE">
```

和 HTML 不同, XML 规定值必须使用引号分隔。在本例中, MODEL 和 COLOR 是 CAR 标签的属性, “MARUTI 800” 是 MODEL 属性的值, 而 “WHITE” 是 COLOR 属性的值。属性命名和标签命名遵循相同的规定。不过, 必要时, 值可以包含空格、标点和实体引用。

所有值都被视为字符串。因此, 如果是标签:

```
<WATER_TANK RADIUS="5" DEPTH="20">
```

“5” 和 “20” 会转换为 XML 环境之外的数值。

应验证文档:

文档应遵循 XML 规则, 否则浏览器或任何其他 XML 阅读器都无法读取此文档。

四、 XML 文档的编写规则

XML 从 Standard Generalized Markup Language (标准通用化标记语言, SGML) 衍生而来, 和 SGML 一样, 它支持使用 DTD。文档类型声明和文档类型定义不相同。文档类型定义缩写为 DTD。DTD 指定了 XML 文档的语法结构, 从而使 XML 解析器能够理解和解释文档的内容。

更明确地说, DTD 定义了元素在文档的树形结构中相关联的方式, 并指定了和某些元素一起使用的属性。因此, 它也包含可以在文档中包含的元素类型。有效的 XML 文档就是符合其 DTD 的文档。DTD 以简单文本文件的形式出现, 可以存储在独立的文件中, 也可以嵌入 XML 文件。引用 DTD 的 XML 文档将包含 <!DOCTYPE> 声明, 此声明包含 DTD 声明, 或指定外部 DTD 的位置。

为什么使用 DTD:

XML 提供了以独立方式来共享数据的应用。相互独立的人群可以达成协议, 使用通用的 DTD 来交换数据。应用可以使用标准 DTD 来验证接收的数据是否有效。DTD 可以用于验证自己的数据。在数据交换领域, 出现了无数旨在定义标准 DTD 论坛。DTD 的目的在于定义 XML 文档的合法构建块。它使用一系列合法元素来定义文档结构。

DTD 结构:

DTD 包括许多组件,如 DOCTYPE 声明、元素声明和属性声明。如前所述,<!DOCTYPE> 声明包含了有关 DTD 位置的信息。一般而言,元素在字典中定义为“复合实体的基本、必要或不可少的要素”,但在不同情况下,它代表不同的对象。在数学中,它是一个集的成员;在化学中,它是由具有相同原子数的原子组成的物质;而在数据库语言(如 SQL)中,它表示表中的一个字段。在 XML 中,元素是文档的一个逻辑组件。在 XML 文档中声明的每个元素都必须在 DTD 中具有对应的元素声明,以便在验证时识别身份。同样地,属性在字典中的定义为“用于表示特性、个性或职责的关联对象”。在 XML 中,它表示元素的特性。一个元素可以包含表示该元素的特性的属性。必须按照在 XML 文档中声明元素的方法,在 DTD 中声明属性。总而言之,DTD 的一般结构如下所示。

```
<!DOCTYPE dtd-name [  
<!ELEMENT element-name (element-content type)>  
<!ATTLIST element-name attribute-name attribute-type  
    default-value>  

```

可以在 XML 文件中声明 DTD,也可以将它存储在独立的文件。如果存储在独立的文件,使用.dtd 扩展名进行保存。

声明元素:

在 DTD 中,使用元素声明来声明 XML 元素。元素声明具有以下语法。

```
<!ELEMENT element-name (element-content type)>
```

例如:

```
<!ELEMENT SHOWROOM (TV|LAPTOP)+>
```

空元素:

EMPTY 元素内容类型指定该元素没有子元素或字符数据。将关键字 EMPTY 放在指定位置,可以声明空元素。语法如下:

```
<!ELEMENT element-name EMPTY>
```

例如:

```
<!ELEMENT img EMPTY>
```

空元素可以具有属性。

带有数据的元素:

带有数据的元素是使用它们的数据类型声明的。此数据类型在括号中指定。语法如下:

```
<!ELEMENT element-name (#CDATA)>  
or  
<!ELEMENT element-name (#PCDATA)>  
or  
<!ELEMENT element-name ANY>
```

例如:

```
<!ELEMENT note (#PCDATA)>
```

#CDATA 指元素包含不会通过解析器进行解析的字符数据。#PCDATA 指元素包含要通过解析器进行解析的数据。ANY 指该元素可以包含零个或零个以上任何声明类型的子元素以及字符数据。因此,它是包含所有已声明元素的混合内容的简略表达方式。

带有子元素(序列)的元素:

要定义带有一个或多个子元素的元素,则将子元素的名称放入括号内。语法如下:

```
<!ELEMENT element-name (child-element-name)>
```

或

```
<!ELEMENT element-name (child-element-name,child-element-name,.....)>
```

例如:

```
<!ELEMENT note (to,from,heading,body)>
```

如果在一个序列中声明子元素,并用逗号将它们分开,则这些子元素必须以其在文档中的顺序来显示。子元素可以具有自己的子元素。

声明相同的元素只出现一次:

以下示例声明了子元素 message,它在 note 元素中仅出现一次。语法如下:

```
<!ELEMENT element-name (child-name)>
```

例如:

```
<!ELEMENT note (message)>
```

声明相同的元素至少要出现一次:

以下示例中的“+”符号声明子元素 message 必须至少在 note 元素中出现一次。语法如下:

```
<!ELEMENT element-name (child-name+)>
```

例如:

```
<!ELEMENT note (message+)>
```

声明相同的元素出现零次或多次:

以下示例中的“*”号声明子元素 message 可以在 note 元素中出现零次或多次。语法如下:

```
<!ELEMENT element-name (child-name*)>
```

例如:

```
<!ELEMENT note (message*)>
```

声明相同的元素出现零次或一次:

以下示例中的“?”号声明子元素 `message` 可以在 `note` 元素中出现零次或一次。语法如下:

```
<!ELEMENT element-name (child-name?)>
```

例如:

```
<!ELEMENT note (message?)>
```

声明混合内容:

以下示例声明 `note` 元素必须至少包含一个 `to` 子元素,有且只有一个 `from` 子元素和一个 `header` 子元素,具有零个或一个 `message` 子元素以及其他已解析的字符数据。

例如:

```
<!ELEMENT note (to+,from,header,message*)>
```

组可以是序列或子元素和/或子组的选择。例如:

序列

```
<!-- 元素 A 由单个元素 B 组成。-->
```

```
<!ELEMENT A (B)>
```

```
<!-- 元素 A 由元素 B 加上元素 C 组成。-->
```

```
<!ELEMENT A (B,C)>
```

```
<!-- 元素 A 由包括选择子组的序列组成。-->
```

```
<!ELEMENT A (B,(C|D),E)>
```

选择

```
<!-- 元素 A 由元素 B 或元素 C 组成。-->
```

```
<!ELEMENT A (B|C)>
```

```
<!-- 元素 A 由包括序列子组的选择组成。-->
```

```
<!ELEMENT A (B|C|(D,E))>
```

例 3 演示如何在 DTD 中使用元素。它使用外部 DTD,本章稍后将详细说明外部 DTD。

例 3:

```
<?xml version="1.0" encoding="gb2312"?>
<!DOCTYPE book SYSTEM "Example3.dtd">
<book>
<details>
  <name>xml 使用详解</name>
  <author>成龙来自&country;</name>
```

```
<publication>Mac graw &rights;</publication>
<ptice>&pricenotation;50</price>
</details>
<details>
<name>xml 揭密</name>
<author>Raghu 来自&count;</name>
<publication>Mac graw &rights;</publication>
<ptice>&pricenotation;45</price>
</details>
</book>
```

此文件保存为 Example 3.xml。

```
<?xml version = "1.0" encoding="gb2312"?>
<!ELEMENT book (details+)>
<!ELEMENT details ( name, author, publication, price)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT publication (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ENTITY country "中国">
<!ENTITY count "印度">
<!ENTITY rights "版权所有">
<!ENTITY pricenotation "$">
```

此文件保存为 Example 3.dtd。例 3 的输出结果下图所示。



在例 3 中使用了外部 DTD，DTD 通过以下语法声明。

```
<!DOCTYPE book SYSTEM "Example3.dtd">
```

在此 DTD 中, 声明哪些元素应用于 Example 3.xml。输出结果如上图所示。

```
<!ELEMENT book (details+)>
<!ELEMENT details ( name, author, publication, price)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT publication (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

例 3 中的代码演示了已声明的元素。Book 元素有子元素 details, details 元素有子元素 name、author、publication 和 price。元素中的#PCDATA 表示包含要通过解析器进行解析的数据。

```
<!ENTITY country "中国">
<!ENTITY count "印度">
<!ENTITY rights "版权所有">
<!ENTITY pricenotation "$">
```

例 3 中的代码演示了在 XML 代码中声明的各种实体。每当遇到该实体名称时, 将会替换分号中的字符。

属性声明:

在以下示例中, 将元素 square 定义为空元素, width 属性类型为 CDATA, width 属性的默认值为 0。而后, 把 width 属性值赋为 100。

DTD 示例:

```
<!ELEMENT square EMPTY>
<!ATTLIST square width CDATA "0">
```

XML 示例:

```
<square width="100"/>
```

Default 属性值:

为属性指定一个默认值, 确保即使 XML 文档的作者不提供值, 该属性也将获取一个值。语法如下:

```
<!ATTLIST element-name attribute-name CDATA "default-value">
```

DTD 示例:

```
<!ATTLIST payment type CDATA "check">
```

XML 示例:

```
<payment type="check">
```

Implied 属性值:

如果开发人员不希望强迫作者提供属性, 而且也没有默认值选项, 则他们需要使用 implied 属性。语法如下:

```
<!ATTLIST element-name attribute-name attribute-type #IMPLIED>
```

DTD 示例:

```
<!ATTLIST contact fax CDATA #IMPLIED>
```

XML 示例:

```
<contact fax="222-899877"/>
```

Required 属性值:

如果没有默认值, 但仍然希望在文档中出现此属性, 则使用 required 属性。语法如下:

```
<!ATTLIST element-name attribute-name attribute-type #REQUIRED>
```

DTD 示例:

```
<!ATTLIST person number CDATA #REQUIRED>
```

XML 示例:

```
<person number="6787"/>
```

Fixed 属性值:

如果希望属性具有固定值, 使作者不能更改, 则使用 fixed 属性值。如果作者提供其他值, XML 解析器将返回一个错误。语法如下:

```
<!ATTLIST element-name attribute-name attribute-type #FIXED "value">
```

DTD 示例:

```
<!ATTLIST sender company CDATA #FIXED "Microsoft">
```

XML 示例:

```
<sender company="Microsft"/>
```

Enumerated 属性类型:

希望属性值成为一组固定合法值的一员时, 使用 enumerated 属性值。语法如下:

```
<!ATTLIST element-name attribute-name (eval|eval|..) default-value>
```

DTD 示例:

```
<!ATTLIST payment type (支票|现金) "现金">
```

XML 示例:

```
<payment type="支票">
```

或

```
<payment type="现金">
```

ID 和 IDREF 属性类型:

ID 是标识符类型, 它应该是唯一的。该属性值用于搜索某个元素的特定实例。每个元素都可以具有 ID 类型的一个属性。语法如下:

```
<!-- Topicid 属性提供 Topic 元素的 ID-->
<!ATTLIST Topic Topicid ID #REQUIRED>
.....
<Topic Topicid="Topic4">
  此 Topic 是 XML
</Topic>
```

IDREF 也是标识符类型, 它应只指向一个元素。IDREF 属性可用于引用其他元素中的一个元素, 如以下代码所示。

```
.....
<!-- Prev 和 Next 属性指向另一元素的 ID-->
<!ATTLIST Topic Topicid ID #REQUIRED>
<!ATTLIST Topic Prev IDREF #IMPLIED>
<!ATTLIST Topic Next IDREF #IMPLIED>
.....
<Topic Topicid="Topic4" Prev="Topic3" Next="Topic8">
  <!-- Topics 5-7 丢失-->
  此 Topic 是 XML
</Topic>
.....
```

IDREFS 属性类型:

此属性将多个元素 ID 作为它的值, 各个 IDREF 值之间用空格分开。它用于指向 XML 文档中的相关元素列表, 如以下代码所示。

```
.....
<!ATTLIST Topic Topicid ID #REQUIRED>
<!ATTLIST Topic Prev IDREF #IMPLIED>
<!ATTLIST Topic Next IDREF #IMPLIED>
<!ATTLIST Topic Xrefs IDREFS #IMPLIED>
.....
<Topic Topicid="Topic4" Prev="Topic3" Next="Topic8"
  Xrefs="Topic1 Topic2">
  <!-- Topics 5-7 丢失-->
```



```
    此 Topic 是 XML
</Topic>
.....
```

ENTITY 和 ENTITIES:

这些属性指向以未解析实体(解析器无法处理的实体)形式存在的外部数据。语法如下所示。

```
<!--属性 a 指向单个未解析实体-->
<!ATTLIST A a ENTITY #IMPLIED>

<!--属性 b 指向多个未解析实体-->
<!ATTLIST A b ENTITIES #IMPLIED>
```

NMTOKEN 和 NMTOKENS:

它们用于指定任何有效的一个或多个 XML 名称。将其他组件与元素(如 Java 类或安全算法)关联时,可以使用这些属性。这些属性以单个/多个记号作为值,如以下代码所示。

```
<!ATTLIST Data Authorised_Users NMTOKENS #IMPLIED>
<Data SECURITY="ON" Authorised_Users="Tom">
.....
</Data>
.....
```

DTD 示例:

DTD 定义生成自定义标签的语言的规则。在一个 DTD 中定义每个元素的详细信息、它们的顺序以及每个元素的属性。DTD 分为两种类型。

内部 DTD:

在 XML 文档的 XML 声明后直接编写内部 DTD。应该在 DOCTYPE 定义中编写内部 DTD,如示例所示,这称为包装。语法如下:

```
<!DOCTYPE root-element [element-declarations]>
```

例 4 演示如何使用内部 DTD。

例 4:

```
<?xml version="1.0" encoding="gb2312"?>
<!DOCTYPE movies
[
<!ELEMENT movies (movie+)>
<!ELEMENT movie (title,actor+,rating)>
<!ELEMENT title (#PCDATA)>
```

```
<!ELEMENT actor (#PCDATA)>
<!ELEMENT rating (#PCDATA)>
<!ATTLIST movie type CDATA #IMPLIED>
]
>
<movies>
  <movie type="冒险片">
    <title> 空中监狱 </title>
    <actor> 尼古拉斯 凯奇</actor>
    <rating>家长指引</rating>
  </movie>
  <movie type="恐怖片">
    <title> 幽灵 </title>
    <actor> 黛米 摩尔</actor>
    <actor> 帕特里克 斯韦兹</actor>
    <rating>家长指引</rating>
  </movie>
</movies>
```

对于内部 DTD, DTD 代码和 XML 代码包含在一个文档中。该文件的扩展名为.xml (如 Example 4.xml)。在例 4 中, DOCTYPE 语句表示 Document Type Declaration (文档类型声明), 而方括号中的语句, 则表示 Document Type Definition (文档类型定义)。这可能会产生混淆, 但从上下文来看就很清楚指的是哪种意思。格式良好的 XML 文档必须包含至少一个根元素, 即单个元素声明。另外, 在声明中指定的 DOCTYPE 名称必须与该根元素匹配, 在本例中为 movies。

外部 DTD:

外部 DTD 在文档内容之外, 并带有扩展名.dtd。在 XML 文件的开头添加的 DTD 引用告诉 XML 处理器在哪里查找外部 DTD、关于其作者的信息、DTD 的目的以及使用的语言。外部 DTD 在 XML 文件的开头通过 SYSTEM 关键字引用。将外部 DTD 声明至 XML 文档的语法如下所示。

```
<?xml version="1.0"?>
<!DOCTYPE movies SYSTEM "Example 5.dtd">
```

在本例中, Example 5.dtd 是在文档内容之外的外部 DTD, 并带有必需的扩展名.dtd。

例 5:

```
<?xml version="1.0" encoding="gb2312" ?>
<!DOCTYPE movies SYSTEM "Example5.dtd">
<movies>
  <movie type="冒险片">
    <title> 空中监狱 </title>
    <actor> 尼古拉斯 凯奇</actor>
    <rating>家长指引</rating>
  </movie>
```

```
<movie type="恐怖片">
  <title> 幽灵 </title>
  <actor> 黛米 摩尔</actor>
  <actor> 帕特里克 斯韦兹</actor>
  <rating>家长指引</rating>
</movie>
</movies>
```

此文件保存为 Example 5.xml。外部 DTD 由文件 Example 5.DTD 提供。

```
<?xml version="1.0" encoding="gb2312" ?>
<!ELEMENT movies (movie+)>
<!ELEMENT movie (title,actor+,rating)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT actor (#PCDATA)>
<!ELEMENT rating (#PCDATA)>
<!ATTLIST movie type CDATA #IMPLIED>
```

DTD 中的内部实体声明:

内部实体的内容在 XML 文档中出现。语法如下:

```
<!ENTITY entity-name "entity-value">
```

DTD 示例:

```
<!ENTITY writer "查尔斯·狄更斯">
<!ENTITY copyright "Copyright XML101.">
```

XML 示例:

```
<author>&writer;&copyright;</author>
```

DTD 中的外部实体声明:

外部实体指内容在 XML 文档之外的实体。SYSTEM 关键字用于指定所有在文档之外的实体。语法如下:

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

在以下示例中, XML 处理器将实体引用替换为指定文档的内容。

```
<!ENTITY writer SYSTEM "http://www.xml101.com/entities/entities.xml">
<!ENTITY copyright SYSTEM "http://www.xml101.com/entities/entities.dtd">
```

XML 示例:

```
<author>&writer;&copyright;</author>
```

DTD 中的参数实体:

只在 DTD 文档中出现。

DTD 示例:

```
.....
<!ENTITY % p "a">
<!ELEMENT roster ((%p;)+)>
<!ELEMENT %p; (name,...)>
.....
```

其中 p 是参数实体, a 是 p 的省略值。根据 p 值的不同, DTD 中 roster 子元素也不同。

下面是参数实体的一个具体应用, 两个不同元素的 XML 文件共同关联一个 DTD 文件。

第一个 XML 文件是学生花名册, 保存为 Example 6.xml。

例 6:

```
<?xml version="1.0" encoding="gb2312" ?>
<!DOCTYPE roster SYSTEM "Example 6.dtd" [
<!ENTITY % p "student">
]>
<roster>
  <student ID="s101">
    <name>李华</name>
    <sex>男</sex>
    <birthday>1978.9.12</birthday>
    <score>98</score>
    <skill>Java</skill>
    <skill>Oracle</skill>
    <skill>C Sharp</skill>
    <skill>SQL Server</skill>
  </student>
</roster>
```

上面代码先用内部 DTD 声明, 把参数实体 p 设为 student, 再引用外部 DTD 验证。

第二个 XML 文件是教师花名册, 保存为 Example 7.xml

例 7:

```
<?xml version="1.0" encoding="gb2312" ?>
<!DOCTYPE roster SYSTEM "Example 6.dtd" [
<!ENTITY % p "teacher">
]>
<roster>
  <teacher ID="t101">
    <name>张老师</name>
    <sex>女</sex>
    <birthday>1968.3.1</birthday>
    <skill>Java</skill>
```

```
<skill>Oracle</skill>
<skill>C Sharp</skill>
<skill>SQL Server</skill>
</teacher>
</roster>
```

上面代码先用内部 DTD 声明, 把参数实体 p 设为 teacher, 再引用外部 DTD 验证。

以上两个 XML 文件用同一个 DTD 验证, DTD 代码保存在 Example 6.dtd 中, 见如下实例。

```
<?xml version="1.0" encoding="gb2312"?>
<!ENTITY % p "a">
<!ELEMENT roster ((%p;)+)>
<!ELEMENT %p; (name,sex,birthday,score?,skill+)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT sex (#PCDATA)>
<!ELEMENT birthday (#PCDATA)>
<!ELEMENT score (#PCDATA)>
<!ELEMENT skill (#PCDATA)>
<!ATTLIST %p; ID #REQUIRED>
```

五、命名空间

命名空间是在 XML 文档中可以用作元素或属性名称的名称集合, 它们标识来自特定域 (标准组织、公司、行业) 的名称。命名空间使浏览器可以执行以下操作。

- 组合来自不同源的文档, 并有助于识别元素或属性的源。
- 访问 DTD 或用于验证文档的元素和属性的其他描述。

Uniform Resource Identifier (统一资源标识符, URI) 识别 XML 的命名空间。URI 包括 Uniform Resource Name (统一资源名称, URN) 和 Uniform Resource Locator (统一资源定位符, URL)。URL 包含对 Web 上的某个文档或 HTML 页面的引用。URN 是标识 Internet 资源的全球唯一编号。

例如: 有 3 个名为 batch 的元素。第一个 batch 指 Aptech 培训中心的一批学员, 第二个 batch 指一批产品, 而第三个 batch 指一批游客。可以使用唯一的 URI 标识 batch 元素。第一个 batch 与 Aptech 计算机教育的 URI 关联, 第二个 batch 与茶业 URI 关联, 第三个 batch 与旅游的域 URI 关联。要在文档中使用该元素, 可以使用以下语法。

```
http://www.Aptech_edu.ac.batch
http://www.tea.org.batch
http://www.digicam.org.batch
```

要通过这种方式逐个引用元素十分麻烦。

命名空间的必要性:

人们开始重用和扩展标准的 DTD 时,因为文档交换,所以对于重名的元素,XML 解析器可能出现冲突。如果使用 DTD 中已经存在的元素或属性名称来扩展该 DTD,解析器将无法获知正在使用的是哪一个。命名空间有助于标准化元素和属性,并为它们加上唯一的标志。

命名空间确保元素名称没有冲突,并阐明它们的来源,但它们不确定如何处理元素。XML 解析器必须知道元素的意义以及如何处理它们。

命名空间的语法:

将一个前缀与可以用作命名空间的 URI 关联,如下所示。

`xmlns:[prefix]="[命名空间的 URI]"`

xmlns: 是保留属性。由于 xml 是保留的字符串,它不能用作前缀名称的开头,可以使用 XML 标签中允许的任何其他字符。前缀用作命名空间的别名。例如:

例 8:

```
<?xml version='1.0' encoding="gb2312"?>
<cameras xmlns:digital="http://www.digicam.org"
          xmlns:photo="http://www.photostudio.org">
  <digital:camera prodID="P663" name="傻瓜相机"
    pixels="410000" output_res="640 x 480" int_mem="2 MB"
    price="300.99"/>
  <photo:camera productID="K29B3" name="超级 35 毫米照相机"
    lens="35 毫米" zoom="70 毫米" warranty="1 年" price="99.00"/>
</cameras>
```

在以上 XML 代码中,有关数码相机的信息属于 digital 命名空间,而有关传统相机的信息则属于 photo 命名空间。这样就能够根据两种相机的特定类型验证和处理它们的信息,使数据更灵活和精确。

请注意,尽管前缀(digital 和 photo)只在元素名称中出现,但该元素的属性也属于该元素的命名空间。这意味着 digital:camera 元素上的所有属性也属于 digital 空间。

属性和命名空间:

除非带有前缀,否则属性属于它们的元素的命名空间。

```
.....
<ins:batch ins:type="thirdbatch">夜班</ins:batch>
<ins:batch type="firstbatch">早班</ins:batch>
<ins:batch>下午班</ins:batch>
</ins:batch-list>
.....
```

在以上代码中,两个属性均视为属于相同的命名空间。

```
.....
xmlns="http://www.Aptech_edu.ac"
```

```
xmlns:tea_batch="http://www.tea.org">
<batch-list>
  <batch type="thirdbatch">夜班</batch>
  <batch tea_batch:type="thirdbatch">第三批茶</batch>
  <batch>下午班</batch>
  .....
```

在上面代码中, 属于"http://www.Aptech_edu.ac"命名空间的 batch 元素拥有茶业领域 ("http://www.tea.org"命名空间) 的 tea_batch:type 属性。

可以包括两个名称相同但属于不同命名空间的属性, 如下所示。

```
<batch type="firstbatch" tea_batch:type="firstbatch">第一批茶</batch>
```

命名空间应用:

以下示例演示了如何应用两个命名空间, 分别是 http://www.Aptech_edu.org 和 <http://www.tea.org>。源文件是 Example 9.xml。

例 9:

```
<?xml version="1.0" encoding="gb2312" ?>
<sample xmlns:ins="http://www.Aptech_edu.org"
  xmlns:tea="http://www.tea.org">
  <ins:batch-list>
    <ins:batch>夜间培训批次</ins:batch>
    <ins:batch>早间培训批次</ins:batch>
    <ins:batch>午间培训批次</ins:batch>
    <ins:batch>
      第一批茶<tea:batch>批号 333 </tea:batch>
    </ins:batch>
    <ins:batch>
      第二批茶<tea:batch>批号 222 </tea:batch>
    </ins:batch>
  </ins:batch-list>
</sample>
```

上述代码是一个批次列表, 但来自两个领域。

一个是培训领域, 命名空间是"http://www.Aptech_edu.org", 前缀是 ins。

另一个是茶叶领域, 命名空间是"http://www.tea.org", 前缀是 tea。

Batch-list 是批次列表。

在例 9 中, 除了那些具有 tea 前缀的元素, 所有 3 个 batch 都属于 ins 代表的命名空间。

这样就能够根据 tea 的两个 batch 的特定类型来验证和处理它们的信息, 使数据更灵活和精确。

第十章 用样式表格式化显示

<p>本章目标: 了解 XML 的应用范围 了解 XML 的文档结构 理解格式正规的 XML 文档的特点 熟悉有效的 XML 文档的编写规则 理解命名空间</p> <p>本章重点: 熟悉有效的 XML 文档的编写规则</p> <p>本章难点: 理解命名空间</p>
--

六、 XML的应用范围

人类一直在不断地尝试改进自己的发明, 其中也包括人类最伟大的发明——文字的构成。第一个文本处理系统是用纸笔记录文字。现在, 计算机文本处理器已经取代了手工处理, 它不仅包含原始文档, 还负责设置格式、出版和管理。在这些方便的功能整合到字处理之前, 是由排字工人遵循书面标记说明来完成所有格式编排的。正是利益于这种实践, 人们将“标记”这个词加入到 HTML 和 XML。顾名思义, 标记是指加上记号。文本处理环境(如 XML)中使用了相同的标记过程。本意讲述标记语言的历史和创建 XML 文档的方法。

使用脚本语言或 DHTML 能够以各种方式显示信息。这就要求必须为相同的输出编写不同的代码以供不同的浏览器使用, 因为这些语言不能跨浏览器兼容。

XML (eXtensible Markup Language, 可扩展标记语言) 克服了这些缺点。顾名思义, XML 是可扩展的, 即开发人员可以定义自己的一组标签, 并使其他的人或程序能够理解这些标签。HTML 是单标记语言, 为特定应用设计, 而 XML 则是一系列的标记语言。因此, XML 比 HTML 灵活得多。实际上, 由于 XML 标签表示了数据的逻辑结构, 不同的应用可以通过不同的方式来解释和使用这些标签。Web 上的数据大多是继承的, XML 继承了 SGML 和 HTML 的优点。也就是说, 它不仅继承了 SGML 的特色, 还结合了 HTML 的特色。它采用了 SGML 的主要框架, 有时, 人们也将 XML 称为 SGML 的子集。因此, HTML 是 SGML 的应用, 而 XML 是 SGML 的子集。下图显示了标记语言的层次结构。


```
<html>
  <body>
    <form>
      <select name="cars">
        <option value="volvo">Volvo
        <option value="saab">Saab
        <option value="fiat">Fiat
        <option value="audi">Audi
      </select>
    </form>
  </body>
</html>
```