HTML 基础

HTML是网页设计的基础语言，也就是所谓的网页结构化语言。根据万维网联盟（World Wide Web Consortium，W3C）规范化的设计目标，网页设计师应该从结构化语言（HTML、XHTML和XML）、表现性语言（CSS）和行为控制语言（ECMAScript、Javascript、DOM）这三个方面入手，系统学习网页设计。本章将介绍HTML语言的基本语法和用法。

学习要点

- 了解HTML基本语法
- 了解网页文档基本结构
- 熟悉常用的HTML标签
- 熟悉常用的HTML属性

训练要点

- 能够正确手写网页基本结构
- 熟练使用常用的结构化标签，设计简单的网页文档
- 正确理解和使用文档的类型、名字空间、文档编码等元信息
- 能够区分HTML和XHTML文档规范的相同点和不同点

1.1 认识HTML

HTML是Hyper Text Markup Language的缩写，中文翻译为超文本标记语言。使用HTML标签编写的文档称为HTML文档，目前最新版本是HTML5，且已广泛使用。

HTML从诞生至今，经历了近30年的发展，其中有很多曲折，经历的版本及发布日期如表1-1所示。

表1-1　HTML语言的发展过程

版本	发布日期	说明
超文本标记语言(第一版)	1993年6月	作为互联网工程任务组（IETF）的工作草案，并非标准
HTML 2.0	1995年11月	在（Request For Comments）RFC 2854于2000年6月发布之后，RFC 1866被宣布过时
HTML 3.2	1996年1月14日	W3C推荐标准
HTML 4.0	1997年12月18日	W3C推荐标准
HTML 4.01	1999年12月24日	微小改进，W3C推荐标准
ISO HTML	2000年5月15日	严格的HTML 4.01版本规范是国际标准化组织和国际电工委员会的标准
XHTML 1.0	2000年1月26日	W3C推荐标准，修订后于2002年8月1日重新发布
XHTML 1.1	2001年5月31日	较1.0版本有微小改进
XHTML 2.0草案	没有发布	2009年，W3C停止了XHTML 2.0工作组的工作
HTML5草案	2008年1月	HTML5的规范先是以草案发布，其中经历了漫长的过程
HTML5	2014年10月28日	W3C推荐标准

提 示

从上面的HTML发展列表来看，HTML没有1.0 版本，这主要是因为当时有很多不同的版本。有些人认为Tim Berners-Lee 的版本应该算初版，他的版本中还没有img 元素，也就是说HTML刚开始时仅能够显示文本信息。

1.2 HTML基础知识

HTML实际上是一种规范、一种标准，它通过标签来标记网页中要显示的各个部分，如文字、图形、动画、声音、表格、链接等。网页文件本身仅是一种文本文件，可以在文本文件中添加标签，浏览器能够识别这些标签，知道如何显示其中的内容，如文字如何处理，画面如何安排，图片如何显示等。浏览器按顺序阅读网页文件，然后根据标签解释和显示其标记的内容。

提 示

不同的浏览器对同一标签可能会有不完全相同的解释，因而可能会有不同的显示效果。

1.2.1 HTML概述

HTML是目前互联网上应用最为广泛的语言，也是构成网页文档的主要语言。HTML文档由HTML标签、属性、文本内容组成。

作为一种网页结构标识语言，HTML易学易懂，当用户熟悉使用该语言后，就可以制作内容丰富、结构复杂、美观大方的网页。HTML语言的主要作用如下所述。

- 使用HTML语言标识文本。例如，定义标题文本、段落文本、列表文本、预定义文本等。
- 使用HTML语言建立超链接，通过超链接可以访问互联网上的所有信息，当使用鼠标单击超链接时，会自动跳转到链接页面。
- 使用HTML语言创建列表，把信息有序地组织在一起，以方便浏览。
- 使用HTML语言在网页中显示图像、声音、视频、动画等多媒体信息，把网页设计得更富表现力。
- 使用HTML语言可以制作表格，以方便显示大量数据。
- 使用HTML语言制作表单，允许在网页内输入文本信息，执行其他用户操作，方便信息互动。

1.2.2 HTML文档结构

HTML文档一般都包含两部分，头部区域和主体区域。HTML文档基本结构由<html>、<head>和<body>标签组成，其中，<html>标签标识HTML文档，<head>标签标识头部区域，<body>标签标识主体区域。

一个完整的HTML文档基本结构如下。

```
<!--HTML 文档开始 -->
<html>
    <head>
        <!-- 头部信息区域，如 <title> 标签定义的网页标题 -->
    </head>
    <body>
        <!-- 主体信息区域，包含网页显示的内容 -->
    </body>
</html>
<!--HTML 文档结束 -->
```

在HTML文档中，标签一般都是成对出现的，第一个标签（如<html>）表示标识的起始位置，而第二个标签（如</html>）表示标识的结束位置。<html>标签包含<head>和

<body>两个标签，<head>和<body>标签是并列排列的。

如果把上面的字符代码放置在文本文件中，再另存为test.html，就可以在浏览器中浏览了。由于这个简单的HTML文档还没有包含任何信息，所以在浏览器中是看不到任何显示内容的。

【随堂练习】

<head>标签包含的信息被称为网页的元信息，在网页中是不显示的，而<body>标签包含的信息被称为网页内容，在网页中能够看见。真的是如此吗？下面通过实际操作来答疑解惑。

新建一个文档，另存为index.html，然后在<head>标签和<body>标签中输入一段文字，在浏览器中预览，看这些文本是否都能够显示出来。在文档中手动输入的代码和内容如下。

```
<html>
<head>Hi，大家好，我在头部区域，你能够看见我吗？
</head>
<body>Hi，大家好，我在主体区域，你能够看见我吗？
</body>
</html>
```

预览效果如图1-1所示。

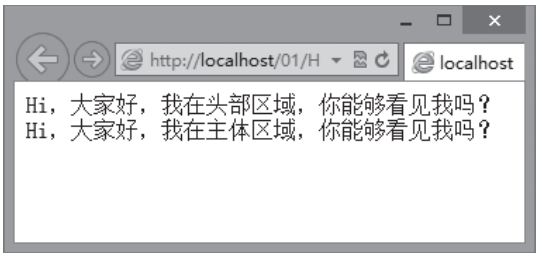


图1-1　测试头部区域和主体区域内容的显示效果

通过上面的示例说明，头部区域的文本只有包含在特定的元信息标签中才能够不被浏览器显示出来。

1.2.3　HTML基本语法

编写HTML文档时，必须遵循HTML语法规范。HTML文档实际上就是一个文本文件，它由标签和文本内容混合组成，但这些标签和文本内容必须遵循一定的语法规则，否则浏览器是无法解析的。

HTML语言的规范条文不多。从逻辑上分析，这些标签包含的内容就表示一类对象，也可以称为网页元素。从形式上分析，这些网页元素通过标签进行分隔，然后表达一定的语义。

- 所有标签都包含在“<”和“>”的起止标识符中。例如，<style>、<head>、<body>和 等。

- 在HTML文档中，绝大多数元素都有起始标签和结束标签，在起始标签和结束标签之间包含的是元素主体。例如，<body>和</body>中间包含的就是网页内容的主体。
- 起始标签包含元素的名称以及可选属性，也就是说元素的名称和属性都必须在起始标签中。结束标签以反斜杠开始，然后附加上元素名称。示例格式如下所示。

```
<tag> 元素主体 </tag>
```

- 元素的属性包含属性名称和属性值两部分，中间通过等号进行连接，多个属性之间通过空格进行分隔。属性与元素名称之间也是通过空格进行分隔。示例代码如下所示。

```
<tag a1="v1" a2="v2" a3="v3" …… an="vn"> 元素主体 </tag>
```

- 少数元素的属性也可能不包含属性值，仅包含一个属性名称。示例格式如下所示。

```
<tag a1 a2 a3 …… an> 元素主体 </tag>
```

- 一般属性值应该包含在引号内，虽然不加引号，浏览器也能够解析，但是读者应该养成良好的操作习惯。
- 属性是可选的，元素包含多少个属性也是不确定的，这主要根据不同的元素而定。不同的元素会包含不同的属性。HTML也为所有元素定义了公共属性，如title、id、class、style等。

虽然大部分标签都是成对出现，但是也有少数标签不是成对的，这些孤立的标签被称为空标签。空标签仅包含起始标签，没有结束标签。示例格式如下所示。

```
<tag>
```

同样，空标签也可以包含很多属性，用来标识特殊效果或者功能。示例操作如下所示。

```
<tag a1="v1" a2="v2" a3="v3" …… an="vn">
```

- 标签可以相互嵌套，形成文档结构。嵌套必须匹配，不能交错嵌套，例如， 是不合法的。合法的嵌套应该是包含或被包含的关系，例如， 或 。
- HTML文档的所有信息必须包含在<html>标签中，所有文档的元信息应包含在<head>子标签中，而HTML传递的信息和网页显示的内容应包含在<body>子标签中。

对于HTML文档来说，除了必须符合基本语法规范外，还必须保证文档结构信息的完整性。完整的文档结构代码如下所示。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> 文档标题 </title>
```

```
</head>
<body></body>
</html>
```

HTML文档主要包括如下内容。

- 必须定义文档的字符编码，一般使用<meta>标签在头部定义，常用字符编码包括中文简体（gb2312）、中文繁体（big5）和通用字符编码（utf-8）。
- 应该设置文档的标题，可以使用<title>标签在头部定义。

HTML文档扩展名为“.htm”或“.html”，保存时必须正确使用扩展名，否则浏览器无法正确地解析。如果要在HTML文档中增加注释性文本，可以在“<!--”和“-->”标识符之间增加，语法格式如下所示。

```
<!-- 单行注释 -->
```

或

```
<!--
多行
注释
--
```

【随堂练习】

浏览器的宽容性也间接增加了用户编写代码的随意性，HTML语法规则虽然不是强制性要求，但是读者应该在学习之初养成良好的编码习惯。

是不是书写的代码不符合语法要求，HTML文档就无法解析？下面通过实际操作来答疑解惑。

新建一个文档，另存为index.html，然后在<body>标签中尝试输入如下的标签结构，在浏览器中预览这些标签是否都能够正确显示。在文档中手动输入的代码和内容如下，预览效果如图1-2所示。

```
<html>
<head>
</head>
<body>
    <div><span> 错误的嵌套结构还能够正确解析吗？ </div></span>
    <p> 段落文本，
    <p> 缺少封闭标签，
    <p> 还能够正确解析吗？
    <!-- 我是注释文本，还能够显示吗 -->
    <!--
    <p> 把标签错放在标签中，我还能够显示吗？ </p>
    -->
</body>
</html>
```

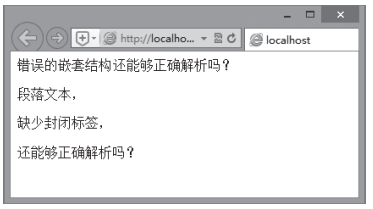


图1-2　浏览器宽容性解析预览效果

以上示例说明，如果错误书写了HTML标签结构，浏览器也能够正确解析它们。虽然这种做法欠妥，有可能纵容用户的随意性，但是浏览器还是以友好的姿态努力解析所有网页。在此，读者不应该以本示例作为榜样，而应该把它视为反面案例。

需要注意的是，此项操作中任何包含在“<!--”和“-->”标签之间的信息都被无情地过滤掉了。

1.2.4　HTML标签

标签是HTML语言最基本的单位，每个标签基本都有“<”和“/>”，大部分标签都是成对出现的。当然也有个别孤标签，即仅有一个起始标签。

由于HTML定义的标签很多，下面就常用标签进行说明。随着读者不断深入学习，相信能够完全掌握HTML所有标签的用法和使用技巧，更详细的说明请参阅“HTML参考手册”。

1. 文档结构标签

此类标签主要用来标识文档的基本结构，主要标签说明如下。

- <html>…</html>：标识HTML文档的起始和终止。
- <head>…</head>：标识HTML文档的头部区域。
- <body>…</body>：标识HTML文档的主体区域。

【随堂练习】

新建一个文本文件，熟练输入下面的代码，然后另存为test.html，并在浏览器中测试。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> 无标题文档 </title>
</head>
<body> 网页正文写在这里……
</body>
</html>
```

2. 文本格式标签

此类标签主要用来标识文本区块，并附带一定的显示格式。主要标签说明如下。

- <title>…</title>：标识网页标题。
- <hi>…</hi>：标识标题文本，其中i可以是1、2、3、4、5、6，分别表示一级、二

级、三级、四级、五级、六级标题。

- … ：标识段落文本。
- <pre>…</pre>：标识预定义文本。
- <blockquote>…</blockquote>：标识引用文本。

【随堂练习】

新建一个文本文件，熟练输入下面的代码，分别使用<h1>和 标签标识网页标题和段落文本，然后另存为test.html，并在浏览器中测试。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> 示例代码 </title>
</head>
<body>
<h1> 文本格式标签 </h1>
<p>&lt;p&gt; 标签标识段落文本 </p>
</body>
</html>
```

3. 字符格式标签

此类标签主要用来标识部分文本字符的语义，很多字符标签可以呈现一定的显示效果。例如，加粗显示、斜体显示或者下划线显示等。主要标签说明如下。

- …：标识强调文本，以加粗效果显示。
- …：标识引用文本，以斜体效果显示。
- <blink>…</blink>：标识闪烁文本，以闪烁效果显示。IE浏览器不支持该标签。
- <big>…</big>：标识放大文本，以放大效果显示。
- <small>…</small>：标识缩小文本，以缩小效果显示。
- […]：标识上标文本，以上标效果显示。
- _…：标识下标文本，以下标效果显示。
- <cite>…</cite>：标识引用文本，以引用效果显示。

【随堂练习】

新建一个文本文件，熟练输入下面的代码，分别使用各种字符格式标签显示一个数学方程式的解法，然后另存为test.html，并在浏览器中测试，显示效果如图1-3所示。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> 示例代码 </title>
</head>
<body>
<p> 例如，针对下面这个一元二次方程：</p>
<p><i>x</i><sup>2</sup>-<b>5</b><i>x</i>+<b>4</b>=0</p>
```

```
<p>我们使用 <big><b>分解因式法</b></big> 来演示解题思路如下：</p>
<p><small>由：</small>(<i>x</i>-1)(<i>x</i>-4)=0</p>
<p><small>得：</small><br /><i>x</i><sub>1</sub>=1<br />
    <i>x</i><sub>2</sub>=4</p>
</body>
</html>
```

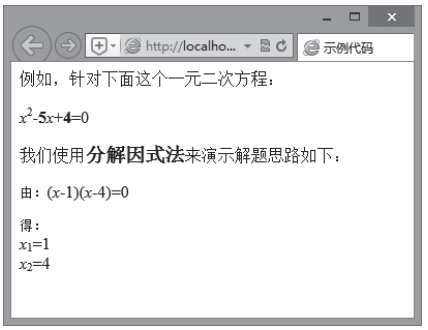


图1-3　字符格式标签示例显示效果

4. 列表标签

在HTML文档中，列表结构可以分为两种类型，有序列表和无序列表。无序列表使用项目符号来标识列表，而有序列表则使用编号来标识列表的项目顺序。主要标签说明如下所述。

- …：标识无序列表。
- …：标识有序列表。
- …：标识列表项目。

【随堂练习】

新建一个文本文件，熟练输入下面的代码，使用无序列表分别显示了一元二次方程求解的四种方法，然后另存为test.html，并在浏览器中测试，显示效果如图1-4所示。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>示例代码</title>
</head>
<body>
<h1>解一元二次方程</h1>
<p>一元二次方程求解有四种方法：</p>
<ul>
    <li>直接开平方法 </li>
    <li>配方法 </li>
    <li>公式法 </li>
    <li>分解因式法</li>
</ul>
</body>
</html>
```

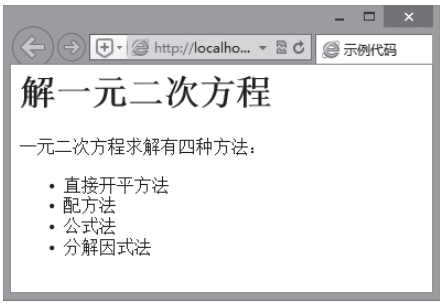


图1-4　无序列表标签示例显示效果

定义列表是一种特殊的结构，它包括词条和解释两块内容。主要标签说明如下所述。

- <dl>…</dl>：标识定义列表。
- <dt>…</dt>：标识词条。
- <dd>…</dd>：标识解释。

【随堂练习】

新建一个文本文件，熟练输入下面的代码，使用定义列表显示两个成语的解释，然后另存为test.html，并在浏览器中测试，显示效果如图1-5所示。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> 示例代码 </title>
</head>
<body>
<h1> 成语词条列表 </h1>
<dl>
    <dt> 知无不言，言无不尽 </dt>
    <dd> 知道的就说，要说就毫无保留。</dd>
    <dt> 智者千虑，必有一失 </dt>
    <dd> 不管多聪明的人，在很多次的考虑中，也一定会出现个别错误。</dd>
</dl>
</body>
</html>
```

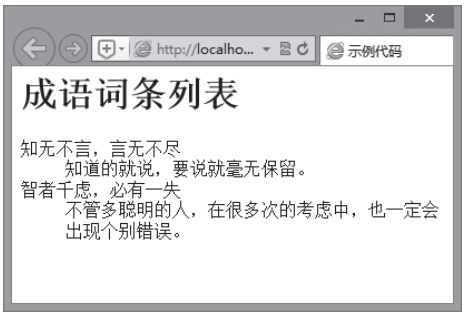


图1-5　定义列表标签示例显示效果

5. 链接标签

此标签可以实现把多个网页联系在一起。主要标签说明如下所述。

- <a>…：标识超链接。

【随堂练习】

新建一个文本文件，熟练输入下面的代码，使用<a>标签定义一个超链接，实现单击该超链接可以跳转到百度首页，然后另存为test.html，并在浏览器中测试，显示效果如图1-6所示。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> 示例代码 </title>
</head>
<body>
<a href="http://www.baidu.com/"> 去百度搜索 </a>
</body>
</html>
```

图1-6　超链接标签示例显示效果

<a>标签还可以定义锚点。锚点是一类特殊的超链接，它可以定位到网页中某个具体的位置。

【随堂练习】

新建一个文本文件，熟练输入下面的代码，使用<a>标签定义一个锚点，实现单击超链接文本就可以跳转到网页的底部，然后另存为test1.html，并在浏览器中测试。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> 示例代码 </title>
</head>
<body>
<a href="#btm"> 跳转到底部 </a>
<div id="box" style="height:2000px; border:solid 1px red;"> 撑开浏览器滚动条 </div>
<span id="btm"> 底部锚点位置 </span>
```

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

6. 多媒体标签

此类标签主要用于引入外部多媒体文件，并进行显示。主要标签说明如下。

- <img/>：嵌入图像。
- <embed>…</embed>：嵌入多媒体。
- <object>…</object>：嵌入对象。

7. 表格标签

此类标签是用来组织和管理数据的，主要标签说明如下所述。

- <table>…</table>：定义表格结构。
- <caption>…</caption>：定义表格标题。
- <th>…</th>：定义表头。
- <tr>…</tr>：定义表格行。
- <td>…</td>：定义表格单元格。

【随堂练习】

新建一个文本文件，熟练输入下面的代码，使用表格标签定义并显示5行3列的数据集，然后另存为test.html，并在浏览器中测试，显示效果如图1-7所示。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>示例代码</title>
</head>
<body>
<table summary="ASCII是英文American Standard Code for Information Interchange的缩写。ASCII编码是通用的计算机编码标准。因为计算机只能接受数字信息，ASCII编码将字符转换为数字来表示，以便计算机能够接受和处理。">
    <caption>ASCII字符集（节选）</caption>
    <tr>
        <th>十进制</th>
        <th>十六进制</th>
        <th>字符</th>
    </tr>
    <tr>
        <td>9</td>
        <td>9</td>
        <td>TAB(制表符)</td>
    </tr>
    <tr>
        <td>10</td>
```

```
            <td>A</td>
            <td> 换行 </td>
        </tr>
        <tr>
            <td>13</td>
            <td>D</td>
            <td> 回车 </td>
        </tr>
        <tr>
            <td>32</td>
            <td>20</td>
            <td> 空格 </td>
        </tr>
    </table>
    </body>
    </html>
```

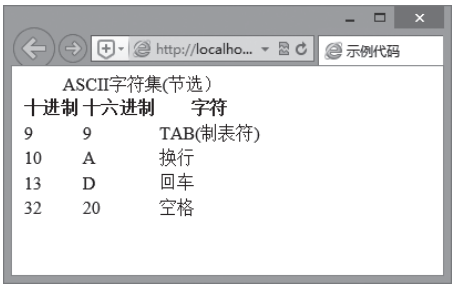


图1-7　表格标签示例显示效果

8. 表单标签

此类标签主要用来制作交互式表单，主要标签说明如下所述。

- <form>…</form>：定义表单结构。
- <input>：定义文本域、按钮和复选框。
- <textarea>…</textarea>：定义多行文本域。
- <select>…</select>：定义下拉列表。
- <option>…</option>：定义下拉列表中的选择项目。

【随堂练习】

新建一个文本文件，熟练输入下面的代码，使用表单标签分别定义单行文本域、密码域、多行文本域、复选框、单选按钮、下拉菜单和提交按钮的复杂表单，然后另存为test.html，并在浏览器中测试，显示效果如图1-8所示。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> 示例代码 </title>
```

```
    </head>
    <body>
    <form id="form1" name="form1" method="post" action="">
        <p>单行文本域：<input type="text" name="textfield" id="textfield"
/></p>
        <p>密码域：<input type="password" name="passwordfield"
id="passwordfield" /></p>
         <p>多行文本域：<textarea name="textareafield" id="textareafield">
</textarea></p>
        <p>复选框：复选框 1<input name="checkbox1" type="checkbox"
value="" />
           复选框 2<input name="checkbox2" type="checkbox" value="" />
        </p>
        <p>单选按钮：
            <input name="radio1" type="radio" value="" />按钮 1
            <input name="radio2" type="radio" value="" />按钮 2</p>
        <p>下拉菜单：
            <select name="selectlist">
                <option value="1">选项 1</option>
                <option value="2">选项 2</option>
                <option value="3">选项 3</option>
            </select>
        </p>
        <p><input type="submit" name="button" id="button" value="提交"
/></p>
    </form>
    </body>
    </html>
```

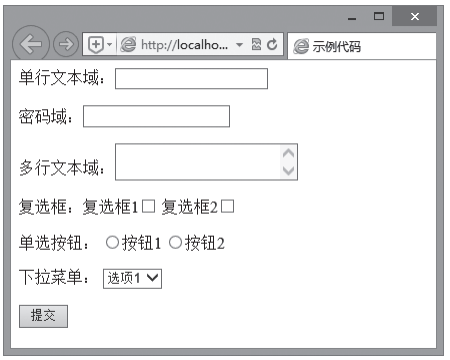


图1-8　表单标签示例显示效果

1.2.5　HTML属性

HTML元素包含的属性众多，这里无法列出所有元素的全部属性，如果读者想查阅每一种标签的全部属性，建议参考“HTML参考手册”。

下面仅对大部分常用标签的公共属性进行分析。公共属性大致可分为基本属性、语言属性、键盘属性、内容属性和延伸属性等类型。

1. 基本属性

基本属性主要包括下面三个，这三个基本属性为大部分元素所拥有。

- class：定义类规则或样式规则。
- id：定义元素的唯一标识。
- style：定义元素的样式声明。

下面这些元素不拥有基本属性。

- html、head：文档和头部基本结构。
- title：网页标题。
- base：网页基准信息。
- meta：网页元信息。
- param：元素参数信息。
- script、style：网页的脚本和样式。

这些元素一般位于文档的头部区域，用来标识网页元信息。

2. 语言属性

语言属性主要是用来定义元素的语言类型，包括两个属性。

- lang：定义元素的语言代码或编码。
- dir：定义文本的方向，包括ltr和rtl取值，分别表示从左向右和从右向左。

下面这些元素不拥有语言属性。

- frameset、frame、iframe：网页框架结构。
- br：换行标识。
- hr：结构装饰线。
- base：网页基准信息。
- param：元素参数信息。
- script：网页的脚本。

【随堂练习】

新建一个网页基本结构，为<html>标签和<body>标签添加属性。为网页代码定义中文简体的语言，字符对齐方式为从左向右，同时为body定义美式英语，然后另存为test.html，代码如下所示。

```
    <html xmlns="http://www.w3.org/1999/xhtml" dir="ltr" xml:lang="zh-CN">
    <body id="myid" lang="en-us">
```

3. 键盘属性

键盘属性定义元素的键盘访问方法，包括两个属性。

- accesskey：定义访问某元素的键盘快捷键。
- tabindex：定义元素的Tab键索引编号。

使用accesskey属性可以使用快捷键“Alt+字母”访问指定URL，但是有的浏览器不能很好地支持此功能。在IE中该属性仅能激活超链接，还需要按Enter键确定，而在FF中没有反应，代码如下所示。

```
<a href="http://www.baidu.com/" accesskey="a">按住 Alt 键，单击 A 键可以链接到百度首页 </a>
```

一般在导航菜单中经常设置快捷键。

tabindex属性用来定义元素的Tab键的访问顺序，可以使用Tab键遍历页面中的所有链接和表单元素。遍历时会按照tabindex的大小决定顺序，当遍历到某个链接时，按Tab键即可打开链接页面。代码如下所示。

```
<a href="#" tabindex="1">Tab 1</a>
<a href="#" tabindex="3">Tab 3</a>
<a href="#" tabindex="2">Tab 2</a>
```

4. 内容属性

内容属性定义元素包含内容的附加信息，这些信息对于元素来说具有重要的补充作用，也避免了元素本身包含的信息不全而被误解。内容属性包括以下五个。

- alt：定义元素的替换文本。
- title：定义元素的提示文本。
- longdesc：定义元素包含内容的大段描述信息。
- cite：定义元素包含内容的引用信息。
- datetime：定义元素包含内容的日期和时间。

alt和title是两个常用的属性，分别定义元素的替换文本和提示文本，很多设计师习惯混用这两个属性，没有刻意去区分它们的语义性。实际上，除了IE浏览器，其他标准浏览器都不会支持它们的混用，这是由于IE浏览器的纵容，才导致很多设计师误以为alt属性就是设置提示文本的。代码如下所示。

```
<a href="URL" title=" 提示文本 "> 超链接 </a>
<img src="URL" alt=" 替换文本 " title=" 提示文本 " />
```

替换文本（Alternate Text）并不是用作工具提示（Tool Tip）的，更确切地说，它并不是为图像提供额外说明信息的，title属性才是负责为元素提供额外说明信息的。

当图像无法显示时，必须准备替换的文本来替换无法显示的图像，这对于图像和图像热点是必须的，因此alt属性只能用在img、area和input元素中（包括applet元素）。对于input元素，alt属性用来替换提交按钮的图片。代码如下所示。

```
<input type="image" src="URL" alt=" 替换文本 " />.
```

为什么要设置替换文本呢？这主要是因为浏览器被禁止显示、不支持或无法下载图像时，通过替换文本给那些不能看到图像的浏览者提供文本说明，这是一个很重要的预

防和补救措施。另外，还应该考虑到网页对于视觉障碍者，或者使用其他用户代理的影响，如对屏幕阅读器、打印机等代理设备的影响。当然，从语义角度考虑，替换文本应该提供图像的简明信息，并保证在上下文中有意义，而对于那些修饰性的图片可以使用空值（alt=""）。

title属性为元素提供提示性的参考信息，这些信息是一些额外的说明，具有非本质性，因此该属性也不是一个必须设置的属性。当鼠标指针移到元素上面时即可看到这些提示信息。title属性不能够用在下面这些元素上。

- html、head：文档和头部基本结构。
- title：网页标题。
- base、basefont：网页基准信息。
- meta：网页元信息。
- param：元素参数信息。
- script：网页的脚本和样式。

相对而言，title属性可以比alt属性设置更长的文本，不过有些浏览器可能会限制提示文本的长度，但是不管怎么规定，提示文本一定要简明扼要，并用在恰当的地方，而不是所有元素身上都要定义一个提示文本，那样反而显得画蛇添足了。提示文本一般多用在超链接上，必须对图标按钮提供提示性的说明信息，否则用户会不明白这些图标按钮的作用。

如果要为元素定义更长的描述信息，则应该使用longdesc属性。longdesc属性可以用来提供链接到一个包含图片描述信息的单独页面或者长段描述信息的位置。示例代码如下所示。

```
<img src="URL" alt="人物照" title="张三于2017-5-1在中国馆留念"
longdesc="这是张三于2017年5月1日在中国馆前的留影，当时天很热" />
```

或

```
<img src="URL" alt="替换文本" longdesc="详细描述图像的网页.html" />
```

这种方法意味着从当前页面链接到另一个页面，由此可能会造成理解上的困难。另外，浏览器对于longdesc属性的支持也不一致，所以尽量避免使用。如果感觉图片的长描述信息很有用，不妨考虑把这些信息简单地显示在同一个文档里，而不是链接到其他页面或者隐藏起来，这样能够保证每个人都可以阅读到。

cite一般用来定义引用信息的URL。例如，有一段文字引自http://www.baidu.com/csslayout/ index.htm，代码如下所示。

```
<blockquote cite="http://www.baidu.com/csslayout/index.htm">
    <p>CSS的精髓是布局，而不是样式，布局需要缜密的结构分析和设计</p>
</blockquote>
```

datetime属性定义包含文本的时间，这个时间可以表示信息的发布时间，也可以是更新时间，代码如下所示。

```
<ins datetime="2017-5-1 8:0:0">2017 年上海世博园 </ins>
```

1.3 XHTML基础

XHTML是XML语言的一个应用，它遵守XML语言的规范和要求。从技术角度分析，这些语法规则是由XML规范定义的。XML文档必须遵守的规则使得生成工具解析文档变得更容易，这些规则也使得XML更容易处理。

用过HTML的用户对于XHTML中的一些规则应该比较熟悉，为了兼容数以万计的现存网页和不同浏览器，XHTML语言兼顾了HTML语法规则。XHTML文档与HTML文档没有太大区别，只是添加了XML语言的基本规范和要求。

1.3.1 XHTML概述

HTML是一种基本的网页设计语言，XHTML是一个基于XML语言的标识语言，与HTML相似，只有一些小的但重要的区别，XHTML就是一个扮演着类似HTML角色的XML。从本质上分析XHTML是一个过渡技术，它结合了XML的强大功能与HTML的简单特性。如果读者掌握了HTML语言的基本用法，以及能熟练使用HTML的标签和属性之后，只需要稍稍阅读本节知识，就可以轻松构建XHTML文档。

2000年底，国际W3C组织公布了XHTML 1.0版本。XHTML 1.0是一种在HTML 4.0的基础上优化和改进的新语言，目的是基于XML的应用。XHTML是一种增强了的HTML，它的可扩展性和灵活性将适应未来网络应用的更多需求。

与HTML相比，XHTML具有如下特点。

- XHTML是要解决HTML语言所存在的严重制约其发展的问题。HTML发展至今存在三个主要缺点：第一，不能适应现在越来越多的网络设备和应用的需要，如手机、Pad（平板电脑）、信息家电都不能直接显示HTML；第二，由于HTML代码不规范且臃肿，浏览器需要足够智能和庞大才能够正确显示HTML；第三，如果页面要改变显示，就必须重新编写HTML。因此HTML需要不断改进才能解决这个问题，于是W3C又制定了XHTML，XHTML是HTML向XML过渡的一个桥梁。
- XML是网页发展的趋势，所以人们希望加入到XML的潮流中。XHTML是替代HTML 4.0语言的标准，使用XHTML 1.0只要遵守一些简单规则就可以设计出既适合XML规范，又适合当前大部分HTML浏览器的页面。也就是说，用户可以直接使用XML，而不需要使用必须支持XML的浏览器。
- XHTML结构非常严谨。HTML结构糟糕得让人震惊，早期的浏览器接受私有的HTML标签，当用户在页面设计完毕后，必须使用各种浏览器来检测页面是否兼容。检测过程中往往会有许多莫名其妙的差异，为此不得不修改设计以适应不同的浏览器。
- XHTML能与其他基于XML的标记语言、应用程序以及协议进行良好的交互工作。
- XHTML是Web标准家族的一部分，能很好地用在无线设备等其他用户代理上。

- 在网站设计方面，XHTML可以帮助改掉代码编写的恶习，帮助用户养成用标记校验来测试页面工作的良好习惯。

1.3.2 XHTML文档结构

一个完整的XHTML文档结构如下所示。

```
<!--[XHTML 文档基本框架 ]-->
<!-- 定义 XHTML 文档类型 -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!--XHTML 文档根元素，其中 xmlns 属性声明文档命名空间 -->
<html xmlns="http://www.w3.org/1999/xhtml">
<!-- 头部信息结构元素 -->
<head>
<!-- 设置文档字符编码 -->
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<!-- 设置文档标题 -->
<title> 无标题文档 </title>
</head>
<!-- 主体内容结构元素 -->
<body>
</body>
</html>
```

XHTML代码不排斥HTML规则，在结构上也基本相似，如果仔细比较，会发现有两个不同点。

1. 定义文档类型

在XHTML文档第一行新增了<!DOCTYPE>元素，该元素用来定义文档类型。DOCTYPE是Document Type（文档类型）的简写，它是设置XHTML文档的版本。使用时应注意该元素的名称和属性必须大写。

DTD（如xhtml1-transitional.dtd）表示文档类型定义，里面包含了文档的规则，网页浏览器会根据预定义的DTD来解析页面元素，并把这些元素组织的页面显示出来。要建立符合网页标准的文档，DOCTYPE声明是必不可少的关键组成部分，除非XHTML确定了一个正确的DOCTYPE，否则页面内的元素和CSS不能正确生效。

2. 声明命名空间

在XHTML文档根元素中必须使用xmlns属性声明文档的命名空间。xmlns是XML Name Space的缩写，中文翻译为命名空间，也有人翻译为名字空间或名称空间。命名空间是收集元素类型和属性名字的一个详细DTD，它允许通过一个URL地址指向来识别命名空间。

XHTML是HTML向XML过渡的标记语言，它需要符合XML规则，因此也需要定义

命名空间。因为XHTML 1.0还不允许用户自定义元素，因此它的命名空间都相同，例如“http://www.w3.org/1999/xhtml”。这就是为什么每个XHTML文档的xmlns值都相同的缘故。

1.3.3 XHTML基本语法

XHTML是根据XML语法简化而来的，因此它遵循XML的文档规范，同时XHTML还大量继承HTML的语言语法规范，因此与HTML语言非常相似，不过它对代码的要求更加严谨。遵循这些要求，对于培养良好的XHTML代码书写习惯是非常重要的。

- 所有的标记都必须有一个相应的结束标记。

在HTML中，用户可以打开许多标签，例如， 和
可能不一定写对应的 和
来关闭它们，但这在XHTML中是不合法的。XHTML要求有严谨的结构，所有标签必须关闭。如果是单独不成对的标签，在标签最后也应该加一个"/"来关闭它。示例代码如下所示。

```
    <br /><img height="80" alt=" 网页设计师 " src="../images/logo_
w3cn_200x80.gif" width="200" />
```

- 所有标签的元素和属性的名字都必须小写。

与HTML不同，XHTML对大小写是敏感的，例如<title>和<TITLE>是不同的标签。但是，XHTML要求所有的标签和属性的名字都必须使用小写，例如<BODY>必须写成<body>。大小写夹杂也是不被认可的，通常Dreamweaver自动生成的属性名字onMouseOver必须修改成onmouseover。

所有XHTML标记都必须合理嵌套，因为XHTML有严谨的结构要求，因此所有的嵌套都必须按顺序进行，示例代码如下所示。

```
    <p><b></p>/b>
```

必须修改为如下样式。

```
    <p><b></b>/p>
```

也就是说，一层一层的嵌套必须是严格对称的。

- 所有的属性值必须用引号（""）括起来。

在HTML中，用户可以不给属性值加引号，但是在XHTML中，属性值必须加引号。示例代码如下所示。

```
    <height=80>
```

必须修改为如下样式。

```
    <height="80">
```

特殊情况下，需要在属性值里使用双引号，可以用单引号（' '）替换，单引号也可以使用转义符“'”，示例代码如下所示。

```
<alt="say&apos;hello&apos;">
```

- <、>、&等特殊符号必须用编码表示。

任何小于号（<）若不是标签的一部分，都必须被编码为“<”；任何大于号（>）若不是标签的一部分，都必须被编码为“>”；任何与号（&）若不是实体的一部分，都必须被编码为“&”。

- 给所有属性赋一个值。

XHTML规定所有属性都必须有一个值，没有值的就重复本身。示例代码如下所示。

```
<td nowrap>
<input type="checkbox" name="shirt" value="medium" checked>
```

必须修改为如下样式。

```
<td nowrap="nowrap">
<input type="checkbox" name="shirt" value="medium" checked="checked">
```

不要在注释内容中使用“- -”。

“--”只能发生在XHTML注释的开头和结束，也就是说，在内容中它们不再有效。示例代码如下所示。

```
<!-- 这里是注释 ---------- 这里是注释 -->
```

可以用等号或者空格替换内部的虚线。

```
<!-- 这里是注释 = = = = = = = = = = = = 这里是注释 -->
```

- 在文档的开头必须定义文档类型。
- 在根元素中应声明命名空间，即设置xmlns属性。
- XHTML规范废除了name属性，而使用id属性作为统一的名称。在IE 4.0及以下版本中应保留name属性，使用时可以同时使用id和name属性。

上面列举的几点是XHTML最基本的语法要求，习惯于HTML的读者应克服代码书写中的随意性，好的习惯会影响一生。

1.3.4 XHTML文档类型分类

XHTML 1.0支持三种DTD声明：过渡型（Transitional）、严格型（Strict）和框架型（Frameset）。

1. 过渡型

这类文档类型对于标签和属性的语法要求不是很严格，允许在页面中使用HTML 4.01的标签（符合XHTML语法标准）。过渡型DTD语句声明如下所示。

```
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w1.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

2. 严格型

这类文档类型对于文档内的代码要求比较严格，不允许使用任何表现层的标签和属性。严格型DTD语句声明如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w1.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

在严格型文档类型中，以下元素将不被支持。

- center：居中（属于表现层）。
- font：字体样式，如大小、颜色和样式（属于表现层）。
- strike：删除线（属于表现层）。
- s：删除线（属于表现层）。
- u：文本下划线（属于表现层）。
- iframe：嵌入式框架窗口（专用于框架文档类型或过渡型文档）。
- isindex：提示用户输入单行文本（与input元素语义重复）。
- dir：定义目录列表（与dl元素语义重复）。
- menu：定义菜单列表（与ul元素语义重复）。
- basefont：定义文档默认字体属性（属于表现层）。
- applet：定义插件（与object元素语义重复）。

在严格型文档类型中，以下属性将不被支持。

- align（支持table包含的相关元素：tr、td、th、col、colgroup、thead、tbody、tfoot）。
- language。
- background。
- bgcolor。
- border（table元素支持）。
- height（img和object元素支持）。
- hspace。
- name（在HTML 4.01 Strict中支持，在XHTML 1.0 Strict中的form和img元素不支持）。
- noshade。
- nowrap。
- target。
- text、link、vlink和alink。
- vspace。
- width（img、object、table、col和colgroup元素支持）。

3. 框架型

这是一种专门针对框架页面所使用的DTD，当页面中含有框架元素时，就应该采用这种DTD。框架型DTD语句声明如下所示。

```
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN"
“http://www.w1.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

使用严格型DTD来制作页面是最理想的方式，但是对于没有深入了解Web标准的网页设计者来说，适合使用过渡型DTD。因为过渡型DTD允许使用表现层元素和属性，适合大多数初级网页制作人员使用。

对于大多数标准网页设计师来说，过渡型DTD（XHTML 1.0 Transitional）是比较理想的选择。因为这种DTD允许使用描述性的元素和属性，也比较容易通过W3C的代码校验。

1.3.5 XHTML文档类型详解

在XHTML文档中，DOCTYPE是一个必要元素，它决定了网页文档的显示规则。网页中通过在首行代码中定义文档类型，来指定页面所使用的HTML的版本类型。在构建符合标准的网页中，只有确定正确的DOCTYPE，HTML文档的结构和样式才能被正常解析和呈现。

实际上，DTD 是一套关于标签的语法规则。DTD文件是一个ASCII码文本文件，后缀名为“.dtd”。如果利用DOCTYPE声明中的URL可以访问指定类型的DTD详细信息。例如，对于XHTML 1.0过渡型DTD的URL为“http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”，在浏览器地址栏中输入该地址即可打开XHTML 1.0过渡型DTD文档，如图1-9所示。

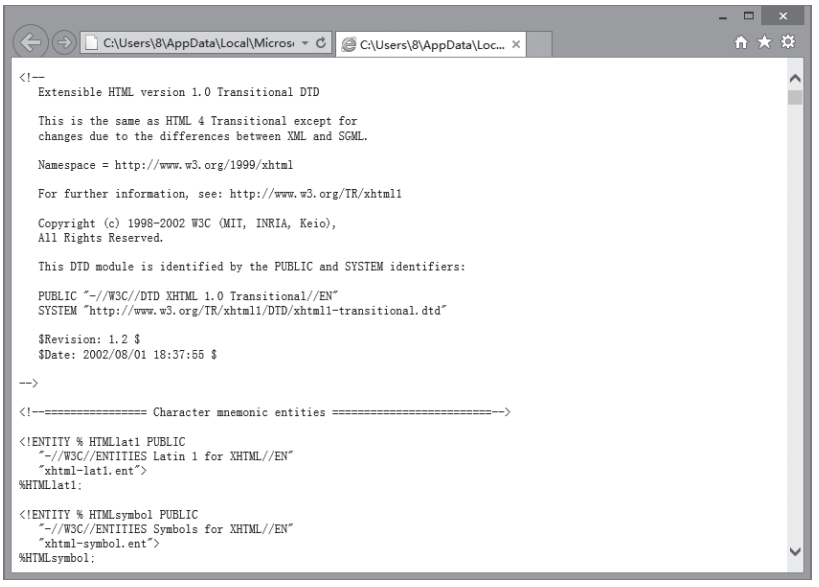


图1-9 XHTML 1.0过渡型DTD文档

一个DTD文档包含元素的定义规则，元素间关系的定义规则，元素可使用的属性、实体或符号规则。这些规则用于标记Web文档的内容。此外还包括了其他规则，它们规定了哪些标签能出现在其他标签中。文档类型不同，它们对应的DTD也不相同。

例如，下面是从XHTML 1.0过渡型DTD文档中截取的有关image元素定义的相关规则。

```
<!--================== Images ================================-->
<!--
   To avoid accessibility problems for people who aren't
   able to see the image, you should provide a text
   description using the alt and longdesc attributes.
   In addition, avoid the use of server-side image maps.
-->
<!ELEMENT img EMPTY>
<!ATTLIST img
  %attrs;
  src          %URI;              #REQUIRED
  alt          %Text;             #REQUIRED
  name         NMTOKEN            #IMPLIED
  longdesc     %URI;              #IMPLIED
  height       %Length;           #IMPLIED
  width        %Length;           #IMPLIED
  usemap       %URI;              #IMPLIED
  ismap        (ismap)            #IMPLIED
  align        %ImgAlign;         #IMPLIED
  border       %Length;           #IMPLIED
  hspace       %Pixels;           #IMPLIED
  vspace       %Pixels;           #IMPLIED
>
```

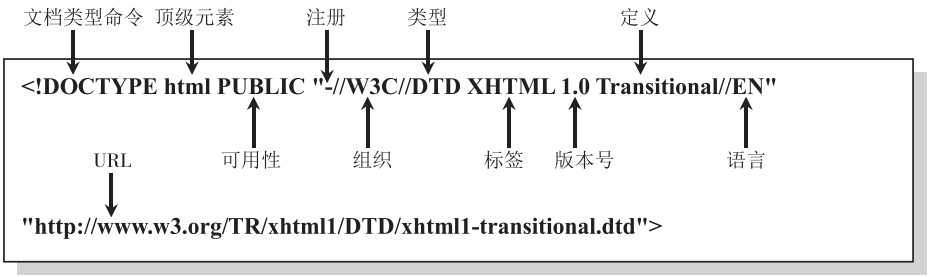
“<!--”和“-->”表示注释，与HTML文档中的注释语句相同，然后使用“<!ELEMENT”命令定义一个image元素，后面的关键字“EMPTY”表示该元素可以为空，不包含其他元素。

使用“<!ATTLIST”命令定义属性，后面跟随的img表示被定义的元素。

“%attrs;”表示属性列表。在跟随的属性列表中，第一列为属性的名称，第二列以“%”标识符定义属性的数据类型。例如，URI表示文件的地址，Text表示字符串文本，Length表示长度，Pixels表示像素等。第三列表示属性的默认值类型，其中“#REQUIRED”表示属性值是必须的；“#IMPLIED”表示属性值不是必须的；“#FIXED value”表示属性值是固定的。想了解有关该文档更详细的规则可以查阅相关资料。

由于不同的浏览器对HTML和CSS语言的解释效果并不完全相同，也就是说不同浏览器的解析规则是不同的。如果页面中没有显示声明DOCTYPE，则不同浏览器会自动采用各自默认的DOCTYPE规则来解析文档中的各种标签和CSS样式码。因此，从浏览器兼容性来考虑，声明DOCTYPE是必须的。

DOCTYPE声明必须放在（X）HTML文档的顶部，在文档类型声明语句的上面不能够包含任何HTML代码，也不能包含HTML注释标签。DOCTYPE声明语句的说明如图1-10所示。



DOCTYPE声明中各个部分说明如下所述。

- 顶级元素：指定DTD中声明的顶级元素类型，这与声明的SGML文档类型相对应。HTML文档默认顶级元素为html。
- 可用性：指定正式公开标识符（FPI）是可公开访问的对象（PUBLIC）还是系统资源（SYSTEM）。默认为PUBLIC，系统资源包括本地文件或URL。
- 注册：指定组织是否由国际标准化组织（ISO）注册。“+”表示组织名称已注册，这里默认选项“-”，表示组织名称未注册。W3C是属于非注册的ISO组织，所以显示为“-”。
- 组织：指定在“!DOCTYPE”声明引用的DTD的创建和维护的团体或组织的名称。HTML语言规范的创建和维护组织为W3C。
- 类型：指定公开文本的类，即所引用的对象类型。HTML默认为DTD。
- 标签：指定公开文本的描述，即对所引用的公开文本的唯一描述性名称，后面可附带版本号。HTML默认为HTML，XHTML默认为XHTML，后面跟随的是其版本号。
- 定义：指定文档类型定义包含Frameset（框架集文档）、Strict（严格型文档）和Transitional（过渡型文档），图1-10中声明的是过渡型文档。严格型文档禁止使用W3C规范中指定将逐步淘汰的元素和属性，而过渡型文档可以包含除frameset元素以外的全部内容。
- 语言：指定公开文本的语言，即用于创建所引用对象的自然语言编码系统。该语言定义已编写为ISO 639语言代码（两个字母要大写），默认为EN（英语）。
- URL：指定所引用对象的位置。

从上面的分析结果可以看到DOCTYPE声明语句的写法是严格遵循一定规则的，只有这样浏览器才能够调用对应文档类型的规则集来解释文档中的标签。所谓的文档类型规则集就是W3C公开发布的一个文档类型定义中包含的规则。

1.3.6 XHTML名字空间

在XHTML文档中，还需要注意另一个容易忽略的问题，即给<html>标签定义名字空间。示例代码如下所示。

```
<html xmlns="http://www.w1.org/1999/xhtml">
```

xmlns是html元素的一个特殊属性。xmlns属性是XHTML Name Space的缩写，中文翻译为名字空间，该属性声明了html顶级元素的名字空间。那么，名字空间在文档中是必须的吗？它有什么作用呢？

在标准设计中，名字空间是必须设置的一个属性，用来定义该顶级元素以及其包含的各级子元素的唯一性。名字空间声明允许通过一个网址指向来识别文档内标签的唯一性。

由于XML语言允许用户自定义标签，这就可能存在你定义的标签名称与别人定义的标签名称发生冲突，从而可能引起标签名称不同，但是标签所表示的语义相同。这些文档在网上自由传播或者相互交换文件时，由于名称相同可能会发生语义冲突。此时，需要为各自的文档指定其语义的限制空间，于是xmlns属性就派上用场了。

为了帮助读者理解这个概念，来举一个简单的示例，下面是张三和李四两个人分别定义的文档。

```
<!-- 张三：自定义文档 -->
<document>
    <name> 书名 </name>
    <author> 作者 </author>
    <content> 目录 </content>
</document>
<!-- 李四：自定义文档 -->
<document>
    <title> 论文题目 </title>
    <author> 作者 </author>
    <content> 论文内容 </content>
</document>
```

文档的根元素都是document，同时文档中含有很多相同的元素名，如果这些文档都在网上共享就会发生语义冲突。

此时，若使用xmlns分别为它们定义一个名字空间就不会发生冲突了。示例代码如下所示。

```
<!-- 张三：自定义文档 -->
<document xmlns="http://www.baidu.com/zhangsan">
    <name> 书名 </name>
    <author> 作者 </author>
    <content> 目录 </content>
</document>
<!-- 李四：自定义文档 -->
<document xmlns="http://www.baidu.com/lisi">
    <title> 论文题目 </title>
    <author> 作者 </author>
    <content> 论文内容 </content>
</document>
```

在上面的代码中，张三的文档名字空间为“http://www.baidu.com/zhangsan”，而李四的名字空间为“http://www.baidu.com/lisi”，虽然他们的文档存在相同的标签，但是借助顶级元素中定义的名字空间，相互之间就不会发生语义冲突了。通俗地说，名字空间就是给文档做一个标签，标明该文档是属于哪个网站的。对于HTML文档来说，由于它的元素是固定的，不允许用户进行定义，所以指定的名字空间永远为“http://www.w1.org/1999/xhtml”。

1.4 HTML5基础

HTML5以HTML4为基础，对HTML4进行了大量的修改。下面简单介绍HTML5在HTML4的基础上进行了哪些修改，HTML5与HTML4之间主要的区别是什么。

1.4.1 HTML5语法

1. 内容类型

HTML5的文件扩展名与内容类型（Content Type）保持不变，也就是说，扩展名仍然为“.html”或“.htm”，内容类型仍然为“text/html”。

2. 文档类型声明

根据HTML5设计化繁为简的准则，对文档类型和字符说明都进行了简化。DOCTYPE声明是HTML文件中必不可少的内容，它位于文件第一行。在HTML 4中，声明方法的代码代码如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

在HTML5中，刻意不使用版本声明，一份文档将会适用于所有版本的HTML。HTML5中的DOCTYPE声明方法（不区分大小写）的代码如下所示。

```
<!DOCTYPE html>
```

另外，当使用工具时，也可以在DOCTYPE声明方式中加入SYSTEM识别符，声明方法的代码如下所示。

```
<!DOCTYPE HTML SYSTEM "about:legacy-compat">
```

在HTML5中，像这样的DOCTYPE声明方式是允许的，不区分大小写，也不区分是单引号还是双引号。

提 示

使用HTML5的DOCTYPE会触发浏览器以标准兼容模式显示页面。众所周知，网页都有多种显示模式，如怪异模式（Quirks）、近标准模式（Almost Standards）和标准模式（Standards），其中标准模式也被称为非怪异模式（No-Quirks）。浏览器会根据DOCTYPE来识别应该使用哪种模式，以及使用什么规则来验证页面。

3. 字符编码

在HTML4中，使用meta元素的形式指定文件中的字符编码，代码如下所示。

```
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
```

在HTML5中，可以使用对<meta>元素直接追加charset属性的方式来指定字符编码，代码如下所示。

```
<meta charset="UTF-8">
```

若两种方法都有效，可以继续使用前一种方式，即通过content元素的属性来指定，但是不能同时混合使用两种方式。在以前的网站代码中可能会存在下面代码所示的标记方式，但在HTML5中，这种字符编码方式将被认为是错误的，代码如下所示。

```
<meta charset="UTF-8" http-equiv="Content-Type" content="text/
html;charset=UTF-8">
```

从HTML5开始，对于文件的字符编码推荐使用UTF-8。

4. 版本兼容性

HTML5的语法是为了保证与之前的HTML语法达到最大程度的兼容而设计的。简单说明如下。

- 可以省略标记的元素。

在HTML5中，元素的标记可以省略。具体来说，元素的标记分为三种类型，不允许写结束标记、可以省略结束标记、开始标记和结束标记全部可以省略。下面简单介绍这三种类型各包括哪些HTML5新元素。

第一，不允许写结束标记的元素有：area、base、br、col、command、embed、hr、img、input、keygen、link、meta、param、source、track、wbr。

第二，可以省略结束标记的元素有：li、dt、dd、p、rt、rp、optgroup、option、colgroup、thead、tbody、tfoot、tr、td、th。

第三，可以省略全部标记的元素有：html、head、body、colgroup、tbody。

提 示

不允许写结束标记的元素是指不允许使用开始标记与结束标记将元素括起来的形式，只允许使用<元素/>的形式进行书写。例如，
...</br>的书写方式是错误的，正确的书写方式为
。当然，在HTML5之前的版本中，
这种写法可以被沿用。

可以省略全部标记的元素是指该元素可以完全被省略。注意，即使标记被省略了，该元素还是以隐藏的方式存在着。例如，将body元素省略不写时，它在文档结构中还是存在的，可以使用document.body进行访问。

- 具有布尔值的属性。

布尔值（boolean）的属性如disabled与readonly等，当只写属性而不指定属性值时，属性值表示为true；如果想要将属性值设为false，可以不使用该属性。另外，要想将属性值设定为true时，也可以将属性名设定为属性值，或将空字符串设定为属性值。示例代码如

下所示。

```
<!-- 只写属性，不写属性值，代表属性为 true-->
<input type="checkbox" checked>
<!-- 不写属性，代表属性为 false-->
<input type="checkbox">
<!-- 属性值 = 属性名，代表属性为 true-->
<input type="checkbox" checked="checked">
<!-- 属性值 = 空字符串，代表属性为 true-->
<input type="checkbox" checked="">
```

- 省略引号。

属性值两边既可以用双引号，也可以用单引号。HTML5在此基础上做了一些改进，当属性值不包括空字符串、小于号、大于号、等于号、单引号、双引号等字符时，属性值两边的引号可以省略。例如，下面的写法都是合法的。

```
<input type="text">
<input type='text'>
<input type=text>
```

【示例】下面通过上面介绍的HTML5语法知识，完全用HTML5编写一个文档，该文档中省略了<html>、<head>、<body>等元素。通过这个示例可以复习一下HTML5的DOCTYPE声明、用<meta>元素的charset属性指定字符编码、省略<p>元素的结束标记、使用<元素/>的方式来结束<meta>元素，以及
元素等本节中所介绍到的知识要点。示例代码如下所示。

```
<!DOCTYPE html>
<meta charset="UTF-8">
<title>HTML5 基本语法 </title>
<h1>HTML5 的目标 </h1>
<p>HTML5 的目标是为了能够创建更简单的 Web 程序，书写出更简洁的 HTML 代码。
<br/> 例如，为了使 Web 应用程序的开发变得更容易，提供了很多 API；为了使 HTML 变得更简洁，开发出了新的属性、新的元素等。总体来说，为下一代 Web 平台提供了许许多多新的功能。
```

这段代码在IE浏览器中的运行结果如图1-11所示。

图1-11　第一个HTML5文档

1.4.2 HTML5元素

HTML5引入了很多新的标记元素，根据内容类型的不同，这些元素被分成了7大类，如表1-2所示。

表1-2 HTML5的内容类型

内容类型	说明
内嵌	在文档中添加其他类型的内容，如audio、video、canvas和iframe等
流	在文档和应用的body中使用的元素，如form、h1和small等
标题	段落标题，如h1、h2和hgroup等
交互	与用户交互的内容，如音频和视频的控件、button和textarea等
元数据	通常出现在页面的head中，设置页面其他部分的表现和行为，如script、style和title等
短语	文本和文本标记元素，如mark、kbd、sub和sup等

上表中所有类型的元素都可以通过CSS来设定样式。虽然canvas、audio和video元素在使用时往往需要其他API来配合，以实现细粒度控制，但它们同样可以直接使用。

1. HTML5新增的结构元素

HTML5定义了一组新的语义化标记来描述元素的内容。虽然语义化标记也可以使用HTML标记进行替换，但是它可以简化HTML页面的设计，并且将来搜索引擎在抓取和索引网页的时候也会利用到这些元素的优势。在目前主流的浏览器中已经可以用这些元素了，新增的语义化标记元素如表1-3所示。

表1-3 HTML5新增的语义化标记元素

元素名称	说明
header	标记头部区域的内容（用于整个页面或页面中的一块区域）
footer	标记脚部区域的内容（用于整个页面或页面中的一块区域）
section	Web页面中的一块区域
article	独立的文章内容
aside	相关内容或者引文
nav	导航类辅助内容

根据HTML5效率优先的设计理念，它推崇表现和内容的分离，所以在HTML5的实际编程中，开发人员必须使用CSS来定义样式。

【示例】在下面示例中分别使用HTML5提供的各种语义化结构标记重新设计一个网页，示例代码如下所示，效果如图1-12所示。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" >
<title>HTML5 结构元素 </title>
```