XML 系列教程之一

——XML 教程

目　　录

1 XML 简介

XML 被设计用来传输和存储数据。

HTML 被设计用来显示数据。

1.1 应该掌握的基础知识

在您继续学习之前，需要对以下知识有基本的了解：

- HTML/XHTML
- JavaScript

1.2 什么是 XML

- XML 指可扩展标记语言（EXtensibleMarkupLanguage）；
- XML 是一种标记语言，很类似 HTML；
- XML 的设计宗旨是传输数据，而非显示数据；
- XML 标签没有被预定义，您需要自行定义标签；
- XML 被设计为具有自我描述性；
- XML 是 W3C 的推荐标准。

1.3 XML 与 HTML 的主要差异

XML 不是 HTML 的替代。XML 和 HTML 为不同的目的而设计：

- XML 被设计为传输和存储数据，其焦点是数据的内容；
- HTML 被设计用来显示数据，其焦点是数据的外观；
- HTML 旨在显示信息，而 XML 旨在传输信息。

1.4 没有任何行为的 XML

XML 是不作为的。也许这有点难以理解，但是 XML 不会做任何事情。XML

被设计用来结构化、存储以及传输信息。

下面是 John 写给 George 的便签，存储为 XML：

```
<note>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

这个标签有标题以及留言。它也包含了发送者和接受者的信息。但是，这个 XML 文档仍然没有做任何事情。它仅仅是包装在 XML 标签中的纯粹的信息。我们需要编写软件或者程序，才能传送、接收和显示出这个文档。

1.5 XML 仅仅是纯文本

XML 没什么特别的。它仅仅是纯文本而已。有能力处理纯文本的软件都可以处理 XML。不过，能够读懂 XML 的应用程序可以有针对性地处理 XML 的标签。标签的功能性意义依赖于应用程序的特性。

1.6 通过 XML 您可以发明自己的标签

上例中的标签没有在任何 XML 标准中定义过（比如 <to> 和 <from>）。这些标签是由文档的创作者发明的。这是因为 XML 没有预定义的标签。

在 HTML 中使用的标签（以及 HTML 的结构）是预定义的。HTML 文档只使用在 HTML 标准中定义过的标签（比如<p>、<h1>等等）。XML 允许创作者定义自己的标签和自己的文档结构。

1.7 XML 不是对 HTML 的替代

XML 是对 HTML 的补充。XML 不是对 HTML 的替代，理解这一点很重要。在大多数 web 应用程序中，XML 用于传输数据，而 HTML 用于格式化并显示

数据。

对 XML 的最好的描述是：XML 是独立于软件和硬件的信息传输工具。

1.8 XML 是 W3C 的推荐标准

XML 于 1998 年 2 月 10 日成为 W3C 的推荐标准。

1.9 XML 无所不在

当我们看到 XML 标准突飞猛进的开发进度，以及大批的软件开发商采用这个标准的日新月异的速度时，真的是不禁感叹这真是令人叹为观止。

目前，XML 在 Web 中起到的作用不会亚于一直作为 Web 基石的 HTML。

XML 无所不在。XML 是各种应用程序之间进行数据传输的最常用的工具，并且在信息存储和描述领域变得越来越流行。

2 XML 的用途

XML 应用于 web 开发的许多方面，常用于简化数据的存储和共享。

2.1 XML 把数据从 HTML 分离

如果你需要在 HTML 文档中显示动态数据，那么每当数据改变时将花费大量的时间来编辑 HTML。

通过 XML，数据能够存储在独立的 XML 文件中。这样你就可以专注于使用 HTML 进行布局和显示，并确保修改底层数据不再需要对 HTML 进行任何的改变。

通过使用几行 JavaScript，你就可以读取一个外部 XML 文件，然后更新 HTML 中的数据内容。

2.2 XML 简化数据共享

在真实的世界中，计算机系统和数据使用不兼容的格式来存储数据。

XML 数据以纯文本格式进行存储，因此提供了一种独立于软件和硬件的数据存储方法。这让创建不同应用程序可以共享的数据变得更加容易。

2.3 XML 简化数据传输

通过 XML，可以在不兼容的系统之间轻松地交换数据。

对开发人员来说，其中一项最费时的挑战一直是在因特网上的不兼容系统之间交换数据。

由于可以通过各种不兼容的应用程序来读取数据，以 XML 交换数据降低了这种复杂性。

2.4 XML 简化平台的变更

升级到新的系统（硬件或软件平台），总是非常费时的。必须转换大量的数据，不兼容的数据经常会丢失。

XML 数据以文本格式存储。这使得 XML 在不损失数据的情况下，更容易扩展或升级到新的操作系统、新应用程序或新的浏览器。

2.5 XML 使您的数据更有用

由于 XML 独立于硬件、软件以及应用程序，XML 使您的数据更可用，也更有用。

不同的应用程序都能够访问您的数据，不仅仅在 HTML 页中，也可以从 XML 数据源中进行访问。

通过 XML，您的数据可供各种阅读设备使用（手持的计算机、语音设备、新闻阅读器等），还可以供盲人或其他残障人士使用。

2.6 XML 用于创建新的 Internet 语言

很多新的 Internet 语言是通过 XML 创建的，其中的例子包括：

- XHTML，最新的 HTML 版本；
- WSDL：用于描述可用的 web service；
- WAP 和 WML：用于手持设备的标记语言；
- RSS：用于 RSS feed 的语言；
- RDF 和 OWL：用于描述资源和本体；
- SMIL：用于描述针针对 web 的多媒体。

2.7 假如开发人员都是理性的

假如他们都是理性的，就让未来的应用程序使用 XML 来交换数据吧。

未来也许会出现某种字处理软件、电子表格程序以及数据库，它们可以使用纯文本格式读取彼此的数据，而不需要使用任何的转换程序。

我们现在能做的只有祈祷微软公司和所有其他的软件开发商在这一方面取得一致了。

3 XML 树结构

XML 文档形成了一种树结构，它从“根部”开始，然后扩展到“枝叶”。

3.1 一个 XML 文档实例

XML 使用了简单的具有自我描述性的语法：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

第一行是 XML 声明。它定义 XML 的版本（1.0）和所使用的编码（ISO-8859-1 = Latin-1/西欧字符集）。

下一行描述文档的根元素（像在说：“本文档是一个便签”）：

```
<note>
```

接下来 4 行描述根的 4 个子元素（to，from，heading 以及 body）：

```
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
```

最后一行定义根元素的结尾：

```
</note>
```

从本例可以设想，该 XML 文档包含了 John 给 George 的一张便签。

XML 具有出色的自我描述性，你同意吗？

3.2 XML 文档形成一种树结构

XML 文档必须包含根元素。该元素是所有其他元素的父元素。

XML 文档中的元素形成了一棵文档树。这棵树从根部开始，并扩展到树的最底端。

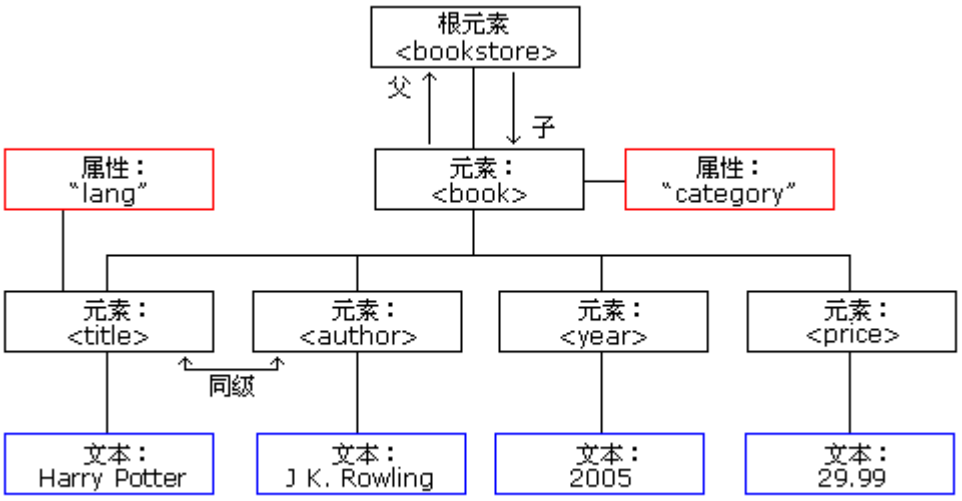
所有元素均可拥有子元素：

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

父、子以及同胞等术语用于描述元素之间的关系。父元素拥有子元素。相同层级上的子元素成为同胞（兄弟或姐妹）。

所有元素均可拥有文本内容和属性（类似 HTML 中）。

3.3 实例



上图表示下面的 XML 中的一本书：

```
<bookstore>
<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>
```

例子中的根元素是<bookstore>。文档中的所有<book>元素都被包含在

<bookstore>中。

<book>元素有 4 个子元素：<title>、< author>、<year>、<price>。

4 XML 语法规则

XML 的语法规则很简单，且很有逻辑。这些规则很容易学习，也很容易使用。

4.1 所有 XML 元素都须有关闭标签

在 HTML，经常会看到没有关闭标签的元素：

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

在 XML 中，省略关闭标签是非法的。所有元素都必须有关闭标签：

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

注释：您也许已经注意到 XML 声明没有关闭标签。这不是错误。声明不属于 XML 本身的组成部分。它不是 XML 元素，也不需要关闭标签。

4.2 XML 标签对大小写敏感

XML 元素使用 XML 标签进行定义。

XML 标签对大小写敏感。在 XML 中，标签 <Letter> 与标签 <letter> 是不同的。

必须使用相同的大小写来编写打开标签和关闭标签：

```
<Message>这是错误的。</message>
<message>这是正确的。</message>
```

注释：打开标签和关闭标签通常被称为开始标签和结束标签。不论您喜欢哪

种术语，它们的概念都是相同的。

4.3 XML 必须正确地嵌套

在 HTML 中，常会看到没有正确嵌套的元素：

```
<b><i>This text is bold and italic</b></i>
```

在 XML 中，所有元素都必须彼此正确地嵌套：

```
<b><i>This text is bold and italic</i></b>
```

在上例中，正确嵌套的意思是：由于<i>元素是在<b>元素内打开的，那么它必须在<b>元素内关闭。

4.4 XML 文档必须有根元素

XML 文档必须有一个元素是所有其他元素的父元素。该元素称为根元素。

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

4.5 XML 的属性值须加引号

与 HTML 类似，XML 也可拥有属性（名称/值的对）。

在 XML 中，XML 的属性值须加引号。请研究下面的两个 XML 文档。第一个是错误的，第二个是正确的：

```
<note date=08/08/2008>
<to>George</to>
<from>John</from>
</note>
```

```
<note date="08/08/2008">
<to>George</to>
<from>John</from>
```

```
</note>
```

在第一个文档中的错误是 note 元素中的 date 属性没有加引号。

4.6 实体引用

在 XML 中，一些字符拥有特殊的意义。

如果你把字符"<"放在 XML 元素中，会发生错误，这是因为解析器会把它当作新元素的开始。

这样会产生 XML 错误：

```
<message>if salary < 1000 then</message>
```

为了避免这个错误，请用一个实体引用来代替"<"字符：

```
<message>if salary &lt; 1000 then</message>
```

在 XML 中，有 5 个预定义的实体引用：

<	<	小于
>	>	大于
&	&	和号
'	'	单引号
"	"	引号

注释：在 XML 中，只有字符"<"和"&"确实是非法的。大于号是合法的，但是用实体引用来代替它是一个好习惯。

4.7 XML 中的注释

在 XML 中编写注释的语法与 HTML 的语法很相似：

```
<!-- This is a comment -->
```

在 XML 中，空格会被保留

HTML 会把多个连续的空格字符裁减为一个：

```
HTML:    Hello          my name is David.
输出:    Hello my name is David.
```

在 XML 中，文档中的空格不会被删节。

4.8 XML 以 LF 存储换行

在 Windows 应用程序中，换行通常以一对字符来存储：回车符（CR）和换行符（LF）。这对字符与打字机设置新行的动作有相似之处。在 Unix 应用程序中，新行以 LF 字符存储。而 Macintosh 应用程序使用 CR 来存储新行。

5 XML 元素

XML 文档包含 XML 元素。

5.1 什么是 XML 元素

XML 元素指的是从（且包括）开始标签直到（且包括）结束标签的部分。

元素可包含其他元素、文本或者两者的混合物。元素也可以拥有属性。

```
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>
```

在上例中，<bookstore>和<book>都拥有元素内容，因为它们包含了其他元素。<author>只有文本内容，因为它仅包含文本。

在上例中，只有<book>元素拥有属性（category="CHILDREN"）。

5.2 XML 命名规则

XML 元素必须遵循以下命名规则：

- 名称可以含字母、数字以及其他的字符；
- 名称不能以数字或者标点符号开始；
- 名称不能以字符"xml"（或者 XML、Xml）开始；
- 名称不能包含空格；
- 可使用任何名称，没有保留的字词。

5.3 最佳命名习惯

使名称具有描述性。使用下划线的名称也很不错。

名称应当比较简短，比如：<book_title>，而不是：<the_title_of_the_book>。

避免"-"字符。如果您按照这样的方式进行命名："first-name"，一些软件会认为你需要提取第一个单词。

避免"."字符。如果您按照这样的方式进行命名："first.name"，一些软件会认为"name"是对象"first"的属性。

避免":"字符。冒号会被转换为命名空间来使用（稍后介绍）。

XML 文档经常有一个对应的数据库 ,其中的字段会对应 XML 文档中的元素。有一个实用的经验，即使用数据库的名称规则来命名 XML 文档中的元素。

非英语的字母比如 éòá 也是合法的 XML 元素名 ,不过需要留意当软件开发商不支持这些字符时可能出现的问题。

5.4 XML 元素是可扩展的

XML 元素是可扩展，以携带更多的信息。

请看下面这个 XML 例子：

```
<note>
<to>George</to>
<from>John</from>
<body>Don't forget the meeting!</body>
</note>
```

让我们设想一下，我们创建了一个应用程序，可将<to>、<from>以及<body>元素提取出来，并产生以下的输出：

```
MESSAGE
To: George
From: John

Don't forget the meeting!
```

想象一下，之后这个 XML 文档作者又向这个文档添加了一些额外的信息：

```
<note>
<date>2008-08-08</date>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

那么这个应用程序会中断或崩溃吗？不会。这个应用程序仍然可以找到 XML 文档中的<to>、<from>以及<body>元素，并产生同样的输出。

XML 的优势之一，就是可以经常在不中断应用程序的情况进行扩展。

6 XML 属性

XML 元素可以在开始标签中包含属性，类似 HTML。

属性（Attribute）提供关于元素的额外信息。

6.1 XML 属性

从 HTML，你会回忆起这个：<img src="computer.gif">。"src"属性提供有关<img>元素的额外信息。

在 HTML 中（以及在 XML 中），属性提供有关元素的额外信息：

```
<img src="computer.gif">
<a href="demo.asp">
```

属性通常提供不属于数据组成部分的信息。在下面的例子中，文件类型与数据无关，但是对需要处理这个元素的软件来说却很重要：

```
<file type="gif">computer.gif</file>
```

6.2 XML 属性必须加引号

属性值必须被引号包围，不过单引号和双引号均可使用。比如一个人的性别，person 标签可以这样写：

```
<person sex="female">
```

或者这样也可以：

```
<person sex='female'>
```

注释：如果属性值本身包含双引号，那么有必要使用单引号包围它，就像这个例子：

```
<gangster name='George "Shotgun" Ziegler'>
```

或者可以使用实体引用：

```
<gangster name="George &quot;Shotgun&quot; Ziegler">
```

6.3 XML 元素 vs.属性

请看这些例子：

```
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
```

```
</person>
```

```
<person>
  <sex>female</sex>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

在第一个例子中 ,sex 是一个属性。在第二个例子中 ,sex 则是一个子元素。两个例子均可提供相同的信息。

没有什么规矩可以告诉我们什么时候该使用属性 ,而什么时候该使用子元素。我的经验是在 HTML 中，属性用起来很便利，但是在 XML 中，您应该尽量避免使用属性。如果信息感觉起来很像数据，那么请使用子元素吧。

6.4 我最喜欢的方式

下面的三个 XML 文档包含完全相同的信息：

第一个例子中使用了 date 属性：

```
<note date="08/08/2008">
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

第二个例子中使用了 date 元素：

```
<note>
<date>08/08/2008</date>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

第三个例子中使用了扩展的 date 元素（这是我的最爱）:

```
<note>
<date>
  <day>08</day>
  <month>08</month>
  <year>2008</year>
</date>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

6.5 避免 XML 属性

因使用属性而引起的一些问题：

- 属性无法包含多个值（子元素可以）;
- 属性无法描述树结构（子元素可以）;
- 属性不易扩展（为未来的变化）;
- 属性难以阅读和维护。

请尽量使用元素来描述数据。而仅仅使用属性来提供与数据无关的信息。

不要做这样的蠢事（这不是 XML 应该被使用的方式）:

```
<note day="08" month="08" year="2008"
to="George" from="John" heading="Reminder"
body="Don't forget the meeting!">
</note>
```

6.6 针对元数据的 XML 属性

有时候会向元素分配 ID 引用。这些 ID 索引可用于标识 XML 元素，它起作用的方式与 HTML 中 ID 属性是一样的。这个例子向我们演示了这种情况：

```
<messages>
  <note id="501">
    <to>George</to>
    <from>John</from>
    <heading>Reminder</heading>
    <body>Don't forget the meeting!</body>
```

```
  </note>
  <note id="502">
    <to>John</to>
    <from>George</from>
    <heading>Re: Reminder</heading>
    <body>I will not</body>
  </note>
</messages>
```

上面的 ID 仅仅是一个标识符，用于标识不同的便签。它并不是便签数据的组成部分。

在此我们极力向您传递的理念是：元数据（有关数据的数据）应当存储为属性，而数据本身应当存储为元素。

7 XML 验证

拥有正确语法的 XML 被称为“形式良好”的 XML。

通过某个 DTD 进行了验证的 XML 是“合法”的 XML。

7.1 形式良好的 XML 文档

一个"形式良好"的 XML 文档拥有正确的语法。

一个"形式良好"的 XML 文档会遵守前几章介绍过的 XML 语法规则：

- XML 文档必须有根元素；
- XML 文档必须有关闭标签；
- XML 标签对大小写敏感；
- XML 元素必须被正确的嵌套；
- XML 属性必须加引号。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>George</to>
```

```
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

7.2 验证 XML 文档

一个合法的 XML 文档是"形式良好"的 XML 文档，同样遵守文档类型定义（DTD）的语法规则：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

在上例中，DOCTYPE 声明是对外部 DTD 文件的引用。下面的段落展示了这个文件的内容。

7.3 XML DTD

DTD 的作用是定义 XML 文档的结构。它使用一系列合法的元素来定义文档结构：

```
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to     (#PCDATA)>
  <!ELEMENT from   (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body   (#PCDATA)>
]>
```

7.4 XML Schema

W3C 支持一种基于 XML 的 DTD 代替者，它名为 XML Schema：

```
<xs:element name="note">

<xs:complexType>
  <xs:sequence>
```

```
    <xs:element name="to"     type="xs:string"/>
    <xs:element name="from"    type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body"    type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

8 XML 验证器

8.1 XML 错误会终止您的程序

XML 文档中的错误会终止你的 XML 程序。

W3C 的 XML 规范声明：如果 XML 文档存在错误，那么程序就不应当继续处理这个文档。理由是，XML 软件应当轻巧，快速，具有良好的兼容性。

如果使用 HTML，创建包含大量错误的文档是有可能的（比如你忘记了结束标签）。其中一个主要的原因是 HTML 浏览器相当臃肿，兼容性也很差，并且它们有自己的方式来确定当发现错误时文档应该显示为什么样子。

使用 XML 时，这种情况不应当存在。

9 XML 浏览器支持

几乎所有的主流浏览器均支持 XML 和 XSLT。

9.1 Mozilla Firefox

从 1.0.2 版本开始，Firefox 就已开始支持 XML 和 XSLT（包括 CSS）。

9.2 Mozilla

Mozilla 含有用于 XML 解析的 Expat，并支持显示 XML+CSS。Mozilla 同

时拥有对 Namespaces 的某些支持。

Mozilla 同样可做到对 XSLT 的执行（XSLT implementation）。

9.3 Netscape

自从版本 8 开始，Netscape 开始使用 Mozilla 的引擎，因此它对 XML/XSLT 的支持与 Mozilla 是相同的。

9.4 Opera

自从版本 9 开始，Opera 已经拥有对 XML/XSLT（以及 CSS）的支持。版本 8 仅支持 XML+CSS。

9.5 Internet Explorer

自从版本 6 开始，Internet Explorer 就开始支持 XML、Namespaces、CSS、XSLT 以及 XPath。

注释：Internet Explorer 5 同样拥有对 XML 的支持，但是 XSL 部分与 W3C 的官方标准不兼容！

10 查看 XML 文件

在所有现代浏览器中，可能够查看原始的 XML 文件。

不要指望 XML 文件会直接显示为 HTML 页面。

10.1 查看 XML 文件

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!--   Copyright w3school.com.cn   -->
- <note>
    <to>George</to>
    <from>John</from>
    <heading>Reminder</heading>
    <body>Don't forget the meeting this weekend!</body>
  </note>
```

打开 XML 文件，XML 文档将显示为代码颜色化的根以及子元素。通过点击元素左侧的加号或减号，可以展开或收起元素的结构。如需查看不带有+和-符号的源代码，请从浏览器菜单中选择“查看源代码”。

注释：在 Netscape，Opera 以及 Safari 中，仅仅会显示元素文本！要查看原始的 XML，请右击页面，然后选择“查看源代码”。

10.2 查看某个无效的 XML 文件

如果浏览器打开了某个有错误的 XML 文件，那么它会报告这个错误。

10.3 为什么 XML 会这样显示

XML 文档不会携带有关如何显示数据的信息。

由于 XML 标签由 XML 文档的作者“发明”，浏览器无法确定像<table>这样一个标签究竟描述一个 HTML 表格还是一个餐桌。

在没有任何有关如何显示数据的信息的情况下，大多数的浏览器都会仅仅把 XML 文档显示为源代码。

在下面的章节，我们会了解几个有关这个显示问题的解决方案，其中会使用 CSS、XSL、JavaScript 以及 XML 数据岛。

11 使用 CSS 显示 XML

通过使用 CSS，可为 XML 文档添加显示信息。

11.1 使用 CSS 显示您的 XML

使用 CSS 来格式化 XML 文档是有可能的。

下面的例子就是关于如何使用 CSS 样式表来格式化某个 XML 文档：

这是样式表：

```
CATALOG
{
background-color: #ffffff;
width: 100%;
}
CD
{
display: block;
margin-bottom: 30pt;
margin-left: 0;
}
TITLE
{
color: #FF0000;
font-size: 20pt;
}
ARTIST
{
color: #0000FF;
font-size: 20pt;
}
COUNTRY,PRICE,YEAR,COMPANY
{
display: block;
color: #000000;
margin-left: 20pt;
}
```

下面是此 XML 文件的一个片断。第二行，<?xml-stylesheet type="text/css" href="cd_catalog.css"?>，把这个 XML 文件链接到 CSS 文件：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="cd catalog.css"?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
```

```
  </CD>
  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
.
.
.
.
</CATALOG>
```

注释：使用 CSS 格式化 XML 不能代表 XML 文档样式化的未来。XML 文档应当使用 W3C 的 XSL 标准进行格式化！

12 使用 XSLT 显示 XML

通过使用 XSLT，您可以向 XML 文档添加显示信息。

12.1 使用 XSLT 显示 XML

XSLT 是首选的 XML 样式表语言。

XSLT（eXtensible Stylesheet Language Transformations）远比 CSS 更加完善。

使用 XSLT 的方法之一是在浏览器显示 XML 文件之前，先把它转换为 HTML，正如以下的这些例子演示的那样：

XSLT 样式表：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited with XML Spy v2007 (http://www.altova.com) -->
<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/1999/xhtml">
  <body style="font-family:Arial,helvetica,sans-serif;font-size:12pt;
        background-color:#EEEEEE">
```

```
    <xsl:for-each select="breakfast_menu/food">
      <div style="background-color:teal;color:white;padding:4px">
        <span style="font-weight:bold;color:white">
        <xsl:value-of select="name"/></span>
        - <xsl:value-of select="price"/>
      </div>
      <div style="margin-left:20px;margin-bottom:1em;font-size:10pt">
        <xsl:value-of select="description"/>
        <span style="font-style:italic">
          (<xsl:value-of select="calories"/> calories per serving)
        </span>
      </div>
    </xsl:for-each>
  </body>
</html>
```

下面是此 XML 文件的一个片断。 第二行，<?xml-stylesheet type="text/xsl" href="simple.xsl"?>，把这个 XSL 文件链接到 XML 文件：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="simple.xsl"?>
<breakfast menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
       two of our famous Belgian Waffles
    </description>
    <calories>650</calories>
  </food>
</breakfast_menu>
```

12.2 在服务器上通过 XSLT 转换 XML

在上例中，XSLT 转换是由浏览器完成的，浏览器读取的是 XML 文件。

在使用 XSLT 来转换 XML 时，不同的浏览器可能会产生不同结果。为了减少这种问题，可以在服务器上进行 XSLT 转换。

请注意，不论转换由服务器还是由浏览器进行，输出结果完成相同。

13 XML 解析器

大多数浏览器都有读取和操作 XML 的内建 XML 解析器。

解析器把 XML 转换为 JavaScript 可访问的对象。

13.1 解析 XML

所有现代浏览器都有读取和操作 XML 的内建 XML 解析器。

解析器把 XML 载入内存，然后把它转换为可通过 JavaScript 访问的 XML DOM 对象。

微软的 XML 解析器与其他浏览器中的解析器之间，存在一些差异。微软的解析器支持 XML 文件和 XML 字符串（文本）的加载，而其他浏览器使用单独的解析器。不过，所有的解析器都包含遍历 XML 树、访问插入及删除节点（元素）及其属性的函数。

在本教程中，我们将为您展示如何创建可同时工作于 Internet Explorer 和其他浏览器中的脚本。

注释：当我们谈及 XML 解析，我们常常会使用有关 XML 元素的术语：节点。

13.2 通过微软的 XML 解析器来加载 XML

微软的 XML 解析器内建于 Internet Explorer 5 以及更高的版本中。

下面的 JavaScript 片段把一个 XML 文档载入解析器中：

```
var xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.load("note.xml");
```

例子解释：

上面代码的第一个行创建一个空的微软 XML 文档对象。

第二行关闭异步加载 ,这样确保在文档完全加载之前解析器不会继续脚本的执行。

第三行告知解析器加载名为"note.xml"的 XML 文档。

下面的 JavaScript 片段把字符串 txt 载入解析器：

```
var xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.loadXML(txt);
```

注释：loadXML()方法用于加载字符串（文本），load()用于加载文件。

13.3 在 Firefox 及其他浏览器中的 XML 解析器

下面的 JavaScript 片段把 XML 文档（"note.xml"）载入解析器：

```
var xmlDoc=document.implementation.createDocument("","",null);
xmlDoc.async="false";
xmlDoc.load("note.xml");
```

例子解释：

上面代码的第一个行创建一个空的 XML 文档对象。

第二行关闭异步加载 ,这样确保在文档完全加载之前解析器不会继续脚本的执行。

第三行告知解析器加载名为"note.xml"的 XML 文档。

下面的 JavaScript 片段把字符串 txt 载入解析器：

```
var parser=new DOMParser();
var doc=parser.parseFromString(txt,"text/xml");
```

例子解释：

上面代码的第一个行创建一个空的微软 XML 文档对象。

第二行告知解析器载入名为 txt 的字符串。

注释：Internet Explorer 使用 loadXML()方法来解析 XML 字符串，而其他浏览器使用 DOMParser 对象。

13.4 跨域访问

出于安全方面的原因，现代的浏览器不允许跨域的访问。

这意味着 ,网页以及它试图加载的 XML 文件 ,都必须位于相同的服务器上。否则，xmlDoc.load()将产生错误"Access is denied"。

14 XML DOM

DOM（Document Object Model，文档对象模型）定义访问和操作文档的一套标准方法。

14.1 XML DOM

XML DOM（XML Document Object Model）定义一套访问和操作 XML 文档的标准方法。

DOM 把 XML 文档作为树结构来查看。能够通过 DOM 树来访问所有元素。可以修改或删除它们的内容，并创建新的元素。元素，它们的文本，以及它们的属性，都被认为是节点。

在下面的例子中，我们使用 DOM 引用从<to>元素中获取文本：

```
xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue
```

- xmlDoc，由解析器创建的 XML 文档；
- getElementsByTagName("to")[0]，第一个<to>元素；
- childNodes[0]，<to>元素的第一个子元素（文本节点）;
- nodeValue，节点的值（文本本身）。

14.2 HTML DOM

HTML DOM（HTML Document Object Model）定义一套访问和操作

HTML 文档的标准方法。

可以通过 HTML DOM 访问所有 HTML 元素。

在下面的例子中，我们使用 DOM 引用来改变 id="to"的 HTML 元素的文本：

```
document.getElementById("to").innerHTML=
```

- document，HTML 文档；
- getElementById("to")，其中的 id="to"的 HTML 元素；
- innerHTML，HTML 元素的内部文本。

14.3 解析 XML 文件——跨浏览器实例

下列代码把一个 XML 文档（"note.xml"）载入 XML 解析器中：

```
<html>
<head>
<script type="text/javascript">
function parseXML()
{
try //Internet Explorer
  {
  xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
  }
catch(e)
  {
  try //Firefox, Mozilla, Opera, etc.
    {
    xmlDoc=document.implementation.createDocument("","",null);
    }
  catch(e)
    {
    alert(e.message);
    return;
    }
  }
xmlDoc.async=false;
xmlDoc.load("note.xml");

document.getElementById("to").innerHTML=
xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
document.getElementById("from").innerHTML=
```

```
xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
document.getElementById("message").innerHTML=
xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
}
</script>
</head>

<body onload="parseXML()">
<h1>W3School.com.cn Internal Note</h1>
<p><b>To:</b> <span id="to"></span><br />
<b>From:</b> <span id="from"></span><br />
<b>Message:</b> <span id="message"></span>
</p>
</body>
</html>
```

输出：

```
W3School.com.cn Internal Note

To: George
From: John
Message: Don't forget the meeting!
```

14.4 重要注释

如需从 XML 中提取文本"John"，语法是：

```
getElementsByTagName("from")[0].childNodes[0].nodeValue
```

在这个 XML 例子中，只有一个<from>标签中，但是仍然需要规定数组的下标[0] ,这是因为 XML 解析器方法 getElementsByTagName()返回所有<from>节点的一个数组。

14.5 解析 XML 字符串——跨浏览器实例

下面的代码加载并解析一个 XML 字符串：

```
<html>
<head>
<script type="text/javascript">
function parseXML()
{
text="<note>";
```

```
text=text+"<to>George</to>";
text=text+"<from>John</from>";
text=text+"<heading>Reminder</heading>";
text=text+"<body>Don't forget the meeting!</body>";
text=text+"</note>";
try //Internet Explorer
  {
  xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
  xmlDoc.async="false";
  xmlDoc.loadXML(text);
  }
catch(e)
  {
  try // Firefox, Mozilla, Opera, etc.
    {
    parser=new DOMParser();
    xmlDoc=parser.parseFromString(text,"text/xml");
    }
  catch(e)
    {
    alert(e.message);
    return;
    }
  }
document.getElementById("to").innerHTML=
xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
document.getElementById("from").innerHTML=
xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
document.getElementById("message").innerHTML=
xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
}
</script>
</head>

<body onload="parseXML()">
<h1>W3School.com.cn Internal Note</h1>
<p><b>To:</b> <span id="to"></span><br />
<b>From:</b> <span id="from"></span><br />
<b>Message:</b> <span id="message"></span>
</p>
</body>
</html>
```

输出：

```
W3School.com.cn Internal Note

To: George
From: John
Message: Don't forget the meeting!
```

注释：Internet Explorer 使用 loadXML()方法来解析 XML 字符串，而其他浏览器使用 DOMParser 对象。

15 XML to HTML

本章讲解如何把 XML 数据显示为 HTML。

15.1 在 HTML 中显示数据

本例遍历一个 XML 文件（cd_catalog.xml），然后把每个 CD 元素显示为一个 HTML 表格行：

```
<html>
<body>

<script type="text/javascript">
var xmlDoc=null;
if (window.ActiveXObject)
{
// code for IE
xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
}
else if (document.implementation.createDocument)
{
// code for Mozilla, Firefox, Opera, etc.
xmlDoc=document.implementation.createDocument("","",null);
}
else
{
alert('Your browser cannot handle this script');
}
if (xmlDoc!=null)
{
xmlDoc.async=false;
```

```
xmlDoc.load("cd_catalog.xml");

document.write("<table border='1'>");

var x=xmlDoc.getElementsByTagName("CD");
for (i=0;i<x.length;i++)
{
document.write("<tr>");
document.write("<td>");
document.write(
x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue);
document.write("</td>");

document.write("<td>");
document.write(
x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue);
document.write("</td>");
document.write("</tr>");
}
document.write("</table>");
}
</script>

</body>
</html>
```

例子解释：

1. 检测浏览器，然后使用合适的解析器来加载 XML；
2. 创建一个 HTML 表格（ <table border="1"> ）；
3. 使用 getElementsByTagName()来获得所有 XML 的 CD 节点；
4. 针对每个 CD 节点，把 ARTIST 和 TITLE 中的数据显示为表格数据；
5. 用</table>结束表格。

16 XMLHttpRequest 对象

XMLHttpRequest 对象提供了在网页加载后与服务器进行通信的方法。

16.1 什么是 XMLHttpRequest 对象

XMLHttpRequest 对象是开发者的梦想，因为您能够：

- 在不重新加载页面的情况下更新网页；
- 在页面已加载后从服务器请求数据；
- 在页面已加载后从服务器接收数据；
- 在后台向服务器发送数据。

所有现代的浏览器都支持 XMLHttpRequest 对象。

16.2 创建 XMLHttpRequest 对象

通过一行简单的 JavaScript 代码 ,我们就可以创建 XMLHttpRequest 对象。

在所有现代浏览器中（包括 IE 7）:

```
xmlhttp=new XMLHttpRequest()
```

在 Internet Explorer 5 和 6 中：

```
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP")
```

实例

```
<script type="text/javascript">
var xmlhttp;
function loadXMLDoc(url)
{
xmlhttp=null;
if (window.XMLHttpRequest)
  {// code for all new browsers
  xmlhttp=new XMLHttpRequest();
  }
else if (window.ActiveXObject)
  {// code for IE5 and IE6
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
if (xmlhttp!=null)
  {
  xmlhttp.onreadystatechange=state Change;
  xmlhttp.open("GET",url,true);
```

```
  xmlhttp.send(null);
  }
else
  {
  alert("Your browser does not support XMLHTTP.");
  }
}

function state Change()
{
if (xmlhttp.readyState==4)
  {// 4 = "loaded"
  if (xmlhttp.status==200)
    {// 200 = OK
    // ...our code here...
    }
  else
    {
    alert("Problem retrieving XML data");
    }
  }
}
</script>
```

注释：onreadystatechange 是一个事件句柄。它的值（state_Change）是一个函数的名称，当 XMLHttpRequest 对象的状态发生改变时，会触发此函数。状态从 0（uninitialized）到 4（complete）进行变化。仅在状态为 4 时，我们才执行代码。

16.3 为什么使用 Async=true

我们的实例在 open()的第三个参数中使用了"true"。

该参数规定请求是否异步处理。

True 表示脚本会在 send()方法之后继续执行 ,而不等待来自服务器的响应。

onreadystatechange 事件使代码复杂化了。但是这是在没有得到服务器响应的情况下，防止代码停止的最安全的方法。

通过把该参数设置为"false" ,可以省去额外的 onreadystatechange 代码。如果在请求失败时是否执行其余的代码无关紧要，那么可以使用这个参数。

16.4 XML/ASP

您也可以把 XML 文档打开并发送到服务器上的 ASP 页面，分析此请求，然后传回结果。

```
<html>
<body>
<script type="text/javascript">
xmlHttp=null;
if (window.XMLHttpRequest)
  {// code for IE7, Firefox, Opera, etc.
  xmlHttp=new XMLHttpRequest();
  }
else if (window.ActiveXObject)
  {// code for IE6, IE5
  xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
if (xmlHttp!=null)
  {
  xmlHttp.open("GET", "note.xml", false);
  xmlHttp.send(null);
  xmlDoc=xmlHttp.responseText;

  xmlHttp.open("POST", "demo dom http.asp", false);
  xmlHttp.send(xmlDoc);
  document.write(xmlHttp.responseText);
  }
else
  {
  alert("Your browser does not support XMLHTTP.");
  }
</script>
</body>
</html>
```

ASP 页面，由 VBScript 编写：

```
<%
set xmldoc = Server.CreateObject("Microsoft.XMLDOM")
```

```
xmldoc.async=false
xmldoc.load(request)

for each x in xmldoc.documentElement.childNodes
   if x.NodeName = "to" then name=x.text
next
response.write(name)
%>
```

通过使用 response.write 属性把结果传回客户端。

16.5 XMLHttpRequest 对象是 W3C 的标准吗

任何 W3C 推荐标准均未规定 XMLHttpRequest 对象。

不过，W3C DOM Level 3 的"Load and Save"规范包含了一些相似的功能性，但是还没有任何浏览器实现它们。

17 XML 应用程序

本节演示由 HTML 和 JavaScript 构建的一个小型 XML 应用程序。

17.1 XML 文档实例

请看下面这个 XML 文档 ("cd_catalog.xml")，它描述了一个 CD 目录：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
.
.
... more ...
.
```

17.2 加载 XML 文档

为了加载 XML 文档（cd_catalog.xml），我们使用了与 XML 解析器那一节中相同的代码：

```
var xmlDoc;
if (window.ActiveXObject)
  {
  // code for IE
  xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
  }
else if (document.implementation.createDocument)
  {
  // code for Firefox, Mozilla, Opera, etc.
  xmlDoc=document.implementation.createDocument("","",null);
  }
else
  {
  alert('Your browser cannot handle this script');
  }
xmlDoc.async=false;
xmlDoc.load("cd_catalog.xml");
```

在本代码执行后，xmlDoc 成为一个 XML DOM 对象，可由 JavaScript 访问。

17.3 把 XML 数据显示为 HTML 表格

以下代码使用来自 XML DOM 对象的数据来填充一个 HTML 表格：

```
document.write("<table border='1'>");

var x=xmlDoc.getElementsByTagName("CD");
for (var i=0;i<x.length;i++)
{
document.write("<tr>");
document.write("<td>");
document.write(
x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue);
document.write("</td>");

document.write("<td>");
document.write(
```

```
x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue);
document.write("</td>");
document.write("</tr>");
}
document.write("</table>");
```

针对 XML 文档中的每个 CD 元素，会创建一个表格行。每个表格行包含两个表格数据单元，其中的数据来自当前 CD 元素的 ARTIST 和 TITLE。

17.4 在任意 HTML 元素中显示 XML 数据

XML 数据可以拷贝到任何有能力显示文本的 HTML 元素。

下面的代码为 HTML 文件的<head>部分。这段代码从第一个<CD>元素中获得 XML 数据，然后在 id="show"的 HTML 元素中显示数据：

```
var x=xmlDoc.getElementsByTagName("CD");
i=0;
function display()
{
artist=
(x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue);
title=
(x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue);
year=
(x[i].getElementsByTagName("YEAR")[0].childNodes[0].nodeValue);
txt="Artist: "+artist+"<br />Title: "+title+"<br />Year: "+year;
document.getElementById("show").innerHTML=txt;
}
```

HTML 的 body 元素包含一个 onload 事件属性，它的作用是在页面已经加载时调用 display()函数。body 元素中还包含了供接受 XML 数据的 元素。

```
<div id='show'></div>
</body>
```

通过上例，你只能看到来自 XML 文档中第一个 CD 元素中的数据。为了导航到数据的下一行，必须添加更多的代码。

17.5 添加导航脚本

为了向上例添加导航（功能），需要创建 next()和 previous()两个函数：

```
function next()
{
if (i<x.length)
  {
  i++;
  display();
  }
}

function previous()
{
if (i>0)
  {
  i--;
  display();
  }
}
```

next()函数确保已到达最后一个 CD 元素后不显示任何东西，previous()函数确保已到达第一个 CD 元素后不显示任何东西。

通过点击 next/previous 按钮来调用 next()和 previous()函数：

```
<input type="button" onclick="previous()" value="previous" />
<input type="button" onclick="next()" value="next" />
```

17.6 现在，把所有的东西组合起来

只需要一点点创新，您就可以创建一个完整的应用程序。

如果您使用本页中学到的知识，再加上一点点想象力，就可以轻松地开发出一个完整的应用程序。

18 XML 命名空间（XML Namespaces）

XML 命名空间可提供避免元素命名冲突的方法。

18.1 命名冲突

由于 XML 中的元素名是预定义的，当两个不同的文档使用相同的元素名时，就会发生命名冲突。

这个 XML 文档携带着某个表格中的信息：

```
<table>
  <tr>
  <td>Apples</td>
  <td>Bananas</td>
  </tr>
</table>
```

这个 XML 文档携带有关桌子的信息（一件家具）：

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

假如这两个 XML 文档被一起使用，由于两个文档都包含带有不同内容和定义的<table>元素，就会发生命名冲突。

18.2 使用前缀来避免命名冲突

此文档带有某个表格中的信息：

```
<h:table>
  <h:tr>
  <h:td>Apples</h:td>
  <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

此 XML 文档携带着有关一件家具的信息：

```
<f:table>
```

```
   <f:name>African Coffee Table</f:name>
   <f:width>80</f:width>
   <f:length>120</f:length>
</f:table>
```

现在，命名冲突不存在了，这是由于两个文档都使用了不同的名称来命名它们的<table>元素（<h:table>和<f:table>）。

通过使用前缀，我们创建了两种不同类型的<table>元素。

18.3 使用命名空间（Namespaces）

这个 XML 文档携带着某个表格中的信息：

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
   <h:tr>
   <h:td>Apples</h:td>
   <h:td>Bananas</h:td>
   </h:tr>
</h:table>
```

此 XML 文档携带着有关一件家具的信息：

```
<f:table xmlns:f="http://www.w3school.com.cn/furniture">
   <f:name>African Coffee Table</f:name>
   <f:width>80</f:width>
   <f:length>120</f:length>
</f:table>
```

与仅仅使用前缀不同，我们为 <table> 标签添加了一个 xmlns 属性，这样就为前缀赋予了一个与某个命名空间相关联的限定名称。

18.4 XML Namespace（xmlns）属性

XML 命名空间属性被放置于某个元素的开始标签之中，并使用以下的语法：

```
xmlns:namespace-prefix="namespaceURI"
```

当一个命名空间被定义在某个元素的开始标签中时，所有带有相同前缀的子元素都会与同一个命名空间相关联。

注释：用于标示命名空间的地址不会被解析器用于查找信息。其惟一的作用是赋予命名空间一个惟一的名称。不过，很多公司常常会作为指针来使用命名空

间指向某个实存的网页，这个网页包含着有关命名空间的信息。

18.5 统一资源标示符（Uniform Resource Identifier（URI））

统一资源标示符是一串可以标示因特网资源的字符。最常用的 URI 是用来标示因特网域名地址的统一资源定位器（URL）。另一个不那么常用的 URI 是统一资源命名（URN）。在我们的例子中，我们仅使用 URL。

18.6 默认的命名空间（Default Namespaces）

为某个元素定义默认的命名空间可以让我们省去在所有的子元素中使用前缀的工作。

请使用下面的语法：

```
xmlns="namespaceURI"
```

这个 XML 文档携带着某个表格中的信息：

```
<table xmlns="http://www.w3.org/TR/html4/">
   <tr>
   <td>Apples</td>
   <td>Bananas</td>
   </tr>
</table>
```

此 XML 文档携带着有关一件家具的信息：

```
<table xmlns="http://www.w3school.com.cn/furniture">
   <name>African Coffee Table</name>
   <width>80</width>
   <length>120</length>
</table>
```

18.7 命名空间的实际应用

当开始使用 XSL 时，您不久就会看到实际使用中的命名空间。XSL 样式表用于将 XML 文档转换为其他格式，比如 HTML。

如果您仔细观察下面的这个 XSL 文档，就会看到大多数的标签是 HTML 标签。非 HTML 的标签都有前缀 xsl，并由此命名空间标示：

"http://www.w3.org/1999/XSL/Transform"：

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
<html>
<body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr>
      <th align="left">Title</th>
      <th align="left">Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

19 XML CDATA

所有 XML 文档中的文本均会被解析器解析。

只有 CDATA 区段（CDATA section）中的文本会被解析器忽略。

19.1 Parsed Data

XML 解析器通常会解析 XML 文档中所有的文本。

当某个 XML 元素被解析时，其标签之间的文本也会被解析：

```
<message>此文本也会被解析</message>
```

解析器之所以这么做是因为 XML 元素可包含其他元素，就像这个例子中，其中的 <name> 元素包含着另外的两个元素(first 和 last)：

```
<name><first>Bill</first><last>Gates</last></name>
```

而解析器会把它分解为像这样的子元素：

```
<name>
  <first>Bill</first>
  <last>Gates</last>
</name>
```

19.2 转义字符

非法的 XML 字符必须被替换为实体引用（entity reference）。

假如您在 XML 文档中放置了一个类似"<"字符，那么这个文档会产生一个错误，这是因为解析器会把它解释为新元素的开始。因此你不能这样写：

```
<message>if salary < 1000 then</message>
```

为了避免此类错误，需要把字符"<"替换为实体引用，就像这样：

```
<message>if salary &lt; 1000 then</message>
```

19.3 CDATA

CDATA 内部的所有东西都会被解析器忽略。

假如文本中包含了大量的"<"和"&"字符，就像编程代码中经常出现的情况一样，那么这个 XML 元素就可以被定义为一个 CDATA 部分。

CDATA 区段开始于"<![CDATA["，结束于"]]>"：

```
<script>
<![CDATA[
function matchwo(a,b)
{
if (a < b && a < 0)
```

```
   {
   return 1
   }
else
   {
   return 0
   }
}
]]>
</script>
```

在上面的例子中，在 CDATA 区段中的所有东西都会被解析器忽略。

19.4 关于 CDATA 区段的注释

CDATA 区段不能包含字符串"]]>"，所以，CDATA 区段的嵌套是不被允许的。

同时也需要确保在"]]>"字符串中没有空格或折行。

20 XML 编码

XML 文档可能包含外国字符，比如挪威语，或者法语。

为了让 XML 解析器读懂这些字符，您需要把 XML 文档存为 Unicode。

20.1 Windows 2000 记事本

Windows 2000 记事本可以把文件保存为 Unicode。

将 XML 文件保存为 Unicode（注意，此文档未包含任何编码属性）：

```
<?xml version="1.0"?>
<note>
  <from>John</from>
  <to>George</to>
  <message>French: êèé</message>
</note>
```

上面的文件，在 IE 5+、Firefox 或者 Opera 中不会出错，但是在 Netscape

6.2 中会出错。

20.2 Windows 2000 Notepad with Encoding

Windows 2000 记事本使用 "UTF-16" 编码将文件保存为 Unicode。

如果您为保存为 Unicode 的 XML 文件添加了一个编码属性，windows 的编码值可能产生错误。

下面的编码，不会报错：

```
<?xml version="1.0" encoding="windows-1252"?>
```

下面的编码，不会报错：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

下面的编码，不会报错：

```
<?xml version="1.0" encoding="UTF-8"?>
```

下面的编码，在 IE 5+、Firefox 或者 Opera 重不会发出错误信息，但是在 Netscape 6.2 中会报错。

```
<?xml version="1.0" encoding="UTF-16"?>
```

20.3 错误消息

如果您试图向 IE 中载入 XML 文档，可能会得到两种指示编码问题的错误：

在文本内容中发现非法字符：

如果 XML 文档中的某个字符与编码属性不匹配 ,您就会得到这个错误消息。通常，当 XML 文件中含有外国字符，且当文件使用类似记事本的单字节编码编辑器保存，以及没有指定编码属性时，您就会得到这个错误消息。

将当前编码切换为不被支持的指定编码：

如果您的文件被保存为 Unicode/UTF-16 ,但是编码属性被指定为单字节编

码（比如 Windows-1252、ISO-8859-1 或者 UTF-8）时，那么您就会得到这个错误消息。或者当您的文档被保存为单字节编码，但编码属性被指定为双字节编码（比如 UTF-16）时，也会得到这个错误消息。

20.4 结论

结论是：编码属性应当被指定为文档被保存时所使用的编码。我最好的避免错误的建议是：

- 使用支持编码的编辑器；
- 确定编辑器使用的编码；
- 在您的 XML 文档中使用相同的编码属性。

21 XML DOM 高级

XML DOM（Document Object Model）定义了访问和操作 XML 文档的标准方法。

21.1 XML DOM

DOM 把 XML 文档视为一种树结构。通过这个 DOM 树，可以访问所有的元素。可以修改它们的内容（文本以及属性），而且可以创建新的元素。元素，以及它们的文本和属性，均被视为节点。

在本教程的较早章节中，我们介绍了 XML DOM，并使用了 XML DOM 的 getElementsByTagName()从 DOM 树中取回数据。

21.2 获取元素的值

下面的代码检索第一个<title>元素的文本值：

```
x=xmlDoc.getElementsByTagName("title")[0].childNodes[0];
```

```
txt=x.nodeValue;
```

结果：txt = "Harry Potter"

21.3 获取属性的值

下面的代码检索第一个<title>元素的"lang"属性的文本值：

```
txt=xmlDoc.getElementsByTagName("title")[0].getAttribute("lang");
```

结果：txt = "en"

21.4 改变元素的值

下面的代码改变第一个<title>元素的文本值：

```
x=xmlDoc.getElementsByTagName("title")[0].childNodes[0];
x.nodeValue="Easy Cooking";
```

21.5 改变属性的值

setAttribute()方法可用于改变已有属性的值，或创建一个新属性。

下面的代码向每个<book>元素添加了名为"edition"的新属性（值是"first"）：

```
x=xmlDoc.getElementsByTagName("book");

for(i=0;i<x.length;i++)
  {
  x[i].setAttribute("edition","first");
  }
```

21.6 创建元素

createElement()方法创建新的元素节点。

createTextNode()方法创建新的文本节点。

appendChild()方法向节点添加子节点（在最后一个子节点之后）。

如需创建带有文本内容的新元素，需要同时创建元素节点和文本节点。

下面的代码创建了一个元素（<edition>）,然后把它添加到第一个<book>元素中：

```
newel=xmlDoc.createElement("edition");
newtext=xmlDoc.createTextNode("First");
newel.appendChild(newtext);

x=xmlDoc.getElementsByTagName("book");
x[0].appendChild(newel);
```

例子解释：

1. 创建<edition>元素；
2. 创建值为"First"的文本节点；
3. 把这个文本节点追加到<edition>元素；
4. 把<edition>元素追加到第一个<book>元素。

21.7 删除元素

removeChild()方法删除指定的节点（或元素）。

下面的代码片段将删除第一个<book>元素中的第一个节点：

```
x=xmlDoc.getElementsByTagName("book")[0];

x.removeChild(x.childNodes[0]);
```

注释：上例的结果可能会根据所用的浏览器而不同。Firefox 把新行字符当作空的文本节点，而 Internet Explorer 不是这样。

22 XML Don't

您应该在使用 XML 时尽量避免本节列出的技术。

22.1 Internet Explorer - XML 数据岛

它是什么？XML 数据岛是嵌入 HTML 页面中的 XML 数据。

为什么要避免使用它？XML 数据岛只在 IE 浏览器中有效。

用什么代替它？您应当在 HTML 中使用 JavaScript 和 XML DOM 来解析并

显示 XML。

22.2 XML 数据岛实例

把 XML 文档绑定到 HTML 文档中的一个<xml>标签。id 属性为数据岛定义标识符，而 src 属性指向 XML 文件：

```
<html>
<body>

<xml id="cdcat" src="cd catalog.xml"></xml>

<table border="1" datasrc="#cdcat">
<tr>
<td><span datafld="ARTIST"></span></td>
<td><span datafld="TITLE"></span></td>
</tr>
</table>

</body>
</html>
```

<table>标签的 datasrc 属性把 HTML 表格绑定到 XML 数据岛。

标签允许 datafld 属性引用要显示的 XML 元素。在这个例子中，要引用的是"ARTIST"和"TITLE"。当 XML 被读取时，会为每个<CD>元素创建额外的行。

22.3 Internet Explorer - 行为

它是什么？Internet Explorer 5 引入了行为（behaviors）。Behaviors 是通过使用 CSS 样式向 XML（或 HTML）元素添加行为的一种方法。

为什么要避免使用它？只有 Internet Explorer 支持 behavior 属性。

使用什么代替它？使用 JavaScript 和 XML DOM（或 HTML DOM）来代替它。

22.4 实例

例子 1 - Mouseover Highlight：

下面的 HTML 文件中的<style>元素为<h1>元素定义了一个行为：

```
<html>
<head>
<style type="text/css">
h1 { behavior: url(behave.htc) }
</style>
</head>

<body>
<h1>Mouse over me!!!</h1>
</body>
</html>
```

以下是 XML 文档"behave.htc"：

```
<attach for="element" event="onmouseover" handler="hig lite" />
<attach for="element" event="onmouseout" handler="low lite" />

<script type="text/javascript">
function hig lite()
{
element.style.color='red';
}

function low lite()
{
element.style.color='blue';
}
</script>
```

这个 behavior 文件包含了一段 JavaScript，以及针对元素的事件句柄。

例子 2 - 打字机模拟：

下面的 HTML 文件中的 <style> 元素为 id 为 "typing" 的元素定义了一个行为：

```
<html>
<head>
<style type="text/css">
#typing
```

```
{
behavior:url(typing.htc);
font-family:'courier new';
}
</style>
</head>

<body>
<span id="typing" speed="100">IE5 introduced DHTML behaviors.
Behaviors are a way to add DHTML functionality to HTML elements
with the ease of CSS.<br /><br />How do behaviors work?<br />
By using XML we can link behaviors to any element in a web page
and manipulate that element.</p>
</span>
</body>
</html>
```

以下是 XML 文档"typing.htc"：

```
<attach for="window" event="onload" handler="beginTyping" />
<method name="type" />

<script type="text/javascript">
var i,text1,text2,textLength,t;

function beginTyping()
{
i=0;
text1=element.innerText;
textLength=text1.length;
element.innerText="";
text2="";
t=window.setInterval(element.id+".type()",speed);
}

function type()
{
text2=text2+text1.substring(i,i+1);
element.innerText=text2;
i=i+1;
if (i==textLength)
  {
  clearInterval(t);
  }
```

```
}
</script>
```

全文完

约 13,000 字（中文字+英文单词）

源自 w3c.org/tutorials

白宇（梦辽软件）整编、校对

2009 年于山西大同