w3school的xml参考手册学习：

XML可以用来描述数据，重点是‘数据是什么’。

HTML则是用来显示数据，重点是‘如何显示数据’。

**什么是XML?**

* XML是E**X**tensible**M**arkup **L**anguage（可扩展的标记语言）的缩写
* XML是一种类似于HTML的标记语言
* XML是用来描述数据的
* XML的标记不是在XML中预定义的，你必须定义自己的标记
* XML使用文档类型定义（DTD）或者模式（Schema：约束）来描述数据
* XML使用DTD或者Schema后就是自描述的语言

HTML是与显示信息相关的, XML则是与描述信息相关的。

XML允许你定义自己的标记以及文档结构。标签不是固定的

在将来的网页开发中，XML将被用来描述、存储数据，而HTML则是用来格式化和显示数据的。

xml只是用来组织、存储和发送信息。

首先要明确的是：XML是被设计用来存储数据、携带数据和交换数据的，他不是为了显示数据而设计的。

XML数据同样可以以“数据岛”的形式存储在HTML页面中。你仍然可以集中精力到使用HTML格式化和显示数据上去。

XML数据是以纯文本格式存储的，那么XML提供了一种与软件和硬件无关的共享数据方法。

XML的语法规则既简单又严格，非常容易学习和使用。

**XML文档使用了自描述的和简单的语法。**

|  |
| --- |
| <?xml version="1.0" encoding="ISO-8859-1"?>  <note>  <to>Lin</to>  <from>Ordm</from>  <heading>Reminder</heading>  <body>Don't forget me this weekend!</body>  </note> |

文档的第1行:XML声明——定义此文档所遵循的XML标准的版本（实际这个xml文件头声明也可以不加），在这个例子里是1.0版本的标准，使用的是ISO-8859-1 (Latin-1/West European)字符集。

文档的第2行是根元素（就像是说“这篇文档是一个便条”）：

|  |
| --- |
| <note> |

文档的第3--6行描述了根元素的四个子节点（to, from, heading,和 body）：

|  |
| --- |
| <to>Lin</to>  <from>Ordm</from>  <heading>Reminder</heading>  <body>Don't forget me this weekend!</body> |

文档的最后一行是根元素的结束：

|  |
| --- |
| </note> |

所有的xml文档必须有一个结束的标记，xml结构语法严格（第一行xml文档的声明不需要结束标记，因为这不是xml文档的一部分），html相对不严格

xml标记是大小写敏感的

因此在XML文档中开始标记和结束标记的大小写必须保持一致。

|  |
| --- |
| <Message>This is incorrect</message> //错误的  <message>This is correct</message> //正确的 |

所有的xml文档必须有一个根元素，xml文档的第一个元素就是根元素；所有的元素都可以有子元素，子元素必须正确的嵌套在父元素中

属性值必须使用双引号”“””，没有双引号的属性值是不符合规定的

如同HTML一样，XML元素同样也可以拥有属性。XML元素的属性以名字/值成对的出现。XML语法规范要求XML元素属性值必须用引号引着。请看下面的两个例子，第一个是错误的，第二个是正确的。

|  |
| --- |
| <?xml version="1.0" encoding="ISO-8859-1"?>  <note date=12/11/99>  <to>Lin</to>  <from>Ordm</from>  <heading>Reminder</heading>  <body>Don't forget me this weekend!</body>  </note> |

|  |
| --- |
| <?xml version="1.0" encoding="ISO-8859-1"?>  <note date="12/11/99">  <to>Tove</to>  <from>Jani</from>  <heading>Reminder</heading>  <body>Don't forget me this weekend!</body>  </note> |

第一个文档的错误之处是属性值没有用引号引着。

正确的写法是: date="12/11/99". 不正确的写法: date=12/11/99.

使用xml文档，空白将被保留，即空白部分不会被解析器自动删除，这一点与html不同：

html中会忽略空白，只保留一个空白，如下：

<a href="http://download.csdn.net/">

首页</a>

xml中空白被保留，做为值的一部分：

<kongbai>hhhh ccc </kongbai>

使用XML, CR –回车/ LF-换行 被转换为 LF（line feed，新行）

xml的注释和html的注释一样：

<!--这是一个注释 -->

XML文档可以被扩展一边携带更多的信息,对于同一个解析器，必须要的信息会被自动忽略

xml文件是可以扩展的，增加了的条目不会影响之前对xml文件的解析。

xml元素之间是父元素和子元素之间的关系：

<book>

<title>**XML 指南**</title>

<prod id="33-657" media="paper"></prod>

<chapter>XML入门简介

<para>什么是HTML</para>

<para>什么是XML</para>

</chapter>

<chapter>XML语法

<para>XML元素必须有结束标记</para>

<para>XML元素必须正确的嵌套</para>

</chapter>

</book>

**XML元素命名**

**XML元素命名必须遵守下面的规则：**

* 元素的名字可以包含子母，数字和其他字符。
* 元素的名字不能以数字或者标点符号开头。
* 元素的名字不能以XML(或者xml，Xml,xMl...)开头。
* 元素的名字不能包含空格。

xml没有保留字除了xml.

非英文/字符/字符串也可以作为XML元素的名字，例如<蓝色理想><经典论坛>这都是完全合法的名字。但是有一些软件不能很好的支持这种命名，所以尽量使用英文字母来命名。

在XML元素命名中不要使用":"，因为XML命名空间需要用到这个十分特殊的字符

xml元素的属性也和html一样在开始标签处。，

元素的属性值必须要使用引号括起来，单引号或者双引号都可以。

什么时候用属性，什么时候用子元素没有一个现成的规则可以遵循。我的经验是属性在HTML中可能相当便利，但在XML中，你最好避免使用。

子元素和属性在大多数时候是可以替换的（单单就显示来说），但多数使用子元素，这样也便于解析。

**应该避免使用属性么？**

这里有一些使用属性引发的问题：

* 属性不能包含多个值（子元素可以）。
* 属性不容易扩展。
* 属性不能够描述结构（子元素可以）。
* 属性很难被程序代码处理。
* 属性值很难通过DTD进行测试。

当属性是某个元素的唯一标识符时，可以使用属性，就像html中的id和name等属性；这样便于dom操作时引用这个xml元素；也可以在数组标签中用来区分数组元素。

元数据（与数据有关的数据）应该以属性的方式存储，而数据本身应该以元素的形式存储。

接口良好的xml文档：指的是符合语法的xml文档，如：

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

<note>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

有效的xml文档：指的是通过DTD描述验证的xml文档，如：

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

<!DOCTYPE note SYSTEM "InternalNote.dtd">

<note>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

DTD:文档类型定义（Document Type Definition）

**DTD定义了XML文档中可用的合法元素。**

DTD的意图在于定义XML文档的合法建筑模块。他通过定义一系列合法的元素决定了XMl文档的内部结构。结构良好的XML文档不一定是有效的XML文档,但有效的XML文档一定是结构良好的XML文档。如果你想了解更多关于DTD的知识可以参考[DTD 指南](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/dtd/index.htm)。

## XML Schema

**XML Schema(XML约束)是基于XML的DTD的替代品。**

W3C使得DTD和Schema可以相互替代，读者可以在[Schema 指南](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/schema/index.htm)中了解到更多信息。

可以使用css来修饰xml文档在浏览器中的显示，但是不推荐这样做。

一个使用css修饰xml的简单例子：

<?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>

还可以使用xsl来为xml文件增加显示信息：

## 使用XSL来显示XML文档

**XSL是XML的可扩展样式单语言。**

XSL (全称是：可扩展的样式单语言，the eXtensible Stylesheet Language)比CSS样式单功能要强大的多。XSL的一个主要的用途就是将XML文档转换成HTML格式的文件，然后再交付给浏览器，由浏览器显示转换的结果:

包含xsl语言的例子：

<?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>

xml数据可以作为数据岛嵌入到html文件中；而且xml可以在html页面中做数据绑定。

<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>

xml支持dom操作。

# XML 命名空间

XML 命名空间提供了一种避免元素命名冲突的方法。

## 命名冲突

因为XML文档中使用的元素不是固定的，那么两个不同的XML文档使用同一个名字来描述不同类型的元素的情况就可能发生。而这种情况又往往会导致命名冲突。请看下面两个例子

这个 XML 文档在table元素中携带了水果的信息:

|  |
| --- |
| <table>  <tr>  <td>Apples</td>  <td>Bananas</td>  </tr>  </table> |

这个 XML 文档在table元素中携带了桌子的信息(家具，不能吃的哦):

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

如果上面两个XML文档片断碰巧在一起使用的话，那么将会出现命名冲突的情况。因为这两个片断都包含了<table>元素，而这两个table元素的定义与所包含的内容又各不相同。

## 使用前缀解决命名冲突问题

下面的XML文档在table元素中携带了信息:

|  |
| --- |
| <h:table>  <h:tr>  <h:td>Apples</h:td>  <h:td>Bananas</h:td>  </h:tr>  </h:table> |

下面的XML文档携带了家具table的信息:

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

现在已经没有元素命名冲突的问题了，因为这两个文档对各自的table元素使用了不同的前缀（语法：前缀:标签名），table元素在两个文档中分别是(<h:table> 和<f:table>)。

通过使用前缀，我们创建了两个不同的table元素。

## 使用命名空间

下面的XML文档在table元素中携带了信息:

|  |
| --- |
| <h:tablexmlns:h="http://www.w3.org/TR/html4/">  <h:tr>  <h:td>Apples</h:td>  <h:td>Bananas</h:td>  </h:tr>  </h:table> |

下面的XML文档携带了家具table的信息:

|  |
| --- |
| <f:tablexmlns:f="http://www.w3schools.com/furniture">  <f:name>African Coffee Table</f:name>  <f:width>80</f:width>  <f:length>120</f:length>  </f:table> |

在上面两个例子中除了使用前缀外，两个table元素都使用了**xmlns**属性，使元素和不同的命名空间关联到一起。

## 命名空间属性

命名空间属性一般放置在元素的开始标记处，其使用语法如下所示:

|  |
| --- |
| xmlns:namespace-prefix="namespace" |

在上面的例子中，命名空间定义了一个Internet 地址:

|  |
| --- |
| xmlns:f="http://www.w3schools.com/furniture" |

W3C 命名规范声明命名空间本身就是一个统一资源标示符，**Uniform Resource Identifier (URI)**。

当我们在元素的开始标记处使用命名空间时，该元素所有的子元素都将通过一个前缀与同一个命名空间相互关联。

注意：用来标识命名空间的网络地址并不被XML解析器调用，XML解析器不需要从这个网络地址中查找信息，该网络地址的作用仅仅是给命名空间一个唯一的名字，因此这个网络地址也可以是虚拟的，然而又很多公司经常把这个网络地址值象一个真实的Web页面，这个地址包含了关于当前命名空间更详细的信息。  
可以访问<http://www.w3.org/TR/html4/>.

## 默认的命名空间

定义一个默认的XML命名空间使得我们在子元素的开始标记中不需要使用前缀。他的语法如下所示:

|  |
| --- |
| <element xmlns="namespace"> |

下面的XML文档在table元素中包含了水果的信息:

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

下面的XML文档包含了家具table的信息:

|  |
| --- |
| <table xmlns="http://www.w3schools.com/furniture">  <name>African Coffee Table</name>  <width>80</width>  <length>120</length>  </table> |

## 使用命名空间

当开始使用XSL的时候,就会发现命名空间使用的是如此频繁。XSL样式单主要用于将XML文档转换成类似于HTML文件的格式。

如果看一下下面的XSL文档，就会发现有很多标记都是HTML标记。那些标记并不是HTML标记，是加了前缀的XSL,这个XSL前缀由命名空间"<http://www.w3.org/TR/xsl>"所标识:

|  |
| --- |
| <?xml version="1.0" encoding="ISO-8859-1"?>  <xsl:stylesheetxmlns:xsl="http://www.w3.org/TR/xsl">  <xsl:template match="/">  <html>  <body>  <table border="2" bgcolor="yellow">  <tr>  <th>Title</th>  <th>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> |

XML CDATA

在XML文档中的所有文本都会被解析器解析。

只有在CDATA部件之内的文本会被解析器忽略。

## 解析数据

**XML 解析器通常情况下会处理XML文档中的所有文本。**

当XML元素被解析的时候，XML元素内部的文本也会被解析:

|  |
| --- |
| <message>This text is also parsed</message> |

XML解析器这样做的原因是XML元素内部可能还包含了别的元素，像下面的例子，name元素内部包含了first和last两个元素:

|  |
| --- |
| <name><first>Bill</first><last>Gates</last></name> |

解析器会认为上面的代码是这样的，一级一级的解析最后得到xml的整体结构:

|  |
| --- |
| <name>  <first>Bill</first>  <last>Gates</last>  </name> |

## 转义字符

**不合法的XML字符必须被替换为相应的实体。**

如果在XML文档中使用类似"<" 的字符, 那么解析器将会出现错误，因为解析器会认为这是一个新元素的开始。所以不应该像下面那样书写代码:

|  |
| --- |
| <message>if salary < 1000 then</message> |

为了避免出现这种情况，一种方法是必须将字符"<" 转换成实体，像下面这样:

|  |
| --- |
| <message>if salary &lt; 1000 then</message> |

下面是五个在XML文档中预定义好的实体:

|  |  |  |
| --- | --- | --- |
| &lt; | < | 小于号 |
| &gt; | > | 大于号 |
| &amp; | & | 和 |
| &apos; | ' | 单引号 |
| &quot; | " | 双引号 |

实体必须以符号"&"开头，以符号";"结尾**。**

**注意:** 只有"<" 字符和"&"字符对于XML来说是严格禁止使用的。剩下的都是合法的，为了减少出错，使用实体是一个好习惯。

## CDATA部件

**在CDATA内部的所有内容都会被解析器忽略。**

如果文本包含了很多的"<"字符和"&"字符——就像象程序代码一样，那么最好把他们都放到CDATA部件中。

一个 CDATA 部件以"**<![CDATA[**" 标记开始，以"**]]>**"标记结束:

|  |
| --- |
| <script>  <![CDATA[  function matchwo(a,b)  {  if (a < b && a < 0) then  {  return 1  }  else  {  return 0  }  }  ]]>  </script> |

在前面的例子中，所有在CDATA部件之间的文本都会被解析器忽略。

### CDATA注意事项:

CDATA部件之间不能再包含CDATA部件（不能嵌套）。

如果CDATA部件包含了字符"]]>" 或者"<![CDATA[" ，将很有可能出错哦。

同样要注意在字符串"]]>"之间没有空格或者换行符。

# XML编码

XML文档可以包含外文字符比如挪威语或者法语（中文当然也可以！这一部分还是不能照原文翻译，下面有些内容是我自己写的）

为了让你的解析器能够明白这些字符，你必须在XML文档中统一字符编码标准。

## Windows 95/98 记事本

**Windows 95/98 记事本不能以Unicode的编码格式保存文件。**

可以使用记事本来编辑和保存包含了外文字符的XML文档 (比如：挪威语或者法语或者中文)

|  |
| --- |
| <?xml version="1.0"?>  <note>  <from>小奀</from>  <to>小林</to>  <message>晚上一起去火锅呀</message>  </note> |

但是如果你用浏览器打开这个用记事本编辑过的XML文档，将会出现一个错误[open it with IE 5.0](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_none.xml)。

## 在Windows 95/98 记事本中使用编码

**Windows 95/98 记事本编辑XML文件必须进行编码属性设置。**

为了避免发生错误，可以在XML文档声明中加上一个encoding属性，指明此XML文档的编码类型，但是不要使用Unicode编码。

下面的编码类型不会导致错误，([用IE 5.0+浏览器观看](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_2312.xml)), 并且汉字显示正常:

|  |
| --- |
| <?xml version="1.0" encoding="gb2312"?> |

下面的编码类型不会导致错误，([用IE 5.0+浏览器观看](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_gbk.xml)), 并且汉字显示正常:

|  |
| --- |
| <?xml version="1.0" encoding="gbk"?> |

下面的编码类型不会导致错误，([用IE 5.0+浏览器观看](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_1252.xml)), 并且汉字显示不正常(乱码):

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

下面的编码类型不会导致错误，([用IE 5.0+浏览器观看](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_8859.xml)), 并且汉字显示不正常(乱码):

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

下面的编码类型不会导致错误，([用IE 5.0+浏览器观看](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_utf8.xml)),并且汉字显示正常:

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

下面的编码类型会导致错误，([用IE 5.0+浏览器观看](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_utf16.xml)):

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

## 使用Windows 2000 记事本

**Windows 2000 记事本可以以Unicode编码格式保存文件。**

Windows 2000 的记事本支持Unicode字符集。如果使用Win2000的记事本以Unicode的编码格式保存XML文档(请注意这里在XML声明中没有编码信息):

|  |
| --- |
| <?xml version="1.0"?>  <note> <from>小奀</from> <to>小林</to> <message>晚上一起去火锅呀</message> </note> |

下面这个文件; [note\_encode\_none\_u.xml](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_none_u.xml), 在IE5.0+中将不会出现错误，但如果你使用网景Netscape 6.2，就可能出错。读者比较 [note\_encode\_none.xml](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_none.xml) 和 [note\_encode\_none\_u.xml](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_none_u.xml)这两个文件，如果单独看他们各自的源文件，是没有区别的，但为什么一个能显示，一个不能显示呢？答案就是Unicode字符集。

## Windows 2000 记事本编码

**Windows 2000记事本还可以以"UTF-16"编码格式保存文件。**

如果你在XML文档中声明了编码属性而又以Unicode编码格式保存文件，将可能发生错误。

下面的代码将会导致错误 ([用浏览器察看](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_1252_u.xml)):

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

下面的代码将会导致错误 ([用浏览器察看](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_8859_u.xml)):

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

下面的代码将会导致错误 ([用浏览器察看](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_utf8_u.xml)):

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

下面这个文件; [note\_encode\_utf16\_u.xml](mk:@MSITStore:E:\deleting\xml_study\XML_manual_cn.chm::/xml_cn/note_encode_utf16_u.xml), 将会在IE5.0+中显示正常，在网景Netscape 6.2浏览器中将会出现错误。

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

## 错误信息

当使用IE5.0或者更高版本浏览XML文档时，可能会遇到两中不同的编码错误:

**在文本内容中发现一个非法字符(An invalid character was found in text content)。**

如果你的XML文档和你的XML文档的编码格式不匹配就可能导致错误发生。通常情况是，XML文档包含一些“非英文”字符，并且使用了单字节编码的编辑器，而且没有在XML文档的声明中设置XML文档的编码格式。   
  
**不支持从当前的编码格式转换成另一种编码格式（Switch from current encoding to specified encoding not supported）。**

如果XML文档以Unicode/UTF-16编码格式保存，但是XML文档的声明中却设置了编码格式是一些单字节的编码（比如 Windows-1252, ISO-8859-1 或者  UTF-8）；或者XML文档以单字节编码格式保存，但是XML文档的声明中却设置了编码格式是一些Unicode/UTF-16的编码形式，这样两种情况都会导致错误发生。

## 结论

结论:在保存XML文档之前在XML文档的声明中设置文档的编码格式，我的一些关于避免错误发生的建议:

* **使用一种支持Unicode编码格式的编辑器。**
* **确信你知道自己正在使用那种编码格式。**
* **在XML文档中使用属性声明设置编码格式（encoding）。**