**Introduction to XML**

(source : http://www.xmlnews.org/docs/xml-basics.html)

XML (Extensible Markup Language) is a popular and widely-implemented standard: you can use XML to create documents and data records that are fully portable and platform-independent. Both the major XMLNews formats, [XMLNews-Story](http://www.xmlnews.org/docs/xmlnews-story.html) and [XMLNews-Meta](http://www.xmlnews.org/docs/xmlnews-meta.html), are based on XML.

This tutorial provides a brief overview of XML to help you become familiar with the markup language's most common features. After reading the tutorial, you should be able to read examples that use XML syntax and understand the basic structure of an XML document. The tutorial concentrates on the technical rather than the business side of XML, and is aimed at technical specialists such as software engineers and documentation writers who are approaching XML for the first time.

Although XML 1.0 is not a complicated format, there are many more details (and much terminology) that this tutorial does not cover. If you are planning to implement software that reads or writes XML directly (rather than through a specialized library), then you will need to refer to the [XML 1.0 Recommendation](http://www.w3.org/TR/REC-xml), which is available online and free of charge from the [World Wide Web Consortium](http://www.w3.org/): the Recommendation is the single authoritative source for all XML work.

1. Markup and Text

Here's a complete (but very simple) XML document:

<?xml version="1.0"?>

<contact-info>

<name>Jane Smith</name>

<company>AT&amp;T</company>

<phone>(212) 555-4567</phone>

</contact-info>

There are two different kinds of information in this example:

1. **markup**, like “<contact-info>” and “&amp;”; and
2. **text** (also known as **character data**), like “Jane Smith” and “(212) 555-4567”.

XML documents mix markup and text together into a single file: the markup describes the structure of the document, while the text is the document's content (actually, sometimes markup can also represent content, as in the case of [references](http://www.xmlnews.org/docs/xml-basics.html#references): more on this point below). Here's the same XML document again, with the markup highlighted to distinguish it from the text:

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

**<contact-info>**

**<name>**Jane Smith**</name>**

**<company>**AT**&amp;**T**</company>**

**<phone>**(212) 555-4567**</phone>**

**</contact-info>**

The rest of this tutorial shows you how to use different kinds of markup and text in an XML document:

* [the XML declaration](http://www.xmlnews.org/docs/xml-basics.html#xmldecl);
* [tags and element](http://www.xmlnews.org/docs/xml-basics.html#elements);
* [attributes](http://www.xmlnews.org/docs/xml-basics.html#attributes);
* [references](http://www.xmlnews.org/docs/xml-basics.html#references); and
* [text](http://www.xmlnews.org/docs/xml-basics.html#text).

2. The XML Declaration

All XML documents can optionally begin with an **XML declaration**. The XML declaration provides at a minimum the number of the version of XML in use:

<?xml version="1.0"?>

Currently, 1.0 is the only approved version of XML, but others may appear in the future.

The XML declaration can also specify the character encoding used in the document:

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

All XML parsers are required to support the Unicode “UTF-8” and “UTF-16” encodings; many XML parser support other encodings, such as “ISO-8859-1”, as well.

There a few other important rules to keep in mind about the XML declaration:

1. the XML declaration is case sensitive: it may not begin with “<?XML” or any other variant;
2. if the XML declaration appears at all, it must be the very first thing in the XML document: not even whitespace or comments may appear before it; and
3. it is legal for a transfer protocol like HTTP to override the encoding value that you put in the XML declaration, so you cannot guarantee that the document will actually use the encoding provided in the XML declaration.

3. Tags and elements

XML tags begin with the less-than character (“<”) and end with the greater-than character (“>”). You use tags to mark the start and end of **elements**, which are the logical units of information in an XML document.

An element consists of a **start tag**, possibly followed by text and other complete elements, followed by an **end tag**. The following example highlights the tags to distinguish them from the text:

**<p><person>**Tony Blair**</person>** is **<function>**Prime

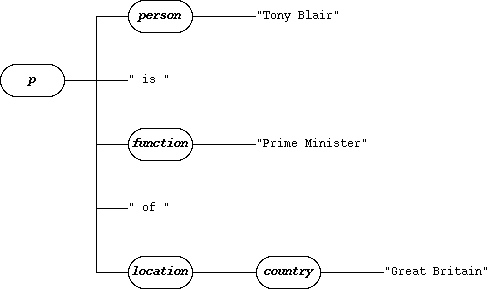
Minister**</function>** of **<location><country>**Great

Britain**</country></location></p>**.

Note that the end tags include a solidus (“/”) before the element's name. There are five elements in this example:

1. the *p* element, that contains the entire example (the *person* element, the text “ is ”, the *function* element, the text “ of ”, and the *location* element);
2. the *person* element, that contains the text “Tony Blair”;
3. the *function* element, that contains the text “Prime Minister”;
4. the *location* element, that contains the *country* element; and
5. the *country* element, that contains the text “Great Britain”.

The following illustration shows this structure as a tree, with *p* (the outermost element) at the root:



There are a few rules to keep in mind about XML elements:

1. Elements may not overlap: an end tag must always have the same name as the most recent unmatched start tag. The following example is not well-formed XML, because “</person>” appears when the most recent unmatched start tag was “<function>”:
2. <!-- WRONG! -->
3. **<function><person>**President**</function>** Habibe<**/person>**

The following example shows the tags properly nested:

**<person><function>**President**</function>** Habibe**</person>**

1. An XML document has *exactly* one root element. As a result, the following example is not a well-formed XML document, because both the *a* and *b* elements occur at the top level:
2. <!-- WRONG! -->
3. <a>...</a>
4. <b>...</b>

The following example fixes the problem by including both the *a* and *b* elements within a new *x* root element:

<x>

<a>...</a>

<b>...</b>

</x>

1. XML element (and attribute) names are case-sensitive, so “location” and “Location” refer to different elements. This is a very nasty trap for people used to working with HTML or other SGML document types, because it can cause surprising bugs in processing software, or can even lead to malformed XML documents, as in the following example:
2. <!-- WRONG! -->
3. <a href="pbear.html">polar bear</A>

This example will cause a parser error because an XML processor considers *a* and *A* to be separate elements, so the start and end tags do not match.

In some cases, an element may exist that has no content (for example, the HTML *hr* element), but the tag is still read by processors. Rather than type a start and end tag with nothing between them (for example, “<hr></hr>”), XML has a special **empty-element tag** that represents both the start tag and the end tag:

<p>Stuff<hr/>

More stuff.</p>

In this example, “<hr/>” represents both the start and the end of the *hr* element; it could just as easily have been written as “<hr></hr>” (which is exactly equivalent).

4. Attributes

In addition to marking the beginning of an [element](http://www.xmlnews.org/docs/xml-basics.html#elements), XML start tags also provide a place to specify **attributes**. An attribute specifies a single property for an element, using a name/value pair. One very well known example of an attribute is *href* in HTML:

<a href="http://www.yahoo.com/">Yahoo!</a>

In this example, the content of the *a* element is the text “Yahoo!”; the attribute *href* provides extra information about the element (in this case, the Web page to load when a user selects the link).

Every attribute assignment consists of two parts: the **attribute name** (for example, *href*), and the **attribute value** (for example, http://www.yahoo.com/). There are a few rules to remember about XML attributes:

1. Attribute names in XML (unlike HTML) are case sensitive: *HREF* and *href* refer to two different XML attributes.
2. You may not provide two values for the same attribute in the same start tag. The following example is not well-formed because the *b* attribute is specified twice:
3. <a b="x" c="y" b="z">....</a>
4. Attribute names should never appear in quotation marks, but attribute values must always appear in quotation marks in XML (unlike HTML) using the " or ' characters. The following example is not well-formed because there are no delimiters around the value of the *b* attribute:
5. <!-- WRONG! -->
6. <a b=x>...</a>

You can use the pre-defined entities “&quot;” and “&apos;” when you need to include quotation marks within an attribute value (see [References](http://www.xmlnews.org/docs/xml-basics.html#references) for details).

Some attributes have special constraints on their allowed values: for more information, refer to the documentation provided with your document type.

5. References

A **reference** allows you to include additional text or markup in an XML document. References always begin with the character “&” (which is specially reserved) and end with the character “;”.

XML has two kinds of references:

**entity references**

An entity reference, like “&amp;”, contains a name (in this case, “amp”) between the start and end delimiters. The name refers to a predefined string of text and/or markup, like a macro in the C or C++ programming languages.

**character references**

A character references, like “&#38;”, contains a hash mark (“#”) followed by a number. The number always refers to the Unicode code for a single character, such as 65 for the letter “A” or 233 for the letter “�”, or 8211 for an en-dash.

For advanced uses, XML provides a mechanism for declaring your own entities, but that is outside the scope of this tutorial. XML also provides five pre-declared entities that you can use to escape special characters in an XML document:

| **Character** | **Predeclared Entity** |
| --- | --- |
| **&** | &amp; |
| **<** | &lt; |
| **>** | &gt; |
| **"** | &quot; |
| **'** | &apos; |

For example, the corporate name “AT&T” should appear in the XML markup as “AT&amp;T”: the XML parser will take care of changing “&amp;” back to “&” automatically when the document is processed.

6. Text

If you are working with 8-bit characters, you can usually type printing characters from the 7-bit (non-accented) US-ASCII character set directly into an XML document, except for the special characters “<” and “&”, and sometimes, “>” (it's best to escape it as well just to be safe). Whenever you need to include one of these three characters in the text of an XML document, simply escape it using an entity reference as described in the [References](http://www.xmlnews.org/docs/xml-basics.html#references) section:

<formula>x &lt; (x + 1)</formula>

For “<”, use “&lt;”, for “&”, use “&amp;”, and for “>”, use “&gt;”.

Above character position 127, things become a little trickier on some systems, because by default XML uses UTF-8 for 8-bit character encoding rather than ISO-8859-1 (Latin Alphabet # 1), which HTML and many computer operating systems use by default. UTF-8 and ISO-8859-1 are both essentially identical with US-ASCII up to position 127; for higher characters (those with accents), UTF-8 uses multi-byte escape sequences.

That means that in a UTF-8 XML document, you cannot simply use a single byte with decimal value 233 to represent “�” (and there is no predefined &eacute; entity as there is in HTML); instead, you must either enter the UTF-8 multi-byte escape sequence, or use a special kind of XML reference called a**character reference**:

<p>That is everyone's favourite caf&#233;.</p>

When your text consists primarily of unaccented Roman characters, this is often the easiest way to escape the occasional accented or non-Roman character. Since “�” appears at position 233 in Unicode (as in ISO-8859-1), the XML parser will read the string correctly as “That is everyone's favourite caf�.”