

XML BASICS*

Connexions
Sarah Coppin
Brent Hendricks
Chuck Bearden

This work is produced by OpenStax-CNX and licensed under the
Creative Commons Attribution License 1.0[†]

Abstract

This module describes XML (eXtensible Markup Language) and the rules that govern its usage. It also explains what a well-formed and valid document is.

1 What is XML?

The eXtensible Markup Language (**XML**) is a **meta-markup language** defined by the World Wide Web Consortium (W3C)¹. It is not strictly a markup language itself, but rather a set of rules for creating markup languages. For our purposes a **markup language** is any language (HTML, for example) that uses tags surrounding text to convey information such as content or format. CNXML², the markup language used by the Connexions Project³ is an example of a language written in XML. There are many other examples at the W3C site. Here is an example of some markup in CNXML.

Example 1

<para>

This is a paragraph in <term>CNXML</term>. Notice that the markup contains tags that express the meaning of the text.

</para>

<para> and </para> are the tags that enclose the text. In XML, tags are always marked by angle brackets (also known as < and >). **Tags** generally come in pairs. An opening tag will look like <tagname>. A closing tag will look like </tagname>, with a / preceding the tag name.

XML allows the separation of presentation from content. For example, HTML has tags such as <u> and <i>, which underline and italicize text respectively. This does not express content information, only

*Version 2.24: Jul 10, 2009 4:21 pm -0500

[†]<http://creativecommons.org/licenses/by/1.0>

¹<http://www.w3.org>

²<http://cnx.rice.edu/cnxml>

³<http://cnx.rice.edu>

formatting. XML allows you to define your own language of tags to represent content. You could create a tag called `<book>` to represent book titles, and create a stylesheet (a separate formatting document), that says that every `<book>` tag should be italicized or underlined. Then when you want to change the presentation of that type of content, you just change one small part of the stylesheet. Also, if you make tags that convey the content of the document, you can enable better searching. For example, you might look for the author of a document by looking at the author tag.

2 Well-formed XML

XML has a few rules that apply to all of its languages, including CNXML. If a document satisfies these rules, then it is **well-formed**. XML documents are required to be well-formed.

- Every tag that is opened must be closed. An opening tag looks like `<module>` and a closing tag looks like `</module>`. There is a shortcut. If your tag contains no other tags (referred to as an **empty tag**), then you can type a `/` before the end of the opening tag and delete the closing tag. For example, `<media> </media>` can be abbreviated `<media/>`.
- Tags must be nested within each other. So, `red <i>and</i> blue` is fine, but `red <i>and blue</i>` is incorrect because the `` and `<i>` tags have overlapping content.
- You must put either single or double quotes around an attribute value. An **attribute** is some sort of information that is associated with a tag and is listed inside of the tag itself. For example, `<module id="m0001">` and `<module id='m0001'>` are fine, but `<module id=m0001>` is incorrect.
- You can also choose to start every document with an **XML declaration**. If you do use the XML declaration, then it has to be the very first thing in the file. It cannot even be preceded by whitespace. It is not considered to be a tag. The XML declaration is as follows. `<?xml version="1.0"?>` You can also include other information such as the encoding of the document or whether the document depends on other files or not.
- There must be one tag that contains all of the other tags. For example in xhtml `<html>` and `</html>` must surround all of the other tags. There are some things that are included at the top of the document that are not tags and that are not included with the tags. The XML declaration is an example of this.

3 Valid XML

It is possible to define a set of rules that apply to all of the tags in a particular XML language. These rules can be defined in a couple of different ways. The most common way is to use a **DTD** (Document Type Definition). Any document which follows all of the rules for that language is called **valid**. A document is not required to be valid in order to be XML. However, it is generally a good idea.

4 Entity References

NOTE: Entity references are no longer supported by CNXML 0.6. Instead, we suggest that you use character references as described below (Section 5: Character References) to add special characters to your module.

XML uses several characters in special ways as part of its markup, in particular the less-than symbol (`<`), the greater-than symbol (`>`), the double quotation mark (`"`), the apostrophe (`'`), and the ampersand (`&`). You've already seen examples of markup using the first four of those previously in this module. But what if you need to these characters in your content, and you don't want them to be treated as part of the markup by XML processors? You can use XML **entity references** for this purpose. The XML Specification defines the following five entity references for use in any well-formed XML document:

- `&` refers to an ampersand (`&`)
- `<` refers to a less-than symbol (`<`)

- `>`; refers to a greater-than symbol (>)
- `"`; refers to a double-quote mark (")
- `'`; refers to an apostrophe (')

Example 2

Suppose you have a document with the following:

```
<para id="p1">The firm was known as Scrooge and Marley.</para>
```

you could replace 'and' with the entity reference `&`:

```
<para id="p1">The firm was known as Scrooge &amp; Marley.</para>
```

All entity references outside the above five must be defined in a document type declaration, and they may only be used in documents that conform to that DTD. Note that an entity reference always begins with `&` and ends with `;`.

5 Character References

You can also use any character defined in **Unicode** in an XML document by means of **character references**. Unicode is a project to define a unique code for every character in any human language. Unicode is very useful any time that you need to use a symbol that is not a part of ASCII.

Character references in XML either begin with `&#`, or they begin with `&#x`, and they end with a semicolon `;`. A character reference contains a representation of a Unicode code point: if it begins with `&#`, then it contains a decimal representation of a Unicode code point; if it begins with `&#x`, then it contains a hexadecimal representation of a Unicode code point.

Example 3

The hexadecimal representation of the Unicode code point for the small 'o' with a stroke is 00F8, and the decimal representation for the same is 248. Therefore, the character references for the small 'o' with a stroke are `ø` and `ø`; So you could write

```
<emphasis>The majestik m&#x00F8;&#x00F8;se</emphasis>
```

or

```
<emphasis>The majestik m&#248;&#248;se</emphasis>
```

or even

```
<emphasis>The majestik m&#x00F8;&#248;se</emphasis>
```

to get

The majestik møøse