

A standard flexible document exchange format, XML - Legend Scrolls



A standard flexible document exchange format, XML

Release: 2008-01-27

Table Of Contents

- [XML](#)
- [Use more than one XML structure in the same document with XML Namespaces](#)
- [XML 1.1](#)



A way to create flexible Markup Languages based on the experience of SGML and HTML produced XML (eXtensible Markup Language). The strict structure rules of XML include normal elements must have a start and end tag, empty elements are defined with a slash before the ending pointy bracket as `<linebreak/>` (empty elements can be a normal element too as long as there is absolutely nothing for the element content), attribute values must always be quoted and comments must only be used within its own special empty element: `<!-- An XML comment -->`. Also as most SGML documents like HTML are case-insensitive: don't care if the element or attribute names are uppercase, lowercase or mixed, XML does care about the casing: is case sensitive. Each element cannot have more than one attribute with the same name and an entity reference must be any of the three forms Named Entity, Hex Entity or Numbered Entity such as `'` (if defined), `'`, `'`. This allows a flexible construct to create markup languages where the XML Author can define their own XML Document elements and attributes. This is the foundation of XML.

Other features that XML brings include native support for Unicode and new special elements such as Processing Instructions (`<?appName attributeOrScripting ?>`) and Character Data Sections (`<![CDATA[This is a character data section.]]>`) that surround the element content to allow certain characters to be as themselves where otherwise would be interpreted as part of the markup and so XML would throw errors at you.

All XML parsers (processors) have a built-in Document Well-Formed Validator in which it checks your XML document against the strict structure rules. Some characters to keep in mind are the quotes in attribute values and the start pointy bracket (`<`) and the ampersand (`&`) in both attribute values, element content and comments: attribute values are commonly quoted with the double quote (`"`) and so you need the named entity, `"`; within the attribute value if you need the double quote in there. Otherwise the XML parser will think you are ending the attribute value before you intended and the rest of the value will cause a Document Well-formed error. But single quotes or apostrophes (`'`) can be used as themselves when double quotes surround the value. Similar for the less used values surrounded by single quotes: double quotes can be used as themselves but single quotes or apostrophes must use the named entity, `'`; first introduced by XML and not available in HTML (`'` hex entity could be used by both HTML and XML as all hex and numbered entities are supported by both).

Start pointy brackets (`<`) are used as part of start and end tags for an element and part of empty elements and other special element markup so in order to use it as itself you need to use the `<` named entity within attribute values,

A standard flexible document exchange format, XML - Legend Scrolls

element content and comments. Ampersands are used as part of Entities such as `"`; so to use it as itself you need the `&` named entity.

As stated earlier when element content is surrounded by a CDATA section these character rules are relaxed so you can use `<` and `&` as they are as long as you don't have `]]>` characters together within the Section as this will end the CDATA section. The named entity `>` can be used instead of the literal end pointy bracket (`>`) but it does not pose any problems towards Document Well-Formedness.

The first element of an XML document is also referred to as the Document Element. An XML Document usually also has an XML Declaration but isn't required. The XML Declaration is a Processing Instruction where the `appName` or target application is `'xml'` and has a version, encoding and possibly a standalone attributes:

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

The encoding dictates which character set the characters are going to be from. By default XML Documents use UTF-8 (8-bit Universal Character Set Transfer Format) which takes advantage of the Unicode support. This attribute is optional.

standalone attribute dictates if the document has any associated documents (such as stylesheets can be used to present the document or a DTD or XML Schema is used to validate the document structure at an XML Language level beyond the Well-Formed Rules). This optional attribute has the values `'yes'` or `'no'` where `'no'` is the default.

XML also provides a few predefined attributes that you can use on any of your XML Documents:

- `xml:lang` provide a global language selector such as `xml:lang="en-GB"` or `xml:lang="en"`, etc.
- `xml:space` has the values of `'default'` (which is the default value) which removes leading and trailing spaces, tabs, newlines and other `'whitespace'` characters from attribute values and element content and reduces two or more whitespace characters between words down to a single whitespace. `'preserve'` is the other value used to keep the spacing. This whitespace handling is for the XML Parser only and not for presentational purposes (CSS has the presentational version covered with the `white-space` property).
- `xml:base` takes a [URI](#) to help construct absolute URIs for relative links and other relative URIs - like the HTML `<base href="...">` element except as it is an attribute it can be used on any element and so any part of the document and any number of `xml:base` attributes can be used.
- `xml:id` is a global unique identifier type attribute.

These attributes can be used on any element or restricted by some form of document validation.

A standard flexible document exchange format, XML - Legend Scrolls

A note on unique identifier (ID) type attributes such as [xml:id](#) or [id](#) from other XML languages: an element can have only one ID type attribute and the value must be unique through the whole document. This is the same for HTML's [id](#) attribute. ID type attribute values can only start with a letter, underscore ([_](#)) or a colon([:](#)) and then as many letters, underscores, colons, numbers, dashes([-](#)) and other characters.

XML has a couple of MIME Media Types that all XML based languages may use: [text/xml](#) and [application/xml](#) which is used in preference over [text/xml](#) these days. Most XML based languages have their own MIME Media Type following the form of [application/xmllanguage+xml](#).

Use more than one XML structure in the same document with XML Namespaces

More than one XML based language may have elements or attributes of the same name. This is fine if you are isolating each XML structure into its own document. But one reason why people and companies like to use XML is to build documents with more than one XML based language within it as each XML language may describe specific things such as 2-d images, math, hyperlinks, etc. In order to uniquely identify which XML based language an element or attribute is from, most XML languages these days have their own XML Namespace.

An XML Namespace is generally a [URI](#) which is associated with an XML Prefix. The Namespaces are declared by an [xmlns:yourprefix](#) attribute usually within the document element but can be in any element. Then the prefix [yourprefix:](#) is prefixed to the element or attribute. Such as:

```
<?xml version="1.0" encoding="UTF-8"?>
<ex:doc xmlns:ex="http://example.com/namespace" xmlns:c="http://www.c.org/charl"
xml:lang="en">
  <ex:chapter number="1" c:signed="me">
    <ex:page number="1">
      <c:emerald name="sam"/>
      <ex:title>A title</ex:title>
      <ex:text c:style="fancyEnglish">blah blah doc
        blah some text blah blah watch out for orbs blah blah
      </ex:text>
    </ex:page>
  </ex:chapter>
</ex:doc>
```

As you can see the main elements are bound to the [http://example.com/namespace](#) namespace by the [ex](#) prefix. The attributes that are not prefixed in an element are automatically bound to the same namespace as the element. Plus the extra elements and attributes (that are on a different namespaced element) are bound to the [http://www.c.org/charl](#)

A standard flexible document exchange format, XML - Legend Scrolls

namespace by the `c` prefix. Each XML based language has a fixed Namespace URI and most have a typical prefix but no prefix is hard wired so you can use your own.

The original XML Namespace which uses `xml` as the prefix is automatically declared by all namespace-aware XML parsers and the use of the original XML Namespace is restricted to only the formally published XML Attributes currently: `xml:lang`, `xml:space`, `xml:base` and `xml:id`. You can have a Default Namespace which has no prefix. Such as:

```
<?xml version="1.0" encoding="UTF-8"?>
<doc xmlns="http://example.com/namespace" xmlns:c="http://www.c.org/charl"
xml:lang="en">
  <chapter number="1" c:signed="me">
    <page number="1">
      <c:emerald name="sam"/>
      <title>A title</title>
      <text c:style="fancyEnglish">blah blah doc
        blah some text blah blah watch out for orbs blah blah
      </text>
    </page>
  </chapter>
</doc>
```

XML 1.1

The original XML specification does have a few limitations that have now been removed in XML 1.1. Rather than only supporting most of Unicode 2 in element and attribute names and processing instruction targets, version 1.1 of XML supports any Unicode based character that is not specifically forbidden allowing support for Unicode 3, 4 and future versions. Also XML 1.1 supports [IRIs](#).

Plus support for the NEL new line character (`#x85`), mostly used on IBM's AIX Operating System, as well as the Unicode Line Separator character (`#x2028`).

A standard flexible document exchange format, XML - Legend Scrolls

This article and others are available online and in other document formats at:

<http://www.legendscrolls.co.uk/webstandards/>

Author: Peter Davison from [Legend Scrolls](#).

Copyright ©2005-2008 Legend Scrolls and Peter Davison.

The Globe icon from Crystal Project Icons: LGPL, Copyright © [Everaldo](#).

All rights reserved.