XML Basics

1. XML – Overview

XML stands for **E**xtensible **M**arkup **L**anguage. It is a text-based markup language derived from Standard Generalized Markup Language (SGML).

XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data. XML is not going to replace HTML in the near future, but it introduces new possibilities by adopting many successful features of HTML.

There are three important characteristics of XML that make it useful in a variety of systems and solutions:

- **XML is extensible**: XML allows you to create your own self-descriptive tags or language, that suits your application.
- **XML carries the data, does not present it**: XML allows you to store the data irrespective of how it will be presented.
- **XML is a public standard**: XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

XML Usage

A short list of XML usage says it all:

- XML can work behind the scene to simplify the creation of HTML documents for large web sites.
- XML can be used to exchange the information between organizations and systems.
- XML can be used for offloading and reloading of databases.
- XML can be used to store and arrange the data, which can customize your data handling needs.
- XML can easily be merged with style sheets to create almost any desired output.
- Virtually, any type of data can be expressed as an XML document.

What is Markup?

XML is a markup language that defines set of rules for encoding documents in a format that is both human-readable and machine-readable. So, what exactly is a markup language? Markup is information added to a document that enhances its meaning in certain ways, in that it identifies the parts and how they relate to each other. More specifically, a markup language is a set of symbols that can be placed in the text of a document to demarcate and label the parts of that document.

Following example shows how XML markup looks, when embedded in a piece of text:

```
<message>
   <text>Hello, world!</text>
</message>
```

This snippet includes the markup symbols, or the tags such as <message>...</message> and <text>... </text>. The tags <message> and </message> mark the start and the end of the XML code fragment. The tags <text> and </text> surround the text Hello, world!.

Is XML a Programming Language?

A programming language consists of grammar rules and its own vocabulary which is used to create computer programs. These programs instruct the computer to perform specific tasks. XML does not qualify to be a programming language as it does not perform any computation or algorithms. It is usually stored in a simple text file and is processed by special software that is capable of interpreting XML.

2. XML – Syntax

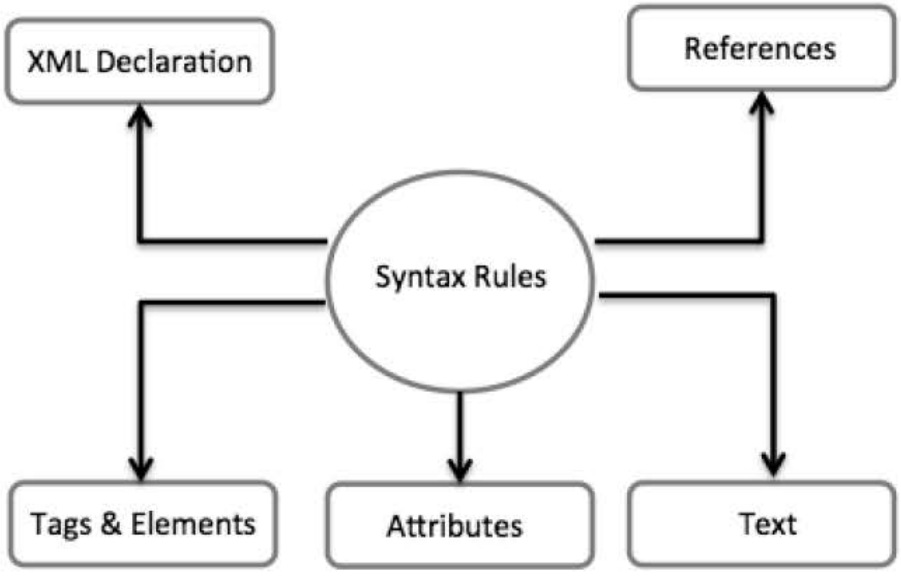
In this chapter, we will discuss the simple syntax rules to write an XML document. Following is a complete XML document:

```
<?xml version="1.0"?>
<contact-info>
<name>Tanmay Patil</name>
<company>TutorialsPoint</company>
<phone>(011) 123-4567</phone>
</contact-info>
```

You can notice, there are two kinds of information in the above example:

- Markup, like <contact-info>
- The text, or the character data, Tutorials Point and (040) 123-4567

The following diagram depicts the syntax rules to write different types of markup and text in an XML document.



Let us see each component of the above diagram in detail.

XML Declaration

The XML document can optionally have an XML declaration. It is written as follows:

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

Where *version* is the XML version and *encoding* specifies the character encoding used in the document.

Syntax Rules for XML Declaration

- The XML declaration is case sensitive and must begin with "**<?xml>**" where "**xml**" is written in lower-case.
- If the document contains XML declaration, then it strictly needs to be the first statement of the XML document.
- The XML declaration strictly needs be the first statement in the XML document.
- An HTTP protocol can override the value of *encoding* that you put in the XML declaration.

Tags and Elements

An XML file is structured by several XML-elements, also called XML-nodes or XML-tags. The names of XML-elements are enclosed in triangular brackets < > as shown below:

```
<element>
```

Syntax Rules for Tags and Elements

Element Syntax: Each XML-element needs to be closed either with start or with end elements as shown below:

```
<element>....</element>
```

or in simple-cases, just this way:

```
<element/>
```

Nesting of Elements: An XML-element can contain multiple XML-elements as its children, but the children elements must not overlap. i.e., an end tag of an element must have the same name as that of the most recent unmatched start tag.

The following example shows incorrect nested tags:

```
<?xml version="1.0"?>
<contact-info>
<company>TutorialsPoint
<contact-info>
</company>
```

The following example shows correct nested tags:

```
<?xml version="1.0"?>
<contact-info>
<company>TutorialsPoint</company>
<contact-info>
```

Root Element: An XML document can have only one root element. For example, following is not a correct XML document, because both the **x** and **y** elements occur at the top level without a root element:

```
<x>...</x>
<y>...</y>
```

The following example shows a correctly formed XML document:

```
<root>
   <x>...</x>
   <y>...</y>
</root>
```

Case Sensitivity: The names of XML-elements are case-sensitive. That means the name of the start and the end elements need to be exactly in the same case.

For example, **<contact-info>** is different from **<Contact-Info>**.

XML Attributes

An **attribute** specifies a single property for the element, using a name/value pair. An XML-element can have one or more attributes. For example:

```
<a href="http://www.tutorialspoint.com/">Tutorialspoint!</a>
```

Here ***href*** is the attribute name and ***http://www.tutorialspoint.com/*** is attribute value.

Syntax Rules for XML Attributes

- Attribute names in XML (unlike HTML) are case sensitive. That is, *HREF* and *href* are considered two different XML attributes.

- Same attribute cannot have two values in a syntax. The following example shows incorrect syntax because the attribute ***b*** is specified twice:

```
<a b="x" c="y" b="z">....</a>
```

- Attribute names are defined without quotation marks, whereas attribute values must always appear in quotation marks. Following example demonstrates incorrect xml syntax:

```
<a b=x>....</a>
```

In the above syntax, the attribute value is not defined in quotation marks.

XML References

References usually allow you to add or include additional text or markup in an XML document. References always begin with the symbol **"&"** which is a reserved character and end with the symbol **";"**. XML has two types of references:

- **Entity References:** An entity reference contains a name between the start and the end delimiters. For example, **&** where *amp* is *name*. The *name* refers to a predefined string of text and/or markup.

- **Character References:** These contain references, such as **A**, contains a hash mark ("#") followed by a number. The number always refers to the Unicode code of a character. In this case, 65 refers to alphabet "A".

XML Text

The names of XML-elements and XML-attributes are case-sensitive, which means the name of start and end elements need to be written in the same case. To avoid character encoding problems, all XML files should be saved as Unicode UTF-8 or UTF-16 files.

Whitespace characters like blanks, tabs and line-breaks between XML-elements and between the XML-attributes will be ignored.

Some characters are reserved by the XML syntax itself. Hence, they cannot be used directly. To use them, some replacement-entities are used, which are listed below:

Not Allowed Character	Replacement Entity	Character Description
<	&lt;	less than
>	&gt;	greater than
&	&amp;	ampersand

'	&apos;	apostrophe
"	&quot;	quotation mark

3. XML – Documents

An **XML document** is a basic unit of XML information composed of elements and other markup in an orderly package. An XML document can contain a wide variety of data. For example, database of numbers, numbers representing molecular structure or a mathematical equation.

XML Document Example

A simple document is shown in the following example:

```
<?xml version="1.0"?>
<contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact-info>
```

The following image depicts the parts of XML document.

Document Prolog Section

Document Prolog comes at the top of the document, before the root element. This section contains:

- XML declaration
- Document type declaration

You can learn more about XML declaration in this chapter : XML Declaration.

Document Elements Section

Document Elements are the building blocks of XML. These divide the document into a hierarchy of sections, each serving a specific purpose. You can separate a document into multiple sections so that they can be rendered differently, or used by a search engine. The elements can be containers, with a combination of text and other elements.

4. XML – Declaration

This chapter covers XML declaration in detail. **XML declaration** contains details that prepare an XML processor to parse the XML document. It is optional, but when used, it must appear in the first line of the XML document.

Syntax

Following syntax shows XML declaration:

```
<?xml
   version="version_number"
   encoding="encoding_declaration"
   standalone="standalone_status"
?>
```

Each parameter consists of a parameter name, an equals sign (=), and parameter value inside a quote. Following table shows the above syntax in detail:

Parameter	Parameter_value	Parameter_description
Version	1.0	Specifies the version of the XML standard used.
Encoding	UTF-8, UTF-16, ISO-10646-UCS-2, ISO-10646-UCS-4, ISO-8859-1 to ISO-8859-9, ISO-2022-JP, Shift_JIS, EUC-JP	It defines the character encoding used in the document. UTF-8 is the default encoding used.
Standalone	yes or no.	It informs the parser whether the document relies on the information from an external source, such as external document type definition (DTD), for its content. The default value is set to *no*. Setting it to *yes* tells the processor there are no external declarations required for parsing the document.

Rules

An XML declaration should abide with the following rules:

- If the XML declaration is present in the XML, it must be placed as the first line in the XML document.
- If the XML declaration is included, it must contain version number attribute.
- The parameter names and values are case-sensitive.
- The names are always in lower case.
- The order of placing the parameters is important. The correct order is: *version, encoding and standalone.*
- Either single or double quotes may be used.
- The XML declaration has no closing tag, i.e. **</?xml>**

XML Declaration Examples

Following are few examples of XML declarations:

XML declaration with no parameters:

```
<?xml >
```

XML declaration with version definition:

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

XML declaration with all parameters defined:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

XML declaration with all parameters defined in single quotes:

```
<?xml version='1.0' encoding='iso-8859-1' standalone='no' ?>
```

5. XML – Tags

Let us learn about one of the most important part of XML, the XML *tags*. **XML tags** form the foundation of XML. They define the scope of an element in XML. They can also be used to insert comments, declare settings required for parsing the environment, and to insert special instructions.

We can broadly categorize XML tags as follows:

Start Tag

The beginning of every non-empty XML element is marked by a start-tag. Following is an example of start-tag:

```
<address>
```

End Tag

Every element that has a start tag should end with an end-tag. Following is an example of end-tag:

```
</address>
```

Note, that the end tags include a solidus ("/") before the name of an element.

Empty Tag

The text that appears between start-tag and end-tag is called content. An element which has no content is termed as empty. An empty element can be represented in two ways as follows:

A start-tag immediately followed by an end-tag as shown below:

```
<hr></hr>
```

A complete empty-element tag is as shown below:

```
<hr />
```

Empty-element tags may be used for any element which has no content.