XML

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

- 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

- 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.

Advantages of XML

1) Readability: xml document is plain text and human readable. To edit or view xml documents, any simple text editors are sufficient.

2) Hierarchical: it has tree structure, to express complex data and its simple to understand.

3) Language Independent: xml documents are language neutral.

4) OS Independent: xml files are OS independent.

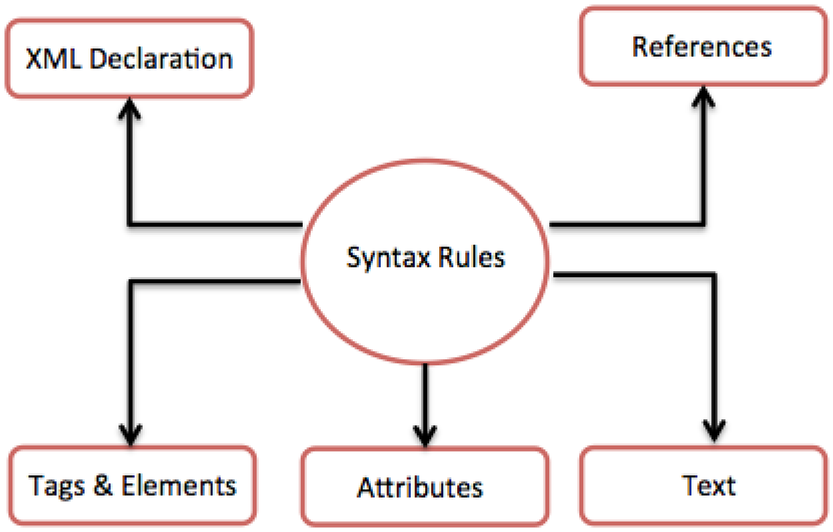
example:

```
<?xml version="1.0"?>
<student>
<name>lalitha</name>
<branch>cse</branch>
<rollnum>00251A0520</rollnum>
</student>
```

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

- markup, like <*student*> and
- the text, or the character data, *cse* and *00251A0520*.

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 below:

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

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

It is called as processing instruction.

Syntax Rules for XML declaration

- The XML declaration is case sensitive and must begin with "<?xml>" where "xml" is written in lower-case.

- If 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. XML-elements' names are enclosed by 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.

Following example shows incorrect nested tags:

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

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>**.

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.gnits.ac.in/">GNITS</a>
```

Here *href* is the attribute name and *http://www.gnits.ac.in/* 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 **&amp;** where *amp* is *name*. The *name* refers to a predefined string of text and/or markup.

Character References: These contain references, such as **&#65;**, 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

rules for XML:

I. always include an XML declaration.

II. placing whitespace characters before the xml declaration is an error.

III. xml is case sensitive.

IV. in an xml document, each start tag must have a matching end tag.

V. attribute values must always be either double quotes or single quotes.

VI. using either a space or a tab in an xml element or attribute name is an error.

VII. xml document must have a root tag.

VIII. comments: <!--   -->

IX. xml elements must be properly nested.

X. white spaces are preserved in xml.

example program:

```
<?xml version="1.0">
<books>
        <book-info>
                <name>web technologies</name>
                <author>chris bates</author>
        </book-info>
        <book-info>
                <name>network security</name>
                <author>william stallings</author>
        </book-info>
</books>
```

there are 3 control structures:

1) comments
2) processing instruction / XML declaration
3) Document Type Declaration (DTD)

DTD:

it holds the rules of grammer for a particular XML Data structure. those rules are used by validating parsers to check that not only is the file valid XML, but that is also obeys its own internal rules.

Syntax:

i) external declaration:

```
        <!DOCTYPE    root-element    SYSTEM     "filename">
```

ii) internal declaration:

```
        <!DOCTYPE     root-element   [element-declaration]>
```