eXtensible Markup Language (XML)

What is XML?

- XML stands for Extensible Markup Language.
- XML is a markup language much like HTML.
- XML tags are not predefined like HTML.
- XML was not designed to display data.
- XML was designed to be self-descriptive.

Definition of XML:

- **Extensible Markup Language (XML) is a markup language based on simple and platform-independent rules for processing and displaying textual information in a structured way.**

Introduction of XML:

- XML provides **customized tags** to format and display textual information.
- The extension of an XML document is **.xml.**
- XML documents are simple text documents that represent data in a platform-neutral manner.
- For example, an XML document generated by an application running on Microsoft Windows can be easily executed in an application running on Sun Solaris.
- The XML syntax allows you to share information between different computers, applications, and organizations without the need of passing through many layers of conversation.

XML Features:

- Structured language
- Platform independent
- Open standard
- Language independent
- Web enabled
- Extensible

– **Structured language:**
  - XML defines the structure of a document using tags.
  - XML is a user-defined language, implying that it allows you to create your own tags.

– **Platform independent:**
  - To execute the XML applications on any operating system.

– **Open standard:**
  - Creates a fair, competitive market for implementations of the standard.

- Customers need not to go to a particular vendor or group to use XML.

– **Language independent:**
  - Implies that you can use XML with any other technology or language, such as Java, .NET and Hypertext Preprocessor(PHP)
– **Web enabled:**
  - HTML content can be re-used to work with XML.
  - XML is frequently used over the web for data transfer.
– **Extensible:**
  - Allow you to create user-define tags.

HTML vs XML:

HTML	XML
It focus on presenting data i.e. **how** data looks.	It focus on describing data, i.e **what** data is.
HTML tags are predefined. E.g. <table>, <form> etc.	XML tags are customized tags. E.g. <name>, <address>,<city> etc.
HTML provides predefined elements and attributes whose behavior is well specified.	XML allows to create your own elements that are specific to your application or business needs.
It does not support case-sensitivity	It support case-sensitivity
It specifies that it is not mandatory to close each and every tag.	It specifies that it is mandatory to close each and every tag.
It specifies that it is not mandatory to enclose attribute values in double quotes(“ ”)	It specifies that it is mandatory to enclose attribute values in double quotes(“ ”)

Advantages of XML:

- Stores data in form of plain text.
- Provides a basic syntax that can be used to share information among different applications.
- Allow you not to restrict to a limited set of tags.
- It is completely compatible with java.
- 100% portable that it can be used to large network with multiple platforms such as the Internet and it can be used on handhelds or palmtops.
- Represents the international standards of web.

Disadvantages of XML:

- Represents redundant and similar syntax for binary data. This redundancy affects the efficiency of an application.
- XML does not provide any specific notion for integer, string, Boolean and so on.
- Requires a processing application so that anyone , anywhere in the world, can read your document.
- Requires XML specific browsers, which are not in market yet.

XML document structure:

- XML defines certain rules for its syntax that specify how to create or structure an XML document.
- The syntax used to create an XML document is called **markup syntax**.
- While creating XML document, remember the following points:
    - XML document must have starting and closing tags.
    - XML tags are case-sensitive.
    - XML elements must be properly nested.
    - XML documents must have one root element.
    - XML attribute's values must be enclosed in double quotes(" ").

Syntax:

```
<?xml version="1.0" encoding="ISO-8859--1" ?>
<root_element>
        <child_element1>
                <subchild_element1  attribute_name="value">
                </subchild_element1>
                <subchild_element2>
                </subchild_element2>
        </child_element1>
        <child_element2>
                <subchild_element1></subchild_element1>
                <subchild_element2></subchild_element2>
        </child_element2>
</root_element>
```

Example:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- This is an Employee element  -->
<Employee>
        <FirstName>Pratham </FirstName>
        <LastName> Patil </LastName>
        <Age>25</Age>
```

<EmpID id="e001"> </EmpID>
</Employee>

XML document:

- XML document contains following sections:
  - XML declaration
  - XML elements
  - XML attributes
  - XML tree
  - XML comments

XML declaration:

- XML declaration statement indicate that the specified document is an XML document.
- XML declaration statement is the first line in the document.
- It defines the XML **version** and **character** encoding.
- Example:

  <?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
- Attributes of XML declaration statement are:
  - **Version:**
    - Represents the version of XML.
    - This attribute must be provided in the declaration.
  - **Encoding:**
    - Specifies the character encoding (character-set) that the document uses.
    - **UTF-8** is the default for documents.
    - This attribute is optional.
  - **Standalone:**
    - Contains two values, yes and no.
    - The "yes" value specifies that the XML document has an internal DTD.
    - The "no" value specifies that XML document is linked to an external DTD or any external entity references.

- **Important point about XML declaration:**
  - XML declaration starts with <?xml and ends with ?>
  - XML declaration must include the version attribute; however, the encoding and standalone attributes are optional.
  - XML declaration must be at the beginning of the file.
  - The version, encoding and standalone attributes must be in the order.

XML elements:

- XML elements are the basic building block of an XML document.
- Every XML document consists of XML elements and the content embedded within these elements.
- A start tag and end tag delimit(restrict) an element in an XML document.
- A start tag is delimited by the < and > characters and an end tag is delimited by the </ and > characters.
- Example: <Employee></Employee>
- The XML document must have a single root element, which is the top most parent element.
- Root element does not have a parent element, but can have one or more child elements.
- Root element also called **document element**.
- Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Employee>
        <FirstName> Priya</FirstName>
        <LastName> Verma </LastName>
</Employee>
```

- **Rules while defining XML element:**
  - Element name can be start with letters or the underscore(_) character, but not numbers or other punctuation characters.
  - After the first characters, you can use numbers and characters, such as hyphen (-) and period (.)
  - Element names cannot contain spaces.
  - Element name cannot contain colon (:) character because it is reserved character. It is used while working with namespaces
  - Element name cannot start with the words, such as **xml**, in uppercase, lowercase, or mixed case.
  - There cannot be a space after the opening character(<); the name of the element must come immediately after it. There can be space before the closing character (>).

- XML elements are divided into two categories:
  - **Empty elements**
  - **Nested elements**

  - **Empty elements:**
    - An empty element does not contain any content or any other element within it.
    - It can have attributes that help in identifying an entity.
    - An empty element can be written without an end tag.
    - Example:

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

```
        <employee id="e001" />
    </employees>
```

- <employee> element is an empty element.

– **Nested element:**

- An XML element can also contain other elements.
- The elements which contain other elements are known as **nested elements**.
- In XML document, the elements must be properly nested, implying the parent element has to be opened before the child element and should be closed after the child element.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<employees>
        <FirstName> Pratham </FirstName>
        <LastName> Parikh </LastName>
</employees>
```

XML attributes:

- XML attributes provide additional information about elements.
- An attribute is a name-value pair contained in the element's start tag.
- Variable names are separated from values by the assignment operator.
- Values of attributes are enclosed either in single quotation marks or double quotation marks.
- Multiple attributes used in a single element are separated by space.
- Example 1:

```
<?xml version="1.0" encoding="UTF-8"?>
<employees>
        <employee id="e001" name="Bharat" />
</employees>
```

- Example 2:

```
<?xml version="1.0" encoding="UTF-8"?>
<student gender="female" id="01">
                <firstname>Maya</firstname>
                <lastname>Purohit</lastname>
</student>
```

XML tree:

- An XML tree represents the elements of an XML document in a tree structure.

- XML tree contain following elements:
  - Root element
  - Parent element
  - Child element
  - Siblings

XML comments:

- Comments are used to specify some information or remarks.
- A comment is not a part of the program and therefore it is not parsed by the parser.
- Similar to HTML, XML comment begins with <!-- and ends with -->

Example:

<!-- This is employee element -->

- Comments can be written anywhere in the XML document, even before the root element.
- You cannot write comment inside a tag or another comment.

Ex:<emp id="e1">

<!– abc -- >

</emp>

XML entity references:

- Some characters are used for constructing tags; and therefore, they cannot be used in the content of an XML document. These characters are known as **entity references**.
- For example, less than (<) character is used to open a tag and greater than character (>) is used to close the tag.
- If these characters(symbols) are used as such in between the content of the element, it will confuse the browser.

XML provides following entity references:

Entity references	Description
<	Represents the less-than (<) symbol.
>	Represents the greater-than (>) symbol.
&	Represents the ampersand (&) symbol.
'	Represents the apostrophe (‘) symbol.
"	Represents the double quotation marks (“)

XML parser:

- XML parser is used to read, update, create and manipulate an XML document.

- To manipulate the XML document, the XML parser loads the document into the computer's memory and then manipulates it using the DOM node-tree structure.
- XML parser is a part of software, which reads the XML document and tests whether or not it is well-formed against the DTD or XML schema.
- XML parsers are divided into **two** categories:
-

  1. Non-validating parser
  2. Validating parser

- Non-validating parser:
  - Checks whether the XML document is well-formed.
  - It does not check the DTD files that are referred in the XML document.
- Validating parser:
  - Validates the XML document as well as DTD files referred in the document.

XML parser examples:

- **expat**
  - XML parser library that is written in C programming language.
  - It is non-validating XML parser.
- **Simple API for XML:**
  - XML parser that provides the mechanism for reading data from an XML document.
- **Xerces:**
  - Allows to read and write XML data.
  - This is an example of validating XML parser.
- **MSXML:**
  - Allows you to build high-performance XML-based applications that provide an almost universally supported way of exchanging documents and data across applications and platforms.

XML Namespace:

- A namespace is a Uniform Resource Identifier (URI) that provides uniqueness to the names of elements and attributes of an XML document.
- It used to avoid the conflicts between the name of the elements or attributes.
- Namespaces provide a method to avoid the element name conflicts.
- The elements having same name may arise ambiguities for a web browser while displaying data.
- Example:

```
<?xml version="1.0" ?>
<details>
    <student>
        <id> 01</id>
```

```
            <name> Bharat</name>
        </students>
        <employees>
            <id> 01</id>
            <name> Ramesh</name>
        </employees>
    </details>
```

- Namespace is used to differentiate elements and attributes of different types.
- A namespace is used when your XML document is going to be shared with other XML documents that may have same element names.
- **In XML, namespace is used by assigning a name prefix to it.**

Example:

```
<?xml version="1.0" ?>
<details>
    <s1:student>
        <s1:id> 01</s1:id>
        <s1:name> Bharat</s1:name>
    </s1:student>
    <e1:employees>
        <e1:id> 01</e1:id>
        <e1:name> Ramesh</e1:name>
    </e1:employees>
</details>
```

When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace.

- Namespaces can also be declared the XML root element:

```
<root
xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="http://www.w3schools.com/furniture">

<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```