# WEB TECHNOLOGIES

XML

## Topics to Cover

- XML: Introduction to XML-Mark up languages
- Features of Mark up languages
- XML Naming rules,
- Building block of XML Document
- Difference between HTML & XML
- Components of XML, XML Parser, DTD’s Using XML with HTML and CSS

## Introduction to XML

- XML (eXtensible Markup Language) is a markup language.
- Xml is designed to store and transport data.
- Xml was released in late 90’s. It was created to provide an easy to use and store self describing data.
- Xml became a W3C Recommendation on February 10,1998
- Xml is not a replacement for HTML.
- Xml is designed to be self-descriptive.
- Xml is designed to carry data , not to display data.
- Xml tags are not predefined. You must define your own tags.
- Xml is platform independent and language independent.

## Why XML

- Platform independent and language independent: The main benefit of xml is that you can use in to take data from a program loke Microsoft SQL, convert in into XML then share that XML with other programs and platforms.
- You can communicate between two platforms which are generally very difficult.
- The main thing which makes XML truly powerful is its international acceptance. Many corporation use XML interfaces for databases, programming, office application mobile phones and more.
- It is due to its platform independent feature.

## Features and Advantages/Benefit of XML:

- XML is widely used in the era of web development.
- It is also used to simplify data storage and data sharing.

The main features or advantages/ Benefits of Xml are given below

1) XML separates data from HTML
   - if you need to display dynamic data in your HTML document, it will take a lot of work to edit the HTML each time the data changes.
   - With XML, data can be stored in separate XML files.
   - This way you can focus on using HTML/CSS for display and layout, and be sure the changes in the underlying data will not required any changes to the HTML.
   - With a few lines of JavaScript code, you can read an external XML file and update the data content of your we page.

2) XML simplifies data sharing

- In the real world, computer systems and databases contain data in incompatible formats.
- XML data is stored I plain text format. This provides a software and hardware independent way of storing data.
- This makes it much easier to create data that can be shared by different applications.

3) XML simplifies data transport

- One of the most time-consuming for developers is to the exchange data between incompatible over the internet.
- Exchanging data as XML greatly reduces this complexity. Since tha data can be read by different incompatible applications.

4) XML simplifies platform change

- Upgrading to new systems (hardware or software platforms) is always tome consuming Large amounts of data must be converted and incompatible data is often lost.
- XML data is stored in text format.
- This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers without losing data.

5) XML increases data availability.

- Different applications can access your data, not only in HTML pages, but also from XML data sources.
- With XML, your data can be available to all king of “reading machines” (Handheld computers, voice machines, new feeds etc) and make it more available for blind peoplr or people with other disabilities.

6) XML can be used to create new internet languages.

- A lot of new internet languages are created with XML.

Here are some examples:

- XHTML
- WSDL for describing available we services.
- WAP and WML as markup languages for handheld devices.
- RSS languages for new feeds.
- RDF and OWL for describing resources and ontology.
- SMIL for describing multimedia for the web.

## XML Example

- Xml documents create a hierarchical structure looks like a tree so it is known as XML Tree that starts at “the root” and branches to “the leaves”

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

And finally the last line defines the end of the root elements

- The first is the XML declaration.
- It defines the XML version(1.0) and the encoding used.
- The next line describes the root element of the document
- The next 4 lines describes 4 child element of the root.
-
```
<to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
```

- Xml documents must contain a root elements.
- This element is the “the parent of all other elements.
- The elements in an XML document from a document tree.
- The tree starts at the root and branches to the lowest level of the tree.
- All elements can have sub elements (child elements).

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

## XML Naming Rules

XML elements must follow these naming rules:

- Element names are case-sensitive
- Element names must start with a letter or underscore
- Element names cannot start with the letters xml (or XML, or Xml, etc)
- Element names can contain letters, digits, hyphens, underscores, and periods
- Element names cannot contain spaces

Any name can be used, no words are reserved (except xml).

## Naming Styles

- There are no naming styles defined for XML elements. But here are some commonly used:

| Style | Example | Description |
|---|---|---|
| Lower case | <firstname> | All letters lower case |
| Upper case | <FIRSTNAME> | All letters upper case |
| Underscore | <first_name> | Underscore separates words |
| Pascal case | <FirstName> | Uppercase first letter in each word |
| Camel case | <firstName> | Uppercase first letter in each word except the first |

If you choose a naming style, it is good to be consistent!

## Budling block of XML documents

- XML documents (and HTML documents) are made up by the following building blocks:

  Elements, Tags, Attributes, Entities, #PCDATA, and CDATA

- **Elements**

  Elements are the main building blocks of both XML and HTML documents.

  Examples of HTML elements are "body" and "table". Examples of XML elements could be "note" and "message". Elements can contain text, other elements, or be empty. Examples of empty HTML elements are "hr", "br" and "img".

- **Tags**

  Tags are used to markup elements.

  A starting tag like <element_name> mark up the beginning of an element, and an ending tag like </element_name> mark up the end of an element.

Examples:
A body element: <body>body text in between</body>.
A message element: <message>some message in between</message>

- **Attributes**

  Attributes provide extra information about elements.

  Attributes are placed inside the start tag of an element. Attributes come in name/value pairs. The following "img" element has an additional information about a source file:

<img src=“computer.jpg”/>

The name of the element is "img". The name of the attribute is "src". The value of the attribute is "computer.gif". Since the element itself is empty it is closed by a " /".

- **#PCDATA**

#PCDATA means parsed character data.

Think of character data as the text found between the start tag and the end tag of an XML element.

#PCDATA is text that will be parsed by a parser. Tags inside the text will be treated as markup and entities will be expanded.

- **CDATA**

CDATA also means character data.

CDATA is text that will NOT be parsed by a parser. Tags inside the text will NOT be treated as markup and entities will not be expanded.

## ▪ Entities

Entities as variables used to define common text. Entity references are references to entities.

Most of you will known the HTML entity reference: "&nbsp;"  that is used to insert an extra space in an HTML document. Entities are expanded when a document is parsed by an XML parser.

The following entities are predefined in XML:

| Entity References | Character |
|---|---|
| &lt; | < |
| &gt; | > |
| &amp; | & |
| &quot; | " |
| &apos; | ' |

## XML Comments Rules

Any text between **<!--** and **-->** characters is considered as a comment.

Following rules should be followed for XML comments −

- Comments cannot appear before XML declaration.
- Comments may appear anywhere in a document.
- Comments must not appear within attribute values.
- Comments cannot be nested inside the other comments.

## Differentiate between HTML and XML

| HTML | XML |
|---|---|
| Is a markup language. | Is a standard markup language that defines other markup languages. |
| Is not case sensitive. | Is case sensitive. |
| Doubles up as a presentation language. | Is not a presentation language nor a programming language. |
| Has its own predefined tags. | Tags are defined as per the need of the programmer. XML is flexible as tags can be defined when needed. |
| Closing tags are not necessarily needed. | Closing tags are used mandatorily. |

| | |
|---|---|
| White spaces are not preserved. | Capable of preserving white spaces. |
| Showcases the design of a web page in the way it is displayed on client-side. | Enables transportation of data from database and related applications. |
| Used for displaying data. | Used for transferring data. |
| Static in nature. | Dynamic in nature. |
| Offers native support. | With the help of elements and attributes, objects are expressed by conventions. |
| Null value is natively recognised. | Xsi:nil on elements is needed in an XML instance document. |
| Extra application code is not needed to parse text. | XML DOM application and implementation code is needed to map text back into JavaScript objects. |

## XML Related Technologies

| No. | Technology | Meaning | Description |
|---|---|---|---|
| 1) | XHTML | Extensible html | It is a clearer and stricter version of XML. It belongs to the family of XML markup languages. It was developed to make html more extensible and increase inter-operability with other data. |
| 2) | XML DOM | XML document object model | It is a standard document model that is used to access and manipulate XML. It defines the XML file in tree structure. |
| 3) | XSL<br>it contain three parts:<br>i) XSLT (xsl transform)<br>ii) XSL<br>iii)XPath | Extensible style sheet language | i) It transforms XML into other formats, like html.<br>ii) It is used for formatting XML to screen, paper etc.<br>iii) It is a language to navigate XML documents. |
| 4) | XQuery | XML query language | It is a XML based language which is used to query XML based data. |
| 5) | DTD | Document type definition | It is an standard which is used to define the legal elements in an XML document. |

| | | | |
|---|---|---|---|
| 6) | XSD | XML schema definition | It is an XML based alternative to dtd. It is used to describe the structure of an XML document. |
| 7) | XLink | XML linking language | xlink stands for XML linking language. This is a language for creating hyperlinks (external and internal links) in XML documents. |
| 8) | XPointer | XML pointer language | It is a system for addressing components of XML based internet media. It allows the xlink hyperlinks to point to more specific parts in the XML document. |
| 9) | SOAP | Simple object access protocol | It is an acronym stands simple object access protocol. It is XML based protocol to let applications exchange information over http. in simple words you can say that it is protocol used for accessing web services. |
| 10) | WSDL | web services description languages | It is an XML based language to describe web services. It also describes the functionality offered by a web service. |
| 11) | RDF | Resource description framework | RDF is an XML based language to describe web resources. It is a standard model for data interchange on the web. It is used to describe the title, author, content and copyright information of a web page. |
| 12) | SVG | Scalable vector graphics | It is an XML based vector image format for two-dimensional images. It defines graphics in XML format. It also supports animation. |
| 13) | RSS | Really simple syndication | RSS is a XML-based format to handle web content syndication. It is used for fast browsing for news and updates. It is generally used for news like sites. |

## XML DTD

- “DTD stands for Document Type Definition.
- It defines the legal building blocks of an XML documents.
- It is used to define document structure with a lot of legal elements and attributes.
- A DTD defines the legal elements of an XML document.
- In simple words we can say that a DTD defines the document structure with a list of legal elements and attributers.
- XML schema is a XML based alternative to DTD.
- Actually DTD and XML schema both are used to form a well formed XML documents.
- We should avoid errors in XML documents because they will stop the XML programs.

**Purpose of DTD**

- Its main purpose is to define the structure on an XML document.
- It contains a list of legal elements and define the structure with the help of them.

**Checking Validation**

- Before proceeding with XML DTD, You must check the validation.
- AN XML documents is called “well-formed” if it contains the correct syntax.
- A well-formed and valid XML documents is one which has been validated against DTD.

## XML document with DTD

- Employee.xml

```
<?xml version=“1.0”?>
<!DOCTYPE employee SYSTEM “employee.dtd”>
<employee>
<firstname>Ruchi</firstname>
<lastname>Sawhney</lastname>
<email>xyz@gmail.com</email>
</employee>
```

- In the above examples, the DOCTYPE refers to an external DTD file.

The content of the file is shown in below paragraph.

File:employee.dtd

```
<!ELEMENT employee (firstname,lastname,email)>

<!ELEMENT firstname(#PCDATA)>

<!ELEMENT lastname(#PCDATA)>

<!ELEMENT email(#PCDATA)>
```

## Description of DTD

- <!DOCTYPE employee : It defines that the root elements of the document is employee
- <!ELEMENT employee : It defines that the employee element contains 3 elements “firstname, lastname and email”.
- <!ELEMENT firstname : It defines that the firstname element is #PCDATA typed.(parse-able data type).
- <!ELEMENT lastname : It defines that the lastname element is #PCDATA typed.(parse-able data type).
- <!ELEMENT email: It defines that the email element is #PCDATA typed.(parse-able data type).

## XML DTD with entity declaration

- A doctype declaration can also define special strings that can be used in the XML file. An entity has three parts:

a. An ampersand (&)

b. An entity name

c. A semicolon(:)

Syntax to declare entity:

```
<!ENTITY entity-name “entity-value”>
```

Example :

```
<? Xml version=“1.0” standalone=“yes”>
<!DOCTYPE author[
<!ELEMENT author (#PCDATA)>
<!ENTITY sj “Lakhmir Singh”>
]>
```

<author &sj </author>

- In the above example, sj is an entity that is used inside the author element
- In such case, it will print the value of sj entity that is “Lakhmir Singh”

## XML Schema

- An XML Schema describes the structure of an XML document, just like a DTD.
- An XML document with correct syntax is called "Well Formed".
- An XML document validated against an XML Schema is both "Well Formed" and "Valid".
- XML Schema is an XML-based alternative to DTD
- 
```
<xs:element name="note">

<xs:complexType>
  <xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

## The Schema above is interpreted like this:

- <xs:element name="note"> defines the element called "note"
- <xs:complexType> the "note" element is a complex type
- <xs:sequence> the complex type is a sequence of elements
- <xs:element name="to" type="xs:string"> the element "to" is of type string (text)
- <xs:element name="from" type="xs:string"> the element "from" is of type string
- <xs:element name="heading" type="xs:string"> the element "heading" is of type string
- <xs:element name="body" type="xs:string"> the element "body" is of type string

## XML Schemas are More Powerful than DTD

- XML Schemas are written in XML
- XML Schemas are extensible to additions
- XML Schemas support data types
- XML Schemas support namespaces

## Why Use an XML Schema?

- With XML Schema, your XML files can carry a description of its own format.
- With XML Schema, independent groups of people can agree on a standard for interchanging data.
- With XML Schema, you can verify data.

## XML Schemas Support Data Types

One of the greatest strengths of XML Schemas is the support for data types:

- It is easier to describe document content
- It is easier to define restrictions on data
- It is easier to validate the correctness of data
- It is easier to convert data between different data types

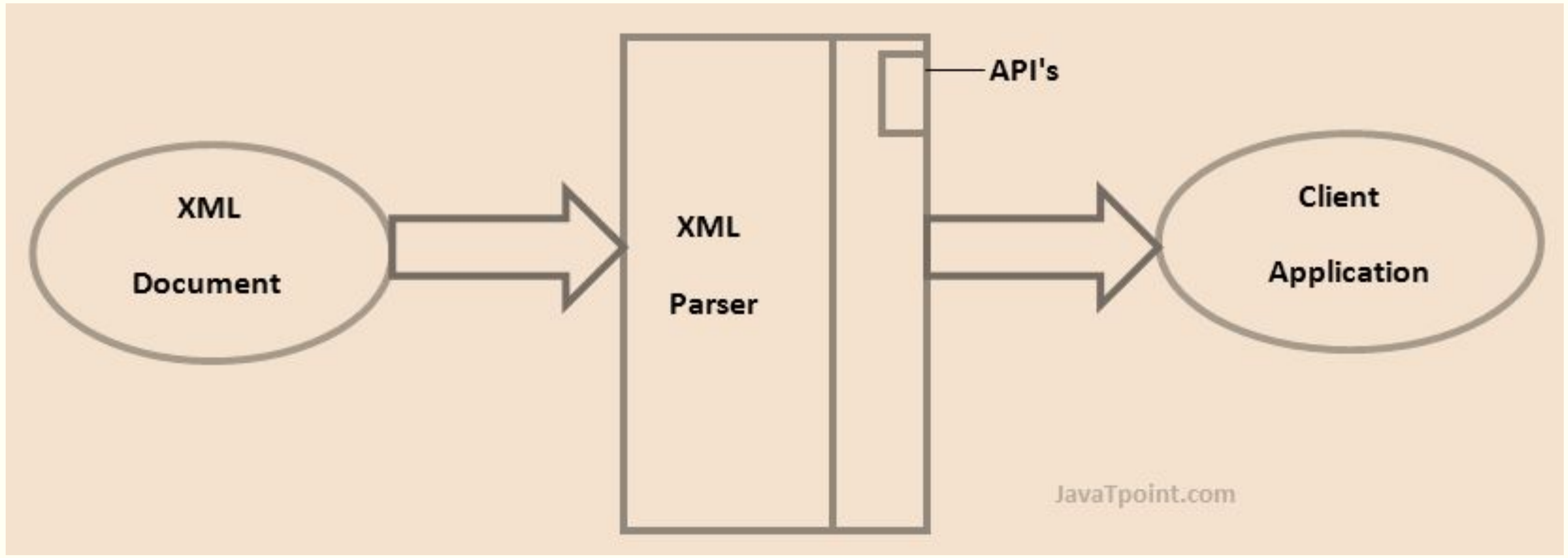
## XML Schemas use XML Syntax

Another great strength about XML Schemas is that they are written in XML:

- You don't have to learn a new language
- You can use your XML editor to edit your Schema files
- You can use your XML parser to parse your Schema files
- You can manipulate your Schemas with the XML DOM
- You can transform your Schemas with XSLT

## XML Parsers

- An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.
- ML parser validates the document and check that the document is well formatted.
- Let's understand the working of XML parser by the figure given below:



## Types of XML Parsers

- These are the two main types of XML Parsers:

1. DOM
2. SAX

## DOM (Document Object Model)

- A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

- Features of DOM Parser
- A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.
- DOM Parser has a tree based structure.
- Advantages
- 1) It supports both read and write operations and the API is very simple to use.
- 2) It is preferred when random access to widely separated parts of a document is required.
- Disadvantages
- 1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory).
- 2) It is comparatively slower than other parsers.

## SAX (Simple API for XML)

- A SAX Parser implements SAX API. This API is an event based API and less intuitive.
- Features of SAX Parser
- It does not create any internal structure.
- Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.
- It is an event based parser, it works like an event handler in Java.
- Advantages
- 1) It is simple and memory efficient.
- 2) It is very fast and works for huge documents.
- Disadvantages
- 1) It is event-based so its API is less intuitive.
- 2) Clients never know the full information because the data is broken into pieces.