**UNIT-II**

XML is a software- and hardware-independent tool for storing and transporting data.

* XML stands for eXtensible Markup Language
* XML is a markup language much like HTML
* XML was designed to store and transport data
* XML was designed to be self-descriptive
* XML is a W3C Recommendation

<note>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

The XML above is quite self-descriptive:

* It has sender information
* It has receiver information
* It has a heading
* It has a message body

But still, the XML above does not DO anything. XML is just information wrapped in tags.

**The Difference between XML and HTML**

XML and HTML were designed with different goals:

* XML was designed to carry data - with focus on what data is
* HTML was designed to display data - with focus on how data looks
* XML tags are not predefined like HTML tags are

**XML Simplifies Things**

* XML simplifies data sharing
* XML simplifies data transport
* XML simplifies platform changes
* XML simplifies data availability

**XML Separates Data from Presentation**

XML does not carry any information about how to be displayed.

The same XML data can be used in many different presentation scenarios.

Because of this, with XML, there is a full separation between data and presentation.

**XML is Often a Complement to HTML**

In many HTML applications, XML is used to store or transport data, while HTML is used to format and display the same data.

**XML Separates Data from HTML**

When displaying data in HTML, you should not have to edit the HTML file when the data changes.

With XML, the data can be stored in separate XML files.

With a few lines of JavaScript code, you can read an XML file and update the data content of any HTML page.

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

<bookstore>

<book category="cooking">

<title lang="en">Everyday Italian</title>

<author>Giada De Laurentiis</author>

<year>2005</year>

<price>30.00</price>

</book>

<book category="children">

<title lang="en">Harry Potter</title>

<author>J K. Rowling</author>

<year>2005</year>

<price>29.99</price>

</book>

<book category="web">

<title lang="en">XQuery Kick Start</title>

<author>James McGovern</author>

<author>Per Bothner</author>

<author>Kurt Cagle</author>

<author>James Linn</author>

<author>Vaidyanathan Nagarajan</author>

<year>2003</year>

<price>49.99</price>

</book>

<book category="web" cover="paperback">

<title lang="en">Learning XML</title>

<author>Erik T. Ray</author>

<year>2003</year>

<price>39.95</price>

</book>

</bookstore>

XML documents form a tree structure that starts at "the root" and branches to "the leaves".

**The XML Tree Structure**



**XML Syntax Rules**

XML Documents Must Have a Root Element

XML documents must contain one root element that is the parent of all other elements:

<root>

<child>

<subchild>.....</subchild>

</child>

</root>

In this example <note> is the root element:

<?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>

**The XML Prolog**

This line is called the XML prolog:

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

* The XML prolog is optional. If it exists, it must come first in the document.
* XML documents can contain international characters
* To avoid errors, you should specify the encoding used, or save your XML files as UTF-8.
* UTF-8 is the default character encoding for XML documents.
* Character encoding can be studied in our Character Set Tutorial.
* UTF-8 is also the default encoding for HTML5, CSS, JavaScript, PHP, and SQL.
* All XML Elements Must Have a Closing Tag
* XML Tags are Case Sensitive
* XML Elements Must be Properly Nested
* XML Attribute Values Must Always be Quoted
* White-space is Preserved in XML

## Entity References: There are 5 pre-defined entity references in XML:

&lt; < less than

&gt; > greater than

&amp; & ampersand

&apos; ' apostrophe

&quot; " quotation mark

**Comments in XML**

The syntax for writing comments in XML is similar to that of HTML:

<!-- This is a comment -->

**XML Elements:**

An XML element is everything from (including) the element's start tag to (including) the element's end tag.

<price>29.99</price>

An element can contain:

* text
* attributes
* other elements
* or a mix of the above

**Empty XML Elements**

An element with no content is said to be empty.

In XML, you can indicate an empty element like this:

<element></element>

You can also use a so called self-closing tag:

<element />

**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).

**XML Attributes**

* XML elements can have attributes, just like HTML.
* Attributes are designed to contain data related to a specific element.
* Attribute values must always be quoted. Either single or double quotes can be used.

For a person's gender, the <person> element can be written like this:

<person gender="female">

or like this:

<person gender='female'>

**XML Attributes for Metadata**

Sometimes ID references are assigned to elements. These IDs can be used to identify XML elements in much the same way as the id attribute in HTML. This example demonstrates this:

<messages>

<note id="501">

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

<note id="502">

<to>Jani</to>

<from>Tove</from>

<heading>Re: Reminder</heading>

<body>I will not</body>

</note>

</messages>

**XML Namespaces:**

XML Namespaces provide a method to avoid element name conflicts.

**Name Conflicts**

In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.

This XML carries HTML table information:

<table>  
  <tr>  
    <td>Apples</td>  
    <td>Bananas</td>  
  </tr>  
</table>

**This XML carries information about a table (a piece of furniture):**

<table>  
  <name>African Coffee Table</name>  
  <width>80</width>  
  <length>120</length>  
</table>

If these XML fragments were added together, there would be a name conflict. Both contain a <table> element, but the elements have different content and meaning.

A user or an XML application will not know how to handle these differences.

**Solving the Name Conflict Using a Prefix**

Name conflicts in XML can easily be avoided using a name prefix.

This XML carries information about an HTML table, and a piece of furniture:

<root>

<h:table xmlns:h="http://www.w3.org/TR/html4/">

<h:tr>

<h:td>Apples</h:td>

<h:td>Bananas</h:td>

</h:tr>

</h:table>

<f:table xmlns:f="https://www.w3schools.com/furniture">

<f:name>African Coffee Table</f:name>

<f:width>80</f:width>

<f:length>120</f:length>

</f:table>

</root>

**(Or)**

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

**XML Parser:**

All major browsers have a built-in XML parser to access and manipulate XML.

XML Parser

The XML DOM (Document Object Model) defines the properties and methods for accessing and editing XML.

However, before an XML document can be accessed, it must be loaded into an XML DOM object.

All modern browsers have a built-in XML parser that can convert text into an XML DOM object.

**Parsing a Text String**

This example parses a text string into an XML DOM object, and extracts the info from it with JavaScript:

**Example**

<html>  
<body>  
<p id="demo"></p>  
<script>  
var text, parser, xmlDoc;  
text = "<bookstore><book>" +  
"<title>Everyday Italian</title>" +  
"<author>Giada De Laurentiis</author>" +  
"<year>2005</year>" +  
"</book></bookstore>";  
parser = new DOMParser();  
xmlDoc = parser.parseFromString(text,"text/xml");  
document.getElementById("demo").innerHTML =  
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;  
</script>  
</body>  
</html>

**XSLT**

XSL (eXtensible Stylesheet Language) is a styling language for XML.

XSLT stands for XSL Transformations.

**<xsl:template>** element is used to build templates.

The match attribute is used to associate a template with an XML element. The match attribute can also be used to define a template for the entire XML document. The value of the match attribute is an XPath expression (i.e. match="/" defines the whole document).

**<xsl:value-of> Element**

The <xsl:value-of> element can be used to extract the value of an XML element and add it to the output stream of the transformation

**<xsl:for-each> Element**

The XSL <xsl:for-each> element can be used to select every XML element of a specified node-set:

**data.xml**

<?xml version="1.0" standalone="yes"?>

<?xml-stylesheet type="text/xsl" href="mySt.xsl"?>

<BooksCatalog>

<book>

<title>Computer Networks</title>

<author>Andrew S. TanenBaum</author>

<isbn>C254854</isbn>

<publisher>PHI</publisher>

<edition>2nd edition</edition>

<price>500</price>

</book>

</BooksCatalog>

**mySt.xsl**

<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">

<html>

<body>

<h2>BOOKS CATALOG </h2>

<table border="3" >

<tr >

<th>Book Title</th>

<th>Book Author</th>

<th>ISBN</th>

<th>Published By</th>

<th>Edition</th>

<th>Price</th>

</tr>

<xsl:for-each select="BooksCatalog/book">

<tr>

<td><xsl:value-of select="title"/></td>

<td><xsl:value-of select="author"/></td>

<td><xsl:value-of select="isbn"/></td>

<td><xsl:value-of select="publisher"/></td>

<td><xsl:value-of select="edition"/></td>

<td><xsl:value-of select="price"/></td>

</tr>

</xsl:for-each>

</table>

</body>

</html>

</xsl:template>

</xsl:stylesheet>

C:\Program Files\Google\Chrome\Application\>chrome.exe --allow-file-access-from-files