What is XML?

- XML stands for e**X**tensible **M**arkup **L**anguage.
- **XML is designed to transport and store data.**
- HTML was designed to display data.
- XML is a markup language much like HTML
- XML was designed to carry data, not to display data
- XML tags are not predefined. You must define your own tags
- XML is designed to be self-descriptive
- XML is Not a Replacement for HTML
- XML is the most common tool for data transmissions between all sorts of applications.
- XML is a W3C Recommendation

The Difference Between XML and HTML

XML and HTML were designed with different goals:

- XML was designed to transport and store data, with focus on what data is
- XML was created to structure, store, and transport information.
- HTML was designed to display data, with focus on how data looks

Role of XML

1. XML Simplifies Data Sharing

In the real world, computer systems and databases contain data in incompatible formats.

2.XML data is stored in 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 challenges for developers is to exchange data between incompatible systems over the Internet.

4.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 concentrate on using HTML/CSS for display and layout, and be sure that changes in the underlying data will not require any changes to the HTML

XML DOCUMENT:

```
<?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>
```

XML document contains following parts:

- **Prolog**
- **Body**

Prolog:

- Xml declaration
- Optional processing instrucion
- Comments
- Document type declartion

Xml Declaration

<?xml version=”1.0”? >

- Specifies its an xml document
- Version is 1.0(mandatory attribute)
- Encoding and standalone (optional attributes).

<?xml version=”1.0” encoding=”utf-8” standalone=”no”? >

- Utf-8:unicode transformation format which is the same character set as ASCII
- Standalone specifies is the XML document depend on other document or not(depedent:yes, not depedent:no)

Processing instruction

Processing instruction starts with <? And ends with ?>.They allow document to contain special instructions that are used tp pass parameters to applications

Comments

<!-- comment text -- >

- Donot use – with in comments
- Never place a comment with in tag
- Never place a comment before xml declarion

BODY:

```
<?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>
```

XML Element SYNTAX:

<tagename attribute=”value”> Content </tagname>

Naming Rules:

Rules should follow while selecting element name

- Names can only contains letters,digits and some special characers.
- Name canot start with a number or puncuation mark
- Names must not contain the string “xml” “XML” “Xml”
- Names canot contain white spaces

Empty element

<line></line> or <line/>

WELL FORMED XML

- Document must have exactly one root element
- All tags must be closed

  <name>Ram (not correct)

  <name> Ram </name> (correct)

- All tags must be nested properly

  <b><i>Incorrect nesting</b></i>

  <b><i>Correct nesting</i></b>

- Xml tags are case sensitive

  <Message>This is correct</Message>

  <Message>This is incorrect</message>

<MESSAGE>This is incorrect</message >

- Attributes must be Quoted

  <speed unit=”rpm”>65777</speed> (correct)

  <speed unit=rpm>65777</speed> (not correct)

- Certain characters are reserved for processing

  <condition>if salary <1000 then</condition> (not correct)

  <condition>if salary &lt; 1000 then</condition>

Predefined Entities:

Entity name	**Character**	**Description**
&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

XML DTD:

- A Document Type Definition (DTD) defines the legal building blocks of an XML document. It defines the document structure with a list of legal elements and attributes.
- A DTD defines the document structure with a list of legal elements and attributes.
- A DTD can be declared inline inside an XML document, or as an external reference.
- DTD is two types I)internal DTD   II)External DTD

External DTD Declaration

If the DTD is declared in an external file, it should be wrapped in a DOCTYPE definition with the following syntax:

<!DOCTYPE root-element SYSTEM "filename">

Xml document

```
<?xml version="1.0" standalone=”yes”?>
< !DOCTYPE note SYSTEM "note.dtd">
<note>
    < to>Tove</to>
    < from>Jani</from>
    < heading>Reminder</heading>
    < body>Don't forget me this weekend!</body>
< /note>
```

And this is the file "note.dtd" which contains the DTD:

```
<!ELEMENT note (to,from,heading,body)>
< !ELEMENT to (#PCDATA)>
< !ELEMENT from (#PCDATA)>
< !ELEMENT heading (#PCDATA)>
< !ELEMENT body (#PCDATA)>
```

The Building Blocks of XML Documents

Seen from a DTD point of view, all XML documents (and HTML documents) are made up by the following building blocks:

- Elements
- Attributes
- Entities
- PCDATA
- CDATA

PCDATA

1. PCDATA means parsed character data.

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

3. PCDATA is text that WILL be parsed by a parser. The text will be examined by the parser for entities and markup.

4.Tags inside the text will be treated as markup and entities will be expanded.

However, parsed character data should not contain any &, <, or > characters; these need to be represented by the &amp; &lt; and &gt; entities, respectively.

CDATA

CDATA 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.

Declaring Elements

In a DTD, XML elements are declared with an element declaration with the following **syntax:**

<!ELEMENT element-name (element-content)>

example:

<!ELEMENT element-name (#PCDATA)>

< !ELEMENT from (#PCDATA)>

Example:
<!ELEMENT element-name (#CDATA)>
< !ELEMENT from (#CDATA)>

Declaring Attributes:

An attribute declaration has the following **syntax:**

<!ATTLIST element-name attribute-name attribute-type attribute-value>

DTD example:

< !ATTLIST   payment mode   CDATA   "check">

XML example:

< payment mode="check" />

Entity:

Entities are variables used to define shortcuts to standard text or special characters.

- Entity references are references to entities

Syntax

<!ENTITY entity-name "entity-value">

Example

DTD Example:

< !ENTITY writer "Donald Duck.">
< !ENTITY copyright "Copyright W3Schools.">

XML example:

< author>&writer;&copyright;</author>

Note: An entity has three parts: an ampersand (&), an entity name, and a semicolon (;).

Internal DTD

```
<?xml version="1.0"?>
< !DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
< note>
     <to>Tove</to>
      <from>Jani</from>
      <heading>Reminder</heading>
     <body>Don't forget me this weekend</body>
< /note>
```

XSLT

- XSL stands for EXtensible Stylesheet Language, and is a style sheet language for XML documents.
- XSLT stands for XSL Transformations. In this tutorial you will learn how to use XSLT to transform XML documents into other formats, like XHTML

CSS = Style Sheets for HTML:

- HTML uses predefined tags, and the meaning of each tag is well understood.
- The <table> tag in HTML defines a table - and a browser knows how to display it.
- Adding styles to HTML elements are simple. Telling a browser to display an element in a special font or color, is easy with CSS.

XSL = Style Sheets for XML

XML does not use predefined tags (we can use any tag-names we like), and therefore the meaning of each tag is not well understood

XSL describes how the XML document should be displayed!

XSL consists of three parts:

- XSLT - a language for transforming XML documents
- XPath - a language for navigating in XML documents
- XSL-FO - a language for formatting XML documents

XSLT = XSL Transformations;

XSLT is the most important part of XSL.

XSLT is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element

Root of XSl Document: <xsl:stylesheet>

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

<xsl:stylesheet version="1.0" xmlns:xsl="www.w3schools.com ">
  </xsl:stylesheet>
```

The <xsl:template> Element

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

The match attribute is used to associate a template with an XML element

Example:

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

<xsl:stylesheet version="1.0" xmlns:xsl=" www.w3schools.com ">
    <xsl:template  match=”/”>
   </xsl:template>
</xsl:styesheet>
```

XML DOM

- The XML DOM defines a standard way for accessing and manipulating XML documents.
- The DOM presents an XML document as a tree-structure.

What is the DOM?:

The DOM defines a standard for accessing documents like XML and HTML:

The DOM is separated into 3 different parts / levels:

- Core DOM - standard model for any structured document
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

What is the XML DOM?:

The XML DOM is:

- A standard object model for XML
- A standard programming interface for XML
- Platform- and language-independent
- A W3C standard

The XML DOM defines the **objects and properties** of all XML elements, and the **methods** (interface) to access them.

Ex:**<?xml version="1.0"?>**

```
<store>

         <HDD>

                  <make> samsung</make>

                  <capacity unit="GB">80</capacity>

                 <speed unit="rpm">7200</speed>

                <price currency="INR">1600</price>

        </HDD>

</store>
```

DOM Nodes:

According to the DOM, everything in an XML document is a **node**.

The DOM says:

- The entire document is a document node
- Every XML element is an element node
- The text in the XML elements are text nodes
- Every attribute is an attribute node
- Comments are comment nodes

Node :Node properties:

Property	**Description**
nodeType	**Holds positive integer,indicates node type**
nodeName	**Holds the name of node**
nodeValue	**Holds node value**
ChildNodes	**It represent array of child nodes of the context node**
firstChild	**It represent first child of the context node**
last Child	**It represent last child of the context node**
Attributes	**It represent array of attribute nodes of the context node**

Document Node:

Property	**Description**
DocumentElement	**This property refers to root node of the document**
docType	**It represent document type declaration for this document**
xmlVersion	**Represent version used to write xml document**

Methods	**Description**
CreateAttribute	**This method creates returns Attr type**
CreateElement	**Creates and returns an Element**
getElementById	**Returns the Element node having specific id value**
getElementsByTagName	**Returens a list Element nodes with specified element name**

Element Node:

Property	Description
tagName	tagName of element node

Methods	Description
getAttribute	Returns the value of attribute with specified name
getElementsByTagName	Returerns a list of Elemets with specified element name
removeAttribute	Removes an attribute with specified name
setAttribute	Adds an attribute with specified name and value

Text Node:

Property	Description
wholeText	It returns text of this node

Methods	Description
replaceWholeText	Replace text node value with another value

Attr Node:

Property	Description
isId	Whether the attribute is an ID attribute
Name	Returns name of the attribute
Value	Returns value of an attribute

Example

The JavaScript code to get the text from the first <title> element in books.xml:

txt=xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue

After the execution of the statement, txt will hold the value "Everyday Italian"

Explained:

- **xmlDoc - the XML DOM object created by the parser.**
- **getElementsByTagName("title")[0] - the first <title> element**
- **childNodes[0] - the first child of the <title> element (the text node)**
- **nodeValue - the value of the node (the text itself)**