UNIT-II [XML]

3.1 XML and its goals :

XML means eXtensible Markup Language

→ Markup is a set of instructions, often called tags, which can be added to text files. when a file is processed by a suitable application, the tags are used to control the structure or presentation of data contained in the file.

Example of markups

1. Microsoft RTF (Rich Text Format)
2. ADOBE PDF (Portable document format)
3. HTML (HyperText Markup Language)

- These describe how data looks but give no information about what it is
- XML is used to describe the structure of a document not the way that is presented.
- XML is a recommendation of the world wide web consortium (W3C)
- XML was developed by IBM in 1998. XML is derived from Standard Generalized Markup Language (SGML), which is a system for defining the markup Language.
- XML is a case-sensitive meta-markup language because it allows you to create custom-defined markup languages
- You can use XML to design domain-specific language that describe the information in a document
- You can create XML documents using text editor and saved with ".xml" extension.

- XML is used to add structural and formatting information to data.

Areas of XML :-

1. XML structuring the data for storage where a ~~relational rel~~ relational database is inappropriate.
2. Structuring the data for presentation on webpages

Advantages of XML :-

1. It acts as a mini database
2. XML is used to create new markup languages
3. Using XML, You can create your own tags
4. XML is used to create structured data.
5. XML is used to transfer data across applications on web
6. XML is language is independent and platform independent

Features of XML :-

1. Self-describing data : Each element and tag in an XML document is self-explanatory. You can easily identify the use of webpage and data contained in the webpage ~~by~~ by looking at code.
2. User-friendly :- You can create your own tags to create a webpage and need not remember any pre-defined tags
3. Viewing data from a standard Tool : You can invoke

an XML document in the Internet Explorer, firefox and all other web browsers used to view data.

4. Uses DTD :- XML uses DTD for defining the elements of XML as well as other languages such as HTML.

5. Content independent :- XML is content independent and does not require you to learn any special programming code to use this language. XML enables you to create your own context-free tags.

6. Platform independent :- XML is platform independent language that can be viewed/run atmost all platforms.

7. Metadata :- XML enables you to provide details about information structure to web sites.

GOALS of XML :-

1. XML should support many number of applications
2. XML should be directly usable over the internet
3. XML should be easier to write programs
4. XML should be used with SGML without any conflict
5. XML documents should be easily and clearly understandable by humans
6. The design of XML documents should be faster to generate
7. The XML document should be clearly and correctly designed

8. The XML document should be simple and easy to create
9. The optional features in XML is kept at absolute minimum (i.e ideally zero)
10. Minimum importance should be given to XML document compactness.

3.2: XML components / Building Blocks of XML :-

Any XML document is just composed of 6 things.

1. Tags
2. Elements
3. Attributes
4. PCDATA
5. CDATA
6. Entities
7. Control information → Comments, Processing instruction, DTD Declarations

1. Tags :- Tags are used to specify a text string that represents an element in an XML document

ex: <author> Winston </author>

Empty tags: In XML, elements can be empty when they do not contain any parsed character data. In the case of an empty elements, the empty tags are must use a forward slash (/) after the name of the element

ex: <fax />

Note: All the tags must be closed by an exact equivalent

because XML is case-sensitive language.

- <author> tag should be closed by </author> tag not by </Author>

Nesting of Tags : Even simple XML document has nesting of tags. They must be nested properly and closed in the reverse order in which they were opened

2. Attributes :- XML attributes store additional information about the data. i.e metadata. (data about data)

- You can use attributes to store IDs, URLs, references and other information not directly relevant to the end user.
- Attributes simply describes the properties of elements.

3. Elements :- XML elements are used to store data.

To create an XML element, you need to follow these naming conventions

* An element must begin with
  - Letters
  - Numbers
  - Underscore
* An element can have special characters such as hyphens (-), Underscore (_), or dot (.).
* There can not be any blank space in tag names.

- Elements can be represented with tags and empty tags

<author> XYZ </author>

simple element.

4. PCDATA :- PCDATA means Parsed Character Data. It is the data which will be suitably parsed by a given parser. In general, PCDATA may be a simple XML file or markup.

5. CDATA :- CDATA means Character Data. this is just opposite to PCDATA. i.e. there is no entity expansion. the CDATA is not parsed at all.

6. Entities :- An Entity is a thing which is used as a part of the document but which is not a simple element.
ex: An image or encrypted digital signature.

7. Control Information :- There are three types of control information

7.1 comments : Comments are used to describe a document inorder to have a proper documentation of the coding. These are ignored by the XML parsers and applications using the XML document. In XML comments are begin with

<!-- and end with -->

ex: <!-- This is a comment -->

- Same type of comment is used for both XML source files and in DTD files.

7.2 Processing Instructions : They used to control applications.

ex: <?xml version="1.0" ?> → XML Prolog

This instruction must be the first statement in your XML document. This instruction tells the applications that data in the file follows the rules of XML version "1.0"

7.3 Document Type Definition (DTD) :- Each valid XML document has associated with DTD. This DTD usually held in a separate file. So it can be used with many documents. The DTD file holds the rules of grammer for a particular XML data structure. Those rules are used by the validating parsers to check that is not only the valid XML file, but it also obeys its own internal rules.

Ex: <!DOCTYPE Course SYSTEM "courses.dtd">

This declaration tells the parser that the XML file is of the courses and it uses a DTD which stored in a file called "courses.dtd"

There are two types of DTDs

1. PUBLIC
2. SYSTEM

- PUBLIC DTDs are pre-defined DTDs. They are confirmed to the international standards such as those W3C used in HTML.
- SYSTEM DTDs: Any DTD which is developed by you or for you.

CREATING XML DOCUMENT :- XML is based on containment model in which each XML element contains

text or other XML elements, called child elements, with in it. When you create an XML code for a webpage, you need to ensure that the element do not contain both data and child elements.

- When you create an XML document, you need to ensure that the document is well-formed
- A well-formed document must have

1. A root element
2. An opening tag and closing tag
3. An appropriate nesting
4. Values always enclosed in the quotation marks.

Syntax

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
    <child>
        <subchild> text </subchild>
    </child>
</root>
```

For example: To create a structure for book, you need a book title, author name and book contents. contents are further divided into sections.

- webpage about book can be structured as follows

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

```
<BookTitle> XML Bible </BookTitle>
<Author> Sam Peter </Author>
<Contents>
    <Chapter1> XML Basics </Chapter1>
    <Chapter2> XML DTDS </Chapter2>
    <Chapter3> XML Schemas </Chapter3>
    <Chapter4> XML Namespaces </Chapter4>
    <Chapter5> XSL Transformations </Chapter5>
</Contents>
</Book>
```


— In the above code, the first instruction is called Processing instruction or XML Prolog. which tells the applications how to handle XML. It also serves as version declaration and says that file is XML which conforms the rules of XML version 1.0.

— The rule states that the parser must halt when it finds an error and that it may return a message back to the calling application.

3.3 Valid or Well-formed :- A well-formed and valid XML document is one which follows all of the below mentioned rules of XML i.e

1. Tags are matched and don't overlap.
2. Empty elements must be ended properly.
3. The document contains XML declaration

4. A valid XML document has its own DTD.

3.4 DOCUMENT TYPE DEFINITION (DTD) :-

The XML has neither meaning nor context without a grammer against which it can be validated. The grammer is called DTD. DTDs are used to give structure to an XML document and valid XML document must have a DTD attached to it. DTD files can be created in text editor and are saved with a ".dtd" extension.

- A DTD can be embedded internally or externally (i.e an external file attached to an XML document)
- Syntax for internal DTD :

```
<!DOCTYPE ROOTELEMENT [ELEMENT Declartions]>
```

- Syntax for external DTD :

```
<!DOCTYPE ROOTELEMENT SYSTEM|PUBLIC "DTD filename">
```

- In the above syntax
  - * the !DOCTYPE indicates to the parser that, this is a DTD file
  - * ROOTELEMENT indicates begining of XML document.
  - * Element declarations specifies information about all of the parent and child elements of the XML document.
  - * SYSTEM| PUBLIC indicates location of the DTD file to the web browser
  - * DTD filename : Specifies name of the DTD file.

Declaring elements in DTD :- The XML document is composed of number of elements in which each of the elements may itself be made from other elements and some of the elements may contain attributes. this structure must be reflected in DTD

- The first node of the XML document is called "root node" root node contains all other elements of document and each xml document must have exactly one root node.
- Each element can either be a container which holds further elements or it can define data.

Syntax

<!ELEMENT elementname type >

- In the above syntax, *) elementname is the name of the element/tag used to markup the element in XML doc.
  *) The "type" specifies the type of the element, also known as the Content Specification model of the element
- An XML supports four types of elements. They are
  1. Empty
  2. Element only
  3. Mixed
  4. Any.

1. Empty : The Empty element does not contain any data Empty elements provide information through attributes. Empty elements are declared as

<!ELEMENT elementname EMPTY>

2. Element only :- This element contains only child elements and no character data. You can declare an element as "Element only" by specifying the content model for the element.

- The content model is a list of child elements and element declaration symbols.
- The element declaration symbols used to specify the content model are

1. Parenthesis (( )) : Enclose a sequence of child nodes / elements
2. comma (,) : Separates items in the order in which they must appear.
3. Pipe (|) : Separates the list of items where only one item can appear at a time
4. No Symbol : It must appear exactly once
5. Question mark (?) : Items must appear exactly once or not at all.
6. Star (*) : The item can appear any number of times.
7. plus (+) : The item must appear at least once.

ex: the content model for Course

<!ELEMENT course (name, (lumsum | installment+), description?, module *) >

3. Mixed elements :- This element contains combination of child elements and character data. The simplest mixed element is one that contains only character data.

Syntax:

<!ELEMENT elementname (#PCDATA) >

PCDATA indicates the element contains parsed character data.

4. ANY Element :- The Any element has virtually no structure and can contain character data.

Syntax:

<!ELEMENT elementname ANY>

DECLARING ATTRIBUTES IN DTD :-

Attributes specify additional information about element and are used inside the tag in the form of name-value pairs

Syntax:

<!ATTLIST elementname attributename attribute type default >

In the above Syntax,

- element name is the name of the element for which the attribute is defined
- attribute name is the name of the attribute
- attribute type defines the type of attribute
- Default specifies the default value of the attribute or a symbol that indicates the use of attribute.
- The values that default can take

1. #REQUIRED: indicates that the attribute is required.
2. #IMPLIED: indicates that the attribute is optional.

3. #FIXED: indicates that the attribute has a fixed value
4. Default: default value of the attribute
5. CDATA: denotes unparsed character data / plain text data
6. Enumerated: denotes a series of string values
7. NOTATION: denotes a notation declared some where else in DTD
8. ENTITY: denotes an external binary entity.
9. ENTITIES: denotes multiple external binary entities separated by white space.
10. ID: denotes a unique identifier
11. NMTOKEN: denotes a name consisting of XML token characters such as letters, numbers, periods, dashes, colons and underscores.
12. NMTOKENS: denotes a series / multiple names consisting of XML token characters.

ex:

```
<!ATTLIST qty amount CDATA #REQUIRED
            unit CDATA "gms" >
```

ENTITIES :- Entities are used to represent complex data. Entity is container which will be filled with some form of content. this content may be include in the XML file

① as an internal entity
② as an external entity (Stored in another file)

Internal Entities: are used to create a small pieces of data which you want to use repeatedly throughout your

schema.

Syntax : <!ENTITY entityname "replacement text">

<!ENTITY pos "pinch of salt" >

& – &
< – <
> – >
&pos – (')
&quot – (")

Usage:

<item> finaly add a &pos; </item>

When an entity is included, the name is preceeded by an ampersand (&) and followed by a semicolon (;)

External Entities :- Almost anything which is data can be included in your XML as an external entity

for example: to add an image to your XML file (8)

<!ENTITY myimage SYSTEM "xy3.jpg" NDATA JPG>

When an XML parser does not understand your entity, a helper application must be specified, the data will then be passed to this helper for processing.

<!NOTATION JPG SYSTEM "mspaint" >

that passes the images to a paint program for viewing.

* Consider the following code that illustrates INTERNAL DTD for Courses.

courses.xml:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE courses [
    <!ELEMENT courses (course)* >
    <!ELEMENT course ( name, ( lumpsum | installment+),
                description?, modules* ) >
    <!ELEMENT name (#PCDATA)>
```

```
<!ELEMENT lumpsum (#PCDATA)>
<!ELEMENT installment (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT modules (#PCDATA)> ]>

  <courses>
    <course>
      <name> M.Tech </name>
      <lumpsum> 42500 </lumpsum>
      <modules> computer science </modules>
      <modules> web programming </modules>
    </course>

    <course>
      <name> B.Tech </name>
      <installment> 20000 </installment>
      <installment> 13000 </installment>
      <description> Undergraduate course </description>
      <modules> Computer basics </modules>
      <modules> Software engg. </modules>
    </course>
</courses>
```

— · —

~~3. XML SCHEMAS:-~~ Limitations of DTD :-

1. DTD itself is not in XML format – more work for parsers
2. Does not express datatypes (weak datatyping)
3. No namespace support
4. Document can override external DTD definitions

5. No Document object model support (DOM)

3.5 XML schemas :-

- The purpose of XML schemas are to define a legal building blocks of XML document just like a DTD.
- XML schemas are successors of DTDs
- XML schemas are XML documents.
- XML schemas will be in most web applications as a replacement for DTDs.
- An XML schema defines
  - → An element that appears in a document
  - → attributes that appears in a document
  - → which elements are child elements
  - → Order of child elements
  - → No. of child elements
  - → whether an element is empty or can include text
  - → Data types for elements and attributes
  - → Default and fixed values for elements and attributes

Advantages / Reasons for use schemas :

XML Schemas are
- extensible to future additions
- richer and more powerful than DTDs
- written in XML
- support data types
- support namespaces.

Syntax:

```
<xsd:element name='ele-name' type='dataType'
    minOccurs='+ve integer' maxOccurs='+ve integer/
    unbounded' />
```

Where name is the name of the element being declared.

- type is the dataType of the element being declared.
- minOccurs specifies the minimum number of times an element can appear.
- maxOccurs specifies the maximum number of time an element can appear; unbounded means an element can appear any number of times.

ex:

```
<xsd:element name='pname' type='xs:string' />
```

Defining complexType Element :- The complexType is one that contains other elements, attributes and mixed content. To declare a complexType,

Syntax:

```
<xsd:complexType>
        skeleton of the complexType.
</xsd:complexType>
```

ex:

```
<xsd:element name="address">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="dno" type="xs:string" />
      <xsd:element name="street" type="xs:string" />
      <xsd:element name="city" type="xs:string" />
    </xsd:sequence>
```

```
</xsd:complexType>
</xsd:element>
```

~~A Simple XML Schema Program for~~

Defining attributes :- Attributes carry additional information regarding elements.

Syntax:

```
<xsd:attribute name='attributename'
               type='datatype' />
```

* A Simple XML Schema that describes product information.

Product.xsd:

```
<?xml version="1.0" encoding='UTF-8" ?>
<xsd:Schema xmlns:xsd = "http://www.w3.org/
    2001/XMLSchema" >
  <xsd:element name ="product" >
   <xsd:complexType >
     <xsd:sequence >
       <xsd:element name="Pname" type ="xs:string" />
       <xsd:element name ="desc" type ='xs:String' />
       <xsd:element name ="price" type = "xs:float" />
       <xsd:element name ="qty" type = "xs:integer" />
     </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
</xsd:Schema>
```

3.5 XSL (Presenting XML) :-

- XSL stands for eXtensible Stylesheet Language
- CSS was designed for Styling HTML pages and can also be used to style XML pages.
- XSL was specifically designed to style XML pages and is much more sophisticated than CSS.
- XSL can convert an XML file into another XML with different. The most comman type of XSL Processing is to convert XML file in HTML file which can be displayed by web browser.

- XSL is the bridge between XML and HTML.

→

- XSL consists of 3 languages. They are:

1. XSLT (XSL Transformations): An XML based language that allows you to transform an XML document into another XML document. (possibly HTML documents).

2. XPath (XML Path) :- XPath is a language to select parts of XML to tranform with XSLT such as elements and attributes

3. XSL-fo (Formatting objects): It is replacement for CSS. objects that specify how data is to be displayed.

How does it work :-

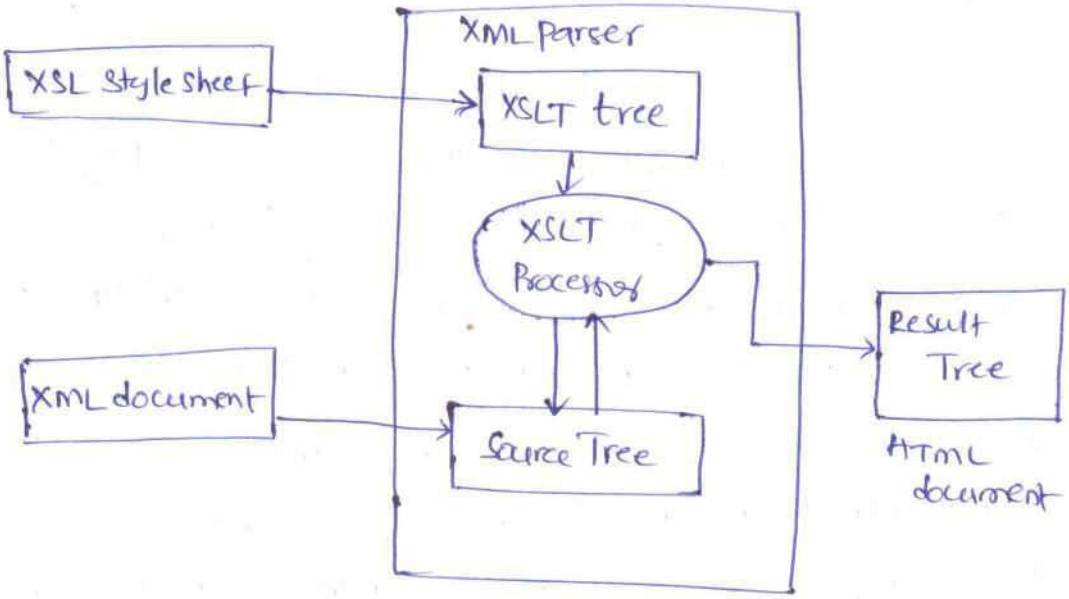


fig: working of XSLT Processor

1. the XML document is parsed into XML Source Tree
2. the XML parser ~~parses~~ parses the XSL stylesheet and create a 'XSLT tree' based on the elements and ~~element~~ attributes used in an XSL document
3. Use xpaths to define templates that matches the parts of the source tree
4. Use XSL processor to transform the matched and put the transformed into result tree.
5. the result tree is output as a ~~xsl~~ result document (HTML document)

- Parts of the source document that not matched by a template are typically copied as unchanged.

The XSL documents must be written ~~on~~ on simple text editor and save with ".xsl" extension

You can add XSL document to XML file by adding the following statement:

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

XSL Elements :- XSLT Provides a number of elements for selecting and formatting data from XML document. and few are used to control the Processor.

- the form of XSL Element which select data.

```
<xsl:element   select = "value" >
```

the value is a pattern which matches a node or set of nodes with in XML document

- Some of the elements provided by XSLT for formatting data are :

* Stylesheet
* value-of
* for-each
* sort
* text.

1. Stylesheet : An XSL Stylesheet contains instructions for transforming XML document. This can be included by using :

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

the Stylesheet element is root element for all XSLT Stylesheet

2. xsl:value-of :- the value-of element displays the value of specified element or attribute

Syntax

```
<xsl:value-of   select = "ele-name/attr-name" />
```

- element name is the name of the element for

for which the value is to be displayed

ex:

```
<xsl:value-of select="Title" />
```

3. xsl:for-each :- for-each element applies a template repeatedly for each node.

Syntax

```
<xsl:for-each select='pattern'>
    action to be performed
</xsl:for-each>
```

In the above Syntax: the pattern can be one of the following:

* element
* parent/child
* ancestor//child

Example:

```
<xsl:for-each select="Title">
  <font color="blue">
   <xsl:value-of select='Pname' />
  </font>
   <xsl:value-of select="price" />
</xsl:for-each>
```

XSL Templates :- A template describes how an XML element and its contents are converted into a format that can be displayed in the browser.

- A template consists of two parts

1. A pattern that identifies an element in XML document

2. Action or processing code that shows the transformation of the resulting element.

1. template element :-

Syntax:

```
<xsl:template match="pattern">
    action to be taken
</xsl:template>
```


the template element is used to define a template for desired output. the pattern can have multiple values. They are:

/$ - starts performing action from root node.

* - matches any element in xml document

ele-name - Performs the action when the named element is encountered.

2. xsl:apply-templates : The apply-templates element is used to instructs the XSLT Processor to find an appropriate node/ template and performs the specified task on each element

Syntax

```
<xsl:apply-templates [select=pattern] >
```

3. xsl:attribute : creates an attribute node. this attribute is then applied to the output document

4. xsl:if : boolean conditions can be tested

5. xsl:when : used in conditional testing

6. xsl:CData : a CDATA section is added to the output document.

3.6 XML Parsers : DOM and SAX :-

- Parsing is the process of reading and validating a program written in one format and converting it into desired format.
- Parsing of XML is process of reading and validating an XML document and converting it into desired format. The program that does this job is called an XML Parser or XML Processor.

- The Parsing Process follows the below steps :
  1. XML Parser reads the XML document from the disk and validates it
  2. Then the XML Parser converts the XML file into an object.
  3. The object is accessed by the application Program &
  4. finally, the application Program produces the desired output.
- Two models are commonly used for Parsers
  1. DOM parser (Document Object Model)
  2. SAX Parser (Simple API for XML)
- These are mostly used by the Java Programmers. Suppose if the application programs are written in Java technology. When an XML Program is presented to a Java Program as

as an object, the application programmers are confused from where it should start parsing and stop it.

- There are two possibilities to solve this:

1. Present the document in bits and pieces, as and when we encounter a certain sections or portions of the document
2. To present the entire document tree at once, where Java program has to think of this document tree one object.

- the first approach, where it goes through XML document node by node till end of the XML file is called (SAX) Simple API for XML

- The second approach, where an entire document is read into the memory as an object and parses its contents as per the requirement is called DOM (Document Object Model)

DOCUMENT OBJECT MODEL (DOM) :-

The DOM is a platform-independent and language-neutral interface that allows documents and scripts to dynamically access and update the contents and style of the document.

- DOM is a collection of nodes with different types of data where nodes are information units that arrange data in an XML document.
- DOM is used to describe the nodes and the relationship between nodes.

- DOM works like an API and is divided into 3 levels

Level 1: provides basic functionality for navigating and manipulating HTML and XML documents such as methods for adding, moving and reading information.

Level 2: It introduces support for namespaces and for stylesheets such as CSS and storing data in a hierarchical structure.

Level 3: It produces complete mapping between DOM and XML.

- It reads the entire document and builds DOM tree
- It provides random access to any nodes in the document DOM tree.
- Allows programs and scripts to build documents, navigate their structure and modify or delete elements and content.

XML DOM :-

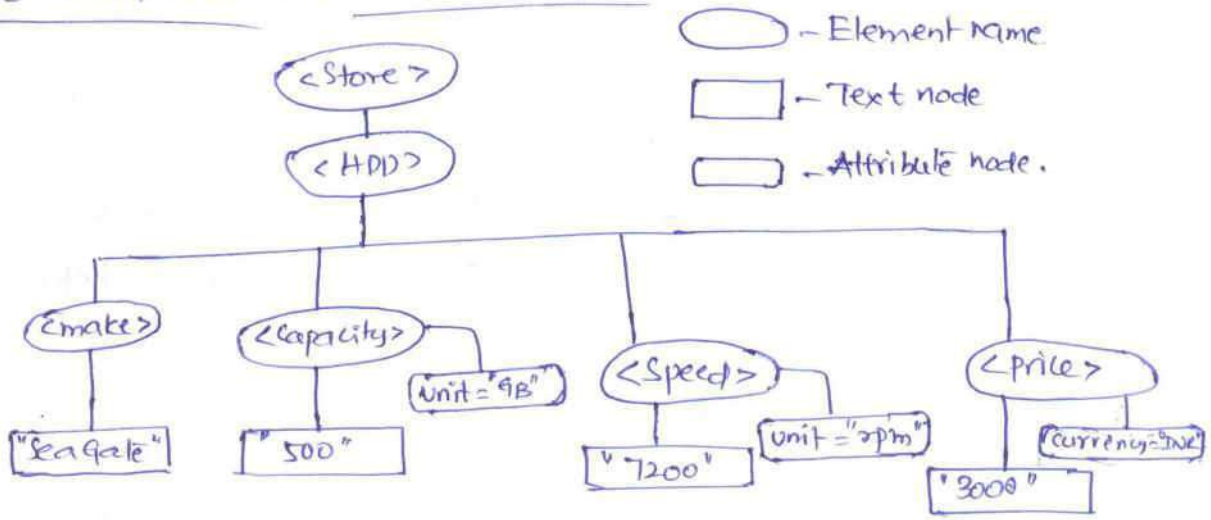
- The XML DOM is a standard object application programming interface (API) that gives developers a pacer to control the structure, content and format of XML document.

- When a DOM parser successfully parses an XML document, the parser create a tree structure in memory that contains document data.
- A DOM Tree has a single root node, that contains all other nodes in the document

- Each node is an object that has properties, methods and events
- Properties are associated with node provide access to the node's name, value and child nodes.
- Methods allocos developers to create, append and delete nodes, load xml document etc and so on.
- Let us explain the XML DOM tree with example. consider the following simple XML document that contains the specification of a hard disk

```
<?xml version="1.0" encoding="UTF-8" ?>
<Store>
  <HDD type="SATA">
    <make> SeaGate </make>
    <Capacity unit="GB"> 500 </Capacity>
    <Speed unit="rpm"> 7200 </Speed>
    <Price currency="INR"> 3000 </Price>
  </HDD>
</Store>
```


- the DOM Tree for above document



- Here Store is root node, HDD is child node where make, Price, Capacity and Speed are Siblings.

* Differences between DOM and SAX :-

DOM	SAX
1. It is a Tree-model parser.	1. It is event-based parser.
2. Dom parser Stores an entire XML document in memory before parsing	2. SAX parser can read XML document node by node.
3. DOM is read and write.	3. SAX is read-only.
4. Dom is preferred when XML file is Small in Size.	4. SAX is preferred when XML file is large in Size.
5. Backward and forward Searching is possible in Dom	5. SAX reads the file from top to bottom and backward navigation is not possible
6. It takes more amount of memory	6. It takes less amount of memory
7. DOM is Slow at runtime	7. SAX is fast and efficient
8. It uses the following packages in java. import javax.xml.parsers.*; import org.w3c.dom.*;	8. It uses the following packages in java. import org.xml.sax.*; import org.xml.sax.helpers .*;

AJAX

AJAX is an acronym for **Asynchronous JavaScript and XML**. It is a group of inter-related technologies like JavaScript, DOM, XML, HTML, CSS etc.

AJAX allows you to send and receive data asynchronously without reloading the web page. So it is fast.

AJAX allows you to send only important information to the server not the entire page. So only valuable data from the client side is routed to the server side. It makes your application interactive and faster.

- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background

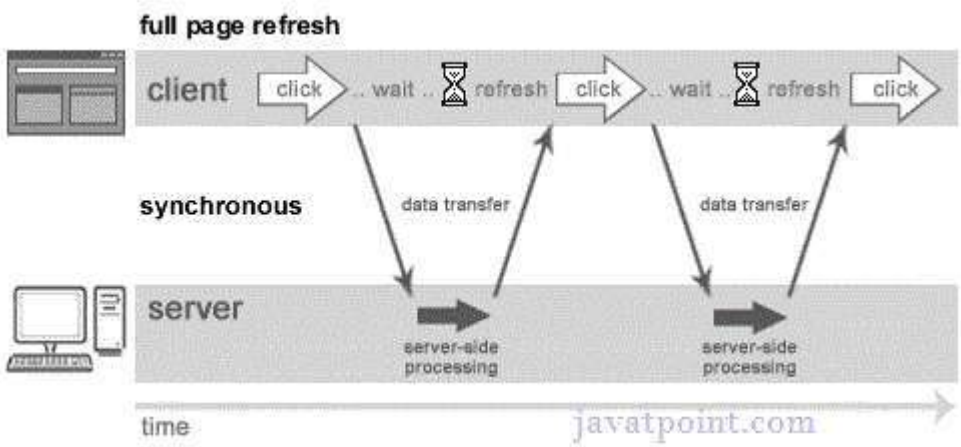
Where it is used?

There are too many web applications running on the web that are using ajax technology like **gmail**, **facebook**,**twitter**, **google map**, **youtube** etc.

Before understanding AJAX, let's understand classic web application model and ajax web application model first.

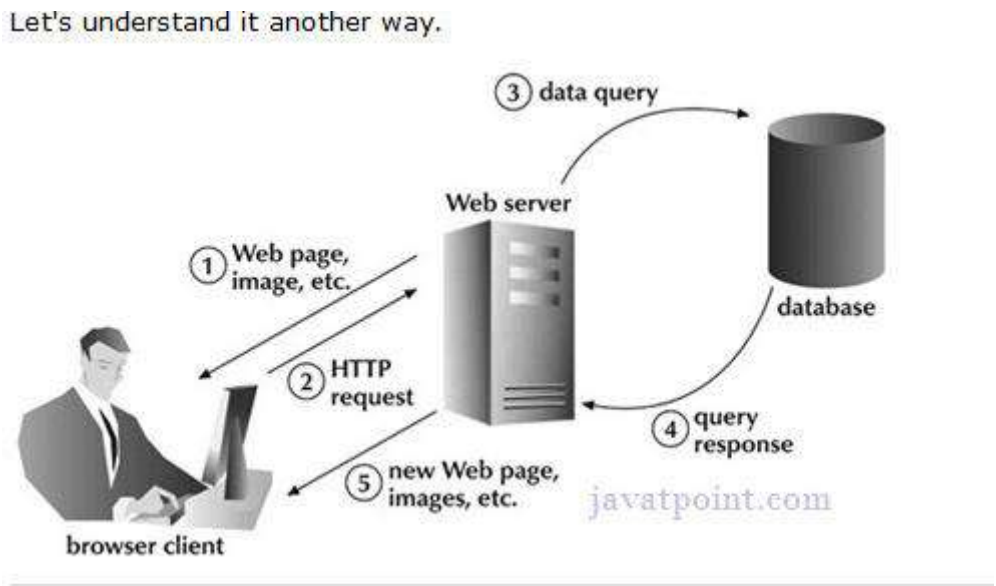
Synchronous (Classic Web-Application Model)

A synchronous request blocks the client until operation completes i.e. browser is not unresponsive. In such case, javascript engine of the browser is blocked.



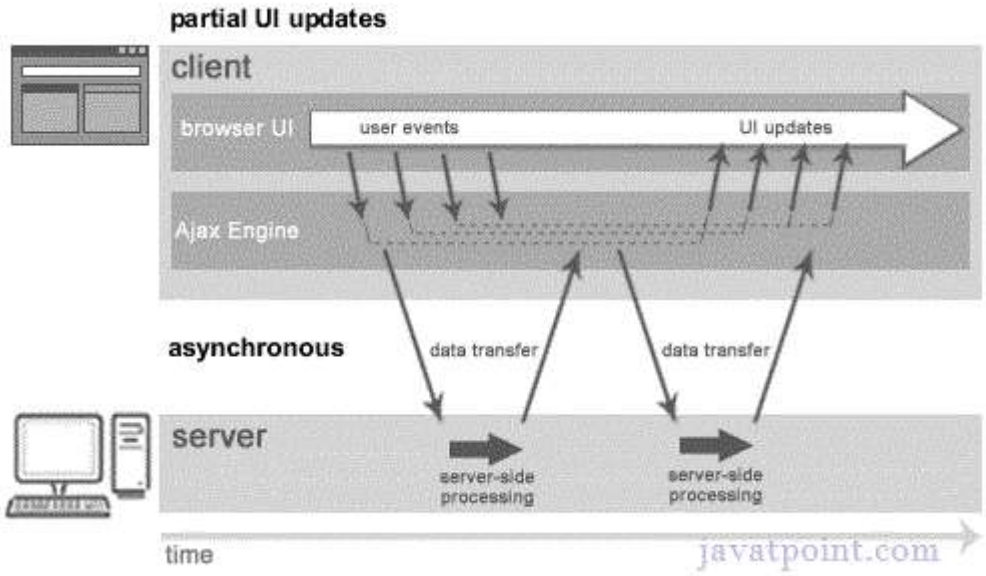
As you can see in the above image, full page is refreshed at request time and user is blocked until request completes.

Let's understand it another way.



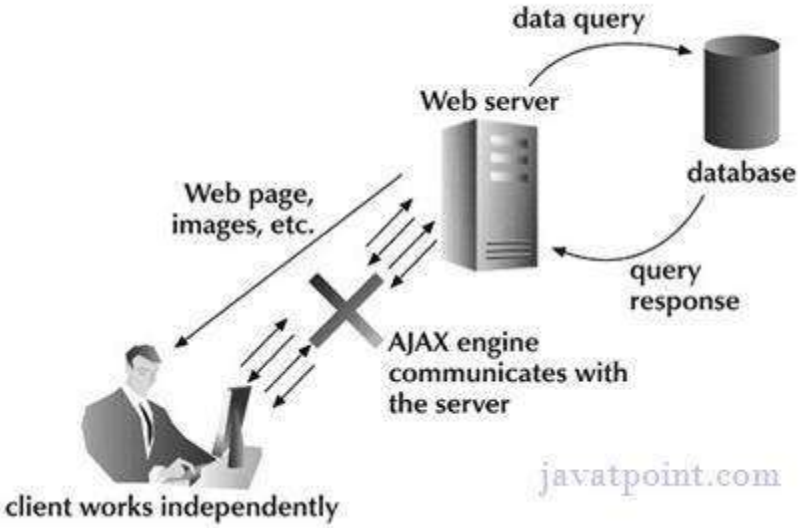
Asynchronous (AJAX Web-Application Model)

An asynchronous request doesn't block the client i.e. browser is responsive. At that time, user can perform another operations also. In such case, javascript engine of the browser is not blocked.



As you can see in the above image, full page is not refreshed at request time and user gets response from the ajax engine.

Let's try to understand asynchronous communication by the image given below.



AJAX Technologies

As describe earlier, ajax is not a technology but group of inter-related technologies. AJAX technologies includes:

- HTML/XHTML and CSS
- DOM
- XML or JSON
- XMLHttpRequest
- JavaScript

HTML/XHTML and CSS

These technologies are used for displaying content and style. It is mainly used for presentation.

DOM

It is used for dynamic display and interaction with data.

XML or JSON

For carrying data to and from server. JSON (Javascript Object Notation) is like XML but short and faster than XML.

XMLHttpRequest

For asynchronous communication between client and server. For more visit next page.

JavaScript

It is used to bring above technologies together.

Independently, it is used mainly for client-side validation.

XMLHttpRequest

An object of XMLHttpRequest is used for asynchronous communication between client and server.

It performs following operations:

1. Sends data from the client in the background
2. Receives the data from the server
3. Updates the webpage without reloading it.

Properties of XMLHttpRequest object

The common properties of XMLHttpRequest object are as follows:

Property	Description
onReadyStateChange	It is called whenever readystate attribute changes. It must not be used with synchronous requests.
readyState	represents the state of the request. It ranges from 0 to 4. **0** UNOPENED open() is not called. **1** OPENED open is called but send() is not called. **2** HEADERS_RECEIVED send() is called, and headers and status are

	available. **3** LOADING Downloading data; responseText holds the data. **4** DONE The operation is completed fully.
reponseText	returns response as text.
responseXML	returns response as XML

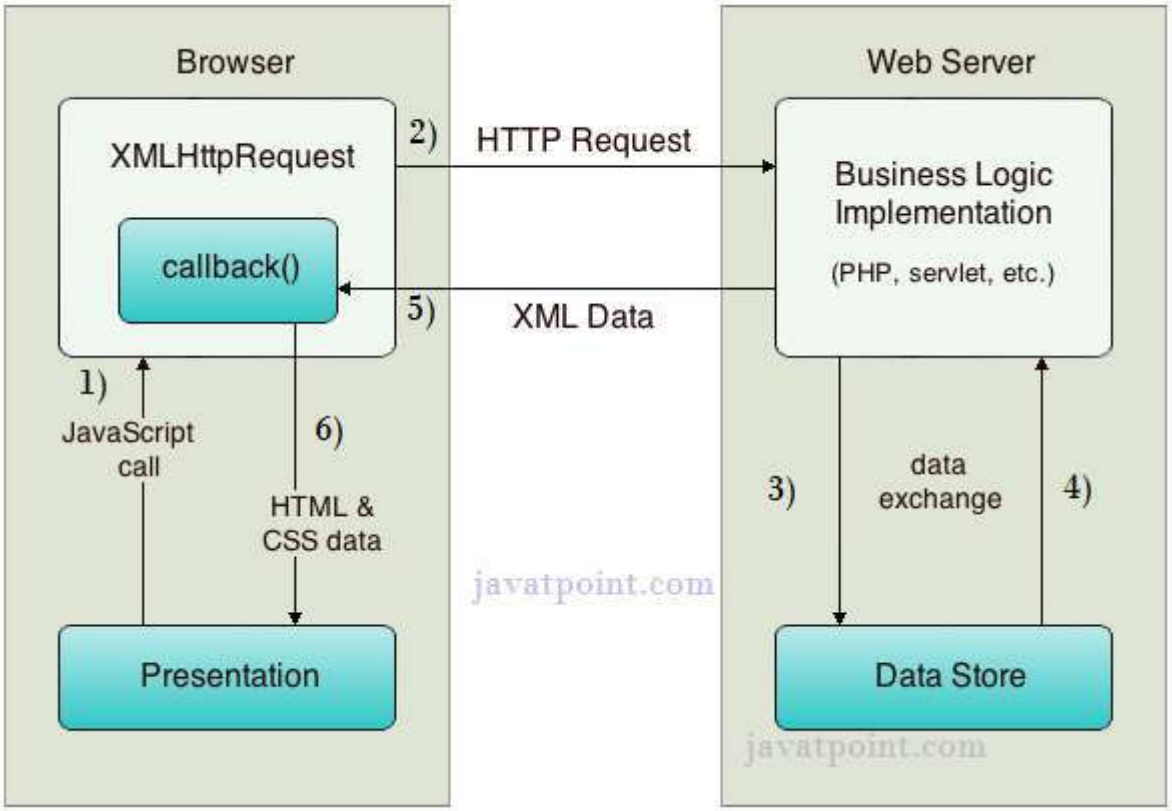
Methods of XMLHttpRequest object

The important methods of XMLHttpRequest object are as follows:

Method	Description
void open(method, URL)	opens the request specifying get or post method and url.
void open(method, URL, async)	same as above but specifies asynchronous or not.
void open(method, URL, async, username, password)	same as above but specifies username and password.
void send()	sends get request.
void send(string)	send post request.
setRequestHeader(header,value)	it adds request headers.

How AJAX works?

AJAX communicates with the server using XMLHttpRequest object. Let's try to understand the flow of ajax or how ajax works by the image displayed below.



As you can see in the above example, XMLHttpRequest object plays a important role.

1. User sends a request from the UI and a javascript call goes to XMLHttpRequest object.
2. HTTP Request is sent to the server by XMLHttpRequest object.
3. Server interacts with the database using JSP, PHP, Servlet, ASP.net etc.
4. Data is retrieved.
5. Server sends XML data or JSON data to the XMLHttpRequest callback function.
6. HTML and CSS data is displayed on the browser.

Web Service Components

There are three major web service components.

1. SOAP
2. WSDL
3. UDDI

SOAP

SOAP stands for Simple Object Access Protocol. It is a XML-based protocol for accessing web services.

SOAP is a W3C recommendation for communication between two applications.

SOAP is XML based protocol. It is platform independent and language independent. By using SOAP, you will be able to interact with other programming language applications.

Advantages of Soap Web Services

WS Security: SOAP defines its own security known as WS Security.

Language and Platform independent: SOAP web services can be written in any programming language and executed in any platform.

Disadvantages of Soap Web Services

Slow: SOAP uses XML format that must be parsed to be read. It defines many standards that must be followed while developing the SOAP applications. So it is slow and consumes more bandwidth and resource.

WSDL dependent: SOAP uses WSDL and doesn't have any other mechanism to discover the service.

WSDL

WSDL is an acronym for Web Services Description Language.

WSDL is a xml document containing information about web services such as method name, method parameter and how to access it.

WSDL is a part of UDDI. It acts as a interface between web service applications.

WSDL is pronounced as wiz-dull.

Features of WSDL

- WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.
- WSDL definitions describe how to access a web service and what operations it will perform.
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.
- WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'.

WSDL Elements

A WSDL document contains the following elements:

- **Definition** : It is the root element of all WSDL documents. It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements described here.
- **Data types** : The data types to be used in the messages are in the form of XML schemas.
- **Message** : It is an abstract definition of the data, in the form of a message presented either as an entire document or as arguments to be mapped to a method invocation.
- **Operation** : It is the abstract definition of the operation for a message, such as naming a method, message queue, or business process, that will accept and process the message.
- **Port type** : It is an abstract set of operations mapped to one or more end-points, defining the collection of operations for a binding; the collection of operations, as it is abstract, can be mapped to multiple transports through various bindings.
- **Binding** : It is the concrete protocol and data formats for the operations and messages defined for a particular port type.
- **Port** : It is a combination of a binding and a network address, providing the target address of the service communication.
- **Service** : It is a collection of related end-points encompassing the service definitions in the file; the services map the binding to the port and include any extensibility definitions.

In addition to these major elements, the WSDL specification also defines the following utility elements:

- **Documentation:** This element is used to provide human-readable documentation and can be included inside any other WSDL element.
- **Import** : This element is used to import other WSDL documents or XML Schemas.

UDDI

- UDDI stands for **Universal Description, Discovery, and Integration.**
- UDDI is a specification for a distributed registry of web services.
- UDDI is a platform-independent, open framework.
- UDDI can communicate via SOAP, CORBA, Java RMI Protocol.
- UDDI uses Web Service Definition Language(WSDL) to describe interfaces to web services.
- UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services.
- UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet.

UDDI has two sections:

- A registry of all web service's metadata, including a pointer to the WSDL description of a service.

- A set of WSDL port type definitions for manipulating and searching that registry.

A business or a company can register three types of information into a UDDI registry. This information is contained in three elements of UDDI.

These three elements are:

- White Pages,
- Yellow Pages, and
- Green Pages.

White Pages

White pages contain:

- Basic information about the company and its business.
- Basic contact information including business name, address, contact phone number, etc.
- A Unique identifiers for the company tax IDs. This information allows others to discover your web service based upon your business identification.

Yellow Pages

- Yellow pages contain more details about the company. They include descriptions of the kind of electronic capabilities the company can offer to anyone who wants to do business with it.
- Yellow pages uses commonly accepted industrial categorization schemes, industry codes, product codes, business identification codes and the like to make it easier for companies to search through the listings and find exactly what they want.

Green Pages

Green pages contains technical information about a web service. A green page allows someone to bind to a Web service after it's been found. It includes:

- The various interfaces
- The URL locations
- Discovery information and similar data required to find and run the Web service.

NOTE : UDDI is not restricted to describing web services based on SOAP. Rather, UDDI can be used to describe any service, from a single webpage or email address all the way up to SOAP, CORBA, and Java RMI services.

UDDI - Technical Architecture

The UDDI technical architecture consists of three parts:

UDDI Data Model

UDDI Data Model is an XML Schema for describing businesses and web services. The data model is described in detail in the "UDDI Data Model" chapter.

UDDI API Specification

It is a specification of API for searching and publishing UDDI data.

UDDI Cloud Services

These are operator sites that provide implementations of the UDDI specification and synchronize all data on a scheduled basis.

The UDDI Business Registry (UBR), also known as the Public Cloud, is a conceptually single system built from multiple nodes having their data synchronized through replication.

The current cloud services provide a logically centralized, but physically distributed, directory. It means the data submitted to one root node will automatically be replicated across all the other root nodes. Currently, data replication occurs every 24 hours.

UDDI cloud services are currently provided by Microsoft and IBM. Ariba had originally planned to offer an operator as well, but has since backed away from the commitment. Additional operators from other companies, including Hewlett-Packard, are planned for the near future.

It is also possible to set up private UDDI registries. For example, a large company may set up its own private UDDI registry for registering all internal web services. As these registries are not automatically synchronized with the root UDDI nodes, they are not considered as a part of the UDDI cloud.

UDDI - With WSDL

The UDDI data model defines a generic structure for storing information about a business and the web services it publishes. The UDDI data model is completely extensible, including several repeating sequence structures of information.

However, WSDL is used to describe the interface of a web service. WSDL is fairly straightforward to use with UDDI.

- WSDL is represented in UDDI using a combination of *businessService*, *bindingTemplate*, and *tModel* information.
- As with any service registered in UDDI, generic information about the service is stored in the *businessService* data structure, and information specific to how and where the service is accessed is stored in one or more associated *bindingTemplate*

structures. Each *bindingTemplate* structure includes an element that contains the network address of the service and has associated with it one or more *tModel* structures that describe and uniquely identify the service.

- When UDDI is used to store WSDL information, or pointers to WSDL files, the *tModel* should be referred to by convention as type *wsdlSpec*, meaning that the *overviewDoc* element is clearly identified as pointing to a WSDL service interface definition.
- For UDDI, WSDL contents are split into two major elements the interface file and the implementation file.

The Hertz reservation system web service provides a concrete example of how UDDI and WSDL works together. Here is the <tModel> for this web service: