UNIT-5
AJAX and XML

AJAX stands for Asynchronous JavaScript And XML, and it describes a set of development techniques used for building websites and web applications. With the help of AJAX user's web browser doesn't need to reload an entire web page when only a small portion of content on the page needs to change. With the help of AJAX we can:

→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.

AJAX is not a programming language rather it is a technology or method which uses a combination of:

→A browser built-in XMLHttpRequest object (to request data from web server).
→JavaScript and HTML DOM (to display or use the data).

AJAX Working:

→An event occurs in a web page. (for e.g. a button is clicked).
→An XMLHttpRequest object is created by JavaScript.
→The XMLHttpRequest object sends a request to web server.
→The server processes the request and sends response back to the web page.
→The response is read by JavaScript and proper action is performed by JavaScript (like page update).

Advantages of AJAX:

i) Speed → It reduces server traffic in both side request as well as time consuming on both side response.
ii) Interaction → AJAX is much responsive.
iii) Asynchronous calls → AJAX makes asynchronous calls to web server.
iv) Form Validation → Form are common element in web page. Validation should be instant and properly, AJAX gives us all of that.
v) Bandwidth Usage → AJAX does not require to reload an entire webpage when only small portion of content needs to change. Hence saves bandwidth.

Disadvantages of AJAX:-

→ Search engines would not be able to index an AJAX application.
→ The server information can not be accessed within AJAX.
→ AJAX is not well integrated with any browser.
→ ActiveX requests are enabled only in IE5 and IE6.
→ Data of all requests is URL-encoded, which increases size of the request.

← Below this ~~lesse~~ upto end of page lesser imp as exam point of view

XMLHttpRequest Object:- All the modern browsers support the XMLHttpRequest object. The XMLHttpRequest object can be used to exchange data with the server.

Syntax for creating an XMLHttpRequest object:

```
variable_name = new XMLHttpRequest();
```

Example:

```
var xhr = new XMLHttpRequest();
```

AJAX Request:- To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

Example:

```
xhr.open("GET", "ajax_info.txt", true);
xhr.send();
```

Method	Description
open(method, url, async)	Specifies type of request. method → the type of request: GET or POST url → the server (file) location. async → true (asynchronous) or false (synchronous).
send()	Sends the request to the server (used for GET)
send(string)	Sends the request to the server (used for POST).

AJAX Response:- The responseText property returns the server response as a JavaScript string, and we can use it accordingly:

Example:- `document.getElementById("demo").innerHTML = xhr.responseText;`

The responseXML property returns the server response as an XML DOM object. getResponseHeader() and getAllResponseHeaders() are the server response methods.

⊛. Introduction to XML and its Application:-

XML stands for eXtensible Markup Language. XML was designed to store and transport data. XML was designed to be both human and machine readable. XML is designed to carry data, not to display data. XML tags are not predefined, we must define our own tags. XML is platform independent and language independent.

Benefits of using XML:-

i) Simplicity → Information coded in XML is easy to read and understand, and it can be processed easily by computers.

ii) Openness → XML is W3C standard, endorsed by software industry market leaders.

iii) Extensibility → There is no fixed set of tags. New tags can be created as they are needed.

iv) Self-description → XML documents contain meta data.

v) Seperates content from presentation → XML tags describe meaning not presentation.

vi) Can embed multiple data types → XML documents can contain any possible data type - from multimedia data to active components.

Applications of XML:-

i) Vector Markup Language (VML) → It is a language based on XML, which allows us to describe and format vectors. It was designed to help the developers address book problems with binary graphic files. The great application of VML is the creation of menus, and navigational elements.

ii) XML based Variant Configuration Language (XVCL) → Variants generally occur in software reuse. XVCL allows us to configure product variants from common core generic, reusable Meta components.

iii) SMIL 2.0 → It allows us to use a text editor to create reusable content such as audio, video that can be delivered through Web.

iv) XML based directories for MXF/AAF applications → Advanced Authoring Format (AAF) is a binary file format for post production environment. It has a model that can be used for describing video, audio etc..

⊛ Syntax Rules for creating XML document: [Imp]

Rule 1: All XML Must Have a Root Element

A root element is simply a set of tag that contains our XML content. In example below <book> is root element.

Example: <book>
<author> James Joyce </author>
<price> 1200 </price>
</book>

Rule 2: All tags must be closed → If we open any tag in some order then we need to close those tags. In above example opening tags <book>, <author>, <price> are closed with closing tags </book>, </author>, </price>.

Rule 3: All tags must be properly nested → When we nest one tag within another, we should pay attention to the order in which we open each tag, and then we close tags in the reverse order.

Example :- <A> <B> Text </B> </A>.

Rule 4: Tag names are case sensitive. → Uppercase and lowercase matter in XML. Opening and closing tags must match exactly. For example; <ROOT>, <Root>, and <root> are three different tags.

Rule 5: Tag Names cannot contain spaces.

Rule 6: Attribute values must appear within Quotes:

For example:- <artist title="author" nationality="USA">

Note:- XML comments are specified in a same way as HTML Comments by: <!-- Comments here -->.

⊛ **XML Elements:-** An XML document contains XML elements. An XML element is everything from the elements start tag to end tag including start and end tags. For example: <price> 29.99 </price> An element can contain text, attributes, other elements, or a mix of these. An element with no content is said to be empty XML element. For example: <price> </price>. Empty elements can have attributes. XML elements are case-sensitive, start with letter or underscore, but cannot contain spaces.

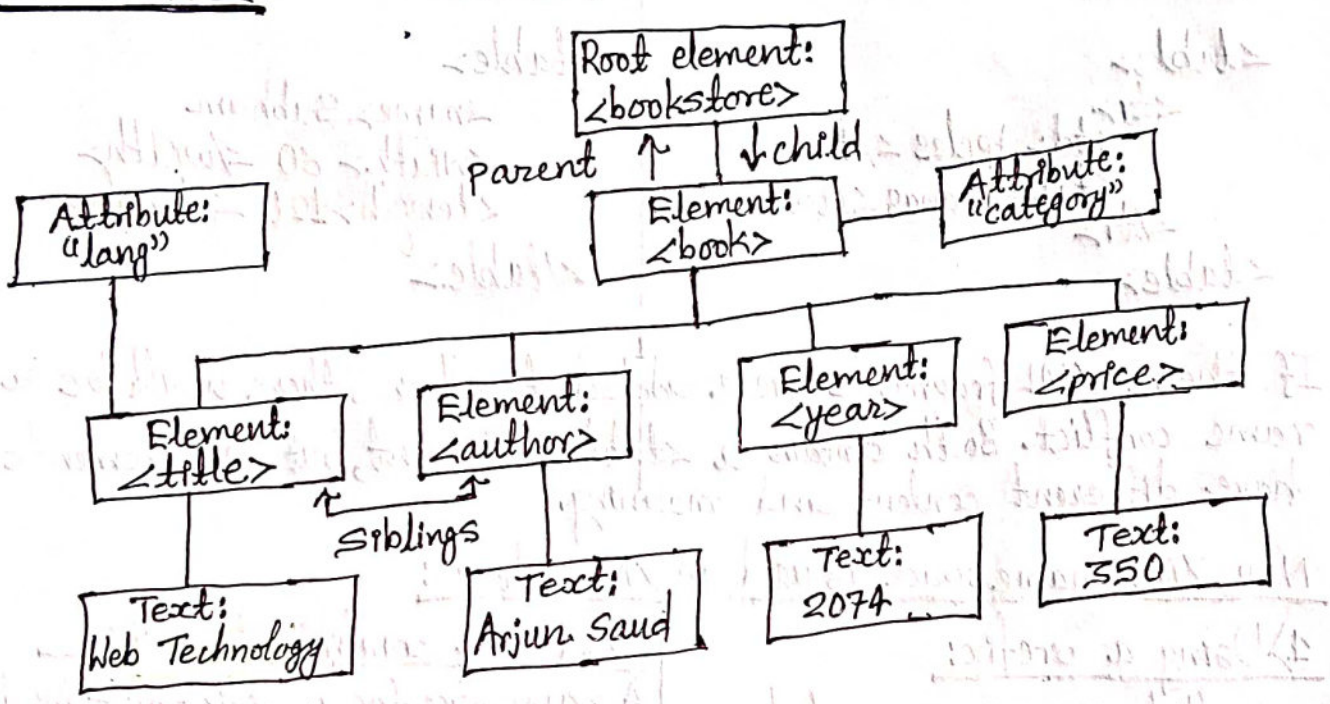
⊛. XML Attributes:-

XML elements can have attributes, just like html. Attributes are designed to contain data related to a specific element. Attribute values must be quoted. Either single or double quotes can be used. Attributes cannot contain multiple values. and tree structures & are not easily expandable.

Example:- <person gender="female">

Here gender is attribute and female is attribute value.

⊛. XML Tree



⊛XML Syntax:-

XML document must contain one root element which is parent of all elements.

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

XML Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
    <to>Students</to>
    <from>Roshan</from>
    <heading>Remainder</heading>
    <body>Don't forget me.</body>
</note>
```

⊛. XML Namespace:- [Imp]

XML Namespace is a mechanism to avoid name conflicts by differentiating elements or attributes within an XML document that may have identical names, but different definitions. XML data has to be exchanged between several applications. Same tag name may have different meanings in different applications. So, it creates confusion on exchanging documents. So, XML namespace is used in XML file as: Using a prefix & Using xmlns attribute.

Consider two XML fragment:

```
//1.xml
<table>
    <tr>
        <td>Apples</td>
        <td>Bananas</td>
    </tr>
<table>
```

```
//2.xml
<table>
    <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.

Now XML namespace is used in XML file as:

1) Using a prefix:

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

Example:

```
<h:table>
    <h:tr>
        <h:td>Apples</h:td>
        <h:td>Bananas</h:td>
    </h:tr>
</h:table>

<f:table>
    <f:width>80</f:width>
    <f:length>120</f:length>
</f:table>
```

In this example there will be no conflict because the two <table> elements have now different names.

2) Using xmlns Attribute

A namespace for a prefix must be defined by an xmlns attribute in start tag of element. Syntax: xmlns:prefix="URI".

Example:

```
<root>
<h:table xmlns:h="http://w3.org">
    <h:tr>
        <h:td>Apples</h:td>
        <h:td>Bananas</h:td>
    </h:tr>
</h:table>

<f:table xmlns:f="http://w3schools.com">
    <f:width>80</f:width>
    <f:length>120</f:length>
</f:table>
</root>
```

⊛. XML schema languages: DTD and XSD:- [V.Imp]

1) Document Type Definition (DTD):- The XML Document Type Declaration, commonly known as DTD, is a way to describe XML language precisely. DTD's check vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XML language. An XML DTD can be either specified inside the document, or it can be kept in a seperate document and then linked separately.

Syntax: Basic syntax of DTD is as follows:-

```
<!DOCTYPE element DTD identifier
[
   declaration1
   declaration2
   ......
]>
```

In above syntax,

- → The **DTD** starts with <!DOCTYPE delimeter.
- → An **element** tells the parser to parse the document from specified root element.
- → The square **brackets** [ ] enclose an optional list of entity declarations called internel subset.

2) Internal DTD: A DTD is referred to as internal DTD if elements are declared within the XML files. To refer it as internal DTD, standalone attribute in XML declaration must be set to **yes**. This means, the declaration works independent of external source.

Syntax: <! DOCTYPE root-element [element-declarations]>
where root-element is the name of root element.

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE address[
   <!ELEMENT address (name,company,phone)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT company (#PCDATA)>
   <!ELEMENT phone (#PCDATA)>
]>
```

If simple example of DTD is asked then we can write this example as it is just by removing standalone="yes" keeping all other things same.

```
<address>
    <name>Roshan Bist</name>
    <company>NoteJunction</company>
    <phone>9806470952</phone>
</address>
```

ii) **External DTD:** In external DTD elements are declared outside the XML file. They are accessed by specifying the system attributes which may be either the legal .dtd file or a valid URL. To refer it as external DTD, standalone attribute in the XML declaration must be set as no. This means, declaration includes information from the external source.

Syntax: `<!DOCTYPE root-element SYSTEM "file-name">`
where file-name is the file with .dtd extension.

Example:
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE address SYSTEM "address.dtd">
<address>
    <name>Roshan Bist</name>
    <company>NoteJunction</company>
    <phone>9806470952</phone>
</address>
```

The content of the DTD file **address.dtd** is as shown below:

```
<!ELEMENT address (name, company, phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
```

Elements in DTD:-

In a DTD elements are declared with an ELEMENT declaration. In DTD, XML elements are declared with following syntax:

```
<!ELEMENT element-name category>
```
OR
```
<!ELEMENT element-name (element-content)>
```

Example: Write same example of internal DTD just by shortening first line ~~others~~ as follows and all other things same:

```
<?xml version="1.0">
    ⋮ //All other lines same.
```

Attributes in DTD: In DTD, attributes are declared with an ATTLIST declaration. An attribute declaration has the following syntax:

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

Example: <!ATTLIST employee id CDATA #REQUIRED>

XML example: <employee id="001" />

element-name → Specifies the name of the element to which the attribute applies.

attribute-name → Specifies the name of the attribute which is included with the element-name.

2) XML Schema Definition (XSD):-

Define XML schema करने के लिए

XML schema is a language which is used for expressing constraint about XML documents. It is used to define the structure of an XML document. It is like DTD but provides more control on XML structure. XML schema is commonly known as XML Schema Definition (XSD).

XSD Definition Types:-

Definition types दो तरह के simple और complex

We can define XML schema elements mainly by following two types:

1) Simple Type: Simple type element is used only in the context of text. Some of the predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date.

Example: <xs:element name="student_name" type="xs:int" />

2) Complex Type: A complex type is a container for other element definitions. This allows us to specify which child elements an element can contain and to provide some structure within our XML documents. In the below example, contact element consists of child elements. This is a container for other <xs:element> definitions, that allows us to build a simple hierarchy of elements in the XML document.

Example:-

```
<xs:element name="contact">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:element name="company" type="xs:string" />
      <xs:element name="phone" type="xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Complete Example for XSD or how to use schema:- (XSD Example):

Same as above example only first two lines added to make it complete-

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="contact">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="company" type="xs:string" />
        <xs:element name="phone" type="xs:int" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

XSD Elements: Elements are the building blocks of XML document. An element can be defined within an XSD as follows:-

```
<xs:element name="x" type="y" />
```

XSD Attributes: Attributes in XSD provide extra information within an element. Attributes can be defined within an XSD as follows:-

```
<xs:attribute name="x" type="y" />
```

⊛. Differences between XML schema and DTD OR Compare XML schema with DTD. [Imp]

→ XML schemas are written in XML while DTD are derived from SGML syntax.

→ XML schemas define datatypes for elements and attributes while DTD doesn't support datatypes.

→ XML schemas allow support for namespaces while DTD does not.

→ XML schemas define number and order of child elements, while DTD does not.

→ XML schemas can be manipulated with XML DOM but it is not possible in case of DDT.

→ XML schemas are extensible while DDT is not extensible.

⊛. XML Style Sheets (XSLT):- [Imp]

first para. describes use of XSLT

2nd para describes working of XSLT

XSLT is a language for transforming XML documents into other XML documents, or other formats such as HTML for web pages, plain text or XSL formatting objects, which may subsequently be converted to other formats such as PDF, Postscript and PNG.

The original document is not changed; rather, a new document is created based on the content of an existing one. Typically, input documents are XML files, but anything from which the processor can build an XQuery and XPath data model can be used, such as relational database tables or geographical information system.

Why do we need XSLT?

⇒ XSLT provides the ability to transform XML data from one format to another automatically. It is the recommended style sheet language for XML.

⇒ XSLT is far more sophisticated than CSS. With XSLT we can add/remove elements and attributes to or from the output file.

⇒ We can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.

⇒ XSLT uses XPath to find information in an XML document.

⊛. Xquery :-

XQuery is a functional query language used to retrive information stored in XML format. It is same as for XML what SQL is for databases. It was designed to query XML data. XQuery is built on XPath expressions. It is a W3C recommendation which is supported by all major databases.

Uses of XQuery:

→ To extract information to use in web service.
→ To generate summary reports.
→ To transform XML data to XHTML.
→ To search web documents for relevant information.

XQuery Features:

→ It is a functional language used to retrive and query XML based data.
→ It is expression-oriented programming language with a simple type system.
→ It is analagous to SQL.
→ It uses XPath expressions to navigate through XML documents.
→ It is a W3C standard and universally supported.

XQuery Advantages:

→ Can be used to retrive both hierarchial and tabular data.
→ Can be used to build web pages.
→ Can be used to query web pages.
→ Can be used to query tree and graphical structures.
→ Can be used to transform XML documents into XHTML documents.

XPath: XPath is a language that describes a way to locate and process items in XML documents by using an addressing syntax based on a path through the documents logical structure or hierarchy. This makes writing programming expressions easier. XPath allows the programmer to deal with the document at a higher level of abstraction. It is a language that is used by and specified as a part of both the XSLT and XPointer. XPath stands for XML Path language and is a W3C recommendation.

W3C → World Wide Web Consortium ← full form