Unit-2:-

XML Introduction: introduction of XML, Defining XML tags, their attribubes and values, Document typedefination XML schemes, Document object model, XHTML, parsing XML Data-DOM and SAX parsers in Java.

① Introduction of XML:-

→ XML stands for Extensible Markup language.

→ XML is a markup language much like HTML.

→ XML was designed to describe data.

→ XML Tags are not predefined in XML. You must define your own tags.

→ XML is self-describing

→ XML uses a DTD to formally describe the data.

→ XML is widely used in web services to transport data over the network.

→ XML is very easy to parse and generate.

→ It provides strong support for unicode characters.

→ XML is widely used in SOA.

→ The default character encoding is UTF-8 for XML documents.

Advantages:-

① It made easy to transport and share data.

② It simplifies the platform change process.

3) It enhances data availbility

4) It supports multilingual documents and unicode.

5) provides relatively easy to learn and Code.

6) XML improves the exchange of data between various platforms

7) It performs validation using DTD and schema.

8) XML separates the data from HTML.

9) ~~I~~ XML indicates how the XML document should look after it is displayed.

Disadvantages:-

1) XML requires a processing application

2) No intrinsic data type support

3) The XML syntax is redundant

4) Does not allow user to create his tags

5) XML doesnot support array.

6) XML's simplicity makes it easy to lose sight of the whole design process.

7) xml can quickly become difficult to read if a lot of information is included in one file.

Features of XML:-

① XML separates data from HTML

- XML allows the data to be stored in separate XML files.

② XML simplifies data sharing

- XML data are kept in plain text format. This provides a software and hardware independent approach to data storage.

- This makes it much easier to create data which different applications can share.

③ XML simplifies data Transport

- The difficulty is significantly minimized by the sharing of data as XML, as the data can be interpreted by different incomptaible applications.

④ XML simplifies platform change

- XML data is kept in text format.

⑤ XML increases data availability

- You can view the data from Various applications, not just through html pages but also from xml data sources.

⑥ xml can be eused to create new internet languages.

- For XML Several new internet languages are being created. They are.

1) xHTML
2) RSS New feeds languages
3) WSDL to identify web services availble;

② **Defining XML Tags:—**

XML tags are one of the important building blocks of XML. we should keep in mind that, XML tags are case sensitive and every starting tag should have a closing tag. A tag starts through <> characters like in HTML and ends through </>. The text between starting and ending tag is called as its ~~innocent~~ content. Tags without any content in it are called as empty tags.

start tag:— Beginning of every non-empty XML element is marked by a start-tag.

eg:<address>

End tag:— every element that has a start tag end with an end-tag.

eg:</address>

Empty tag:— An element which has no content.

Eg: <hr></hr>

For example,

<book> starting tag
Basics of computers content
</book> Ending tag
<book></book> Empty tag.

XML Tags Rules:-

Rule 1:-

XML tags are case-sensitive.

```
<address> This is Wrong syntax </address>
```

because of the case difference in two tags, which is treated as erroneous syntax in XML.

```
<address> This is correct syntax </address
```

Where we use the same case to name the start and the end tag.

Rule-2:-

XML tags must be closed in an appropriate order.

eg:-

```
<outer-element>
<internal_element>
   This tag is closed before the outer_ele
</internal_element>
</outer_element>.
```

③ XML attributes:-

XML attributes are part of XML elements. An element can have multiple unique attributes. Attribute gives more information about XML elements. They define properties of elements. An XML attribute is always a name-value pair.

Syntax:-

```
<element-name attribute 1 attribute 2>
---- content
</element-name>
```

where, element-name is name of the element. The name its case in the start and end tags must match. An empty element has following syntax.

```
<name attribute 1 attribute 2---/>
```

attribute 1, attribute 2 are attributes of the element separated by white spaces. An attribute defines a property of the element. An attribute is written as

```
name="Value"
```

Eg:-

```
<?xml Version ="1.0"?>
<contact info>
<address category="residence">
<name> Alisha </name>
<phone> 01-678912 </phone>
</address>
</contact info>
```

Attribute types:-

① CData | string type:- CData is a string type. CDATA is character data. It takes any literal string as a value. It can contain any character except those reserved by XML.

Example:- <actor role="joker">

where role is the attribute of the element joker. This takes a string value.

ii, **Tokenized Types:—**

These types are text strings that follow certain rules for the format and content. The syntax is

attribute token.

Types:—

① ID — A unique text string

② IDREF - A reference to an ID value.

③ IDREFs - A list of ID values separated by white space

④ Entity - A reference to an external unparsed entity.

⑤ entities - A list of entities separated by white space.

⑦ NMTOKEN - An accepted XML name

⑧ NMTOKENS - A list of XML names separated by white spaces.

iii, **Enumerated type:—**

These types are attributes that are limited to a set of possible values.

General form:-

attribute (value1|value2|value3|---)

Types:—

① **Notation:—** It will be declared somewhere else in XML document.

② Enumeration:— It allows you to define a specific list of values that the attribute value must match.

Features of XML attributes:—

1) These attributes are unique.
2) one element can have many numbers of attributes
3) one attribute can be assigned to different elements simultaneously.
4) The scope of the attribute is limited to the element to which it is assigned.
5) one attribute can not hold multiple values at the same time.
6) An attribute name should not appear twice or more in the same/empty element tag.
7) proper use of "<>" should be done
8) Attributes should be declared already before using them under DTD.//

④ XML Values:—

An XML value represents well-formed XML in the form of XML document, XML content, or sequence.

An XML value that is stored in a table as a value of common defined with XML data type must be a well formed XML document. XML values are processed in an internal representation that is not comparable to any string value including another XML value. An XML value can be transformed into a serialized string value represeriting an XML document using the XML SERIALIZE Function. An XML value can be implicitly parsed or serialized when exchanged with application string and binary data types. The XML data type has no defined maximum length.

Restrictions when using XML values:- You can use XML values in the same contexts in which you can use other data types XML values are valid in.

- CACT a parameter marker, XML, or NULL to XML
- IS NULL predicate
- COUNT and COUNT_BIG aggregate functions
- XML scaler functions
- A SELECT list without DISTINCT
- SET and VALUES INTO
- procedure parameters
- Trigger correlation Variables.
- INSERT VALUES clause, UPDATE SET clause, and MERGE.

XML values cannot be used directly in the following places. where expressions are allowed, an XML value can be used.

- A SELECT list containing the DISTINCT keyword
- A GROUP BY clause
- An ORDER BY clause
- An aggregate function with the DISTINCT keyword
- A primary, unique, or foreign key
- A check constraint
- An index column.

(5) Document Type Defination:-

DTD stands for Document Type Defination. It is used To validate the XML documents. XML provides facility to create your own DTDs for XML document. DTD specifies the structure of the XML document. DTD is a part of the file (or) separate document file. A DTD is associated with an XML document via a document type declaration, which is a tag that appears near the start of the XML document. An XML document with correct syntax is called "well formed" and An XML document validated against a DTD is both "well formed" and "valid"!

DTD Syntax:-

```
<!DOCTYPE element DTDidentifier
[ declaration1
  declaration2
]>
```

Types of DTD:-

(1) Internal DTD:- if the DTD is declared inside the XML file, it must be wrapped inside the <!DOCTYPE> declaration.

Syntax:-

```
<!DOCTYPE root-element [element-declarations]>
```

Example:-

```
<?XML Version = "1.0" ?>
<!DOCTYPE address [
<!ELEMENT address (name, company, phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
]>

<address>
<name> Ramya </name>
<company> ABC company </company>
<phone> 99999999 </phone>
</address>
```

② External DTD:-

if the DTD is declared in an external file, the <!DOCTYPE> declaration must contain a reference to the DTD file:

Syntax:-

```
<!DOCTYPE root-element SYSTEM "file-name">
```

Example:-

```
<?XML Version = "1.0">
<!DOCTYPE address SYSTEM "address.dtd">
<address>
<name> Ramya</name>
<company> ABC Company </company>
<phone> 22222222</phone>
</address>
```

Interpretation of DTD:-

!DOCTYPE note defines the root element of this document is note.

!ELEMENT note defines that the note element must Contain four elements: "to, from, heading, body."

!ELEMENT to defines the to element to be of type "#PCDATA".

!ELEMENT from defines the from element to be of type "#PCDATA".

!ELEMENT heading defines the heading element to be of type "#PCDATA".

!ELEMENT body defines the body element to be of type "#PCDATA".//

* DTD-XML building blocks (or) elements:—

① elements
② Attributes
③ Entities
④ PCDATA
⑤ CDATA.

① elements:— elements can contain text, other elements, or be empty.

Syntax:— <!ELEMENT elementname(content-type ~~or~~ or content-model)>.

Three types of elements:

① Empty element:— An element that can not contain any content.

~~<!Element elementname EMPTY>~~
~~<!ELEMENT~~

Syntax:— <!ELEMENT elementname category>

② Any element:— An element that can contain any content.

Syntax:— <!ELEMENT test ANY>

③ Mixed element:— Elements that can contain a set of content alternatives.

Syntax:— <!ELEMENT test<#PCDATA|name>

② Attributes:— Attributes provide extra information about elements. It always come in name/value pairs.

Syntax:—

<!! ATTLIST elementname attributename valuetype
[attributetype]["default"]>

The attribute-value can be one of the following

Value	Explanation
#required	attribute is required
#Implied	" " optional
# FIXED value	" value is fixed.

(3) Entities:-

Entities are used to define shortcuts To special characters. They can be expanded when a document is parsed by XML parser. An entity has three parts: an ampersand (&), an entity name, and a semicolon (;). There are 2 types

(1) internal entity:- The content may be included in XML file.

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

(2) External entity:- The content may be stored in another file.

Syntax:- <!ENTITY entity-name SYSTEM "URI/URL">

(4) PCDATA:- It means parsed character data. The text that WILL be parsed by a parser. The Text will be examined by the parser for entities and markup.

```
<name>
 <first> Ramya </first>
 <last> Naidu </last>
</name>
```

(5) **CDATA:-** It means character data. The text that will Not be parsed by a parser.

Example:-

```
<script>
<![CDATA[
<message>welcome to Tutorials point</message>
]]>
</script>
```

(6) **XML schemas:-**

An XML schema describes the structure of an XML document. The XML schema language is also referred to as XML schema definition. An XML schema defines:

- elements that can appear in a document.
- Attributes that can appear in a document.
- which elements are child elements
- The order of child elements
- The number of child elements
- Data types for attributes and elements
- XML schemas defines the elements of XML files.

Syntax:-

```
<xs:schema xmlns:xs="______"> (or)
   <?XML Version="1.0"?>
    <xs:schema>
       --
       --
      </xs:schema>.
```

Program:-

```
<?XML Version="1.0"?>
<xsd:schema xmlns:xsd="http:www.w3.
                org/2001/xml schema">
```

```
<xsd:element name="details">
 <xsd:complex type>
 <xsd:sequence>
 <xsd:element name"firstname"type="xsd.string"/>
 <xsd:element name"lastname"type="xsd.string"/>
 <xsd:element name "age"type="xsd:integer"/>
 <xsd:element name"dob" type="xsd:date"/>
 </xsd:sequence>
 </xsd:complex type>
  </xsd:element>
   </xsd:schema>
```

XML Schema types:-

① Complex type:- A container for other element definations. It uses the namespace <xsd:complexType>

```
<xs:element name="perconal info">
<xs:complex type>
 <xs:sequence>
 <xs:elementname="name"type="xs:string"/>
 <xs:element name="company"type="xs:string"/>
 <xs:element name="phone" type="xs:int"/>
 </xs:sequence>
  </xs:complex type>
  </xs:element>;
```

② Simple type:- This is used only in the context of the text.

```
<xsd:element name ="age">
<xsd:simple type>
<xsd:restriction base="xs:integer">
<xsd:minInclusive value ="0"/>
<xsd:maxInclusive value=" 100"/>
</xsd:restriction >
</xsd:simple type>
</xsd:elements>.
```

XML schema data types:-

A data type specifies the type of content that an element can hold. These data types can be classified as follows:

1. primitives:- The fundamental data types of XSD.
2. Derived:- Defined by using other data types called basic types.
3. Atomic:- Are primitive(or) derived datatypes that cannot be broken down into smallerunits.
4. List:- Are derived data types that contains a set of values of atomic data types.
5. Union:- Are derived from the atomic and list data types.

Advantages of XML schema:-

1) XSD enables you to create your own datatypes.
2) XSD enables you to specify restrictions on data.

3) XML schema is extensible

4) The syntax for defining an XSD is the same as the syntax used for XML documents.//

DTD vs XSD:—

DTD	XSD
It doesn't support data types.	It supports datatypes for elements and attributes.
It doesn't support name space.	It supports namespace
It doesn't define order for child elements.	It defines order for child elements.
It is not extensible	It is extensible
It is not simple to learn	It is simple to learn because you don't need to learn new language.
It is derived from SGML syntax.	XSDs are written in XML.
It provides less control on XML structure.	It provides more control on XML structure.

7) Document object Model (DOM):—

The XML DOM defines a Standward way ~~of~~ for accessing and manipulating XML documents.

It provides a structural representation of the document enabling you do modify its content and visual representation by using a scripting language. if you write an application which uses a DOM-complaint XML parser then your application will function in a certain way.

According to DOM:—

1) The entire document is a document node
2) Every XML element is an element node
3) The text in the XML are text node
4) Every attribute is an attribute node.
5) Comments are comment nodes.

The XML DOM views an XML documents as an node tree. All the nodes in the tree have a relationship to each other. All nodes can be accesse through the tree. Their contents can be modified (or) deleted and new elements can be created. The DOM is separated into 3 different levels:

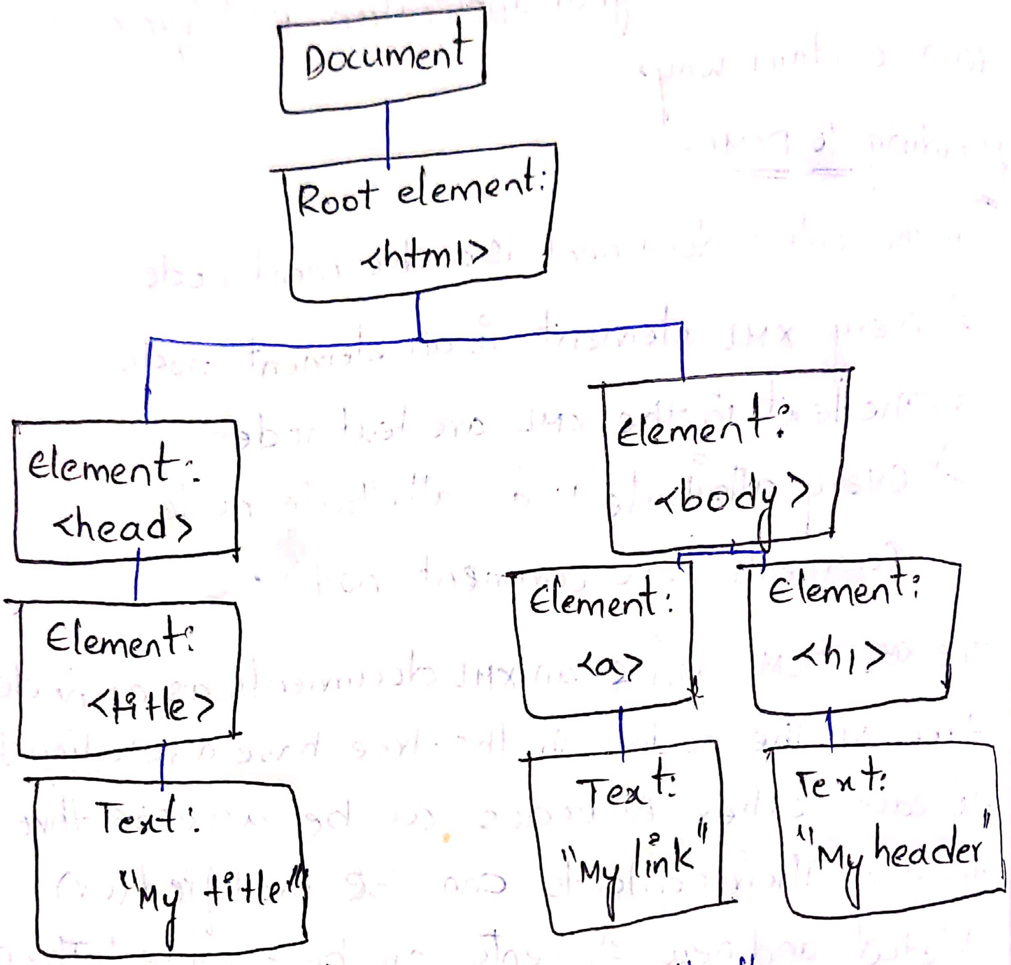
① Core DOM — standard model for any structured document
② XML DOM — standard model for XML document
③ HTML DOM — standard model for HTML document.

Sample:—

```
<html>
<head>
<title>DOM EXAMPLE </title>
</head>
<body>
 <h1>DOM CLass lesson </h1>
```

```
<p> Hello world </p>
</body>
</html>
```

Representation of DOM:- (for above code) in tree format

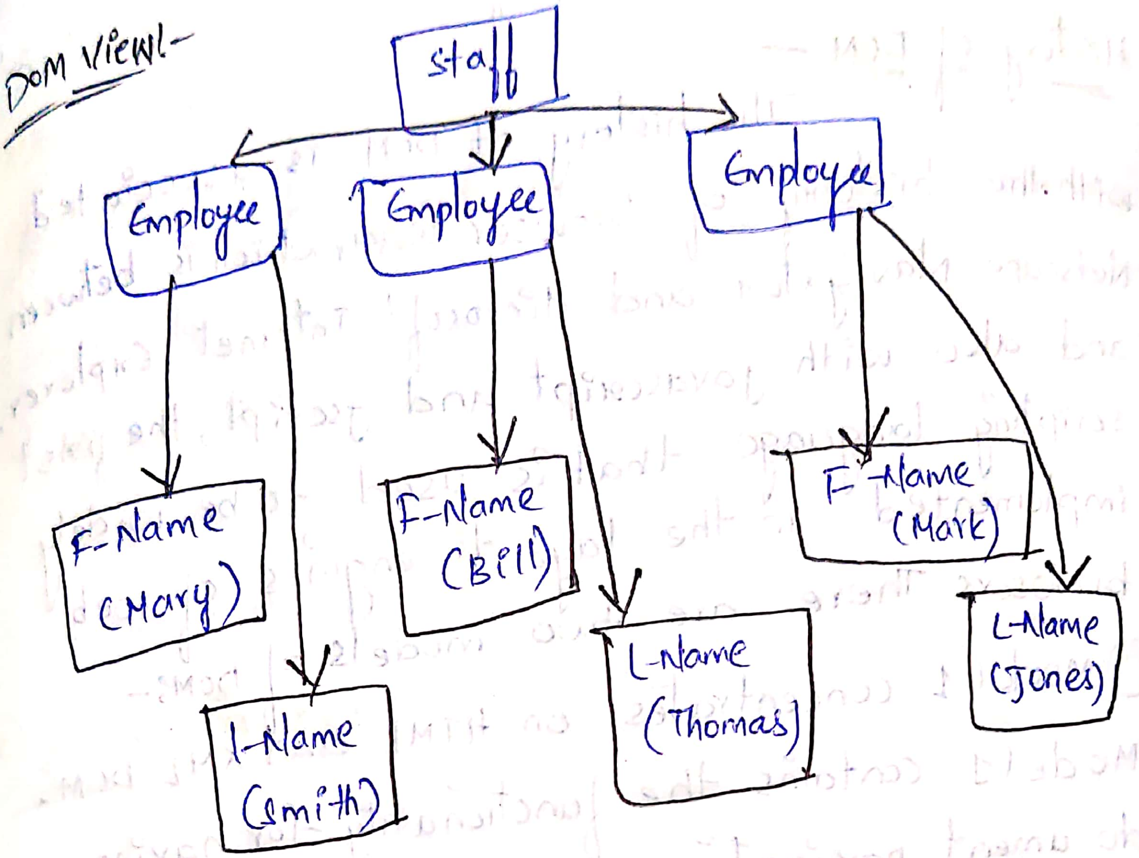


Example:- (XML View)

```
<?xml version = "1.0"?>
<staff>
<employee>
<F-Name>Mary </F-Name>
<L-Name>Smith</L-Name>
</Employee>
<employee>
<F-Name>Bill</F-Name>
<L-Name>Thomas</L-Names>
</Employee>
<Employee>
<F-Name>Mark</F-Names>
<L-Name>Jones</L-Name>
```

</Employee>
</staff>

DOM View:-



Advantages:-

1) XML DOM is traversable
2) Robust API for the DOM tree
3) Relatively simple to modify the data structure and extract data.
4) XML DOM provides us ~~information~~ platform and independence in languages.
5) XML dom is editable and dynamic.

Disadvantages:-

1) stores the entire document in memory.
2) operational speed is slower due to larger use of memory.

3) It consumes more memory when the XML structure becomes large.

History of DOM:— The history of DOM is associated with the history of browser waar, which is between Netscape Navigaton and Microsoft Internet Explorer, and also with javascript and jscript, the first scripting language that is used to be widely implemented in the layout engines of web browsers. There are two models of DOM:—

① Model 1 concentrates on HTML and XML DOM. Model 1 contains the functionality for having document navigation and management.

level 1 became a 1st recommendation of W3C in october 1998.

② Model 2 became a recommendation of W3C on 13 November 2000. In this, we can add the styl object to the DOM to manage the style informati in the document.

level 2 also defines an event modeland provide the support of XML namespaces.

Types:—

There are three types of API's which are specified by DOM standard:

① properties:—

DOM has a property of style sheet, that can be used to get the list of css files, it has a little property:-

2) Events:-

we call event method e.g "onreadystatechnge" this code is for the event handler.

3) Methods:-

use methods e.g: files, delete, files.info, files.list. These are the methods on the file,

~~8) XHTML in XML:-~~

8) XHTML:-

XHTML stands for Extensible Hypertext Markup language. It is a cross between HTML and XML Language. XHTML is HTML defined as XML application. It is supported by all major browsers. XHTML documents are well-formed and parsed using standard XML parsers, unlike html which requires a lenient HTML specific parser.

Features:-

1) unlike html, which is standard generalized markup language based, XHTML is XML-based.
2) XHTML documents has only one root element.
3) Sustainability is more pronounced in case of XHTML than HTML.
4) XHTML elements need to be nester properly

and" should always be in lower case.

5) As XHTML supports ~~can~~ wide range of applications, using the same complex websites can be created.

6) As the error processing routines are shorter, future browsers can support faster processing of XHTML documents.

7) As they are XML conforming, XHTML documents are easily viewed edited using standard XML tools.

8) It is a restrictive subset of XML and needs to be parsed with help of standard XML parsers.

XHTML VS HTML:—

HTML	XHTML
1) HTML stands for hypertext Markup language.	1) XHTML stands for Extensible hypertext markup language.
2) It is an SGML application	2) It is an XML application
3) Tim-berners-lee proposed it in 1987.	3) The world wide web Consortium recommended it in 2000.
4) HTML is not case Sensitive.	4) XHTML is case Sensitive

5) HTML is less expressive	5) XHTML is more expressive
6) HTML is not mandatory for a single root element	6) XHTML documents must contain atleast one root element
7) Attribute values are not significant in HTML	7) Attribute values are important in XHTML.
8) HTML uses a format such as 	8) ~~XHTML uses~~ All unclosed tags must be closed in XHTML.
9) All content can be included in the body ~~language~~. element.	9) All contents must be put in blocks.

Components of xhtml:-

① Document type defination (DTD):-

DTD was optional in HTML, but it is now required in XHTML. So each XHTML document must begin with a DTD declaration and a line of code indicating that you about to start writing XHTML code and it specifies the language/script used to encode the text. These are 3 varieties; 1) transitional DTD

2) strict DTD

3) frameset DTD

2) **Head:-**

It includes information about the document, its title, and other attributes that are not considered document content.

3) **Body:-**

This body tag provides the content of web pages with multiple tags//

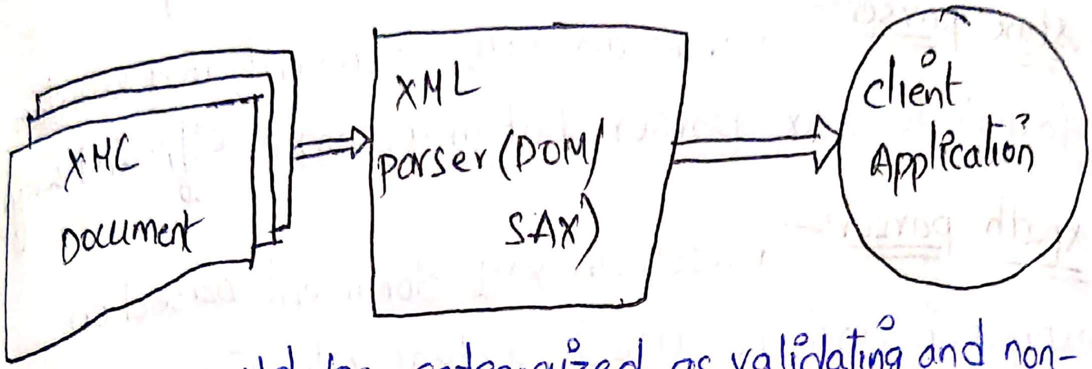
(9) parsing XML Data:-

XML parsers arealso known as XML processor is defined as splitting the information into some component parts means which read the XML file and stores all avaible functions in memory to the rest of the program code. The XML parser is designed to read the XML document and create a way for programs to use the XML.

Working of XML parser:-

The mainjob of XML parser is to access/ modify the data in the document. The parser contains installed software packages for the client applications to interface with and also does the validation process of the XML documents. The component parts are compared with the DTD/schema pattern for checking. It triggers the event when it finds opening and closing tags in a file while parsing line-by-line.

parser covers two sections: a lexer and a parser. A lexer takes input characters from the file and produces tokens like (<,>, tagnames). The parser takes this token and constructs a tree-based syntax with respective to grammar. with DOM, the whole document is read, while in case of SAX parser reads node by node and throws parsing events.



The parser could be categorized as validating and non-validating.

Validating Parser:— It needs a document type Declaration to parse and gives an error if the respective document doesn't match with DTD and constraints.

Non-validating:— This parser eliminates DTD and the parser checks for the well-formed document.

Types of XML parsers:—

① DOM parser:— It is a W3C standard and converts the XML document which is to parsed into a collection of objects and uses DOM API. DOM is much easier to use as sorting and searching process is made faster.

② SAX parser:— SAX is simple API for XML and meant has push parser also considered to be stream oriented XML parser. It is used in case of high-performance applications like where the XML files is too larged and comes with the community based standard and requires less memory.

③ JDOM parser:- parses an XML document in a similar fashion to DOM parser but in an easier way.

④ Stax parser:- parses an XML document in a similar fashion to SAX parser but in a more efficient way.

⑤ xpath parser:- parses an XML document based on expression and is used extensively in conjunction with XLST.

⑩ DOM and SAX parsers in Java:-

DOM Parser:- 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. DOM exposes the whole document to applications. The tree structure makes easy to describe an XML document. A tree structure contains root element, child element. The XML DOM makes a tree structure view for an XML document.

Advantages:-

1) It supports both read and write operations

2) It is good when random access to widely separated parts of a document is required.

Disadvantages:-

1) It is memory inefficient

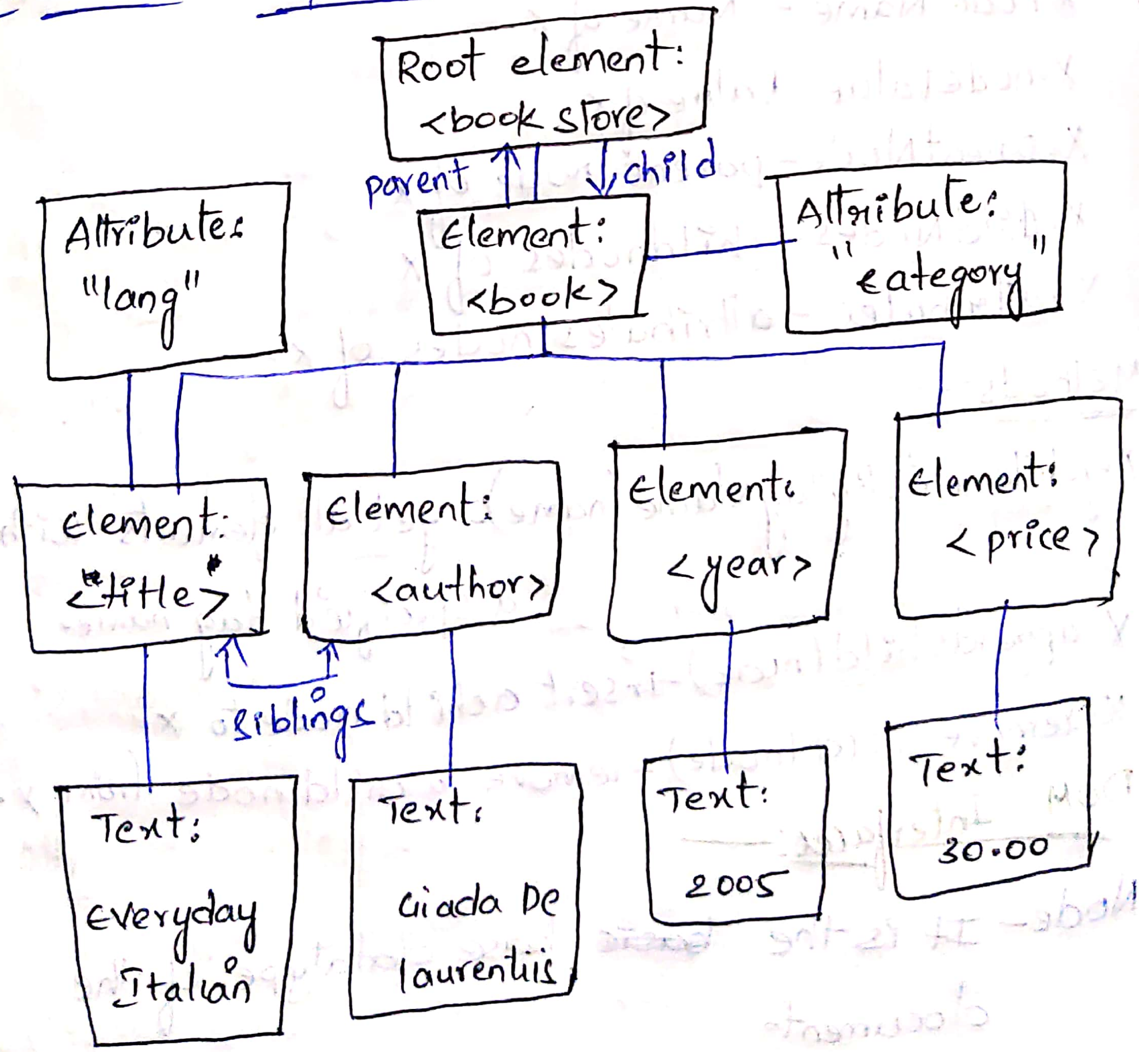
2) It seems complicated, although not really.

The XML DOM defines the objects and properties of all XML elements, and the methods to access them. The XML DOM is a standard for how to get, change, add (or) delete XML elements.

Example:-

```
<bookstore>
<book category ="cooking">
<title lang="en">Everday Italian </title>
<author>Giada De Laurentiis </author>
<Year>2005</year>
<price>30.00</price>
</book>
</bookstore>
```

Tree structure representation:-



DOM Nodes:-

everything in XML document is node.
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//

The parser reads XML into memory and converts XML into XML DOM object which is accessible from javascript. The MS parser supports loading of both XML files and XML strings. other browsers use separate browsers.

DOM properties and Methods:-

properties:-

- X.node Name - Name of X
- X.nodeValue-Value of X
- X.parentNode-parent node of X
- X.child Nodes - childnodes of X
- X.attributes - attributes nodes of X.

Methods:-

- X.getElementsByTagName(name)- get all elements with a specified tag name.
- X.appendChild(node)-insert a child node to X
- X.remove Child (node) -remove a child node from X.

DOM Interfaces:-

Node- It is the ~~basic~~ base datatype of the document.

Element- The objects in the document are elements.

Text- The content of an element.

Attr- represents the attribute of an element

Document- It refers to the entire XML document.

SAX Parser:-

A SAX parser implements SAX API. This API is an event based API and less intiative. It doesnot first create any internal structure. client does not specify what methods to call. client just overrides the methods of the API and place his own code inside there. when the parser encounters start-tag, end-tag etc., it thinks of them as events. SAX parser is event-based. client application seems to bejust receiving the data inactively, from the data flow point of view.

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 intiative.
2) clients never know the full information because the data is broken into pieces.

Attributes Interface:-

1) int getLength()-Returns number of attributes.
2) string getQName (int index)

3) string getvalue(int index)

4) string getvalue(string qname)

Content Handler Interface:—

① void startDocument()—called at the beginning of a document.

② Void endDocument()— called the end of a document.

③ Void startElement(string uri, string localName, string qname, Attribute atts)—called at the beginning of an element.

④ Void endElement(string uri, string localName, string qname, Attribute atts)—called at the end of an element.

⑤ Void characters (char[]ch, int start, int length)— called when character data is encountered.

⑥ void ignorableWhitespace (char[]ch, int start, int length)— called when a DTD is present and ignorable whitespace is encountered.

⑦ Void processingInstruction (string target, string data)— called when a processing instruction is recognized.

⑧ Void skippedEntity(string name)—called when an unresolved entity is encountered.

⑨ Void startPrefixMapping (string prefix, string uri)— called when anew namespace mapping is defined.

⑩ Void endprefixMapping (string prefix) — called when namespace definition end its scope//