UNIT-II

XML (extensible Markup Language).

XML stands for extensible Markup language. It is a text-based mark-up language derived from Standard Generalised markup language (SGML). XML tags identify the data and are used to store and organise the data, rather than specifying how to display it like HTML tags, which are used to display the data.

XML can work behind the scene to simplify the creation of HTML documents for large websites. It can be used to exchange the information between organisations and systems. XML can be used to store and arrange the data and it is also used for offloading and reloading the databases. XML can easily be merged with style sheets to create almost any desired output.

Advantages of XML :-

1.) XML document is human readable and we can edit any XML document in simple text editors.

2.) The XML document is language neutral. That means a Java program can generate an XML document.

3.) Every XML document has a tree structure. Hence, Complex data can be arranged systematically & can be understood in simple manner.

4.) XML files are independent of an Operating System.

features of XML :-

→ XML is intended for transport (or) Storage the data.

→ The most important feature of XML is that use is able to define & use his own tag.

→ XML Contains only data & does not Contain any formatting information.

→ XML permits the document writer to create the tags for any type of information.

→ Seaeching, Sorting, Rendering (or) manipulating the X document is possible using extended Stylesheet language (XSL).

→ Some commonly used web browsers like internet Explorer & mozilla firefox provide Support to the tags in XML.

Key Points :-

1.) XML is case sensitive.

Eg:- <message> Hi </message> (WRONG).

<message> Hi </message> (CORRECT).

2.) In XML Start tag must have matching end tag.

Eg:- <Starting> Hi </end> (WRONG)

<Starting> Hi </Starting> (CORRECT).

3.) The elements in XML must be properly nested.

Eg:- <one> <two> Hi </one> </two> (WRONG)

<one> <two> Hi </two> </one> (CORRECT).

4.) XML must be saved with ".xml".

Comments in XML:-

The Syntax of writing comments in xml is similar to HTML Comments.

Eg:- <!--- This line is a Comment line --->.

Simple xml pgm:-

```
<?xml version = "1.0"?>
<employees> ⟶ Root element
    <employee> ⟶ Container element
        <name> Sai Eswari </name>
        <dept> IT </dept>
    </employee>.
</employee>
```

Defining XML TAGS:-

In xml the basic entity is element. The elements are used for defining the tags. mostly only one element is used to define a single tag.

We can broadly categorite xml tags as follows.

i) Start tag:- The beginning of every non-empty xml element is marked by a start-tag.

Ex:- < element name >

ii) End tag:- Every element that has a start tag should tag end with an end tag.

Ex:- </ element name >

iii) Empty tag:- An element which has no content is termed as empty. An empty tag can be

defined by putting a '/' (forward slash) before closing bracket.

Ex:- < student /> .

* Space (or) tab character is not allowed in the element name (or) in attribute name.

In the above Simple XML Pgm "employees" is a root element. The root element is a element which is Parent of all other elements in XML document.

The other elements are called Container elements. There can be one (or) more elements inside Container elements. Such elements are called child elements. In the above Example "employee" is a container element which contains the child elements "name" and "dept".

Attributes & values :-

Attributes are designed to contain data related to a specific element. Attributes must be enclosed within double quotes (or) Single quote.

Ex:- < Person gender = "female" > .

XML Elements vs Attributes

Ex1:- < Person gender = "female" >
< first name > Dutta < / first name >
< last name > Sai Eswari < / last name >
< / person >

Ex2:- < person >
< gender > female < / gender >
< first name > Dutta < / first name > .

</last name> Sai Eswari </last name>.

</person>.

In the "Ex1" gender in an attribute. In the "Ex2" gender is an element. Both examples provide same info.

Name Spaces in XML

Sometimes we need to create two different elements by the same name. The XML document allows us to create different elements which are having the common name. This technique is known as namespace.

Ex:

```
<file_Description>
  <text fname = "input.txt">
    <describe> It is text file </describe>.
  </text>.
  <text fname = "flower.jpg">
    <describe> It is an image file </describe>
  </text>.
</file_Description>.
```

The above document doesnot produce any error although the element text is used for two different attribute values.

DOCUMENT TYPE DEFINATION (DTD):

The document type defination is used to define the basic building blocks of any XML document. Using DTD we can specify various elements types, attributes and their relationship. Basically DTD is used to specify the set of rules for structuring

dates in any XML file.

If we want to put some information about Students in XML file then, generally we use tag Student followed by his/her name, address, Standard and marks. That means we are actually specifying the manner by which information should be arranged in XML file.

Building Blocks of XML

1.) Elements :- Elements are the main building blocks of both XML & HTML. These are used for defining the tags. They typically consist of opening & closing tags. Mostly only one element is used to define a single tag.

Ex:- <body> Some text </body>

<message> Some text </message>

2.) Attributes :- Attributes are generally used to specify the values of the element. They provide extra information about elements. These are specified with in double quotes.

Ex:- <flag type = "True">

The name of the element is "flag", name of the attribute is "type", value of the attribute is "True".

3) Entities :- Some characters have special meanin in XML, like less than sign (<) that defines starting of XML tag.

following entities are predefined in XML.

Entity Reference	Character.
<	<
>	>
&	&
"	"
&apos	'

4.) CDATA :- CDATA Stands for 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.

5.) PCDATA :- PCDATA means Parsed character data. PCDATA is text that will be parsed by a parser. The text will be examined by Parser for entities and markup. Tags inside the text will be treated as markup and entities will be expanded. Parsed character data should not contain any &, <, or > characters, these need to be represented by & < and > entities.

→ XML document Contains data where as DTD files Contains Rules that apply to data.

→ DTD most Saved with '.dtd' extension.

→ An XML file Contains an Reference to DTD file

→ A DOCTYPE declaration in an XML document Specifies that we want to include a Reference to DTD file.

→ DTD defines doc Structure with a list of legal elements & attributes.

Syntax

```
<!DOCTYPE DOCUMENT [
<!ELEMENT ELEMENTNAME 1 (attribute name)
          appearance of attributes>.
<!ELEMENT ELEMENTNAME 2>
    :          :          :
<!ELEMENT ELEMENT NAME n>
<!ATLIST element_name attribute_name
          attribute_type default_value>
]>.
```

Types of DTD :- There are 2 types of DTD's.

1.) Internal DTD :- In this we enclose DTD in a <DOCTYPE> element providing the name of root element of document.

Syntax :

```
<!DOCTYPE Root_element [element_declaration]>
```

Root_element is the name of root element and element declaration is where you declare the elements.

Ex:

```
<? xml version = "1.0" ?>
<!DOCTYPE employee [
        <!ELEMENT name (# PCDATA)>
        <!ELEMENT dept (# PCDATA)>.
]>.
<employee>
        <name> Sai Eswari </name>
        <dept> IT </dept>
</employee>.
```

2.) External DTD:- In external DTD elements are declared outside the xml file. They are accessed by specifying the attributes which may be either the legal '.dtd' file (or) valid URL.

If the DTD is declared in an external file, the <!DOCTYPE> defination must contain a reference to DTD file.

Syntax:-

<!DOCTYPE Root-element SYSTEM "filename.dtd">

Ex:- <?xml version = "1.0" ?>
<!DOCTYPE employee SYSTEM "emp.dtd">
<employee>
<name> Sai Eswari </name>
<dept> IT </dept>
</employee>

} Save with .xml.

<!ELEMENT name (#PCDATA)>
<!ELEMENT dept (#PCDATA)>

} Save with emp.dtd.

Uses of DTD

1.) with DTD each of your xml files can carry the description of its own format.
2.) with DTD independent groups of people can agree to use a standard DTD for interchanging data.
3.) your application can use standard DTD to verify that the data you received from outside world is valid.
4.) you can also use DTD to verify your own data.
5.) DTD's are relatively simple and compact.

Limitations Of DTD

→ They are not written in XML Syntax, which means you have to learn a new syntax inorder to use

→ DTD's are not Object Oriented.

→ DTD's donot support namespaces very well.

→ we can't use multiple DTD to validate one XML Doc.

→ large DTD's are hard to read and maintain.

XML SCHEMAS

→ The XML Schemas are used to represent the structure of XML document. The goal (or) purpose of XML Schema is to define the building blocks of an XML document. These can be used as an alternative to XML DTD. The XML schema language is called as XML Schema Definition (XSD) language. The XML Schema became the world wide web Consortium (W3C) Recommendation in 2001.

→ XML Schema defines elements, attributes, elements having child elements, order of child elements. It also defines fixed and default value of elements & attributes.

→ XML Schema also allows the developer to use data types.

Ex:- Student schema.xsd.

```
<?xml version = "1.0"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/ XMl Schema">
<xs:element name = "student">
<xs:complexType>
<xs:Sequence>
<xs:element name = "name" type = "xs:string"/>
```

```
<xs:element name = "address" type = "xs:string" />.
<xs:element name = "std" type = "xs:string" />.
<xs:element name = "marks" type = "xs:string" />.
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schemas
```

Simple Type :- Simple type element is used only in the context of text. Some of predefined Simple types are:

xs: integer , xs: boolean, xs: string , xs: date

Complex Type :- A complex type is a container for other element definations. This allows you to specify which child elements an element can contain & to provide some structure within your XML documents.

```
<xs:element name = "Address">
<xs:complex Type>
<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: Complex type>
</xs: element>
```

Global Type :- With global type, you can define a single type in your document, which can be used by all other references. for example, suppose you want to generalize the person & Company for different addresses of Company.

```
<xs: element name = "Address Type">
<xs: Complex Type>
```

```
<xs: Sequence>
<xs: element name = "name" type = "xs: String" />
<xs: element name = "Company" type = "xs: string" />
</xs: Sequence>
</xs: Complex Type>
</xs: element>.

<xs: element name = "Address 1">.
<xs: Complex Type>.
<xs: Sequence>
<xs: element name = "address" type = "Address Type" />
<xs: element name = "phone1" type = "xs: int" />.
</xs: Sequence>.
</xs: Complex Type>
</xs: element>
<xs: element name = "Address 2">.
<xs: Complex Type>.
<xs: Sequence>
<xs: element name = "Address" type = "Address Type" />.
<xs: element name = "phone2" type = "xs: int" />
</xs: Sequence>.
</xs: Complex Type>.
</xs: Sequence>.
</xs: Complex Type>.
</xs: element>
```

Advantages :-

1.) The schemas are more specific.
2.) The schema provide the support for data types.
3.) The schema is aware of name spaces.

4.) XML schema allows the creation of complex and reusable content models easily.

Advantages of Schema over DTD:-

1.) Both schemas & DTD's are useful for defining structural components of XML. But the DTD's are basic & cannot be much specific for complex operations.

2.) The schemas provide support for defining type of data. The DTD's do not have this ability.

3.) The schemas are namespace aware & DTD's are not.

4.) The XML schema is written in XML itself & has a large number of built in & desired types.

5.) Large no. of web applications can be built using XML schema. On the other hand relatively simple & compact operations can be built using DTD.

DOCUMENT OBJECT MODEL (DOM)

DOM is a language-neutral & platform-independent object model used to represent XML documents. DOM helps scripts & programs to access, add, delete & edit content, structure and style of XML documents dynamically.

DOM is standardized by world wide web consortium (W_3C). The primary objective of this standard was to model HTML document in an object-oriented way so that it could be exposed to scripts, and scripts could access & manipulate HTML documents through this object model dynamically.

DOM defines the objects and properties and methods (interface) to access all XML elements.

DOM specification is divided into 3 major parts:-

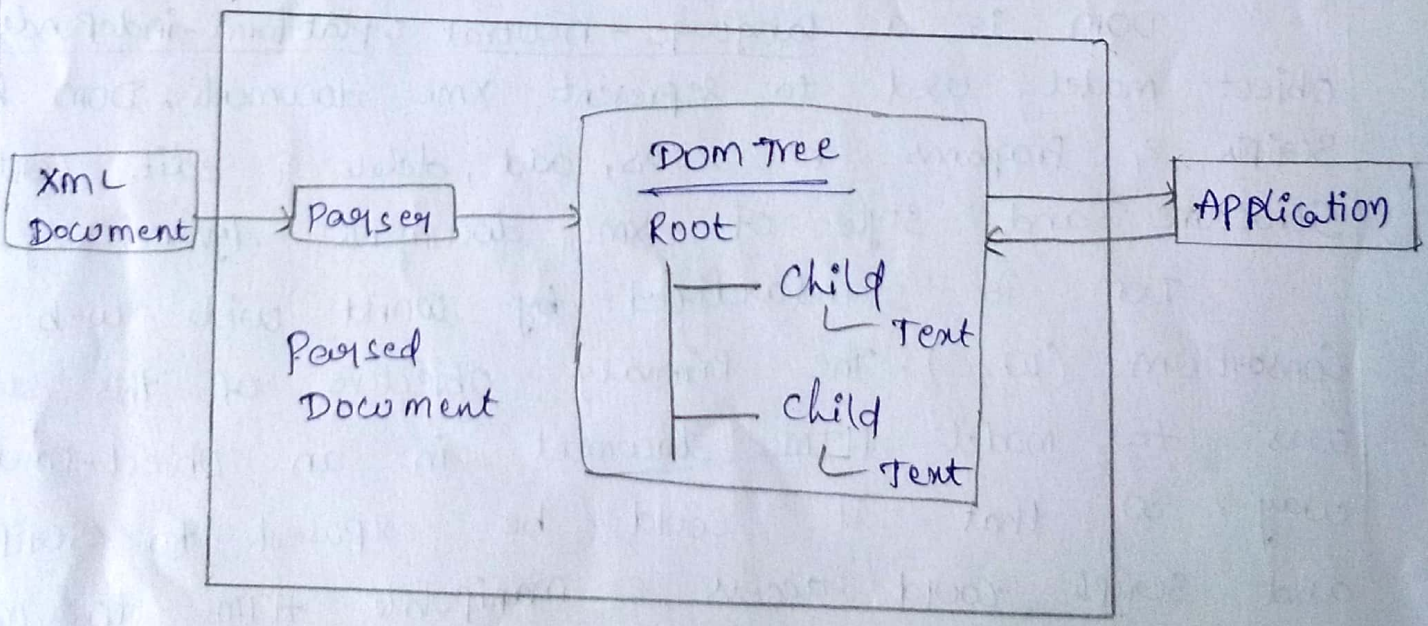
1.) DOM core :- This Portion defines the basic set of interfaces and Objects for any structured docum

2.) XML DOM :- This Part specifies the standard set of objects and interfaces for XML document only.

3.) HTML DOM :- This Part defines the interfaces and objects for HTML documents only.

XML DOM is a standard object model for XML. XML documents have hierarchy of informational units. called nodes. DOM is a standard Programming interface of describing those nodes and the Relationship between them.

As XML & DOM also Provides an API that allows a developer to add, edit, move (or) remove nodes at any point on the tree in order to create an application.



Above is the diagram for the DOM structure. which depicts that Parser evaluates an XML document as DOM structure by traversing through each node.

A DOM document is a collection of nodes (or) pieces of information, organised in hierarchy. Some types of nodes may have child nodes of various types and others are leaf nodes that cannot have anything below them in the document. Below is a list of the nodes types, and which node types they may have as children.

1.) Document :- Only one Document type node exists for each XML document. It is essentially a container of all components of an XML document such as XML declaration, elements, attributes, comments, entities and so on.

2.) Element :- This interface represents an elements in XML document. Elements may have attributes. It has several useful methods to get an Attr node (or) value of attribute by its name.

3.) Attr :- An Attr type node represents an attribute of an Element node ------ text, Entity Reference.

4.) Text :- It represents the textual content of Element type node. The text content of an element node is represented as separate Text type node.

5.) Document fragment :- The implementation of Document object can be heavyweight as a large no. of methods and properties have been defined for it---- element, Processing Instruction, Comment, Text, CDATA Section, Entity Reference.

6.) Document Type :- It provides interfaces to get information about document, including list of entities defined for this document.

7.) Entity Reference :- A node of this type represents entity reference in the document.--- Element, Proce Instruction, Comment, Text, CDATA Section, Entity Refere

8.) Processing Instruction :- This interface represents a "Processing instruction", which is used in XML to Provide specific information about document - Processor.

9.) Comment :- It represents a comment in an XML document. Note that all the characters between the starting '<!--' and ending '-->' delimiters are not Processed and are called comment.

10.) CDATA section :- This type of node represents CDATA section in XML doc. Note that no lexical check is done on content of CDATA section.

11.) Entity :- It represents a known, either unparsed or Passed entity.--- Element, Processing Instruction, Comment, Text, CDATA Section, Entity Reference.

Eg:- Consider DOM representation of following XML doc node.xml.

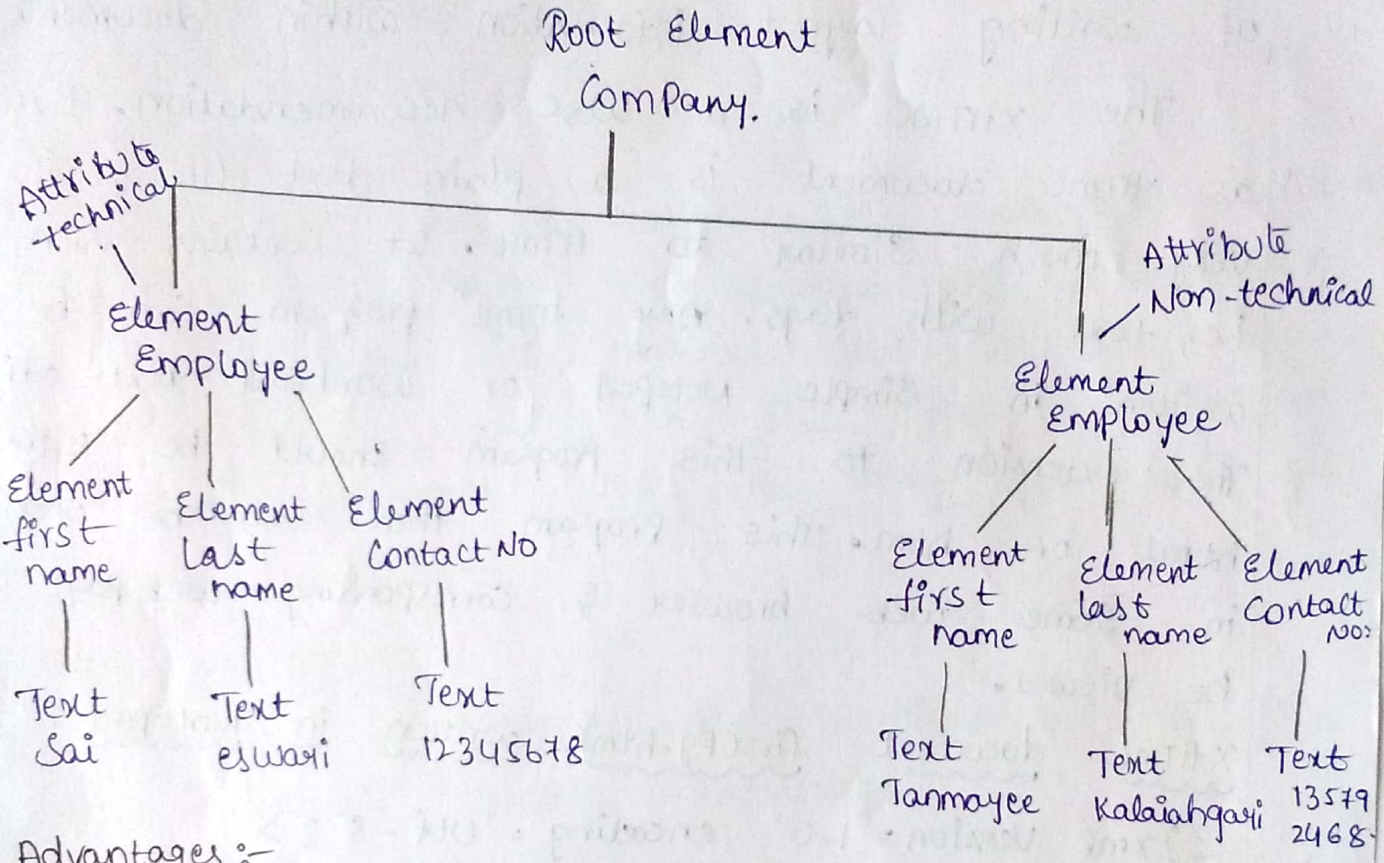
```
<?xml version ="1.0"?>
<Company>
    <Employee category = "technical">
    <firstname> Sai </firstname>
    <Last Name> Eswari </Last name>
    <Contact NO:> 12345678 </Contact NO:>
    </Employee>

    <Employee Category = "non-technical">
    <first Name> Tanmayee </first Name>
    <Last Name> Kalaiahgari </Last Name>
    <Contact NO> 135792468 </Contact NO>
```

</Employee>

</Company>

The Document object model of above XML document would be as follows:



Advantages:-

→ XML DOM is language and Platform independent.

→ XML Dom is traversible - Information in XML Dom is organised in hierarchy which allows developer to navigate around hierarchy looking for specific information.

→ XML DOM is modifiable - It is dynamic in nature Providing developer a scope to add, edit, move (or) remove nodes at any Point on tree.

Disadvantages:-

→ It Consumes more memory.

→ Due to larger usage of memory its operational Speed, Compared to SAX is slower.

XHTML

XHTML stands for extensible Hypertext markup language. Hypertext is simply a piece of text works as a link. Markup language is language of writing layout information within documents.

The XHTML is a W3C Recommendation. Basically as XHTML document is a plain text file & it is very much similar to HTML. It contains rich text ie, text with tags. Any HTML program can be written in simple Notepad or word pad text editors. The extension to this program should be either html or htm. This program then can be opened in some web browser & corresponding web page can be viewed.

XHTML document firstPg.html written in Notepad :-

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional
    //EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-dtd">
<html xmlns:"https://www.w3.org/1999/xhtml">
    <head>
        <title> my page </title>
    <body>
        <h2> This is my first web page </h2>
    </body>
</html>
```

Every XHTML document must begin with an xml declaration element that simply identifies document because it is based on XML. Thus

In this line the xml version is specified as 1.0 & encoding as utf i.e., unicode encoding.

Note that XHTML document consists of some Strings enclosed within angular brackets. Such Strings are called tags. Hence various tags that are written in above doc are <html>, <h2>, <head> & <body>.

Generally code for any scripting language is written in some text editor like Notepad. while saving XHTML document written in Notepad, save it using file extension '.html' and for file type select option "All files" otherwise your file may get saved as "first pg. html. txt".

HTML Vs XHTML

XHTML is syntactically identical to HTML. But XHTML follows certain restrictions.

HTML:-

1.) The HTML tags are case insensitive. Hence <body> or <BODY) or <Body> are treated as one & the same.

2.) we can omit the closing tags sometimes in HTML document.

3.) In HTML the attribute values is not always necessary to quote. In fact numeric attribute values are rarely quoted in HTML. Only when special characters (or) white spaces are present in attribute values then only it is essential to put quotes around them in HTML.

4.) In HTML there are some implicit attribute values.

5.) In HTML even if we do not follow the nesting rules strictly it does not cause much difference.

XHTML:-

1.) The XHTML is case sensitive and all the tags in XHTML document must be written in lowercase.

2.) for every tag there must be closing tag. Some browsers get confused if the closing tag is not given. There are two ways by which we can mention the closing tags.

```
<a href = "firstPage.html"></a>
(Or)
<a href = "firstPage.html" />.
```

3.) In every XHTML document the attribute values must be quoted.

4.) In every XHTML the attribute values must be specified explicitly.

5.) In XHTML document the nesting rules must be strictly followed. These nesting rules are:

a.) A form element cannot contain another form element.

b.) An anchor element does not contain another form element.

c.) List elements cannot be nested in the list element

d.) If there are two nested elements then the inner element must be enclosed first before closing outer element.

e.) Text elements cannot be directly nested in form elements.

Standard Structure :-

An XHTML document consist of 3 main Parts.

1.) DOCTYPE declaration.

2.) <head> Section.

3.) <body> Section.

```
<!DOCTYPE >
<html>
    <head>
        <title> ----- </title>
    </head>
    <body>
        ------
        ------
    </body>
</html>
```

The DOCTYPE declaration should be

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional //EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">.

DOCTYPE specifies the document type. The Document types is specified by DTD. The XHTML Syntax Rules are specified by file xhtml 11.dtd file.

There are 3 types of xhtml DTD's and those along with their uses are given below.

1.) XHTML 1.0 Strict :- when we want a clean markup Code then this type of dtd is used.

2.) XHTML 1.0 Transitional :- when we want to use Some html features in the existing XHTML document.

3.) XHTML 1.0 frameset :- When we want to ma... of frames in the XHTML document.

PARSING XML DATA : DOM and SAX PARSERS.

The primary goal of an XML processor is to parse the given XML document. Java has a rich source of in-built API's for parsing the given XML document. It is parsed in two ways.

- Tree based parsing.
- Event based parsing.

As we know that DOM is used to parse the XML document using the tree structure. We can access the information of XML document by interacting with the tree nodes. On the other hand Simple API for XML (i.e., SAX) allows us to access the information of XML document using sequence of events. Thus there are two methods of parsing the XML document.

- Parsing using DOM (tree based).
- Parsing using SAX (event based).

Difference between DOM and SAX.

SAX	DOM
1.) Simple API for XML	1.) Document object models.
2.) It read an xml file as node by node.	2.) It stores entire xml doc
3.) It detects an events	3.) It builts an in-memory tree representation on xml document.

4) It informs the event to our application program.

5) In SAX, application pgm acts upon the event.

6) Parsing continuous till all the events (or) end of xml is parsed.

7) It uses following package
import javax.xml.Parsers;

8) It uses less amount of memory.

4) It hands over the tree to our application program.

5) In DOM, application pgm handles the xml tree.

6) It parses the whole xml document at a time.

7) It uses following package
import javax.xml.parsers.*;

8) It uses more amount of memory.

DOM & SAX pgm example.

Dom.java

```
import java.io.*;
import java.xml.Parser.*;
import org.w3c.dom.*;
import org.xml.sax.*;
Public class DOM
{
  Static public void main (String [] arg)
  {
    try {
        System.out.Print("enter name of document");
        BufferedReader input = new BufferedReader (new InputStreamReader (System.in));
        String file_name = input.readLine();
        file fp = new file (file_name);
```

```
if (fp.exists())
{
  try {
      Document Builderfactor factory_obj = Document
                                  Builder factory.newInstance();
      Document Builder builder = factory_obj.new Document
                                        Builder();
      Inpat Source ip_src = new InputSource (file_name);
      Document doc = builder.Parser (ip_src);
      S.O.Pln (file_name +" is well-formed");
    }
   catch (Exception e)
   {
    S.O.Pln (file_name + " is not well-formed");
    System.exit(1);
   }
}
else{
     S.O.P. (" file not found");
   }
}
catch (IO Exception.ex)
{
  ex. Print stack Trace();
}
}
}
```

Student.xml

```
<?xml version="1.0"?>
<student>
    <Roll no> 10 </Roll no>
    <Personal info>
        <Name> Sai </Name>
        <Phone> 123456 </phone>
    </Personal info>
    <class> 10th </class>
    <Sub> Maths </Sub>
    <marks> 98 </marks>
    <Roll no> 20 </Roll no.>
    <personal info>
        <Name> Parvathi </name>
        <phone> 789012 </phone>
        </Personal info>
        <class> 10th </class>
        <sub> Eng </sub>
        <marks> 100 </marks>
</student>
```

Output:

```
D:\SAX_example> javac Dom.java
D:\SAX_example> java DOM
Enter name of document: student.xml
Student.xml is well-formed!
D:\SAX_example>
```