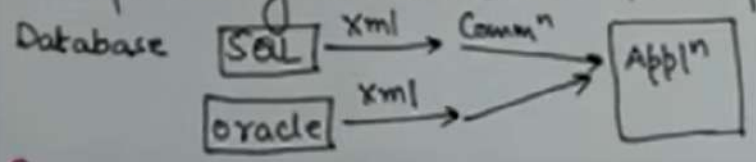
(XML)-①

Easy Engineering Classes – Free YouTube Lectures

For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

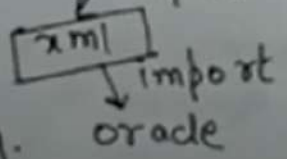
XML :- eXtensible Markup Language. → you can create your own tags.

- → used to Store and transport data.
- → Self descriptive.
- → Used to Carry data (Not used to display data)
- → Self-defined tags.
- → Platform and Language independent.
- → Helps in easy communication b/w two platforms.



Features and Advantages:-

i) Separates data from Html.
ii) Simplifies data Sharing.
iii) Simplifies data transport.
iv) Increases data availability.
v) Simplifies platform change.



XML Example: (College)

↳ Hierarchial Structure. ↳ class1 (child) College ← root, class2

```
<?xml version="1.0" encoding="ISO-8859-1"?>]
<college>
  <class1>
    <Name> Amit </Name>
    <RollNo> 1 </Roll No>
  </class1>
  <class 2>
    <Name> Mohan </Name>
    <Roll No> 2 </Roll No>
  </class2>
</college>
```

↳ declaration

← Root Element

→ child Element.

HTML	XML
i) Display Data (Look and Feel).	i) Transport and Store the data.
ii) Markup language itself	ii) Provide framework to define markup languages.
iii) Not Case sensitive	iii) Case-Sensitive
iv) Predefined Tags	iv) Can Create own tags
v) Static	v) Dynamic

HTML eg:-

```
<html>
<body>
<p> HTML INTRO
</p>
</body>
</html>
```

Predefined Tag

display

XML eg:-

```
<College>
<Class>
<Name> </Name>
</Class>
</College>
```

Custom tags

transport

XML DTD:- XML Document Type Definition/Declaration
- → used to describe XML Language precisely.
- → used to define structure of a XML document.
- → Contains list of legal elements.
- → used to perform validation.

DTD SYNTAX:-

```
<! DOCTYPE element DTD Identifier
  [ declaration 1
    declaration 2
  ]>
```

Types of DTD

Internal

elements are declared within the XML files.

Syntax:

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

External

elements are declared outside XML file.

Syntax:

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

Internal DTD Example

```
<?xml version="1.0" encoding
 ="UTF-8">
<!DOCTYPE Address [
  <!Element Address (Name,
   Company, phone)>
  <!Element Name (#PCDATA)
  <!Element Company (#PCDATA)
  ⋮
]>
<Address>
  ① <Name>___ </Name>
  ② <Company>___ </Company>
  ③ <Phone>___ </Phone>
</Address>
```

Root Element

DTD

XML

Root Element

External DTD Example

Add.dtd

XML File

```
<!DOCTYPE Address
 SYSTEM "Add.dtd">
```

(XML Namespaces)

Easy Engineering Classes – Free YouTube Lectures

For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

XML Namespaces:- used to avoid element name conflict in XML document.

→ It is a set of unique names.

→ Identified by URI (Uniform Resource Identifier)

→ Attribute name must start with "xmlns"

Syntax:- <element xmlns: name = "URI" >

Prefix → URI

element and Attributes names belongs to URL.

Conflict: Generally conflict occurs when we try to mix XML documents from diff. XML Application.

Eg. of Conflict.

1.xml

```
<class>
  <name>____</name>
</class>
```

2.xml

```
<class>
  <name>____
  </name>
</class>
```

Conflict occurs due to same element name.

Example of Namespace: 1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<c1:class xmlns:c1="class1....">
  <c1:name> Aman </c1:name>
</c1:class>
```

URI

Prefix

Now there will be no conflict due to namespace.

2.xml

```
<c2:class xmlns:c2="Class 2...">
  <c2:name> Aman </c2:name>
</c2:class>
```

(XML- Schemas)

Easy Engineering Classes – Free YouTube Lectures

For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

XML-Schemas:- Commonly known as XML Schema Definition (XSD). It is used to describe & validate the structure and content of XML Data.

→ It is a method of expressing constraints about XML documents.

→ It is like DTD but provides more control on XML structure.

Syntax: <xs: schema xmlns: xs="____">

Definition Types

Simple Type

used only in the content of the text.

eg:- xs: Int, xs: String.

<xs: element name="Phone" type="xs: int"/>

Complex Type

It is the container for other element definitions. Allows you to specify which child elements an element can contain & to provide some structure within your XML documents.

eg:- of Complex Type (Add.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs: schema xmlns: xs="schemal....">
  <xs: element name="Address">
    <xs: complex Type>
    <xs: sequence>
      <xs: element name="Name" type="xs: string"/>
      <xs: element name="Phone" type="xs: int"/>
    </xs: sequence>
  </xs: complex type>
</xs: element>
</xs: schema>
```

Child elements should appear in sequence.

(Add.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<Address
  xsi: schemalocation="____Add.xsd">
  1<Name> Aman </Name>
  2<Phone> 9810 </Phone>
</Address>
```

Path of xml schema def^n.

123

abcd

Error

Easy Engineering Classes – Free YouTube Lectures

For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

HTML

i) Display Data (Look and Feel).

ii) Markup language itself

iii) Not Case sensitive

iv) Predefined Tags

v) Static

eg:-
```
<html>
<body>
<p> HTML INTRO
</p>
</body>
</html>
```
Predefined Tag

display

XML

i) Transport and Store the data.

ii) Provide framework to define markup languages.

iii) Case-Sensitive

iv) Can Create own tags

v) Dynamic

eg:-
```
<College>
<Class>
<Name> </Name>
</Class>
</College>
```
Custom tags

transport

DTD

i) Document Type definition

ii) doesn't support data-types.

iii) doesn't support name space.

iv) Doesn't define Order for child Elements

v) Not Extensible

eg:-
```
<!DOCTYPE Address[
<!Element Address (Name)
<!Element Name (#PCDATA)
]>
```
root

XSD

i) XML Schema definition.

ii) Supports

iii) Supports

iv) Order Can be defined.

v) Extensible → namespace.

eg:-
```
<xs: element name="Address">
  <xs: Complex Type>
    <xs: sequence>
      <xs: elemenent name="name" type="xs:string"/>
    </xs: sequence>
  </xs: Complex type>
</xs: element>
```
order

Easy Engineering Classes – Free YouTube Lectures

For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

(XML-DOM)

XML-DOM:- DOM is Document Object Model.

↳ DOM document is a collection of nodes or pieces of info^n organized in a hierarchy.

→ Tree-Based.

→ Defines a standard way to access and manipulate documents.

→ Programmer can modify / delete their content and can also create new elements.

→ Elements, their content (text & Attributes) are all known as Nodes.

Example: <h2 id="demo"> </h2>

HTML DOM.

```
<button type="button" onclick="
document.getElementById('demo).
InnerHTML="Hello"> click
</button>
```

```
<College>
  <class>
  <title>
    <Gender>
```

node

Tree like hierarchy.

node.

Example of XML DOM:-

```
//College.xml
<?xml version="1.0" encoding="UTF-8"?>
<College>
  <Branch Category="IT">
    <Students>60</Students>
    <HOD> ABC </HOD>
  </Branch>
  <Branch Category="ECE">
    <Students>100</Students>
    <HOD> XYZ </HOD>
  </Branch>
</College>
```

Get Value of XML element

```
t = xmlDoc.getElementsByTagName("Students")
[0].childNodes[0].
nodeValue;
```

o/p = 60

```
[1].childNodes[1].
nodeValue;
```

o/p:- 100

Easy Engineering Classes – Free YouTube Lectures

For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India

XML DOM Properties (x = node object)

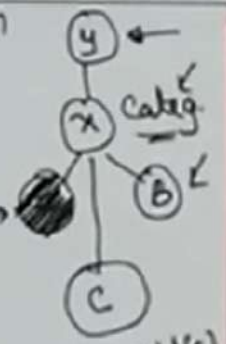
i) x.nodeName → name of x

ii) x.nodeValue → Value of x

iii) x.parentNode → Parent node of x

iv) x.childNodes → Child nodes of x

v) x.attributes → attributes of x.



x.appendChild(c)

XML DOM Methods

replace name with Tag name

i) x.getElementsByTagName(name) → get all elements with "name".

ii) x.appendChild(node) → insert a child node to x

iii) x.removeChild(node) → removes a child node from x.

x.removeChild(z);

⇧ SCROLL TO TOP

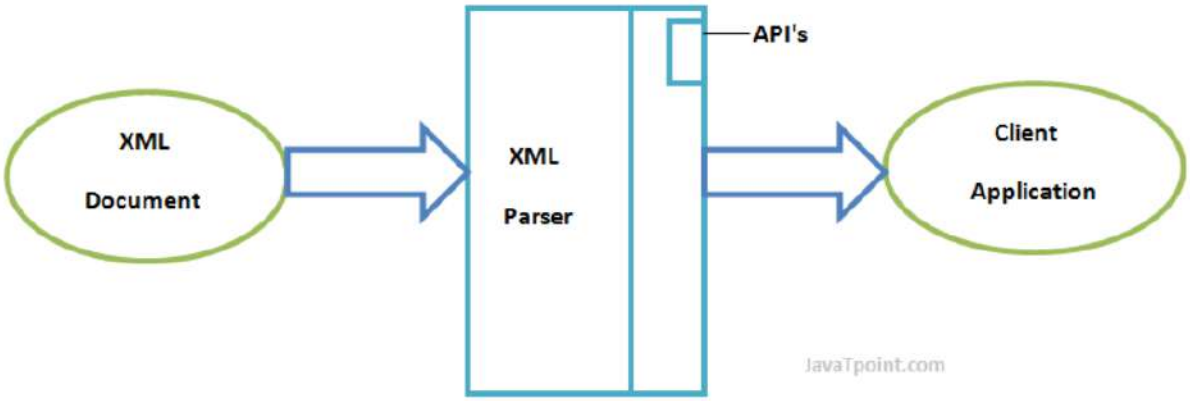
XML Parsers

← Prev Next →

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.

Let's understand the working of XML parser by the figure given below:



Types of XML Parsers

⇧ SCROLL TO TOP

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.

Features of DOM Parser

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

Advantages

1) It supports both read and write operations and the API is very simple to use.

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

Disadvantages

1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory).

SAX (Simple API for XML)

A SAX Parser implements SAX API. This API is an event based API and less intuitive.

Features of SAX Parser

It does not create any internal structure.

Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.

It is an event based parser, it works like an event handler in Java.

Advantages

1) It is simple and memory efficient.

2) It is very fast and works for huge documents.

his own code inside method.

It is an event based parser, it works like an event handler in Java.

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 intuitive.

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

Next Topic XML DOM

1)	DTD stands for **Document Type Definition**.	XSD stands for XML Schema Definition.
2)	DTDs are derived from **SGML** syntax.	XSDs are written in XML.
3)	DTD **doesn't support datatypes**.	XSD **supports datatypes** for elements and attributes.
4)	DTD **doesn't support namespace**.	XSD **supports namespace**.
5)	DTD **doesn't define order** for child elements.	XSD **defines order** for child elements.
6)	DTD is **not extensible**.	XSD is **extensible**.
7)	DTD is **not simple to learn**.	XSD is **simple to learn** because you don't need to learn new language.
8)	DTD provides **less control** on XML structure.	XSD provides **more control** on XML structure.

</TABLE></CENTER></body></html>

Difference :

S. No.	DTD	XSD
1.	DTD is a set of markup declarations that define a document type for an SGML.	XSD specifies how to describe the elements in an XML document formally.
2.	DTD stands for Document Type Definition.	XSD stands for XML Schema Definition.
3.	DTD provides less control over the XML structure.	XSD provides more control over the XML structure.
4.	DTD does not support data types.	XSD supports data types.

DTD is used to define the structure of XML documents.

2. SAX :

a. SAX stands for Simple API for XML and works directly with an XML.

b. SAX is an event-driven API that allows us to interpret a web file that uses XML.

c. SAX takes the control of event specifies by the programmer and handles the situation.

Advantages :

i. It is simple and memory efficient.

ii. It is very fast and works for huge documents.

Disadvantages :

i. It is event based so its API is less intuitive.

ii. Clients never know the full information because the data is broken into pieces.

Que 2.27. **"Document Type Definition (DTD) in XML is necessary", justify the statement with suitable example. Under which**

XML Schema (XSD) Beginner Tutorial with Demo

```
<class>
    <student>
        <firstname>Graham</firstname>
        <lastname>Bell</lastname>
        <age>20</age>
    </student>
</class>
```

XML

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="class">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="student">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="firstname" type="xs:string"/>
       <xs:element name="lastname" type="xs:string"/>
       <xs:element name="age" type="xs:int"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

XSD

4:53 / 9:43