UNIT – 1

Syllabus:

CSS 3: Syntax structure, using style sheets, Box model, Grid, Flexbox. Responsive Web Design using Media Queries, use of viewport, Transition, Animation. CSS Framework: Bootstrap.

XML: Introduction, Syntax, Validating XML with Document type definition and XML Schemas.

---

eXtensible Markup Language (XML)

➢ INTRODUCTION:

- XML is a markup language much like HTML and is derived from Standard Generalized Markup Language (SGML).
- XML was ***designed to store and transport data*** and was designed to be self-descriptive
- ***Definition:*** *"XML defines a set of rules for encoding documents in a format that is both Human Readable and Machine Understandable"*
- It is a W3C Recommendation.
- The first XML version 1.0 was published in 1998.
- HTML limits you to use only predefined tags, whereas XML ***allows us to create our own tags*** (user-defined tags).
- It is used to *exchange data or information* across the Internet.
- It is used to *describe the Structure of a document* not the way it is presented.
- In XML data is stored in Plain Text format as it is designed to be Portable and Platform Independent language.
- It is a Case Sensitive Language.
- XML files are saved with extension .xml

Advantages of XML:

- *Separates Data from HTML:* With XML, data can be stored in separate XML files. As a result, changes in underlying data will not require any changes to the HTML. With a few lines of JavaScript code, we can read an external XML file and update the data contents of a web page.
- *XML simplifies data sharing:* In XML, data is stored in plain text format. This provides a software and hardware independent way of storing data.

- The XML document is *language neutral*. That means a Java program can generate an XML document and this document can be parsed by Perl program.
- *XML simplifies data transport*: Exchanging data as XML greatly reduces the complexity of exchanging data between in compatible systems over the internet, since the data can be read by different in compatible applications
- *XML simplifies platform changes:* XML data is stored in text format. This makes it easier to expand or upgrade to new Operating Systems, applications, browsers, hardware without losing data.
- XML files are independent of an operating system.
- *Makes data more available*: Different applications can access data from XML data sources.

Differences between XML and HTML:

	XML	**HTML**
1.	eXtensible Markup Language	Hyper Text Markup Language
2.	It is used to store and transport the data.	It is used to present the content and is a presentation language
3.	Focus is on what data is i.e describes the structure of a document	Focus is on how data looks
4.	It supports user defined tags.	It supports only predefined tags
5.	XML is case sensitive	HTML is not case sensitive
6.	Root element is user defined and only one root element is allowed.	Root element is <HTML>
7.	All XML elements must have a Closing tag	Not required to close each and every tag Example: <img><input>
8.	Attribute values need to be quoted.	Attribute values may or may not be quoted.
9.	Saved as .xml files	Saved as .html files

- **BASIC STRUCTURE / SYNTAX OF AN XML DOCUMENT:**

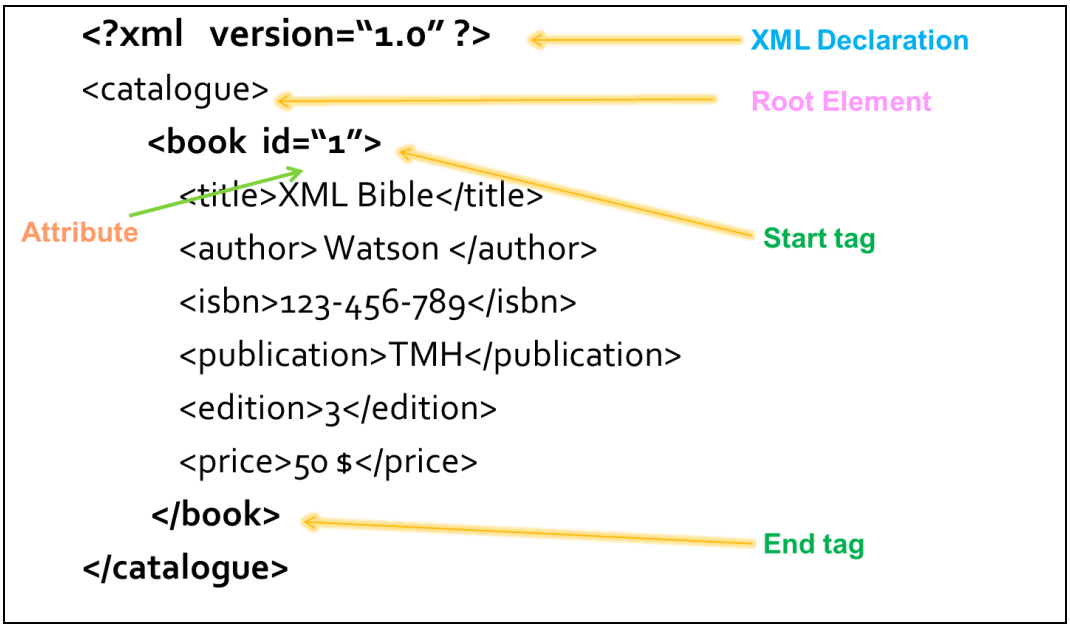
```
<?xml  version="1.0"  encoding="UTF-8" ?>
<rootelement>
  <childelement>
    <subchild>    -    </subchild>
      .
      .
  </childelement>
</rootelement>
```

- XML Document mainly consists of 2 parts:
  - **XML Prolog**
  - **XML Body** – Consists of elements and attributes
  - The *prolog* of an XML document comprises everything from the start of the file to the document root tag. It may contain the
    - XML declaration,
    - Processing Instructions,
    - Comments, and a
    - DTD.
  - The first line in an XML file is the XML declaration.
    - identifies the document as being XML.
    - defines the XML standard version the label adheres to, and also
    - defines the character set to be used.
- **XML Syntax rules:**
  - XML documents must contain one root element that is the parent of all other elements.
  - The XML prolog is optional. If it exists, it must come first in the document.
  - All XML Elements Must Have a Closing Tag
  - XML Tags are Case Sensitive
  - XML Elements Must be Properly Nested - Closed in the reverse order in which they are opened
  - XML Attribute Values Must Always be Quoted

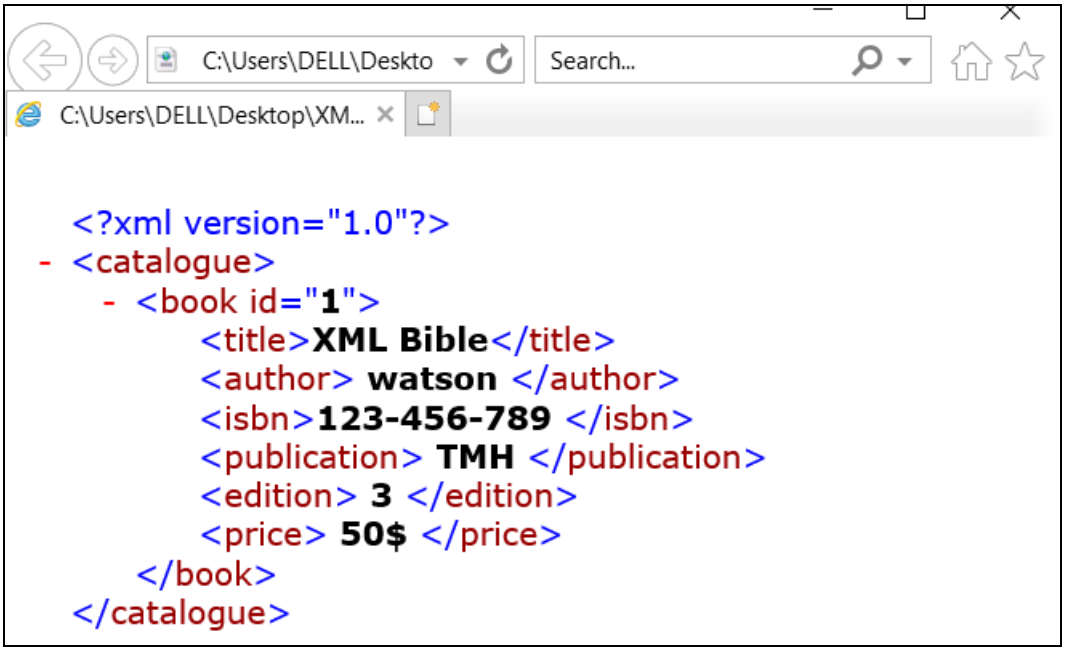
- White-space is Preserved in XML and it does not truncate multiple white-spaces
- XML documents that conform to the syntax rules above are said to be "Well Formed" XML documents.

Example :

book.xml



If the above book.xml is opened using any of the browsers (Example, Internet Explorer), following is the output.

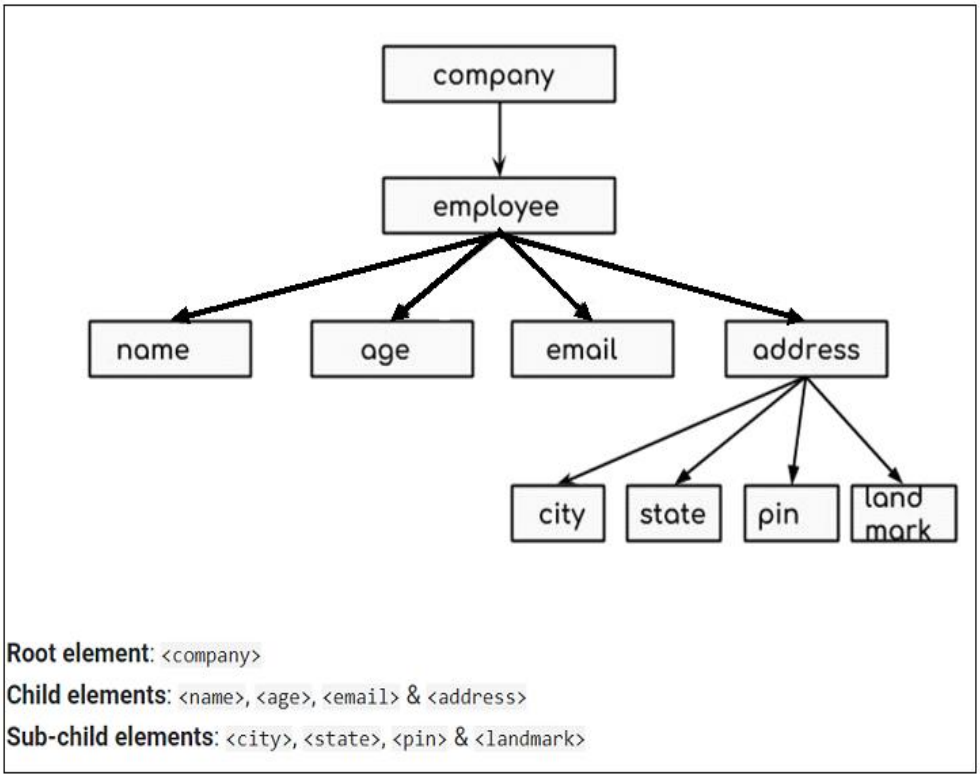


XML Tree Structure:

- XML documents have a hierarchical structure and can conceptually be interpreted as a tree structure, called an XML tree.

Employee.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<company>
  <employee>
    <name>Negan</name>
    <age>40</age>
    <email>imnegan@twd.com</email>
    <address>
      <city>Noida</city>
      <state>Uttar Pradesh</state>
      <pin>201301</pin>
      <landmark>Near hill top</landmark>
    </address>
  </employee>
</company>
```



10. **BASIC BUILDING BLOCKS:**

- XML documents are composed of 3 things:

1. Elements

- Nested Tags
- Case Sensitive
- Empty Tags
- Attributes

2. Control Information

- Comments
- Processing Instructions
- DTD

3. Entities

1. Elements:

Elements are the **main building blocks** of both XML and HTML documents and are used for defining the tags. The elements typically consist of opening and closing tag. Mostly only one element is used to define a single tag.

- The syntax of writing any element for opening tag is <element-name>
- The syntax of writing any closing element for closing tag is </element-name>
- An empty tag can be defined by putting a / (forward slash) before closing bracket.
- A space or a tab character is not allowed in the element name or in attribute name.
- *Examples:*
  <body> some text </body>
  <message>some text</message>

1.1 Nested Tags:

- Tags inside Tags
- Should be properly nested and closed in reverse order in which they were opened
- **Example:**

```
<catalogue>
        <book>
                ----
        </book>
</catalogue>
```

1.2 Case Sensitive:

- Generally, Lowercase letters are used for tags
- Example: <author> & <Author> are different

1.3 Empty tags/elements:

- Elements without content
- Place '/' before closing angle bracket
- Example: <br **/**>

1.4 Attributes:

- Attributes provide **extra information about elements**.
- Attributes are always placed inside the opening tag of an element. Attributes always come in name/value pairs.
- Example: <book **id="101"** > …….. </book>

2. Control Information:

2.1 XML Comments:

- A comment starts with <!-- and ends with -->
- Following rules should be followed for XML comments −
  - Comments cannot appear before XML declaration.
  - Comments may appear anywhere in a document.
  - Comments must not appear within attribute values.
  - Comments cannot be nested inside the other comments.
- Syntax: **<!--** Comment Text **-- >**
- Example: **<!--** XML Introduction **-->**

2.2 Processing Instructions:

- Used to pass information to applications
- Need to be processed by the parser.
- Syntax: <?target instruction ?>
- Example: <?xml-stylesheet href = "mySheet.css" type = "text/css"?>

2.3 Document Type Definition:

- Defines the basic building blocks of an XML file

3. Entities:

- Some characters have a special meaning in XML, like the less than sign (<) that defines the start of an XML tag.
- *Example:* <message>salary **<** 1000</message>
  - **<** generates an error because the parser interprets it as the start of a new element.
  - To avoid this error, replace the "<" character with an entity reference <

- *Definition:* Entities are variables used to define shortcuts to common text or complex data.
- Entities were referenced in XML as **&entityname;**
- *Example :*

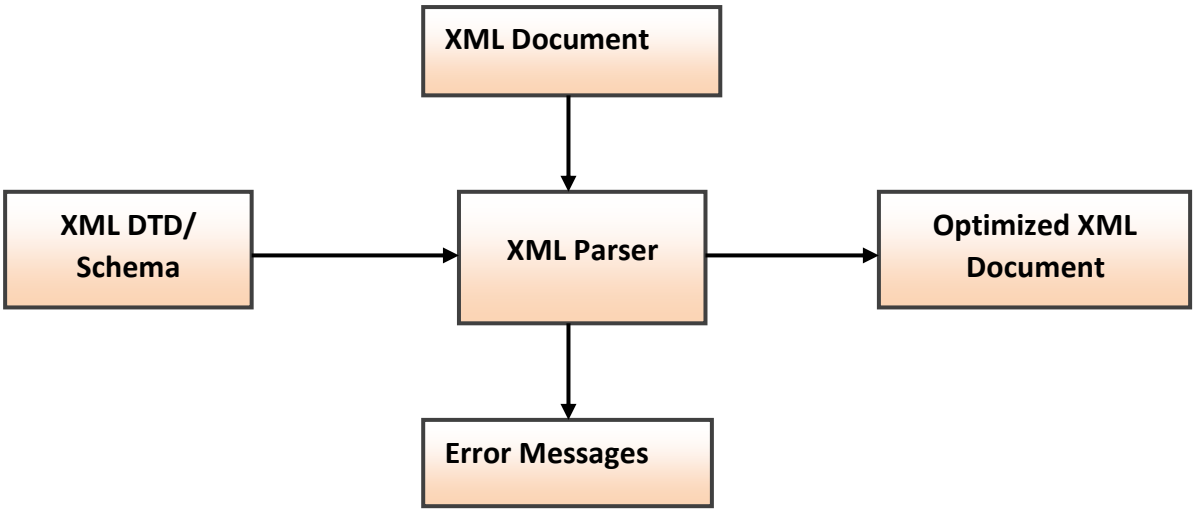
  kalam **"Avul Pakir Jainulabdeen Abdul Kalam"**

  entityname *Value*

  In XML: <name> **&kalam** </name>
- Entities are expanded when a document is parsed by an XML parser.
- 3 types of entities:
  - Internal Entities
  - External Entities
  - Predefined Entities
- The following entities are predefined in XML:

Entity Reference	Character
&lt	<
&gt	>
&amp	&
&quot	"
&apos	'

- **VALIDATING AN XML FILE:**
  - The primary goal of any XML processor/parser is to parse the given XML document.



- Parsers are used to check whether a given XML document
  i. Is Well formed or not – It follows all of the syntax rules of XML.
  ii. Is Valid or not - It obeys its own internal rules defined in the DTD or XML Schema
- An XML file can be validated using the following specifications.
  1. DTD (Document type definition)
  2. XML Schema.

1. DOCUMENT TYPE DEFINITION (DTD):

- The document type definition used to define the basic building block of any xml document.
- Using DTD we can specify the various elements types, attributes and their relationship with one another.
- **Example:** Elements, attributes, entities, order of elements and their occurrence
- Basically DTD is used to specify the set of rules for structuring data in any XML file.
- It is the grammar against which XML document is to be validated.
- Many developers recommend writing DTDs for the XML applications.
- DTD standards are defined by the W3C.

Data Types in DTD:

1) PCDATA

- PCDATA means Parsed Character DATA.
- PCDATA is text that WILL be parsed by a parser. The text will be examined by the parser for entities and mark up.
- It is used with Elements
- Tags inside the text will be treated as markup and entities will be expanded.

2) CDATA

- CDATA means Character DATA.
- It is used with Attributes.
- CDATA is plain text character data that is not parsed by the XML parser.

DTD Syntax for Elements:

- **Simple Elements:** Elements that contain only text
  Syntax:
  <!ELEMENT element-name (#PCDATA) >
  Example:
  <title> This is a title </title>
  <!ELEMENT title (#PCDATA) >

- **Compound Elements**: Elements that contain child elements
  Syntax:
  <!ELEMENT element-name (child1, child2, -- - ->
  Example:

```
<employee>
        <name>XXX</name>
        <id>123</id>
        <dept>CSE</dept>
</employee>
<!ELEMENT employee (name, id, dept)>
```

DTD Syntax for Attributes:

- Attributes are (name,value) pairs
- They add additional information about element's content
- Syntax:

  <!ATTLIST element-name attribute-name attribute-type attribute-value) >
- Attribute values:
  - Value - default value
  - #REQUIRED - mandatory attribute
  - #IMPLIED – may or may not be present
  - #FIXED value - value cannot be changed

Occurrence indicators in DTD:

Operator	Syntax	Description
None	a	Exactly one occurrence of a
*	a*	Zero or more occurrences of a
+	a+	One or more occurrences of a
?	a?	Zero or one occurrence of a

- **TYPES OF DTD:**

There are two ways of writing DTDs

1. Internal DTD
2. External DTD

Internal DTD

The DTD can be embedded directly in the XML document as a part of it.

The general syntax for an internal DTD is

```
<!DOCTYPE root-element [
    <!-- Rest of DTD -->
]>
<RootElement>
<!-- Rest of XML File-->
```

The keyword DOCTYPE specifies that a DTD is to be used by the document.

The following rules must be followed:

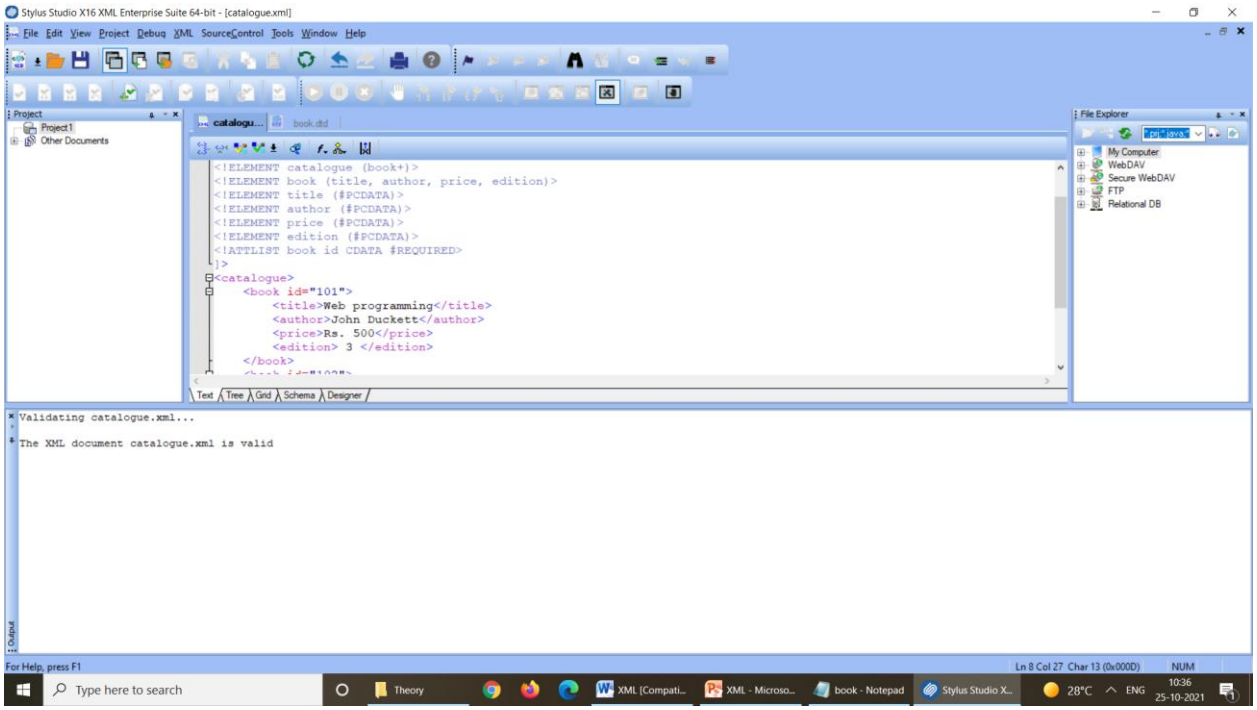
- The keyword DOCTYPE must be in upper case (Well formedness constraint).
- The document type declaration must appear before the first element in the document (Well formedness constraint).
- Following the word DOCTYPE (root-element in this case) must match with the name of the root-element(Top-level element) in the xml document.

Example:

Books.xml

```
<?xml version="1.0"?>
<!DOCTYPE catalogue [
    <!ELEMENT catalogue (book+)>
    <!ELEMENT book (title, author, price, edition)>
    <!ELEMENT title (#PCDATA)>
    <!ELEMENT author (#PCDATA)>
    <!ELEMENT price (#PCDATA)>
    <!ELEMENT edition (#PCDATA)>
]>
<!ATTLIST book id CDATA #REQUIRED>>
<catalogue>
    <book id="101">
        <title>Web programming</title>
        <author>John Duckett</author>
```

```
        <price>Rs. 500</price>
        <edition> 3 </edition>
    </book>
    <book id="102">
        <title>Computer Networks</title>
        <author>Tanenbaum</author>
        <price>Rs. 700</price>
        <edition> 7 </edition>
    </book>
</catalogue>
```



External DTD

- In this type, an external DTD file is created and is saved with extension .dtd
- The external DTD file must be included in the corresponding XML file using <!DOCTYPE>

Syntax: <!DOCTYPE RootElement SYSTEM|PUBLIC "URL of .DTD file" >

The following example illustrates the use of external DTD.

Step 1: Creation of DTD file [Validatebooks.dtd]

```
<!ELEMENT catalogue (book+)>
<!ELEMENT book (title, author, price, edition)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT edition (#PCDATA)>
<!ATTLIST book id CDATA #REQUIRED>
```

Step 2: XML document with DOCTYPE to include external DTD [BooksFile.xml]

```
<?xml version="1.0"?>
<!DOCTYPE catalogue SYSTEM "Validatebooks.dtd">
<catalogue>
        <book id="101">
                <title>Web programming</title>
                <author>John Duckett</author>
                <price>Rs. 500</price>
                <edition> 3 </edition>
        </book>
        <book id="102">
                <title>Computer Networks</title>
                <author>Tanenbaum</author>
                <price>Rs. 700</price>
                <edition> 7 </edition>
        </book>
</catalogue>
```

Disadvantages of DTDs:

- Doesn't support different data types – It supports only the *text string data type.*
- Syntax is different from XML
- Doesn't support namespaces –
  - Namespace is a mechanism by which element and attribute names can be assigned to groups.
  - Used to resolve naming conflicts

---

2. XML SCHEMA:

- XML Schema is commonly known as XML Schema Definition (XSD) language.
- It is similar to a database schema that describes the data in a database.
- It is used to describe and validate the structure and the content of XML data.
- It describes the basic building blocks of XML document i.e elements, attributes, data types and their order of occurrence.
- Schemas support different data types (numeric, boolean, data, time, string)

- Schema element supports Namespaces.

Syntax :

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">


</xs:schema>
```

XML Schema Data types:

- String – set of characters placed within quotes.
- Date – format should be ”yyyy – mm – dd”
- Numeric (decimal, short, int, integer, float, long, double)
- Boolean – value can be true/false
- Time – format should be “hh : mm : ss”
- Syntax:
  - <xs:element name=“xxx” type=“xs:datatype” />

XML Schema syntax for elements:

- **Simple Type** – contains only text
  - <xs:element name=“xxx" type="xs:datatype“ />
  - **Example:**

    XML: <phone> 1234567890 </phone >

    <xs:element name="phone " type="xs:int">
- **Complex Type** – contains attributes and/or child elements
  - <xs:element name="xxx">
    <xs:complexType>
    <xs:sequence>
    <! – Child Elements -->
    </xs:sequence>
    <! – Attributes -->
    </xs:complexType>
    </xs:element>
  - **Example:**
    - XML: <address>
      <name></name>
      <company></company>
      <phone></phone>

```
        </address>
    o   <xs:element name="Address">
            <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>
```

XML Schema syntax for Attributes:

Syntax:

<xs:attribute name = “xxx” type=“xs:datatype />

Values for attributes:

- Default
  <xs:attribute name = “xxx” type=“xs:datatype" default= “DefaultValue” />
- Fixed
  <xs:attribute name = “xxx” type=“xs:datatype" fixed= “FixedValue” />
- Optional
- Required
  <xs:attribute name = “xxx” type=“xs:datatype use= “required”/>

Syntax to include XML Schema in an XML file:

- <rootelement **xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation=".xsd"**>

Example:

XML:

```
<?xml version="1.0"?>
<catalogue>
        <book id="101">
                <title> XML Bible </title>
                <author> Watson </author>
                <edition> 3</edition>
        </book>
</catalogue>
```

XML Schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema
<xs:element name=”catalogue>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="book" maxOccurs=”unbounded”>
          <xs:complexType>
            <xs:sequence>
                <xs:element name="title" type="xs:string" />
                <xs:element name="author" type="xs:string" />
                <xs:element name="edition" type="xs:int" />
            </xs:sequence>
          </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="id" type="xs:int"  use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Difference between DTD and XSD

	DTD	XSD
1)	DTD stands for Document Type Definition.	XSD stands for XML Schema Definition.
2)	DTDs are derived from SGML syntax. Their syntax is different from XML	XSDs are written in XML.
3)	DTD doesn't support data types other than text.	XSD supports different data types 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