1. **Explain XML with advantages and disadvantages.**

**XML** is a markup language designed to **store, structure, and transport data** in a **human-readable** and **machine-readable** format. It is widely used in web services, configuration files, and data exchange between different systems.

**Key Features of XML:**

✔ **Self-descriptive** – Uses tags to define data structure.  
✔ **Platform-independent** – Works across different operating systems and programming languages.  
✔ **Extensible** – Users can create custom tags.  
✔ **Supports Validation** – Can be validated using **DTD** or **XML Schema (XSD)**.  
✔ **Supports Unicode** – Can represent text in multiple languages.

**Advantages of XML**

**Structured Data Representation**

* Organizes data hierarchically (parent-child relationships).

**Interoperability**

* Can be parsed and processed by different programming languages (Java, Python, C#, etc.).

**Supports Metadata with Attributes**

* Additional information can be stored using attributes.

<book category="fiction"> ... </book>

**Validation Support**

* Ensures data consistency using **DTD** or **XML Schema**.

**Human and Machine Readable**

* Easier to understand than binary formats.

**Widely Used in Web Services (SOAP, RSS, SVG, etc.)**

**Disadvantages of XML**

**Verbose Syntax**

* Requires more storage and bandwidth compared to **JSON** or **binary formats**.

**Slower Parsing**

* Processing large XML files can be slower than JSON or CSV.

**Complexity**

* More difficult to parse than simpler formats like JSON.

**No Built-in Data Types**

* Unlike XML Schema, **DTD does not support data types** (e.g., integer, date).

**Redundant Tags Increase File Size**

* Example:

<name>John</name> <!-- JSON: {"name": "John"} -->

**Q.2. Write XML code to demonstrate the use of elements and attributes.**

1. **Elements** (Nested Tags):
   * <catalog>, <book>, <title>, <author>, <year>, <price>
   * Elements define the **structure** and **content** of the data.
2. **Attributes** (Inside Opening Tags):
   * id="101", category="Fiction", lang="en", currency="USD"
   * Attributes provide **additional metadata** about an element.

<?xml version="1.0" encoding="UTF-8"?>

<catalog>

<!-- Book with attributes and nested elements -->

<book id="101" category="Fiction">

<title lang="en">The Great Gatsby</title>

<author>F. Scott Fitzgerald</author>

<year>1925</year>

<price currency="USD">12.99</price>

</book>

<!-- Another book -->

<book id="102" category="Science">

<title lang="en">A Brief History of Time</title>

<author>Stephen Hawking</author>

<year>1988</year>

<price currency="USD">15.50</price>

</book>

</catalog>

**Q.3. Explain XML namespace with example.**

**Namespace** in XML is a mechanism to avoid **element name conflicts** when combining XML documents from different sources. It assigns a unique identifier (URI) to a group of tags.

Imagine two XML documents with a <table> element:

* One defines a **wooden table** (furniture).
* Another defines a **data table** (HTML).

Without namespaces, the parser wouldn’t know which <table> is being referred to.

**Syntax of XML Namespace**

1. **Declare a namespace** using xmlns: <root xmlns:prefix="URI">
   * prefix is an alias (e.g., furniture, html).
   * URI is a unique identifier (often a URL, but it doesn’t need to resolve).
2. **Use the namespace** with the prefix: <prefix:element>Content</prefix:element>

**Example:**

<?xml version="1.0" encoding="UTF-8"?>

<root xmlns:furniture="http://example.com/furniture"

xmlns:html="http://example.com/html">

<!-- Furniture table -->

<furniture:table>

<furniture:material>Oak</furniture:material>

</furniture:table>

<!-- HTML table -->

<html:table>

<html:tr>

<html:td>Data Cell</html:td>

</html:tr>

</html:table>

</root>

**Q.4. Explain XML namespace with example.**

**What is a DTD?**

A **DTD (Document Type Definition)** defines the structure and legal elements of an XML document. It acts like a "rulebook" that specifies:

* What elements can appear
* How elements can be nested
* What attributes elements can have
* The order of elements

**Types of DTDs**

1. **Internal DTD** (Embedded in the XML file)
2. **External DTD** (Stored in a separate .dtd file)

**Example: Internal DTD**

<?xml version="1.0"?>

<!DOCTYPE bookstore [

<!ELEMENT bookstore (book+)> <!-- Must have 1+ books -->

<!ELEMENT book (title, author, price)>

<!ATTLIST book category CDATA #REQUIRED> <!-- Required attribute -->

<!ELEMENT title (#PCDATA)> <!-- Text content -->

<!ELEMENT author (#PCDATA)>

<!ELEMENT price (#PCDATA)>

]>

<bookstore>

<book category="fiction">

<title>Harry Potter</title>

<author>J.K. Rowling</author>

<price>29.99</price>

</book>

</bookstore>

**Example: External DTD**

<?xml version="1.0"?>

<!DOCTYPE bookstore SYSTEM "books.dtd">

<bookstore>

<book category="fiction">

<title>Lord of the Rings</title>

<author>J.R.R. Tolkien</author>

<price>24.99</price>

</book>

</bookstore>

**DTD File (books.dtd):**

<!ELEMENT bookstore (book+)>

<!ELEMENT book (title, author, price)>

<!ATTLIST book category CDATA #REQUIRED>

<!ELEMENT title (#PCDATA)>

<!ELEMENT author (#PCDATA)>

<!ELEMENT price (#PCDATA)>

**Q.5. Explain XML Schema with example.**

XML Schema (XSD - XML Schema Definition) is a more powerful alternative to DTD that:

* Defines the structure and data types of XML documents
* Supports namespaces
* Uses XML syntax (unlike DTD)
* Provides strong data validation (numbers, dates, patterns, etc.)

**Example : Simple Book Schema (XSD)**

**XML File (**book.xml**):**

<?xml version="1.0"?>

<book xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="book.xsd">

<title>Learning XML</title>

<author>John Doe</author>

<year>2023</year>

<price currency="USD">29.99</price>

</book>

**Schema File (**book.xsd**)**

<?xml version="1.0"?>

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

<xs:element name="book">

<xs:complexType>

<xs:sequence>

<xs:element name="title" type="xs:string"/>

<xs:element name="author" type="xs:string"/>

<xs:element name="year" type="xs:integer"/>

<xs:element name="price">

<xs:complexType>

<xs:simpleContent>

<xs:extension base="xs:decimal">

<xs:attribute name="currency" type="xs:string" use="required"/>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

**Q.6. Create book\_list,xml file that stores the information about books. The XML file should contain the title, author, publisher, price and year.**

**Now display the content of that XML file using HTML table.**

1. **XML File:**
   * Contains book data in a structured format with root element <book\_list>
   * Each book has <title>, <author>, <publisher>, <price>, and <year> elements
2. **HTML Page:**
   * Uses JavaScript to fetch and parse the XML file
   * Dynamically creates table rows for each book
   * Applies CSS styling for a clean presentation
3. **Features:**
   * Responsive table design
   * Error handling for XML loading
   * Modern fetch API for XML loading
   * Zebra-striping for better readability

**a. book\_list.xml**

<?xml version="1.0" encoding="UTF-8"?>

<book\_list>

<book>

<title>Atomic Habits</title>

<author>James Clear</author>

<publisher>Penguin Random House</publisher>

<price>14.99</price>

<year>2018</year>

</book>

<book>

<title>The Psychology of Money</title>

<author>Morgan Housel</author>

<publisher>Harriman House</publisher>

<price>12.50</price>

<year>2020</year>

</book>

<book>

<title>Deep Work</title>

<author>Cal Newport</author>

<publisher>Grand Central Publishing</publisher>

<price>16.99</price>

<year>2016</year>

</book>

</book\_list>

**b. display\_books.html**

<!DOCTYPE html>

<html>

<head>

<title>Book List</title>

<style>

table {

border-collapse: collapse;

width: 80%;

margin: 20px auto;

}

th, td {

border: 1px solid #ddd;

padding: 8px;

text-align: left;

}

th {

background-color: #f2f2f2;

}

tr:nth-child(even) {

background-color: #f9f9f9;

}

</style>

</head>

<body>

<h1 style="text-align: center;">Book Collection</h1>

<table id="bookTable">

<thead>

<tr>

<th>Title</th>

<th>Author</th>

<th>Publisher</th>

<th>Price ($)</th>

<th>Year</th>

</tr>

</thead>

<tbody id="bookData">

<!-- Books will be inserted here by JavaScript -->

</tbody>

</table>

<script>

// Load XML file

fetch('book\_list.xml')

.then(response => response.text())

.then(xmlString => {

const parser = new DOMParser();

const xmlDoc = parser.parseFromString(xmlString, "text/xml");

// Get all book elements

const books = xmlDoc.getElementsByTagName("book");

const tableBody = document.getElementById("bookData");

// Loop through books and add to table

for (let book of books) {

const row = document.createElement("tr");

const title = book.getElementsByTagName("title")[0].textContent;

const author = book.getElementsByTagName("author")[0].textContent;

const publisher = book.getElementsByTagName("publisher")[0].textContent;

const price = book.getElementsByTagName("price")[0].textContent;

const year = book.getElementsByTagName("year")[0].textContent;

row.innerHTML = `

<td>${title}</td>

<td>${author}</td>

<td>${publisher}</td>

<td>${price}</td>

<td>${year}</td>

`;

tableBody.appendChild(row);

}

})

.catch(error => console.error('Error loading XML:', error));

</script>

</body>

</html>