**XML Overview**

XML (Extensible Markup Language) is a markup language designed to store and transport data, focusing on simplicity, generality, and usability across the Internet.

**1. Introduction**

* **Purpose**: Define and transport structured data.
* **Features**: Human-readable, platform-independent, self-descriptive tags.

**2. Tree Structure**

* XML documents form a **tree structure** with one root element and nested child elements.
  + **Example**:

xml

Copy code

<root>

<child>

<subchild>Data</subchild>

</child>

</root>

**3. Syntax**

* Case-sensitive tags.
* Each element must have a start (<tag>) and end (</tag>) tag.
* Proper nesting of tags is required.
* XML must be well-formed (correct syntax).

**4. Elements**

* **Building blocks** of XML.
* Can contain text, other elements, or both.
* **Example**:

xml

Copy code

<note>

<to>John</to>

<from>Jane</from>

</note>

**5. Attributes**

* Provide additional information about elements.
* Written within the start tag.
  + **Example**:

xml

Copy code

<person age="30">John Doe</person>

**6. Namespaces**

* Used to avoid element name conflicts by qualifying names with a prefix.
  + **Example**:

xml

Copy code

<bookstore xmlns:fiction="http://example.com/fiction">

<fiction:book>Title</fiction:book>

</bookstore>

**7. Display**

* XML does not define how data is displayed; it's only for data storage and transport.
* Displayed using **XSLT** or transformed into HTML/CSS.

**8. HTTP Request**

* **XMLHttpRequest** object is used in web development to request data from a server.
  + **Example**:

javascript

Copy code

var xhttp = new XMLHttpRequest();

xhttp.open("GET", "data.xml", true);

xhttp.send();

**9. Parser**

* Converts XML into readable data for programs.
* Types:
  + **DOM Parser**: Loads the entire XML into memory as a tree.
  + **SAX Parser**: Reads XML as a stream of events, more memory efficient.

**10. DOM (Document Object Model)**

* An interface to interact with XML data as a tree structure.
* Provides methods to navigate, modify, and access elements.
  + **Example**:

javascript

Copy code

var xmlDoc = xhttp.responseXML;

var title = xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;

**11. XPath**

* A query language used to navigate through elements and attributes in an XML document.
  + **Example**:

xpath

Copy code

/bookstore/book[price>30]

**12. XSLT (Extensible Stylesheet Language Transformations)**

* Used to transform XML documents into other formats like HTML or plain text.
  + **Example**:

xml

Copy code

<xsl:stylesheet>

<!-- Transformation logic -->

</xsl:stylesheet>

**13. XQuery**

* A powerful query language designed to query XML data across collections of documents.
  + **Example**:

xquery

Copy code

for $book in doc("books.xml")/bookstore/book

where $book/price > 30

return $book/title

**14. XLink**

* Defines hyperlinks within XML documents.
  + **Example**:

xml

Copy code

<link xlink:href="http://example.com" xlink:type="simple">Visit</link>

**15. Validator**

* Validates whether an XML document adheres to a specific **DTD** or **XML Schema**.

**16. DTD (Document Type Definition)**

* Defines the structure, rules, and constraints of an XML document.
  + **Example**:

xml

Copy code

<!DOCTYPE note [

<!ELEMENT note (to,from,heading,body)>

]>

**17. Schema**

* XML Schema (XSD) defines the structure of an XML document with more precision than DTD.
  + **Example**:

xml

Copy code

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

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

</xs:schema>

**18. Server**

* XML is commonly used for server-client communication.
* Servers send and receive XML data for web applications (REST APIs, SOAP).