Unit-I

Role of XML- XML language basics- XML revolution- XML technology family, Simple XML file creation- XML Namespaces

**Extensible Markup Language (XML)**

Extensible Markup Language (XML) is a simple data description language. XML is a metalanguage (literally a language about languages) defined by the World Wide Web Consortium (W3C). XML is a set of rules and guidelines for describing structured data in plain text rather than proprietary binary representations.

*************

**Introduction**

The need of XML: Extending the Enterprise

There are three major aspects to extending the enterprise from a relatively constrained network to the broad reach of the Web. The global connectivity, coupled with Internet standards for communication, has spawned new ideas about how to leverage this new capability. There are three major aspects to extending the enterprise. The most commonly considered aspect is the

- **Business-to-consumer (B2C)** connection, exploiting opportunities that abound in online commerce.
- **Business-to-employee (B2E)** connection, adding efficiencies in operations and customer contact by using the Web instead of proprietary networks.
- A third area, businesses trying to survive in competitive environments, is the **business-to-business (B2B)** connection.

Together, these opportunities are driving what is seen as the extended enterprise, a mix of traditional networks and the loose space of the Web.

Figure 1.1. The extended enterprise combines traditional networks with the power of the Web, opening up new opportunities in B2B, B2C, and B2E commerce.
The Role of XML

XML is a specification for defining new markup languages.

XML is a metalanguage (literally a language about languages) defined by the World Wide Web Consortium (W3C). XML is a set of rules and guidelines for describing structured data in plain text rather than proprietary binary representations.

XML has enabled industry vocabularies and protocols.

XML has given rise to numerous vertical industry vocabularies in support of B2B e-commerce, horizontal vocabularies that provide services to a wide range of industries, and XML protocols that have used XML's simple power of combination to open up new possibilities for doing distributed computing. As Figure shows, XML's influence has been felt in three waves, from industry-specific vocabularies, to horizontal industry applications, to protocols that describe how businesses can exchange data across the Web.

Figure 1.2. XML has been widely used as a language for a variety of applications ranging from vertical industry vocabularies, to horizontal industry applications, to protocols.
XML Language Basics

XML is simple. Technically, it's a language for creating other languages based on the insertion of tags to help describe data. XML is a combination of tags and content in which the tags add meaning to the content.

XML stands for "Extensible Markup Language," a language developed by the World Wide Web Consortium (W3C). It is considered a meta language because it is used to define other languages through the use of markup tags, which add structure and meaning to documents.

While XML markup tags look like HTML tags, they allow designers to describe the content rather than the format of the text they enclose. When describing data, developers are free to name their tags in whatever way best describes their content. This flexibility is what makes XML an extensible language.

XML Fundamentals

Although XML includes several language components, most individual XML vocabularies can be read and understood by focusing on three commonly used XML structures:

- **Elements**
- **Attributes**
- **Entities**.

**Elements and attributes:** are used to describe content

**Entities** are substitutes for special or commonly used character strings.

**Elements**

An XML element is made up of a start tag, an end tag, and data in between. The start and end tags describe the data within the tags, which is considered the value of the element. Example:

```
<book> The complete reference Java 2 </book>
```

open tag content close tag

Element
Example

```xml
<Customer>
  <Name>RAMESH</Name>
  <PhoneNum>9442121215</PhoneNum>
  <FaxNum>044-45678345</FaxNum>
  <E-Mail>ramesh@gmail.com</E-Mail>
</Customer>
```

Start tags such as `<Name>` begin an element that contains the actual data. End tags such as `</Name>` mark the end of an element definition.

Elements are the primary means for describing data in XML. The rules for composing elements are flexible, allowing for different combinations of text content, attributes, and other elements.

There are three ways elements are used in XML documents.

1. Simple Content

Text or other data appears between start and end tags. The start tag has the same name as the end tag except that the end tag begins with a slash. The following element has a start tag, content, and an end tag.

```xml
<author> DR.E.BALAGURUSAMY </author>
```

2. Element as Container for Other Elements

An element may contain other elements, providing a hierarchical or tree data structure. The following `book` element contains the `author` and `title` elements.

```xml
<book>
  <author> DR.E.BALAGURUSAMY </author>
  <title> XML AND WEB SERVICES </title>
</book>
```

3. Empty Element as Container for Attributes

When an element has no content but only attributes, there is a shorthand way of writing the element that bypasses the need for both a start and end tag. An element written with a slash following the tag name indicates an empty element, as in
<book/> which is shorthand for <book></book>

The most common use of empty elements is to hold attribute data.

```xml
<book title="XML AND WEB SERVICES" author="DR.E.BALAGURUSAMY"/>
<book isbn="0-102-9393-3">
  XML AND WEB SERVICES
</book>
```

**Element Naming Rules**

The following are official naming rules for XML which must be followed.

- Names can contain letters, numbers, and other characters.
- Names must not begin with a number or punctuation.
- Names must not start with the string "xml" in any upper- or lowercase form.
- Names must not contain spaces.

Also, when designing element names, one should not use the colon, since it is reserved for use with XML namespaces.

**XML Attributes**

An element can optionally contain one or more attributes. An attribute is a name-value pair separated by an equal sign (=). Example:

```xml
<CITY PIN="600025">Chennai</CITY>
```

It's also possible to represent the data using attributes within a single element:

```xml
<Customer name="RAMESH" phone="9442121215"
  fax="044-45678345" email="ramesh@gmail.com "/>
```

Attributes provide additional information about elements. In HTML, for example, attributes are used to specify the name of an image file when loading an HTML document.

```html
<img src="computer.gif">
```
An attribute is used to describe something about the data itself. For example, in the following XML the attribute `use` might tell a program handling the data that the file is not required.

```xml
<file use="optional">computer.gif</file>
```

In XML, attribute values must always be enclosed in quotation marks, either single or double.

- XML allows data to be stored in either elements or attributes.
- Elements and attributes can be named to give the data meaning.
- Start tags and end tags define elements that are the basis for XML tree-structured representations of documents.
- Elements can contain text data and/or other elements

**Entities**

Entities are used to substitute one string for another in an XML document. For example, if a phrase such as "XML and the Data Revolution" is repeated frequently in a document, one can define a shortcut entity declaration in the DTD.

```xml
<!ENTITY xdr " XML and the Data Revolution ">
```

Then, when you want to use the full phrase, you use `&xdr;` and it will be substituted in the XML document. Using entities can help avoid misspellings and the tediousness of typing the same thing over and over.

**Predefined Entities**

XML has adopted five predefined entities from the HTML world. The ampersand (`&`), greater-than (`>`), lesser-than (`<`), double-quote (`"`), and apostrophe (`'`) characters are represented within XML documents as `&amp;`, `"`, `&lt;`, `&gt;`, `&quot;`, `&apos;`, respectively.

If the entities are long, it's possible to store the information separately in another file. This can be accomplished through an external entity reference, which uses the XML keyword `SYSTEM` between the entity name and URL of the file.
<!ENTITY text SYSTEM "http://my.url.here">

Parameter Entities

While entities are useful for creating substitution strings within XML documents, it's often useful to define shortcuts in a DTD to make writing a DTD easier. This is where parameter entities come in. A parameter entity is defined by inserting a percent sign prior to the entity name. Once defined, a parameter entity can be substituted by surrounding the parameter name with a percent sign and semicolon.

**********

CDATA

The XML CDATA section is used to prevent the processing of a portion of data. When an XML document is parsed, all the XML is processed except the data inside a CDATA section. This allows the inclusion of content that may confuse an XML processor.

For example, if an XML document contains greater-than or ampersand characters, as many programs or scripts do, one can define a CDATA section to contain this data. A CDATA section starts with "<![CDATA[ " and ends with "]]>". The following example shows how a CDATA section may be used to include script code within an XML element named script.

<script>
<![CDATA[
function compare(a,b) {
if (a < b) then {
return 1
}
else {
return 0
}
}
]]>
</script>

**********
**Processing Instructions**

XML allows the use of special instructions in order to pass information to programs that may read the document. A processing instruction begins with " <? " and ends with " ?>". Immediately after the " <? " is a target name that is used to let a program know who the content of the processing instruction is intended for. For example, the following is a processing instruction intended for a program that is looking for the name agent.

```xml
<?agent process="yes" priority="high">
```

**XML Declaration**

Most XML documents begin an XML declaration of the form:

```xml
<?xml version="1.0">
```

Although the XML declaration looks like a processing instruction, it is technically not. If present, it must be the first thing in a document.

*********

**Create a Simple XML file**

```xml
<?xml version="1.0" encoding="utf-8" ?>
<books>
  <book>
    <name> Fundamental XML </name>
    <author> Surya </author>
    <level> Intro </level>
  </book>
  <book>
    <name> Java for Beginners </name>
    <author> Sathya </author>
    <level> Beginner </level>
  </book>
</books>
```

The first line is the XML declaration. It defines the XML version (1.0) and the encoding used (is utf-8 character set). The next line describes the root element of the document <books>. The next lines describe the child elements <book>, <author>, <level>
The XML Advantage

- **XML files are human-readable**. XML was designed as text so that someone can always read it to figure out the content. Such is not the case with binary data formats.
- **Widespread industry support exists for XML**. Numerous tools and utilities are being provided with Web browsers, databases, and operating systems, making it easier and less expensive for small and medium-sized organizations to import and export data in XML format.
- Major relational databases now have the native capability to read and generate XML data.
- A large family of XML support technologies is available for the interpretation and transformation of XML data for Web page display and report generation.

XML: Design by Omission

Much of XML's success stems from what it does not address

In addition to XML's advantages, XML's widespread success and use derives more from what it does not address.

1. **No display is assumed.** Unlike HTML, XML makes no assumptions about how tags will be rendered in a browser or other display device. Auxiliary technologies such as style sheets add this capability.
2. **There is no built-in data typing.** DTDs and XML Schema provide support for defining the structure and data types associated with an XML document.
3. **No transport is assumed.** The XML specification makes no assumption about how XML is to be transported across the Internet. This has opened the door to creative ideas about delivering XML by means of HTTP, FTP, or Simple Mail Transfer Protocol (SMTP).
XML and the Web

XML integrates with standard Web protocols such as HTTP and FTP.

Figure shows how XML may be used to communicate directly with partners and suppliers. Instead of exchanging data about purchases and orders either manually or over proprietary networks, data vocabularies can be defined using XML and delivered from server to server using standard protocols such as HTTP or FTP.

Messaging middleware supports the asynchronous delivery of XML. Associated with this ability to move data freely across the Web is the rise in the use of messaging servers and software that sit between conversational participants. These servers, supporting what is known as Message Oriented Middleware, are playing an important role in the new extended enterprise by providing guarantees of delivery and the ability to broadcast communications to multiple recipients.

Web services makes B2B direct connections feasible over the Web. A third aspect of the new extended enterprise is the emergence of Web services. Web services represent the next evolutionary step for the Web, extending it from a network that provides services to humans to one that provides services to software looking to connect with other software.
SOAP

SOAP is an XML based protocol.

**SOAP is the XML glue that lets clients and providers talk to each other and exchange XML data.** As Figure shows, SOAP builds on XML and common Web protocols (HTTP, FTP, and SMTP) to enable communication across the Web. *SOAP brings to the table a set of rules for moving data, either directly in a point-to-point fashion or by sending the data through a message queue intermediary.*

SOAP, an XML-based protocol, gains its global scope through the power of combination with Internet protocols such as HTTP, FTP, and SMTP.

**SOAP opens up new options for distributed computing.**

One of the main implications of SOAP is a change in how we think about distributed computing. Prior to SOAP, there were three basic options for doing distributed computing: Microsoft's Distributed Component Object Model (DCOM), Java's Remote Method Invocation (RMI), or the Object Management Group's Common Object Request Broker Architecture (CORBA). These technologies are still in. *Their drawback is that they limit the potential reach of the enterprise to servers that share the same object infrastructure. With SOAP, however, the potential space of interconnection is the entire Web itself, which is why there is such intense interest in technologies that can influence the power of SOAP. One of these technology efforts is web services.*
Web Services

Web services builds on a SOAP foundation.

Web services is both a process and set of protocols for finding and connecting to software exposed as services over the Web. By assuming a SOAP foundation, Web services can concentrate on what data to exchange instead of worrying about how to get it from point A to point B, which is the job of SOAP.

Although SOAP may be used with a variety of protocols, the only bindings specified in the proposed SOAP specification are for HTTP.

Web services facilitates software interaction.

A Web service can be anything from a movie review service, to a real-time weather advisory, to an entire hotel- and airline-booking package. The Web services technical infrastructure ensures that services even from different vendors will interoperate to create a complete business process.

Web services = repository + client + provider.

As Figure illustrates, the interconnections opened up by the Web make possible a new way of interacting through the registration, discovery, and connection of software packaged as Web services—there are three major aspects to Web Services:

- A service provider provides an interface for software that can carry out a specified set of tasks.
- A service requester discovers and invokes a software service to provide a business solution. The requester will commonly invoke a remote procedure call on the service provider, passing parameter data to the provider and receiving a result in reply.
- A broker manages and publishes the service. Service providers publish their services with the broker and requests access those services by creating bindings to the service provider.
Figure 1.5. The Web services framework provides protocols and a process for clients to discover and connect to web-based services.
XML: The Three Revolutions

The three revolutions: data, architecture, and software.

As Figure illustrates, the three areas of impact are

- **Data**: which XML frees from the confines of fixed, program-dependent formats;
- **Architecture**: with a change in emphasis from tightly coupled distributed systems to a more loosely coupled confederation based on the Web
- **Software**: with the realization that software evolution is a better path to managing complexity than building monolithic applications.
a) The Data Revolution
Data is now free to travel the Web. Prior to XML, data was very much proprietary, closely associated with applications that understood how data was formatted and how to process it. Now, XML-based industry-specific data vocabularies provide alternatives to specialized Electronic Data Interchange (EDI) solutions by facilitating B2B data exchange and playing a key role as a messaging infrastructure for distributed computing. XML enables the creation of program-independent data formats.

XML’s strength is its data independence. XML is pure data description, not tied to any programming language, operating system, or transport protocol. In the distributed computing this is a fundamental idea. The data is free to move about globally without the constraints imposed by tightly coupled transport-dependent architectures. XML’s sole focus on data means that a variety of transport technologies may be used to move XML across the Web. XML does this by focusing on data and leaving other issues to supporting technologies.

The Code, Data, Document Culture:

XML has emerged from a document culture. To understand XML’s impact on the computing world, it’s useful to place XML in perspective. As Figure 1.9 shows, XML comes out of a document culture that is distinct from the code and data cultures. The code culture is characterized by a focus on programming languages, beginning with FORTRAN and Algol, C, C++, and Java. The data culture
is characterized by COBOL, data processing, and databases. From a code perspective, data is something to be transported by procedure calls. From a data perspective, data is something to be stored in databases and manipulated.

Code and data have defined systems thinking. The late 1980s and early 1990s saw code and data combine in the form of object-oriented languages such as C++, Smalltalk, Java, and Object COBOL. And yet, object technology was only a partial answer.

XML opens up options for treating code as data. XML's emergence from the data-oriented document culture has forced a rethinking about application development. XML brings to the computing world is a technology that allows data to be freed from the constraints created by code-centric infrastructures. XML now permits data to stand on its own. It allows code to be treated as data. As 1.10 illustrates, XML offers an alternative to both EDI and technologies such as CORBA, RMI, and DCOM that lock data transfer into underlying networks and object infrastructures.

Figure 1.10. XML in combination with Web protocols allows data to be independent of network, programming language, or platform.

b) The Architectural Revolution

Simplicity and the ability to combine different standards are driving forces behind W3C deliberations. Together these XML-based technology initiatives open up new possibilities for distributed computing that influence the existing infrastructure of the Web and create a transition
from object-based distributed systems to architectures based on Web services that can be discovered, accessed, and assembled using open Web technologies. The focal point of this change in architectural thinking has been a move from tightly coupled systems based on established infrastructures such as CORBA, RMI, and DCOM, each with their own transport protocol, to loosely coupled systems riding atop standard Web protocols such as TCP/IP. Although the transport protocols underlying CORBA, RMI, and DCOM provide for efficient communication between nodes, their drawback is their inability to communicate with other tightly coupled systems or directly with the Web.

XML and the Web have enabled the loose coupling of software components. Loosely coupled Web-based systems, on the other hand, provide what has long been considered the universal connectivity. Using TCP/IP as the transport, systems can establish connections with each other using common open-Web protocols. Although it is possible to build software bridges linking tightly coupled systems with each other and the Web, such efforts are not trivial and add another layer of complexity on top of an already complex infrastructure. As Figure 1.11 shows, the loose coupling of the Web makes possible new system architectures built around message-based middleware or less structured peer-to-peer interaction.

![Figure 1.11. XML in combination with Web protocols has opened up new possibilities for distributed computing based on message passing as well as peer-to-peer interaction.](image-url)
c) The Software Revolution

XML is part of a software revolution centered around combination and surprise. XML is also part of a revolution in how we build software. During the 1970s and 1980s, software was constructed as monolithic applications built to solve specific problems. The problem with large software projects is that, by trying to tackle multiple problems at once, the software is often ill-suited to adding new functionality and adapting to technological change. In the 1990s a different model for software emerged based on the concept of simplicity. As Figure 1.12 illustrates, instead of trying to define all requirements up front, this new philosophy was built around the concept of creating building blocks capable of combination with other building blocks that either already existed or were yet to be created.

The Web is an example of the power of combination. Each of the contributing technologies focused on doing one thing well without inhibiting interconnection with other technologies. The essential idea was to maximize the possibility of interaction and watch systems grow. The result is the Web, a product of the confluence of forces that include the Internet, HTML, and HTTP.

The power of combination is finding its way not only into software construction but up the development chain to software specification and design. Rather than hoping to meet the needs of users, design is now more collaborative, bringing in stakeholders early to ensure maximum feedback and the benefits of collaborative thinking. Figure 1.14 illustrates how this collaborative model is used by the W3C, the Internet Engineering Task Force, and Sun in its Java Community Process.
Figure 1.14. Part of the software revolution includes collaboration on specification and design. Examples include the Internet Engineering Task Force, the W3C, and Sun's Java Community Process.

**Collaboration in Software Specification and Design**

- **Internet Engineering Task Force**
  - Formed in 1992
  - Charter: build an international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet's architecture and operation
  - Standards include those for Internet infrastructure including security, routing, and user services

- **World Wide Web Consortium**
  - Formed in 1994
  - Charter: lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability
  - Standards (published as W3C Recommendations) include XML, XSLT, CSS, HTML, X/TML, DOM, and XML namespaces

- **Java Community Process**
  - Formed in 1995
  - Charter: develop and revise Java technology specifications, reference implementations, and technology compatibility kits
  - Standards are based on Java Specification Requests which describe both proposed and final specifications for the Java platform. There are currently over 100 JSRs including standards for Java 2 Enterprise Edition and Java Server Pages

*******
**XML Technology Family**

XML derives its strength from a variety of supporting technologies. XML is not just a technology for defining data vocabularies. Surrounding XML is a wide variety of XML standards and initiatives that act in combination with XML to address many of the issues associated with bringing XML into mainstream computing, namely presentation, structure, and transformation. As Figure 2.1 shows, the XML core includes XML itself, based on the XML 1.0 specification, and namespaces, the specification that allows XML documents from different sources to be combined and yet be able to disambiguate elements with the same name from different sources. Following is a list of other categories in the XML technology:

**Structure and data types:**
When using XML to exchange data among clients, partners, and suppliers, it's important to be able to define how XML documents should be structured. **DTDs and XML Schema** provide this capability. 

*DTDs focusing primarily on structure by specifying what elements and attributes are considered valid for a particular XML instance document.* DTDs have limited capability to specify data types. **XML Schema** is a more recent initiative of the World Wide Web Consortium (W3C) that puts a more conventional data processing spin on describing XML data with more precision than with DTDs.

**XML presentation technologies:**
In keeping with an important design pattern for strong systems, XML purposely separates data content from presentation through the introduction of supporting technologies that focus on delivering content to users via a variety of devices and presentation media. Among the technologies for presentation are

- **XHTML**, a modular XML-conformant replacement for HTML;
- **CSS** for controlling the display properties of HTML or XML in Web browsers;
- **XSL** and **XSL Formatting Objects (XSL-FO)** for formatting XML for various output media; **XForms** for collecting data from Web forms and returning XML;
- **VoiceXML**, for delivering content to voice-enabled devices;
- **Wireless Markup Language (WML)**, for delivery to wireless devices enabled for Wireless Application Protocol (WAP).
XML manipulation technologies:
Manipulation technologies provide the capability to extract and transform XML in different ways. These technologies play an important role in server-based XML processing for business-to-business (B2B) data manipulation and exchange.

- **XSLT** is widely used to transform XML from one format to another;
- **XPath** is a technology used by other XML technologies to navigate through an XML tree structure and zero in on particular elements or subtrees;
- **XLink** is a technology for creating and describing links between resources and for enabling links that go beyond the simple unidirectional links of the current Web;
- **XQuery** is an evolving technology for extracting and querying XML repositories.

**Other related technologies:** The XML family of technologies also includes initiatives for working with metainformation, which is literally information about the information contained in an XML document. Technologies in this space include **RDF and InfoSet**.

The core of XML and its key components and extensions are:

- **XML 1.0 syntax**, including Document Type Definitions (DTDs)
- **Namespaces** in XML
- **XML Schema** (or one of its alternatives or supplemental validation tools: XDR, SOX, RELAX, TREX, and The Schematron)

Some of the key features that are commonly required in XML applications.
- Describing XML data structure: The XML Information Set (XML Infoset) and XML Path Language (XPath)
- Navigating & Linking: XML Linking (XLink), XML Pointer Language (XPointer), XML Inclusions (XInclude), XML Fragment Interchange (XFI), and XML Query Language (XQuery).
- Transforming & Presenting: XSLT and XSL-FO (XSL Formatting Objects)

******

**XML Namespaces**

Namespaces eliminates the ambiguity of the same name from different providers.

When developing XML documents, it is common to refer to element and attribute names that share a common context as vocabulary. The possibility arises that an element or attribute in that vocabulary may have a name that is identical to an element or attribute used by someone else in a different vocabulary.

XML namespaces is a simple technology solution that allows element and attribute names to be distinguished from the similarly named elements and attributes of other XML users. XML namespace solve this ambiguity problem by associating explicit namespace (or vocabulary) with elements and attributes in XML document. Thus a namespace is essentially a set of names in which all the elements and attribute names can be guaranteed to be unique.

Figure 2.3 illustrates how a namespace may be used to disambiguate duplicate element names. XML namespaces solves the problem of clashing names by providing for a unique prefix to be attached to the beginning of element and attribute names. In practice, a company's Web address is often used as the unique prefix, but technically the namespaces specification allows any Uniform Resource Identifier (URI) to be used. URIs are more general than the common Uniform Resource Locator (URL) and include just about any unique name one wants to use. The usual result, though, is that when namespaces are used in an XML document, the official element name is actually a two-part name: the name of the XML namespace plus the name of the element or attribute.
Namespaces are needed to distinguish between identical element names from different data sources

Namespace Declarations

There are several ways to add a namespace to an XML document so that, when software is processing XML data from ZwiftBooks, it will see ZwiftBooks elements as http://www.zwiftbooks.com:title instead of just title. The simplest approach is to declare a namespace in a top-level element and let all the elements and attributes under the top element come under the scope of the namespace. For example, the following XML document adds a ZwiftBooks namespace to an XML book description document.

```xml
<book xmlns="http://www.zwiftbooks.com">
  <isbn>0-596-00058-8</isbn>
  <title>XML in a Nutshell</title>
  <author>Harold, Elliotte Rusty</author>
</book>
```

In this example, the namespace declaration is applied to the book element by adding the predefined attribute xmlns and giving as its value the unique URL of ZwiftBooks, http://www.zwiftbooks.com. Because the xmlns attribute appears in the book element, all subelements (isbn, title, and author) are included in the namespace.

Namespace Abbreviations

Namespace abbreviations may be used to simplify writing and reading the XML. The namespaces specification also makes it possible to use abbreviations for namespaces in order to make XML documents more readable. Figure 2.5 shows how we can define a shortcut name, zbooks, so that
anywhere that zbooks appears in a document, a software program processing the document will replace it with the actual namespace, http://www.zwiftbooks.com.

Figure 2.5. An XML document using a namespace abbreviation.

The shortcut term zbooks now refers to the namespace http://www.zwiftbooks.com

    <zbooks:isbn>0-596-00058-8</zbooks:isbn>
    <zbooks:title>XML in a Nutshell</zbooks:title>
    <zbooks:author>Harold, Elliotte Rusty</zbooks:author>
</zbooks:book>

All elements that begin with zbooks belong to the http://www.zwiftbooks.com namespace

*****
UNIT-I-PART-A QUESTIONS

1. Define XML.

- Extensible Markup Language (XML) is a simple data description language. XML is a metalanguage (literally a language about languages) defined by the World Wide Web Consortium (W3C). XML is a set of rules and guidelines for describing structured data in plain text rather than proprietary binary representations.
- XML is a software- and hardware-independent tool for storing and transporting data, allowing designers to describe the content rather than the format of the text they enclose.

2. What is an XML element? Give example.
An XML element is made up of a start tag, an end tag, and data in between. The start and end tags describe the data within the tags, which is considered the value of the element. Example:

```
<book> The complete reference Java 2 </book>
```

Start tags such as `<Name>` begin an element that contains the actual data. End tags such as `</Name>` mark the end of an element definition.

3. Explain the various types of elements in XML

There are three ways elements are used in XML documents.
Simple Content

Text or other data appears between start and end tags. The start tag has the same name as the end tag except that the end tag begins with a slash. The following element has a start tag, content, and an end tag.

<author> DR.E.BALAGURUSAMY </author>

Element as Container for Other Elements

An element may contain other elements, providing a hierarchical or tree data structure. The following book element contains the author and title elements.

<book>
<author> DR.E.BALAGURUSAMY </author>
<title> XML AND WEB SERVICES </title>
</book>

Empty Element as Container for Attributes

When an element has no content but only attributes, there is a shorthand way of writing the element that bypasses the need for both a start and end tag. An element written with a slash following the tag name indicates an empty element, as in

<book/> which is shorthand for <book></book>

The most common use of empty elements is to hold attribute data.

<book title="XML AND WEB SERVICES" author="DR.E.BALAGURUSAMY"/>


Attributes provide additional information about elements.

An element can optionally contain one or more attributes. An attribute is a name-value pair separated by an equal sign (=). Example:

<CITY PIN="600025">Chennai</CITY>
It's also possible to represent the data using attributes within a single element:

<Customer name="RAMESH" phone="9442121215"
fax="044-45678345" email="ramesh@gmail.com ">
</Customer>

5. What are XML entities?

Entities are used to substitute one string for another in an XML document. For example, if a phrase such as "XML and the Data Revolution" is repeated frequently in a document, one can define a shortcut entity declaration in the DTD.

<!ENTITY xdr " XML and the Data Revolution ">

Then, when you want to use the full phrase, you use &xdr; and it will be substituted in the XML document. Using entities can help avoid misspellings and the tediousness of typing the same thing over and over.

6. What are predefined entities?

XML has adopted five predefined entities from the HTML world. The ampersand ( &), greater-than ( >), lesser-than ( <), double-quote ( " ), and apostrophe ( ' ) characters are represented within XML documents as " &amp; ", " &quot; , and " &apos; ", respectively.

7. State the advantages of XML

• XML files are human-readable. XML was designed as text so that, in the worst case, someone can always read it to figure out the content. Such is not the case with binary data formats.
• Widespread industry support exists for XML. Numerous tools and utilities are being provided with Web browsers, databases, and operating systems, making it easier and less expensive for small and medium-sized organizations to import and export data in XML format.
• Major relational databases now have the native capability to read and generate XML data.
• A large family of XML support technologies is available for the interpretation and transformation of XML data for Web page display and report generation.
8. What is a web service?

A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response.

A Web service is a software service used to communicate between two devices on a network. Web service is a software application with a standardized way of providing interoperability between disparate applications. **Examples**

- A movie review service
- A real-time weather advisory
- Entire hotel- and airline-booking package.

9. Define SOAP

SOAP is an XML based protocol.

SOAP is the XML glue that lets clients and providers talk to each other and exchange XML data. SOAP builds on XML and common Web protocols (HTTP, FTP, and SMTP) to enable communication across the Web. SOAP brings to the table a set of rules for moving data, either directly in a point-to-point fashion or by sending the data through a message queue intermediary.

10. State the Three XML revolutions

**Data**: which XML frees from the confines of fixed, program-dependent formats;

**Architecture**: with a change in emphasis from tightly coupled distributed systems to a more loosely coupled confederation based on the Web; and

**Software**: with the realization that software evolution is a better path to managing complexity than building monolithic applications.

11. What are the uses of DTD and XML Schema?
Structure and data types: When using XML to exchange data among clients, partners, and suppliers, it's important to be able to define how XML documents should be structured. DTDs and XML Schema provide this capability.

12. What are the presentation technologies available in XML?

XML presentation technologies

- XHTML, a modular XML-conformant replacement for HTML;
- CSS for controlling the display properties of HTML or XML in Web browsers;
- XSL and XSL Formatting Objects (XSL-FO) for formatting XML for various output media; XForms for collecting data from Web forms and returning XML;
- VoiceXML, for delivering content to voice-enabled devices; and

13. Write a note on XML Manipulation technologies

Manipulation technologies provide the capability to extract and transform XML in different ways.

- XSLT is widely used to transform XML from one format to another
- XPath is a technology used by other XML technologies to navigate through an XML tree structure and zero in on particular elements or subtrees
- XLink is a technology for creating and describing links between resources and for enabling links that go beyond the simple unidirectional links of the current Web
- XQuery is an evolving technology for extracting and querying XML repositories.

14. Define XML Namespaces

XML Namespace is a mechanism to avoid name conflicts by differentiating elements or attributes within an XML document that may have identical names, but different definitions.
PART- B QUESTIONS

1. Explain the three revolutions of XML

2. Explain the importance of namespaces in XML

3. Explain the various types of elements and attributes in XML

4. Create an XML document for a vegetarian hotel which provides breakfast, lunch and Tiffin.

5. Briefly discuss about : SOAP , web services