4.4 XML (Extensible Markup Language)

XML is a text-based markup language derived from Standard Generalized Markup Language (SGML).It is a new markup language, developed by the W3C (World Wide Web Consortium), mainly to overcome the limitations in HTML. **Markup** is information added to a document that enhances its meaning in certain ways, in that it identifies the parts and how they relate to each other. More specifically, a markup language is a set of symbols that can be placed in the text of a document to demarcate and label the parts of that document.

XML is a markup language that defines set of rules for encoding documents in a format that is both human-readable and machine-readable.

XML is not a programming language. It is usually stored in a simple text file and is processed by special software that is capable of interpreting XML. XML don't have pre - defined tags and the tags are stricter than HTML.

XML is extensible: XML allows the user to create user defined self-descriptive tags, or language that suits the application.

XML carries the data, does not present it: XML allows storing the data irrespective of how it will be presented.

XML is a public standard: XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

Features of XML

- XML simplifies the creation of HTML documents for large web sites.
- XML can be used to exchange the information between organizations and systems.
- XML can be used for offloading and reloading of databases.
- XML can be used to store and arrange the data, which can customize the user data handling needs.
- XML can easily be merged with style sheets to create almost any desired output.
- Any type of data can be expressed as an XML document
- It has syndicated content, where content is being made available to different web sites.
- It suits well for electronic commerce applications where different organizations collaborate to serve a customer.
- It supports scientific applications with new markup languages for mathematical and chemical formulas.
- It also supports electronic books with new markup languages to express rights and ownership.
- XML could also be used in handheld devices and smart phones with new markup languages optimized for these devices.

Syntax Rules

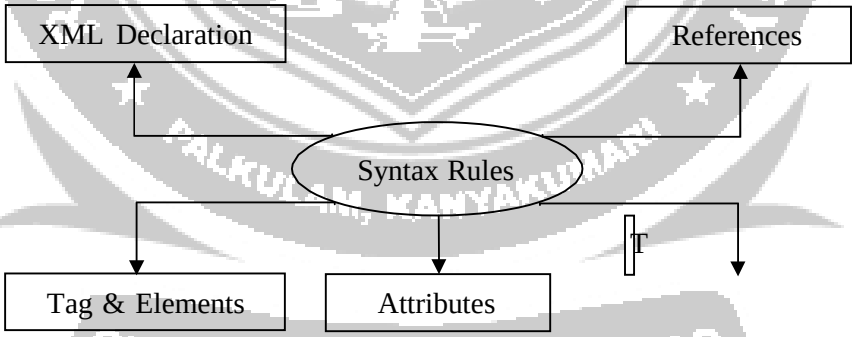

Fig 4.1 Syntax Rules of XML

The XML document can optionally have an XML declaration.

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

Version is the XML version and encoding specifies the character encoding used in the document.

Syntax Rules for XML declaration

- The XML declaration is case sensitive and must begin with "<?xml>" where "xml" is written in lower-case.
- If document contains XML declaration, then it strictly needs to be the first statement of the XML document.
- The XML declaration strictly needs be the first statement in the XML document.
- An HTTP protocol can override the value of encoding that the user put in the XML declaration.

➢ **Tags and Elements**

An XML file is structured by several XML-elements also called **XML-nodes** or XML-tags. XML-elements' names are enclosed by angular brackets <>.

<element>... </element> or

<element/>

Nesting of elements

An XML-element can contain multiple XML-elements as its children, but the children elements must not overlap. i.e., an end tag of an element must have the same name as that of the most recent unmatched start tag.

```
<?xml version="1.0"?>
<contact-info>
<company>abc</company>
<contact-info>
```

- **Root element:** An XML document can have only one root element.
- **Case sensitivity:** The names of XML-elements are case-sensitive.

➢ **Attributes**

An attribute specifies a single property for the element, using a name/value pair. An XML-element can have one or more attributes. It is possible to attach additional information to elements in the form of attributes. The names follow the same rules as element names.

<tel preferred=”true”>513-555-8889</tel>

- **XML References**

  References allow the user to add or include additional text or markup in an XML document. References always begin with the symbol "&", which is a reserved character and end with the symbol ";". XML has two types of references:

  - **Entity References:** An entity reference contains a name between the start and the end delimiters. The entity reference is replaced by the content of the entity.
  - **Character References:** A letter is replaced by its Unicode character code. Character references that start with &#x provides a hexadecimal representation of the character code.

XML Text

The names of XML-elements and XML-attributes are case-sensitive, which means the name of start and end elements need to be written in the same case. To avoid character encoding problems, all XML files should be saved as Unicode UTF-8 or UTF-16 files. Whitespace characters like blanks, tabs and line-breaks between XML-elements and between the XML-attributes will be ignored. Some characters are reserved by the XML syntax itself. Hence, they cannot be used directly.

XML Documents

An XML document is a basic unit of XML information composed of elements and other markup in an orderly package. An XML document can contains wide variety of data.

```
<?xml version="1.0"?> //document prolog
<contact-info> //all the following lines are document element
<name>Sharanya</name>
<company>abc</company>
<phone>(044)257887296</phone>
</contact-info>
```

The XML documents are divided into: Document prolog section and document element sections.

Document prolog: The document prolog comes at the top of the document, before the root element. This section contains: XML declaration and Document type declaration. The first line in the above example belongs to prolog section.

Document element: Document Elements are the building blocks of XML. These divide the document into a hierarchy of sections, each serving a specific purpose. The other lines except the first line come under document element section.

XML Declaration

The XML declaration is the first line of the document. The declaration identifies the document as an XML document. The declaration also lists the version of XML used in the document. The declaration can contain other attributes to support other features such as character set encoding.

Syntax:

```
<?xml
  version="version_number"
  encoding="encoding_declaration"
  standalone="standalone_status"
?>
```

- **Version**- Specifies the version of the XML standard used.
- **Encoding declaration**- It defines the character encoding used in the document. UTF-8 is the default encoding used.
- **Standalone**- It informs the parser whether the document relies on the information from an external source, such as external document type definition (DTD), for its content. The default value is set to no. Setting it to yes tells the processor there are no external declarations required for parsing the document.

Rules for XML declaration

- If the XML declaration is present in the XML, it must be placed as the first line in the XML document.
- If the XML declaration is included, it must contain version number attribute.
- The Parameter names and values are case-sensitive.
- The names are always in lower case.
- The order of placing the parameters is important. The correct order is: version, encoding and standalone.
- Either single or double quotes may be used.
- The XML declaration has no closing tag i.e. </?xml>

Example	Description
<?xml >	XML declaration with no parameters.
<?xml version="1.0">	XML declaration with version definition.

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>	XML declaration with all parameters defined.
<?xml version='1.0' encoding='iso-8859-1' standalone='no' ?>	XML declaration with all parameters defined in single quotes.

XML tags

XML tags form the foundation of XML. They define the scope of an element in the XML. They can also be used to insert comments, declare settings required for parsing the environment and to insert special instructions.

- **Start Tag:** The beginning of every non-empty XML element is marked by a start-tag.

  Example: <address>.

- **End Tag:**Every element that has a start tag should end with an end-tag.

  Example: </address>

- **Empty Tag:** The text that appears between start-tag and end-tag is called content. An element which has no content is termed as empty. An empty element can be represented in two ways:

  (1) A start-tag immediately followed by an end-tag:<hr></hr>

  (2) A complete empty-element tag : <hr />

XML Tags Rules

- XML tags are case-sensitive.
- XML tags must be closed in an appropriate order, i.e., an XML tag opened inside another element must be closed before the outer element is closed.

XML Elements

XML elements can be defined as building blocks of an XML. Each XML document contains one or more elements, the scope of which are either delimited by start and end tags, or for empty elements, by an empty-element tag.

<element-name attribute1 attribute2>

....content

</element-name>

- **element-name** is the name of the element.

- attribute1, attribute2 are **attributes** of the element separated by white spaces. An attribute defines a property of the element. It associates a name with a value, which is a string of characters. An attribute is written as: name = "value".

Empty Element: An empty element is an element with no content.

Example: <name attribute1 attribute2.../>

XML Elements Rules

- An element name can contain any alphanumeric characters. The only punctuation mark allowed in names are the hyphen (-), under-score (_) and period (.).
- Names are case sensitive.
- Start and end tags of an element must be identical.
- An element, which is a container, can contain text or elements.

XML Attributes

Attributes are part of the XML elements. An element can have multiple unique attributes. Attribute gives more information about XML elements. To be more precise, they define properties of elements. An XML attribute is always a name-value pair.

<element-name attribute1 attribute2 >

....content….

< /element-name>

where attribute1 and attribute2 has the following form: name = "value"

The following are the types of attributes:

- **String:** It takes any literal string as a value. CDATA is a String type. CDATA is character data. This means, any string of non-markup characters is a legal part of the attribute.
- **Tokenized:** This is more constrained type. The validity constraints noted in the grammar are applied after the attribute value is normalized.
- **Enumerated**: This has a list of predefined values in its declaration, out of which, it must assign one value. There are two types of enumerated attribute:
  - **NotationType:** It declares that an element will be referenced to a NOTATION declared somewhere else in the XML document.
  - **Enumeration:** Enumeration allows the user to define a specific list of values that the attribute value must match.

Element Attribute Rules

An attribute name must not appear more than once in the same start-tag or empty-element tag. An attribute must be declared in the Document Type Definition (DTD) using an Attribute-List Declaration. Attribute values must not contain direct or indirect entity references to external entities. The replacement text of any entity referred to directly or indirectly in an attribute value must not contain either less than sign <.

XML other features

- **Comments:**

  A comment starts with <!-- and ends with -->. Comments cannot appear before XML declaration. Comments can appear anywhere in a document. Comments must not appear within attribute values. Nested comments are not allowed.

- **Whitespaces:**

  Whitespace is a collection of spaces, tabs, and newlines. XML document contain two types of white spaces Significant Whitespace and Insignificant Whitespace. A **significant Whitespace** occurs within the element which contain text and markup present together.

  <name>Adhithya Ramanan</name>

Insignificant whitespace means the space where only element content is allowed.

<address.category="residence">

Differences between XML and HTML

XML	HTML
XML was designed to be a software and hardware independent tool used to transport and store data, with focus on what data is.	HTML was designed to display data with focus on how data looks.
XML provides a framework for defining markup languages.	HTML is a markup language itself.
XML is neither a programming language nor a presentation language.	HTML is a presentation language.
XML is case sensitive.	HTML is case insensitive.
XML is used basically to transport data between the application and the database.	HTML is used for designing a web-page to be rendered on the client side.
In XML custom tags can be defined and the tags are invented by the author of the XML document.	HTML has its own predefined tags

XML makes it mandatory for the user the close each tag that has been used.	HTML is not strict if the user does not use the closing tags.
XML preserve white space.	HTML does not preserve white space.
XML is about carrying information, hence it is dynamic.	HTML is about displaying data, hence it is static.

4.5 DOCUMENT TYPE DECLARATION (DTD) and XML SCHEMAS

The document type declaration attaches a DTD to a document. It is a way to describe XML language precisely.

<!DOCTYPE element DTD identifier

[declaration1

declaration2

.....]>

The DTD starts with <!DOCTYPE delimiter. An element tells the parser to parse the document from the specified root element. DTD identifier is an identifier for the document type definition, which may be the path to a file on the system or URL to a file on the internet. If the DTD is pointing to external path, it is called **External Subset.** The square brackets [ ] enclose an optional list of entity declarations called **InternalSubset.**

Internal DTD

A DTD is referred to as an internal DTD if elements are declared within the XML files. To refer it as internal DTD, standalone attribute in XML declaration must be set to yes.

<!DOCTYPE root-element [element-declarations]>

root-element is the name of root element and element-declarations is where the user declare the elements.

Internal DTD

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE address [
<!ELEMENT address (name, company, phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>]>
<address> <name>Sharanya</name>
<company>abc</company>
<phone>12347890</phone> </address>
```

Start Declaration- Begin the XML declaration with following statement

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

DTD- Immediately after the XML header, the document type declaration follows, commonly referred to as the DOCTYPE:

<!DOCTYPE address [

The DOCTYPE declaration has an exclamation mark (!) at the start of the element name. The DOCTYPE informs the parser that a DTD is associated with this XML document.

DTD Body- The DOCTYPE declaration is followed by body of the DTD, where elements, attributes, entities, and notations are declared.

End Declaration - Finally, the declaration section of the DTD is closed using a closing bracket and a closing angle bracket (]>).

Rules

- The document type declaration must appear at the start of the document.
- The element declarations must start with an exclamation mark.
- The Name in the document type declaration must match the element type of the root element.

External DTD

In external DTD elements are declared outside the XML file. They are accessed by specifying the system attributes which may be either the legal .dtd file or a valid URL. To refer it as external DTD, standalone attribute in the XML declaration must be set as no. This means, declaration includes information from the external source.

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

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<!DOCTYPE address SYSTEM "address.dtd">

<address> <name>Sharanya</name>

<company>abc</company>

<phone>1234689</phone> </address>

address.dtd

<!ELEMENT address (name,company,phone)> <!ELEMENT name (#PCDATA)>

<!ELEMENT company (#PCDATA)> <!ELEMENT phone (#PCDATA)>

Types of external DTD

- **System Identifiers:** A system identifier enables the user to specify the location of an external file containing DTD declarations.

```
<!DOCTYPE name SYSTEM "address.dtd" [...]>
```

- **Public Identifiers:** Public identifiers provide a mechanism to locate DTD resources.

```
<!DOCTYPE name PUBLIC "-//Beginning XML//DTD Address Example//EN">
```

Public identifiers are used to identify an entry in a catalog. Public identifiers can follow any format, however, a commonly used format is called **Formal Public Identifiers, or FPIs**.

Advantages of DTD

- The XML processor enforces the structure, as defined in the DTD.
- The application easily accesses the document structure.
- The DTD gives hints to the XML processor—that is, it helps separate indenting from content.
- The DTD can declare default or fixed values for attributes. This might result in a smaller document.

XML SCHEMAS

XML Schema is commonly known as XML Schema Definition (XSD). It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="contact">
<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> </xs:schema>
```

- **Elements:** An element can be defined within an XSD as follows:

```
<xs:element name="x" type="y"/>
```

- **Definition Types**

  - **Simple Type** - Simple type element is used only in the context of the text. Some of predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date.

    Example: <xs:element name="phone_number" type="xs:int" />

  - **Complex Type** - A complex type is a container for other element definitions. This allows the user to specify which child elements an element can contain and to provide some structure within the user's XML documents.

```
<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>
```

  - **Global Types** - With global type, the user can define a single type in the user's document, which can be used by all other references.

```
<xs:element  name="AddressType">
<xs:complexType><xs:sequence>
<xs:element  name="name"  type="xs:string"  />
<xs:element name="company" type="xs:string" />
</xs:sequence></xs:complexType>
</xs:element>
```

Now let us use this type in our example as below:

```
<xs:element  name="Address1">
<xs:complexType><xs:sequence>
<xs:element  name="address"  type="AddressType"  />
<xs:element  name="phone1" type="xs:int" />
</xs:sequence></xs:complexType>
```

```
</xs:element>
<xs:element name="Address2">
<xs:complexType><xs:sequence>
<xs:element name="address" type="AddressType" />
  <xs:element name="phone2" type="xs:int" />
</xs:sequence></xs:complexType>
</xs:element>
```

Instead of having to define the name and the company twice (once for Address1 and once for Address2), we now have a single definition.

- **Attributes**

Attributes in XSD provide extra information within an element. Attributes have name and type property as shown below:

```
<xs:attribute name="x" type="y"/>
```