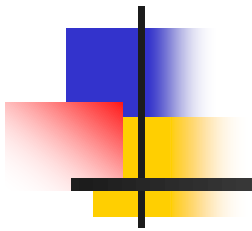


Module 1

Introduction to XML

Module Overview

In this module, you will learn about:

- Introduction to XML
- Exploring XML
- Working with XML
- XML Syntax

Lesson 1 – Introduction to XML

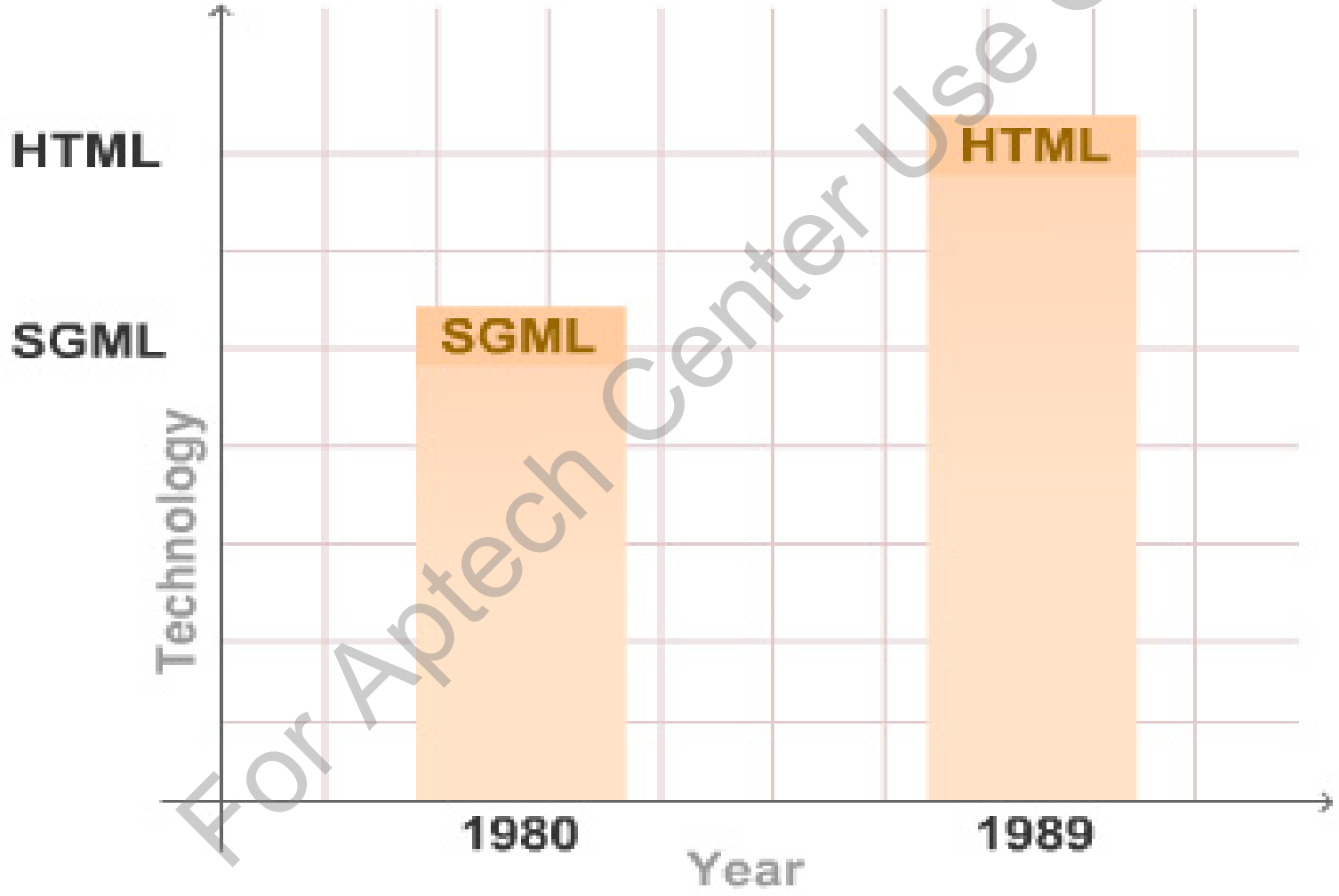
In this first lesson, **Introduction to XML**, you will learn to:

- Outline the features of markup languages and list their drawbacks.
- Define and describe XML.
- State the benefits and scope of XML.

Features of Markup Languages 1-3

- Generalized Markup Language (GML) helps the documents to be edited, formatted, and searched by different programs using its content-based tags.
- Standard Generalized Markup Language (SGML) is a coding scheme for developing specialized markup languages.
- Hyper Text Markup Language (HTML) is used for sharing information by using hyperlinked text documents.

Features of Markup Languages 2-3



Features of Markup Languages 3-3

Features

- GML describes the document in terms of its format, structure and other properties.
- SGML ensures that system can represent the data in its own way.
- HTML used ASCII text, which allows the user to use any text editor.

Drawbacks

- GML and SGML were not suited for data interchange over the web.
- HTML instructions is used to display content rather than content they encompass.

Evolution of XML 1-2

- XML is a W3C recommendation.
- It is a set of rules for defining semantic tags that break a document into parts.
- XML was developed over HTML.

Evolution of XML 2-2

HTML	XML
HTML was designed to display data.	XML was designed to carry data.
HTML displays data and focuses on how data looks.	XML describes data and focuses on what data is.
HTML displays information.	XML describes information.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
- <Watch>
    <name>Titan</name>
    <price>$50</price>
    <description>A brown diamond studded watch</description>
  </Watch>
```

Features of XML

- XML stands for Extensible Markup Language
- XML is a markup language much like HTML
- XML was designed to describe data
- XML tags are not predefined
- XML uses a Document Type Definition (DTD) or an XML Schema to describe data

XML Markup 1-2

- XML markup defines the physical and logical layout of the document.
- XML markup divides a document into separate information containers called elements.
- A document consists of one outermost element, called `root` element that contains all the other elements, plus some optional administrative information at the top, known as XML declaration.

XML Markup 2-2

Code Snippet

```
<?xml version=”1.0” encoding=”iso-8859-1” ?>
-  <FlowerPlanet>
 <Name>Rose</Name>
  <Price>$1</Price>
  <Description>Red in color</Description>
  <Number>700</Number>
</FlowerPlanet>
```

where,

`<Name>`, `<Price>`,`<Description>` and `<Number>` tags are elements

`<FlowerPlanet>` and `</FlowerPlanet>` are the root elements

Benefits of XML

- Data independence - separates the content from its presentation
- Easier to parse - absence of formatting instructions makes it easy to parse
- Reducing Server Load - semantic and structural information enables it to be manipulated by any application, can now be performed by clients
- Easier to create – can easily create with the most primitive text processing tools
- Web Site Content – can be transformed into a number of other formats
- Remote Procedure Calls – protocol that allows objects on one computer to call objects on another computer
- E-Commerce - can be used as an exchange format

Lesson 2 – Exploring XML

In this second lesson, **Exploring XML**, you will learn to:

- Describe the structure of an XML document.
- Explain the lifecycle of an XML document.
- State the functions of editors for XML and list the popularly used editors.
- State the functions of parsers for XML and list names of commonly used parsers.
- State the functions of browsers for XML and list the commonly used browsers.

XML Document Structure 1-4

- The two sections of an XML document are:
  - Document Prolog
  - Root Element

- An XML document consists of a set of unambiguously named "entities".
- Every document starts with a "root" or document entity. All other entities are optional.
- A single entity name can represent a large amount of text. The alias name is used each time some text is referenced and the processor expands the contents of the alias.

XML Document Structure 2-4

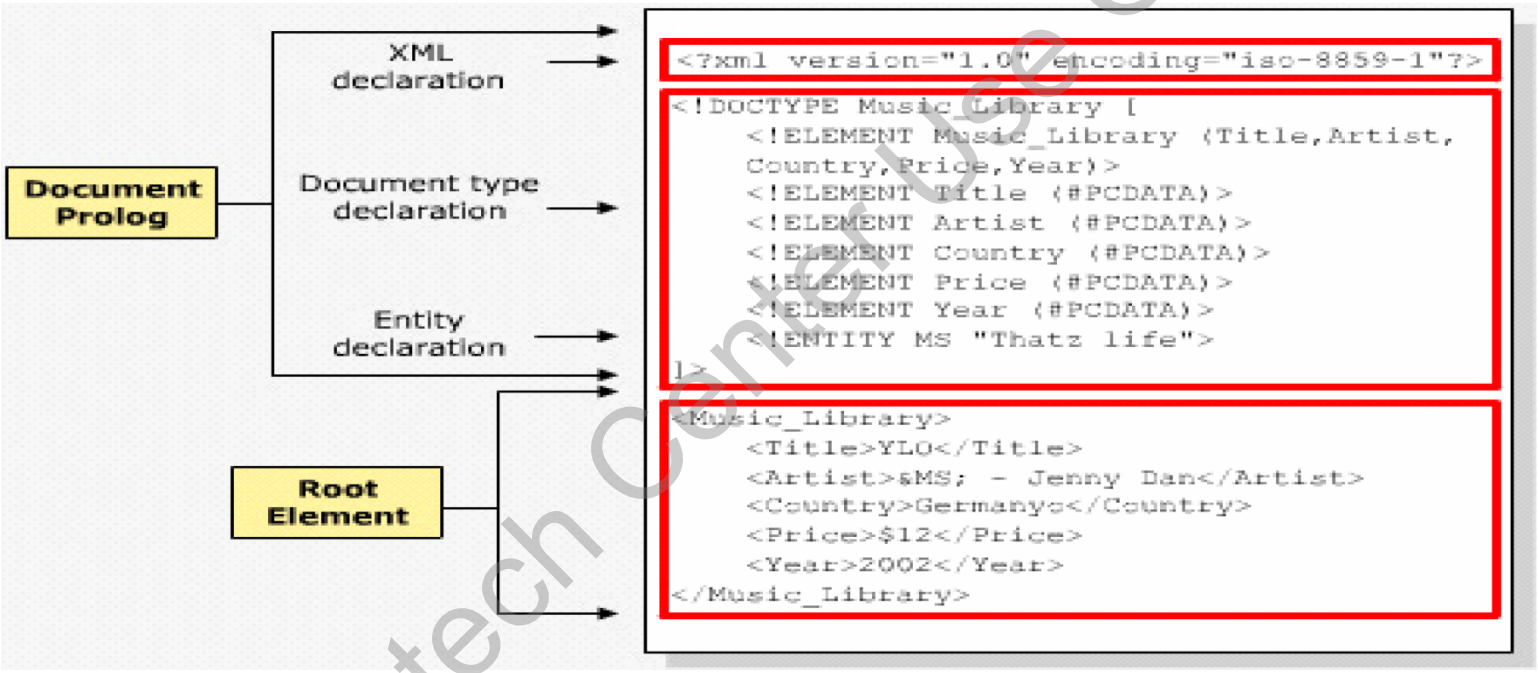
Document Prolog

- Document prolog contains metadata and consists of two parts - XML Declaration and Document Type Declaration.
- XML Declaration specifies the version of XML being used.
- Document Type Declaration defines entities' or attributes' values and checks grammar and vocabulary of markup.

Root Element

- Also called a document element.
- It must contain all the other elements and content in the document.
- An XML element has a start tag and end tag.

XML Document Structure 3-4



XML Document Structure 4-4

Code Snippet

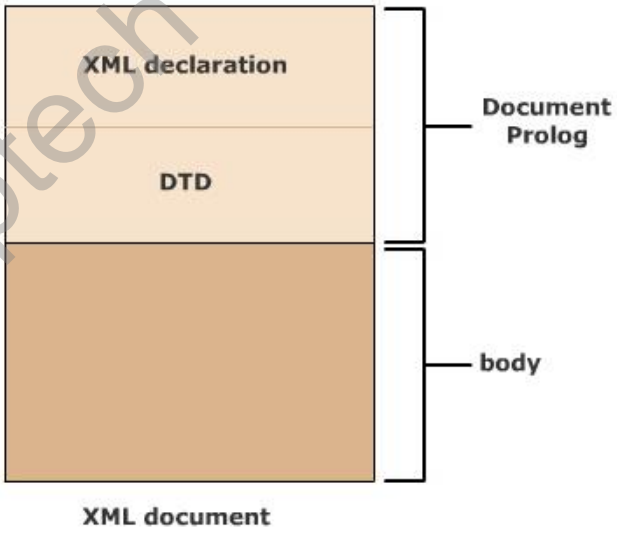
```
<?xml version="1.0" encoding="iso-8859-1"?>
 <!DOCTYPE Music_Library [
     <!ELEMENT Music_Library (Title,Artist,Country,Price,Year)>
     <!ELEMENT Title (#PCDATA)>
     <!ELEMENT Artist (#PCDATA)>
     <!ELEMENT Country (#PCDATA)>
     <!ELEMENT Price (#PCDATA)>
     <!ELEMENT Year (#PCDATA)>
     <!ENTITY MS "Thatz life">
 ]>
 <Music_Library>
    <Title>YLO</Title>
    <Artist>&MS; - Jenny Dan</Artist>
    <Country>Germanyo</Country>
    <Price>$12</Price>
    <Year>2002</Year>
</Music_Library>
```

where,

The first block indicates xml declaration and document type declaration. `Music_Library` is the root element.

Logical Structure 1-2

- The logical structure gives information about the elements and the order in which they are to be included in the document.
- It shows how a document is constructed rather than what it contains.
- Document Prolog forms the basis of the logical structure of the XML document.
- XML Declaration and Document Type Definition are its components.



Logical Structure 2-2

Code Snippet

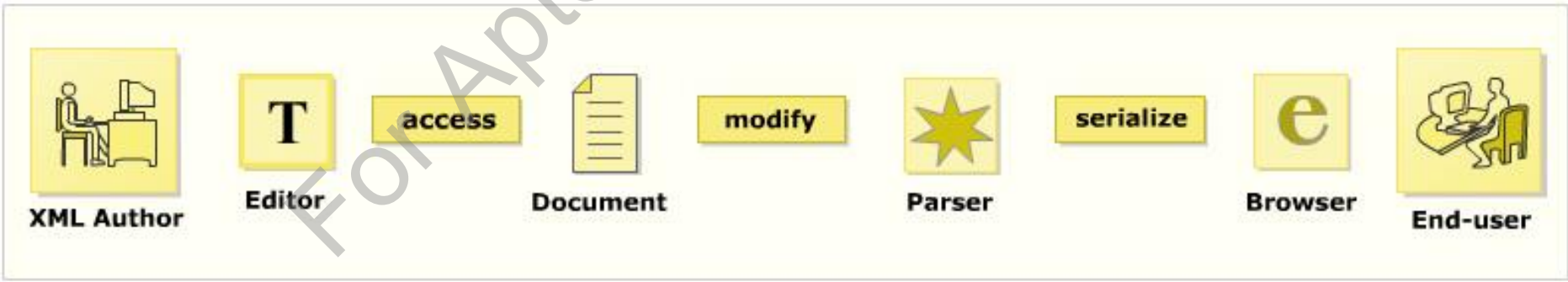
```
<?xml version=”1.0” encoding=”iso-8859-1” ?>
```

Code Snippet

```
<!DOCTYPE Music_Library SYSTEM
“Mlibrary.dtd”>
```

XML Document Life Cycle

1. XML document creation
2. Scanning
3. Parsing
4. Access
5. Conversion into Application program
6. Modification
7. Serialization



Editors

Main Functions:

- Add opening and closing tags to the code
- Check for validity of XML
- Verify XML against a DTD/Schema
- Perform series of transforms over a document
- Color the XML syntax
- Display the line numbers
- Present the content and hide the code
- Complete the word

Some popularly used editors are:

- XMLwriter
- XML Spy
- XML Pro
- XMLmind
- XMetal

Parsers 1-2

- **An XML parser:**
  - Reads the document
  - Verifies it for its well-formedness
  - After verification, converts it into a tree of elements
- **Commonly used parsers:**
  - Crimson, Xerces, Oracle XML Parser, JAXP, MSXML
- **Type of parsers:**
  - Non Validating parser
  - Validating parser

Parsers 2-2

Non Validating parser

- It checks the well-formedness of the document.
- Reads the document and checks for its conformity with XML standards.

Validating parser

- It checks the validity of the document using DTD.

Browsers

- Web browsers can format XML data and display it to the user.
- Other programs like database, Musical Instrument Digital Interface (MIDI) program or a spreadsheet program may present XML data accordingly.

- Commonly used web browsers:
  - Netscape, Mozilla, Internet Explorer, Firefox, Opera

Lesson 3 – Working with XML

In this third lesson, **Working with XML**, you will learn to:

- Explain the steps towards building an XML document.
- Define what is meant by well-formed XML.

Creating an XML document

- An XML document has three main components:
  - Tags(markup) and text(content)
  - DTD or Schema
  - Formatting or display specifications

- The steps to build an XML document are as follows:
  1. Create an XML document in an editor
  2. Save the XML document
  3. Load the XML document in a browser

Exploring the XML document 1-3

- Various building blocks of an XML document:
  - XML Version Declaration
  - Document Type Definition
  - Document instance in which the content is defined by the mark up

Exploring the XML document 2-3

```
1  <?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
2
3  <!DOCTYPE student [
4  <!ELEMENT student (name,dob,bloodgroup,rollnumber)>
5  <!ELEMENT name (#PCDATA)>
6  <!ELEMENT dob (#PCDATA)>
7  <!ELEMENT bloodgroup (#PCDATA)>
8  <!ELEMENT rollnumber (#PCDATA)>
9  ]>
10
11 <student>
12      <name>James</name>
13      <dob>26th September, 1983</dob>
14      <bloodgroup>A positive</bloodgroup>
15      <rollnumber>17</rollnumber>
16 </student>
```

Characters are encoded using various encoding formats. The character encoding is declared in encoding declaration.

XML declaration tells the version of XML, the type of character encoding being used and the markup declaration used in the XML document.

XML declaration should start with the five character string, <?xml. It indicates that the document is an XML document.

Exploring the XML document 3-3

```
1  <?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
2
3  <!DOCTYPE student [
4  <!ELEMENT student (name,dob,bloodgroup,rollnumber)>
5  <!ELEMENT name (#PCDATA)>
6  <!ELEMENT dob (#PCDATA)>
7  <!ELEMENT bloodgroup (#PCDATA)>
8  <!ELEMENT rollnumber (#PCDATA)>
9  ]>
10
11 <student>
12      <name>James</name>
13      <dob>26th September, 1983</dob>
14      <bloodgroup>A positive</bloodgroup>
15      <rollnumber>17</rollnumber>
16 </student>
```

Indicates the presence of external markup declarations. "Yes" indicates that there are no external markup declarations and "no" indicate that external markup declarations might exist.

Declares and defines the elements used in the document class.

Defines the content of the XML document.

Meaning in Markup

- Structure
- Semantic
- Style

Well-formed XML document 1-3

- XML tags should be valid
- Length of the tags depend on the XML processors
- XML attributes should be valid
- XML documents should be verified

- Minimum of one element is required
- XML tags are case sensitive
- Every start tag should end with end tag
- XML tags should be nested properly

Well-formed XML document 2-3

Code Snippet

```
<Book>
<Name>Good XML</Name>
<Cost>$20</Cost>
</Book>
```

Well-formed XML document 3-3

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<Library>
  <Name>
    4ULibrary
    #1, A Mansion, Izaho
    $10000
</Library>
```

Non Well- Formed Document

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Library>
  <Name>4ULibrary</Name>
  <Address> #1, A Mansion, Izaho </Address>
  <Auction>$10000</Auction>
</Library>
```

Well-Formed Document

Lesson 4 – XML Syntax

In this last lesson, **XML Syntax**, you will learn to:

- State and describe the use of comments and processing instructions in XML.
- Classify character data that is written between tags.
- Describe entities, DOCTYPE declarations and attributes.

Comments 1-3

- Used for the people to give information about the code
- Usually made for the content

```
<?xml version="1.0" encoding="iso-8859-1" ?>
- <Name NickName="John">
    <First>John</First>
    <!-- John is yet to pay the term fees  -->
    <Last>Brown</Last>
    <Semester>Final</Semester>
  </Name>
```

Comments 2-3

- Rules:
  - Comments should not include "-" or "
  - Comments should not be placed within a tag or entity declaration
  - Comments should not be placed before the XML declaration
  - Comments can be used to comment the tag sets
  - Comment should not be nested

Comments 3-3

Code Snippet

```
<Name NickName='John'>
   <First>John</First>
   <!--John is yet to pay the term fees-->
<Last>Brown</Last> <Semester>Final</Semester>
</Name>
```

Processing Instructions 1-2

- The main objective is to present some special instructions to the application.
- All the processing instructions should begin with an identifier called target.

Syntax

```
<?PITarget <instruction>?>
```

where,

`PITarget` is the name of the application that should receive the processing instructions.

`<instruction>` is the instruction for the application

Processing Instructions 2-2

Code Snippet

```
<Name NickName='John'>
   <First>John</First>
   <!--John is yet to pay the term fees-->
   <Last>Brown</Last>
   <?feesprocessor SELECT fees FROM
STUDENTFEES?>
   <Semester>Final</Semester>
</Name>
```

where,

`feesprocessor` is the name of the application that receives the processing instruction

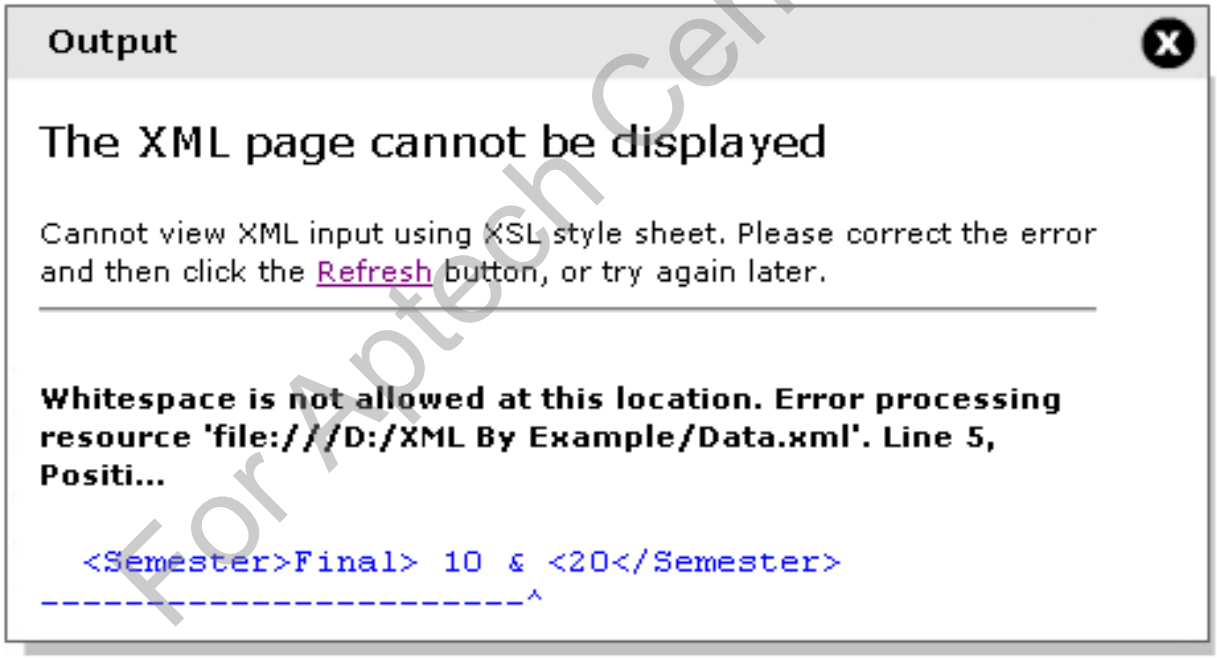
`SELECT fees FROM STUDENTFEES` is the instruction.

Classification of character data

- Character data describes the document's actual content with the white space.
- The character data can be classified into:
  - Character Data (CDATA)
  - Parsed Character Data (PCDATA)

PCDATA 1-2

- The data that is parsed by the parser is called as PCDATA.
- The main objective is to present some special instructions to the application.



PCDATA 2-2

Code Snippet

```
<Name nickname='John'>
   <First>John</First>
   <!--John is yet to pay the term fees-->
   <Last>Brown</Last>
   <Semester>Final> 10 & <20</Semester>
</Name>
```

CDATA

- CDATA part begins with a "<![CDATA[" and ends with "]]>".
- CDATA sections cannot be nested.

Code Snippet

```
<?xml version="1.0" standalone="yes"?>
<Svg>
  <Desc>Three shapes</Desc>
  <Para>But the formula was <![CDATA[if (&x1 +
&x2)]]> which
resulted in 7.</Para>
</Svg>>
```

Entities 1-3

- XML document is made up of large amount of information called as entities.
- Every entity consists a name and a value.
- Entity reference consists of an ampersand (&), the entity name, and a semicolon (;).

Entities 2-3

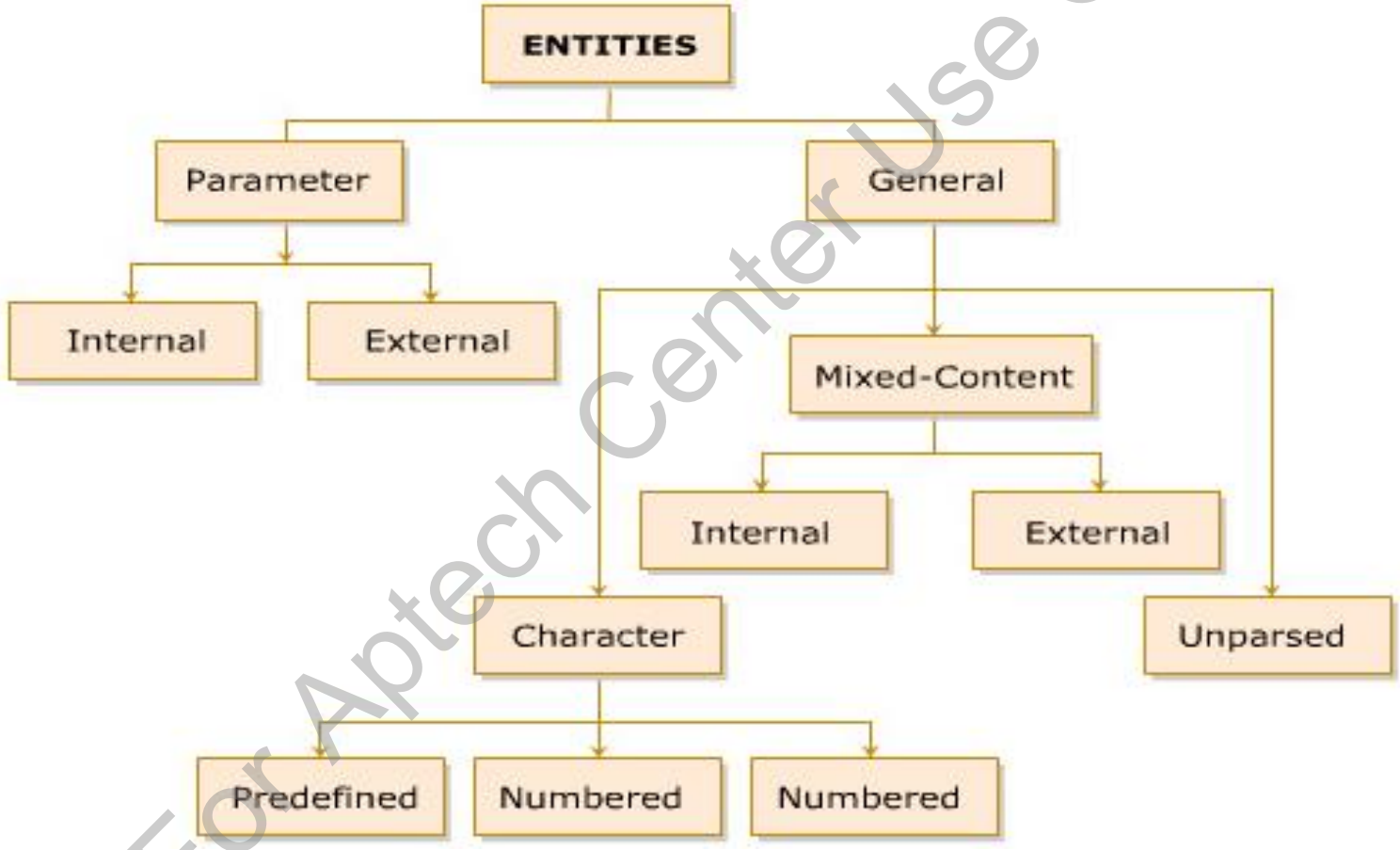
Predefined Entity	Description	Output
&lt;	produces the left angle bracket	<
&gt;	produces the right angle bracket	>
&amp;	produces the ampersand	&
&apos;	produces a single quote character	'
&quot;	produces a double quote character	"

Entities 3-3

Code Snippet

```
<?xml version="1.0"?>
<!DOCTYPE Letter [
<!ENTITY address "15 Downing St Floor 1">
<!ENTITY city "New York">
]>
<Letter>
<To>&quot;Tom Smith&quot;</To>
      <Address>&address;</Address>
      <City>&city;</City>
    <Body>
     Hi! How are you?
     The sum is &gt; $1000
     </Body>
     <From>ARNOLD</From>
</Letter>
```

Entity Categories 1-3



Entity Categories 2-3

- **General Entity**
  - These entities are used within the document content.

Code Snippet

```
<<!ENTITY % ADDRESS "text that is to be
represented by an entity">
```

A well-formed parameter entity will look like a general entity, except that it will include the "%" specifier.

Entity Categories 3-3

- **Parameter Entity**
  - These entities are used only in the DTD.

Code Snippet

```
<!DOCTYPE MusicCollection [
    <!ENTITY R "Rock">
    <!ENTITY S "Soft">
    <!ENTITY RA "Rap">
    <!ENTITY HH "Hiphop">
    <!ENTITY F "Folk">
    ]>
```

DOCTYPE declarations 1-3

- **DTD File:**

Syntax

```
<! DOCTYPE name_of_root_element
      SYSTEM "URL of the external DTD subset" [
       Internal DTD subset
]>
```

where,

`name_of_root_element` is the name of the root element

`SYSTEM` is the url where the DTD is located

`[Internal DTD subset]` are declarations in the document

DOCTYPE declarations 2-3

Code Snippet

```
<?xml version=”1.0” encoding=”ISO-8859-1”?>
<!DOCTYPE program SYSTEM “HelloJimmy.dtd”>
<Program>
<Comments>
This is a simple Java Program. It will display
the message “Hello Jimmy,How are you?” on
execution.
</Comments>
<Code> public static void main(String[] args) {
System.out.println(“Hello Jimmy,How are you?”);
// Display the string.
}
}
</Code>
</Program>
```

DOCTYPE declarations 3-3

- Document Type Definition defines the elements in the document

```
<!ELEMENT Program (comments, code)>
<!ELEMENT Comments (#PCDATA)>
<!ELEMENT Code (#PCDATA)>
```

- Output:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE program (View Source for full doctype...)>
- <Program>
    <Comments>This is a simple Java Program. It will display the message "Hello
      Jimmy,How are you?" on execution.</Comments>
    <Code>public static void main(String[] args) { System.out.println("Hello
      Jimmy,How are you?"); // Display the string. } }</Code>
  </Program>
```

Attributes

- Attributes are case sensitive and must start with a letter or underscore.

Syntax

```
<elementName attName1=”attValue2”
attName2=”attValue2”...>
```

Code Snippet

```
<?xml version=”1.0” ?>
  <Player Sex=”male”>
    <FirstName>Tom</FirstName>
    <LastName>Federer</LastName>
  </Player>
```

Summary 1-2

- **Introduction to XML**
  - XML was developed to overcome the drawbacks of earlier markup languages.
  - XML consists of set of rules that describe the content to be displayed in the document.
  - XML markup contains the content in the information containers called as elements.

- **Exploring XML**
  - XML is divided into two parts namely, document prolog and root element.
  - An XML editor creates the XML document and the parser validates the document.

Summary 2-2

- **Working with XML**
  - XML document is divided into XML Version Declaration, DTD and the document instance in which the markup defines the content.
  - The XML markup is again categorized into structural, semantic and stylistic.
  - The output of the XML document is displayed in the browser if it is well formed.

- **XML Syntax**
  - Comments are used in the document to give information about the line or block of code.
  - The content in XML document is divided into markup and character data.
  - The entities in XML are divided into general entities and parameter entities.
  - A DTD can be declared either internally or externally.