Unit 7 – Introduction to XML

Meta-markup Languages

- A markup language allows the user to identify individual elements of a document, e.g., what is a paragraph, a heading, or an unordered list.
- Used in combination with a style sheets, data can be properly presented on a web page, a slide show, or any other method that is appropriate for the data.

Meta-markup Languages

- A meta-markup language is a little different; it doesn't specify a document – it specifies a ***language***.
- SGML (***S***tandard ***G***eneralized ***M***arkup ***L***anguage) and XML (e***X***tensible ***M***arkup ***L***anguage) are examples of meta-markup languages.

SGML

- SGML was based on GML (***G***eneralized ***M***arkup ***L***anguage), which was developed at IBM in the 1960s. SGML was developed in 1974.
- SGML was intended to allow for the sharing of machine-readable documents.
- While it was used in the printing and publishing industry, its complexity kept it from wider use.
- SGML was used as the basis for HTML.

XML

- HTML describes the layout of information but conveys no information about its meaning. This limits the ability to retrieve information from an HTML document automatically.
- One solution to get around HTML's limitation is for groups of users to define and use their own set of tags and attribute and use a meta-markup language to implement them.
- XML is a simpler language than SGML and therefore more useful.

Using XML

- XML is not a replacement for XHTML. It is intended to provide a way to label data in a way that can be analyzed and manipulated automatically.
- XML is normally used together with a style sheet and an appropriate processor to produce a suitable XHTML document based on the XML file and the style sheet.

Syntax of XML

- XML has two levels of syntax:
  - The general low-level syntax within the XML doucment.
  - The higher-level syntax specified by DTD (Document Type Definitions) or XML schemas.
- XML documents can contain:
  - data elements of the document
  - markup declarations (instructions to the XML parser)
  - processing instructions (instructions for an application process that will process the data).

Elements of XML

- All XML documents begin with an XML declaration, which identifies the document as XML, and provides the version number of the XML standard being used and the encoding standard:

```
<?xml version = "1.0" encoding = "utf-8"?>
```

- Comments in XML are the same as in XHTML:

```
<!– This is a comment -->
```

Names in XML

- XML names are used to identify elements and attributes.
  - XML names must begin with a letter or an underscore and can contain letter, underscores, digits, hyphens and periods.
  - XML names are case sensitive; e.g., **`Body`**, **`body`** and **`BODY`** are three different names in XML.
  - There are no limits to the length of XML names.

Basic Syntax Rules

- Every XML documents defines a root element and that root element's tag must appear on the first line of XML code.
- All other elements must be nested within that element.
- For a XHTML document, the root tag is **`html`**.
- Every XML element must have a closing tag:
  - **`<myTag>`** … **`</myTag>`**
  - **`<myTag />`**

Sample XML Document

```
<?xml version = "1.0" encoding = "utf-8"?>
<ad>
  <year> 1960 </year>
  <make> Cessna <make>
  <model> Centurian </model>
  <color> Yellow with white trim </color>
  <location>
    <city> Gulfport </city>
    <state> Mississippi </state>
  </location>
</ad>
```

Another Sample XML Document

```
<?xml version = "1.0" encoding = "utf-8"?>
<bookstore>
 <book category="CHILDREN">
   <title>Harry Potter</title>
   <author>J K. Rowling</author>
   <year>2005</year>
   <price>29.99</price>
  </book>
  <book category="WEB">
   <title>Learning XML</title>
   <author>Erik T. Ray</author>
   <year>2003</year>
   <price>39.95</price>
  </book>
</bookstore>
```

XML Attributes

- In XML, attributes can be used to provide additional information about elements in an XML document.
- Example:

```
<file type = "gif"> computer.gif </file>
```

- Attributes must be enclosed in quotation marks (single or double)

```
<file type = 'gif'> computer.gif </file>
```

is also valid.

Attributes Or Nested Tags

- Is it better to add an additional attribute to an element or to define a nested element?
- Sometimes there is no choice – an image can only be an attribute (XML only handles text data).
- Nested tags can be added to any existing tag to describe its growing size and complexity – attributes give no information about this.

A Tag With One Attribute

```
<!-- A tag with one attribute -->
<patient name = "Maggie Dee Magpie ">
.... ….
</patient>
```

A Tag With One Nested Tag

```
<!-- A tag with one nested tag -->
<patient>
  <name> Maggie Dee Magpie
  </name>
.... ….
</patient>
```

An Extra Level Of Nested Tags

```
<!-- A tag with one nested tag, which
  contains three nexted tags -->
<patient>
  <name>
    <first> Maggie </first>
    <middle> Dee </middle>
    <last> Magpie </last>
  </name>
.... ….
</patient>
```

XML and Auxiliary Files

- An XML document often uses two auxiliary files:
  - One file specifies its tag set and structural syntactic rules. This can be either a DTD or an XML schema.
  - One file contains a style sheet to describe how the document's content is to be printed and/or displayed. This can be either a Cascading Style Sheet or an XSLT Style Sheet.

XML Document Structure

- An XML document consists of one or more entities that are logically related sets of data.
- The document entity describes the document as a whole and is usually subdivided into other entities.
- These other entities may (or may not) be physically located in the same file.
- Entity names can be any length

Entity Names

- Entity names can be any length.
- They must begin with a letter, a dash or a colon.
- The remaining characters can be letters, digits, periods, dashes, underscores or colons.
- Adding an amersand before and asemi-colon after a reference name turns it into a reference.
  - **`&apple_image`** is a reference to the entity **`apple_image`**.

Character Data Sections

- When a document requires several predefined entities near each other, it becomes hard to read; therefore, we can use a character data section.
- Character data sections are not parsed and appear in an XML document as they are written.
- Character data sections cannot contain tags because they are considered literal text they do not mark up the document.

CDATA

- Their basic syntax is:
  - **<![CDATA [***content***]]>**
- An example:

```
<![CDATA [The last word of the line is >>> here <<<]]>
```

- This is clearly superior to writing:

```
The last word of the line is &gt; &gt;
&gt; here &lt; &lt; &lt;
```

- If I wrote

```
<![CDATA [The form of the tag is &lt; tag name &lt;]]>
```

  I would get:

```
The form of the tag is &lt; tag name &lt;
```

Document Type Definitions

- A document type definition (DTD) is a set of rules specifying how a set of elements can appear in an XML document as well as entity declarations.
- While XML documents do not require DTDs, it allows the programmer to check an XML document for validity.
- A DTD can be internal (placed inside the XML document) or external (placed in a separate file that the XML document references).

DTD Syntax

- A DTD is a sequence of declarations:

  <!keyword … **>**
- There are 4 valid keywords:
  - **ELEMENT** – used to define tags
  - **ATTLIST** – used to define tag attributes
  - **ENTITY** – used to define entities
  - **NOTATION** – used to define data type notations

Declaring Elements

- Element declarations are a form that is similar to BNF.
- Each element declaration specifies the structure of ***one*** element, containing its names, its constituents (if it has child elements) or the data type of its parent (if it is a leaf).

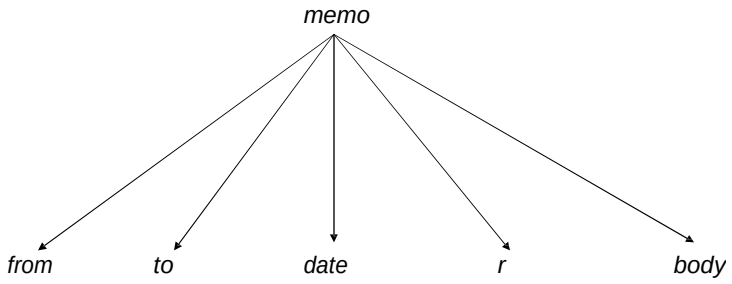
Declaring Non-leaf Elements

- The general form of an element declaration if there are child elements is:

  <!ELEMENT ElementName (*ChildElementList***)>**
- An example:

```
<!ELEMENT memo (from, to, date, re, body)>
```

Document Tree Structure



Child Element Specification Modifiers

- Normally, an element specification indicate one occurrence of an element.
- Using a modifier allows the programmer to have multiple occurrences of an element.

Modifier	**Meaning**
+	One or more occurrences
*****	Zero or more occurrences
?	Zero or one occurrence

Declaration With Element Modifiers

```
<!ELEMENT person (parent+, age, spouse?, sibling*>
```

indicates

- One or more parents
- Age
- Spouse (optional)
- Zero or more siblings

Declaring a Leaf

- The syntax for an element that does not have child elements is:

  <!ELEMENT *element_name* **(#PCDATA)>**

- An example

  <!ELEMENT year (#PCDATA)>

Declaring Attributes

- Attributes of an element are specified separately from the element declaration.
- An attribute declaration must include:
  - the attribute's name
  - the element to which it belongs
  - its type

Syntax for Attribute Declarations

- If the element has only one attribute it can be declared:

  <! ATTLIST *ElmntName AttribName AttribType [DefltVal]***>**

- Multiple attributes can be declared separately or in one declaration:

  <! ATTLIST *ElmntNm1 AttribNm1 AttribType1 DefltVal1*
  ElmntNm2 AttribNm2 AttribType2 DefltVal2
  … …
  ElmntNmN AttribNmN AttribTypeN1 DefltValN **>**

Possible Default Values

Value	**Meaning**
A value	The quoted value, which is used if none is specified in an element
#FIXED *value*	The quoted value, which every element will have and which cannot be changed
#REQUIRED	No default value is given; every instance of the element must specify a value
#IMPLIED	No default value is given (the browser chooses the default value); the value may or may not be specified in an element.

Declaring Attributes : An Example

- The declarations:

```
<! ATTLIST airplane places CDATA "4">
<! ATTLIST airplane engineType CDATA #REQUIRED>
<! ATTLIST airplane price CDATA #IMPLIED>
<! ATTLIST airplane manufactr CDATA #FIXED "Cessna">
```

- This is a valid XML element for this DTD:

```
<airplane places = "10" engineType = "jet>
  </airplane>
```

Declaring Entities

- General entities can be referenced anywhere in the content of an XML document.
- Parameter entities can only be referenced in DTDs.
- Syntax:

  <!ENTITY *[%] entityName "entityValue*">
- If the percent sign is included, it is a parameter entity.

Referencing Entities

- An entity can be referenced in a declaration by placing an ampersand before and a semi-colon after the name:

```
<!ATTLIST airport airportName CDATA &jfk; >
```

External Text Entities

- An entity can be too long to be placed within the DTD; they can be pages long.
- This can be handled by placing it in a different file. These are called external text entities:

<!ENTITY *entityName* **SYSTEM "***fileLocation***">**

A Sample DTD

```
<?xml version = "1.0" encoding = "utf-8"?>

<!-- planes.dtd - a document type definition for
                  the planes.xml document, which
                  specifies a list of used airplanes
                  for sale -->

<!ELEMENT planes_for_sale (ad+)>
<!ELEMENT ad (year, make, model, color, description,
              price?, seller, location)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT make (#PCDATA)>
<!ELEMENT model (#PCDATA)>
<!ELEMENT color (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT seller (#PCDATA)>
```

```
<!ELEMENT location (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>

<!ATTLIST seller phone CDATA #REQUIRED>
<!ATTLIST seller email CDATA #IMPLIED>

<!ENTITY c "Cessna">
<!ENTITY p "Piper">
<!ENTITY b "Beechcraft">
```

Internal DTDs

- DTDs can appear in the same file as the XML document or in a file of their own.
- If the DTD is included in the XML file, it's include in a **DOCTYPE** tag:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE planes [
    All the declarations go here
    ]>
    The XML document goes here
```

External DTDs

- If the DTD is in its own file, the XML document contains a DOCTYPE tag identifying the DTD file
  <!DOCTYPE *XMLDocumentRootName* **SYSTEM** "*DTDFileName*"**>**
- In our case, it is
  <!DOCTYPE planes_for_sale SYSTEM "planes.dtd">

planes.xml

```
<?xml version = "1.0" encoding = "utf-8"?>
<!-- planes.xml - A document that lists ads for
                  used airplanes  -->

<!DOCTYPE planes_for_sales SYSTEM "planes.dtd">

<planes_for_sale>
  <ad>
    <year> 1977 </year>
    <make> &c; </make>
    <model> Skyhawk </model>
    <color> Light blue and white </color>
    <description>   New paint, nearly new interior,
              685 hours SMOH, full IFR King avionics
    </description>
    <price> 23,495 </price>
```

```
    <seller phone = "555-222-3333"> Skyway Aircraft
    </seller>
    <location>
      <city> Rapid City </city>
      <state> South Dakota </state>
    </location>
  </ad>

  <ad>
    <year> 1965 </year>
    <make> &p; </make>
    <model> Cherokee </model>
    <color> Gold </color>
    <description>   240 hours SMOH, dual NAVCOMs,
             DME,  new Cleveland brakes, great shape
    </description>
    <price> 23, 495 </price>
```

```
    <seller phone = "555-333-2222"
            email = "jseller@www.axl.com">
              John SellerSkyway Aircraft </seller>
    <location>
      <city> St. Joseph </city>
      <state> Missouri </state>
    </location>
  </ad>

</planes_for_sale>
```