eXtensible Markup Language (XML)

eXtensible Markup Language (XML)

- Extensible Markup Language (XML) is derived from SGML (ISO 8879).
- Created by W3C : designed to store and transport data
- XML is used in the **exchange of data across applications** on the Web
- XML tags are not predefined – all are user defined tags
- XML is designed to be self-descriptive

XML Sample

```
<?xml version="1.0"?>
   <book>
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2007-11-01</publish_date>
    </book>
```

HTML vs XML

- XML was designed to **transport and store data** with focus on the data , HTML was designed to **display data** with focus on the looks**.**
- HTML includes **pre-defined tags** to allow the author to specify how each piece of content should be **presented to the end user.**
- XML allows to **create own tags to describe the data,** with main focus on organising the data with good descriptive tags (or elements).
- NOT a replacement for HTML

XML Separates Data from HTML

- Display **dynamic data** in HTML document
  - Editing HTML for each time data changes is lot of work
- **Solution**
  - Store data in separate XML files.
  - use HTML/CSS for display and layout, implies changes in the underlying data will not require any changes to the HTML.
  - Use JavaScript code, to read an external XML file and update the data content of the web page.

XML Design Goals

- The design goals of XML emphasize
  - simplicity,
  - generality, and
  - usability over the Internet.
- XML data is stored in text format
- Strong support for different languages via **Unicode**

XML applications

- Used in several document formats like RSS, XHTML, SOAP, etc are modelled using XML
- Languages based on XML - http://en.wikipedia.org/wiki/List_of_XML_markup_languages
  - MathML - a language describing mathematical notation
  - SAML - authentication and authorization data
- XML is also used as the base language for communication protocols, such as **XMPP**.

XML Document

- XML Document has different parts
  - **Prolog**
    - XML Declaration
    - DTD Declaration
  - **Root Element (Document)**
    - Elements (Nested Elements)
    - Element Attributes and Values
    - Data

XML Prolog

- The prolog is an optional component of the XML document.
- If included, the prolog must be **appear before the root element.**
- A prolog consists of two parts:
  - **the XML declaration**
  - **the Document Type Declaration (DTD)**

XML Declaration

- This defines the version of XML used.
- The declaration is not absolutely necessary, but recommended to be included
- Generally version and encoding format is included in declaration
- The current version is 1.0

<?xml version="1.0" encoding="ISO-8859-15">

XML DTD

- The Document Type Declaration is a file that contains the necessary rules that the XML code in this file must follow
- There are two type declarations that may be used to reference an external DTD: **PUBLIC and SYSTEM.**
- **Eg:**
- <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

XML Declaration and DTD

- XML document will have the XML declaration first, followed by the DTD declaration.

  <?xml version="1.0"?>

  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

XHTML Example

- **!DOCTYPE** - Tell the XML processor that this piece of code defines the Document Type Definition
- **html** - Specifies the root element of the XML document. Here our example is an HTML file, which has <html> as the root element.
- **PUBLIC** - Specifies that this a publicly available DTD.
- **"//W3C//DTD XHTML 1.0 Transitional//EN" "** - The definition of the public document.
- **"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"** - The physical location of the DTD.

XML Element

- An **XML element** is everything from (including) the element's start tag to (including) the element's end tag.
- An element can contain:
  - other elements
  - text
  - attributes
  - or a mix of all of the above...

XML Naming Rules

- XML elements must follow these naming rules:
  - Names can contain **letters, numbers, and other characters**
  - Names cannot start with a number or punctuation character
  - Names cannot start with the letters xml (or XML, or Xml, etc)
  - Names cannot contain spaces
  - Any name can be used, no words are reserved.

XML Elements

- **All XML Elements Must Have a Closing Tag**
  - The XML declaration is not a part of the XML document itself, and it has no closing tag.
  - If there is no text content then combine start and end tag - <element />
- **XML Tags are Case Sensitive**
  - Opening and closing tags must be written with the same case
  - <Message>This is incorrect</message>
  - **<message>This is correct</message>**

XML Elements

- **XML Elements Must be Properly Nested**
  - Eg: <a> **** some data ****
- **XML Documents Must Have a Root Element**
  - XML documents must contain one element that is the **parent of all other elements.**
  - This element is called the **root element.**

XML Attributes

- XML elements can have attributes, just like HTML.
- Attributes provide **additional information** about an element.
- **Metadata (data about data)** like id should be stored as attributes, and the data itself should be stored as elements.

XML Attributes

- **XML Attribute Values Must be Quoted**
  - XML elements can have attributes in name/value pairs just like in HTML
  - Values are mandatory in XML
    - <student **present= “true”**> - valid
    - <student **present**> - invalid
  - **In XML, the attribute values must always be quoted – either ' or “ can be used**
  - Eg: <book category=**"Web Developement"**>

    <tool name= “Tester's Tool”>

XML Data Syntax

- **XML Data - Text**
- **Comments**

  <!-- This is a comment. It is similar to HTML comments -->
- **White-space is Preserved in XML**
  - HTML truncates multiple white-space characters to one single white-space, With XML, the white-space in a document is **not truncated**.
- **XML Stores New Line as LF**
  - Windows uses Carriage Return (CR) and Line Feed (LF)
  - Unix/Mac uses Line Feed (LF)
  - XML stores a new line as LF.

XML Data Syntax

- **Entity References**
  - Some characters have a special meaning in XML like <, >, &, ', “
  - **Error:**

    <message>if **salary < 1000** then</message>
  - **Replace them with entity reference**

    <message>if **salary < 1000** then</message>
  - There are 5 predefined entity references in XML:
    - < < less than
    - > > greater than
    - & & ampersand
    - ' ' apostrophe
    - " " quotation mark

XML Entities

- An entity is a symbolic representation of information.
- The format of an entity in XML is **an ampersand (&), followed by the name of the symbol, and concluded with a semicolon.**

  &name;

User defined entities

- An entity must be created in the Document Type Definition (DTD).
- **Syntax**

  <!ENTITY entityName "The text to replaced">
- Eg: <!ENTITY intro "Hello XML”>
- **Using entities**

  <Book>

  <author>James</author>

  <description>**&intro;**</description>

  </book>

Relationship between elements

- The terms **parent, child, and sibling** are used to describe the relationships between elements.
- Parent elements have children.
- Children on the same level are called siblings.

XML Documents Form a Tree Structure

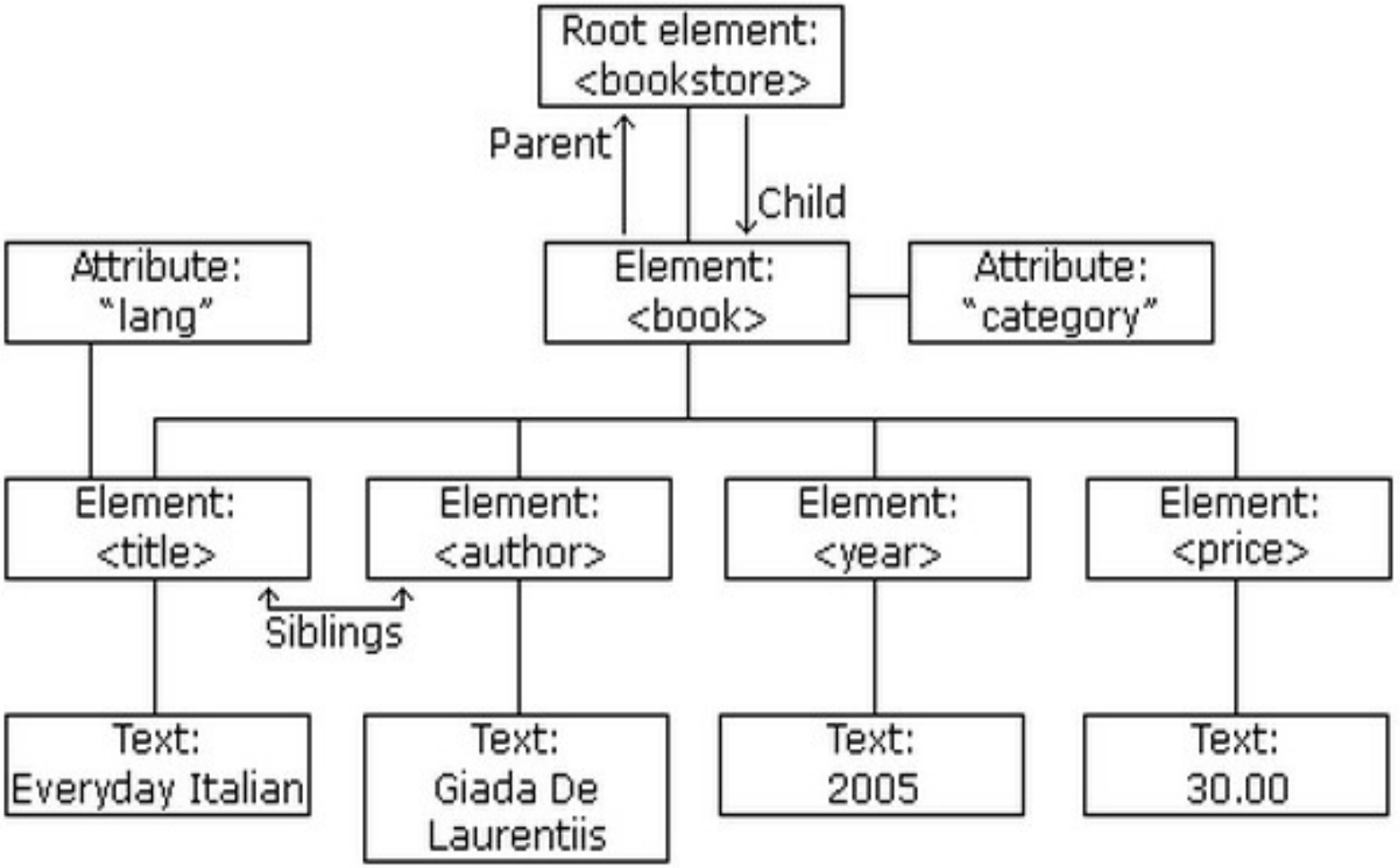
- XML documents must contain a **root** element, which is "the parent" of all other elements.
- The **elements** in an XML document form a **document tree**.
- The tree starts at the root and branches to the lowest level of the tree.
- All elements can have **text content and attributes** (just like in HTML).

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

bookstore.xml

```
<?xml version="1.0"?>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
<book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```

XML Tree – One book from bookstore.xml



XML Validation

- **Well Formed XML**
  - XML with correct syntax is "Well Formed" XML.
- **Valid XML**
  - XML validated against a DTD or XML Schema is "Valid" XML.

Well Formed XML Documents

- A "Well Formed" XML document has correct XML syntax.
  - XML documents must have a root element
  - XML elements must have a closing tag
  - XML tags are case sensitive
  - XML elements must be properly nested
  - XML attribute values must be quoted

Well Formed XML Documents

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Ann</to>
<from>Bob</from>
<heading>Reminder</heading>
<body>Test Reminder</body>
</note>
```

Valid XML Documents

- A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a Document Type Definition (DTD)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Ann</to>
<from>Bob</from>
<heading>Reminder</heading>
<body>Test Reminder</body>
</note>
```

Note.dtd

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

XML Namespaces

- Problem with user defined tags :
  - xml element overlap - there is the chance that the element's name already exists
  - Eg:
  - <?xml version="1.0" encoding="ISO-8859-15"?>
  - <html>
  - <body>
  - Welcome to my Health Resource
  - </body>
  - **<body>**
  - **<height>6ft</height>**
  - **<weight>155 lbs</weight>**
  - **</body>**
  - </html>

XML Namespaces

- Solution – **XML Namespaces**
  - a special type of **reserved XML attribute** that is placed in an XML tag.
  - like a prefix that is attached to any namespace created.
  - This attribute prefix is **"xmlns:"**, **XML NameSpace.**
  - The colon is used to separate the prefix from the namespace.
  - xmlns must have a unique value that no other namespace in the document has.

XML with namespaces

- <?xml version="1.0" encoding="ISO-8859-15"?>
- **<html:html xmlns:html='http://www.w3.org/TR/xhtml1/'>**
- <html:body>
- <html:p>Welcome to my Health Resource</html:p>
- </html:body>
- **<health:body xmlns:health='http://www.example.org/health'>**
- **<health:height>6ft</height>**
- **<health:weight>155 lbs</weight>**
- **</health:body>**
- </html:html>

Local Namespace

- Namespace defined against the root element - to be used for the whole document, and hence prefix all child elements with the same namespace.
- Local Namespace - define namespaces against a child node.
- This way, we could use multiple namespaces within the same document if required.

Example Local Namespace

- <books>
- <book>
- <bk:title xmlns:bk="http://somebooksite.com/book_spec">
- The Dream Saga
- </bk:title>
- <author>Matthew Mason</author>
- </book>
- ...
- </books>

Multiple Namespaces

- <bk:books xmlns:bk="http://somebooksite.com/book_spec">
- <bk:book>
- <bk:title>The Dream Saga</bk:title>
- <bk:author>Matthew Mason</bk:author>
- <pub:name xmlns:pub="http://somepublishingsite.com/spec">
- Sid Harta Publishers
- </pub:name>
- <pub:email>author@sidharta.com.au</pub:email>
- </bk:book>
- ...
- </bk:books>

Default Namespace

- The prefix used when defining the namespace, is used in each element that referred to the namespace.
- The default namespace is one where prefix is not applied

  Eg: There is no prefix specified with xmlns
- <books xmlns="http://somebooksite.com/book_spec">
- <book>
- <title>The Dream Saga</title>
- <author>Matthew Mason</author>
- </book>
- ...
- </books>

Practise Questions

- Create a well formed XML document to
  - Represent your profile – name, rollno, cgpa, area of interest, etc
  - Describe your current semester courses.
  - Draw the tree structure for both
  - Identify the relationships between the elements