EXtensible Markup Language XML

What is XML?

- XML stands for EXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to carry data, not to display data
- XML tags are not predefined. You must define your own tags
- XML is designed to be self-descriptive
- XML is a W3C Recommendation

XML Does not DO Anything

- Maybe it is a little hard to understand, but XML does not DO anything. XML was created to structure, store, and transport information.
- The following example is a note to Sam from Kandy, stored as XML:

```
<note>
<to>Nimal</to>
<from>Kamal</from>
<heading> Note</heading>
<body>Coming home this week</body>
</note>
```

- The note above is quite self descriptive. It has sender and receiver information, it also has a heading and a message body.
- But still, this XML document does not DO anything. It is just pure information wrapped in tags. Someone must write a piece of software to send, receive or display it.

XML is Just Plain Text

- XML is nothing special. It is just plain text. Software that can handle plain text can also handle XML.
- However, XML-aware applications can handle the XML tags specially. The functional meaning of the tags depends on the nature of the application.

With XML You Invent Your Own Tags

- The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.
- That is because the XML language has no predefined tags.
- The tags used in HTML (and the structure of HTML) are predefined. HTML documents can only use tags defined in the HTML standard (like <p>, <h1>, etc.).
- XML allows the author to define his own tags and his own document structure.

XML is Not a Replacement for HTML

- It is important to understand that XML is not a replacement for HTML. In most web applications, XML is used to transport data, while HTML is used to format and display the data.
- My best description of XML is this:
- **XML is a software- and hardware-independent tool for carrying information.**

How Can XML be Used?

- XML is used in many aspects of web development, often to simplify data storage and sharing.
  - XML Separates Data from HTML
  - XML Simplifies Data Sharing
  - XML Simplifies Data Transport
  - XML Simplifies Platform Changes
  - XML Makes Your Data More Available
  - XML is Used to Create New Internet Languages

XML Separates Data from HTML

- If you need to display dynamic data in your HTML document, it will take a lot of work to edit the HTML each time the data changes.
- With XML, data can be stored in separate XML files. This way you can concentrate on using HTML for layout and display, and be sure that changes in the underlying data will not require any changes to the HTML.
- With a few lines of JavaScript, you can read an external XML file and update the data content of your HTML.

XML Simplifies Data Sharing

- In the real world, computer systems and databases contain data in incompatible formats.
- XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data.
- This makes it much easier to create data that different applications can share.

XML Simplifies Data Transport

- With XML, data can easily be exchanged between incompatible systems.
- One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet.
- Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

XML Simplifies Platform Changes

- Upgrading to new systems (hardware or software platforms), is always very time consuming. Large amounts of data must be converted and incompatible data is often lost.
- XML data is stored in text format. This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

XML Makes Your Data More Available

- Since XML is independent of hardware, software and application, XML can make your data more available and useful.
- Different applications can access your data, not only in HTML pages, but also from XML data sources.
- With XML, your data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc), and make it more available for blind people, or people with other disabilities.

XML is Used to Create New Internet Languages

- A lot of new Internet languages are created with XML.
- Here are some examples:
  - XHTML the latest version of HTML
  - WSDL for describing available web services
  - WAP and WML as markup languages for handheld devices
  - RSS languages for news feeds
  - RDF and OWL for describing resources and ontology
  - SMIL for describing multimedia for the web

XML Tree

- XML documents form a tree structure that starts at "the root" and branches to "the leaves".
  - An Example XML Document
    - XML documents use a self-describing and simple syntax:
    - The first line is the XML declaration.
    - It defines the XML version (1.0) and the encoding used (ISO-8859-1 = Latin-1/West European character set).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
 <to>Nimal</to>
 <from>Kamal</from>
 <heading>Note</heading>
 <body>Coming home this week</body>
</note>
```

XML Tree …

- The next line describes the **root element** of the document (like saying: "this document is a note"):

```
<note>
```

- The next 4 lines describe 4 child elements of the root (to, from, heading, and body):

```
<to>Nimal</to>
<from>Kamal</from>
<heading>Note</heading>
<body>Coming home this week</body>
```

XML Tree …

- And finally the last line defines the end of the root element:

```
<note>
```

- You can assume, from this example, that the XML document contains a note to Sam from Kandy.
- Don't you agree that XML is pretty self-descriptive?

XML Documents Form a Tree Structure

- XML documents must contain a **root element**. This element 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 sub elements (child elements):

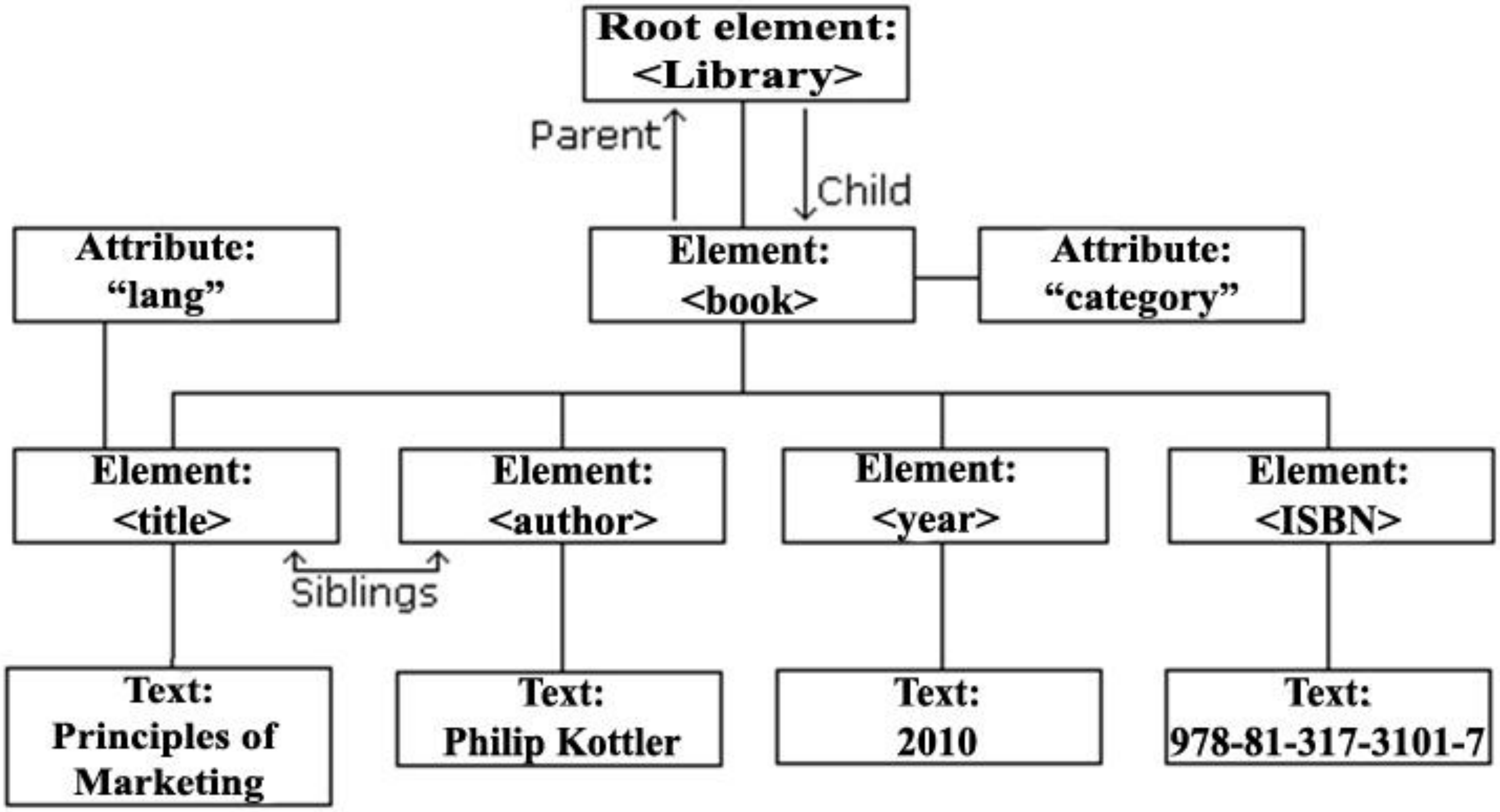
XML Documents Form a Tree Structure

- XML documents must contain a **root element**. This element 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 sub elements (child elements):

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

- 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 (brothers or sisters).
- All elements can have text content and attributes (just like in HTML).

Example:


Root element: <Library>
Parent
Child
Attribute: “lang”
Element: <book>
Attribute: “category”
Element: <title>
Element: <author>
Element: <year>
Element: <ISBN>
Siblings
Text: Principles of Marketing
Text: Philip Kottler
Text: 2010
Text: 978-81-317-3101-7

The image above represents one book in the XML below:

```
<library>
  <book category="Marketing">
    <title lang="en">Principles of
Marketing</title>
    <author>Philip Kotler </author>
    <year>2010</year>
    <ISBN>978-81-317-3101-7</ISBN>
  </book>
  <book category="Management">
    <title lang="en">Enterpreneurship</title>
    <author>Donald F. Kuratko</author>
    <year>2009</year>
    <ISBN>978-81-315-0561-8</ISBN>
  </book>
  <book category=" Management ">
    <title lang="en"> Management </title>
    <author>James A.F. stoner</author>
    <year>2009</year>
    <ISBN>978-81-317-0704-3</ISBN>
  </book>
</library>
```

- The root element in the example is <bookstore>.
- All <book> elements in the document are contained within <bookstore>.
- The <book> element has 4 children:
- <title>,< author>, <year>, <price>.

XML Syntax Rules

- The syntax rules of XML are very simple and logical. The rules are easy to learn, and easy to use.
  - **All XML Elements Must Have a Closing Tag**
    - In XML, it is illegal to omit the closing tag. All elements **must** have a closing tag:
  - **XML Tags are Case Sensitive**
    - XML elements are defined using XML tags.
    - XML tags are case sensitive. With XML, the tag <Letter> is different from the tag <letter>.
    - Opening and closing tags must be written with the same case:

```
<Message>This is incorrect</message>
<message>This is correct</message>
```

- "Opening and closing tags" are often referred to as "Start and end tags". Use whatever you prefer. It is exactly the same thing.

XML Syntax Rules …

- **XML Elements Must be Properly Nested**
- In HTML, you might see improperly nested elements:

```
<b><u>This text is bold and italic</b></u>
```

- In XML, all elements **must** be properly nested within each other:

```
<b><u>This text is bold and italic</u></b>
```

- In the example above, "Properly nested" simply means that since the <i> element is opened inside the <b> element, it must be closed inside the <b> element.

XML Syntax Rules …

- XML documents must contain one element that is the **parent** of all other elements. This element is called the **root** element.

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

XML Syntax Rules …

- **XML Attribute Values Must be Quoted**

```
<note date=12/12/2013>
  <to>Nimal</to>
  <from>Kamal</from>
</note>
```

```
<note date="12/12/2013">
  <to>Nimal</to>
  <from>Kamal</from>
</note>
```

•The error in the first document is that the date attribute in the note element is not quoted.

XML Syntax Rules …

- **Comments in XML**
  - The syntax for writing comments in XML is similar to that of HTML.
  - <!-- This is a comment -->
- **White-space is Preserved in XML**
  - HTML truncates multiple white-space characters to one single white-space:

HTML:	Hello my name is Nimal
Output:	Hello my name is Nimal.

  - With XML, the white-space in a document is not truncated.

What is an 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, simple text or a mixture of both. Elements can also have attributes.

```
<library>
 <book category="Marketing">
  <title>Principles of
Marketing</title>
  <author>Philip Kotler</author>
  <year>2010</year>
  <ISBN>978-81-317-3101-7</ISBN>
 </book>
 <book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
 </book>
</bookstore>
```

- In the example above, <bookstore> and <book> have element contents, because they contain other elements. <author> has text content because it contains text.
- In the example above only <book> has an attribute (category=" WEB ").

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.

Best Naming Practices

- Make names descriptive. Names with an underscore separator are nice: <first_name>, <last_name>.
- Names should be short and simple, like this: <book_title> not like this: <the_title_of_the_book>.
- Avoid "-" characters. If you name something "first-name," some software may think you want to subtract name from first.
- Avoid "." characters. If you name something "first.name," some software may think that "name" is a property of the object "first."
- Avoid ":" characters. Colons are reserved to be used for something called namespaces (more later).
- XML documents often have a corresponding database. A good practice is to use the naming rules of your database for the elements in the XML documents.
- Non-English letters like éòá are perfectly legal in XML, but watch out for problems if your software vendor doesn't support them.

XML Elements are Extensible

- XML elements can be extended to carry more information.
  - Look at the following XML example:

```
<note>
<to>Nimal</to>
<from>Kamal</from>
<body> Coming home this week </body>
</note>
```

Let's imagine that we created an application that extracted the <to>,< from>, and <body> elements from the XML document to produce this output:

```
MESSAGE
To: Nimal
From: Kamal
Coming home this week
```

XML Elements are Extensible …

- Imagine that the author of the XML document added some extra information to it:

```
<note>
<date>2013-11-10</date>
<to>Nimal</to>
<from>Kamal</from>
<heading>Notice</heading>
< Coming home this week </body>
</note>
```

- Should the application break or crash?
- No. The application should still be able to find the <to>, <from>, and <body> elements in the XML document and produce the same output.
- One of the beauties of XML, is that it can often be extended without breaking applications.

XML Attributes

- XML elements can have attributes in the start tag, just like HTML.
- Attributes provide additional information about elements.
- Attributes often provide information that is not a part of the data.
- In the example below, the file type is irrelevant to the data, but important to the software that wants to manipulate the element:

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

XML Attributes Must be Quoted

- Attribute values must always be enclosed in quotes, but either single or double quotes can be used. For a person's sex, the person tag can be written like this:

```
<person sex="female">
```

- or like this:

```
<person sex='female'>
```

- If the attribute value itself contains double quotes you can use single quotes, like in this example:

```
<gangster name='George "Shotgun" Ziegler'>
```

- or you can use character entities:

```
<gangster name="George &quot;Shotgun&quot; Ziegler">
```

XML Elements vs. Attributes

- Take a look at these examples:

```
<person sex="female">
  <firstname>Namal</firstname>
  <lastname>Perera</lastname>
</person>
```

```
<person>
  <sex>female</sex>
  <firstname>Chamari</firstname>
  <lastname>Weerasena</lastname>
</person>
```

In the first example sex is an attribute. In the last, sex is an element. Both examples provide the same information.

Avoid XML Attributes?

- Some of the problems with using attributes are:
  - attributes cannot contain multiple values (elements can)
  - attributes cannot contain tree structures (elements can)
  - attributes are not easily expandable (for future changes)
- Attributes are difficult to read and maintain. Use elements for data. Use attributes for information that is not relevant to the data.

XML Attributes for Metadata

- Sometimes ID references are assigned to elements. These IDs can be used to identify XML elements in much the same way as the ID attribute in HTML. This example demonstrates this:

```
<messages>
 <note id="501">
  <to>Nimal</to>
  <from>Kamal</from>
  <heading>Reminder</heading>
  <body> Coming home this week </body>
 </note>
 <note id="502">
  <to>Kamal</to>
  <from>Nimal</from>
  <heading>Re: Reminder</heading>
  <body>I will not</body>
 </note>
</messages>
```

- The ID above is just an identifier, to identify the different notes. It is not a part of the note itself.
- What I'm trying to say here is that metadata (data about data) should be stored as attributes, and that data itself should be stored as elements.

Well Formed XML Documents

- A "Well Formed" XML document has correct XML syntax.
- The syntax rules were described in the previous chapters:
  - 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

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Nimal</to>
<from>Kamal</from>
<heading>Reminder</heading>
<body> Coming home this week </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>Nimal</to>
<from>Kamal</from>
<heading>Reminder</heading>
<body> Coming home this week </body>
</note>
```

The DOCTYPE declaration in the example above, is a reference to an external DTD file. The content of the file is shown in the paragraph below.

XML DTD

- The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements:

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

The DOCTYPE declaration in the example above, is a reference to an external DTD file. The content of the file is shown in the paragraph below.