



Lesson #14:

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 not a replacement for HTML. HTML is about displaying information, while XML is about carrying information.

What is XML?



- XML documents do 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 nothing special. It is just plain text. Software that can handle plain text can also handle XML.
- XML Separates Data from HTML.

Why use XML?



- 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 and transport because it is pure text format.

Why use XML?



- 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 versatile

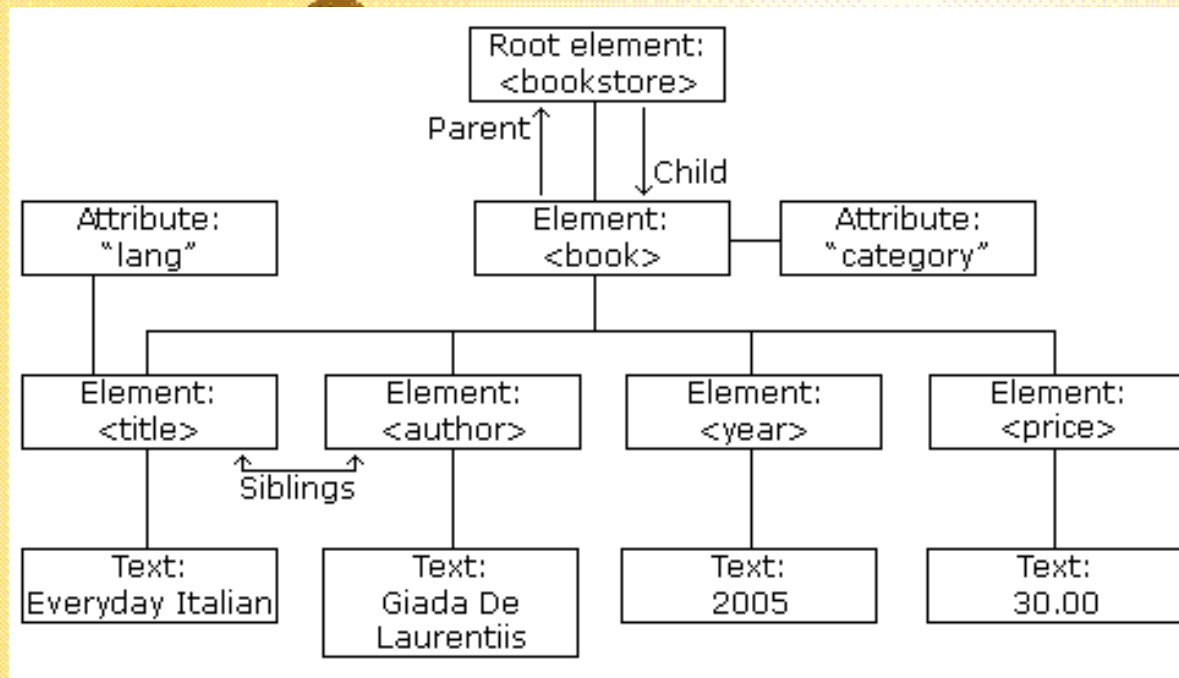


- A lot of new Internet languages are created with XML:
- XHTML, the latest version of HTML .
- WSDL for describing available web services.
- WAP and WML as markup languages for handheld devices.
- RSS languages for news feeds.
- RDF and OWL for describing resources and ontology.
- SMIL for describing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

XML structure



- XML documents form a tree structure that starts at "the root" and branches to "the leaves".



```
<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>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

XML syntax



- All elements must have a closing tag.
- 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
- In XML, all elements must be properly nested within each other.
- XML documents must contain one element that is the parent of all other elements. This element is called the root element.

XML syntax



- In XML the attribute value must always be quoted.
- Some characters have a special meaning in XML. If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element. To avoid this error, replace the "<" character with an entity reference.

There are 5 predefined entity references in XML:

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

Note: Only the characters "<" and "&" are strictly illegal in XML. The greater than character is legal, but it is a good habit to replace it.

XML syntax



- The syntax for writing comments in XML is similar to that of HTML:

```
<!-- This is a comment -->
```
- Unlike HTML, the white-space in a document is not truncated in XML.
- XML elements must follow these naming rules: Names can contain letters, numbers, and other characters; they cannot start with a number or punctuation character; they cannot start with the letters xml (or XML, or Xml, etc), they cannot contain spaces (avoid . : - in names).

XML attributes



- Attributes provide additional information about elements.
- See this page for more information:

http://www.w3schools.com/xml/xml_attributes.asp

XML validation



- XML with correct syntax is called "Well Formed" XML.
- XML that is validated against a DTD is called "Valid" XML.
- See this page for more information:

http://www.w3schools.com/xml/xml_dtd.asp

XML display



- By default, XML is displayed only in its raw form (source) in the browser.
- Unlike HTML, errors in XML code will be indicated.
- XML documents do not carry information about how to display the data. Since XML tags are "invented" by the author of the XML document, browsers do not know if a tag like `<table>` describes an HTML table or a dining table.

Ex: www.xmlfiles.com/examples/plant_catalog.xml

XML display with CSS



- With CSS (Cascading Style Sheets) you can add display information to an XML document.
- See this page for more information:

http://www.w3schools.com/xml/xml_display.asp

- Formatting XML with CSS is not the most common method. W3C recommend using XSLT instead.
- We will revisit XML display after the lesson on XSLT.

XML display with JavaScript



- With an XMLHttpRequest you can communicate with your server from inside a web page.
- You can update a web page with new data without reloading the page.
- You can request and receive new data from a server after the page has loaded.
- You can communicate with a server in the background.

XML display with JavaScript



- Creating an XMLHttpRequest object is done with one single line of JavaScript:
`var xmlhttp=new XMLHttpRequest()`
- Most browsers have a built-in XML parser to read and manipulate XML. The parser converts XML into a JavaScript accessible object (the XML DOM).
- See this page:
www.w3schools.com/xml/xml_parser.asp

XML display with JavaScript



- Creating an XMLHttpRequest object is done with one single line of JavaScript:

```
var xmlhttp=new XMLHttpRequest()
```
- Most browsers have a built-in XML parser to read and manipulate XML. The parser converts XML into a JavaScript accessible object (the XML DOM).
- See this page:
www.w3schools.com/xml/xml_parser.asp

XML display with JavaScript



- The DOM (Document Object Model) defines a standard way for accessing and manipulating documents. Let's get the text from the `<ARTIST>` element using XML DOM:
`xmlDoc.getElementsByTagName("ARTIST")`
`[0].childNodes[0].nodeValue`
- `xmlDoc`: the XML document created by the parser.
- `getElementsByTagName("ARTIST")[0]`: the first `<ARTIST>` element.
- `childNodes[0]`: the first child of the `<ARTIST>` element (the text node).
- `nodeValue`: the value of the node (the text itself).

XML display with JavaScript



- We can then transfer it into the regular HTML DOM for use in JavaScript:

```
document.getElementById("ARTIST").innerHTML =  
xmlDoc.getElementsByTagName("ARTIST")  
[0].childNodes[0].nodeValue
```

- You can also get the value of an attribute if the element has it.

```
attrib = xmlDoc.getElementsByTagName("title")  
[0].getAttribute("lang");
```

- Changing the value of an element:

```
x=xmlDoc.getElementsByTagName("title")[0].childNodes[0];  
x.nodeValue="Gone with the Wind";
```

XML display with JavaScript



- Changing the value of an attribute:

```
x=xmlDoc.getElementsByTagName("book");  
for(i=0; i<x.length; ++i) {  
    x[i].setAttribute("edition","first");}
```

- Creating an element.

```
newel=xmlDoc.createElement("edition");  
newtext=xmlDoc.createTextNode("First");  
newel.appendChild(newtext);  
x=xmlDoc.getElementsByTagName("book");  
x[0].appendChild(newel);
```

- Removing an element:

```
x=xmlDoc.getElementsByTagName("book")[0];  
x.removeChild(x.childNodes[0]);
```

XML Namespaces



- In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.
- This XML carries HTML table information:

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

- This XML carries information about a table (a piece of furniture):

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

XML Namespaces



- If these previous XML fragments were added together, there would be a name conflict. Both contain a `<table>` element, but the elements have different content and meaning.
- Name conflicts in XML can easily be avoided using a name prefix. The following carries information about an HTML table, and a piece of furniture:

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

XML Namespaces



- When using prefixes in XML, a so-called namespace for the prefix must be defined.
- The namespace is defined by the `xmlns` attribute in the start tag of an element. The namespace declaration has the following syntax.
`xmlns:prefix="URI"`.
- When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace.
- Namespaces can be declared in the elements where they are used or in the XML root element

XML Namespaces



- The URLs for the namespaces just point to a page describing the elements (usually 404 not found however).

```
<root>
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
<f:table xmlns:f="http://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
</root>
```

```
<root
xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="http://www.w3schools.com/furniture">
  <h:table>
    <h:tr>
      <h:td>Apples</h:td>
      <h:td>Bananas</h:td>
    </h:tr>
  </h:table>
  <f:table>
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>
</root>
```



End of lesson