INTERNET & WEB APPLICATION DEVELOPMENT SWE 444

Fall Semester 2008-2009 (081)

Module 4 (VII): XML DOM

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives/Outline

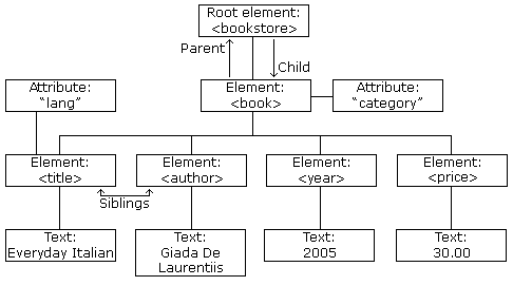
- Objectives
  - Understand the role of XML DOM
  - Learn how to manipulate XML document using the XML DOM
- Outline
  - What is DOM?
  - The XML DOM
  - The XML DOM Nodes
  - XML DOM Parsers
  - Node Properties
  - Node Methods
  - Examples

What is DOM?

- DOM is a W3C standard.
- DOM defines a standard for accessing documents like XML and HTML:
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*
- DOM is separated into 3 different parts / levels:
  - Core DOM - standard model for any structured document
  - XML DOM - standard model for XML documents
  - HTML DOM - standard model for HTML documents
- All nodes can be accessed through the tree. Their contents can be modified or deleted, and new elements can be created.
- Properties and methods define the programming interface to the DOM.

The XML DOM

- The XML DOM defines a standard way for accessing and manipulating XML documents.
- It views an XML document as a tree-structure called a node-tree.
  - XML elements, attributes, and text as nodes
  - The tree starts at the root node and branches out to the text nodes at the lowest level of the tree:



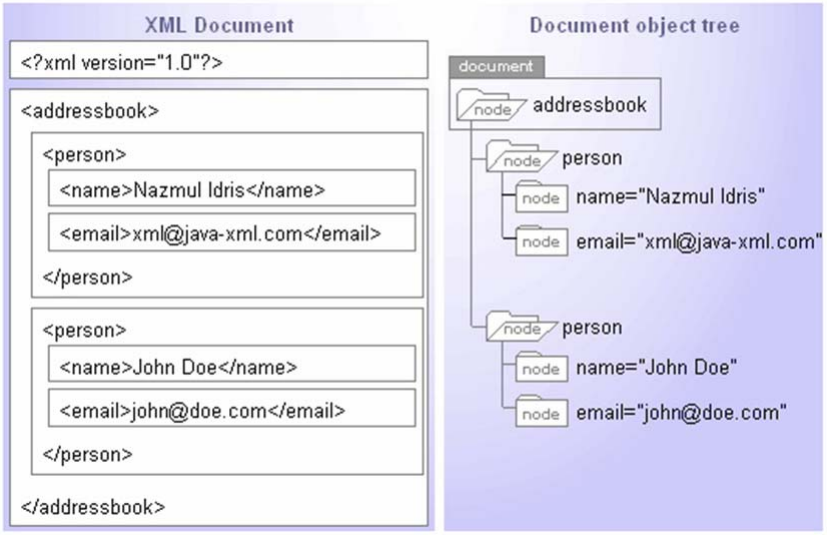
The XML DOM (cont.)

- The XML DOM is a W3C Recommendation
- XML is provides a standard programming interface for XML documents that is platform- and language-independent
- In other words:
  - **The XML DOM is a standard for how to get, change, add, or delete XML elements.**

The XML DOM Nodes

- In XML DOM, everything in an XML document is a node
  - The entire document is a document node
  - Every XML element is an element node
  - The text in the XML elements are text nodes
  - Every attribute is an attribute node
  - Comments are comment nodes
- A common error in XML DOM processing is to expect an element node to contain text.
  - However, the text of an element node is always stored in a text node
  - E.g.: for **<year>2005</year>**, the element node <year>, holds a text node with the value "2005".
    - In other words, "2005" is **not** the value of the <year> element!

DOM Example 1



DOM Example 2

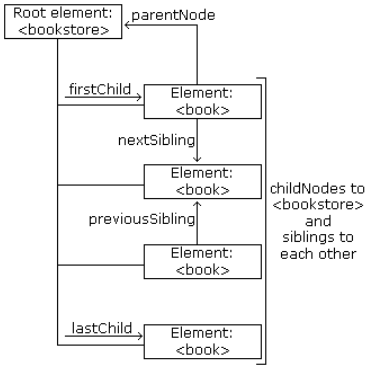
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
    <book category="cooking">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
    <book category="children">
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>
```

DOM Example 2 (cont.)

```
    <book category="web">
        <title lang="en">XQuery Kick Start</title>
        <author>James McGovern</author>
        <author>Per Bothner</author>
        <author>Kurt Cagle</author>
        <author>James Linn</author>
        <author>Vaidyanathan Nagarajan</author>
        <year>2003</year>
       <price>49.99</price>
    </book>
    <book category="web" cover="paperback">
        <title lang="en">Learning XML</title>
        <author>Erik T. Ray</author>
        <year>2003</year>
        <price>39.95</price>
     </book>
</bookstore>
```


Relationship between Nodes

- The nodes in the node tree have a hierarchical relationship to each other.
- The terms parent, child, and sibling are used to describe the relationships. Parent nodes have children. Children on the same level are called siblings (brothers or sisters).
- In a node tree, the top node is called the root
- Every node, except the root, has exactly one parent node
- A node can have any number of children
- A leaf is a node with no children
- Siblings are nodes with the same parent



Parsing the XML DOM

- Most browsers have a build-in XML parser that can be used to read and manipulate XML.
- The parser reads XML into memory and converts it into an XML DOM object that can be accesses with JavaScript.
- Microsoft's XML parser is built into IE 5 and higher.
- There are some differences between Microsoft's XML parser and the parsers used in other browsers.
  - Microsoft parser supports loading of both XML files and XML strings (text), while other browsers use separate parsers.
- However, all parsers contain functions to traverse XML trees, access, insert, and delete nodes.
- However, before an XML document can be accessed and manipulated, it must be loaded into an XML DOM object.
- In this tutorial we will show you how to create scripts that will work in both IE and other browsers.

Loading XML with Microsoft's XML Parser

- The following JavaScript fragment loads an XML document ("books.xml") into the parser:

```
xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.load("books.xml");
```

  - first line creates an empty Microsoft XML document object.
  - second line turns off asynchronized loading, to make sure that the parser will not continue execution of the script before the document is fully loaded.
  - third line tells the parser to load an XML document called "books.xml".
- The following JavaScript fragment loads a string called txt into the parser:

```
xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.loadXML(txt);
```

XML Parser in Firefox and Other Browsers

- The following JavaScript fragment loads an XML document ("books.xml") into the parser:

```
xmlDoc=document.implementation.createDocument("","",null);
xmlDoc.async="false";
xmlDoc.load("books.xml");
```

  - first line creates an empty XML document object.
  - second line turns off asynchronized loading,
  - third line tells the parser to load an XML document called "books.xml".
- The following JavaScript fragment loads a string called txt into the parser:

```
parser=new DOMParser();
xmlDoc=parser.parseFromString(txt,"text/xml");
```


Parsing an XML File - A Cross browser Example

```
<html>
<body>
<script type="text/javascript">
try //Internet Explorer
  {
  xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
  }
catch(e)
  {
  try //Firefox, Mozilla, Opera, etc.
    {
    xmlDoc=document.implementation.createDocument("","",null);
    }
  catch(e) {alert(e.message)}
  }
try
  {
  xmlDoc.async=false;
  xmlDoc.load("books.xml");
  document.write("xmlDoc is loaded, ready for use");
  }
catch(e) {alert(e.message)}
</script>
</body>
</html>
```

Error: Access Across Domains

- For security reasons, modern browsers does not allow access across domains.
  - Thus both the web page and the XML file it tries to load, must be located on the same server.
- Otherwise the xmlDoc.load() method, will generate the error "Access is denied".

Parsing an XML String - A Cross browser Example

```
<html>
<body>
<script type="text/javascript">
text="<bookstore>"
text=text+"<book>";
text=text+"<title>Everyday Italian</title>";
text=text+"<author>Giada De Laurentiis</author>";
text=text+"<year>2005</year>";
text=text+"</book>";
text=text+"</bookstore>";

try //Internet Explorer
  {
  xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
  xmlDoc.async="false";
  xmlDoc.loadXML(text);
  }
catch(e)
  {
  try //Firefox, Mozilla, Opera, etc.
    {
    parser=new DOMParser();
    xmlDoc=parser.parseFromString(text,"text/xml");
    }
  catch(e) {alert(e.message)}
  }
document.write("xmlDoc is loaded, ready for use");
</script>
</body>
</html>
```

XML DOM Load Function

- The code for loading XML documents can be stored in a function
  - can be stored in the <head> section of an HTML page, and called from a script in the page
  - can be stored in an external file to make sure the same code is used in all pages,

```
function loadXMLDoc(dname)
{
try //Internet Explorer
  {
  xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
  }
catch(e)
  {
  try //Firefox, Mozilla, Opera, etc.
    {
    xmlDoc=document.implementation.createDocument("","",null);
    }
  catch(e) {alert(e.message)}
  }
try
  {
  xmlDoc.async=false;
  xmlDoc.load(dname);
  return(xmlDoc);
  }
catch(e) {alert(e.message)}
return(null);
}
```


XML DOM Load Function

- Using an external load function stored in a file called "loadxmldoc.js“

```
<html>
<head>
<script type="text/javascript" src="loadxmldoc.js">
</script>
</head>

<body>
<script type="text/javascript">
xmlDoc=loadXMLDoc("books.xml");
document.write("xmlDoc is loaded, ready for use");
</script>
</body>
</html>
```

Node Properties

- Each node is an **object** with the following important properties:
  - nodeName
  - nodeValue
  - nodeType
  - parentNode
  - childNodes
  - firstChild
  - lastChild
  - previousSibling
  - nextSibling
  - attributes
- **Properties** are often referred to as something that is (i.e. nodename is "book"),
- Examples
  - x.nodeName - the name of x
  - x.nodeValue - the value of x
  - x.parentNode - the parent node of x
  - x.childNodes - the child nodes of x
  - x.attributes - the attributes nodes of x
- The **documentElement** property of the XML document is the root node

Node Properties (cont.)

- The nodeName property specifies the name of a node.
  - nodeName is read-only
  - nodeName of an element node is the same as the tag name
  - nodeName of an attribute node is the attribute name
  - nodeName of a text node is always #text
  - nodeName of the document node is always #document
- The nodeValue property specifies the value of a node.
  - nodeValue for element nodes is undefined
  - nodeValue for text nodes is the text itself
  - nodeValue for attribute nodes is the attribute value
  - Example1: Get the Value of an Element

    ```
    txt=x.nodeValue;
    ```
  - Example 2: Change the Value of an Element

    ```
    x.nodeValue="Easy Cooking";
    ```

Node Properties (cont.)

- The nodeType property specifies the type of node.
  - nodeType is read only.
  - A node may be a reference to an element, its attributes, or text from the document
  - The most important node types are:

Node type	NodeType
Element	1
Attribute	2
Text	3
Comment	8
Document	9

Node Methods

- **Methods** are often referred to as something that is done (i.e. delete "book")
- Common methods are
  - `getElementsByTagName` method – allows accessing an element with a specified tag name
  - `setAttribute` method – changes the attribute
  - `cloneNode` method - duplicates a node
  - `getNodeName` method - returns the node's name
  - `getNodeType` method - returns the node's type
  - `getNodeValue` method - returns the node's value
  - `getParentNode` method - returns the node's parent's name
  - `hasChildNodes` method - true if has child nodes
  - `insertBefore` method - inserts a node before a specified child node
  - `removeChild` method - removes the child node
  - `replaceChild` method - replaces one child with another
  - `setNodeValue` method - sets node's value
  - `appendChild` method - adds a child node to an existing node
  - `removeChild` method – removes a child node
  - `removeAttribute` method – removes an attribute

Accessing Nodes

- A node can be accessed by the getElementsByTagName() method
  - Syntax

    node.getElementsByTagName("*tagname*");

    returns a node list which is an array of nodes
  - Examples

    x.getElementsByTagName("title");

    returns all title elements under the x element

    x.getElementsByTagName("title");

    returns all title elements in the document

    - elements in the node list can be accessed by index number (starting from 0), e.g.

```
xmlDoc=loadXMLDoc("books.xml");
x=xmlDoc.getElementsByTagName("title");
y=x[2];              // to access the third title element
```

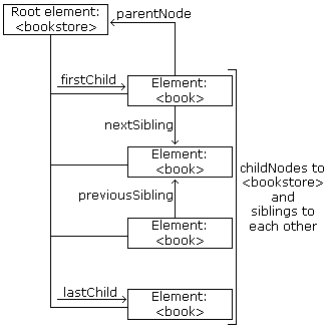

Example

- Loop through all title elements

```
xmlDoc=loadXMLDoc("books.xml");
x=xmlDoc.getElementsByTagName("title");
for (i=0;i<x.length;i++) {
  document.write(x[i].childNodes[0].nodeValue);
  document.write("<br />");
}
```

Navigating DOM Nodes

- Accessing nodes in the node tree via the relationship between nodes, is often called "navigating nodes".
- In the XML DOM, node relationships are defined as properties to the nodes:
  - parentNode
  - childNodes
  - firstChild
  - lastChild
  - nextSibling
  - previousSibling



Example 1

- The following code loops through the child nodes, that are also element nodes, of the root node

```
xmlDoc=loadXMLDoc("books.xml");

x=xmlDoc.documentElement.childNodes;

for (i=0;i<x.length;i++)
{
  if (x[i].nodeType==1)
  {//Process only element nodes (type 1)
  document.write(x[i].nodeName);
  document.write("<br />");
  }
}
```

Example 2

- Traversing the Node Tree

```
<html>
<head>
<script type="text/javascript" src="loadxmlstring.js"></script>
</head>
<body>
<script type="text/javascript">
text="<book>";
text=text+"<title>Everyday Italian</title>";
text=text+"<author>Giada De Laurentiis</author>";
text=text+"<year>2005</year>";
text=text+"</book>";

xmlDoc=loadXMLString(text);

// documentElement always represents the root node
x=xmlDoc.documentElement.childNodes;
for (i=0;i<x.length;i++)
{
document.write(x[i].nodeName);
document.write(": ");
document.write(x[i].childNodes[0].nodeValue);
document.write("<br />");
}
</script>
</body>
</html>
```

Output:

```
title: Everyday Italian
author: Giada De Laurentiis
year: 2005
```


Example 3

- Ignoring Empty Text Between Elements

```
xmlDoc=loadXMLDoc("books.xml");

x=xmlDoc.documentElement.childNodes;

for (i=0;i<x.length;i++)
{
if (x[i].nodeType==1)
  {// only process element nodes
  document.write(x[i].nodeName);
  document.write("<br />");
  }
}
```

Change the Value of an Attribute

- In the DOM, attributes are nodes.
- Unlike element nodes, attribute nodes have text values.
- The way to change the value of an attribute, is to change its text value.
- This can be done using the setAttribute() method or using the nodeValue property of the attribute node.
- Example 1 – Using setAttribute()

```
xmlDoc=loadXMLDoc("books.xml");
x=xmlDoc.getElementsByTagName('book');
x[0].setAttribute("category","food");
```


Change the Value of an Attribute (cont.)

- Example 2 - Using nodeValue

```
xmlDoc=loadXMLDoc("books.xml");
x=xmlDoc.getElementsByTagName("book")[0]
y=x.getAttributeNode("category");
y.nodeValue="food";
```

Creating XML Using DOM Methods

```
<script type="text/javascript">
var comment, stud1, stud2, studs
function mkRecord(){
  comment = document.createComment('My Tullab records');
  studs = document.createElement('tullab');
  stud1 = createStudent("G. Al-Good", 4.0);
  stud2 = createStudent("P. Al-Probation", 1.7);
  studs.appendChild(comment);
  studs.appendChild(stud1);
  studs.appendChild(stud2);
  showObject(studs);
}
function createStudent(name, gpa){
   var result = document.createElement('talib');
   nm = document.createElement('name');
   GPA = document.createElement('gpa');
   nm.appendChild(document.createTextNode(name));
   GPA.appendChild(document.createTextNode(gpa));
   result.appendChild(nm);
   result.appendChild(GPA);
   return result;
}
</script>
```


Q & A

?

References

- Some useful links with examples and other resources:
  - W3School DOM Tutorial
    - http://www.w3schools.com/dom/default.asp
  - MSXML 4.0 SDK