W3C Document Object Model

Teodor Rus

rus@cs.uiowa.edu

The University of Iowa, Department of Computer Science

The W3C DOM

- W3C DOM standardizes programming interface for handling XML documents
- The W3C DOM specifies a way of treating a document as a *tree of nodes*
- Possible node types are: element, attribute, text, CDATA section, entity, entity reference, processing instruction comment, document, document type, document fragment, notation

Example:

Tree node representation of the document:

```
<?xml version="1.0" encoding="UTF-8">
<DOCUMENT>
   <GREETING>
      Hellow from XML
   </GREETING>
   <MESSAGE>
      Welcome to the wild and woolly world of XML.
   </MESSAGE>
</DOCUMENT>
```

is in Figure 1

Figure 1

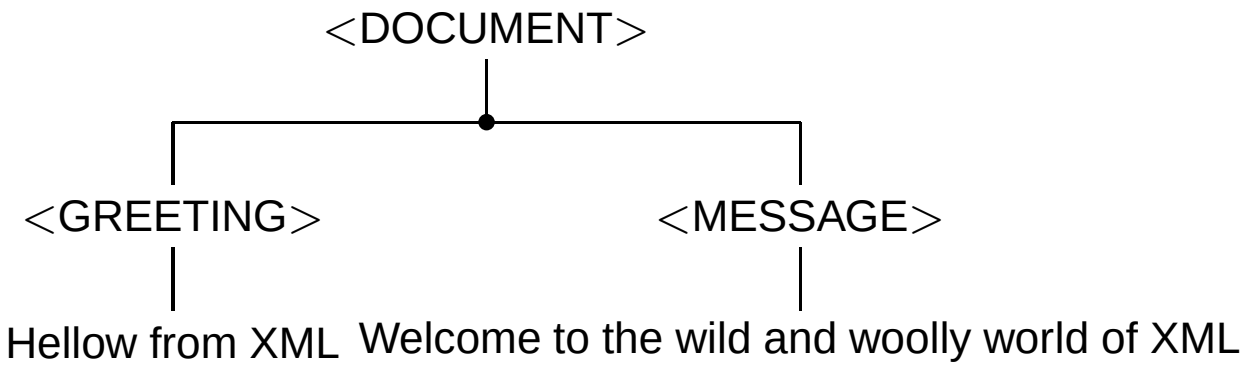

Figure 1: Tree node representation of a document

Navigating along DOM:

- DOM model supports primitive to navigate along various branches of a document tree such as `nextChild, lastSibling, firstChild`, etc.
- There are three levels of DOM:
  1. Level 0: this is DOM itself referred to by W3C
  2. Level 1: current W3C recommendation. Concentrate on HTML and XML document models
  3. Level 2: includes stylesheet object model and adds functionality for manipulating style info
  4. Level 3: still in the planning stage.

More on document levels:

- Documentation for Level 1 is available at `www.w3c.org/TR/REC-DOM-Level-1/`.
- Level 2: let the user traverse a document, has a built-in event model, support XML namespaces.
  `www.w3c.org/TR/REC-DOM-Level-2/`
- Level 3: will address document loading and saving, and content models, such as DTDs and schemas with validation support
  Will address document views and formating, key events, and event group. No documentation yet

Observations:

- Only complete implementation of XML DOM seems to be in Internet Explorer version 5 and later
- Documentation for Microsoft DOM is at:
  `http://msdn.microsoft.com/library/psdk/xmlsdk/`
- Microsoft sites is not stable, hence you may need to search for “xml dom"
- Due to Explorer support for DOM Level 1 we will use here this model

Level 1 XML DOM Objects

- `Document`: the document objects
- `DocumentFragment`: reference to <!DOCTYPE> element
- `EntityReference`: reference to an entity
- `Element`: an element
- `Attr`: an attribute
- `ProcessingInstruction`: a processing instruction
- `Comment`: content of an XML comment
- `Text`: text content of en element
- `CDATASection`: reference to CDATA section
- `Entity`: a parsed or unparsed entity in the XML document

Level 1 objects, continuation

- `Notation`: holds a notation
- `Node`: a single node in a document tree
- `NodeList`: a list of node objects (iteration and indexed access)
- `NamedNodeMap`: allows iteration and access by name to attributes

Microsoft names and objects

- Defines a set of “base objects", foundation for XML DOM
- Top-level object is `DOMDocument`, only one created directly. Other objects are reached through DOMDocument.

Base objects:

- `XMLDOMNode`, supports data types, namespaces, DTDs, XML scheme
- `XMLDOMNodeList`, iteration and indexed access
- `XMLDOMNamedNodeMap`, iteration and access by name to attributes
- `XMLDOMParseError`, error number, line number, character position text description `XMLHttpRequest`, allows communication with HTTP servers
- `XSLRuntime`, supports methods for XSL stylesheets

Other XML Dom Objects:

- `XMLDOMAttribute`: an attribute object
- `XMLDOMCDATASection`: handles CDATA sections so that text is not interpreted as markup language
- `XMLDOMCharacterData`: provides methods for text manipulation
- `XMLDOMComment`: provides the content of an XML comment
- `XMLDOMDocumentFragment`: lightweight object useful for tree insert operations
- `XMLDOMDocumentType`: holds info connected to the document type declaration
- `XMLDOMElement`: stands for the element object

Other, continuation

- `XMLDOMEntity`: a parsed or unparsed entity in XML document
- `XMLDOMEntityReference`: an entity reference node
- `XMLDOMImplementation`: supports general DOM methods
- `XMLDOMNotation`: holds a notation as declared in a DTD or schema
- `XMLDOMProcessingInstruction`: a processing instruction
- `XMLDOMText`: provides text content of an element or attribute

Observations:

- The list of objects is quite substantial and each object can contain its own properties, methods, and events
- The properties, methods, events are either specified in W3C XML DOM or are added by Microsoft
- For a good understanding of these object see textbook pages 304–318

Creating a DOM Object:

- One can create a DOMObject either using `Microsoft.XMLDOM` class or using `XML data islands`
- Using `MicrosoftXMLDOM` class: load document into the object with a load method:

```
function readXMLDocument()
  {
   var xmldoc
   xmldoc = new ActiveXObject("Microsoft.XMLDOM")
   xmldoc.load("file.xml")
   ...
  }
```

where file.xml contains the document

Note:

- To specify another version of MSXML use:

```
xmldoc = new ActiveXObject("MSXML2.DOMDocument.4.0")
```

However, not all users will have MSXML 4.0 installed.

Creating a data island

- Use <XML> HTML element and then use `XMLDocument` property of that element to gain access to the corresponding document object:

```
<XML ID="meetingsXML" SRC="file.xml"></XML>
<SCRIPT LANGUAGE="JavaScript">
  function readXMLDocument()
    {
     xmldoc = document.all("meetingsXML").XMLDocument
     ...
    }
```

- See the base set of properties for this object on pages 305–308 of the textbook

XMLDOMNodeList object

```
function readXMLDocument()
  {
   var xmldoc, nodeList
   xmldoc = new ActiveXObject("Microsoft.XMLDOM")
   xmldoc.load("file.xml")
   nodeList = xmlDoc.getElementsByTagName("PERSON")
   ...
  }
```

Properties and methods:

- `length`: number of items in `noteList`
- `item`: random access to node in `nodeList`
- `nextNode` next node in `nodeList`
- `reset` reset the list iterator

Example file.xml

```
<?xml version="1.0">
<MEETINGS>
   <MEETING TYPE="informal">
      <MEETING_TITLE> XML in the real world</MEETING_TITLE>
      <MEETING_NUMBER> 2097 </MEETING_NUMBER>
      <SUBJECT> XML </SUBJECT>
      <DATE> 6/1/2003 </DATE>
      <PEOPLE>
         <PERSON ATTENDANCE="present">
            <FIRST_NAME> Edward </FIRST_NAME>
            <LAST_NAME> Samson </LAST_NAME>
         </PERSON>
         <PERSON ATTENDANCE="absent">
            <FIRST_NAME> Ernestine </FIRST_NAME>
            <LAST_NAME> Johnson </LAST_NAME>
         </PERSON>
```

Example file.xml, continuation

```
            <PERSON ATTENDANCE="present">
               <FIRST_NAME> Betty </FIRST_NAME>
               <LAST_NAME> Richardson </LAST_NAME>
            </PERSON>
         </PEOPLE>
      </MEETING>
</MEETINGS>
```

Loading XML Documents

- Create a document object using `Microsoft.XMLDOM` class
- Use the new document object in an HTML file

```
<HTML>
  <HEAD><TITLE> Reading XML element values </TITLE></HEAD>
  <SCRIPT LANGUAGE="JavaScript">
    function readXMLDocument()
      {
       var xmldoc, meetingsNode
       xmldoc = new ActiveXObject("Microsoft.XMLDOM")
       xmldoc.load("file.xml")
       meetingsNode=xmldoc.documentElement
       ...
      }
   </SCRIPT>
</HTML>
```

Moving around the document

- Use methods such as `firstChild, nextChild, previousChild, lastChild, firstSibling, nextSibling, previousSibling, lastSibling`

```
function readXMLDocument()
  {
   var xmldoc, meetingsNode, meetingNode
   xmldoc = new ActiveXObject("Microsoft.XMLDOM")
   xmldoc.load("file.xml")
   meetingsNode=xmldoc.documentElement
   meetingNode = meetingsNode.firstChild
   ...
  }
```

Track down

- Track down the third <PERSON> element inside <PEOPLE> element
- Since <PEOPLE> element is the last child of the <MEETING> element we have:

```
var xmldoc, meetingsNode, meetingNode, peopleNode
xmldoc = new ActiveXObject("Microsoft.XMLDOM")
xmldoc.load("file.xml")
meetingsNode=xmldoc.documentElement
meetingNode = meetingsNode.firstChild
peopleNode = meetingNode.lastChild
...
```

Move around

- The third person in the <PEOPLE> is the last child.
- Hence, third person is obtained by:

```
var xmldoc, meetingsNode, meetingNode, peopleNode
var personNode
xmldoc = new ActiveXObject("Microsoft.XMLDOM")
xmldoc.load("file.xml")
meetingsNode=xmldoc.documentElement
meetingNode = meetingsNode.firstChild
peopleNode = meetingNode.lastChild
personNode = peopleNode.lastChild
...
```

More move around

```
var xmldoc, meetingsNode, meetingNode, peopleNode
var personNode, first_nameNode, last_nameNode
xmldoc = new ActiveXObject("Microsoft.XMLDOM")
xmldoc.load("file.xml")
meetingsNode=xmldoc.documentElement
meetingNode = meetingsNode.firstChild
peopleNode = meetingNode.lastChild
personNode = peopleNode.lastChild
first_nameNode = personNode.firstChild
last_nameNode = first_nameNode.nextSibling
...
```

Function manipulating text node

```
function readXMLDocument()
{
 var xmldoc, meetingsNode, meetingNode, peopleNode
 var personNode, first_nameNode, last_nameNode outputText
 xmldoc = new ActiveXObject("Microsoft.XMLDOM")
 xmldoc.load("file.xml")
 meetingsNode=xmldoc.documentElement
 meetingNode = meetingsNode.firstChild
 peopleNode = meetingNode.lastChild
 personNode = peopleNode.lastChild
 first_nameNode = personNode.firstChild
 last_nameNode = first_nameNode.nextSibling
 outputText = "Third name:"+firstNameNode.firstChild.nodeValue
              + ' ' + last_nameNode.firstChild.nodeValue
 messageDIV.innerHTML = outputText
}
```

Complete example:

```
<HTML>
  <HEAD> <TITLE> Reading XML element values </TITLE>
  <SCRIPT LANGUAGE="JavaScript">
   function readXMLDocument()
    {
     var xmldoc,meetingsNode,meetingNode,peopleNode,messageDIV
     var personNode,first_nameNode,last_nameNode,outputText
     xmldoc = new ActiveXObject("Microsoft.XMLDOM")
     xmldoc.load("file.xml")
     meetingsNode=xmldoc.documentElement
     meetingNode = meetingsNode.firstChild
     peopleNode = meetingNode.lastChild
     personNode = peopleNode.lastChild
     first_nameNode = personNode.firstChild
     last_nameNode = first_nameNode.nextSibling
     outputText="Third name:"+firstNameNode.firstChild.nodeValue
                 + ' ' + last_nameNode.firstChild.nodeValue
     messageDIV.innerHTML = outputText } </SCRIPT>
  </HEAD>
```

Example, continuation

```
<BODY>
  <CENTER>
    <H1> Reading XML element values </H1>
    <INPUT TYPE="BUTTON" VALUE="Get name of third person"
           ONCLICK="readXMLDocument()">
    <P>
    <DIV ID="messageDIV"></DIV>
  </CENTER>
</BODY>
</HTML>
```

Note: we added a button with the caption "Get ..." This button calls the JavaScript function readXMLDocument()

Using XML Data Islands

- XML Data Islands are used to embed XML documents inside HTML pages
- Explorer 5 supports and HTML <XML> element which is not part of HTML standard
- With <XML> element one can enclose an XML document as follows:

```
<XML ID="greeting">
     <DOCUMENT>
        <GREETING> Hi there XML!</GREETING>
     </DOCUMENT>
</XML>
```

Attributes of <XML> element

- `ID`: used to refer to the <XML> element in JavaScript code
- `NS`: URI of the XML namespace used by XML content Set it to an URI
- `PREFIX`: namespace prefix of the XML contents. Set it to an alphanumeric string
- `SRC`: source for the XML document if document is external. Set it to an URI

Using <XML> element

- Access <XML> element in the HTML document using its `ID` value in the code
- Use `document.all(ID.value)` to reach <XML>
- To get the document object corresponding to XML document one can further use `XMLDocument` property

Example use <XML> element

```
<HTML>
  <HEAD>
    <TITLE> Reading XML element values with data islands </TITLE>
    <XML ID="meetingsXML" SRC="file.xml"</XML>
    <SCRIPT LANGUAGE="JavaScript">
      function readXMLDocument()
         {
          var xmldoc,meetingsNode,meetingNode,peopleNode
          var messageDIV,personNode,first_nameNode,
          var last_nameNode,outputText
          xmldoc = document.all("meetingsXML).XMLDocument
          meetingsNode=xmldoc.documentElement
          meetingNode = meetingsNode.firstChild
          peopleNode = meetingNode.lastChild
          personNode = peopleNode.lastChild
          first_nameNode = personNode.firstChild
          last_nameNode = first_nameNode.nextSibling
```

Example, continuation

```
        outputText = "Third name:"+' '+
                     firstNameNode.firstChild.nodeValue+' '+
                     last_nameNode.firstChild.nodeValue
        messageDIV.innerHTML = outputText
      }
    </SCRIPT>
  </HEAD>
<BODY>
  <CENTER>
    <H1> Reading element values with XML data islands </H1>
    <INPUT TYPE="BUTTON" VALUE="Get name of third person"
           ONCLICK="readXMLDocument()">
    <P>
    <DIV ID="messageDIV"></DIV>
  </CENTER>
</BODY>
</HTML>
```

Note:

Entire XML document can be enclosed in the <XML> element. See text page 325–326 for an example

Observations:

- We used `XMLDocument` property of the object corresponding to XML data island to get the document object
- One can also use the `documentElement` property of the data island directly to get the root element of the XML document

Using documentElement

```
<HTML>
  <HEAD>
    <TITLE>
       Reading XML element values with XML data islands
    </TITLE>
    <XML ID="meetingsXML" SRC="file.xml"></XML>
    <SCRIPT LANGUAGE="JavaScript">
      function readXMLDocument()
         {
          var xmldoc,meetingsNode,meetingNode,peopleNode
          var messageDIV,personNode,first_nameNode,
          var last_nameNode,outputText
          meetingsNode=meetingsXML.documentElement
          meetingNode = meetingsNode.firstChild
          peopleNode = meetingNode.lastChild
          personNode = peopleNode.lastChild
          first_nameNode = personNode.firstChild
          ...
</HTML>
```