XML DOM & Parser

The Document Object Model (DOM) is a W3C standard. It defines a standard for accessing documents like HTML and XML.

The XML DOM is a standard for how to get, change, add, or delete XML elements. Every XML DOM contains information in hierarchical units called Nodes and the DOM describes these nodes and the relationship between them. Hence. we may say that the information is stored in a Node Tree which the developer can navigate looking for specific information.

The most common types of nodes in XML DOM are −

Document Node − Complete XML document structure is a document node.

Element Node − Every XML element is an element node. This is also the only type of node that can have attributes.

Attribute Node − Each attribute is considered an attribute node. It contains information about an element node, but is not actually considered to be children of the element.

Text Node − The document texts are considered as text node. It can consist of more information or just white space.

Consider the following xml file and its DOM tree:

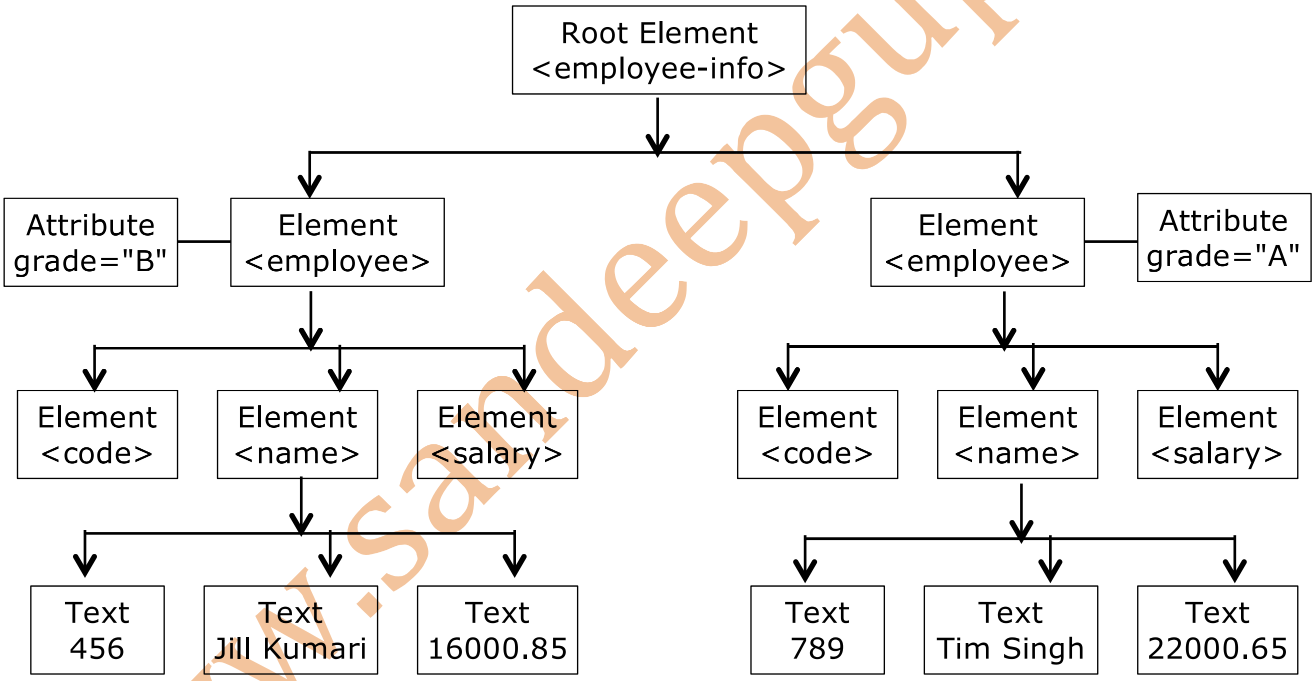
```
<?xml version = "1.0" encoding = "UTF-8"?>

<employee-info>

   <employee grade="B">
     <code>456</code>
     <name>Jill Kumari</name>
     <salary>16000.85</salary>
   </employee>

   <employee grade="A">
     <code>789</code>
     <name>Tim Singh</name>
     <salary>22000.65</salary>
   </employee>

</employee-info>
```



It is quite clear that according to the XML DOM, everything in an XML document is a node:

- The root element is the root node
- Every XML element is an element node
- The text in the XML elements are text nodes
- Every attribute is an attribute node

All major browsers have a built-in XML parser to access and manipulate XML. This built-in XML parser can convert text into an XML DOM object. Some common XML parsers are:

Sr. No.	Parser
1	**JAXP** Sun Microsystem's Java API for XML Parsing (JAXP)
2	**msxml** Microsoft's XML parser (msxml) version 2.0 is built-into Internet Explorer 5.5
3	**4DOM** 4DOM is a parser for the Python programming language
4	**XML::DOM** XML::DOM is a Perl module to manipulate XML documents using Perl

In JavaScript, we use the function DOMParser() to instantiate a new parser. Then we use the parseFromString() method to convert the xml text into xml document object. Given below is an example:

```
var xmlText = ajaxRequest.responseText;
var parser = new DOMParser();
var xmlDom = parser.parseFromString(xmlText,"text/xml");
```

employee.xml

```
<?xml version = "1.0" encoding = "UTF-8"?>

<employee-info>

   <employee grade="A">
     <code>123</code>
     <name>Jack Lal</name>
     <salary>28000.75</salary>
   </employee>

   <employee grade="B">
     <code>456</code>
     <name>Jill Kumari</name>
     <salary>16000.85</salary>
   </employee>

   <employee grade="A">
     <code>789</code>
     <name>Tim Singh</name>
     <salary>22000.65</salary>
   </employee>

  <employee grade="A">
     <code>321</code>
     <name>Ana Kadam</name>
     <salary>18000.45</salary>
  </employee>

</employee-info>
```

xmldom.html

```
<!DOCTYPE html>
<html>
<head>
<title>XML DOM and Parser</title>
<script>
var  ajaxRequest = new XMLHttpRequest();

function process()
{
        ajaxRequest.open('GET', 'employees.xml', true);
        ajaxRequest.onreadystatechange = handleServerResponse;
        ajaxRequest.send(null);
}

function handleServerResponse()
{       ...
        ...
}

</script>
</head>
<body  onload= "process()" >
<h2>Employees Details</h2>
<p id="results"></p>
</body>
</html>
```

XML XPath

XPath stands for XML Path Language. XPath is an official recommendation of the World Wide Web Consortium (W3C). It defines a language to find information in an XML file. XPath uses path expressions to select nodes or node-sets in an XML document. Xpath has around 200 built-in functions. XPath expressions can be used in JavaScript, Java, PHP, Python, C and C++, and lots of other languages.

Selecting Nodes

XPath uses path expressions to select nodes in an XML document. The node is selected by following a path or steps. The most useful path expressions are listed below:

Expression	Description
nodename	Selects all nodes with the name "*nodename*"
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

In XPath we can also use operators like +, -, *, div, mod, =, !=, <, <=, >, >=, or, and.

```
<!DOCTYPE html>
<html>
<head>
        <title>XPATH</title>
        <script>
                var  ajaxRequest = new XMLHttpRequest();

                function process()
                {
                        ajaxRequest.open('GET', 'employee.xml', true);
                        ajaxRequest.onreadystatechange = handleServerResponse;
                        ajaxRequest.send(null);
                }

                function handleServerResponse()
                {
                        ...
                }
        </script>
        </head>
        <body  onload= "process()">
                <h2>Employees Details</h2>
                <p id="results"></p>
        </body>
</html>
```

XSL

XSL stands for Extensible Stylesheet Language. In case of HTML document, tags are predefined such as table, div etc and the browser knows how to add style to them and display those using CSS styles. But in case of XML documents, tags are not predefined. In order to understand and style an XML document, World Wide Web Consortium (W3C) developed XSL which can act as XML based Stylesheet Language. An XSL document specifies how a browser should render an XML document.

Following are the main parts of XSL −

XSLT − used to transform XML document into various other types of document.

XPath − used to navigate XML document.

XSL-FO − used to format XML document. (discontinued in 2013)

What is XSLT

XSLT, Extensible Stylesheet Language Transformations, provides the ability to transform XML data from one format to another automatically.

How XSLT Works

An XSLT stylesheet is used to define the transformation rules to be applied on the target XML document. XSLT stylesheet is written in XML format. XSLT Processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML or text format. is to be displayed to the end-user.

Advantages

1. Independent of programming. Transformations are written in a separate xsl file which is again an XML document.

2. Output can be altered by simply modifying the transformations in xsl file. No need to change any code. So Web designers can edit the stylesheet and can see the change in the output quickly.

Important XSLT tags

<xsl:value-of> tag puts the value of the selected node as per XPath expression, as text.

<xsl:for-each> tag applies a template repeatedly for each node.

<xsl:if> tag specifies a conditional test against the content of nodes.

The following operators can be used:

=	(equal)
!=	(not equal)
&lt;	less than
&gt;	greater than

employee1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="employee1.xsl"?>
<employee-info>

  <employee grade="A">
    <code>123</code>
    <name>Jack Lal</name>
    <salary>28000.75</salary>
  </employee>

  <employee grade="B">
    <code>456</code>
    <name>Jill Kumari</name>
    <salary>16000.85</salary>
  </employee>

  <employee grade="A">
    <code>789</code>
    <name>Tim Singh</name>
    <salary>22000.65</salary>
  </employee>

 <employee grade="A">
    <code>321</code>
    <name>Ana Kadam</name>
    <salary>18000.45</salary>
 </employee>

</employee-info>
```

employee1.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">

 <html>
 <body>
  <h2>Employee Details</h2>
  <table border="1">
   <tr bgcolor="lightblue">
    <th>Code</th>
    <th>Name</th>
    <th>Salary</th>
   </tr>
   <xsl:for-each select="employee-info/employee">
          <xsl:sort select="name"/>
          <tr>
                <td><xsl:value-of select="code" /></td>
                <td><xsl:value-of select="name" /></td>
                <td><xsl:value-of select="salary" /></td>
          </tr>
   </xsl:for-each>
  </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

employee2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="employee2.xsl"?>
<employee-info>

   <employee grade="A">
     <code>123</code>
     <name>Jack Lal</name>
     <salary>28000.75</salary>
   </employee>

   <employee grade="B">
     <code>456</code>
     <name>Jill Kumari</name>
     <salary>16000.85</salary>
   </employee>

   <employee grade="A">
     <code>789</code>
     <name>Tim Singh</name>
     <salary>22000.65</salary>
   </employee>

  <employee grade="A">
     <code>321</code>
     <name>Ana Kadam</name>
     <salary>18000.45</salary>
  </employee>

</employee-info>
```

employee2.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>Employee Details</h2>
    <table border="1">
      <tr bgcolor="lightblue">
        <th>Code</th>
        <th>Name</th>
        <th>Salary</th>
      </tr>
      <xsl:for-each select="employee-info/employee">
                  <xsl:if test="salary &gt; 20000">
                           <tr>
                                     <td><xsl:value-of select="code" /></td>
                                     <td><xsl:value-of select="name" /></td>
                                     <td><xsl:value-of select="salary" /></td>
                           </tr>
                  </xsl:if>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

XML DTD

The XML Document Type Declaration, commonly known as DTD, is used to define the structure of an XML document. An XML document with correct syntax is called "Well Formed". An XML document validated against a DTD is both "Well Formed" and "Valid". An XML DTD can be either specified inside the document, or it can be kept in a separate document and then linked with xml file.

1) Internal DTD

A DTD is referred to as an internal DTD if its elements are declared within the XML file. To refer it as internal DTD, 'standalone' attribute in XML declaration must be set to yes. This means, the declaration works independent of an external source. Following is the syntax of internal DTD –

<!DOCTYPE root-element [element-declarations]>

where *root-element* is the name of root element and *element-declarations* is where you declare the elements.

Following is a simple example of xml file with internal DTD –

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
<!DOCTYPE address [
  <!ELEMENT address (name,company,phone)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT company (#PCDATA)>
  <!ELEMENT phone (#PCDATA)>
]>

<address>
  <name>Sandeep Gupta</name>
  <company>Study Circle</company>
  <phone>9821882868</phone>
</address>
```

Let us go through the above code –

Start Declaration – Begin the XML declaration with the following XML header

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
```

DTD – Immediately after the XML header, the *document type declaration* follows. In the statement

```
<!DOCTYPE address [
```

'DOCTYPE' informs the parser that a DTD is associated with this XML document.

DTD Body – The DOCTYPE declaration is followed by body of the DTD, where you declare elements.

```
<!ELEMENT address (name,company,phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone_no (#PCDATA)>
```

<!ELEMENT name (#PCDATA)> defines the element *name* to be of type "#PCDATA". Here #PCDATA means parse-able text data.

End Declaration − Finally, the declaration section of the DTD is closed using a closing bracket and a closing angle bracket. This effectively ends the definition, and thereafter, the XML document follows immediately.

Rules

- The document type declaration must appear at the start of the document (preceded only by the XML header) − it is not permitted anywhere else within the document.
- Similar to the DOCTYPE declaration, the element declarations must start with an exclamation mark.

2) External DTD

In external DTD, elements are declared outside the XML file. They are accessed by specifying a .dtd file or a valid URL. To refer it as external DTD, standalone attribute in the XML declaration must be set as 'no'. This means, declaration includes information from the external source. Following is the syntax for external DTD −
<!DOCTYPE root-element SYSTEM "file-name">
where file-name is the file with .dtd extension.

The following example shows external DTD usage −

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "no" ?>
<!DOCTYPE address SYSTEM "address.dtd">
<address>
   <name>Sandeep Gupta</name>
   <company>Study Circle</company>
   <phone>9821882868</phone>
</address>
```

The content of the DTD file address.dtd is as shown –

```
<!ELEMENT address (name,company,phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
```

XHTML

XHTML stands for eXtensible HyperText Markup Language. XHTML was developed by combining the strengths of HTML and XML. XHTML is almost identical to HTML 4.01 with only few differences. XHTML is a cleaner and stricter version of HTML 4.01.

Why use XHTML ??

Many pages on the internet contain "bad" HTML.
This HTML code works fine in most browsers (even if it does not follow the HTML rules):

```
<body>
  <h1>This is a heading
  <p>This is a paragraph
</body>
```

However, some browsers may not be able to interpret the above mark-up correctly. Hence it is advised to make our HTML document compliant to XHTML after which all browsers can correctly interpret our document.

Here are the important points to remember while writing a new XHTML document or converting existing HTML document into XHTML document –

1. Write a DOCTYPE declaration at the start of the XHTML document.
2. Write all XHTML tags and attributes in lower case only.
3. Close all XHTML tags properly.
4. Nest all the tags properly.
5. Quote all the attribute values.
6. Forbid Attribute minimization.
7. Replace the name attribute with the id attribute.
8. Deprecate the language attribute of the script tag.

Here is the detail explanation of the above XHTML rules –

DOCTYPE Declaration

All XHTML documents must have a DOCTYPE declaration at the start.

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

Case Sensitivity

XHTML is case sensitive markup language. All the XHTML tags and attributes need to be written in lower case only.

```
<!-- This is invalid in XHTML -->
<A Href="www.study-circle.org">Study Circle</A>

<!-- Correct XHTML way of writing this is as follows -->
<a href="www.study-circle.org">Study Circle</a>
```

Closing the Tags

Each and every XHTML tag should have an equivalent closing tag, even empty elements should also have closing tags. Here is an example showing valid and invalid ways of using tags −

```
<!-- This is invalid in XHTML -->
<p>This paragraph is not written according to XHTML syntax.

<!-- This is also invalid in XHTML -->
<img src="/images/xhtml.gif" >
```

The following syntax shows the correct way of writing above tags in XHTML. Difference is that, here we have closed both the tags properly.

```
<!-- This is valid in XHTML -->
<p>This paragraph is not written according to XHTML syntax.</p>

<!-- This is also valid now -->
<img src="/images/xhtml.gif" />
```

Attribute Quotes

All the values of XHTML attributes must be quoted. Otherwise, your XHTML document is assumed as an invalid document. Here is the example showing syntax −

```
<!-- This is invalid in XHTML -->
<img src="/images/xhtml.gif" width=250 height=50 />

<!-- Correct XHTML way of writing this is as follows -->
<img src="/images/xhtml.gif" width="250" height="50" />
```

Attribute Minimization

XHTML does not allow attribute minimization. It means you need to explicitly state the attribute and its value. The following example shows the difference −

```
<!-- This is invalid in XHTML -->
<option selected>

<!-- Correct XHTML way of writing this is as follows -->
<option selected="selected">
```

The id Attribute

The id attribute replaces the name attribute. Instead of using name = "name", XHTML prefers to use id = "id". The following example shows how –

```
<!-- This is invalid in XHTML -->
<img src="/images/xhtml.gif" name="xhtml_logo" />

<!-- Correct XHTML way of writing this is as follows -->
<img src="/images/xhtml.gif" id="xhtml_logo" />
```

The language Attribute

The language attribute of the script tag is deprecated. The following example shows this difference –

```
<!-- This is invalid in XHTML -->

<script language="JavaScript" type="text/JavaScript">
  document.write("Hello XHTML!");
</script>

<!-- Correct XHTML way of writing this is as follows -->

<script type="text/JavaScript">
  document.write("Hello XHTML!");
</script>
```

Nested Tags

You must nest all the XHTML tags properly. Otherwise your document is assumed as an incorrect XHTML document. The following example shows the syntax –

```
<!-- This is invalid in XHTML -->
<b><i> This text is bold and italic</b></i>

<!-- Correct XHTML way of writing this is as follows -->
<b><i> This text is bold and italic</i></b>
```

Element Prohibitions

Element	Prohibition
<a>	Must not contain other <a> elements.
<pre>	Must not contain the <img>, <object>, <big>, <small>, <sub>, or <sup> elements.
<button>	Must not contain the <input>, <select>, <textarea>, <label>, <button>, <form>, <fieldset>, <iframe> or <isindex> elements.
<label>	Must not contain other <label> elements.
<form>	Must not contain other <form> elements.

DHTML

DHTML is NOT a language or a web standard.

DHTML is a term used to describe the technologies that make web pages dynamic and interactive.

Dynamic HTML or DHTML, is an umbrella term for a collection of technologies used together to create interactive and animated websites by using a combination of a static markup language (such as HTML), a client-side scripting language (such as JavaScript), a presentation definition language (such as CSS), and the Document Object Model (DOM).