Title: Implementation of XML Parser and XML DOM (Document Object Model).

Aims:

- Understanding XML Parser.
- Getting knowledge of XML DOM.
- Syntax, rules, and structure of XML DOM.
- Implementation of XML DOM.

XML Parser

- The XML DOM (Document Object Model) defines the properties and methods for **accessing** and **editing** XML.
- However, to do the tasks, the **XML file** must be loaded to the **XML DOM object**.
- **XML Parser** is **used** to load the respective XML file to the XML DOM object.
- All major browsers have a built-in XML parser.

The code given below is loading an XML document to a DOM object using XML Parser:

```
<html>
<body>

<p id = "txtNotice"></p>

<script>

        var text, parser, xmlDoc;

        text =  "<bookstore><book>" +
                "<title>Everyday Italian</title>" +
                "<author>Giada De Laurentiis</author>" +
                "<year>2005</year>" +
                "</book></bookstore>";

        parser = new DOMParser();

        xmlDoc = parser.parseFromString(text, "text/xml");



</script>

</body>
</html>
```

XML DOM (Document Object Model)

- DOM stands for Document Object Model.
- The XML DOM defines a standard way for accessing and manipulating XML documents.
- It presents an XML document as a tree-structure.
- Saying conveniently, XML DOM is a standard for how to get, change, add or delete XML elements.
- The XML DOM views an XML document as a tree-structure. The tree structure is called a **node-tree.**
- The tree content can be modified or deleted, and new elements can be created.
- All XML elements can be accessed through the XML DOM.
- XML DOM is a standard programming interface for XML.
- XML DOM is a platform and language independent.

1. Get the value of an XML elements

The following code retrieves the text value of the first <title> element in an XML element.

```
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
```

2. Loading an XML String

The following program loads a text string into an XML DOM object and extracts the info from it with JavaScript programming.

```
<html>
<body>

<p id = "txtNotice"></p>

<script>
        var text, parser, xmlDoc;

        text =  "<bookstore><book>" +
                "<title>Everyday Italian</title>" +
                "<author>Giada De Laurentiis</author>" +
                "<year>2005</year>" +
                "</book></bookstore>";
        parser = new DOMParser();

        xmlDoc = parser.parseFromString(text, "text/xml");

        document.getElementById("txtNotice").innerHTML =
        xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;

</script>

</body>
</html>
```

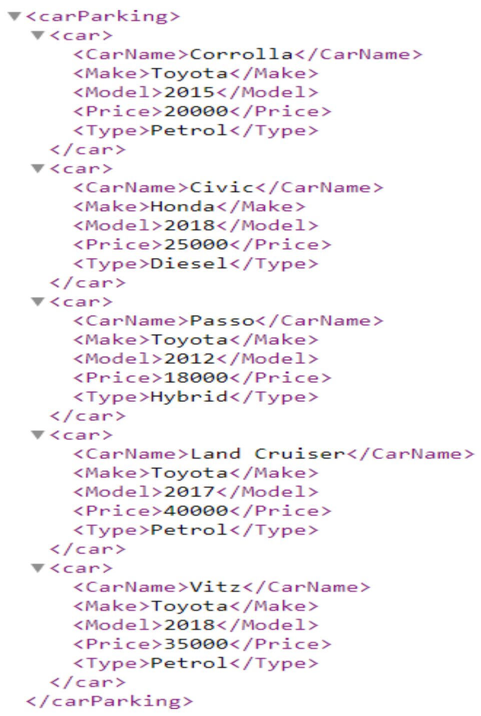
Exercise - 01

Use the above code and insert another book element to the XML file (In the text variable). Display all two book details in a html table. The Structure is given below.

Title	Author	Year
Everyday Italian	Giada De Laurentiis	2005
Java Programming	John Willson	2014

Exercise - 02

Look at the XML code below and convert it into the text format. Use the XML parser and convert it into XML DOM. Finally, display all five elements (cars) in a html table similar to the below table.



```
▼<carParking>
  ▼<car>
     <CarName>Corrolla</CarName>
     <Make>Toyota</Make>
     <Model>2015</Model>
     <Price>20000</Price>
     <Type>Petrol</Type>
   </car>
  ▼<car>
     <CarName>Civic</CarName>
     <Make>Honda</Make>
     <Model>2018</Model>
     <Price>25000</Price>
     <Type>Diesel</Type>
   </car>
  ▼<car>
     <CarName>Passo</CarName>
     <Make>Toyota</Make>
     <Model>2012</Model>
     <Price>18000</Price>
     <Type>Hybrid</Type>
   </car>
  ▼<car>
     <CarName>Land Cruiser</CarName>
     <Make>Toyota</Make>
     <Model>2017</Model>
     <Price>40000</Price>
     <Type>Petrol</Type>
   </car>
  ▼<car>
     <CarName>Vitz</CarName>
     <Make>Toyota</Make>
     <Model>2018</Model>
     <Price>35000</Price>
     <Type>Petrol</Type>
   </car>
 </carParking>
```

CarName	Make	Model	Price	Type
Corrolla	Toyota	2015	20000	Petrol
Civic	Honda	2018	25000	Diesel
Passo	Toyota	2012	18000	Hybrid
Land Cruiser	Toyota	2017	40000	Petrol
Vitz	Toyota	2018	35000	Petrol

3. XML DOM Properties

There are some typical DOM properties, such as:

- x.nodeName – the name of node object **x**
- x.nodeValue – the value of node object **x**
- x.parentNode – the parent node of node object **x**
- x.childNodes – child nodes of node object **x**
- x.attributes – the attribute nodes of node object **x**

4. XML DOM Methods

- x.getElementsByTagName(*name*) - get all elements with a specified tag name
- x.appendChild(*node*) - insert a child node to x
- x.removeChild(*node*) - remove a child node from x

5. XML DOM Nodes

According to the XML DOM, everything in an XML document is **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.

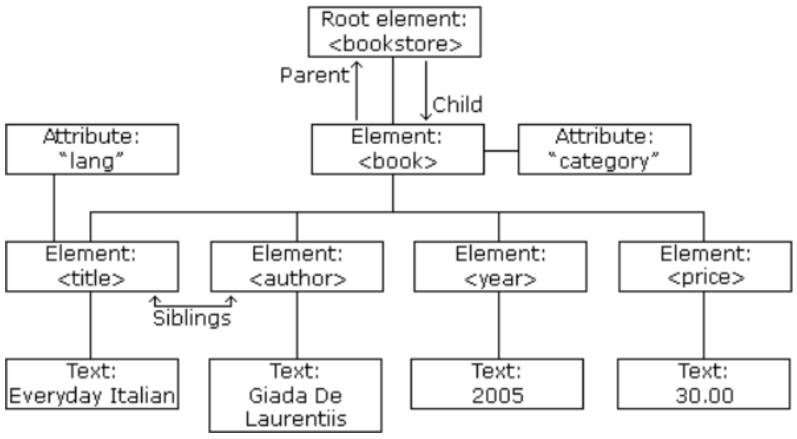
Look at the following example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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">XQuery Kick Start</title>
    <author>James McGovern</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>
```

In the above example:

- Root node is **<bookstore>**
- The root node holds 4 <book> nodes.
- Each <book> node holds 4 child nodes such, <title>, <author>, <year>, and <price>
- The child node contains one text node each “Everyday Italian”, “Giada De Laurentiis”, “2005”, “30.00”

6. XML DOM Node Tree



7. Node parent, children, and siblings

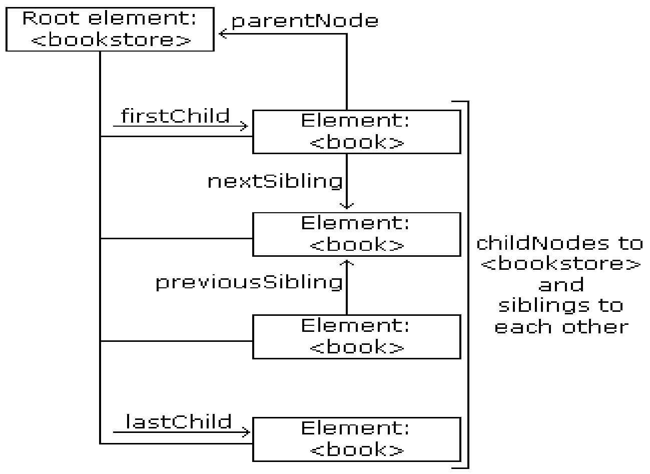


Fig 01: Node tree and the relationship between the nodes.

The nodes in the node tree have a hierarchical relationship to each other. The parent, child and sibling are used to describe the relationship.
Note: XML data is structured in a tree form, it can be traversed without knowing the exact structure of the tree and without knowing the type of data contained within.

8. Accessing Nodes

With DOM you can access every node in an XML document. The access can be done in three ways,

1. By using the **getElementsByTagName()** method.
2. By looping through (traversing) the nodes tree.
3. By navigating the node tree, using the node relationships.

1. getElementsByTagName() Method

❖ getElementsByTagName() returns all elements with a specified tag name.

Syntax: - *node*.getElementsByTagName(*"tagname"*);

9. DOM Node List

❖ The getElementsByTagName() method returns a node list. A node list is an array of nodes.

x = xmlDoc.getElementsByTagName("title");

❖ The <title> elements in x can be accessed by index number. To access the third <title> you can write

y = x[2];

❖ Nodes in the node list are accessed by arrays start from 0.

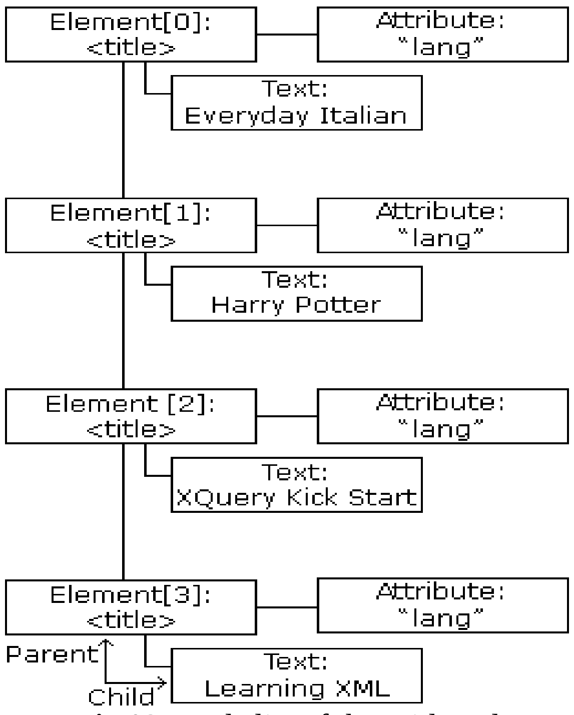


Fig 02: Node list of the <title> element.

❖ If we need first<title> element in the node list(x)

var txt = x[0].childNodes[0].nodeValue;

DOM Node List Length

❖ The length property defines the length of a node list

```
x = xmlDoc.getElementsByTagName("title").length;
```

10. Node Types

1. **documentElement** property of the XML document is the root node.
2. **nodeName** property of a node is the name of the node.
3. **nodeType** property of a node is the type of the node.

XML DOM Node Information

- The nodeName, nodeValue, and nodeType properties contain information about nodes.
- In the XML DOM, each node is an **object.** Objects have methods and properties, that can be accessed and manipulated by programming language.

Three important node properties are:

- nodeName
- nodeValue
- nodeType

01. nodeName property

It specifies the name of a node. This name property has some rules.

- 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.

02. nodeValue property

It specifies the value of a node.

- nodeValue for each element is undefined.
- nodeValue for text value is the text itself.
- nodeVlaue for attributes is the attribute value.

03. nodeType property

The nodeType property specifies the type of the node. nodeType is read-only.

DOM Attribute List

The attributes property of an element node returns a list of attribute nodes.

```
x = xmlDoc.getElementsByTagName('book')[0].attributes
```

Above code returns a list of attribute nodes from the first<book> element. If we need attribute list length use,

```
var v = x.lenght
```

Task:

Create an XML document and read the XML file in HTML document using Javascript and Implement all the above.