Experiment No. : 15

Title : Navigating the Document Object Model tree.

Objectives:

1. To understand DOM structure.
2. To understand the representation of xml document using DOM.
3. To learn DOM interfaces and methods defined in the interfaces to navigate DOM tree.

Theory:

The Document Object Model (DOM) is an and platform-independent application programming interface (API) for valid HTML and well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. XML presents the data as documents, and the DOM may be used to manage this data.

The Document Object Model (DOM), defined by the W3C DOM Working Group, is a set of interfaces for building an object representation, in the form of a tree, of a parsed XML document. Once the DOM tree is build, it can be manipulated with DOM methods such as insert and remove.

Important differences between DOM and SAX Parser-

1. Read and write access to document.
2. Entire document is parsed into the main memory.
3. Random access to entire contents of the document.

DOM builds an object representation of the document and manipulate it in memory, adding a new element or deleting an existing one. The name "Document Object Model" was chosen because it is an "object model", in the traditional object oriented design sense: documents are modelled using objects, and the model encompasses not only the structure of a document, but also the behaviour of a document and the objects of which it is composed.

DOM Levels:

The DOM is broken into two parts: 1) DOM Core Level 1 2) DOM core level 2

The W3C DOM provides a standard set of objects for HTML and XML documents, and a standard interface for accessing and manipulating them.

The DOM Structure Model

The DOM represents documents as a hierarchy of **Node** objects

Nodes: According to the DOM, everything in an XML document is a node.

The DOM says that:

- The entire document is a document node
- Every XML tag is an element node
- The texts contained in the XML elements are text nodes
- Every XML attribute is an attribute node
- Comments are comment nodes

Node Hierarchy

Nodes have a hierarchical relationship to each other. All nodes in an XML document form a document tree (or node tree). Each element, attribute, text, etc. in the XML document represents a node in the tree. The tree starts at the document node and continues to branch out until it has reached all text nodes at the lowest level of the tree.

The terms "parent" and "child" are used to describe the relationships between nodes. Some nodes may have child nodes, while other nodes do not have children (leaf nodes).

Following are the node types defined in the DOM

NodeType	Node name
1	ELEMENT_NODE
2	ATTRIBUTE_NODE
3	TEXT_NODE
4	CDATA_SECTION_NODE
5	ENTITY_REFERENCE_NODE
6	ENTITY_NODE
7	PROCESSING_INSTRUCTION_NODE
8	COMMENT_NODE
9	DOCUMENT_NODE
10	DOCUMENT_TYPE_NODE
11	DOCUMENT_FRAGMENT_NODE
12	NOTATION_NODE

Node Information

Every node has some properties that contain some information about the node.
The properties are:

- nodeName
- nodeValue
- nodeType

The DOM also specifies a **NodeList** interface to handle ordered lists of Nodes, such as the children of a Node, or the elements returned by the getElementsByTagName method of the Element interface, and also a **NamedNodeMap** interface to handle unordered sets of nodes referenced by their name attribute, such as the attributes of an Element.

Key Terms: DOM, DOM interfaces, Document Object, Root element, Node, NodeList

Algorithm:

1. create dom object representing the xml document.

   ```
   DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
   DocumentBuilder builder = factory.newDocumentBuilder();
   Document document = builder.parse("priceList.xml");
   ```

2. get the document root element by using document's getDocumentElement( ).

3. Traverse the document using dom's root elements as root node and using following Node interface methods.
    a. NodeList getChildNodes( ).
    b. String getNodeName( ).
    c. Boolean hasChildNodes( ).
4. Use following NodeList interface methods to find no.of children in a node and to access nodes in the node list.
    a. int getLength( ).
    b. Node item(int index).