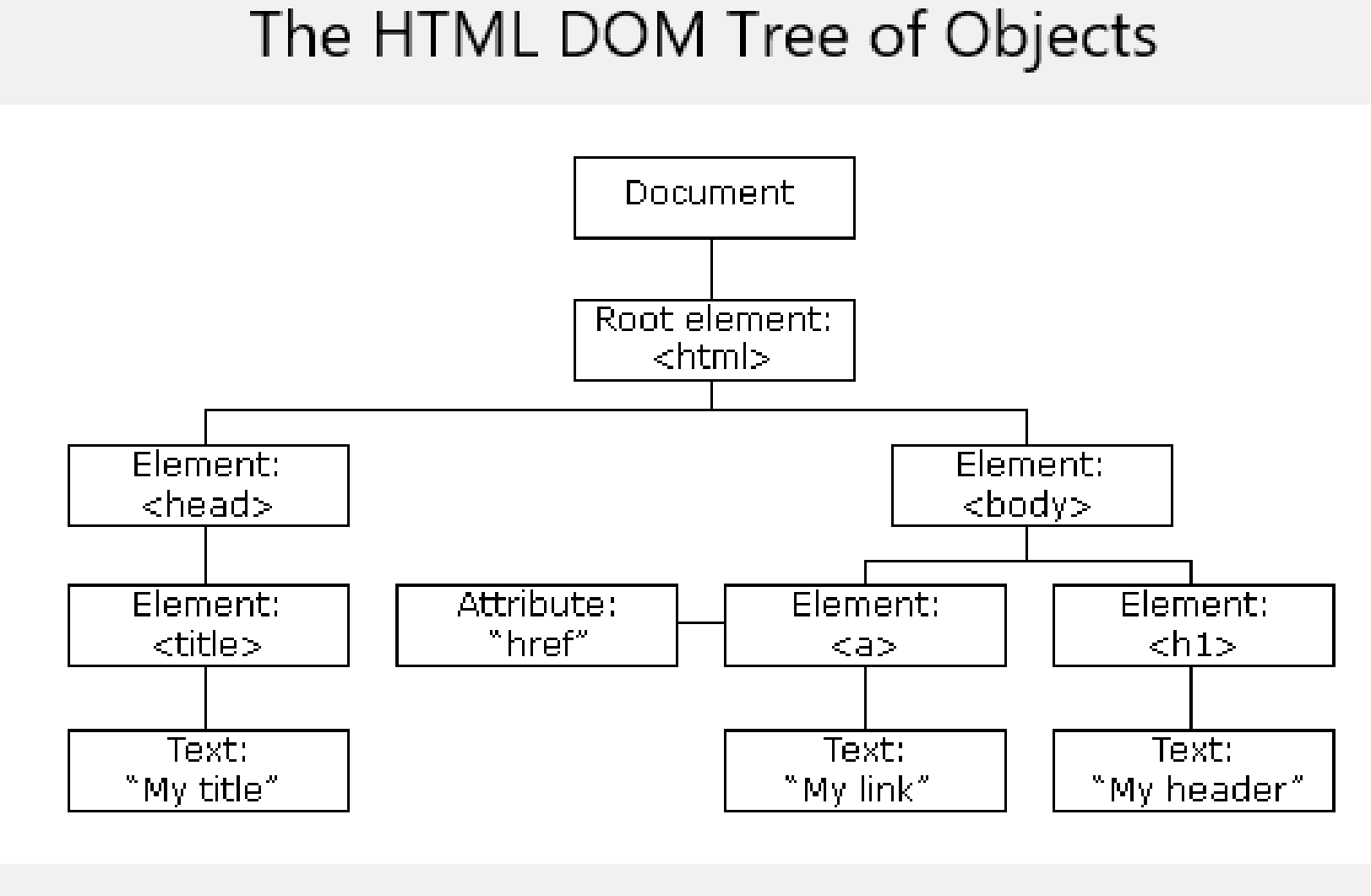
Document Object Model (DOM™)

- The W3C DOM standard is separated into 3 different parts:
- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

- **HTML DOM (Document Object Model).**
- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**:

- **The HTML DOM Tree of Objects**



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

```
<html>
<body>
<h1>My First Page</h1>
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

Output:

My First Page

Hello World!

Finding HTML Elements

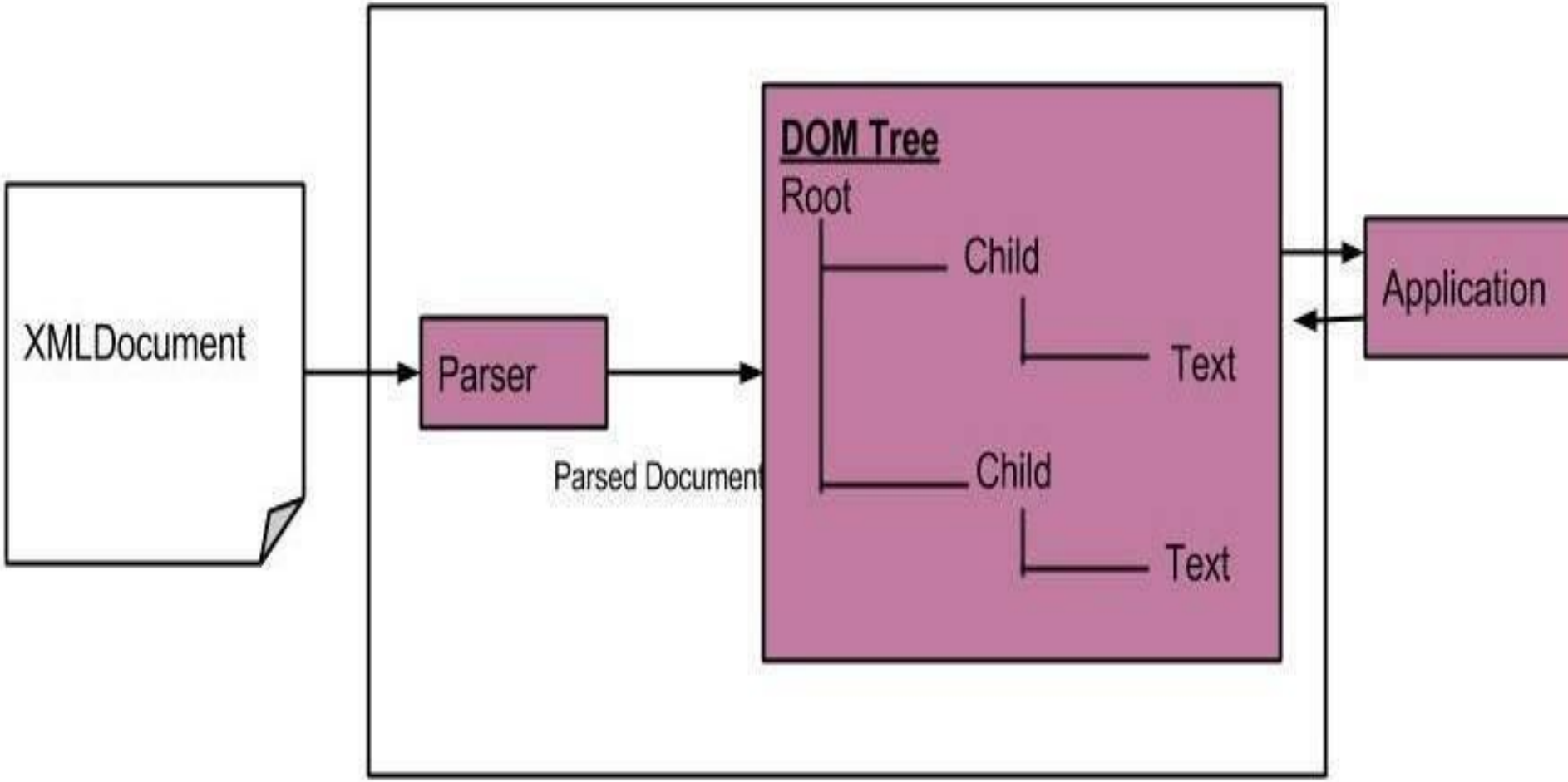
Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements

Method	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element

DOM is a way of describing the nodes and the relationships between them.

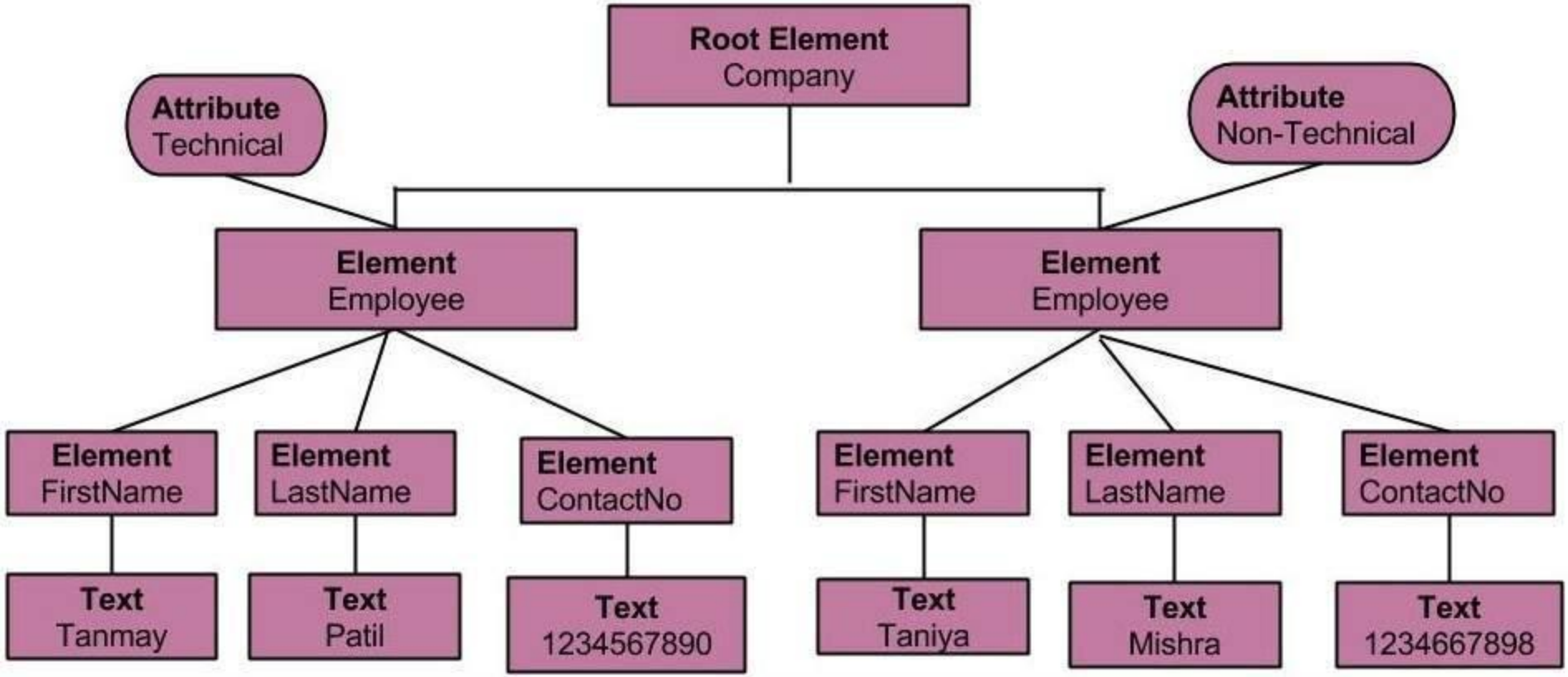
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*
- A DOM Document is a collection of nodes or pieces of information organized in a hierarchy. This hierarchy allows a developer to navigate through the tree looking for specific information. Because it is based on a hierarchy of information, the DOM is said to be tree based.
- The XML DOM, on the other hand, also provides an API that allows a developer to add, edit, move, or remove nodes in the tree at any point in order to create an application.


XMLDocument
Parser
Parsed Document
DOM Tree
Root
Child
Text
Child
Text
Application

Xml and its DOM tree node structure:

```
<?xml version="1.0"?>
 <Company>
 <Employee category="technical">
<FirstName>Tanmay</FirstName>
<LastName>Patil</LastName>
<ContactNo>1234567890</ContactNo> </Employee>
<Employee category="non-technical">
<FirstName>Taniya</FirstName>
<LastName>Mishra</LastName>
<ContactNo>1234667898</ContactNo> </Employee>
</Company>
```

Document Object Model of the above XML document would be as follows:

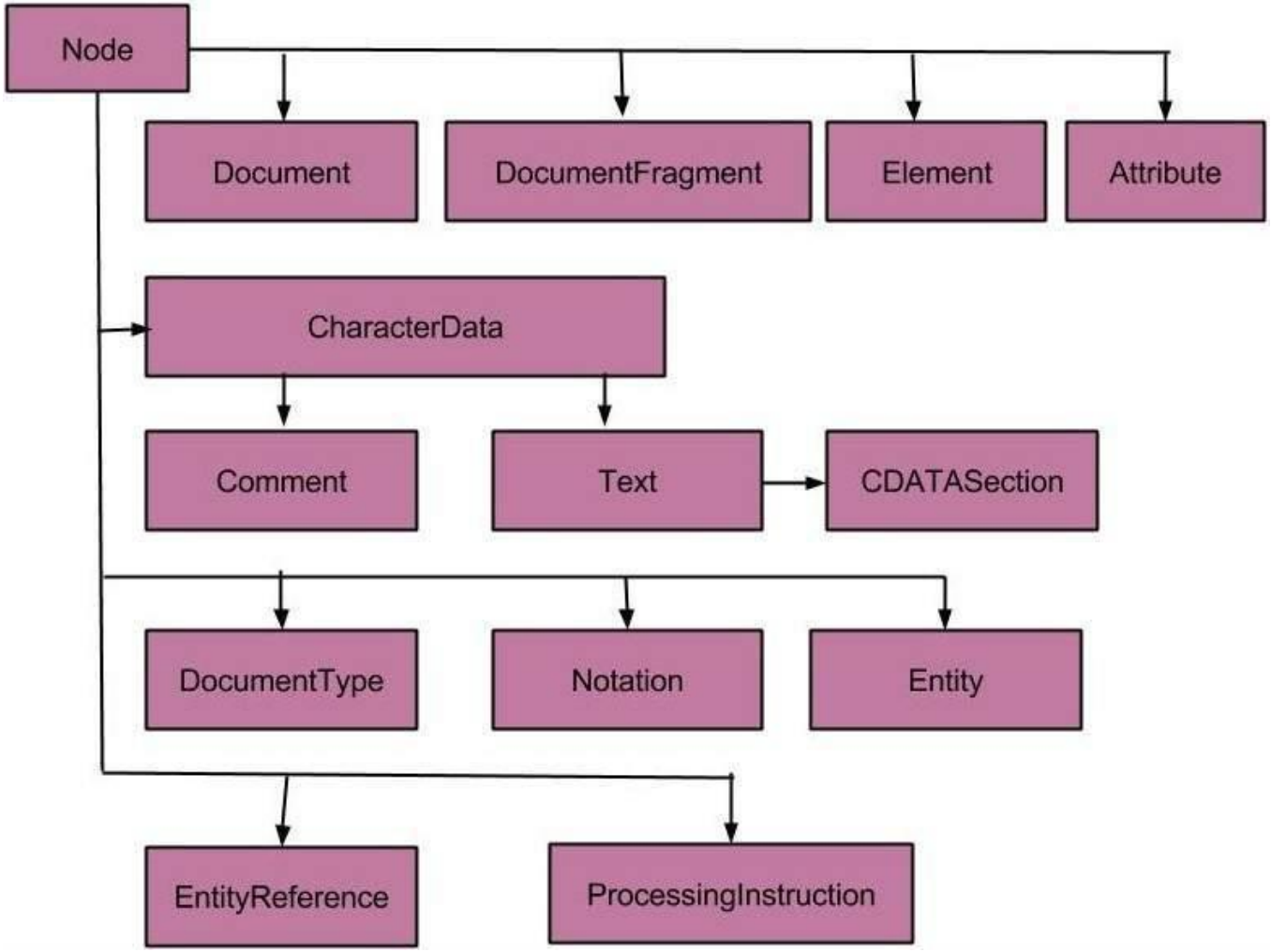


From the above diagram we can interface:

- The *parent node* can have multiple nodes called as *child* nodes. These *child* nodes can have additional node called as *attribute* node. In the above example we have two attribute nodes *Technical* and *Non-Technical*. The *attribute* node is not actually a child of the element node, but is still associated with it.
- These *child* nodes in turn can have multiple child nodes. The text within the nodes is called as *text* node.
- The node objects at the same level are called as siblings.
- The DOM Identifies:
  - the objects to represent the interface and manipulate the document.
  - the relationship among the objects and interfaces.

Node Types

- The following pictorial representations lists all the node



b

Parser

A *parser* is a software application that is designed to analyze a document, in our case XML document and do something specific with the information. Some of the DOM based parsers are listed in the table below:

- JAXP: Sun Microsystem's *Java API for XML Parsing (JAXP)* is available at no
- charge from **java.sun.com/xml**.
- XML4J: IBM's *XML Parser for Java (XML4J)* is available at no charge from **www.alphaworks.ibm.com/tech/xml4j**.
- Xerces : Apache's *Xerces Java Parser* is available at no charge from **xml.apache.org/xerces**.
- Msxml: Microsoft's XML parser (*msxml*) version 2.0 is built-into Internet Explorer 5.5. Version 3.0 is also available at no charge from **msdn.microsoft.com/xml**.
- 4DOM : 4DOM is a parser for the Python programming language and is available at no charge from **fourthought.com/4Suite/4DOM**.
- XML::DOM XML::DOM is a Perl module that we use to manipulate XML documents using Perl. For additional information, visit **www-4.ibm.com/software/developer/library/xml-perl2**.

In a tree-based API like DOM, the parser traverses the XML file and creates the corresponding DOM objects. Then you can traverse the DOM structure back and forth

XML DOM Properties

These are some typical DOM properties:

- 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
- Note: In the list above, x is a node object.

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
- Note: In the list above, x is a node object.

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to display the cookies associated with this document.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
   document.getElementById("demo").innerHTML =
   "Cookies associated with this document: " + document.cookie;
}
</script>

</body>
</html>
```

Output:

Click the button to display the cookies associated with this document.

Try it

When click will give all cookies

XML and Supplementary Technologies

- The W3C Document Object Model (DOM)
  - an API that allows developers to programmatically manage and access XML nodes
  - allows programmers to update and change XML documents within an application
  - reads the whole XML file and then stores a hierarchical tree structure containing all elements within the document
  - This tree has a single root node, which is the root element, and may contain many children, each of which represents an XML element

Method Name Description

- **createElement** Creates an element node.
- **createAttribute** Creates an attribute node.
- **createTextNode** Creates a text node.
- **createComment** Creates a comment node.
- **createProcessingInstruction** Creates a processing instruction node.
- **createCDATASection** Creates a **CDATA** section node.
- **getDocumentElement** Returns the document's root element.
- **appendChild** Appends a child node.
- **getChildNodes** Returns the child nodes.

Fig. Some **Document** methods.

Method Name Description

- **appendChild** Appends a child node.
- **cloneNode** Duplicates the node.
- **getAttributes** Returns the node's attributes.
- **getChildNodes** Returns the node's child nodes.
- **getNodeName** Returns the node's name.
- **getNodeType** Returns the node's type (e.g., element, attribute, text, etc.).
- **getNodeValue** Returns the node's value.
- **getParentNode** Returns the node's parent.
- **hasChildNodes** Returns **true** if the node has child nodes.
- **removeChild** Removes a child node from the node.
- **replaceChild** Replaces a child node with another node.
- **setNodeValue** Sets the node's value.
- **insertBefore** Appends a child node in front of a child node.

Fig. **Node** methods.

some node types that may be returned by method **getNodeType**

Node Type Description

- **Node.ELEMENT_NODE** Represents an element node.
- **Node.ATTRIBUTE_NODE** Represents an attribute node.
- **Node.TEXT_NODE** Represents a text node.
- **Node.COMMENT_NODE** Represents a comment node.
- **Node.PROCESSING_INSTRUCTION_NODE** Represents a processing instruction node.
- **Node.CDATA_SECTION_NODE** Represents a **CDATA** section node.

Fig. Some node types.

Method Name Description

- **getAttribute** Returns an attribute's value.
- **getTagName** Returns an element's name.
- **removeAttribute** Removes an element's attribute.
- **setAttribute** Sets an attribute's value.

Fig.**Element** methods