**Interactive Development with JavaScript – Module 11**

**DOM and DOM Diagram**

Generally, the DOM is a programming interface for markup languages such as HTML and XML. Using this programming interface, we can manipulate a DOM by changing the structure, content, and/or style. In the DOM we will have nodes; however, when manipulating an HTML document, we will refer to them as elements. Some of these elements are referred to as objects, properties, methods and events. In this class we will focus on the JavaScript Application Programming Interface (API). Using JavaScript, we can manipulate the DOM as well as reacting to events. Although every browser builds a type of a DOM structure, each one having different implementations, so testing on various browsers helps us to address problems we either correct or identify the browsers our applications support.

![Diagram

Description automatically generated]()The DOM programming interface makes altering an HTML document easier; however, we should remember the HTML markup language and a DOM structure are not programming languages. Programming the DOM interface using JavaScript can start out with simple syntax and progress to a more complicated syntax. A simple DOM diagram:

**DOM Nodes/Elements**

Some of the types of objects, or elements, in a DOM are Document, Node, Element, NodeList, and Attribute.

* Document
  + The Document object is the root HTML document.
  + Syntax structure examples:
    - document.getElementById(id)
    - document.getElementByTagName(tagName)
    - document.getElementByClassName(className)
* Node
  + All objects in a DOM are nodes inside the document object.
* Element
  + All HTML nodes are referred to as elements or element nodes. An element object in the DOM represents an HTML element such as a <table>, <p>, <div>, or <a>
  + Syntax structure examples:
    - element.innerHTML(id)
    - element.attribute(attributename)
    - element.style.property
* NodeList
  + The DOM NodeList is similar to an array (collection) of elements in the document object. This collection holds document elements (nodes) that are returned by methods such as getElementByTagName. Because a NodeList is not an array, methods such as push(), pop(), or join() cannot be used on a NodeList, however elements may be accessed by array type syntax of either list.item(i) or list[i].

**DOM Interface**

The DOM interface provides methods and properties we can use to manipulate/change documents.

* Accessing Nodes
  + getElementsByTagName()
    - This method will return a NodeList of all elements in a document or elements matching a tag name.
    - Elements in the document
    - document.getElementsByTagName(tagName);
    - Elements in a child element
    - element.getElementsByTagName(tagName);
  + getElementById()
    - This method returns an element with a matching ID.
    - document.getElementById(elementID);
  + getElementsByClassName()
    - This method returns a NodeList collection of elements which have all the given class names.
    - Elements in the document
    - document.getElementsByClassName(className) ;
    - Element's child elements
    - element.getElementsByClassName(className);
  + getAttribute()
    - Gets an element’s attribute value.
    - element.getAttribute(attributename);
  + setAttribute()
    - Sets an element’s attribute value.
    - element.setAttribute(attributeName, attributeValue);
  + hasAttribute()
    - Checks for an element’s attribute existence.
    - element.hasAttribute(attributeName);
  + removeAttribute()
    - Removes an attribute from an element.
    - element.removeAttribute(attributeName);
* Altering Nodes
  + setAttribute()
    - Adds an attribute to an element giving it an initial value.
    - Workable style change.
    - element.setAttribute("style", "background-color:
    - red;");
    - Better approach is to alter a style object’s property.
    - element.style.backgroundColor = "red";
    - You can use the style object’s property with the following syntax styles:
      * Return
        + element.style.property
      * Set
        + element.style.property = value
  + innerHTML
    - The innerHTML sets or returns an element’s inner HTML content.
    - Return the innerHTML content:
    - document.getElementById(ID).innerHTML;
    - Set the innerHTML content:
    - document.getElementById(ID).innerHTML = "New
    - Content ";
    - Delete the innerHTML content:
    - document.getElementById(ID).innerHTML = "";
* Add, Remove, Replace Nodes
  + createElement()
    - This method creates an element with a specified name.
    - document.createElement(elementName);
  + appendChild()
    - This method appends a child element to a given parent element.
    - element.appendChild(element);
  + insertBefore()
    - Adds an element before an existing element you specify.
    - node.insertBefore(newElement, existingElement);
  + removeChild()
    - This method removes a specified child element from the specified element.
    - element.removeChild(element);

[**Module 11 Programming Assignment (Click Here To Submit Your Assignment)**](https://cyberactive.bellevue.edu/webapps/assignment/uploadAssignment?content_id=_13970479_1&course_id=_513834_1&group_id=&mode=view)

Attached Files:

* [[File](https://cyberactive.bellevue.edu/bbcswebdav/pid-13970479-dt-content-rid-84392330_2/xid-84392330_2) Module 11 Example Zip File](https://cyberactive.bellevue.edu/bbcswebdav/pid-13970479-dt-content-rid-84392330_2/xid-84392330_2) (840.679 KB)

**Assignment:**

For this assignment:

* 1. Create a page similar to the example attached to this assignment that alters the DOM based on user input.
  2. Create a function that is executed when an image is clicked, using the “onclick” MouseEvent.