WT Laboratory 4

Laboratory Continuous Assessment (LCA)

Understanding of the Objective (5)	Performance (5)	Journal Submission and Ethics (Neatness, Handwriting, Timely submission) (5)	Orals (5)	Total (20)	Remarks	Instructor's Sign

Aim: Write a client-side script with JavaScript to access and manipulate Document Object Model (DOM) objects in an HTML web page.

Objectives:

1. To learn basic functioning of DOM objects.
2. To understand how to retrieve and manipulate DOM objects in an HTML web page.

Theory:

1. HTML Document Object Model
2. DOM methods, properties
3. Javascript Event Handlers
4. Linking external Javascript file to HTML file

FAQ:

1. How do you access an element of DOM having id=”abc”?
2. Provide a sample code for accessing DOM element having class=”blue”.
3. How to access all the <p> elements using DOM methods?
4. Give an example of how to modify an attribute value using DOM.
5. How do you modify the CSS properties using HTML DOM?
6. Give an example code of adding and removing a node from the DOM tree.

(Minimum 3 handwritten pages)

Note: Student is expected to attach this page as a title page of an assignment.

Document Object Model

The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents the page so that programs can change the document structure, style, and content.

- The DOM represents the document as nodes and objects. That way, programming languages can connect to the page.
- A Web page is a document. This document can be either displayed in the browser window or as the HTML source. But it is the same document in both cases.
- The Document Object Model (DOM) represents that same document so it can be manipulated. The DOM is an object-oriented representation of the web page, which can be modified with a scripting language such as JavaScript.

For example, following html code represents sample code,

HTML Snippet

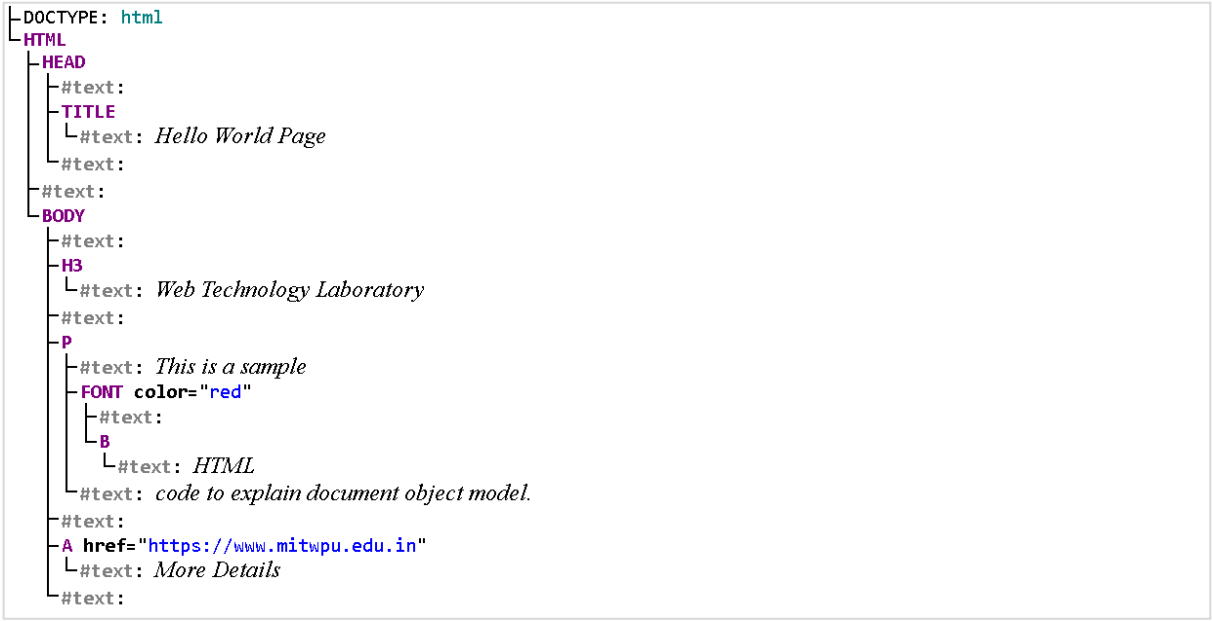
```
<!DOCTYPE html>
<html>
<head> <title> Hello World Page </title> </head>
<body>
<h3> Web Technology Laboratory </h3>
<p>
This is a sample <font color="red"> <b> HTML </b></font>
code to explain document object model.
</p>
<a href="https://www.mitwpu.edu.in"> More Details </a>
</body>
</html>
```

Rendered View (Web Browser)

Web Technology Laboratory

This is a sample HTML code to explain document object model.

More Details



DOM and JavaScript

- The DOM is not a programming language, but without it, the JavaScript language wouldn't have any model or notion of web pages, HTML documents, XML documents, and their component parts (e.g. elements).
- Every element in a document—the document as a whole, the head, tables within the document, table headers, text within the table cells—is part of the document object model for that document, so they can all be accessed and manipulated using the DOM and a scripting language like JavaScript.
- The page content is stored in the DOM and may be accessed and manipulated via JavaScript, so that we may write this approximate equation:

  API (HTML or XML page) = **DOM** + **JS** (scripting language)
- The DOM was designed to be independent of any particular programming language, making the structural representation of the document available from a single, consistent API.

  DOM defines the following

  - The HTML elements as **objects**
  - The **properties** of all HTML elements
  - The **methods** to access all HTML elements
  - The **events** for all HTML elements

Common Properties in HTML DOM

HTML DOM properties are **values** (of HTML Elements) that you can set or change.

Property	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.style.*property = new style*	Change the style of an HTML element

Common Methods Involved in HTML DOM

- HTML DOM methods are used as a way to generate the result without much of the written code and it can be easily used.
- HTML DOM methods provide a way to manipulate HTML elements without much efforts.
- In DOM context, Document and window objects are the objects whose interfaces you generally use most often in DOM programming.
- In simple terms, the window object represents something like the browser, and the document object is the root of the document itself.
- The following is a brief list of common APIs in web using the DOM:
  document.getElementById(id) –
  Allows to access HTML element associated with a specified id.
  document.getElementsByTagName(name) –
  Allows to get all elements using tag name that is associated with it.
  document.getElementsByClassName()
  Allows to get all elements using class name associated with it
  document.appendChild(node) –
  Allows to insert a child node to x and append the child node at the end of the tree.
  document.removeChild(node) –
  Allows to remove a child node from x and remove from the tree area also.

Events in HTML DOM

HTML DOM Events allow JavaScript to register different event handlers on elements in an HTML document.
Example events:

- When a user clicks the mouse
- When a web page has loaded
- When an image has been loaded
- When the mouse moves over an element
- When an input field is changed
- When an HTML form is submitted
- When a user strokes a key

Accessing the DOM Elements

- Different browsers have different implementations of the DOM, and these implementations exhibit varying degrees of conformance to the actual DOM standard (a subject we try to avoid in this documentation), but every web browser uses some document object model to make web pages accessible via JavaScript.

Accessing Element Having particular id

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

<h2>My First Page</h2>

<p id="demo"></p>

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

</body>
</html>
```

Accessing element not having an id attribute (getElementsByTagName())

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

<h2>Finding HTML Elements by Tag Name</h2>
<p>This is first p element on this html page</p>
<p>This is second p element on the page</p>
<p id="demo">some text</p>

<script>
var y = document.getElementsByTagName("p");
var a = document.getElementById("demo");
a.innerHTML = 'Accessing text in the second p tag: ' + y[1].innerHTML;
</script>

</body>
</html>
```

Accessing element not having a particular class (getElementsByClassName())

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

<div class="example">First div element with class="example".</div>
<div class="example">Second div element with class="example".</div>
<button onclick="myFunction()">Change Text</button>

<script>
function myFunction() {
  var x = document.getElementsByClassName("example");
  x[0].innerHTML = "Hello World!";
}
</script>

</body>
</html>
```

Modifying the CSS properties of Element

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

<p id="p1">Hello World!</p>
<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color = "blue";
document.getElementById("p2").style.fontFamily = "Arial";
document.getElementById("p2").style.fontSize = "larger";
</script>
</body>
</html>
```

Adding Elements to HTML DOM Tree:

Another example. This function creates a new H1 element, adds text to that element, and then adds the H1 to the tree for this document:

HTML Code

```
<html>
  <head>
    <script>
      // run this function when the document is loaded
      window.onload = function() {

        // create a couple of elements in an otherwise empty HTML page
        var heading = document.createElement("h1");
        var heading_text = document.createTextNode("Big Head!");
```

```
        heading.appendChild(heading_text);
        document.body.appendChild(heading);
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

DOM View (DOM Representation of HTML Code)

```
-HTML
  ├HEAD
  │ ├#text:
  │ ├SCRIPT
  │ │ └#text: // run this function when the document is loaded window.onload = function() { // create a couple of elements
  │ │   in an otherwise empty HTML page var heading = document.createElement("h1"); var heading_text =
  │ │   document.createTextNode("Big Head!"); heading.appendChild(heading_text); document.body.appendChild(heading);
  │ │   }
  │ └#text:
  ├#text:
  └BODY
    ├#text:
    └H1
      └#text: Big Head!
```

Rendered View (Web Browser)

Big Head!

Removing Elements from HTML DOM Tree:

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

<div>
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>


<button onclick="myFunction()">Remove Element</button>

<script>
function myFunction() {
  var elmnt = document.getElementById("p1");
  elmnt.remove();
}
</script>

</body>
</html>
```

Javascript Event Handlers

Event handlers are the functions which get executed when a particular event occurs. Sample code for event handler using external Javascript file shown below.

bgcolor.html file code:

```
<!DOCTYPE html>
<html lang="en">

  <head>
    <title>Learning the DOM</title>
  </head>

  <body>
    <h1>Document Object Model</h1>
    <button id="changeBackground">Change Background Color</button>

    <script src="script.js"></script>
  </body>

</html>
```

script.js file code:

```
b = document.getElementById('changeBackground');

b.addEventListener('click', myfunction);
function myfunction()
{
  document.body.style.backgroundColor = 'red';
}
```

Laboratory Assignment:

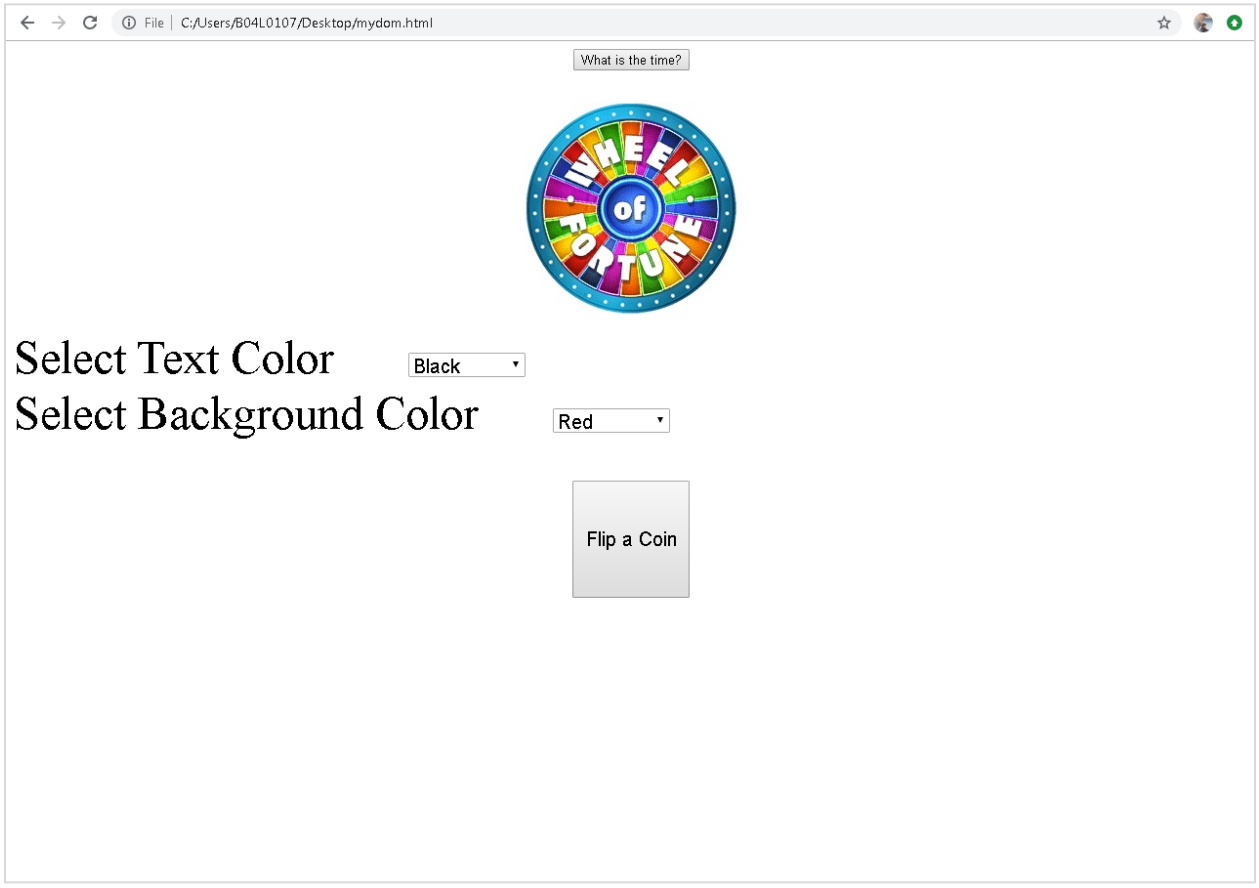
HTML Code Snippet:

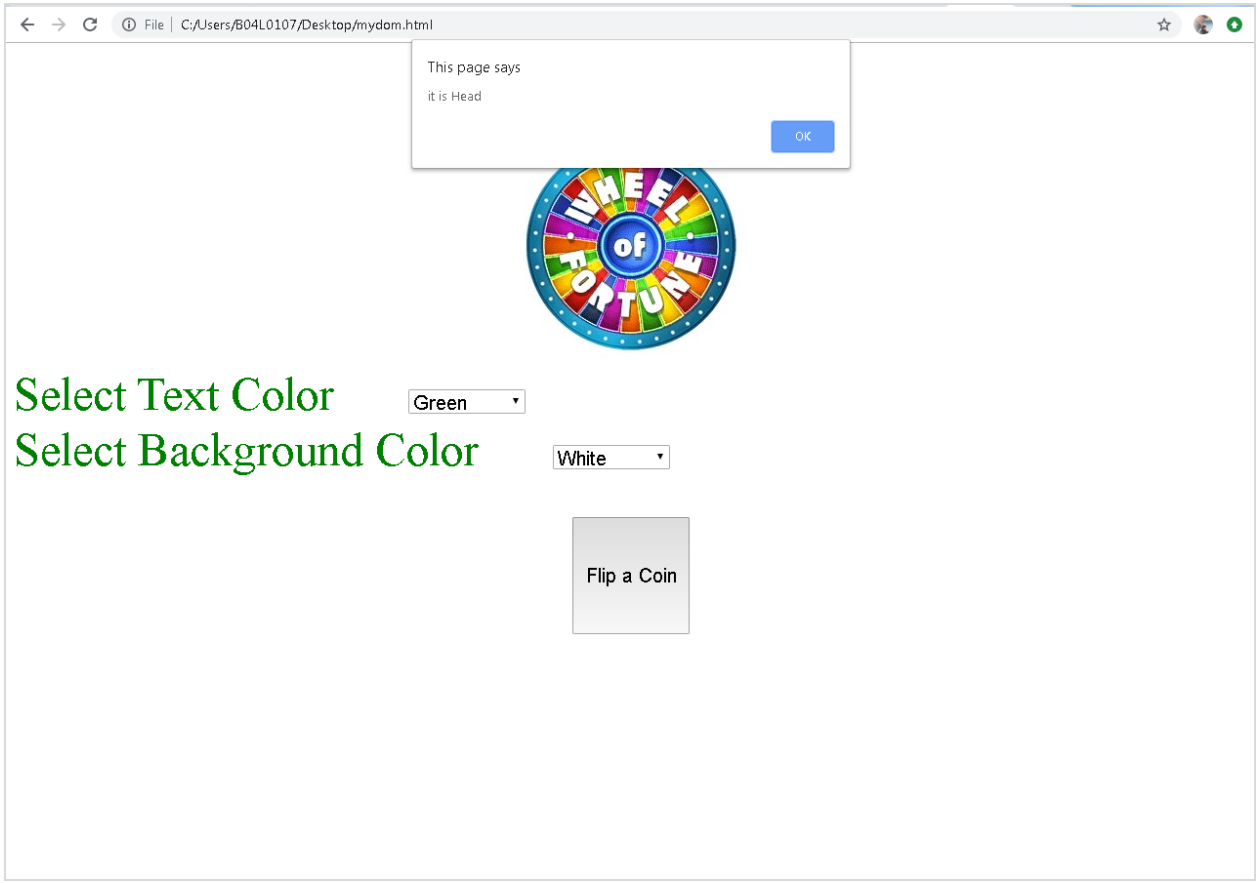
```
<html>
  <head>
    <title>DOM Tests</title>
    <script type="application/javascript">
 function changeImage() {
//alert(document.getElementById("im4").src);

        if (document.getElementById("im4").src ==
"file:///C:/Users/B04L0107/Desktop/img1.jpg")
        {
            document.getElementById("im4").src = "img2.gif";
        }
        else{  document.getElementById("im4").src = "img1.jpg";  }       }

function coinFlip()
{     (Math.floor(Math.random() * 2) == 0) ? alert('it is Head') : alert('it is
Tail') }




function setBodyAttr(attr, value)
{
      if (document.body)
       eval('document.body.'+attr+'="'+value+'"');
      else
       notSupported();
    }
</script>
</head>

<body>
<center>
```

```
<button onclick="getElementById('demo').innerHTML=Date()">What is the
time?</button>
<p id="demo"></p>
</center>
<center>
        <img src="img1.jpg" id="im4" onclick="changeImage()" />

</center>
      <font size=20>Select Text Color      </font>

        <select style="font-size:20px; width: 120px; height: 25px;"
onChange="setBodyAttr('text',this.options[this.selectedIndex].value);">

          <option value="black">Black

          <option value="red">Red
   <option value="Green">Green

          <option value="purple">Purple

          <option value="blue">Blue
   <option value="brown">Brown

        </select><br>
<font size=20>Select Background Color      </font>

        <select style="font-size:20px; width: 120px; height: 25px;"
onChange="setBodyAttr('bgColor',

        this.options[this.selectedIndex].value);">
      <option value="red">Red
   <option value="Green">Green

          <option value="purple">Purple

          <option value="blue">Blue
   <option value="brown">Brown

          <option value="white">White

          <option value="lightgrey">Gray

        </select>
<br>
       <br><br>
    <center>
```

```
        <input type="button" style="font-size:20px; width: 120px; height: 120px;"
value="Flip a Coin" onclick="coinFlip()" />
</center>
</form>
    </div>
</body>
</html>
```

Rendered View





Platform/ Languages used: Windows/Ubuntu, Web Browser(Google Chrome/Mozilla Firefox), HTML Code Editor(Notepad/Adobe Dreamweaver/Sublime Text)

Ref:
https://developer.mozilla.org/en-S/docs/Web/API/Document_Object_Model/

Conclusion:
Hence in this way we have implemented a program with HTML & JavaScript to access and manipulate Document Object Model (DOM) objects in an HTML web page.