LAB # 3

Document Object Model [DOM]

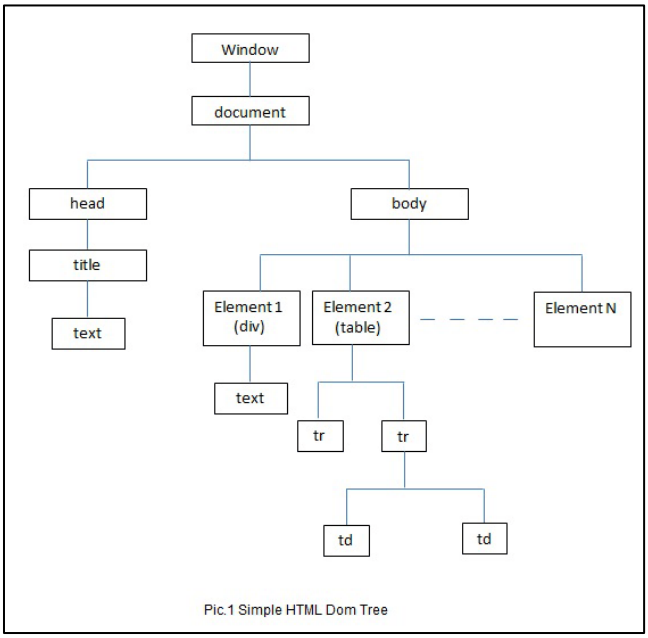
OBJECTIVE

To learn what is DOM and the APIs available for manipulating them.

THEORY

When html is parsed and rendered by browser, it generates a memory representation of all the html elements in a hierarchal fashion and make them available to be manipulated called DOM (Document Object Model). Whenever some changes are made to the DOM via API, a similar change is made to the actual html document and the document is re-rendered. Same is true when some changes are made to the html elements directly, it affects is counter-part DOM.

DOM manipulation includes creating new elements, removing new elements, modifying, searching, and appending them. It can be used to change the style, set the inner HTML of the parent element, bind events to the DOM elements etc. The DOM API is available via *document* object.



Pic.1 Simple HTML Dom Tree

DOM is for interaction with html elements of the *document* as depicted in the picture above, but when we need to manipulate the *browser* features, we use BOM (Browser Object Model). BOM provides features which include alerts, windows, timers, navigation, history, location, cookies etc., and also their APIs to manipulate them.

DOM – *Searching Elements*

For searching, we have following DOM APIs available which are frequently used to search DOM elements:

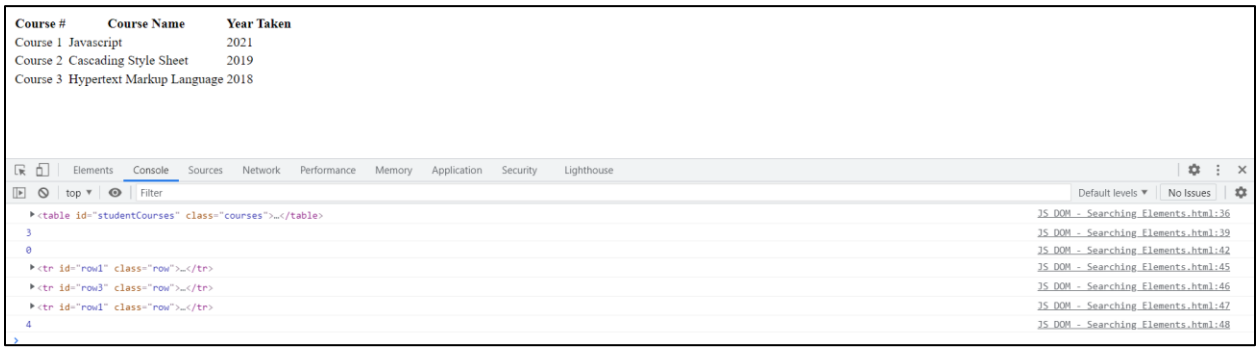
1. *document.getElementById*
   This is used to search any element within the DOM using element's ID.
2. *document.getElementByClassName*
   This is used to search elements within the DOM using element's class name.
3. *document.getElementsByTagName*
   This is used to search elements within the DOM using element's tag name.
4. *document.querySelector*
   This is used to search first matched element within the DOM using element's CSS selector. That is, we use similar search pattern as is used in CSS to match element.
5. *document.querySelectorAll*
   This is used to search all matched element within the DOM using element's CSS selector.

Sample Program

```
<!DOCTYPE html>
<html>
	<head>
		<title>Lab#3 – DOM</title>
	</head>
	<body>
		<table id=”studentCourses” class=”courses”>
			<thead>
				<tr>
					<th>Course #</th>
					<th>Course Name</th>
					<th>Year Taken</th>
				</tr>
			</thead>
			<tbody>
				<tr id=”row1” class=”row”>
					<td>Course 1</td>
					<td>Javascript</td>
					<td>2021</td>
				</tr>
				<tr id=”row2” class=”row”>
					<td>Course 2</td>
					<td>Cascading Style Sheet</td>
					<td>2019</td>
				</tr>
```

```
                        <tr id=”row3” class=”row”>
                            <td>Course 3</td>
                            <td>Hypertext Markup Language</td>
                            <td>2018</td>
                        </tr>
                    </tbody>
                </table>
                <script>
                    //document.getElementById - Search by element's ID
                    console.log(document.getElementById('studentCourses'));

                    //document.getElementByClassName - Search by element's CSS class
name
                    console.log(document.getElementsByClassName('row').length);

                    //document.getElementsByTagName - Search by element's tag name
                    console.log(document.getElementsByTagName('row').length);

                    //document.querySelector - Generic search using any of CSS selector
                    console.log(document.querySelector('#row1'));
        //Search element using ID
                    console.log(document.querySelector('#row3'));
        //Search element using ID
                    console.log(document.querySelector('.row'));
        //Search first element from document top using CSS class name
                    console.log(document.querySelectorAll('tr').length);          //Search all
elements using tag name and return elements array
                </script>
        </body>
</html>
```

Output



DOM – *Adding, Removing, Updating Elements*

To manipulate DOM, we have following frequently used APIs:

1. *createElement*
   This is used to create any new element
2. *appendChild*
   This is used to insert new child elements into any existing element
3. *removeChild*
   This is used to delete any element from its parent
4. *innerHTML*
   This is used to update HTML of any element. This will just update the content of it. Using this, either text or HTML can be updated as its content.

Sample Program

```
<!DOCTYPE html>
<html>
	<head>
		<title>Lab#3 – DOM </title>
	</head>
	<body>
		<table id="studentCourses" class="courses">
			<thead>
				<tr>
					<th>Course #</th>
					<th>Course Name</th>
					<th>Year Taken</th>
				</tr>
			</thead>
			<tbody>
				<tr id="row1" class="row">
					<td>Course 1</td>
					<td>Javascript</td>
					<td>2021</td>
				</tr>
				<tr id="row2" class="row">
					<td>Course 2</td>
					<td>Cascading Style Sheet</td>
					<td>2019</td>
				</tr>
				<tr id="row3" class="row">
					<td>Course 3</td>
					<td>Hypertext Markup Language</td>
					<td>2018</td>
				</tr>
			</tbody>
		</table>
```

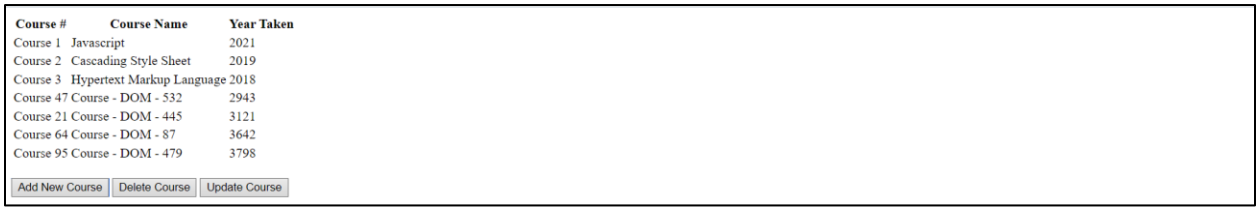
```
                        </br>

                        <input type="button" value="Add New Course"
onclick="addNewCourse();" />
                        <input type="button" value="Delete Course" onclick="deleteCourse();" />
                        <input type="button" value="Update Course" onclick="updateCourse();"
/>

                        <script>

                                //This declarative function will be called on 'Add New Row' button
click and add new row
                                function addNewCourse(){

                                        //1 - Find the 'body' element
                                        const body = document.querySelector('tbody');

                                        //2 - Create elements using document.createElement API
(method)
                                        const row = document.createElement('tr');
        //Create 'tr' element
                                        const tdCourseID = document.createElement('td');
        //Create 'td' element
                                        const tdCourseName = document.createElement('td');
        //Create 'td' element
                                        const tdYearTaken = document.createElement('td');
        //Create 'td' element

                                        const courseId = ((Math.round(Math.random() * 100)) + 4);
                                        const courseName = ((Math.round(Math.random() * 500))
+ 50);
                                        const yearTaken = (Math.round(Math.random() * 2020)) +
2020;
                                        row.id = 'row' + courseId;

                                        //3 - Update td's 'innerHTML' (basically element content -
anything within the angle brackets)
                                        tdCourseID.innerHTML = 'Course ' + courseId;
                                        tdCourseName.innerHTML = 'Course - DOM - ' +
courseName;
                                        tdYearTaken.innerHTML = yearTaken;

                                        //4 - Insert newly created 'td's into newly created 'tr' using
element's appendChild API
                                        row.appendChild(tdCourseID);
```

```
                row.appendChild(tdCourseName);
                row.appendChild(tdYearTaken);

                //5 - Finally insert newly created 'row' into 'body' element
                body.appendChild(row);
        }

        function deleteCourse(){
                //Ask for course# using prompt
                const courseId = prompt('Please enter course#');

                //If user has entered course#
                if(courseId){
                        //Find the row containing the 'course #' entered by
the user
                        const row = document.getElementById('row' +
courseId);
                        if(row){
                                //Find the parent of the 'row' just found so
that it can be removed from its child list
                                const body =
document.querySelector('tbody');

                                //Finally remove it from 'tbody' element
                                body.removeChild(row);
                        }
                }
        }

        function updateCourse(){
                //Ask the course# and year taken
                const newCourseYear = prompt('Please enter course# and
new year delimited by comma');

                //If the user has entered course# and year taken
                if(newCourseYear){
                        //Split the comma separated input entered by the
user. Example input; 3,2021
                        const courseId = newCourseYear.split(',')[0];
                        const courseYear = newCourseYear.split(',')[1];

                        //Find the row containing the 'course #' entered by
the user
                        const row = document.getElementById('row' +
courseId);
                        if(row){
```

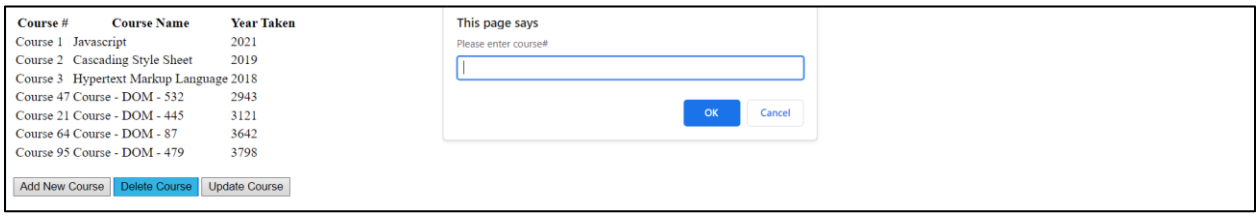
```
                                                    //Find the 'year taken' td element within the
'row' entered by the user
                                                    const tdYearTaken =
row.querySelector('#row' + courseId + ' > td:nth-child(3)');

                                                    //Finally update the new 'year taken' entered
by the user using 'innerHTML' property of element
                                                    tdYearTaken.innerHTML = courseYear;
                                                }
                                            }
                                        }

                    </script>
            </body>
</html>
```

Output

Output – Add New Course



Output – Delete Course



Output – Update Course

DOM – *Event Binding and Styling Elements*

Sample Program

```
<!DOCTYPE html>
<html>
	<head>
		<title>Lab#3 – DOM </title>
	</head>
		<body>

		<style>
			.button-style {
				border: 1px solid blue;
				border-radius: 25px;
				background-color: red;
				color: white;
				display: inline-block;
				width: 150px;
			}
		</style>


		<table id="studentCourses" class="courses">
			<thead>
				<tr>
					<th>Course #</th>
					<th>Course Name</th>
					<th>Year Taken</th>
				</tr>
			</thead>
			<tbody>
				<tr id="row1" class="row">
					<td>Course 1</td>
					<td>Javascript</td>
```

```
                        <td>2021</td>
                    </tr>
                    <tr id="row2" class="row">
                        <td>Course 2</td>
                        <td>Cascading Style Sheet</td>
                        <td>2019</td>
                    </tr>
                    <tr id="row3" class="row">
                        <td>Course 3</td>
                        <td>Hypertext Markup Language</td>
                        <td>2018</td>
                    </tr>
                </tbody>
            </table>

            </br>

            <input id="addTableBorder" type="button" value="Add Table Border" />
            <input id="addRowColor" type="button" value="Add Row Color" />
            <input id="addButtonStyling" type="button" value="Add Button Styling" />

            <script>
                const btnTableBorder = document.getElementById('addTableBorder');
                const btnRowColor = document.getElementById('addRowColor');
                const btnButtonStyling = document.getElementById('addButtonStyling');

                //Add click event to the button using addEventListener API
                btnTableBorder.addEventListener('click', function(event) {
                    //Find the 'table' element
                    const table = document.getElementById('studentCourses');
                    //Update its 'border' style
                    table.style.border = '1px solid green';
                });

                //Add click event to the button using addEventListener API
                btnRowColor.addEventListener('click', function(event) {
                    //Find the 'row' element
                    const row = document.querySelector('#row2');

                    //Update its 'background-color' and 'color' style
                    row.style.backgroundColor = 'yellow';
                    row.style.color = 'brown';
                });

                //Add click event to the button using addEventListener API
                btnButtonStyling.addEventListener('click', function(event) {
```

```
                        //Add the 'button-style' CSS class created above within the
'<style>' tag
                        btnButtonStyling.classList.add('button-style');
                    });
                </script>
        </body>
</html>
```

Output – Add Table Border

Output – Add Row Color

Output – Add Button Styling

Lab Task

1 Create a new *Student Profile* page
   a. Add *Title* using suitable *h1* or *h2* tag
   b. Add an *introduction* paragraph along with image using *p* and *img* tags respectively
   c. Add a table listing down *courses* taken using *table* tag
      i. Add course Id, course Name, credit hours and year taken in
      ii. Add at least four courses using *row* tag
      iii. At the bottom of table, create a *div* tag and add following buttons
         1. Search and highlight the course row input by the user (using window.prompt)
         2. Add new Course
         3. Update Course
         4. Remove Course
   d. Add *Interests* section
      i. Add link using *a* tag pointing to SSUET university home page
      ii. Add link using a tag pointing to SSUET university Home -> Students -> Scholarship Notice page
      iii. Add link using a tag pointing to SSUET university Home -> News and Events -> Events Calendar page

Home Task

1. Redirect to Student Profile page after login

   a. Before moving to the *Student Profile* page, show the success message on login button click using *window.alert* and redirect it to the Student Profile page in a new tab using *window.open*

   b. Send the query params (Student Id) from the login page to the *Student Profile* page

   c. Check if the query param (Student Id) is present or not. If not, then redirect the user to the *Registration Form* created in Lab#1