LAB # 2

Basic JavaScript

OBJECTIVE

To get familiar with JS Basics.

THEORY

Javascript (JS) is a scripting language which is run by the browser. It is also called a client-side language. The JS engine available in the browser parses the JS code and executes it. There are different types of functionality available in JS which can be used to perform different operations like manipulating the DOM (Document Object Model – a memory representation of HTML), interacting with the server using AJAX (Asynchronous Javascript and XML – a set of APIs available for interactions with the server), scheduling the operations using timers (setTimeout and setInterval), using browser features a.k.a, BOM (Browser Object Model) which include doing navigation, using cookies, sessions, cookies etc.

JS provides different building blocks to write a program which include variables, functions, objects, loops, data types, events etc. Besides basic building blocks we have APIs (Application Programming Interface – basically methods or functions) for manipulating DOM, CSS and BOM among others.

JS (Javascript) adds interaction capability to the document. Its main responsibility is to respond against various events going on in the life cycle of the HTML document. When any event like clicking on the button occurs, JS can be used to decide what to do with that button Click event. It can be used to take input from the user and generate output for her upon any interaction like when the download is completed, it can notify her of the download completed state via handling downloadCompeted event etc.

JS building blocks - *Variables*

```
<!DOCTYPE html>
<html>
	<head>
		<title>Lab#1 – HTML, JS, and CSS Fundamentals</title>
	</head>
	<body>
		<script>
			//Variables using var keyword – They are scoped within the function in which they are declared. Different data type are available like number, string, boolean, date etc.
			function helloFunctionScopedVariables()
			{
				var a = 1;
				var b = "Hello Student";
				var c = true;
				var d = new Date();

				if( a >= 1){
					var e = 2;
				}

				console.log("FUNCTION SCOPED");
				console.log(a);
				a = e;
				console.log(a) //Prints 2
				console.log(b);
				console.log(c);
				console.log(d);
				console.log(e);
			}
			//Variables using let and const keywords - They are scoped withing the braces “{“ “}”
			function helloBracesScopedVariables()
			{
				const PI = 1.421; //Once assigned, its value can never be modified
				let a = 1;
				let b = "Hello Student";
				let c = true;
				let d = new Date();

				console.log("BRACES SCOPED");
				if( a >= 1) {
					let e = 2;
					console.log(e);
```

```
                        }

                        console.log(a);
                        console.log(b);
                        console.log(c);
                        console.log(d);

                        a = e;   //Error – "e is not defined" as e is scoped within the braces and not available outside of it
                    }

                    helloFunctionScopedVariables();
                    helloBracesScopedVariables();
                </script>
        </body>
</html>
```

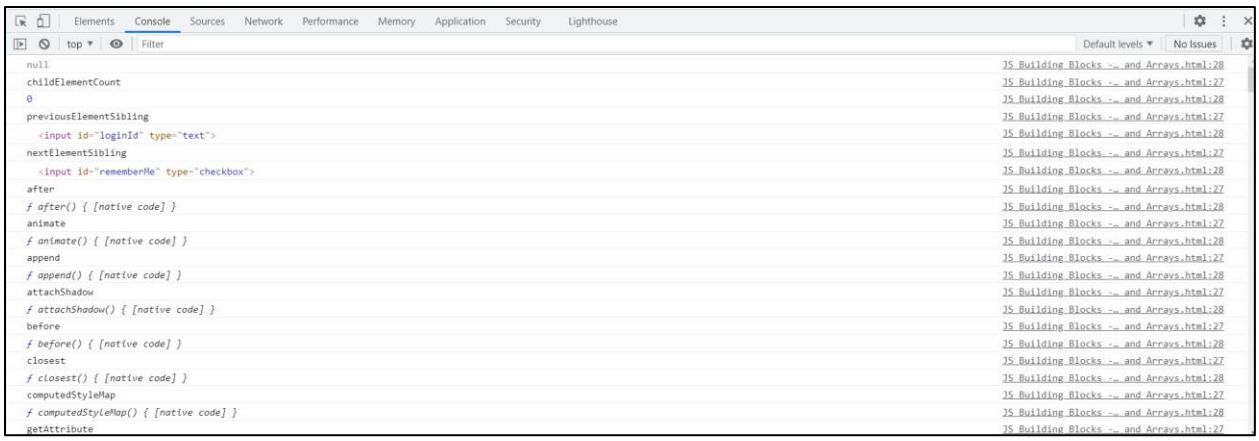
Output

JS building blocks – *Loops and Arrays*

```
<!DOCTYPE html>
<html>
        <head>
                <title>Lab#1 – HTML, JS, and CSS Fundamentals</title>
        </head>
        <body>
                <input id="loginId" type="text" />
                <input id="password" type="password" />
                <input id="rememberMe" type="checkbox" />
                <script>
                        const loginIdElement = document.querySelector('#loginId');
                        const passwordElement = document.querySelector('#password);
                        const rememberMeElement = document.querySelector('#rememberMe');
```

```
                const inputs = [loginIdElement, passwordElement, rememberMeElement];

                //Sequential looping over the elements array using for loop
                for(let i = 0; i < inputs.length; i++)
                {
                        console.log(inputs[i].attributes["type"].value);

                        const inputObject = inputs[i];

                        //looping over object properties using for-in loop
                        for(let prop in inputObject)
                        {
                                console.log(prop);
                                console.log(inputObject[prop]);
                        }


                        //looping over object properties using for-of loop
                        for(let propertyValue of inputObject.getAttributeNames())
                        {
                                console.log(propertyValue);
                        }
                }
        </script>
    </body>
</html>
```

Output



JS building blocks – *Functions and Objects*

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

```
<head>
	<title>Lab#1 – HTML, JS, and CSS Fundamentals</title>
</head>
<body>
	<script>
		//Define a teacher object using literal syntax
		let teacher = {
			id: 123,
			name: "teacher1",
			age: 35,
			class: "CS",
			dob: new Date(1986, 10, 10),
			isGoldMedalist: true
		};

		//Define multiple student objects with the help of a loop
		let students = []; //Start with empty list of students
		for(let s = 0; s < 10; s++)
		{
			let student = {
				id: s + 1,
				name: "student" + (s + 1),
				age: 19 + s,
				class: "HTML,CSS,JS",
				dob: new Date(2002 + s, 10, 10),
				isGoldMedalist: false
			};
			students.push(student); //insert newly created student into students array using array's method push
		}

		//function is created using function expression. Functions are objects like other normal objects
		let printTeacherName = function(teacher, noOfTimes) {
			let t = 1;
			while(t++ <= noOfTimes){
				console.log(teacher.name);
			}
		};

		console.log(printTeacherName.name);

		printTeacherName(teacher, 3);
		printTotalStudentsLength();

		//function is created using arrow syntax.
```

```
                const calculateStudentsCombinedAge = () => {
                        let studentsCombinedAge = 0;
                        for(let s = 0; s < students.length; s++) {
                                studentsCombinedAge += students[s].age;
                        }
                        return studentsCombinedAge;
                };

                console.log(calculateStudentsCombinedAge.name);

                let combinedStudentsAge = calculateStudentsCombinedAge();
                console.log(combinedStudentsAge);

                //function is created using declaration syntax
                function printTotalStudentsLength(){
                        console.log(students.length);
                }

                console.log(printTotalStudentsLength.name);
        </script>
    </body>
</html>
```

Output

JS building blocks – *Events*

```
<!DOCTYPE html>
<html>
        <head>
                <title>Lab#1 – HTML, JS, and CSS Fundamentals</title>
        </head>
        <body>
                <input type=”text” placeholder=”Enter some value”
onchange=”valueIsChanged(this)” />
                <input type=”button” value=”Log Current Date and Time”
onclick=”logCurrentDateTime()” />
```

```
                <script>
                        //This function will be executed whenever the value is changed in the
textbox, and will log old and new values for textbox
                        function valueIsChanged(e){
                                console.log(`Current value is: ${e.value}`);
                        }

                        //This function will be executed whenever the button is clicked, and will
log the current date and time using DateTime object's now() function
                        function logCurrentDateTime(){
                                console.log(new Date());
                        }
                </script>
        </body>
</html>
```

Output

Lab Task

1 Create a *Login Form* using HTML
   a. Add Title (using h1 or h2 tag)
   b. Add a *div*
   c. Add a *form* element within the *div*
   d. Add Login ID, Password textboxes and group them in a section (using *div* for section and <*input type=”text”* … for textboxes) and put the section under the *form*
   e. Set Login ID and Password as mandatory inputs (using *required* element attribute)
   f. Add OK and Cancel buttons (using <*input type=”button”* …)
2 Add styling to the *Login Form* using CSS
   a. Create a separate style file and link it with HTML file within the <*head*> section using <*link*> tag
   b. Add color to title, labels, inputs, and buttons text (color)
   c. Add borders to the textboxes (pixels, color, and border style)
   d. Right-align labels (using text-align of the parent/containing element)
   e. Add border to the *form* element (same as step but with different color)
   f. Center-align the form on the screen (using *width* and *margin-left* and *margin-right* set as *auto*)
3 Add interactions for its OK and Cancel buttons using JS
   a. Add click event for both OK and Cancel buttons (using *onclick* element event attribute)
   b. When OK button is clicked, show alert with “Login is clicked” message (using *alert(“yourMessage”)*)
   c. When Cancel button is clicked, show alert with “Cancel is clicked” message (same as step b)

Home Task

1 Enhance the *Login Form* created in Lab Task above using HTML
   a. Add Remember Me checkbox
   b. Add an Image beside the form
2 Style the Login Form using CSS
   a. Change the background-color of the textboxes when focus is set
   b. Set the image width and height
   c. Position the image on the top-left corner using absolute and relative position combinations
3 Extend the OK and Cancel button functionality by using JS
   a. Create a function called *findInputs()* and return all the inputs in an array. Call the function from the OK button click and assign the output to an array
   b. Create a function called *validateInputs()* and return true or false after looping through each input element obtained from the *findInputs()* function and verifying if mandatory Login ID and Password are filled in or not. Call the *validateInputs()* function from the OK button click after *findInputs()* function
   c. Show message to enter input if *validateInputs()* output is false and return from the click event.
   d. Create a function called *login()* and check if Remember Me is checked or not. If it is checked, show message in an alert as “Logged in successfully with Remember

Me marked as set", otherwise, just show "Logged in successfully". Call the *login()* function on button click

e. Create a function called *clear()* and clear all the inputs. Call the function from Cancel button when it is clicked.