Department of
Computer Science and Engineering

Title: Introduction to JavaScript (JS)

Web Programming Lab

CSE 301

Green University of Bangladesh

1 Objective(s)

- To be familiar with JavaScript (JS).
- To be familiar with JS syntax, variable, operator, array, string, function, object, and event.

2 JavaScript Introduction

One of the three languages that every web developer should master is JavaScript:

- HTML to define the content of web pages
- CSS to specify the layout of web pages
- JavaScript to program the behavior of web pages

HTML Content Can Be Changed by JavaScript. Two common forms of getting HTML content using JS:

- getElementById()
- getElementByClass()

The following example "finds" an HTML element (with id="demo") and changes its content (innerHTML) to "Hello JavaScript":

```
1     document.getElementById("demo").innerHTML = "Hello JavaScript";
```

```
1  In this illustration, JavaScript modifies the value of an image tag's src (
       source) attribute:
2  <!DOCTYPE html>
3  <html>
4  <body>
5  <h2>What Can JavaScript Do? </h2>
6  <p>JavaScript can change HTML attribute values. </p>
7  <p>In this case JavaScript changes the value of the src (source) attribute of an
       image.</p>
8  <button onclick="document. getElementById('myImage').src='pic_bulbon.gif'">Turn
       on the light</button>
9  <img id="myImage" src="pic_bulboff.gif" style="width:100px">
10
11 <button onclick="document. getElementById('myImage').src='pic_bulboff.gif'">Turn
       off the light</button>
12
13 </body>
14 </html>
```

Where to use JavaScript in webpage?
JavaScript code is inserted using <script> and </script> tags.

```
1  <script>
2  document.getElementById("demo").innerHTML = "My First JavaScript";
3  </script>
```

3 JavaScript Variables

There are Four popular ways of declaring variable in js:

- Using var
  This type of variables can be redeclared and reassigned again.

- Using let
  This type of variables cant be redeclared but can be reassigned again.
- Using const
  This type of variables cant be redeclared and cant be reassigned again.
- Using nothing

What are Variables?
Variables are data storage containers (storing data values).

```
Examples:
var x = 5;
var y = 6;
var z = x + y;
let x = 5;
let y = 6;
let z = x + y;
const price1 = 5;
x = 5;
y = 6;
z = x + y;
```

4 JavaScript Operators

Types of JavaScript Operators
There are different types of JavaScript operators:

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators
- Conditional Operators
- Type Operators

The Assignment Operator (=) assigns a value to a variable.

```
// Assign the value 5 to x
let x = 5;
// Assign the value 2 to y
let y = 2;
// Assign the value x + y to z:
let z = x + y;
```

5 JavaScript if, else, and else if

Conditional statements are used to perform different actions based on different conditions. In JavaScript we have the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

5.1 The if/else Statement

Use the if statement to specify a block of JavaScript code to be executed if a condition is true.

Syntax

```
    if (condition)
{
  //  block of code to be executed if the condition is true
}
```

Use the else statement to specify a block of code to be executed if the condition is false.

Syntex

```
    if (condition) {
  //  block of code to be executed if the condition is true
} else {
  //  block of code to be executed if the condition is false
}
```

```
Example:
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript if .. else</h2>

<p>A time-based greeting:</p>

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

<script>
const hour = new Date().getHours();
let greeting;

if (hour < 18) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}

document.getElementById("demo").innerHTML = greeting;
</script>

</body>
</html>
```

6 Function

A JavaScript function is a block of code designed to perform a particular task. A JavaScript function is executed when something invokes it (calls it).

```
// Function to compute the product of p1 and p2
function myFunction(p1, p2) {
  return p1 * p2;
}
```

6.1 JavaScript Function Syntax

A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses (). Function names can contain letters, digits, underscores, and dollar signs (same rules as variables). The parentheses may include parameter names separated by commas: (parameter1, parameter2, ...)

The code to be executed, by the function, is placed inside curly brackets:

```
function name(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Function parameters are listed inside the parentheses () in the function definition. Function arguments are the values received by the function when it is invoked. Inside the function, the arguments (the parameters) behave as local variables.

6.2 Function Invocation

The code inside the function will execute when "something" invokes (calls) the function:

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

6.3 Function Return

When JavaScript reaches a return statement, the function will stop executing. If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement. Functions often compute a return value. The return value is "returned" back to the "caller":

```
let x = myFunction(4, 3);   // Function is called, return value will end up in x

function myFunction(a, b) {
  return a * b;             // Function returns the product of a and b
}
```

6.4 The () Operator Invokes the Function

Using the example above, toCelsius refers to the function object, and toCelsius() refers to the function result. Accessing a function without () will return the function object instead of the function result.

```
function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
document.getElementById("demo").innerHTML = toCelsius;
```

6.5 Functions Used as Variable Values

Functions can be used the same way as you use variables, in all types of formulas, assignments, and calculations. Instead of using a variable to store the return value of a function:

```
let x = toCelsius(77);
let text = "The temperature is " + x + " Celsius";
```

You can use the function directly, as a variable value:

```
let text = "The temperature is " + toCelsius(77) + " Celsius";
```

7 JavaScript Objects

You have already learned that JavaScript variables are containers for data values. This code assigns a simple value (Fiat) to a variable named car:

```
let car = "Fiat";
```

Objects are variables too. But objects can contain many values. This code assigns many values (Fiat, 500, white) to a variable named car:

```
const car = {type:"Fiat", model:"500", color:"white"};
```

The values are written as name:value pairs (name and value separated by a colon). It is a common practice to declare objects with the const keyword.

7.1 Object Definition

You define (and create) a JavaScript object with an object literal:

```
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

Spaces and line breaks are not important. An object definition can span multiple lines:

```
const person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```

7.2 Object Properties

The name:values pairs in JavaScript objects are called properties:

Property	Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

7.3 Accessing Object Properties

You can access object properties in two ways:

```
objectName.propertyName # person.lastName;
objectName["propertyName"] # person["lastName"];
```

7.4 Object Methods

Objects can also have methods. Methods are actions that can be performed on objects. Methods are stored in properties as function definitions.

```
const person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

In the example above, this refers to the person object. I.E. this.firstName means the firstName property of this. I.E. this.firstName means the firstName property of person.

7.5 Accessing Object Methods

You access an object method with the following syntax:

```
objectName.methodName() # name = person.fullName();
```

8 JavaScript Event

HTML events are "things" that happen to HTML elements. When JavaScript is used in HTML pages, JavaScript can "react" on these events.

8.1 HTML Events

An HTML event can be something the browser does, or something a user does. Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

Often, when events happen, you may want to do something. JavaScript lets you execute code when events are detected. HTML allows event handler attributes, with JavaScript code, to be added to HTML elements. In the following example, an onclick attribute (with code), is added to a <button> element:

```
<button onclick="document.getElementById('demo').innerHTML = Date()">The time is
    ?</button>
```

In the example above, the JavaScript code changes the content of the element with id="demo". In the next example, the code changes the content of its own element (using this.innerHTML):

```
<button onclick="this.innerHTML = Date()">The time is?</button>
```

JavaScript code is often several lines long. It is more common to see event attributes calling functions:

```
<button onclick="displayDate()">The time is?</button>
```

8.2 Common HTML Events

Here is a list of some common HTML events:

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

8.3 JavaScript Event Handlers

Event handlers can be used to handle and verify user input, user actions, and browser actions:

- Things that should be done every time a page loads
- Things that should be done when the page is closed
- Action that should be performed when a user clicks a button

- Content that should be verified when a user inputs data

Many different methods can be used to let JavaScript work with events:

- HTML event attributes can execute JavaScript code directly
- HTML event attributes can call JavaScript functions
- You can assign your own event handler functions to HTML elements
- You can prevent events from being sent or being handled

9 String

JavaScript strings are for storing and manipulating text. A JavaScript string is zero or more characters written inside quotes.

```
let text = "John Doe";
```

You can use single or double quotes:

```
let carName1 = "Volvo XC60";  // Double quotes
let carName2 = 'Volvo XC60';  // Single quotes
```

You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

```
let answer1 = "It's alright";
let answer2 = "He is called 'Johnny'";
let answer3 = 'He is called "Johnny"';
```

9.1 String Length

To find the length of a string, use the built-in length property:

```
let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
let length = text.length;
```

9.2 JavaScript Strings as Objects

Normally, JavaScript strings are primitive values, created from literals:

```
let x = "John";
```

But strings can also be defined as objects with the keyword new:

```
let y = new String("John");
```

Do not create Strings objects. The new keyword complicates the code and slows down execution speed.

9.3 String Properties and Methods

Normally, strings like "John Doe", cannot have methods or properties because they are not objects. But with JavaScript, methods and properties are also available to strings, because JavaScript treats strings as objects when executing methods and properties.

9.3.1 JavaScript String Methods

There are many JavaScript default methods. Some of them are listed below:

Name	Description
charAt()	Returns the character at a specified index (position)
concat()	Returns two or more joined strings
indexOf()	Returns the index (position) of the first occurrence of a value in a string
slice()	Extracts a part of a string and returns a new string
split()	Splits a string into an array of substrings
substr()	Extracts a number of characters from a string, from a start index (position)

Splitting a string using the split() method is demonstrated below:

```
<script>
let text = "How are you doing today?";
const myArray = text.split(" ");

document.getElementById("demo").innerHTML = myArray;
</script>

# split() splits the string into an array of substrings, and returns the array
    as: How, are, you, doing, today?
```

10 Array

An array is a special variable, which can hold more than one value:

```
const cars = ["Saab", "Volvo", "BMW"];
```

Using an array literal is the easiest way to create a JavaScript Array.

```
const array_name = [item1, item2, ...]; # const cars = ["Saab", "Volvo", "BMW"];
```

It is a common practice to declare arrays with the const keyword. Spaces and line breaks are not important. A declaration can span multiple lines:

```
const cars = [
  "Saab",
  "Volvo",
  "BMW"
];
```

You can also create an array, and then provide the elements:

```
const cars = [];
cars[0]= "Saab";
cars[1]= "Volvo";
cars[2]= "BMW";
```

The following example also creates an Array, and assigns values to it:

```
const cars = new Array("Saab", "Volvo", "BMW");
```

The two examples above do exactly the same. There is no need to use new Array(). For simplicity, readability and execution speed, use the array literal method.

10.1 Accessing Array Elements

You access an array element by referring to the index number:

```
const cars = ["Saab", "Volvo", "BMW"];
let car = cars[0];
```

Array indexes start with 0. [0] is the first element. [1] is the second element.

10.2 Changing an Array Element

This statement changes the value of the first element in cars:

```
const cars = ["Saab", "Volvo", "BMW"];
cars[0] = "Opel";
```

10.3 Array Properties and Methods

The real strength of JavaScript arrays are the built-in array properties and methods:

```
cars.length // Returns the number of elements
cars.sort() // Sorts the array
```

10.3.1 The length Property

The length property of an array returns the length of an array (the number of array elements).

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let length = fruits.length;
```

The length property is always one more than the highest array index.

10.3.2 Adding Array Elements

The easiest way to add a new element to an array is using the push() method:

```
const fruits = ["Banana", "Orange", "Apple"];
fruits.push("Lemon");  // Adds a new element (Lemon) to fruits
```

10.4 Associative Arrays

Many programming languages support arrays with named indexes. Arrays with named indexes are called associative arrays (or hashes). JavaScript does not support arrays with named indexes. In JavaScript, arrays always use numbered indexes.

```
const person = [];
person["firstName"] = "John";
person["lastName"] = "Doe";
person["age"] = 46;
person.length;     // Will return 0
person[0];         // Will return undefined
```

10.5 JavaScript new Array()

JavaScript has a built-in array constructor new Array(). But you can safely use [] instead. These two different statements both create a new empty array named points:

```
const points = new Array();
const points = [];
```

These two different statements both create a new array containing 6 numbers:

```
const points = new Array(40, 100, 1, 5, 25, 10);
const points = [40, 100, 1, 5, 25, 10];
```

11 Discussion & Conclusion

Based on the focused objective(s) to understand the concepts of JavaScript, the additional lab exercise made me more confident in the fulfillment of the objectives(s).

12 Lab Task (Please implement yourself and show the output to the instructor)

Write a JavaScript function that reverses a number.

- Example x = 32243
- Expected Output : 34223

13 Lab Exercise (Submit as a report)

Write a JavaScript function that checks whether a passed string is palindrome or not? Try to incorporate the concepts of function, object, and event in your solution.

Note: A palindrome is word, phrase, or sequence that reads the same backward as forward, e.g., madam.

14 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.