LAB # 05

Statement Purpose:

To familiarize the students with

- ⊡ JavaScript Syntax
- ⊡ Variables, Let, Const
- ⊡ Operators
  - Arithmetic
  - Assignment
- ⊡ Data Types
- ⊡ Arrays

JavaScript:

JavaScript (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well. It is a prototype-based, multi-paradigm scripting language that is dynamic, and supports object-oriented, imperative, and functional programming styles.

JavaScript runs on the client side of the web, which can be used to design / program how the web pages behave on the occurrence of an event. JavaScript is an easy to learn and also powerful scripting language, widely used for controlling web page behavior.

Contrary to popular misconception, JavaScript is not "Interpreted Java". In a nutshell, JavaScript is a dynamic scripting language supporting prototype-based object construction. The basic syntax is intentionally like both Java and C++ to reduce the number of new concepts required to learn the language. Language constructs, such as if statements, for and while loops, and switch and try ... catch blocks function the same as in these languages (or nearly so).

JavaScript can function as both a procedural and an object-oriented language. Objects are created programmatically in JavaScript, by attaching methods and properties to otherwise empty objects at run time, as opposed to the syntactic class definitions common in compiled languages like C++ and Java. Once an object has been constructed it can be used as a blueprint (or prototype) for creating similar objects.

JavaScript Where To

The <script> Tag

In HTML, JavaScript code is inserted between <script> and </script> tags.

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

JavaScript Functions and Events

A JavaScript function is a block of JavaScript code, that can be executed when "called" for.

For example, a function can be called when an **event** occurs, like when the user clicks a button.

JavaScript in <head> or <body>

You can place any number of scripts in an HTML document.

Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

JavaScript in <head>

```
<!DOCTYPE html>

<html>

<head>

<script>

function myFunction() {

  document.getElementById("demo").innerHTML = "Paragraph changed.";

}

</script>

</head>

<body>

<h2>Demo JavaScript in Head</h2>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

</body>

</html>
```

JavaScript in <body>

```
<!DOCTYPE html>
<html>
<body>
<h2>Demo JavaScript in Body</h2>
<p id="demo">A Paragraph.</p>
<button type="button" onclick="myFunction()">Try it</button>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</body>
</html>
```

External JavaScript

```
<script src="myScript.js"></script>
```

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

Using innerHTML

To access an HTML element, JavaScript can use the document.getElementById(id) method.

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

```
<body>

<h2>My First Web Page</h2>

<p>My First Paragraph.</p>

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

<script>

document.getElementById("demo").innerHTML = 5 + 6;

</script>

</body>

</html>
```

Using document.write()

For testing purposes, it is convenient to use document.write():

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
document.write(5 + 6);
</script>

</body>
</html>
```

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<button type="button" onclick="document.write(5 + 6)">Try it</button>
```

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

Using window.alert()

You can use an alert box to display data:

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```

Using console.log()

For debugging purposes, you can call the console.log() method in the browser to display data.

Example

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

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

JavaScript Print

JavaScript does not have any print object or print methods.

You cannot access output devices from JavaScript.

The only exception is that you can call the window.print() method in the browser to print the content of the current window.

Example

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

<button onclick="window.print()">Print this page</button>

</body>
</html>
```

JavaScript Statements

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

<h2>JavaScript Statements</h2>

<p>A <b>JavaScript program</b> is a list of <b>statements</b> to be executed by a computer.</p>

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

<script>
let x, y, z;  // Statement 1
x = 5;        // Statement 2
y = 6;        // Statement 3
z = x + y;    // Statement 4
document.getElementById("demo").innerHTML =
"The value of z is " + z + ".";
</script>

</body>
</html>
```

JavaScript Statements

JavaScript statements are composed of: Values, Operators, Expressions, Keywords, and Comments.

Semicolons ;

Semicolons separate JavaScript statements.

Add a semicolon at the end of each executable statement:

Examples

```
let a, b, c;  // Declare 3 variables
a = 5;        // Assign the value 5 to a
b = 6;        // Assign the value 6 to b
c = a + b;    // Assign the sum of a and b to c
```

When separated by semicolons, multiple statements on one line are allowed:

```
a = 5; b = 6; c = a + b;
```

JavaScript Syntax

JavaScript syntax is the set of rules, how JavaScript programs are constructed:

```
// How to create variables:
var x;
let y;

// How to use variables:
x = 5;
y = 6;
let z = x + y;
```

JavaScript Values

The JavaScript syntax defines two types of values:

- Fixed values
- Variable values

Fixed values are called **Literals**.

Variable values are called **Variables**.

JavaScript Literals

The two most important syntax rules for fixed values are:

1. **Numbers** are written with or without decimals:

```
<h2>JavaScript Numbers</h2>

<p>Number can be written with or without decimals.</p>

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

<script>

document.getElementById("demo").innerHTML = 10.50;

</script>
```

Strings are text, written within double or single quotes:

```
"John Doe"

'John Doe'
```

JavaScript Variables

In a programming language, **variables** are used to **store** data values.

JavaScript uses the keywords var, let and const to **declare** variables.

An **equal sign** is used to **assign values** to variables.

In this example, x is defined as a variable. Then, x is assigned (given) the value 6:

```
let x;
x = 6;
```

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

<script>

let x;

x = 6;

document.getElementById("demo").innerHTML = x;

</script>
```

JavaScript Operators

JavaScript uses **arithmetic operators** (+ - * /) to **compute** values:

```
(5 + 6) * 10
```

JavaScript uses an **assignment operator** (=) to **assign** values to variables:

```
let x, y;
x = 5;
y = 6;
```

JavaScript Identifiers / Names

Identifiers are JavaScript names.

Identifiers are used to name variables and keywords, and functions.

The rules for legal names are the same in most programming languages.

A JavaScript name must begin with:

- A letter (A-Z or a-z)
- A dollar sign ($)
- Or an underscore (_)

Subsequent characters may be letters, digits, underscores, or dollar signs.

JavaScript is Case Sensitive

All JavaScript identifiers are **case sensitive**.

The variables lastName and lastname, are two different variables:

```
<h2>JavaScript is Case Sensitive</h2>

<p>Try to change lastName to lastname.</p>

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

<script>

let lastname, lastName;

lastName = "Doe";

lastname = "Peterson";

document.getElementById("demo").innerHTML = lastName;

</script>
```

JavaScript and Camel Case

Historically, programmers have used different ways of joining multiple words into one variable name:

Hyphens:

first-name, last-name, master-card, inter-city.

Hyphens are not allowed in JavaScript. They are reserved for subtractions.

Underscore:

first_name, last_name, master_card, inter_city.

Upper Camel Case (Pascal Case):

FirstName, LastName, MasterCard, InterCity.

Lower Camel Case:

JavaScript programmers tend to use camel case that starts with a lowercase letter:

firstName, lastName, masterCard, interCity.

JavaScript Variables

4 Ways to Declare a JavaScript Variable:

- Using var
- Using let
- Using const
- Using nothing

What are Variables?

Variables are containers for storing data (storing data values).

In this example, x, y, and z, are variables, declared with the var keyword:

```
<h1>JavaScript Variables</h1>

<p>In this example, x, y, and z are variables.</p>

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

<script>
```

```
var x = 5;

var y = 6;

var z = x + y;

document.getElementById("demo").innerHTML = "The value of z is: " + z;

</script>
```

```
<body>
<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!</button>

</body>
```

Activity 1:

```
<p id="demo">Click the button to display the cookies associated with this document.</p>

<button onclick="myFunction()">Try it</button>

<script>

function myFunction() {

  document.getElementById("demo").innerHTML = "Cookies associated with this document: " + document.cookie;

}
```

Activity 2:

```
<script>

function myFunction() {

  document.getElementById("demo").innerHTML = document.domain;

}

</script>
```

Activity 3:

```
<script>

document.getElementById("demo").innerHTML = document.lastModified;
```

```
</script>
```

Activity 4:

```
<script>
document.getElementById("demo").innerHTML = "The title of this document is: " +
document.title;
</script>
```

Activity 5:

```
<p>Click the button to open a new window and add some content.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction() {
  var w = window.open();
  w.document.open();
  w.document.write("<h2>Hello World!</h2>");
  w.document.close();
}
</script>
```

Activity 6:

```
<script>
document.getElementById("demo").innerHTML = document.URL
</script>
Query Selector
<h1>The Document Object</h1>
<h2>The querySelector() Method</h2>
<h3>Add a background color to the first p element:</h3>
<p>This is a p element.</p>
<p>This is a p element.</p>
<script>
document.querySelector("p").style.backgroundColor = "red";
</script>
```

Activity 7:

```
<h1>The Document Object</h1>
<h2>The querySelectorAll() Method</h2>

<p>Add a background color all elements with class="example":</p>
<h2 class="example">A heading</h2>
<p class="example">A paragraph.</p>

<script>
const nodeList = document.querySelectorAll(".example");
for (let i = 0; i < nodeList.length; i++) {
  nodeList[i].style.backgroundColor = "red";
}
</script>
```

Task 1: Write a JavaScript code to Change HTML Attribute Values. This should apply on any image e.g, on and off the bulb .

Task 2: Write a JavaScript Code to Change HTML Styles (CSS)

Task 3: Write a JavaScript code to Hide and show the HTML Elements