EXPERIMENT NO. 05

DATE OF PERFORMANCE: **GRADE:**

DATE OF ASSESSMENT: **SIGNATURE OF LECTURER/ TTA:**

AIM: Introduction to Java Script.

THEORY:

JAVASCRIPT:

An object-oriented computer programming language commonly used to create interactive effects within web browsers.

JavaScript is the programming language of HTML and the Web. Programming makes computers do what you want them to do.

JavaScript is a scripting language, that is, a lightweight programming language that is interpreted by the browser engine when the web page is loaded.

WHY STUDY JAVASCRIPT?

JavaScript is one of the 3 languages all web developers must learn:

1. **HTML to define the content of web pages**
2. **CSS to specify the layout of web pages**
3. **JavaScript to program the behavior of web pages.**

JAVASCRIPT CAN CHANGE HTML CONTENT:

One of many JavaScript HTML methods is getElementById().

This example uses the method to "find" an HTML element (with id="demo") and changes the element content (innerHTML) to "Hello JavaScript":

Example: document.getElementById("demo").innerHTML = "Hello JavaScript";

JAVASCRIPT WHERE TO:

JavaScript can be placed in the <body> and the <head> sections of an HTML page.

The <script> Tag

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

Example:

```
<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 "asked" for. For example, a function can be executed 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>:

A JavaScript function is placed in the <head> section of an HTML page.

The function is invoked (called) when a button is clicked:

Example

```
<!DOCTYPE html>
<html>

<head>
<script>
function myFunction() {
   document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
```

```
<body>

<h1>My Web Page</h1>

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

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

</body>
</html>
```

JAVASCRIPT IN <BODY>:

JavaScript function is placed in the <body> section of an HTML page. The function is invoked (called) when a button is clicked:

Example

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

<h1>My Web Page</h1>

<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:

Scripts can also be placed in external files:

myScript.js

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

External scripts are practical when the same code is used in many different web pages.

JavaScript files have the file extension .js.

To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

Example:

```
<!DOCTYPE html>
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

EXTERNAL JAVASCRIPT ADVANTAGES:

Placing JavaScripts in external files has some advantages:

- **It separates HTML and code**
- **It makes HTML and JavaScript easier to read and maintain**
- **Cached JavaScript files can speed up page loads.**

JAVASCRIPT OUTPUT:

JavaScript does NOT have any built-in print or display functions.

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

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

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 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>
```

USING INNERHTML:

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

The id attribute defines the HTML element. The innerHTML property defines the HTML content:

Example

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

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

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

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

USING CONSOLE.LOG():

In your browser, you can use the console.log() method to display data.

Activate the browser console with F12, and select "Console" in the menu.

Example

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

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

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

</body>
</html>

JAVASCRIPT PROGRAMS:

A computer program is a list of "instructions" to be "executed" by the computer.

In a programming language, these program instructions are called statements.

JavaScript is a programming language.

JavaScript statements are separated by semicolons.

Example

```
var x = 5;
var y = 6;
var z = x + y;
```

JAVASCRIPT STATEMENTS:

JavaScript statements are composed of:

Values, Operators, Expressions, Keywords, and Comments.

JAVASCRIPT VALUES:

The JavaScript syntax defines two types of values: Fixed values and variable values.

Fixed values are called literals. Variable values are called variables.

JAVASCRIPT LITERALS:

The most important rules for writing fixed values are:

Numbers are written with or without decimals:

10.50
1001

JAVASCRIPT OPERATORS:

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

```
var x = 5;
var y = 6;
```

JAVASCRIPT EXPRESSIONS:

An expression is a combination of values, variables, and operators, which computes to a value.

The computation is called an evaluation.

For example, 5 * 10 evaluates to 50:

```
5 * 10
```

JavaScript Keywords

JavaScript keywords are used to identify actions to be performed.

The var keyword tells the browser to create a new variable:

```
var x = 5 + 6;
var y = x * 10;
```

JAVASCRIPT COMMENTS:

Not all JavaScript statements are "executed".

Code after double slashes // or between /* and */ is treated as a comment.

Comments are ignored, and will not be executed:

```
var x = 5;   // I will be executed

// var x = 6;   I will NOT be executed
```

JAVASCRIPT IDENTIFIERS:

Identifiers are names.

In JavaScript, identifiers are used to name variables (and keywords, and functions, and labels).

JavaScript is Case Sensitive

All JavaScript identifiers are case sensitive.

The variables lastName and lastname, are two different variables.

```
lastName = "Doe";
lastname = "Peterson";
```

JAVASCRIPT PROGRAMS:

Most JavaScript programs contain many JavaScript statements.

The statements are executed, one by one, in the same order as they are written.

Example

```
var x = 5;
var y = 6;
var z = x + y;
document.getElementById("demo").innerHTML = z;
```

SEMICOLONS :

Semicolons separate JavaScript statements.

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

JavaScript White Space

JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.

```
var person = "Hege";
var person="Hege";
```

JAVASCRIPT CODE BLOCKS:

JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.

The purpose of code blocks is to define statements to be executed together.

Example

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Hello Dolly.";
  document.getElementById("myDIV").innerHTML = "How are you?";
}
```

JAVASCRIPT KEYWORDS:

JavaScript statements often start with a keyword to identify the JavaScript action to be performed.

Keyword	Description
break	Terminates a switch or a loop
continue	Jumps out of a loop and starts at the top
debugger	Stops the execution of JavaScript, and calls (if available) the debugging function
do ... while	Executes a block of statements, and repeats the block, while a condition is true
for	Marks a block of statements to be executed, as long as a condition is true
function	Declares a function
if ... else	Marks a block of statements to be executed, depending on a condition
return	Exits a function
switch	Marks a block of statements to be executed, depending on different cases
try ... catch	Implements error handling to a block of statements
var	Declares a variable

JAVASCRIPT VARIABLES:

JavaScript variables are containers for storing data values.

Example

```
var x = 5;
var y = 6;
var z = x + y;
```

From the example above, you can expect:

- **x stores the value 5**
- **y stores the value 6**
- **z stores the value 11**

JAVASCRIPT DATA TYPES:

JAVASCRIPT DATA TYPES:

JavaScript variables can hold many data types: numbers, strings, arrays, objects and more:

```
var length = 16;                    // Number
var lastName = "Johnson";           // String
var cars = ["Saab", "Volvo", "BMW"];    // Array
var x = {firstName:"John", lastName:"Doe"};    // Object
```

THE CONCEPT OF DATA TYPES:

To be able to operate on variables, it is important to know something about the type.

Without data types, a computer cannot safely solve this:

```
var x = 16 + "Volvo";
```

JavaScript will treat the example above as:

```
var x = "16" + "Volvo";
```

When adding a number and a string, JavaScript will treat the number as a string.

JavaScript evaluates expressions from left to right. Different sequences can produce different results:

```
var x = 16 + 4 + "Volvo";
```

Result:

20Volvo

JAVASCRIPT HAS DYNAMIC TYPES:

JavaScript has dynamic types. This means that the same variable can be used as different types:

Example

```
var x;               // Now x is undefined
var x = 5;           // Now x is a Number
var x = "John";      // Now x is a String
```

JAVASCRIPT STRINGS:

A string (or a text string) is a series of characters like "John Doe".

Strings are written with quotes. You can use single or double quotes:

Example

```
var carName = "Volvo XC60";   // Using double quotes
var carName = 'Volvo XC60';   // Using single quotes
```

JAVASCRIPT NUMBERS:

JavaScript has only one type of numbers.

Numbers can be written with, or without decimals:

Example

```
var x1 = 34.00;     // Written with decimals
var x2 = 34;        // Written without decimals
```

JAVASCRIPT BOOLEANS:

Booleans can only have two values: true or false.

Example

```
var x = true;
var y = false;
```

JAVASCRIPT ARRAYS:

JavaScript arrays are written with square brackets.

Array items are separated by commas.

The following code declares (creates) an array called cars, containing three items (car names):

Example:

var cars = ["Saab", "Volvo", "BMW"];

JAVASCRIPT OBJECTS:

JavaScript objects are written with curly braces.

Object properties are written as name:value pairs, separated by commas.

Example

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

THE TYPEOF OPERATOR:

You can use the JavaScript typeof operator to find the type of a JavaScript variable:

Example

```
typeof "John"                // Returns string
typeof 3.14                  // Returns number
typeof false                 // Returns boolean
typeof [1,2,3,4]             // Returns object (not array, see note below)
typeof {name:'John', age:34} // Returns object
```

UNDEFINED:

In JavaScript, a variable without a value, has the value undefined. The typeof is also undefined.

Any variable can be emptied, by setting the value to undefined. The type will also be undefined.

Example

person = undefined; // Value is undefined, type is undefined

NULL:

In JavaScript null is "nothing". It is supposed to be something that doesn't exist.

Unfortunately, in JavaScript, the data type of null is an object.

var person = null; // Value is null, but type is still an object

JAVASCRIPT OPERATORS:

Let us take a simple expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and '+' is called the operator. JavaScript supports the following types of operators.

- **Arithmetic Operators**
- **Comparision Operators**
- **Logical (or Relational) Operators**
- **Assignment Operators**
- **Conditional (or ternary) Operators**

ARITHMETIC OPERATORS:

Sr.No	Operator and Description
1	**+ (Addition)** Adds two operands. **Ex:** A + B will give 30
2	**- (Subtraction)** Subtracts the second operand from the first. **Ex:** A - B will give -10
3	*** (Multiplication)** Multiply both operands. **Ex:** A * B will give 200
4	**/ (Division)** Divide the numerator by the denominator. **Ex:** B / A will give 2
5	**% (Modulus)** Outputs the remainder of an integer division.**Ex:** B % A will give 0

6	**++ (Increment)** Increases an integer value by one. **Ex:** A++ will give 11
7	**-- (Decrement)** Decreases an integer value by one. **Ex:** A-- will give 9

COMPARISON OPERATORS:

Sr.No	Operator and Description
1	**= = (Equal)** **Checks if the value of two operands are equal or not, if yes, then the condition becomes true.** **Ex: (A == B) is not true.**
2	**!= (Not Equal)** **Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true.** **Ex: (A != B) is true.**
3	**> (Greater than)** **Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true.** **Ex: (A > B) is not true.**
4	**< (Less than)** **Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true.** **Ex: (A < B) is true.**
5	**>= (Greater than or Equal to)** **Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true.** **Ex: (A >= B) is not true.**

6	**<= (Less than or Equal to)** **Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true.** **Ex: (A <= B) is true.**

BITWISE OPERATORS:

JavaScript supports the following bitwise operators.

Assume variable A holds 2 and variable B holds 3, then

Sr.No	Operator and Description
1	**& (Bitwise AND)** **It performs a Boolean AND operation on each bit of its integer arguments.** **Ex: (A & B) is 2.**
2	**| (BitWise OR)** **It performs a Boolean OR operation on each bit of its integer arguments.** **Ex: (A | B) is 3.**
3	**^ (Bitwise XOR)** **It performs a Boolean exclusive OR operation on each bit of its integer arguments. Exclusive OR means that either operand one is true or operand two is true, but not both.** **Ex: (A ^ B) is 1.**
4	**~ (Bitwise Not)** **It is a unary operator and operates by reversing all the bits in the operand.** **Ex: (~B) is -4.**
5	**<< (Left Shift)** **It moves all the bits in its first operand to the left by the number of places specified in the second operand. New bits are filled with zeros. Shifting a value left by one position is equivalent to multiplying it by 2, shifting two positions is equivalent to**

	multiplying by 4, and so on. **Ex: (A << 1) is 4.**
6	**>> (Right Shift)** **Binary Right Shift Operator. The left operand's value is moved right by the number of bits specified by the right operand.** **Ex: (A >> 1) is 1.**
7	**>>> (Right shift with Zero)** **This operator is just like the >> operator, except that the bits shifted in on the left are always zero.Ex: (A >>> 1) is 1.**

MISCELLANEOUS OPERATOR:

CONDITIONAL OPERATOR (? :)

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

Sr.No	Operator and Description
1	**? : (Conditional)** **If Condition is true? Then value X : Otherwise value Y**

PROGRAM 1:JAVASCRIPT ARITHMETIC OPERATOR.

```
<html>
  <body>

    <script type="text/javascript">
      <!--
        var a = 33;
        var b = 10;
        var c = "Test";
        var linebreak = "<br />";

        document.write("a + b = ");
        result = a + b;
```

```
        document.write(result);
        document.write(linebreak);

        document.write("a - b = ");
        result = a - b;
        document.write(result);
        document.write(linebreak);

        document.write("a / b = ");
        result = a / b;
        document.write(result);
        document.write(linebreak);

        document.write("a % b = ");
        result = a % b;
        document.write(result);
        document.write(linebreak);

        a = ++a;
        document.write("++a = ");
        result = ++a;
        document.write(result);
        document.write(linebreak);

        b = --b;
        document.write("--b = ");
        result = --b;
        document.write(result);
        document.write(linebreak);
      //-->
    </script>

    Set the variables to different values and then try...
  </body>
</html>
```

OUTPUT:

PROGRAM 2: USE OF COMPARISON OPERATOR.

```
<html>
  <body>

    <script type="text/javascript">
        var a = 10;
        var b = 20;
        var linebreak = "<br />";

        document.write("(a == b) => ");
        result = (a == b);
        document.write(result);
        document.write(linebreak);

        document.write("(a < b) => ");
        result = (a < b);
        document.write(result);
        document.write(linebreak);

        document.write("(a > b) => ");
        result = (a > b);
        document.write(result);
        document.write(linebreak);
```

```
        document.write("(a != b) => ");
        result = (a != b);
        document.write(result);
        document.write(linebreak);

        document.write("(a >= b) => ");
        result = (a >= b);
        document.write(result);
        document.write(linebreak);

        document.write("(a <= b) => ");
        result = (a <= b);
        document.write(result);
        document.write(linebreak);
          </script>

    Set the variables to different values and different operators and then try...
  </body>
</html>
```

OUTPUT:

PROGRAM 3: USE OF BITWISE OPERATOR.

```
<html>

  <body>

    <script type="text/javascript">
      <!--
        var a = 2; // Bit presentation 10
        var b = 3; // Bit presentation 11
        var linebreak = "<br />";

        document.write("(a & b) => ");
        result = (a & b);
        document.write(result);
        document.write(linebreak);

        document.write("(a | b) => ");
        result = (a | b);
        document.write(result);
        document.write(linebreak);

        document.write("(a ^ b) => ");
        result = (a ^ b);
        document.write(result);
        document.write(linebreak);

        document.write("(~b) => ");
        result = (~b);
        document.write(result);
        document.write(linebreak);

        document.write("(a << b) => ");
        result = (a << b);
        document.write(result);
        document.write(linebreak);

        document.write("(a >> b) => ");
        result = (a >> b);
        document.write(result);
        document.write(linebreak);
      //-->
    </script>

  </body>
</html>
```

OUTPUT:

PROGRAM 4: USE OF TERNARY OPERATOR.

```
<html>
  <body>

    <script type="text/javascript">
      <!--
         var a = 10;
         var b = 20;
         var linebreak = "<br />";

         document.write ("((a > b) ? 100 : 200) => ");
         result = (a > b) ? 100 : 200;
         document.write(result);
         document.write(linebreak);

         document.write ("((a < b) ? 100 : 200) => ");
```

```
        result = (a < b) ? 100 : 200;
        document.write(result);
        document.write(linebreak);
      //-->
    </script>

    <p>Set the variables to different values and different operators and then try...</p>
  </body>
</html>
```

OUTPUT:

PROGRAM 5: USE OF DATA TYPE.

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

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

<script>
var x = "Volvo" + 16 + 4;
document.getElementById("demo").innerHTML = x;
</script>

</body>
</html>
```

OUTPUT: