Sinhgad Technical Education Society's

RMD Sinhgad School of Engineering, Warje, Pune

Department of Electronics & Telecommunication Engineering

LABORATORY MANUAL

For

JAVASCRIPT

B.E. (E & TC)
(2019 COURSE)

Prepared By

Mrs. Varsha Nanavare

Sinhgad Technical Education Society

RMD Sinhgad School of Engineering, Warje, PUNE.

Department of Electronics & Telecommunication Engineering

404184 (C) : Java Script (Elective - III) Lab

Course Objectives, Laboratory Objectives & Outcomes

❖ *Course Objectives*

1. To learn the syntax and semantics of Java script.
2. To understand the data types and variables in Java script.
3. To learn how functions and objects are used in Java script.
4. To learn how to use regular expressions in java script for handling various string operations.
5. To understand the concept of object models and event handling in java script programs.
6. To learn the use of java script for controlling Windows and form handling

❖ *Laboratory Objective*

The objective of this lab is to Design and develop Web applications using different data types and variables in JavaScript, regular expression, function, object and Event Handling in JavaScript, so that students get an idea of JavaScript programming.

❖ *Laboratory Outcomes*

On completion of the course, student will be able to

1. Use basic features of java script.
2. Use relevant data types for developing application in java script.
3. Use the function and objects as self-contained, with data passing in and out through well-defined interfaces in development of small systems.
4. Apply the regular expression for Text matching and manipulation.
5. Explore use of the various aspects of JavaScript object models that are fundamental to the proper use of the language.
6. Develop the application using windows controlling and form handling.

Sinhgad Technical Education Society's

RMD Sinhgad Technical institute Campus

RMD Sinhagd School of Engineering, Warje, Pune -58

Department of Electronics & Telecommunication Engineering

INDEX

Sr. No.	Title of the Experiment	Performance Date	Submission Date	Marks	Sign
1.	Write a JavaScript program to calculate area of triangle, area of rectangle and area of circle				
2.	Write a JavaScript program to generate the multiplication table of a given number.				
3.	Write a JavaScript program to following operations on a given string, • Reverse string • Replace characters of a string. • String is Palindrome.				
4.	Write a JavaScript program to compare two strings using various methods.				
5.	Write a JavaScript program that will create a countdown timer.				
6.	Write a JavaScript program that will create an array and perform following operations • To remove specific element from the array. • Check if an array contains a specified value. • To empty an array				
7.	Write a JavaScript program to illustrate different Set operations like- • Union • Intersection • Difference • Set Difference				
8.	Write a JavaScript program to create a Home page of any website and change background color using • On mouse over event • On focus event				
9.	Design and implement a simple calculator using Java script for operations like addition multiplication, subtraction, division, square of a number				

Ms. Varsha Nanavare
Subject Teacher

Mrs. Varsha Kshirsagar
HOD E&TC Engg.

Experiment No._____ *Date___/___/2020*

TITLE OF EXPERIMENT: - A Program to generate the multiplication table of a given number.

DIVISION:_____ **BRANCH:** _____

BATCH:_____ **ROLL NO.:** _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 1

Aim: Write a JavaScript program to generate the multiplication table of a given number.

Prerequisites :

- For this **Javascript Lab**, it is assumed that you have a prior knowledge of HTML coding. It would help if you had some prior exposure to object-oriented programming concepts and a general idea on creating online applications.
- To understand this experiment, you should have the knowledge of the basic **JavaScript, for loop, arithmetic operators, relational operators.**

Editor:

1.	NotePad
2.	Visual studio code

Theory:

Definition and Usage:

The prompt() method displays a dialog box that prompts the user for input.

The prompt() method returns the input value if the user clicks "OK", otherwise it returns null.

The **prompt()** method is used to display a dialog box with an optional message prompting the user to input some text. It is often used if the user wants to input a value before entering a page. It returns a string containing the text entered by the user, or null.

Syntax:

prompt(message, default)

- **message** is a string of text to display to the user. It can be omitted if there is nothing to show in the prompt window i.e. it is optional.
- **default** is a string containing the default value displayed in the text input field. It is also optional.

Example:

Prompt for a user name and output a message:

```
let person = prompt("Please enter your name", "Harry Potter");

if (person != null) {
  document.getElementById("demo").innerHTML =
  "Hello " + person + "! How are you today?";
}
```

parseInt() function

The **parseInt()** function is used to accept the string ,radix parameter and convert it into an integer. The radix parameter is used to specify which numeral system to be used, for example, a radix of 16 (hexadecimal) indicates that the number in the string should be parsed from a hexadecimal number to a decimal number. If the string does not contain a numeric value then it returns NaN i.e, not a number.

Syntax:

parseInt(Value, radix)

Parameters: This function accepts two parameters as mentioned above and described below:

- **Value:** This parameter contains a string which is converted to an integer.
- **radix:** This parameter represents the radix or base to be used and it is optional.
- **Return value:** It returns a number and if the first character can't be converted to a number then the function returns NaN. It actually returns a number parsed up to that point where it encounters a character that is not a number in the specified radix(base).

- **Example:**

```
<script>

   var v1 = parseInt("3.14");

     document.write('Using parseInt("3.14") = '

   + v1 + "<br>"); </script>
```

- **Output:**
  Using parseInt("3.14") = 3

Example 1: Multiplication Table Up to 10

```
// program to generate a multiplication table

// take input from the user
const number = parseInt(prompt('Enter an integer: '));

//creating a multiplication table
for(let i = 1; i <= 10; i++) {

    // multiply i with number
    const result = i * number;

    // display the result
    console.log(`${number} * ${i} = ${result}`);
}
```

Output

```
Enter an integer: 3
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
```

In the above program, the user is prompted to enter an integer value. Then, the for loop is used to iterate through **1** to **10** to create a multiplication table.

Example 2: Multiplication Table Up to a Range

```
/* program to generate a multiplication table
upto a range */

// take number input from the user
const number = parseInt(prompt('Enter an integer: '));

// take range input from the user
const range = parseInt(prompt('Enter a range: '));
```

```
//creating a multiplication table
for(let i = 1; i <= range; i++) {
   const result = i * number;
   console.log(`${number} * ${i} = ${result}`);
}
```

Output

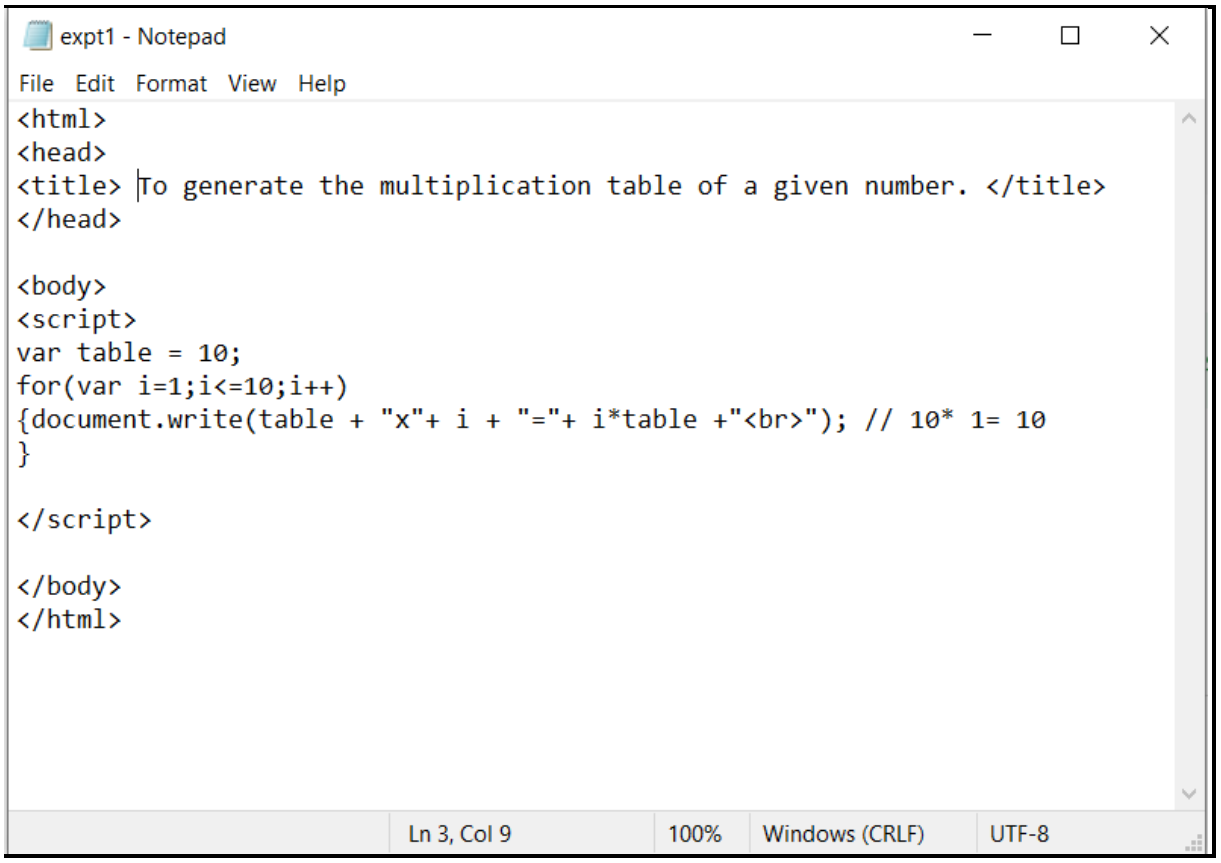
```
Enter an integer: 7
7* 1 = 7
7* 2= 14
7*3 = 21
7 * 4 = 28
7 * 5 = 35
```

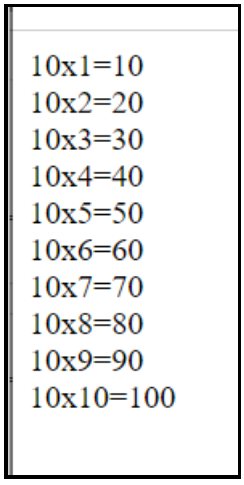
In the above example, the user is prompted to enter an integer and also a range for which they want to create a multiplication table.

Program:

```
<html>
<head>
<title> To generate the multiplication table of a given number. </title>
</head>
<body>
<script>
var table = 10;
for(var i=1;i<=10;i++)
{document.write(table + "x"+ i + "="+ i*table +"<br>"); // 10* 1= 10
}
</script>
</body>
</html>
```

Screenshot's of Output:





Conclusion:

*Experiment No.*_____ *Date*___/___/*22*

TITLE OF EXPERIMENT: - A Program to calculate area of triangle, area of rectangle and area of circle.

DIVISION:_____ **BRANCH:** _____

BATCH:_____ **ROLL NO.:** _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 2

Aim: Write a JavaScript program to calculate area of triangle, area of rectangle and area of circle.

Prerequisites:

- For this **Javascript Lab**, it is assumed that you have a prior knowledge of HTML coding. It would help if you had some prior exposure to object-oriented programming concepts and a general idea on creating online applications.
- To understand this experiment, you should have the knowledge of the basic **JavaScript, Javascript operators, javascript Math sqt().**

Editor:

1.	NotePad
2.	Visual studio code

Theory:

A) Area of triangle :

The **area of a triangle** is defined as the total space occupied by the three sides of a triangle in a 2-dimensional plane.
The area of a triangle can be calculated using various methods. In this expt., we will be calculating the area using two different approaches.

These two methods are as follows:-

1. Using **Base** and **Height** of the Triangle
2. Using the **Sides** of the Triangle

Case 1: If you know the base and height of a triangle, you can find the area using the formula:

```
area = (base * height) / 2
```

Example 1: <u>Area When Base and Height is Known</u>

```
//JavaScript Program To Calculate The Area of a Triangle

var base = parseInt(prompt("Enter the base: "));

var height = parseInt(prompt("Enter the height: "));

//Calculating the area

var area = (base * height) / 2;

//Display Output

console.log("Base: " + base);

console.log("Height: " + height);

console.log("The area of the triangle is " + area);
```

Output

```
Base: 3
Height: 4
The area of the triangle is 6
```

How Does This Program Work?

```
var base = parseInt(prompt("Enter the base: "));
var height = parseInt(prompt("Enter the height: "));
```

1) The user is asked to enter the value of **base** and **height** of the triangle.

```
//Calculating the area
var area = (base * height) / 2;
```

2) Then, we calculate the area of the triangle using **½ x Base x Height**.

```
//Display Output
console.log("Base: " + base);
console.log("Height: " + height);
console.log("The area of the triangle is " + area);
```

3) Finally, the area of the triangle is printed on the screen using **console.log()** function.

OR

```
const baseValue = prompt('Enter the base of a triangle: ');
const heightValue = prompt('Enter the height of a triangle: ');

// calculate the area
const areaValue = (baseValue * heightValue) / 2;

console.log(
  `The area of the triangle is ${areaValue}`
);
```

Output

```
Enter the base of a triangle: 4
Enter the height of a triangle: 6
The area of the triangle is 12
```

Case 2: If you know all the sides of a triangle, you can find the area using Herons' formula. If a, b and c are the three sides of a triangle, then

$$s = (a+b+c)/2$$

$$area = \sqrt{(s(s-a)*(s-b)*(s-c))}$$

Example 2: <u>Area When All Sides are known</u>

```
// JavaScript program to find the area of a triangle


const side1 = parseInt(prompt('Enter side1: '));
const side2 = parseInt(prompt('Enter side2: '));
const side3 = parseInt(prompt('Enter side3: '));

// calculate the semi-perimeter
const s = (side1 + side2 + side3) / 2;

//calculate the area
const areaValue = Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));

console.log(`The area of the triangle is ${areaValue}`);
```

1) We calculate the semi-perimeter of the triangle using **s = (a + b + c) / 2**.

```
var area = Math.sqrt(s * (s - first_side) * (s - second_side) * (s - third_side));
```

2) Then, we calculate the area of the triangle using the formula:

Area = [s((s-a)(s-b)(s-c))]½**

3) The **Math.sqrt()** method is used to calculate the square root of the number.

```
console.log("Area: " + area);
```

4) Finally, the area of the triangle is printed on the screen using the **console.log()** method.

Output

```
Enter side1: 3
Enter side2: 4
Enter side3: 5
The area of the triangle is 6
```

B) To Calculate Area Of Rectangle :

The process to calculate area of a rectangle is very easy when you know the formula for the area of a rectangle. The space within the perimeter of the rectangle is an area of the rectangle.

Formula:

$a = l * w$

where:

a = area of rectangle

l = length of the rectangle

w = width of the rectangle

- By multiplying the length and width area of a rectangle is calculated.
- Functions need to be called before they can execute. That is because, area is a function in JavaScript code that you hold, so you need to *call it*, and you also need to pass the parameters to it.

```
<script type="text/javascript">
```

```
/*
   Javascript program to find the area of a rectangle
*/

   var l, w, a;
   l = 8;
   w = 6;

   /* Calculate area of rectangle */
   a = l * w;

   document.write("Area of rectangle = " + a + " units");

</script>
```

Output Screen:

```
Area of rectangle = 48 units
```

OR

```
<script>

// Javascript program to find area of rectangle


// Utility function
function areaRectangle(width, length)
{
        let area = width * length;
        return area;
}


// Driver program

let width = 5;
let length = 6;
document.write("Area = " + areaRectangle(width, length) + "<br>");

</script>
```

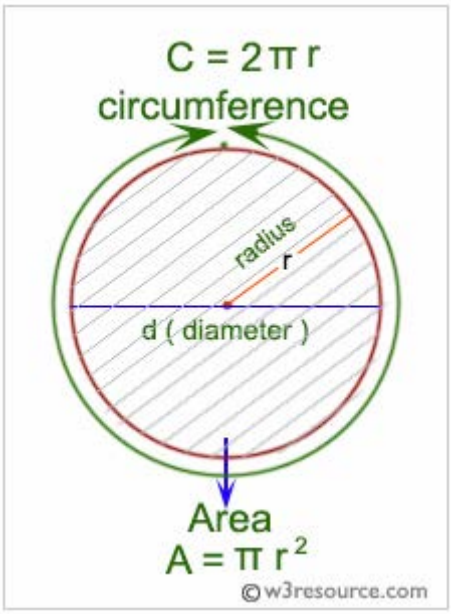
Output

```
Area = 30
```

C) To Calculate Area Of Circle:

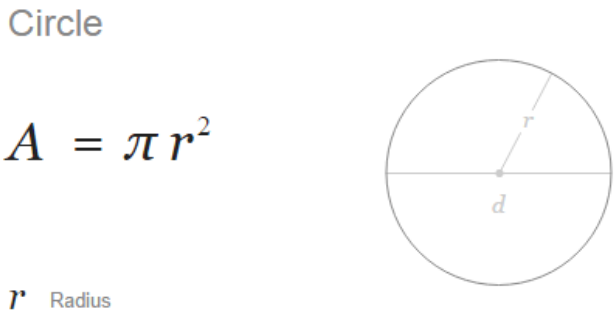
In geometry, the area enclosed by a circle of radius r is πr2. Here the Greek letter π represents a constant, approximately equal to 3.14159, which is equal to the ratio of the circumference of any circle to its diameter.

The circumference of a circle is the linear distance around its edge.



We use the following mathematical formula for calculating area of the circle In JavaScript:

Area of Circle= radius * radius * PI
circumference of the circle = 2 * radius * PI



Where, *radius* is the radius of the circle, &

PI is the constant value of the pi.

Approach: First we will create three input fields using <input type=”number”> tag to holds number input. After filling the input value, when user click the button, then the JavaScript function Area() will be called.

In JavaScript function, we use **document.getElementById("side1").value** to get the input value and then apply **parseInt()** method on it to get the input value in number. And then use simple mathematical formula to find the area of circlr and use **document.getElementById("display").innerHTML** to display the output on the screen.

```
<html>
<head>
<title> JavaScript Program To Calculate The Area of a Circle </title>
</head>
<body>
   <h1>Area of a circle</h1>
   Enter the radius for your circle:
   <input type="text" id="txtRadius" />
   <input type="button" value="Calculate" onClick=CalculateArea() />
   <script>
      function CalculateArea() {
         var radius = document.getElementById('txtRadius').value;
         alert("The area of the circle is " + (radius * radius * Math.PI));
      }
   </script>
</body>
</html>
```

Program:

A) To Calculate Area Of Triangle

```
<html>
<head>
```

```
<title> JavaScript Program To Calculate The Area of a Triangle </title>
</head>
<body>
<script>
// enter the value of base and height of the triangle
const baseValue = prompt('Enter the base of a triangle: ');
const heightValue = prompt('Enter the height of a triangle: ');
// calculate the area
const areaValue = (baseValue * heightValue) / 2;
//Display Output
console.log(`The area of the triangle is ${areaValue}`);
</script>
</body>
</html>
```

B) To Calculate Area Of Rectangle

```
<html>
<head>
<title> JavaScript Program To Calculate The Area of a Rectanble </title>
</head>
<body>
<script type="text/javascript">
/*
   Javascript program to find the area of a rectangle
*/
   var l, w, a;
   l = 8;
   w = 6;
     /* Calculate area of rectangle */
   a = l * w;
```

```
    document.write("Area of rectangle = " + a + " units");
</script>
</body>
</html>
```

c) To Calculate Area Of Circle:

```
<html>
<head>
<title> JavaScript Program To Calculate The Area of a Circle </title>
</head>
<body>
   <h1>Area of a circle</h1>
   Enter the radius for your circle:
   <input type="text" id="txtRadius" />
   <input type="button" value="Calculate" onClick=CalculateArea() />
   <script>
      function CalculateArea() {
         var radius = document.getElementById('txtRadius').value;
         alert("The area of the circle is " + (radius * radius * Math.PI));
      }
   </script>
</body>
</html>
```

Screenshot's of Output:

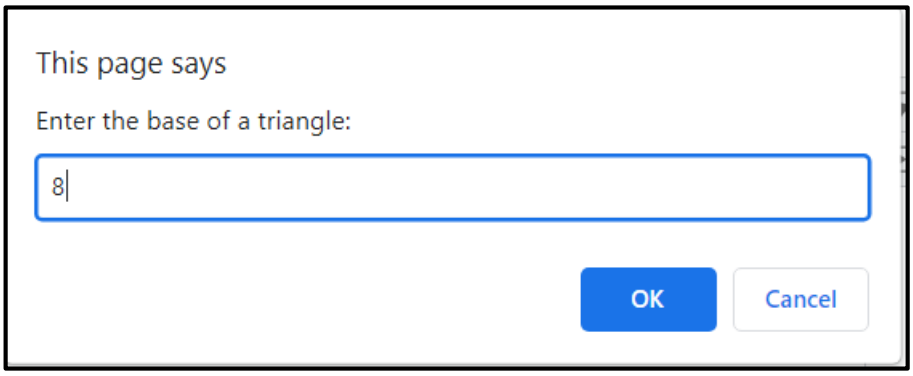
A) To Calculate Area Of Triangle

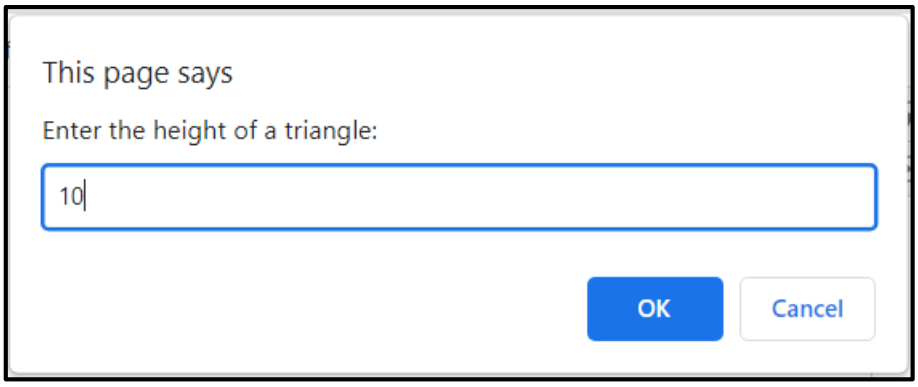
expt2 - Notepad

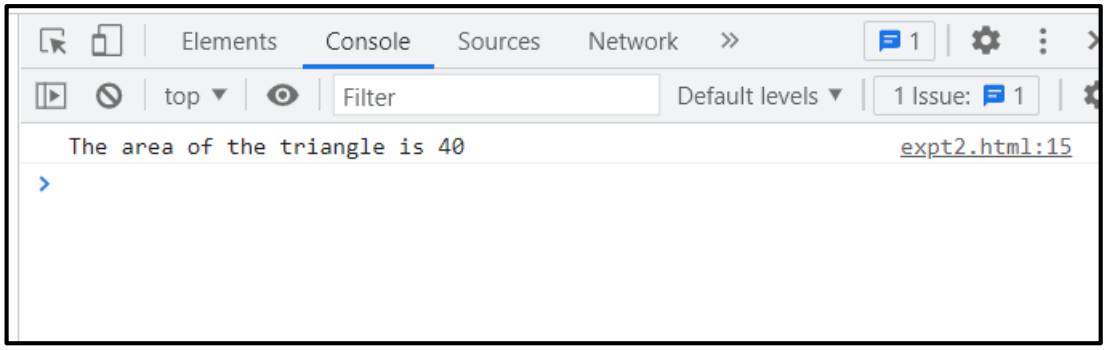
File Edit Format View Help

```
<html>
<head>
<title> JavaScript Program To Calculate The Area of a Triangle </title>
</head>

<body>
<script>

const baseValue = prompt('Enter the base of a triangle: ');
const heightValue = prompt('Enter the height of a triangle: ');

// calculate the area
const areaValue = (baseValue * heightValue) / 2;

console.log(
  `The area of the triangle is ${areaValue}`
);


</script>
```

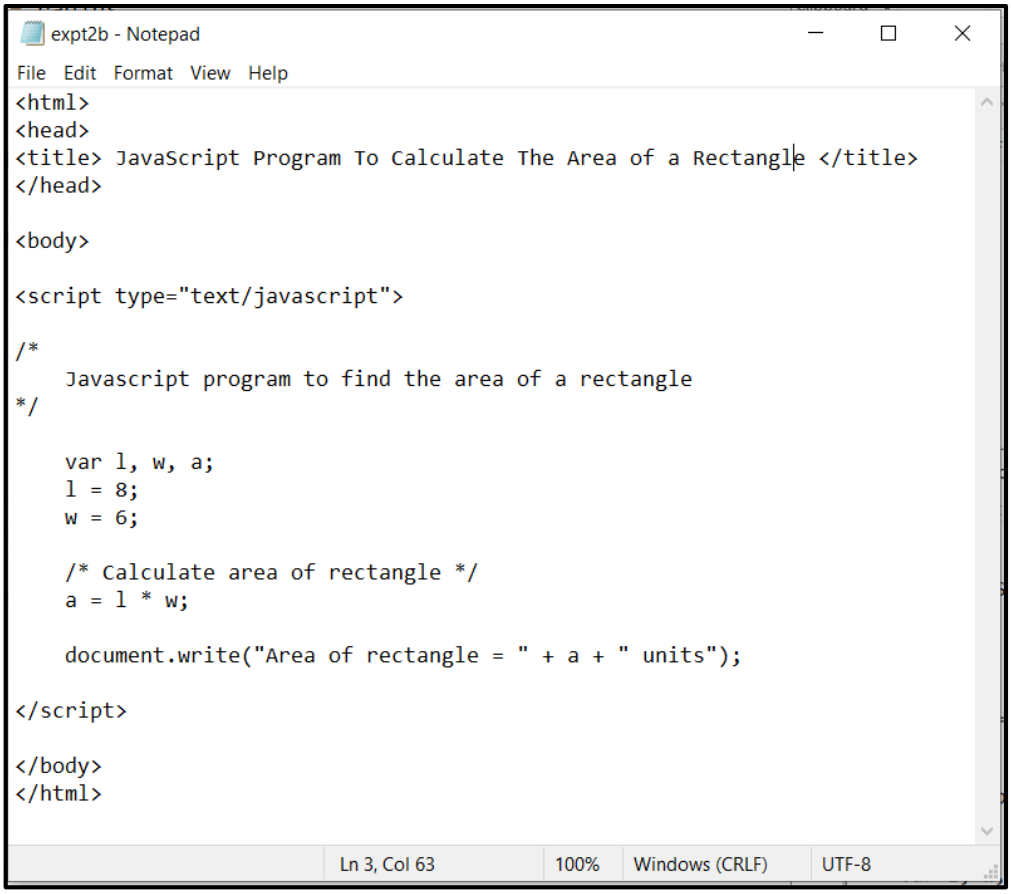
Ln 18, Col 1 | 100% | Windows (CRLF) | UTF-8

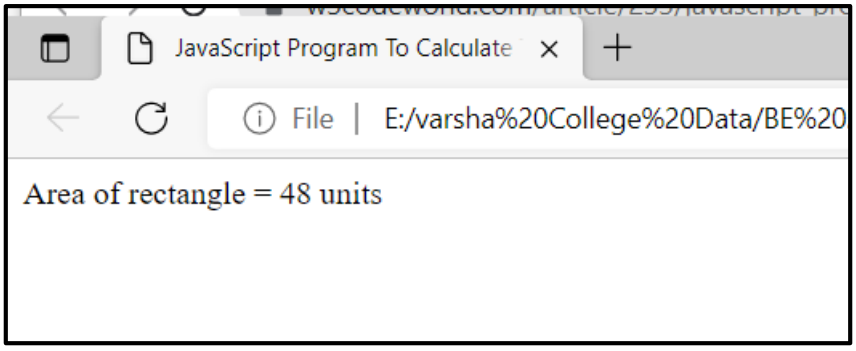






B) To Calculate Area Of Rectangle



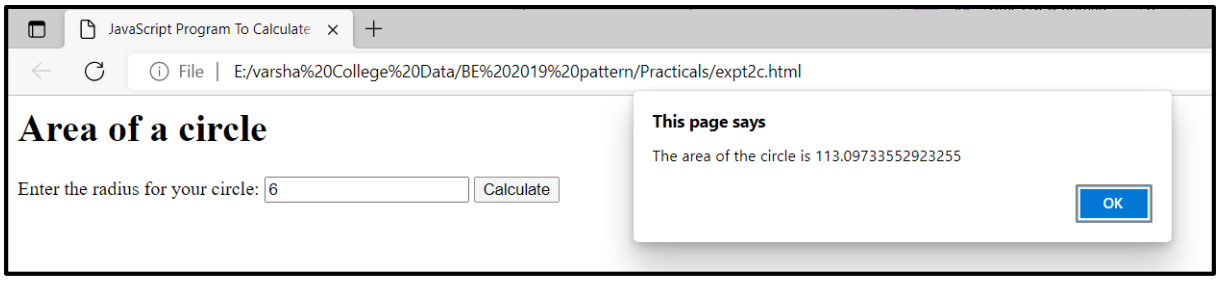


c) To Calculate Area Of Circle:

expt2c - Notepad

File Edit Format View Help

```
<html>
<head>
<title> JavaScript Program To Calculate The Area of a Circle </title>
</head>

<body>
    <h1>Area of a circle</h1>
    Enter the radius for your circle:
    <input type="text" id="txtRadius" />
    <input type="button" value="Calculate" onClick=CalculateArea() />
    <script>
        function CalculateArea() {
            var radius = document.getElementById('txtRadius').value;
            alert("The area of the circle is " + (radius * radius *
Math.PI));
        }
    </script>
</body>

</html>
```



Conclusion:

Experiment No._____ *Date___/___/2020*

TITLE OF EXPERIMENT: - A Program to Reverse string, Replace characters of a string, String is Palindrome operations on a given string.

DIVISION:_____ BRANCH: _____

BATCH:_____ ROLL NO.: _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 3

Aim: Write a JavaScript program to following operations on a given string,

- Reverse string
- Replace characters of a string.
- String is Palindrome.

Prerequisites:

- For this **Javascript Lab**, it is assumed that you have a prior knowledge of HTML coding. It would help if you had some prior exposure to object-oriented programming concepts and a general idea on creating online applications.
- To understand this experiment, you should have the knowledge of the basic JavaScript,

✓ JavaScript String,
✓ JavaScript Function
✓ Function Expressions
✓ JavaScript String replace()

Editor:

1.	NotePad
2.	Visual studio code

Theory:

Strings:

Strings are used for storing text. A String variable contains a collection of characters surrounded by double quotes:

Example

Create a variable of type String and assign it a value:

```
String greeting = "Hello";
```

String Length

A String in Java is actually an object, which contain methods that can perform certain operations on strings. For example, the length of a string can be found with the length() method:

A. Reverse string :

Example 1: Reverse a String Using for Loop

```
// program to reverse a string

function reverseString(str) {

   // empty string
   let newString = "";
   for (let i = str.length - 1; i >= 0; i--)
{
      newString += str[i];
   }
   return newString;
}

// take input from the user
const string = prompt('Enter a string: ');

const result = reverseString(string);
console.log(result);
```

Output

```
Enter a string: hello world
dlrow olleh
```

In the above program, the user is prompted to enter a string. That string is passed to the reverseString() function.

Inside the reverseString() function,

- An empty newString variable is created.
- The for loop is used to iterate over the strings. During the first iteration, str.length - 1 gives the position of the last element. That element is added to the newString variable.

  This process continues for all the string elements.
- The value of i decreases in each iteration and continues until it becomes **0**.

Example 2: Reverse a String Using built-in Methods

For this, we will use three methods: the String.prototype.split() method, the Array.prototype.reverse() method and the Array.prototype.join() method.

- The split() method splits a String object into an array of string by separating the string into sub strings.
- The reverse() method reverses an array in place. The first array element becomes the last and the last becomes the first.
- The join() method joins all elements of an array into a string.

Steps:

Step 1. Use the split() method to return a new array
Step 2. Use the reverse() method to reverse the new created array
Step 3. Use the join() method to join all elements of the array into a string
Step 4. Return the reversed string

```
// program to reverse a string

function reverseString(str) {

    // Step 1. Use the split() method to return a new array of strings
    const arrayStrings = str.split(""); // var splitString = "hello".split("");
    // ["h", "e", "l", "l", "o"]


    // Step 2. Use the reverse() method to reverse the new created array elements
    const reverseArray = arrayStrings.reverse(); // var reverseArray = ["h", "e", "l", "l",
"o"].reverse();
    // ["o", "l", "l", "e", "h"]

    // Step 3. Use the join() method to join all elements of the array into a string
    const joinArray = reverseArray.join("");  // var joinArray = ["o", "l", "l", "e", "h"].join("");
    // "olleh"


    // Step 4.return the reversed string
    return joinArray;  // "olleh"
```

```
}

// take input from the user
const string = prompt('Enter a string: ');

const result = reverseString(string);
console.log(result);
```

Output

```
Enter a string: hello
olleh
```

In the above program, the built-in methods are used to reverse a string.

- First, the string is split into individual array elements using the split() method. str.split("") gives ["h", "e", "l", "l", "o"].
- The string elements are reversed using the reverse() method. arrayStrings.reverse() gives ["o", "l", "l", "e", "h"].
- The reversed string elements are joined into a single string using the join() method. reverseArray.join("") gives olleh.

B. Replace characters of a string.

Example: Replace First Occurrence of a Character in a String

```
// program to replace a character of a string

const string = 'Mr Red has a red house and a red car';

// replace the characters
const newText = string.replace('red', 'blue');

// display the result
console.log(newText);
```

Output

```
Mr Red has a blue house and a red car
```

In the above program, the replace() method is used to replace the specified string with another string.

When a string is passed in the replace() method, it replaces only the first instance of the string. So if there is a second match in the string, it won't be replaced.

You could also pass a **regular expression (regex)** inside the replace() method to replace the string.

Example 2: Replace Character of a String Using RegEx

```
// program to replace a character of a string

const string = 'Mr Red has a red house and a red car';

// regex expression
const regex = /red/g;

// replace the characters
const newText = string.replace(regex, 'blue');

// display the result
console.log(newText);
```

Output

```
Mr Red has a blue house and a blue car
```

In the above program, a **regex** expression is used as the first parameter inside the replace() method.

/g refers to **global**. It means that all the matching characters in the string are replaced.

Since JavaScript is case-sensitive, **R** and **r** are treated as different values.

You could also use the regex to perform case-insensitive replacement using **/gi,** where i represents case-insensitive.

C. String is Palindrome

A string is a palindrome if it is read the same from forward or backward.

For example, **dad** reads the same either from forward or backward. So the word **dad** is a palindrome. Similarly, **madam** is also a palindrome.

Example 1: Check Palindrome Using for Loop

```
// program to check if the string is palindrome or not

function checkPalindrome(string) {

    // find the length of a string
    const len = string.length;

    // loop through half of the string
    for (let i = 0; i < len / 2; i++) {

        // check if first and last string are same
        if (string[i] !== string[len - 1 - i]) {
            return 'It is not a palindrome';
        }
    }
    return 'It is a palindrome';
}

// take input
const string = prompt('Enter a string: ');

// call the function
const value = checkPalindrome(string);

console.log(value);
```

Output

```
Enter a string: madam
It is a palindrome
```

In the above program, the **checkPalindrome()** function takes input from the user.

- The length of the string is calculated using the length property.
- The for loop is used to iterate up to the half of the string. The if condition is used to check if the first and the corresponding last characters are the same. This loop continues till the half of the string.
- During the iteration, if any character of the string, when compared with its corresponding last string is not equal, the string is not considered a palindrome.

Example 2: Check Palindrome using built-in Functions

```
// program to check if the string is palindrome or not

function checkPalindrome(string) {

    // convert string to an array
    const arrayValues = string.split('');

    // reverse the array values
    const reverseArrayValues = arrayValues.reverse();

    // convert array to string
    const reverseString = reverseArrayValues.join('');

    if(string == reverseString) {
        console.log('It is a palindrome');
    }
    else {
        console.log('It is not a palindrome');
    }
}

//take input
const string = prompt('Enter a string: ');

checkPalindrome(string);
```

Output

```
Enter a string: hello
It is not a palindrome
```

In the above program, the palindrome is checked using the built-in methods available in JavaScript.

- The split('') method converts the string into individual array characters.

```
const arrayValues = string.split(''); // ["h", "e", "l", "l", "o"]
```

- The reverse() method reverses the position in an array.

```
// ["o", "l", "l", "e", "h"]
const reverseArrayValues = arrayValues.reverse();
```

- The join('') method joins all the elements of an array into a string.

```
const reverseString = reverseArrayValues.join(''); // "olleh"
```

- Then the if...else statement is used to check if the string and the reversed string are equal. If they are equal, the string is a palindrome.

  Note: The multiple lines of code can be reduced and written in one line:

```
const reverseString = string.split('').reverse().join('');
```

Program:

A. Reverse string :

```
<html>
<head>
<title> program to reverse a string </title>
</head>
<body>
<script type="text/javascript">
function reverseString(str) {

   // return a new array of strings
   const arrayStrings = str.split("");
```

```
    // reverse the new created array elements
    const reverseArray = arrayStrings.reverse();

    // join all elements of the array into a string
    const joinArray = reverseArray.join("");
        // return the reversed string
    return joinArray;
}
 // take input from the user
const string = prompt('Enter a string: ');

const result = reverseString(string);
console.log(result);
 </script>
</body>
</html>
```

B. Replace characters of a string.

```
<html>
<head>
<title> program to reverse a string </title>
</head>
<body>
<script type="text/javascript">
// program to replace a character of a string

const string = 'Mr Red has a red house and a red car';

// regex expression
const regex = /red/g;
```

```
// replace the characters
const newText = string.replace(regex, 'blue');

// display the result
console.log(newText);
</script>
</body>
</html>
```

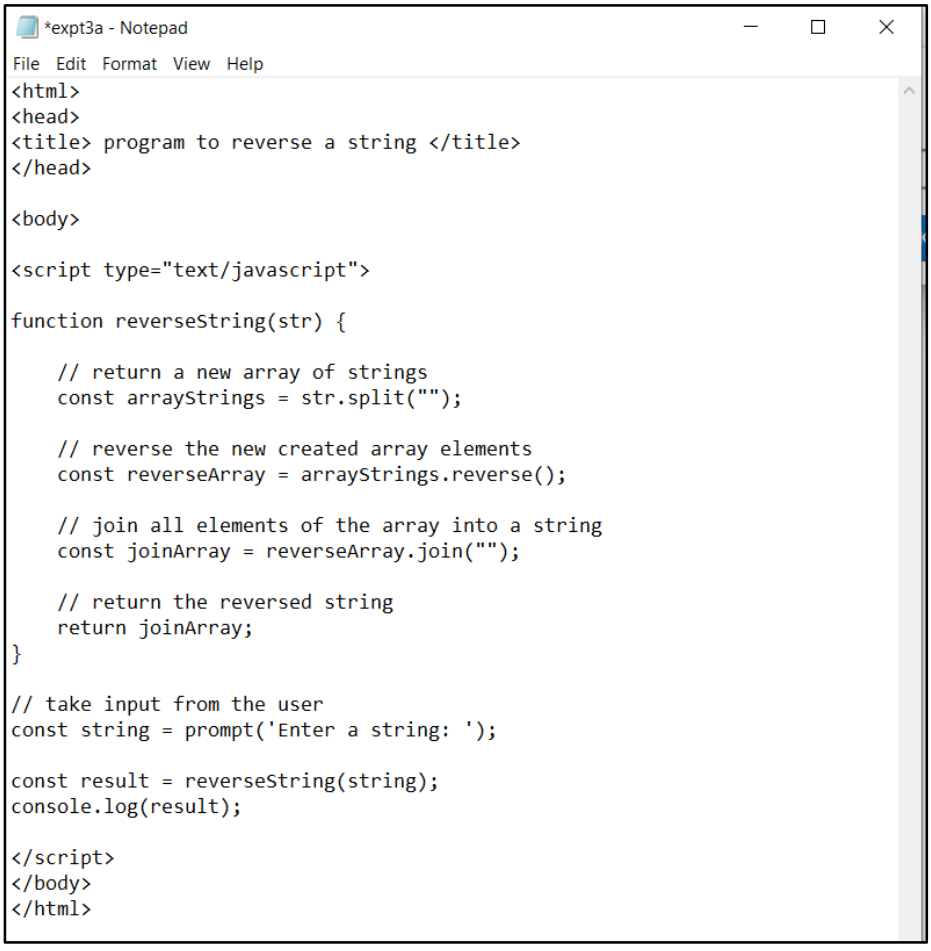
C. String is Palindrome

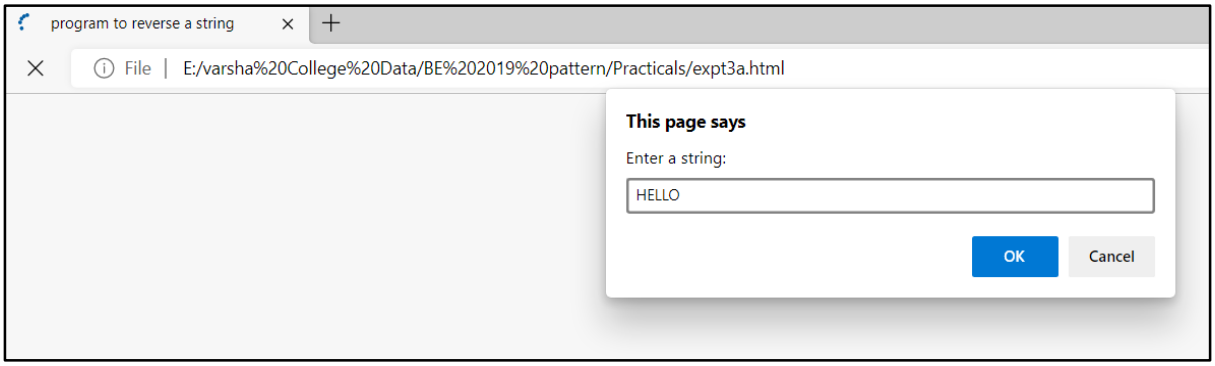
```
</head>
<body>
<script type="text/javascript">
// program to check if the string is palindrome or not
function checkPalindrome(string) {

    // convert string to an array
    const arrayValues = string.split('');

    // reverse the array values
    const reverseArrayValues = arrayValues.reverse();

    // convert array to string
    const reverseString = reverseArrayValues.join('');

    if(string == reverseString) {
        console.log('It is a palindrome');
    }
    else {
        console.log('It is not a palindrome');
```

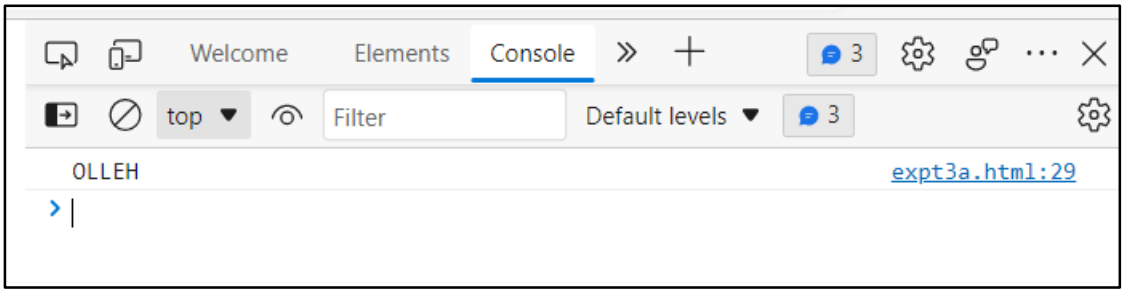
```
   }
}


//take input
const string = prompt('Enter a string: ');
checkPalindrome(string);
</script>
</body>
</html>
```

Screenshot's of Output:

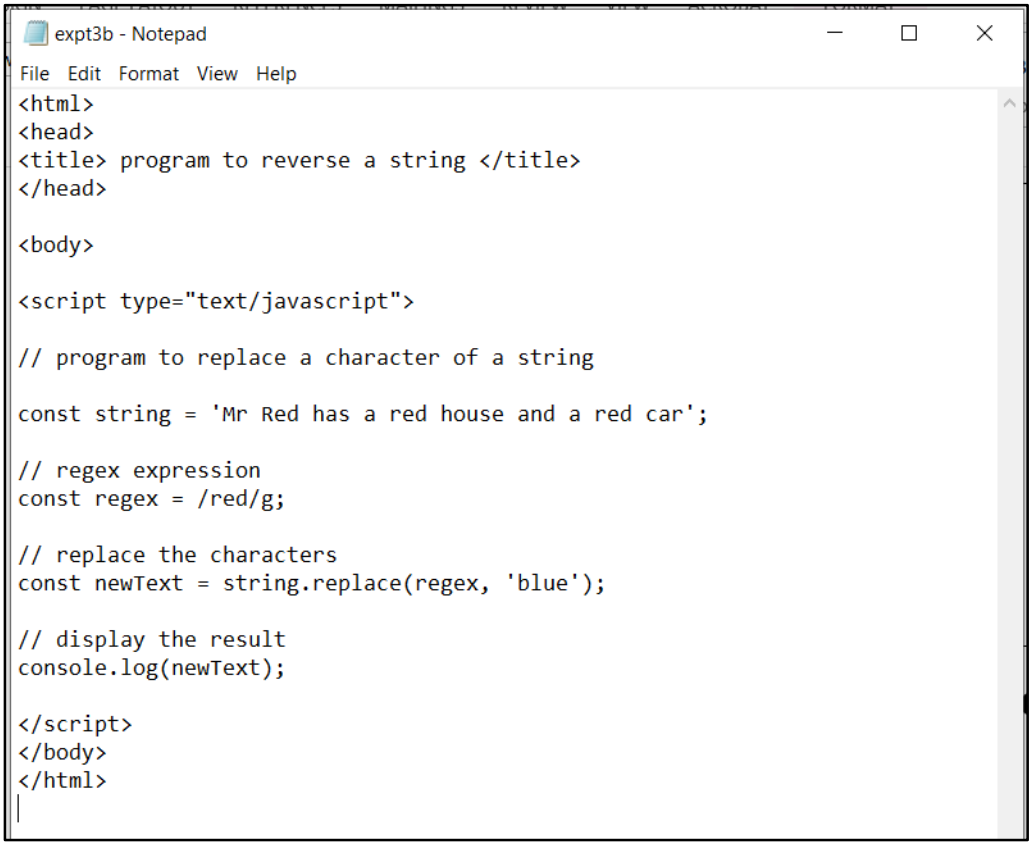
a. Reverse string :

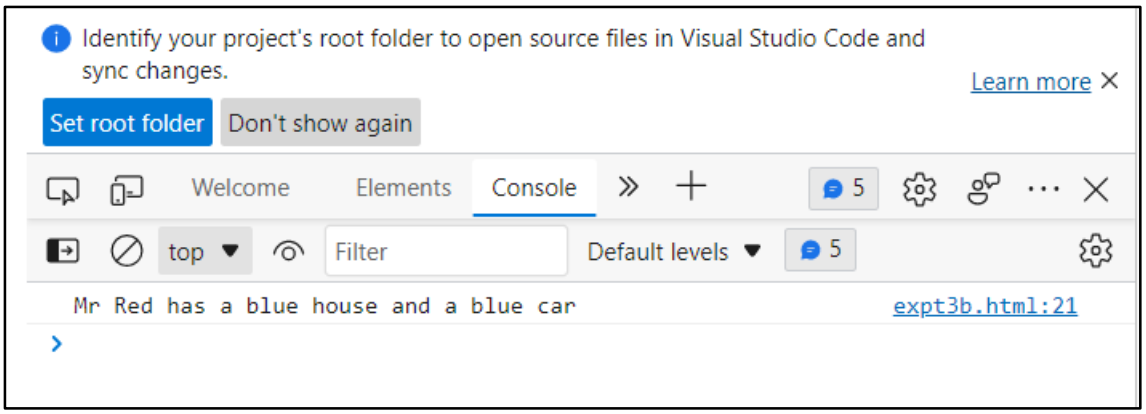






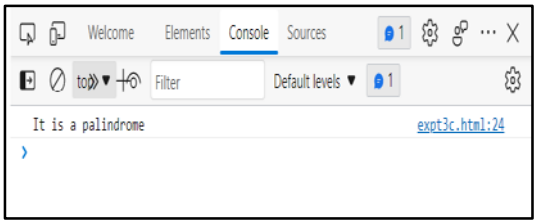
b. Replace characters of a string.





c. String is Palindrome

```
File  Edit  Format  View  Help
</head>

<body>

<script type="text/javascript">

// program to check if the string is palindrome or not

function checkPalindrome(string) {

    // convert string to an array
    const arrayValues = string.split('');

    // reverse the array values
    const reverseArrayValues = arrayValues.reverse();

    // convert array to string
    const reverseString = reverseArrayValues.join('');

    if(string == reverseString) {
        console.log('It is a palindrome');
    }
    else {
        console.log('It is not a palindrome');
    }
}

//take input
const string = prompt('Enter a string: ');

checkPalindrome(string);

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



Conclusion:

Experiment No._____ *Date___/___/2020*

TITLE OF EXPERIMENT: - A Program to compare two strings using various methods.

DIVISION:_____ **BRANCH:** _____

BATCH:_____ **ROLL NO.:** _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 4

Aim: Write a JavaScript program to compare two strings using various methods.

Prerequisites:

- For this **Javascript Lab**, it is assumed that you have a prior knowledge of HTML coding. It would help if you had some prior exposure to object-oriented programming concepts and a general idea on creating online applications.
- To understand this experiment, you should have the knowledge of the basic JavaScript, **JavaScript String, Javascript String toUpperCase(), JavaScript Regex, Javascript String localeCompare()**

Editor:

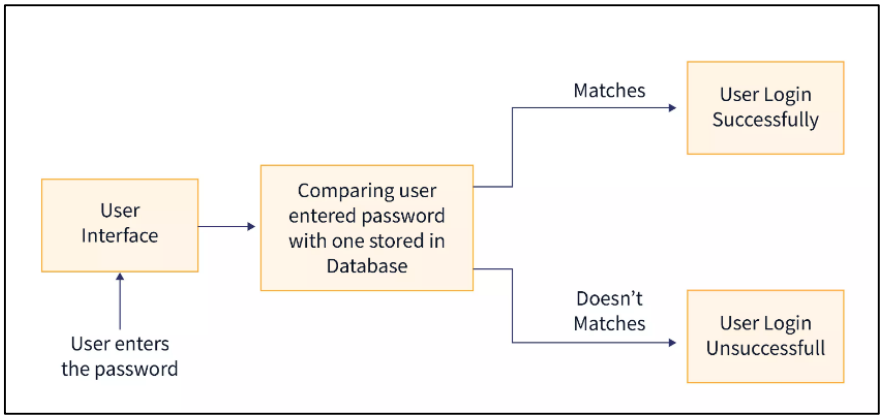
1.	NotePad
2.	Visual studio code

Theory:

Introduction:

Two strings in Javascript can be compared to check whether they are the same or not using different methods like toUpperCase(), localeCompare(), etc. We can also compare two strings in javascript using RegEx. Operators like greater than, or less than or equality operators to compare two strings.

Eg.: Imagine we are implementing login authentication for a website. In this case, we are supposed to take the username input from the user to check if the username entered is the same as the original username of the user that is stored in our database. Now the simplest way to do it is to compare the entered username with the one stored in our database; if they match, then let the user login otherwise, do not let the user login.



Now, as we know that the inputs are usually strings. Thus, we need to compare two strings in order to perform tasks like the above. Now the question is 'How do we compare two strings in Javascript?'

The following methods are used to compare two strings in javascript:

1. Using equality operators
2. Using toUpperCase()
3. Using RegEx
4. Using localeCompare()

1. Using equality operators (Case-insensitive String Comparison)

Imagine we are taking name inputs from the users and checking if that name is present in our database. In such cases, the two strings need not be strictly equal, i.e. the strings can have a different case. Such situations are known as **case-insensitive** string comparison situations. The **Equality operator** (==) is used to check if both the strings are the same. When comparing strings, case insensitivity refers to not taking into account uppercase and lowercase letters. The method **toLowerCase()** and **toUpperCase()** in javascript is used to determine case insensitive comparison of two strings in javascript.

Definition:- The javascript equality operator, which is used to compare two values on both sides and it will return the result as true or false.

Syntax

```
string1 == string2
```

Here, string1 and string2 are two strings that are being compared.

Return value

- **boolean**

The expression returns true if *string1* and *string2* are equal otherwise it returns false.

Example:

```
var stringFirst = "javascript world";
var stringSecond = "javascript world";
var res = '';

if(stringFirst == stringSecond)
{
  res = 'strings are equal';
}else
{
  res = 'strings are not equal';
}

document.write( "Output :- " + res );
```

```
Output:- strings are equal
```

Remember some points of JavaScript Equality Operator:

- If this method returns true, strings are equal.
- If this method returns false, strings are not equal

2. Using toUpperCase()

As discussed earlier, The toUpperCase() method is used to return the calling string value converted into the uppercase.

Example: Using toUpperCase()

```
// js program to perform string comparison

const string1 = 'JavaScript Program';
const string2 = 'javascript program';

// compare both strings
const result = string1.toUpperCase() === string2.toUpperCase();

if(result) {
   console.log('The strings are similar.');
} else {
   console.log('The strings are not similar.');
}
```

Output

```
The strings are similar.
```

In the above program, two strings are compared. Here,

- The toUpperCase() method converts all the string characters to uppercase.
- === is used to check if both the strings are the same.
- The if...else statement is used to display the result as per the condition.

Note: You can also use the toLowerCase() method to convert all the strings to lowercase and perform the comparison.

3. JS String Comparison Using RegEx

Example: JS String Comparison Using RegEx

```
// program to perform string comparison

const string1 = 'JavaScript Program';
const string2 = 'javascript program';

// create regex
const pattern = new RegExp(string1, "gi");

// compare the stings
const result = pattern.test(string2)

if(result) {
   console.log('The strings are similar.');
} else {
   console.log('The strings are not similar.');
}
```

Output

```
The strings are similar.
```

In the above program, the RegEx is used with the test() method to perform case insensitive string comparison. In the RegEx pattern, "g" syntax denotes **global** and "gi" syntax denotes **case insensitive** comparisons.

4. localeCompare() Method

You can use the localeCompare() Method of javascript to compare two strings in javascript.

Definition:-The javascript localeCompare() method is used to on a string object for comparing two strings. This method returns a number that tells whether the string comes before, after, or is equal to the compareString in sort order.

Syntax:-

string.localeCompare(compareString);

Note:- this method will return 0, -1 or 1. This method does case-sensitive comparing.

Remember some points of localeCompare() method

- -1 if the string is sorted before the *compareString*
- 0 if the two strings are equal
- 1 if the string is sorted after the *compareString*, two strings are not equal

```
var stringFirst = "javascript world";
var stringSecond = "javascript world";

var res = stringFirst.localeCompare(stringSecond);
document.write( "Output :- " + res + "<br>");

var stringThird = "javascript world";
var stringFourth = "javascript";

var res1 = stringThird.localeCompare(stringFourth);
document.write( "Output :- " + res1 );
```

Result of the above code is:

```
Output :- 0
Output :- 1
```

Example: Using localeCompare()

```
// program to perform case insensitive string comparison
```

```
const string1 = 'JavaScript Program';
const string2 = 'javascript program';

const result = string1.localeCompare(string2, undefined, { sensitivity: 'base' });

if(result == 0) {
   console.log('The strings are similar.');
} else {
   console.log('The strings are not similar.');
}
```

Output

```
The strings are similar.
```

In the above program, the localeCompare() method is used to perform case insensitive string comparison. The localeCompare() method returns a number that indicates whether a reference string comes before, or after, or is the same as the given string. Here, { sensitivity: 'base' } treats **A** and **a** as the same.

Program:

a. Using equality operators

```
<html>
<head>
<title> program to perform string comparison </title>
</head>
<body>
<script type="text/javascript">
var stringFirst = "javascript world";
var stringSecond = "javascript world";
var res = '';


if(stringFirst == stringSecond)
{
```

```
  res = 'strings are equal';
}else
{
  res = 'strings are not equal';
}
 document.write( "Output :- " + res );
</script>
</body>
</html>
```

Output

```
Output:-  strings are equal
```

b. Using toUpperCase()

```
<html>
<head>
<title> program to perform string comparison </title>
</head>
<body>
<script type="text/javascript">
const string1 = 'Are you like JavaScript Program';
const string2 = 'javascript program';

// compare both strings
const result = string1.toUpperCase() === string2.toUpperCase();

if(result) {
    console.log('The strings are similar.');
} else {
    console.log('The strings are not similar.');
}
</script>
```

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

Output

```
The strings are not similar.
```

c. JS String Comparison Using RegEx

```
<html>
<head>
<title> program to perform string comparison </title>
</head>
<body>
<script type="text/javascript">
const string1 = 'JavaScript Program';
const string2 = 'javascript program';

// create regex
const pattern = new RegExp(string1, "gi");

// compare the stings
const result = pattern.test(string2)



if(result) {
    console.log('The strings are similar.');
} else {
    console.log('The strings are not similar.');
}
</script>
</body>
</html>
```

Output

```
The strings are similar.
```

d. localeCompare() Method

```
<html>
<head>
<title> program to perform case insensitive string comparison </title>
</head>
<body>
<script type="text/javascript">
const string1 = 'This is the JavaScript Program';
const string2 = 'this is the javascript program';

const result = string1.localeCompare(string2, undefined, { sensitivity: 'base' });

if(result == 0) {
   console.log('The strings are similar.');
} else {
   console.log('The strings are not similar.');
}

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

Output

```
The strings are similar.
```

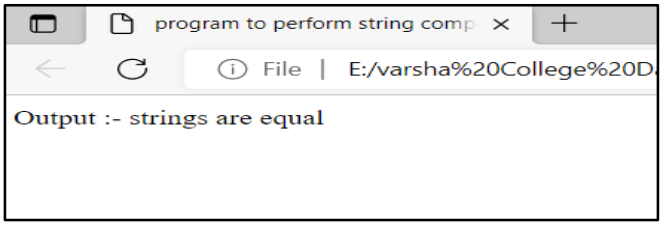
Screenshot's of Output:

a. Using equality operators

expt4a - Notepad

File Edit Format View Help

```
<html>
<head>
<title> program to perform string comparison </title>
</head>
<body>
<script type="text/javascript">

var stringFirst = "javascript world";
var stringSecond = "javascript world";
var res = '';

if(stringFirst == stringSecond)
{
  res = 'strings are equal';
}else
{
  res = 'strings are not equal';
}

document.write( "Output :- " + res );



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

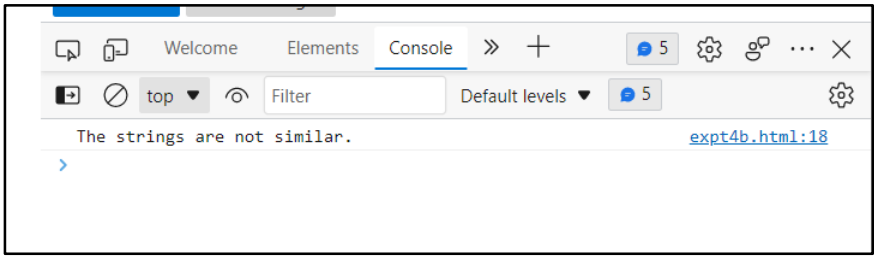


b. Using toUpperCase()

expt4b - Notepad

File Edit Format View Help

```
<html>
<head>
<title> program to perform string comparison </title>
</head>
<body>
<script type="text/javascript">


const string1 = 'Are you like JavaScript Program';
const string2 = 'javascript program';

// compare both strings
const result = string1.toUpperCase() === string2.toUpperCase();

if(result) {
    console.log('The strings are similar.');
} else {
    console.log('The strings are not similar.');
}



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

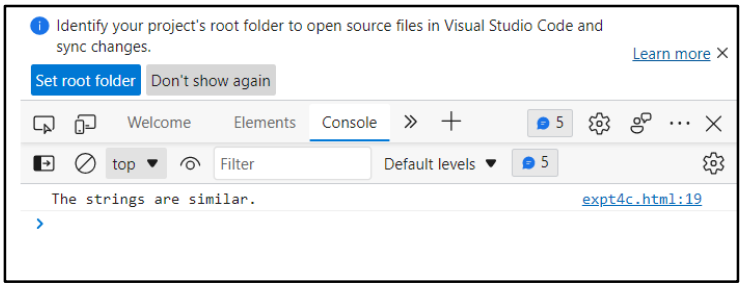


c. JS String Comparison Using RegEx

expt4c - Notepad

File Edit Format View Help

```
<html>
<head>
<title> program to perform string comparison </title>
</head>
<body>
<script type="text/javascript">


const string1 = 'JavaScript Program';
const string2 = 'javascript program';

// create regex
const pattern = new RegExp(string1, "gi");

// compare the stings
const result = pattern.test(string2)

if(result) {
    console.log('The strings are similar.');
} else {
    console.log('The strings are not similar.');
}



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

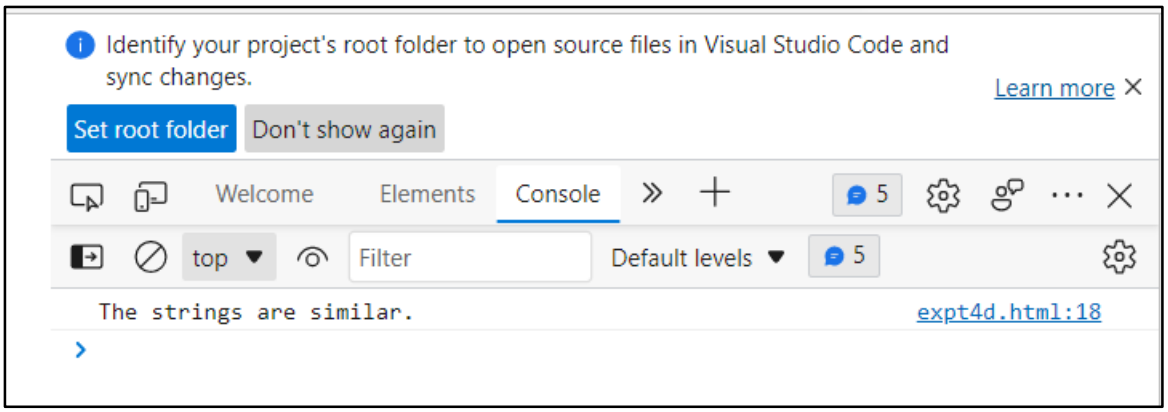


d. localeCompare() Method

expt4d - Notepad

File Edit Format View Help

```
<html>
<head>
<title> program to perform case insensitive string comparison </title>
</head>

<body>

<script type="text/javascript">



const string1 = 'This is the JavaScript Program';
const string2 = 'this is the javascript program';

const result = string1.localeCompare(string2, undefined, { sensitivity:
'base' });

if(result == 0) {
    console.log('The strings are similar.');
} else {
    console.log('The strings are not similar.');
}



</script>

</body>
</html>
```


Identify your project's root folder to open source files in Visual Studio Code and sync changes.
Learn more
Set root folder
Don't show again
Welcome
Elements
Console
5
top
Filter
Default levels
5
The strings are similar.
expt4d.html:18

Conclusion:

Experiment No._____ *Date___/___/2020*

TITLE OF EXPERIMENT: - A Program to create a countdown timer

DIVISION:_____ **BRANCH:** _____

BATCH:_____ **ROLL NO.:** _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 5

Aim: Write a JavaScript program that will create a countdown timer

Prerequisites:

- For this **Javascript Lab**, it is assumed that you have a prior knowledge of HTML coding. It would help if you had some prior exposure to object-oriented programming concepts and a general idea on creating online applications.
- To understand this experiment, you should have the knowledge of the following JavaScript programming topics:

✓ JavaScript Math floor()
✓ JavaScript Date and Time
✓ Javascript setInterval()

Editor:

1.	NotePad
2.	Visual studio code

Theory:

A countdown timer is an accurate timer that can be used for a website or blog to display the countdown to any special event, such as a birthday or anniversary. For countdown timer you know the following methods;

1) The clearInterval() Method
2) The setinterval() Method
3) The setTimeout() Method
4) The clearTimeout() Method

1) Window setInterval() :

Definition and Usage:

The setInterval() method calls a function at specified intervals (in milliseconds).

The setInterval() method continues calling the function until clearInterval() is called, or the window is closed.

1 second = 1000 milliseconds.

Note:

To execute the function only once, use the setTimeout() method instead.

To clear an interval, use the **id** returned from setInterval():

```
myInterval = setInterval(function, milliseconds);
```

Then you can to stop the execution by calling clearInterval():

```
clearInterval(myInterval);
```

Syntax

```
setInterval(function, milliseconds, param1, param2, ...)
```

Parameters

Parameter	Description
function	Required. The function to execute
milliseconds	Required. The execution interval. If the value is less than 10, 10 is used

param1, *param2*, ...	Optional. Additional parameters to pass to the *function* Not supported in IE9 and earlier.

Return Value

Type	Description
A number	The id of the timer. Use this id with clearInterval() to cancel the timer.

Examples

Display "Hello" every second (1000 milliseconds):

```
setInterval(function () {element.innerHTML += "Hello"}, 1000);
```

```
<html>
<body>
<h1>The Window Object</h1>
<h2>The setInterval() Method</h2>
<p id="demo"></p>
<script>
const element = document.getElementById("demo");
setInterval(function() {element.innerHTML += "Hello"}, 1000);
</script>
</body>
</html>
```

OR

Call displayHello every second:

```
setInterval(displayHello, 1000);
```

```
<html>
<body>

<h1>The Window Object</h1>
<h2>The setInterval() Method</h2>

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

<script>
setInterval(displayHello, 1000);

function displayHello() {
  document.getElementById("demo").innerHTML += "Hello";
}
</script>

</body>
</html>
```

Output:

2) clearInterval()

Definition and Usage

The clearInterval() method clears a timer set with the setInterval() method.

Note

To clear an interval, use the id returned from setInterval():

```
myInterval = setInterval(function, milliseconds);
```

Then you can to stop the execution by calling clearInterval():

```
clearInterval(myInterval);
```

Syntax

clearInterval(*intervalId*)

Parameters

Parameter	Description
intervalId	Required. The interval id returned from setInterval().

3) setTimeout()

Definition and Usage

The setTimeout() method calls a function after a number of milliseconds.

1 second = 1000 milliseconds.

Note:

The setTimeout() is executed only once.

If you need repeated executions, use setInterval() instead.

Use the clearTimeout() method to prevent the function from starting.

To clear a timeout, use the **id** returned from setTimeout():

```
myTimeout = setTimeout(function, milliseconds);
```

Then you can to stop the execution by calling clearTimeout():

```
clearTimeout(myTimeout);
```

Syntax:

```
setTimeout(function, milliseconds, param1, param2, ...)
```

Parameters

Parameter	Description
function	Required. The function to execute.
milliseconds	Optional. Number of milliseconds to wait before executing. Default value is 0.
param1, *param2*, ...	Optional. Parameters to pass to the *function.* Not supported in IE9 and earlier.

Return Value

Type	Description
A number	The id of the timer. Use this id with clearTimeout(id) to cancel the timer.

4) clearTimeout()

Definition and Usage

The clearTimeout() method clears a timer set with the setTimeout() method.

Note:

To clear a timeout, use the **id** returned from setTimeout():

myTimeout = setTimeout(*function*, *milliseconds*);

Then you can to stop the execution by calling clearTimeout():

```
clearTimeout(myTimeout);
```

Syntax:

```
clearTimeout(id_of_settimeout)
```

Parameters

Parameter	Description
timeout id	Required. The id returned by the setTimeout() method.

JavaScript Math floor()

The Math.floor() function rounds down a number to the next smallest integer.

Example

```
let number = 38.8;

// round number to nearest smallest number
let roundedNumber = Math.floor(number);

console.log(roundedNumber);

// Output: 38
```

Math.floor() Syntax

The syntax of the Math.floor() function is:

```
Math.floor(x)
```

floor(), being a static method, is called using the Math class name.

Math.floor() Parameters

The Math.floor() function takes in:

- x - A number

Math.floor() Return Value

- Returns the largest integer less than or equal to a given number.
- Returns 0 for null.

STEPS:

Steps of a countdown timer are:

1. Set a valid end date.
2. Calculate the time remaining.
3. Convert the time to a usable format.
4. Output the clock data as a reusable object.
5. Display the clock on the page, and stop the clock when it reaches zero.

Step 1 : Set a Valid End Date
The Valid end date and time should be a string in any of the formats understood by JavaScript's Date.parse() method.
To create a countdown timer using Javascript, we first need to declare a variable that holds the date and time that we want our countdown timer to run down to. We create a **Date** object and then call the **getTime() function** on this object. The **getTime() method** makes the **countDownDate** variable hold the milliseconds since Jan 1, 1970, 00:00:00.000 GMT.

```
var deadline = new Date("dec 31, 2017 15:37:25").getTime();
```

Step 2 : Calculate Remaining Time

First we calculate the time remaining by subtracting the deadline by current date and time then we calculate the number of days,hours,minutes and seconds. The Math.floor() function is used to return the largest integer less than or equal to a given number.

```
var now = new Date().getTime();
var t = deadline - now;
var days = Math.floor(t / (1000 * 60 * 60 * 24));
var hours = Math.floor((t%(1000 * 60 * 60 * 24))/(1000 * 60 * 60));
var minutes = Math.floor((t % (1000 * 60 * 60)) / (1000 * 60));
var seconds = Math.floor((t % (1000 * 60)) / 1000);
```

Step 3: Output the result
In the code below the result is given as output by id="demo"

```
document.getElementById("demo").innerHTML = days + "d " + hours +
"h " + minutes + "m " + seconds + "s ";
```

Step 4: Write some text if the countdown is over
If the countdown timer is over then “expired” will be displayed on the screen.

```
    if (t < 0) {
        clearInterval(x);
        document.getElementById("demo").innerHTML = "EXPIRED";
    }
}, 1000);
```

Program:

```
        <!-- Display the countdown timer in an element -->
<p id="demo"></p>

<script>
// Set the date we're counting down to
var countDownDate = new Date("Sep 1, 2022 15:35:25").getTime();

// Update the count down every 1 second
var x = setInterval(function() {

  // Get today's date and time
  var now = new Date().getTime();

  // Find the distance / Timeleft between now and the countdown date
  var distance = countDownDate - now;

  // Time calculations for days, hours, minutes and seconds
  var days = Math.floor(distance / (1000 * 60 * 60 * 24));
  var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60));
  var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
  var seconds = Math.floor((distance % (1000 * 60)) / 1000);

  // Display the result in the element with id="demo"
  document.getElementById("demo").innerHTML = days + "d " + hours + "h "
  + minutes + "m " + seconds + "s ";

  // If the count down is finished, write some text
  if (distance < 0) {
    clearInterval(x);
    document.getElementById("demo").innerHTML = "EXPIRED";
```

```
  }
}, 1000);
</script>
```

Output:

```
0d 0h 0m 23s
```

EXPIRED

Applications of Countdown Timer

- Used during Events to display the time left for its commencement.
- Used by online commerce websites to display time left for an ongoing sale.
- Used by websites during promotions
- Used in Car racing games,football games etc
- Used in Auction Websites to display the left for placing bids.

Benefits of making a countdown timer in JavaScript than using plugins

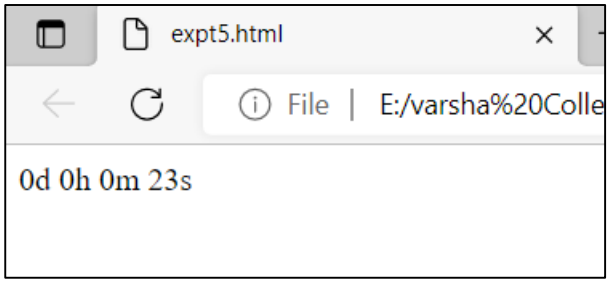
- The code will be lightweight because it will have zero dependencies.
- The website will perform better because there won't be any need of loading external scripts and style sheets.
- The user gets more control because he has built the clock to behave exactly the way he wants it to rather than trying to bend a plugin according to his will.

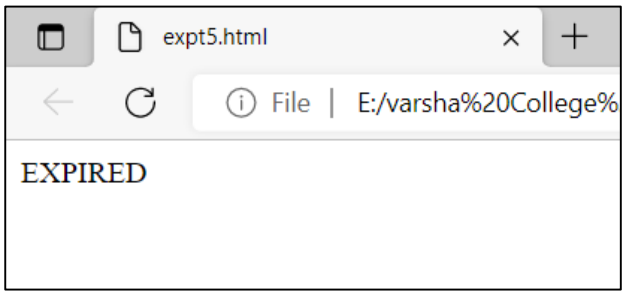
Program:

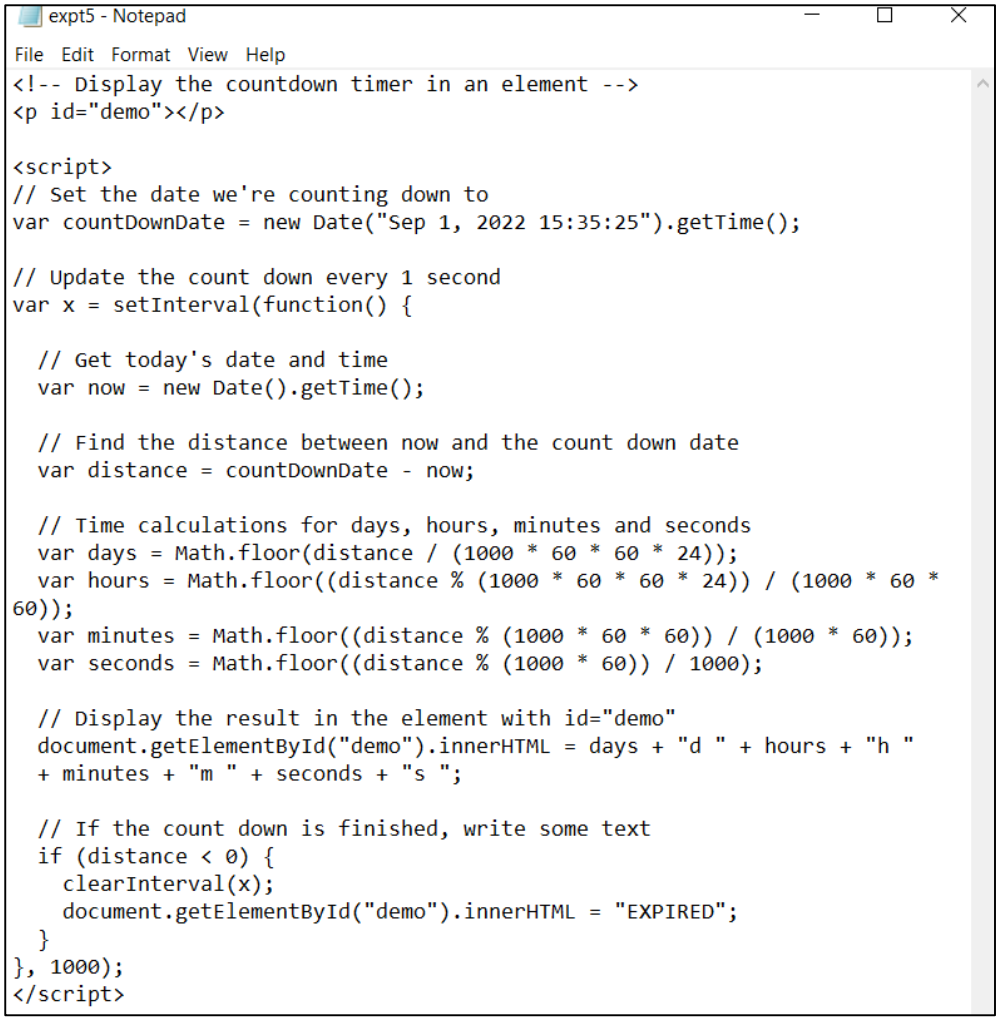
```
<!-- Display the countdown timer in an element -->
<p id="demo"></p>

<script>
// Set the date we're counting down to
var countDownDate = new Date("Sep 1, 2022 15:35:25").getTime();

// Update the count down every 1 second
var x = setInterval(function() {

  // Get today's date and time
  var now = new Date().getTime();

  // Find the distance between now and the count down date
  var distance = countDownDate - now;

  // Time calculations for days, hours, minutes and seconds
  var days = Math.floor(distance / (1000 * 60 * 60 * 24));
```

```
  var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60));
  var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
  var seconds = Math.floor((distance % (1000 * 60)) / 1000);

  // Display the result in the element with id="demo"
  document.getElementById("demo").innerHTML = days + "d " + hours + "h "
  + minutes + "m " + seconds + "s ";

  // If the count down is finished, write some text
  if (distance < 0) {
    clearInterval(x);
    document.getElementById("demo").innerHTML = "EXPIRED";
  }
}, 1000);
</script>
```

Screenshot's of Output:






expt5 - Notepad
File Edit Format View Help
<!-- Display the countdown timer in an element -->

<script>
// Set the date we're counting down to
var countDownDate = new Date("Sep 1, 2022 15:35:25").getTime();
// Update the count down every 1 second
var x = setInterval(function() {
// Get today's date and time
var now = new Date().getTime();
// Find the distance between now and the count down date
var distance = countDownDate - now;
// Time calculations for days, hours, minutes and seconds
var days = Math.floor(distance / (1000 * 60 * 60 * 24));
var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000 * 60 *
60));
var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
var seconds = Math.floor((distance % (1000 * 60)) / 1000);
// Display the result in the element with id="demo"
document.getElementById("demo").innerHTML = days + "d " + hours + "h "
+ minutes + "m " + seconds + "s ";
// If the count down is finished, write some text
if (distance < 0) {
clearInterval(x);
document.getElementById("demo").innerHTML = "EXPIRED";
}
}, 1000);
</script>

Conclusion:

Experiment No._____ *Date___/___/2020*

TITLE OF EXPERIMENT: - A Program to create an array and perform following operations

- **To remove specific element from the array.**
- **Check if an array contains a specified value.**
- **To empty an array**

DIVISION:_____ **BRANCH:** _____

BATCH:_____ **ROLL NO.:** _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 6

Aim: Write a JavaScript program that will create an array and perform following operations

• To remove specific element from the array.
• Check if an array contains a specified value.
• To empty an array

Prerequisites:

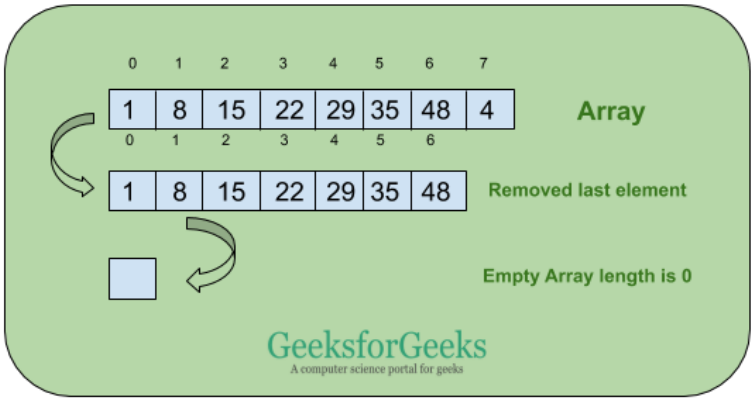
- For this **Javascript Lab**, it is assumed that you have a prior knowledge of HTML coding. It would help if you had some prior exposure to object-oriented programming concepts and a general idea on creating online applications.
- To understand this experiment, you should have the knowledge of the basic JavaScript, JavaScript Array includes(), JavaScript Array indexOf(), JavaScript Arrays

Editor:

1.	NotePad
2.	Visual studio code

Theory:

A) To remove specific element from the array.



JavaScript array is a single variable that is used to store the elements or a group of values. You can add or remove elements from array in any position. In this article, we will discuss different ways to remove elements from array. There are many methods that is used to remove elements from JavaScript array which are discussed below:

- **pop() function:** This method is use to remove elements from the end of an array.
- **shift() function:** This method is use to remove elements from the start of an array.
- **splice() function:** This method is use to remove elements from the specific index of an array.
- **filter() function:** This method is use to remove elements in programmatically way.

Note: There are some other methods that are created by JavaScript inbuilt methods. Below examples illustrate the methods to remove elements from a JavaScript array:

1) Remove Array elements by using pop() method: This method is used to remove the last element of the array and returns the removed element. This function decreases the length of the array by 1.

Example 1:

```
<script>

// JavaScript code to illustrate pop() function

// to remove array elements



function func() {

    var arr = ["shift", "splice", "filter", "pop"];
```

```
    // Popping the last element from the array
    var popped = arr.pop();
    document.write("Removed element: " + popped + "<br>");
    document.write("Remaining elements: " + arr);
}
func();
</script>
```

Output:

```
Removed element: pop
Remaining elements: shift, splice, filter
```

Example 2:

```
<script>
// Declare and initialize an array
var array = ["pop", "splice", "filter", "shift"]

document.write("Original array: " + array + "<br>")

// Loop run while array length not zero
```

```
while (array.length) {



    // Remove elements from array

    array.pop();

}

document.write("Array Length: " + array.length )



</script>
```

Output:

```
Original array: pop, splice, filter, shift
Array Length: 0
```

2) Remove Array elements by using shift() method: This method is used to remove the first element of the array and reducing the size of original array by 1.

Example:

```
<script>

// JavaScript code to illustrate shift() method

// to remove elements from array

function func() {

    var arr = ["shift", "splice", "filter", "pop"];



    // Removing the first element from array
```

```
    var shifted = arr.shift();

    document.write("Removed element: " + shifted + "<br>");

    document.write("Remaining elements: " + arr);

}

func();

</script>
```

Output:

```
Removed element: shift
Remaining elements: splice, filter, pop
```

3) Remove Array elements by using splice() method: This method is used to modify the contents of an array by removing the existing elements and/or by adding new elements. To remove elements by splice() method you can specify the elements in different ways.

Example 1: Use the indexing of splice method to remove elements from a JavaScript array.

```
<script>

// JavaScript code to illustrate splice() function



function func() {

    var arr = ["shift", "splice", "filter", "pop"];



    // Removing the specified element from the array
```

```
    var spliced = arr.splice(1, 1);

    document.write("Removed element: " + spliced + "<br>");

    document.write("Remaining elements: " + arr);

}

func();

</script>
```

Output:

```
Removed element: splice
Remaining elements: shift, filter, pop
```

4) Remove Array elements by using filter() method: This method is used to create a new array from a given array consisting of only those elements from the given array which satisfy a condition set by the argument function. To remove elements by filter() method you can specify the elements in different ways.

Example: Use the value of filter method to remove elements from a JavaScript array.

```
<script>



// JavaScript to illustrate filter() method

function isPositive( value ) {

    return value > 0;

}
```

```
function func() {

    var filtered = [101, 98, 12, -1, 848].filter( isPositive );

    document.write("Positive elements in array: " + filtered);

}

func();

</script>
```

Output:

```
Positive elements in array: 101, 98, 12, 848
```

5) Remove Array elements by using Remove Method: Creating a remove method using filter method to remove elements from a JavaScript array. This methods works in reverse order.

Example:

```
<script>



// Declare and initialize an array

var array = ["lowdash", "remove", "delete", "reset"]



// Using filter method to create a remove method

function arrayRemove(arr, value) {



   return arr.filter(function(geeks){
```

```
    return geeks != value;
  });


}


var result = arrayRemove(array, "delete");
document.write("Remaining elements: " + result)
</script>
```

Output:

```
Remaining elements: lowdash, remove, reset
```

6) Remove Array elements by Delete Operator: Use the delete operator to remove elements from a JavaScript array.

Example:

```
<script>


// Declare and initialize an array
var array = ["lowdash", "remove", "delete", "reset"]


// Delete element at index 2
var deleted = delete array[2];
```

```
document.write("Removed: " + deleted + "<br>");

document.write("Remaining elements: " + array);

</script>
```

Output:

```
Removed: true
Remaining elements: lowdash, remove,,reset
```

7) Remove Array elements by Clear and Reset Operator: Use clear and reset operator to remove elements from a JavaScript array.

Example 1:

```
<script>



// Declare and initialize an array

var array = ["lowdash", "remove", "delete", "reset"]



// Sorting array in another array

var arraygeeks = array



// Delete each element of array

array = []

document.write("Empty array: " + array + "<br>")
```

```
document.write("Original array: " + arraygeeks)



</script>
```

Output:

```
Empty array:
Original array: lowdash, remove, delete, reset
```

8) Remove Array elements using a simple for() loop and a new array: Here a simple for() will be run over the array and the except for the removed element, remaining elements will be pushed into the new array which will be declared inside the function or a method.

Example:

```
let removeElement = (array, n) => {
  let newArray = [];



  for (let i = 0; i < array.length; i++) {
    if (array[i] !== n) {
      newArray.push(array[i]);
    }
  }
  return newArray;
};



let passed_in_array = [1, 2, 3, 4, 5];
```

```
let element_to_be_removed = 2;

let result = removeElement(passed_in_array, element_to_be_removed);



console.log("Remaining elements: " + result);



// This code is contributed by Aman Singla...
```

Output:

```
Remaining elements: 1,3,4,5
```

B) Check if an array contains a specified value.

Check Array Using includes()

```
// program to check if an array contains a specified value

const array = ['you', 'will', 'learn', 'javascript'];

const hasValue = array.includes('javascript');

// check the condition
if(hasValue) {
   console.log('Array contains a value.');
} else {
   console.log('Array does not contain a value.');
}
```

Output

```
Array contains a value.
```

In the above program, the `includes()` method is used to check if an array contains a specified value.

- The `includes()` method returns `true` if the value exists in the array.

- The if...else statement is used to display the result as per the condition.

Check Array Using indexOf()

```
// program to check if an array contains a specified value

const array = ['you', 'will', 'learn', 'javascript'];

const hasValue = array.indexOf('javascript') !== -1;

// check the condition
if(hasValue) {
    console.log('Array contains a value.');
} else {
    console.log('Array does not contain a value.');
}
```

Output

```
Array contains a value.
```

In the above program, the indexOf() method is used with the if...else statement to check if an array contains a specified value.

The indexOf() method searches an array and returns the position of the first occurrence. If the value cannot be found, it returns **-1**.

Note: Both includes() and indexOf() are case sensitive. Hence, **J** and **j** are different.

C) To empty an array

Suppose you have the following array and want to remove all of its elements:

```
let a = [1,2,3];
```
Code language: JavaScript (javascript)

The following shows you several methods to make an array empty.

1) **Assigning it to a new empty array**

This is the fastest way to empty an array:

```
a = [];
```

This code assigned the array a to a new empty array. It works perfectly if you do not have any references to the original array.

See the following example:

```
let b = a;
a = [];
console.log(b); // [1,2,3]
```
Code language: JavaScript (javascript)

In this example, first, the b variable references the array a. Then, the a is assigned to an empty array. The original array still remains unchanged.

2) **Setting its length to zero**

The second way to empty an array is to set its length to zero:

```
a.length = 0;
```

The length property is read/write property of an Array object. When the length property is set to zero, all elements of the array are automatically deleted.

3) **Using splice() method**

The third way to empty an array is to remove all of its elements using the splice() method as shown in the following example:

```
a.splice(0,a.length);
```
Code language: CSS (css)

In this solution, the splice() method removed all the elements of the a array and returned the removed elements as an array.

4) **Using pop() method**

The fourth way to empty an array is to remove each element of the array one by one using the while loop and pop() method:

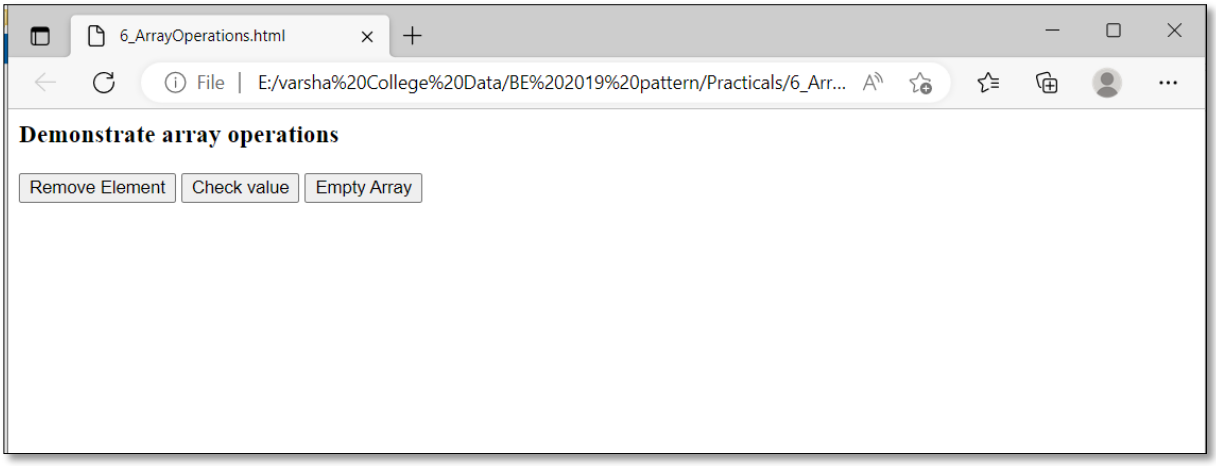
```
while(a.length > 0) {
   a.pop();
}
```

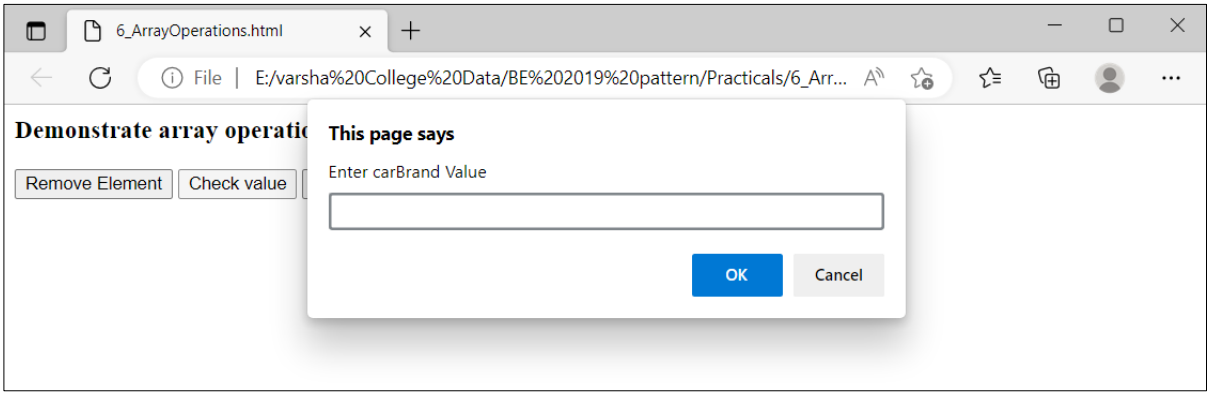
Program:

```
<html>
<body>
<h3>Demonstrate array operations</h3>

<button onclick="removeElement()">Remove Element</button>
<button onclick=" chkValue()">Check value</button>
<button onclick="emptyArray()">Empty Array</button>

<script>



function removeElement() {
   var shoeBrand = ["Nike", " Adidas", " Sparks", " RedTape"];

   console.log("Elements in array before removing: <br>" + shoeBrand );

   // Removing last element from the array
   var poppedElement = shoeBrand.pop();
   console.log("Removed last element from array using pop(): " + poppedElement );

    //display remaining elements present in array after removing
   console.log("New array: " + " " + shoeBrand);

    //removing 2 elemnts from position '1'.
   shoeBrand.splice(1,2);
   console.log("Removed elements  from array using spilce(1,2) : New Array: " + " " +
shoeBrand );
```

```
}

function chkValue(){
  var carBrand = ["Maruti", " BMW", " Kia", " Tata"];
   console.log(carBrand);
    var str1 =   prompt("Enter carBrand Value");
   const hasValue = carBrand.includes(str1)
   console.log(carBrand);
   console.log("Value entered: " + str1);
  //check the condition
   if(hasValue) {
    console.log('Array contains: ' + str1);
    }
    else {
    console.log('Array does not contain:  ' + str1);
          }
}

function emptyArray(){
 var carBrand = ["Maruti", " BMW", " Kia", " Tata"];
  console.log(carBrand);

  carBrand.splice(0,carBrand.length);
//while(a.length > 0) {
  //  a.pop();
  console.log("Empty Array " + " " + carBrand );
}
</script>
</body>
</html>
```

Screenshot's of Output:





Conclusion:

*Experiment No.*_____ *Date___/___/2020*

TITLE OF EXPERIMENT: - A Program to illustrate different Set operations like-

- **Union**
- **Intersection**
- **Difference**
- **Set Difference**

DIVISION:_____ **BRANCH:** _____

BATCH:_____ **ROLL NO.:** _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 7

Aim: Write a JavaScript program to illustrate different Set operations like-
• Union
• Intersection
• Difference
• Set Difference

Prerequisites:

- For this **Javascript Lab**, it is assumed that you have a prior knowledge of HTML coding. It would help if you had some prior exposure to object-oriented programming concepts and a general idea on creating online applications.
- To understand this experiment, you should have the knowledge of the basic **JavaScript, for loop, arithmetic operators, relational operators,**

- JavaScript Set and WeakSet
- JavaScript for... of Loop
- JavaScript Function and Function Expressions

Editor:

1.	NotePad
2.	Visual studio code

Theory:

Examples of different Set operations:

Input: A = {0, 2, 4, 6, 8}

```
B = {1, 2, 3, 4, 5}


Output:
 Union: [0, 1, 2, 3, 4, 5, 6, 8]
 Intersection: [2, 4]
 Difference: [8, 0, 6]
 Symmetric difference: [0, 1, 3, 5, 6, 8]
```

Union

Union (a ∪ b): create a set that contains the elements of both set a and set b.

```
let a = new Set([1,2,3]);
let b = new Set([4,3,2]);
let union = new Set([...a, ...b]);
   // {1,2,3,4}
```

The pattern is always the same:

- Convert one or both sets to arrays.
- Perform the operation.
- Convert the result back to a set.

As explained in above, the spread operator (...) inserts the elements of something iterable (like a set) into an array. Therefore, [...a, ...b] means that a and b are converted to arrays and concatenated. It is equivalent to [...a].concat([...b]).

Intersection

Intersection (a ∩ b): create a set that contains those elements of set a that are also in set b.

```
let a = new Set([1,2,3]);
let b = new Set([4,3,2]);
let intersection = new Set(
   [...a].filter(x => b.has(x)));
   // {2,3}
```

Steps: Convert a to an array, filter the elements, convert the result to a set.

Difference

Difference (a \ b): create a set that contains those elements of set a that are not in set b. This operation is also sometimes called *minus* (-).

```
let a = new Set([1,2,3]);
let b = new Set([4,3,2]);
let difference = new Set(
   [...a].filter(x => !b.has(x)));
   // {1}
```

Example 1: Set Union Operation

```
// perform union operation
// contain elements of both sets
function union(a, b) {
   let unionSet = new Set(a);
   for (let i of b) {
      unionSet.add(i);
   }
   return unionSet
}

// two sets of fruits
const setA = new Set(['apple', 'mango', 'orange']);
const setB = new Set(['grapes', 'apple', 'banana']);

const result = union(setA, setB);

console.log(result);
Run Code
```

Output

```
Set {"apple", "mango", "orange", "grapes", "banana"}
```

The set union operation combines elements of both sets into one.

A new set unionSet is created using new Set(). The unionSet variable contains all the values of setA. Then, the for...of loop is used to iterate through all the elements of setB and add them to unionSet using the add() method.

The set does not contain duplicate values. Hence, if the set contains the same value, the latter value is discarded.

Example 2: Set Intersection Operation

```
// perform intersection operation
```

```
// elements of set a that are also in set b

function intersection(setA, setB) {

    let intersectionSet = new Set();

    for (let i of setB) {

        if (setA.has(i)) {

            intersectionSet.add(i);

        }

    }

    return intersectionSet;

}



// two sets of fruits

const setA = new Set(['apple', 'mango', 'orange']);

const setB = new Set(['grapes', 'apple', 'banana']);

const result = intersection(setA, setB);

console.log(result);
```

Output

```
Set {"apple"}
```

The set intersection operation represents elements that are present in both setA and setB.

A new set intersectionSet is created using new Set(). Then, the for...of loop is used to iterate through the setB. For every element that is present in both setA and setB, they are added to the intersection set.

Example 3: Set Difference Operation

```
// perform difference operation
// elements of set a that are not in set b
function difference(setA, setB) {
   let differenceSet = new Set(setA)
   for (let i of setB) {
      differenceSet.delete(i)
   }
   return differenceSet
}

// two sets of fruits
const setA = new Set(['apple', 'mango', 'orange']);
const setB = new Set(['grapes', 'apple', 'banana']);

const result = difference(setA, setB);

console.log(result);
Run Code
```

Output

```
Set {"mango", "orange"}
```

The set difference operation represents elements that are present in one set and not in another set.

The differenceSet contains all the elements of setA. Then, the for...of loop is used to iterate through all the elements of setB. If the element that is present in setB is also available in setA, that element is deleted using delete() method.

Example 4: Set Subset Operation

```
// perform subset operation
// true if all elements of set b is in set a
function subset(setA, setB) {
    for (let i of setB) {
        if (!setA.has(i)) {
            return false
        }
    }
    return true
}

// two sets of fruits
const setA = new Set(['apple', 'mango', 'orange']);
const setB = new Set(['apple', 'orange']);

const result = subset(setA, setB);

console.log(result);
```

Output

```
true
```

The set subset operation returns true if all the elements of setB are in setA.

The for...of loop is used to loop through the elements of setB. If any element that is present is setB is not present in setA, false is returned.

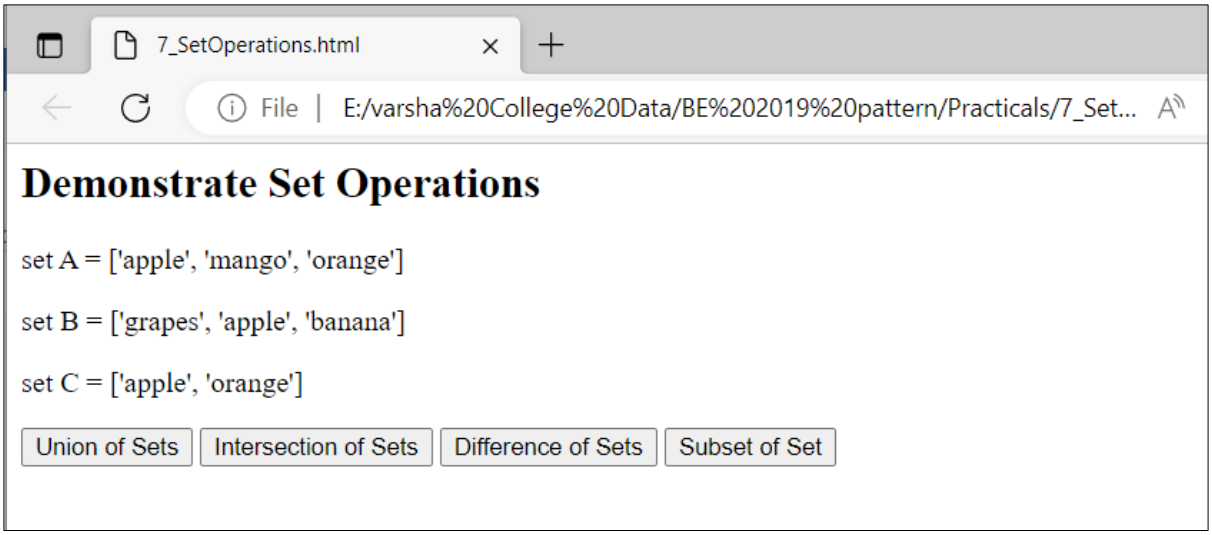
Program:

```
<html>
<head>
<h2> Demonstrate Set Operations </h2>
<p> set A = ['apple', 'mango', 'orange'] </p>
<p> set B = ['grapes', 'apple', 'banana'] </p>
<p> set C = ['apple', 'orange'] </p>
<button onclick="union()">Union of Sets</button>
<button onclick="intersection()">Intersection of Sets</button>
```

```
<button onclick="difference()">Difference of Sets</button>
<button onclick="subset()">Subset of Set</button>
<script>

// two sets of fruits
const setA = new Set(['apple', 'mango', 'orange']);
const setB = new Set(['grapes', 'apple', 'banana']);
const setC = new Set(['apple', 'orange']);

function union() {
    let unionSet = new Set(setA);
    for (let i of setB) {
        unionSet.add(i);
    }
console.log(unionSet);

}

function intersection() {
    let intersectionSet = new Set();

    for (let i of setB) {
        if (setA.has(i)) {
            intersectionSet.add(i);
        }
    }
   console.log(intersectionSet);
}

function difference() {
```

```
    let differenceSet = new Set(setA)
    for (let i of setB) {
        differenceSet.delete(i)
    }
console.log(differenceSet);
}

function subset() {
    for (let i of setC) {
        if (!setA.has(i)) {
            console.log(false);
        }
    }


 console.log(true);
}



</script>
</head>
</html>
```

Screenshot’s of Output:



Conclusion:

Experiment No._____ *Date___/___/2020*

TITLE OF EXPERIMENT: - A Program to create a Home page of any website and change background color using

- **On mouse over event**
- **On focus event**

DIVISION:_____ BRANCH: _____

BATCH:_____ ROLL NO.: _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 8

Aim: Write a JavaScript program to create a Home page of any website and change background color using
• On mouse over event
• On focus event

Prerequisites:

- For this **Javascript Lab**, it is assumed that you have a prior knowledge of HTML coding. It would help if you had some prior exposure to object-oriented programming concepts and a general idea on creating online applications.
- To understand this experiment, you should have the knowledge of the basic **JavaScript,** On mouse over event, On focus event

Editor:

1.	NotePad
2.	Visual studio code

Theory:

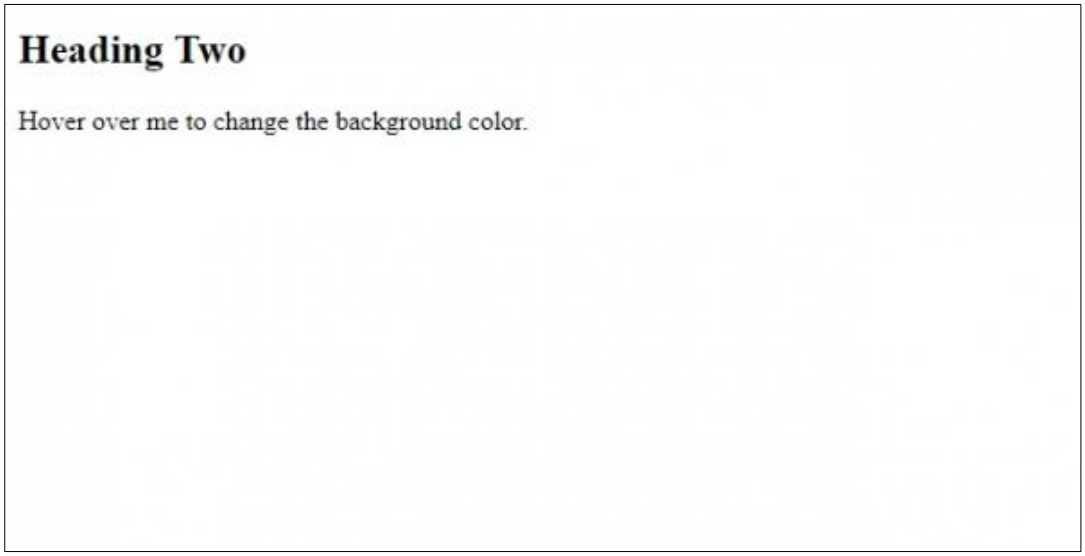
The onmouseover property allows you set a script when the mouse pointer is moved onto an element. To change the background color, use the HTML DOM backgroundColor property.

Let us see an example to implement the onmouseover property and change the background color −

Example

```
<!DOCTYPE html>
<html>
<body>
<h2>Heading Two</h2>
  <a onmouseover="document.body.style.backgroundColor ='orange'">Hover over me to change the background color.</a><br>
</body>
</html>
```

Output



Now hover over the text to change the background color of the web page −

The working of onmouseover event is shown by changing the colours of a paragraph by taking the mouse over a particular colour.

Syntax:
document.bgColor = 'nameOfColor'

HTML code that will change the colour of the background when the mouse is moved over a particular colour. Background colour property specifies the background colour of an element.

Example:

```
<html>
 <p>
  <!-- when mouse is move over the colour name, colour change -->
  <a onmouseover="document.bgColor='greem'">Bright Green</a><br>
  <a onmouseover="document.bgColor='red'">Red</a><br>
  <a onmouseover="document.bgColor='magenta'">Magenta</a><br>
  <a onmouseover="document.bgColor='purple'">Purple</a><br>
  <a onmouseover="document.bgColor='blue'">Blue</a><br>
  <a onmouseover="document.bgColor='yellow'">Yellow</a><br>
  <a onmouseover="document.bgColor='black'">Black</a><br>
  <a onmouseover="document.bgColor='orange'">Orange</a><br>
 </p>
</html>
```

Output:

Initially, the background colour is white-

Bright Green
Red
Magenta
Purple
Blue
Yellow
White
Black
Orange

When the mouse is moved over the “Bright Green” color-

Example:

Before we get started, let’s create an HTML element with an ID:

```
<div id="myID"></div>
```

Now add some CSS style to give it a height and width as well as the background color red:

```
#myID {

height: 400px;

width: 500px;

background-color: red;

}
```

Now if we run the code, we will able to see a rectangle with the red background color. In this time, there will be no change in background color occur if we mouse over the element. To change the background color, we need to write some JavaScript code that is given below:

```
            getElementById("myID").addEventListener("mouseover", function() {

document.getElementById("myID").style.backgroundColor = "green";

});

document.getElementById("myID").addEventListener("mouseout", function() {

document.getElementById("myID").style.backgroundColor = "red";
```

```
});
```

In the above JavaScript code, first, we use the mouseover event to change the background color from red to green.

We also want to bring back the color to red after we move our mouse out of the element. So we have also used the mouseout event. This event will occur when we will remove our mouse cursor from the element.

Now we are ready to run our code. let's see how it works if we test it on our browser.

If you run the complete code on your browser, then you will be able to see the red rectangle. But after you keep your mouse cursor over the element, the color will change into the green. When you remove the mouse cursor from the element, the background color will again come back to red.

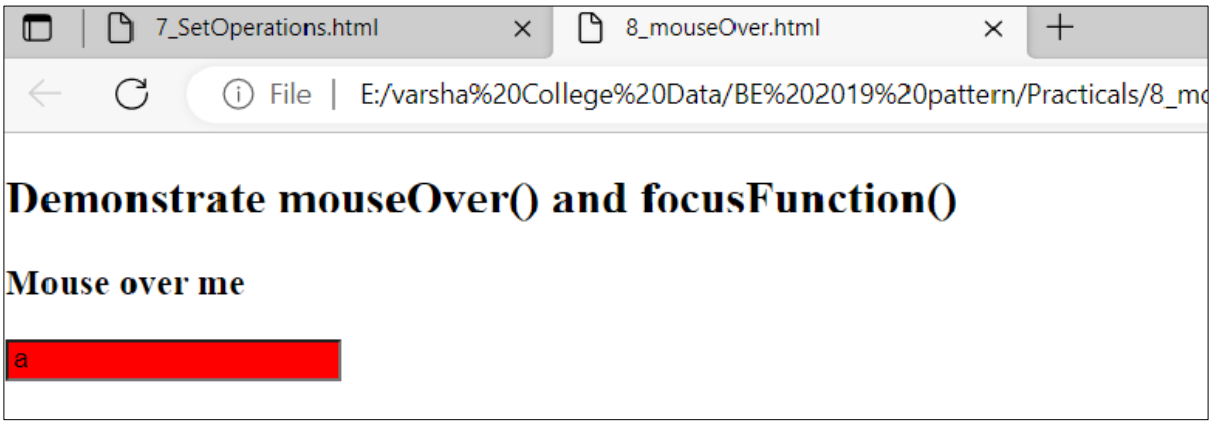
So we are able to successfully change the background color on mouse hover with JavaScript.

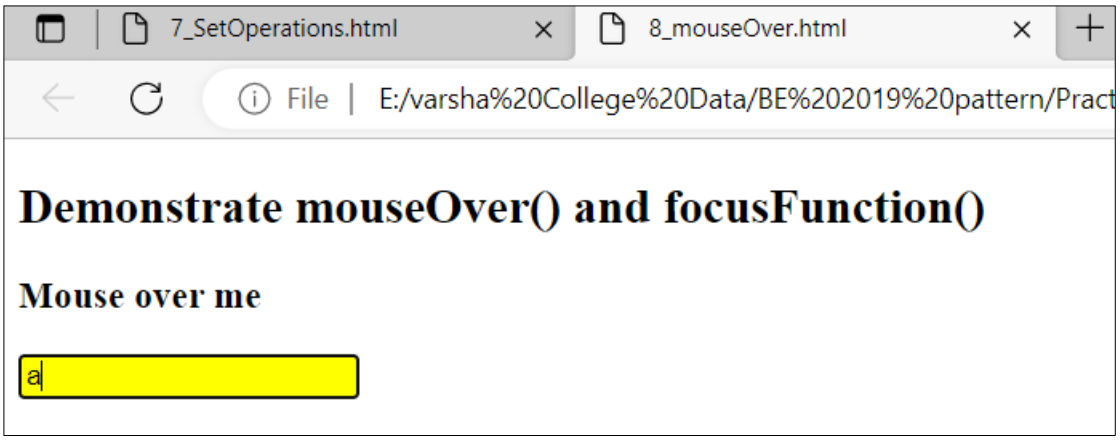
Program:

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

<h2>Demonstrate mouseOver() and focusFunction()</h2>

<h3 id="demo" onmouseover="mouseOver()" onmouseout="mouseOut()">Mouse over me</h3>

<input type="text" placeholder = "Enter Text" id="focus" onfocus="focusFunction()" onblur="blurFunction()">

<script>
function mouseOver() {
  document.getElementById("demo").style.color = "red";
}

function mouseOut() {
```

```
  document.getElementById("demo").style.color = "black";
}

function focusFunction() {
document.getElementById("focus").style.background = "yellow";
}

function blurFunction() {
document.getElementById("focus").style.background = "red";
}
</script>

</body>
</html>
```

Screenshot's of Output:




7_SetOperations.html
8_mouseOver.html
File | E:/varsha%20College%20Data/BE%202019%20pattern/Pract
Demonstrate mouseOver() and focusFunction()
Mouse over me
a

Conclusion:

Experiment No._____ *Date___/___/2020*

TITLE OF EXPERIMENT: - Design and implement a simple calculator using Java script for operations like addition multiplication, subtraction, division, square of a number etc:

- Design a calculator like text field for input and output, buttons for numbers and operations etc.
- Validate input values
- Prompt / Alerts for invalid values etc.

DIVISION:_____ **BRANCH:** _____

BATCH:_____ **ROLL NO.:** _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 9

Aim: Design and implement a simple calculator using Java script for operations like addition, multiplication, subtraction, division, square of a number etc:

- Design a calculator like text field for input and output, buttons for numbers and operations etc.
- Validate input values
- Prompt / Alerts for invalid values etc.

Prerequisites:

- For this **Javascript Lab**, it is assumed that you have a prior knowledge of HTML coding. It would help if you had some prior exposure to object-oriented programming concepts and a general idea on creating online applications.
- To understand this experiment, you should have the knowledge of the basic **JavaScript, for loop, JavaScript if...else Statement, JavaScript switch Statement, Function.**

Editor:

1.	NotePad
2.	Visual studio code

Theory:

The **Calculator** is a portable device used in our daily life to perform various mathematical functions such as **addition, subtraction, multiplication, division**, root, etc. However, we have scientific or sophisticated calculators used to solve complex tasks such as trigonometry functions, degrees, exponential operators, log functions, hyperbolic functions, square root, and so on.

Example 1: Simple Calculator with if..else if...else

```
// program for a simple calculator

// take the operator input
const operator = prompt('Enter operator ( either +, -, * or / ): ');

// take the operand input
const number1 = parseFloat(prompt('Enter first number: '));
const number2 = parseFloat(prompt('Enter second number: '));

let result;

// using if...else if... else
if (operator == '+') {
    result = number1 + number2;
}
else if (operator == '-') {
    result = number1 - number2;
}
else if (operator == '*') {
    result = number1 * number2;
}
else {
    result = number1 / number2;
}

// display the result
console.log(`${number1} ${operator} ${number2} = ${result}`);
```

Output

```
Enter operator ( either +, -, * or / ): *
Enter first number: 3.4
Enter second number: 5.6
3.4 * 5.6 = 19.04
```

In the above example, the user is prompted to enter an operator (either **+**, **-**, ***** or **/**) and two numbers.

The `parseFloat()` converts the numeric string value to a floating-point value.

The if...else if...if statement is used to check the condition that the user has entered for the operator. The corresponding operation is performed and the output is displayed.

Example 2: Simple Calculator with switch

```
// program for a simple calculator
let result;

// take the operator input
const operator = prompt('Enter operator ( either +, -, * or / ): ');

// take the operand input
const number1 = parseFloat(prompt('Enter first number: '));
const number2 = parseFloat(prompt('Enter second number: '));

switch(operator) {
   case '+':
       result = number1 + number2;
      console.log(`${number1} + ${number2} = ${result}`);
      break;

   case '-':
       result = number1 - number2;
      console.log(`${number1} - ${number2} = ${result}`);
      break;

   case '*':
       result = number1 * number2;
      console.log(`${number1} * ${number2} = ${result}`);
      break;

   case '/':
       result = number1 / number2;
      console.log(`${number1} / ${number2} = ${result}`);
      break;

   default:
      console.log('Invalid operator');
      break;
}
```

Output

```
Enter operator: +
Enter first number: 4
Enter second number: 5
4 + 5 = 9
```

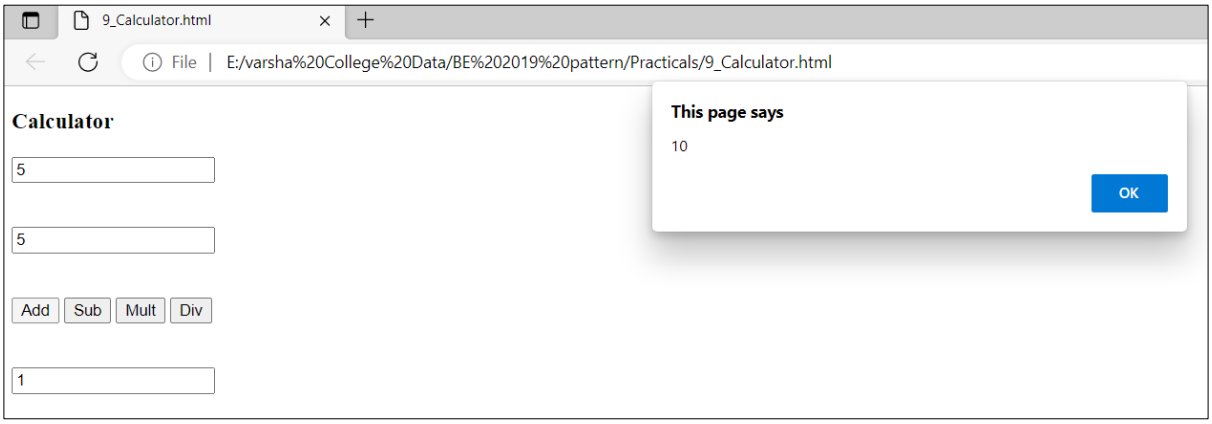
In above program, the user is asked to enter either **+**, **-**, ***** or **/**, and two numbers. Then, the switch statement executes cases based on the user input.

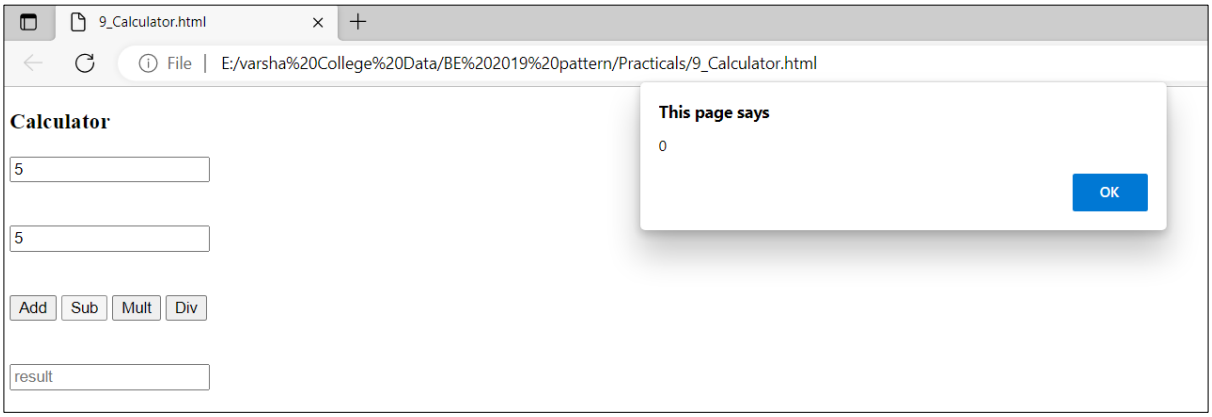
Program:

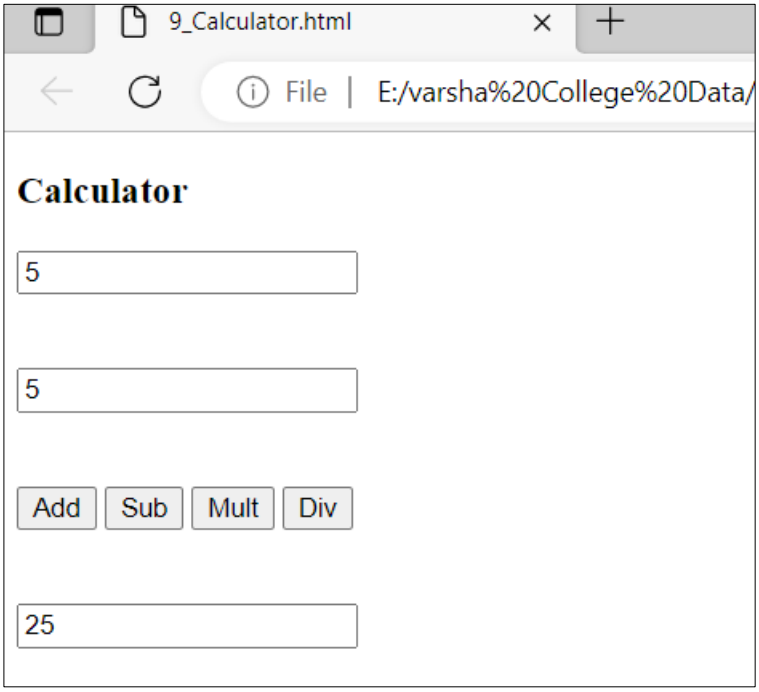
```
<!DOCTYPE html>
<html>
<body>
<h3>Calculator</h3>
<input id ="text1" placeholder = "Enter Num1">
<br>
<br>
<br>
<input id ="text2" placeholder = "Enter Num2">
<br>
<br>
<br>
<button onclick="sum()" id = "btnl">Add</button>
<button onclick="diff()" id = "btnl">Sub</button>
<button onclick="mul()" id = "btnl">Mult</button>
<button onclick="div()" id = "btnl">Div</button>
<br>
<br>
<br>
<input id = "text3" placeholder = "result">
<script>
function sum(){
var x = parseFloat(document.getElementById("text1").value);
```

```
var y = parseFloat(document.getElementById("text2").value);
var s1 = x + y;
document.getElementById("text3").value = s1;
alert(s1);
}
function diff(){
var x = parseFloat(document.getElementById("text1").value);
var y = parseFloat(document.getElementById("text2").value);
var s2 = x - y;
document.getElementById("text3").value = s2;
alert(s2);
}
function mul(){
var x = parseFloat(document.getElementById("text1").value);
var y = parseFloat(document.getElementById("text2").value);
var s3 = x * y;
document.getElementById("text3").value = s3;
alert(s3);
}
function div(){
var x = parseFloat(document.getElementById("text1").value);
var y = parseFloat(document.getElementById("text2").value);
var s4 = x / y;
document.getElementById("text3").value = s4;
alert(s4);
}

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

Screenshot's of Output:







Conclusion: