Vidyalankar Polytechnic

**Program: Computer Engineering**
**(NBA Accredited)**

# A Laboratory Manual for Client-Side Scripting Language (22519)

Prepared By

**Yogita Khandagale**

A Laboratory Manual for

# Client-Side Scripting Language

# (22519)

Semester - V

## Institute Vision and Mission

### Vision

To achieve excellence in imparting technical education so as to meet the professional and societal needs.

### Mission

- Developing technical skills by imparting knowledge and providing hands on experience.
- Creating an environment that nurtures ethics, leadership and team building.
- Providing industrial exposure for minimizing the gap between academics & industry.

## Department of Computer Engineering (NBA Accredited)

### Vision

To empower students with domain knowledge of Computer Engineering and interpersonal skills to cater to the industrial and societal needs.

### Mission

- Developing technical skills by explaining the rationale behind learning.
- Developing interpersonal skills to serve the society in the best possible manner.
- Creating awareness about the ever changing professional practices to build industrial adaptability.

## Program Educational Objectives (PEO)

| Program Educational Objectives (PEOs) | |
|---|---|
| PEO-1 | Provide socially responsible, environment friendly solutions to Information technology related broad-based problems adapting professional ethics. |
| PEO-2 | Adapt state-of-the-art Information Technology broad-based techniques to work in multi-disciplinary work environments. |
| PEO-3 | Solve broad-based problems individually and as a team member communicating effectively in the world of work. |

| Program Outcomes (POs) | |
|---|---|
| 1 | **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems. |
| 2 | **Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods. |
| 3 | **Design/ development of solutions**: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs. |
| 4 | **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements. |
| 5 | **Engineering practices for society, sustainability and environment:** Apply appropriate technology in context of society, sustainability, environment and ethical practices. |
| 6 | **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities. |
| 7 | **Life-long learning Ability:** To analyse individual needs and engage in updating in the context of technological changes. |

| Program Specific Outcomes (PSOs) | |
|---|---|
| 1 | **Computer Software and Hardware Usage:**<br>Use state-of-the-art technologies for operation and application of computer software and hardware. |
| 2 | **Computer Engineering Maintenance:**<br>Maintain computer engineering related software and hardware systems. |

| Course Outcomes (COs) | |
|---|---|
| CO-1 | Create interactive web pages using program flow control structure. |
| CO-2 | Implement Arrays and functions in Java Script. |
| CO-3 | Create event-based web forms using Java Script. |
| CO-4 | Use Java Script for handling cookies. |
| CO-5 | Create interactive web page using regular expressions for validations |
| CO-6 | Create Menus and Navigations in Web Pages. |

Vidyalankar Polytechnic

**Certificate**

This is to certify that

Mr/Ms.....

Roll No..... of Fifth Semester of Diploma in **Computer Engineering** of Institute Vidyalankar Polytechnic (Institute Code: 0568) has completed the Term Work satisfactory in Course **Client Side Scripting Language (22519)** for the Academic Year .....

As prescribed in the Curriculum.

Place:..... Enrollment No:.....

Date:..... Seat No.....

Subject Teacher Head of Department Principal

**List of Experiments and Progressive Assessment for Term Work**

Academic Year:…… Name of Faculty……

Course and Code:…… Subject and Code:……

Name of Candidate:……..

Enrollment No…… Roll No……

| Sr. No. | Title of Experiment | Date of Performance | Date of Submission | Assessment Mark out of 50 | Sign. Of Teacher |
|---|---|---|---|---|---|
| 1 | Write simple Java Script with HTML for arithmetic expression evaluation and message printing. | | | | |
| 2 | Develop Java Script to use decision making and looping statements. | | | | |
| 3 | Develop Java Script to implement Array functionalities. | | | | |
| 4 | Develop Java Script to implement functions | | | | |
| 5 | Develop Java Script to implement Strings. | | | | |
| 6 | Create a Web Page using Form Elements. | | | | |
| 7 | Create a Web Page to implement Form Events-Part-1 | | | | |
| 8 | Create a Web Page to implement Form Events-Part-2 | | | | |
| 9 | Develop a Web Page using Intrinsic Java Functions. | | | | |
| 10 | Develop a Web Page for creating session and persistent cookies. Observe the effects with Browser cookie settings. | | | | |
| 11 | Develop a Web Page for placing the window on the screen and working with child window. | | | | |

| Sr. No. | Title of Experiment | Date of Performance | Date of Submission | Assessment Mark out of 50 | Sign. Of Teacher |
|---|---|---|---|---|---|
| 12 | Develop a Web Page for validation of form fields using regular expressions. | | | | |
| 13 | Create a Web Page with Rollovers effect. | | | | |
| 14 | Develop a Web Page for implementing Menus. | | | | |
| 15 | Develop a Web Page for implementing Status Bar and Web Page Protection. | | | | |
| 16 | Develop a Web Page for implementing Slideshow, Banner. | | | | |
| **Total Marks** | | | | | |
| **Out of 50** | | | | | |
| **BSA:** 1) Work with <iframe><br>2) Work with jQuery | | | | | |

**Assessment Scheme**

| Performance Indicators (15 Marks) | | Weightage |
|---|---|---|
| **Process related (15 marks)** | | 30 % |
| 1 | Logic formation | 10 % |
| 2 | Debugging Ability | 10 % |
| 3 | Follow ethical practices | 10 % |
| **Product related (35 Marks)** | | 70 % |
| 4 | Expected Output | 30 % |
| 5 | Timely Submission | 30 % |
| 6 | Answer to sample Questions | 10 % |
| | | 100 % |

Vidyalankar Polytechnic

**Program: Information Technology**

| | |
|---|---|
| Course: Client-Side Scripting | Course Code: **22519** |
| Semester: **V** | Class: |
| Laboratory No: | Name of Subject Teacher: |
| Name of Student: | Roll ID and Enrollment: |

**Experiment No: 01**

**Write simple Java Script with HTML for arithmetic expression evaluation and message printing.**

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | 1)Notepad editor<br>2)Any Web Browser |

JavaScript is a limited-featured programming language used by web developers to do things that HTML cannot do, Using HTML we can only design a web page but you can not run any logic on web browser like addition of two numbers, check any condition, looping statements (for, while), decision making statement (if-else) at client side. All these are not possible using HTML So for perform all these tasks at client side you need to use JavaScript.

**Theoretical Background:** Javascript is a widely used scripting language originally developed by Netscape for both client-side and server-side scripting. Client-side Javascript is used widely and supported well by major browsers including NN, IE, AOL, MOZILLA, and Opera. We shall present client-side Javascript for adding dynamism and interactivity to Web pages and will refer to it simply as Javascript.

**Proposition-1: First Java Script Program**

Create a php webpage and print "hello world".

```
<html >
<head>
<title>Hello world! JavaScript</title>
</head>
<body>
<script>
document.write('Hello, world!')
</script>
</body>
</html>
```

Output:

Hello World

**Proposition-2: Expressions**

Many JavaScript statements contain a mathematical expression that tells the browser to perform a mathematical operation.

### 1.4.1 Primary Expressions:

The simplest expressions, known as *primary expressions*, are those that stand alone—they do not include any simpler expressions. Primary expressions in JavaScript are constant or *literal* values, certain language keywords, and variable references. Literals are constant values that are embedded directly in your program.

**Example:**

```
1.23            // A number literal
"hello"         // A string literal
/pattern/       // A regular expression literal
```

JavaScript syntax for number literals was covered in Numbers. String literals were documented in Text. Some of JavaScript's reserved words are primary expressions:

```
true                // Evalutes to the boolean true value
false           // Evaluates to the boolean false value
null                // Evaluates to the null value
this                // Evaluates to the "current" object
```

Another type of primary expression is the bare variable reference:

```
i                   // Evaluates to the value of the variable i.
sum             // Evaluates to the value of the variable sum.
undefined       // undefined is a global variable, not a keyword like null
```

When any identifier appears by itself in a program, JavaScript assumes it is a variable and looks up its value. If no variable with that name exists, the expression evaluates to the undefined value.

### Parts of an Expression

A *mathematical expression* consists of two parts: operands and operators.

An *operand* is the value. An *operator* is the symbol that tells the browser how to evaluate the mathematical expression.

The operands are the numbers in the following mathematical expression.

The addition symbol (+) is the operator. The browser evaluates this mathematical expression by adding the value on the right side of the operator to the value on the left side of the operator:

```
1 + 1
```

### Proposition-3: Types of Operators

JavaScript uses five types of operators:

1. Arithmetic Operators
2. Logical Operators
3. Assignment Operators
4. Comparison Operators
5. Conditional Operators.
6. Bitwise Operators

**1) Arithmetic operators:** JavaScript supports the following arithmetic operators , Assume A hold 10 and B holds 20

| Operator | Description | Example |
|---|---|---|
| + | Addition | A + B will give 30 |
| - | Subtraction | A - B will give -10 |
| * | Multiplication | A * B will give 200 |
| / | Division | B / A will give 2 |
| % | Modulus | B % A will give 0 |
| ++ | Increment | A++ will give 11 |
| -- | Decrement | A-- will give 9 |

**2) Logical Operators:** Logical operators are used to combine two logical expressions into one expression. A logical expression is an expression that evaluates to either true or false. Logical expressions are used in JavaScript to make decisions.

| Operator | Description | Example |
|---|---|---|
| && (Logical AND) | If both the operands are non-zero, then the condition becomes true. | (A && B) is true. |
| \|\| (Logical OR) | If any of the two operands are non-zero, then the condition becomes true. | (A \|\| B) is true. |
| ! (Logical NOT) | Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. | (A && B) is false. |

**3) Assignment Operators:** The *assignment operator* assigns the value from the right side of the operator to the variable on the left side of the operator.

| Operator | Description | Example |
|---|---|---|
| = | Assign | Assigns values from the right side operand to the left side operand<br>**Ex:** C = A + B will assign the value of A + B into C |
| += | Add value then assign | It adds the right operand to the left operand and assigns the result to the left operand.<br>**Ex:** C += A is equivalent to C = C + A |
| -= | Subtract value then assign | It subtracts the right operand from the left operand and assigns the result to the left operand.<br>**Ex:** C -= A is equivalent to C = C - A |
| *= | Multiply value the assign | It multiplies the right operand with the left operand and assigns the result to the left operand.<br>**Ex:** C *= A is equivalent to C = C * A |
| /= | Divide value then assign | It divides the left operand with the right operand and assigns the result to the left operand.<br>**Ex:** C /= A is equivalent to C = C / A |
| %= | Modulus value then assign | It takes modulus using two operands and assigns the result to the left operand. |

| | | |
|---|---|---|
| | | **Ex:** C %= A is equivalent to C = C % A |

**4) Comparison Operators:** *Comparison operators,* are used to compare two values. The result of the comparison is either true or false. Assume variable A holds 10 and variable B holds 20

| Operator | Description | Example |
|---|---|---|
| == | Equivalency | Checks if the value of two operands are equal or not, if yes, then the condition becomes true.<br>Ex: (A == B) is not true. |
| != | Not equivalent | Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true.<br>Ex: (A != B) is true. |
| > | 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.<br>Ex: (A > B) is not true. |
| < | 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.<br>Ex: (A < B) is true. |
| >= | 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.<br>Ex: (A >= B) is not true. |
| <= | 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.<br>Ex: (A <= B) is true. |

**5) Conditional Operators:**

The conditional operator tells the browser to take a specific action after evaluating an expression. The conditional operator has three parts: The first part is a logical expression, which you'll recall is an expression that evaluates to either true or false. The second part is the action the browser must take if the expression is true. The third part is the action the browser must take if the expression is false. The first and second parts of the conditional operator are separated by a question mark (?). The second part and the third parts are separated by a colon (:).

| Operators | Description |
|---|---|
| Expression? value1 : value2 | If expression is true, then use value1;<br>otherwise, use value2 |
| delete | Delete the property of an object. |
| new | Create an object. |
| typeof | Returns the type of an object. |

**6) Bitwise Operators:**

Bitwise operators perform their operations on such binary representations, but they return standard JavaScript numerical values.

| Operator | Name | Description |
|---|---|---|
| & | AND | Sets each bit to 1 if both bits are 1 |
| \| | OR | Sets each bit to 1 if one of two bits is 1 |
| ^ | XOR | Sets each bit to 1 if only one of two bits is 1 |
| ~ | NOT | Inverts all the bits |
| << | Zero filled left shift | Shifts left by pushing zeros in from the right and let the leftmost bits fall off |
| >> | Signed right shift | Shifts right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off |
| >>> | Zero fill right shift | Shifts right by pushing zeros in from the left, and let the rightmost bits fall off |

**Example:**

| Operation | Result | Same as | Result |
|---|---|---|---|
| 5 & 1 | 1 | 0101 & 0001 | 0001 |
| 5 \| 1 | 5 | 0101 \| 0001 | 0101 |
| ~5 | 10 | ~0101 | 1010 |
| 5 << 1 | 10 | 0101 << 1 | 1010 |
| 5 ^ 1 | 4 | 0101 ^ 0001 | 0100 |
| 5 >> 1 | 2 | 0101 >> 1 | 0010 |
| 5 >>> 1 | 2 | 0101 >>> 1 | 0010 |

**Alert Dialog Box:** An alert dialog box pops on the screen to display a message and stays on the screen until someone clicks the OK button that appears in the dialog box. You display an alert dialog box by calling the alert function and passing it the text that you want to be displayed.

**Example:**

```
<html>
<body>
<script type="text/javascript">
var a = 33;
var b = 10;
var c = "Test";
result = a + b;
alert(result);

result = a - b;
alert(result);

result = a * b;
alert(result);
```

```
result = a / b;
alert(result);

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

**Student Activity:**

1. List the application of JavaScript.
2. List the features of JavaScript.
3. How to create an object in JavaScript?
4. Explain variable in javascript.
5. Differentiate between prompt () and alert () methods.
6. Write a program to execute the conditional operator(ternary) in JavaScript.
7. Write a program to perform arithmetic operations by using
   a) prompt()  //accepts the inputs from user
   b) variable assignment  // var a=10;  var b= 20;
   c) function definition
8. State the use of property getter and setter in JavaScript.

| Marks Obtained | | | Dated Signature of Teacher |
|---|---|---|---|
| **Process Related (15)** | **Product Related (35)** | **Total (50)** | |
| | | | |

https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_state_forin

https://www.tutorialspoint.com/javascript/javascript_forin_loop.htm

https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_nav_platform

https://www.w3schools.com/jsref/prop_nav_platform.asp

## Experiment No: 02

**Develop a JavaScript to use decision making and looping statements.**

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | 1)Notepad editor<br>2)Any Web Browser |

**Practical Significance:**

Generally, instructions are executed sequentially. In some cases, it is necessary to change the sequence of executions based on certain conditions. For this purpose, decision control structure is required.

**Theoretical Background:**

**a) if statement**

The if statement is one of the most powerful statements that you'll use in JavaScript, because it enables you to have the browser execute some statements only if certain conditions are met while your JavaScript is running.

**Syntax:**

```
if (conditional expression)
{
//This is where the code block appears.
}
```

**b) if-else Statement:**

If...else statement first checks the condition. If condition is true, then true statement block is executed. If condition is false, then false statement block is executed.

**Syntax:**

```
if (condition)
{
   // if TRUE then execute this code
}
else
{
   // if FALSE then execute this code
}
```

**if…else if Statement**

The if...else if statement tells the browser, "If the condition is true, then execute these statements, else evaluate another condition. If the other condition is true, then execute these other statements."

**Syntax:**

```
if (conditional expression)
{ //Place statements here.
}
else if (conditional expression)
{ //Place statements here.  }
```

**switch…case Statement**

A switch...case statement tells the browser to compare a switch value with a series of case values. If the switch value matches a case value, then the browser executes statements that are placed beneath the case value.

**Syntax:**

```
switch (value)
{
case value1:
//Place statements here.
break;
case value2:
//Place statements here.
break;
default:
//Place statements here.
}
```

**Loop Statement**

A loop is used to execute one or more statements repeatedly, without your having to duplicate those statements in your JavaScript. You can use four types of loops in a JavaScript: a for loop, for in loop, while loop, and do...while loop.

**1. For Loop:** The for loop tells the browser to execute statements within the for loop until a condition statement returns false. The browser then continues by executing the statement or statements below the for loop until the test condition is false.

**syntax:**

```
for ( initializer; conditional expression ; post loop statements)
{
//Place statements here.
}
```

**For…in Loop:**

The for...in statement iterates over the enumerable properties of an object. For each distinct property, statements can be executed.

**Syntax:**

```
For (variable in object)
{
statement or block to execute
…
}
```

**While Loop:**

The while loop tells the browser to execute one or more statements continually as long as a condition defined in the while loop is true.

**Syntax:**

```
while (conditional expression)
{
//Place statements here.
}
```

**do…while Loop:**

The do...while loop operates similarly to the while loop, except that statements within the code block execute at least once, because the browser doesn't evaluate the conditional expression condition until the end of the code block.

**Syntax:**

```
do
{
// statements.
} while (conditional expression)
```

**Program Code: Write a program to print the greatest number among two.**

```
 <html>
<head>
<title>Practical 2</title>
</head>
<body>
<h1>If statement</h1>
<script>
var a=10;
var b=20;
```

```
document.write("a="+a+"<br>");
document.write("b="+b+"<br>");
if(a > b){
document.write("a is greater than b");
}
if(a < b){
document.write("a is smaller than b");
}
</script>
</title>
</html>
```

**Output:**

a is smaller than b

**Program Code: Write a program to demonstrate the use of Switch statement.**

```
<html>
<body>
<script>
var day;
switch (new Date().getDay())
{
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
break;
case 3:
    day = "Wednesday";
    break;
case 4:
   day = "Thursday";
   break;
 case 5:
   day = "Friday";
   break;
 case  6:
 day = "Saturday";
          }
          document.write("Today is " + day);
```

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

**Practical related questions:**

1. Explain For…in loop ?
2. Difference between if…else and switch statement.
3. Difference between while and do while.

**Exercise:**

1. Write a JavaScript program to print the factorial of a given number.
2. Write a JavaScript program to print the Fibonacci series till 10 number.
3. Write a code to find out whether the year is leap year or not.

**Experiment No: 03**

**Develop a JavaScript to implement the array functionality.**

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | Notepad Editor<br>Any web browser |

**Practical Significance:**

An array is very similar to a variable in that an array tells the browser to reserve a place in memory that can be used to store information. An array can comprise one or multiple elements. Each element is like a variable in that an array element refers to a memory location where information can be temporarily stored.

**Theoretical Background:**

An array is very similar to a variable in that an array tells the browser to reserve a place in memory that can be used to store information. An array can comprise one or multiple elements. Each element is like a variable in that an array element refers to a memory location where information can be temporarily stored. An array is identified by a unique name, similar to the name of a variable. A number called an *index* identify es an array element. The combination of the array name and an index is nearly the same as a variable name. In your JavaScript, you use both the array name and the index to specify a particular memory location.

**Declaring an Array**

This declaration statement has five parts: the first part is the var syntax; the second part is the array name, which you create; the third part is the assignment operator; the fourth part is the new operator;
and the fifth part is the Array() constructor. All these parts are shown here:

```
var products = new Array();
```

**Initializing an Array**

Initialization is the process of assigning a value when either a variable or an array is declared. When initializing an array, you place the value within the parentheses of the Array() constructor.

```
var products = new Array('aaa')'
```

In the real world, an array usually has more than one array element, with each element having its own value. Therefore, you'll fi nd yourself having to initialize the array with more than one value.

**How Many Elements Are in the Array?**

The length property of the array object contains the number of elements contained in the array.

var len = products.length;

**Practical related questions:**

1. How to sort the element in the array?
2. Explain property of an array.

3.What is the use of slice() method in array?
4.Diferentiate between join() and concat() methods?
5.Write the methods for following operations on array?
i)remove the first element from the array
ii)add the new element at the end of the array?
iii)remove the element from the bottom of the array?
iv)create new array using elements of another array?

**Exercise:**

1. Write the output for following:

```
<script>
var car= new Array();
car.push("Mercedes");
car.push("Ferrari");
car.push("Lamborghini");
car.push("BMW");
car.push("Bugatti");
for(var i=0; i<car.length; i++){
document.write(car[i]+"<br>");
}
</script>
```

```
<script>
var car= new Array();
car.push("Mercedes");
car.push("Ferrari");
car.push("Lamborghini");
car.push("BMW");
car.sort();
for(var i=0; i<car.length; i++){
document.write(car[i]+"<br>");
}
</script>
```

2. Create an array for following data and display data.

| Name | Sub1 | Sub2 | Sub3 |
|---|---|---|---|
| aaa | 45 | 26 | 23 |
| bbb | 46 | 49 | 50 |
| ccc | 24 | 50 | 48 |

**Experiment No: 04**

Develop a JavaScript to implement functions.

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | Notepad Editor<br>Any web browser |

**Practical Significance:**

A function as part of your JavaScript that has a name and contains one or more statements. You name the function and write the statement(s) that are contained within the function. You then use the name of the function elsewhere in your JavaScript whenever you want the browser to execute those statements. A function can be called from anywhere in the JavaScript fi le or in the HTML document.

**Theoretical Background:**

**Defining a Function**

A function must be defined before it can be called in a JavaScript statement. The best place to defi ne a function is at the beginning of a JavaScript that is inserted in the <head> tag, because then all subsequent JavaScripts on the web page will know the definition of that function.

A function definition consists of four parts: the name, parentheses, a code block, and an optional return keyword.

**Writing a Function Definition**

It is called Salary() and tells the browser the steps that are necessary to give you a raise in pay (at least on
paper). This function contains all the values needed to calculate your new salary; therefore, no argument is needed:

```
function Salary()
{
var salary = 50000 * 1.5
alert("Your new salary is " + salary)
}
```

**Adding Arguments**

An *argument* is one or more variables that are declared within the parentheses of a function definition.

```
function Salary(OldSalary)
{
var Salary = OldSalary * 1.25
alert("Your new salary is " + Salary)
}
```

**Adding Multiple Arguments**

You can use as many arguments as necessary for the function to carry out its task.
Each argument must have a unique name, and each argument within the parentheses must be separated by a comma.

```
function Salary(OldSalary, PercIncrease)
{
var Salary = OldSalary * (1 + (PercIncrease / 100))
alert("Your new salary is " + NewSalary)
}
```

**Practical related questions:**

1. True or False. A comma must separate arguments in a function definition.
a. True  b. False

2. A code block is used in a
a. Function call  b. Function definition c. Return value  d. Argument

3. True or False. A function can be called by HTML code in a web page.
a. True  b. False

4.A variable is out of scope when:
a. The statement that calls a function ignores the value returned by the function
b. The variable cannot be accessed by a statement
c. A variable isn't defi ned in a function
d. A variable is passed to a function

5.A local variable can be accessed,
a. Only by functions defined within the JavaScript
b. Only outside of a function
c. Only by the function that defined it
d. From anywhere in the JavaScript

**Exercise:**

Calculate the factorial of a number by
1)calling function without argument
2)calling function with argument.

**Experiment No: 05**

**Develop a JavaScript to implement strings.**

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | Notepad Editor<br>Any web browser |

**Theoretical Background:**

The **JavaScript string** is an object that represents a sequence of characters.
There are 2 ways to create string in JavaScript

1. By string literal
2. By string object (using new keyword)

**1) By string literal**

The string literal is created using double quotes. The syntax of creating string using string literal is given below:

var stringname="string value";

**2) By string object (using new keyword)**

The syntax of creating string object using new keyword is given below:

var stringname=new String("string literal");

**JavaScript String Methods:**

| Methods | Description |
|---|---|
| charAt() | It provides the char value present at the specified index. |
| charCodeAt() | It provides the Unicode value of a character present at the specified |
| concat() | It provides a combination of two or more strings. |
| indexOf() | It provides the position of a char value present in the given string. |

| | |
|---|---|
| lastIndexOf() | It provides the position of a char value present in the given string by searching a character from the last position. |
| search() | It searches a specified regular expression in a given string and returns its position if a match occurs. |
| match() | It searches a specified regular expression in a given string and returns that regular expression if a match occurs. |
| replace() | It replaces a given string with the specified replacement. |
| substr() | It is used to fetch the part of the given string on the basis of the specified starting position and length. |
| substring() | It is used to fetch the part of the given string on the basis of the specified index. |
| slice() | It is used to fetch the part of the given string. It allows us to assign positive as well negative index. |
| toLowerCase() | It converts the given string into lowercase letter. |
| toLocaleLowerCase() | It converts the given string into lowercase letter on the basis of host?s current locale. |
| toUpperCase() | It converts the given string into uppercase letter. |
| toLocaleUpperCase() | It converts the given string into uppercase letter on the basis of host?s current locale. |
| toString() | It provides a string representing the particular object. |
| valueOf() | It provides the primitive value of string object. |
| split() | It splits a string into substring array, then returns that newly created array. |
| trim() | It trims the white space from the left and right side of the string. |

**Practical related questions:**

1. How to remove the white spaces between the strings?
2. How to convert text in lowercase into title case?

**Exercise:**

1.Write a program to check whether the given staring is palindrome or not?
2.Write a program to count the number of vowels into the string?

**Experiment No: 06**

**Create a webpage using form elements.**

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | 1)Notepad editor<br>2)Any Web Browser |

**Practical Significance:**

It seems that no matter what web site you visit these days, you are asked to fill out a form—be it an order form, subscription form, membership form, financial form, survey, and the list goes on. Although forms may seem invasive, prying into our private affairs, forms are the only practical way to collect information that is necessary to conduct business on the Internet.

**Theoretical Background:**

Forms are created using HTML form elements such as buttons and check boxes. Forms used by commercial web sites also interact by using JavaScript. A JavaScript is used for a variety of purposes, including data validation and for dynamically interacting with elements of a form. A *form* is a section of an HTML document that contains *elements* such as radio buttons, text boxes, check boxes, and option lists. HTML form elements are also known as *controls*. Elements are used as an efficient way for a user to enter information into a form. Forms are used for all kinds of purposes. In a business, forms are used to gather order information from a customer. Forms are also used for online surveys. Teachers use forms for online tests. Information entered into a form is sent to the web server for processing when the user clicks a submit button. The program that processes the form is called a *Common Gateway Interface (CGI)* program.

Many applications require that some information contained on a form be verified using a validation process. Two common ways to validate information on a form are by using CGI programs and Javascript. A CGI program validates information after the form is submitted. A JavaScript can validate information whenever one of several events occurs while the form is displayed on the screen Validation should occur on both the client (via JavaScript) and the server (via a CGI program). The client-side validation provides immediate feedback and reduces load on the server. It's good practice to validate again on the server because you don't always know that the JavaScript executed properly on the browser.

**Responding to Form Events**

A JavaScript executes in response to an event that occurs while a form is displayed on the screen. An event is something the user does to the form, such as clicking a button, selecting a radio button, or moving the cursor away from an element on the form. The browser also fi res

events when the page finishes loading from the server. You can execute a script each time one of the form events listed in Table:

| **Event** | **Description** |
|---|---|
| onload | Executes when the browser finishes loading a window or all frames within a frameset |
| onunload | Executes when the browser removes a document from a window or frame |
| onclick | Executes when the mouse button is clicked over an element |
| ondblclick | Executes when the mouse button is double-clicked over an element |
| onmousedown | Executes when the mouse button is clicked while the mouse cursor is over an element |
| onmouseup | Executes when the mouse button is released while the mouse cursor is over an element |
| onmouseover | Executes when the mouse cursor moves onto an element |
| onmousemove | Executes when the mouse cursor is moved while over an element |
| onmouseout | Executes when the mouse cursor is moved away from an element |
| onfocus | Executes when an element receives focus |
| onblur | Executes when an element loses focus |
| onkeypress | Executes when a key is pressed and released |
| onkeydown | Executes when a key is held down |
| onkeyup | Executes when a key is released |
| onsubmit | Executes when a form is submitted |
| onreset | Executes when a form is reset |
| onselect | Executes when text is selected in a text field |
| onchange | Executes when an element loses input focus and the value of the element has changed since gaining focus |

An event is associated with an element of a form as a attribute defined within the opening tag of the element. You assign this attribute the name of the JavaScript function that you want executed when the event occurs.

**Program:**

```
<html>
<head>
<title>Registration Form</title>
</head>
<body>
<form name="MyForm" method="post" action="">
<h2>Register your account</h2>
Enter your Firstname: <input type="text" name="FName" id="FName"><br><br>
Enter Your Lastname: <input type="text" name="LName" id="LName"><br><br>
Enter your Username: <input type="text" name="UName" id="UName"><br><br>
Enter your Password: <input type="password" name="pass" id="pass"><br><br>
```

```
Enter you Gender: <input type="radio" name="Gender" id="Male">Male
<input type="radio" name="Gender" id="Female">Female<br><br>
Your Hobbies? <input type="checkbox" name="Hobbies" id="Cricket">Cricket
<input type="checkbox" name="Football" id="Cricket">Football<br><br>
Select your city: <select name="City">
<option name="Mumbai">Mumbai</option>
<option name="Pune">Pune</option>
<option name="Nagpur">Nagpur</option>
<option name="Ahmedabad">Ahmedabad</option>
</select><br><br>
<input type="submit"> <input type="reset">
</form>
</body>
</html>
```

**Exercise:**

1)Write a program to create the registration form for creating gmail account.

2)Write a JavaScript program for evaluating checkbox selection.

3)Write a JavaScript program for Changing Attribute Values Dynamically.

**Experiment No: 07**

Create a webpage to implement form events. Part 1

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | 1)Notepad editor<br>2)Any Web Browser |

**Practical Significance:**

A JavaScript executes in response to an event that occurs while a form is displayed on the screen. An event is something the user does to the form, such as clicking a button, selecting a radio button, or moving the cursor away from an element on the form. The browser also fires events when the page finishes loading from the server.

**Theoretical Background**

**onclick Event Type**

This is the most frequently used event type, which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.

```
<html>
  <head>
    <script>
        function sayHello() {
          alert("Hello World")
        }
    </script>
  </head>
  <body>
    <p>Click the following button and see result</p>
    <form>
      <input type="button" onclick="sayHello()" value="Say Hello" />
    </form>
  </body>
</html>
```

**onmouseover and onmouseout Event**

These two event types will help you create nice effects with images or even with text as well. The onmouseover event triggers when you bring your mouse over any element and the onmouseout triggers when you move your mouse out from that element.

```
<html>
  <head>
    <script>
        function over() {
          document.write ("Mouse Over");
        }
        function out() {
          document.write ("Mouse Out");
        }
    </script>
  </head>
  <body>
    <p>Bring your mouse inside the division to see the result:</p>
    <div onmouseover="over()" onmouseout="out()">
      <h2> This is inside the division </h2>
    </div>
  </body>
</html>
```

**Practical Example**

```
 <html>
<head>
<title>Mouse Events</title>
</head>
<body>
<h2>Following are the Mouse Events</h2>
<input type="button" name="btn1" onmouseover="alert('You pointed your mouse on the
Button');" value="Hover mouse on me"><br><br>
<input type="button" name="btn2" onmouseout="alert('You pointed your mouse out of the
Button');" value="Hover out of me"><br><br>
<input type="button" name="btn3" onclick="alert('You left-clicked on the Button');"
value="Left-click on me"><br><br>
<input type="button" name="btn4" oncontextmenu="alert('You right-clicked on the
Button');" value="Right-click on me"><br><br>
</body>
</html>
```

**Practical related questions:**

1) Explain Java Intrinsic functions.
2) Write the events which are used checkbox and buttons.
3) Explain the use of with keyword in javascript.
4) What is the use of oncontextmenu event?

**Exercise:**

1) Write a program for changing the option list dynamically.
2) Develop a program for as we enter the firstname and lastname , email is automatically generated.

**Experiment No: 08**

Create a webpage to implement form events. Part 2

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | 1)Notepad editor<br>2)Any Web Browser |

**Practical Significance:**

A JavaScript executes in response to an event that occurs while a form is displayed on the screen. An event is something the user does to the form, such as clicking a button, selecting a radio button, or moving the cursor away from an element on the form. The browser also fires events when the page finishes loading from the server.

**Theoretical Background**

When you interact with the keyboard, the keyboard events are fired. There are three main keyboard events:

keydown – fires when you press a key on the keyboard and it fires repeatedly while you holding down the key.

keyup – fires when you release a key on the keyboard.

keypress – fires when you press a character keyboard like a,b, or c, not the left arrow key, home, or end keyboard, … The keypress also fires repeatedly while you hold down the key on the keyboard.

The keyboard events typically fire on the text box, through all elements support them.

When you press a character key once on the keyboard, three keyboard events are fired in the following order:

keydown

keypress

keyup

Both keydown and keypress events are fired before any change made to the text box, whereas the keyup event fires after the changes have made to the text box. If you hold down a character key, the keydown and keypress are fired repeatedly until you release the key.

When you press a non-character key, the keydown event is fired first followed by the keyup event. If you hold down the non-character key, the keydown is fired repeatedly until you release the key.

**Handling keyboard events:**

To handle a keyboard event, you follow these steps:

First, select the element on which the keyboard event will fire. Typically, it is a text box.

Then, use the element.addEventListener() to register an event handler.

Suppose that you have the following text box with the id message:

```
<input type="text" id="message">
```

The following illustrates how to register keyboard event listeners:

```
msg.addEventListener("keydown", (event) => {
    // handle keydown
});
msg.addEventListener("keypress", (event) => {
    // handle keypress
});

msg.addEventListener("keyup", (event) => {
    // handle keyup
});
```

If you press a character key, all three event handlers will be called.

**Example:**

```
<html>
  <head><title>Javascript Form Events : onClick Event</title>
      <script>
    function Japan(){
      alert("konnichiwa");
```

```
    }
      function India(){
      alert("namaste");
    }
        function Germany(){
      alert("Guten Tag");
    }
  </script>
  </head>
<body>
  <p>Hello in Different Countries</p>
  <form>
    <input type="button" value="Japan"
    onclick ="Japan()" />

    <input type="button" value="India"
    onclick ="India()" />

    <input type="button" value="Germany"
    onclick ="Germany()" />
  </form>

</body>
</html>
```

**Practice Questions**

1) What event occurs when an element comes into focus?

a. onblur

b. onfocus

c. onselect

d. onchange

2) What event occurs when a user highlights text in a text fi eld?

a. onblur

b. onfocus

c. onselect

d. onchange

3)True or False. A JavaScript function can only change attributes of an element that calls the JavaScript function.

a. True

b. False

**Exercise:**

1)Write a program to demonstrate the use of onchange event.

2)Write a JavaScript program to demonstrate the addEventListener ().

**Experiment No: 09**

Develop a webpage using Intrinsic java functions.

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | 1)Notepad editor<br>2)Any Web Browser |

**Practical Significance:**
A JavaScript has a special set of functions called intrinsic functions that mimic actions of the submit button and reset button of a form.

**Theoretical Background**

The HTML <input> src Attribute is used to *specify the URL of the image to be used as a submit Button*. This attribute is not used with <input type="image">

Syntax:

```
<input src="URL">
```

Attribute Values: It contains a single value URL which specifies the link of source image. There are two types of URL link which are listed below:

- Absolute URL: It points to another webpage.
- Relative URL: It points to other files of the same web page.

Following example, we have used one <img> tag to simulate the functionality of submit button. Before writing the code make sure one "submit.jpg" picture should save in your folder.

**Program:**

```
<form action=" ">
  <label for="fname">Institute Name:</label>
  <input type="text" id="name" name="name"><br><br>
  <input type="image" src="submit.jpg" alt="Submit" width="130" height="48" onclick="myFunction()">
</form>
<p id="demo"></p>
<script>
  function myFunction()
```

```
        {
            var x = document.getElementById("name").value;
            document.write("You Submitted:<h2>" +x +"</h2>");
    }
</script>
```

**Exercise:**

1)Write a program to disable and enabled text field.

2)Write a JavaScript program to change the value of an element that the user cannot change (a read-only element)

## Experiment No: 10

Develop a webpage for creating session and persistent cookies. Observe the effects with Browser cookie settings.

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | 1)Notepad editor<br>2)Any Web Browser |

**Practical Significance:**

A *cookie* is a small piece of information that a web site writes to your hard disk when you visit the site.

**Theoretical Background**

A JavaScript can be used to create cookies whenever someone visits the web page that contains the script. A JavaScript can also be used to read cookies stored on a user's computer, and it uses the information stored in a cookie to personalize the web page that a user visit.

Cookies are usually small text files, given ID tags that are stored on your computer browser directory or program data subfolders.

There are two types of cookies namely: session cookie and persistent cookie.

**1. Session cookie** are created temporarily in your browser's subfolder while you are visiting a website. Once you leave the site, the session cookie is deleted.

**2.Persistant Cookie** files remain in your browser's subfolder and are activated again once you visited the website that created that particular cookie.

### 10.1 Creating a Cookie:

The text of a cookie must contain a *name-value pair*, which is a name and value of the information. When you write your JavaScript, you decide on the name and the value.

You cannot include semicolons, commas, or white space in the name or the value unless you precede these characters with the escape character (\). The escape character tells the browser that the semicolon, comma, or white space is part of the name or value and not a special character.

You simply assign the cookie to the window.document.cookie object. The browser automatically writes the cookie to memory when it reads this assignment statement in your JavaScript, unless you set an expiration date for the cookie, which then causes the cookie to be written to the computer's hard disk.

Every cookie has four parts: a name, an assignment operator, a value, and a semicolon.

The semicolon is a delimiter and not part of the value. A delimiter is a character that indicates where something ends, which in this case is the end of the cookie.
This statement creates a cookie, where CustomerName is the name and ABC is the value:

window.document.cookie = "CustomerName= ABC;"

**Program:**

```
<html>
<head>
<title>Write Cookie</title>
<script>
function WriteCookie()
{
with (document.CookieWriter)
{
document.cookie =
"CustomerName="+ customer.value+";"
alert("Cookie Written")
}
}
</script>
</head>
<body>
<form name="CookieWriter" action="" >
Enter your name:
<input type="text" name="customer"
onchange="WriteCookie()"/>
</FORM>
</body>
</html>
```

### 10.2 Setting the Expiration Date

You can extend the life of a cookie beyond the current browser session by setting an expiration date and saving the expiration date within the cookie. The expiration date is typically an increment of the current date.
A date is stored in a variable of a Date data type.

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

```
<title>Write Cookie with Expiration Date</title>
<script>
function WriteCookie()
{
var expireDate = new Date
expireDate.setMonth(expireDate.getMonth()+3)
with (document.CookieWriter)
{
var CustomerName = customer.value
document.cookie = "CustomerName1="+
CustomerName+";expires="+expireDate.toGMTString()
alert("Cookie Written")
}
}
</script>
</head>
<body>
<form name="CookieWriter" action="" >
Enter your name: <input type="text"
name="customer" onchange="WriteCookie()" />
</FORM>
</body>
</html>
```

**Exercise:**

1) Write a program to read the cookie.
2)Write a program to delete the cookie.

**Experiment No: 11**

**Develop a Web Page for placing the window on the screen and working with child window.**

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | 1)Notepad editor<br>2)Any Web Browser |

**Practical Significance:**

**Window object**

- Window object is a top-level object in Client-Side JavaScript.
- Window object represents the browser's window.
- It represents an open window in a browser.
- It supports all browsers.
- The document object is a property of the window object. So, typing window.document.write is same as document.write.
- All global variables are properties and functions are methods of the window object.

**Theoretical Background**

**Window Object Properties**

| Property | Description |
|---|---|
| Document | It returns the document object for the window (DOM). |
| Frames | It returns an array of all the frames including iframes in the current window. |
| Closed | It returns the Boolean value indicating whether a window has been closed or not. |
| History | It returns the history object for the window. |
| innerHeight | It sets or returns the inner height of a window's content area. |
| innerWidth | It sets or returns the inner width of a window's content area. |
| Length | It returns the number of frames in a window. |
| Location | It returns the location object for the window. |
| Name | It sets or returns the name of a window. |
| Navigator | It returns the navigator object for the window. |

| | |
|---|---|
| Opener | It returns a reference to the window that created the window. |
| outerHeight | It sets or returns the outer height of a window, including toolbars/scrollbars. |
| outerWidth | It sets or returns the outer width of a window, including toolbars/scrollbars. |
| Parent | It returns the parent window of the current window. |
| Screen | It returns the screen object for the window. |
| screenX | It returns the X coordinate of the window relative to the screen. |
| screenY | It returns the Y coordinate of the window relative to the screen. |
| Self | It returns the current window. |
| Status | It sets the text in the status bar of a window. |
| Top | It returns the topmost browser window that contains frames. |

**Window Object Method**

| Method | Description |
|---|---|
| alert() | It displays an alert box. |
| confirm() | It displays a dialog box. |
| prompt() | It displays a dialog box that prompts the visitor for input. |
| setInterval() | It calls a function or evaluates an expression at specified intervals. |
| setTimeout() | It evaluates an expression after a specified number of milliseconds. |
| clearInterval() | It clears a timer specified by setInterval(). |
| clearTimeOut() | It clears a timer specified by setTimeout(). |
| close() | It closes the current window. |
| open() | It opens a new browser window. |
| createPopup() | It creates a pop-up window. |
| focus() | It sets focus to the current window. |
| blur() | It removes focus from the current window. |
| moveBy() | It moves a window relative to its current position. |
| moveTo() | It moves a window to the specified position. |
| resizeBy() | It resizes the window by the specified pixels. |
| resizeTo() | It resizes the window to the specified width and height. |
| print() | It prints the content of the current window. |
| scrollBy() | It scrolls the content by the specified number of pixels. |

| | |
|---|---|
| scrollTo() | It scrolls the content to the specified coordinates. |

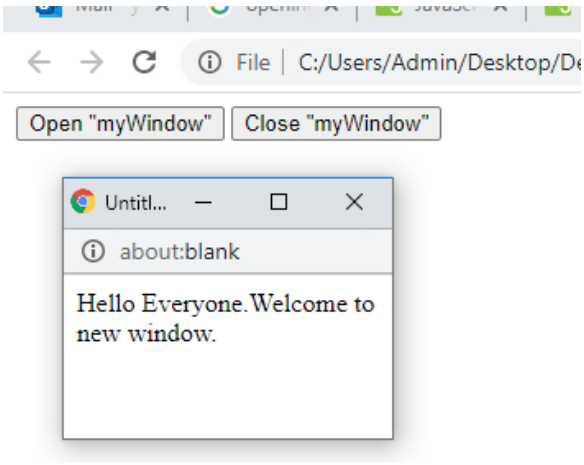
**Program:** Open a new window and close that window on button click event using open( ) and close ( ).

```
<html>
<body>

<button onclick="openWin()">Open "myWindow"</button>
<button onclick="closeWin()">Close "myWindow"</button>

<script>
var myWindow;

function openWin()
{
  myWindow = window.open("", "myWindow", "width=200,height=100");
  myWindow.document.write("<p>Hello Everyone.Welcome to new window.</p>");
}

function closeWin()
{
  myWindow.close();
}
</script>
</body>
</html>
```

Output:

After clicking on “open myWindow” button, new window will be open.

Click on “Close myWindow” button, newly open window will be closed.




**Window position:**

A new window always displayed on the screen at the location which is specified by user and location is specified by setting the left and top properties of new window as shown below:

```
window.open("http://vpt.edu.in", "windowName", top=500,left=500,width=400,height=400");
```

The **innerWidth** property returns the width of a window's content area.
The **innerHeight** property returns the height of a window's content area.
Syntax:
window.innerWidth
window.innerHeight
The **outerWidth** property returns the outer width of the browser window, including all interface elements (like toolbars/scrollbars).
The **outerHeight** property returns the outer height of the browser window, including all interface elements (like toolbars/scrollbars).
Syntax:
window.outerWidth
window.outerHeight
These properties are read-only.
Code: To retrieve the dimensions of window:

```
<html>
<body>
<div id="demo"></div>
<script>
var txt = "";
txt += "<p>innerWidth: " + window.innerWidth + "</p>";
txt += "<p>innerHeight: " + window.innerHeight + "</p>";
txt += "<p>outerWidth: " + window.outerWidth + "</p>";
txt += "<p>outerHeight: " + window.outerHeight + "</p>";
document.getElementById("demo").innerHTML = txt;
</script>
</body> </html>
```

Output:

innerWidth: 606

innerHeight: 448

outerWidth: 1366

outerHeight: 728

Window provides two methods which also deal with positioning of windows named scrollBy() and moveTo() method.

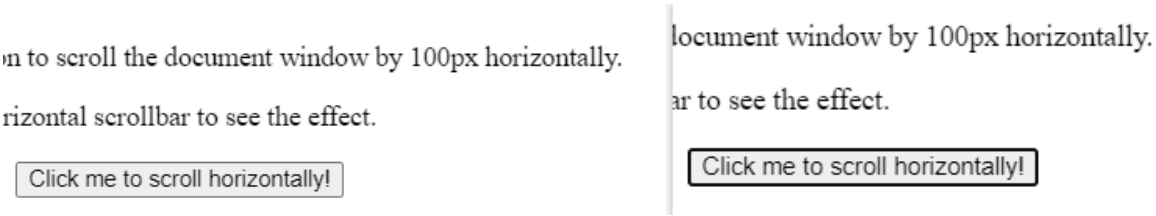
**scrollBy():** The scrollBy() method scrolls the document by the specified number of pixels.

Syntax: window.scrollBy(xnum, ynum)

Code: Scroll the document by 100px vertically:

```
<html>
<head>
<style>
body
{
  width: 5000px;
}
button
{
  position: fixed;
}
</style>
</head>
<body>
<p>Click the button to scroll the document window by 100px horizontally.</p>
<p>Look at the horizontal scrollbar to see the effect.</p>
<button onclick="scrollWin()">Click me to scroll horizontally!</button><br><br>
<script>
function scrollWin()
{
  window.scrollBy(100, 0);
}
</script>
</body>
</html>
```

Output:

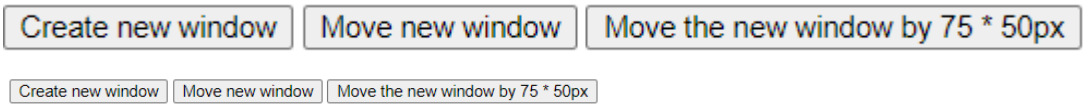


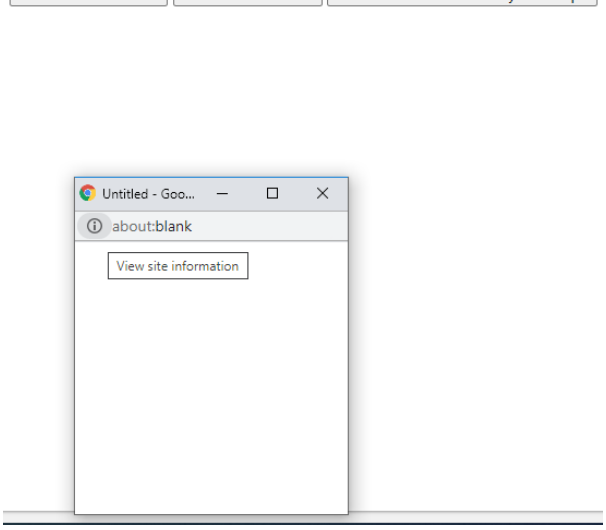

The **moveTo()** method moves a window's left and top edge to the specified coordinates.
Syntax: window.moveTo(*x, y*)
The **focus()** method is used to give focus to an element (if it can be focused).
Syntax: *HTMLElementObject*.focus()

Code: simple example of moveTo ( )

```
<html>
<body>
<button onclick="openWin()">Create new window</button>
<button onclick="moveWinTo()">Move new window</button>
<button onclick="moveWinBy()">
Move the new window by 75 * 50px
</button>
<script>
var myWindow;
function openWin()
{
  myWindow = window.open("", "", "width=250, height=250");
}
function moveWinTo()
{
  myWindow.moveTo(150, 150);
  myWindow.focus();
}
function moveWinBy()
{
  myWindow.moveBy(75, 50);
  myWindow.focus();
}
</script>
</body>
</html>
```

Output:




## Worrking with child windows:

Creation of multiple window is possible by creating and opening window in a loop using window.open( ) method.

The only thing needs to take care is to pass proper dimension for window so that newly created window will not appear one upon another.

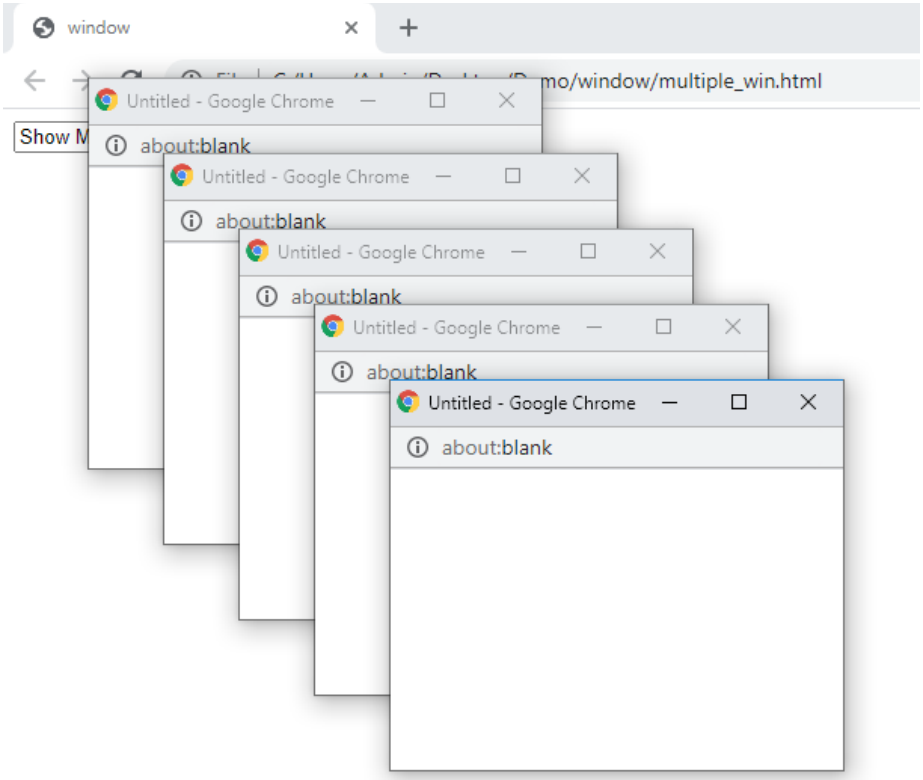
Code: In following example, x variable is assigned to top dimension of window and y variable is assigned to left dimension of window.

```
<html>
<head>
<title>window </title>
<script type="text/javascript">
function show()
{
for( i=0; i< 250 ; i=i+50)
{
var x=i+50;
var y=i+50;
winobj=window.open('','win' +i, 'top='+x+ ',left='+y+',width=300,height=200');
}
}
```

```
</script>
</head>
<body>
<input type="multiple" value="Show Multiple Windows" type="button" onclick="show()"/>
</body>
</html>
```

Output:

After clicking on “Show Multiple Windows” button 5 browsers will open.




**Exercise:**

1) Write a program to demonstrate the use of resizeBy() and resizeTo().
2)Write a program to demonstrate the use of scrollBy () and scrollTo().
3) Writing a number after a delay using setInterval ( ) method. In this example, numbers are displayed in a textarea after a 1 second.

**Experiment No: 12**

Develop a Web Page for validation of form fields using regular expressions.

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | 1)Notepad editor<br>2)Any Web Browser |

**Practical Significance:**

A **Regular Expression** is a sequence of characters that constructs a search pattern. When you search for data in a text, you can use this search pattern to describe what you are looking for.


**Theoretical Background**

A regular expression can be a **single character** or a more complicated pattern. It can be used for any type of text search and text replace operations. A Regex pattern consist of simple characters, such as /abc/, or a combination of simple and special characters, such as **/ab*c/** or **/example(d+).d*/.**

In JavaScript, a regular expression is an object that describes a pattern of characters.
The JavaScript **RegExp** class represents regular expressions, and both String and RegExp define methods. It uses regular expressions to perform **pattern-matching** and **search-and-replace** functions on text.

**Syntax:**

A regular expression is defined with the **RegExp ()** constructor as:

```
var pattern = new RegExp(pattern, attributes);
```

or simply

```
var pattern = /pattern/attributes;
```

Here,

- **Pattern –** A string that specifies the pattern of the regular expression or another regular expression.

**Attributes –** An optional string containing attributes that specify global, case-insensitive, and multi-line matches.

For example, you may want to make sure that users enter a valid email address correctly in a sign-up form, complete with @ sign and a domain, like .com, before sending their input to a server-side program for processing.
In the code example below, we'll check two simpler cases in a web form:

- a name input, which should contain only letters
- a zip (post) code input, which should contain 5 numbers

```
<html>
<body>
<form onsubmit="regEx(); return false">
Name:<br/>
<input type="text" /><br/> <br/>
Post/ZIP code (5 digits):<br/>
<input type="text" /><br/>
<br/>
<input type="submit" />
</form>
<script>
function regEx(){
function testName(){
var name = document.getElementsByTagName("input")[0].value;
var nameTest = name.search(/[a-zA-Z]/g);
if (nameTest == -1||name == ""){
alert("Please enter only letters in this field.");
}
}
function testZip(){
var zip = document.getElementsByTagName("input")[1].value;
var zipTest = zip.match(/\d/g);
if (zipTest.length != 5||zip == ""){
alert("Please enter 5 numbers in this field.");
}
}
testName();
testZip();
}
</script>
</body>
</html>
```

Exercise:

1) Write a code that accepts username and Aadhar card as input texts. When the user enters Aadhar-card number, the javascript validates card number and displays whether card number is valid or not.
2) Write a code to validate email-id.
3) Demonstrate the use of quantifiers.

**Experiment No: 13**

**Create a Web Page with Rollovers effect.**

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | 1)Notepad editor<br>2)Any Web Browser |

**Practical Significance:**

Rollover means a webpage changes when the user moves his or her mouse over an object on the page. It is often used in advertising. There are two ways to create rollover, using plain HTML or using a mixture of JavaScript and HTML.

**Theoretical Background:**

Though HTML can be used to create rollovers, it can only perform simple actions. If you wish to create more powerful rollovers, you need to use JavaScript. To create rollovers in JavaScript, we need to create a JavaScript function.

**Image Rollover:**

Code: Changing image using onmouseover and onmouseout (rollover and rollback)

```
<html>
<head>
<title>JavaScript Image Rollovers</title>
</head>
<body>
<img src="blue.png" name="image1" onmouseover="src='car.jpg' "
onmouseout="src='blue.png' ">
  </img>
</body>
</html>
```

**TextRollover:**

We can also create a rollover and rollback fo text using the **onmouseover** and **onmouseout.**

Code: Changing image (rollover and rollback)

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

```
text rollovers</title>
</head>
<body>
<table border="1" width="100%">
<tbody>
<tr valign="top">
<td width="50%">
<a><img height="500" src="blue.png" width="900" name="clr"></a></td>
<td><a onmouseover="document.clr.src='blue.png' ">
<b><u> Blue Color</u></b></a>
<br>
<a onmouseover="document.clr.src='red.png' ">
<b><u> Red Color</u></b></a>
<br>
<a onmouseover="document.clr.src='green.png' ">
<b><u> Green Color</u></b></a>
</td>
</tr>
</tbody>
</table>
</body>
</html>
```

Output:


**Exercise:**

1) Create a rollover effect that can change the color of its text.
2) Create a rollover effect that can change the image. (Set of 4 images)

**Experiment No: 14**

**Develop a Web Page for implementing Menus.**

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | 1)Notepad editor<br>2)Any Web Browser (prefer firefox) |

**Practical Significance:**
A menu consists of a set of options which are presented to the user. Menus are common in graphical user interface such as Windows or Mac OS.

**Theoretical Background:**

The Menu component provides the pull-down menu element that's common in most graphical user interfaces (GUIs). Using a familiar GUI element will reduce the interface learning curve of your web site or application for new users, as well as help all users more easily find what they're looking for. Having menus that contain links to sections at various levels in your web site can improve both the navigation of the site and the real estate of your web pages.

**There are various types menus are available in javascript:**

1) Pull-down(drop-down) menu
2) Floating (Fixed) menu
3) Chain Select menu
4) Tab menu
5) Popup menu
6) Sliding menu
7) Highlighted menu
8) Folding tree menu
9) Context menu
10) Scrollable menu-vertical and horizontal
11) Side Bar menu

To create a menu, usually require cascading style sheet , html and javascript.

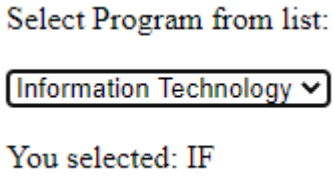
**Example:**

Following example provides four list elements as name of branches. When you select a branch from list, selected branch will be displayed as output.

```
<html>
```

```
<body>
<p>Select Program from list:</p>
<select id="mySelect" onchange="myFunction()">
  <option value="CO">Computer Engg</option>
  <option value="IF">Information Technology</option>
  <option value="EJ">Electronics and Tele</option>
  <option value="CE">Chemical Engg</option>
</select>
<p id="demo"></p>
<script>
function myFunction()
{
  var x = document.getElementById("mySelect").value;
  document.getElementById("demo").innerHTML = "You selected: " + x;
}
</script>
</body>
</html>
```

Output:




**Popup Menu:**

A popup menu appears as the user moves the mouse cursor over a parent menu item. The popup menu contains child menu items that are associated with the parent menu item.

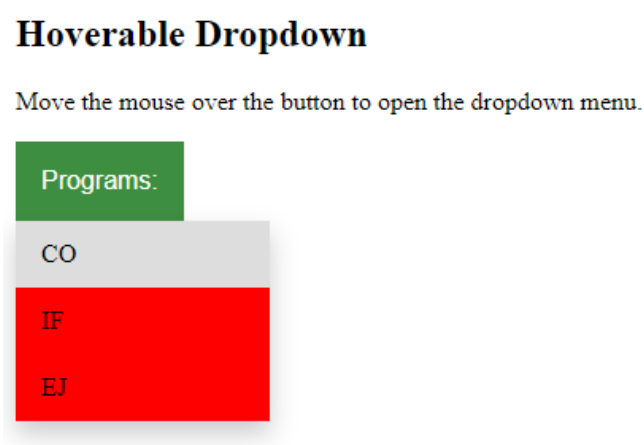
Code:

```
<html>
<head>
<style>
.dropbtn {
  background-color: Blue;
  color: white;
  padding: 16px;
  font-size: 16px;
  border: none;
```

```
}
.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: red;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  z-index: 1;
}

.dropdown-content a {
  color: black;
  padding: 12px 16px;
  text-decoration: none;
  display: block;
}

.dropdown-content a:hover {background-color: #ddd;}

.dropdown:hover .dropdown-content {display: block;}

.dropdown:hover .dropbtn {background-color: #3e8e41;}
</style>
</head>
<body>

<h2>Hoverable Dropdown</h2>
<p>Move the mouse over the button to open the dropdown menu.</p>

<div class="dropdown">
  <button class="dropbtn">Programs:</button>
  <div class="dropdown-content">
    <a href="#">CO</a>
```

```
    <a href="#">IF</a>
    <a href="#">EJ</a>
  </div>
</div>

</body>
</html>
```

Output:




**Exercise:**

1) Demonstrate the use of chain select menu.
2) Demonstrate the use of context menu.

**Experiment No: 15**

**Develop a Web Page for implementing Status Bar and Web Page Protection.**

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | 1)Notepad editor<br>2)Any Web Browser |

**Practical Significance:**
You need to secure your website, which means putting protection in place to keep out hackers, bugs, and other online nasties. Otherwise, your data could be at risk, your site could crash.

**Theoretical Background:**

**Status bar:**

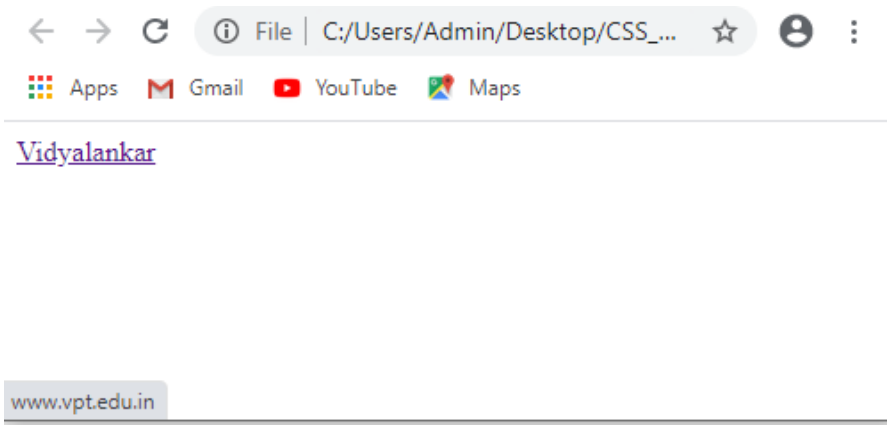
The **status** property of the Window interface was originally intended to set the text in the status bar at the bottom of the browser window. However, the HTML standard now requires setting window.status to have no effect on the text displayed in the status bar.

Syntax:

```
window.status = string;
var value = window.status;
```

```
 <html>
<head>
<title>JavaScript Status Bar</title></head>
<body>
  <a href="http://www.vpt.edu.in"
   onMouseOver="window.status='Vidyalankar';return true"
   onMouseOut="window.status='';return true">
Vidyalankar
  </a>
</body>
</html>
```

Output:




**Note:** "window.status" does not supported any browser.

**Protecting web pages:**

There is nothing secret about your web page. Anyone with a little computer knowledge can use a few mouse clicks to display your HTML code, including your JavaScript, on the screen. In this, you'll learn how to hide your JavaScript and make it difficult for malicious hackers to extract e-mail addresses from your web page.
The following example shows you how to disable the visitor's right mouse button while the browser displays your web page. All the action occurs in the JavaScript that is defined in the <head> tag of the web page.

```
<html>
<head>
<script>
window.onload = function()
{
document.addEventListener("contextmenu", function(e)
{
e.preventDefault();
}, false);}
</script>
<body>
<h3>Right click on screen,Context Menu is disabled</h3>
</body>
</html>
```

Exercise:

1) Demonstrate the use of status bar.
2) How to Concealing Your E-mail Address?
3) Why should protect our webpages?

**Experiment No: 16**

**Develop a Web Page for implementing Slideshow, Banner.**

**Resources required:**

| Hardware | Software |
|---|---|
| Computer System | 1)Notepad editor<br>2)Any Web Browser |

**Practical Significance:**

The banner advertisement is the hallmark of every commercial web page. It is typically positioned near the top of the web page, and its purpose is to get the visitor's attention by doing all sorts of clever things.

A slideshow is similar in concept to a banner advertisement in that a slideshow rotates multiple images on the web page. However, unlike a banner advertisement, a slideshow gives the visitor the ability to change the image that's displayed: the visitor can click the Forward button to see the next image and the Back button.

**Theoretical Background:**

The JavaScript must do the following:

1. Load banner advertisements into an array.
2. Determine whether the browser supports the image object.
3. Display a banner advertisement.
4. Pause before displaying the next banner advertisement.

You load the banner advertisements into an array by declaring an Array() object and initializing it with the file name of each banner advertisement. For example, suppose you have three banner advertisements that are contained in the 1.jpg, 2.jpg, and 3.jpg files. Here's how you'd load them into an Array() object:

Banners = new Array('1.jpg','2.jpg','3.jpg')

Example:

```
<html >
<head>
<title>Banner Ads</title>
<script>

Banners = new Array('1.jpg','2.jpg','3.jpg');
CurrentBanner = 0;
function DisplayBanners()
```

```
{
if (document.images);
 {
CurrentBanner++;
if (CurrentBanner == Banners.length)
{
CurrentBanner = 0;
}
document.RotateBanner.src= Banners[CurrentBanner];
setTimeout("DisplayBanners()",1000);
}
}

</script>
</head>
<body onload="DisplayBanners()" >
<center>
<img src="1.jpg" width="400"
height="75" name="RotateBanner" />
</center>
</body>
</html>
```

**Creating a slideshow:**

First, set the **slideIndex** to 1. (First picture)
Then call **showDivs()** to display the first image.
When the user clicks one of the buttons call **plusDivs()**.
The plusDivs() function **subtracts** one or **adds** one to the slideIndex.
The **showDiv()** function hides (**display="none"**) all elements with the class name "mySlides", and displays (**display="block"**) the element with the given slideIndex.
If the slideIndex is **higher than** the number of elements (x.length), the slideIndex is set to zero.
If the slideIndex is **less than** 1 it is set to number of elements (x.length).

```
<html>
<title>slideshow</title>
<body>
<h2 class="w3-center">Manual Slideshow</h2>
```

```
<div class="w3">
  <img class="mySlides" src="y.jpg" style="width:50%">
  <img class="mySlides" src="yy.jpg" style="width:50%">
  <img class="mySlides" src="yyy.jpg" style="width:50%">
  <img class="mySlides" src="yyyy.jpg" style="width:50%">

  <button class="aa" onclick="plusDivs(-1)">&#10094;Back</button>
  <button class="bb" onclick="plusDivs(1)">&#10095;Forward</button>
</div>

<script>
var slideIndex = 1;
showDivs(slideIndex);

function plusDivs(n)
{
  showDivs(slideIndex += n);
}

function showDivs(n)
 {
  var i;
  var x = document.getElementsByClassName("mySlides");
  if (n > x.length)
     {
     slideIndex = 1
     }

  if (n < 1)
     {
     slideIndex = x.length
     }
  for (i = 0; i < x.length; i++)
     {
     x[i].style.display = "none";
     }
  x[slideIndex-1].style.display = "block";
}
```

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

**Exercise:**

1) Create rotating banner Ads with URL links.
2) Create a slideshow with group of four images, also simulate the next and previous transition between slides in javascript.

**Beyond Syllabus Experiments:**

## Work with <iframe>

The <iframe> tag specifies an inline frame.
An inline frame is used to embed another document within the current HTML document.

**Attributes**

| Attribute | Value | Description |
|---|---|---|
| allow | | Specifies a feature policy for the <iframe> |
| allowfullscreen | true<br>false | Set to true if the <iframe> can activate fullscreen mode by calling the requestFullscreen() method |
| height | pixels | Specifies the height of an <iframe>. Default height is 150 pixels |
| name | text | Specifies the name of an <iframe> |
| src | URL | Specifies the address of the document to embed in the <iframe> |
| srcdoc | HTML_code | Specifies the HTML content of the page to show in the <iframe> |
| width | pixels | Specifies the width of an <iframe>. Default width is 300 pixels |

**Exercise:**

1) Write a code to incorporate 3 windows by using <iframe>.
2) Demonstrate how iframe is used as a target for the hyperlinks.

**Beyond Syllabus Experiments:**

## Work with jQuery

jQuery is a fast, lightweight, and feature-rich JavaScript library that is based on the principle "write less, do more". It's easy-to-use APIs makes the things like HTML document traversal and manipulation, event handling, adding animation effects to a web page much simpler that works seamlessly across all the major browsers like Chrome, Firefox, Safari, Internet Explorer, etc.

jQuery was originally created by John Resig in early 2006. The jQuery project is currently run and maintained by a distributed group of developers as an open-source project.

jQuery also gives you the ability to create an Ajax based application in a quick and simple way. Big companies like Google, Microsoft and IBM are using the jQuery for their applications.

Simple example to display “Hello VP” using jQuery:

```
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>jQuery Document Ready Demo</title>
   <link rel="stylesheet" type="text/css" href="/examples/css/style.css">
   <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
   <script>
      $(document).ready(function(){
         $("p").text("Hello VP!");
      });
   </script>
</head>
<body>
   <p>Not loaded yet.</p>
</body>
</html>
```

**Exercise:**

1) Create a code on hover(onmouseover) event in jQuery.
2) Create slide-up and slide-down effect in jQuery.