Client Side Scripting Language

EDITION : 2019

Sub Code : 22519

MSBTE I SCHEME PATTERN
T. Y. DIPLOMA SEM V (ELECTIVE)
COMP ENGINEERING / IT PROGRAM GROUP
(CO/CM/IF/CW)

INCLUDES-I SCHEME PATTERN

SAMPLE PAPERS

TECHNICAL PUBLICATIONS
An Up-Thrust for Knowledge

A. A. Puntambekar

As per Revised Syllabus of

MSBTE - I SCHEME

Client Side Scripting Language

T. Y. Diploma (Semester - V) Elective
Computer Engineering / IT Program Group (CO/CM/IF/CW)

Mrs. Anuradha A. Puntambekar

M.E. (Computer)
Formerly Assistant Professor in
P.E.S. Modern College of Engineering,
Pune

TECHNICAL PUBLICATIONS
An Up-Thrust for Knowledge

Website : www.technicalpublications.org
https://www.facebook.com/technicalpublications

Client Side Scripting Language

T.Y. Diploma (Semester - V) Elective
Computer Engineering / IT Program Group (CO/CM/IF/CW)

First Edition : June 2019

Published by :
TECHNICAL PUBLICATIONS
Amit Residency, Office No.1, 412, Shaniwar Peth, Pune - 411030, M.S. INDIA
Ph.: +91-020-24495496/97, Telefax : +91-020-24495497
Email : sales@technicalpublications.org Website : www.technicalpublications.org

Printer :
Yogiraj Printers & Binders
Sr.No. 10\1A,
Ghule Industrial Estate, Nanded Village Road,
Tal-Haveli, Dist-Pune - 411041.

Price : ₹ 130/-
ISBN 978-93-89180-38-1

MSBTE I

PREFACE

The importance of **Client Side Scripting Language** is well known in various fields. Overwhelming response to my books on various subjects inspired me to write this book. The book is structured to cover the key aspects of the subject **Client Side Scripting Language**.

The book uses plain, lucid language to explain fundamentals of this subject. The book provides logical method of explaining various complicated concepts and stepwise methods to explain the important topics. Each chapter is well supported with necessary illustrations, practical examples and solved problems. All the chapters in the book are arranged in a proper sequence that permits each topic to build upon earlier studies. All care has been taken to make students comfortable in understanding the basic concepts of the subject.

Representative questions have been added at the end of each chapter to help the students in picking important points from that chapter.

The book not only covers the entire scope of the subject but explains the philosophy of the subject. This makes the understanding of this subject more clear and makes it more interesting. The book will be very useful not only to the students but also to the subject teachers. The students have to omit nothing and possibly have to cover nothing more.

I wish to express my profound thanks to all those who helped in making this book a reality. Much needed moral support and encouragement is provided on numerous occasions by my whole family. I wish to thank the **Publisher** and the entire team of **Technical Publications** who have taken immense pain to get this book in time with quality printing.

Any suggestion for the improvement of the book will be acknowledged and well appreciated.

Author
A. A. Puntambekar

SYLLABUS

Client Side Scripting Language (22519)

Teaching Scheme			Credit (L + T + P)	Examination Scheme											
				Theory						Practical					
L	T	P		Paper Hrs.	ESE		PA		Total		ESE		PA		Total	
					Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
3	-	2	5	3	70	28	30*	00	100	40	25#	10	25	10	50	20

Unit	Unit Outcomes (UOs) (in cognitive domain)	Topics and Sub - topics
Unit - I **Basics of JavaScript Programming**	1a. Create object to solve the of given problem. 1b. Develop JavaScript to implement the switch-case statement for the given problem. 1c. Develop JavaScript to implement loop for solving the given iterative problem. 1d. Display properties of the given object using getters and setters. 1e. Develop program using. basic features of JavaScrip to solve the given problem.	1.1 Features of J avaScript 1.2 Object Name, Property, method, Dot syntax, main event. 1.3 Values and Variables 1.4 Operators and Expressions, Primary expressions, Object and Array initializers, function definition expression, property access expressions, invocation expressions. 1.5 If Statement, if. ... else, if.. elseif, nested if statement. 1.6 Switch ... case statement 1.7 Loop statement - for loop, for. . .in loop, while loop, do ... while loop, continue statement. 1.8 Querying and setting properties and deleting properties property getters and setters.

Unit - II **Array, Function and String**	2a. Create array to solve the given problem. 2b. Perform the specified string and String manipulation operation on the given String(s). 2c. Develop JavaScript to implement the given function. 2d. Develop JavaScript to convert the given Unicode to character form. 2e. Develop JavaScript to convert the given character to Unicode and vice-versa.	2.1 Array - declaring an Array, Initializing an Array, defining an Array elements, Looping an Array, Adding an Array an element, sorting an Array element, Combing an Array elements into a String, changing elements of an Array, Objects as associative Arrays. 2.2 Function - defining a function, writing a function, adding an arguments, scope of variable and arguments, 2.3 Calling a function - calling a function with or without an argument, calling function from HTML, function calling another function. Returning a value from a function. 2.4 String - manipulate a string joining a string retrieving a character from given position, retrieving a position of character in a string, dividing text, copying a sub string, converting stringm to number and numbers to string a Unicode of a character - charCodeAt(), from CharcCode().
Unit - III **Form and Event Handling**	3a. Write JavaScript to design a form to accept input values for the given problem. 3b. Use JavaScript to implement form events to solve the given problem. 3c. Develop JavaScript to dynamically assign specified attribute value to the given form control. 3d. Use the given intrinsic function with specified parameters.	3.1 Building blocks of a Form, properties and methods of form, button, text, text area, checkbox, radio button, select element. 3.2 Form events- mouse event, key events. 3.3 Form objects and elements. 3.4 Changing attribute value dynamically. 3.5 Changing option list dynamically 3.6 Evaluating checkbox selection 3.7 Changing a label dynamically 3.8 Manipulating form elements 3.9 Intrinsic JavaScript functions, disabling elements, read only elements.
Unit - IV **Cookies and Browser Data**	4a. Create cookies based on the given problem. 4b. Develop JavaScript to manage a cookie in the given manner. 4c. Write JavaScript to manipulate the specified attributes of window object in the given manner. 4d. Write JavaScript to create browser history of the given object.	4.1 Cookies - basic of cookies, reading a cookie value, writing a cookie value, creating a cookies, deleting a cookies, setting the expiration date of cookie. 4.2 Browser - opening a window, giving the new window focus, window position, changing the content of window, closing a window, scrolling a web page, multiple windows at once, creating a web page in new window, JavaScript in URLs, JavaScript security, Timers, Browser location and history.

Unit - V **Regular Expression, Rollover and Frames**	5a. Compose relevant regular expression for the given character pattern search. 5b. Develop JavaScript to implement validations using the given regular expression. 5c. Create frames based on the the given problem. 5d. Create window object as per the given problem. 5e. Develop JavaScript for creating rollover effect for the given situation.	5.1 Regular Expression - language of regular expression, finding non matching characters, entering a range of characters, matching digits and non digits, matching punctuations and symbols, matching words, replacing a the text using regular expressions, returning the matched of regular expression object properties. 5.2 Frames - create a frame, invisible borders of frame, calling a child windows, changing a content and focus of a child window, writing to a child window, accessing elements of another child window. 5.3 Rollover- creating rollver, text rollver, Multiple actions for rollover, more efficient rollover.
Unit - VI **Menus, Navigation and Web Page Protection**	6a. Develop JavaScript to manage the given status bar. 6b. Develop JavaScript to create the given banner. 6c. Develop JavaScript to create the given slide show. 6d. Develop JavaScript to create the given Menu. 6e. Write JavaScript to protect a webpage in the specified manner.	6.1 Status bar - builds a static message, changing the message using rollover, moving the message along the status bar 6.2 Banner - loading and displaying banner advertisement. Linking a banner advertisement to url 6.3 Slide Show- creating a slide show 6.4 Menus- creating a pulldown menu, dynamically changing a menu, validating menu selection, Floating menu, chain select menu, tab menu, pop-up menu, sliding menu, highlighted menu, folding a tree menu, context menu, scrollable menu, side bar menu. 6.5 Protecting web page - hiding your code, disabling the right mouse button, JavaScript, concealing email address. 6.6 Frameworks of javasript and its application.

TABLE OF CONTENTS

Unit - III

Unit - IV

Unit - V

Unit - VI

UNIT - I

1 Basics of JavaScript Programming

1.1 Features of JavaScript

- JavaScript was developed by Netscape in 1995. At that time its name was LiveScript. Later on Sun Microsystems joined the Netscape and then they developed LiveScript. And later on its name is changed to JavaScript.

Following are some features of JavaScript

1. **Browser support :** For running the JavaScript in the browser there is no need to use some plug-in. Almost all the popular browsers support JavaScripting.
2. **Structure programming syntax :** The Javascript supports much commonly the structured language type syntax. Similar to C-style the programming blocks can be written.
3. It automatically inserts the semicolon at the end of the statement, hence there is no need to write semicolon at the end of the statement in JavaScript.
4. **Dynamic typing :** It supports dynamic typing, that means the data type is bound to the value and not to the variable. For example one can assign integer value to a variable say 'a' and later on can assign some string value to the same variable in JavaScript.
5. **Run time evaluation :** Using the eval function the expression can be evaluated at run time.
6. **Support for object :** JavaScript is object oriented scripting language. However handling of objects in JavaScript is somewhat different that the conventional object oriented programming languages. JavaScript has a small number of in-built objects.
7. **Regular expression :** JavaScript supports use of regular expressions using which the text-pattern matching can be done. This feature can be used to validate the data on the web page before submitting it to the server.
8. **Function programming :** In JavaScript functions are used. One function can accept another function as a parameter. Even, one function can be assigned to a variable just like some data type. The function can be run without giving the name.

1.1.1 How to Write JavaScript Document?

The JavaScript can be directly embedded within the HTML document or it can be stored as external file.

Syntax

The syntax of directly embedding the JavaScript in the HTML is

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

There are two important attributes of script tag – **type** and **language.**

The **type** attribute can be written as follows –

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

The **language** attribute can be written as follows –

```
<script language="javascript">
//script here
</script>
```

Ex. 1.1.1 : ***Write a JavaScript to display Welcome message in JavaScript.***

Sol. :

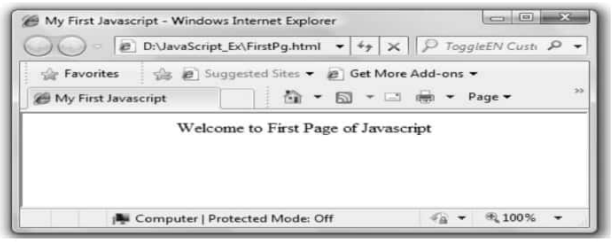
```
<!DOCTYPE html >
<html >
    <head>
        <title> My First Javascript </title>
    </head>
    <body>
        <center>
        <script type="text/javascript">
        /*This is the First JavaScript*/
        document.write(" Welcome to First Page of Javascript");
        </script>
        </center>
    </body>
</html>
```

How to run JavaScript Document ?

For running your JavaScript, just open the Web browser like Internet Explorer, Mozilla FireFox, Chrome and type the name of the file on the address bar along with its complete pathname.

For example – My JavaScript program has a name **FirstPg.html** and it is present at the location **d:/JavaScript_Ex** folder. Hence the complete path is typed at the address bar

Output



Script Explanation :

In above script

1) We have embedded the javaScript within
   <script type="text/javascript">
   …
   </script>
2) A comment statement using /* and */. Note that this type of comment will be recognized only within the <script> tag. Because, JavaScript supports this kind of comment statement and not the XHTML document.

3) Then we have written **document.write** statement, using which we can display the desired message on the web browser. To print the desired message on the web browser we write the message in double quotes inside the **document.write** method.

- **For example**

```
document.write("Welcome to First Page of Javascript");
```

Comments in JavaScript

JavaScript supports following comments

1. The // i.e a single line comment can be used in JavaScript.
2. The /* and */ can be used as a multi-line comment.
3. The XHTML <!--> and <--> is also allowed in JavaScript

1.2 Object Name, Property, Method, Dot Syntax, Main Event

In this section we will discuss some basic concepts that are used in JavaScript.

Object Name

- JavaScript is a object oriented programming language.
- That means programs are written using objects.
- Object is nothing but some entity. In JavaScript document, window, forms, fields, buttons are some popularly used objects.
- Each object is identified by either an ID or by a name.
- Sometimes an array or collection of objects can also be created and particular object can be accessed from that array.

Property

- Property is actually a value associated with each object.
- For example – for the object **window** the **height** and **width** are the properties that are associated with it.
- Each object has its own set of properties.

Method

- **Method** is a function or a process associated with each object.
- For example – for the **document** object the method is **write**. To this write method we pass the string which we want to get displayed on the browser window.

Dot Syntax

- Every object is associated with some property and method.
- For accessing the properties and methods of an object we use dot operator.
- For example –

```
document.write("Hello")//accessing method write of
                       //document object.
```

Main Event

- Event is something that causes JavaScript to execute the code.
- For example – if you press a key for F1 and function for F1 gets executed.
- In JavaScript when user submits the form, clicks some button, writes something to text box then corresponding events get triggered.
- Execution of appropriate code on occurrence of event is called **event handling**.
- Normally some functions are defined for event handling. These functions are called event handler.

1.3 Values and Variables

1.3.1 Values

JavaScript uses six types of values – number, String, Boolean, null, object and function

Number

Number is a numeric value that can be integer or float. It can be used calculation.

String

String is a collection of characters. It is enclosed within single quote or double quote. A string may contain numbers but these numbers can not be used for calculations.

Boolean

The Boolean values are **true** and **false**. These values can be compared with the variables or can be used in assignment statement.

Null

The null value can be assigned by using the reserved **word null**. The **null** means no value. If we try to access the null value then a runtime error will occur.

Object

Object is for an entity that represents some value. For example – window object is a value using which the window is created. Form is an object upon which some components such as button, checkbox, radio buttons can be placed and used.

Function

Function is intended for execution of some task. There can be predefined function and user defined function. For example – **alert()** is a predefined function, using which a popup window having some message can be displayed. Similarly, one can write his/her own function called user defined function in JavaScript. The keyword **function** is used while defining the user defined function.

1.3.2 Variables

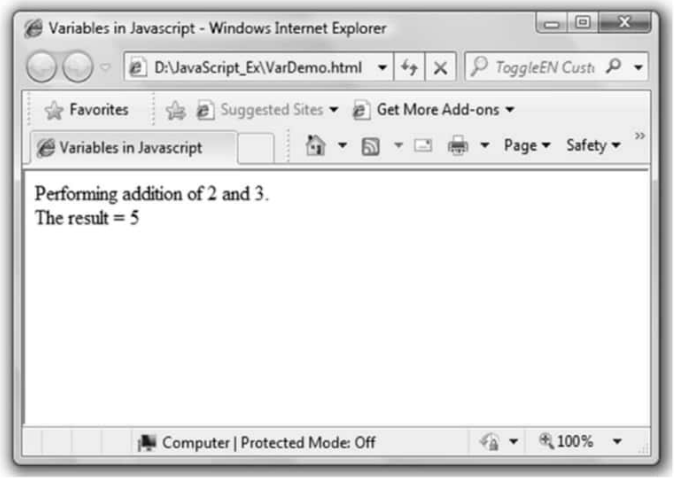
In JavaScript we can declare the variable using the reserved word **var.** The value of this variable can be any thing; it can be numeric or it can be string or it can be a Boolean value.

JavaScript[VarDemo.html]

```
<!DOCTYPE html >
<html >
 <head>
 <title> Variables in Javascript </title>
</head>
 <body>
  <script type="text/javascript">
   var a,b,c;
   var string;
   a=2;
   b=3;
   c=a+b;
   string ="The result = ";
   document.write("Performing addition of 2 and 3.  "+"<br/>");
   document.write(string);
   document.write(c);
  </script>
</body>
</html>
```

Variable declaration is done using **var**. Note that there is no data type required for handling variables.

Output



Note that using **var** we can define the variable which is of type numbers(2 , 3 or 5) as well as the string "The result = ".

Rules for Using Variables

Following are some conventions used in JavaScript for handling the identifiers -

1. Identifiers must begin with either letter or underscore or dollar sign. It is then followed by any number of letters, underscores, dollars or digits.
2. There is no limit on the length of identifiers.
3. The letters in the identifiers are case-sensitive. That means the identifier INDEX, Index, index, inDex are considered to be distinct.
4. Programmer defined variable names must not have upper case letters.

1.3.3 Keywords

Keywords are basically the reserved words are the special words associated with some meaning. Various keywords that are used in JavaScript are enlisted as below –

break	continue	delete	for	in	return	throw	var	with
case	default	else	function	instanceof	switch	try	void	
catch	do	finally	if	new	this	typeof	while	

1.4 Operators and Expressions

Various operators used by JavaScript are as shown in following table -

Type	Operator	Meaning	Example
Arithmetic	+	Addition or unary plus	c = a+b
	–	Subtraction or unary minus	d = –a
	*	Multiplication	c=a*b
	/	Division	c=a/b
	%	Mod	c=a%b
Relational	<	Less than	a<4
	>	Greater than	b>10
	<=	Less than equal to	b<=10
	>=	Greater than equal to	a>=5
	==	Equal to	x==100
	!=	Not equal to	m!=8
Logical	&&	And operator	0&&1
	||	Or operator	0 ||1
Assignment	=	Is assigned to	a=5
	+=	add a value then assign	a+=5 means a=a+5
	-=	subtract a value then assign	a-=10 means a=a-10
	/=	divide the value then assign	a/=2
	*=	multiply and then assign	a*=5
Increment	++	Increment by one	++i or i++
Decrement	--	Decrement by one	– – k or k– –

Conditional Operator

The conditional operator is ? The syntax of conditional operator is

```
Condition?expression1:expression 2
```

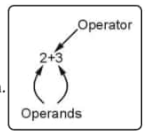
Where expression1 denotes the true condition and expression2 denotes false condition.

For example :

```
a>b?true:false
```

This means that if a is greater than b then the expression will return the value true otherwise it will return false.

Expressions



JavaScript makes use of mathematical statements using operators and operands.

These statements are called expressions. For example – Figure shows a simple expression.

1.5 If Statement

As a selection statement we use various types of if statements. These statements are -

1. **if** - The syntax of if is

```
if(condition)
        statement if the condition is true
```

2. **if…else** - The syntax of if…else is

```
if(condition)
        statement if the condition is true
else
        statement if the condition is false
```

3. **if…else if -** The syntax of if…else if is

```
if(condition)
        statement if the condition is true
else if(condition)
        statement if another condition is true
else if(condition)
        statement if another condition is true
…
else
        statement
```

Some times we can have **nested** if…statements, which work similar to C or C++. Here is a sample JavaScript

JavaScript[IfDemo.html]

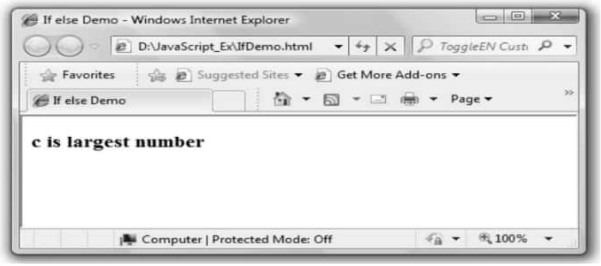
```
<!DOCTYPE html>
<html>
<head>
  <title>If else Demo</title>
</head>
<body>
 <script type="text/javascript">
  var a,b,c;
  a=10;b=20;c=30;
  if(a>b)
  {
   if(a>c)
    document.write("<h3>a is largest number</h3>");
   else
    document.write("<h3>c is largest number</h3>");
```

```
 }
 else
 {
  if(b>c)
    document.write("<h3>b is largest number</h3>");
 else
    document.write("<h3>c is largest number</h3>");
 }
 </script>
</body>
</html>
```

Script Explanation

The above document is to find the largest number from the given three numbers. We have used nested if…else statements. The script is pretty simple, in which first of all we have compared a with b, if a is greater than b then we have compared a with c. Similarly in the else part b is compared with c. Thus all the three numbers get compared with each other and the largest number is found out among the three. Since already in our program we have set the a, b and c values as 10, 20 and 30 respectively the output should be c is **largest number** and here comes the output on web browser.

Output



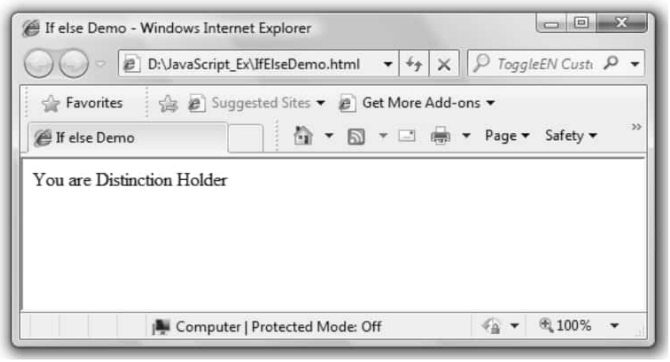
We can use various operators such as arithmetic operator, logical operator, relational operators and so on in the if statement. Following JavaScript makes use of such operators.

JavaScript[IfElseDemo.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>If else Demo</title>
</head>
<body>
 <script type="text/javascript">
  var marks;
```

```
    marks=80;
    if(marks<40)
      document.write("You are failed");
    else if(marks>=40 && marks<50)
      document.write("You are passed");
    else if(marks>=50 && marks<60)
      document.write("You have got Second class");
    else if(marks>=60 && marks<66)
      document.write("You have got First class");
    else
      document.write("You are Distinction Holder");
  </script>
</body>
</html>
```

Output



In above script if we change the values of the marks variable then appropriate message will get displayed on the web browser.

1.6 Switch Case Statement

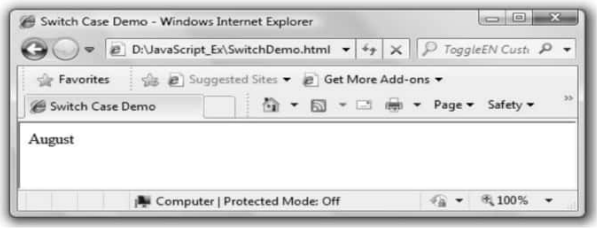
Switch case statement is basically to execute the desired choice. The syntax of this control structure is similar to that in C or C++.

```
switch(choice)
{
 case identifier: statement
          break
 ...
 default: statement
}
```

JavaScript[SwitchDemo.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>Switch Case Demo</title>
</head>
<body>
 <script type="text/javascript">
  d=new Date();
  ch=d.getMonth();
  switch(ch)
  {
   case 0: document.write("January");
        break;
   case 1: document.write("February");
        break;
   case 2: document.write("March");
        break;
   case 3: document.write("April");
        break;
   case 4: document.write("May");
        break;
   case 5: document.write("June");
        break;
   case 6: document.write("July");
        break;
   case 7: document.write("August");
        break;
   case 8: document.write("September");
        break;
   case 9: document.write("October");
        break;
   case 10: document.write("November");
        break;
   case 11: document.write("December");
        break;
  }
 </script>
</body>
</html>
```

Output



Script Explanation : In this script,

(1) We have used the object **Date** which returns the value of the current date.

(2) Then using **getMonth()** method the month value can be obtained. The month value starts from 0 to 11 representing January to December. This value is basically obtained from system date function. The month is taken in variable **ch** according to the value in **ch** the corresponding case will get executed.

1.7 Loop Statement

1.7.1 while Loop

while statements help us in implementing the iterative logic of the program. The syntax of **while** is as follows -

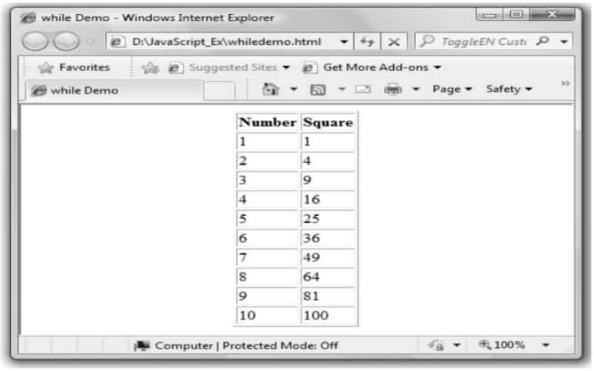
```
Some initial condition;
while(terminating condition)
{
 some statements;
 stepping condition;
}
```

The **while** is implemented by following JavaScript.

```
<!DOCTYPE html>
<html>
<head>
  <title>while Demo</title>
</head>
<body>
 <table border=1 align="center">
  <th>Number</th><th>Square</th>
  <script type="text/javascript">
   i=1;
   while (i<=10)
   {
    document.write("<tr><td>"+i+"</td><td>"+(i*i)+"</td></tr>");
    i++;
```

```
    }
  </script>
</table>
</body>
</html>
```

Output



Number	Square
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

1.7.2 do..while Loop

The do-while loop is similar to the while loop, the only difference is that the do-while executes at least once. The syntax of do…while is

```
do
{
 …
}while(condition);
```

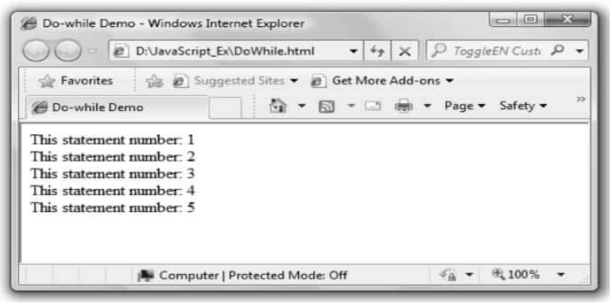
The following JavaScript illustrates use of do…while

JavaScript[DoWhile.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>Do-while Demo</title>
</head>
<body>
 <script type="text/javascript">
  counter=1;
  do
  {
    document.write("This statement number: "+counter);
    document.write("<br>");
```

```
    counter++;
  }while(counter<=5);
 </script>
</body>
</html>
```

Output



1.7.3 for Loop

This is the most commonly used programming construct. The syntax of for loop is

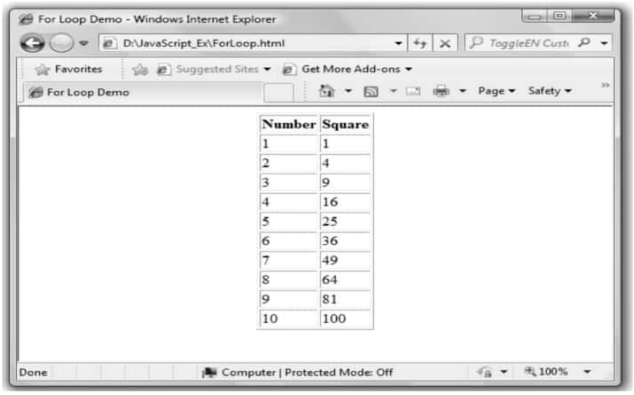
for(initial condition; terminating condition; stepping condition)

Here is a JavaScript which makes use of for loop

JavaScript[ForLop.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>For Loop Demo</title>
</head>
<body>
 <table border=1 align="center">
  <th>Number</th><th>Square</th>
  <script type="text/javascript">
   for (i=1; i<=10; i++)
   {
document.write("<tr><td>"+i+"</td><td>"+(i*i)+"</td></tr>");
   }
  </script>
</table>
</body>
</html>
```

Output



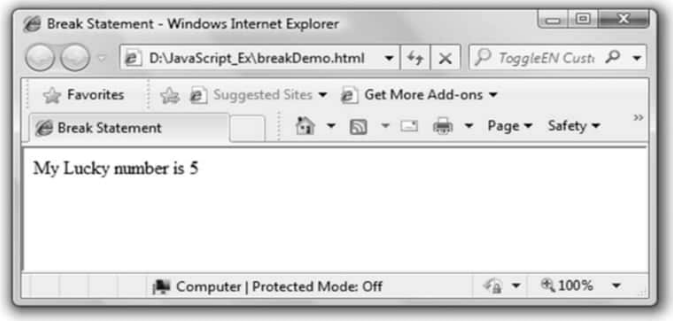
1.7.4 break Statement

Similar to C or C++, the break statement is used to break the loop. It is given be keyword **break.** The following script shows the use of break.

JavaScript[breakDemo.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>Break Statement</title>
</head>
<body>
   <script type="text/javascript">
    for(i=10;i>=0;i--)
    {
      if(i==5)
       break;
    }
    document.write("My Lucky number is "+i);
</script>
</body>
</html>
```

Output



1.7.5 The continue Statement

The continue statement is used in a loop in order to continue(skip). The keyword **continue** is used to make use of continue statement in a loop.

Javascript Program[ContinueDemo.html]

```
<!DOCTYPE html>
<html>
<head>
<title>Continue Statement</title>
</head>
<body>
<script type="text/javascript">
  for(i=10;i>=0;i--)
  {
    if(i==5)
    {
      x=i;
      continue;
    }
    document.write(i);
    document.write("<br>");
  }
  document.write("The number "+x+" is missing in above list" );
</script>
</body>
</html>
```

Output

1.8 Querying and Handling Properties

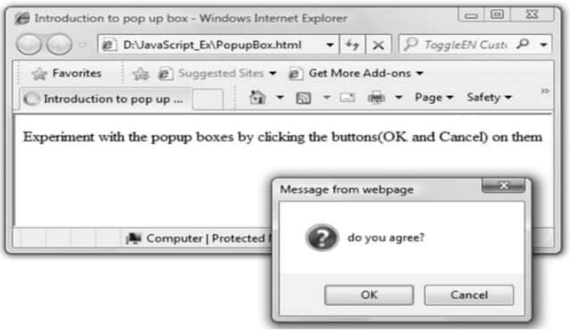
- One of the important features of JavaScript is its **interactivity** with the user.
- There are **three types of popup boxes** used in JavaScript by which user can interact with the browser.

alert box : In this type of popup box some message will be displayed.	
confirm box : In this type of popup box in which the message about confirmation will be displayed. Hence it should have two buttons **Ok** and **Cancel.**	
Prompt box is a type of popup box which displays a text window in which the user can enter something. Hence it has two buttons **Ok** and **Cancel.**	

JavaScript[PopupBox.html]

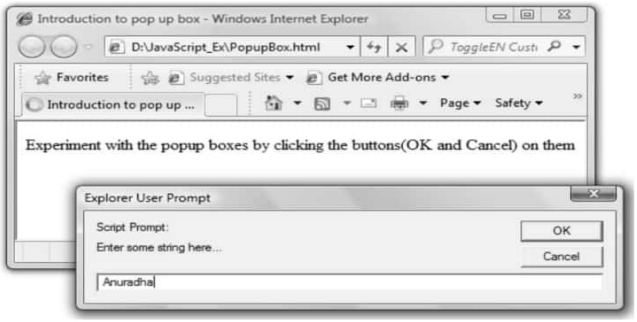
```
<!DOCTYPE html>
<html >
 <head>
  <title>Introduction to pop up box</title>
</head>
<body>
<p>Experiment with the popup boxes by clicking the buttons(OK and Cancel) on them</p>

  <script type="text/javascript">
   if(confirm("do you agree?"))
     alert("You have agreed");
   else
     input_text=prompt("Enter some string here...","");
   /*the value entered in prompt box is returned
   and stored in the variable text */
     alert("Hi "+input_text);
  </script>
</body>
</html>
```

Output

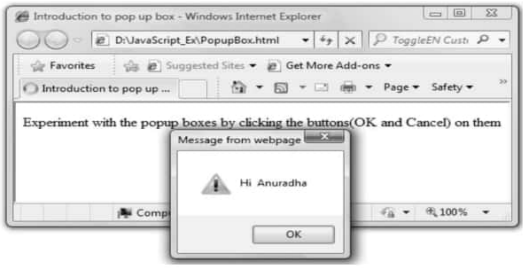


If we click on OK button then an alert box will appear.

On load, the confirm box appears and if we click on Cancel button then the prompt box will appear. We can type some string within it.



Click OK button and we will get the alert box as follows -



Thus alert, confirm and prompt boxes cause the browser to wait for user response. The user responds by clicking the button on these popup boxes.

Script Explanation

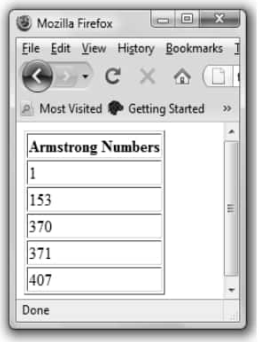
On loading this script using web browser first of all a confirm box will be displayed. If we click on OK button an alert box will appear. Otherwise a prompt box will be displayed. Again if we click on the OK button of prompt box again an alert box will appear which will display the string which you have entered on prompt box. If you run the above script you will get all these effects.

1.9 More Examples on JavaScript

Ex. 1.9.1 : ***Develop a javascript to generate 'ARMSTRONG NUMBERS' between the range 1 to 100. [Eg : 153 is an Armstrong number, since sum of the cube of the digits is equal to the number i.e. [$1^3 + 5^3 + 3^3 = 153$]***

Sol. :

```
<html>
<body>
<table border="1" allign="center">
<th>Armstrong Numbers</th>
<script type="text/javascript">
var num,i,temp,sum;
var n=0;
i=1;
do
{
 num=i;
 sum=0;
 while(num>0)
 {
  n=num%10;
  n=parseInt(n);
  num=num/10;
  num=parseInt(num);
  sum=sum+(n*n*n);
 }
 if(sum==i)
 {
  document.write("<tr><td>"+i+"</td></tr>");
 }
 i++;
}while(i<=1000);
</script>
</table>
</body>
</html>
```

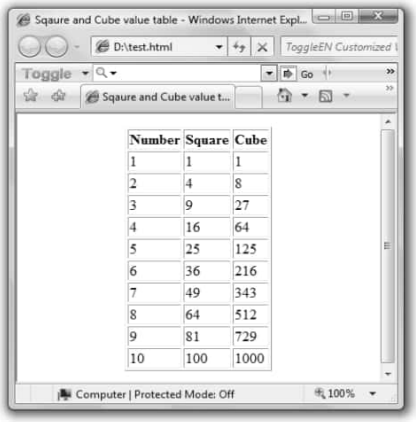


Ex. 1.9.2 : ***Write a javascript that displays table as per following: (Calculate the squares and cubes of the numbers from 0 to 10)***

Number	Square	Cube
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

Sol.

```
<html>
<head>
<title>Sqaure and Cube value table</title>
</head>
<body>
<table border=1 align="center">
<th>Number</th><th>Square</th><th>Cube</th>
<script type="text/javascript">
for (i=1; i<=10; i++)
{
document.write("<tr><td>"+i+"</td><td>"+(i*i)+"</td><td>"+(i*i*i)+"</td></tr>");
}
</script>
</table>
</body>
</html>
```



Number	Square	Cube
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

Ex. 1.9.3 : *Write a script that reads an integer and displays whether it is a prime number or not.*

Sol. :

```
<html>
 <head>
  <title>PRIME NUMBER DEMO</title>
 </head>
 <body>
  <script type="text/javascript">
  var num=prompt("Type number here.","");
  var b;
  var flag=1;
  for(i=2;i<num;i++)
  {
    b=num%i;
    if(b==0)
    {
     flag=0;
     break;
    }
  }
  if(flag==0)
    alert(num+" is not a prime number");
  else
    alert(num+" is a prime number");
</script>
</body>
</html>
```

Output

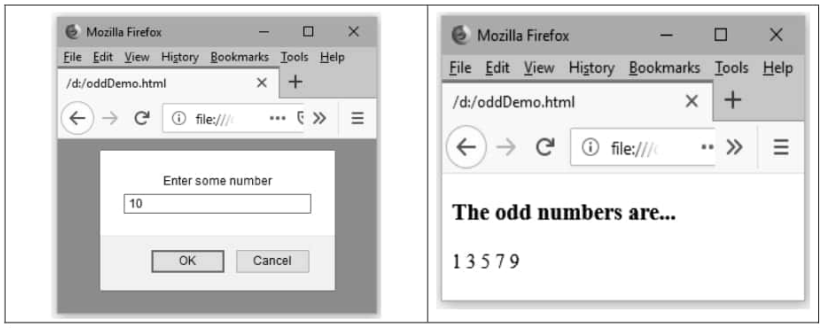
Ex. 1.9.4 : *Write a JavaScript program which accepts N as input and print first N odd numbers.*

Sol. :

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
    function fun(str)
    {
        var num=Number(str);
        var i,j,k,count;
        document.write("<h3>"+" The odd numbers are..."+"</h3>");
        for(i=0;i<=num;i++)
        {
            if(i%2!=0)
                document.write("  "+i);
        }
    }
 </script>
</head>
<body>
<script type="text/javascript">
```

```
        var input_str=prompt("Enter some number","");
        fun(input_str);
    </script>
    </body>
    </html>
```

Output



Review Questions

1. *Explain any two features of JavaScript*
2. *Write a JavaScript to display Welcome message in JavaScript.*
3. *Explain the terms – Object Name, Property, Method, Dot Syntax*
4. *Explain six types of values in JavaScript*
5. *What is conditional operator in JavaScript?*
6. *Write a JavaScript to display squares of 1 to 10 numbers using while loop.*
7. *Write a JavaScript to display squares of 1 to 10 numbers using for loop.*
8. *Develop a JavaScript to generate 'ARMSTRONG NUMBERS' between the range 1 to 100. [Eg : 153 is an Armstrong number, since sum of the cube of the digits is equal to the number i.e. [$1^3 + 5^3 + 3^3 = 153$]*
9. *Write a JavaScript program which accepts N as input and print first N odd numbers.*

❑❑❑

UNIT - II

2 Array, Function and String

2.1 Array

- **Definition of Arrays :** Arrays is a collection of similar type of elements which can be referred by a common name.
- Any element in an array is referred by an array name followed by "[" followed by position of the element followed by].
- The particular position of element in an array is called array index or subscript.

For example –

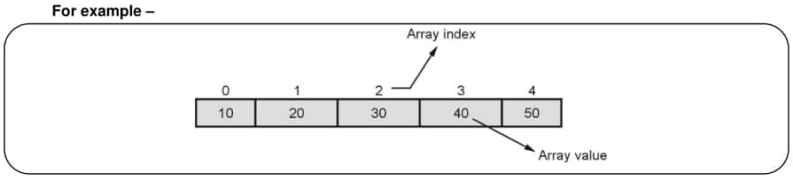


Fig. 2.1.1 Arrays

- Normally the first element in an array is stored at 0^{th} location, however we can start storing the element from any position.

2.1.1 Declaring an Array

- In JavaScript the array can be created using **Array** object.
- Suppose, we want to create an array of 10 elements then we can write,

```
var ar = new Array(10);
```

- Using new operator we can allocate the memory dynamically for the arrays.
- In the brackets the size of an array is mentioned and the var ar denotes the name of the array. Thus by the above sentence an array ar will be created in which we can store 10 elements at the most. Sometimes the above statement can be written like this

```
var ar;
    ar=new Array(10);
```

2.1.2 Initializing an Array

- **Initialization** is the process of assigning a value when either a variable or array is declared. For example

```
int count=10//variable initialization
```

- While initializing an array -
  1) Declare name of the array
  2) Make use of the keyword **new**
  3) Each value within an array as an 'array element' must be separated by comma.
- For example –

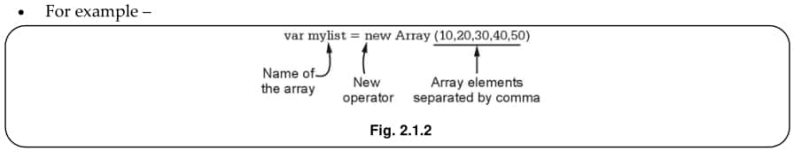


Fig. 2.1.2

2.1.3 Defining an Array Elements

- The elements in the array are stored from index 0.
- For example – elements in array 10,20,30,40,50 are stored as follows –

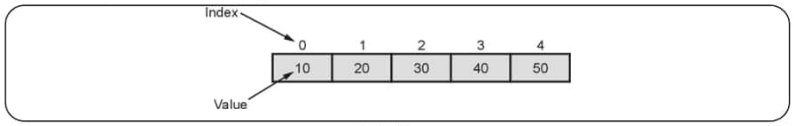


Fig. 2.1.3

- Following is a simple JavaScript that illustrates the initialization of array by defining the array elements.

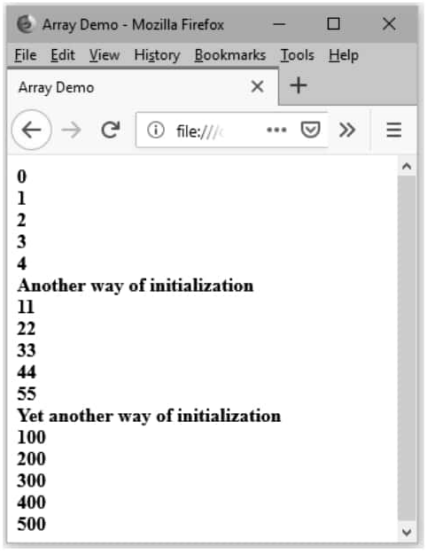
JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
 <title>Array Demo</title>
</head>
<body>
<strong>
 <script type="text/javascript">
  a=new Array(5);//creation of array
  for(i=0;i<5;i++)
  {
   a[i]=i;
   document.write(a[i]+"<br>");//displaying array
    }
   document.write("Another way of initialization"+"<br>");
   b=new Array(11,22,33,44,55);//creation of array
   for(i=0;i<5;i++)
   {
   document.write(b[i]+"<br>");//displaying array
   }
   document.write("Yet another way of initialization"+"<br>");
```

```
        var c=[100,200,300,400,500];//creation of array
        for(i=0;i<5;i++)
        {
        document.write(c[i]+"<br>");//displaying array
        }
        </script>
        </strong>
    </body>
    </html>
```

Script Explanation : In above JavaScript, as you can notice that, an array can be initialized in three different ways which is shown by boldface. Hence an output of above script will be

Output



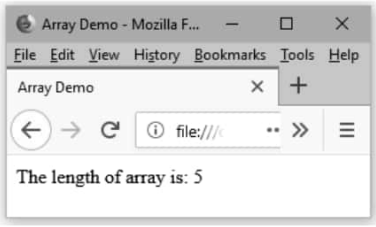
Ex. 2.1.1 : *Write a JavaScript to define the array elements and to find the length of array.*

Sol. :

```
<!DOCTYPE html>
<html>
<head>
    <title>Array Demo</title>
</head>
<body>
 <script type="text/javascript">
      a=new Array(11,22,33,44,55);//creation of array
      var len=a.length
```

```
        document.write("The length of array is: "+len)
        </script>
</body>
</html>
```

Output



Script Explanation : In above JavaScript, We have used **length** property to calculate length of the array. This actually gives total number of elements in the array.

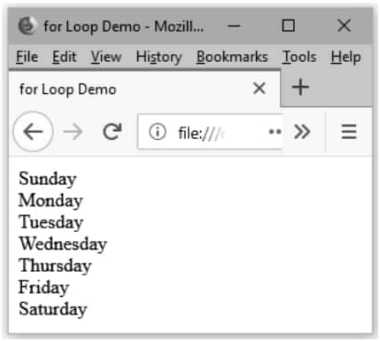
2.1.4 Looping an Array

- Looping an array means visiting each element present in the array.
- Following JavaScript illustrates how to loop or iterate through the elements of array –

JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
        <title>for Loop Demo</title>
</head>
<body>
    <script type="text/javascript">
        Days=new Array();
        Days[0]="Sunday";
        Days[1]="Monday";
        Days[2]="Tuesday";
        Days[3]="Wednesday";
        Days[4]="Thursday";
        Days[5]="Friday";
        Days[6]="Saturday";
        for(i=0;i<Days.length;i++)
        {
            document.write(Days[i]+"<br>");
        }
    </script>
</body>
</html>
```

Output



2.1.5 Adding an Array Element

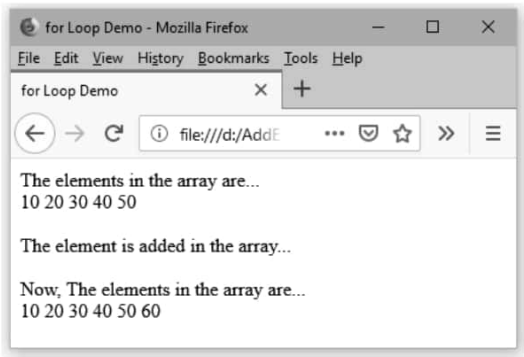
- We can add the element in the array at the end. This increases the size of the array.
- Following example illustrates this idea –

JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
    <title>for Loop Demo</title>
</head>
<body>
    <script type="text/javascript">
        a=new Array();
        a[0]=10;
        a[1]=20;
        a[2]=30
        a[3]=40
        a[4]=50;
        document.write("The elements in the array are...<br/>");
        for(i=0;i<a.length;i++)
        {
            document.write(a[i]+" ");
        }
        document.write("<br/><br/>The element is added in the array...<br/>");
        a[a.length]=60;
        document.write("<br/>Now, The elements in the array are...<br/>");
        for(i=0;i<a.length;i++)
        {
            document.write(a[i]+" ");
        }
```

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

Output



2.1.6 Sorting an Array Element

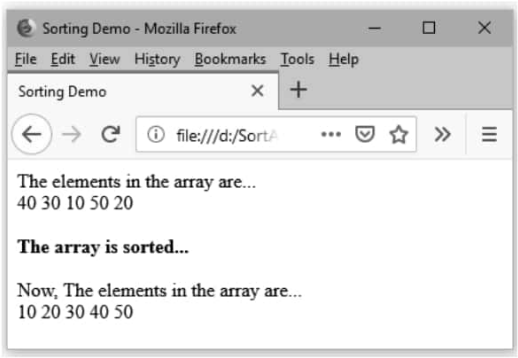
- The elements can be sorted using the built in function **sort.**
- Sorting is basically a process of arranging the elements in ascending order or increasing order.
- For example

JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
    <title>Sorting Demo</title>
</head>
<body>
    <script type="text/javascript">
        a=new Array();
        a[0]=40;
        a[1]=30;
        a[2]=10
        a[3]=50
        a[4]=20;
        document.write("The elements in the array are...<br/>");
        for(i=0;i<a.length;i++)
        {
            document.write(a[i]+" ");
        }
        document.write("<br/><br/><b>The array is sorted...</b><br/>");
        a.sort();
```

```
        document.write("<br/>Now, The elements in the array are...<br/>");
        for(i=0;i<a.length;i++)
        {
            document.write(a[i]+" ");
        }
    </script>
</body>
</html>
```

Output



2.1.7 Combining Array Elements into String

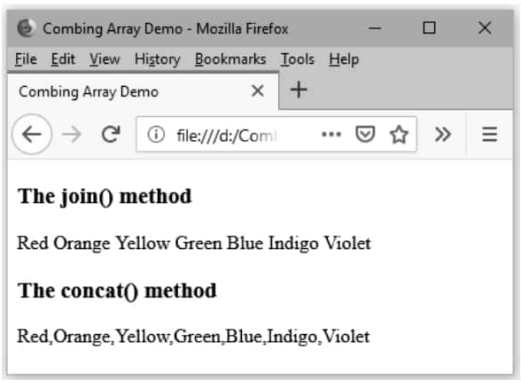
- In JavaScript it is possible to combine the array elements into a string.
- We need to use two functions for combining array elements into string and those are –
  1) Join() 2) concat()
- There is **difference** between join() and concat() function. The **concat()** method separates each value with a comma. The **join()** method also uses a comma to separate values, but you can specify a character other than a comma to separate values.

JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
    <title>Combing Array Demo</title>
</head>
<body>
    <script type="text/javascript">
        a=new Array();
        a[0]="Red";
        a[1]="Orange";
        a[2]="Yellow"
        a[3]="Green"
```

```
        a[4]="Blue";
        a[5]="Indigo";
        a[6]="Violet";
        document.write("<h3> The join() method</h3>");
        var str1=a.join(' ');
        document.write(str1);
        document.write("<h3> The concat() method</h3>");
        var str2=a.concat();
        document.write(str2);

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

Output



2.1.8 Changing Elements of an Array

There are various methods that help in changing the elements of array.

(1) Shift Method

This method removes the first element of the array. For example –

```
<!DOCTYPE html>
<html>
<head>
    <title>Changing Array Demo</title>
</head>
<body>
    <script type="text/javascript">
        a=new Array();
        a[0]=10
        a[1]=20
        a[2]=30
```

```
        a[3]=40
        a[4]=50
        document.write("<h4> The shift() method</h4>");
        var num=a.shift();
        document.write(num);
        document.write("<h4>The elements in array are ...</h4>");
        for(i=0;i<a.length;i++)
        document.write(a[i]+" ");
    </script>
</body>
</html>
```

Output

(2) Push Method

The push method is used to create a new element at the end of the array.

For example –

```
<!DOCTYPE html>
<html>
<head>
    <title>Changing Array Demo</title>
</head>
<body>
    <script type="text/javascript">
        a=new Array();
        a[0]=10
        a[1]=20
```

```
        a[2]=30
        a[3]=40
        a[4]=50
        document.write("<h4>The elements in array are ...</h4>");
        for(i=0;i<a.length;i++)
        document.write(a[i]+" ");
        document.write("<h4> Calling The push() method</h4>");
        a.push(60);
        document.write("<h4>The elements in array are ...</h4>");
        for(i=0;i<a.length;i++)
        document.write(a[i]+" ");
    </script>
</body>
</html>
```

Output

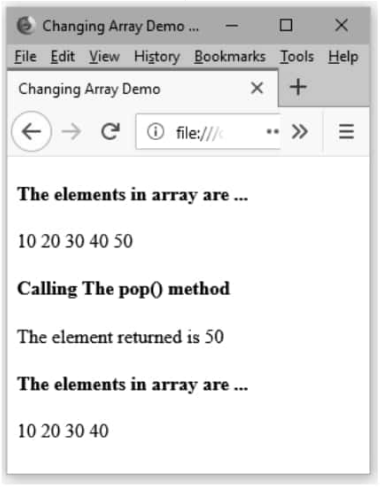
(3) Pop Method

This method returns and removes the last element of the array. For example

```
<!DOCTYPE html>
<html>
<head>
    <title>Changing Array Demo</title>
</head>
<body>
    <script type="text/javascript">
        a=new Array();
```

```
        a[0]=10
        a[1]=20
        a[2]=30
        a[3]=40
        a[4]=50
        document.write("<h4>The elements in array are ...</h4>");
        for(i=0;i<a.length;i++)
            document.write(a[i]+" ");
        document.write("<h4> Calling The pop() method</h4>");
        var val=a.pop();
        document.write("The element returned is "+val)
        document.write("<h4>The elements in array are ...</h4>");
        for(i=0;i<a.length;i++)
            document.write(a[i]+" ");
    </script>
</body>
</html>
```

Output



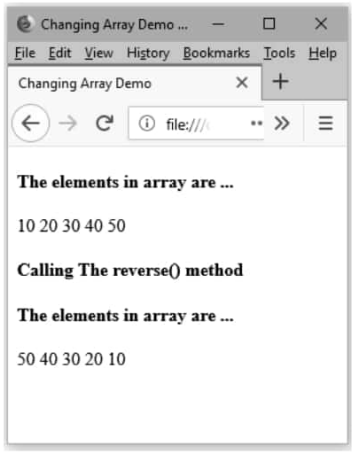
(4) Reverse Method

The reverse method is used to reverse the elements present in the array. For example

```
<!DOCTYPE html>
<html>
<head>
    <title>Changing Array Demo</title>
</head>
```

```
<body>
    <script type="text/javascript">
        a=new Array();
        a[0]=10
        a[1]=20
        a[2]=30
        a[3]=40
        a[4]=50
        document.write("<h4>The elements in array are ...</h4>");
        for(i=0;i<a.length;i++)
            document.write(a[i]+" ");
        document.write("<h4> Calling The reverse() method</h4>");
        a.reverse();
        document.write("<h4>The elements in array are ...</h4>");
        for(i=0;i<a.length;i++)
            document.write(a[i]+" ");
    </script>
</body>
</html>
```

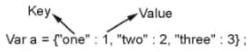
Output



2.1.9 Objects as Associative Array

Associative array is a specialized array in which the elements are stored in **(Key,value)** pair.

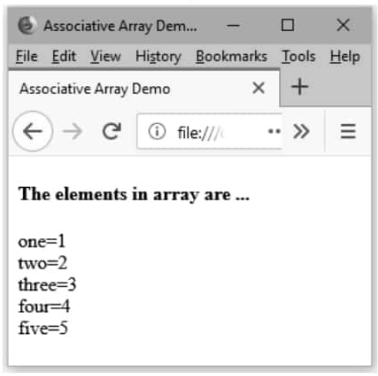
We can create associative array as



JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
    <title>Associative Array Demo</title>
</head>
<body>
    <script type="text/javascript">
        a=new Object();
        a["one"]=1;
        a["two"]=2;
        a["three"]=3;
        a["four"]=4;
        a["five"]=5;
        document.write("<h4>The elements in array are ...</h4>");
        for(i in a)
            document.write(i+"="+a[i]+"<br/>");
    </script>
</body>
</html>
```

Output



2.2 Function

- We can write the functions in the JavaScript for bringing the modularity in the script.
- Separate functions can be created for each separate task. This ultimately helps in finding the bug from the program efficiently.

2.2.1 Defining a Function

- We can define the function anywhere in the script either in head or body section or in both. But it is a standard practice to define the function in the head section and call that function from the body section.
- The keyword **function** is used while defining the function.
- The syntax for defining the function is

```
function name_of _function (arg1,arg2,…argn)
{
 …
  Statements
}
```

2.2.2 Writing a Function

Here is a simple illustration in which we have written a function **my_fun()**

JavaScript[FunDemo.html]

```
<!DOCTYPE html>
<html>
<head>
 <title>Function Demo</title>
 <script type="text/javascript">
  function my_fun()
  {
    document.write("This statement is within the function");
  }
</script>
</head>
<body>
<center>
  <script type="text/javascript">
    document.write("This statement is before a function call");
    document.write("<br>");
    my_fun(); //   call to the function
  </script>
</center>
</body>
</html>
```

Definition of function

Output

Script Explanation

The above code is pretty simple. We have defined one function named **my_fun** in the **head** section of the HTML document and called it from the **body** section. The corresponding **write** statements are used to display the messages on the browser window.

2.2.3 Adding an Arguments

- We can pass some arguments to the function.
- The syntax is

```
function function_name(aregument1, argument2…,argumentn)
{
//body of function
}
```

2.2.4 Scope of Variable and Argument

- Scope is the block or area of program in which particular variable or argument is accessible.
- The scope is defined using two types of variables - **Local Scope** and **Global Scope.**

Local Scope

If a variable is defined inside a function then that variable is a local variable and its scope is a local scope. That also means, that the local variable is accessible only within a function in which it is defined. It is not accessible outside that function.

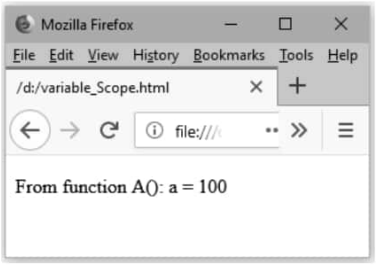
Following program shows the use of local variable.

JavaScript Example for Demonstrating scope of Local Variable

```
<!DOCTYPE html>
<html>
 <head>
    <script type="text/javascript">
       function A()
       {
```

```
            var a=100//local variable
            document.getElementById("id1").innerHTML=
            "From function A(): a = "+a
        }
        function B()
        {
            document.getElementById("id2").innerHTML=
            "From function B(): a = "+a
        }
    </script>
</head>
<body>
<p id="id1"></p>
<p id="id2"></p>
<script type="text/javascript">
A()
B()
</script>
</body>
</html>
```

Output



Script Explanation : In above JavaScript,

(1) We have created two functions namely A() and B().

(2) Inside the **A()** function variable a is declared and initialized to the value 100. The value of this variable is displayed using the element …

(3) Inside the **B()** function, we are trying to display the value of same variable a using … . But note that the output does not display value of variable a through function B(). This is because variable a is a local variable declared in function A(). Hence its scope is upto the function A(). Outside this function, the variable **a** is undefined.

Global Variable

A variable is called global variable , if it is defined outside the function. The variable having global scope is accessible by any function.

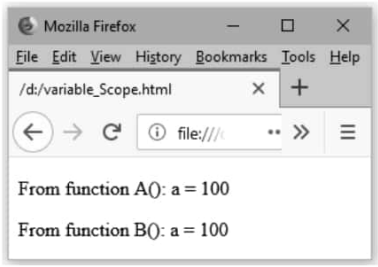
Following example illustrates the use of global variable.

JavaScript Example for Demonstrating the scope of Global Variable

```
<!DOCTYPE html>
<html>
    <head>
        <script type="text/javascript">
            var a=100//Global variable
            function A()
            {
                document.getElementById("id1").innerHTML
                ="From function A(): a = "+a
            }
            function B()
```

```
            {
                document.getElementById("id2").innerHTML
                ="From function B(): a = "+a
            }
        </script>
    </head>
    <body>
    <p id="id1"></p>
    <p id="id2"></p>
    <script type="text/javascript">
    A()
    B()
    </script>
    </body>
</html>
```

Output



2.3 Calling a Function

2.3.1 Calling a Function with Argument

In JavaScript, we can pass an argument to the function. Following JavaScript shows how to pass an argument to the function. In the following program I have written a simple function for addition of two numbers. The values of two numbers are 10 and 20 respectively. Inside this function the addition of these two arguments is carried out and result is displayed on the Browser page.

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        function add(a,b)//function to which arguments a and b are  passed
        {
            c=a+b
            document.write("Addition = "+c)
        }
    </script>
</head>
<body>
    <h4> Passing Arguments to the function </h4>
    <script type="text/javascript">
        var x=10;
        var y=20
        add(x,y)
    </script>
</body>
</html>
```

2.3.2 Calling a Function without an Argument

A function can also be called without passing any argument. In this case, all the required variables are declared and used within that function. Following program shows how to call a function without passing an argument.

JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        function add()
        {
            var a=10
            var b=20
            c=a+b
            document.write("Addition = "+c)
        }
    </script>
</head>
<body>
    <h4> Function without passing argument</h4>
    <script type="text/javascript">
    add()
    </script>
</body>
</html>
```

Output

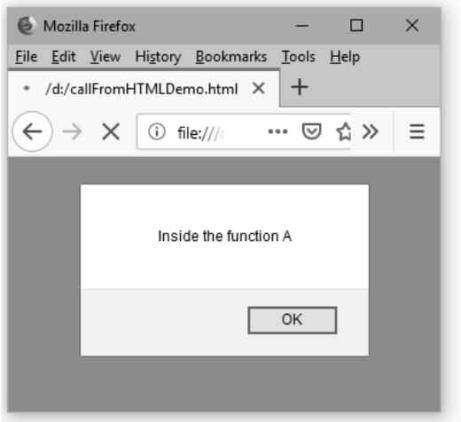
2.3.3 Calling Function from HTML

For calling a function from HTML normally, JavaScript events are used. For example –

JavaScript Document

```
<!DOCTYPE html>
<html>
    <head>
        <script type="text/javascript">
            function A()
            {
                alert("Inside the function A");
            }
        </script>
    </head>
    <body onload="A()">
    </script>
    </body>
</html>
```

Output

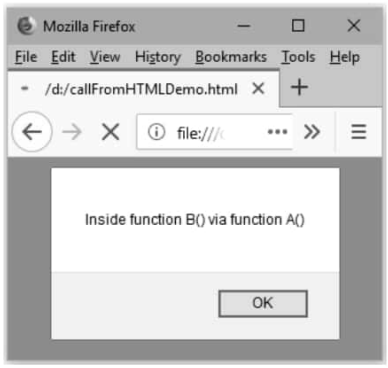


2.3.4 Function Calling another Function

We can call one function from another function. This is called nested functions. In the following code, there is a call for function B() from function A().

```
<!DOCTYPE html>
<html>
    <head>
        <script type="text/javascript">
            function A()
            {
                B();//calling another function
            }
            function B()
            {
                alert("Inside function B() via function A()");
            }
        </script>
    </head>
    <body onload="A()">
    </script>
    </body>
</html>
```

Output



2.3.5 Returning a Value from Function

- We can return some value from the function using a keyword **return.**
- This return value is either stored in variable or directly displayed on the browser window.
- Following is a simple illustration in which the value is returned from the function and is directly displayed on the browser window using **document.write.**

JavaScript[FunRetDemo.html]

```
<!DOCTYPE html>
<html>
<head>
 <title>Function Demo</title>
 <script type="text/javascript">
  function my_fun()
  {
    str="I am function returned value";
    return str;
  }
</script>
</head>
<body>
<center>
 <script type="text/javascript">
   document.write("This statement is before a function call");
   document.write("<br>");
   document.write("<h3>"+my_fun()+"<h3>");
 </script>
</center>
</body>
</html>
```

Output

2.4 String

- String is a collection of characters.
- Some commonly used methods of string object are concatenating two strings, converting the string to upper case or lower case, finding the substring of a given string and so on.
- The string is written within the quotes- either single quote or double quote. For example –

```
var myname='ABC'
```

2.4.1 Manipulating a String

Manipulating a string means, changing one string to another, joining two strings, changing the string from upper case to lower case or from lower case to upper.

There are some commonly used methods of string

Method	Meaning
concat(str)	This method concatenates the two strings. For example s1.concat(s2) will result in concatenation of string s1 with s2.
charAt(index_val)	This method will return the character specified by value index_val.
substring(begin,end)	This method returns the substring specified by begin and end character.
toLowerCase()	This function is used to convert all the uppercase letters to lower case.
toUpperCase()	This function is used to convert all the lowercase letters to upper case.
valueOf()	This method returns the value of the string.

There is one important property of string object and that is **length.** For example

```
var my_str="Hello";
var len;
len=my_str.length;
```

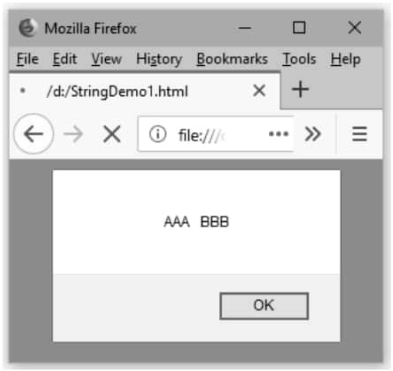
Length of the string "Hello" will be stored in the variable len

2.4.2 Joining a String

We can join two strings using + operator. For example –

```
<!DOCTYPE html>
<html>
    <script type="text/javascript">
    var Firstname="AAA"
    var Lastname="  BBB"
    var name=Firstname+Lastname
    alert(name);
    </script>
    </body>
</html>
```

Output



2.4.3 Receiving a Character from Given Position

The **charAt()** is a method that returns the character from the specified index. The index of a first character is 0; the second character is 1, and so on. Javascript characters in a string are indexed from left to right.

If no index is provided to **charAt()** method, then the default is 0. Following program illustrates how to receive character from given position.

JavaScript Document

```
<!DOCTYPE html>
<html>
    <script type="text/javascript">
    var str="ABC"
    alert("The character at first position of string: "+str+" is: "+str.charAt(0))
    </script>
    </body>
</html>
```

Output

2.4.4 Retrieving a Position of Character in a String

- For finding the position of particular character we use indexOf function.
- It returns the position of the first occurrence of a specified value in a string.
- This method is case sensitive.
- **Example**

```
<!DOCTYPE html>
<html>
        <script type="text/javascript">
        var str="I love India - my country"
        var n=str.indexOf("India");
        alert("The word 'India' is from index: "+n+" in the text: "+str)
        </script>
        </body>
</html>
```

Output

2.4.5 Dividing Text

- Any text is made up of words. We can divide the text into a collection of these words. For example – User may enter his/her complete name as first name, middle name and surname. By dividing the text, we can get firstname, middle name and surname separated.
- Usually **split()** function is used to divide the text.
- The split() method creates a new array and then copies portions of the string, called a substring, into its array element.
- **Syntax :** The split method is used in the following manner –
  var newStrArray=textname.split(delimiter character)

JavaScript Document

```
<!DOCTYPE html>
<html>
        <script type="text/javascript">
        var Name="Shivaji Shahaji Bhosale"
        var list=Name.split(' ')
        alert("First Name: "+list[0]+"\nMiddle Name: "+list[1]+"\nLast Name: "+list[2])
        </script>
        </body>
</html>
```

Output

2.4.6 Copying a Substring

There are two useful methods for copying a substring – i) substring() and ii) substr()

1. Use of substring

The syntax of substring is

substring(start, end)

where start indicates the starting index and end indicates the ending index for extracting substring.

Following program shows use of substring() function for retrieving the substring and then copying it to some other string at the end.

```
<!DOCTYPE html>
<html>
    <script type="text/javascript">
    var Name1="Shivaji Shahaji Bhosale"
    var Name2="Sambhaji"
    var sub_str=Name1.substring(16,23)
    document.write("<br/>The Name1: "+Name1)
    document.write("<br/>The Surname: "+sub_str)
    document.write("<br/>The Name2: "+Name2)
    document.write("<br/><b>Copying the surname for Name2...</b>")
    var Name3=Name2+" "+sub_str
    document.write("<br/>The Name3: "+Name3)
    </script>
    </body>
</html>
```

Output

2. Use of substr

The **substr** method is also used to extract the substring from the text. The syntax for substr is as follows

```
substr(start,length)
```

where start indicates the starting index and length indicates the number of characters to be extracted.

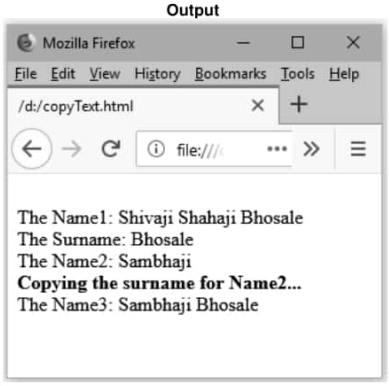
Following example shows how to extract some substring using **substr()** method and then copy this substring to another text.

JavaScript Document

```
<!DOCTYPE html>
<html>
    <script type="text/javascript">
    var Name1="Shivaji Shahaji Bhosale"
```

```
    var Name2="Sambhaji"
    var sub_str=Name1.substr(16,7)
    document.write("<br/>The Name1: "+Name1)
    document.write("<br/>The Surname: "+sub_str)
    document.write("<br/>The Name2: "+Name2)
    document.write("<br/><b>Copying the surname for Name2...</b>")
    var Name3=Name2+" "+sub_str
    document.write("<br/>The Name3: "+Name3)
    </script>
    </body>
</html>
```

Output



2.4.7 Converting String to Number and Number to String

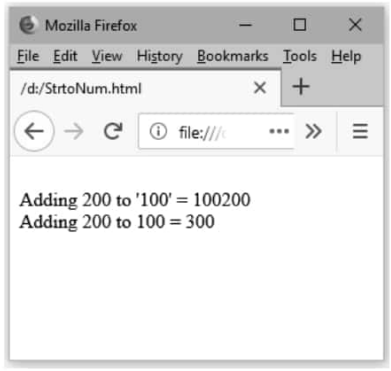
1. Converting String to Number

For converting string to number we have different types of functions based on type of numbers – That means if string is to be converted to integer(a number without decimal point) then the method **parseInt()** is used. If string is to be converted to float value then the method **parseFloat()** is used.

Example

```
<!DOCTYPE html>
<html>
    <script type="text/javascript">
    var str="100";
    var a=parseInt(str)//converting string 100 to number 100
    var b=200;
    result=str+b
    document.write("<br/>Adding 200 to '100' = "+result)
    result=a+b
    document.write("<br/>Adding 200 to 100 = "+result)
    </script>
    </body>
</html>
```

Output



2. Converting Number to String

Using **toString()** method we can convert a number to string. For example

```
<!DOCTYPE html>
<html>
    <script type="text/javascript">
    var num=100;
    var str=num.toString()
    document.write("<br/>The 100 value is converted to string "+str)
    </script>
    </body>
</html>
```

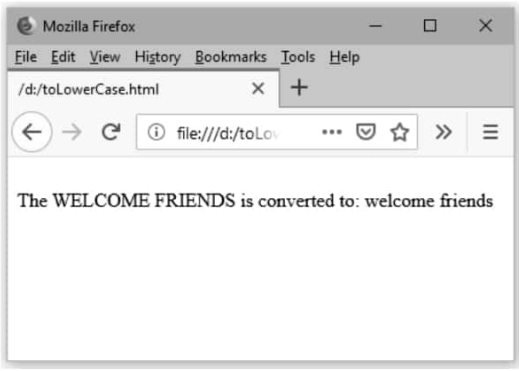
Output

2.4.8 Changing Case of String

Using toLowerCase(), we can convert the capital letters in the string to small letter. For example

```
<!DOCTYPE html>
<html>
    <script type="text/javascript">
    var str="WELCOME FRIENDS";
    var newstr=str.toLowerCase()
    document.write("<br/>The "+str+" is converted to: "+newstr)
    </script>
    </body>
</html>
```

Output

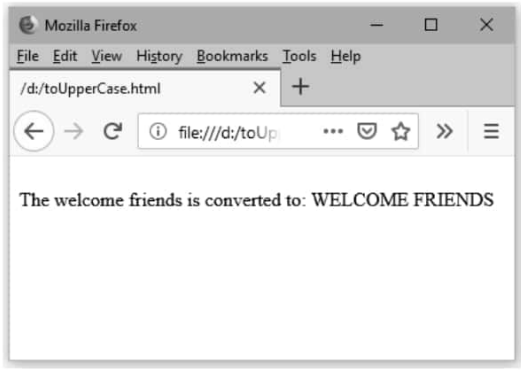


Using **toUpperCase()** method we can convert the small letters to capital letters.

For example –

```
<!DOCTYPE html>
<html>
    <script type="text/javascript">
    var str="welcome friends";
    var newstr=str.toUpperCase()
    document.write("<br/>The "+str+" is converted to: "+newstr)
    </script>
    </body>
</html>
```

Output

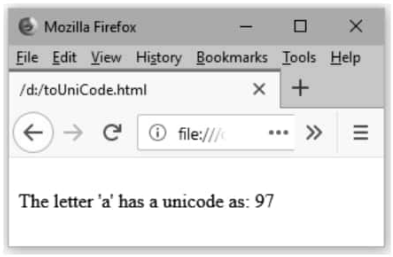


2.4.9 Finding Unicode of a Character

- Computer can not understand characters. It understands only numbers. Hence whenever user types some letter, it gets converted automatically to a number called **Unicode number.** The computer can understand this Unicode number only.
- Unicode is a standard that assigns a number to every character, number, and symbol that can be displayed on a computer screen.
- The **charCodeAt()** is a methods that returns Unicode of a string. For example –

```
<!DOCTYPE html>
    <html>
    <script type="text/javascript">
    var ch='a'
    var n=ch.charCodeAt()
    document.write("<br/>The letter '"+ch+"' has a unicode as: "+n)
    </script>
    </body>
</html>
```

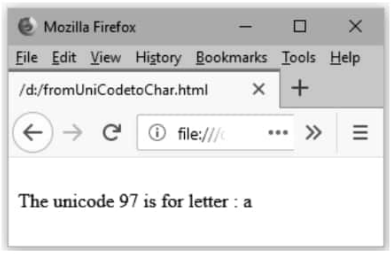
Output



- Similarly,we can obtain the letter or character from a Unicode using the method **fromCharCode().** For example

```
<!DOCTYPE html>
<html>
    <script type="text/javascript">
    var n=97
    var ch=String.fromCharCode(n)
    document.write("<br/>The unicode "+n+" is for letter : "+ch)
    </script>
    </body>
</html>
```

Output



Review Questions

1. *Define array.*
2. *How to declare array in JavaScript.*
3. *Write a JavaScript to define the array elements and to find the length of array.*
4. *Write a JavaScript for sorting the array elements.*
5. *Explain the use of join() function with illustrative example.*
6. *Explain the use of push and pop functions.*
7. *Write a JavaScript to reverse the elements of array.*
8. *Write a short note - object as associative array in JavaScript.*
9. *Write a syntax for defining the function.*
10. *Explain the scope of variable with the help of programming example.*
11. *Explain the method of calling a function from HTML.*
12. *What is string ? Enlist some commonly used methods for manipulating string.*
13. *How to retrieve a position of desired word from the string.*
14. *Write a JavaScript to convert a string to number.*

❑❑❑

UNIT - III

3 Form and Event Handling

3.1 Basics of a Form

3.1.1 Building Blocks of Form

- Form is a typical layout on the web page by which a user can interact with the web page.
- Typical component of forms are **text, text area, checkboxes, radio buttons and push buttons.** These components of form are also called as **form controls or controls.**
- HTML allows us to place these form components on the web page and send the desired information to the destination server.
- All these form contents appear in the **<form>** tag. The form has an **attribute action** which gets executed when user clicks a button on the form.

Uses of Form

Forms are used in following manner

(1) Forms are used to collect the information from customer or students for online registrations

(2) Forms are used for online survey.

(3) Forms are used for conducting online examination, for getting feedbacks and so on.

(4) The information present in the form is submitted to the server for further processing

3.1.2 Properties and Methods of Form

- The commonly used properties and methods of the form are enlisted in the following table

Attribute	Description
action	It specifies the url where the form should be submitted.
method	It specifies the HTTP methods such as GET, POST get: Default. Appends the form-data to the URL in name/value pairs: URL?name=value&name=value post: Sends the form-data as an HTTP post transaction
name	This attribute denotes the name of the form.
target	It specifies the target of the address in the action attribute The target values can be as follows - _blank: Opens in a new window _self: Opens in the same frame as it was clicked (default)

_parent	Opens in the parent frameset
_top	Opens in the full body of the window
framename	Opens in a named frame

The form can be written inside as <body> tag as follows –

```
<body>
    <form name="myform" action="/myServerPage.phtp" method="GET" target="_blank">
        //code for placing form controls here
        …
        …
        …
    </form>
</body>
```

3.1.3 Text

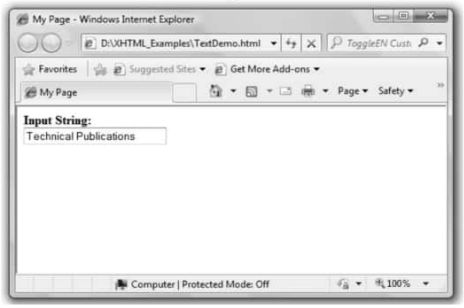
- **Text** is typically required to place one line text. For example if you want to enter some name then it is always preferred to have Text field on the form.
- The text field can be set using
  <input type="text" size="30" name ="username" value=" ">
  The input type is **text** and the **value** of this text field is " " That means the blank text field is displayed initially and we can enter the text of our choice into it. There is **size** parameter which allows us to enter some size of the text field.
- Some other parameters or attributes can be
  - **maxlength** that allows us to enter the text of some maximum length.
  - **name** indicates name of the text field.
  - **align** denotes the alignment of the text in the text field. The alignment can be left, right, bottom and top.

Example Code

HTML Document [TextDemo.html]

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Page</title>
    </head>
    <body>
    <form>
        <b>Input String:</b><br/><input type="text" size="25" value="">
    </form>
    </body>
</html>
```

Output



Script Explanation :

In above document

1) We have the label "Input String" just before the **<input>** tag. We can also specify the label by using the **<label>** tag as follows -
   <label>Input String:
<input type="text" size="25" value=""></label>
2) Thus the label gets bound to the text box. This aspect is always beneficial for a web programmer because using label control we can focus on the corresponding text box contents.
3) Initially the text box field is blank. We can type some text inside this text box.

Ex. 3.1.1 : ***How will you create password field in a HTML form ?***

Sol. : The password field is typically created on the form. Hence following code can be written to create it -

```
<form name="form1">
 Password:<input type="password"/>
</form>
```

3.1.4 Text Area

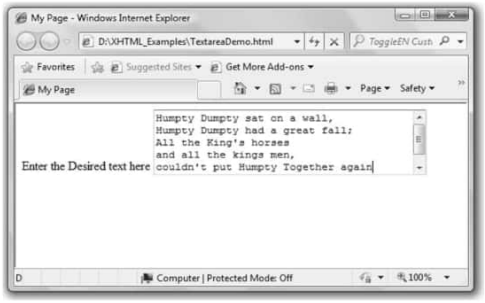
Text field is a form component which allows us to enter single line text, what if we want to have multiple line text? Then you must use textarea component.

HTML Document [TextareaDemo.html]

```
<!DOCTYPE html>
<html>
<head>
    <title>My Page</title>
</head>
<body>
    <form>
        Enter the Desired text here
        <textarea cols="40" rows="5" name="myname">
        </textarea>
        </form>
```

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

Output



Various parameters that can be set for the text area can be

- **row** denotes total number of rows in the text area.
- **col** specifies total number of columns in the text area.
- **name** denotes the name of the text area which can be utilised for handling that component for some specific purpose.
- **wrap** can be **virtual** or **physical.** If the **wrap** is virtual then the line breaks get disappeared when the text is actually submitted to the server. But if the wrap is assigned to the **physical** then the line breaks (if any) appear as it is in the text.

3.1.5 Checkbox

- It is the simplest component which is used particularly when we want to make some selection from several options.
- For having the checkbox we have to specify the input type as **checkbox.** For example
  `<input type="checkbox" name="option1" value="mango" checked="checked">Mango<br/>`
- If we want to get the checkbox displayed as checked then set **checked="checked"**

Example Code

HTML Document[ChkBoxDemo.html]

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Form with Check box</title>
    </head>
    <body>
        <form name ="checkboxForm">
        <div align="center"><br>
        Select the fruit(s) of your choice<br/>
```

```
        <input type="checkbox" name="option1" value="mango" checked="checked">Mango<br/>
        <input type="checkbox" name="option2" value="apple">Apple<br/>
        <input type="checkbox" name="option3" value="guava">Guava<br/>
        </div>
        </form>
    </body>
</html>
```

Output

Script Explanation

1) In the above program to set some checkbox in checked state we can mention the attribute **checked** as **checked.**
2) We can set the **value** attribute as " " but this then the checkbox will not get associated with any value. The **Mango, Apple** and **Guava** are the labels of the checkboxes.

3.1.6 Radio Button

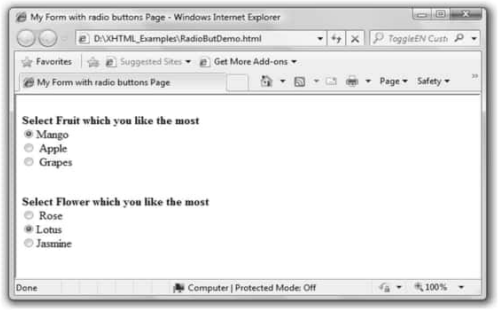
- This form component is also use to indicate the selection from several choices.
- Using **input type="radio"** we can place radio button on the web page.
- This component allows us to make only one selection at a time.
- We can create a group of some radio button component.
- Following HTML document displays the radio buttons for two different groups.

HTML Document[RadioButDemo.html]

```
<!DOCTYPE html>
<html >
        <head>
            <title>My Form with radio buttons Page</title>
        </head>
        <body>
        <form name="myform">
            <div align="left"><br>
            <b>Select Fruit which you like the most</b><br/>
```

```
        <input type="radio" name="group1" value="Mango">Mango<br/>
        <input type="radio" name="group1" value="Apple" checked> Apple<br/>
        <input type="radio" name="group1" value="Grapes"> Grapes
        <br/><br/><br/>
        <b>Select Flower which you like the most</b><br/>
        <input type="radio" name="group2" value="Rose"> Rose<br/>
        <input type="radio" name="group2" value="Lotus">Lotus<br/>
        <input type="radio" name="group2" value="Jasmine" checked>Jasmine<br/>
        </div>
    </form>
  </body>
</html>
```

Output



Ex. 3.1.2 : ***What is the difference between group of checkbox buttons and group of radio buttons ?***

Sol. : • The checkbox and radio buttons are used for making the selection from a group of choices.

- When a user selects (checks) a checkbox, its value gets assigned as the current value of the checkbox group's control name.
- A checkbox group's control name may get paired with several current values if the user selects more than one checkbox.
- Radio buttons work just like checkboxes except they are typically set up to be mutually exclusive of one another, i.e. when one is selected, all the others are automatically deselected.

3.1.7 Button

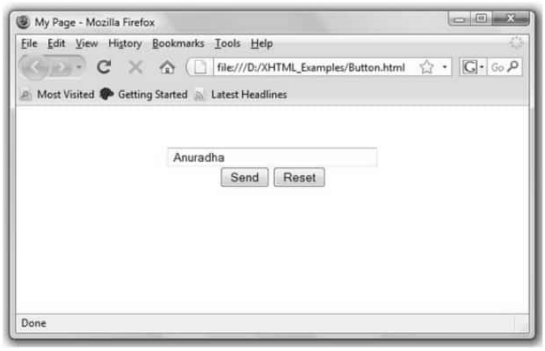
- We can create the button using **<input type ="button">** tag.
- There are two types of buttons that can be created in HTML. One is called **submit** button and the another one is **reset** button.
- Various parameters of submit button are
  1) **name** denotes the name of the submit button.
  2) **value** is for writing some text on the text on the button.
  3) **align** specifies alignment of the button.

Example Code

HTML Document[Button.html]

```
<!DOCTYPE html>
<html >
    <head>
    <title> My Page </title>
    </head>
    <body>
<form name="myform" action="http://www.localhost.com/cgi-bin/hello.cgi" method="POST">
            <div align="center">
            <br/><br/>
                <input type="text" size="35" value=" ">
                <br><input type="submit" value="Send">
                <input type="reset" value="Reset"><br>
        </div>
            </form>
        </body>
 </html>
```

Output



Script Explanation

1) In above HTML document, we have used the form whose name is "**myform**".
2) There are two attributes associated with the form tag and those are **action** and **method**. The **action** parameter indicates the address and the cgi script where the contents should go and **method** parameter is for the methods for submitting the data. The **method** can be **get** or **post**. Thus by specifying the action and method for a form we can send the desired data at desired location.

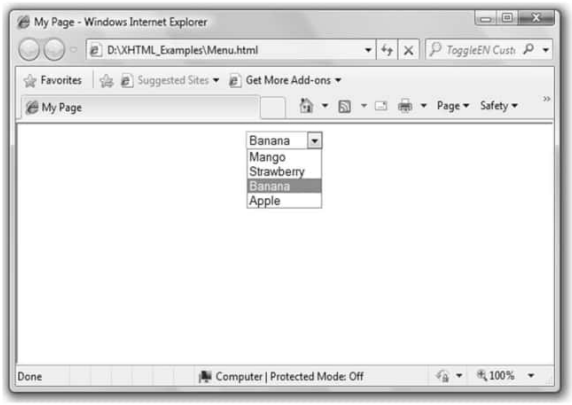
3.1.8 Select Element

- HTML allows us to have pop down menu on the web page so that the desired selection can be made.
- The parameter **select** is for the menu component and **option** parameter is for setting the values to the options of drop down menu.
- We can make some specific option selected by **selected value = .**
- In the following HTML document we have created one drop down menu in which various fruits are enlisted. By default "Banana" is set as selected.

HTML Document [Menu.html]

```
<!DOCTYPE html>
<html >
        <head>
            <title> My Page </title>
        </head>
        <body>
        <form name="myform">
        <div align="center">
            <select name="My_Menu">
                <option value="Mango">Mango</option>
                <option value="Strawberry">Strawberry</option>
                <option selected value="Banana">Banana </option>
                <option value="Apple">Apple</option>
            </select>
        </div>
        </form>
        </body>
</html>
```

Output



3.1.9 Examples on Form Design

Ex. 3.1.3 : *Create a HTML document that has the form with the following controls*

a) A text box to collect the customer name.

b) Four checkboxes, one for the following items :

i. Four HTML textbooks for ₹ 1000 *ii. Eight XML textbooks for ₹ 2000*

iii. Four Javabeans books for ₹ 2500 *iv. Eight UML textbooks for ₹ 1500*

c) A collection of radio buttons that are labelled as follows –

i. Cash ii. Cheque/DD iii. Creadit card.

Sol. :

```
<!DOCTYPE html>
<html >
 <body>
<center><h2>REGISTRATION FORM</h2></center>
<hr/>
<form>
<table>
<tr>
<td>Name:</td>
<td><input type="text" size="25"value=""></td>
</tr>
<tr>
<td>Select the desired Items:</td>
<td>
<select name="f">
<option value="4 HTML Books(Rs. 1000)">4 HTML Books(Rs. 1000)</option>
<option value="8 XML Books(Rs. 2000)">8 XML Books(Rs. 2000)</option>
<option value="4 JavaBeans Books(Rs.2500)">4 JavaBeans Books(Rs.2500)</option>
<option value="8 UML Books(Rs.1500)">8 UML Books(Rs.1500)</option>
</select>
</td>
</tr>
<tr>
<td>
Mode of Payment:
</td>
</tr>
<tr>
<td><input type="radio" name="group1" value="Cash">Cash</td>
</tr>
<tr>
<td><input type="radio" name="group1" value="Cheque/DD">Cheque/DD</td>
</tr>
<tr>
```

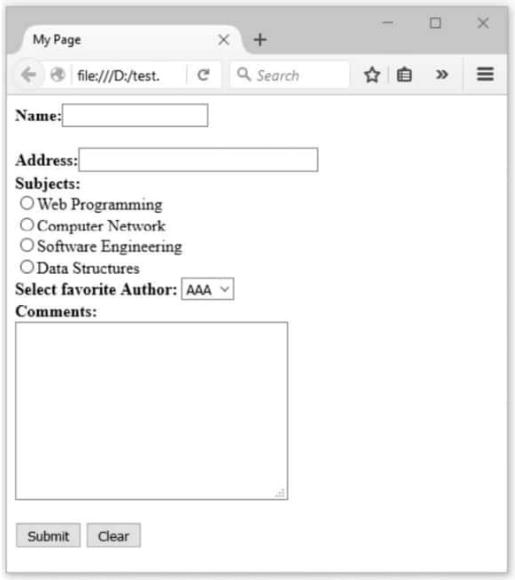
```
<td><input type="radio" name="group1" value="Credit Card">Credit Card</td>
</tr>
<tr></tr><tr></tr><tr></tr>
<td><input type="submit" value="Submit"></td>
<td><input type="reset" value="Reset"></td>
</tr>
</table>
</form>
</body>
```

Ex. 3.1.4 : *Write a form to collect details of a user such as name, address, radio button to choose subject of book he wants to buy, dropdown to choose favorite author, comments for the last book he read.*

Sol. :

```
<!DOCTYPE html >
<html >
 <head>
  <title>My Page</title>
</head>
  <body>
      <form>
<b>Name:</b><input type="text" size="20" value=""><br/>
     <b>Address:</b><input type="text" size="35" value=""><br/>
     <b>Subjects:</b><br/>
     <input type="radio" name="authors" value="Web Programming">Web Programming<br/>
     <input type="radio" name="authors" value="Computer Network">Computer Network<br/>
     <input type="radio" name="authors" value="Software Engineering">Software Engineering<br/>
     <input type="radio" name="authors" value="Data Structures">Data Structures<br/>
     <b>Select favorite Author:</b>
     <select name="MyMenu">
      <option value="AAA">AAA</option>
      <option value="BBB">BBB</option>
      <option value="CCC">CCC</option>
      <option value="DDD">DDD</option>
     </select>
     </br>
     <b>Comments:</b></br>
     <textarea cols="30" rows="10" name="comments">
     </textarea>
     <br/><br/>
     <input type="submit" value="Submit"/>
     <input type="reset" value="Clear"/>
     </form>
     </body>
</html>
```

Output



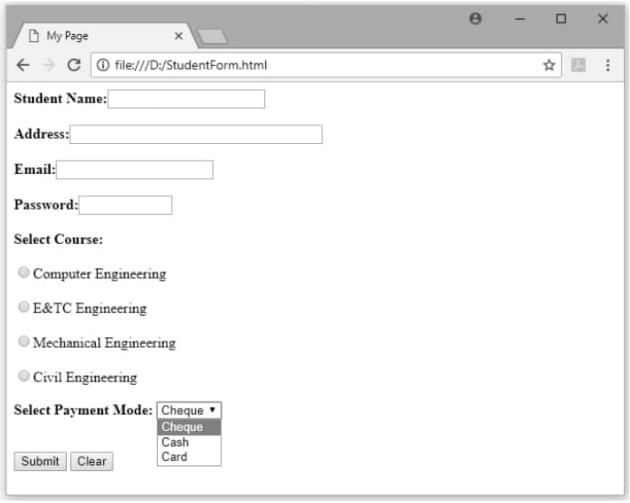
Ex. 3.1.5 : *Design a simple HTML form for filling the information for registration of a student.*

Sol. : HTML Form for Student Registration

```
<!DOCTYPE html>
<html >
 <head>
  <title>My Page</title>
</head>
  <body>
    <form>
        <b>Student Name:</b><input type="text" size="20" value=""><br/><br/>
        <b>Address:</b><input type="text" size="35" value=""><br/><br/>
        <b>Email:</b><input type="text" size="20" value=""><br/><br/>
        <b>Password:</b><input type="password" size="10" value=""><br/><br/>
        <b>Select Course:</b><br/><br/>
        <input type="radio" name="courses" value="Computer Engineering">Computer Engineering<br/><br/>
        <input type="radio" name="courses" value="E&TC Engineering">E&TC Engineering<br/><br/>
        <input type="radio" name="courses" value="Mechanical Engineering">Mechanical
Engineering<br/><br/>
        <input type="radio" name="courses" value="Civil Engineering">Civil Engineering<br/><br/>
        <b>Select Payment Mode:</b>
```

```
        <select name="MyMenu">
         <option value="Cheque">Cheque</option>
         <option value="Cash">Cash</option>
         <option value="Card">Card</option>
        </select>
        <br/><br/><br/>
        <input type="submit" value="Submit"/>
        <input type="reset" value="Clear"/>
        </form>
        </body>
    </html>
```

Output



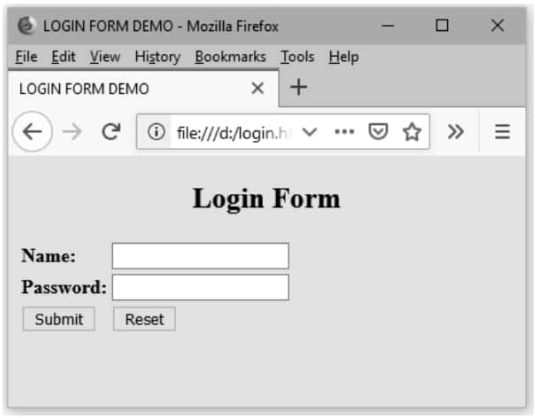
Ex. 3.1.6 : *Write a form to make login and password.*

Sol. :

```
<html>
    <head>
        <title>LOGIN FORM DEMO</title>
    </head>
    <body bgcolor="khaki">
        <center>
            <h2>Login Form</h2>
        </center>
        <form name="form1">
            <table>
                <tr>
                <td><b>Name:</b></td>
```

```
                <td><input type="text" name="userName">
                </tr>
                <tr>
                <td><b>Password:</b></td>
                <td><input type="password"name="pwd"></td>
                </tr>
                <tr>
                <td><input type="submit" name="submit" value="Submit"></td>
                <td><input type="reset" name="reset" value="Reset"></td>
                </tr>
            </table>
        </form>
    </body>
    </html>
```

Output



3.2 Form Events

- Event is an activity that represents a change in the environment. For example mouse clicks, pressing a particular key of keyboard represent the events.
- A JavaScript event is an **action** that can be detected by JavaScript. Many of them are **initiated by user actions** but some are generated by the browser itself. We say then that an event is **triggered** and then it can be caught by JavaScript functions, which then do something in **response**.
- **Event handler** is a script that gets executed in response to these events. Thus event handler enables the web document to respond the user activities through the browser window.
- Events are specified in lowercase letters and these are case-sensitive.

- Commonly used events and tag attributes are enlisted in the following table -

Events	Intrinsic event attribute	Meaning	Associated tags
blur	Onblur	Losing the focus.	<button> <input> <a> <textarea> <select>
change	onchange	On occurrence of some change.	<input> <textarea> <select>
click	onclick	When user clicks the mouse button.	<a> <input>
dblclick	ondblclick	When user double clicks the mouse button.	<a> <input> <button>
focus	onfocus	When user acquires the input focus.	<a> <input> <select> <textarea>
keyup	onkeyup	When user releases the key from the keyboard.	Form elements such as input,button,text,textarea and so on.
keydown	onkeydown	When user presses the key down	Form elements such as input,button,text,textarea and so on.
keypress	onkeypress	When user presses the key.	Form elements such as input,button,text,textarea and so on.
mousedown	onmousedown	When user clicks the left mouse button.	Form elements such as input,button,text,textarea and so on.
mouseup	onmouseup	When user releases the left mouse button.	Form elements such as input,button,text,textarea and so on.
mousemove	onmousemove	When user moves the mouse.	Form elements such as input,button,text,textarea and so on.
mouseout	onmouseout	When the user moves the mouse away from some element	Form elements such as input,button,text,textarea and so on.
mouseover	onmouseover	When the user moves the mouse away over some element	Form elements such as input,button,text,textarea and so on.
load	onload	After getting the document loaded.	<body>
reset	onreset	When the reset button is clicked.	<form>

submit	onsubmit	When the submit button is clicked.	<form>
select	onselect	On selection.	<input> <textarea>
unload	onunload	When user exits the document.	<body>

Table 3.2.1

Example of Event Handling

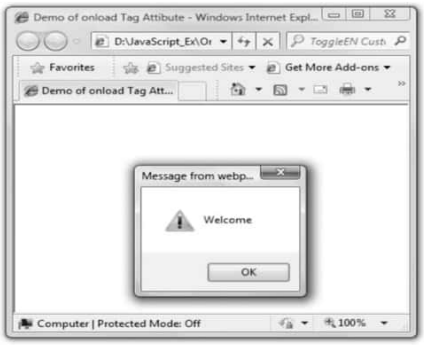
- To understand how events works in JavaScript let us put some form components on the JavaScript.
- The **onload** event gets activated as soon as the web page gets loaded in the browser's window. Following script along with its output illustrate the **onload** tag attribute.

JavaScript[OnloadDemo.html]

```
<!DOCTYPE html>
<html >
 <head>
  <title>Demo of onload Tag Attibute</title>
  <script type="text/javascript">
   function my_fun()
   {
   //This message will be displayed on page loading

    alert("Welcome");
   }
     </script>
</head>
<body onload="my_fun()">
</body>
</html>
```

When web document gets loaded on the browser window then **my_fun()** will be called

Output



3.2.1 Mouse Event

- Mouse Events are used to capture the interactions made by user using mouse. Various commonly used mouse Events are described in the following table

Event	Description
onclick	The mouse was clicked on an element.
ondblclick	The mouse was double clicked on an element.
onmousedown	The mouse was pressed down over an element.
onmouseup	The mouse was released over an element.
onmouseover	The mouse was moved (not clicked) over an element.
onmouseout	The mouse was moved off of an element.
onmousemove	The mouse was moved while over an element.

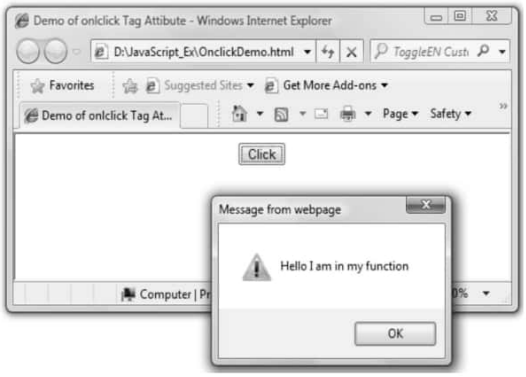
For example -

- Following is a simple JavaScript in which on the button click we have called a function my_fun(). This is a simple function in which we have displayed some message using alert popup box.

JavaScript[OnclickDemo.html]

```
<!DOCTYPE html >
<html >
 <head>
  <title>Demo of onlclick Tag Attibute</title>
<script type="text/javascript">
 function my_fun()
 {
   alert("Hello I am in my function");
 }
</script>
</head>
<body>
 <center>
 <form>
   <input type="button" value="Click" onclick="my_fun()">
 </form>
</center>
</body>
</html>
```

Output

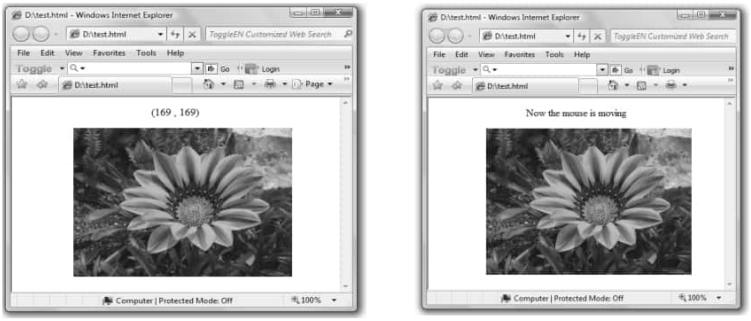


Ex. 3.2.1 : *Insert an image into a web page. Write a script which displays the message when the mouse is over the image. The co-ordinates of the mouse should be displayed if click attempted on the image.*

Sol. :

```
<html>
<head>
<script type="text/javascript">
    function my_fun1()
    {
        points.innerText="Now the mouse is moving";
    }
    function my_fun2()
    {
        points.innerText="("+event.offsetX+" , "+event.offsetY+")";
    }
</script>
</head>
    <body onmousemove="my_fun1()" onmousedown="my_fun2()">
        <center>
            <span id="points">(0,0)</span>
            <img src="d:\flower1.gif" style="position:absolute;top:50;left:90">
        </center>
    </body>
</html>
```

Output



3.2.2 Key Event

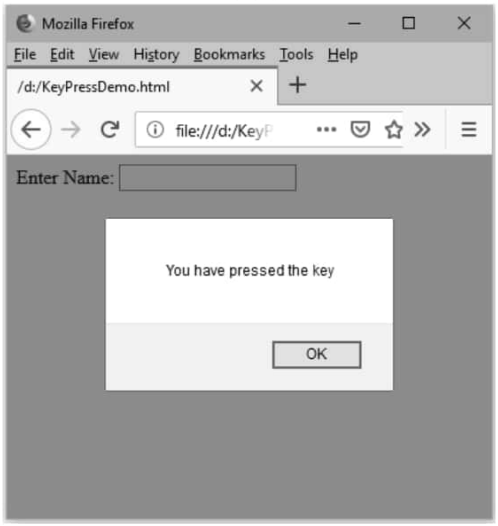
- The keyboard events are the events that occur when the user interacts using the keyboard. Various types of keyboard events are enlisted in the following table -

Event	Description
onkeydown	When user presses the key on the keyboard.(this happens first)
onkeypress	The user presses a key(this happens after onkeydown).
onkeyup	The user releases a key that was down.(this happens last).

JavaScript Example

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        function MyKeyfunction()
        {
            alert("You have pressed the key");
        }
    </script>
 </head>
 <body>
    <form>
        Enter Name: <input type="text" onkeypress="MyKeyfunction()"/>
    </form>
</body>
</html>
```

Output



3.3 Form Objects and Elements

- Website contains various objects. The very first object which we see is **window**.
- The **window** object contains the HTML document which is called as **document** object. This **document** object has several important functionalities. Out of which, the most commonly used functionality is **write()** method. We can display any message on Web page using following statement

```
document.write("Hello, How are you?");
```

- Note that one can omit the use of **window**. That means instead of writing **window.document.write** if we write **document.write** then it is perfectly allowed.
- A **form** is placed on **document**. The form objects are stored in the array called **forms**. They appear in the order in which the forms appear in the document.
- Each form can be refereed by using the index of the **forms** array. For example- suppose we want to refer second form then we write

```
document.forms[1]
```

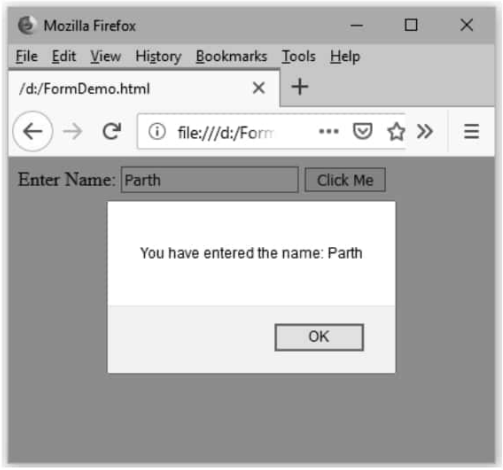
- A form can be referred by its **name**. For instance – if a form has a name **form1** then we can refer this form as document.forms.form1

  JavaScript Example : Following is a program in which we can refer the form element **text** and display the value in **text box**

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

```
                <script type="text/javascript">
                    function Myfunction()
                    {
                        alert("You have entered the name: "
                        +document.forms.myform.username.value);
                    }
                </script>
            </head>
<body>
<form name="myform">
        Enter Name: <input type="text" name="username"/>
        <input type="button" value="Click Me" onclick="Myfunction()"/>
</form>
</body>
</html>
```

Output



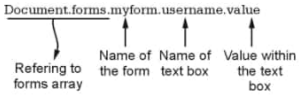
Script Explanation : In above JavaScript,

(1) We have created a form within <body> </body> section. On this form there are two elements - text field and a button.

(2) The text field has a name 'username'.

(3) On the button click, we triggered an event **onclick** and this event can be handled using the function named **Myfunction.**

(4) The **Myfunction** is defined in the <head> </head> section.This function refers to the text field value by the statement



(5) Using the alert popup box the text typed within the textbox will be displayed.

We can modify the above given Myfunction as follows for efficient use

```
function Myfunction()
{
with(document.forms.myform.username)
    {
        alert("You have entered the name: "+value);
    }
}
```

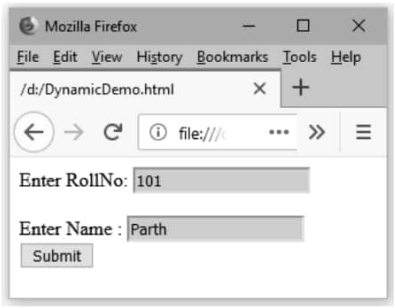
3.4 Changing Attribute Value Dynamically

- It is possible to change the attributes of the form elements dynamically. That means – during form feeling process itself, the color or font of the text field can be changed. This dynamic change helps the user to notify the importance changes in the form fields.
- Following is an example of changing the attribute value dynamically. In this example we are changing the background color of the text box as the user enters some data in it.

JavaScript Example

```
<!DOCTYPE html>
<html>
<head>
        <script type="text/javascript">
            function ChangeMe(Ele)
            {
                Ele.style.background='pink'
            }
        </script>
</head>
<body>
<form name="myform">
        Enter RollNo: <input type="text" name="roll" onchange="ChangeMe(this)"/>
        <br/><br/>
        Enter Name : <input type="text" name="name" onchange="ChangeMe(this)"/>
        <br/>
        <input type="submit" value="Submit"/>
</form>
</body>
</html>
```

Output



3.5 Changing Option List Dynamically

- Option list represents the list of one or more than one items which can be chosen by the user. In a web application, it is a common practice to change the contents of the option list base on some category chosen.
- That means JavaScript allows to change the items present in the list dynamically. Following example illustrate this idea

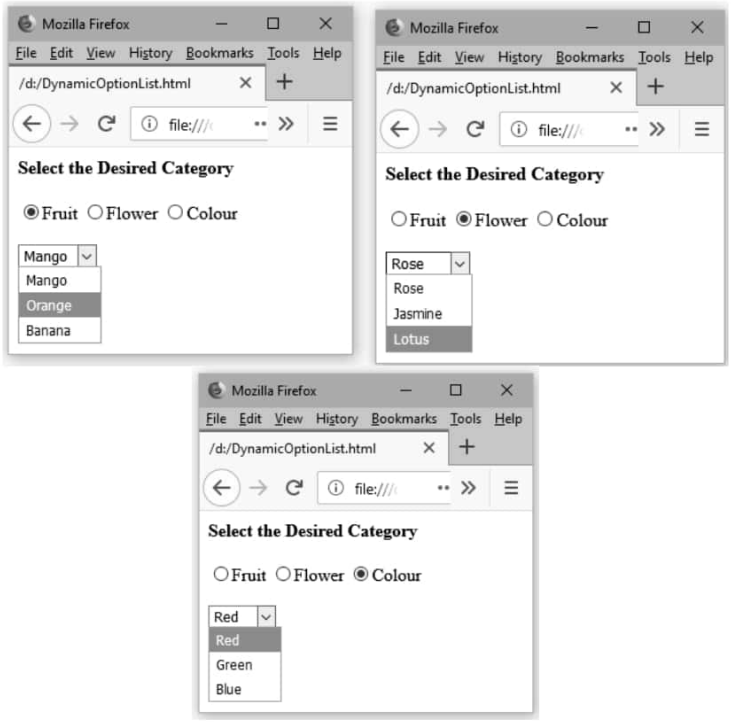
Ex. 3.5.1 : ***Write a JavaScript to create three categories - Fruit, Flower and Colour. Based on the selection of category, the items in the option list must get changed.***

Sol. : DynamicOptionList.html

```
<html>
<head>
    <script type="text/javascript">
        function MySelection(val)
        {
            with(document.forms.myform)
            {
                if(val==1)
                {
                    choices[0].text="Mango"
                    choices[0].value=1
                    choices[1].text="Orange"
                    choices[1].value=2
                    choices[2].text="Banana"
                    choices[2].value=3
                }
                if(val==2)
                {
```

```
                    choices[0].text="Rose"
                    choices[0].value=1
                    choices[1].text="Jasmine"
                    choices[1].value=2
                    choices[2].text="Lotus"
                    choices[2].value=3
                }
                if(val==3)
                {
                    choices[0].text="Red"
                    choices[0].value=1
                    choices[1].text="Green"
                    choices[1].value=2
                    choices[2].text="Blue"
                    choices[2].value=3
                }

            }
        }
    </script>
</head>
<body>
    <h4>Select the Desired Category</h4>
    <form name="myform">
        <input type="radio" name="items" checked="true" value=1
        onclick=MySelection(this.value)>Fruit
        <input type="radio" name="items" value=2
        onclick=MySelection(this.value)>Flower
        <input type="radio" name="items" value=3
        onclick=MySelection(this.value)>Colour
        <br/><br/>
        <select name="choices">
            <option Value=1>Mango
            <option Value=2>Orange
            <option Value=3>Banana
        </select>
    </form>
</body>
</html>
```

Output



3.6 Evaluating Check Box Selection

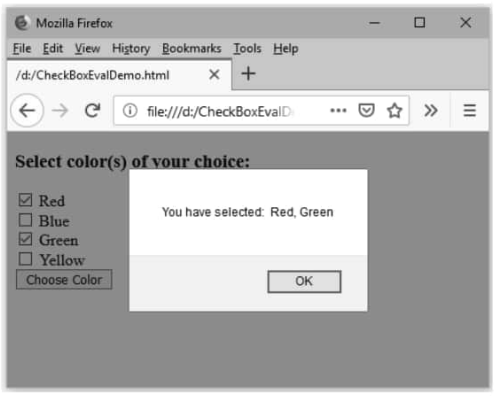
- Evaluating checkbox selection is a simple technique using which we can display the names of check boxes that are selected.
- Following JavaScript illustrates this concept.

JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        function MyFunction()
        {
            var str="You have selected: ";
```

```
                with(document.forms.myform)
                {
                    if(red.checked==true)
                        str+=" Red";
                    if(blue.checked==true)
                        str+=", Blue";

                    if(green.checked==true)
                        str+=", Green";
                    if(yellow.checked==true)
                        str+=", Yellow";
                }
                alert(str);
            }
    </script>
</head>
<body>
<h3> Select color(s) of your choice: </h3>
<form name="myform">
    <input type="checkbox" name="red" value="red"> Red<br/>
    <input type="checkbox" name="blue" value="blue"> Blue<br/>
    <input type="checkbox" name="green" value="green"> Green<br/>
    <input type="checkbox" name="yellow" value="yellow"> Yellow<br/>
    <input type="button" name="Color" value="Choose Color" onclick="MyFunction()">
</form>
</body>
```

Output



3.7 Changing a Label Dynamically

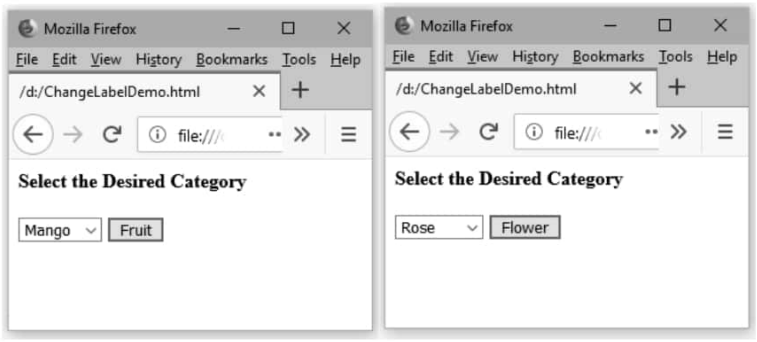
- We can change the label of any form element dynamically. The same element can be used for multiple purpose by simply changing the label.
- For example - Following is a JavaScript in which we are just changing the label of the "reset" button. Hence same button can be used for "Fruit" list display or "Flower" list display. The options of the List box also get changed accordingly.

JavaScript Document

```
<html>
<head>
    <script type="text/javascript">
        function MySelection(val)
        {
            with(document.forms.myform)
            {
                if(val=="Fruit")
                {
                    Button_Label.value="Flower"
                    choices[0].text="Rose"
                    choices[0].value=1
                    choices[1].text="Jasmine"
                    choices[1].value=2
                    choices[2].text="Lotus"
                    choices[2].value=3
                }
                if(val=="Flower")
                {
                    Button_Label.value="Fruit"
                    choices[0].text="Mango"
                    choices[0].value=1
                    choices[1].text="Orange"
                    choices[1].value=2
                    choices[2].text="Banana"
                    choices[2].value=3
                }
            }
        }
    </script>
</head>
<body>
    <h4>Select the Desired Category</h4>
    <form name="myform">
        <select name="choices">
            <option Value=1>Mango
            <option Value=2>Orange
            <option Value=3>Banana
        </select>
```

```
            <input type="reset" name="Button_Label" value="Fruit"
            onclick="MySelection(this.value)"/>
            <br/><br/>
        </form>
    </body>
    </html>
```

Output



3.8 Manipulating Form Elements

- We can manipulate the form elements before submitting it to web server. For that purpose we can keep some of the fields hidden and at the time of submitting the form, the desired value can be set to the hidden field so that the assigned value for hidden field can be submitted.
- Following example shows that - the user enters roll number and name. The registration id for the student can be formed by taking first two characters of 'name' followed by the roll number. Initially the registration id field is kept hidden and at the time of submitting the form this value is assigned to the registration field.

JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        function MyFunction()
        {
            with(document.forms.myform)
            {
                    if(name.value.length>0 && roll.value.length>0)
                    regid.value=name.value.charAt(0)+
                    name.value.charAt(1)+
                    roll.value
```

```
            }
        }
    </script>
</head>
<body>
    <form name="myform">
        Roll Number:<input type="text" name="roll"/>
        <br/><br/>
        Name      :<input type="text" name="name"/>
        <br/><br/>
        Reg.ID    :<input type="hidden" name="regid"/>
        <br/><br/>
        <input type="submit" name="Submit" value="Submit" onclick="MyFunction()"/>
    </form>
</body>
</html>
```

3.9 Intrinsic Functions, Disabling Elements and Read-only Elements

3.9.1 Intrinsic Functions

- Intrinsic functions means the built in functions that are provided by JavaScript.
- The javaScript provides the intrinsic functions for Submit or Reset button. One can use these functionalities while submitting the form or resetting the form fields.
- The **submit()** method of the form object can be used to send the form to the server in exactly same way as if the user has pressed the Submit button.

JavaScript Example

```
<!DOCTYPE html>
<html>
<body>
        <form name="myform">
        Roll Number:<input type="text" name="roll"/>
        <br/><br/>
        Name      :<input type="text" name="name"/>
        <br/><br/>
        <img src="submit.gif" onclick="javascript:document.forms.myform.submit()"/>
        <br/><br/>
        </form>
</body>
</html>
```

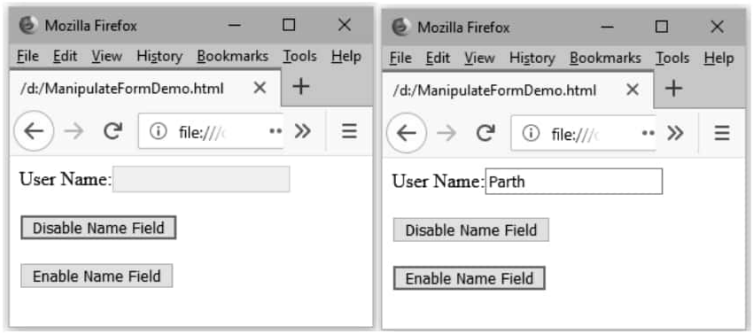
3.9.2 Disabling Elements

- We can restrict some fields on the form by using **disabled.**
- If disabled property of particular form element is set to true then user can not edit that element. Similarly on setting disabled property to **false** we can edit the field.

For example

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        function EnableFunction()
        {
            document.forms.myform.name.disabled=false
        }
            function DisableFunction()
        {
            document.forms.myform.name.disabled=true
        }
    </script>
</head>
<body>
    <form name="myform">
    User Name:<input type="text" name="name"/>
    <br/><br/>
    <input type="button" value="Disable Name Field" onclick="DisableFunction()"/>
    <br/><br/>
<input type="button" value="Enable Name Field" onclick="EnableFunction()"/>
    </form>
</body>
</html>
```

Output



3.9.3 Read-only Elements

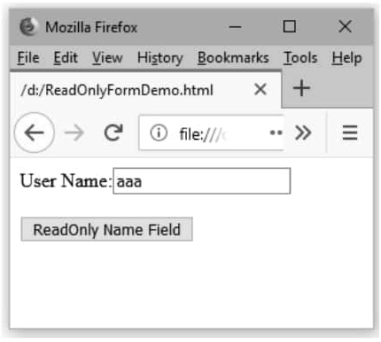
- Sometimes we need to set some value to a field which user should not change. To restrict user from changing the value of particular field we make that element **readonly** by setting **readonly=true.**
- If the readonly attribute is set to false, then anyone, including the user entering information into the form, can change the value of the element.

- Following example illustrates the use of readonly element

JavaScript Example

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        function ReadOnlyFunction()
        {
            document.forms.myform.name.readOnly=true
        }
    </script>
</head>
<body>
    <form name="myform">
    User Name:<input type="text" name="name"/>
    <br/><br/>
    <input type="button" value="ReadOnly Name Field" onclick="ReadOnlyFunction()"/>
    </form>
</body>
</html>
```

Output



Review Questions

1. *What is form?*
2. *Explain two uses of forms*
3. *Explain in brief the properties and methods used in form*
4. *How will you create password field in a HTML form ?*
5. *Explain the use of button element with the help of example*
6. *What is the difference between group of checkbox buttons and group of radio buttons ?*
7. *Design a simple HTML form for filling the information for registration of a student.*
8. *Write a form to make login and password.*

9. *What is event handler. Give an example of event handler in Javascript*
10. *Explain various commonly used mouse Events in JavaScript*
11. *Explain any two keyboard events in JavaScript*
12. *Explain how to change attribute values dynamically?*
13. *Write a JavaScript to create three categories - Fruit, Flower and Colour. Based on the selection of category, the items in the option list must get changed.*
14. *What is intrinsic function?*
15. *How to disable particular element on the form?*
16. *What is the use of readyonly element in JavaScript?*

❑❑❑

Notes

UNIT - IV

4 Cookies and Browser Data

4.1 Cookies

4.1.1 Basics of Cookies

- Cookies is a small piece of data stored in a file on your computer.
- The information present in the cookies is accessed by the web browser.
- Cookies remembers the information about the user in the following ways –

  Step 1 : When the user visits the web page his/her name can be stored in a cookie.

  Step 2 : Next time when the user visits the page, the cookie remembers his/her name.
- Cookies are saved in name-value pair. For example –

  Username = AAA BBB
- Cookie is a small text file which contains following data :
  - A **name-value pair** containing the actual data
  - An **expiry date** after which it is no longer valid
  - The **domain and path** of the server it should be sent to
- There are two types of cookies : **Session cookies** and **persistent cookies**
- **Session Cookies :** Session cookies is a cookie that remains in temporary memory only while user is reading and navigating the web site. The cookie is automatically deleted when the user exits the browser application.
- **Persistent Cookies :** A persistent cookie is a cookie that is assigned an expiration date. A persistent cookie is written to the computer's hard disk and remains there until the expiration date has been reached; then it is deleted.
- JavaScript can create, read or delete a cookie using **document.cookie** property.

4.1.2 Creating Cookies

Creations of cookies is a simple technique. For creating a cookie we need to assign cookie value to **window.document.cookie.** For instance

```
document.cookie="UserName=Anand; "
                 ↑        ↑     ↑
                Name     Value  Delimiter
```

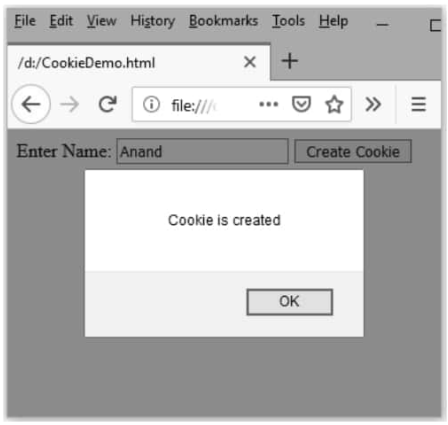
Thus the name value pair separated by = sign and terminated by a delimiter like semicolon(;), the cookie can be assigned to **document.cookie.** Instead of **window.document.cookie** we can simply write **document.cookie**

Ex. 4.1.1 ***Write a JavaScript for creating a cookie for the user name.***

Sol. :

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
     function CreateCookie()
     {
         with(document.form1)
         {
             document.cookie="UserName"+username.value+";" alert("Cookie is created");
         }
     }
 </script>
<head>
<body>
 <form name="form1">
     Enter Name: <input type="text" name="username"/>
     <input type="button" value="Create Cookie" onclick="CreateCookie()"/>
 </form>
</body>
</html>
```

Output



Script Explanation : In above JavaScript,

1) We have placed a one text element and one button element on the form. In the textfield user can enter the value for the cookie and then click the button 'Create Cookie'
2) On clicking the button, the function **CreateCookie()** is called

3) Inside the function **CreateCookie(),** for the **form1**, the cookie is set using the statement

```
document.cookie="UserName"+ username.value +";"
```

In above statement, the **username.value** is the value entered by the user. This value is assigned to the document.cookie.

That's it. The cookie is created!!. This message is flashed on the browser by using the **alert** box.

4.1.3 Reading a Cookie Value

- It is a common practice to create a cookie and then read the value of the cookie created.
- The **document.cookie** string will keep a list of **name=value** pairs separated by semicolons, where **name** is the name of a cookie and **value** is its string value.
- Using the **split()** function the string of cookies is break into key and values.
- The **split()** method finds the = character in the cookie, and then takes all the characters to the left of the = and store them into array **array [0]**.
- Next the **split()** method takes all the characters from the right of the = up to ; but not including the semicolon, and assign those characters to array **array [1]**.

Thus we can obtain the name value pair in **array[0]** and **array[1]** respectively.

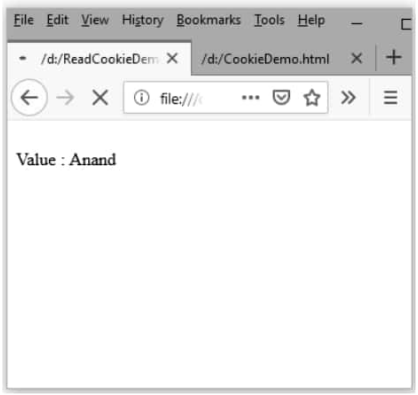
Following example illustrates how to read the cookies value.

Ex. 4.1.2 : ***Write a Javascript to read the cookies the user has already set.***

Sol. :

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
     function ReadCookie()
     {
         with(document.form1)
         {
             value = document.cookie.split('=')[1];
             document.write ("<br/>Value : " + value);
         }
     }
 </script>
</head>
<body>
 <form name="form1">
     <input type="button" value="Read Cookie" onclick="ReadCookie()"/>
 </form>
</body>
</html>
```

Output



4.1.4 Deleting Cookies

Cookies get deleted automatically when the browser session ends or its expiration date is reached.

Hence by setting previous expiry date we can delete the cookie.

Following JavaScript illustrate the process of deleting cookies.

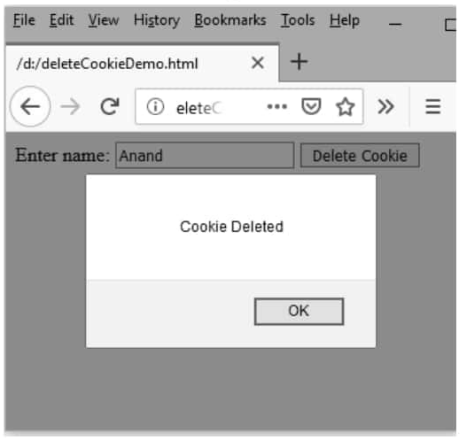
Ex. 4.1.3 : ***Write a JavaScript to delete the cookie.***

Sol. :

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
     function DeleteCookie()
     {
         expiryDate=new Date()
         expiryDate.setMonth(expiryDate.getMonth()-1)  // Earlier date of expiry is set
         with(document.form1)
         {
             document.cookie="UserName"+username.value+";"
             document.cookie="expires="+expiryDate.toUTCString ()+";"
             alert("Cookie Deleted");
         }
     }
 </script>
</head>
<body>
 <form name="form1">
     Enter name: <input type="text"
      name="username"/>
```

```
        <input type="button" value="Delete Cookie"
         onclick="DeleteCookie()"/>
    </form>
</body>
</html>
```

Output



Script Explanation : In above JavaScript,

1) To delete a cookie we are simply setting the expiry date of the cookie to a time in the past.
2) We take a **Date** variable **expiryDate.** The **getMonth()** returns the system provided month. We just set this month to past month by subtracting one from current month.
3) The new expiry date is assigned to the cookie.
4) Then the cookie is written by setting a new expiry date. This is how the cookie gets deleted.

4.1.5 Setting Expiration Date of Cookie

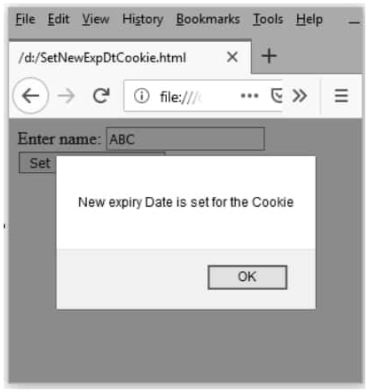
- We can set the expiration date of cookie by extending the life of a cookie beyond the current browser session. Then, we need to save this extended expiry date within the cookie. This can be done by setting the **'expires'** attribute to a date and time.
- For that matter we need three important built in functions –
  1) **getMonth() :** This method returns the current month based on the system clock of the computer running the JavaScript.
  2) **setMonth() :** This method assigns the month to Date variable
  3) **toGMTString() :** This method returns the value of the Date variable to a string that is in the format of Greenwich Mean Time, which is then assigned to the cookie.

Ex. 4.1.4 : ***Write a JavaScript to set the new expiry date of cookies.***

Sol. :

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
     function SetExpDtCookie()
     {
         expiryDate=new Date
         expiryDate.setMonth(expiryDate.getMonth()+1)
         with(document.form1)
         {
             document.cookie="UserName"+username.value+";"
             document.cookie="expires="+expiryDate.toUTCString ()+";"
             alert("New expiry Date is set for the Cookie");
         }

     }
 </script>
</head>
<body>
 <form name="form1">
     Enter name: <input type="text"
     name="username"/>
      <input type="button" value="Set New Expiry Date" onclick="SetExpDtCookie()"/>
 </form>
</body>
</html>
```

Output



4.2 Browser

- It is possible to open a new browser window from a currently running JavaScript. One can determine the size, location of this window, toolbar, scroll bar or any other style that normally the browser windows have.
- Once the new browser window is set up, it is also possible to change the contents within that window dynamically.

4.2.1 Opening a Window

- It is possible to open a new window from a JavaScript by simply clicking a button.
- For that purpose the **window** object is used. This **window** object has various useful properties and methods.
- To open a new window we use **open()** method of **window** object.

Syntax

The syntax of using open() method is

```
window.open(URL, name, syle)
```

where

URL : The URL specifies the URL of the web page that will appear in new window. This is an optional parameter.

name : The name that could be assigned to a window. This is an optional parameter

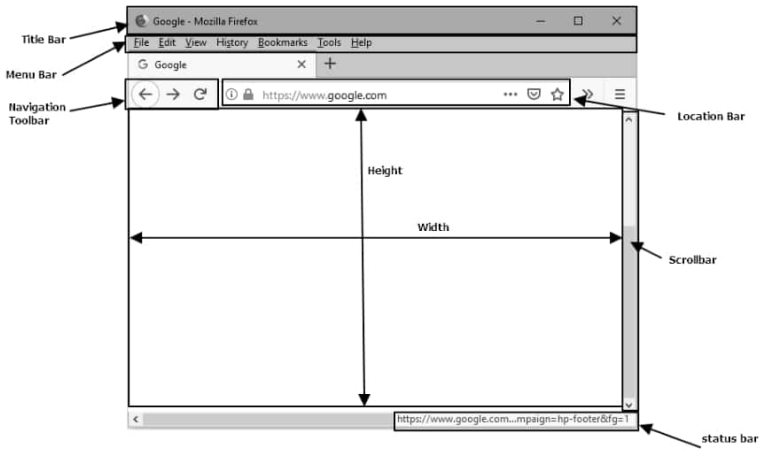
style : The style of the window includes various parameters such as toolbar, scrollbar, location,height and width of window and so on. This is an optional parameter. Various styles that can be possible are enlisted in the following table

Style	Purpose
fullscreen=yes|no|1|0	To display the browser in full-screen mode.
height=pixels	The height of the window. Min. value is 100
left=pixels	The left position of the window. Negative values not allowed
location=yes|no|1|0	To display the address field. Opera only
menubar=yes|no|1|0	To display the menu bar
resizable=yes|no|1|0	To determine if the window is resizable.
scrollbars=yes|no|1|0	To display scroll bars.
status=yes|no|1|0	To add a status bar
titlebar=yes|no|1|0	To display the title bar.
toolbar=yes|no|1|0	To display the browser toolbar.
top=pixels	To determine the top position of the window. Negative values not allowed
width=pixels	The width of the window. Min. value is 100

Return Value

A newly created window gets opened up on success and null on failure

Layout and Parts of Window

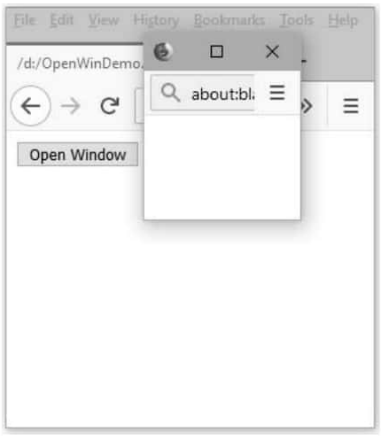


Ex. 4.2.1 : *Write a JavaScript to open a new window.*

Sol. :

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
     function OpenWindowFunction()
     {
         var mywin=window.open("","","width=100,height=100")

     }
 </script>
</head>
<body>
 <form name="form1">
     <input type="button" value="Open Window" onclick="OpenWindowFunction()"/>
 </form>
</body>
</html>
```

Output

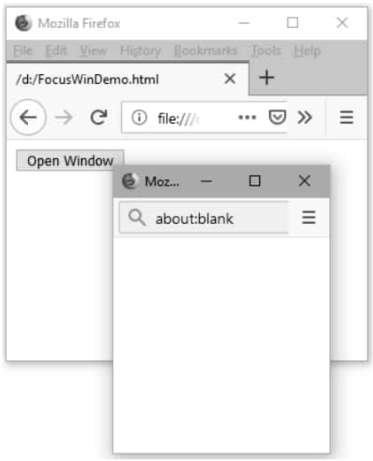


4.2.2 Giving the New Window Focus

A focus to a new window can be given using **focus()** method. Following JavaScript illustrates the use of focus() method

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
      function OpenWindowFunction()
      {

          var mywin=window.open("","MyWindow",
          "status=0,toolbar=0,location=0, menubar=0,directories=0,
          resizable=0,width=200,height=200")
          this.focus()
      }
 </script>
</head>
<body>
 <form name="form1">
      <input type="button" value="Open Window" onclick="OpenWindowFunction()"/>
 </form>
</body>
</html>
```

Output



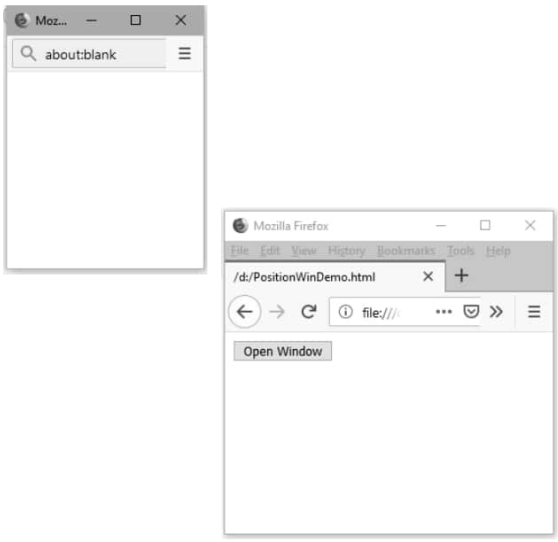
4.2.3 Window Position

We can set the desired position for the window. Using the **left** and **top** attribute values the window position can be set.

JavaScript Example

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
     function OpenWindowFunction()
     {
         Var mywin=window.open ("","MyWindow","left=0, top=0,width=200,height=200")
     }
 </script>
</head>
<body>
 <form name="form1">
     <input type="button" value="Open Window" onclick="OpenWindowFunction()"/>
 </form>
</body>
</html>
```

Output



Script Explanation : In above JavaScript, as we set the style attributes as left=0, right=0 in **open()** method, the window is created and positioned at the extreme left and top location of screen.

4.2.4 Changing the Contents of Window

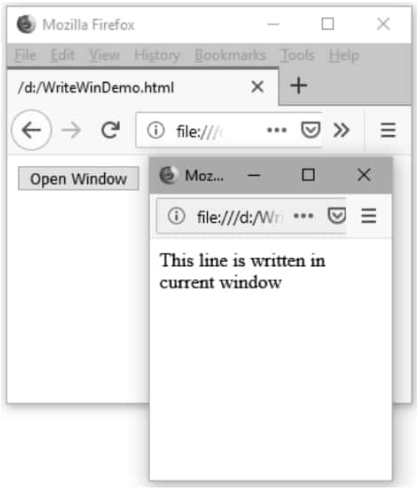
By writing some text to the newly created window we can change the contents of a window. For example – In the following JavaScript we have written some text using **write** method

JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
     function OpenWindowFunction()
     {
         var mywin=window.open("","MyWindow","width=200,height=200")
         mywin.document.write("<p> This line is written in current window</p>");
     }
 </script>
</head>
<body>
 <form name="form1">
     <input type="button" value="Open Window" onclick="OpenWindowFunction()"/>
```

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

Output



4.2.5 Closing a Window

The most simple operation about the window is to close it. It can be closed using the function **close().**

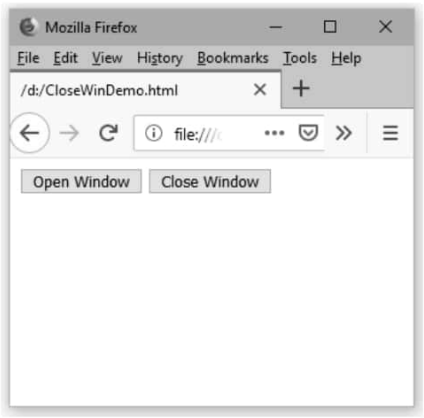
For example -

CloseWinDemo.html

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
     var mywin;
     function OpenWindowFunction()
     {

         mywin=window.open("","MyWindow","left=500, top=400,width=200,height=200")
     }
     function CloseWindowFunction()
     {
         mywin.close();
     }
 </script>
</head>
```

```
<body>
 <form name="form1">
     <input type="button" value="Open Window" onclick="OpenWindowFunction()"/>
     <input type="button" value="Close Window" onclick="CloseWindowFunction()"/>
 </form>
</body>
</html>
```

Output



Script Explanation : In above Java script,

(1) We have created two buttons – one for opening the window and another is for closing the window. On clicking the **Open Window** button the function **OpenWindowFunction()** is called and on clicking the **Close Window** button the function **CloseWindowFunction()** is called. Inside this function close() method is called to close the window.

4.2.6 Scrolling a Web Page

We can scroll horizontally or vertically using **ScrollTo()** function. Foe example –

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
     function ScrollFunction()
     {
         window.scrollTo(300,0)
     }
</script>
</head>
<body>
 <form name="form1">
```

```
 <input type="button" value="Scroll" onclick="ScrollFunction()"/>
 </form>
</body>
</html>
```

4.2.7 Multiple Windows at a Glance

It is possible to open up multiple windows at a time. It is simple to open multiple windows. Simply put **Open()** method in a loop

For example

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
     function OpenWindowFunction()
     {
         for(i=0;i<5;i++)
         {
             var mywin=window.open("","MyWindow"+i,"width=100,height=100")
         }
     }
 </script>
</head>
<body>
 <form name="form1">
     <input type="button" value="Open Window" onclick="OpenWindowFunction()"/>
 </form>
</body>
</html>
```

Output

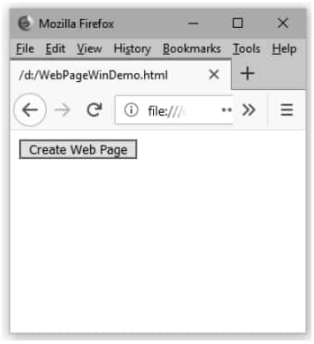
4.2.8 Creating a Web Page in New Window

- We can create a web page using the **window** object with the help of **write** method.
- The only thing is that inside the **write()** we have to write the contents of the web page with the help of html elements such as <head>, <body>, <title>,<h1> and so on.

JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
      function CreateFunction()
      {
          var mywin=window.open("","MyWindow","width=300,height=300")
          mywin.document.write("<html>");
          mywin.document.write("<head>");
          mywin.document.write("<title>WEB SITE DEMO</title>");
          mywin.document.write("</head>");
          mywin.document.write("<body>");
          mywin.document.write("<h2>This is a new Web Page</h2>");
          mywin.document.write("<h3>Welcome User!!!</h3>");
          mywin.document.write("</body>");
          mywin.document.write("</html>");
      }
 </script>
</head>
<body>
 <form name="form1">
      <input type="button" value="Create Web Page" onclick="CreateFunction()"/>
 </form>
</body>
</html>
```

Output



4.2.9 JavaScript in URLs

The Javascript code can be included in client side. JavaScript can be specified in URL using the pseudo-protocol specifier.

This special protocol type specifies that the body of the URL is arbitrary JavaScript code to be interpreted by the JavaScript interpreter.

For example we can type following code in URL bar

```
Javascript:alert(0)
```

This will show a pop up window of alert box.

If the JavaScript code in a **javascript :** URL contains multiple statements then these statements must be separated by semicolons.

For example –

```
javascript:var t = new Date(); " The time is: " + t;
```

But for the security reasons modern web browsers are not allowing to execute "javascript:" code in URL.

4.2.10 JavaScript Security

- Javascript has several security issues that need widespread attention.
- One of the most common JavaScript security vulnerabilities is **Cross-Site Scripting (XSS).** Cross-Site Scripting vulnerabilities enable attackers to manipulate websites to return malicious scripts to visitors. These malicious scripts then execute on the client side in a manner determined by the attacker. This vulnerability may cause the user data theft, account tampering and so on.
- JavaScript vulnerabilities can be both client-side problems and server-side as well as hackers are able to steal server-side data and infect the users with malware. Therefore, **server-side JavaScript injection** is also a much more dangerous problem in an application.

- **Cross-Site Request Forgery (CSRF) attack** is another issue in Javascript Cross-Site Request Forgery involves taking over or impersonating a user's browser session by hijacking the session cookie. CSRF attacks can trick users into executing malicious actions the attacker wants, or into taking unauthorized actions on the website.

4.2.11 Timers

Using **window** object it is possible to execute the code at specified time intervals.

Two functions are supported by **window** object for handling timing events and those are –

1. setTimeout **2. setInterval**

Let's discuss them

(1) setTimeout

This method executes function after waiting for a specified number of milliseconds.

Syntax

```
window.setTimeout(function, milliseconds) ;
```

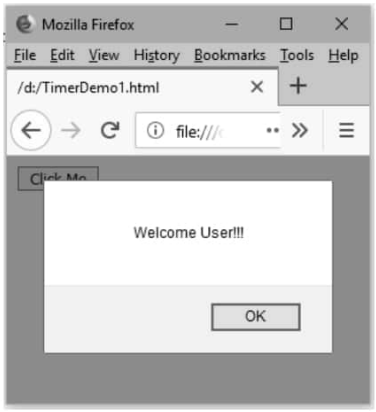
Function to be executed

Number of milliseconds before execution take place

Example

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
     function MyFunction()
     {
         alert("Welcome User!!!");
     }
 </script>
</head>
<body>
 <form name="form1">
     <input type="button" value="Click Me" onclick="setTimeout(MyFunction,4000);"/>
 </form>
</body>
</html>
```

Output



(2) SetInterval

This function executes the specific function at every give time interval

Syntax

```
window.setInterval(function, milliseconds) ;
```

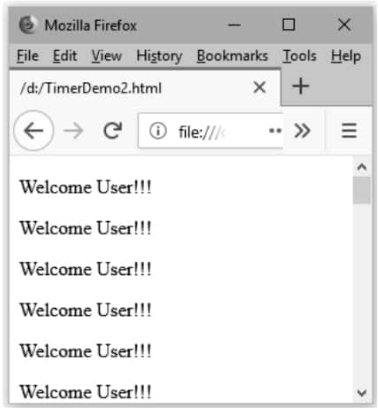
Function to be executed — Length of time interval between each execution

Example

```
<!DOCTYPE html>
<html>
<body>
        <p id="ID"></p>
        <script type="text/javascript">
        var t=setInterval(MyFunction,2000);
        function MyFunction()
        {
            document.getElementById("ID").innerHTML+="<p>Welcome User!!!</p>";
        }
 </script>
</body>
</html>
```

Output

Note that the welcome message will be displayed repeatedly on the browser window after some specific time interval.



4.2.12 Browser Location and History

(1) Location

The **window.location** object is useful for finding out the current location or path of the web page. Various properties used by **window.location** are described in the following table

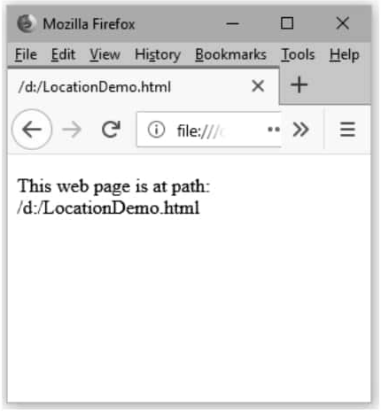
Sr. No.	Property	Purpose
1.	window.location.hostname	It returns the name of the host on which the web page is running.
2.	window.location.pathname	It returns the path name at which the web page is located. It includes folder and file name.
3.	window.location.protocol	It returns the web protocol used such as HTTP, File, HTTPS.
4.	window.location.assign	It loads the new document

Ex. 4.2.2 : ***Write a JavaScript to display the pathname of the web page using window.location object.***

Sol. :

```
<!DOCTYPE html>
<html>
<body>
    <p id="ID"></p>
    <script type="text/javascript">
    document.getElementById("ID").innerHTML = "This web page is at path: " + window.location.pathname;
 </script>
</body>
</html>
```

Output

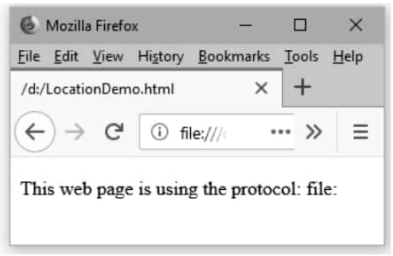


Ex. 4.2.3 : *Write a JavaScript to display the protocol of the web page using window.location object.*

Sol. :

```
<!DOCTYPE html>
<html>
<body>
 <p id="ID"></p>
 <script type="text/javascript">
document.getElementById("ID").innerHTML = "This web page is using the protocol: " + window.location.protocol;
 </script>
</body>
</html>
```

Output



(2) History

The **window.history** object is used for displaying browser history.

There are two important methods of **window.browser**

(1) window.history.back() : This method loads the previous URL in the history list. It is like clicking Back button on Browser window.

(2) window.history.forward() : This method loads the next URL in the history list. It is like clicking Forward button on Browser window.

JavaScript Example

```
<html>
<head>
    <script type="text/javascript">
    function MoveBack()
    {
        window.histoy.back();
    }
    function MoveForward()
    {
        window.histoy.forward();
    }
 </script>
</head>
<body>
 <form name="form1">
    <input type="button" value="Back" onclick="MoveBack()"/>
    <input type="button" value="Forward" onclick="MoveForward()"/>
 </form>
</body>
</html>
```

Review Questions

1. *What is cookies ?*
2. *Explain how to create cookies in JavaScript.*
3. *Explain how to read cookies in JavaScript.*
4. *Write a JavaScript to delete the cookie.*
5. *Explain the functions getMonth(), setMonth(), toGMTString()*
6. *Explain the syntax for opening a window.*
7. *Explain the use of focus() method for window object.*
8. *Write a JavaScript to change the contents of the newly created window.*
9. *Write a JavaScript to open multiple windows at a time.*
10. *What is the use of setTimeout() function? Write a JavaScript to illustrate it.*
11. *What is the use of setInterval() function? Write a JavaScript to illustrate it.*
12. *Write a JavaScript to display the pathname of the web page using window.location object.*

❑❑❑

Notes

UNIT - V

5 Regular Expression, Rollover and Frames

5.1 Regular Expression

Definition : Regular Expression is a special text string that defines the search pattern. It is a logical expression.

For example – for counting specific characters in a string or to replace some substring by another substring we need to create a regular expression.

We can create a regular expression pattern using forward slash /. For instance –

re = /abc/

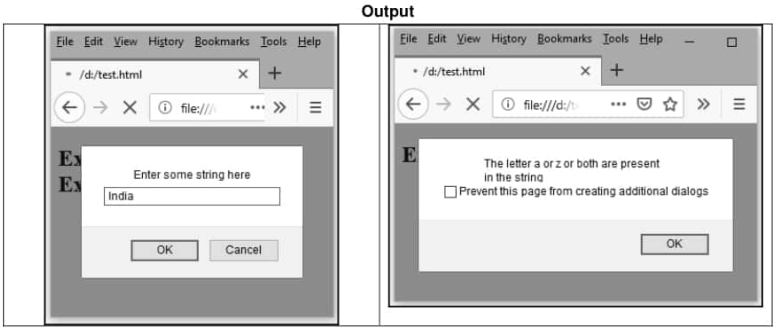
Regular expression is a powerful way for searching and replacing the characters in the string.

Ex. 5.1.1 : ***Write a Java program to test if the string contains the letter 'a' or 'z' or both.***

Sol. :

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
     function TestString(str)
     {
         re=/[az]/;
         if(re.test(str))
         {
             alert("The letter a or z or both are present in the string");
         }
         else
         {
             alert("String does not contain a or z or both");
         }
     }
 </script>
</head>
<body>
<h2>Example of Regular Expression</h2>
<script type="text/javascript">
 var input_str=prompt("Enter some string here","");
 TestString(input_str);
</script>
</body>
</html>
```

Output



Script Explanation : In above JavaScript,

(1) We have accepted input string using prompt pop up box. Whatever input string is entered by the user is stored in variable **input_str.**

(2) This string is then passed to the function **TestString** function. This section is written in the <head> section.

(3) In this function the regular expression is written as

re=/[az]/;

The regular expression begins and ends with slash /.

(4) a pair of square brackets [ ] appears following the first slash. This tells the browser to search the text for characters that appear within the brackets. In this expression, two characters are within the square brackets : 'a' and 'z' , which tells the browser to determine whether the text includes a or z, or both. That's what is the regular expression!

(5) This regular expression is assigned to the variable **re.**

(6) Using **test()** method we can search for desired characters from the input string **input_str.**

5.1.1 Language of Regular Expression

The words of regular expression are called special characters. Various special characters that can be used in Regular expression along with their meanings are written in the following table :

Special Character	Meaning
.	Any character except newline
A	The character a
ab	The string ab
a|b	a or b
a*	0 or more a's
\	Escapes a special character
[ab-d]	One character of: a, b, c, d
[^ab-d]	One character except: a, b, c, d
[\b]	Backspace character
\d	One digit
\D	One non-digit
\s	One whitespace
\S	One non-whitespace
\w	One word character
\W	One non-word character
*	0 or more
+	1 or more
?	0 or 1
{2}	Exactly 2
{2, 5}	Between 2 and 5
{2,}	2 or more

(...)	Group of pattern
^	Start of string
$	End of string
\b	Word boundary
\n	Newline
\r	Carriage return
\t	Tab
\0	Null character

Methods that use regular expressions

Method	Description
exec	A RegExp method that executes a search for a match in a string. It returns an array of information or null on a mismatch.
test	A RegExp method that tests for a match in a string. It returns true or false.
match	A String method that executes a search for a match in a string. It returns an array of information or null on a mismatch.
matchAll	A String method that returns an iterator containing all of the matches, including capturing groups.
search	A String method that tests for a match in a string. It returns the index of the match, or –1 if the search fails.
replace	A String method that executes a search for a match in a string, and replaces the matched substring with a replacement substring.
split	A String method that uses a regular expression or a fixed string to break a string into an array of substrings.

5.1.2 Finding Non Matching Characters

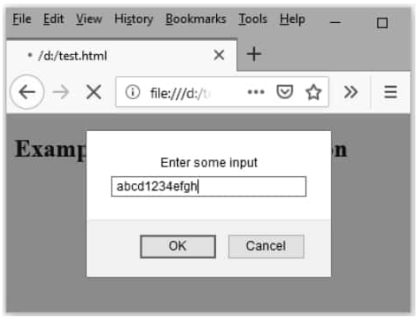
We can find the non matching characters from the given text by placing ^ as the first character within a square [ ].

Ex. 5.1.2 : ***Write a Java program that checks whether the string entered by the user contains digits or not.***

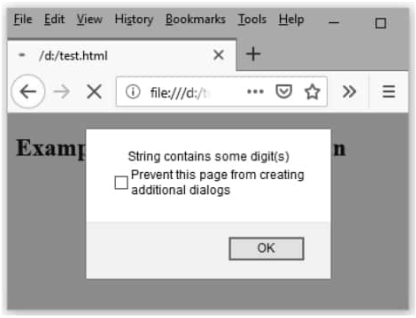
Sol. :

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        function TestString(str)
        {
            re=/[^[0-9]]/; // [0-9] indicates any digit
            if(re.test(str))
            {
                alert("The string does not contain
                        any digit");
            }
            else
            {
                alert("String contains some digit(s)");
            }
        }
    </script>
</head>
<body>
<h2>Example of Regular Expression</h2>
<script type="text/javascript">
    var input_str=prompt("Enter some input","");
    TestString(input_str);
</script>
</body>
</html>
```

Output



Click OK button and you will get -



5.1.3 Entering a Range of Characters

For matching any digit we need not have to enter every digit right from 0 to 9. Similarly for matching letters we need not have to test with every single alphabet. We can achieve this by entering range of characters.

For example – to match a digit we must have a regular expression as [0-9]. Thus placing the range within a square bracket helps us to evaluate a complete range of set of characters.

Suppose we enter [j-s] then that means match the characters j, k, l, m, n, o, p, q, r and s.

5.1.4 Matching Digits and Non Digits

Determining whether the string contains digits or non digits is a common task in any search pattern. For instance – in the application of validating telephone number this is the most wanted task.

This can be simplified by JavaScript by writing the regular expression.

If we write \d then that means search the text for digits and if we write \D then that means search the text for non-digits.

5.1.5 Matching Punctuations and Symbols

- The \w special symbol tells the browser to determine whether the text contains a letter, number, or an underscore.
- The \W special symbol tells the browser to determine whether the text contains other than a letter, number, or an underscore.
- Using \W is equivalent to using [a-zA-Z0-9_]. The last _ indicates space character.

5.1.6 Matching Words

- Any word in the text is defined as set of characters. A word is determined by a word boundary that is the space between two words.
- The boundary can be defined by using special symbol \b
- For example – From the string 'Boycott' we can get a match for 'cot' by using regular expression /\bcot\b/

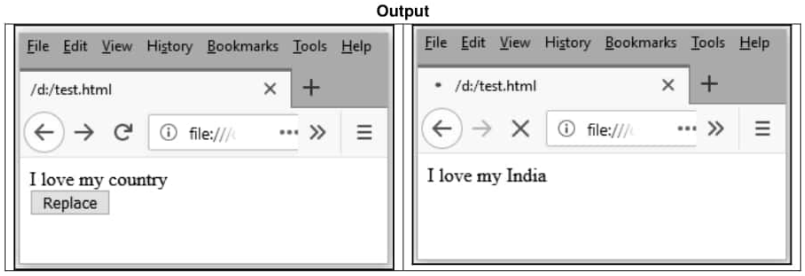
5.1.7 Replacing a Text using Regular Expressions

Using **replace** function we can replace the desired pattern. The first parameter in the replace function is the string which is to be replaced and the second parameter is the replacing string.

For example- Consider following JavaScript in which the word 'country' is replaced by 'India'

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
      function TestString(str)
      {
          new_str=str.replace("country","India");
          document.write(new_str);
      }
    </script>
</head>
<body>
    <script type="text/javascript">
    str="I love my country";
    document.write(str);
    </script>

    <form name="form1">
        <input type="button" value="Replace"
          onclick="TestString(str)">
    </form>
</body>
</html>
```

Output



5.1.8 Returning a Matched Character

The **exec()** method searches string for text that matches with the regular expression. If it finds a match , it returns an array of results, otherwise it returns null.

If we want to search for particular pattern from a text then exec() method can be used as follows –

pattern.**exec(text)**

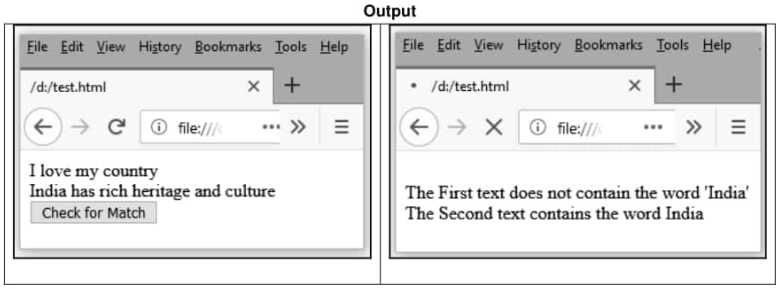
This function returns an array of matched result.

Example Program

```
<!DOCTYPE html>
<html>
<head>
 <script type="text/javascript">
     function TestString(str1,str2)
     {
         re=/India/g;
         result1=re.exec(str1);
         result2=re.exec(str2);
         if(result1)
         document.write("<br/>The First text contains the word "+result1);
         else
         document.write("<br/>The First text does not contain the word 'India' ");

         if(result2)
         document.write("<br/>The Second text contains the word "+result2);
         else
         document.write("<br/>The Second text does not contain the word 'India' ");
     }
 </script>
</head>
<body>
 <script type="text/javascript">
 str1="I love my country";
 str2="India has rich heritage and culture";
 document.write(str1);
```

```
document.write("<br/>"+str2);
</script>
<form name="form1">
    <input type="button" value="Check for Match" onclick="TestString(str1,str2)">
</form>
</body>
</html>
```

Output



5.1.9 Regular Expression Object Properties

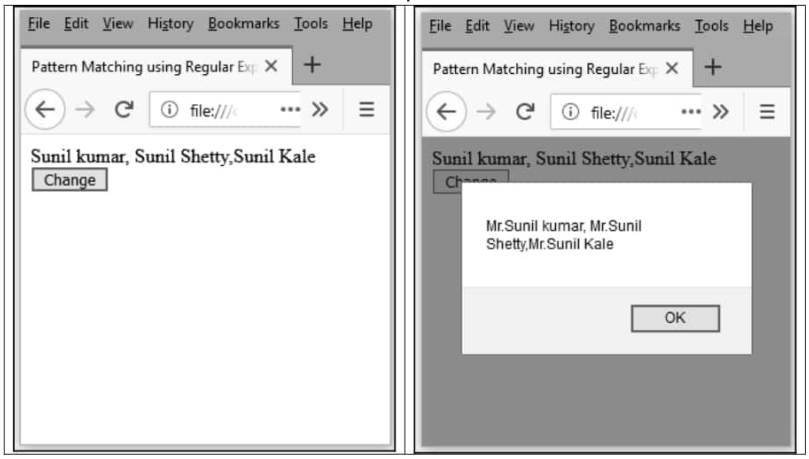
There are various regular expression object properties that help in matching particular word, character, last character, index at which to start the next match and so on. These properties are enlisted in the following table –

Regular Expression Object	Properties
$1 (through $9)	Parenthesized substring matches
$_	Same an input
$*	Same as multiline
$&	Same as lastMatch
$+	Same as lastParen
$`	Same as leftContent
$'	Same as rightContext
global	Search globally (g modifier in use)
ignoreCase	Search case-insensitive (i modifier in use)
input	The string to search if no string is passed
lastIndex	The index at which to start the next match
lastMatch	The last match characters
lastParen	The last parenthesized substring match
leftContext	The substring to the left of the most recent match
multiline	Whether strings are searched across multiple lines
prototype	Allows the addition of properties to all objects
rightContext	The substring to the right of the most recent match
source	The regular expression pattern itself

Example Program

```
<!DOCTYPE html>
<html>
<head>
 <title>Pattern Matching using Regular Expression </title>
 <script type="text/javascript">
      function TestString(str)
      {
      // for each Sunil - replace it with Mr. and then Sunil
      alert(str.replace(/Sunil/g, 'Mr.$&'));
      }
 </script>
</head>
<body>
 <script type="text/javascript">
      str= "Sunil kumar, Sunil Shetty,Sunil Kale";
      document.write(str);
 </script>
 <button onclick="TestString(str)">Change</button>
</body>
</html>
```

Output

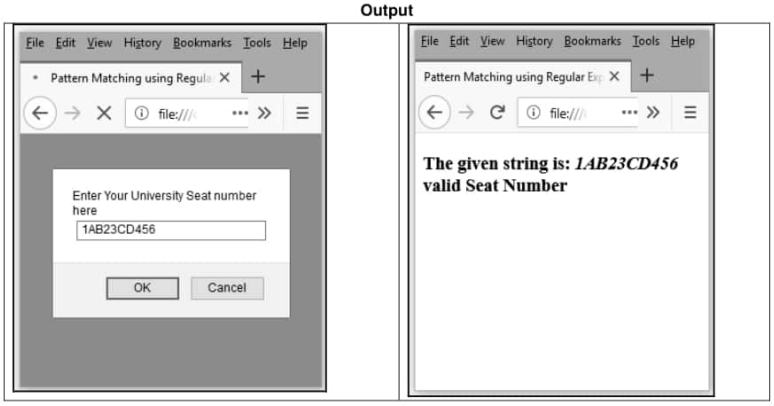


Ex. 5.1.3 : *Develop and demonstrate, using Javascript document that collects USN(the valid format is: A digit from 1 to 4 followed by two upper-case characters followed by two digits followed by two upper case characters followed by three digits:no embedded spaces allowed) of the user.Event handler must be included for the form element that collects this information to validate the input. Message in the alert windows must be produced when errors are detected.*

Sol. :

```
<!DOCTYPE html>
<html>
<head>
<title>Pattern Matching using Regular Expression </title>
<script type="text/javascript">
function TestString(str)
{
  document.write("The given string is: ");
  document.write("<em>"+str+"</em>"+"<br/>");
  var i=str.match(/[1234][A-Z]{2}\d{2}[A-Z]{2}\d{3}/);
  if(i==null)
   return false;
  else
  return true;
 }
</script>
</head>
<body>
  <h3>
    <script type="text/javascript">
      var input_str=prompt("Enter Your University Seat number here","");
      if(TestString(input_str))
        document.write("valid Seat Number");
 else
        document.write("Invalid Seat Number");
    </script>
    </h3>
</body>
</html>
```

Output



5.2 Frames

- HTML frames allow us to present documents in **multiple views.**
- Using multiple views we can keep certain information visible and at the same time other views are scrolled or replaced.
- **For example,** within the same window, one frame can display a company information, a second frame can be a navigation menu and a third frame may display selected document that can be scrolled through or replaced by navigating in the second frame.
- Various frames can be set in one browser window .

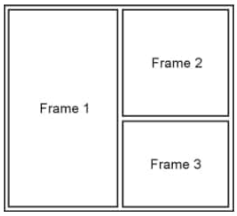


Fig. 5.2.1 Frames

5.2.1 Create a Frame

- To set the frames in the browser window we use frame set. For example :

```
<frameset cols="150,*">
```

will allow us to divide the window into two columns (i.e. in two vertical frames). One frame occupying the size of 150 pixels and the other occupies the remaining portion of the window. Here * means any number of pixels.

Similarly

```
<frameset rows="*,120">
```

will divide the window into two rows (i.e. in two horizontal frames). The second part of horizontal frame will be of 120 pixels and upper horizontal frame will occupy remaining portion of the window.

Similarly we can also specify the frameset in percentage form. For example

```
<frameset rows="30%,70%">
```

Using frameset we can divide the rows and columns in any number of frames. For example

```
<frameset rows = "20%,30%,50%" cols = "30%,*">
```

This will create a frames in the browser's window as follows -

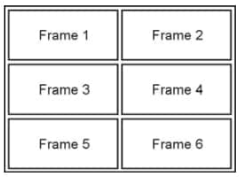


Fig. 5.2.2 Frames

- In every layout frame we can load the desired html page by using frame src. For example :

```
<frame src="D:\\html_examples\\bulleted1.html"
name="Left_Vertical">
```

By this statement, we are loading the web page **bulleted1.html** in the specific frame and the frame is named as **Left_Vertical.**

Attributes in frameset tag

Attribute	Value	Purpose
cols	pixels % *	It specifies the number and size of columns in a frameset
rows	pixels % *	It specifies the number and size of rows in a frameset.

Attributes of frame tag

The <frame> tag has no end tag. The <frame> tag defines one frame within a <frameset> tag. Various attributes of frame tag are

Attribute	Value	Purpose
frameborder	0 or 1	Value 1 specifies that the frame is displayed with border and 0 indicates that there is no border.
name	Some name	It specifies name of the frame

Nosize		Due to this attribute, we cannot resize the particular frame.
scolling	yes, no or auto	It specifies whether or not to display the scrollbar along with the frame.
src	URL	It specifies the name of the Document to be displayed within the frame.

Example of Browser containing frame

Step 1 : Create a main HTML document which will display three HTML documents in three vertical frames

FrameSet.html

```
<!DOCTYPE html>
<html>
<frameset cols="25%,*,50%">
  <frame src="frame1.html">
  <frame src="frame2.html">
  <frame src="frame3.html">
</frameset>
</html>
```

Step 2 : Create frame1.html, frame2.html and frame3.html files as follows –

Frame1.html

```
<!DOCTYPE html>
<html>
<body>
    <h1> Frame 1 </h1>
</body>
</html>
```

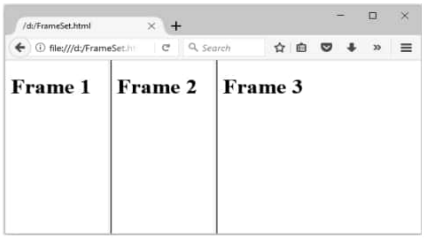
Frame2.html

```
<!DOCTYPE html>
<html>
<body>
    <h1> Frame 2 </h1>
</body>
</html>
```

Frame3.html

```
<!DOCTYPE html>
<html>
<body>
    <h1> Frame 3 </h1>
</body>
</html>
```

Output



Script Explanation :

The above HTML document, the **<frameset>** tag is used to define frameset. The **col** attribute is used to create the three column frames.

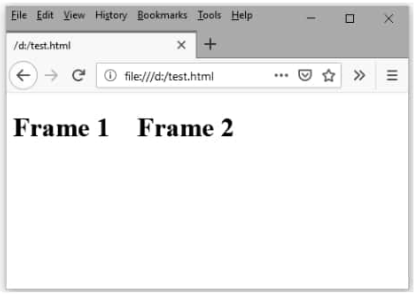
5.2.2 Invisible Borders of Frame

The borders of the frames can be made invisible by setting the attributes "frameborder" and "border" = 0.

For example –

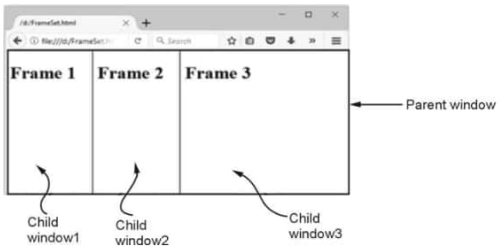
```
<!DOCTYPE html>
<html>
<frameset cols="30%,70%">
  <framesrc="frame1.html"frameborder="0" border="0"/>
  <framesrc="frame2.html"frameborder="0" border="0"/>
</frameset>
</html>
```

Output



5.2.3 Calling a Child Window

The main window defining the frames in it is called parent window and each frame contains the child windows. For example –



We can call one child window from another child window. Following is a JavaScript in which we are calling the function defined in another child window from one window.

Example Program

Step 1 : Create parent window script in which two frames are defined –

Test.html

```
<!DOCTYPE html>
<html>
<frameset cols="30%,70%">
  <frame src="frame1.html" name="LeftPage"/>
  <frame src="frame2.html" name="RightPage"/>
</frameset>
</html>
```

Step 2 : Write the **frame1.html** file as a **LeftPage frame.** The code for this is as follows –

frame1.html

```
<!DOCTYPE html>
<html>
<body>
 <h1> Frame 1 </h1>
 <form>
      <input type="button" name="Frame1" value ="Click Me" onclick="parent.RightPage.MyFunction()"/>
 </form>
</body>
</html>
```

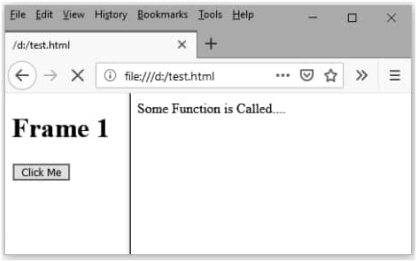
Step 3 : The definition of **MyFunction** is present in the second frame. Hence the code for this is -

Frame2.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Frame 2</title>
    <script language="Javascript"
      type="text/javascript">
      function MyFunction()
      {
          document.write("Some Function is Called....");
      }
    </script>
</head>
<body>
    <h1> Frame 2 </h1>
</body>
</html>
```

Step 3 : Now we will execute the parent JavaScript i.e. **test.html** on web browser

Just click the **Click Me** button and you can see the changes in another child window –



5.2.4 Changing the Content and Focus of Child Window

You can change the content of a child window from a JavaScript function by modifying the source web page for the child window.

We can assign the new source to the child window's **href** attribute.

Example Program : In the following example we are displaying two frames – Frame1 on the left hand side and Frame to at the right hand side. When the user clicks the button present in the frame1, the frame2 is removed and is replaced by **Frame3.**

Step 1 : We will write the JavaScript for displaying two frames on the parent window –

MainWindow.html

```
<!DOCTYPE html>
<html>
<frameset cols="30%,70%">
  <frame src="frame1.html" name="LeftPage"/>
  <frame src="frame2.html" name="RightPage"/>
</frameset>
</html>
```

Step 2 : The frame1 contains one button componenet. The code for this is as follows –

Frame1.html

```
<!DOCTYPE html>
<html>
<head>
<script language="Javascript" type="text/javascript">
        function MyFunction()
        {
            parent.RightPage.location.href='frame3.html'
        }
    </script>
</head>
<body>
    <h1> Frame 1 </h1>
    <form>
        <input type="button" name="Frame1"
          value ="Click Me" onclick="MyFunction()"/>
    </form>
</body>
</html>
```

Step 2 : The code for frame2 is as follows –

Frame2.html

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

```
    <title>Frame 2</title>
</head>
<body>
    <h1> Frame 2 </h1>
</body>
</html>
```

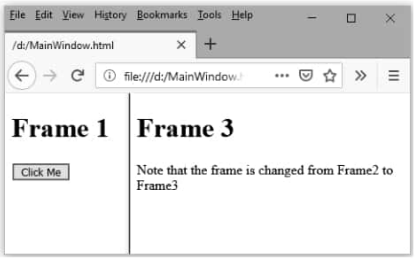
Step 3 : The code for Frame3 is as follows –

Frame3.html

```
<!DOCTYPE html>
<html>
<body>
    <h1> Frame 3 </h1>
    <p> Note that the frame is changed from Frame2
        to Frame3</p>
</body>
</html>
```

Step 4 : Get the output on the Web browser by loading MainWindow.html. The output is as follows –

Just click the Click Me button and you will get the following output



5.2.5 Accessing Elements of Another Child Window

It is possible to change the elements of one frame from another frame. For example – we can change the label of 'button' component of right frame from one the left frame. For that purpose we will add the code

```
parent.RightPage.form2.Frame2.value=New_Label
```

Following Java program illustrates this

Ex. 5.2.1 : ***Write JavaScript to change the label of button element present in frame2 from frame1.***

Sol. :

Step 1 : We will write the main JavaScript file which will load both the frames – Frame1 and Frame2

MainWindow.html

```
<!DOCTYPE html>
<html>
<frameset cols="30%,70%">
  <frame src="frame1.html" name="LeftPage"/>
  <frame src="frame2.html" name="RightPage"/>
</frameset>
</html>
```

Step 2 : The frame1 contains a single button. We expect that on clicking this button the label of button element of second frame should be changed.

Frame1.html

```
<!DOCTYPE html>
<html>
<head>
<script language="Javascript" type="text/javascript">
    function MyFunction()
    {
        parent.RightPage.form2.Frame2.value="Calculate"
    }
    </script>
</head>
<body>
    <h1> Frame 1 </h1>
    <form name="form1">
        <input type="button" name="Frame1"
          value ="Click Me" onclick="MyFunction()"/>
    </form>
</body>
</html>
```

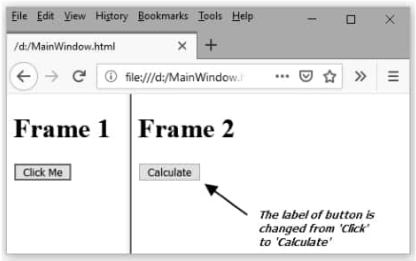
Step 3 : The second frame contains a single button element.

Frame2.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Frame 2</title>
</head>
<body>
    <h1> Frame 2 </h1>
    <form name="form2">
        <input type="button" name="Frame2"
          value ="Click"/>
    </form>
</body>
</html>
```

Step 4 : Open the web browser such as Mozilla FireFox or Chrome and load the MainWindow.html

Just click the **Click Me** button and see the change in second frame



5.3 Rollover

Rollover means change in the appearance of the object when user moves his or her mouse over an object on the page.

The rollover effect is mainly used in web page designing for **advertising purpose.**

5.3.1 Creating Rollover

On many web pages, javascript rollovers are handled by adding an onmouseover and onmouseout event on images.

(1) **onmouseover** is triggered when the mouse moves over an element

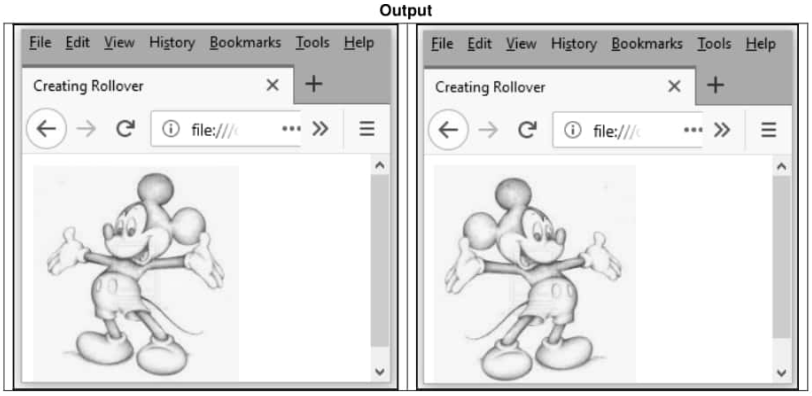
(2) **onmouseout** is triggered when the mouse moves away from the element

This is an example of how it works.

RolloverDemo.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Creating Rollover</title>
</head>
<body>
<a>
<img src="mickey1.jpg"
onmouseover="src=' mickey2.jpg'"
onmouseout="src=' mickey1.jpg'">
</a>
</body>
</html>
```

Output



5.3.2 Text Rollover

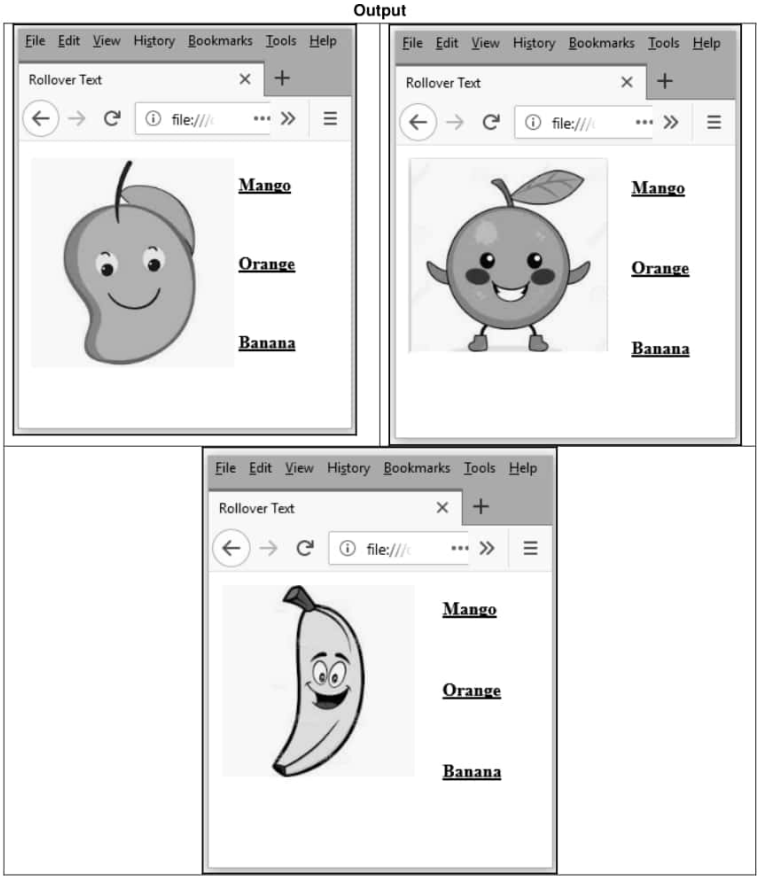
- Text rollover is a technique in which whenever user rollover the text , JavaScript allows to change the page element usually some graphics image.
- Carefully placed rollovers can enhance a visitor's experience when browsing the web page.
- Following example illustrates this idea

Ex. 5.3.1 : ***Write a JavaScript in which when user rolls over the name of fruit, the corresponding image should be displayed. For instance – if user rolls over the text "Mango"; the image of Mango should be displayed.***

JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
    <title>Rollover Text</title>
</head>
<body>
    <table>
        <tr>
            <td>
                <a>
                    <IMG src="MangoImg.jpg"
                     name="fruit">
                </a>
            </td>
            <td>
                <a onmouseover="document.fruit.src='
                  MangoImg.jpg'">
                    <b><u>Mango</u></b>
                </a>
                <br/><br/><br/><br/>
                <a onmouseover="document.fruit.src
                  ='OrangeImg.jpg'">
                    <b><u>Orange</u></b>
                </a>
                <br/><br/><br/><br/>
                <a onmouseover="document.fruit.src
                  ='banana.jpg'">
                    <b><u>Banana</u></b>
                </a>
            </td>
        </tr>
    </table>
</body>
</html>
```

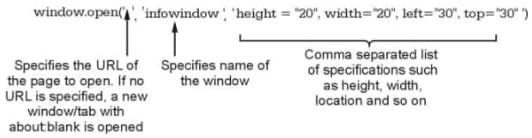
Output



Clearly, the above output illustrates that when user rolls over the text **Mango ,** the image of mango will be displayed, if user rolls over the text **Orange,** the image of orange will be displayed and if user rolls over the text **Banana,** the image of banana will be displayed.

5.3.3 Multiple Actions for Rollover

- Suppose user is rolling the cursor over the text, then instead of simply changing the image we can display more window displaying some features or additional information about the item on which the mouse is rolling over. This process is referred as multiple actions for rollover.
- Due to this effect visitor gets more information at a glance.
- We can open additional window using the built in function **Open().** This function is invoked using the object **Window.**
- The open() method opens a new browser window, or a new tab. The close() window closes the window.
- The window. open() function can be written as follows



This function returns the object or instance of a window. We store this instance in a variable named MyWindow.

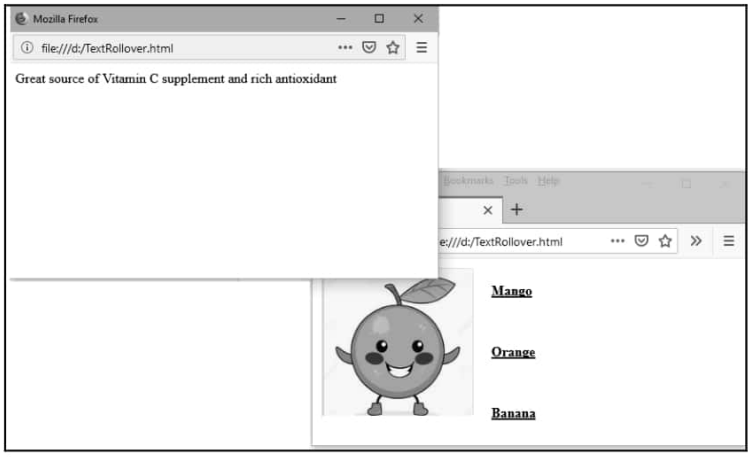
- Then using **MyWindow.document.write()** function we can write the information to this opened window. Thus it is possible to write additional information by opening and writing the contents to additional window.
- Following example illustrates this idea.

Ex. 5.3.2 : ***Write a JavaScript in which when user rolls over the name of fruit, the corresponding image should be displayed. Along with the appropriate image display, additional window should pop up displaying the benefit of each fruit.***

Sol. :

```
<!DOCTYPE html>
<html>
<head>
 <title>Rollover Text</title>
 <script language="Javascript">
     function MyFunction(choice)
     {
         if(choice==1)
         {
             document.fruit.src='MangoImg.jpg'
             MyWindow=window.open('','infoWindow',
             'height="20",width="20",left="30",top="30"')
             MyWindow.document.write(
             'Great source of Vitamin A and Vitamin C supplement')
         }
         if(choice==2)
         {
             document.fruit.src='OrangeImg.jpg'
```

```
                MyWindow=window.open('','infoWindow',
                'height="20",width="20",left="30",top="40"');
                MyWindow.document.write(
                'Great source of Vitamin C supplement and rich antioxidant')
            }
            if(choice==3)
            {
                document.fruit.src='banana.jpg'
                MyWindow=window.open('','infoWindow',
                'height="20",width="20",left="30",top="50"');
                MyWindow.document.write(
                'Full of Potassium and Fiber')
            }
        }
    </script>

</head>
<body>
<table>
<tr>
<td>
    <a>
        <IMG src="MangoImg.jpg"  name="fruit">
    </a>
</td>
<td>
    <a onmouseover="MyFunction(1)"
      onmouseout="MyWindow.close();">
    <b><u>Mango</u></b>
    </a>
    <br/><br/><br/><br/>
    <a onmouseover="MyFunction(2)"
      onmouseout="MyWindow.close();">
    <b><u>Orange</u></b>
    </a>
    <br/><br/><br/><br/>
    <a onmouseover="MyFunction(3)"
      onmouseout="MyWindow.close();">
    <b><u>Banana</u></b>
    </a>
</td>
</tr>
</table>
</body>
</html>
```



5.3.4 More Efficient Rollover

For efficient use of rollover, the images can be stored in an array and required images are displayed when the web page is loaded.

This makes the rollover action efficient because – the images are already collected and loaded in the array. The required image is displayed when user rollover particular text.

Ex. 5.3.3 : ***Write a JavaScript in which when user rolls over the name of fruit, the corresponding image should be displayed. For instance – if user rolls over the text "Mango"; the image of Mango should be displayed. Make use of array for storing the image files.***

Sol. :

JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
        <title>Rollover Text</title>
        <script language="Javascript">
        Rollimage = new Array() //creating array of three images
        Rollimage[0]=new Image(100,100)
        Rollimage[0].src='MangoImg.jpg' //assigning each image at each index of array

        Rollimage[1]=new Image(100,100)
        Rollimage[1].src='OrangeImg.jpg'

        Rollimage[2]=new Image(100,100)
        Rollimage[2].src='banana.jpg'

 </script>
</head>
```

```
<body>
        <table>
            <tr>
                <td>
                    <a>
                            <IMG src='MangoImg.jpg'  name="fruit">
                    </a>
                </td>
            <td>
                    <a onmouseover="document.fruit.src=Rollimage[0].src">
                        <b><u>Mango</u></b>
                    </a>
                    <br/><br/><br/><br/>
                    <a onmouseover="document.fruit.src=Rollimage[1].src">
                        <b><u>Orange</u></b>
                    </a>
                    <br/><br/><br/><br/>
                    <a onmouseover="document.fruit.src=Rollimage[2].src">
                        <b><u>Banana</u></b>
                    </a>
                </td>
            </tr>
    </table>
</body>
</html>
```

Output

It is same as in example 5.3.1.

Review Questions

1. *Define the term - Regular expression.*
2. *Write a Java program that checks whether the string entered by the user contains digits or not.*
3. *Explain any four commonly used methods in regular expression.*
4. *How will you specify the range of characters using a regular expression.*
5. *Give the regular expression for matching digits and non digits.*
6. *Explain any four regular expression object properties.*
7. *What is frame ?*
8. *Explain frameset tag along with the attributes used in it.*
9. *Write a JavaScript to display frames without border.*
10. *Write a JavaScript to change the contents of one frame from another frame.*
11. *Write JavaScript to change the label of button element present in frame2 from frame1.*
12. *What is rollover ?*
13. *What are the methods used most commonly during the rollover ?*
14. *Write a JavaScript in which when user rolls over the name of fruit, the corresponding image should be displayed. For instance - if user rolls over the text "Mango"; the image of Mango should be displayed.*
15. *Write a JavaScript in which when user rolls over the name of fruit, the corresponding image should be displayed. Along with the appropriate image display, additional window should pop up displaying the benefit of each fruit.*

❑❑❑

UNIT - VI

6 Menus, Navigation and Web Page Protection

6.1 Status Bar

Status bar is present at the bottom of browser window. It enhances the readability of the text present on the web page, when user scrolls over it.

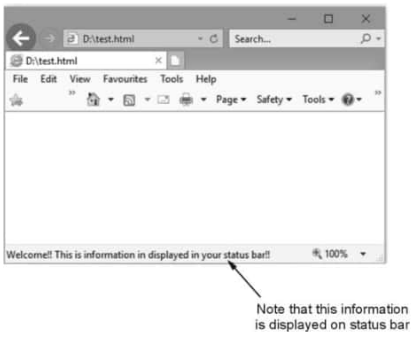
6.1.1 Build Static Message

We can build a static message which is displayed on the status bar. This message remains permanently present in the status bar.

Java Script

```
<!DOCTYPE html>
<html>
<body>
<script>
 window.status = "Welcome!! This is information
 in displayed in your status bar!!";
</script>
</body>
</html>
```

Output



6.1.2 Changing the Message using Rollover

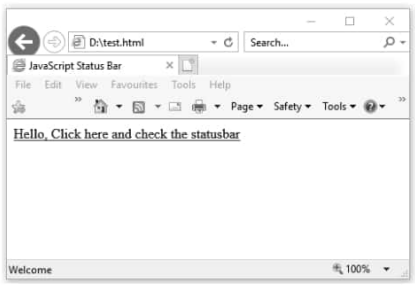
We will use **onMouseOver** and **onMouseOut** events of a hyper link to display or manage the messages.

We can use **window.status** on **OnMouseOver** event to change the status in the status bar. You can display a javascript status bar message whenever your users hover over your hyperlinks.

Java Script

```
<html>
<head>
<title>JavaScript Status Bar</title></head>
<body>
  <a href="../"
  onMouseOver="window.status='Welcome';return true"
  onMouseOut="window.status='';return true">
Hello, Click here and check  the statusbar
  </a>
</body>
</html>
```

Output



6.1.3 Moving the Message along the Status Bar

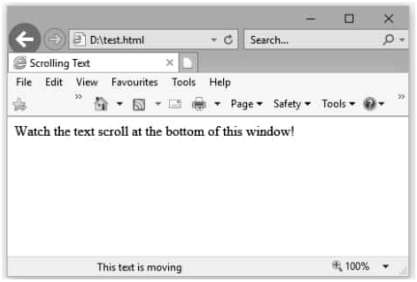
For moving the message in the status bar we need to increment the current position of the text one character ahead in a loop. This will give us the effect of moving the text. This will be displayed by using the **setInterval** function, which displays the moved text after some milliseconds.

Following javascript illustrates this idea

Java Script

```
<html>
<head>
<title>Scrolling Text</title>
<script language="JavaScript">

var currentPosition = 0
var targetPos = 100
var blanks = ""

function scroll_function(msg, milliseconds) {
window.setInterval("display('"+msg+"')", milliseconds)
}

function display(msg)
{
    window.defaultStatus = blanks + msg
    ++currentPosition
    blanks += " "
    if(currentPosition > targetPos)
    {
        currentPosition = 0
        blanks = ""
    }
}
</script>
</head>
<body onload="scroll_function
  ('This text is moving', 100)">
<p>Watch the text scroll at the
  bottom of this window!</p>
</body>
</html>
```

Output



6.2 Banner

6.2.1 Loading and Displaying Banner Advertisement

- It is a typically rectangular advertisement placed on a Web site either above, below or on the sides of the Web site's main content and is linked to the advertiser's own Web site.
- It is also referred as **banner ad.**
- The banner may contain text or graphics images.
- Following is a simple JavaScript that displays the banners in which it slides from one banner to another.

Prerequisite : In our example,

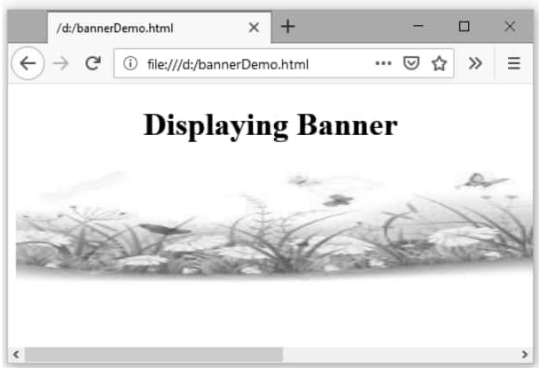
(1) we have created four images and we named them as banner1.jpg, banner2.jpg, banner3.jpg, and banner4.jpg.

(2) Save the images in the same folder as the HTML file which will be used to show the banners. These images can be created by using some graphics tools or image files can be downloaded from Internet.

JavaScript Document

```
<html>
<head>
<script language="Javascript">
 MyBanners=new Array('banner1.jpg','banner2.jpg',
 'banner3.jpg','banner4.jpg');
 banner_count=0;
 function DisplayBanners()
 {
      if (document.images)
      {
           banner_count++;
           if (banner_count==MyBanners.length)
           {
                banner_count=0;
           }
           document.BannerChange.src=MyBanners[banner_count];
           setTimeout("DisplayBanners()",2000);
      }
 }
</script>
<body onload="DisplayBanners()">
<center>
      <h1> Displaying Banner </h1>
      <img src="banner1.jpg" width="900" height="120" name="BannerChange"/>
</center>
</body>
</html>
```

Output



Script Explanation : In above JavaScript,

(1) We have created an array of four image files. Each image is considered as one banner.

(2) In the function **DisplayBanners()** , the images are displayed continuously one by one with some time interval.

(3) The time interval can be set using the built in function **setTimeout,** by calling the function **DisplayBanner** repeatedly.

6.2.2 Linking a Banner Advertisement to URL

In any web site the banner appear mainly for advertising purpose. Hence it is essential to link those banners with their corresponding web site. For example – suppose you are going through some web site and if some banner flashes on your web page that makes an advertisement about online mobile store, then it can tempt you to go through such online mobile store. In such case, this flashing banner must be linked up with the corresponding web site.

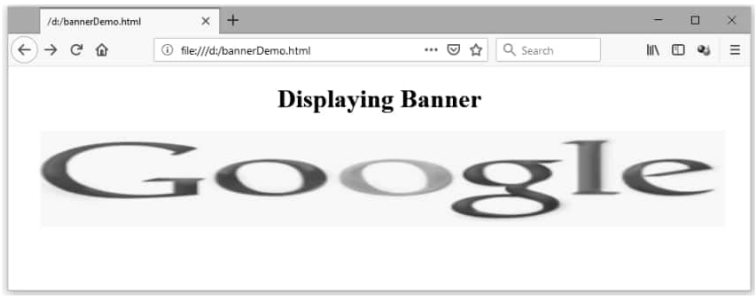
Following is a simple example – in which I have created four banners of four famous web sites of Google, Gmail, Facebook and Wikipedia. If user clicks on the banner of 'Google' then the Google's web page must be displayed.

JavaScript Document

```
<html>
<head>
<script language="Javascript">
 MyBanners=new Array('banner1.jpg','banner2.jpg','banner3.jpg','banner4.jpg');
 MyBannerLinks=new Array('www.google.com/','www.gmail.com/','www.facebook.com/','www.wikipedia.com/')
 banner_count=0;
 function DisplayLinks()
 {
     document.location.href="http://"+MyBannerLinks[banner_count];
 }
 function DisplayBanners()
 {
     if (document.images)
     {
         banner_count++;
         if (banner_count==MyBanners.length)
         {
             banner_count=0;
         }
         document.BannerChange.src=MyBanners[banner_count];
         setTimeout("DisplayBanners()",2000);
     }
 }
</script>
<body onload="DisplayBanners()">
<center>
     <h1> Displaying Banner </h1>
     <a href="javascript: DisplayLinks()">
```

```
    <img src="banner1.jpg" width="900" height="120" name="BannerChange"/></a>
</center>
</body>
</html>
```

Output



If we click on above image then the following output can be obtained

6.3 Slide Show

A slide show is a presentation of a series of still images on a projection screen or electronic display device, typically in a prearranged sequence.

In JavaScript we can create a slide show by using the Array of images. Let us discuss how to create a slide show with simple JavaScript

6.3.1 Creating a Slide Show

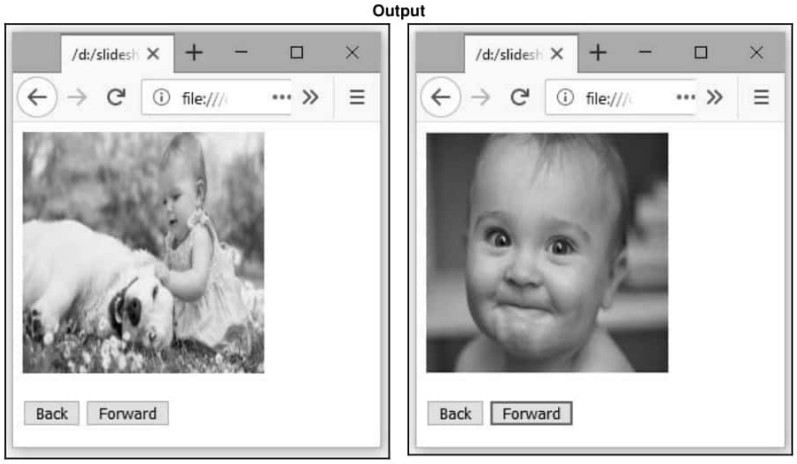
For creation of slide show we need to have some image files. These images can be created using some Graphics tool or some photographs.

In the following JavaScript the slide show is created and the transition from one slide to another can be possible using the **Back** and **Forward** button.

JavaScript Document

```
<!DOCTYPE html>
<head>
        <script language="Javascript">
        MySlides=new Array('slide1.jpg','slide2.jpg','slide3.jpg','slide4.jpg')
        i=0
        function DisplaySlides(SlideNumber)
        {
            i=i+SlideNumber;
            if (i>MySlides.length-1)
            {
                i=0;
            }
                if (i<0)
                {
                    i=MySlides.length-1;
                }
                document.SlideID.src=MySlides[i];
        }
 </script>
 </head>
 <body>
        <img src="slide1.jpg" name="SlideID" width="200" height="200" />
        <br/><br/>
        <input type="button" value="Back" onclick="DisplaySlides(-1)">
        <input type="button" value="Forward" onclick="DisplaySlides(1)">
</body>
</html>
```

Output



Script Explanation : In above JavaScript

(1) An array named **MySlides** is created in which the four image files act as four slides.

(2) In <body> section, the image tag is used to place the first slide.

(3) Then two buttons are placed – one for moving back in slide show and anther is for moving forward in slide show. For each transition(either forward or back the function **DisplaySlides** will be called by passing initial –1 or +1 value.

(4) **DisplaySlides** is a simple function in which the index i of slide is incremented by one. When the slide show reaches to maximum slide number it is reset and the slide show is possible from beginning.

6.4 Menus

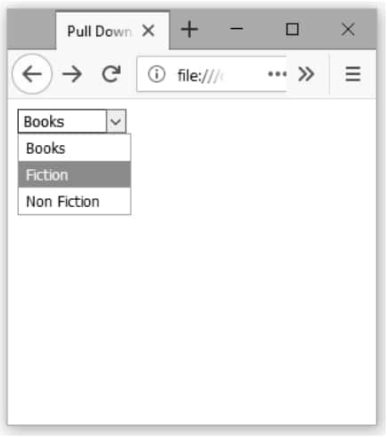
6.4.1 Creating Pulldown Menu

- A web site is normally a collection of various web pages. A visitor to this site navigates from one page to another. If a menu of these web pages is created then it becomes easy for a visitor to select appropriate web page.
- The **<select>** element is used to create a pulldown menu
- The **<option>** tags inside the <select> element define the available options in the list.

JavaScript Document

```
<!DOCTYPE>
<html>
<head>
<title>Pull Down Menu</title>
<script language="Javascript">
function Display(Ch)
{
    MyPage = Ch.options[Ch.selectedIndex].value
    if (MyPage != "")
    {
        window.location = MyPage
    }
}
</script>
</head>
<body onload="document.Form1.MyMenu.
    selectedIndex=0">
    <form action="" name="Form1">
    <select name="MyMenu" onchange="Display(this)">
            <option>Books</option>
            <option
value="Fiction.html">Fiction</option>
            <option value="NonFiction.html">
                Non Fiction</option>
        </select>
    </form>
</body>
</html>
```

Output



On clicking any of the choice in above document, you will be directed to corresponding HTML page.

6.4.2 Dynamically Changing the Menu

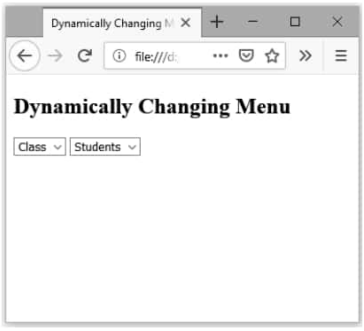
Dynamically changing menu means the items present in the menu are changing automatically. Following is a Javascript example in which two menus are created – one for category of students in the Class and other menu is for displaying names of boys students or names of girl students based on the selection. Thus other menu displaying appropriate names of students is a **dynamic menu.**

JavaScript

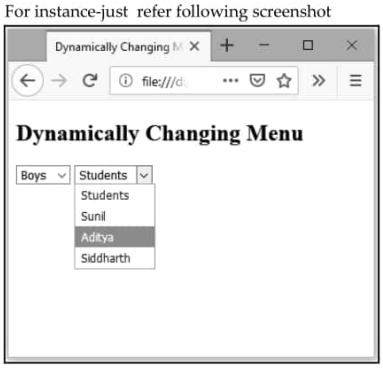
```
<!DOCTYPE html>
<html>
<head>
        <title>Dynamically Changing Menu Options</title>
        <script language="Javascript">
        BoysList = new Array('Sunil','Aditya','Siddharth')
        GirlsList = new Array('Archana','Sharda', 'Swati')
        function DisplayStudents(Class)
        {
            // clearing current options
            for(i=document.Form1.Students.options.length-1;i>0; i--)
            {
                document.Form1.Students.options.remove(i)
            }
            Category = Class.options[Class.selectedIndex].value
            if (Category != "") //somethig is selected
            {
                if (Category == '1')
                {
                    for (i=1; i<=BoysList.length;i++)
                    {
                        //displaying names of boys
                        document.Form1.Students.options[i] =new Option(BoysList[i-1])
                    }
                }
                if (Category == '2')
                {
                    for (i=1; i<=GirlsList.length;i++)
                    {
                        //displaying names of girls
                        document.Form1.Students.options[i] = new Option(GirlsList[i-1])
                    }
                }
            }
        }
 </script>
</head>
<body onload="document.Form1.ClassList.selectedIndex=0">
        <h2> Dynamically Changing Menu</h2>
 <form name="Form1">
    <select name="ClassList" onchange="DisplayStudents(this)">
        <option value="0">Class</option>
        <option value="1">Boys</option>
        <option value="2">Girls</option>
    </select>
    <select name="Students">
```

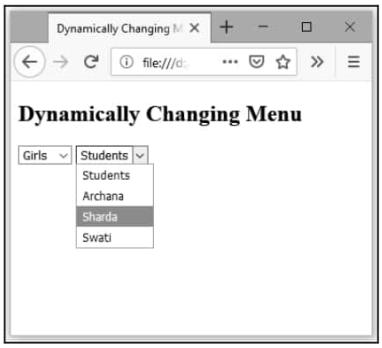
```
            <option value="0">Students</option>
        </select>
    </form>
</body>
</html>
```

Output



Just Select the category of students from 'Class' menu and dynamically the 'Students' menu will change. For instance-just refer following screenshot





6.4.3 Validating Menu Selection

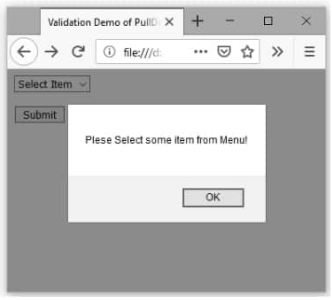
The common problem that user normally creates is – he/she forgets to select particular item from the menu and if such item is required for processing further then it creates severe problem.

This problem can be solved in JavaScript by checking whether the items are selected from the menu or not.

JavaScript Document

```
<!DOCTYPE html>
<html>
<head>
<title>Validation Demo of PullDown Menu</title>
<script language="Javascript">
function ValidateForm(formvalue)
{
    item = formvalue.MyMenu.selectedIndex;
    if(formvalue.MyMenu.options[item].value=="")
    {
        alert("Plese Select some item from Menu!!!");
        return false;
    }
}
</script>
</head>
<body>
    <form action="" name="Form1"
        onsubmit="return ValidateForm(this)">
        <select name="MyMenu">
            <option value="">Select Item</option>
            <option value="1">Desktop</option>
            <option value="2">Laptop</option>
            <option value="3">Server</option>
            <option value="1">Monitor</option>
            <option value="1">iPad</option>
        </select>
        <p>
            <input type="submit" value="Submit" />
            <input type="reset" />
        </p>
    </form>
</body>
</html>
```

Output



Script Explanation : In above JavaScript,

(1) We have defined a pulldown menu in **<body>** tag. This menu is having a name **MyMenu.**

(2) Next we need a **Submit** button to submit the form to the server.

(3) When user clicks the Submit button, it is expected that he/she should have selected some item from the Menu. Validation is a process by which we check if user selects some item or not. For that purpose we have written a function named **ValidateForm.** This function is called on **onsubmit** attribute.

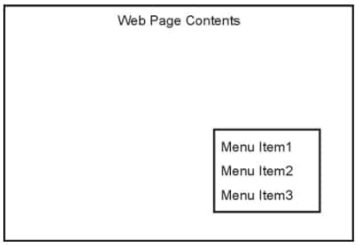
(4) The **ValidateForm** function is defined in **<head>** tag.

(5) In the **ValidateForm** function we pass the form object. Using the form object in variable **formvalue** we can get the value of Selected Index. If user does not select any item then string **formvalue.MyMenu.options[item].value** is blank or null and if it is so then using **alert** window pops up and display the message suggesting user to select some item.

6.4.4 Floating Menu

The JavaScript allows to create dynamic menus which move along with scrolling. Such floating menu will be always visible on screen.

They appear to "float" on top of the page as you scroll. For example –



6.4.5 Chain Select Menu

Chain Select Menu is a kind of menu in which there are more than one set of menus and options selected from the first pulldown menu determines the options for second pulldown menu and options of second pulldown menu determines the options for third pulldown menu.

In above case if we select the country as India,the various states of India will dislayed. If we select Maharashtra as state then in the third menu only Cities in Maharashtra state will be displayed.

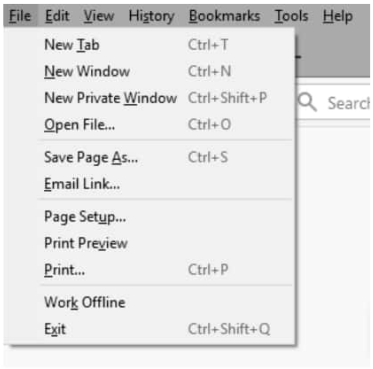
6.4.6 Tab Menu

Tab menus display a one- or two-word description of the menu option within a tab.

A more complete description is displayed below the tab bar as the visitor moves the mouse cursor over the tab. Following fig. represents the tabbed menu.

6.4.7 Popup Menu

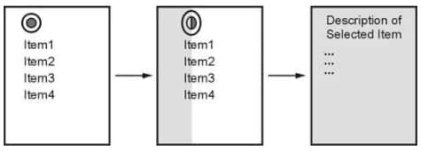
The popup menu contains lower-level menu items that are associated with the top-level menu item. The popup menu appears as you move mouse over each menu item. For example –



This menu is also called as context menu.

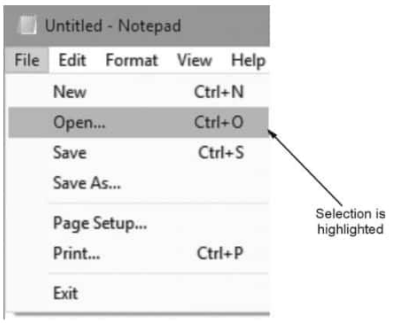
6.4.8 Sliding Menu

These menus are basically off-screen elements that slide into view when you click or tap on something that looks like an arrow, a hamburger icon, or something else that indicates a menu will appear. For example - When you decide to bring up the menu (clicking/tapping on the circle as is the case in our example), the menu magically slides into view :



6.4.9 Highlighted Menu

When the visitor moves the cursor over a menu item, the browser displays a box around the item with a shadow at the bottom of the box. If the visitor selects the item, the highlight shadow appears at the top of the box rather than at the bottom of the box. The highlighted menu is ideal to use to identify a menu option before the visitor actually makes a selection.



6.4.10 Folding a Tree Menu

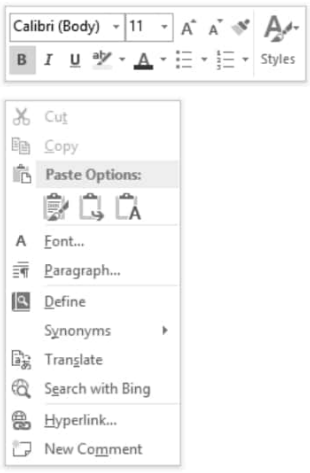
It is a classic menu used in desktop applications. This menu helps to navigate file folders. This tree consists of one or more closed folders each of which appears alongside the folder's name

The tree expands when user clicks a closed folder.

6.4.11 Context Menu

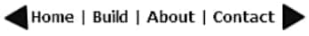
The context menu is a menu that pops up when user clicks right mouse button. The location of the context menu is determined by the position of mouse.

For example –



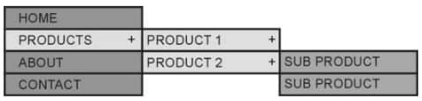
6.4.12 Scrollable Menu

Sometimes there is limited space on the web page for displaying all the menu options. Then in such a case, only limited menu options are displayed and remaining options can be accessed by scrolling left or right. Following is a sample scrollable menu.



6.4.13 Side Bar Menu

The side bar menu displays a menu on the side of the web page. Options on this menu can be linked to other web pages or to other menu options.



Visitors can link to other menus by moving the mouse cursor over a menu item. The menu that is associated with that item pops onto the screen.

Moving the cursor away from the menu item closes the popup menu, and the side bar menu remains on the screen.

6.5 Protecting Web Page

- It is possible to view the source code of some developed web site using the right clicking and choosing **View Source Code** option. This way many new developers get the ideas of writing HTML and JavaScript.
- However, this is not an appropriate practice as it leads to the tendency of borrowing the work of someone else without permission. Thus it is essential to protect your web page.
- There are two ways of protecting your web page –

(1) Disabling the right mouse button

(2) Storing the JavaScript on Web server.

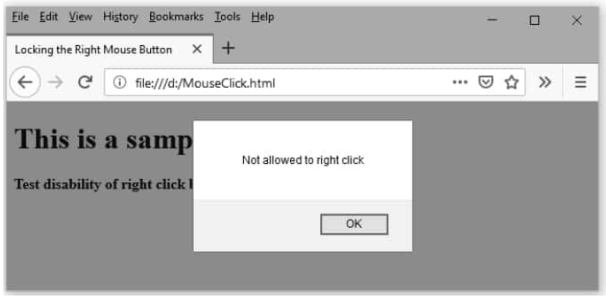
Ex. 6.5.1 : ***Write a JavaScript that disables the right click button and displays the message 'Right click button is disabled'***

Sol. :

```
<!DOCTYPE html>
<html>
 <head>
 <title>Locking the Right Mouse Button</title>
 <script language=JavaScript>
     function RightClickDisable()
     {
         alert('Not allowed to right click');
         return false;
     }
     function InternetExploreBrowser()
     {
         if (event.button==2)
         {
             RightClickDisable();
             return false
         }
     }
     document.oncontextmenu=new Function("RightClickDisable();return false")
 </script>
</head>
<body>
         <h1> This is a sample web page</h1>
         <h4> Test disability of right click button by clicking right button</h4>
</body>
</html>
```

Output

Note that if we right click on the web page, then the message about the disability of right click button is represented.



6.6 Frameworks of JavaScript and its Application

- A JavaScript framework is an application framework written in JavaScript.
- It differs from a JavaScript library in its control flow:a library offers functions to be called by its parent code, whereas a framework defines the entire application design.
- A developer does not call a framework; instead it is the framework that will call and use the code in some particular way.
- For examples : AngularJS, Ember.js, Meteor.js.

Review Questions

1. *What is the use of status bar in a web application ?*
2. *Write a program to display a javascript status bar message whenever your users hover over your hyperlinks.*
3. *Write a JavaScript to scroll the status bar message horizontally.*
4. *What is banner ad ?*
5. *Write a JavaScript to create and display banner.*
6. *Write a JavaScript that illustrates linking of banner advertisement to URL*
7. *What is slide show ?*
8. *Write a JavaScript to show the creation of slide show.*
9. *Explain with suitable example how to create a pull down menu.*
10. *What is dynamic menu ?*
11. *How to create dynamically changing menu in JavaScript.*
12. *What is float menu ?*
13. *Explain chain select menu with necessary illustration.*
14. *What is tab menu ? Explain it with diagrammatic representation.*
15. *Explain folding tree menu.*
16. *What is the use of scrollable menu ?*
17. *Write a JavaScript that disables the right click button and displays the message 'Right click button is disabled.'*
18. *Write short note on – Frameworks of JavaScript and its applications.*

❑❑❑

SOLVED SAMPLE TEST PAPER - I
Client Side Scripting Language
T.Y. Diploma (Sem - V) Elective

Computer Engg./IT Program Group (CO/CM/IF/CW)

Time : 1 Hour] **[Total Marks : 20**

Instructions :

(1) All questions are compulsory.

(2) Illustrate your marks with neat sketches wherever necessary

(3) Figures to the right indicate full marks.

(4) Assume suitable data if necessary.

(5) Preferably, write the answers in sequential order.

Q.1 Attempt any FOUR. **[8]**

a) *What are arithmetic and logical operators used in JavaScript ?* **(Refer section 1.4)**

b) *How will you define function ?* **(Refer section 2.2.1)**

c) *What is string ?* **(Refer section 2.4)**

d) *What is form event ?* **(Refer section 3.2)**

e) *What is the use of Text component ?* **(Refer section 3.1.3)**

f) *What is the use of switch case statement ?* **(Refer section 1.6)**

Q.2 Attempt any THREE. **[12]**

a) *Write a script that reads an integer and displays whether it is a prime number or not.* **(Refer example 1.9.3)**

b) *Write a JavaScript to call a function with argument for addition of two numbers.* **(Refer section 2.3.1)**

c) *Explain - form objects and elements.* **(Refer section 3.3)**

d) *Write a JavaScript to illustrate keyboard event.* **(Refer section 3.2.2)**

e) *Write a JavaScript to convert a string to number.* **(Refer section 2.4.7)**

f) *Explain the use of continue statement with the help of example.* **(Refer section 1.7.5)**

❑❑❑

SOLVED SAMPLE TEST PAPER - II
Client Side Scripting Language
T.Y. Diploma (Sem - V) Elective

Computer Engg./IT Program Group (CO/CM/IF/CW)

Time : 1 Hour] **[Total Marks : 20**

Instructions :

(1) All questions are compulsory.

(2) Illustrate your marks with neat sketches wherever necessary

(3) Figures to the right indicate full marks.

(4) Assume suitable data if necessary.

(5) Preferably, write the answers in sequential order.

Q.1 Attempt any FOUR. **[8]**

a) *What are the attributes used to set the position of window ?* **(Refer section 4.2.3)**

b) *Give the syntax for opening a window.* **(Refer section 4.2.1)**

c) *What is frame ?* **(Refer section 5.2)**

d) *What is rollover ?* **(Refer section 5.3)**

e) *What is banner ad ?* **(Refer section 6.2.1)**

f) *What is dynamic menu ?* **(Refer section 6.4.2)**

Q.2 Attempt any THREE. **[12]**

a) *What is the method of setting expiry date of cookie ?* **(Refer section 4.1.5)**

b) *Write a JavaScript to open a new window.* **(Refer example 4.2.1)**

c) *How will you make the roll over more efficient ?* **(Refer section 5.3.4)**

d) *Explain how to set the frame borders invisible.* **(Refer section 5.2.2)**

e) *Write a JavaScript to create a simple pulldown menu.* **(Refer section 6.4.1)**

f) *Explain - sliding menu and scrollable menu* **(Refer sections 6.4.8 and 6.4.12)**

❑❑❑

SOLVED SAMPLE QUESTION PAPER

Client Side Scripting Language

T.Y. Diploma (Sem - V) Elective

Computer Engg./IT Program Group (CO/CM/IF/CW)

Time: 3 Hours] **[Total Marks : 70**

Instructions :

(1) All questions are compulsory.

(2) Illustrate your answers with neat sketches wherever necessary.

(3) Figures to the right indicate full marks.

(4) Assume suitable data if necessary

(5) Preferably, write the answers in sequential order.

Q.1 Attempt any FIVE of the following. **[10]**

a) *Explain any two features of JavaScript.* **(Refer section 1.1)**

b) *What is conditional operator in JavaScript ?* **(Refer section 1.4)**

c) *Define array* **(Refer section 2.1)**

d) *Explain two uses of forms* **(Refer section 3.1)**

e) *What is cookies ?* **(Refer section 4.1.1)**

f) *Define the term - Regular expression.* **(Refer section 5.1)**

g) *What is the use of status bar in a web application ?* **(Refer section 6.1)**

Q.2 Attempt any THREE of the following. **[12]**

a) *Explain the terms - Property, Method, Dot Syntax.* **(Refer section 1.2)**

b) *Explain six types of values in JavaScript.* **(Refer section 1.3)**

c) *Explain the use of push and pop functions.* **(Refer section 2.1.8)**

d) *How will you create password field in a HTML form ?* **(Refer example 3.1.1)**

Q.3 Attempt any THREE of the following. **[12]**

a) *Explain how to create cookies in JavaScript.* **(Refer section 4.1.2)**

b) *Explain any four commonly used methods in regular expression.* **(Refer section 5.1.1)**

c) *What are the methods used most commonly during the rollover ?* **(Refer section 5.3.1)**

d) *Write a JavaScript to create and display banner.* **(Refer section 6.2.1)**

Q.4 Attempt any THREE of the following. **[12]**

a) *Write a JavaScript to display Welcome message in JavaScript.* **(Refer example 1.1.1)**

b) *Explain the scope of variable with the help of programming example* **(Refer section 2.2.4)**

c) *What is the difference between group of checkbox buttons and group of radio buttons ?* **(Refer example 3.1.2)**

d) *Explain the syntax for opening a window.* **(Refer section 4.2.1)**

e) *Explain frameset tag along with the attributes used in it.* **(Refer section 5.2.1)**

Q.5 Attempt any TWO of the following. **[12]**

a) *Develop a JavaScript to generate 'ARMSTRONG NUMBERS' between the range 1 to 100. [Eg : 153 is an Armstrong number, since sum of the cube of the digits is equal to the number i.e. [$1^3 + 5^3 + 3^3 = 153$]* **(Refer example 1.9.1)**

b) *Write a short note - Object as associative array in JavaScript* **(Refer section 2.1.9)**

c) *Write a JavaScript to create three categories - Fruit, Flower and Colour. Based on the selection of category, the items in the option list must get changed.* **(Refer example 3.5.1)**

Q.6 Attempt any TWO of the following. **[12]**

a) *Write a JavaScript in which when user rolls over the name of fruit, the corresponding image should be displayed. Along with the appropriate image display, additional window should pop up displaying the benefit of each fruit.* **(Refer example 5.3.2)**

b) *What is the use of setTimeout() function ? Write a JavaScript to illustrate it.* **(Refer section 4.2.11)**

c) *Write a JavaScript that disables the right click button and displays the message 'Right click button is disabled'* **(Refer example 6.5.1)**

❑❑❑

Books available @

MUMBAI

Student Agencies Pvt. Ltd.
Ph. - 022 - 40496161 / 31

Vidyarthi Sales Agencies
Ph. - 022 - 23829330
022 - 23851416 / 23867279

Bharat Sales
Ph. - 022 - 23821307 / 7580

AHMEDNAGAR

Shripad Granth Bhandar
Ph. - 9922664979

SANGAMNER

Amrut Book Stall
Ph. - 9850663354

NARAYANGAON

Sunil General Stores
Ph. - 9850725770

KOPARGAON

Mauli Books & General Store
Ph. - 9890451467

Kohinoor Book Stall
Ph. - 9371719996

KARAD

Archana Bazar
Ph. - 7588065287

NASHIK

Rahul Book Centre
Ph. - 0253 - 27424287

Maharashtra Pustak Bhandar
Ph. - 0253 - 2317506

New India Book House
Ph. - 9623123458

Anmol Pustakalaya
Ph. - 9822306289

Anmol Books
Ph. - 0253 - 2505501

Om Pustakalaya
Ph. - 9422246809

Pragati Books & Stationers
Ph. - 7721014040

New Om Stationers
& General Stores
Ph. - 0253 - 2378874

Shri Ganesh Book Depot
Ph. - 9890335806

Eakveera Books
Ph. - 9422754746

Yuvraj Books & Stationers
Ph. - 9850725770

New Anand Pustakalaya
Ph. - 9881540440 / 7720054040

JALGAON

Parvati Traders
Ph. - 9422277738

SANGALI

G. R. Tamhankar
Ph. - 9422041091

Mirji Book House
Ph. - 9422613980

Sankalp Enterprises
Ph. - 9823619823

KOLHAPUR

Granth The Book Word
Ph. - 9922295522

DHULE

Kushal Book Shop
Ph. - 02562 - 280468

LATUR

Gatagat Stores
Ph. - 9422611032

AKOLA

Harne Book Depot
Ph. - 9922669647

AMRAVATI

Navyug Book Stall
Ph. - 7212672664

BARAMATI

Sankalp World
Ph. - 7020837475 / 9765851000

YEOTMAL

Shri Aadhinath Book Centre
Ph. - 9422866072

NAGPUR

Venus Book Centre
Ph. - 9975768899

AURANGABAD

New Arihant Stationers & Book Centre
Ph. - 9850697469 / 9420263132

Umade Book Centre
Ph. - 9881218686

Maya Book Centre
Ph. - 9766660310

Vidhyarthi Bhandar
Ph. - 9764422777

BULDHANA

Agrasen Sales Corporation
Ph. - 9422181542 / 9422882542

PUNE REGION CONTACT
Devendra : 9763209871

NASHIK REGION CONTACT
Shashikant : 8888861609

Email : sales@technicalpublications.org