Chapter- 1
Clint Side Scripting

Introduction to Java Script

☞ Java Script is the scripting language of the web. It is used in millions of web pages to improve the design, validate forms, detect browser, and create cookies and much more.

☞ Java script is the most popular scripting language on the internet and works in all major browsers such as Internet Explorer, Mozilla, Firefox, Netscape and Opera.

☞ Java Script is a lightweight programming language. It consists of lines of executable computer code. It is usually embedded directly into HTML pages. It can be used without purchasing a license.

Characteristics of Java Scripts

☞ The characteristics of java script are as follows:

1. **Java Script gives HTML designers a programming tool**

☞ HTML authors are normally not programmer, but java script is a scripting language with a very simple syntax. Almost anyone can put small "snippets" of code into their HTML pages.

2. **Java Script can put dynamic text into an HTML Pages**

☞ A java script document like this: document.write("<h1>"+name+"</h1>") can write a variable text into an HTML page.

3. **Java Script can react to events**

☞ A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element.

4. **Java Script can read and write HTML element**

☞ A JavaScript can read and change the content of an HTML element.

5. **Java Script can be used to validate data**

☞ A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing.

6. **Java Script can be used to detect the visitor's browser**

☞ A JavaScript can be used to detect the visitor's browser.

7. **Java Script can be used to create cookies**

☞ A JavaScript can be used to store and retrieve information on the visitor's computer.

Advantages and disadvantages of Client-side Scripting

Advantages of Client-side Scripting

i. It immediately response to users
ii. Through this more responsive design and interaction with the user is possible.
iii. It does not need to send requests to the server.
iv. Loading time is fast compared to other models.

v. It improves usability in browsers
vi. It allows more control over the behavior and presentation of the web widget tools.
vii. It is reusable.
viii. Allow for more interactivity by immediately responding to user's action.
ix. My improve the usability of web sites for users whose browsers support scripts.
x. Can give developers more control over the look and behavior of their web widgets.
xi. Are reusable and obtainable from many free resources.

Disadvantages of client-side scripting

i. Not all browsers support scripts, therefore, users might experience errors if no alternatives.
ii. Different browsers and browser versions support scripts differently, thus more quality assurance testing is required.
iii. More development time and effort might be required (if the scripts are not already available through other resources).
iv. Developers have more control over the look and behavior of their web widgets, however, usability problems can arise if a web widget looks like a standard control but behaves differently or vice-versa.

How does JavaScript Works?

- ☞ JavaScript in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.
  - ❖ JavaScript in body section will be executed **WHILE** the page load.
  - ❖ JavaScript in the head section will be executed when **CALLED**.

Script in the body Section

- ☞ Script to be executed when the page loads go in the body section. When you place a script in the body section it generate the content of the page.

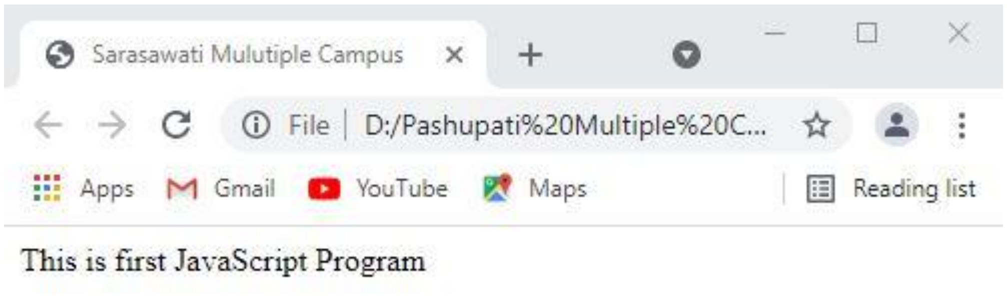
Syntax:

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY>
<script type="text/javascript">







</script>

</BODY>
</HTML>
```

Example:

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY>
<script type="text/javascript">
document.write("This is first JavaScript Program");
</script>

</BODY>
</HTML>
```

Out Put:


Sarasawati Mulutiple Campus
File | D:/Pashupati%20Multiple%20C...
Apps
Gmail
YouTube
Maps
Reading list

This is first JavaScript Program

JavaScript in a Head Section

- Script to be executed when they are called, or when an event is **triggered**, go in the head section. When you place script in the head section, you will ensure that the script is loaded before anyone uses it.
- Script that contains functions go in the head section of the document. Then we can be sure that the script is loaded before the function is called.

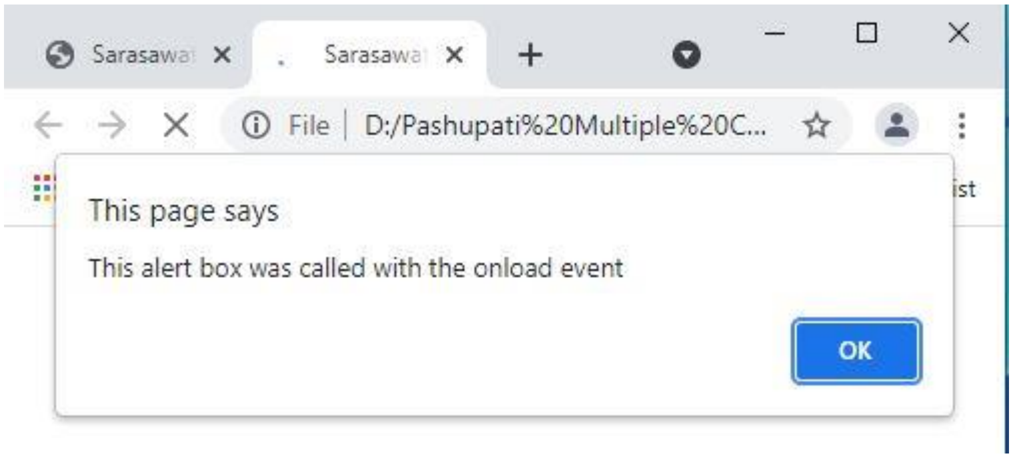
Syntax:

```
<HTML>
<HEAD>


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

</script>

</HEAD>
<BODY>

</BODY>
</HTML>
```

Example:

```
<HTML>
<HEAD>
<TITLE></TITLE>
<script type="text/javascript">
function message()
{
        alert("This is Alert box was called with the onload event");
}
</script>

</HEAD>
<BODY onload="message()">

</BODY>
</HTML>
```

Out Put:



This page says

This alert box was called with the onload event

Note: If you press **OK** then page load will be completed.

Script in the both body and head section

☞ You can place an unlimited number of script in your document, so you can have script in both the body section and head section.

Syntax:

```
<HTML>
<HEAD>
<TITLE></TITLE>
<script type="text/javascript">

</script>

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

```
</script>

</BODY>
</HTML>
```

How to use external JavaScript

- ☞ Sometime you might want to run the same JavaScript on several pages, without having the write the same script on every page.
- ☞ To simplify this, you can write a JavaScript in an external file. Save the external JavaScript file with a **.js** file extension. Eg.Script.js

Syntax:

```
<HTML>
<HEAD>
<TITLE></TITLE>
<script src="xxx.js">
</script>

</HEAD>
<BODY>
</BODY>
</HTML>
```

Note:

- ❖ JavaScript is a case sensitive
- ❖ JavaScript takes whitespace character.

How to break up a code line in JavaScript

- ☞ You can break up code line within a text string with a backslash.

Example:

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY>
<script type="text/javascript">
document.write("This is first JavaScript Program\This message is written when page load");
</script>
```

</BODY>
</HTML>

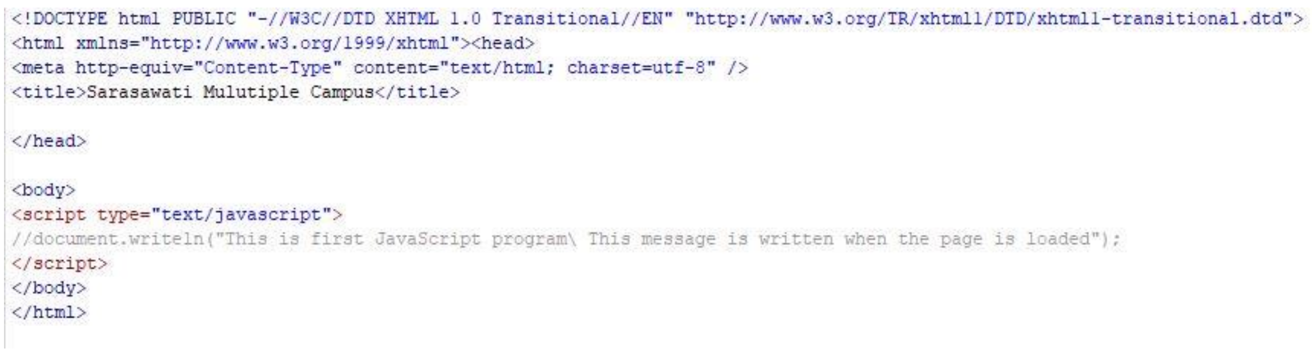
Comments in JavaScript

- ❖ Single line comment
- ☞ You can add comment to your script by using two slashes//(forward shashes).

Example:

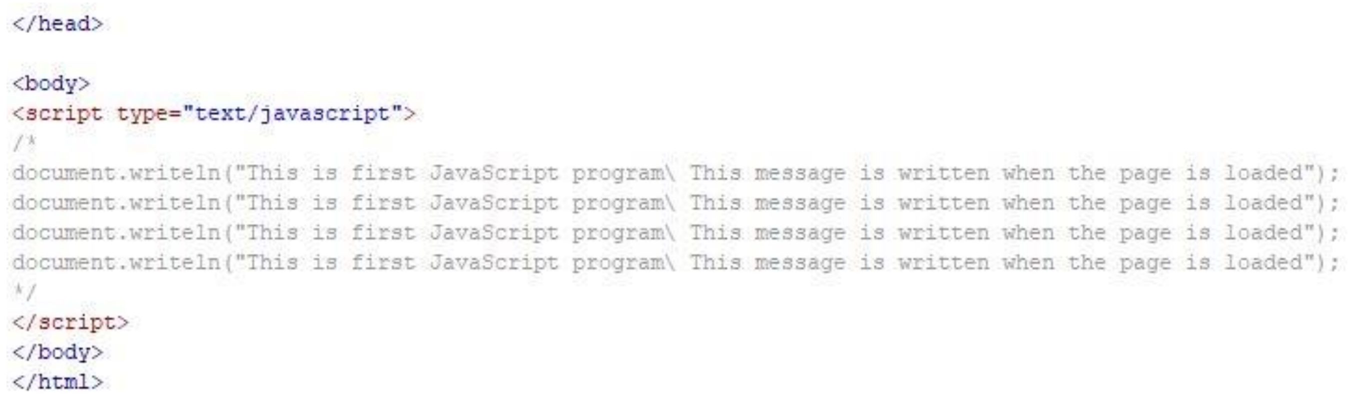
```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY>
<script type="text/javascript">
//document.write("This is first JavaScript Program\This message is written when page load");
</script>

</BODY>
</HTML>
```

Out Put:



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Sarasawati Mulutiple Campus</title>

</head>

<body>
<script type="text/javascript">
//document.writeln("This is first JavaScript program\ This message is written when the page is loaded");
</script>
</body>
</html>
```

- ❖ **Multiline Comment**
- ☞ Multiline comment start with /* and end with */



```
</head>

<body>
<script type="text/javascript">
/*
document.writeln("This is first JavaScript program\ This message is written when the page is loaded");
document.writeln("This is first JavaScript program\ This message is written when the page is loaded");
document.writeln("This is first JavaScript program\ This message is written when the page is loaded");
document.writeln("This is first JavaScript program\ This message is written when the page is loaded");
*/
</script>
</body>
</html>
```

JavaScript Variables

☞ Variable is a container for information you want to store. A variable value can change during the script. You can refer to a variable by name to see its value or to change its value.

Rules for Variable name

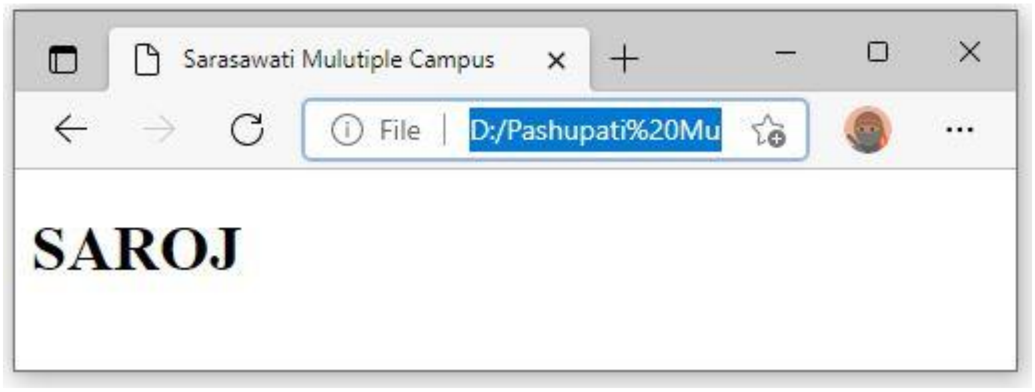
❖ Variable names are case Sensitive. Eg Name, name, NAME all are different
❖ They must begin with a letter or the underscore character. Eg Total, total_marks etc.

Declaration of Variable

☞ Variable can be declare with **var** statement.
Syntax:
var variable_name = some value;
eg. var name="SAROJ"

Example:

```
<HTML>
<HEAD>
<TITLE></TITLE>

</HEAD>
<BODY>
<script type="text/javascript">
var name="SAROJ";
document.write("<h1>"+name+"</h1>");
</script>

</BODY>
</HTML>
```
Out Put:



SAROJ

NOTE:

You can also create a variable without the **var** statement.

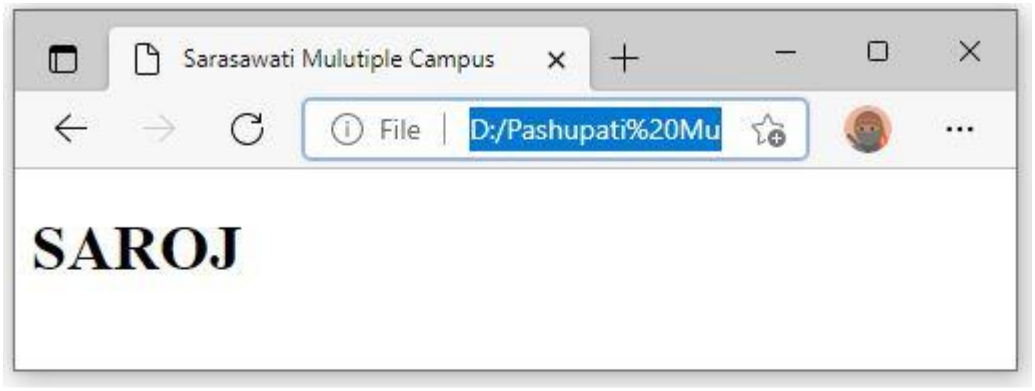
Eg. name="SAROJ"

Example:

Example:

```
<HTML>
<HEAD>
<TITLE></TITLE>

</HEAD>
<BODY>
<script type="text/javascript">
name="SAROJ";
document.write("<h1>"+name+"</h1>");
</script>

</BODY>
</HTML>
```

Out Put:



SAROJ

Lifetime of Variables

- ☞ When declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called **local variables**. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.
- ☞ If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

CONDITIONAL STATEMENT

- ☞ Conditional statement is use to regulate the flow of script.
- ☞ In JavaScript we have the following conditional statements.

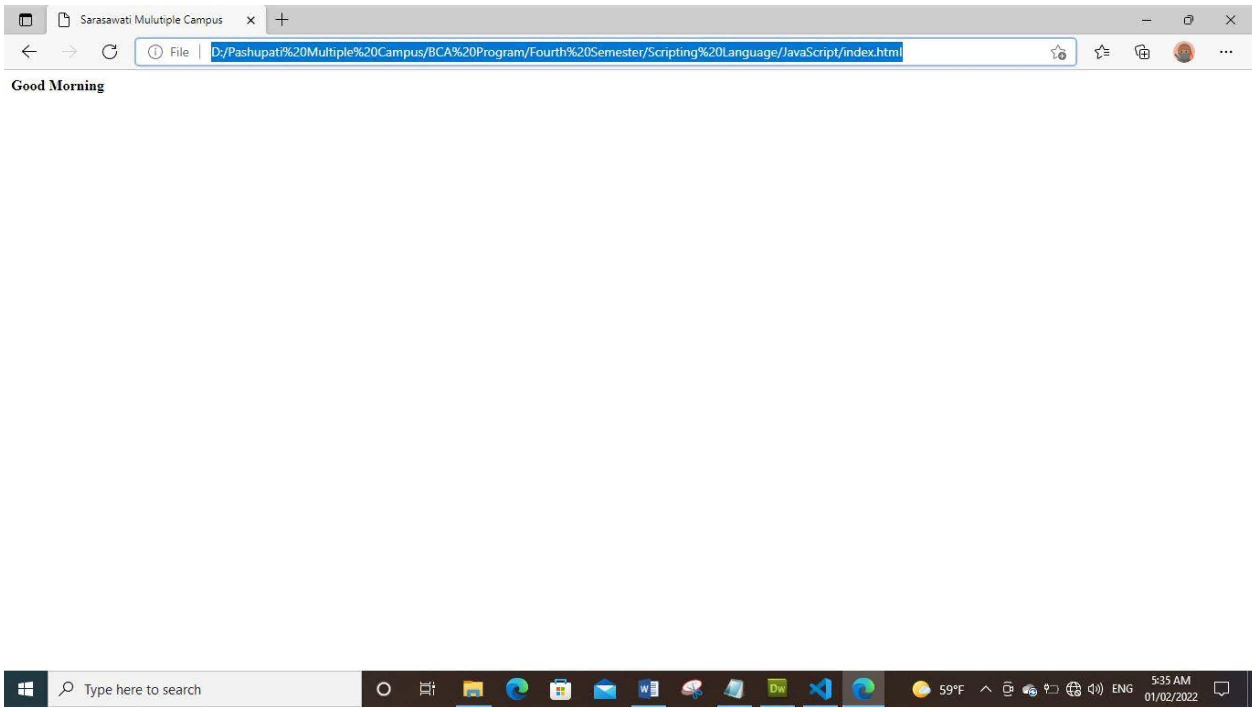
1. **if statement**
    - ☞ When the given condition is true it print the statement otherwise it skips the statement.

      Syntax:
      ```
      if(condition)
      {
              //code to be executed if condition is truce.
      }
      ```

Example: Write a "Good Morning" greeting if the time is less than 12.

```
<HTML>
<HEAD>
<TITLE></TITLE>

</HEAD>
<BODY>
<script type="text/javascript">
var d= new Date();
var time=d.getHours();
if(time<12)
{
        document.write("<b>Good Morning</b>");
}
</script>

</BODY>
</HTML>
```

Out Put:


Sarasawati Mulutiple Campus
File
D:/Pashupati%20Multiple%20Campus/BCA%20Program/Fourth%20Semester/Scripting%20Language/JavaScript/index.html
Type here to search
59°F
ENG
5:35 AM
01/02/2022

Good Morning

2. **if-else statement**
    - ☞ When the given condition is true if part will be execute otherwise else part will be execute.

Syntax:

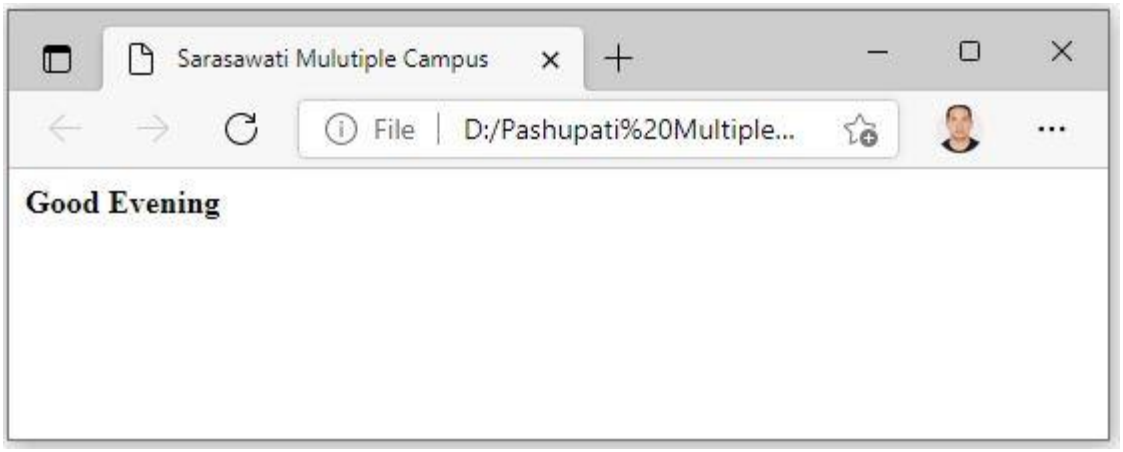
```
if(condition)
{
        //statement-1
}
else
{
        //statement-2
}
```

Example:

```
<HTML><head>
```

```
<title>Sarasawati Mulutiple Campus</title>
</head>
<body>
<script type="text/javascript">
var d =new Date();
var time=d.getHours();
if(time<12)
{
  document.write("<b>Good Morning</b");
}
else
{
        document.writeln("<b>Good Evening</b>");
}

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

Out Put:



Good Evening

3. **if...else if... else statement**

☞ Compound if-else statement print only one statement at a time when more than one condition is given.

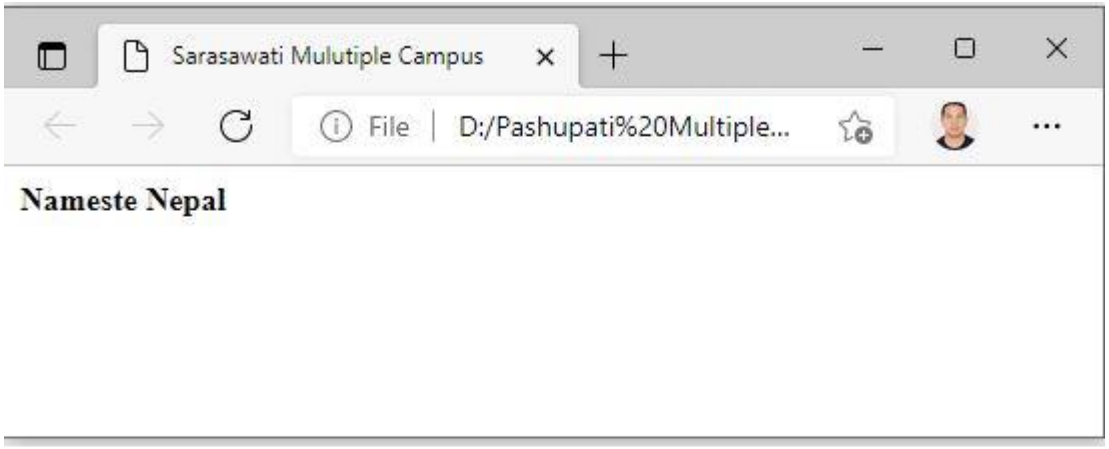
Syntax:

```
if(condition-1)

{

        //code to be executed if condition-1 is true.

}

else if(condition-2)

{

        //code to be executed if condition-2 is true.

}

else if(condition-3)

{

        //code to be executed if condition-3 is true.

}

else

{

        //code to be executed if condition-1, condition-2, condition-3 are not true.

}
```

Example

```
<HTML>
<head>
<title>Sarasawati Mulutiple Campus</title>
</head>
<body>
<script type="text/javascript">
var d =new Date();
var time=d.getHours();
if(time<10)
{
  document.write("<b>Good Morning</b");
}
else if(time>10&& time<16)
{
        document.writeln("<b>Good Day</b>");
}
else
{
        document.writeln("<b>Nameste Nepal</b>");
}

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

Out Put:



Nameste Nepal

4. **switch statement**
   - ☞ You should use the switch statement if you want to select one of many block of code to be executed. Conditional statements in JavaScript are used to perform different actions based on different conditions.

Syntax:

```
switch(exp)

{

        case 1:

        execute code bolock-1

        break;

        case 2:

        execute code block-2

        break;

        default:
```

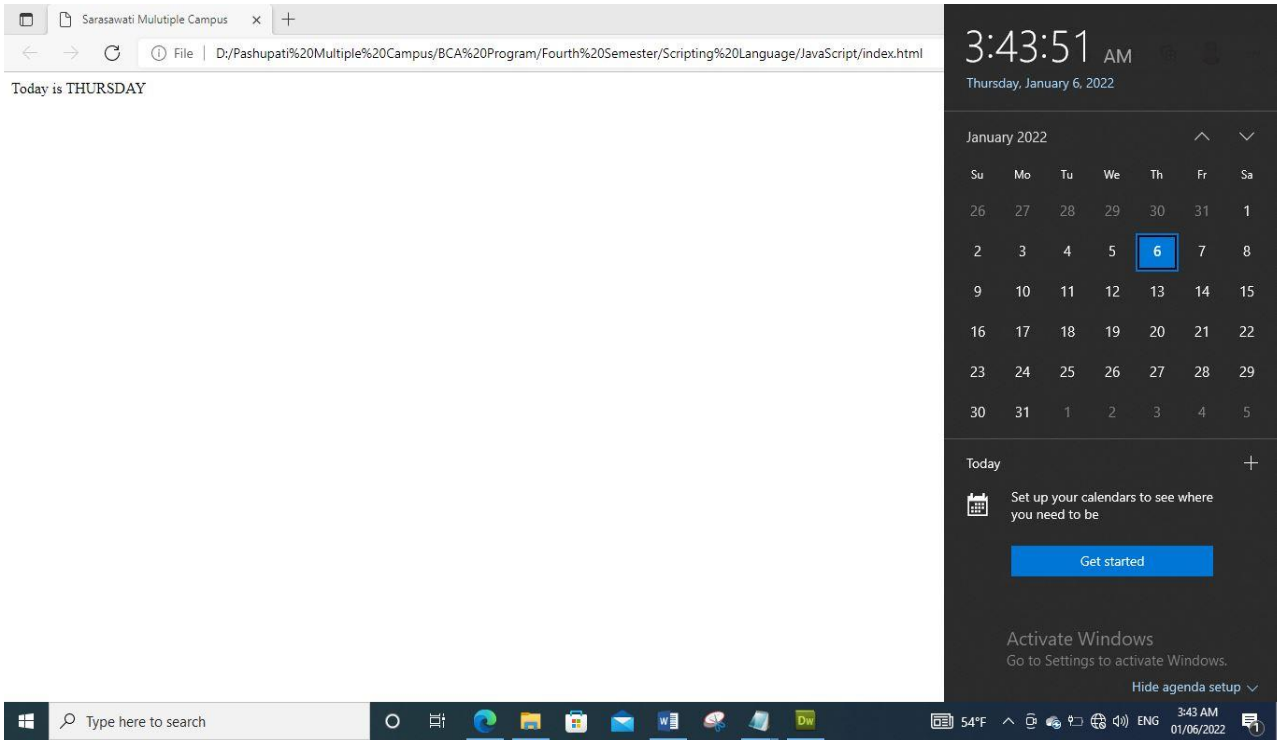
```
        code to be executed if exp is different from case-1 and case-2ss
}
```

Example:

```
<body>
<script type="text/javascript">
var d =new Date();
day=d.getDay();
switch (day)
{
        case 0:
        document.writeln("Today is SUNDAY");
        break;
        case 1:
        document.writeln("Today is MONDAY");
        break;
        case 2:
        document.writeln("Today is TUESDAY");
        break;
        case 3:
        document.writeln("Today is WEDNESDAY");
        break;
        case 4:
        document.writeln("Today is THURSDAY");
        break;
        case 5:
        document.writeln("Today is FRIDAY");
        break;
        case 6:
```

```
        document.writeln("Today is SATURDAY");
        break;
        default:
        document.writeln("Case does not match");

}

</script>
</body>
```

Out Put:



```
Today is THURSDAY
```

Sarasawati Mulutiple Campus
File | D:/Pashupati%20Multiple%20Campus/BCA%20Program/Fourth%20Semester/Scripting%20Language/JavaScript/index.html
3:43:51 AM
Thursday, January 6, 2022
January 2022
Today
Set up your calendars to see where you need to be
Get started
Activate Windows
Go to Settings to activate Windows.
Hide agenda setup
Type here to search
54°F
ENG
3:43 AM
01/06/2022

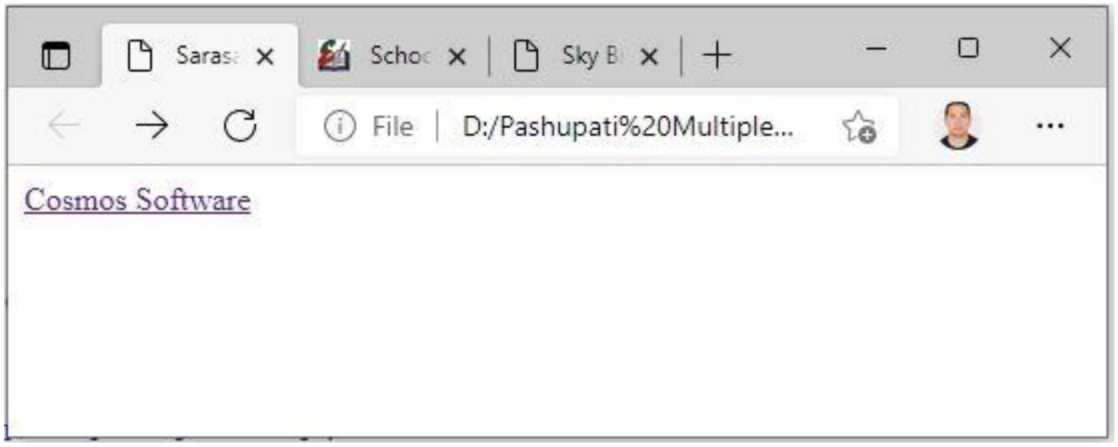
Random link

☞ This example demonstrates a link, when you click on the link it will take you to ***cosmosnepal.net*** or skybridgeacademy.edu.np. There is 50% chance for each of them.

Example:

```
<body>
<script type="text/javascript">
var r=Math.random()
if(r>0.5)
{
        document.writeln("<a href='http://cosmosnepal.net/'>Cosmos Software</a>")
}
else
{
        document.writeln("<a href='http://www.skybridgeacademy.edu.np/'>Sky Bridge
Academy")
}


</script>
</body>
```

Out put:



```
Cosmos Software
```

JAVA SCRIPT OPERATORS

☞ Operator is a symbol which performs operation between two or more operand.

☞ The different types of operator used in JavaScript are:
1. Arithmetic Operator
2. Assignment Operator
3. Comparison Operator
4. Logical Operators
5. String Operator
6. Conditional Operator

1. Arithmetic Operator

Operator	Description	Example	Result

+	It is used to perform addition operation	X=5
		Y=3
		Z=x+y
			8

-	It is used to perform subtraction operation	Z=x-y	2
*	It is used to perform multiplication operation	Z=x*y	15
/	It is used to perform division operation	Z=x/y	1
%	It is used to perform mode or remainder	Z=x%y	2
++	It increment the value by 1	X=2 X++	3
--	It increment the value by 1	X=3 x--	2

2. Assignment Operator

Operator	Example	Is the same as
=	X=y	X=y
+=	X+=y	X=x+y
-=	x-=y	X=x-y
*=	X*=y	X=x*y
/=	x/=y	X=x/y
%=	X%=y	X=x%y

3. Comparison Operator

Operators	Description	Example
==	It is used to check equality	(x==y),x=5,y=6, returns false

===	Is equal to(checks for both value and type)	X=5
		Y="5"
		X==y , returns true
		X===y, returns false

!=	Is not equal to	5!=6, returns true
>	Is greater than	5>4 , it returns true
>=	Is greater than equal to	5>=6, it returns false
<	Is less than	5<6, it returns true
<=	Is less than or equal to	5<=6, it returns true

4. Logical Operators

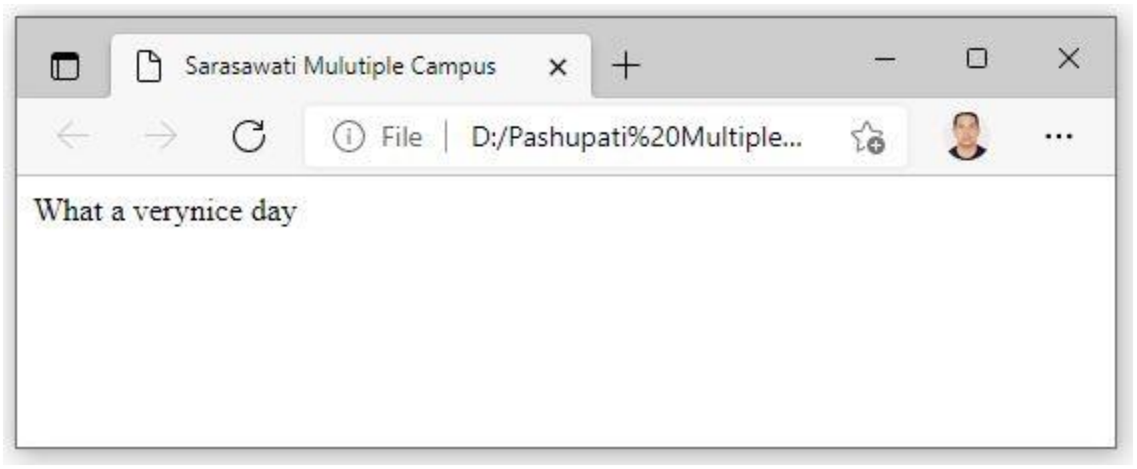
Operators	Description	Example
&&	Logical AND	X=6,y=3 (x<10 && y>1) returns true
||	Logical OR	X=6, y=3 (x==6||y==5) returns true
!	NOT	X=6,y=3 !(x==y) returns true

String Operators

☞ String operator is most often text, for example "Hello World". To stick two or more string variables together, use the + operator.
Example:
```
Txt1="What a very"
Txt2="nice day"
Txt3=Txt1+Txt2

OR
Txt3=Txt1 +" "+Txt2
```

Example:
```
<body>
<script type="text/javascript">
txt1="What a very"
txt2="nice day"
txt3=txt1+txt2
document.writeln(txt3)
</script>
</body>
```

Out Put:



What a verynice day

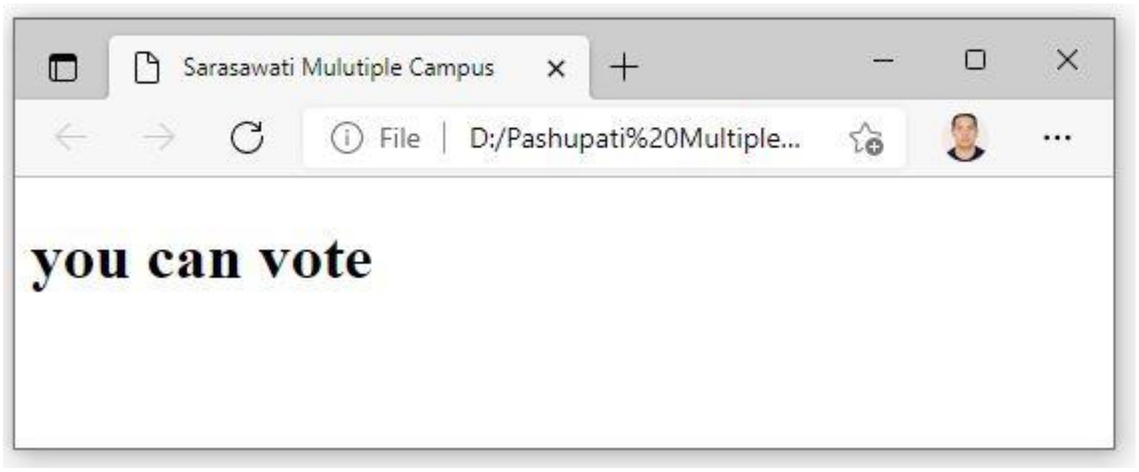
Conditional Operator

- ☞ JavaScript also contains a conditionals operator that assigns a value to a variable to based on some condition.
  Syntax:
  Variable_name=(condition)? Value-1:Value-2

Example:

```
<body>

<script type="text/javascript">

age=20

check_age=(age>=18)?"you can vote": "You cannot vote"

document.writeln("<h1>" + check_age +"</h1>")

</script>

</body>
```

Out Put:



you can vote

JavaScript Popup Boxes

- In JavaScript we can create three kinds of popup boxes.
  i. Alert Box
  ii. Confirm Box
  iii. Prompt Box

I. Alert Box

- An alert box is often used if you want to make sure information comes through to the user.
- When an alert box pops up, the user will have to c lick "OK" to proceed.

Syntax:

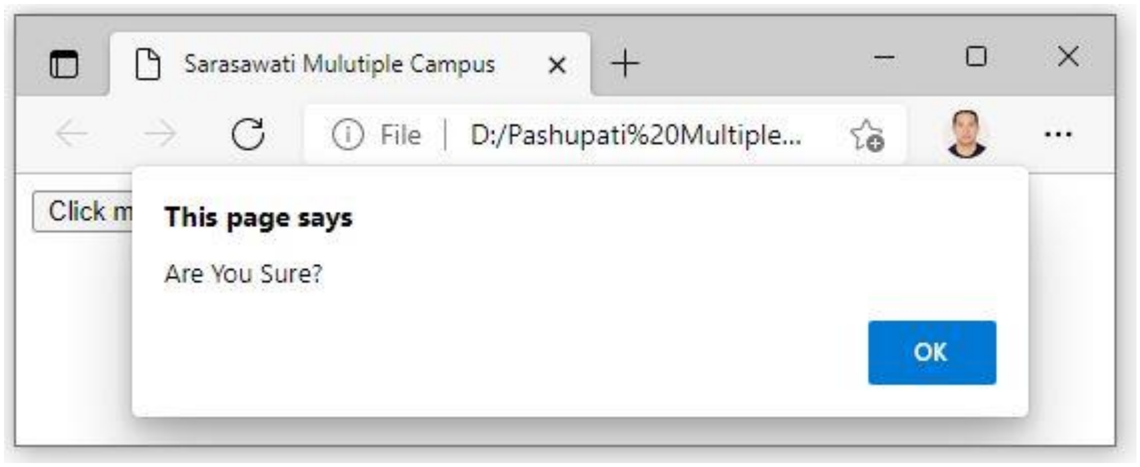
alert("Message")

Example

```
<title>Sarasawati Mulutiple Campus</title>
<script type="text/javascript">
function display_alert()
{
        alert("Are You Sure?")
}
</script>
</head>
```

```
<body>
<input type="button" onclick="display_alert()" value="Click me to show Alert" />
</body>
```

Out Put:



Sarasawati Mulutiple Campus

D:/Pashupati%20Multiple...

This page says

Are You Sure?

II. Confirm Box

- ☞ A confirm box is often used if you want the user to verify or accept something.
- ☞ When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- ☞ If the user click "OK", the box returns **true**. If the user clicks "Cancel" the box returns **false**.

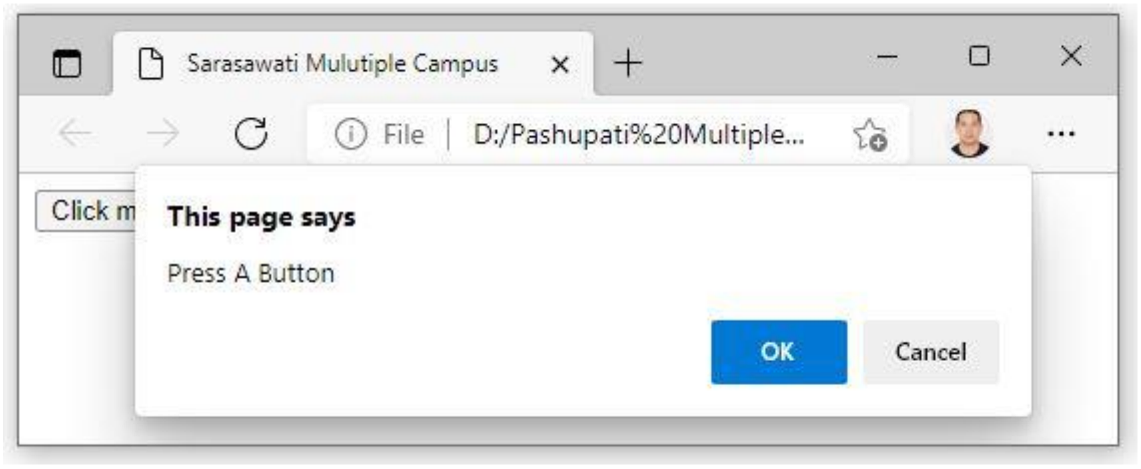
Syntax:

confirm("message text")

Example:

```
<title>Sarasawati Mulutiple Campus</title>
<script type="text/javascript">
function display_confirm()
{
        var r=confirm("Press A Button")
        if(r==true)
        {
                document.writeln("You Pressed OK!")
        }
        else
        {
                document.writeln("You Pressed Cancel")
        }
}
</script>
</head>

<body>
<input type="button" onclick="display_confirm()" value="Click me to show Confirm Box"
/>
</body>
```

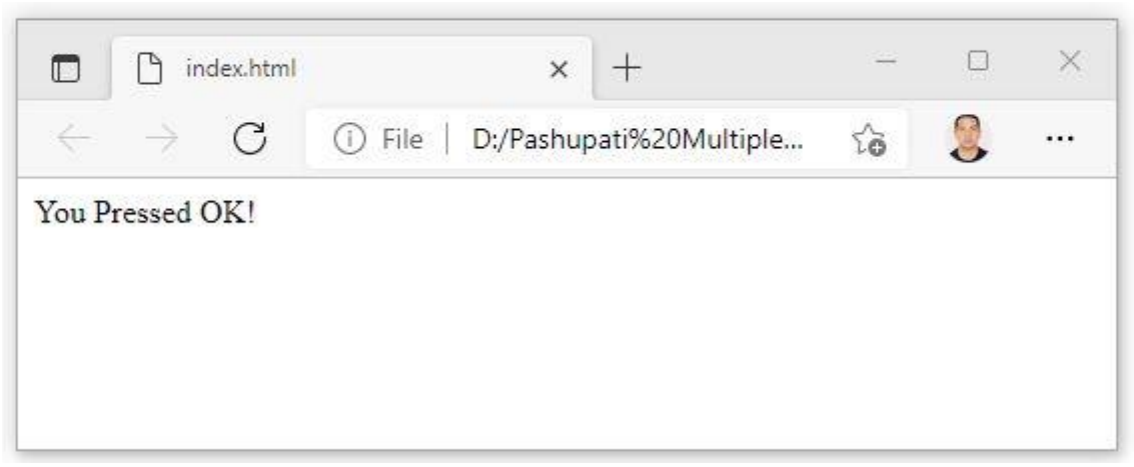
Out Put:



Sarasawati Mulutiple Campus
File | D:/Pashupati%20Multiple...
Click m
OK
Cancel

This page says

Press A Button



You Pressed OK!

III. Prompt Box

- ☞ A prompt box is often used if you want the user to input a value before entering a page.
- ☞ When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering a input value.
- ☞ If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax:

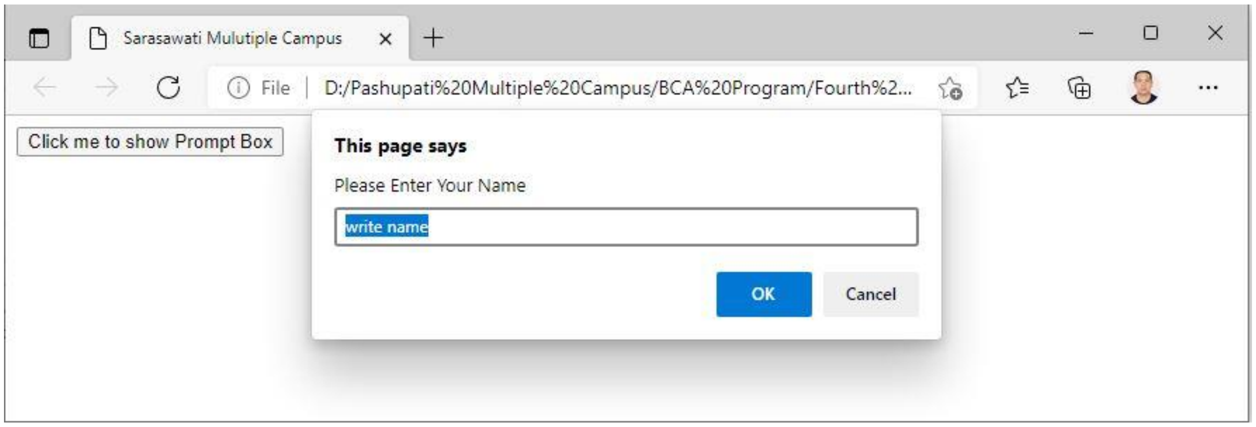
prompt("some text" , "default value")

Example:

```
<title>Sarasawati Mulutiple Campus</title>

<script type="text/javascript">

function display_prompt()

{

        var name=prompt("Please Enter Your Name","write name")

        if(name!=null && name!="")

        {

                document.writeln("Hello" + "<b>"+ name+ "</b>"+"!How are you today?")

        }

}
```

```
</script>
</head>

<body>
<input type="button" onclick="display_prompt()" value="Click me to show Prompt Box" />
</body>
```

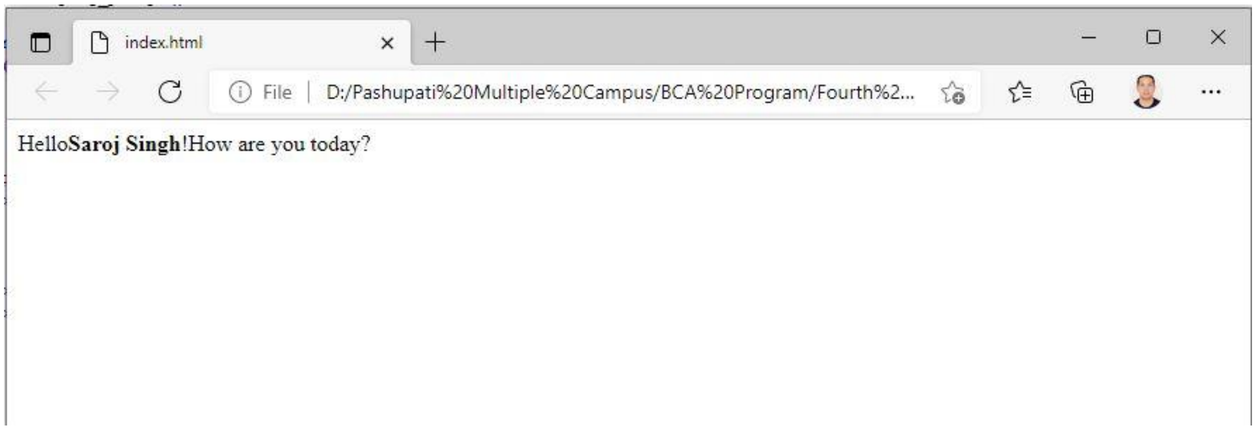
Out Put:


Sarasawati Mulutiple Campus
File
D:/Pashupati%20Multiple%20Campus/BCA%20Program/Fourth%2...
Click me to show Prompt Box
OK
Cancel

This page says

Please Enter Your Name

write name



HelloSaroj Singh!How are you today?

JavaScript Loop

Loop:

- When one statement requires number of time then it is said to be a loop.
- The different types of loop used in JavaScript are:
  I. while loop
  II. do-while loop

III. for loop

I. while loop

☞ while loop print the statement after checking its condition.

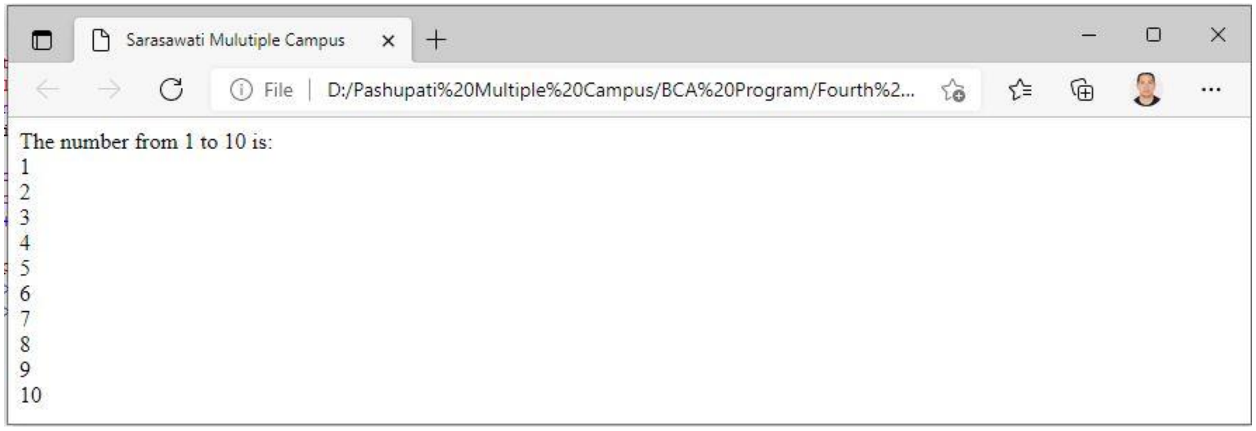
Syntax:

```
while(var<=endvalue)
{
        //code to be executed
}
```

Example:

```
<title>Sarasawati Mulutiple Campus</title>


</head>


<body>

<script type="text/javascript">

var i=1

document.writeln("The number from 1  to 10 is:<br>")

while(i<=10)

{

        document.writeln(""+i)

        document.writeln("<br>")

        i++;

}

</script>

</body>
```

Out Put:

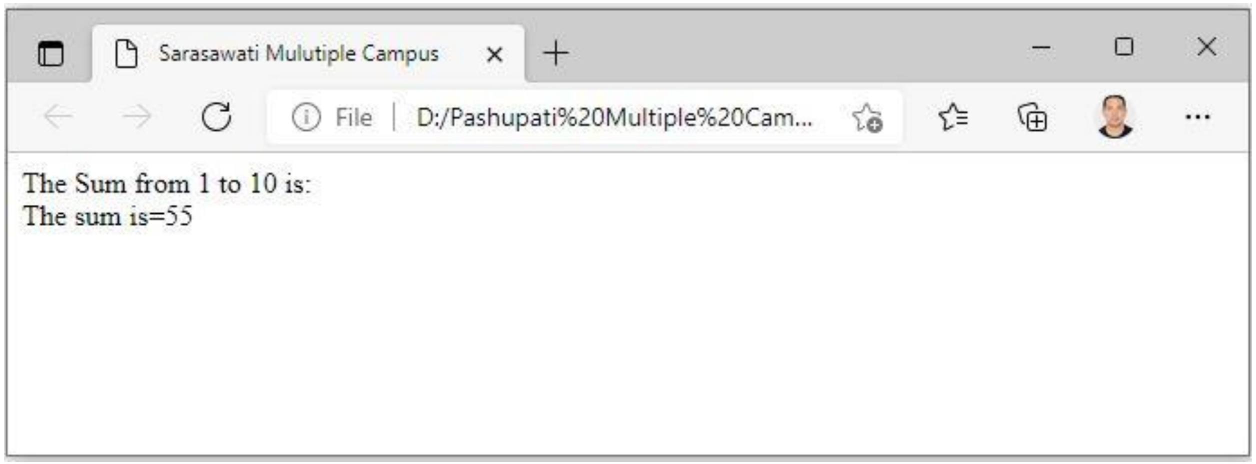

Sarasawati Mulutiple Campus
File
D:/Pashupati%20Multiple%20Campus/BCA%20Program/Fourth%2...

```
The number from 1 to 10 is:
1
2
3
4
5
6
7
8
9
10
```

Exampe-2:

```
<title>Sarasawati Mulutiple Campus</title>

</head>

<body>
<script type="text/javascript">
var i=1
var sum=0
document.writeln("The Sum from 1  to 10 is:<br>")
while(i<=10)
{
        sum=sum+i
        i++;
}
document.writeln("The sum is="+ sum)
</script>
</body>
```

Out Put:



```
The Sum from 1 to 10 is:
The sum is=55
```

II. do-while loop

- do-while loop print the statement at least one before checking its condition.

Syntax:

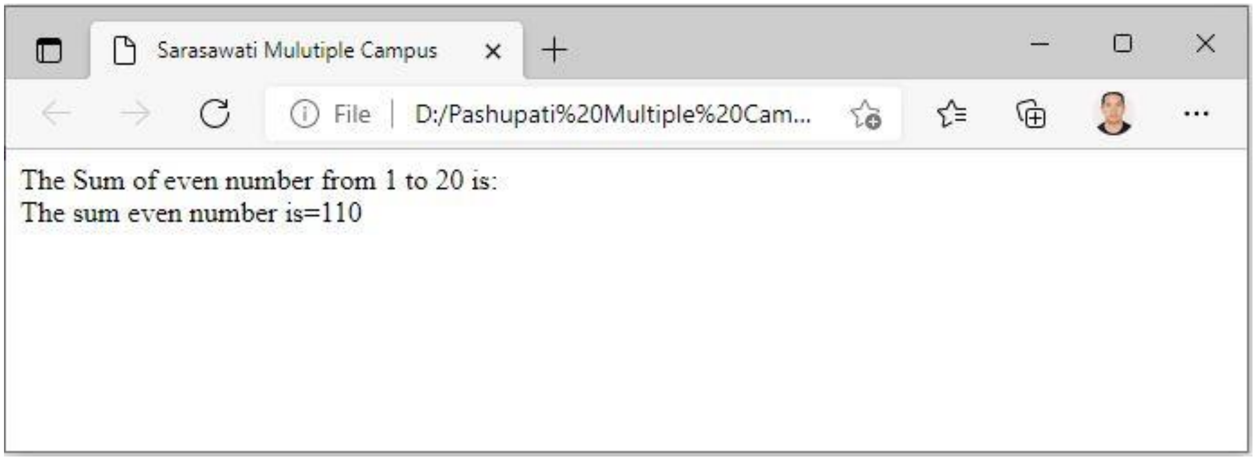
```
do
{
    //code to be executed
}while(var<=endvalue)
```

Example:

```
<body>
<script type="text/javascript">
var i=2
var sum=0
document.writeln("The Sum of even number  from 1  to 20 is:<br>")
do
{
        sum=sum+i
        i=i+2
}while(i<=20)
```

```
document.writeln("The sum even number is="+ sum)
</script>
</body>
```

Out Put:



```
The Sum of even number from 1 to 20 is:
The sum even number is=110
```

III. for Loop

☞ The for loop is used when you know in advance how many times the script should run.

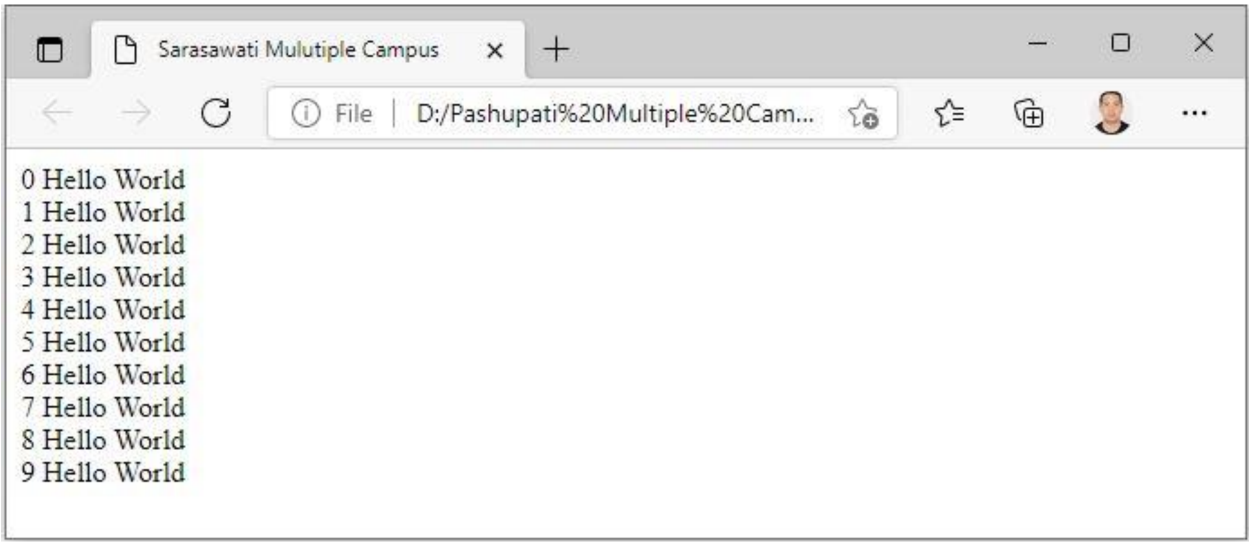
Syntax:

```
for(var=startvalue;  var<=endvalue; var=var+increment)
{
        //code to be executed
}
```

Example:

```
<body>
<script type="text/javascript">
var i
for(i=0;i<10;i++)
{
        document.writeln(i)
        document.writeln("Hello World<br>")
}
</script>
</body>
```

Out Put:

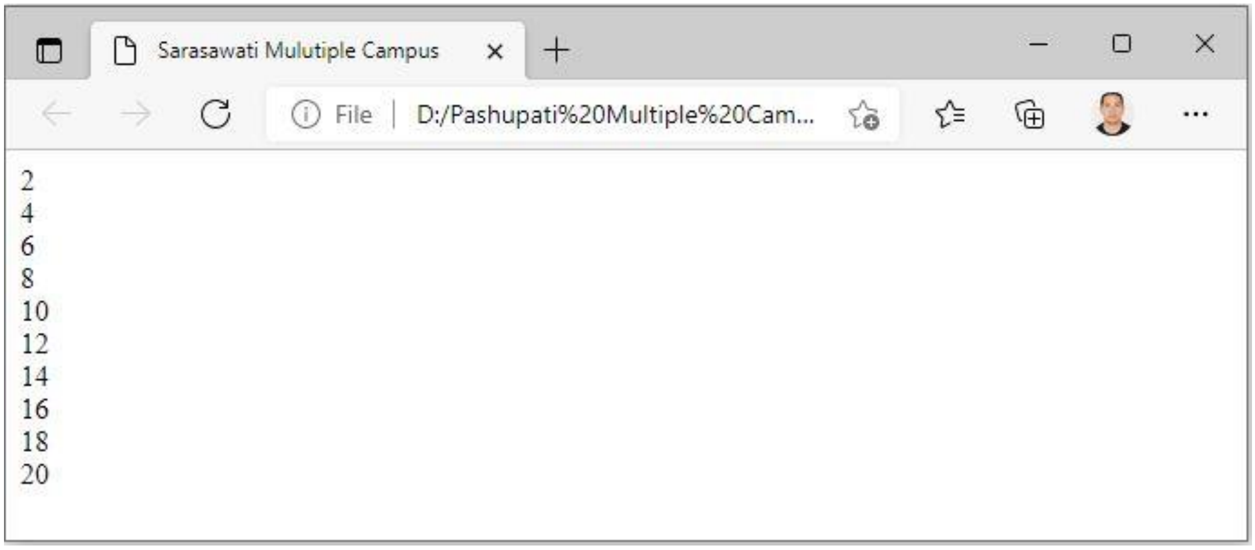


```
0 Hello World
1 Hello World
2 Hello World
3 Hello World
4 Hello World
5 Hello World
6 Hello World
7 Hello World
8 Hello World
9 Hello World
```

Example-2:

```
<body>
<script type="text/javascript">
var n=2
for(i=1;i<=10;i++)
{
        var r=n*i
        document.write(r)
        document.writeln("<br>")
}
</script>
</body>
```

Out Put:

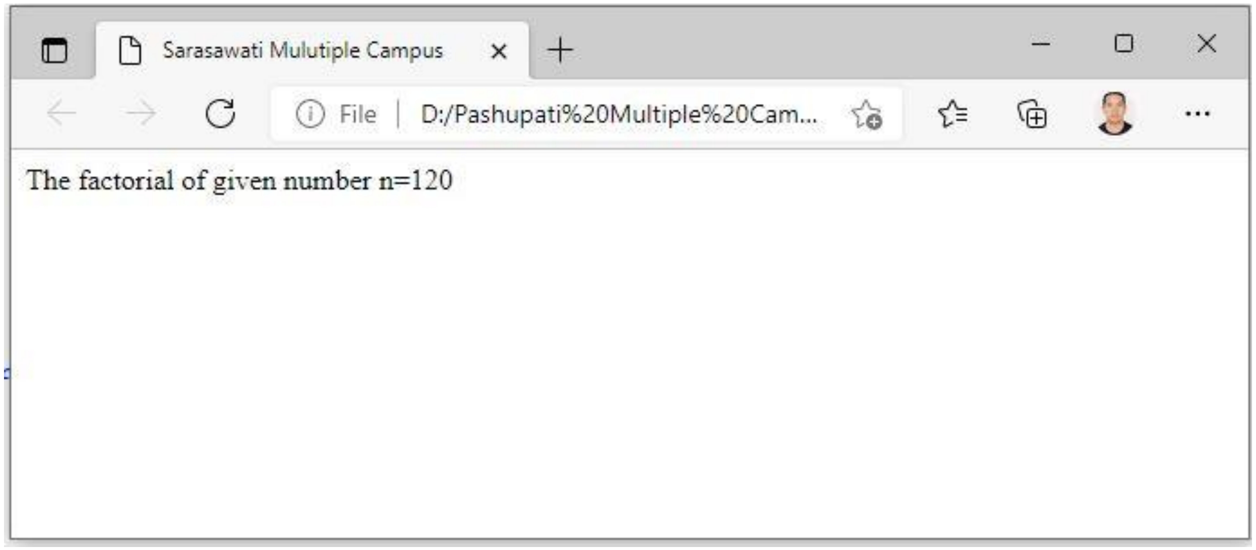


```
2
4
6
8
10
12
14
16
18
20
```

Factorial of given number n

```
<body>
<script type="text/javascript">
var n=5
var fact=1
for(i=n;i>=1;i--)
{
        fact=fact*i



}
document.write("The factorial of given number n="+fact)
</script>
</body>
```

Out Put:


Sarasawati Mulutiple Campus
File | D:/Pashupati%20Multiple%20Cam...

```
The factorial of given number n=120
```

JavaScript break and continue Statement

- There are two special statements that can be used inside loops: break and continue.

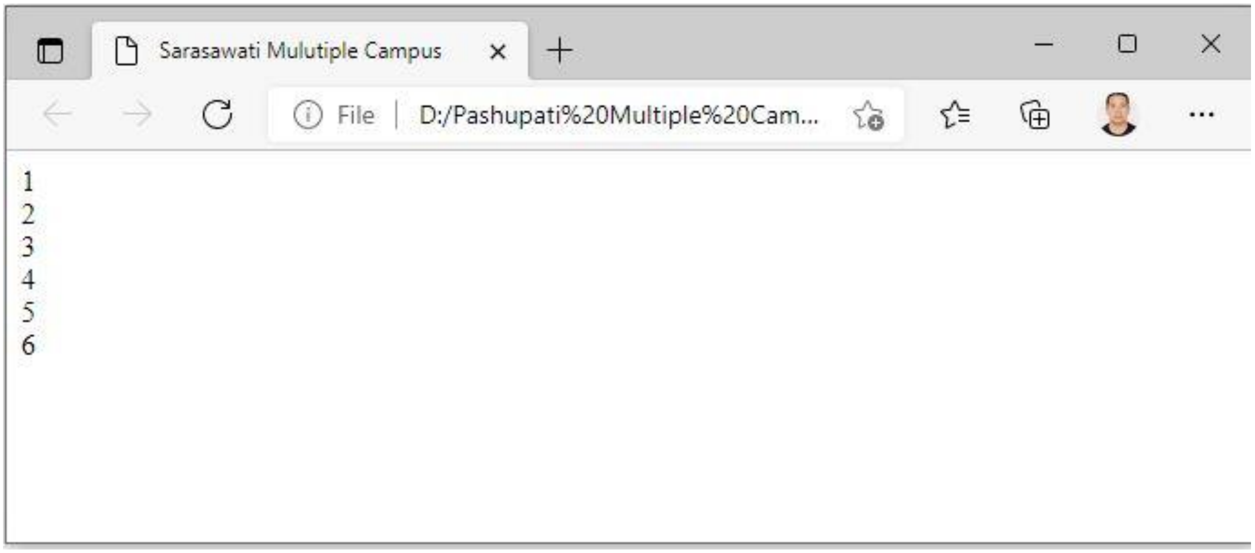
Break statement

- The break command will break the loop and continue executing the code that follows after the loop (if any).

Example:

```
<body>

<script type="text/javascript">


for(i=1;i<=10;i++)

{


        if(i==7)

        {

                break

        }

        document.writeln(""+i)

        document.writeln("<br>")
}

</script>
</body>
```

Out Put:



```
1
2
3
4
5
6
```

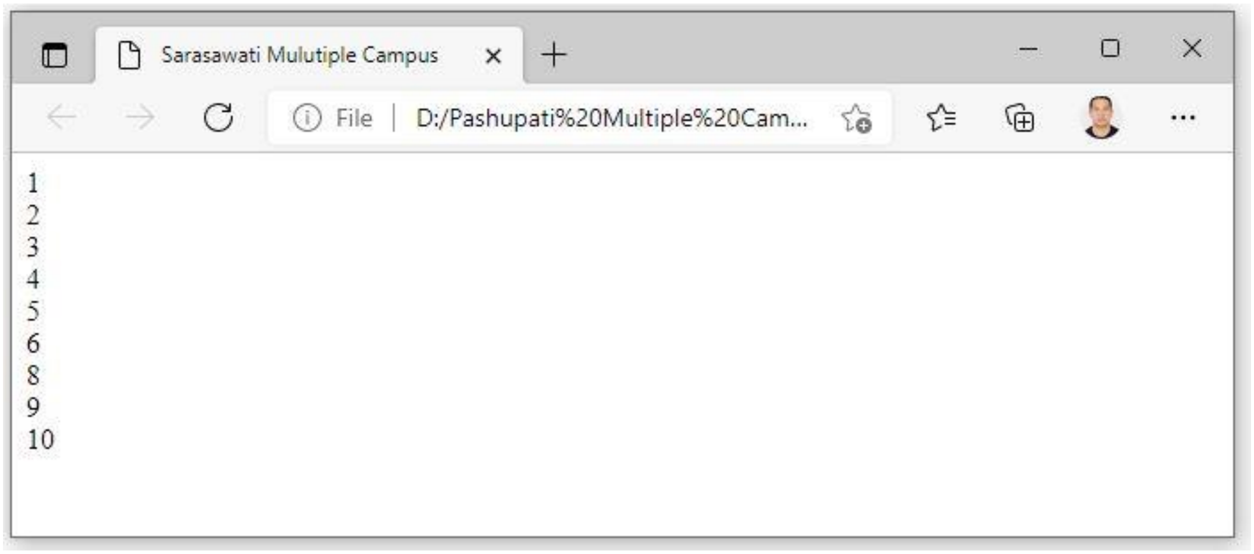
Continue Statement

- ☞ The continue command will break the current loop and continue with the next value.

Example:

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

for(i=1;i<=10;i++)
{

        if(i==7)
        {
                continue
        }
        document.writeln(""+i)
        document.writeln("<br>")

}

</script>
</body>
```

Out Put:


Sarasawati Mulutiple Campus
File | D:/Pashupati%20Multiple%20Cam...

```
1
2
3
4
5
6
8
9
10
```

JavaScript for….in Statement

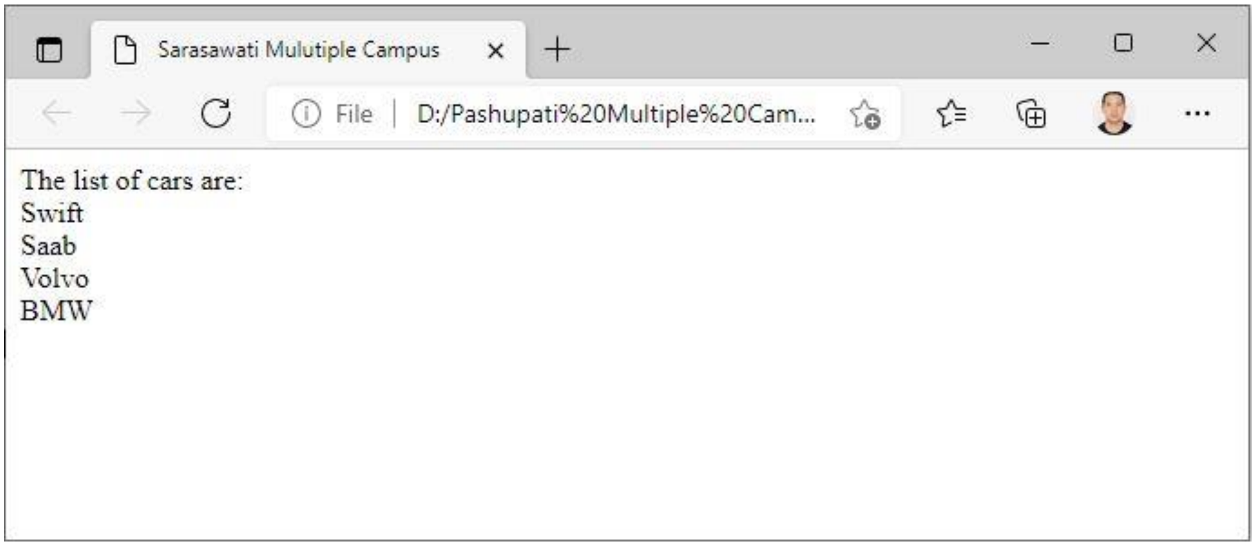
☞ The for… in statement is used to loop through the elements of an array or through the properties of an object.
☞ The code in the body of the for…in loop is executed once for each element/property.

```
Syntax:
for(variable in object)
{
        //code to be executed
}
```

Example:

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


var x
var mycars=new Array()
mycars[0]="Swift"
mycars[1]="Saab"
mycars[2]="Volvo"
mycars[3]="BMW"
document.writeln("The list of cars are:<br>")
for(x in mycars)
{
        document.writeln(mycars[x]+"<br>")
}


</script>
</body>
```

Out Put:



```
The list of cars are:
Swift
Saab
Volvo
BMW
```

JavaScript Functions

- ☞ A function is a set of instruction which performs some kinds of task.
- ☞ To keep the browser from executing a script as soon as the page is loaded, you can write your script as a function.
- ☞ A function contains some code that will be executed only by an event or by a call to that function.
- ☞ You may call a function from anywhere within the page (or even from other pages if the function is embedded in an external .js file).
- ☞ Functions are defined at the beginning of a page, in the <head> section.
- ☞ Function is a reusable code-block that will be executed by an event, or when the function is called.

How to Define a Function

Syntax:

```
function function_name( var-1, var-2……..var-n)

{

        //some code

}
```

Note:

A function with no parameters must include the parentheses () after the function name.

Eg.

```
function function_name()

{

        //some code

}
```

Return statement

- ☞ The return statement is used to specify the value that is returned from the function. So, functions that are going to return a value must use the return statement.

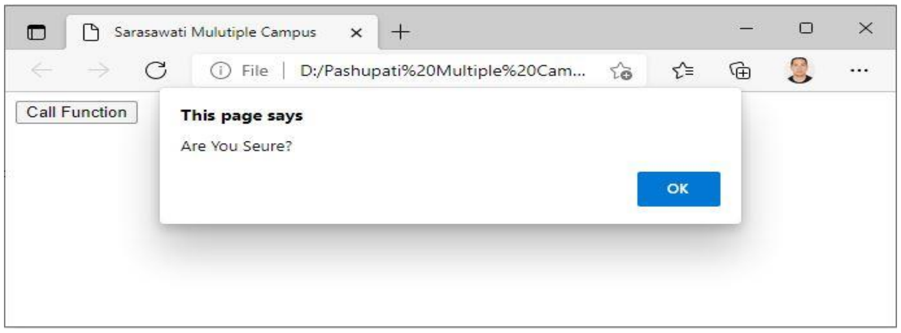
Example:

```
function prod(a,b)

{
        x=a*b
        return x;
}
product=prod(2,3)
```

1. **How to call a function**

```
<title>Sarasawati Mulutiple Campus</title>
<script type="text/javascript">
function displaymessage()
{
        alert("Are You Seure?")
}

</script>
</head>

<body>
<form>
<input type="button" onclick="displaymessage()" value="Call Function" />
</form>
</body>
```

Out Put:



This page says

Are You Seure?

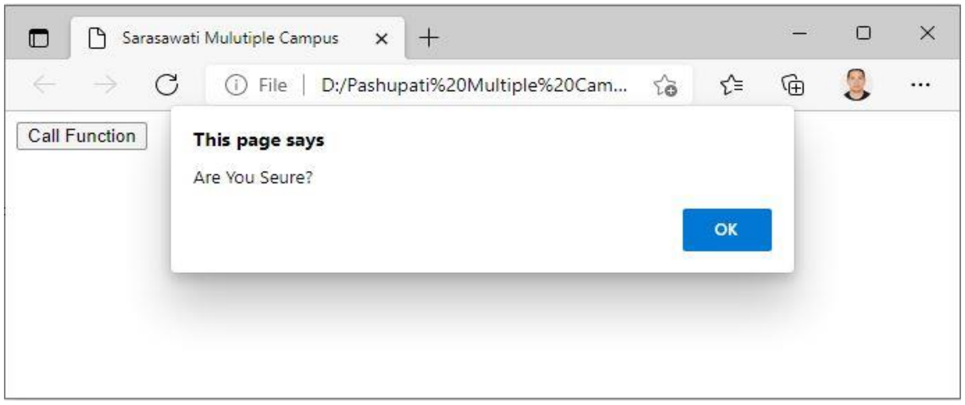
2. **Function with argument**

☞ Function with arguments means how to pass a variable to a function, and use the variable in the function.

Example:

```
<title>Sarasawati Mulutiple Campus</title>
<script type="text/javascript">
function displaymessage(text)
{
        alert(text)
}


</script>
</head>


<body>
<form>
<input type="button" onclick="displaymessage('Are You Seure?')" value="Call Function" />
</form>
</body>
```

Out Put:



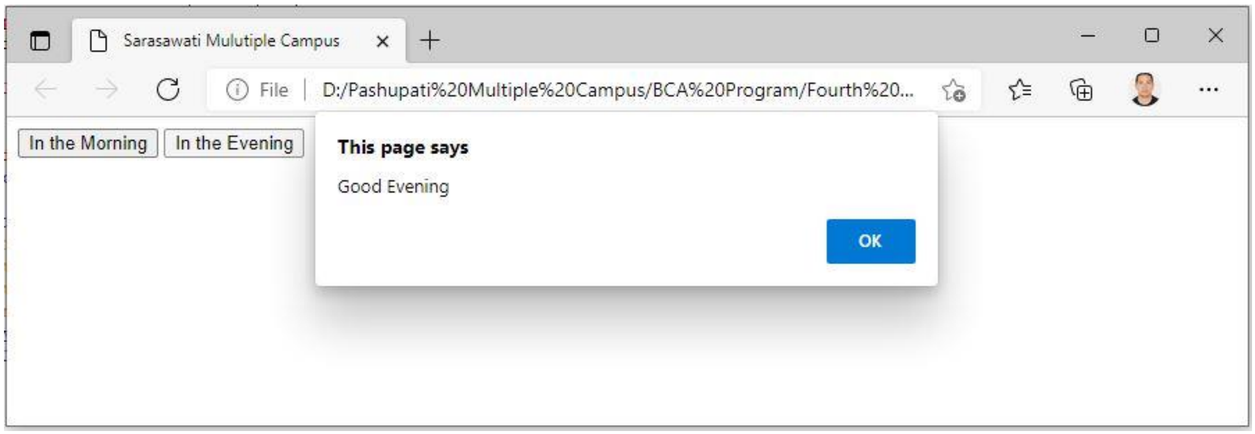
This page says

Are You Seure?

3. **Function with arguments two**

```
<title>Sarasawati Mulutiple Campus</title>
<script type="text/javascript">
function displaymessage(text)
{
        alert(text)
}

</script>
</head>

<body>
<form>
<input type="button" onclick="displaymessage('Good Morning')" value="In the Morning"
/>
<input type="button" onclick="displaymessage('Good Evening')" value="In the Evening" />
</form>
</body>
```

Out Put:


Sarasawati Mulutiple Campus
File
D:/Pashupati%20Multiple%20Campus/BCA%20Program/Fourth%20...
In the Morning
In the Evening
OK

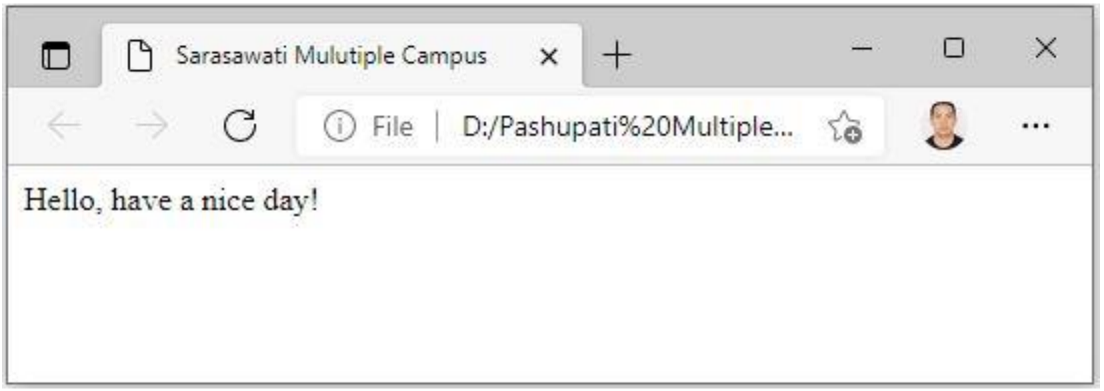
This page says

Good Evening

4. Function that return a value

```
<title>Sarasawati Mulutiple Campus</title>
<script type="text/javascript">
function MyFunction()
{
        return("Hello, have a nice day!")
}

</script>
</head>

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

document.write(MyFunction())
</script>
</body>
```

Out Put:

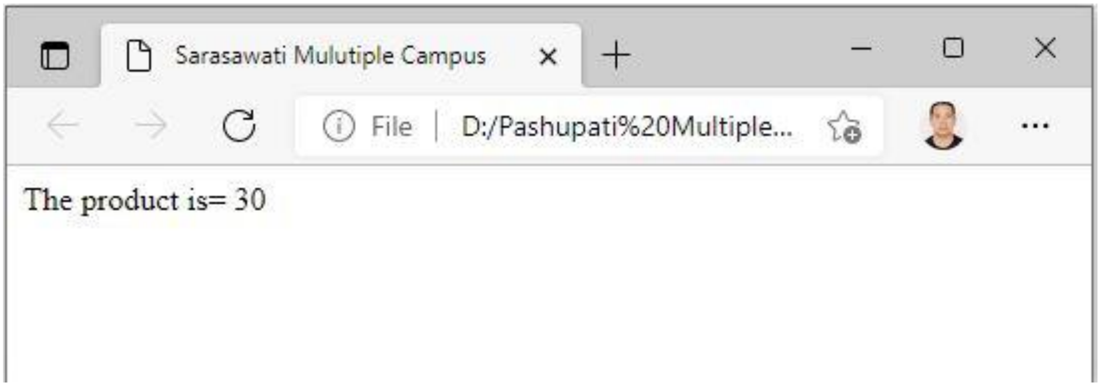


```
Hello, have a nice day!
```

5. Function with argument and that return a value

```
<title>Sarasawati Mulutiple Campus</title>
<script type="text/javascript">
function product(a,b)
{
    return a*b
}

</script>
</head>

<body>
<script type="text/javascript">
document.writeln("The product is=")
document.write(product(5,6))
</script>
</body>
```

Out Put:



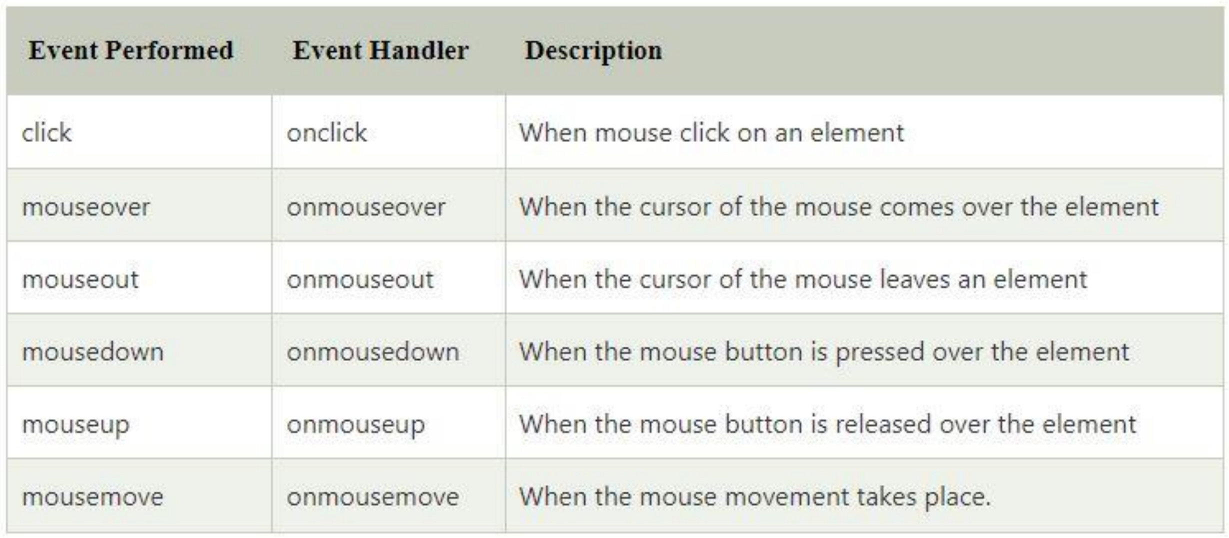
```
The product is= 30
```

JavaScript Events

☞ Events are actions that can be detected by JavaScript.

☞ By using JavaScript, we have the ability to create dynamic web pages. Events are action that can be e detected by JavaScript.
☞ Every element on a web page has certain events which can trigger JavaScript functions. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button.
☞ We define the events in the HTML tags.
☞ Some example of events are: mouse click, an image loading, mouse over, selecting an input box, submitting an HTML form etc.
☞ Some of the HTML events and their event handlers are:

1. **Mouse Events**



Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

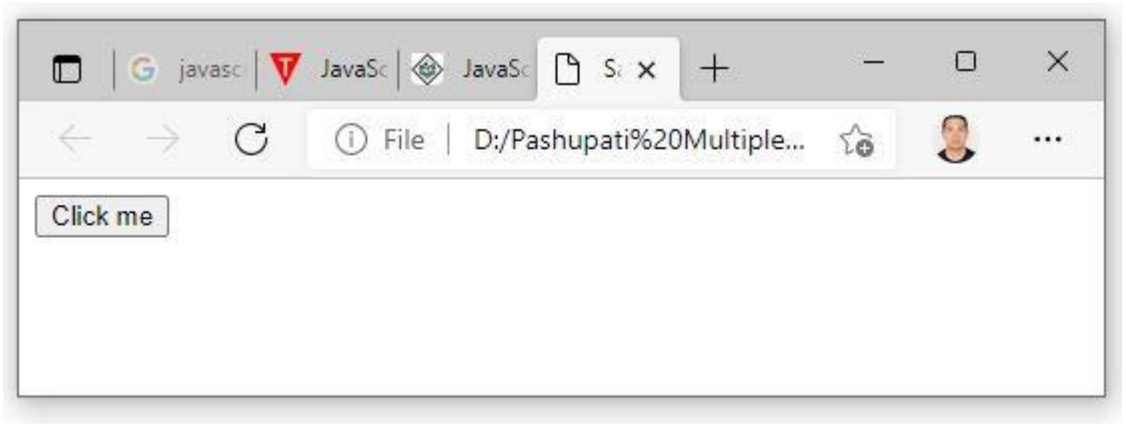
Example-1: Click event

JavaScript to demonstrate Click event

```
<title>Sarasawati Mulutiple Campus</title>
<script type="text/javascript">
function clickevent()
{
        document.writeln("This is click event")
}


</script>
</head>
```

```
<body>
<form>
<input type="button" onclick="clickevent()" value="Click me" />

</form>
</body>
```

Out Put:



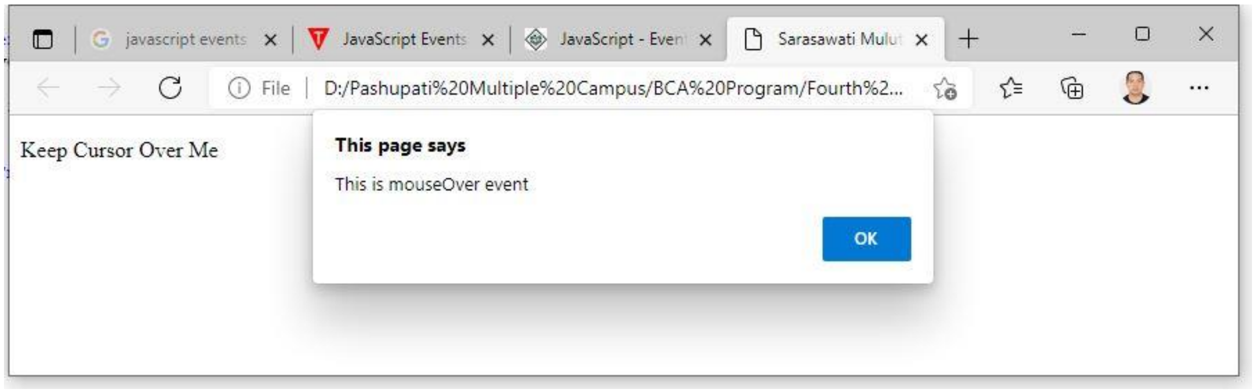
Example-2: mouseOver Event

Example program to demonstrate mouseOver event

```
<title>Sarasawati Mulutiple Campus</title>

</head>

<body>
<script type="text/javascript">
function mouseoverevent()
{
        alert("This is mouseOver event")
}
</script>
<p onmouseover="mouseoverevent()">Keep Cursor Over Me</p>
```

```
</body>
```

Out Put:

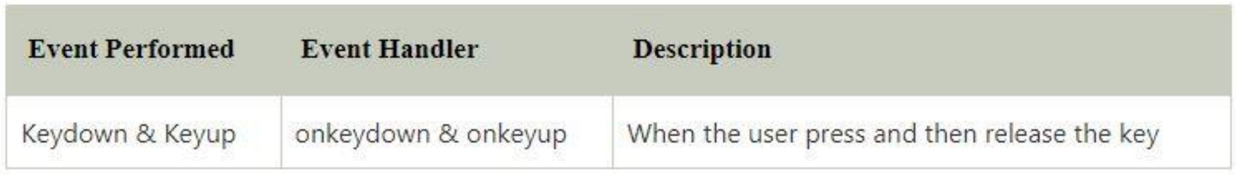


Keep Cursor Over Me

This page says

This is mouseOver event

2. **Keyboard Event**

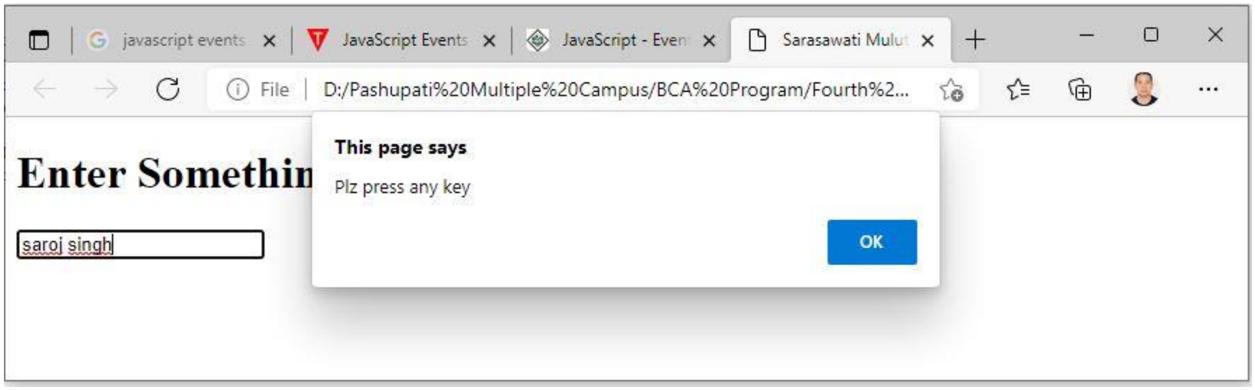


Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

Example:

```
<body>
<h1>Enter Something Here</h1>
<input type="text" id="txt" onkeydown="keydownevent()" />
<script type="text/javascript">
function keydownevent()
{
        document.getElementById("txt")
        alert("Plz press any key")
}
</script>

</body>
```

Out Put:


Enter Somethin

saroj singh

This page says

Plz press any key

3. Form Event

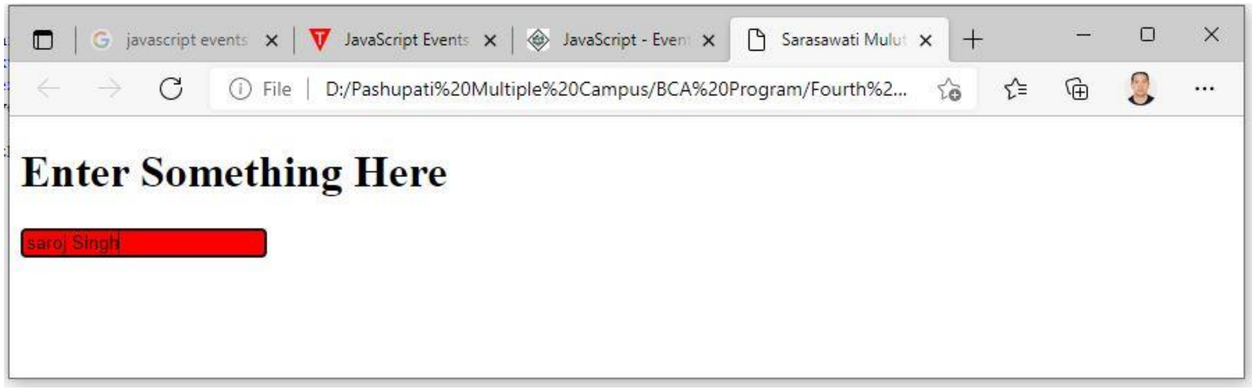
Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

Example:

```
<body>
<h1>Enter Something Here</h1>
<input type="text" id="txt" onfocus="focusevent()" />
<script type="text/javascript">
function focusevent()
{
        document.getElementById("txt").style.background="red"

}
```

```
</script>

</body>
```

Out Put:



Enter Something Here

saroj Singh

4. Windows / Document Events

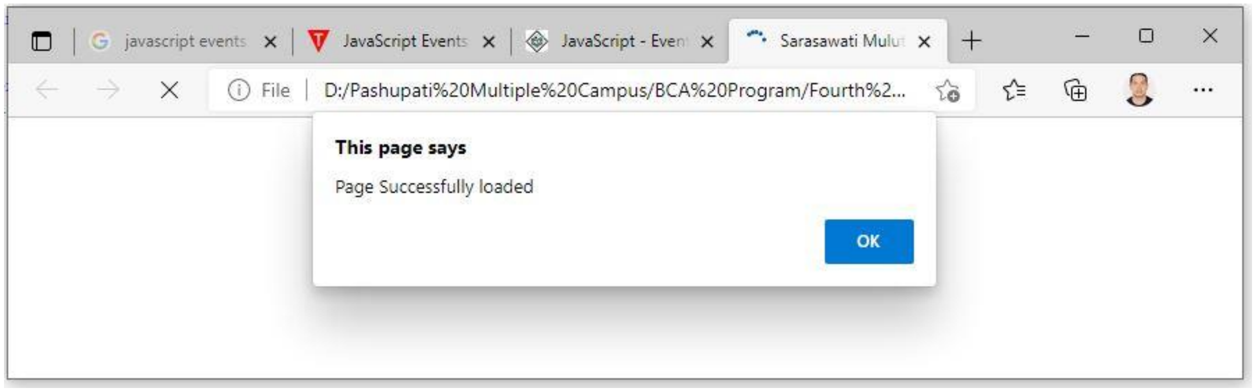
Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

Exampe:

```
<body onload="window.alert('Page Successfully loaded')">



<script type="text/javascript">
document.writeln("Page is successfully loaded")
</script>
```

</body>

Out Put:



This page says

Page Successfully loaded

JavaScript Objects

- A JavaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.
- JavaScript is an object-based language i.e. Object Oriented Languages. Everything is an object in JavaScript.
- JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

How to create object in JavaScript:

- Object can be created in JavaScript in three ways:
  1. By object literal
  2. By creating instance of object directly (using new keyword)
  3. By using an object constructor (using new keyword)

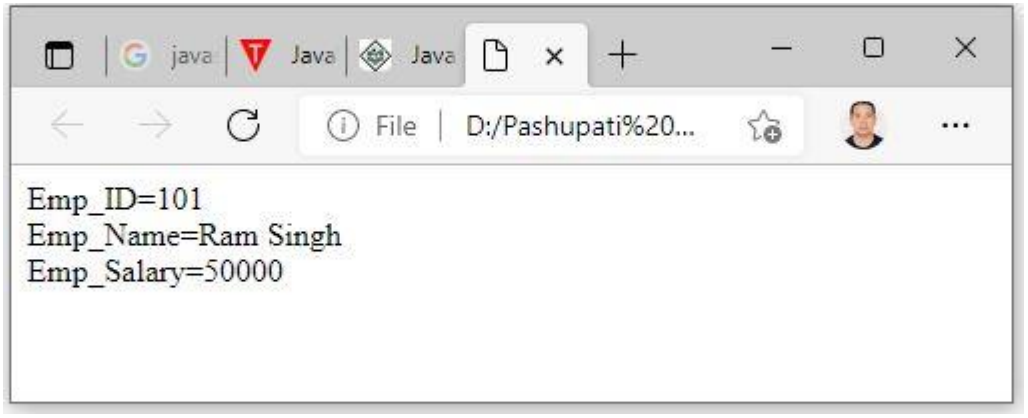
JavaScript Object by using object literal

Syntax:
Object_Name={property1:value1,property2:value2.....propertyN:valueN}

Example:
<body>

```
<script type="text/javascript">
emp={id:101,name:"Ram Singh", salary:50000}
document.writeln("Emp_ID="+emp.id+"<br>"+"Emp_Name="+emp.name+"<
br>"+"Emp_Salary="+emp.salary)
</script>

</body>
```

Out Put:



```
Emp_ID=101
Emp_Name=Ram Singh
Emp_Salary=50000
```

By creating instance of object directly (using new keyword)

- ☞ Object can be created directly using new keyword

Syntax:

var object_name=new Object();

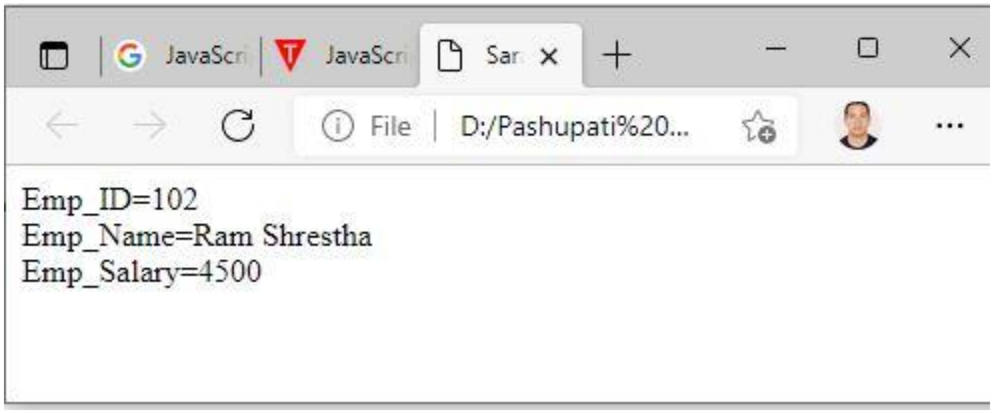
Example:

```
<body>

<script type="text/javascript">

var emp= new Object();

emp.id=102

emp.name="Ram Shrestha"

emp.salary=4500

document.writeln("Emp_ID="+emp.id+"<br>"+"Emp_Name="+emp.name+"<br>"+"
Emp_Salary="+emp.salary)
```

</script>

</body>

Out Put:



```
Emp_ID=102
Emp_Name=Ram Shrestha
Emp_Salary=4500
```

By using an object constructor (using new keyword)

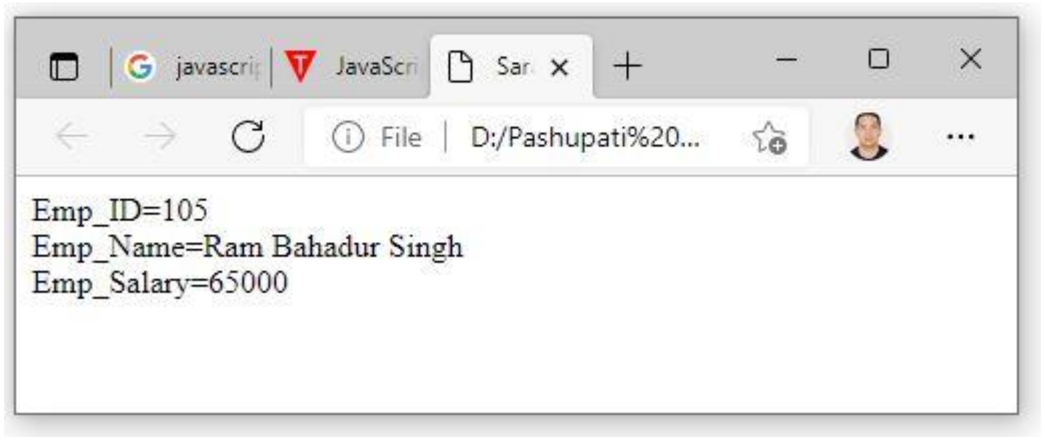
- ☞ By using an object constructor we need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.
- ☞ The **this keyword** refers to the current object.

Example:

```
</head>

<body>

<script type="text/javascript">

function employeedetail(id,name,salary)

{

        this.id=id

        this.name=name

        this.salary=salary

}

emp=new employeedetail(105,"Ram Bahadur Singh",65000)
```

```
document.writeln("Emp_ID="+emp.id+"<br>"+"Emp_Name="+emp.name+"<br>"+"
Emp_Salary="+emp.salary)
</script>


</body>
```

Out Put:



```
Emp_ID=105
Emp_Name=Ram Bahadur Singh
Emp_Salary=65000
```

JavaScript Date Object

- ☞ The JavaScript date object can be used to get year, month and day. You can display a timer on the webpage by the help of JavaScript date object.
- ☞ You can use different Date constructors to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.
- ☞ There are four variant of **Date constructor** to create date object.

1. Date()
2. Date(milliseconds)
3. Date(date String)
4. Date(year, month, day, hours, minute, seconds, milliseconds)

JavaScript Date Method

Some JavaScript Date Method and their descriptions are as follows:

Method	Descriptions
getDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of local time.
getDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of local time.
getFullYears()	It returns the integer value that represents the year on the basis of local time.
getHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of local time.
getMilliseconds()	It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time.
getMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of local time.
getMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of local time.
getSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of local time.
getUTCDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of universal time.
getUTCDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of universal time.
getUTCFullYears()	It returns the integer value that represents the year on the basis of universal time.
getUTCHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of universal time.
getUTCMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of universal time.
getUTCMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of universal time.
getUTCSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of universal time.
setDate()	It sets the day value for the specified date on the basis of local time.
setDay()	It sets the particular day of the week on the basis of local time.
setFullYears()	It sets the year value for the specified date on the basis of local time.
setHours()	It sets the hour value for the specified date on the basis of local time.

setMilliseconds()	It sets the millisecond value for the specified date on the basis of local time.
setMinutes()	It sets the minute value for the specified date on the basis of local time.
setMonth()	It sets the month value for the specified date on the basis of local time.
setSeconds()	It sets the second value for the specified date on the basis of local time.
setUTCDate()	It sets the day value for the specified date on the basis of universal time.
setUTCDay()	It sets the particular day of the week on the basis of universal time.
setUTCFullYears()	It sets the year value for the specified date on the basis of universal time.
setUTCHours()	It sets the hour value for the specified date on the basis of universal time.
setUTCMilliseconds()	It sets the millisecond value for the specified date on the basis of universal time.
setUTCMinutes()	It sets the minute value for the specified date on the basis of universal time.
setUTCMonth()	It sets the month value for the specified date on the basis of universal time.
setUTCSeconds()	It sets the second value for the specified date on the basis of universal time.
toDateString()	It returns the date portion of a Date object.
toISOString()	It returns the date in the form ISO format string.
toJSON()	It returns a string representing the Date object. It also serializes the Date object during JSON serialization.
toString()	It returns the date in the form of string.
toTimeString()	It returns the time portion of a Date object.
toUTCString()	It converts the specified date in the form of string using UTC time zone.
valueOf()	It returns the primitive value of a Date object.

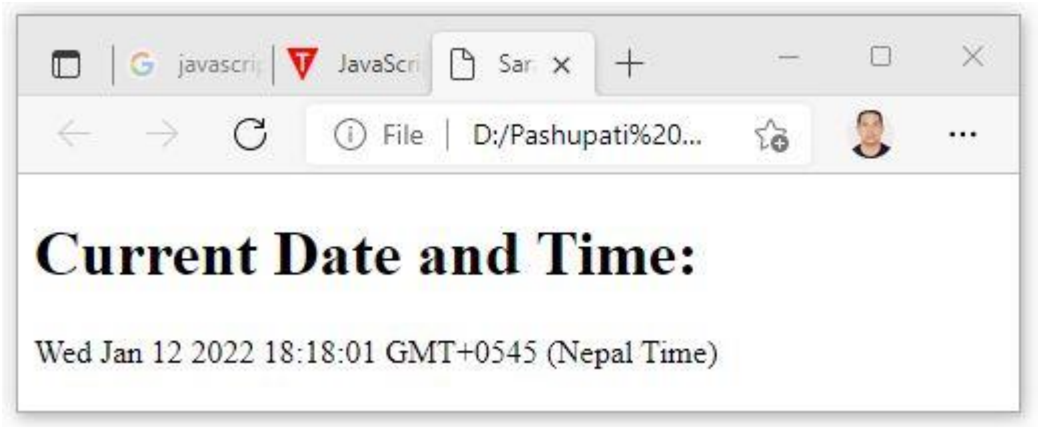
Example:

Current Date and Time:

```
<body>

<h1>Current Date and Time:</h1><span id="time"></span>
```

```
<script type="text/javascript">
var today=new Date()
document.getElementById('time').innerHTML=today
</script>

</body>
```

Out Put:



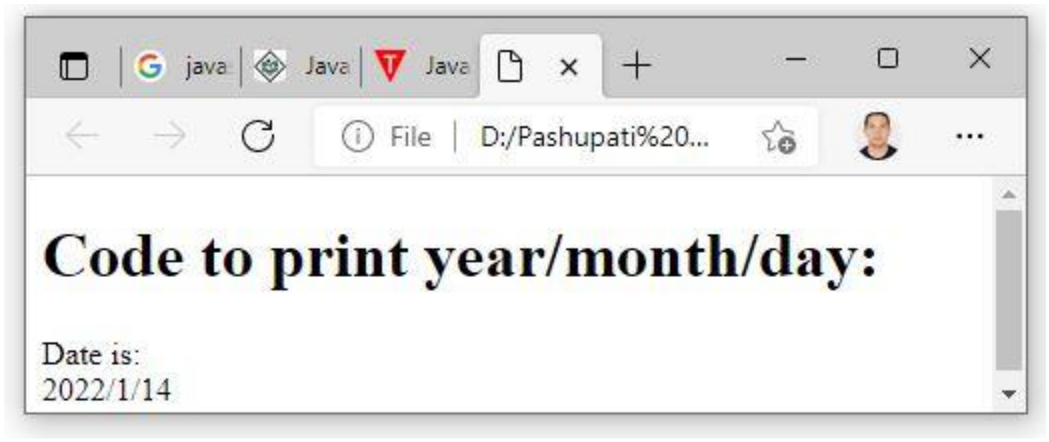
Current Date and Time:

Wed Jan 12 2022 18:18:01 GMT+0545 (Nepal Time)

Example-2: Code to print **Year/month/day**

```
<body>
<h1>Code to print year/month/day:</h1>
<script type="text/javascript">
var date=new Date()
var day=date.getDate()
var month=date.getMonth()+1
var year=date.getFullYear()
document.writeln("Date is:<br>"+year+"/"+month+"/"+day)
</script>
</body>
```

Out Put:



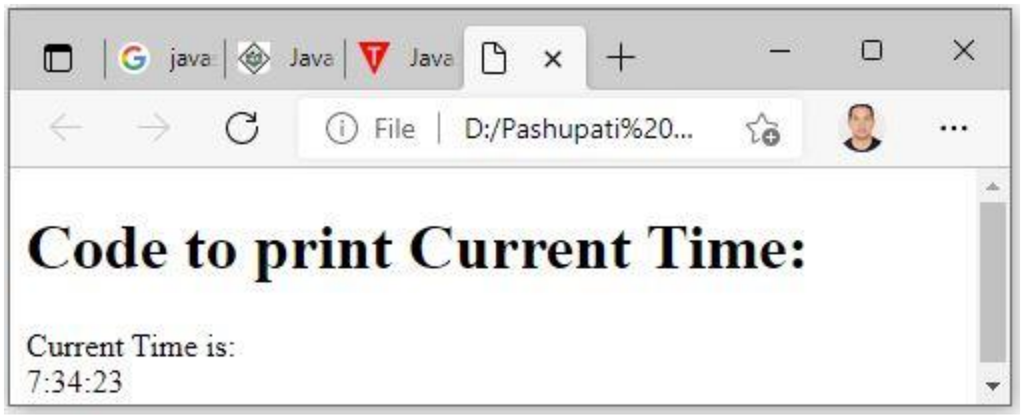
Code to print year/month/day:

Date is:
2022/1/14

Example-3: Current Time Display

```
<body>

<h1>Code to print Current Time:</h1><span id="time"></span>

<script type="text/javascript">

var today=new Date()

var h=today.getHours()

var m=today.getMinutes()

var s=today.getSeconds()

document.getElementById('time').innerHTML="Current Time
is:<br>"+h+":"+m+":"+ s

</script>

</body>
```

Out Put:



```
Code to print Current Time:

Current Time is:
7:34:23
```

How to make Digital Clock Using JavaScript

- ☞ Digital clock can be created by using **date object**.
- ☞ There are two ways to set interval in JavaScript:
  - ❖ By setTimeout()
  - ❖ By setInterval()

```
<body>
<h1>Code to print Current Time:</h1><span id="time"></span>
<script type="text/javascript">
window.onload=function(){getTime();}
function getTime(){
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
// add a zero in front of numbers<10
m=checkTime(m);
s=checkTime(s);
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
setTimeout(function(){getTime()},1000);
}
//setInterval("getTime()",1000);//another way
function checkTime(i){
if (i<10){
  i="0" + i;
 }
return i;
}
</script>
</body>
```

Out Put:

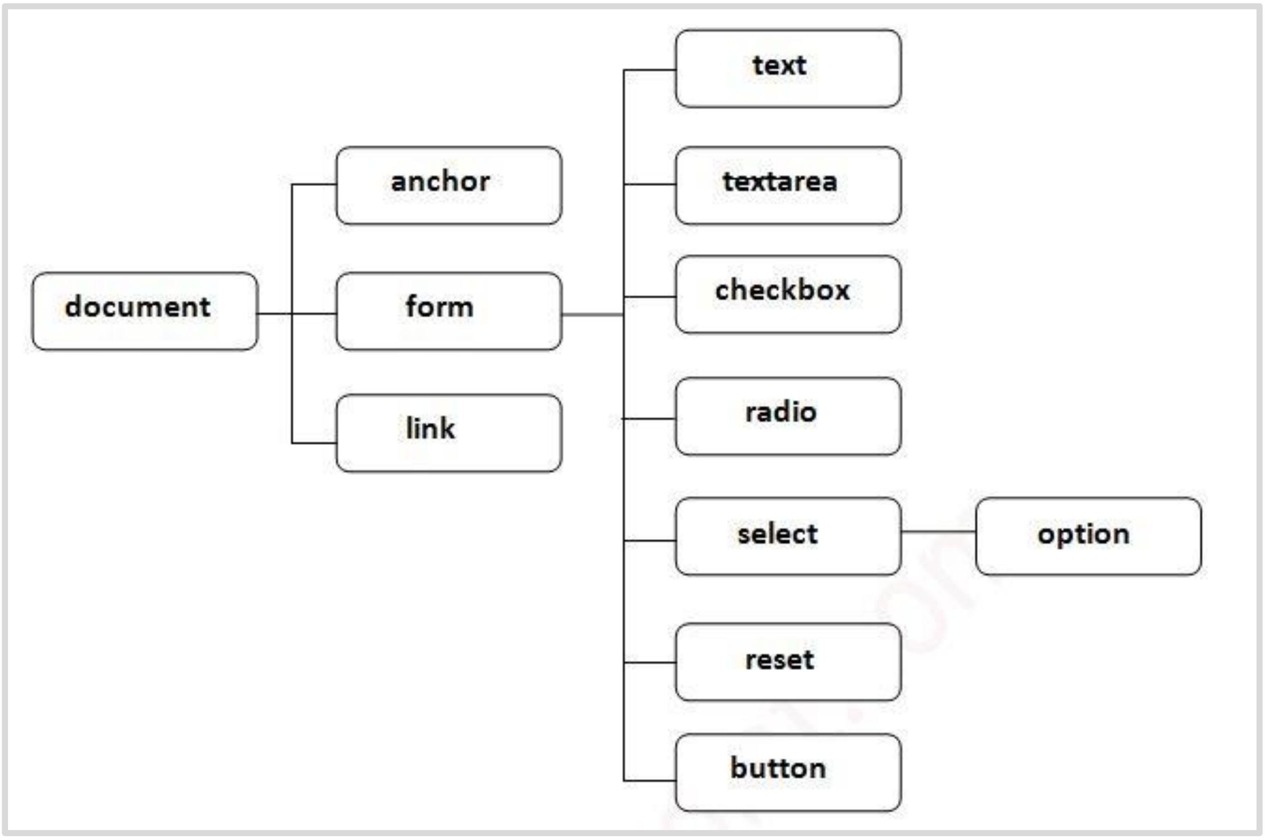
JavaScript Document Object Model (DOM)

Document Object Model

- The document object represents the whole html document.
- When html document is loaded in the browser, it becomes a document object. It is the root element that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.
- It is the object of window. So **"window.document"** is same as **"document"**.

Properties of document object

- The properties of document object that can be accessed and modified by the document object.



Methods of document object

- We can access and change the contents of document by its methods.

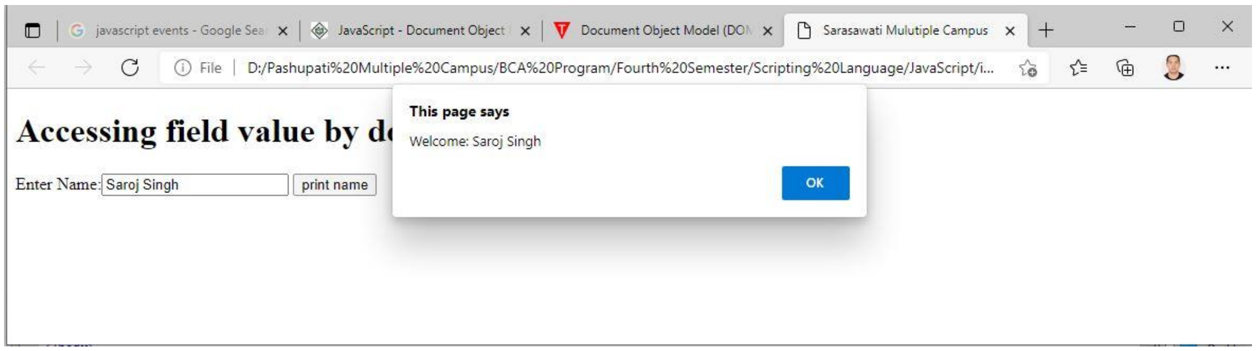
☞ The important methods of document object are as follows:

Method	Description
write("string")	It is used to write the given string on the document.
writeln("string")	It is used to write the given string on the document with newline character at the end.
getElementById()	It is used to return the element having the given id value.
getElementsByName()	It is used to return all the elements having the given name value.
getElementsByTagName()	It is used to return all the elements having the given tag name.
getElementsByClassName()	It is used to return all the elements having the given class name.

Accessing field value by document object

```
<body>
<h1>Accessing field value by document object</h1>
<script type="text/javascript">
function printvalue(){
var name=document.form1.name.value;
alert("Welcome: "+name);
}
</script>
 <form name="form1">
Enter Name:<input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
</script>
</body>
```

Out Put:


javascript events - Google Sea
JavaScript - Document Object
Document Object Model (DOM
Sarasawati Mulutiple Campus
File | D:/Pashupati%20Multiple%20Campus/BCA%20Program/Fourth%20Semester/Scripting%20Language/JavaScript/i...
OK

Accessing field value by d

Enter Name: Saroj Singh [print name]

This page says

Welcome: Saroj Singh

Explanation:

In this example, we are going to get the value of input text by user. Here, we are using document.form1.name.value to get the value of name field. Here, document is the root element that represents the html document. **form1** is the name of the form.**name** is the attribute name of the input text. value is the property, that returns the value of the input text.

document.getElementById() method

☞ The **document.getElementById() method** returns the element of specified id.

Exampe:

```
<body>
<h1>Accessing field value by document object</h1>
<script type="text/javascript">
function getcube(){
var number=document.getElementById("number").value;
alert(number*number*number);
}
</script>
<form>
Enter No:<input type="text" id="number" name="number"/><br/>
```

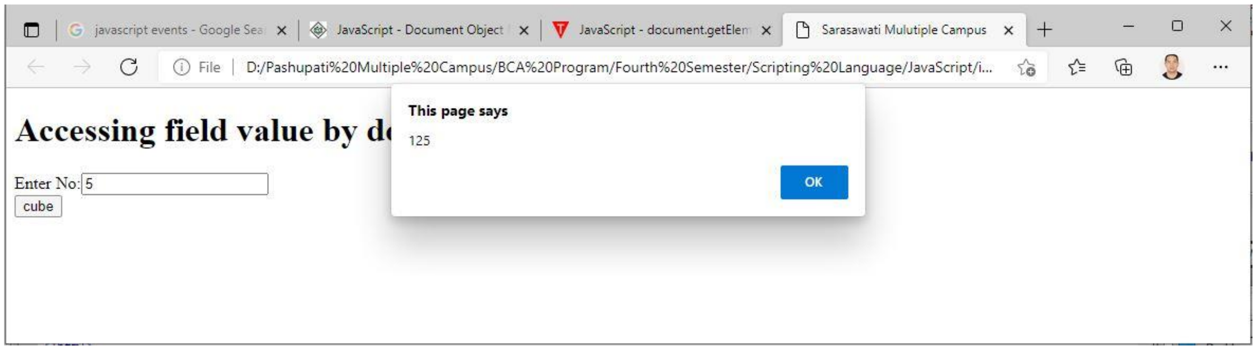
<input type="button" value="cube" onclick="getcube()"/>

</form>

</script>

</body>

Out Put:


javascript events - Google Sea
JavaScript - Document Object
JavaScript - document.getElem
Sarasawati Mulutiple Campus
File
D:/Pashupati%20Multiple%20Campus/BCA%20Program/Fourth%20Semester/Scripting%20Language/JavaScript/i...
OK

Accessing field value by d

Enter No: 5

cube

This page says

125

GetElementsByClassName()

- The getElementsByClassName() method is used for selecting or getting the elements through their class name value.
- This DOM method returns an array-like object that consists of all the elements having the specified classname.
- On calling the getElementsByClassName() method on any particular element, it will search the whole document and will return only those elements which match the specified or given class name.

Syntax:

Var variabl_name=document.getELementsByClassName('name');

Example:

<body>

<div class="Class">

This is a simple class implementation

</div>

<script type="text/javascript">

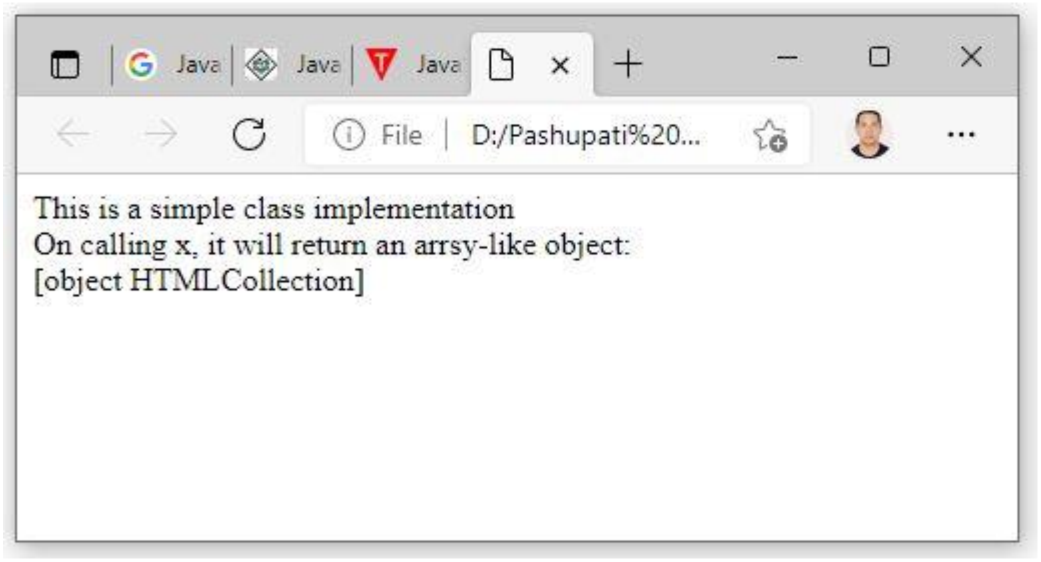
var x=document.getElementsByClassName('Class');

document.write("On calling x, it will return an arrsy-like object: <br>"+x);

</script>

</body>

Out Put:



This is a simple class implementation
On calling x, it will return an arrsy-like object:
[object HTMLCollection]

document.getElementsByName() method

☞ The document.getElementsByName() method returns all the element of specified name.

Syntax:

document.getElementsByName("name")

Example:

<body>

<script type="text/javascript">

function totalelements()

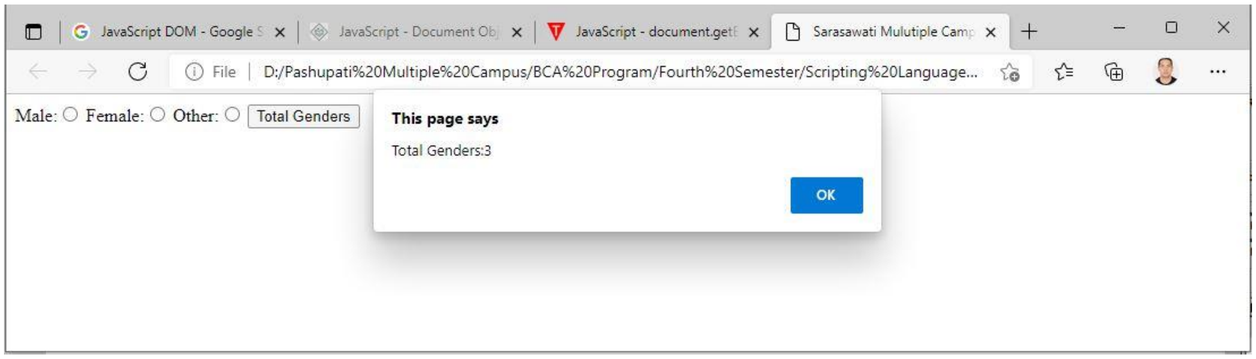
{

var allgenders=document.getElementsByName("gender");

alert("Total Genders:"+allgenders.length);

```
}
</script>
<form>
Male:<input type="radio" name="gender" value="male">
Female:<input type="radio" name="gender" value="female">
Other:<input type="radio" name="gender" value="Other">

<input type="button" onclick="totalelements()" value="Total Genders">
</form>
</body>
```

Out Put:



This page says
Total Genders:3

document.getElementsByTagName()

☞ The document.getElementsByTagName() method returns all the element of specified tag name.

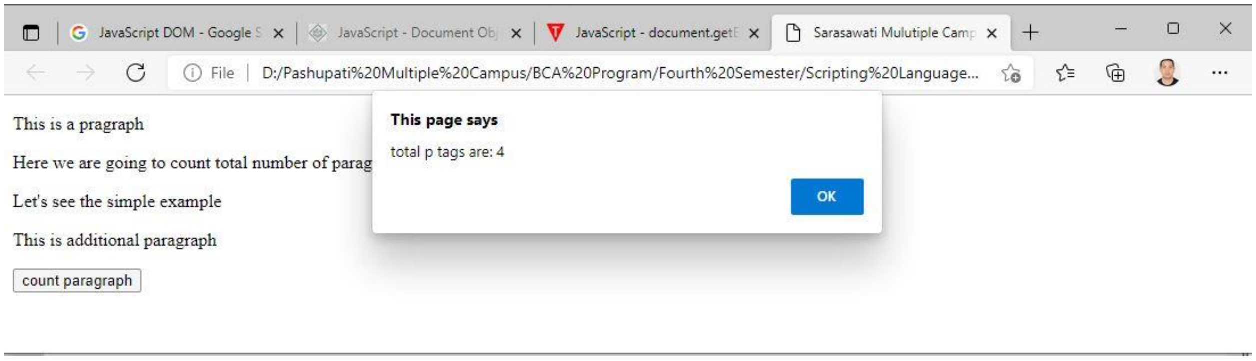
Syntax:

document.getElementsByTagName("name")

Exampe:

```
<body>
<script type="text/javascript">
function countpara(){
var totalpara=document.getElementsByTagName("p");
alert("total p tags are: "+totalpara.length);
}
</script>
<p>This is a pragraph</p>
<p>Here we are going to count total number of paragraphs by getElementByTagName() method.</p>
<p>Let's see the simple example</p>
<p>This is additional paragraph</p>
<button onclick="countpara()">count paragraph</button>
</body>
```

Out Put:


JavaScript DOM - Google S
JavaScript - Document Ob
JavaScript - document.getE
Sarasawati Mulutiple Camp
File
D:/Pashupati%20Multiple%20Campus/BCA%20Program/Fourth%20Semester/Scripting%20Language...
OK
count paragraph

This is a pragraph

Here we are going to count total number of parag

Let's see the simple example

This is additional paragraph

This page says

total p tags are: 4

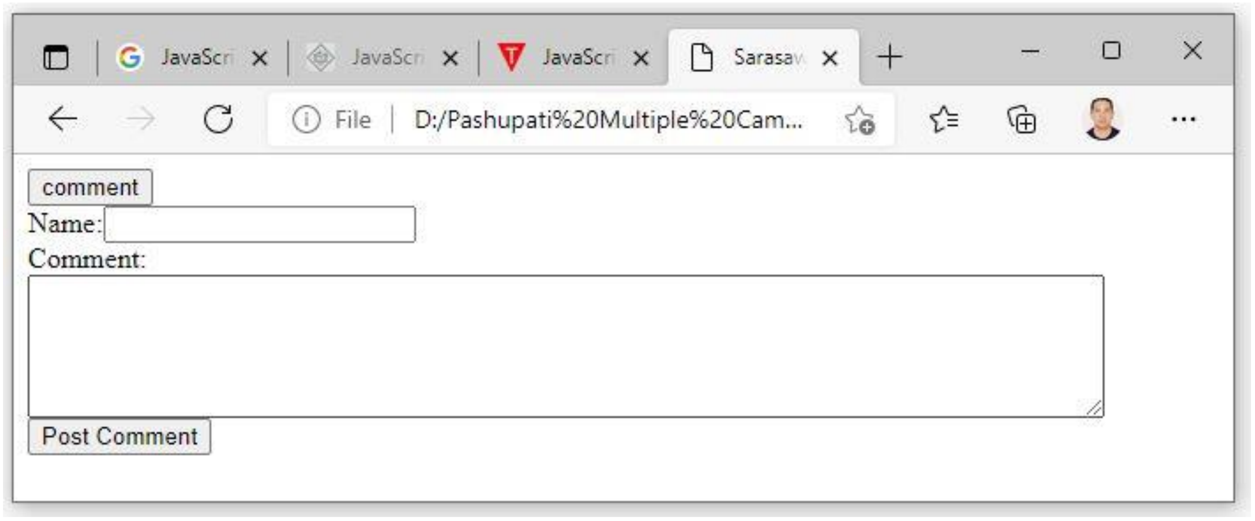
innerHTML Property

- The innerHTML property can be used to write the dynamic html on the html document.
- It is used mostly in the web pages to generate the dynamic html such as registration form, comment form, links etc.

Example:

```
<body>
<script type="text/javascript" >
function showcommentform() {
var data="Name:<input type='text' name='name'><br>Comment:<br><textarea rows='5' cols='80'></textarea><br><input type='submit' value='Post Comment'>";
document.getElementById('mylocation').innerHTML=data;
}
</script>
<form name="myForm">
<input type="button" value="comment" onclick="showcommentform()">
<div id="mylocation"></div>
</form>
</body>
```

Out Put:



innerText properties

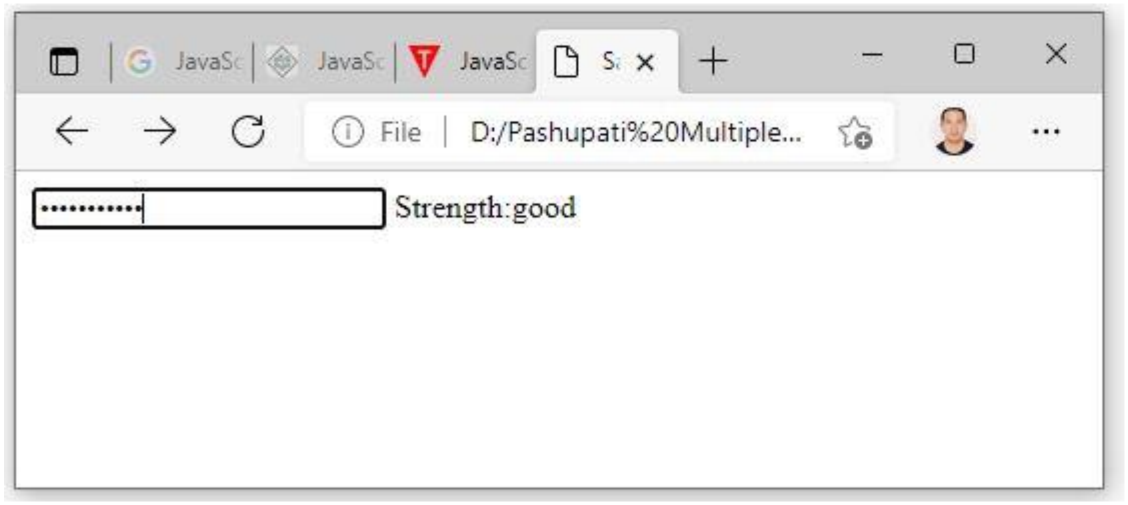
- ☞ The innerText property can be used to write the dynamic text on the html document. Here, text will not be interpreted as html text but a normal text.
- ☞ It is used mostly in the web pages to generate the dynamic content such as writing the validation message, password strength etc.

Example:

```
<title>Sarasawati Mulutiple Campus</title>
<script type="text/javascript" >
function validate() {
var msg;
if(document.myForm.userPass.value.length>5){
msg="good";
}
else{
msg="poor";
}
document.getElementById('mylocation').innerText=msg;
 }

</script>
</head>

<body>
<form name="myForm">
<input type="password" value="" name="userPass" onkeyup="validate()">
Strength:<span id="mylocation">no strength</span>
```

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

Out Put:



Strength:good

JavaScript Validation

☞ JavaScript provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally performs two functions.

- ❖ **Basic Validation –** First of all, the form must be checked to make sure all the mandatory fields are filled in. It would require just a loop through each field in the form and check for data.

- ❖ **Data Format Validation –** Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.

Form Validation

☞ It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user.

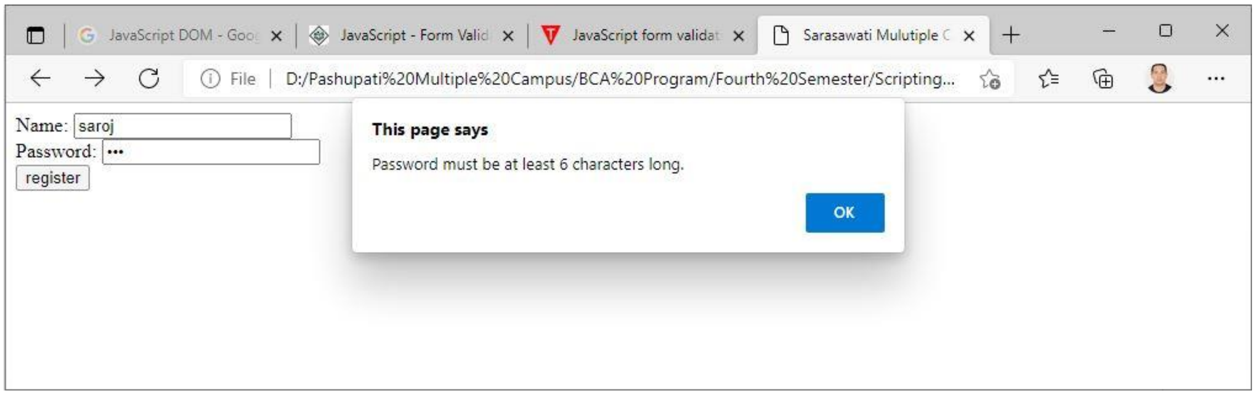
☞ JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation. Most of the web developers prefer JavaScript form validation.

☞ Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.

Exampe:

```
<title>Sarasawati Mulutiple Campus</title>
<script type="text/javascript" >
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
  alert("Name can't be blank");
  return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters long.");
  return false;
  }
}

</script>
</head>

<body>
<form name="myform" method="post" action="abc.jsp" onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
</body>
```

Out Put:



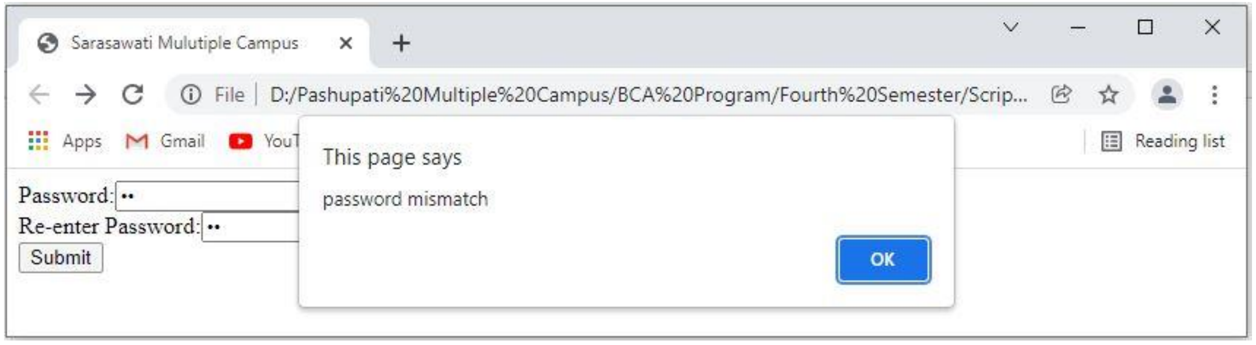
This page says

Password must be at least 6 characters long.

Retype Password Validation

```
<title>Sarasawati Mulutiple Campus</title>
<script type="text/javascript" >
function matchpass(){
var firstpassword=document.f1.password.value;
var secondpassword=document.f1.password2.value;

if(firstpassword==secondpassword){
return true;
}
else{
alert("password mismatch");
return false;
}
}

</script>
</head>

<body>
<form name="f1" action="register.jsp" onsubmit="return matchpass()">
Password:<input type="password" name="password" /><br/>
Re-enter Password:<input type="password" name="password2"/><br/>
<input type="submit">
</form>
</body>
```

Out Put:



This page says

password mismatch

Sarasawati Mulutiple Campus
D:/Pashupati%20Multiple%20Campus/BCA%20Program/Fourth%20Semester/Scrip...
Password:
Re-enter Password:
Submit
OK

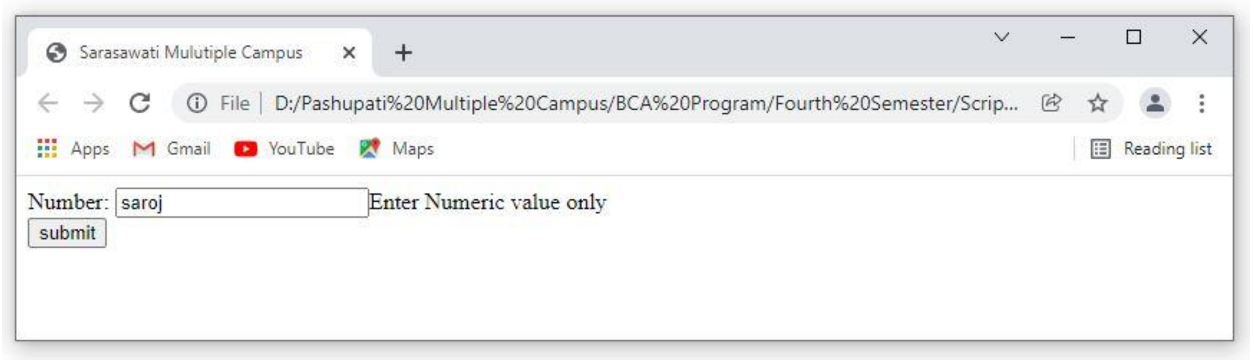
Number Validation

☞ Text field allow number value only. By using NaN() function.

```
<title>Sarasawati Mulutiple Campus</title>
<script type="text/javascript" >
function validate(){
```

```
var num=document.myform.num.value;
if (isNaN(num)){
  document.getElementById("numloc").innerHTML="Enter Numeric value only";
  return false;
}else{
  return true;
  }
}
</script>
</head>

<body>
<form name="myform" onsubmit="return validate()" >
Number: <input type="text" name="num"><span id="numloc"></span><br/>
<input type="submit" value="submit">
</form>
</body>
```

Out Put:


Sarasawati Mulutiple Campus
File | D:/Pashupati%20Multiple%20Campus/BCA%20Program/Fourth%20Semester/Scrip...
Apps
Gmail
YouTube
Maps
Reading list

```
Number: saroj Enter Numeric value only
submit
```

E-mail validation

☞ We can validate the email by the help of JavaScript.

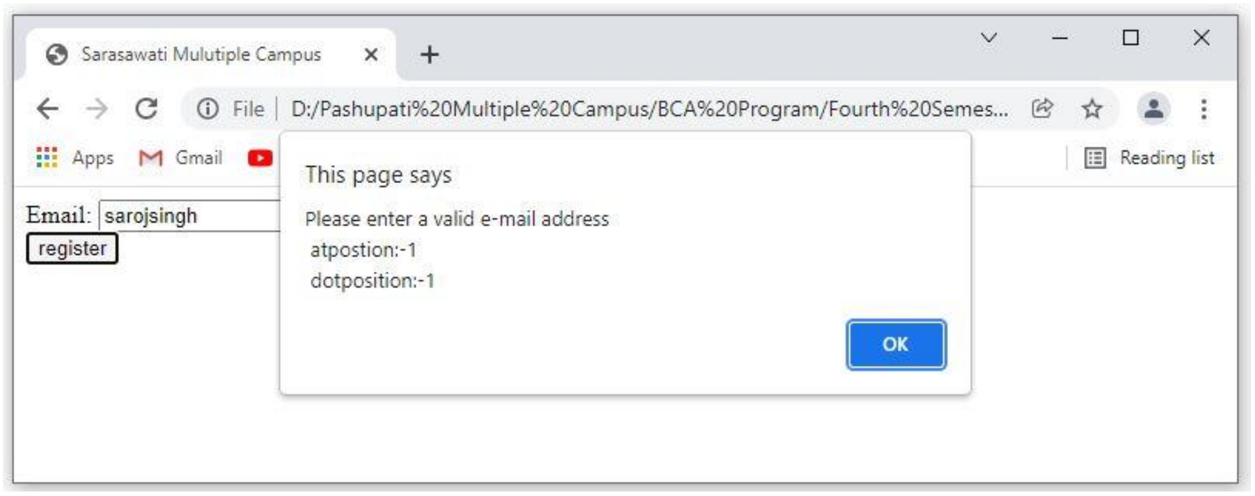
☞ There are many criteria that need to be follow to validate the email id such as:

- ❖ Email id must contain the @ and . character
- ❖ There must be at least one character before and after the @.
- ❖ There must be at least two characters after . (dot).

Example:

```
<title>Sarasawati Mulutiple Campus</title>
<script type="text/javascript" >
function validateemail()
{
var x=document.myform.email.value;
var atposition=x.indexOf("@");
var dotposition=x.lastIndexOf(".");
if (atposition<1 || dotposition<atposition+2 || dotposition+2>=x.length){
 alert("Please enter a valid e-mail address \n atpostion:"+atposition+"\n dotposition:"+dotposition);
 return false;
 }
}
</script>
</head>
<body>
<form name="myform" method="post" action="#" onsubmit="return validateemail();">
Email: <input type="text" name="email"><br/>
 <input type="submit" value="register">
</form>
</body>
```

Out Put:


Sarasawati Mulutiple Campus
File | D:/Pashupati%20Multiple%20Campus/BCA%20Program/Fourth%20Semes...
Apps
Gmail
Reading list
Email: sarojsingh
register
OK

This page says

Please enter a valid e-mail address
atpostion:-1
dotposition:-1

JavaScript Cookies

- ☞ Cookies were originally invented by Netscape to give 'memory' to web servers and browser.
- ☞ A cookie is an amount of information that persists between a server-side and a client side.
- ☞ A web browser stores this information at the time of browsing.
- ☞ A cookie contains the information as a string generally in the form of a name-value pair separated by semi-colons.
- ☞ It maintains the state of a user and remembers the user's information among all the web pages.

How Cookies Works?

- ☞ When a user sends a request to the server, then each of that request is treated as a new request sent by the different user.
- ☞ So, to recognize the old user, we need to add the cookie with the response from the server.
- ☞ Browser at the client-side.
- ☞ Now, whenever a user sends a request to the server, the cookie is added with that request automatically. Due to the cookie, the server recognizes the users.
- ☞ A cookies is nothing but a small text file that's stored in your browser. It contains some 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

Name-value pair

- ☞ Each cookie has a name-value pair that contains the actual information. The name of the cookie is for your benefit, you will search for this name when reading out the cookie information.

Expiry date

- ☞ Each cookie has an expiry date after which it is trashed. If you don't specify the expiry date the cookie is trashed when you c lose the browser. This expiry date should be in UTC (Greenwich) time.

Domain and path

- ☞ Each cookie also has a domain and a path. The domain tells the browser to which domain the cookie should be sent. If you don't specify it, it becomes the domain of the page that sets the cookie.
- ☞ The path gives you the chance to specify a directory where the cookie is active. So, if you want the cookie to be only sent to pages in the directory user, set the path to /user. Usually the path is set to /, which means the cookie is valid throughout the entire domain.

document.cookie

- ☞ Cookie can be created, read and erased by JavaScript. They are accessible through the property document.cookie. Through you can treat document.cookie as if it's a string, it isn't really, and you have only access to the name-value pairs.

General syntax:

Document.cookie='username=abcd; expires=sun,18 Mar 2022 09:03:58 UTC; path=/'

OR

Note:

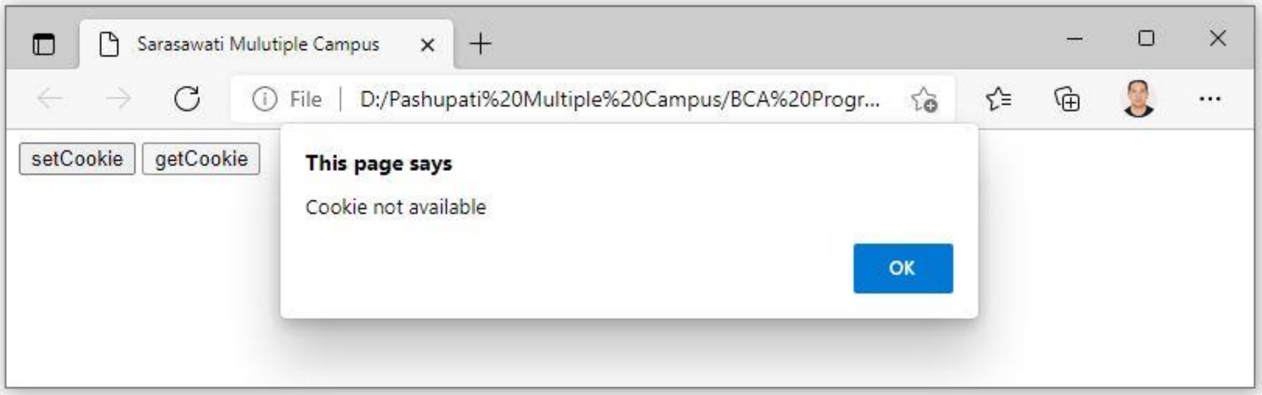
- ☞ In JavaScript, we can create, read, update and delete a cookie by using **document.cookie** property.
  Syntax:

  ```
  document.cookie="name=value";
  ```

Eample-1 : (to set and get cookie)

```
<body>
<input type="button" value="setCookie" onclick="setCookie()">
<input type="button" value="getCookie" onclick="getCookie()">
<script type="text/javascript" >
function setCookie()
  {
     document.cookie="username=saroj singh";
  }
  function getCookie()
  {
     if(document.cookie.length!=0)
     {
     alert(document.cookie);
     }
     else
     {
     alert("Cookie not available");
     }
  }

</script>
</body>
```

Out Put:


Sarasawati Mulutiple Campus
File | D:/Pashupati%20Multiple%20Campus/BCA%20Progr...
setCookie
getCookie
OK

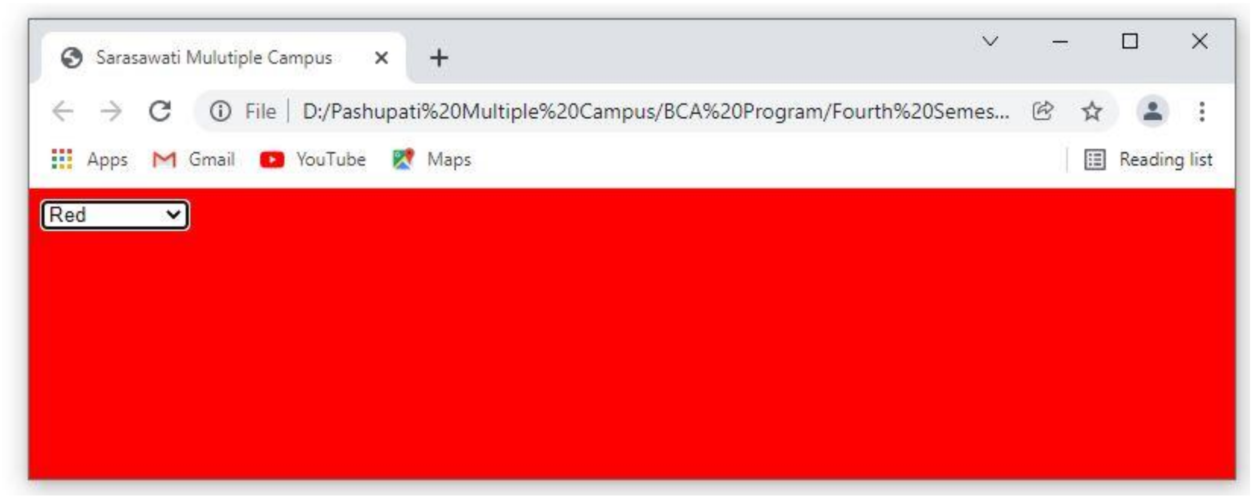
This page says

Cookie not available

Example-2:

```
<body>
 <select id="color" onchange="display()">
        <option value="Select Color">Select Color</option>
        <option value="yellow">Yellow</option>
        <option value="green">Green</option>
        <option value="red">Red</option>
      </select>
<script type="text/javascript" >
 function display()
        {
          var value = document.getElementById("color").value;
          if (value != "Select Color")
          {
            document.bgColor = value;
            document.cookie = "color=" + value;
          }
        }
        window.onload = function ()
        {
            if (document.cookie.length != 0)
            {
                var array = document.cookie.split("=");
                document.getElementById("color").value = array[1];
                document.bgColor = array[1];
            }
        }
</script>
</body>
```

Out Put:


Sarasawati Mulutiple Campus
File | D:/Pashupati%20Multiple%20Campus/BCA%20Program/Fourth%20Semes...
Apps
Gmail
YouTube
Maps
Reading list
Red

Cookie Attributes

☞ JavaScript provides some optional attributes that enhance the functionality of cookies. Here, is the list of some attributes with their description.

Attributes	Descriptions
expires	It maintains the state of a cookie up to the specified date and time.

max-age	It maintains the state of a cookie up to the specified time. Here, time is given in seconds.
path	It expands the scope of the cookie to all the pages of a website.
domain	It is used to specify the domain for which the cookie is valid.

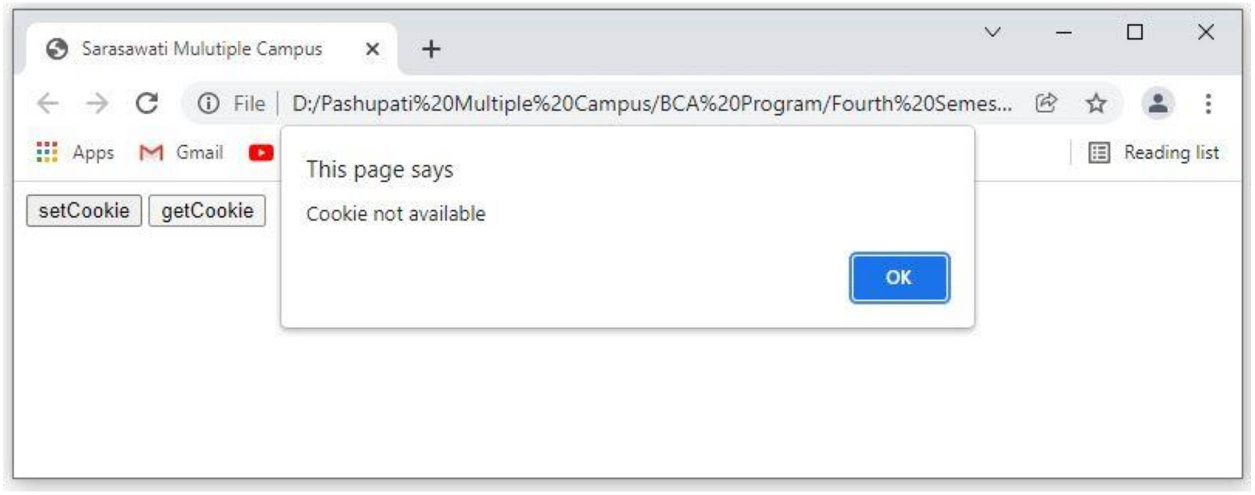
Cookie expires attribute

☞ The cookie expires attribute provides one of the ways to create a persistent cookie. Here, a date and time are declared that represents the active period of a cookie. Once the declared time is passed, a cookie is deleted automatically.

```
<body>
 <input type="button" value="setCookie" onclick="setCookie()">
<input type="button" value="getCookie" onclick="getCookie()">
<script type="text/javascript" >
 function setCookie()
   {
     document.cookie="username=saroj singh;expires=Sun, 20 Aug 2023 12:00:00
UTC";
   }
   function getCookie()
   {
     if(document.cookie.length!=0)
     {
       var array=document.cookie.split("=");
     alert("Name="+array[0]+" "+"Value="+array[1]);
     }
     else
     {
     alert("Cookie not available");
     }
```

```
  }

</script>
</body>
```

Out Put:



This page says

Cookie not available

Sarasawati Mulutiple Campus
D:/Pashupati%20Multiple%20Campus/BCA%20Program/Fourth%20Semes...
Apps
Gmail
Reading list
setCookie
getCookie
OK

Cookie max-age attribute

☞ The cookie max-age attribute provides another way to create a persistent cookie. Here, time is declared in **seconds.** A cookie is valid up to the declared time only.

Example:

```
<body>
 <input type="button" value="setCookie" onclick="setCookie()">
<input type="button" value="getCookie" onclick="getCookie()">
<script type="text/javascript" >
 function setCookie()
   {
      document.cookie="username=Ram Shrestha;max-age=" + (60 * 60 * 25 * 365)
+ ";"
```

```
        }
        function getCookie()
        {
           if(document.cookie.length!=0)
           {
              var array=document.cookie.split("=");
           alert("Name="+array[0]+" "+"Value="+array[1]);
           }
           else
           {
           alert("Cookie not available");
           }
        }
     </script>
     </body>
```

Cookie path attribute

☞ If a cookie is created for a webpage, by default, it is valid only for the current directory and sub-directory. JavaScript provides a path attribute to expand the scope of cookie up to all the pages of a website.

Exampe:

```
<body>
 <input type="button" value="setCookie" onclick="setCookie()">
<input type="button" value="getCookie" onclick="getCookie()">
<script type="text/javascript" >
 function setCookie()
   {
        document.cookie="username=Ram Shrestha;max-age=" + (60 * 60 * 25 * 365) +
";path=/;"
    }
    function getCookie()
    {
        if(document.cookie.length!=0)
        {
            var array=document.cookie.split("=");
        alert("Name="+array[0]+" "+"Value="+array[1]);
        }
        else
        {
        alert("Cookie not available");
        }
    }
</script>
</body>
```

Deleting a Cookie in JavaScript

- ☞ These are the following ways to delete a cookie:
- ❖ A cookie can be deleted by using expire attribute.
- ❖ A cookie can also be deleted by using max-age attribute.
- ❖ We can delete a cookie explicitly, by using a web browser.

Exampe:

```
<body>
 <input type="button" value="setCookie" onclick="setCookie()">
<input type="button" value="getCookie" onclick="getCookie()">
```

```
<script type="text/javascript" >
function setCookie()
{
  document.cookie="name=Ram Shrestha; expires=Sun, 20 Aug 2000 12:00:00 UTC";


}
function getCookie()
{
  if(document.cookie.length!=0)
  {
  alert(document.cookie);
  }
  else
  {
    alert("Cookie not avaliable");
  }
}
</script>
</body>
```

Handling Regular Expression

- ☞ Regular expressions are patterns used to match character combinations in strings. JavaScript, regular expressions are also objects.
- ☞ These patterns are used with the exec() and test() methods of Regular Expression and with the match(), matchAll(), replace(), replaceAll(), search() and split() methods of String.
- ☞ Creating a regular expression you construct a regular expression in one of two ways:

- ❖ Using a regular expression literal, which consists of a pattern enclosed between slashes, as follows:

  ```
  Let re=/ab+c/;
  ```
- ❖ Calling the constructor function of the Regular Expression object as follows:

  ```
  Let re=new RegExp('ab+c');
  ```

Using the constructor function provides runtime compilation of the regular expression. Use the constructor function when you know the regular expression pattern will be changing or you don't know the pattern and are getting it from another source such as user input.

Regular Expression Methods

- ☞ There are mainly two methods for testing regular expressions.
  I. RegExp.prototype.test()
  II. RegExp.prototype.exec()

RegExp.prototype.test()

- ☞ This method is used to test whether a match has been found or not. It accepts a string which we have to test against regular expression and returns true or false depending upon if the match is found or not.
  Eg.

  ```
  var regex=/hello/;
  var str='hello world';
  var result=regex.test(str);
  console. log(result);
  //returns true
  ```

RegExp.prototype.exec()

- ☞ This method returns an array containing all the matched groups. It accepts a string that we have to test against a regular expression.

```
var regex =/hello/;
var str='hello world';
var result=regex.exec(str);
console.log(result);
//returns ['hello', index:0, input:'hello world', groups: undefined]
```

Simple Regex Patterns

It is the most basic pattern , which simply matches the literal text with the test string.

```
var regex=/hello/;
console.
//true.
```

Flags:
Regular expressions have five optional flags or modifiers. Let's discuss the two most important flags:
g- Global search don't return after the first match
i- Case-insensitive search
You can also combine the flags in a single regular expression. Note that their order doesn't have any effect on the result.

Meta-characters

- ☞ Meta-characters are characters with a special meaning. There are many Meta characters but I am going to cover the most important ones here.

  \d – Match any digit character (same as [0-9].
  \w – Match any word character. A word character is any letter, digit, and underscore. (same as [a-z A-Z0-9_]) i.e. alphanumeric character.
  \s – Match a whitespace character (spaces, tabs etc).
  \t – Match a tab character only.
  \b – Find a match at beginning or ending of a word. Also known as word boundary.
  . – (period) matches any character except for new line.
  \D – Match any non-digit character (same as [^0-9]).
  \W – Match any non-word character (same as ^a-zA-Z0-9_]).
  \S – Match a non-whitespaces character.

Quantifiers:

- ☞ Quantifiers are symbols that have a special meaning in a regular expression.

+ - Matches the preceding expression 1 or more times.
* - Matches the preceding expression 0 or more times.
? – Matches the preceding expression 0 or 1 time, which is preceding pattern is optional.
^ - Matches the beginning of the string, the regular expression that follows it should be at the start of the test string. i.e. the caret(^) matches the start of string.
$- Matches the end of string, that is the regular expression that precedes it should be at the end of the test string. The dollar ($) sign matches the end of the string.
{N} – Matches exactly N occurrences of the preceding regular expression.
{N,} – Matches at least N occurrences of the preceding regular expression.
{N, M} – Matches at least N occurrences of at most M occurrences of the preceding regular expression (where M>N).
Alternation X|Y – Matches either X or Y.

Example: Match any 10 digit number:

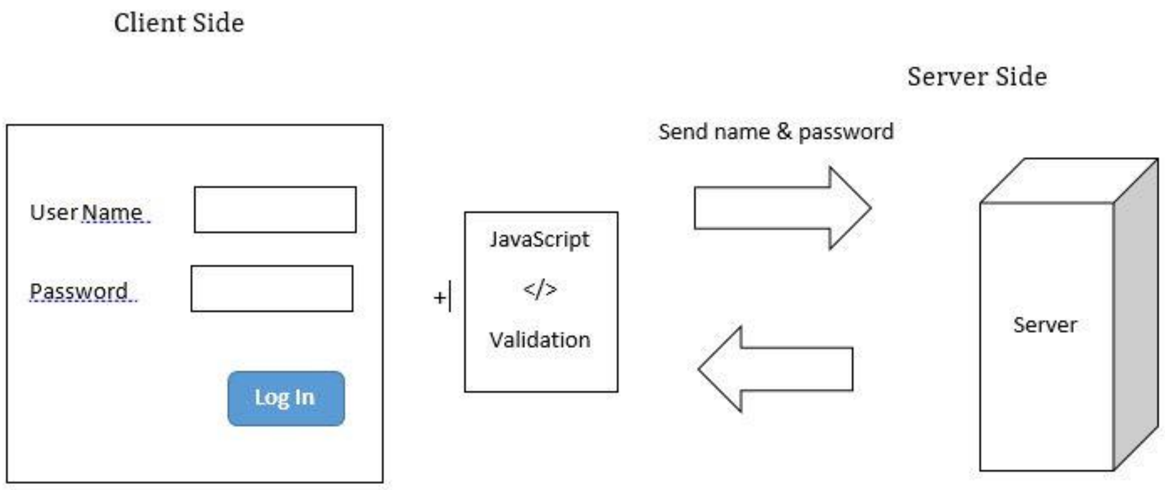
```
var regex= /^\d{10}$/;
console.log(regex.test('9995484545));
//true
```

Match a date with following format DD-MM-YYYY or DD-MM-YY

```
var regex = /^(\d{1,2}-){2}\d{2}(\d{2})?$/;
console.log(regex.test('01-01-1990'));
//true
console.log(regex.test('01-01-90'));
//true
console.log(regex.test('01-01-190'));
//false
```

Client Side Validation

- ☞ Validation is the process of testing and ensuring that the user has entered required and properly formatted information through the web form.
- ☞ Client side validation is the process in which all the validation and error recovery is carried out on the client side or users browser.
- ☞ It is also known as form validation, is a process of verifying whether the data inputted by the users on the web page is complete or not.
- ☞ The data given by the user will be sent to the server to check accuracy. If any mistake persists, that has to be sent back to the user for correction. As this is a time-taking process, the data will be checked by JavaScript before submitting. This will save a lot of time.
- ☞ Efficient web-programming is possible only by making the information secure. JavaScript Forms authentication is defined as a process of validating the user credentials against some authority.
- ☞ Form Authentication will be possible only when JavaScript is used along ASP.net or PHP.
- ☞ When the form is submitted without providing the data in the form field, JavaScript will show a pop-up message restricting the form submission.
- ☞ It includes the following validation task:
    - a. Field check
    - b. Date check
    - c. Numeric text field check
    - d. Pattern check


Client Side
Server Side
Send name & password
User Name
Password
Log In
+
JavaScript
</>
Validation
Server