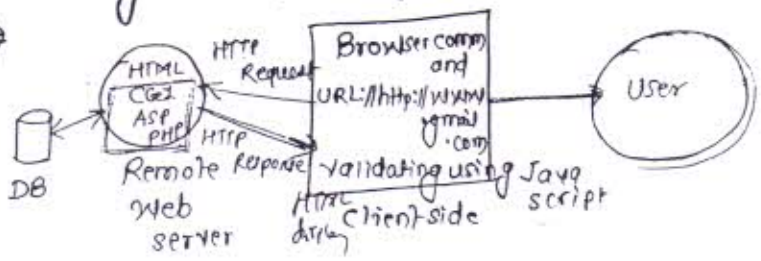
JAVASCRIPTS

JAVASCRIPTS

JAVASCRIPT

Working Principles of Client-side And Server-side Scripting.

⇒



★ <u>Other Client-side Scriptings:</u>

1) BASIC: From this lang. implemented VB Script.
2) PERL: It is client-side scripting not supported by all Browsers.

Netscape Navigator ⟶ Live Script ⟶ JavaScript
By Brendan Eich in 1995

★ JAVASCRIPT ★

★ Difference Betn Scripting & Languages.

Scripting	Language
1) Light weight programing.	1) Heavy weight programing.
Ex: document.write ("NIT"); ← Javascript	Ex: include <stdio.h> main() { — }
2) echo 'NIT'; ← shell	
3) Msgbox ("NIT") ← In VBscript	
4) Print ("Welcome"); ← PERL	
2) Easy to Understand.	2) Complex to Understand.
3) Easy to Implement.	3) Trouble to Implement.
4) No header files required.	4) Header files are Mandatory.
5) No special libraries.	5) Required special libraries.
6) No special compilers.	6) Every lang. they have their own compilers.
7) Interpreter based.	7) Compiler Based.
8) Loosely Typed programing (No variable declaratn), No need to follow comma, semicolon etc.	8) Strictly Typed programing (variable declaratn required semicolon, comma, quotatn mandatory)
9) Validatn purpose	9) Verificatn & validatn Both.
10) Poor Graphics Implementatn	10) Rich Graphics implementatn.
11) Ex: Javascript, VBScript, perlScript etc.	11) Ex: 'C', C++, Java, COBOL, FORTRAN.

★ Introductn to JavaScript:

It is First Web Scripting lang. It is client side scripting lang developed by "Brendan Eich" in Netscape Corporatn at 1995. It is Supporting all major modern browsers like google chrome, Mozilla firefox, Opera, Safari, IE.

Netscape Corporatn came out with a client side Scripting i.e. known as Livescript. It is the First Web scripting lang.

Microsoft Corporatn javaScript version is JScript, It was 1st introduced with IE version 3. (It is initially Livescript, Later Renamed as Javascript)

★ Nature of JavaScript:

It is object Based programing Language. It is based on Object Oriented programing concept. It's syntax is quite similar to C, C++ and Java. It is Easy & Simple to implement.

Q: What is JavaScript?

→ JavaScript is satisfying the following list of pts.

i) It is basically livescript.
ii) It is pure client-side programing tool.
iii) It can integrate with html tags.
iv) It is lightweight scripting.
v) It is interpreted programing.
vi) It is designers programing tool.
vii) It can react to events.
viii) It can read & write html elements.

★ It is used to validate data
★ It is used to create cookies.

vi) It can integrate with any server-side programming.
vii) Open source and cross platform
viii) It creates & Readd with Events
ix) JavaScript can integrate with html & CSS called DHTML.

★ **JavaScript Syntax:**

It consist of group of (Stat.) (Javascript Elements) that are placed within the following Tags. <script> . . . . </script>

Ex:
```
<Script language = "javascript" type= "text/javascript">
    . . . .
    . . . .
</Script>
```

Ex:
```
<script language = "javascript" type="text/javascript">
document.write ("Welcome To Client_side Scripting");
document.write ("<br/>")
document.write ("It is Live Script")
document.write ("<br/>")
document.write ("From Netscape Corporation, First Web Scripting");
</Script>
```

SAVE AS: JavaScript(Dhtml).htm

O/P:
Welcome To Client_side Scripting
It is Live Script
From Netscape Corporation, First Web Scripting

> **Note:** Here, " " (Inverted comma) is very Important If we didn't put Inverted Comma, Then O/P will be blank.

★ **JavaScript Structure:**

Generally JavaScript follows the following structure

```
<html>
  <head>
    <title> example </title>
      <script language ="JavaScript">
        <! -
        - - ->
      </Script>
  </head>
  <body>
    . . .
    . . . . .
  </body>
</html>
```

★ **NOTE:** It is a preferred way to keep. <script> tag within <head> tag. ★

Ex:
```
<html>
<head>
<title>
Working with Javascript (DHTML) ~~scripting~~
</title>
<script language ="JavaScript" type =
    "text/javascript">
document.write("<h1> Welcome to Client
    Side Scripting </h1>");
document.write("<b>It is Live Script </b>")
document.write("<br/>")
document.write("<s> From Netscape Corporation,
    First Web Scripting </s>");
</Script>
</head>
<body bgcolor= lightgreen>
</body>
</html>
```

SAVE AS: JavaScript(Dhtml).htm

O/P: Welcome to Client side Scripting (H1)
It is a live Script → Bold
~~From Netscape Corporation, First Web scripting~~ → Strikethr.

Observation.
Note: Background color is light Green

Semicolon Not Necessary after Parenth.

*Script:

17. SEPT. 2012

Generally Script or <Script> tag contains following 2 Essential Attributes.

1) Language
2) Type.

1) **Language:**
This Attribute specifies what Scripting lang. we are using.

2) **Type:**
This Attribute indicates & Recommend to interpreter what value should be set.

Ex:
```
<script language="javaScript" type="text/javascript">
    JavaScript code
    Javascript code
    Javascript code
</script>
```

JavaScript Comments:
Comments are non Executable stat. or ignore stat. Using this Comment Notat^n we can declare customized stat. as for the End user requirem^n
In JavaScript, Comment Notat^ns are divided into following 2 Types:

1) Single line Comments.
2) Multi line Comments.

1) **Single Line Comments:**
These comments are Restricted to within the line.
The Notat^n is // (Double forward slash).

Ex:
```
document.write("Hello"); // This will write "Hello"
/*
 document.write("<h1> This is a header </h1>");
 document.write("<P> This is a paragraph </P>");
*/
```
} Multiple line Comments.

2) **Multiline Comments:**
These comments are Related to Multiple java stat. are multiple lines.
Notation: /* */
Ex:

* Debugging JavaScript Errors:

Generally JavaScript is a disable component in a Browsers. We should enable JavaScript Option manually.

1) **Enable JavaScript in Internet Explorer:**
i) Click on Tools Menu
ii) Choose Internet Options from Menu
iii) Click the security tab on the Internet Options pop up.
iv) Select Custom level button to access your security Setting.

v) scroll down to Scripting section,
vi) Check Enable Button for Active Scripting.
vii) select Finish & confirm the Msg.

* **<script> Tag Description:**

It is a Tag which indicates script interpreter required to x'late lines Betn <Script> = </Script> These Tags.

If U didn't mention <script> tag, Browser Interpreter are printing as Normal strings. These strings displayed on the webpage.

SAVE AS: JavaScript2.htm

```
<Script type="text/javascript">
  document.write("It is 'Live' Script");
  document.write("<br/>");
  document.write ("It is "JavaScript" from Netscape");
</Script>
```

O/P
It is 'Live' script
It is "JavaScript" from Netscape.

* **Working with Single & double quotes in JavaScript:**

JavaScript supports single & double quotes to produce the same Results. The ~~Above~~ below Example will make U a clear picture on single & double quotes.

SAVE AS: JavaScript3.htm

```
<Script type="text/JavaScript">
  document.write ("It is 'Live' Script");
  document.write ("<br/>");
  document.write ("It is "JavaScript" from Netscape");
  document.write ("JavaScript is "NOT" Java");
</Script>
```

O/P:
It is 'Live' script
It is "JavaScript" From Netscape JavaScript is "NOT" Java.

* **Working with Java Script Special Characters:**

In JavaScript, U can add special characters to ~~indexed~~ a text screen by using / (back slash) sign.

Insert a special character:

black slash is used to insert apostrophes, new lines, quotes & other special characters into a text screen.

```
<Script>
var ~~text~~ txt = "We r the so-called "Good" from the North.";
document.write(txt);
</Script>
```

SAVE AS: JavaScript4.htm

O/P: Blank

In the above Example, We must placed \ (back slash) before each double quote.

```
<Script>
var txt = "We r the So-called \"Good\" from the North.";
document. Write (txt);
</Script>
```

space, space

SAVE AS:
JavaScript4UsingBackSlashes.htm

Output:
We r so-called "Good" from the North

Code	O/P
\'	Single quote
\"	double quote
\\	back slash
\n	new line
\r	Carriage Return
\t	tab
\b	backspace
\f	form feed

★ Whitespaces & Line Breaks:

⇒ JavaScript ignores spaces, Tabs, & Newlines. That appear in JavaScript using this Whitespaces & Line Breaks make Ur Script more Readable.

1) **Whitespace:**

The following Examples are equivalent

```
var name = "nit";
var name =  "nit";
```

JavaScript is Case Sensitive:

JavaScript is basically case-sensitive programing. A fn named "myfunction" is not equal to "myFunction" & a variable named "myVar" is not the same as "myvar".

Break up a Code Line:

We can break up a code line within a txt string with a backslash

SAVE AS: JavaScriptUsingBackSlash.htm

O/P: Hello World!

```
<Script>
document. write ("Hello\
World!");
</Script>
```

Correct Code

```
<Script>
document. write\
("Hello World!");
</Script>
```

Incorrect Code

★ Difference Betn document.write & Window.document.write

⇒ There is No diff. betn the above stat. Here, The Window object is highest level object, It contain other objects & their methods.

Document is object content inside the Window object, Write is a method of document object.

In that situatn, Window object is not Necessary.

Ex:
```
<script>
 document.write ("Helloworld!");
 document.write ("<br/>");
 Window.document.write ("Naresh i Tech");
</script>
```

Ex:
```
<script>
 document.write ("Hello World!");
 document.writeln ("Naresh i Tech");
</script>
```

Ex:
```
<head>
<script>
 function_Show_Message()
  {
   alert ("Welcome To Pop_upBoxes");
  }
</script>
<head>
<body>
<button onclick="show_Message()"
```

★ Semicolons or optional: 18.SEPT.2012

In Javascript, Every stat. should ends with semicolon It is always optional.

Ex:
```
<script>
 document.write ("Welcome To Java Script");
 document.write ("<br/>")
 document.write ("Client side Scripting");
</script>
```

★ Note: ★ closing every stat. with a semicolon always good scripting practice.

★ Script with <html> tags:

```
<script>
document.write ("<font color="#FF0000" size=7 face=tahoma> JavaScript + HTML =
                                                        DHTML</Font>")
document.write ("<br/>");
document.write ("<b> It is bold </b>");
</script>
```

★ Javascript placed html file:

```
<html>
  <head>
    <title> Javascript </title>
    <script>
    </script>
  </head>
  <body>
    <script>
    </script>
```

<script> Tag can be used anywhere. Not compulsory or Restrict^n that <script> tag used only under <body> tag or <head> or any other tag.

In html document we can include javascript part Anywhere. But there are following most preferred ways.

1) Script in <head>.... </head> section
2) Script in <body>..... </body> section
3) Script in <body>... </body> & <head>... </head>
4) Script in External file & then include in <head>... </head> section.

★ Javascript code:

It is a sequence of javascript stat. Each stat. is executed by browser in the sequence they are Return.

```
<Script type = "text/javascript">
document. write ("<h1> This is Heading </h1>");
document. write ("<P> This is a Paragraph </P>");
</script>
```

The above Ex. is JS code. It executes Sequentially.

★ Javascript Blocks:

Javascript stat. can be grouped together in Blocks. The purpose of block is to make the seq. of stat. execute together.

```
<Script type = "text/javascript">
{
document. write ("<h1> This is a Heading </h1>");
document. write ("<P> It is a First Block </P>");
}
{
document. write ("<h1> It is a Second Block </h1>");
}
</script>
```

The above ex. is JS Block. It executes Blockswise.

★ JavaScript POPUP Boxes:

JavaScript has 3 kinds of POPUP Boxes.

1) Alert Box
2) Confirm Box
3) Prompt Box

1) Alert Box: Used to display Alert Popup, when user click on "ok" to proceed to next step.

Syntax: alert ("Message");

```
<html>
  <head>
    <title> POP UP Boxes </title>
   <script>
     alert ("Welcome To MyWebpage");
     alert ("Bye...");
   </script>
  </head>
   <body>
    <script>
      alert ("Welcome To Body of the Page");
      alert ("Bye...");
    </script>
   </body>
  </html>
```

★ How to Write Text on Multiple lines in an alert box?

⇒ We should not use
 Tag here, alert() is a method of window object that can't interprete HTML tags.

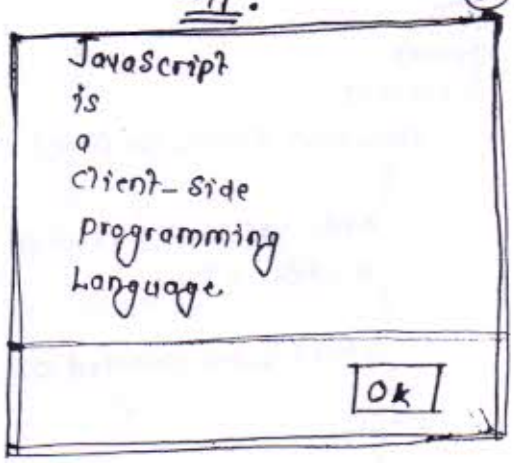
If U break text into multiple lines we should use new line escape character. It consist of (\) Backslash symbol & an Alphabet.

The following are commonly used escape characters.

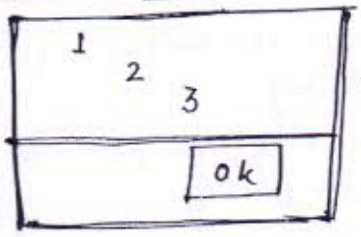
\n : Inserts a New line.
\t : Inserts a Tab.
\r : Carriage return.
\b : Backspace.
\f : Form feed.
\' : Single quote.
\" : Double quote.
\\ : Backslash.

Ex:
```
<Script>
alert ("JavaScript\n is\n a\n client_side\n
        programming\n language");
</Script>
```

⟹ O/P:



Ex: O/P:



⟹
```
<Script>
alert ("1\n\t2\n\t\t3");
</Script>
```

Alert with functions:

Ex:
```
<head>
<Script>
 function show_message()
 {
  alert ('It is alert popup');
  alert ("Bye...");
 }
 </Script>
 </head>
 <body>
 <input type="Submit" onclick="show_message()" value="ShowMe"/>
 </body>
```

2) Confirm Box:

It is often used, If u want to verify or accept something When a Confirm Box ~~observe~~ popup, User should get Ok or Cancel to proceed. If User clicks ok, it returns True, If User clicks Cancel, It returns False.

Syntax:
```
confirm ("Message");
```

Ex:
```
<head>
  <script>
    confirm ("Click ok or Cancel");
  </script>
</head>
```

Ex: 2
```
<head>
<script>
XYZ = confirm ("Click ok or Cancel");
if (XYZ == True)
{
 alert ("User Selected Ok Button");
}
else
{
 alert ("User Selected Cancel Button");
}
</Script>
</head>
```

Ex:

```
<head>
<script>
  function show_confirm()
   {
    xyz = confirm ("click ok or cancel");
    if (xyz == True)
    {
     alert ("User selected ok Button");
    }
    else
    {
     alert ("User selected Cancel Button");
    }
   }
</script>
</head>
<body>
<button onclick="show_confirm()">
          show_message </button>
```

3) **Prompt Box:**

A prompt Box is used to flp a value before entering a page. When a prompt Box pops up, a user should click either Ok or Cancel.

If User selected Ok, it Returns the flp value.

If user select Cancel, it Returns Null.

Syntax:

```
prompt ("sometext","defaultvalue");
```

Ex: ~~script~~ ~~head~~

~~prompt ("Enter any number")~~

```
<head>
<script>
 xyz = prompt ("Enter Any Number", "9");
 alert ("Entered Number is: " +xyz);
</script>
</head>
```

~~Ex:~~

~~<head>~~

~~<script>~~

~~xyz=~~

External JavaScript:

JavaScript can be placed in External Java Script files. Extension is .Js.

Note: External script can't contains <Script> tag.

To use external script, point to the .js file we r using "src" attribute.

To create external JavaScript file, follow the following steps.

Step1: Create JavaScript file & save with .js extension.

```
document.write ("<h1> Welcome to Js
      External programing!! </h1>");
document.write ("<h2> bye.!! </h2>");
```

Step2: prepare html file & call the Java Script.

```
<html>
<head>
<Script type="text/javascript" src="myscript
      .js"></script>
</head>
<body>
</body>
</html>
```

Save file with .html extension or .htm extension

Step1

Ex: 2) Prepare Java Script

```
function show_alert()
{
alert ("Hi");
}

function show_confr()
 {
 confr ("Hellow");
 }
```

Step2:
prepare Html file.

```
<html>
  <head>
   <script type="text/javascript" src="myscript.js"></script>
  </head>
  <body>
  <button onclick="show_alert()"> Show</button>
  <button onclick="show-confr()"> show_confr</button>
  </body>
</html>
```

Date: 20.SEPT.2012

* Working with javascript Variables:

Rule1:

i) Every variable should starts with an Alphabet or underscore.
ii) Variable should not contains unnecessary special characters.
iii) Variable should not contains embedded periods. (dots)
iv) Variables are case sensitive.
v) Count, Count and COUNT are 3 diff. variables.
vi) Variable should be reasonable length.
vii) Variables declared with 'var' keyword.

Ex:
```
var x;
var Carname;
```
variant: collect° of Datatype

These are the Empty variables. If u want, U can assign values to these variables.

```
var x=5;
var name="NIT";
```

Q: Write a Script, Display sum of 2 nos.

⇒
```
<html> <head> <title>variables </title>
<script type="text/javascript">
 var a=10;
 var b =20
 var c =a+b
 document.write("The value of x is" +x);
 document.write("<br>");
 document.write("The value of y is" +x);
 document.write("<br>");
 document.write("The value of z is" +x);
 document.write("<br>");
</script>
</head>
</body>
</html>
```

Q: Write a script to read 2 values, to display sum and difference.

```
<html>
<head>
<title> Variables </title>
<script type="text/javascript">
var x = prompt("Enter any value: ", "10");
var y = prompt ("Enter any value:", "20");
var z=x+y;
var P=x-y;
document.write ("The value of z is" +z);
document.write ("<br>");
document.write ("The value of P is" +P);
</script>
</head>
<body>
</body>
</html>
```

★ String Variables:

In Javascript, String variables should be enclosed in either single or double quotat^n marks.

Ex: greeting = "Hello there"
Warning = 'Be Careful'

U may include a single quote within a double quoted string. OR

A double quote within a single quoted string.

Ex:
greeting = "\"Hello there\" is a greeting"
Warning = '\'Be careful\' is a warning'

★ Numeric Variables:

Creating a Numeric Variable is as simple as assigning a value.

Ex: Count = 42
temperature = 98.4

★ <noscript> tag:

Used to provide alternate content for users, when Ur browser unable to support javascript, intersitual^n no script will be executed.

Syntax:

```
<noscript> . . . . </noscript>
```

Ex:

```
<html>
<head>
<title> NOSCRIPT TAG </title>
<script>
  function show_Alert()
  {
  alert ("Working with NOSCRIPT");
  }
</script>
</head>
<body>
<noscript>
 OOPS U r using too lower versions
 Is unable to support...
</noscript>
<input type="Submit" onclick="show_
   Alert()" value="NOSCRIPT"/>
</body>
</html>
```

Note:

If Browser supports, Script part will be executed, otherwise Noscript part will be Executed.

★ JAVASCRIPT Operators: ★

In Javascript, Operators are divided into following types.

1) Arithmetic Operators:

Using these operators we can perform arithmetic operat^n for mathematical operat^n.

Following Table describes operator descript^n Example.

Operator	Descript^n	Example
+	Addition	j+12
−	Subtraction	j−22
*	Multiplication	j*7
/	Division	j/3.14
%	Modulus (division Remainder)	j%6
++	increment	++j
−−	Decrement	−−j

2) Assignment/Comparison:

Comparison Operators are Generally used to Compare values, The following table describes operator descript^n & example.

Operator	Descript^n	Example
==	is equal to	j==42
!=	Is not equal to	j!=17
>	Is greater than	j>0
<	Is less than	j<100
>=	Is greater than or Equal to	j>=23
<=	Is less than or Equal to	j<=23
===	Is equal to (and of the same type)	j===56
!==	Is not equal to (and of the same type)	j!=='1'

3) Logical Operators:

To compare logical Expressions.

operator	Description	Example
&&	And	j==1 && k==2
||	Or	j<100 || j>0
!	NOT	!(j==k)

★ String Concatenation: (+)

The + operator used on strings. To add 2 string variables this operator required.

Ex:

```
Str1 = "Nil";
Str2 = "Pin2";
Str3 = Str1+Str2;
```

Ex: 2⇒ In a prog. 2 prompt boxes existed If U r reading values thr. prompt, By default it is treating as "Strings."

Ex: Nil 88 28
28 Nil Nil

```
<html>
<body>
<Script type="text/javascript">
var x;
x=8+8;
document.write(x);
document.write("<br/>");
x=8+"5";
document.write(x);
document.write("<br/>");
x="8"+"8";
document.write(x);
document.write("<br/>");
</script>
</body>
</html>
```

★ Working with Conditional Control statements:

JavaScript contains following conditional stat.

1) If statement
2) If.. else statement
3) If..else If..else statement
4) Switch statement

1) If statement:

Using this statement, we can execute a part of the code, If a specified cond^n is True.

Syntax:

```
if (Condition)
{
Code to be executed if cond^n true
Code to be executed if cond^n true
}
```

Ex:

```
<html>
  <head>
    <title> If condition </title>
    <script>
      Var x = prompt ("Enter any no:", '10')
      if (x==10)
      {
      alert ("Entered Number is Equal:");
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

2) If...else:

Using this stat, we can execute a part of code if the gvn cond^n is True, we can execute another part of code ~~with~~ if the gvn cond^n is False.

Syntax:

```
if(Condition)
{
Code to be executed if cond^n is True
Code to be executed if cond^n is True
}
else
{
Code to be executed if cond^n is False.
Code to be executed if cond^n is False
}
```

Ex:

```
<html>
  <head>
    <title> If condition </title>
    <script>
    Var x=prompt("Enter any no: ",'10')
    if (x>10)
    {
    alert ("Entered no. is Greater");
    }
    else
    {
    alert ("Entered no. is Less");
    }
    </script>
  </head>
  <body>
  </body>
</html>
```

3) if... else if... else:

By using we can select one of select block of code to be executed.

Syntax:

```
if (condition 1)
{
code to be executed if cond^n 1 is True
}
else if (condition 2)
{
code to be executed if cond^n 2 is True
}
else
{
code to be executed if neither cond^n 1 nor cond^n 2
}
```

Ex:

```
<html>
<head>
<title> if...else if statement </title>
<script type="text/javaScript">
Var b = prompt ("Enter any number", "100");
if(b>100)
{
alert ("Number is Greater");
}
else if (b<100)
{
alert ("Number is less");
}
else
{
alert ("Number is equal");
}
</script>
</head>
<body>
</body>
</html>
```

Ex:

```
<html>
<head>
<title> if...else if statement </title>
<script type="text/javaScript">
Var book = prompt ("Enter Ur favourite book", "html, CSS, JS, HTML5");
if(book=="html")
{
alert ("U r selected HTML");
}
else if (book=="CSS")
{
alert ("U r selected CSS");
}
else if (book=='js')
{
alert ("U r selected js");
}
else if (book=="HTML5")
{
alert ("U r selected HTML5");
}
else
{
alert ("Invalid Book");
alert ("Thank U...");
}
</script>
</head>
<body>
</body>
</html>
```

~~4) Switch statements~~ —

```
<html>
 <head>
 <title> if...else if statement Button</title>
 <script type="text/javascript">
 function fav_book()
 {
 var book=prompt("Enter Ur favourite book
                    (html, css, js, HTML5)");
  if (book=='html')
  {
  alert ("U r Selected HTML");
  }
  else if (book=='css')
  {
  alert ("U r selected CSS");
  }
  else if (book=='js')
  {
  alert ("U r selected javascript");
  }
  else if (book=="HTML5")
   {
   alert ("U r selected HTML5");
   }
   else
   {
   alert ("Invalid Book");
   alert ("Thank U");
   }
 }
</script>
</head>
 <body>
<button onclick="fav_book()"> fav-book </button>
</body>
</html>
```

4) Switch Statements:

By using switch case statement we can select a specific lock code to be executed. Switch case stat. are alternative ways of executing. It is an Enhanced version of if...else stat.

Syntax:

```
Switch(n)
{
 case1:
    execute code block1
    break;
 case2:
    execute code block2
    break;
 default:
    Code to be executed if n is different
    from case1
}
```

Ex:

```
<html>
<head>
 <title> Switch case Statements </title>
 <script type="text/javascript">
 var alpha = prompt("Enter any Alphabet
                    :", "A");
 Switch (alpha)
 {
 Case 'A';
 alert("It is a Book");
 break;
 Case 'B';
 alert("It is a Book");
 break;
 Case 'C':
 alert ("It is a Book");
 break;
 default:
 alert ("Invalid Book");
}
```

Ex: <html>

~~<head>~~

<title> Switch With Date </title>

<head>

<script type="text/javascript">

function MyDay()

{

var x;

var d = new Date().getDay();

switch (d)

{

case 0:

x = "It is Sunday";

break;

case 1:

x = "It is Monday";

break;

case 2:

x = "It is Tuesday";

break;

case 3:

x = "It is Wednesday";

break;

case 4:

x = "It is Thursday";

break;

case 5:

x = "It is Friday";

break;

case 6:

x = "It is Saturday";

break;

}

document.getElementById("demo").innerHTML = x;

}

</script>

</head>

<body>

 Click the button to display what day it is Today

<button onClick="MyDay()">Today</button>

</body>

</html>

~~Ex: <html>~~

~~<head>~~

~~<title> Switch with colors </title>~~

~~<head>~~

★ JavaScript Keywords / Reserved Words:

JavaScript contains following list of Keywords or Reserved Words.

abstract, boolean, break, byte, case, catch, char, class, const, continue, debugger, default, delete, do, double, else, enum, extends, false, final, finally, float, for, function, goto, if, implements, import, ininstanceof, int, interface, long, native, new, null, package, private, protected, public, return, short, static

★ JavaScript Looping Control Statements:

In JavaScript, there are 2 kinds of loops.

1) for loop
2) while loop

1) for loop:

We can execute block of stat. repeatedly until the gvn cond^n is false. If false, it terminates the loop. It contains following 3 parameters.

i) An Initializat^n Expression
ii) A condit^n Expression
iii) A Modificat^n Expression.

The above parameters are seperated by semicolon

→ for (expr1; expr2; expr3)

Syntax:

for (initializat^n; test cond^n; iterat^n stat)

{

Stat.(s) to be executed if test cond^n is True

Stat (s) to be executed if test cond^n is True

Stat (s) to be executed if test cond^n is True

}

Q: A script To display 1 To 10 nos.

⇒
```
<html>
  <title> Working with for </title>
  <head> ~~<body>~~
  <body
  <Script type="text/javascript">
   for (i=0; i<=6; i++)
   {
    document.write("The no. is" +i);
    document.write("<br>");
   }
  </Script>
 </body>
</html>
```

Q: A script to display HTML Headings styles.

Ex:
```
<html>
  <head>
   <title> For loop </title>
  </head>
  <body>
  <Script type="text/javascript">
  for (i=0; i<=6, i++)
  {
   document.write ("<h" +i+ "> This is Heading" +i);
   document.write ("<h"+i+">");
  }
  </Script>
 </body>
</html>
```

No space

Output:

2) While loop:

It executes blocks of stat. Repeatedly until the gvn cond is False, If False, it Terminates the loop.

Syntax:
```
While (variable <= endvalue)
  {
   code to be executed
   code to be executed
  }
```

Ex:

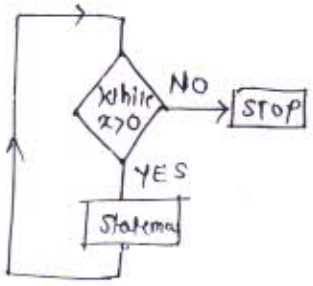
```
<html>
 <head>
 <title> While Loop </title>
 </head>
 <body>
 <script type = "text /javascript">
var i = 0;
  while (i<=5)
  {
   document. write("The value is:" + i)
   document. write ("<br/>");
   i++;
  }
 </script>
 </body>
</html>
```

3) **do - while loop:**

Executes blocks of stat repeatedly n+1 times.

Syntax:

```
do
 {
 Code to be Executed
 Code to be Executed
 }
while (variable<=end value);
```



Ex:

```
<html>
 <head>
 <title> do_ while loop </title>
 </head>
 <body>
 <script type = "text/javascript">
 var i = 0;
 do
  {
   document. write ("The value is:" +i);
   document. write ("<br/>");
   i++;
  }
 while (i<=5)
 </script>
 </body>
 </html>
```

★ Working With break And Continue :

1) break :

Using this stat. we can break execut^n of the prog. under a specific cond^n.

Ex:
```
<script type="text/javascript">
 i=0;
 do
 {
 if (i==2)
  {
   break;
  }
document.write("The no.is" +i);
document.write("<br/>");
i++;
}
while (i<=5);
</script>
```

2) Continue :

Using this stat. we can execute group of stat. repeatedly, within the gvn cond^n matching the stat. not executed.

Ex:
```
<html>
 <body>
  <script type="text/javascript">
  Var i=0;
   for (i=0; i<=10; i++)
    {
     if (i==4)
      {
       continue;
      }
     document.write("The number is" +i);
     document.write("<br />");
    }
  </script>
 </body>
</html>
```

Output :

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10

★ Function ★

A f^n is a block of code that will be executed only by an occurence of an Event at that time f^n is called.

★ Working with JavaScript Function:

Funct^n is a group of Reuseable code which can be called anywhere in Ur prog. This Eliminates the need of writing same code Again And Again.

This will help programmers to write modular code.

U can divide Ur big prog. in a no. of small & manageable f^ns.

It is called Code Reuseability.

Functions can be defined Both in the head and body sections.

Syntax:

```
function functionname(var1, var2, ..., varx)
{
 specify some code
 specify some code
 specify some code
}
```

Ex:

```
<html>
  <body>
   <script type="text/javascript">
   function HTML_Headings()
   {
   for (i=1; i<=6; i++)
   {
   document.write("<h" +i+">The Heading is:" +i);
   document.write("</h"+i+">");
   }
  }
  </script>
 </head>
 <body>
 <button onclick="HTML_Headings()">HTML_Headings</button>
 </body>
 </html>
```

Output:

The Heading is: 1
The Heading is: 2
The Heading is: 3
The Heading is: 4
The Heading is: 5
The Heading is: 6

Ex: Write a Script To display msg From Textbox.

```
⇒ <head>
  <script>
  Function DisplayMessage(message)
  {
   alert("Ur message is:" + message);
  }
  </script>
 </head>
 <body>
  <Form>
   Enter Message: <input type='text' name='message'>
                  <input type='Submit' onclick="DisplayMessage(Form.message.value)"
                 value="DMessage"/>
  </Form>
 </body>
```

★ **Function Calling :**

In JavaScript fn is divided into the following 2 parts:

1) Called function
2) Calling function

Ex:

```
<script language="JavaScript">
 function showAddress()
 {
  document.write("Subba Raju");
  document.write("<br>");
  document.write("Sr. Quality Analyst");
  document.write("<br>");
  document.write("Naresh i Technology");
  document.write("<br>");
  document.write("Hyderabad");
 }
 </script>
 <script language="JavaScript">showAddress();
 </script>
```

Called

Calling

Output :

Q: Write a Script to Return Values from a function.

⇒
```
<head>
<script>
    function myFunc()
    {
     var a=10;
     alert("My value is: "+a);
    }
<\/script>
<\/head>
<body>
<p> Click the button to display the value... <\/p>
<input type="Submit" onclick="myFunc()" value="Display_value"/>
<\/body>
```

★ **Working With Return Stat:**

It is used to specify the value i.e. return from fn. A JavaScript fn can have an optional return stat. This is required if U want to return a value from the fn.

```
<script>
function myFunc()
 {
  return("WelCome to JavaScript");
 }
<\/script>
<script>
document.write(myFunc());
<\/script>
```

Q: Write a Script to Return sum of 2 nos.

⇒
```
<script>
    function add(x,y)
     {
      return x+y;
     }
<\/script>
<script>
document.write("The Sum is: " + add(2,3));
<\/script>
```

★ Calling a f^n with Arguments:

When U call a f^n, U can pass some values to it, These values r called Arguments or parameters. These Arguments can be used inside the f^n. U can send as many arguments as U like seperated by commas. (,)

Syntax:

```
Function myFunction (var1, var2)
{
 Js statements
 Js statements
 Js statements
}
```

Ex:

```
<head>
<script>
function Emp (name, job)
{
alert ("You are" +name+ ", Job is:" +Job);
}
</script>
</head>
<body>
<button onClick="Emp ('ksRaju', 'S Engineer')"
    > passingArgs </button>
</body>
```

Ex:

```
<head>
<script>
function Emp_Details (name, job)
{
alert ("You are" +name+ ", Job is" + Job);
}
</script>
</head>
<body>
<button onClick="Emp_Details ('ksRaju', 'S Engineer')">
                    ksRaju </button>
<button onClick="Emp_Details ('Ravi', 'Sr. Engineer')">
                    Ravi </button>
```

Lifetime of JavaScript Variables:

1) Local JavaScript Variables:

A variable declared within JavaScript fn becomes local that we can access within the fn. U can have local variables with the same name in diff. fns. Local variables are recognized by the fn in which they are declared. Local variables r deleted as soon as the fn Execun is completed.

Ex:
```
<script>
 year = 2011;
 function Global_Local()
 {
  month=2;
 }
 Global_Local();
 document.write("year is:" +year
                 +", Month is:"+month);
</script>
```

```
<script>
 var z= 10; ⇒ Global variable
 function fun1()
 {
  var x=10; ⇒ Local variable
 }
 function fun2()
 {
  var y=20; ⇒ Local variable
 }
 function fun3()
 {
  var x=30; ⇒ Local variable
 }
</script>
```

2) Global JavaScript Variables:

Variables declared outside the fn becomes Global These variables can access in all scripts & fns. Global variables are deleted when U close the page.

Ex:
```
<script>
 year=2012;
 function_Local_Variables()
 {
  month=2;
 }
 function Global_Local_Variables()
 {
  month=4;
 }
 Local_Variables();
 document.write("year is:" +year+", month is:" +month);
 document.write("<br/>");
 Global_Local_Variables();
```

* Background Colors:

In Javascript, bgcolor is a property in document object. By using this property, we can change background color of webpages. The Ex. describes, bgcolor property.

```
<head>
 <script>
  Function Bg_Color()
   {
    document.bgcolor="red";
   }
   ~~document~~function Bg_color1()
    {
     document.bgcolor="green";
    }
    function Bgcolor2()
     {
      document.bgcolor="blue";
     }
 </script>
</head>
<body>
 <input type="submit" onclick="Bg_Color()" value='color_Red'/>
 <input type="submit" onclick="Bg_color1()" value='color_Green'/>
 <input type="submit" onclick="Bg_color2()" value='color_Blue'/>
</body>
```

document.body.bgcolor: [Imp: Without (body) Tag, The O/P will not Generate]

This property executes only in body part of the webpage.

Method1

```
<html>
<body>
<script> document.body.bgcolor="teal"; </script>
<P> The background color was changed by a Script </P>
</body>
</html>
```

Method2

```
<html>
<body>
<input type="submit" onclick="document.body.bgcolor="teal" Value='Click Me'>
<P> the background color was changed by a script </P>
</body>
</html>
```

Q: Write a script to change webpage background color with the help of dropdown.

⇒
```
<body>
<Form name="bgcolorform"> Try it Now: Select onChange="if (this.selectedIndex!=0)
document.bgColor= this.options [this.selectedIndex] option value="choose"> set
    background color option value="FF0000">Dark Red
                       option value="00FF00">Dark Green
                       Option value="0000FF">Dark Blue
                       Option value="000000">Black
                       Option value="FFFFFF">White
  </select>
 </Form>
</body>
```

* Real-time Application on JavaScript Application:

```
<html>
 <head>
  <title> JavaScript Tutorial </title>
  <Script language="JavaScript">
  Function getFullName()
  {
   var firstName, lastName, fullName;
   firstName= document.CreditApplication.txtFirstName.value;
   lastName= document.CreditApplication.txtLastName.value;
   fullName= firstName + " " +lastName; document.CreditApplication.txtFullName
   ~~docum~~                                .value = fullName;
  </script>
 </head>
<body>
  <h2> RealTime Application </h2>
   Font face="Verdana, Tahoma, Arial" size="3">
   Form Name="Credit Application">
  <table border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td width="100"> FirstName: </td>
    <td><input type="text" name="txtFirstName" size="14"></td> </tr>
  <tr>
    <td width="100"> Full Name </td>
    <td><input type="text" name="text~~full~~Firstname" size="30"></td></tr>
<tr>
    <td Width="100"><br><br> </td>
  <td>
```

RealTime Application

First Name: [ ]

Last Name: [ ]

Full Name: [ ]

[Show FullName]

```
<input type="button" value="Show Full Name" onclick="getFullName()">
</td>
</tr>
</table>
</form>
```

Built_in Functions:

JavaScript contains several built_in f^n like ~~alert~~

1) alert
2) confirm
3) prompt etc.

Javascript print f^n:

This send the contents of Webpage to the user's printer. It is equivalent to file menu print option.

This print method we should click in a event. When user performing the event, the method will be executed.

```
<body>
  <form>
    <input type="submit" value="printThisPage" onclick="window.print()"/>
  </form>
</body>
```

javascript_window.location ~~=~~:

Using this property, we can redirect current Webpage. Whenever U redirect, U should mention a new location.

```
<body>
  <script> window.location = "http://www.nareshit.com";
  </script>
</body>
```

25. SEPT. 2012

★ JavaScript Events:

By using javascript, we have the ability to create dynamic Webpages. Events or Actions that can be detected by javascript. Every element on a Webpage has certain Events which can Trigger in JavaScript.

Ex:

We r using onClick event on a button, that indicates when user click on button it returns the f^n returned value. Events r normally used in Combinat^n with f^n

Ex: Clicking a button
A page is finished loading.
An image is finished loading.
Moving mouse-cursor over an element.
Entering an I/P field.
Submitting a form.

* javascript Events:

Event	Description
i) abort	occurs when the user cancels loading of an image.
ii) click	occurs when the user clicks on a link or form element change occur
iii) error	occurs when an error happens during loading of a document or img
iv) focus	occurs when s/p is focus, gvn to a form element or a window
v) load	occurs when a page is loaded into Navigator.
vi) mouseout	occurs when the user moves the ptr off
vii) mouseover	occurs when the user moves the ptr over
viii) reset	occurs when the user clears form using reset button.
ix) select	occurs when the user selects a form elements field.
x) submit	occurs when a form is submitted.
xi) unload	occurs when a user leaves a page.

Ex:

onClick:

This event occurs, when user clicks on an Element.

Syntax:

1) In HTML,

```
<element onClick = " SomeJavaScriptCode">
```

2) In JavaScript,

```
Object.onClick = "SomeJavaScriptCode"
```

* The following html elements, not supported onClick Event:
<base>, <bdo>, <br>, <head>, <html>, <iframe>, <meta>, <param>, <script>, <style> and <title>.

* onclick supported JavaScript Object:

Document, Window.

Ex:

```
<script>
function Myfunc()
{
alert ("It is called function");
}
</script>
<body>
<button onclick="Myfunc()"> ClickMe </button>
</body>
```

Element → button, Event → onclick, JsCode → Myfunc()

Ex:
```
<head>
  <script>
    function Display_Date()
    {
    document.getElementById('demo').innerHTML = Date();
    }
  </script>
</head>
<body>
<button onclick="Display_Date()"> Display_Date </button>
<p id='demo'> </p>
</body>
```

★ **onload:**

This event occurs, when an object has been loaded. It is most often used within the body element to execute a script once a webpage has completely loaded.

Syntax:

1) In HTML, :⇒ <element onload="SomeJavaScriptCode">

2) In JavaScript :⇒ object.onload = "someJavaScriptCode"

Supported <HTML> Tags:

<body>, <frame>, <frameset>, <iframe>, <img>, <link>, <script>, <style> etc.

Supported JavaScript Objects:

Document, Window, XMLDocument, XMLHttpRequest

Ex:
```
<head>
  <script>
    function Page_Load()
    {
    alert ("Page Loaded Successfully");
    }
  </script>
</head>
  <body onload="Page_Load()">
  <P> To load the page Refresh or Reopen </P>
</body>
```

Ex: 2

```
<head>
  <script>
    function Image_Load()
    {
    alert ("Image Loaded Successfully");
    }
  </script>
</head>
<body>
  <P> To load the image Refresh or Reopen - - <IP>
    <img src = "html5.png' height = 60 width = 70 onload = "Image_load()" alt='html logo'
  />
</body>
```

* **Onunload:**

This event occurs once the page has unloaded. onunload occurs when user navigates away frm the page. (By clicking on a link, Submitting the form, closing the Browser Window)

Syntax:

1) In HTML: <element onunload="SomeJavaScript Code">

2) In JavaScript: Object.onunload = "SomeJavaScript Code"

Supported HTML Tags <body>, <frameset>

Supported JavaScript objects Window.

Ex:

```
<head>
  <script>
    function Page_unload()
    {
    alert ("Page Unloaded Successfully");
    }
  </script>
</head>
<body onunload = "Page_unload()">
<P> Page Refresh or Closed the Page unload occurs. . . <IP>
</body>
```

Ex:

```
<script>
 function imgBig(x)
 {
   x.style.height="100px";
   x.style.width="100px";
 }
 Function imgSmall(x)
 {
  x.style.height = "50px";
  x.style.width = "50px";
 }
</script>
 <body>
   <P> Take the Mouse ptr. to the Image. . . </p>
   <img src="html5.png" onmousemove="ImgBig(this
      onmouseout="ImgSmall(this)" width=50 height=50 </img>
   </body>
```

Onmousemove:

When user moves mouse ptr. over an element, Then that event occurs.

Syntax:

1) In HTML:

```
<element onmousemove="SomeJavaScriptCode">
```

In JavaScript:

```
object.onmousemove = "SomeJavaScriptCode"
```

Not supported in HTML

<base>, <bdo>, <br>, <head>, <html>, <iFrame>, <meta>, <param>, <script>, <style> and title

Supported javascript object

document, window

26. SEPT. 2012

Ondoubleclick:

This event occurs, when user double click on an element.

Syntax:

1) In HTML:

```
<element ondoubleclick="SomeJavaScriptCode">
```

2) In JavaScript:

```
object.ondblclick="SomeJavaScriptCode"
```

Ondoubleclick not supported elements:

<base>, <bdo>, <br>, <head>, <html>, <iframe>, <meta>, <param>, <script>, <style>, <title> etc.

Supported Javascript Objects:

Document, Window.

Ex:1)
```
<head>
<script>
Function Display_Date()
{
document.getElementById("demo").innerHTML=Date();
}
</script>
</head>
<body>
<P> Double click on following Button...</P>
<button ondblclick= Display_Date()"> Display</button>
<p id='demo'></P>
</body>
```

Ex:2)
```
<html>
<head>
<script>
function Copy_Text()
{
document.getElementById("field2").value=document.getElementById("field1").value;
}
</script></head>
<body>
Field1: <input type="text" id="field1" value=
Field 2: <input type="text" id="field2" "NareshiTech"/><br/>
<br/><br/>
```

O/P:

Field1: [Hello World!] → U can put any name

Field2: [ ]

[Copy Text] → On Double Click Cursor move on field 2 and copy first field1 content.

★ onmouseOver And onmouseout:

onmouseover event occurs when you bring Ur mouse ptr. over an element. onmouseout event occurs when you takes Ur mouse out of an element.

Ex:1)
```
<A HREF="http://www.nareshit.com/" onmouseover="document.bgColor='#FFFF00'
onmouseout='document.bgColor='#FFFFEE'">
Move Ur Mouse Over Me!
</A>
```

Ex2:)
```
<script>
function M_Over()
{
alert("Mouse Over");
}
function M_Out()
{
alert("MouseOut");
}
</script>
<body>
<button onmouseover="M_over()" onmouseout="M_out">
Bring Mouse pointer </button>
</body>
```

Ex:

```
<marquee onmouseover="this.stop()" onmouseout="this.start()">
    Bring Pointer </marquee>
```

Marquee Tag with ondoubleclick

```
<marquee behaviour="scroll" direction="left"
    scrollamount="6"
    id="marquee1">DblClick </marquee>
<input type="button" value="slower" ondblclick="document.getElementById('marquee1').
    setAttribute('scrollamount', 1, 0);">
```

o/p:

DblClick

Slower | Medium | Faster

What are Event Handlers?

⇒ Event Handlers are scripts, which programmer using to handle events. In JavaScript, There are the following list of Event Handlers.

Object	Event Handlers Available
i) Selection List	onBlur, onChange, onFocus.
ii) Text Element	onBlur, onChange, onFocus, onSelect
iii) Textarea Element	onBlur, onChange, onFocus, onSelect.
iv) Button Element	onClick
v) checkBox	onClick
vi) Radio Button	onClick
vii) Hypertext Link	onClick, onMouseOver, onMouseOut
viii) Clickable Imagemaps (Area)	onMouseOver, onMouseOut
ix) Reset Button	onClick
x) Submit Button	onClick
xi) Document	onLoad, onUnload, onError
xii) Window	onLoad, onUnload, onBlur, onFocus
xiii) Framesets	onBlur, onFocus
xiv) Form	onSubmit, onReset
xv) Image	onLoad, onError, onAbort

Form Events:

The following Events are at Form level.

1) **onBlur**: This Event occurs, when we r leaving the object. It is very useful in form validat^n.

Mainly used for validation

Note: onBlur Event is opposite of onFocus.

Syntax: In HTML:

```
<element onblur="someJavaScriptCode">
```

In JavaScript:

```
object.onblur= "SomeJavaScript Code"
```

Not Supported HTML Elements:

<base>, <bdo>, <br>, <head>, <html>, <iframe>, <meta>, <param>, <script>, <style> & <title>

Supported JavaScript Objects:

Document, Window.

Ex:

```
<html> <head>
<script>
    function Upper_Case()
    {
      document.getElementById ('fname');
        x.value =x.value.toUpperCase();
    }
    function Lower_Case()
    {
      document.getElementById ('Lname');
        x.value=x.value.toLowerCase();
    }
</script> </head>
<body>
First Name: <input type="text" onblur="Upper_Case()" id='fname'/>
Last Name: <input type="text" onblur="Lower_Case()" id='Lname'/>
</body>
</html>
```

2) **onFocus**: This Event occurs, when an element gets focused.

onFocus is most often used with <input> tag, <select> tag & <a> tag.

Syntax:

1) HTML, <element onFocus="SomeJavaScriptCode">

2) JavaScript,

object.onFocus= " SomeJavaScript Code"

Not Supported Elements: <base>, <bdo>, <br>, <head>, <html>, <iframe>, <meta>, <param>, <script>, <style> and <title>

Supported HTML Elements: document, window.

Ex:

```
<head>
<script>
Function SetStyle(x)
{
document.getElementById(x).style.background='green';
}
</script>
</head>
<body>
First Name: <input id='fname' onFocus="SetStyle(this.id)">
<br>
Last Name: <input id='lname' onFocus="SetStyle(this.id)">
</body>
</html>
```

3) **onResize**: This Event occurs, when the size of an Element has changed.

Syntax:

1) HTML:

<element onresize="SomeJavaScript Code">

2) JavaScript:

object.onresize="SomeJavaScript Code"

Ex:

```
<script>
Function ShowMsg()
{
alert ("Window Resized");
}
</script>
<body onresize="ShowMsg()">
<P> Restore or Maximize the Browser.</P>
</body>
```

* Working With JavaScript Errors And Exceptions.

There are 3 Types of Errors in programming.

1) Syntax Errors
2) Runtime Errors
3) Logical Errors

1) Syntax Errors:

These are called parsing Errors, occured at compiletime For Traditional programing lang, At Interprete time for javaScript.

Ex:
```
<script type="text/javascript">
 <!...
  Window.print (;
  //.->
 </script>
```

In above Ex, Syntax Error occurs, In JavaScript a block contains a Syntax Error. It is affected to complete Block.

2) Runtime Errors:

These r called Exceptions occured during Execut^n.

Ex:
```
<Script type="text/javaScript">
 <!..
  Window.printmec();
  //..>
 </script>
```

3) Logical Errors:

These r the most difficult types of Errors to Track down, These Errors r not the result of syntax or Runtime Error. It is the mistake in the logic that drives your script. These errors unable to catch Bcoz, it depends on Ur Business Requirements.

* JavaScript try_catch... finally statements:

That stat. allows U to test a block of code for Errors. In Latest JavaScript version, added Exception Handling Capabilities.

Catching Errors:

When browsing webpages on Internet, sometimes JavaScript alert box displays asking "Do U wish to Continue?" or "Do U wish to debug?" These msgs r Very useful for Developers, not End users.

try-catch statement:

It allows U to test, a block of code for Errors. That try block contains the code to be run, catch block contains code to be executed, If an Error occurs.

Syntax:

```
<script>
try
{
Code to run [break;]
}
catch(e)
{
Code to run if an excepⁿ occurs [break;]
}
</script>
```

Ex:

```
<html>
<head>
<script>
function MyFun()
{
var a=10;
alert("The value is: " + a);
}
</script>
</head>
<button onClick="MyFun()"> Click </button>
</body>
</html>
```

O/P:

Ex:

```
<html>
<head>
<script>
function MyFun()
{
var a=10;
try
{
alrt("The value is: " +a);
}
catch(e)
{
alert(e.description)
}
}
</script>
</head>
<body>
<button onClick="MyFun()"> Click </button>
</body>
</html>
```

O/P:

Ex:

```
<script>
alrt("Welcome To Try");
alert("Thank U..");
</script>
```

In the above script, There is runtime Error, We should handle with try... catch block. like as follows:

```
<script>
try
{
alrt("Welcome To Try");
}
catch(e)
{
alert(e.description)
}
alert("Thank U..");
</script>
```

O/P:

finally: By using this stat. we can display block of statements which r need to execute. It is not related to catch block.

Ex:
```
<script>
 function MyFun()
 {
 var a=10;
 try
 {
 alert("The value is: "+a);
 }
 catch(e)
 {
 alert(e.description)
 }
 finally
 {
 alert("Always Executing");
 }
}
```

28. SEPT. 2012

throw statement:

This stat. allows U to create an Exceptn. If U use this stat. together with try-catch stat., U can control the flow of the prog.

Syntax:
throw exception

Note: throw is written in lowercase letters. If U write uppercase javascript generates an Error.

Ex: throw statement

```
<script>
var x=prompt("Enter the no. betn 5 and 10");
try
{
if (x>10)
 {
  throw "Err1";
 }
else if (x<5)
  {
   throw "Err2";
  }
else if(isNaN(x))
  {
   throw "Err3";
  }
}
catch (err)
 {
  if(err=="Err1")
   {
    document.write("Error! The value is too High.");
   }
  if(err=="Err2")
   {
    document.write("Error! The value is too Low.");
   }
  if(err="Err3")
   {
    document.write("Error! The value is not a number");
   }
</script>
</body>
</html>
```

In above Example, a given value is higher than 10 or less than 5 or not a no. We r going to throw an Error. That Error is caught by the catch argument & displays proper error msg.

Window.onerror: It is a feature in JavaScript, The Error Event is fired whenever a mistake available on window object. It is an Event handler whenever an Exceptn occured.

Ex:
```
<Script>
 Window.onerror=function()
 {
 alert("Error Handled");
 }
</Script>
<body>
<button onclick="Myfunc()">
      ClickMe </button>
</body>
```

The onError event handler provides 3 pieces of informn.

1) Error Msg.
2) URL
3) Line Number.

1) **Error Message:**
The same msg that the browser would display for the given error.

2) **URL:**
The file in which the error occured.

3) **Line Number:**
The line no. in the gvn URL caused the Error.

Ex:
```
<Script type="text/javaScript">
window.onerror=function(msg,url,line)
{
 alert("Message:"+msg);
 alert("Url:"+url);
 alert("Line number:"+line);
}
</Script>
</head>
<body>
<P> Click the following to see the Result! </P>
<form>
<input type="button" value="Click me"
   onclick=
</form>
```

★ Working With JavaScript Objects

In the javascript objects are the following Types.

1) Array Objects
2) Boolean Object
3) Date object
4) Math object
5) String object
6) Number Object
7) RegExp object
8) Global properties & Functions.

Browser objects:

1) Window Object
2) Navigator object
3) Screen Object
4) History object
5) Location Object

Core DOM objects:

1) DOM Node
2) DOM NodeList
3) DOM NamedNodeMap
4) DOM Document
5) DOM Element
6) DOM Attr

HTML DOM objects:

1) Document Object
2) Event Object
3) HTMLElement Object
4) Anchor Object
5) Area Object
6) Base Object
7) Body Object
8) Button Object
9) Form Object
10) Image Object
11) Input Button object
12) Input Checkbox object
13) Input File object
etc.

Methods:

Methods are actions that can be perform an object.

Ex: Methods
Resize
Close
open
Maximize
Minimize
Restore etc.

Ex:
```
<script>
var txt = "Naresh i Technologies";
document.write(txt.toUpperCase());
document.write("<br/>");
document.write(txt.toLowerCase());
</script>
```
SAVE AS: JavaScriptMethods.htm

NARESH TECHNOLOGIES 01. OCT. 2012
naresh technologies

1) Javascript Array Object: Array object is used to store multiple values in a single variable.

★ Creating Javascript Array: Creating an Array slightly diff. from creating a normal variable Bcoz, JavaScript has variables & properties associated with Arrays.

Creating an Array: In JavaScript, we can create an Array in following 3 Ways.

Way1: Regular

```
var myNames = new Array();
myNames[0] = "Ravi";
myNames[1] = "Nilesh";
myNames[2] = "Pintu";
```

Way2: Condensed

```
var myNames = new Array("Ravi", "Nilesh",
                        "Pintu");
```

Way3: Literal

```
var myNames = ["Ravi", "Nilesh", "Pintu"];
```

★ Introduct^n To JavaScript Object:

It is an object based programing.
It allows u to define ur own objects & make ur own variable Tag.
Every object is a special kind of data.
An object has properties & Methods.

Properties: properties r the values associated with an object.

properties : Width
Height
Name
Length

Ex:
```
<script>
var txt = "Naresh i Technology";
document.write(txt.length);
</script>
```

We can create objects as a container for related variables.

Variables of an object are called properties.

To define a JavaScript object we use the new Object() keyword.

Once, we created a new object, we can assign properties to it. These properties behaves exactly like variables.

Ex:

```
<html>
  <head>
    <title> Person Object </title>
    <script>
      person = new Object();
      person.name = "Nilesh";
      person.age = 25;
      person.sex = 'Male';
      person.height = 5.5
document.write (person.name + " is "
              + person.age + " years old, "
              + person.sex + " and "
              + person.height + " Foot Tall.");
    </script>
  </head>
</html>
```

SAVE AS: JavaScriptObject.htm

O/P : Nilesh is 25 years old Male and 5.5 Foot Tall.

There is a keyword associated with an object which is 'WITH'. WITH creates a kind of halfway Referrence. We can drop the repeated referrences to 'person', The person has already made with keyword.

```
<html>
  <head>
    <title> Person Object with WITH </title>
    <script>
person = new Object();
  with (person)
  {
    name = 'Ram';
    age = 30;
    sex = 'male';
  } height = 6;
document.write (name + ' is ' + age +
              years old, ' + sex + ', and '
              + height + ' Foot Tall');
    </script>
  </head>
</html>
```

SAVE AS: JavaScriptObjectWithWITH.htm

O/P : Ram is 30 years old Male and 6 Foot Tall

★ Remember following pts. While Using Arrays.

i) Array is a special type of variable.

ii) Values are stored into an Array by using the Arrayname.

Ex: myArray[2] = "Hello World";

iii) Values in an Array are accessed by the ArrayName & Location of the value.

Ex: myArray[2];

iv) JavaScript has Built-in-Functions for Arrays.

★ Access An Array:

U referr to an element in an Array by referring to the index number.

Ex: var name = myNames[0];

The above stat. modifies the 1st element

myNames[0] = "Pawan";

★ **Note:** ★ Here, [0] is first element & [1] is second Element.

★ Array Methods And Properties: ★

Array object contains predefined properties And Methods.

```
var x = myNames.length   → Property
var y = myNames.indexOf("Raju")
                 ↑ Method
```

Q: Write a Script using length property.

→
```
<html>
  <body>
    <script>
      var i;
      var mynames = new Array();
      mynames[0] = "Nil";
      mynames[1] = "Pin2";
      mynames[2] = "Pawan";
      for(i=0; i<mynames.length; i++)
      {
      document.write(mynames[i] + "<br/>");
      }
    </script>
  </body>
</html>
```

SAVE AS: ArrayMethodsUsingLengthProperty.htm

O/P: Nil
Pin2
Pawan

Q: Write a Script using sort() method.

⇒ In JavaScript, we can sort an Array Alphabetically with the help of sort() method.

```
<html>
  <body>
    <script>
      ~~var myArray();~~
      var x = new Array();
      x[0] = "Football";
      x[1] = "Baseball";
      x[2] = "Cricket";
      x.sort();
      document.write(x[0] + x[1] + x[2]);
    </script></body></html>
```

SAVE AS: ~~[illegible]~~ ArrayMethodsUsingSortMethod.htm

O/P: ~~[illegible]~~ Baseball, Cricket, Football.

★ Arrays with JavaScript Functions

Q: Write a Script to display length of an Array.

⇒
```
<html>
  <head>
    <script>
      function myFunction()
      {
      var fruits = ["Banana", "Orange", "Apple",
                    "Mango"];
      var x = document.getElementById("demo");
      x.innerHTML = fruits.length;
      }
    </script>
  <body>
    <p id="demo">Click the Button to
                 Create an Array length
                 </p>
    <button onClick="myFunction()">
      Length</button>
  </body>
</html>
```

Remember it should not capital otherwise O/P will not Generate.

SAVE AS: LengthOfAnArrayUsingJavaScript.htm

O/P: Click the Button To Create An Array Length

[Length] ← After click O/P will be 4

Q: Write a Script to change an Image when U Clicked on the Next Button.

⇒

Expected O/P

[image]

[Next] ⇒ Button

```
<Script>
arr = new Array
('fish.jpg','smiley.jpg','nature.jpg','Nature3.jpg','Nature5.jpg');
~~'Nature3.jpg','Nature5.jpg');~~
i=0;
```

SAVE AS: ArrayImagesNXtButton.htm

```
Function Fun1()
{
i++
if (i==5)
{
i--;
alert("No more images")
}
else
{
document.getElementById('img1').src="img/"+arr[i];
}
}
</Script>
<img src="img/fish.jpg" width="300" height="250" id="img1">
<br>
<input type="button" value="NEXT" onclick="fun1()">
```

02.OCT.2012

Q: Write a script to display Reverse Array Elements.

⇒

```
<head>
  <script>
    var names=["Nil", "Pin2", "Pawan"];
    function myfunc()
    {
     names.reverse();
     var x=document.getElementById('demo');
     x.innerHTML=names;
    }
  </Script>
</head>
<body>
  ~~<button onclick="myfunc()~~
   <p id='demo'> Click on following button to get Reverse array elements...!</p>
   <button onclick="myFunc()"> Reverse_AElement</button>
</body>
```

Pop:

Using this method, remove last element of an Array

```
<head>
 <script>
   var names = ["zyx", "abc", "aaa"];
   function myfunc()
   {
    names.sort();
    var x = document.getElementById('demo')
   x.innerHTML = names;
   }
 </script>
</head>
 <body>
  <p id = 'demo'> Click on following Buttons to Remove Array Elements </p>
  <button onclick = "myfunc()"> Pop_Elements </button>
  </button>
```

★ Javascript Boolean object:

Boolean object represents boolean values either True or False. The boolean object is used to convert a non boolean value to a boolean value.

Syntax: For creating a boolean Object

```
var myBoolean = new Boolean();
```

Ex:

```
<script>
   var b1 = new Boolean(0);
   var b2 = new Boolean(1);
   document.write("0 is Boolean" + b1 + "<br/>");
   document.write("1 is Boolean" + b2 + "<br/>");
</script>
```

O/P:

0 is Boolean false
1 is Boolean True.

If boolean object has no initial value or the passed value any one of the following

0
-0
null
" "
false
undefined
NaN (Not a Number)

* Boolean object properties:

Properties	Description
1) Constructor	It returns the fn that created the boolean objects prototype.
2) prototype	It allows u to add properties and methods to a boolean object.

Methods:

Method	Description
i) toString()	Converts a boolean value to a string & Returns the Result.
ii) valueOf()	It returns the primitive value of a boolean object.

Q: Write a script using toString() object Method.

→
```
<html>
  <Script>
    function myFunction()
    {
    var myvar = new Boolean(1);
    var x = document.getElementById("demo");
    x.innerHTML = myvar.toString();
    }
  </Script>
  <body>
    <p id="demo"> Click the Button to Display the Button to get String </p>
    <button onclick="myfunction()" > Display</button>
  </body>
</html>
```

* Working with JavaScript Date Object:

Date object is a datatype built-in into the javascript. date objects are created with the new date. Once, a date object is created, no. of methods allows U to operate on it.

Most method simply allows U to get and set the year, month, second, date, hour etc.

Q: Write a script to display current date.

→
```
<head>
  <Script>
    function Display()
    {
     document.getElementById('demo').innerHTML = Date();
    }
  </Script>
</head>
<body>
  <p id='demo'> Click on this button to display date time... !!</p>
  <button onclick="Display()"> Display_Date </button>
</body>
```

The date object we can instantiate in the following 4 ways.

Way1): var x = new Date();

Way2): var x = new Date(milliseconds);

Way3): var x = new Date(dateString);

Way4): var x = new Date(year, month, day, milliseconds);

Date object contains following 7 Arguments.

1) year
2) Month
3) Day
4) Hour
5) Minute
6) seconds
7) Milliseconds.

These 7 arguments are mainly contains date object.

Ex:

```
<script>
  var dt= Date();
  document.write("Date And Time is: "+dt);
</script>
```

Get the javascript Time:

The Date object has been created, That variable holds current Time And Date. To get the Inform^n we need to display using the following f^ns.

1) getTime(): displays no. of Milliseconds.
2) getSeconds(): Displays no. of Seconds.
3) get Minutes(): Displays no. of Minites.
4) getHours(): Displays no. of Hours.
5) getDay(): Displays day of Week. (0-6) 0=Sunday 6= saturday.
6) getMonth(): Number of Month (0-11)
7) getDate(): Displays day of Month (0-31)
8) getFullYear(): The four digit year.

Ex: Write a Script to Get Current Month, Date, Year.

```
<script>
    var currentTime = new Date();
    var month = currentTime.getMonth()+1
    var day = currentTime.getDate()
    var year = currentTime.getFullYear()
    document.write(month + "/" + day + "/" + year)
</script>
```

Ex: Write a Script To Display Get Fullyear. (H.W)

```
<script>
    function DisplayYear()
    {
     var d= new date();
     var x=document.getElementById('demo');
        x.innerHTML = d.getFullYear();
    }
</script>
<body>
  <p id='demo'> click the button to get Full year...!! </p>
  <button onclick = "Display_Year()" > Full Year </button>
</body>
```

★ Set Dates:

Date object supports set_dates, we can easily manipulate the date by using the methods available in the date object

1) SetDate(): Using this method, we can set the Day of Month to the Date object.

Syntax:

Date.setDate(day)

Ex:

Write a script to display the setDate.

⇒

```
<html>
  <script>
    function myFunction()
    {
     var d = new Date();
     d.setDate(15);
     var x = document.getElementById("demo");
     x.innerHTML=d;
    }
  </script>
```

```
<body>
  <p id="demo"> Click the Button to display date after changing day of Month</p>
  <button onclick="myFunction()"> display </button>
<\body>
```

★ Javascript String Object:

03.Oct.2012

~~Script~~ The string object is used to manipulate a stored piece of text. String objects are created with new String().

Syntax:

```
var txt=new String("string");  OR
var txt="string";
```

★ String object Properties:

String object has a few properties.

1) length:

This property returns the length of a string in characters.

Syntax:

```
string.length
```

Ex:

```
<script>
  var str="NareshiTech";
  document.write(str.length);
<\script>
```

~~Methods~~ ★ String object Methods:

String object contains several methods like ... toUpperCase(), toLowerCase(), charAt().

Ex:

```
<script>
  var str="NareshiTech";
  document.write(str.charAt(3));
  document.write("<br/>");
  document.write(str.toUpperCase());
  document.write("<br/>");
  document.write(str.toLowerCase());
<\script>
```

~~<body>~~

~~<p id="demo">, Click the Button to display the string~~

```
<head>
<script>
 function UpperCase()
 {
  var str="Naresh i Technologies";
  document.getElementById('demo').innerHTML=str.toUpperCase();
 }
 function Lowercase()
 {
  var str="Naresh i Technologies";
  document.getElementById('demo1').innerHTML=str.toLowerCase();
 }
 function char_Index()
 {
  var str="Naresh i Technologies";
  document.getElementById('demo2').innerHTML=str.charAt(5);
 }
</script>
</head>
<body>
 <p id='demo'> Click to change UpperCase </p>
 <button onClick="UpperCase()"> UpperCase </Button>
 <p id='demo1'> Click To change to Lowercase </p>
 <button onClick="LowerCase()"> LowerCase </button>
 <p id='demo2'> Click To display Index char </p>
 <button onclick="Char-Index()"> Display_char </button>
</body>
```

SAVE AS:

O/P:

Javascript String Match():

This method searches a string for a match. If No match found, it returns Null.

Syntax:

```
String.match(regexp)
```

Ex:

```
<head>
 <script>
  var str="JavaScript";
  document.write(str.match("scripT"));
 </script>
</head>
```

O/P: Null

★ Javascript String Styles:

Javascript contains all formatted string styles like html

```
<script>
 var txt="Naresh i Tech";
 document.write("<P>Big: "+txt.big()+"</p>");
 document.write("<P>Small: "+txt.small()+"</P>");
 document.write("<P>Bold: "+txt.bold()+"</P>");
 document.write("<P>Italic: "+txt.italics()+"</P>");
```

```
document.write("<P> Fixed: " + txt.fixed() + "</P>");
document.write("<P> Strike: " + txt.strike() + "</P>");
document.write("<P> fontcolor: " + txt.fontcolor("green") + "</P>");
document.write("<P> fontsize: " + txt.fontsize(6) + "</P>");
</script>
```

String Replace:

By using this Method, we can Replace the string.

Ex:

```
<script>
var txt = "visit to Nagpur";
document.write (txt.replace("Nagpur", "chandrapur"));
</script>
```

★ JavaScript Math Object:

It allows U to perform mathematical Tasks. The Math object contains several mathematical constants & Methods.

★ Math object properties.

1) JavaScript PI property:

Ex: Write a script To display PI value.

```
⇒ <head>
    <script>
      function PIValue()
      {
      document.getElementById("demo").innerHTML = Math.PI;
      }
    </script>
  </head>
  <body bgcolor = lightblue>
  <p id='demo'> Click to display PI Approximate value </p>
   button onclick = "PIValue()"> Display_PI </button>
  </body>
```

2) abs

it Returns absolute value of x.

Math.abs(-1)

Q: Write a script to display set of Absolute values when U Click the Button.

⇒
```
<html>
  <script>
   function Myfunction()
   {
    var a = Math.abs(7.25);
    var b = Math.abs(-7.25);
    var c = Math.abs (Null);
    var d = Math.abs ("Hello");
    var e = Math.abs (2+3);
    var x = document.getElementById ("demo");
      x.innerHTML= a+ "<br/>" +b+ "<br/>" +c+ "<br/>" +d+ "<br/>" +e;
   }
  </script>
  <body>
  <p id='demo'> Click the Button to the Absolute Value of different numbers </p>
 <button onclick="myfunction()"> Absolute_value </button>
 </body>
</html.
```

round():

```
<script>
   document.write (Math.round (0.60) + "<br/>");
   document.write (Math.round (-4.40) + "<br/>");
</script>
```

Javascript Number Object:

In Javascript, The number object is an object wrapper for primitive numeric values. Number objects are created with new Number().

Syntax:
```
var num= new Number(value);
```

Number object properties:

Property	Description
MAX_VALUE	It Returns the largest no. possible in Javascript.
MIN_VALUE	It Returns the Smallest no. possible in Javascript.
NAN	Represents "Not-A-Number" value.
prototype	Allows U to add properties & Methods to an object.

Q: Write a script to display Max & Min value in a Javascript.

⇒
```
<script>
  function Max_VALUE()
  {
  document.getElementById('demo').innerHTML=Number.Max_Value;
```

```
function Min_Value()
{
document.getElementById('demo1').innerHTML=Number.Min_Value;
}
</script>
</head>
<body>
<p id='demo'> Click to display Max value in JavaScript </p>
<button onClick="Max_VALUE()"> Max_value </button>
<p id="demo"> Click to display Min value in JavaScript </p>
<button onClick = "Min_VALUE()"> Min_value </button>
</body>
```

04.OCT.2012

★ JavaScript Regular Expression Object:

Q: What is RegExp?

→ It is an object that describes a pattern of characters. When U search in a Text U can use a pattern to describe what U r searching for.

A simple pattern can be a one single character, a more complicated pattern can consist of more characters.

Regular Expressions are used to perform powerful pattern matching & search and replace f^n on Text.

Syntax:

1) var patt = new RegExp(pattern, modifiers);

OR

2) var patt = /pattern/modifiers;

pattern specifies the pattern of an Expression.

Modifier specifies if search should be Global or Case sensitive or Multiline matches etc.

Brackets:

Brackets are used to find a Range of Characters.

The following Table describes Expression And Descript^n.

Expression	Description
[abc]	Find Any character Bet^n the Brackets.
[^abc]	Find Any character Not Bet^n the Brackets.
[0-9]	Find Any digit 0-9.
[A-Z]	Find Any character from Uppercase A to Uppercase Z
[adgk]	Find Any character in the given set.
[A-z]	Find Any character from Uppercase A to Lowercase z
[a-z]	Find any character from Lowercase a to lowercase z.
[^adgk]	Find any character outside the given set.
[red|blue|green]	Find any of the Alternative specified.

Quantifiers :

The following Table describes Quantifiers & Descript^n.

Quantifier	Descript^n
n+	Matches any string that contains atleast one 'n'.
n*	Matches any string that contains '0' or more occurances of 'n'.
n?	Matches any string that contains '0' or '1' occurances of 'n'.
n$	Matches any string with 'n' at the end of the string.
^n	Matches any string that begins with 'n'.

Meta Character :

It is simply an alphabetical character preceded by a backslash (\)

The following Table describes character & Descript^n.

character	Description
. (period)	Matches a Single character
\s	A Whitespace character
\S	Not a Whitespace character.
\d	A Digit Bet^n 0-9.
\D	A Non-digit
\w	A Word character (a-z, A-Z, 0-9, _)
\W	A Nonword character
[\b]	A Literal Backspace.
[aeiou]	Matches a single character in the gvn set.
[^aeiou]	Matches a single character outside the gvn set.

RegExp object Properties :

This object contains the following properties.

Property	Description
Global	Specifies if the "g" modifier is set.
ignoreCase	Specifies if the "i" modifier is set.
Multiline	Specifies if the "m" modifier is set.

JavaScript ignoreCase property :

This property specifies whether 'i' modifier is set. This property Returns True is 'i' modifier is set. Otherwise, It returns False.

Syntax:

RegExpObject.ignoreCase.

Ex:

Q: Write a Script using 'i' modifier, & check whether 'i' modifier is set or not. (14)

⇒
```
<html>
  <body>
    <Script>
      var str = "Goto Naresit!";
      var patt1 = /NIT/i;
      if (patt1.ignoreCase)
      {
       document.write ("'i' modifier is set!");
      }
     else
      {
       document.write ("'i' modifier is Not set!");
      }
    </Script>
  </body>
</html>
```

★ RegExp Methods

Methods	Description
Compile()	Compile a Regular Expression
test()	Test for a match in a string, it returns True or False.
exec()	Test for a match in a string, it returns the first Match.

Ex: Write a Script using Test() Method.

⇒
```
<Script>
    var str = "Hello World!";
    var patt = /Hello/g;
    var result = patt.test (str);
  document.write ("Returned Value: " + Result);
</Script>
```

★ Working With Browser Objects:

This object contains inform^n abt the Browser. It has the following properties.

Property	Description
appCodeName	It Returns the code name of the Browser.
appName	It Returns the Name of the Browser.
appVersion	It Returns the version inform^n of the Browser.
cookieEnabled	It Determines whether cookie are enabled in the Browser.

Ex:
```
<Script>
   document.write ('Navigator info: ' + navigator.AppVersion);
</Script>
```

Q: Write a Script to Display Browser Name.

⇒
```
<script>
    document.write("Browser Name: " + navigator.appName);
</script>
```

★ Navigator Object Methods:

Method	Description
javaEnabled()	Specifies in Browser javascript enable or not.

Ex:
```
<script>
    document.write("Browser Name: " + navigator.javaEnabled());
</script>
```

Screen Object:

This object contains inform^n abt the visitor's screen.

```
<script>
  document.write("screen width is: " + screen.width);
</script>
```

Properties	Description
availHeight	Returns the Height of the Screen. (Excluding the Windows Taskbar)
availWidth	Returns the width of the screen. (Excluding the Windows Taskbar)
height	Returns the total height of the screen.
width	Returns the total width of the screen.

★ History Object:

This object contains the URL visited by the user. This object is path of window object. We can access thr. window.history property.

property	Description
length	Returns the no. of URLs in the history list.

History object Methods:

Method	Description
back()	It loads the previous url in the history list.
forward()	It loads the Next url in the history list.
go()	It loads the specific url in the history list.

★ Length property:

By using this property, we can return URL's in the History list.

(Note: Internet Explorer & Opera starts at '0', Firefox, chrome & Safari starts at '1')

Syntax: history.length

Ex:
```
<body>
    <script type="text/javascript">
    document.write ("Number of URL are:" + history.length);
  </script>
</body>
```

Ex:
```
Function HBack()
{
document.write ("Number of URL
}
```

05. OCT. 2012

Ex:
```
<html>
  <head>
    <script>
      function Back_History()
      {
      window.history.back()
      }
      function Forward_History()
      {
      window.history.forward()
      }
    </script>
  </head>
  <body>
    <button onclick="Back_History()">His_Back</butt
    <button onclick="forward_History()">His_Forward</butt
  </body>
</html>
```

Note: If History is available Then it works only, Otherwise, It's Not Work"

SAVE AS:

O/P:

Ex: Write a script to check Textbox Empty or Not.
```
<html>
  <head>
    <script>
      Function notEmpty()
      {
      var myTextField=document.getElementById('myText');
      if (myTextField.value!=" ")
      {alert("User Entered Text");
      }
      else
      {alert("User Not Entered Text");
      }
    }</script>
  </head>
  <body>
  Enter Any Text: <input type='text' id='myText'>
  <button onclick="notEmpty()">check_form</button>
  </body>
</html>
```

SAVE AS:

O/P:

Enter Any Text: [ ]

Ex:
```
<script type="text/javascript">
  function cal()
  {
  alert(document.getElementById("value1").value + document.getElementById("value2").value)
  }
<script>
<body>
  <form>
    <input type="text" id="value1" size="3" value="20"/>
    + <input type="text" id="value2" size="3" value="100"/>
<input type="button" value="Submit" onClick="cal()"/>
  </form>
<\body>
```

Expected O/P: [ ] + [ ] [Submit]

★ Window object:

(JSON: JavaScript Object Notat^n)

Window object represents an open window in a Browser. If a Document contain frames the Browser creates one window object for the html document. And one Additional window object for each frame.

NOTE: There is No public standard that applies to the window object, All Browser support it.

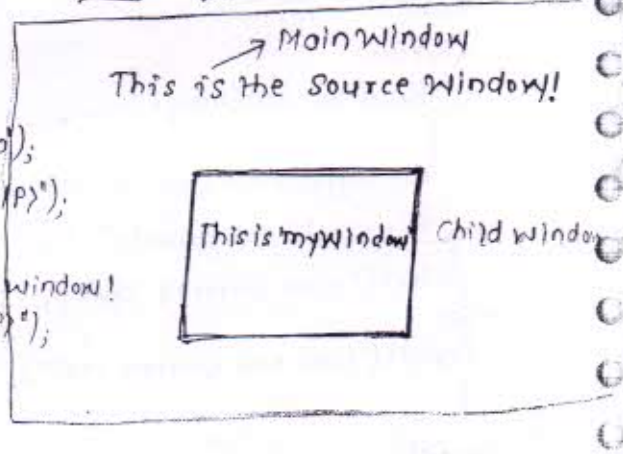
★ Window opener property:

It returns a referrence to the window that created the window. When opening a window with window.open method.

Syntax: window.opener

Ex:
```
<head>
<script>
  function openWin()
  {
  myWindow=window.open("","",'width=200,height=100');
  myWindow.document.write('<P> This is 'my_window' <\P>');
  myWindow.focus();
  myWindow.opener.document.write("<P> This is source window! <\P>");
  }
<\script>
<\head>
<body>
<input type="button" value="Open" onclick="openwin()"/>
<\body>
```

O/P: [Open 'myWindow']



Location Object:

This object contains inform^n abt the current URL. The Locat^n object is the part of Window object & it is accessed thr. the Window.location property.

Location object properties.

Property	Description
hash	Returns the Anchor portion of URL
host	Returns the hostname & portion of URL.
hostname	Returns the hostname of a URL.
href	Returns the Entire URL.
pathname	Returns the pathname of URL.
port	Returns the port number ~~of~~ the server URL.
protocol	Returns the protocol of a URL.
search	Returns the query portion of a URL.

Q: Write a Script To Display URL.

→
```
<script>
  document.write(location.href);
</script>
```

~~Ex:~~
~~<script>~~
~~function~~

* Location object Methods:

Methods	Descript^n
assign()	Loads a New document
reload()	Reload the current document.
replace()	Replaces the current document with a New one.

Ex:
```
<head>
<script>
    function Replace_Location()
    {
    window.location.replace("http://www.nareshit.com")
    }
</script>
</head>
<body>
<button onclick="Replace_Location() ~~[illegible]~~ New_Location </button>
```

★ Javascript Browser Object:

06.08.2013

In Javascript, Navigator object contains all inf. abt visitor's browser name version cookie enabled etc. Using Navigator object, we can do following list of jobs.

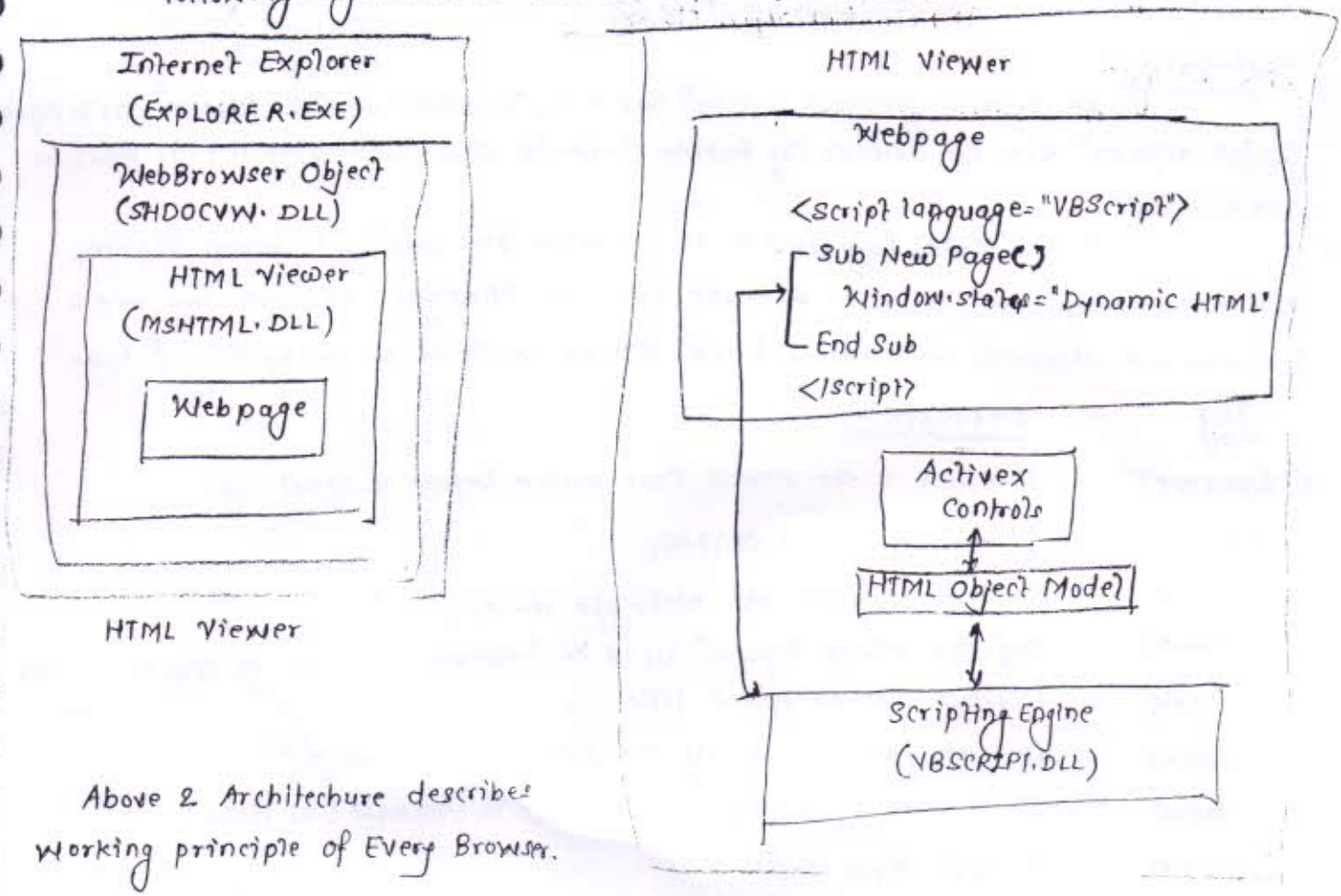
BrowserCodeName : Mozilla.
BrowserName : Netscape
BrowserVersion : 5.0 (Window)
CookiesEnabled : true
Platform : Win32
User_agent header etc.

Ex:
```
<body>
<div id="example"></div>
<script>
txt="<P> Browser CodeName: " + navigator.appCodeName + "</P>"
txt+="<P> BrowserName : " + navigator.appName + "</P>";
txt+="<P> Browser Version: " + navigator.appVersion + "</P>";
txt+="<P> Browser Cookies Enabled: " + navigator.cookieEnabled + "</P>"
txt+="<P> Platform: " + navigator.platform + "</P>";
txt+="<P> User-agent header: " + navigator.userAgent + "</P>";
document.getElementById("example").innerHTML=txt;
</script>
</body>
```

Q: How Javascript Works?

⇒ Browser means collect^n of libraries, objects, controls, Engines, Interpreters etc. Whenever, we r executing any Hypertxt Resources or Style Resources or Script Resources that time browser calling internally that interpreter to execute.

Following dig. describes internal layers of a Browser.



Above 2 Architechure describes working principle of Every Browser.

Executing scripts.

★ Write a script to change when U perform onclick event.

08.OCT.2012

⇒
```
<html>
  <body>
   <script>
    function changeImage()
    {
     element = document.getElementById ('myimage')
     if (element.src.match ("bubon"))
      {
       element.src="pic_bulboff.gif";
      }
    else
      {
       element.src="pic_bulbon.gif";
      }
   }
   </script>
```

<Img id="myimage" onclick="changeImage()" border src="pic_bulboff.gif" width="100" height="180">

<P> Click To Turn ON/OFF the light !! </P>

* Working with DOM objects

↳ Document object Model

1) <Head> Tag:

It contains general Inform^n like meta information, doctype inform^n, style inform^n, Script inform^n etc. The <head> Tag inside elements shouldn't display in body part in webbrowser.

<head>
<meta>
<title>
<link>
<style>
</head>

The <meta> Tag used to the improve the ranking in search Engine, The <style> Tag used to apply more styles on webpages. <script> tag used to develope dynamic webpages. head section contains following list of tags.

Tag	Descriptions
<! DOCTYPE>	Defines a document Type, write before a start Tag
<html>	To indicate html contents
<head>	Defines inform^n abt webpage docs.
<meta>	Defines meta inform^n used to improve rankings in search Engines
<title>	Defines the document title.
<link>	Defines the link using the External style sheet.
<base>	It is used to specify URL relative to webpage URL links.
<style>	To apply style sheets.
<script>	To validate webpage on client_side etc.

Introduction To DOM:

DOM stands for "Document object Model". It W3C std. The DOM defines a standard for accessing html & xml documents.

W3C DOM is a platform & lang. neutral interface that allows programs & Scripts to dynamically access & update the content, structure & style of a document.

W3C DOM std. is seperated into 3 different parts:

1) Core DOM : It is std. Model for Any Structured Document.
2) XML DOM : It is std. Model for XML documents.
3) HTML DOM : It is std. Model for HTML Documents.

1) Core DOM:

The Node object represents a Node in html document, The document is
i) The Document
ii) An Element
iii) An Attribute
iv) Text
v) A Comment.

★ Node object properties:

The following table describes a few Node object properties

Property	Descriptions
1) attributes	Returns a collect^n of a Nodes attributes.
2) baseURI	It Returns an absolute base URI of a Node.
3) childNodes	It Returns a Node_list of a childNodes for a Node.
4) firstChild	It Returns the firstChild of a Node.
5) lastChild	It Returns the lastChild of a Node.

baseURI property:

It returns the URI of the page, that specified node can be a document, element, Attribute or Any otherNode type.

Note: This property supports all other major browsers Except Internet Explorer.

Syntax:
node.baseURI

Q: Write a script to display the URI of current page.

```
<html>
  <script>
    function myFunction()
    {
    var x=document.getElementById("demo");
    x.innerHTML = document.body.baseURI;
    }
  </script>
  <body>
   <p id="demo">
   Click the Button To Display the Base URI </p>
   <button onclick="myFunction()">
   Display_URI </button>
  </body>
</html>
```

XML DOM

XML DOM defines the object and properties of all XML elements, Methods Etc.
XML Stands for Extensible Markup language. It is designed to transport & store to data
It is easy & simple to learn. To learn XML U should have html & javascript fundamentals.

Q: What is XML?

⇒ i) It is a Markup language like html.
ii) It is designed to carry data, Not to display data.
iii) XML Tags are not predefined, U must define Ur own Tag.
iv) XML is designed to be self Descriptive.
v) XML is a W3C Recommendat^n.

Q: Diff. Bet^n HTML And XML.

⇒

HTML	XML
i) W3C Recommendat^n (Standard)	i) W3C Recommendat^n (Standard)
ii) predefined Tags.	ii) User defined Tags.
iii) To design to display data.	iii) Transport & store data.
iv) Every documents starts with <html> Tag	iv) Every documents starts with <ROOT> Tag.
v) No Case Sensitive.	v) 100% Case Sensitive.
vi) We need to follow Tag Hierarchy.	vi) We must follow Tag Hierarchy.
vii) Every open Tag No need to close	vii) Every openened tag must be closed.

★ **XML Syntax Rules:**

These Rules r simple and Logical. All XML elements must have a closing Tag.

Ex: <P> This is a paragraph

} It is Illegal in XML

<P> This is a paragraph </P>

} This is a current format in XML

Ex: <Message> This is incorrect </message>
<message> This is correct </message>

In XML, elements must be properly Nested.
In html, We can see improperly Nested Elements.

Ex: This Text is Bold & Italic

In XML, All Elements must be proper.

Ex: This is Bold & Italic.

* XML documents must have root Element. It is parent of all other Elements.

Ex:
```
<root>
    <child>
        <subchild>.... </subchild>
    </child>
</root>
```

* XML Attribute values must be Quoted.

Ex:
```
<note date = 12/11/2007>
    <to> Ravi </to>
    <from> Raju </from>
</note>
```
Incorrect

09.OCT.2013

* Comments in XML:

These comments are similar to html. Comment Notat^n are

```
<!-- This is a comment-->
```

HTML:
- Font color: Black
- one paragraph
- one space
- Font size: 3
- one line
- Font Family: Times New Roman

By default in HTML

* Whitespace is preserved in XML.
* HTML truncates multiple whitespace characters to one single whitespace.
* XML not Truncated.

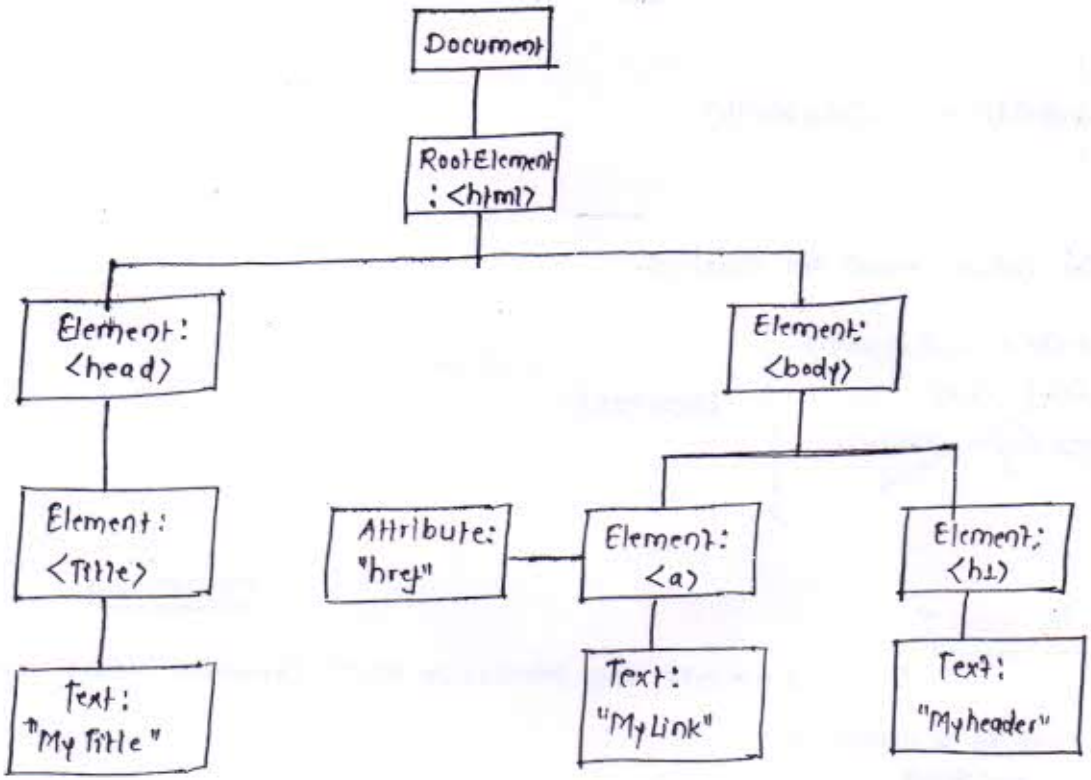
* What is html DOM? **HTML DOM**

⇒ A std. object model for html, A std. programing interface for html & W3C std. html DOM defines the objects & properties of all html elements & the methods to access them. html DOM is a std. for how to get, change, add or delete html elements.

DOM Nodes: According to W3C, html DOM std, Everything in an html document is a Node.

i) One Entire document is a document Node.
ii) Every Html element is an Element Node.
iii) The Text inside html Elements are Text Nodes.
iv) Every HTML attribute is an Attribute Node.
v) Comments are Comment Nodes.

★ HTML DOM Tree Structure



In the above DOM Tree structure, Top Node is called the Root. Every Node has exactly one parent, a Node can have any no. of children. Siblings are Nodes with the same parent.

Ex:
```
<html>
  <head>
    <title> DOM Tutorial </title>
  </head>
  <body>
    <h1> DOM Lesson One </h1>
    <P> Hello World! </P>
  </body>
</html>
```

★ Recognize following pts:

i) <html> Node has No parent (Root Node)
ii) <html> Node has 2 child Nodes (Head & Body)
iii) <head> Node has 1 child Node (Title)
iv) <Title> has 1 child Node (Text Node)
v) <h1> and <P> nodes are siblings.
vi) <head> element is first child of html.
vii) <body> element is Last child of html.
viii) <h1> element is First child of Body.
ix) <P> element is Last child of the Body.

★ html DOM properties:

properties are values of Nodes that U can get or set.

1) innerHTML property:

It is useful for getting or replacing the contents of html elements.

Ex:
```
<Script>
  Function Max_VALUE()
    {
      document.getElementById('demo').innerHTML = Number.Max_Value;
    }
  Function Min_VALUE()
    {
      document.getElementById('demo1').innerHTML = Number.Min_value;
```

* HTML DOM Methods:

Methods r actions u can perform on Nodes. It contains following methods.

i) getElementById
ii) appendChild
iii) removeChild Etc.

Q: Write a Script By using getElementById method

⇒
```
<script>
  <body>
    <p id="intro"> Hello World! </p>
     This Example demonstrates the <b>getElementById</b> Methods.
  </body>
</html>
```

Q: Write a Script To Disable a Button.

⇒
```
<html>
  <script>
    function disableElement()
     {
      document.getElementById ("firstbtn").disabled=true;
     }
  </script>
  <body>
  <form> Buttons: <input type= "button" id ="firstbtn" value="ok">
                  <input type = "button" id ="secondbtn" value= "ok">
  </form>
  <button on_click ="disableElement()"> Disable Button </button>
  </body>
```

O/P:

Buttons: [ok] [ok]

[Disable Button]

Q: Write a Script to get the value of a Button.

⇒
```
<body>
  <input type="button" id = "button1" value= "Cancel!">
  <p> Text on the Button is :
  <script>
    document.write (document.getElementById ("button1").value);
  </script>
 </p>
</body>
```

★ JAVASCRIPT VALIDATIONS (JSvalidations) ★ 10.Oct.2020

Validation : O/P level as per User Requirements

Verification: programmer doing at low level like debugging etc.

ii) If a text f/p is all alphanumeric characters.
iii) If a text has the current no. of characters.
iv) If the selected item frm drop down valid or not.
v) Entered email id valid or not.
vi) User entered fields length valid or Not.
vii) Entered valid date and format Etc.

Q: Write a Script to check the text field Empty or Not. If Empty display Entered Some text, If Entered value display

```
⇒ <script>
   function func()
   {
    if (txt == " ")
    {
     alert ("Empty");
    }
    else
    {
     alert ("document.getElement
                ById ("txt").value");
    }
   </script>
 </head>
 <body>
   Enter text : <input type="text" id="txt" />
 <input type = "button" onclick="notEmpty()"
          value = "Form checker" />.
 </body>
</html>
```

OR

```
<html>
 <head>
  <script>
   function notEmpty()
   {
    var myTextField=document.getElementById("my text");
    if (myTextField.value != " ")
    {
     alert ("You Entered: " + myTextfield.value)
    }
    else
    {
     alert ("Would u please enter some text?");
    }
   }
  </script>
 </head>
```

O/P :

Enter Any Text : [ ] [Fromchecker]

```
<body>
  Enter Any Text:<input type="text" id="my text" />
  <input type="button" onclick="notEmpty()" value="Form Checker" />
</body>
</html>
```

Imp

Write a Script to validate Textfields, when User Entered display silver border, when user moving without Entering display Red Border.

```
<head>
<title> Validating HTML Forms using JS </title>
<Script>
  function funremove (cid)
  {
   if (document.getElementById (cid).value == "Enter Text")
     document.getElementById (cid).value = " "
  }
  function funchklen (len, cid)
   {
    if (len==0)
   }
   document.getElementById (cid).value="Enter Text"
   document.getElementById (cid).style.borderColor="red"
  }
  else
  {
   document.getElementById (cid).style.borderColor="silver"
  }
 }
</Script>
</head>
<body>
 <form name="uservalidations">
 Username: <input type="text" id="txt1" onblur="funchklen (this.value.length,'txt1')"
          value="Enter Text" onclick="funremove ('txt1')"> <br>
 Password: <input type="text" id="txt2" onblur="funchklen (this.value.length,'txt2')"
          onclick="funremove ('txt2')"> <br>
</form>
</body>
```

Username: [ ]

Password : [ ]

Validating Text Fields with a msg when user leaving the object.

like

Username: [Enter Text] Please Fill Field

Password: [Scott123]

```
<head>
  <title>
<script>
  function funremove(cid,spanid)
   {
    if(document.getElementById(cid).value == "Enter Text")
     {
      document.getElementById(cid).value=" "
     }
   }
   function funchklen(len,cid,spanid)
    {
     if (len==0)
     {
      document.getElementById(cid).value="Enter text"
      document.getElementById(cid).style.borderColor="red"
      document.getElementById(spanid).innerHTML="Please Fill Field"
     }
     else
      document.getElementById(cid).style.borderColor="silver"
      document.getElementById(spanid).innerHTML=" "
    }
   </script>
Username: <input type="text" id="txt1" onblur="funchklen(this.value.length,'txt1','sp1')"
        value="Enter Text" onclick="funremove('txt1')"> span id="sp1"></span> <br>
password: <input type="text" id="txt2" onblur="funchklen(this.value.length,'txt2')"
        onclick="funremove('txt2')">
</form>
</body>
```

nareshTech@gmail.com → Top level domain

★ Validating Email id:

Generally Electronic mail id having following 5 parts.

1) Combinat^n of letters, numbers, periods, Hiphens, signs And Underscores

Ex: naresh1_Tech.info4u

2) "@" symbol

3) Combinat^n of Letters, numbers, Hiphens, periods etc. Ex: Way2sms,

4) period

5) Top level domain.

Ex: naresh1Tech @ gmail. com → Top level domain

- naresh1Tech → User optional
- @ → seperator Bet^n User optional & Domain
- gmail → Secondary domain (SCD)
- . → period Seperator Bet^n Secondary & ~~Third~~ Top level domain

google experiments.com

Emailid: [ksraju] Invalid Emailid

Ex:
```
<script>
 function chkemail(val,sid)
 {
 exp=/^([a-z A-Z 0-9_.-])+@([a-zA-Z0-9_.-])+\.([a-zA-Z])+([a-z A-Z])+/;
 if(!exp.test(val))
 {
 document.getElementById(sid).innerHTML="Invalid Email id"
 document.getElementById(sid).style.color="Red"
 }
 else
 {
 document.getElementById(sid).innerHTML=" "
 }
}
</script>
```

Sir ⇒ nithim1javascript@gmail.com

By John Resig

Subject: Evening 6-7

jQuery.com

Book: simplyjavascript 3rd Edition or 4th Edition

JSON ⇒ Javascript Object Notat^n

javascriptkit.com

```
EmailId: <input type="text" id="txtemail" value="example@domain.com" onblur="chkemail
        (this.value,'sp2')"> <span id="sp2"></span>
```

Client side platform: html + css + Js + JQ + xml + JSON + HTML5 + CSS3 + AJAX

OOP'S CONCEPTS

OOP's

php 5.0 is supporting object Oriented programming Concept like Inheritance, Abstract class, Interfaces etc. Ooops in programming lang. to increase the performance of Application, Security and Reuseability.

By using OOps Concept we can develop the applicat^n with specific design pattern like 3-Tier Architecture, N-Tier Architecture, MVC Architecture etc

In OOps Concept, The primary Template is a class, used to maintain the Methods and Variables.

Class:

Class is a collect^n of properties and methods. According to the OOps Concept, Everything we should place within the class.

Methods:

A Function declarat^n within the class is called as Method.

Property:

Variables within the class, we can call as property.

By Defoult, we cannot call the class member directly from the script. If User want to call class member, first we need to create an object to that class

Without a class we write a prog. So that, C++ is not a Fully Object Oriented programing.

By Using a 'class' keyword we can create the class.

By Using 'new' keyword, we can allocate the new memory location to load the class contents.

By defoult, Every class internally contains the defoult constructor. In class name, this constructor is used to load the class content into newly created memory location.

Field:

A variable within the class, we called as Field.

object:

Object is an instance of a class, we can call the class properties with the help of an object.

Progs How To create an Object.

```
<?php
class cls1
{
function funcls()
 {
 echo "From class";
 }
}
function fun1()
 {
 echo "From ext fun";
 }
fun1();
$obj=new cls1();
$obj->funcls();
?>
```

new:

By using this keyword we can allocate new memory allocat^n.

Constructor:

Constructor is a special type of method which contains classname as methodname.

Every class internally contains one constructor. We can call it as default constructor.

This default constructor is used to load the class contents into the RAM memory locat^n.

We can also create the constructor explicitely using classname → 1st way or __(underscore underscore) → 2nd way construct keyword. We can call this constructor as explicit constructor.

Explicit constructors we r using to initialize the variables, to open the DATABASE connections when class is loading.

Constructor with __construct keyword is called as Explicit constructor. (By default it executed)

```
<?php
class cls1
{
function __construct()
 {
 echo "From __cons";
 }
 function cls1()
 {
 echo "cons with class";
 }
}
$obj=new cls1();
?>
```

Destructor: → Not used in web Applicat^n

Destructor is also a special type of f^n executes when the class object is destroying. We can explicitely creates the destructors by using __(underscore underscore) destruct keyword.

By default, in PHP when the script execut^n is completed, the destructor will execute otherwise, by using unset f^n we can destroy the object of class at the middle of script Execut^n.

Destructor we are using to uninitialize variables, to close the db connect^n etc.

```
function __destruct()
{
echo "Dest is executed";
}
```

To declare the variables within the class we should use any access specifier or var datatype.

```
<?php
class clsinfo
{
var $sno=100;
}
$obj=new clsinfo();
echo $obj->sno;
```

★ Inheritance:

DATE: 06.01.2012

It is a concept of accessing the features of one class from another class. If we inherit the class features into another class, we can access Both class properties the object of class to which class we r inheriting the properties. We can extends the features of a class by using 'extends' keyword.

★ Types of Inheritance:

1) Single Inheritance:

In this concept, we have only 2 classes. One is Base class Another is Derived class.

Ex:
```
<? php
    class cls1{
            function fun1()
            {
            echo "from class1";
            }
        }
    Class cls2 extends cls1
        {
            function fun2()
            {
            echo "from class2";
            }
        }
    $obj = new cls2();
    $obj -> fun1();
}>
```

✗ php Not supported
2) Multiple Inheritance:

In this concept, A child class can have 'n' number of parent classes. php doesn't support this concept. (Bcoz, of Ambiguity)

3) Multilevel Inheritance:

In this concept, a class can behaves as Both Base class & Derived class.

4) Hierarchial Inheritance:

In this concept, A Base class can have no. of derived classes

Doesn't supported by php

5) Hybrid Inheritance:

It is a combination of Single, Multiple, Multilevel & Hierarchial Inheritances. php doesn't support this concept.

★ Some Helper Functions in PHP to get the Information About Class And Object.

1) get_class:

By using this, we can get the classname of an object.

Ex:
```
<?php
class cls1 {
}
$obj = new cls1();
echo get_class($obj);
?>
```

2) get_class_vars:

To get the all variables of a class as Array Elements.

Ex:
```
<?php
class cls1 {
    var $x = 100;
    var $y = 200;
}
print_r(get_class_vars("cls1"));
?>
```

3) get_class_methods:

To get the all methods of a class as an Array.

Ex:
```
<?php
class cls1 {
    function fun1() { }
    function fun2() { }
}
print_r(get_class_methods("cls1"));
?>
```

4) get_declare_classes:

To get the all declare classes in current script along with predefined classes.

Ex:
```
<?php
class cls1 {
}
print_r(get_declared_classes());
?>
```

5) get_object_vars:

To get the all variables of an object as an Array.

Ex:
```
<?php
class cls1 {
    var $x = 100;
    var $y = 200;
}
$obj = new cls1();
print_r(get_object_vars($obj));
?>
```

6) class_exists:

To check whether the specified class is existed or Not.

```
<?php
class cls1 {
}
echo class_exists("cls1");
?>
```

7) **is_subclass_of:** By using this f^n, we can check whether the 1st class is subclass of 2nd class or not.

Ex:
```
<?php
class cls1{
}
class cls2 extends cls1
{
}
echo is_subclass_of("cls2", "cls1");
?>
```

8) **method_exists:** By using this f^n, we can check whether the f/p method is existed or not.

Ex:
```
<?php
class cls1{
function fun1()
{
}
}
echo method_exists("cls1","fun1");
?>
```

9) **interface_exist:** To check the f/p interface is existed or Not. (By using this f^n, we can check whether the specified interface is existed in current script or Not)

10) **property_exists:** To check whether the f/p properties are Existed or Not.

Ex:
```
<?php
Class cls1
echo property_exists("cls1","x");
?>
```

★ **$this:** By using $this, we can call the properties of a class with that class methods.

Ex:
```
<?php
error_reporting(E_ALL);
class cls1
{
var $x = 100;
function fun1()
{
$x = 1234;
echo $x;
echo $this->x;
}
}
$obj = new cls1();
$obj->fun1();
?>
```

★ **Scope Resolution Operator (::)**

By using this operator, we can access the class properties with out the help of an object. Sometimes, we need to call the class properties without help of an object. That time we can go for Scope Resol^n Operator.

parent: By using this keyword, we can access the properties of parent class from derived class.

Ex:
```
<?php
class cls1{
function fun1()
{
echo "from Parent Class";
}
}
class cls2 extends cls1{
function fun1()
{
echo "from Derived class";
}
function fun2()
{
$this->fun1();
parent::fun1();
```

```
$obj = new cls2();
$obj -> fun2();
?>
```

Static:

It is a keyword we can directly access, the static class properties with the help of classname. (Without an Object)

Ex:
```
<?php
class cls1
{
static function fun1()
{
echo "hi";
}
}
cls1::fun1();
?>
```

08.OCT.2012

return:

By using this keyword, we can return a value from a F^n

Ex:
```
<?php
class cls1
{
function fun1()
{
echo "From method";
return 100;
}
}
$obj = new cls1();
$x = $obj -> fun1();
echo $x;
?>
```

★ Access Specifiers:

By using access specifiers we can specify the scope of accessibility to the class contents.

By default, we have 3 Types of access specifiers in php. Those are

1) public
2) private
3) protected.

1) public:

Allows the accessability from parent class, derived class, and outside the class.

2) protected:

We can access from Baseclass & derived class.

3) private: These property we can access within the class only where we declared. We cannot access from derived class & outside the class.

	Within class	Derived class	Outside class
public	✓	✓	✓
protected	✓	✓	✗
private	✓	✗	✗

Ex:
```
<?php
class clsBase
{
public $pub = "public var";
protected $pro = "protected var";
private $pri = "private var";
function funBase()
{
echo $this -> pub;
echo $this -> pro;
echo $this -> pri;
}
}
```

```
class clsDer extends clsBase
 {
  function funDer()
   {
    echo $this->pub;
    echo $this->pro;
    echo $this->pri;
   }
 }
$obj=new clsBase();
// $obj->funBase();
// echo $obj->pub;
$obj1=new clsDer();
// $obj1->funDer();
$obj1->funBase();
?>
```

★ **Abstract Classes:**

By using Abstract classes, we can create the Abstract methods as well as Non Abstract Methods within the class.

Abstract classes we r using to provide a structure to the class contents.

Abstract methods are method declarat^n. They don't have implementat^n.

In Abstract class, can't be instantiate means we can't create object for abstract class

By using 'abstract' keyword, we can get the abstract classes.

(★ PHP doesn't support Overloading)

```
<?php
abstract class clsabs
 {
  abstract function fun1();
  abstract function fun2();
  function funx()
   {
    echo 'From Concrete Method';
   }
 }
class cls3 extends clsabs
 {
  function fun1()
   {
    echo "Implementation from der class";
   }
  function fun2(){}
 }
$obj=new cls3();
$obj->fun1();
$obj->funx();
?>
```

★ **Interfaces:**

Interfaces r same as abstract classes But interfaces doesn't allow method def^n. Interfaces can't maintain Non_abstract methods.

By using 'interface' keyword, we can create the interfaces in Applicat^n.

09.Oct.2018

★ Differences Bet^n Abstract classes & Interface

Abstract class	Interface
i) Abstract class comes under partial Abstract^n.	i) Interface comes under fully Abstract^n.
ii) Abstract classes can maintain abstract methods & Non abstract Methods.	ii) Interfaces can maintain only abstract methods.
iii) In Abstract classes, we can create the variables	iii) In Interfaces, we can't create the variables.
iv) In Abstract classes, we can use any access specifier	iv) In Interface, we can use only public access specifier.
v) By using 'extends' keyword we can access the abstract class features from derived class	v) By using 'implement' keyword we can get interface from derived class
vi) Multiple Inheritance is not possible.	vi) By using interfaces multiple inheritance is possible.

2) View:

View represents the presentat^n of application. View object refers to the model remains same if there is any modification's in the Business Logic.

In other words, we can say that it is the responsibility of view to maintain consistency in it's presentat^n & the model changes.

3) Controller:

Whenever, User sends a request for something, it always go thr. controller. A controller is a responsible for intercepting the request from view & passes to the model for appropriate action. After the action has been taken on the data, the controller is responsible for directly the appropriate view to the user. In Graphical User Interfaces, controller & view works very closely together.

15. oct. 2012

Step1:
SAVE AS: controller.php

```
<?php
 include "model.php";
 if (isset ($_REQUEST['mt']))
 {
  $tit = $_REQUEST['mt'];
  $tt = clsmodel::getinfo ($tit);
  include "view2.php";
 }
 else
 {
  $titles = clsmodel::gettitles ();
  include "view.php";
 }
?>
```

Step2: SAVE AS: model.php

```
<?php
 include "conn.php";
 class clsmodel
 {
  static function gettitles()
  {
   __connect();
   $data = mysql_query("select mtitle
                 from tbl_movies");
   $i = 0;
   $arr = array();
 while ($rec = mysql_fetch_row ($data))
  {
   $arr [$i] = $rec [0];
   $i++;
  }
  return $arr;
 }
  static function getinfo ($t)
  {
   __connect();
   $data = mysql_query ("select * from
   tbl_movies where mtitle = '$t'");
   $rec = mysql_fetch_row ($data);
   return $rec;
  }
 }
?>
```

Step3: SAVE AS: conn.php

```
<?php
 Function __connect()
  {
   mysql_connect ("localhost","root", " ");
   mysql_select_db ("test");
  }
?>
```

Step4: SAVE AS: View.php

```
<table border = '2'>
<?php
 for ($i=0 ; $i<count ($titles); $i++)
  {
   echo "<tr><td><a href = 'controller.php?mt = $titles [$i]'> $titles [$i] </tr>";
  }
?>
```

Step6: Save As: View2.php

```
<?php
$str=<<<abc;
<h2 style='color:red'> Titles: $rv[0]
<h2 style='color:green'> Director: $rv[1]
<h2 style='color:blue'> Casting: $rv[2]
<h2 style='color:Black'> Report: $rv[3]
    abc;
    echo $str;
?>
```

★

★ **DOM Document:**

By using this class we can create XML file. Also we can read the elements of XML file. XML is used to x'ferr the Data Bet^n the Heterogeneous platforms. It stands for "Extensible Markup Language".

★ **Difference Bet^n the HTML And XML:**

HTML	XML
1) HTML is used for User Interface Design.	i) XML is used for x'ferring the data.
2) HTML is a Case Insensitive.	ii) XML is a Case Sensitive.
3) HTML doesn't required <RootElement>	iii) XML requires <RootElement>.
4) No need to close opened <Html> Tags.	iv) Every Tag U must closed.
5) HTML contains predefined Tags.	v) XML contains User defined Tags.

RootElement:

Ex:

```
<RootElement>
    <Child> XYZ </Child>
    <Child>
        <Child>
            ABC
        </Child>
    </Child>
</RootElement>
```

Functions

Functions available in DOM document class.

Create Element():
By using this f^n, we can create new Element.

append_child():
To append an Element as child element.

Create Text Node()
To create the Text Content to an Element.

GET ELEMENTS BY TAGNAME():
Get the Elements to specify the Tag.

NodeValue():
By using this property we can ~~change~~ get the content of an Element.

load():
To load the XML document to the current applicat^n.

Savexml():
To save the xml element.

Ex:
```
<?php
 $doc=new domdocument();
 $root = $doc->Createelement("myfile");
 $doc->appendchild($root);
 $sn = $doc->Createelement("sno");
 $root->appendChild($sn);
 $txt = $doc->CreateTextNode("100");
 $sn->appendChild($txt);
 $add = $doc->CreateElement("Address");
 $root->appendChild($add);
 $city = $doc->CreateElement("city");
 $add->appendChild($city);
 $txt = $doc->CreateTextNode("Hyd");
 $city->appendChild($txt);
 $cont = $doc->Savexml();
 file_put_contents("myfile.xml", $cont);
 echo "XML File is Created";
?>
```

Ex:
```
<?php
 if (isset($_POST['sub']))
  {
   $tname = $_post['txttab'];
   mysql_connect("localhost", "root", " ");
   mysql_select_db("test");
   $data = mysql_query("select * from $tname);
   $doc = new domdocument();
   $root = $doc->createelement($tname);
   $doc->appendchild($root);
   $fc = mysql_num_fields($data);
   while ($rec = mysql_fetch_row($data))
   {
    $row = $doc->Createelement("Row");
    $root->appendchild($row);
    for ($i=0; $i<$fc; $i++)
    {
     $fname = mysql_field_name($data, $i);
     $child = $doc->Createelement($fname);
     $row->appendchild($child);
     $txt = $doc->Createtextnode($rec[$i]);
     $child->appendchild($txt);
    }
   }
   $cont = $doc->Savexml();
   file_put_contents("$tname.xml", $cont);
   echo "File is Created";
  }
?>
<Form method="post" action=" ">
Table Name: <input type='text' name='txttab'>
<br>
<input type="Submit" value="Click" name="Sub">
</Form>
```

Ex:

```
<?php
$doc = new domdocument();
$doc->load ("tbl_parent.xml");
$rows = $doc->getelementsbytagname ("Row");
foreach ($rows as $row)
{
$pid = $row->getelementsbytagname ("pid");
$txt = $pid->item(0)->nodeValue;
echo $txt;
$pname = $row->getelementsbytagname("pname");
$txt1 = $pname->item(0)->nodeValue;
echo $txt1;
echo "<br>";
}
?>
```

★ Exception Handling :→ 17.Oct.2012

Exception is a base class in Object Oriented programing library used to handle the runtime Errors.

Errors are mainly divided into 3 Types.

1) Compilation Error
2) Runtime Error (Exception)
3) Logical Error.

We can rectify compilat^n Error & Logical Error at the time of applicat^n implement^n But Runtime Errors we cannot Rectify.

Runtime Errors will occur based on the Input values given by the user at the time of applicat^n Execut^n. To get Runtime Error, the program Execut^n will stop without any Intimat^n.

To prevent the applicat^n Execut^n from Undetermined Terminat^n, we r handling Exception.

Exception class is providing 2 constructors

1) A constructor with Single Argument
2) A Constructor with double (Two) Argument

★ Methods of Exception class

1) getMessage():
By using this method, we can get the Error msg.

2) getLine():
To get the Error line number.

3) getCode():
To get the Error Code.

4) getFile():
To get the Filename where the Except^n is occured.

★ try():

By using try block we can maintain the executable statements.

★ Catch():

If there is any Except^n occured in the try block, the stat. of catch block will execute.

throw:

By using this keyword, we can throw the Error Msg into the Excepti^n class.

Ex:

```
<?php
if (isset ($_POST['sub']))
{
$fname= $_POST['txtfile'];
try
  {
  $fp=fopen($fname,"r");
  if (!$fp)
  {
  throw new Exception("file is not Available");
  }
  else
  echo "file is Opened";
  }
  catch(Exception $e)
  {
  $str="Error occured(ie)". $e->getmessage().' in line no "$e->getline()." in prog".
  $e->getfile();
  echo "<span style='color:red'> $str </span>";
}
  try
  {
  $db= mysql_connect("localhost","root"," ");
  if(!$db)
  {
  throw new Exception (mysql_error());
  }
  else
  echo "DB is connected";
  }
  catch (Exception $e)
  {
  $str = $e->getmessage();
  echo "<span style='color:red'> $str </span>";
  }
}
?>
<form method="post" action=" ">
Filename: <input name="txtfile"> <br> <input type="submit" name="sub" value="Get">
```

that is

Ex:: <?php

```
include "myexception.php";
try {
   $con = mysql_connect ("localhost", "root", " ");
   if (!$con)
   throw new myexception ("1001");
   else
   echo "connected";
}
catch (myexception $e)
{
echo $e->getmymessage();
}
?>
```

save: myexception.php

```
<?php
class myexception extends Exception
{
function myexception ($ecode)
{
$this->ec = $ecode;
}
function getmymessage ()
{
$fp = fopen ("error.txt","r");
while ($line = fgets ($fp))
{
list ($em, $ec) = explode (":", $line);
$ec = trim ($ec);
if ($this->$ec == sec)
return $em;
}
}
}
```

save: error.php

Database not connected : 1001

File is not Opened : 1002

FTP Error : 1003

29.Oct.2012

★ Object copy :

It is a concept of copying one object into another object. means we can referr a single address location with multiple reference variables.

```
<?php
class clsinfo
{
var $sno=100;
}
$obj=new clsinfo();
echo $obj->sno;
$obj1 = $obj;
echo $obj1 -> sno;
$obj1 -> x = 500;
echo $obj -> x;
?>
```

★ object cloning :

It is a concept of creating a duplicate object with the existing object. By using clone() fn, we can create the duplicate object of an existing object.

```
<?php
class clsinfo
{
var $sno=100;
}
$obj=new clsinfo()
echo $obj->sno;
$obj1 = clone ($obj);
$obj->sno = 500;
echo $obj1 -> sno;
?>
```

18.OCT.2013

AJAX

AJAX

AJAX

AJAX

18.Oct.2012 (177)

i) AJAX stands for 'Asynchronous Javascript And XML'.

ii) AJAX we r using to send the Request to the server without Submitting the Webpage.

iii) By Using AJAX we can increase the performance of Applicat^n instead of Submitting the Entire Webpage to the server. We have sending the requested data decreases the N/W Traffic.

iv) Without submitting the Webpage, we can Execute server side script that's why User can see the Webpage continuously & also he can perform another task mean while.

v) Instead of submitting the Webpage, we can sending rtd. data with decreases the Bandwidth consumpt^n of Webserver.

★ **Asynchronous:**

By using Asynchronous process, we can send multiple Request to the server with irrespective of previous Responses.

★ **JavaScript:**

By Using JavaScript, Document Object Model (DOM), we can create AJAX Object And we can send that object to the server.

★ **XML:**

Data x'ferrs Bet^n Browser And Server in the form of XML.

★ **STEPS TO CREATE AJAX OBJECT AND SEND THAT OBJECT TO THE SERVER**

i) Find out the Browser Type and create object to that Browser class.

ii) By using 'navigator.appname' we can find the Browser Type. If The Browser is Internet Explorer we can create object for Activex object class otherwise, we can create object for XML http request class.

iii) Load the object with some values like method (get/post), Serverside Script name synchronized / Asynchronized process.

iv) Send that object to the server to execute Server_side script.

★ **Object Methods**

1) **Open():**

By using this method, we can open the object by specifying serverside scriptname, method, synchronized / Asynchronized process.

2) **Send():**

Send the object to the server.

3) **Abort():**

3) **abort():** To stop the Execut^n of AJAX object.

★ **Object Properties:**

1) **Ready state:** By using this property we can get the state of AJAX object. It contains 5 different states.

i) If state is "0", object initialized.
ii) If state is "1", Object Loading.
iii) If state is "2", ~~loaded~~ object loaded.
iv) If state is "3" Interactive.
v) If state is "4" completed successfully.

2) **Response state:** By using this property, we can get the Text content what is available in AJAX object.

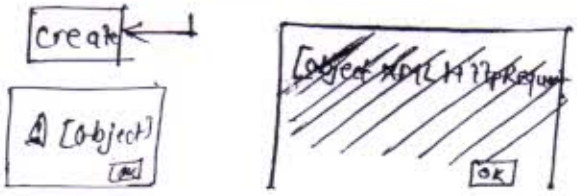
3) **onreadystatechange:** This Event will fire for every state change of AJAX object.

Ex: Prog. To create An AJAX object.

```
<script>
function fun1()
{
if(navigator.appName=="Microsoft Internet Explorer")
Obj=new ActiveXObject("Msxml2.XML HTTP");
else
Obj=new XMLHttpRequest();
alert(obj)
}
</script>
<input type="button" value="Create" onclick="fun1()">
```

SAVE AS: CreateAjaxObject.htm

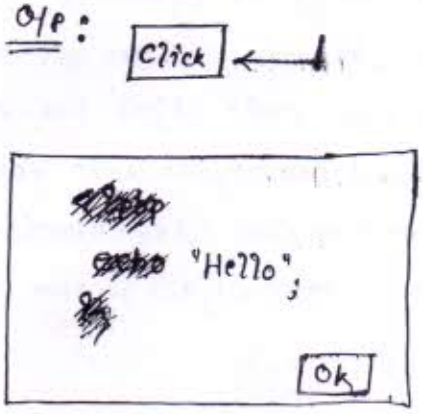
O/P:




```
<script>
function fun1()
{
if(navigator.appName=="Microsoft Internet Explorer")
Obj=new ActiveXObject("Msxml2.XML HTTP")
else
Obj=new XMLHttpRequest();
obj.open("get","Hello.php",true)
obj.onreadystatechange=fun2
obj.send()
}
function fun2()
{
if(obj.readyState==4)
{
alert(obj.responseText)
}
}
</script>
<input type="button" value="Click" onclick="fun1()">
```

SAVE AS: Ajax1.html

O/P:



Ex:

SAVE AS: Ajax2.htm

```
<script>
 function fun1 (pname)
 {
  if (navigator.appName=="Microsoft Internet Explorer")
 obj=new ActiveXObject("Msxml2.XML HTTP")
else
 obj=new XMLHttpRequest();
 obj.open("get","pname",true)
 obj.onreadystatechange=fun2
 obj.send()
 }
 function fun2()
 {
  if (obj.readyState==4)
  {
  alert(obj.responseText)
  }
 }
</script>
<input type="button" value="page1" onclick="fun1('Page1.php')">
<Input type="button" value="page2" onclick="fun1('page2.php')">
```

☆ Program of changing the Cricket Score

Ex: Within 2seconds.

PART: I

SAVE AS: CricketScoreRead.htm

```
<script>
 setInterval("fun1()",2000)
 function fun1()
 {
  if (window.ActiveXObject)
   {
    obj=new ActiveXObject("Msxml2.XML HTTP");
   }
 else
   obj=new XMLHttpRequest();
   obj.open("post","CricketScoreRead.php",true);
   obj.send();
   obj.onreadystatechange=function()
   {
    if(obj.readyState==4)
    {document.getElementById('div1').innerHTML=obj.responseText
    }
   }
 }
</script>
<body onload="fun1()">
 <div id="div1" style="font-size:30"></div>
</body>
```

PART: II SAVE AS: CricketScoreRead.php

```
<php
mysql_connect("localhost","root"," ");
mysql_select_db("test");
$data = mysql_query("select * from tbl_score");
$rec = mysql_fetch_row($data);
$str <<<abc
<h2 style='color:red'> Runs : $rec[0]<br>
<h2 style='color:green'> Wickets: $rec[1]<br>
<h2 style='color:blue'> Overs : $rec[2]
abc;
echo $str;
}>
```

O/P:

Runs: 245

Overs: 46

Wickets: 8

Note: In this prog, we also create a database of name tbl_score with 4 columns i.e. id, Runs, Overs, Wickets.

Ex: The important thing is, if we change in database like Runs, Wickets or Over, it automatically changed in Browser.

Program of Checking Username is Available OR Not.

get.php

```
<? php
qs=$_SERVER['QUERY_STRING'];
mysql_connect("localhost","root"," ");
mysql_select_db(test);
$data = mysql_query("select * from tbl_user where uname='$qs'");
echo
mysql_num_rows($data);
}>
```

ajax.html

```
<script>
function fun1(val)
{
if(val.length!=0)
{
document.getElementById('sp1').innerHTML="<img src=loading.gif Width=20>"
obj= new XMLHttpRequest();
obj.open("post","get.php?" +val,true)
obj.send();
obj.onreadystatechange= fun2
}
}
function fun2()
{
if(obj.readyState==4)
{
if (obj.responseText==1)
{
document.getElementById('sp1').style.color="red"
```

```
document.getElementById('sp1').innerHTML = "Username is Not Available"
}
else
{
document.getElementById('sp1').style.color="green"
document.getElementById('sp1').innerHTML="Username is Available"
}
}
}
</Script>
Username : <input id='txtuname' onblur='fun1(this.value)'>
<Span id="sp1"> </span>
```

★ A program to Create a dropdown list of Countries with states, with the help of Mysql connectivity.

⇒

```
<Script>
function fun1(val)
{
obj=new XMLHttpRequest();
obj.open("POST", "getst.php?"+val, true)
obj.send()
obj.onreadystatechange=function()
{
if(obj.readyState==4)
{
document.getElementById('sp1').innerHTML=obj.responseText
}
}
}
</Script>
Countries : <select id="drpcon" onchange="fun1(this.value)">
<?php
mysql_connect("localhost", "root", " ");
mysql_select_db("test");
$data=mysql_query("select * from tbl_country");
echo
"<option> ---Select--- </option>";
while ($rec=mysql_fetch_row($data))
{
echo "<option value=$rec[0]> $rec[1] </option>";
}
?>
</Select>
<Span id="sp1"> </Span>
```

getcit.php

```
<?php
$qs= $_SERVER['QUERY_STRING'];
mysql_connect("localhost","root", " ");
mysql_select_db("test");
$data= mysql_query("select sname from tbl_state where cid=$qs");
echo "states: <select>";
while ($rec =mysql_fetch_row($data))
 {
 echo "<option> $rec[0] </option>";
 }
 echo "</select>";
?>
```

Prog: How To Access Gmail Account Username With DB 22.OCT.2012

```
<script>
  function funover(t)
  {
   t.style.backgroundColor="lightblue"
  }
  function funout(t)
  {
   t.style.backgroundColor="white"
   t.style.color="black"
  }
</script>
<style>
  li
  {
   list-style:none;
   padding-top:10;
   padding-left:0;
   font-size:25;
   cursor:pointer;
  }
  #div1
  {
   border:1px solid silver;
   padding-left:0;
  }
</style>
<script>
  function funinsert(un)
  {
   document.getElementById('txt1').value=un
   document.getElementById('div1').style.visibility="hidden"
  }
  function fun1(val)
  {
   if(val.length!=0)
   document.getElementById('div1').style.visibility=
                                        "visible"
  else
   document.getElementById('div1').style.visibility="hidden"
   if(navigator.appName=="Microsoft Internet Explorer")
   obj=new ActiveXObject("MsXml2.XMLHTTP");
   else
    obj=new XMLHttpRequest();
    obj.open("post","get.php?"+val,true)
    obj.send();
    obj.onreadystatechange=fun2
   }
   function fun2()
   {
    if (obj.readyState==4)
    {
     document.getElementById('div1').innerHTML=
          obj.responseText
    }
   }
</script>
<table cellpadding="0" cellspacing="0">
<tr><td> Username </td>
<input type="text" id="txt1" style="font-size:
      onkeyup="fun1(this.value)">
<tr><td><td><div id="div1">Hi</div>
  </td></tr>
```

get.php

```
<?php
mysql_connect("localhost", "root", " ");
mysql_select_db("test");
$qs = $_SERVER['QUERY_STRING'];
$data= mysql_query("select * from emp where ename like '$qs%' ");
while ($rec = mysql_fetch_row ($data))
 {
  echo "<li onclick = funinsert('$rec[1]')
  onmouseover = 'funover(this)'
  onmouseout = 'funout (this)'> $rec [1]</li>";
 }
?>
```

*

```
<script>
  function fundel(id)
  {
   obj1=new XMLHttpRequest()
   obj1.open ("post","del.php?"+id,true)
   obj1.send (null)
   obj1.onreadystatechange=function()
   {
    if (obj1.readyState==4)
   getrec();
  }
}
 function getrec()
 {
  obj=new XMLHttpRequest()
  obj.open ("post","get.php", true)
  obj.send()
  obj.onreadystatechange= function()
  {
   if (obj.readystate=4)
   {
   document.getElementById('div1').innerHTML=obj.responseText
   }
  }
}
</script>
<body onload = "getrec()">
<div id ="div1">
</div>
</body>
```

get.pho

```
<?php
 mysql_connect("localhost","root"," ");
 mysql_select_db ("test");

 while ($rec=mysql_fetch_row($data))
 {
  echo "<tr><td> $rec[0] <td> $rec [1] <td> $rec [2] <td><img src='b_drop.png'
        onclick = fundel ($rec[0])>";
  }
?>
</table>
```

del.php

```
<?php
$qs=$_SERVER['QUERY_STRING'];
mysql_connect("localhost","root"," ");
mysql_select_db("test");
mysql_query("delete from emp where empid=$qs");
?>
```

* Chat Application :

25.OCT.2012

```
<script src="jquery-1.3.js"> </script>
<script>
  function funimage(r)
 {
  str="<img src="+r.src+' width=20>' $ ('#spbig').html (str)
 }
  function funenter( )
  {
   uname= $ ("#txtuser ").val() document.getElementById ('divname').style.visibility
                                                                    =hidden"
   document.getElementById('lab1').style.visibility= "visible";
  }
  setInterval ('read()", 2000)
  function read()
  {
   $.post ("read.php", { }, function (data)
   {
    $("#div1").html (data) })
   }
 </script>
```

```
<script>
 function funsend()
 {
  imgstr=$("#spbig").html() tval=$("#txtmsg").val() tval="<b>" +uname+ "</b>
   <i> says:</i>" +tval
  tval=tval+imgstr $("#spbig").html(" ")
  $.post("write.php", {qs:tval}, function()
  {
   $("#txtmsg").val(" ")})
 }
</script>
<body onload="read()" bgcolor="lightblue" >
 <table align="center" style="visibility: hidden" id="tab1">
 <tr><td> <div id="div1" style="width:200; height:150; overflow:auto; background-colo
   : white"> <td> <span id="spsmall">
  <span id="spsmall">
  <img src=wink.gif onclick='funimage(this)'>
  <img src=smile.gif onclick='funimage(this)'>
  <img src=tongue.gif onclick='funimage(this)'>
 </span>
 <br>
 <span id="spbig">
 </span>
 <tr><td><input type="text" id="txtmsg"> <input type="button" value="SEND"
      onclick="funsend()">
 </div>
</table>
<div id="divname" style="top:100; left:100; position: absolute">
 Username: <input id='txtuser'> <br> <input type="button" value="Enter" onclick="funenter
                                                                                  ()">
 </div>
</body>
```

read.php

```
<?php
 echo file_get_contents("chat.txt")
?>
```

write.php

```
<?php
$txt=$_REQUEST['qs'];
$txt.="<br>";
$str=file_get_contents("chat.txt");
$fp=fopen("chat.txt","w");
$txt.=$str;
fwrite($fp,$txt);
```