..:: JavaScript ::..

❑ Client Side Scripting Language:

- The Scripts which are executed only by browsers without connecting the server is known as "Client Side Scripting". It is browser dependent.
- Code is visible to user & not secure.
- Execute at client machine, so it takes too much less time.
- It can be disabled by end user.
- Browser can respond immediately to users when key pressed, mouse movements, click etc.
- For e.g.: Check input validity before sending data to server.

- **Examples when we use script?**
  - Complementary form pre-processing.
  - To get data about user's screen/browser.
  - Online games.

- **Examples:**
  - Javascript
  - VBscript

- **Advantages:**
  - Scripts executed only by the browser without connecting the server
  - It takes too less time to execute the script code.
  - It can be disabled by end user.
  - Browser can respond immediately when key pressed, mouse movements, clicks etc.

- **Disadvantages:**
  - Client-side scripts are browser dependent.
  - Scripting code is visible to end user, so it cannot be secure.

❑ Server Side Scripting Language:

- Scripts executed by the web server & processed by the server are called Server side scripting.
- Scripts code executes at server.
- It takes too much time because it required sending request to the server for execution.
- It is not browser dependent.
- It can't be disabled by end user.
- They can have access to files & databases that would not normally be available to visitor.
- It is more secure.

- Examples when we use script?
  - Password Protection
  - Browser Customization
  - Form Processing
  - Building & Displaying Pages created & from DB.
- **Examples:**
  - PHP,ASP,JSP
- **Advantages**
  - It can't be disabled by end user.
  - It is not browser dependent.
  - They can have access to files & databases that would not normally be available to visitor.
- **Disadvantage:**
  - It takes too much time.

❑ Some Useful Terms:

1. Web Server:

- Computers which stores WebPages in form of directories & files, provides these files to be read are called web servers.
- Web server runs some special software like IIS and Apache.
- **Functionality of Web Servers:**
  - Web site management.
  - Accept client request for information.
  - Response to a client request by providing the page with request information.

2. Web Client/Web Browser:

- To access information stored in the format of text, graphics, animation in the web pages user must connect with the web server.
- Computers that offer the facility to read information stored in the web pages are called web client.
- Web client run special App. Software called web browser, which allows the following functionality.
- **Functionality of web Client:**
  - Connect to the appropriate Server
  - Query the server for information to be read.
  - Provide an interface to read the information return by the server.

3. Web Site:

- The Collection of Web pages, a Graphics and Multimedia object is known as web Site.
- A Web site is a related collection of World Wide Web (WWW) files that includes a beginning file called a home page.

- A very large Web site may be spread over a number of servers in different geographic locations.
- IBM is a good example; its Web site consists of thousands of files spread out over many servers in world-wide locations.

4. Web Page:

- The page which is developed by using a web language is known as Web page.
- A web page or webpage is a document or resource of information that is suitable for the World Wide Web and can be accessed through a web browser and displayed on a computer screen.
- This information is usually in HTML format, and may provide navigation to other web pages via hypertext links.
- Web pages may be retrieved from a local computer or from a remote web server. The web server may restrict access only to a private network, e.g. a corporate intranet, or it may publish pages on the World Wide Web. Web pages are requested and served from web servers using Hypertext Transfer Protocol (HTTP).
- Web pages may consist of files of static text stored within the web server's file system (static web pages), or the web server may construct the HTML for each web page when it is requested by a browser (dynamic web pages).
- The World Wide Web Consortium (W3C) and Web Accessibility Initiative (WAI) recommend that all WebPages should be designed with all of these options in mind.

❑ History of JavaScript:-

- Origins of JavaScript…..
- Originally developed by Netscape, as Live Script
- Became a joint venture of Netscape and Sun in Dec.,1995, renamed JavaScript
- Now standardized by the European Computer Manufacturers Association as ECMA-262 Edition 3.
- We'll call collections of JavaScript code scripts, not programs
- JavaScript is client side scripting language
  - This means that JavaScript code is written into an HTML page.
- What is scripting language?
  - Scripting language is just a lightweight programming language.

❑ What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages.
- The web pages developed in HTML contains certain limitations that, they can't be embedded with multimedia components, such as audio-video, graphics etc. To remove these limitations, we are using JavaScript in HTML.
- JavaScript is a client-side scripting language.

- A scripting language is a lightweight programming language.
- JavaScript is usually embedded directly into HTML pages.
- JavaScript is an interpreted language (means that scripts. execute without preliminary compilation). Everyone can use JavaScript without purchasing a license. i.e. It's an open source.

❑ Difference Between JAVASCRIPT AND JAVA:

JAVASCRIPT	JAVA
* Javascript is Scripting Language.	* Java is programming language.
* It supports runtime system.	* It supports compile-time system.
* No distinction between types of Objects.	* Objects are divided into classes and instances.
* Added to any object dynamically.	* Classes and instances cannot have properties and methods added dynamically.
* No need to declare all variables, classes and methods.	* Its needed.
* Javascript is not pure OOP.	* Java is pure OOP.

❑ Features Of JavaScript :

JavaScript is an Interpreted Language –

- JavaScript does not require any preliminary compilation.
- JavaScript's syntax is completely interpreted by Browser.

JavaScript easily embedded with HTML –

- JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages.

JavaScript can put dynamic text into an HTML page –

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

JavaScript can handle Events –

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

JavaScript can be used to validate data –

- This is the most striking feature of JavaScript i.e. it can be used to validate form data before it is submitted to a server.
- This saves the server from extra processing. This is called client-side validation.

JavaScript can be used to detect the visitor's browser –

- A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser.

JavaScript can be used to create cookies –

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

❑ How to insert JavaScript in the page?

- To insert a JavaScript into an HTML page, we use the <script> tag. Inside the <script> tag we use the type attribute to define the scripting language.

```
1) <html>
   <body>
          <script type="text/javascript">
           ...
          </script>
 </body>
 </html>

2) <html>
   <body>
          <script language="javascript">
           ...
          </script>
</body>
</html>
```

Example:-

```
<html>
<body>
        <script type="text/javascript">
                document.write("<h1>Hello  World!</h1>");
         </script>
</body>
</html>
```

Where to put Javascript?

1) You can add JavaScript in <Head> section
2) You can add JavaScript in <Body>section
3) You can add JavaScript in both <Head> and <Body> section

NOTE: You can also add External JavaScript File having extension ".js".

JavaScript in <Head> Section.

- Scripts to be executed when they are called, or when an event is triggered, go in the head section.
- Scripts in head section will be executed immediately while page loads into browser.
- If you place a script in the head section, you will ensure that the *script is loaded before anyone uses it.*

```
<html>
<head>
    <script type="text/javascript">
     function message()
    {
     alert("This alert box was called with the onload event");
     }
    </script>
</head>
<body onload="message()">
</body>
</html>
```

Javascript in <Body> Section.

- Scripts to be executed when the page loads go in the body section.
- If you place a script in the body section, it generates the content of a page.

```
<html>
<head>
</head>
<body>
        <script type="text/javascript">
         document.write("This message is written by JavaScript");
         </script>
</body>
</html>
```

Javascript in both <Head> and <Body>

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

```
<html>
<head>
        <script type="text/javascript">
          ....
        </script>
</head>
<body>
        <script type="text/javascript">
         ....
        </script>
</body>
```

❑ Comments in Javascript:

- The Unexecutable part of any program is known as "Comment".
- Comments are generally used to give instruction about the Program, and only useful for programmer.
- Comments can be added to explain the JavaScript, or to make the code more readable.

//This is single line comment.

/* This is Multi

line comments */

❑ Some important things to know when scripting with javascript.

- **Javascript is case sensitive :**

A function named "myfunction" is not same as "myFunction" and a variable named "my Var" is not same as "myvar".

- **Symbols :**

Opening symbols, like ({ [" ' must always have a matching closing symbols, like ' "] }).

- **White space :**

JavaScript ignores extra spaces. You can add white space to your script to make it more readable. The following lines equivalent:

Name="Anil" or Name=" Anil"

- **Insert special characters:**

  You can also insert special characters like ("': &) with a backslash.
  For eg: document.write ("You \& I sing \"Happy Birthday\".");
  Output: You & I sing "Happy Birthday".

❑ Variables

- **What is variable?**
  - A variable is a name given to the memory location.
  - A variable is a "container" for information you want to store.
  - During the execution of the program the value of the variable will be changed.

- **JavaScript Variables**
  - JavaScript variables are used to hold values or expressions.
  - A variable can have a short name, like x, or a more descriptive name, like car_name.

- **Rules for JavaScript variable names:**
  - Variable names are case sensitive (y and Y are two different variables)
  - Variable names must begin with a letter or the underscore character.

- **Declaring (Creating) JavaScript Variables**
  - You can create a variable with the "var" statement.
    var variable-name = some value

- **Example :**
  - var x = 10 (now variable x holds value 10)

❑ Data Types & Literals

- The predefined identification of any variable is known as Data type. Or we can say that the data type defines the characteristics of variables.
- JavaScript can't support data types. But it supports Literals.
- Literals are used to represent values of variables of particular data type in JavaScript.
- These are fixed values not variables that you literally provide in your script.

Integer Literals:

- Integer Literals can be expressed as Whole Numbers without any decimal points.
- It can be positive or negative number.
- Example :
  - var a = 112;
  - var b = - 44;

Floating Point Literals:

- Floating Point Literals can be expressed as numbers with decimal points.
- It can be positive or negative number.
- Example :
    - var a = 1.23;
    - var b = -4.9

Boolean Literals:

- Boolean Literals can be expressed as only with two values i.e. TRUE or FALSE.
- You can just assign true or false values, it will automatically take 0 as true value and 1 as false value.
- Example :
    - var a = true;
    - var b = false;

String Literals:

- String Literals can be expressed as a sequence of characters. It always enclose with either single quote(') or double quotes(").
- Example :
    - var str1 = "Hello"
    - str2 = 'World'

Array Literals:

- Array Literals can be expressed as a list of values that are called elements of Array.
- The values may be numbers, strings or any type of Literal.
- Example :
    - var arr = new Array(3)
    - arr[0] = "one"
    - arr[1] = "two"
    - arr[2] = "three"

Object Literals:

- Object Literals can be expressed as a list of pairs of property names and associated values of an Object enclosed in { }.
- NOTE: It is just a User Defined Object which is shown later on.

NULL Literals:

- Null Literal consists of single value, which identifies a NULL i.e. empty.
- It is used to set initial value of variable that is different from other valid values.
- Example:
    - var temp = null;
    - document. write("Value of temp is :"+temp);

❑ Dialog Boxes

- "Dialog Box" is nothing but a small separate window that contents provided by end user.
- In JavaScript, Dialog Boxes are very useful because they allow the script to interact with visitors where the visitor must provide a response to Dialog Box before script can continue processing.
- All Dialog Boxes in Javascript are Modal Dialog Boxes and don't permit anything else to happen until Dialog Box receives a response from the user.
- Javascript provides the ability to get user input or display small amount of text to the user by using Dialog Boxes.
- There are three types of Dialog Boxes provided by Javascript :
    - Alert Dialog Box.
    - Confirm Dialog Box.
    - Prompt Dialog Box.

ALERT Dialog Box:

- Alert Dialog Box is use to display some information on web browser window.
- Alert Dialog Box is used to give warning messages to end users.
- The alert Dialog Box displays string passed to the alert () method as well as OK button.
- The JavaScript and HTML, in which this code is held, will not continue processing until OK button pressed.

When we used? :

- A message to display when user inputs invalid information.
- An invalid result from Calculation.
- A warning that a service is not available.
- **Syntax :**

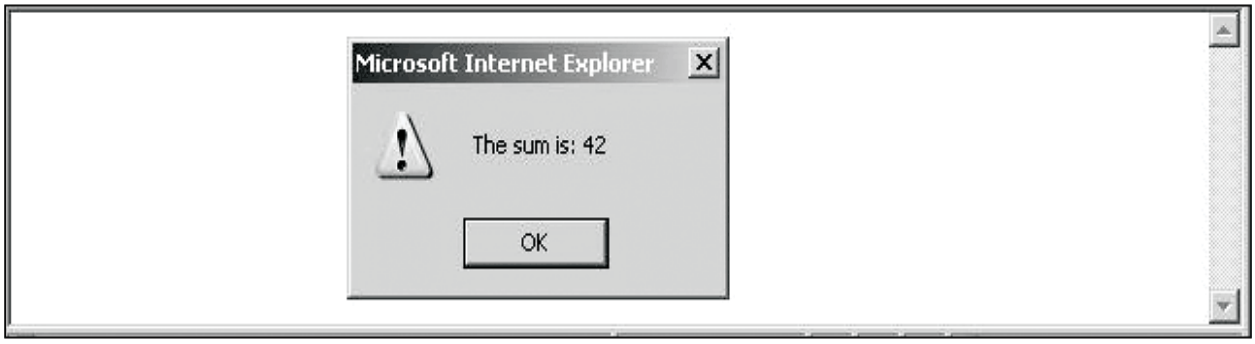
```
alert("message")
```

- **Exemple :**

```
<html>
<body>
    <script language="javascript">
        var a=17,b=25
        var c = a+b
        alert("The Sum is :"+c)
    </script>

</body>
</html>
```

- **Output:**



CONFIRM Dialog Box :

- A Confirm Dialog Box is often used if we want the user to verify or accept something.
- A Confirm Dialog Box display predefined message with OK and CANCEL button.
- When a confirm Dialog Box pop up, the user will have to click either OK or CANCEL to proceed.
- If you press the OK button then it will return TRUE if you press CANCEL button then it will return FALSE.

When we used?

- A message will display when we want any confirmation from end user.
- **Syntax:**
  - **Confirm ("Message")**
- **Example:**

```
<html>
<body>
    <script language="javascript">
        confirm(" Do you want to continue this download?");
    </script></body></html>
```

- **Output:**

PROMPT Dialog Box:

- When you want to get the value from user at that time, the prompt dialog box is used.
- Prompt dialog box will display with a pre-defined message, a text box for user input and with two command buttons.(OK,CANCEL)
- When a prompt dialog box pops up, the user will have to click either OK or CANCEL button to proceed after entering input value.
- If the user clicks OK, then the value that user entered will stored inside the variable. If the user clicks CANCEL,"NULL" will stored inside the variable.

WHEN we used?

- In online games, ask for end user to enter name.
- **Syntax:**

  Var name=prompt("Message",default value)
- **Example:**

```
<html>
    <body>
        <script language="javascript">
            nm=prompt("What is your name?")
        </script>
    </body>
</html>
```

- **Output:**

❑ Decision Statements

- Some times when you write code, you want to perform different actions for different decisions, at that time you can use conditional statements in your code.
- JavaScript provides a complete range of basic programming constructs. They are as given below:

If statement	Use this statement if you want to execute some code only if a specified condition is true.
If …else statement	Use this statement if you want to execute some code if the true and another code if the condition is false.
If…else if …else statement	Use this statement if you want to select one or many blocks of code to be executed.
Switch statement	Use this statement if you want to select one of many blocks of code to be executed.

If statement:

- You can use the if statement if you want to execute some code only when the condition is true.
- The block of statements is executed only when the specified condition is true.
- **Syntax:**

```
If (Condition)
{
        Block of statements
}
```

- If the condition is evaluated to true than the block of statements will be executed, then control will transfer out of block.
- If the condition is evaluated to false than the block of code will be skipped and the control will transfer out of block by skipping the block of statements.
- **Example:**

```
<script language="Javascript">
        var a=10,b=20;
        if(a>b)
        {
                document.write("A is big");
        }
</script>
```

If… else statement:

- The if else construct is an extentation of simple if statement.
- If…else construct is used to execute some code when the condition is true or false.
- You can handle the true block or false block with the help of if… else construct.
- **Syntax:**

```
If(condition)
{
        code for true block
}
else
{
        code for false block
}
```

- If the condition is evaluated to true then the code for true block will be executed.
- If the condition is evaluated to false then by skipping the true block the control will transfer in else part and the code for false block will be executed.
- **Example:**

```
<script language="Javascript">
        var a=20,b=10;
        if (a>b)
        {
                document.write("A  is Big");
        }
        else
        {
                document.write("B  is Big");
        }
</script>
```

if...else if...else statement:

- You can use the if…else if…else statement if you want to select one of many sets of lines to execute.
- This structure gives you facility to specify the condition with each else statement.
- **Syntax:**

```
if(condition1)
{
        code for block 1
}
else if (condition 2)
{
        code for block 2
}
else
{       code to be executed if condition1 and condition2 are not  true.
}
```

- In this structure the condition 1 will be evaluated first, if the condition 1 is evaluated to true then the code for block 1 will be execute.
- If the condition 1 is evaluated to true then and then the control proceeds to the condition 2.
- As soon as the true condition is found then the respected block of code will be executed and then control will go out of the whole if..else if…else block.
- If all the conditions are evaluated to false then the final else block will be executed.

Switch statement

- You should use the switch statement if you want to select one of many blocks of code to be executed.
- First we have a single expression that is evaluated once.

- The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed.
- Use break to prevent the code from running into the next case automatically.
- If expression is not match with any case then default block is executed.
- **Syntax:**

```
Switch(n)
{
    case 1:
        code block 1
        break
    case 2:
        code block 2
        break
    default:
        code to be executed if n is different from case 1 and 2
}
```

- **Example:**

```
<script language="Javascript">
    ch=1;
    switch(ch)
    {
        case '1':
            c=a+b;
            document.write("Addition is"+c);
        case '2':
            c=a-b;
            document.write("Subtraction is"+c);
        case '3':
            c=a*b;
            document.write("Mulitplication is"+c);
        case '4':
            c=a/b;
            document.write("Division is"+c);
        default:
            document.write("Invalid choice");
    }
</script>
```

❑ Looping Structure in Java script:

What is Loop?

- When you want to perform a particular task no. of times at that time the concepts of loop is used.
- JavaScript supports 3 types of looping structure they are as given below:
    - While()
    - Do…while()
    - For()

while() Statement:

- The while() statement is the simplest one from all among the looping structure.
- The while statement is used to execute the statements when the condition is evaluated to true.
- This is also known as an "Entry Controlled Looping Statement" because the condition is evaluated first and if the condition is evaluated to true then & then the body of the loop will be executed.
- **Syntax:**

```
While (Condition)
{
        Body of the Loop
}
```

- **Example:**

```
<script language="Javascript">
        var cnt=1;
        while(cnt<=10)
        {
                document.write("<br>"+cnt);
                cnt++ ;
        }
</script>
```

do...while() statement:

- The do…while() statement is used to execute the statements no. of times as same as while() statement.
- But the difference is that the location of condition. This is the reason that's why it is called "Exit control Loop statement".
- **Syntax:**

```
do
{
        Body of the Loop
}
While (Condition)
```

- **Example:**

```
<script language="Javascript">
        var cnt=1;
        do
        {
                document.write("<br>"+cnt);
                cnt += 2;
        }
        while(cnt<=20)
</script>
```

for() statement:

- The for() statement is another "Entry Controlled Loop" statement.
- It is used when you know in advanced how many times you want to iterate the loop.
- It allows to specify intitialization, testing and increment/decrement, just within a single line of statement.
- **Syntax:**

```
For (initialization;condition;incre/decre )
{
        body of the loop
}
```

- **Example:**

```
<script language="javascript">
        for(i=0;i<=5;i++)
        {
                document.write("Atmiya <br>")
        }
</script>
```

For...In Statement

- The for...in statement loops through the properties of an object.
- **Syntax**

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

- Note: The code in the body of the for...in loop is executed once for each property.
- **Example**

```
var person={fname:"John",lname:"Doe",age:25};
var x;

for (x in person)
{
    document.write(person[x] + " ");
}
```

❑ Array in Java script:

- An Array is an ordered set of values that we refers with its name and index.
- Javascript does not have any explicit array data type.
- However, we can use the predefined Array Object and its methods to work with arrays in our Application.
- The array object has so many methods for manipulating arrayas in various ways.
- The array object is used to store set of values in single variable name.
- The array Literals are also Array Object.

Creating an Array :

- We can create array object in either of three ways:
- **Syntax:**
  - arrayObjectName = new Array()
  - arrayObjectName = new Array(arrayLength)
  - arrayObjectName = new Array(Element 0, Element 1 ,..,Element N)

- arrayObjectName : It is either the name of a new Object or existing Object.
- Element 0,Element 1 : It is a list of values for array elements.
- arrayLength : It is an initial length of array.
- **Example :**
  - var emp = new Array()
    emp[0] = "Samir"
    emp[1] = "Sanjiv"
    emp[2] = "Kumar"

Populating an Array:

- We can populate an array by assigning values to its elements.
- The values which we can assign to its elements may be string and numbers.
- **Example :**
  - var myArray = new Array("AAA",12,"bbb")

Accessing an Array:

- We can refer to a particular element in an array by referring to the name of array and index number.
- **Example:**

```
document.write("<br>First Employee Name : "+emp[0])
document.write("<br>Second Employee Name : "+emp[1])
document.write("<br>Third Employee Name : "+emp[2])
```

- **Output:**

```
Samir
Sanjiv
Kumar
```

Modifying values in existing Array:

- To modify a value in an existing array, just add a new value to the array with specified Index Number.
- **Example:**
  - emp[0] = "Anil"
  - document.write(emp[0]) // prints "Anil" instead of "Samir"

❑ User Define Functions in Java script:

- **What is Function?**
  - "Function" is self contain block of statements that perform a particular task.
  - Function provides the ability to combine no. of statements in single unit.
- Javascript provides Functions through which you can perform certain task.
- Javascript allows users to create our own Functions, are called "User Defined Functions".
- User Defined Function first need to be declared after that function can be invoked by calling it using its name when it was declared.
- The data needed to execute a function is passed as a Parameter.
- **Syntax:**

```
function function-name([parameter list])
{
        block of statements
        [return statement]
}
```

  - Functions are declared and created using "**function"** keyword.
  - "**function-name**" is the appropriate name of the function.
  - **parameter-list**" are optional and passed to the function appears in parentheses and commas separate member of list.
  - "**return**" statement can be used to return any valid expression that evaluates a single value.
- **Example:**

```
<html>
   <head>
      <script language="javacript">
            function displayMsg( )
            {
                  alert("Welcom to Digital City…");
            }
      </script>
    </head>
    <body>
       <script language="javascript">
            displayMsg()
       </script>
    </body>
</html>
```

- **Note:** A function code will be executed only by
  - When user fires any event.
  - When a call to that function (anywhere within a page).
- **Scope of Variables in Functions:**
  - The variable declared or assigned outside the function, is called "global variable".
  - The variable declared or assigned inside the function,is called "local variable".

- Example:

```
<script language="javascript">
        var count = 10;     // global variable
        function scope()
        {
                var i = 20; // local variable
                document.write(i);      //  20
                document.write(count); // 10
        }
        document.write(i);          //undefined.
        document.write(count);    //10
</script>
```

- **Passing parameters and Return values:**
  - We can pass no. of parameters inside the function and we can return values also.
  - Example:

```
<html>
   <head>
        <script>
                function scope(a,b)
                {
                        c = a+b;
                        return c ;
                }
        </script>
    </head>
    <body>
        <script>
                c = scope(10,20)
                document.write("Sum :"+c)
        </script>
</body></html>
```

- **FORM OF UDF**

function functionname() { function body; do *this;* *}*	Function functionname(argument1,argument2) { function body; do *this;* *}*
function functionname(argument1,argument2) { function body; do *this;* *return value;* *}*	function functionname() { function body; do *this;* *return value;* *}*

- **TYPES OF FUNCTION**
  - Simple function [without argument and without return]
  - Function with arguments
  - Function with arguments with return
  - Function without arguments with return

- **Simple functi on [without argument and without return]**

```
<html>
   <head>
      <script language="javascript">
            var name=""
            function hello()
            {
                  name = prompt("Enter your name")
                  alert("Welcome " + name)
            }
            function bye()
            {
                  alert("Good bye " + name)
            }
            hello()
            bye()
      </script>
   </head>
</html>
```

- **Function with arguments**

```
<html>
   <head>
      <script language="javascript">
            var name=""
            function hello(name)
            {
                  alert("Welcome " + name)
            }
            function bye()
            {
                  alert("Good bye " + name)
            }
            hello("World")
            name="Friends"
            bye(name)
      </script>
   </head>
</html>
```

- **Function with arguments with return**

```
<html>
    <head>
        <script language="javascript">
                var name=""
                function hello(usrname)
                {
                        return usrname+" pooja"
                }
                name=hello("world")
                document.write(name)
        </script>
    </head>
</html>
```

- **Function without arguments with return**

```
<html>
   <head>
        <script language="javascript">
                var name=""
                function hello()
                {
                        return "pooja"
                }
                name=hello()
                document.write(name)
        </script>
    </head> </html>
```

❑ Built-in Functions

eval() :

- The eval() function evaluates a string without reference to a particular objects.
- **Syntax:**
  - eval(string-expression)
- **Example:**

```
<script language=”javascript”>
    var no1 = prompt(“Enter First Value”);
    var no2 = prompt(“Enter Second Value”);
    sum = eval(no1) + eval(no2);
    document.write(“Sum : ”+sum);
</script>
```

parseInt():

- The parseInt() function parses a string and returns an integer.

- **Syntax:**
  - parseInt(string)
- **Example:**

```
<script language=”javascript">
    document.write(parseInt("15") + "<br />");
    document.write(parseInt("15.34") + "<br />");
    document.write(parseInt("25 35 45") + "<br />");
    document.write(parseInt(" 100 ") + "<br />");
    document.write(parseInt("30 years") + "<br />");
    document.write(parseInt("Hello") + "<br />");
    document.write(parseInt("15",10)+ "<br />");
    document.write(parseInt("015")+ "<br />");
    document.write(parseInt("15",8)+ "<br />");
    document.write(parseInt("0x10")+ "<br />");
    document.write(parseInt("10",16)+ "<br />");
</script>
```

- **Output:**

```
15
15
25
100
30
NaN
15
8
8
16
16
```

parseFloat():

- The parseFloat() function parses a string and returns a floating point number.
- This function determines if the first character in the specified string is a number. If it is, it parses the string until it reaches the end of the number, and returns the number as a number, not as a string.

- **Syntax:**
  - parseFloat(string);

- **Example:**

```
<script type="text/javascript">
    document.write(parseFloat("15") + "<br />");
    document.write(parseFloat("15.46") + "<br />");
    document.write(parseFloat("35 45 55") + "<br />");
    document.write(parseFloat(" 100 ") + "<br />");
    document.write(parseFloat("30 years") + "<br />");
    document.write(parseFloat("Hello") + "<br />");
</script>
```

- **Output:**

```
15
15.46
35
100
30
NaN
```

❑ Javascript Objects

What is Object?

- "Object" is just a special kind of data. Each Object has its own state and behaviour.
- State of the Object refers as Property.
- Behaviour of the Object refers as Methods.

Properties:

- Properties are used to define the characteristics of an Object.
- Properties are used to access the data values present in the object.
- To access the property of an Object:-
  - **ObjectName.propertyName**

Methods:

- Methods are the actions that can be performed on Objects.
- Object's behaviour, we can change by using accessing their methods.
- To access the method of an Object:-
  - **ObjectName.methodName**

- Javascript is object-oriented scripting language. So Javascript allows you to use with objects.
- Javascript provides two types of Objects

  - **Built-In Objects :** Javascript provides two types of Built-In Objects.
    - Native Objects
      - String Object
      - Math Object
      - Array Object
      - Date Object
    - Browser Objects
      - Window Object
      - History Object
      - Location Object
      - Screen Object
      - Navigator Object
      - Document Object

Built-In Objects :

➢ String Object:

- This object is used to manipulate the String.
- **Property:**

Length	Returns the no. of characters in a String.

- **Methods:**

  - bold:
  - italics:
  - strike:
  - sup:
  - sub:
  - fontcolor:
  - fontsize:
  - link:
  - blink:
  - charAt:
  - concat:
  - indexOf:
  - lastIndexOf:
  - match:
  - replace:
  - search:
  - slice:
  - substr:
  - substring:
  - toLowercase:
  - toUppercase:

❑ bold:

- This method is used to display a string in a bold font.
- Syntax:
  Stringobject.bold()

❑ Italics:

- This method is used to display a string in italics font.
- Syntax:
  Stringobject.italics()

❑ Big:

- This method is used to display a string in big font.
- Syntax:
  Stringobject.big()

❑ Small:

- This method is used to display a string in small font.
- Syntax:
  Stringobject.small()

❑ Sub:

- This method is used to give subscript effect to the string.
- Syntax:
  Stringobject.sub()

❑ Sup:

- This method is used to give superscript effect to the string
- Syntax:
  Stringobject.sup()

- **Fontsize:**
  - This method is used to specify the size of font.
  - Syntax:
    Stringobject.fontsize(size)

- **Fontcolor:**
  - This method is used to specify the color of font.
  - Syntax:
    Stringobject.fontcolor(color)

  - You can specify the color in 3 ways:
  - By specifying the color code.
  - By specifying the color name.
  - By using the rgb() function as an argument.

- **Blink**
  - This method is used to display a blinking string.
  - Can't support by the internet explorer.
  - Syntax:
    Stringobject.blink()

- **Link:**
  - This method is used to display a string as a hyperlink.
  - Syntax:
    Stringobject.link()

- **Strike:**
  - This method is used to give strikethrough effects to the string.
  - Syntax:
    Stringobject.strike()

  - **Example:**

```
<script language="Javascript">
      var myString = "Good Morning"
      var myString2 = " Happy World"
      var str="st";
      var str1="2";
      document.write("<b>My String is</b>: "+myString);
      document.write("<b>Bold</b> : "+myString.bold());
      document.write("<b>Italics</b> : "+myString.italics());
      document.write("<b>Strike</b>: "+myString.strike());
      document.write("<b>Superscript</b> : 1"+str.sup());
      document.write("<b>Subscript</b>H : "+str1.sub()+"O");
      document.write("<b>FontColor</b>:"+myString.fontcolor("red"));
      document.write("<b>FontColor</b>"+myString.fontcolor("rgb
(0,255,0)") );
```

```
        document.write("<b>FontColor</b>:String.fontcolor("0000FF");
        document.write("<b>FontSize</b> : "+myString.fontsize(15));
        document.write("<b>Link</b>"+myString.link("javascript") );
</script>
```

- **CharAt:**
  - The charAt () method is used to find the character at specified position.
  - This method is case sensitive.
  - Syntax:
    Stringobject.charAt(index)
  - Example:
    var str=”My string”
    document. write(charAt(3));
  - Output:
    S

- **Concat:**
  - This method is used to concat (merge) two or more strings.
  - Syntax:
    Stringobject.concat (stringX,stringX,……X)
  - Example:
    var st1=”This is“
    var st2=”Computer”
    document.write(st1.concat(st2))
  - Output:
    This is computer

- **indexOf:**
  - This method returns the position of the first occurrence of a specified string value in a string object.
  - It is case sensitive.
  - Returns -1 if no match found.
  - Syntax:
    Stringobject.indexOf(stringvalue,startIndex)
    Here the string value is required and start index is optional.
  - Example:

```
var str = "original equipment manufacturer";
var s = "equip is at position " + str.indexOf("equip");
s += "<br />";
s += "abc is at position " + str.indexOf("abc");
document.write(s);
```

  - Output:
    - equip is at position 9
    - abc is at position -1

- **lastIndexOf:**
  - This method returns position of the last occurrence of a specified string value.
  - It is case sensitive.
  - It returns -1 if no match found.
  - Syntax:
    Stringobject.lastIndexOf(stringValue,startIndex)
  - Example:
    var str="Hello World"
    document.write(str.lastIndexOf("o"))
  - Output:
    7

- **Match:**
  - It searches for specified string value in a string.
  - It is similar to indexOf() and lastIndexOf(), but returns specified string, instead of position of the string.
  - Case sensitive.
  - Returns null if no match found.
  - **Syntax:**
    stringObject.match(search value)
  - **Example:**
    Str="Have a nice Day";
    document.write(str.match("Have"))
    document.write(str.match("Nice"))
  - **Output:**
    Have
    Null

- **Replace:**
  - The replace() method is used to replace some characters with some other characters in a string.
  - The replace() method is case sensitive.
  - Syntax:
    stringObject.replace (findstring, newstring)
  - Example:
    var str="Play Cricket";
    document.write (str.repalce ("Cricket"," Chess"))
  - Output:
    Play Chess

- **Search:**
  - The search() method is used to search a string for specified value.
  - The search() method returns the position of the specified value in the. If no match was found it returns -1.
  - This method is case sensitive.

- Syntax:
  StringObject.search(searchvalue)
- Example:
  var str="Have a nice day"
  document.write (str.search ("day"))
- Output:
  12

- **Slice:**
  - The slice() method extracts a part of a string and returns the extracted part in a new string.
  - You can use negative numbers to select from the end of the string.
  - If the end is not specified, slice() selects all characters from the specified start position and to the end of the string.
  - Syntax:
    stringObject.slice(start,end)
  - Example:
    var str="Hello, How are you?"
    document.write(str.slice(6));
    document.write(str.slice(6,12));
  - Output:
    How are you?
    How a

- **Substr:**
  - The substr() method extracts a specified number of characters in a string.
  - To extracts from the end of the string, use a negative start number. The start index starts at 0.
  - If the length parameter is omitted, this method extracts to the end of the string.
  - Syntax:
    stringObject.substr(start,length)
  - Example:
    var str="Good Morning";
    document.write(str.substr(3));
    document.write(str.substr(3,7));
  - Output:
    d Morning
    d Morni

- **Substring:**
  - The substring () method extracts the characters in a string between two specified indices.
  - To extract the character from the end of the string, use negative start number. The start index starts at 0.
  - If the stop parameter is omitted, this method extracts to the end of the string.

- Syntax:
  stringObject.substring(start,stop)
- Example:
  var str="Hello world";
  document.write(str.substr(3))
  document.write(str.substr(3,7))
- Output:
  lo world
  lo w

- **toLowerCase:**
  - The toLowerCase() method is used to display a string in lowercase letters.
  - **Syntax:**
    stringObject.toLowerCase()
  - **Example:**
    var str="Hello World!"
  - **Output:**
    hello world!

- **toUpperCase:**
  - The toUpperCase() method is used to display a string in up-percase letters.
  - **Syntax:**
    stringObject.toUpperCase()
  - **Example:**
    var str="Hello World!"
  - **Output:**
    HELLO WORLD!

STRING OBJECT PROPERTY:

- **length:**
  - This property is used to count the length of the specified string.
  - **Syntax:**
    stringObject.length
  - **Example:**
    var str="Hello World";
  - **Output:**
    11

Math Object:

- Math Object provides standard library of mathematical constants and functions.
- The "Constants" are defined as properties of Math Object.
- The "Functions" are defined as methods of Math Object.

- **Methods:**

❑ **abs:**
- It returns absolute value of a number.
- Syntax:
  Math.abs(number)
- Example:
  document.write(Math.abs(-7.25));
  document.write(Math.abs(7.25));

❑ **ceil:**
- It returns value of number rounded UPWARDS to the nearest integer.
- Syntax:
  Math.ceil(number)
- Example:
  document.write(Math.ceil(0.60))
  document.write(Math.ceil(0.40))
  document.write(Math.ceil(- 5.1))
  document.write(Math.ceil(5.1))

❑ **floor:**
- It returns the value of number rounded DOWNWARDS to the nearest integer.
- Syntax:
  Math.floor(number)
- Example:
  document.write(Math.floor (0.60))
  document.write(Math. floor (0.40))
  document.write(Math. floor (- 5.1))
  document.write(Math. floor (5.1))

❑ **cos:**
- It returns the value of cosine of number.
- It returns numeric value between -1 to 1.
- Syntax:
  Math.cos (number)
- Example:
  document.write(Math.cos (3))

❑ **sin:**
- It returns the value of sine of a number.
- It returns the numeric value between -1 to 1.
- Syntax:
  Math.sin(number)
- Example:
  document.write(Math.sin (3))

- **tan:**
  - It returns the number that represents tangent of an angle.
  - Syntax:
    Math.tan(number)
  - Example:
    document.write(Math.tan (1))
- **log:**
  - It returns the natural log (base E) of number.
  - If the number is negative(-), NaN is returned.
  - Syntax:
    Math.log(number)
  - Example:
    document.write(Math.log(1))
- **pow:**
  - It returns the value of X to power of Y.
  - Syntax:
    Math.pow(x,y)
  - Example:
    document.write(Math.pow(2,3));
    document.write(Math.pow(5,2));
- **random:**
  - It returns a random number between 0 to 1.
  - Syntax:
    Math.random()
  - Example:
    document.write(Math.random());
- **max:**
  - It returns the number with highest value of two specified numbers.
  - Syntax:
    Math.max(x,y)
  - Example:
    document.write(Math.max(10,15));
- **min:**
  - It returns the number with lowest value of two specified numbers.
  - Syntax:
    Math.min(x,y)
  - Example:
    document.write(Math.min(10,15));
- **round:**
  - It rounds the number to nearest integer.
  - Syntax:
    Math.round(number)
  - Example:
    document.write(Math.round(0.60));
    document.write(Math.round(0.49));

- **sqrt:**
  - It returns the square root of a number.
  - Syntax:
    Math.sqrt(number)
  - Example:
    document.write(Math.sqrt(16));

Array Object

- An array is the collection of elements having similar datatypes or we can say that array provides the facility to store multiple values in single variable name.
- **Property:**

- **length:**
  - It returns the length of the Array.
  - Syantax:
    ArrayObject.length
  - Example:

```
<script  languagae="javacript">
        var subArr = new Array()
        arr[0] = "PHP"
        arr[1] = "C"
        arr[2] = "VB"
        document.write(subArr.length)
</script>
```

  - Output:
    3

- **Methods:**

- **concat():**
  - It is used to join two or more Arrays.
  - This method does not change the original arrays.
  - It only returns a copy of joined arrays.
  - Syntax:
    ArrayObject.concat()
  - Example:

```
<script languagae="javacript">
    var subArr = new Array()
    subArr[0] = "PHP"
    subArr [1] = "C"
    subArr [2] = "VB"

    var moreArr = new Array()
    moreArr[0] = "Java"
    moreArr[1] = "Oracle"

    document.write(subArr.concat(moreArr))
    document.write(subArr.length)
</script>
```

- Output:
  PHP,C,VB,Java,Oracle
  3

- **join():**
  - It is used to put all the elements of array into string.
  - The elements will be seprated by Seprator, comma(,) is default if you omitted the parameter.
  - Syntax:
    ArrayObject.join(seprator)
  - Example:

```
<script languagae="javacript">
    var subArr = new Array()
    arr[0] = "PHP"
    arr[1] = "C"
    arr[2] = "VB"
    document.write(subArr.join(":"))
</script>
```

  - Output:
    PHP: C :VB

- **pop():**
  - It is used to remove and return "Last Element" of Array.
  - This method changes the length of original Array.
  - Syntax:
    ArrayObject.pop()
  - Example:

```
<script  languagae="javacript">
    var subArr = new Array()
    arr[0] = "PHP"
    arr[1] = "C"
    arr[2] = "VB"
    document.write(subArr.pop())
</script>
```

  - Output:
    VB

- **shift():**
  - It is used to remove and return "First Element" of Array.
  - This method changes the length of original Array.
  - Syntax:
    ArrayObject.shift()
  - Example:

```
<script languagae="javacript">
    var subArr = new Array()
    arr[0] = "PHP"
    arr[1] = "C"
    arr[2] = "VB"
    document.write(subArr.shift())
</script>
```

- Output:
  PHP

- **push():**
  - It is used to add one or more elements to end of the Array and returns new length.
  - Synatx:
    ArrayObject.push(newElement1,newElement2,…)
  - Example:
    ```
    <script  languagae="javacript">
        var subArr = new Array()
        arr[0] = "PHP"
        arr[1] = "C"
        arr[2] = "VB"
        document.write(subArr.push("java"))
    </script>
    ```
  - Output:
    4

- **unshift():**
  - It is used to add one or more elements to the beginning of the Array and returns new length.
  - Syntax:
    ArrayObject.unshift(newElement1,newElement2,…)
  - Example:
    ```
    <script  languagae="javacript">
        var subArr = new Array()
        arr[0] = "PHP"
        arr[1] = "C"
        arr[2] = "VB"
        document.write(subArr.unshift("oracle"))
    </script>
    ```
  - Output:
    4

- **reverse():**
  - It is used to reverse the order of elements in Array.
  - It changes the original Array.
  - Syntax:
    ArrayObject.reverse()
  - Example:
    ```
    <script  languagae="javacript">
        var subArr = new Array()
        arr[0] = "PHP"
        arr[1] = "C"
        arr[2] = "VB"
        document.write(subArr.reverse())
    </script>
    ```

- **Output:**
  VB,C,PHP

- **slice():**
  - Its is used to extract a section of an Arrya and returns a new Array.
  - Syntax:
    ArrayObject.slice(start_index,end_index)
  - Example:

```
<script  languagae="javacript">
    var subArr = new Array()
    arr[0] = "PHP"
    arr[1] = "C"
    arr[2] = "VB"
    document.write(subArr.slice(0,1))
</script>
```

  - Output:
    PHP

- **sort():**
  - It is used to sort the elements of an Array.
  - This method wil sort the elements alphabetically by default.
  - Numbers will not be sorted correctly.
  - To sort numbers, we must create function that compare numbers.
  - After sort() method, Array is changed.
  - Syntax:
    ArrayObject.sort()
  - Example:

```
<script languagae="javacript">
    var subArr = new Array()
    arr[0] = "PHP"
    arr[1] = "C"
    arr[2] = "VB"
    document.write(subArr.sort())      </script>
```

  - Output:
    C,PHP,VB

DATE OBJECT:

- Javascript does not have the datatype as Date.
- However, we can use Date object & it's methods to work with dates and times.
- Some important things about methods of date object.

Seconds & minutes	0 to 59
Hours	0 to 23
Day	0(Sunday) to 6(Saturday)
Date	1 to 31
Month	0(January) to 11 (December)
Year	Years since 1900

- Methods:

❑ Date()

- This method returns today's date and time.
- **Syntax:**
  dateObject.Date()
- **Example:**
  var d=new Date()
  document.write(d.Date())

❑ getDate()

- The getDate() method returns the day of the month.
- The value returned by getDate() us a number between 1 and 31.
- This method is always used in conjunction with a Date object.
- Syntax:
  dateObject.getDate()
- Example:
  var d=new Date()
  document.write(d.getDate())

❑ getDay()

- The getDay() method returns a number that represents the day of the week.
- The value returned by getDay() is a number between 0 and 6. Sunday is 0, Monday is 1 and so on..
- This method is always used in conjunction with a Date.
- Syntax:
  dateObject.getDay()
- Example:
  var d=new Date()
  document.write(d.getDay())

❑ getMonth()

- The getMonth() method reurns the month as a number.
- The value returned by the getMonth() is a number between 0 and 11. January is 0 and February is 1 and so on..
- This method is always used in conjunction with a Date object.
- Syntax:
  dateObject.getMonth()
- Example:
  var d=new Date()
  document.write(d.getMonth())

❑ getYear()

- The getYear() method returns the month, as a number.
- The value returned by getMonth() is a number between 0 & 11. January is 0, February is 1 and So on…

- This method is always used in conjunction with date object.
- Syntax:
  dateObjecct.getYear()
- Example:
  var d=new Date()
  document.write(d.getYear())

❑ getFullYear()

- The getFullYear() method returns a four digit number that represents a year.
- This method is always used in conjunction with a date object.
- Syntax:
  dateObject.getFullYear()
- Example:
  var d=new Date()
  document.write(d.getFullYear())

❑ getHours()

- The getHours() method returns the hour of a time.
- The value returned by this method is a number that represents the Hour of a day.
- This method is always used in conjunction with date object.
- Syntax:
  dateObject.getHours()
- Example:
  var d=new Date()
  document.write(d.getHours())

❑ getMinutes()

- The getMinutes() method returns the minutes of a time.
- The value returned by the getMinutes() is a number that represents minutes of a current Hour.
- This method is used with conjunction with the date object.
- Syntax:
  dateObject.getMinutes()
- Example:
  var d=new Date()
  document.write(d.getMinutes())

❑ getSeconds()

- This method returns the seconds of a time.
- The value returned by this method is a number that represents the second of a current time.
- This method is always used in conjunction with date object.
- Syntax:
  dateObject.getSeconds()
- Example:
  var d=new Date()
  document.write(d.getSeconds())

- **getMilliseconds()**
  - The getMilliseconds() returns the milliseconds of a time.
  - The value returned by this method is a number that represents a milliseconds of a current time.
  - This method is always used in conjunction with date object.
  - Syntax:
    Dateobject.getMilliseconds()
  - Example:
    d=new Date()
    document.write(d.getMilliseconds())

- **getTime()**
  - The getTime() returns the number of milliseconds since midnight of January 1, 1970.
  - This method is always used in conjunction with Date object.
  - Syntax:
    dateObject.getTime()
  - Example:
    var d=new Date()
    document.write(d.getTime())
- **setDate()**
  - The setDate() method is used to set the day of the month.
  - The value set by setDate() is a number between 1 and 31, that represents the day number of month.
  - This method is always used in conjunction with Date object.
  - Syntax:
    dateObject.setDate(day)
  - Example:
    var d=new Date()
    d.setDate(4)

- **setMonth()**
  - The setMonth() method is used to set the month.
  - You can set month by specifying the value of month and value of day.
  - The month value in setMonth() is a number between 0 and 11. January is 0, February is 1 and so on… and the day value in setMonth() is a number between 1 and 31, represents the day of month.
  - This method is always used in conjunction with Date object.
  - Here the parameter day is optional and month is required.
  - Syntax:
    dateObject.setMonth(month,day)
  - Example:
    var d=new Date()
    d.setMonth(8)

- **setFullYear()**
  - The setFullYear() method is used to set the year.
  - You can set the year by specifying the year, month and day value.
  - The year value represents the four digit value specifies the year.
  - The month value represents the number between the 0 and 11 specifies the month.
  - The day value represents the number between 1 and 31, specifies the date.
  - Here the parameter year is required and month and day are optional.
  - Syntax:
    dateObject.setFullYear(year,month,day)
  - Example:
    var d=new Date()
    d.setFullYear(2009)

- **setYear()**
  - The setYear() method is used to set the year.
  - The value set by the setYear() is a numeric value that represents the year value.
  - Always used in conjunction with date Object.
  - Syntax:
    dateObject.setYear(year)
  - Example:
    var d=new Date()
    d.setYear(year)

- **setHours()**
  - The setHours() method is used to set the hour of a specified time.
  - You can set hours by specifying the values of hours, minutes, seconds and milliseconds.
  - Here the parameter hours is required. Whereas minutes, seconds, milliseconds are optional.
  - Always used in conjunction with date object.
  - Syntax:
    dateObject.setHours(hour,minutes,seconds,millisec)
  - Example:
    var d=new Date()
    d.setHours(10)

- **setMinutes()**
  - The setMinutes() method is used to set the minutes of a specified time.
  - You can set the minutes by specifying the values of minutes, seconds and milliseconds.
  - Here the parameter minute is required where as seconds and milliseconds are optional.
  - Always used in conjunction with date object.

- Syntax:
  dateObject.setMinutes(minute,seconds,milliseconds)
- Example:
  var d=new Date()
  d.setMinutes(20)

❑ setSeconds()

- The setSeconds() method is used to set the seconds of a specified time.
- You can set the seconds by specifying the values of seconds and milliseconds.
- Here the parameter second is required and millisecond is optional.
- Always used in conjunction with date Object.
- Syntax:
  dateObject.setSeconds(seconds,milliseconds)
- Example:
  var d=new Date()
  d.setSeconds(20)

➢ Window Object

- **WINDOWS**

  When you load your browser application, a window immediately appears. This window is known as browser window or window in short. You can use JavaScript to open a window, as the result of a button click or any other operation. Since each window is represented as a distinct window object, opening a new window actually creates another window object.

- **CREATE A WINDOW**

  WindowVar=[window].open("URL","WindowName", [Window Attributes],height,width)

Window Attribute	Description
Toolbar	Back, forward and other buttons button
Location	Address bar displaying the current URL
Directories	What's new, What's Cool and other buttons in that row
Status	Status bar at the bottom of the window
Menubar	Displays menu bar to the window (File, Edit , View…)
Scrollbars	Displays scroll bar to the window
Resizable	Allows resizing of the window
Width	Windows width in pixel
Height	Windows height in pixel

EXAMPLE

```
<html>
  <head>
    <script type="text/javascript">
      function open_win()
      {
        window.open("","my1","menubar,status,toolbar,height=
        100,width=500")
        window.open("","my2","fullscreen,menubar,status,toolbar")
      }
    </script>
  </head>
  <body>
    <form>
      <input type=button value="OpenWindow"onclick="open_win()"/>
    </form>
  </body>
</html>
```

EXAMPLE

```
<html>
  <head>
    <script type="text/javascript">
      function closeWin()
      {
        myWindow.close()
      }
    </script>
  </head>
  <body>
    <script type="text/javascript">
      myWindow=window.open('','','width=200,height=100)
      myWindow.document.write("This is 'myWindow'")

    </script>
    <form>
      <input type="button" value="Close 'myWindow'"
      onclick="closeWin()" />
    </form>
  </body>
</html>
```

EXAMPLE

```
<html>
  <head>
      <script type="text/javascript">
      function resizeWindow1()
      {
            window.resizeBy(-100,-100)

            /*method is used to resize a window by the specified pixels*/
      }

      function resizeWindow2()
      {
            window.resizeTo(500,300)

            /*method is used to resize the window to the specified width
            and height.*/
      }
      </script></head>

  <body>
      <form>
            <input type="button" onclick="resizeWindow1()"
            value="Resize window By">
            <input type="button" onclick="resizeWindow2()"
            value="Resize window To">

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

➢ History Object

- The browser maintains a list of most recent URL's, which can be viewed in IE as well as Netscape. The history list behaves like a LIFO queue (Last In First Out).
- The history list is represented in JavaScript by the window. history object. It allows u to deal with list but not with the data.

History Attribute	Description
length[property]	Returns the number of elements in the history list
back()	Loads the previous URL in the history list
~~next()~~ forward()	Loads the next URL in the history list
go()	Loads a specific page in the history list

EXAMPLE

```
<html>
   <head>
      <script language = "JavaScript">
         function ShowHistory()
         {
               alert("U have accessed " +  history.length +    " Web Pages ")
         }
       </script>
   </head>
   <body onload="ShowHistory()">
   </body>
</html>
```

EXAMPLE

```
<html>
  <head>
       <script type="text/javascript">
          function goBack()
          {
              history.forward()
          }
       </script>
  </head>
  <body>
     <form>
       <input type="button" value="Back" onclick="goBack()">
     </form>
  </body>
</html>
```

EXAMPLE

```
<html>
   <head>
      <script type="text/javascript">
         function goBack()
         {
            history.go(-3)
         }
      </script>
   </head>
    <body>
       <input type="button" value="Back" onclick="goBack()" />
    </body>
</html>
```

➢ Location Object

- The Location object is actually a JavaScript object, not an HTML DOM object. The Location object is automatically created by the JavaScript runtime engine and contains information about the current URL.
- Example: Send a user to a new location.
- The Location object is part of the Window object and is accessed through the window.location property.IE: Internet Explorer, F: Firefox, O: Opera.

Location Attribute	Description
hash	Sets or returns the URL from the hash sign (#)
host	Sets or returns the hostname and port number of the current URL
hostname	Sets or returns the hostname of the current URL
href	Sets or returns the entire URL
pathname	Sets or returns the path of the current URL
port	Sets or returns the port number of the current URL
protocol	Sets or returns the protocol of the current URL
search	Sets or returns the URL from the question mark (?)
assign()	Loads a new document
reload()	Reloads the current document
replace()	Replaces the current document with a new one

EXAMPLE

```
<html>
   <head>
      <script type="text/javascript">
         function newDoc()
               {
                  window.location.assign("http://www.yahoo.com")
               }
      </script>
   </head>
   <body>
       <input type="button" value="Load new document"
       onclick="newDoc()" />
   </body>
</html>
```

EXAMPLE

```
<html>
   <head>
      <script type="text/javascript">
         function reloadPage()
                {
                      window.location.reload()
               }
      </script>
   </head>
   <body>
      <input type="button" value="Reload page" onclick="reloadPage()">
   </body>
</html>
```

Example

```
<html>
   <head>
      <script type="text/javascript">
         function replaceDoc()
                {
                      window.location.replace("http://www.yahoo.com")
                 }
      </script>
   </head>
   <body>
         <input type="button" value="Replace document"
       onclick="replaceDoc()" />
   </body>
</html>
```

Example

```
<html>
   <body>
      <script type="text/javascript">
         document.write(location.search);
      </script>
   </body>
</html>
```

Example

```
<html>
   <body>
      <script type="text/javascript">
         document.write(location.href);
      </script>
   </body>
</html>
```

➢ Navigator Object

- The Navigator object is actually a JavaScript object, not an HTML DOM object. The Navigator object is automatically created by the JavaScript runtime engine and contains information about the client browser. **IE:** Internet Explorer, **F:** Firefox, **O:** Opera.

Navigator Attribute	Description
appCodeName	Returns the code name of the browser
appName	Returns the name of the browser
appVersion	Returns the platform and version of the browser
browserLanguage	Returns the current browser language
cookieEnabled	Returns a Boolean value that specifies whether cookies are enabled in the browser
cpuClass	Returns the CPU class of the browser's system
onLine	Returns a Boolean value that specifies whether the system is in offline mode
platform	Returns the operating system platform
systemLanguage	Returns the default language used by the OS
userAgent	Returns the value of the user-agent header sent by the client to the server
userLanguage	Returns the OS' natural language setting
javaEnabled()	Specifies whether or not the browser has Java enabled
taintEnabled()	Specifies whether or not the browser has data tainting enabled

Example

```
<html>
   <body>
      <script type="text/javascript">
         document.write("<p>Browser: ")
         document.write(navigator.appName + "</p>")
         document.write("<p>Browserversion: ")
         document.write(navigator.appVersion + "</p>")
         document.write("<p>Code: ")
         document.write(navigator.appCodeName + "</p>")
         document.write("<p>Platform: ")
         document.write(navigator.platform + "</p>")
         document.write("<p>Cookies enabled: ")
         document.write(navigator.cookieEnabled +"</p>")
         document.write("<p>Browser's useragent header")
         document.write(navigator.userAgent + "</p>")
      </script>
   </body>
</html>
```

Example

```
<html>
   <body>
      <script type="text/javascript">
         var x = navigator
       document.write("CodeName=" + x.appCodeName )
       document.write("MinorVersion=" + x.appMinorVersion )
       document.write("Name=" + x.appName + "<br>")
       document.write("Version=" + x.appVersion + "<br>")
       document.write("CookieEnabled=" + x.cookieEnabled)
       document.write("CPUClass=" + x.cpuClass + "<br>")
       document.write("OnLine=" + x.onLine + "<br>")
       document.write("Platform=" + x.platform + "<br>")
       document.write("UA=" + x.userAgent + "<br>")
       document.write("BrowserLanguage=" + x.browserLanguage)
       document.write("SystemLanguage=" + x.systemLanguage
       document.write("UserLanguage=" + x.userLanguage)
      </script>
  </body>
</html>
```

➢ User Defined Objects :

- JavaScript allows you to create "USER DEFINED OBJECT".
- A User Defined Object will also be associated with properties and methods.
- For Example:
  - A person is an object. The Properties of person's include name, height, weight, age etc. All persons have these properties but the values of those properties will differ from person to person. The person's methods could be eat(), sleep(), work(), play() etc.
- There are different ways to create new Object.
  - Create an Object with direct initialization of Properties.
  - Create a direct instance of Object.
  - Create a template of an Object.

Create a direct instance of an object

Example

```
<html>
      <body>
            <script language="javascript">
                  myobj = new Object();
                  myobj.subject = "Web development"
                  myobj.part1="Dhtml"
                  myobj.part2="Javascript"
                  myobj.part3="Php and Mysql"
```

```
                document.write(myobj.subject+"<br>")
                document.write(myobj.part1+"<br>")
                document.write(myobj.part2+"<br>")
                document.write(myobj.part3+"<br>")
            </script>
        </body>
    </html>
```

Create a template of an object

Example

```
    <html>
        <body>
            <script language="javascript">
                function myobject(subject,part1,part2,part3)
                {
                    this.subject = subject
                    this.part1= part1
                    this.part2= part2
                    this.part3= part3
                }
                myobj = new myobject("Web development",
                        "Dhtml","JavaScript","Php & Mysql");
                document.write(myobj.subject+"<br>")
                document.write(myobj.part1+"<br>")
                document.write(myobj.part2+"<br>")
                document.write(myobj.part3+"<br>")
            </script>
        </body>
    </html>
```

❑ Document Object Model

- An HTML page is rendered in a browser.
- The browser assembles all the elements contained in the html page, downloaded from the web server in its memory.
- Once done the browser then renders these objects in the browser window.
- One the HTML page is rendered in the browser window, the browser can no longer recognize individual HTML elements.
- To create an interactive web page it is imperative that the browser continues to recognize individual HTML objects even after they are rendered in the browser window.
- This allows the browser to access the properties of these objects using the built in methods of the objects.
- Once the properties of an object are accessible then the functionality of the object can be controlled at will.

- JavaScript enabled browsers are capable of recognizing individual objects in an HTML page, after the page has been rendered in the browser, because the JavaScript enabled browser recognize and uses the Document Object Model.
- Using the Document Object Model javascript enabled browsers identify the collection of web page objects (web page elements) that have to be dealt with while rendering an HTML based, web page in the browser window.
- The top most objects in the DOM are the browser itself. The next level in DOM is the browser's window. The next level is the document displayed in the browser window.

❑ Working With Forms in JavaScript.

DOM methods and properties

1. getElementById()
2. getElementByName()
3. innerHTML property

1.getElementById()

- If you want to quickly access the value of an HTML input, than this method can be used.
- It return a reference of the HTML element (control).
- Syntax:

  getElementById(Control Reference);
- For example:

```
<head>
    <script language=javascript>
     Function notEmpty()
     {
          var mytextfield=document.getElementById("mytext");
          if(mytextfield.value!=" ")
          {
               alert("You entered"+mytextfield.value);
           }
           else
          {
               alert("Would you please enter some text?");
          }
     }
</script>
<body>
   <form>
        <input type=text id="mytext">
        <input type=button onClick=notEmpty() value="Form
Checker">
    </form>
</body>
```

- Here document.getElementById returned a reference to our HTML element mytext.
- We stored this reference into a variable, mytextfield, and then used the value property that all input elements have to use to grab the value the user enters.
- getElementById is a method or a function of the document object. This means you can only access it by using document.getElementById.

2. getElementByName()

- It returns a collection of objects with the specified NAME.
- Syntax:
  document.getElementByName(Name);
- for example:

```
<head>
    <script language=javascript>
       function getElements()
        {
            var x=document.getElementByName("myinput");
            alert(x.length);
         }
    </script>
    </head>
    <body>
       <form>
            <input  type=text name="myinput">;
            <input  type=text name="myinput">;
            <input  type=text name="myinput">;

            <input type=button onClick=getElements() value="How
    many Elements named MyInputs">
    </form>
    </script>
</body>
```

3. innerHTML property

- The easiest way to get or modify the content of an element is by using the innerHTML property.
- When the innerHTML property is set, the given string completely replaces the existing content of the object. If the string contains HTML tags, the string is parsed and formatted as it is placed into the document.

- This property is accessible at run time as the document is being parsed;
- Each HTML element has an innerHTML property that defines both the HTML code and the text that occurs between that element's opening and closing tag. By changing an element's innerHTML after some user interaction, you can make much more interactive pages.
- However, using innerHTML requires some preparation if you want to be able to use it easily and reliably. First, you must give the element you wish to change an id. With that id in place you will be able to use the getElementById function, which works on all browsers.
- After you have that set up you can now manipulate the text of an element. To start off, let's try changing the text inside a bold tag.

JavaScript Code:

```
<script type="text/javascript">
function changeText()
{
    document.getElementById('boldStuff').innerHTML = 'Fred Flinstone';
}
</script>
<p>Welcome to the site <b id='boldStuff'>dude</b> </p>
<input type='button' onclick='changeText()' value='Change Text'/>
Onclick=document.bgColor=Red
```

- ❑
- ❑ **EVENTS**

WHAT IS EVENTS

- It describes actions that occur as a result of user interaction with the web page or other browser related activities. When a button is clicked or a mouse has been moved or even when a key has been pressed an event is said to be occurred.

EVENT HANDLING

- The response in conjunction to the occurrence of events by the web browser is called as event handling.

EVENT HANDLER

- The code that performs this processing is called event handler.

Event Handling Attribute	Explanation
OnAbort	Loading of a image is aborted as the result of a user action
OnBlur	A form element, document, frameset, window loses the current input focus
OnChange	When selection is modified and loses the current focus
OnClick	Form element or link is clicked
OnDblClick	Form element or link is double clicked
OnDragDrop	A dragged object is dropped in a window or frame
OnError	Error occurs during loading of a window
OnFocus	A form element, document, frameset, window receives the focus
OnKeyDown	A key has been pressed
OnKeyPress	A key has been pressed and released
OnKeyUp	A key has been released
OnLoad	A form element, document, frameset, window has been loaded
OnMouseDown	A mouse button has been pressed
OnMouseMove	The mouse has been moved
OnMouseOut	The mouse had been moved out of a link or image map
OnMouseOver	The mouse had been moved over of a link or image map
OnMouseUp	The mouse button has been released
OnMove	The window or frame has been moved
OnReset	The form has been reset by clicking the reset button
OnResize	The window or a frame has been resized
OnSelect	Text has been selected
OnSubmit	A form has been submitted
OnUnload	A document or a frame set is exited

Example [onClick]

```
<html>
   <body>
      <script language = "JavaScript">
         function connect()
                {
              check = confirm(" Wanna Visit Hotmail Site ")
          if(check)
                return true
          else
                return false
         }
      </script>
```

```
        <A href = "http://www.hotmail.com"  OnClick  = "return    connect()">
              Click to contect
         </A>
    </body>
</html>
```

Example [onMouseOver and onMouseOut]

```
<html>
   <body>
      <script language = "JavaScript">
         count = 0
         function over()
              {
                    alert(" The Over count = " + ++count)
              }
         function out()
              {
                     alert(" The Out count = " + --count)
              }
      </script>
       <A href = " " OnMouseOver = "over()" >
              Click to Check The Mouse Over  Count
        </A> <br>
       <A href = " " OnMouseOut = "out()" >
              Click to Check The Mouse Out  Count
       </A>
   </body>
</html>
```

Example [onBlur and onFocus]

```
<html>
       <body>
         <script language = "JavaScript">
            count = 0
            function Black()
                {
                      document.bgColor = "Black"
                }
            function White()
                {
                    document.bgColor = "Grey"
                }
        </script>
```

```
        <form name = "frm1" >
            <input type = "text"   name = "txt1" OnBlur="White()"
                                OnFocus = "Black()">
             <input type = "button" name="but" value = "Click"
                                OnClick="Black()">
        </form>
    </body>
</html>
```

Example[onLoad and onUnLoad]

```
<body onLoad="alert('Page is Load')"
          onUnLoad="alert('Page is unload')">
 </body>
```

❑ Timer Event

- The timer event is used to execute the statement[s] repeatedly at some interval.
- The interval will be in milliseconds. Once the timer event is started it will continue the execution until the event is been stopped

Example [timer Event/timer function]

```
<html>
   <head>
      <script type="text/javascript">
        function timedMsg()
              {
                    var t=setTimeout("alert('5 seconds!')",5000)
              }
      </script>
   </head>
   <body>
      <form>
          <input type="button" value="Display timed alertbox!"
       onClick="timedMsg()">
      </form>
      <p>Click on the button above. An alert box will be displayed after 5
seconds.</p>
   </body> </html>
```

Example

```
<html>
        <head>
                <script type="text/javascript">
                        //Only for IE
                        var msg = "     Your Name Goes Here       "
                        function scrollme()
                        {
                                window.status = msg
                                msg = msg.substring(1,msg.length) +
                        msg.substring(0,1)
                            timer = setTimeout("scrollme()",200)
                          }
                           scrollme()
                    </script>
                </head>
</html>
```

Example [stop timer control/disable timer control]

```
<html>
   <head>
      <script type="text/javascript">
         var c=0; var t
         function timedCount()
         {
            document.getElementById('txt').value=c
            c=c+1
            t=setTimeout("timedCount()",1000)
         }
         function stopCount()
         {
            clearTimeout(t)
         }
      </script>
   </head>
   <body>
      <form>
               <input type="button" value="Start count!"
onClick="timedCount()">
               <input type="text" id="txt">
               <input type="button" value="Stop count!"
        onClick="stopCount()">
      </form>
   </body>
</html>
```

❑ Cookies

- Cookies are the temporary storage area where the information is to be stored. The cookies are the seen in the address bar of the browser, history of the websites visited, etc.
- The Cookies which are created are present in the local machine and located in C:\documents and settings\admin\cookies.
- These activated cookies are stored in the form of text document where in the information about the cookie name, time of visit, etc is stored. For an individual website individual text file is created in local machine.
- Its propose is to identify the user, to keep continuities between users and to allow a web server to personalize a web page, depending on previously submitted.
- The cookies depend on time and cookie file is not exceeds then 4KB. To destroy the cookie it has to be done manually by deleting it from the given above location.
    - To Set the Cookie in JavaScript, the syntax is:
    - Set-cookie:NAME=value; EXPIRES=date; PATH=path; DOMAIN=domain; SECURE
    - Where NAME parameter is where the name of the cookie is specified.
    - EXPIRES=date is where the date is to be specified when the cookie should expire.
    - DOMAIN is where URL for which the cookie is valid is to be specified.
    - SECURE is used when transmitted over a secure link.

- **Example:**

```
<html>
  <head>
     <script language=”javascript”>
          Function setcookie()
          {
              document.cookie=”Welcome”;
          }
     </script>
     <body onload=”setcookie();”>
         <script language=”javascript”>
             var s=document.cookie;
             document.write(s);
         </script>
     </body>
 </html>
```

Another Example: Function that displays a welcome message if the cookie is set, and if the cookie is not set it will display a prompt box, asking for the name of the user:-

```
Function checkCookie()
{
  Username=getCookie('username');
  If(username!=null&&username!="")
  {
    alert('Welcome  again'+username"'!');
  }
else
{
      Username=prompt('Please enter your Name:',"");
       if(username!=null&&username!="")
       {
         setCookie('username',username,365);
        }
 }
}
```

❑ Common Mistakes

- There are seven common mistakes made by programmers.
- Some of these you'll learn to avoid as you become more experienced, but others may haunt you forever!

❑ Undefined Variables

- JavaScript is actually very easygoing when it comes to defining your variables before assigning values to them.
- For example, the following will implicitly create the new global variable abc and assign it to the value 23:

```
abc = 23;
```

- Although strictly speaking, you should define the variable explicitly with the var keyword like this:

```
var abc = 23;
```

- Whether or not you use the var keyword to declare a variable has a consequence of what scope the variable has; so it is always best to use the var keyword.
- If a variable is used before it has been defined, an error will arise.

❑ Case Sensitivity

- This is a major source of errors, particularly because it can be difficult to spot at times.
- For example, spot the three case errors in the following code:

```
var myName = "Jeremy";
If (myName == "jeremy")
  alert(myName.toUppercase());
```

- Here we find 3 type of error due to the case sensitivity. If is not correct because it must be written into small case.
- Second thing is we are comparing the word 'jeremy' instead of 'Jeremy'.
- The third fault is with the toUpperCase() method of the String object contained in myName.

❑ **Incorrect Number of Closing Braces**

- In the following code, you define a function and then call it. However, there's a deliberate mistake.
- See if you can spot where it is.

```
function myFunction()
{
x = 1;
y = 2;
if (x <= y)
{
if (x == y)
{
alert("x equals y");
}
}
myFunction();
```

- Now you can see that the ending curly brace of the function is missing.
- When there are a lot of if, for, or do while statements, it's easy to have too many or too few closing braces.
- This type of problem is much easier to spot with formatted code.

❑ **Incorrect Number of Closing Parentheses**

- Take a look at the following code:

```
if (myVariable + 12) / myOtherVariable < myString.length)
```

- Spot the mistake? The problem is the missing parenthesis at the beginning of the condition.
- You want myVariable + 12 to be calculated before the division by myOtherVariable is calculated, so quite rightly you know you need to put it in parentheses.
- It's very easy to miss a parenthesis or have one too many when you have many opening and closing parentheses.

❑ **Using Equals (=) Rather than Is Equal To (==)**

- Consider the following code:

```
var myNumber = 99;
if (myNumber = 101)
{
  alert("myNumber is 101");
}
else
{
  alert("myNumber is " + myNumber);
}
```

- You'd expect, at first glance, that the alert() method in the else part of the if statement would execute, telling us that the number in myNumber is 99, but it won't.
- This code makes the classic mistake of using the assignment operator (=) instead of the equality operator (==).
- Hence, instead of comparing myNumber with 101, this code sets myNumber to equal 101.

❑ **Using a Method as a Property and Vice Versa**

- Another common error is where either you forget to put parentheses after a method with no parameters, or you use a property and do put parentheses after it.
- When calling a method, you must always have parentheses following its name; otherwise, JavaScript thinks that it must be a pointer to the method or a property.
- For example, examine the following code:

```
var nowDate = new Date();
alert(nowDate.getMonth);
```

- The first line creates an instance of the Date reference type. The second line attempts to call the getMonth() method of the newly created Date object, except the parentheses are missing.
- Just as you should always have parentheses after a method, you should never have parentheses after a property; otherwise, JavaScript thinks you are trying to use a method of that object:

```
var myString = "Hello, World!";
alert(myString.length());
```

- The second line adds parentheses after the length property, making JavaScript think it is a method.
- As a rule of thumb, use parentheses at the end of the function name when you want to execute the function, and leave the parentheses off when passing the function to another function or property.

❑ **Missing Plus Signs During Concatenation**

- In the following code, there's a deliberate concatenation mistake:

```
var myName = "Jeremy";
var myString = "Hello";
var myOtherString = "World";
myString = myName + " said " + myString + " " myOtherString;
alert(myString);
```

- There should be a + operator between " " and myOtherString in the fourth line of code.

- Although easy to spot in just a few lines, this kind of mistake can be harder to spot in large chunks of code.
- Also, the error message this type of mistake causes can be misleading.
- These most common mistakes are errors caused by the programmer.

- There are other types of errors, called run-time errors, that occur when your code executes in the browser, and they aren't necessarily caused by a typo, missing curly brace, parenthesis, or other pitfalls discussed.

❑ Error-Handling in JavaScript

JavaScript Try...Catch Statement

The try...catch statement allows you to test a block of code for errors.

JavaScript - Catching Errors

When browsing Web pages on the internet, we all have seen a JavaScript alert box telling us there is a runtime error and asking "Do you wish to debug?". Error message like this may be useful for developers but not for users. When users see errors, they often leave the Web page. This will teach you how to catch and handle JavaScript error messages

The try...catch Statement

The try...catch statement allows you to test a block of code for errors. The try block contains the code to be run, and the catch block contains the code to be executed if an error occurs.

Syntax

```
Try
{
     //Run some code here
 }
catch(err)
{
    //Handle errors here
}
```

Note that try...catch is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Examples

The example below is supposed to alert "Welcome guest!" when the button is clicked. However, there's a typo in the message() function. alert() is misspelled as adddlert(). A JavaScript error occurs. The catch block catches the error and executes a custom code to handle it. The code displays a custom error message informing the user what happened:

Example

```
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
try
  {
  adddlert("Welcome guest!");
  }
catch(err)
  {
  txt="There was an error on this page.”;
  txt+="Error description: " + err.description;
  txt+="Click OK to continue";
  alert(txt);
  }
}
</script></head>
<body>
<input type="button" value="View message" onclick="message()" />
</body></html>
```

The next example uses a confirm box to display a custom message telling users they can click OK to continue viewing the page or click Cancel to go to the homepage. If the confirm method returns false, the user clicked Cancel, and the code redirects the user. If the confirm method returns true, the code does nothing:

Example

```
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
try
  {
  adddlert("Welcome guest!");
  }
catch(err)
  {
  txt="There was an error on this page";
  txt+="Click OK to continue viewing this page,";
  txt+="or Cancel to return to the home page";
  if(!confirm(txt))
    {
    document.location.href="http://www.XYZ.com/";
    }
  }
}
</script>
</head>

<body>
<input type="button" value="View message" onclick="message()" />
</body>

</html>
```

The throw Statement

The throw statement can be used together with the try...catch statement, to create an exception for the error. The throw statement allows you to create an exception. If you use this statement together with the try...catch statement, you can control program flow and generate accurate error messages.

Syntax

```
throw(exception)
```

The exception can be a string, integer, Boolean or an object.Note that throw is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Example:The example below determines the value of a variable called x. If the value of x is higher than 10, lower than 0, or not a number, we are going to throw an error. The error is then caught by the catch argument and the proper error message is displayed:

Example

```
<html><body>
<script type="text/javascript">
var x=prompt("Enter a number between 0 and 10:","");
try
  {
  if(x>10)
    {
    throw "Err1";
    }
  else if(x<0)
    {
    throw "Err2";
    }
  else if(isNaN(x))
    {
    throw "Err3";
    }
  }
catch(er)
  {
  if(er=="Err1")
    {
    alert("Error! The value is too high");
    }
  if(er=="Err2")
    {
    alert("Error! The value is too low");
    }
  if(er=="Err3")
    {
    alert("Error! The value is not a number");
    }
  }</script></body></html>
```