Sagi Rama Krishnam Raju Engineering College

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Lecture Notes

on

JAVASCRIPT

JAVASCRIPT

Objective:

- To develop interactive web applications. Validate HTML forms

Syllabus:

Elements of Objects in Java Script, Dynamic HTML with Java Script.

LEARNING MATERIAL

INTRODUCTION TO JAVA SCRIPT

Web pages are two types

- Static web page: there is no specific interaction with the client
- Dynamic web page: web page which is having interactions with client and as well as validations can be added.

Script means small piece of Code.

Scripting Language is a high-level programming language, whose programs are interpreted by another program at run time rather than compiled by the computer processor.

BY using JavaScript we can create interactive web pages. It is designed to add interactivity to HTML pages.

Scripting languages are of 2 types.

- client-side scripting languages
- servers-side scripting languages

In general Client-side scripting is used for performing simple validations at client-side;

Server-side scripting is used for database verifications.

- Client-side scripting languages : VBScript, JavaScript and Jscript.
- Server-side scripting languages : ASP, JSP, Servlets and PHP etc.

- Simple HTML code is called static web page, if you add script to HTML page it is called dynamic page.
- Netscape Navigator developed JavaScript and Microsoft's version of JavaScript is Jscript.

Features of JavaScript:

- JavaScript is a lightweight, interpreted programming language means that scripts execute without preliminary compilation.
- It is an Object-based Scripting Language.
- Designed for creating network-centric applications. It is usually embedded directly into HTML Pages.
- Java script code as written between <script>---- </script> tags All Java script statements end with a semicolon
- Java script ignores white space
- Java script is case sensitive language
- Script program can be saved as either .js or .html Complementary to and integrated with Java.
- Open and cross-platform.

Advantages of JavaScript:

- Can put dynamic text into an HTML page Used to Validate form input data
- Javascript code can react to user events
- Can be used to detect the visitor's browser

Limitations of JavaScript:

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multithreading or multiprocessor capabilities.

JAVA Vs JAVASCRIPT:

JAVA	JAVASCRIPT
1. Object Oriented Programming Language	2.1 Object based Scripting Language
2. Platform Independent	2.2 Browser Dependant
3. It is both compiled and interpreted	2.3 It is interpreted at runtime
4. It is used to create server side applications and standalone programming	2.4 It is used to make the web pages more interactive
5. Java is a strongly typed language	2.5 JavaScript is not strongly typed(Loosely Typed)
6. Developed by sun Microsystems	2.6 Developed by Netscape
7. Java Programs can be standalone	2.7 JavaScript must be placed inside an HTML document to function

Embedding JavaScript in an HTML Page:

Embed a JavaScript in an HTML document by using <script> and </script> html tags.

Syntax:

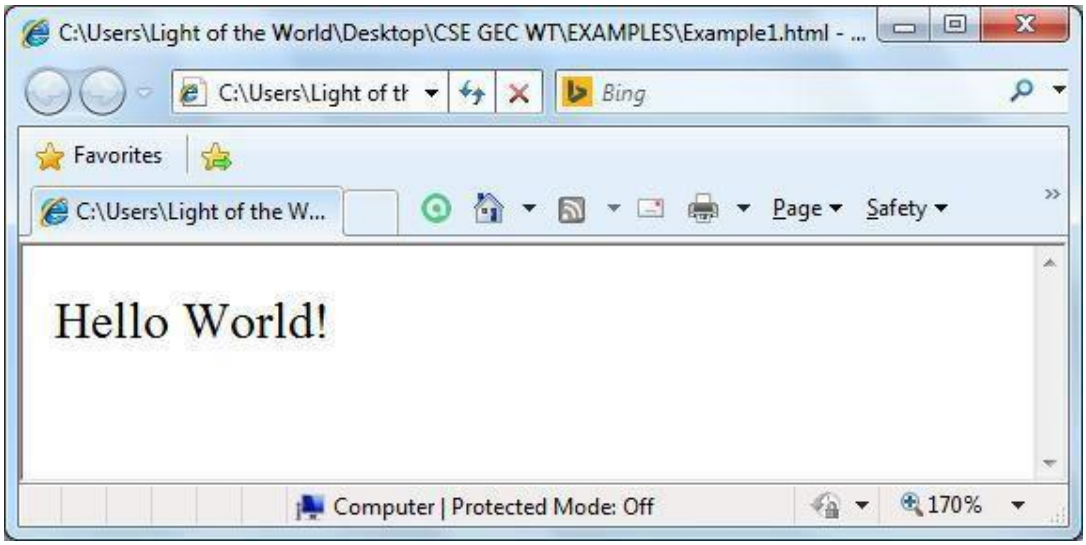
```
<script ...>
        JavaScript code
</script>
```

<script > tag has the following attributes.

Type	Refers to the MIME (Multipurpose Internet Mail Extensions) type of the script.
Language	This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.

Example:

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



Comments in JavaScript:

JavaScript supports both C-style and C++-style comments.

Thus:

- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.

VARIABLES:

- Like any programming language JavaScript has variables.
- Stores data items used in the script.
- Strict rules governing how you name your variables (Much like other languages):

Naming Conventions for Variables:

- Variable names **must** begin with a letter, digit or underscore;
- You can't use spaces in names
- Names are **case sensitive** so the variables fred, FRED and frEd all refer to **different** variables,
- It is not a good idea to name variables with similar names
- You **can't use a reserved word** as a variable name, **e.g.** var.

Creating Variables

- Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword as follows.

```
<script type="text/javascript">
        var name;
        var rollno;
</script>
```

- Storing a value in a variable is called **variable initialization**. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

```
<script type="text/javascript">
        var name = "Aziz";
        var rollno=501;
</script>
```

Scope of Variables in JavaScript:

The scope of a variable is the region of your program in which it is defined and is accessible.
JavaScript variables have only two scopes.

Global Variables: A global variable has global scope which means it can be defined and used anywhere in your JavaScript code.

Local Variables: A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Automatically Global:

- If you assign a value to a variable that has not been declared, it will automatically become a **GLOBAL** variable.
- This code example will declare a global variable **price**, even if the value is assigned inside a function.

Example:

```
myFunction();
//code here can use price
function myFunction()
{
    price = 250; //has Global scope
}
```

DATA TYPES:

JavaScript has only four types of data

- Numeric
- String
- Boolean
- Null

Numeric :

- Integers such as 108 or 1120 or 2016
- Floating point values like 23.42, -56.01 and 2E45.
- No need todifferentiate between.
- In fact variables can change type within program.

String:

- A String is a Collection of character.
- All of the following are strings:

  "Computer", "Digital" , "12345.432".

- Put quotes around the value to a assign a variable: name = "Uttam K.Roy";

Boolean:

- Variables can hold the values true and false. Used a lot in conditional tests (later).

Null:

- Used when you don't yet know something.
- A null value means one that has not yet been decided.
- It does not mean nil or zero and should NOT be used in that way.

FUNCTIONS:

- A function is a group of reusable code which can be called anywhere in your program.
- This eliminates the need of writing the same code again and again.
- It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.
  - Like any other advanced programming language, JavaScript also supports all the features necessary to write modular code using functions.
- We were using these functions again and again, but they had been written in core JavaScript only once.
- JavaScript allows us to write our own functions as well.

Function Definition

- Before we use a function, we need to define it.
- The most common way to define a function in JavaScript is
- By using keyword **function**, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax:

```
<script type="text/javascript">
function functionname(parameter-list)
{
        statements
}
</script>
```

Example:

```
<script type="text/javascript">
function sayHello()
{
    alert("Hello.. How are You");
}
</script>
```

Calling a Function:

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

```
<html>
<head>
<script type="text/javascript">
function sayHello()
{
        document.write ("Hello there!");
}
</script>
</head>
<body>
        <p>Click the following button to call the function</p>
<form>
        <input type="button" onclick="sayHello()" value="Say Hello">
</form>
<p>Use different text in write method and then try...</p>
</body>
</html>
```


C:\Users\Light of the World\Desktop\CSE GEC WT\EXAMPLES\Example1.html - Windows In...
Favorites
Page
Safety
Tools
Click the following button to call the function
Say Hello
Done
Computer | Protected Mode: Off
170%


C:\Users\Light of the World\Desktop\CSE GEC WT\EXAMPLES\Example1.html - Windo...
Favorites
Page
Safety
Tools
Hello..Welcome to JavaScript
Computer | Protected Mode: Off
170%

OPERATORS

JavaScript supports the following types of operators.

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical (or Relational) Operators
- Conditional (or ternary) Operators

Arithmetic Operators:

JavaScript supports the following arithmetic operators:

Assume variable A holds 10 and variable B holds 20, then:

Operator	Descrition	Example
+	Adds two numbers or joins two strings	20+10 returns 30
-	Substracts two numbers or represents a negative number	20-10 returns 10
*	Multiplies two numbers	20*10 returns 200
/	Divides two numbers evenly and returns the quotient	20/10 returns 2
%	Divides two numbers and returns the remainder	20%10 returns 0
++	Increments the value of a number by 1 Prefix (Pre-increment) Suffix (Post-increment)	m = 20 n=++m assigns 21 to n
		m = 20 n=m++ assigns 21 to n

Assignment Operators:

Operator	Descrition	Example
=	Assigns the value on the right hand side to the variable on left hand side	m=20
+=	Adds the right hand side operand to the left hand side operand and assigns the result to the left hand side operand.	m = 20 n = 10 m+=n assigns 30 to m
-=	Subtracts the right hand side operand from the left hand side operand and assigns the result to the left hand side operand.	m = 20 n = 5 m-=n assigns 15 to m
*=	Multiplies the right hand side operand and the left hand side operand and assigns the result to the left hand side operand.	m = 20 n = 10 m*=n assigns 200 to m
/=	Devides the left hand side operand by the right hand side operand and assigns the quotient to the left hand side operand.	m = 20 n = 10 m/=n assigns 2 to m
%=	Devides the left hand side operand by the right hand side operand and assigns the remainder to the left hand side operand.	m = 20 n = 10 m%=n assigns 0 to m

Comparision Operator

Operator	Descrition	Example
==	Returns true if both the operands are equal otherwise returns false	20==10 returns false
!=	Returns true if both the operands are not equal otherwise returns false	20 !=10 returns true
>	Returns true if left hand side operand Is greater than the right hand side operand. otherwise returns false	20 > 10 returns true
>=	Returns true if left hand side operand is greater than or equal to the right hand side operand. otherwise returns false	20 >= 10 returns true
<	Returns true if left hand side operand Is less than the right hand side operand. otherwise returns false	20 < 10 returns false
<=	Returns true if left hand side operand is less than or equal to the right hand side operand. otherwise returns false	20 <= 10 returns false

Logical (or Relational) Operators:

Operator	Descrition	Example
&&	Returns true only if both operands are the true, otherwise returns false	True && True returns True
||	Returns true only if either the of operands are true. It returns false when both the operands are false	True || False returns True
!	Negates the operand	!true returns false

Conditional or Ternary Operators

Operator	Descrition	Example
?:	Returns the second operand if the first operand is true, otherwise returns the third operand.	Result=(20 > 10)? 20 : 10 Here, 20 is assigned to Result

CONTROL FLOW STATEMENTS:

In JavaScript we have the following conditional statements:

- Use **if** to specify a block of code to be executed, if a specified condition is true
- Use **else** to specify a block of code to be executed, if the same condition is false
- Use **else if** to specify a new condition to test, if the first condition is false
- Use **switch** to specify many alternative blocks of code to be execute

The if Statement

Syntax

```
if (condition)
 {
   block of code to be executed if the condition is true
 }
```

The else Statement

Use the **else** statement to specify a block of code to be executed if the

```
condition is false. if (condition)
{
   block of code to be executed if the condition is true

}
 else
 {
   block of code to be executed if the condition is false
 }
```

The else if Statement

Use the **else if** statement to specify a new condition if the first condition is false.

Syntax:

```
if (condition1)
{
    block of code to be executed if condition1 is true
}
else if (condition2)
{
    block of code to be executed if the condition1 is false and condition2 is true
}
else
 {
   block of code to be executed if the condition1 is false and condition2 is false
}
```

Switch Statement:

Use the switch statement to select one of many blocks of code to be executed.

Syntax:

```
switch(expression) {
    case n:
        code block
        break;
    case n:
        code block
        break;
    default:
        default code block
}
```

This is how it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case. If there is a match, the associated block of code is executed.

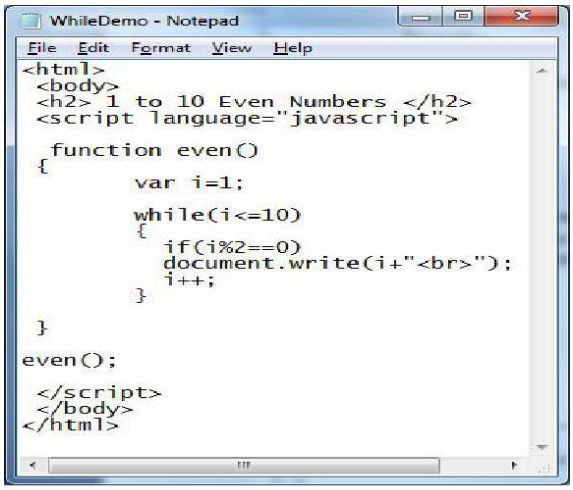
The While Loop

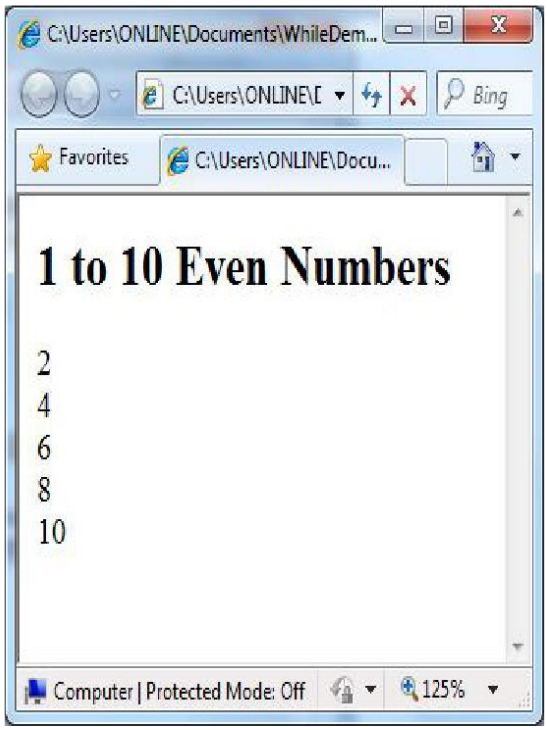
Syntax:

```
while (condition)
 {
    code block to be executed
}
```

Example:

Write a JavaScript code to print 1 to 10 even numbers using while loop.





The Do/While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
do
 {
    code block to be executed
}
while (condition);
```

The for Loop:

The for loop has the following syntax:

```
for (statement 1; statement 2; statement 3)

     {
                    code block to be executed
     }
```

Statement 1 is executed before the loop (the code block) starts.
Statement 2 defines the condition for running the loop (the code block).
Statement 3 is executed each time after the loop (the code block) has been executed.

Example:

Write a JavaScript code to print 1 to 10 even numbers using for loop.

ForDemo - Notepad

File Edit Format View Help

```
<html>
 <body>
 <h2> 1 to 10 Even Numbers </h2>
 <script language="javascript">

  function even()
 {
        for(i=1;i<=10;i++)
        {
          if(i%2==0)
          document.write(i+"<br>");
        }
 }

even();

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

C:\Users\ONLINE\Documents\ForDemo.html - Wind...

C:\Users\ONLINE\[Bing

Favorites C:\Users\ONLINE\Documents\ForDemo.html

1 to 10 Even Numbers

2
4
6
8
10

Computer | Protected Mode: Off 125%

OBJECTS IN JAVA SCRIPT: (BUILT-IN OBJECTS)

- An Object is a thing.
- There are pre defined objects and user defined objects in Javascript. Each object can have properties and methods:
- A property tells you something about an object. A method performs an action
- The following are some of the Pre defined objects/Built-in Objects.

- Document
- Window
- Browser/Navigator Form
- String
- Math
- Array
- Date

HTML DOM

- Document Object Model (DOM) is an interface that allows programs / scripts to dynamically access and change the content, structure and style of a document that includes forms.
- HTML DOM can be used to access and manipulate HTML elements using JavaScript.
- In the HTML DOM (Document Object Model), everything is a **node**:
- The document itself is a document node All HTML elements are element nodes All HTML attributes are attribute nodes
- Text inside HTML elements are text nodes Comments are comment nodes
- The Objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a Web document.

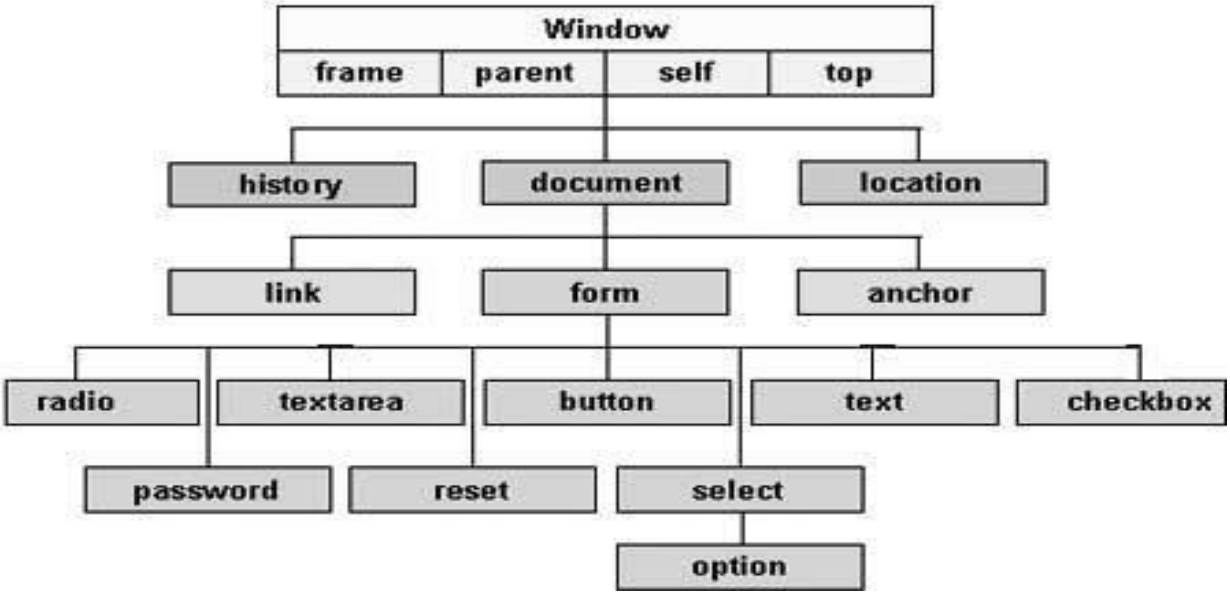
  Window object – Top of the hierarchy. It is the outmost element of the object hierarchy.

  Document object – Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page.

Form object – Everything enclosed in the <form>...</form> tags sets the form object.

Form control elements – The form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.

Here is a simple hierarchy of a few important objects –



THE DOCUMENT OBJECT

When an HTML document is loaded into a web browser, it becomes a **document object.**

- The document object is the root node of the HTML document and the "owner" of all other nodes (element nodes, text nodes, attribute nodes, and comment nodes).
- The document object provides properties and methods to access all node objects, from within JavaScript.

  Tip: The document is a part of the Window object and can be accessed as window.document.

Properties		
alinkColor	-	The color of active links
bgColor	-	Sets the background color of the web page. It is set in the <body> tag. The following code sets the background color to white.
Title	-	The name of the current document as described between the header TITLE tags.
URL	-	The location of the current document.
vlinkColor	-	The color of visited links as specified in the <body> tag

Methods

getElementById(*id*)	-	Find an element by element id
getElementsByTagName(*name*)	-	Find elements by tag name
getElementsByClassName(*name*)	-	Find elements by class name
write(text)	-	Write into the HTML output stream
writeln(text)	-	Same as write() but adds a new line at the end of the output

WINDOW OBJECT:

- The **window** object is supported by all browsers. It represents the browser's window. All global JavaScript objects, functions, and variables automatically become members of the window object.
- Global variables are properties of the window object. Global functions are methods of the window object.
- Even the document object (of the HTML DOM) is a property of the window object: window.document.getElementById("header"); is the same as: document.getElementById("header");

Properties

defaultStatus - This is the default message that is loaded into the status bar when the window loads.

opener The object that caused the window to open.

status - The status bar is the bar on the lower left side of the browser and is used to display temporary messages

length - The number of frames that the window contains.

Methods

- alert("message") - The string passed to the alert function is displayed in an alert dialog box.
- open("URLname","Windowname",["options"]) - A new window is opened with the name specified by the second parameter.
- close() - This function will close the current window or the named window.
- confirm("message") The string passed to the confirm function is displayed in the confirm dialog box.
- prompt("message","defaultmessage") - A prompt dialog box is displayed with the
- message passed as the prompt question or phrase.

BROWSER OBJECT/NAVIGATOR OBJECT

It is used to obtain information about client browser.

Properties

- appName- Returns Browser Name
- appVersion- Returns Browser Version
- appUserAgent- It Returns User Agent plugins- It will display Plugins.
- mimeTypes – It will Return Mime type supported by browser

FORM OBJECT:

Properties

- action - The action attribute of the Top of Form element
- length - Gives the number of form controls in the form
- method- The method attribute of the Top of Form element
- name - The name attribute of the Top of Form element

target - The target attribute of the Top of Form element

Methods

reset()- Resets all form elements to their

default values submit()- Submits the form

Properties of Form Elements

The following table lists the properties of form elements

- checked - Returns true when checked or false when not
- form - Returns a reference to the form in which it is part of length - Number of options in the <select> element.
- name - Accesses the name attribute of the element
- selectedIndex - Returns the index number of the current selected item value - selectedIndex attribute of the form element or the current input

STRING OBJECT:

String The string object allows you to deal with strings of text.

Properties

length - The number of characters in the string.

Methods:

- charAt(index) - Returns a string containing the character at the specified location.
- indexOf(pattern) - Returns -1 if the value is not found and returns the index of the first character of the first string matching the pattern in the string.
- indexOf(pattern, index) - Returns -1 if the value is not found and returns the index of the first character of the first string matching the pattern in the string. Searching begins at the index value in the string.
- lastIndexOf(pattern) - Returns -1 if the value is not found and returns the index of the first character of the last string matching the pattern in the string.
- lastIndexOf(pattern, index) - Returns -1 if the value is not found and returns the index of the first character of the last string matching the pattern in the string. Searching begins at the index value in the string.
- split(separator) - Splits a string into substrings based on the separator character.
- substr(start, length) - Returns the string starting at the "start" index of the string Continuing for the specified length of characters unless the end of the string is found first.
- substring(start, end) - Returns the string starting at the "start" index of the string and ending at "end" index location, less one.

- toLowerCase() - Returns a copy of the string with all characters in lower case.
- toUpperCase() - Returns a copy of the string with all characters in upper case.

DATE OBJECT:

The Date object is used to work with dates and times

- getDate() - Get the day of the month. It is returned as a value between 1 and 31. getDay() - Get the day of the week as a value from 0 to 6
- getHours() - The value returned is 0 through 23. getMinutes() - The value returned is 0 through 59.
- getMonth() - Returns the month from the date object as a value from 0 through 11. getSeconds() - The value returned is 0 through 59.
- getTime() - The number of milliseconds since January 1, 1970. getYear() - Returns the numeric four digit value of the year.
- setDate(value) - Set the day of the month in the date object as a value from 1 to 31. setHours(value) - Set the hours in the date object with a value of 0 through 59.
- setMinutes(value) - Set the minutes in the date object with a value of 0 through 59. setMonth(value) - Set the month in the date object as a value of 0 through 11.
- setSeconds(value) - Set the seconds in the date object with a value of 0 through 59.
- setTime(value) - Sets time on the basis of number of milliseconds since January 1, 1970.
- setYear(value) - Set the year in the date instance as a 4 digit numeric value.

EVENT HANDLING:

JavaScript is an Event Driven System

Event:

An Event is "any change that the user makes to the state of the browser"

There are 2 types of events that can be used to trigger script:

1. Window Events
2. User Events

- Window Events, which occurs when A page loads or unloads Focus is being moved to or away from a window or frame After a period of time has elapsed
- User Events, which occur when the user interacts with elements in the page using mouse or a keyboard.

Event Handlers:

Event handlers are Javascript functions which you associate with an HTML element as part of its definition in the HTML source code.

Syntax: <element attributes eventAttribute="handler">

Attribute	**Description**
Onblur	The input focus is moved from the object
Onchange	The value of a field in a form has been changes by the user by entering or deleting data
Onclick	Invoked when the user clicked on the object.
Ondblclick	Invoked when the user clicked twice on the object.
Onfocus	Input focus is given to an element

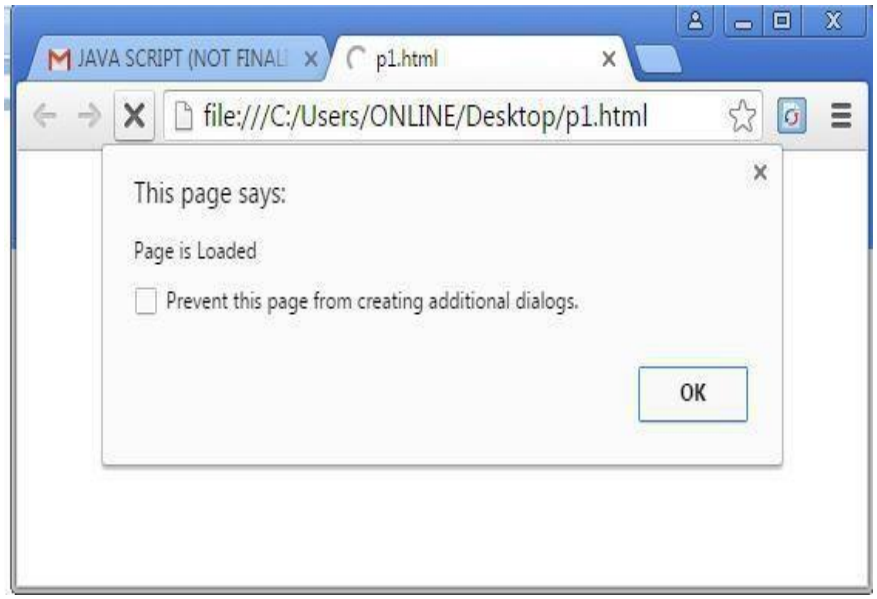
Onkeydown	Invoked when a key was pressed over an element.
Onkeypress	Invoked when a key was pressed over an element then released.
Onkeyup	Invoked when a key was released over an element.
Onload	When a page is loaded by the browser
Onmousedown	The cursor moved over the object and mouse/pointing device was pressed down.
Onmousemove	The cursor moved while hovering over an object.
Onmouseout	The cursor moved off the object
onmouseover	The cursor moved over the object (i.e. user hovers the mouse over the object).
Onmouseup	The mouse/pointing device was released after being pressed down.
Onmove	A window is moved, maximized or restored either by the user or by the script
Onresize	A window is resized by the user or by the script
onmousewheel	Invoked when the mouse wheel is being rotated.
Onreset	When a form is reset
Onselect	Invoked when some or all of the contents of an object is selected. For example, the user selected some text within a text field.
Onsubmit	User submitted a form.
Onunload	User leaves the Page

Examples:

1.
```
<html>
<head>
<script
    language="javascript">
    function fun()
    {
            alert("Page is Loaded");
    }
</script>
</head>
```

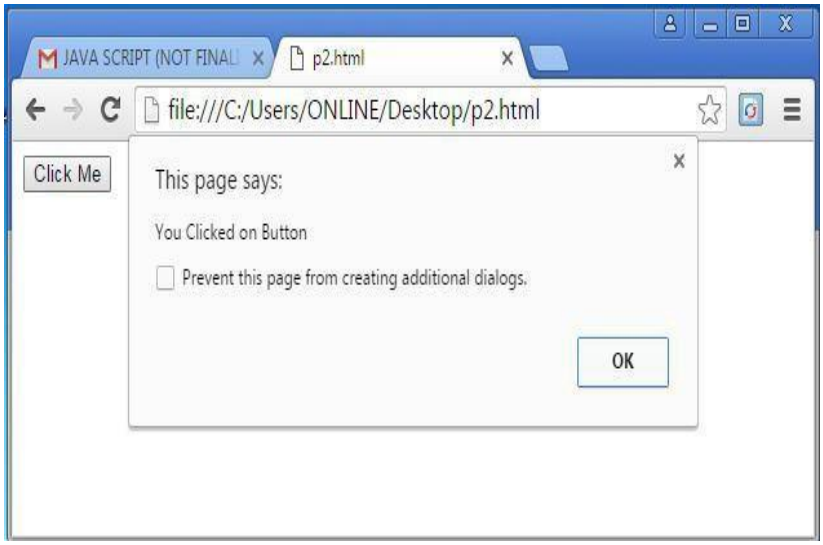
```
<body
onload="fun()">
</body>
</html>
```

Output:



2.
```
<html>
<head>
<script
    language="javascript">
    function fun()
    {
            alert("You Clicked on Button");
    }
</script>
</head>
<body>
    <input type="button" value="Click Me"
onClick="fun()"> </body>


</html>
```

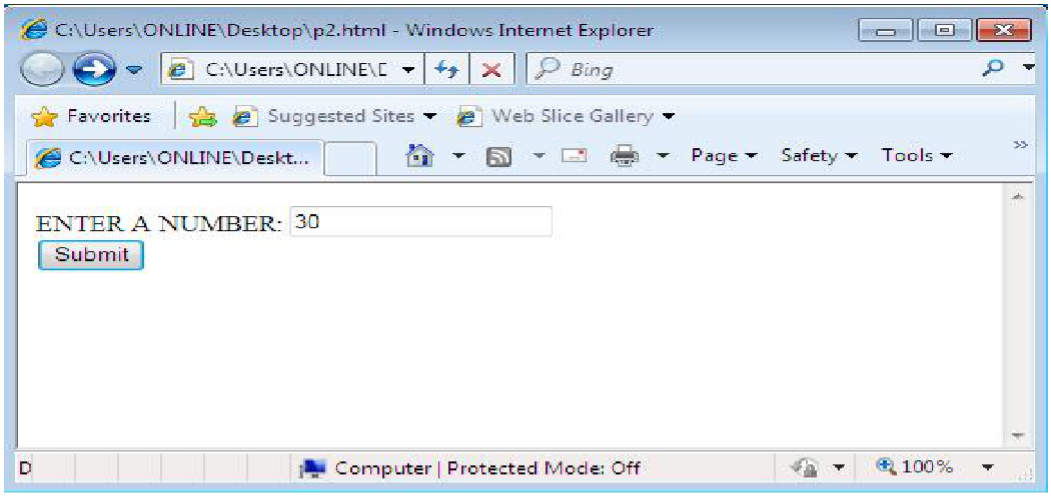
Output:

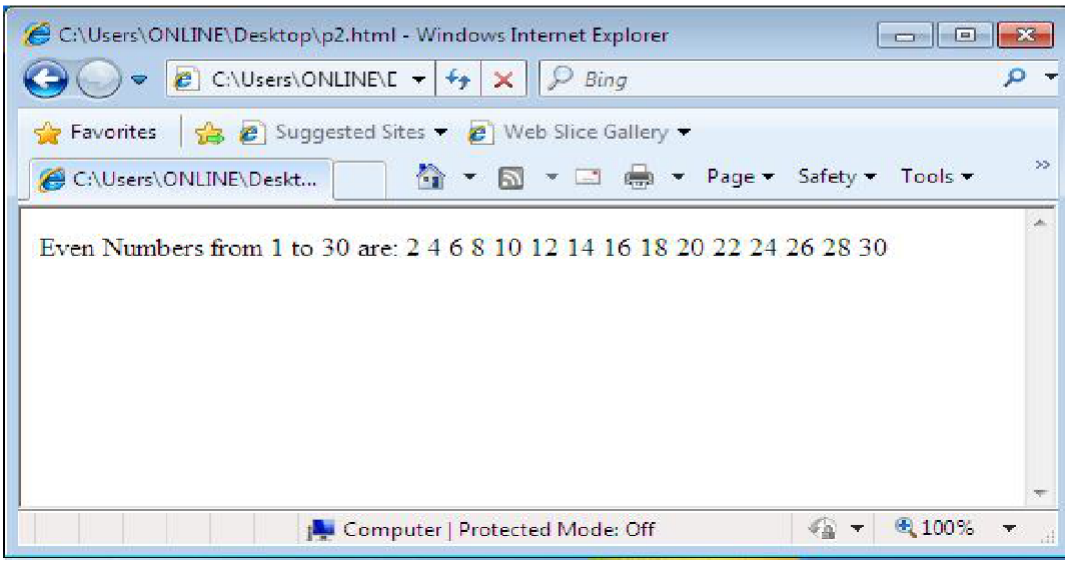


.

3.
```
<html>
<head>

<script
        language="javascript
        "> function fun1()
        {
            n=parseInt(f1.t1.value);
            document.writeln("Even Numbers from 1 to
            "+n+" are:"); for(i=1;i<=n;i++)
            {
                    if(i%2==0)
                            document.write(i+" ");
            }
        }
</script>
</head>
<body>
        <form name="f1" onSubmit="fun1()">
                <label>ENTER A NUMBER:</label>
                <input type="text" name="t1">
                <br>
                <input type="submit" value="Submit">
        </form>
</body>
</html>
```

Output:


C:\Users\ONLINE\Desktop\p2.html - Windows Internet Explorer
Bing
Favorites
Suggested Sites
Web Slice Gallery
C:\Users\ONLINE\Deskt...
Page
Safety
Tools
ENTER A NUMBER: 30
Submit
Computer | Protected Mode: Off
100%


C:\Users\ONLINE\Desktop\p2.html - Windows Internet Explorer
Bing
Favorites
Suggested Sites
Web Slice Gallery
C:\Users\ONLINE\Deskt...
Page
Safety
Tools
Even Numbers from 1 to 30 are: 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
Computer | Protected Mode: Off
100%

DHTML WITH JAVASCRIPT:

- It refers to the technique of making web pages dynamic by client-side scripting to manipulate the document content and presentation
- Web pages can be made more lively, dynamic or interactive by DHTML techniques. DHTML is **not** a markup language or a software tool.
- DHTML involves the following aspects.

HTML - For designing static web pages

JAVASCRIPT - For browser scripting

CSS (Cascading Style Sheets) - For style and presentation

control DOM(Document Object Model) - An API for scripts

to access and manipulate the web page as a document.

So, **DHTML = HTML + CSS + JAVASCRIPT + DOM**

HTML Vs DHTML

HTML	DHTML
1. It is used to create static web pages.	1. Used to create dynamic web pages.
2. Consists of simple HTML tags.	2. Made up of HTML tags+CSS+javascript+DOM
3. It is a markup language.	3. It is a technique to make web pages dynamic through client-side programming.
4. Do not allow to alter the text and graphics on the web page unless web page gets changed.	4. DHTML allows you to alter the text and graphics of the web page without changing the entire web page.
5. Creation of HTML web pages is simple.	5. Creation of DHTML web pages is complex.

6. Web pages are less interactive.	6. Web pages are more interactive.
7. HTML sites will be slow upon client-side technologies.	7. DHTML sites will be fast enough upon client-side technologies.

Example:

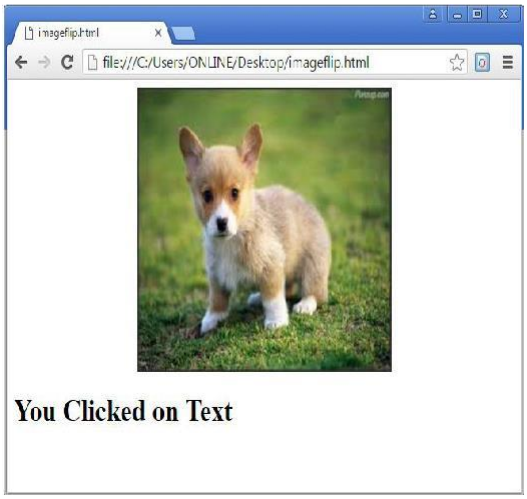
```
<html>
<head>
<script language="javascript">
        function img1()
        {
                i1.src="image2.jpg";
        }

        function img2()
        {



                i1.src="image1.jpg";
        }
        function fun1()
        {
                h11.innerText="You Clicked on Text";
        }
</script>
</head>
<body>
        <center>
                <img src="image1.jpg" id="i1" width="320" height="290"
                        onmouseover="img1()" onmouseout="img2()">
        </center>
        <h1 id="h11" onclick="fun1()">Click on
Text</h1> </body>
</html>
```

Output:


imageflip.html
Click on Text


imageflip.html
file:///C:/Users/ONLINE/Desktop/imageflip.html
Click on Text


imageflip.html
file:///C:/Users/ONLINE/Desktop/imageflip.html
Click on Text


imageflip.html
file:///C:/Users/ONLINE/Desktop/imageflip.html
You Clicked on Text

VALIDATIONS using DHTML

- Everyone must have filled an online form at some stage, a form usually asks for information related to name, phone no, address, credit-card no etc.
- Incase you didn't provide information in the format specified by the form field or leave it empty, the message will appear and form cannot be submitted. This is done using a Javascript program on the client side, the Javascript program uses a regular expression pattern to test the input of each form field.

Regular Expression

A regular expression is a sequence of characters that forms a search pattern.

- When you search for data in a text, you can use this search pattern to describe what you are searching for.
- A regular expression can be a single character, or a more complicated pattern.
- Regular expressions can be used to perform all types of text search and text replace operations.
- /n,/r,/t match literal newline, carraige return, tab
- \\, \/, * match a special character literally, ignoring or escaping its special meaning
- […] Match any one character between the brackets
- [^…] Match any one character not between the brackets
- . Match any character other than the newline
- \w, \W Match any word\non-word character
- \s, \S Match any whitespace/non-whitespace
- \d, \D Match any digit/non-digit
- ^, $ require match at beginning/end of a string or in multi-line mode, beginning/end of a line
- \b, \B require a match at a word/non-word boundary
- ? Optional term : Match zero or one time

- + Match previous term one or more times
- * Match term zero or more times
- {n} Match pervious term n times
- {n,} Match previous term n or more times
- {n,m} Match prev term at least n time but no more than m times
- a | b Match either a or b
- (sub) Group sup-expression *sub* into a single term and remember the text that it matched
- \n Match exactly the same chars that were matched by sup-expression number n
- $n In replacement strings, substitute the text that matched the nth sub-expression

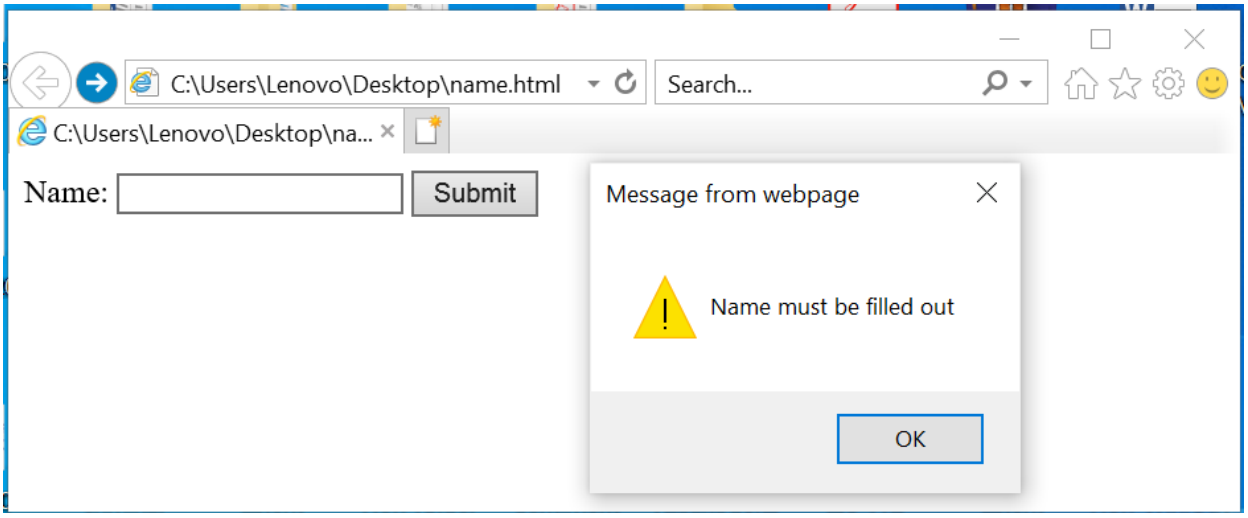
A simple six digit zipcode can be checked using regular expression which matches exactly six digits : /^d{6}$/ or /^[0-9][0-9][0-9][0-9][0-9][0-9]$/
Phone number var phoneno = /^\d{10}$/

Example: validation for fill name

```
<html>
<html>
<head>
<script>
function validateForm() {
  var x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
</script>
</head>
<body>

<form name="myForm" action="/action_page.php" onsubmit="return validateForm()"
method="post">
```

```
  Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>

</body>
</html>
```

Output:



Example: email validation

```
<html>
<head>
<script type="text/javascript">
function fun()
{
var a=f1.t1.value;

b=/^[a-zA-Z0-9]+([\.-]?[a-zA-Z0-9]+)*@\w+([\.-]?(\w{2,3}))?(\.\w{2,3})$/;
if(b.test(a)==true)
{
alert("valid email-id");
}
else
{
alert("invalid email-id");
}
}
```

```
</script>
</head>
<body>
<form name="f1">
<input type="text" name="t1">
<input type="button" value="validate" onclick="fun()">
</form>
</body>
</html>
```

Output: