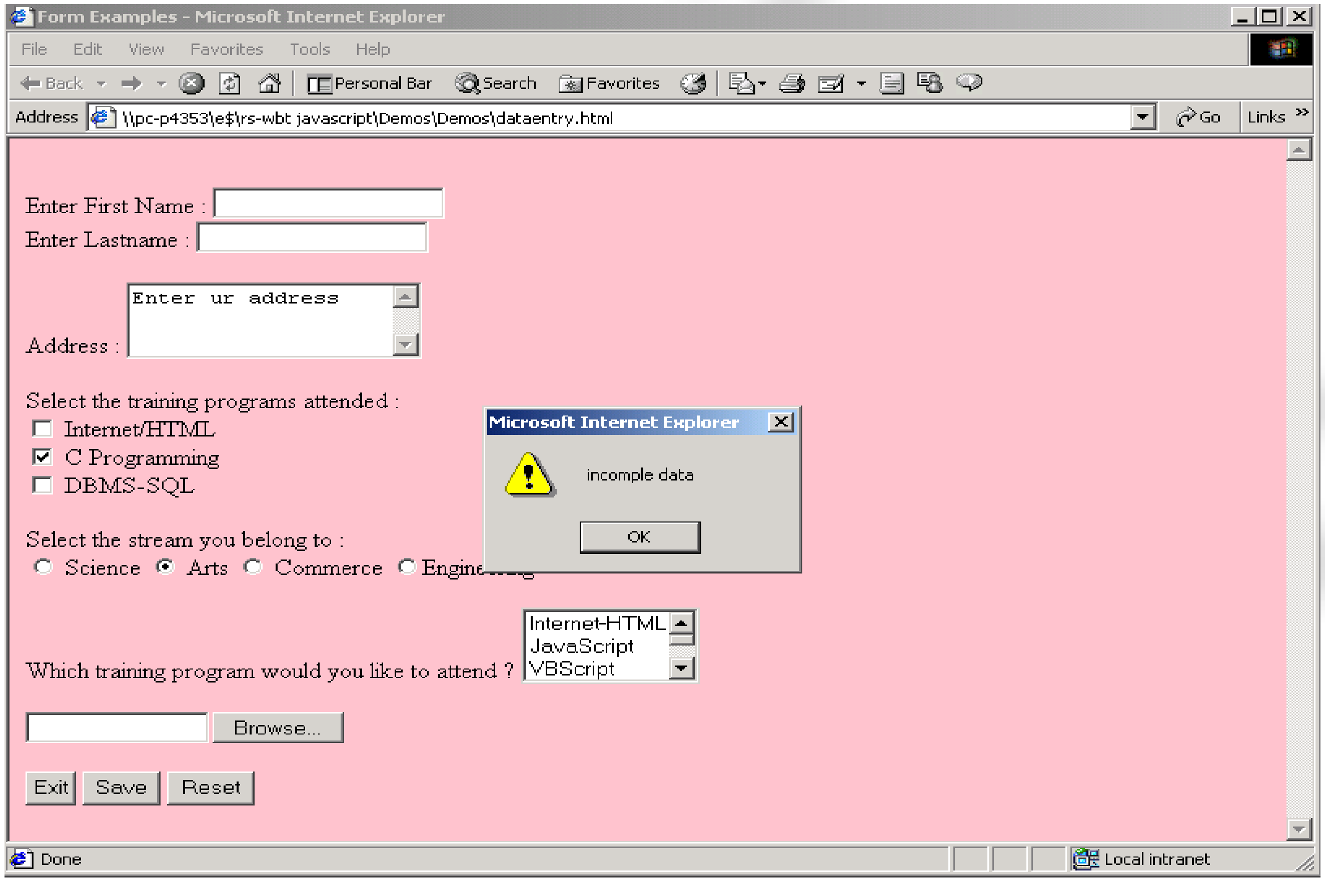
JavaScript

Introduction to JavaScript

Lesson Objectives

- To understand the following topics:
  - Basic Concepts of JavaScript
  - JavaScript and Java
  - Embedding JavaScript in HTML

Basic Concepts of JavaScript


Form Examples - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Personal Bar Search Favorites
Address \\pc-p4353\e$\rs-wbt javascript\Demos\Demos\dataentry.html
Go Links

Enter First Name :

Enter Lastname :

Address : Enter ur address

Select the training programs attended :

☐ Internet/HTML
☑ C Programming
☐ DBMS-SQL

Select the stream you belong to :

○ Science ● Arts ○ Commerce ○ Engineering

Which training program would you like to attend ? Internet-HTML / JavaScript / VBScript

Browse...

Exit Save Reset

Microsoft Internet Explorer

incomple data

OK

What is JavaScript ?

- JavaScript is Netscape's cross-platform, object-based scripting language.
- It is lines of executable computer code that can be inserted into a HTML page.
- It is a lightweight programming language.
- *Client-side JavaScript* extends the core language by supplying objects to control a browser and its Document Object Model.
- *Server-side JavaScript* extends the core language by supplying objects relevant to running JavaScript on a server.

JavaScript and Java

JavaScript	Java
▶ Interpreted	▶ Compiled (bytecodes) and interpreted
▶ Object based	▶ Object oriented
▶ Codes embedded in HTML	▶ Applets distinct from HTML
▶ Variable data types not declared	▶ Variable data types declared

Embedding JavaScript in HTML

- The <SCRIPT> tag

```
<SCRIPT>
        JavaScript statements …
</SCRIPT>
```

- Ending statements with a semicolon?

- Specifying the JavaScript version

<SCRIPT LANGUAGE="JavaScript1.2">

Embedding JavaScript in HTML

- The <SCRIPT> tag

```
<SCRIPT>
        JavaScript statements …
</SCRIPT>
```

- Ending statements with a semicolon?

- Specifying the JavaScript version

```
<SCRIPT LANGUAGE="JavaScript1.2">
```

Embedding JavaScript in HTML (Contd.)

- Hiding Scripts with Comment tags

```
<script type="text/javascript">
        <!--
                    some statements …
        // -->
</script>
```

- Specifying a File of JavaScript code

```
<SCRIPT SRC="common.js"></SCRIPT>
```

Embedding JavaScript in HTML (Contd.)

- Using Quotation Marks

```
document.write(“<A HREF=„A.HTML‟>Link to next page”)
```

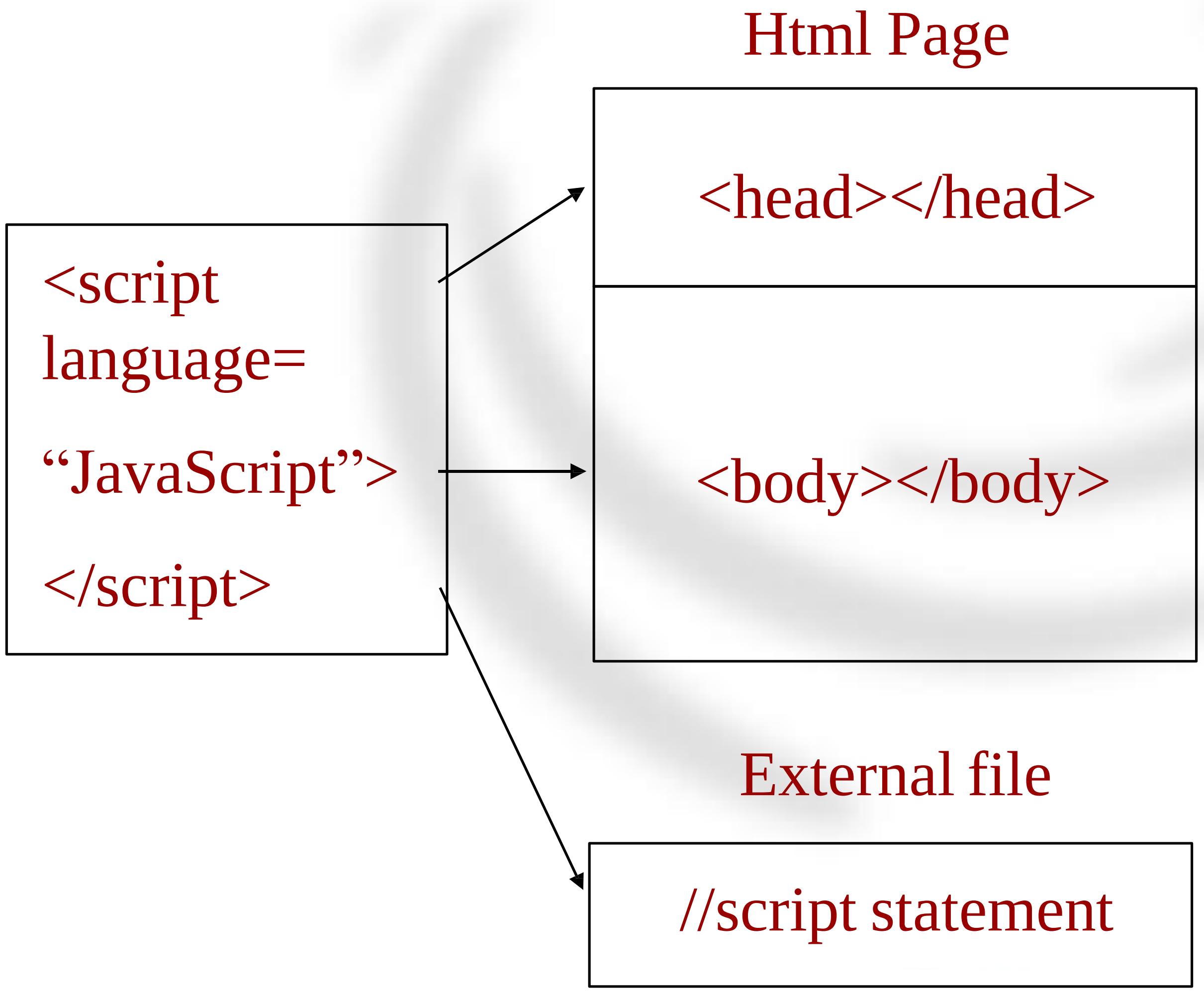
Embedding JavaScript in HTML (Contd.)

- Specifying alternate content with the NOSCRIPT tag

```
<NOSCRIPT>
        Your browser has JavaScript turned off.
</NOSCRIPT>
```

Where to write JavaScript ?

- Head Section
- Body Section
- External File



JavaScript in Head Section

```
<HTML>
<HEAD>
<TITLE>Script tag  in Head Section</TITLE>
<SCRIPT language="Javascript">
<!--
   document.write("<h1>message displayed due to script in head</h1>")
- ->
</ SCRIPT >
</HEAD>
<BODY>
</BODY>
</HTML>
```

JavaScript in Body Section

```
<HTML>
<HEAD>
<TITLE>Script tag in Body</TITLE>
</HEAD>
<BODY >
<SCRIPT language=“Javascript">
document.write("<h1>message displayed due to script in body</h1>")
</SCRIPT>
</BODY>
</HTML>
```

JavaScript in External File

```
<HTML>
<HEAD>
<TITLE>script tag in external file</TITLE>
<SCRIPT src="common.js">
<!– No javascript statements can be written here→
</ SCRIPT>
</HEAD>
<BODY>
< SCRIPT>
document.write("Display value of a variable”+msg)
</ SCRIPT >
</BODY>
</HTML>
```

External js File

```
var msg

msg=“<h1>declared in external js file</h1>"
```

Contents of Common.js

Demo

- Hello.html
- Head_section.html
- Extern_file.html
- Comm.js
- Var_ex.html
- Confirm_ex.html

JavaScript

The JavaScript Language

Lesson Objectives

- To understand the following topics:
  - Data Types and Variables
  - JavaScript Operators
  - Control Structures and Loops
  - JavaScript Functions

Overview

- JavaScript Language:
  - ➢ Data Types and Variables
  - ➢ JavaScript Operators and Expressions
  - ➢ String Operator
  - ➢ Control Structures and Looping
  - ➢ Functions
  - ➢ Using the arguments Array
  - ➢ Predefined Functions
  - ➢ Using Global and Local Variables
  - ➢ Summary

Data Types in JavaScript

- JavaScript is a free-form language. You do not have to declare all variables, classes, and methods.
- Data Types in JavaScript are:
  - Number (4.156, 39)
  - String (“This is JavaScript”)
  - Boolean (true or false)
  - Null (null)

Data Types in JavaScript (Contd..)

- JavaScript variables are said to be loosely typed
- Defining variables: var variableName = value
- Rules when choosing a variable name:
  - Can include letters of the alphabet, digits 0-9 and the underscore (_) character and is case-sensitive.
  - Cannot include spaces or any other punctuation characters.
  - First character of the variable name must be either a letter or the underscore character.
  - No official limit on the length of a variable name, but must fit within a line.

Data Types in JavaScript (Contd..)

- Scope of variables

```
<script language="Javascript">  var
companyName="mycompany"
function f(){
var employeeName="Tom"
document.write("Welcome to "+companyName+",
   "+employeeName)
 }
</script>
```

Global Variable

Local Variable

JavaScript Operators : Arithmetic

Operator	Description	Example	Result
▶ +	Addition	2 + 2	4
▶ -	Subtraction	5 – 2	3
▶ *	Multiplication	4 * 5	20
▶ /	Division	5 / 2	2.5
▶ %	Modulus	10 % 8	2
▶ ++	Increment	x = 5; x++	x = 6
▶ --	Decrement	x = 5; x--	x = 4

JavaScript Operators : Comparison

Operator	Description	Example	Result
▸ ==	is equal to	5 == 8	false
▸ !=	is not equal	5 != 8	true
▸ >	is greater than	5 > 8	false
▸ <	is less than	5 <= 8	true
▸ >=	is greater or equal	5 >= 8	false
▸ <=	is less or equal	5 <= 8	true

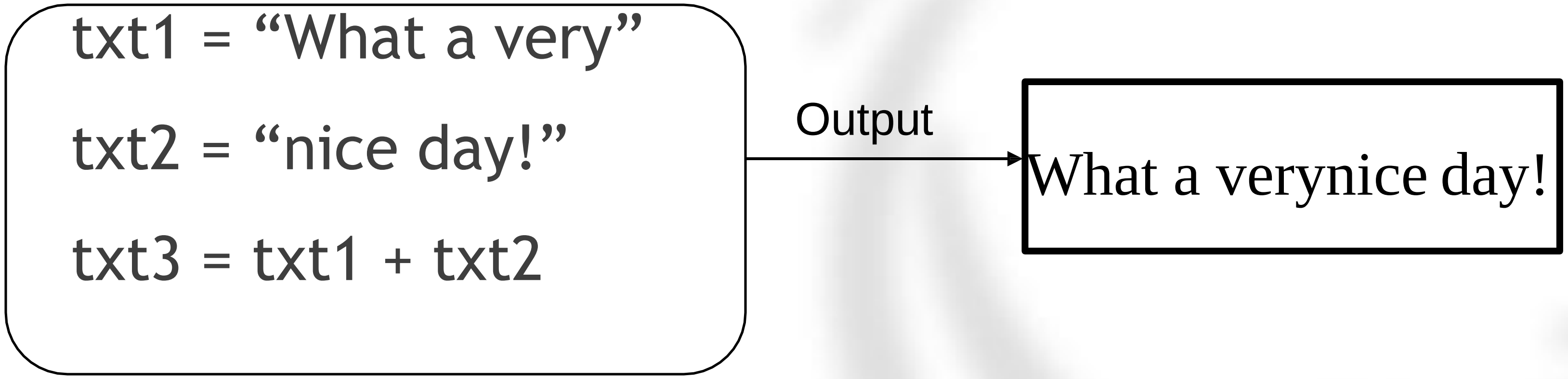
JavaScript Operators : Assignment

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

JavaScript Operators : Logical

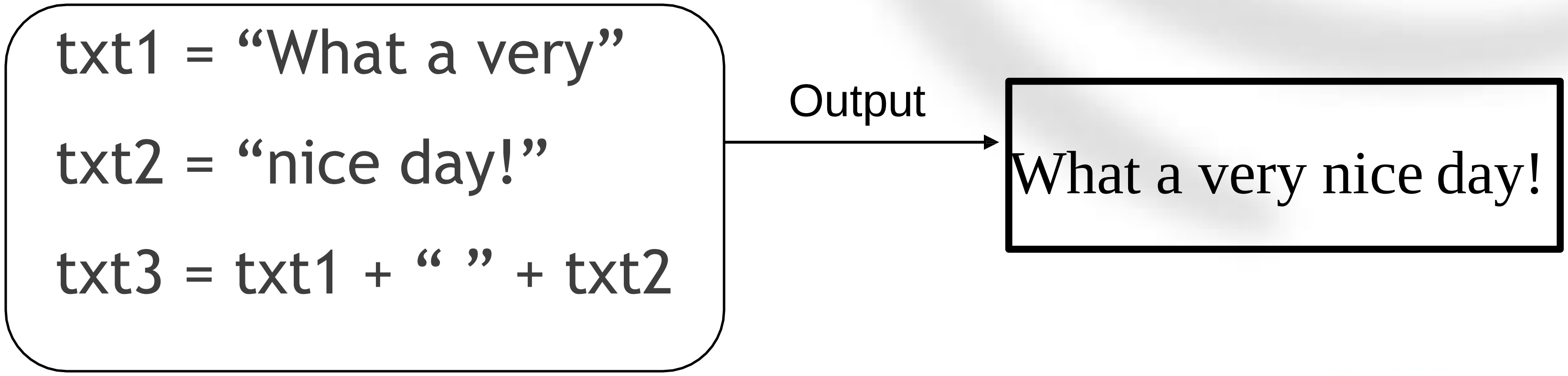
Operator	Description	Example
▶ &&	and	x = 6; y = 3 x < 10 && y > 1 returns true
▶ ||	or	x = 6; y = 3 x < 10 || y > 5 returns true
▶ !	not	x = false !x returns true

String Operator (+)



```
txt1 = “What a very”
txt2 = “nice day!”
txt3 = txt1 + txt2
```

```
What a verynice day!
```



```
txt1 = “What a very”
txt2 = “nice day!”
txt3 = txt1 + “ ” + txt2
```

```
What a very nice day!
```

typeof Operator

▸ typeof	undefinedvariable	“undefined”
▸ typeof	33	“number”
▸ typeof	“abcdef”	“string”
▸ typeof	true	“boolean”
▸ typeof	null	“object”

Demo

- Typeof_ex.html

Control Structures and Loops

- JavaScript supports the usual control structures:
  - the conditionals: if, if...else, and switch;
  - the iterations: for, while, break, and continue

The if Statement

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

```
if(a>10) {
document.write(“Greater than
10”)
} else {
document.write(“Less than 10”)
}
```

Shorthand

```
document.write( (a>10) ? “Greater than 10” : “Less than 10”
    );
```

The Switch Statement

```
switch (variable) {
      case outcome1 :{
//stmts for outcome 1 }
      case outcome2 :{
 //stmts outcome 2 }
  …
    default: {
 //none of the outcomes
 is chosen }
```

```
switch (day) {
  case "Monday" : {
document.write("weekday")
break;}
 case "Saturday": {
document.write("weekday")
break}
  …
 default: {
document.write("Invalid day of the week")
}
```

The for and while Statements

```
for( [initial expression;][condition;][increment expression] ) {
        statements
}
```

```
for(var i=0;i<10;i++){
document.write(“Hello”);}
```

```
while(condition) {
    statements
}
```

```
while(i<10) {
   document.write(“Hello”);
   i++;}
```

The for and while Statements (contd..)

```
while(condition) {
        statements
}
```

The Break and Continue Statements

- Break
  - Writing break inside a switch, for, while control structure will cause the program to jump to the end of the block. Control resumes after the block, as if the block had finished.
- Continue
  - Writing continue inside a loop will cause the program to jump to the test condition of the structure and re-evaluate and perform instruction of the loop. Control resumes at the next iteration of the loop.

Demo -for loop

- For_ex.html

The Function Statement

- The function statement

```
function myFunction (arg1, arg2, arg3) {
      statements
      return }
```

$\leftarrow$ The return keyword returns a value.

- How to call a function

```
myFunction( “abc”, “xyz”, 4 )
            or
 myFunction()
```

The Function Statement (Contd..)

- Using the arguments array:

```
arguments[i]

functionName.arguments[i]
```

i - ordinal number of the argument starting at zero

arguments.length - Total number of arguments

The Function Statement (Contd..)

```
function myConcat(separator) {
    result = “”
    for(var i=1; i<arguments.length;i++) {
            result += arguments[i] + separator
    }
    return result
}
```

myConcat(“,” , “red” , “orange” , “blue”)

// returns “red, orange, blue”

Predefined Functions

- **eval:**

  Evaluates a string of JavaScript code without reference to a particular object.

```
eval (expr)
      where expr is a string to be evaluated
```

- **isFinite:**

  Evaluates an argument to determine whether it is a finite number.

Predefined Functions (Contd..)

isFinite (number)
    where number is the number to evaluate

- isNaN :

Evaluates an argument to determine if it is “NaN” (not a number)

isNaN (testValue)
    where testValue is the value you want to evaluate

Predefined Functions (Contd..)

- **parseInt** and **parseFloat:**

  Returns a numeric value for string argument.

```
parseInt (str)

parseFloat (str)
```

```
parseInt(str, radix)
```

returns an integer of the specified radix of the string argument

Predefined Functions (Contd..)

- **Number** and **String** :

  Convert an object to a number or a string.

```
Number (objectReference)
String (objectReference)
```

```
D = new Date (430054663215)
x = String(D)
// returns "Thu Aug 18 04:37:43 GMT-0700 (PDT) 1983"
```

Global and Local Variables

- Variables that exist only inside a function are called **Local variables.**
- The values of such *Local variables* cannot be changed by the main code or other functions.
- Variables that exist throughout the script are called **Global variables.**
- Their values can be changed anytime in the code and even by other functions.

Demo

- If_ex.html
- Switch_ex.html
- Break_con_ex.html
- Fun_ex.html
- Num_string_fun.html

JavaScript

Arrays

Lesson Objectives

- The above tasks will be learnt under the following topics in this lesson:
  - Creating an Empty Array
  - Populating an Array
  - Deleting Arrays and Array Entries
  - Array Object Properties
  - Array Object Methods

Concept of Array Objects

- An **array** is the sole JavaScript data structure provided for storing and manipulating ordered collections of data.
- For creating an empty array, you can use the following:

```
var myArray = new Array()
var myCDCollection = new Array(500)
myCDCollection [700] = “Gloria Estefan/Destiny”
collectionSize = myCDCollection.length // result = 701
```

Concept of Populating an Array

- Populating an array:

```
solarSys = new Array(2)
solarSys[0]  =  "Mercury"
solarSys[1] = "Venus“

solarSys = new Array(“Mercury”, “Venus”, “Earth”, “Mars”,
“Jupiter”, “Saturn”, “Uranus”, “Neptune”, “Pluto”)
```

```
onePlanet = solarSys[4] // result = “Jupiter”
```

Concept of Populating an Array

```
earth = new Array()
earth.diameter = "7920 miles"
earth.distance = "93 million miles"
earth.year = "365.25 days"
earth.day = "24 hours"
earth.length // result = 4
```

```
earth.diameter // result = “7920 miles”
earth[“diameter”] // result = “7920 miles”
earth[0] // result = null
```

Concept of Deleting an Array Entry

- Deleting an array element eliminates the index from the list of accessible index values.
- This does not reduce the array‟s length, as in the given sequence of statements.

```
myArray.length// result: 5
delete myArray[2]
myArray.length// result: 5
myArray[2] // result: undefined
```

Concept of Array Object Methods

- JavaScript provides the following array object methods:
  - arrayObject.reverse()
  - arrayObject.slice(startIndex, [endIndex])
  - arrayObject.join(separatorString)
- The code snippet here shows the usage of join method.
  - In this, myArray contents will be joined and placed into arrayText by using the comma separator"

```
var arrayText = myArray.join(“,”)
```

Concept of Array Object Methods

- arrayObject.sort([compareFunction])

```
myArray = new Array(12, 5, 200, 80)
```

```
function
compare(a,b) {
return a - b
}
myArray.sort(compar
e)
```

```
function compare(a,b) {
// last character of array strings
var aComp = a.charAt(a.length
- 1)
var bComp = b.charAt(b.length
- 1)
if (aComp < bComp) {return -1}
if (aComp > bComp) {return 1}
return 0
}
```

JavaScript

Working with Objects

Lesson Objectives

- To understand the following topics:
  - ➢ Object and Properties
  - ➢ Creating New Objects
  - ➢ Creating New Objects: An Example
  - ➢ Deleting Objects

Overview

- Working with Objects
  - ➢ Objects and Properties
  - ➢ Creating New Objects
  - ➢ Defining Properties for an Object Type
  - ➢ Using this for Object References
  - ➢ Defining Methods for an Object Type
  - ➢ A Complete Example
  - ➢ Deleting Objects
  - ➢ Summary

Working with Objects

- JavaScript is designed on a simple object-based paradigm.
- An object is a construct with properties that are JavaScript variables or other objects.
- An object has functions associated with it that are known as the object's *methods*.
- In addition to predefined objects in JavaScript, you can define your own objects.

Creating New Objects

- Using Object Initializers

```
objName = {property1:value1, property2:value2, … }
```

```
myHonda = {color:“red”, wheels:4, engine:{cylinders:4,
size:2}}
```

Creating New Objects (Contd.)

- Using Constructors
  - ➢ Define the object type by writing a constructor function.
  - ➢ Create an instance of the object with new.

```
function car(make, model, year) {
            this.make = make
            this.model = model
            this.year = year
}
```

```
mycar = new car(
            "Eagle" ,
            "Talon Tsi" ,
            1993)
```

Creating New Objects (Contd..)

```
Function person(name, age) {
        this.name = name
        this.age = age
}
```

```
ken = new person( “Ken” ,
        33 )
```

```
function car(make, model, year, owner) {
        this.make = make
        this.model = model
        this.year = year
        this.owner = owner
}
```

```
car1 = new car( “Mazda” , “Miata”, 1990, ken )
```

Creating New Objects (Contd..)

- Accessing properties

```
car1.year=2000
document.write(car1.model)
document.write(car1.owner.name)
```

```
car1.color = "black"
```

- Adding properties to a previously defined object

Creating New Objects (Contd.)

- Defining properties for an object type:

```
car.prototype.color = null
car1.color = “black”
```

- Defining methods:

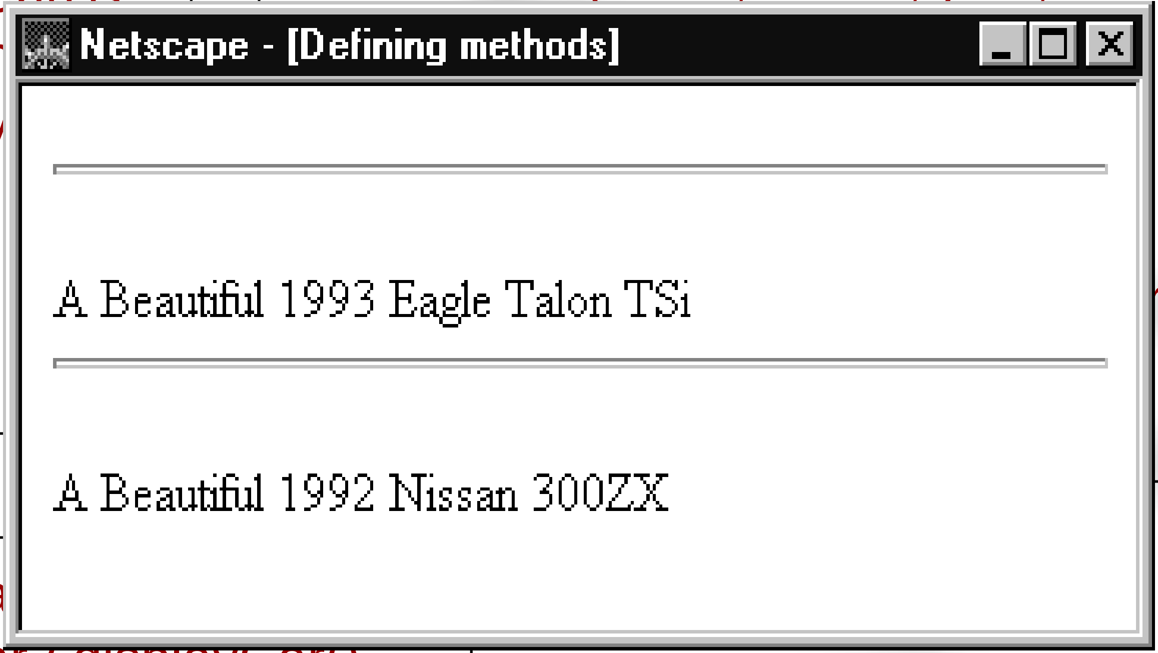
```
obj.methodName = function_name
obj.methodName(params)
```

Creating New Objects (Contd.)

```
function displayCar() {
document.writel
beautiful" + this.y
+ " " + this.make
+ this.model
 }
```

```
function car(make, model, year, owner) {
```

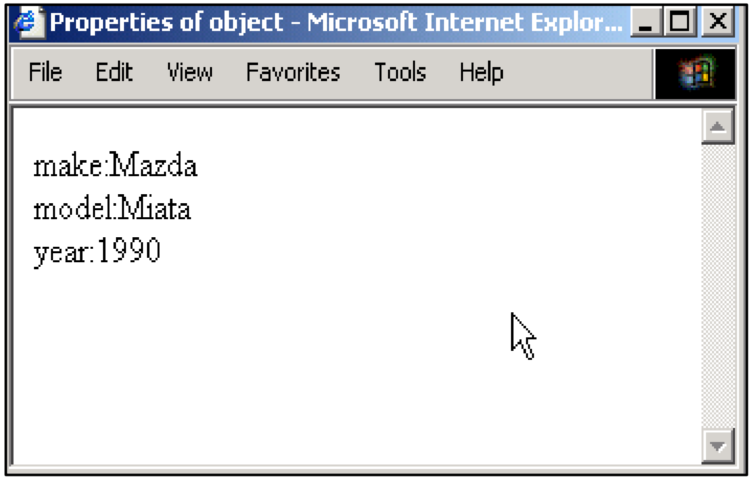
```
ca
car2.displayCar()
```



```
A Beautiful 1993 Eagle Talon TSi

A Beautiful 1992 Nissan 300ZX
```

Object Properties: An Example


Properties of object - Microsoft Internet Explor...
File Edit View Favorites Tools Help

```
make:Mazda
model:Miata
year:1990
```

```
myobj=new
car("Mazda","Miata",1990)
for (var i in myobj) {
        prop   =   i   +   ":"   +
myobj[i]+"<BR>"
        document.write(prop)
}
```

Creating Objects: Using „with" Keyword

- **with object:**

```
with (objectName)
{ statement }
```

```
with Math
{
          x = PI * x
          y = x * sin(PI)
}
```

Deleting Objects

- You can remove an object by using the delete operator.

```
mobs=new Car()
delete myobj    // removes the object and returns true
```

Demo

- Complete_ex.html
- Instance_obj.html
- Temp_obj.html

JavaScript

Document Object Model

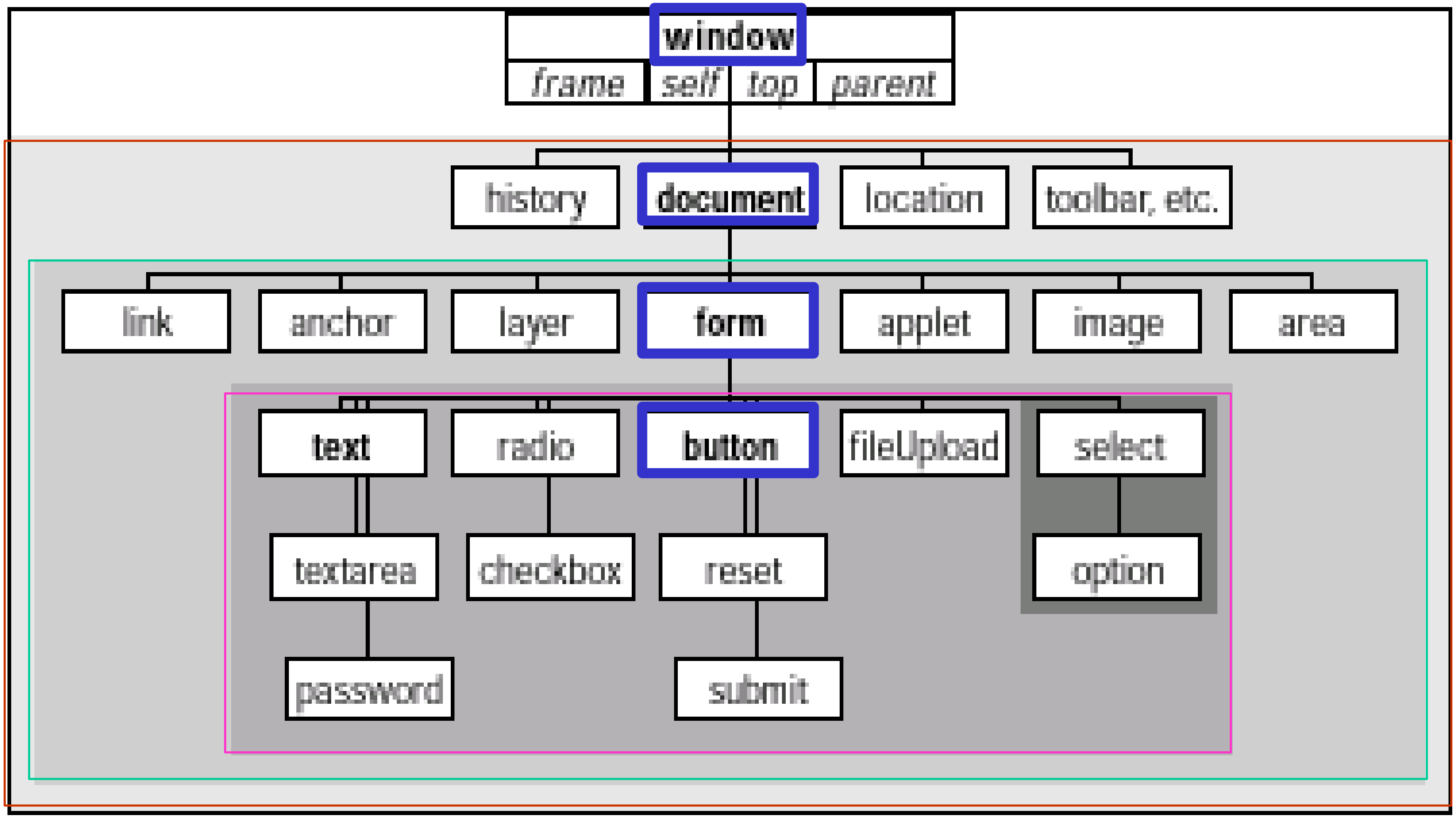
Lesson Objectives

- After completing this module you will be able to:
  - ➢ Understand the JavaScript Object Model.
  - ➢ Understand the *Window* object, it"s properties and methods.
  - ➢ Understand the *Frame* object, it"s properties and methods.

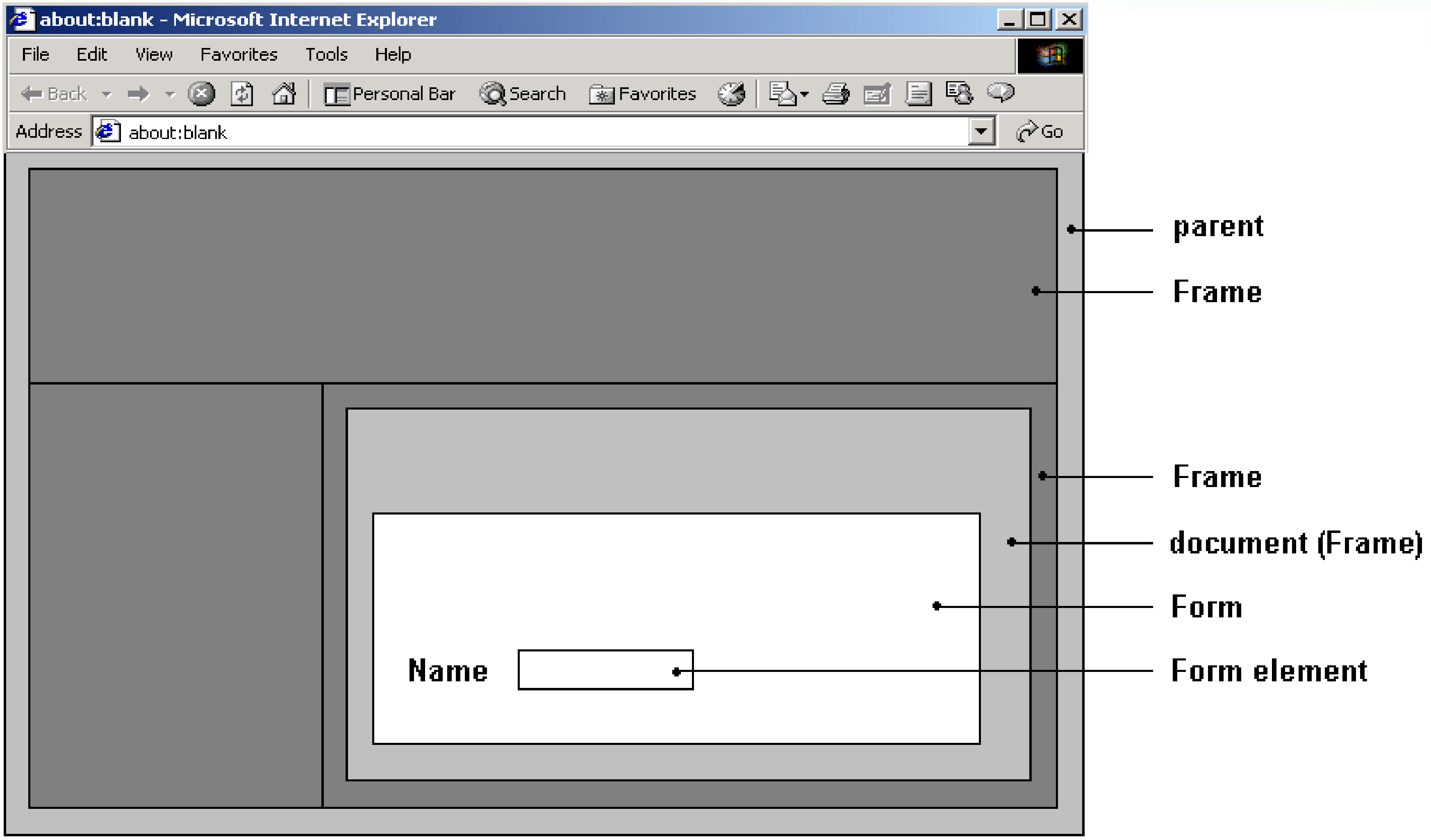
Module Coverage

- Topics covered in this module are:
  - JavaScript Document Object Model
    - Object Properties and Event Handlers
  - Working with the *Window* Object
  - Working with the *Frame* Object

JavaScript Document Object Model



JavaScript Document Object Model


about:blank - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Personal Bar Search Favorites
Address about:blank Go
parent
Frame
Frame
document (Frame)
Form
Name
Form element

Object Properties

- Define a particular, current setting of an object.
- Property names are case-sensitive.
- Each property determines it"s own read-write status.
- Any property you set survives as long as the document remains loaded in the window.
- For example:

```
document.forms[0].phone.value = “555-1212”

document.forms[0].phone.delimiter = “-”
```

Object Methods

- Command the script gives to that object.
- Some methods return values, but that is not a prerequisite.
- Predefined by the object model
  - Assign additional methods to an existing object.

Event Handlers

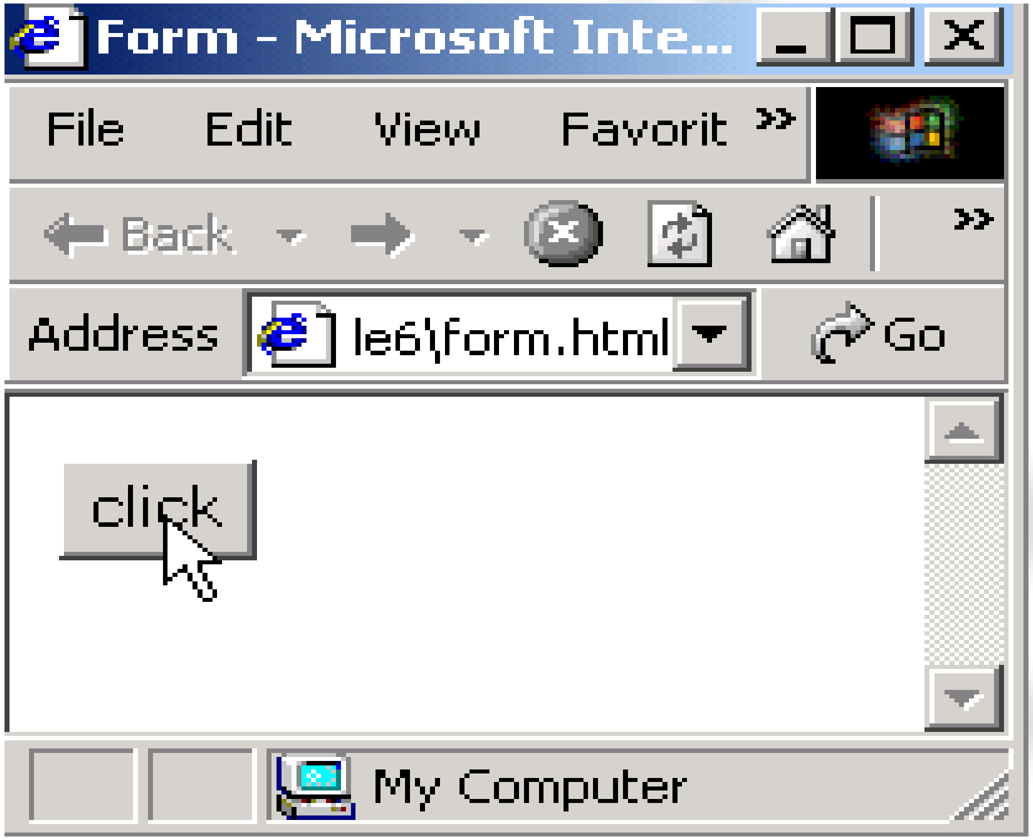
- Specify how an object reacts to an event.
  - Event can be triggered by a user action or a browser action.
  - In the earliest JavaScript-enabled browser, event handlers were defined inside HTML tags as extra attributes.
- Event handlers as methods:

```
document.formName.button1.onclick=f1()
```

- Event handlers as properties:

```
<INPUT TYPE=”button” NAME=”button1” onClick=”f1()”>
```

Event Handlers (Contd..)



Working with Window Object

- Window object:
  - Unique position at the top of the JavaScript object hierarchy.
    - Exalted location gives *window* object a number of properties and behaviors unlike other objects.
  - Can be omitted from object references.
    - Since everything takes place in a window.

Window Object Properties

- *defaultStatus* and *status*

```
window.defaultStatus="Javascript Examples"
```

- *parent*
- *frames*

```
parent.frames.length                                parent.frames[0]
```

- *onerror*

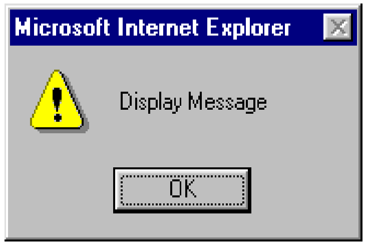
```
window.onerror=null
```

- opener

Window Object Methods

- alert(message)

```
window.alert("Display Message")
```

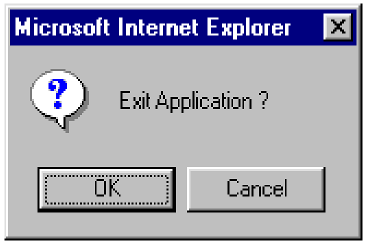


Microsoft Internet Explorer

Display Message

- confirm(message)

```
window.confirm("Exit Application ?")
```



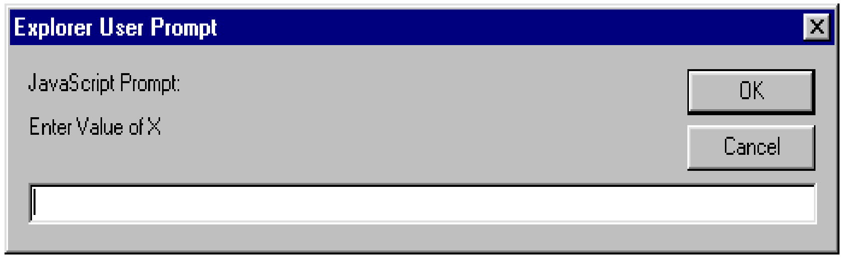
Microsoft Internet Explorer

Exit Application ?

- prompt(message,[defaultReply])

```
var input=

window.prompt("Enter value of X")
```



Explorer User Prompt

JavaScript Prompt:

Enter Value of X

Window Object Methods

- open(**“URL”, “windowName”**[, “windowFeatures”])

```
newwin=window.open(“new/URL”,”NewWindow”,

   “toolbar,status,resizable”)
```

- close()
- moveBy(deltaX,deltaY), moveTo(x,y)
- resizeBy(deltaX,deltaY),

  resizeTo(outerwidth,outerheight)

- **scrollBy(deltaX,deltaY), scrollTo(x,y)**

Frame Object

- Properties, methods and event handlers are same as the window object.
- Behaves exactly like a window object, except that it is created as part of a frameset by another document.
- Event Handlers:

▶ OnBlur	▶ onDragDrop	▶ onMove	▶ onUnload
▶ OnFocus	▶ onLoad	▶ onResize	

Demo

- Window_object.html
- setTimeOut_method.html
- Window_ex.html
- setInterval_method.html