- What is JavaScript?

  JavaScript was designed to add interactivity to HTML pages

  JavaScript is a scripting language (a scripting language is a lightweight programming language)

  A JavaScript consists of lines of executable computer code A 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

- Are Java and JavaScript the Same?

  NO! Java and JavaScript are two completely different languages in both concept and design!

  Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

- What can a JavaScript Do? / Application of JavaScript

  JavaScript gives HTML designers a programming tool - HTML authors are normally not programmers, but 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 react to events - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element

  JavaScript can read and write HTML elements - A JavaScript can read and change the content of an HTML element

  JavaScript can be used to validate data - A JavaScript can be used to validate form data before it is submitted to a server, this will save the server from extra processing

  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 Put a JavaScript Into an HTML Page

To insert a JavaScript into an HTML page, we use the <script> tag (also use the

type attribute to define the scripting language).

So, the <script type="javascript"> and </script> tells where the JavaScript starts and ends:

```
<script type="javascript">
...
</script>
```

The word document.write is a standard JavaScript command for writing output to a page.

By entering the document.write command between the <script type="javascript"> and </script> tags, the browser will recognize it as a JavaScript command and execute the code line.

For E.g. :

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

The code above will produce this output on an HTML page:

Hello World!

Note: If we had not entered the <script> tag, the browser would have treated the document.write ("Hello World!") command as pure text, and just write the entire line on the page.

❖ Some important things to know when scripting with JavaScript.

JavaScript is Case Sensitive

- A function named "myfunction" is not the same as "myFunction" and a variable named "myVar" is not the same as "myvar".
- JavaScript is case sensitive - therefore watch your capitalization closely when you create or call variables, objects and functions.

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

White Space

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

```
name="A
nil"
name =
"Anil"
```

Break up a Code Line

- You can break up a code line within a text string with a backslash. The example below will be displayed properly:

```
document.write("H
ello \ World!")
```

- However, you cannot break up a code line like this:

```
document.w
rite \
("Hello
World!")
```

The example above will generate an error!

Insert Special Characters

- You can also insert special characters (like " ' ; &) with a backslash:

```
document.write ("You \& I sing \"Happy
Birthday\".") output: You & I sing "Happy
Birthday".
```

Comments

- Start a comment with two slashes "//":

```
E.g. : sum=a + b   //this is a comment
```

- Using /* and */ to create a multi-line comment:

```
E.g. /* This is a
        comment
        block. It
        contains
        several
        lines*/
```

Ending Statements With a Semicolon

- In JavaScript semicolons are optional!
- However, semicolons are required if you want to put more than one statement on a single line.

E.g. semicolons not required here

```
document.write ("Hello")
document.write ("Welcome To RKU ")
```

E.g. semicolons required here

document.write (“Hello”) ; document.write (“Welcome To RKU”)

❖ Where to Put the JavaScript

JavaScripts in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.

Scripts in the head section:

- Scripts to be executed when they are called, or when an event is triggered go in the head section.
- When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

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

Scripts in the body section:

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

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

Scripts in both the body and the head section:

- 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>
</html>
```

❖ Variables

A variable is a "container" for information you want to store.

- A variable's value can change during the script. You can refer to a variable by name to see its value or to change its value.
- Rules for variable names:
  - Variable names are case sensitive
  - They must begin with a letter or the underscore character

Declare a Variable

- You can create a variable with the var statement:

  var strname = some value
- You can also create a variable without the var statement:
  strname = some value

Assign a Value to a Variable

- You can assign a value to a variable like this:

  var strname = "Anil"

  Or

  strname = "Anil"
- The variable name is on the left side of the expression and the value you want to assign to the variable is on the right. Now the variable "strname" has the value "Anil".

Lifetime of Variables

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

are declared, and ends when the page is closed.

❖ JavaScript Operators

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 y=2 x+y	4
-	Subtraction	x=5 y=2 x-y	3
*	Multiplication	x=5 y=4 x*y	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 ; x++	x=6
--	Decrement	x=5 ; x--	x=4

■ Assignment Operators

Operator Example		Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y

/=	x/=y	x=x/y
%=	x%=y	x=x%y

Comparison / Relational Operators

	Operator Descrition	Example
= =	is equal to	5==8 returns false
= = =	is equal to (checks for both value and type)	x=5 y="5" x==y returns true x===y returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operators

	OperatorDescription	Example
&&	And	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x = =5 || y = =5) returns false
!	Not	x=6 y=3 !(x= =y) returns true

String Operator

- To stick two or more string variables together, use the + operator.

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

The variable txt3 now contains "What a very nice day!".

Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax
variablename=(condition)?value1:value2
Example
greeting=(visitor=="PRES")?"Dear President ":"Dear "

If the variable visitor is equal to PRES, then put the string "Dear President
" in the variable named greeting.
If the variable visitor is not equal to PRES, then put the string "Dear " into
the variable named greeting.

❖ Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- 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 condition is true and another code if the condition is false
- if...else if....else statement - use this statement if you want to select one of 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 should use the if statement if you want to execute some code only if a specified condition is true.
- Code is executed only if specified condition is true.

Syntax

```
if (condition)
{
```

```
        Code to be executed if condition is true
    }
```

Example

```
<script
    type="text/javascript">
    var a=10
    var b=20

    if (a > b)
    {
        document.write("<b> A Is Greater </b>")
    }
</script>
```

Note: if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

If...else Statement

- If you want to execute some code if a condition is true and another code if the condition is not true, use the if....else statement.

Syntax:

```
if (condition)
{
    code to be executed if condition is true
}
else
{
    code to be executed if condition is not true
}
```

Example

```
<script type="javascript"> var
    a=10
    var b=20

    if (a > b)
    {
        document.write("<b> A Is Greater </b>")
    }
    else
    {
        document.write("<b> B Is Greater </b>")
    }
</script>
```

If...else if...else Statement

- You should use the if....else if...else statement if you want to select one of many sets of lines to execute.

Syntax

```
if (condition1)
{
      code to be executed if condition1 is true
}
else if (condition2)

{
      code to be executed if condition2 is true
}
else
{
      code to be executed if condition1 and condition2 are not true
}
```

Example

```
<script
      type="text/javascript
      "> var d = new
      Date()
      var time = d.getHours()

      if (time<10)
      {
            document.write("<b>Good morning</b>")
      }
      else if (time>10 && time<16)
      {
            document.write("<b>Good day</b>")
      }
      else
      {
            document.write("<b>Hello World!</b>")
      }
</script>
```

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 n (most often a variable) 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.

Switch Statement

Syntax switch(n)

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

Example

```
<script type="text/javascript">
      //You will receive a different greeting
      based //on what day it is. Note that
      Sunday=0, //Monday=1, Tuesday=2,
      etc.
      var d=new
      Date()
      theDay=d.get
      Day()

      switch (theDay)
      {
            case 5:
                  document.write("Finally
                  Friday") break
            case 6:
                  document.write("Super
                  Saturday") break
            case 0:
                  document.write("Sleepy
                  Sunday") break
            default:
                  document.write("I'm looking forward to this
            weekend!")
```

```
        }
    </script>
```

❖ JavaScript Loops

Very often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript there is two different kind of loops:

- for - loops through a block of code a specified number of times
- while - loops through a block of code while a specified condition is true

The for Loop

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

Syntax

```
for ( initialize ; condition ; expression)
{
    code to be executed
}
```

- The for loop has three section initialize, condition & expression.
- Initialize section executed first and that is evaluated once.
- Condition section check the condition if condition is true then code to be executed else executed the next statement after for loop.
- Expression section has any type of expression that change the conditional variable value.

Example

```
<html>
<body>
        <script
            type="text/javascript
            "> var i=0
            for ( i=0 ; i<=5 ; i++ )
            {
                document.write("The number is
                " + i) document.write("<br>")
            }
    </script>
```

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

Output

```
The number is 0 The
number is 1 The
number is 2 The
number is 3 The
number is 4 The
number is 5
```

Explanation: The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs.

The while loop

- The while loop is used when you want the loop to execute and continue executing while the specified condition is true.
- While loop is also known as entry control loop. Because in while loop first

  check the condition and then executed the code.

Syntax

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

Example

```
<html>
<body>
 <script ype="text/javascript"> var i=0
                    while (i<=5)
                    {
                          document.write("The number is
                          " + i) document.write("<br />")
                          i=i+1
                    }
 </script> </body>
```

Result

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
```

The number is 5

Explanation: The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs.

The do...while Loop

- The do...while loop is a variant of the while loop.
- This loop will always execute a block of code ONCE, and then it will repeat the loop as long as the specified condition is true.
- This loop will always be executed once, even if the condition is false, because the code are executed before the condition is tested.
- Do…while loop is also known as exit control loop.

Syntax do

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

Example

```
<html>
<body>
        <script
              type="text/javascript
              "> var i=5
              do
              {
                    document.write("The number is " + i)
                    document.write("<br>")
                    i=i+1
               }
              while
        (i<0)
        </script>
</body>
</html>
```

Result

The number is 5

❖ JavaScript break and continue Statements

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

Break

- The break command will break the loop and continue executing the

code that follows after the loop (if any).

Example

```
<html>
<body>
        <script
            type="text/javascript
            "> var i=0
            for (i=0;i<=10;i++)
            {
                if (i==3){break}
                document.write("The number is
                " + i) document.write("<br>")
            }
    </script>
    </body>
    </html>
```

Result

The number is 0
The number is 1
The number is 2

Continue

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

Example

```
<html>
<body>
        <script
            type="text/javascript
            "> var i=0
            for (i=0;i<=5;i++)
            {

            if (i==3){continue}
                document.write("The number is
                " + i) document.write("<br>")
            }
    </script>
    </body>
    </html>
```

Result

The number is 0
The number is 1
The number is 2

The number is 4
The number is 5

❖ Dialog Boxes

JavaScript provides the ability to pickup user input or display small amount of text to the user by using dialog boxes. These dialog boxes appear as separate windows and their content depends on the information provided by the user.

There are three types of dialog boxes provided by JavaScript : Alert dialog box, Prompt dialog box, Confirm dialog box

Alert Dialog Box

- Alert dialog box use to display some textual information on web browser window.

Syntax alert (“message”)

- The alert dialog box display the string passed to the alert( ) method, as well as ok button.
- The JavaScript and the HTML program, in which this code snippet is held, will not continue processing until the OK button press.
- The alert dialog box can be used to display a message or display some information. Fro instance:
  - A message is display when user input invalid information
  - An invalid result is the output of a calculation.
  - A warning that a service is not available.

Example :

```
<html>
  <body>
    <script language=javascript>
     alert (“ Welcome To RKU”);
          document.write (“<IMG src=pict.jpg”>);
       </script>
  </bod>
  </html>
```

Confirm Box

- A confirm box is often used if you want the user to verify or accept something.

- A confirm box display the predefine message And OK & Cancel Button.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax:

```
confirm("sometext")
```

Example :

```
<html>
<body>
<script language=javascript>
if ( confirm (" Are You Ready ?" ) )
{ document.write ("Ok Button Press"); }
else
{ document.write ("Cancel Button Press"); }
</script>
</body>
</html>
```

Prompt Box

- A prompt box is often used if you want the user to input a value before entering a page.
- A prompt box display a predefine message, a textbox for user input And display OK & Cancel Button.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax:

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

- Prompt method required two block of information :
  - A message to be display
  - Default value set in textbox.

Example :

```
<html>
```

```
<body>
        <script language=javascript>
            document.write ("WelCome To RKU <br>")
            document.write( prompt ("Enter Name
            :","Anil") )
</script>
</body>
</html>
```

❖ JavaScript Arrays

- The Array object is used to store a set of values in a single variable name. We define an Array object with the new keyword.
- The following code line defines an Array object called myArray:

```
var myArray=new Array()
```

- There are two ways of adding values to an array (you can add as many values as you need to define as many variables you require).

1:

```
var mycars=new
Array()
mycars[0]="City
Honda"
mycars[1]="Volvo"
mycars[2]="BMW"
```

You could also pass an integer argument to control the array's size:

```
var mycars=new
Array(3)
mycars[0]=" City
Honda "
mycars[1]="Volvo"
mycars[2]="BMW"
```

2:

```
var mycars=new Array("City Honda","Volvo","BMW")
```

- Accessing Arrays
  - You can refer to a particular element in an array by referring to the name of the array and the index number.
  - The index number starts at 0.
  - The following code line:

```
document.write(mycars[2])
```

Result :

BMW

- Modify Values in Existing Arrays
  - To modify a value in an existing array, just add a new value to the array

with a specified index number:

```
mycars[0]="Opel"
document.write(mycars[0])
```

Result :

Opel

❖ JavaScript User Define Function:

- Function offer the ability to group together JavaScript program code that perform a specific task into a single unit that can be used repeatedly whenever required in JavaScript program
- A user defines function first need to be declared and coded. Once this is done the function can be invoked by calling it using the name given to the function when it was declared.
- Function can accept information in the form of arguments and can return results.

Syntax :

```
function function_name ( [ parameter list ] )
{
   block of JavaScript
   code [ return
   expression ]
}
```

Function are declared and created using function keyword.

Function name is always starting from alphabet character. It also allowed special char _ only. Function name is case sensitive.

Parameter passed to the function appears in parentheses and commas separate member of the list.

The return statement can be used to return any valid expression that evaluates a single value.

A function contains some code that will be executed only by an event or by a call to that function.

- You may call a function from anywhere within the page.
- Functions are defined at the beginning of a page, in the <head> section.

Example

```
<html>
<head>
        <script
            type="text/javascrip
            t"> function
            displaymessage()
            {
                alert("Hello World!")
            }
    </script>
    </head>
    <body>
        <form>
            <input type="button" value="Click me!"
                onclick="displaymessage()" >
        </form>
</body>
</html>
```

Explanation: If the line: alert("Hello world!!"), in the example above had not been written within a function, it would have been executed as soon as the line was loaded. Now, the script is not executed before the user hits the button. We have added an onClick event to the button that will execute the function displaymessage() when the button is clicked.

❖ JavaScript Objects

JavaScript is an Object Oriented Programming (OOP) language.

An OOP language allows you to define your own objects and make your own variable types.

Object Oriented Programming

- JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types.
- JavaScript has many built-in objects.
- An object is just a special kind of data. An object has properties and methods.

Properties

- Properties are the values associated with an object.
- In the following example we are using the length property of the String object to return the number of characters in a string:

```
<script type="text/javascript">
var txt="Hello World!"
document.write(txt.length)
</script>
```

Output:
1
2

- Methods
  - Methods are the actions that can be performed on objects.
  - In the following example we are using the toUpperCase() method of the String object to display a text in uppercase letters:

```
<script type="text/javascript"> var
        str="Hello world!"
        document.write(str.toUpperCase())
</script>
```

Output:
HELLO WORLD!

❖ String object

The String object is used to manipulate a stored piece of text. The String object method & property are given below:

❖ String Object Methods

- big ( ) — The big() method is used to display a string in a big font. The small() method is used to display a string in a small font.
- small ( )
- blink ( )
  - The blink() method is used to display a blinking string.
  - This method does not work in Internet Explorer.
- bold ( ) — The bold() method is used to display a string in bold.
- italics ( ) — The italics() method is used to display a string in italic.
- strike ( ) — The strike() method is used to display a string with a

strikethrough.

- sub ( ) The sub() method is used to display a string as subscript. sup ( ) The sup() method is used to display a string as superscript.

Examle :

```
<script type="text/javascript">
    var str="Hello world!"
    document.write(str.big() + '<br>' )
    document.write(str.small() + '<br>')
    document.write(str.bold() + '<br>')
    document.write(str.italics() + '<br>' )
    document.write(str.strike() + '<br>' )
    document.write(str.blink() + '<br>' )
    document.write(str.sup() + '<br>' )
    document.write(str.sub() + '<br>' )
</script>
```

- fontcolor ( ) The fontcolor() method is used to display a string in a specified color.

Syntax : stringObject.fontcolor(color)

Parameter	Description
Color	Required. Specifies a font-color for the string. The value can be a color name (red), an RGB value (rgb(255,0,0)), or a hex number (#FF0000)

- fontsize ( ) The fontsize() method is used to display a string in a specified size.

Syntax stringObject.fontsize(size)

Parameter	Description
Size	Required. A number that specifies the font size. Range 1 to 7

- link ( ) The link() method is used to display a string as a hyperlink.

Syntax: stringObject.link(stringURL)

Example

```
<script language="javascript">
    var str="link"
```

```
        document.write(str.fontcolor("yellow") + '<br>')
        document.write(str.fontsize(5) + '<br>')
        document.write(str.link("http://www.xyz.com") + '<br>')
</script>
```

- charAt ( )
  - The charAt() method returns the character at a specified position.
  - The first character in the string is at position 0.

Syntax stringObject.charAt(index)

Parameter	Description
index	Required. A number representing a position in the string

Example

```
<script type="text/javascript">
        var str="Hello world!"
        document.write(str.charAt(1))
</script>
```

Output: e

- concat( ) The concat() method is used to join two or more strings.

Syntax stringObject.concat(stringX,stringX,...,stringX)

Parameter	Description
stringX	Required. One or more string objects to be joined to a string

Example

```
<script type="text/javascript">
        var str1="Hello "
        var str2="world!"
        document.write(str1.concat(str2))
</script>
```

Output:

Hello world!

- indexOf ( )
  - The indexOf() method returns the position of the first occurrence of a specified string value in a string.
  - The indexOf() method is case sensitive!
  - This method returns -1 if the string value to search for never occurs.

Syntax

stringObject.indexOf(searchvalue,fromindex)

Parameter	Description
searchvalue	Required. Specifies a string value to search for
fromindex	Optional. Specifies where to start the search

Example

```
<script language="javascript">
    var str="Hello world!"
    document.write(str.indexOf("Hello") + "<br />")
    document.write(str.indexOf("World") + "<br />")
    document.write(str.indexOf("world"))
</script>
```

Output :

```
0
-1
6
```

lastIndexOf ( )

- The lastIndexOf() method returns the position of the last occurrence of a specified string value, searching backwards from the specified position in a string.
- The lastIndexOf() method is case sensitive!
- This method returns -1 if the string value to search for never occurs.

Syntax

stringObject.lastIndexOf(searchvalue,fromindex)

Parameter	Description
search	Required. Specifies a string value to search for
fromindex	Optional. Specifies where to start the search. Starting backwards in the string

Example

```
<script type="text/javascript">
    var str="Hello world!"
    document.write(str.lastIndexOf("Hello") + "<br />")
    document.write(str.lastIndexOf("World") + "<br />")
    document.write(str.lastIndexOf("world"))
</script>
```

Output:

```
0
-1
```

6

match ( )

- The match() method searches for a specified string value in a string.
- This method is similar to indexOf() and lastIndexOf(), but it returns the specified string, instead of the position of the string.
- The match() method is case sensitive!
- This method returns null if the string value to search for never occurs.

Syntax

stringObject.match(searchvalue)

Parameter	Description
searchvalue	Required. Specifies a string value to search for

Example

```
<script type="text/javascript">
    var str="Hello world!"
    document.write(str.match("world") + "<br />")
    document.write(str.match("World") + "<br />")
    document.write(str.match("worlld") + "<br/>")
    document.write(str.match("world!"))
</script>
```

Output:

```
world
null
null
world!
```

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)

Parameter	Description
findstring	Required. Specifies a string value to find. To perform a global search add a 'g' flag to this parameter and to perform a case-insensitive search add an 'i' flag
newstring	Required. Specifies the string to replace the found value from findstring

Example

```
<script   type="text/javascript">
      var  str1="Visit  Microsoft!"
      var  str2="Visit  Microsoft!"
      var str3="Visit Microsoft!"
      var str4="Welcome to Microsoft! "
      var str5="Welcome to Microsoft! "

      document.write(str1.replace(/Microsoft/, "W3Schools") + „<br>)
      document.write(str2.replace(/microsoft/, "W3Schools") + „<br>)
      document.write(str3.replace(/microsoft/i, "W3Schools") + „<br>)
      str4=str4 + "We are proud to announce that Microsoft have "
      str4=str4 + "one of the largest Web Developers site in the world."
      document.write(str4.replace(/Microsoft/g, "W3Schools") + „<br>)

      str5=str5 + "We are proud to announce that Microsoft have "
      str5=str5 + "one of the largest Web Developers site in the world."
      document.write(str5.replace(/microsoft/gi, "W3Schools") + „<br>)
</script>
```

Output :

```
Visit W3Schools!
Visit Microsoft!
Visit W3Schools!
Welcome to W3Schools! We are proud to announce that W3Schools have one of the largest Web Developers site in the world. Welcome to W3Schools! We are proud to announce that W3Schools have one of the largest Web Developers site in the world.
```

search ( )

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

Syntax

```
stringObject.search(searchstring)
```

Parameter	Description
searchstring	Required. The value to search for in a string. To perform a case-insensitive search add an 'i' flag

Example

```
<script type="text/javascript">
      var str="Visit W3Schools!"
      document.write(str.search(/W3Schools/))
      document.write(str.search(/w3schools/))
```

```
        document.write(str.search(/w3schools/i))
</script>

Output :
        6
        -1
        6
```

slice ( )

- The slice() method extracts a part of a string and returns the extracted part in a new string.
- You can use negative index numbers to select from the end of the string.
- If 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)

Parameter	Description
start	Required. Specify where to start the selection. Must be a number
end	Optional. Specify where to end the selection. Must be a number

Example

```
<script type="text/javascript"> var
        str="Hello happy world!"
        document.write(str.slice(6))
        document.write(str.slice(6,11))
</script>
Output :
        happy world!
        happy
```

substr ( )

- The substr() method extracts a specified number of characters in a string, from a start index.
- To extract characters 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)
```

Parameter	Description
start	Required.Where to start the extraction. Must be a numeric value
length	Optional. How many characters to extract. Must be a numeric value.

Example

```
<script language="javascript">
    var str="Hello world!"
    document.write(str.substr(3))
    document.write(str.substr(3,7))
</script>
```

Output :

```
lo world!
lo worl
```

substring ( )

- The substring() method extracts the characters in a string between two specified indices.
- To extract characters from the end of the string, use a 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)
```

Parameter	Description
start	Required. Where to start the extraction. Must be a numeric value
stop	Optional. Where to stop the extraction. Must be a numeric value

Example

```
<script type="text/javascript">
    var str="Hello world!"
    document.write(str.substring(3))
    document.write(str.substring(3,7))
</script>
```

Output :

```
lo world!
lo w
```

- toLowerCase ( )
  - The toLowerCase() method is used to display a string in lowercase letters. Syntax : stringObject.toLowerCase()

Example

```
<script type="text/javascript">
    var str="Hello World!"
    document.write(str.toLowerCase())
</script>
```

Output: hello world!

- toUpperCase ( )
  - The toUpperCase() method is used to display a string in uppercase letters. Syntax: stringObject.toUpperCase()

Example

```
<script type="text/javascript">
    var str="Hello world!"
    document.write(str.toUpperCase())
</script>
```

Output :

HELLO WORLD!

❖ String Object Property

- length
  - The length property returns the number of characters in a string. Syntax stringObject.length

Example

```
<script type="text/javascript">
    var txt="Hello World!"
    document.write(txt.length)
</script>
```

Output :

12

❖ Math object

The math object provides methods and property to move beyond simple arithmetic manipulations offered by arithmetic operators.

❖ Math Object Methods

- abs ( ) The abs() method returns the absolute value of a number.

Syntax Math.abs(x)

Parameter	Description
X	Required. Must be a numeric value

Example

```
<script type="text/javascript">
      document.write(Math.abs(7.25) + "<br />")
      document.write(Math.abs(-7.25) + "<br />")
      document.write(Math.abs(7.25-10))
</script>
```

Output :

```
7.25
7.25
2.75
```

- ceil ( )
  - The ceil() method returns the value of a number rounded UPWARDS to the nearest integer.

Syntax Math.ceil(x)

Parameter	Description
x	Required. A number

Example

```
<script type="text/javascript">
   document.write(Math.ceil(0.60) + "<br />")
   document.write(Math.ceil(0.40) + "<br />")
   document.write(Math.ceil(5) + "<br />")
   document.write(Math.ceil(5.1) + "<br />")
   document.write(Math.ceil(-5.1) + "<br />")
   document.write(Math.ceil(-5.9))
</script>
```

Output :

```
1
1
5
6
-5
-5
```

- floor ( )

- The floor() method returns the value of a number rounded DOWNWARDS to the nearest integer.

Syntax Math.floor(x)

Parameter	Description
x	Required. A number

Example

```
<script type="text/javascript">
    document.write(Math.floor(0.60) + "<br />")
    document.write(Math.floor(5.1) + "<br />")
    document.write(Math.floor(-5.9))
</script>
```

Output :
```
0
5
-6
```

exp ( )

- The exp() returns the value of Ex, where E is Euler's constant (approximately 2.7183) and x is the number passed to it.
- E is the Euler's constant, which is the base of natural logarithms (approximately 2.7183).

Syntax Math.exp(x)

Parameter	Description
x	Required. A number

Example

```
<script type="text/javascript">
    document.write(Math.exp(1) + "<br />")
    document.write(Math.exp(-1) + "<br />")
</script>
```

Output :
```
2.718281828459045
0.36787944117144233
```

cos ( )

- The cos() method returns the cosine of a number.
- The cos() method returns a numeric value between -1 and 1, which represents the cosine of the angle.

Syntax Math.cos(x)

Parameter	Description
x	Required. A number

Example

```
<script language="javascript">
      document.write(Math.cos(3) + "<br />")
      document.write(Math.cos(-3) + "<br />")
      document.write(Math.cos(0) + "<br />")
</script>
```

Output :

```
-0.9899924966004454
-0.9899924966004454
1
```

sin ( )

- The sin() method returns the sine of a number.
- The sin() method returns a numeric value between -1 and 1, which represents the sine of the argument.

Syntax

Math.sin(x)

Parameter	Description
x	Required. A number

Example

```
<script type="text/javascript">
   document.write(Math.sin(3) + "<br />")
   document.write(Math.sin(-3) + "<br />")
   document.write(Math.sin(0) + "<br />")
</script>
```

Output :

```
0.1411200080598672
-0.1411200080598672
0
```

tan ( )

- The tan() method returns a number that represents the tangent of an angle.

Syntax: Math.tan(x)

Parameter	Description
x	Required. A number

Example

```
<script type="text/javascript">
    document.write(Math.tan(0.50) + "<br />")
    document.write(Math.tan(-0.50) + "<br />")
    document.write(Math.tan(5) + "<br />")
</script>
```

Output :

```
0.5463024898437905
-0.5463024898437905

-3.380515006246586
```

log ( )

- The log() method returns the natural logarithm (base E) of a number.
- If the parameter x is negative, NaN is returned.

Syntax Math.log(x)

Parameter	Description
x	Required. A number

Example

```
<script type="text/javascript">
    document.write(Math.log(2.7183) + "<br />")
    document.write(Math.log(1) + "<br />")
    document.write(Math.log(0) + "<br />")
    document.write(Math.log(-1))
</script>
```

Output :

```
1.0000066849139877
0 -
Infinity
NaN
```

pow ( )The pow() method returns the value of x to the power of y (xy).

Syntax : Math.pow(x,y)

Parameter	Description
x	Required. A number
y	Required. A number

Example

```
<script language="javascript">
   document.write(Math.pow(1,10) + "<br />")
```

```
        document.write(Math.pow(2,3) + "<br />")
        document.write(Math.pow(-2,3) + "<br />")
    </script>
```

Output:

```
1
8
-8
```

- random ( ) The random() method returns a random number between 0 and 1.

Syntax Math.random()

Example

```
<script language="javascript">
    document.write(Math.random())
</script>
```

Output :

0.8582012591003447

- max ( )
  - The max() method returns the number with the highest value of two specified numbers.

Syntax Math.max(x,y)

Parameter	Description
x	Required. A number
y	Required. A number

Example

```
<script type="text/javascript">
    document.write(Math.max(5,7) + "<br />")
    document.write(Math.max(-3,5) + "<br />")
</script>
```

Output :

```
7
5
```

- min ( )
  - The min() method returns the number with the lowest value of two specified numbers.

Syntax Math.min(x,y)

Parameter	Description

x	Required. A number
y	Required. A number

Example

```
<script type="text/javascript">
      document.write(Math.min(5,7) + "<br />")
      document.write(Math.min(-3,5) + "<br />")
</script>
Output :
      5
      -3
```

- round ( ): The round() method rounds a number to the nearest integer.

Syntax Math.round(x)

Parameter	Description
X	Required. A number

Example

```
<script language= “javascript">
   document.write(Math.round(0.60) + "<br
   />") document.write(Math.round(0.49) +
   "<br />") document.write(Math.round(-4.60))
</script>
Output :
      1
      0
      -5
```

❖ Date object

❖ Date Object Methods

- Date ( ) The Date() method returns today's date and time.

  Syntax Date()

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

Syntax: dateObject.getDay()

getMonth ( )

- The getMonth() method returns the month, as a number.
- The value returned by getMonth() is a number between 0 and 11. January is 0, February is 1 and so on.
- This method is always used in conjunction with a Date object.

Syntax: dateObject.getMonth()

getYear ( )

- The getYear() method returns the year, as a two-digit OR a four-digit number.
- The value returned by getYear() is not always 4 numbers! For years between 1900 and 1999 the getYear() method returns only two digits. For years before 1900 and after 1999 it returns four digits!
- This method is always used in conjunction with a Date object.
- The getYear() method should no longer be used. Use the getFullYear() method instead!!

Syntax dateObject.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

```
<script type="text/javascript">
      var d = new Date("December 03, 2005 01:15:26")
 var born = new Date("July 21, 1983 01:15:00")

      document.write(Date( ) + '<br>' )
      document.write(d.getDate( ) + '<br>')
      document.write(d.getMonth( ) + '<br>')
```

```
        document.write(d.getDay( ) + '<br>')
        document.write(d.getYear( ) + '<br>')
        document.write(born.getYear( ) + '<br>')
        document.write("I was born in " + born.getFullYear( ) + '<br>')
        document.write(d.getFullYear() + '<br>')
</script>
```

Output :

```
        Sat Dec 03 01:15:26 2005
        3 11
        6
        2005
        I was born in 1983
        83 2005
```

getHours ( )

- The getHours() method returns the hour of a time.
- Note: The value returned by getHours() is a two digit number. However, the return value is not always two digits, if the value is less than 10 it only returns one digit.
- Note: This method is always used in conjunction with a Date object.

Syntax

```
dateObject.getHours()
```

getMinutes ( )

- The getMinutes() method returns the minutes of a time.
- The value returned by getMinutes() is a two digit number. However, the return value is not always two digits, if the value is less than 10 it only returns one digit.
- This method is always used in conjunction with a Date object.

Syntax dateObject.getMinutes( )

getSeconds ( )

- The getSeconds() method returns the seconds of a time.
- The value returned by getSeconds() is a two digit number. However, the return value is not always two digits, if the value is less than 10 it only returns one digit.
- This method is always used in conjunction with a Date object.

Syntax : dateObject.getSeconds()

getMilliseconds ( )

- The getMilliseconds() method returns the milliseconds of a time.

- The value returned by getMilliseconds() is a three digit number. However, the return value is not always three digits, if the value is less than 100 it only returns two digits, and if the value is less than 10 it only returns one digit.
- This method is always used in conjunction with a Date object.

Syntax dateObject.getMilliseconds()

getTime( )

- The getTime() method returns the number of milliseconds since midnight of January 1, 1970.

Syntax dateObject.getTime()

Example

```
<script type="text/javascript">
 var d = new Date("December 03, 2005 01:15:26")

 document.write("Hour :=" + d.getHours( ) + '<br>' )
 document.write("Minutes :=" + d.getMinutes( ) + '<br>')
 document.write("Seconds :=" + d.getSeconds( ) + '<br>')
 document.write("Milliseconds :=" + d.getMilliseconds( ) + '<br>')
 document.write(d.getTime( ) + " milliseconds since 1970/01/01")
</script>
```

Output :

```
Hour :=1
Minutes :=15
Seconds :=26
Milliseconds :=0
1133590526000 milliseconds since 1970/01/01
```

setDate( )

- The setDate() method is used to set the day of the month. Syntax dateObject.setDate(day)

Parameter	Description
day	Required. A numeric value (from 1 to 31) that represents a day in a month

setMonth ( )

- The setMonth() method is used to set the month.
- The value set by setMonth() is a number between 0 and 11. January is 0,

February is 1 and so on.

- This method is always used in conjunction with a Date object.

Syntax: dateObject.setMonth(month,day)

Parameter	Description
month	Required. A numeric value between 0 and 11 representing the month
day	Optional. A numeric value between 1 and 31 representing the day

setFullYear ( )

- The setFullYear() method is used to set the year.

Syntax : dateObject.setFullYear(year,month,day)

Parameter	Description
year	Required. A four-digit value representing the year
month	Optional. A numeric value between 0 and 11 representing the month
day	Optional. A numeric value between 1 and 31 representing the date

setYear ( )

- The setYear() method is used to set the year.
- If the year parameter is a two-digit number, like setYear(91), it will be perceived as 1991. To specify a year before 1900 or after 1999, always use four digits!

Syntax dateObject.setYear(year)

Parameter	Description
Year	Required. A two or four digit number that indicates the year

Example :

```
<script language=”javascript">
 var d = new Date("December 03, 2005 01:15:26")

 document.write("Month Day :=" + d.getDate( ) + '<br>')
 d.setDate(15)
 document.write("Month Day After setDate(15) :=" + d.getDate( ) + '<br>')
 d.setMonth(8)
```

```
 document.write("Month :=" + d.getMonth( ) + '<br>')
 d.setYear(2004)
 document.write("Year :=" + d.getYear( ) + '<br>')
 d.setFullYear(90);
 document.write("Year :=" + d.getFullYear( ) + '<br>')
</script>
```

Output :

Month Day :=3
Month Day After setDate(15) :=15
Month :=8
Year :=2004
Year :=90

setHours ( )

- The setHours() method is used to set the hour of a specified time.
- If one of the parameters above is specified with a one-digit number, JavaScript adds one or two leading zeros in the result.

Syntax dateObject.setHours(hour,min,sec,millisec)

Parameter	Description
hour	Required. A numeric value between 0 and 23 representing the hour
min	Optional. A numeric value between 0 and 59 representing the minutes
sec	Optional. A numeric value between 0 and 59 representing the seconds
millisec	Optional. A numeric value between 0 and 999 representing the milliseconds

setMinutes( )

- The setMinutes() method is used to set the minutes of a specified time.
- If one of the parameters above is specified with a one-digit number, JavaScript adds one or two leading zeros in the result.

Syntax dateObject.setMinutes(min,sec,millisec)

Parameter	Description
min	Required. A numeric value between 0 and 59 representing the minutes
sec	Optional. A numeric value between 0 and 59 representing the seconds
millisec	Optional. A numeric value between 0 and 999 representing the milliseconds

setSeconds ( )

- The setSeconds() method is used to set the seconds of a specified time.
- If one of the parameters above is specified with a one-digit number, JavaScript adds one or two leading zeros in the result.

Syntax

dateObject.setSeconds(sec,millisec)

Parameter	Description
sec	Required. A numeric value between 0 and 59 representing the seconds
millisec	Optional. A numeric value between 0 and 999 representing the milliseconds

❖ Array object

❖ Array Object

Methods concat ( )

- The concat() method is used to join two or more arrays.
- This method does not change the existing arrays; it only returns a copy of the joined arrays.

Syntax

arrayObject.concat(arrayX,arrayX,.....,arrayX)

Parameter	Description
arrayX	Required. One or more array objects to be joined to an array

Example

```
<script type="text/javascript">
    var arr = new Array(3)
    arr[0] = "Jani"
    arr[1] = "Tove"
    arr[2] = "Hege"
    var arr2 = newArray(3)
    arr2[0] = "John"
    arr2[1] = "Andy"
    arr2[2] = "Wendy"
    document.write(arr.concat(arr2))
</script>
OutPut :
    Jani,Tove,Hege,John,Andy,Wendy
```

join ( )

- The join() method is used to put all the elements of an array into a string.
- The elements will be separated by a specified separator.
- Comma is the default separator for the join() method.

Syntax arrayObject.join(separator)

Parameter	Description
separator	Optional. Specifies the separator to be used

Example

```
<script type="text/javascript">
        var arr = new Array(3)
        arr[0] = "Jani"
        arr[1] = "Hege"
        arr[2] = "Stale"
document.write(arr.join() + "<br />")
        document.write(arr.join("."))
</script>
```

Output :

```
Jani,Hege,Stale
Jani.Hege.Stale
```

pop ( )

- The pop() method is used to remove and return the last element of an array.
- This method changes the length of the array.
- To remove and return the first element of an array, use the shift() method.

Syntax: arrayObject.pop()

Example

```
<script type="text/javascript">
        var arr = new Array(3)
        arr[0] = "Jani"
        arr[1] = "Hege"
        arr[2] = "Stale"
        document.write(arr + "<br />")
document.write(arr.pop() + "<br />")
        document.write(arr)
</script>
```

Output :

```
Jani,Hege,Stale
Stale Jani,Hege
```

push ( )

- The push() method adds one or more elements to the end of an array and returns the new length.
- This method changes the length of the array.
- To add one or more elements to the beginning of an array, use the unshift() method.

Syntax

arrayObject.push(newelement1,newelement2,....,newelementX)

Parameter	Description
newelement1	Required. The first element to add to the array
newelement2	Optional. The second element to add to the array
newelementX	Optional. Several elements may be added

Example

```
<script type="text/javascript">
    var arr = new Array(3)
    arr[0] = "Jani"
    arr[1] = "Hege"
    arr[2] = "Stale"
document.write(arr + "<br />")
    document.write(arr.push("Kai Jim") + "<br
    />") document.write(arr)
</script>
Output :
    Jani,Hege,Stale
    4
    Jani,Hege,Stale,Kai Jim
```

reverse ( )

- The reverse() method is used to reverse the order of the elements in an array.
- The reverse() method changes the original array.

Syntax arrayObject.reverse()

Example

```
<script type="text/javascript">
    var arr = new Array(3)
    arr[0] = "Jani"
    arr[1] = "Hege"
    arr[2] = "Stale"
    document.write(arr + "<br />")
    document.write(arr.reverse())
</script>
```

Output :

Jani,Hege,Stale
Stale,Hege,Jani

shift ( )

- The shift() method is used to remove and return the first element of an array.
- This method changes the length of the array.
- To remove and return the last element of an array, use the pop() method. SyntaxarrayObject.shift()

Example

```
<script type="text/javascript">
    var arr = new Array(3)
    arr[0] = "Jani"
    arr[1] = "Hege"
    arr[2] = "Stale"
    document.write(arr + "<br />")
    document.write(arr.shift() + "<br
    />") document.write(arr)
</script>
```

Output :

Jani,Hege,Stale
Jani Hege,Stale

sort ( )

- The sort() method is used to sort the elements of an array.
- The sort() method will sort the elements alphabetically by default. However, this means that numbers will not be sorted correctly (40 comes before 5). To sort numbers, you must create a function that compare numbers. ▪ After using the sort() method, the array is changed.

Syntax arrayObject.sort(sortby)

Parameter	Description
Sortby	Optional. Specifies the sort order. Must be a function

Example

```
<script type="text/javascript">
    var arr = new Array(6)
    arr[0] = "Jani"
    arr[1] = "Hege"
    arr[2] = "Stale"
    arr[3] = "Kai Jim"
    arr[4] = "Borge"
```

```
        arr[5] = "Tove"
        document.write(arr + "<br />")
        document.write(arr.sort())
</script>
Output :
        Jani,Hege,Stale,Kai Jim,Borge,Tove
        Borge,Hege,Jani,Kai Jim,Stale,Tove
```

❖ Array Object Property

- length The length property sets or returns the number of elements in an array.

Syntax arrayObject.length

Example

```
<script type="text/javascript">
        var arr = new Array(3)
        arr[0] = "John"
        arr[1] = "Andy"
        arr[2] = "Wendy"
        document.write("Original length: " +
        arr.length) document.write("<br />")
        arr.length=5
        document.write("New length: " + arr.length)
</script>
Output :
        Original length:
        3 New length: 5
```

❖ User Define Object

JavaScript permit the creation of user define object.

A user define object will also be associated with properties and methods.

- After creation of such an object, any number of instances of this object can be created & used.

Let's illustrate with an example: A person is an object. Properties are the values associated with the object. The persons' properties include name, height, weight, age, skin tone, eye color, etc. All persons have these properties, but the values of those properties will differ from person to person. Objects also have methods. Methods are the actions that can be performed on objects. The

persons' methods could be eat(), sleep(), work(), play(), etc.

There are different ways to create a new object:

- Create a direct instance of an object
- Create a template of an object

Create a direct instance of an object

- The following code creates an instance of an object and adds four properties to it:

```
personObj=new Object()
personObj.firstname="Anil"
personObj.lastname="Patel
" personObj.age=25
personObj.eyecolor="blue"
```

- Adding a method to the personObj is also simple. The following code adds a method called eat() to the personObj:

```
personObj.eat=eat & Also declare the UDF
        function eat ( ) { code }
```

Create a template of an object

- The template defines the structure of an object:

```
function person(firstname,lastname,age,eyecolor)
{
    this.firstname=firstname
    this.lastname=lastname
    this.age=age
    this.eyecolor=eyecolor
}
```

- The template is just a function. Inside the function you need to assign things to this.propertyName.

- Here "this" refer to the current object in focus.
- For Example, this.age will refer to the age of the current object.
- Once you have the template, you can create new instances of the object, like this:

```
x=new person("Anil","Patel",25,"blue")
y=new person("Sunil","Shah",18,"green")
```

- You can also add some methods to the person object. This is also done inside the template:

```
function person(firstname,lastname,age,eyecolor)
{
    this.firstname=firstname
```

```
        this.lastname=lastname
        this.age=age
        this.eyecolor=eyecolor

        this.newlastname=newlastname
    }
```

- Note that methods are just functions attached to objects. Then we will have to write the newlastname() function:

```
function newlastname( )
{
    code
}
```

❖ Form Object Elements

Form Element	Description & Syntax
Text	A Text field (<INPUT TYPE = text)
Password	A password textbox (<INPUT TYPE = password)
Checkbox	A checkbox (<INPUT TYPE = checkbox)
Radio	A radio (<INPUT TYPE = radio)
Reset	A reset button (<INPUT TYPE = reset)
Submit	A submit button (<INPUT TYPE = submit)
Hidden	A field that contain a value but is not displayed with in a form (<INPUT TYPE = hidden)
Select	A Select list (<SELECT> <OPTION> <OPTION> ---------- ---------- </SELECT>)
TextArea	A multiple line text entry field (<TEXTAREA> Default Value </TEXTAREA>)

❖ Property Of Form Elements

Element	Property	Description

Name	Name	
Text Password TextArea Radio Checkbox Select Submit Reset Hidden	Name	Indicate the name of the Object. This name can be used for referencing the object in future, when required.
Text Password TextArea Radio Checkbox Submit Reset Hidden	Value	Indicate the current value of the element.
Select		It contain any value indicated in the option tag.
Text Password TextArea	DefaultValue	Indicates the default value of the object.
Radio Checkbox	Checked	Indicates the current status of the object , whether checked or unchecked
Radio Checkbox	DefaultChecked	Indicate the default status of the element.
Radio	Length	Indicate the number of radio buttons in a group
Radio	Index	Indicate the index value of the currently selected radio button.
Select		Contain the index value in the current option of the option array.
Select	SelectedIndex	Contain the index number of the currently selected option.
Select	Text	Contain the value of the text display in the menu for the specific option.
Select	Selected	Indicate the current status of the option.

❖ Standard Method Or Events Of Form Elements

Element Name	Method / Event	Description
Text Password TextArea	onFocus( )	Fires when the form cursor enters into an object

Text Password TextArea	onBlur ( )	Fires when the form cursor is moved away from an object
Text Password TextArea	onSelect ( )	Fire when text is selected in an object.
Text Password TextArea	onChange( )	Fire when text is changed in an Object.
Radio Checkbox Submit Reset	onClick ( )	Fires when an object is clicked on.

❖ JavaScript Events

Event	The event occurs when...
onabort	Loading of an image is interrupted
onblur	An element loses focus
onchange	The content of a field changes
onclick	Mouse clicks an object
ondblclick	Mouse double-clicks an object
onerror	An error occurs when loading a document or an image
onfocus	An element gets focus
onkeydown	A keyboard key is pressed
onkeypress	A keyboard key is pressed or held down
onkeyup	A keyboard key is released
onload	A page or an image is finished loading
onmousedown	A mouse button is pressed
onmousemove	The mouse is moved
onmouseout	The mouse is moved off an element
onmouseover	The mouse is moved over an element
onmouseup	A mouse button is released
onreset	The reset button is clicked
onresize	A window or frame is resized
onselect	Text is selected
onsubmit	The submit button is clicked
onunload	The user exits the page

- onabort The onabort event occurs when loading of an image is aborted.

Syntax onabort="SomeJavaScriptCode"

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following JavaScript objects:

image

Example

- In this example we will call a function if the loading of the image is aborted:

```
<html>
<head>
<script type="text/javascript">
        function abortImage()
        {
        alert('Error: Loading of the image was aborted')
        }
</script>
</head>
<body>
        <img src="image_w3default.gif" onabort="abortImage()">
</body>
</html>
```

- onblur The onblur event occurs when an object loses focus.

Syntax onblur="SomeJavaScriptCode"

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following JavaScript objects:

button, checkbox, frame, password, radio, reset, submit, text, textarea, window

Example

- In this example we will execute some JavaScript code when a user leaves an input field:

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

```
function upperCase()
{
      var x=document.getElementById("fname").value
      document.getElementById("fname").value=x.toUpperCase()
}
</script>
</head>
<body>
      Enter your name:
      <input type="text" id="fname"
onblur="upperCase()"> </body>
</html>
```

- onchange The onchange event occurs when the content of a field changes.

Syntax onchange="SomeJavaScriptCode"

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following JavaScript objects: select, text, textarea

Example

- In this example we will execute some JavaScript code when a user changes the content of an input field:

```
<html>
<head>
<script type="text/javascript">
function upperCase(x)
{
 var y=document.getElementById(x).value
 document.getElementById(x).value=y.toUpperCase()
}
</script>
</head>
<body>
      Enter your name:
      <input type="text" id="fname"
onchange="upperCase(this.id)"> </body>
</html>
```

- onclick The onclick event occurs when an object gets clicked.

Syntax onclick="SomeJavaScriptCode"

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following JavaScript objects:
button, document, checkbox, link, radio, reset, submit

Example

- In this example the text in the first input field will be copied to the second input field when a button is clicked:

```
<html>
<body>
        Field1: <input type="text" id="field1" value="Hello
        World!"> <br />
        Field2: <input type="text" id="field2">
        <br /><br />

        Click the button below to copy the content of Field1 to Field2.
        <br />
        <input type=submit value="Copy Text"
              onclick="document.getElementById('field2').value=
              document.getElementById('field1').value">
</body>
</html>
```

- ondblclick The ondblclick event occurs when an object gets double-clicked.

Syntax ondblclick="SomeJavaScriptCode"

Parameter	Description
SomeJavaScriptCode	Required. Specifies a JavaScript to be executed when the event occurs.

Supported by the following JavaScript objects: document, link

Example

- In this example the second field changes according to the first field when you double-click on the button:

```
<html>
<body>
        Field1: <input type="text" id="field1" value="Hello
        World!"> <br />
        Field2: <input type="text" id="field2">
        <br /><br />
        Click the button below to copy the content of Field1 to Field2.
        <br />
```

```
        <input type=submit value=”copy text”
              ondblclick="document.getElementById('field2').value=
              document.getElementById('field1').value">
    </body>
    </html>
```

- onerror The onerror event is triggered when an error occurs loading a document or an image.

  Syntax onerror="SomeJavaScriptCode"

  Supported by the following JavaScript objects: window, image

  Example

  - In this example an alert box will be displayed if an error occurs when loading an image:

```
<img src="image.gif"
onerror="alert('The image could not be loaded.')">
```

- onfocus The onfocus event occurs when an object gets focus.

  Syntax onfocus="SomeJavaScriptCode"

  Supported by the following JavaScript objects: button, checkbox, fileUpload, layer, frame, password, radio, reset, select, submit, text, textarea, window

  Example

  - In this example the background color of the input fields change when they get focus:

```
<html>
<head>
<script type="text/javascript">
 function setStyle(x)
 {
       document.getElementById(x).style.background="yellow"
 }
</script>
</head>
<body>
 First name: <input type="text" onfocus="setStyle(this.id)"
 id="fname"> <br />
 Last name: <input type="text" onfocus="setStyle(this.id)"
id="lname"> </body>
</html>
```

- Onkeydown The onkeydown event occurs when a keyboard key is pressed.

- Browser differences: Internet Explorer uses event.keyCode to retrieve the character that was pressed and Netscape/Firefox/Opera uses event.which.

Syntax onkeydown="SomeJavaScriptCode"

Supported by the following JavaScript objects: document, image, link, textarea

Example

```
<html>
<body>
<script type="text/javascript">
    function noNumbers(e)
    {
        var keynum
        if(window.event) // IE
        {
        keynum = e.keyCode
        }
        else if(e.which) // Netscape/Firefox/Opera
        {
        keynum = e.which
        }
        alert (keynum)
    }
</script>

<form>
    <input type="text" onkeydown=noNumbers(event)
/> </form>
</html>
```

onkeypress

- The onkeypress event occurs when a keyboard key is pressed or held down.
- Browser differences: Internet Explorer uses event.keyCode to retrieve the character that was pressed and Netscape/Firefox/Opera uses event.which.

Syntax onkeypress="SomeJavaScriptCode"

Supported by the following JavaScript objects: document, image, link, textarea

Example

```
<html>
<body>
<script type="text/javascript">
    function noNumbers(e)
    {
```

```
                var keynum
                if(window.event) // IE
                {
                        keynum = e.keyCode
                }
                else if(e.which) // Netscape/Firefox/Opera
                {
                        keynum = e.which
                }
                alert (keynum)
        }
</script>
<form>
        <input type="text" onkeypress=noNumbers(event)
/> </form>
</html>
```

- onkeyup The onkeyup event occurs when a keyboard key is released.

Syntax onkeyup="SomeJavaScriptCode"

Supported by the following JavaScript objects: document, image, link, textarea

Example

```
<html>
<body>
<script type="text/javascript">
        function noNumbers(e)

        {
                var keynum
                if(window.event) // IE
                {
                        keynum = e.keyCode
                }
                else if(e.which) // Netscape/Firefox/Opera
                {
                        keynum = e.which
                }
                alert (keynum)
        }
</script>
<form>
        <input type="text" onkeyup=noNumbers(event)
/> </form>
</html>
```

- onmousedown The onmousedown event occurs when a mouse button is

clicked or down.

Syntax onmousedown="SomeJavaScriptCode"

Supported by the following JavaScript objects: button, document, link

Example

```
<img src="image_w3default.gif"
onmousedown="alert('You clicked the picture!')">
```

- onmouseupThe onmouseup event occurs when a mouse button is released or up.

Syntax onmouseup="SomeJavaScriptCode"

Supported by the following JavaScript objects: button, document, link

Example

```
<img src="image_w3default.gif"
onmousedown="alert('release the mouse button')">
```

- onmouseover The onmouseover event occurs when the mouse pointer moves over a specified object.

Syntax onmouseover="SomeJavaScriptCode"

Supported by the following JavaScript objects: button, document, link

Example

```
<img src="image_w3default.gif"
onmouseover="alert(„mouse pointer over')">
```

- onmousemove The onmousemove event occurs when the mouse pointer is moved.

Syntax onmousemove="SomeJavaScriptCode"

Supported by the following JavaScript objects: button, document, link

Example

```
<img src="image_w3default.gif"
onmousemove="alert(„mouse pointer move ')">
```

- onmouseout The onmouseout event occurs when the mouse pointer moves away from a specified object.

Syntax onmousemove="SomeJavaScriptCode"

Supported by the following JavaScript

objects: button, document, link

Example

```
<img src="image_w3default.gif"
onmouseout="alert(„mouse pointer Out ')">
```

- onselect The onselect event occurs when text is selected in a text or textarea field.

Syntax onselect="SomeJavaScriptCode"

Supported by the following JavaScript objects: text, textarea

Example

```
<form>
    Select text: <input type="text" value="Hello world!"
    onselect="alert('You have selected some of the text.')">
    <br /><br />
    Select text: <textarea cols="20" rows="5"
    onselect="alert('You have selected some of the text.')">
    Hello world!</textarea>
</form>
```

- onsubmit The onsubmit event occurs when the submit button in a form is clicked.

Syntax onsubmit="SomeJavaScriptCode"

Supported by the following JavaScript objects: form

Example

```
<form name="testform" action="jsref_onsubmit.asp"
    onsubmit="alert('Hello ' + testform.fname.value +'!')">

        What is your name?<br />
        <input type="text" name="fname" />
        <input type="submit" value="Submit" />
</form>
```

- onreset The onreset event occurs when the reset button in a form is clicked.

Syntax onreset="SomeJavaScriptCode"

Supported by the following JavaScript objects: form

Example

```
<form onreset="alert('The form will be reset')">
```

```
Firstname: <input type="text" name="fname" value="John"
/> <br />
Lastname: <input type="text" name="lname" />
<br /><br />
<input type="reset"
value="Reset"> </form>
```

- onresize The onresize event occurs when a window or frame is resized.

Syntax onresize="SomeJavaScriptCode"

Supported by the following JavaScript objects: window

Example

```
<body onresize="alert('You have changed the size of the
window')"> </body>
```

- onload The onload event occurs immediately after a page or an image is loaded.

Syntax onload="SomeJavaScriptCode"

Supported by the following JavaScript objects: image, layer, window

Example

```
<html>
<head>
<script type="text/javascript">
function load()
{
window.status="Page is loaded"
}
</script>
</head>
<body onload="load()">
</body>
</html>
```

- onunload The onunload event occurs when a user exits a page.

Syntax onunload="SomeJavaScriptCode"

Supported by the following HTML tags: <body>, <frameset>

Supported by the following JavaScript objects: window

Example

```
<body onunload="alert('The onunload event was triggered')"></body>
```

❖ Cookies

A cookie is a variable that is stored on the visitor's computer.

- When user requests a page, an HTTP request is sent to the server. The request includes a header that defines several information, including the page being requested.
- The server returns the HTTP response that also include header. The contains information about the document being return & also contain some information.
- Cookies information is shared between the client browser and server using fields in the HTTP header.
- When a user requests a page for the first time, a cookies can be stored in the browser by a set-cookie entry in the header of the response from the server.
- The set-cookie field include information to be stored in the cookie along with several optional piece of information including expire date, path and server information and if the cookies required security.
- In future a user request a page then the browser send the stored cookies information to the server in a request header.
- Set-cookie Syntax :

Set-cookie: NAME=value ; EXPIRES=date ; PATH=path ;
DOMAIN=domain; SECURE

- The NAME=value is required piece of information & all other are optional.

Name	Description
NAME=value	Specifies the name of the cookie.
EXPIRES=date	Specifies the expiry date of the cookies. After this date the cookies will no longer to stored by the client. The form of date is DD-MON-YY HH:MM:SS
PATH=path	Specifies the path portion of the URL for which the cookie is valid. If the URL matches both the PATH and DOMAIN then the cookie is sent to the server. The default value is the current web page.
DOMAIN=domain	Specifies the domain portion of the URL for which the cookie is valid. The default value is the current domain.

SECURE	Specifies that the cookie should only be transmitted over a secure link.

Example :

```
<html>
<head
```

```
>
<script
  type="text/javascript">
  function newcookie( )
  {
    document.cookie = "name=Anil"
    document.cookie = "age=18"
  }
</script>
<body onload="newcookie()">
<script>
  var str= document.cookie
  document.write ("Cookie String : <b>" + str +"</b>") document.write
  ("<br> Total Char in Cookie : <b>" + str.length +
"</b>")
</script
>
</body>
</html>
```

❖ Browser Detection

- So, sometimes it can be very useful to detect the visitor's browser type and version, and then serve up the appropriate information.
- The best way to do this is to make your web pages smart enough to look one way to some browsers and another way to other browsers.
- JavaScript includes an object called the Navigator object that can be used for this purpose.
- The Navigator object contains information about the visitor's browser name, browser version, and more.
- The Navigator Object
  - The JavaScript Navigator object contains all information about the visitor's browser.
  - Some properties of the Navigator object:

Property	Description
appName	Name of the browser
appVersion	Version of the browser
appMinorVersion	Minor version number
platform	Operating System Being used
cookieEnabled	Does the web-browser support cookie?

userAgent	Name of the user agent
browserLanguage	Give the browser language
systemLanguage	Return the system language
userLanguage	Return the user language

Example :

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

Output :

```
Name=Microsoft Internet Explorer
Version=4.0 (compatible; MSIE 5.01; Windows NT 5.0)
MinorVersion=0
CookieEnabled=true
Platform=Win32
UA=Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
BrowserLanguage=en-us
SystemLanguage=en-us
UserLanguage=en-us
```

❖ JavaScript Timing Events

With JavaScript, it is possible to execute some code NOT immediately after a function is called, but after a specified time interval. This is called timing events.

It's very easy to time events in JavaScript. Two key methods are used:

- setTimeout() - executes a code some time in the future
- clearTimeout() - cancels the

setTimeout() setTimeout()

Syntax

```
var t=setTimeout("javascript statement",milliseconds)
```

- The setTimeout() method returns a value - In the statement above, the value is stored in a variable called t. If you want to cancel this setTimeout(), you can refer to it using the variable name.
- The first parameter of setTimeout() is a string that contains a JavaScript statement.
- The second parameter indicates how many milliseconds from now you want to execute the first parameter.
- There are 1000 milliseconds in one second.

Example

- When the button is clicked in the example below, an alert box will be displayed after 5 seconds.

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

Example - Infinite Loop

- To get a timer to work in an infinite loop, we must write a function that calls itself. In the example below, when the button is clicked, the input field will start to count (for ever), starting at 0:

```
<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)
        }
</script>

</head>
<body>
        <form>
                <input type="button" value="Start count!"
                onClick="timedCount()">
                <input type="text" id="txt">
        </form>
</body>
</html>
```

clearTimeout()

Syntax clearTimeout(setTimeout_variable)

Example

- The example below is the same as the "Infinite Loop" example above. The only difference is that we have now added a "Stop Count!" button that stops the timer:

```
<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>
```

WISH YOU ALL THE BEST