

# **JavaScript Programming Constructs**

**Vijesh M. Nair**  
**Assistant Professor**  
**Dept. of CSE (AI-ML)**





# Introduction to JavaScript

## Incorporating JavaScript into HTML

- ❖ JavaScript programs require the `<SCRIPT>` tag in .html files

```
<script type = "text/javascript">  
    ACTUAL JavaScript code here  
</script>
```

- ❖ These can appear in either the `<HEAD>` or `<BODY>` section of an html document
  - Functions and code that may execute multiple times is typically placed in the `<HEAD>`
  - Code that needs to be executed only once, when the document is first loaded is placed in the `<BODY>`



# Introduction to JavaScript

## Using an External JavaScript

- ❖ JavaScript can also be placed in external files.
- ❖ External JavaScript files often contain code to be used on several different web pages.
- ❖ External JavaScript files have the file extension .js.
- ❖ To use an external script, point to the .js file in the "src" attribute of the <script> tag:

### Example:

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





# Introduction to JavaScript

## Hiding Scripts from Other Browsers

- ❖ Some browsers may not support scripting
- ❖ To be safe, you can put your scripts in html comments.
- ❖ This way browsers that do not recognize JavaScript will look at the programs as comments

```
<HTML>
<HEAD> <TITLE></TITLE></HEAD>
<BODY>
    Here's the result:
    <SCRIPT LANGUAGE="JavaScript">
        <!-- HIDE FROM OTHER BROWSERS
        // Output "It Works!"
            document.writeln("It works!<BR>");
        // STOP HIDING FROM OTHER BROWSERS -->
    </SCRIPT>
</BODY>
</HTML>
```



# JavaScript Statements

- ❖ JavaScript is a sequence of statements to be executed by the browser.
- ❖ A JavaScript statement is a command to a browser.
- ❖ The purpose of the command is to tell the browser what to do.
- ❖ This JavaScript statement tells the browser to write "Hello World" to the web page:
  - `document.write("Hello World");`
- ❖ It is normal to add a semicolon at the end of each executable statement.
- ❖ The semicolon is optional (according to the JavaScript standard), but using semicolons makes it possible to write multiple statements on one line.



# JavaScript Statements

## JavaScript Code

- ❖ JavaScript code (or just JavaScript) is a sequence of JavaScript statements.
- ❖ Each statement is executed by the browser in the sequence they are written.
- ❖ This example will write a heading and two paragraphs to a web page:

Example:

```
<script type="text/javascript">
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another
paragraph.</p>");
</script>
```





# JavaScript Statements

## JavaScript Blocks

- ❖ JavaScript statements can be grouped together in blocks.
- ❖ Blocks start with a left curly bracket {, and end with a right curly bracket }.
- ❖ The purpose of a block is to make the sequence of statements execute together.
- ❖ This example will write a heading and two paragraphs to a web page:

Example

```
<script type="text/javascript">
{
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
}
</script>
```



# JavaScript Comments

- ❖ Comments can be added to explain the JavaScript, or to make the code more readable.
- ❖ Single line comments start with `//`.
- ❖ Multi line comments start with `/*` and end with `*/`.

```
<html>
<body>

<script type="text/javascript">
// This is Comment 1..

/*
This is Comment 2..
this is multi-line comment
*/

document.write("<h1>ITEC 229</h1>"); // prints the text to the screen in <H1> format.

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





# JavaScript Variables

- ❖ JavaScript variables are used to hold values or expressions.
- ❖ A variable can have a short name, like x, or a more descriptive name, like carname.
- ❖ Rules for JavaScript variable names:
  - Variable names are case sensitive (y and Y are two different variables)
  - Variable names must begin with a letter or the underscore character
- ❖ A variable's value can change during the execution of a script.
- ❖ You can refer to a variable by its name to display or change its value.



# JavaScript Variables

## Declaring (Creating) JavaScript Variables

- ❖ Creating variables in JavaScript is most often referred to as "declaring" variables.
- ❖ You declare JavaScript variables with the **var** keyword:
  - `var x;`  
`var carname;`
- ❖ After the declaration shown above, the variables are empty (they have no values yet).
- ❖ However, you can also assign values to the variables when you declare them:
  - `var x=5; // will hold the value 5,`  
`var carname="Volvo"; // and carname will hold the value Volvo`

**Note:** When you assign a text value to a variable, use quotes around the value.

**Note:** If you redeclare a JavaScript variable, it will not lose its value.





# JavaScript Variables

## Local JavaScript Variables

- ❖ A variable declared within a JavaScript function becomes **LOCAL** and can only be accessed within that function. (the variable has local scope).
- ❖ You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.
- ❖ Local variables are destroyed when you exit the function.





# JavaScript Variables

## Global JavaScript Variables

- ❖ Variables declared outside a function become **GLOBAL**, and all scripts and functions on the web page can access it.
- ❖ Global variables are destroyed when you close the page.
- ❖ If you declare a variable, without using "var", the variable always becomes **GLOBAL**.
- ❖ If you assign values to variables that have not yet been declared, the variables will automatically be declared as global variables.
- ❖ The statements given below will declare the variables x and carname as global variables (if they don't already exist).
  - `x=5;`  
`carname="Volvo";`



# JavaScript Operators

## JavaScript Arithmetic Operators

- ❖ Arithmetic operators are used to perform arithmetic between variables and/or values.
- ❖ Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result	
+	Addition	x=y+2	x=7	y=5
-	Subtraction	x=y-2	x=3	y=5
*	Multiplication	x=y*2	x=10	y=5
/	Division	x=y/2	x=2.5	y=5
%	Modulus (division remainder)	x=y%2	x=1	y=5
++	Increment	x=++y	x=6	y=6
		x=y++	x=5	y=6
--	Decrement	x=--y	x=4	y=4
		x=y--	x=5	y=4



# JavaScript Operators

## JavaScript Assignment Operators

- ❖ Assignment operators are used to assign values to JavaScript variables.
- ❖ Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0





# JavaScript Operators

## The + Operator Used on Strings

- ❖ The + operator can also be used to add string variables or text values together.
- ❖ To add two or more string variables together, use the + operator;
  - `txt1="What a very";`  
`txt2="nice day";`  
`txt3=txt1+txt2;`
- ❖ After the execution of the statements above, the variable `txt3` contains "What a verynice day".



# JavaScript Operators

## The + Operator Used on Strings

- ❖ To add a space between the two strings, insert a space into one of the strings:

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

or insert a space into the expression:

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

- ❖ After the execution of the statements above, the variable txt3 contains:  
"What a very nice day"



# JavaScript Operators

## Comparison Operators

- ❖ Comparison operators are used in logical statements to determine equality or difference between variables or values.
- ❖ Given that  $x=5$ , the table below explains the comparison operators:
- ❖ Comparison operators can be used in conditional statements to compare values and take action depending on the result.

Operator	Description	Example
<code>==</code>	is equal to	<code>x==8</code> is false <code>x==5</code> is true
<code>===</code>	is exactly equal to (value and type)	<code>x===5</code> is true <code>x==="5"</code> is false
<code>!=</code>	is not equal	<code>x!=8</code> is true
<code>&gt;</code>	is greater than	<code>x&gt;8</code> is false
<code>&lt;</code>	is less than	<code>x&lt;8</code> is true
<code>&gt;=</code>	is greater than or equal to	<code>x&gt;=8</code> is false
<code>&lt;=</code>	is less than or equal to	<code>x&lt;=8</code> is true





# JavaScript Operators

## Logical Operators

- ❖ Logical operators are used to determine the logic between variables or values.
- ❖ Given that  $x=6$  and  $y=3$ , the table below explains the logical operators:

Operator	Description	Example
&&	and	$(x < 10 \ \&\& \ y > 1)$ is true
	or	$(x == 5 \    \ y == 5)$ is false
!	not	$!(x == y)$ is true



# JavaScript Operators

## 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 has the value of "PRES", then the variable greeting will be assigned the value "Dear President " else it will be assigned "Dear".





# Conditional Statements

- ❖ Conditional statements are used to perform different actions based on different conditions.
- ❖ 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 to execute some code only if a specified condition is true
  - **if..else statement** - use this statement 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 to select one of many blocks of code to be executed
  - **switch statement** - use this statement to select one of many blocks of code to be executed





# Conditional Statements

## if statement

- ❖ Use the if statement to execute some code only if a specified condition is true.
- ❖ Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!
- ❖ Notice that there is no `..else..` in this syntax. You tell the browser to execute some code **only if the specified condition is true.**

### Syntax:

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



# Conditional Statements

## if statement

### Example

```
<script type="text/javascript">
var total=0;
var even=0;
for (x=1; x<=10; x++)
{
    total = total + x;
    if ((x%2)==0)
    {
        even = even + x; }
}
document.write( "The total sum: "+total+"<br>");
document.write( "The sum of even values:"+ even);
</script>
```

### Result

The total sum:55  
The sum of even values:30



# Conditional Statements

## if..else statement

### Syntax:

```
if (expression)  
{  
    statement(s) to be executed if expression is true  
}  
  
else  
{  
    statement(s) to be executed if expression is false  
}
```





# Conditional Statements

## if..else statement

### Example

```
<script type="text/javascript">
var total = prompt("Enter your total", "50");

if (total>=50)
{
    alert("PASSED");
}
else
{
    alert("FAILED");
}

</script>
```



# Conditional Statements

## if..else if statement

### Syntax:

```
if (expression 1)
{ statement(s) to be executed if expression 1 is true }

else if (expression 2)
{ statement(s) to be executed if expression 2 is true }

else if (expression 3)
{ statement(s) to be executed if expression 3 is true }

else
{ statement(s) to be executed if no expression is true }
```



# Conditional Statements

## if..else if statement

### Example

```
<script type="text/javascript">
var age = prompt("Enter your age", "25");

if (age > 0 && age <= 3)
    {alert("WOW Baby");}
else if (age >= 4 && age < 13)
    {alert("You are child");}
else if (age >= 13 && age < 31)
    {alert("You are still young");}
else if (age >= 31 && age < 46)
    {alert("Middle Age :/");}
else if (age >= 46 && age < 71)
    {alert("You are old");}
else if (age >= 71 && age < 101)
    {alert("Get ready for a new world!!!");}
else
    {alert("Ageless");}
</script>
```





# Conditional Statements

## switch case statement

### Syntax:

```
switch (expression)
{
  case condition 1: statement(s)
  break;
  case condition 2: statement(s)
  break;
  ...
  case condition n: statement(s)
  break;
  default: statement(s)
}
```



# Conditional Statements

## switch case statement

### Example

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

var flower=prompt("Which flower whould you like to buy?","rose");

switch (flower) {
    case "rose":
        alert(flower + " costs 10TL");
        break;
    case "daisy":
        alert(flower + " costs 3TL");
        break;
    case "orchild":
        alert(flower + " costs 25TL");
        break;
    default:
        alert("SORRY there is no such a flower in our shop");
        break; }
</script>
```

# Loops



## **for loop**

For loop is the most compact form of looping and includes the following three important parts:

- ❖ The loop initialization where we initialize our counter to a starting value.
- ❖ The initialization statement is executed before the loop begins.
- ❖ The test statement which will test if the given condition is true or not. If condition is true then code given inside the loop will be executed otherwise loop will come out.
- ❖ The iteration statement where you can increase or decrease your counter.



# Loops



## for loop

- ❖ You can put all the three parts in a single line separated by a semicolon.

### Syntax:

```
for (initialization; test condition; iteration statement)
{
    statement(s) to be executed if test condition is true
}
```



# Loops

## for loop

### Example

```
<script type="text/javascript">
document.write("<h2>Multiplication table from (1 to 10)</h2>");
document.write("<table border=2 width=50%");

for (var i = 1; i <= 10; i++ ) {    //this is the outer loop
    document.write("<tr>");
    document.write("<td>" + i + "</td>");

    for ( var j = 2; j <= 10; j++ ) { // inner loop
        document.write("<td>" + i * j + "</td>");
    }
    document.write("</tr>");
}
document.write("</table>");
</script>
```



# Loops

## for loop

### Output

#### Multiplication table from (1 to 10)

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100



# Loops



## while loop

- ❖ loops through a block of code as long as a specified condition is true.

### Syntax:

```
while (condition)
{
    statement(s) to be executed if expression is true
}
```



# Loops

## while loop

### Example

```
<script type="text/javascript">
    var total=0;
    var i=0;
    var j;

    while (i<= 10) {
        j=i%2;
        if (j!= 0)
        {
            total+=i;} i++
    }
    document.write("Total of odd numbers between 1-10:"+total);
</script>
```

### Result

Total of odd numbers between 1-10: 25

# Loops



## do..while loop

- ❖ similar to the while loop except that the condition check happens at the end of the loop.
- ❖ this means that the loop will always be executed at least once, even if the condition is false.

### Syntax:

```
do
{
    statement(s) to be executed;
}
while(condition);
```





# Loops

## do..while loop

### Example

```
<script type="text/javascript">
var num = 0;

do{
num = num + 5;
document.write(num + "<br />");
}
while (num < 25);
</script>
```

### Output

```
5
10
15
20
25
```



# Functions

## ❖ Defining a function to create an object

```
function house(rms, stl, yr, gar){  
  this.rooms = rms;  
  this.style = stl;  
  this.yearBuilt = yr;  
  this.hasGarage = gar;  
}  
var myhouse = new house(8, "Ranch", 1990, true)
```



# Functions

❖ **Every object is an array of its property values and every array is an object**

- 0-based indexing

❖ **Thus,**

- ***myhouse[0] = 8***
- ***myhouse[1] = "Ranch"***
- ***myhouse[2] = 1990***
- ***myhouse[3] = true***





# Functions

❖ **Every object is also an associative array**

❖ **Thus,**

- ***myhouse["rooms"] = 8***
- ***myhouse["style"] = "Ranch"***
- ***myhouse["yearBuilt"] = 1990***
- ***myhouse["hasGarage"] = true***



# Functions

## ❖ Can dynamically extend an object instance

***yourhouse = new house(12, "Tudor",  
1934, true);***

***yourhouse.hasPorch = "false";***

***yourhouse.windows = 46;***

- Doesn't affect other object instances nor the object itself



# Functions

## ❖ A variable-length array-of-strings object

```
function stringarr(howMany, initStr) {  
    this.length = howMany;  
    for (var j = 1; j <= howMany; j++) {  
        this[j] = initStr;  
    }  
}
```





## for..in Statement

```
for (varName in objName) {  
    forBody  
}
```


❖ ***varName*** takes on the successive property names of the object ***objName***




## for..in Statement

```
function showAny(anyObj) {  
  for (var iter in anyObj) {  
    document.write("<br>Property " +  
      iter + " is " + anyObj[iter]);  
  }  
  document.write("<br>");  
}
```

## Methods




```
function house(rms, stl, yr, garp) {  
    this.length = 5;  
    this.rooms = rms;  
    this.style = stl;  
    this.yearBuilt = yr;  
    this.hasGarage = gar;  
    this.show = mshowHouse;  
}
```






## Methods



```
function mshowHouse( ) {  
    var nprops = this.length;  
    for (var iter = 1; iter < nprops;  
        iter++) {  
        document.write("<br>Property " +  
            iter + " is " + this[iter]);  
    }  
    document.write("<br>");  
}  
myhouse.show( );
```



Thank You !

