INTRODUCTION TO JAVASCRIPT

Objectives

by the end of lecture, students should be able to:

- Define what type of programming language is JavaScript.
- Describe and practice the syntax of JavaScript.
- Insert a JavaScript program into HTML file.
- Describe and practice Control Structures
- Describe and practice Operators

History of JavaScript

- JavaScript made its first appearance in Netscape 2.0 in 1995 with the name **LiveScript**.
- Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java.
- The general-purpose of the language has been embedded in Netscape, Internet Explorer, and other web browsers.
- Microsoft soon released its own version called JScript
- The ECMA-262 Specification defined a standard version of the core JavaScript language.
  - ECMAScript Edition 3 is widely supported and is what we will call "JavaScript"
- JavaScript is the most popular scripting language on the internet

WHAT IS JAVASCRIPT?

- JavaScript is a scripting language (a scripting language is a lightweight programming language) produced by Netscape for use within HTML Web pages.
- JavaScript is loosely based on Java and it is built into all the major modern browsers. (such as Internet Explorer, Mozilla, Firefox, Netscape, Opera)
- JavaScript has object-oriented capabilities.
  - program elements are organized into "objects"
- JavaScript is case sensitive.
  - E.g var a /= var A
- Complementary to and integrated with Java and HTML
- Open and cross-platform

What is JavaScript?

JavaScript is an interpreted language

- Means that scripts execute without preliminary compilation
- JavaScript code is not compiled but translated by the translator. This translator is embedded into the browser and is responsible for translating JavaScript code.
- A program called an interpreter runs on the client's computer.
- Interpreted programming language can run on any platform or O/S as long as there is an interpreter that runs on that particular setup.

What is JavaScript? (continued)

From the W3C Schools Site:

- JavaScript was designed to add interactivity to HTML pages.
- JavaScript is a scripting language.
- A scripting language is a lightweight programming language.
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation).
- Everyone can use JavaScript without purchasing a license.

What is JavaScript? (continued)

Conclusively;

- JavaScript is a lightweight, interpreted programming language with object-oriented capabilities that allows you to build interactivity and dynamism into static HTML pages.

Common uses of JavaScript

1. Form validation – check user input before submission into the server.
2. It makes HTML pages more dynamic and interactive.
3. Improve the design.
4. Image manipulation,
5. Dynamic changes of content.
6. Popups
7. Hide/show page elements
8. Image rollover swaps
9. Cookies

Where You can Write JavaScript

JavaScript can be placed in:

- a tags (Mouse Over)
- The <head> portion of a document
- The <body> portion of a document
- An externally-referenced file

➢ Script in <body>...</body> and <head>...</head> sections.

➢ Script in an external file and then include in <head>...</head> section.

➢ Frameworks – many commons functions are available as 'frameworks'

Comments in JAVASCRIPT

Older browsers may ignore the <script> tags and display JavaScript code in the body of the page.

- To avoid this, you can enclose your JavaScript code within HTML comments tag,
  Syntax:

  <!– and // -->

Example

<script language=”JavaScript”>

- <!-document.write(“Your browser supports JavaScript.”); // -->

</script>

1. The <!-- and --> tags begin and end the HTML comment. This will prevent older browsers from displaying the script, while allowing the script to work on browsers that support it.
2. The // in the last line is a JavaScript comment—this prevents the HTML comment tag from being detected as a JavaScript error.

Comments in JAVASCRIPT

- JavaScript supports both C-style and C++-style comments. Thus:
- Any text between a **//** and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters **/*** and ***/** is treated as a comment. This may span multiple lines.

- JavaScript also recognizes the HTML comment opening sequence **<!--**. JavaScript treats this as a single-line comment, just as it does the **//** comment.
- The HTML comment closing sequence **-->** is not recognized by JavaScript so it should be written as **//-->**.

Case Sensitivity

JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.

NOTE: Care should be taken while writing variable and function names in JavaScript.

JavaScript Syntax

In order for the interpreter to understand what you've written, there are some rules you must follow, i.e., syntax.

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

- Place within the **<script>... </script>** tag in HTML web page.

- The **<script>** tag alert the browser program to begin interpreting all the text between these tags as a script.

- The JavaScript interpreter that will be running your code will look line-by-line through the code to see what to do.

JavaScript Syntax (Continued)

The <script> Tag

The <script> tag is used to embed a client-side script (JavaScript).

The <script> element either contains scripting statements, or it points to an external script file through the src attribute.

Whitespace and Line Breaks:

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs.

- you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

JavaScript Syntax (Continued)

The script tag takes two important attributes:

```
<script language="javascript" type="text/javascript">
JavaScript code;
</script>
```

language: explicitly declares to the browser that your script is a JavaScript (instead of VBscript, for example).

Typically, its value will be JavaScript.

- Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute. => **It is deprecated.**

type: This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

- It validates JavaScript as part of a well form document.

JavaScript Syntax (Continued)

- The <noscript> tag is used to display a message to non-JavaScript browsers.
- The **<noscript>** tag: Defines an alternate content for users that have disabled scripts in their browser or have a browser that doesn't support client-side scripts.
  - Browsers that support JavaScript ignore the text between the <noscript> tags.
  - Older browsers ignore the <noscript> tags and display the text.

Example:

```
<noscript>
  Sorry...JavaScript is needed to go ahead.
</noscript>
```

JavaScript Syntax (Continued)

Semicolons are Optional:

1. If your statements are each placed on a separate line, **there is no need** of ;

```
<script language="javascript" type="text/javascript">
  <!--
  var1 = 10
  var2 = 20
  //-->
  </script>
```

2. If your statements are all placed on a single line, **there is need** of ;

```
<script language="javascript" type="text/javascript">
<!--
var1 = 10; var2 = 20
//-->
</script>
```

JavaScript Syntax (Continued)

Comments in JavaScript:

- JavaScript supports both C-style and C++-style comments, Thus:
- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->

JavaScript Syntax (Continued)

E.g external file :

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

script element used
to load and execute
JavaScript code

How to Include JavaScript in Your Web Page

- You can place the **<script>** tag containing your JavaScript **anywhere** within you web page **but** it is preferred to keep it within the <head> tags.

1. Place library script such as the jQuery library in the head section.
   - Using JavaScript Framework (Jquery, Prototype, MooTools, script.aculo.us)
   - Use the Minified Version of frameworks
2. Place normal script in the head unless it becomes a performance/page load issue.
3. Including it as a link to an external file (file.js).
   1. E.g. <script src="file.js"></script>
4. Place script that impacts the render of the page before the closing **</body>** tag.

Your First JavaScript Script:

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

Each statement should end with a semicolon (;)

Screen Output & Keyboard Input

- The JavaScript model for the HTML document is the `Document` object
- The model for the browser display window is the `Window` object
  - The `Window` object has two properties, `document` and `window`, which refer to the `Document` and `Window` objects, respectively.

- The `Document` object has a method, `write`, which dynamically creates content.
  - The parameter is a string, often concatenated from parts, some of which are variables

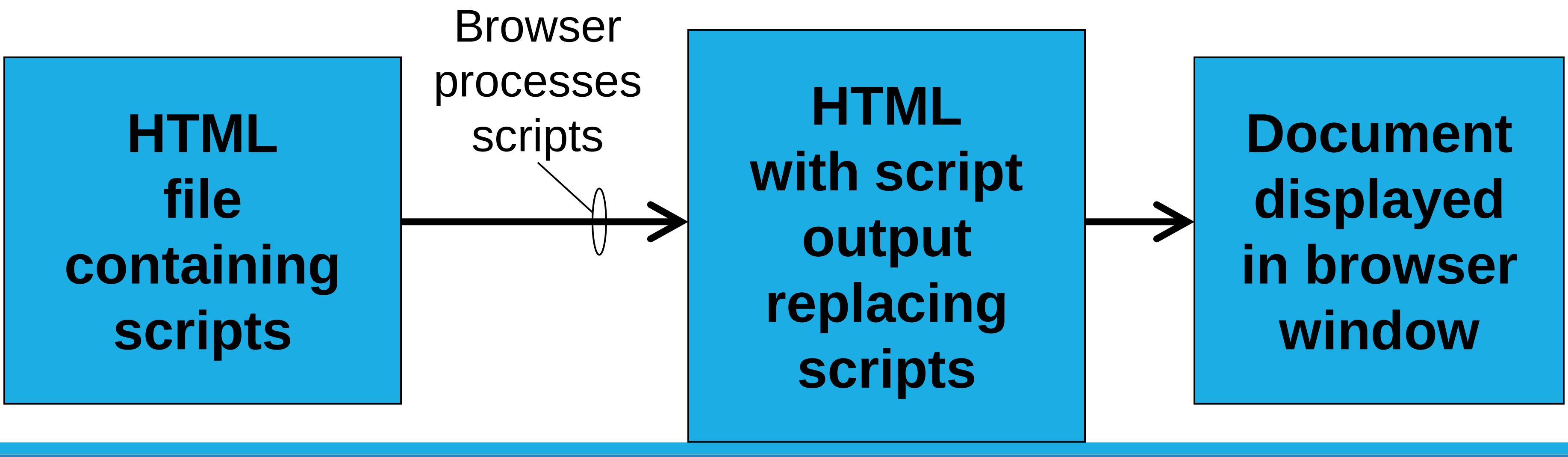
e.g., `document.write("Answer: " + result + "<br />");`

  - The parameter is sent to the browser, so it can be anything that can appear in an HTML document (`<br />`, but not `\n`)

- The `Window` object has three methods for creating dialog boxes, `alert`, `confirm`, and `prompt`

Intermediate File vs. HTML Output

some difficult concept is that the output of a JavaScript script is not output to the browser window, but instead is output to the “intermediate” HTML file that the browser will interpret for display.


HTML file containing scripts
Browser processes scripts
HTML with script output replacing scripts
Document displayed in browser window

Intermediate File vs. HTML Output (continued)

Since the JavaScript output is to be interpreted by a browser as HTML, the output must contain HTML tags.

Example – Assume we want a heading level 1 with a line break in the middle:

- **Wrong:**
  ```
  document.write("This is my \n page title");
  ```

- **Right:**
  ```
  document.write("<h1>This is my <br /> page
  title</h1>");
  ```

write vs. writeln

There are two document object methods used to write in JavaScript:

`document.write(`*`string`*`)`

Note that **write()** does NOT add a new line after each statement

`document.writeln(`*`string`*`)`

Note that **writeln()** add a new line after each statement

Screen Output (continued)

write vs. writeln (continued)

```
<!DOCTYPE html>
<html>
<body>
<pre>
<script>
document.write("Hello World!");
document.write("Have a nice day!");
</script>
</pre>
<pre>
<script>
document.writeln("Hello World!");
document.writeln("Have a nice
day!");
</script>
</pre>
</body>
</html>
```

<u>Output</u>

```
Hello World!Have a nice day!
```

```
Hello World!
Have a nice day!
```

Prompting as Page Loads

Remember that scripts within the body are executed as they are encountered

Take advantage of this by prompting the user for information as the page loads using a function:

window.prompt()

Screen Output (continued)

Double vs. Single Quotes

There are three ways to embed quotation marks within a string:

1. Use single quotes within a string identified using double quotes
2. Use double quotes within a string identified using single quotes
3. Use the JavaScript escape characters \' or \"

Examples:

1. `document.write("<a class='menu'>");`
2. `document.write('<a class="menu">');`
3. `document.write("<a class=\"menu\">");`

Screen Output (continued)

1. `alert("Hej! \n");`

   - Parameter is plain text, not HTML
   - Opens a dialog box which displays the parameter string and an `OK` button
     - It waits for the user to press the `OK` button

2. `confirm("Do you want to continue?");`

   - Opens a dialog box and displays the parameter and two buttons, `OK` and `Cancel`
   - Returns a Boolean value, depending on which button was pressed (it waits for one)

3. `prompt("What is your name?", "");`

   - Opens a dialog box and displays its string parameter, along with a text box and two buttons, `OK` and `Cancel`

   - The second parameter is for a default response if the user presses `OK` without typing a response in the text box (waits for `OK`)

An alert is a box that pops up to give the user a message. Here's code for an alert that displays the message "Thanks for your input!"

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

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

A prompt box asks the user for some information and provides a response field for her answer.

Screen Output (continued)

JavaScript Popup Boxes 1

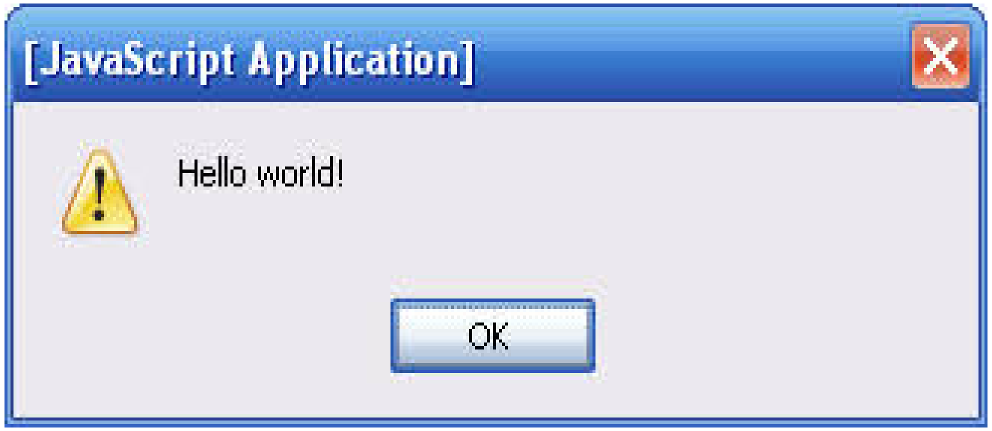
Alert Box

- a JS command that pops up a dialog box with a message
- An alert box is often used if you want to make sure information comes through to the user.
- When an alert box pops up, the user will have to click "OK" to proceed.

```
<script>

alert("Hello World!")

</script>
```



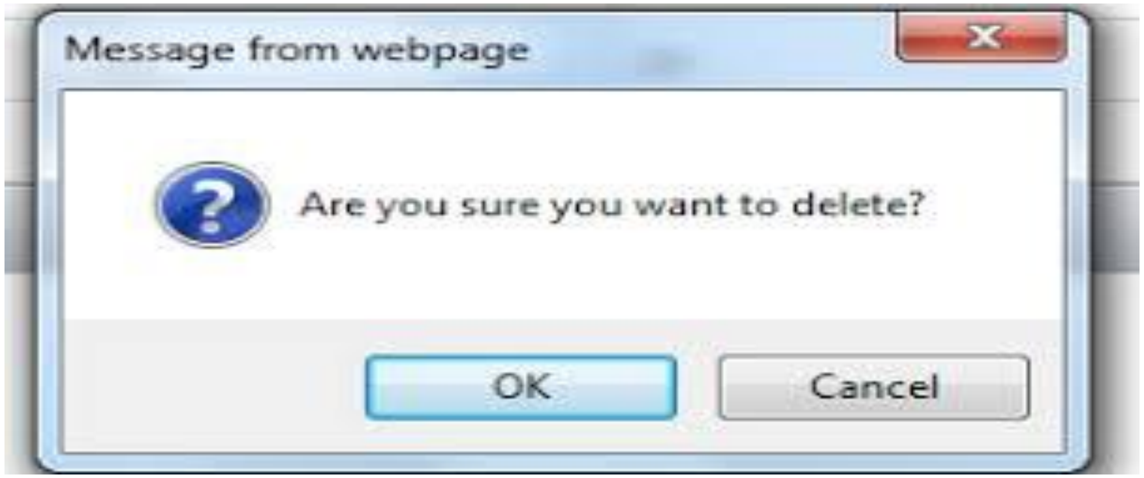
Hello world!

Screen Output (continued)

JavaScript Popup Boxes - 2

Confirm Box

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


Message from webpage

Are you sure you want to delete?

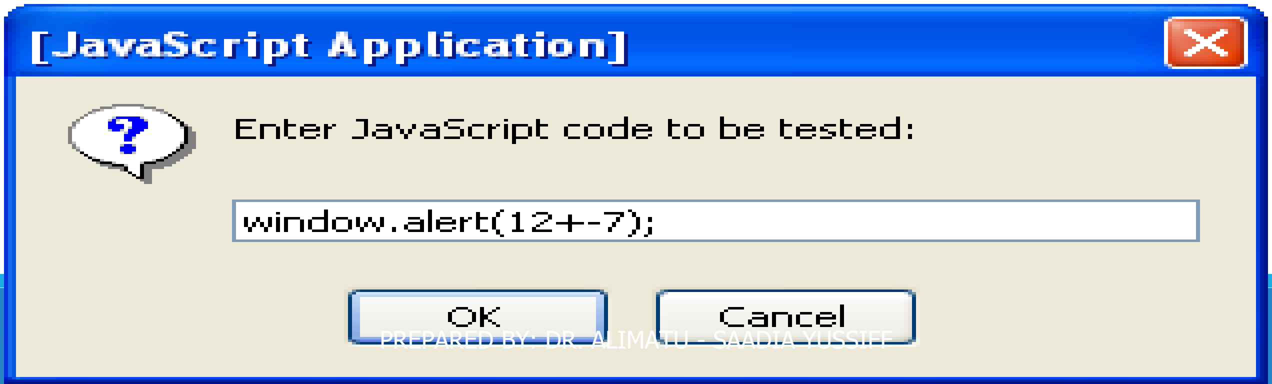
Screen Output (continued)

JavaScript Popup Boxes - 3

Prompt Box

- A prompt box is often used if you want the user to input a value before entering a page.
- 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.

```
var inString = window.prompt("Enter JavaScript code to be tested:",
                                    "");
```



Enter JavaScript code to be tested:

```
window.alert(12+-7);
```

JS Examples -1

```
<!DOCTYPE html>
<html>
<body>
<pre>
<script>
    Var X = 3;
    Var Y =20 * x + 12;
    alert(y);
</script>
```

Examples -2

```
<script>
s1=12
s2=28
toplam=s1+s2
document.write(“Michael Edna is: "+toplam);
</script>
```

JavaScript Operators

Operators are used to perform operation on one, two or more operands. Operator is represented by a symbol such as +, =, *, % etc. Following are the operators supported by JavaScript –

1. Arithmetic Operators
2. Comparison Operators
3. Logical (or Relational) Operators
4. Assignment Operators
5. Conditional (or ternary) Operators

JavaScript Operators
Arithmetic Operators

Operator	Description	Example	Result

+	**Addition** operator
	Add two operands
		x=2
		y=2
		x+y
			4

-	**Subtraction operator**
	Subtract second operand from the first.
		x=5
		y=2
		x-y
			3

*	**Multiplication** operator
	Multiply two operands
		x=5
		y=4
		x*y
			20

/	**Division** operator Divide numerator by denominator	15/5 5/2	3 2,5
%	**Modulus** operator **(division remainder)** gives remainder of the division.	5%2 10%8 10%2	1 2 0
++	**Increment** operator increases integer value by one	x=5 x++	x=6
- -	**Decrement operator** **decreases integer value by one**	x=5 X - -	x=4

JavaScript Operators – 2

Assignment Operators

Operator	Description	Example
=	**Simple Assignment operator** Assigns values from right side operands to left side operand.	C = A + B will assign value of A + B into C
+=	**Add AND assignment operator** It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	**Subtract AND assignment operator** It subtracts right operand from the left operand and assign the result to left operand	C - = A is equivalent to C = C - A
*=	**Multiply AND assignment operator** It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	**Divide AND assignment operator** It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	**Modulus AND assignment operator** Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

JavaScript Operators - 3

Comparison Operators

Operator	Description	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

JavaScript Operators - 4

Logical (or Relational) Operators

Operator	Description	Example
&&	**Logical AND operator returns true if both operands are non zero.**	**x=6** **y=3** **(x < 10 && y > 1) returns true**
||	**Logical OR operator returns true If any of the operand is non zero**	**x=6** **y=3** **(x==5 || y==5) returns false**
!	**Logical NOT operator complements the logical state of its operand.**	**x=6** **y=3** **!(x==y) returns true**

JavaScript Operators - 5
Conditional Operator

It is also called ternary operator, since it has three operands.

Operator	Description	Example
?:	Conditional Expression	If Condition is true? Then value X : Otherwise value Y

JavaScript Reserved Words

A list of all the reserved words in JavaScript are given in the following table. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

abstract	else	Instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

JavaScript Variables

A variable is a name associated with a piece of data

Variables allow you to store and manipulate data in your programs

Think of a variable as a mailbox which holds a specific piece of information

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.

JavaScript Variables

Declaring a Variable

You can declare a variable in two ways:

1. With the keyword **var** .
   - For example, var x = 42 . This syntax can be used to declare both local and global variables, depending on the execution context.
2. With the keyword **const** or **let** .
   - For example, let y = 13 . This syntax can be used to declare a block-scope local variable.

JavaScript Variables

Rules to declare variable in JavaScript:

- Variable names are case sensitive
  - strname STRNAME (not same)

- Variable name can only be started with an underscore (_) or a letter (from a to z or A to Z), or dollar ($) sign.
- In JavaScript variable names are case sensitive i.e. a is different from A.
- Numbers (0 to 9) can only be used after a letter.
- No other special character is allowed in variable name.
- No reserved words are allowed in variable name.

JavaScript Variable Scope

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

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

❖**Global variables** are defined outside any function, and can be used anywhere.

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

❖**Local variables** are defined within a function, and can be used only within that function.

- The var keyword is only truly required when you declare a local variable within a function that has the same name as a global variable. Using var will insure that a local variable is created rather than assigning a new value to the global one.

JavaScript Variable Scope

Example

```
<script type="text/javascript">
<!--
var myVar = "global"; // Declare a global variable
function checkscope( ) {
var myVar = "local"; // Declare a local variable
document.write(myVar);
}
//-->
</script>
```

Data Types

1. Primitive Data Types

No.	Datatype Description
1.	**String:** Can contain groups of character as single value. It is represented in double quotes.E.g. var x= “tutorial”.
2.	**Numbers:** Contains the numbers with or without decimal. E.g. var x=44, y=44.56;
3.	**Booleans:** Contain only two values either true or false. E.g. var x=true, y= false.
4.	**Undefined:** Variable with no value is called Undefined. E.g. var x;
5.	**Null:** If we assign null to a variable, it becomes empty. E.g. **var x=null;**

2. Non-Primitive/ Composite Data Types

No.	Datatype Description
1.	**Array:** Can contain groups of values of same type. E.g. var =[1,2,3,55];
2.	**Objects:** Objects are stored in property and value pair. E.g. var rectangle = { length: 5, breadth: 3};

Variables & Data Types

JavaScript is *untyped Language*;

- It does not have explicit data types
- No way to specify that a particular variable represents an integer, string, or real number
- This means that a JavaScript variable can hold a value of any data type. Unlike many other languages, you don't have to tell JavaScript during variable declaration what type of value the variable will hold.
- The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically

Implicit Data Types

Although JavaScript does not have explicit data types, it does have implicit data types

- If you have an expression which combines two numbers, it will evaluate to a number
- If you have an expression which combines a string and a number, it will evaluate to a string. **Examples:**

```
var x = 4;

var y = 11;

var z = “cat”;

var q = “17”;
```

```
Ans = x + y;
 Ans => 15

Ans = z + x;
 Ans => cat4

Ans = x + q;
 Ans => 417
```

```
Ans = x + y + z;
  Ans => 15cat

Ans = q + x + y;
  Ans => 17411
```

Outputting in JavaScript

```
<script>
x=“Hello World!”
document.write(x)
</script>

<script>
x=“John Kofi….”
document.write(“Merhaba” + x)
</script>
```

JavaScript Control Structures

controls the flow of execution of a program. Following are the several control structure supported by JavaScript.

There are three basic types of control structures in JavaScript: the `if` statement, the `while` loop, and the `for` loop

Each control structure manipulates a block of JavaScript expressions beginning with { and ending with }

Conditional Statements

The if statement is The fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

To perform different actions for different decisions.

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

Conditional Statements

The If Statement

The `if` statement allows JavaScript programmers to make decisions and execute statements conditionally.

Use an `if` statement whenever you come to a “fork” in the program

Syntax

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

Example

```
<html>
    <body>
    <script type="text/javascript">
        var num = prompt("Enter Number");
        if (num > 0)
        {
            alert("Given number is Positive!!!");
        }
    </script>
    </body>
</html>
```

Conditional Statements

if...else statement

If – Else is a two-way decision statement. It is used to make decisions and execute statements conditionally.

Syntax

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

Example

```
<script>
x=3
if(x<0)
{
alert (“negative”)
}
else
{
alert (“positive”)
}
</script>
```

Conditional Statements Examples

if...else statement

```
<script>
c=confirm(“are you a member?”)
if(c)
{
alert (“I am a member”)
}
else
{
alert (“I am not a member”)
}
</script>
```

```
<script>
p=prompt(“animal name)
if(p==“Dog")
{
alert(“yes")
}
else
{
alert(“No")
}
```

Conditional Statements Examples

Switch Statement

- Switch is used to perform different actions on different conditions.

- It is used to compare the same expression to several different values.

Syntax

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

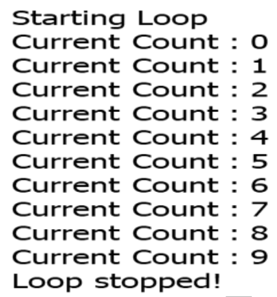
Example

```
<script type="text/javascript">
  <!--
    var grade='A';
    document.write("Entering switch block<br/>");
    switch (grade) {
      case 'A': document.write("Good job<br/>");
        break;
      case 'B': document.write("Pretty good<br/>");
        break;
      case 'C': document.write("Passed<br/>");
        break;
      case 'D': document.write("Not so good<br/>");
        break;
      case 'F': document.write("Failed<br/>");
        break;
      default:  document.write("Unknown grade<br/>")
    }
    document.write("Exiting switch block");
  //-->
</script>
```

Looping

- A group of statements that is repeated until a specified condition is met
- very powerful programming tools;
- allow for more efficient program design
- ideally suited for working with arrays

The **While** Loop



```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
Loop stopped!
```

Example

The while loop is used to execute a block of code while a certain **condition** is true.

is to execute a statement or code block repeatedly as long as expression is true. Once expression becomes false, the loop will be exited.

Syntax

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

```
<script type="text/javascript">
  <!--
    var count = 0;
    document.write("Starting Loop" +
  "<br/>");
    while (count < 10){
      document.write("Current Count
  : " + count + "<br/>");
          count++;
    }
    document.write("Loop
  stopped!");
  //-->
</script>
```

result –
```
Starting Loop
Current Count : 0
Loop stopped!
```

The do while Loop

Example

The do...while loop is similar to the while loop except that the condition check happens at the end of the loop.

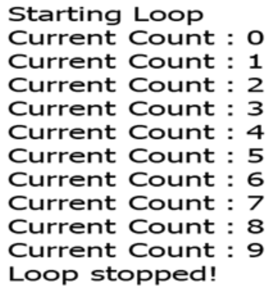
This means that the loop will always be executed at least once, even if the condition is false.

Syntax
```
do{ Statement(s) to be executed; }
while (expression);
```

```
<script type="text/javascript">
  <!--
    var count = 0;
    document.write("Starting Loop" +
  "<br/>");
    do{
      document.write("Current Count
  : " + count + "<br/>");
      count++;
    }
while (count < 0);
    document.write("Loop
  stopped!");
  //-->
</script>
```

Difference between While Loop and Do – While Loop

While Loop	Do – While Loop
In while loop, first it checks the condition and then executes the program.	In Do – While loop, first it executes the program and then checks the condition.
It is an entry – controlled loop.	It is an exit – controlled loop.
The condition will come before the body.	The condition will come after the body.
If the condition is false, then it terminates the loop.	It runs at least once, even though the conditional is false.
It is a counter-controlled loop.	It is a iterative control loop.



```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
Loop stopped!
```

The For Loop

Example

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

1. The loop initialization where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
2. 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.
3. The iteration statement where you can increase or decrease your counter.

Syntax

```
for (initialization; test condition; iteration
statement){

  Statement(s) to be executed if test condition is
true

}
```

```
<script type="text/javascript">
  <!--
    var count;
    document.write("Starting Loop" +
"<br/>");
    for(count = 0; count < 10; count++){
      document.write("Current Count : " +
count );
      document.write("<br/>");
    }
    document.write("Loop stopped!");
  //-->
</script>
```

Using the if Keyword

The if statement allows you to test conditions. E.g.:

```
if (score == 0) alert(“You lose!”);
```

An if statement has two parts:

1. a condition (score==0 in the example) and an
2. action (the alert statement in the example.)

If you need more than one statement in the action, you enclose them in braces ({ }) E.g.:

```
if (score==10) {
alert(“You win!”);
score=0;
}
else {
alert(“You lose!”);  }
```

Using for Loops

To handle repetitive tasks, use the for keyword in JavaScript to execute a block of statements a number of times. example:

```
for (i=1; i<10; i++) { //statements to repeat go here }
```

The for statement above includes three elements:

1. An initial value for a variable (i=1 in the example)
2. A condition that must remain true for the loop to continue (i<10 in the example)
3. An increment expression to change the value of the variable (i++ in the example)

Using while Loops

Sometimes, rather than executing statements a certain number of times, you’ll want them to continue to execute until something happens.

You can use the while keyword to create this type of loop. Here’s an example:

```
while (score < 10) { //statements to repeat go here }
```

This statement would execute the block of code in braces over and over until the variable score’s value reached 10. The loop itself doesn’t change the variable’s value.

Using Arguments

Functions can have one or more arguments, or variables passed to the function.

To define a function with arguments, you include one or more variable names within the parentheses. If you are using more than one argument, separate them with commas.

For example, the following is the definition for a function to display an HTML paragraph:

```
function Para(text) {
document.write(“<p>” + text + “</p>”); }
```

How to call the function:

```
Para(“Welcome to my paragraph.”);
```

Returning Values

Functions can also return a value to the script that called the function. This allows you to create functions that answer questions. For example:

```
function Average(a,b,c) {
total = a + b + c;
return total / 3;
}
```

How to call the function:

```
z = Average(2,4,6);
document.write(“The average is “ + z);
```

Example: For Loop

Example

```
// Print the numbers 1
 through 10

for (i=1; i<= 10; i++)
 document.write(i);
```

i=1 initializes the counter

i<=10 is the target value

i++ updates the counter at the end of the loop

Example: For Loop (shorten the code)

```
<SCRIPT
        LANGUAGE=
        "JavaScript">
document.write("1");
document.write("2");
document.write("3");
document.write("4");
document.write("5");
</SCRIPT>
```

Example

```
<SCRIPT
        LANGUAGE=
        "JavaScript">

for (i=1; i<=5; i++)
  document.write(i);
```

JavaScript editing tools.

Some of them are listed here:

1. Microsoft FrontPage: Microsoft has developed a popular HTML editor called FrontPage. FrontPage also provides web developers with a number of JavaScript tools to assist in the creation of interactive websites.
2. Macromedia Dreamweaver MX: Macromedia Dreamweaver MX is a very popular HTML and JavaScript editor in the professional web development crowd. It provides several handy prebuilt JavaScript components, integrates well with databases, and conforms to new standards such as XHTML and XML.
3. Macromedia HomeSite 5: HomeSite 5 is a well-liked HTML and JavaScript editor from Macromedia that can be used to manage personal websites effectively.

Advantages of JavaScript

Less server interaction: You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.

⍰ **Immediate feedback to the visitors:** They don't have to wait for a page reload to see if they have forgotten to enter something.

⍰ **Increased interactivity:** You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.

⍰ **Richer interfaces:** You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Limitations of JavaScript

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features:

 Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.

 JavaScript cannot be used for networking applications because there is no such support available.

 JavaScript doesn't have any multithreading or multiprocessor capabilities.