Introduction to Javascript

Objective

- Have a basic understanding of Javascript
- Know how to write and run Javascript code

Content

1. What is Javascript?
2. Where to try?
3. Output
4. Statements
5. Variables
6. Operators
7. Data types
8. Functions
9. Conditionals
10. Loops
11. Scope

1. What is Javascript?

Javascript is a programming language used in web programming

2. Where to try?

- HTML <script> tag
  - Internal
  - External
- Console

2. Where to try?

- HTML <script> tag
  - Internal

    Put your JS code right inside the <script> tag.

    Example:

```
<script>
        alert(“Hello world”);
</script>
```

2. Where to try?

- HTML <script> tag
  - External

    Write your JS code in a file with **.js** extension and put that file's path into src attribute of script tag

    Example:

```
<script src="path-to-file.js"></script>
```

2. Where to try?

- Console

  In a modern browser, press F12 and go to Console tab.
  Write your JS code and hit enter, it will run immediately.

3. Output

In this course we will mainly practice on a console then simply we just have to use **console.log()** to log output to console.

4. Statements

- In HTML, JavaScript statements are "instructions" to be "executed" by the web browser.
- Most JavaScript programs contain many JavaScript statements. The statements are executed, one by one, in the same order as they are written.
- (Optional) Semicolons separate JavaScript statements. Add a semicolon at the end of each executable statement.

5. Variables

JavaScript variables are containers for storing data values.

Example:

```
var x = 5;
var y = 6;
var z = x + y;
```

6. Operators

- Assignment
- Arithmetic
- Comparison
- Logical
- Type
- Bitwise

6. Operators

- Assignment operators

The assignment operator (=) assigns a value to a variable.

Operator	Example	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

6. Operators

- Arithmetic operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

6. Operators

- Comparison operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to

6. Operators

- Comparison operators

Operator	Description
<=	less than or equal to
?	ternary operator

6. Operators

- Logic operators

Operator	Description
&&	logical and
||	logical or
!	logical not

6. Operators

- Type operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

6. Operators

- Bitwise operators

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	Zero fill left shift	5 << 1	0101 << 1	1010	10
>>	Signed right shift	5 >> 1	0101 >> 1	0010	2
>>>	Zero fill right shift	5 >>> 1	0101 >>> 1	0010	2

7. Data types

- Data types in Javascript:
  - string
  - number
  - boolean
  - null
  - undefined
  - object
  - symbol

7. Data types

- JavaScript variables can hold many data types: numbers, strings, objects and more.

Example:

```
var length = 16;                                    // Number
var lastName = "Johnson";                           // String
var x = {firstName:"John", lastName:"Doe"};         // Object
```

7. Data types

- JavaScript has dynamic types. This means that the same variable can be used to hold different data types.

Example:

```
var x;                  // Now x is undefined
var x = 5;              // Now x is a Number
 var x = "John";         // Now x is a String
```

8. Functions

- Definition
  - A JavaScript function is a block of code designed to perform a particular task.
  - A JavaScript function is executed when "something" invokes it (calls it).

8. Functions

- Syntax
  - A JavaScript function is defined with the **function** keyword, followed by a **name**, followed by parentheses **()**.
  - Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
  - The parentheses may include parameter names separated by commas: (**parameter1**, **parameter2**, ...)
  - The code to be executed, by the function, is placed inside curly brackets: **{}**

8. Functions

- Syntax

```
function name(parameter1, parameter2, parameter3) {
    code to be executed
}
```

- Example

```
function myFunction(a, b) {
    return a * b;
}
var x = myFunction(5, 6); // x = 30
```

9. Conditionals

- Conditional statements are used to perform different actions based on different conditions.

- In JavaScript we have the following conditional statements:
  - Use **if** to specify a block of code to be executed, if a specified condition is true
  - Use **else** to specify a block of code to be executed, if the same condition is false
  - Use **else if** to specify a new condition to test, if the first condition is false
  - Use **switch** to specify many alternative blocks of code to be executed

9. Conditionals

- Syntax

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

9. Conditionals

- Syntax

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

9. Conditionals

- Example

```
if (time < 10) {
    greeting = "Good morning";
} else if (time < 20) {
    greeting = "Good day";
} else {
    greeting = "Good evening";
}
```

9. Conditionals

- Example

```
switch (new Date().getDay()) {
    case 0:
        day = "Sunday";
        break;
    case 1:
        day = "Monday";
        break;
    case 2:
        day = "Tuesday";
        break;
    case 3:
        day = "Wednesday";
        break;
    ...
}
```

10. Loops

- For
- While
- Do/While

10. Loops

- For
  - Syntax

```
for (statement 1; statement 2; statement 3) {
    code block to be executed
}
```

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

10. Loops

- For
  - Example

```
var text = "";

for (var i = 0; i < 5; i++) {
    text += "The number is " + i + "<br>";
}
```

10. Loops

- While
  - Definition

    The while loop loops through a block of code as long as a specified condition is true.

  - Syntax

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

10. Loops

- While
  - Example

```
var text = "";
var i = 0;

while (i < 10) {
    text += "The number is " + i;
    i++;
}
```

10. Loops

- Do/While
  - Definition

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

  - Syntax

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

10. Loops

- Do/While
  - Example

```
var text = "";
var i = 0;

do {
    text += "The number is " + i;
    i++;
} while (i < 10);
```

11. Scope

- In Javascript, scope is the set of variables, objects, and functions you have access to.
  a. Local variables
  b. Global variables

11. Scope

a. Local variables

- Variables declared within a JavaScript function, become **LOCAL** to the function.
- Local variables have **local scope**: They can only be accessed within the function

Example:

```
// code here can not use carName

function myFunction() {
    var carName = "Volvo";

    // code here can use carName
}
```

11. Scope

b. Global variables

- A variable declared outside a function, becomes **GLOBAL**.
- A global variable has **global scope**: All scripts and functions on a web page can access it.

Example:

```
var carName = " Volvo";

// code here can use carName

function myFunction() {
    // code here can use carName
}
```

12. Hoisting

- Hoisting is JavaScript's default behavior of moving declarations to the top.
  a. JS declarations are hoisted
  b. JS initializations are not hoisted

12. Hoisting

a. JS declarations are hoisted

- In JavaScript, a variable can be used before it has been declared.

- Example:

```
x = 5; // Assign 5 to x

console.log(x);

var x; // Declare x
```

12. Hoisting

b. Global variables

- JavaScript only hoists declarations, not initializations.

- Example:

```
var x = 5; // Initialize x

console.log(x); // 5
console.log(y); // undefined

var y = 7; // Initialize y
```

Thank you