**JavaScript**

What is “JavaScript”?

JavaScript is a popular programming language that has a wide range of applications.

History of JavaScript:

JavaScript was previously used mainly for making webpages interactive such as form validation, animation, etc. Nowadays, JavaScript is also used in many other areas such as server-side development, mobile app development and so on.

JavaScript Variable

Variable is divided into 2 parts:

Primitive;

Object;

**Type of Object & Primitive**

|  |  |
| --- | --- |
| **Primitive** | **Object** |
| Number | Function |
| String | Array |
| Boolean | Object interial |
| Undefined | Many more… |
| Null |
| Symbol |
| Bigint |

**JavaScript Arithmetic Operators**

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Description | Example | Result |
| + | Addiction | 3 + 11 | 14 |
| - | Subtraction | 9 – 4 | 5 |
| \* | Multiplication | 3 \* 4 | 12 |
| / | Division | 21 / 7 | 3 |
| % | Modulus | 21 % 8 | 5 |
| ++ | Increment | A=5; ++A | A=6 |
| * - | Decrement | A=5; --A | A=4 |

**Logical operation**

|  |  |  |
| --- | --- | --- |
| **Operation** | **Description** | **Example** |
| Let x=7 and y=4 |
| && | And | (x<10 && y<10) is true |
| || | Or | (x==5 || y==5) is false |
| ! | Not | !(x==y) is true |

**Comparison operation**

|  |  |  |  |
| --- | --- | --- | --- |
| **>** | **>=** | **<** | **<=** |
| **==** | **!=** | **===** | **!==** |

**Condition**

**Condition if/else Statement**

If/else;

**Condition Ternary operator**

Condition ? True : False

**Condition switch Statement**

The switch statement is used to perform different actions based on different conditions.

**Loop for**

**The syntax of the for loop is:**

for(initialExpression; condition; updateExpression ) {

// for loop body

}

Here,

1. The initialExpression initializes and/or declares variables and executes only once.

2. The condition is evaluated. \* If the condition is false, the for loop is terminated. \* If the condition is true, the block of code inside of the for loop is executed.

3. The updateExpression updates the value of initialExpression when the condition is true.

4. The condition is evaluated again. This process continues until the condition is false.

**Loop while**

**The syntax of the while loop is:**

while (condition) {

// while loop body

}

Here,

1. A while loop evaluates the condition inside the parenthesis().

2. If the condition evaluates to true, the code inside the while loop is executed.

3. The condition is evaluated again.

4. This process continues until the condition is false.

5. When the condition evaluates to false, the loop stops.

**Loop do/while**

The do...while statement creates a loop that executes a specified statement until the test condition evaluates to false. The condition is evaluated after executing the statement, resulting in the specified statement executing at least once.

**Function**

**There are 3 ways of writing a function in JavaScript**

1. Function declaration;
2. Function Express (arrow function, anonymous function);
3. Immediately Invoked Function Expression (IIFE);

**Function declaration**

**Example:**

function Decloration (name) {

return name;

}

Decloration(“Hello”);

**The function declaration defines a function with the specified parameters**.

• A function is declared using the function keyword.

• The basic rules of naming a function are similar to naming a variable. It is better to write a descriptive name for your function. For example, if a function is used to add two numbers, you could name the function add or addNumbers.

**Function IIFE**

**Example:**

var firstName = “John”;

(function (a,b) {

var firstName = “Doe”;

console.log (firstName) //Doe

} ) ( )

console.log(firstName) //John

An IIFE (Immediately Invoked Function Expression) is a function that runs the moment it is invoked or called in the JavaScript event loop. Having a function that behaves that way can be useful in certain situations. IIFEs prevent pollution of the global JS scope.

**Function Expression**

//anonymous function

let anonymous = function (parameter) {  
 return parameter;

}

anonymous(“h1”)

//arrow function

let arrow = (parameter) => {

return parameter;

}

console.log (arrow(“h1”))

A function expression is very similar to and has almost the same syntax as a function declaration. The main difference between a function expression and a function declaration is the function name, which can be omitted in function expressions to create anonymous and arrow functions.