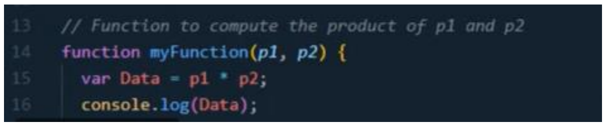
JAVASCRIPT

JavaScript, is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive.



```
// Function to compute the product of p1 and p2
function myFunction(p1, p2) {
  var Data = p1 * p2;
  console.log(Data);
```

Properties are terms that can define a particular object.

Methods are the actions that can be performed by an object or performed to an object.

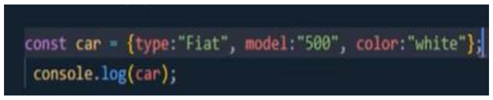
.notations are used to represent the properties and methods of particular objects.

Assign name is used to assign names for objects.

Object is a list of key value pairs embedded inside curly brackets.

Constant are those for which its value will not change.

Variables are those for which its value will change.



```
const car = {type:"Fiat", model:"500", color:"white"};
 console.log(car);
```

Keywords that provides Block Scope

Variables declared with **the let keyword** inside a { } block cannot be accessed from outside the block

Variables declared with the **var keyword** inside a { } block can be accessed from outside the block.

JAVASCRIPT OPERATORS

The **Assignment Operator** (=) assigns a value to a variable.

The **Multiplication Operator** (*) multiplies numbers:

CONSTRUCTURES

It is used for creating many objects of the same type.

It is considered to name constructor functions with an upper-case first letter.

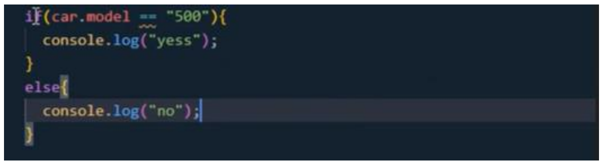
CONDITIONAL STATEMENTS

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.



```
if(car.model == "500"){
  console.log("yess");
}
else{
  console.log("no");
}
```

LOOPING STATEMENTS

For loops: - are used to execute a block of code a number of times.

While loops: - execute a block of code while a specified condition is true.

Do-while loops: - execute a block of code once, and then repeat the block while a specified condition is true.

BREAK AND CONTINUE STATEMENTS STATEMENT

The break statement "jumps out" of a loop.

The continue statement "jumps over" one iteration in the loop.

SET

A JavaScript Set is a collection of unique values.

Each value can only occur once in a Set.

The values can be of any type, primitive values or objects.

MAPS

map () creates a new array from calling a function for every array element.

map () does not execute the function for empty elements.

Map () does not change the original array.

DIFFRENCE BETWEEN OBJECTS AND MAPS

OBJECT

Objects are not directly iterable.

Objects do not have any size property.

The datatype of key for an object should be string.

The key for an object is not maintained well.

MAP

Maps are directly iterable.

Maps have size property.

The datatype of key for map can be of any datatype.

In maps objects are maintained by insertion.

TYPE CONVERSION

The global method **Number ()** converts a variable (or a value) into a number.

The global method **String ()** converts a number (or a value) into a variable.

Functions

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).

```
//function

function basefunction(v1,v2){

   console.log(v1);

   console.log(v2);

   return v1+v2;

}

const x = basefunction(1,2);

console.log(x);


//FunctionExpression

const y = function(a,b){

   return b,a

};

console.log(y(1,2))


//function hoisting

basefunction(5,6)  //calling the function before it created

function basefunction(v1,v2){

   console.log(v1);

   console.log(v2);

   return v1+v2;
```

```
}

//shelf involking function
(function() {
    let x ='hello';
    console.log(x)
})();

//function objects
function myFunction(a,b){
    return arguments.length;
}
const z = myFunction(4,3);
console.log(z)

//arrow function
const w = (a,b) => a * b;
console.log(w(25,30));

//life time of JS variable
var b = mainfunction(1,2);
console.log(b);  //global variable
var c = 250;
function mainfunction(v1,v2){
    var c = 300;  //local variable
    console.log(c);

    return v1+v2

}
```

DOM

The DOM, or Document Object Model, is a programming interface for web documents. It represents the structure of a webpage as a tree of objects, where each object corresponds to a different part of the document, such as elements, attributes, and text. The DOM provides a way for programs to dynamically access and manipulate the content, structure, and style of web pages using languages like JavaScript.

The DOM (Document Object Model) is a representation of a web page's structure, created by the browser when it loads an HTML document. It organizes the elements of the webpage in a hierarchical tree structure, where each element, attribute, and piece of text is represented as an object. This model serves as a bridge between web content and scripts, enabling dynamic interaction with the page's elements. With JavaScript, developers can access and modify the DOM, allowing them to change content, style, and structure in response to user actions or other events. In essence, the DOM provides a way for JavaScript to interact with and manipulate the elements of a webpage, making dynamic and interactive web experiences possible.