**push():**

myArray.push (“asdf”, 5); → add the elements to the end of the array

**pop():**

myArray.pop(); → remove the last element of the array

**shift():**

myArray.shift(); → remove the first element of the array

**unshift():**

myArray.unshift (“asdf”, 5); → add the elements to the beginning of the array

**function:**

function myArray(a, b){

console.log(a-b);

}

myArray(10, 5);

**sum, sub, mul, div:**

output= 5;

output += 10;

output -= 3;

output \*= 2;

output /= 15;

**\*\*** Variables which are used without the var keyword within a function are automatically created in the global scope.

var a= 10; //global v

function num1(){

b = 5; //global v

var c = 6; //local v

}

**\*** It is possible to have both *local* and *global* variables with the same name.

**Object:**

ar dogs = {

Fido: "Mutt", Hunter: "Doberman", Snoopie: "Beagle"

};

var myDog = "Hunter";

var myBreed = dogs[myDog];

console.log(myBreed); // "Doberman"

dynamic way of using objects:

var someObj = {

propName: "John"

};

function propPrefix(str) {

var s = "prop";

return s + str;

}

var someProp = propPrefix("Name"); // someProp now holds the value 'propName'

console.log(someObj[someProp]); // "John"

- **add property:**

myObj.name = “john doe”;

- delete property:

delete myObj.name;

Sometimes it is useful to check if the property of a given object exists or not. We can use the .hasOwnProperty(“propname”) method of objects to determine if that object has the given property name. .hasOwnProperty() returns true or false if the property is found or not.

Example:

var myObj = {

top: "hat",

bottom: "pants"

};

myObj.hasOwnProperty("top"); // true

myObj.hasOwnProperty("middle"); // false

**Here is a nested object:**

var ourStorage = {

"desk": {

"drawer": "stapler"

},

"cabinet": {

"top drawer": {

"folder1": "a file",

"folder2": "secrets"

},

"bottom drawer": "soda"

}

};

ourStorage.cabinet["top drawer"].folder2; // "secrets"

ourStorage.desk.drawer; // "stapler"

**Here is an example of how to access a nested array:**

var ourPets = [

{

animalType: "cat",

names: [

"Meowzer",

"Fluffy",

"Kit-Cat"

]

},

{

animalType: "dog",

names: [

"Spot",

"Bowser",

"Frankie"

]

}

];

ourPets[0].names[1]; // "Fluffy"

ourPets[1].names[0]; // "Spot"

**var vs let:**

var→ it defines variables locally and it is **function** scoped.  
let → it defines variables locally and it is **block** scoped. Outside a block, it won’t response.

Function() {

var I = “function scope”;

let I = “function scope”;

if (true) {

let I = “block scope”;

console.log(I));

}

console.log(I));

}

output: block scope

block scope (if use var)

output: block scope

function scope (if use let)

var catName = "Quincy";

var quote;

var catName = "Beau";

// let catName = "Beau"; //:4 Uncaught SyntaxError: Identifier 'catName' has already been declared

function catTalk() {

catName = "Oliver";

quote = catName + " says Meow!";

}

catTalk();

**const:**

const makes a variable read-only. We can not change a variable once we declared.

**Dialogue box:**

- alert()

- confirm()

var test = confirm(“Do you want?”)

if(test == true) {

return “Congrates”;

} else {

return “See you later”;

}

- prompt(): it show show the text with an input box where we can write something as input. And also we can use the input value.

Var test = promtp(“Write your name: ”);

console.log(test);

// Var test = promtp(“Write your name: ”, “write here”);

“write here” will be shown the input box.

parseInt(“5”): convert a string to integer

parseInt(“10010”, 2): convert a binary string to integer

Object.freeze(ObjName): It will make the object read-only

**Use Arrow Functions to Write Concise Anonymous Functions:**

const myFunc = function() {

const myVar = "value";

return myVar;

}

const myFunc = () => {

const myVar = "value";

return myVar;

}

When there is no function body, and only a return value, arrow function syntax allows you to omit the keyword return as well as the brackets surrounding the code. This helps simplify smaller functions into one-line statements

const myFunc = () => "value";

[**reduce()**](https://www.javascripttutorial.net/javascript-array-reduce/) **function**