

---

# Session 17

---

Mohamed Emary

May 24, 2024

## 1 Object

Object in JS is a collection of key-value pairs. Keys are also called properties and values can be any data type.

Objects are used to store multiple values in a single variable, these values are related to each other.

```
1 var student = {  
2   name: "Mohamed",  
3   age: 25,  
4   id: 12345,  
5   courses: ["Math", "Physics", "Chemistry"],  
6 };
```

Objects are non-primitive data types, and when you try `console.log(typeof student)` it will return `object`.

Being a *non-primitive* data type means that the object can store **multiple values**, and these values can be of **different data types**.

Printing the object will show all the properties and their values.

```
1 console.log(student);  
2 // Output:  
3 // {  
4 //   name: 'Mohamed',  
5 //   age: 25,  
6 //   id: 12345,  
7 //   courses: [ 'Math', 'Physics', 'Chemistry' ]  
8 // }
```

To access the properties of an object, you can use the `.` operator.

### Note:

When you see the dot operator with any variable, it means that the variable is an object.

# 1 OBJECT

---

```
1 console.log(student.name); // Mohamed
2 console.log(student.age); // 25
```

When you try to access a property that doesn't exist, it will return undefined.

```
1 console.log(student.city); // undefined
2 student.city = "Cairo";    // Set a new property
3 console.log(student.city); // Get the new property
```

Now when you try to print the object, you will see the new property.

```
1 console.log(student);
2 // Output:
3 // {
4 //   name: 'Mohamed',
5 //   age: 25,
6 //   id: 12345,
7 //   courses: [ 'Math', 'Physics', 'Chemistry' ],
8 //   city: 'Cairo'
9 // }
```

You can put an object inside another object.

```
1 var student = {
2   name: "Mohamed",
3   age: 25,
4   id: 12345,
5   courses: ["Math", "Physics", "Chemistry"],
6   address: {
7     city: "Cairo",
8     street: "Tahrir",
9   },
10 };

```

To access the nested object, you can use the dot operator.

```
1 console.log(student.address.city); // Cairo
2 console.log(student.address.street); // Tahrir
```

You can also put functions inside an object.

```
1 var student = {
2   name: "Mohamed",
3   age: 25,
4   id: 12345,
5   courses: ["Math", "Physics", "Chemistry"],
6   address: {
7     city: "Cairo",
8     street: "Tahrir",
9   },
10   sayHello: function() {
11     console.log("Hello, I'm a student");
12   },
13 };

```

To call the function, you can use the dot operator.

## 2 JS BUILT-IN OBJECTS

---

If you don't use the parentheses (), it will return the function itself.

```
1 student.sayHello;
2 // Output:
3 // f () {
4 //   console.log("Hello, I'm a student");
5 // }
```

To call the function, you should use the parentheses ().

```
1 student.sayHello(); // Hello, I'm a student
```

Using the sayHello function inside a console.log will print but also return undefined because the function doesn't return anything.

```
1 console.log(student.sayHello());
2 // Output:
3 // Hello, I'm a student
4 // undefined
```

But if you use a function like this:

```
1 var student = {
2   name: "Mohamed",
3   age: 25,
4   id: 12345,
5   courses: ["Math", "Physics", "Chemistry"],
6   address: {
7     city: "Cairo",
8     street: "Tahrir",
9   },
10  sayHello: function() {
11    return `Hello, I'm ${student.name}`;
12  },
13 };
```

It will return the string without undefined.

```
1 console.log(student.sayHello());
2 // Output:
3 // Hello, I'm Mohamed
```

### Note:

When you put a function inside an object, it's called a **method**.

## 2 JS Built-in Objects

JavaScript has many built-in objects like window, document, console.

### 2.1 window

window is a super global object that contains all global variables, functions, and objects.

Example functions in the window object are alert, prompt.

## 2 JS BUILT-IN OBJECTS

---

```
1 | window.alert("Hello, World!");
2 | window.prompt("What's your name?");
```

Since window is the super global object, you can access its properties and methods without using the window keyword.

This will also work:

```
1 | alert("Hello, World!");
2 | prompt("What's your name?");
```

### 2.2 document

document is another object that we can use to manipulate the HTML document.

```
1 | document.getElementById("id");
```

#### Note:

Try this code:

```
document.getElementById("id").innerHTML = "Hello, World!";
```

Why this code works? isn't getElementById a method not an object?

That is because the method getElementById returns an object so using the dot operator . allows you to access the properties of the object that the method returns.

### 2.3 console

console is also an object that has a method log that we use to print messages.

```
1 | console.log("Hello, World!");
```

### 2.4 Math

math is an object that has many properties and methods to perform mathematical operations.

```
1 | console.log(Math.PI); // 3.141592653589793
2 | console.log(Math.round(4.7)); // 5
3 | console.log(Math.floor(4.7)); // 4
4 | console.log(Math.ceil(4.4)); // 5
5 | console.log(Math.pow(2, 3)); // 8
6 | console.log(Math.sqrt(64)); // 8
7 | console.log(Math.abs(-4.7)); // 4.7
8 | console.log(Math.min(0, 150, 30, 20, -8, -200)); // -200
9 | console.log(Math.max(0, 150, 30, 20, -8, -200)); // 150
10 | console.log(Math.random()); // Random number between 0 and 1
```

Mathematical expressions of the above methods:

- $\text{Math.PI} = \pi$
- $\text{Math.round}(4.7) \approx 5$
- $\text{Math.floor}(4.7) = \lfloor 4.7 \rfloor = 4$
- $\text{Math.ceil}(4.4) = \lceil 4.4 \rceil = 5$

### 3 ARRAY

---

- `Math.pow(2, 3) = 23 = 8`
- `Math.sqrt(64) =  $\sqrt{64}$  = 8`
- `Math.abs(-4.7) = | - 4.7 | = 4.7`

```
1 console.log(Math.round(Math.random() * 10)); // Random number between 0 and
   ↪ 10
```

Example to show a random number with a button click:

In HTML body:

```
1 <div>
2     <p id="rand"></p>
3     <button id="btn">Get a random Number</button>
4 </div>
5 <script src="./script.js"></script>
```

In script.js:

```
1 var btn = document.getElementById("btn");
2 btn.onclick = getRand;
3 function getRand() {
4     var p = document.getElementById("rand");
5     p.innerHTML = Math.round(Math.random() * 10);
6 }
```

### 3 Array

Array is a collection of elements. Elements can be of any data type.

Array is a special type of object:

```
1 var nums = [1, 2, 3, 4, 5];
2 console.log(typeof nums); // object
```

And since objects are non-primitive data types, arrays can store multiple values of different data types.

Array syntax is a pair of square brackets [] with elements separated by commas.

```
1 var fruits = ["Apple", "Banana", "Orange"];
2 var mix = [
3     [1, 2, 3],
4     'Apple',
5     25,
6     true,
7     function() { return "Hello, World!"; },
8     { name: 'Mohamed', age: 25 }
9 ];
10
11 console.log(mix[0][1]); // 2
12 console.log(mix[1]);    // Apple
13 console.log(mix[4]);    // [Function]
14 console.log(mix[4]());  // Hello, World!
15 console.log(mix[5].name); // Mohamed
```

## 5 FUNCTIONAL PROGRAMMING IN JS

---

Suppose you have multiple products and each of these products is an object, and you want to store all these products in one variable, you can use an array.

```
1 var products = [  
2   { name: "Apple", price: 10 },  
3   { name: "Banana", price: 5 },  
4   { name: "Orange", price: 7 },  
5 ];
```

Each item in the array has an index starting from 0 (**not 1**) and you can access the elements using that index.

```
1 var fruits = ["Apple", "Banana", "Orange"];  
2 console.log(fruits[0]); // Apple  
3 console.log(fruits[2]); // Orange  
4 console.log(fruits);    // ["Apple", "Banana", "Orange"]
```

`fruits[0]` is read as “fruits of 0”.

To print each item in the array, you can use a loop.

```
1 var fruits = ["Apple", "Banana", "Orange"];  
2 for (var i = 0; i < fruits.length; i++) {  
3   console.log(fruits[i]);  
4 }
```

`fruits.length` returns the number of elements in the array, and since the last index is `fruits.length - 1`, the loop should run from 0 to `fruits.length - 1` so we use `i < fruits.length` instead of `i <= fruits.length` in the loop condition.

## 4 Object vs Array

	Object	Array
Type	Object & Non-primitive	Object & Non-primitive
Syntax	{ prop1: val1, prop2: val2, ... }	[ elem1, elem2, ... ]
Element Access	object.prop	array[index]
Use Case	Store different properties of an element	Store multiple elements
Index	Key	Number
Example	{ name: 'Mohamed', age: 25 }	['Apple', 'Banana']

*See the exercise at the end of video 7: object vs array & exercise*

## 5 Functional Programming in JS

JavaScript applies functional programming concepts like:

1. You can assign a function to a variable.

```
1 var x = function() {  
2   return "Hello, World!";  
3 };
```

2. Functions can be properties of objects.

```
1 var obj = {  
2   sayHello: function() { // method  
3     return "Hello there";  
4   },  
5 };
```

3. Functions can be returned from another function.

```
1 function twoNumAvg(sum) {  
2   return sum / 2;  
3 }  
4  
5 function getAvg(a, b) {  
6   var sum = a + b;  
7   return twoNumAvg(sum);  
8 }  
9  
10 console.log(getAvg(10, 20)); // 15
```

4. Functions can be passed as arguments to other functions.

```
1 function twoNumAvg(sum) {  
2   return sum / 2;  
3 }  
4  
5 function sum(a, b) {  
6   return a + b;  
7 }  
8  
9 console.log(twoNumAvg(sum(10, 20))); // 15
```

### 6 Summary

Objects:

- Object is a collection of key-value pairs.
- Objects are used to store multiple values in a single variable.
- Objects are non-primitive data types.
- Objects can store multiple values of different data types.

Arrays:

- Array is a collection of elements.
- Array is a special type of object.
- Arrays can store multiple values of different data types.

Built-in Objects:

- `window` is a super global object.
- `document` is an object that we can use to manipulate the HTML document.
- `console` is an object that has a method `log` to print messages.
- `Math` is an object that has many properties and methods to perform mathematical operations.

Functional Programming:

- JavaScript applies functional programming concepts.
  - Functions can be assigned to a variable.
  - Functions can be properties of objects.
  - Functions can be returned from another function.
  - Functions can be passed as arguments to other functions.