



OBJECTS AND ARRAYS COURSE

APRIL 2022

YONATAN BENEZRA

- DO NOT DISTRIBUTE -

- DO NOT COPY – ALL RIGHTS RESERVED TO THE
AUTHOR –

JavaScript Objects

JavaScript objects are just like real-life objects.

As an example, let's discuss a typical table.

Imagine a table with the following properties:

Color: white

Number-of-legs: 4

Height: 1.2 meters

Width: 50 cm

Length: 50 cm

Write the above data between curly braces and you will receive a JS Object!

Review the below example:

```
const table = {  
  color: "white"  
  numberOfLegs: 4  
  height: 1.2  
  width: 0.5  
  length: 0.5  
}
```

Exercise:

Create an object called chair, enter the properties of the chair you are currently sitting on and console log it. See what you get.

*Notes:

- The values are written as name:value pairs (name followed by value, separated by a colon).
- It is common practice to declare objects with the const keyword.
- Spaces and line breaks are not important. An object definition can span multiple lines.
- The name:value pairs in JavaScript objects are called properties.

Accessing Object Properties

There are two ways to access object properties:

- `objectName.propertyName`
- `objectName["propertyName"]`

Example:

```
console.log(table.color) //answer: "white"  
console.log(table[width] //answer = 0.5
```

Exercise:

console.log the color and number of wheels on your chair object.

Object Methods

Objects can have methods as well.

Methods are actions that can be performed on objects.

Methods are stored in properties as function definitions.

Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue
fullName	function() {return this.firstName + " " + this.lastName;}

A method is a function stored as a property.

If you access a method without the () parentheses, it will return the function's definition.

Example:

```
const name = person.fullName()
```

```
const person = {  
  firstName: "John",  
  lastName : "Doe",  
  id      : 5566,  
  fullName : function() {  
    return this.firstName + " " + this.lastName;  
  }  
}
```

```
}
```

```
};
```

In the example above, “this” refers to the person object.
this.firstName means the firstName property of person.
this.lastName means the lastName property of person.

What is “this”?

In JavaScript, the “this” keyword refers to an object.
Which object it refers to depends on how the “this” is being invoked (used or called).
The “this” keyword refers to different objects depending on how it is used:

- In an object method, this refers to the object.
- Alone, this refers to the global object.
- In a function, this refers to the global object.
- In a function, in strict mode, this is undefined.
- In an event, this refers to the element that received the event.
- In methods, such as call(), apply(), and bind(), “this” can refer to any object.

“this” is not a variable. It is a keyword. You cannot change the value of “this”.

“This” Keyword

In a function definition, “this” refers to the “owner” of the function.
In the example above, “this” is the person object that “owns” the fullName function.
In other words, this.firstName means the firstName property of this object.

Once we get to the React Lessons in the next module, the “this” keyword will not be used very frequently.

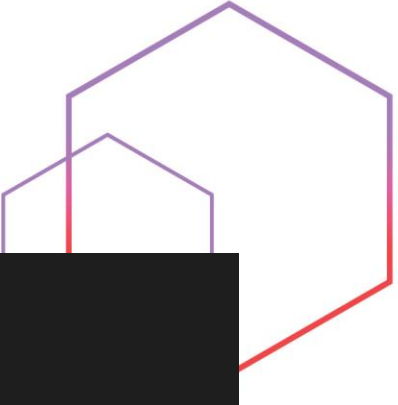
But for where we are right now, it is a very important concept to understand.

Arrays

An array is a special variable, which can hold more than one value.
Arrays can help us create lists of values, as shown below:

```
const students = ["Dani", "Revivo", "Haim"];
```

If you have a list of items, such as a list of student names, for example, storing each student’s name in a single variable would look like this:



```
let student1 = "Dani";
let student2 = "Revivo";
let student3 = "Haim";
```

What if you had not 3 students, but 300?
The solution is an array!

An array can hold many values under a single name, and you can access the values by referring to an index number, as follows:

```
console.log(students[2])
```

Exercise:

What will the above code display?

Read [here](#) why this is happening.

Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

```
const array_name = [item1, item2, ...];
```

It is common practice to declare arrays with the const keyword.
Spaces and line breaks are not important.

Alternatively, you can create an array, and then provide the elements:

Example:

```
const students = [];  
cars[0]= "Dani";  
cars[1]= "Revivo";  
cars[2]= "Haim";
```

From the above example you can deduct how values can be added to the list.
There are additional ways that can be used as well, which we will learn in future lessons.

