JavaScript Objects

In real life, **objects** are things like: houses, cars, people, animals, or any other subjects.

Here is a **car object** example:

Properties	Methods
car.name = Fiat car.model = 500 car.weight = 850kg car.color = white	car.start() car.drive() car.brake() car.stop()

Object Properties

A real life car has **properties** like weight and color: car.name = Fiat, car.model = 500, car.weight = 850kg, car.color = white.

Car objects have the same **properties**, but the **values** differ from car to car.

Object Methods

A real life car has **methods** like start and stop:

car.start(), car.drive(), car.brake(), car.stop().

Car objects have the same **methods**, but the methods are performed **at different times**.

JavaScript Objects

Objects are variables too. But objects can contain many values.

This code assigns **many values** (Fiat, 500, white) to an **object** named car:

Example

```
const car = {type:"Fiat", model:"500", color:"white"};
```

Note: It is a common practice to declare objects with the `const` keyword.

JavaScript Object Definition

Define a JavaScript Object

- Using an Object Literal
- Using the `new` Keyword
- Using an Object Constructor

JavaScript Object Literal

An object literal is a list of **name:value** pairs inside curly braces **{}**.

```
{firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"}
```

Note:

- **name:value pairs** are also called **key:value pairs**.
- **object literals** are also called **object initializers**.

Creating a JavaScript Object

These examples create a JavaScript object with 4 properties:

Examples

```
// Create an Object
const person = {firstName:"John", lastName:"Doe", age:50,
eyeColor:"blue"};
```

Spaces and line breaks are not important. An object initializer can span multiple lines:

```
// Create an Object
const person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```

Below example creates an empty JavaScript object, and then adds 4 properties.

```
// Create an Object
const person = {};
```

```
// Add Properties
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
```

Using the new Keyword

This example create a new JavaScript object using `new Object()`, and then adds 4 properties:

Example

```
// Create an Object
const person = new Object();

// Add Properties
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
```

Note: There is no need to use `new Object()`. For readability, simplicity and execution speed, use the **object literal** method.

Object Properties

The **named values**, in JavaScript objects, are called **properties**.

Property	Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

Accessing Object Properties

You can access object properties in two ways:

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

Examples

```
person.lastName;
person["lastName"];
```

JavaScript Object Methods

Methods are **actions** that can be performed on objects. They are **function definitions** stored as **property values**.

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

Example

```
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**.

Objects are the Commander in JavaScript

1. **Objects** are containers for **Properties** and **Methods**.
2. **Properties** are named **Values**.
3. **Methods** are **Functions** stored as **Properties**.
4. **Properties** can be primitive values, functions, or even other objects.

In JavaScript, almost "everything" is an object.

- Objects are objects
- Maths are objects
- Functions are objects
- Dates are objects
- Arrays are objects
- Maps are objects
- Sets are objects

All JavaScript values, except primitives, are objects.

JavaScript Primitives are Immutable

Primitive values are immutable (they are hardcoded and cannot be changed).

if x = 3.14, you can change the value of x, but you cannot change the value of 3.14.

Value	Type	Comment
"Hello"	string	"Hello" is always "Hello"
3.14	number	3.14 is always 3.14
true	boolean	true is always true
false	boolean	false is always false
null	null (object)	null is always null
undefined	undefined	undefined is always undefined

JavaScript Objects are Mutable

Objects are mutable: They are addressed by reference, not by value.

If person is an object, the following statement will not create a copy of person:

```
const x = person;
```

- The object x is **not a copy** of person. The object x **is** person.
- The object x and the object person share the same memory address.
- Any changes to x will also change person:

Example

```
//Create an Object
const person = {
  firstName:"John",
  lastName:"Doe",
  age:50, eyeColor:"blue"
}

// Try to create a copy
const x = person;

// This will change age in person:
x.age = 10;
```