JavaScript Objects

❮ Previous Next ❯

Real Life Objects

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

Here is a **car object** example:

Car Object	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.

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 Variables

JavaScript variables are containers for data values.

This code assigns a **simple value** (Fiat) to a **variable** named car:

Example

```
let car = "Fiat";
```

Try it Yourself »

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"};
```

Try it Yourself »

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

Learn more about using `const` with objects in the chapter: JS Const.

JavaScript Object Definition

How to 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

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

Try it Yourself »

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"
};
```

Try it Yourself »

This 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";
```

Try it Yourself »

Using the new Keyword

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

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

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

Try it Yourself »

Note:

The examples above do exactly the same.

But, 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**.

lastName	Doe
age	50
eyeColor	blue

Objects written as name value pairs are similar to:

- Associative arrays in PHP
- Dictionaries in Python
- Hash tables in C
- Hash maps in Java
- Hashes in Ruby and Perl

Accessing Object Properties

You can access object properties in two ways:

```
objectName.propertyName
```

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

Examples

```
person.lastName;
```

Try it Yourself »

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

Try it Yourself »

Methods 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;
  }
};
```

Try it Yourself »

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**.

In JavaScript, Objects are King.

If you Understand Objects, you Understand JavaScript.

Methods are **Functions** stored as **Properties**.

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

A **primitive value** is a value that has no properties or methods.

3.14 is a primitive value

A **primitive data type** is data that has a primitive value.

JavaScript defines 7 types of primitive data types:

- `string`
- `number`
- `boolean`
- `null`
- `undefined`
- `symbol`
- `bigint`

Immutable

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

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"
}

// Create a copy
```

Try it Yourself »

Note:

You will learn a lot more about objects in the following chapters.

Exercise ?

Consider the following object:

```
const car = {
  brand: 'Volvo',
  model: 'EX90'
};
```

How many properties do the object have?

- ○ 0
- ○ 1
- ○ 2
- ○ 3

Submit Answer »

COLOR PICKER

FOR BUSINESS CONTACT US

Top Tutorials

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
How To Tutorial
SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
PHP Tutorial
Java Tutorial
C++ Tutorial
jQuery Tutorial

Top References

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference
Angular Reference
jQuery Reference

Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
Java Examples
XML Examples
jQuery Examples

Get Certified

HTML Certificate
CSS Certificate
JavaScript Certificate
Front End Certificate
SQL Certificate
Python Certificate
PHP Certificate
jQuery Certificate
Java Certificate
C++ Certificate
C# Certificate
XML Certificate