Objetos JavaScript

‹ Anterior Próximo ›

Em JavaScript, os objetos são reis. Se você entende objetos, entende JavaScript.

Em JavaScript, quase "tudo" é um objeto.

- Booleanos podem ser objetos (se definidos com a `new` palavra - chave)
- Os números podem ser objetos (se definidos com a `new` palavra - chave)
- Strings podem ser objetos (se definidos com a `new` palavra - chave)
- As datas são sempre objetos
- Matemática são sempre objetos
- Expressões regulares são sempre objetos
- Matrizes são sempre objetos
- Funções são sempre objetos
- Objetos são sempre objetos

Todos os valores JavaScript, exceto primitivos, são objetos.

JavaScript primitivos

Um **valor primitivo** é um valor que não possui propriedades ou métodos.

Um **tipo de dados primitivo** são dados que possuem um valor primitivo.

JavaScript define 5 tipos de tipos de dados primitivos:

- `string`
- `number`
- `boolean`
- `null`
- `undefined`

Os valores primitivos são imutáveis (são codificados permanentemente e, portanto, não podem ser alterados).

se x = 3,14, você pode alterar o valor de x. Mas você não pode alterar o valor de 3,14.

Valor	Modelo	Comente
"Olá"	fragmento	"Olá" é sempre "Olá"
3,14	número	3,14 é sempre 3,14
verdadeiro	boleano	verdade é sempre verdade
falso	boleano	falso é sempre falso
nulo	nulo (objeto)	nulo é sempre nulo
Indefinido	Indefinido	indefinido é sempre indefinido

Objects are Variables

JavaScript variables can contain single values:

Example

```
let person = "John Doe";
```

Try it Yourself »

JavaScript variables can also contain many values.

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

Object values are written as **name : value** pairs (name and value separated by a colon).

Example

```
let person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

Try it Yourself »

A JavaScript object is a collection of **named values**

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

Example

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

Try it Yourself »

Object Properties

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

Property	Value
firstName	John
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

Object Methods

Methods are **actions** that can be performed on objects.

Object properties can be both primitive values, other objects, and functions.

An **object method** is an object property containing a **function definition**.

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

JavaScript objects are containers for named values, called properties and methods.

You will learn more about methods in the next chapters.

Creating a JavaScript Object

With JavaScript, you can define and create your own objects.

There are different ways to create new objects:

- Create a single object, using an object literal.
- Create a single object, with the keyword `new`.
- Define an object constructor, and then create objects of the constructed type.
- Create an object using `Object.create()`.

Using an Object Literal

This is the easiest way to create a JavaScript Object.

Using an object literal, you both define and create an object in one statement.

An object literal is a list of name:value pairs (like age:50) inside curly braces {}.

The following example creates a new JavaScript object with four properties:

Example

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

Try it Yourself »

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

Example

```
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:

Example

```
const person = {};
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
```

Try it Yourself »

Using the JavaScript Keyword new

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

Example

```
const person = new Object();
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
```

Try it Yourself »

The examples above do exactly the same.

Mas não há necessidade de usar `new Object()`.

Para facilitar a leitura, simplicidade e velocidade de execução, use o método literal do objeto.

Objetos JavaScript são mutáveis

Os objetos são mutáveis: eles são endereçados por referência, não por valor.

Se a pessoa for um objeto, a seguinte declaração não criará uma cópia da pessoa:

```
const x = person;  // Will not create a copy of person.
```

O objeto x **não** é **uma cópia** da pessoa. É **uma** pessoa. Tanto x quanto pessoa são o mesmo objeto.

Qualquer mudança em x também mudará a pessoa, porque x e pessoa são o mesmo objeto.

Exemplo

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

const x = person;
x.age = 10;     // Will change both x.age and person.age
```

Tente você mesmo »

‹ Anterior Próximo ›