JS JSON
Depuração JS
Guia de estilo JS
JS Best Practices
JS Erros
JS Performance
Palavras Reservadas JS

Objetos JS
Definições de Objeto
Propriedades do objeto
Métodos de Objeto
Exibição de objeto
Acessores de objeto
Construtores de objetos
Protótipos de objeto
Referência de Objeto
Mapa de Objetos ()
Conjunto de objetos ()

Construtores de objetos JavaScript

❮ Anterior Próximo ❯

Exemplo

```
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eye;
}
```

Tente você mesmo "

É considerada uma boa prática nomear funções de construtor com uma primeira letra maiúscula.

Tipos de objetos (projetos) (classes)

Os exemplos dos capítulos anteriores são limitados. Eles apenas criam objetos únicos.

Às vezes, precisamos de um " **plano** " para criar muitos objetos do mesmo "tipo".

The way to create an "object type", is to use an **object constructor function**.

In the example above, `function Person()` is an object constructor function.

Objects of the same type are created by calling the constructor function with the `new` keyword:

```
const myFather = new Person("John", "Doe", 50, "blue");
const myMother = new Person("Sally", "Rally", 48, "green");
```

Try it yourself »

The **this** Keyword

In JavaScript, the thing called `this` is the object that "owns" the code.

The value of `this`, when used in an object, is the object itself.

In a constructor function `this` does not have a value. It is a substitute for the new object. The value of `this` will become the new object when a new object is created.

Note that `this` is not a variable. It is a keyword. You cannot change the value of `this`.

Adding a Property to an Object

Adding a new property to an existing object is easy:

Example

```
myFather.nationality = "English";
```

Try it Yourself »

The property will be added to myFather. Not to myMother. (Not to any other person objects).

Adding a Method to an Object

Adding a new method to an existing object is easy:

Example

```
myFather.name = function () {
  return this.firstName + " " + this.lastName;
};
```

Try it Yourself »

The method will be added to myFather. Not to myMother. (Not to any other person objects).

Adding a Property to a Constructor

You cannot add a new property to an object constructor the same way you add a new property to an existing object:

Example

```
Person.nationality = "English";
```

Try it Yourself »

To add a new property to a constructor, you must add it to the constructor function:

Example

```
function Person(first, last, age, eyecolor) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eyecolor;
  this.nationality = "English";
}
```

Try it Yourself »

This way object properties can have default values.

Adding a Method to a Constructor

Your constructor function can also define methods:

Example

```
function Person(first, last, age, eyecolor) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eyecolor;
  this.name = function() {
    return this.firstName + " " + this.lastName;
  };
}
```

Try it Yourself »

You cannot add a new method to an object constructor the same way you add a new method to an existing object.

Adding methods to an object constructor must be done inside the constructor function:

Example

```
function Person(firstName, lastName, age, eyeColor) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.age = age;
  this.eyeColor = eyeColor;
  this.changeName = function (name) {
    this.lastName = name;
  };
}
```

The changeName() function assigns the value of name to the person's lastName property.

Now You Can Try:

```
myMother.changeName("Doe");
```

Try it Yourself »

JavaScript knows which person you are talking about by "substituting" **this** with **myMother**.

Built-in JavaScript Constructors

JavaScript has built-in constructors for native objects:

```
new String()    // A new String object
new Number()    // A new Number object
new Boolean()   // A new Boolean object
new Object()    // A new Object object
new Array()     // A new Array object
new RegExp()    // A new RegExp object
new Function()  // A new Function object
new Date()      // A new Date object
```

Try it Yourself »

The `Math()` object is not in the list. `Math` is a global object. The `new` keyword cannot be used on `Math`.

Did You Know?

As you can see above, JavaScript has object versions of the primitive data types `String`, `Number`, and `Boolean`. But there is no reason to create complex objects. Primitive values are much faster:

Use string literals `""` instead of `new String()`.

Use number literals `50` instead of `new Number()`.

Use boolean literals `true / false` instead of `new Boolean()`.

Use object literals `{}` instead of `new Object()`.

Use array literals `[]` instead of `new Array()`.

Use pattern literals `/()/` instead of `new RegExp()`.

Use function expressions `() {}` instead of `new Function()`.

Example

```
let x1 = "";             // new primitive string
let x2 = 0;              // new primitive number
let x3 = false;          // new primitive boolean

const x4 = {};           // new Object object
const x5 = [];           // new Array object
const x6 = /()/          // new RegExp object
const x7 = function(){}; // new function
```

Try it Yourself »

String Objects

Normally, strings are created as primitives: `firstName = "John"`

But strings can also be created as objects using the `new` keyword:
`firstName = new String("John")`

Aprenda por que strings não devem ser criadas como objetos no capítulo JS Strings .

Objetos de número

Normalmente, os números são criados como primitivos: `x = 30`

Mas os números também podem ser criados como objetos usando a `new` palavra-chave:
`x = new Number(30)`

Aprenda por que os números não devem ser criados como objetos no capítulo Números JS .

Objetos Booleanos

Normalmente, os booleanos são criados como primitivos: `x = false`

Mas os booleanos também podem ser criados como objetos usando a `new` palavra-chave:
`x = new Boolean(false)`

Aprenda por que os booleanos não devem ser criados como objetos no capítulo JS Booleans .

❮ Anterior Próximo ❯

Reportar erro Fórum Cerca de Comprar

Principais tutoriais
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

Web Courses
HTML Course
CSS Course
JavaScript Course
Front End Course
SQL Course
Python Course
PHP Course
jQuery Course
Java Course
C++ Course
C# Course
XML Course

Get Certified »