# Construtores de objetos JavaScript

[❮ Anterior](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/js_object_methods.asp&usg=ALkJrhjT3KiMekdZgJzjvqWfMIYEU121kw)[Próximo ❯](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/js_object_prototypes.asp&usg=ALkJrhiSYheN0uMQBuP1DUI84udxl6-iyw)

## Usando um Construtor de Objetos

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

Às vezes, gostamos de ter um "tipo de objeto" que possa ser usado para criar muitos objetos de um tipo.

A maneira padrão de criar um "tipo de objeto" é usar uma função de construtor de objetos:

### Exemplo

function Person(first, last, age, eye) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age;  
    this.eyeColor = eye;  
}  
var myFather = new Person("John", "Doe", 50, "blue");  
var myMother = new Person("Sally", "Rally", 48, "green");

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_object_constructor&usg=ALkJrhj_5HqiWCGioDpYChC99oQ6nE4BMg)

A função acima (Pessoa) é um construtor de objetos.

É considerada uma boa prática para nomear a função do construtor com uma primeira letra maiúscula.

Depois de ter um construtor de objetos, você pode criar novos objetos do mesmo tipo:

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

## A palavra chave

Em JavaScript, o objeto chamado **isso** é o objeto que "possui" o código JavaScript.

O valor **disto** , quando usado em uma função, é o objeto que "possui" a função.

O valor **disso** , quando usado em um objeto, é o próprio objeto.

**Esta** palavra **-** chave em um construtor de objetos não possui um valor. É apenas um substituto para o novo objeto.

O valor **deste** se tornará o novo objeto quando o construtor é usado para criar um objeto.

Observe que **esta** não é uma variável. É uma palavra-chave. Você não pode alterar o valor **disso** .

## Built-in JavaScript Constructors

O JavaScript possui construtores incorporados para objetos nativos:

### Exemplo

var x1 = new Object();    // A new Object object  
var x2 = new String();    // A new String object  
var x3 = new Number();    // A new Number object  
var x4 = new Boolean();   // A new Boolean object  
var x5 = new Array();     // A new Array object  
var x6 = new RegExp();    // A new RegExp object  
var x7 = new Function();  // A new Function object  
var x8 = new Date();      // A new Date object

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_object_builtin&usg=ALkJrhjAlrveIwD3oO-a4sCeolc1-ov6vg)

O objeto Math () não está na lista. A matemática é um objeto global. A nova palavra-chave não pode ser usada em Matemática.

## Você sabia?

Como você pode ver, o JavaScript possui versões de objeto dos tipos de dados primitivos String, Number e Boolean.

Não há motivos para criar objetos complexos. Os valores primitivos executam muito mais rápido.

E não há motivo para usar o novo Array (). Use literais de matriz em vez disso: []

E não há motivo para usar o novo RegExp (). Use literais padrão em vez disso: / () /

E não há motivo para usar a nova Função (). Use as expressões de função em vez disso: function () {}.

E não há motivo para usar o novo Objeto (). Use literais de objetos em vez disso: {}

### Exemplo

var x1 = {};            // new object  
var x2 = "";            // new primitive string  
var x3 = 0;             // new primitive number  
var x4 = false;         // new primitive boolean  
var x5 = [];            // new array object  
var x6 = /()/           // new regexp object  
var x7 = function(){};  // new function object

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_object_best&usg=ALkJrhgRkqsonQxstYYP-P6QrZI4XvBIQA)

## String Objects

Normalmente, as cadeias de caracteres são criadas como primitivas: **var firstName = "John"**

Mas as cadeias de caracteres também podem ser criadas como objetos usando a **nova** palavra-chave: **var firstName = new String ("John")**

Saiba por que as strings não devem ser criadas como objeto no capítulo [JS Strings](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/js_strings.asp&usg=ALkJrhjAzOUB93tq872mMxqSSXWIaDoYJQ) .

## Objetos numéricos

Normalmente, os números são criados como primitivas: **var x = 123**

Mas os números também podem ser criados como objetos usando a **nova** palavra-chave: **var x = new Number (123)**

Saiba por que os números não devem ser criados como objeto no capítulo [JS Numbers](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/js_numbers.asp&usg=ALkJrhj7CM2GlY7w9rR75lynj81dOtIJJQ) .

## Objetos booleanos

Normalmente, os booleanos são criados como primitivas: **var x = false**

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

Saiba por que os booleanos não devem ser criados como objeto no capítulo [JS Booleans](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/js_booleans.asp&usg=ALkJrhgJ9yVhiJDIueZKx5YuQRHnZgBcMQ) .