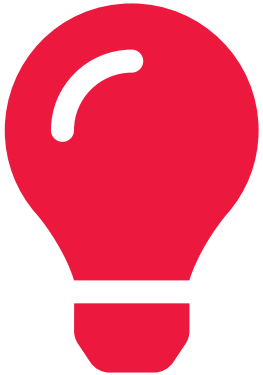
Objetos Literais

Certified Developer
The Ultimate Tech Degree

DigitalHouse >
Coding School

Podemos dizer que **objetos** literais **são representações** em código **de um elemento** da vida real.



Temas

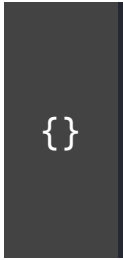
1. Estrutura, propriedades e métodos
2. Funções construtoras

1 Estrutura, propriedades e métodos

Estrutura básica

Um **objeto** é uma estrutura de dados que pode conter **propriedades** e **métodos**.

Para criá-lo, usamos as chaves de abertura e fechamento {}.



```
let carro = {
    placa: 'ADF1238'
};
```

Propriedade

Definimos o nome da **propriedade** do objeto. Neste caso, a propriedade se chama: **placa**.

Dois pontos

Separa o nome da propriedade de seu valor.

Valor

Pode ser **qualquer tipo de dado** que conhecemos. Neste caso, o valor é: **'ADF1238'**

Propriedades de um objeto

Um objeto pode ter quantas propriedades quisermos. Se houver mais de uma, nós as separamos por vírgulas `,`.

Com a notação `objeto.propriedade` acessamos o valor de cada uma delas.

```
{}
let tenista = {
    nome: 'Roger',
    sobrenome: 'Federer'
};

console.log(tenista.nome) // Roger
console.log(tenista.sobrenome) // Federer
```

Métodos de um objeto

Uma propriedade pode armazenar qualquer tipo de dados.

Se uma propriedade armazena uma **função**, diremos que é um **método** do objeto.

```
{}
let tenista = {
    nome: 'Roger',
    idade: 38,
    ativo: true,
    saudacao: function() {
        return 'Olá, me chamo Roger';
    }
};
```

Execução de métodos de um objeto

Para executar um método de um objeto, usamos a notação `objeto.metodo()`.
Os parênteses no final, são o que fazem o método executar.

```
{}
let tenista = {
    nome: 'Roger',
    idade: 38,
    saudacao: function() {
     return 'Olá, me chamo Roger';
    }
};

console.log(tenista.saudacao()); // Olá, me chamo Roger
```

Trabalhando dentro de um objeto

A palavra-chave **this** se refere ao próprio objeto onde estamos, ou seja, o próprio objeto onde escrevemos a palavra.

Com a notação `this.propriedade` acessamos o valor de cada propriedade interna daquele objeto.

```
{}
let tenista = {
    nome: 'Roger',
    sobrenome: 'Federer',
    saudacao: function() { return 'Olá, me chamo ' + this.nome; }
};

console.log(tenista.saudacao()); // Olá, me chamo Roger
```

2 Funções Construtoras

Funções construtoras

O JavaScript nos dá mais uma opção para criar um objeto, por meio do uso de uma **função construtora**.

A função construtora nos permite montar um molde, e então criar todos os objetos de que precisamos.

A função recebe um parâmetro para cada propriedade que queremos atribuir ao objeto.

```
{}
function Carro(marca, modelo){
    this.marca = marca;
    this.modelo = modelo;
};
```

Estrutura de uma função construtora

```
function Carro(marca, modelo){
    this.marca = marca;
    this.modelo = modelo;
};
```

Nome

Definimos um **nome** para a função, que será o nome do nosso **construtor**.

Por convenção, geralmente nomeamos funções construtoras com a **primeira letra maiúscula**. Isso é para diferenciá-los das funções normais.

Estrutura de uma função construtora

```
function Carro(marca, modelo){
    this.marca = marca;
    this.modelo = modelo;
};
```

Parâmetros

Definimos o número de parâmetros que consideramos necessários **para criar** nosso objeto.

Todos **os parâmetros serão obrigatórios** para criar o objeto, a menos que definamos de outra forma.

Estrutura de uma função construtora

```
function Carro(marca, modelo){
    this.marca = marca;
    this.modelo = modelo;
};
```

Propriedades

Com a notação `this.propriedade` definimos a propriedade do objeto que estamos criando naquele momento.

Em geral, os valores das propriedades serão aqueles que vêm por meio dos parâmetros.

Instanciar um objeto

A função construtora **Carro()** espera dois parâmetros: marca e modelo. Para criar um objeto Carro, devemos usar a palavra-chave **new** e chamar a função passando os parâmetros esperados.

```
{}  let meuCarro = new Carro('Ford', 'Falcon');
```

Quando executamos o método **new** para criar um objeto, o que ele retorna é uma instância, ou seja, na variável **meuCarro** teremos uma instância do objeto **Carro** armazenada. Usando a mesma função construtora, podemos instanciar quantos carros quisermos.

```
{}  let outroCarro = new Carro('Chevrolet', 'Corvette');
```


DigitalHouse>
Coding School