

JavaScript

Objetos

Introdução

- ▷ Objetos em Javascript são coleções de pares chave-valor que podem ser utilizados para representar entidades complexas.
- ▷ Um objeto pode ser instanciado de uma das seguintes formas:

```
const objeto1 = {};  
const objeto2 = new Object();
```

- ▷ Em ambos os casos, um objeto vazio (sem propriedades) foi criado.

Propriedades

- ▷ É possível criar propriedades em objetos de diferentes formas:
 - Atribuindo valores a elas. Para isso, basta colocar um ponto após o nome da variável que aponta para o objeto e informar o nome da propriedade, seguido do operador de atribuição e do valor a ser armazenado na propriedade.

```
const obj = {};  
obj.matricula = 123;  
obj.nome = "João";
```

Propriedades

- ▷ É possível criar propriedades em objetos de diferentes formas:
 - Outra forma é a sintaxe abaixo, onde o nome da propriedade é informado em uma string entre colchetes.

```
const obj = {};  
obj["matricula"] = 123;  
obj["nome"] = "João";
```

- Quando esta sintaxe é utilizada, também é possível utilizar como chave ou nome da propriedade uma string que contenha espaços.

Propriedades

- ▷ É possível criar propriedades em objetos de diferentes formas:
 - É possível criar as propriedades e inicializá-las na criação do objeto. Para isso, as propriedades devem ser passadas entre as chaves e o nome da propriedade deve ser separado de seu valor através de dois pontos.

```
const obj = {  
  matricula: 123,  
  nome: "João"  
};
```

Propriedades

- ▷ É possível adicionar propriedades a um objeto que já foi inicializado.

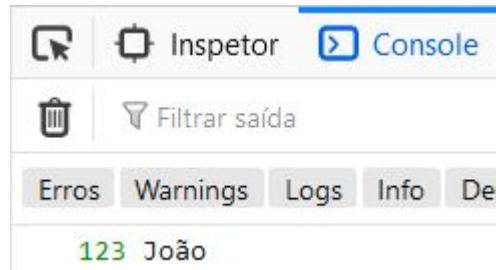
```
const obj = {  
  matricula: 123,  
  nome: "João"  
};  
  
obj.email = "joao@email.com";
```

Propriedades

- ▷ Lendo valores de propriedades:

```
const obj = {  
  matricula: 123,  
  nome: "João"  
};  
  
console.log(obj.matricula, obj.nome);
```

Resultado:



Opções para instanciação

- ▷ Veja esta outra forma de se instanciar um objeto:

```
const matricula = 123;  
const nome = "João";  
  
const obj = {  
  matricula: matricula,  
  nome: nome  
};
```


Opções para instanciação

- ▷ É possível simplificar o código à esquerda da forma apresentada à direita:

```
const matricula = 123;  
const nome = "João";  
  
const obj = {  
    matricula: matricula,  
    nome: nome  
};
```

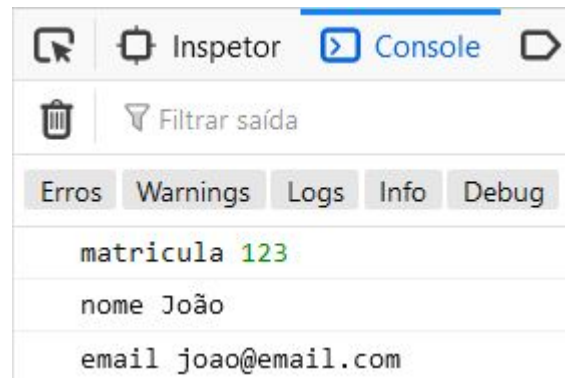
```
const matricula = 123;  
const nome = "João";  
  
const obj = {  
    matricula,  
    nome  
};
```

Percorrendo as propriedades

- ▶ Percorrendo as propriedades de um objeto utilizando o comando *for..in*:

```
const obj = {  
  matricula: 123,  
  nome: "João",  
  email: "joao@email.com"  
};  
  
for (let prop in obj) {  
  console.log(prop, obj[prop]);  
}
```

Resultado:

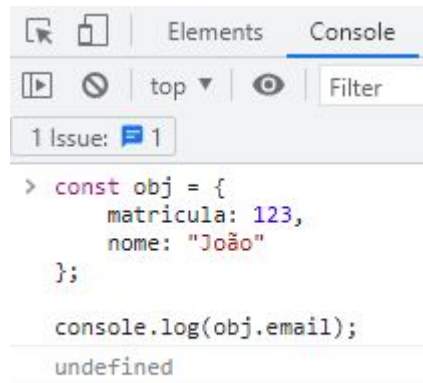


Checando a existência de uma propriedade

- Quando uma propriedade não existe em um objeto Javascript, ao ler o valor da mesma, undefined é retornado.

```
const obj = {  
  matricula: 123,  
  nome: "João"  
};  
  
console.log(obj.email);
```

Resultado:

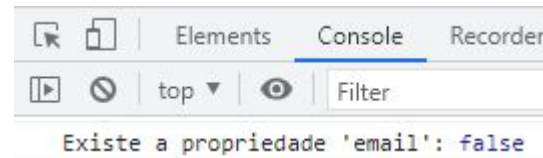


Checando a existência de uma propriedade

- ▶ A conferência abaixo não é suficiente para identificar se o objeto em questão possui a propriedade 'email':

```
const obj = {  
  matricula: 123,  
  nome: "João",  
  email: undefined  
};  
  
console.log("Existe a propriedade  
'email':", obj.email !== undefined);
```

Resultado:

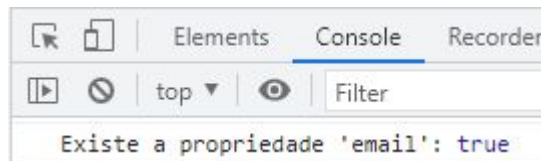


Checando a existência de uma propriedade

- ▷ A forma correta de checar se uma propriedade existe em um objeto é utilizando o operador *in*:

```
const obj = {  
  matricula: 123,  
  nome: "João",  
  email: undefined  
};  
  
console.log("Existe a propriedade  
'email':", "email" in obj);
```

Resultado:



Métodos

- ▶ Além de propriedades, objetos podem conter métodos:

```
const obj = {  
  nome: "João",  
  sobrenome: "da Silva"  
};  
  
obj.metodo = () => console.log("Teste");  
obj.metodo();
```

Métodos

▷ Alternativas:

```
const obj = {  
  nome: "João",  
  sobrenome: "da Silva",  
  metodo: function() {  
    console.log("Teste");  
  }  
};  
  
obj.metodo();
```

```
const obj = {  
  nome: "João",  
  sobrenome: "da Silva",  
  metodo() {  
    console.log("Teste");  
  }  
};  
  
obj.metodo();
```

this

- Podemos implementar métodos que acessam as propriedades, porém a implementação abaixo apresenta erro:

```
const obj = {  
  nome: "João",  
  sobrenome: "da Silva",  
  getNomeCompleto() {  
    return nome + " " + sobrenome;  
  }  
};  
  
obj.getNomeCompleto();
```

Resultado:

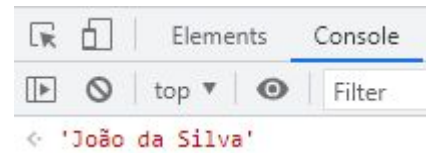
```
✖ ▶ Uncaught ReferenceError: nome is not defined  
    at Object.getNomeCompleto (<anonymous>:5:1)  
    at <anonymous>:9:5
```


this

- Para acessar as propriedades do objeto, deve-se utilizar o **this**:

```
const obj = {  
  nome: "João",  
  sobrenome: "da Silva",  
  getNomeCompleto() {  
    return this.nome + " " + this.sobrenome;  
  }  
};  
  
obj.getNomeCompleto();
```

Resultado:



Object destructuring

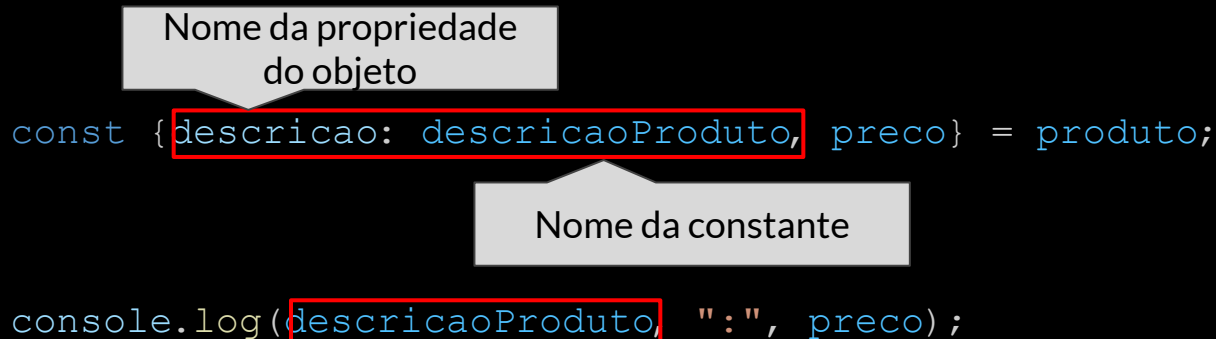
- É possível declarar constantes ou variáveis contendo os valores das propriedades de um objeto seguindo a sintaxe abaixo:

```
const produto = {  
  codigo: 123,  
  descricao: "Produto 1",  
  preco: 12.34  
};  
  
const {descricao, preco} = produto;  
  
console.log(descricao, ":", preco);
```



Object destructuring

- ▷ Se a variável ou constante a ser declarada precisar possuir um nome diferente da propriedade do objeto, é possível fazer desta forma:



The diagram illustrates object destructuring with two code snippets. The first snippet, `const {descricao: descricaoProduto, preco} = produto;`, has a callout box labeled "Nome da propriedade do objeto" pointing to the `descricao` property in the object literal. The second snippet, `console.log(descricaoProduto, ":", preco);`, has a callout box labeled "Nome da constante" pointing to the `descricaoProduto` variable name. In both snippets, the specific names being highlighted (`descricao` and `descricaoProduto`) are enclosed in red rectangular boxes.

```
const {descricao: descricaoProduto, preco} = produto;
```

```
console.log(descricaoProduto, ":", preco);
```

Object destructuring

- Utilizando "...", é possível obter todas as propriedades de um objeto, criando um novo objeto:

```
const p1 = {  
  codigo: 123,  
  descricao: "Produto 1",  
  preco: 12.34  
};  
  
const p2 = {...p1};  
  
p2.preco = 10;  
  
console.log(p1.descricao, p1.preco);  
console.log(p2.descricao, p2.preco);
```

Produto 1 12.34

Produto 1 10

Object destructuring

- Utilizando "...", é possível obter todas as propriedades de um objeto, criando um novo objeto e **atribuindo novas propriedades ou alterando propriedades existentes**:

```
const p1 = {  
  codigo: 123,  
  descricao: "Produto 1",  
  preco: 12.34  
};  
  
const p2 = {...p1, preco: 10};  
console.log(p1.descricao, p1.preco);  
console.log(p2.descricao, p2.preco);
```

Produto 1 12.34

Produto 1 10

Object destructuring

- ▶ Utilizando “...” também é possível obter todas as outras propriedades de um objeto que não foram atribuídas a uma variável ou constante:

```
const produto = {  
  codigo: 123,  
  descricao: "Produto 1",  
  preco: 12.34  
};
```

Descrição: Produto 1

Outras propriedades:

▶ Object { codigo: 123, preco: 12.34 }

```
const {descricao, ...outrasPropriedades} = produto;  
console.log("Descrição:", descricao);  
console.log("Outras propriedades:", outrasPropriedades);
```

Fim de material

Dúvidas?