

Programação Orientada a Objetos

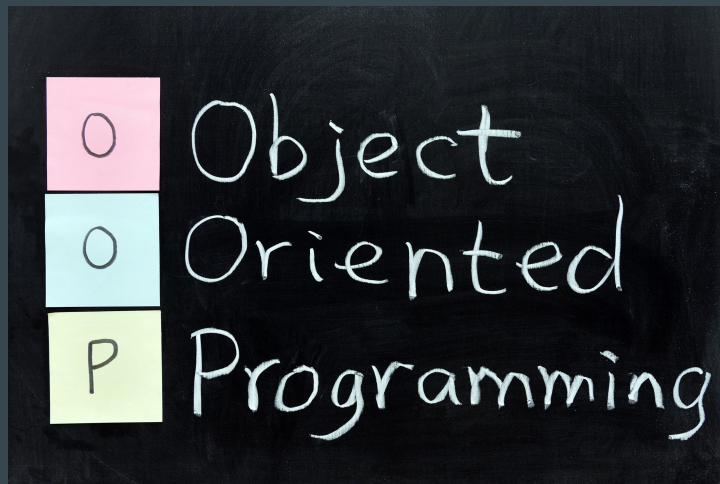
...

M04 - Objetos

M04 - Objetos

1. Objetos

- a. Criação de objetos
- b. Propriedades
- c. Métodos
- d. Iteração de objetos



M04 - Objetos

1. Objetos > Criação de objetos

M04 - Objetos

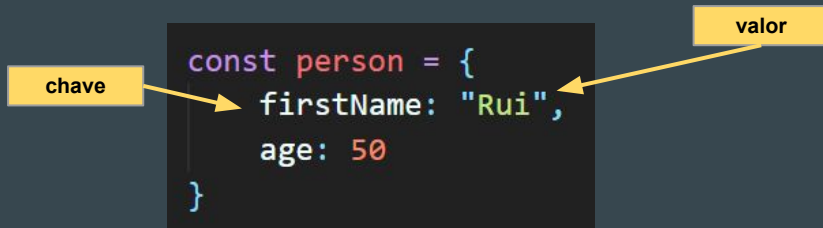
1. Objetos > Criação de objetos

- Até agora só vimos tipos de dados primitivos
 - strings ("Rui Silva")
 - números (3.14)
 - booleanos (true, false)
 - null e undefined
- Um **objeto**
 - um tipo de dados complexo
 - representa uma instância de uma entidade a modelar
 - contém um conjunto de pares de chave-valor

M04 - Objetos

1. Objetos > Criação de objetos

- O conteúdo de um objeto é composto por **propriedades** (separadas por vírgulas)
- As propriedades consistem num par **chave:valor**
 - **chaves** devem ser strings ou símbolos
 - **valores** podem ser de qualquer tipo (incluindo funções, arrays ou outros objetos)



```
const person = {  
  firstName: "Rui",  
  age: 50  
}
```

- Por exemplo, a chave **firstName** tem o valor **"Rui"**
- Objetos podem ser vazios

```
const person = {}
```

M04 - Objetos

1. Objetos > Criação de objetos

- Comparação de Objetos
 - Em JavaScript, os objetos são um **tipo de referência**
 - Dois objetos distintos nunca são iguais, mesmo com as mesmas propriedades
 - Eles apontam para um endereço de memória completamente diferente
 - Objetos que partilham uma referência comum são verdadeiros na comparação

```
const num = 2
const str = "2"

console.log(num == str)    // true
console.log(num === str)  // false
```

```
const obj1 = {name: "Rui"}
const obj2 = {name: "Rui"}

console.log(obj1 == obj2)  // false
console.log(obj1 === obj2) // false
```

```
const obj1 = {name: "Rui"}
const obj2 = obj1

console.log(obj1 == obj2)  // true
console.log(obj1 === obj2) // true
```

M04 - Objetos

1. Objetos > Criação de objetos

- Existem várias formas para criar objetos:
 - a. criar um único objeto através de um **literal** objeto
 - b. criar um único objeto através da palavra-chave `new`
 - c. definir um construtor de objeto, e depois criar objetos do tipo do construtor
 - d. usar **classes** (estudadas mais à frente)

M04 - Objetos

1. Objetos > Criação de objetos

- Criar um literal objeto
 - lista de pares **chave:valor** dentro de {}
 - simples e legível
 - possibilidade de criação do objeto numa única declaração

```
const person = {firstName: "Rui", lastName: "Silva", age: 50, eyeColor: "azul" }  
// ou  
const person = {  
  firstName : "Rui",  
  lastName : "Silva",  
  age : 50,  
  eyeColor : "azul"  
}
```


M04 - Objetos

1. Objetos > Criação de objetos

- Criar um literal objeto
 - baseado em variáveis
- Ou o inverso (**desestruturação**)
 - quebra da estrutura de um objeto
 - pode-se extrair dados de arrays ou objetos em variáveis distintas

```
const firstName = "Rui"  
const age = 50  
  
const person = { firstName, age }
```

```
const emp = {name: "João", age: 22}  
const {name, age} = emp  
  
console.log(name)    // João  
console.log(age)     // 22
```

M04 - Objetos

1. Objetos > Propiedades

M04 - Objetos

1. Objetos > Propriedades

- Um objeto JavaScript é uma coleção de propriedades desordenadas
- As propriedades geralmente pode ser adicionadas, alteradas e removidas
- Sintaxe: `objeto.propriedade`

```
const person = {  
  firstName : "Rui",  
  lastName  : "Silva"  
}
```

acesso a uma propriedade
do objeto person

```
console.log(person.lastName) // Silva
```

M04 - Objetos

1. Objetos > Propriedades

- Sintaxes alternativas:
 - objeto.propriedade
 - objeto["propriedade"]
 - objeto[expressão]

```
const person = {  
  firstName : "Rui",  
  lastName : "Silva"  
}  
  
console.log(person.lastName)  
  
console.log(person["lastName"])  
  
const x = "lastName"  
console.log(person[x])
```

M04 - Objetos

1. Objetos > Propriedades

- Propriedades calculadas (computed properties):
 - Definição de nomes de propriedades à custa do valor de uma variável

computed property

```
const feature = "color"
const car = {
  plate : "12-SA-23",
  [feature] : "azul"
}

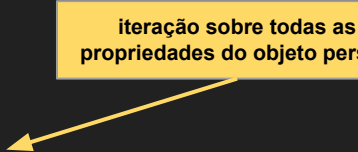
console.log(car.color) // azul
```

M04 - Objetos

1. Objetos > Propriedades

- Iteração:
 - A declaração `for ... in` percorre as propriedades de um objeto
 - O nº de iterações do ciclo é igual ao nº de propriedades
 - Existem outras técnicas, mas esta é a mais rápida!

```
const person = {  
  firstName: "Rui",  
  lastName: "Silva",  
  age: 50  
}  
  
let text = ""  
for (let prop in person) {  
  text += `nome: ${prop} valor: ${person[prop]} \n`  
}  
  
console.log(text)  
  
/*  
  nome: firstName valor: Rui  
  nome: LastName valor: Silva  
  nome: age valor: 50  
*/
```



M04 - Objetos

1. Objetos > Propriedades

- Adição de propriedades:
 - Pode-se adicionar novas propriedades a um objeto existente, basta dar-lhe um valor

```
const person = {  
  firstName: "Rui",  
  lastName: "Silva",  
  age: 50  
}  
  
person.city = "porto"  
  
console.log(person.city) // porto
```

Pode parecer que esta linha causaria um erro, mas não há nenhum problema. Isso é porque **const** contém uma **referência** para o objeto **user**. A linha faz alterações dentro do objeto, mas não mexe na referência.

M04 - Objetos

1. Objetos > Propriedades

- Remoção de propriedades:
 - A palavra-chave **delete** elimina uma propriedade de um objeto

```
const person = {firstName : "Rui", age : 50}  
delete person.age  
console.log(person.age) // undefined
```

- Após a remoção, a propriedade não pode ser utilizada antes de ser adicionada novamente

M04 - Objetos

1. Objetos > Métodos

M04 - Objetos

1. Objetos > Métodos

- Métodos são ações que podem ser executadas em objetos
- Um método JavaScript é uma propriedade que contém uma definição de função

```
const person = {  
  firstName: "Rui",  
  lastName: "Silva",  
  fullName: function () {  
    return `${this.firstName} ${this.lastName}`  
  }  
}
```

// Invocação do método

```
console.log(person.fullName()) // Rui Silva
```

Sintaxe mais abreviada

```
fullName() {  
  return `${this.firstName} ${this.lastName}`  
}
```

M04 - Objetos

1. Objetos > Métodos

- Palavra reservada **this**

```
const person = {  
  firstName: "Rui",  
  lastName: "Silva",  
  fullName: function () {  
    return `${this.firstName} ${this.lastName}`  
  }  
}  
  
// Invocação do método  
console.log(person.fullName()) // Rui Silva
```

Para aceder ao próprio objeto, um método pode usar a palavra-chave **this**

M04 - Objetos

1. Objetos > Iteração

M04 - Objetos

1. Objetos > Iteração

- Pode iterar sobre um conjunto de objetos (por exemplo, dentro de um array)

```
const library = [  
  {  
    title: 'The Road Ahead',  
    author: 'Bill Gates',  
    readingStatus: true  
  },  
  {  
    title: 'Os Lusíadas',  
    author: 'Luis de Camões',  
    readingStatus: false  
  },  
  {  
    title: 'Ensaio Sobre a Cegueira',  
    author: 'José Saramago',  
    readingStatus: true  
  }  
];
```

M04 - Objetos

1. Objetos > Iteração

- Pode iterar sobre um conjunto de objetos: uso do ciclo `for...of`

```
const library = [  
  {  
    title: 'The Road Ahead',  
    author: 'Bill Gates',  
    readingStatus: true  
  },  
  {  
    title: 'Os Lusíadas',  
    author: 'Luis de Camões',  
    readingStatus: false  
  },  
  {  
    title: 'Ensaio Sobre a Cegueira',  
    author: 'José Saramago',  
    readingStatus: true  
  }  
];
```

```
// Imprimir na consola os livros já lidos  
let result = ''  
for (let book of library) {  
  if (book.readingStatus === true) {  
    result += `${book.title} \n`  
  }  
}  
  
console.log(result)  
// The Road Ahead  
// Ensaio Sobre a Cegueira
```