

INTRODUÇÃO A ORIENTAÇÃO À OBJETOS ***(AULA 18)***

CURSO BÁSICO DE PROGRAMAÇÃO COM JAVASCRIPT

MAYARA MARQUES

mmrosatab@gmail.com

SUMÁRIO

- O que é OO?
- Objetos
- Instâncias
- Como criar objetos?
 - Objetos literais
 - Funções construtoras
 - Classes
- Por que usar OO?
- Mão na massa

O QUE É OO?

Orientação a objetos é um ***paradigma de programação*** que se baseia no conceito de ***objetos*** para ***representar os seres e coisas do mundo real***.

OBJETOS

Objetos são entidades que representam algo do mundo real, possuindo **características e comportamentos.**

OBJETOS

Mas como podemos identificar ou saber o que pode ser um objeto?

OBJETOS

Carro



Um carro normalmente apresenta algumas **características e comportamentos.**

OBJETOS

Carro



- Características
 - Motor
 - Marca
 - Cor
 - Ano de fabricação
- Comportamentos
 - Acelerar
 - Frear

OBJETOS

Carro



- Características
 - Motor
 - Marca
 - Cor
 - Ano de fabricação

- Comportamentos
 - Acelerar
 - Frear

Pode-se representar diferentes tipos de carros com essas **características e comportamentos**.



OBJETOS

Pessoa



- Características
 - Nome
 - Idade
 - Profissão
- Comportamentos
 - Andar
 - Falar
 - Dormir

OBJETOS

Pessoa



- Características
 - Nome
 - Idade
 - Profissão

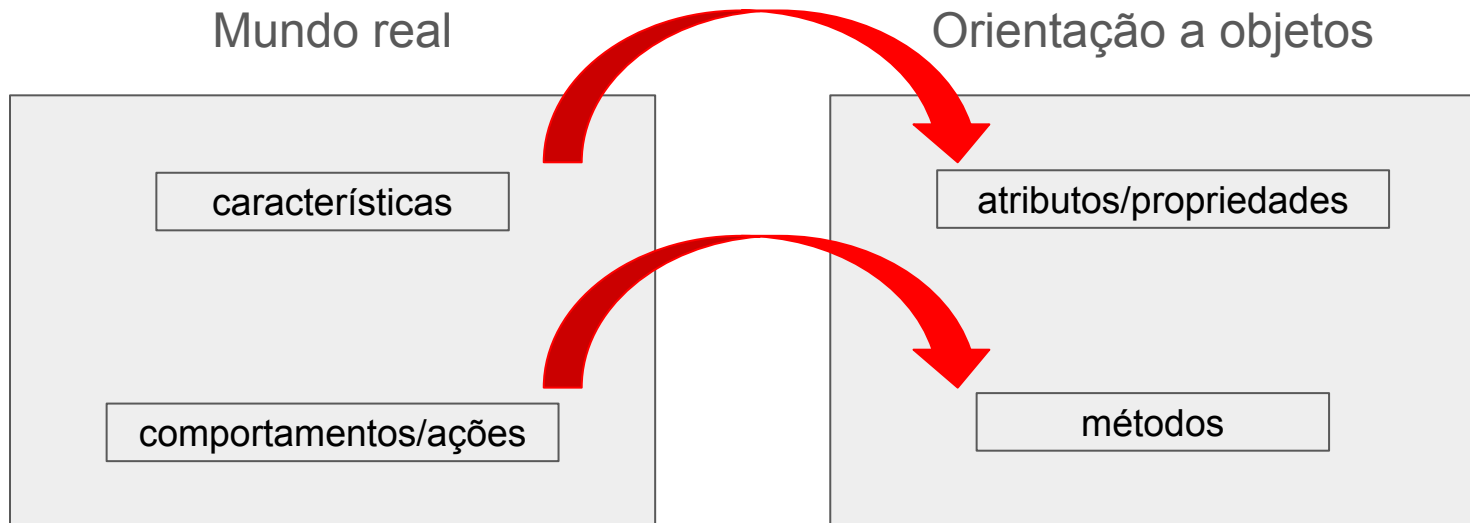
- Comportamentos
 - Andar
 - Falar
 - Dormir
 - Comer

Pode-se representar diferentes tipos de pessoas com essas **características e comportamentos.**



OBJETOS

Dentro da orientação a objetos essas **características** e **comportamentos** são chamados respectivamente de **atributos** e **métodos**.



INSTÂNCIA

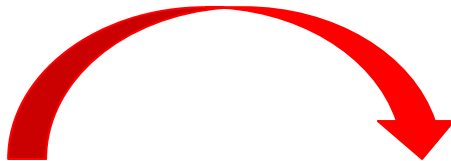
Uma **instância** é um objeto criado a partir de um **modelo especificado**. Nesse modelo foram descritas suas **características** e **comportamentos**.

INSTÂNCIA

Carro



Modelo



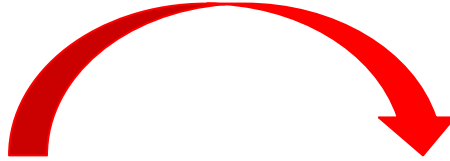
Instância

INSTÂNCIA

Pessoa



Modelo



Instância

OBJETOS

Como criar objetos em JavaScript?

Os objetos em JavaScript podem ser criados de diferentes formas. Veremos algumas como **objetos literais**, **funções construtoras que retornam objetos literais** ou **classes**.

COMO CRIAR OBJETOS EM JAVASCRIPT?

Objetos literais

Propriedades

Método

```
// Objeto literal
const person = {
  name: 'João',
  age: 30,
  greet: function greet() {
    console.log(`Hello, my name is ${this.name}`)
  }
}
```

```
// Acessando propriedades
console.log(person.name) // João
person.greet() // Hello, my name is João
```


COMO CRIAR OBJETOS EM JAVASCRIPT?

Objetos literais

```
// Objeto literal
const person = {
  name: 'João',
  age: 30,
  greet: function greet() {
    console.log(`Hello, my name is ${this.name}`)
  }
}
```

```
// Acessando propriedades
console.log(person.name) // João
person.greet() // Hello, my name is João
```

Simples e direto, contudo, esta forma de criar objetos não permite criar vários objetos semelhantes.

COMO CRIAR OBJETOS EM JAVASCRIPT?

Função construtora

```
// função construtora
function Car(model, year) {
  this.model = model
  this.year = year
  this.on = function() {
    console.log(`The ${this.model} is on!`)
  }
}
```

Funções construtoras, por convenção, são nomeadas com inicial maiúscula, diferente de funções normais.

Em JavaScript, por convenção:

Funções construtoras começam com letra maiúscula → **Car, Person, Animal**, etc.

Funções normais começam com letra minúscula → **greet, calculate, sum**.

COMO CRIAR OBJETOS EM JAVASCRIPT?

Função construtora

```
// função construtora
function Car(model, year) {
  this.model = model
  this.year = year
  this.on = function() {
    console.log(`The ${this.model} is on!`)
  }
}
```

O **this** é utilizado para vincular (ou associar) os atributos e métodos ao objeto que será gerado.

COMO CRIAR OBJETOS EM JAVASCRIPT?

Função construtora

```
// função construtora
function Car(model, year) {
  this.model = model
  this.year = year
  this.on = function() {
    console.log(`The ${this.model} is on!`)
  }
}
```

```
const myCar = new Car('Fusca', 1978)
myCar.on() // The Beetle is on!
```

A palavra reservada **new** em JavaScript pode ser usada para criar uma instância de um objeto baseado em uma função construtora.

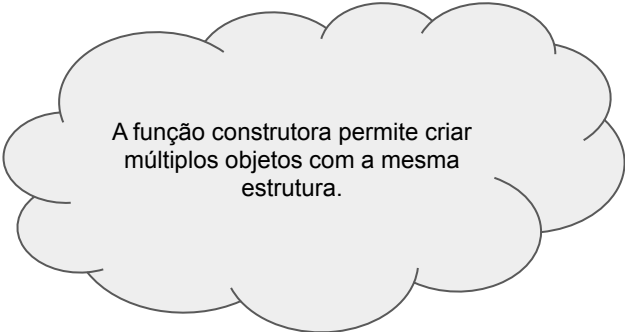
COMO CRIAR OBJETOS EM JAVASCRIPT?

Função construtora

```
// função construtora
function Car(model, year) {
  this.model = model
  this.year = year
  this.on = function() {
    console.log(`The ${this.model} is on!`)
  }
}
```

```
const myCar1 = new Car('Pickup', 1978)
myCar1.on() // The Pickup is on!
```

```
const myCar2 = new Car('Minivan', 2023)
myCar2.on() // The Minivan is on!
```



A função construtora permite criar múltiplos objetos com a mesma estrutura.

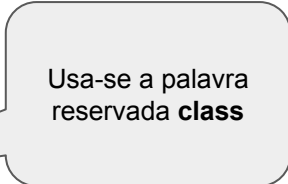
COMO CRIAR OBJETOS EM JAVASCRIPT?

Classes

As classes são uma forma de definir templates para criar objetos. Uma maneira mais moderna de fazer a mesma coisa que é feito com funções construtoras.

COMO CRIAR OBJETOS EM JAVASCRIPT?

Classes



Usa-se a palavra reservada **class**

```
class Person {  
  constructor(name, age) {  
    this.name = name  
    this.age = age  
  }  
  
  speak() {  
    console.log(`Hello, my name is ${this.name}`)  
  }  
}
```

```
const person1 = new Person('Maria', 25)  
person1.speak() // Hello, my name is Maria
```

COMO CRIAR OBJETOS EM JAVASCRIPT?

Classes

O método **constructor** serve para inicializar a instância de um objeto criado a partir da classe. Ele é **chamado automaticamente** quando você cria uma nova instância da classe usando a palavra-chave **new**.

```
class Person {  
  constructor(name, age) {  
    this.name = name  
    this.age = age  
  }  
  
  speak() {  
    console.log(`Hello, my name is ${this.name}`)  
  }  
}
```

```
const person1 = new Person('Maria', 25)  
person1.speak() // Hello, my name is Maria
```


POR QUE USAR OO?

Por que usar OO?

- Modularidade:
 - Divide o código em partes menores e reutilizáveis.
- Reutilização de código:
 - O mesmo código pode ser utilizado em diferentes partes do programa.
- Abstração:
 - Oculta os detalhes da implementação e foca no comportamento.



MÃO NA MASSA

MÃO NA MASSA



1. Crie uma classe chamada "Calculadora" com métodos para realizar operações de soma, subtração, multiplicação e divisão.
2. Crie uma classe chamada "Pessoa" com os seguintes atributos: nome, idade e gênero. Em seguida, crie um método chamado "apresentar" que exiba uma mensagem com os dados da pessoa.
3. Crie uma classe chamada "ContaBancaria" com os atributos: número da conta, saldo e titular. Crie métodos para depositar, sacar e exibir o saldo.
4. Crie uma classe chamada "Retângulo" com os atributos: comprimento e largura. Implemente métodos para calcular a área e o perímetro do retângulo.
5. Crie uma classe chamada "Livro" com os atributos: título, autor e ano de publicação. Crie um método para exibir as informações do livro.

MÃO NA MASSA



6. Crie uma classe chamada "Aluno" com os atributos: nome, matrícula e notas. Implemente um método para calcular a média das notas.
7. Crie uma classe chamada "ConversorTemperatura" com métodos para converter uma temperatura de Celsius para Fahrenheit e vice-versa.
8. Crie uma classe chamada "Animal" com os atributos: nome e som. Implemente um método para reproduzir o som do animal.

DESAFIO



Como programadores, alguns dos nossos desafios são pesquisar, entender e aplicar novos conceitos quase que diariamente. Para nos acostumarmos com esse ritmo, vamos fazer uma pesquisa antes de iniciar o desafio.

Pesquise e estude a class **Date** do JavaScript. Entenda seu funcionamento e principais métodos.

Após a pesquisa e entendimento, você deve elaborar um programa que resolva o problema elucidado no próximo slide.

DESAFIO



Calendário Inteligente

Criar uma classe em JavaScript chamada **CalendarHelper** que encapsule a funcionalidade de obter o nome do dia da semana a partir de uma data.

A classe deve ter um método chamado **getDayName(dateString)**, que recebe uma string representando uma data e retorna o nome do dia da semana correspondente.

A classe deve ser reutilizável, permitindo que possamos criar diferentes instâncias se necessário.

Adicione um método **isWeekend(dateString)** que retorna true se a data for um sábado ou domingo e false caso contrário.

Certifique-se de testar diferentes datas para validar sua implementação.

Prazo: uma semana!!!

DESAFIO



Exemplos de uso da classe



```
const calendar = new CalendarHelper ()
console.log(calendar.getDayName ("2024-08-05")) // Deve imprimir: "Monday"

console.log(calendar.isWeekend ("2024-08-03")) // true (sábado)
console.log(calendar.isWeekend ("2024-08-05")) // false (segunda-feira)
```

REFERÊNCIAS

- Apostila Caelum Estruturação de páginas usando HTML e CSS
- <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript>
- <https://www.w3schools.com/js/default.asp>
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Expressions_and_operators
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date