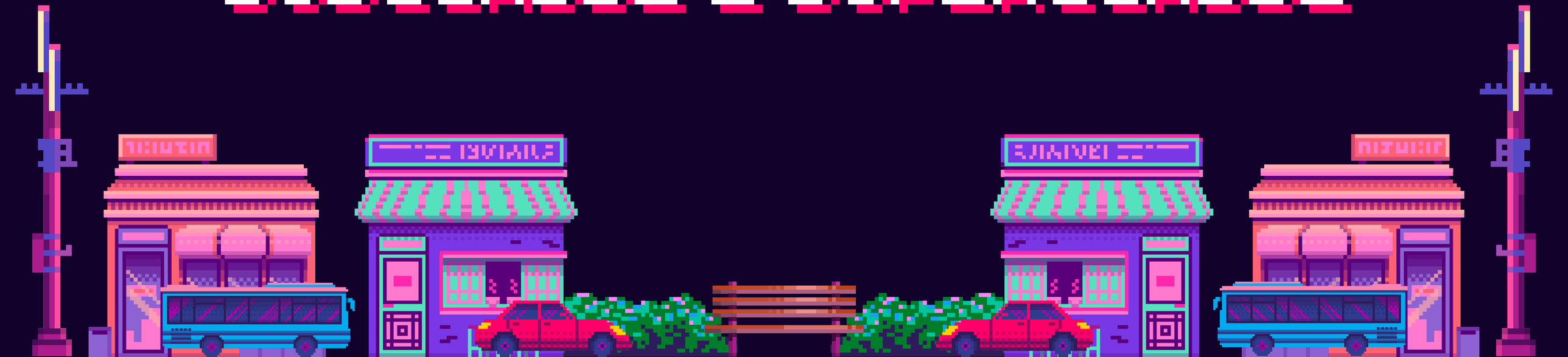

HELLO
WORLD

HERANÇA

SUBCLASSE E SUPERCLASSE



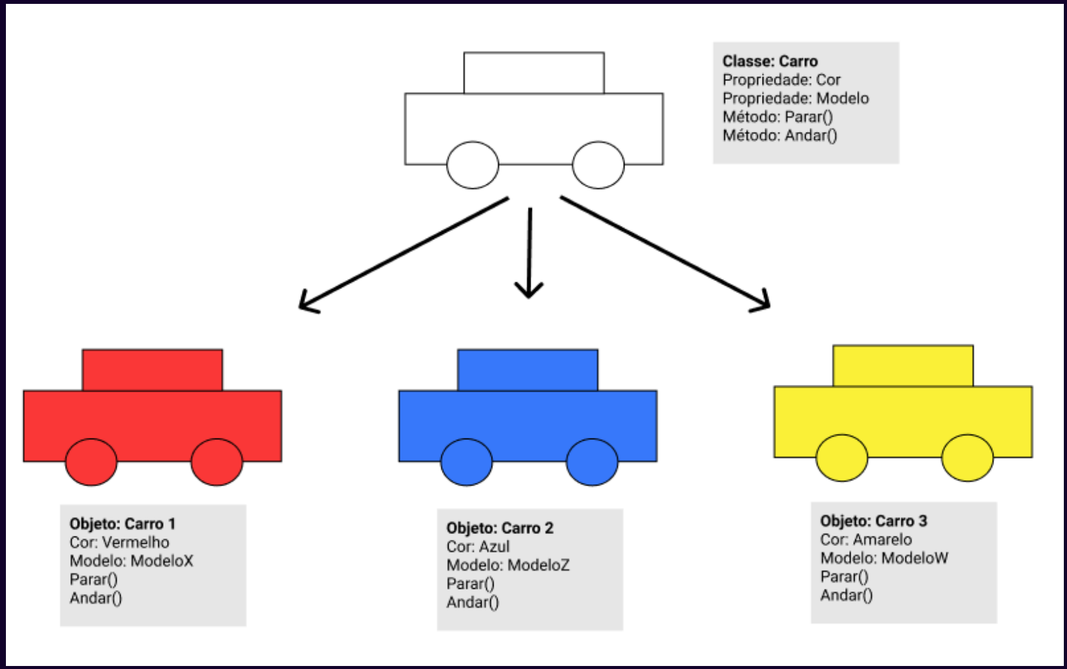
HERANÇA

Herança é um conceito fundamental em orientação a objetos que permite que uma classe (chamada de classe filha ou subclasse) herde propriedades e métodos de outra classe (chamada de classe pai ou superclasse). Isso permite a criação de uma nova classe que é uma extensão da classe existente, reutilizando e especializando funcionalidades.

SABER MAIS

EXEMPLO ABSTRATO: CARROS

Imagine que temos uma classe base chamada **Carro**. Esta classe contém atributos e métodos comuns a todos os carros.

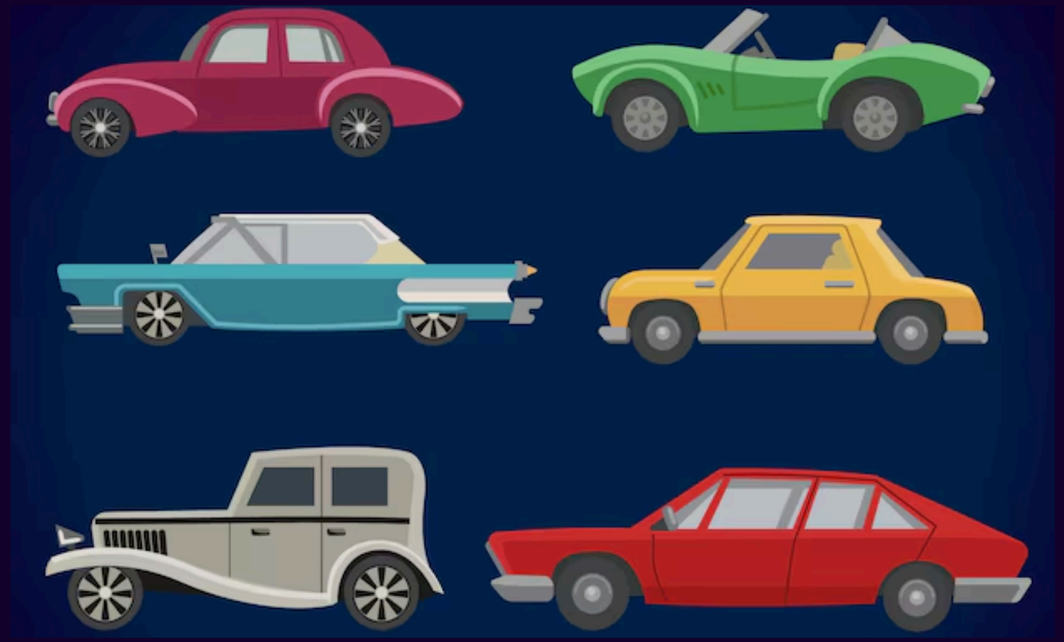



```javascript
class Carro {
  constructor(modelo, cor) {
    this.modelo = modelo;
    this.cor = cor;
  }

  andar() {
    console.log(`${this.modelo} ${this.cor} está andando.`);
  }

  parar() {
    console.log(`${this.modelo} ${this.cor} está parando.`);
  }
}
```

```javascript
// Instanciando os carros
const carroVermelho = new Carro('ModeloX', 'vermelho');
const carroAzul = new Carro('ModeloZ', 'azul');
const carroAmarelo = new Carro('ModeloW', 'amarelo');

// Utilizando os métodos das instâncias
carroVermelho.andar(); // Saída: ModeloX vermelho está andando.
carroVermelho.parar(); // Saída: ModeloX vermelho está parando.

carroAzul.andar(); // Saída: ModeloZ azul está andando.
carroAzul.parar(); // Saída: ModeloZ azul está parando.

carroAmarelo.andar(); // Saída: ModeloW amarelo está andando.
carroAmarelo.parar(); // Saída: ModeloW amarelo está parando.
```

CLASSE DERIVADA: CARROESPORTIVO

Agora, vamos criar uma classe derivada chamada **CarroEsportivo**, que representa um tipo específico de carro, um carro esportivo. Esta classe herda as propriedades e métodos da classe **Carro**, mas também adiciona funcionalidades específicas para carros esportivos.



```
class CarroEsportivo extends Carro {
  constructor(marca, modelo, ano, velocidadeMaxima) {
    super(marca, modelo, ano); // Chama o construtor da classe Carro
    this.velocidadeMaxima = velocidadeMaxima;
  }

  acelerar() {
    console.log(`${this.marca} ${this.modelo} está acelerando a ${this.velocidadeMaxima} km/h.`);
  }
}
```

Na classe **CarroEsportivo**, o método super é usado para chamar o construtor da classe **Carro** e inicializar os atributos herdados. Além disso, adicionamos um novo atributo **velocidadeMaxima** e um método acelerar específico para carros esportivos.

```
const meuCarro = new Carro('Toyota', 'Corolla', 2021);
meuCarro.ligar(); // Saída: Toyota Corolla está ligado.
meuCarro.desligar(); // Saída: Toyota Corolla está desligado.

const meuCarroEsportivo = new CarroEsportivo('Ferrari', '488', 2022, 340);
meuCarroEsportivo.ligar(); // Saída: Ferrari 488 está ligado.
meuCarroEsportivo.acelerar(); // Saída: Ferrari 488 está acelerando a 340 km/h.
```

O meuCarro usa os métodos ligar e desligar da classe Carro, enquanto o meuCarroEsportivo usa esses mesmos métodos, além do método acelerar adicionado pela classe CarroEsportivo.

MODULARIZAÇÃO

A herança permite que a classe CarroEsportivo reutilize o código da classe Carro, evitando duplicação e facilitando a manutenção do código. Carros esportivos têm todas as funcionalidades de um carro comum, mas também podem ter funcionalidades adicionais específicas para sua categoria. Essa abordagem torna o código mais modular e organizado, seguindo o princípio da reutilização e especialização.

SABER MAIS

SUPER()

O método super() é uma parte essencial do sistema de hernça em JavaScript, especialmente quando se trabalha com classes. Ele é utilizado para acessar e chamar métodos da classe pai (ou superclasse) a partir da classe filha (ou subclasse). Aqui está uma explicação detalhada sobre como e por que usar o super():

```
class Animal {
  constructor(nome) {
    this.nome = nome;
  }

  fazerSom() {
    console.log(`${this.nome} faz um som.`);
  }
}

class Cachorro extends Animal {
  constructor(nome, raca) {
    super(nome); // Chama o construtor da superclasse Animal
    this.raca = raca;
  }

  fazerSom() {
    super.fazerSom(); // Chama o método da superclasse Animal
    console.log(`${this.nome} faz au au.`);
  }
}

const meuCachorro = new Cachorro('Rex', 'Labrador');
meuCachorro.fazerSom();
// Saída:
// Rex faz um som.
// Rex faz au au.
```

SOBREESCREVENDO MÉTODOS NA CLASSE

01

A classe Dog sobrescreve o método fazerSom() da classe Animal. Quando a instância de Cachorro chama fazerSom(), a versão de Cachorro é executada em vez da versão da classe base. Isso permite personalizar o comportamento de métodos herdados.
Se você quiser, pode usar o método da classe base dentro do método sobrescrito da classe estendida com a função super().

```
1 class Cachorro extends Animal {
2   fazerSom() {
3     super.fazerSom();  // Chama o método `fazerSom()` da classe `Animal`
4     console.log(`${this.nome} também está latindo!`);
5   }
6 }
```

ADIÇÃO DE NOVOS MÉTODOS NA CLASSE

02

Além de sobrescrever métodos, podemos adicionar novos métodos nas classes estendidas que não existem na classe base, como fizemos com fetch() no exemplo acima. Isso permite que as subclasses tenham funcionalidades adicionais que não estão disponíveis nas classes de onde herdam.

```
class Cachorro extends Animal {
  fazerSom() {
    super.fazerSom();  // Chama o método `fazerSom()` da classe `Animal`
    console.log(`${this.nome} também está latindo!`);
  }

  brincar() {
    console.log(`${this.nome} está brincando com a bola.`);
  }
}
```

VANTAGENS

01

Reutilização de Código:
Você pode criar funcionalidades comuns em uma classe base e reutilizá-las em várias subclasses, evitando duplicação de código.

02

Personalização:
Subclasses podem sobrescrever métodos herdados para adaptar comportamentos sem modificar a classe base.

03

Expansão:
Além de sobrescrever, você pode adicionar novos métodos à subclasse, expandindo suas capacidades.

OBSERVAÇÕES

01 Classes **não** são "hoisted"

02 Uso obrigatório de super() nas subclasses

03 Classes tem strict mode ativado por padrão

04 Métodos em classes não podem ser usados antes da declaração

05 As classes podem ter métodos estáticos definidos

06 Em métodos de classe, o "this" sempre se refere à instância da classe


BORA
CODAR