Dia 8 - Node.js

Classes

Centro de Alta Performance - SECITECI



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 1 / 25

Roteiro

1 Desafio

2 Objetivo da aula

3 Introdução à Orientação a Objetos

4 Definição de Classe

Instanciação de objeto

5 Encapsulamento

6 Herança

7 Polimorfismo

8 Métodos estáticos Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 2 / 25

Desafio

Desafio

Problema: Formas Geométricas

Contexto: Suponha que você esteja desenvolvendo um programa para lidar com formas geométricas simples. Crie uma classe base chamada Forma que tenha um método chamado calcularArea() que retorna a área da forma.



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 3 / 25

Desafio

Desafio

Problema: Formas Geométricas

Descrição do problema: A seguir, crie três classes derivadas da classe Forma: Retângulo, Trapézio e Círculo. Cada uma dessas classes deve herdar o método calcularArea() da classe Forma e calcular suas área de forma específica.



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 4 / 25

Objetivo da aula

Objetivo da disciplina

Entender os conceitos de orientação a objetos.

Saber criar uma classe.

Saber trabalhar com os conceitos de orientação a objetos.



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 5 / 25

Introdução à Orientação a Objetos

Conceitos

A orientação a objetos é um paradigma de programação que se baseia no conceito de objetos.

Um objeto é uma instância de uma classe e representa uma entidade do mundo real, contendo dados e comportamentos associados. A orientação a objetos visa organizar o código em unidades autônomas e reutilizáveis, facilitando o desenvolvimento e manutenção de software.

É amplamente utilizado em várias linguagens de programação, como Java, C++, Python e JavaScript.



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 6 / 25

Introdução à Orientação a Objetos

Princípios da Orientação a Objetos

Encapsulamento: Os dados e comportamentos relacionados são agrupados em um único objeto, que controla o acesso a essas informações.

Herança: Permite criar novas classes a partir de classes existentes, aproveitando atributos e métodos já implementados.

Polimorfismo: Diferentes objetos podem responder de maneira diferente a uma mesma mensagem, permitindo o uso de interfaces comuns.

Abstração: Permite modelar objetos do mundo real em termos de suas características essenciais, ignorando detalhes desnecessários.



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 7 / 25

Introdução à Orientação a Objetos

Exemplo de Orientação a Objetos

Vamos considerar um exemplo simples de orientação a objetos: uma classe "Carro".

A classe "Carro"pode ter atributos como marca, modelo, cor e ano. Também pode ter métodos como "acelerar", "frear"e "ligar/desligar". Cada objeto da classe "Carro"será uma instância única com seus próprios valores para os atributos.

A orientação a objetos permite criar e interagir com vários objetos "Carro"de forma independente.



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 8 / 25

Definição de Classe

Classe

Para criar uma classe em JavaScript, usamos a palavra-chave *class* A classe tem um método construtor que inicializa as propriedades da classe



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 9 / 25

Definição de Classe

Exemplo de Classe

class Carro {

constructor(marca, modelo, ano) {

this.marca = marca;

this.modelo = modelo;

this.ano = ano;

}

acelerar() {

console.log(‘O ${this.modelo} está acelerando!‘); }

}



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 10 / 25

Definição de Classe

Classe

Neste exemplo, definimos uma classe chamada Carro.

A classe tem um construtor que inicializa as propriedades marca, modelo e ano.

Também definimos um método acelerar, que simplesmente é uma função que é definida dentro de uma classe e pode ser invocada em instâncias dessa classe.



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 11 / 25

Definição de Classe Instanciação de objeto

Instanciação

Uma instância de classe em JavaScript é um objeto.

Quando você instancia uma classe, você está criando um novo objeto que herda os métodos e propriedades da classe.



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 12 / 25

Definição de Classe Instanciação de objeto

Instanciação

Em outras palavras, uma instância de classe é uma cópia única da classe que contém seus próprios valores de propriedades.

Para criar uma instância da classe Carro, usamos a palavra-chave new. Aqui está um exemplo:

let meuCarro = new Carro("Ford", "Mustang", 2022);



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 13 / 25

Definição de Classe Instanciação de objeto

Exemplo de Classe

Podemos acessar as propriedades e métodos do objeto da seguinte maneira:

console.log(meuCarro.marca); // Ford

console.log(meuCarro.acelerar()); // O Mustang está aceler



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 14 / 25

Encapsulamento

Encapsulamento - Conceito

O encapsulamento é o conceito de ocultar a complexidade de implementação de um objeto e expor apenas o que é necessário para o uso externo.

Em JavaScript, não há suporte nativo para encapsulamento, mas é possível simular esse conceito usando funções construtoras e closures.



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 15 / 25

Encapsulamento

Exemplo

function ContaBancaria(saldoInicial) {

let saldo = saldoInicial;

this.getSaldo = function() {

return saldo;

};

this.depositar = function(valor) {

saldo += valor;

};

this.sacar = function(valor) {

if (saldo >= valor) {

saldo -= valor;

} else {

console.log("Saldo insuficiente");

} 

};

}

Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 16 / 25

Encapsulamento

Exemplo

let minhaConta = new ContaBancaria(1000);

console.log(minhaConta.getSaldo()); // 1000

minhaConta.depositar(500);

console.log(minhaConta.getSaldo()); // 1500

minhaConta.sacar(2000); // Saldo insuficiente

minhaConta.sacar(1000);

console.log(minhaConta.getSaldo()); // 500

Experimente acessar usando esse código: console.log(minhaConta.saldo);



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 17 / 25

Herança

Herança - Conceito

Quando uma classe estende outra classe, ela herda suas propriedades e métodos

class Animal {

constructor(nome) {

this.nome = nome;

}

fazerBarulho() {

console.log(‘${this.nome} está fazendo barulho.‘); }

}



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 18 / 25

Herança

Exemplo

class Cachorro extends Animal {

constructor(nome, raca) {

super(nome);

this.raca = raca;

}

latir() {

console.log(‘${this.nome} (${this.raca}) está latindo.‘); }

}



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 19 / 25

Herança

Exemplo

Agora, vamos criar um objeto meuCachorro que é uma instância da classe Cachorro:

let meuCachorro = new Cachorro("Fido", "Labrador Retriever");

console.log(meuCachorro.nome); // Fido

console.log(meuCachorro.raca); // Labrador Retriever

// Fido está fazendo barulho.

console.log(meuCachorro.fazerBarulho());

// Fido (Labrador Retriever) está latindo.

console.log(meuCachorro.latir());



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 20 / 25

Polimorfismo

Exemplo

Polimorfismo é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura), mas comportamentos distintos.

class Animal {

constructor(nome) {

this.nome = nome;

}

fazerBarulho() {

console.log(‘${this.nome} está fazendo barulho.‘);

}

}



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 21 / 25

Polimorfismo

Exemplo

class Cachorro extends Animal {

constructor(nome, raca) {

super(nome);

this.raca = raca;

}

fazerBarulho() {

console.log(‘${this.nome} (${this.raca}) está latindo.‘);

}

}

class Gato extends Animal {

constructor(nome, cor) {

super(nome);

this.cor = cor;

}

fazerBarulho() {

console.log(‘${this.nome} (${this.cor}) está miando.‘);

}

} Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 22 / 25

Polimorfismo

Exemplo

Agora, vamos criar um exemplo de uso do polimorfismo com as classes Cachorro e Gato:

const cachorro = new Cachorro(’Rex’, ’Golden Retriever’); const gato = new Gato(’Miau’, ’preto’);

cachorro.fazerBarulho();

// Saída: "Rex (Golden Retriever) está latindo."

gato.fazerBarulho();

// Saída: "Miau (preto) está miando."



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 23 / 25

Métodos estáticos

Conceito

Métodos estáticos são métodos que pertencem à classe em si, em vez de pertencerem a uma instância da classe.

class Utilitarios {

static formatarNumero(numero) {

return numero.toLocaleString("pt-BR");

}

}

console.log(Utilitarios.formatarNumero(1000)); // 1.000



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 24 / 25

Vamos ao desafio !



Classes (Centro de Alta Performance - SECITECI) Dia 8 - Node.js 25 / 25