Estruturas da Linguagem TypeScript

Já foi mostrado as estruturas abordadas na disciplina de Estruturas de Dados, agora vamos abordar algumas estruturas da linguagem TypeScript.

Classes

Uma classe no TypeScript representa um objeto. Elas permitem definir propriedades e métodos, e também podem ser estendidas e instanciadas para criar objetos com comportamentos e estados específicos.

```
class Pessoa {
  // Propriedades
  public nome: string; // public - pode ser acessada de fora da classe
  private cpf: string; // private - não pode ser acessada de fora da classe
  protected idade: number; // protected - apenas pela classe e subclasses

  constructor(nome: string, cpf: string, idade: number) {
    this.nome = nome;
    this.idade = idade;
    this.cpf = cpf;
  }

  // Metodos
  public exibirInformacoes(): void {
    console.log(`Nome: ${this.nome}`);
    console.log(`Idade: ${this.idade}`);
    console.log(`CPF: ${this.cpf}`);
  }

  // Getters e Setters
  get seuCPF(): string {
    return this.cpf;
  }

  set seuCPF(cpf: string) {
    if (cpf.length === 14) this.cpf = cpf;
  }
}

const pessoa1 = new Pessoa("Yan", "123.456.789-10", 20);
pessoa1.exibirInformacoes();
pessoa1.seuCPF = "123.456.789-11";
```

Interface

Uma interface no TypeScript representa um contrato que uma classe ou objeto deve seguir. Ela serve para definir o formato e o comportamento de um objeto.

```
interface Moto {
  marca: string;
  modelo: string;
  cilindrada: number;
  exibirDetalhes(): void;
}

const moto1: Moto = {
  marca: "Yamaha",
  modelo: "MT-07",
  cilindrada: 689,

  exibirDetalhes(): void {
    console.log(
      `Moto: ${this.marca} - ${this.modelo}, Cilindrada: ${this.cilindrada} cc`
    );
  },
};

// Tambem podemos criar uma classe que implementa a interface
class classeMoto implements Moto {
  marca: string;
  modelo: string;
  cilindrada: number;

  constructor(marca: string, modelo: string, cilindrada: number) {
    this.marca = marca;
    this.modelo = modelo;
    this.cilindrada = cilindrada;
  }

  exibirDetalhes(): void {
    console.log(
      `Moto: ${this.marca} - ${this.modelo}, Cilindrada: ${this.cilindrada} cc`
    );
  }
}

const moto2: classeMoto = new classeMoto("Yamaha", "R15", 155);

moto1.exibirDetalhes(); // Moto: Yamaha - MT-07, Cilindrada: 689 cc
moto2.exibirDetalhes(); // Moto: Yamaha - R15, Cilindrada: 155 cc
```

Type

Os type são similares às interfaces, mas permitem definições mais complexas, como uniões e interseções.

```
type Pessoa = {
  nome: string;
  idade: number;
};

type Programador = {
  linguagem: string;
  experiencia: number;
};

type Desenvolvedor = Pessoa & Programador;

const desenvolvedor1: Desenvolvedor = {
  nome: "Yan",
  idade: 20,
  linguagem: "TypeScript",
  experiencia: 1,
};
```

Enum

Os enum nos permitem definir um conjunto de constantes que representam um conjunto finito de valores. Elas ajudam a manter o código mais organizado e legibilidade.

```
enum DiasDaSemana {
  Segunda = 1,
  Terca = 2,
  Quarta = 3,
  Quinta = 4,
  Sexta = 5,
  Sabado = 6,
  Domingo = 7,
}

console.log(DiasDaSemana.Segunda); // 1
```