Arrays, Union Types e Interfaces

Miguel Costa
miguelangcosta@gmail.com

Objetivos

- Aprender a usar **arrays tipados**
- Introduzir **interfaces** para organizar objetos
- Explorar **Union Types** e variáveis opcionais (?)
- Preparar para **classes e objetos** na próxima aula
- Continuar aplicando conceitos no **gestor de tarefas**

Arrays tipados

Em TypeScript, podemos declarar **arrays com tipos específicos**.

```
let tarefas: string[] = ["Comprar pão", "Estudar TS"];
let numeros: number[] = [1, 2, 3];
```

- Um array de string **não pode conter números**
- Um array de number **não pode conter strings**

Boas práticas: sempre tipar arrays para que o TypeScript detecte erros cedo.

Union Types

Ciências ULisboa

Union Types permitem que uma variável aceite **mais de um tipo**, útil quando o valor pode variar entre tipos específicos, como string ou number.

```
let id: number | string;

id = 123;      // OK
id = "abc123";  // OK
// id = true;   // ❌ Erro: boolean não é permitido
```

Dica: Usar Union Types quando a variável pode ter valores diferentes, mas limitados a tipos conhecidos.

Arrays com Union

Ciências ULisboa

Podemos ter arrays que aceitam **mais de um tipo de valor** usando |:

```
let ids: (string | number)[] = [1, "abc", 3];
```

Cada elemento pode ser **string ou number**

Útil para listas de IDs, inputs variados ou valores opcionais

Objetos e Interfaces

Ciências ULisboa

O que são interfaces?

- Uma **interface** é como um **molde para objetos**.
- Define **quais propriedades e tipos** um objeto deve ter, mas **não cria código executável**.
- Serve para garantir que objetos sigam uma estrutura consistente.

```typescript
interface Tarefa {
  titulo: string;
  prioridade: "alta" | "media" | "baixa";
  concluida?: boolean; // ? = opcional
}

let tarefa1: Tarefa = {
  titulo: "Estudar TypeScript",
  prioridade: "alta",
};
```

Interfaces são **tipo “contratos”**.

Não geram código JS, apenas ajudam o TypeScript a verificar tipos.

Variáveis opcionais (?)

Ciências ULisboa

- Marcar propriedades como **opcionais** permite flexibilidade
- Útil quando nem todo objeto precisa ter todos os campos

```
interface Tarefa {
  titulo: string;
  descricao?: string;
}

let tarefa2: Tarefa = { titulo: "Aprender TS" };
```

descricao pode estar presente ou não

Evita erro de "propriedade não existe"

Interfaces

- Uma interface **não contém implementação**, mas pode **definir que um objeto terá funções**.
- Apenas declara a assinatura da função, **sem corpo**.

```typescript
interface Tarefa {
  id: number;
  titulo: string;
  concluida: boolean;
  marcarConcluida(): void; // apenas assinatura
}
```

```typescript
const tarefa: Tarefa = {
  id: 1,
  titulo: "Estudar TS",
  concluida: false,
  marcarConcluida() {
    this.concluida = true;
  }
};
```

A interface define **como o objeto deve ser**, mas **não cria comportamento sozinho**.

Um objeto que siga a interface deve **implementar essa função**

Classes

Ciências ULisboa

É um **molde** ou **modelo** que define:

- **Propriedades** (dados)
- **Métodos** (funções/comportamentos)
- Não ocupa memória até que seja criada uma instância.
- Serve para criar **vários objetos com a mesma estrutura e comportamento**.

```
class Tarefa {
  id: number;
  titulo: string;
  concluida: boolean = false;

  constructor(id: number, titulo: string) {
    this.id = id;
    this.titulo = titulo;
  }

  marcarConcluida() {
    this.concluida = true;
  }
}
```

Questões