6. Interfaces e Tipos Customizados

TypeScript oferece poderosas ferramentas para a definição de estruturas de dados complexas e contratos de função através de interfaces e tipos customizados. Neste capítulo, vamos explorar como essas definições podem ser usadas para tipar parâmetros de funções, garantindo que os dados passados e retornados sejam rigorosamente verificados.

Definindo Interfaces

1. **Interface para Parâmetros de Função**
    - **Defina uma interface que especifique a estrutura esperada para os parâmetros de uma função.**

```typescript
interface Usuario {
    nome: string;
    idade: number;
}

function imprimeUsuario(usuario: Usuario): void {
    console.log(`Nome: ${usuario.nome}, Idade: ${usuario.idade}`);
}
```

2. **Interface com Método Opcional**
    - **Interfaces podem incluir métodos opcionais, o que aumenta a flexibilidade na implementação.**

```typescript
interface Produto {
    nome: string;
    preco: number;
    descrever?(): string;
}

function etiquetaProduto(produto: Produto): string {
    return produto.descrever ? produto.descrever() : `${produto.nome} custa R$${prod
uto.preco}`;
}
```

Usando Tipos (Type Aliases)

- **Type Alias para Funções**

- **Type aliases podem ser usados para definir assinaturas de funções, tornando o código mais limpo e organizado.**

```
type ProcessadorPagamento = (quantia: number, conta: string) => boolean;

function processarPagamento(processador: ProcessadorPagamento, quantia: number, con
ta: string) {
    return processador(quantia, conta);
}
```

- **Combinando Tipos com Uniões**
  - **Utilize uniões para criar tipos que podem aceitar múltiplas formas de dados.**

```
type EntradaFormulario = string | number;

function entradaDados(campo: string, valor: EntradaFormulario) {
    console.log(`Entrada para ${campo}: ${valor}`);
}
```

Exercícios para Praticar

1. **Interface para Funções:**
   - Defina uma interface `Contato` com propriedades para `email` e `telefone`. Crie uma função que aceite um `Contato` e imprima os detalhes de contato.
2. **Usando Type Alias em Funções:**
   - Crie um type alias `OperacaoMatematica` que define uma função que aceita dois números e retorna um número. Implemente funções para soma, subtração, multiplicação e divisão usando este alias.
3. **Interface com Métodos Opcionais:**
   - Crie uma interface `Configuracao` com uma propriedade opcional `background` (string). Escreva uma função que defina a configuração de um aplicativo e trate a ausência de `background`.
4. **Funções com Type Alias de União:**
   - Implemente uma função que aceite `string` ou `string[]` como entrada e retorne uma string sempre, tratando adequadamente a entrada caso seja um array.
5. **Parâmetros Complexos com Interfaces:**
   - Defina uma interface `Jogo` com `nome`, `preco` e um método opcional `jogar`. Crie uma função que aceite um `Jogo` e imprima uma descrição ou convide o usuário a jogar, se disponível.