Funções em TypeScript: Guia Completo

Por Escola Dnc

Introdução

O TypeScript é uma linguagem que estende o JavaScript, adicionando recursos poderosos como tipagem estática. Neste ebook, vamos explorar em detalhes como criar e utilizar funções em TypeScript, aproveitando seus recursos avançados para escrever código mais robusto e seguro.

Configuração do Ambiente

Antes de começarmos, é importante ter o ambiente configurado corretamente:

- Certifique-se de ter o Node.js instalado
- Instale o TypeScript globalmente: `npm install -g typescript`
- Configure seu editor de código (recomendamos o Visual Studio Code)

Para compilar arquivos TypeScript, use o comando:

```
npx tsc nomeDoArquivo.ts
```

Isso gerará um arquivo JavaScript correspondente.

Fundamentos de Funções em TypeScript

Declaração Básica de Funções

Em TypeScript, as funções são declaradas de forma similar ao JavaScript, mas com a adição de tipos:

```
function nomeDaFuncao(parametro1: tipo, parametro2: tipo): tipoDeRetorno {  // corpo da função  return valor;}
```

Exemplo:

```
function saudacao(nome: string): void {  console.log(`Olá, ${nome}!`);}
```

Tipos de Parâmetros e Retorno

- **Parâmetros:** Podem ser tipados como `string`, `number`, `boolean`, etc.
- **Retorno:** Use `void` para funções sem retorno, ou especifique o tipo de retorno (ex: `string`, `number`)

Importante: O TypeScript infere automaticamente o tipo de retorno se não especificado, mas é uma boa prática declará-lo explicitamente.

Funções com Múltiplos Parâmetros

Você pode adicionar quantos parâmetros forem necessários:

```
function desejarBoasVindas(nome: string, idade: number, internado: boolean): void {  console.log(`Boas-vindas, ${nome}!`);
```

```
console.log(`Você tem ${idade} anos de idade.`);
console.log(internado ? "Você está internado." : "Você não está
internado.");}
```

Recursos Avançados de Funções

Parâmetros Opcionais

Use o símbolo `?` para marcar parâmetros como opcionais:

```
function obterMensagemBoasVindas(nome: string, idade: number,
telefoneFixo?: string): string {  let mensagem = `Boas-vindas,
${nome}! Você tem ${idade} anos de idade.`;    if (telefoneFixo) {
mensagem += ` Seu número de telefone fixo é ${telefoneFixo}.`;  }
return mensagem;}
```

Funções com Retorno de Valor

Quando uma função retorna um valor, especifique o tipo de retorno:

```
function calcularIdade(anoNascimento: number): number {  const
anoAtual = new Date().getFullYear();  return anoAtual -
anoNascimento;}
```

Uso de Template Strings

Template strings permitem interpolação de variáveis de forma mais legível:

```
function criarMensagemPersonalizada(nome: string, idade: number):
string {  return `Olá, ${nome}! Você tem ${idade} anos.`;}
```

Boas Práticas e Dicas

1. **Nomeação clara:** Use nomes descritivos para funções e parâmetros.
2. **Funções puras:** Sempre que possível, crie funções que não alterem estado externo.
3. **Evite efeitos colaterais:** Funções devem fazer uma coisa e fazê-la bem.
4. **Uso de tipos:** Aproveite ao máximo o sistema de tipos do TypeScript.
5. **Documentação:** Use comentários JSDoc para documentar funções complexas:

```
/** * Calcula o IMC (Índice de Massa Corporal) * @param peso Peso em kg * @param altura Altura em metros * @returns O valor do IMC calculado */function calcularIMC(peso: number, altura: number): number {  return peso / (altura * altura);}
```

6. **Testes:** Escreva testes unitários para suas funções para garantir seu funcionamento correto.

Exemplos Práticos

Exemplo 1: Função de Cálculo

```
function calcularArea(largura: number, altura: number): number {
return largura * altura;}const areaRetangulo = calcularArea(5,
3);console.log(`A área do retângulo é: ${areaRetangulo}`);
```

Exemplo 2: Função com Lógica Condicional

```
function classificarIMC(imc: number): string {  if (imc < 18.5)
return "Abaixo do peso";  if (imc < 25) return "Peso normal";  if
(imc < 30) return "Sobrepeso";  return "Obeso";}const classificacao
= classificarIMC(27.5);console.log(`Classificação do IMC:
${classificacao}`);
```

Exemplo 3: Função com Parâmetros Opcionais e Padrão

```
function criarPerfil(nome: string, idade: number, profissao?:
string, cidade: string = "Não informada"): string {  let perfil =
`Nome: ${nome}, Idade: ${idade}`;  if (profissao) perfil += `,
Profissão: ${profissao}`;  perfil += `, Cidade: ${cidade}`;  return
perfil;}console.log(criarPerfil("Ana", 28,
"Engenheira"));console.log(criarPerfil("Carlos", 35));
```

Conclusão

Funções em TypeScript oferecem uma maneira poderosa e segura de estruturar seu código. Com a tipagem estática, você pode prevenir erros comuns e melhorar a manutenibilidade do seu projeto. Pratique criando funções com diferentes níveis de complexidade e explore os recursos avançados do TypeScript para aprimorar suas habilidades de desenvolvimento.

Lembre-se: a prática leva à perfeição. Quanto mais você trabalhar com funções em TypeScript, mais confortável e proficiente você se tornará. Continue explorando e experimentando para aproveitar ao máximo o que o TypeScript tem a oferecer!