

Vitor Amadeu Souza

Introdução ao

TypeScript

Parte IV

© 2025 by Cerne Tecnologia e Treinamento Ltda.

© 2025 by Vitor Amadeu Souza

Nenhuma parte desta publicação poderá ser reproduzida sem autorização prévia e escrita de **Cerne Tecnologia e Treinamento Ltda.** Este livro publica nomes comerciais e marcas registradas de produtos pertencentes a diversas companhias. O editor utiliza as marcas somente para fins editoriais e em benefício dos proprietários das marcas, sem nenhuma intenção de atingir seus direitos.

Fevereiro de 2025

Direitos reservados por:

Cerne Tecnologia e Treinamento Ltda

Produção: Cerne Tecnologia e Treinamento

E-mail da Empresa: cerne@cerne-tec.com.br

Home Page: www.cerne-tec.com.br.com.br

Atendimento ao Consumidor: sac@cerne-tec.com.br

Contato com o Autor: vitor@cerne-tec.com.br



FEITO NO BRASIL

“Conhecimento é poder.”

Francis Bacon

Cerne Tecnologia

A Cerne Tecnologia tem uma equipe preparada para desenvolvimento de projetos eletrônicos em diversas áreas: Médica, Entretenimento, Industrial, Robótica, Científica, Automobilística, Aeronáutica, etc. Trabalhamos com tecnologia microcontrolada usando o PIC, ARM, AVR, 8051, dsPIC, PIC24, PIC32 além do Arduino, Raspberry, Beaglebone etc. Desenvolvemos o projeto desde sua concepção até a entrega do produto final, passando pelas etapas de esquema elétrico, protótipo e desenvolvimento de circuito impresso.

Desenvolvemos aplicativos para smartphones/tablets Android, iOS, Blackberry, Windows Phone e no desenvolvimento de softwares a nível PC para plataforma Windows, usando ferramentas como o Visual Basic, C# e C++.

Atuamos na parte de montagem de placas, onde podemos fornecer ambos os serviços de desenvolvimento de projetos e produção ou apenas um destes.

Desenvolvemos esquemas elétricos e layout de PCI, tanto em tecnologia convencional como SMD.

Temos a flexibilidade de customizar um de nossos produtos, de modo a atender a uma necessidade específica do cliente, tornando o custo de desenvolvimento menor se comparado a construção de um projeto desde a sua fase inicial.

Desenvolvemos e fornecemos kits didáticos para diversos microcontroladores além de apostilas, livros e e-books.

Na hora de desenvolver um projeto ou equipar seu laboratório não hesite em nos contatar. Entre em contato conosco através do endereço cerne-tec.com.br para obter mais informações.



cerne-tec.com.br

Sumário

Capítulo I – Metodologia de desenvolvimento	6
1. Introdução	6
 Capítulo II – Programação em TypeScript	 7
1. Modificadores de Acesso (private, public, protected)	7
2. Type Alias	8
3. Funções Genéricas	9
4. Constraints em Genéricos	11
5. Mapeamento de Tipos	12
6. Nullable Types	13
7. Tipo never	14
8. Literal Types	16
9. Type Inference (Inferência de Tipos)	17
10. Optional Chaining	18
11. Nullish Coalescing Operator	19

Capítulo I

Metodologia de desenvolvimento

1. Introdução

Este livro tem como objetivo ensinar o desenvolvimento web utilizando TypeScript, explorando seus recursos de forma prática e aplicada. Ao longo desta série, diversas aplicações serão desenvolvidas, permitindo uma compreensão aprofundada da linguagem e de suas principais funcionalidades. Cada volume abordará temas específicos, conforme detalhado no sumário, proporcionando um aprendizado progressivo e estruturado.

Capítulo II

Programação em TypeScript

1. Modificadores de Acesso (`private`, `public`, `protected`)

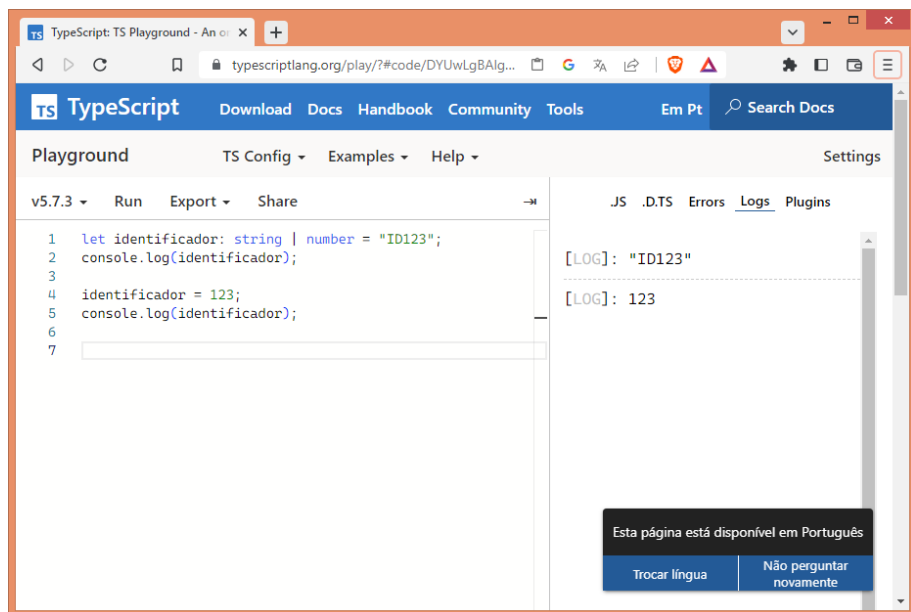
Este exemplo em TypeScript declara uma variável chamada `identificador` que pode armazenar valores do tipo `string` ou `number`, devido ao uso do operador de união (`string | number`). Inicialmente, a variável é atribuída com um valor do tipo `string` ("ID123") e, em seguida, ela recebe um valor do tipo `number` (123). No momento da execução, o `console.log` imprime o valor da variável, primeiro mostrando a string "ID123", e depois o número 123. O TypeScript permite essa flexibilidade, pois a variável foi definida com um tipo que aceita mais de uma possibilidade de valor.

```
let identificador: string | number = "ID123";

console.log(identificador);

identificador = 123;

console.log(identificador);
```



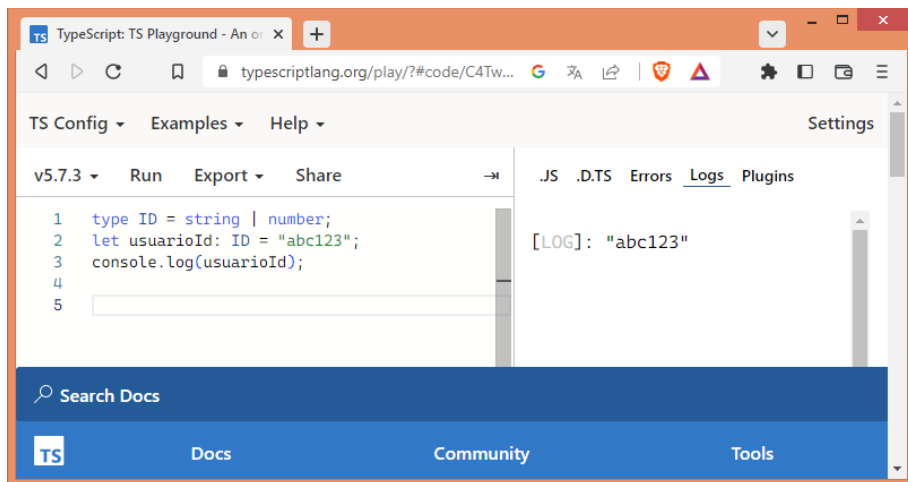
2. Type Alias

O código define um tipo personalizado chamado ID, que pode ser tanto string quanto number. Em seguida, a variável `usuarioid` é declarada com esse tipo e recebe inicialmente o valor `"abc123"`, que é uma string. Quando o código é executado, o `console.log` exibe `"abc123"` no console. O uso de um alias de tipo (`type ID = string | number;`) facilita a reutilização desse tipo em diferentes partes do código, tornando-o mais legível e modular.

```
type ID = string | number;

let usuarioId: ID = "abc123";

console.log(usuarioId);
```



3. Funções Genéricas

O código define uma função genérica chamada `identidade<T>`, onde `T` representa um tipo genérico que pode ser qualquer tipo de dado. A função recebe um parâmetro chamado `valor` e simplesmente o retorna, garantindo que o tipo de entrada seja o mesmo do retorno. Ao chamar `identidade(10)`, o TypeScript infere que `T` é do tipo `number`, então a função retorna `10`. Da mesma forma, ao chamar `identidade("Olá")`, o tipo inferido é `string`, e a função retorna `"Olá"`.

No console, serão exibidos os valores 10 e "Olá", respectivamente. Esse uso de genéricos torna a função flexível e reutilizável, mantendo a segurança de tipos.

```
function identidade<T>(valor: T): T {  
    return valor;  
}  
  
let numero = identidade(10);  
console.log(numero);  
  
let texto = identidade("Olá");  
console.log(texto);
```

