

Vitor Amadeu Souza

Introdução ao

TypeScript

Parte II

© 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.



Sumário

Capítulo I – Metodologia de desenvolvimento	6
1. Introdução	6
 Capítulo II – Programação em TypeScript	 7
1. Função com Tipo de Retorno	7
2. Função com Parâmetros Opcionais	9
3. Função com Parâmetros Default	11
4. Função anônima	13
5. Tipos de Objetos	14
6. Interfaces	15
7. Classes	16
8. Herança em Classes	18

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. Função com Tipo de Retorno

Neste exemplo, temos uma função chamada `somar`, que recebe dois parâmetros do tipo `number`: `a` e `b`. A função retorna a soma desses dois números. A função é definida da seguinte forma: `function somar(a: number, b: number): number`. Isso significa que ela espera receber dois números como entrada e retornará também um número. Dentro da função, a expressão simplesmente adiciona os valores de `a` e `b` usando o operador de soma (+) e retorna o resultado.

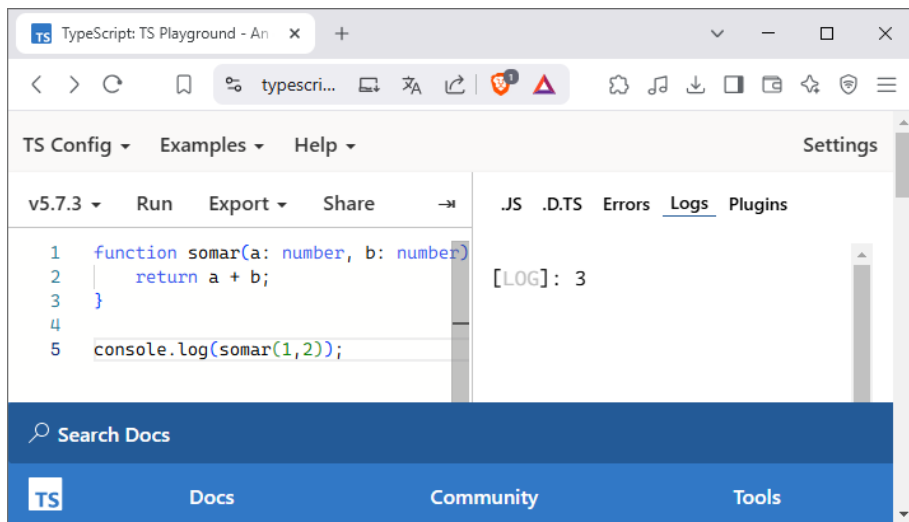
Quando você chama a função com os argumentos 1 e 2 — ou seja, `console.log(somar(1, 2))` — a saída será "3". Isso ocorre porque a função realiza a operação matemática corretamente e retorna essa soma como resultado.

Essa abordagem é útil para criar funções reutilizáveis que realizam cálculos específicos. Além disso, o uso explícito dos tipos (`number`) ajuda na manutenção e segurança do código ao garantir

que apenas valores numéricos sejam passados para a função. Caso tentasse passar um valor não numérico (como uma string), TypeScript alertaria sobre um erro de tipo antes mesmo da execução do código.

```
function somar(a: number, b: number): number
{
    return a + b;
}

console.log(somar(1,2));
```



2. Função com Parâmetros Opcionais

No exemplo há uma função chamada `saudacao`, que recebe dois parâmetros: `nome` e `idade`. O parâmetro `nome` é obrigatório e deve ser do tipo `string`, enquanto o parâmetro `idade` é opcional (indicado pelo símbolo de interrogação após seu nome) e deve ser do tipo `number`.

A função retorna uma mensagem de saudação personalizada baseada nos valores fornecidos. Se a `idade` for fornecida (ou seja, se ela não for `null` ou indefinida), a função inclui essa informação na mensagem; caso contrário, ela apenas menciona o nome.

Quando a função com os argumentos `"João"` e `10` — ou seja, `console.log(saudacao("João", 10))` — a saída será `"Olá, João, você tem 10 anos."`. Isso ocorre porque a condição dentro da expressão ternária (`idade ? ... : ...`) é verdadeira quando há um valor para `idade` (neste caso, `10`), então ela inclui essa informação na mensagem.

Essa abordagem permite criar funções flexíveis que podem lidar com diferentes conjuntos de dados sem exigir todos os parâmetros sempre. Além disso, o uso explícito dos tipos (`string` e opcionalmente

number) ajuda na manutenção e segurança do código ao garantir que apenas valores compatíveis sejam passados para a função.

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

console.log(saudacao("João", 10));
```

