

Vitor Amadeu Souza

Introdução ao

TypeScript

Parte III

© 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. Modificadores de Acesso (private, public, protected)	7
2. Classes Abstratas	9
3. Função com Parâmetros Default	11
4. Função anônima	13
5. Tipos de Objetos	14
6. Interfaces	15
7. Getter e Setter	16
8. Modificadores de Classe (readonly)	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

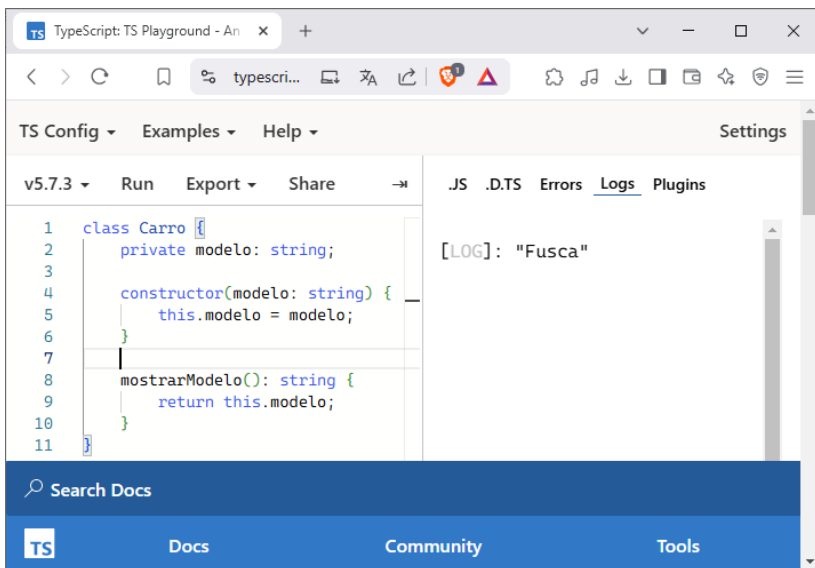
1. Modificadores de Acesso (`private`, `public`, `protected`)

O código define uma classe chamada `Carro`, que possui uma propriedade `modelo` do tipo `string`. Essa propriedade é declarada como privada (`private`), o que significa que só pode ser acessada dentro da própria classe. No construtor, a propriedade `modelo` é inicializada com o valor passado como argumento. Para permitir o acesso ao `modelo` do carro, a classe fornece um método público chamado `mostrarModelo()`, que retorna o valor da propriedade privada.

Ao instanciar um objeto carro com o modelo "Fusca" e chamar `console.log(carro.mostrarModelo())`, a saída será "Fusca". Esse exemplo demonstra o conceito de encapsulamento no TypeScript, garantindo que os atributos internos da classe só sejam acessados por meio de métodos específicos.

```
class Carro {  
    private modelo: string;
```

```
        constructor(modelo: string) {  
            this.modelo = modelo;  
        }  
  
        mostrarModelo(): string {  
            return this.modelo;  
        }  
    }  
  
    let carro = new Carro("Fusca");  
    console.log(carro.mostrarModelo());
```



2. Classes Abstratas

O código define uma classe abstrata chamada `Forma`, que contém um método abstrato `area()`. Como essa classe é abstrata, ela não pode ser instanciada diretamente e serve apenas como um modelo para outras classes que a estendem. A classe `Circulo` herda de `Forma` e implementa o método `area()`, que retorna a área do círculo usando a fórmula $\pi \times \text{raio}^2$. No construtor, o raio é recebido como parâmetro e atribuído à propriedade `raio`.

Ao instanciar `circulo` com `raio 10` e chamar `console.log(circulo.area())`, a saída será `314.1592653589793`, correspondente à área do círculo com esse raio. Esse exemplo demonstra o uso de classes abstratas e herança no TypeScript, garantindo que todas as formas que estendam `Forma` implementem o método `area()`.

```
abstract class Forma {  
    abstract area(): number;  
}  
  
class Circulo extends Forma {  
    raio: number;
```

```

    constructor(raio: number) {

        super();

        this.raio = raio;

    }

    area(): number {

        return Math.PI * this.raio ** 2;

    }

}

let circulo = new Circulo(10);

console.log(circulo.area());

```

