

# Programação Front-end – AULA 20

Matheus Moresco

Engenharia de Software - 3º Período

2025/01

# Introdução

- Compreender o que é o TypeScript e como ele se diferencia do JavaScript.
- Identificar as principais vantagens do uso de TypeScript em projetos frontend.
- Escrever código básico em TypeScript.
- Compilar TypeScript para JavaScript.
- Configurar e utilizar o TypeScript em projetos reais.

# O que é o TypeScript?

- O **TypeScript** é uma linguagem de programação de código aberto desenvolvida pela Microsoft, que é um **superconjunto do JavaScript** — ou seja, tudo que você pode fazer em JavaScript também pode ser feito em TypeScript, **mas com funcionalidades adicionais**, principalmente relacionadas à **tipagem estática**.

**TypeScript = JavaScript + Tipagem Estática + Recursos de Linguagens Fortemente Tipadas (como Java ou C#)**



# Diferenças entre JavaScript e TypeScript

- Tipagem dinâmica vs. tipagem estática
- Inferência de tipos
- Interfaces e tipos personalizados
- Classes e visibilidade (public, private, etc.)

# Tipagem Estática

- Em JavaScript, você pode escrever:

```
let idade = 25;  
idade = "vinte e cinco"; // Isso é válido, mas pode causar erro em tempo de execução
```

- Em TypeScript:

```
let idade: number = 25;  
idade = "vinte e cinco"; // ✗ Erro em tempo de compilação
```

# Verificação de Erros

- **JavaScript:** Os erros só são detectados **em tempo de execução**.

```
function soma(a, b) {  
    return a + b;  
}  
  
soma(1, "2"); // resultado: "12"
```

- **TypeScript:** Os erros são detectados **em tempo de compilação**.

```
function soma(a: number, b: number): number {  
    return a + b;  
}  
  
soma(1, "2"); // ✗ erro detectado antes de rodar
```

# Orientação a Objetos Avançada

## JavaScript

- Suporte a classes e objetos, mas com limitações.

```
class Pessoa {  
  constructor(nome) {  
    this.nome = nome;  
  }  
}
```

## TypeScript

- Suporte completo a OOP com **modificadores de acesso, interfaces, tipos genéricos, etc**

```
class Pessoa {  
  private nome: string;  
  
  constructor(nome: string) {  
    this.nome = nome;  
  }  
  
  public getNome(): string {  
    return this.nome;  
  }  
}
```

# Compilação

- **JavaScript:**
  - Não precisa de compilador. É interpretado diretamente pelo navegador.
- **TypeScript:**
  - **Precisa ser compilado** para JavaScript antes de ser executado pelo navegador.

```
tsc arquivo.ts # gera um arquivo arquivo.js
```



# Suporte de IDE e Autocompletar

- **JavaScript:**

- Suporte limitado a autocompletar, pois os tipos não são explícitos.

- **TypeScript:**

- Suporte avançado em editores como VS Code: autocompletar, sugestão de código, refatoração inteligente, etc.

# Interfaces e Tipos Personalizados

- **JavaScript:**

- Não possui suporte nativo a interfaces.

- **TypeScript:**

- Permite criar **interfaces e tipos personalizados** para representar melhor os dados:

```
interface Usuario {  
    nome: string;  
    idade: number;  
}  
  
const user: Usuario = { nome: "João", idade: 30 };
```

# Por que usar TypeScript no frontend?

- **Mais segurança e menos bugs:** detecta erros antes mesmo de rodar o código.
- **Melhor manutenção:** facilita refatorações em grandes projetos.
- **Melhor documentação automática:** o uso de tipos deixa o código mais compreensível.
- **Melhor integração com frameworks modernos,** como Angular (que usa TypeScript por padrão).

# Configurando o Ambiente

- Instalar o **Node.js** (vem com o npm):
  - Baixe em: <https://nodejs.org/>
- Instalando o TypeScript
  - Comando para instalação global: *npm install -g typescript*
- Criar um projeto
- Criar o **tsconfig.json**
  - Esse arquivo diz ao TypeScript como compilar o projeto.
  - Comando: *tsc --init*

# Programando com TypeScript

- Organizamos o projeto da seguinte forma:

```
meu-projeto-ts/  
├─ src/           ← Onde ficará o código TypeScript  
├─ dist/          ← Onde será gerado o JavaScript  
├─ index.html     ← Arquivo HTML principal  
├─ tsconfig.json  ← Configuração do TypeScript  
└─ package.json   ← Configuração do projeto Node (opcional)
```

# Programando com TypeScript

- Criaremos arquivos do tipo .ts na pasta 'src'.

- src/index.ts:

```
const nome: string = "TypeScript";  
console.log(`Olá, ${nome}!`);
```

- No arquivo de index.htm, carregamos os arquivos .js gerados pelo compilador

- Index.htm:

```
<body>  
  <h1>Hello TypeScript</h1>  
  <script src="../dist/index.js"></script>  
</body>
```

# Compilando o Código

- Compile manualmente
  - Isso vai compilar todos os arquivos .ts da pasta src/ para .js na pasta dist/.
  - Comando: `tsc`
- Compilar no modo automático:
  - O compilador vai ficar "assistindo" mudanças e recompilando em tempo real.
  - Comando: `tsc -watch`

## Principais comandos

Ação	Comando
Criar projeto npm	<i>npm init -y</i>
Instalar TypeScript globalmente	<i>npm install -g typescript</i>
Criar tsconfig.json	<i>tsc --init</i>
Compilar manualmente	<i>tsc</i>
Compilar automaticamente	<i>tsc --watch</i>



## Exemplo pratico

- Criar um projeto com o npm
- Criar um arquivo .tsconfig
- Compilar os arquivos .ts para .js
- Criar uma página de index.html para usar os scripts

## Exercício

- Em uma página com os campos Nome e comentário.
- Usar o typescript para mostrar os comentários na tela.
- Código inicial no repositório git em 'Exemplos praticos\pagina\_comentários'
- Código funcional usando JavaScript. Transformar em TypeScript.
- Envio via Forms: <https://forms.gle/4GdQug94kMwTofJ4A>
- Prazo: 22/05/2025 as 23:59
- Em grupo: no máximo 4 pessoas