Introdução

Com esse guia você vai ser capaz de dar os seus primeiros passos práticos no mundo da programação com TypeScript.

Importante:

Para tirar melhor proveito desse guia, é bom que você já tenha uma boa base de JavaScript básico, caso você não tenha recomendo fortemente fazer primeiro as OT's de Javascript.

Sistema CPV

O sistema CPV é um sistema de 3 passos para aprender programação e aplicar os conceitos aprendidos. É o método que eu usei quando estava começando na programação e que uso até hoje para aprender novas tecnologias.

CPV Significa:

Conceito → Aprender um conceito, a teoria;

Pratica → Praticar o conceito aprendido com exercícios ou projetos práticos;

Vitrine → Expor o que você aprendeu para o mundo, colocar em um repositório do GitHub, fazer um post no Linkedin, enfim, faz uma vitrine do seu conhecimento.

Recomendo fortemente que você aplique esse sistema para tirar o melhor proveito.

1. Introdução ao TypeScript

O que é TypeScript?

TypeScript é uma linguagem de programação desenvolvida pela Microsoft que adiciona tipos estáticos ao JavaScript. Isso significa que, enquanto o JavaScript é dinâmico e mais flexível, o TypeScript ajuda a capturar erros antes mesmo da execução do código, durante a fase de desenvolvimento!

Com TypeScript, você pode escrever aplicações mais robustas e com menos bugs. É especialmente útil em projetos grandes, onde a complexidade e o número de desenvolvedores envolvidos tornam o controle de qualidade mais desafiador.

Por que usar TypeScript?

1. **Segurança de tipo: Reduz os bugs em tempo de execução ao verificar os tipos durante a compilação.**
2. **Ferramentas de desenvolvimento: Aproveite autocompletação poderosa, refatoração e muito mais.**
3. **Compatibilidade com JavaScript: O TypeScript é um superset de JavaScript, então todo código JavaScript é também um código TypeScript válido.**
4. **Adoção na comunidade: Grandes frameworks como Angular, React e Vue.js têm adotado e recomendado o uso de TypeScript para projetos de grande escala.**

Um breve exemplo de código

Vamos ver uma rápida comparação para entender o poder do TypeScript:

JavaScript:

```
function soma(a, b) {
    return a + b;
}

console.log(soma(5, "3"));  // Resultado: "53"
```

TypeScript:

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

console.log(soma(5, 3));  // Resultado: 8
// O TypeScript emitiria um erro se tentássemos passar uma string aqui.
```

No exemplo TypeScript, definimos tipos para os parâmetros e para o retorno da função. Isso ajuda a evitar erros como a concatenação acidental de números e strings, um problema comum em JavaScript.

Esse é o começo da jornada com TypeScript! Com esses fundamentos, você já começa a perceber as vantagens de adicionar tipos ao JavaScript.

2. Configurando um Ambiente TypeScript

Instalação do Node.js e NPM

Antes de instalar o TypeScript, você precisa ter o Node.js e o NPM (Node Package Manager) instalados no seu sistema. O Node.js é um runtime de JavaScript que permite executar código JS no seu computador, e o NPM é o gerenciador de pacotes que usaremos para instalar o TypeScript.

1. Baixe e instale o Node.js: Visite nodejs.org e baixe a versão recomendada para a maioria dos usuários.
2. Verifique a instalação: Abra seu terminal e digite os seguintes comandos para verificar se o Node.js e o NPM foram instalados corretamente: Você deverá ver as versões instaladas exibidas no terminal.

```
node -v
npm -v
```

Instalando o TypeScript

Com o Node.js e o NPM prontos, instalar o TypeScript é um processo simples:

```
npm install -g typescript
```

Este comando instala o TypeScript globalmente em seu sistema, permitindo que você o use em qualquer projeto.

Configurando o compilador TypeScript

O próximo passo é configurar o compilador TypeScript. Isso é feito através de um arquivo chamado `tsconfig.json`, que define como o TypeScript compila o código JS.

1. Crie um novo projeto: Crie uma nova pasta para o seu projeto e navegue até ela no terminal.
2. Inicialize um projeto Node.js: Isso cria um arquivo `package.json` padrão.

```
npm init -y
```

3. Crie um arquivo `tsconfig.json`:
    - Você pode gerar um `tsconfig.json` padrão com o seguinte comando:

```
tsc --init
```

- Abra o `tsconfig.json` e ajuste as configurações conforme necessário. As mais importantes incluem:
    - `"target": "es6"`: Define a versão do ECMAScript para a saída do JS.
    - `"module": "commonjs"`: Define o sistema de módulos.
    - `"strict": true`: Ativa todas as restrições de tipo para uma segurança máxima.

Exemplo de Projeto TypeScript

Para garantir que tudo está funcionando, vamos criar um simples script TypeScript:

1. Crie um arquivo `index.ts`:
    - Adicione o seguinte código:

```
let message: string = "Hello, TypeScript!";
console.log(message);
```

Compile seu código:

```
tsc index.ts
```

Isso deve criar um arquivo `index.js` compilado a partir do seu TypeScript.

Execute o script:

```
node index.js
```

Você deve ver "Hello, TypeScript!" sendo impresso no terminal.

UniSENAI

Atividade

1. **O que é TypeScript e quais são suas principais vantagens em relação ao JavaScript? Cite pelo menos duas vantagens.**

2. **Analise o seguinte código em TypeScript e explique qual é a diferença em relação ao equivalente em JavaScript:**

```typescript
function multiplicar(a: number, b: number): number {
    return a * b;
}
console.log(multiplicar(4, "5"));
```

3. **Quais os passos necessários para instalar o TypeScript e configurar um novo projeto? Liste pelo menos três etapas.**

4. **O que é o arquivo `tsconfig.json` e qual a sua importância na configuração do TypeScript? Cite pelo menos duas configurações que podem ser definidas nesse arquivo.**

5. **Depois de criar um arquivo TypeScript (`index.ts`), quais comandos você deve executar para compilar e rodar o código? Descreva brevemente o que acontece em cada etapa.**