INTRODUÇÃO AO TYPESCRIPT

MARCOS CAMARGO

DESCRIÇÃO

TypeScript é uma linguagem de programação desenvolvida pela Microsoft que adiciona tipagem estática ao JavaScript. É projetada para facilitar a escrita de códigos robustos e escaláveis, especialmente em projetos de grande escala. Este e-book irá guiá-lo pelos fundamentos do TypeScript, suas principais características e como utilizá-lo em seus projetos.

Índice

O que é TypeScript?

TypeScript é um superconjunto de JavaScript que introduz tipos estáticos opcionais, permitindo aos desenvolvedores detectar erros de tipo durante o desenvolvimento, antes de executar o código. Ele transpila para JavaScript, o que significa que você pode usá-lo em qualquer ambiente onde JavaScript é executado.

Vantagens do TypeScript

- Tipagem Estática: Permite detectar erros de tipo em tempo de compilação.
- Melhor Integração de IDE: Ferramentas como VS Code oferecem excelente suporte ao TypeScript, com autocompletar e verificação de tipos.
- Maior Manutenibilidade: Código mais fácil de entender e refatorar.
- Compatibilidade: TypeScript é totalmente compatível com JavaScript, permitindo a integração gradual em projetos existentes.

Instalação e Configuração

Pré-requisitos

- Node.js instalado

Instalação
Para instalar o TypeScript, você pode usar o npm (Node Package Manager):

```
npm install -g typescript
```

Configuração
Você pode inicializar um projeto TypeScript criando um arquivo tsconfig.json:

```
tsc --init
```

Exemplo de um arquivo tsconfig.json básico:

```
{
  "compilerOptions": {
    "target": "es6",
    "module": "commonjs",
    "strict": true,
    "outDir": "./dist",
    "rootDir": "./src"
  }
}
```

Tipos Básicos

Tipos Primitivos

- TypeScript oferece os mesmos tipos primitivos que o JavaScript, mas com a adição de tipagem estática.

```
let isDone: boolean = false;
let age: number = 30;
let userName: string = "John";
let list: number[] = [1, 2, 3, 4];
```

Tipos Complexos

- Você pode definir tipos mais complexos usando arrays e objetos.

```
let tuple: [number, string] = [1, "John"];
```

Interfaces e Tipos Personalizados

Interfaces permitem definir a forma de um objeto.

```typescript
interface User {
  id: number;
  name: string;
}

let user: User = {
  id: 1,
  name: "John"
};
```

Classes e Herança

TypeScript suporta classes e herança, permitindo o uso de orientação a objetos.

```typescript
class Animal {
  name: string;
  constructor(name: string) {
    this.name = name;
  }

  move(distanceInMeters: number = 0) {
    console.log(`${this.name} moved ${distanceInMeters}m.`);
  }
}

class Dog extends Animal {
  bark() {
    console.log('Woof! Woof!');
  }
}

const dog = new Dog("Rex");
dog.bark();
dog.move(10);
```

Módulos

Os módulos ajudam a organizar o código em diferentes arquivos e namespaces.

```
// module1.ts
export class Car {
  make: string;
  constructor(make: string) {
    this.make = make;
  }
}

// main.ts
import { Car } from './module1';
let myCar = new Car("Toyota");
```

Compilação e Ferramentas

Use o comando tsc para compilar arquivos TypeScript para JavaScript.

```
tsc main.ts
```

Você pode configurar um processo de build automatizado com ferramentas como Webpack e Gulp.

Exemplos Práticos

Exemplo 1: Funções Tipadas

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

let sum = add(5, 3);
console.log(sum);
```

Exemplo 2: Manipulação de DOM

```
let button = document.createElement('button');
button.textContent = "Click me";
button.onclick = () => {
  alert("Button clicked!");
};
document.body.appendChild(button);
```

Conclusão

TypeScript é uma poderosa ferramenta que pode transformar a maneira como você escreve e mantém código JavaScript. Com sua tipagem estática e recursos avançados de desenvolvimento, ele permite criar aplicativos mais robustos e escaláveis.

MARCOS CAMARGO