Conceitos do Typescript

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Introdução a Typescript

TypeScript é uma linguagem de programação desenvolvida pela Microsoft que adiciona tipagem estática opcional ao JavaScript. Enquanto o JavaScript é uma linguagem dinâmica, o que significa que o tipo das variáveis é determinado em tempo de execução, o TypeScript permite que você defina explicitamente os tipos em tempo de compilação. Isso ajuda a detectar erros antes de o código ser executado, tornando o desenvolvimento mais seguro e previsível.

Introdução a Typescript

TypeScript é um superconjunto de JavaScript, ou seja, todo código JavaScript é código TypeScript válido, mas o TypeScript adiciona recursos a mais, como:

- Tipagem estática: Definição de tipos como string, number, boolean, entre outros.
- Interfaces: Definir estruturas claras para objetos.
- Enumerações (Enums): Melhor organização de valores relacionados.
- Classes e modificadores de acesso: Suporte a programação orientada a objetos, com modificadores como public, private, e protected.

Como Funciona?

- Compilação: O TypeScript precisa ser compilado para JavaScript, já que os navegadores e ambientes como o Node.js não entendem TypeScript diretamente.

- Ferramentas de Desenvolvimento: TypeScript melhora a experiência com ferramentas como autocomplete, verificação de erros em tempo real, e intellisense, oferecendo mais previsibilidade e segurança ao escrever código.

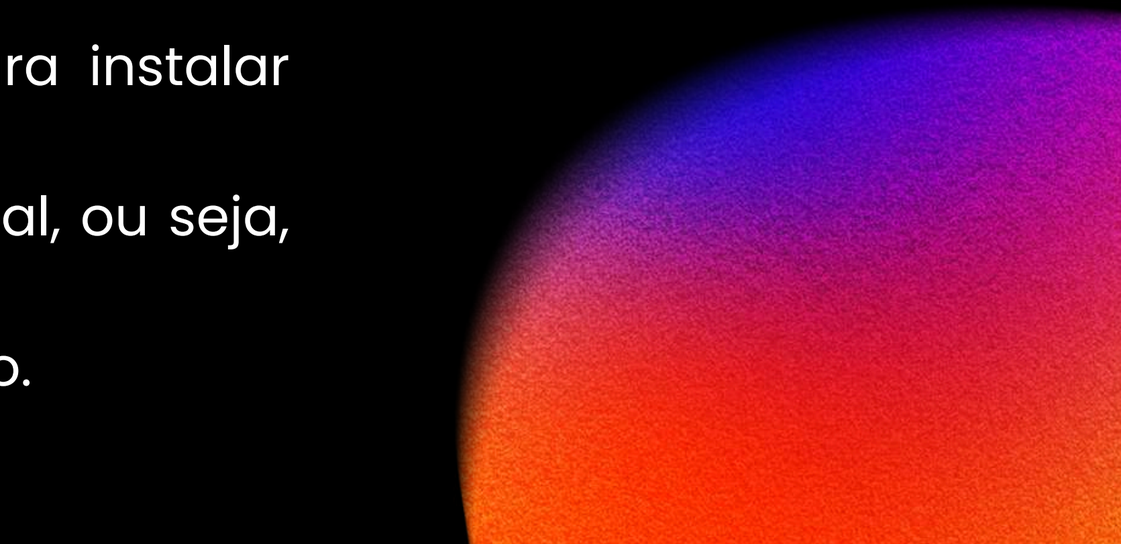
Instalação e Configuração Básica do TypeScript

1. Instalando o TypeScript:
Primeiro, você precisa instalar o TypeScript na sua máquina globalmente, para que possa utilizá-lo em qualquer projeto.

```
npm install -g typescript
```

O que esse comando faz?

- npm: O Node Package Manager, usado para instalar pacotes.
- install -g: Instala o TypeScript de forma global, ou seja, estará disponível em todos os projetos.
- typescript: O pacote que está sendo instalado.



Compilando TypeScript

Depois de instalar o TypeScript, você pode compilar arquivos .ts. Vamos criar um exemplo simples para testar.

Passo 1: Crie um arquivo chamado exemplo.ts

```ts
// exemplo.ts
let saudacao: string = "Olá, TypeScript!"; // Declara uma variável do tipo string
console.log(saudacao); // Imprime a saudação no console
```

Compilando TypeScript

Passo 2: Compile o arquivo TypeScript para JavaScript.

```
tsc exemplo.ts
```

Isso vai gerar um arquivo exemplo.js contendo o seguinte código JavaScript:

```
// exemplo.js
var saudacao = "Olá, TypeScript!"; // Declara uma variável do tipo string
console.log(saudacao); // Imprime a saudação no console
```

Compilando TypeScript

O que aconteceu?

- O TypeScript foi compilado para JavaScript, removendo os tipos no processo.
- Agora você pode executar o arquivo exemplo.js normalmente com o Node.js:

```
node exemplo.js
```

3. Configurando o TypeScript com tsconfig.json

Se você não quer ficar compilando arquivos um por um toda vez, pode usar um arquivo de configuração chamado tsconfig.json. Esse arquivo define as regras que o compilador do TypeScript deve seguir no projeto.

Passo 1: Crie o tsconfig.json.

```
tsc --init
```

Isso vai criar um arquivo de configuração básico para o TypeScript, chamado tsconfig.json.

3. Configurando o TypeScript com tsconfig.json

Passo 2: Entenda os principais parâmetros do tsconfig.json.

Aqui está um exemplo simples e comentado do tsconfig.json:

```
{
  "compilerOptions": {
    "target": "es5",  // Define o alvo de compilação. ES5 garante compatibilidade com navegadores mais antigos.
    "module": "commonjs",  // Define o sistema de módulos. 'commonjs' é usado em projetos Node.js.
    "strict": true,  // Habilita verificações rigorosas de tipos, recomendável para evitar erros.
    "outDir": "./dist",  // Define o diretório de saída dos arquivos compilados.
    "rootDir": "./src",  // Define o diretório raiz dos arquivos TypeScript.
    "esModuleInterop": true  // Habilita a compatibilidade com módulos ES6 e CommonJS.
  },
  "include": ["src/**/*"],  // Inclui todos os arquivos TypeScript dentro da pasta 'src'.
  "exclude": ["node_modules", "**/*.test.ts"]  // Exclui a pasta 'node_modules' e arquivos de teste.
```

Explicação Detalhada dos Parâmetros:

1."target": "es5"

- Define para qual versão do JavaScript o TypeScript será compilado.
- No exemplo, usamos ES5 (ECMAScript 5), amplamente suportado por navegadores antigos.

"2.module": "commonjs"

- Define o sistema de módulos. CommonJS é o padrão para o Node.js.
- Se você estiver usando TypeScript no navegador com ES6, pode mudar para "module": "es6".

Explicação Detalhada dos Parâmetros:

"3.strict": true

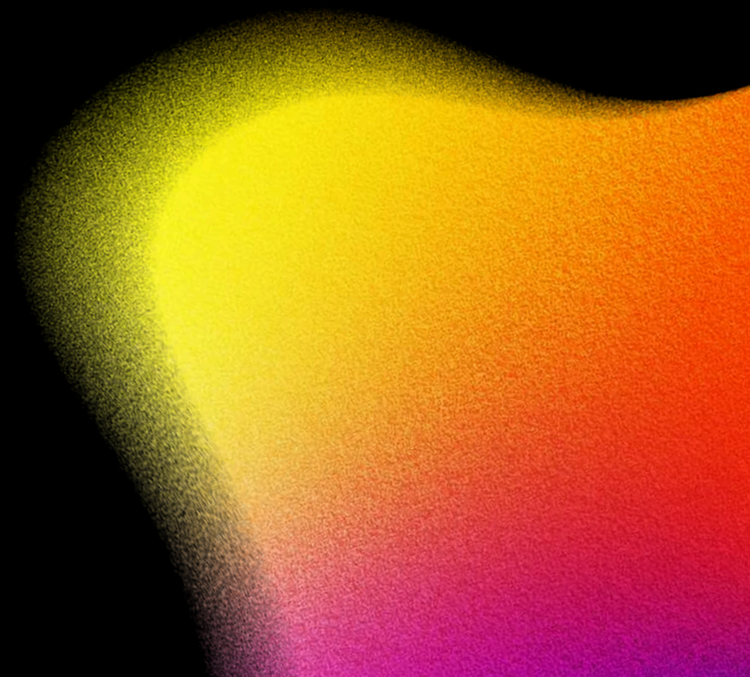
- Habilita o modo estrito, que força verificações rigorosas de tipos e ajuda a evitar erros de forma mais eficaz.

"4.outDir": "./dist"

- Especifica a pasta onde os arquivos JavaScript compilados serão salvos.

"5.rootDir": "./src"

- Define a pasta onde estão os arquivos .ts. Todos os arquivos TypeScript serão lidos a partir dessa pasta.



Explicação Detalhada dos Parâmetros:

6. “esModuleInterop": true
   - Isso permite que você importe módulos compatíveis com CommonJS e ES6 sem problemas de sintaxe.

"7.”include": ["src/**/*"]
   - Inclui todos os arquivos TypeScript dentro da pasta src e subpastas.

8."exclude"["node_modules", "**/*.test.ts"]
   - Exclui a pasta node_modules e arquivos de teste da compilação.

Exemplo de Projeto com tsconfig.json

Estrutura de Pastas:

```
meu-projeto/
  ├── dist/          # Arquivos compilados (gerados automaticamente)
  ├── src/           # Código TypeScript
  │   └── app.ts     # Arquivo TypeScript
  └── tsconfig.json # Arquivo de configuração do TypeScript
```

Arquivo TypeScript (exemplo app.ts):

```ts
// src/app.ts
function saudacao(nome: string): string {
    return `Olá, ${nome}!`; // Retorna uma saudação personalizada
  }

  console.log(saudacao("TypeScript")); // Chama a função e imprime no console
```

Compilando o Projeto:

Agora que você tem o tsconfig.json configurado, pode compilar o projeto de uma vez usando o comando tsc.

Todos os arquivos TypeScript serão compilados automaticamente e os arquivos JavaScript aparecerão na pasta dist.

Resumo

- O TypeScript adiciona tipagem estática ao JavaScript, permitindo detectar erros antes da execução.
- Instalamos o TypeScript globalmente para compilar arquivos .ts.
- Criamos o tsconfig.json para gerenciar a compilação do projeto.
- Exploramos as configurações básicas do TypeScript e organizamos o código em uma estrutura clara de pastas.

Gostou do conteúdo? Tem mais nas redes sociais!
basta clicar nos ícones abaixo:

@nath_dev_

Nathalia de Macedo Martins

Nathalia-Macedo