Passo a Passo: Criando um Projeto TypeScript no VS Code

Pré-requisitos:

1. **Node.js e npm (ou yarn):** Certifique-se de que você tem o Node.js instalado. O npm (Node Package Manager) vem junto com ele. Você pode baixá-lo em nodejs.org.
    - Verifique a instalação:

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

2. **Visual Studio Code (VS Code):** Baixe e instale em code.visualstudio.com.

Vamos Começar:

1. Crie uma Pasta para o Projeto:

- Abra seu terminal ou prompt de comando.
- Crie uma nova pasta para o seu projeto e navegue até ela:

  ```
  mkdir meu-projeto-ts
  cd meu-projeto-ts
  ```

2. Abra a Pasta no VS Code:

- Ainda no terminal, dentro da pasta do projeto, digite:

  ```
  code .
  ```

  (Isso abrirá a pasta atual no VS Code)
- Ou abra o VS Code e vá em `File > Open Folder...` (ou `Arquivo > Abrir Pasta...`) e selecione a pasta que você criou.

3. Inicialize um Projeto Node.js (package.json):

- Abra o terminal integrado do VS Code (`Ctrl + ``` ou View > Terminal`).
- Execute o comando para criar um arquivo `package.json`. Este arquivo rastreia as dependências do seu projeto e outras informações.

  ```
  npm init -y
  ```

  (O `-y` aceita todas as opções padrão, o que é suficiente para começar.)

4. Instale o TypeScript como Dependência de Desenvolvimento:

- No terminal integrado do VS Code, instale o TypeScript:

  ```
  npm install --save-dev typescript
  ```

    - `--save-dev`: Salva o TypeScript como uma dependência de desenvolvimento, pois ele é necessário apenas durante o processo de desenvolvimento/compilação, não em produção (o código final será JavaScript).

5. Crie o Arquivo de Configuração do TypeScript (tsconfig.json):

- Este arquivo informa ao compilador TypeScript como compilar seu código.
- Você pode criar um arquivo `tsconfig.json` básico automaticamente usando o `npx` (que executa pacotes npm sem instalá-los globalmente):

  ```
  npx tsc --init
  ```

- Isso criará um arquivo `tsconfig.json` com muitas opções comentadas. Para um projeto simples, podemos começar com algumas configurações chave. Abra o `tsconfig.json` e ajuste-o (ou substitua pelo exemplo abaixo, removendo os comentários para simplificar):

```
{
  "compilerOptions": {
    /* Configurações Básicas */
    "target": "ES2020",              // Especifica a versão do ECMAScript de destino (ex: "ES5", "ES2016", "ESNext")
    "module": "CommonJS",            // Especifica o sistema de módulos (ex: "None", "CommonJS", "ES6", "ES2015", "ESNext")
    "outDir": "./dist",              // Redireciona a estrutura de saída para o diretório especificado.
    "rootDir": "./src",              // Especifica o diretório raiz dos arquivos de entrada.

    /* Checagens Estritas */
    "strict": true,                  // Habilita todas as opções de verificação de tipo estritas. (RECOMENDADO)
    // "noImplicitAny": true,        // Levanta erro em expressões e declarações com um tipo 'any' implícito. (Incluído com "s
    // "strictNullChecks": true,     // Quando ativado, 'null' e 'undefined' têm seus próprios tipos distintos. (Incluído com

    /* Interoperabilidade de Módulos */
    "esModuleInterop": true,         // Permite interoperabilidade com módulos CommonJS.
    "forceConsistentCasingInFileNames": true, // Garante que referências a arquivos no mesmo projeto tenham o mesmo casing.

    /* Avançado */
    "skipLibCheck": true             // Pula a verificação de tipo de todos os arquivos de declaração (.d.ts).
  },
  "include": [
    "src/**/*"                      // Quais arquivos/pastas incluir na compilação
  ],
  "exclude": [
    "node_modules",                 // Quais arquivos/pastas excluir
    "**/*.spec.ts"                  // Excluir arquivos de teste, por exemplo
  ]
}
```

- **Importante:**
  - `"outDir": "./dist"` : Os arquivos JavaScript compilados irão para uma pasta chamada `dist` .
  - `"rootDir": "./src"` : Seus arquivos TypeScript (`.ts`) devem ficar em uma pasta chamada `src` .
  - `"strict": true` : É altamente recomendado para aproveitar ao máximo os benefícios do TypeScript.

6. Crie sua Estrutura de Pastas e o Primeiro Arquivo TypeScript:

- Na raiz do seu projeto no VS Code, crie uma pasta chamada `src` .
- Dentro da pasta `src` , crie um arquivo chamado `index.ts` (ou qualquer outro nome que preferir).

```
meu-projeto-ts/
├── node_modules/
├── src/
│   └── index.ts
├── package.json
├── package-lock.json
└── tsconfig.json
```

- Adicione algum código TypeScript simples ao `src/index.ts` :

```
function saudar(nome: string): string {
    return `Olá, ${nome}! Bem-vindo(a) ao TypeScript.`;
}

let usuario: string = "Estudante";
console.log(saudar(usuario));

// Tente introduzir um erro de tipo para ver o VS Code e o compilador agindo:
// usuario = 123; // Isso mostrará um erro no VS Code
```

7. Compile o Código TypeScript:

- No terminal integrado do VS Code, execute o compilador TypeScript:

```
npx tsc
```

- Se tudo estiver configurado corretamente, você verá uma nova pasta `dist` criada na raiz do projeto, contendo o arquivo `dist/index.js` (o JavaScript compilado).

```
meu-projeto-ts/
├── dist/
│   └── index.js  <-- Arquivo JavaScript compilado
├── node_modules/
├── src/
│   └── index.ts
├── package.json
├── package-lock.json
└── tsconfig.json
```

8. Execute o Código JavaScript Gerado:

- No terminal, execute o arquivo JavaScript usando Node.js:

```
node dist/index.js
```

- Você deverá ver a saída: `Olá, Estudante! Bem-vindo(a) ao TypeScript.`

9. (Opcional) Adicione Scripts ao `package.json` para Facilitar:

- Abra o `package.json` e adicione alguns scripts à seção `"scripts"`:

```
{
  "name": "meu-projeto-ts",
  "version": "1.0.0",
  "description": "",
  "main": "dist/index.js", // Ponto de entrada principal após a compilação
  "scripts": {
    "build": "tsc", // Compila o projeto
    "start": "node dist/index.js", // Executa o código compilado
    "dev": "tsc -w & nodemon dist/index.js" // Compila em modo watch e reinicia o servidor com nodemon (requer 'nodemon' insta
    // Se não quiser usar nodemon, pode ser "tsc -w" para apenas recompilar
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "typescript": "^5.0.0" // A versão pode variar
  }
}
```

- Agora você pode usar:
  - `npm run build` para compilar.
  - `npm run start` para executar o código compilado.
  - Para o script `dev`, você precisaria instalar o `nodemon`: `npm install --save-dev nodemon`. O `tsc -w` compila automaticamente quando você salva um arquivo `.ts`.

Dicas Extras:

- **VS Code IntelliSense:** O VS Code tem um excelente suporte para TypeScript. Ele usará seu `tsconfig.json` para fornecer autocompletar, verificação de erros em tempo real e muito mais.
- **Modo Watch:** Para recompilar automaticamente quando você fizer alterações nos seus arquivos `.ts`, use:

```
npx tsc -w
```

- **ts-node:** Para executar arquivos TypeScript diretamente sem compilar manualmente para um arquivo `.js` separado (ótimo para desenvolvimento):
  1. Instale: `npm install --save-dev ts-node @types/node`
     - `@types/node` fornece as definições de tipo para o ambiente Node.js.
  2. Execute: `npx ts-node src/index.ts`
  3. Você pode adicionar um script no `package.json`: `"serve": "ts-node src/index.ts"` e rodar com `npm run serve`.

Pronto! Você criou e configurou seu primeiro projeto TypeScript no VS Code.
`Prof. José Carlos Flores`