TypeScript

INTRODUÇÃO AO TYPESCRIPT

Prof. Me. Hélio Esperidião

O que é...

- TypeScript é uma linguagem de programação de código aberto desenvolvida pela Microsoft.
- É um superconjunto sintático estrito de JavaScript e adiciona tipagem estática opcional à linguagem.
- Foi criado com o objetivo de incluir recursos que não estão presentes no JS. Por meio dele é possível definir a tipagem estática, parâmetros e retorno de funçõ

introdução

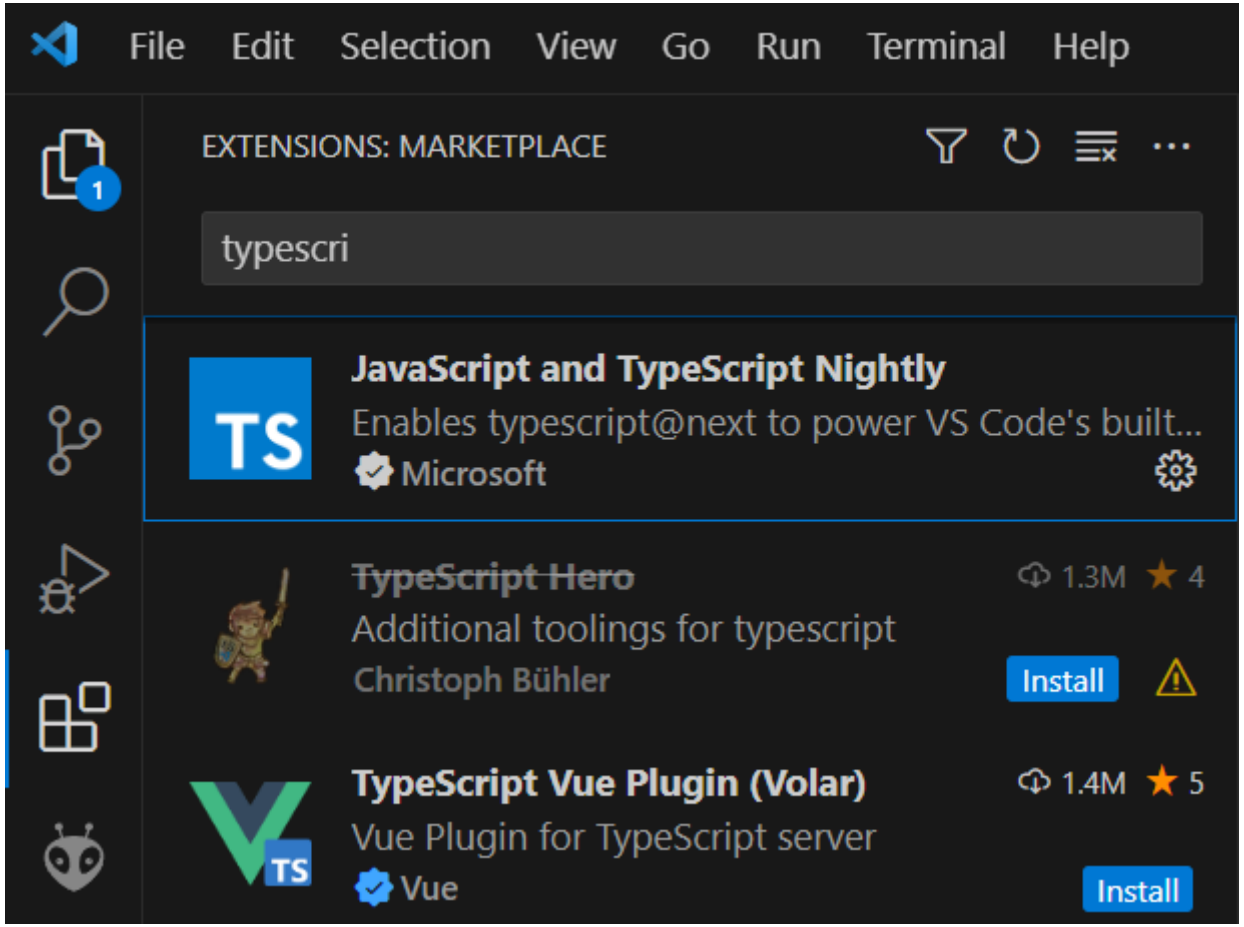
- Quando se escreve em TypeScript, é possível especificar os tipos de variáveis, parâmetros de funções e objetos, além de criar interfaces para descrever a estrutura de um objeto.
- Esse recurso possibilita que o compilador do TypeScript analise o código durante a compilação, detectando erros antes mesmo da execução. TypeScript pode ser empregado em todos os contextos em que o JavaScript
- É frequentemente adotado em projetos extensos e aplicações, devido à sua capacidade de dimensionamento e facilitação da manutenção do código.

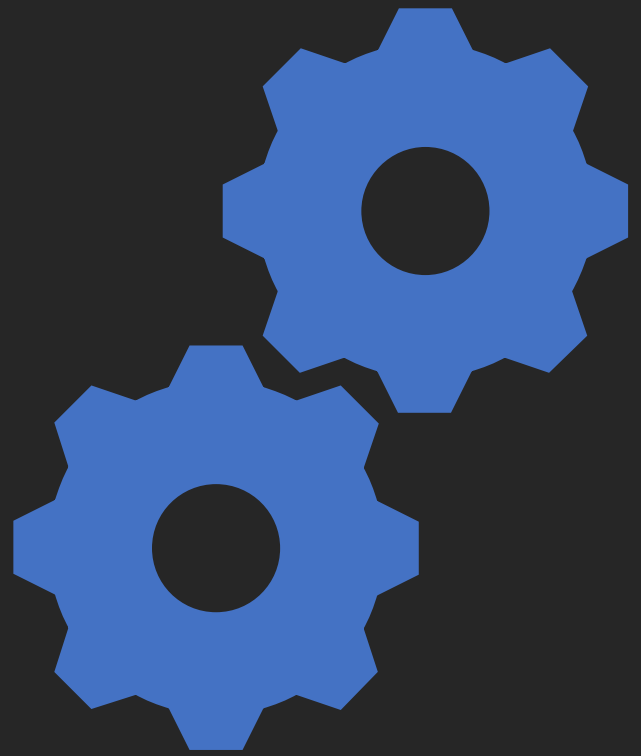
Qual é a diferença entre TypeScript e JavaScript?

- TypeScript é uma “extensão” mais completa do JavaScript, há diferenças entre eles. A primeira delas é que no TypeScript existe uma tipagem estática e, no JavaScript, uma linguagem dinâmica.

- o TypeScript possui um funcionamento voltado a orientação de objetos, já no JS há uma programação estruturada na linguagem. Além disso, o Typescript é uma linguagem compilada e que transpila para Javascript.

Instale o plugin do typescript





Configuração de console

- Abaixo segue a configuração básica para execução e compilação de aplicações .ts

Comando	Funcionalidade
npm init –yes	Inicializar um projeto javscript node
npm install express dotenv --save	Instala e salva na aplicação o express
npm i -D typescript @types/express @types/node --save	Instala e salva na aplicação os tipos do express
npx tsc --init	Inicializa o typescript
mkdir public	Cria a pasta public onde serão posicionados os arquivos “compilados” para serem publicados em um servidor web.
mkdir src	Cria um arquivo src para armazenar todos os arquivos fontes da aplicação.
cd src	Acessa as pasta src
npx tsc	“compila” a aplicação.
node public/main	Roda o arquivo main.js dentro da passa public

tsconfig.json

- Arquivo de configuração base da “compilação” typescript
- O arquivo é gerado automaticamente e deve ser configurado conforme o próximo slide.

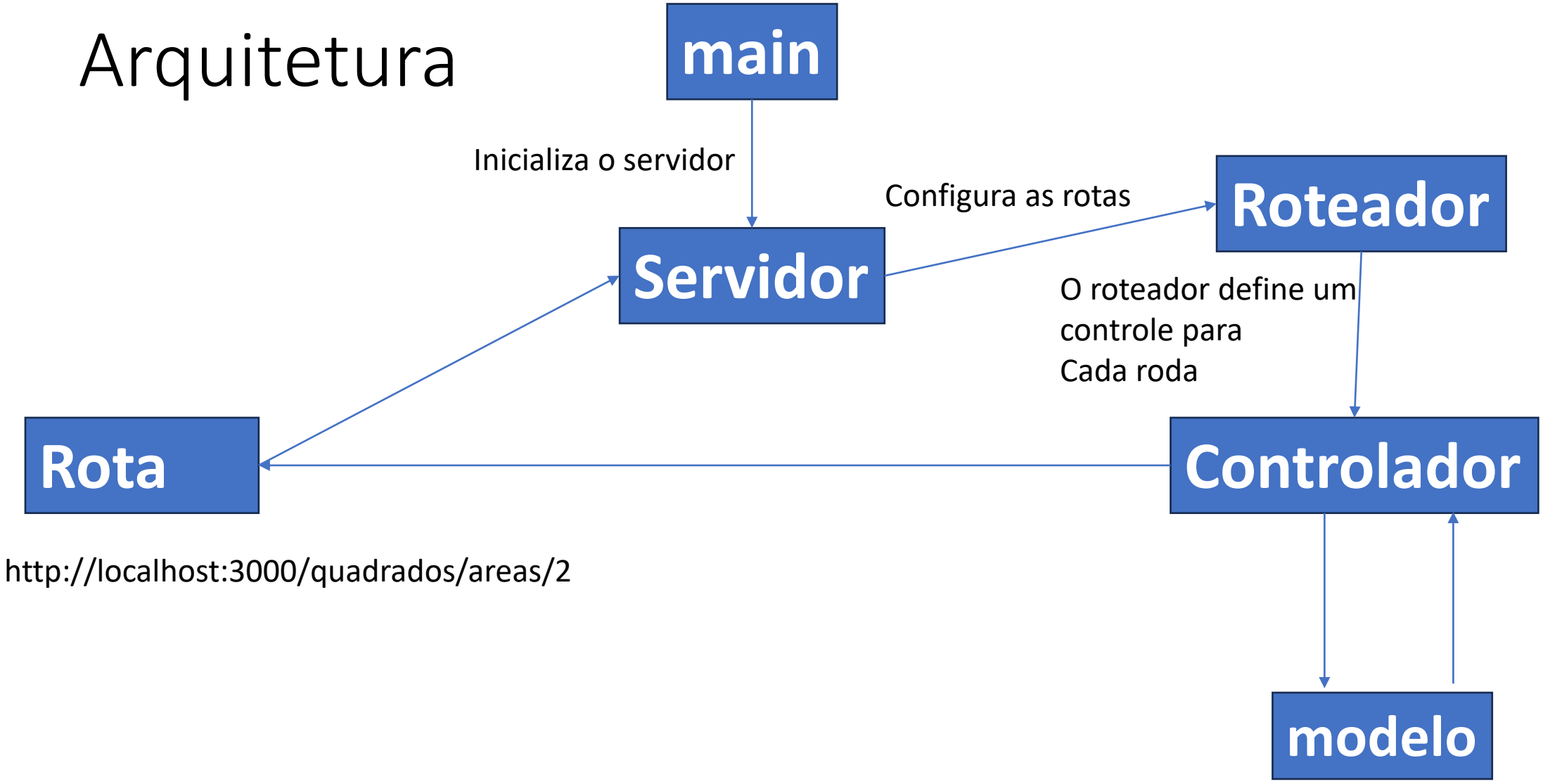
tsconfig.json

```
{
  "compilerOptions": {
    /* Language and Environment */
    "target": "es2016",                                  /* Set the JavaScript language version for emitted JavaScript and include compatible library declarations. */
    "experimentalDecorators": true,                      /* Enable experimental support for legacy experimental decorators. */
    "emitDecoratorMetadata": true,                       /* Emit design-type metadata for decorated declarations in source files. */

    /* Modules */
    "module": "commonjs",                                /* Specify what module code is generated. */
    "resolveJsonModule": true,                           /* Enable importing .json files. */

    /* Emit */
    "outDir": "./public",                                /* Specify an output folder for all emitted files. */

    /* Interop Constraints */
    "esModuleInterop": true,                             /* Emit additional JavaScript to ease support for importing CommonJS modules. This enables
'allowSyntheticDefaultImports' for type compatibility. */
    "forceConsistentCasingInFileNames": true,            /* Ensure that casing is correct in imports. */

    /* Type Checking */
    "strict": true,                                      /* Enable all strict type-checking options. */

    /* Completeness */
    "skipLibCheck": true   ,                             /* Skip type checking all .d.ts files. */


  },
  "include": [ "src/**/*" ],
  "exclude": [ "src/**/*.txt" ],
  "hooks": [ "copy-files" ] // hooks is a new property you can add to tsconfig to add custom hooks

}
```

Arquitetura

- Aplicação
  - Public
  - Src
    - Controlador
    - Modelo
    - Public
    - Roteador
  - Main.ts
  - Servidor.ts

Arquitetura



http://localhost:3000/quadrados/areas/2

TypeScript e mysql

- É necessário configurar algumas etapas a mais:

Comando	Descrição
npm init –yes	Inicializa um projeto npm
npm install express dotenv --save	Instala o express dentro do projeto
npm i -D typescript @types/express @types/node --save	Instala os tipos do express
npm install md5 --save	instala o md5 dentro do projeto
npx tsc --init	Inicializa o “compilador” typescript
mkdir public	Cria a pasta public para armazenar os arquivo públicos da aplicação
mkdir src	
cd src	Dentro da pasta src posicione seus arquivos fonte .ts E todos os arquivos que compõem a aplicação
npm install mysql2 --save	Instala e salva o mysql2

Configuração de console

npm install --save @types/jsonwebtoken	Instala os tipos do jsonwebtoken
npm install tsc-hooks --dev --save	#utilizado para copiar a pasta public do src para build
npx tsc	
node public/main	Rodar o arquivo main.js dentro da pasta public compilada

tsconfig.json

- Arquivo de configuração base da “compilação” typescript
- O arquivo é gerado automaticamente e deve ser configurado conforme o próximo slide.

tsconfig.json

```
{
  "compilerOptions": {
    /* Language and Environment */
    "target": "es2016",                                  /* Set the JavaScript language version for emitted JavaScript and include compatible library declarations. */
    "experimentalDecorators": true,                      /* Enable experimental support for legacy experimental decorators. */
    "emitDecoratorMetadata": true,                       /* Emit design-type metadata for decorated declarations in source files. */

    /* Modules */
    "module": "commonjs",                                /* Specify what module code is generated. */
    "resolveJsonModule": true,                           /* Enable importing .json files. */

    /* Emit */
    "outDir": "./public",                                /* Specify an output folder for all emitted files. */

    /* Interop Constraints */
    "esModuleInterop": true,                             /* Emit additional JavaScript to ease support for importing CommonJS modules. This enables
'allowSyntheticDefaultImports' for type compatibility. */
    "forceConsistentCasingInFileNames": true,            /* Ensure that casing is correct in imports. */

    /* Type Checking */
    "strict": true,                                      /* Enable all strict type-checking options. */

    /* Completeness */
    "skipLibCheck": true   ,                             /* Skip type checking all .d.ts files. */


  },
  "include": [ "src/**/*" ],
  "exclude": [ "src/**/*.txt" ],
  "hooks": [ "copy-files" ] // hooks is a new property you can add to tsconfig to add custom hooks

}
```

Arquitetura

- Src
  - Controlador
  - Modelo
  - Public
  - Roteador
- Main.ts
- Servidor.ts

Arquitetura

- Após a “compilação” será gerada uma pasta chamada public.
- Todos os arquivos .ts são convertidos para .js e todos os outros tipos de arquivos e pastas são copiados para a pasta public.
- A pasta public será para publicar a sua aplicação.