O que é o Node.JS?

O JavaScript figura hoje como uma das linguagens mais utilizadas, e em grande parte isso se deve ao fato de ser uma linguagem base para dezenas de frameworks com alta popularidade e adesão na comunidade de desenvolvimento.

Por se tratar de uma linguagem popularmente conhecida para a construção de aplicações web mais interativas, o JavaScript possui grande foco no Front-end (client side), ou seja, é comumente utilizada para rodar no “lado cliente” da aplicação. Com a evolução das tecnologias web, tornou-se possível fazer o JavaScript rodar também no Back-end, e é nesse momento de consolidação de tecnologias e soluções que surge o Node.js.

Mas, afinal, o que é o Node.js? O Node.js é um ambiente de execução do código JavaScript do lado servidor (server side), que na prática se reflete na possibilidade de criar aplicações standalone (autossuficientes) em uma máquina servidora, sem a necessidade do navegador.

O que é o TypeScript?

Mais conhecido como um superset do Javascript, ou seja, um conjunto de ferramentas, o TypeScript 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 funções.

Além de ser uma ferramenta orientada a objetos, fortemente tipada e que pode ser escrita em qualquer ambiente de desenvolvimento, o TypeScript quando instalado via gerenciador de pacotes JS, permite checar erros e utilizar outros compiladores que suportam este mecanismo.

O nome TypeScript surgiu da combinação de palavras "JavaScript" + “Type”(“Tipo” em português), representa a sua finalidade mais importante: a tipagem estática, na qual possibilita programar tanto do lado do cliente (client-side), como no lado do servidor (server-side). Logo, o TypeScript eleva o nível de produtividade e ainda garante o desenvolvimento de aplicações complexas, eficazes e seguras.

Instalando o TypeScript

Criar um diretório para ficar todos os arquivos do projeto em POO. Escolha o local e crie um diretório.

Por exemplo: Criar uma pasta com o seu nome e dentro dela outra pasta chamada “Projeto Padaria”.

Abrir o Windows Power Sheel e posicionar a linha de comando na pasta criada utilizando os comandos “cd” caso seja Windows e “ls” caso seja no Linux.

Com o NodeJS instalado, vamos iniciar o projeto com o comando abaixo:

```
npm init
```

Após a execução do comando, será criado um arquivo “package.json” na raiz da pasta que contém as configurações do projeto. Inicialmente não existe nenhuma biblioteca adicionada ao projeto, pois ele foi iniciado do zero.

Para adicionar a biblioteca do “TypeScript” ao projeto, digite a seguinte linha de comando:

```
npm install typescript
```

Após isso, os arquivos do typescript serão baixados para a pasta “node_modules” que está na raiz do projeto e será adicionada uma entrada no arquivo “package.json” para essa biblioteca. Todas as bibliotecas do projeto estarão referenciadas no atributo “dependencies” desse arquivo.

Para criar o arquivo de configuração do TypeScript, digitar o seguinte comando:

```
npx tsc –init
```

Após isso, será criado um arquivo chamdo “tsconfig.json” na raiz do projeto que contém a configuração padrão para o TypeScript.

Para organizar os códigos fontes que serão gerados em JavaScript a partir do TypeScript em um diretório separado, criaremos um diretório chamado “dist” na raiz da pasta do projeto e adicionaremos o atributo abaixo na configuração do TypeScript (no arquivo “tsconfig.json”):

```
"outDir": "./dist"
```

Criando Primeiro Arquivo TypeScript

Criar o arquivo chamado “app.ts” na raiz do projeto com o seguinte código:

```
let idade: number = 15;
let nome: string = 'Paula';

console.log(`nome: ${nome}, idade: ${idade}`);
```

Executando arquivo TypeScript via linha de comando

Para gerar o JavaScript a partir do código TypeScript criado, executar o seguinte comando:

```
npx tsc
```

Com isso será criado um arquivo “app.js” dentro da pasta “dist” que poderá ser executado utilizando o NodeJS.

Para executar o arquivo, basta digitar o comando abaixo na linha de comando:

```
node ./dist/app.js
```

Com isso, o resultado irá aparecer na linha de comando.

Lendo Informações do Usuário no Prompt de Comando utilizando Node.JS

Como dito anteriormente, o JavaScript não foi uma linguagem criada para se trabalhar no prompt de comando, mas sim integrada aos navegadores de internet para dar mais dinamicidade às páginas de internet. Para ler informações do navegador em um navegador, usa-se a função “prompt” que irá lançar um campo para o usuário digitar uma informação. No entanto, isso não funciona se executado no Power Shell.

Para usar o prompt na linha de comando é precisa instalar um módulo para tal. Para adicionar o módulo de prompt ao projeto, digitar o seguinte comando na pasta raiz do projeto:

```
npm install --save prompt-sync
```

Os arquivos do módulo serão baixado na pasta “node-modules” e adicionado ao “dependencies” do arquivo “package.json”. Alterando o programa anterior para ler o nome e a idade do usuário, ficará assim:

```
var prompt = require('prompt-sync')();

let idade: number = prompt('Qual a sua idade?');
let nome: string = prompt('Qual a sua idade?');

console.log(`nome: ${nome}, idade: ${idade}`);
```

Criando Classes utilizando o TypeScript

Criar uma pasta chamada “source” que terá os códigos-fontes em TypeScript. Para as classes de domínio, criaremos o diretório “model” dentro da pasta “source” na qual criaremos todas as classes a partir de agora.

Será criado um arquivo para cada classe do problema. Antes de definir as classes, apresentemos os tipos nativos do TypeScritpt:

string	Utilizado para cadeia de caracteres e caracteres.
number	Utilizado tanto para números inteiros, quanto para números reais.
boolean	Verdadeiro ou falso
undefined	Tipo de dados indefinido. Aceitará qualquer tipo de dados
function	Recebe uma função que pode ser executada
array	Contém um conjunto de dados

Nossos tipos de dados poderão ser criados. Cada Classe ou Enum (enumeração) criada será um novo tipo de dados que poderá ser associado a uma variável de um código fonte ou a um atributo de outro objeto.

Para criar uma nova classe, crie um arquivo que irá conter a classe. Por exemplo, para criar a classe “Pessoa”, criar o arquivo “Pessoa.ts”. Para a classe pessoa, teremos o seguinte código:

```
class Pessoa {
        nome: string;
        idade: number;
        estaVivo; boolean;
}
```

Uma classe sempre será precedida pela palavra “class” (iniciando em minúsculo) e sempre terá seu nome iniciado em maiúsculo. Dentro da classe, que terá seu escopo limitado pelas chaves, estarão os atributos e os métodos. Na classe Pessoa, como o exemplo acima, foi colocado somente os atributos da classe, que são: nome, idade e estaVivo.

O nome do atributo sempre virá primeiro, iniciando com letra minúscula, acompanhado de dois pontos e o tipo daquele atributo. Ao final da declaração do nome e do tipo do atributo, existe um ponto e vírgula que irá separar cada atributo, apesar de estar declarado um em cada linha.

Bibliografia

Bessa, A. (s.d.). *Node.JS: o que é, como funciona esse ambiente de execução JavaScript e um Guia para iniciar*. Fonte: Alura: https://www.alura.com.br/artigos/node-js

Gado, W. (s.d.). *Classes no TypeScript*. Fonte: Treinaweb: https://www.treinaweb.com.br/blog/classes-no-typescript