Aula 3

Desenvolvimento *Web* - *Front End*

Prof. Mauricio Antonio Ferste

1

Conversa Inicial

2

TypeScript

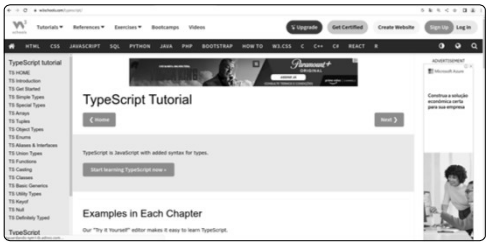
- **Quando utilizados em conjunto, o TypeScript, módulos e componentes fornecem uma base sólida para o desenvolvimento de aplicativos escaláveis e robustos no Angular. Eles ajudam a melhorar a legibilidade, a modularidade e a reutilização do código, promovendo uma estrutura organizada e facilitando a colaboração entre os desenvolvedores. Além disso, o TypeScript fornece recursos avançados, como a tipagem estática, que contribuem para a detecção precoce de erros e uma melhor experiência de desenvolvimento**

3

Typescript.com

4

W3C TypeScript



5

JavaScript *versus* TypeScript
- adições do TypeScript

6

Variáveis

```
let nome: string = "João"; // Variável 'nome' do tipo string
let idade: number = 30; // Variável 'idade' do tipo number
let ativo: boolean = true; // Variável 'ativo' do tipo boolean
```

7

Funções

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

let resultado: number = somar(8, 3); // A variável 'resultado' será do tipo
```

8

Funções

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

let resultado: number = somar(5, 3); // A variável 'resultado' será do tipo
```

9

Tipos de objetos

```
interface Pessoa {
  nome: string;
  idade: number;
  ativo: boolean;
}

let pessoa: Pessoa = {
  nome: "Maria",
  idade: 25,
  ativo: true
};
```

10

Classes

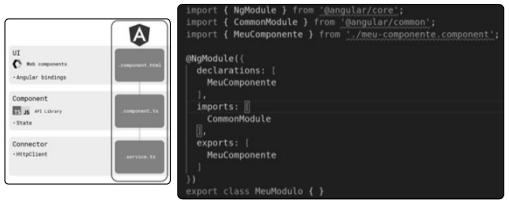
```
class Animal {
    name: string;

    constructor(name: string) {
      this.name = name;
    }

    ola() {
      console.log(`Olá, sou um ${this.name}.`);
    }
  }

  var animal = new Animal("Gato");
  animal.ola();
```

11

TypeScript no Angular

12

Angular @NgModule



13

Algumas configurações

- **1. tsconfig.json (já visto): este é o arquivo de configuração principal do TypeScript. Ele define as opções de compilação e as configurações do projeto. Nele, você pode especificar o diretório de saída, a versão do ECMAScript a ser usada, habilitar ou desabilitar recursos específicos do TypeScript e muito mais**

14

- **2. tslint.json: este arquivo define as regras e configurações para o linter TypeScript. O *linter* ajuda a manter um código consistente e livre de erros, aplicando regras de estilo e boas práticas de programação. Você pode definir suas próprias regras ou usar configurações predefinidas**
- **3. angular.json: este arquivo de configuração é específico do Angular e define as configurações do projeto, como as dependências, os arquivos a serem compilados, a estrutura do diretório, as tarefas de construção, a configuração do servidor de desenvolvimento e muito mais. É um arquivo essencial para a configuração geral do projeto Angular**

15

- **4. karma.conf.js: se você estiver usando o Karma como *test runner* para seus testes unitários no Angular, este arquivo contém a configuração para a execução dos testes. Você pode definir os arquivos a serem testados, os *frameworks* a serem usados, os *reporters* de resultados e outras opções relacionadas aos testes**

16

- **5. protractor.conf.js: se você estiver usando o Protractor para executar testes de ponta a ponta (e2e) no Angular, este arquivo contém a configuração para os testes e2e. Você pode definir os caminhos dos arquivos de teste, as configurações do navegador, as URLs a serem testadas e outras opções relevantes para os testes e2e**

17

Módulos Angular – componentes

18

Criando um módulo

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { MeuComponente } from './meu-componente.component';

@NgModule({
  declarations: [
    MeuComponente
  ],
  imports: [
    CommonModule
  ],
  exports: [
    MeuComponente
  ]
})
export class MeuModulo { }
```

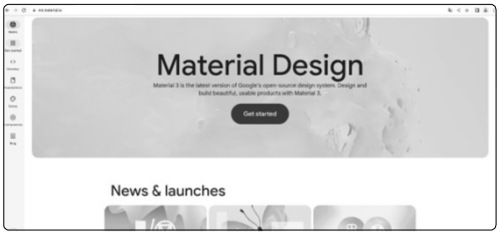
19

Criando um módulo

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { MeuModulo } from './meu-modulo.module';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    MeuModulo
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

20

Algumas explicações sobre Angular material



21

***Lifecycly hooks* – ganchos ou gatilhos do ciclo de vida dos componentes**

22

Eventos

- **ngOnChanges: este método é chamado quando uma ou mais propriedades vinculadas a um componente mudam. Ele recebe um objeto que contém as mudanças detectadas**
- **ngOnInit: este método é chamado quando um componente é inicializado. Ele é executado depois que todas as propriedades vinculadas são definidas e depois que o construtor do componente é executado**

23

- **ngDoCheck: este método é chamado durante cada detecção de mudança do Angular. Ele permite que você execute ações personalizadas para verificar se o estado do componente mudou**
- **ngAfterContentInit: este método é chamado depois que o Angular inicializa o conteúdo de um componente que usa a diretiva *ng-content***

24

- **ngAfterContentChecked: este método é chamado depois que o Angular verifica o conteúdo de um componente que usa a diretiva *ng-content***
- **ngAfterViewInit: este método é chamado depois que o Angular inicializa a exibição de um componente e suas visualizações filhas**
- **ngAfterViewChecked: este método é chamado depois que o Angular verifica a exibição de um componente e suas visualizações filhas**
- **ngOnDestroy: este método é chamado quando um componente é destruído. É usado para limpar recursos, como eventos ou assinaturas de observáveis**

25

Passo a passo

- **Acompanhe o passo a passo de criação...**

26

Comunicação entre componentes

27

HTML Injection

```
@Component({
    selector: 'app-meu-componente',
    template:
      `<div>
        <h2>Título do Componente</h2>
        <ng-content></ng-content>
      </div>
    `
  })
  export class MeuComponenteComponent { }
```

```
<app-meu-componente>
  <p>Conteúdo inserido no componente</p>
</app-meu-componente>

```

28

Binding no Angular

```
<p>Meu nome é {{ nome }}</p>

```

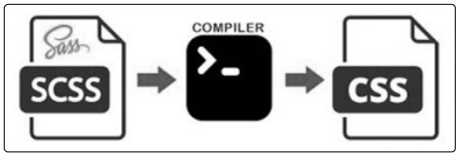
29

Binding no Angular

```
<p>Meu nome é {{ nome }}</p>

```

30

Estilos



31