Angular

Programação para Internet I
IFB - Tecnologia em Sistemas para Internet

Marx Gomes van der Linden

Angular

- Framework em TypeScript para o desenvolvimento de front-end em aplicações web
  - Desenvolvido pelo Google
  - Single-page applications
- Solução completa para desenvolvimento front-end

Angular vs. AngularJS

Angular vs. AngularJS

- AngularJS (2010)
  - Primeira versão do framework
  - Usa JavaScript
  - Descontinuado
- Angular (2016 em diante)
  - Versão mais moderna
  - Usa TypeScript
  - Incompatível com AngularJS

Instalando e inicializando

- Para instalar o Angular globalmente no sistema:

```
npm install -g @angular/cli
```

- Para criar uma nova aplicação:

```
ng new nome-do-app
```

ou

```
npx ng new nome-do-app
```

- Alternativa online: https://stackblitz.com

Opções que usaremos

```
? Do you want to enforce stricter type checking and stricter
 bundle budgets in the workspace?
  This setting helps improve maintainability and catch bugs
ahead of time.
  For more information, see https://angular.io/strict Yes
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
```

Compilando e executando

- Para compilar e executar durante o desenvolvimento:

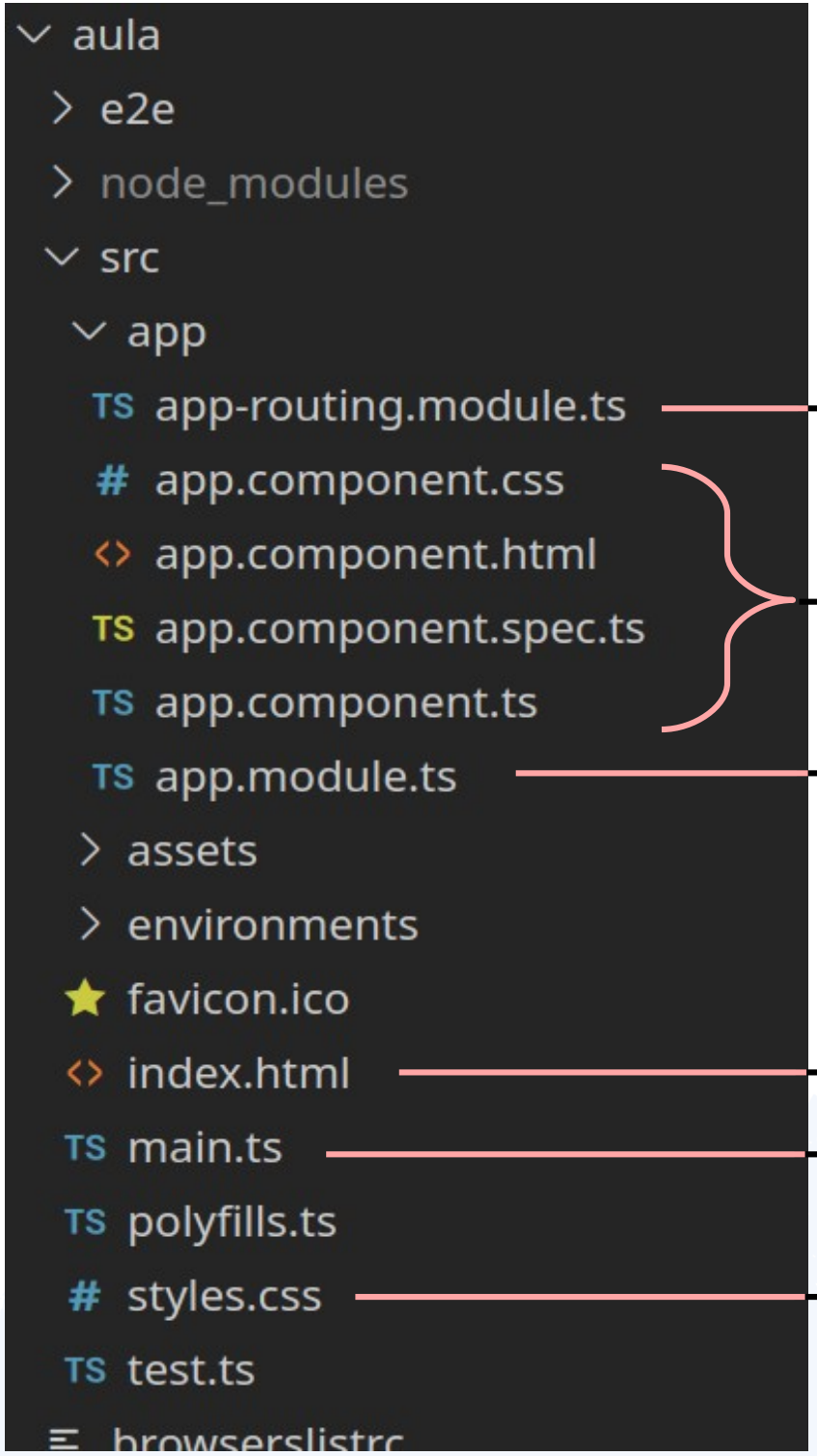
```
cd nome-do-app
```

```
ng serve --open
```

- Para gerar a versão de produção:

```
ng build --prod
```

Estrutura da aplicação



Componentes

- Uma aplicação angular é formada por **componentes**, que reúnem um elemento visual com sua lógica de programação
- Um componente inclui:
  - Uma tag HTML que o identifica
  - Um arquivo **.html** (HTML dinâmico do componente)
  - Um arquivo **.css** (Folha de estilo)
  - Um arquivo **.ts** (Código da lógica do componente)
  - Um arquivo **.spec.ts** (Código de testes)

Interpolação de texto

- Qualquer campo da classe TS pode ser usada no texto do arquivo HTML com a sintaxe: `{{ nomedaPropriedade }}`

Exemplo

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  titulo = 'texto de exemplo';
}
```

app.component.html

```
<p>Este é um {{titulo}}</p>
```

Pipes

- É possível usar **pipes** para formatar valores:

```
{{'eXempLo' | uppercase}}
```
→ EXEMPLO

```
{{'eXempLo' | lowercase}}
```
→ exemplo

```
{{'eXempLo' | titlecase}}
```
→ Exemplo

```
{{'1983-04-01' | date}}
```
→ 1 de abr. de 1983

```
{{120040.50 | number}}
```
→ 120.040,5

```
{{500.2 | currency:'BRL'}}
```
→ R$ 500,20

```
{{0.245 | percent}}
```
→ 25%

```
{{ {  chave: 'valor',
      elementos: ['a', "b", 'c']
   } | json
}}
```
→
```
{ "chave": "valor",
"elementos": [ "a",
"b", "c" ] }
```

Configuração do locale

- Para os pipes **date** e **number** funcionarem corretamente em português, acrescentar a **app.modules.ts**:

```
import { LOCALE_ID } from '@angular/core';
import { registerLocaleData } from '@angular/common';
import localePt from '@angular/common/locales/pt';
registerLocaleData(localePt);

//(…)
providers: [
    { provide: LOCALE_ID, useValue: "pt-BR" }
],
//(…)
```

Interpolação de propriedades

- Qualquer campo da classe TS pode ser usada como o valor da propriedade de uma tag HTML com a seguinte sintaxe:

```
<tag [propriedade]="valor">
```

app.component.css

```
.verde {
    color: green;
}

.vermelho {
    color: red
}
```

app.component.ts

```
// (…)
export class AppComponent {
  titulo = 'texto de exemplo';
  cor = 'vermelho';
}
```

app.component.html

```
<p [class]="cor">Este é um {{titulo}}</p>
```

Aplicação de eventos

- Para adicionar um evento a uma tag HTML, use a sintaxe:

```
<tag (evento)="código">
```

  - » O evento deve ser escrito sem o prefixo ”on”

app.component.ts

```
//(…)
export class AppComponent {
  titulo = 'texto de exemplo';
  cor = 'vermelho';

  escreve() {
    console.log("Evento funciona!");
  }
}
```

app.component.html

```
<p [class]="cor">Este é um {{titulo}}</p>
<button (click)="escreve()">Vai!</button>
```

Programação declarativa

- O Angular adota o paradigma **declarativo** na atribuição de valores para os componentes
- O componente sempre reflete os valores atuais de cada propriedade
  - » Não é necessário atualizar manualmente!

app.component.ts

```
export class AppComponent {
  titulo = 'texto de exemplo';
  cor = 'vermelho';

  mudaValores() {
    this.titulo = "novo valor do texto!";
    this.cor = 'verde';
  }
}
```

app.component.html

```
<p [class]="cor">Este é um {{titulo}}</p>
<button (click)="mudaValores()">Vai!</button>
```

Criando um novo componente

- Para criar um novo componente, use o comando:

```
ng generate component nome-do-componente
```

  - Todos os arquivos necessários serão criados automaticamente