



iscte

INSTITUTO
UNIVERSITÁRIO
DE LISBOA



emprego
digital

Módulo 9: Introdução ao React.js

Aula 01

Introdução ao Angular

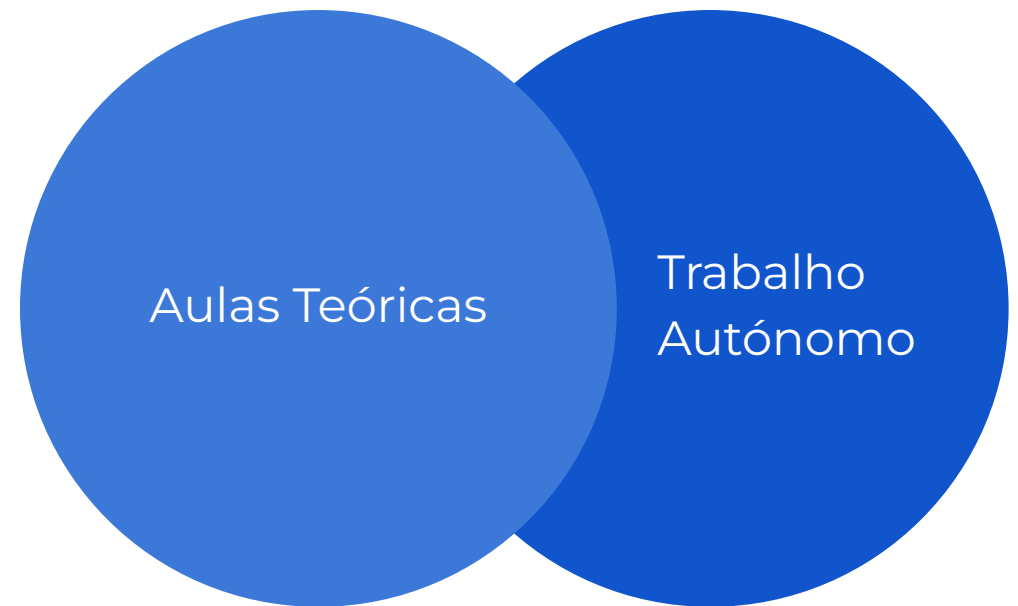


Programa do módulo

- Introdução ao Angular;
- Componentes;
- Criação de pequena aplicação com informação estática;
- Apresentação de listas;
- Ciclo de vida dos componentes;
- Alteração de conteúdos com Hooks;
- Apresentação condicional de conteúdo;
- Alimentação de conteúdos a partir de API;
- **Projeto**

Método de trabalho

- A cada 3,5h de aula teórica correspondem 3,5h de trabalho autónomo.
- Durante o trabalho autónomo, devem colocar dúvidas através do canal de discussão criado no [Discord](#).



Avaliação

Trabalho autónomo Submissão obrigatória no moodle	20%
Teste	20%
Projeto final	50%
Avaliação contínua (qualidade e organização do trabalho, autonomia, sentido de responsabilidade, assiduidade, etc)	10%

Frameworks, o que são?

- São um conjunto de ferramentas (bibliotecas e classes) que oferecem funcionalidades específicas.
- Podem ser consideradas como um padrão a seguir.
- São uma “camada” instalada por cima de uma certa linguagem de programação, e tem como principal objetivo resolver problemas recorrentes.

E porque é que existem?

- Porque muitas vezes existe uma necessidade de se desenvolverem sempre funcionalidades iguais e parecidas em muitas aplicações.
- “Incluem blocos de código prontos a usar”.

Frameworks: vantagens

- **Redução de tempo**

- Estas ferramentas já possuem muitas funcionalidades desenvolvidas e/ou simplificam o desenvolvimento de novas funcionalidades

- **Segurança**

- São códigos altamente analisados, o que garante um bom nível de segurança e atualizações constantes.

- **Legibilidade**

- A arquitetura destas frameworks é *clean* e simples de se entender e coerente em todos os projetos.

Frameworks: desvantagens

- **Configuração**

- As frameworks são complexas e exigem configurações, por vezes, demoradas.

- **Dependência**

- O sistema fica dependente da framework utilizada. Em caso de problemas na framework todo o sistema fica comprometido.

- **Códigos desnecessários**

- Frameworks muito completas podem ter muito código que não precisamos no nosso projeto, mas que em todo o caso estará lá, tornando a aplicação mais “pesada”.

Exemplos de Frameworks

- Javascript (Frontend)
 - Angular
 - Vue
 - Ember.js
 - React
- Javascript (Backend)
 - Express
 - Node.js
- CSS
 - Bootstrap
 - Materialize
- PHP
 - Laravel
 - Symfony
- Java
 - Spring
- Python
 - Django
- Ruby
 - RubyOnRails

O que é o Angular?

- Framework de design de aplicações e desenvolvimento de plataformas para criar SPAs de forma eficiente e sofisticada
- Apesar de ser uma framework muito orientada a frontend pode também ser utilizada para o desenvolvimento de aplicações Fullstack realizando por exemplo, pedidos HTTP a um servidor em backend.

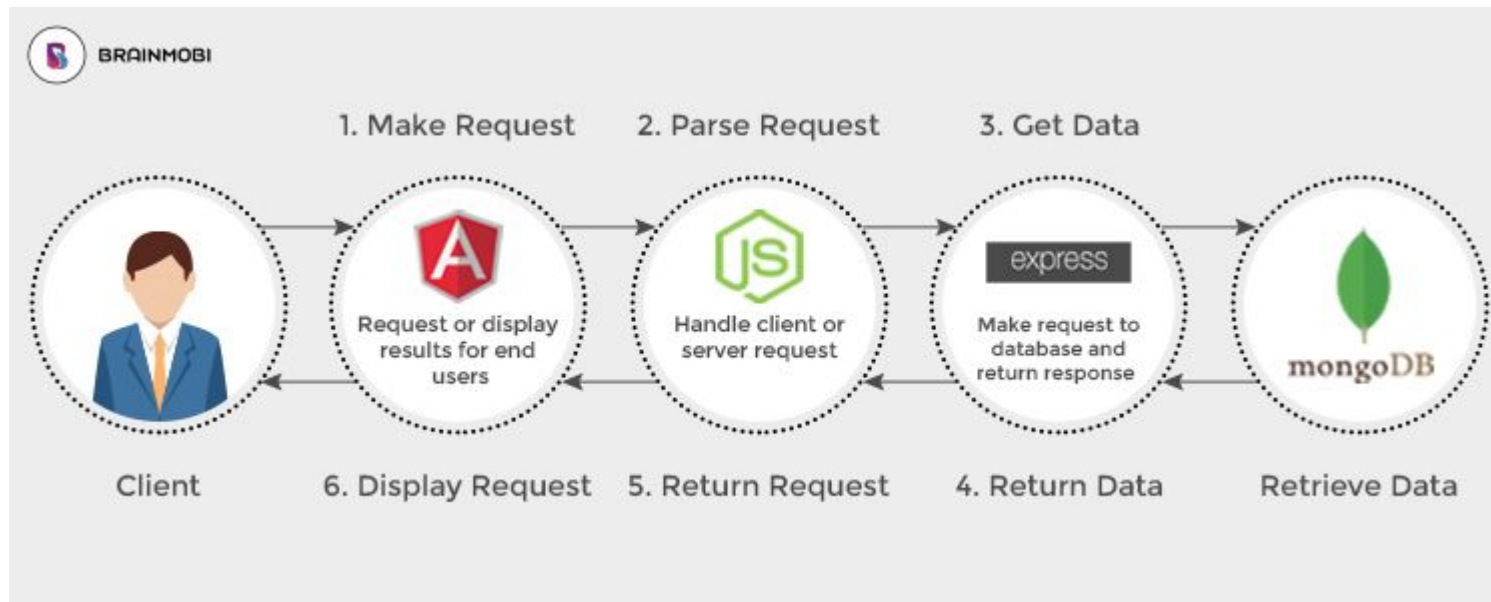


Porquê escolher Angular?

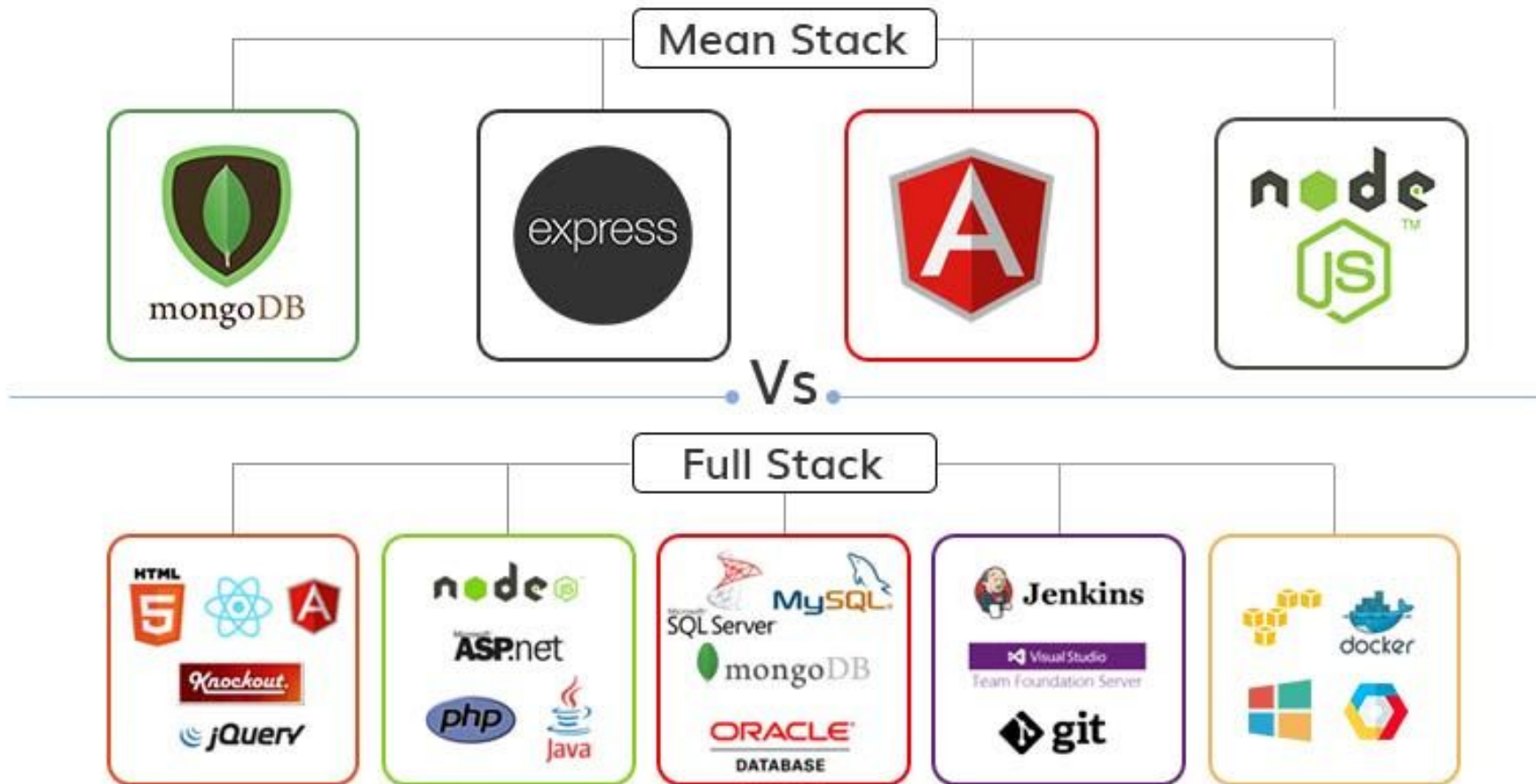
- Criação prática e dinâmica de User Interfaces e Frontend Apps
- Framework com funcionalidades muito completas (router, http, etc)
- Baseada em Typescript
- RxJS - eficiente, permite programação assíncrona
- Orientada a Testes
- Muito popular entre negócios empresariais

Porquê escolher Angular?

- Integrado na MEAN Stack: estrutura javascript de fácil utilização, que é altamente preferível para a criação de websites e aplicações dinâmicas.



Porquê escolher Angular?



Components em Angular

- Os componentes permitem-nos separar a interface em diferentes “peças”, que poderão ser reutilizadas em locais diferentes e poderão ter validações e comportamentos próprios.
- Podem incluir template (HTML), lógica e estilo

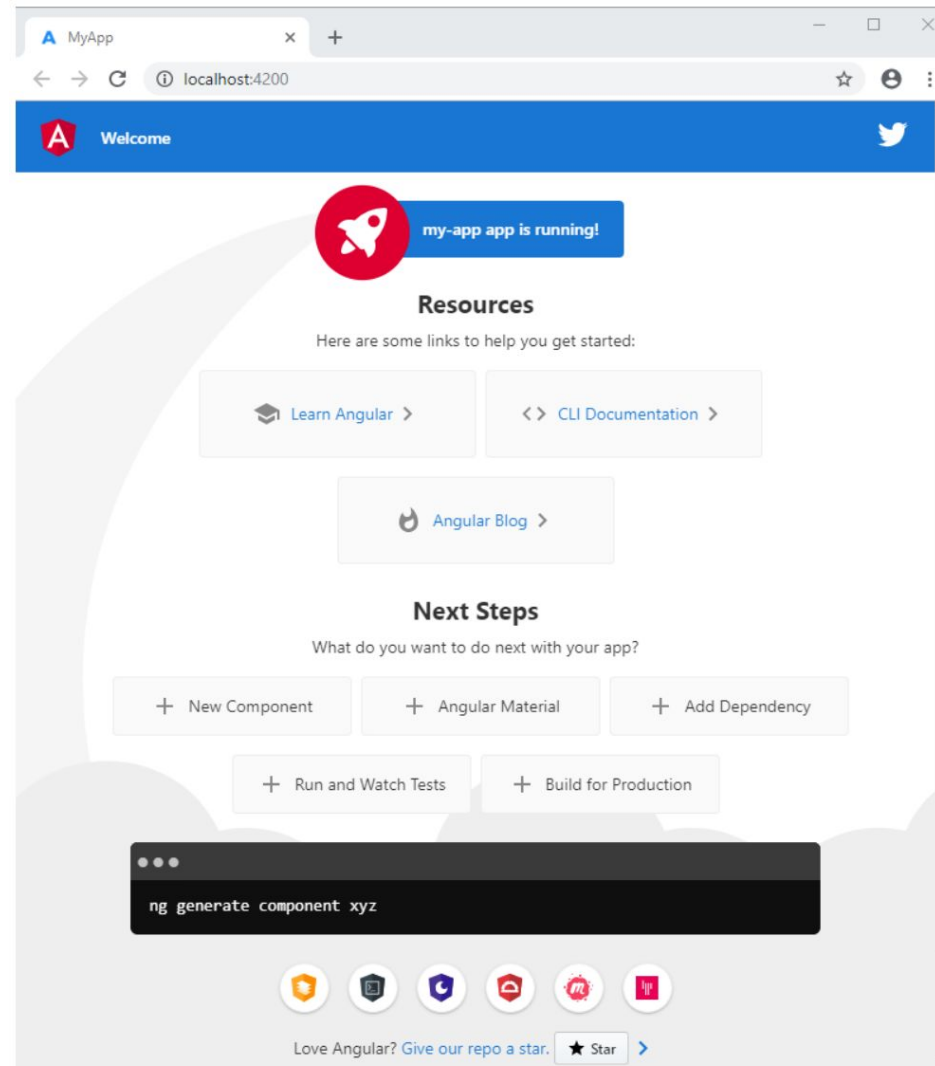
```
@Component({  
  selector:    'app-hero-list',  
  templateUrl: './hero-list.component.html',  
  providers:   [ HeroService ]  
})  
export class HeroListComponent implements OnInit {  
  /* . . . */  
}
```

Serviços em Angular

- Servem para aumentar a modularidade e a reusabilidade dos componentes
- Separando a “View” do componente relacionada com a funcionalidade, de outros tipos de processamento conseguimos obter classes de componentes mais eficientes e robustas
- Assim, o componente pode delegar tarefas aos serviços. Por exemplo: Validar inputs do user ou obter dados do servidor

**Mas como é que
começamos a mexer nisto?**

Criação de aplicação Angular



Angular CLI


- “Command-line” interface (CLI) é uma ferramenta standard de desenvolvimento em Angular.
- Permite-nos:
 - Simplicidade na criação de aplicações em Angular;
 - Rapidez na execução de “production builds”
 - Agilidade na criação de componentes, serviços, etc.

Instalação de Angular CLI e Criação de App


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



```
ng new my-app
```



```
cd my-app  
ng serve --open
```



Exercício 1

- Na aplicação Angular que acabámos de criar:
 - Explore os ficheiros e o output gerados;
 - Substitua o conteúdo e imprima na página o famoso “Hello World”

[Angular - Workspace and project file structure](#)

Exercício 1

Uma resolução interessante poderia ser reutilizando @Component:

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

@Component({
  selector: 'hello-world',
  template: `
    <h2>Hello World</h2>
    <p>This is my first component!</p>
  `
})
export class HelloWorldComponent {
  // The code in this class drives the component's
  behavior.
}
```

Interpolação de conteúdo

- Quando queremos intercalar valores dinâmicos num bloco HTML, devemos utilizar as chavetas à volta do nosso valor e defini-lo no componente

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

@Component ({
  selector: 'hello-world-interpolation',
  templateUrl:
    './hello-world-interpolation.component.html'
})
export class HelloWorldInterpolationComponent {
  message = 'Hello, World!';
}
```

<p>{{ message }}</p>



@Input

- Para garantir a reusabilidade dos componentes, um padrão comum é a partilha de dados entre um componente pai e um ou mais componentes filho.

src/app/item-detail/item-detail.component.ts

```
import { Component, Input } from '@angular/core'; // First, import Input
export class ItemDetailComponent {
  @Input() item = ''; // decorate the property with @Input()
}
```

src/app/item-detail/item-detail.component.html

```
<p>
  Today's item: {{item}}
</p>
```

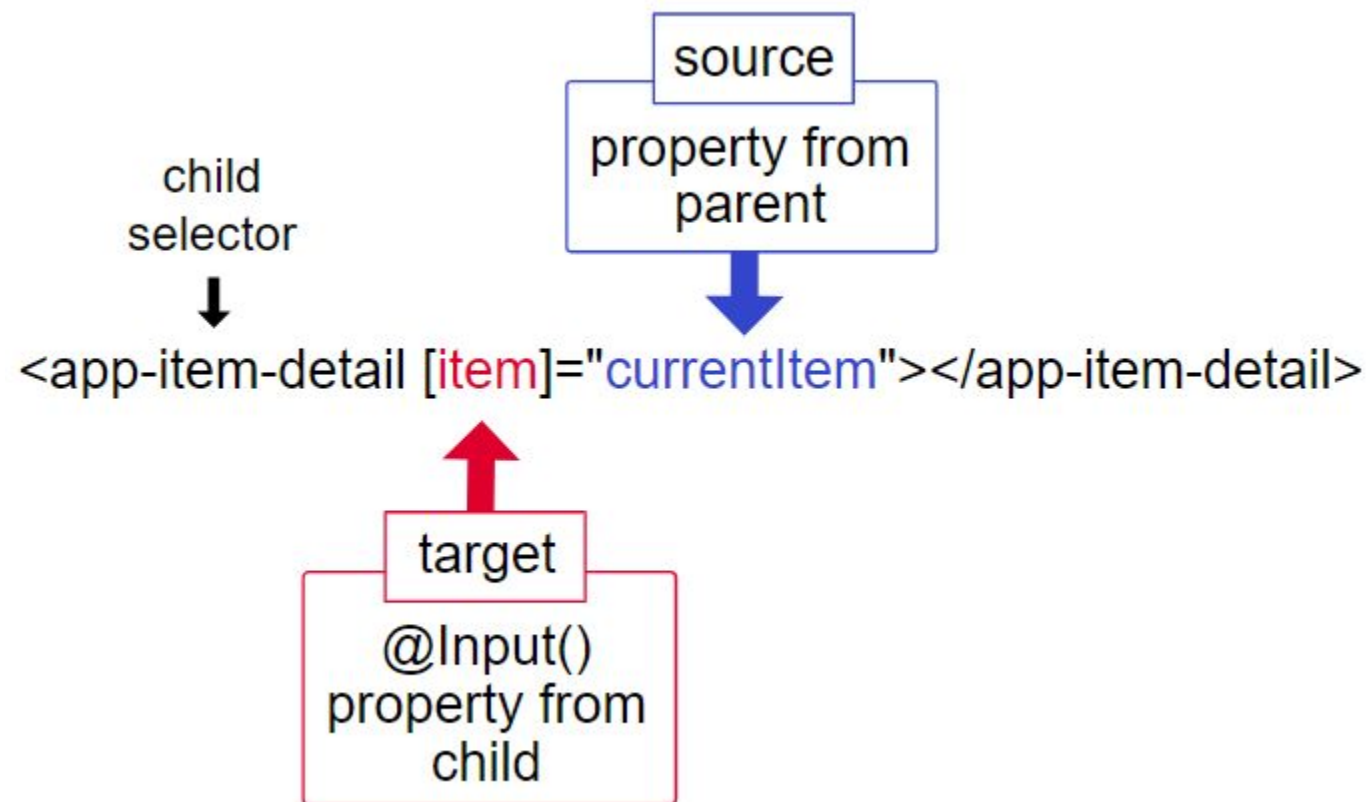
src/app/app.component.html

```
<app-item-detail [item]="currentItem"></app-item-detail>
```

src/app/app.component.ts

```
export class AppComponent {
  currentItem = 'Television';
}
```

@Input



Exercício 2

- Utilizando esta lógica de inputs crie uma aplicação estática que mostre a piada do dia.

Ex: **Piada do dia:** O que são dois pontos cinza no oceano? Twobarões

Exercício 3

- Pretende-se que seja criada a livraria UPskill. Implemente um componente livro com as seguintes informações:
 - Título do livro
 - Autor
 - ISSN

Exercício 4 (TA)

- No mesmo projeto adicione uma imagem do livro correspondente e apresente 5 sugestões de leitura ao utilizador. Aproveite e crie também um novo componente: um campo inicial com o nome da livraria e um campo de pesquisa (por agora estático) de livros. Não se esqueça da experiência do utilizador - utilize as opções de estilo que achar convenientes para por a aplicação bonita.



O futuro profissional começa aqui

iscte

INSTITUTO
UNIVERSITÁRIO
DE LISBOA



emprego
digital



UPskill