

# Angular



# O que é o Angular?

- O Angular é um framework de desenvolvimento de aplicações web.
- Possui código aberto e é mantido pelo Google.
- É utilizado para construir aplicações de páginas única.



# Principais características do Angular

- **Arquitetura baseada em componentes:** Cada parte da interface do usuário é encapsulada em um componente.
- **Baseado em TypeScript:** É uma linguagem estática ao JavaScript. Isso ajuda a detectar erros em tempo de compilação.
- **Injeção de dependências:** Facilita a gestão de serviços e a comunicação entre componentes.
- **Roteamento:** Possui um sistema de roteamento integrado.



# Principais características do Angular

- **Binding de dados:** Permite que as alterações no modelo de dados sejam refletidas na interface do usuário e vice-versa.;
- **Diretivas:** São uma forma de estender o HTML com novo comportamentos.
- **RxJS:** Utiliza a biblioteca RxJS para programação reativa, permitindo trabalhar com fluxos de dados assíncronos e eventos.



# Requisitos



Node.js



Angular CLI



Editor de Código



Navegador atualizado



@fpfttech.educacional

# Node.js

- Node.js é um ambiente de execução de JavaScript que permite executar código do lado do servidor.
- Ele construído com sobre o motor V8 do Google Chrome e compila o código de máquina nativo.
- Download: [Node.js — Executar a JavaScript em Toda Parte](#)
- Comando para verificar a instalação: **node --version**



# Node Version Manager (NVM)

- Gerenciador de Versão do Node.
- Utiliza a linha de comando para instalar as versões do Node.js
- Download: [Releases · coreybutler/nvm-windows](https://github.com/coreybutler/nvm-windows/releases)
- Comando para verificar a instalação: **nvm --version**



@fpfttech.educacional

# Node Package Manager (NPM)

- Repositório onde são armazenados bibliotecas JavaScript.
- É instalado junto ao Node.js
- Comando para verificar a instalação: **npm --version**





# Angular CLI

- Angular CLI é uma ferramenta de linha de comando.
- Foi criada para simplificar o processo de criação, construção, manutenção de aplicações Angular.
- Foi criada em 2016 como parte do esforço para melhorar o projeto Angular.



@fpfttech.educacional

# Angular CLI

- Como instalar: **npm install -g @angular/cli**
- Comando para verificar a instalação: **ng version**
- Documentação: **ng help**
  - [CLI Reference · Overview · Angular](#)



# Como criar um projeto Angular?

- Comando: **ng new <project-name> --routing**
- **ng**: Invoca a Angular CLI
- **new**: Informa que queremos criar um novo projeto.
- **<project-name>**: Nome do projeto
- **--routing**: Informa que queremos utilizar a estrutura de rotas do Angular.



# História

- O Angular inicialmente foi criado com o nome AngularJS.
- Foi criado em 2009 por Misko Hevery e Adam Abrons.
- Em 2016 a equipe do projeto anunciou que reescreveria completamente o projeto, resultando no Angular 2.
- A nova versão teve mudanças significativas, é foi construída em TypeScript em vez de JavaScript.



# Versionamento

- O Angular adora o clíco de lançamento semestral.
- Atualmente estamos na versão 18 como Long Term Support (LTS).
- Mas já possuímos a versão 19 para uso de desenvolvedores, que deve entrar em LTS em novembro de 2025.



# Compatibilidade

Angular	Node.js	TypeScript	RxJS
19.2.x	<code>^18.19.1    ^20.11.1    ^22.0.0</code>	<code>&gt;=5.5.0 &lt;5.9.0</code>	<code>^6.5.3    ^7.4.0</code>
19.1.x	<code>^18.19.1    ^20.11.1    ^22.0.0</code>	<code>&gt;=5.5.0 &lt;5.8.0</code>	<code>^6.5.3    ^7.4.0</code>
19.0.x	<code>^18.19.1    ^20.11.1    ^22.0.0</code>	<code>&gt;=5.5.0 &lt;5.7.0</code>	<code>^6.5.3    ^7.4.0</code>
18.1.x    18.2.x	<code>^18.19.1    ^20.11.1    ^22.0.0</code>	<code>&gt;=5.4.0 &lt;5.6.0</code>	<code>^6.5.3    ^7.4.0</code>
18.0.x	<code>^18.19.1    ^20.11.1    ^22.0.0</code>	<code>&gt;=5.4.0 &lt;5.5.0</code>	<code>^6.5.3    ^7.4.0</code>
17.3.x	<code>^18.13.0    ^20.9.0</code>	<code>&gt;=5.2.0 &lt;5.5.0</code>	<code>^6.5.3    ^7.4.0</code>
17.1.x    17.2.x	<code>^18.13.0    ^20.9.0</code>	<code>&gt;=5.2.0 &lt;5.4.0</code>	<code>^6.5.3    ^7.4.0</code>
17.0.x	<code>^18.13.0    ^20.9.0</code>	<code>&gt;=5.2.0 &lt;5.3.0</code>	<code>^6.5.3    ^7.4.0</code>

- [Version compatibility · Angular](#)



# Projeto: Hello World

- Objetivos:
  - Ter o primeiro contato com o Angular.
  - Descobrir os tópicos fundamentais do Angular.
  - Crie um repositório de exemplos, que iram servir como referência nos próximos projetos.
- Utilizando o comando abaixo, crie um projeto com o nome “hello-world”.
- Comando: **ng new hello-world --routing**



## Projeto: Hello World

- No terminal, execute o comando para iniciar o projeto: **ng serve -o**
  - O parâmetro -o abre automaticamente o navegador padrão com a URL do projeto.
- Edite o componente app.component.html.
- Apague todo o código HTML.
- Escreva: **<h1>Hello World</h1>**





# Estrutura de um projeto Angular

- Pastas:
  - **node\_modules**: É onde são armazenados dependências do projeto.
  - **src**: Código-fonte da aplicação.
  - **app**: Componente principal da aplicação.
  - **assets**: Recursos estáticos como imagens e fontes.
  - **environments**: Configurações de ambiente.



@fpfttech.educacional

# Estrutura de um projeto Angular

- **index.html**: Arquivo principal.
- **main.ts**: Ponto de entrada da aplicação.
- **styles.css**: Estilos globais da aplicação.
- **angular.json**: Configurações do projeto Angular.
- **package.json**: Dependências e scripts do projeto.
- **tsconfig.json**: Configurações do TypeScript.
- **tslint.json**: Configurações do TSLint
- **app.config.ts**: Configurações da aplicação.
- **app.component.js**: Componente raiz.
- **app.routes.ts**: Configurações de rotas.



# README.md

- Serve como uma introdução e guia para o projeto, fornecendo informações essenciais que ajudam os outros desenvolvedores e usuários a entenderem o propósito, a configuração e o uso do projeto.
- Utiliza uma linguagem chamada **Markdown**, que é uma linguagem de marcação assim como o HTML.
- No Markdown é possível criar elementos como:
  - Títulos
  - Listas
  - Tabelas
  - Links
  - Exemplificar códigos



# Cabeçalhos

- Cabeçalhos/Títulos são criados utilizando o símbolo # para criar cabeçalhos em diferentes níveis

```
# Cabeçalho de Nível 1  
## Cabeçalho de Nível 2  
### Cabeçalho de Nível 3
```



# Estilos de Texto

- Usando asteriscos ou sublinhados para itálico e negrito.

*\*itálico\** ou *itálico*

**\*\*negrito\*\*** ou **negrito**



# Links

- Adicionando hyperlinks:

```
[Documentação do Angular](https://angular.dev/)
```

- Imagens

```
![Texto alternativo](url-da-imagem)
```



# Códigos

- Destacando trechos de código:

```
~ código em linha ~
```

- Blocos de código:

```
~ ~ ~ bash  
undefined  
~ ~ ~
```



# Componentes

- São blocos de construção dos aplicativos Angular.
- Cada componente representa uma parte de uma página.
- Cada componente é uma classe TypeScript.
- Cada componente possui:
  - Um template (HTML)
  - Uma lógica (TypeScript)
  - Estilos (CSS)
- O componente permite encapsular a lógica e a apresentação, tornando o código mais modular e reutilizável.





# Estrutura de um Componente

- Um componente Angular é composto por três partes principais:
  - **Classe:** A classe do componente contém a lógica do aplicativo. E onde são definidas as propriedades e métodos que controlam o comportamento do componente.
  - **Template:** O template é o HTML que define a estrutura visual do componente.
  - **Estilos:** Os estilos são aplicados ao componente e podem ser definidos em um arquivo CSS separado ou diretamente no decorador do componente. Eles afetam apenas o componente em questão.



# Como criar um componente?

- Utilizando a Angular CLI, execute o seguinte comando no terminal: **ng generate component <component-name>**
- **ng**: Invoca a Angular CLI.
- **generate**: Comando para gerar algo.
- **component**: O que é para ser gerado.
- **<component-name>**: Nome do componente.



@fpfttech.educacional

# @Component

- O decorator @Component é aplicado a uma classe e aceita um objeto de configurações que define várias propriedades. As principais propriedades são:
  - **seletor**: Define o seletor que será usado para instanciar o componente HTML.
  - **templateUrl**: Especifica o caminho para o arquivo HTML que contém o template do componente.
  - **styleUrls**: Um array que contém os caminhos para os arquivos CSS.



# Ciclo de Vida de um Componente

- O ciclo de vida de um componente Angular se refere ao conjunto de **etapas** que um componente passa desde a sua criação até a destruição.
- O Angular fornece uma serie **hooks** que permitem que você execute código em momentos específicos do ciclo de vida.
- Esses **hooks** são métodos que podem ser implementados em uma classe de componente.
- É implementado usando a palavra reservada **implements**.



- **ngOnInit()**: Chamado uma vez, após a primeira exibição do componente e após as propriedades serem inicializadas.
- **ngOnChanges()**: Chamado sempre que uma propriedade vinculada a dados do componente muda.
- **ngOnDestroy()**: Chamado uma vez antes que o componente seja destruído.
- Documentação: [Lifecycle · Angular](#)



# Interpolation

- É uma técnica utilizada para exibir dados dinâmicos no template HTML.
- Ela permite que você insira valores de propriedades da classe do componente diretamente no HTML.



# Property-binding

- É uma forma de vincular valores do componente TypeScript às propriedades dos elementos HTML.
- Permite que seja definida dinamicamente o valor de uma propriedade de um elemento.
- É feito utilizando colchetes [ ] ao redor da propriedade de um elemento HTML.



# Event-binding

- É um mecanismo que permite a um componente responder a eventos do usuário que ocorrem no template, como cliques, pressionamento de teclas, movimentos do mouse, entre outros.
- A sintaxe do event-binding utiliza parênteses ( ) ao redor do nome do evento e dentro deles coloca-se a expressão ou método a ser chamado no componente.
- Os eventos fazem referência aos eventos existentes em JavaScript.





# Two-Way Data Binding

- É uma técnica que permite a sincronização automática e simultânea dos dados entre o modelo (a classe do componente) e a visão (o template HTML).
- É implementado usando a diretiva **[(ngModel)]**, que combina o property-binding e o event-binding.
- Para usar o **ngModel** você precisa importar o modulo **FormsModule**.



# Formulários: Template-driven

- Formulários do tipo template-driven são uma abordagem para criar formulários onde a maior parte da lógica e configurações acontecem diretamente no template HTML.
- É ideal para formulários simples.
- É especialmente intuitiva para quem está começando com o Angular.
- Necessário importar o módulo **FormsModule**.



# Formulários: Reactive

- Formulários do tipo Reactive Forms no Angular são uma abordagem para criar formulários onde toda a lógica e configurações são definidas diretamente na classe TypeScript do componente.
- Permite validações mais flexíveis, sendo síncronas ou assíncronas, além e reagir dinamicamente a mudanças nos controles.
- Necessário importar o modulo: **ReactiveFormsModule**.



# Validators

- **required:** Verifica se o controle não está vazio.
- **minLength/maxLength:** Verifica se o valor do controle tem um comprimento mínimo/máximo de acordo com o especificado.
- **pattern:** Verifica se o valor do controle corresponde a uma expressão regular.
- **email:** Verificar se o valor do controle é um endereço de e-mail válido.



# FormBuilder

- Utilizar o **FormBuilder** é uma forma mais simples e prática de criar um “**FormGroup**” e seus controles.
- Reduz a verbosidade ao criar formulários.
- Necessário importar o módulo: **ReactiveFormsModule**.



# Gestão de Estados

- O Angular fornece uma maneira de gerenciar o estado dos formulários, incluindo a validação, interação do usuário e a manipulação de dados.
- Os formulários no Angular possuem vários estados que podem ser monitorados e gerenciados:
  - **valid/invalid**: Indica se o formulário ou o controle especificado é válido ou inválido.
  - **dirty**: Indica se o usuário interagiu com o controle, ou seja, se o valor foi alterado.
  - **touched**: Indica se o controle foi tocado, ou seja, se o usuário focou e saiu do controle.
  - **pristine**: O oposto de Dirty, indica se o controle não foi alterado desde a sua inicialização.
  - **disabled**: Indica se o controle está desabilitado e não pode ser interagido.



# Roteamento

- As rotas definem a relação em URL e Componentes.
- Cada rota mapeia a URL específica de um componente que deve ser exibido quando essa URL é acessada.
- É fornecido pelo modulo **RouterModule** de **@angular/router**.
- O componente para exibição do conteúdo é conhecido como **<router-outlet>**.



# Diretivas

- Diretivas que alteram a aparência ou comportamento de um elemento DOM existente.
- Não modificam a estrutura do DOM (não adicionam ou removem elementos).
- São aplicadas como atributos nos elementos HTML.
- Exemplo clássico: alterar estilos, classes, comportamento visual.
- São divididas em:
  - **Diretivas de Componentes:** São diretivas que possuem um template associado.
  - **Diretivas Estruturais:** Alteram a estrutura do DOM, adicionando ou removendo elementos.
  - **Diretivas de Atributos:** Alteram a aparência ou o comportamento de um elemento existente.





# Diretivas de Componente

- São diretivas que possuem um template associado. Cada componente no Angular é, na verdade, uma diretiva de componente.
- Exemplo: **@Component** é uma diretiva de componente.



# Diretivas Estruturais

- Alteram a estrutura do DOM, adicionando ou removendo elementos. Elas geralmente começam com um asterisco (\*) ou um arroba (@) no template.
- Exemplos:
  - **\*ngIf**: Adiciona ou remove um elemento com base em uma condição.
  - **\*ngFor**: Repete um elemento para cada item em uma lista.
  - **\*ngSwitch**: Alterna entre diferentes elementos com base em uma expressão.



# Exercício

- Crie um componente chamado: “**am-i-old**”.
- Adicione uma input do tipo **number**.
- Valide para receber apenas números utilizando reactive forms do angular.
- Baseado na idade passada, deve ser exibido se a pessoa é criança, adolescente, adulto ou idoso.



# Diretivas de Atributo

- Alteram a aparência ou o comportamento de um elemento existente. Elas não têm um template associado.
- Exemplos:
  - **ngClass**: Adiciona ou remove classes CSS de um elemento.
  - **ngStyle**: Aplica estilos CSS a um elemento.
  - Diretivas personalizadas que você pode criar.



# Property Decorators

- São funções especiais que você aplica a propriedades dentro das classes dos componentes para modificar seu comportamento ou fornecer metadados ao framework Angular.
- Dois dos **property decorators** mais comuns em Angular são:
  - **@Input:** Permite que uma propriedade de um componente receba valores do componente pai, ou seja, é uma forma de passar dados para o componente filho.
  - **@Output:** Permite que um componente emita eventos que podem ser capturados por seu componente pai.



# Pipes

- **Transformadores de dados** que modificam a exibição em templates.
- Usados para **formatar** texto, números, datas e outros valores.
- Podem ser **encadeados** para **múltiplas transformações**.
- Opera nos dados apenas para a visualização, não modifica dados originais.
- Sintaxe Básica
  - Usa o símbolo de pipe vertical (|) em expressões
  - Formato: {{ valor | nomeDoPipe:parametro1:parametro2 }}
  - Pode receber parâmetros opcionais após dois pontos



# Pipes Populares

- **uppercase/lowercase**: Converter texto para MAIÚSCULO/minúsculo.
- **date**: Formatar datas
- **currency**: Formata valores monetários
- **number**: Formata números



@fpfttech.educacional

# Pipe Date

- Transformador especializado para valores de data e hora.
- Converte objetos Date, timestamps ou strings de data em formatos legíveis.
- Adapta-se automaticamente ao locale configurado na aplicação.
- Oferece múltiplos formatos predefinidos e personalizáveis.
- [DatePipe · Angular](#)





# Sintaxe Básica

- `date[:formato][:timezone][:locale]`
- **valor**: Data a ser formatada (Date, número, string ISO)
- **formato**: String de formato ou formato predefinido
- **timezone**: Fuso horário para exibição (opcional)
- **locale**: Configuração regional específica (opcional)



# Pipe number

- Transformador de valores numéricos para exibição formatada.
- Converte números em strings com formatação regional.
- Controla separadores decimais, agrupamento de dígitos e precisão.
- Respeita configurações de localização (locale).
- Estrutura:
  - {inteiros}.{mínimo-decimais}-{máximo-decimais}



# Pipe currency

- Transformador especializado para valores **monetários**.
- Formata números como **moedas** com **símbolos apropriados**.
- Aplica convenções monetárias específicas de cada país/região.
- Integrado com o sistema de internacionalização (i18n) do Angular.
- [DecimalPipe · Angular](#)



# Sintaxe Básica

- `currency[:código][:exibição][:digitosInfo][:locale]`
- **valor**: O número a ser formatado como moeda
- **código**: Código ISO da moeda (ex: 'BRL', 'USD')
- **exibição**: Como exibir o símbolo ('code', 'symbol', 'symbol-narrow')
- **digitosInfo**: Formato de dígitos (similar ao pipe number)
- **locale**: Configuração regional específica



- PrimeNG é uma biblioteca de componentes de interface do usuário para Angular.
- Oferecendo uma ampla gama de componentes ricos e personalizáveis.
- Desenvolvida pela PrimeTek, ela facilita a criação de interfaces atraentes e funcionais em aplicações web, integrando-se perfeitamente ao Angular.
- [PrimeNG - Angular UI Component Library](#)



# Angular Material

- Biblioteca de componentes **UI oficial** para Angular.
- Baseada no design system **Material Design** do Google.
- Facilita a criação de interfaces modernas, responsivas e acessíveis.
- [Angular Material UI component library](#)



# Versionamento

- Angular Material segue o mesmo ciclo de versões do Angular.
- Versão compatível com a versão principal do Angular:
  - Angular 19 ↔ Angular Material 19
- Lançamentos regulares alinhados com as versões do Angular.
- Atualizações garantem compatibilidade e novidades sincronizadas.



# História

- Lançamento Inicial (2014) – O Google apresentou o Material Design, uma abordagem unificada para interfaces de usuário. Como o Angular já era um framework popular, surgiu a necessidade de uma biblioteca oficial para integrar Material Design ao Angular.
- AngularJS Material (2015) – A primeira versão foi feita para AngularJS (v1.x), fornecendo componentes como botões, diálogos e tabelas seguindo as diretrizes do Material Design.
- Angular Material (2016) – Com o lançamento do Angular 2, foi criada a nova versão do Angular Material, aproveitando as funcionalidades modernas do framework, como componentes reutilizáveis e bindings eficientes.





# Exercício

- Crie um componente chamado “**angular-material**”.
- Utilizando a documentação do Angular Material, crie os seguintes componentes da documentação:
  - input
  - button
  - autocomplete
  - toggle
    - slide
    - button
  - datepicker
  - icons



# Dialog

- Um Dialog é um componente de interface do usuário que exibe informações ou solicitações ao usuário em uma janela modal.
- Utilizado para interações que requerem atenção do usuário, como confirmações, formulários ou mensagens de erro.
- Segue as diretrizes do Material Design, proporcionando uma experiência visual consistente.

