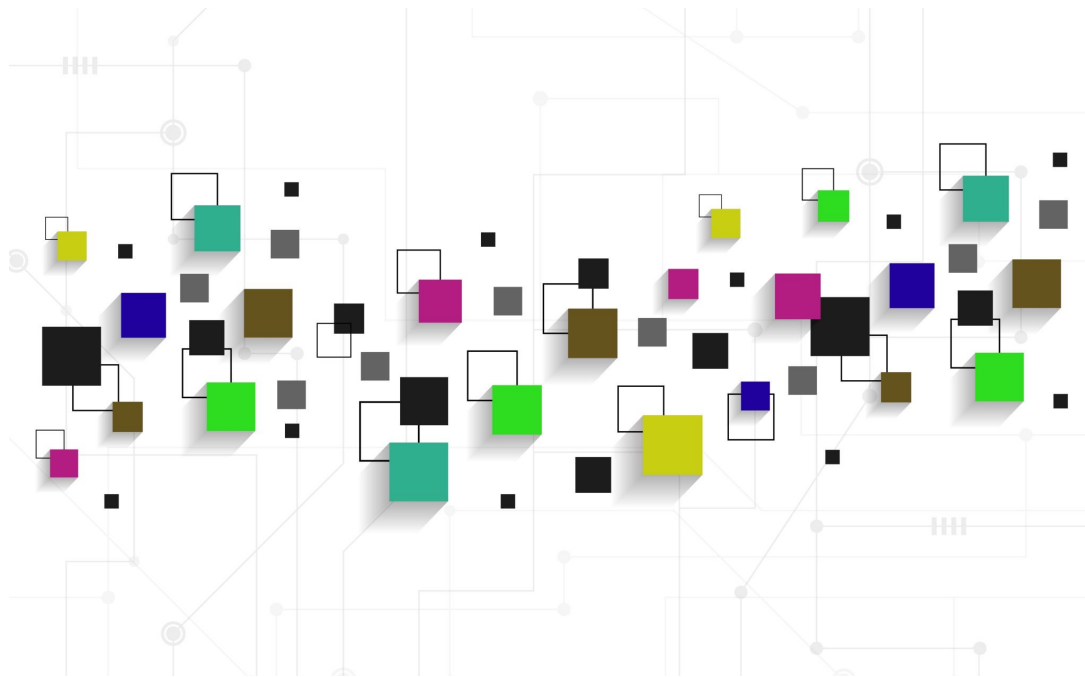




# Introdução



- Angular é um framework de desenvolvimento para a construção front end, que são aplicações da web do lado do cliente;
- Mantido pelo Google e é amplamente utilizado para criar aplicações web dinâmicas e Single Page Applications (SPA);
- Escrito em TypeScript e oferece uma abordagem baseada em componentes para o desenvolvimento de aplicações;
- A arquitetura é pensada no conceito de módulos e componentes



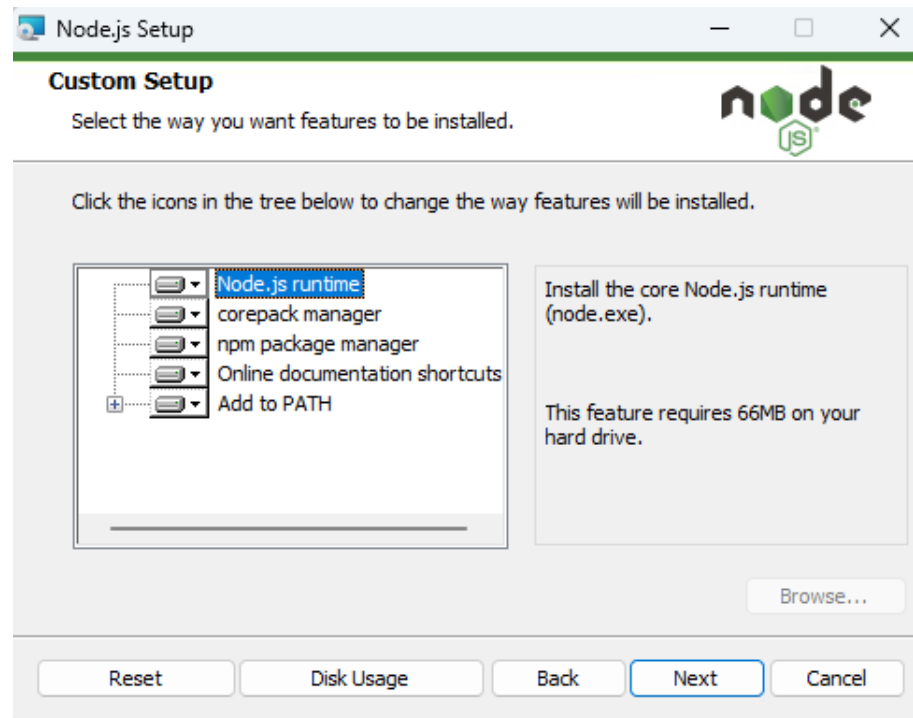
## Single Page Applications (SPA)

- Implementação de aplicativo da web que carrega apenas um único documento da web;
- Em seguida, atualiza o conteúdo do corpo desse único documento por meio de APIs JavaScript, como **XMLHttpRequest** e **Fetch**, quando conteúdo diferente deve ser mostrado.
- Permite que os usuários usem sites sem carregar páginas totalmente novas do servidor;
- Resulta em ganhos de desempenho e uma experiência mais dinâmica;

# Configuração do Ambiente

- Para usar o Angular, é preciso estar ambientado com o seguinte:
  - JavaScript
  - HTML
  - CSS
  - TypeScript
- Para instalar o Angular:
  - Node.js;
    - <https://nodejs.org/dist/v20.10.0/node-v20.10.0-x64.msi>
- Angular, Angular CLI e aplicativos Angular dependem de pacotes npm;

# Configuração do Ambiente



# Configuração do Ambiente

- Depois do Node.js instalado;
  - Abrir o terminal;
  - Digitar: `npm install -g @angular/cli`
  - Para Windows, digitar no powerShell:
    - `Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned`
- Testar:
  - `ng new minhaApp --no-standalone --routing --ssr=false`

# Configuração do Ambiente

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Instale o PowerShell mais recente para obter novos recursos e aprimoramentos! https://aka.ms/PSWindows

PS C:\Users\uesc> npm -v
10.2.3
PS C:\Users\uesc> npm install -g @angular/cli

added 227 packages in 26s

45 packages are looking for funding
  run 'npm fund' for details
npm notice
npm notice New patch version of npm available! 10.2.3 -> 10.2.4
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.2.4
npm notice Run npm install -g npm@10.2.4 to update!
npm notice
PS C:\Users\uesc> Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
PS C:\Users\uesc> S|
```

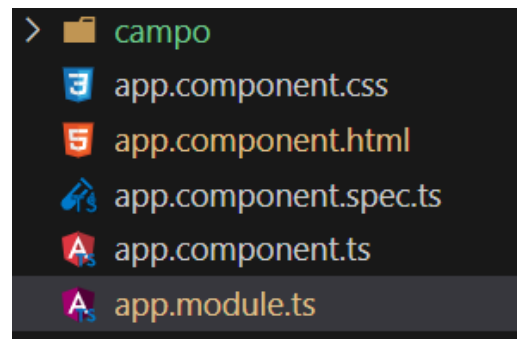
# Hello World

- Criar um novo espaço de trabalho e uma aplicação inicial;
  - Crie uma pasta onde será armazenados suas aplicações angular;
  - Entre dentro da pasta e digite:
    - `ng new minhaApp --no-standalone --routing --ssr=false`
    - `ng serve`
    - `localhost: 4200`



# Módulos

- Os módulos são utilizados para organizar a aplicação em blocos funcionais;
- Um módulo agrupa componentes, diretivas, pipes e serviços relacionados;
- Cada aplicação Angular tem pelo menos um módulo raiz chamado de AppModule.



```
export class AppModule { }
```

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';
import {TabelaComponent} from './tabela/tabela.component';
import { CirculoComponent } from './circulo/circulo.component';
import { FormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent,
    TabelaComponent,
    CirculoComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

# Componentes

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

@Component({
  selector: 'app-circulo',
  templateUrl: './circulo.component.html',
  styleUrls: ['./circulo.component.css']
})
export class CirculoComponent implements OnInit {
```

- Cada aplicativo Angular possui pelo menos um componente;
- Define uma classe que encapsulam a lógica e a apresentação de parte da interface do usuário e está associada a um modelo HTML que define uma visualização a ser exibida em um ambiente de destino
- Conecta uma hierarquia de componentes ao modelo de objeto de documento da página (DOM);

# Diretivas

- Instruções no HTML que dizem ao Angular como modificar ou manipular o DOM (instruções em DOM);
- Permite estender HTML com novos atributos.
- O Angular possui diretivas embutidas, como **ngIf** e **ngFor**, que são usadas para controle de fluxo e repetição no template.

# Categorias de Directives

- Attribute Directives;
- Structural Directives;
- Components.

# Attribute Directives

- As Diretivas de Atributos são responsáveis por manipular a **aparência** e o **comportamento** dos elementos DOM;
- Alterar o estilo dos elementos DOM;
- Usadas para ocultar ou mostrar condicionalmente elementos DOM específicos;
- Angular fornece muitas diretivas de atributos integradas, como NgStyle, NgClass, etc.
- Também é possível criar as próprias diretivas de atributos personalizadas para a funcionalidade desejada.

# Structural Directives;

- Responsáveis por alterar a estrutura do DOM;
- Funcionam adicionando ou removendo elementos do DOM, ao contrário das Diretivas de Atributos que apenas alteram a aparência e o comportamento do elemento;
- O nome sempre começa com um prefixo asterisco (\*), diferentemente da diretiva de atributos que não contém nenhum prefixo;
- As três diretivas estruturais integradas ao angular mais populares são:
  - \*NgIf, \*NgFor e \*NgSwitch.

# Components

- São diretivas com templates;
- A diferença entre Componentes e os outros dois tipos de diretivas é o Template;
- As Diretivas de Atributos e Estruturais não possuem templates;
- O componente é uma versão mais limpa das Diretivas com um template.



# ngStyle

```
<div class="flex-item">  
  <div [ngStyle]="{'backgroundColor': getTempo(), 'color': 'white', 'padding':'10px'}">status</div>  
</div>
```

```
getTempo() {  
  return this.chovendo ? 'blue' : 'red';  
}
```



- Diretiva NgStyle utilizada para alterar o estilo do elemento div com base no resultado retornado pela função getTempo()

# \*ngIf

```
<div class="flex-item">
  
  <ng-template #templateDoAtletismo>
    
  </ng-template>
</div>
```


```
export class CampoComponent {
  chovendo=false;
  correndo=false;
```


- Diretiva estrutural usada para adicionar elementos ao DOM de acordo com alguma condição.

# ngClass

```
<div class="flex-item">
  
  <ng-template #templateDoAtletismo>
    
  </ng-template>
</div>
```

- Diretiva que adiciona ou remove classes CSS em um elemento HTML;

```
.coelhoQuente {
  border:  red solid;
}

.coelhoFrio{
  border:  blue solid;
}
```

# \*ngFor

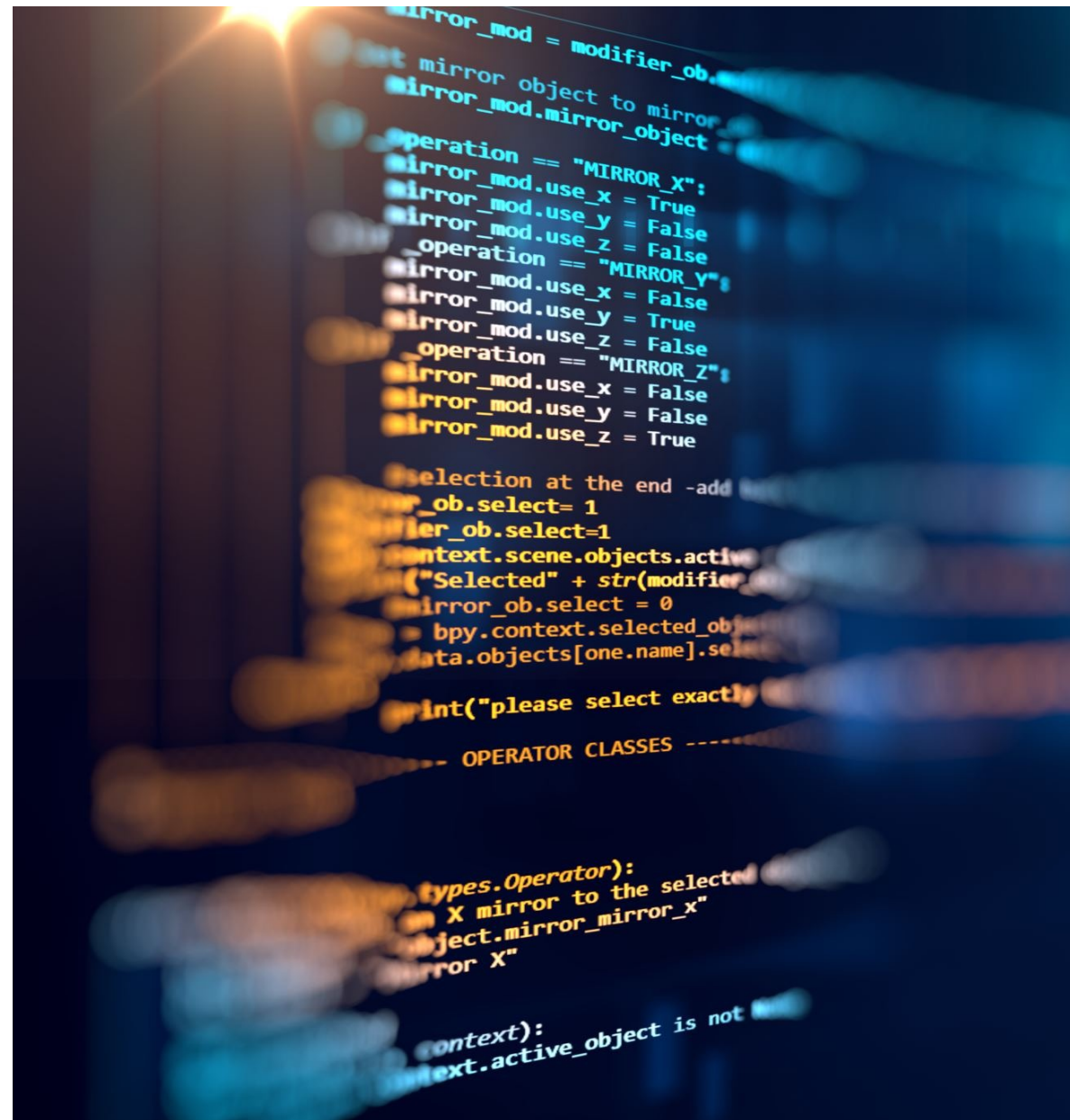
```
avioes = [  
  { engine: 'Merlin 66', model: 'Spitfire', manufacturer: 'Supermarine' },  
  { engine: 'Double Wasp', model: 'Corsair', manufacturer: 'Vought' },  
  { engine: 'R-2800-59', model: 'Thunderbolt P47', manufacturer: 'Republic' },  
  { engine: 'DB 605', model: 'Bf 109', manufacturer: 'Messerschmitt' },  
  { engine: 'Jumo 213', model: 'Fw 190', manufacturer: 'Focke-Wulf' },  
  { engine: 'Merlin', model: 'Mustang P51', manufacturer: 'North American' },  
];
```

```
<div class="flex-item">  
  <ol>  
    <li *ngFor="let aero of avioes">  
      Engine: {{aero.engine}}, Model: {{aero.model}},  
      Manufacturer: {{aero.manufacturer}}  
    </li>  
  </ol>  
</div>
```

- Percorrer elementos de um Array e exibir os dados no template (View);

## Serviços


- Utilizados para encapsular lógica de negócios compartilhada entre componentes;
- São injetáveis e podem ser compartilhados por toda a aplicação;
- Ideias para manipulação de dados, chamadas de API, e outros.





# Criando Componente Tabela manualmente

- Adicionar uma pasta tabela em app/
- Criar o arquivo tabela.component.ts na pasta tabela;
- Criar o arquivo tabela.component.html
- Criar o arquivo tabela.component.css



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

@Component({
  selector: 'app-tabela',
  templateUrl: './tabela.component.html',
  styleUrls: ['./tabela.component.css']
})

export class TabelaComponent {

}
```

## Adicionar o componente tabela no app.module.ts

- Registrar o componente tabela em app.modules.ts no decorator @NgModule
- Adicionar a tag <app-tabela> em app.component.html



```
<h1>Bem vindo o módulo front-end da Residência de Software</h1>

<hr/>

<app-tabela></app-tabela>
```

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

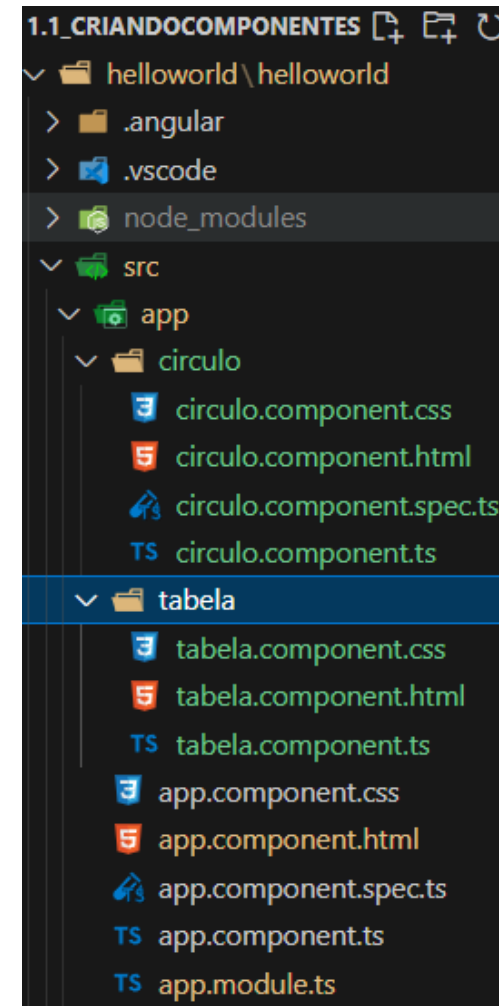
import { AppComponent } from './app.component';
import { TabelaComponent } from './tabela/tabela.component';

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



# Criando componente com CLI

- `ng generate component <nomeDoComponente>`
- Forma abreviada:
  - `ng g c <nomeDoComponente>`



# Seletores

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'exemplo1';
}
```


- Um seletor é uma das propriedades do objeto que usamos junto com a configuração do componente;
- é usado para identificar cada componente exclusivamente na árvore de componentes;
- A propriedade selector juntamente com seu valor <app-root> é usada para identificar o componente do aplicativo na árvore HTML DOM uma vez que ele é renderizado no navegador.

```
@Component({  
  selector: 'app-tucano',  
  //selector: '[app-tucano]',  
  //selector: '.app-tucano',  
  templateUrl: './tucano.component.html',  
  styleUrls: ['./tucano.component.css']  
})
```

# Seletor como nome do elemento

```
<!-- estilo de seletor de elemento do componente tucano -->  
<app-tucano></app-tucano>
```

o nome do seletor representa o componente como o elemento completo;



Seletor como  
Atributo

```
@Component({  
  //selector: 'app-tucano',  
  selector: '[app-tucano]',  
  templateUrl: './tucano.component.html',  
  styleUrls: ['./tucano.component.css']  
})  
export class TucanoComponent {  
  
}
```

```
<!-- estilo de seletor de atributo do componente tucano -->  
<h2>Componente em Ação</h2>  
<div app-tucano></div>
```

- seletor como um atributo de um elemento, assim como é feito com outros elementos HTML.

```
@Component({  
  //selector: 'app-tucano',  
  //selector: '[app-tucano]',  
  selector: '.app-tucano',  
  templateUrl: './tucano.component.html',  
  styleUrls: ['./tucano.component.css']  
})
```

## Seletor como uma Classe

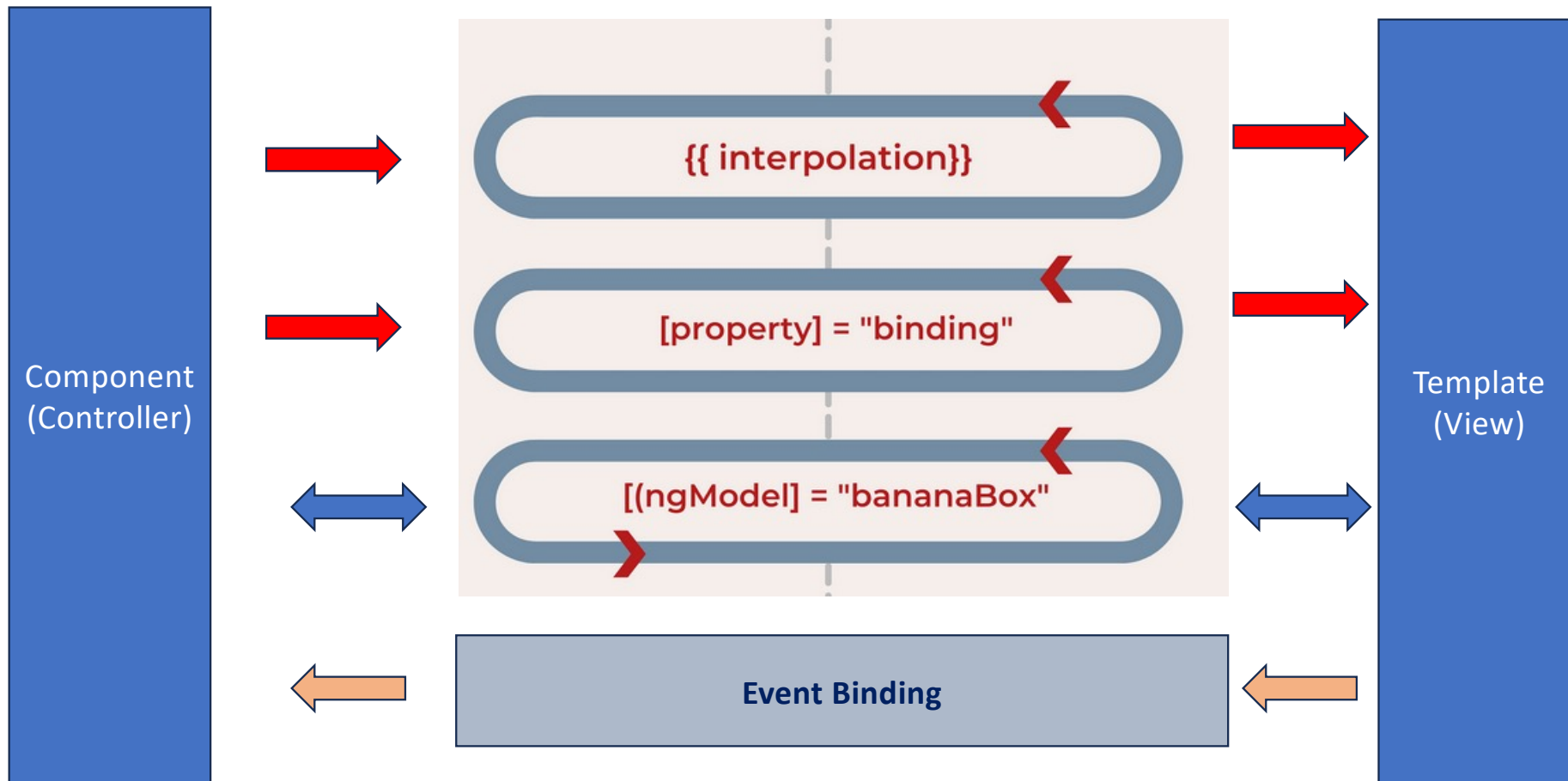
```
<!-- estilo de seletor de classe do componente tucano -->  
<div class="app-tucano"></div>
```

- nome do seletor como a classe e o elemento será convertido para a classe no HTML DOM.

# Data binding

- O data binding (vinculação de dados) é a sincronização entre o modelo (model) e a visualização (view);
- Angular segue a arquitetura MVC, que significa Model-View-Controller;
- No template (View), precisamos acessar os dados fornecidos pelo componente.
- Os dados são acessados utilizando data binding.

# Data binding



# {{Interpolation}} binding

```
export class AraraComponent {  
  nome:string = 'Arara';  
  imagem:string = 'data:image/jpeg;base64,/9j/4';  
  descricao: string = 'A arara-canindé, também';  
  link: string = 'https://pt.wikipedia.org/wiki';  
  
  public getToString(): string {  
    return this.nome + ' - ' + this.descricao;  
  }  
}
```

```
<!-- interpolation -->  
<div class="descricao">  
  {{ getToString() }}  
</div>  
<br>  

```



```
export class AraraComponent {
  nome:string = 'Arara';
  imagem:string = 'data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAD/2wCEAAkGBx';
  descricao: string = 'A arara-canindé, também conhecida como arara-de-barriga-a';
  link: string = 'https://pt.wikipedia.org/wiki/Arara-canind%C3%A9';

  url2: string = 'https://i0.wp.com/blog.bioparquedorio.com.br/wp-content/uploa';

  public getToString(): string {
    return this.nome + ' - ' + this.descricao;
  }
}
```

```
<!-- property binding -->
<h2>Arara2</h2>
<img [src]="url2" alt="Arara Canindé2">
```

## Property binding

- É possível vincular todas as propriedades dos elementos HTML

[https://www.w3schools.com/jsref/dom\\_obj\\_all.asp](https://www.w3schools.com/jsref/dom_obj_all.asp)

# ngModel

- A diretiva ng-model vincula o valor dos controles HTML (input, select, textarea) aos dados do aplicativo;

```
<!--Event binding 4, two way binding-->  
<h3>ngModel</h3>  
<input type="text" [(ngModel)]= "nome2">  
<p id="textodigitado">{{ nome2 }}</p>
```

```
nome2:string = 'Tamanduá';
```

App.module.ts

```
imports: [  
  BrowserModule,  
  FormsModule  
],
```

# Event binding

- Permite ouvir e responder às ações do usuário, como pressionamentos de teclas, movimentos do mouse, cliques e toques.

```
<!--Event binding-->  
<button (click)="onClick()">Clique aqui</button>
```

```
public onClick(): void {  
  alert('Clicou em mim!');  
}
```

```
<button (mouseenter)="entrouBotao()" (mouseleave)="saiuBotao()"  
(dblclick)="cliqueDuplo()" (click)="cliqueSimples()">Eventos</button>
```

# Parâmetros com Event binding

- Uma maneira comum de tratar eventos é passar o objeto de evento, \$event, para o método que trata o evento;
- O objeto \$event geralmente contém informações de que o método precisa, como o nome de um usuário ou o URL de uma imagem;
- O evento alvo determina a forma do objeto \$event;
- Se o evento de destino for um evento de elemento DOM nativo, então \$event será um objeto de evento DOM, com propriedades como target e target.value.

## Parâmetros com Event binding

```
<label for="txtEventoParametro">Digite um texto:</label>  
<input type="text" (input)="digitandoTexto($event)" id="txtEventoParametro">
```

```
<p id="textodigitado">{{ txtEventoParametro }}</p>
```

```
public digitandoTexto(event: Event) {  
    this.txtEventoParametro = (<HTMLInputElement>event.target).value;  
    console.log(event);  
}
```

Digite um texto:

Texto digitado:

**Residência**

# Exercício 1

- Crie um novo projeto Angular;
- Crie um novo componente chamado Acesso;
- Limpe todo o conteúdo de app.componente.html e adicione apenas o seletor do novo componente Acesso;
- Em acesso.componente.html crie um campo input e o associe com uma propriedade **permissao** utilizando ngModel e associe um evento **input** a uma função onLogando();
- Além do campo input, crie um button que deve ficar desabilitado;
- Crie uma propriedade do tipo Array string no código ts do componente e a inicialize com os valores root, admin e visitante;
- Habilite o button apenas se o usuário digitar algum nome que está contido no array. Associe um evento click ao button e quando o usuário clicar, troque o texto de um <p> no html contendo bem vindo usuário <exemplo> você está logado.

# Referências

- <https://developer.mozilla.org/en-US/docs/Glossary/SPA>
- <https://angular.io/guide/architecture>
- <https://www.pluralsight.com/guides/understanding-the-purpose-and-use-of-the-selector-in-angular>
- [https://www.w3schools.com/angular/angular\\_databinding.asp](https://www.w3schools.com/angular/angular_databinding.asp)
- <https://docs.npmjs.com/cli/v10/commands/npm-install>
- <https://angular.io/guide/property-binding>
- <https://angular.io/guide/event-binding>
- [https://www.w3schools.com/angular/angular\\_model.asp](https://www.w3schools.com/angular/angular_model.asp)

# Referências

- <https://www.freecodecamp.org/news/angular-directives-learn-how-to-use-or-create-custom-directives-in-angular-c9b133c24442/>