***Curso Angular CLI***

**1 – Criação de componentes**

Ng g c ciclo

c – de Componente

**2 – Utilização desses componentes um dentro do outro**

import { NgModule } from '@angular/core';

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

import { AppRoutingModule } from './app-routing.module';

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

import { CursoModule } from './curso/curso.module';

import { GavetaNavComponent } from './gaveta-nav/gaveta-nav.component';

import { DataBindingComponent } from './data-binding/data-binding.component';

import { FormsModule } from '@angular/forms';

import { InputPropertyComponent } from './input-property/input-property.component';

import { OutputPropertyComponent } from './output-property/output-property.component';

import { CicloComponent } from './ciclo/ciclo.component';

@NgModule({

  declarations: [

    AppComponent,

    GavetaNavComponent,

    DataBindingComponent,

    InputPropertyComponent,

    OutputPropertyComponent,

    CicloComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    CursoModule,

    FormsModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

**3 – Uso de Proprety Binding**

@Input() valorInicial: number = 10; - Declara dentro do ciclo.componente.ts

<app-ciclo [valorInicial]="valor"></app-ciclo> - Dentro do componente Pai da pagina

4 - Inputs, buttons,

 <button class="btn btn-primary" (click)="decrementarContador()">-</button>

    <input type="text" [value]="valor" readonly>

    <button class="btn btn-primary" (click)="incrementarContador()">+</button>

incrementos, decrementos

export class OutputPropertyComponent implements OnInit {

  @Input() valor: number = 0;

  @Output() mudouValor = new EventEmitter();

  constructor() { }

  ngOnInit(): void {

  }

  incrementarContador() {

    this.valor++;

    this.mudouValor.emit({ novoValor: this.valor });

  }

  decrementarContador() {

    this.valor--;

    this.mudouValor.emit({ novoValor: this.valor });

  }

}

**3 comandos do angular cli executar no terminal – fornece para o desenvolvimento**

1 - ng lint – escaneia o código para ver se está em boa pratica de desenvolvimento

2 – ng test – Realiza teste em todo código fonte criado onde tenha a extensão, .spec.ts e realiza testes para checar se estão tudo em ordem o código fonte criado.

3 – ng e2e – teste de integração teste em and two and

**Gerando Build de produção que vai ser feito o deploy para os usuários poderem usar.**

Comando a executar - terminal

1 – ng build – será criado um diretório dist

**Diretivas – com interpolação**

* **Diretivas Estruturais – interagem com a view e modificam a estrutura do DOM ou código do HTML**

**- ngFor**

**- ngIf – condicional para adicionar no HTML**

* **Diretivas de Atributos – Interagem com o elemento em que foram aplicadas**

**- ng-class**

**- ng-style**

**Diretivas: ngSwitch, ngSwitchCase e ngSwitchDefault**

<nav class="navbar navbar-dark bg-primary">

    <div class="nav navbar-nav">

      <a class="nav-item nav-link"

         [class.active]="aba == 'home'"

         (click)="aba = 'home'">Home</a>

      <a class="nav-item nav-link"

        [class.active]="aba == 'mapa'"

        (click)="aba = 'mapa'">Mapa</a>

      <a class="nav-item nav-link"

      [class.active]="aba == 'lista'"

      (click)="aba = 'lista'">Lista</a>

    </div>

  </nav>

  <div class="container" [ngSwitch]="aba"  >

    <p \*ngSwitchCase="'mapa'">Modo Mapa ativado</p>

    <p \*ngSwitchCase="'lista'">Modo Lista ativado</p>

    <p \*ngSwitchDefault>Home</p>

  </div>

  <h5>Removendo o \* e usando template</h5>

  <div class="container" [ngSwitch]="aba"  >

    <ng-template [ngSwitchCase]="'mapa'">

      <p>Modo Mapa ativado</p>

    </ng-template>

    <ng-template [ngSwitchCase]="'lista'">

      <p>Modo Lista ativado</p>

    </ng-template>

    <ng-template [ngSwitchCase]="'home'" ngSwitchDefault>

      <p>Home</p>

    </ng-template>

  </div>

**Directive: ngFor**

**Directive: Sobre o templates**

**Utilizando ngIf no html**

**Directive: ngClass**

**Directive: ngStyle**

**Operador Elvis**

Diretivas

Introdução e tipos de diretivas no Angular

Diretivas: ngIf

Diretivas: ngSwitch, ngSwitchCase e ngSwitchDefault

Diretivas: ngFor

Diretivas: sobre o \* e template

Diretivas: ngClass

Diretivas: ngStyle

Operador Elvis

ng-content

Criando uma diretiva de atributo: ElementRef e Renderer

Diretivas: HostListener e HostBinding

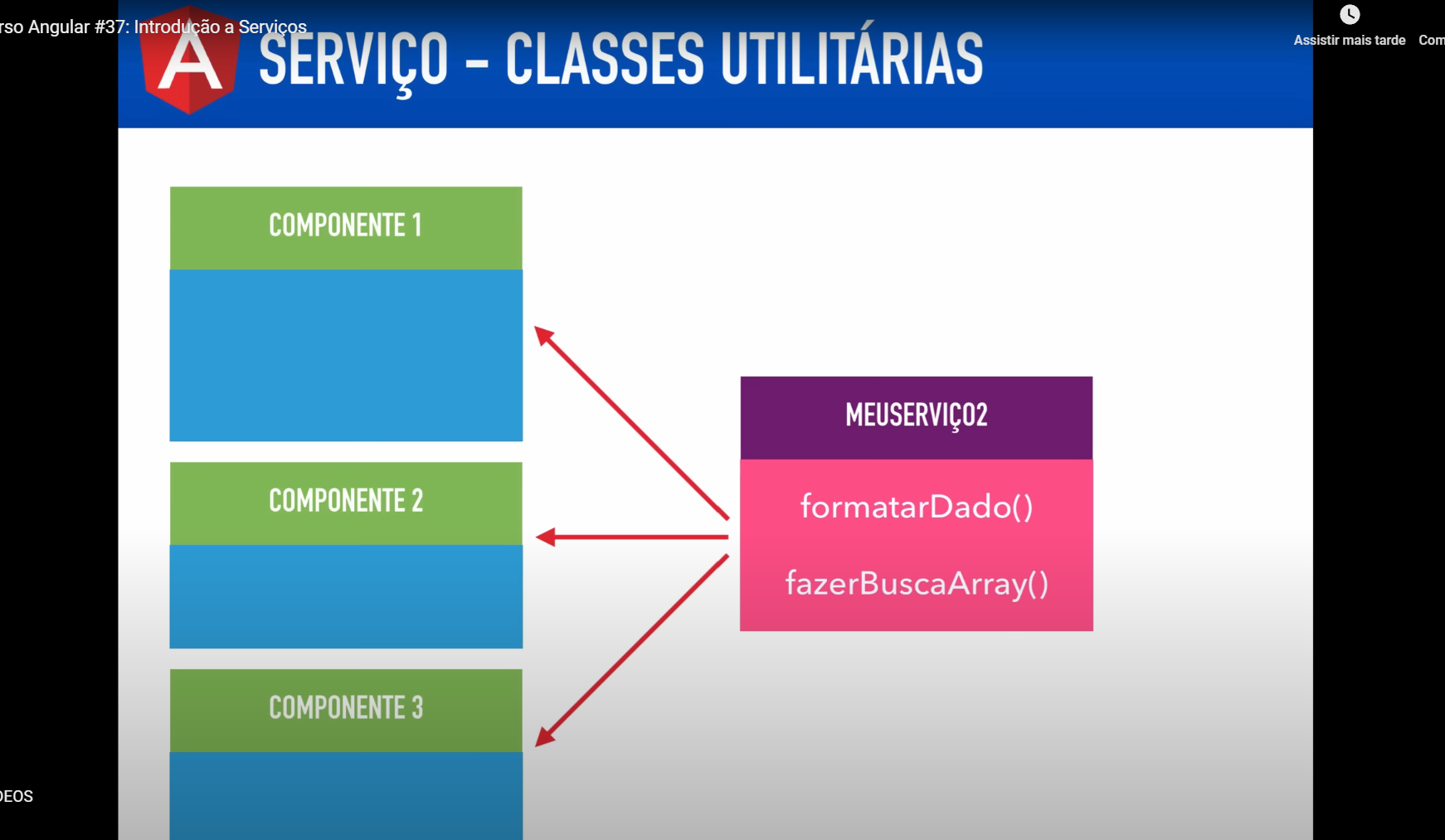
Diretivas: Input e Property Binding

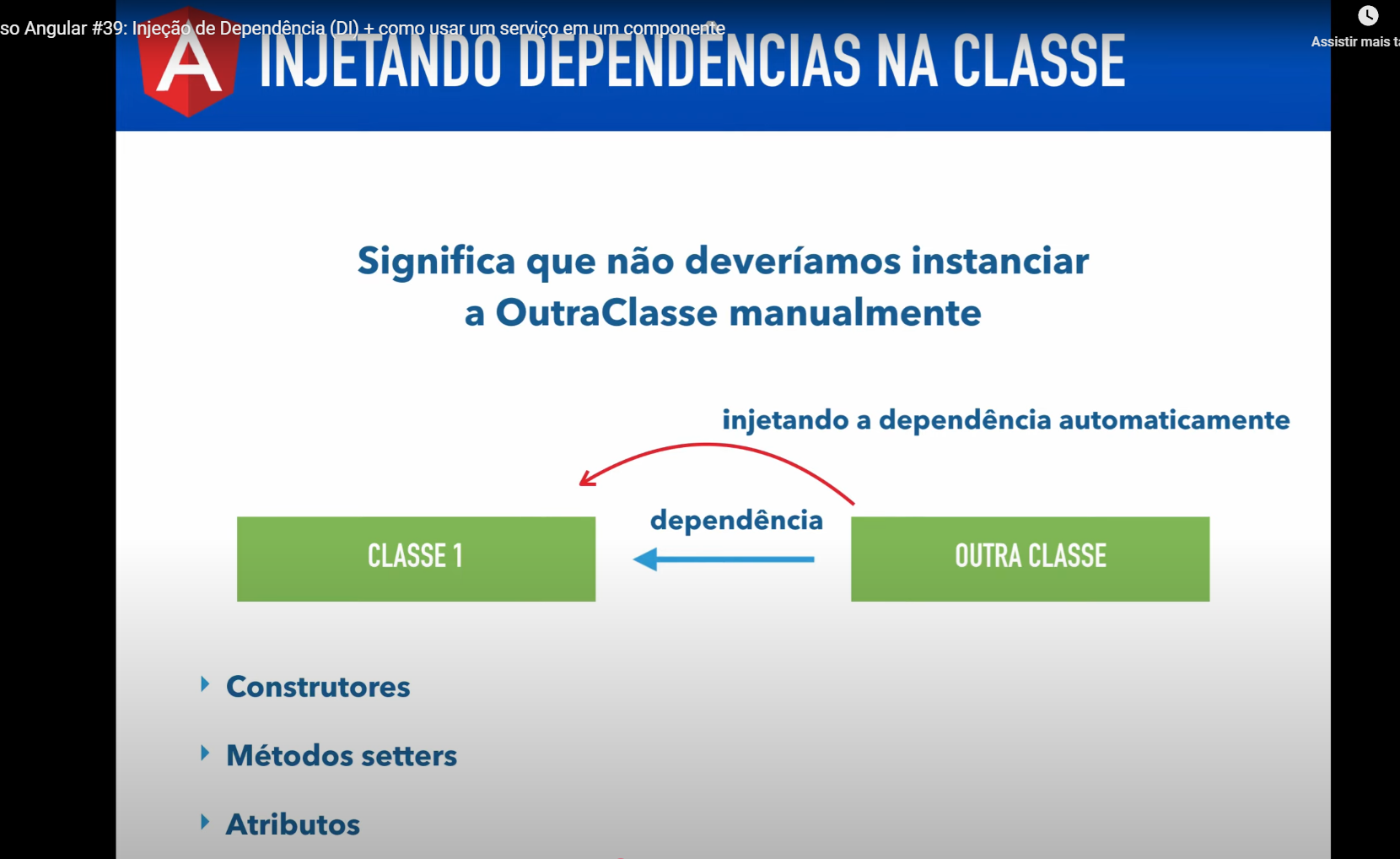
Criando uma diretiva de estrutura (ngElse)

**Serviços**

Classes responsáveis por buscar dados do servidor e enviar dados do servidor

Não e preciso colocar logica nessa classe.





**Rotas**

Rotas: Introdução

Rotas: Configurando rotas simples

Rotas: RouterLink: definindo rotas no template

Rotas: Aplicando CSS em rotas ativas

Rotas: Definindo e extraindo parâmetros de roteamento

Rotas: Escutando mudanças nos parâmetros de roteamento

Rotas Imperativas: Redirecionamento via código

Rotas: Definindo e extraindo parâmetros de url (query)

Rotas: Criando um módulo de rotas

Criando um módulo de funcionalidade

Rotas: Criando um módulo de rotas de funcionalidade

Rotas Filhas

Rotas Filhas: desenvolvendo as telas

Rotas: Dica de Performance: Carregamento sob demanda (lazy loading)

Rotas: Tela de Login e como não mostrar o Menu (NavBar)

Usando Guarda de Rotas: CanActivate

Usando Guarda de Rotas: CanActivateChild

Usando Guarda de Rotas: CanDeactivate

Usando Guarda de Rotas: CanDeactivate com Interface Genérica

Usando Guarda de Rotas: Resolve: carregando dados antes da rota ser ativada

Usando Guarda de Rotas: CanLoad: como não carregar o módulo sem permissão

Definindo rota padrão e wildcard (rota não encontrada)

Estilo de url: HTML5 ou usando #