Como Funciona o `ElementRef` em Serviços no Angular 🚀

O `ElementRef` no Angular é utilizado para acessar e manipular diretamente elementos do DOM. Embora seja mais comum vê-lo dentro de **componentes** e **diretivas**, também é possível utilizá-lo em **serviços** para modificar a estrutura HTML de um elemento passado como parâmetro.

📌 Como Usar `ElementRef` em Serviços

Normalmente, `ElementRef` não pode ser injetado diretamente dentro de um **service** porque os serviços no Angular não estão associados diretamente a elementos do DOM. No entanto, podemos **passar uma referência a um elemento HTML** como argumento para um método dentro do serviço e manipulá-lo.

📌 Exemplo 1: Criando um Elemento Dinamicamente no Serviço

1️⃣ Criando o Serviço: `teste.service.ts`

```
import { ElementRef, Injectable } from "@angular/core";

@Injectable({
  providedIn: 'root', // 🔹 Disponibiliza o serviço globalmente na aplicação
})
export class TesteService {

  // 🔹 Método que recebe um ElementRef e adiciona um novo <div> ao elemento referenciado
  createElement(elRef: ElementRef) {
    const novaDiv = document.createElement('div'); // 🔹 Criamos um novo elemento <div>

    novaDiv.textContent = 'Sou a nova div'; // 🔹 Definimos o texto dentro da <div>
```

```
    novaDiv.classList.add('bg-red'); // 🔹 Adicionamos uma classe CSS para estilização

    elRef.nativeElement.appendChild(novaDiv); // 🔹 Adicionamos essa nova <div> ao elemento passado como referência
  }
}
```

2️⃣ Criando um Componente que Usa o Serviço: `app.component.ts`

```
import { Component, ElementRef } from '@angular/core';
import { TesteService } from './services/teste.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  constructor(
    private readonly _elRef: ElementRef, // 🔹 Capturamos o elemento raiz do componente
    private readonly _testeService: TesteService // 🔹 Injetamos o serviço
  ) {}

  // 🔹 Método para criar um novo elemento chamando o serviço
  createElement() {
    this._testeService.createElement(this._elRef);
  }
}
```

3️⃣ Template (HTML): `app.component.html`

```
<!-- 🔹 O botão chama `createElement()`, que adiciona uma <div> vermelha -->
<button (click)="createElement()">Criar Elemento</button>
```

4️⃣ Estilos CSS: `app.component.scss`

```
.bg-red {
```

```
    background-color: red;
    color: white;
    padding: 10px;
    margin-top: 10px;
    border-radius: 5px;
}
```

📌 📌 Como Funciona?

- O **serviço** recebe um `ElementRef` e cria uma `<div>`.
- O **componente** passa o `ElementRef` do próprio elemento raiz (`this._elRef`).
- O **HTML** contém um botão para acionar a criação da `<div>`.
- O **CSS** estiliza a nova `<div>`.

🔹 **Resultado:** Toda vez que o botão for pressionado, uma nova `<div>` vermelha aparecerá.

📌 Exemplo 2: Alterando o Estilo de um Elemento via Serviço

Podemos usar `ElementRef` em um serviço para modificar diretamente as propriedades CSS de um elemento.

1️⃣ Criando o Serviço: `style.service.ts`

```
import { ElementRef, Injectable } from "@angular/core";

@Injectable({
    providedIn: 'root',
})
export class StyleService {

    // 🔹 Método que altera o fundo de um elemento passado como parâmetro
    changeBackground(elRef: ElementRef, color: string) {
        elRef.nativeElement.style.backgroundColor = color;
    }
}
```

2️⃣ Criando um Componente que Usa o Serviço: `app.component.ts`

```
import { Component, ElementRef, ViewChild } from '@angular/core';
import { StyleService } from './services/style.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  @ViewChild('box') box!: ElementRef; // 🔹 Capturamos o elemento <div>
com #box

  constructor(private _styleService: StyleService) {}

  changeColor() {
    this._styleService.changeBackground(this.box, 'blue'); // 🔹 Muda a cor de
fundo para azul
  }
}
```

3️⃣ Template (HTML): `app.component.html`

```
<!-- 🔹 Elemento alvo que terá seu fundo alterado -->
<div #box class="box">Clique no botão para mudar minha cor!</div>

<!-- 🔹 Botão que chama `changeColor()` e muda o fundo do <div> -->
<button (click)="changeColor()">Mudar Cor</button>
```

4️⃣ Estilos CSS: `app.component.scss`

```
.box {
    width: 200px;
    height: 100px;
    background-color: lightgray;
    text-align: center;
    display: flex;
    align-items: center;
    justify-content: center;
    margin-top: 10px;
```

}

📌 Como Funciona?

1. O serviço `StyleService` recebe um `ElementRef` e um valor de cor.
2. Ele altera a propriedade `backgroundColor` do elemento.
3. O componente usa `@ViewChild` para capturar a `<div>` do template.
4. Quando o botão é clicado, a cor de fundo muda para azul.

🔹 **Resultado:** Ao clicar no botão, a `<div>` muda de cor dinamicamente.

📌 O Que Mais Podemos Fazer com `ElementRef` em Serviços?

✅ **Adicionar elementos dinamicamente ao DOM**
✅ **Alterar propriedades CSS de elementos**
✅ **Modificar o conteúdo de um elemento**
✅ **Aplicar ou remover classes CSS**
✅ **Criar animações simples alterando estilos**

📌 Conclusão

- **`ElementRef`** é útil para manipular o DOM diretamente dentro de serviços.
- No entanto, **não podemos injetá-lo diretamente** no serviço, então passamos ele como argumento para os métodos do serviço.
- Podemos usá-lo para **criar elementos, alterar estilos e modificar conteúdo HTML**.

🚀 Agora você pode utilizar `ElementRef` dentro de serviços para criar e modificar elementos dinamicamente no Angular! 🚀