

## Como Criar um Elemento HTML e Adicioná-lo ao Template do Componente no Angular

No Angular, podemos criar elementos HTML dinamicamente e adicioná-los ao template do componente utilizando **ElementRef**, **Renderer2** ou até mesmo o **ViewContainerRef** para manipulação dinâmica. Vamos explorar diferentes maneiras de fazer isso com exemplos.

---

### 1 Criando Elementos com ElementRef

O **ElementRef** permite acessar diretamente elementos do DOM no Angular.

#### ♦ Exemplo Simples com ElementRef

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

@Component({
  selector: 'app-root',
  template: `
    <button (click)="createElement()">Criar Elemento</button>
    <div #container></div> <!-- O local onde vamos inserir o novo elemento -->
  `,
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  @ViewChild('container', { static: true }) container!: ElementRef;

  createElement() {
    // Criamos um novo elemento <div>
    const novaDiv = document.createElement('div');

    // Adicionamos um texto à nova <div>
    novaDiv.textContent = 'Sou uma nova div!';

    // Adicionamos uma classe CSS à nova <div>
    novaDiv.classList.add('nova-classe');

    // Adicionamos a nova <div> dentro do elemento container
```

```
this.container.nativeElement.appendChild(novaDiv);
}
}
```

### ✓ O que acontece nesse exemplo?

- Criamos dinamicamente um novo elemento `<div>`.
  - Adicionamos um **texto** e uma **classe CSS** a ele.
  - Inserimos essa nova `<div>` dentro do `#container`, um elemento do nosso template.
- 

## 2 Criando Elementos com **Renderer2** (Forma Segura)

O **Renderer2** é a abordagem recomendada pelo Angular, pois evita manipulações diretas do DOM e melhora a segurança e compatibilidade com diferentes plataformas (ex: SSR, Web Workers).

### ♦ Exemplo com **Renderer2**

```
import { Component, ElementRef, Renderer2, ViewChild } from
'@angular/core';
```

```
@Component({
  selector: 'app-root',
  template: `
    <button (click)="createElement()">Criar Elemento</button>
    <div #container></div>
  `,
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  @ViewChild('container', { static: true }) container!: ElementRef;

  constructor(private renderer: Renderer2) {}

  createElement() {
    // Criamos uma nova div usando o Renderer2
```

```
const novaDiv = this.renderer.createElement('div');

// Criamos um nó de texto e adicionamos à div
const texto = this.renderer.createText('Sou uma nova div criada pelo
Renderer2!');
this.renderer.appendChild(novaDiv, texto);

// Adicionamos uma classe CSS ao elemento
this.renderer.addClass(novaDiv, 'nova-classe');

// Inserimos a nova div dentro do container
this.renderer.appendChild(this.container.nativeElement, novaDiv);
}
}
```

### ✓ Por que usar **Renderer2**?

- Evita manipulação direta do DOM.
  - Melhor compatibilidade com SSR e outras plataformas Angular.
  - Melhora a segurança contra ataques XSS.
- 

## 3 Criando Elementos com **ViewContainerRef** e **ComponentFactoryResolver**

Podemos criar componentes inteiros dinamicamente em vez de apenas elementos HTML.

### ◆ Criando um Componente Dinamicamente

Vamos supor que temos um componente chamado **DynamicComponent**:

📌 **dynamic.component.ts**

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

```
@Component({
  selector: 'app-dynamic',
  template: `<p class="dynamic">Componente Criado Dinamicamente!</p>`,
```

```

    styles: ['.dynamic { background: lightblue; padding: 10px; border-radius: 5px;
  }]
})
export class DynamicComponent {}

```

### Criando o Componente Dinamicamente no **app.component.ts**

```

import { Component, ComponentFactoryResolver, ViewChild,
ViewContainerRef } from '@angular/core';
import { DynamicComponent } from './dynamic.component';

@Component({
  selector: 'app-root',
  template: `
    <button (click)="createDynamicComponent()">Criar Componente</button>
    <ng-container #container></ng-container> <!-- Local onde o componente
será inserido -->
  `
})
export class AppComponent {
  @ViewChild('container', { read: ViewContainerRef }) container!:
ViewContainerRef;

  constructor(private componentFactoryResolver: ComponentFactoryResolver)
  {}

  createDynamicComponent() {
    // Criamos uma fábrica do componente dinâmico
    const componentFactory =
this.componentFactoryResolver.resolveComponentFactory(DynamicCompone
nt);

    // Limpamos o conteúdo anterior do container
    this.container.clear();

    // Criamos o componente dinamicamente dentro do container
    this.container.createComponent(componentFactory);
  }
}

```

### ✓ O que acontece nesse exemplo?

- Criamos um **novo componente** (`DynamicComponent`) **dinamicamente**.
  - O botão **adiciona esse componente** no template em tempo de execução.
  - Podemos criar quantos componentes quisermos **sem precisar declará-los no HTML**.
- 

## 4 Criando Elementos e Manipulando Eventos Dinamicamente

Podemos criar eventos dinamicamente em elementos gerados.

### ◆ Adicionando Eventos a um Elemento Criado

```
createElement() {  
  const novaDiv = this.renderer.createElement('div');  
  const texto = this.renderer.createText('Clique em mim!');  
  
  this.renderer.appendChild(novaDiv, texto);  
  this.renderer.addClass(novaDiv, 'clicavel');  
  
  // Adicionando um evento de clique ao elemento criado dinamicamente  
  this.renderer.listen(novaDiv, 'click', () => {  
    alert("Você clicou na nova div!");  
  });  
  
  this.renderer.appendChild(this.container.nativeElement, novaDiv);  
}
```

### ✓ O que acontece aqui?

- Criamos um `<div>` dinamicamente.
  - Adicionamos um **evento de clique** a essa `<div>`.
  - Quando clicamos nela, um **alerta** aparece.
-

## ◆ Comparação Entre Métodos


Método	Vantagens	Desvantagens
<code>ElementRef</code>	Simples e fácil de usar	Manipula diretamente o DOM (menos seguro)
<code>Renderer2</code>	Seguro e compatível com SSR	Sintaxe um pouco mais extensa
<code>ViewContainerRef</code>	Permite criar componentes inteiros dinamicamente	Mais complexo

---

## Conclusão

Agora você sabe **como criar elementos HTML dinamicamente** no Angular e quais são as **melhores práticas** para isso. Cada método tem seu uso específico:

- ✓ **Se quiser criar elementos simples:** Use `ElementRef`.
- ✓ **Se quiser criar elementos de forma segura:** Use `Renderer2`.
- ✓ **Se quiser criar componentes inteiros dinamicamente:** Use `ViewContainerRef`.

 **Agora é sua vez!** Teste esses métodos no seu projeto e veja qual se encaixa melhor!