**Angular ng-container**

The **ng-container**is an [Angular](https://www.geeksforgeeks.org/angular-tutorial/) element that acts as a grouping mechanism. It does not render any additional HTML in the DOM which makes it great for scenarios where you need to apply structure directives like **\*ngIf, \*ngFor,**or **\*ngSwitch**but do not want to create additional elements like **div or span**that could disrupt your layout or styles. In this article, we explain about what is ng-container and examples with related outputs in different approaches for your reference.

**Why use ng-container?**

There are some reasons why you should use ng-container:

* **Keeps your DOM clean**: ng-container helps keep your HTML clean by not adding extra elements to the page.
* **Combine Multiple Directives:**ng-container allows you to use Angular directives together without adding extra elements to your HTML.
* **Improve Performance:** ng-container helps keep things efficient making your page run a little smoother.

**Using ng-container with \*ngIf**

This approach allows us to conditionally include or exclude a section of the DOM without introducing extra HTML elements.

**Syntax**

<ng-container \*ngIf="isLoggedIn">  
 <p>Welcome, user!</p>  
</ng-container>

This is particularly useful when you want to include or exclude multiple elements based on a single condition without wrapping them in a parent element.

**Using ng-container with \*ngFor**

This approach lets you iterate over a list of items and render them in the DOM without adding extra elements.

**Syntax**

<ng-container \*ngFor="let user of users">  
 <p>{{ user.name }}</p>  
 <p>{{ user.email }}</p>  
</ng-container>

This is ideal for scenarios where you need to render multiple elements for each item in a list but don't want to add extra wrapper around those elements.

**Using ng-container with ngSwitch**

This approach allows you to conditionally render different templates based ona switch expression.

**Syntax**

<ng-container [ngSwitch]="user.role">  
 <p \*ngSwitchCase="'admin'">Admin Dashboard</p>  
 <p \*ngSwitchCase="'user'">User Dashboard</p>  
 <p \*ngSwitchDefault>Unknown role</p>  
</ng-container>

This is useful when you have multiple conditions and want to selectively render different parts of the template based on the evaluated expression without adding unnecessary parent elements.

**Using ng-container with ngIf and ngFor Combination**

This approach demonstrates how to combine ngIf and ngFor.

**Syntax:**

<ul>  
 <ng-container \*ngFor="let user of users">  
 <ng-container \*ngIf="user.isActive">  
 <li>{{ user.name }} - {{ user.email }}</li>  
 </ng-container>  
 </ng-container>  
</ul>