ng-container in Angular

5 Comments / 2 minutes of reading / March 9, 2023

← Template Reference Variable

Angular Tutorial

ngTemplate →

In this guide let us look at `ng-container` in Angular. We use this to create a dummy section in the template, without rendering it in the HTML. This is a pretty useful feature while we work with the structural directives like ngIf, ngFor & ngSwitch. We also look at some of the use cases of the same.

Table of Contents

What is ng-container ?

ng-container allows us to create a division or section in a template without introducing a new HTML element. The ng-container does not render in the DOM, but

content inside it is rendered. ng-container is not a directive, component, class, or interface, but just a syntax element.

For Example

The following template

```
<h1> ng-Container</h2>
<p>Hello  world! </p>
<ng-container>            //This is removed from the final HTML
 Container's content.
</ng-container>
```

Renders as

```
<h1> ng-Container</h2>
<p>Hello  world! </p>
Container's content.
```

You can see that the element is absent in the final HTML

Uses of ng-container

It is a very useful directive. Especially when working with structural directives like ngIf, ngFor, etc.

ng-Container with ngFor

For Example, consider the following items. We want to display the items as a list, but only the active items. This requires two directives ngFor to loop through the items and ngIf to check if the items are `active`

```
items= [
  { name:'Angular', active:true},
  { name:'React', active:true},
  { name:'Typescript', active:true},
  { name:'FoxPro', active:false},
  { name:'Javascript', active:true},
  { name:'ASP.NET Core', active:true},
  { name:'DBase', active:false}
]
```

Without `ng-container`, the only way to achieve this is by using the `span` element as shown. This adds the unnecessary DOM element. and it may also cause issues with the CSS.

```
<ul>
 <span *ngFor="let item of items;">
   <li *ngIf="item.active">
     {{item.name}}
   </li>
 </span>
</ul>
```

By Replacing the span with ng-container our HTML renders correctly without those extra span elements

```
<ul>
 <ng-container *ngFor="let item of items;">
   <li *ngIf="item.active">
     {{item.name}}
   </li>
 </ng-container>
</ul>
```

ng-container examples

ng-container with ngIf

The div of the ngIf is not necessary here.

```
<div *ngIf="items1">        //Replace the div with ng-container as shown below
 <div *ngFor="let item of items1;">
   {{item.name}}
 </div>
</div>
```

```
```

```
<ng-container *ngIf="items1">
 <div *ngFor="let item of items1;">
  {{item.name}}
 </div>
</ng-container>

```

ngSwitch with/without ng-container

```

<div [ngSwitch]="value">
 <span *ngSwitchCase="0">Text one</span>
 <span *ngSwitchCase="1">Text two</span>
</div>

<div [ngSwitch]="value">
 <ng-container *ngSwitchCase="0">Text one</ng-container>
 <ng-container *ngSwitchCase="1">Text two</ng-container>
</div>


```

ngTemplateOutlet

The container is also used as a placeholder for injecting a dynamic template using the ngTemplateOutlet .

```

<ng-container *ngTemplateOutlet="loading"></ng-container>

```

References

- Structural Directives

Read More

1. Angular Directives