*ngIf and *ngFor on same element causing error



I'm having a problem with trying to use Angular's *ngFor and *ngIf on the same element.

371

When trying to loop through the collection in the *ngFor, the collection is seen as null and consequently fails when trying to access its properties in the template.



```
★ 61
```

```
@Component({
  selector: 'shell',
  template: `
    <h3>Shell</h3><button (click)="toggle()">Toggle!</button>
    <div *ngIf="show" *ngFor="let thing of stuff">
      {{log(thing)}}
      <span>{{thing.name}}</span>
    </div>
})
export class ShellComponent implements OnInit {
 public stuff:any[] = [];
 public show:boolean = false;
  constructor() {}
  ngOnInit() {
   this.stuff = [
      { name: 'abc', id: 1 },
      { name: 'huo', id: 2 },
      { name: 'bar', id: 3 },
      { name: 'foo', id: 4 },
      { name: 'thing', id: 5 },
      { name: 'other', id: 6 },
```

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email



Sign up with Facebook



toggle() {

I know the easy solution is to move the *ngIf up a level but for scenarios like looping over list items in a ul , I'd end up with either an empty 1i if the collection is empty, or my 1i s wrapped in redundant container elements.

Example at this plnkr.

Note the console error:

```
EXCEPTION: TypeError: Cannot read property 'name' of null in [{{thing.name}} in
ShellComponent@5:12]
```

Am I doing something wrong or is this a bug?



edited Feb 1 at 11:07

asked Jan 7 '16 at 14:37



4,787 11 40 71

stackoverflow.com/questions/40529537/... i'd go with ng-container - robert king Dec 21 '17 at 22:29

Possible duplicate of Angular filtered table – Cobus Kruger Aug 29 '18 at 13:41

14 Answers



570

Angular v2 doesn't support more than one structural directive on the same element.

As a workaround use the <ng-container> element that allows you to use separate elements for each structural directive, but it is not stamped to the DOM.

Join Stack Overflow to learn, share knowledge, and build your career.





```
</div>
</ng-container>
```

<ng-template> (<template> before Angular v4) allows to do the same but with a different syntax which is confusing and no longer
recommended

```
<ng-template [ngIf]="show">
    <div *ngFor="let thing of stuff">
        {{log(thing)}}
        <span>{{thing.name}}</span>
        </div>
</ng-template>
```

edited Aug 20 '18 at 13:05



Edric 9,133 7 39

answered Sep 19 '16 at 5:27



Günter Zöchbauer 356k 82 1120 1027

- 5 Thanks a lot. Surprisingly is still undocumented: github.com/angular/angular.io/issues/2303 Alex Fuentes Jan 20 '17 at 12:38
- How will code look like when we have to have *nglf inside *ngFor ? I.e. IF condition will be based on value of a loop element. Yuvraj Patil Jan 24 '17 at 12:24 🖍
- 14 Just put ngFor at the <ng-container> element and the ngIf at the <div> . You can also have two nested <ng-container> wrapping the <div> . <ng-container> is just a helper element that will not be added to the DOM. Günter Zöchbauer Jan 24 '17 at 12:28 /
- 3 I'd suggest using <ng-container> . It behaves the same as <template> but allows to use the "normal" syntax for structural directives. Günter Zöchbauer May 9 '17 at 18:32
- 2 <u>Documentation says</u>: "One structural directive per host element": "There's an easy solution for this use case: put the *nglf on a container element that wraps the *ngFor element." just reiterating heringer Feb 21 '18 at 20:02



As everyone pointed out even though having multiple template directives in a single element works in angular 1.x it is not allowed in Angular 2. you can find more info from here: https://github.com/angular/angular/angular/issues/7315

50

2016 angular 2 beta

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email



but for some reason above does not work in 2.0.0-rc.4 in that case you can use this

Updated Answer 2018

With updates, right now in 2018 angular v6 recommend to use <ng-container> instead of <template>

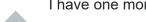
so here is the updated answer.

edited Sep 17 '18 at 8:53

answered Jul 14 '16 at 16:22



imal hasaranga perera 5,142 2 34 28



I have one more solution.

Use [hidden] instead of *ngIf

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email



The difference is that *ngIf will remove the element from the DOM, while [hidden] actually plays with the css style by setting display:none

answered Feb 9 '18 at 10:37



Ajmal sha

1,711 1 17 33



As @Zyzle mentioned, and @Günter mentioned in a comment (https://github.com/angular/angular/issues/7315), this is not supported.

29

With



```
         {{log(thing)}}
         <span>{{thing.name}}</span>
```

there are no empty elements when the list is empty. Even the element does not exist (as expected).

When the list is populated, there are no redundant container elements.

The <u>github discussion (4792)</u> that @Zyzle mentioned in his comment also presents another solution using <template> (below I'm using your original markup - using <div> s):

This solution also does not introduce any extra/redundant container elements.

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email





1 I'm not sure why this isn't the accepted answer. <template> is the way to add a parent element that won't show up in the output. — Evan Plaice Jan 25 '16 at 19:28



You can't have ngFor and ngIf on the same element. What you could do is hold off on populating the array you're using in ngFor until the toggle in your example is clicked.

5

Here's a basic (not great) way you could do it: http://plnkr.co/edit/Pylx5HSWIZ7ahoC7wT6P

answered Jan 7 '16 at 14:48



Zyzle

2 13 17

Why he cant have both? Elaborate please – maurycy Jan 7 '16 at 14:53

- 1 There's a discussion around that here github.com/angular/angular/issues/4792 Zyzle Jan 7 '16 at 14:55
- 1 I know why that's happening, it's just to improve quality of the answer, plainly saying you can't is not really a good answer, wont you agree? maurycy Jan 7 '16 at 15:14

Sure, they shouldn't be used together just because putting them in certain order to template doesn't guarantee that they will be executed in the same order. But this does not explain what exactly happens when 'Cannot read property 'name' of null' is thrown. – Estus Flask Jan 7 '16 at 15:46

Both *ngFor and *ngIf (with asterisk) are structural directives and they generate <template> tag. Structural directives, like ngIf, do their magic by using the HTML 5 template tag. – Pardeep Jain Apr 4 '16 at 7:17



This will work but the element will still in the DOM.



```
.hidden{
    display: none;
}
```



Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email





answered Aug 15 '16 at 17:25



This is a very easy hack for <select> <option> combination, which I simply want to show filtered items instead of the full list – davyzhang Feb 17 '17 at 16:27 /



Table below only lists items that have a "beginner" value set. Requires both the *ngFor and the *ngIf to prevent unwanted rows in html.

3

Originally had *ngIf and *ngFor on the same tag, but doesn't work. Added a <div> for the *ngFor loop and placed *ngIf in the tag, works as expected.

edited Dec 19 '16 at 20:43



Irnzcig 2.804

2,804 4 25

answered Dec 19 '16 at 19:59

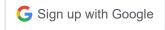


charliebear240

61 1

Join Stack Overflow to learn, share knowledge, and build your career.

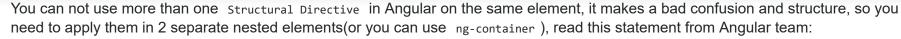
Sign up with email







3





Someday you'll want to repeat a block of HTML but only when a particular condition is true. You'll try to put both an *ngFor and an *ngIf on the same host element. Angular won't let you. You may apply only one structural directive to an element.

The reason is simplicity. Structural directives can do complex things with the host element and its descendents. When two directives lay claim to the same host element, which one takes precedence? Which should go first, the NgIf or the NgFor? Can the NgIf cancel the effect of the NgFor? If so (and it seems like it should be so), how should Angular generalize the ability to cancel for other structural directives?

There are no easy answers to these questions. Prohibiting multiple structural directives makes them moot. There's an easy solution for this use case: put the *nglf on a container element that wraps the *ngFor element. One or both elements can be an ng-container so you don't have to introduce extra levels of HTML.

So you can use ng-container (Angular4) as the wrapper (will be deleted from the dom) or a div or span if you have class or some other attributes as below:

```
<div class="right" *ngIf="show">
    <div *ngFor="let thing of stuff">
        {{log(thing)}}
        <span>{{thing.name}}</span>
        </div>
</div>
```

answered Jul 2 '17 at 8:26





in html:

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email





in css:

```
.disabled-field {
    pointer-events: none;
    display: none;
}
```

answered Apr 19 at 5:14





Updated to angular2 beta 8

Now as from angular beta 8 we can use *ngIf and *ngFor on same component see here.



Alternate:

Sometimes we can't use HTML tags inside another like in tr, th (table) or in li (ul). We cannot use another HTML tag but we have to perform some action in same situation so we can HTML5 feature tag <template> in this way.

ngFor using template:

```
<template ngFor #abc [ngForOf]="someArray">
    code here....
</template>
```

nglf using template:

```
<template [ngIf]="show">
code here....
</template>
```

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email











```
<div *ngFor="let thing of show ? stuff : []">
    {{log(thing)}}
    <span>{{thing.name}}</span>
</div>
```

answered Jan 16 '18 at 10:13





You can also use ng-template (instead of template. See the note for the caveat of using template tag) for applying both *ngFor and nglf on the same HTML element. Here is an example where you can use both *nglf and *ngFor for the same tr element in the angular table.



Note:

The caveat of using just the template tag instead of ng-template tag is that it throws StaticInjectionError in some places.

answered Feb 1 at 10:39



Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email







answered Nov 2 '16 at 10:55



li items are only displayed if it has a name. - Rajiv Nov 2 '16 at 10:56

How does this answer add value here? It doesn't provide anything that's not provided by the other answers already or did I miss something? – Günter Zöchbauer Nov 2 '16 at 11:09



You can't use multiple structural directive on same element. Wrap your element in ng-template and use one structural directive there







Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email



