What is the difference between [ngFor] and [ngForOf] in angular2?

43

4

As per my understanding, *Both are doing the same* functions. But,

- `ngFor` would be works like as collections?
- `ngForOf` would be works like as generics?

Is my understanding is correct? or Could you please share more difference's(details) about `ngFor` and `ngForOf` ?

angular typescript ngfor

edited Jul 19 '17 at 9:30

asked Apr 13 '17 at 9:47

Ramesh Rajendran
20.1k 22 93 165

1 Good question! i also wanted to ask! btw, your links for both collections and generics are same – Sajeetharan Apr 13 '17 at 9:58

@Sajeetharan Typo issue. Sorry for that. Now I have updated my question. thanks – Ramesh Rajendran Apr 13 '17 at 10:03

4 github.com/angular/angular/commit/… – yurzui Apr 13 '17 at 10:12

@yurzui, good catch. So it seems as though the class was previously NgFor and it got changed in the move to angular v4, and the key thing is that we as a community were none the wiser. Awesome backwards compatibility – snorkpete Apr 13 '17 at 10:18

5 Answers

57

`ngFor` and `ngForOf` are not two distinct things - they are actually the selectors of the NgForOf directive.

If you examine the source, you'll see that the NgForOf directive has as its selector: `[ngFor][ngForOf]` , meaning that both attributes need to be present on an element for the directive to 'activate' so to speak.

+100

So,

```
<div *ngFor="let item of items"></div>
```

desugars to:

```
<template [ngFor]="let item of items">
    <div></div>
</template>
```

This first de-sugaring is due to the '*'. The next de-sugaring is because of the micro syntax: "let item of items". The Angular compiler de-sugars that to:

```
<template ngFor let-item="$implicit" [ngForOf]="items">
  <div>...</div>
</template>
```

(where you can think of $implicit as an internal variable that the directive uses to refer to the current item in the iteration).

In its canonical form, the ngFor attribute is just a marker, while the ngForOf attribute is actually an input to the directive that points to the the list of things you want to iterate over.

You can check out the Angular microsyntax guide to learn more.

edited Jul 19 '17 at 10:04

answered Apr 13 '17 at 10:03

snorkpete

10.4k 3 30 47

1 Isn't `ngFor` the actual directive whereas `ngForOf` is "just" a property of the "de-sugared" version? – devnull69 Apr 13 '17 at 10:05

well, technically, ngForOf is the name of the class that implements the directive (again, see the source). It's just that no one actually refers to it by that name - ngFor is basically how its referred to in all documentation and conversations. – snorkpete Apr 13 '17 at 10:08

1 Your desugaring examples are both quite wrong. I don't think there are 2 steps, and the end result would be `<template ngFor let-item="$implicit" [ngForOf]="items"> </template>` – Günter Zöchbauer Jul 19 '17 at 9:36

1 @GünterZöchbauer, you are right - i put the directives in the wrong place - i updated the code examples. And you're probably also right that the de-sugaring happens in just one step, but i think in this case, it's useful to separate the desugaring of the structural directive from the desugaring of the let

1

In my opinion what I got from the angular document,

- `[ngFor]` is not **`type safe`**
- `[NgForOf]` is **`type safe`**

Because both class details are little different

- `ngFor` Class type is `any` type
- But `ngForOf` class detail is `ngForOf : NgIterable<T>`

`ngForOf` looks like generics which I have already mentioned in my question.

edited Feb 19 '18 at 0:53
WebDevBooster
9,354 7 34 48

answered Apr 13 '17 at 10:24
Ramesh Rajendran
20.1k 22 93 165

2 Interesting. However, since they are used from within templates, this matters less in practice than it otherwise would. That said, the number of Angular APIs, including ones not intended for template use such as `FormBuilder` , which are not type checked is troubling. – Aluan Haddad Apr 24 '17 at 0:12

1

NgFor can iterate an array but in ngContainer, ngContent we cannot iterate ngFor directly so For iterating we can use [ngForOf] . Here i is variable, ratings is an array.

Ex.

```
<ng-template ngFor let-i [ngForOf]="ratings">
                <option *ngIf="i<=22" [ngValue]="i">{{i}}:00 Hrs.</option>
  </ng-template>
```

At this example, I want to apply a ngFor and ngIf simultaneously, so i used it.

The other view is that at a normal if we apply ngFor then at rendering it converts it into

```
<template ngFor let-i [ngForOf]="ratings">
   <HTML Element>...</HTML Element>
 </template>
```

edited Feb 16 '18 at 7:09

answered Feb 15 '18 at 7:30

rajeev omar

124 1 6

0

Explicit version

- `(<template ngFor ...>)` allows to apply the directive to multiple elements at once

Implicit version

- only wraps the element where it is applied

answered Apr 26 '17 at 0:21

user7427848

5

`ngFor` is a structural directive of Angular which replaces the `ng-repeat` attribute of AngularJS

You can use `ngFor` as a shorthand

```
<li *ngFor="let item of items">{{item.name}}</li>
```

or as the longhand version

```
<template ngFor let-item="$implicit" [ngForOf]="items">
  {{item.name}}
</template>
```

answered Apr 13 '17 at 10:06

`shorthand` and `longhand` means no of data size? or what? – Ramesh Rajendran Apr 13 '17 at 10:08

It means: Difference of usage for the same purpose and effect – devnull69 Apr 13 '17 at 10:09

Okay now I understand it. But whenever am seeing this line in the document `ngForOf: NgIterable<T>: The value of the iterable expression. Useful when the expression is more complex then a property access, for example when using the async pipe (userStreams | async).` really not understandable. – Ramesh Rajendran Apr 13 '17 at 10:12

3 Let's say your `items` collection is not a simple collection but rather a collection which is not yet available and will only be available after some asynchronous call. So you won't have `items` yet but only an Observable or Promise (let's call it futureItems). In that case you'd like to use the async pipe `let item of (futureItems|async)` which is not possible. But you can use the longhand version with `[ngForOf]="futureItems | async"` – devnull69 Apr 13 '17 at 10:29

protected by Ramesh Rajendran Apr 2 '18 at 8:37

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?