Why array functions like 'filter' and 'map' doesn't work in 'ngFor' expression of Angular?

Asked 2 years, 3 months ago Active 2 years, 2 months ago Viewed 601 times

2

While working with angular template I have found that array functions like `slice()` works absolutely fine with `ngFor` directive expression like follow

```
<div *ngFor="let item of arry.slice(3)">
  {{ item.name }}
</div>
```

where as when I tried to use array functions like `filter()` or `map()` in `ngFor` expression like follow

```
<div *ngFor="let item of arry.filter(i => i.marks > 40)">
  {{ item.name }}
</div>
```

it gives me angular template parse error

```
 Zone: <root> ; Task: Promise.then ; Value: Error: Template parse errors:
Parser Error: Bindings cannot contain assignments at column 25 in [let item of arry.filter(i => i.marks > 40)]
```

In order to understand reasons behind such behaviour I was trying to find out some document explaining about such limitations but have found none. Any lead or explanation will be appreciated.

javascript arrays angular ngfor

edited Aug 3 '17 at 18:06

asked Jun 20 '17 at 18:32
Ajax
752 9 25

Yes, it's a templating error. It's strictly not allowed in angular. It is not a browser restriction – PierreDuc Jun 20 '17 at 18:37

3 I'd start reading here: angular.io/guide/template-syntax#template-expressions I like to make my templates as simple as possible and do all the business logic in the TypeScript code. – Rob Zuber Jun 20 '17 at 18:38

1 @Skeptor, it can't parse that text and thinks the equals sign is an assignment. – Rob Zuber Jun 20 '17 at 18:42

1 Answer

4 ✓

As mentioned in comments, Angular is quite restrictive about the types of expressions allowed in templates. There are good reasons for this: complex expressions involving function calls, for example, do not play well with change detection. Remember that these expressions are not actually JavaScript/TypeScript; they are merely JavaScript-**like**.

If you really want to do this filtering in the template, then

```
<div *ngFor="let item of arry">
  <ng-container *ngIf="item.marks > 40">
    {{ item.name }}
  </ng-container>
</div>
```

Another alternative is to write a pipe to do the filtering. However, the Angular team recommends against this as well.

In general, more complex logic can and should be written in your TS logic in the component.

By the way, for `slice` , it is recommended to use the slice pipe. Angular will be able to optimize this better than it could `.slice()` .

Somewhat related: Support arrow functions in template syntax.

edited Jun 20 '17 at 19:01 answered Jun 20 '17 at 18:54
user663031

"...Another alternative is to write a pipe to do the filtering. However, the Angular team recommends against this as well..." Does it means that ppl shouldn't use pipes? Also, where can I found it in docs? – developer033 Jun 20 '17 at 19:05

Great resource, didnt knew it was a bad practice to use pipes for those purpose. Thanks for the link – Jota.Toledo Jun 20 '17 at 19:19

What If I make a custom pipe *pure* for filtering? Why does it must be impure? – developer033 Jun 20 '17 at 19:20