What is let-* in Angular 2 templates?

95

23

I came across a strange assignment syntax inside an Angular 2 template.

```
<template let-col let-car="rowData" pTemplate="body">
    <span [style.color]="car[col.field]">{{car[col.field]}}</span>
</template>
```

It appears that `let-col` and `let-car="rowData"` create two new variables `col` and `car` that can then be bound to inside the template.

Source: https://www.primefaces.org/primeng/#/datatable/templating

What is this magical `let-*` syntax called?

How does it work?

What is the difference between `let-something` and `let-something="something else"` ?

angular angular2-template primeng

edited May 17 at 22:19

asked Mar 23 '17 at 13:53
Steven Liekens
6,625 2 39 63

3 @NiekT. this is different, let-* in angular 2 is template variable scoping – Sterling Archer Mar 23 '17 at 13:57

2 angular.io/docs/ts/latest/guide/… search the word "let " (with a space) and go to around the 9th one. There is a good explanation of what this template variable does – Sterling Archer Mar 23 '17 at 13:58

@SterlingArcher Thanks for the correction, I'm quite new to JS and Angular myself. – Nodon Darkeye Mar 23 '17 at 13:59

1 Answer

96

`ngOutletContext` was renamed to `ngTemplateOutletContext`

See also https://github.com/angular/angular/blob/master/CHANGELOG.md#500-beta5-2017-08-29

original

Templates (`<template>` , or `<ng-template>` since 4.x) are added as embedded views and get passed a context.

With `let-col` the context property `$implicit` is made available as `col` within the template for bindings. With `let-foo="bar"` the context property `bar` is made available as `foo` .

For example if you add a template

```
<ng-template #myTemplate let-col let-foo="bar">
  <div>{{col}}</div>
  <div>{{foo}}</div>
</ng-template>

<!-- render above template with a custom context -->
<ng-template [ngTemplateOutlet]="myTemplate"
             [ngTemplateOutletContext]="{
                                           $implicit: 'some col value',
                                           bar: 'some bar value'
                                        }"
></ng-template>
```

See also this answer and ViewContainerRef#createEmbeddedView.

`*ngFor` also works this way. The canonical syntax makes this more obvious

```
<ng-template let-item [ngForOf]="items" let-i="index" let-odd="odd">
  <div>{{item}}
</ng-template>
```

where `NgFor` adds the template as embedded view to the DOM for each `item` of `items` and adds a few values (`item` , `index` , `odd`) to the context.

See also Using $implict to pass multiple parameters

edited Aug 4 '18 at 5:06 answered Mar 23 '17 at 14:07

1 Thanks for explaining `ngOutletContext` . That was the missing link between what I already knew and the information that I couldn't find in the documentation. – Steven Liekens Mar 23 '17 at 14:17

I don't think it is called `ngTemplateOutletContext` as you've suggested in the release of angular 5. The docs also don't mention anything about it being deprecated. angular.io/api/common/NgTemplateOutlet – Jessycormier Jan 5 '18 at 19:49

5 is not yet released. Not sure what the docs show. The changelog doesn't have anything new about it since then. – Günter Zöchbauer Jan 5 '18 at 21:08

1 Thank you for this answer, there is a strong lack of documentation on what the `*` syntax is doing. – dook Feb 8 at 20:06

Shouldn't be the second ng-template (the one with ngTemplateOutlet) really ng-template. Maybe ng-container would be better? Both will work I guess, but the ng-container is semantically more correct. Or am I wrong? – Ondrej Peterka Mar 21 at 17:51