Angular2 access global variables from HTML template

Asked 3 years, 2 months ago Active 1 year, 6 months ago Viewed 21k times

27

1

I have a global variable to store the list of countries like this:

```
export var COUNTRY_CODES = ["AD", "AE", "AF" /* and more */];
```

In one of my component, I'd imported the variable using normal import statement

```
import { COUNTRY_CODES } from "../constants";
```

I am able to access this global variable freely in my component code, but failed to achieve something like this on the HTML template:

```
<option *ngFor="let countryCode of COUNTRY_CODES" [value]="countryCode">{{countryCode |
countryName}}</option>
```

I could just pass along the global variable to component by defining a local variable and assign the global variable to it during initialization.

```
ngOnInit() {
  this.countryCodes = COUNTRY_CODES;
}
```

And change the `ngFor` to loop on this local variable to make it works.

My question: Is this the right way to do? I'm not totally comfortable with defining bridging variables every time I want to use global variables in my template.

angular

edited Jan 12 '18 at 0:43 asked Jun 10 '16 at 6:08

1 Id suggest having a service with the global variables or constants and injecting it where needed. – Ed Morales Jun 10 '16 at 6:55

4 Answers

27

You are creating a variable `countryCodes` in you component but the view is accessing `COUNTRY_CODES` instead*

Global identifiers like `Array`, `window`, `document`, class and enum names and global variables can't be accessed directly from within the template.

The scope of the template is the component class instance.

What you can do if you need access to any of these, is to create a getter in your component like

```
import { COUNTRY_CODES } from "../constants";

@Component(...)
export class MyComponent {
  get countryCodes() { return COUNTRY_CODES; }
  // or countryCodes = COUNTRY_CODES;
}
```

then it can be used in the template like

```
<option *ngFor="let countryCode of countryCodes" [value]="countryCode">{{countryCode |
countryName}}</option>
```

Using a shared service like suggested in the other answers works similar. What's the better approach depends on the concrete use case. Services are easy to mock for unit tests in contrary to global variables.

See also

- Select based on enum in Angular2
- How to bind a list in Angular2?

1 1 365k 83 1167 1056

1 Worked with a getter, but not with just a plain property. Thanks – Kon Feb 11 at 19:09

3

First #

you should note that there is a problem in your COUNTRY_CODES. you have put two double quotes at beginning.

It should be ,

```
Export var COUNTRY_CODES=["AD","AE","AF"];
```

Second #

Once you pass const value to `this.countryCode` , you should use it in ngfor loop like,

```
*ngFor = "let cc in countryCode" [value]="cc"
```

OR

If you directly want to use it within HTML, Id suggest to make a `sharedService` to define global variable.

UPDATE

Other way is you can use `provide function and define your constant there in provide function` within `bootstrap function` then inject it into respective component to use it.

Check out that way here,

Working Example

edited Jun 10 '16 at 7:49
Pardeep Jain
47.5k 21 112 159

answered Jun 10 '16 at 7:08
micronyks
39.6k 12 80 116

1

change this as

```
export var COUNTRY_CODES = ["AD", "AE", "AF"];
```

your code have extra `"` in your array so you have to remove this first than your code run as smothly

working plunker

edited Jun 10 '16 at 7:28

answered Jun 10 '16 at 7:09

Pardeep Jain
47.5k 21 112 159

The downvote is IMHO inappropriate, but your answer is missing the part that the template accesses `COUNTRY_CODES` but the variable is named `countryCodes`. You corrected it in your Plunker but the answer doesn't reflect that. I guess this is the cause for the downvote. – Günter Zöchbauer Jun 10 '16 at 7:33

yeah agree but i haven't added that in answer because there is mistake in the Globalvaribale only not in the template so posting juts mistaken part @GünterZöchbauer – Pardeep Jain Jun 10 '16 at 7:47

1

I wanted todo a similar thing, but with a lot of single constants.

Defining a getter for every single one was really annoying - as was the methode of creating a service for these constants, as it meant i could only use them where i can actually inject it (or again additional work).

I found that using inline templates, fixed it for me - as it allows to compile the constants into the template - using typescripts multiline `${variable}` template syntax - only thing to look out is that the values have to be wrapped according to their type. Becouse the array.toString() methode would result in '1,2,3,4' so you need to wrap it in "[]"/"'string'" so angular-templating engine picks it up as an array/string again.

Edit: I just saw there was a similar methode mentioned, but i also don't want to inject every single value into the constuctor, as that would be as uncomfortable as defining a member per value.

constants.ts:

```
export const TEST_STRING = 'route';
```

component.ts:

```
import {
 TEST_STRING,
 TEST_ARRAY
} from 'constants.ts'
@Component({
 selector: 'hn-header',
 template: '
     <a [routerLink]="['${TEST_STRING}']">Link</a>
     <div *ngFor="let test of [${TEST_ARRAY}];">{{test}}</div>
 '
 styleUrls: ['./header.component.css']
})
export class HeaderComponent {...}
```

results in:

```
<a href="route">Link</a>
<div>1</div>
<div>2</div>
<div>3</div>
<div>4</div>
```

answered Feb 27 '18 at 22:17

Florian
163 11