How can I declare a global variable in Angular 2 / Typescript? [closed]

Asked 3 years, 4 months ago Active 2 months ago Viewed 244k times

149

47

I would like some variables to be accessible everywhere in an `Angular 2` in the `Typescript` language. How should I go about accomplishing this?

typescript angular

edited May 12 at 1:01
Jeff
54 1 9

asked Mar 22 '16 at 15:40
supercobra
7,060 7 37 44

closed as primarily opinion-based by Louis, Mark, sirrocco, Sergey Kudriavtsev, YePhIcK Apr 20 '17 at 14:36

Many good questions generate some degree of opinion based on expert experience, but answers to this question will tend to be almost entirely based on opinions, rather than facts, references, or specific expertise.

If this question can be reworded to fit the rules in the help center, please edit the question.

2 If they're static variables there's no need to use services. Just add a variable in some file and then import it everywhere you need it. – Eric Martinez Mar 22 '16 at 16:00

Unfortunately Angular2 having exception at runtime, saying `Uncaught ReferenceError: Settings is not defined`. The class `Settings` with public static variables is set to export and have imported where it used. – Sen Jacob May 5 '16 at 15:56

I know this post is old and that there is many valid answers. But like Eric mentioned. If its a simple value that you want to declare and have access to through out your application you can create a class and export the class with a static property. Static variables are associated with a class rather than an instance of the class. You can import the class and will be able to access the property from the class.directly. – De Wet Ellis Sep 1 '17 at 6:09

10 Answers

Here is the simplest solution w/o `Service` nor `Observer` :

176

Put the global variables in a file an export them.

```
//
// ===== File globals.ts
//
'use strict';

export const sep='/';
export const version: string="22.2.2";
```

To use globals in another file use an `import` statement: `import * as myGlobals from './globals';`

Example:

```
//
// ===== File heroes.component.ts
//
import {Component, OnInit} from 'angular2/core';
import {Router} from 'angular2/router';
import {HeroService} from './hero.service';
import {HeroDetailComponent} from './hero-detail.component';
import {Hero} from './hero';
import * as myGlobals from './globals'; //<==== this one

export class HeroesComponent implements OnInit {
    public heroes: Hero[];
    public selectedHero: Hero;
    //
    //
    // Here we access the global var reference.
    //
    public helloString: string="hello " + myGlobals.sep + " there";

         ...

        }
    }
```

Thanks @eric-martinez

edited May 6 '17 at 5:13
snorkpete
10.4k 3 30 47

answered Mar 23 '16 at 16:43
supercobra
7,060 7 37 44

3 Got an error on the import statement. had to use `import * as globs from 'globals'` – Mike M Oct 31 '16 at 12:46

13 Why did you use "require" instead of import from ? – Ayyash Nov 5 '16 at 8:14

4 I would use `export const` instead of `export var` - you really want to make sure those global variables cannot be changed – Michal Boska Dec 21 '16 at 13:50

1 Importing like this is not valid in TypeScript anymore. Please update your answer. Correct would be: `import * as myGlobals from 'globals'` – Mick Jan 13 '17 at 10:07

7 `import * as myGlobals from './path/to/globals';` – Timothy Zorn Feb 11 '17 at 0:23

84

I like the solution from @supercobra too. I just would like to improve it slightly. If you export an object which contains all the constants, you could simply use es6 **import** the module without using **require**.

I also used Object.freeze to make the properties become true constants. If you are interested in the topic, you could read this post.

```
// global.ts

 export const GlobalVariable = Object.freeze({
     BASE_API_URL: 'http://example.com/',
     //... more of your variables
 });
```

Refer the module using import.

```
//anotherfile.ts that refers to global constants
import { GlobalVariable } from './path/global';

export class HeroService {
    private baseApiUrl = GlobalVariable.BASE_API_URL;

    //... more code
}
```

edited Apr 2 '17 at 3:38

answered Sep 7 '16 at 2:59
Tim Hong
2,216 16 22

this to me is the best solution because (1) it's the simplest with least amount of code and (2) it doesn't require you to Inject some darn service into every single component or place you want to use it in, nor does it require you to register it in @NgModule. I can't for the life of me figure out why it would be necessary to create an Angular 2 Service to do this, but perhaps there's something I'm overlooking? I'm using this great solution for now but please do let me know why the other more complicated answers here are better? – FireDragon Dec 13 '16 at 1:16

8 Your GlobalVariable is not a variable. Its a constant. – Priya R Dec 22 '16 at 19:17

@PriyaR LOL, yes, you are right. I assumed the main objective of the question was to have a safe way to access some values globally, so I improvised. Otherwise, feel free to change const to var, you get your variable. – Tim Hong Dec 22 '16 at 22:53

The down side of Object.freeze is that values are not typed. Anyway, wrapping values in a class is a better design from my perspective. So we have to choose between typed properties and true constants. – Harps Jan 24 '18 at 16:09

how to set GlobalVariable.BASE_API_URL in another component..? – Sunil Chaudhry May 30 '18 at 6:01

52

A shared service is the best approach

```
export class SharedService {
  globalVar:string;
}
```

But you need to be very careful when registering it to be able to share a single instance for your whole application. You need to define it when registering your application:

```
bootstrap(AppComponent, [SharedService]);
```

but not to define it again within the `providers` attributes of your components:

```
@Component({
  (...)
  providers: [ SharedService ], // No
  (...)
})
```

Otherwise a new instance of your service will be created for the component and its sub components.

You can have a look at this question regarding how dependency injection and hierarchical injectors work in Angular2:

- What's the best way to inject one service into another in angular 2 (Beta)?

You can notice that you can also define `Observable` properties in the service to notify parts of your application when your global properties change:

```
export class SharedService {
  globalVar:string;
  globalVarUpdate:Observable<string>;
  globalVarObserver:Observer;

  constructor() {
    this.globalVarUpdate = Observable.create((observer:Observer) => {
      this.globalVarObserver = observer;
    });
  }

  updateGlobalVar(newValue:string) {
    this.globalVar = newValue;
    this.globalVarObserver.next(this.globalVar);
  }
}
```

See this question for more details:

- Delegation: EventEmitter or Observable in Angular2

edited May 23 '17 at 12:18
Community ♦
1 1

answered Mar 22 '16 at 15:49
Thierry Templier
155k 29 335 308

Seems this one is different though. Seems @Rat2000 thinks our answer is wrong. I usually leave this decision to others than peaple providing competing answers but if he is convinced our answers are wrong then I think it's valid. The docs he linked to in a comment to my answer mention that it's DISCOURAGED but I see no disadvantage and the arguments in the docs are quite weak. It's also quite common to add providers to bootstrap. What would be the purpose of this argument anyway. And what about `HTTP_PROVIDERS` and similar, should they also not be added to `bootstrap()` ? – Günter Zöchbauer Mar 22 '16 at 16:04

2 Yes I just read the argument and the section in the doc. Honnestly I don't really understand why it's discouraged from the doc. Is it a way to define a logical split: what is Angular2 core specific (routing providers, http providers) when bootstrapping and what is application-specific in the application component injector. That said we can only have one sub injector (the application one) for the root one (defined when bootstrapping). Does I miss something? Moreover in the doc regarding hierarchical injectors, service providers are defined within the root injector ;-) – Thierry Templier Mar 22 '16 at 16:14

3 The only argument I see is that keeping the scope as narrow as possible and using the root component is at least in theory slightly narrower than using `bootstrap()` but in practice it doesn't matter. I think listing them in `boostrap()` makes the code easier to understand. A component has providers, directives, a template. I find this overloaded without global providers listed there as well. Therefore I prefer `bootstrap()` . – Günter Zöchbauer Mar 22 '16 at 16:18

2 and how does one reference such global variables? even after bootstrapping the service, calling `alert(globalVar)` results in an error. – Blauhirn Dec 20 '16 at 18:12

I haven't tried this yet, but you will want something like: alert(this.SharedService.globalVar) – trees_are_great Mar 16 '17 at 13:30

36

See for example [Angular 2 - Implementation of shared services]

```
@Injectable()
export class MyGlobals {
  readonly myConfigValue:string = 'abc';
}

@NgModule({
  providers: [MyGlobals],
  ...
})

class MyComponent {
  constructor(private myGlobals:MyGlobals) {
    console.log(myGlobals.myConfigValue);
  }
}
```

or provide individual values

```
@NgModule({
  providers: [{provide: 'myConfigValue', useValue: 'abc'}],
  ...
})

class MyComponent {
  constructor(@Inject('myConfigValue') private myConfigValue:string) {
    console.log(myConfigValue);
  }
}
```

edited Feb 15 '18 at 14:46
Demodave
3,827 4 29 43

answered Mar 22 '16 at 15:41
Günter Zöchbauer
358k 82 1136 1036

Since Angular2 beta 7(I think) you should not register your service directly in the root component(aka bootstrap). You can however inject there a specific provider if you want to override something in your application. – Mihai Mar 22 '16 at 15:51

1 Not sure what you mean. Of course you can register a service in `bootstrap()` . `bootstrap()` and root component are two different things. When you call `bootstrap(AppComponent, [MyService])` you register the service in `boostrap()` and `AppComponent` is the root component. The docs mention somewhere that it's prefered to register providers (service) in the root components `providers: ['MyService']` but I haven't found any argument in favor or against `bootstrap()` or root component yet. – Günter Zöchbauer Mar 22 '16 at 15:54

You can find your argument in the angular 2guide section Dependency Injection(angular.io/docs/ts/latest/guide/dependency-injection.html). Like they say, you can do it but it is DISCOURAGED. This user is asking for the best way to iinject something witch clearly your solution is not corect. same goes for @ThierryTemplier – Mihai Mar 22 '16 at 15:59

1 I think the more important quote from the Angular doc is "The bootstrap provider option is intended for configuring and overriding Angular's own preregistered services, such as its routing support." I rarely put services in bootstrap myself, and I'm glad to see the docs now suggest that. – Mark Rajcok Mar 22 '16 at 19:44

1 don't forget to export the class – Demodave Feb 15 '18 at 14:46

15

Create Globals class in **app/globals.ts**:

```
import { Injectable } from '@angular/core';

Injectable()
export class Globals{
    VAR1 = 'value1';
    VAR2 = 'value2';
}
```

In your component:

```
import { Globals } from './globals';

@Component({
    selector: 'my-app',
    providers: [ Globals ],
    template: `<h1>My Component {{globals.VAR1}}<h1/>`
})
export class AppComponent {
    constructor(private globals: Globals){
    }
}
```

Note: You can add Globals service provider directly to the module instead of the component, and you will not need to add as a provider to every component in that module.

```
@NgModule({
    imports: [...],
    declarations: [...],
    providers: [ Globals ],
    bootstrap: [ AppComponent ]
})
export class AppModule {
}
```

edited Oct 9 '16 at 18:28

answered Oct 9 '16 at 18:17
gradosevic
2,660 2 21 40

This is the best answer, as it offers a more portable approach than having to add the service to each component across the app. Thank you! – Vidal Quevedo Jan 12 '17 at 20:47

4 The code is working. But note that injecting class `Globals` with also adding it to `providers: [ ... ]` means you cannot change a value inside one component and then ask for the updated value inside a second component. Every time you inject `Globals` it is a fresh instance. If you want to change this behaviour, simply do **NOT** add `Globals` as provider. – Timo Bähr Jan 23 '17 at 13:58

just a note, it should be `@Injectable()` – mast3rd3mon Apr 18 '18 at 9:21

11

IMHO for Angular2 (v2.2.3) the best way is to add services that contain the global variable and inject them into components without the `providers` tag inside the `@Component` annotation. By this way you are able to share information between components.

A sample service that **owns** a global variable:

```
import { Injectable } from '@angular/core'

@Injectable()
export class SomeSharedService {
  public globalVar = '';
}
```

A sample component that **updates** the value of your global variable:

```
import { SomeSharedService } from '../services/index';

@Component({
  templateUrl: '...'
})
export class UpdatingComponent {

  constructor(private someSharedService: SomeSharedService) { }

  updateValue() {
    this.someSharedService.globalVar = 'updated value';
  }
}
```

A sample component that **reads** the value of your global variable:

```
import { SomeSharedService } from '../services/index';

@Component({
  templateUrl: '...'
})
export class ReadingComponent {

  constructor(private someSharedService: SomeSharedService) { }

  readValue() {
    let valueReadOut = this.someSharedService.globalVar;
    // do something with the value read out
  }
}
```

> Note that `providers: [ SomeSharedService ]` should **not** be added to your `@Component` annotation. By not adding this line injection will always give you the same instance of `SomeSharedService` . If you add the line a freshly created instance is injected.

answered Jan 23 '17 at 14:55
Timo Bähr
915 10 19

But without adding the line of providers, I got an error like this: `Unhandled Promise rejection: No provider for SomeSharedService` – Rocky Mar 16 '17 at 14:14

I see. I should add `providers: [SomeSharedService]` in the parent module file. Thanks. – Rocky Mar 16 '17 at 14:18

This doesn't work when we are lazy loading modules. – Ipradhap Nov 11 '18 at 1:10

7

I don't know the best way, but the easiest way if you want to define a global variable inside of a component is to use `window` variable to write like this:

```
window.GlobalVariable = "what ever!"
```

you don't need to pass it to bootstrap or import it other places, and it is globally accessibly to all JS (not only angular 2 components).

answered Jun 30 '16 at 17:16
Mahdi Jadaliha
1,287 10 19

1 I'd say it's the worst way. Using a static variable isn't more complicated but isn't that ugly either ;-) – Günter Zöchbauer Jun 30 '16 at 17:19

2 I agree it makes difficult to manage. However I ended to use them in development till I find what I want to put in productions. In static variable you have to import them again and again everywhere you want to use, beside there was a case I was producing my view on the go with jquery in angular components - there was no template, and to add events to produced DOM using static variable is pain. – Mahdi Jadaliha Jun 30 '16 at 20:48

1 Plus, it is not static, you can change the value from everywhere! – Mahdi Jadaliha Jun 30 '16 at 20:56

1 It also wrecks server side rendering. Stay away from directly manipulating the window or document. – Erik Honn Oct 28 '16 at 8:23

It is only 1 line, it does not follow Angular guidelines – Jonathan Muller May 20 at 5:48

5

That's the way I use it:

global.ts

```
export var server: string = 'http://localhost:4200/';
export var var2: number = 2;
export var var3: string = 'var3';
```

to use it just import like that:

```
import { Injectable } from '@angular/core';
import { Http, Headers, RequestOptions } from '@angular/http';
import { Observable } from 'rxjs/Rx';
import * as glob from '../shared/global'; //<== HERE

@Injectable()
export class AuthService {
    private AuhtorizationServer = glob.server
}
```

EDITED: Droped "_" prefixed as recommended.

edited Apr 25 at 17:21
Ferie
478 8 22

answered Sep 23 '16 at 22:48
Guilherme Teubl
651 8 8

Do not use "_" as a prefix for private properties.github.com/Microsoft/TypeScript/wiki/Coding-guidelines – crh225 Sep 29 '16 at 21:08

4

I think the best way is to share an object with global variables throughout your application by exporting and importing it where you want.

First create a new **.ts** file for example **globals.ts** and declare an object. I gave it an **Object** type but you also could use an any type or {}

```
export let globalVariables: Object = {
 version: '1.3.3.7',
 author: '0x1ad2',
 everything: 42
};
```

After that import it

```
import {globalVariables} from "path/to/your/globals.ts"
```

And use it

```
console.log(globalVariables);
```

edited May 23 '17 at 11:33
Community ♦
1 1

answered Dec 14 '16 at 14:01
0x1ad2
6,004 7 26 40

3

I like the answer of @supercobra, but I would use the const keyword as it is in ES6 already available:

```
//
// ===== File globals.ts
//
'use strict';

export const sep='/';
export const version: string="22.2.2";
```

answered Jul 20 '16 at 11:50

Zolcsi

1,406 7 11