Use absolute paths for module imports

Adrian Fâciu Follow
Nov 4, 2017 · 2 min read

Handling imports is a bit more trickier to manage due to paths and constant refactoring that one will do inside a more complex application.
When importing modules they are usually separated in two:

- importing libraries modules, which is quite nice, since we use only the name and they are resolved from *node_modules* folder
- importing modules from our own application

When doing the second the web is, unfortunately, full of examples that use relative paths to get the needed file:

```
import { foo } from '../bar';
```

Which is quite ok, but it goes quickly wrong when the file we need is several levels up or down.

Then we get something like:

```
import { foo } from '../../../../bar';
```

Which is hard to write/read and will break as soon as we move the file somewere else.
Since **TypeScript 2.0**, we have an awesome compiler setting called **baseUrl**, and we can configure it in the tsconfig.json file like:

```
{
  "compilerOptions": {
    "baseUrl": "./src"
  }
}
```

If you are using *Angular CLI* this will be already setup for you. In the tsconfig file for the app it's set

Once this is done we can use a path starting from that base url, so most of our imports will be transformed to something like:

```
import { foo } from 'app/bar'
```

I would use this for almost all of the imports throughout an application. Those that import from the same folder might be left as they are though.
All our imports will be shorter to write, easy to understand and most important: **will not break as we refactor and move files around**.
With a few barrel files strategically placed we will have a very nice structure with imports that look like:

416

```
import { UserService } from 'app/core/services';
import { SearchComponent } from 'app/shared/components';
```

If you want to go a bit further there is also a setting for creating path mappings, but as of now, most editors and IDEs have some issues with correctly resolving this and in my opinion it is a bit harder to understand what is happening.
It should be clear enough why this way of importing modules is a bit better and more error proof.
We should all use this more when working on large application codebases.

Angular Typescript Modules Import Export Best Practices

416 claps




WRITTEN BY

Adrian Fâciu

passionate software developer

Follow


Upgrade

See responses (1)

More From Medium

416

More from Adrian Fâciu

Creating a toast service with Angular CDK

Adrian Fâciu in...
Nov 21, 2018 · 14 ...

1.7K

Related reads

Angular Router Series: Secondary Outlets Primer

Nate Lapinski in...
Sep 17, 2018 · 4 mi...

1.4K

Related reads

Creating a custom form field control compatible with Reactive Forms and Angular Material

Vassiliki Dalakiari i...

416