# 

# Create NPM module using YO and library generator

I used <https://github.com/jvandemo/generator-angular2-library> and found it useful for generating npm modules from angular 2, 4, 5 and so on.

## Process

### Step 1: Install and create container to generate the NPM module

$ npm install -g yo

$ npm install -g generator-angular2-library

make a new directory and cd into it:

$ mkdir angular-library-name

$ cd angular-library-name

and generate your new library:

$ yo angular2-library

### Step 2: Create npm module

* Create an angular module in your angular app. Copy and paste your angular module in the src folder as created above.
* Rename the module.ts of the src folder to index.ts
* Remove all samples from src and let stay the package.json and other system files. Following command will generate the npm module in dist folder:

$ npm run build

* Now publish the NPM module by cd dist, npm login, npm publish

### important points

* You must provide exclusive export statements in the module.ts file which is copied into src folder after renaming it as index.ts. For every component and every service there should be independent export \* statement as follows. If you have a nested module in your existing module then there should be one export for the nested module also. In present case there is a nested module broker in the current module.

export \* from './my-graph/my-graph.component';

export \* from './my-graph/info/info.component';

export \* from './neuro.service';

export \* from './neuro1.service';

export \* from './broker/broker.module';

* In nested module’s module.ts file you must exclusively export its components and services. In present case broker.module is a nested module and it has a broker.service and broker.component. So in broker.module.ts following lines are required:

export \* from './broker/broker.component';

export \* from './broker.service';

* If you have a service in parent module then the module.ts or index.ts must treat the service as follows. Parent module name is NeuroGraph1Module and it has got two services NeuroService and Neuro1Service. Conventionally there is one static method created as forRoot() which returns the services as providers. These service need not exist in any other providers[] statement.

export class NeuroGraph1Module {

static forRoot(): ModuleWithProviders {

return {

ngModule: NeuroGraph1Module,

providers: [NeuroService, Neuro1Service]

};

}

}

In app.module.ts file you must add the forRoot() when using the NPM module as follows:

NeuroGraph1Module.forRoot()

* If you have services in a nested module then these have to be treated accordingly. In module.ts file of the nested module provide the forRoot() statement as follows. The nested broker module has one broker service in present case.

export class BrokerModule {

static forRoot(): ModuleWithProviders {

return {

ngModule: BrokerModule,

providers: [BrokerService]

};

}

}

* Since the parent module NeuroGraph1 module has the nested Broker Module so in NeuroGraph1 module consume the Broker module with forRoo()

imports: [

CommonModule

, BrokerModule.forRoot()

],

Sample tested code is provided in present folder

### Point to note

I used npm script “npm run libgen” in the neuro-app’s package.json file to generate the NPM module. I noticed that **npm install npmModuleName** or **npm install npmModuleName@latest** or **npm install npmModuleName@latest –force** do not install the latest versions. Please use **npm update npmModuleName**. This installs the latest version of npm module.



Get details of all modules at npmjs.org

* To search a module

*Npm search sql*

* To show all modules a project is using

*Npm ls*

* To update all packages to latest

*Npm update*

npm install -g npm-check-updates

npm-check-updates or ncu can be used to correct the version numbers in package.json file. Command is ncu –upgradeAll or ncu –u.

This will check the internet repository, find the latest versions available and change your package.json file with latest versions of modules. E.g if your package.json has a moduleA with version 2.1 and internet repository has available new version as 2.3 then your package.json file will contain moduleA: 2.3. It will not change the content of node\_modules.

To load a module use require(‘http’) command. http is module. Node searches the module in node-modules folder recursively

If you give npm install –g express then express will be installed globally