Publishing Angular Module by Murthy

//**never import BrowserModule.**

If you need the common directives (\*ngIf, \*ngFor…), import CommonModule.

import { NgModule } from '@angular/core';

import { CommonModule } from '@angular/common';

@NgModule({

imports: [CommonModule]

})

export class AmazingModule {}

If your module is about creating new components, directives or pipes, do not forget to **export them**. Declared ones are only accessible inside your module.

|  |
| --- |
|  |
|  |  |
|  | import { PrivateComponent } from './private.component'; |
|  | import { PublicComponent } from './public.component'; |
|  |  |
|  | @NgModule({ |
|  | declarations: [ |
|  | PrivateComponent, |
|  | PublicComponent |
|  | ], |
|  | exports: [PublicComponent] |
|  | }) |
|  | export class AmazingModule {} |

**do *not* mix components/directives/pipes and services in the same module**. Why?

* **A service provided in a module will be available everywhere** in the app, so your module should be imported only once, in the user app *root*module (like the Http module).
* **An exported component/directive/pipe will only be available in the module importing yours**, so your module should be imported in every user module (root and/or feature modules) that need them (like the CommonModule).

**Build tools**

It’s where I started to struggle. So I managed to copy how official Angular modules work, like the HttpModule. They use:

* [typescript](https://www.typescriptlang.org/), via the Angular compiler (ngc), for **transpiling**,
* [rollupjs](http://rollupjs.org/) for **packaging**,
* [uglify-js](https://github.com/mishoo/UglifyJS2) for **minifying**.

npm install **@angular/compiler @angular/compiler-cli typescript rollup uglify-js** --save-dev

Typescript configuration:

{

"compilerOptions": {

"baseUrl": ".",

"declaration": true,

"stripInternal": true,

"experimentalDecorators": true,

"strictNullChecks": true,

"noImplicitAny": true,

"module": "es2015",

"moduleResolution": "node",

"paths": {

"@angular/core": ["node\_modules/@angular/core"],

"rxjs/\*": ["node\_modules/rxjs/\*"]

},

"rootDir": ".",

"outDir": "dist",

"sourceMap": true,

"inlineSources": true,

"target": "es5",

"skipLibCheck": true,

"lib": [

"es2015",

"dom"

]

},

"files": [

"index.ts"

],

"angularCompilerOptions": {

"strictMetadataEmit": true

}

}

* **explicit "paths" to other modules you use are *needed***, as the final bundle won’t include them directly (more on that later).
* **"angularCompilerOptions": { "strictMetadataEmit": true } is *needed* to be [AoT](https://angular.io/docs/ts/latest/cookbook/aot-compiler.html" \t "_blank) compatible**.
* "declaration": true is important to generate type definitions files, so the user will have Intellisense for your module.
* "noImplicitAny": true and "strictNullChecks": true are recommended to avoid errors, and to be compatible with all user configurations. "noImplicitAny": true must be respected since Angular 4.0, and "strictNullChecks": true starting from Angular 4.1.
* "module": "es2015" is important for performance, and "sourceMap": true for debugging, but nothing specific here.
* "stripInternal": true avoid useless declarations for internal APIs and "skipLibCheck": true avoid being blocked by (harmless) errors in the librairies you use.

### Rollup configuration

**Angular modules are delivered in UMD format**, so your rollup.config.jsshould be set consequently.

export default {

entry: 'dist/index.js',

dest: 'dist/bundles/amazing.umd.js',

sourceMap: false,

format: 'umd',

moduleName: 'ng.amazing',

globals: {

'@angular/core': 'ng.core',

'rxjs/Observable': 'Rx',

'rxjs/ReplaySubject': 'Rx',

'rxjs/add/operator/map': 'Rx.Observable.prototype',

'rxjs/add/operator/mergeMap': 'Rx.Observable.prototype',

'rxjs/add/observable/fromEvent': 'Rx.Observable',

'rxjs/add/observable/of': 'Rx.Observable'

}

}

Npm scripts :

{

"scripts": {

"transpile": "ngc",

"package": "rollup -c",

"minify": "uglifyjs dist/bundles/amazing.umd.js --screw-ie8 --compress --mangle --comments --output dist/bundles/amazing.umd.min.js",

"build": "npm run transpile && npm run package && npm run minify"

}

}

Then:

**npm run build**

Note that transpiling is not done directly by TypeScript, **you should use the Angular compiler (ngc)** : it’s TypeScript with some additional Angular magic.

### Publishing on npm

**Do not publish everything on npm, only the dist directory.**

You’ll need to create **a new and specific dist/package.json**. For example :

{

"name": "angular-amazing",

"version": "1.0.0",

"description": "An amazing module for Angular.",

"main": "bundles/amazing.umd.js",

"module": "index.js",

"typings": "index.d.ts",

"keywords": [

"angular",

"angular2",

"angular 2",

"angular4"

],

"author": "Your name",

"license": "MIT",

"repository": {

"type": "git",

"url": "https://github.com/youraccount/angular-amazing.git"

},

"homepage": "https://github.com/youraccount/angular-amazing",

"bugs": {

"url": "https://github.com/youraccount/angular-amazing/issues"

},

"peerDependencies": {

"@angular/core": "^2.4.0 || ^4.0.0",

"rxjs": "^5.0.1"

}

}

Some specific points :

* "version" *must* follow [**semantic versioning**](http://semver.org/). **Any breaking change means a major number increment** (even if it’s a small change). And when you’ll modify your module to stay up to date with Angular, it’s a minor number increment.
* **"main" and "module" paths are *needed* for user imports***.*"typings" path is for Intellisense.
* "licence": "MIT": **an open-source licence is important**, or your module is useless. Angular uses the MIT licence, and you should stick to it.

 can now **publish your module:**

cd **dist**  
**npm publish**