

How to Create New Angular Libraries

In this Article we are going to talk about Libraries in Angular 13. We will be covering the following topics:

- Reusability with Angular Libraries:
 - Here we introduce the idea of reusability and the concept of building libraries in Angular apps.
- Creating a New Library:
 - Here we will walk through steps to generate a new library using the `ng generate`.
- Rebuilding an app using libraries:
 - Here we walk through an example where we convert a portion of an existing Angular 13 app to a library and then use it.
- Publishing your library:
 - Here we walk through publishing a library, keeping Ivy related features, version compatibility etc. in mind.

According to angular.io, some of the features worth mentioning are:

1. DEVELOP ACROSS ALL PLATFORMS

- Learn one way to build applications with Angular and reuse your code and abilities to build apps for any deployment target. For web, mobile web, native mobile, and native desktop.

2. SPEED & PERFORMANCE

- Achieve the maximum speed possible on the Web Platform today, and take it further, via [Web Workers](#) and server-side rendering.
- Angular puts you in control over scalability. Meet huge data requirements by building data models on RxJS, Immutable.js, or another push model.

3. INCREDIBLE TOOLING

- Build features quickly with simple, declarative templates. Extend the template language with your own components and use a wide array of existing components.
- Get immediate Angular-specific help and feedback with nearly every IDE and editor. All this comes together so you can focus on building amazing apps rather than trying to make the code work.

4. LOVED BY MILLIONS

- From prototype through global deployment, Angular delivers the productivity and scalable infrastructure that supports Google's largest applications.

Reusability with Angular Libraries

A Library in Angular App is a reusable code that determines [Angular](#) components, services, or projects meant to be called by other projects. These projects are like a normal application. The only difference these have over Angular applications is that these projects cannot execute on their own. We need to import these projects to use them.

We make libraries to solve some generic issues like the unified user interface or data presentation or data entry issues. Angular developers create these libraries for such general-purpose solutions. Later these libraries can be adapted in different applications for re-usage of the solution.

An Angular Library can be built as an Angular application, later published as an npm package. These npm packages can be shared with different applications deploying the libraries.

Libraries can be published as third-party libraries or Angular Team libraries.

Some of the examples are:

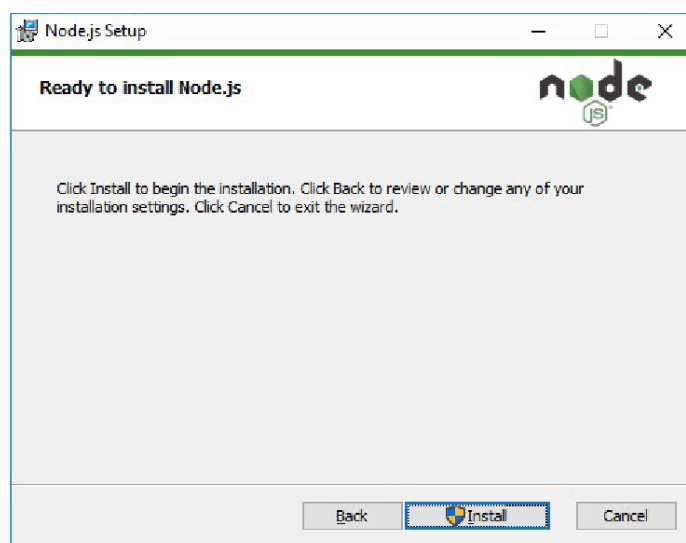
1. `ReactiveFormsModule` from the `@angular/forms` library is used to add reactive forms to an Angular app. We add this library package by using `ng add @angular/forms` and then importing the `ReactiveFormsModule` from the `@angular/forms`.
2. To turn an Angular App into a progressive application we add the service worker library. This turns our code into a Progressive Web App (PWA).
3. A very large and general-purpose library is the [Angular Material](#). It is a library or a component that helps in constructing attractive, sophisticated, functional web app and reusable, and adaptable UI components.

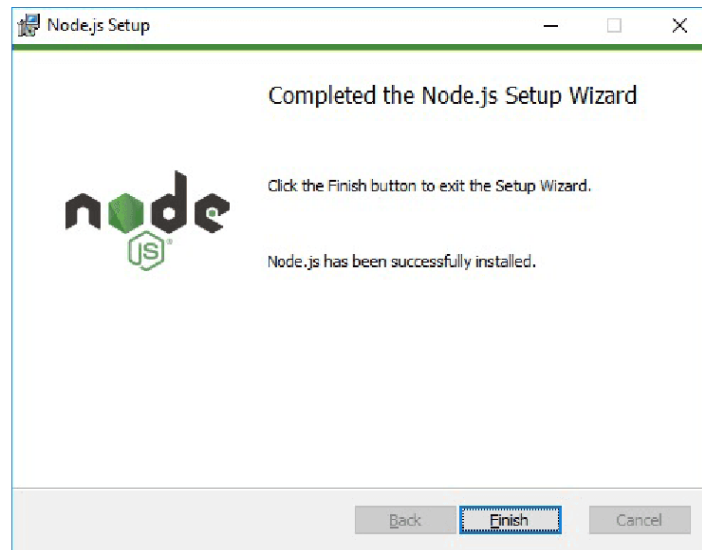
Creating a New Library

Let us build a Library in Angular App step-by-step:

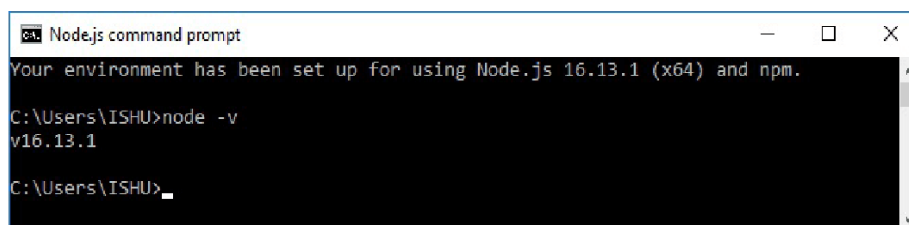
Step 1: Install Node.js:

- Angular requires Node.js version 14.X.X or later. You can download it from [Nodejs.org](https://nodejs.org).
 - Latest Version is: node-v16.13.1-x64
- Install node.js once downloaded:





- Once you have installed [Node.js](#) on your system, open node.js command prompt.
- To check your version, run `node -v` in a terminal/console window.



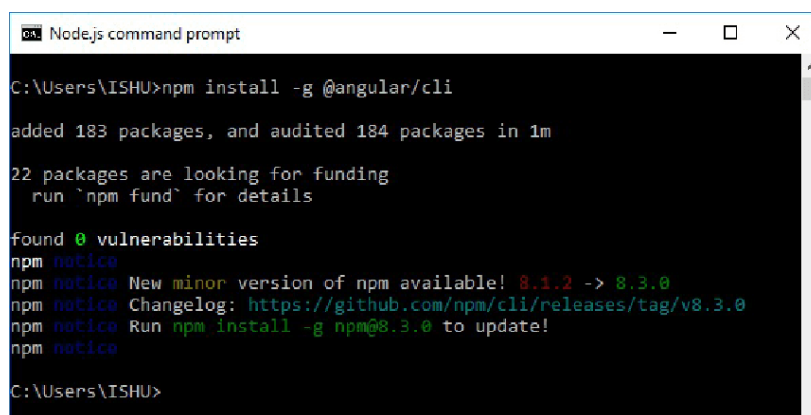
Step 2: Use npm to Install Angular CLI

- Use the following command to install Angular CLI

```
npm install -g @angular/cli
```

Or

```
npm install -g @angular/cli@latest
```



Or

- Just go to Angular CLI official website [Angular.io](https://angular.io).
- You will see the whole cli command to create an Angular app. You need to run the first command to install Angular CLI. These steps are the same for Windows and Mac.



- To check Node and Angular CLI version, use `ng --version` command.

```

Node.js command prompt
C:\Users\ISHU>ng --version

Angular CLI
Angular CLI: 13.1.1
Node: 16.13.1
Package Manager: npm 8.1.2
OS: win32 x64

Angular:
...

Package      Version
-----
@angular-devkit/architect    0.1301.1 (cli-only)
@angular-devkit/core        13.1.1 (cli-only)
@angular-devkit/schematics   13.1.1 (cli-only)
@schematics/angular         13.1.1 (cli-only)

C:\Users\ISHU>

```

Step 3: Create an app called ngApp4Library

Syntax:

```
ng new app_name
```

```
C:\>ng new ngApp4Library
```

It asks for

Would you like to add Angular routing? Yes

Which stylesheet format would you like to use?

> CSS

....

```

C:\Users\ISHU\Desktop>ng new ngApp4Library
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE ngApp4Library/angular.json (3083 bytes)
CREATE ngApp4Library/package.json (1079 bytes)
CREATE ngApp4Library/README.md (1059 bytes)
CREATE ngApp4Library/tsconfig.json (863 bytes)
CREATE ngApp4Library/.editorconfig (274 bytes)
CREATE ngApp4Library/.gitignore (620 bytes)
CREATE ngApp4Library/.browserslistrc (600 bytes)
CREATE ngApp4Library/karma.conf.js (1432 bytes)
CREATE ngApp4Library/tsconfig.app.json (287 bytes)
CREATE ngApp4Library/tsconfig.spec.json (333 bytes)
CREATE ngApp4Library/.vscode/extensions.json (130 bytes)
CREATE ngApp4Library/.vscode/launch.json (474 bytes)
CREATE ngApp4Library/.vscode/tasks.json (938 bytes)
CREATE ngApp4Library/src/favicon.ico (948 bytes)
CREATE ngApp4Library/src/index.html (299 bytes)
CREATE ngApp4Library/src/main.ts (372 bytes)
CREATE ngApp4Library/src/polyfills.ts (2338 bytes)
CREATE ngApp4Library/src/styles.css (80 bytes)
CREATE ngApp4Library/src/test.ts (745 bytes)
CREATE ngApp4Library/src/assets/.gitkeep (0 bytes)
CREATE ngApp4Library/src/environments/environment.prod.ts (51 bytes)
CREATE ngApp4Library/src/environments/environment.ts (658 bytes)
CREATE ngApp4Library/src/app/app-routing.module.ts (245 bytes)
CREATE ngApp4Library/src/app/app.module.ts (393 bytes)
CREATE ngApp4Library/src/app/app.component.html (23364 bytes)
CREATE ngApp4Library/src/app/app.component.spec.ts (1094 bytes)
CREATE ngApp4Library/src/app/app.component.ts (217 bytes)
CREATE ngApp4Library/src/app/app.component.css (0 bytes)
- Installing packages (npm)...
  
```

```

C:\Users\ISHU\Desktop>ng new ngApp4Service
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
√ Packages installed successfully.
'git' is not recognized as an internal or external command,
operable program or batch file.
Nothing to be done.
C:\Users\ISHU\Desktop>
  
```

Step 4: Generate Library via CLI:

Syntax: for creating a Library

```
ng generate library <name> [options]
```

```
ng g library <name> [options]
```

- Let us generate the required library: here we are going to create “my-lib” Library.

- Go to the app folder and install the required Library: “my-lib”:

ng generate library my-lib

```

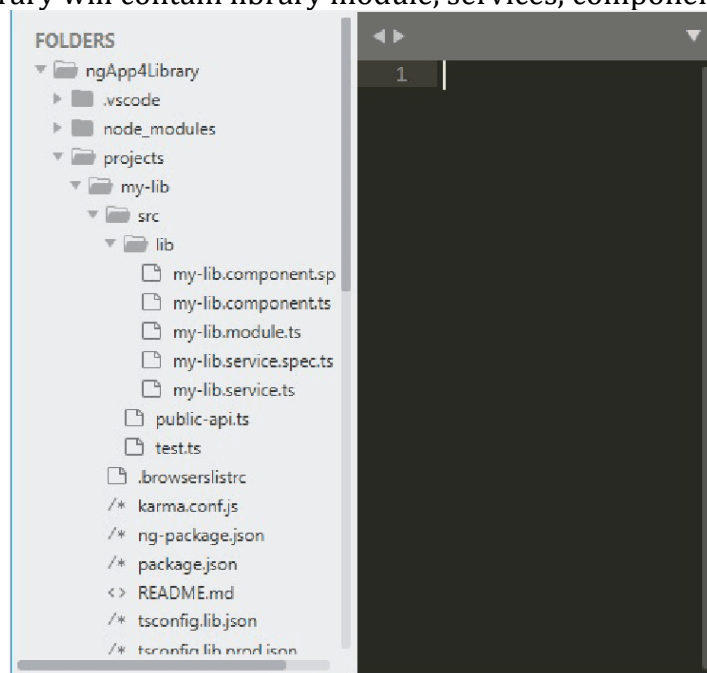
Node.js command prompt
C:\Users\ISHU\Desktop>cd ngApp4Library

C:\Users\ISHU\Desktop\ngApp4Library>ng generate library my-lib
CREATE projects/my-lib/.browserslistrc (600 bytes)
CREATE projects/my-lib/karma.conf.js (1427 bytes)
CREATE projects/my-lib/ng-package.json (155 bytes)
CREATE projects/my-lib/package.json (185 bytes)
CREATE projects/my-lib/README.md (978 bytes)
CREATE projects/my-lib/tsconfig.lib.json (333 bytes)
CREATE projects/my-lib/tsconfig.lib.prod.json (240 bytes)
CREATE projects/my-lib/tsconfig.spec.json (309 bytes)
CREATE projects/my-lib/src/test.ts (763 bytes)
CREATE projects/my-lib/src/public-api.ts (155 bytes)
CREATE projects/my-lib/src/lib/my-lib.module.ts (238 bytes)
CREATE projects/my-lib/src/lib/my-lib.component.spec.ts (620 bytes)
CREATE projects/my-lib/src/lib/my-lib.component.ts (264 bytes)
CREATE projects/my-lib/src/lib/my-lib.service.spec.ts (353 bytes)
CREATE projects/my-lib/src/lib/my-lib.service.ts (134 bytes)
UPDATE angular.json (4054 bytes)
UPDATE package.json (1108 bytes)
UPDATE tsconfig.json (963 bytes)
✓ Packages installed successfully.

C:\Users\ISHU\Desktop\ngApp4Library>

```

- This will create a library project my-lib into our ngApp4library project.
- my-lib Library will contain library module, services, components, etc.



Step 5: Edit the library ts file: Give a functionality to your library

As you can see our library has its own module, service, and component. We can add more components, services, directives, pipes, and modules as per our needs.

The Library file that we need to edit is my-lib. It appears in the folder C:\Users\ISHU\Desktop -> ngApp4Library -> projects -> my-lib -> src -> lib. The file to be edited is: **my-lib.component.ts**

Add the following code:

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'lib-my-lib',
  template: `
<form method="post">

  <div >

    <label for = "username"> <b> Username: </b> </label>

    <input type = "text" placeholder = "Enter Username here" name =
"username" style = "margin:10px;" required>

    <br/>

    <label for = "passwd"> <b> Password: </b> </label>

    <input type = "password" placeholder = "Enter Password here" name =
"passwd" style = "margin:10px;" required>

    <br/>

    <button type = "submit"> Login </button>

  </div>

</form>

`,
  styles: [
  ]
})
```



```

}))

export class MyLibComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

}
}

```

Rebuilding an app using libraries

1. Before consuming the library, we need to build an Angular library.
2. Here we will build the library for local (same application) usage.
3. Then we will re-build the library for global (any application) usage.

Creating a New Library

1. Build the library and consume it in the same application:

- To build the library, we run the following command:

```
ng build <library name>
```

- Here our library name is my-lib, thus the command we need is: **ng build my-lib**

```

C:\Users\ISHU\Desktop\ngApp4Library>ng build my-lib
Building Angular Package

-----
Building entry point 'my-lib'
-----
✓ Compiling with Angular sources in Ivy partial compilation mode.
✓ Generating FESM2020
✓ Generating FESM2015
✓ Copying assets
✓ Writing package manifest
✓ Built my-lib

-----
Built Angular Package
 - from: C:\Users\ISHU\Desktop\ngApp4Library\projects\my-lib
 - to:   C:\Users\ISHU\Desktop\ngApp4Library\dist\my-lib
-----

Build at: 2021-12-19T09:49:39.306Z - Time: 24392ms

C:\Users\ISHU\Desktop\ngApp4Library>

```

- The command will generate the library **dist folder**
- Next step is to implement the library into our current project: ngApp4Library:

- For this step, we need to import this library in our main app (ngApp4Library).
- In app.module.ts import my-lib library module as shown: **app.module.ts**:

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

import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

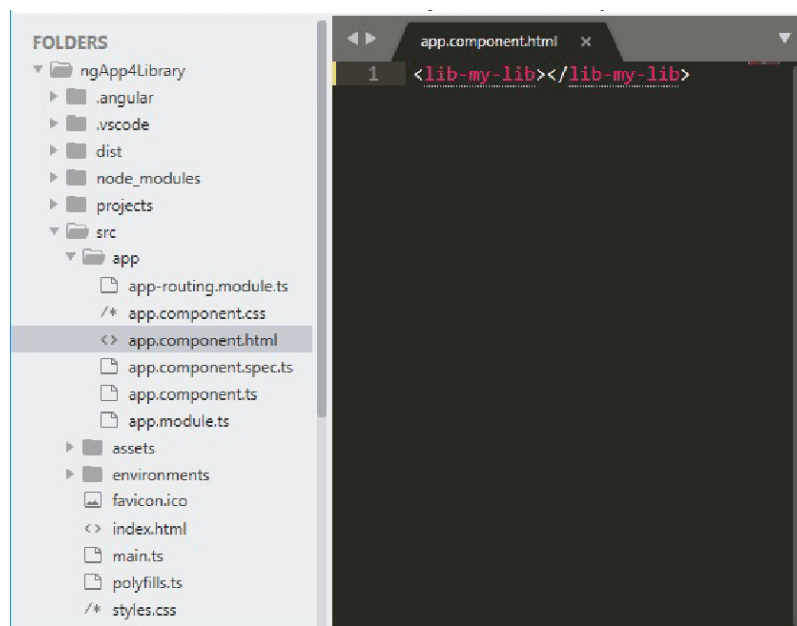
import { MyLibModule } from 'my-lib';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    MyLibModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})

export class AppModule { }
```

- Now, we simply add the my-lib library in the HTML file.
- Please note that the selector for the library used here is: **lib-my-lib**
- To know about the selector (**lib-my-lib**), we can check the file: ngApp4Library -> projects -> my-lib -> src -> lib-> **my-lib.component.ts**
- Now open and edit: **app.component.html** from the main App folder: **ngApp4Library\src\app\ app.component.html**:

```
<lib-my-lib></lib-my-lib>
```



- Now we can start our application from the Node.js command line as: **ng serve**

```
ngcc

C:\Users\ISHU\Desktop\ngApp4Library>ng serve
✓ Browser application bundle generation complete.

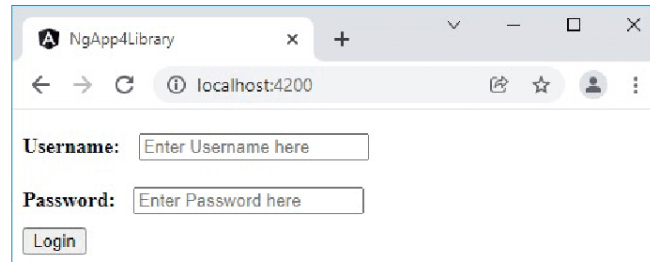
Initial Chunk Files | Names          | Raw Size
vendor.js           | vendor         | 2.00 MB
polyfills.js        | polyfills      | 339.25 kB
styles.css, styles.js | styles         | 212.51 kB
main.js             | main           | 12.91 kB
runtime.js          | runtime        | 6.86 kB
                    | Initial Total  | 2.56 MB

Build at: 2021-12-19T09:59:00.153Z - Hash: 75195efb3aef2954 - Time: 51699ms

** Angular Live Development Server is listening on localhost:4200, open your browser on
http://localhost:4200/ **

✓ Compiled successfully.
```

- Open the favourite browser and type the default Angular app URL to check the output: localhost:4200/.



2. Rebuild the library and consume it from some other application:

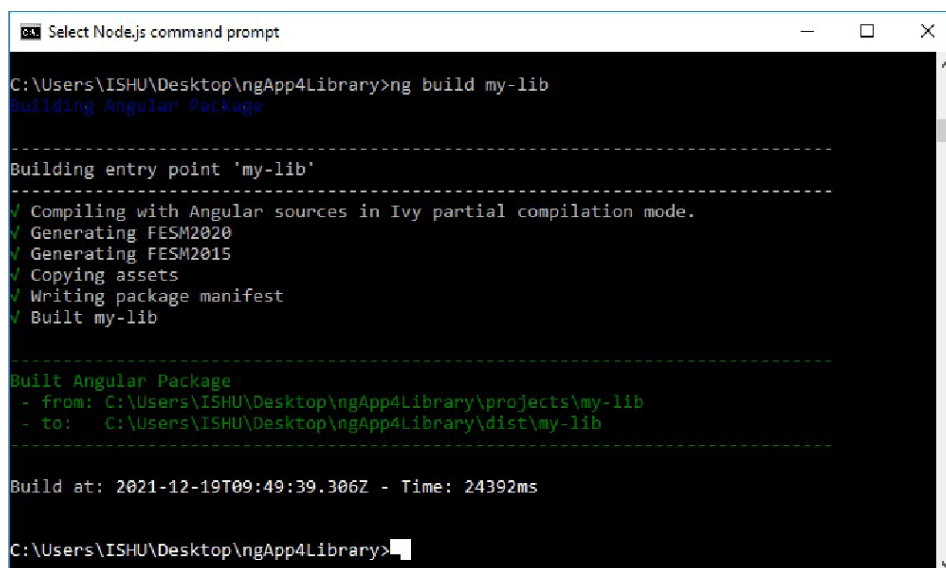
We go through the following steps to implement the Library into another project:

Step 1:

- To Re-build the library, we run the following command:

```
ng build <library name>
```

- Here our library name is my-lib, thus the command we need is: **ng build my-lib**



- The command will generate the library **dist folder**
- Now, create a new Angular application: ngAppClient in a new command window.
We need to let the library application run.

```
ng new ngAppClient
```

```

C:\Users\ISHU\Desktop>ng new ngAppClient
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE ngAppClient/angular.json (3071 bytes)
CREATE ngAppClient/package.json (1077 bytes)
CREATE ngAppClient/README.md (1057 bytes)
CREATE ngAppClient/tsconfig.json (863 bytes)
CREATE ngAppClient/.editorconfig (274 bytes)
CREATE ngAppClient/.gitignore (620 bytes)
CREATE ngAppClient/.browserslistrc (600 bytes)
CREATE ngAppClient/karma.conf.js (1430 bytes)
CREATE ngAppClient/tsconfig.app.json (287 bytes)
CREATE ngAppClient/tsconfig.spec.json (333 bytes)
CREATE ngAppClient/.vscode/extensions.json (130 bytes)
CREATE ngAppClient/.vscode/launch.json (474 bytes)
CREATE ngAppClient/.vscode/tasks.json (938 bytes)
CREATE ngAppClient/src/favicon.ico (948 bytes)
CREATE ngAppClient/src/index.html (297 bytes)
CREATE ngAppClient/src/main.ts (372 bytes)
CREATE ngAppClient/src/polyfills.ts (2338 bytes)
CREATE ngAppClient/src/styles.css (80 bytes)
CREATE ngAppClient/src/test.ts (745 bytes)
CREATE ngAppClient/src/assets/.gitkeep (0 bytes)
CREATE ngAppClient/src/environments/environment.prod.ts (51 bytes)
CREATE ngAppClient/src/environments/environment.ts (658 bytes)
CREATE ngAppClient/src/app/app-routing.module.ts (245 bytes)
CREATE ngAppClient/src/app/app.module.ts (393 bytes)
CREATE ngAppClient/src/app/app.component.html (23364 bytes)
CREATE ngAppClient/src/app/app.component.spec.ts (1088 bytes)
? Packages installed successfully.
'git' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\ISHU\Desktop>

```

- Now from the build, check the paths destined as “to”

```

C:\Users\ISHU\Desktop\ngApp4Library>ng build my-lib
Building Angular Package

-----
Building entry point 'my-lib'
-----

✓ Compiling with Angular sources in Ivy partial compilation mode.
✓ Generating FESM2020
✓ Generating FESM2015
✓ Copying assets
✓ Writing package manifest
✓ Built my-lib

-----
Built Angular Package
- from: C:\Users\ISHU\Desktop\ngApp4Library\projects\my-lib
- to:   C:\Users\ISHU\Desktop\ngApp4Library\dist\my-lib
-----

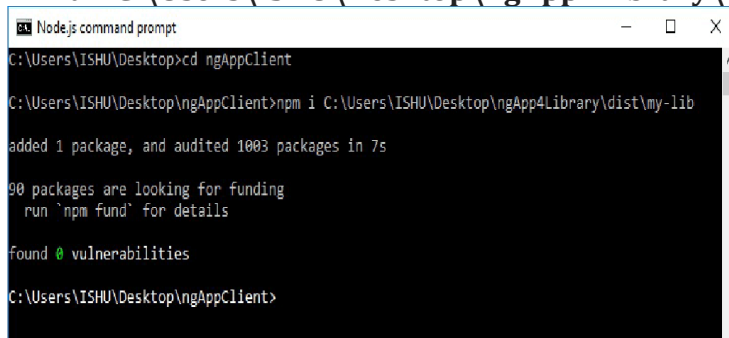
Build at: 2021-12-19T09:49:39.306Z - Time: 24392ms

C:\Users\ISHU\Desktop\ngApp4Library>

```

- Here, the “to” part gives us the value: C:\ Users\ ISHU\ Desktop\ ngApp4Library\ dist\ my-lib
- So, now copy this path: “C:\ Users\ ISHU\ Desktop\ ngApp4Library\ dist\ my-lib”
- Next, we open the terminal of ngAppClient project and install this library with the following command into the ngAppClient project:
 - **npm i C:\ Users\ ISHU\ Desktop\ ngApp4Library\ dist\ my-lib**
 - **C:\Users\ISHU\Desktop>cd ngAppClient**

- C:\Users\ISHU\Desktop\ngAppClient>npm i
- C:\Users\ISHU\Desktop\ngApp4Library\dist\my-lib



```

Node.js command prompt
C:\Users\ISHU\Desktop>cd ngAppClient
C:\Users\ISHU\Desktop\ngAppClient>npm i C:\Users\ISHU\Desktop\ngApp4Library\dist\my-lib
added 1 package, and audited 1003 packages in 7s
90 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
C:\Users\ISHU\Desktop\ngAppClient>

```

- After installation of the library in the client application, we can import the library into the app.module.ts of the client app. After importing the library module, we can easily use those services, components, etc.
- To use the library:
 - Open Client Project -> app.module.ts file and edit it to add the library:

App.module.ts:

```

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

import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { MyLibModule } from 'my-lib';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [

```

```

    BrowserModule,

    AppRoutingModule,

    MyLibModule

  ],

  providers: [],

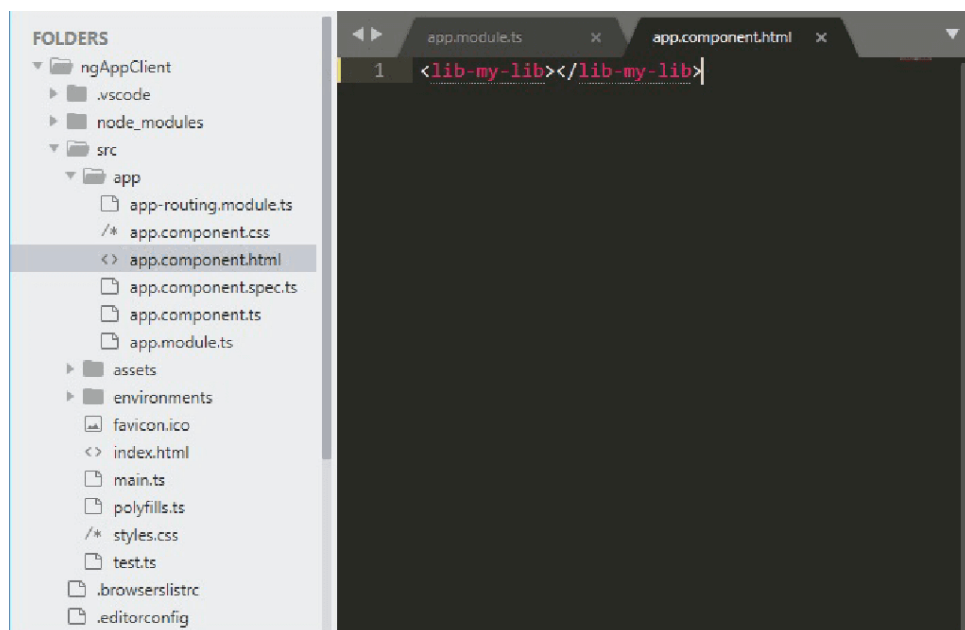
  bootstrap: [AppComponent]
})

export class AppModule { }

```

- Now, we simply add the my-lib library in the client project app.component.html file.
- Please note that the selector for the library used here is: **lib-my-lib**
- To know about the selector (**lib-my-lib**), we can check the file: ngApp4Library -> projects -> my-lib -> src -> lib-> **my-lib.component.ts**
- And then open and edit: **app.component.html** from the client App folder: **ngAppClient\src\app\app.component.html**:

```
<lib-my-lib></lib-my-lib>
```

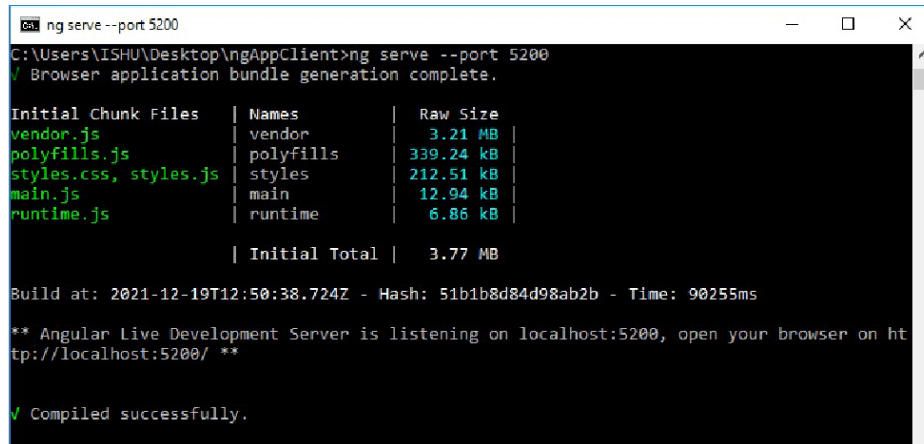


- We can now start our client application for the Node.js command line as:

```
ng serve.
```

- In case the Library project is running in the default port, 4200, we can change the port of the client app to 5200 by the following command:

```
ng serve --port 5200
```



```

C:\Users\ISHU\Desktop\ngAppClient>ng serve --port 5200
✓ Browser application bundle generation complete.

Initial Chunk Files | Names          | Raw Size
vendor.js           | vendor         | 3.21 MB
polyfills.js        | polyfills      | 339.24 kB
styles.css, styles.js | styles        | 212.51 kB
main.js             | main           | 12.94 kB
runtime.js          | runtime        | 6.86 kB
                    | Initial Total  | 3.77 MB

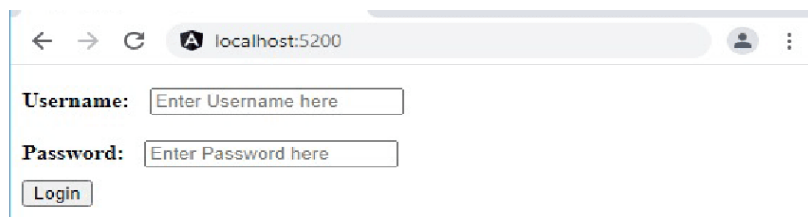
Build at: 2021-12-19T12:50:38.724Z - Hash: 51b1b8d84d98ab2b - Time: 90255ms

** Angular Live Development Server is listening on localhost:5200, open your browser on http://localhost:5200/ **

✓ Compiled successfully.

```

- Open the favorite browser and type the default Angular app URL to check the output: localhost:5200/.



Publishing your library

We publish the library to make the library available on npm. For publishing the library, all we need to do is create a production build, and then run npm publish command from the library project's dist directory.

The Syntax is as follows:

```

ng build <library name> --prod

cd dist/<library name>

npm publish

```


Here the Library project is **ngApp4library**, and the library is **my-lib**, so we run the following commands:

```
ng build my-lib --prod  
  
cd dist/my-lib  
  
npm publish
```

If you have not published anything before, you will need to create an npm account first and log in into your npm account and then publish your library.

The Purpose of Angular Libraries

Angular Libraries are very useful in case we want to re-use components, services, modules, etc. in our application. We just need to add the published library in our project, and that's it, the components, modules, services, are all ready to be used in the application.

To create a library, we generate it by “ng generate” command, built it by “ng build” command, publish by “[npm](#) publish” command. To use a library we install it by “ng i ” command.

Building an Angular Library for npm and for Local Use

For our current project, my software team is building an [Angular](#) library that can be used in a few different applications. We're creating these components in a separate project and using [Storybook](#) to test our components and gain confidence that they work correctly before being used in other applications.

We're deploying this library to [npm](#) for use in the other projects that use these components. But, we also want to be able to import the components locally to test them without having to re-deploy to npm for every little check.

Building for npm

Our [component library](#) project is laid as follows, excluding extraneous files and folders:



Our component library exists in the `projects/components` directory. The `public-api.ts` file contains exports for every module and component exposed by the library.

Deploying a new version of the component library to npm is accomplished by running an `npm run publish` script which we have set up in the root `package.json`. (Note that each command is on its own line for ease of readability. In an actual `package.json`, this should all be one long line.)

```
JSON
"publish": "npm install &&
cd projects/components &&
npm version patch &&
cd ../.. &&
npm run build &&
git commit -m bump projects/components/package.json &&
cd dist/components &&
npm publish"
```

This increments the patch version of the component library, builds the library, commits the new version, and publishes the component library to npm in one easy command. It also ensures that it happens correctly every time.

Building to Run Locally

What if you want to test a component in another project without pushing to npm? Often answers to, “How do I install a module locally with npm?” just tell you to run `npm install /path`. This should work if you’re writing plain JavaScript but won’t for our components which are written in TypeScript. We found that building and packaging the code allows us to then import it locally using these commands in the component library root run:

```
BASH
npm run build
cd dist/components
npm pack
```

This builds and packages your code into a compressed file named using the name and version from your `package.json`, e.g: `ui-library-0.0.1.tgz`. Then

you can navigate to the project you want to use the components in and run:

```
BASH
npm install /path/to/dist/ui-library-0.0.1.tgz
```

This will create an entry in your `package.json` which should be similar to:

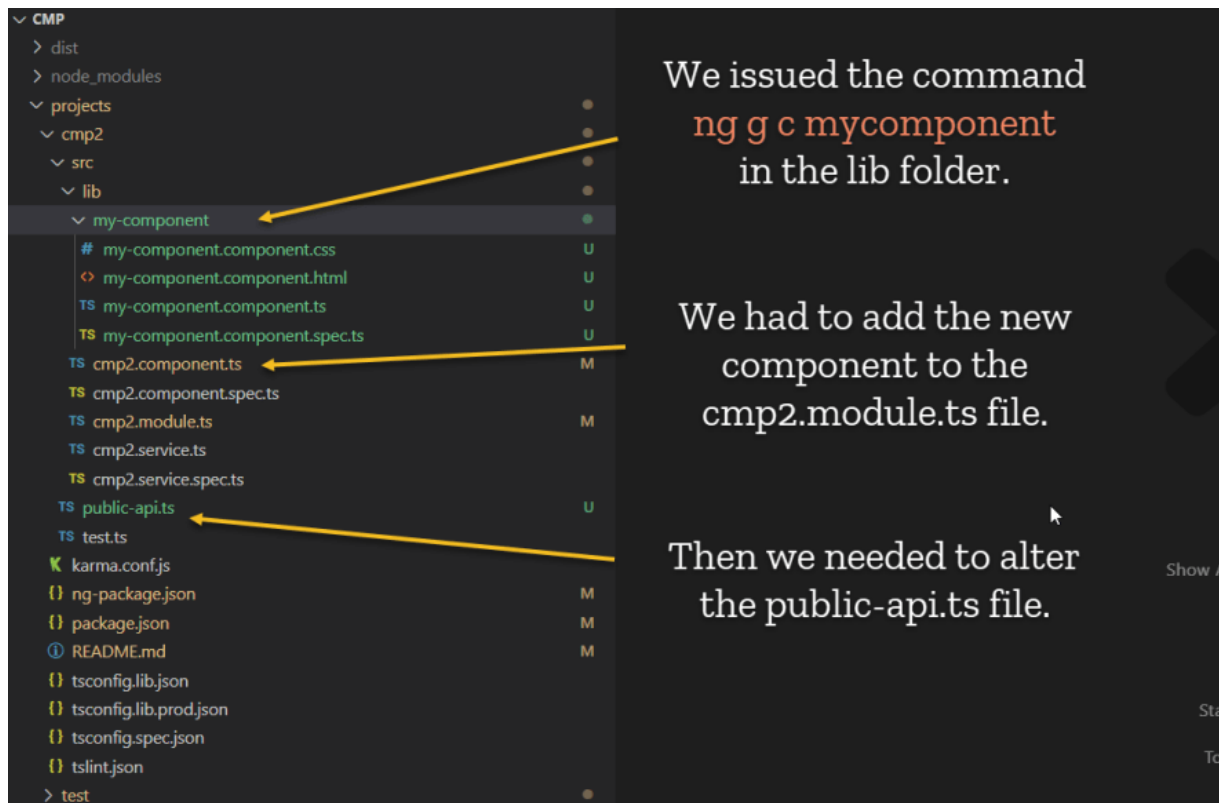
```
JSON
"dependencies": {
  "ui-library": "file:/path/to/dist/ui-library-0.0.1.tgz",
},
```

Angular Library Complete

You can now see your components running in another local application! This will help you test that both your components are working correctly in other environments and that your other project isn't breaking your component. That's great since both of these can happen frequently when building complex components in a separate project.

Angular Libraries : Adding New Components

In this post we are going to show how to add new components, compile and make your library ready for importing with all the new changes!



The new component's selector was:

```
@Component({
  selector: 'lib-my-component',
  templateUrl: './my-component.component.html',
  styleUrls: ['./my-component.component.css']
})
```

The `cmp2.module.ts` became:

```
import { NgModule } from "@angular/core";
import { Cmp2Component } from "../cmp2.component";
import { MyComponentComponent } from "../my-component/my-component.component";
```

```
@NgModule({
  declarations: [Cmp2Component, MyComponentComponent],
  imports: [],
  exports: [Cmp2Component, MyComponentComponent],
})
export class Cmp2Module {}
```

Finally, the `public-api.ts` became:

```
/*
 * Public API Surface of cmp2
 */

export * from "./lib/cmp2.service";
export * from "./lib/cmp2.component";
export * from "./lib/cmp2.module";
export * from "./lib/my-component/my-component.component";
```

Now we need to compile the library to pick up new changes.

Note

✓ When you compile libraries they compile with [AOT](#) compiler, this is why these components will run faster in your application.

```
PS C:\Users\con-petejo\source\repos\cmp> ng build cmp2
Building Angular Package
*****
It is not recommended to publish Ivy libraries to NPM repositories.
Read more here: https://v9.angular.io/guide/ivy#maintaining-library-compatibility
*****

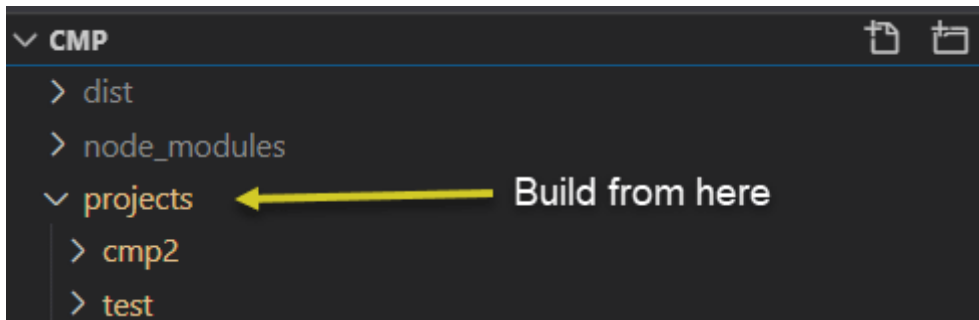
-----
Building entry point 'cmp2'
-----

Compiling TypeScript sources through ngc
Bundling to FESM2015
Bundling to FESM5
Bundling to UMD
Minifying UMD bundle
Writing package metadata
Built cmp2

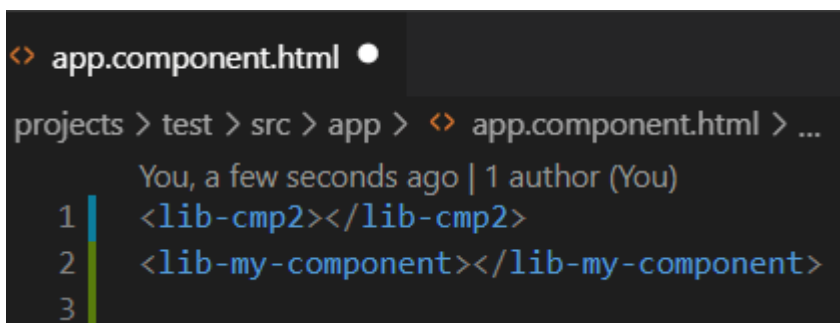
-----
Built Angular Package
- from: C:\Users\con-petejo\source\repos\cmp\projects\cmp2
- to:   C:\Users\con-petejo\source\repos\cmp\dist\cmp2
-----
```

To run the **ng build** command on your library, you need to be at the root of the folder structure.

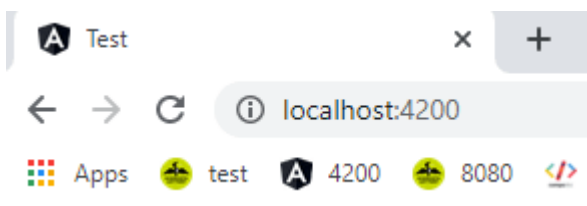
The **angular.json** file, located in that root directory, is used for **ng build**. It contains a Projects section with each project configuration. Ng build looks in there to determine how to compile the Projects listed there.



Add the HTML



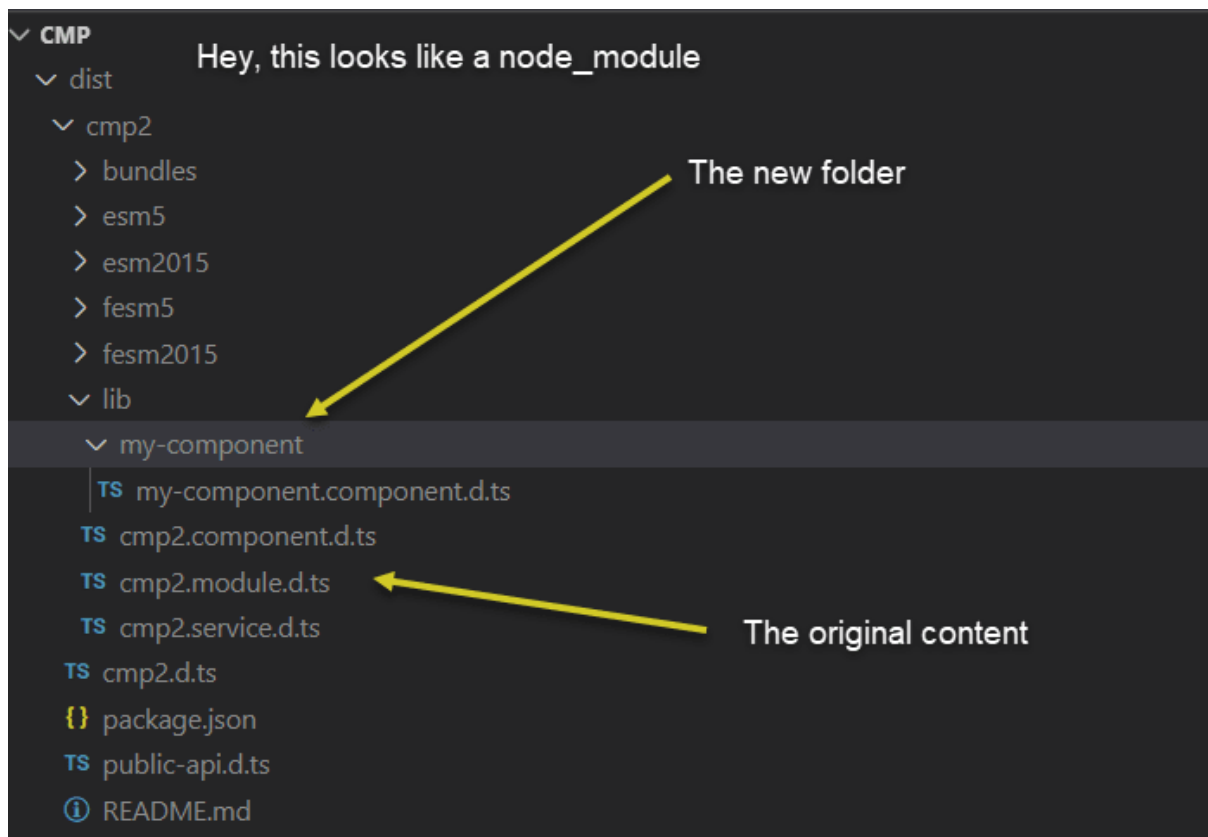
Save and observe the new changes.



cmp2 works!

my-component works!

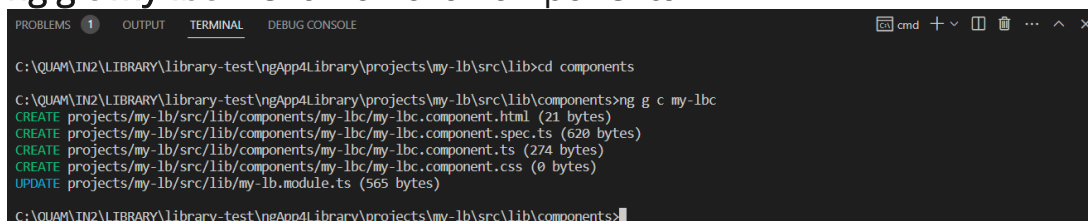
The DIST Folder



It pays in more ways than one to have importable modules from your own Angular custom library. You will eventually be able to double your production by just reusing components. Your customers will like the fact that the [time to render is vastly improved](#).

Dodavanje komponente na postojeći library

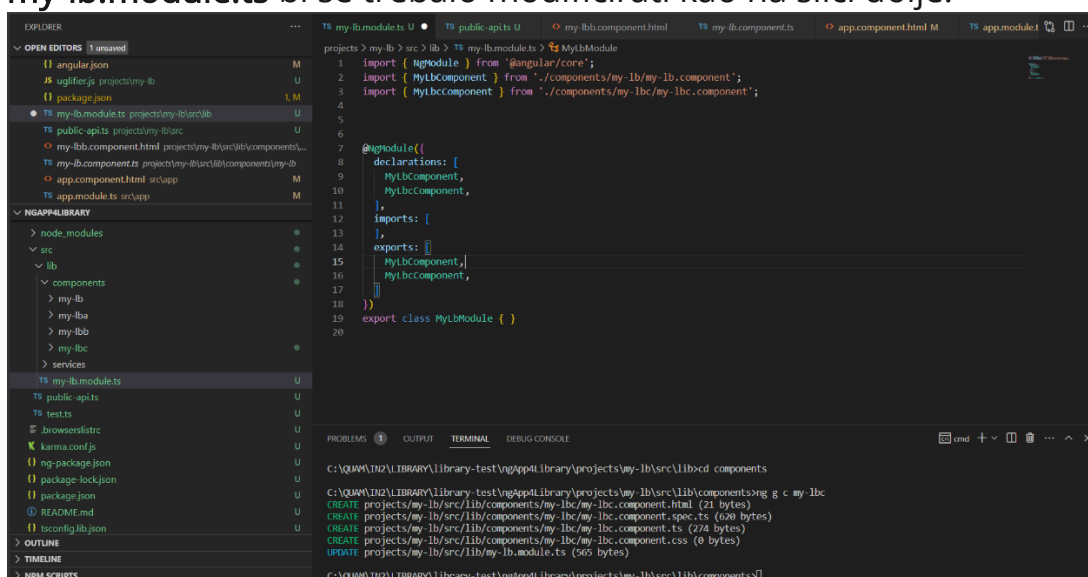
- Kreiramo direktorij `components` unutar `path` `ngApp4Library\projects\my-lb\src\lib` te premjestimo datoteke:
 - `my-lb.component.spec.ts` i
 - `my-lb.component.ts` u direktorij `.\components\my-lb`
- Pozicioniramo se na direktorij `path` `ngApp4Library\projects\my-lb\src\lib\components` i komandom: `ng g c my-lbc` kreiramo novu komponentu



```

C:\QUAM\IN2\LIBRARY\library-test\ngApp4Library\projects\my-lb\src\lib>cd components
C:\QUAM\IN2\LIBRARY\library-test\ngApp4Library\projects\my-lb\src\lib\components>ng g c my-lbc
CREATE projects\my-lb\src\lib\components\my-lbc\my-lbc.component.html (21 bytes)
CREATE projects\my-lb\src\lib\components\my-lbc\my-lbc.component.spec.ts (620 bytes)
CREATE projects\my-lb\src\lib\components\my-lbc\my-lbc.component.ts (274 bytes)
CREATE projects\my-lb\src\lib\components\my-lbc\my-lbc.component.css (0 bytes)
UPDATE projects\my-lb\src\lib\my-lb.module.ts (565 bytes)
C:\QUAM\IN2\LIBRARY\library-test\ngApp4Library\projects\my-lb\src\lib\components>
  
```

- `my-lb.module.ts` bi se trebalo modificirati kao na slici dolje:

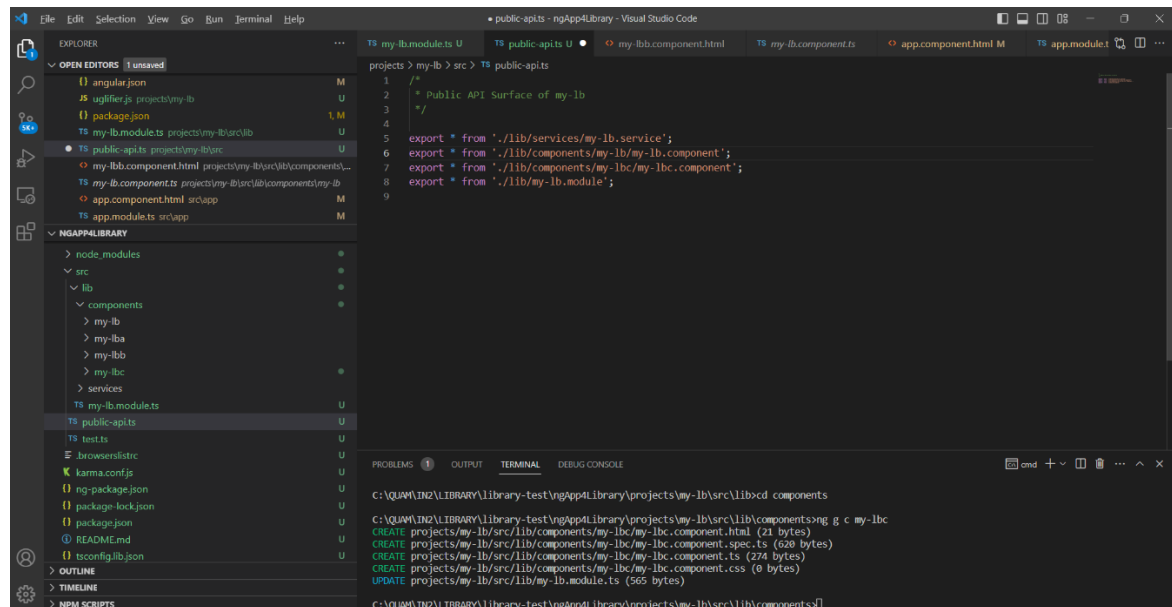


```

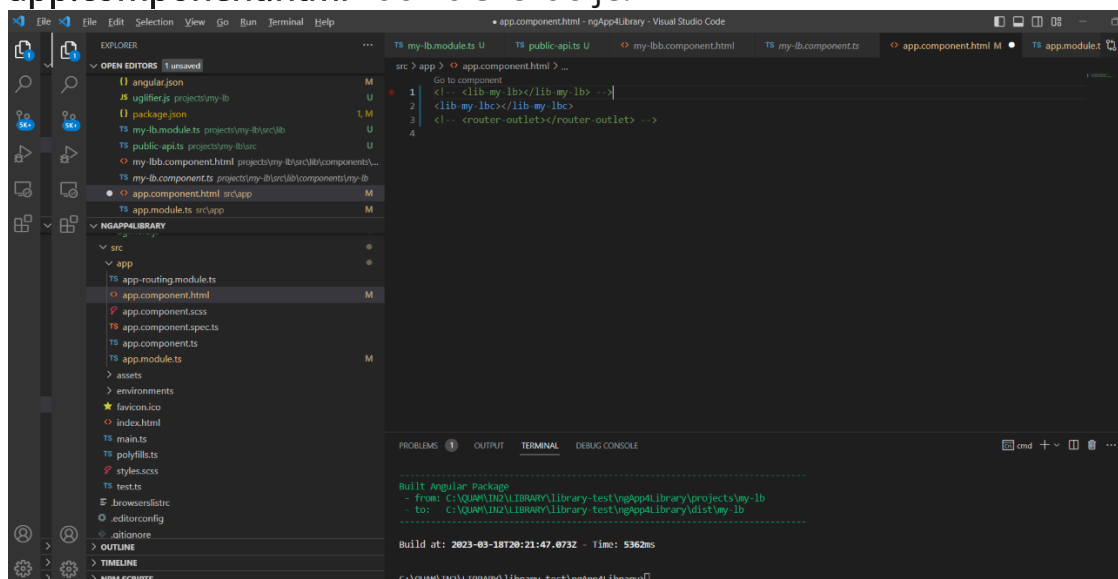
import { NgModule } from '@angular/core';
import { MyLbComponent } from './components/my-lb/my-lb.component';
import { MyLbcComponent } from './components/my-lbc/my-lbc.component';

@NgModule({
  declarations: [
    MyLbComponent,
    MyLbcComponent,
  ],
  imports: [
  ],
  exports: [
    MyLbComponent,
    MyLbcComponent,
  ],
})
export class MyLbModule {}
  
```

- Time je library definiran sa postojećom i jednom novom komponentom – još jedino moramo te iste komponente postaviti „dostupnima“ izvan library-ja izmjenom datoteke `public-api.ts` kao na slici dolje:



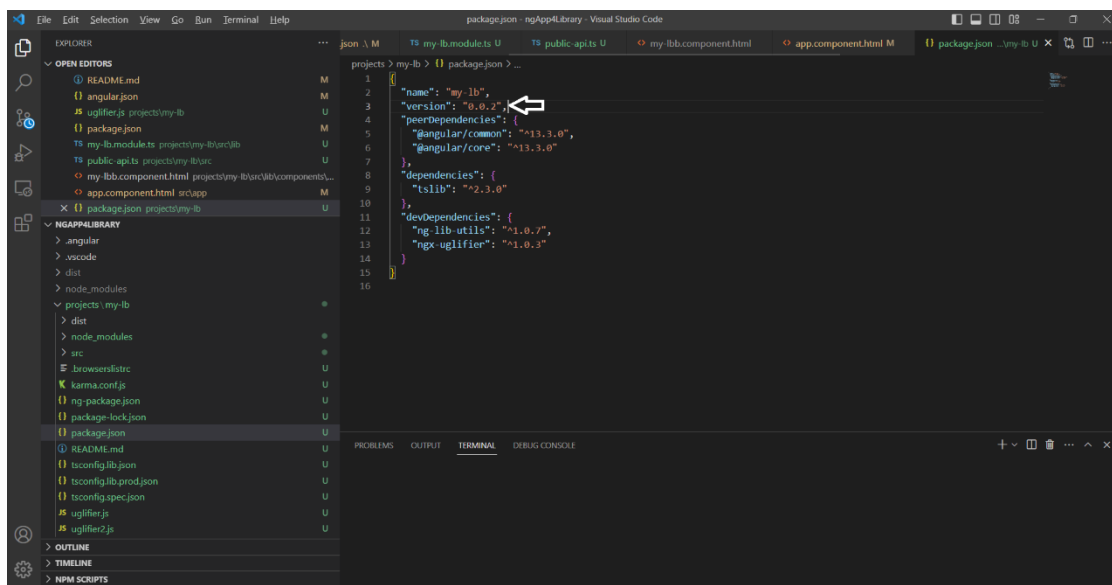
- Obzirom da smo u prethodnom poglavlju objasnili kako se dodaje nova komponenta za library sa napomenom:
Now we need to compile the library to pick up new changes.
Imajući u vidu da se ovdje radi o library-ju (externom projektu) kako bi komponente bile vidljive ne samo prilikom kreiranja nove komponente već i za svaku izmjenu – **MORAMO IZVRŠITI BUILD:**
ng build my-lib
- Tek sada smo spremni u master projektu izmijeniti datoteku **app.component.html** kao na slici dolje:



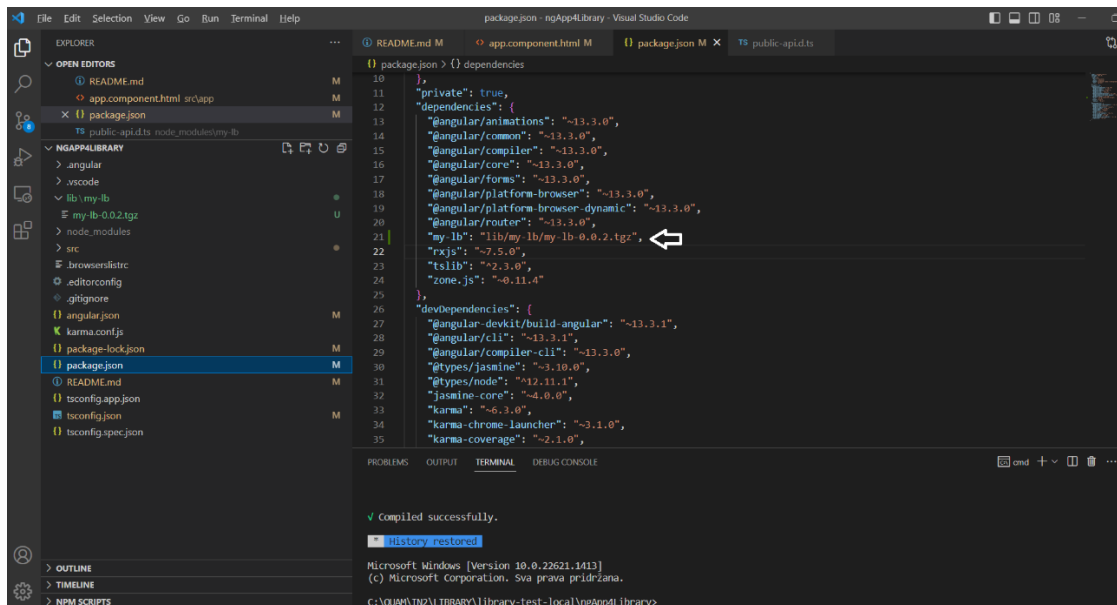
- ng serve** komandom ispravno se unutar preglednika prikazuje nova komponenta

Distribucija package my-lb

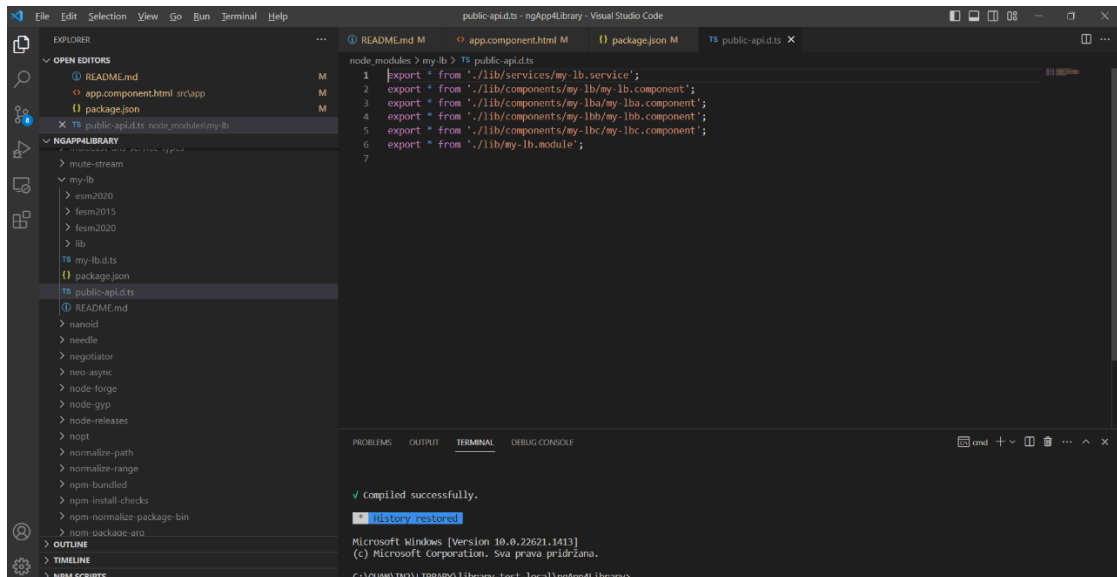
- Prije distribucije library-ja moramo obavezno promijeniti verziju build-a, naime npm cache-ira distribucije te nakon izmjene vraća „stare“ verzije, da bismo to izbjegli package se određuje prema verziji build-a (wiki bi trebao sadržavati detaljnije informacije za svaki build).



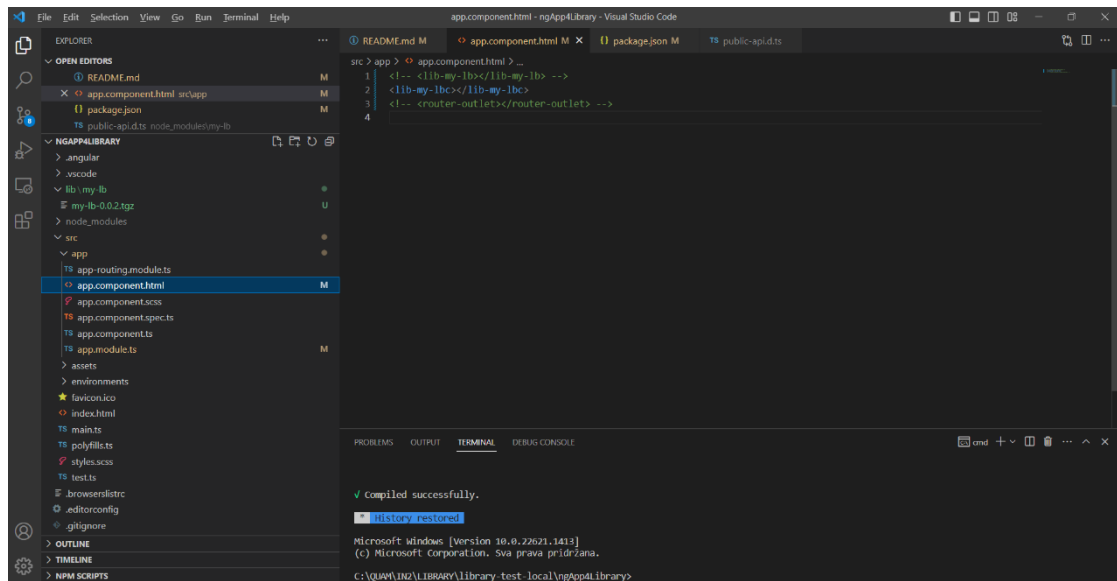
- Zatim slijedi build: **npm build my-lb --configuration production**
- Ukoliko je build prošao bez problema – sljedeći korak je izrada **.tgz** datoteke za distribuciju.
- Pozicioniramo se na **path\ngApp4Library\dist\my-lb** i izvršimo komandu: **npm pack**
- Generiranu datoteku: **my-lb-0.0.2.tgz** kopiramo u direktorij za distribuciju:
path\ngApp4LibraryImplementation\lib\my-lb\my-lb-0.0.2.tgz
- U datoteci **package.json** ažuriramo novu verziju kao na slici dolje – obzirom da „stara“ verzija sadrži u **node_modules** direktoriju verziju 0.0.1 uputno je obrisati cijeli direktorij (node_modules) te ponovno pokrenuti **npm i**



- Provjerimo da li datoteka `public-api.d.ts` sadrži nove definicije library-ja u `node_modules\my-lb` direktoriju:



- Ukoliko smo ispravno ažurirali library – možemo napraviti izmjene u datoteci `app.component.html` kao na slici dolje



- **ng serve** komanda bi trebala ispravno renderirati ažurirane izmjene na library-ju.