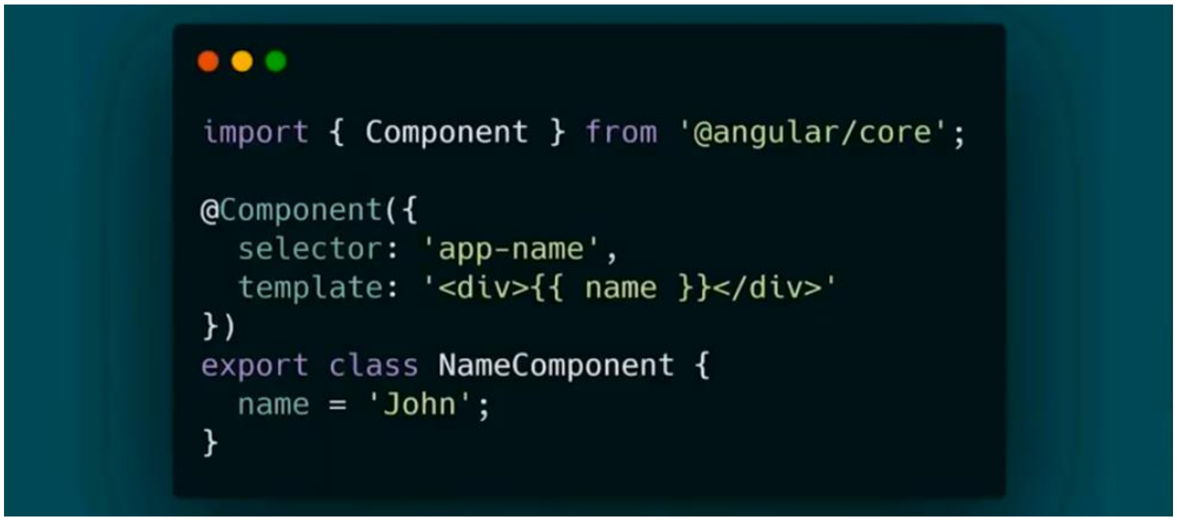
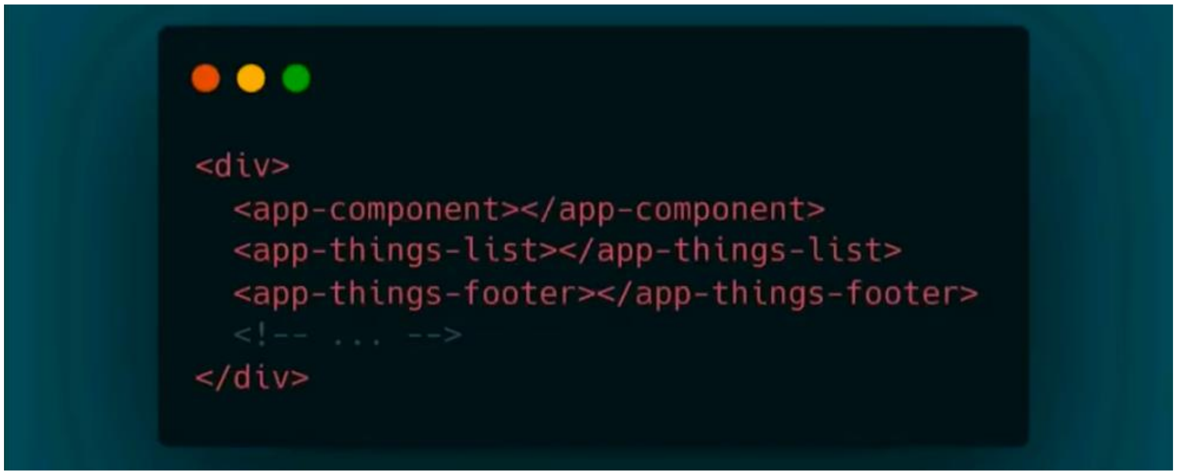
ARCHITECTING ANGULAR APPS
W/ LIBRARIES
Fabian Gosebrink
Architecting Angular Applications with Angular Libraries

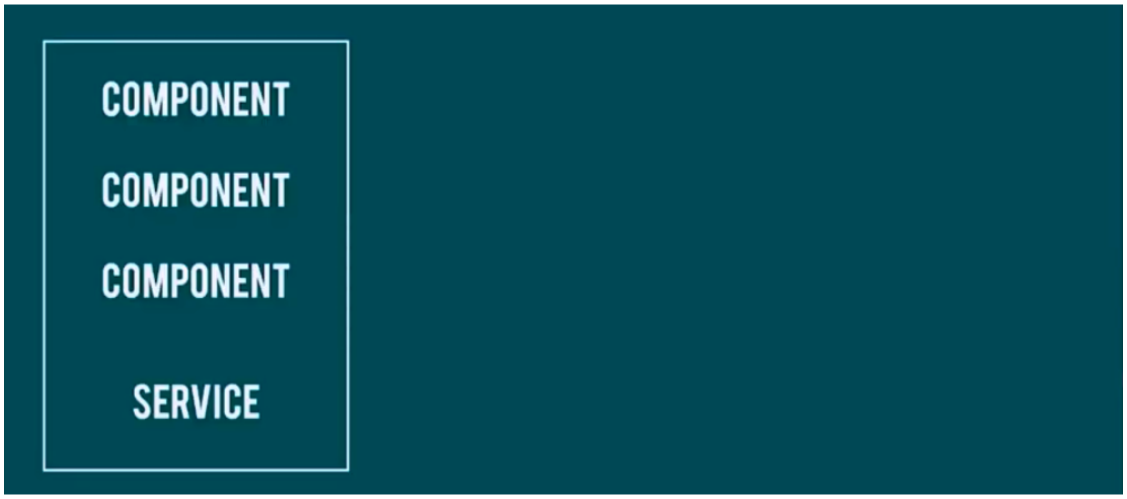
Angular offers a large ecosystem when it comes to separation and architecture of your application. There are often pieces of code that you don't just want to reuse within your application, but to make available to other applications in your organization or via **package managers** like **npm** over the Internet. This is where angular libraries come into play. In this talk, Fabian Gosebrink explores the way Angular Libraries are built, what the Angular Package format is good for, and how we can move code from an existing application to an Angular Library to reuse the code across multiple applications. This makes scaling and the architecture of angular applications a breeze.



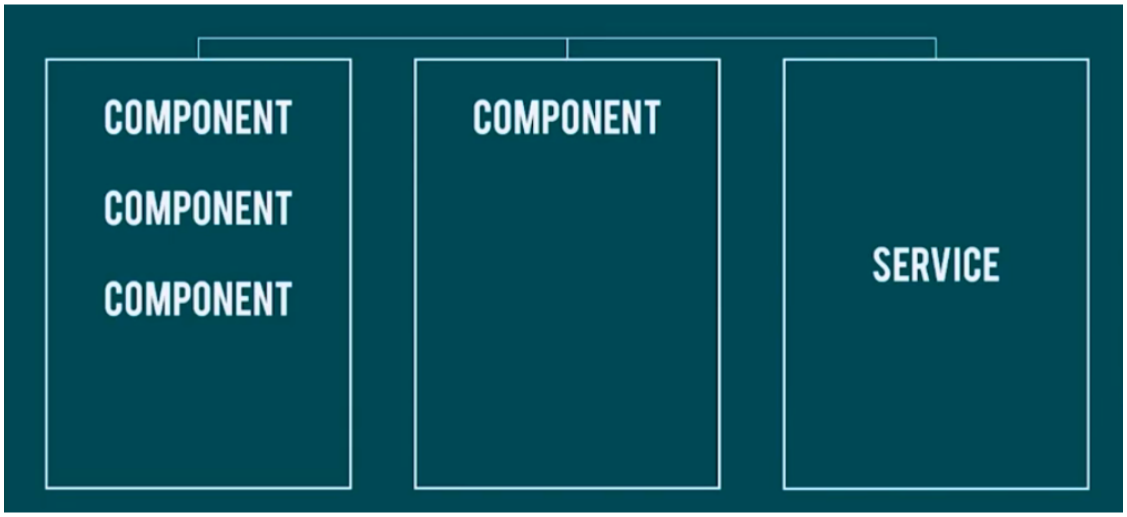
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-name',
  template: '<div>{{ name }}</div>'
})
export class NameComponent {
  name = 'John';
}
```



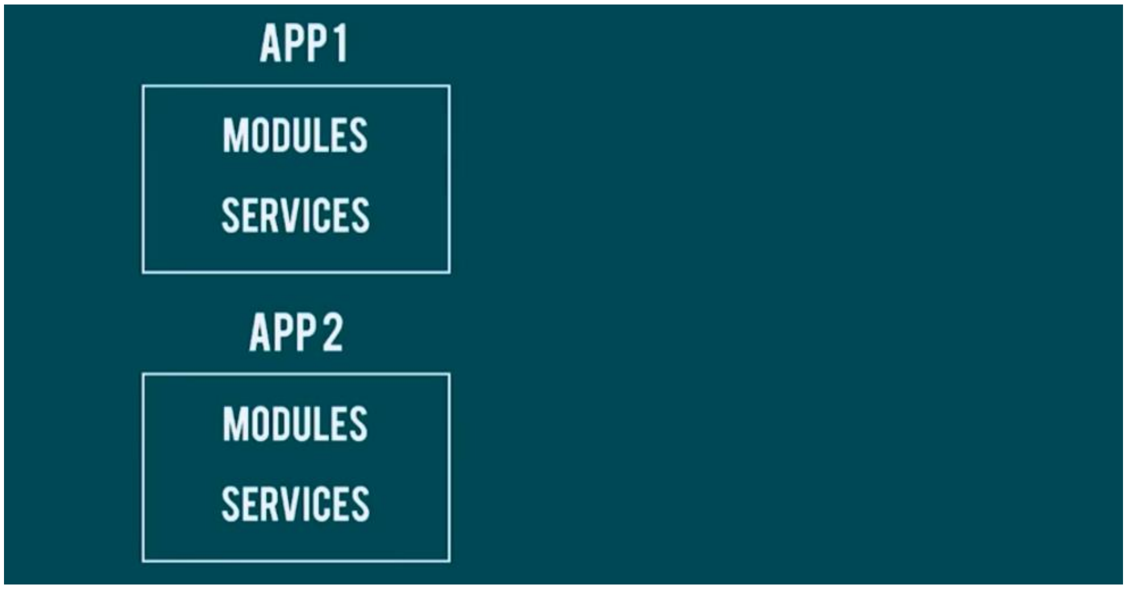
```
<div>
  <app-component></app-component>
  <app-things-list></app-things-list>
  <app-things-footer></app-things-footer>
  <!-- ... -->
</div>
```

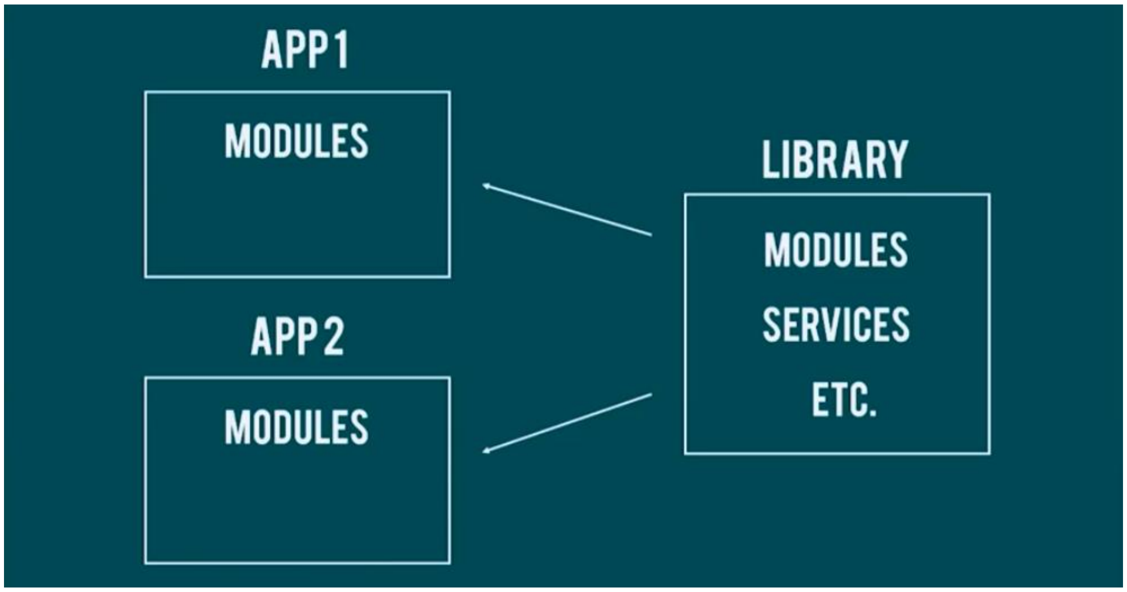


We can also organize our components and services into modules and use injections as below



This works well for a single application, what do we do with sharing code between different applications as below?





We can use libraries as above

REUSABILITY

COMPLEXITY

REQUIREMENTS

Let us see the requirements of an Angular library

PLATFORM INDEPENDENT

Don’t refer to any DOM specific things

BUNDLED

Bundle into the smallest library file before shipping

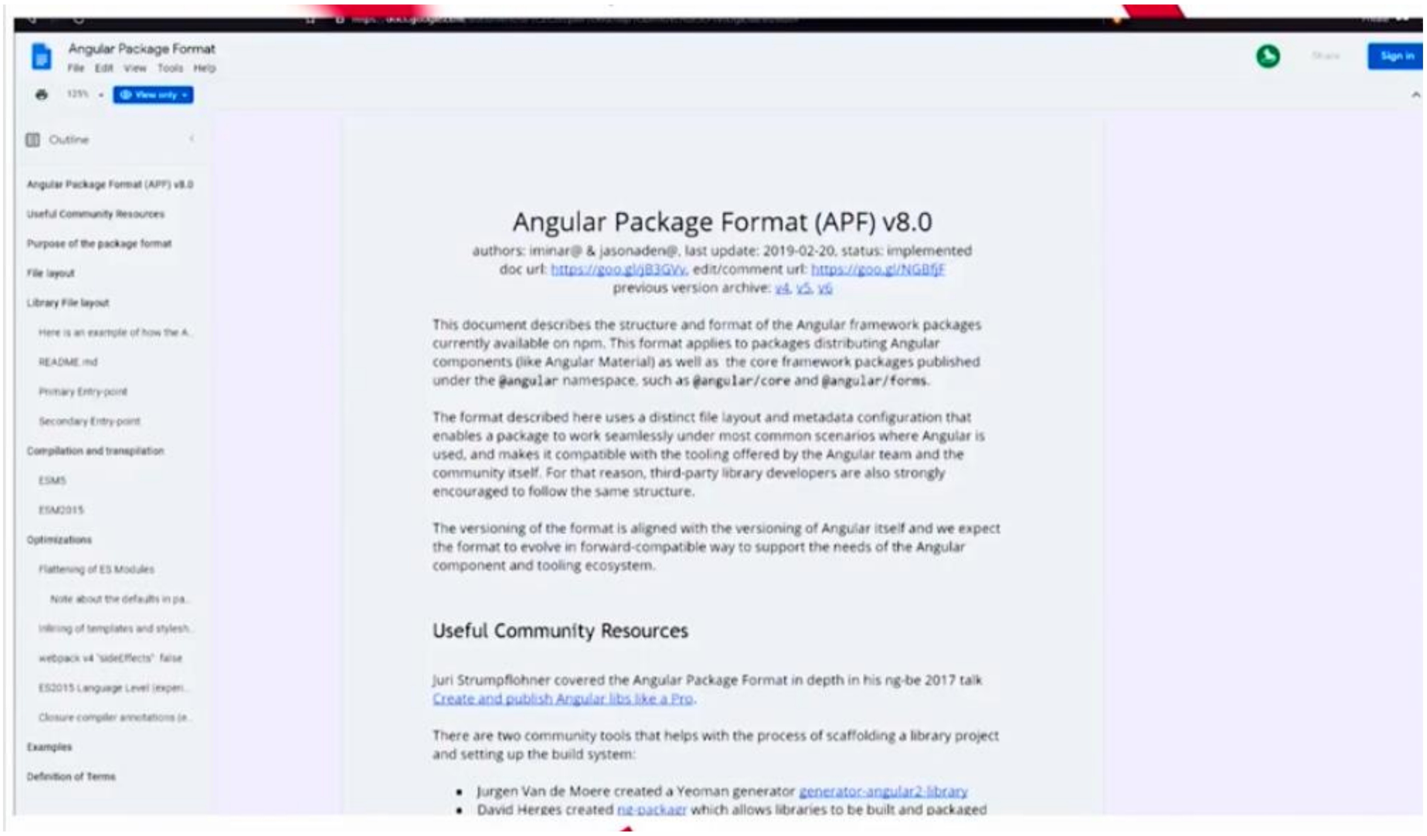
AOT

Should be AOT compiled so that the consumer doesn’t need the AOT compiler to use our library

TYPESCRIPT

ANGULAR PACKAGE FORMAT

This is the recommended way to distribute packages, it is a Google Doc contract



Angular Package Format

File Edit View Tools Help

125% View only

Outline

Angular Package Format (APF) v8.0

authors: iminar@ & jasonaden@, last update: 2019-02-20, status: implemented
doc url: https://goo.gl/jB3GVv, edit/comment url: https://goo.gl/NGBfjF
previous version archive: v4, v5, v6

This document describes the structure and format of the Angular framework packages currently available on npm. This format applies to packages distributing Angular components (like Angular Material) as well as the core framework packages published under the `@angular` namespace, such as `@angular/core` and `@angular/forms`.

The format described here uses a distinct file layout and metadata configuration that enables a package to work seamlessly under most common scenarios where Angular is used, and makes it compatible with the tooling offered by the Angular team and the community itself. For that reason, third-party library developers are also strongly encouraged to follow the same structure.

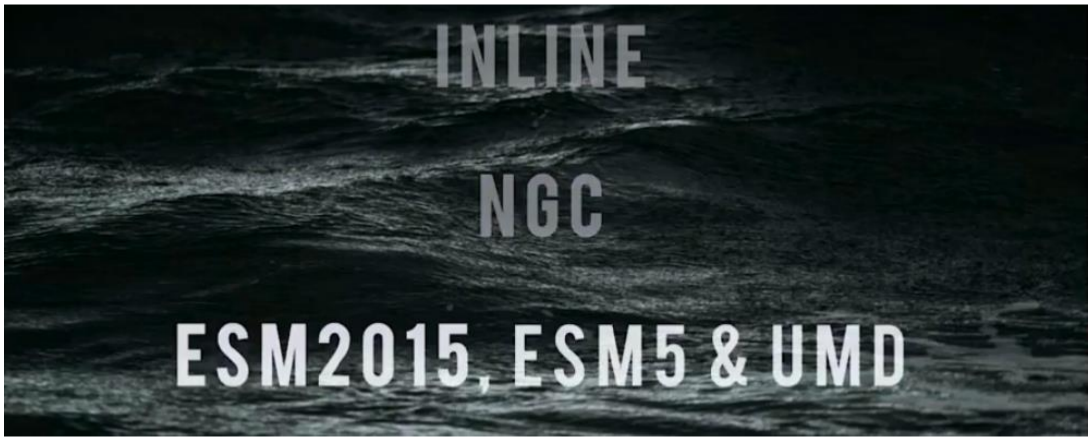
The versioning of the format is aligned with the versioning of Angular itself and we expect the format to evolve in forward-compatible way to support the needs of the Angular component and tooling ecosystem.

Useful Community Resources

Juri Strumpflohner covered the Angular Package Format in depth in his ng-be 2017 talk Create and publish Angular libs like a Pro.

There are two community tools that helps with the process of scaffolding a library project and setting up the build system:

- Jurgen Van de Moere created a Yeoman generator generator-angular2-library
- David Herges created ng-packagr which allows libraries to be built and packaged



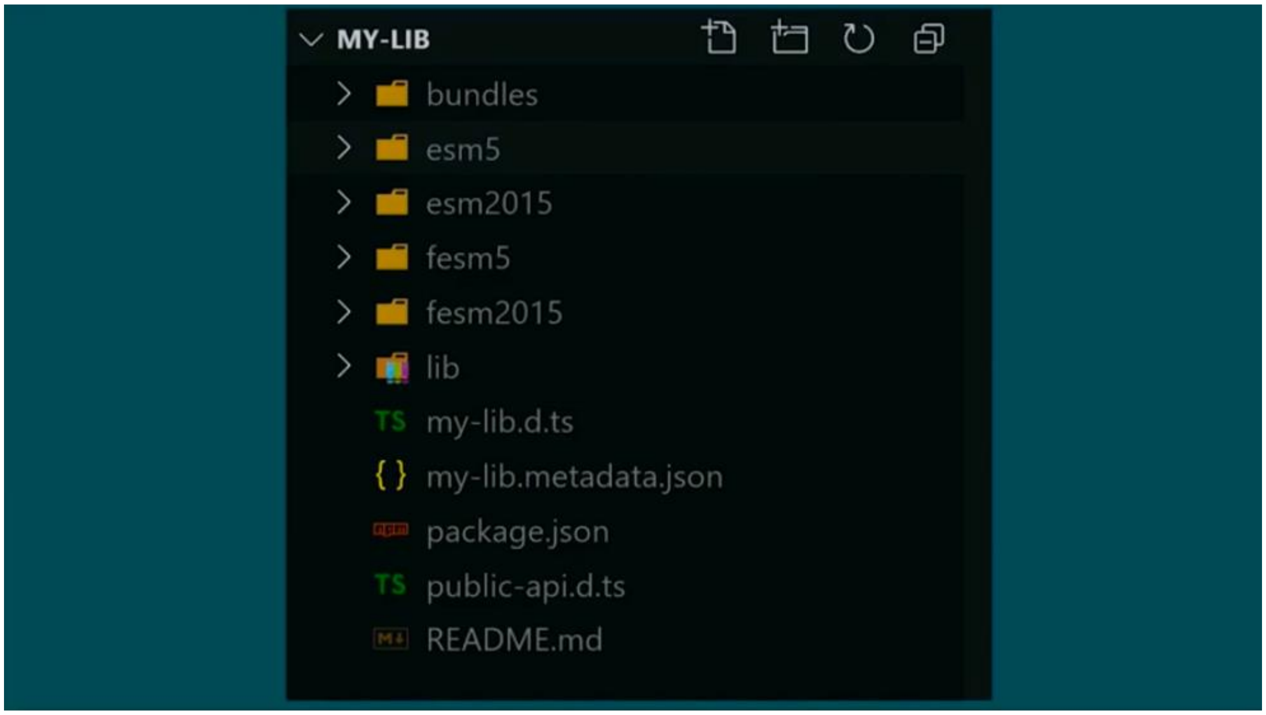
INLINE

NGC

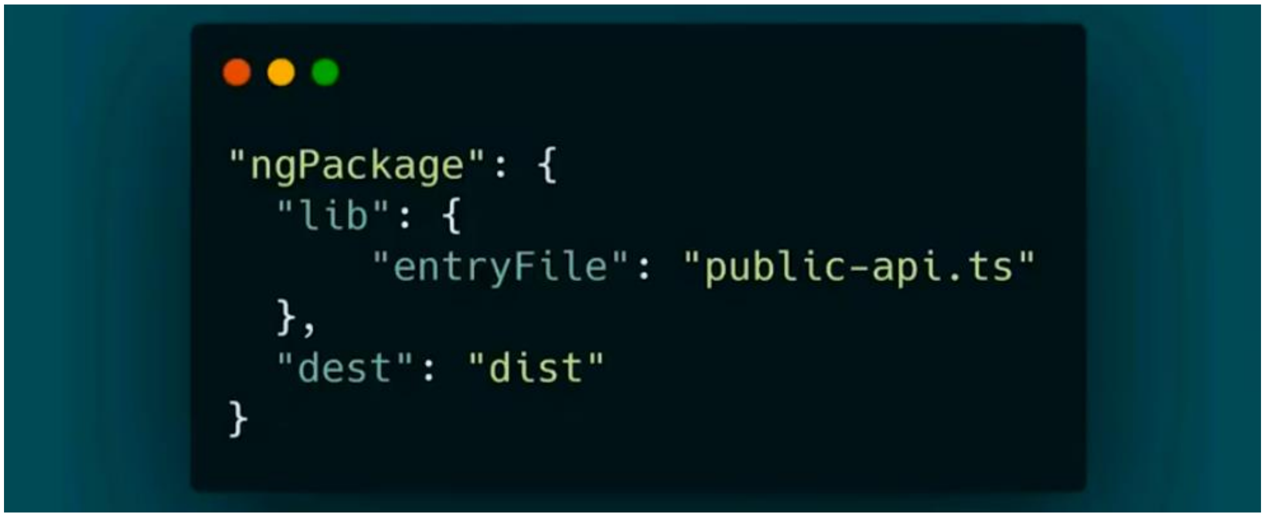
ESM2015, ESM5 & UMD

What are the steps to create an Angular library?

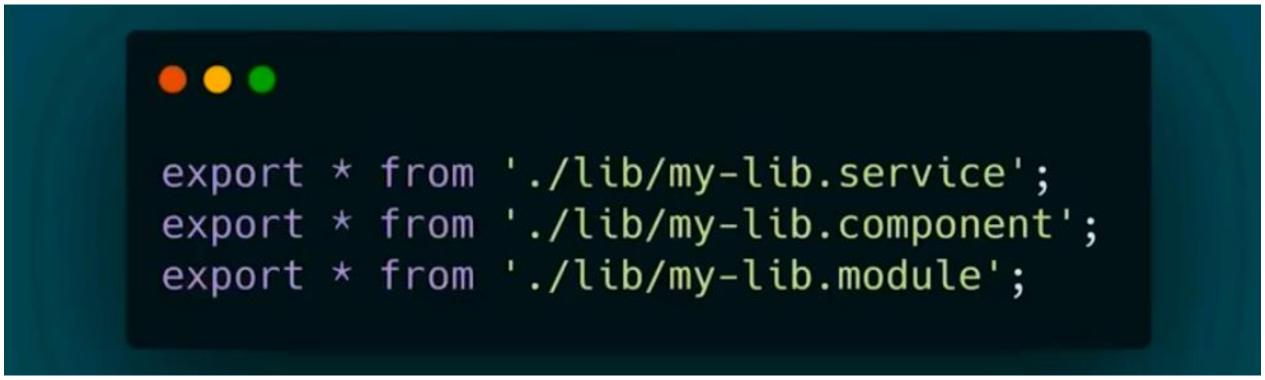
NG PACKAGR



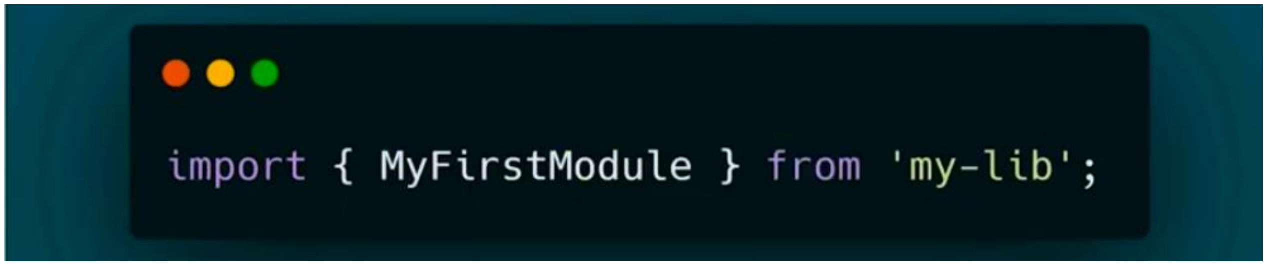
```
MY-LIB
  bundles
  esm5
  esm2015
  fesm5
  fesm2015
  lib
  my-lib.d.ts
  my-lib.metadata.json
  package.json
  public-api.d.ts
  README.md
```



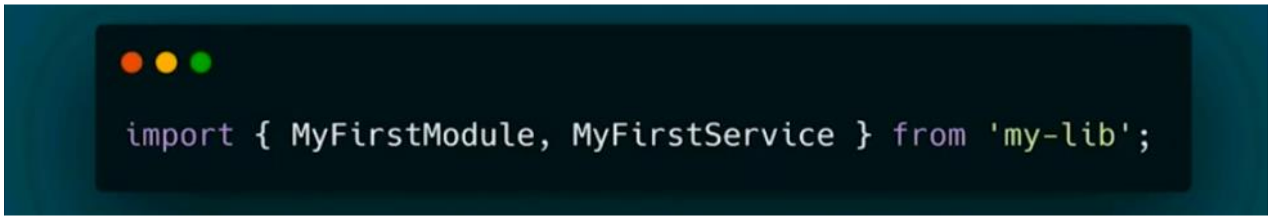
```
"ngPackage": {
  "lib": {
      "entryFile": "public-api.ts"
  },
  "dest": "dist"
}
```



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

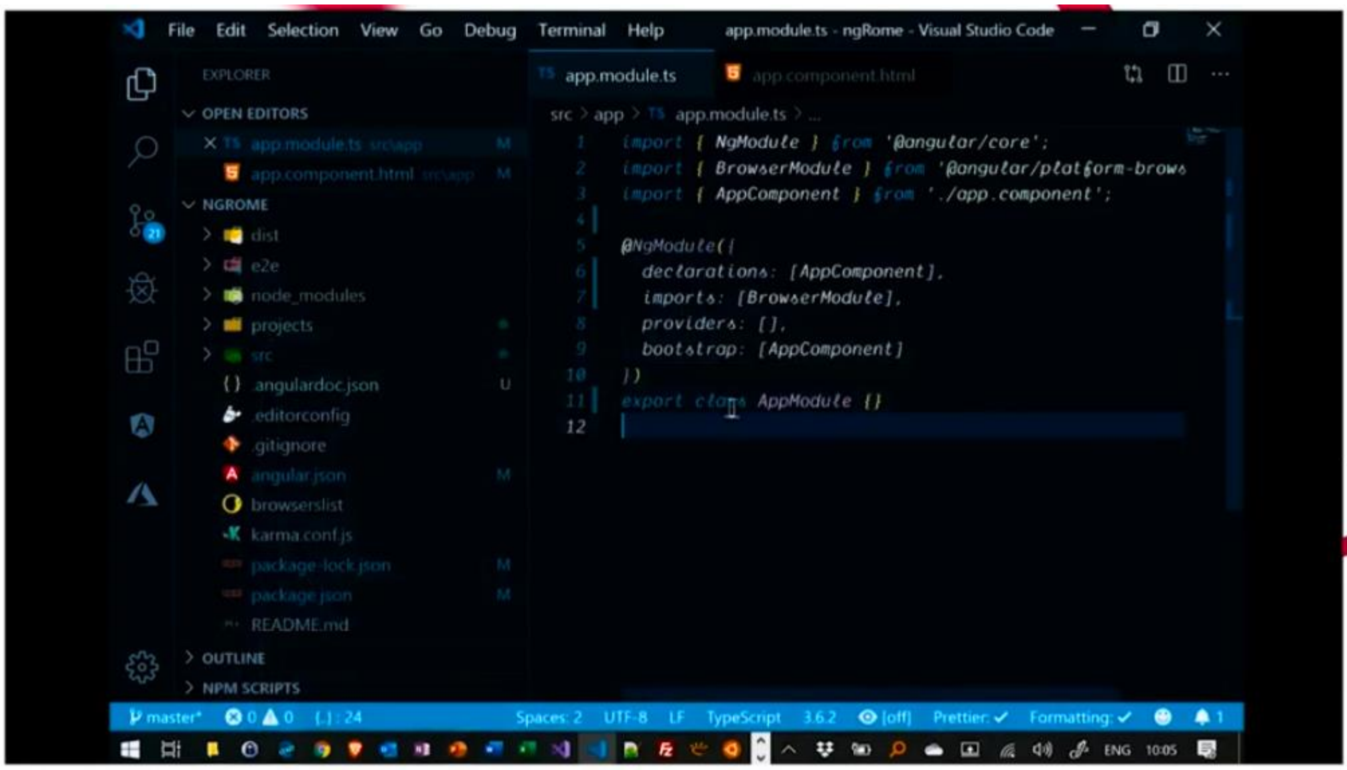


```
import { MyFirstModule } from 'my-lib';
```



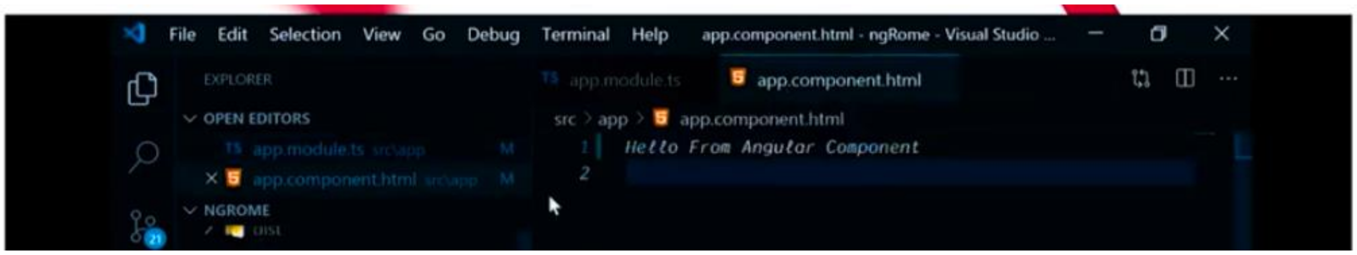
```
import { MyFirstModule, MyFirstService } from 'my-lib';
```

CODE



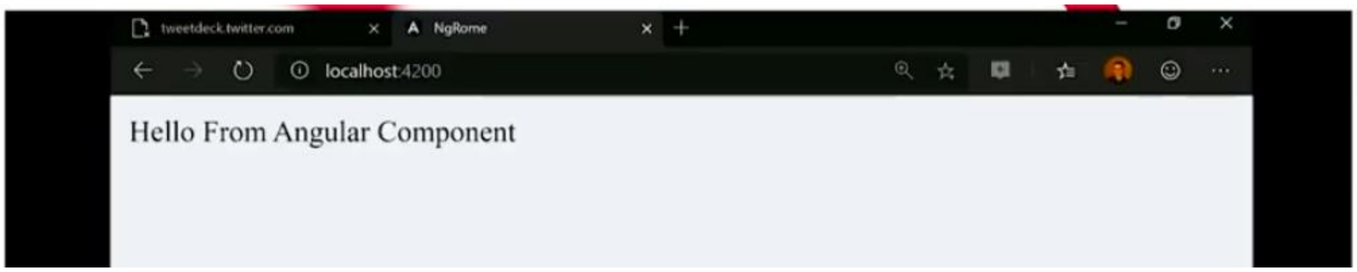
File Edit Selection View Go Debug Terminal Help
app.module.ts - ngRome - Visual Studio Code
EXPLORER
OPEN EDITORS
app.module.ts src\app
app.component.html src\app
NGROME
dist
e2e
node_modules
projects
src
angulardoc.json
.editorconfig
.gitignore
angular.json
browserslist
karma.conf.js
package-lock.json
package.json
README.md
OUTLINE
NPM SCRIPTS
src > app > app.module.ts > ...
master* Spaces: 2 UTF-8 LF TypeScript 3.6.2 Prettier: ✓ Formatting: ✓

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-brows
import { AppComponent } from './app.component';

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

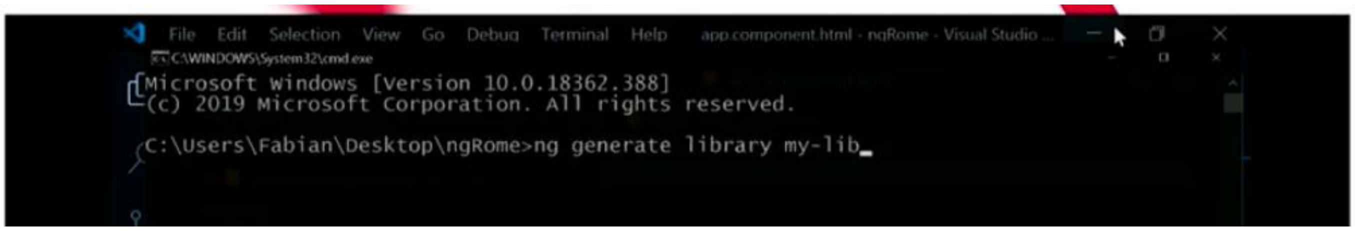


File Edit Selection View Go Debug Terminal Help
app.component.html - ngRome - Visual Studio ...
EXPLORER
OPEN EDITORS
app.module.ts src\app
app.component.html src\app
NGROME
dist
src > app > app.component.html

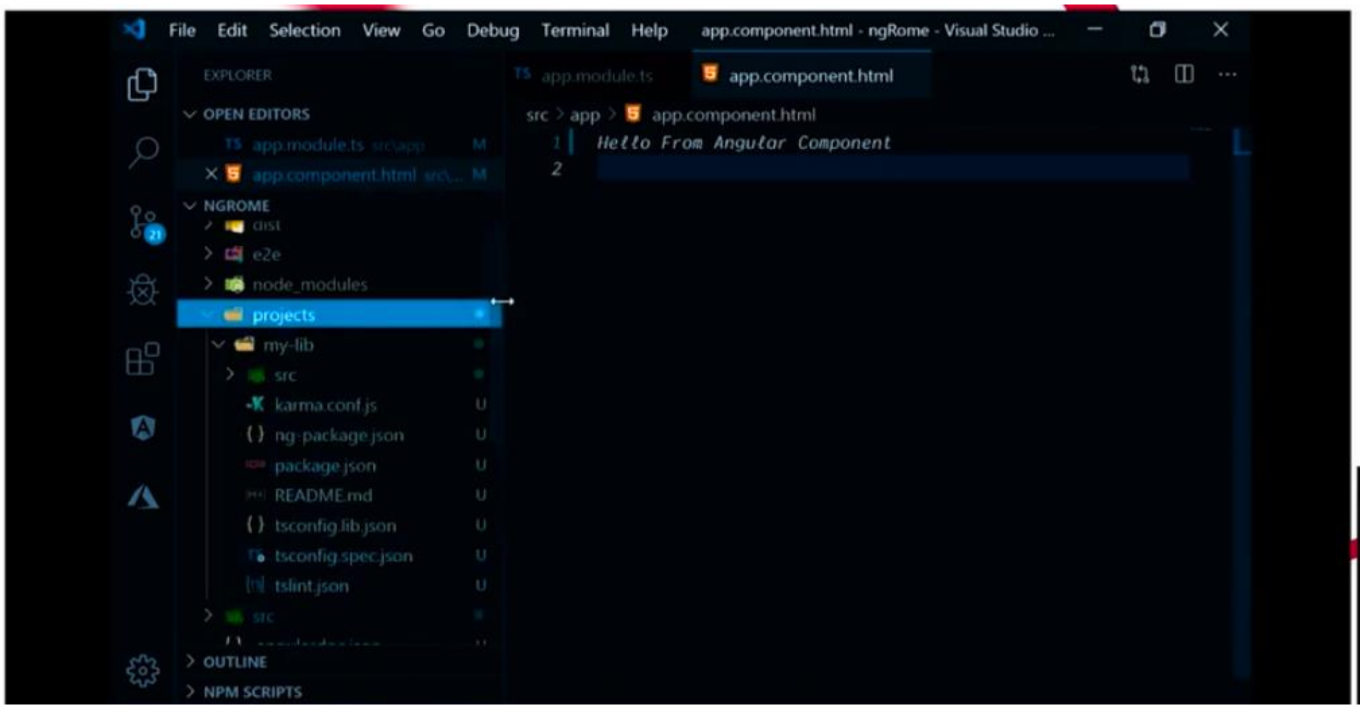
```
Hello From Angular Component
```



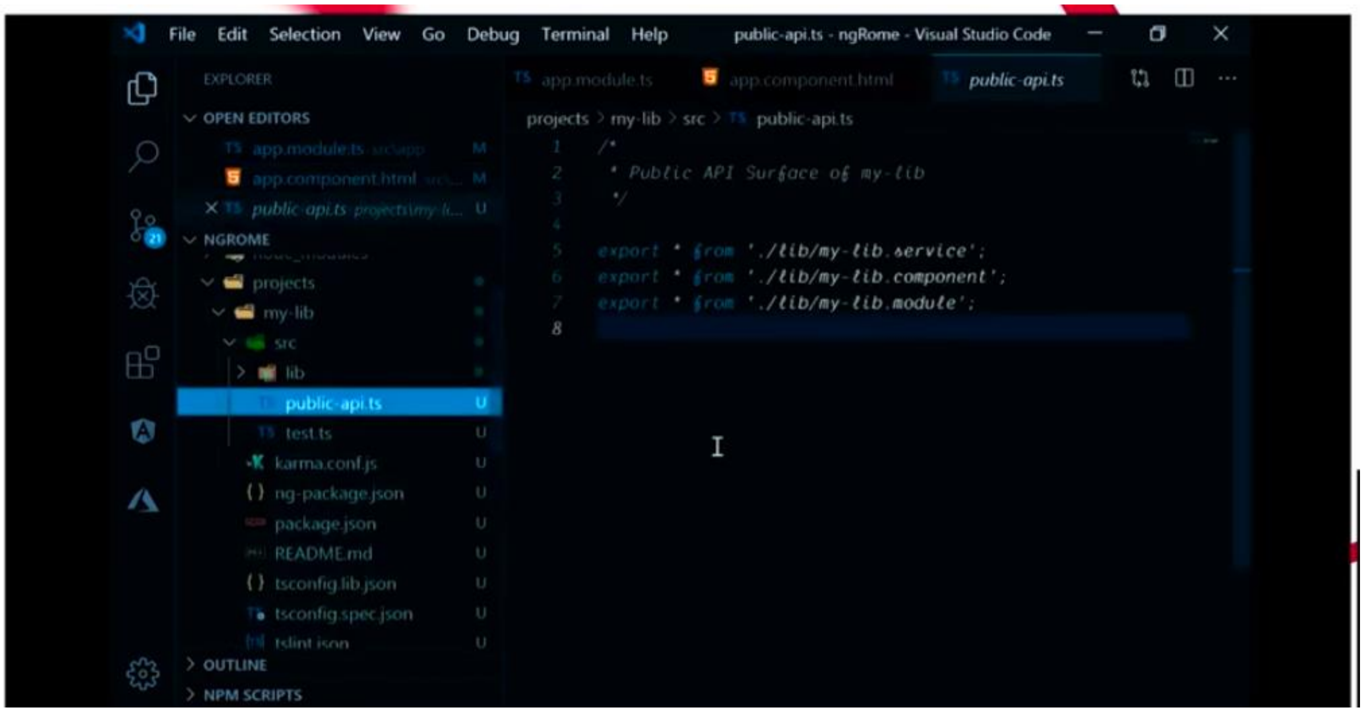
Hello From Angular Component



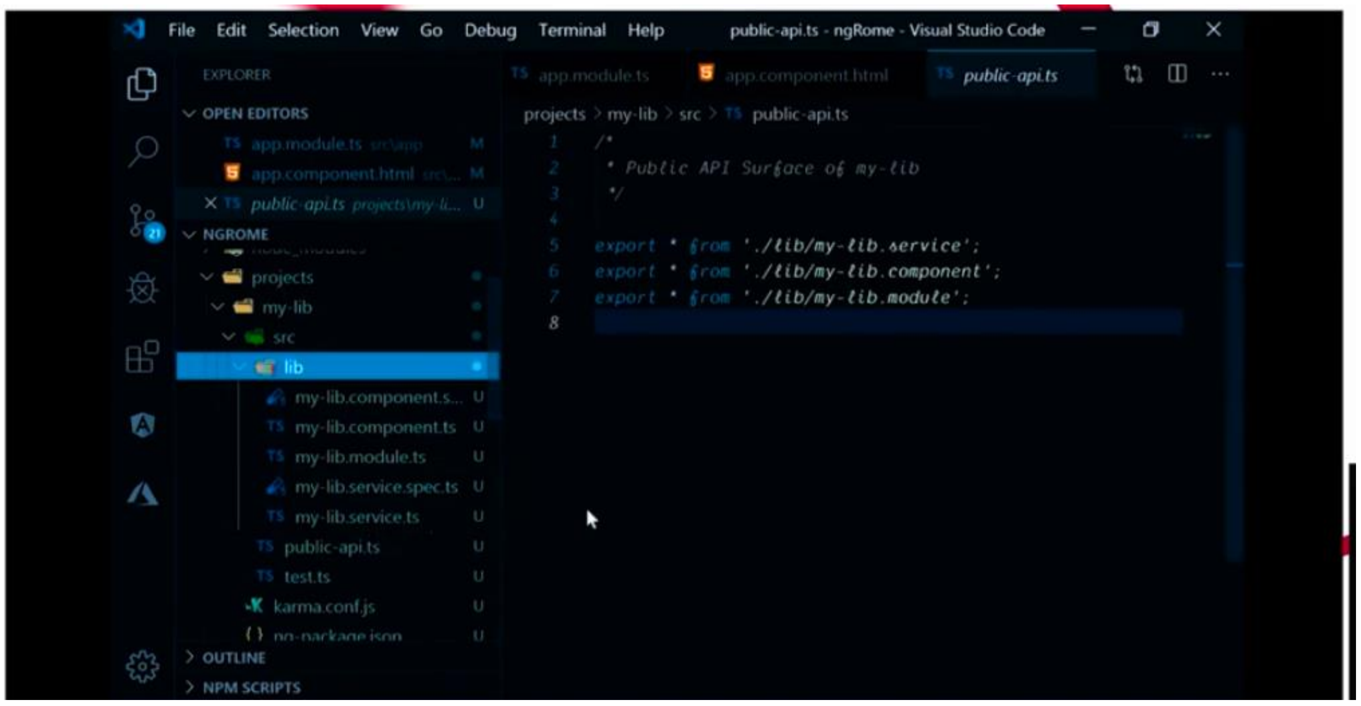
```
Microsoft Windows [Version 10.0.18362.388]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Fabian\Desktop\ngRome>ng generate library my-lib
```



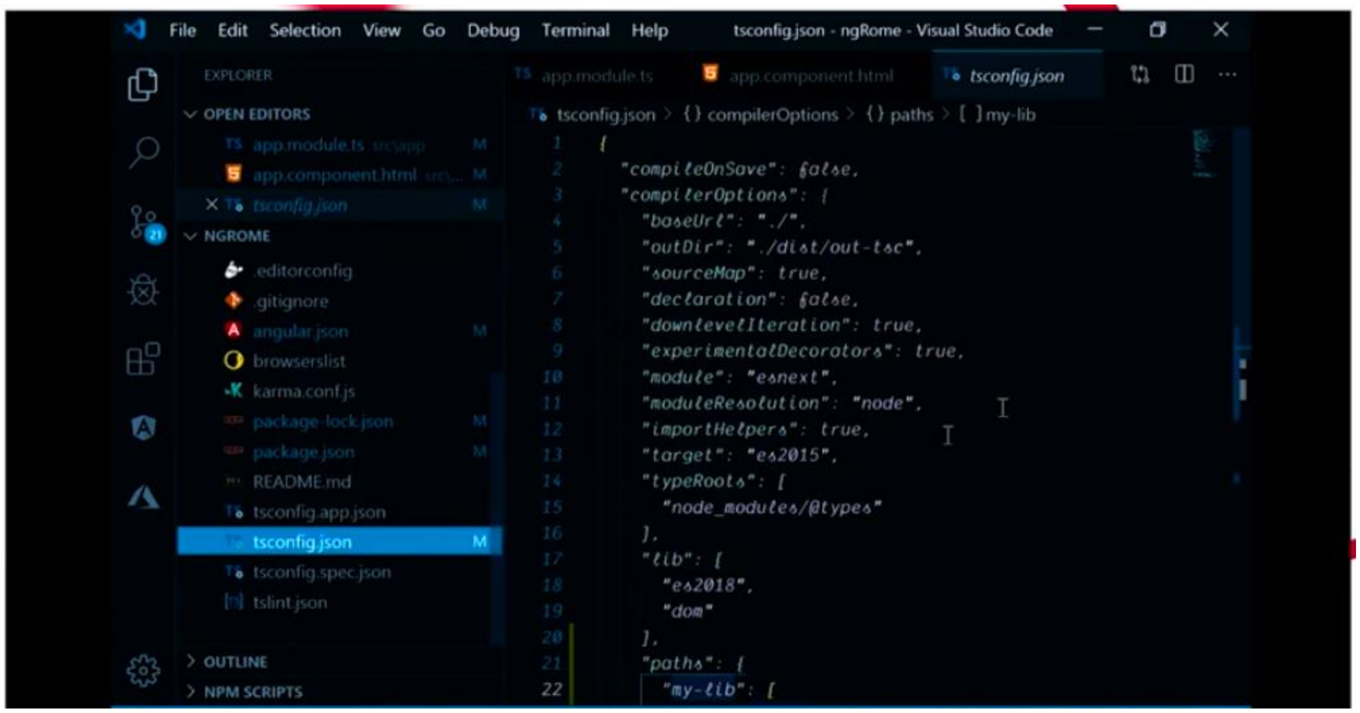
```
Hello From Angular Component
```



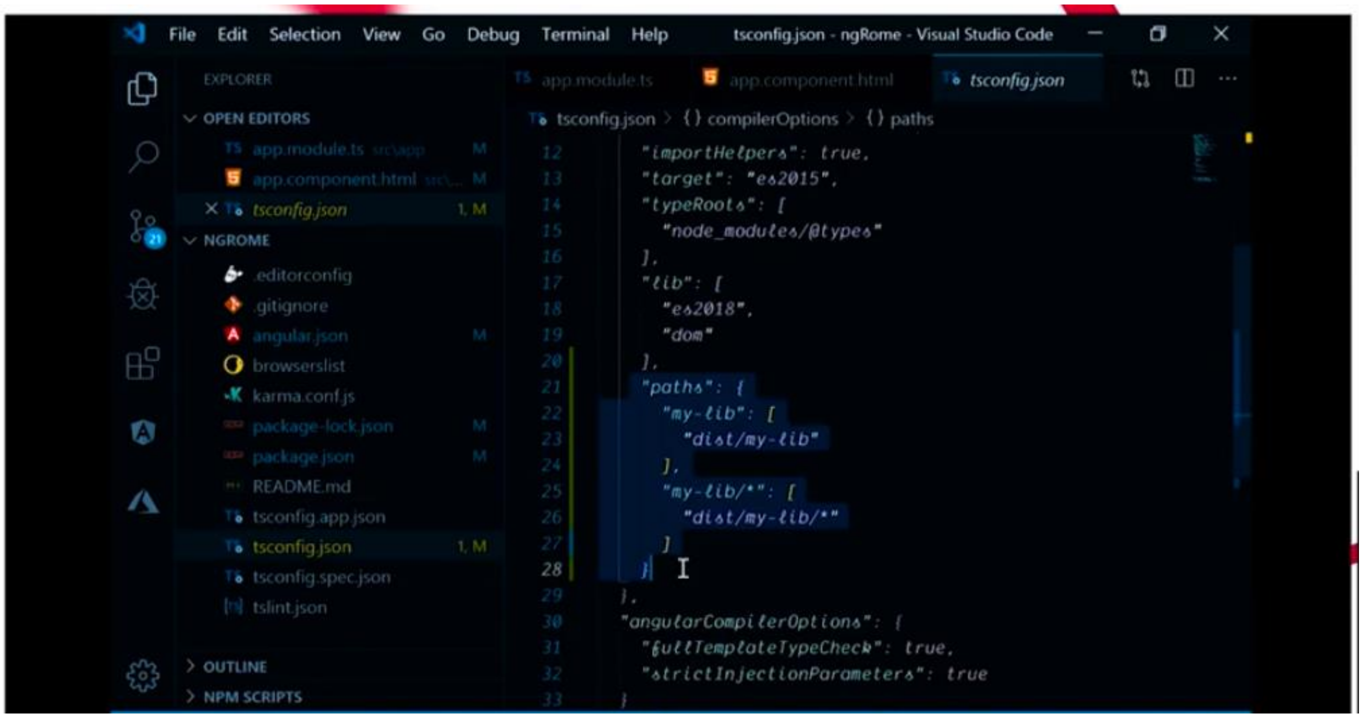
```
/*
 * Public API Surface of my-lib
 */

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



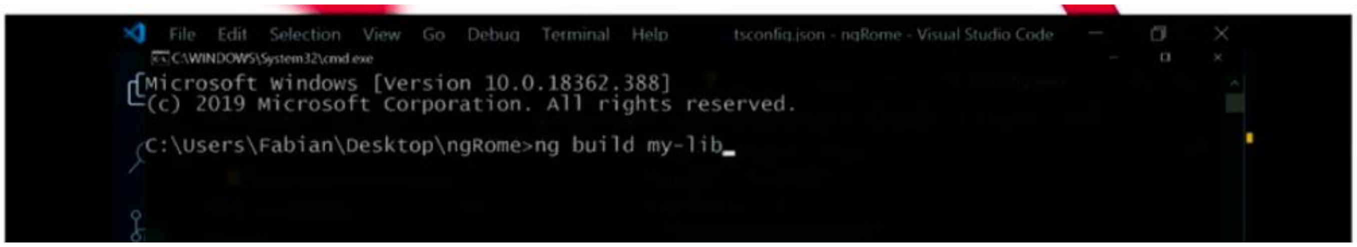
```
/*
 * Public API Surface of my-lib
 */

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



```
{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./",
    "outDir": "./dist/out-tsc",
    "sourceMap": true,
    "declaration": false,
    "downlevelIteration": true,
    "experimentalDecorators": true,
    "module": "esnext",
    "moduleResolution": "node",
    "importHelpers": true,
    "target": "es2015",
    "typeRoots": [
      "node_modules/@types"
    ],
    "lib": [
      "es2018",
      "dom"
    ],
    "paths": {
      "my-lib": [
```

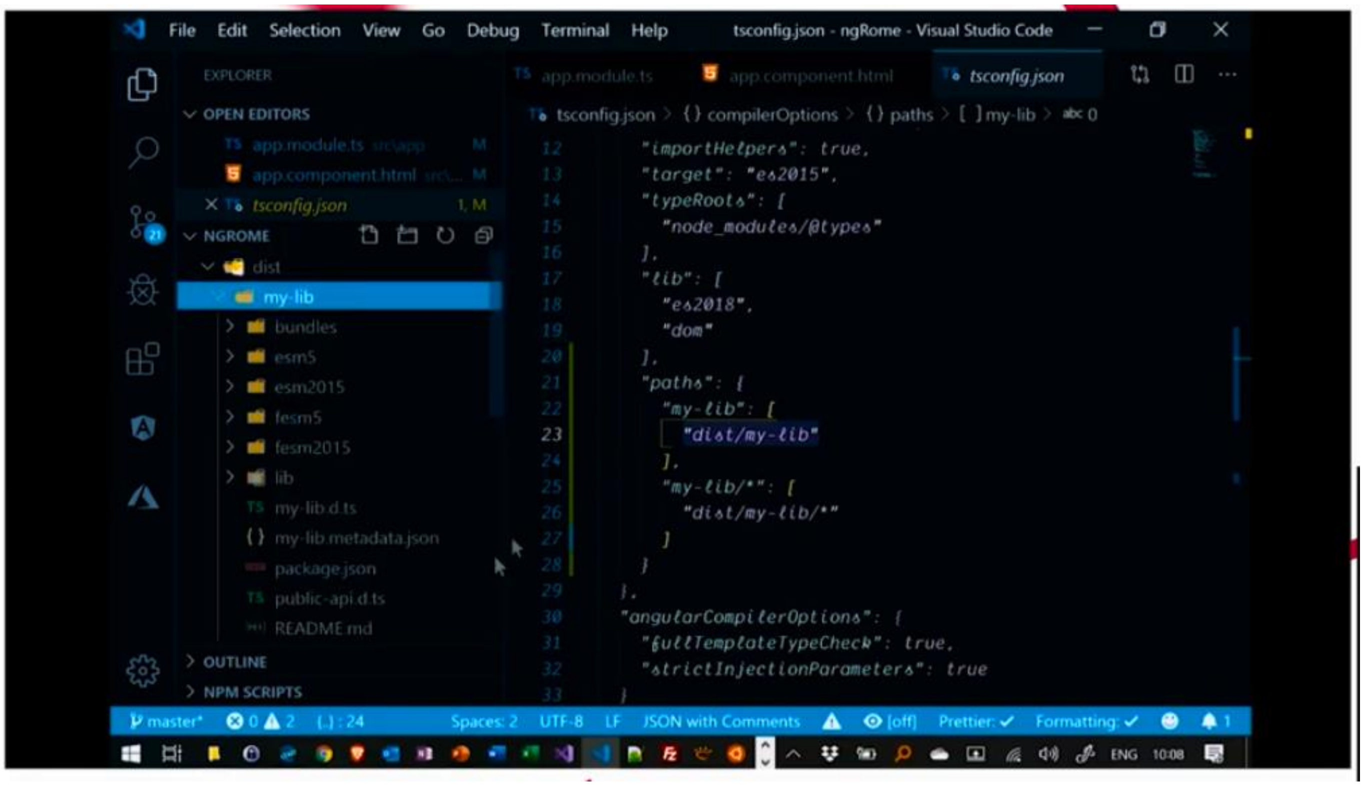


```
    "importHelpers": true,
    "target": "es2015",
    "typeRoots": [
      "node_modules/@types"
    ],
    "lib": [
      "es2018",
      "dom"
    ],
    "paths": {
      "my-lib": [
        "dist/my-lib"
      ],
      "my-lib/*": [
        "dist/my-lib/*"
      ]
    }
  },
  "angularCompilerOptions": {
    "fullTemplateTypeCheck": true,
    "strictInjectionParameters": true
  }
```

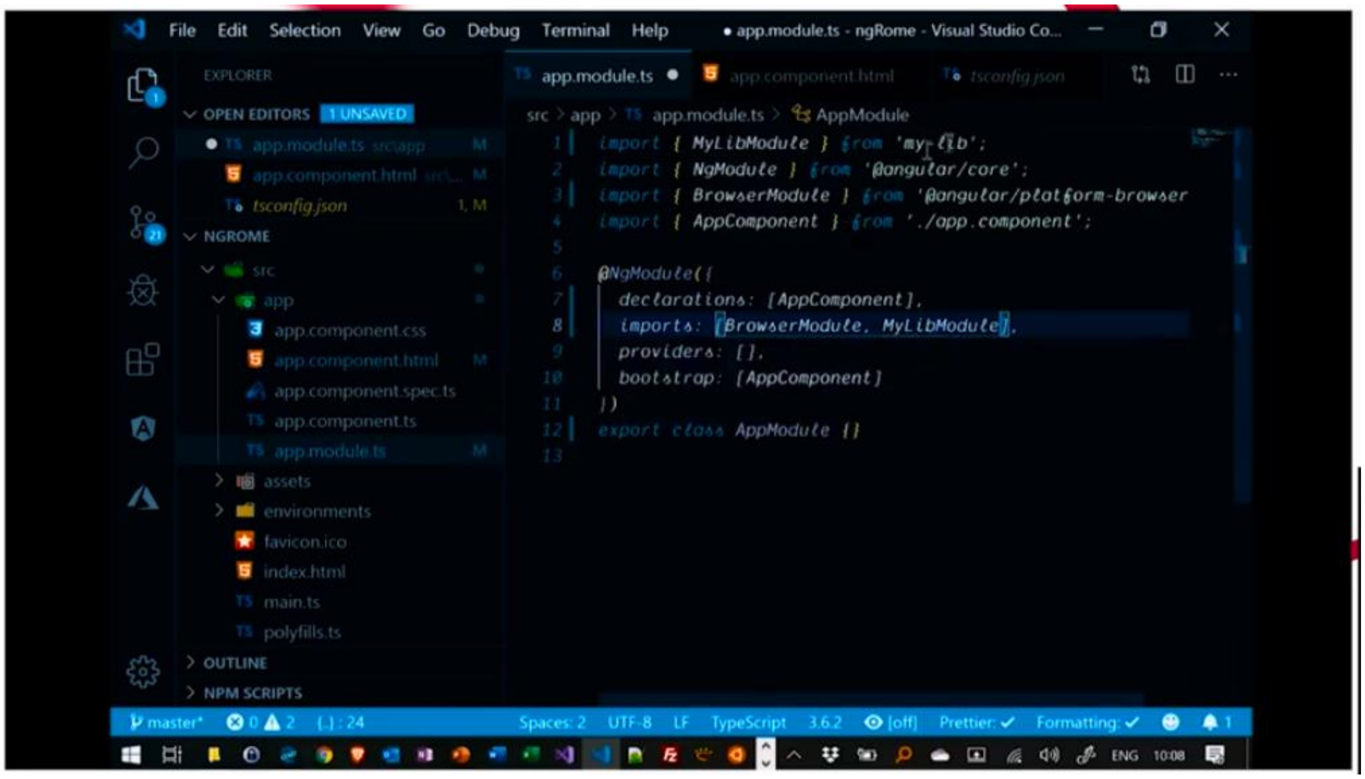


```
Microsoft Windows [Version 10.0.18362.388]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Fabian\Desktop\ngRome>ng build my-lib_
```

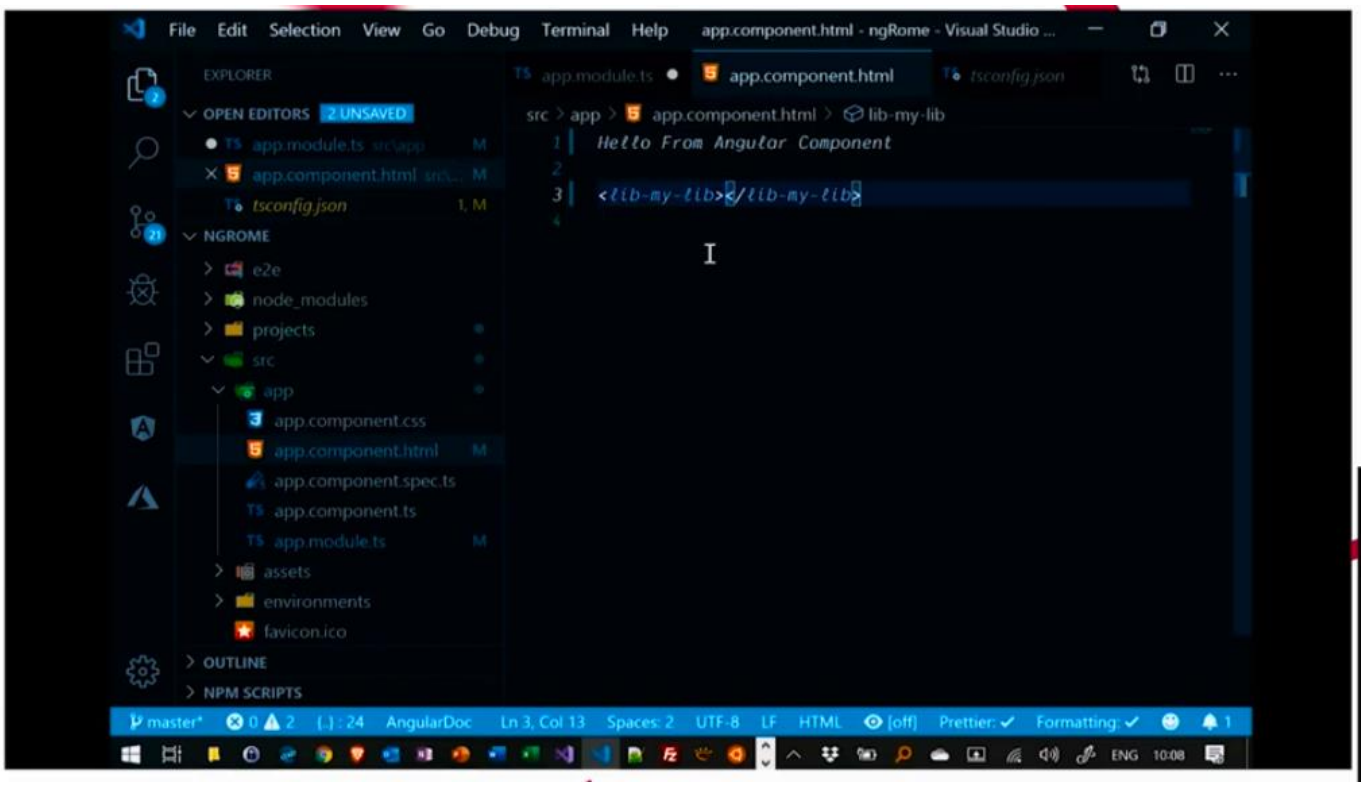
This will cause the ng-packager to run and build your **dist** folder with the files inside it.


File Edit Selection View Go Debug Terminal Help
tsconfig.json - ngRome - Visual Studio Code
EXPLORER
OPEN EDITORS
app.module.ts
app.component.html
tsconfig.json
NGROME
dist
my-lib
bundles
esm5
esm2015
fesm5
fesm2015
lib
my-lib.d.ts
my-lib.metadata.json
package.json
public-api.d.ts
README.md
OUTLINE
NPM SCRIPTS
tsconfig.json > {} compilerOptions > {} paths > [ ] my-lib > abc 0
master* 0 2 {.}: 24 Spaces: 2 UTF-8 LF JSON with Comments {off} Prettier: ✓ Formatting: ✓
ENG 10:08

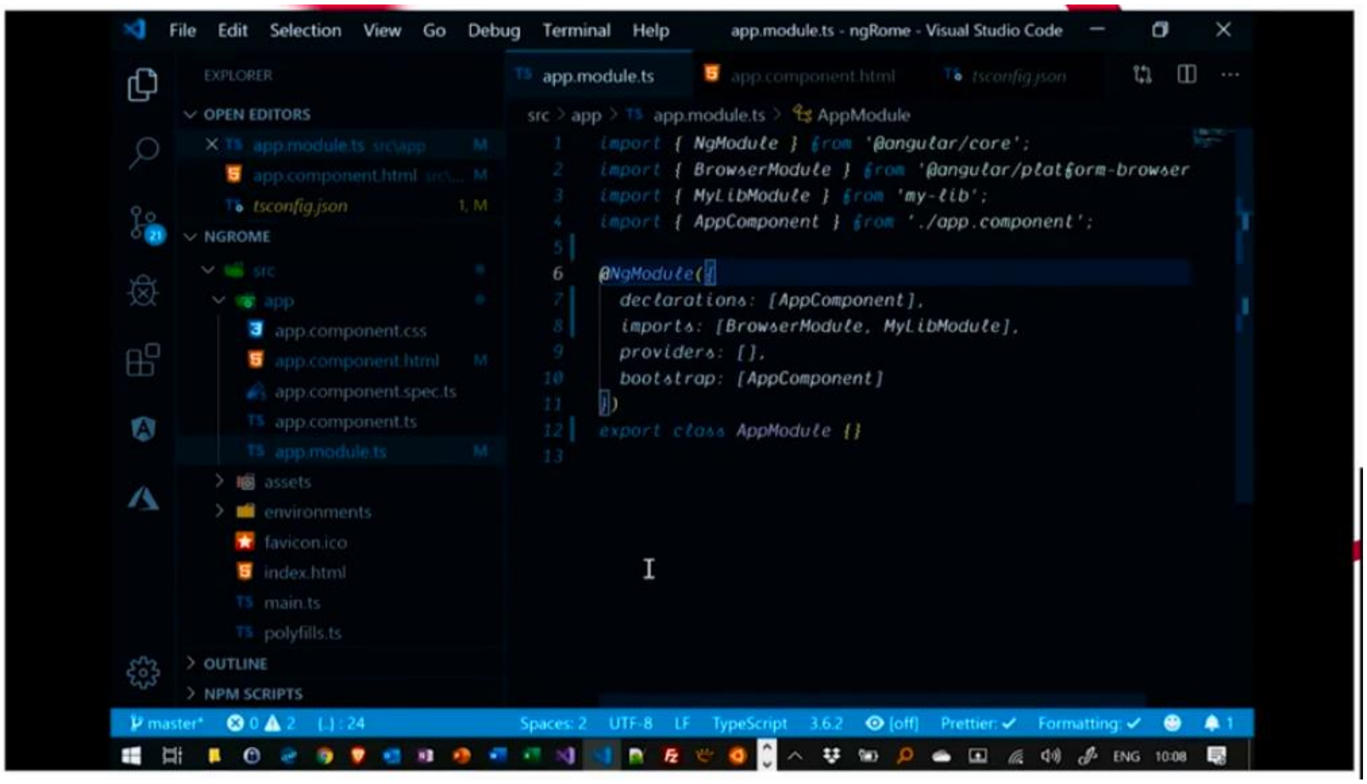
```
    "importHelpers": true,
    "target": "es2015",
    "typeRoots": [
      "node_modules/@types"
    ],
    "lib": [
      "es2018",
      "dom"
    ],
    "paths": {
      "my-lib": [
        "dist/my-lib"
      ],
      "my-lib/*": [
        "dist/my-lib/*"
      ]
    }
  },
  "angularCompilerOptions": {
    "fullTemplateTypeCheck": true,
    "strictInjectionParameters": true
  }
```



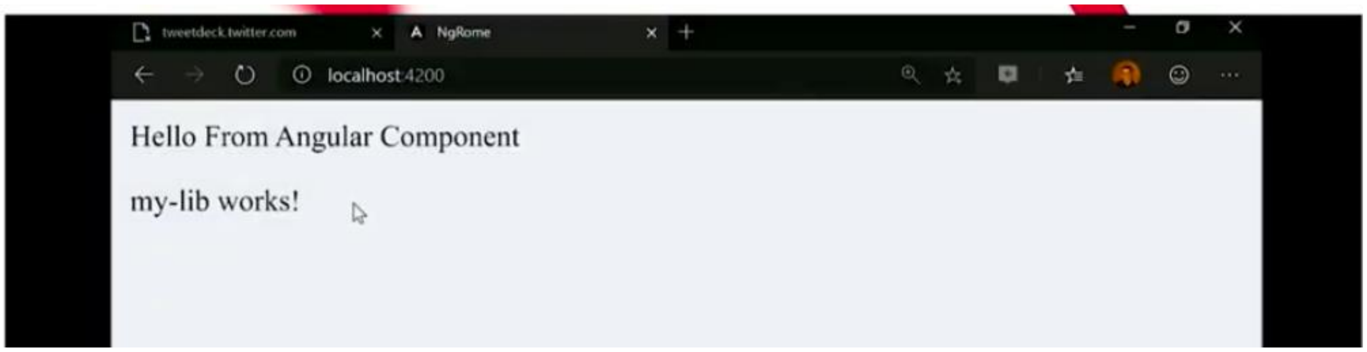
```
import { MyLibModule } from 'my-lib';
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser
import { AppComponent } from './app.component';

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



```
Hello From Angular Component

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

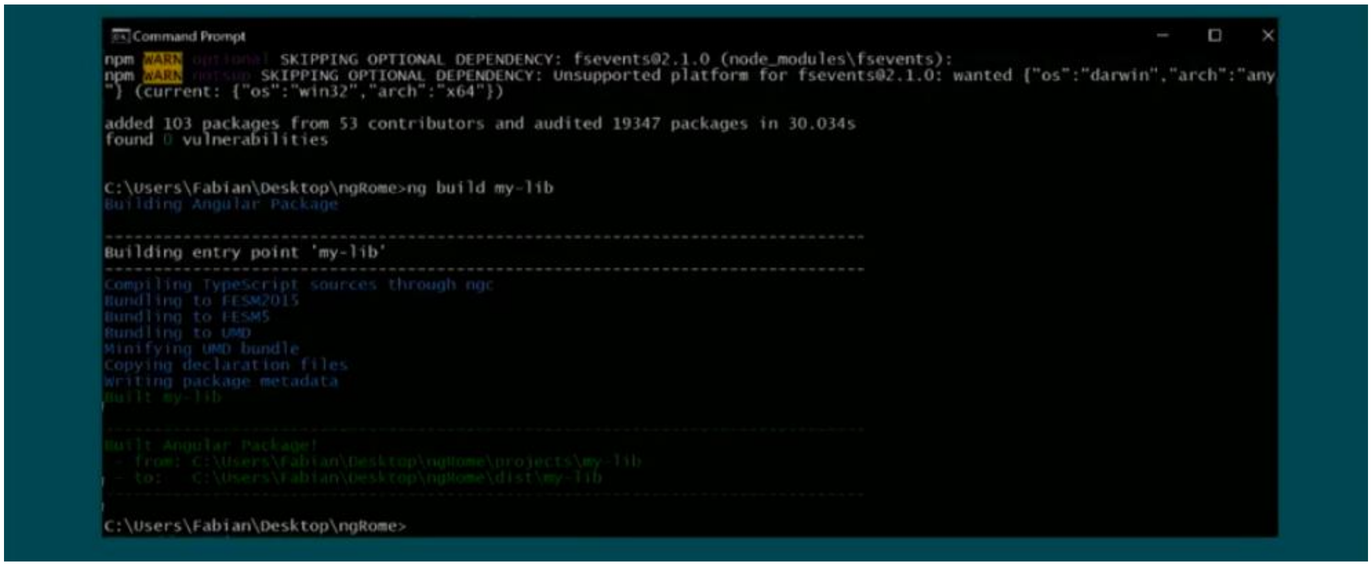


```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser
import { MyLibModule } from 'my-lib';
import { AppComponent } from './app.component';

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



Hello From Angular Component

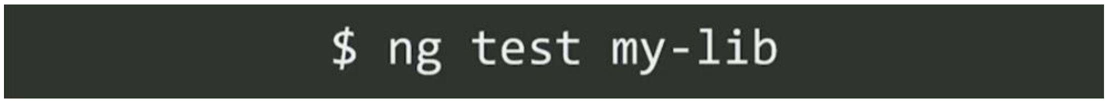
my-lib works!



```
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.0 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.0: wanted {"os":"darwin","arch":"any
"} (current: {"os":"win32","arch":"x64"})

added 103 packages from 53 contributors and audited 19347 packages in 30.034s
found 0 vulnerabilities


C:\Users\Fabian\Desktop\ngRome>ng build my-lib
Building Angular Package

------------------------------------------------------------------------------
Building entry point 'my-lib'
------------------------------------------------------------------------------
Compiling TypeScript sources through ngc
Bundling to FESM2015
Bundling to FESM5
Bundling to UMD
Minifying UMD bundle
Copying declaration files
Writing package metadata
Built my-lib

------------------------------------------------------------------------------
Built Angular Package!
 - from: C:\Users\Fabian\Desktop\ngRome\projects\my-lib
 - to:   C:\Users\Fabian\Desktop\ngRome\dist\my-lib
------------------------------------------------------------------------------

C:\Users\Fabian\Desktop\ngRome>
```

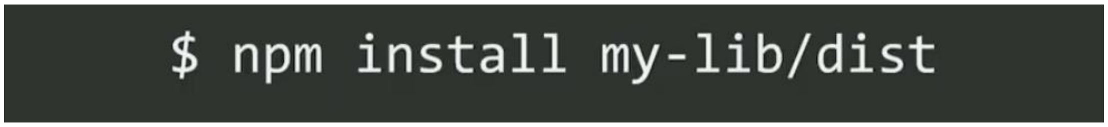
```
$ ng build my-lib --watch
```

This command will rebuild the Angular library on the fly when you change the code.



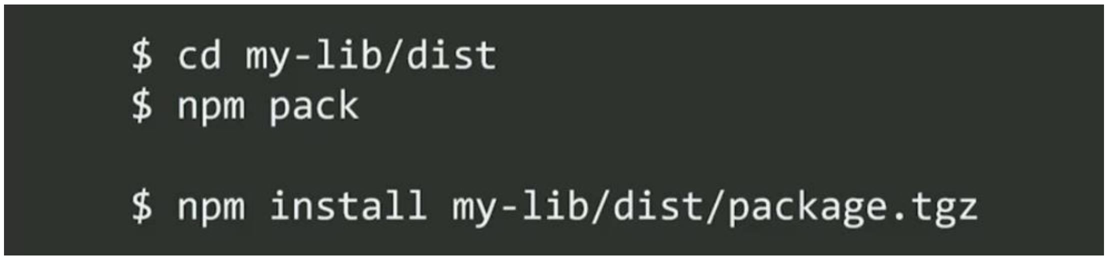
```
$ ng test my-lib
```

This will run only the test for the Angular library and you can see the results in the Angular CLI



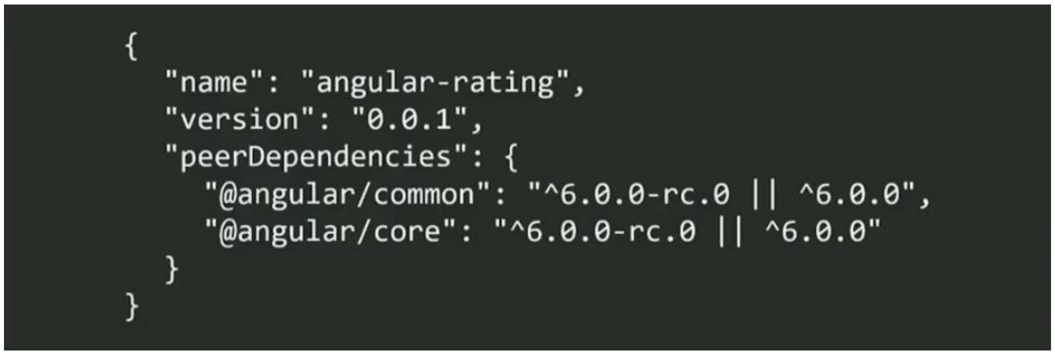
```
$ npm install my-lib/dist
```

To use the Angular library locally, you can just **$ npm install** it to a local folder for your **dists** files

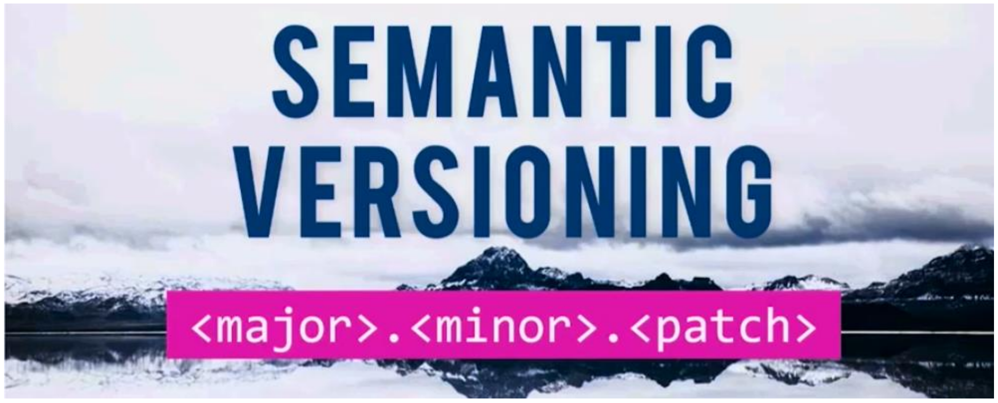


```
$ cd my-lib/dist
$ npm pack

$ npm install my-lib/dist/package.tgz
```

Alternatively, you can **cd** into a folder and run the **npm pack** command to get the **.tgz** file created to be then installed

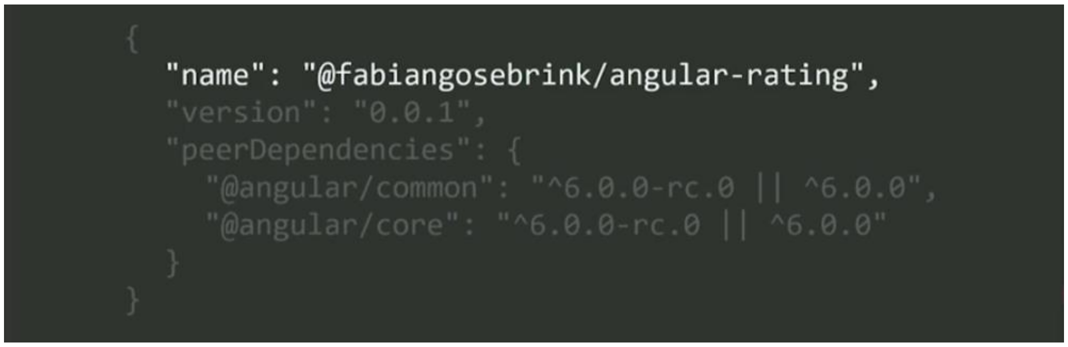


```
{
  "name": "angular-rating",
  "version": "0.0.1",
  "peerDependencies": {
    "@angular/common": "^6.0.0-rc.0 || ^6.0.0",
    "@angular/core": "^6.0.0-rc.0 || ^6.0.0"
  }
}
```

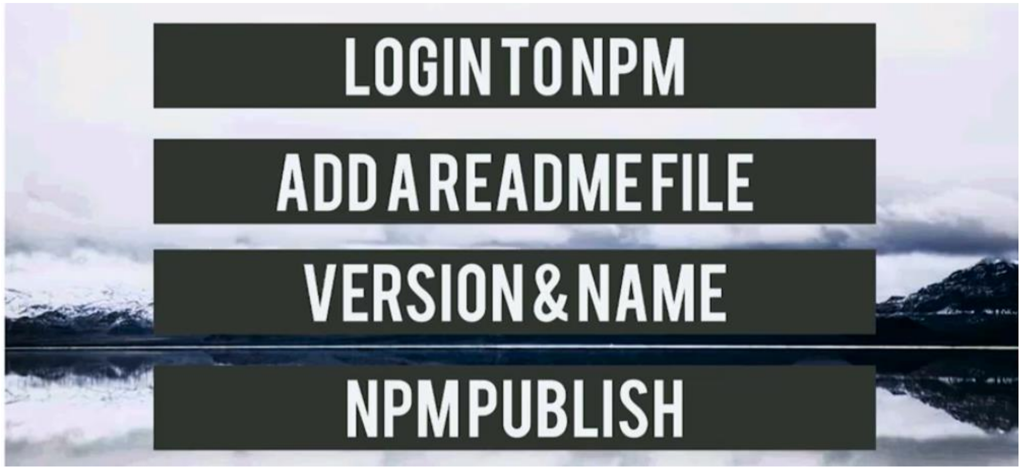


SEMANTIC VERSIONING

`<major>.<minor>.<patch>`



```
{
  "name": "@fabiangosebrink/angular-rating",
  "version": "0.0.1",
  "peerDependencies": {
    "@angular/common": "^6.0.0-rc.0 || ^6.0.0",
    "@angular/core": "^6.0.0-rc.0 || ^6.0.0"
  }
}
```

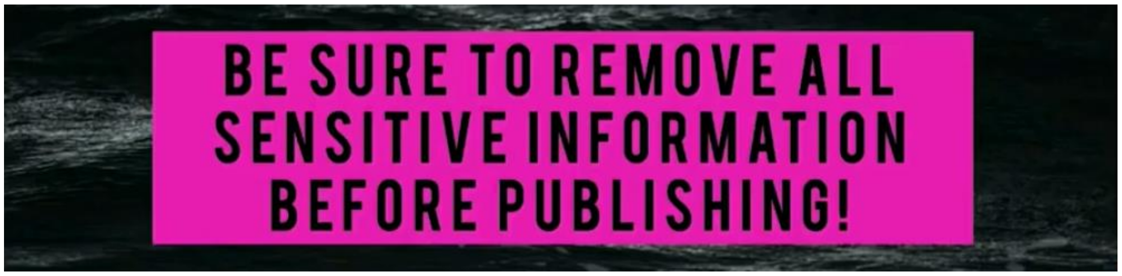


LOGIN TO NPM

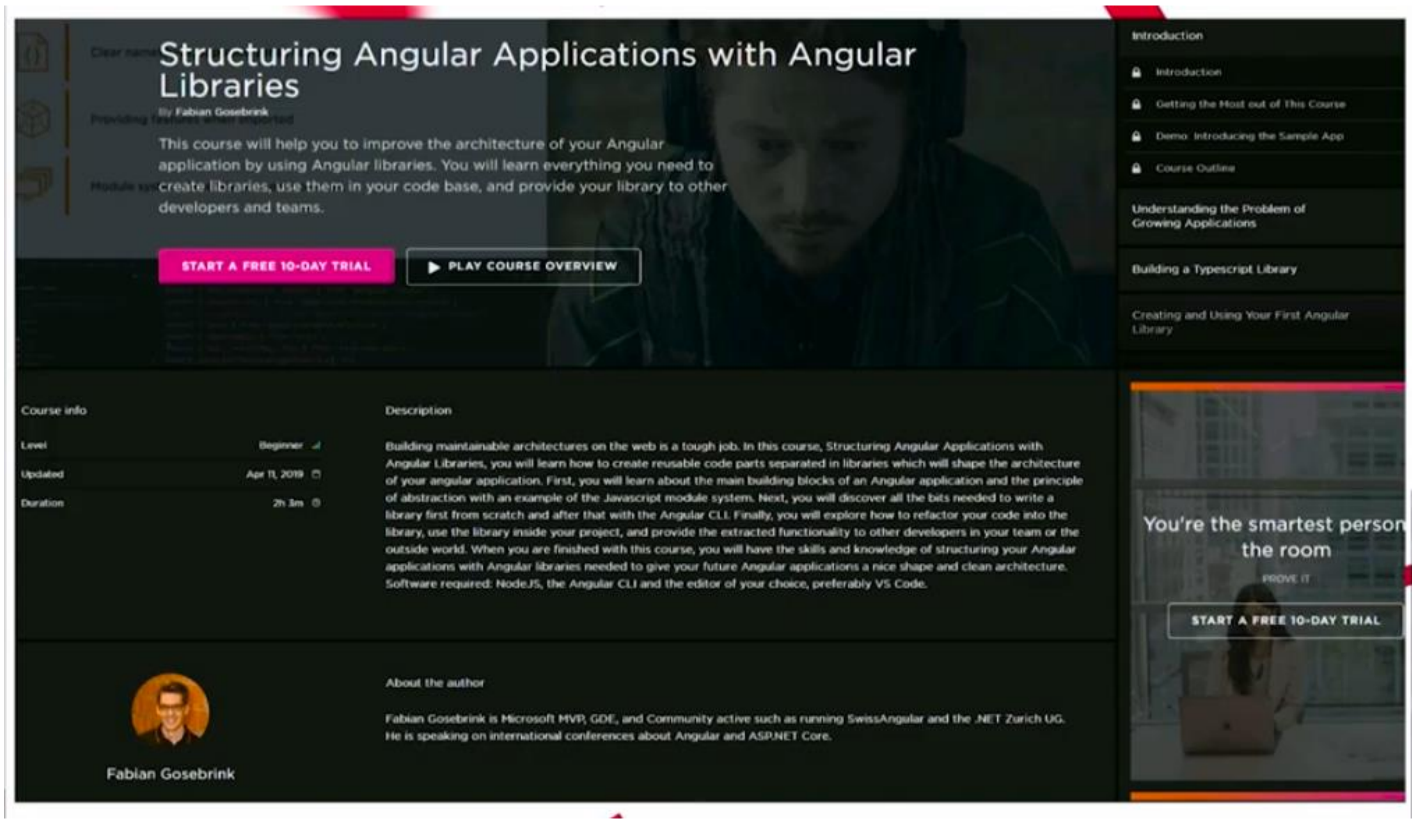
ADD A README FILE

VERSION & NAME

NPM PUBLISH



BE SURE TO REMOVE ALL SENSITIVE INFORMATION BEFORE PUBLISHING!



Structuring Angular Applications with Angular Libraries

By Fabian Gosebrink

This course will help you to improve the architecture of your Angular application by using Angular libraries. You will learn everything you need to create libraries, use them in your code base, and provide your library to other developers and teams.

Introduction

- Introduction
- Getting the Most out of This Course
- Demo: Introducing the Sample App
- Course Outline

Understanding the Problem of Growing Applications

Building a Typescript Library

Creating and Using Your First Angular Library

Course info

Level	Beginner
Updated	Apr 11, 2019
Duration	2h 3m

Description

Building maintainable architectures on the web is a tough job. In this course, Structuring Angular Applications with Angular Libraries, you will learn how to create reusable code parts separated in libraries which will shape the architecture of your angular application. First, you will learn about the main building blocks of an Angular application and the principle of abstraction with an example of the Javascript module system. Next, you will discover all the bits needed to write a library first from scratch and after that with the Angular CLI. Finally, you will explore how to refactor your code into the library, use the library inside your project, and provide the extracted functionality to other developers in your team or the outside world. When you are finished with this course, you will have the skills and knowledge of structuring your Angular applications with Angular libraries needed to give your future Angular applications a nice shape and clean architecture. Software required: NodeJS, the Angular CLI and the editor of your choice, preferably VS Code.

About the author

Fabian Gosebrink is Microsoft MVP, GDE, and Community active such as running SwissAngular and the .NET Zurich UG. He is speaking on international conferences about Angular and ASP.NET Core.

Fabian Gosebrink

You're the smartest person the room
PROVE IT
START A FREE 10-DAY TRIAL