* 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.
* Summary

**About Angular 13:**

Angular 13 is the latest version of the Angular family. Angular applications are basically designed for Single Page Applications, where all controls are required in single-page architecture. Angular is an application-based design framework, which provides the Cross-Platform development environment for creating the most sophisticated Single Page Applications (SPAs). These SPAs are efficient enough for giving us a component to view the model environment, making it an MVC or MVVM application.

Angular provides a Progressive Web App architecture, which means that an application made in Angular gives us an App-like experience with high-performance. Angular has zero-step installation, making its up-gradation using modern web capabilities possible, even if offline. Angular has the capability of making a cross-platform Desktop App that can be created using MAC, Windows, or Linux OS using the same Angular methods. Angular can be rendered in [Node.js](https://www.knowledgehut.com/tutorials/node-js/node-js-introduction), [Microsoft .Net](https://www.knowledgehut.com/programming/microsoft-net-framework-training), [PHP](https://www.knowledgehut.com/web-development/php-and-mysql-training), and many other servers by giving the output in HTML-CSS format. This even optimizes the App for SEO. Angular framework turns our templates into [JavaScript](https://www.knowledgehut.com/tutorials/javascript-tutorial/javascript-overview) Virtual machine codes which are highly optimized. With Angular CLI, the Command-Line tools, we can build and deploy Apps faster than ever before. Angular uses Karma for unit tests and Protractor for scenario tests making the applications made in [Angular](https://www.knowledgehut.com/tutorials/angular) more stable. Check out the [Full Stack Developer online course](https://www.knowledgehut.com/web-development/fullstack-development-bootcamp-training) for more information.

Know more about [angular cli](https://www.knowledgehut.com/blog/web-development/install-angular-cli).

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](https://www.knowledgehut.com/blog/web-development/turbocharge-web-apps-with-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](https://www.knowledgehut.com/web-development/angular-js-training) 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](https://www.knowledgehut.com/blog/web-development/angular-material-component-libraries). It is a library or a component that helps in constructing attractive, sophisticated, functional web app and reusable, and adaptable UI components.

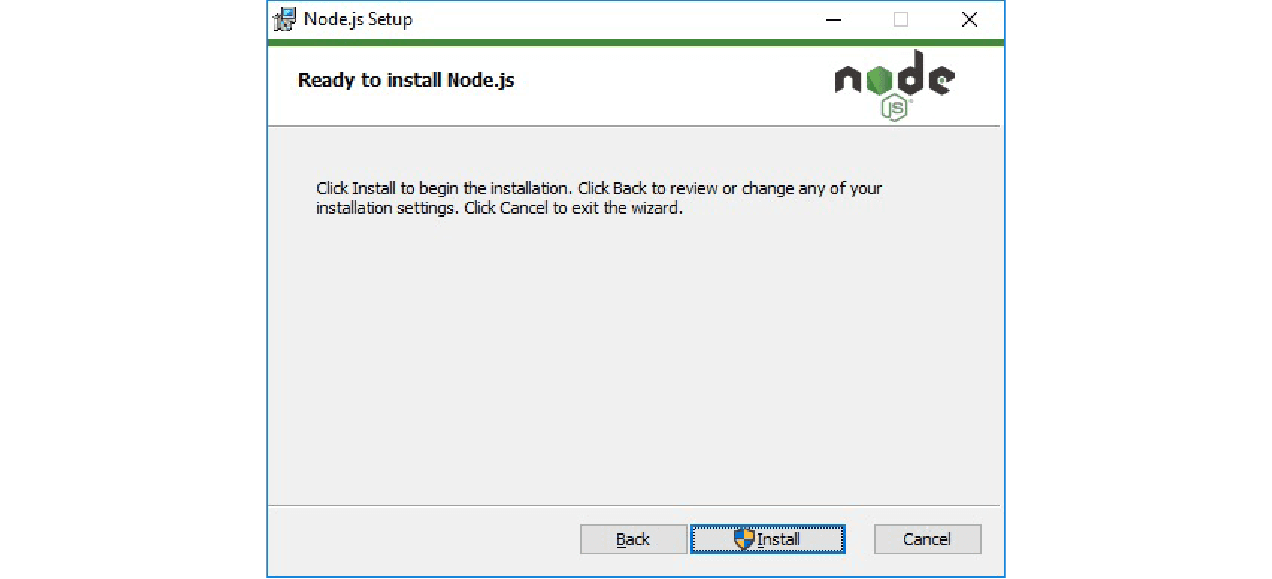
To advance your career in web development, enroll in [learning Web Development](https://www.knowledgehut.com/web-development-courses) course.

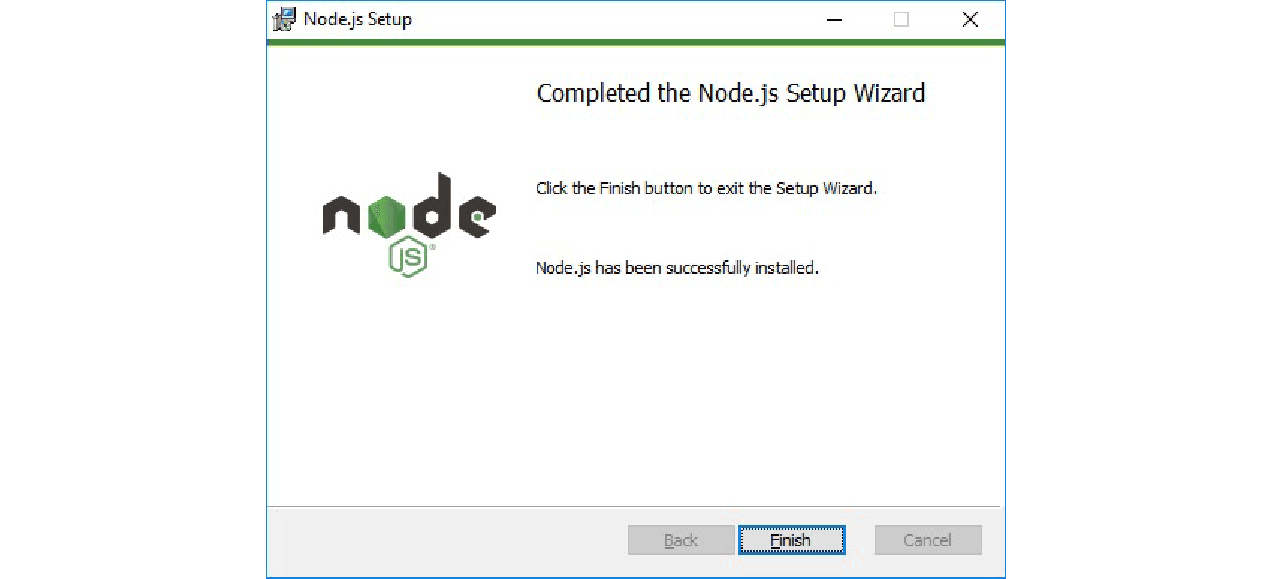
**Creating a New Library**

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

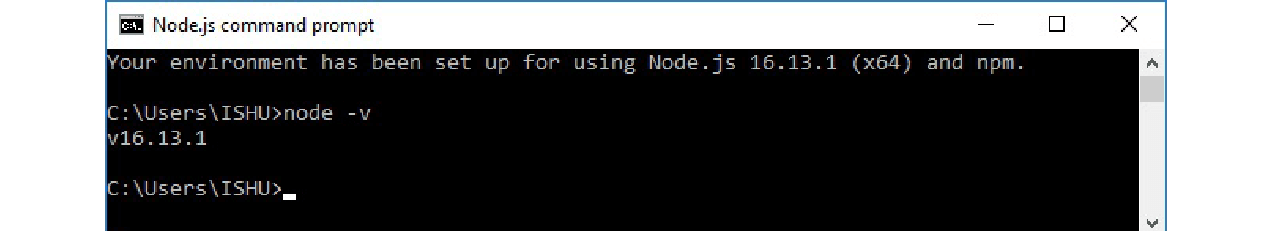
**Step 1:** Install Node.js:

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





* Once you have installed [Node.js](https://www.knowledgehut.com/web-development/node-js-training) 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

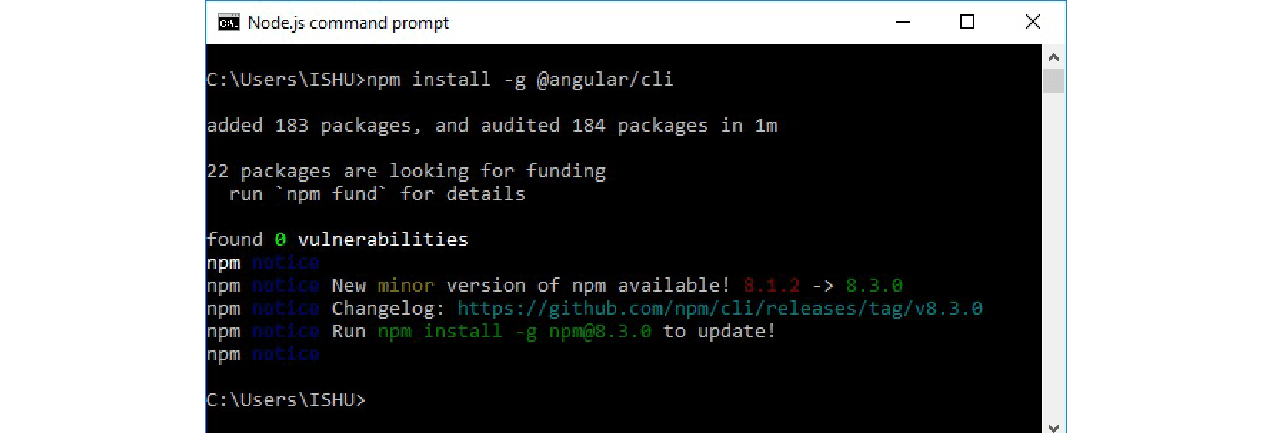
npm install -g @angular/cli

Copy Code

Or

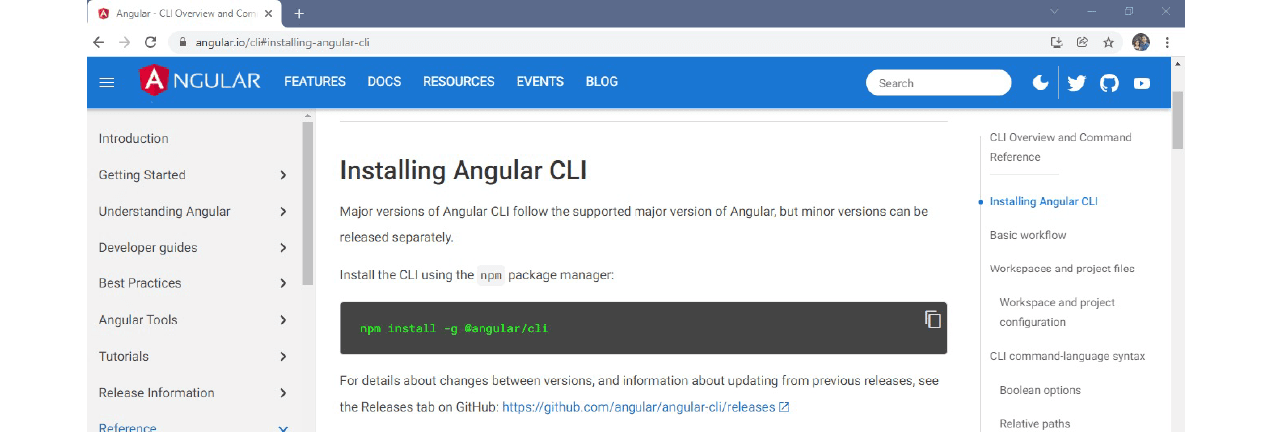
npm install -g @angular/cli@latest

Copy Code

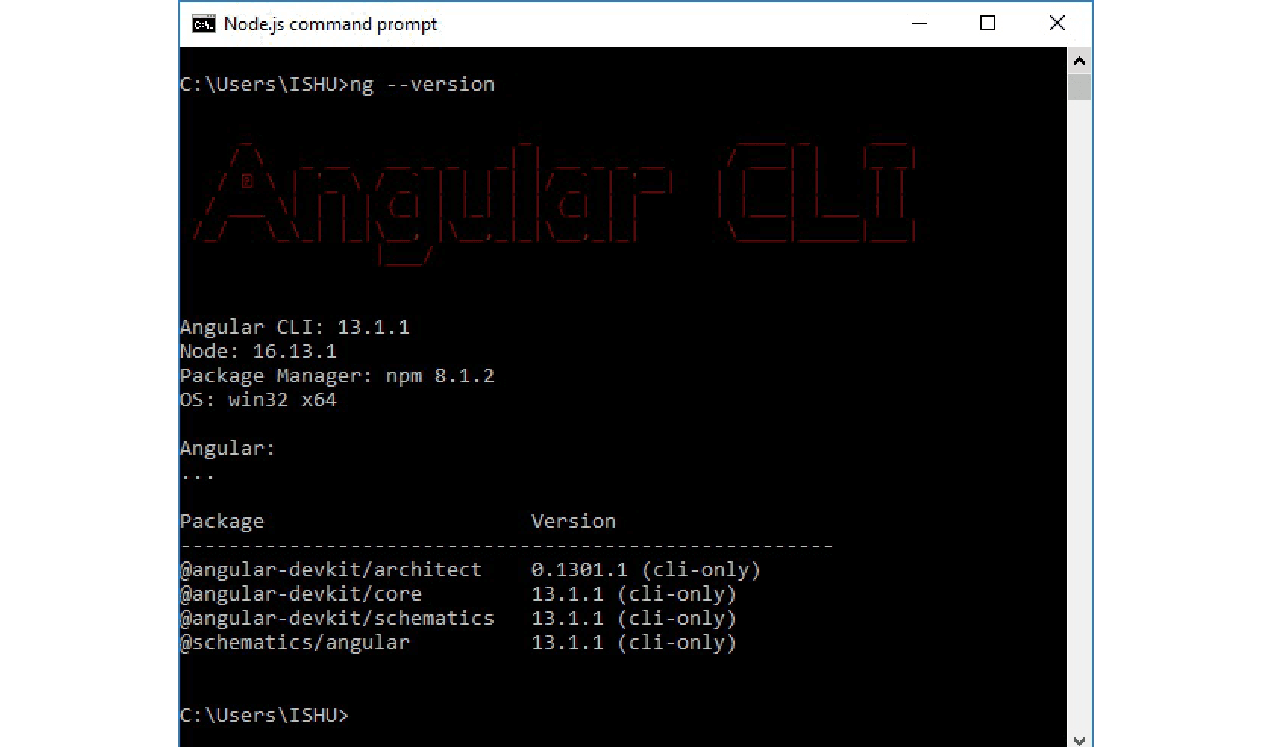


Or

* Just go to Angular CLI official website [Angular.io](https://cli.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.



**Step 3:** Create an app called ngApp4Library

**Syntax:**

ng new app\_name

C:\>ng new ngApp4Library

Copy Code

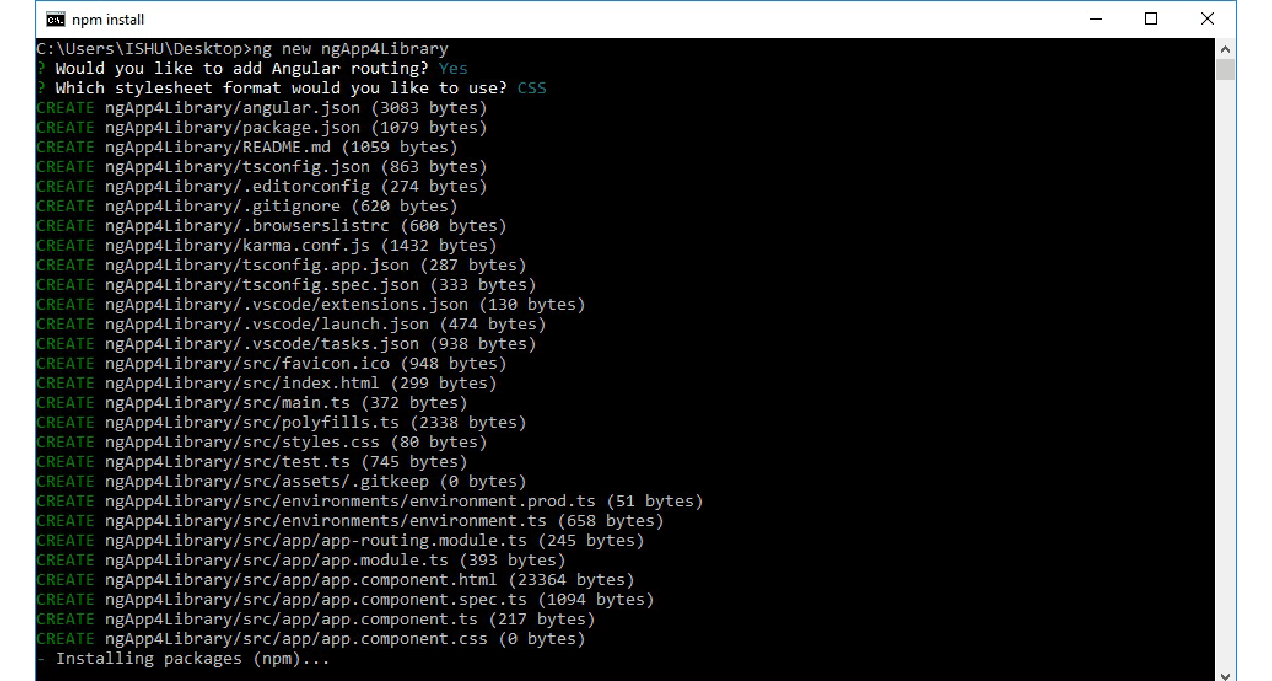
It asks for

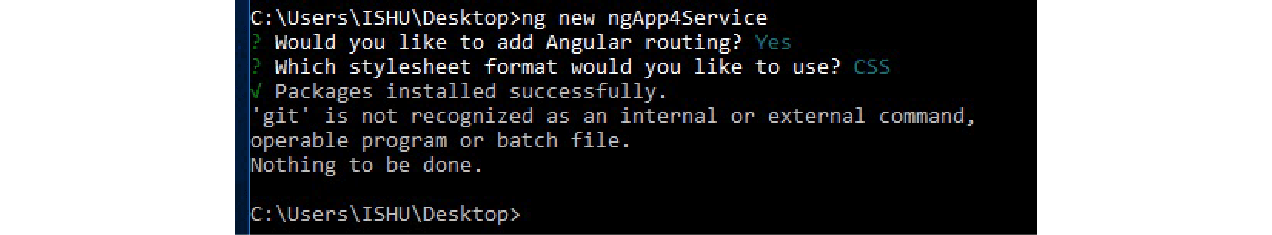
Would you like to add Angular routing? Yes

Which stylesheet format would you like to use?

> CSS

….





**Step 4:** Generate Library via CLI:

**Syntax:** for creating a Library

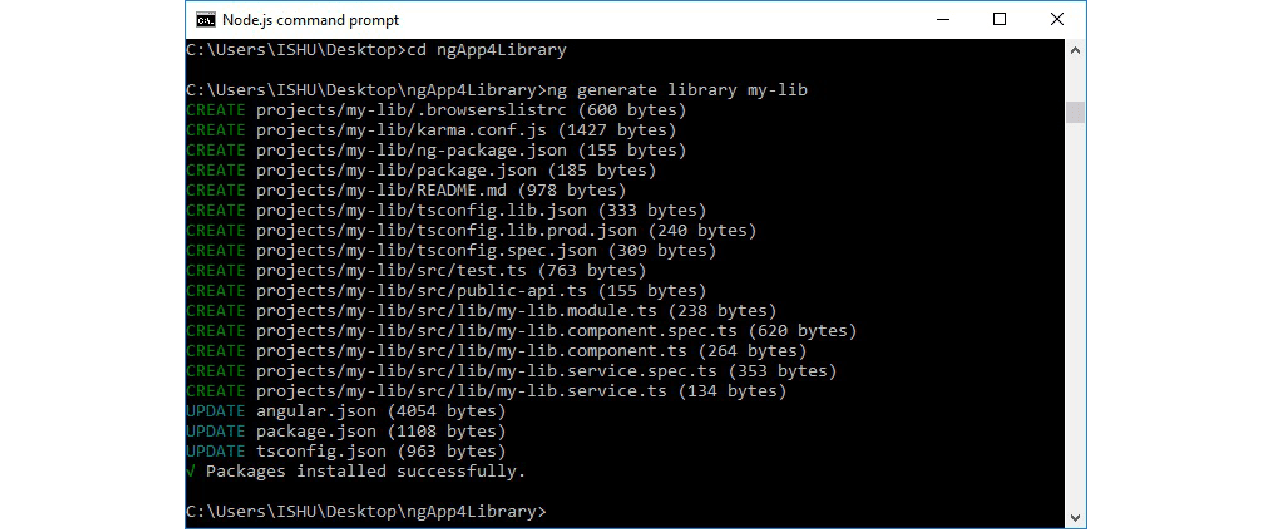
ng generate library <name> [options]

ng g library <name> [options]

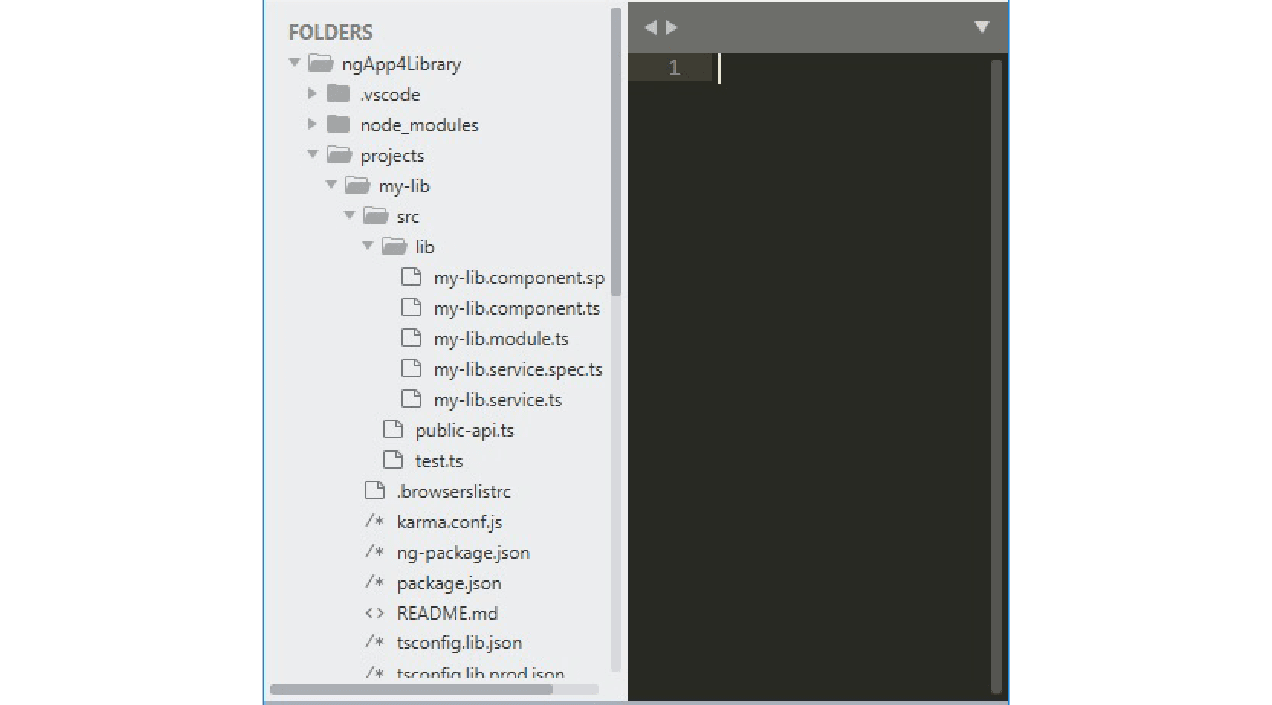
Copy Code

* 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-lb**



* 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 {

  }

}

Copy Code

**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**

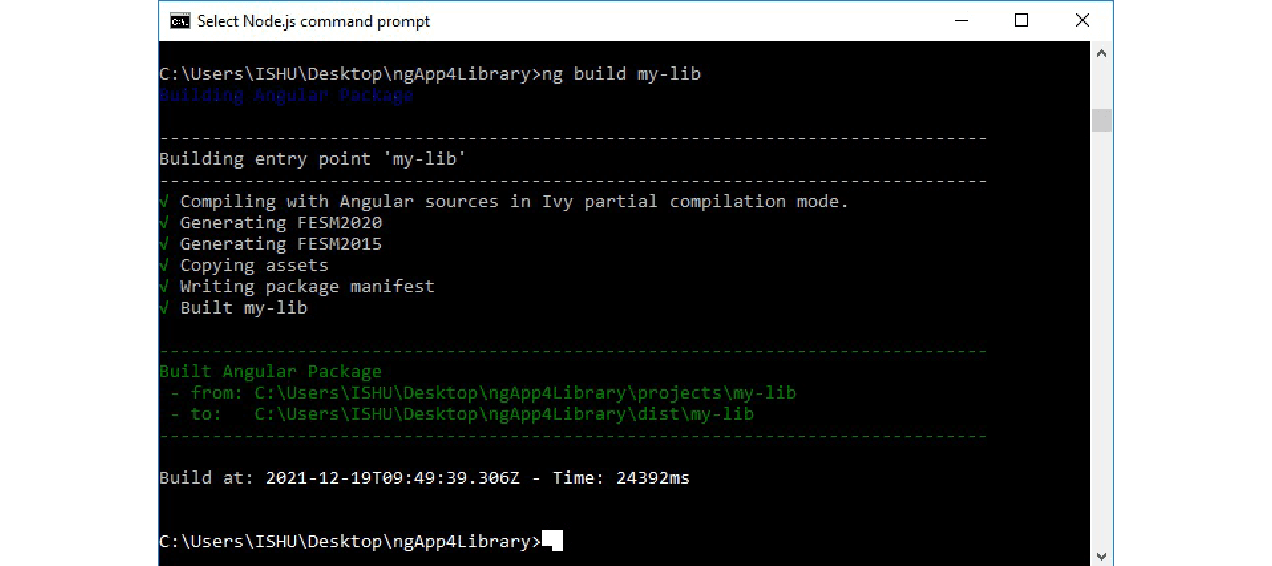
1. Build the library and consume it in the same application:

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

ng build <library name>

Copy Code

* 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**
* 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]

})

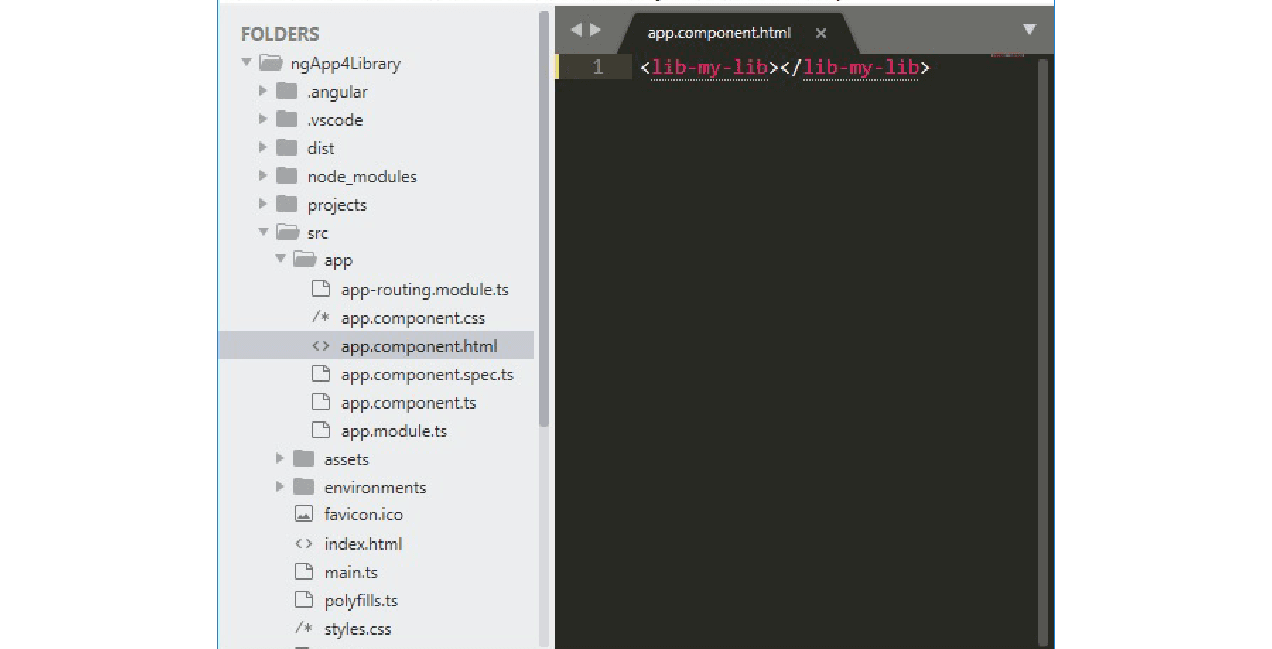
export class AppModule { }

Copy Code

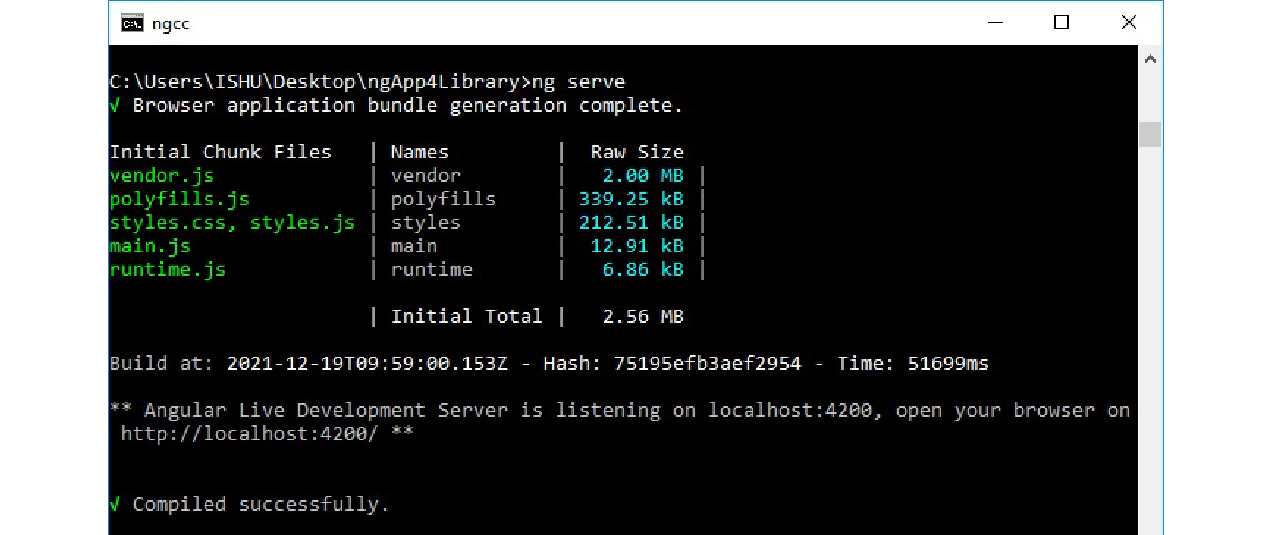
* Now, we simply add the my-lib library in the HTML file.
* Please note that the selector for the library used here is: **lib-my-lib**
* To know about the selector (**lib-my-lib**), we can check the file: ngApp4Library -> projects -> my-lib -> src -> lib->  **my-lib.component.ts**
* Now open and edit: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

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

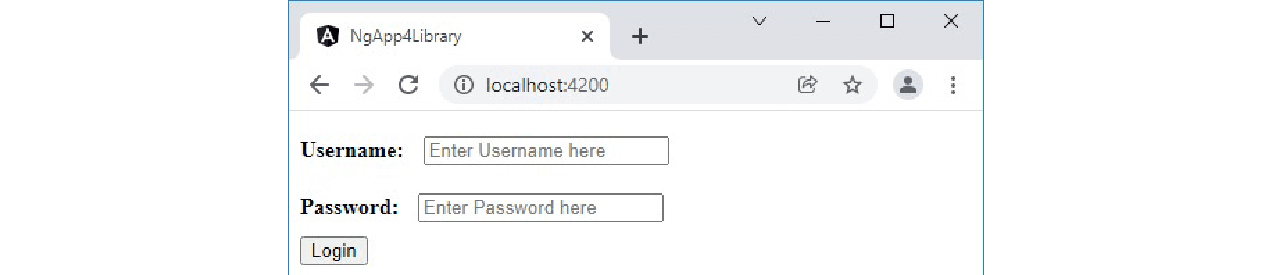
Copy Code



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



* Open the favourite browser and type the default Angular app URL to check the output: [localhost:4200/](http://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:

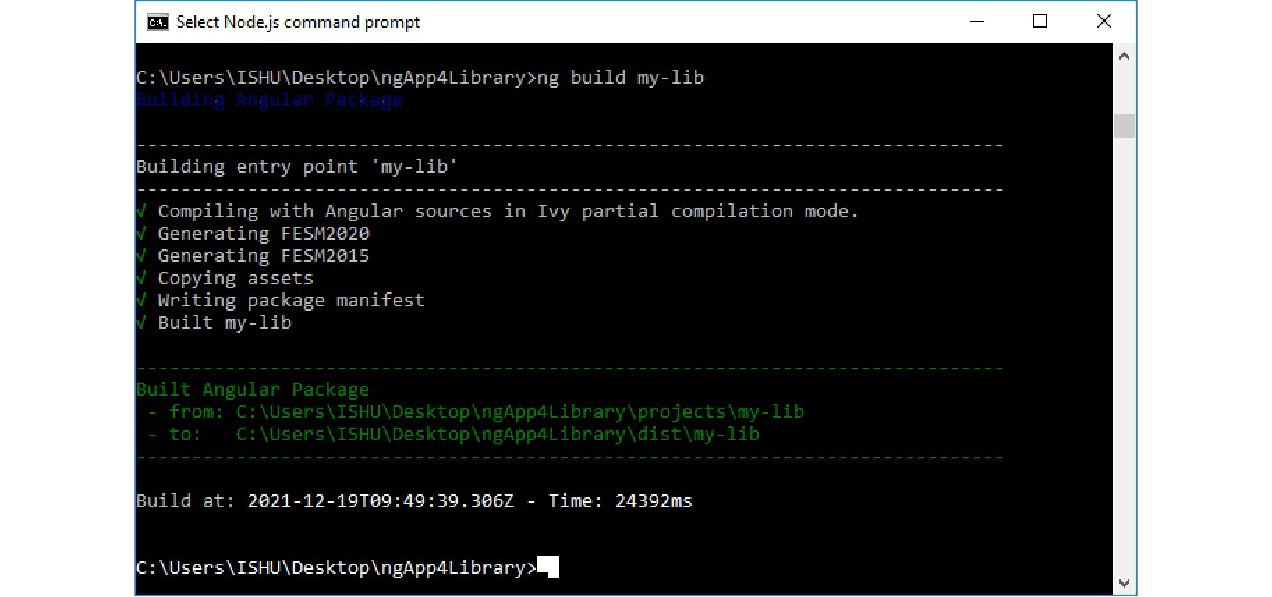
**Step 1:**

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

ng build <library name>

Copy Code

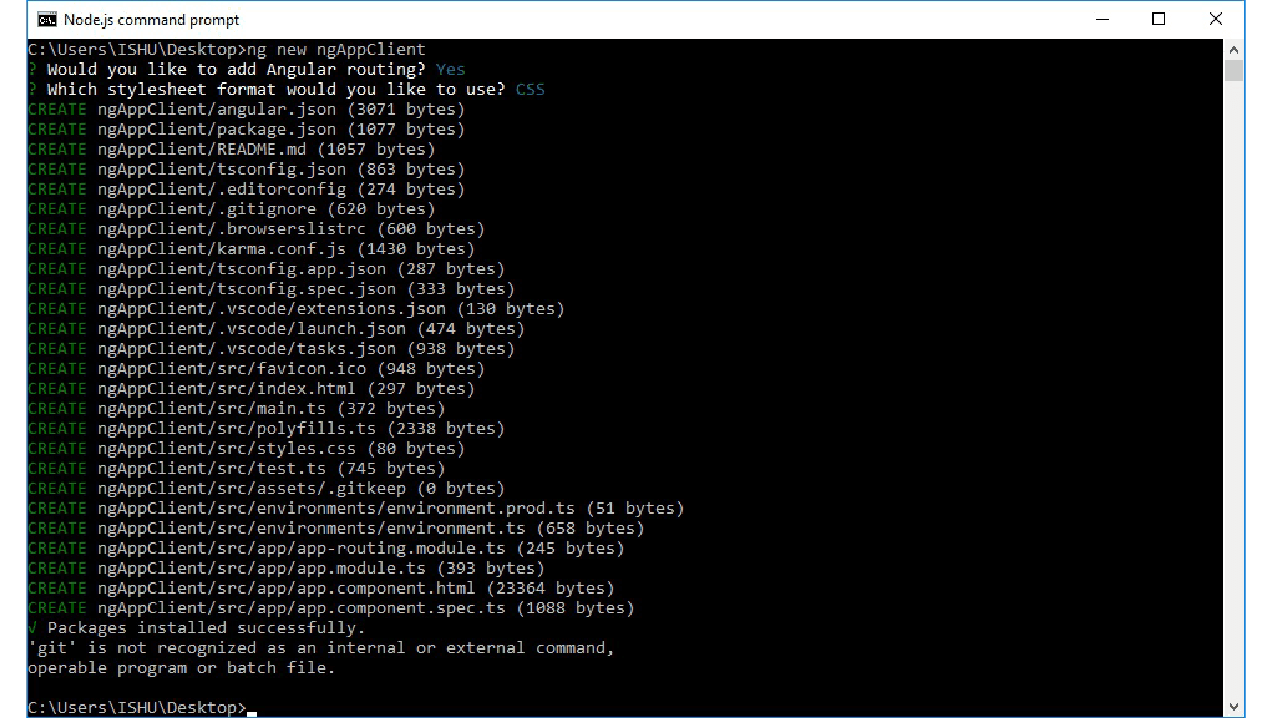
* Here our library name is my-lib, thus the command we need is: **ng build my-lib**



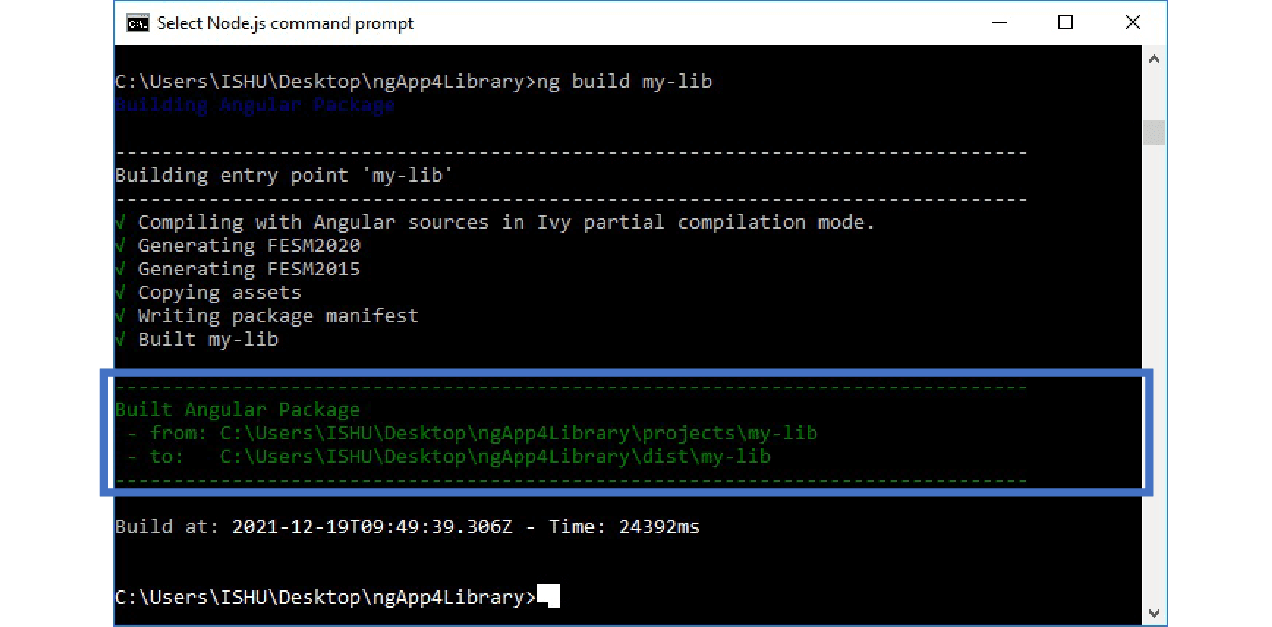
* The command will generate the library **dist folder**
* Now, create a new Angular application: ngAppClient in a new command window. We need to let the library application run.

ng new ngAppClient

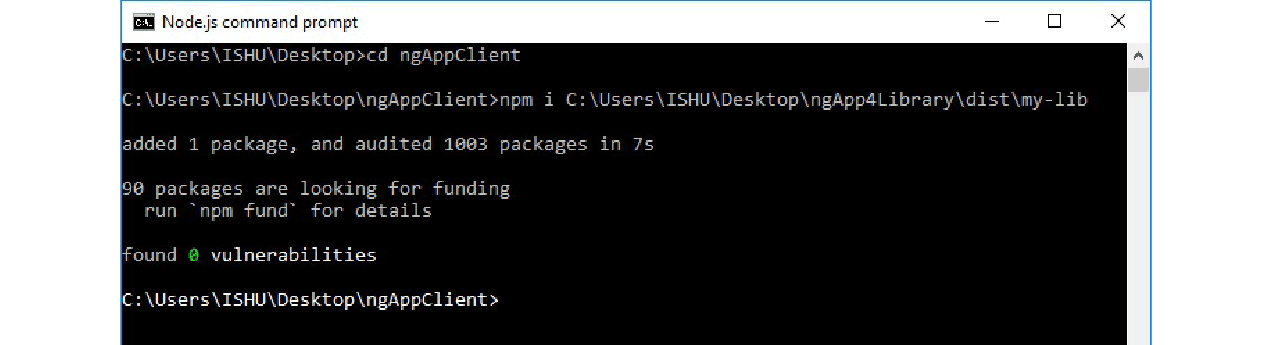
Copy Code



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



* 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**



* 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]

})

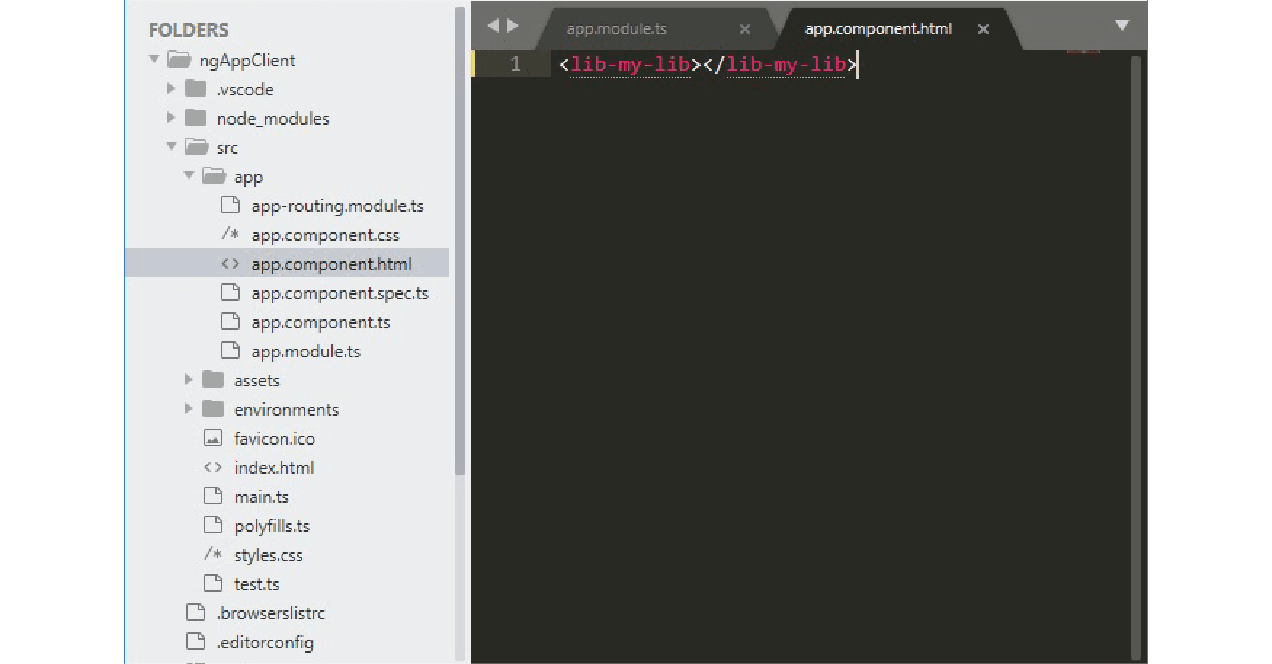
export class AppModule { }

Copy Code

* 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>

Copy Code



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

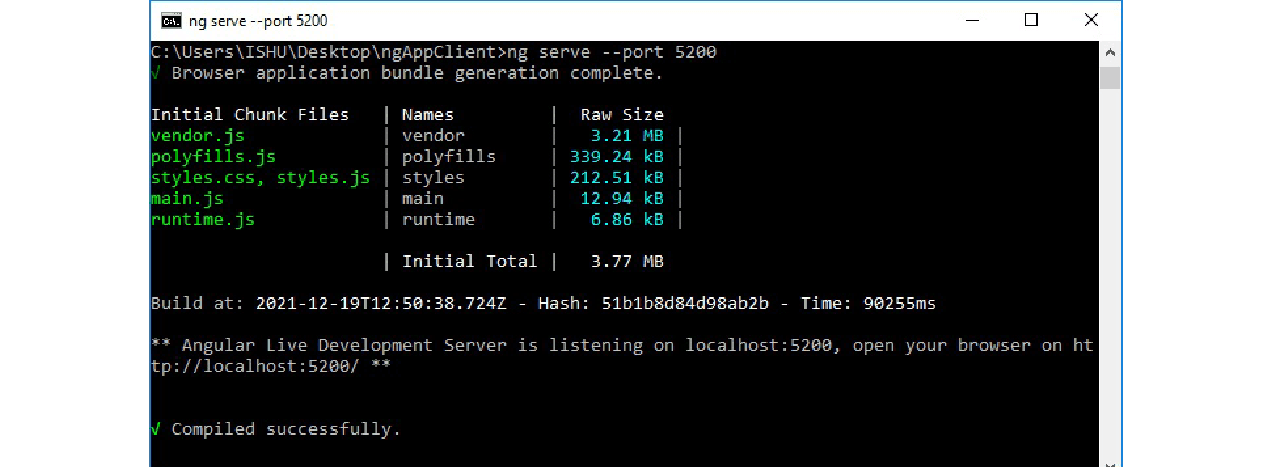
ng serve.

Copy Code

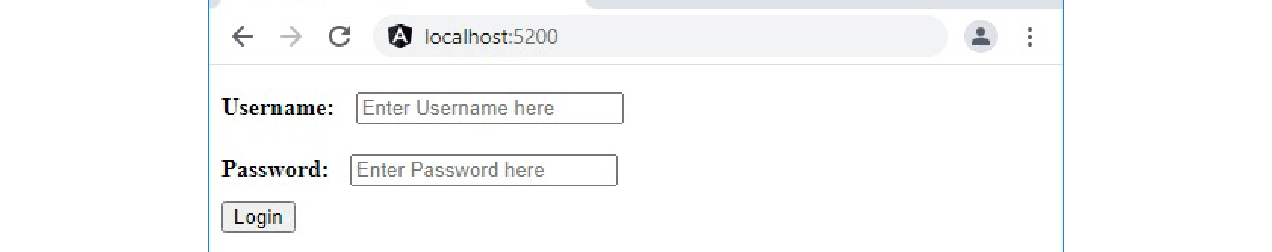
* In case the Library project is running in the default port, 4200, we can change the port of the client app to 5200 by the following command:

ng serve --port 5200

Copy Code



* Open the favorite browser and type the default Angular app URL to check the output: [localhost:5200/](http://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

Copy Code

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

Copy Code

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](https://www.knowledgehut.com/blog/web-development/npm-in-nodejs) publish” command. To use a library we install it by “ng i “ command.