Angular CLI

The Angular CLI is a command-line interface for Angular developed by Angular Team. This tool helps us quickly get started with creating the Angular Application.

## Why Angular CLI

Angular is easy to learn Front End Framework. But setting development environment is pretty complicated. There are many decisions, you have when you choose Angular as your JavaScript Framework.

You have an option of choosing Javascript, Typescript or dart. You have to choose a module loader among SystemJs & Webpack etc. You need to pickup your testing Framework.

Once you make a choice, then you have set up a lot of libraries and packages. Each of these libraries comes with their own configuration files. For instance, you need to create Package.Json file and add all the required libraries. Create Webpack or SystemJS configuration files. Configuration files are also required for Typescript & Typings etc. You need to configure the Testing Framework also.

Instead, you can install Angular CLI and get started right away.

## What is Angular CLI

The Angular CLI helps you to quickly create an Angular application with all the configuration files and packages in one single command. It also helps us to add features ([components](https://www.tektutorialshub.com/angular/angular-component/), [directives](https://www.tektutorialshub.com/angular/angular-directives/), [services](https://www.tektutorialshub.com/angular/angular-services/), etc) to existing Angular applications.It helps us to test , build & distribute our application

The Angular CLI creates the Angular Application and uses Typescript, Webpack ( for Module bundling), Karma ( for unit testing), Protractor ( for an end to end testing).

## Installing Angular CLI

The first step is to install the Angular CLI. This can be done by using the following command.



The above command installs the latest version of Angular CLI in your machine. Note that we have used the -g flag, (which stands for global) installs the Angular CLI system-wide so that you can use it in your all projects.



Since, the Angular Version 6, the Angular CLI follows the same Version No as the Angular. Hence for Angular 7, the corresponding version of the Angular CLI is 7.

The Angular CLI version 1.7 was for Angular 5 and Angular CLI 1.4 was for Angular 4

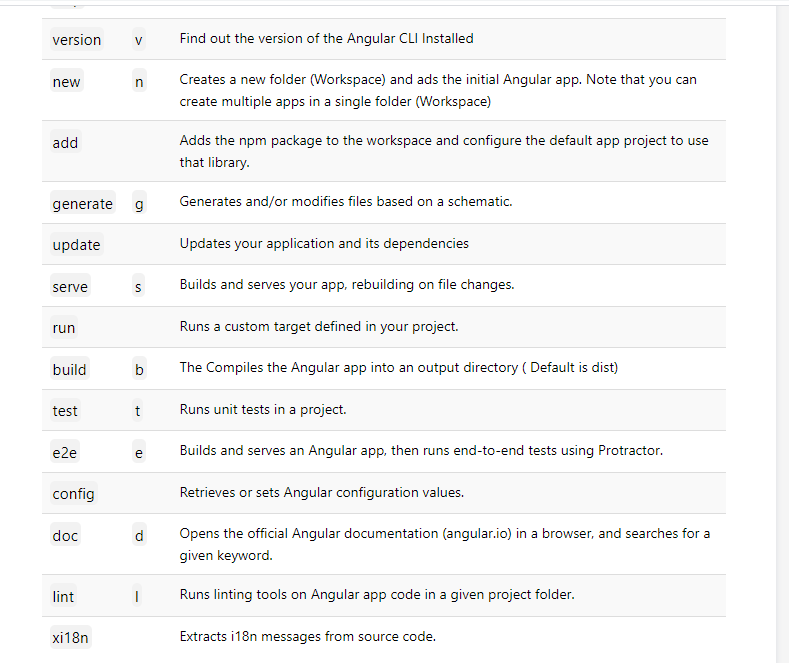
**You can keep track of the latest Angular CLI release from this link https://github.com/angular/angular-cli/releases**

### [Angular CLI Version Check](https://www.tektutorialshub.com/angular/how-to-check-angular-cli-version/)

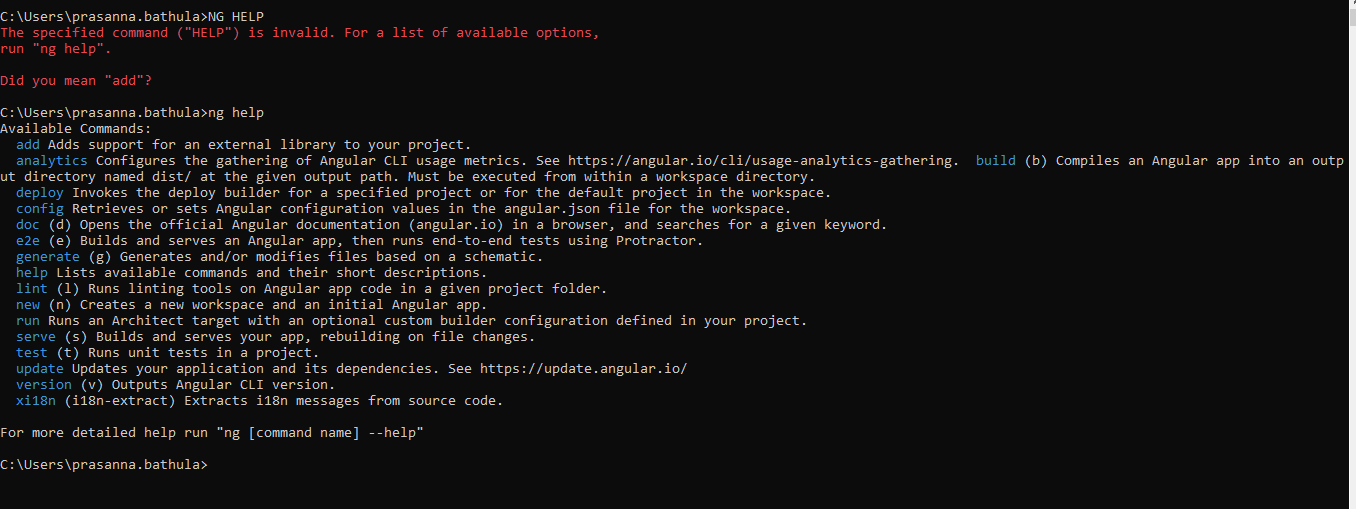
You can find out the [Current Installed Angular CLI Version](https://www.tektutorialshub.com/angular/how-to-check-angular-cli-version/) by Using the Command



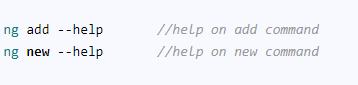
## Angular CLI Commands

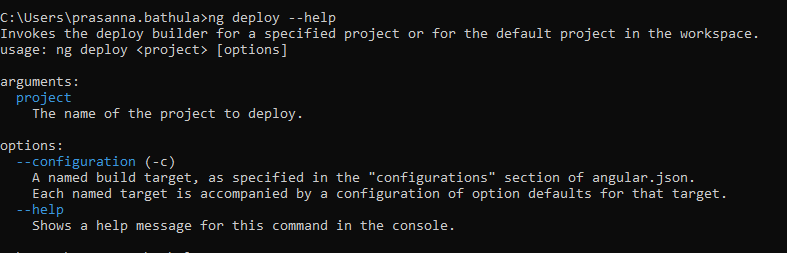


## Getting Help



Getting help on individual commands us the syntax ng [command name] --help. For Example





## Creating the Application with ng new

The ng new command is used to create new folder and creates an App with the provided name.

The command will ask you

* **What name would you like to use for the project?**  
  Enter the name of the project here. “GettingStarted”
* **Would you like to add Angular Routing?**  
  Answer this as Yes unless you do not want to add [Angular Routing](https://www.tektutorialshub.com/angular/angular-routing-navigation/).
* **Which stylesheet format would you like to use?**  
  You arrow keys to select the available options from CSS, SCSS, SASS, LESS, and Stylus

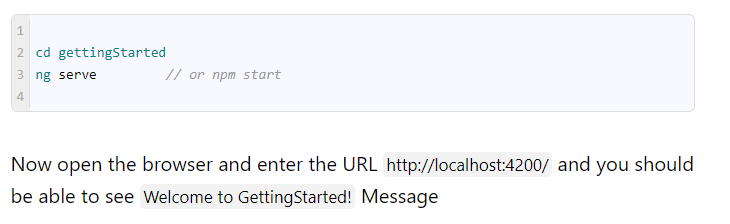


The above command does the following

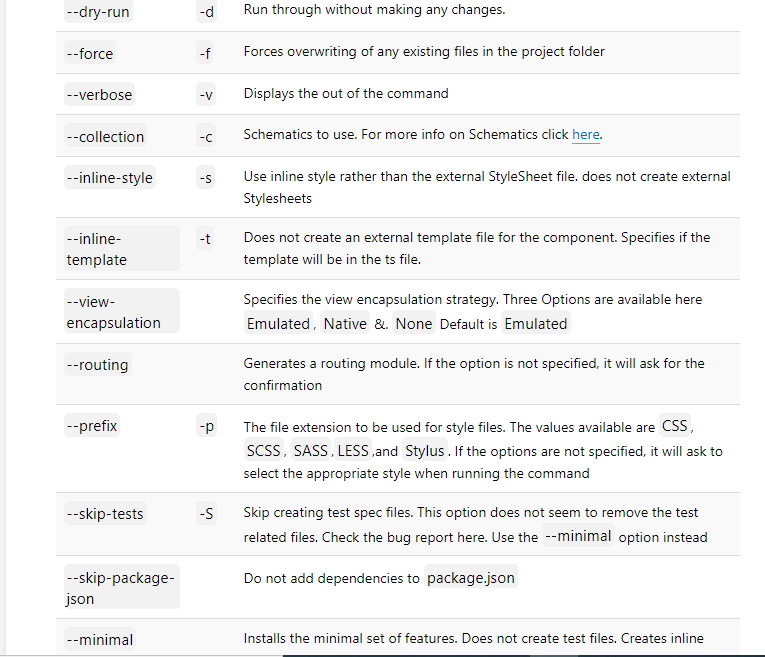
1. Creates a new directory GettingStarted is created
2. Downloads and installs Angular libraries and any other dependencies
3. Installs and configures TypeScript.
4. Installs and configures Karma & Protractor for testing
5. Initialises the Git.

### Running the Application

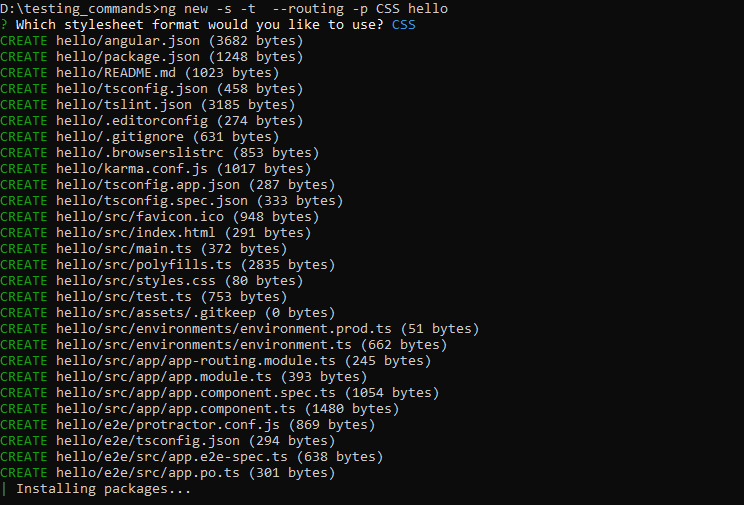
To run the application cd into the folder and run either ng server or use npm start (which runs the ng serve behind the scene)



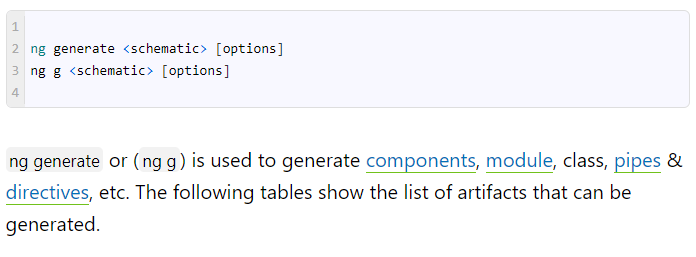
### ng new options

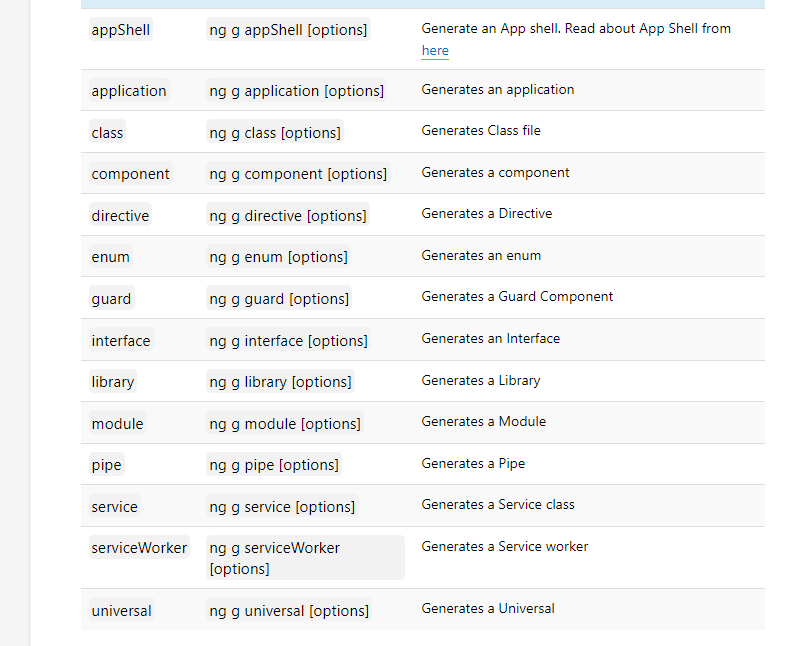


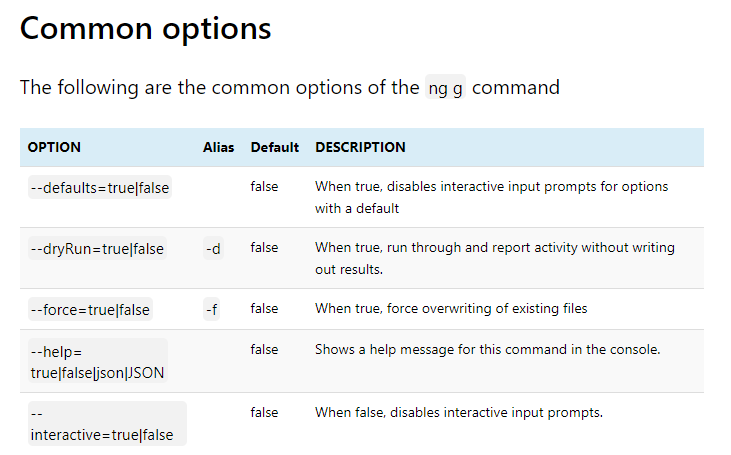




## ng generate







### Component

The following command generates the [component](https://www.tektutorialshub.com/angular/angular-component/).

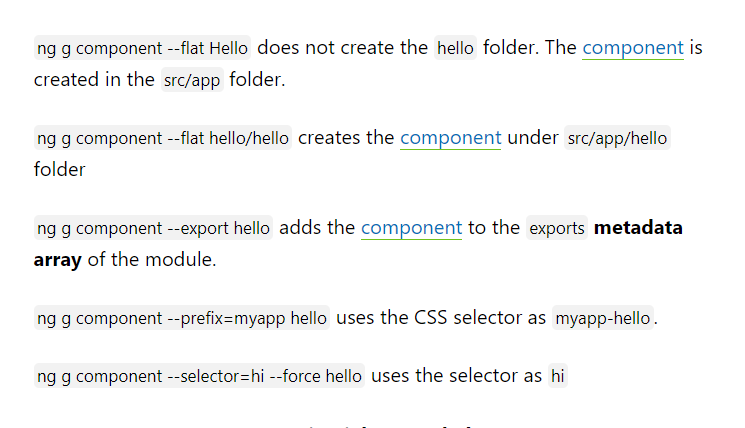
ng g component <name> [options]

Examples

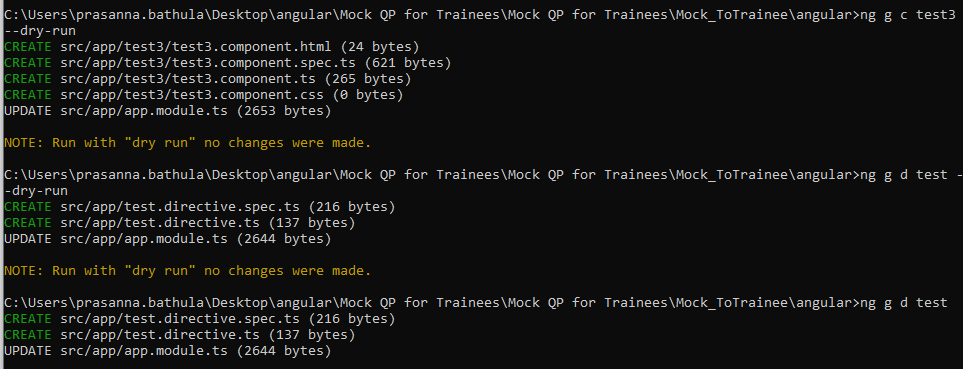
ng g component Hello

The above command does the following

* Creates the Hello folder under the src/app folder
* Create the HelloComponent along with CSS, Spec & Template file under the Hello folder
* Imports the HelloComponent in the root module and adds it to the declarations array
* The CSS selector use the format app-<name>. i.e app-hello

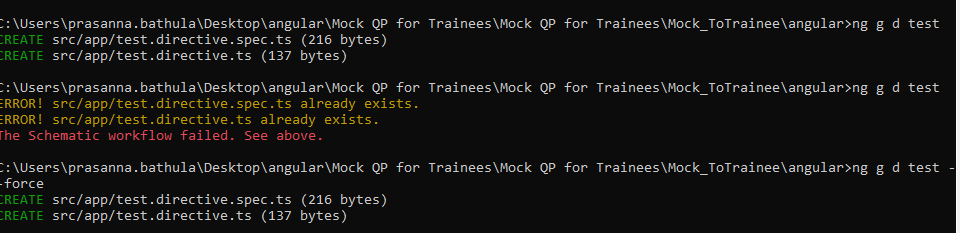


--dry-run:



No Changes were made that means ,When we use –dry-run it just shows structure (what files will be created) but it not creates that in the project. In above diagram showing src/app/test3/all files but they are not created.

--force: If file present it recreates again if not also it creates.



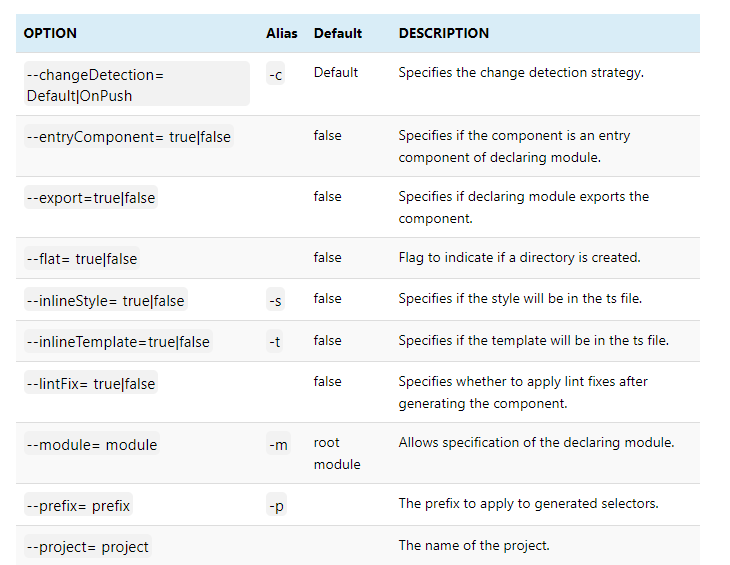
#### generate component inside module

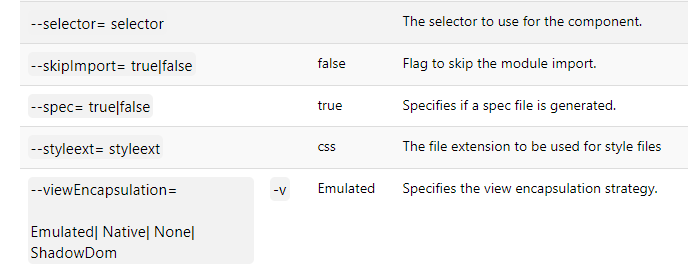
ng g module account generates the code for account module

ng g component --module=account account/hello adds the [component](https://www.tektutorialshub.com/angular/angular-component/) to the account module

ng g component --module=account hello this works provided the current directory is module directory i.e src/app/account

The following is the list of all available options





### 

### Directive

ng g directive <name> [options] is used to generate the directive. For Example ng g directive Some generates the Some.Directive.ts. The following code is generated. It also generates the some.directive.spec.ts



The above command does not create the folder. You can use the ng g directive directive/Some, which will create the directive under the folder directive.

You can use the command ng g directive --flat=false Some, which will create the directive under the folder Some (folder name is same as the directive name)

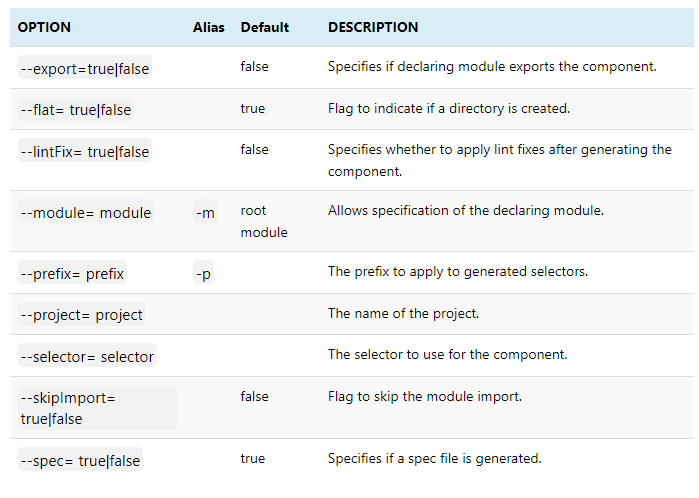
You can use the --force flag to force overwriting of the files, if the files already exists

Use the--module flag to add the directive to a module other than the root module.

Use the --prefix or --selector flag to change the CSS Selctor.

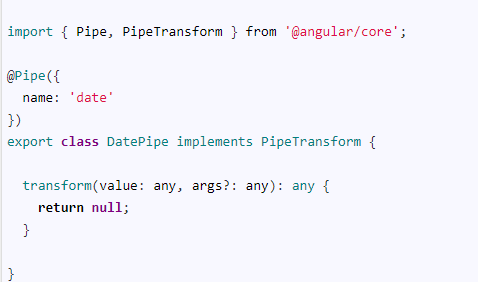
You can use the --export & --skipImport to add the directive to exports and imports metadata of the module

The following is the complete list of available options



### [Pipe](https://www.tektutorialshub.com/angular/angular-pipes/)

ng generate pipe <name> [options] is used to generate the [pipe](https://www.tektutorialshub.com/angular/angular-pipes/). For Example, ng g pipe Date generates the date.pipe.ts. The following code is generated. It also generates the date.pipe.ts.spec.ts



The above command does not create the folder. You can use the ng g pipe pipes/Date, which will create the [pipe](https://www.tektutorialshub.com/angular/angular-pipes/) under the folderpipes.

You can use the command ng g pipe --flat=false Date, which will create the[pipe](https://www.tektutorialshub.com/angular/angular-pipes/) under the folder Date (folder name is the same as the pipe name)

You can use the --force flag to force overwriting of the files, if the files already exists

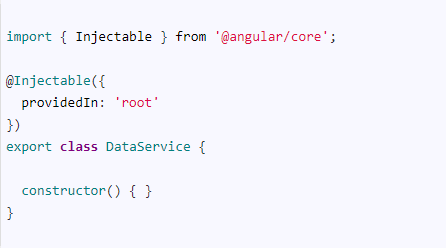
Use the--module flag to add the [pipe](https://www.tektutorialshub.com/angular/angular-pipes/) to a module other than the root module.

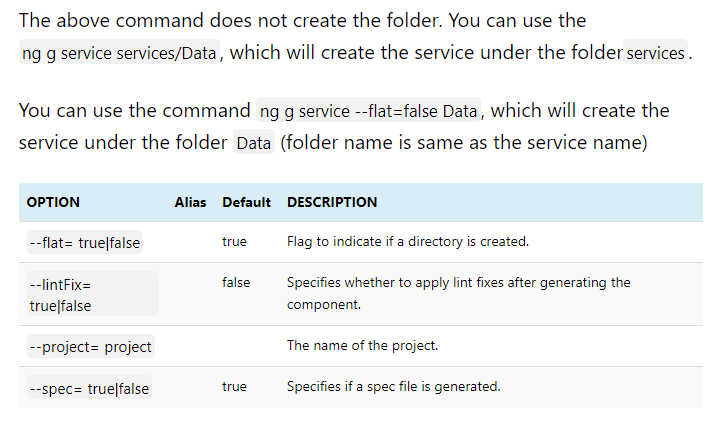
You can use the --export & --skipImport to add the directive to exports and imports metadata of the module

The following is the complete list of available options

### Service

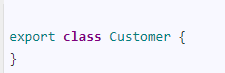
ng generate service <name> [options] is used to generate the service. For Example ng g service Data generates the data.service.ts &data.service.ts.spec.ts. The following is the sample code generated by the command

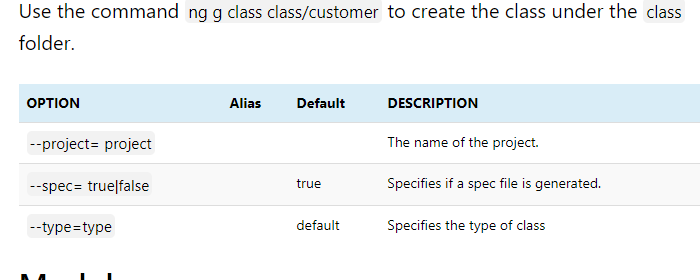




### class

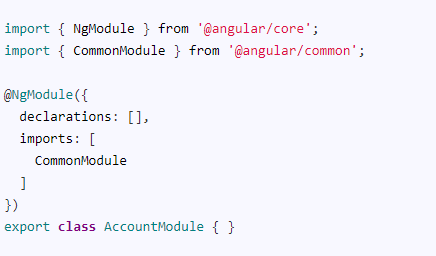
ng generate class <name> [options] is used to generate the class. For Example ng g class customer generates the customer.ts in the current folder. The following is the sample code generated by the command





### Module

use the ng g module <name> [options] command to generate the module. For Example ng g module account generates the account module under the folder src/app/account. The following code is generated

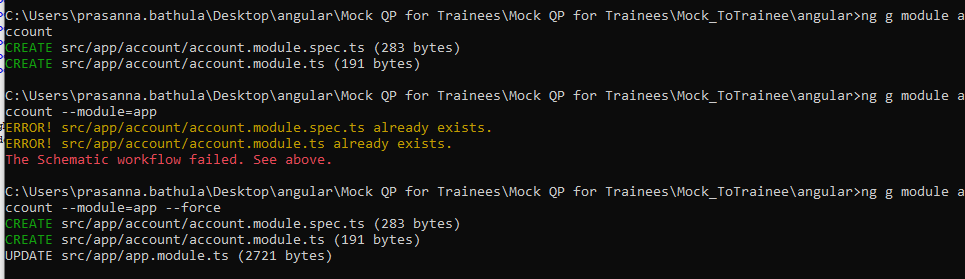


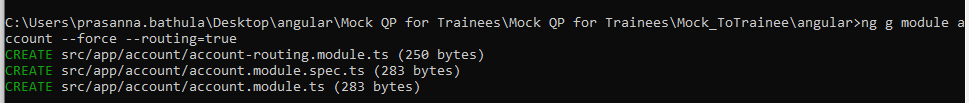
ng g module account --module=app generates the account module and adds it to the imports metadata array of AppModule

ng g module account --routing=true --force. The --force flag overwrites the previously generated files. The --routing=true also creates the account-routing.module. The routes are registered with the forChild(routes)

ng g module account --force --routing=true --routingScope=Root registers the

routes with the forRoot(routes)





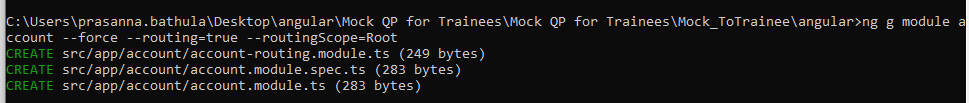
In account rounting module:

@NgModule({

imports: [RouterModule.forChild(routes)],

exports: [RouterModule]

})



const routes: Routes = [];

@NgModule({

imports: [RouterModule.forRoot(routes)],

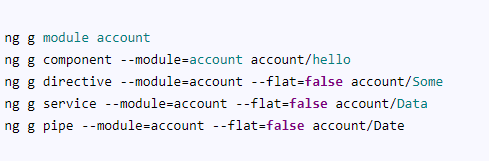
exports: [RouterModule]

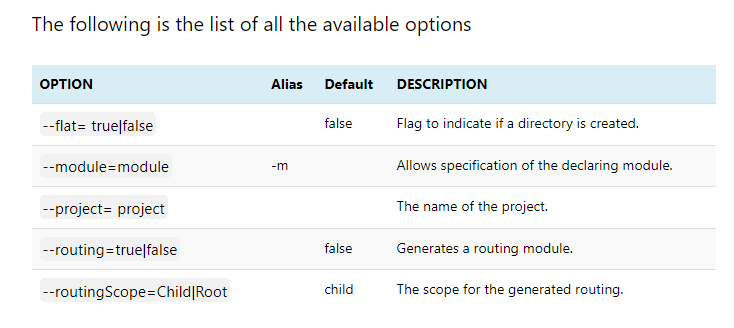
})

#### Adding Components, Pipes, Directives to module

You can add component, directive, [pipe](https://www.tektutorialshub.com/angular/angular-pipes/) & services to module by using the flag --module=[ModuleName] and naming the component as [ModuleName]/[ComponentnNme].

|  |
| --- |
| For Example, The following commands creates the account module and adds the HelloComponent, SomeDirective, DataSerivce & [DatePipe](https://www.tektutorialshub.com/angular/formatting-dates-with-angular-date-pipe/) to the module |





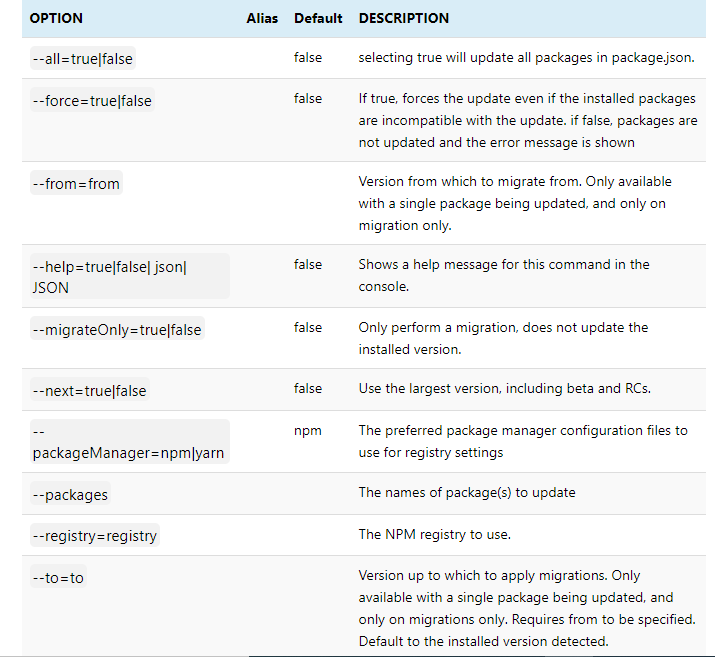
# How to update Angular to latest version

The Angular Team release a new version of the Angular at regular intervals. To keep up with the latest version, we need to update or upgrade our Angular Application.

## ng update

 makes it easier to update the application and its dependencies





## Find out what changed

Before upgrading, you need to know the features that are changed, new features that are added and more importantly the breaking changes that were introduced and  API’s deprecated and or planned to be deprecated

### Find out what’s new in Angular latest version

To find out the list of changes/bug fixes in the new version of Angular, you can read it from the [changelog](https://github.com/angular/angular/blob/master/CHANGELOG.md).

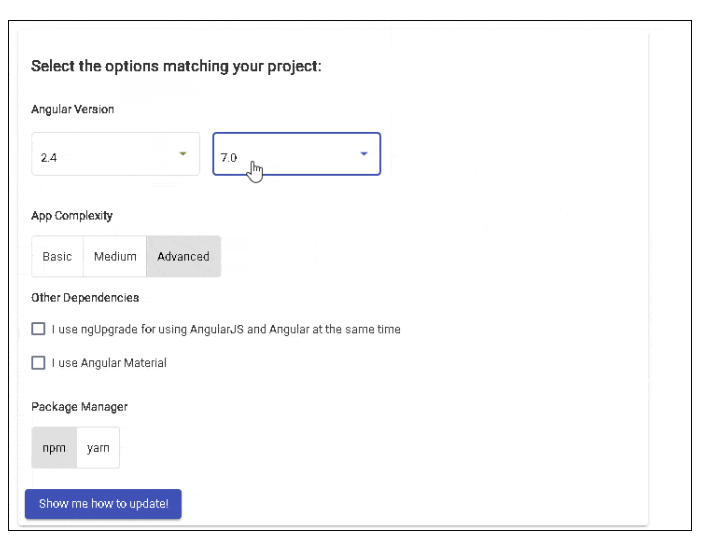
### Find out what needs to be changed

Once you know the list of changes, the next step is to find out what needs to be changed in your app so that you can safely upgrade to the next version. This you can find out from the [Angular Update Guide](https://update.angular.io/).

Once you open the above site, you need to follow these steps and the guide will list you the changes required

* Choose the Current version Angular and the version you wish to upgrade
* Select the App Complexity as Advanced
* Choose other dependencies
* Choose your package manager
* Click on Show me how to update

The Application tells you the steps needed to upgrade.



The above gives the detailed steps needed to update the Angular to the latest version. The list contains three sections. **Before Update**, **During the update**, **After update**. All you needed to is to follow those steps.

## Upgrading using ng update

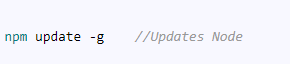
The steps mentioned in the previous section should be sufficient to upgrade the application. The above section contains the ng update commands needed to upgrade the app.

This section explains the steps involved in upgrading the Angular app using ng update

1. Update Node.js to the latest version.
2. Install Angular CLI Globally & Locally
3. Run ng update @angular/cli to update configuration files
4. Update the Core Packages & Dependencies

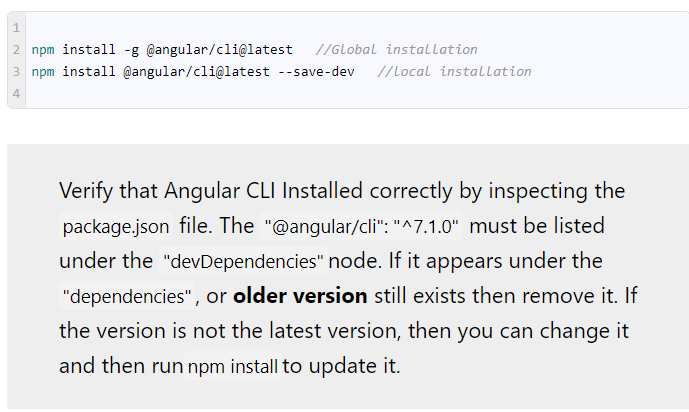
### Update Node.js to the latest version

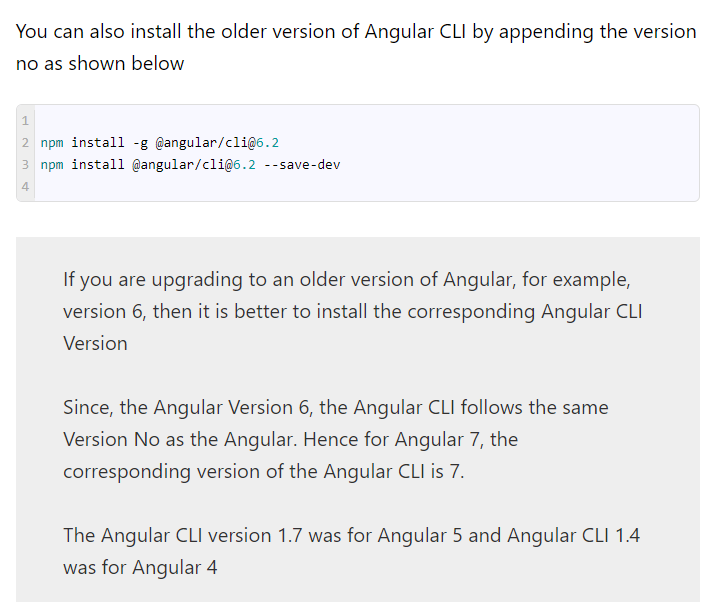
You can run the following command to update the Node.js or visit the [Node.js website](https://nodejs.org/en/) and download the latest version and install it



### Install Angular CLI Globally & Locally

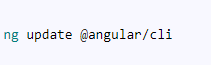
The following command installs the latest version of Angular CLI. The current version as of today is 7.1.0.  Click to find out the [Angular CLI Versions](https://www.npmjs.com/package/@angular/cli?activeTab=versions)





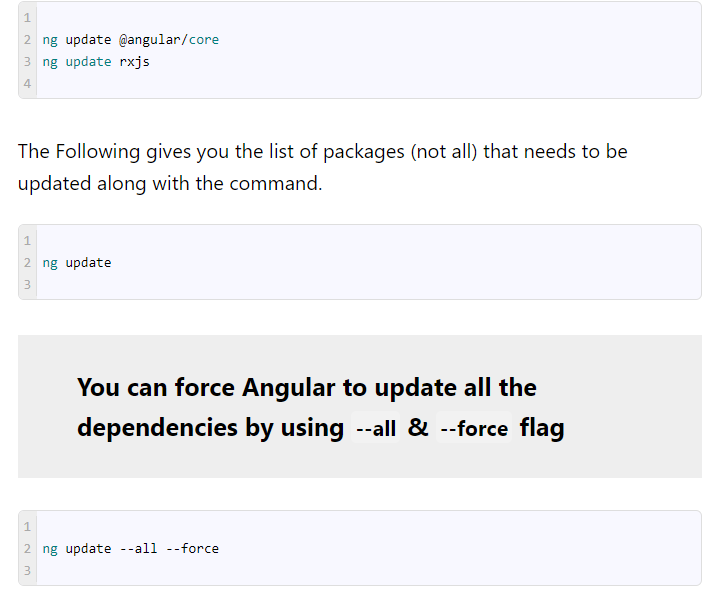
### Run ng update to update configuration files

The next step is to update the various configuration files like angular.json, karma.conf.js etc. This is done by running the following command



### Update core packages & dependencies

The next step is to update the Angular Core packages & Dependencies. The following command updates the Angular core packages & rxjs.



And if you encounter an error after running the above steps, then you can remove the  node\_modules folder & package-lock.json file and run npm install

## Tips to upgrade

### Update to the latest version as and when available

The Angular releases a Major version every six months. The minor updates & bug fixes are released much more frequently. Updating to the latest version once in a month or two makes it easier to upgrade. This keeps the changes to minimum & manageable.

### Keep a watch on Deprecated features

The Major version of Angular may introduce a breaking change. It also may deprecate some of the API. All of these are available at [Changelog](https://github.com/angular/angular/blob/master/CHANGELOG.md). The deprecated changes are usually supported at least for two major versions as per the policy. This gives you at least one year to make the necessary changes. Plan and work on those changes

# Create Multiple Angular Apps in One Project

The Angular CLI since version 6 allows us to create a multi-project workspace to manage multiple Angular apps. We do that first by creating an empty workspace. A Workspace is a collection of Angular apps, projects, or libraries. Later we can add multiple projects to the workspace.

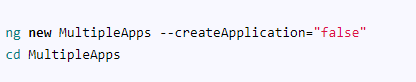
## Advantages

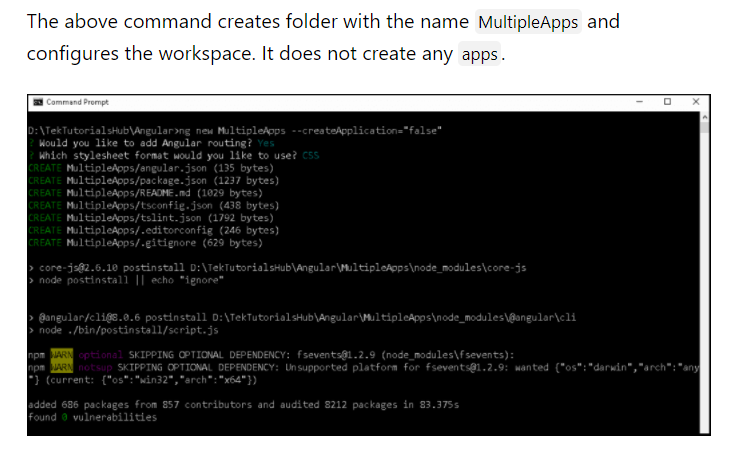
There are several advantages of having Multiple Angular Apps in One Project.

1. One is you do not have to run the time consuming npm install for every app.
2. The node\_modules folder is shared with all the other apps saving disk space.
3. All the apps can be updated to the next version easily.
4. A single source-control repository (such as [git](https://git-scm.com/)).

## Create the Empty Workspace

 The createApplication="false" option introduced in [Angular 7](https://www.tektutorialshub.com/angular-tutorial/) now stops the creation of the initial app. It only creates the workspace





## Add a new Project to Workspace

Now, to create a new app under the workspace, we need to use the ng generate application command The first app created is marked as the default app.



If you use the ng new inside the workspace, it will throw the following error.

**The new command requires to be run outside of a project, but a project definition was found at “D:\MultipleApps\angular.json”**

## Run the App

There are three ways in which you can run the app.

* Use the ng serve gettingStarted
* Use the --project flag ng serve --project="gettingStarted"
* Open the angular.json and locate the defaultProject and change the name of the project to gettingStarted and run ng serve

## Add Another Project to the workspace

To create another app, run the ng generate application again.



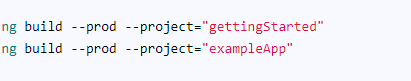
## Run the App

And use the ng serve to run it



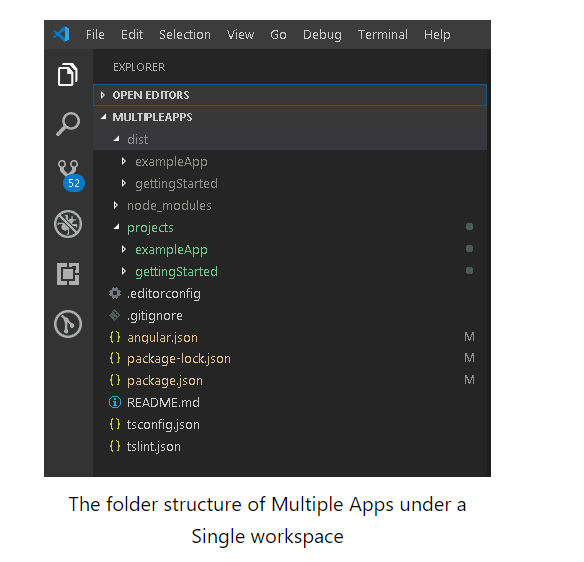
## Building the App for Production

Use ng build to build the app with --project option.



## Folder Structure

The folder structure is similar to the Single App workspace, except for the following



### projects folder

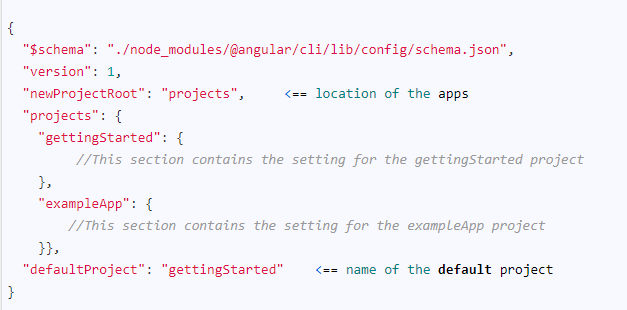
The src folder is gone. Instead, we have a projects folder. Each app we create gets its own folder under the projects folder.

### dist folder

The dist folder now has a folder for each of the new app.

### Angular.json

The angular.json, contains the configuration settings for the workspace. Here is the shortened version of Angular.json for the above code.



newProjectRoot: node points to the location of the projects folder.

projects: contain a section for each app in the workspace. Each section contains configuration for the compiler.

defaultProject: The default project use, when you run the ng serve, ng build etc.

# How to Install Angular and Set Up Development Environment

## 1) Choosing and Installing an Editor

your choice including any editions of **Visual Studio, Eclipse, Atom** etc.

We will be using **Visual Studio code** as our editor. Visual studio code is an open source project and free to use. It can run on Windows, Linux & OS X. It has IntelliSense support, easy integration with Git and support for debugging the applications. It has a great support for Typescript

### Installing Visual Studio Code

## Install Package Manager

To install Angular and dependencies, we are going to **Node Package Manager** or NPM.

### Node Package Manager

It is the Package manager of choice for the Javascript projects. NPM is used to install libraries, Packages & applications from Public repositories.

#### Why it is Required

We are going to use NPM to install [Angular](https://www.tektutorialshub.com/angular-tutorial/), [Typescript](https://www.tektutorialshub.com/typescript-tutorial/), and any other packages/modules required by our application. NPM is can be used to upgrade these packages as and when necessary. Without NPM, we have to download and install all these packages manually.

#### Installing NPM

You need to install NPM on your machine if it is not already installed.You can test whether the NPM is installed or not by using the command prompt



If NPM is installed, then it will return the version No. The current version No of the NPM is 6.13.7.

NPM is installed as part of the NodeJS

#### NPM Configuration

The NPM requires the Package.json file, which should contain the list of modules/packages used in your Application. We need to add all the list of dependencies required by our application the configuration file.

Once the Package.json is created, we can install all the dependencies, by running the following command from the command prompt



## Choosing a Language

We need to choose a language to write our angular application. The Angular Applications must be written in Javascript. This gives us few options, including the current version of Javascript (i.e. ES5), ES2015, Typescript, and Dart, etc.

The [**Typescript**](https://www.tektutorialshub.com/typescript-tutorial/) is a popular choice here. The Angular code is also written using [Typescript](https://www.tektutorialshub.com/typescript-tutorial/).

### What is Typescript

The [Typescript](https://www.tektutorialshub.com/typescript-tutorial/) is a superset of Javascript. The Code written in Typescript cannot be used directly in the web browser. It must be compiled to Javascript before running in the web browser. This process is known as Transpiling.

Typescript is an Open source language. It is built and maintained by Microsoft.

### Why Typescript

The Javascript is the default language of the web. But it does come with few problems. It does not support types. This makes it prone to run-time errors, it also makes it difficult to debug. Javascript does not support namespaces. Code Organisation is another major issue with Javascript code. Typescript Solves all these problems.

### Features of Typescript

[Typescript](https://www.tektutorialshub.com/typescript-tutorial/) also supports Modules, classes, Interfaces, and Generics. This makes Typescript an ideal choice for our Angular Application.

[Typescript](https://www.tektutorialshub.com/typescript-tutorial/) Supports all current specifications of Javascript (i.e. ES5 & ES2015 ). The major advantages of [Typescript](https://www.tektutorialshub.com/typescript-tutorial/) are that you can transpile it to either ES5 or ES2015. Currently, not many browsers support ES2015. So you have to transpile your code to ES5. But in future, as more and more browser becomes ES2015 compliant, you do not have to worry about upgrading your Javascript code. Typescript does that to you behind the scene.

### Installing Typescript

The Typescript is automatically installed by Angular when we install the Angular CLI

Angular CLI also installs and maintains the [Typescript](https://www.tektutorialshub.com/typescript-tutorial/) configuration files, i. e. tsconfig.json and typings.json.

## Installing Angular

We install Angular using the Angular Command Line Interface (Angular CLI)

The [Angular CLI](https://www.tektutorialshub.com/angular/angular-cli-tutorial/) helps us to quickly create an Angular application with all the configuration files and packages in one single command. It also helps us to add features ([components](https://www.tektutorialshub.com/angular/angular-component/), [directives](https://www.tektutorialshub.com/angular/angular-directives/), [services](https://www.tektutorialshub.com/angular/angular-services/), etc) to existing Angular applications.

The [Angular CLI](https://www.tektutorialshub.com/angular/angular-cli-tutorial/) creates the Angular Application and uses Typescript, Webpack ( for Module bundling), Karma ( for unit testing), Protractor ( for an end to end testing).

### Installing Angular CLI

We use the npm command to install the Angular.



## Module Loaders

The initial version of Angular 2 allowed us to choose the Module Loader. But the later versions of Angular comes bundled with the **Webpack module loader**.

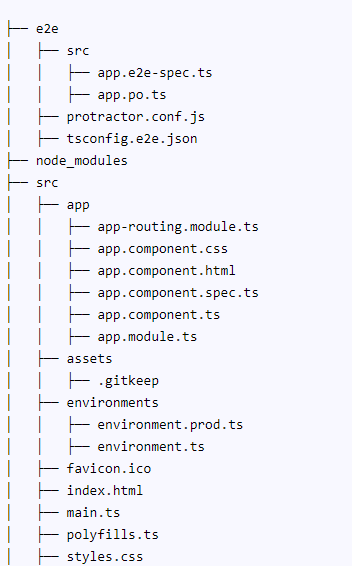
Hence, we do not need to install it.

### Creating a new Angular Application



## Angular project Folder structure

Open the GettingStarted Folder from Visual Studio Code and you will see the following folder structure





The root folder application contains subfolders e2e, node\_modules and src.  It also contains the few configuration files

.editorconfig: This is the configuration file for the Visual Studio code editor. You can visit **http://editorconfig.org** for more information.

.gitignore: Git configuration to make sure autogenerated files are not committed to source control.

angular.json: This is the configuration file for Angular CLI. The older versions of the Angular used the file angular-cli.json

browserslist: Ensures the [compatibility of the Angular app with different browsers](ttps://angular.io/guide/build#configuring-browser-compatibility).

karma.conf.js: The Configuration file for the [karma test runner](https://karma-runner.github.io/).

package.json: The [package.json is an npm configuration file](https://docs.npmjs.com/creating-a-package-json-file/), that lists the third-party packages that your project depends on. We also have[package-lock.json](https://docs.npmjs.com/files/package-lock.json/)

README.md: The Read me file

tsconfig.json, tsconfig.app.json & tsconfig.spec.json are Typescript configuration files. The [tsconfig.json](https://www.typescriptlang.org/docs/handbook/tsconfig-json.html) is the Typescript compiler configuration file. This file specifies the compiler options required for the Typescript to compile (transpile) the project. The tsconfig.app.json is used for compiling the code, while tsconfig.spec.json for compiling the tests

tslint.json: [tslint](https://www.npmjs.com/package/tslint) is a static code analysis tool. We use this to check Typescript code quality. To check if TypeScript source code complies with coding rules. TSLint checks your TypeScript code for readability, maintainability, and functionality errors

### e2e

This folder contains the files required for end to end tests by protractor. Protractor allows us to test our application against a real browser. You can learn more about protractor from this [link](http://www.protractortest.org/#/)

### node\_modules

All our external dependencies are downloaded and copied here by NPM Package Manager.

### src

This where our application lives.

## app folder

The Angular CLI has created a simple application, which works out of the box. It creates the root component, a root module, a unit test class to test the component. Now let us see each component of the application one at a time

The src folder is where our application lives.

### The Component

The app.component.ts is the component that is added to the project by Angular CLI. You will find it under the folder app/src

### Root Module

Angular organizes the application code as [Angular modules](https://www.tektutorialshub.com/angular/angular-modules/). The Modules are closely related blocks of code in functionality. Every application must have at least one module.

The Module, which loads first is the[root Module](https://www.tektutorialshub.com/angular/angular-modules/#root-module). This Module is our root module.

The root module is called app.module.ts.

### App Routing Module

The AppRoutingModule in the file app-routing.module.ts defines the Routes of the application. These Routes tells Angular how to move from one part of the application to another part or one View to another View.

The Routes defined in the constant const routes: Routes = [];, which is empty

This Module is defined as a separate Module and is imported in  AppModule.

### Bootstrapping our root module

The app.component.html is the file, which we need to show it the user. It is bound to AppComponent component. We indicated that the AppComponent is to be bootstrapped when AppModule is loaded

Now we need to ask the Angular to load the AppModule when the application is loaded. This is done in main.ts file

The main.ts file is found under the src folder. The code is as follows.



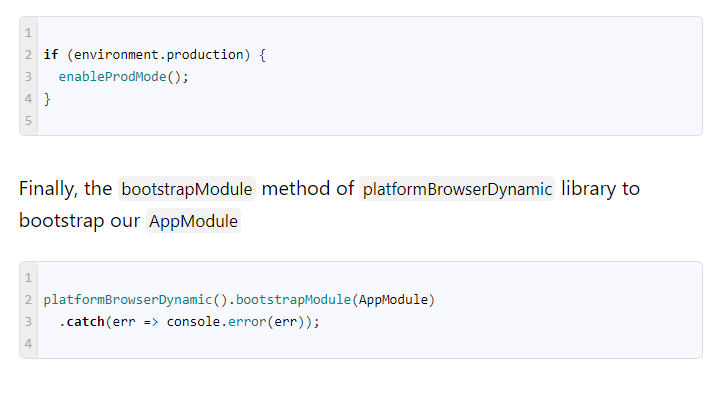
The second line of the import is platformBrowserDynamic library. This library contains all the functions required to bootstrap the angular application.

We also import is enableProdMode from @angular/core library. The Angular’s code by default runs in development mode. The development mode runs few assertions and checks, which helps in debugging the application. The enableProdMode enables the production only if the current build [Angular environment](https://www.tektutorialshub.com/angular/angular-environment-variables/) is production

We also need to import our AppModule.

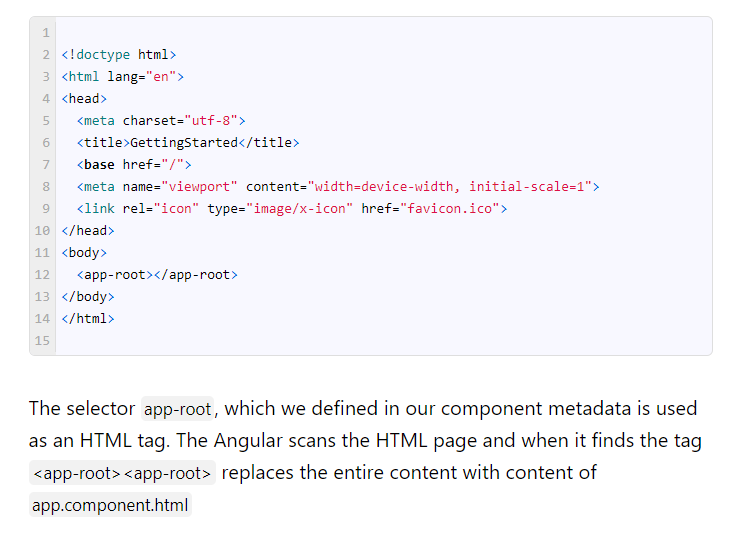
The last import is the environment, which is in the folder src/environments. The file environment.ts contains the contents for the current environment. The development environment uses the environment.ts file. When you build the project for the production environment, then the environment.prod.ts will be used. The environment are configured in angular.json`

In the next line, our code checks the [environmental variable](https://www.tektutorialshub.com/angular/angular-environment-variables/) and enable the production mode, if the environment is production.



### index.html

Index.html is the entry point of our application.



## Other files & folders

We have a few other files under the app folder.

### Assets

A folder where you can put images and anything else to be copied wholesale when you build your application.

### Environments

The environment folder is where we define [environment variables](https://www.tektutorialshub.com/angular/angular-environment-variables/) for various build setups. The build setups can be development, production, testing & staging. The Angular has creates two build environments out of the box. One is development, which is the default and the other one in Production. The two files environment.ts is the default for development and the environment.prod.ts is for the production build.

### polyfills.ts

Different browsers have different levels of support of the web standards. Polyfills help normalize those differences. You can read more about [browser support guide](https://angular.io/guide/browser-support) about the browsers supported by Angular

### styles.css

Your [Angular global styles](https://www.tektutorialshub.com/angular/angular-global-css-styles/) go here. Most of the time you’ll want to have [local styles in your components](https://www.tektutorialshub.com/angular/angular-component-styles/) for easier maintenance, but styles that affect all of your apps need to be in a central place.

### test.ts

This is the main entry point for your unit tests. It has some custom configuration that might be unfamiliar, but it’s not something you’ll need to edit.

/\*

The above command compiles the Angular application and invokes the **Webpack development server**. The server keeps a watch on our project folder. If you make any changes in the code, it compiles the project again.

You can also use npm start.

The Webpack Development server listens on HTTP Port 4200. Hence open the browser and type **http://localhost:4200/** and you will see **GettingStarted app is running** displayed on the browser.

\*/

# Bootstrapping in Angular: How It Works Internally

 We use ng new to [create a new Angular project](https://www.tektutorialshub.com/angular/angular-create-first-application/). It generates lots of boilerplate codes. It also configures the [Typescript](https://www.tektutorialshub.com/typescript-tutorial/), Webpack, Karma, & Protractor. The app, when run displays a simple HTML page with several useful links to Angular. Now let us break up this app and look at what happens when the app starts until it displays the HTML page

### What is a Bootstrapping

Bootstrapping is a technique of initializing or loading our Angular application.

 The Angular takes the following steps to load our first view.

1. Index.html loads
2. Angular, Third-party libraries & Application loads
3. Main.ts the application entry point
4. Root Module
5. Root Component
6. Template

## Index.html Loads First

Web apps need a starting point. Index.html is usually the first page to load.



There are no javascript files Neither stylesheet file.Contains only below



How do Angular loads ?.  To Find out, let us build our application

### Building Application

To run our application, we use

npm start (npm start command actually translates into ng serve.)

ng serve does build our application but does not save the compiled application to the disk. It saves it in memory and starts the development server.

We use ng build to build our app.



 This will build and copy the output files to the dist folder

Use ng build --prod to build and distribute the app for production. For testing/debugging use ng build. The production build optimizes, minimize and uglify the code.

Now open the dist and open the index.html.



You can see that the compiler included five script files. They are runtime, polyfills, styles, vendor, & main. All these files have two versions one is es5 & the other one es2015

Since the Angular 7**,**we have new feature called **conditional polyfill loading**. Now Angular builds two script files, one for es2015 & another for es5. The es2015 (es6) is for modern browser and es5 is older browsers, which do not support the new features of es2015.  
  
Note the nomodule attribute, which tells the modern browser to ignore the script and do not load it. Hence es5 scripts are not loaded in the modern browsers

runtime.js: Webpack runtime file  
polyfills.js – Polyfill scripts for supporting the variety of the latest modern browsers  
styles.js – This file contains the global style rules bundled as javascript file.  
vendor.js – contains the scripts from the Angular core library and any other 3rd party library.  
main.js – code of the application.

#### What is Webpack?

Webpack is a bundler. it scans our application looking for javascript files and merges them into one ( or more) big file. Webpack has the ability to bundle any kind of file like JavaScript, CSS, SASS, LESS, images, HTML, & fonts, etc.

The [Angular CLI](https://www.tektutorialshub.com/angular/angular-cli-tutorial/) uses Webpack as a module bundler. Webpack needs a lot of configuration options to work correctly. The Angular CLI sets up all these configuration options behind the scene.

The Webpack traverses through our application looking for javascript and other files and merges all of them into one or more bundles. In our example application, it has created five files.

## Application Loads

So when index.html is loaded, the Angular core libraries, third-party libraries are loaded. Now the angular needs to locate the entry point.

### Application Entry point

The entry point of our application is main.ts. You will find it under the src folder.

### angular.json

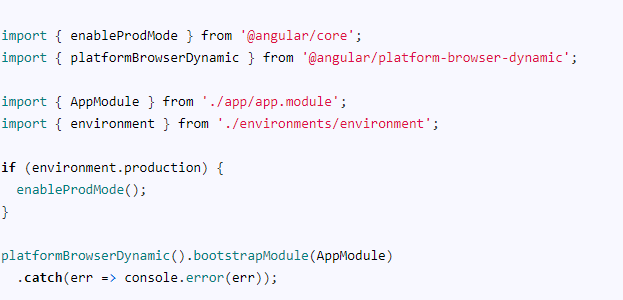
The Angular finds out the entry point from the configuration file angular.json.

The angular-cli.json was the configuration file in Angular 5 and before. It is now angular.json since the version Angular 6.



The main entry under the node projects -> GettingStarted -> architect -> build -> options points towards the src/main.ts. This file is the entry point of our application.

## main.ts Application entry point

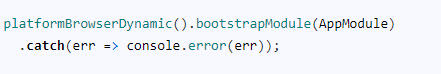


### What is platformBrowserDynamic

platformBrowserDynamic is the module, which is responsible for loading the Angular application in the desktop browser.



The AppModule is the Root Module of the app.  Every application built in Angular must have at least one module. The module, which is loaded first when the application is loaded is called a root module.



TheplatformBrowserDynamic loads the root module by invoking the bootstrapModule and giving it the reference to our Root module i.e AppModule

## Root Module

The angular bootstrapper loads our root module AppModule



The root module must have at least one root component. The root component is loaded, when the module is loaded by the Angular.

##### Bootstrap

The component that angular should load, when this Angular Module loads. The component must be part of this module. We want AppComponent load when AppModule loads, hence we list it here.

The Angular reads the bootstrap metadata and loads the AppComponent

## Component

Finally, we arrive at AppComponent, which is the root [component](https://www.tektutorialshub.com/angular/angular-component/) of the AppModule.



This property contains an HTML template, which is going to be displayed in the browser. The template file is app.component.html

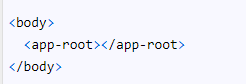
##### Selector

This property specifies the CSS Selector, where our template will be inserted into the HTML. The CSS Selector in our code is app-root

## Template

The AppComponent defines the template as app.component.htmland the CSS Selector is app-root

Our index.html already have the app-root CSS selector defined



The Angular locates app-root in our index.html and renders our template between those tags.