**ANGULAR JS WORKSHOP**

v8 engine of chrome.

developed c++ around v8 engines to access file system sof server, networks and ports connectivity.

nodeJS is JS runtime env.

installing required packages using npm.

<https://www.npmjs.com/>

instaling bootstrap in with npm:

<https://www.npmjs.com/search?q=bootstrap>

Angular->static webpage, single page, google product

**starting build project:**

1. ng new project\_name

2. yes-->yes

3. use css and press enter.

4. angular.json has all the settings required by angular.

5. package.json defines all the dependencies of the projects.

"devDependencies" and "dependencies" are required when the project is deployed.

all there are in node\_modules folder.

6. inside===> package.json

"dependencies": {

"@angular/animations": "^15.1.0",

this is the folder node\_modules/@angular/animations....

7. package-lock.json==> defines all the dependencies that are inside node\_modules.

8. all tsconfig files are typescript configurations.

9. further going to minimal version-> remove testing folders, git.

10. develop temporary server==> **ng serve**

//starting server.

11. **@ngmodule in src/app/app.module.ts** is a decorator. which defines app module has some declerations depends on some imports.

using external modules we go for providers[].

12. app.component.spec.ts is for testing purpose.

13. app.component.ts has all class files.

14. app.component.html will have alll html files.

15. component.ts file is making link betn html and css.

16. cmd==> **ng new contactApp --minimal --skip-git --skip-tests**

//skips git and tests that we don't need.

//it skips the template html and css files and urls.

17. go to angular.json ==> configuration of angular application

18. line 10 ==>

"inlineTemplate": true,

"inlineStyle": true,

we can simply remove these two lines(10, 11).

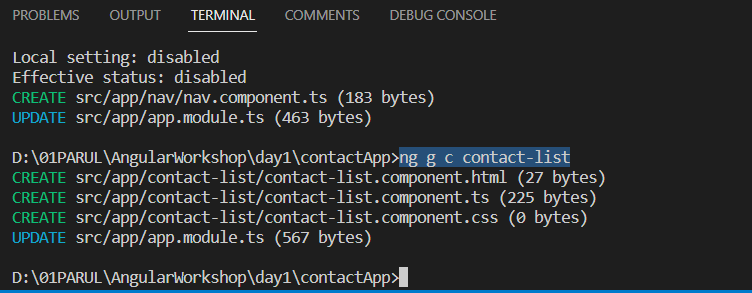
this will not show inline styles when we create a new component.

19. CREATING A COMPONENT.

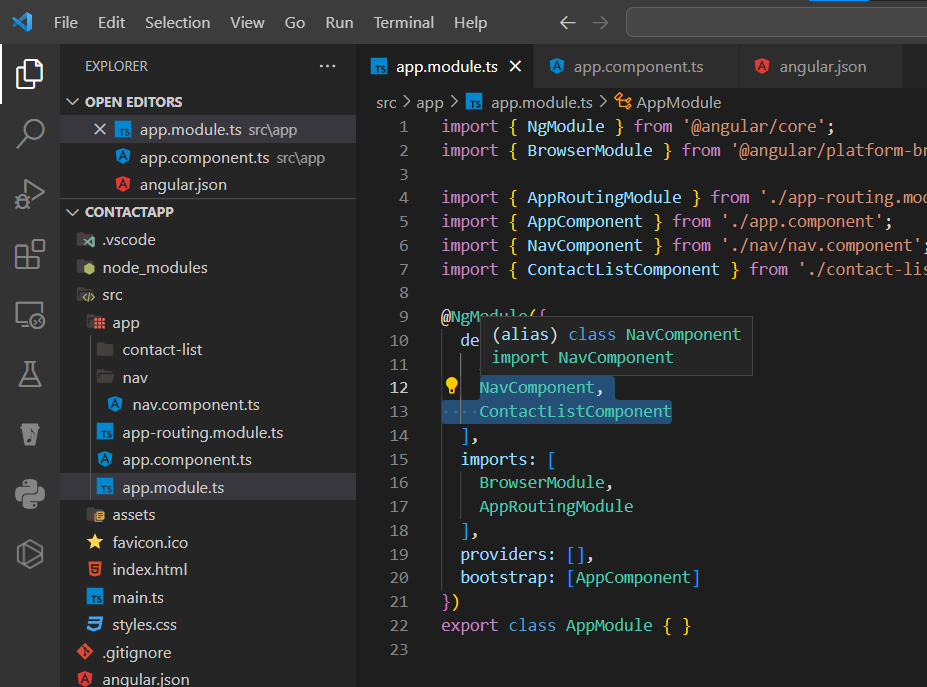
Cmd🡺 **ng generate component component\_name**

Short🡪 **ng g c component\_name**

//creates a new folder with component\_name inside src/app/comp\_name

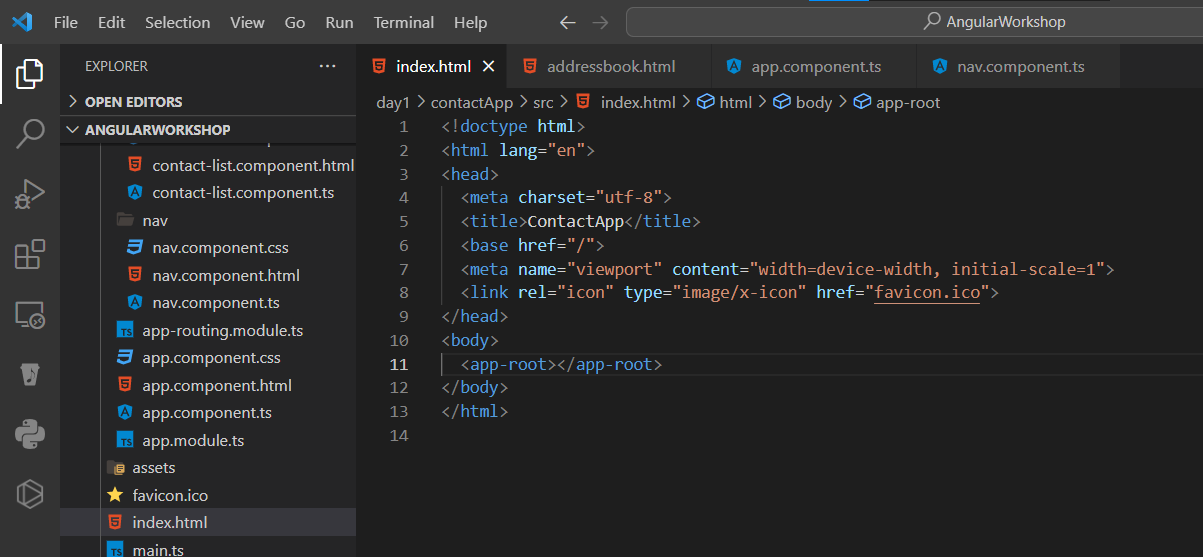


20. after creating a component it will show in ***app.module.js***

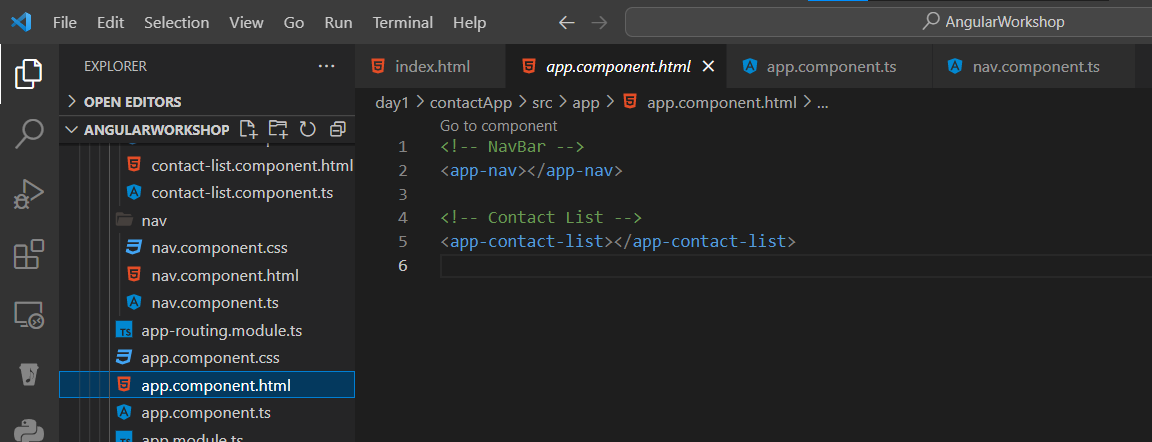


Default browser loads the **index.html**

It says whatever is in app-nav selector that’s loaded.

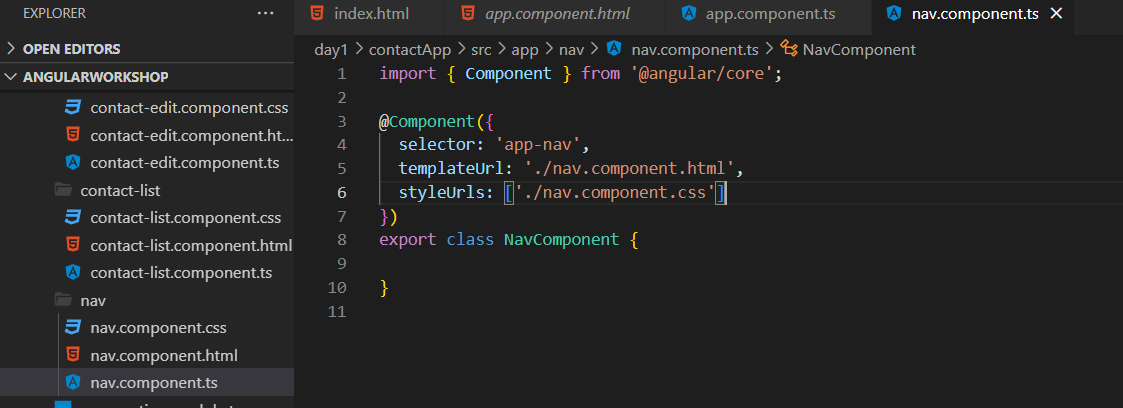


Above **app-root** tag is calling the root **app.component.html** file.



In above image, **app-nav** tag is the selector from **nav.component.ts** inside nav folder.

**src/app/nav/nav.component.ts**

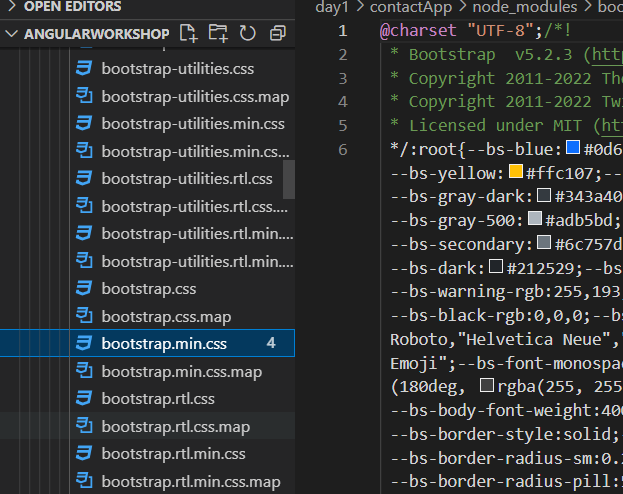
****

**\*\*\*After installing any package just check in package.json just to confirm if it’s added.**

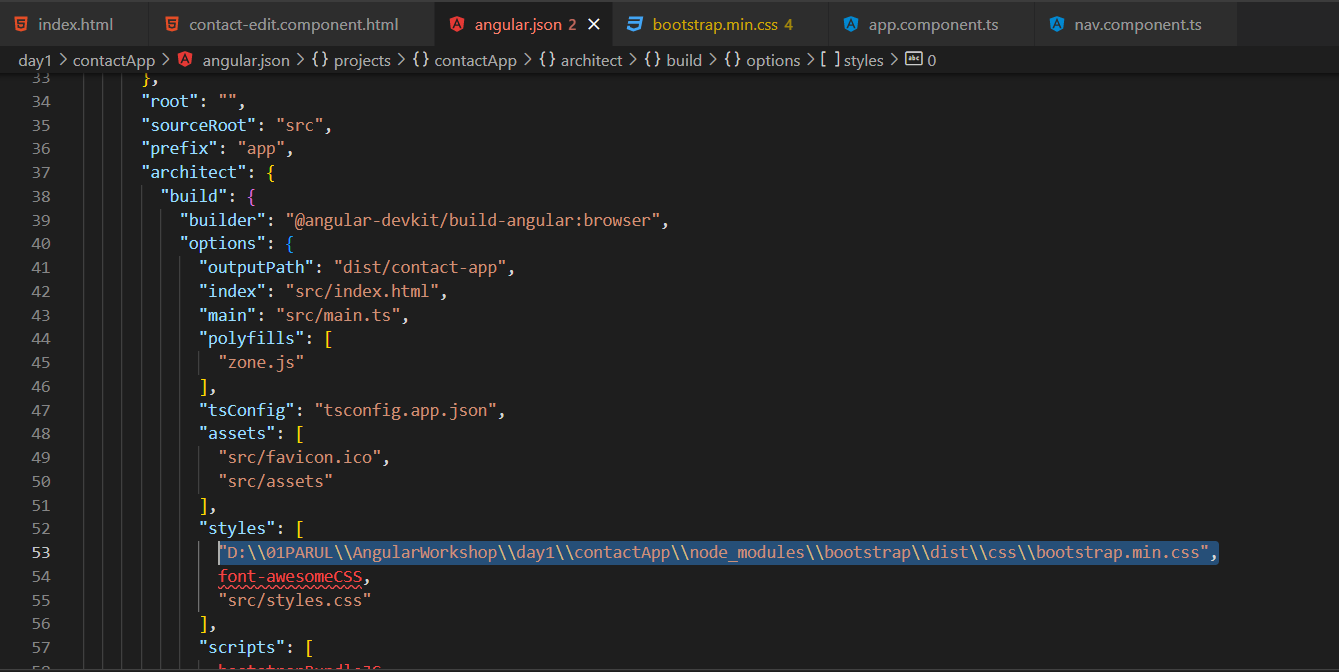
🡪install bootstrap inside project folder as:

Cmd🡪npm I bootstrap

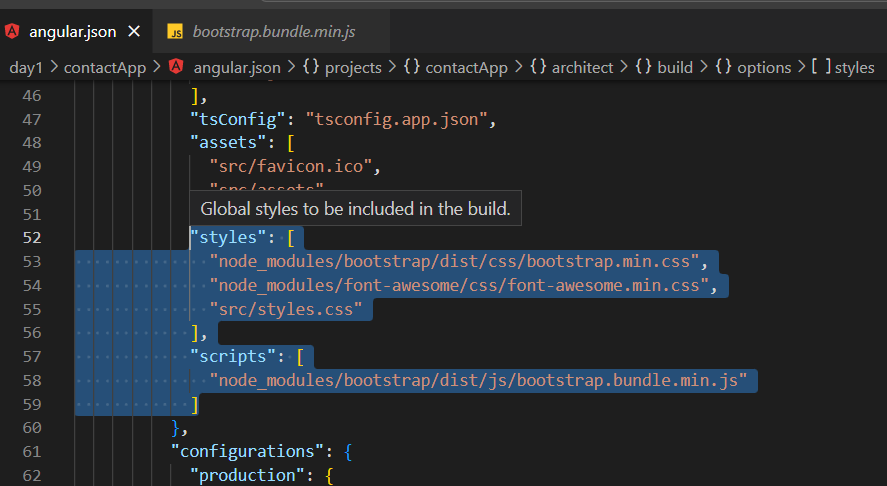
Cmd🡪npm I font-awesome



🡪This is the file we will be including inside **angular.json** file. Above image is the bootstrap.min picture.



Similarly add **font-awesome.min.css** from node\_modules*…..(paths given in below picture selected part.)*



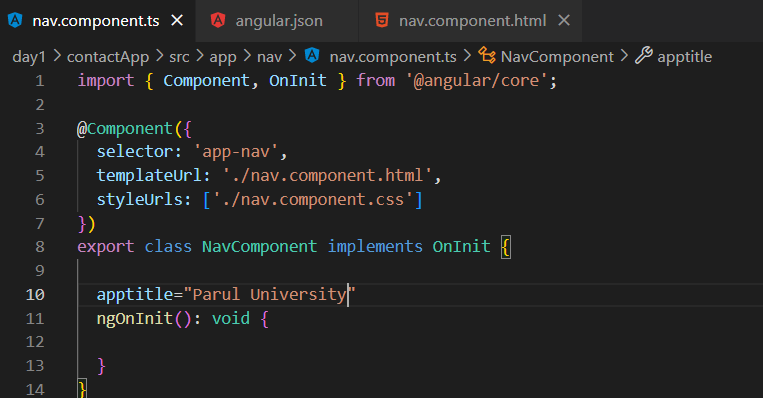
If there is a same class with diff functions and going to display same class on same page then class conflicts.

This problem is resolved with encapsulation.

**STRING INTERPOLATION**

<strong>{{ apptitle }}</strong>

This apptitle is a variable inside nav.component.ts and written inside nav.component.html



**PIPES:**

Pipe operators for currency symbols; 🡪 <https://angular.io/api/common/CurrencyPipe>

**Structural directives:**

Ng class is a attribute

\*property binding

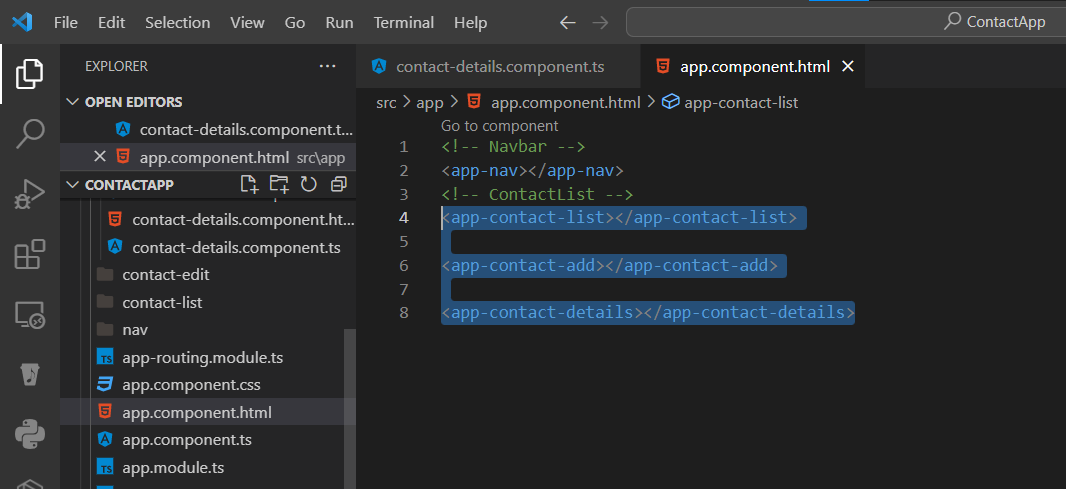
\*event binding

\*

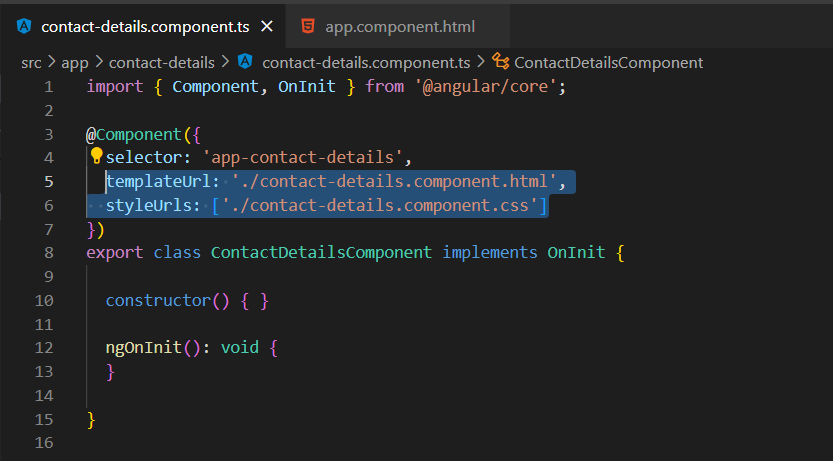
**🡪DEFAULT** the browser access the **src.app.component.html** file. So to display **contact-details,** **contact-add** and every other components that we have created, we use the text inside selector.



🡪And in **app.component.html,** write the selector as tag.



* This below shows that the app-contact-details will have the html file and css files.



* Binding to a variable in html: with [] big bracket.

[disabled]=”!haverights” this wil create a haveright variable.

* Any .ts file works as JS file.
* Properties are binded into [] big bracket.
* Events are binded into () small bracket.
* Two way binding:

Suppose there is a form, default name is set as **‘aaa’** and that may be updated by the user.

While the user updates the form name that get’s affected in the class too. Inside **.ts** file.

Example:

**[(ngModel)]=’fname’** 🡪 this is inside the html file in a input tag.

* Go to the **.ts** file create a fname variable and set a name **‘aaa’** that wil be displayed as in above step.

