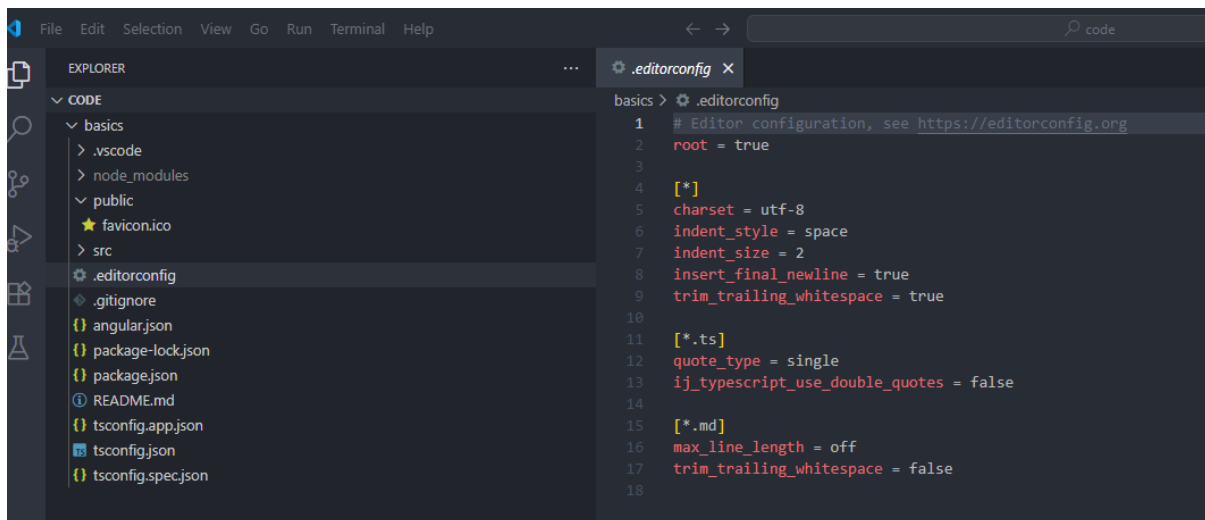


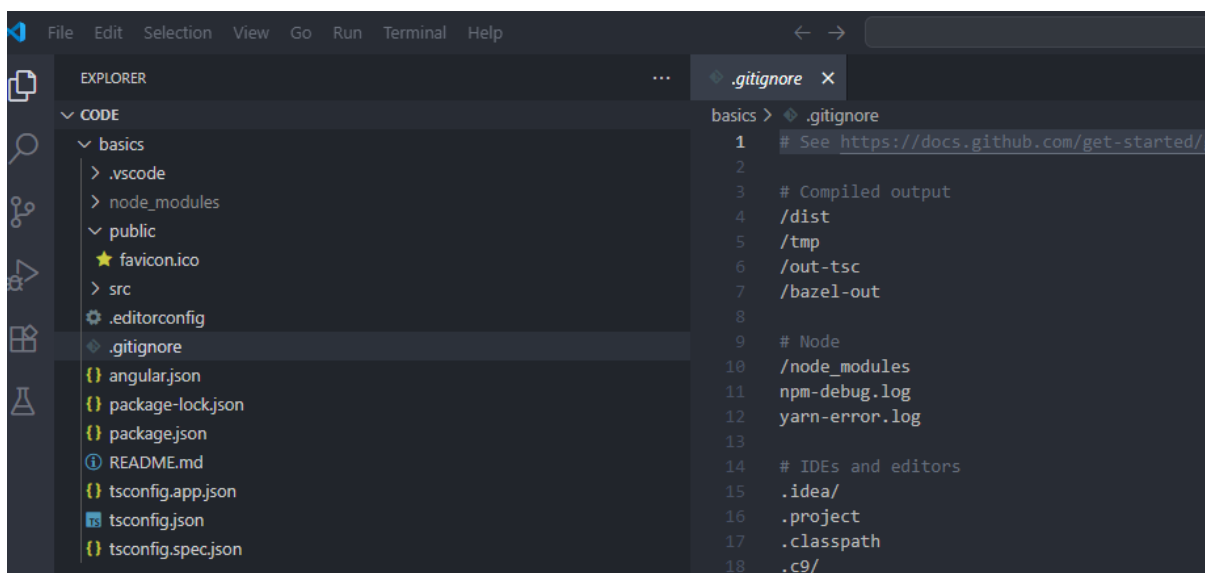
.editorconfig



```
1 # Editor configuration, see https://editorconfig.org
2 root = true
3
4 [*]
5 charset = utf-8
6 indent_style = space
7 indent_size = 2
8 insert_final_newline = true
9 trim_trailing_whitespace = true
10
11 [*.ts]
12 quote_type = single
13 ij_typescript_use_double_quotes = false
14
15 [*.md]
16 max_line_length = off
17 trim_trailing_whitespace = false
18
```

This is a configuration file for editors. → Used for Teams

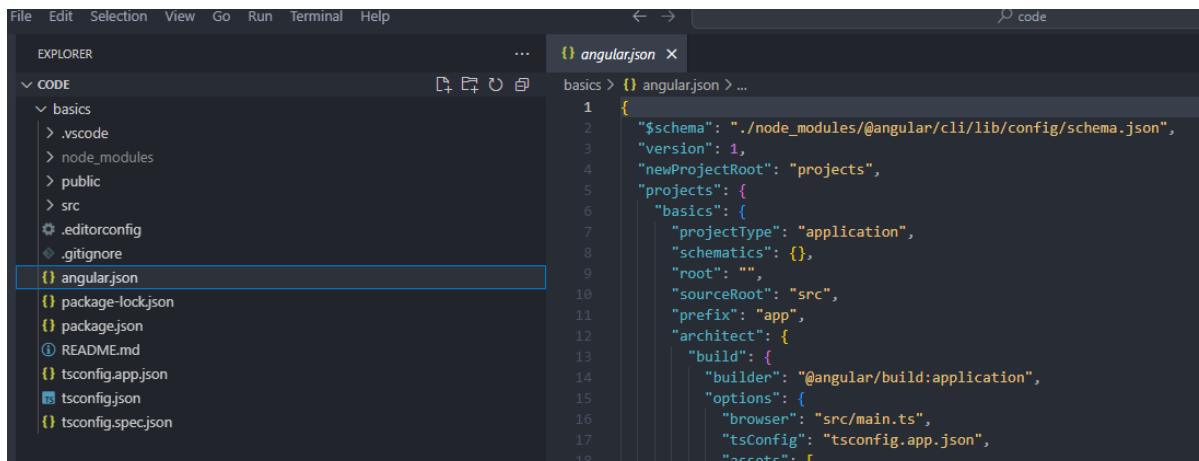
.gitignore



```
1 # See https://docs.github.com/get-started/g
2
3 # Compiled output
4 /dist
5 /tmp
6 /out-tsc
7 /bazel-out
8
9 # Node
10 /node_modules
11 npm-debug.log
12 yarn-error.log
13
14 # IDEs and editors
15 .idea/
16 .project
17 .classpath
18 .c9/
```

This file will tell git to ignore a list of files when performing commitments.

angular.json



```

1 {
2   "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
3   "version": 1,
4   "newProjectRoot": "projects",
5   "projects": {
6     "basics": {
7       "projectType": "application",
8       "schematics": {},
9       "root": "",
10      "sourceRoot": "src",
11      "prefix": "app",
12      "architect": {
13        "build": {
14          "builder": "@angular/build:application",
15          "options": {
16            "browser": "src/main.ts",
17            "tsConfig": "tsconfig.app.json",
18            "assets": [

```

This file configures the workspace of our project; we can change how Angular builds our project for production and development.

Unlike the other files, I want to dive into the options of this file, I am going to fold some of these options for readability:

The first option is called: Schema – Angular is capable of generating files for our projects aside from the initial starter files, it needs instructions for generating those files, this option is a path to provide instructions on the various files generated with Angular, this file is also used for existing code whenever new changes are introduced.

The next option is called Version – This option will contain the version of the configuration. We are currently using version 1 of the configuration. If the Angular team decides to change how the configuration file is formatted, this value would be higher.

Afterward, we have the **newProjectRoot** option, believe it or not, we can manage multiple Angular applications at once, you may want to do this if you have a large app that needs to be broken down into several smaller apps, for example, you may have an application for customers and another for administrators. Code related to administrative actions shouldn't be loaded with customer actions. Both apps may feed into the same API, therefore they share a relationship, we can break this type of app into two apps.

The new project root option will be the name of the directory for hosting additional apps. For example, if we create a new project from within this Angular project, it will be placed inside the **projects** directory.

The next option is called **projects** – it'll contain a list of projects currently registered with Angular, we've created one app so far, which is the **basics** app. Inside this object, we'll have more configuration options for this specific project. The most crucial option is called **architect**:

```

angular.json X
basics > {} angular.json > {} projects > {} basics > {} architect > {} build
1  {
2    "$schema": "../../node_modules/@angular/cli/lib/config/schema.json",
3    "version": 1,
4    "newProjectRoot": "projects",
5    "projects": {
6      "basics": {
7        "projectType": "application",
8        "schematics": {},
9        "root": "",
10       "sourceRoot": "src",
11       "prefix": "app",
12       "architect": {
13         "build": {
14           "builder": "@angular-devkit/build-angular:browser",
15           "options": {
16             "outputPath": "dist/basics",
17             "index": "src/index.html",
18             "main": "src/main.ts",
19             "polyfills": "src/polyfills.ts",
20             "tsConfig": "src/tsconfig.app.json",
21             "assets": [
22               "src/favicon.ico",
23               "src/assets"
24             ],
25             "styles": [
26               "src/styles.css"
27             ],
28             "scripts": []
29           },
30           "configurations": {
31             "production": {
32               "optimization": true,
33               "outputPath": "dist/basics",
34               "sourceMap": false,
35               "namedChunks": false,
36               "aot": true,
37               "extractLicenses": true,
38               "vendorChunk": false,
39               "buildOptimizer": true
40             }
41           }
42         },
43         "serve": {
44           "builder": "@angular-devkit/build-angular:dev-server",
45           "options": {
46             "browserTarget": "basics:build"
47           }
48         },
49         "extract-i18n": {
50           "builder": "@angular-devkit/build-angular:extract-i18n",
51           "options": {
52             "browserTarget": "basics:build"
53           }
54         },
55         "test": {
56           "builder": "@angular-devkit/build-angular:karma",
57           "options": {
58             "main": "src/test.ts",
59             "polyfills": "src/polyfills.ts",
60             "tsConfig": "src/tsconfig.spec.json",
61             "karmaConfig": "src/karma.conf.js",
62             "styles": [
63               "src/styles.css"
64             ],
65             "scripts": []
66           }
67         }
68       }
69     }
70   }
71 }

```

The **architect** options contain a list of commands for building the project. Building is the process of compiling our code, for example, we need to build our TypeScript code into JavaScript code if we want to run our app in the browser. Angular needs to build our project differently based on what we plan on doing with the app. If we want to develop the application, we will use the **serve** command. If we want to ship our app to production, we would use the **build** command. As you can probably tell, each of these objects correlate to a command in the CLI. These commands will build the project differently.

```

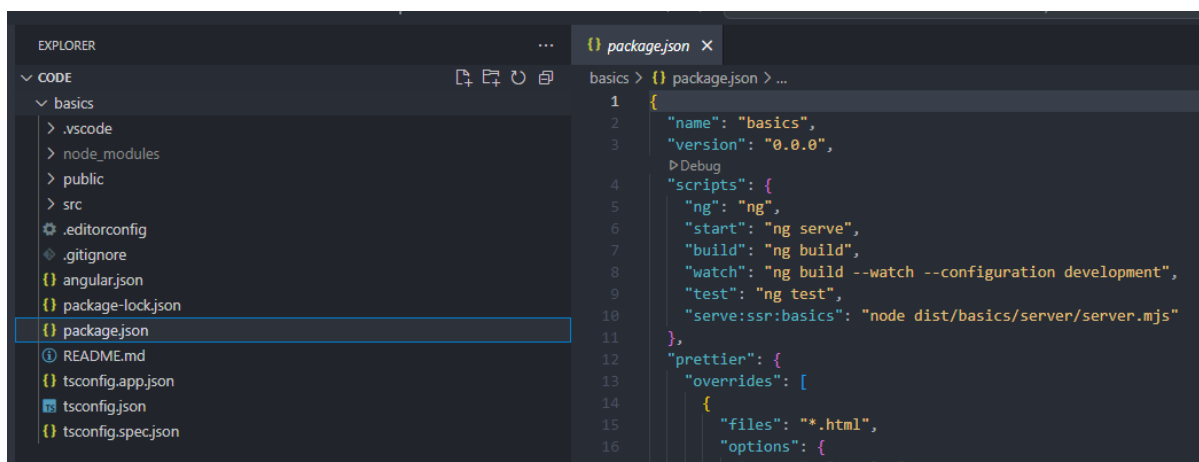
2    "architect": {
3      "build": {
4        "builder": "@angular-devkit/build-angular:browser",
5        "options": {
6          "outputPath": "dist/basics",
7          "index": "src/index.html",
8          "main": "src/main.ts",
9          "polyfills": "src/polyfills.ts",
10         "tsConfig": "src/tsconfig.app.json",
11         "assets": [
12           "src/favicon.ico",
13           "src/assets"
14         ],
15         "styles": [
16           "src/styles.css"
17         ],
18         "scripts": []
19       },
20       "serve": {
21         "builder": "@angular-devkit/build-angular:dev-server",
22         "options": {
23           "browserTarget": "basics:build"
24         }
25       },
26       "extract-i18n": {
27         "builder": "@angular-devkit/build-angular:extract-i18n",
28         "options": {
29           "browserTarget": "basics:build"
30         }
31       },
32       "test": {
33         "builder": "@angular-devkit/build-angular:karma",
34         "options": {
35           "main": "src/test.ts",
36           "polyfills": "src/polyfills.ts",
37           "tsConfig": "src/tsconfig.spec.json",
38           "karmaConfig": "src/karma.conf.js",
39           "styles": [
40             "src/styles.css"
41           ],
42           "scripts": []
43         }
44       }
45     }
46   }

```

If we want to configure the building process, we can do so through the respective option. The **build** command will build the project for production. The **serve** command will build the project for development. The **extract-i18n** command will extract translations from a project. Lastly, the **test** command will build the project for testing.

We aren't going to be running through every command provide by Angular. However it's always good to know where you can modify the behavior of a command.

package.json



It contains a list of dependencies for our project, if we look inside the scripts object:

```
"scripts": {
  "ng": "ng",
  "start": "ng serve",
  "build": "ng build",
  "watch": "ng build --watch --configuration development",
  "test": "ng test",
  "serve:ssr:basics": "node dist/basics/server/server.mjs"
},
```

We'll find some commands for starting and building the project.

tsconfig files:

There are three TypeScript configuration files:

- tsconfig.app.json – Runs when we're compiling our application
- tsconfig.json – Both tsconfig.app.json and tsconfig.spec.json files extend the tsconfig.json file. The extends option in those files allow us to Import settings from another file.

```

{} tsconfig.spec.json X
basics > {} tsconfig.spec.json > ...
1  /* To learn more about Typescript configurati
2  /* To learn more about Angular compiler optio
3  {
4  ↗ "extends": "./tsconfig.json",
5    "compilerOptions": {
6      "outDir": "./out-tsc/spec",
7      "types": [
8        "jasmine"
9      ]
10   },
11   "include": [
12     "src/**/*.ts"
13   ]
14 }
15

```

```

{} tsconfig.app.json X
basics > {} tsconfig.app.json > ...
1  /* To learn more about Typescript conf
2  /* To learn more about Angular compile
3  {
4  ↗ "extends": "./tsconfig.json",
5    "compilerOptions": {
6      "outDir": "./out-tsc/app",
7      "types": [
8        "node"
9      ]
10   },
11   "include": [
12     "src/**/*.ts"
13   ],
14   "exclude": [
15     "src/**/*.spec.ts"
16   ]
17 }
18

```

- tsconfig.spec.json – Runs for tests

```

{} tsconfig.app.json
ts tsconfig.json
{} tsconfig.spec.json

```