@NgModule

NgModule provides grouping the components, directives, pipes and services to build a functional application. There are advantages of using NgModule and it helps to organize a web application to smaller blocks of functionality. Also, it is much easier to extend the web application by including new modules. Encapsulation is another important feature of module and a web application component (like routing) can be loaded when user points to certain sections in the web browser.

```
43
44  @NgModule({
45    declarations: [
46      AppComponent, SortComponent
47    ],
48    exports: [
49      MatSortModule,
50      MatTableModule,
51      MatToolbarModule,
52      MatTooltipModule,
53      CdkTableModule,
54      AppComponent,
55      SortComponent
56    ],
57    imports: [
58      BrowserModule,
59      HttpClientModule,
60      BrowserAnimationsModule,
61      FormsModule,
62      HttpModule,
63      MatNativeDateModule,
64      ReactiveFormsModule,
65      MatSortModule, MatTableModule
66    ],
67    providers: [],
68    bootstrap: [AppComponent],
69    entryComponents: [SortComponent]
70  })
71  export class AppModule { }
72
```

Now let's look at the above codes and each property inside @NgModule decorator:

### Declarations:

This is the section to define all the components, directives and pipes that are defined and to be used inside the module. Angular throws an error at runtime when a directive or a component is not declared. If a component needs to be in multiple places (module) then it needs to be handled with a separate module and import that in the main module.

### Exports:

Here we define all the components, directives or pipes that an Angular app needs to export as public classes. This way the app module is providing them to other modules if they get imported. These classes stay private and visible to other component declared in the NgModule if not exported.

### Imports:

This is to share modules when modules are incorporate their directives, component and pipes. Importing modules/classes from @angular makes it possible to access directives to use in the application (NgIf, NgClass, EmailValidator and more).

### Providers:

The classes that are configured by providers are available for dependency injection (DI). Any sub-components or modules can then get the same instance of that @Injectable via dependency injection. Injector creates singleton object of a class configured by providers.

### Bootstrap:

Defines the root-component of the Application.

### Entry Component (entryComponent):

Angular can start with more than one bootstrap component and with its own location in the host web page. A bootstrap component is automatically an **entryComponent**. More than one component can be bootstrapped and only one component in the list can be the root component of the application.

For complete information: https://angular.io/guide/ngmodule-faq