

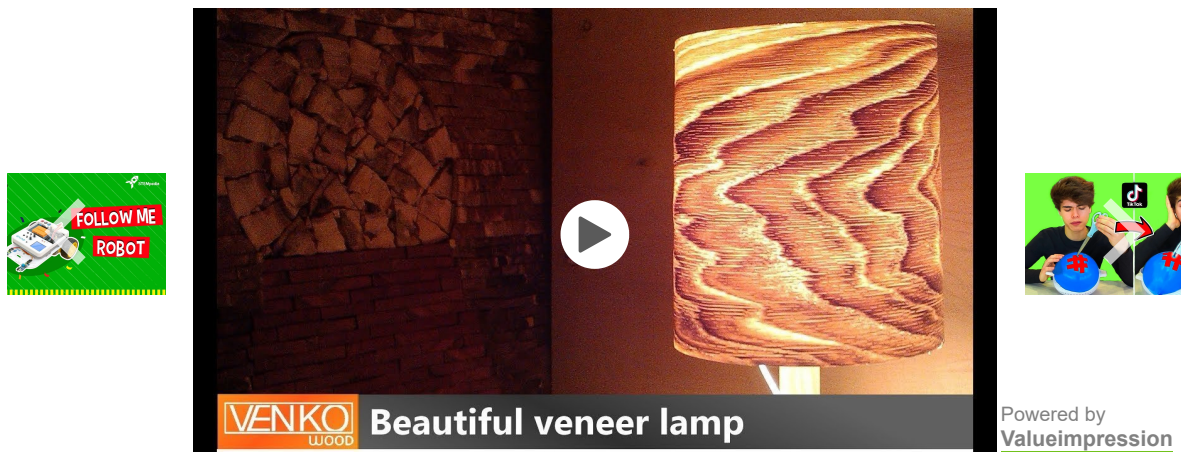
# Introduction to Angular Modules

1 Comment / 8 minutes of reading

← Router Events

Routing Between Modules →

The [building blocks of Angular Applications](#) consist of [Components](#), [Templates](#), [Directives](#), [Pipes](#), and [Services](#). We build a lot of such blocks to create a complete application. As the application grows bigger in size managing these blocks become difficult. The Angular Provides a nice way to organize these blocks in a simple and effectively using Angular modules (Also known as *ngModules*).



The Module is an overloaded term. It means different things depending on the context. There are two kind of modules exists in Angular itself. One is the **JavaScript Module** and the other one is **Angular Module**.

Applies to: Angular 2 to the latest edition of i.e. Angular 8. Angular 9, Angular 10, Angular 11, Angular 12

## Table of Contents

[What is Angular Module](#)

[JavaScript Modules](#)

[How to Create an Angular Module](#)

[ngModule](#)

- [Declarations array](#)
- [Providers array](#)
- [Imports array](#)
- [Exports array](#)
- [Bootstrap](#)
- [EntryComponents](#)
- [Angular Module Example](#)
  - [Create the application](#)
  - [Routing Module](#)
  - [Root Module](#)
  - [Creating a Module](#)
    - [Components](#)
    - [Home Module](#)
  - [Using HomeModule in AppModule](#)
- [Summary](#)

## What is Angular Module

The Angular module (also known as ngModule) help us to organize the application parts into cohesive blocks of functionality. Each block is focused on providing a specific functionality or a feature.

The Angular module must implement a specific feature. The Components, Directives, Services which implement such a feature, will become part of that Module.

The Modular design helps to keep the Separation of concerns. Keep the features together. Makes it easy to maintain the code. Makes it easier to reuse the code.

The Angular itself is built on the concept of Modules. The `@angular/core` is the main Angular module, which implements the Angular's core functionality, low-level services, and utilities.

The Features like Forms, HTTP and Routing are implemented as separate Feature modules such as [FormsModule](#), [HttpClientModule](#), and [RouterModule](#). There are Many third-party libraries built around Angular such as [Material Design](#), [Ionic](#), etc.

Angular Applications are assembled from the modules. The module exports [Component](#), [directive](#), [service](#), [pipe](#), etc, which can be then imported in another module

## JavaScript Modules vs. NgModules

## JavaScript Modules

The JavaScript Modules , which also goes by the name JS modules or ES modules or ECMAScript modules are part of the JavaScript Language. The JS Modules are stored in a file. There is exactly one module per file and one file per module. These modules contain small units of independent, reusable code. They export a value, which can be imported and used in some other module.

The following is a Javascript Module which exports the `SomeComponent` .

```
1  
2 export class SomeComponent {  
3   //do something  
4 }  
5
```

The `SomeOtherComponent` is another JavaScript Module, which imports and uses the `SomeComponent` .

```
1  
2 import { SomeComponent } from './some.component';  
3  
4 export class SomeOtherComponent {  
5   /do some other thing  
6 }  
7
```

## How to Create an Angular Module

The Angular Module must declare the [Components](#), [Pipes](#), and [Directives](#) it manages. We Create the Angular Modules by decorating it with the `NgModule` decorator.

### BEST ANGULAR BOOKS

The Top 8 [Best Angular Books](#), which helps you to get started with Angular

## ngModule

The `NgModule()` decorator is a function that takes a single metadata object, whose properties describe the module. The most important properties are as follows.

```
1
2 @NgModule({
3   declarations: [ ],
4   imports:     [ ],
5   providers:   [ ],
6   exports:     [ ],
7   bootstrap:  [ ],
8   entrycomponents: [ ]
9 })
10
```

### Declarations array

This is where [components](#), [directives](#), and [pipes](#) that belong to this `NgModule` are declared.

You should add only those, which belong to this module. The Component cannot belong to more than one module.

The services must not be declared here. They are defined in the [Angular Providers](#) array.

### Providers array

The [Services](#), which you want to add to the global collection of services are added here. The services are then available for injection via [dependency injection](#).

Remember :  
Declarations is for Components,  
Directives & Pipes.  
Providers are for Services

Remember:

Services are global in scope. Services added to Providers array are available for injection in the entire application

### **Suggested Readings**

[Angular Services](#)

[Dependency Injection in Angular](#)

[Angular Injector](#)

[Angular Providers](#)

[Angular Hierarchical Dependency Injection](#)

### **Imports array**

If you want this NgModule require any feature or functionality, then those modules need to be imported here. Any [components](#), [directives](#), and [pipes](#) that are defined and exported in that module can be used in this module.

### **Exports array**

If you want other modules to use the [component](#), [pipes](#), [directives](#) of this NgModule, then those must be specified here.

**Remember:**

Only those components declared here are visible to the other `ngModules`, when they import this module.

**Bootstrap**

The main component of this module, which needs to be loaded when the module is loaded is specified here.

This is a must if you are the first module ( **called the root module**) that is loaded when the Angular App starts. It is the responsibility of the root module to load the first view and it is done by specifying the component here.

If the module is not the root module, then you should keep this blank

**Suggested Reading**

[How Angular Bootstraps your Application](#)

**EntryComponents**

The components that are dynamically loaded need to be declared here.

The components are loaded when angular

- Finds the Component Selector in the HTML
- Declared in the bootstrap array
- Declared in the root definition

If your component is not listed any of the above, then it needs to be declared in `EntryComponent` so that Angular knows where to find them and compile.

The following image should make it clear, how the `ngModule` metadata works

## Angular Module Example

Let us create an example application with multiple modules to demonstrate the use of the Angular Modules.

### Create the application

```
1  
2 ng new --routing --style css ModuleDemo  
3
```

Use `npm start` to run the program and test everything is ok.

```
1  
2 Cd ModuleDemo  
3 npm start  
4
```

The App Contains two Modules. i.e. `AppModule` & `AppRoutingModule`

### Routing Module

The `AppRoutingModule` is an Angular Module, which specifically defines the application Routes

```
1  
2 import { NgModule } from '@angular/core';  
3 import { Routes, RouterModule } from '@angular/router';
```

```
4
5 const routes: Routes = [];
6
7 @NgModule({
8   imports: [RouterModule.forRoot(routes)],
9   exports: [RouterModule]
10 })
11 export class AppRoutingModule { }
12
```

This module does not contain any components, pipes or directives. Hence it does not need to declare anything in the declaration array

No services are part of this Module, Hence providers array is also not needed.

The Angular RouterModule is needed for the routing to work, Hence it is imported. The Router module requires the routes to work, hence it is passed to the RouterModule in the forRoot method.

The application needs only one instance of the services provided by the RouterModule. But every Angular module might contain routes of its own and needs to import RouterModule and register the routes.

Angular has a nice way to solve this problem. The Services are registered only if you call the forRoot method. forRoot method must be called only once and in the Root module. In the other modules, we will be using the forChild method to register the routes

## Root Module

The first module that gets loaded in called **Root Module**, which is conventionally named as AppModule and is automatically created by the [Angular CLI](#).

The following is the auto-generated code of the AppModule. The class is decorated with the @NgModule decorator

```
1
2 import { BrowserModule } from '@angular/platform-browser';
3 import { NgModule } from '@angular/core';
4
5 import { AppRoutingModule } from './app-routing.module';
6 import { AppComponent } from './app.component';
```



```
7
8 @NgModule({
9   declarations: [
10     AppComponent
11   ],
12   imports: [
13     BrowserModule,
14     AppRoutingModule
15   ],
16   providers: [],
17   bootstrap: [AppComponent]
18 })
19 export class AppModule { }
20
```

The AppModule has only one component i.e AppComponent and it does not have any pipes or directives. Hence the declaration array contains only AppComponent

The imports metadata should contain the other modules that is being used by this module. Hence it imports the BrowserModule, which is an Angular Library. It also requires Routes, hence we are importing AppRoutingModule

The AppModule does not define any services. Hence the providers array is empty

Finally, we need to load the AppComponent when the App starts, hence we define it in bootstrap array

## Creating a Module

Let us add another Module to our above application.

Create a folder home under src/app folder

## Components

Let us add three components. HomeComponent, AboutUsComponent, ContactUsComponent.

Create a folder called pages under the home folder. Create a three folder under pages aboutus, contactus & home.

### home/pages/aboutus/about-us.component.ts

```
1
2 import { Component } from '@angular/core';
3
```

```
4 @Component({  
5   templateUrl: './about-us.component.html',  
6 })  
7 export class AboutUsComponent  
8 {  
9 }  
10
```

### home/pages/aboutus/about-us.component.html

```
1  
2 <h1> About Us Page</h1>  
3
```

### home/pages/contactus/contact-us.component.ts

```
1  
2 import { Component } from '@angular/core';  
3  
4 @Component({  
5   templateUrl: './contact-us.component.html',  
6 })  
7 export class ContactUsComponent  
8 {  
9 }  
10
```

### home/pages/contactus/contact-us.component.html

```
1  
2 <h1> Contact Us</h1>  
3
```

### home/pages/home/home.component.ts

```
1  
2 import { Component } from '@angular/core';  
3  
4 @Component({  
5   templateUrl: './home.component.html',  
6 })  
7 export class HomeComponent  
8 {  
9 }  
10
```

### home/pages/home/home.component.html

```
1  
2 <h1> Welcome To Module Demo</h1>  
3
```

## home/pages/index.ts

```
1  
2 export * from './aboutus/about-us.component';  
3 export * from './contactus/contact-us.component';  
4 export * from './home/home.component';  
5
```

## Home Module

Now, our components are ready, we can now create the HomeModule

Create the home.module.ts under the home folder

```
1  
2 import { NgModule } from '@angular/core';  
3 import { RouterModule, Routes } from '@angular/router';  
4 import { CommonModule } from '@angular/common';  
5  
6 import { AboutUsComponent, ContactUsComponent, HomeComponent } from './pages';  
7  
8 const routes: Routes = [  
9   { path: '', component: HomeComponent },  
10  { path: 'home', component: HomeComponent },  
11  { path: 'contactus', component: ContactUsComponent },  
12  { path: 'aboutus', component: AboutUsComponent },  
13 ];  
14  
15 @NgModule({  
16   declarations: [AboutUsComponent, ContactUsComponent, HomeComponent],  
17   imports: [  
18     CommonModule,  
19     RouterModule.forChild(routes),  
20   ],  
21   providers: [],  
22 })  
23 export class HomeModule { }  
24
```

We decorate the HomeModule class with @NgModule to let Angular know that it is a Angular Module

Next, use the declarations array to declare the three components that we have created.

The `HomeModule` do not expose any services, hence we keep the `providers` array empty

We are keeping the `Routes` in the module itself. You can create the separate routing module similar to the `AppRoutingModule`. That will help to keep the `HomeModule` code clean.

`HomeModule` requires `CommonModule` hence it is added to the `imports` array. The `RouterModule` module is imported and routes are registered with the call to `forChild(routes)`. `forChild` registered the routes but does not registers any services.

### home/index.ts

```
1
2 export * from './pages';
3 export * from './home.module';
4
```

### Using HomeModule in AppModule

Next step is to import the `HomeModule` in the `AppModule`. First import the `HomeModule`

```
1
2 import { HomeModule } from './home';
3
```

Next, add `HomeModule` to the imports metadata of the `@NgModule`

```
1
2 imports: [
3   BrowserModule,
4   AppRoutingModule,
5   HomeModule
6 ],
7
```

Next, we need to use the Components of the `HomeModule`. Open the `app.component.ts` and add the following code

```
1
2 <ul>
3   <li>
4     <a class="navbar-brand" routerLink="/">Home</a>
5   </li>
6   <li>
7     <a class="navbar-brand" routerLink="/aboutus">About</a>
```

```
8 </li>
9 <li>
10   <a class="navbar-brand" routerLink="/contactus">Contact</a>
11 </li>
12 </ul>
13
14 <router-outlet></router-outlet>
15
```

All we have done is to use them `routerLink` to create a menu item.

### Suggested Readings:

[Angular Router](#)

Copy the following CSS to `app.component.css`

```
1
2 ul {
3   list-style-type: none;
4   margin: 0;
5   padding: 0;
6   overflow: hidden;
7   background-color: #333333;
8 }
9
10 li {
11   float: left;
12 }
13
14 li a {
15   display: block;
16   color: white;
17   text-align: center;
18   padding: 16px;
19   text-decoration: none;
20 }
21
22 li a:hover {
23   background-color: #111111;
24 }
25
```

Finally, run the app

## Summary

In this tutorial, we learned what is Angular Modules are and how to create an Angular Module.

[← Router Events](#)[Routing Between Modules →](#)

1 thought on "Introduction to Angular Modules"

**GANESH RAMKRISHNA BORDE**

JULY 22, 2021 AT 8:34 PM

Very simple language with more explanation

[Reply](#)

## Leave a Comment

Your email address will not be published.

Type here..

Name\*

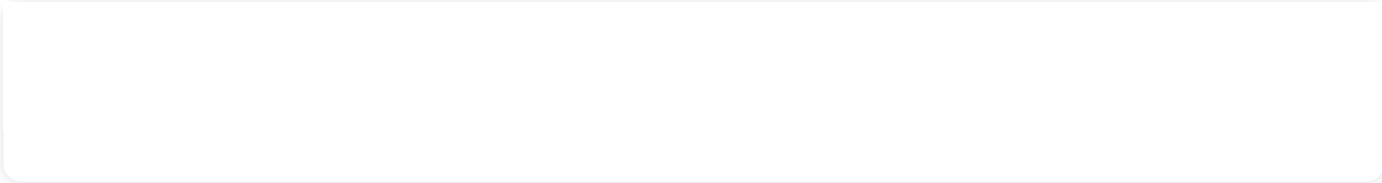
Email\*

Website

Post Comment »

This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)





---

Copyright © 2021 TekTutorialsHub |