# Angular Bootstrapping

Every application has at least one Angular module, the *root* module that you bootstrap to launch the application. By convention, it is usually called AppModule.

**import** { BrowserModule } **from '@angular/platform-browser'**;  
**import** { NgModule } **from '@angular/core'**;  
  
**import** { AppComponent } **from './app.component'**;  
**import** { App2Component } **from './app2/app2.component'**;  
  
@NgModule({  
 **declarations**: [  
 AppComponent,  
 App2Component  
 ],  
 **imports**: [  
 BrowserModule  
 ],  
 **providers**: [],  
 **bootstrap**: [AppComponent, App2Component]  
})  
**export class** AppModule { }

In the file index.html

<**body**>  
 <**app-root**></**app-root**>  
 <**p**>second paragrah:</**p**>  
 <**app-root2**></**app-root2**>  
</**body**>

The @NgModule decorator identifies AppModule as an NgModule class. @NgModule takes a metadata object that tells Angular how to compile and launch the application.

* declarations—this application's lone component.
* imports—import BrowserModule to have browser specific services such as DOM rendering, sanitization, and location.
* providers—the service providers.
* bootstrap—the root component that Angular creates and inserts into the index.html host web page.

## The declarations array

The module’s declarations array tells Angular which components, directives and pipes belongs to that module. You must declare every component in exactly one NgModule class. If you use a component without declaring it, Angular returns an error message.

These declared classes are visible within the module but invisible to components in a different module unless they are exported from this module and the other module imports this one(module).

## The imports array

It tells Angular about other NgModules that this module needs. This list of modules that export components, directives, or pipes that the component templates in this module can reference.

In this case, the component is AppComponent and App2Component, which references components, directives, or pipes in BrowserModule.

## The Providers array

The providers array is where you list the services the app needs. When you list services here, they are available **app-wide**. You can scope them when using feature modules and lazy loading.

## The bootstrap array

The application launches by bootstrapping the root AppModule (in file main.ts).

platformBrowserDynamic().bootstrapModule(AppModule)  
 .catch(err => ***console***.log(err));

Among other things, the bootstrapping process creates the component(s) listed in the bootstrap array and inserts each one into the browser DOM.

Each bootstrapped component is the base of its own tree of components. Inserting a bootstrapped component usually triggers a cascade of component creations that fill out that tree.

While you can put more than one component tree on a host web page, most applications have only one component tree and bootstrap a single root component.

# NgModules

Angular libraries are NgModules, such as *FormsModule*, *HttpClientModule*, and *RouterModule*. Many third-party libraries are available as NgModules such as *Material Design*, *Ionic*, and *AngularFire2*.

Modules can be loaded eagerly when the application starts or lazy loaded asynchronously by the router.

NgModule metadata does the following:

* declarations: Declares which components, directives, and pipes belong to the module.
* exports: Makes some of those components, directives, and pipes public so that other module's component templates can use them.
* imports: Imports other modules with the components, directives, and pipes that components in the current module need.
* providers: Provides services that the other application components can use.

Every Angular app has at least one module, the root module. You bootstrap that module to launch the application.

## Frequently Used Modules

The following are frequently used Angular modules with examples of some of the things they contain:

|  |  |  |
| --- | --- | --- |
| NgModule | Import it from | Why you use it |
| BrowserModule | @angular/platform-browser | When you want to run your app in a browser |
| CommonModule | @angular/common | When you want to use **NgIf**, **NgFor** |

BrowserModule imports CommonModule, which contributes many common directives such as ngIf and ngFor. Additionally, BrowserModule re-exports CommonModule making all of its directives available to any module that imports BrowserModule.

For apps that run in the browser, import BrowserModule in the root AppModule because it provides services that are essential to launch and run a browser app. BrowserModule’s providers are for the whole app so it should only be in the root module, not in **feature modules**. **Feature modules** only need the common directives in CommonModule; they don’t need to re-install app-wide providers.

If you do import BrowserModule into a lazy loaded feature module, Angular returns an error telling you to use CommonModule instead.

## Feature Modules