

## Frequently-used modules

An Angular application needs at least one module that serves as the root module. As you add features to your app, you can add them in 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 application in a browser
CommonModule	@angular/common	When you want to use <code>NgIf</code> , <code>NgFor</code>
FormsModule	@angular/forms	When you want to build template driven forms (includes <code>NgModel</code> )
ReactiveFormsModule	@angular/forms	When you want to build reactive forms
RouterModule	@angular/router	When you want to use <code>RouterLink</code> , <code>.forRoot()</code> , and <code>.forChild()</code>
HttpClientModule	@angular/common/http	When you want to communicate with a server using the HTTP protocol

## Importing modules

When you use these Angular modules, import them in `AppModule`, or your feature module as appropriate, and list them in the `@NgModule imports` array. For example, in the basic application generated by the [Angular CLI](#), `BrowserModule` is the first import at the top of the `AppModule`, `app.module.ts`.

```
/* import modules so that AppModule can access them */
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [ /* add modules here so Angular knows to use them */
    BrowserModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

The imports at the top of the array are JavaScript import statements while the `imports` array within `@NgModule` is Angular specific. For more information on the difference, see [JavaScript Modules vs. NgModules](#).

## BrowserModule and CommonModule

`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 applications that run in the browser, import `BrowserModule` in the root `AppModule` because it provides services that are essential to launch and run a browser application. `BrowserModule`'s providers are for the whole application 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.



BrowserModule error

## More on NgModules

You may also be interested in the following:

- [Bootstrapping](#).
- [NgModules](#).
- [JavaScript Modules vs. NgModules](#).