Types of feature modules

There are five general categories of feature modules which tend to fall into the following groups:

- Domain feature modules.
- Routed feature modules.
- Routing modules.
- Service feature modules.
- Widget feature modules.

While the following guidelines describe the use of each type and their typical characteristics, in real world apps, you may see hybrids.

Feature Module	Guidelines
Domain	Domain feature modules deliver a user experience dedicated to a particular application domain like editing a customer or placing an order. They typically have a top component that acts as the feature root and private, supporting sub-components descend from it. Domain feature modules consist mostly of declarations. Only the top component is exported. Domain feature modules rarely have providers. When they do, the lifetime of the provided services should be the same as the lifetime of the module. Domain feature modules are typically imported exactly once by a larger feature module.

	They might be imported by the root `AppModule` of a small application that lacks routing.
Routed	Routed feature modules are domain feature modules whose top components are the targets of router navigation routes. All lazy-loaded modules are routed feature modules by definition. Routed feature modules don't export anything because their components never appear in the template of an external component. A lazy-loaded routed feature module should not be imported by any module. Doing so would trigger an eager load, defeating the purpose of lazy loading.That means you won't see them mentioned among the `AppModule`

	imports. An eager loaded routed feature module must be imported by another module so that the compiler learns about its components. Routed feature modules rarely have providers for reasons explained in Lazy Loading Feature Modules. When they do, the lifetime of the provided services should be the same as the lifetime of the module. Don't provide application-wide singleton services in a routed feature module or in a module that the routed module imports.
Routing	A routing module provides routing configuration for another module and separates routing concerns from its companion module. A routing module typically does the following:

- Defines routes.
- Adds router configuration to the module's imports.
- Adds guard and resolver service providers to the module's providers.
- The name of the routing module should parallel the name of its companion module, using the suffix "Routing". For example, `FooModule` in `foo.module.ts` has a routing module named `FooRoutingModule` in `foo-routing.module.ts`. If the companion module is the root `AppModule`, the `AppRoutingModule` adds router configuration to its imports with `RouterModule.forRoot(routes)`. All other routing modules are children that import `RouterModule.forChild(routes)`.
- A routing module re-exports the `RouterModule` as a convenience so that components of the companion

	module have access to router directives such as `RouterLink` and `RouterOutlet`. • A routing module does not have its own declarations. Components, directives, and pipes are the responsibility of the feature module, not the routing module. A routing module should only be imported by its companion module.
Service	Service modules provide utility services such as data access and messaging. Ideally, they consist entirely of providers and have no declarations. Angular's `HttpClientModule` is a good example of a service module. The root `AppModule` is the only module that should import service modules.

Widget	A widget module makes components, directives, and pipes available to external modules. Many third-party UI component libraries are widget modules. A widget module should consist entirely of declarations, most of them exported. A widget module should rarely have providers. Import widget modules in any module whose component templates need the widgets.

The following table summarizes the key characteristics of each feature module group.

Feature Module	Declarations	Providers	Exports
Domain	Yes	Rare	Top compon
Routed	Yes	Rare	No
Routing	No	Yes (Guards)	RouterM
Service	No	Yes	No
Widget	Yes	Rare	Yes

More on NgModules

You may also be interested in the following:

- Lazy Loading Modules with the Angular Router.
- Providers.