Angular Modules and Optimizing Angular Apps

What are Modules?

- **Modules** bundle Angular **building blocks** together so that Angular is **aware** of the features
- Building-Blocks (within **AppModule**)
  - AppComponent
  - ProductsComponent
  - HighlightDirective
  - ProductsService
- Traits
  - Angular **analyzes NgModules** to *understand* your application and its features
  - Angular **modules define** all **building-blocks** your app uses
    - Components, Directives, Services
  - An app **requires** at least **one module** (AppModule), but may be split into multiple modules
  - Core Angular features are included in Angular modules (e.g. FormsModule) to **load them only when needed**
  - You **can't use a feature/building-block** without including it in a module

Analyzing the AppModule

- **Arrays**
  - **declarations** → Contains all **components**, **directives**, and **pipes** used in the application
  - **imports** → Allows you to **import other modules** into **AppModule**
  - **providers** → Contains the **services** you wish to share **app-wide**
  - **bootstrap** → Defines which **component** is available in **index.html**
  - **entryComponents** → Components that Angular loads **imperatively**

- **AppRoutingModule** → Holds our route configuration
- **All modules work independently** in Angular
  - They don't communicate directly with each other
  - You must **export** modules for communication

Getting Started with Feature Modules

- **Feature Modules**
  - Module that **groups together components**, **directives**, **pipes**, and **services** that are used in a **certain feature area** of the application
- Creating a module
  - Create a **<module-name>.module.ts** file
  - **Export** a **class** of the module's name
  - Add **@NgModule decorator**
    - In it, add **declarations** array containing all relevant components
    - Create **export** array, and add all components there too
  - In **AppModule**, add the new module to the **imports** array

Splitting Modules Correctly

- Following only what we presented before will culminate in **<router-outlet>** errors
- To remedy this, we must also add an **imports** array to the new module with **RouterModule**
- We must do the same for any other important Angular package's within **AppModule**'s **import** array
  - Import **CommonModule** instead of **BrowserModule** within the new module as the latter can only be imported **once**
  - The only **exception** to this rule is with **services** → These are declared in **AppModule**

Adding Routes to Feature Modules

- Create a **<module-name>-routing.module.ts** file
  - Create a **class** with the module's name
  - With the **@NgModule decorator**, add an **imports** array with **RouterModule**

    - Create a variable containing all relevant routes from **AppRoutingModule**
        - This is an array containing the one routing object
- Within the new module's **imports** array, invoke the **forChild(<routes-variable>)** method on **RouterModule**
- Within the new module's **exports** array, add **RouterModule**
- In the **feature module**, **import <module-name>-routing.module.ts**

Component Declarations

- You not only add components that you intend on using to the **declarations** array, but also any components that are touched via **routing**
- It's **not enough** to add touched routes to the routing file

Understanding Shared Modules

- **Shared Module**
    - Contains components, directives, modules, etc that are **used by numerous other modules**
    - Access via the **imports** array
    - Helps **eliminate code redundancy**
- Like before, we still **declare everything** that is **used within this module**
- To make it available elsewhere, we also **export** it
- You may only **declare** components, directives, and pipes **once**
    - To remedy this, **export/import** the component, directive, or pipe elsewhere

Understanding the Core Module

- **CoreModule** makes **AppModule** leaner by containing services that would otherwise pollute **AppModule**
    - For this, **CoreModule** would then be imported to **AppModule**

Understanding Lazy Loading

- **Lazy Loading** allows us to specify exactly what content we wish to load instead of loading all content at once
- This is why it's good to make modules → We can easily load entire modules, ignoring all others

Implementing Lazy Loading

- For lazy-loading to work, your **feature module** must have its own **route config** with **forChild**
- The **base route** within the **feature module** should be an **empty string**
- In **AppRoutingModule**, add a new route
  - This contains the value from the **feature module's original base route**
  - Instead of a component, uses **loadChildren**
  - Example → **{ path: 'recipes', loadChildren: () => import('./recipes/recipes.module').then(m => m.RecipesModule) }**
- Must then **remove** the **lazily-loaded module** from the **imports** array of **AppModule**
- It's good to only **lazy-load pages** that **users may never see**, not ones they'll always encounter

Pre-Loading Lazy-Loaded Code

- We can avoid delays by **pre-loading lazily-loaded modules**
- In **AppRoutingModule**, within **RouterModule.forRoot(. . .)**, add argument **{ preloadingStrategy: PreloadAllModules }**
- Downloads other files while the user is idle doing something else
- Best of **both worlds** → Fast initialization and fast subsequent downloads

Modules and Services

- Places we can **provide services**
  - **AppModule**
    - The same instance is available **application-wide**
    - Uses **root-injector**
    - Should be the **default**

- **AppComponent** (or other components)
  - The service is available in the **component-tree**
  - Uses **component-specific injector**
  - Use if the service is only relevant for the **component tree**
- **Eager-Loaded** Module
  - The same instance is available **application-wide**
  - Uses **root injector**
  - **Avoid** this
- **Lazy-Loaded** Modules
  - The service is available in the **loaded module**
  - Uses **child injector**
  - Only use if the service should be **scoped to the loaded module**