**MODULES**

**Types**

**Domain:** Wraps up all the components needed to implement one single feature

**Routed:** Same as a Domain module, but these components are tied to routes (only displayed when the user is at some particular address)

**Routing:** Defines routing rules (eg. Show a specific component when a user is at a specific route)

**Service:** Defines services that will be used in multiple parts of the app

**Widget:** Defines some reusable components that will be used in multiple other modules

**NEW MODULE:** ng g(enerate) n(ew) <module\_name> --routing *(specifies that the component uses a specific route)*

To generate a new component or other inside a module, **ng generate <type> <module>/<element-name>.**

**EXPORTING COMPONENTS FROM MODULES**

In order for components to be used outside a module, they need to be exported first and then imported to the other module that will use them.

1. Inside parent element, inside @NgModule settings, add:

exports: [

    ComponentToBeExported

  ]

1. Inside importing module
   1. import { ImportedModule } from './modules/moduleName/moduleName.module';
   2. add module to the imports array

**MODULE PROPERTIES**

**Declarations:** List of components, pipes, directives that are created in this module. Updated automatically when something new is generated

**Imports:** List of other modules that this module depends upon

**Exports:** List of components, pipes, directives that this module makes available to other modules

**Providers: Old** way of connecting modules and services

**Bootstrap:** Used by the AppModule only to declare what component will be displayed first

**Module Routing**

Inside the *elements-routing.module.ts* add to the routes constant the url path and the component that needs to be loaded. Eg:

const routes: Routes = [

  { path: 'elements', component: ElementsHomeComponent }

];

The app module uses all the routing rules included in the module. So, when we navigate to this url, we see what the module specifies

The <router-outlet> of the app.component.html ‘reads’ any URL and searches if there is any module/component with that name and then renders it inside the outlet.

When creating anchor elements, we need to use the ***routerLink*** attribute, because otherwise the page refreshes and reloads all resources. The **route*rLink***navigates inside the router and decides which module to show inside the *outlet.*

The *routerLinkActive* specifies which class will be applied when the link is active.

**Routing Hierarchy**

In the *app.module.ts*, rearrange the imports as needed. The *AppRoutingModule* should go last to allow any other routing first.

**Using the router inside components/modules**

1. Inject the constructor with a parameter of type *Router.*
2. Use the property wherever needed.

**LAZY LOADING**

1. Select which modules should be lazy loaded
2. Remove any imports of these elements anywhere
3. In the *AppRoutingModule*, define a Route in the *‘routes’* array to specify when to load that module.

{ path: 'anyElement',

    loadChildren: () => import('./modules/elements/anyElement.module').then((*m*)=>*m*.anyElementModule)

  },

1. Modify the routing in the modules themselves, because the routing rules are cumulative. When navigating to the element root, the path should be empty

  { path: '', component: ElementsHomeComponent }