# Lazy loading

Lazy loading is design pattern which loads NgModules as needed.

- By default, NgModules are eagerly loaded, which means that as soon as the app loads, so do all the NgModules, whether or not they are immediately necessary.

- Lazy loading helps keep initial bundle sizes smaller, which in turn helps decrease load times.

There are two main steps to setting up a lazy-loaded feature module:

1. **Create the feature module with the CLI, using the --route flag.**
2. **Configure the routes.**

## Create the feature module with the CLI, using the --route flag

Let us create a feature module with a component to route to. To create it, enter the following command in terminal.

Syntax:

**ng generate module module-name –routing**

Ex:

ng generate module TestLazy –routing

By the above command, a new folder “TestLazy” is created and two files in it one is “**testlazy.module.ts**” and “ **testlazy-routing.module.ts**”.

Now create a feature component to this module.

Ex:

ng generate component testlazy/testlazy

This command declares “test lazy component” in “testlazy module”.

Because the new module is meant to be lazy-loaded, the command does not add a reference to the new feature module in the application’s root module. Now, let us add declare route “testlazy” to the “router” array of root module.

**app-routing.module.ts:**

```
const routes: Routes = [
  {
   path: 'customers',
  loadChildren: () => import('./testlazy/testlazy.module').then(m =>
                                                    m.TestlazyModule)
 } ];
```

Notice that the lazy-loading syntax uses loadChildren followed by a function that uses the browser's built-in import('...') syntax for dynamic imports.

Now let us set UI with router links.

```
<a routerLink=”testlazy”>TestLazy</a>

<router-outlet></router-outlet>
```

## Configuring routes

Now ,let us visit **testlazy.module.ts**. In this, “**TestlazyModule**” is registered and “**TestlazyComponent**” class were already registered to this module.

The “**app-routing.module.ts**” ,then imports the feature module(**testlazy.module.ts**) using JavaScript's dynamic import.

The routing module(**testlazy-routing.module.ts**) of feature module imports its own feature component defined in the **testlazy.component.ts** file, along with the other JavaScript import statements. It then maps the empty path to the **TestlazyComponent**.

**Testlazy-routing.module.ts:**

```
const routes: Routes = [
                        { path: ' ', component: TestlazyComponent }
                    ];
```

The path here is set to an empty string because the path in **AppRoutingModule** is already set to “ testlazy”. Every route in this routing module is a child route.

## Testing the lazy module

Launch the application using http://localhost:4200. Now you get entire app except the lazy loading module. When even we visit that route, then it will be received from server. Open developer tools and network tab in it . There we can observe this