Lazy Loading in Angular

@coder_aishya

Module Loading types

- **Eager loading (default strategy)**
- Lazy Loading
- Preloading

What is Lazy Loading?

- Lazy loading is the technique where angular **loads the Modules only when needed** rather than all loading all modules at once. It is also called on-demand loading.
- By default, **Angular Loads the modules eagerly.**
- Lazy loading helps keep **initial bundle sizes smaller**, which in turn **helps decrease load times.**

@coder_aishya

Note:

The **Lazy loading works at the module level.** i.e. you can lazy load only the Angular Modules. We cannot lazy load the Individual components.

Syntax

Angular 8 & higher Versions:

AppRoutingModule

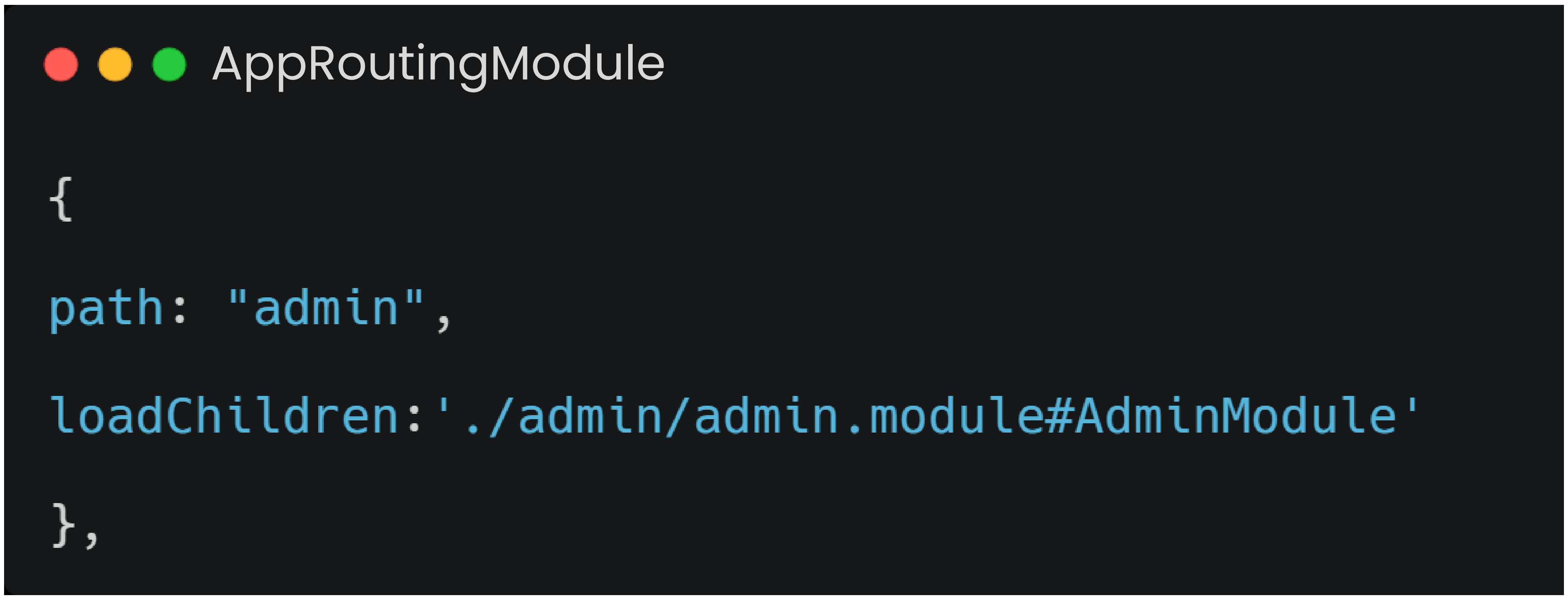
```
{
  path: "admin",
  loadChildren: () => import('./admin/admin.module').
                      then(m => m.AdminModule)
}
```

@coder_aishya

loadChildren

- The loadChildren is where we configure the Lazy Loading.
- We need to **provide call back function to loadChildren argument**. The call back must load the AdminModule.
- We **use the dynamic import syntax using the import method.**
- The import method loads the module from the path, which we provide as the argument to it.
- The lazy loaded module **loads only for the first visit of the URL**, it will not load when we revisit that URL again.

Angular 7 & lower versions Syntax:


AppRoutingModule

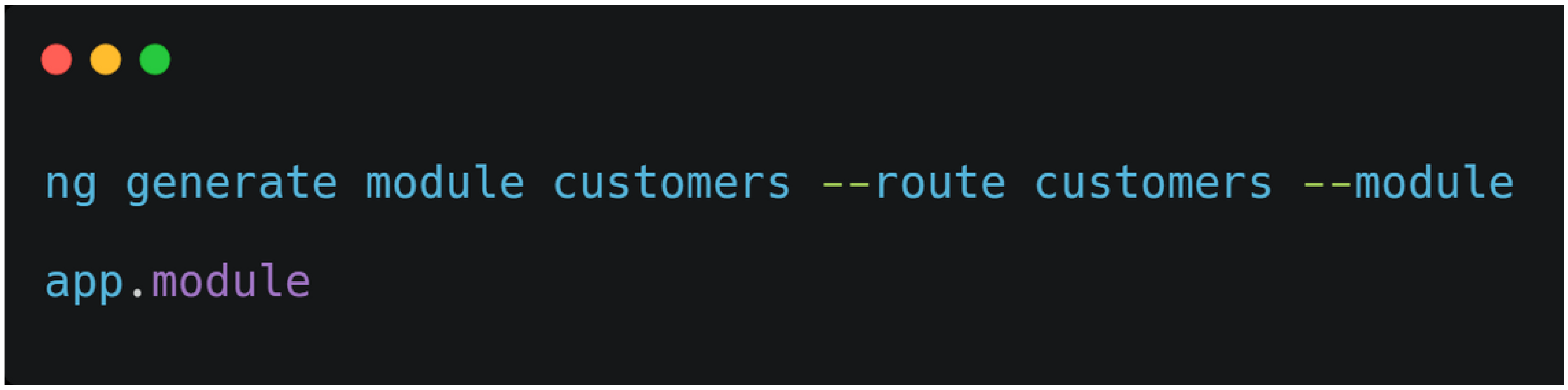
```
{
path: "admin",
loadChildren:'./admin/admin.module#AdminModule'
},
```

- The loadChildren **accepts the value as string**. The string is split into two sections separated by #.
- The **first part is the full path to the module file** (without the ts extension). In the example above ./admin/admin.module points to admin.module.ts file. The **second part is the export class name of the Module**. i.e AdminModule

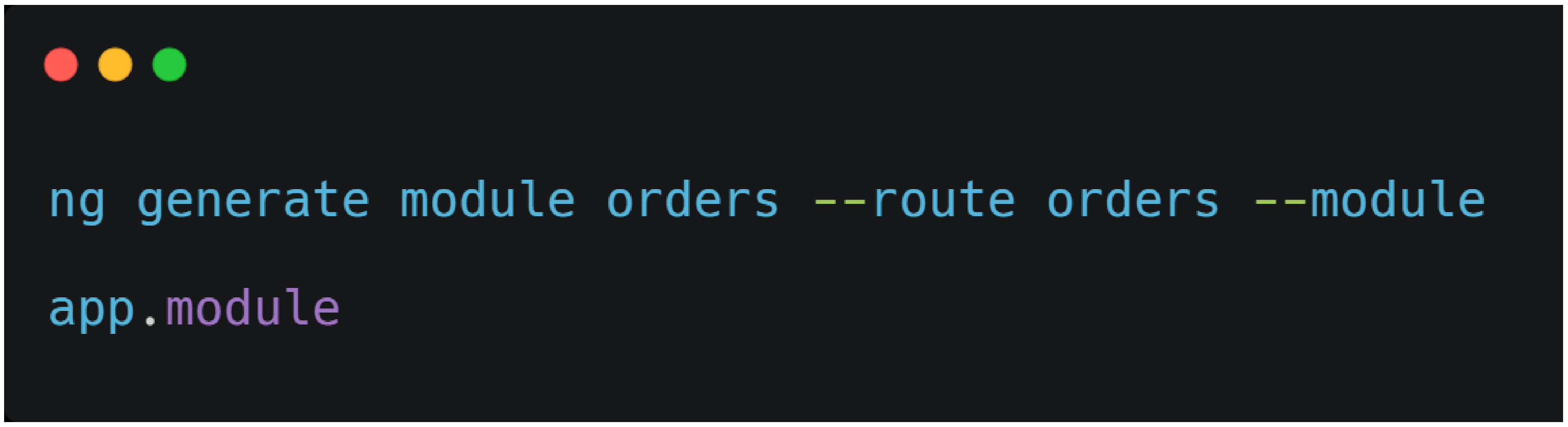
@coder_aishya

Setting Up Lazy Loading

Create two angular modules using command



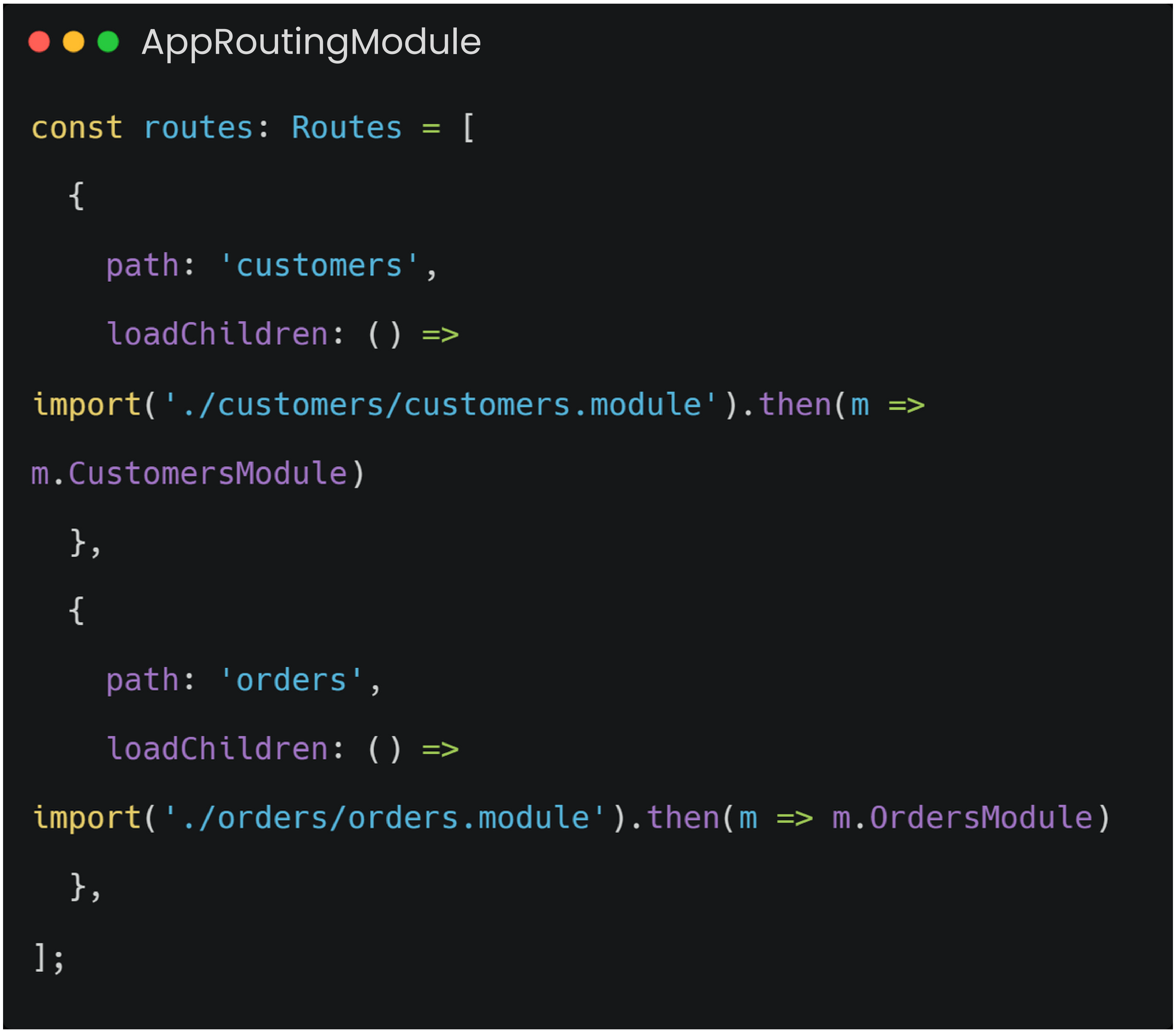
```
ng generate module customers --route customers --module
app.module
```



```
ng generate module orders --route orders --module
app.module
```

@coder_aishya

Define routes in AppModule using Lazyloading syntax

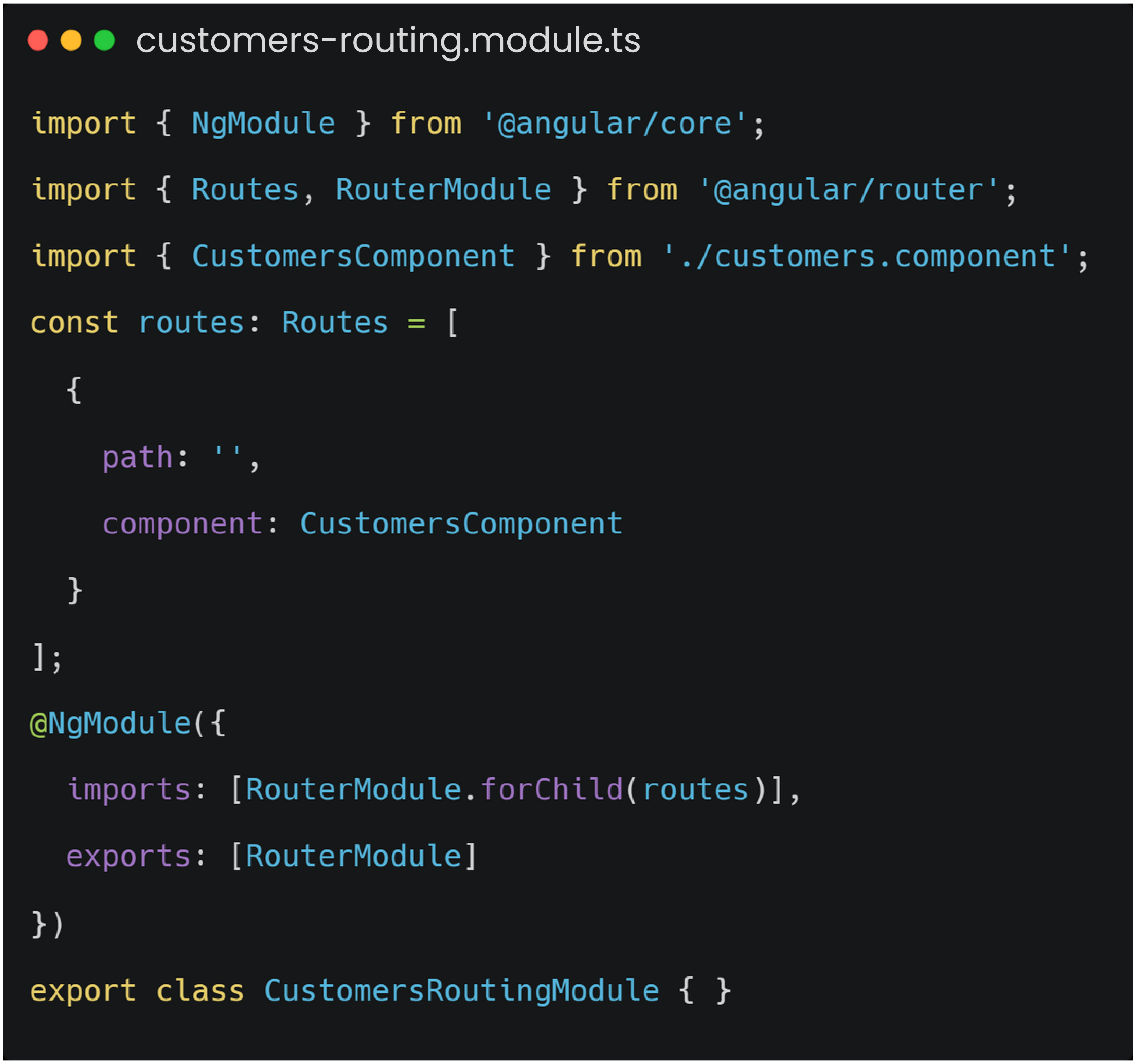


AppRoutingModule

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

@coder_aishya

Inside the feature module routing file, link the routes to be lazy loaded



customers-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { CustomersComponent } from './customers.component';
const routes: Routes = [
  {
    path: '',
    component: CustomersComponent
  }
];
@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class CustomersRoutingModule { }
```

@coder_aishya

forRoot()

- Use forRoot() **only once in the application**, inside the AppRoutingModule.
- This lets Angular know that the AppRoutingModule is a routing module and forRoot() specifies that this is the root(starting point of app) routing module.

forChild()

- Use RouterModule.forChild(routes) to **feature routing modules.**
- This way, Angular knows that the route list is only responsible for providing additional routes and is intended for feature modules. You **can use forChild() in multiple modules**.

Follow me for more

aishwarya-dhuri

coder_aishya