

# Angular Modules

---



**John Papa**

PRINCIPAL ARCHITECT

@john\_papa [www.johnpapa.net](http://www.johnpapa.net)



# Overview



## **Routed Modules**

- Eager Loading
- Lazy Loading

## **Preload Strategies**

## **Feature Modules**

## **Providers**



# Angular Modules

Help organize our applications into cohesive blocks of functionality



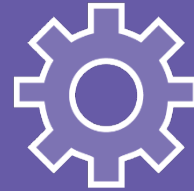
# Basic Types of Feature Modules



Domain



Routed

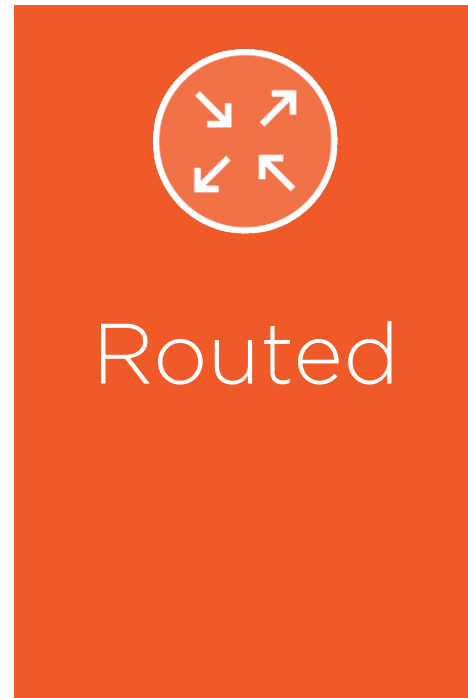


Service

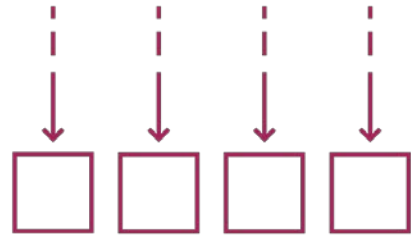


Widget

# Routed Modules Are Eagerly or Lazily Loaded



# Eager Loading Modules



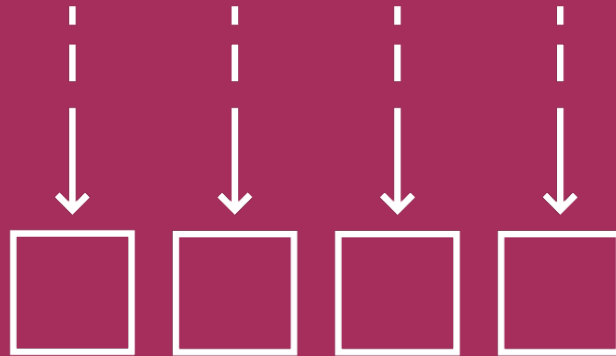
# Eager Loading

Loading a feature module at startup, with the root module.  
Begin by creating a **feature module** with routing.



# Eagerly Loading the Modules

Modules are loaded immediately





## characters.module.ts

```
import {
  CharactersRouterModule,
  routedComponents
} from './characters-routing.module';

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    CharactersRouterModule
  ],
  declarations: [routedComponents],
  providers: [CharacterService],
})
export class CharactersModule { }
```

Import routing module

## characters- routing.module.ts

```
const routes: Routes = [
  {
    path: 'characters',
    component: CharactersComponent,
    children: [
      { path: '', component: CharacterListComponent },
      { path: ':id', component: CharacterComponent },
    ]
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class CharactersRouterModule { }
```

Configure  
routes

forChild  
for  
feature  
module  
routes

Create routing module, and import into feature module

## app.module.ts

```
@NgModule({  
  imports: [  
    BrowserModule, FormsModule, HttpClientModule,  
    CharactersModule, AppRoutingModule  
  ],  
  declarations: [AppComponent],  
  bootstrap: [AppComponent]  
})  
export class AppModule { }
```

Import Characters  
module and its routes

## Importing Modules

Import the CharactersModule

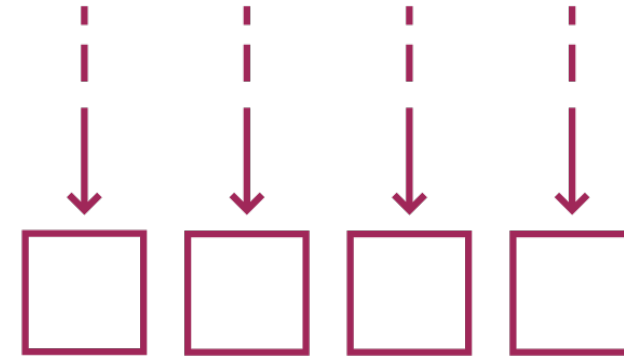
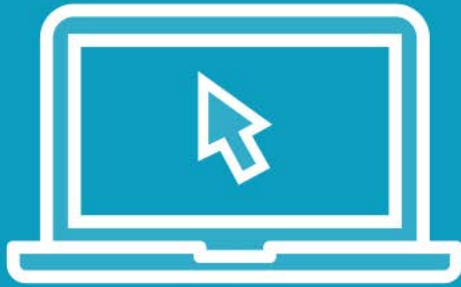
We get its routes

Order matters

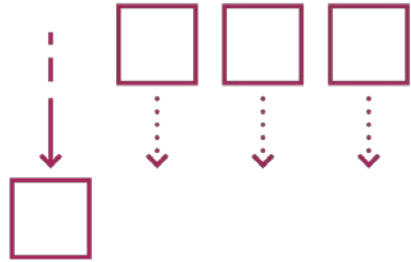


# Eager Loading Modules

Demo



# Lazy Loading Modules

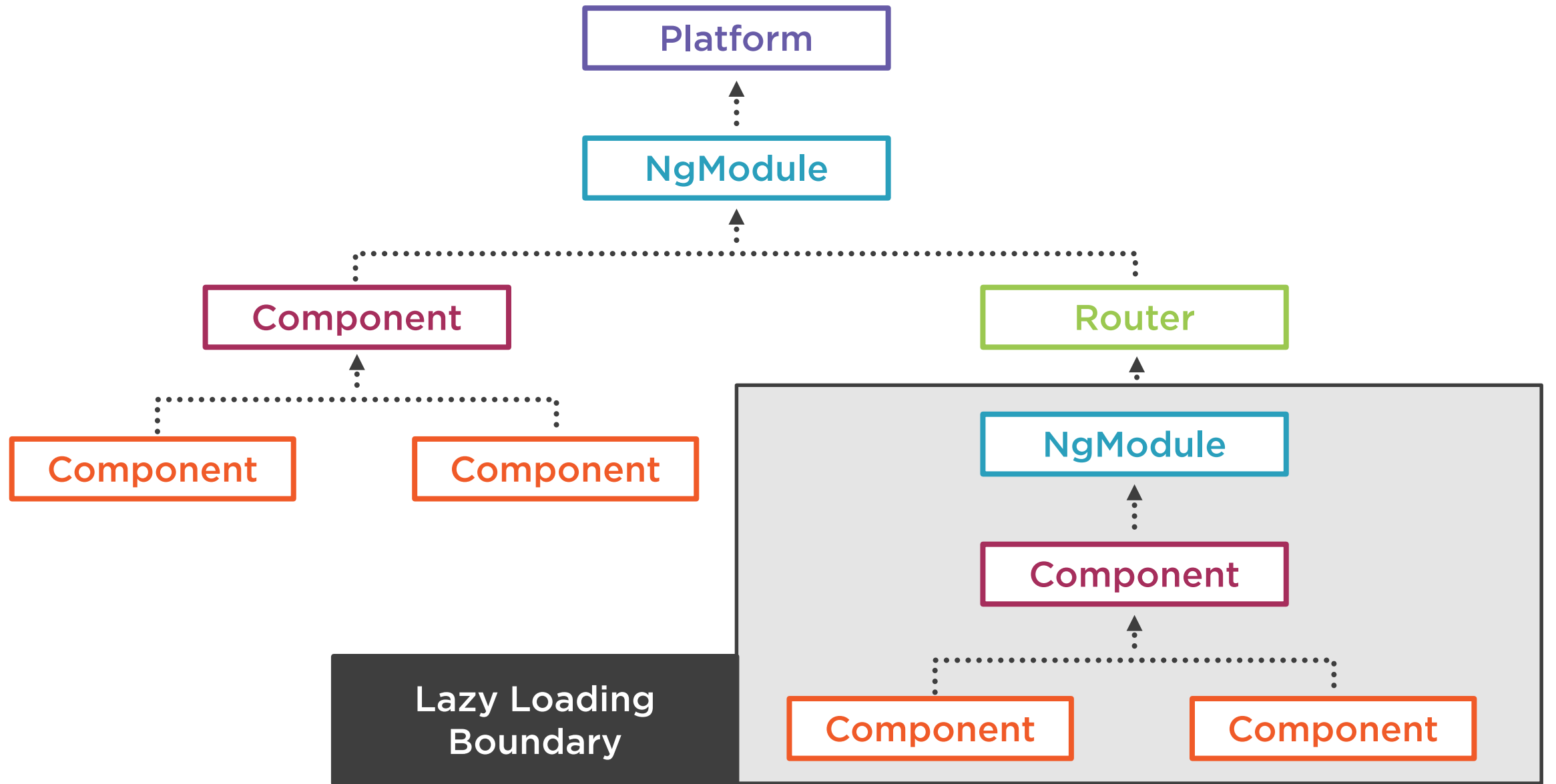


# Lazy Loading

Loading modules on demand, as needed, just in time

Lowers the initial payload, improving the app startup experience





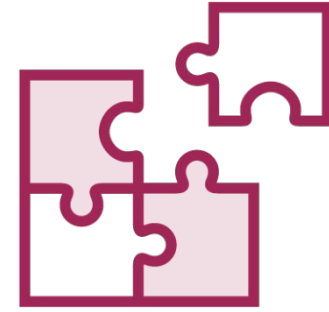
# Routing to Modules



## Eager Loading

Loads at startup

Ideal for modules users need immediately



## Lazy Loading

Loads on demand

Ideal for modules not immediately in the workflow



```
const routes: Routes = [  
  { path: '', pathMatch: 'full', redirectTo: 'characters' },  
  { path: 'characters', loadChildren: 'app/characters/characters.module#CharactersModule' },  
  { path: 'vehicles', loadChildren: 'app/vehicles/vehicles.module#VehiclesModule' },  
  { path: '**', pathMatch: 'full', component: PageNotFoundComponent },  
];
```

## Lazy Loading

Use **loadChildren**

Load the module by **path # name**

Do not import nor reference the module directly

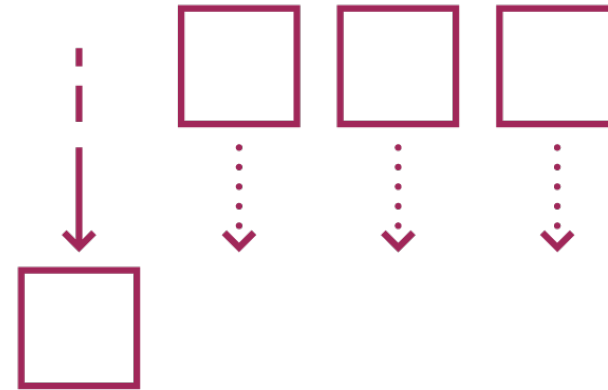


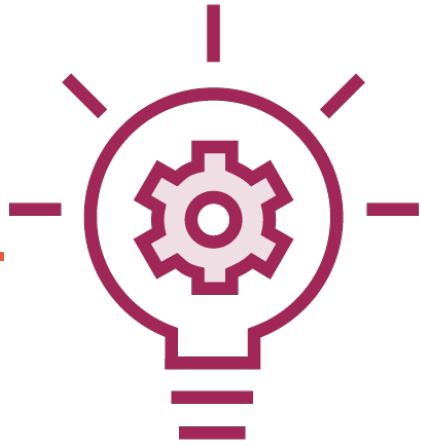


Demo



Lazy Loading





# Preload Strategies



# Preload Strategies

When a user navigates to a lazily loadable module, the content has to be fetched from the server, which may cause a delay.

The router can preload lazily loadable modules in the background



## app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, NoPreloading, Routes, RouterModule } from '@angular/router';

@NgModule({
  imports: [RouterModule.forRoot(routes, { preloadStrategy: PreloadAllModules })],
  exports: [RouterModule],
})
export class AppRoutingModule { }
```

Preload Strategy

## Preload Strategies

We can preload all or none, out of the box

Pass the strategy to **forRoot**



# Custom Preload Strategies



## preload-strategy.ts

Implement the interface

```
export class PreloadSelectedModulesList implements PreloadingStrategy {  
  preload(route: Route, load: Function): Observable<any> {  
    return route.data && route.data['preload'] ? load() : of(null);  
  }  
}
```

## app-routing.module.ts

Add the preload Observable

```
{  
  path: 'speakers', loadChildren: 'app/speakers/speakers.module#SpeakersModule',  
  data: { preload: true }  
},
```

## app-routing.module.ts

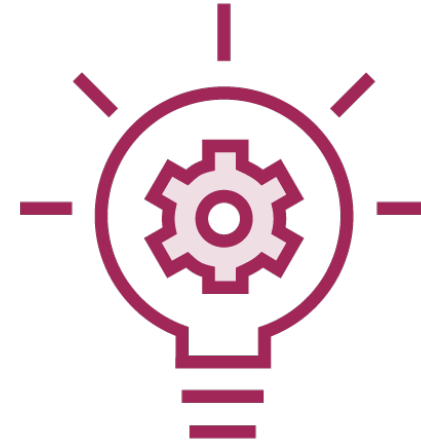
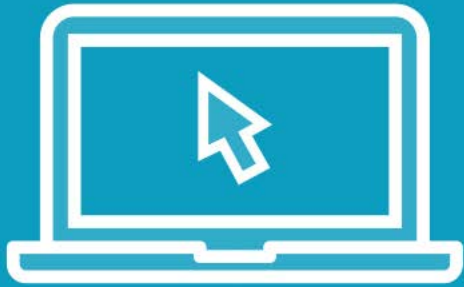
Provide the Strategy

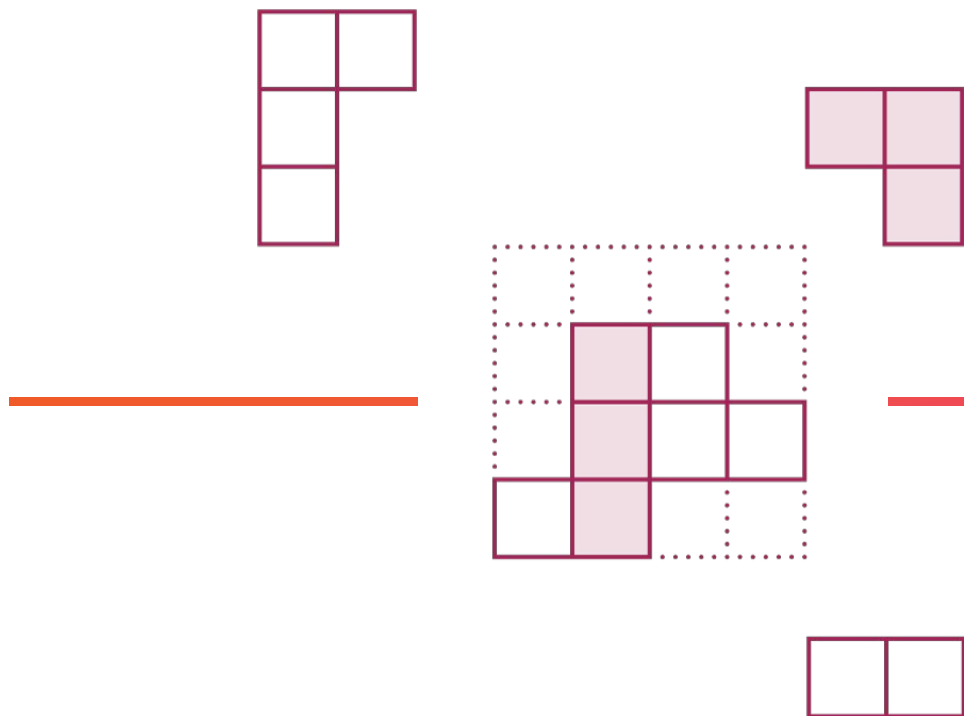
```
@NgModule({  
  imports: [RouterModule.forRoot(routes, { preloadingStrategy: PreloadSelectedModulesList })],  
  exports: [RouterModule],  
  providers: [PreloadSelectedModulesList]  
})  
export class AppRoutingModule { }
```



# Preload Strategies

Demo





# Feature Modules





# Basic Types of Feature Modules



Domain



Routed



Service



Widget

# Basic Types of Feature Modules



Domain



Routed



Service



Widget

e.g. MenuModule



Domain

Deliver a user experience dedicated to a particular application domain.

Typically imported once by the root AppModule, without routing.

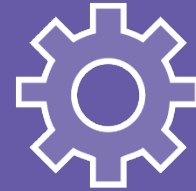




Domain



Routed



Service



Widget



e.g. CharactersModule



Routed

**Routed feature modules are Domain feature modules who are the targets of navigation routes.**

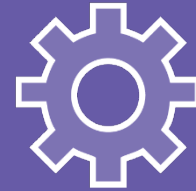




Domain



Routed



Service



Widget

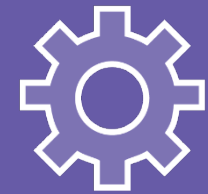


e.g. CoreModule

Service feature modules contain singleton services used by anyone across the app.

Generally all providers, no components.

We import them once in App Root Module.



Service

**Do not import these  
in other modules**

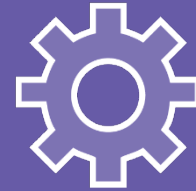




Domain



Routed



Service



Widget





e.g. SharedModule

Shared feature modules generally contain components, directives and pipes.

We import these in every module that uses them.



Widget



# AppModule



MenuModule



CoreModule



CharactersModule



VehiclesModule



SharedModule



# AppModule



MenuModule



CoreModule



CharactersModule



VehiclesModule



SharedModule



# AppModule



CoreModule



CharactersModule



VehiclesModule



SharedModule



MenuModule



# AppModule



MenuModule



CoreModule



CharactersModule

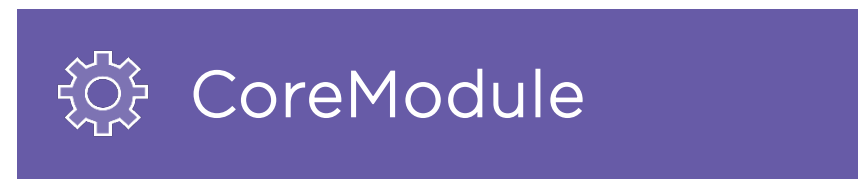
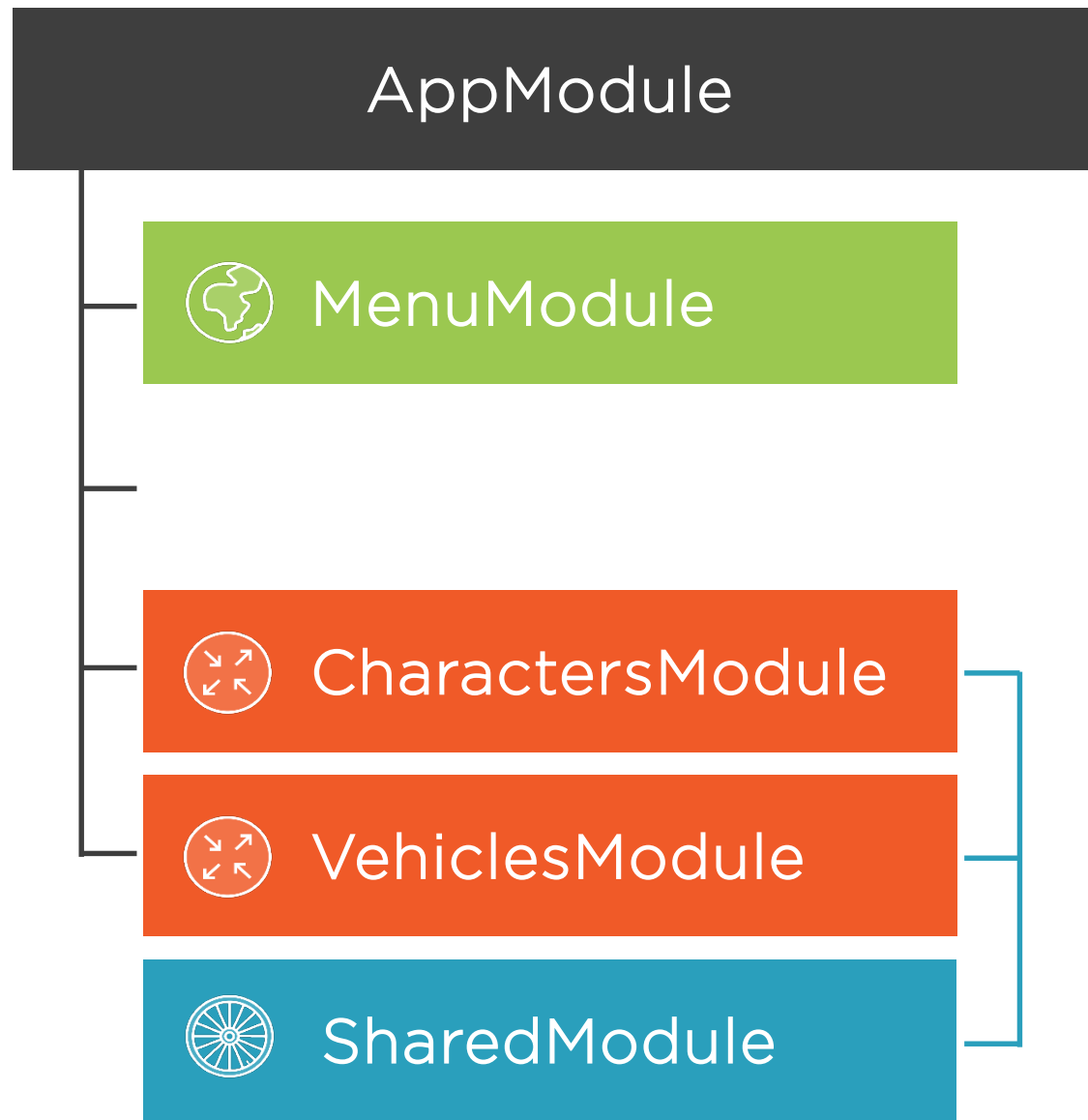


VehiclesModule



SharedModule





# AppModule



MenuModule



CoreModule



CharactersModule

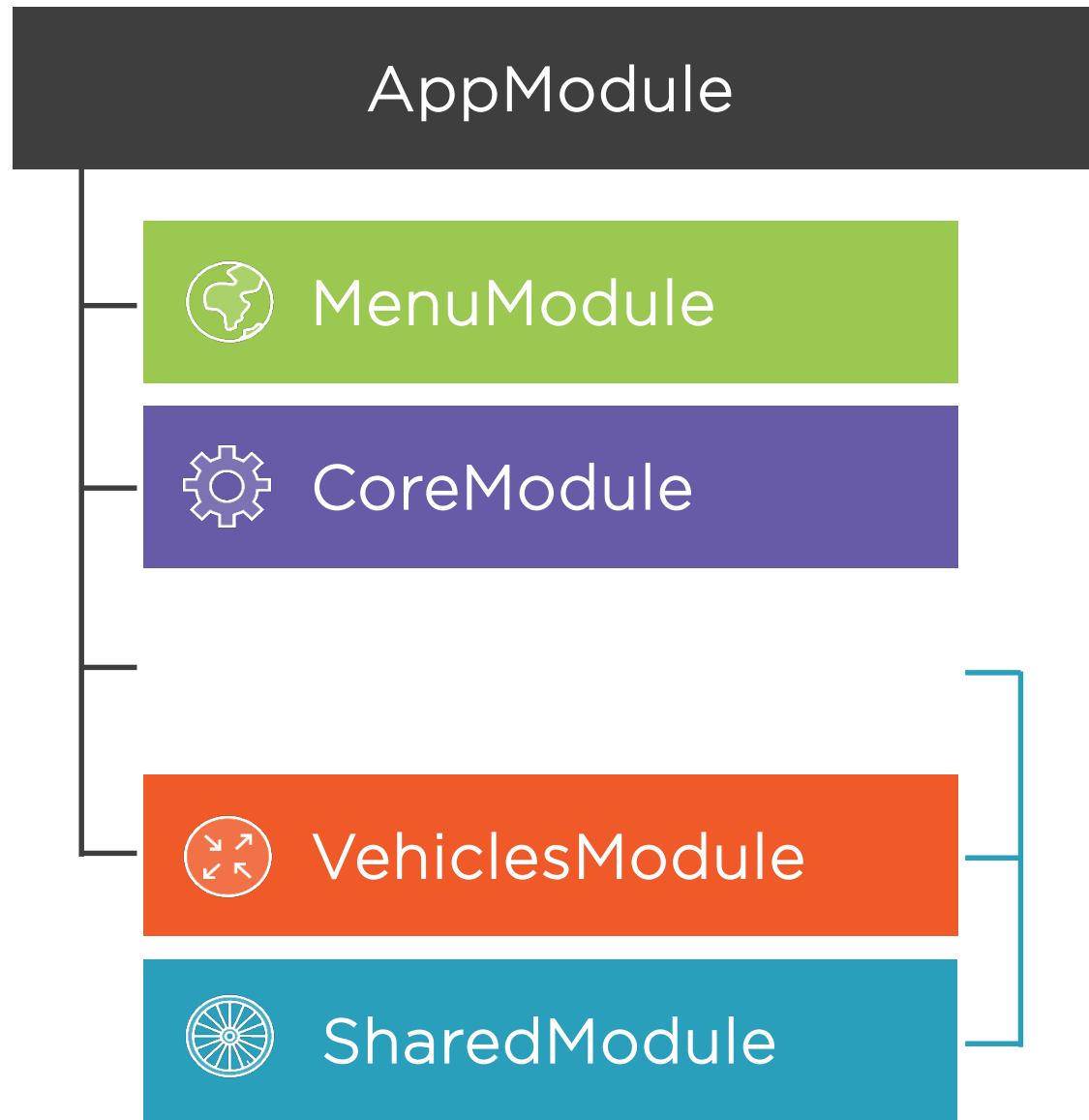


VehiclesModule



SharedModule







# AppModule



MenuModule



CoreModule



CharactersModule

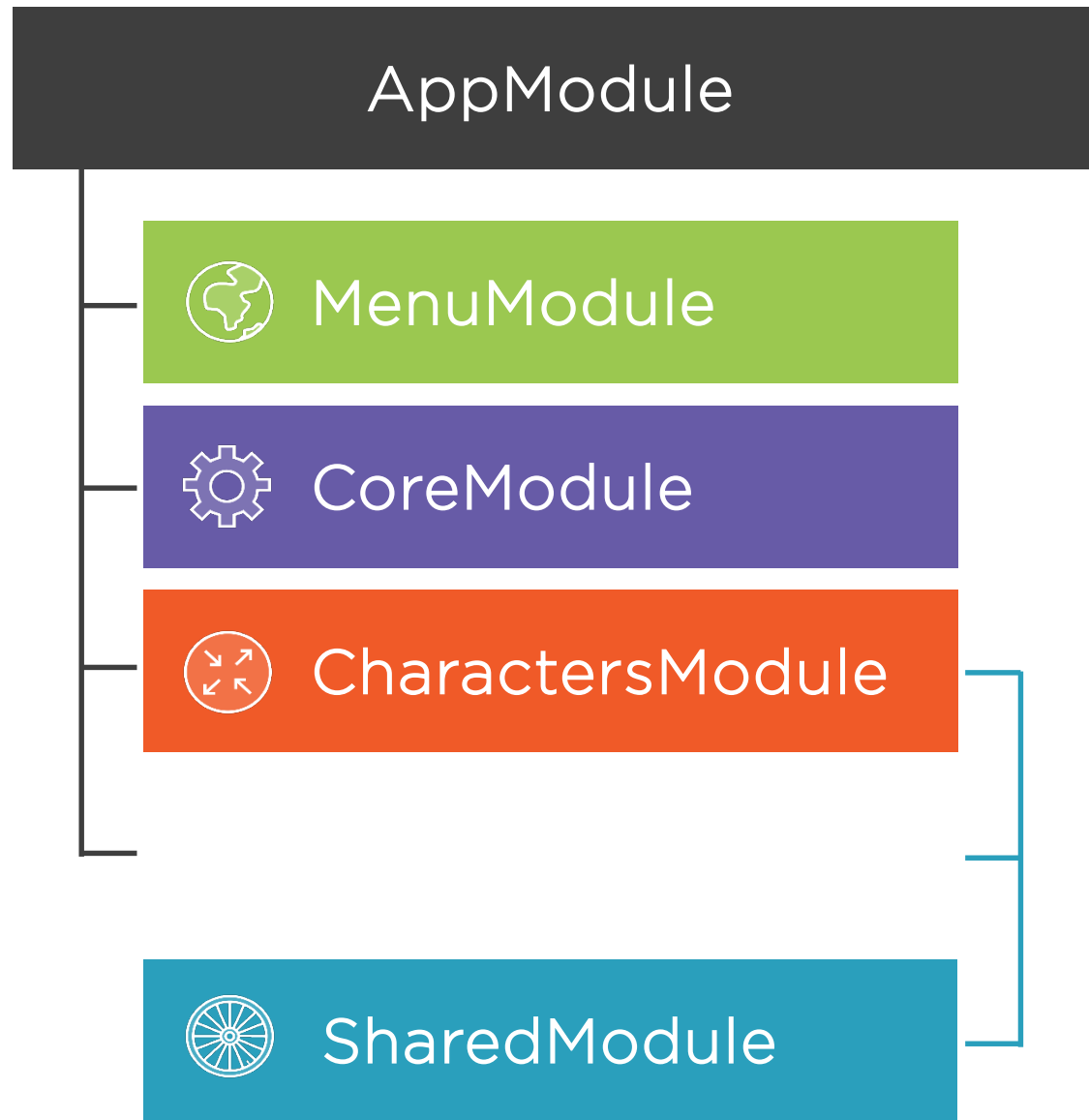


VehiclesModule



SharedModule





# AppModule



MenuModule



CoreModule



CharactersModule



VehiclesModule



SharedModule



# AppModule



MenuModule



CoreModule



CharactersModule







VehiclesModule



SharedModule

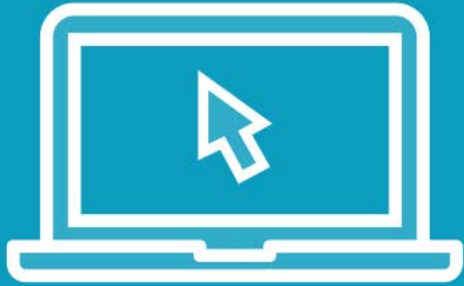


# Basic Types of Feature Modules

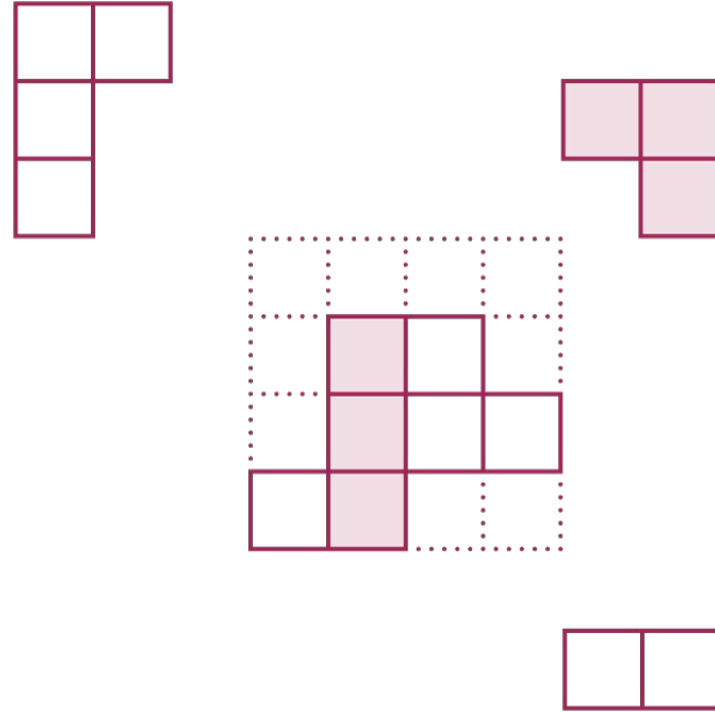
		Declarations	Providers	Exports	Imported By	Examples
	Domain	✓	Rare	Some	Feature	MenuModule
	Routed	✓	Rare	✗	Nobody if Lazy	CharactersModule
	Service	✗	✓	✗	AppModule	CoreModule, HttpModule
	Widget	✓	Rare	✓	Feature	CommonModule, SharedModule



Demo



# Feature Modules





Providers




It is important to consider  
how we want our providers to  
behave in the module system





# Providing Services and Their Behavior



A service provided  
in a component, is  
accessible to that  
component and to  
its children



# Providing Services and Their Behavior

When we import a module with providers, we'll get a new instance of that service upon injection



# Providing Services and Their Behavior

A service provided in a component, is accessible to that component and to its children

Be careful putting providers in a module that can be imported more than once

When we import a module with providers, we'll get a new instance of that service upon injection



## login.component.ts

```
@Component({  
  moduleId: module.id,  
  templateUrl: 'login.component.html',  
  providers: [LoginService]  
})  
export class LoginComponent implements OnDestroy {
```

This component and its children can inject it

## Component Providers

LoginService is only used by LoginComponent

Available to this component and its children



# Module Providers

Provided to the root injector

Available everywhere

```
@NgModule({  
  imports: [...  
  ],  
  declarations: [AppComponent, routableComponents],  
  providers: [  
    CanActivateAuthGuard, CharacterService, UserProfileService, VehicleService  
  ],  
  bootstrap: [AppComponent]  
})  
export class AppModule { }
```

Provide to the root injector

**app.module.ts**

# Summary



Eager Loading

Lazy Loading

Preload Strategies

Identifying Feature Modules

Providers

