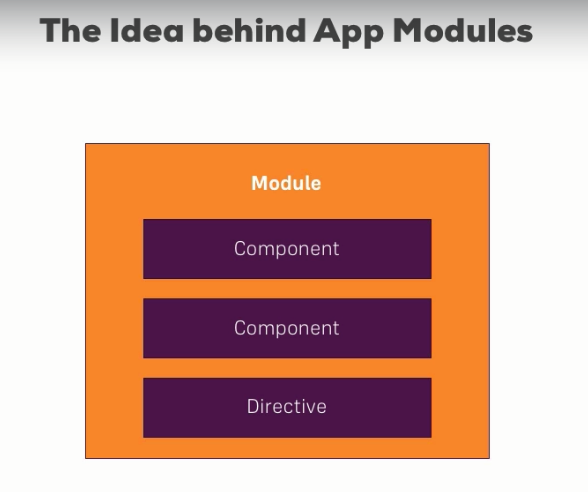
**Section 21 Using Angular Modules and Optimizing Apps**

**Section 21: Lecture 267//Module Introduction**

1. This section is about how to use modules and in general how you can optimize your angular application.
2. Modules play an important role in optimizing an application.
3. We will learn how we can increase the performance of the app and how we can decrease the file size and restructure your code in an easier way.

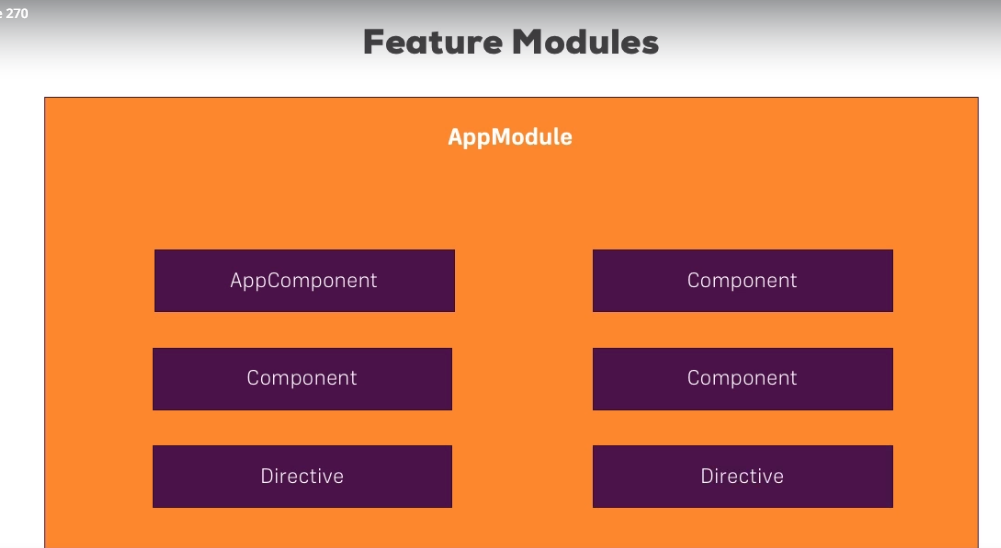
**Section 21: Lecture 268//The Idea behind Modules**

1. 

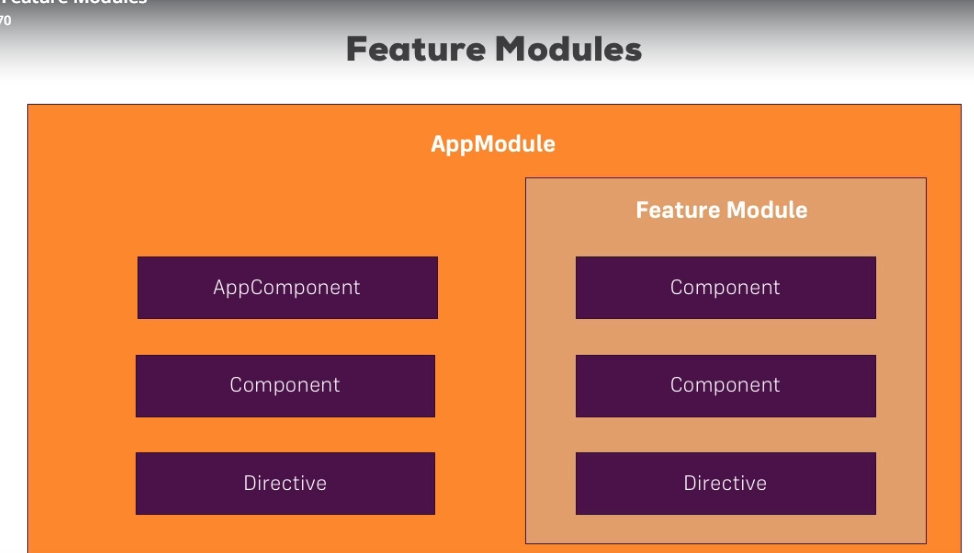
**Section 21: Lecture 269//Understanding the App Module**

**Section 21: Lecture 270//Understanding Feature Modules**

1. A typical custom module added by us in our app is called the feature module. Our application as of now is only having app module.
2. AppModule:



1. Feature Module:



**Section 21: Lecture 271//Creating a Recipe Feature Module**

1. In the recipes folder in our metadata we can see a lot of files or components, now each of this component is defined in the AppModule file.
2. All these components belong to our recipe feature. Now, we will go to recipe folder and will create our own file named recipe.module.ts.
3. Now, we will move all the recipe related components in this module and will leave the services in the app.module.ts as these are also used by the other parts of the Application.
4. CommonModule gives us access to all the common modules.
5. recipes.module.ts:
6. import { NgModule } from "@angular/core";
7. import { RecipesComponent } from "./recipes.component";
8. import { RecipeStartComponent } from "./recipe-start/recipe-start.component";
9. import { RecipeListComponent } from "./recipe-list/recipe-list.component";
10. import { RecipeEditComponent } from "./recipe-edit/recipe-edit.component";
11. import { RecipeDetailComponent } from "./recipe-detail/recipe-detail.component";
12. import { RecipeItemComponent } from "./recipe-list/recipe-item/recipe-item.component";
13. import { ReactiveFormsModule } from "@angular/forms";
14. import { CommonModule } from "@angular/common";
15. @NgModule({
16. declarations:[
17. RecipesComponent,
18. RecipeStartComponent,
19. RecipeListComponent,
20. RecipeEditComponent,
21. RecipeDetailComponent,
22. RecipeItemComponent
23. ],
24. imports: [
25. CommonModule,
26. ReactiveFormsModule
27. ]
28. })
29. export class RecipesModule{
31. }

6. app.module.ts:

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { FormsModule } from '@angular/forms';

import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';

import { HeaderComponent } from './header/header.component';

import { ShoppingListComponent } from './shopping-list/shopping-list.component';

import { ShoppingEditComponent } from './shopping-list/shopping-edit/shopping-edit.component';

import { DropdownDirective } from './shared/dropdown.directive';

import { ShoppingListService } from './shopping-list/shopping-list.service';

import { AppRoutingModule } from './app-routing.module';

import { RecipeService } from './recipes/recipe.service';

import { DataStorageService } from './shared/data-storage.service';

import { SignupComponent } from './auth/signup/signup.component';

import { SigninComponent } from './auth/signin/signin.component';

import { AuthService } from './auth/auth.service';

import { AuthGuard } from './auth/auth-guard.service';

import { RecipesModule } from './recipes/recipes.module';

@NgModule({

declarations: [

AppComponent,

HeaderComponent,

ShoppingListComponent,

ShoppingEditComponent,

DropdownDirective,

SignupComponent,

SigninComponent

],

imports: [

BrowserModule,

FormsModule,

HttpModule,

AppRoutingModule,

RecipesModule

],

providers: [ShoppingListService, RecipeService, DataStorageService, AuthService, AuthGuard],

bootstrap: [AppComponent]

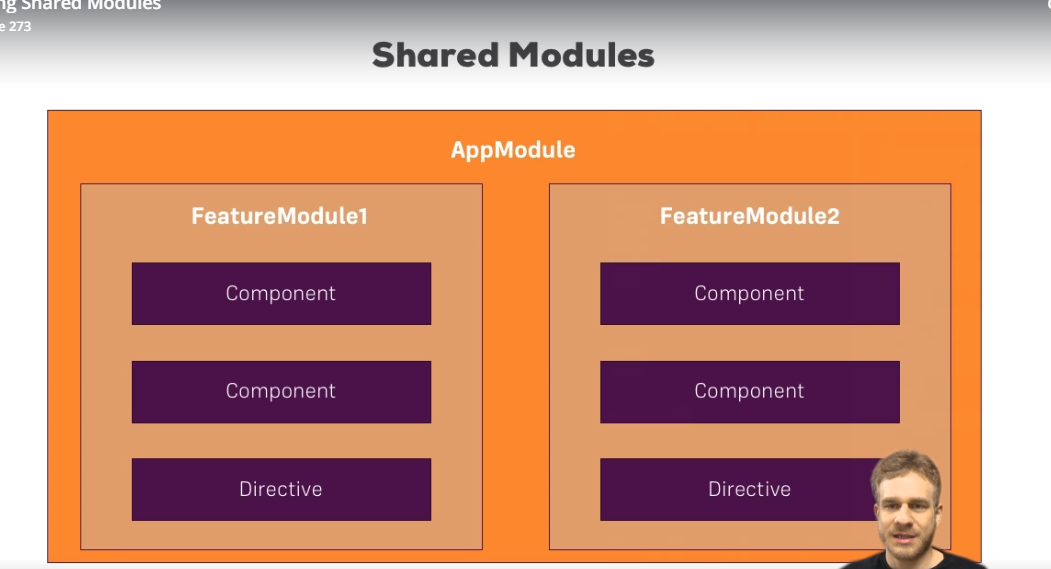
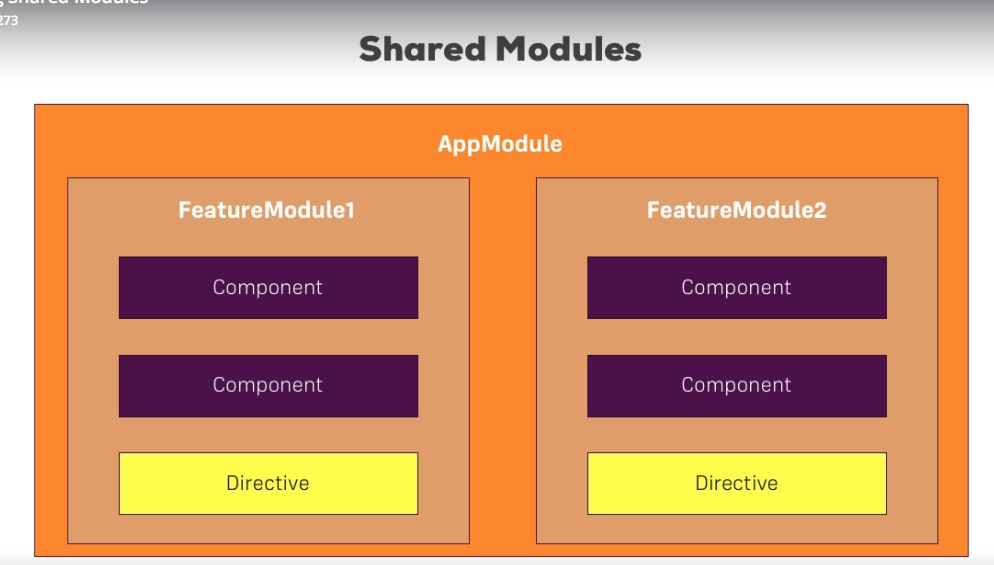
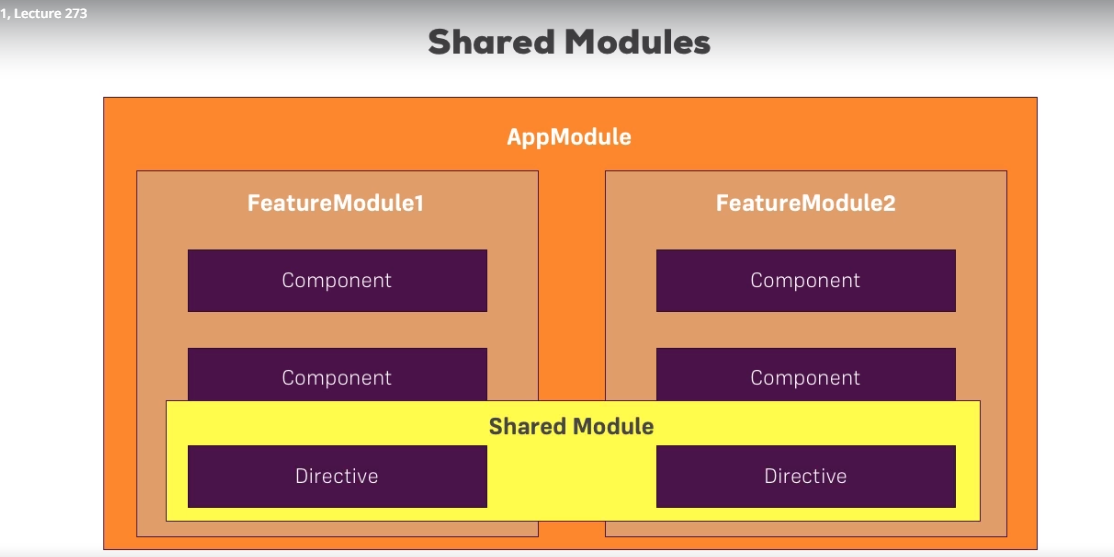
})

export class AppModule { }

**Section 21: Lecture 272//Registering Routes in a Feature Module**

1. Now, we are using app routing module in recipes module but we have declared it only in app module – this is a problem
2. Point is - if we create a feature module we also have to move the routes related to feature modules.
3. Now, we will create a new routing module in the recipes module i.e. recipes-routing.module.ts
4. recipes.routing.module:
5. import { NgModule } from "@angular/core";
6. import { Routes, RouterModule } from "@angular/router";
7. import { RecipesComponent } from "./recipes.component";
8. import { RecipeStartComponent } from "./recipe-start/recipe-start.component";
9. import { RecipeEditComponent } from "./recipe-edit/recipe-edit.component";
10. import { AuthGuard } from "../auth/auth-guard.service";
11. import { RecipeDetailComponent } from "./recipe-detail/recipe-detail.component";
12. const recipesRoutes: Routes = [
13. { path: 'recipes', component: RecipesComponent, children: [
14. { path: '', component: RecipeStartComponent },
15. { path: 'new', component: RecipeEditComponent, canActivate: [AuthGuard] },
16. { path: ':id', component: RecipeDetailComponent },
17. { path: ':id/edit', component: RecipeEditComponent, canActivate: [AuthGuard] },
18. ] }
19. ]
20. @NgModule({
21. imports:[
22. RouterModule.forChild(recipesRoutes)
23. ],
24. exports: [recipesRoutes]
25. })
26. export class RecipesRoutingModule{
27. }
28. recipes.module.ts:
29. import { NgModule } from "@angular/core";
30. import { RecipesComponent } from "./recipes.component";
31. import { RecipeStartComponent } from "./recipe-start/recipe-start.component";
32. import { RecipeListComponent } from "./recipe-list/recipe-list.component";
33. import { RecipeEditComponent } from "./recipe-edit/recipe-edit.component";
34. import { RecipeDetailComponent } from "./recipe-detail/recipe-detail.component";
35. import { RecipeItemComponent } from "./recipe-list/recipe-item/recipe-item.component";
36. import { ReactiveFormsModule } from "@angular/forms";
37. import { CommonModule } from "@angular/common";
38. import { RecipesRoutingModule } from "./recipes-routing.module";
39. @NgModule({
40. declarations:[
41. RecipesComponent,
42. RecipeStartComponent,
43. RecipeListComponent,
44. RecipeEditComponent,
45. RecipeDetailComponent,
46. RecipeItemComponent
47. ],
48. imports: [
49. CommonModule,
50. ReactiveFormsModule,
51. RecipesRoutingModule
52. ]
53. })
54. export class RecipesModule{
56. }
57. app.module.ts:
58. import { BrowserModule } from '@angular/platform-browser';
59. import { NgModule } from '@angular/core';
60. import { FormsModule } from '@angular/forms';
61. import { HttpModule } from '@angular/http';
62. import { AppComponent } from './app.component';
63. import { HeaderComponent } from './header/header.component';
64. import { ShoppingListComponent } from './shopping-list/shopping-list.component';
65. import { ShoppingEditComponent } from './shopping-list/shopping-edit/shopping-edit.component';
66. import { DropdownDirective } from './shared/dropdown.directive';
67. import { ShoppingListService } from './shopping-list/shopping-list.service';
68. import { AppRoutingModule } from './app-routing.module';
69. import { RecipeService } from './recipes/recipe.service';
70. import { DataStorageService } from './shared/data-storage.service';
71. import { SignupComponent } from './auth/signup/signup.component';
72. import { SigninComponent } from './auth/signin/signin.component';
73. import { AuthService } from './auth/auth.service';
74. import { AuthGuard } from './auth/auth-guard.service';
75. import { RecipesModule } from './recipes/recipes.module';
76. @NgModule({
77. declarations: [
78. AppComponent,
79. HeaderComponent,
80. ShoppingListComponent,
81. ShoppingEditComponent,
82. DropdownDirective,
83. SignupComponent,
84. SigninComponent
85. ],
86. imports: [
87. BrowserModule,
88. FormsModule,
89. HttpModule,
90. AppRoutingModule,
91. RecipesModule
92. ],
93. providers: [ShoppingListService, RecipeService, DataStorageService, AuthService, AuthGuard],
94. bootstrap: [AppComponent]
95. })
96. export class AppModule { }

**Section 21: Lecture 273//Understanding Shared Modules**

1. 
2. 
3. 

**Section 21: Lecture 274//Creating a shared Module**

1. Here we will create a new shared module in the shared folder, and then we will export the DropdownDirective from this module.
2. If we create a directive it needs to be declared only once, but at least once.
3. The idea behind the shared module is that, now we can import the shared module into any module.
4. Now, to be able to use the drop down directive we must also export it first.
5. We know that we can only access that things which are available inside the module. To use things outside the module we must be able to export things from certain outside module and then import that module here.
6. Now, we can also add something else to this shared module i.e. the common module.
7. shared.module.ts:
8. import { NgModule } from "@angular/core";
9. import { DropdownDirective } from "./dropdown.directive";
10. import { CommonModule } from "@angular/common";
11. @NgModule({
12. declarations: [
13. DropdownDirective
14. ],
15. exports: [
16. CommonModule,
17. DropdownDirective
18. ]
19. })
20. export class SharedModule{
21. }

8. recipes.module.ts:

import { NgModule } from "@angular/core";

import { RecipesComponent } from "./recipes.component";

import { RecipeStartComponent } from "./recipe-start/recipe-start.component";

import { RecipeListComponent } from "./recipe-list/recipe-list.component";

import { RecipeEditComponent } from "./recipe-edit/recipe-edit.component";

import { RecipeDetailComponent } from "./recipe-detail/recipe-detail.component";

import { RecipeItemComponent } from "./recipe-list/recipe-item/recipe-item.component";

import { ReactiveFormsModule } from "@angular/forms";

import { CommonModule } from "@angular/common";

import { RecipesRoutingModule } from "./recipes-routing.module";

import { SharedModule } from "../shared/shared.module";

@NgModule({

declarations:[

RecipesComponent,

RecipeStartComponent,

RecipeListComponent,

RecipeEditComponent,

RecipeDetailComponent,

RecipeItemComponent

],

imports: [

CommonModule,

ReactiveFormsModule,

RecipesRoutingModule,

SharedModule

]

})

export class RecipesModule{

}

9. app.module.ts:

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { FormsModule } from '@angular/forms';

import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';

import { HeaderComponent } from './header/header.component';

import { ShoppingListComponent } from './shopping-list/shopping-list.component';

import { ShoppingEditComponent } from './shopping-list/shopping-edit/shopping-edit.component';

import { ShoppingListService } from './shopping-list/shopping-list.service';

import { AppRoutingModule } from './app-routing.module';

import { RecipeService } from './recipes/recipe.service';

import { DataStorageService } from './shared/data-storage.service';

import { SignupComponent } from './auth/signup/signup.component';

import { SigninComponent } from './auth/signin/signin.component';

import { AuthService } from './auth/auth.service';

import { AuthGuard } from './auth/auth-guard.service';

import { RecipesModule } from './recipes/recipes.module';

import { SharedModule } from './shared/shared.module';

@NgModule({

declarations: [

AppComponent,

HeaderComponent,

ShoppingListComponent,

ShoppingEditComponent,

SignupComponent,

SigninComponent

],

imports: [

BrowserModule,

FormsModule,

HttpModule,

AppRoutingModule,

RecipesModule,

SharedModule

],

providers: [ShoppingListService, RecipeService, DataStorageService, AuthService, AuthGuard],

bootstrap: [AppComponent]

})

export class AppModule { }

**Section 21: Lecture 275//Creating a shopping list feature module**

1. shopping-list.module.ts:
2. import { NgModule } from "@angular/core";
3. import { ShoppingEditComponent } from "./shopping-edit/shopping-edit.component";
4. import { ShoppingListComponent } from "./shopping-list.component";
5. import { CommonModule } from "@angular/common";
6. import { FormsModule } from "@angular/forms";
7. @NgModule({
8. declarations: [
9. ShoppingEditComponent,
10. ShoppingListComponent
11. ],
12. imports: [
13. CommonModule,
14. FormsModule
15. ]
16. })
17. export class ShoppoingListModule{
18. }
19. app.module.ts:
20. import { BrowserModule } from '@angular/platform-browser';
21. import { NgModule } from '@angular/core';
22. import { HttpModule } from '@angular/http';
23. import { AppComponent } from './app.component';
24. import { HeaderComponent } from './header/header.component';
25. import { ShoppingListService } from './shopping-list/shopping-list.service';
26. import { AppRoutingModule } from './app-routing.module';
27. import { RecipeService } from './recipes/recipe.service';
28. import { DataStorageService } from './shared/data-storage.service';
29. import { SignupComponent } from './auth/signup/signup.component';
30. import { SigninComponent } from './auth/signin/signin.component';
31. import { AuthService } from './auth/auth.service';
32. import { AuthGuard } from './auth/auth-guard.service';
33. import { RecipesModule } from './recipes/recipes.module';
34. import { SharedModule } from './shared/shared.module';
35. import { ShoppoingListModule } from './shopping-list/shopping-list.module';
36. @NgModule({
37. declarations: [
38. AppComponent,
39. HeaderComponent,
40. SignupComponent,
41. SigninComponent
42. ],
43. imports: [
44. BrowserModule,
45. HttpModule,
46. AppRoutingModule,
47. RecipesModule,
48. SharedModule,
49. ShoppoingListModule
50. ],
51. providers: [ShoppingListService, RecipeService, DataStorageService, AuthService, AuthGuard],
52. bootstrap: [AppComponent]
53. })
54. export class AppModule { }

**Section 21: Lecture 276//Loading Components via Sectors vs routing**

**Section 21: Lecture 277//A Common Gotcha**

1. Forms module should be imported in every module.

**Section 21: Lecture 278//Creating the Auth Feature Module**

1. Let’s build our auth feature module
2. Any module which is not app.module is a child module.
3. auth.module.ts:
4. import { NgModule } from "@angular/core";
5. import { SigninComponent } from "./signin/signin.component";
6. import { SignupComponent } from "./signup/signup.component";
7. import { FormsModule } from "@angular/forms";
8. import { AuthRoutingModule } from "./auth-routing.module";
9. @NgModule({
10. declarations:[
11. SigninComponent,
12. SignupComponent
13. ],
14. imports: [
15. FormsModule,
16. AuthRoutingModule
17. ]
18. })
19. export class AuthModule{
20. }
21. auth-routing.module.ts:
22. import { NgModule } from "@angular/core";
23. import { Routes, RouterModule } from "@angular/router";
24. import { SignupComponent } from "./signup/signup.component";
25. import { SigninComponent } from "./signin/signin.component";
26. const authRoutes: Routes = [
27. { path: 'signup', component: SignupComponent },
28. { path: 'signin', component: SigninComponent }
29. ];
30. @NgModule({
31. imports:[
32. RouterModule.forChild(authRoutes)
33. ],
34. exports:[
35. RouterModule
36. ]
37. })
38. export class AuthRoutingModule{
39. }

6. app.module.ts:

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';

import { HeaderComponent } from './header/header.component';

import { ShoppingListService } from './shopping-list/shopping-list.service';

import { AppRoutingModule } from './app-routing.module';

import { RecipeService } from './recipes/recipe.service';

import { DataStorageService } from './shared/data-storage.service';

import { AuthService } from './auth/auth.service';

import { AuthGuard } from './auth/auth-guard.service';

import { RecipesModule } from './recipes/recipes.module';

import { SharedModule } from './shared/shared.module';

import { ShoppoingListModule } from './shopping-list/shopping-list.module';

import { AuthModule } from './auth/auth.module';

@NgModule({

declarations: [

AppComponent,

HeaderComponent

],

imports: [

BrowserModule,

HttpModule,

AppRoutingModule,

RecipesModule,

SharedModule,

ShoppoingListModule,

AuthModule

],

providers: [ShoppingListService, RecipeService, DataStorageService, AuthService, AuthGuard],

bootstrap: [AppComponent]

})

export class AppModule { }

7. app-routing.mosule.ts:

import { NgModule } from '@angular/core';

import { Routes, RouterModule } from '@angular/router';

import { ShoppingListComponent } from './shopping-list/shopping-list.component';

const appRoutes: Routes = [

{ path: '', redirectTo: '/recipes', pathMatch: 'full' },

{ path: 'shopping-list', component: ShoppingListComponent }

];

@NgModule({

imports: [RouterModule.forRoot(appRoutes)],

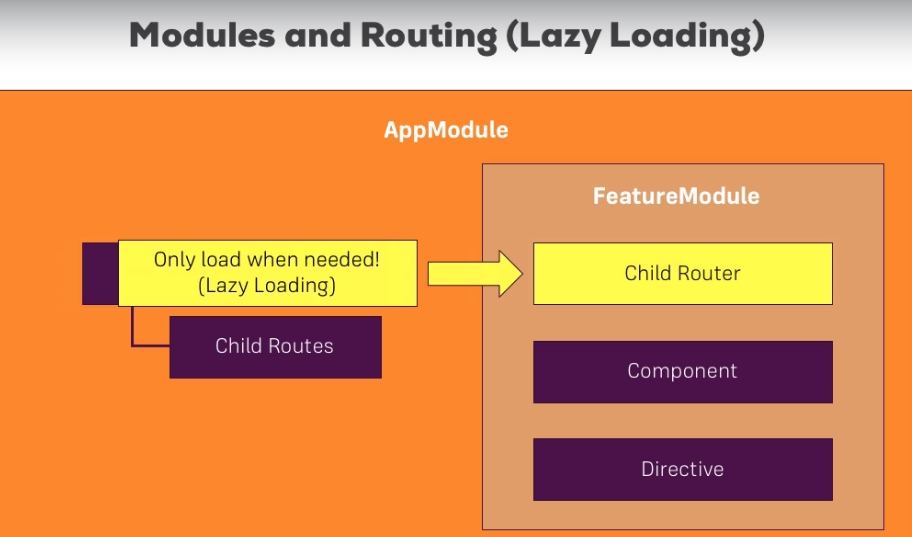
exports: [RouterModule]

})

export class AppRoutingModule {

}

**Section 21: Lecture 279//Modules and Routing(Lazy Loading)**

1. In the last lectures we built a lot of feature modules and shared modules. This far we did this to make our modules leaner and to restructure our code. And to quickly see which elements our app features use.
2. It’s a best practice to restructure our project so that it becomes maintainable. But it does not improve our app performance. Now, we will look at how we can improve our app performance.
3. Now, there are chances that the user does not visit any particular module in our application.
4. Our JavaScript bundle which contains our whole app loads when the user visits the index page. A lot of the code might never be used as the user does not visit that part of the application.
5. So, we can lazily load our feature module and the child router instead of loading the whole code in the application.
6. That means that this module will only be used if we visit the route which will lead to this page.
7. 

**Section 21: Lecture 280//Adding lazy loading to the Recipes Module**

1. Let’s add Lazy loading to our recipe book application.
2. In this application we have 3 feature areas
3. The auth feature
4. The shopping-list feature
5. The recipes feature.
6. In our app we might not always visit the shopping list area. Now, here we will also add a new component to make our recipes area optional to visit.
7. ng g c home --spec false
8. As of now, we have defined the shopping-list module in the AppModule, so, it gets invoked eagerly when AppModule loads and also recipes module.
9. app-routing.module.ts:
10. import { NgModule } from '@angular/core';
11. import { Routes, RouterModule } from '@angular/router';
12. import { ShoppingListComponent } from './shopping-list/shopping-list.component';
13. import { HomeComponent } from './home/home.component';
14. const appRoutes: Routes = [
15. { path: '', component: HomeComponent },
16. { path: 'recipes', loadChildren: './recipes/recipes.module#RecipesModule'},
17. { path: 'shopping-list', component: ShoppingListComponent }
18. ];
19. @NgModule({
20. imports: [RouterModule.forRoot(appRoutes)],
21. exports: [RouterModule]
22. })
23. export class AppRoutingModule {
24. }
25. app.module.ts:
26. import { BrowserModule } from '@angular/platform-browser';
27. import { NgModule } from '@angular/core';
28. import { HttpModule } from '@angular/http';
29. import { AppComponent } from './app.component';
30. import { HeaderComponent } from './header/header.component';
31. import { ShoppingListService } from './shopping-list/shopping-list.service';
32. import { AppRoutingModule } from './app-routing.module';
33. import { RecipeService } from './recipes/recipe.service';
34. import { DataStorageService } from './shared/data-storage.service';
35. import { AuthService } from './auth/auth.service';
36. import { AuthGuard } from './auth/auth-guard.service';
37. import { SharedModule } from './shared/shared.module';
38. import { ShoppoingListModule } from './shopping-list/shopping-list.module';
39. import { AuthModule } from './auth/auth.module';
40. import { HomeComponent } from './home/home.component';
41. @NgModule({
42. declarations: [
43. AppComponent,
44. HeaderComponent,
45. HomeComponent
46. ],
47. imports: [
48. BrowserModule,
49. HttpModule,
50. AppRoutingModule,
51. SharedModule,
52. ShoppoingListModule,
53. AuthModule
54. ],
55. providers: [ShoppingListService, RecipeService, DataStorageService, AuthService, AuthGuard],
56. bootstrap: [AppComponent]
57. })
58. export class AppModule { }

9. recipes-routing.module.ts:

import { NgModule } from "@angular/core";

import { Routes, RouterModule } from "@angular/router";

import { RecipesComponent } from "./recipes.component";

import { RecipeStartComponent } from "./recipe-start/recipe-start.component";

import { RecipeEditComponent } from "./recipe-edit/recipe-edit.component";

import { AuthGuard } from "../auth/auth-guard.service";

import { RecipeDetailComponent } from "./recipe-detail/recipe-detail.component";

const recipesRoutes: Routes = [

{ path: '', component: RecipesComponent, children: [

{ path: '', component: RecipeStartComponent },

{ path: 'new', component: RecipeEditComponent, canActivate: [AuthGuard] },

{ path: ':id', component: RecipeDetailComponent },

{ path: ':id/edit', component: RecipeEditComponent, canActivate: [AuthGuard] },

] }

]

@NgModule({

imports:[

RouterModule.forChild(recipesRoutes)

],

exports: [recipesRoutes]

})

export class RecipesRoutingModule{

}

**Section 21: Lecture 281//Protecting Lazy Loaded Routes with canLoad**

What if you want to use route protection (canActivate  to be precise) on lazily loaded routes?

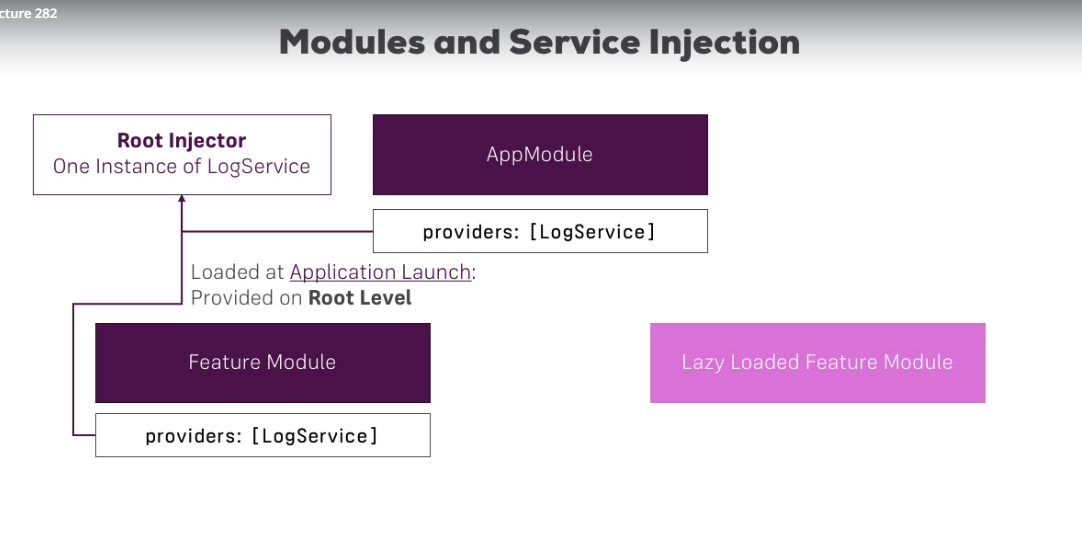
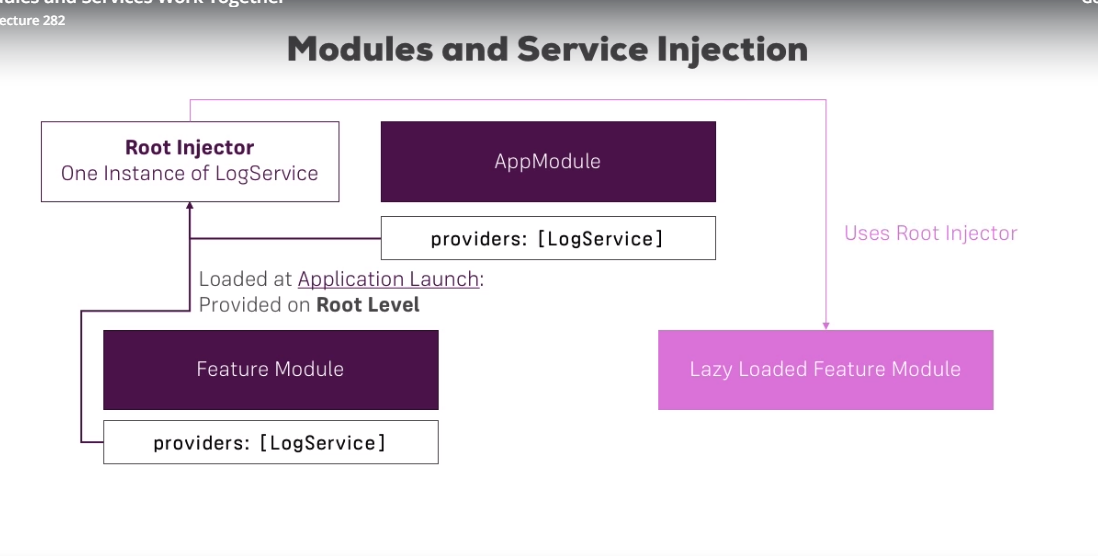
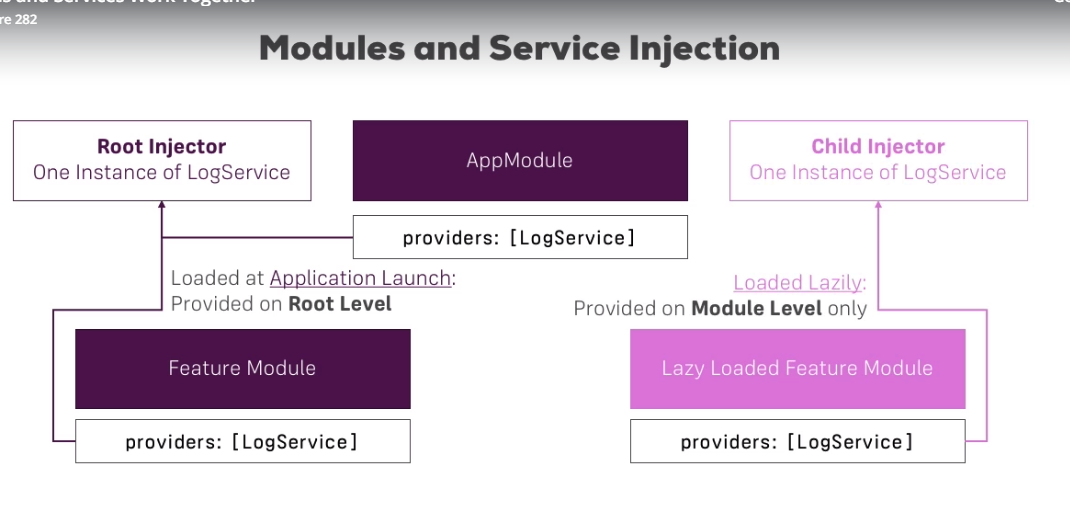
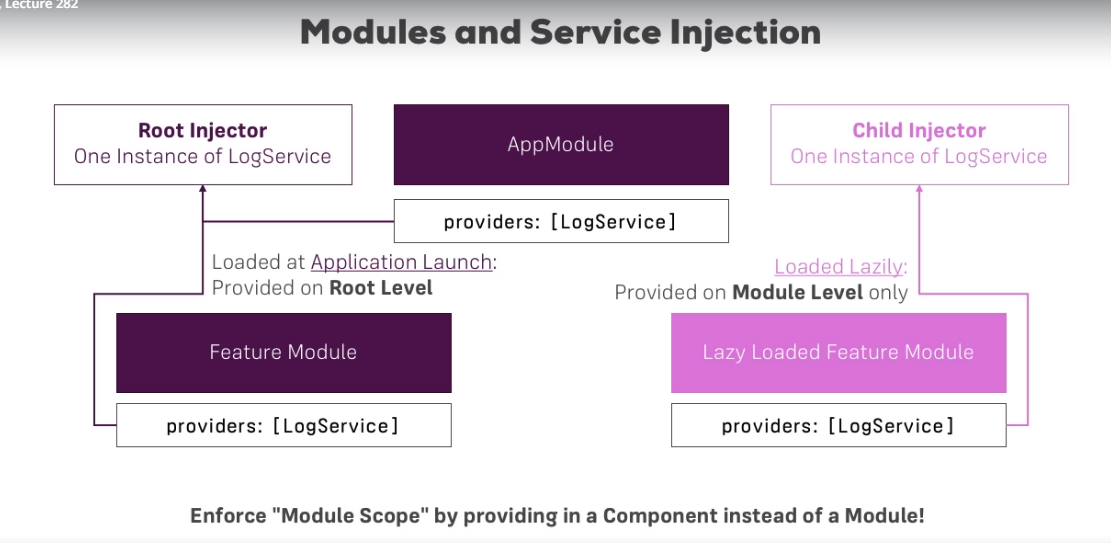
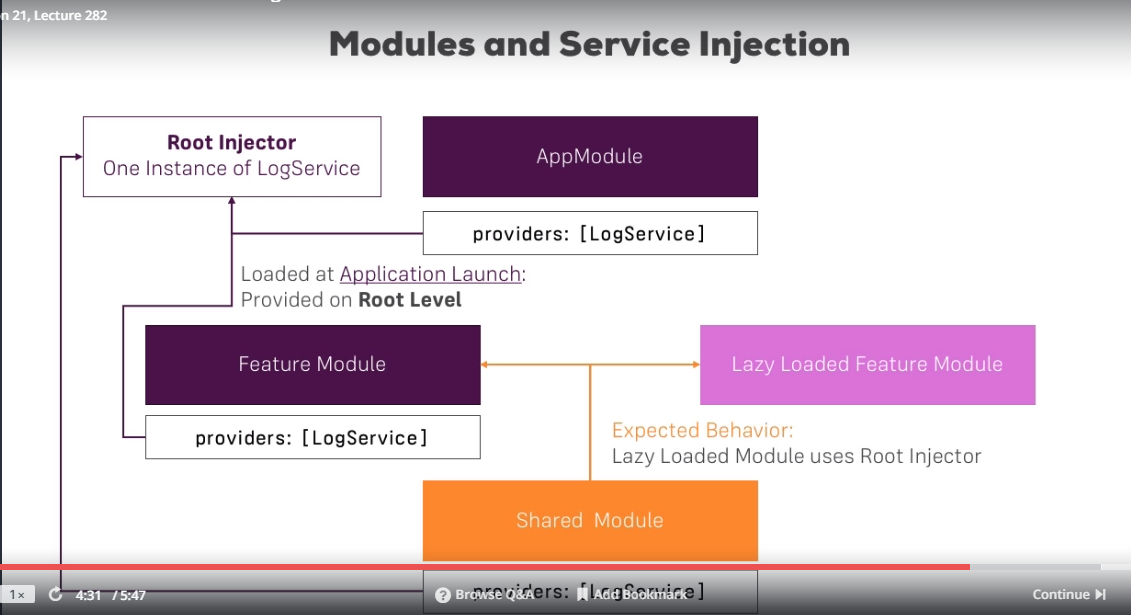
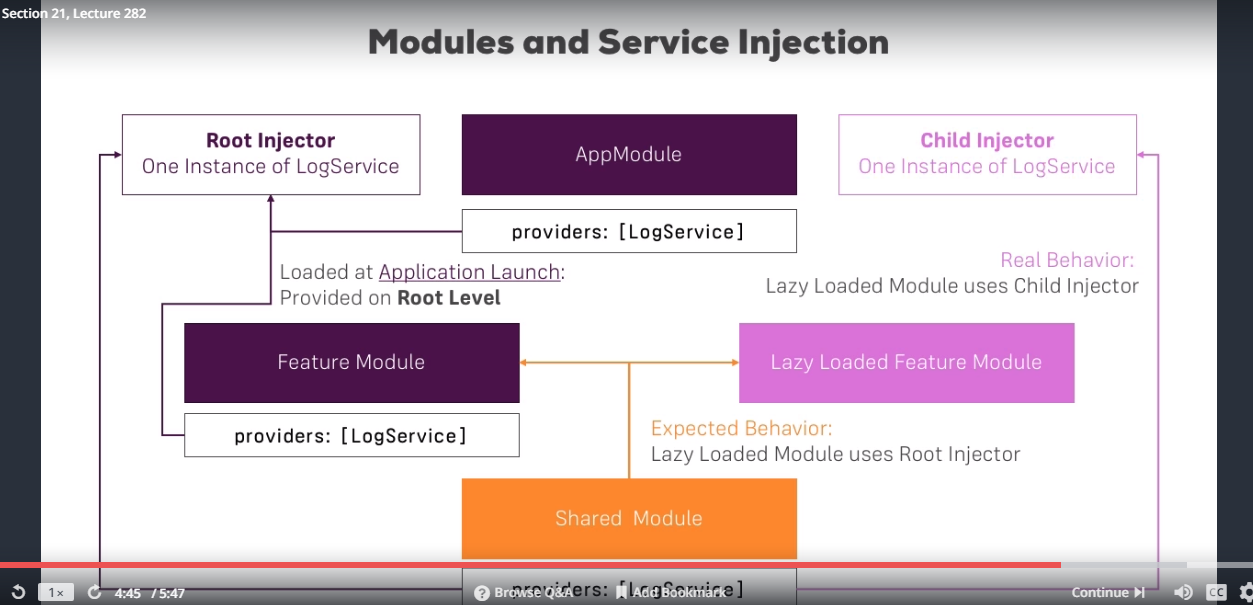
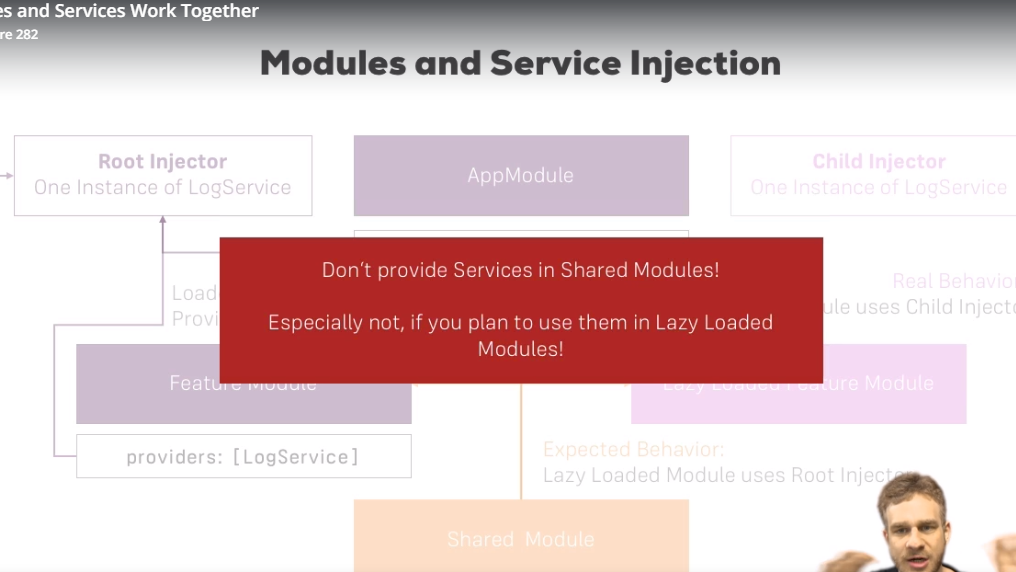
You can add canActivate to the lazy loaded routes but that of course means, that you might load code which in the end can't get accessed anyways. It would be better to check that BEFORE loading the code.

You can enforce this behavior by adding the canLoad  guard to the route which points to the lazily loaded module:

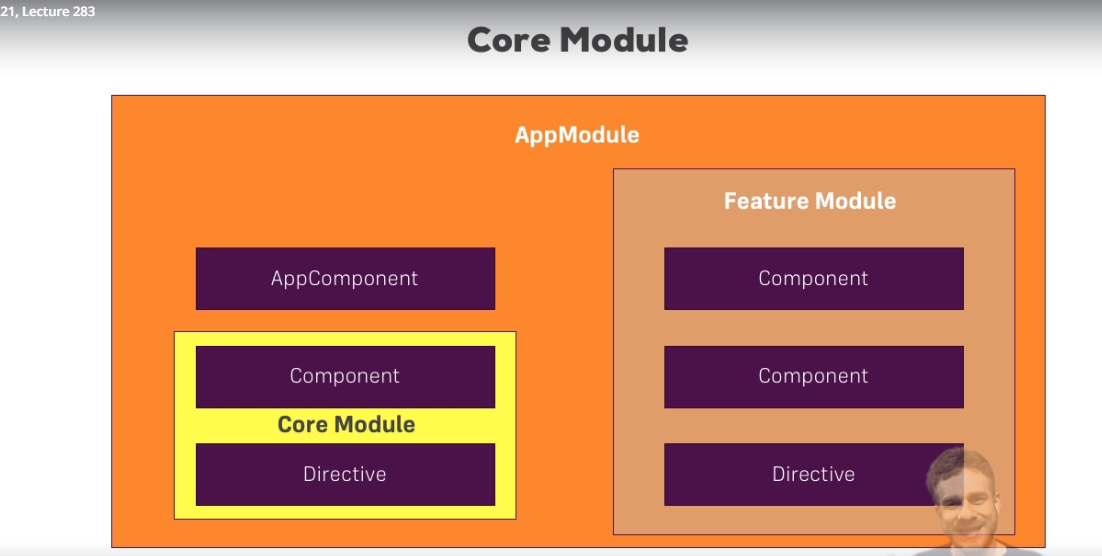
{ path: 'recipes', loadChildren: './recipes/recipes.module#RecipesModule', canLoad: [AuthGuard] }

In this example, the AuthGuard  should implement the [CanLoad interface](https://angular.io/docs/ts/latest/api/router/index/CanLoad-interface.html).

**Section 21: Lecture 282//How the modules and services Together**

1. Here we will find out how modules and services are connected and how they are connected behind the scene.
2. 
3. 
4. 
5. 
6. 
7. 
8. 

**Section 21: Lecture 283//Understanding the Core Module**

1. 

**Section 21: Lecture 284//Creating a Basic Core Module**

1. We will use the core module to put om all the providers in there, unlike the shared module the core module will only be imported by the root module.
2. So we cannot have the providers array in the code module, because we will not have the correct behavior.
3. So, let’s add a new folder in our app that is the core folder. Now, first of all we will move the header in the core module
4. core.module.ts
5. import { NgModule } from "@angular/core";
6. import { HeaderComponent } from "./header/header.component";
7. import { HomeComponent } from "./home/home.component";
8. import { SharedModule } from "../shared/shared.module";
9. import { AppRoutingModule } from "../app-routing.module";
10. @NgModule({
11. declarations: [
12. HeaderComponent,
13. HomeComponent
14. ],
15. imports: [
16. SharedModule,
17. AppRoutingModule
18. ],
19. exports: [
20. AppRoutingModule,
21. HeaderComponent
22. ]
23. })
24. export class CoreModule{
25. }

5. app.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';

import { ShoppingListService } from './shopping-list/shopping-list.service';

import { AppRoutingModule } from './app-routing.module';

import { RecipeService } from './recipes/recipe.service';

import { DataStorageService } from './shared/data-storage.service';

import { AuthService } from './auth/auth.service';

import { AuthGuard } from './auth/auth-guard.service';

import { SharedModule } from './shared/shared.module';

import { ShoppoingListModule } from './shopping-list/shopping-list.module';

import { AuthModule } from './auth/auth.module';

import { CoreModule } from './core/core.module';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

HttpModule,

AppRoutingModule,

SharedModule,

ShoppoingListModule,

AuthModule,

CoreModule

],

providers: [ShoppingListService, RecipeService, DataStorageService, AuthService, AuthGuard],

bootstrap: [AppComponent]

})

export class AppModule { }

**Section 21: Lecture 285//Restructuring Services to use the Child Injector**

1. We have already discussed the we don’t need to put the providers in the shared module i.e. the services; however, we can still put them in the multiple modules.
2. The behavior of the providers can change when they are used with the lazy loaded.
3. We will remove the providers from the app module and we will move them in the core module.
4. Only guards should be provided in the routing modules and no other provider should be added.
5. core.module.ts:
6. import { NgModule } from "@angular/core";
7. import { HeaderComponent } from "./header/header.component";
8. import { HomeComponent } from "./home/home.component";
9. import { SharedModule } from "../shared/shared.module";
10. import { AppRoutingModule } from "../app-routing.module";
11. import { ShoppingListService } from "../shopping-list/shopping-list.service";
12. import { RecipeService } from "../recipes/recipe.service";
13. import { DataStorageService } from "../shared/data-storage.service";
14. import { AuthService } from "../auth/auth.service";
15. import { AuthGuard } from "../auth/auth-guard.service";
16. @NgModule({
17. declarations: [
18. HeaderComponent,
19. HomeComponent
20. ],
21. imports: [
22. SharedModule,
23. AppRoutingModule
24. ],
25. exports: [
26. AppRoutingModule,
27. HeaderComponent
28. ],
29. providers: [ShoppingListService, RecipeService, DataStorageService, AuthService, AuthGuard]
30. })
31. export class CoreModule{
32. }

6. app.module.ts:

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';

import { AppRoutingModule } from './app-routing.module';

import { SharedModule } from './shared/shared.module';

import { ShoppoingListModule } from './shopping-list/shopping-list.module';

import { AuthModule } from './auth/auth.module';

import { CoreModule } from './core/core.module';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

HttpModule,

AppRoutingModule,

SharedModule,

ShoppoingListModule,

AuthModule,

CoreModule

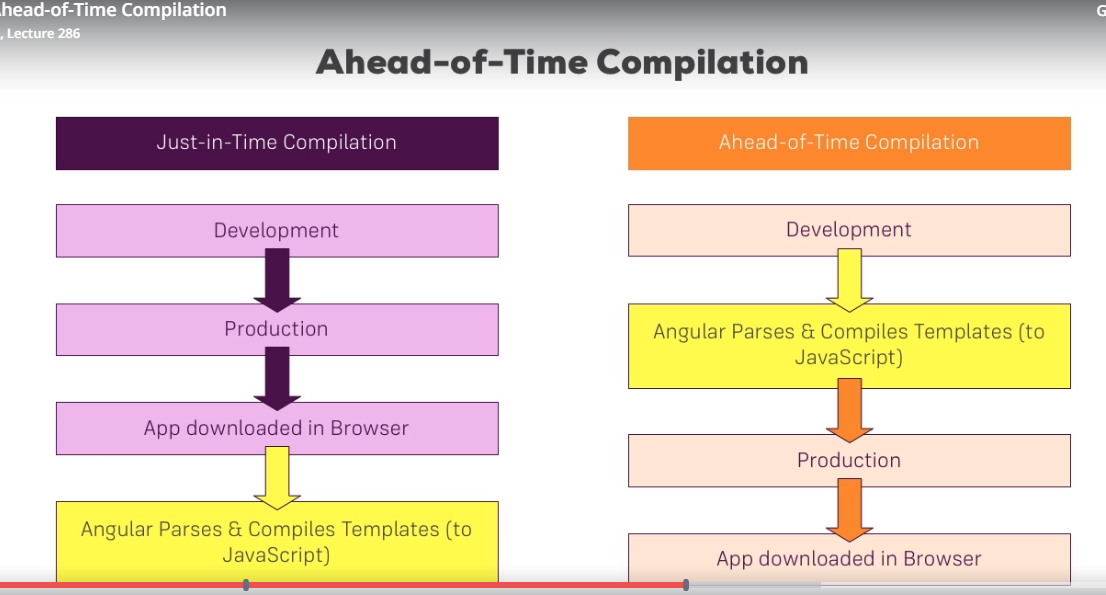
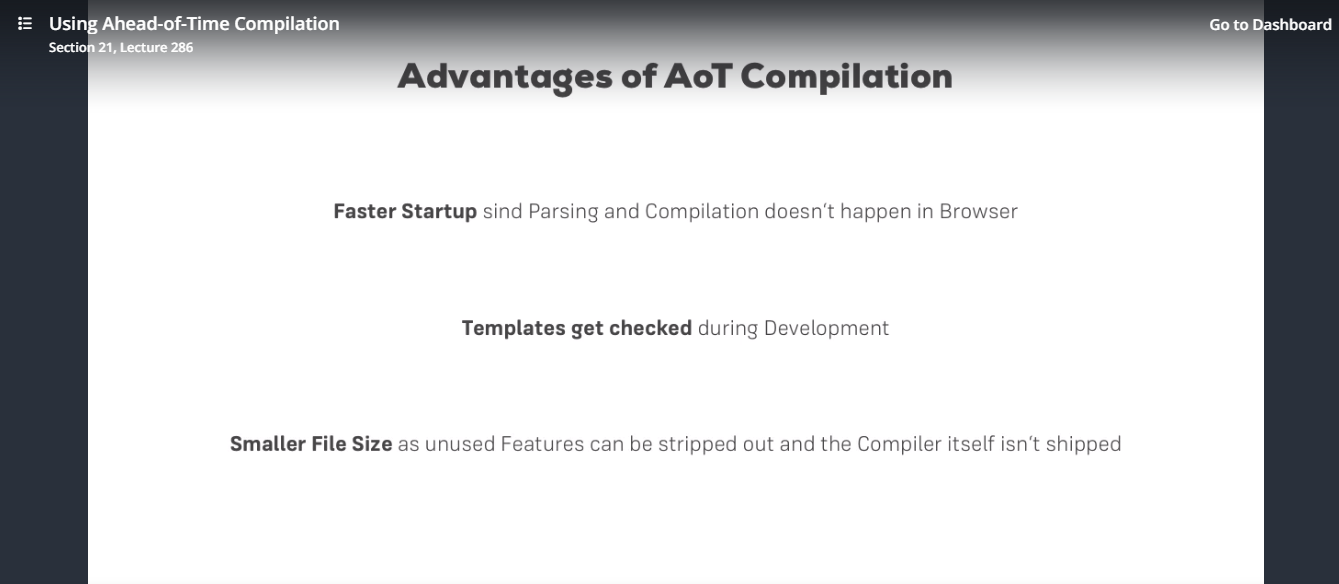
],

bootstrap: [AppComponent]

})

export class AppModule { }

**Section 21: Lecture 286//Using Ahead of time compilation**

1. Angular compiles your HTML templates also; angular compiles this HTML code into JavaScript.
2. We cannot represent our HTML code in JavaScript without angular 2.
3. Accessing JavaScript is faster than DOM. So that’s the main reason i.e. performance improvement.
4. So, JavaScript can do this compilation in 2 different places i.e. Just-In-Time compilation and Ahead-of-time compilation.
5. JIT is default and we used it throughout the course
6. 
7. 

**Section 21: Lecture 287//How to use AoT Compilation with the CLI**

1. ng build - -prod - -aot

**Section 22: Lecture 288//Preloading Lazy Loaded Routes**

1. app-routing.module.ts:
2. import { NgModule } from '@angular/core';
3. import { Routes, RouterModule, PreloadAllModules } from '@angular/router';
4. import { ShoppingListComponent } from './shopping-list/shopping-list.component';
5. import { HomeComponent } from './core/home/home.component';
6. const appRoutes: Routes = [
7. { path: '', component: HomeComponent},
8. { path: 'recipes', loadChildren: './recipes/recipes.module#RecipesModule'},
9. { path: 'shopping-list', component: ShoppingListComponent }
10. ];
11. @NgModule({
12. imports: [
13. RouterModule.forRoot(appRoutes, {preloadingStrategy: PreloadAllModules})
14. ],
15. exports: [RouterModule]
16. })
17. export class AppRoutingModule {
18. }

**Section 21: Lecture 289//Wrap Up**

=====================END OF SECTION==========================================