REVIEW MEETING 3

09/02/2021 TO 15/02/2021

CHAPTER 6 : ROUTING AND NAVIGATING PAGES

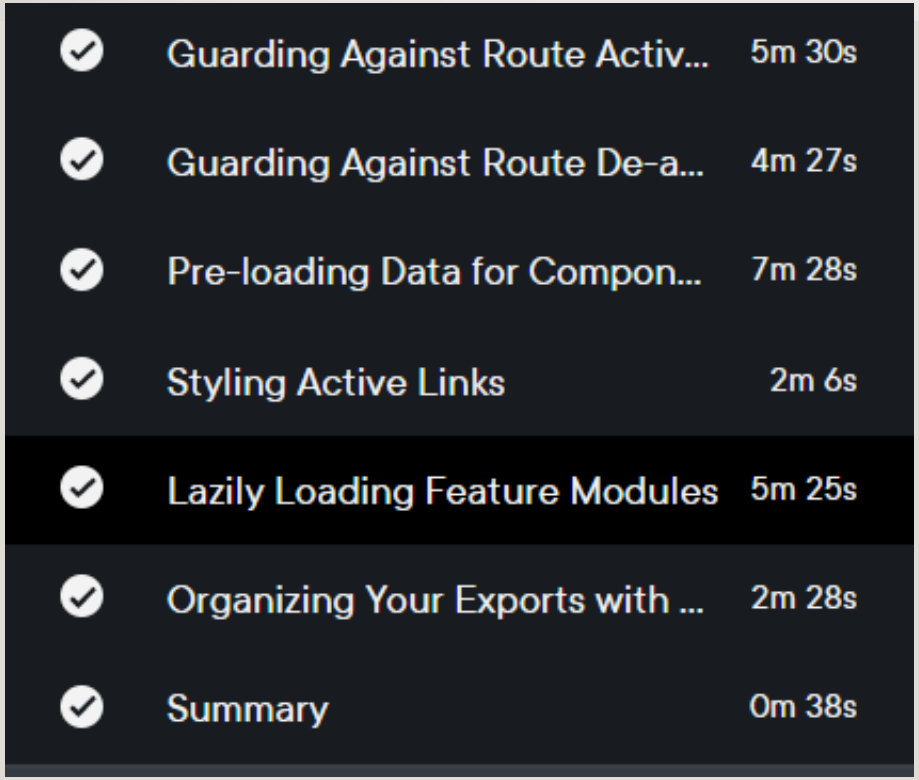
```
export const appRoutes:Routes =
    {path: 'events/new', compone                    vate: ['ca
    {path: 'events', component: EventsListComponent, resolve : {events: Event
    {path: 'events/:id', component: EventDetailsComponent, canActivate: [Ever
    {path: 'events/session/new', component: CreateSessionComponent},
    {path: '', redirectTo: '/events', pathMatch: 'full'},
    // event path set is referenced in other files using the router module or
    {path: '404', component:Error404Component},
    {path: 'user', loadChildren: './user/user.module#UserModule'}
]
```

```
(alias) class EventsListComponent
import EventsListComponent
```

```
@NgModule({
  imports: [
    BrowserModule,
    RouterModule.forRoot(appRoutes),
    FormsModule,
    ReactiveFormsModule
  ],
```

```
template: `
  <nav-bar></nav-bar>
  <router-outlet></router-outlet>
`
```

```
</li>
<li><a [routerLink] = "['/events/new']" routerLinkActive = "active" >Create Event</a></li>
<li class="dropdown">
  <a href="#" class="dropdown-toggle" data-toggle="dropdown" >
    Events
```


Guarding Against Route Activ... 5m 30s
Guarding Against Route De-a... 4m 27s
Pre-loading Data for Compon... 7m 28s
Styling Active Links 2m 6s
Lazily Loading Feature Modules 5m 25s
Organizing Your Exports with ... 2m 28s
Summary 0m 38s

CH 7 : COLLECTING DATA WITH FORMS AND VALIDATION

Template based forms

Reactive forms

Template Driven Forms Features

- Easy to use
- Suitable for simple scenarios and fails for complex scenarios
- Similar to AngularJS
- Two way data binding(using `[(NgModel)]` syntax)
- Minimal component code
- Automatic track of the form and its data(handled by Angular)
- Unit testing is another challenge

Reactive Forms Features

- More flexible, but needs a lot of practice
- Handles any complex scenarios
- No data binding is done (immutable data model preferred by most developers)
- More component code and less HTML markup
- Reactive transformations can be made possible such as
  - Handling a event based on a debounce time
  - Handling events when the components are distinct until changed
  - Adding elements dynamically
- Easier unit testing

TEMPLATE FORMS

- Template form : create form directly in html template, but complex tasks such as cross field validation is hard. Also cant unit test validation logic using template forms
- 1. create login.comp , html template for it
- 2. add routesto user.route.ts and imports to user.module
- 3. Import FormsModule to user.module.ts : gives access to different template- based forms features
- 4. Using [(ngModel)] in login.comp.ts for two way binding of data
- [] : component to html direction ; () : html to component direction
- 5. The data entered in the form are binding to ngForm. //localForm is a local variable here
- 6. Using (ngSubmit) : when you submit the form the login method on our component would be called

CH : 8

Passing data to child component and back using @input and @output and Event Emitter

Show in GitHub

CH 9: CONTENT PROJECTION

- Use of collapsible event details to show optimize User Interface
- Learnt about class and attribute selectors to view desired data when required

CH:10 PIPES

- We use pipes to transform strings, currency amounts, dates, and other data for display. Pipes are simple functions you can use in template expressions to accept an input value and return a transformed value. Pipes are useful because you can use them throughout your application, while only declaring each pipe once. For example, you would use a pipe to show a date as April 15, 1988 rather than the raw string format.
- Created custom pipes
- Learnt how to filter and sort out event data

CH : 11 DEPENDENCY INJECTION

- Used dependency injection to shorten code for toastr module and inject its use into other components without having the wrap the working of it in a separate class.