angular

# Angular 8 (formerly Angular 2) - The Complete Guide

## 1. Getting Started:

In the next lecture, we're going to build our first little app!

If the CLI prompts you to**answer some questions**(some versions do that), you can simply hit **ENTER** for all questions. This will accept the default settings which are fine for this course.

The CLI generates a different welcome screen than you're going to see in my video though. No worries, you'll still be able to follow along without issues! Just make sure to code along **so that your code equals mine** - Angular itself didn't change a bit :)

Depending on the CLI version you're using, you might also need to add the FormsModule  to the imports[]  array in your app.module.ts  file (add it if you don't see it there). You might not fully understand what that all means but we're going to cover that in this course, no worries.

If you don't have FormsModule  in imports[]  in AppModule , please do add it and also add an import at the top of that file: import { FormsModule } from '@angular/forms';

-----

If you want to **dive deeper into the CLI** and learn more about its usage, have a look at its official **documentation**: <https://github.com/angular/angular-cli/wiki>

**You encountered issues during the installation of the CLI or setup of a new Angular project?**

A lot of problems are solved by making sure you're using the latest version of NodeJS, npm and the CLI itself.

**Updating NodeJS:**

Go to nodejs.org and download the latest version - uninstall (all) installed versions on your machine first.

**Updating npm:**

Run [sudo] npm install -g npm  (sudo  is only required on Mac/ Linux)

**Updating the CLI**

[sudo] npm uninstall -g angular-cli @angular/cli

npm cache clean

[sudo] npm install -g @angular/cli

**Here are some common issues & solutions:**

1. **Creation of a new project takes forever (longer than 3 minutes)**  
   That happens on Windows from time to time => Try running the command line as administrator
2. **You get an EADDR error (Address already in use)**  
   You might already have another ng serve process running - make sure to quit that or use ng serve --port ANOTHERPORT  to serve your project on a new port
3. **My changes are not reflected in the browser (App is not compiling)**  
   Check if the window running ng serve  displays an error. If that's not the case, make sure you're using the latest CLI version and try restarting your CLI

6. Editing the First App

Nếu khi cài có câu hỏi gì thì chọn rồi ấn enter

Angular cho phép bạn trộn HTML tĩnh và dynamic html

Data binding nghĩa là server sẽ tính toán và trả về => render HTML

<input type="text" [(ngModel)]="name">

ngModel sẽ nói angular lắng nghe bất cứ thay đổi khi bạn nhập vào và lưu trữ vào thuộc tính name

Import để sử dụng ngModule:

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

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

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

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

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

FormsModule

],

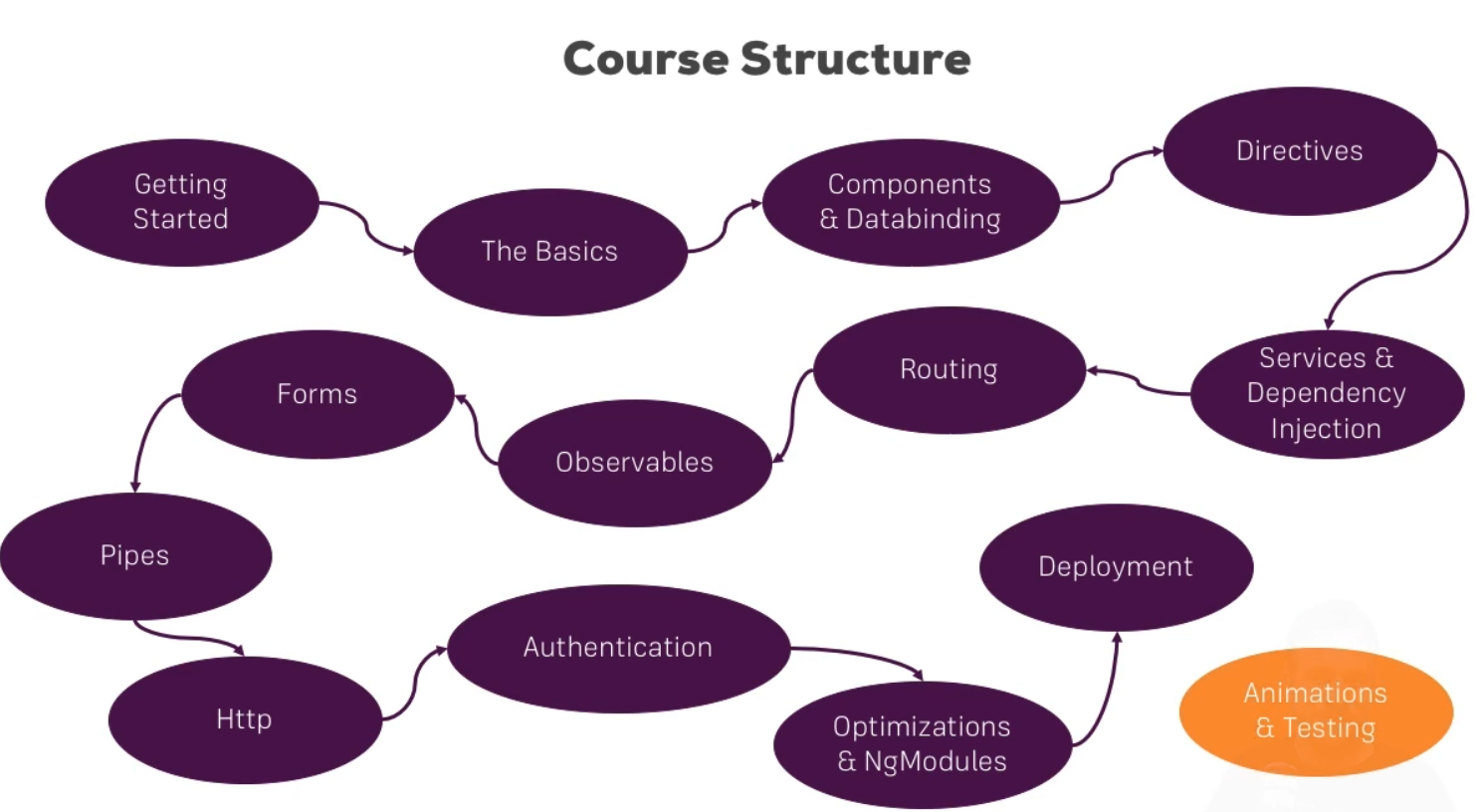
providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

* Tạo ra ứng dụng tự động thay đổi giá trị hiển thị ra khi nhập vào input



9. What is TypeScript

Typescripts nhiều tính năng hơn js: classes, interface,…

Bạn định nghĩa kiểu dữ liệu rõ ràng và không giống như js, được biên dịch sang js

10. A Basic Project Setup using Bootstrap for Styling

Npm install –save bootstrap@3

=> install locally

Tại file **angular.json** sửa lại :

"styles": [

"node\_modules/bootstrap/dist/css/bootstrap.min.css",

"src/styles.css"

],

In the next lecture, we set up the course project. For that, we'll install the Bootstrap CSS Framework.

In this course, we use version 3 of the framework, install it via npm install --save bootstrap@3  => The @3  is important!

Additionally, when using a project created with Angular CLI 6+ (check via ng -v ), you'll have an angular.json  file instead of an .angular-cli.json  file. In that file, you still need to add Bootstrap to the styles[]  array as shown in the next video, but the path should be node\_modules/bootstrap/dist/css/bootstrap.min.css , **NOT** ../node\_modules/bootstrap/dist/css/bootstrap.min.css . The leading ../  must not be included.

Also see this lecture - I do show the complete setup process there: <https://www.udemy.com/the-complete-guide-to-angular-2/learn/v4/t/lecture/6655614/>

## 2. The Basics

Tại file main.ts

platformBrowserDynamic().bootstrapModule(AppModule)

// khai báo module

**Angular is a js framework, change your DOM(‘html’) at runtime**

**Component**: divide your application into many components => chia thành các phần có thể tái SD

4. Creating a New Component

Component đơn giản là 1 Typescripts class

Tạo folder server sau đó vào module khai báo

import { Component } from '@angular/core'; // chỉ rõ cái cần import

@Component({

selector: 'server-root',

templateUrl: './ server.component.html',

styleUrls: ['./ server.component.css']

})

export class ServerComponent {

}

7. Creating Components with the CLI & Nesting Components

Ng generate component servers

Ng g c servers

Ng g c servers –-spec false

=> tạo ra k có file test

Ng g c servers/child –-spec false

=> tạo ở trong folder servers

Nếu định nghĩa template trong 1 file thì sử dụng dấu `` nếu viết trên nhiều dòng:

@Component({

selector: 'server-root',

// selector: '[server-root]', là 1 thuộc tính

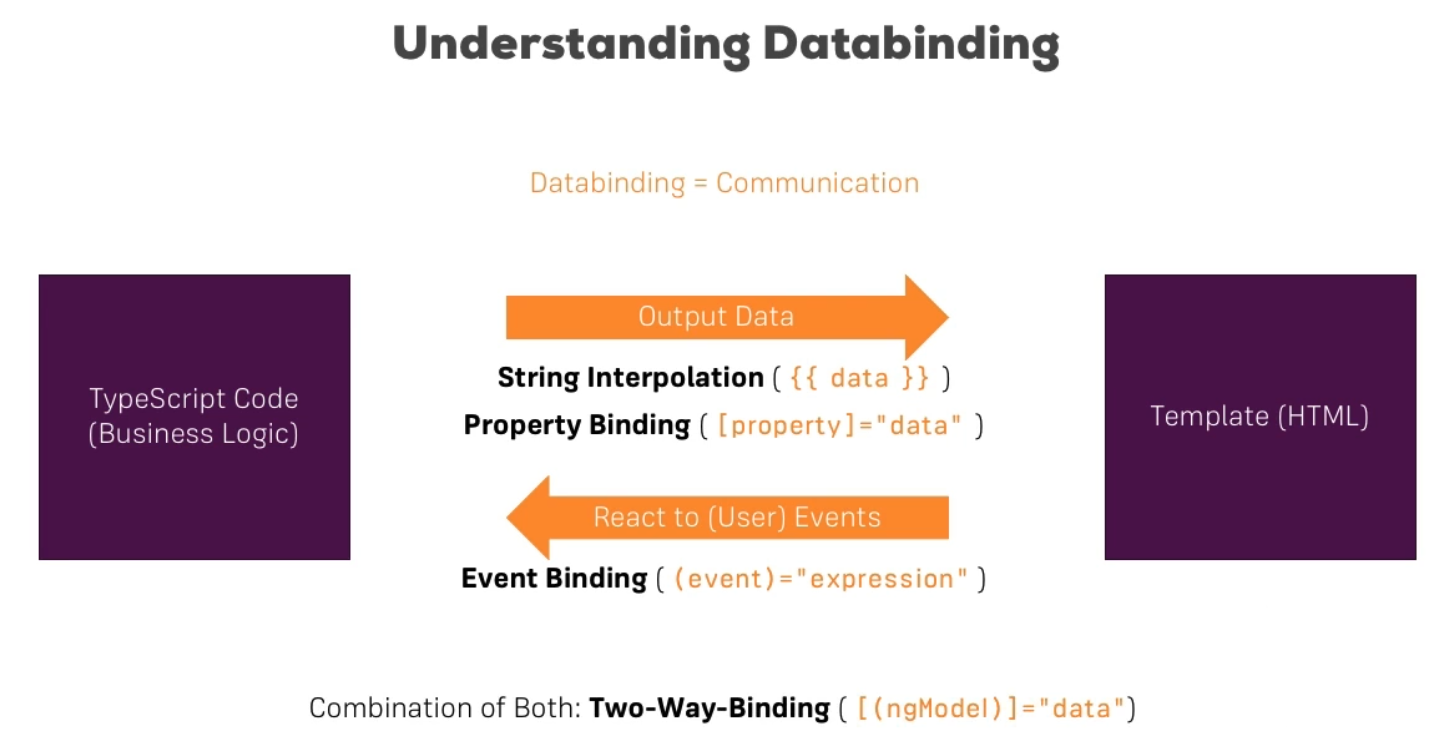
// selector: '.server-root', là 1 class

template: `code in here…`,

styles: [`code in here…`]

})

13. What is Databinding



14. String Interpolation

Có thể khai báo thêm kiểu dữ liệu cho biến trong typescripts

serverId: number = 10;

serverStatus: string = 'offline';

Bien:number = 10;

Output property by use {{ }}: những gì viết trong dấu {{ }} cuối cùng cũng return về string và nó chỉ được viết trên 1 dòng

Có thể gọi hàm getAnything() trong dấu trên cũng được

<p>{{ 'Server' }} with ID {{ serverId }} is {{ getServerStatus()}}</p>

15. Property Binding

<button

class="btn btn-primary"

[disabled]="!allowNewServer"

(click)="onCreateServer()">Add Server</button>

Dùng dấu [] để biểu diễn cho property binding

*<!--<p [innerText]="allowNewServer"></p>-->*

<p>{{ serverCreationStatus }}</p> => trả về string

17. Event Binding

Sử dụng dấu ()

19. Passing and Using Data with Event Binding

<label>Server Name</label>

*<!--<input-->*

*<!--type="text"-->*

*<!--class="form-control"-->*

*<!--(input)="onUpdateServerName($event)">-->*

*<!--<p>{{ serverName }}</p>-->*

$event đơn giản là data được emitted with event

Ở file .ts

onUpdateServerName(event: Event) {

console.log(event);

this.serverName = (<HTMLInputElement>event.target).value;

}

Important: For Two-Way-Binding (covered in the next lecture) to work, you need to enable the ngModel  directive. This is done by adding the FormsModule  to the imports[]  array in the AppModule.

You then also need to add the import from @angular/forms  in the app.module.ts file:

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

21. Two-Way-Databinding

<input

type="text"

class="form-control"

[(ngModel)]="serverName">

Nếu thay đổi trên server name thì nó sẽ được cập nhật lại trên file html, còn one-way data binding sẽ update 1 chiều thôi

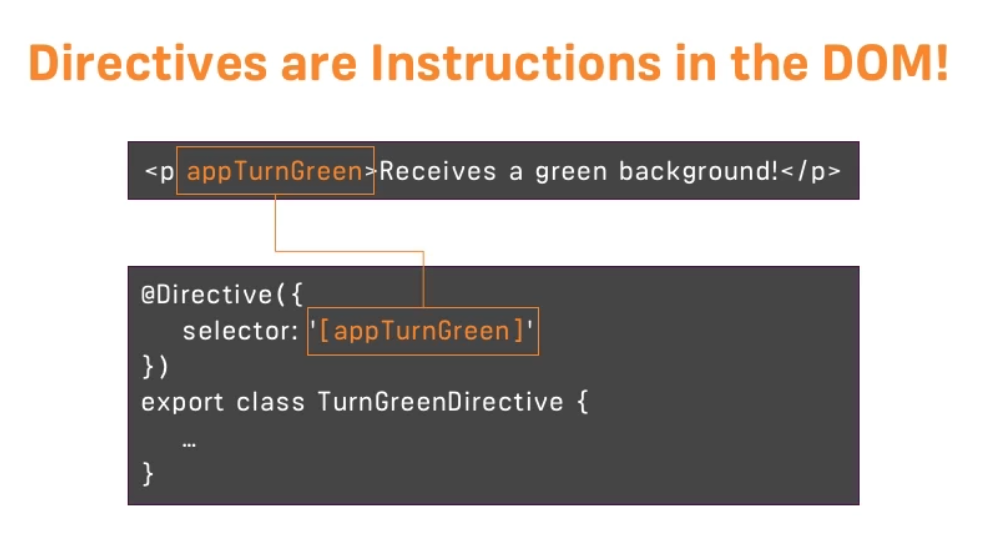
Sửa lại để khi bấm button thì add new

onCreateServer() {

this.serverCreationStatus = 'Server was created! Name is ' + this.serverName;

}

25. Understanding Directives



26. Using ngIf to Output Data Conditionally

Dấu \* là 1 kí tự bắt buộc vì nó là cấu trúc của directive nghĩa là nó sẽ thay đổi cấu trúc của DOM

Điều kiện đặt trong dấu “” có thể là hàm trả về T or F

<p \*ngIf="!serverCreated; else noServer"> Server is created. Server name is {{serverName}}</p>

<ng-template #noServer>

<p>No Server is created.</p>

</ng-template>

28. Styling Elements Dynamically with ngStyle

Không như structural directive, attribute directive không thêm hay xóa element mà chỉ thay đổi element

33.2 basics-final.zip

Tại folder server: thay đổi để server online có màu khác offline

Tại file server.component.ts

constructor() {

this.serverStatus = Math.random() > 0.5 ? 'online' : 'offline';

}

getServerStatus() {

return this.serverStatus;

}

getColor() {

return this.serverStatus === 'online' ? 'green' : 'red';

}

Tại file html

<p

[ngStyle]="{backgroundColor: getColor()}">

{{ 'Server' }} with ID {{ serverId }} is {{ getServerStatus() }}

</p>

29. Applying CSS Classes Dynamically with ngClass

File server.ts

styles: [`

.online {

color: white;

}

`]

File html

<p

[ngStyle]="{backgroundColor: getColor()}"

[ngClass]="{online: serverStatus === 'online'}">

{{ 'Server' }} with ID {{ serverId }} is {{ getServerStatus() }}

</p>

30. Outputting Lists with ngFor

<app-server \*ngFor="let server of servers"></app-server>

Bài tập: 33.1 basics-ngfor-addition.zip

*<!--<button class="btn btn-primary" (click)="showSecret = !showSecret">Display Details</button>-->*

<button class="btn btn-primary" (click)="onToggleDetails()">Display Details</button>

<p \*ngIf="showSecret">Secret Password = tuna</p>

<div

\*ngFor="let logItem of log; let i = index"

[ngStyle]="{backgroundColor: i >= 4 ? 'blue' : 'transparent'}"

[ngClass]="{'white-text': i >= 4}"

>{{ logItem }}</div>

let i = index: truy cập chỉ số theo vòng lặp

## 3. Course Project - The Basics

In the next lecture, we set up the course project. For that, we'll install the Bootstrap CSS Framework.

In this course, we use version 3 of the framework, install it via npm install --save bootstrap@3  => The @3  is important!

Additionally, when using a project created with Angular CLI 6+ (check via ng -v ), you'll have an angular.json  file instead of an .angular-cli.json  file. In that file, you still need to add Bootstrap to the styles[]  array as shown in the next video, but the path should be node\_modules/bootstrap/dist/css/bootstrap.min.css , **NOT** ../node\_modules/bootstrap/dist/css/bootstrap.min.css . The leading ../  must not be included.

Also see this lecture - I do show the complete setup process there: <https://www.udemy.com/the-complete-guide-to-angular-2/learn/v4/t/lecture/6655614/>

Tại file app.component.html:

<app-header></app-header>

<div class="container">

<div class="row">

<div class="col-md-12">

<app-recipes></app-recipes>

<app-shopping-list></app-shopping-list>

</div>

</div>

</div>

Tại file recipes.component.html;

<div class="row">

<div class="col-md-5">

<app-recipe-list></app-recipe-list>

</div>

<div class="col-md-7">

<app-recipe-detail></app-recipe-detail>

</div>

</div>

Tại file recipe-list.component.html :

<div class="row">

<div class="col-xs-12">

<button class="btn btn-success">New Recipe</button>

</div>

</div>

<hr>

<div class="row">

<div class="col-xs-12">

<app-recipe-item></app-recipe-item>

</div>

</div>

Tại file shopping-list.component.html

<div class="row">

<div class="col-xs-10">

<app-shopping-edit></app-shopping-edit>

<hr>

<ul class="list-group">

</ul>

</div>

</div>

7. Adding a Navigation Bar

Header

<nav class="navbar navbar-default">

<div class="container-fluid">

<div class="navbar-header">

<a href="#" class="navbar-brand">Recipe Book</a>

</div>

<div class="collapse navbar-collapse">

<ul class="nav navbar-nav">

<li><a href="#">Recipes</a></li>

<li><a href="#">Shopping List</a></li>

</ul>

<ul class="nav navbar-nav navbar-right">

<li class="dropdown">

<a href="#" class="dropdown-toggle" role="button">Manage <span class="caret"></span></a>

<ul class="dropdown-menu">

<li><a href="#">Save Data</a></li>

<li><a href="#">Fetch Data</a></li>

</ul>

</li>

</ul>

</div>

</div>

</nav>

The way we added it, the Navbar will collapse on smaller screens. Since we didn't implement a Hamburger menu, that means that there's no way of accessing our links on smaller screens.

You can either add such a menu on your own (see below), or you replace collapse navbar-collapse  with just navbar-default.

Adding a Hamburger Menu:

Alternatively, if you want to make the navigation bar responsive, please replace these lines in header.component.html:

<div class="navbar-header">

<a routerLink="/" class="navbar-brand">Recipe Book</a>

</div>

<div class="collapse navbar-collapse">

with these lines:

<div class="navbar-header">

<button type="button" class="navbar-toggle" (click)="collapsed = !collapsed">

<span class="icon-bar" \*ngFor="let iconBar of [1, 2, 3]"></span>

</button>

<a routerLink="/" class="navbar-brand">Recipe Book</a>

</div>

<div class="navbar-collapse" [class.collapse]="collapsed" (window:resize)="collapsed = true">

and add this line to header.component.ts:

collapsed = true;

9. Creating a Recipe Model

Tại file recipe.model.ts

export class Recipe {

public name: string;

public description: string;

public imagePath: string;

constructor(name: string, desc: string, imagePath: string) {

this.name = name;

this.description = desc;

this.imagePath = imagePath;

}

}

11. Outputting a List of Recipes with ngFor

File recipe-list.component.ts

export class RecipeListComponent implements OnInit {

recipes: Recipe[] = [

new Recipe('A Test Recipe', 'This is simply a test', 'https://upload.wikimedia.org/wikipedia/commons/1/15/Recipe\_logo.jpeg'),

new Recipe('A Test Recipe', 'This is simply a test', 'https://upload.wikimedia.org/wikipedia/commons/1/15/Recipe\_logo.jpeg')

];

constructor() { }

ngOnInit() {

}

}

Tại file recipe-list.component.html sửa lại để đổ data:

<div class="row">

<div class="col-xs-12">

<button class="btn btn-success">New Recipe</button>

</div>

</div>

<hr>

<div class="row">

<div class="col-xs-12">

<a

href="#"

class="list-group-item clearfix"

\*ngFor="let recipe of recipes">

<div class="pull-left">

<h4 class="list-group-item-heading">{{ recipe.name }}</h4>

<p class="list-group-item-text">{{ recipe.description }}</p>

</div>

<span class="pull-right">

<img

[src]="recipe.imagePath"

alt="{{ recipe.name }}"

class="img-responsive"

style="max-height: 50px;">

</span>

</a>

<app-recipe-item></app-recipe-item>

</div>

</div>

12. Displaying Recipe Details

recipe-detail.component.html

<div class="row">

<div class="col-xs-12">

<img src="" alt="" class="img-responsive">

</div>

</div>

<div class="row">

<div class="col-xs-12">

<h1>Recipe Name</h1>

</div>

</div>

<div class="row">

<div class="col-xs-12">

<div class="btn-group">

<button

type="button"

class="btn btn-primary dropdown-toggle">

Manage Recipe <span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li><a href="#">To Shopping List</a></li>

<li><a href="#">Edit Recipe</a></li>

<li><a href="#">Delete Recipe</a></li>

</ul>

</div>

</div>

</div>

<div class="row">

<div class="col-xs-12">

Description

</div>

</div>

<div class="row">

<div class="col-xs-12">

Ingredients

</div>

</div>

13. Working on the ShoppingListComponent

14. Creating an Ingredient Model

ingredient.model.ts

export class Ingredient {

constructor(public name: string, public amount: number) {}

}

Thêm public vào nó sẽ tự động gán

Thêm khai báo class .ts và shopping-list.component.html

<div class="row">

<div class="col-xs-10">

<app-shopping-edit></app-shopping-edit>

<hr>

<ul class="list-group">

<a

class="list-group-item"

style="cursor: pointer"

\*ngFor="let ingredient of ingredients"

>

{{ ingredient.name }} ({{ ingredient.amount }})

</a>

</ul>

</div>

</div>

16. Adding a Shopping List Edit Section

shopping-edit.component.html

<div class="row">

<div class="col-xs-12">

<form>

<div class="row">

<div class="col-sm-5 form-group">

<label for="name">Name</label>

<input type="text" id="name" class="form-control">

</div>

<div class="col-sm-2 form-group">

<label for="amount">Amount</label>

<input type="number" id="amount" class="form-control">

</div>

</div>

<div class="row">

<div class="col-xs-12">

<button class="btn btn-success" type="submit">Add</button>

<button class="btn btn-danger" type="button">Delete</button>

<button class="btn btn-primary" type="button">Clear</button>

</div>

</div>

</form>

</div>

</div>

## 4. Debugging

Augury tool

## 5. Components & Databinding Deep Dive

2. Splitting Apps into Components

Ng g c cockpit –-spec false

Ng g c server-element –-spec false

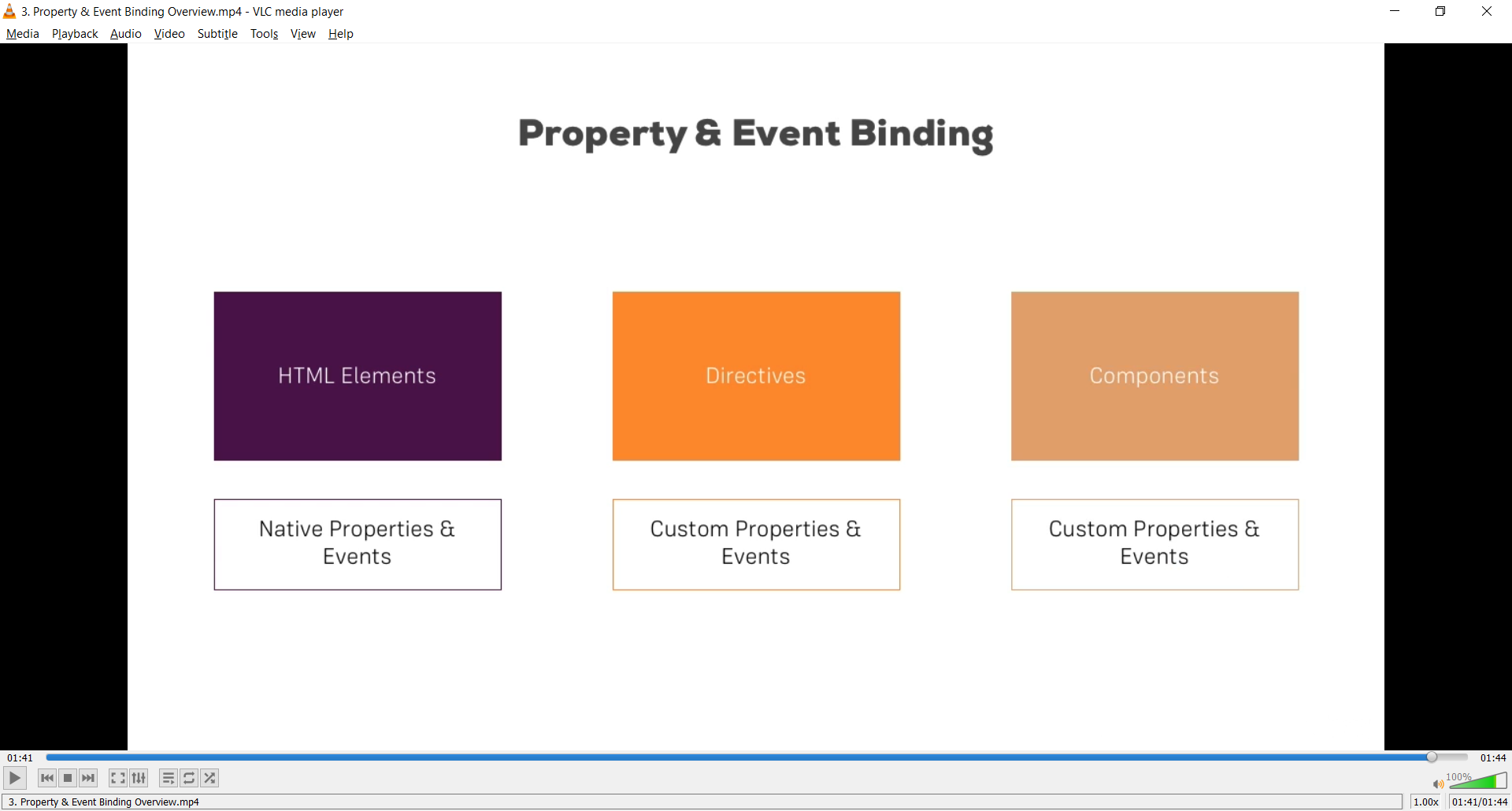
App tạo server và blue print server(khi hiển thị sẽ in nghiêng và có màu khác): get server được tạo ra bởi cockpit và thêm vào list server ở app component

3. Property & Event Binding Overview

Ta cần gửi và nhận data từ component

Có thể sử dụng event binding không những ở các phần tử HTML mà còn những thuộc tính bên trong và event

Có thể sử dụng Directive với custom prop với ngClass, ngStyle



4. Binding to Custom Properties

Tất cả các thuộc tính định nghĩa trong component chỉ được truy cập ở bên trong chúng mà bên ngoài không thể truy cập được

Nếu muốn expose prop cho parent element thì sử dụng **@Input**

@Input('srvElement') element: {type: string, name: string, content: string};

Nếu muốn đặt alias cho element lại thì sử dụng như trên

5. Assigning an Alias to Custom Properties

@Input() name: string;

@Input(‘aliasName’) name: string;

6. Binding to Custom Events

How to inform app-component event emitted from cockpit

File app-component.html

<app-cockpit

(serverCreated)="onServerAdded($event)"

(bpCreated)="onBlueprintAdded($event)"

></app-cockpit>

File app-component.ts

onServerAdded(serverData: {serverName: string, serverContent: string}) {

this.serverElements.push({

type: 'server',

name: serverData.serverName,

content: serverData.serverContent

});

}

onBlueprintAdded(blueprintData: {serverName: string, serverContent: string}) {

this.serverElements.push({

type: 'blueprint',

name: blueprintData.serverName,

content: blueprintData.serverContent

});

}

Tại file cockpit.ts

@Output() serverCreated = new EventEmitter<{serverName: string, serverContent: string}>();

@Output('bpCreated') blueprintCreated = new EventEmitter<{serverName: string, serverContent: string}>();

onAddServer() {

this.serverCreated.emit({

serverName: this. *newServerName*,

serverContent: this.*newServerCOntent*

});

}

onAddBlueprint() {

this.blueprintCreated.emit({

serverName: this. *newServerName*,

serverContent: this.*newServerCOntent*

});

}

7. Assigning an Alias to Custom Events

8. Custom Property and Event Binding Summary

9. Understanding View Encapsulation

Shallow DOM is a technology that is not supported by all browsers where each element has own shallow DOM behind it, where you can assign styles to each element and that is the default behavior of view encapsulation in angular

Sửa file css của server-element

F12 kiểm tra class ngcontent-ui-1,2,3…

10. More on View Encapsulation

File server-element.component.ts

@Component({

selector: 'app-server-element',

templateUrl: './server-element.component.html',

styleUrls: ['./server-element.component.css'],

encapsulation: ViewEncapsulation. Emulated *// None, Native*

})

Emulated là default nên bạn không cần add

None có nghĩ là không sử dụng View encapsulation

Native cũng nhưu emaulated…

11. Using Local References in Templates

cockpit.component.html: có thể sử dụng reference chỉ trong temaplate

<label>Server Name</label>

*<!--<input type="text" class="form-control" [(ngModel)]="newServerName">-->*

<input

type="text"

class="form-control"

#serverNameInput>

<label>Server Content</label>

*<!--<input type="text" class="form-control" [(ngModel)]="newServerContent">-->*

<input

type="text"

class="form-control"

#serverContentInput>

<button

class="btn btn-primary"

(click)="onAddServer(serverNameInput)">Add Server</button>

<button

class="btn btn-primary"

(click)="onAddBlueprint(serverNameInput)">Add Server Blueprint</button>

cockpit.component.ts

truy cập đến value in template

@ViewChild('serverContentInput', { static: false }) serverContentInput: ElementRef;

onAddServer(nameInput: HTMLInputElement) {

this.serverCreated.emit({

serverName: nameInput.value,

serverContent: this.serverContentInput.nativeElement.value

});

}

onAddBlueprint(nameInput: HTMLInputElement) {

this.blueprintCreated.emit({

serverName: nameInput.value,

serverContent: this.serverContentInput.nativeElement.value

});

}

In **Angular 8**, the @ViewChild() syntax which you'll see in the next lecture needs to be changed slightly:

Instead of:

@ViewChild('serverContentInput') serverContentInput: ElementRef;

use

@ViewChild('serverContentInput', {static: true}) serverContentInput: ElementRef;

The same change (add { static: true } as a second argument) needs to be applied to ALL usages of @ViewChild() (and also @ContentChild() which you'll learn about later) IF you plan on accessing the selected element inside of ngOnInit().

If you DON'T access the selected element in ngOnInit (but anywhere else in your component), set static: false instead!

This is a temporary adjustment which will NOT be required anymore once Angular 9 is released!

13. Getting Access to the Template & DOM with @ViewChild

@ViewChild('serverContentInput', { static: true }) serverContentInput: ElementRef;

True vì chúng ta sẽ sử dụng trong ngOnInit

Nên sử dụng string in… hay pro binding thay vì this.serverContentInput.nativeElement.value để gán giá trị output

14. Projecting Content into Components with ng-content

Khi muốn chuyển đoạn html bên dưới từ server-element ra ngoài

<strong \*ngIf="serverElement.type === 'server'" style="color: red">{{ serverElement.content }}</strong>

<em \*ngIf="serverElement.type === 'blueprint'">{{ serverElement.content }}</em>

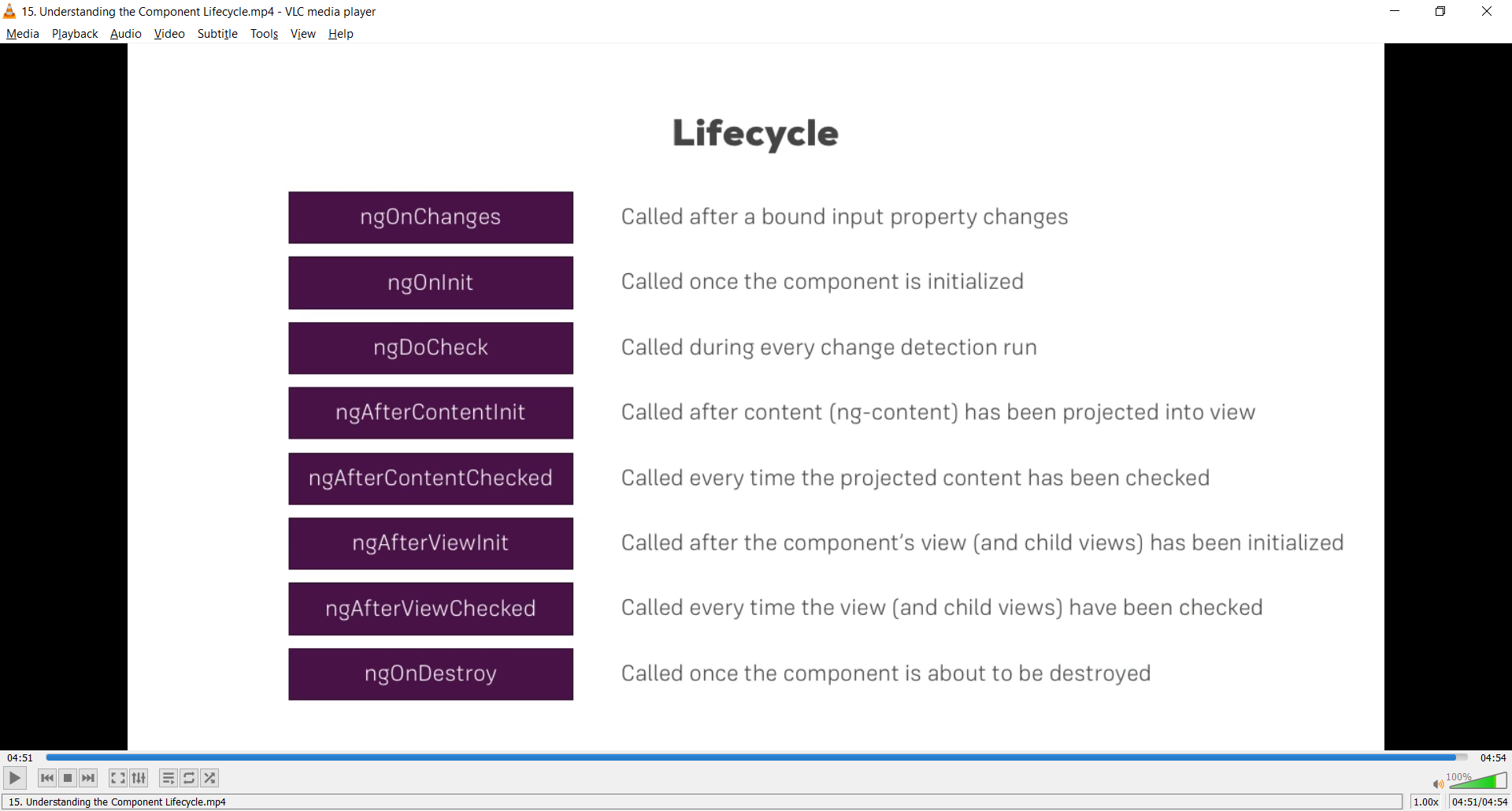
Thay bằng

<ng-content></ng-content>

15. Understanding the Component Lifecycle

ngOnChange với @Input nhận giá trị mới

ngOnInit chạy sau constructor



16. Seeing Lifecycle Hooks in Action

File server-element.component.ts

export class ServerElementComponent implements

OnInit,

OnChanges,

DoCheck,

AfterContentInit,

AfterContentChecked,

AfterViewInit,

AfterViewChecked,

OnDestroy {

constructor() {

console.log('constructor called!');

}

ngOnChanges(changes: SimpleChanges) {

console.log('ngOnChanges called!');

console.log(changes);

}

ngOnInit() {

console.log('ngOnInit called!');

console.log('Text Content: ' + this.header.nativeElement.textContent);

console.log('Text Content of paragraph: ' + this.paragraph.nativeElement.textContent);

}

ngDoCheck() {

console.log('ngDoCheck called!');

}

// chạy trước DoCheck

ngAfterContentInit() {

console.log('ngAfterContentInit called!');

console.log('Text Content of paragraph: ' + this.paragraph.nativeElement.textContent);

}

ngAfterContentChecked() {

console.log('ngAfterContentChecked called!');

}

ngAfterViewInit() {

console.log('ngAfterViewInit called!');

console.log('Text Content: ' + this.header.nativeElement.textContent);

}

ngAfterViewChecked() {

console.log('ngAfterViewChecked called!');

}

ngOnDestroy() {

console.log('ngOnDestroy called!');

}

File app.component.ts

onChangeFirst() {

this.serverElements[0].name = 'Changed!';

}

onDestroyFirst() {

this.serverElements.splice(0, 1);

}

// serverElements sẽ có cùng địa chỉ trong bộ nhớ với mảng ở ngoài nên khi thay đổi thì giá trị sẽ được cập nhật vào mảng

app.component.html thêm thuộc tính name:

<button class="btn btn-primary" (click)="onChangeFirst()">Change first Element</button>

<button class="btn btn-danger" (click)="onDestroyFirst()">Destroy first Component</button>

<app-server-element

\*ngFor="let serverElement of serverElements"

[srvElement]="serverElement"

[name]="serverElement.name">

Server-element.component.ts

*<!--<div class="panel-heading">{{ element.name }}</div>-->*

<div class="panel-heading" #heading>{{ name }}</div>

Server-element.component.ts

@Input() name: string;

17. Lifecycle Hooks and Template Access

server-element.component.html thêm heading

<div class="panel-heading" #heading>{{ name }}</div>

server-element.component.ts

@ViewChild('heading', {static: true}) header: ElementRef;

ngOnInit() {

console.log('ngOnInit called!');

console.log('Text Content: ' + this.header.nativeElement.textContent);

console.log('Text Content of paragraph: ' + this.paragraph.nativeElement.textContent);

}

// lần này mới có giá trị

ngAfterViewInit() {

console.log('ngAfterViewInit called!');

console.log('Text Content: ' + this.header.nativeElement.textContent);

}

In **Angular 8**, the @ContentChild() syntax which you'll see in the next lecture needs to be changed slightly:

Instead of:

@ContentChild('contentParagraph') paragraph: ElementRef;

use

@ContentChild('contentParagraph', {static: true}) paragraph: ElementRef;

The same change (add { static: true } as a second argument) needs to be applied to ALL usages of @ContentChild() if you use the selected element inside of ngOnInit (as we do it in the lectures).

If you DON'T use the selected element in ngOnInit, set static: false instead.

19. Getting Access to ng-content with @ContentChild

app.component.html thêm #contentParagraph

<p #contentParagraph>

<strong \*ngIf="serverElement.type === 'server'" style="color: red">{{ serverElement.content }}</strong>

<em \*ngIf="serverElement.type === 'blueprint'">{{ serverElement.content }}</em>

</p>

@ContentChild('contentParagraph', {static: true}) paragraph: ElementRef;

Server-element.component.ts

ngAfterContentInit() {

console.log('ngAfterContentInit called!');

console.log('Text Content of paragraph: ' + this.paragraph.nativeElement.textContent);

}

20. Wrap Up

22. [OPTIONAL] Assignment Solution

## 6. Course Project - Components & Databinding

App shopping list and recipe hiển thị tùy loại khi bấm vào nav bar

header.component.html bắt event click

<ul class="nav navbar-nav">

        <li><a href="#" (click)="onSelect('recipe')">Recipes</a></li>

        <li><a href="#" (click)="onSelect('shopping-list')">Shopping List</a></li>

      </ul>

header.component.ts

import { Component, EventEmitter, Output } from '@angular/core';

@Component({

  selector: 'app-header',

  templateUrl: './header.component.html'

})

export class HeaderComponent {

  @Output() featureSelected = new EventEmitter<string>();

  onSelect(feature: string) {

    this.featureSelected.emit(feature);

  }

}

app.component.html

<app-header (featureSelected)="onNavigate($event)"></app-header>

app.component.ts

export class AppComponent {

  loadedFeature = 'recipe';

  onNavigate(feature: string) {

    this.loadedFeature = feature;

  }

}

app.component.html load lại tùy kiểu

<div class="col-md-12">

      <app-recipes \*ngIf="loadedFeature === 'recipe'"></app-recipes>

      <app-shopping-list \*ngIf="loadedFeature !== 'recipe'"></app-shopping-list>

    </div>

3. Passing Recipe Data with Property Binding

Copy từ recipe-list sang item

<a

  href="#"

  class="list-group-item clearfix"

  (click)="onSelected()">

  <div class="pull-left">

    <h4 class="list-group-item-heading">{{ recipe.name }}</h4>

    <p class="list-group-item-text">{{ recipe.description }}</p>

  </div>

  <span class="pull-right">

        <img

          [src]="recipe.imagePath"

          alt="{{ recipe.name }}"

          class="img-responsive"

          style="max-height: 50px;">

      </span>

</a>

recipe-list.component.html

<div class="col-xs-12">

    <app-recipe-item

      \*ngFor="let recipeEl of recipes"

      [recipe]="recipeEl"

></app-recipe-item>

  </div>

recipe-item.component.ts

 @Input() recipe: Recipe;

4. Passing Data with Event and Property Binding (Combined)

Khi click vào list thì recipe detail tương ứng sẽ được load

recipe-item.component.ts

  @Output() recipeSelected = new EventEmitter<void>();

  constructor() { }

  ngOnInit() {

  }

  onSelected() {

    this.recipeSelected.emit();

  }

recipe-list.component.html

<div class="col-xs-12">

    <app-recipe-item

      \*ngFor="let recipeEl of recipes"

      [recipe]="recipeEl"

      (recipeSelected)="onRecipeSelected(recipeEl)"></app-recipe-item>

  </div>

recipe-list.component.ts

@Output() recipeWasSelected = new EventEmitter<Recipe>();

  onRecipeSelected(recipe: Recipe) {

   this.recipeWasSelected.emit(recipe);

  }

recipes.component.html

app-recipe-list

      (recipeWasSelected)="selectedRecipe = $event"></app-recipe-list>

recipes.component.ts

export class RecipesComponent implements OnInit {

  selectedRecipe: Recipe;

  constructor() { }

  ngOnInit() {

  }

}

recipes.component.html

<div class="col-md-7">

    <app-recipe-detail

      \*ngIf="selectedRecipe; else infoText"

      [recipe]="selectedRecipe"></app-recipe-detail>

    <ng-template #infoText>

      <p>Please select a Recipe!</p>

    </ng-template>

  </div>

recipe-detail.component.ts

export class RecipeDetailComponent implements OnInit {

  @Input() recipe: Recipe;

  constructor() { }

  ngOnInit() {

  }

}

recipe-detail.component.html

<div class="col-xs-12">

    <img

      [src]="recipe.imagePath"

      alt="{{ recipe.name }}"

      class="img-responsive"

      style="max-height: 300px;">

  </div>

In case you're hitting an error in the next lecture, make sure you have FormsModule added to your imports[] in the AppModule.

@NgModule({

  declarations: [

    AppComponent,

    HeaderComponent,

    RecipesComponent,

    RecipeListComponent,

    RecipeDetailComponent,

    RecipeItemComponent,

    ShoppingListComponent,

    ShoppingEditComponent

  ],

  imports: [

    BrowserModule,

    FormsModule,

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

6. Allowing the User to Add Ingredients to the Shopping List

shopping-edit.component.html thêm local ref

<div class="col-sm-5 form-group">

          <label for="name">Name</label>

          <input

            type="text"

            id="name"

            class="form-control"

            #nameInput>

        </div>

        <div class="col-sm-2 form-group">

          <label for="amount">Amount</label>

          <input

            type="number"

            id="amount"

            class="form-control"

            #amountInput>

        </div>

<button class="btn btn-success" type="submit" (click)="onAddItem()">Add</button>

shopping-edit.component.ts

export class ShoppingEditComponent implements OnInit {

  @ViewChild('nameInput', { static: false }) nameInputRef: ElementRef;

  @ViewChild('amountInput', { static: false }) amountInputRef: ElementRef;

  @Output() ingredientAdded = new EventEmitter<Ingredient>();

  constructor() { }

  ngOnInit() {

  }

  onAddItem() {

    const ingName = this.nameInputRef.nativeElement.value;

    const ingAmount = this.amountInputRef.nativeElement.value;

    const newIngredient = new Ingredient(ingName, ingAmount);

    this.ingredientAdded.emit(newIngredient);

  }

}

shopping-list.component.html

<app-shopping-edit

      (ingredientAdded)="onIngredientAdded($event)"></app-shopping-edit>

    <hr>

shopping-list.component.ts

onIngredientAdded(ingredient: Ingredient) {

    this.ingredients.push(ingredient);

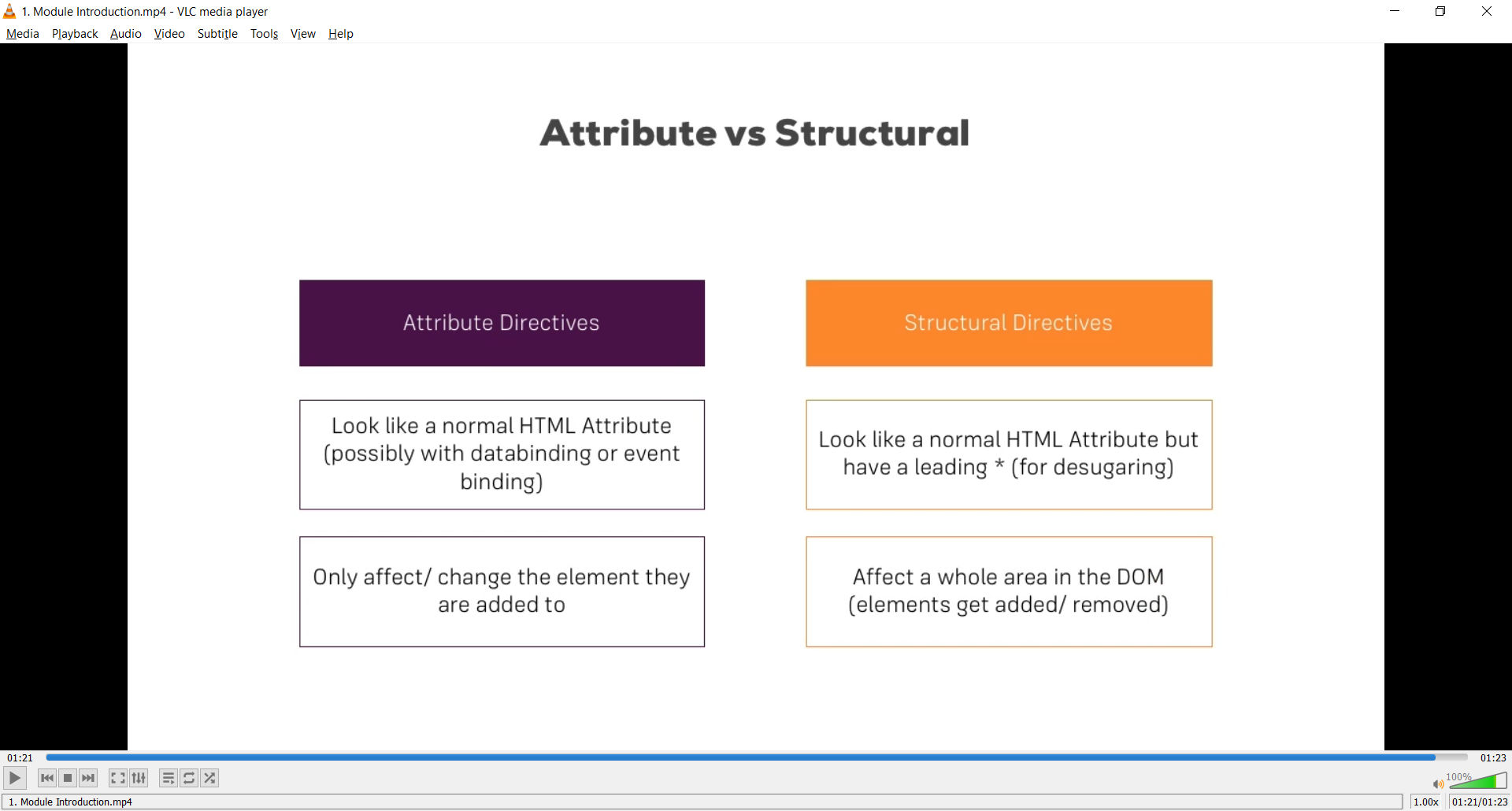
  }

## 7. Directives Deep Dive

2. ngFor and ngIf Recap

Attribute vs structural directive:

* Attribute: directive like attr
* Structural: change structure of the DOM and do the same with attr



Chia làm 2 mảng số chẵn và số lẻ nếu true thì hiển thị mảng chẵn và ngược lại, dùng ngIf để kiểm tra điều kiện

export class AppComponent {

*// numbers = [1, 2, 3, 4, 5];*

  oddNumbers = [1, 3, 5];

  evenNumbers = [2, 4];

  onlyOdd = false;

  value = 5;

}

3. ngClass and ngStyle Recap

File html

*<!--<div \*ngIf="!onlyOdd">-->*

*<!--<li-->*

*<!--class="list-group-item"-->*

*<!--[ngClass]="{odd: even % 2 !== 0}"-->*

*<!--[ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"-->*

*<!--\*ngFor="let even of evenNumbers">-->*

*<!--{{ even }}-->*

*<!--</li>-->*

*<!--</div>-->*

*<!--<ng-template [ngIf]="!onlyOdd">-->*

*<!--<div>-->*

*<!--<li-->*

*<!--class="list-group-item"-->*

*<!--[ngClass]="{odd: even % 2 !== 0}"-->*

*<!--[ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"-->*

*<!--\*ngFor="let even of evenNumbers">-->*

*<!--{{ even }}-->*

*<!--</li>-->*

*<!--</div>-->*

*<!--</ng-template>-->*

4. Creating a Basic Attribute Directive

Tạo folder basic-highlight

basic-highlight.directive.ts

import { Directive, ElementRef, OnInit } from '@angular/core';

@Directive({

  selector: '[appBasicHighlight]' // sau này k cần [] trong element

})

export class BasicHighlightDirective implements OnInit {

  constructor(private elementRef: ElementRef) {

  }

  ngOnInit() {

    this.elementRef.nativeElement.style.backgroundColor = 'green';

  }

}

// constructor(private elementRef: ElementRef) tự động assign value

Module

@NgModule({

  declarations: [

    AppComponent,

    BasicHighlightDirective,

    BetterHighlightDirective,

    UnlessDirective

  ],

  imports: [

    BrowserModule,

    FormsModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

file html

      <p appBasicHighlight>Style me with basic directive!</p>

Nhưng cách trên k tốt

5. Using the Renderer to build a Better Attribute Directive

Ng g d better-highlight

Vào module khai báo BetterHighlightDirective

constructor(private elRef: ElementRef, private renderer: Renderer2){ }

ngOnInit() {

*this.renderer.setStyle(this.elRef.nativeElement, 'background-color', 'blue');*

  }

Sau đó sửa lại html như phần 4

In the last lecture, we used the Angular Renderer class to change the style of a HTML element. As explained in that lecture, you should use the Renderer for any DOM manipulations.

Of course, you can do more than simply change the styling of an element via setStyle(). Learn more about the available Renderer methods [here](https://angular.io/api/core/Renderer2).

<https://angular.io/api/core/Renderer2>

7. Using HostListener to Listen to Host Events

 @HostListener('mouseenter') mouseover(eventData: Event) {

*this.renderer.setStyle(this.elRef.nativeElement, 'background-color', 'blue');*

  }

  @HostListener('mouseleave') mouseleave(eventData: Event) {

*this.renderer.setStyle(this.elRef.nativeElement, 'background-color', 'transparent');*

  }

8. Using HostBinding to Bind to Host Properties

  @HostBinding('style.backgroundColor') backgroundColor: string;

9. Binding to Directive Properties

@Input() defaultColor: string = 'transparent';

  @Input('appBetterHighlight') highlightColor: string = 'blue';

File html

      <p [appBetterHighlight]="'red'" defaultColor="yellow">

10. What Happens behind the Scenes on Structural Directives

Dấu \* đại diện cho structural directive

*<!--<ng-template [ngIf]="!onlyOdd">-->*

Thay cho ghi kiểu \*ngIf

11. Building a Structural Directive

Ng g d unless

import { Directive, Input, TemplateRef, ViewContainerRef } from '@angular/core';

@Directive({

  selector: '[appUnless]'

})

export class UnlessDirective {

  @Input() set appUnless(condition: boolean) {

    if (!condition) {

      this.vcRef.createEmbeddedView(this.templateRef);

    } else {

      this.vcRef.clear();

    }

  }

  constructor(private templateRef: TemplateRef<any>, private vcRef: ViewContainerRef) { }

}

Phương thức setter của prop sẽ execute khi prop thay đổi bất kì

Vào module khai báo UnlessDirective

File html

<div \*appUnless="onlyOdd">

          <li

            class="list-group-item"

            [ngClass]="{ odd: even % 2 !== 0 }"

            [ngStyle]="{

              backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'

            }"

            \*ngFor="let even of evenNumbers"

          >

            {{ even }}

          </li>

        </div>

12. Understanding ngSwitch

<div [ngSwitch]="value">

        <p \*ngSwitchCase="5">Value is 5</p>

        <p \*ngSwitchCase="10">Value is 10</p>

        <p \*ngSwitchCase="100">Value is 100</p>

        <p \*ngSwitchDefault>Value is Default</p>

      </div>

## 8. Course Project - Directives

1. Building and Using a Dropdown Directive

Recipe book app

Vào folder share tạo file dropdown.directive.ts

import { Directive, HostListener, HostBinding } from '@angular/core';

@Directive({

  selector: '[appDropdown]'

})

export class DropdownDirective {

  @HostBinding('class.open') isOpen = false;

  @HostListener('click') toggleOpen() {

    this.isOpen = !this.isOpen;

  }

}

recipe-detail.component.html

<div

      class="btn-group"

      appDropdown>

      <button

        type="button"

        class="btn btn-primary dropdown-toggle">

        Manage Recipe <span class="caret"></span>

      </button>

      <ul class="dropdown-menu">

        <li><a href="#">To Shopping List</a></li>

        <li><a href="#">Edit Recipe</a></li>

        <li><a href="#">Delete Recipe</a></li>

      </ul>

    </div>

Nếu k dùng directive class="btn-group open" nó sẽ có khung bao và hiện lên

declarations: [

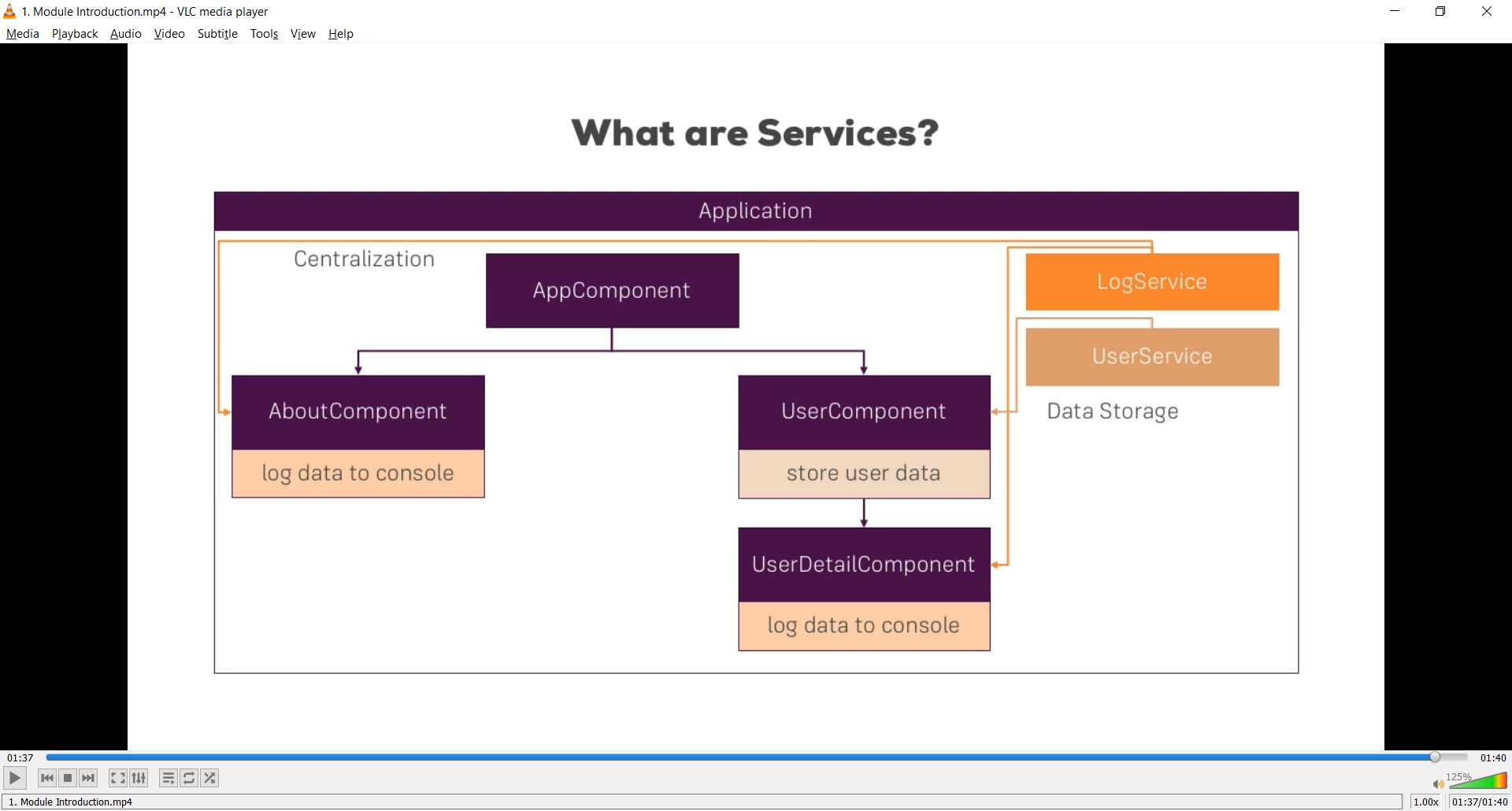
    DropdownDirective

  ],

Vào header thêm để show như recipe-detail

        <li class="dropdown" appDropdown>

## 9. Using Services & Dependency Injection



Khi sử dụng hàm lặp lại hay để storage data, communicate between component => service

App create an acc and change status of acc

3. Creating a Logging Service

Tạo file logging.service.ts

export class LoggingService {

  logStatusChange(status: string) {

    console.log('A server status changed, new status: ' + status);

  }

}

// sau đó import vào tạo instance bằng new sử dụng bt => cách sai

4. Injecting the Logging Service into Components

Inform angular to require an instance => inject an instance to component

  constructor(private loggingService: LoggingService) {}

Khai báo để inform angular how to create

@Component({

  selector: 'app-account',

  templateUrl: './account.component.html',

  styleUrls: ['./account.component.css'],

*// providers: [LoggingService]*

})

Và gọi bằng cách

*this.loggingService.logStatusChange(status);*

5. Creating a Data Service

accounts.service.ts

export class AccountsService {

  accounts = [

    {

      name: 'Master Account',

      status: 'active'

    },

    {

      name: 'Testaccount',

      status: 'inactive'

    },

    {

      name: 'Hidden Account',

      status: 'unknown'

    }

  ];

  constructor(private loggingService: LoggingService) {}

  addAccount(name: string, status: string) {

    this.accounts.push({name: name, status: status});

    this.loggingService.logStatusChange(status);

  }

  updateStatus(id: number, status: string) {

    this.accounts[id].status = status;

    this.loggingService.logStatusChange(status);

  }

}

Xóa event emit đi:

App.component.ts

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent implements OnInit {

  accounts: {name: string, status: string}[] = [];

  ngOnInit() {

    this.accounts = this.accountsService.accounts;

  }

}

app.component.html

<div class="container">

  <div class="row">

    <div class="col-xs-12 col-md-8 col-md-offset-2">

      <app-new-account></app-new-account>

      <hr>

      <app-account

        \*ngFor="let acc of accounts; let i = index"

        [account]="acc"

        [id]="i"></app-account>

    </div>

  </div>

</div>

New-account.component.ts

*providers: [LoggingService],.. thêmy cũ*

export class NewAccountComponent {

  constructor(private loggingService: LoggingService,

              private accountsService: AccountsService) {

    );

  }

  onCreateAccount(accountName: string, accountStatus: string) {

    this.accountsService.addAccount(accountName, accountStatus);

*this.loggingService.logStatusChange(accountStatus);*

  }

}

File account.component.ts

export class AccountComponent {

  @Input() account: {name: string, status: string};

  @Input() id: number;

  constructor(private loggingService: LoggingService,

              private accountsService: AccountsService) {}

  onSetTo(status: string) {

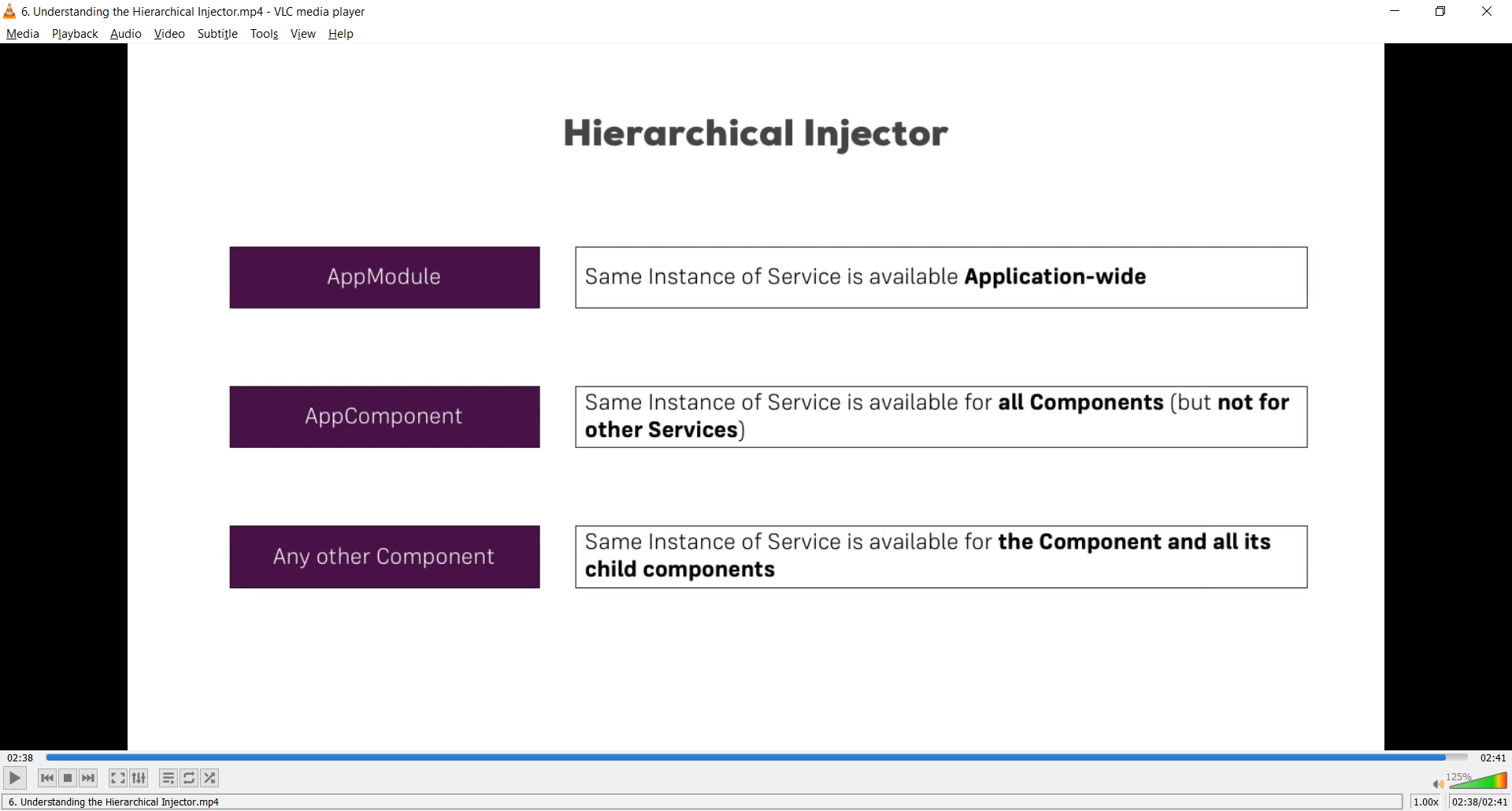
    this.accountsService.updateStatus(this.id, status);

*this.loggingService.logStatusChange(status);*

  }

}

6. Understanding the Hierarchical Injector



Service được cung cấp cho 1 component và tất cả các con của nó

7. How many Instances of Service Should It Be

Khi sử dụng providers thì nó sẽ k hoạt động như mong muốn

Nếu new-account component khai báo instance mới nó sẽ override instance được khai báo trong app-component => muốn có 1 instance thì chỉ cần khai báo trong app-component

8. Injecting Services into Services

app.module.ts

@NgModule({

  declarations: [

    AppComponent,

    AccountComponent,

    NewAccountComponent

  ],

  imports: [

    BrowserModule,

    FormsModule,

  ],

  providers: [AccountsService, LoggingService],

  bootstrap: [AppComponent]

})

accounts.service.ts thêm:

  constructor(private loggingService: LoggingService) {}

@Injectable()

export class AccountsService

Thông báo service is injectable or something is can be injected in there => inject service into another service

9. Using Services for Cross-Component Communication

Cần cung cấp event để component A giao tiếp vs B

  statusUpdated = new EventEmitter<string>();

account.component.ts

onSetTo(status: string) {

    this.accountsService.updateStatus(this.id, status);

*// this.loggingService.logStatusChange(status);*

    this.accountsService.statusUpdated.emit(status);

  }

new-account.component.ts

constructor(private loggingService: LoggingService,

              private accountsService: AccountsService) {

    this.accountsService.statusUpdated.subscribe(

      (status: string) => alert('New Status: ' + status)

    );

  }

Bài tập

If you're using **Angular 6+** (check your package.json  to find out), you can provide application-wide services in a different way.

Instead of adding a service class to the providers[]  array in AppModule , you can set the following config in @Injectable() :

@Injectable({providedIn: 'root'})

export class MyService { ... }

This is exactly the same as:

export class MyService { ... }

and

import { MyService } from './path/to/my.service';

@NgModule({

...

providers: [MyService]

})

export class AppModule { ... }

Using this new syntax is **completely optional**, the traditional syntax (using providers[] ) will still work. The "new syntax" does offer one advantage though: Services **can be loaded lazily**by Angular (behind the scenes) and redundant code can be removed automatically. This can lead to a better performance and loading speed - though this really only kicks in for bigger services and apps in general.

* Tạo file users.service.ts
* import { Injectable } from '@angular/core';
* import { CounterService } from './counter.service';
* @Injectable()
* export class UserService {
* activeUsers = ['Max', 'Anna'];
* inactiveUsers = ['Chris', 'Manu'];
* constructor(private counterService: CounterService) {}
* setToActive(id: number) {
* this.activeUsers.push(this.inactiveUsers[id]);
* this.inactiveUsers.splice(id, 1);
* this.counterService.incrementInActiveToActive();
* }
* setToInactive(id: number) {
* this.inactiveUsers.push(this.activeUsers[id]);
* this.activeUsers.splice(id, 1);
* this.counterService.incrementActiveToInactive();
* }
* }
* File counter.service.ts
* export class CounterService {
* activeToInactiveCounter = 0;
* inactiveToActiveCounter = 0;
* incrementActiveToInactive() {
* this.activeToInactiveCounter++;
* console.log('Active to Inactive: ' + this.activeToInactiveCounter);
* }
* incrementInActiveToActive() {
* this.inactiveToActiveCounter++;
* console.log('Inactive to Active: ' + this.inactiveToActiveCounter);
* }
* }

active-users.component.ts

export class ActiveUsersComponent implements OnInit {

  users: string[];

  constructor(private userService: UserService) {}

  ngOnInit() {

    this.users = this.userService.activeUsers;

  }

  onSetToInactive(id: number) {

    this.userService.setToInactive(id);

  }

}

active-users.component.html

<h3>Active Users</h3>

<ul class="list-group">

  <li

    class="list-group-item"

    \*ngFor="let user of users; let i = index">

    {{ user }} | <a href="#" (click)="onSetToInactive(i)">Set to Inactive</a>

  </li>

</ul>

app.module.ts

@NgModule({

  declarations: [

    AppComponent,

    ActiveUsersComponent,

    InactiveUsersComponent

  ],

  imports: [

    BrowserModule,

    FormsModule,

  ],

  providers: [CounterService],

  bootstrap: [AppComponent]

})

export class AppModule { }

nếu có Inject mới khai báo trong này

app.component.ts

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

import { UserService } from './users.service';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css'],

  providers: [UserService]

})

export class AppComponent {

}

## 10. Course Project - Services & Dependency Injection

2. Setting up the Services

App recipe book

Tạo file recipe.service.ts và shopping-list.service.ts trong folder tương ứng

3. Managing Recipes in a Recipe Service

recipe.service.ts

import { EventEmitter, Injectable } from '@angular/core';

import { Recipe } from './recipe.model';

import { Ingredient } from '../shared/ingredient.model';

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

@Injectable()

export class RecipeService {

  private recipes: Recipe[] = [

    new Recipe('A Test Recipe', 'This is simply a test', 'https://upload.wikimedia.org/wikipedia/commons/1/15/Recipe\_logo.jpeg'),

    new Recipe('Another Test Recipe', 'This is simply a test', 'https://upload.wikimedia.org/wikipedia/commons/1/15/Recipe\_logo.jpeg')

  ];

  getRecipes() {

    return this.recipes.slice(); // trả về mảng mới copy lại

  }

}

recipes.component.ts vì truy cập service chỉ trong phần này

@Component({

  selector: 'app-recipes',

  templateUrl: './recipes.component.html',

  styleUrls: ['./recipes.component.css'],

  providers: [RecipeService]

})

recipe-list.component.ts thêm constructor và vào hàm init thêm

export class RecipeListComponent implements OnInit {

  recipes: Recipe[];

  constructor(private recipeService: RecipeService) {

  }

  ngOnInit() {

    this.recipes = this.recipeService.getRecipes();

  }

}

4. Using a Service for Cross-Component Communication

Khi nhấn vào item ở trong list thì detail được update

recipe.service.ts

  recipeSelected = new EventEmitter<Recipe>();

recipe-item.component.ts

export class RecipeItemComponent implements OnInit {

  @Input() recipe: Recipe;

  constructor(private recipeService: RecipeService) { } // add

  ngOnInit() {

  }

  onSelected() {

    this.recipeService.recipeSelected.emit(this.recipe); // add

  }

}

Xóa hàm onRecipeSelected ở file recipe-list.component.html

<app-recipe-item

      \*ngFor="let recipeEl of recipes"

      [recipe]="recipeEl"

      (recipeSelected)="onRecipeSelected(recipeEl)"></app-recipe-item>

recipe-list.component.ts xóa những thứ k cần thiết

export class RecipeListComponent implements OnInit {

  recipes: Recipe[];

  constructor(private recipeService: RecipeService) {

  }

  ngOnInit() {

    this.recipes = this.recipeService.getRecipes();

  }

}

Xóa event tròn file recipeWasSelected ở recipes.component.html

<app-recipe-list

      (recipeWasSelected)="selectedRecipe = $event"></app-recipe-list>

recipes.component.ts

export class RecipesComponent implements OnInit {

  selectedRecipe: Recipe;

  constructor(private recipeService: RecipeService) { } // add

  ngOnInit() {

    this.recipeService.recipeSelected

      .subscribe(

        (recipe: Recipe) => {

          this.selectedRecipe = recipe;

        }

      );

  }

}

5. Adding the Shopping List Service

shopping-list.service.ts

import { Ingredient } from '../shared/ingredient.model';

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

export class ShoppingListService {

  ingredientsChanged = new EventEmitter<Ingredient[]>();

  private ingredients: Ingredient[] = [

    new Ingredient('Apples', 5),

    new Ingredient('Tomatoes', 10),

  ];

  getIngredients() {

    return this.ingredients.slice();

  }

}

Vào app-modules khai báo service

shopping-list.component.ts

export class ShoppingListComponent implements OnInit {

  ingredients: Ingredient[];

  constructor(private slService: ShoppingListService) { } // inject

  ngOnInit() {

    this.ingredients = this.slService.getIngredients();

  }

}

Vào service thêm

addIngredient(ingredient: Ingredient) {

    this.ingredients.push(ingredient);

    this.ingredientsChanged.emit(this.ingredients.slice());

  }

Vào xóa sự kiện emit trong file shopping-edit.component.ts

  @Output() ingredientAdded = new EventEmitter<Ingredient>();

this.ingredientAdded.emit(newIngredient); // xóa

chuyển thành

constructor(private slService: ShoppingListService) { }

  onAddItem() {

    const ingName = this.nameInputRef.nativeElement.value;

    const ingAmount = this.amountInputRef.nativeElement.value;

    const newIngredient = new Ingredient(ingName, ingAmount);

    this.slService.addIngredient(newIngredient);

  }

Vào file shopping-list xóa hàm onIngredientAdded

<app-shopping-edit

      (ingredientAdded)="onIngredientAdded($event)"></app-shopping-edit>

6. Using Services for Pushing Data from A to B

Khi ấn vào k add được vì lúc trả về là array copy

Vào service tạo

  ingredientsChanged = new EventEmitter<Ingredient[]>();

addIngredient(ingredients: Ingredient[]) {

    this.ingredients.push(...ingredients);

    this.ingredientsChanged.emit(this.ingredients.slice());

  }

Sửa lại hàm trong file shopping-list.component.ts

ngOnInit() {

    this.ingredients = this.slService.getIngredients();

    this.slService.ingredientsChanged

      .subscribe(

        (ingredients: Ingredient[]) => {

          this.ingredients = ingredients;

        }

      );

  }

7. Adding Ingredients to Recipes

Làm nút To shopping list work

Cập nhật model Recipe

export class Recipe {

  public name: string;

  public description: string;

  public imagePath: string;

  public ingredients: Ingredient[];

}

Cập nhật lại service

private recipes: Recipe[] = [

    new Recipe(

      'Tasty Schnitzel',

      'A super-tasty Schnitzel - just awesome!',

      'https://upload.wikimedia.org/wikipedia/commons/7/72/Schnitzel.JPG',

      [

        new Ingredient('Meat', 1),

        new Ingredient('French Fries', 20)

      ]),

    new Recipe('Big Fat Burger',

      'What else you need to say?',

      'https://upload.wikimedia.org/wikipedia/commons/b/be/Burger\_King\_Angus\_Bacon\_%26\_Cheese\_Steak\_Burger.jpg',

      [

        new Ingredient('Buns', 2),

        new Ingredient('Meat', 1)

      ])

  ];

recipe-detail.component.html thêm đoạn output ingredients

<div class="row">

  <div class="col-xs-12">

    <ul class="list-group">

      <li

        class="list-group-item"

        \*ngFor="let ingredient of recipe.ingredients">

        {{ ingredient.name }} - {{ ingredient.amount }}

      </li>

    </ul>

  </div>

</div>

8. Passing Ingredients from Recipes to the Shopping List (via a Service)

        <li><a (click)="onAddToShoppingList()" style="cursor: pointer;">To Shopping List</a></li>

recipe-detail.component.ts

constructor(private recipeService: RecipeService) { }

onAddToShoppingList() {

    this.recipeService.addIngredientsToShoppingList(this.recipe.ingredients);

  }

recipe.service.ts

constructor(private slService: ShoppingListService) {}

  addIngredientsToShoppingList(ingredients: Ingredient[]) {

    this.slService.addIngredients(ingredients);

  }

Nhớ thêm @Injectable()

shopping-list.service.ts

addIngredients(ingredients: Ingredient[]) {

*// for (let ingredient of ingredients) {*

*//   this.addIngredient(ingredient);*

*// }*

    this.ingredients.push(...ingredients);

    this.ingredientsChanged.emit(this.ingredients.slice());

  }

## 11. Changing Pages with Routing

In our app, we got three sections:

* Home
* Servers
  + View and Edit Servers
  + A Service is used to load and update Servers
* Users
  + View Users

This app will be improved by adding routing but definitely feel free to play around with it - besides routing, everything should be working fine.

4. Setting up and Loading Routes

App.module.ts

const appRoutes: Routes = [

  { path: "", component: HomeComponent },

  { path: "users", component: UsersComponent },

  { path: "servers", component: ServersComponent }

];

// và phải khai báo

imports: [BrowserModule, FormsModule, RouterModule.forRoot(appRoutes)]

app.component.html

<div class="row">

    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">

      <router-outlet></router-outlet>

    </div>

  </div>

5. Navigating with Router Links

Khi thay /servers sẽ load lại => issue

        <li role="presentation"><a href="/servers">Servers</a></li>

App.component.html sử dụng routerLink

<li role="presentation"

          <a routerLink="/">Home</a>

        </li>

        <li role="presentation">

          <a routerLink="servers">Servers</a>

        </li>

        <li role="presentation"\>

          <a [routerLink]="['users']">Users</a>

        </li>

6. Understanding Navigation Paths

Tại file servers.component.html

routerLink="/servers" // phải có dấu / mới load được vì nó là absolute path

./servers currently path

../servers go up a level

7. Styling Active Router Links

<ul class="nav nav-tabs">

        <li role="presentation"

            routerLinkActive="active"

            [routerLinkActiveOptions]="{exact: true}">

          <a routerLink="/">Home</a>

        </li>

        <li role="presentation"

            routerLinkActive="active">

          <a routerLink="servers">Servers</a>

        </li>

        <li role="presentation"

            routerLinkActive="active">

          <a [routerLink]="['users']">Users</a>

        </li>

      </ul>

[routerLinkActiveOptions]="{exact: true} tránh lúc nào cũng được active ở tab HOME vì nó kiểm tra nếu url có chứa thì sẽ active nên cần them cái này

8. Navigating Programmatically

[home.component.html](http://home.component.html)

<button class="btn btn-primary" (click)="onLoadServer(1)">Load Server 1</button>

File [home.component.ts](http://home.component.ts)

export class HomeComponent implements OnInit {

  constructor(private router: Router) { } // inject

  onLoadServer(id: number) {

*// complex calculation*

    this.router.navigate(['/servers']);

  }

}

9. Using Relative Paths in Programmatic Navigation

servers.component.html

*<button class="btn btn-primary" (click)="onReload()">Reload Page</button>-->*

Servers.component.ts

constructor(private serversService: ServersService,

              private router: Router

) { }

  onReload() {

*this.router.navigate(['servers']);*

  }

Navigate method doesn’t know route you are currently on is different routerLink

Nếu muốn biết route mà bạn đang ở đó thì SD:

constructor(private serversService: ServersService,

              private router: Router,

              private route: ActivatedRoute) {

  }

*this.router.navigate(['servers'], {relativeTo: this.route});*

10. Passing Parameters to Routes

  { path: "users/:id", component: UsersComponent },

11. Fetching Route Parameters

user.component.ts

constructor(private route: ActivatedRoute) { } // inject

  ngOnInit() {

    this.user = {

      id: this.route.snapshot.params['id']

    };

  }

12. Fetching Route Parameters Reactively

user.component.html

<p>User with ID {{ user.id }} loaded.</p>

<p>User name is {{ user.name }}</p>

<hr>

<a [routerLink]="['/users', 10, 'Anna']">Load Anna (10)</a>

Khi ở trang này bấm vào sẽ không cập nhật được data

user.component.ts

ngOnInit() {

    this.user = {

      id: this.route.snapshot.params['id'],

      name: this.route.snapshot.params['name']

    };

    this.paramsSubscription = this.route.params

      .subscribe(

        (params: Params) => {

          this.user.id = params['id'];

          this.user.name = params['name'];

        }

      );

  }

Observable is an easy way to subscribe to some event may happen in the future

Param is an observable

Bởi vì ta đang ở trên component mà mình định load nên angular sẽ k tạo ra instance mới

13. An Important Note about Route Observables

paramsSubscription: Subscription;

ngOnInit() {

    this.user = {

      id: this.route.snapshot.params['id'],

      name: this.route.snapshot.params['name']

    };

    this.paramsSubscription = this.route.params

      .subscribe(

        (params: Params) => {

          this.user.id = params['id'];

          this.user.name = params['name'];

        }

      );

  }

ngOnDestroy() {

    this.paramsSubscription.unsubscribe();

  }

14. Passing Query Parameters and Fragments

  { path: "servers/:id/edit", component: ServersComponent }

servers.component.html

<a

        [routerLink]="['/servers', server.id]"

        [queryParams]="{allowEdit: server.id === 3 ? '1' : '0'}"

        fragment="loading"

        href="#"

        class="list-group-item"

        \*ngFor="let server of servers">

        {{ server.name }}

      </a>

home.component.ts

onLoadServer(id: number) {

*// complex calculation*

    this.router.navigate(['/servers', id, 'edit'], {queryParams: {allowEdit: '1'}, fragment: 'loading'});

  }

Localhost:4200/servers/5/edit?allowEdit=3#loading

15. Retrieving Query Parameters and Fragments

edit-server.component.ts

constructor(private serversService: ServersService,

              private route: ActivatedRoute,

              private router: Router) {

  }

  ngOnInit() {

// 2 cách để access query params

    console.log(this.route.snapshot.queryParams);

    console.log(this.route.snapshot.fragment);

    this.route.queryParams

      .subscribe(

        (queryParams: Params) => {

          this.allowEdit = queryParams['allowEdit'] === '1' ? true : false;

        }

      );

    this.route.fragment.subscribe();

    const id = +this.route.snapshot.params['id'];

    this.server = this.serversService.getServer(id);

*// Subscribe route params to update the id if params change*

    this.serverName = this.server.name;

    this.serverStatus = this.server.status;

  }

16. Practicing and some Common Gotchas

users.component.html thêm

<a

        [routerLink]="['/users', user.id, user.name]"

        href="#"

        class="list-group-item"

        \*ngFor="let user of users">

        {{ user.name }}

      </a>

servers.component.html

<a

        [routerLink]="['/servers', server.id]"

        [queryParams]="{allowEdit: server.id === 3 ? '1' : '0'}"

        fragment="loading"

        href="#"

        class="list-group-item"

        \*ngFor="let server of servers">

        {{ server.name }}

      </a>

Server.component.ts

  constructor(private serversService: ServersService,

              private route: ActivatedRoute,

              private router: Router) {

  }

  ngOnInit() {

    // lấy tham số từ /user/:id khi click

*// const id = +this.route.snapshot.params['id'];*

*// this.server = this.serversService.getServer(id);*

*// this.route.params*

*//   .subscribe(*

*//     (params: Params) => {*

*//       this.server = this.serversService.getServer(+params['id']);*

*//     }*

*//   );*

  }

Comment servers.component.html

*<app-server></app-server>*

17. Setting up Child (Nested) Routes

servers.component.html

<div class="col-xs-12 col-sm-4">

    <router-outlet></router-outlet>

*<!--<button class="btn btn-primary" (click)="onReload()">Reload Page</button>-->*

*<!--<app-edit-server></app-edit-server>-->*

*<!--<hr>-->*

*<!--&lt;!&ndash;<app-server></app-server>&ndash;&gt;-->*

  </div>

const appRoutes: Routes = [

  { path: "", component: HomeComponent },

  { path: "users", component: UsersComponent },

  { path: "users/:id/:name", component: UserComponent },

  { path: "servers", component: ServersComponent, children: [

    { path: ":id", component: ServerComponent },

    { path: ":id/edit", component: EditServerComponent }

  ] }

];

users.component.html

<div class="col-xs-12 col-sm-4">

*<!--<app-user></app-user>-->*

    <router-outlet></router-outlet>

  </div>

18. Using Query Parameters - Practice

server.component.html

<h5>{{ server.name }}</h5>

<p>Server status is {{ server.status }}</p>

<button class="btn btn-primary" (click)="onEdit()">Edit Server</button>

server.component.ts

onEdit() {

    this.router.navigate(['edit'], {relativeTo: this.route, queryParamsHandling: 'preserve'});

  }

relativeTo: this.route load đường dẫn hiện tại thêm /edit vào

edit-server.component.ts

this.route.queryParams

      .subscribe(

        (queryParams: Params) => {

          this.allowEdit = queryParams['allowEdit'] === '1' ? true : false;

        }

      );

File html

<h4 \*ngIf="!allowEdit">You're not allowed to edit!</h4>

<div \*ngIf="allowEdit">

19. Configuring the Handling of Query Parameters

Fix params bị mất khi ấn vào Link ở list server rồi ấn vào button Edit server thêm queryParamsHandling: 'preserve'

20. Redirecting and Wildcard Routes

Tạo component page not found để url sai sẽ vào đây

{ path: ":not-found", component: PageNotFoundComponent },

{ path: "\*\*", relativeTo: 'not-found'  } // phải để cuối cùng

21. Important Redirection Path Matching

In our example, we didn't encounter any issues when we tried to redirect the user. But that's not always the case when adding redirections.

By default, Angular matches paths by prefix. That means, that the following route will match both /recipes  and just /

{ path: '', redirectTo: '/somewhere-else' }

Actually, Angular will give you an error here, because that's a common gotcha: This route will now **ALWAYS** redirect you! Why?

Since the default matching strategy is "prefix" , Angular checks if the path you entered in the URL does **start with the path** specified in the route. Of course every path starts with ''  (Important: That's no whitespace, it's simply "nothing").

To fix this behavior, you need to change the matching strategy to "full" :

{ path: '', redirectTo: '/somewhere-else', pathMatch: 'full' }

Now, you only get redirected, if the full path is ''  (so only if you got NO other content in your path in this example).

22. Outsourcing the Route Configuration

Tạo file app-routing.module.ts

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

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

import { PageNotFoundComponent } from './page-not-found/page-not-found.component';

import { EditServerComponent } from './servers/edit-server/edit-server.component';

import { ServerComponent } from './servers/server/server.component';

import { ServersComponent } from './servers/servers.component';

import { UserComponent } from './users/user/user.component';

import { UsersComponent } from './users/users.component';

import { HomeComponent } from './home/home.component';

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

import { CanDeactivateGuard } from './servers/edit-server/can-deactivate-guard.service';

import { ErrorPageComponent } from './error-page/error-page.component';

import { ServerResolver } from './servers/server/server-resolver.service';

const appRoutes: Routes = [

  { path: '', component: HomeComponent },

  { path: 'users', component: UsersComponent, children: [

    { path: ':id/:name', component: UserComponent }

  ] },

  {

    path: 'servers',

*// canActivate: [AuthGuard],*

    canActivateChild: [AuthGuard],

    component: ServersComponent,

    children: [

    { path: ':id', component: ServerComponent, resolve: {server: ServerResolver} },

    { path: ':id/edit', component: EditServerComponent, canDeactivate: [CanDeactivateGuard] }

  ] },

*// { path: 'not-found', component: PageNotFoundComponent },*

  { path: 'not-found', component: ErrorPageComponent, data: {message: 'Page not found!'} },

  { path: '\*\*', redirectTo: '/not-found' }

];

@NgModule({

  imports: [

*// RouterModule.forRoot(appRoutes, {useHash: true})*

    RouterModule.forRoot(appRoutes)

  ],

  exports: [RouterModule]

})

export class AppRoutingModule {

}

Module

imports: [

    BrowserModule,

    FormsModule,

    AppRoutingModule

  ],

23. An Introduction to Guards

24. Protecting Routes with canActivate

Tạo file auth-guard.service.ts

Chúng ta sẽ define that angular will execute this code before a route is loaded

canActivate(route: ActivatedRouteSnapshot,

              state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean

return về Observable, run asynchronously

auth.service.ts

export class AuthService {

  loggedIn = false;

  isAuthenticated() {

    const promise = new Promise(

      (resolve, reject) => {

        setTimeout(() => {

          resolve(this.loggedIn);

        }, 800);

      }

    );

    return promise;

  }

  login() {

    this.loggedIn = true;

  }

  logout() {

    this.loggedIn = false;

  }

}

Thêm Injectable và constructor

@Injectable()

export class AuthGuard implements CanActivate, CanActivateChild {

  constructor(private authService: AuthService, private router: Router) {}

canActivate(route: ActivatedRouteSnapshot,

              state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {

    return this.authService.isAuthenticated()

      .then(

        (authenticated: boolean) => {

          if (authenticated) {

            return true;

          } else {

            this.router.navigate(['/']);

          }

        }

      );

  }

Vào app-routing.module thêm *canActivate*

{

    path: 'servers',

*// canActivate: [AuthGuard],*

    canActivateChild: [AuthGuard],

    component: ServersComponent,

    children: [

    { path: ':id', component: ServerComponent, resolve: {server: ServerResolver} },

    { path: ':id/edit', component: EditServerComponent, canDeactivate: [CanDeactivateGuard] }

  ] },

Nhớ khai báo

  providers: [ServersService, AuthService, AuthGuard],

25. Protecting Child (Nested) Routes with canActivateChild

Implement thêm CanActivateChild

@Injectable()

export class AuthGuard implements CanActivate, CanActivateChild {

  constructor(private authService: AuthService, private router: Router) {}

  canActivate(route: ActivatedRouteSnapshot,

              state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {

    return this.authService.isAuthenticated()

      .then(

        (authenticated: boolean) => {

          if (authenticated) {

            return true;

          } else {

            this.router.navigate(['/']);

          }

        }

      );

  }

  canActivateChild(route: ActivatedRouteSnapshot,

                   state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {

    return this.canActivate(route, state);

  }

}

Thêm canActivateChild: [AuthGuard],

26. Using a Fake Auth Service

Vào home tạo btn login và log out

27. Controlling Navigation with canDeactivate

edit-server.component.ts

Thêm thuộc tính changesSaved = false;

onUpdateServer() {

    this.serversService.updateServer(this.server.id, {name: this.serverName, status: this.serverStatus});

    this.changesSaved = true;

    this.router.navigate(['../'], {relativeTo: this.route});

  }

Tạo file edit-server/ can-deactivate-guard.service.ts

import { Observable } from 'rxjs/Observable';

import { CanDeactivate, ActivatedRouteSnapshot, RouterStateSnapshot } from '@angular/router';

export interface CanComponentDeactivate {

  canDeactivate: () => Observable<boolean> | Promise<boolean> | boolean;

}

export class CanDeactivateGuard implements CanDeactivate<CanComponentDeactivate> {

  canDeactivate(component: CanComponentDeactivate,

                currentRoute: ActivatedRouteSnapshot,

                currentState: RouterStateSnapshot,

                nextState?: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {

    return component.canDeactivate();

  }

}

app-routing.module.ts

    { path: ':id/edit', component: EditServerComponent, canDeactivate: [CanDeactivateGuard] }

app.module.ts

providers:[ServersService, AuthService, AuthGuard, CanDeactivateGuard]

edit-server.component.ts

canDeactivate(): Observable<boolean> | Promise<boolean> | boolean {

    if (!this.allowEdit) {

      return true;

    }

    if ((this.serverName !== this.server.name || this.serverStatus !== this.server.status) && !this.changesSaved) {

      return confirm('Do you want to discard the changes?');

    } else {

      return true;

    }

  }

// thêm

const id = +this.route.snapshot.params['id'];

    this.server = this.serversService.getServer(id); // thay vì update id = 1

*// Subscribe route params to update the id if params change*

28. Passing Static Data to a Route

Tạo component error page

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute, Data } from '@angular/router';

@Component({

  selector: 'app-error-page',

  templateUrl: './error-page.component.html',

  styleUrls: ['./error-page.component.css']

})

export class ErrorPageComponent implements OnInit {

  errorMessage: string;

  constructor(private route: ActivatedRoute) { }

  ngOnInit() {

*// this.errorMessage = this.route.snapshot.data['message'];*

    this.route.data.subscribe(

      (data: Data) => {

        this.errorMessage = data['message'];

      }

    );

  }

}

app-routing.module.ts

*// { path: 'not-found', component: PageNotFoundComponent },*

  { path: 'not-found', component: ErrorPageComponent, data: {message: 'Page not found!'} },

Khai báo ErrorPageComponent

29. Resolving Dynamic Data with the resolve Guard

Tạo file server-resolver.service.ts

import { Resolve, ActivatedRouteSnapshot, RouterStateSnapshot } from '@angular/router';

import { Observable } from 'rxjs/Observable';

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

import { ServersService } from '../servers.service';

interface Server {

  id: number;

  name: string;

  status: string;

}

@Injectable()

export class ServerResolver implements Resolve<Server> {

  constructor(private serversService: ServersService) {}

  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<Server> | Promise<Server> | Server {

    return this.serversService.getServer(+route.params['id']);

  }

}

Khai báo

providers: [ServersService, AuthService, AuthGuard, CanDeactivateGuard, ServerResolver],

  bootstrap: [AppComponent]

file routing

    { path: ':id', component: ServerComponent, resolve: {server: ServerResolver} },

server.component.ts

ngOnInit() {

    this.route.data

      .subscribe(

        (data: Data) => {

          this.server = data['server'];

        }

      );

*// const id = +this.route.snapshot.params['id'];*

*// this.server = this.serversService.getServer(id);*

*// this.route.params*

*//   .subscribe(*

*//     (params: Params) => {*

*//       this.server = this.serversService.getServer(+params['id']);*

*//     }*

*//   );*

  }

Sử dụng cho asyn data

30. Understanding Location Strategies

*RouterModule.forRoot(appRoutes, {useHash: true})*

## 12. Course Project - Routing

1. Planning the General Structure

App recipe book

2. Setting Up Routes

Tạo file app-routing.module.ts

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

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

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

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

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

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

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

const appRoutes: Routes = [

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

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

    { path: '', component: RecipeStartComponent },

    { path: 'new', component: RecipeEditComponent },

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

    { path: ':id/edit', component: RecipeEditComponent },

  ] },

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

];

@NgModule({

  imports: [RouterModule.forRoot(appRoutes)],

  exports: [RouterModule]

})

export class AppRoutingModule {

}

pathMatch: 'full' nếu không bất kì path nào cũng redirect nên báo lỗi

Khai báo module

imports: [

    BrowserModule,

    FormsModule,

    AppRoutingModule

  ],

app.component.html

<app-header (featureSelected)="onNavigate($event)"></app-header>

<div class="container">

  <div class="row">

    <div class="col-md-12">

      <router-outlet></router-outlet>

    </div>

  </div>

</div>

<router-outlet></router-outlet> component được render

3. Adding Navigation to the App

Header.component.html

<li routerLinkActive ="active"><a routerLink="/recipes">Recipes</a>

</li>

        <li routerLinkActive="active"><a routerLink="/shopping-list">Shopping List</a></li>

header.component.ts sửa thành

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

@Component({

  selector: 'app-header',

  templateUrl: './header.component.html'

})

export class HeaderComponent {

}

4. Marking Active Routes

Thêm routerLinkActive

5. Fixing Page Reload Issues

Khi bấm vào list recipe thì page sẽ bị load lại

Vào file recipe-item.component.html, recipe-detail.component.html, header.component.html xóa href=#

<a

  style="cursor: pointer;"

  [routerLink]="[index]"

  routerLinkActive="active"

  class="list-group-item clearfix">

  <div class="pull-left">

    <h4 class="list-group-item-heading">{{ recipe.name }}</h4>

    <p class="list-group-item-text">{{ recipe.description }}</p>

  </div>

  <span class="pull-right">

        <img

          [src]="recipe.imagePath"

          alt="{{ recipe.name }}"

          class="img-responsive"

          style="max-height: 50px;">

      </span>

</a>

Thêm style="cursor: pointer;"

6. Child Routes Challenge

7. Adding Child Routing Together

Ng g c recipes/recipe-start

Nội dung file html

<h3>Please select a Recipe!</h3>

Khai báo RecipeStartComponent trong module

recipes.component.html

<div class="row">

  <div class="col-md-5">

    <app-recipe-list></app-recipe-list>

  </div>

  <div class="col-md-7">

    <router-outlet></router-outlet>

  </div>

</div>

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

    { path: '', component: RecipeStartComponent },

    { path: 'new', component: RecipeEditComponent },

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

    { path: ':id/edit', component: RecipeEditComponent },

  ] },

8. Configuring Route Parameters

recipe-detail.component.ts

export class RecipeDetailComponent implements OnInit {

  recipe: Recipe;

  id: number;

  constructor(private recipeService: RecipeService,

              private route: ActivatedRoute,

              private router: Router) {

  }

  ngOnInit() {

    this.route.params

      .subscribe(

        (params: Params) => {

          this.id = +params['id'];

          this.recipe = this.recipeService.getRecipe(this.id);

        }

      );

  }

}

Vào recipe xóa các event không cần thiết

Service

getRecipe(index: number) {

    return this.recipes[index];

  }

9. Passing Dynamic Parameters to Links

recipe-item.component.html

<a

  style="cursor: pointer;"

  [routerLink]="[index]"

  routerLinkActive="active"

  class="list-group-item clearfix">

  <div class="pull-left">

    <h4 class="list-group-item-heading">{{ recipe.name }}</h4>

    <p class="list-group-item-text">{{ recipe.description }}</p>

  </div>

  <span class="pull-right">

        <img

          [src]="recipe.imagePath"

          alt="{{ recipe.name }}"

          class="img-responsive"

          style="max-height: 50px;">

      </span>

</a>

Thêm ở file .ts:  @Input() index: number;

recipe-list.component.html

<div class="col-xs-12">

    <app-recipe-item

      \*ngFor="let recipeEl of recipes; let i = index"

      [recipe]="recipeEl"

      [index]="i"></app-recipe-item>

  </div>

11. Adding Editing Routes

Tạo component recipe-edit

Vào recipe-list thêm

 <div class="col-xs-12">

    <button class="btn btn-success" (click)="onNewRecipe()">New Recipe</button>

  </div>

export class RecipeListComponent implements OnInit {

  recipes: Recipe[];

  constructor(private recipeService: RecipeService,

              private router: Router,

              private route: ActivatedRoute) {

  }

  ngOnInit() {

    this.recipes = this.recipeService.getRecipes();

  }

  onNewRecipe() {

    this.router.navigate(['new'], {relativeTo: this.route}); // current route

  }

}

Để new lên trước nếu không nó sẽ nhận new là id mà parse => err

{ path: 'new', component: RecipeEditComponent },

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

    { path: ':id/edit', component: RecipeEditComponent },

12. Retrieving Route Parameters

recipe-edit.component.ts

export class RecipeEditComponent implements OnInit {

  id: number;

  editMode = false;

  constructor(private route: ActivatedRoute) { }

  ngOnInit() {

    this.route.params

      .subscribe(

        (params: Params) => {

          this.id = +params['id'];

          this.editMode = params['id'] != null; // pb với new

        }

      );

  }

}

13. Programmatic Navigation to the Edit Page

recipe-detail.component.html

<li><a style="cursor: pointer;" (click)="onEditRecipe()">Edit

 Recipe</a></li>

constructor(private recipeService: RecipeService,

              private route: ActivatedRoute,

              private router: Router) {

  }

onEditRecipe() {

    this.router.navigate(['edit'], {relativeTo: this.route});

*// this.router.navigate(['../', this.id, 'edit'], {relativeTo: this.route}); cách 2*

  }

There's one thing I forgot to clean up here (will be cleaned up later in the course). Feel free to do the cleanup right now though.

Our app.component.html file looks like that:

<app-header (featureSelected)="onNavigate($event)"></app-header>

<div class="container">

<div class="row">

<div class="col-md-12">

<router-outlet></router-outlet>

</div>

</div>

</div>

The (featureSelected)="..."  event listener is a relict of our "old" navigation approach using ngIf. We no longer need it, so feel free to change this template to:

<app-header></app-header>

<div class="container">

<div class="row">

<div class="col-md-12">

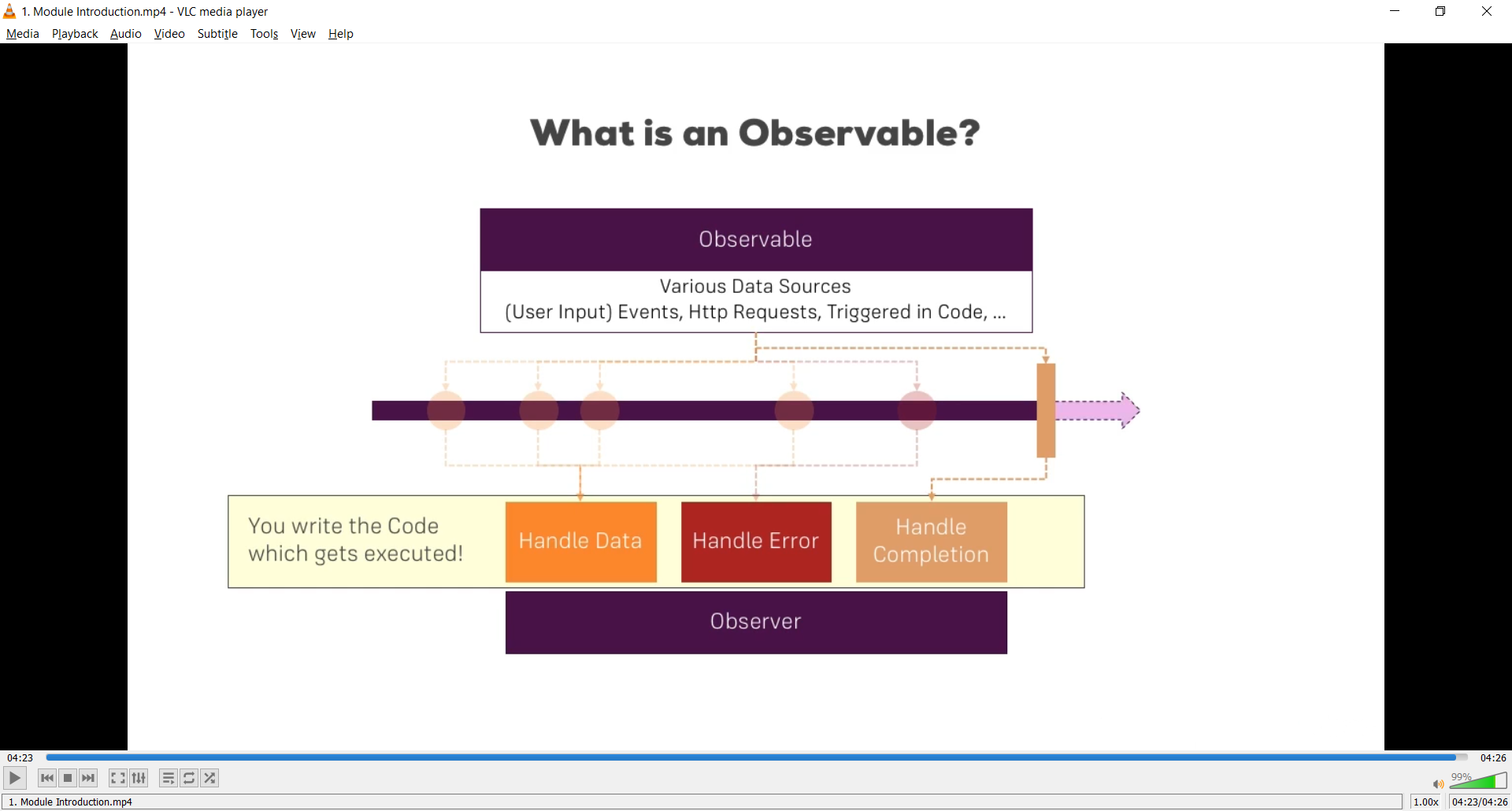
<router-outlet></router-outlet>

</div>

</div>

</div>

## 13. Understanding Observables



Useful Resources:

* Official Docs: <https://rxjs-dev.firebaseapp.com/>
* RxJS Series: <https://academind.com/learn/javascript/understanding-rxjs/>
* Updating to RxJS 6: <https://academind.com/learn/javascript/rxjs-6-what-changed/>

Observables nằm trong rxjs

home.component.ts

export class HomeComponent implements OnInit, OnDestroy {

  private firstObsSubscription: Subscription;

  constructor() {

  }

  ngOnInit() {

    this.firstObsSubscription = interval(1000).subscribe(count => {

      console.log(count);

    });

  }

  ngOnDestroy(): void {

    this.firstObsSubscription.unsubscribe();

  }

}

Hàm interval sau 1s sẽ emit event, khi vào trang chủ sẽ tự động tăng biến count nếu quay lại thì sẽ có thêm 1 biến đếm bắt đầu từ 0 và tang song song => destroy

Param k cần unsubscribe bởi vì nó được cung cấp bởi angular nên không cần làm thủ công

4. Building a Custom Observable

Home

ngOnInit() {

*// this.firstObsSubscription = interval(1000).subscribe(count => {*

*//   console.log(count);*

*// });*

    const customIntervalObservable = Observable.create(observer => {

      let count = 0;

      setInterval(() => {

        observer.next(count); // emit

        if (count === 5) {

          observer.complete();

        }

        if (count > 3) {

          observer.error(new Error('Count is greater 3!'));

        }

        count++;

      }, 1000);

    });

    this.firstObsSubscription = customIntervalObservable.subscribe(data => {

      console.log(data);

    }, error => {

      console.log(error);

      alert(error.message);

    }, () => {

      console.log('Completed!');

    });

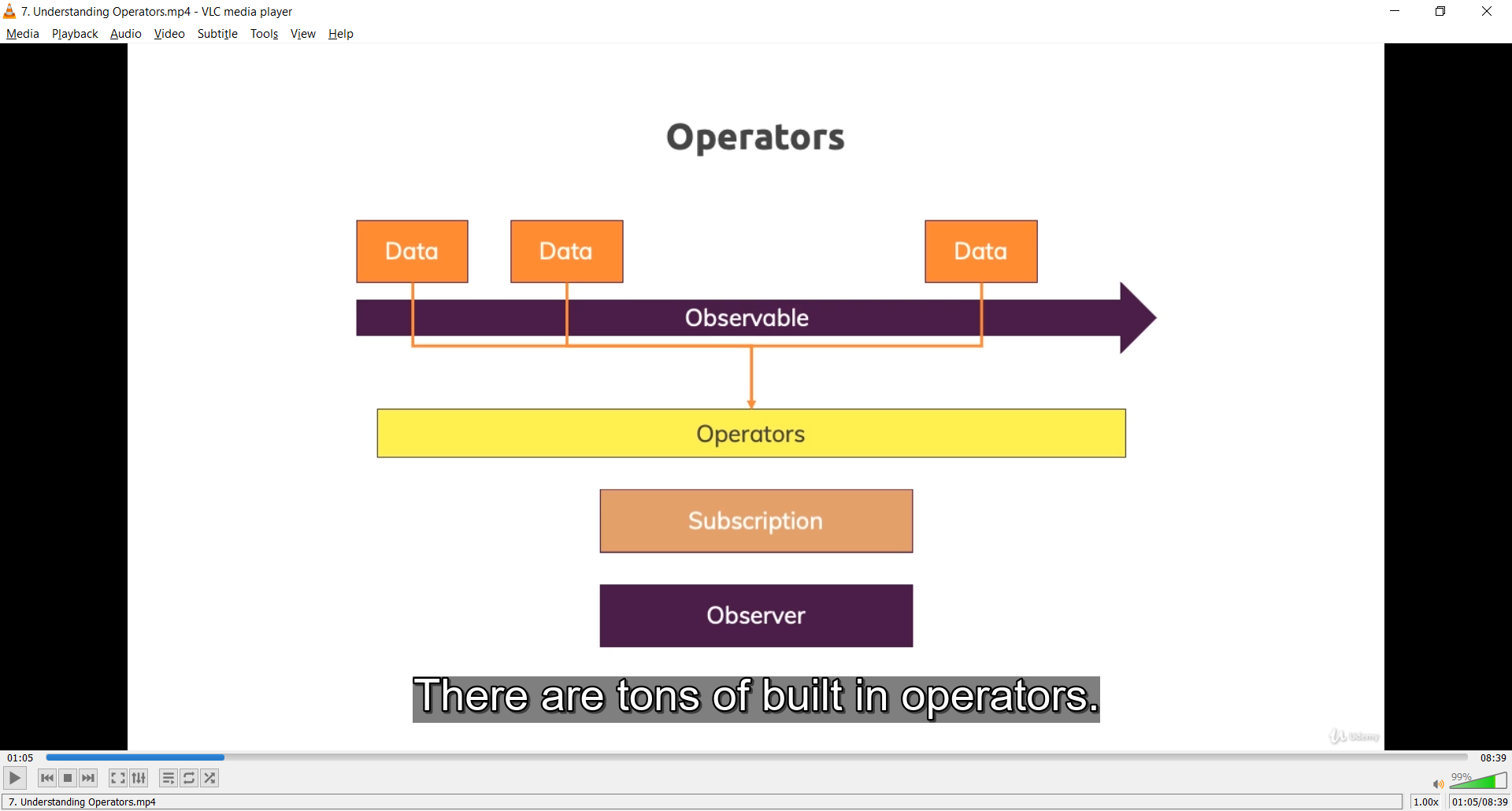
  }

  ngOnDestroy(): void {

    this.firstObsSubscription.unsubscribe();

  }

7. Understanding Operators



this.firstObsSubscription = customIntervalObservable.pipe(filter(data

 => {

      return data > 0;

    }), map((data: number) => {

      return 'Round: ' + (data + 1);

    })).subscribe(data => {

      console.log(data);

    }, error => {

      console.log(error);

      alert(error.message);

    }, () => {

      console.log('Completed!');

    });

8. Subjects

user.component.html

<p>User with <strong>ID {{ id }}</strong> was loaded</p>

<button class="btn btn-primary" (click)="onActivate()">Activate</button>

user.component.ts

constructor(private route: ActivatedRoute, private userService: UserSe

rvice) {

  }

  ngOnInit() {

    this.route.params.subscribe((params: Params) => {

      this.id = +params.id;

    });

  }

  onActivate() {

    this.userService.activatedEmitter.next(true);

  }

user.service.ts

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

import { Subject } from 'rxjs';

@Injectable({providedIn: 'root'})

export class UserService {

  activatedEmitter = new Subject<boolean>();

}

app.component.ts

export class AppComponent implements OnInit, OnDestroy {

  userActivated = false;

  private activatedSub: Subscription;

  constructor(private userService: UserService) {

  }

  ngOnInit() {

    this.activatedSub = this.userService.activatedEmitter.subscribe(didActivate => {

      this.userActivated = didActivate;

    });

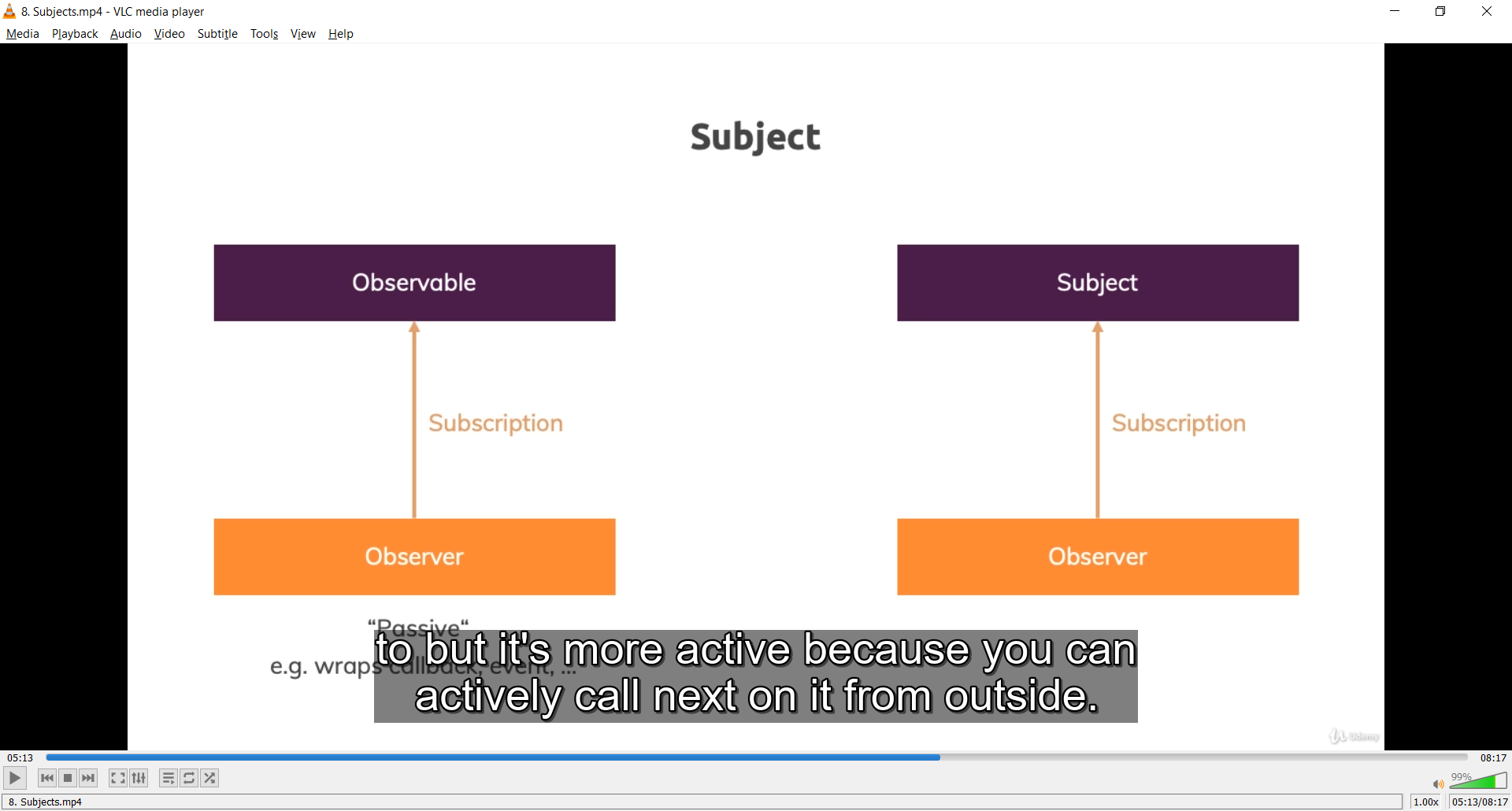
  }

  ngOnDestroy(): void {

    this.activatedSub.unsubscribe();

  }

}



## 14. Course Project - Observables

EventEmitter push new info from A to B

Vào app.component xóa hàm onNavigate

export class ShoppingListService {

  ingredientsChanged = new Subject<Ingredient[]>();

thay hết emit => next

shopping-list.component.ts

export class ShoppingListComponent implements OnInit, OnDestroy {

  ingredients: Ingredient[];

  private igChangeSub: Subscription;

  constructor(private slService: ShoppingListService) {

  }

  ngOnInit() {

    this.ingredients = this.slService.getIngredients();

    this.igChangeSub = this.slService.ingredientsChanged

      .subscribe(

        (ingredients: Ingredient[]) => {

          this.ingredients = ingredients;

        }

      );

  }

  ngOnDestroy(): void {

    this.igChangeSub.unsubscribe();

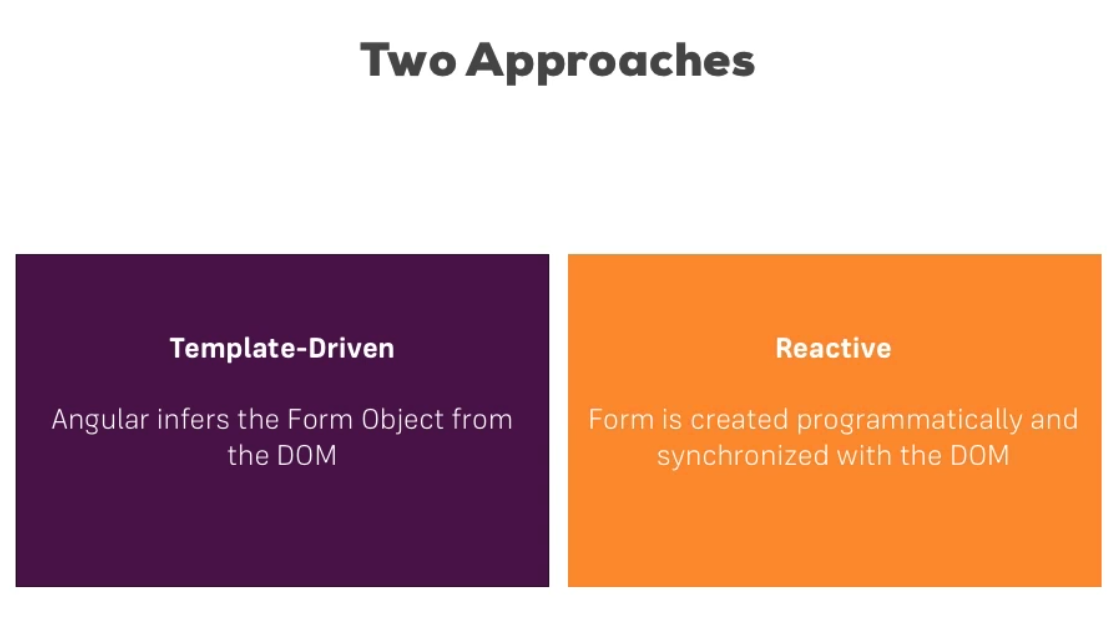
  }

}

 xóa recipeSelected = new EventEmitter<Recipe>(); trong service và

recipes component

## 15. Handling Forms in Angular Apps



Reactive: set up html code and connect manually you can fine tune every a little piece in the form

<form>

        <div id="user-data">

          <div class="form-group">

            <label for="username">Username</label>

            <input type="text" id="username" class="form-control">

          </div>

          <button class="btn btn-default" type="button">Suggest an Username</button>

          <div class="form-group">

            <label for="email">Mail</label>

            <input type="email" id="email" class="form-control">

          </div>

        </div>

        <div class="form-group">

          <label for="secret">Secret Questions</label>

          <select id="secret" class="form-control">

            <option value="pet">Your first Pet?</option>

            <option value="teacher">Your first teacher?</option>

          </select>

        </div>

        <button class="btn btn-primary" type="submit">Submit</button>

      </form>

5. TD Creating the Form and Registering the Controls

Import FormsModule

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

@NgModule({

  declarations: [

    AppComponent

  ],

  imports: [

    BrowserModule,

    FormsModule,

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

Sau đó vào thẻ input thêm ngModel(two-way data binding) và name attr in HTML

<input

              type="email"

              id="email"

              class="form-control"

              ngModel

              name="email"

              required

              email

              #email="ngModel">

6. TD Submitting and Using the Form

File html

      <form (ngSubmit)="onSubmit()" #f="ngForm">

#f="ngForm"> : cho phép truy cập vào form

Nếu để #f thì phải truyền như tham số vào hàm onSubmit, ghi log ra để xem

app.component.ts

*// onSubmit(f: HTMLElementForm) { Nếu để #f*

*//   console.log(f);*

*// }*

*// onSubmit(form: NgForm) {*

*//   console.log(form);*

*// }*

7. TD Understanding Form State

8. TD Accessing the Form with @ViewChild

Nếu không truyền tham số vào hàm thì sử dụng @ViewChild

  @ViewChild('f', { static: false }) signupForm: NgForm;

onSubmit() {

    this.submitted = true;

    this.user.username = this.signupForm.value.userData.username;

    this.user.email = this.signupForm.value.userData.email;

    this.user.secretQuestion = this.signupForm.value.secret;

    this.user.answer = this.signupForm.value.questionAnswer;

    this.user.gender = this.signupForm.value.gender;

    this.signupForm.reset();

  }

9. TD Adding Validation to check User Input

Thêm require, email => nhập giá trị hợp lên ghi log ra console giá trị valid = true và F12 để kiểm tra angular sẽ thêm các class

10. Built-in Validators & Using HTML5 Validation

Which Validators do ship with Angular?

Check out the Validators class: <https://angular.io/api/forms/Validators> - these are all built-in validators, though that are the methods which actually get executed (and which you later can add when using the reactive approach).

For the template-driven approach, you need the directives. You can find out their names, by searching for "validator" in the official docs: <https://angular.io/api?type=directive> - everything marked with "D" is a directive and can be added to your template.

Additionally, you might also want to enable HTML5 validation (by default, Angular disables it). You can do so by adding the ngNativeValidate  to a control in your template.

11. TD Using the Form State

<button

          class="btn btn-primary"

          type="submit"

          [disabled]="!f.valid">Submit</button>

File css

input.ng-invalid.ng-touched {

  border: 1px solid red;

}

12. TD Outputting Validation Error Messages

<span class="help-

block" \*ngIf="!email.valid && email.touched">Please enter a valid email!</span>

Thêm local reference #email=”ngModel”

13. TD Set Default Values with ngModel Property Binding

Thêm biến defaultQuestion ở file ts và để ngModel trong dấu [] là prop

<div class="form-group">

          <label for="secret">Secret Questions</label>

          <select

            id="secret"

            class="form-control"

            [ngModel]="defaultQuestion"

            name="secret">

            <option value="pet">Your first Pet?</option>

            <option value="teacher">Your first teacher?</option>

          </select>

        </div>

14. TD Using ngModel with Two-Way-Binding

defaultQuestion = 'teacher';

  answer = '';

file html

<div class="form-group">

          <textarea

            name="questionAnswer"

            rows="3"

            class="form-control"

            [(ngModel)]="answer"></textarea>

        </div>

        <p>Your reply: {{ answer }}</p>

15. TD Grouping Form Controls

Sử dụng ngModelGroup để group sau này truy cập ra phải .userData

userData is local reference

 <div

          id="user-data"

          ngModelGroup="userData"

          #userData="ngModelGroup">

          <div class="form-group">

            <label for="username">Username</label>

            <input

              type="text"

              id="username"

              class="form-control"

              ngModel

              name="username"

              required>

          </div>

          <button

            class="btn btn-default"

            type="button"

            (click)="suggestUserName()">Suggest an Username</button>

          <div class="form-group">

            <label for="email">Mail</label>

            <input

              type="email"

              id="email"

              class="form-control"

              ngModel

              name="email"

              required

              email

              #email="ngModel">

            <span class="help-block" \*ngIf="!email.valid && email.touched">Please enter a valid email!</span>

          </div>

        </div>

<p \*ngIf="!userData.valid && userData.touched">User Data is invalid!</p>

16. TD Handling Radio Buttons

  genders = ['male', 'female'];

<div class="radio" \*ngFor="let gender of genders">

          <label>

            <input

              type="radio"

              name="gender"

              ngModel

              [value]="gender"

              required>

            {{ gender }}

          </label>

        </div>

17. TD Setting and Patching Form Values

suggestUserName() {

    const suggestedName = 'Superuser';

*// this.signupForm.setValue({*

*//   userData: {*

*//     username: suggestedName,*

*//     email: ''*

*//   },*

*//   secret: 'pet',*

*//   questionAnswer: '',*

*//   gender: 'male'*

*// });*

*// cách này k ghi đè giá trị*

    this.signupForm.form.patchValue({

      userData: {

        username: suggestedName

      }

    });

  }

18. TD Using Form Data

<div class="row" \*ngIf="submitted">

    <div class="col-xs-12">

      <h3>Your Data</h3>

      <p>Username: {{ user.username }}</p>

      <p>Mail: {{ user.email }}</p>

      <p>Secret Question: Your first {{ user.secretQuestion }}</p>

      <p>Answer: {{ user.answer }}</p>

      <p>Gender: {{ user.gender }}</p>

    </div>

  </div>

19. TD Resetting Forms

    this.signupForm.reset();

có thể sử dụng hàm setValue()

21. Introduction to the Reactive Approach

export class AppComponent implements OnInit {

  genders = ['male', 'female'];

  signupForm: FormGroup;

khai báo ReactiveFormsModule

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

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

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

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

@NgModule({

  declarations: [

    AppComponent

  ],

  imports: [

    BrowserModule,

    ReactiveFormsModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

23. Reactive Creating a Form in Code

Sử dụng hàm onInit

ngOnInit() {

// create control

    this.signupForm = new FormGroup({

      'userData': new FormGroup({

        'username': new FormControl(null, Validators.required),

        'email': new FormControl(null, [Validators.required, Validators.email])

      }),

      'gender': new FormControl('male')

    });

  }

New FormControl nhận 2 tham số, thứ nhất là giá trị khởi tạo, tham số thứ 2 là single validator, tham số thứ 3 là potential async validator

24. Reactive Syncing HTML and Form

Thông báo sử dụng my formGroup tham chiếu đến signupForm bằng cách thêm vào thẻ form

Sử dụng formControlName thay cho name

Có thể sử dụng               [formControlName]="’username’"

 <form [formGroup]="signupForm" (ngSubmit)="onSubmit()">

        <div formGroupName="userData">

          <div class="form-group">

            <label for="username">Username</label>

            <input

              type="text"

              id="username"

              formControlName="username"

              class="form-control">

          </div>

          <div class="form-group">

            <label for="email">email</label>

            <input

              type="text"

              id="email"

              formControlName="email"

              class="form-control">

          </div>

        </div>

 <div class="radio" \*ngFor="let gender of genders">

          <label>

            <input

              type="radio"

              formControlName="gender"

              [value]="gender">{{ gender }}

          </label>

        </div>

25. Reactive Submitting the Form

Để submit thì thêm hàm (ngSubmit)="onSubmit()

onSubmit() {

    console.log(this.signupForm);

    this.signupForm.reset();

  }

26. Reactive Adding Validation

Nếu muốn config tham số thì truyền tham số ở file ts

ngOnInit() {

// create control

    this.signupForm = new FormGroup({

      'userData': new FormGroup({

        'username': new FormControl(null, Validators.required),

        'email': new FormControl(null, [Validators.required, Validators.email])

      }),

      'gender': new FormControl('male')

    });

  }

27. Reactive Getting Access to Controls

<span

\*ngIf="!signupForm.get('userData.username').valid && signupForm.get('userData.username').touched" class="help-block">Invalid </span>

<span

          \*ngIf="!signupForm.valid && signupForm.touched"

          class="help-block">Please enter valid data!</span>

file css

input.ng-invalid.ng-touched {

  border: 1px solid red;

}

28. Reactive Grouping Controls

Tạo form Group để chứa form control

        <div formGroupName="userData"> </div>

29. Reactive Arrays of Form Controls (FormArray)

      'hobbies': new FormArray([])

onAddHobby() {

    const control = new FormControl(null, Validators.required);

    (<FormArray>this.signupForm.get('hobbies')).push(control);

  }

File html: formArrayName="hobbies" connect

<div formArrayName="hobbies">

          <h4>Your Hobbies</h4>

          <button

            class="btn btn-default"

            type="button"

            (click)="onAddHobby()">Add Hobby</button>

          <div

            class="form-group"

            \*ngFor="let hobbyControl of signupForm.get('hobbies').controls; let i = index">

            <input type="text" class="form-control" [formControlName]="i">

          </div>

        </div>

30. Reactive Creating Custom Validators

  forbiddenUsernames = ['Chris', 'Anna'];

forbiddenNames(control: FormControl): {[s: string]: boolean} {

    if (this.forbiddenUsernames.indexOf(control.value) !== -1) {

      return {'nameIsForbidden': true};

    }

    return null;

  }

this.signupForm = new FormGroup({

      'userData': new FormGroup({

        'username': new FormControl(null, [Validators.required, this.forbiddenNames.bind(this)]),

        'email': new FormControl(null, [Validators.required, Validators.email], this.forbiddenEmails)

      }),

      'gender': new FormControl('male'),

      'hobbies': new FormArray([])

    });

Phải có bind this nó mới hiểu this trong hàm kia

31. Reactive Using Error Codes

<span

              \*ngIf="!signupForm.get('userData.username').valid && signupForm.get('userData.username').touched"

              class="help-block">

              <span \*ngIf="signupForm.get('userData.username').errors['nameIsForbidden']">This name is invalid!</span>

              <span \*ngIf="signupForm.get('userData.username').errors['required']">This field is required!</span>

            </span>

32. Reactive Creating a Custom Async Validator

forbiddenEmails(control: FormControl): Promise<any> | Observable<any>

{

    const promise = new Promise<any>((resolve, reject) => {

      setTimeout(() => {

        if (control.value === 'test@test.com') {

          resolve({'emailIsForbidden': true});

        } else {

          resolve(null);

        }

      }, 1500);

    });

    return promise;

  }

33. Reactive Reacting to Status or Value Changes

*// this.signupForm.valueChanges.subscribe(*

*//   (value) => console.log(value)*

*// );*

    this.signupForm.statusChanges.subscribe(

      (status) => console.log(status)

    );

34. Reactive Setting and Patching Values

this.signupForm.setValue({

      'userData': {

        'username': 'Max',

        'email': 'max@test.com'

      },

      'gender': 'male',

      'hobbies': []

    });

    this.signupForm.patchValue({

      'userData': {

        'username': 'Anna',

      }

    });

// cách 2

    this.signupForm.reset();

có thẻ chỉ rõ field cần reset

## 16. Course Project - Forms

2. TD Adding the Shopping List Form

Recipe book app

shopping-edit.component.html

Nhớ khai báo FormModule trước

Không cần sử dụng local ref như cũ thêm ngForm và name

<form (ngSubmit)="onSubmit(f)" #f="ngForm">

      <div class="row">

        <div class="col-sm-5 form-group">

          <label for="name">Name</label>

          <input

            type="text"

            id="name"

            class="form-control"

            name="name"

            ngModel

          >

        </div>

        <div class="col-sm-2 form-group">

          <label for="amount">Amount</label>

          <input

            type="number"

            id="amount"

            class="form-control"

            name="amount"

            ngModel

          >

        </div>

      </div>

      <div class="row">

        <div class="col-xs-12">

          <button

            class="btn btn-success"

            type="submit"

            [disabled]="!f.valid">{{ editMode ? 'Update' : 'Add' }}</button>

        </div>

      </div>

    </form>

shopping-edit.component.ts

onSubmit(form: NgForm) {

    const value = form.value;

    const newIngredient = new Ingredient(value.name, value.amount);

3. Adding Validation to the Form

<div class="col-sm-2 form-group">

          <label for="amount">Amount</label>

          <input

            type="number"

            id="amount"

            class="form-control"

            name="amount"

            ngModel

            required

            pattern="^[1-9]+[0-9]\*$"

          >

        </div>

Check số dương bằng pattern

4. Allowing the Selection of Items in the List

shopping-list.component.html thêm event onEditItem

<a

        class="list-group-item"

        style="cursor: pointer"

        \*ngFor="let ingredient of ingredients; let i = index"

        (click)="onEditItem(i)"

      >

        {{ ingredient.name }} ({{ ingredient.amount }})

      </a>

onEditItem(index: number) {

    this.slService.startedEditing.next(index); // emit event

  }

Service

Thêm   startedEditing = new Subject<number>();

shopping-edit.component.ts lắng nghe lấy ra item mà click đổ data vào form

  @ViewChild('f', { static: false }) slForm: NgForm; // local ref

subscription: Subscription;

  editMode = false;

  editedItemIndex: number;

  editedItem: Ingredient;

  constructor(private slService: ShoppingListService) { }

  ngOnInit() {

    this.subscription = this.slService.startedEditing

      .subscribe(

        (index: number) => {

          this.editedItemIndex = index;

          this.editMode = true;

          this.editedItem = this.slService.getIngredient(index);

          this.slForm.setValue({

            name: this.editedItem.name,

            amount: this.editedItem.amount

          })

        }

      );

  }

ngOnDestroy() {

    this.subscription.unsubscribe();

  }

5. Loading the Shopping List Items into the Form

shopping-list.service.ts

getIngredient(index: number) {

    return this.ingredients[index];

  }

6. Updating existing Items

Hiển thị đúng tên btn khi edit

<button

            class="btn btn-success"

            type="submit"

            [disabled]="!f.valid">{{ editMode ? 'Update' : 'Add' }}</button>

Service

updateIngredient(index: number, newIngredient: Ingredient) {

    this.ingredients[index] = newIngredient;

    this.ingredientsChanged.next(this.ingredients.slice());

  }

onSubmit(form: NgForm) {

    const value = form.value;

    const newIngredient = new Ingredient(value.name, value.amount);

    if (this.editMode) {

      this.slService.updateIngredient(this.editedItemIndex, newIngredient);

    } else {

      this.slService.addIngredient(newIngredient);

    }

    this.editMode = false;

    form.reset();

  }

7. Resetting the Form

Hàm add

    this.editMode = false;

form.reset(); // clear

8. Allowing the the User to Clear (Cancel) the Form

onClear() {

    this.slForm.reset();

    this.editMode = false;

  }

9. Allowing the Deletion of Shopping List Items

<button

            class="btn btn-danger"

            type="button"

            (click)="onDelete()"

            \*ngIf="editMode">Delete</button>

onDelete() {

    this.slService.deleteIngredient(this.editedItemIndex);

    this.onClear();

  }

Service

deleteIngredient(index: number) {

    this.ingredients.splice(index, 1);

    this.ingredientsChanged.next(this.ingredients.slice());

  }

10. Creating the Template for the (Reactive) Recipe Edit Form

Cách 2

recipe-edit.component.html tạo template bình thường

Nhớ khai báo reactiveFormModule

11. Creating the Form For Editing Recipes

Tạo hàm init

private initForm() {

    let recipeName = '';

    let recipeImagePath = '';

    let recipeDescription = '';

    let recipeIngredients = new FormArray([]);

    if (this.editMode) {

      const recipe = this.recipeService.getRecipe(this.id);

      recipeName = recipe.name;

      recipeImagePath = recipe.imagePath;

      recipeDescription = recipe.description;

// thêm sau

      if (recipe['ingredients']) {

        for (let ingredient of recipe.ingredients) {

          recipeIngredients.push(

            new FormGroup({

              'name': new FormControl(ingredient.name, Validators.required),

              'amount': new FormControl(ingredient.amount, [

                Validators.required,

                Validators.pattern(/^[1-9]+[0-9]\*$/)

              ])

            })

          );

        }

      }

    }

    this.recipeForm = new FormGroup({

      'name': new FormControl(recipeName, Validators.required),

      'imagePath': new FormControl(recipeImagePath, Validators.required),

      'description': new FormControl(recipeDescription, Validators.required),

      'ingredients': recipeIngredients

    });

  }

12. Syncing HTML with the Form

Thêm FormControlName, FormGroup vào file html

13. Adding Ingredient Controls to a Form Array

<div

            class="row"

            \*ngFor="let ingredientCtrl of recipeForm.get('ingredients').controls; let i = index"

            [formGroupName]="i"

            style="margin-top: 10px;">

            <div class="col-xs-8">

              <input

                type="text"

                class="form-control"

                formControlName="name">

            </div>

            <div class="col-xs-2">

              <input

                type="number"

                class="form-control"

                formControlName="amount">

            </div>

            <div class="col-xs-2">

              <button

                type="button"

                class="btn btn-danger"

                (click)="onDeleteIngredient(i)">X</button>

            </div>

          </div>

Fix

In the last lecture, we added some code to access the controls of our form array:

\*ngFor="let ingredientCtrl of recipeForm.get('ingredients').controls; let i = index"

This code will fail as of the latest Angular version.

You can fix it easily though. Outsource the "get the controls" logic into a method of your component code (the .ts file):

getControls() {

return (<FormArray>this.recipeForm.get('ingredients')).controls;

}

In the template, you can then use:

\*ngFor="let ingredientCtrl of getControls(); let i = index"

This adjustment is required due to the way TS works and Angular parses your templates (it doesn't understand TS there).

15. Adding new Ingredient Controls

onAddIngredient() {

    (<FormArray>this.recipeForm.get('ingredients')).push(

      new FormGroup({

        'name': new FormControl(null, Validators.required),

        'amount': new FormControl(null, [

          Validators.required,

          Validators.pattern(/^[1-9]+[0-9]\*$/)

        ])

      })

    );

  }

16. Validating User Input

File css

input.ng-invalid.ng-touched,

textarea.ng-invalid.ng-touched {

  border: 1px solid red;

}

17. Submitting the Recipe Edit Form

onSubmit() {

*// const newRecipe = new Recipe(*

*//   this.recipeForm.value['name'],*

*//   this.recipeForm.value['description'],*

*//   this.recipeForm.value['imagePath'],*

*//   this.recipeForm.value['ingredients']);*

    if (this.editMode) {

      this.recipeService.updateRecipe(this.id, this.recipeForm.value);

    } else {

      this.recipeService.addRecipe(this.recipeForm.value);

    }

    this.onCancel();

  }

Service

  recipesChanged = new Subject<Recipe[]>();

addRecipe(recipe: Recipe) {

    this.recipes.push(recipe);

    this.recipesChanged.next(this.recipes.slice());

  }

  updateRecipe(index: number, newRecipe: Recipe) {

    this.recipes[index] = newRecipe;

    this.recipesChanged.next(this.recipes.slice());

  }

  deleteRecipe(index: number) {

    this.recipes.splice(index, 1);

    this.recipesChanged.next(this.recipes.slice());

  }

recipe-list.component.ts

ngOnInit() {

    this.subscription = this.recipeService.recipesChanged

      .subscribe(

        (recipes: Recipe[]) => {

          this.recipes = recipes;

        }

      );

    this.recipes = this.recipeService.getRecipes();

  }

18. Adding a Delete and Clear (Cancel) Functionality

Recipe-detail

onDeleteRecipe() {

    this.recipeService.deleteRecipe(this.id);

    this.router.navigate(['/recipes']);

  }

Recipe-edit

onCancel() {

    this.router.navigate(['../'], {relativeTo: this.route});

  }

19. Redirecting the User (after Deleting a Recipe)

20. Adding an Image Preview

Thêm local ref imagePath; truy cập imagePath.value

<div class="row">

        <div class="col-xs-12">

          <div class="form-group">

            <label for="imagePath">Image URL</label>

            <input

              type="text"

              id="imagePath"

              formControlName="imagePath"

              class="form-control"

              #imagePath>

          </div>

        </div>

      </div>

      <div class="row">

        <div class="col-xs-12">

          <img [src]="imagePath.value" class="img-responsive">

        </div>

      </div>

21. Providing the Recipe Service Correctly

Khai báo service trong module để đảm bảo có 1 instance được tạo

22. Deleting Ingredients and Some Finishing Touches

recipe-edit.component

onDeleteIngredient(index: number) {

    (<FormArray>this.recipeForm.get('ingredients')).removeAt(index);

  }

Thêm recipe-list.component

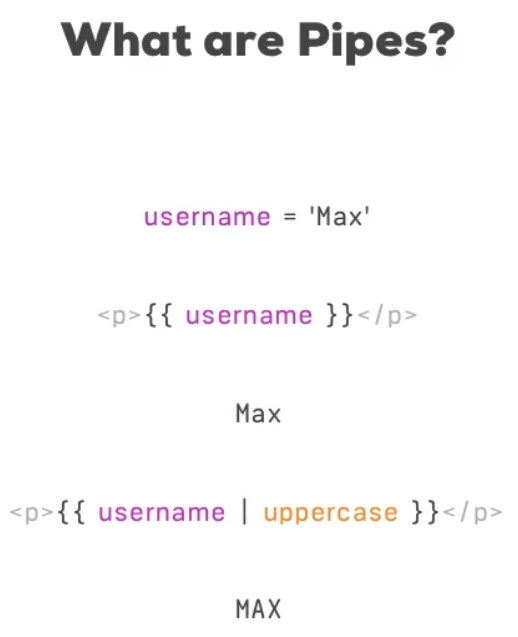
ngOnDestroy() {

    this.subscription.unsubscribe();

  }

## 17. Using Pipes to Transform Output

Pipe: Transform output value to out template



<ul class="list-group">

        <li

          class="list-group-item"

          \*ngFor="let server of servers"

          [ngClass]="getStatusClasses(server)">

          <span

            class="badge">

            {{ server.status }}

          </span>

          <strong>{{ server.name }}</strong> | {{ server.instanceType }} | {{ server.started }}

        </li>

      </ul>

Thay đổi

<strong>{{ server.name | shorten:15 }}</strong> |

          {{ server.instanceType | uppercase }} |

          {{ server.started | date:'fullDate' | uppercase }}

3. Parametrizing Pipes

date:'fullDate'

<https://angular.io/guide/pipes>

<https://angular.io/api?query=pipe>

5. Chaining Multiple Pipes

Lưu ý thứ tự từ trái qua phải

6. Creating a Custom Pipe

Tạo file shorten.pipe.ts

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({

  name: 'shorten'

})

export class ShortenPipe implements PipeTransform {

  transform(value: any, limit: number) {

    if (value.length > limit) {

      return value.substr(0, limit) + ' ...';

    }

    return value;

  }

}

Khai báo

declarations: [

    AppComponent,

    ShortenPipe,

    FilterPipe

  ],

8. Example Creating a Filter Pipe

      <input type="text" [(ngModel)]="filteredStatus">

Gõ

Ng generate pipe filter

Ng g p filter

filter.pipe.ts

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({

  name: 'filter',

  pure: false

})

export class FilterPipe implements PipeTransform {

  transform(value: any, filterString: string, propName: string): any {

    if (value.length === 0 || filterString === '') {

      return value;

    }

    const resultArray = [];

    for (const item of value) {

      if (item[propName] === filterString) {

        resultArray.push(item);

      }

    }

    return resultArray;

  }

}

<li

          class="list-group-item"

          \*ngFor="let server of servers | filter:filteredStatus:'status'"

          [ngClass]="getStatusClasses(server)">

9. Pure and Impure Pipes (or How to fix the Filter Pipe)

Khi thêm mới filter sẽ không update nên cần thêm pure: false nó sẽ tính toán lại khi bất cứ thay đổi nào happen => affect performance

@Pipe({

  name: 'filter',

  pure: false

})

10. Understanding the async Pipe

appStatus = new Promise((resolve, reject) => {

    setTimeout(() => {

      resolve('stable');

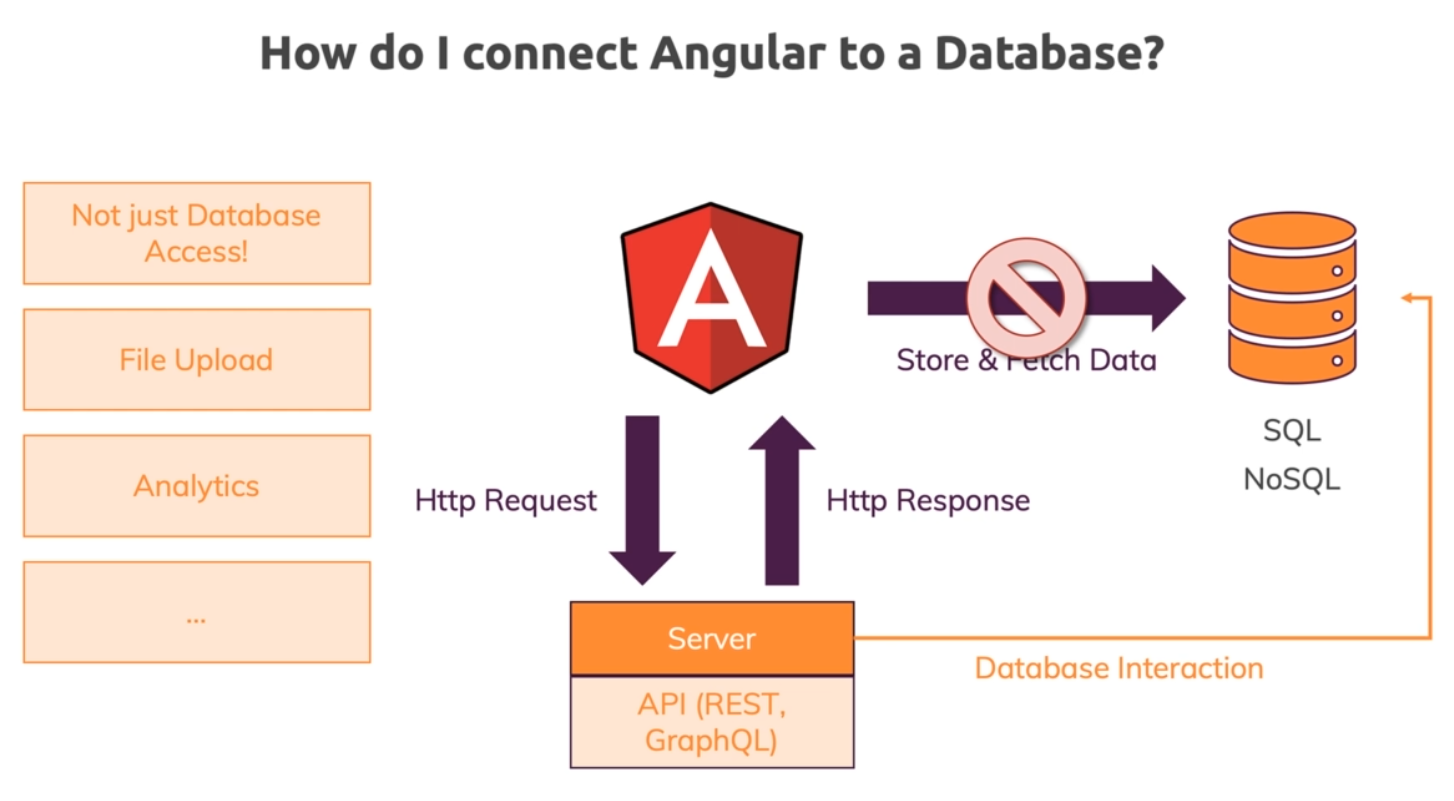
    }, 2000);

  });

      <h2>App Status: {{ appStatus | async}}</h2>

Để angular listen change of obj cần pipe để transform data easier

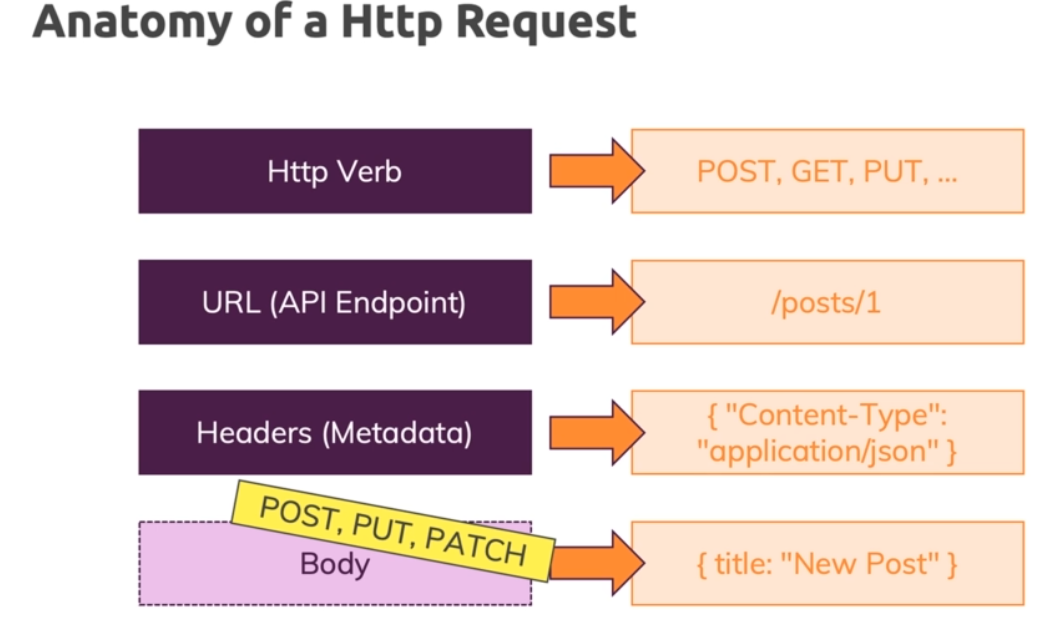
## 18. Making Http Requests



<https://academind.com/learn/javascript/hide-javascript-code/>

<https://academind.com/learn/node-js/building-a-restful-api-with/>

4. The Anatomy of a Http Request



5. Backend (Firebase) Setup

Vào firebase/ Go to the console/ Create a project/

Vào mục database/ real time db/ Chọn start in test mode

6. Sending a POST Request

6.1 http-02-post-requests.zip

  constructor(private http: HttpClient) {}

onCreatePost(postData: { title: string; content: string }) {

*// Send Http request*

    this.http

      .post(

        'https://ng-complete-guide-c56d3.firebaseio.com/posts.json',

        postData

      )

      .subscribe(responseData => {

        console.log(responseData);

      });

  }

Import HttpClientModule

@NgModule({

  declarations: [AppComponent],

  imports: [BrowserModule, FormsModule, HttpClientModule],

  providers: [],

  bootstrap: [AppComponent]

})

F12 vào tab network để check, hàm post return a Observable

7. GETting Data

Fetch data làm như post

8. Using RxJS Operators to Transform Response Data

private fetchPosts() {

    this.http

      .get('https://ng-complete-guide-c56d3.firebaseio.com/posts.json')

      .pipe(

        map(responseData => {

          const postsArray = [];

          for (const key in responseData) {

            if (responseData.hasOwnProperty(key)) {

              postsArray.push({ ...responseData[key], id: key });

            }

          }

          return postsArray;

        })

      )

      .subscribe(posts => {

*// ...*

        console.log(posts);

      });

  }

9. Using Types with the HttpClient

.get<{ [key: string]: Post }>

// nghĩa là 1 key có kiểu string sẽ có gtri là Post

10. Outputting Posts

 <p \*ngIf="loadedPosts.length < 1 && !isFetching">No posts available!

</p>

11. Showing a Loading Indicator

12. Using a Service for Http Requests

fetchPosts() {

    return this.http

      .get<{ [key: string]: Post }>(

        'https://ng-complete-guide-c56d3.firebaseio.com/posts.json'

      )

      .pipe(

        map(responseData => {

          const postsArray: Post[] = [];

          for (const key in responseData) {

            if (responseData.hasOwnProperty(key)) {

              postsArray.push({ ...responseData[key], id: key });

            }

          }

          return postsArray;

        }),

        catchError(errorRes => {

*// Send to analytics server*

          return throwError(errorRes);

        })

      );

  }

constructor(private http: HttpClient, private postsService:

PostsService) {}

  onFetchPosts() {

*// Send Http request*

    this.isFetching = true;

    this.postsService.fetchPosts().subscribe(

      posts => {

        this.isFetching = false;

        this.loadedPosts = posts;

      },

      error => {

        this.error = error.message;

        console.log(error);

      }

    );

  }

14. Sending a DELETE Request

onClearPosts() {

*// Send Http request*

    this.postsService.deletePosts().subscribe(() => {

      this.loadedPosts = [];

    });

  }

15. Handling Errors

Vào rule chỉnh để test

this.postsService.fetchPosts().subscribe(

      posts => {

        this.isFetching = false;

        this.loadedPosts = posts;

      },

      error => {

        this.error = error.message;

        console.log(error);

      }

    );

16. Using Subjects for Error Handling

  error = new Subject<string>();

// hàm post

this.http

      .post<{ name: string }>(

        'https://ng-complete-guide-c56d3.firebaseio.com/posts.json',

        postData

      )

      .subscribe(

        responseData => {

          console.log(responseData);

        },

        error => {

          this.error.next(error.message);

        }

      );

File angular

 ngOnInit() {

    this.errorSub = this.postsService.error.subscribe(errorMessage => {

      this.error = errorMessage;

    });

Nhớ destroy description

17. Using the catchError Operator

this.http

      .get<{ [key: string]: Post }>(

        'https://ng-complete-guide-c56d3.firebaseio.com/posts.json'

      )

      .pipe(

        map(responseData => {

          const postsArray: Post[] = [];

          for (const key in responseData) {

            if (responseData.hasOwnProperty(key)) {

              postsArray.push({ ...responseData[key], id: key });

            }

          }

          return postsArray;

        }),

        catchError(errorRes => {

*// Send to analytics server*

          return throwError(errorRes);

        })

      );

18. Error Handling & UX

onHandleError() {

    this.error = null;

  }

19. Setting Headers

this.http

      .get<{ [key: string]: Post }>(

        'https://ng-complete-guide-c56d3.firebaseio.com/posts.json',

        {

          headers: new HttpHeaders({ 'Custom-Header': 'Hello' })

        }

      )

20. Adding Query Params

Có thể xài hàm set để trực tiếp

let searchParams = new HttpParams();

    searchParams = searchParams.append('print', 'pretty');

    searchParams = searchParams.append('custom', 'key');

    return this.http

      .get<{ [key: string]: Post }>(

        'https://ng-complete-guide-c56d3.firebaseio.com/posts.json',

        {

          headers: new HttpHeaders({ 'Custom-Header': 'Hello' }),

          params: searchParams

        }

      )

21. Observing Different Types of Responses

Có thể observe là body để chỉ nhận body, response để xem nhiều thứ khác như stt code

.post<{ name: string }>(

        'https://ng-complete-guide-c56d3.firebaseio.com/posts.json',

        postData,

        {

          observe: 'response'

        }

      )

deletePosts() {

    return this.http

      .delete('https://ng-complete-guide-c56d3.firebaseio.com/posts.json', {

        observe: 'events',

        responseType: 'text'

      })

      .pipe(

        tap(event => {

          console.log(event);

          if (event.type === HttpEventType.Sent) {

*// ...*

          }

          if (event.type === HttpEventType.Response) {

            console.log(event.body);

          }

        })

      );

  }

22. Changing the Response Body Type

Thêm

        responseType: 'text' hay json

.delete('https://ng-complete-guide-c56d3.firebaseio.com/posts.json', {

        observe: 'events',

        responseType: 'text'

      })

23. Introducing Interceptors

auth-interceptor.service

import { HttpInterceptor, HttpRequest, HttpHandler } from '@angular/co

mmon/http';

export class AuthInterceptorService implements HttpInterceptor {

  intercept(req: HttpRequest<any>, next: HttpHandler) {

    console.log('Request is on its way');

    return next.handle(req);

  }

}

Khi req leave app

Khai báo:

providers: [

    {

      provide: HTTP\_INTERCEPTORS,

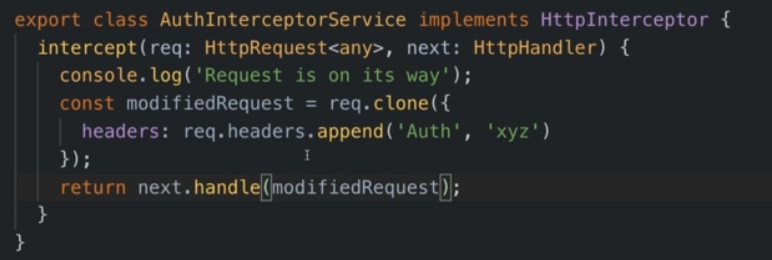
      useClass: AuthInterceptorService,

      multi: true

    }

  ],

24. Manipulating Request Objects



Khi muốn thay đổi request obj

25. Response Interceptors

Có thể thay đổi response



26. Multiple Interceptors

Tạo file logging-intercepter.service.ts

## 19. Course Project - Http

3. Setting Up the DataStorage Service

data-storage.service.ts ở header active 2 button

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

import { HttpClient } from '@angular/common/http';

import { map } from 'rxjs/operators';

import { Recipe } from '../recipes/recipe.model';

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

@Injectable({ providedIn: 'root' })

export class DataStorageService {

  constructor(private http: HttpClient, private recipeService: RecipeService) {}

  storeRecipes() {

    const recipes = this.recipeService.getRecipes();

    this.http

      .put(

        'https://ng-course-recipe-book-65f10.firebaseio.com/recipes.json',

        recipes

      )

      .subscribe(response => {

        console.log(response);

      });

  }

  fetchRecipes() {

    this.http

      .get<Recipe[]>(

        'https://ng-course-recipe-book-65f10.firebaseio.com/recipes.json'

      )

      .pipe(

        map(recipes => {

          return recipes.map(recipe => {

            return {

              ...recipe,

              ingredients: recipe.ingredients ? recipe.ingredients : []

            };

          });

        })

      )

      .subscribe(recipes => {

        this.recipeService.setRecipes(recipes);

      });

  }

}

Nhớ import HttpClientModule

recipes-resolver.service

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

import {

  Resolve,

  ActivatedRouteSnapshot,

  RouterStateSnapshot

} from '@angular/router';

import { Recipe } from './recipe.model';

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

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

@Injectable({ providedIn: 'root' })

export class RecipesResolverService implements Resolve<Recipe[]> {

  constructor(

    private dataStorageService: DataStorageService,

    private recipesService: RecipeService

  ) {}

  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {

    const recipes = this.recipesService.getRecipes();

    if (recipes.length === 0) {

      return this.dataStorageService.fetchRecipes();

    } else {

      return recipes;

    }

  }

}

tap(recipes => {

          this.recipeService.setRecipes(recipes);

        })

children: [

      { path: '', component: RecipeStartComponent },

      { path: 'new', component: RecipeEditComponent },

      {

        path: ':id',

        component: RecipeDetailComponent,

        resolve: [RecipesResolverService]

      },

      {

        path: ':id/edit',

        component: RecipeEditComponent,

        resolve: [RecipesResolverService]

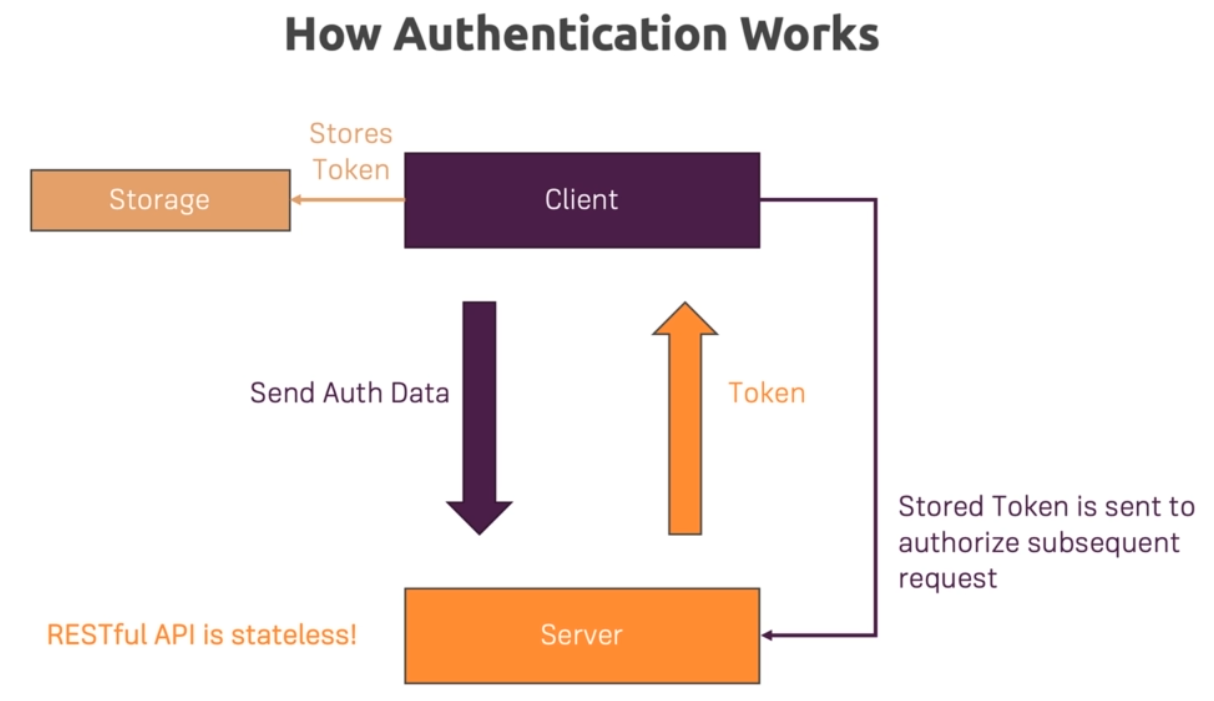
      }

    ]

## 20. Authentication & Route Protection in Angular

RestAPI là stateless vì vậy server không quan tâm đến client nên session không được sử dụng => token: string được mã hóa và giải mã bởi client, token được tạo ra ở server và mã hóa bằng thuật toán chỉ có server biết

Chỉ có server có thể validate incoming token



3. Adding the Auth Page

auth.component.html

<div class="row">

  <div class="col-xs-12 col-md-6 col-md-offset-3">

    <form>

      <div class="form-group">

        <label for="email">E-Mail</label>

        <input type="email" id="email" class="form-control" />

      </div>

      <div class="form-group">

        <label for="password">Password</label>

        <input type="password" id="password" class="form-control" />

      </div>

      <div>

        <button class="btn btn-primary">Sign Up</button> |

        <button class="btn btn-primary">Switch to Login</button>

      </div>

    </form>

  </div>

</div>

auth.component.ts

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

@Component({

  selector: 'app-auth',

  templateUrl: './auth.component.html'

})

export class AuthComponent {}

Nhớ khai báo component

  { path: 'auth', component: AuthComponent }

Vào header thêm

<li routerLinkActive="active">

          <a routerLink="/auth">Authenticate</a>

        </li>

4. Switching Between Auth Modes

export class AuthComponent {

  isLoginMode = true;

  onSwitchMode() {

    this.isLoginMode = !this.isLoginMode;

  }

  onSubmit(form: NgForm) {

    console.log(form.value);

    form.reset();

  }

}

File html

<button

          class="btn btn-primary"

          type="submit"

          [disabled]="!authForm.valid"

        >

          {{ isLoginMode ? 'Login' : 'Sign Up' }}

        </button>

        |

        <button class="btn btn-primary" (click)="onSwitchMode()" type="button">

          Switch to {{ isLoginMode ? 'Sign Up' : 'Login' }}

        </button>

5. Handling Form Input

    <form #authForm="ngForm" (ngSubmit)="onSubmit(authForm)">

Thêm

ngModel

name="email"

6. Preparing the Backend

8. Preparing the Signup Request

Project setting copy web api key paste vào url

auth.service.ts

interface AuthResponseData {

  kind: string;

  idToken: string;

  email: string;

  refreshToken: string;

  expiresIn: string;

  localId: string;

}

@Injectable({ providedIn: 'root' })

export class AuthService {

  constructor(private http: HttpClient) {}

  signup(email: string, password: string) {

    return this.http

      .post<AuthResponseData>(

        'https://www.googleapis.com/identitytoolkit/v3/relyingparty/signupNewUser?key=AIzaSyDb0xTaRAoxyCgvaDF3kk5VYOsTwB\_3o7Y',

        {

          email: email,

          password: password,

          returnSecureToken: true

        }

      )

      .pipe(

        catchError(errorRes => {

          let errorMessage = 'An unknown error occurred!';

          if (!errorRes.error || !errorRes.error.error) {

            return throwError(errorMessage);

          }

          switch (errorRes.error.error.message) {

            case 'EMAIL\_EXISTS':

              errorMessage = 'This email exists already';

          }

          return throwError(errorMessage);

        })

      );

  }

}

Auth.ts

onSubmit(form: NgForm) {

    if (!form.valid) {

      return;

    }

    const email = form.value.email;

    const password = form.value.password;

    this.isLoading = true;

    if (this.isLoginMode) {

*// ...*

    } else {

      this.authService.signup(email, password).subscribe(

        resData => {

          console.log(resData);

          this.isLoading = false;

        },

        errorMessage => {

          console.log(errorMessage);

          this.error = errorMessage;

          this.isLoading = false;

        }

      );

    }

    form.reset();

  }

10. Adding a Loading Spinner & Error Handling Logic

<https://loading.io/css/>

Vào share tạo folder loading tạo component loading

<div \*ngIf="isLoading" style="text-align: center;">

      <app-loading-spinner></app-loading-spinner>

    </div>

    <form #authForm="ngForm" (ngSubmit)="onSubmit(authForm)" \*ngIf="!isLoading">

11. Improving Error Handling

Xem lại thêm

Hàm sign up

private handleError(errorRes: HttpErrorResponse) {

    let errorMessage = 'An unknown error occurred!';

    if (!errorRes.error || !errorRes.error.error) {

      return throwError(errorMessage);

    }

    switch (errorRes.error.error.message) {

      case 'EMAIL\_EXISTS':

        errorMessage = 'This email exists already';

        break;

      case 'EMAIL\_NOT\_FOUND':

        errorMessage = 'This email does not exist.';

        break;

      case 'INVALID\_PASSWORD':

        errorMessage = 'This password is not correct.';

        break;

    }

    return throwError(errorMessage);

  }

File auth.ts

signup(email: string, password: string) {

    return this.http

      .post<AuthResponseData>(

        'https://www.googleapis.com/identitytoolkit/v3/relyingparty/signupNewUser?key=AIzaSyDb0xTaRAoxyCgvaDF3kk5VYOsTwB\_3o7Y',

        {

          email: email,

          password: password,

          returnSecureToken: true

        }

      )

      .pipe(

        catchError(this.handleError),

        tap(resData => {

          this.handleAuthentication(

            resData.email,

            resData.localId,

            resData.idToken,

            +resData.expiresIn

          );

        })

      );

  }

12. Sending Login Requests

  registered?: boolean; // kiểu optional boolean

auth.component

    let authObs: Observable<AuthResponseData>;

rồi sửa lại để không lặp lại code

13. Login Error Handling

14. Creating & Storing the User Data

Tạo file user.model

export class User {

  constructor(

    public email: string,

    public id: string,

    private \_token: string,

    private \_tokenExpirationDate: Date

  ) {}

  get token() { // getter

    if (!this.\_tokenExpirationDate || new Date() > this.\_tokenExpirationDate) {

      return null;

    }

    return this.\_token;

  }

}

auth.service

  user = new BehaviorSubject<User>(null);

Thêm hàm tap: thực hiện action mà k change response

private handleAuthentication(

    email: string,

    userId: string,

    token: string,

    expiresIn: number

  ) {

    const expirationDate = new Date(new Date().getTime() + expiresIn \* 1000);

    const user = new User(email, userId, token, expirationDate);

    this.user.next(user);

  }

15. Reflecting the Auth State in the UI

authObs.subscribe(

      resData => {

        console.log(resData);

        this.isLoading = false;

        this.router.navigate(['/recipes']);

      },

      errorMessage => {

        console.log(errorMessage);

        this.error = errorMessage;

        this.isLoading = false;

      }

    );

Thêm button log out

Header

ngOnInit() {

    this.userSub = this.authService.user.subscribe(user => {

      this.isAuthenticated = !!user; // ? false: true

      console.log(!user);

      console.log(!!user);

    });

  }

Xem lại

## 21. Dynamic Components

2. Adding an Alert Modal Component

Vào folder share tại component alert

<div class="backdrop" (click)="onClose()"></div>

<div class="alert-box">

  <p>{{ message }}</p>

  <div class="alert-box-actions">

    <button class="btn btn-primary" (click)="onClose()">Close</button>

  </div>

</div>

@Component({

  selector: 'app-alert',

  templateUrl: './alert.component.html',

  styleUrls: ['./alert.component.css']

})

export class AlertComponent {

  @Input() message: string;

  @Output() close = new EventEmitter<void>();

  onClose() {

    this.close.emit();

  }

}

auth.component.html

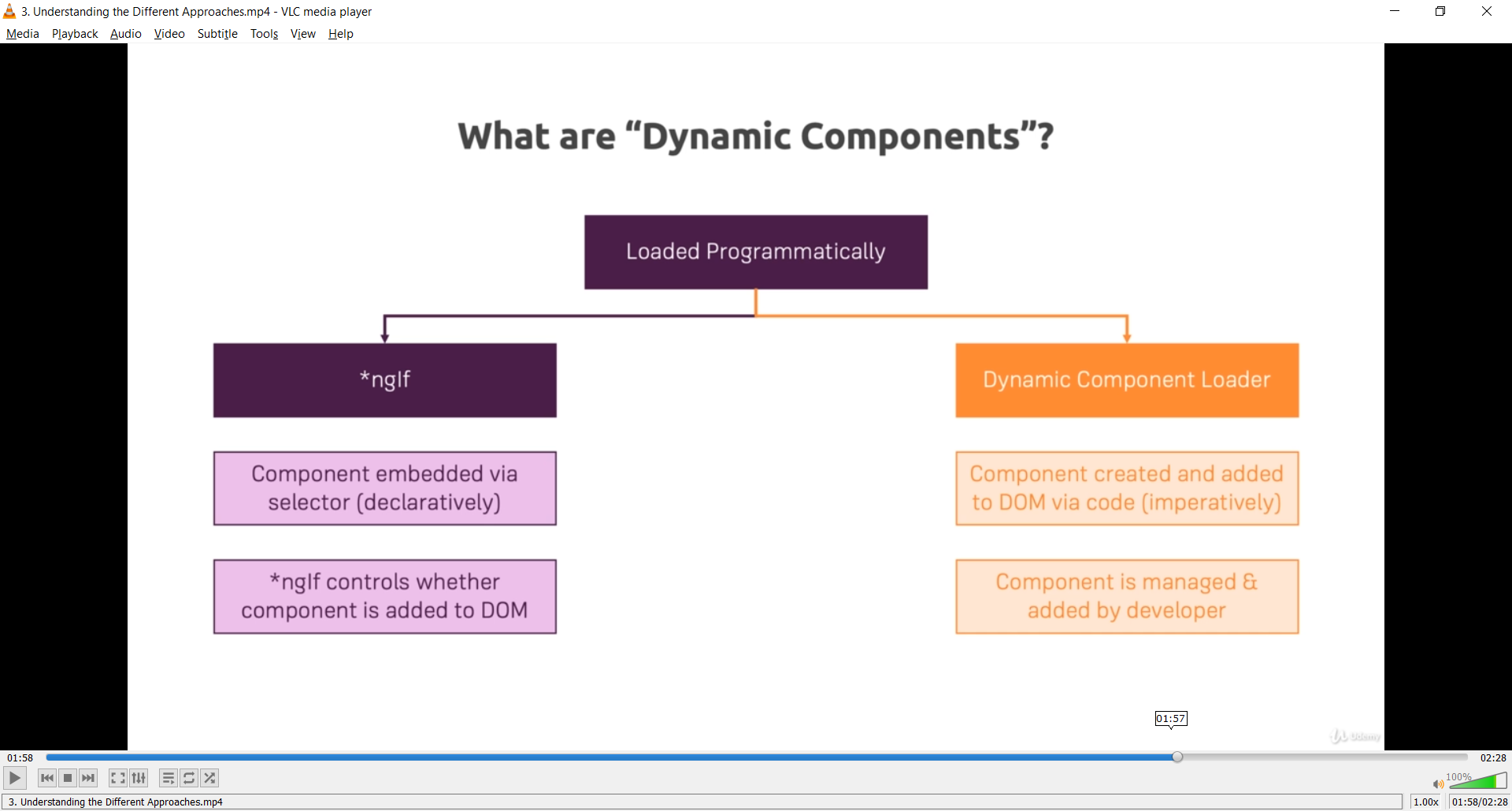
*<app-alert*

*[message]="error"*

*\*ngIf="error"*

*(close)="onHandleError()"*

*></app-alert>*



4. Using ngIf

5. Preparing Programmatic Creation

auth.component.ts

constructor(

    private authService: AuthService,

    private router: Router,

    private componentFactoryResolver: ComponentFactoryResolver

  ) {}

private showErrorAlert(message: string) {

*// const alertCmp = new AlertComponent();*

    const alertCmpFactory = this.componentFactoryResolver.resolveComponentFactory(

      AlertComponent

    );

    const hostViewContainerRef = this.alertHost.viewContainerRef;

    hostViewContainerRef.clear();

    const componentRef = hostViewContainerRef.createComponent(alertCmpFactory);

    componentRef.instance.message = message;

    this.closeSub = componentRef.instance.close.subscribe(() => {

      this.closeSub.unsubscribe();

      hostViewContainerRef.clear();

    });

  }

Tạo placeholder.directive.ts

import { Directive, ViewContainerRef } from '@angular/core';

@Directive({

  selector: '[appPlaceholder]'

})

export class PlaceholderDirective {

  constructor(public viewContainerRef: ViewContainerRef) {}

}

6. Creating a Component Programmatically

<ng-template appPlaceholder></ng-template>

auth.component.ts

  @ViewChild(PlaceholderDirective, { static: false }) alertHost: Place

holderDirective;

7. Understanding entryComponents

app.module

bootstrap: [AppComponent],

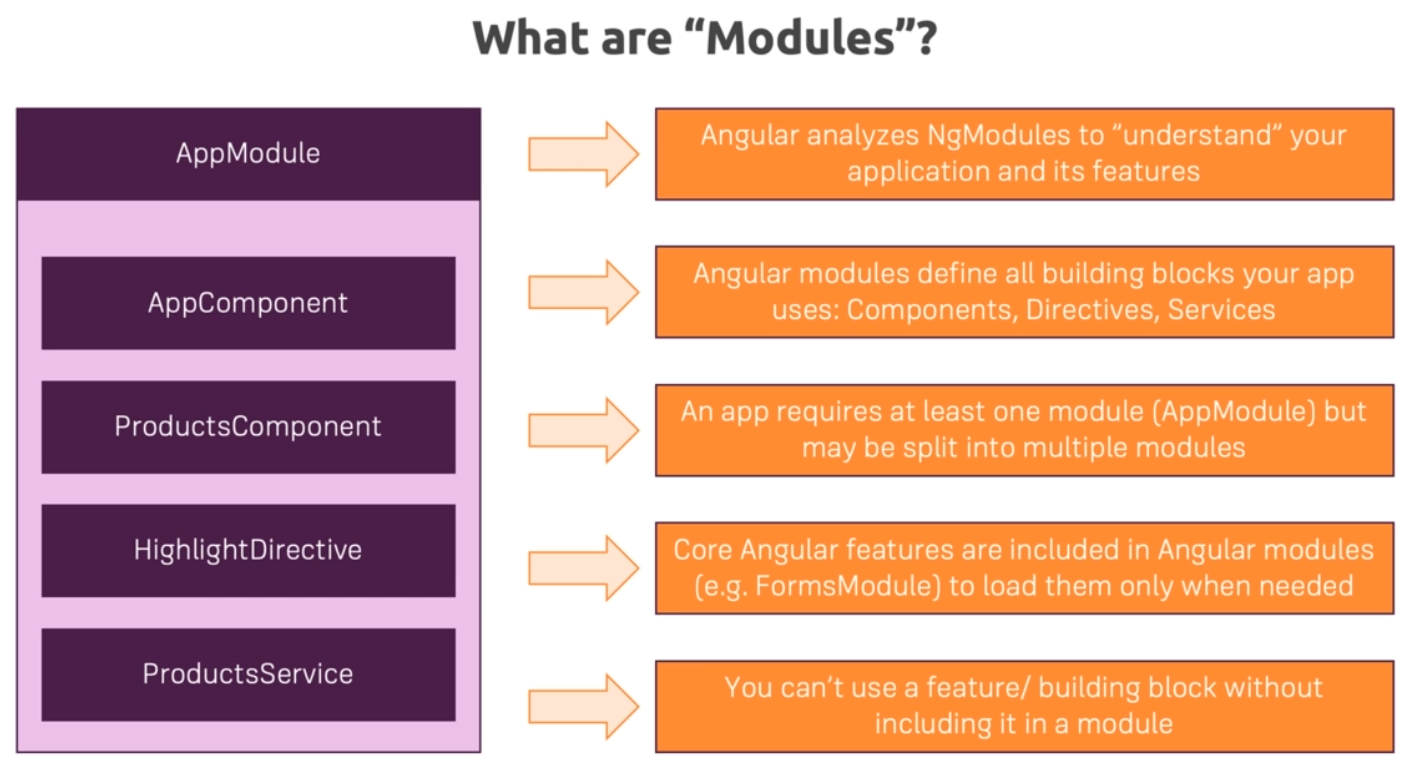
  entryComponents: [

    AlertComponent

  ]

8. Data Binding & Event Binding

## 22. Angular Modules & Optimizing Angular Apps



4. Getting Started with Feature Modules

Tạo file recipes.module

@NgModule({

  declarations: [

    RecipesComponent,

    RecipeListComponent,

    RecipeDetailComponent,

    RecipeItemComponent,

    RecipeStartComponent,

    RecipeEditComponent

  ],

  imports: [RouterModule, CommonModule, ReactiveFormsModule],

  exports: [

    RecipesComponent,

    RecipeListComponent,

    RecipeDetailComponent,

    RecipeItemComponent,

    RecipeStartComponent,

    RecipeEditComponent

  ]

})

export class RecipesModule {}

vào app

imports: [

    BrowserModule,

    FormsModule,

    ReactiveFormsModule,

    HttpClientModule,

    AppRoutingModule,

    RecipesModule

  ],

CommonModule fix lỗi cho ngIf, for

6. Adding Routes to Feature Modules

recipes-routing.module

imports: [

    RouterModule,

    CommonModule,

    ReactiveFormsModule,

    RecipesRoutingModule

  ],

7. Component Declarations

recipes.module

xóa export trong này

@NgModule({

  declarations: [

    RecipesComponent,

    RecipeListComponent,

    RecipeDetailComponent,

    RecipeItemComponent,

    RecipeStartComponent,

    RecipeEditComponent

  ],

  imports: [

    RouterModule,

    ReactiveFormsModule,

    RecipesRoutingModule,

    SharedModule

  ]

})

8. The ShoppingList Feature Module

@NgModule({

  declarations: [ShoppingListComponent, ShoppingEditComponent],

  imports: [

    FormsModule,

    RouterModule.forChild([

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

    ]),

    SharedModule

  ]

})

export class ShoppingListModule {}

9. Understanding Shared Modules

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

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

import { AlertComponent } from './alert/alert.component';

import { LoadingSpinnerComponent } from './loading-spinner/loading-spinner.component';

import { PlaceholderDirective } from './placeholder/placeholder.directive';

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

@NgModule({

  declarations: [

    AlertComponent,

    LoadingSpinnerComponent,

    PlaceholderDirective,

    DropdownDirective

  ],

  imports: [CommonModule],

  exports: [

    AlertComponent,

    LoadingSpinnerComponent,

    PlaceholderDirective,

    DropdownDirective,

    CommonModule

  ],

  entryComponents: [AlertComponent]

})

export class SharedModule {}

10. Understanding the Core Module

Sử dụng @Injectable({ providedIn: 'root' }) thay vì khai báo trong

app.modules

core.module.ts

@NgModule({

  providers: [

    ShoppingListService,

    RecipeService,

    {

      provide: HTTP\_INTERCEPTORS,

      useClass: AuthInterceptorService,

      multi: true

    }

  ]

})

export class CoreModule {}

11. Adding an Auth Feature Module

auth.module.ts

@NgModule({

  declarations: [AuthComponent],

  imports: [

    CommonModule,

    FormsModule,

    RouterModule.forChild([{ path: '', component: AuthComponent }]),

    SharedModule

  ]

})

export class AuthModule {}

12. Understanding Lazy Loading

Chỉ load những gì mình cần, xóa url ở file con

app-routing.module.ts

const appRoutes: Routes = [

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

  { path: 'recipes', loadChildren: './recipes/recipes.module#RecipesModule' },

  {

    path: 'shopping-list',

    loadChildren: './shopping-list/shopping-list.module#ShoppingListModule'

  },

  {

    path: 'auth',

    loadChildren: './auth/auth.module#AuthModule'

  }

];

recipes-routing.module.ts

path: '',

    component: RecipesComponent,

app.modules

@NgModule({

  declarations: [AppComponent, HeaderComponent],

  imports: [

    BrowserModule,

    HttpClientModule,

    AppRoutingModule,

    SharedModule,

    CoreModule

  ],

  bootstrap: [AppComponent]

})

14. Alternative Lazy Loading Syntax

If you're using Angular 8+, you can use an alternative syntax for specifying lazy-loaded routes:

Instead of

const routes: Routes = [{

path: 'your-path',

loadChildren: './your-module-path/module-name.module#ModuleName'

}];

you can use

const routes: Routes = [{

path: 'your-path',

loadChildren: () => import('./your-module-path/module-name.module').then(m => m.ModuleName)

}];

Please note, that you need to ensure that in your tsconfig.json file, you use

"module": "esnext",

instead of

"module": "es2015",

Why would you use this syntax? In the future, it'll replace the "string-only" approach (i.e. the first alternative mentioned here). It also will give you better IDE support.

15. More Lazy Loading

16. Preloading Lazy-Loaded Code

App-routing

@NgModule({

  imports: [

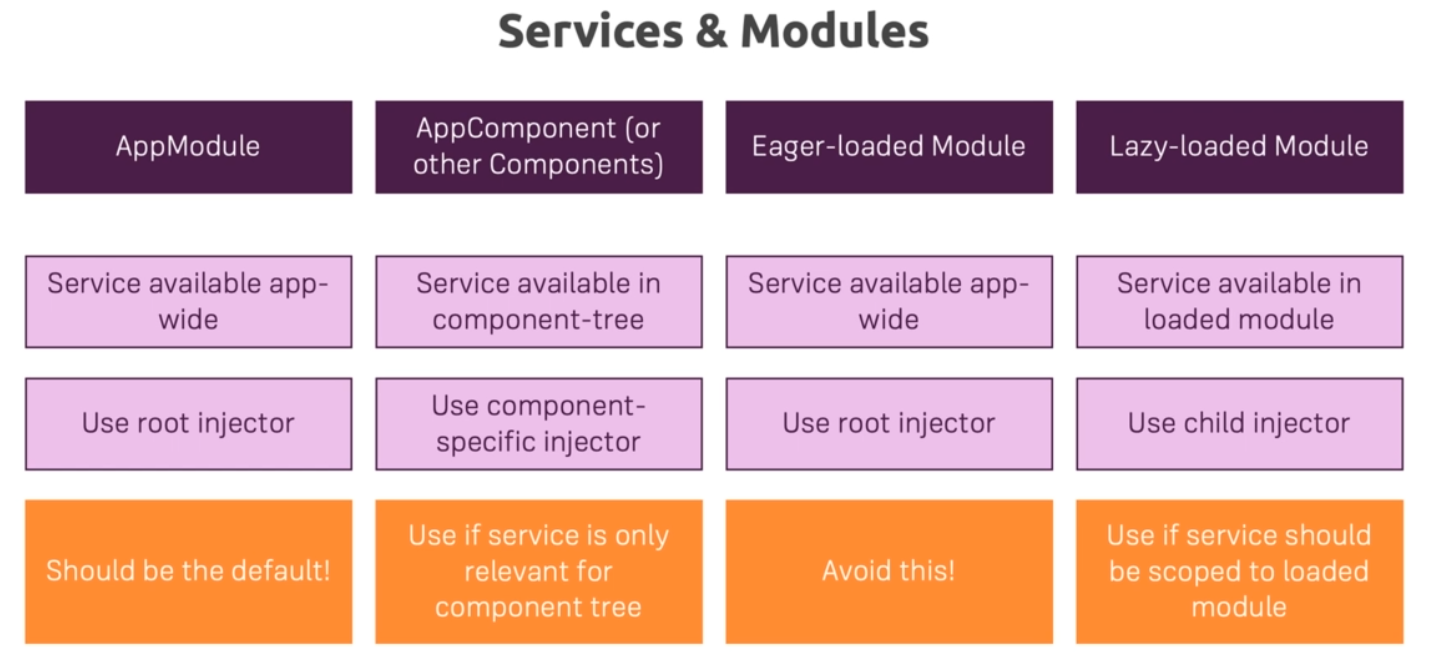
    RouterModule.forRoot(appRoutes, { preloadingStrategy: PreloadAllModules })

  ],

  exports: [RouterModule]

})

17. Modules & Services



18. Loading Services Differently

Tạo file logging.service.ts

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

*// @Injectable({ providedIn: 'root' })*

export class LoggingService {

  lastlog: string;

  printLog(message: string) {

    console.log(message);

    console.log(this.lastlog);

    this.lastlog = message;

  }

}

Thêm   *// providers: [LoggingService] ở module để test*

File shopping list module thì khác instance

19. Ahead-of-Time Compilation

Recipe-edit

get ingredientsControls() {

    return (this.recipeForm.get('ingredients') as FormArray).controls;

  }

Vào folder dist sau khi build

Ng build --prod



## 23. Deploying an Angular App

environment.ts,

environment.prod.ts

export const environment = {

  production: false,

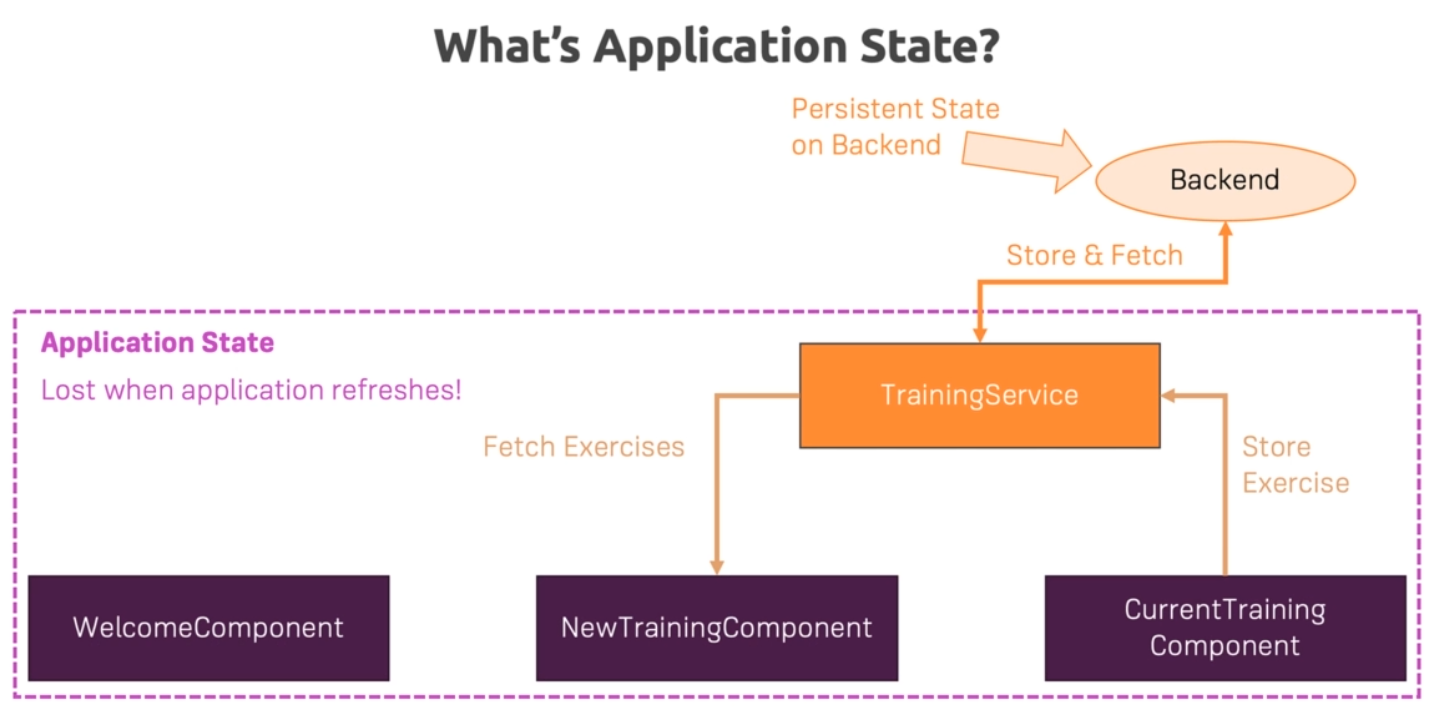
  firebaseAPIKey: 'AIzaSyDb0xTaRAoxyCgvaDF3kk5VYOsTwB\_3o7Y'

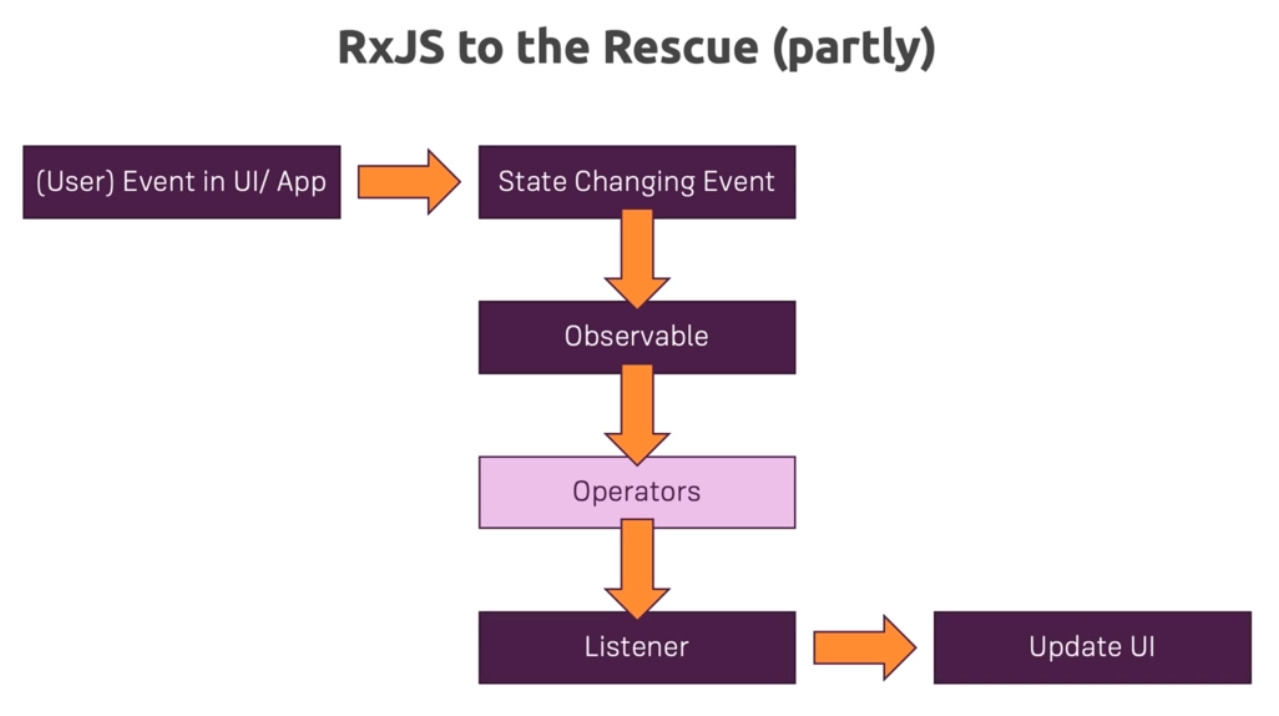
};

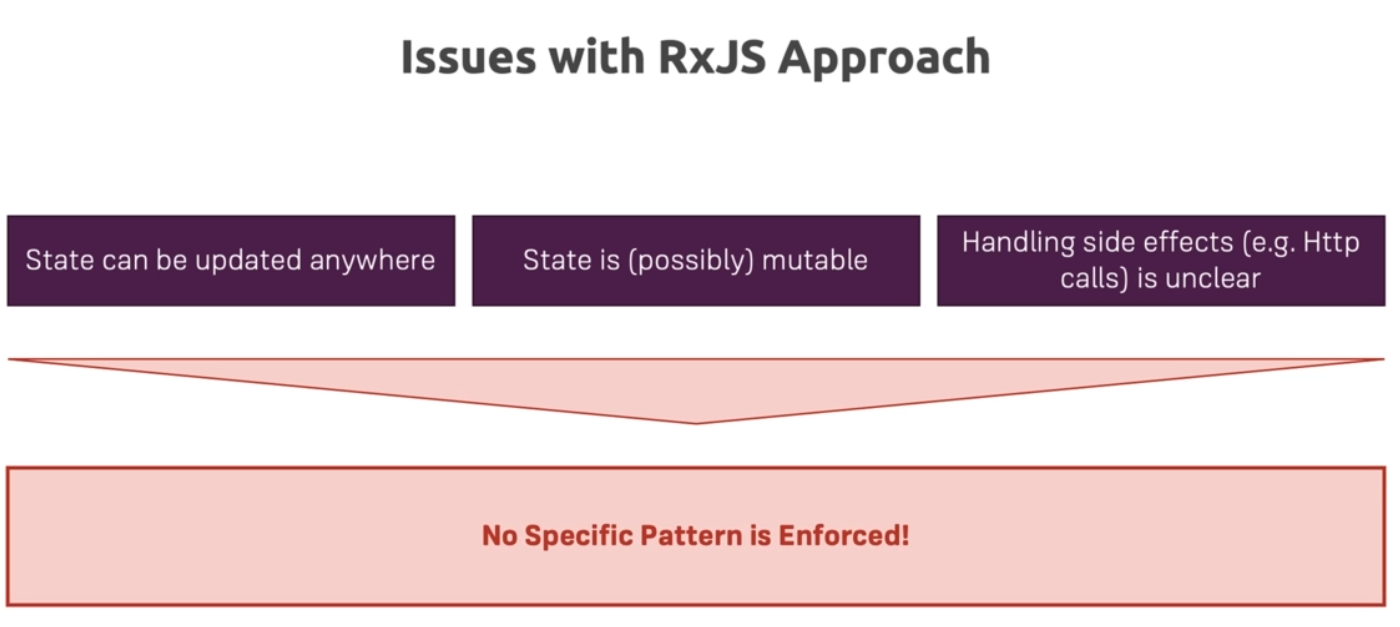
import { environment } from '../../environments/environment';

## 24. Bonus Working with NgRx in our Project

Nếu app lớn hơn các state phụ thuộc vào component, component phụ thuộc vào service

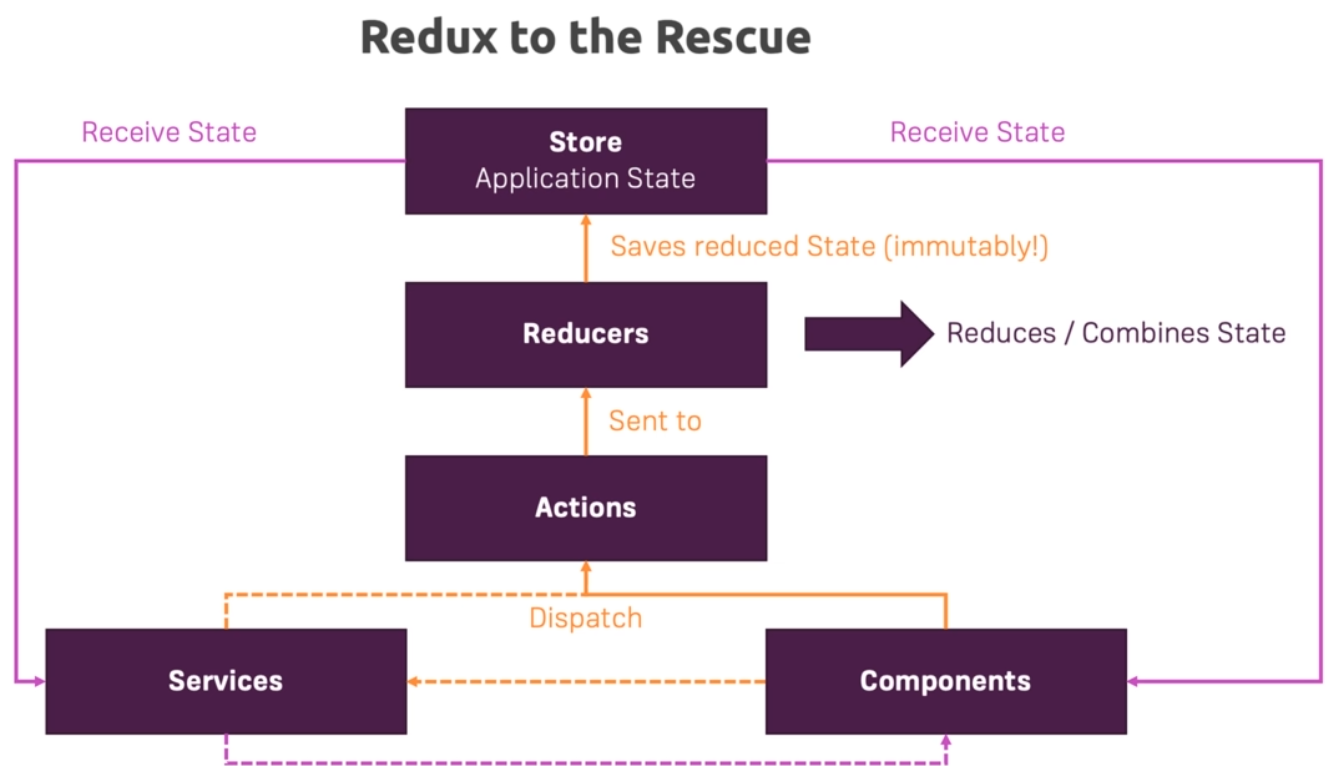


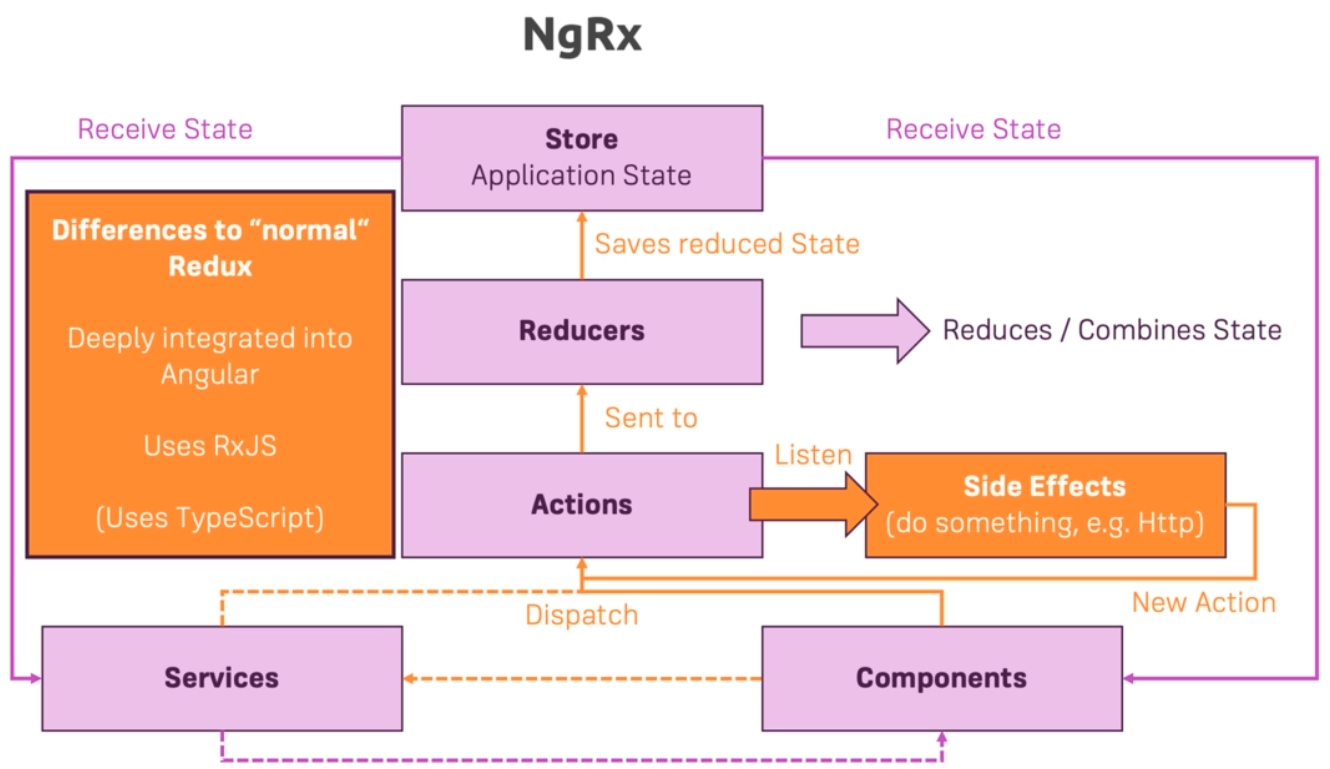




Redux là 1 js obj lớn gồm all data

Reducer là js func that get current state return a copy of old state





4. Getting Started with Reducers

Npm install –-save @ngrx/store

Tạo file shopping-list.reducer.ts

State is immutable

const initialState = {

  ingredients: [

    new Ingredient('Apples', 5),

    new Ingredient('Tomatoes', 10),

  ]

};

export function shoppingListReducer(state = initialState, action: Action) {

  switch (action.type) {

    case 'ADD\_INGREDIENT':

      return {

        ...state,

        ingredients: [...state.ingredients, action]

      };

  }

}

6. Understanding & Adding Actions

Tạo file shopping-list.actions

import { Action } from '@ngrx/store';

import { Ingredient } from '../../shared/ingredient.model';

export const ADD\_INGREDIENT = 'ADD\_INGREDIENT';

export class AddIngredient implements Action {

  readonly type = ADD\_INGREDIENT;

  payload: Ingredient; // riêng tên này có thể thay đổi trừ type

}

Khi export nhiều thứ từ 1 class

shopping-list.reducer

import { Ingredient } from '../../shared/ingredient.model';

import \* as ShoppingListActions from './shopping-list.actions';

const initialState = {

  ingredients: [new Ingredient('Apples', 5), new Ingredient('Tomatoes', 10)]

};

export function shoppingListReducer(

  state = initialState,

  action: ShoppingListActions.AddIngredient

) {

  switch (action.type) {

    case ShoppingListActions.ADD\_INGREDIENT:

      return {

        ...state,

        ingredients: [...state.ingredients, action.payload]

      };

    default:

      return state;

  }

}

7. Setting Up the NgRx Store

app.module.ts

import { StoreModule } from '@ngrx/store';

import { shoppingListReducer } from './shopping-list/store/shopping-list.reducer';

imports: [

    BrowserModule,

    HttpClientModule,

    AppRoutingModule,

    StoreModule.forRoot({shoppingList: shoppingListReducer}),

    SharedModule,

    CoreModule

  ],

So now all action will dispatch to reducer

8. Selecting State

shopping-list.component.ts

constructor(

    private slService: ShoppingListService,

    private loggingService: LoggingService,

    private store: Store<{ shoppingList: { ingredients: Ingredient[] } }> // thêm

  ) {}

  ingredients: Observable<{ ingredients: Ingredient[] }>;

ngOnInit() {

    this.ingredients = this.store.select('shoppingList');

*// this.ingredients = this.slService.getIngredients();*

*// this.subscription = this.slService.ingredientsChanged.subscribe(*

*//   (ingredients: Ingredient[]) => {*

*//     this.ingredients = ingredients;*

*//   }*

*// );*

    this.loggingService.printLog('Hello from ShoppingListComponent ngOnInit!');

  }

shopping-list.component.html

 <a

        class="list-group-item"

        style="cursor: pointer"

        \*ngFor="let ingredient of (ingredients | async).ingredients; let i = index"

        (click)="onEditItem(i)"

      >

(ingredients | async).ingredients bằng cách này nó sẽ subcribe ingredients observable

Reducer

export function shoppingListReducer(

  state = initialState,

  action: ShoppingListActions.ShoppingListActions

) {

  switch (action.type) {

    case ShoppingListActions.ADD\_INGREDIENT:

      return {

        ...state,

        ingredients: [...state.ingredients, action.payload]

      };

    case ShoppingListActions.ADD\_INGREDIENTS:

      return {

        ...state,

        ingredients: [...state.ingredients, ...action.payload]

      };

    default:

      return state;

  }

}

9. Dispatching Actions

shopping-edit.component.ts

constructor(

    private slService: ShoppingListService,

    private store: Store<{ shoppingList: { ingredients: Ingredient[] } }>

  ) {}

onSubmit(form: NgForm) {

    const value = form.value;

    const newIngredient = new Ingredient(value.name, value.amount);

    if (this.editMode) {

      this.slService.updateIngredient(this.editedItemIndex, newIngredient);

    } else {

*// this.slService.addIngredient(newIngredient);*

      this.store.dispatch(new ShoppingListActions.AddIngredient(newIngredient));

    }

    this.editMode = false;

    form.reset();

  }

shopping-list.actions.ts

tạo

  constructor(public payload: Ingredient) {}

  action: ShoppingListActions.AddIngredient

10. Multiple Actions

Thêm action

export const ADD\_INGREDIENTS = 'ADD\_INGREDIENTS';

export class AddIngredients implements Action {

  readonly type = ADD\_INGREDIENTS;

  constructor(public payload: Ingredient[]) {}

}

export type ShoppingListActions = AddIngredient | AddIngredients;

Recipe.service

    private store: Store<{ shoppingList: { ingredients: Ingredient[] }

addIngredientsToShoppingList(ingredients: Ingredient[]) {

*// this.slService.addIngredients(ingredients);*

    this.store.dispatch(new ShoppingListActions.AddIngredients(ingredients));

  }

11. Preparing Update & Delete Actions

export class UpdateIngredient implements Action {

  readonly type = UPDATE\_INGREDIENT;

  constructor(public payload: Ingredient ) {}

}

export class DeleteIngredient implements Action {

  readonly type = DELETE\_INGREDIENT;

}

export type ShoppingListActions =

  | AddIngredient

  | AddIngredients

  | UpdateIngredient

  | DeleteIngredient;

12. Updating & Deleting Ingredients

case ShoppingListActions.UPDATE\_INGREDIENT:

      const ingredient = state.ingredients[state.editedIngredientIndex];

      const updatedIngredient = {

        ...ingredient,

        ...action.payload

      };

      const updatedIngredients = [...state.ingredients];

      updatedIngredients[state.editedIngredientIndex] = updatedIngredient;

      return {

        ...state,

        ingredients: updatedIngredients,

        editedIngredientIndex: -1,

        editedIngredient: null

      };

    case ShoppingListActions.DELETE\_INGREDIENT:

      return {

        ...state,

        ingredients: state.ingredients.filter((ig, igIndex) => {

          return igIndex !== state.editedIngredientIndex;

        }),

        editedIngredientIndex: -1,

        editedIngredient: null

      };

Hàm filter trả về arr copy

13. Expanding the State

Reducer

editedIngredient: null,

  editedIngredientIndex: -1

// thêm state in reducer

export interface State {

  ingredients: Ingredient[];

  editedIngredient: Ingredient;

  editedIngredientIndex: number;

}

Tạo app.reducer.ts

import { ActionReducerMap } from '@ngrx/store';

import \* as fromShoppingList from '../shopping-list/store/shopping-list.reducer';

import \* as fromAuth from '../auth/store/auth.reducer';

export interface AppState {

  shoppingList: fromShoppingList.State;

  auth: fromAuth.State;

}

export const appReducer: ActionReducerMap<AppState> = {

  shoppingList: fromShoppingList.shoppingListReducer,

  auth: fromAuth.authReducer

};

Sau đó vào shopping-list,edit, service cập nhật lại kiểu cho store trong hàm tạo

    private store: Store<fromApp.AppState>

14. Managing More State via NgRx

export const START\_EDIT = 'START\_EDIT';

export const STOP\_EDIT = 'STOP\_EDIT';

shopping-list.component.ts

onEditItem(index: number) {

*// this.slService.startedEditing.next(index);*

    this.store.dispatch(new ShoppingListActions.StartEdit(index));

  }

shopping-edit.component.ts

ngOnInit() {

    this.subscription = this.store

      .select('shoppingList')

      .subscribe(stateData => {

        if (stateData.editedIngredientIndex > -1) {

          this.editMode = true;

          this.editedItem = stateData.editedIngredient;

// update form

          this.slForm.setValue({

            name: this.editedItem.name,

            amount: this.editedItem.amount

          });

        } else {

          this.editMode = false;

        }

      });

  }

onClear() {

    this.slForm.reset();

    this.editMode = false;

    this.store.dispatch(new ShoppingListActions.StopEdit());

  }

  onDelete() {

*// this.slService.deleteIngredient(this.editedItemIndex);*

    this.store.dispatch(new ShoppingListActions.DeleteIngredient());

    this.onClear();

  }

  ngOnDestroy() {

    this.subscription.unsubscribe();

    this.store.dispatch(new ShoppingListActions.StopEdit());

  }

15. Removing Redundant Component State Management

17. One Root State

auth.reducer.ts

import { User } from '../user.model';

export interface State {

  user: User;

}

const initialState: State = {

  user: null

};

export function authReducer(state = initialState, action) {

  return state;

}

app.module.ts import

    StoreModule.forRoot(fromApp.appReducer),

app.reducer.ts

import { ActionReducerMap } from '@ngrx/store';

import \* as fromShoppingList from '../shopping-list/store/shopping-list.reducer';

import \* as fromAuth from '../auth/store/auth.reducer';

export interface AppState {

  shoppingList: fromShoppingList.State;

  auth: fromAuth.State;

}

export const appReducer: ActionReducerMap<AppState> = {

  shoppingList: fromShoppingList.shoppingListReducer,

  auth: fromAuth.authReducer

};

18. Setting Up Auth Reducer & Actions

auth.actions

import { Action } from '@ngrx/store';

export const LOGIN = 'LOGIN';

export const LOGOUT = 'LOGOUT';

export class Login implements Action {

  readonly type = LOGIN;

  constructor(

    public payload: {

      email: string;

      userId: string;

      token: string;

      expirationDate: Date;

    }

  ) {}

}

export class Logout implements Action {

  readonly type = LOGOUT;

}

export type AuthActions = Login | Logout;

reducer

import { User } from '../user.model';

import \* as AuthActions from './auth.actions';

export interface State {

  user: User;

}

const initialState: State = {

  user: null

};

export function authReducer(

  state = initialState,

  action: AuthActions.AuthActions

) {

  switch (action.type) {

    case AuthActions.LOGIN:

      const user = new User(

        action.payload.email,

        action.payload.userId,

        action.payload.token,

        action.payload.expirationDate

      );

      return {

        ...state,

        user: user

      };

    case AuthActions.LOGOUT:

      return {

        ...state,

        user: null

      };

    default:

      return state;

  }

}

19. Dispatching Auth Actions

auth.service

constructor(

    private http: HttpClient,

    private router: Router,

    private store: Store<fromApp.AppState>

  ) {}

Sửa hàm auto login và logout

20. Auth Finished (For Now...)

auth-interceptor.service, auth.guard

header.component

constructor(

    private dataStorageService: DataStorageService,

    private authService: AuthService,

    private store: Store<fromApp.AppState>

  ) {}

  ngOnInit() {

    this.userSub = this.store

      .select('auth')

      .pipe(map(authState => authState.user))

      .subscribe(user => {

        this.isAuthenticated = !!user;

        console.log(!user);

        console.log(!!user);

      });

  }

21. And Important Note on Actions

Thêm prefix cho action

export const ADD\_INGREDIENT = '[Shopping List] Add Ingredient';

export const ADD\_INGREDIENTS = '[Shopping List] Add Ingredients';

export const UPDATE\_INGREDIENT = '[Shopping List] Update Ingredient';

export const DELETE\_INGREDIENT = '[Shopping List] Delete Ingredient';

export const START\_EDIT = '[Shopping List] Start Edit';

export const STOP\_EDIT = '[Shopping List] Stop Edit';

22. Exploring NgRx Effects

Npm install –-save @ngrx/effects

23. Defining the First Effect

Tạo file auth.effects.ts

  authLogin is a observable

export class AuthEffects {

  @Effect()

  authLogin = this.actions$.pipe(

    ofType(AuthActions.LOGIN\_START)

  );

  constructor(

    private actions$: Actions,

    private http: HttpClient,

    private router: Router

  ) {}

}

24. Effects & Error Handling

Action

export class LoginStart implements Action {

  readonly type = LOGIN\_START;

  constructor(public payload: { email: string; password: string }) {}

}

Hoàn thiện

  @Effect() // nhận biết là effect để handle

authLogin = this.actions$.pipe

Nó sẽ k bao giờ die trong suốt application

Hàm of() return non error observable

Khai báo app.module

    EffectsModule.forRoot([AuthEffects]),

26. Managing UI State in NgRx

export const LOGIN\_FAIL = '[Auth] Login Fail';

auth.component.ts

ngOnInit() {

    this.store.select('auth').subscribe(authState => {

      this.isLoading = authState.loading;

      this.error = authState.authError;

      if (this.error) {

        this.showErrorAlert(this.error);

      }

    });

  }

27. Finishing the Login Effect

@Effect({ dispatch: false })

  authSuccess = this.actions$.pipe(

    ofType(AuthActions.LOGIN),

    tap(() => {

      this.router.navigate(['/']);

    })

  );

28. Preparing Other Auth Actions

export const AUTHENTICATE\_SUCCESS = '[Auth] Login';

export const AUTHENTICATE\_FAIL = '[Auth] Login Fail';

export const SIGNUP\_START = '[Auth] Signup Start';

Sửa lại service và reducer

29. Adding Signup

30. Further Auth Effects

auth.component

onHandleError() {

    this.store.dispatch(new AuthActions.ClearError());

  }