



Giới thiệu về Angular

1. **Angular** là 1 framework của javascript được google phát triển để phát triển ứng dụng web.
2. **Angular** hỗ trợ đa nền tảng web lẫn di động (dùng Ionic build ra các ứng dụng di động).
3. Viết chủ yếu bằng code typescript (gần với oop hơn dễ tiếp cận).
4. Hướng tiếp cận dựa trên **component, module**.

Phiên bản mới Angular 9

1. **Angular 9** là phiên bản mới nhất của angular. Các phiên bản **Angular 2+** là các phiên bản tiếp theo của **angular js** (angular 1) nhưng có thêm 1 số khái niệm mới. (Nếu biết angular 1 thì học angular 2+ sẽ nhanh hơn tuy nhiên không biết cũng không sao chỉ cần nắm vững javascript là được)
2. **Angular 9** hỗ trợ bản **TypeScript 3.7**
3. **Angular 9** xuất hiện tính năng mới là **Ivy render engine**. Ivy là cơ chế render mới của Angular, giúp giảm kích thước bundle của ứng dụng.

Phiên bản mới Angular 9

4. **Angular 9** còn xuất hiện công cụ build mới là **Bazel**, giúp build nhanh hơn . Chỉ build và deploy những gì thay đổi không phải toàn bộ ứng dụng.
5. Để cập nhật phiên bản mới của angular cho các dự án cũ, chúng ta có thể tham khảo các hướng dẫn sau:

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

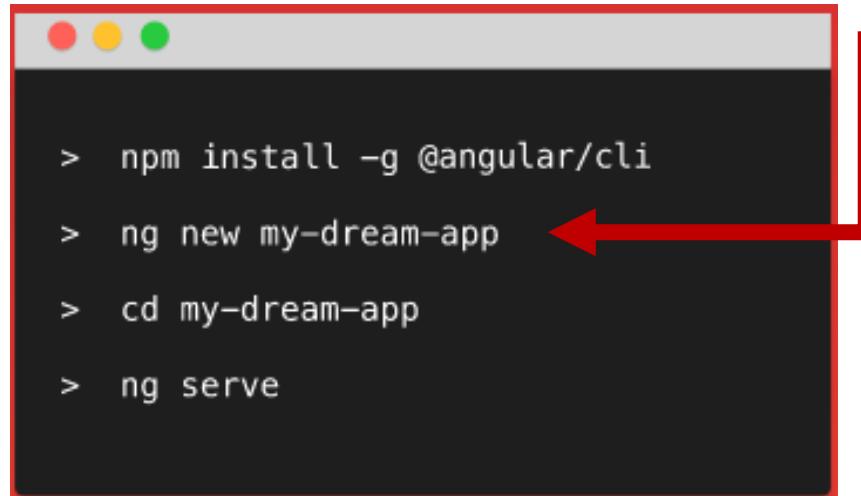
<https://update.angular.io/>

Một số khái niệm trong Angular

- Module
- Components
- Phân tách module (Nhóm component) – SharedModule
- Databinding
- Directives
- Input, Output, ViewChild,
- ViewChildren
- Hướng dẫn angular Material
- Template – Routing
- Form-Validation
- Pipes
- Service - Observable – Object – HTTP
- Get Post Put Delete (http service)
- Hướng dẫn xây dựng dự án
- Guard
- RXJS DOM
- Animation
- Khác ...

Cài đặt Angular CLI

1. Angular CLI là bộ công cụ do google cung cấp để phát triển ứng dụng bằng angular.
2. Angular Cli giúp tạo dự án, quản lý tập tin và thực hiện nhiều tác vụ khác nhau như test bundle và deploy dự án.



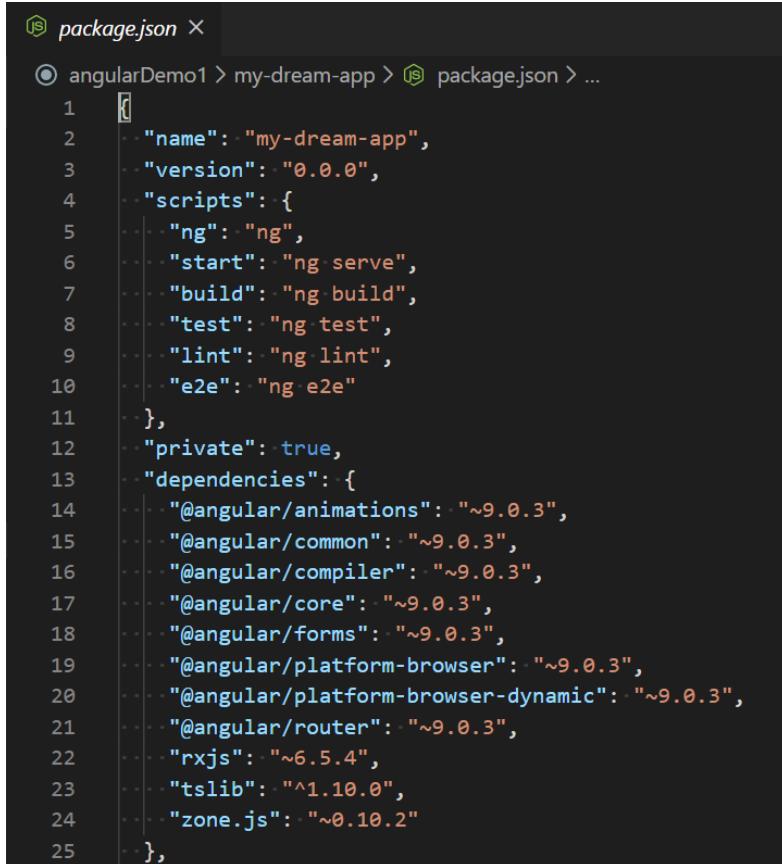
```
> npm install -g @angular/cli
> ng new my-dream-app ← Red arrow pointing to this line
> cd my-dream-app
> ng serve
```

A screenshot of a terminal window with a red border. It shows four command-line entries. The third entry, 'ng new my-dream-app', is highlighted with a red arrow pointing from the explanatory text below it.

**Cú pháp để tạo ứng dụng mới
my-dream-app** (Tên ứng dụng)

Trang angular: <https://angular.io/> (basic)
Trang aglcli: <https://cli.angular.io/>
Tài liệu:
<https://github.com/angular/angular-cli/wiki>

Giới thiệu về cấu trúc thư mục và tập tin



The screenshot shows a code editor window with the title "package.json" and a "package.json" icon. The path in the top bar is "angularDemo1 > my-dream-app > package.json > ...". The code itself is a JSON object representing the package configuration:

```
1  {
2    "name": "my-dream-app",
3    "version": "0.0.0",
4    "scripts": {
5      "ng": "ng",
6      "start": "ng serve",
7      "build": "ng build",
8      "test": "ng test",
9      "lint": "ng lint",
10     "e2e": "ng e2e"
11   },
12   "private": true,
13   "dependencies": {
14     "@angular/animations": "~9.0.3",
15     "@angular/common": "~9.0.3",
16     "@angular/compiler": "~9.0.3",
17     "@angular/core": "~9.0.3",
18     "@angular/forms": "~9.0.3",
19     "@angular/platform-browser": "~9.0.3",
20     "@angular/platform-browser-dynamic": "~9.0.3",
21     "@angular/router": "~9.0.3",
22     "rxjs": "~6.5.4",
23     "tslib": "^1.10.0",
24     "zone.js": "~0.10.2"
25   },
26 }
```

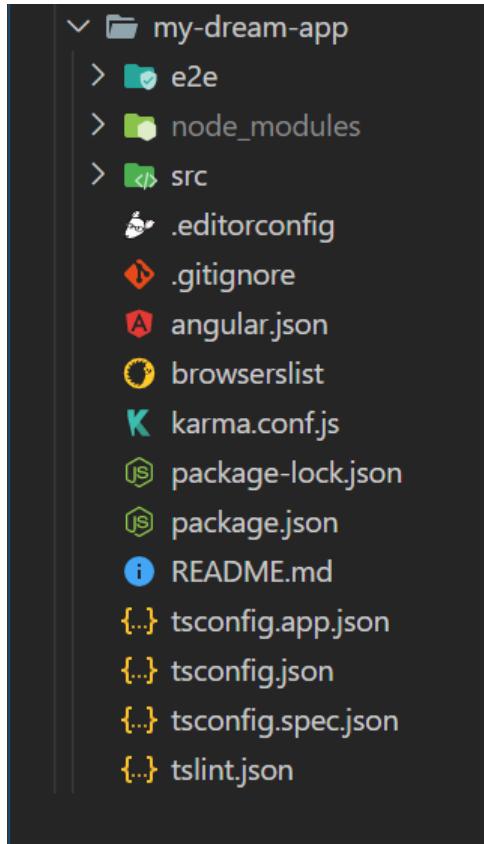
Tương tự typescript, package.json lưu trữ các thông tin của ứng dụng. Bao gồm cả những gói tin thư viện được cài sẵn trong ứng dụng. Trong đó chứa rất nhiều gói của angular 2 được cài sẵn.

Giới thiệu về cấu trúc thư mục và tập tin

```
tsconfig.json ×  
◎ angularDemo1 > my-dream-app > tsconfig.json > {} compilerOptions > [ ] lib  
1  {  
2    "compileOnSave": false,  
3    "compilerOptions": {  
4      "baseUrl": "./",  
5      "outDir": "./dist/out-tsc",  
6      "sourceMap": true,  
7      "declaration": false,  
8      "downlevelIteration": true,  
9      "experimentalDecorators": true,  
10     "module": "esnext",  
11     "moduleResolution": "node",  
12     "importHelpers": true,  
13     "target": "es2015",  
14     "typeRoots": [  
15       "node_modules/@types"  
16     ],  
17     "lib": [  
18       "es2018",  
19       "dom"  
20     ],  
21   },  
22   "angularCompilerOptions": {  
23     "fullTemplateTypeCheck": true,  
24     "strictInjectionParameters": true  
25   }  
26 }  
27 }
```

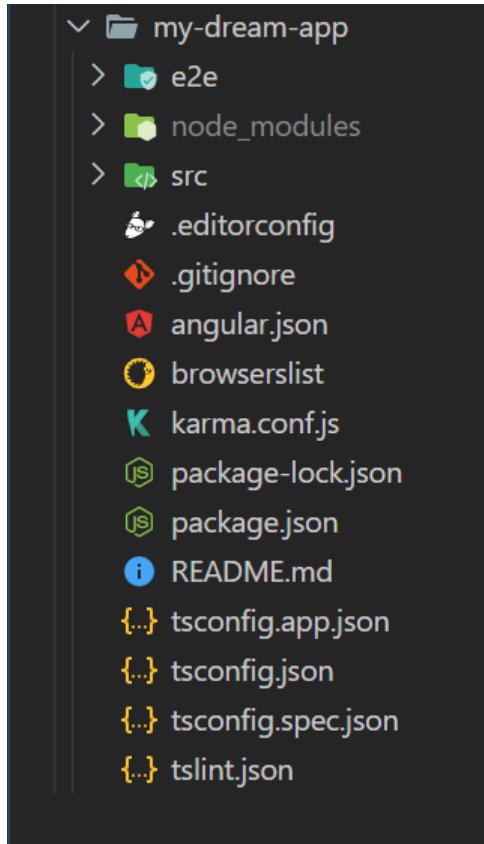
File này dùng để cấu hình biên dịch cú pháp
typescript => js

Giới thiệu về cấu trúc thư mục và tập tin



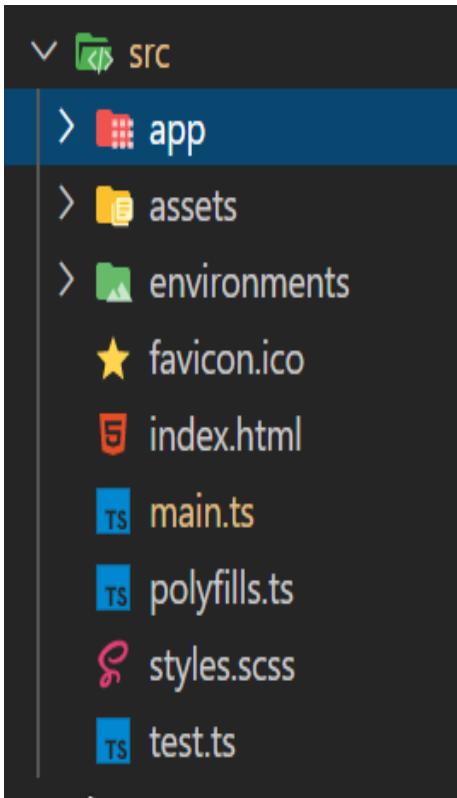
- e2e: Thư mục này dùng để chứa các tập tin cho testing
- node_modules: Chứa các module cần thiết cho ứng dụng
- src: Đây là thư mục sẽ chứa toàn bộ source code của ứng dụng
- .editorconfig: Chứa các cấu hình liên quan đến phần
- .gitignore: chứa thông tin những tập tin hoặc thư mục sẽ bị ignore không được commit lên Git Repository.

Giới thiệu về cấu trúc thư mục và tập tin



- angular.json: tập tin chứa cấu hình cho Angular CLI, giúp build ứng dụng Angular.
- browserslist: điều chỉnh CSS và JS để hỗ trợ cho nhiều trình duyệt đặc biệt (IE9-11, Egde 16...)
- karma.conf.js: Tập tin cấu hình cho Karma, liên quan nhiều đến phần testing
- package-lock.json: Dùng để lock version cho các module dependencies
- README.md: hiển thị thông tin về Git Repository.
- tslint.json: Tập tin cấu hình để kiểm tra lỗi cho các tập tin .ts (TypeScript) trong Angular project

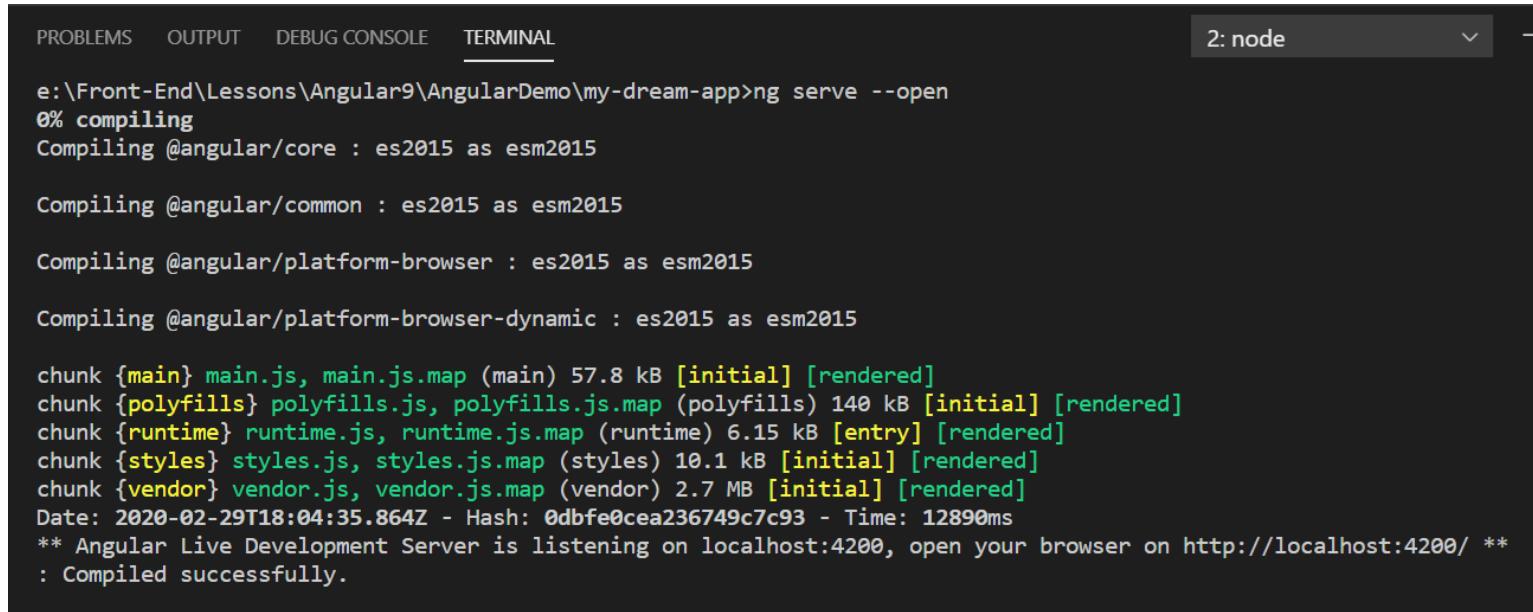
Giới thiệu về cấu trúc thư mục và tập tin



- App: Đây là thư mục sẽ chứa toàn bộ code của ứng dụng Angular.
- Assets: Thư mục này sẽ chứa các file ảnh, CSS, custom JavaScript của ứng dụng Angular
- Environments: thư mục giúp chúng ta định nghĩa các tập tin cấu hình cho những môi trường khác nhau đó.
- index.html: Trang chủ của ứng dụng Angular
- main.ts: Chứa code bootstrapping cho ứng dụng Angular
- polyfill.ts: định nghĩa các chuẩn giúp ứng dụng có thể chạy được trên mọi trình duyệt.
- style.css Định nghĩa style CSS cho ứng dụng Angular
- test.ts: Code để chạy test
- tsconfig.json: Tập tin định nghĩa việc compile cho TypeScript

Chạy thử ứng dụng angular đầu tiên

Sau khi tải và cài đặt thành công angular cli ta sử dụng cú pháp **ng serve --open** để project bắt đầu build và đóng gói tất cả các thư viện + source code (module component...)



The screenshot shows a terminal window with the following content:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
2: node +
```

```
e:\Front-End\Lessons\Angular9\AngularDemo\my-dream-app>ng serve --open
% compiling
Compiling @angular/core : es2015 as esm2015

Compiling @angular/common : es2015 as esm2015

Compiling @angular/platform-browser : es2015 as esm2015

Compiling @angular/platform-browser-dynamic : es2015 as esm2015

chunk {main} main.js, main.js.map (main) 57.8 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 140 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 10.1 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 2.7 MB [initial] [rendered]
Date: 2020-02-29T18:04:35.864Z - Hash: 0dbfe0cea236749c7c93 - Time: 12890ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
: Compiled successfully.
```

Nguyên tắc hoạt động của project angular cli

- Khi khởi động website nó sẽ chạy file **index.html** đầu tiên
- Angular tự động đóng gói file và thêm vào các file cần thiết trong quá trình khởi chạy, trong đó có **main.ts**

```
index.html ×  
1  <!doctype html>  
2  <html lang="en">  
3  <head>  
4  | <meta charset="utf-8">  
5  | <title>Phim</title>  
6  | <base href="/">  
7  
8  | <meta name="viewport" content="width=device-width, initial-scale=1">  
9  | <link rel="icon" type="image/x-icon" href="favicon.ico">  
10 </head>  
11 <body>  
12 | <app-root></app-root> ←  
13 </body>  
14 </html>
```

Component **<app-root>**
chứa tất cả nội dung của
toàn ứng dụng

Nguyên tắc hoạt động của project angular cli

- Khi đến main.ts sẽ gọi đến file app.module.ts

AngularDemo > my-dream-app > src > `main.ts` > ...

```
1 import { enableProdMode } from '@angular/core';
2 import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3
4 import { AppModule } from './app/app.module';
5 import { environment } from './environments/environment';
6
7 if (environment.production) {
8   enableProdMode();
9 }
10
11 platformBrowserDynamic().bootstrapModule(AppModule)
12   .catch(err => console.error(err));
13
```

Dòng này khai báo cho angular biết AppModule là module gốc quản lý toàn bộ ứng dụng web



Nguyên tắc hoạt động của project angular cli

- Khi đến main.ts sẽ gọi đến file app.module.ts

AngularDemo > my-dream-app > src > `main.ts` > ...

```
1 import { enableProdMode } from '@angular/core';
2 import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3
4 import { AppModule } from './app/app.module';
5 import { environment } from './environments/environment';
6
7 if (environment.production) {
8   enableProdMode();
9 }
10
11 platformBrowserDynamic().bootstrapModule(AppModule)
12   .catch(err => console.error(err));
13
```

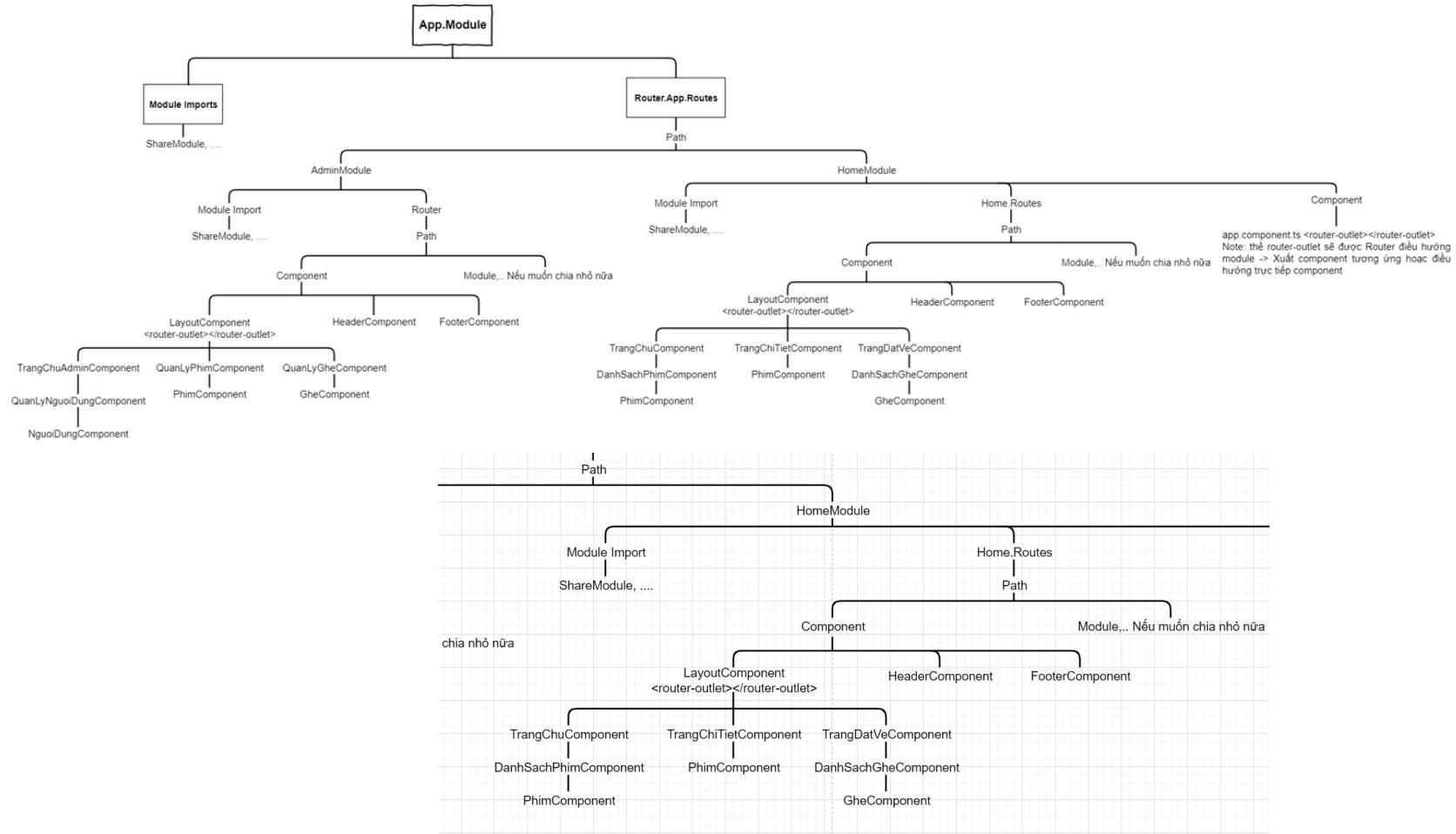
Dòng này khai báo cho angular biết AppModule là module gốc quản lý toàn bộ ứng dụng web



Module

1. Khái niệm về module

- Module là 1 Class dùng để đóng gói 1 chức năng cụ thể của ứng dụng.
- Có nhiều loại module ví dụ như:
 - BrowserModule: được sử dụng ở trên chứa tất cả các dependencies(các gói thư viện từ node_module sử dụng cho ứng dụng) cần thiết để chạy Angular trên trình duyệt.
 - HttpClientModule, FormModule, RoutingModule... (Ta sẽ tìm hiểu ở các phần sau).
- Nếu xét về mối quan hệ giữa module và component thì module giống như 1 group của component quản lý các component. 1 Module có thể quản lý nhiều component và mỗi component phải được quản lý bởi 1 module nào đó.
- app.module là nơi bắt đầu khởi chạy của ứng dụng. Nó gọi là module gốc và nó chứa 1 component gốc là app.component

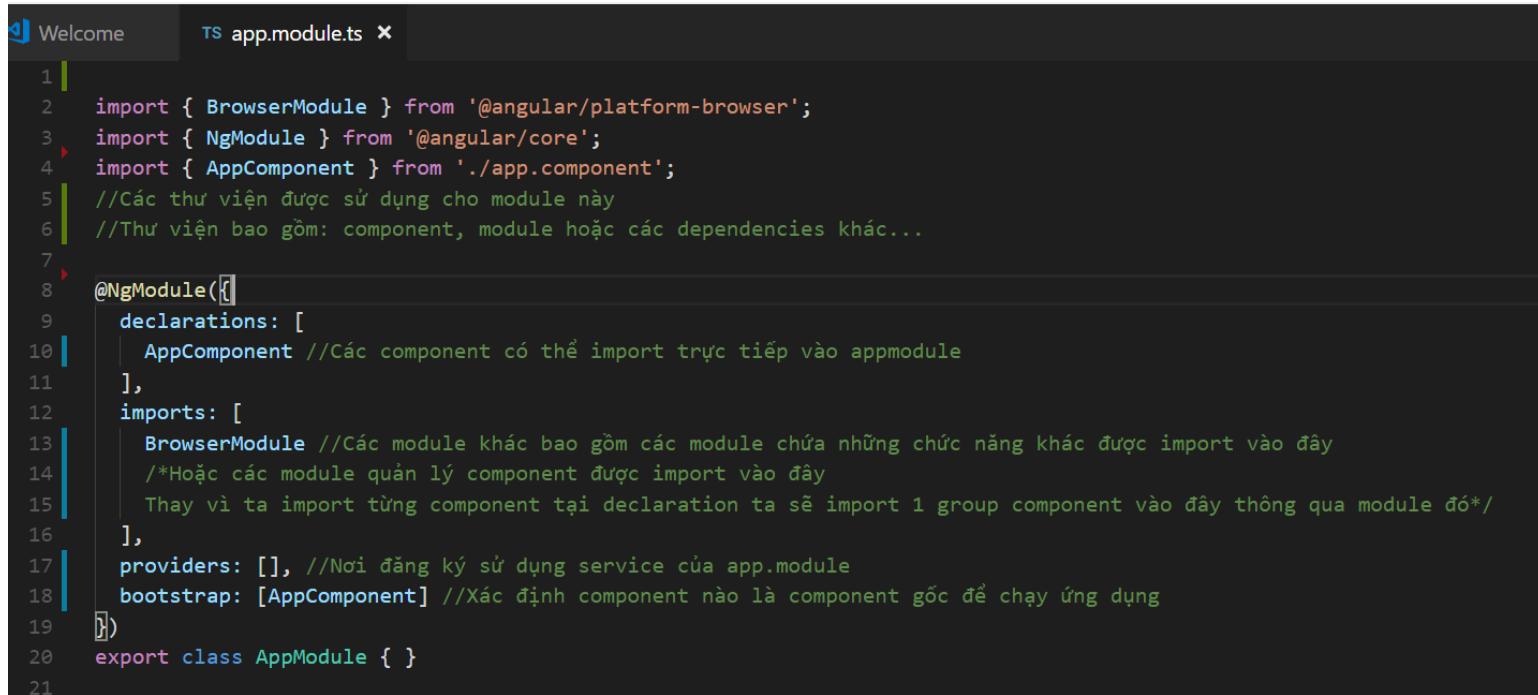


Module

2. Ví dụ về app.module.ts (module chính trên toàn ứng dụng)

Cú pháp tạo module

ng g module [tên module]



The screenshot shows a code editor window with the title "Welcome" and the file name "app.module.ts". The code editor displays the following TypeScript code for an Angular application module:

```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 import { AppComponent } from './app.component';
4 //Các thư viện được sử dụng cho module này
5 //Thư viện bao gồm: component, module hoặc các dependencies khác...
6
7 @NgModule({
8   declarations: [
9     AppComponent //Các component có thể import trực tiếp vào appmodule
10   ],
11   imports: [
12     BrowserModule //Các module khác bao gồm các module chứa những chức năng khác được import vào đây
13     /*Hoặc các module quản lý component được import vào đây
14     Thay vì ta import từng component tại declaration ta sẽ import 1 group component vào đây thông qua module đó*/
15   ],
16   providers: [], //Nơi đăng ký sử dụng service của app.module
17   bootstrap: [AppComponent] //Xác định component nào là component gốc để chạy ứng dụng
18 })
19 export class AppModule { }
```

Component

1. Khái niệm về component

- Component biểu diễn một phần giao diện UI của web site(file.html).
- Nói 1 cách đơn giản 1 component là 1 thẻ HTML do mình tự định nghĩa, trong thẻ đó chứa nội dung html do mình biên soạn.
- Một component bao gồm:
 - Giao diện html
 - Css của giao diện html đó
 - Selector (tên thẻ component do ta tự đặt)
 - Ngoài ra còn chứa 1 class javascript để xử lý cho component đó và được export ra ngoài.

Component

App component (Thuộc app module)

HomeLayout

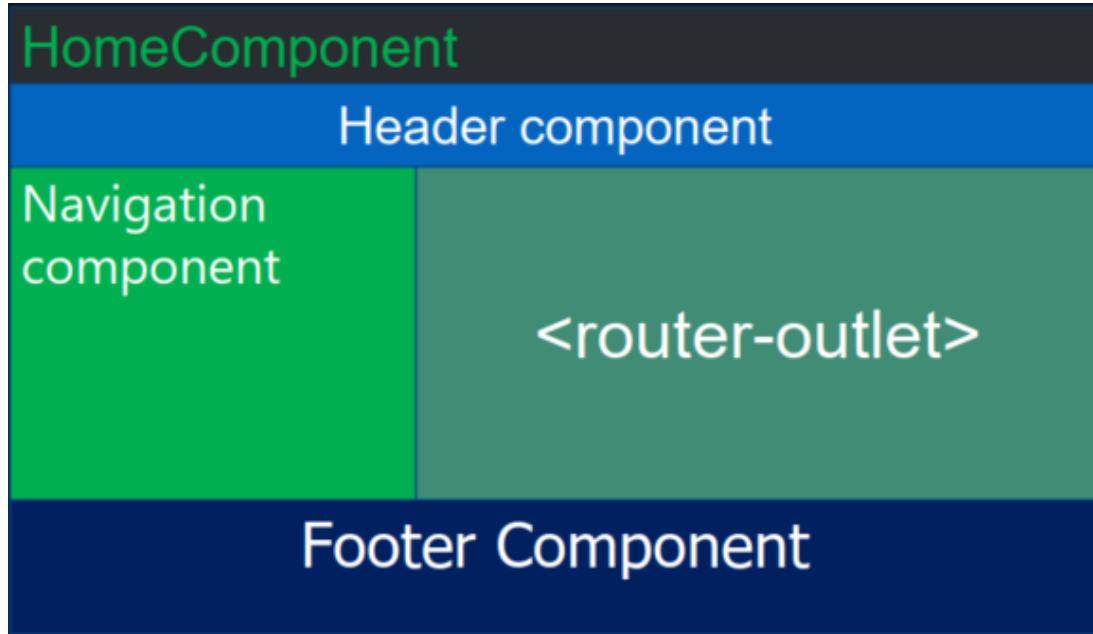
COMPONENT

AdminLayout

COMPONENT

Để chuyển đổi qua lại giữa các trang
(hay component) ta có khái niệm
RouteConfig (Sẽ học ở các slide sau)

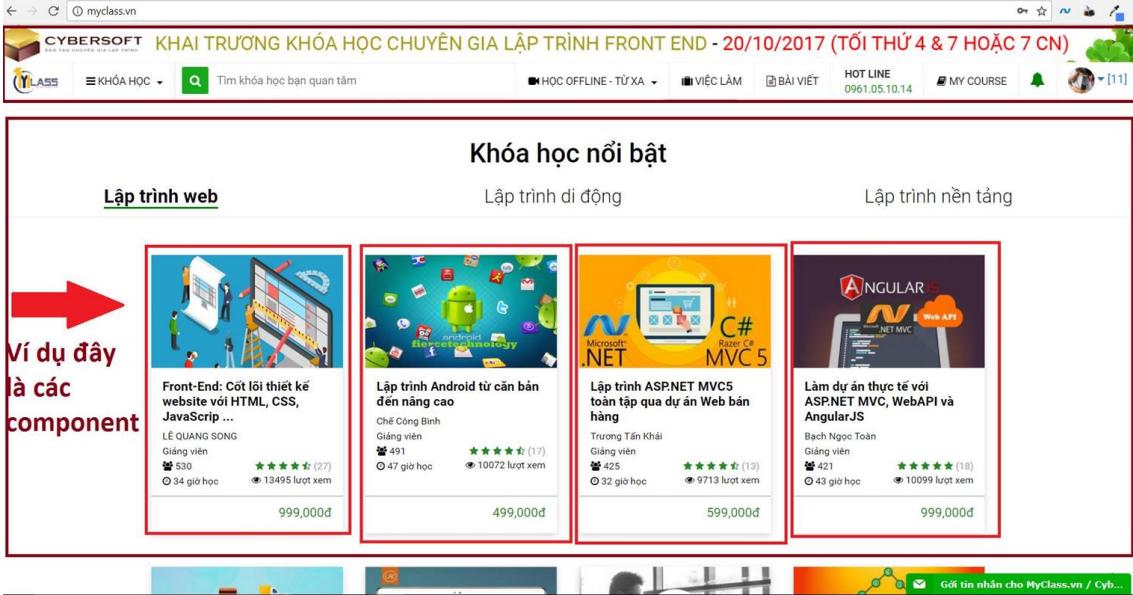
Component



Một component được cấu tạo từ các component con và tất cả component đó được quản lý bởi 1 module.

Component

Ví dụ ta có ứng dụng web myclass.vn là app.module thì ta sẽ có main component chứa tất cả các component khác là app.component.



Lập trình web

Ví dụ đây
là các
component

Khai trương khóa học chuyên gia lập trình front end - 20/10/2017 (tối thứ 4 & 7 hoặc 7 CN)

CYBERSOFT

Khóa học

Tìm khóa học bạn quan tâm

HOT LINE
0961.05.10.14

MY COURSE

[11]

Khóa học nổi bật

Lập trình di động

Lập trình nền tảng

Front-End: Cơ bản thiết kế website với HTML, CSS, JavaScript ...
LÊ QUANG SONG
Giảng viên
530 giờ học
13495 lượt xem
999,000đ

Lập trình Android từ cơ bản đến nâng cao
Chế Công Bình
Giảng viên
491 giờ học
47 giờ học
10072 lượt xem
499,000đ

Lập trình ASP.NET MVC5 toàn tập qua dự án Web bán hàng
Trương Tân Khái
Giảng viên
425 giờ học
32 giờ học
9713 lượt xem
599,000đ

Làm dự án thực tế với ASP.NET MVC, WebAPI và AngularJS
Bach Ngoc Toan
Giảng viên
421 giờ học
43 giờ học
10099 lượt xem
999,000đ

Gửi tin nhắn cho MyClass.vn / Cyb...

- + Trong app.component chứa TrangChuComponent.
- + TrangChuComponent lại chứa các thành phần khóa học component,
- + Trong app.component cũng chứa Trang chi tiết khóa học component
- + Trang chi tiết cũng lại chứa các component khóa học và 1 số component khác

Component

2. Ví dụ: Ta thử tạo 1 component header tại appmodule.

Cú pháp tạo component

ng g component [tên component]

Trong header chứa các file định nghĩa header component

The screenshot shows the VS Code interface with the Explorer sidebar open, displaying the project structure. The 'src' folder contains an 'app' folder, which in turn contains a 'header' folder. Inside 'header', there are four files: 'header.component.css', 'header.component.html', 'header.component.spec.ts', and 'header.component.ts'. The 'header.component.ts' file is currently selected and shown in the main editor area. The code defines a component named 'HeaderComponent' with a selector of 'app-header', a template URL of './header.component.html', and a style URL of './header.component.css'. It also implements the 'OnInit' interface and has a constructor.

```
TS header.component.ts
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-header',
5   templateUrl: './header.component.html',
6   styleUrls: ['./header.component.css']
7 })
8 export class HeaderComponent implements OnInit {
9
10   constructor() { }
11
12   ngOnInit() {
13   }
14
15 }
16
```

Tiến hành import HeaderComponent vào App module

The screenshot shows the 'app.module.ts' file in the VS Code editor. The code imports 'BrowserModule' and 'NgModule' from '@angular/platform-browser' and 'AppComponent' from './app.component'. It then imports 'HeaderComponent' from './header/header.component'. The '@NgModule' block declares 'AppComponent' and 'HeaderComponent' as declarations, imports 'BrowserModule', and bootstrap with 'AppComponent'. Finally, it exports the 'AppModule' class.

```
TS app.module.ts
1
2 import { BrowserModule } from '@angular/platform-browser';
3 import { NgModule } from '@angular/core';
4 import { AppComponent } from './app.component';
5 //Import component header vào app.module
6 import { HeaderComponent } from './header/header.component';
7
8 @NgModule({
9   declarations: [
10     AppComponent, HeaderComponent
11   ],
12   imports: [
13     BrowserModule
14   ],
15   providers: [],
16   bootstrap: [AppComponent]
17 })
18 export class AppModule { }
```

Component

Để hiển thị HeaderComponent ta làm thế nào ???

The screenshot shows the Visual Studio Code interface with the following components:

- EXPLORER**: Shows the project structure. The `src/app/header` folder is expanded, showing files: `header.component.css`, `header.component.html`, `header.component.spec.ts`, and `header.component.ts`. The `header.component.html` file is currently selected.
- OPEN EDITORS**: Shows the `app.component.html` file, which contains the code:

```
<app-header></app-header>
```

 This code is highlighted with a yellow box.
- STARTANGULAR2_5**: Shows the project's root directory with subfolders: `e2e`, `node_modules`, and `src`.
- Panels**:
 - PREVIEW**: Shows the rendered HTML output of the `app.component.html` file, displaying the `<app-header></app-header>` placeholder.
 - EDITOR**: Shows the `header.component.ts` file content:

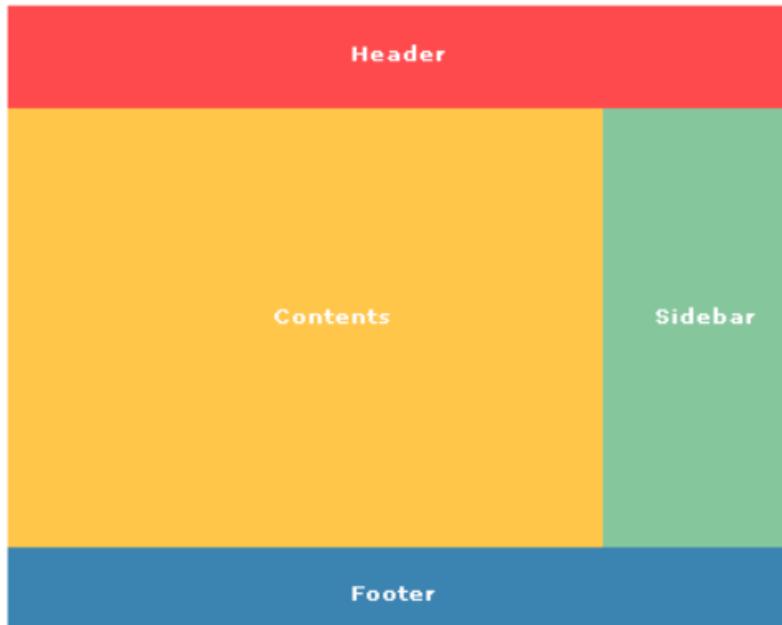
```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.css']
})
export class HeaderComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }
}
```

Bài tập 1:



➤ Yêu cầu:

- Chia component như hình đưa vào app.component.
- Gợi ý:
 - Sử dụng bootstrap bằng cách chèn cdn vào file index.html.
 - Sau đó chia layout (Chia cột) rồi chèn các thẻ component vào các cột

Bài tập 1:

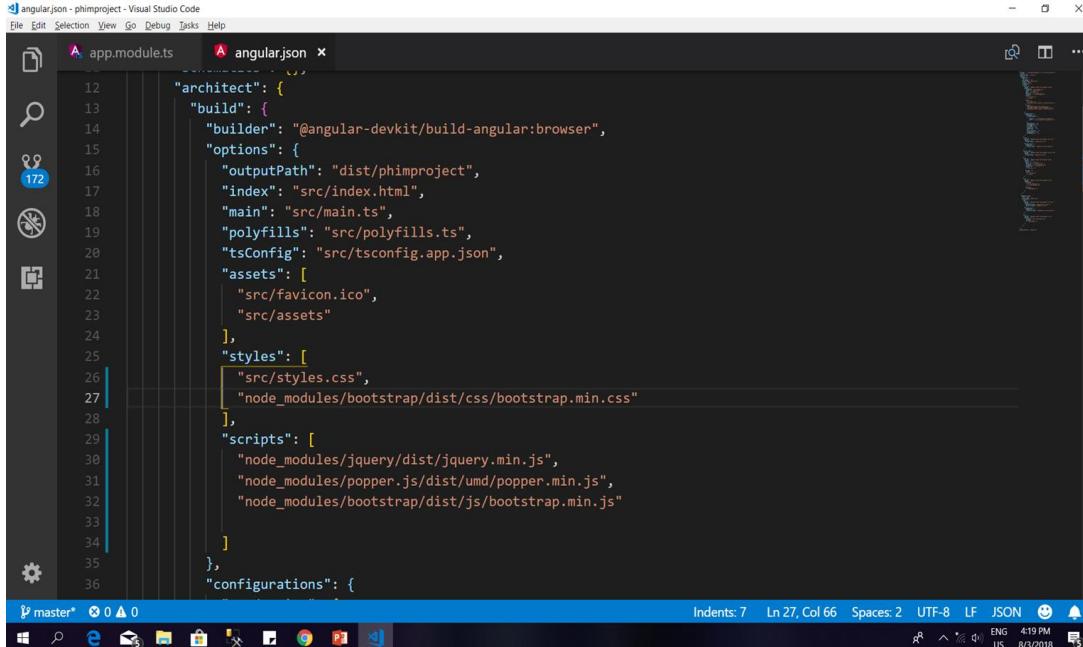
➤ Hướng dẫn cài đặt bootstrap 4

- Cách 1:

Đưa đường dẫn bootstrap vào file index.html (Cách dễ nhất)

- Cách 2

1. npm install bootstrap jquery popper.js –save
2. Vào file **angular.json** thêm vào đường dẫn tới css và javascript để đóng gói webpack.
3. Tiến hành **ng serve** lại



```
angular.json - phimproject - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

app.module.ts angular.json

12 "architect": {
13   "build": {
14     "builder": "@angular-devkit/build-angular:browser",
15     "options": {
16       "outputPath": "dist/phimproject",
17       "index": "src/index.html",
18       "main": "src/main.ts",
19       "polyfills": "src/polyfills.ts",
20       "tsConfig": "src/tsconfig.app.json",
21       "assets": [
22         "src/favicon.ico",
23         "src/assets"
24       ],
25       "styles": [
26         "src/styles.css",
27         "node_modules/bootstrap/dist/css/bootstrap.min.css"
28       ],
29       "scripts": [
30         "node_modules/jquery/dist/jquery.min.js",
31         "node_modules/popper.js/dist/umd/popper.min.js",
32         "node_modules/bootstrap/dist/js/bootstrap.min.js"
33       ]
34     },
35   },
36   "configurations": {
```

master* 0 ▲ 0 Indents: 7 Ln 27, Col 66 Spaces: 2 UTF-8 LF JSON ENG 4:19 PM US 8/3/2018

Bài tập 2:

The screenshot shows a website layout with the following components labeled:

- headercomponent**: The top navigation bar with links to Home, About, Services, and Contact.
- IndexComponent (viền vàng)**: The main header section containing the business name "Business Name or Tagline" and a large placeholder "1920 x 400".
- SliderComponent (Viền tím)**: A purple-bordered section below the main header.
- What We Do**: A section with a brief description of services and a "Call to Action" button.
- Contact Us**: A section with contact information for "Start Bootstrap".
- IndexContentComponent (Viền lục)**: A green-bordered section containing three cards, each with a title, a short description, and a "Find Out More" button.
- ItemComponent (Lưu ý thiết kế 1 component sau đó copy ra 3 component)**: A blue-bordered section containing three cards, each with a title, a short description, and a "Find Out More" button.
- FooterComponent**: The footer section with copyright information: "Copyright © Your Website 2017".

➤ Yêu cầu

+Tạo và thiết kế component tương ứng.

+Bố cục giao diện các component như hình ảnh.

Lưu ý: Không cần dàn layout chi tiết cho nội dung từng component, có thể copy bootstrap



Bài tập 3: Dàn layout có nội dung

Phim đang chiếu | Phim sắp chiếu

Avenger 3

Edward Smith

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quis, explicabo. Iusto, quod, et, non, in, modis, possit, ne, perspicillat, magnum.

★★★★★

Kết án lồng tiếng Việt | Kế Sanh Lồng tiếng Việt

2018 | 2018 | 2018 | 2018 | 2018 | 2018

★★★★★ | ★★★★★ | ★★★★★ | ★★★★★ | ★★★★★ | ★★★★★

LATEST MOVIE

KẾ SANH LỒNG TIẾNG HỘI | Avenger 3

24h hội hè | Edward Smith

Bé ngoan bé hư | Kế Sanh Lồng tiếng Việt

Haben hoa rong chát | Kế Sanh Lồng tiếng Việt

Kế sán mèo | Kế Sanh Lồng tiếng Việt

Món ma ném quay | Kế Sanh Lồng tiếng Việt

Phát họa | Kế Sanh Lồng tiếng Việt

2018 | 2018 | 2018 | 2018 | 2018 | 2018

★★★★★ | ★★★★★ | ★★★★★ | ★★★★★ | ★★★★★ | ★★★★★

KẾ SANH LỒNG TIẾNG HỘI | Ferdinand

Audi cool dung | Kế Sanh Lồng tiếng Việt

Bé ngoan bé hư | Kế Sanh Lồng tiếng Việt

Haben hoa rong chát | Kế Sanh Lồng tiếng Việt

Kế sán mèo | Kế Sanh Lồng tiếng Việt

Phát họa | Kế Sanh Lồng tiếng Việt

2018 | 2018 | 2018 | 2018 | 2018 | 2018

★★★★★ | ★★★★★ | ★★★★★ | ★★★★★ | ★★★★★ | ★★★★★

TOP MOVIE

KẾ SANH LỒNG TIẾNG HỘI | KẾ SANH LỒNG TIẾNG HỘI | KẾ SANH LỒNG TIẾNG HỘI | KẾ SANH LỒNG TIẾNG HỘI

Kế Sanh lồng tiếng Việt | Kế Sanh lồng tiếng Việt | Kế Sanh lồng tiếng Việt | Kế Sanh lồng tiếng Việt

2018 | 2018 | 2018 | 2018

★★★★★ | ★★★★★ | ★★★★★ | ★★★★★

KẾ SANH LỒNG TIẾNG HỘI | KẾ SANH LỒNG TIẾNG HỘI | KẾ SANH LỒNG TIẾNG HỘI | KẾ SANH LỒNG TIẾNG HỘI

Kế Sanh lồng tiếng Việt | Kế Sanh lồng tiếng Việt | Kế Sanh lồng tiếng Việt | Kế Sanh lồng tiếng Việt

2018 | 2018 | 2018 | 2018

★★★★★ | ★★★★★ | ★★★★★ | ★★★★★

Avenger 3

Edward Smith

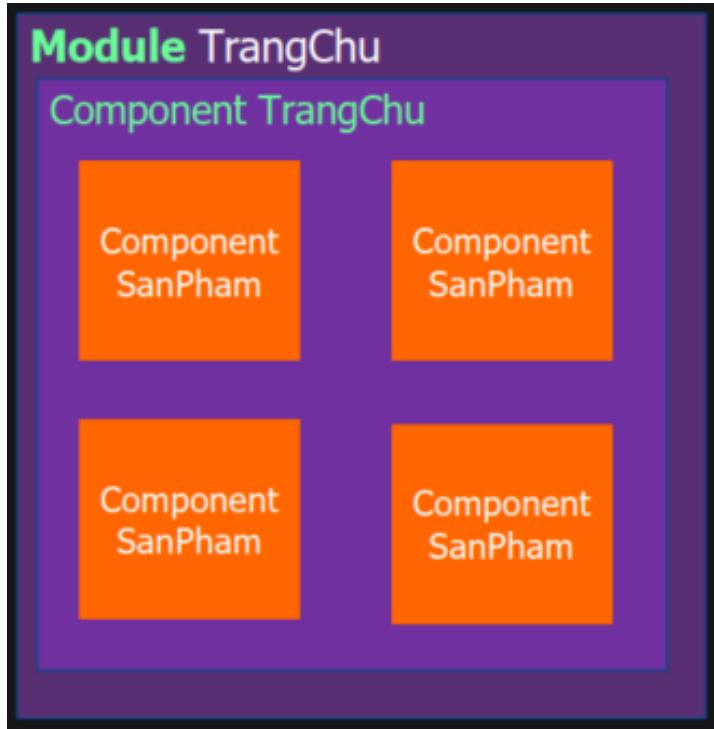
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quis, explicabo. Iusto, quod, et, non, in, modis, possit, ne, perspicillat, magnum.

★★★★★

➤ Yêu cầu:

Phân tích xem trang này có bao nhiêu component.
Xây dựng các component tương ứng

Phân tách module – Nhóm các component

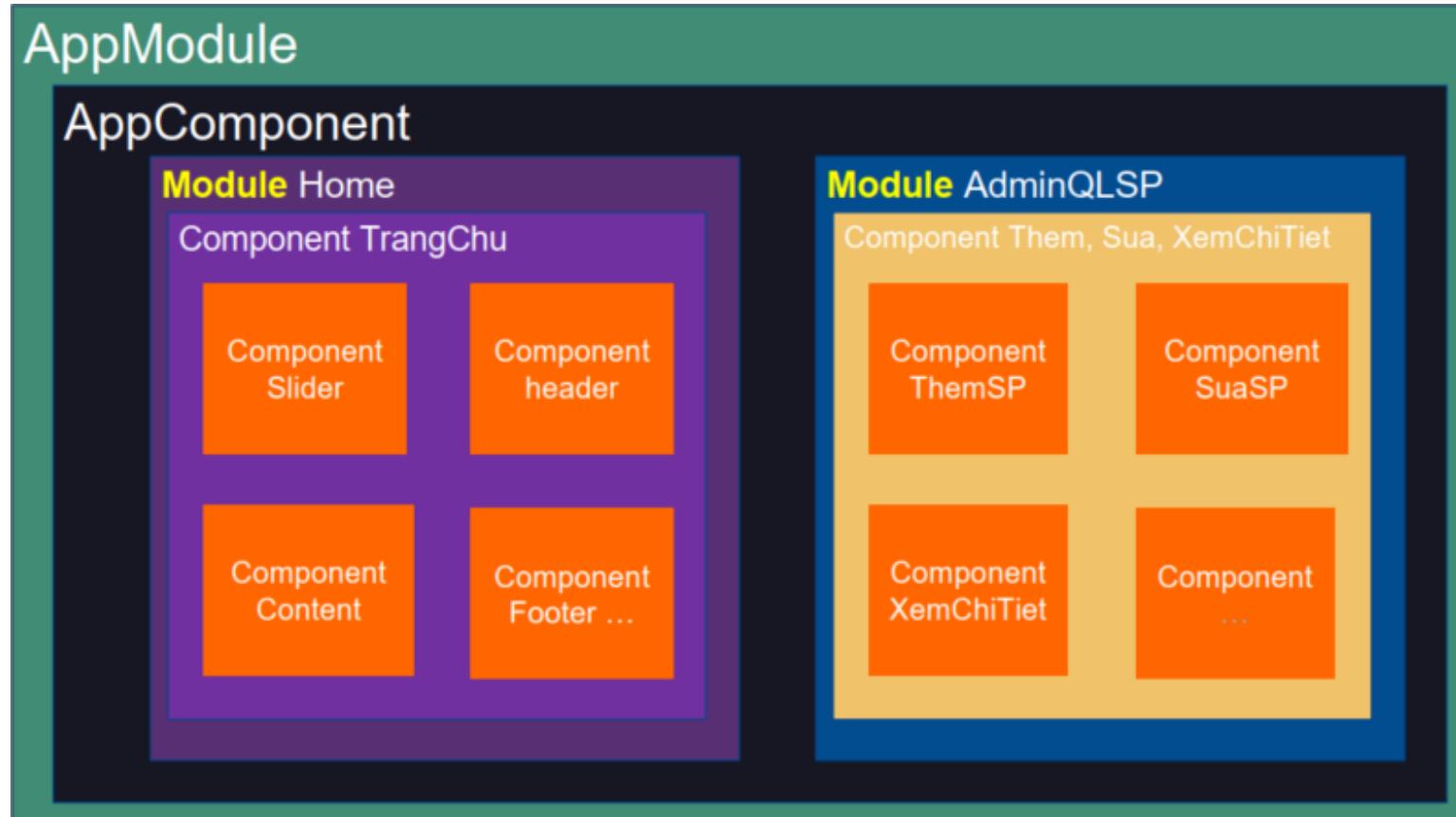


- ❖ Tại sao phải phân tách module?
- ❖ Phân tách module như thế nào?

Việc phân tách module giúp ta dễ quản lý ứng dụng, chia nhỏ công việc, dễ dàng bảo trì và nâng cấp.

Phân tách module dựa trên nhóm các component phục vụ cho 1 chức năng nào đó. Ví dụ: Nhìn vào hình bên ta thấy Module Trang chủ chứa các component để hình thành nên trang chủ.

Phân tách module – Nhóm các component



Phân tách module – Nhóm các component

➤ Ví dụ về Layout module (Một module con)

TS app.module.ts TS layout.module.ts x

```
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { HeaderComponent } from './header/header.component';
4 import { SiderbarComponent } from './siderbar/siderbar.component';
5 import { FooterComponent } from './footer/footer.component';
6
7 @NgModule({
8   imports: [
9     CommonModule
10   ],
11   exports:[HeaderComponent,SiderbarComponent,FooterComponent],
12   //Exports những component của module này ra bên ngoài để
13   //khi ở module khác import module này vào có thể sử dụng được những component này
14   declarations: [HeaderComponent, SiderbarComponent, FooterComponent]
15   //declarations: các component được import vào module này (giống như khai báo component)
16 })
17 export class LayoutModule {}
```

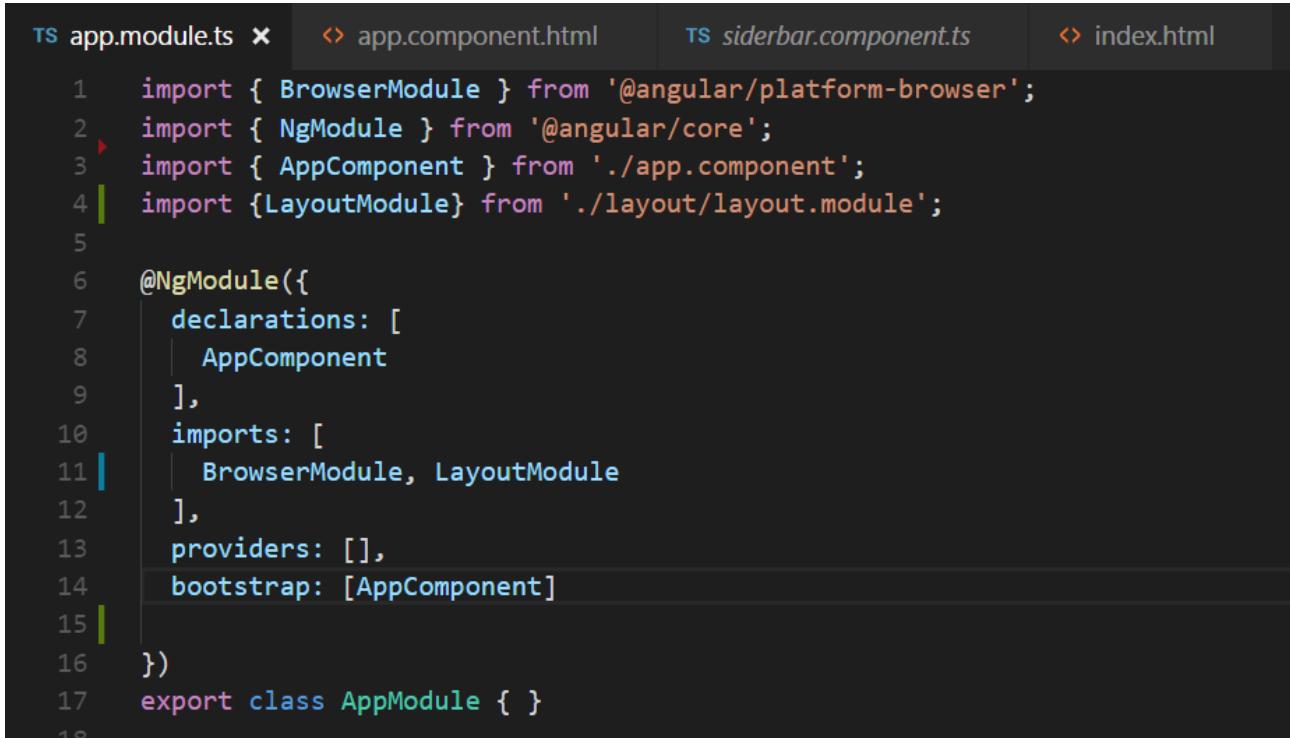
Ví dụ ta có 1 module **layout** chứa các component: **Header, Footer, Sidebar**

Vậy làm cách nào ta đưa được các component *Header,Footer, SiderBar* vào *app.component* sử dụng ???

=> Ta phải **exports** các **component** đó ra thì khi **import module này** vào *app.module* component này mới được hiểu

Phân tách module – Nhóm các component

Tại app.module khi muốn dùng component thì phải khai báo từng component ở declaration nhưng do ta gom nhóm module lại nên chỉ cần imports module đó vào là được



The screenshot shows a code editor with the following tabs: 'app.module.ts' (active), 'app.component.html', 'siderbar.component.ts', and 'index.html'. The code in 'app.module.ts' is as follows:

```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 import { AppComponent } from './app.component';
4 import {LayoutModule} from './layout/layout.module';

5
6 @NgModule({
7   declarations: [
8     AppComponent
9   ],
10  imports: [
11    BrowserModule, LayoutModule
12  ],
13  providers: [],
14  bootstrap: [AppComponent]
15 })
16 export class AppModule { }
```

Khái niệm về Template và Styles

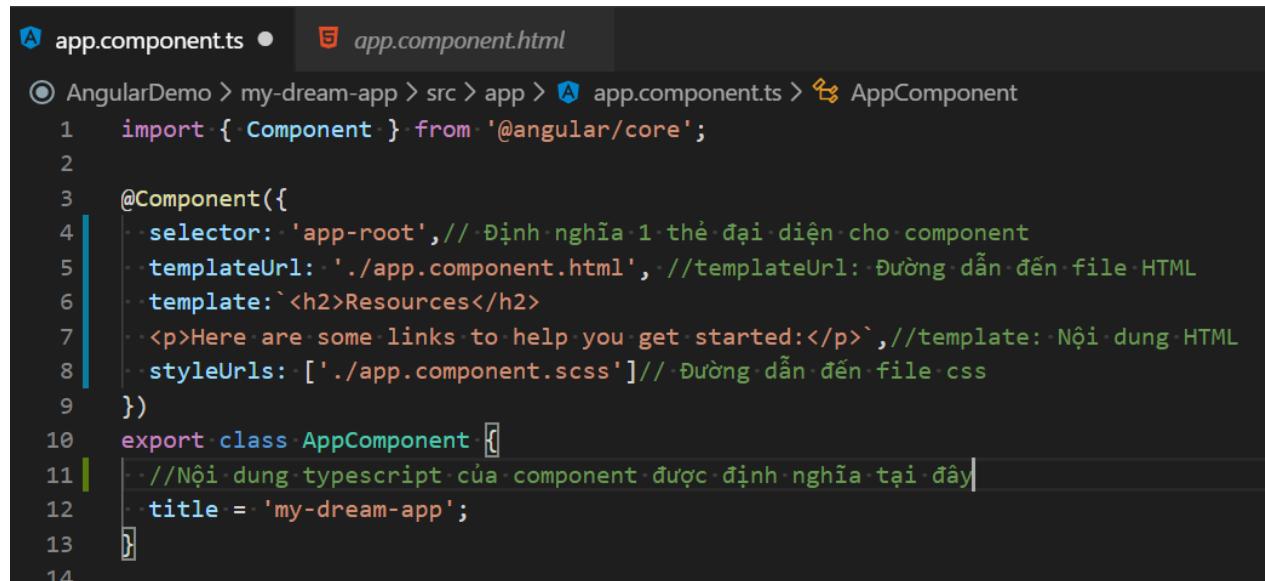
Template là 1 thành phần html của component.

Trong 1 component có 2 dạng định nghĩa template.

+**Template**: Định nghĩa nội dung html của component đó trực tiếp

+**TemplateUrl**: Định nghĩa nội dung html của component đó thông qua url đến file .html.

Tương tự style
cũng có **Styles**, và
StyleUrls



The screenshot shows a code editor with two tabs: `app.component.ts` and `app.component.html`. The `app.component.ts` tab is active, displaying the following TypeScript code:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',// Định nghĩa 1 thẻ đại diện cho component
5   templateUrl: './app.component.html',// templateUrl: Đường dẫn đến file HTML
6   template: `<h2>Resources</h2>
7 <p>Here are some links to help you get started:</p>`,//template: Nội dung HTML
8   styleUrls: ['./app.component.scss']// Đường dẫn đến file css
9 })
10 export class AppComponent {
11   //Nội dung typescript của component được định nghĩa tại đây
12   title = 'my-dream-app';
13 }
14
```

The `app.component.html` tab is visible in the background. The status bar at the bottom of the editor shows the path: AngularDemo > my-dream-app > src > app > `app.component.ts` > AppComponent.

Metadata

Thực chất 1 **component** chỉ là 1 **class**. Nó không phải là 1 component cho tới khi ta khai báo nó với angular.

Để khai báo 1 **class** với angular rằng nó là 1 **component**, ta sẽ gắn thẻ **metadata** vào class này.

Trong **typescript** để gắn thẻ metadata ta dùng **decorator**

Ở đây **@Component** chính là **decorator**

Class **SiderbarComponent** được định nghĩa thành component khi có **@Component**

```
TS siderbar.component.ts ×  
1 import { Component, OnInit } from '@angular/core';  
2  
3 @Component({  
4   selector: 'app-siderbar',  
5   templateUrl: './siderbar.component.html',  
6   styleUrls: ['./siderbar.component.css']  
7 })  
8 export class SiderbarComponent implements OnInit {  
9   constructor() { }  
10  ngOnInit() {  
11    }  
12  }  
13 }
```

Data-binding

Data binding là 1 tính năng của framework hiện đại, chúng đồng bộ dữ liệu được hiển thị trên view(template html) và model (class typescript).

Nói 1 cách đơn giản databinding là sự giao tiếp giữa Typescript và HTML

Các cơ chế binding dữ liệu:

- ❖ **One-way binding**

- Interpolation
- Property binding
- Event binding

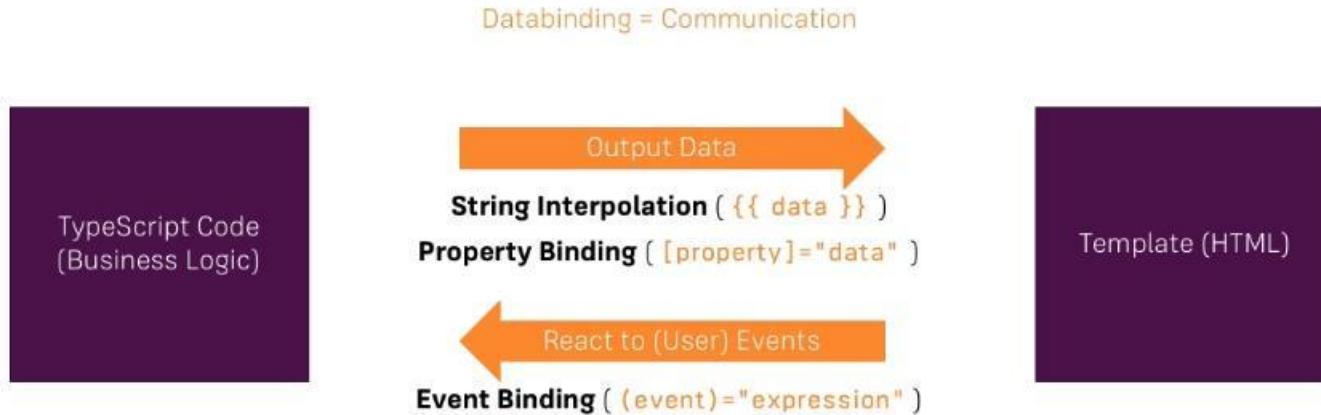
- ❖ **Two-way binding**

Data-binding

❖ Oneway binding

Oneway binding sử dụng các biến trong class component để binding data ra view hay còn gọi là template (Interpolation Binding) hoặc ngược lại nhận sự kiện từ user thông qua sự kiện (Event Binding)

Understanding Databinding



Data-binding

❖ Oneway binding - Interpolation

Interpolation Binding được viết với cú pháp `{{tenbien}}` hoặc `{{tenham()}}`. Nếu dữ liệu model thay đổi => view thay đổi.

```
ts onewaybinding.component.ts ✘

1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-onewaybinding',
5   template: `<input value="{{name}}>` ,
6   styleUrls: ['./onewaybinding.component.css']
7 })
8 export class OnewaybindingComponent implements OnInit {
9   name:string = "cybersoft";
10  constructor() { }
11  ngOnInit() {
12  }
13 }
```

Data-binding

❖ One-way binding - Property binding

Property binding cũng là một dạng **one-way binding** dữ liệu từ model được binding ra template (view) dưới dạng các thuộc tính của thẻ kết hợp dấu ngoặc []

```
ts propertybinding.component.ts x
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-propertybinding',
5   template: `<input [value] = 'name'>` , ←
6   styleUrls: ['./propertybinding.component.css']
7 })
8
9 export class PropertybindingComponent implements OnInit {
10   public name:string = "cybersoft";
11   constructor() { }
12
13   ngOnInit() {
14   }
15
16 }
17
```

Binding thông qua thuộc tính
value của thẻ input: [value]

Data-binding

❖ Oneway binding - Event Binding

Event binding thì ngược lại với 2 cách binding trước dữ liệu được binding từ view về model thông qua các sự kiện (Giống với event trong javascript hay jQuery).

```
TS eventbinding.component.ts x
1 import { Component, OnInit } from '@angular/core';
2 @Component({
3   selector: 'app-eventbinding',
4   template: `
5     <input type="text" />
6     <button (click)="DisplayName()" >Submit</button>
7   `,
8   styleUrls: ['./eventbinding.component.css']
9 })
10 export class EventbindingComponent implements OnInit {
11   public name:string = "cybersoft";
12   constructor() { }
13   DisplayName()
14   {
15     console.log(this.name);
16   }
17   ngOnInit() {
18   }
19
20 }
```

Để binding với sự kiện click ta dùng:
(sự kiện) = “phương thức xử lý”

Sự kiện ở đây là click,
phương thức DisplayName
(click) = “DisplayName()”

Data-binding

❖ Oneway binding - Event Binding

Event binding với tham số.

```
ts eventbinding.component.ts x
1 import { Component, OnInit } from '@angular/core';
2 @Component({
3   selector: 'app-eventbinding',
4   template: `
5     <input type="text" #thamso index='12' />
6     <button (click)="DisplayName(thamso)">Submit</button>
7   `,
8   styleUrls: ['./eventbinding.component.css']
9 })
10 export class EventbindingComponent implements OnInit {
11   public name:string = "cybersoft";
12   constructor() { }
13   DisplayName(thamso)
14   {
15     console.log(thamso.value); //Lấy value
16     //console.log(thamso.getAttribute("index")); //Lấy attribute
17   }
18   ngOnInit() {
19   }
20 }
21 }
```

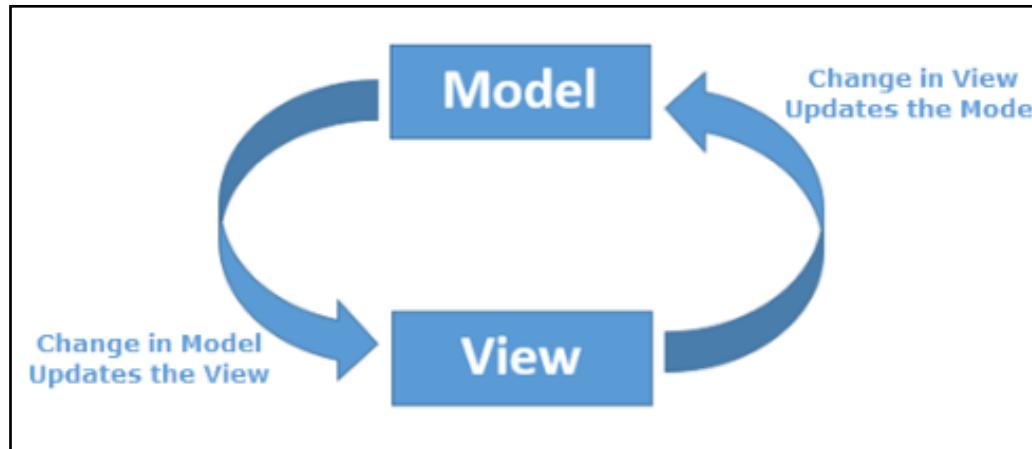
#thamso đại diện cho thẻ input. Muốn lấy giá trị value của thẻ thì .value tương tự muốn lấy các thuộc tính khác thì .thuộc tính khác

Data-binding

❖ Two-way binding

Là một sự kết hợp giữa property binding và event binding.

- + Model thay đổi => view thay đổi
- + View thay đổi => model thay đổi



Bài Tập 4:

Register form

Email:
Dùng [(ngModel)]

FullName:
Enter full name

Submit

Email: Dùng [(ngModel)]
FullName: Dùng event binding

➤ Yêu cầu:

- + Tạo 1 component với nội dung như hình vẽ.
- + Dùng interpolation để binding 2 thuộc tính **Email** và **FullName** ra 2 dòng text dưới nút submit.
- + Dùng event binding để khi người dùng nhập vào **FullName** nhấn nút **submit** thì phần text màu cam thay đổi thành nội dung nhập vào
- + Dùng two-way binding để thực hiện khi người dùng gõ vào **Email** thì dòng text màu đỏ phía dưới thay đổi

Directives

Directives là một thành phần mở rộng HTML, hay nói cách khác là các thuộc tính (properties) của các thẻ HTML do Angular định nghĩa thêm. Directives phải tuân thủ theo nguyên tắc của Angular .

Có 3 loại directive:

- Components
- Structural directives
- Attribute directives

Structural directives

Dùng để thay đổi diện mạo của dom bằng cách thêm hoặc bớt các phần tử của DOM.

- *ngIf
- *ngSwitch
- *ngFor

Structural directives

*ngIf: dùng để ẩn hiện theo điều kiện nào đó là true hoặc là false.

```
1 import { Component, OnInit } from '@angular/core';
2 @Component({
3   selector: 'app-directive',
4   template: `
5     <div *ngIf='status'>cybersoft</div>
6     <button (click) = "Hidden()"> Hidden </button>
7     <button (click) = "Show()"> Show </button>
8   `,
9   styleUrls: ['./directive.component.css']
10 })
11 export class DirectiveComponent implements OnInit {
12   public status:boolean = true;
13   Hidden()
14   {
15     this.status = false;
16   }
17   Show()
18   {
19     this.status = true;
20   }
}
```

Xem ví dụ ta thấy:

Ta có 1 div với nội dung cybersoft.

Div này được **hiển thị** khi giá trị **status = true**, và **ẩn đi** khi giá trị **status = false**.

Ngoài ra ta còn có thể thay thế biến đó thành 1 **biểu thức điều kiện**

Ví dụ: <div *ngIf=(number>2)></div>

Hoặc phương thức ...

Structural directives

*ngIf else: ở những phiên bản sau angular hỗ trợ ta thêm directive ng-template và ngIf else

- **ng-template:**

Về tính chất nó giống như 1 component (tái sử dụng) tuy nhiên phạm vi sử dụng chỉ được khai báo và sử dụng trong component chứa nó thôi.

- **ng-if else:**

Giống như các ngôn ngữ lập trình directive if else giúp ta làm nhiệm vụ khi điều kiện if không thỏa (Thay vì phải dùng 2 lệnh if).

Structural directives

if-else.component.html

```
1 <div *ngIf="status; else NoidungElse">
2
3   <button (click)="DangXuat()"> Đăng xuất </button>
4
5   <h5>Xin chào! <b>Cybersoft</b></h5>
6 </div>
7
8 <ng-template #NoidungElse>
9   <button (click)="DangNhap()"> Đăng nhập </button>
10  <h5>Chưa đăng nhập</h5>
11 </ng-template>
```

Khi điều kiện if không thỏa mãn
hiển thị phần nội dung template

```
export class IfElseComponent implements OnInit {
  status: boolean = true;
  constructor() { }
  ngOnInit() {}
  DangNhap() {
    this.status = true;
  }
  DangXuat() {
    this.status = false;
  }
}
```

localhost:9999

Xin chào! Cybersoft

Đăng xuất

localhost:9999

Chưa đăng nhập

Đăng nhập

Ý nghĩa:

- + Ban đầu ta có thuộc tính **status = true**.
- + Ta xét điều kiện **if** của biến **status** này nếu **true** nó sẽ hiển thị phần nội dung phía trên là **xin chào cybersoft**.
- + Ngược lại (**else**) thì nó sẽ hiển thị phần **ng-template** có **#NoidungElse** lên mà không hiển thị dòng **xin chào cybersoft**.
- + Để thử trường hợp ngược lại ta xây dựng 1 nút button với sự kiện click làm thay đổi trạng thái của biến **status** này.

Structural directives

*ngSwitch: dùng để ẩn hiện theo điều kiện giá trị điều kiện

```
2
3  @Component({
4    selector: 'app-ngswitch',
5    template: `
6      <div [ngSwitch]="dkSwitch" >
7        <div *ngSwitchCase="'red'" style="color:red"> M&uacute;a đỏ </div>
8        <div *ngSwitchCase="'blue'" style="color:blue"> M&uacute;a xanh </div>
9        <div *ngSwitchCase="'green'" style="color:green"> M&uacute;a xanh l&agrave; </div>
10       <div *ngSwitchDefault style="color:orange"> M&uacute;a cam m&acirc;nh đ&igrave;n </div>
11     </div>
12     <select #t (change)= [changeColor(t.value)]>
13       <option value="red">red</option>
14       <option value="green">green</option>
15       <option value="blue">blue</option>
16       <option value="orange">orange</option>
17     </select>
18   `,
19   styleUrls: ['./ngswitch.component.css']
20 })
21 export class NgswitchComponent implements OnInit {
22   public dkSwitch:string = 'red';
23   changeColor(value){
24     this.dkSwitch = value;
25 }
```

[ngSwitch]=“dieukien”

tag nào chứa giá trị

*ngSwitchCase = “giá trị của biến điều kiện” Thì tag đó được hiển thị

Structural directives

*ngFor: Cấu trúc lặp dùng để duyệt mảng hoặc danh sách các phần tử trong mảng object

```
ts ng-for.component.ts ●

1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-ng-for',
5    template: `
6      <h1>Duyệt mảng object</h1>
7      <div *ngFor="let item of DSNV">{{item.ten}}: {{item.tuoi}}</div>
8      <h1>Duyệt mảng thông thường </h1>
9      <div *ngFor="let item of Colors">{{item}}</div>
10     `,
11    styleUrls: ['./ng-for.component.css']
12  })
13  export class NgForComponent implements OnInit {
14
15    //Mảng object: Dùng *ngFor để duyệt mảng object
16    public DSNV:any = [{ten:"khai",tuoi:18},{ten:"long",tuoi:17},{ten:"minh",tuoi:22}]
17    //Mảng
18    public Colors = ["red","green","blue"]
19    constructor() { }
20
21    ngOnInit() {
22    }
```

Structural directives

*ngFor: Ngoài ra để lấy được vị trí hay còn gọi là chỉ mục (index) của từng phần tử được duyệt ta có thể khai báo thêm biến để hứng lấy giá trị này.

```
@Component({
  selector: 'app-ngfor',
  template: `
    <div *ngFor="let cauhoi of DanhSachCauHoi; let STT = index">
      <h5><b>{{STT}} .</b> {{cauhoi}}</h5>
    </div>`,
  styleUrls: ['./ngfor.component.css']
})
export class NgforComponent implements OnInit {

  private DanhSachCauHoi:Array<any> = [
    "Tên của bạn là gì ?",
    "Bạn bao nhiêu tuổi",
    "Bạn đến từ đâu ?"
  ]
  constructor() {}
  ngOnInit() {
  }
}
```

Lấy vị trí phần tử của mảng gán vào biến chạy STT

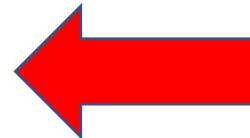
Kết quả:

- ngFor
 - 0 . Tên của bạn là gì ?
 - 1 . Bạn bao nhiêu tuổi
 - 2 . Bạn đến từ đâu ?

Multiple Structural Directives

Khi ta sử dụng nhiều Structural directives trên cùng 1 tag như hình bên dưới thì sẽ bị lỗi.

```
<div class="lesson" *ngIf="lessons"
      *ngFor="let lesson of lessons">
    <div class="lesson-detail">
      {{lesson | json}}
    </div>
</div>
```



Lỗi

Multiple Structural Directives

Do vậy ta phải tách ra dùng thêm 1 div ở ngoài để kiểm tra điều kiện. Nhưng nếu làm như vậy thì trên giao diện ta sẽ dư 1 thẻ div.

```
<div *ngIf="lessons">  
  <div class="lesson" *ngFor="let lesson of lessons">  
    <div class="lesson-detail">  
      {{lesson }}  
    </div>  
  </div>  
</div>
```

ng-container giúp ta sử dụng kết hợp được các structural directive nhưng không cần phải tốn thêm 1 tag hiển thị trên giao diện.



```
<ng-container *ngIf="lessons">  
  <div class="lesson" *ngFor="let lesson of lessons">  
    <div class="lesson-detail">  
      {{lesson }}  
    </div>  
  </div>  
</ng-container>
```

Ng-container & *ngTemplateOutlet

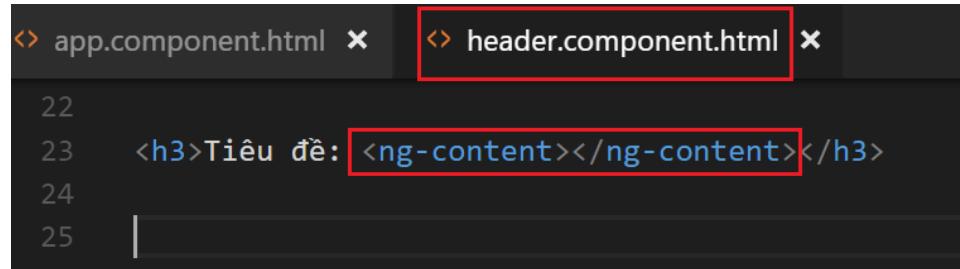
Để sử dụng template bất kỳ đâu trên giao diện ta dùng **ng-container** + ***ngTemplateOutlet**

```
<ng-container *ngTemplateOutlet="template"> </ng-container>
<ng-template #template>
    <div> Nội dung template </div>
</ng-template>
```

ng-content

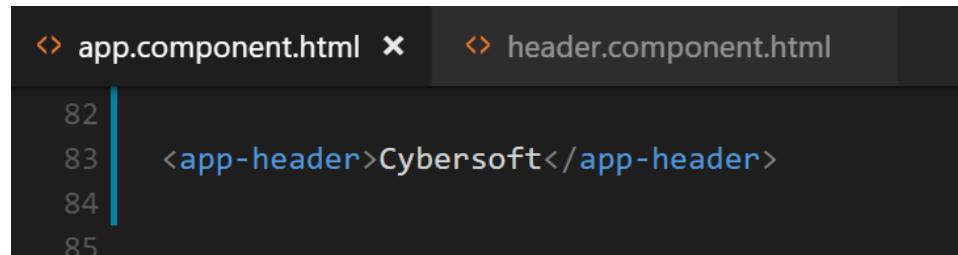
Sử dụng khi ta muốn chèn nội dung từ component cha đến component con tại vị trí chỉ định.

- Tại header component



```
22
23   <h3>Tiêu đề: <ng-content></ng-content></h3>
24
25
```

- Tại app component



```
82
83   <app-header>Cybersoft</app-header>
84
85
```

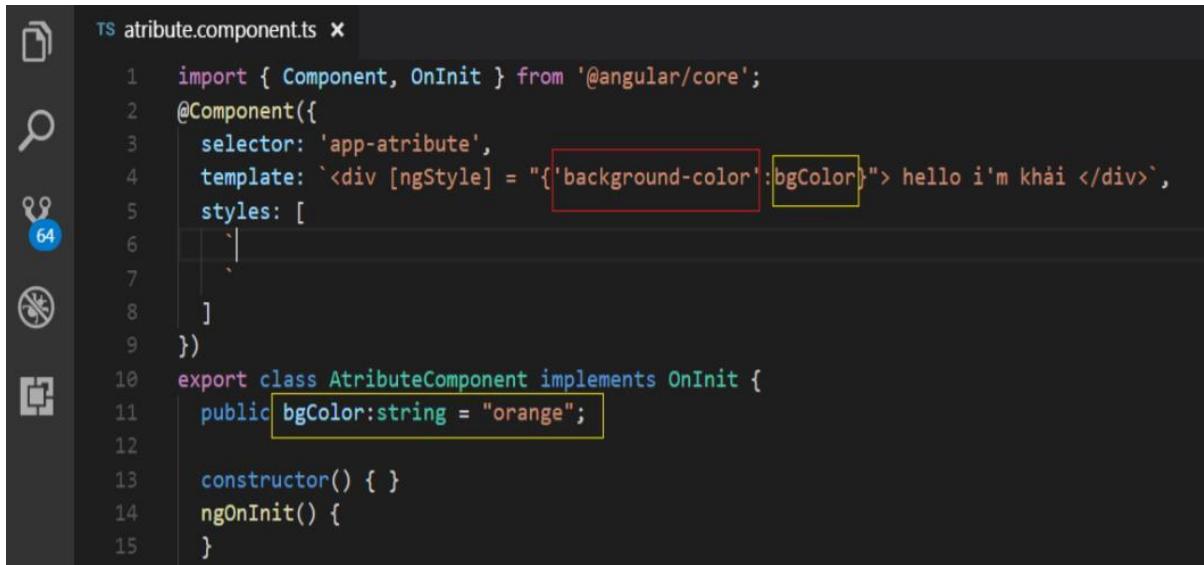
Attribute directives

*ngClass: Trong trường hợp áp dụng 1 hoặc 2 class trên một div là **classA** và **classB** thì ta sử dụng như sau

```
2  @Component({
3    selector: 'app-attribute',
4    template: `<div [ngClass] = "{mauchu:mauchu,fontchu:fontchu}"> cybersoft </div>`,
5    styles: [
6      `
7        .mauchu{
8          color:red;
9        }
10       .fontchu{
11         font-size:25px;
12       }
13     `
14   ]
15 })
16 export class AtributeComponent implements OnInit {
17   public mauchu:boolean = true;
18   public fontchu:boolean = true;
19   constructor() { }
20   ngOnInit() {
21 }
```

Attribute directives

*ngStyle: Ít sử dụng thường dùng cho các giá trị động được gán vào giá trị của thuộc tính css. VD: backgroundImage.



```
TS attribute.component.ts x
1 import { Component, OnInit } from '@angular/core';
2 @Component({
3   selector: 'app-attribute',
4   template: `<div [ngStyle] = "{'background-color':bgColor}"> hello i'm khai </div>`,
5   styles: [
6     `
7   `
8 ]
9 })
10 export class AtributeComponent implements OnInit {
11   public bgColor:string = "orange";
12
13   constructor() { }
14   ngOnInit() {
15 }
```

*ngStyle:

- Điều kiện của ngstyle là 1 biểu thức.
- Trong đó vùng màu đỏ là thuộc tính css vùng màu vàng là giá trị.
- Giá trị này có thể được lấy từ biến(Thuộc tính) hoặc phương thức()

Attribute directives

***ngNonBindable:** Sử dụng khi không muốn binding giá trị dữ liệu ra html. (Thường đối với chuỗi ký tự đặt biệt {{}})

```
import { Component, OnInit } from '@angular/core';
          Model
@Component({
  selector: 'app-baitap2',
  templateUrl: './baitap2.component.html',
  styleUrls: ['./baitap2.component.sass']
})
export class Baitap2Component implements OnInit {
  public name:string = "abc";
  constructor() { }

  ngOnInit() {
  }
}
```

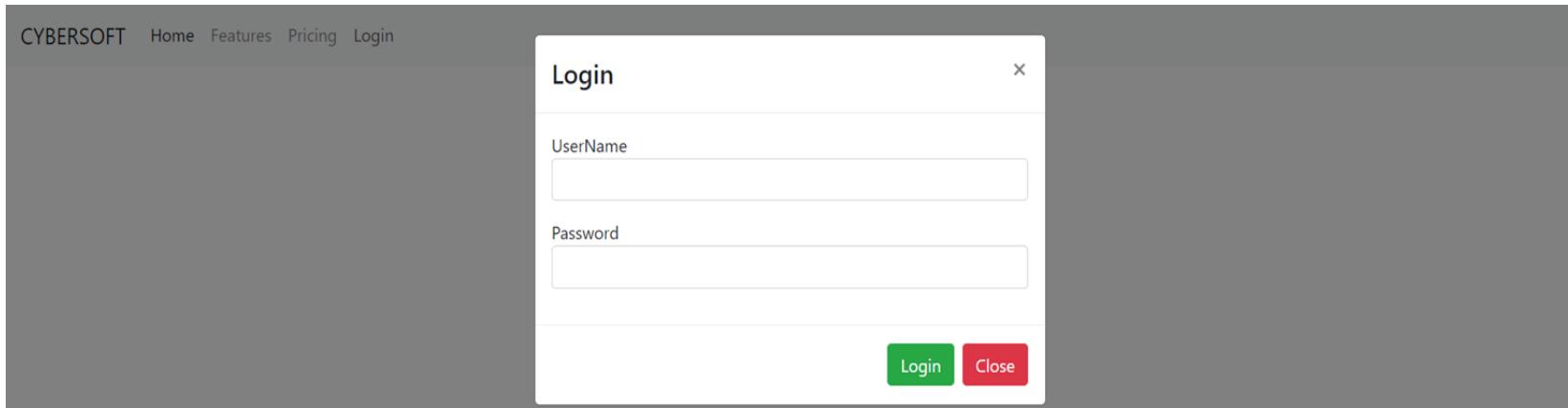
The screenshot shows a development environment with two tabs open: 'baitap2.component.html' and 'baitap2.component.ts'. In the 'baitap2.component.html' tab, there is a line of code: {{name}}. The 'ngNonBindable' attribute is highlighted with a yellow underline. In the 'baitap2.component.ts' tab, the variable 'name' is defined as a public string with a value of 'abc'. Below the tabs is a 'Browser' window showing the URL 'localhost:4200'. Inside the browser, the text '{{name}}' is displayed, which is also highlighted with a yellow box, demonstrating that the attribute directive prevents Angular from performing binding on that specific element.

Bài tập 5:

Yêu cầu:

- Tạo module cho bài tập này, tạo 1 component với giao diện như hình sử dụng bootstrap (hoặc tự thiết kế).
- Phần model bao gồm 2 thuộc tính
 - Thuộc tính **name** (Hiển thị tên người dùng sau khi đăng nhập)
 - Thuộc tính **isLogin** để xác định generate tại chữ Login ra giao diện hay là tên người dùng sau khi đã đăng nhập.
- Thực hiện chức năng cho nút login:
 - Nếu login với **username: cybersoft, password: cybersoft** thành công, thì lưu vào localStorage (Xét storage để kiểm tra cho câu hỏi tiếp theo)
- Dùng **ngIf** để xét điều kiện trên thanh menu là chữ Login hay là thuộc tính name sau khi người dùng đăng nhập.

Bài tập 5:



Bài tập 6:

Yêu cầu:

- Tạo module cho bài tập này, tạo 1 component với giao diện như hình sử dụng bootstrap (hoặc tự thiết kế).
- Phần model component có 1 thuộc tính Mảng danh sách sản phẩm.
- Dùng event binding để xây dựng sự kiện cho nút thêm sản phẩm (Lấy thông tin từ các input tạo thành object push vào thuộc tính mảng danh sách sản phẩm).
- Dùng structural directive *ngFor để binding dữ liệu ra table bên dưới.
- Dùng [ngClass] để xét màu cho các dòng chẵn lẻ.

Bài tập 6:

Quản lý danh mục sản phẩm

Mã SP
MTB

Tên SP
Máy tính bảng

Giá
100000

Thêm sản phẩm

STT	Mã SP	Tên SP	Giá
1	DT	Điện thoại	100000
2	LT	Laptop	100000
3	MTB	Máy tính bảng	100000

Attribute directives

```
ts high-light-directive.directive.ts ✘
1 import { Directive, ElementRef, OnInit } from '@angular/core';
2
3 @Directive({
4   selector: '[appHighLightDirective]'          Lớp đối tượng dùng để DOM
5 })
6 export class HighLightDirective implements OnInit{
7
8   constructor(private elementRef: ElementRef) {          Khai báo directive cũng giống như component tuy nhiên directive
9     Khai báo directive cũng giống như component tuy nhiên directive
10    không có thành phần Template (giao diện html) và Style (thành
11    phần css)
12    ngOnInit(){
13      this.elementRef.nativeElement.style.backgroundColor = 'green';          Cú pháp dùng để DOM tương tự JS : Gán background = màu xanh lục
14    }
15 }
```

*Attribute directive (Tự định nghĩa): Để định nghĩa 1 attribute directive cũng giống như định nghĩa 1 component tuy nhiên directive không có thành phần template (giao diện html) và style (thành phần css).

Cách Sử dụng: Muốn sử dụng directive ở module nào thì import và declaration lại module đó, từ đó có thể sử dụng ở bất kì component nào trong module đó.

```
<div appHighLightDirective> Cybersoft </div>
```

Attribute directives

*Renderer2: 1 tính năng của angular dùng để DOM đến các thành phần của HTML thông qua các phương thức rõ ràng và cụ thể hơn.

```
import { Directive, ElementRef ,OnInit , Renderer2 } from '@angular/core';

@Directive({
  selector: '[appHighLightDirective]'
})
export class HighLightDirectiveDirective implements OnInit{

  constructor(private elementRef:ElementRef, private render:Renderer2) {

  }
  ngOnInit(){
    this.elementRef.nativeElement.style.backgroundColor = 'green';
    //Đối tượng Renderer2 giúp dom đến element nhanh chóng và cụ thể (dễ dàng) hơn
    //Tương tự code phía trên
    this.render.setStyle(this.elementRef.nativeElement, 'background-color','green');
  }
}
```

Attribute directives

***HostListener** ('Tên sự kiện'): Dùng để định nghĩa sự kiện cho directive attribute.

```
export class HighLightDirective implements OnInit{  
  constructor(private elementRef: ElementRef, private render: Renderer2) {  
    } Định nghĩa sự kiện tương tự JQUERY  
    @HostListener('mouseenter') SuKienHover(eventData:Event)  
    {  
      this.render.setStyle(this.elementRef.nativeElement, 'background-color', 'green');  
    }  
    @HostListener('mouseleave') SuKienMouseLeave(eventData:Event)  
    {  
      this.render.setStyle(this.elementRef.nativeElement, 'background-color', 'blue');  
    }  
}
```

Attribute directives

*HostBinding ('thuộc tính của đối tượng html'): Dùng để ràng buộc thuộc tính của đối tượng html chứa attribute directive.

```
//Ràng buộc thuộc tính background Color của Element
@HostBinding('style.backgroundColor') bgColor:string = 'red';

@HostListener('mouseenter') SuKienHover(eventData:Event)
{
    this.bgColor = 'green' ;
}

@HostListener('mouseleave') SuKienMouseLeave(eventData:Event)
{
    this.bgColor = 'red';
}
```

Bài tập 7:

Cho mảng dữ liệu:

```
[  
    {MaSP:1,TenSP:"Sony XZ",Gia:1000},  
    {MaSP:2,TenSP:"Sony XZ2",Gia:1000},  
    {MaSP:3,TenSP:"HTC U Ultra",Gia:1000},  
    {MaSP:4,TenSP:"HTC U12 Plus",Gia:1000},  
    {MaSP:5,TenSP:"Iphone XS MAX",Gia:1000},  
    {MaSP:6,TenSP:"Iphone XR",Gia:1000},  
    {MaSP:7,TenSP:"Xiaomi Mi Note 3",Gia:9900},  
    {MaSP:8,TenSP:"Xiaomi Mi 8",Gia:1000},  
    {MaSP:9,TenSP:"Galaxy Note 9",Gia:1000},  
    {MaSP:10,TenSP:"Galaxy S9 Plus",Gia:1000},  
    {MaSP:11,TenSP:"Nokia X9",Gia:1000},  
];
```

Yêu cầu:

- Xây dựng giao diện thể hiện dữ liệu dưới dạng table.
- Xây dựng chức năng thêm sản phẩm, Xóa SP.
- Sử dụng directive phân trang dữ liệu từ thư viện: **ng-pagination**

Bài tập structural directive - CYBERSOFT

Quản lý danh mục sản phẩm		
Mã SP	Tên SP	Giá
1	Sony XZ	1,000
2	Sony XZ2	1,000
3	HTC U Ultra	1,000
4	HTC U12 Plus	1,000
5	Iphone XS MAX	1,000
6	Iphone XR	1,000
7	Xiaomi Mi Note 3	9,900
8	Xiaomi Mi 8	1,000
9	Galaxy Note 9	1,000
10	Galaxy S9 Plus	1,000

« Previous 1 2 Next »

Bài tập 7:

Hướng dẫn chi tiết

B1: Tạo 1 module đặt tên **BaiTapQLSPModule**

- Trong **BaiTapQLSPModule** tạo một component **BaiTapQLSPComponent** dùng bootstrap dàn layout như hình.

B2: Sử dụng directive ***ngFor** để tạo nội dung cho table => Load dữ liệu từ mảng đổ ra table (Dùng properties binding hoặc interpolation)

B3: Xây dựng chức năng thêm sản phẩm, xóa sản phẩm dựa vào eventbinding.

B4: Xây dựng tính năng phân trang dựa vào thư viện **ngx-pagination**.

- Cài đặt: **npm install ngx-pagination**
- Tài liệu: <https://www.npmjs.com/package/ngx-pagination>

Giao tiếp giữa các component

@Input

@Output

@ViewChild

@ViewChildren

@Input

- @Input: là cú pháp truyền dữ liệu từ component **cha** sang component **con**

```
ts parent.component.ts x
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-parent',
5    template: `<app-child *ngFor='let child of childList'
6      [nameChild]='child.name' [ageChild]='child.age'
7      ></app-child>`,
8    styleUrls: ['./parent.component.scss']
9  })
10 export class ParentComponent implements OnInit {
11
12   childList = [
13     { name: 'Nguyen Van A', age: 20 },
14     { name: 'Nguyen Van B', age: 21 },
15     { name: 'Nguyen Van C', age: 22 },
16   ];
17   constructor() { }
18
19   ngOnInit() {
20   }
21
22 }
23 |
```

CHA

```
ts child.component.ts x
1  import { Component, OnInit, Input } from '@angular/core';
2
3  @Component({
4    selector: 'app-child',
5    template: `<div>
6      Họ tên: {{nameChild}}
7      Tuổi: {{ageChild}}
8      </div>`,
9    styleUrls: ['./child.component.scss']
10 })
11 export class ChildComponent implements OnInit {
12   @Input() nameChild;
13   @Input() ageChild;
14
15   constructor() { }
16
17   ngOnInit() {
18   }
19
20 }
21 |
```

CON

@Output

- @Output: là cú pháp truyền dữ liệu từ **component con sang component cha**, thông qua **sự kiện** (click, change, keyup...) được khai báo bởi **EventEmitter**

```
TS child.component.ts x
1 import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';
2
3 @Component({
4   selector: 'app-child',
5   template: `<div>
6     Họ tên: {{nameChild}}
7     Tuổi: {{ageChild}}
8     <label>Thêm con: <input #newItem></label>
9     <button (click)="addNewItem(newItem.value)">
10       Thêm con vào danh sách ở cha
11     </button>
12   </div>`,
13   styleUrls: ['./child.component.scss']
14 })
15 export class ChildComponent implements OnInit {
16   @Input() nameChild;
17   @Input() ageChild;
18
19   @Output() newItemEvent = new EventEmitter<string>();
20
21   constructor() { }
22
23   ngOnInit() {
24   }
25
26   addNewItem(value: string) {
27     this.newItemEvent.emit(value);
28   }
29 }
30 }
```

CON

```
TS parent.component.ts x
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-parent',
5   template: `<app-child *ngFor='let child of childList'
6     [nameChild]='child.name' [ageChild]='child.age'
7     (newItemEvent)="addItem($event)">
8   </app-child>`,
9   styleUrls: ['./parent.component.scss']
10 })
11 export class ParentComponent implements OnInit {
12
13   childList = [
14     { name: 'Nguyen Van A', age: 20 },
15     { name: 'Nguyen Van B', age: 21 },
16     { name: 'Nguyen Van C', age: 22 },
17   ];
18   constructor() { }
19
20   ngOnInit() {
21   }
22
23   addItem(item) {
24     this.childList.push(item);
25   }
26 }
27 
```

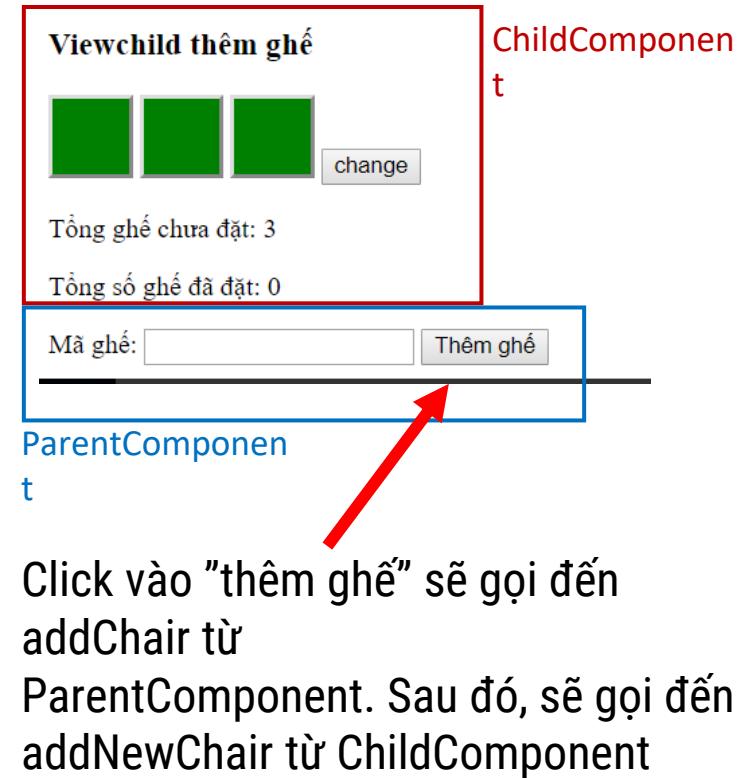
CHA

@ViewChild

- **@ViewChild:** dùng để truy vấn DOM đến component hoặc thẻ trong HTML.
@ViewChild trả về 1 đối tượng như: ElementRef, TemplateRef, ViewRef,
ComponentRef, và ViewContainerRef

@ViewChild (DOM đến component)

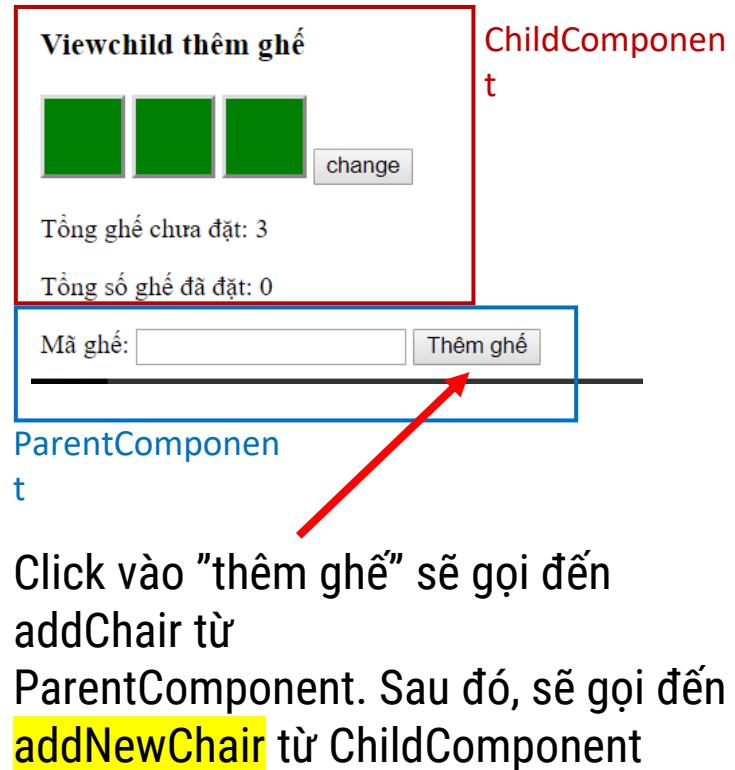
```
TS parent.component.ts x
1 import { Component, OnInit, ViewChild } from '@angular/core';
2 import { ChildComponent } from '../child/child.component';
3
4 @Component({
5   selector: 'app-parent',
6   template: `<h3>ViewChild thêm ghế</h3>
7   <app-child></app-child>← Component
8   Mã ghế: <input #chairId>
9   <button (click)="addChair(chairId.value)">Thêm ghế</button>
10  `,
11  styleUrls: ['./parent.component.scss']
12})
13 export class ParentComponent implements OnInit {
14
15   @ViewChild(ChildComponent, { static: false }) chair: ChildComponent;
16   constructor() { }
17
18   ngOnInit() {    DOM đến component
19   }                ChildComponent
20
21   addChair(maghe: string) {
22     this.chair.addNewChair(maghe, true);
23   }
24 }
| CH             Gọi addNewChair từ
25 | A             ChildComponent
```



@ViewChild (DOM đến component)

```
ts child.component.ts x
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-child',
5   template: `<div *ngFor='let chair of chairList'
6     | <div class="chair"></div>
7   </div>
8   <div>Tổng số ghế chưa đặt: {{totalChairNotBooked}}</div>
9   <div>Tổng số ghế đã đặt: {{totalChairBooked}}</div>
10  `,
11  styleUrls: ['./child.component.scss']
12})
13export class ChildComponent implements OnInit {
14  totalChairBooked;
15  totalChairNotBooked;
16  chairList: Array<any> = [
17    { chairId: 1, status: true },
18    { chairId: 2, status: false }
19  ];
20
21  constructor() { }
22
23  ngOnInit() {
24
25
26    addNewChair(chairIdFromParent: string, statusChairFromParent: boolean) {
27      const newChairObj = {
28        chairId: chairIdFromParent,
29        status: statusChairFromParent
30      };
31      this.chairList.push(newChairObj);
32    }
33  }
}
```

CO
N



@ViewChild (DOM đến tag HTML)

Import ViewChild và ElementRef từ angular core

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

Đặt #name cho thẻ cần DOM

```
<input type="text" (change)='ChangeName()' value="FullName" #inputFullName /> |
```

Khai báo kiểu dữ liệu cho thuộc tính DOM từ #name là ElementRef

```
@ViewChild('inputFullName') inputFullName: ElementRef  
  
ChangeName()  
{  
  console.log(this.inputFullName.nativeElement);  
}
```

Để lấy được thuộc tính và phương thức giống như javascript ta chấm đến thuộc tính nativeElement. Ví dụ để lấy value: **this.inputFullName.nativeElement.value**

@ViewChildren

- **View Children:** dùng để DOM đến danh sách các thẻ hoặc các component có cùng selector

@ViewChildren (DOM danh sách các selector)

```
TS parent.component.ts ×
1 import { Component, OnInit, ViewChildren, QueryList } from '@angular/core';
2 import { ChildComponent } from '../child/child.component';
3
4 @Component({
5   selector: 'app-parent',
6   template: `<h3>ViewChild thêm ghế</h3>
7     <app-child></app-child>
8     <button (click)="editChair()">Sửa ghế</button>
9     `,
10 styleUrls: ['./parent.component.scss']
11 })
12 export class ParentComponent implements OnInit {
13
14   @ViewChildren(ChildComponent) tagListChild: QueryList<
15     ChildComponent
16   >;
17   constructor() { }
18
19   ngOnInit() {
20   }
21
22   editChair() {
23     this.tagListChild.map(item => {
24       item.chairList.forEach(chair => {
25         chair.status = true;
26       });
27     });
28   }
29 }
30 
```

Kết quả trả về là 1 mảng đối tượng
nên dùng .map hoặc .forEach để duyệt

ChildComponent

ViewChildren trả về
QueryList

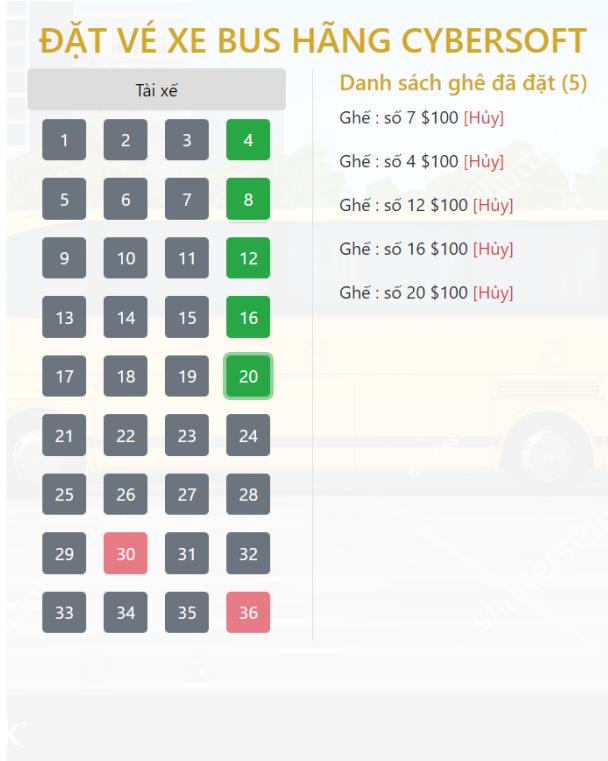
CH
A

```
TS child.component.ts ×
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-child',
5   template: `<div *ngFor='let chair of chairList'>
6     <div class="chair"></div>
7     <div>Tổng số ghế chưa đặt: {{totalChairNotBooked}}</div>
8     <div>Tổng số ghế đã đặt: {{totalChairBooked}}</div>
9     `,
10 styleUrls: ['./child.component.scss']
11 })
12 export class ChildComponent implements OnInit {
13   totalChairBooked;
14   totalChairNotBooked;
15   chairList: Array<any> = [
16     { chairId: 1, status: true },
17     { chairId: 2, status: false }
18   ];
19
20   constructor() { }
21
22   ngOnInit() {
23   }
24
25   addNewChair(chairIdFromParent: string, statusChairFromParent: boolean) {
26     const newChairObj = {
27       chairId: chairIdFromParent,
28       status: statusChairFromParent
29     };
30     this.chairList.push(newChairObj);
31   }
32 }
```

CO
N

Đặt vấn đề: sửa tất cả thuộc tính của mỗi selector child được sinh ra từ ParentComponent

Bài tập 8:



Yêu cầu cơ bản

- **Phân tích component** dựa vào hình bên trái.
- **Sao chép mảng ghế cho trước.**
- Yêu cầu hiện thực các ghế trên xe buýt bằng control button
- Phân tích lớp đối tượng và các phương thức xử lý.
- Dùng **input output** hoặc **viewchild** để xử lý sự kiện khi click vào ghế thì phía bên phải danh sách ghế sẽ được thêm vào 1 dòng tương ứng. Và tổng tiền sẽ được cộng lên.

```
{SoGhe:1,TenGhe: "số 1", Gia:100, TrangThai:false},  
{SoGhe:2,TenGhe: "số 2", Gia:100, TrangThai:false},  
{SoGhe:3,TenGhe: "số 3", Gia:100, TrangThai:false},  
{SoGhe:4,TenGhe: "số 4", Gia:100, TrangThai:false},  
{SoGhe:5,TenGhe: "số 5", Gia:100, TrangThai:false},  
{SoGhe:6,TenGhe: "số 6", Gia:100, TrangThai:false},  
{SoGhe:7,TenGhe: "số 7", Gia:100, TrangThai:false},  
{SoGhe:8,TenGhe: "số 8", Gia:100, TrangThai:false},  
{SoGhe:9,TenGhe: "số 9", Gia:100, TrangThai:false},  
{SoGhe:10,TenGhe: "số 10", Gia:100, TrangThai:false},  
{SoGhe:11,TenGhe: "số 11", Gia:100, TrangThai:false},  
{SoGhe:12,TenGhe: "số 12", Gia:100, TrangThai:false},  
{SoGhe:13,TenGhe: "số 13", Gia:100, TrangThai:false},  
{SoGhe:14,TenGhe: "số 14", Gia:100, TrangThai:false},  
{SoGhe:15,TenGhe: "số 15", Gia:100, TrangThai:false},  
{SoGhe:16,TenGhe: "số 16", Gia:100, TrangThai:false},  
{SoGhe:17,TenGhe: "số 17", Gia:100, TrangThai:false},  
{SoGhe:18,TenGhe: "số 18", Gia:100, TrangThai:false},  
{SoGhe:19,TenGhe: "số 19", Gia:100, TrangThai:false},  
{SoGhe:20,TenGhe: "số 20", Gia:100, TrangThai:false},  
{SoGhe:21,TenGhe: "số 21", Gia:100, TrangThai:false},  
{SoGhe:22,TenGhe: "số 22", Gia:100, TrangThai:false},  
{SoGhe:23,TenGhe: "số 23", Gia:100, TrangThai:false},  
{SoGhe:24,TenGhe: "số 24", Gia:100, TrangThai:false},  
{SoGhe:25,TenGhe: "số 25", Gia:100, TrangThai:false},  
{SoGhe:26,TenGhe: "số 26", Gia:100, TrangThai:false},  
{SoGhe:27,TenGhe: "số 27", Gia:100, TrangThai:false},  
{SoGhe:28,TenGhe: "số 28", Gia:100, TrangThai:false},  
{SoGhe:29,TenGhe: "số 29", Gia:100, TrangThai:false},  
{SoGhe:30,TenGhe: "số 30", Gia:100, TrangThai:true},  
{SoGhe:31,TenGhe: "số 31", Gia:100, TrangThai:false},  
{SoGhe:32,TenGhe: "số 32", Gia:100, TrangThai:false},  
{SoGhe:33,TenGhe: "số 33", Gia:100, TrangThai:false},  
{SoGhe:34,TenGhe: "số 34", Gia:100, TrangThai:false},  
{SoGhe:35,TenGhe: "số 35", Gia:100, TrangThai:false},
```

Hướng dẫn cài đặt primeNG

<https://www.primefaces.org/primeng/#/setup>

Bước 1: npm install primeng --save

Bước 2: npm install primeicons --save

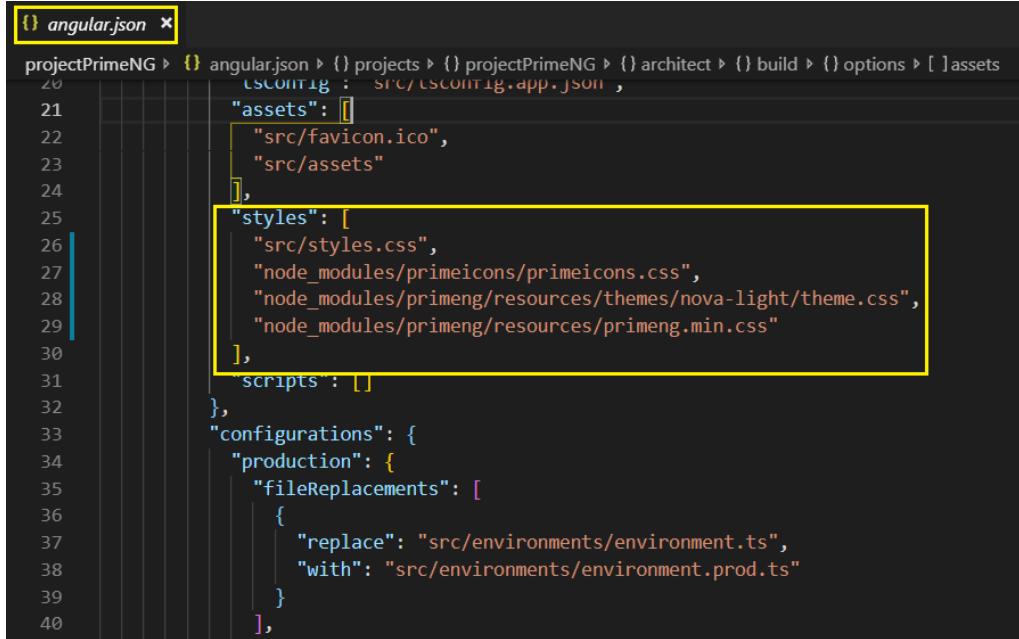
Bước 3: npm install @angular/animations --save

Bước 4: Tạo 1 module với tên primeNG.module.ts (Module chứa các thư viện control primeNG)

```
projectPrimeNG ▶ src ▶ app ▶ shared ▶ Modules ▶ prime-ng ▶ TS prime-ng.module.ts ▶ PrimeNgModule
1
2 import { NgModule } from '@angular/core';
3 import {BrowserAnimationsModule} from '@angular/platform-browser/animations'; //Thư viện animation
4
5 @NgModule({
6   imports: [
7     BrowserAnimationsModule //Những thư viện sử dụng các control primeNG được import vào đây
8   ],
9   exports:[
10     BrowserAnimationsModule //Những thư viện sử dụng các control primeNG được export ra để module nào sử dụng control primeNG thì import
11   ]
12 }
13 )
14 export class PrimeNgModule { }
```

Hướng dẫn cài đặt primeNG

Bước 5: đóng gói các thư viện css của primeNG (thuộc tính styles của file angular.json)



```
0 angular.json x
1
2 projectPrimeNG > {} angular.json > {} projects > {} projectPrimeNG > {} architect > {} build > {} options > [ ]assets
3   20     "esconfig": "src/esconfig.app.json",
4   21     "assets": [
5   22       "src/favicon.ico",
6   23       "src/assets"
7   24     ],
8   25     "styles": [
9   26       "src/styles.css",
10      "node_modules/primeicons/primeicons.css",
11      "node_modules/primeng/resources/themes/nova-light/theme.css",
12      "node_modules/primeng/resources/primeng.min.css"
13    ],
14    "scripts": []
15  },
16  "configurations": {
17    "production": {
18      "fileReplacements": [
19        {
20          "replace": "src/environments/environment.ts",
21          "with": "src/environments/environment.prod.ts"
22        }
23      ],
24    }
25  }
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```

Hướng dẫn cài đặt primeNG

Bước 6: cách sử dụng

Vào trang primeNG

The screenshot shows the PrimeNG documentation website. On the left, there's a sidebar with categories: Input, Button (highlighted with an orange border), Data, Panel, Overlay, File, and Menu. The main content area has tabs: Documentation (highlighted with an orange border) and Source. Below the tabs, there's a section titled 'Import' containing the code: `import {ButtonModule} from 'primeng/button';`. Under 'Getting Started', it says: 'Button is either applies as a component using p-button element or a directive on its own.' It shows two examples of HTML code:

```
<button pButton type="button" label="Click" ></button>
<p-button label="Click" ></p-button>
```

At the bottom, there's a snippet of Angular template code: `<p-button label="Click" icon="pi pi-check" (onClick)="Print()"></p-button>`.

A screenshot of the `prime-ng.module.ts` file. It contains the following code:

```
1 import { NgModule } from '@angular/core';
2 import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
3 import { ButtonModule } from 'primeng/button'; //Sử dụng button control primeNG
4
5 @NgModule({
6   imports: [
7     BrowserAnimationsModule, ButtonModule //Những thư viện sử dụng các control
8   ],
9   exports:[
10     BrowserAnimationsModule, ButtonModule //Những thư viện sử dụng các control
11   ]
12 })
13
14 )
15 export class PrimeNgModule { }
```

A red arrow points from the 'Documentation' tab on the PrimeNG website to this code block.

A screenshot of the `app.module.ts` file. It contains the following code:

```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { PrimeNgModule } from './shared/Modules/prime-ng/prime-ng.module';
7
8 @NgModule({
9   declarations: [
10     AppComponent
11   ],
12   imports: [
13     BrowserModule,
14     AppRoutingModule,
15     PrimeNgModule //Import PrimeNgModule here
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
```

A red arrow points from the 'Source' tab on the PrimeNG website to this code block. A yellow arrow points from the `PrimeNgModule` import statement in the `app.module.ts` code back to the corresponding line in the `prime-ng.module.ts` code.

Tại module nào cần sử dụng các control này thì import PrimeNgModule vào.

Bài tập:

The screenshot shows a user interface for managing student information. At the top, there is a search bar labeled "Tim kiem hoc vien" and a dropdown menu labeled "Lớp" containing "FrontEnd" and "myclass". Below this is a table titled "Thông tin học viên" with columns "Mã HV", "TenHV", and "Mã Lớp". The table contains 10 rows of data.

Mã HV	TenHV	Mã Lớp
11	Huynh Tuấn Anh	3
12	Phạm Tân Trường	3
13	Trịnh Minh Triết	3
14	Liễu Thanh Thanh	3
15	Thiều Ngọc Anh	3
6	Dương Hải Thái	2
7	Châu Ái Mỹ	2
8	Đàm Thu Trang	2
9	Lê Quang Minh	2
10	Phạm Công Trí	2

Yêu cầu:

- Dùng PrimeNG control để thiết kế giao diện và chức năng như hình

Gợi ý:

- Sử dụng primeNG button, AutoComplete để filter search, PrimeTable và PrimePanel hiển thị dữ liệu dạng bảng
- npm install @angular/cdk --save
- import {DropdownModule} from 'primeng/dropdown'

```
data: any[] = [ { MaHV: 1, TenHocVien:"Nguyễn Trần Trung Quân", MaLop:1},
```

```
• { MaHV: 2, TenHocVien:"Hồ Quang Hiếu", MaLop:1},  
• { MaHV: 3, TenHocVien:"Phạm Quỳnh Anh", MaLop:1},  
• { MaHV: 4, TenHocVien:"Đặng Trung Hiếu", MaLop:1},  
• { MaHV: 5, TenHocVien:"Lê Minh Long", MaLop:1},  
• { MaHV: 6, TenHocVien:"Dương Hải Thái", MaLop:2}, { MaHV: 7, TenHocVien:"Châu Ái Mỹ", MaLop:2},  
• { MaHV: 8, TenHocVien:"Đàm Thu Trang", MaLop:2},  
• { MaHV: 9, TenHocVien:"Lê Quang Minh", MaLop:2},  
• { MaHV: 10, TenHocVien:"Phạm Công Trí", MaLop:2},  
• { MaHV: 11, TenHocVien:"Huỳnh Tuấn Anh", MaLop:3},  
• { MaHV: 12, TenHocVien:"Phạm Tân Trường", MaLop:3},  
• { MaHV: 13, TenHocVien:"Trịnh Minh Triết", MaLop:3},  
• { MaHV: 14, TenHocVien:"Liễu Thanh Thanh", MaLop:3},  
• { MaHV: 15, TenHocVien:"Thiều Ngọc Anh", MaLop:3},  
• { MaHV: 16, TenHocVien:"Trương Ngọc Băng Di", MaLop:4},  
• { MaHV: 17, TenHocVien:"Trần Thiệu Tường", MaLop:4},  
• { MaHV: 18, TenHocVien:"Phạm Đức Thắng", MaLop:4},  
• { MaHV: 19, TenHocVien:"Trần Hồng Minh", MaLop:4},  
• { MaHV: 20, TenHocVien:"Thái Phuong Châu", MaLop:4}, ]
```

```
array: any[] = [  
{ MaLop: 1, TenLop: "cybersoft" }, { MaLop: 2, TenLop: "myclass" }, { MaLop: 3, TenLop: "FrontEnd" }, { MaLop: 4, TenLop: "FullStack" }];
```

Hướng dẫn cài đặt material angular

<https://material.angular.io/guide/getting-started>

Bước 1: npm I --save @angular/material @angular/cdk @angular/animations

Bước 2: ng add @angular/material

Bước 4: Tạo 1 module với tên material.module.ts (Module chứa các thư viện control material)

```
TS material.module.ts ×
1 import { NgModule } from '@angular/core';
2
3 import { MatButtonModule, MatCheckboxModule} from '@angular/material'; Nơi import những thư viện từ material angular
4 @NgModule({
5   imports: [MatButtonModule, MatCheckboxModule],//Sử dụng material control nào import material đó vào
6   exports: [MatButtonModule, MatCheckboxModule],//Export ra để các module khác khi cần import vào có thể sử dụng
7 })
8 export class MaterialModule { }
```

Hướng dẫn cài đặt material angular

Bước 5: cách dùng

Tham khảo cách sử dụng controls

Material Components CDK Guides

Checkbox

OVERVIEW API EXAMPLES

`<mat-checkbox>Check me!</mat-checkbox>` Sử dụng

Import control module cần sử dụng

```
material.module.ts ×
1 import { NgModule } from '@angular/core';
2
3 import { MatButtonModule, MatCheckboxModule } from '@angular/material';
4 @NgModule({
5   imports: [MatButtonModule, MatCheckboxModule], //Sử dụng material control nào import material đó vào
6   exports: [MatButtonModule, MatCheckboxModule], //Export ra để các module khác khi cần import vào có thể sử dụng
7 })
8 export class MaterialModule { }
```

Nơi import những thư viện từ material angular

Import material module vào module cần sử dụng

```
app.module.ts app.component.html ×
1
2 <mat-checkbox>Check me!</mat-checkbox>
3 
```

```
14 import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
15 @NgModule({
16   declarations: [
17     AppComponent,
18   ],
19   imports: [
20     BrowserModule, BrowserAnimationsModule, MaterialModule //Đối với một số control material có animation ta import vào thư viện này
21   ],
22   providers: [],
23   bootstrap: [AppComponent]
24 })
25 
```

```
app.module.ts ×
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 import { AppComponent } from './app.component';
4 import { [MaterialModule] } from './ShareModule/material/material.module';
5 @NgModule({
6   declarations: [
7     AppComponent
8   ],
9   imports: [
10     BrowserModule,
11     [MaterialModule], // Module nào cần sử dụng đến material thì import vào
12   ],
13   providers: [],
14   bootstrap: [AppComponent]
15 })
16 export class AppModule { }
```

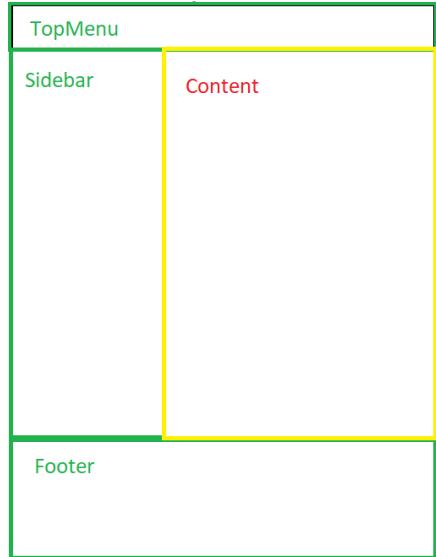
Module nào cần sử dụng đến material thì import vào

Template - Routing

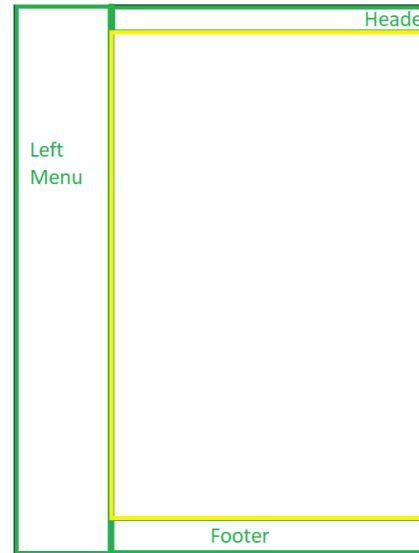
Template là gì ?

Template là những mẫ layout dàn trang, bố cục thiết kế sẵn. Thay vì ta phải xây dựng layout cho từng trang, ta chỉ cần dựng 1 template, các trang khác sẽ kế thừa template này.

Ví dụ:



Template HOME
Các trang khác sẽ có chung các thành phần: TopMenu, Sidebar, Footer.
Phần Content sẽ thay đổi từng trang



Template ADMIN
Các trang khác sẽ có chung các thành phần: Header, LeftMenu, Footer.
Phần Content sẽ thay đổi từng trang

Template - Routing

Routing là gì?

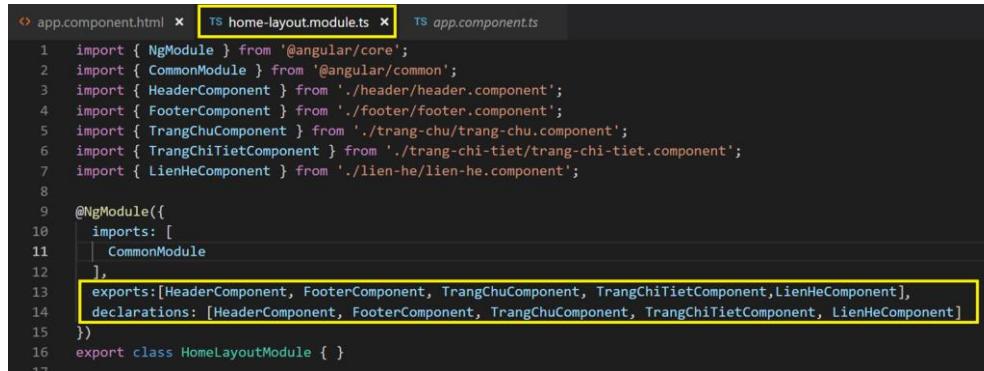
Routing là cơ chế quản lý các component, module dựa trên các url.

RoutingModule là module có trong @angular/router, giúp điều hướng và hiển thị nội dung theo url.

Các cách chia routes:

Cách 1: Chia routes theo component

Ta có các component: Header, Footer, TrangChu, TrangChiTiet, LienHe



```
app.component.html x home-layout.module.ts x app.component.ts
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { HeaderComponent } from './header/header.component';
4 import { FooterComponent } from './footer/footer.component';
5 import { TrangChuComponent } from './trang-chu/trang-chu.component';
6 import { TrangChiTietComponent } from './trang-chi-tiet/trang-chi-tiet.component';
7 import { LienHeComponent } from './lien-he/lien-he.component';
8
9 @NgModule({
10   imports: [
11     CommonModule
12   ],
13   exports:[HeaderComponent, FooterComponent, TrangChuComponent, TrangChiTietComponent,LienHeComponent],
14   declarations: [HeaderComponent, FooterComponent, TrangChuComponent, TrangChiTietComponent, LienHeComponent]
15 })
16 export class HomeLayoutModule { }
```

Template - Routing

Cách 1: Chia routes theo component

Thiết lập routes tại app.module.ts hoặc tạo 1 module mới có tên AppRoutingModule

```
app.component.html x  TS home-layout.module.ts x  TS app.module.ts x
4 import { AppComponent } from './app.component';
5 import { HomeLayoutModule } from './home-layout/home-layout.module';
6 import { TrangChuComponent } from './home-layout/trang-chu/trang-chu.component';
7 import { TrangChiTietComponent } from './home-layout/trang-chi-tiet/trang-chi-tiet.component';
8 import { LienHeComponent } from './home-layout/lien-he/lien-he.component';
9 import { Routes, RouterModule } from '@angular/router'; //Lớp đối tượng giúp chia link url (Định tuyến)
10
11 const appRoutes:Routes = [
12 {
13   //Khi người dùng chỉ gõ domain thì nó sẽ thay nội dung thẻ <router-outlet> bằng component TrangChu
14   path:"",component:TrangChuComponent
15 },
16 //Tương tự khai báo các đường dẫn khác cho website
17 {
18   // Hoặc người dùng gõ domain/trangchu :nội dung thẻ <router-outlet> bằng component TrangChu
19   path:"trangchu",component:TrangChuComponent
20 },
21 {
22   path:"trangchitiet",component:TrangChiTietComponent // domain/trangchitiet => load ComTrangChiTiet vào nội dung <router-outlet>
23 },
24 {
25   path:"lienhe",component:LienHeComponent // domain/trangchitiet => load ComTrangChiTiet vào nội dung <router-outlet>
26 },
27
28
29 @NgModule({
30   declarations: [
31     AppComponent
32   ],
33   imports: [
34     BrowserModule,HomeLayoutModule , RouterModule.forRoot(appRoutes) // Gán đối tượng appRoutes (khai báo các đường dẫn) vào đối tượng Router (vì là app nên dùng forRoot())
35   ],
36
37   bootstrap: [AppComponent]
38 })
39 export class AppModule { }
```

Class Routes dùng để khai báo các đường dẫn mà sẽ được thay thế vào nội dung thẻ <router-outlet>

1 đối tượng Routes (appRoutes) chứa 1 mảng các đường dẫn được liệt kê khi người dùng gõ trực tiếp vào url trên browser thì nó sẽ load component tương ứng vào nội dung thẻ <router-outlet>

Để làm được điều đó ta cần gắn đối tượng đó vào RouterModule (module định tuyến của toàn ứng dụng). Vì app chúng ta chạy khởi điểm từ node gốc là appCom nên ta dùng phương thức forRoot()

Template - Routing

Cách 1: Chia routes theo component

```
app.component.html ×
1 <app-header></app-header>
2 <router-outlet></router-outlet>
3 <app-footer></app-footer>
```

Phản nội dung sẽ được thay thế khi ta thay đổi url của website tương ứng với routes

```
header.component.html ×
1 <nav>
2   <a routerLink="/trangchu">Trang chủ</a>
3   <a routerLink="/trangchitiet">Chi tiết</a>
4   <a routerLink="/lienhe">Liên hệ</a>
5 </nav>
```

Thay vì bắt người dùng gõ đường dẫn trên trình duyệt, ta để các link thẻ a tương tự href

```
home-layout.module.ts ×
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { HeaderComponent } from './header/header.component';
4 import { FooterComponent } from './footer/footer.component';
5 import { TrangChuComponent } from './trang-chu/trang-chu.component';
6 import { TrangChiTietComponent } from './trang-chi-tiet/trang-chi-tiet.component';
7 import { LienHeComponent } from './lien-he/lien-he.component';
8 import { RouterModule } from '@angular/router';
9
10 @NgModule({
11   imports: [
12     CommonModule, RouterModule
13   ],
14   exports: [HeaderComponent, FooterComponent, TrangChuComponent, TrangChiTietComponent, LienHeComponent],
15   declarations: [HeaderComponent, FooterComponent, TrangChuComponent, TrangChiTietComponent, LienHeComponent]
16 })
17 export class HomeLayoutModule { }
```

Vì các thẻ a tại **header.component.html** có **routerLink**, nên ta cần **import RouterModule** vào **HomeLayoutModule.ts**

Template - Routing

Cách 1: Chia routes theo component



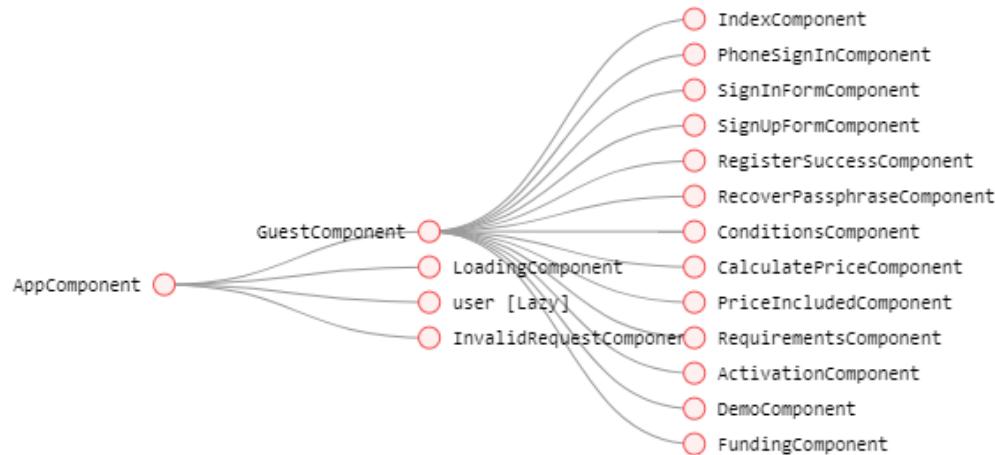
Kết quả: khi thay đổi url, **nội dung chỉ thay đổi tại nơi chứa thẻ router-outlet**. Hãy để ý khi ta thay đổi url bằng cách click vào link tại header, trang không cần phải load lại => điều này giúp cho google index website, đồng thời cải thiện tốc độ load của ứng dụng.

Template - Routing

Cách 2: Chia routes theo module và template

Khi ứng dụng lớn (có nhiều component), việc quản lý các routes như cách 1 là không hề đơn giản.

=> Giải pháp: quản lý routes theo module



Template - Routing

Cách 2: Chia routes theo module và template

Để dễ dàng tiếp cận, ta sẽ thực hiện demo:

Ví dụ: hệ thống có 2 module: HomeModule và AdminModule

HomeModule có các trang: TrangChu, TrangChiTiet, TrangDangNhap,...

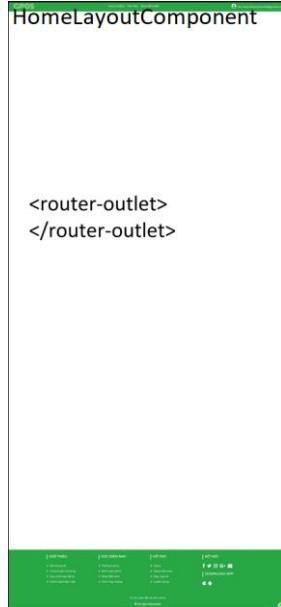
AdminModule có các trang: QuanLyPhim, QuanLyNguoiDung,...

Mỗi module có template khác nhau => Ta không thể làm cách 1 là chỉ có duy nhất 1 phần <router-outlet> => ứng với mỗi template, ta có 1 <router-outlet

Template - Routing

Ví dụ: Home template

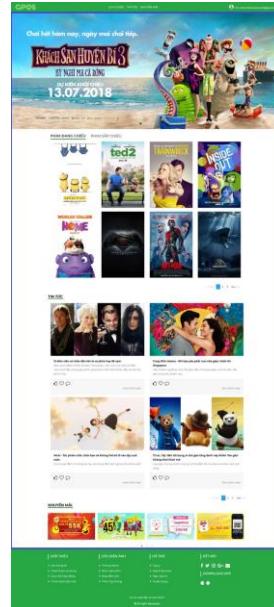
HomeModule



Router-
outlet

localhost:4200/home
Hoặc localhost:4200

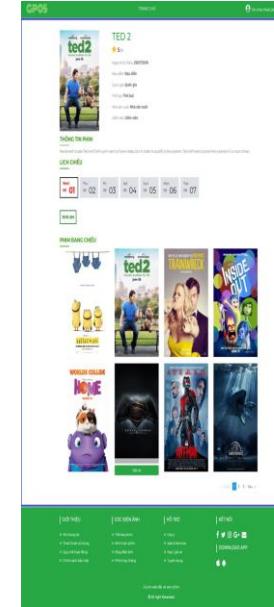
Trang Chu kế thừa từ
HomeLayout



Router-
outlet

localhost:4200/home/trangchu
Hoặc localhost:4200/trangchu

Trang Chi Tiet kế thừa từ
HomeLayout



localhost:4200/home/trangchitiet/1
Hoặc localhost:4200/trangchitiet/1

Template - Routing

Ví dụ: Home template

Cấu hình routes Home

```
home.module.ts      app.module.ts      home.layout.module.spec.ts
1 import { BrowserModule, Title } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5 import { HomeModule } from './home-layout/home.module';
6
7 import { Routes, RouterModule } from '@angular/router'; //Lớp đổi tượng giúp chia link url (Định tuyến)
8
9 const appRoutes: Routes = [
10   { path: '', loadChildren: () => HomeModule }, //Khi người dùng gõ vào localhost:4200 => web sẽ load module Home
11   { path: 'home', loadChildren: () => HomeModule } ////Khi người dùng gõ vào localhost:4200/home => web cũng sẽ load module Home
12 ]
13
14 @NgModule({
15   declarations: [
16     AppComponent
17   ],
18   imports: [
19     BrowserModule, RouterModule.forRoot(appRoutes) // Gán đổi tượng appRoutes (khai báo các đường dẫn) vào đối tượng Router (vì là app nên
20   ],
21
22   bootstrap: [AppComponent]
23 })
24 export class AppModule { }
```

Không cần import homeModule vào mục imports nữa lúc này appRoutes sẽ định tuyến module. Giảm tải cho việc quản lý lên app.component.ts

↓

app.module.ts

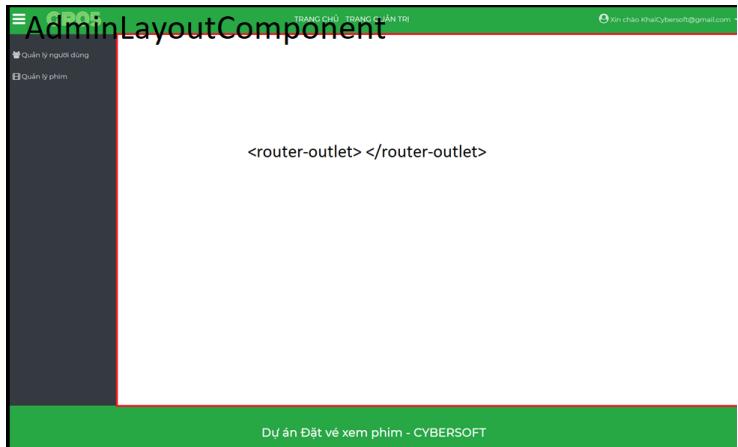
home.module.ts

Định tuyến trên HomeModule

```
home.module.ts      app.module.ts      home.layout.module.spec.ts
4 import { FooterComponent } from './footer/footer.component';
5 import { TrangChuComponent } from './trang-chu/trang-chu.component';
6 import { TrangChiTietComponent } from './trang-chi-tiet/trang-chi-tiet.component';
7 import { LienHeComponent } from './lien-he/lien-he.component';
8 import { RouterModule, Routes } from '@angular/router';
9 import { HomeLayoutComponent } from './home-layout/home-layout.component';
10
11 const homeRoutes: Routes = [
12   {
13     path: '', component: HomeLayoutComponent, children: [ //Trong HomeLayoutComponent ta cũng bố trí 1 thẻ router-outlet
14       {
15         path: '', component: TrangChuComponent, //TrangChuComponent là một ứng dụng component trang chủ
16         // Nếu người dùng /home và không /giá thì nó sẽ load tại router-outlet của HomeLayoutComponent
17         path: '', component: TrangChuComponent
18       },
19       {
20         //hoặc định tuyến thêm 1 đường dẫn khác cũng gọi về component trangchu
21         //http://localhost:4200/home/trangchitiet(theo nhánh module Home) hoặc http://localhost:4200/ (theo nhánh module '')
22         //http://localhost:4200/home/trangchitiet(theo nhánh module Home) hoặc http://localhost:4200/trangchu (theo nhánh module '')
23         path: 'trangchu', component: TrangChuComponent
24       },
25       {
26         //Tương tự load http://localhost:4200/home/trangchitiet (theo nhánh module home)
27         //http://localhost:4200/trangchitiet (theo nhánh module '')
28         path: 'trangchitiet', component: TrangChiTietComponent
29       },
30     ]
31   }
32 ]
33 @NgModule({
34   imports: [
35     CommonModule, RouterModule.forChild(homeRoutes) // Gán homeRoutes vào RouterModule
36   ],
37   //Không phải export ra nữa vì khi người dùng trỏ về /home thì nó sẽ load ra module này
38   declarations: [HeaderComponent, FooterComponent, TrangChuComponent, TrangChiTietComponent, LienHeComponent, HomeLayoutComponent]
39 })
40 export class HomeModule { }
```

Template - Routing

Ví dụ: Admin template



localhost:4200/admin

localhost:4200/admin/quanlynguoidun

GPOS

TRẠNG CHỦ TRẠNG QUẢN TRỊ

Kin chào KhaiCybersoft@gmail.com

Quản lý người dùng

Quản lý phim

DANH SÁCH NGƯỜI DÙNG

Tìm kiếm

Tài khoản	Họ tên	Email	Số điện thoại	Loại người dùng	X	Y
admin1	Nguyễn Văn A	admin1@gmail.com	1234567789	Quản trị		
admin1g05	GPOS	admin1g05@gmail.com	1234567789	Quản trị		
ben123456	ben123456	ben123456@gmail.com	09234567890	Khách hàng		
ben1234567	ben1234567	ben1234567@gmail.com	09324567890	Khách hàng		
ben111	binh	binh@gmail.com	095909099	Khách hàng		
demo1234	demo1234	demo@gmail.com	09234567890	Quản trị		
demo123456	demo123456	demo@gmail.com	09234567890	Khách hàng		
nguoiluong1	nguoiluong1	nguoiluong1@gmail.com	09234567890	Quản trị		
nguoiluong123	nguoiluong123	nguoiluong123@gmail.com	9234567890	Khách hàng		
talkhoan3	Tài khoản 3	talkhoan3@gmail.com	1234566666	Khách hàng		

Tổng người dùng: 11

Dự án Đặt vé xem phim - CYBERSOFT

localhost:4200/admin/quanlyphi

localhost:4200/admin/quanlyphi

GPOS

TRẠNG CHỦ TRẠNG QUẢN TRỊ

Kin chào KhaiCybersoft@gmail.com

Quản lý người dùng

Quản lý phim

DANH SÁCH PHIM

Tìm kiếm

Mã phim	Tên phim	Ngày khởi chiếu	Đánh giá	X	Y
5	Minions	28/07/2018	5 ⚡		
15	Ted 2	28/07/2018	5 ⚡		
25	Trainwreck	28/07/2018	5 ⚡		
35	Inside Out	28/07/2018	5 ⚡		
45	Home	28/07/2018	5 ⚡		
55	Batman v Superman: Dawn of Justice	28/07/2018	5 ⚡		
65	Ant-Man	28/07/2018	5 ⚡		
75	Jurassic World	28/07/2018	5 ⚡		
85	Fantastic Four	28/07/2018	5 ⚡		
95	Mad Max: Fury Road	28/07/2018	5 ⚡		

Tổng phim: 19

Dự án Đặt vé xem phim - CYBERSOFT

Template - Routing

Cấu hình routes Admin

```
home.module.ts app.module.ts home-layout.module.spec.ts
1 import { BrowserModule, Title } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5 import { HomeModule } from './home-layout/home.module';
6
7 import { Routes, RouterModule } from '@angular/router'; //Lớp đổi tượng giúp chia link url (Định tuyến)
8 import { AdminModule } from './admin/admin.module';
9
10 const appRoutes: Routes = [
11   { path: '', loadChildren: () => HomeModule }, //khi người dùng gõ vào localhost:4200 => web sẽ load module Home
12   { path: 'home', loadChildren: () => HomeModule }, //khi người dùng gõ vào localhost:4200/home => web cũng sẽ load module Home
13   { path: 'admin', loadChildren: () => AdminModule } //localhost:4200/admin => load module admin
14 ];
15
16 ]
17
18 @NgModule({
19   declarations: [
20     AppComponent
21   ],
22   imports: [
23     BrowserModule, RouterModule.forRoot(appRoutes) // Gán đổi tượng appRoutes (khai báo các đường dẫn) vào đối tượng Router (vì )
24   ],
25   bootstrap: [AppComponent]
26 })
27 export class AppModule {}
```

Người dùng gõ localhost:4200/admin => layout admin.
Trong AdminModule sẽ có component Layout và các trang con kế thừa từ layout

app.module.ts

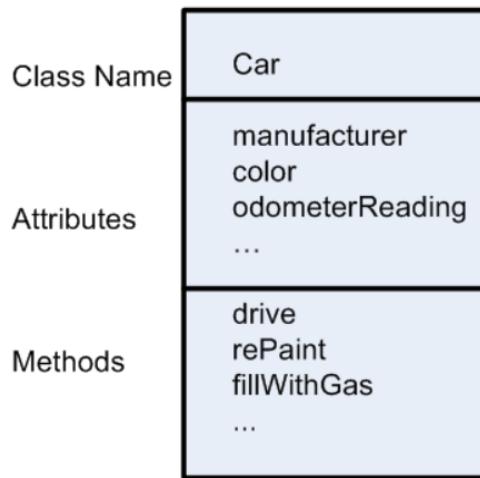
```
home.module.ts
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { AdminLayoutComponent } from './admin-layout/admin-layout.component';
4 import { QuanLyNguoiDungComponent } from './quan-ly-nguoi-dung/quan-ly-nguoi-dung.component';
5 import { QuanLyPhimComponent } from './quan-ly-phim/quan-ly-phim.component';
6 import { Routes, RouterModule } from '@angular/router';
7
8 const adminRoutes: Routes = [
9   {
10     path: '',
11     component: AdminLayoutComponent,
12     children: [ //Trong AdminLayoutComponent ta cung bo tri i the <router-outlet>
13       { path: 'quanlynguoidung', component: QuanLyNguoiDungComponent }, //localhost:4200/admin/quanlynguoidung => load ra component QuanLyNguoiDungComponent
14       { path: 'quanlyphim', component: QuanLyPhimComponent } //localhost:4200/admin/quanlyphim => load ra component QuanLyPhimComponent
15     ]
16   }
17
18 @NgModule({
19   imports: [
20     CommonModule, RouterModule.forChild(adminRoutes)
21   ],
22   declarations: [AdminLayoutComponent, QuanLyNguoiDungComponent, QuanLyPhimComponent]
23 })
24 export class AdminModule {}
```

Template admin

Các trang con kế thừa từ Template admin. Nội dung sẽ được load vào thẻ <router-outlet>

Object Class

Object class (TypeScript đã học) là prototype



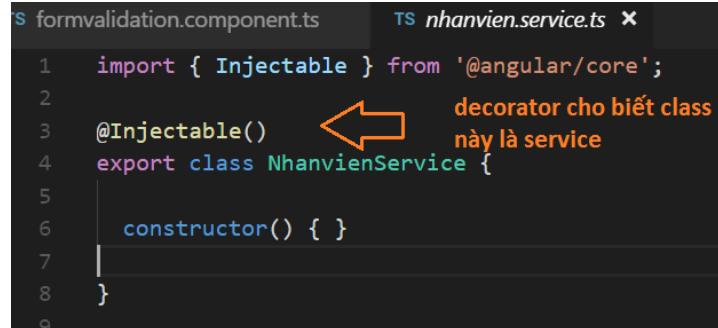
Cú pháp tạo class: `ng g class [tên class]`

Service

Service là một class đóng gói một vài chức năng và cung cấp các phần còn lại của ứng dụng (component).

Thông thường, service chứa các phương thức thao tác dữ liệu back-end thông qua GET, POST, PUT, DELETE được hỗ trợ bởi HttpClient và Observable của Angular.

Ngoài ra, ta còn có thể trung chuyển dữ liệu giữa các component thông qua service



```
formvalidation.component.ts      TS nhanvien.service.ts x
1  import { Injectable } from '@angular/core';
2
3  @Injectable()    ← decorator cho biết class
4  export class NhanvienService {
5
6    constructor() { }
7
8  }
```

Khởi tạo service: `ng g service [tên service]`

Service

Sử dụng service: sử dụng service tại component nào thì import service tại component đó. Nếu muốn sử dụng service cho toàn ứng dụng thì import service vào app.module

```
import { Component, OnInit } from '@angular/core';
import { NhanvienService } from '../../../../../service/nhanvien.service';

@Component({
  selector: 'app-testservice',
  template: `
    <div *ngFor='let nv of DanhSachNhanVien'>
      Mã NV:{{nv.manv}}, Họ tên: {{nv.hoten}}, Năm sinh: {{nv.namsinh}}
    </div>`,
  styleUrls: ['./testservice.component.css']
})
export class TestserviceComponent implements OnInit {
  public DanhSachNhanVien:Array<any>;

  constructor(nvService:NhanvienService) {
    //Gọi service để lấy dữ liệu danh sách nhân viên hiển thị ra component
    this.DanhSachNhanVien = nvService.GetNhanVien();
  }
}
```

Service

service

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

@Injectable()
export class NhanvienService {
  //Danh sách nhân viên hiện tại là any sau này nó sẽ là kiểu dữ liệu của Class object
  private DanhSachNhanVien:Array<any> = [
    {manv:'01',hoten:'nhân viên 1',namsinh:'1993'},
    {manv:'02',hoten:'nhân viên 2',namsinh:'1991'},
    {manv:'03',hoten:'nhân viên 3',namsinh:'1992'},
    {manv:'04',hoten:'nhân viên 4',namsinh:'1990'}
  ]
  //Phương thức trong service
  public GetNhanVien():Array<any>
  {
    //Sau này tại đây ta sẽ liên kết với api từ backend (server) để lấy dữ liệu
    return this.DanhSachNhanVien;
  }
  constructor() { }
}
```

component

```
import { Component, OnInit } from '@angular/core';
import { NhanvienService } from '../../../../../service/nhanvien.service';

@Component({
  selector: 'app-testservice',
  template: `
    <div *ngFor='let nv of DanhSachNhanVien'>
      Mã NV:{{nv.manv}}, Họ tên: {{nv.hoten}}, Năm sinh: {{nv.namsinh}}
    </div>`,
  styleUrls: ['./testservice.component.css']
})
export class TestserviceComponent implements OnInit {
  public DanhSachNhanVien:Array<any>;
  constructor(nvService:NhanvienService) {
    //Gọi service để lấy dữ liệu danh sách nhân viên hiển thị ra component
    this.DanhSachNhanVien = nvService.GetNhanVien();
  }
}
```

module

```
TS formvalidation.component.ts      TS app.module.ts      TS testservice.module.ts      TS nhanvien.service.ts
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { TestserviceComponent } from './testservice/testservice.component';
4  import { NhanvienService } from '../../../../../service/nhanvien.service';
5  @NgModule({
6    imports: [
7      CommonModule
8    ],
9    exports:[TestserviceComponent],
10   declarations: [TestserviceComponent],
11   providers:[NhanvienService] //Import service vào module này
12 })
13 export class TestserviceModule { }
```

Service

Từ phiên bản 6, angular tự động thêm thuộc tính providedIn: 'root' vào file service, giúp lược bỏ bước import và provider service ở module

Sử dụng service, ta chỉ cần truyền vào hàm khởi tạo của component

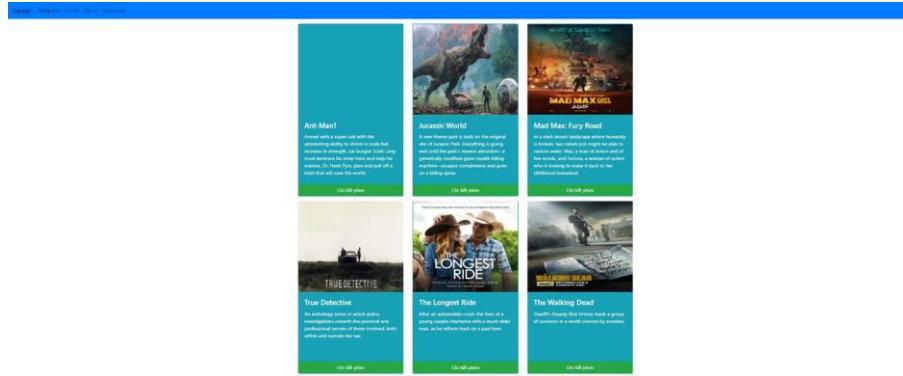
```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class PhimService {

  constructor() { }
}
```

Bài tập 9:

- Cho mảng dữ liệu phim
- Xây dựng service QuanLyPhim
- Tạo class đối tượng Phim và DanhSachPhim
- Xây dựng 1 trang component QuanLyPhim trong HomeModule
- Thực hiện load dữ liệu phim ra như hình bên dưới



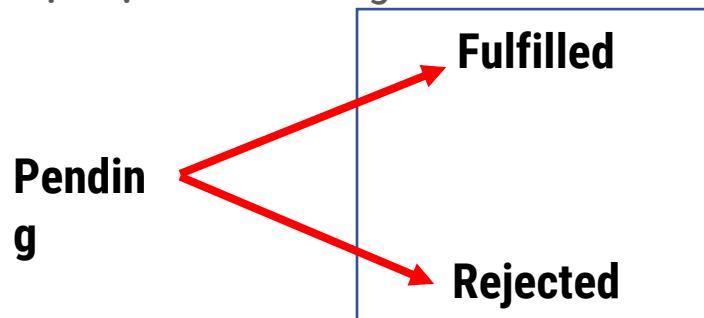
Observable

Promise

Promise và Observable đều là các kỹ thuật dùng để xử lý dữ liệu bất đồng bộ trong javascript nói chung và Angular nói riêng.

Promise tương tự như ajax, kết quả trả về từ một hành động sẽ có các trạng thái:

- Fulfilled: thành công làm gì
- Rejected: thất bại làm gì
- Pending: chờ xử lý hoặc bị từ chối làm gì



Fulfilled, Rejected gọi
là Settled tức là đã xử
lý xong

```

function ShowMessage()
{
    var promise = new promise(successData, failData);
    var message = 'hello';
    var request = new XMLHttpRequest();

    request.open('GET', 'url');
    request.onload = function() {
        if (request.status == 200) {
            //Hàm tham số đầu tiên trong biểu thức then (Thành công)
            successData(request.responseText,request.status);
        } else {
            //Hàm tham số đầu tiên trong biểu thức then (Thất bại)
            failData(request.statusText,request.status);
        }
    };
    //Gọi promise:
    promise.then(successData,failData).cache(function(){
        console.log("error")
    });
}

```

```

function successData(result,status)
{
    message = result; //(1)
    alert(message);
}
function failData(error,status)
{
    message = error; //(2)
    alert(error)
}

```

 Kết quả của 1 hàm promise sẽ trả về 1 đối tượng promise nên setup tất cả các hàm xử lý chỉ cần .then() để quản lý các promise.

Observable

Observable cũng như promise, là kỹ thuật xử lý dữ liệu bất đồng bộ. Observable trong Angular có các chức năng tối ưu hơn như:

- Khả năng trả về nhiều kết quả (đối với promise, sau mỗi lần xử lý dữ liệu, nó chỉ trả về một giá trị duy nhất)
- Khả năng huỷ bỏ request (đối với promise, đã request thì không có cách huỷ bỏ request đó)
- Khả năng retry
- Khả năng xử lý kết quả trước khi trả về (Với Promise, do đặc tính chỉ trả về một kết quả duy nhất nên việc xử lý kết quả có thể thực hiện ở bước **sau khi đã nhận dc kết quả**)

Ví dụ như bài toán đầu tiên message in ra là hello thay vì là 1 trong 2 giá trị tương lai nó sẽ trả về thì observable nó nhận cả 2 giá trị tại 2 thời điểm và binding lên model. Thêm 1 điều nữa request được gửi đi ở observable thì có thể hủy được.

HttpClient (HttpClientModule)

HttpClientModule cung cấp service dùng để giao tiếp với back-end (server) thông qua một số phương thức như GET, POST, PUT, DELETE...

```
TS app.module.ts x
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5 import { AppRoutingModule } from './app-routing.module';
6 import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
7 import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';
8 import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
9 import { ModalComponent } from './modal/modal.component';
10 import { AuthInterceptor } from './_core/guards/jwt.interceptor';
11
12 @NgModule({
13   declarations: [AppComponent, PageNotFoundComponent, ModalComponent],
14   imports: [
15     BrowserModule,
16     AppRoutingModule,
17     HttpClientModule, // highlighted
18     BrowserAnimationsModule
19   ],
20   providers: [
21     {
22       provide: HTTP_INTERCEPTORS,
23       useClass: AuthInterceptor,
24       multi: true
25     }
26   ],
27   bootstrap: [AppComponent]
```

GET: thường dùng để lấy dữ liệu

POST: thường dùng để tạo mới dữ liệu

PUT: thường dùng để cập nhật dữ liệu

DELETE: thường dùng để xoá dữ liệu

Để dùng được HttpClient, ta import HttpClientModule vào toàn ứng dụng thông qua appModule

Thực Hành: HTTP - GET

```
ts quan-ly-phim.service.ts ×
1  import { Injectable } from '@angular/core';
2  import { Observable, throwError } from 'rxjs';
3  import { HttpClient } from '@angular/common/http';
4  import { tap, catchError } from 'rxjs/operators';
5
6  @Injectable({
7    providedIn: 'root'
8  })
9  export class QuanLyPhimService {
10    constructor(private http: HttpClient) { }
11
12    getListMovie(): Observable<any> {
13      const url = `http://movie0706.cybersoft.edu.vn/api/QuanLyPhim/LayDanhSachPhim?maNhom=GP01`;
14      return this.http.get(url).pipe(
15        tap((data: any) => {
16          //Loading
17        }),
18        catchError(err => {
19          return this.handleError(err);
20        })
21      );
22    }
}
```

Import Observable

Import HttpClient

Khai báo HttpClient

Api để lấy danh sách phim từ back-end

Phương thức get của HttpClient

Tap: 1 operator trong RXJS, xử lý dữ liệu trả về

Kết quả trả về được chứa trong pipe, là 1 observable

Thực Hành: HTTP - GET

```
ts quan-ly-phim.service.ts      ts danh-sach-phim.component.ts ×
1  import { Component, OnInit, OnDestroy } from '@angular/core';
2  import { QuanLyPhimService } from '../../../../../core/services/quan-ly-phim.service';
3  import { Subscription } from 'rxjs';
4
5  @Component({
6    selector: 'app-danh-sach-phim',
7    templateUrl: './danh-sach-phim.component.html',
8    styleUrls: ['./danh-sach-phim.component.scss']
9  })
10 export class DanhSachPhimComponent implements OnInit, OnDestroy {
11   danhSachPhim: any = [];
12   private subListMovie = new Subscription();
13
14   constructor(
15     private quanLyPhimService: QuanLyPhimService,
16   ) {}
17
18   ngOnInit() {
19     this.layDanhSachPhim();
20   }
21
22   layDanhSachPhim() {
23     this.quanLyPhimService.getListMovie().subscribe(data => {
24       this.danhSachPhim = data;
25       console.log(this.danhSachPhim);
26     });
27   }
28
29   ngOnDestroy() {
30     this.subListMovie.unsubscribe();
31   }
}
```

Khai báo sử dụng service

Subscribe theo dõi biến đổi observable, có sự thay đổi sẽ tự động gọi hàm bên trong nó

Truyền dữ tham số thông qua url (RouterLink)

Truyền tham số là gì ? Tại sao phải truyền tham số?

Cũng tương tự input trong component ta thiết kế 1 trang hay 1 component đều muốn tái sử dụng được cho nhiều nội dung khác nhau. Tương tự đối với các thành phần layout cũng vậy.

=> Ví dụ: Các bạn xây dựng 1 layout để hiển thị thông tin chi tiết của 1 phim hay 1 sản phẩm, bài viết ... như hình bên dưới. Đều muốn chỉ xây dựng duy nhất 1 layout thay vì mỗi sản phẩm dàn mỗi layout => 10 sản phẩm ta có 10 layout (Không ai làm như vậy cả). Giống như input trong component người ta dùng tham số để đưa vào các RouteLink các giá trị, ứng với giá trị nào sẽ show thông tin với phần nội dung tương ứng.

Truyền dữ tham số thông qua url (RouterLink)

The screenshot shows a web browser window with the URL nhaccuautui.com/bai-hat/chan-ai-orange-ft-khoi-ft-chau-dang-khoa-ybwdiMjKeYD9.html highlighted with a red box. The page displays the song 'Chân Ái' by Orange, Khói, Châu Đăng Khoa. The video player shows the first few seconds of the video. To the right, there's a sidebar with a 'GỢI Ý DÀNH CHO BẠN' section, a 'NGHE TIẾP' section listing songs like 'Tâm Trí Yêu Em' and 'Chàng Võ Nàng', and a 'TÌM QUSIC CỦA TÔI' sidebar.

Chân Ái - Orange, Khói, Châu Đăng Khoa

Nghe nhạc > Bài hát Nhạc Trẻ > Orange, Khói, Châu Đăng Khoa

Chân Ái - Orange, Khói, Châu Đăng Khoa

7.609.296

Orange, Khói, Châu Đăng Khoa

cần một người

00:03 / 04:23

128kbps

Cung cấp bởi:
NCT Music Distribution

Thêm vào Chia sẻ Thích 7.4K Chia sẻ

Ca khúc Chân Ái do ca sĩ Orange, Khói, Châu Đăng Khoa thể hiện, thuộc thể loại Nhạc Trẻ. Các bạn có thể nghe, download (tài nhạc) bài hát chan ai mp3, playlist/album, MV/Video chan ai miễn phí tại NhacCuaTui.com.

Lời bài hát: Chân Ái

Nhạc sĩ: Châu Đăng Khoa

Lời đăng bởi: dauahii

Tự dừng buồn thế

Tự dừng lâng lâ

GỢI Ý DÀNH CHO BẠN

Thưởng thức những ca khúc phù hợp nhất với bạn

▶ Nghe bài hát

NGHE TIẾP

Tâm Trí Yêu Em
Kiến Vinh

Chàng Võ Nàng
Rum

SAY
Lena Lena, T.R.I

6 Năm Không Gặp
T-Akayz

Có Một Ngày Buồn Như T...
Nguyễn Hà

xem thêm

Có 2 trường hợp truyền nhận dữ liệu thông qua link (routerlink) và sự kiện.

- ❖ Truyền nhận 1 tham số
- ❖ Truyền nhận nhiều tham số

Thực hành truyền tham số

Bước 1: Trong HomeModule cấu hình route cho trang chi tiết như hình bên dưới

```
16 const homeRoutes:Routes = [
17   {
18     path: '',component:LayoutComponent,children:
19     [
20       {path: '',component:TrangChuComponent},
21       {path: 'trangchu',component:TrangChuComponent},
22
23       {path: 'chitiet/:id',component:ChiTietComponent},
24
25       {path: 'datve',component:DatVeComponent},
26       {path: 'pipe',component:PipeComponent},
27       {path: 'quanlynv',component:QuanLyNhanVienComponent},
28       {path: 'quanlyphim',component:QuanLyPhimComponent},
29     ]
30   }
31 ]
32
33 @NgModule({
34   imports: [
35     CommonModule,RouterModule.forChild(homeRoutes),FormsModule
36   ],
37   declarations: [TrangChuComponent, ChiTietComponent, DatVeComponent,LayoutComponent, PipeComponent, Shortcu
38
39 })
40 export class HomeModule { }
```

Thực hành truyền tham số

Bước 2: Tại QuanLyPhimComponent.html ta để đường link dẫn đến trang chi tiết như hình bên dưới.

```
TS quan-ly-phim.service.ts ✘ TS quan-ly-phim.component.ts ✘ quan-ly-phim.component.html ✘ TS home.module.ts  
angularfe10 › src › app › home › quan-ly-phim › quan-ly-phim.component.html › ...  
1  <div class="container mt-3">  
2      <div class="row">  
3          <div class="col-md-4 mb-3" *ngFor="let phim of dsPhim.MangPhim">  
4              <div class="card text-white bg-info">  
5                  <img class="card-img-top" height="300" [src]="phim.HinhAnh" alt="">  
6                  <div class="card-body">  
7                      <div style="height:150px; overflow: hidden;" class="card-text">{{phim.MoTa}}</div>  
8                  </div>  
9                  <a routerLink="/chitiet/{{phim.MaPhim}}" class="btn btn-success">Chi tiết phim</a>  
10                 </div>  
11             </div>  
12         </div>  
13     </div>
```

Thực hành truyền tham số

Bước 3: Sử dụng đối tượng activatedRoute để lấy tham số và dùng service gửi về backend

```
TS quan-ly-phim.service.ts      TS chi-tiet.component.ts ✘   TS quan-ly-phim.component.ts      TS quan-ly-phim.component.html      TS home.module.ts
angular10 › src › app › home › chi-tiet › TS chi-tiet.component.ts › ...
1  import { Component, OnInit, OnDestroy } from '@angular/core';
2  import { ActivatedRoute } from '@angular/router'; ← Lớp đối tượng dùng để lấy tham số từ url
3  import { QuanLyPhimService } from 'src/app/core/Services/quan-ly-phim.service';
4  import { Subscription } from 'rxjs';
5
6  @Component({
7    selector: 'app-chi-tiet',
8    templateUrl: './chi-tiet.component.html',
9    styleUrls: ['./chi-tiet.component.css']
10 })
11 export class ChiTietComponent implements OnInit,OnDestroy {
12   phim: any = {};
13   private MaPhim;
14   private subParam: Subscription;
15   constructor(
16     private activateRouter: ActivatedRoute, //Đối tượng dùng để lấy tham số từ url
17     private qlPhimService: QuanLyPhimService
18   ) {}
19
20   ngOnInit() {
21     this.subParam = this.activateRouter.params.subscribe((params) => {
22       this.MaPhim = params.MaPhim;
23     });
24
25     this.qlPhimService.LayChiTietPhim(this.MaPhim).subscribe((phim: any) => {
26       this.phim = phim
27       console.log(this.phim)
28     });
29   }
30   ngOnDestroy(){
31     this.subParam.unsubscribe();
32   }
33 }
34 }
```

Thông qua đối tượng activatedRoute dùng thuộc tính params để lấy tham số từ url

Từ tham số thông qua service phim ta gọi api để lấy dữ liệu từ backend về thể hiện ra giao diện

Truyền nhận nhiều tham số

Bước 1: Tạo 1 route bình thường không có tham số

```
const homeRoutes: Routes = [
  {
    path: '', component: LayoutComponent, children:
    [
      {path: '', component: TrangChuComponent},
      {path: 'trangchu', component: TrangChuComponent},

      // {path: 'chitiet/:id', component: ChiTietComponent},
      {path: 'chitiet', component: ChiTietComponent},
      {path: 'datve', component: DatVeComponent},
      {path: 'pipe', component: PipeComponent},
      {path: 'quanlynv', component: QuanLyNhanVienComponent},
      {path: 'quanlyphim', component: QuanLyPhimComponent},
    ]
  }
]
```

Truyền nhận nhiều tham số

Bước 2: Tại trang quản lý phim thay vì ta truyền routerlink dạng 1 tham số sau khí tự '/' cuối cùng. Thì ta có thể truyền thông qua thuộc tính queryParams. Hoặc có thể viết link?thamso1=[giatri]&thamso2=[giatri] => có 2 cách viết

```
angularfe10 ▶ src ▶ app ▶ home ▶ quan-ly-phim ▶ quan-ly-phim.component.html ▶ div.container.mt-3 ▶ div.row ▶ div.col-md-4.mb-3 ▶ div
1
2   <div class="container mt-3">
3     <div class="row">
4       <div class="col-md-4 mb-3" *ngFor="let phim of dsPhim.MangPhim">
5         <div class="card text-white bg-info">
6           <img class="card-img-top" height="300" [src]="phim.HinhAnh" alt="">
7           <div class="card-body">
8             <div style="height:150px; overflow: hidden;" class="card-text">{{phim.MoTa}}</div>
9           <!-- <a routerLink="/chitiet?MaPhim={{phim.MaPhim}}&TenPhim={{phim.TenPhim}}" class="btn btn-success">Chi tiết phim</a>
10          <a [routerLink]=["/chitiet"]
11            [queryParams]={MaPhim:phim.MaPhim,TenPhim:phim.TenPhim} |
12            class="btn btn-success">Chi tiết phim</a>
13
14
15        </div>
16      </div>
17    </div>
18  </div>
```

Truyền nhận nhiều tham số

Bước 3: Để lấy thông tin các tham số từ routink ta cũng dùng đối tượng activateRouter thông qua thuộc tính queryparam.

```
ngOnInit() {  
  
    this.subParam = this.activateRouter.queryParams.subscribe((params) => {  
        this.MaPhim = params.MaPhim;  
        this.TenPhim = params.TenPhim;  
    })  
}
```

Form – Form Validation (Form)

Dùng để hỗ trợ việc lấy giá trị từ Form người dùng nhập vào đưa vào biến **model** (thuộc tính kiểu dữ liệu object ta tạo trong class của component đó). Trong phần two-way binding chúng ta đã thao tác với FormsModule 1 lần rồi.

Form - validation

The screenshot shows a simple form with three input fields and one button:

- Email: A text input field.
- Name: A text input field.
- School: A text input field.
- Submit: A button labeled "Submit".

```
37 export class FormvalidationComponent implements OnInit {  
38   public schools:any = [{id:'1',name:'cybersoft'}, {id:'1',name:'myclass'}];  
39  
40   constructor() { }  
41   Submit(value:any)  
42   {  
43     console.log(value);  
44   }  
45   ngOnInit() {  
46   }  
47 }  
48 }
```

```
<form #registerForm="ngForm" class="form-horizontal" (ngSubmit) = "Submit(registerForm.value)">  
  <div class="form-group">  
    <label for="inputEmail3" class="col-sm-2 control-label">Email</label>  
    <div class="col-sm-10">  
      <input type="email" class="form-control" id="inputEmail3" name="email" placeholder="Email" ngModel>  
    </div>  
  </div>  
  <div class="form-group">  
    <label for="Name" class="col-sm-2 control-label">Name</label>  
    <div class="col-sm-10">  
      <input type="text" class="form-control" id="Name" name="name" placeholder="Name" ngModel>  
    </div>  
  </div>  
  <div class="form-group">  
    <label for="school" class="col-sm-2 control-label">School</label>  
    <div class="col-sm-10">  
      <select id="school" class="form-control" name="school" ngModel>  
        <option *ngFor='let school of schools' [value]="school.id">{{school.name}} </option>  
      </select>  
    </div>  
  </div>  
  <div class="form-group">  
    <div class="col-sm-offset-2 col-sm-10">  
      <button type="submit" class="btn btn-default">Submit</button>  
    </div>  
  </div>  
</form>
```

1/ Đối tượng **ngForm** sẽ lấy các giá trị từ các **ngModel** thông qua thuộc tính **name**.
2/ Sau đó ta sẽ dùng sự kiện **(submit)** gọi đến phương thức xử lý do chúng ta định nghĩa: **Submit** vào tham số là đối tượng **ngForm** (biến **registerForm.value**)

Form – Form Validation (Form)

Trong angular 6 hỗ trợ ta 1 module dành cho việc nhập liệu Form và kiểm tra giá trị đó là: **FormsModule**.

```
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { FormvalidationComponent } from './formvalidation/formvalidation.component';
4 //Import FormsModule
5 import {FormsModule} from '@angular/forms';
6 @NgModule({
7   imports: [
8     CommonModule,FormsModule
9   ],
10  exports:[FormvalidationComponent],
11  declarations: [FormvalidationComponent]
12 })
13 export class FormvalidationModule { }
```

import FormsModule tại module cần sử dụng FormValidation

Form – Form Validation (Form)

❖ Trạng thái của giá trị đầu vào thẻ input

- : **untouched** : Trường này chưa được chạm vào.
- : **touched** : Trường này đã được chạm vào.
- : **pristine** : Trường này chưa được thay đổi.
- : **dirty** : Trường này đã được thay đổi.
- : **invalid** : Trường có nội dung không hợp lệ.
- : **valid** : Trường có nội dung hợp lệ.

Giá trị trả lại khi ta kiểm tra các trạng thái sẽ là **true**. Nếu ngược lại thì sẽ là **false**

Form – Form Validation (Form)

❖ Kiểm tra rỗng với required

```
<input type="email" class="form-control" id="email" name="email" placeholder="Email" #email='ngModel' required
<! -- Kiểm tra rỗng --
<div *ngIf="email.errors && (email.dirty || email.touched)">
    <div *ngIf="email.errors.required"> ← Xét trong lỗi đó có
        Email không được rỗng !
    </div>| ← Nếu có sự thay đổi của thẻ input#email
    </div>| hoặc #email được chạm vào. && #email
    </div>| errors (required, không đúng định dạng,
    </div>| quá ký tự ...) thì sẽ hiển thị div này.
    </div>| Ví dụ (true && true = true) nên hiển thị
    </div>
```

Form – Form Validation (Form)

- ❖ Kiểm độ dài ký tự với max-length & min-length

```
<div class="col-sm-10">
  <input type="email" minlength='7' maxlength='15' #email='ngModel' required ngModel class="form-control"
  !-- Kiểm tra rỗng -->
  <div *ngIf="email.errors && (email.dirty || email.touched)" class="alert alert-danger">
    !-- Kiểm tra rỗng -->
    <div *ngIf="email.errors.required">
      Email không được rỗng !
    </div>
    !-- Kiểm tra độ dài -->
    <div *ngIf = "(email.errors.minLength || email.errors.maxLength)">
      Độ dài từ 7 - 15 ký tự
    </div>
  </div>
</div>
```

Nếu độ dài nằm ngoài
khoảng
 $7 < X < 15$ thì sẽ hiển thị
div này



Form – Form Validation (Form)

❖ Kiểm tra định dạng chuỗi nhập vào với pattern

```
<input type="email" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,3}$" minlength='7' maxlength='15' #email=''\>
<! -- Kiểm tra rỗng -->
<div *ngIf="email.errors && (email.dirty || email.touched)" class="alert alert-danger">
<! -- Kiểm tra rỗng -->
    <div *ngIf="email.errors.required">
        Email không được rỗng !
    </div>
<! -- Kiểm tra độ dài -->
    <div *ngIf = "(email.errors.minLength || email.errors.maxLength)">
        Độ dài từ 7 - 15 ký tự
    </div>
<! -- Kiểm tra định dạng chuỗi nhập liệu -->
    <div *ngIf = "email.errors.pattern">
        Email không đúng định dạng
    </div>
</div>
</div>
```



Kiểm tra định dạng với email thông qua thuộc tính pattern nếu đúng sẽ hiển thị div báo lỗi

Tham khảo thêm các pattern:

https://www.w3schools.com/tags/att_input_pattern.asp

Form – Form Validation (Form)

- ❖ Gợi ý kiểm tra password trùng nhau

```
<div class="form-group">
  <label class="control-label col-sm-2" for="pass">pass:</label>
  <div class="col-sm-10">
    <input type="password" class="form-control" id="pass" name="pass" #pass placeholder="Enter pass" ngModel>
  </div>
</div>
<div class="form-group">
  <label class="control-label col-sm-2" for="repass">re pass:</label>
  <div class="col-sm-10">
    <input type="password" #repass [ngClass]="{'ng-invalid':(pass.value!=repass.value), 'ng-valid':!(pass.value!=repass.value)}">
    <div *ngIf = "repass.errors || pass.value!=repass.value" class="alert alert-danger">
      <div *ngIf = "pass.value!=repass.value">
        password không trùng
      </div>
    </div>
  </div>
</div>
```



Ta dùng 2 biến đại diện cho 2 input **#pass #repass**
+Kiểm tra **2 giá trị** nó có khác (**!=**) nhau không nếu khác nhau thì hiển thị div báo lỗi.
+Đồng thời tự xét tay thuộc tính **ng-invalid hay ng-valid** cho input thông qua **property binding** thuộc tính directive **ngClass**.

Form – Form Validation (Form)

❖ Trạng thái đầu vào của thẻ Form

`pristine` : Không có trường nào được thay đổi.

`dirty` : Có một trường hoặc nhiều trường đã được thay đổi.

`invalid` : Nội dung Form là không hợp lệ.

`valid` : Nội dung Form là hợp lệ.

`submitted` : Form đã được submit.

Tương tự như thẻ input thì các thuộc tính của Form cũng có giá trị trả lại là true hoặc false.

```
<div class="form-group">
  <div class="col-sm-offset-2 col-sm-10">
    <button type="submit" *ngIf='!registerForm.submitted' [disabled]='!registerForm.valid' class="btn btn-defau
  </div>
</div>
```

 Form chưa submit hiển thị, submit
rồi thì ẩn đi

 Form chưa nhập hợp lệ thì không cho
click, nhập hợp lệ thì enabled lên

Form – Form Validation (Form)

❖ Ngoài ra angular còn cung cấp ta thêm 1 số class CSS để validate.

AngularJS tự động thêm các class CSS cho Form và các thẻ Input tùy thuộc vào trạng thái của chúng. Đồng thời nó sẽ tự xóa đi class đó khi dữ liệu hợp lệ.

➤ Đối với Input

`ng-untouched` : Trường này chưa được chạm vào.

`ng-touched` : Trường này đã được chạm vào.

`ng-pristine` : Trường này chưa được thay đổi.

`ng-dirty` : Trường này đã được thay đổi.

`ng-valid` : Trường có nội dung hợp lệ.

`ng-invalid` : Trường có nội dung không hợp lệ.

`ng-valid-key` : key tương ứng với mỗi validation. Ví dụ: `ng-valid-required`, trường đã có nội dung

`ng-invalid-key` : Ví dụ: `ng-invalid-required`, trường không có nội dung

➤ Đối với form

`ng-pristine` : Không có trường nào được thay đổi.

`ng-dirty` : Có một trường hoặc nhiều trường đã được thay đổi.

`ng-valid` : Nội dung Form là hợp lệ.

`ng-invalid` : Nội dung Form là không hợp lệ.

`ng-valid-key` : key tương ứng với mỗi validation. Ví dụ: `ng-valid-required`, có một hoặc nhiều hơn một trường đã có nội dung

`ng-invalid-key` : Ví dụ: `ng-invalid-required`, chưa trường nào có nội dung

Form – Form Validation (Form)

💡 Ví dụ:

```
styles: [`  
    input.ng-invalid {  
        border-left:5px solid red;  
    }  
    input.ng-valid {  
        border-left:5px solid green;  
    }  
`]
```

➤ Kết quả:

Form - validation

Email: khai@g
Độ dài từ 7 - 15 ký tự
Email không đúng định dạng

Name: cybersoft

Name: cybersoft

Submit

- Ta thêm vào khi input hợp lệ tất cả thì thẻ input có border left là màu xanh.
- Ngược lại nếu chưa hợp lệ thì thẻ input có border left là màu đỏ

Form – Form Validation (Form)

➤ Cách set giá trị cho form từ Modal

Bước 1: Dùng viewchild dom tới Form đăng ký ngoài html thông qua local reference “registerForm” đại diện cho thẻ form

Bước 2: Vì giá trị form trả về là 1 object, nên khi ta set giá trị cho form cũng phải đưa vào 1 object với key là name của các ô input, value là giá trị value của input

HTML

```
<form [ngForm]="registerForm" class="form-horizontal" (ngSubmit) = "Submit(registerForm.value)">
<div class="form-group">
  <label for="inputEmail3" class="col-sm-2 control-label">Email</label>
  <div class="col-sm-10">
    <input type="email" class="form-control" id="inputEmail3" name="email" placeholder="Email" ngModel>
  </div>
</div>
<div class="form-group">
  <label for="Name" class="col-sm-2 control-label">Name</label>
  <div class="col-sm-10">
    <input type="text" class="form-control" id="Name" name="name" placeholder="Name" ngModel>
  </div>
</div>
<div class="form-group">
  <label for="school" class="col-sm-2 control-label">School</label>
  <div class="col-sm-10">
    <select id="school" class="form-control" name="school" ngModel>
      <option *ngFor='let school of schools' [value]="'school.id">'{{school.name}} </option>
    </select>
  </div>
</div>
<div class="form-group">
  <div class="col-sm-offset-2 col-sm-10">
    <button type="submit" class="btn btn-default">Submit</button>
  </div>
</div>
</form>
```

TypeScript

```
)>
export class DangkyComponent implements OnInit {
  @ViewChild('registerForm') dangkyF:NgForm

  SetValueUser(){
    this.dangkyF.setValue({
      TaiKhoan:'hieu',
      MatKhau:'hieu',
      confirmPass:'hieu',
      HoTen:'hieu',
      Email:'hieu',
      SoDT:'hieu',
      MaLoaiNguoiDung:'KhachHang',
      gender:'famale'
    })
  }
}
```

Bài tập tổng hợp

Please Sign Up It's free and always will be.

First Name Last Name

Display Name

Email Address

Password Confirm Password

Register

❖ Tạo form như hình:

- Tạo 1 component (trang) UserLogin thiết kế giao diện như hình
- Kiểm tra rỗng
- Kiểm tra tên từ 6 – 50 ký tự
- Kiểm tra định dạng email
- Kiểm tra password và confirm password
- Sau khi hợp lệ bấm register thì nó sẽ in ra table bên dưới. Với các cột tương ứng.
- Thêm 1 textbox để tìm kiếm theo họ tên hoặc email.

Bài tập tổng hợp

Please Sign Up It's free and always will be.

The form consists of six input fields arranged vertically. At the top is a horizontal progress bar with segments in green, yellow, red, purple, and blue. Below the progress bar are two rows of two input fields each. The first row contains 'First Name' and 'Last Name'. The second row contains 'Display Name' and 'Email Address'. The third row contains 'Password' and 'Confirm Password'. At the bottom is a large blue button labeled 'Register'.

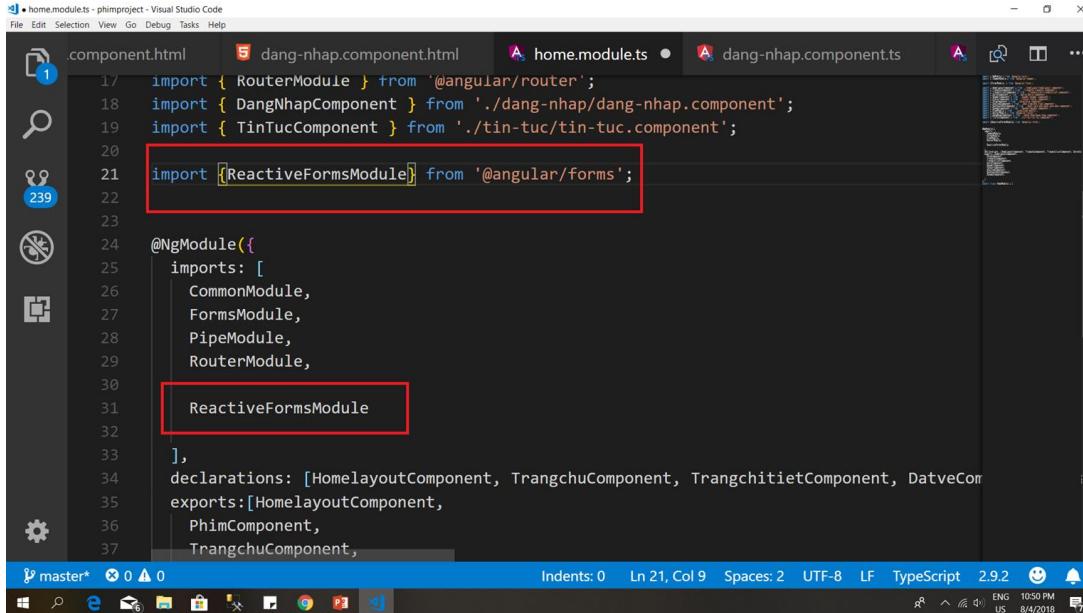
- Tạo thêm 1 component chứa Form đăng nhập với 2 input là email và password.
- Tạo 1 component hiển thị 2 link đăng ký, đăng nhập.
- Khi click vào link đăng ký thì form đăng ký hiển thị lên. Khi click vào link đăng nhập thì form đăng nhập hiển thị form đăng ký ẩn và ngược lại.
- Xây dựng chức năng cho form đăng nhập kiểm tra đăng nhập khi người dùng đăng nhập thành công xuất ra Xin chào email và ẩn form đó đi.

ReactiveForm

Ngoài FormsModule, Angular hỗ trợ thêm các module để handle form, trong đó có ReactiveFormsModule.

Cách sử dụng ReactiveFormsModule :

- Bước 1: import ReactiveFormsModule vào module nào cần sử dụng



```
home.module.ts - phimproject - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

.component.html      dang-nhap.component.html    home.module.ts •   dang-nhap.component.ts
1/  import { RouterModule } from '@angular/router';
18 import { DangNhapComponent } from './dang-nhap/dang-nhap.component';
19 import { TinTucComponent } from './tin-tuc/tin-tuc.component';
20
21 import { ReactiveFormsModule } from '@angular/forms';
22
23
24 @NgModule({
25   imports: [
26     CommonModule,
27     FormsModule,
28     PipeModule,
29     RouterModule,
30
31     ReactiveFormsModule
32
33   ],
34   declarations: [HomelayoutComponent, TrangchuComponent, TrangchitietComponent, DatveCom
35   exports:[HomelayoutComponent,
36     PhimComponent,
37     TrangchuComponent,
```

Indents: 0 Ln 21, Col 9 Spaces: 2 UTF-8 LF TypeScript 2.9.2 ENG 10:50 PM US 8/4/2018

ReactiveForm

- Bước 2: Xây dựng form Đăng Nhập ở giao diện HTML
- Bước 3: Xây dựng ReactiveForm Đăng Nhập ở Modal (typescript)
- Bước 4: Kết nối 2 form, dữ liệu người dùng nhập vào HTML được truyền vào ReactiveForm ở Modal
- Bước 5: Viết phương thức DangNhap, console.log ra value của form

```
<div class="container">
  <div class="row">
    <div class="col-5 mx-auto">
      <form [FormGroup]="formDangNhap" (ngSubmit)="DangNhap()">
        <h4 class="display-4">Đăng Nhập</h4>
        <div class="form-group">
          <label for="">Tài Khoản</label>
          <input type="text" class="form-control" formControlName="TaiKhoan" />
        </div>
        <div class="form-group">
          <label for="">Mật Khẩu</label>
          <input type="text" class="form-control" formControlName="MatKhau" />
        </div>
        <div class="form-group text-center">
          <button type="submit" class="btn btn-success">Đăng nhập</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

```
dang-nhap.component.html
import { AuthService } from '../../../../../services/auth.service';
import { FormGroup, FormControl } from '@angular/forms';

@Component({
  selector: 'app-dang-nhap',
  templateUrl: './dang-nhap.component.html',
  styleUrls: ['./dang-nhap.component.css']
})
export class DangNhapComponent implements OnInit {
  public formDangNhap: FormGroup;

  ngOnInit() {
    this.formDangNhap = new FormGroup({
      'TaiKhoan': new FormControl(null),
      'MatKhau': new FormControl(null)
    })
  }
}
```

import từ angular forms

Khởi tạo ReactiveForm, Kiểu dữ liệu là FormGroup

Các trường giá trị của reactiveForm, có thể hiểu các FormControl này giống như các ô input, có name là TaiKhoan và value ban đầu là null

ReactiveForm

➤ Kết quả:

Đăng Nhập

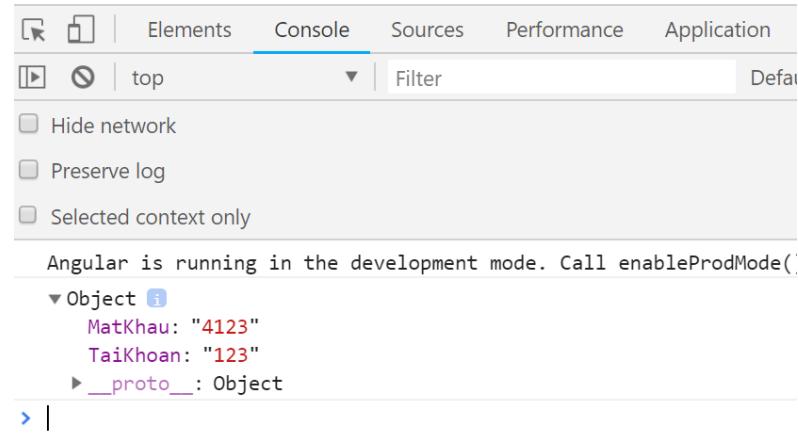
Tài Khoản

123

Mật Khẩu

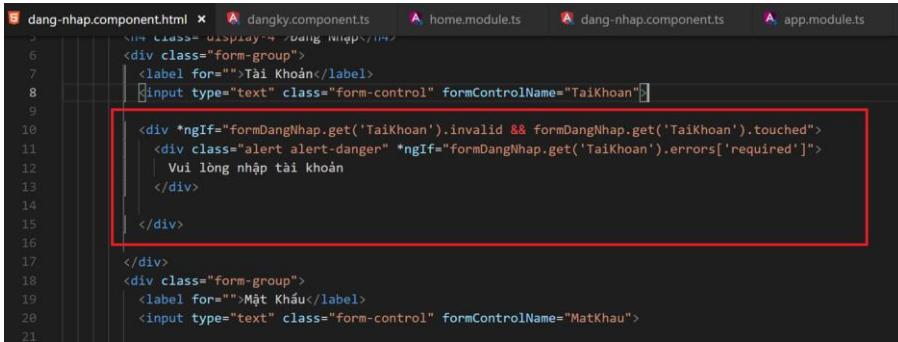
4123

Đăng nhập



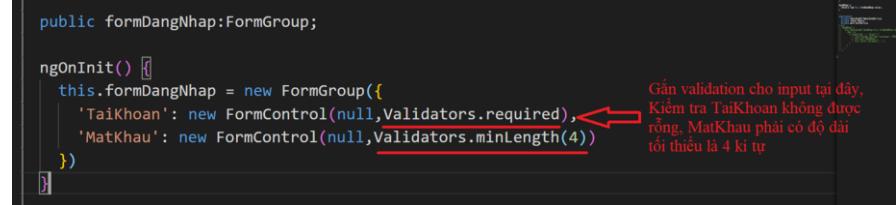
Validation với ReactiveForm

Ở HTML, kiểm tra không hợp lệ thì hiện alert



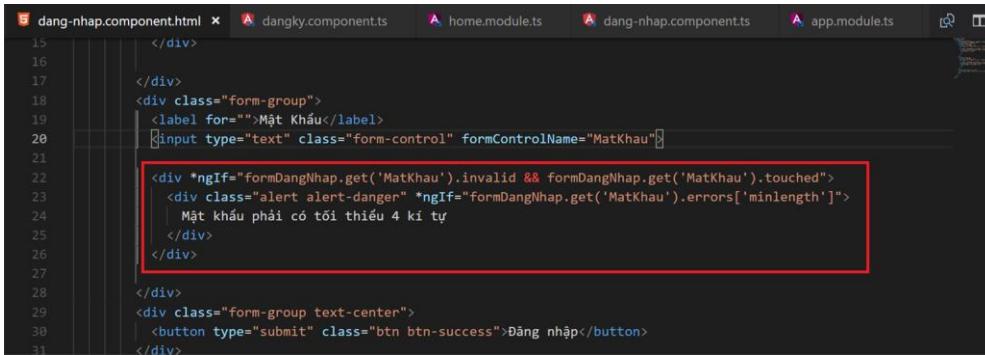
```
dang-nhap.component.html
5   class="form-group">
6     <div class="form-group">
7       <label for="">Tài Khoản</label>
8       <input type="text" class="form-control" formControlName="TaiKhoan">
9
10      <div *ngIf="formDangNhap.get('TaiKhoan').invalid && formDangNhap.get('TaiKhoan').touched">
11        <div class="alert alert-danger" *ngIf="formDangNhap.get('TaiKhoan').errors['required']">
12          Vui lòng nhập tài khoản
13        </div>
14
15      </div>
16    </div>
17    <div class="form-group">
18      <label for="">Mật Khẩu</label>
19      <input type="text" class="form-control" formControlName="MatKhau">
20
21      <div *ngIf="formDangNhap.get('MatKhau').invalid && formDangNhap.get('MatKhau').touched">
22        <div class="alert alert-danger" *ngIf="formDangNhap.get('MatKhau').errors['minlength']">
23          Mật khẩu phải có tối thiểu 4 kí tự
24        </div>
25
26      </div>
27
28    </div>
29    <div class="form-group text-center">
30      <button type="submit" class="btn btn-success">Đăng nhập</button>
31    </div>
```

Gắn validators trực tiếp vào formControl



```
public formDangNhap: FormGroup;
ngOnInit() {
  this.formDangNhap = new FormGroup({
    'TaiKhoan': new FormControl(null,Validators.required),
    'MatKhau': new FormControl(null,Validators.minLength(4))
  })
}
```

Gắn validation cho input tại đây,
Kiểm tra TaiKhoan không được
rỗng. MatKhau phải có độ dài
tối thiểu là 4 kí tự



```
dang-nhap.component.html
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

Kết hợp nhiều validators

Ở HTML, kiểm tra không
hợp lệ thì hiện alert



```
15   </div>
16
17   </div>
18   <div class="form-group">
19     <label for="">Mật Khẩu</label>
20     <input type="text" class="form-control" formControlName="MatKhau">
21
22     <div *ngIf="formDangNhap.get('MatKhau').invalid && formDangNhap.get('MatKhau').touched">
23       <div class="alert alert-danger" *ngIf="formDangNhap.get('MatKhau').errors['minlength']">
24         Mật khẩu phải có tối thiểu 4 kí tự
25       </div>
26       <div class="alert alert-danger" *ngIf="formDangNhap.get('MatKhau').errors['required']">
27         Vui lòng nhập mật khẩu
28       </div>
29     </div>
30   </div>
31 
```

Gắn validators trực
tiếp vào formControl

```
export class DangNhapComponent implements OnInit {
  public formDangNhap: FormGroup;
  ngOnInit() {
    this.formDangNhap = new FormGroup({
      'TaiKhoan': new FormControl(null, Validators.required),
      'MatKhau': new FormControl(null,[Validators.required,Validators.minLength(4)])
    })
  }
  DangNhap(){
    console.log(this.formDangNhap.value);
  }
}
```

Bài tập tổng hợp

A screenshot of a movie database application. At the top, there are three buttons: a blue plus sign for adding new movies, an orange edit/pencil icon, and a red delete/icon. Below this is a table header with columns: #, No., Hình ảnh (Image), Tên Phim (Movie Name), Sản xuất (Production), Ngày công chiếu (Release Date), and Trailer. The first row has a checkbox, ID 0, and a placeholder image. The second row has a checkbox, ID 1, and shows the movie "Minions" with production "Kyle Bald", release date "2015-07-10T00:00:00", and a trailer link. The third row has a checkbox, ID 2, and shows the movie "ted2" with production "Troy Kotsur", release date "2012-06-26T00:00:00", and a trailer link. A watermark for "Activate Windows" is visible at the bottom right.

#	No.	Hình ảnh	Tên Phim	Sản xuất	Ngày công chiếu	Trailer
□	0		jkakjhajkhjkdas	fgsdfgsdfg	2018-06-12T00:00:00	dsfgsdfgsdfg
□	1		Minions	Kyle Bald	2015-07-10T00:00:00	https://www.youtube.com/watch?v=Wfql_DoHRKc
□	2		ted2	Troy Kotsur	2012-06-26T00:00:00	Activate Windows Go to Settings to activate Windows.

A screenshot of an "Add Movie" form. It consists of several input fields arranged in a grid-like structure. From top-left to bottom-right, the fields are: "ID" (text input), "Title" (text input), "Description" (text input), "Rating" (text input), "Director" (text input), "Release Date" (text input), "Trailer" (text input), and "Image Url" (text input). At the bottom center is a large orange button labeled "ADD MOVIE".

ID	Title
Description	
Rating	Director
Release Date	Trailer
Image Url	
ADD MOVIE	

Tạo form add phim như hình:

- Kiểm tra rỗng
- Kiểm tra description trên 20 kí tự
- Kiểm tra Title phải là chữ
- Kiểm tra Rating phải là số
- Sau khi hợp lệ bấm Add Movie , tiến hành thêm phim vào mảng DanhSachPhim
- Tạo bảng động hiển thị danh sách phim
- Tên phim hiển thị ra bảng nhỏ hơn 10 ký tự, lớn hơn 10 kí tự thì cắt bớt và thêm dấu "..."
- Ngày chiếu hiển thị theo định dạng NgàyTháng-Năm Giờ : phút : giây
- ID hiển hiện ra bảng phải là chữ in hoa.
- Thêm button Cập nhật, sửa lại phim.

Thực Hành: HTTP - POST

Navbar Trang chủ Danh sách phim Tin tức Đăng ký

Đăng ký tài khoản

Tài khoản

Mật khẩu

Email

Họ tên

Số điện thoại

Đăng ký

Thực Hành: HTTP - POST

- Ứng dụng kiến thức phần Form (Two-way binding) để lấy các values người dùng nhập vào.
- Sau khi lấy được tất cả các values. Viết service http post để register user.

Thực Hành: HTTP - POST

dang-ky.component.html

TS dang-ky.component.ts

src > app > home > dang-ky > TS dang-ky.component.ts > ...

```
9  export class DangKyComponent implements OnInit {
10  @ViewChild("formRegister", { static: false }) formRegister;
11
12  constructor(private dataService: DataService) {}
13
14  ngOnInit() {}
15
16  register(value) {
17    const user = {
18      taiKhoan: value.TaiKhoan,
19      matKhau: value.MatKhau,
20      email: value.Email,
21      soDt: value.SoDT,
22      maNhom: "GP01",
23      maLoaiNguoiDung: "KhachHang",
24      hoTen: value.HoTen
25    };
26    this.dataService.dangKy(user).subscribe(
27      data => {
28        alert("Đăng ký thành công");
29      },
30      err => {}
31    );
32 }
```

TS data.service.ts

dang-ky.component.html

dang-ky.component.ts

src > app > _core > services > TS data.service.ts > DataService > dangKy

```
65  dangKy(user): Observable<any> {
66    const url = "http://movie0706.cybersoft.edu.vn/api/QuanLyNguoiDung/DangKy";
67    return this.http.post(url, user).pipe(
68      tap(() => {}),
69      catchError(err => {
70        return this.handleError(err);
71      })
72    );
73 }
```

Template - Routing - Admin

- Như đã tạo ra phần Template Admin ở phần trước (Có Dashboard và các trang quản trị cho website: Thêm, xóa, sửa người dùng...).
- Để vào được các trang quản trị. Bắt buộc chúng ta phải có trang LoginAdmin để login vào.
- Tạo trang đăng nhập để vào được Template Admin (Làm tương tự như chức năng đăng ký - gọi http post).

The screenshot shows a web browser window with the URL `localhost:4200/admin` in the address bar. The page title is "Dang nhap". The form has two input fields: "Tài khoản" and "Mật khẩu", both with placeholder text. Below the inputs is a green button labeled "Đăng nhập". The browser's toolbar and various tabs are visible at the top.

Dang nhap

Tài khoản

Mật khẩu

Đăng nhập

Template - Routing - Admin

admin.component.html TS admin.component.ts TS data.service.ts

```
src > app > admin > TS admin.component.ts > AdminComponent > login
10  export class AdminComponent implements OnInit {
11    constructor(private dataService: DataService, private router: Router) {}
12
13    ngOnInit() {}
14
15    login(value) {
16      const user = {
17        taiKhoan: value.TaiKhoan,
18        matKhau: value.MatKhau
19      };
20
21      this.dataService.dangNhap(user).subscribe(data => {
22        if (data.maLoaiNguoiDung === "QuanTri") {
23          alert("Login thanh cong");
24          localStorage.setItem("UserAdmin", JSON.stringify(data));
25          this.router.navigate(["/admin/dashboard"]);
26        } else {
27          alert("K co quyen");
28        }
29      });
30    }
31 }
```

n.component.html

TS admin.component.ts

TS data.service.ts

src > _core > services > TS data.service.ts > DataService > dangNhap

```
dangNhap(user): Observable<any> {
  const url =
    "http://movie0706.cybersoft.edu.vn/api/QuanLyNguoiDung/DangNhap";
  return this.http.post(url, user).pipe(
    tap(() => {}),
    catchError(err => {
      | return this.handleError(err);
    })
  );
}
```

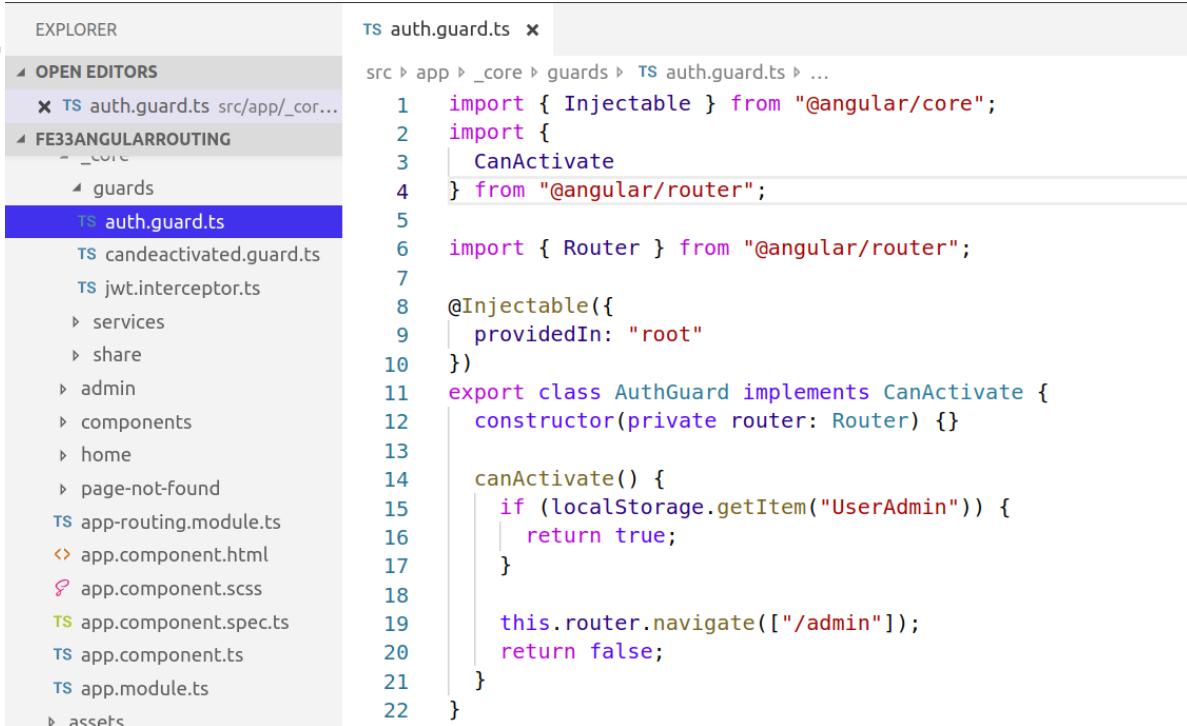
Guard

- Chúng ta đã tạo thành công các trang quản trị thành công.
- Nhưng các bạn có nghĩ nếu như đường link trang quản trị bị người khác biết và cố tình truy cập vào thì sao? Chắc chắn hệ thống website sẽ bị lủng chỗ này.
- Vậy Guard sẽ giúp bảo vệ những đường link nhạy cảm này. Có nghĩa: nếu chưa login thành công thì cho dù biết link cũng không truy cập vào được.

Guard - thực hiện

- Tạo folder guards trong thư mục _core.
- Mở Terminal tại folder guards cài đặt theo cú pháp:

ng g guard <tên file>



The screenshot shows a code editor with two panes. The left pane is the Explorer view, displaying the project structure. The right pane is the code editor showing the content of `auth.guard.ts`.

EXPLORER

- OPEN EDITORS
- TS auth.guard.ts src/app/_cor...
- FE33ANGULARROUTING
 - _core
 - guards
 - TS auth.guard.ts**
 - TS candeactivated.guard.ts
 - TS jwt.interceptor.ts
 - services
 - share
 - admin
 - components
 - home
 - page-not-found
 - TS app-routing.module.ts
 - app.component.html
 - app.component.scss
 - TS app.component.spec.ts
 - TS app.component.ts
 - TS app.module.ts
 - assets

TS auth.guard.ts

```
src > app > _core > guards > TS auth.guard.ts > ...
1 import { Injectable } from "@angular/core";
2 import {
3   CanActivate
4 } from "@angular/router";
5
6 import { Router } from "@angular/router";
7
8 @Injectable({
9   providedIn: "root"
10 })
11 export class AuthGuard implements CanActivate {
12   constructor(private router: Router) {}
13
14   canActivate() {
15     if (localStorage.getItem("UserAdmin")) {
16       return true;
17     }
18
19     this.router.navigate(["/admin"]);
20     return false;
21   }
22 }
```

Guard - thực hiện

- Sau khi đã tạo guard xong, lúc này sẽ sử dụng chúng bằng cách bảo vệ đường link nào thì mở routing liên quan đó ra config.
- Tại object routing thêm thuộc tính canActivate: [AuthGuard].

```
TS admin-routing.module.ts ✘  
src › app › admin › TS admin-routing.module.ts › routes  
1 import { NgModule } from "@angular/core";  
2 import { Routes, RouterModule } from "@angular/router";  
3 import { AdminComponent } from "./admin.component";  
4 import { AuthGuard } from "../_core/guards/auth.guard";  
5  
6 const routes: Routes = [  
7   {  
8     path: "",  
9     component: AdminComponent  
10    },  
11    {  
12      path: "dashboard",  
13      loadChildren: "./dashboard/dashboard.module#DashboardModule",  
14      canActivate: [AuthGuard]  
15    }  
16];
```

AuthInterceptor

- Tại Template Admin, implement chức năng thêm người dùng.
- Bạn có nghĩ nếu chức năng thêm người dùng này. Một người nào đó lấy được đường link API và gửi request thêm người dùng thì sao? => Lủng rồi :(
- Cách làm để khắc phục là khi gửi request lên server để thêm người dùng. BE sẽ yêu cầu FE gửi kèm theo token để BE xác thực người dùng.

==> Interceptor.

AuthInterceptor - thực hiện

- Thiết kế UI để thêm người dùng.
- Chức năng làm tương tự trang đăng ký (khác nhau ở chỗ để thêm được user thì cần phải gửi thêm token lên server).

Thêm tài khoản

Tài khoản

Mật khẩu

Email

Họ tên

Số điện thoại

Add User

AuthInterceptor - thực hiện

- Tại folder guards tạo thêm file jwt.interceptor

The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar:
 - OPEN EDITORS: TS jwt.interceptor.ts
 - FE33ANGULARROUTING
 - e2e
 - node_modules
 - src
 - app
 - _core
 - guards
 - TS auth.guard.ts
 - TS candeactivated.guard.ts
 - TS jwt.interceptor.ts
 - services
 - share
 - admin
 - dashboard
 - TS dashboard-routing.mod...
 - TS dashboard.component....
 - TS dashboard.component.s...
 - TS dashboard.component.s...
 - TS dashboard.component.ts
 - TS dashboard.module.ts
 - OUTLINE
- TS jwt.interceptor.ts** code editor:

```
4  import { HttpRequest, HttpHandler } from '@angular/common/http';
5  @Injectable()
6  export class AuthInterceptor implements HttpInterceptor {
7      constructor() {}
8
9      intercept(req: HttpRequest<any>, next: HttpHandler) {
10         // Get the auth token from the service.
11         let authToken = '';
12
13         if (localStorage.getItem("UserAdmin")) {
14             authToken = JSON.parse(localStorage.getItem("UserAdmin")).accessToken;
15         }
16
17         // Clone the request and replace the original headers with
18         // cloned headers, updated with the authorization.
19         const authReq = req.clone({
20             headers: req.headers.set("Authorization", `Bearer ${authToken}`)
21         });
22
23         // send cloned request with header to the next handler.
24         return next.handle(authReq);
25     }
26
27 }
28 }
```

AuthInterceptor - thực hiện

- Sau đó config vào app.module.ts

```
ts jwt.interceptor.ts      ts app.module.ts ×
src > app > ts app.module.ts > ...
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';
7 import { AuthInterceptor } from './core/guards/jwt.interceptor';
8 import { PageNotFoundComponent } from './page-not-found/page-not-found.compon
9
10 @NgModule({
11   declarations: [AppComponent, PageNotFoundComponent],
12   imports: [BrowserModule, AppRoutingModule, HttpClientModule],
13   providers: [
14     { provide: HTTP_INTERCEPTORS, useClass: AuthInterceptor, multi: true }
15   ],
16   bootstrap: [AppComponent]
17 })
18 export class AppModule {}
```

Guard - CanDeactivate

- ❑ Quay lại chức năng cho người dùng đăng ký user. Trong một form đăng ký có rất nhiều ô input để nhập liệu. Trong trường hợp ngẫu nhiên nào đó, người dùng chuyển trang khác hoặc reload lại trang thì sao? Mọi dữ liệu nhập trước đó trong input sẽ mất hết.
- ❑ CanDeactivate giúp người dùng xác nhận lại trước khi chuyển trang. Sẽ khắc phục được mất value trong input khi không muốn.

Guard - CanDeactivate

- ☐ Tạo file candeactivated.guard.ts trong folder guards.

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows the project structure under "OPEN EDITORS".
 - Note-Angular.txt
 - candeactivated.guard.ts (highlighted)
 - FE3ANGULARROUTING
 - app
 - _core
 - guards
 - auth.guard.ts
 - candeactivated.guard.ts (highlighted)
 - jwt.interceptor.ts
 - services
 - share
 - admin
 - dashboard
 - dashboard-routing.mod...
 - dashboard.component....
- Note-Angular.txt**: A text file in the background.
- candeactivated.guard.ts**: An Angular guard file.

```
src > app > _core > guards > TS candeactivated.guard.ts > ...
1 import { Injectable } from '@angular/core';
2 import {
3   CanDeactivate
4 } from '@angular/router';
5 import { DangKyComponent } from "src/app/home/dang-ky/dang-ky.component";
6
7 @Injectable({
8   providedIn: "root"
9 })
10 export class CandeactivatedGuard implements CanDeactivate<DangKyComponent> {
11   canDeactivate(component) {
12     return (
13       component.canDeactivate() ||
14       window.confirm("Bạn có muốn rời khỏi trang không?")
15     );
16   }
17 }
```

Guard - CanDeactivate

- Quay lại dang-ky component để config.

The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar:
 - OPEN EDITORS**: 1 UNSAVED
 - TS candeactivated.guard.ts
 - TS dang-ky.component.ts src...
 - Untitled-1
 - FE33ANGULARROUTING**:
 - e2e
 - node_modules
 - src
 - app
 - _core
 - admin
 - components
 - home
 - dang-ky
 - TS dang-ky-routing.module...
 - dang-ky.component.html
 - dang-ky.component.scss
 - TS dang-ky.component.spe...
- TS candeactivated.guard.ts**:

```
src/app/home/dang-ky/TS dang-ky.component.ts DangKyComponent canDeactivate
1 import { Component, OnInit, ViewChild, HostListener } from '@angular/core';
2 import { DataService } from "src/app/_core/services/data.service";
3
4 @Component({
5   selector: "app-dang-ky",
6   templateUrl: "./dang-ky.component.html",
7   styleUrls: ['./dang-ky.component.scss']
8 })
9 export class DangKyComponent implements OnInit {
10   @ViewChild("formRegister", { static: false }) formRegister;
11
12   constructor(private dataService: DataService) {}
13
14   ngOnInit() {}
15
16   @HostListener("window:beforeunload", ["$event"])
17   canDeactivate() {
18     return !this.formRegister.dirty;
19   }
20 }
```
- TS dang-ky.component.ts**:

```
DangKyComponent canDeactivate
src/app/home/dang-ky/TS dang-ky.component.ts DangKyComponent canDeactivate
1 import { Component, OnInit, ViewChild, HostListener } from '@angular/core';
2 import { DataService } from "src/app/_core/services/data.service";
3
4 @Component({
5   selector: "app-dang-ky",
6   templateUrl: "./dang-ky.component.html",
7   styleUrls: ['./dang-ky.component.scss']
8 })
9 export class DangKyComponent implements OnInit {
10   @ViewChild("formRegister", { static: false }) formRegister;
11
12   constructor(private dataService: DataService) {}
13
14   ngOnInit() {}
15
16   @HostListener("window:beforeunload", ["$event"])
17   canDeactivate() {
18     return !this.formRegister.dirty;
19   }
20 }
```
- Untitled-1**: (empty)

Pipes

❖ Ví dụ:

```
TS pipe.component.ts ✘

 1  import { Component, OnInit } from '@angular/core';
 2  @Component({
 3    selector: 'app-pipe',
 4    template:
 5      <div> {{info | uppercase}} </div>
 6      <div> {{info | lowercase}} </div>
 7      <div> {{percent | percent}} </div>
 8      <div> {{percent | percent:'2.3-5'}} </div>
 9      <div> {{percent | number:'3.1-6' }} </div>
10      <div> {{objectJson | json }} </div>
11      <div> {{array | slice:1:5}} </div>
12    ,
13    styles: ['./pipe.component.css']
14  })
15  export class PipeComponent implements OnInit {
16    private info:string = "Học viện đào tạo cybersoft";
17    private percent:number = 32321.962;
18    private objectJson:Object = {hoten:'Minh',lop:'frontend01',diem:[10,2,6]};
19    private array:Array<string> = ['pt1','pt2','pt3','pt4','pt5'];
20    constructor() {}
21    ngOnInit() {
22    }
```

<https://angular.io/docs/ts/latest/api/#/?apiFilter=pipe&type=pipe> (rất nhiều pipe gồm định dạng format animation ...) tham khảo thêm.

Thực hành: tạo pipe rút gọn chuỗi

- B1: Trong folder **app**, tạo ra module **PipeModule** quản lý các pipe do ta tự định nghĩa
- B2: Dùng cli tạo ra pipe với cú pháp: **ng g pipe shortcut**
- B3: Định dạng nội dung của ShortcutPipe

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

@Pipe({
  name: 'shortcut'
})
export class ShortcutPipe implements PipeTransform{
  transform(value, limit){
    return value.substr(0,limit)+"..."
  }
}
```

Thực hành: tạo pipe rút gọn chuỗi

B4: Ở Pipe, tiến hành import và declaration và exports ShorcutPipe để PipeModule quản lý tương tự như 1 component (Angular Cli đã hỗ trợ)

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ShortcutPipe } from './shortcut.pipe';

@NgModule({
  imports: [
    CommonModule
  ],
  declarations: [ShortcutPipe],
  exports:[ShortcutPipe]
})
export class PipeModule { }
```

B5: Import **PipeModule** vào module nào sử dụng **ShorcutPipe**

```
import { DangnhapComponent } from './dangnhap/dangnhap.component';
import { DangkyComponent } from './dangky/dangky.component';

import {PipeModule} from '../pipe/pipe.module';

@NgModule({
  imports: [
    CommonModule,RouterModule, PipeModule,FormsModule
  ],
  declarations: [TrangchuComponent, TrangchitietComponent, TrangdatgheComponent]
})
export class HomeModule { }
```

Lifecycle

Một component có vòng đời bắt đầu khi Angular khởi tạo component và hiển thị khung nhìn component cùng với các khung nhìn con của nó. Vòng đời tiếp tục với phát hiện thay đổi, khi Angular kiểm tra xem khi nào các thuộc tính ràng buộc dữ liệu thay đổi và cập nhật cả chế độ xem và component khi cần. Vòng đời kết thúc khi Angular phá hủy cá thể component và xóa mẫu được kết xuất của nó khỏi DOM. Các lệnh có vòng đời tương tự, như Angular tạo, cập nhật và hủy các thể hiện trong quá trình thực thi.

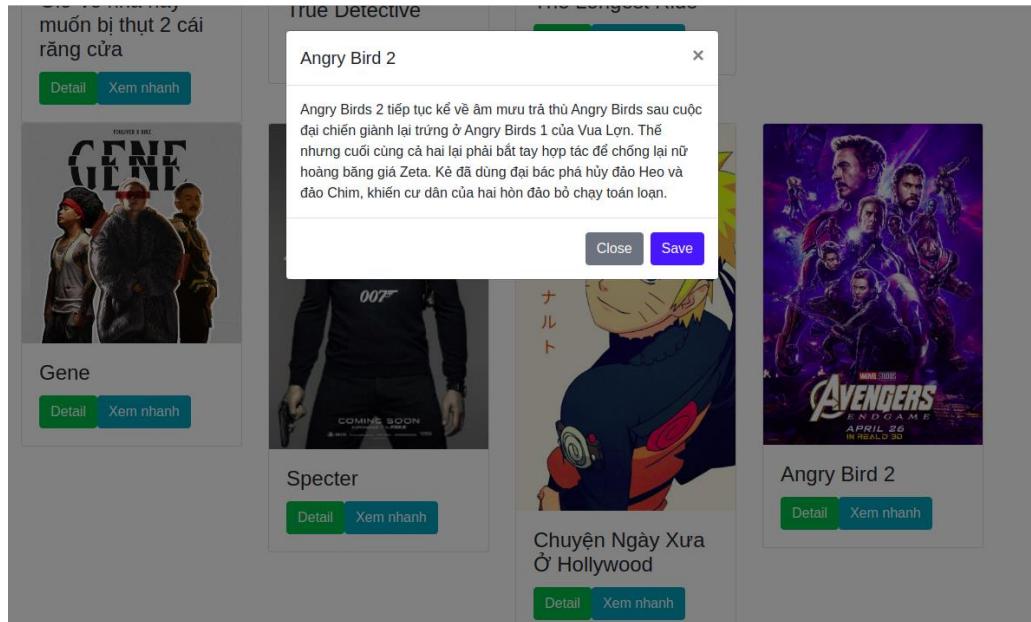
Tham khảo: <https://angular.io/guide/lifecycle-hooks>

ts trang-chu.component.ts x

```
src > app > home > trang-chu > TS trang-chu.component.ts > TrangChuComponent
6   styleUrls: ['./trang-chu.component.scss']
7 }
8 export class TrangChuComponent implements OnInit {
9
10  constructor() {
11    console.log("constructor");
12  }
13
14  ngOnChanges() {
15    console.log("ngOnChanges");
16  }
17
18  ngOnInit() {
19    console.log("ngOnInit");
20  }
21
22  ngAfterViewInit() {
23    console.log("ngAfterViewInit");
24  }
25
26  ngOnDestroy() {
27    console.log("ngOnDestroy");
28  }
29 }
```

BehaviorSubject

- ❑ Là một service tập hợp data, giúp share data để các component có thể truy cập đến mà không cần truyền qua nhiều cấp độ component
- ❑ Tại trang danh sách phim, tạo thêm button "Xem nhanh". Khi click sẽ show popup hiển thị nội dung thông tin nhanh của bộ phim cần xem mà không cần vào trang chi tiết phim



BehaviorSubject

- Tạo một file share-data để thiết kế service BehaviorSubject
- Tạo biến detailMovie từ đối tượng BehaviorSubject, giá trị khởi tạo là object.
- Tạo shareDetailMovie là một Observable để các component có thể truy xuất get data .
- Tạo phương thức sharingDataDetailMovie để component thực hiện action đẩy data vào

```
src > app > _core > share > TS share-data.service.ts > ...
1 import { Injectable } from '@angular/core';
2 import { BehaviorSubject } from 'rxjs';
3
4 @Injectable({
5   providedIn: "root"
6 })
7 export class ShareDataService {
8   private detailMovie = new BehaviorSubject({} as object);
9   shareDetailMovie = this.detailMovie.asObservable();
10
11 constructor() {}
12
13 sharingDataDetailMovie(movie) {
14   | this.detailMovie.next(movie);
15 }
16 }
```

BehaviorSubject

- ☐ Tại component movie khi action click vào button "Xem nhanh", gọi đến phương thức sharingDataDetailMovie để push thông tin chi tiết phim lên service.



The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar:
 - OPEN EDITORS**: Note-Angular.txt, share-data.service.ts, movie.component.ts (selected)
 - FE33ANGULARROUTING**: components, modal, movie (selected), movie.component.html, movie.component.scss, movie.component.spec.ts, movie.component.ts
- FILES** tab bar: Note-Angular.txt, share-data.service.ts, movie.component.ts
- Code Editor**:

```
src > app > components > movie > TS movie.component.ts > ...
1 import { Component, OnInit, Input } from "@angular/core";
2 import { ShareDataService } from "src/app/_core/share/share-data.service";
3
4 @Component({
5   selector: "app-movie",
6   templateUrl: "./movie.component.html",
7   styleUrls: ["./movie.component.scss"]
8 })
9 export class MovieComponent implements OnInit {
10   @Input() ...
11
12   constructor(private data: ShareDataService) {}
13
14   ngOnInit() {}
15
16   xemNhanh() {
17     | this.data.sharingDataDetailMovie(this.movie);
18   }
19 }
```

BehaviorSubject

- ☐ Khi trên service đã được push thông tin phim, lúc này ngay tại component modal connect đến service truy xuất đến shareDetailMovie để get data hiển thị thông tin.

The screenshot shows the VS Code interface with the Explorer sidebar on the left and the Editor on the right. The Explorer sidebar shows the project structure under 'OPEN EDITORS' and 'FE33ANGULARROUTING'. The 'modal.component.ts' file is selected in the list. The Editor window displays the following TypeScript code:

```
TS modal.component.ts x
src > app > components > modal > TS modal.component.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { ShareDataService } from '../../../../../_core/share/share-data.service';
3
4 @Component({
5   selector: "app-modal",
6   templateUrl: "./modal.component.html",
7   styleUrls: ["./modal.component.scss"]
8 })
9 export class ModalComponent implements OnInit {
10   movie: any;
11
12   constructor(private data: ShareDataService) {}
13
14   ngOnInit() {
15     this.data.shareDetailMovie.subscribe(data => {
16       |   this.movie = data;
17     });
18   }
19 }
```

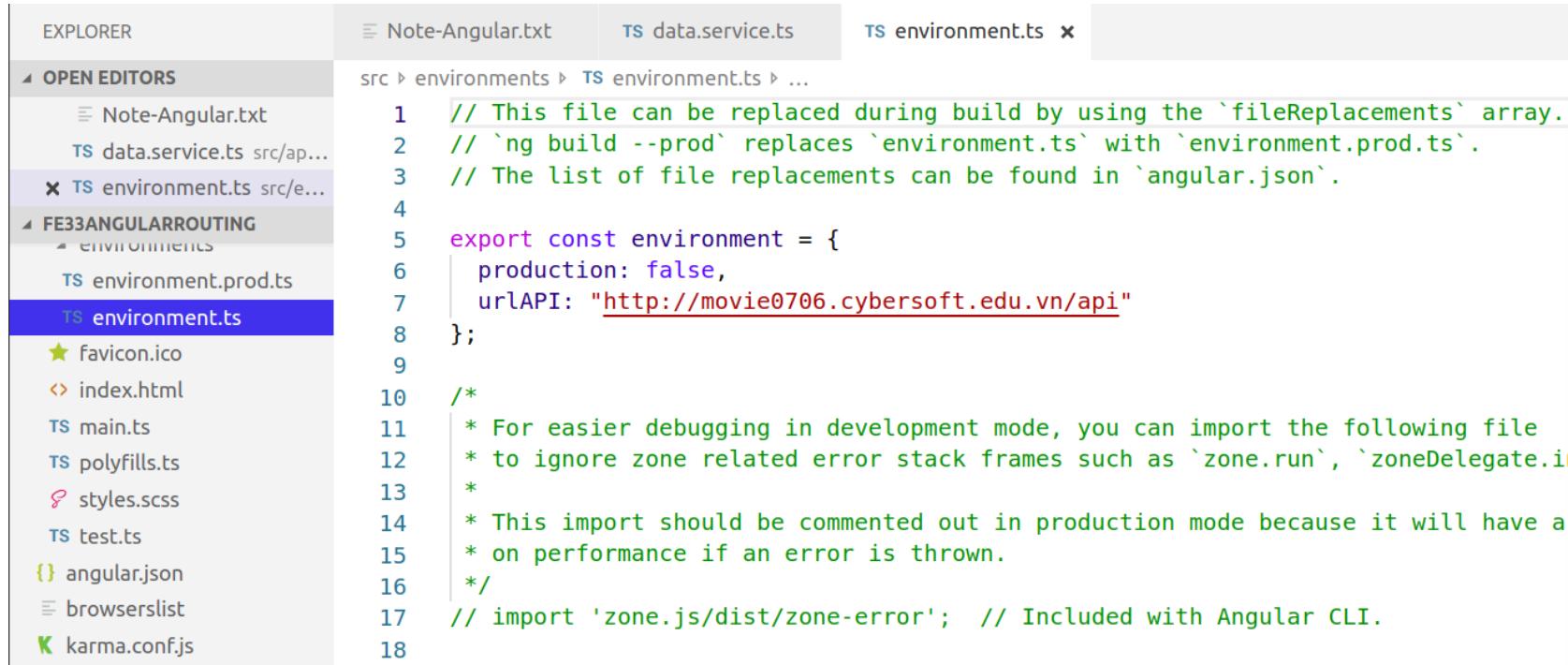
The code defines a modal component that uses a `ShareDataService` to get movie data via its `shareDetailMovie` BehaviorSubject. The component then sets this data to its own `movie` property.

Refactor Service

- Trong phần data.service gọi api, chúng ta thấy là cứ mỗi chức năng đều phải viết một hàm. Nếu về sau dự án càng lớn chức năng càng nhiều thì số lượng hàm sẽ tăng dần đến file sẽ bị phình to ra. Nhưng nhìn lại các hàm đó đều chỉ sử dụng các methods HTTP.
- Refactor code chúng ta sẽ định nghĩa ra hàm để tái sử dụng chung cho các chức năng liên quan đến các methods HTTP (GET, POST, PUT, ...).

Refactor Service

- ☐ Ở file environment.ts khai báo hằng số urlAPI là domain API. Sau này nếu domain thay đổi chúng ta chỉ cần đổi duy nhất một chỗ này.



The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows the project structure with files like Note-Angular.txt, data.service.ts, environment.ts (selected), environment.prod.ts, main.ts, polyfills.ts, styles.scss, test.ts, angular.json, browserslist, and karma.conf.js.
- Tabs**: Note-Angular.txt, data.service.ts, environment.ts (active).
- environment.ts Content**:

```
1 // This file can be replaced during build by using the `fileReplacements` array.
2 // `ng build --prod` replaces `environment.ts` with `environment.prod.ts`.
3 // The list of file replacements can be found in `angular.json`.
4
5 export const environment = {
6   production: false,
7   urlAPI: "http://movie0706.cybersoft.edu.vn/api"
8 };
9
10 /*
11  * For easier debugging in development mode, you can import the following file
12  * to ignore zone related error stack frames such as `zone.run`, `zoneDelegate.i
13  *
14  * This import should be commented out in production mode because it will have a
15  * on performance if an error is thrown.
16  */
17 // import 'zone.js/dist/zone-error'; // Included with Angular CLI.
18
```

Refactor Service

- Ở file data.service viết viết hàm get, post (đại diện cho chức năng liên quan đến method GET, POST) tái sử dụng cho tất cả các chức năng.

EXPLORER Note-Angular.txt TS data.service.ts TS environment.ts

OPEN EDITORS
Note-Angular.txt
TS data.service.ts src/ap...
TS environment.ts src/e...

FE33ANGULARROUTING
e2e
node_modules
src
app
_core
guards
services
TS data.service.ts
share
admin
components
home
page-not-found
TS app-routing.module.ts
app.component.html

```
src > app > _core > services > TS data.service.ts > DataService
1 import { Injectable } from '@angular/core';
2 import { Observable, throwError } from "rxjs";
3 import { HttpClient, HttpHeaders } from "@angular/common/http";
4 import { tap, catchError } from "rxjs/operators";
5 import { environment } from "../../../../../environments/environment";
6
7 let urlAPI = "";
8
9 const httpOptions = {
10   headers: new HttpHeaders({
11     | "Content-Type": "application/json"
12   })
13 };
14
15 @Injectable({
16   providedIn: "root"
17 })
18
19 export class DataService {
20   constructor(private http: HttpClient)
21   | urlAPI = environment.urlAPI;
22 }
```

EXPLORER Note-Angular.txt TS data.service.ts TS environment.ts

OPEN EDITORS
Note-Angular.txt
TS data.service.ts src/ap...
TS environment.ts src/e...

FE33ANGULARROUTING
_core
guards
services
TS data.service.ts
share
admin
components
home
page-not-found
TS app-routing.module.ts
app.component.html
app.component.scss

```
src > app > _core > services > TS data.service.ts > DataService
24
25   get(uri): Observable<any>{
26     return this.http.get(` ${urlAPI}/ ${uri}`).pipe(
27       tap(() => {}),
28       catchError(err => {
29         | return this.handleError(err);
30       })
31     )
32
33   post(uri, data): Observable<any>{
34     return this.http.post(` ${urlAPI}/ ${uri}`, data, httpOptions).pipe(
35       tap(() => {}),
36       catchError(err => {
37         | return this.handleError(err);
38       })
39     )
40 }
```

Refactor Service

- Quay lại DanhSachPhim component, không sử dụng service getListMovie() nữa mà sử dụng service get() truyền tham số phù hợp đã định nghĩa.
- Tương tự Refactor cho các chức năng khác.

```
EXPLORER Note-Angular.txt TS data.service.ts TS danh-sach-phim.component.ts x TS environment.ts  
OPEN EDITORS Note-Angular.txt TS data.service.ts src/app/home/danh-sach-phim/danh-sach-phim.component.ts ...  
TS danh-sach-phim.com... TS environment.ts src/e...  
FE33ANGULARROUTING TS danh-sach-phim-rou... TS danh-sach-phim.co... TS danh-sach-phim.co... TS danh-sach-phim.co... TS danh-sach-phim.co... TS danh-sach-phim.co... TS danh-sach-phim.mo...  
danh-sach-phim.co... tin-tuc trang-chu  
15  
16  ngOnInit() {  
17 |   this.layDanhSachPhim();  
18 }  
19  
20 layDanhSachPhim() {  
21 |   const uri = "QuanLyPhim/LayDanhSachPhim?maNhom=GP01";  
22 |   this.subListMovie = this.dataService.get(uri).subscribe(data => {  
23 |     console.log(data);  
24 |     this.listMovie = data;  
25 |   });  
26 }  
27  
28 ngOnDestroy(){  
29 |   this.subListMovie.unsubscribe();  
30 }  
31 --
```

Sharing Modules

- Trong Angular, một Component không thể **declarations** ở hai modules khác nhau.
- Vì thế nếu tái sử dụng Component ở nhiều modules khác nhau thì không thể.
- Sharing Modules sẽ giúp Component có thể sử dụng được ở nhiều modules khác.
- Ví dụ Modal Component được tái sử dụng rất nhiều ở những modules.
- Tạo share modules, khai báo Modal Component bên dưới

```
EXPLORER Note-Angular.txt share.module.ts
OPEN EDITORS
  Note-Angular.txt
  share.module.ts src/a...
FE33ANGULARROUTING
  guards
  services
  share
    share-data.service.ts
  share.module.ts
  admin
  components
  home

src > app > _core > share > share.module.ts > ShareModule
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { ModalComponent } from 'src/app/components/modal/modal.component';
4
5 @NgModule({
6   declarations: [ModalComponent],
7   exports: [ModalComponent],
8   imports: [
9     CommonModule
10   ]
11 })
12 export class ShareModule { }
```

Sharing Modules

- Muốn sử dụng Modal Component ở đâu thì phải xác định được mình đang đứng tại module nào, mở module đó lên và import share.module vào.
- Ở DanhSachPhim Component muốn sử dụng Modal Component, ta phải mở danh-sach-phim.module import share.module vào.

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows the project structure with files like "Note-Angular.txt", "share.module.ts", and "danh-sach-phim.module.ts".
- OPEN EDITORS**: Shows the currently open file: "danh-sach-phim.module.ts".
- FE33ANGULARROUTING**: Shows the routing configuration for the "danh-sach-phim" module.
- Editor Content (danh-sach-phim.module.ts)**: Displays the TypeScript code for the "DanhSachPhimModule".

```
src > app > home > danh-sach-phim > TS danh-sach-phim.module.ts > DANH SACH PHIM MODULE
4   import { DanhSachPhimRoutingModule } from './danh-sach-phim-routing.module';
5   import { DanhSachPhimComponent } from './danh-sach-phim.component';
6   import { MovieComponent } from 'src/app/components/movie/movie.component';
7   import { SharedModule } from 'src/app/_core/share/share.module';
8
9
10 @NgModule({
11   declarations: [DanhSachPhimComponent, MovieComponent],
12   imports: [CommonModule, DanhSachPhimRoutingModule, SharedModule]
13 })
14 export class DanhSachPhimModule {}
```

Alias

- ❑ Như chúng ta thấy, nếu đứng tại Modal component, muốn sử dụng phương thức nào liên quan đến share-data.service, phải truy cập vào đúng đường dẫn thư mục.



The screenshot shows the VS Code interface with the Explorer sidebar on the left and the code editor on the right. The Explorer sidebar lists files under 'OPEN EDITORS' and 'FE33ANGULARROUTING'. The 'modal.component.ts' file is selected in the 'OPEN EDITORS' list. The code editor displays the 'modal.component.ts' file with the following content:

```
ts modal.component.ts ✘
src > app > components > modal > ts modal.component.ts > ModalComponent > movie
1 import { Component, OnInit } from '@angular/core';
2 import { ShareDataService } from '../../../../../_core/share/share-data.service';
3
4 @Component({
5   selector: "app-modal",
6   templateUrl: "./modal.component.html",
7   styleUrls: ["./modal.component.scss"]
8 })
9 export class ModalComponent implements OnInit {
10   movie: any;
11
12   constructor(private data: ShareDataService) {}
13
14   ngOnInit() {
15     this.data.shareDetailMovie.subscribe(data => {
16       |   this.movie = data;
17     });
18   }
19 }
```

The line 'import { ShareDataService } from '../../../../../_core/share/share-data.service';' is highlighted with a red rectangle, indicating it is the subject of discussion in the slide.

Alias

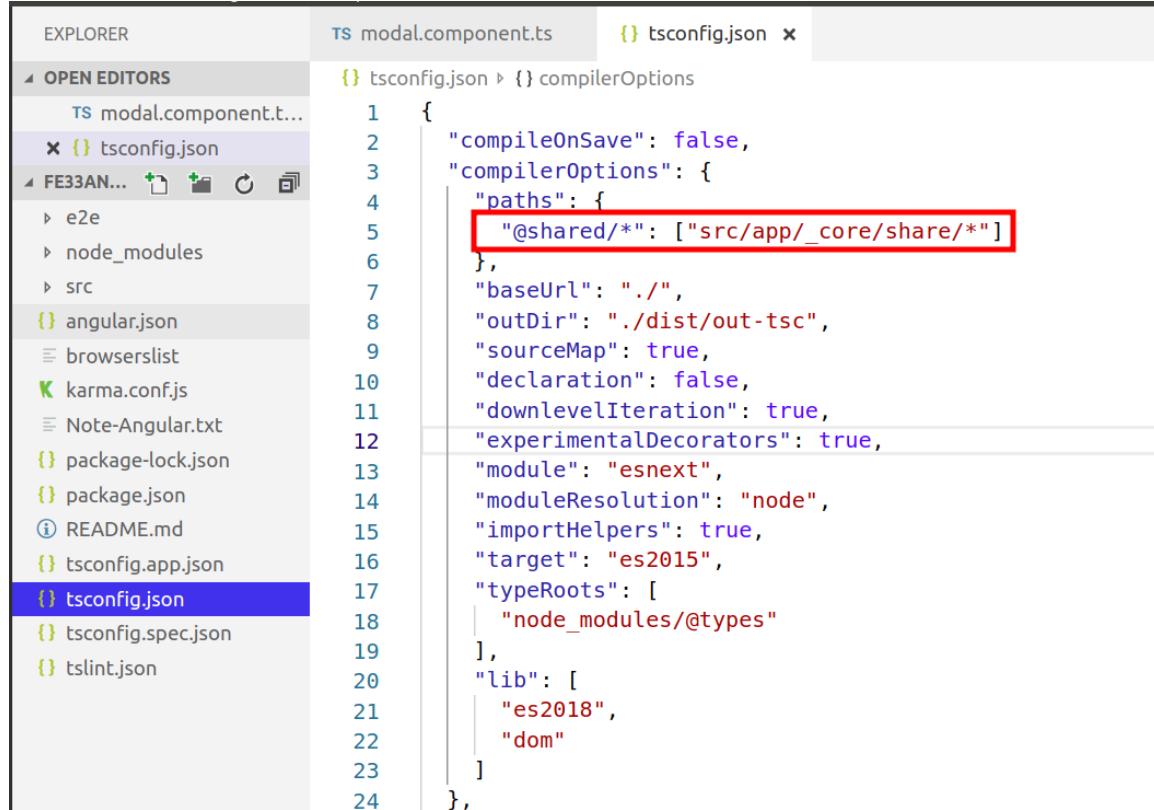
- ☐ Khi đã định nghĩa alias thì không cần phải đi ra, đi vào cho đúng đường dẫn, mà ta có thể import đường dẫn như bên dưới.
- ☐ Chúng ta đang import giống với Angular đang import những đường dẫn default.



```
src > app > components > modal > TS modal.component.ts > ...
1 import { Component, OnInit } from "@angular/core";
2 import { ShareDataService } from "@shared/share-data.service";
3
4 @Component({
5   selector: "app-modal",
6   templateUrl: "./modal.component.html",
7   styleUrls: ["./modal.component.scss"]
8 })
9 export class ModalComponent implements OnInit {
10   movie: any;
11
12   constructor(private data: ShareDataService) {}
13
14   ngOnInit() {
15     this.data.shareDetailMovie.subscribe(data => {
16       this.movie = data;
17     });
18   }
19 }
```

Alias

- ❑ Nhưng để hiểu được alias chúng ta cần phải config ở file tsconfig.json

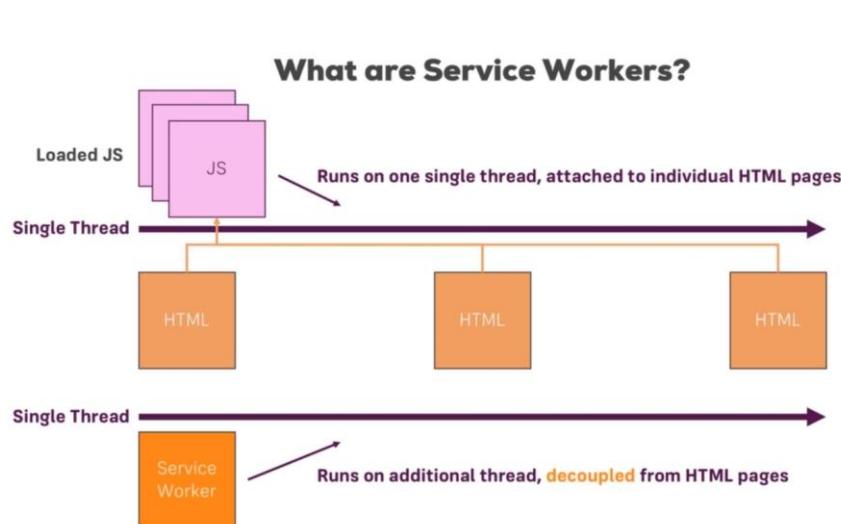


```
EXPLORER          modal.component.ts      tsconfig.json x
OPEN EDITORS
modal.component.ts
tsconfig.json
FE33AN... e2e node_modules src angular.json browserslist karma.conf.js Note-Angular.txt package-lock.json package.json README.md tsconfig.app.json tsconfig.json tsconfig.spec.json tslint.json

{} tsconfig.json > {} compilerOptions
  1  {
  2    "compileOnSave": false,
  3    "compilerOptions": {
  4      "paths": {
  5        "@shared/*": ["src/app/_core/share/*"]
  6      },
  7      "baseUrl": "./",
  8      "outDir": "./dist/out-tsc",
  9      "sourceMap": true,
 10     "declaration": false,
 11     "downlevelIteration": true,
 12     "experimentalDecorators": true,
 13     "module": "esnext",
 14     "moduleResolution": "node",
 15     "importHelpers": true,
 16     "target": "es2015",
 17     "typeRoots": [
 18       "node_modules/@types"
 19     ],
 20     "lib": [
 21       "es2018",
 22       "dom"
 23     ]
 24   },
```

Service worker là gì?

- Tại trình duyệt, javascript được xử lý ở một luồng đơn nhất
- Ngoài ra ta có thể chạy service worker ở trên một luồng khác.
- Luồng này chạy tách rời và có thể chạy ngầm ở background
- Service worker có thể lắng nghe các request được gửi tới API và cache reponse trả về từ API.
- Trang web vẫn có thể làm việc mà không có internet.



Sử dụng service worker?

- Bước 1: gõ lệnh ng add @angular/pwa để cài package cần thiết vào angular app
- Các file được sinh ra:

The screenshot shows the Visual Studio Code interface with two tabs open: 'app.module.ts' and 'ngsw-config.json'. The 'EXPLORER' sidebar on the left shows the project structure with files like 'src/app', 'node_modules', and 'src'. The 'OPEN EDITORS' tab shows the code for 'ngsw-config.json'. The code defines asset groups for 'app' and 'assets' with specific resources and install modes.

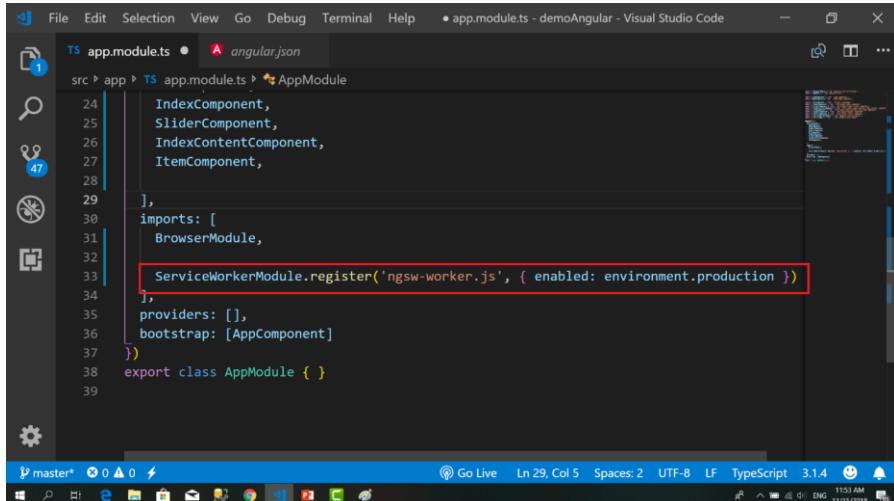
```
1  {
2   "index": "/index.html",
3   "assetGroups": [
4     {
5       "name": "app",
6       "installMode": "prefetch",
7       "resources": {
8         "files": [
9           "/favicon.ico",
10          "/index.html",
11          "/**.css",
12          "/**.js"
13        ]
14      }
15    },
16    {
17      "name": "assets",
18      "installMode": "lazy",
19      "updateMode": "prefetch",
20      "resources": {
21        "files": [
22          "/assets/**"
23        ]
24      }
25    }
26  }
27}
```

The screenshot shows the Visual Studio Code interface with two tabs open: 'app.module.ts' and 'manifest.json'. The 'EXPLORER' sidebar on the left shows the project structure with files like 'src/app', 'node_modules', and 'src'. The 'OPEN EDITORS' tab shows the code for 'manifest.json'. The manifest object contains details about the application, such as name, short name, theme color, background color, and start URL.

```
1  {
2   "name": "demoAngular",
3   "short_name": "demoAngular",
4   "theme_color": "#1976d2",
5   "background_color": "#fafafa",
6   "display": "standalone",
7   "scope": "/",
8   "start_url": "/",
9   "icons": [
10     {
11       "src": "assets/icons/icon-72x72.png",
12       "sizes": "72x72",
13       "type": "image/png"
14     },
15     {
16       "src": "assets/icons/icon-96x96.png",
17       "sizes": "96x96",
18       "type": "image/png"
19     }
20   ]
21 }
```

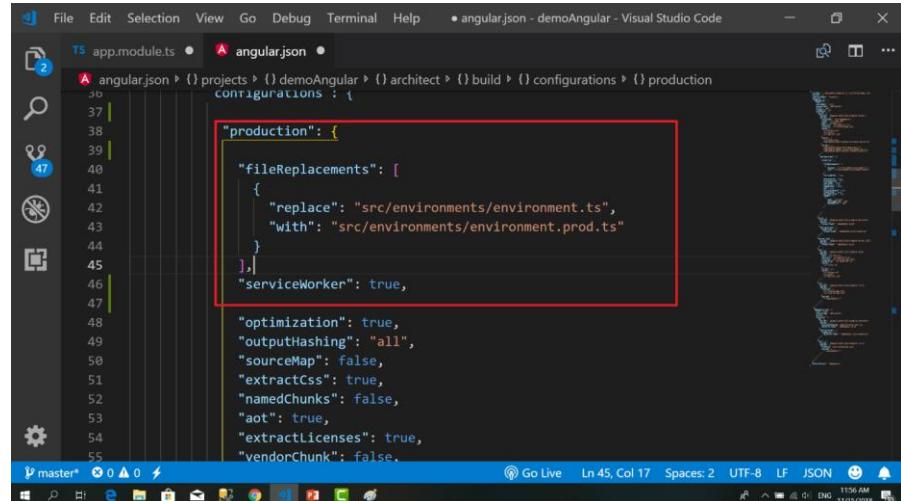
Sử dụng service worker?

Trong AppModule:



```
File Edit Selection View Go Debug Terminal Help • app.module.ts - demoAngular - Visual Studio Code
TS app.module.ts ● A angular.json
src > app > TS app.module.ts > AppModule
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
IndexComponent,
SliderComponent,
IndexContentComponent,
ItemComponent,
],
imports: [
BrowserModule,
ServiceWorkerModule.register('ngsw-worker.js', { enabled: environment.production })
],
providers: [],
bootstrap: [AppComponent]
)
export class AppModule { }
```

Trong angular.json: Service worker sẽ được add vào khi ta build ra thư mục dist



```
File Edit Selection View Go Debug Terminal Help • angular.json - demoAngular - Visual Studio Code
TS app.module.ts ● A angular.json ●
src > angular.json > {} projects > {} demoAngular > {} architect > {} build > {} configurations > {} production
configurations : t
"production": {
  "fileReplacements": [
    {
      "replace": "src/environments/environment.ts",
      "with": "src/environments/environment.prod.ts"
    }
  ],
  "serviceWorker": true,
  "optimization": true,
  "outputHashing": "all",
  "sourceMap": false,
  "extractCss": true,
  "namedChunks": false,
  "aot": true,
  "extractLicenses": true,
  "vendorChunk": false,
  "buildOptimizer": true
}
master* 0 ▲ 0 ⚡ Go Live Ln 29, Col 5 Spaces: 2 UTF-8 LF TypeScript 3.1.4 ⓘ 🔍
master* 0 ▲ 0 ⚡ Go Live Ln 45, Col 17 Spaces: 2 UTF-8 LF JSON ⓘ 🔍
```

Sử dụng service worker?

Bước 2: gõ lệnh ng build --prod để build project với mode production

The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure under 'DEMOANGULAR'. The 'dist' folder is selected and highlighted with a blue bar. Inside 'dist', there are several files and folders: 'demoAngular', 'assets', '3rdpartylicenses.txt', 'favicon.ico', 'index.html', 'main.bd72e4f643654ef5d274.js', 'manifest.json', 'ngsw-worker.js', 'ngsw.json', 'polyfills.c6871e56cb80756a5498.js', 'runtime.ec2944dd8b20ec099bf3.js', 'safety-worker.js', 'scripts.d273e48091cb14ac29de.js', 'styles.e27f57d38ba5b68b9655.css', and 'worker-basic.min.js'. A red rectangular box highlights the 'main.bd72e4f643654ef5d274.js' file. On the right, the Editor pane shows the 'angular.json' file with the 'production' configuration section expanded. The code in the editor is as follows:

```
  "production": {
    "fileReplacements": [
      {
        "replace": "src/assets/icon.png",
        "with": "src/assets/icon-production.png"
      }
    ],
    "serviceWorker": true,
    "optimization": true,
    "outputHashing": "all",
    "sourceMap": false,
    "extractCss": true,
    "namedChunks": true,
    "aot": true,
    "extractLicense": false,
    "vendorChunk": true,
    "buildOptimizer": true,
    "budgets": [
      {
        "label": "main",
        "maxSize": 1000000
      }
    ]
  }
```

Sử dụng service worker?

Bước 3: cài đặt npm install -g lite-server để chạy thư mục dist

Bước 4: chạy lite-server tại thư mục dist được build ra

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "DEMOANGULAR". A red box highlights the "dist" folder, which contains files like "index.html", "main.bd72e4f6436...", "manifest.json", "ngsw-worker.js", "ngsw.json", "polyfills.c6871e56c...", "runtime.ec2944dd...", "safety-worker.js", "scripts.d273e4809...", "styles.e27f57d38b...", and "worker-basic.min.js".
- Code Editor:** Displays the "angular.json" file. A yellow box highlights the "production" configuration section:

```
    "production": {  
        "fileReplacements": [  
            {  
                "replace": "src/environments/environment.ts",  
                "with": "src/environments/environment.prod.ts"  
            }  
        ],  
        "serviceWorker": true,  
        "optimization": true,  
        "outputHashing": "all",  
        "sourceMap": false,  
    }
```
- Terminal:** Shows the command "lite-server" being run in the terminal, indicated by a red box. The output shows the path "c:\CYBERSOFT\cybersoft\FE11\angular\demoAngular\dist\demoAngular>lite-server".
- Bottom Status Bar:** Provides information about the file ("master"), line and column ("Ln 45, Col 17"), spaces ("Spaces: 2"), encoding ("UTF-8"), line separator ("LF"), JSON support, and system status.

Sử dụng service worker?

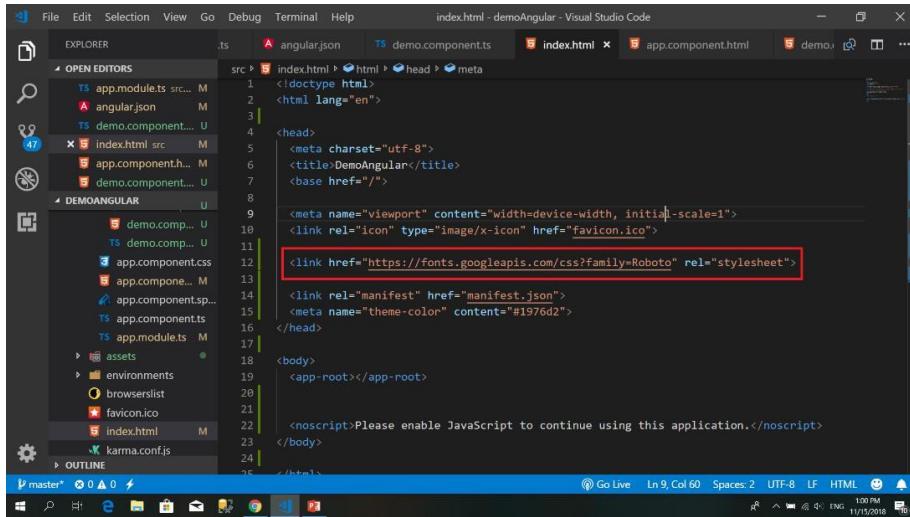
- Kết quả: file service worker được add vào, app vẫn chạy dù ngắt internet vì dữ liệu bây giờ được lấy từ service worker.
- Tuy nhiên, các đường dẫn và dữ liệu động lấy từ api không hoạt động

The screenshot shows the Chrome DevTools Network tab for a local host application. In the 'Service Workers' section, a service worker named 'ngsw-worker.js' is registered under the URL 'localhost'. The status indicates it is active and running. A red box highlights this registration entry.

The screenshot shows the Chrome DevTools Network tab with a waterfall chart. The chart displays various network requests, including 'ng-validate.js', 'ng-validate.js', 'manifest.json', 'favicon.ico', 'manifest.json', 'icon-144x144.png', 'ngsw-worker.js', and 'ngsw-worker.js'. A red box highlights the first two requests, which are both labeled '(from ServiceWorker)' in the initiator column. The waterfall chart also shows the time taken for each request and the total load time.

Sử dụng service worker?

Nếu sử dụng cdn ở index.html, chẳng hạn như sử dụng font từ google font, ta cần thêm một vài set up



File Edit Selection View Go Debug Terminal Help index.html - demoAngular - Visual Studio Code

EXPLORER .ts angular.json demo.component.ts index.html app.component.html demo...

OPEN EDITORS src index.html

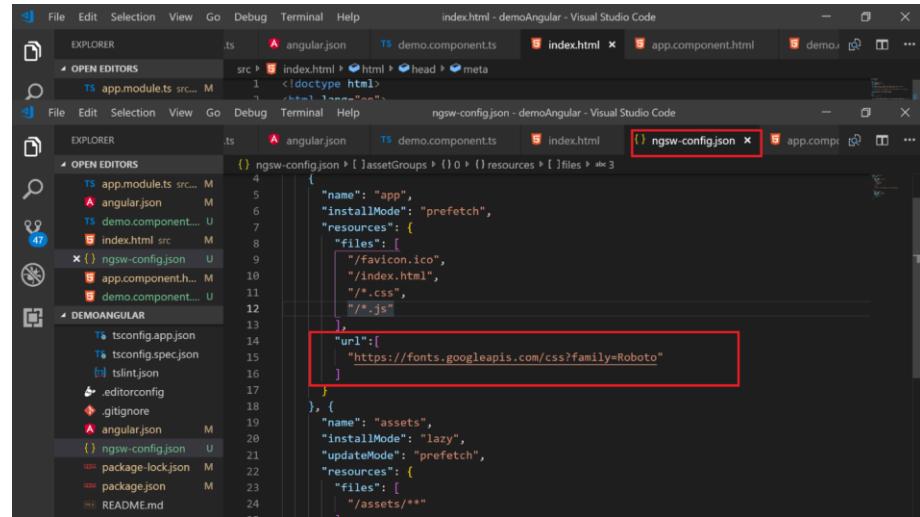
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>DemoAngular</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
    <link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet">
```

DEMOANGULAR

assets environments browserlist favicon.ico index.html karma.conf.js

OUTLINE

master* 0 0 0 0 Go Live Ln 9, Col 60 Spaces 2 UTF-8 LF HTML



File Edit Selection View Go Debug Terminal Help ngservice-worker-config.json - demoAngular - Visual Studio Code

EXPLORER .ts angular.json demo.component.ts index.html app.component.html demo...

OPEN EDITORS src index.html

```
{
  "name": "app",
  "installMode": "prefetch",
  "resources": {
    "files": [
      "/favicon.ico",
      "/index.html",
      "/*.css",
      "/*.js"
    ],
    "url": [
      "https://fonts.googleapis.com/css?family=Roboto"
    ]
  }
}, {
  "name": "assets",
  "installMode": "lazy",
  "updateMode": "prefetch",
  "resources": {
    "files": [
      "/assets/**"
    ]
  }
}
```

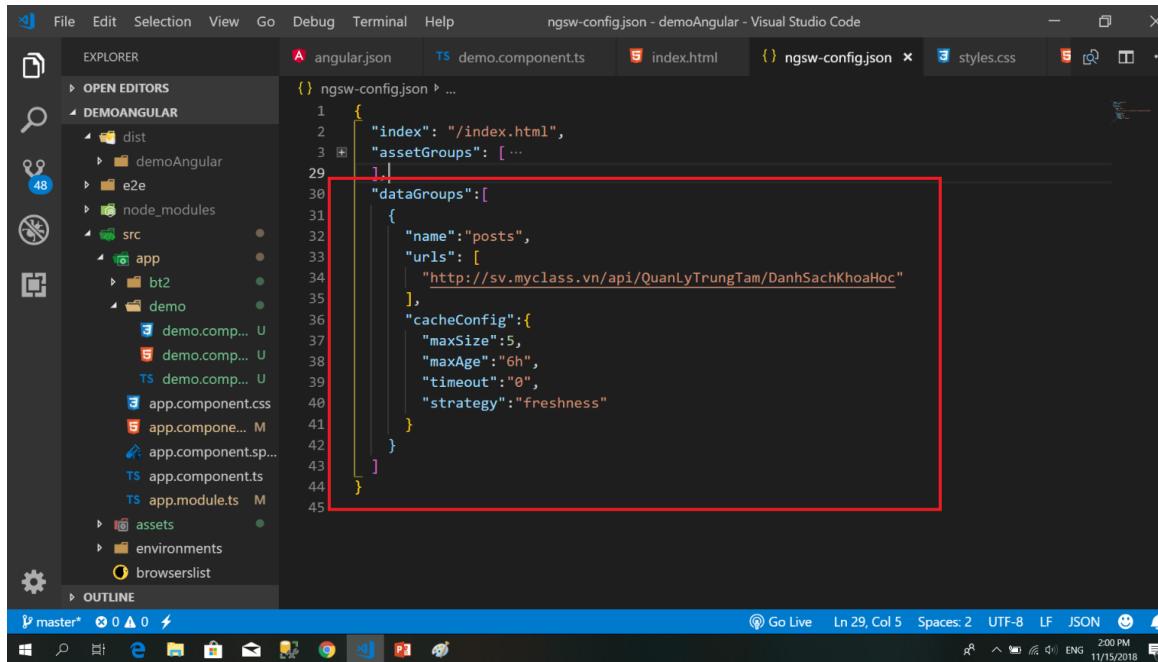
DEMOANGULAR

tsconfig.app.json tsconfig.spec.json tslint.json editorconfig gitignore angular.json ngservice-worker-config.json package-lock.json package.json README.md

Sử dụng service worker?

Load dữ liệu động với service worker:

Thêm một group vào file ngsw-config.json như sau



```
1  {
2    "index": "/index.html",
3    "assetGroups": [ ... ],
4    "dataGroups": [
5      {
6        "name": "posts",
7        "urls": [
8          "http://sv.myclass.vn/api/QuanLyTrungTam/DanhSachKhoaHoc"
9        ],
10       "cacheConfig": {
11         "maxSize": 5,
12         "maxAge": "6h",
13         "timeout": "0",
14         "strategy": "freshness"
15       }
16     }
17   }
```

Sử dụng service worker?

Trong đó:

- Name tự đặt
- urls: chứa các api sử dụng
- cacheConfig : điều chỉnh một số thiết lập khi lưu cache
 - maxSize: số lượng response tối đa cache được
 - maxAge: thời gian lưu trữ tối đa, sau đó sẽ tiến hành fetch lại dữ liệu mới
 - Timeout: sau một khoảng thời gian đợi phản hồi từ api, sẽ fetch dữ liệu từ catch
 - Strategy: gồm 2 giá trị (freshness | performance)
 - Freshness: ưu tiên chờ fetch dữ liệu từ api lên trước, quá thời gian set ở timeout mới lấy dữ liệu từ cache
 - Performance: ưu tiên lấy dữ liệu từ cache lên trước, fetch dữ liệu từ api và update sau

Sử dụng service worker?

Trong đó:

- Name tự đặt
- urls: chứa các api sử dụng
- cacheConfig : điều chỉnh một số thiết lập khi lưu cache
 - maxSize: số lượng response tối đa cache được
 - maxAge: thời gian lưu trữ tối đa, sau đó sẽ tiến hành fetch lại dữ liệu mới
 - Timeout: sau một khoảng thời gian đợi phản hồi từ api, sẽ fetch dữ liệu từ catch
 - Strategy: gồm 2 giá trị (freshness | performance)
 - Freshness: ưu tiên chờ fetch dữ liệu từ api lên trước, quá thời gian set ở timeout mới lấy dữ liệu từ cache
 - Performance: ưu tiên lấy dữ liệu từ cache lên trước, fetch dữ liệu từ api và update sau

**CHÚC CÁC BẠN HOÀN THÀNH ĐỒ ÁN
THẬT TỐT VÀ ĐẠT
ĐƯỢC MỤC TIÊU BAN ĐẦU HƯỚNG ĐẾN
KHI ĐẾN VỚI
CYBERSOFT ACADEMY**