



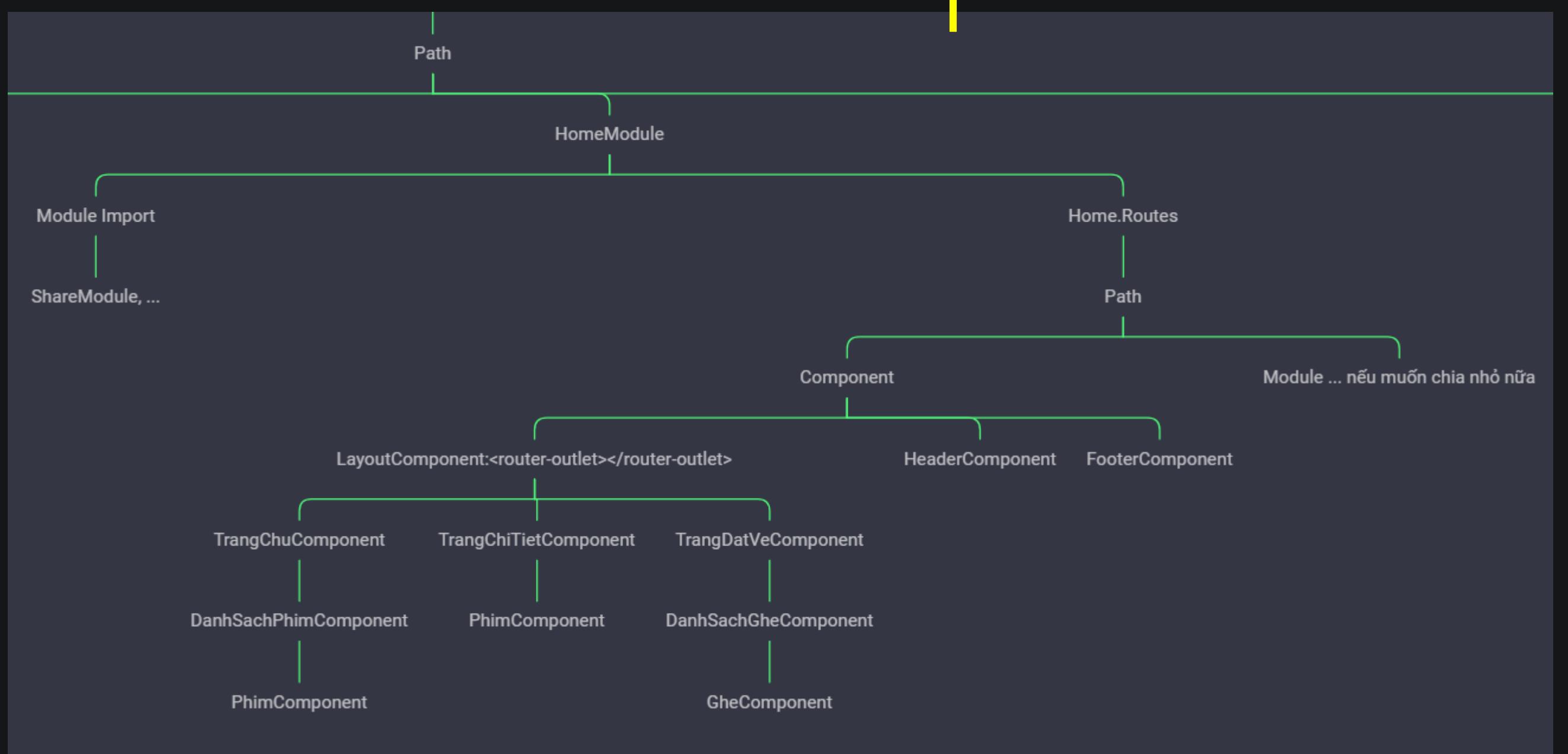
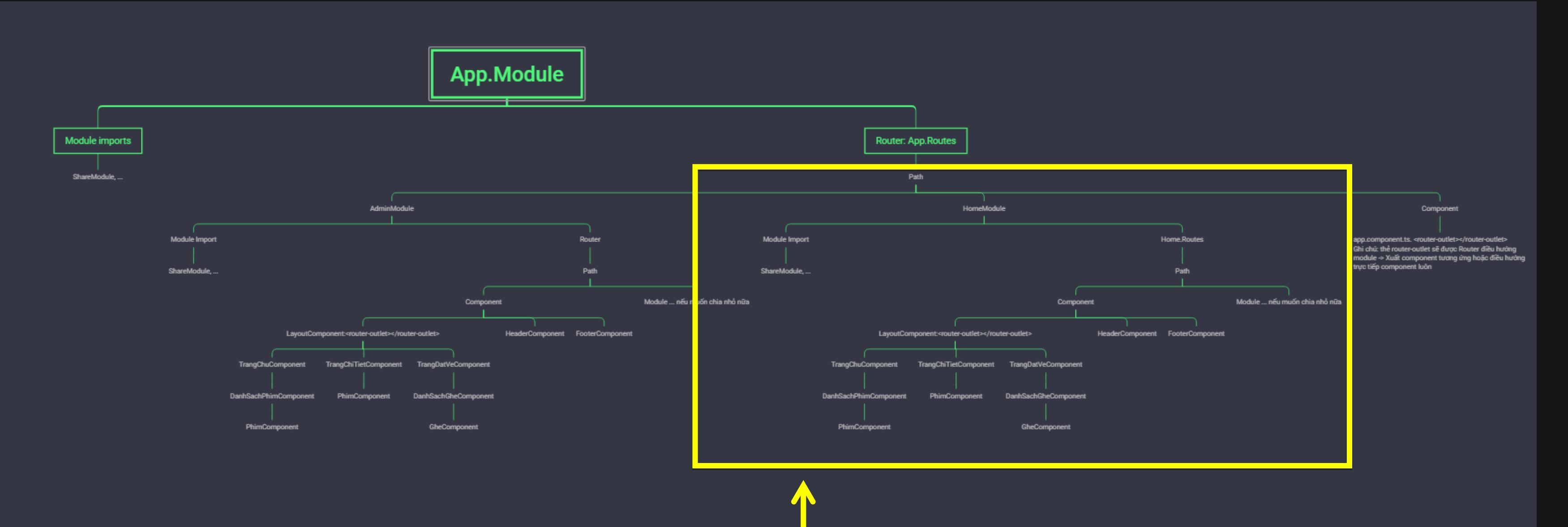
Angular 6

1

Introduction to Angular 6

Giới thiệu về angular 6

1. **Angular js 2(4-5-6)** hay gọi tắt là **angular 2(4-5-6)** là 1 framework của javascript được google phát triển để phát triển ứng dụng web.
2. **Angular 2(4-5-6)** là 1 phiên bản tiếp theo của angular js (angular 1) có thêm 1 số khái niệm mới. (Nếu biết angular 1 thì học **angular 2(4-5-6)** sẽ nhanh hơn tuy nhiên không biết cũng không sao chỉ cần nắm vững javascript là được).
3. **Angular 2(4-5-6)** hỗ trợ đa nền tảng web lẫn di động (dùng Ionic build ra các ứng dụng di động).
4. Viết chủ yếu bằng code **typescript** (gần với oop hơn dễ tiếp cận).
5. Hướng tiếp cận dựa trên **component, module**.



App component (Thuộc app module)

HomeLayout

COMPONENT

AdminLayout

COMPONENT



Để chuyển đổi qua lại giữa các trang (hay component) ta có khái niệm RouteConfig

❖ Component

HomeComponent

Header component

Navigation
component

<router-outlet>

Footer Component

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

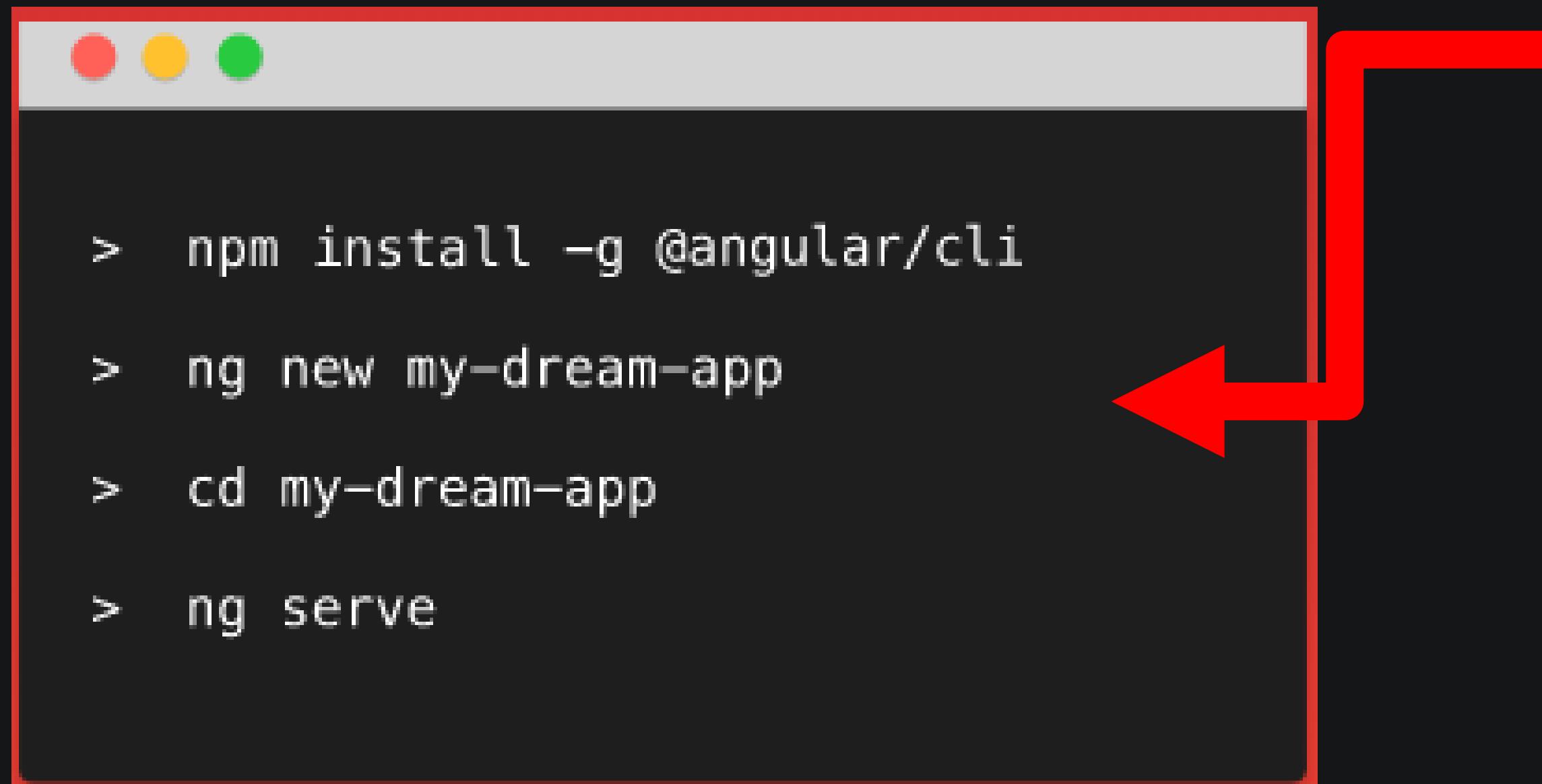
Một số khái niệm trong angular 6

- Module
- Components
- Phân tách module (Nhóm component) – SharedModule
- Databinding
- Directives
- Input, Output, ViewChild, ViewChildren
- Hướng dẫn angular Masterial
- 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

- Angular CLI là bộ công cụ do google cung cấp để phát triển ứng dụng bằng angular
- Angular Cli giúp tạo sẵn các cấu trúc thư mục ứng dụng và các tập tin cấu hình.

```
> npm install -g @angular/cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
```



Cú pháp để tạo thư mục cho ứng dụng
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

```
{ package.json *  
1  []  
2    "name": "angular6",  
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": "^6.1.0",  
15     "@angular/common": "^6.1.0",  
16     "@angular/compiler": "^6.1.0",  
17     "@angular/core": "^6.1.0",  
18     "@angular/forms": "^6.1.0",  
19     "@angular/http": "^6.1.0",  
20     "@angular/platform-browser": "^6.1.0",  
21     "@angular/platform-browser-dynamic": "^6.1.0",  
22     "@angular/router": "^6.1.0",  
23     "core-js": "^2.5.4",  
24     "rxjs": "^6.0.0",  
25     "zone.js": "~0.8.26"  
26   },  
}
```

Tương tự typescript thư mục package.json lưu trữ các thông tin của ứng dụng.

Bao gồm cả những gói tin thư viện được cài sẵn trong ứng dụng.

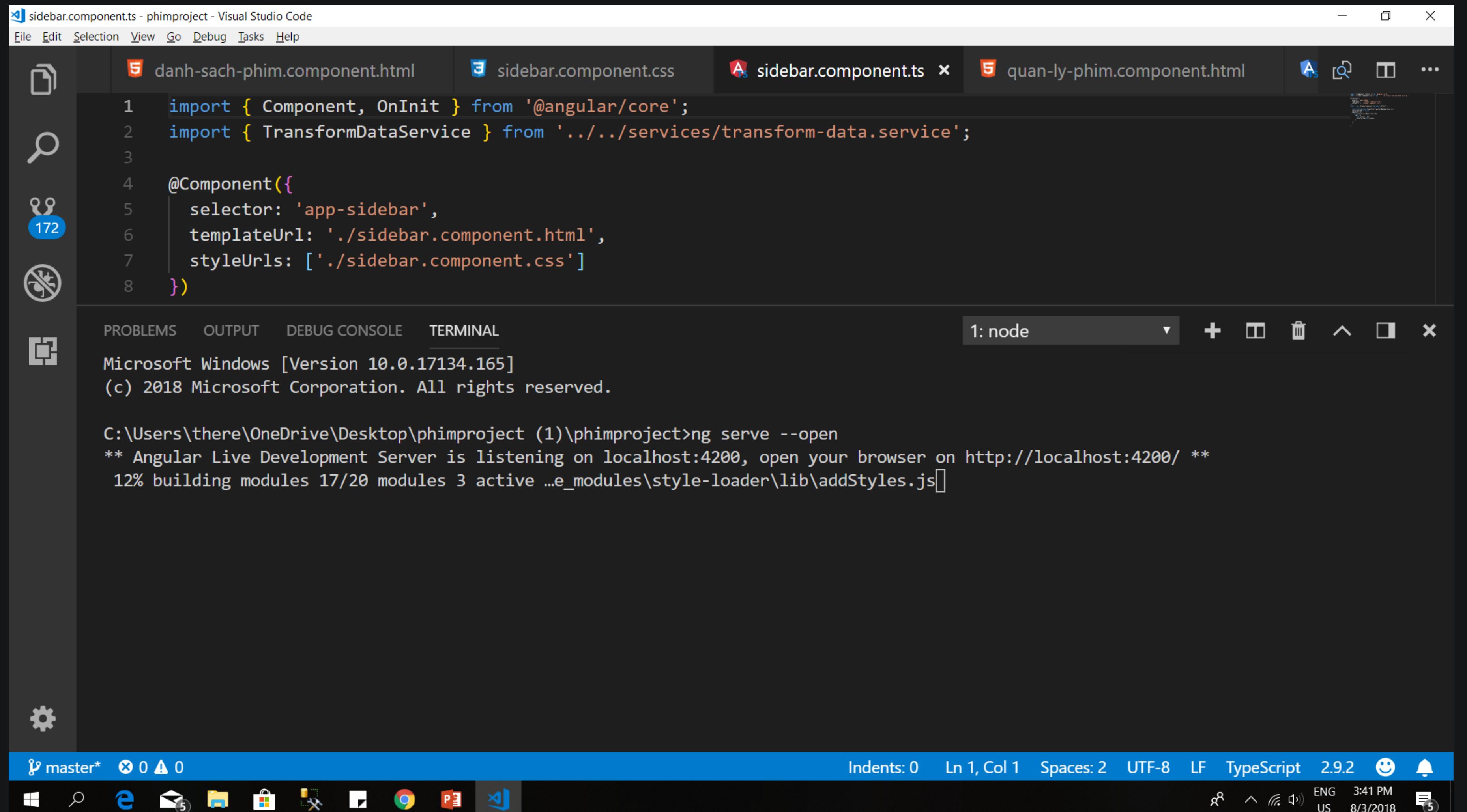
Trong đó chứa rất nhiều gói của angular 2 được cài sẵn.

```
{} tsconfig.json *  
1 {  
2   "compileOnSave": false,  
3   "compilerOptions": {  
4     "baseUrl": "./",  
5     "outDir": "./dist/out-tsc",  
6     "sourceMap": true,  
7     "declaration": false,  
8     "module": "es2015",  
9     "moduleResolution": "node",  
10    "emitDecoratorMetadata": true,  
11    "experimentalDecorators": true,  
12    "target": "es5",  
13    "typeRoots": [  
14      "node_modules/@types"  
15    ],  
16    "lib": [  
17      "es2017",  
18      "dom"  
19    ]  
20  }  
21}  
22
```

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

Cài đặt 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 Visual Studio Code interface. In the top bar, there are tabs for 'danh-sach-phim.component.html', 'sidebar.component.css', 'sidebar.component.ts', and 'quan-ly-phim.component.html'. The 'sidebar.component.ts' tab is active, displaying the following code:

```

1 import { Component, OnInit } from '@angular/core';
2 import { TransformDataService } from '../../../../../services/transform-data.service';
3
4 @Component({
5   selector: 'app-sidebar',
6   templateUrl: './sidebar.component.html',
7   styleUrls: ['./sidebar.component.css']
8 })

```

On the left side, there are icons for file operations, search, and other tools. The bottom left shows a terminal window with the following text:

```

Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\there\OneDrive\Desktop\phimproject (1)\phimproject>ng serve --open
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
12% building modules 17/20 modules 3 active ...e_modules\style-loader\lib\addStyles.js

```

The status bar at the bottom shows the current branch is 'master*', with 0 changes and 0 warnings. It also displays file statistics like 'Indent: 0', 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'TypeScript 2.9.2', and a smiley face icon.

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***

```
④ 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.

Kế đến main.ts sẽ gọi đến file app.module.ts

```
TS main.ts    X

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 //Câu lệnh dùng để khởi tạo appmodule để nó chạy đầu tiên
12 platformBrowserDynamic().bootstrapModule(AppModule) ←
13   .catch(err => console.log(err));
14
```

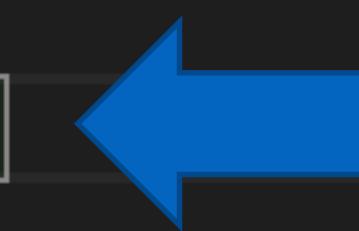
Dòng này khai báo cho angular biết được
AppModule là module gốc quản lý toàn bộ ứng
dụng web



app.module.ts

```
8  @NgModule({  
9    declarations: [  
10      AppComponent  
11    ],  
12    imports: [  
13      BrowserModule, HomeModule, HttpModule, AdminModule  
14    ],  
15    providers: [],  
16  
17    bootstrap: [AppComponent]  
18  })  
20  export class AppModule { }  
21
```

Nói cho angular biết được **AppComponent** là component gốc, vứt vào lấy html của nó parse ra là được



master* 0 0



Indent: 1 Ln 18, Col 28 Spaces: 2 UTF-8 LF TypeScript 2.9.2



ENG US 3:47 PM 8/3/2018

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:**
 - ❖ Browser module: 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 2 trên trình duyệt.
 - ❖ HttpModule, 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 module nào đó.**
- ❑ Đối với **app.module** thì đây là nơi bắt đầu khởi chạy của ứng dụng. Nó gọi là **module gốc** và nó **chứa 1 component gốc là app.component**.

Lấy ví dụ về app.module.ts (module chính trên toàn ứng dụng)

```
1 | Welcome TS app.module.ts X
2 |
3 | import { BrowserModule } from '@angular/platform-browser';
4 | import { NgModule } from '@angular/core';
5 | import { AppComponent } from './app.component';
6 | //Các thư viện được sử dụng cho module này
7 | //Thư viện bao gồm: component, module hoặc các dependencies khác...
8 |
9 | @NgModule([
10 |   declarations: [
11 |     AppComponent //Các component có thể import trực tiếp vào appmodule
12 |   ],
13 |   imports: [
14 |     BrowserModule //Các module khác bao gồm các module chứa những chức năng khác được import vào đây
15 |     /*Hoặc các module quản lý component được import vào đây
16 |      Thay vì ta import từng component tại declaration ta sẽ import 1 group component vào đây thông qua module đó*/
17 |   ],
18 |   providers: [], //Nơi đăng ký sử dụng service của app.module
19 |   bootstrap: [AppComponent] //Xác định component nào là component gốc để chạy ứng dụng
20 | ])
21 | export class AppModule { }
```

Cú pháp tạo module

ng g module [tên module]

2. Khái niệm về component

- Component biểu diễn giao diện UI (file.html).
- Nói 1 cách đơn giản 1 component là 1 thẻ do mình định nghĩa trong thẻ đó chứa các 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.

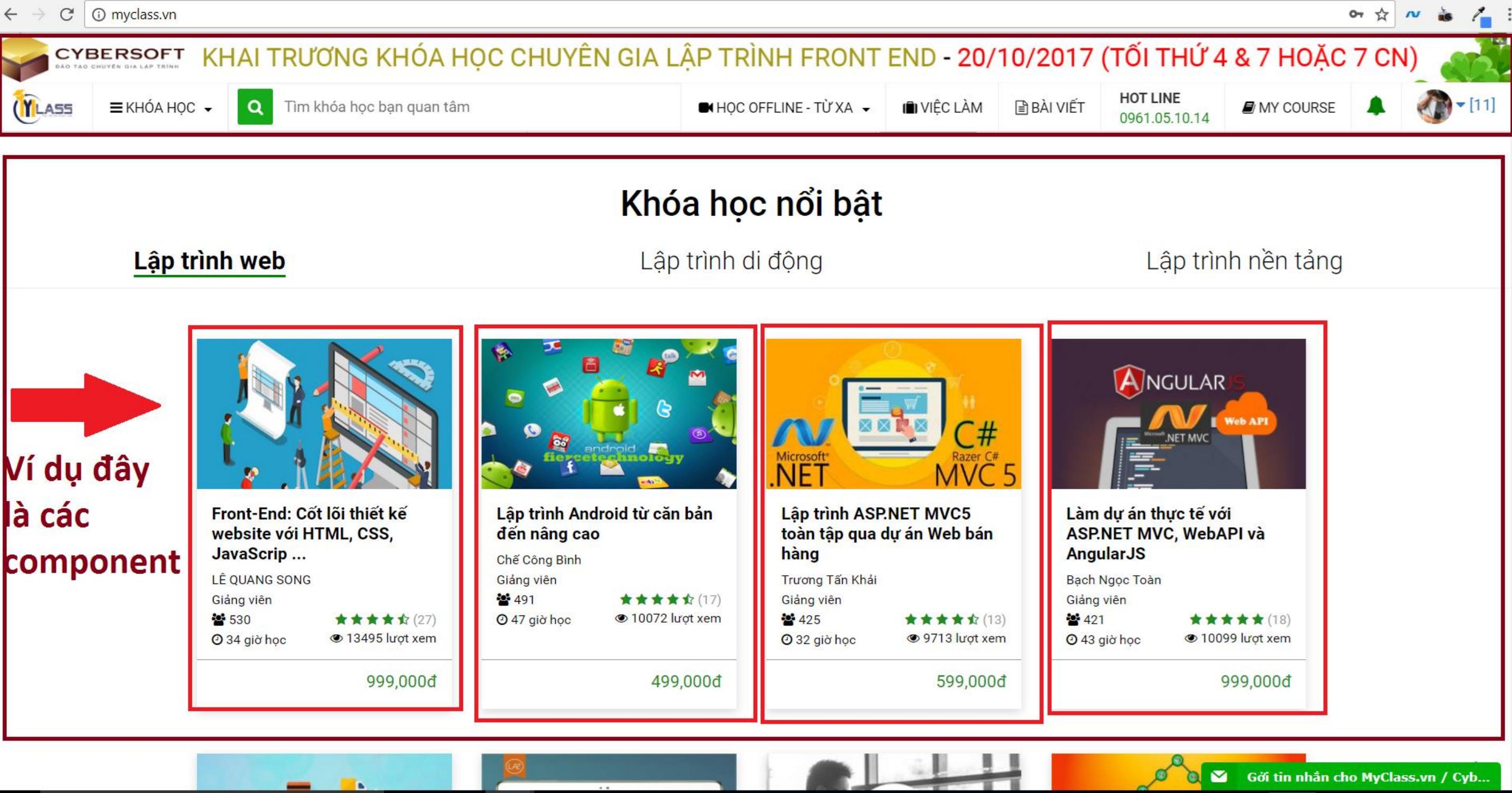
Ví dụ ta có ứng dụng web myclass.vn là app.module thì ta sẽ có component chứa tất cả các trang khác là app.component.

+ Trong app.component chứa TrangChuComponent.

+ Rồi trong trangchu lại chứa các thành phần khóa học component, ...

+ Trong app.component cũng chứa Trang chi tiết khóa học component

+ Rồi trong trang chi tiết cũng lại chứa các component khóa học và 1 số component khác

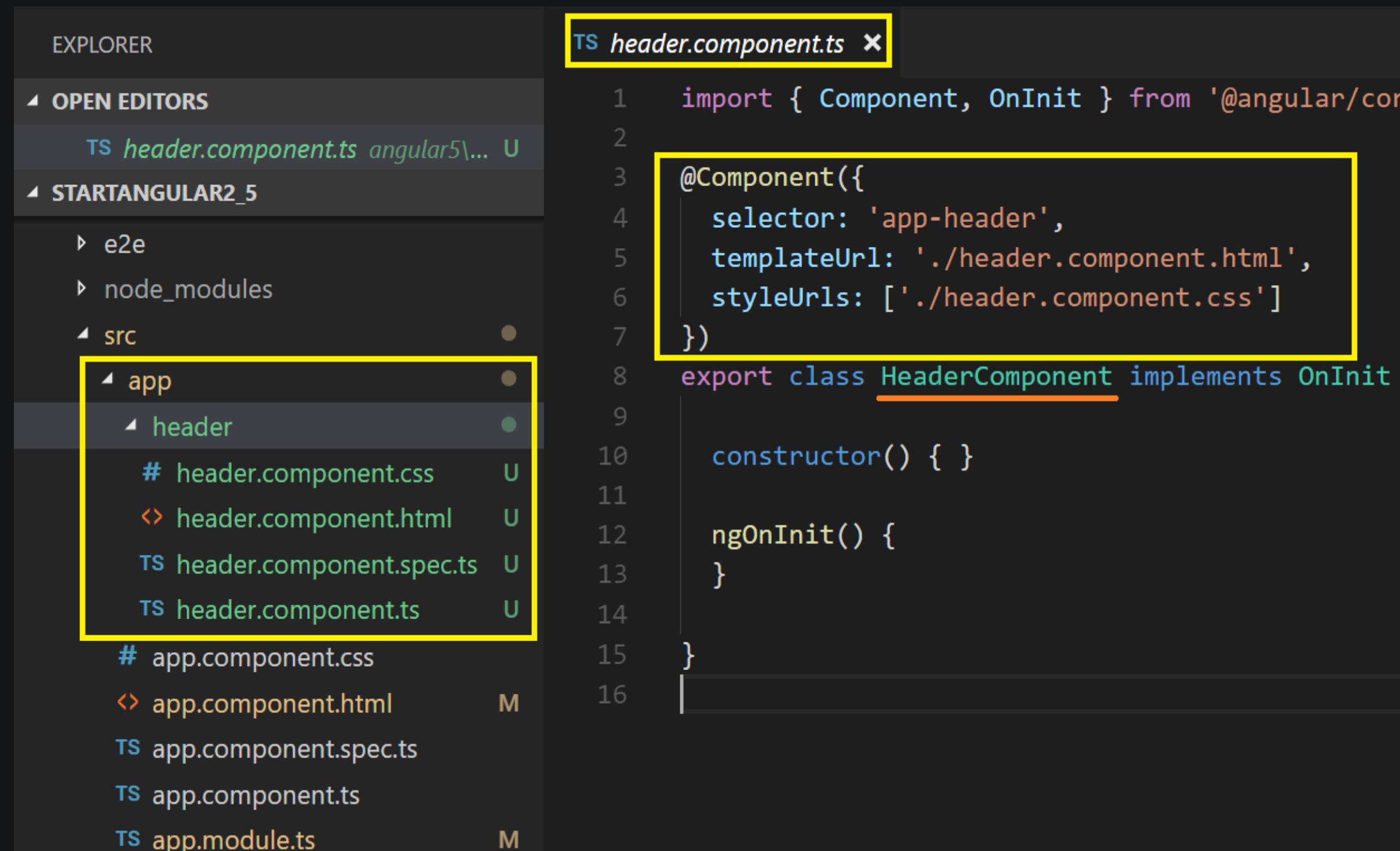


Khóa học nổi bật

Lập trình web	Lập trình di động	Lập trình nền tảng	
Ví dụ đây là các component  <p>Front-End: Cốt lõi thiết kế website với HTML, CSS, JavaScript ... LÊ QUANG SONG Giảng viên 530 Giảng viên 34 giờ học 999,000đ</p>	 <p>Lập trình Android từ căn bản đến nâng cao Chế Công Bình Giảng viên 491 Giảng viên 47 giờ học 13495 lượt xem 499,000đ</p>	 <p>Lập trình ASP.NET MVC5 toàn tập qua dự án Web bán hàng Trương Tấn Khải Giảng viên 425 Giảng viên 32 giờ học 9713 lượt xem 599,000đ</p>	 <p>Làm dự án thực tế với ASP.NET MVC, WebAPI và AngularJS Bạch Ngọc Toàn Giảng viên 421 Giảng viên 43 giờ học 10099 lượt xem 999,000đ</p>

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

Trong header chứa các file định nghĩa header component



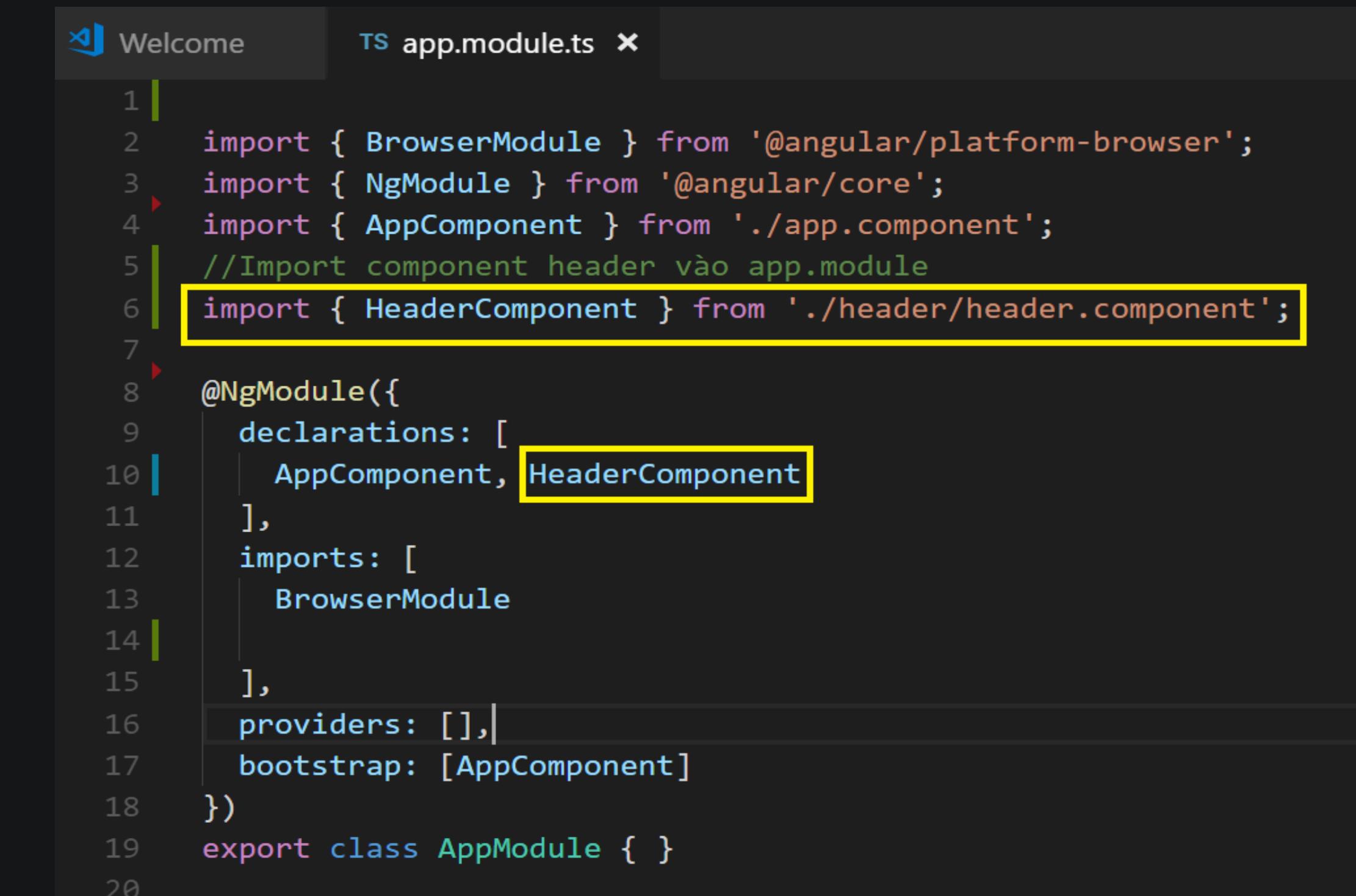
The screenshot shows the VS Code interface with the Explorer and Editor panes. In the Explorer pane, the file structure is visible under 'src': 'app' -> 'header'. The 'header' folder contains 'header.component.css', 'header.component.html', 'header.component.spec.ts', and 'header.component.ts'. The 'header.component.ts' file is open in the Editor pane. The code defines a component named 'HeaderComponent' with a selector of 'app-header' and a template URL of './header.component.html'. It also includes a style URL of './header.component.css' and implements the OnInit interface.

```

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 đăng ký cho app module quản lý HeaderComponent



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 also imports 'HeaderComponent' from './header/header.component'. The '@NgModule' block registers 'AppComponent' and 'HeaderComponent' as declarations, imports 'BrowserModule', and specifies 'AppComponent' as the bootstrap provider.

```

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 { }
19
20

```

Cú pháp tạo component

ng g component [tên component]

Để hiển thị HeaderComponent ta làm thế nào ???

EXPLORER

OPEN EDITORS

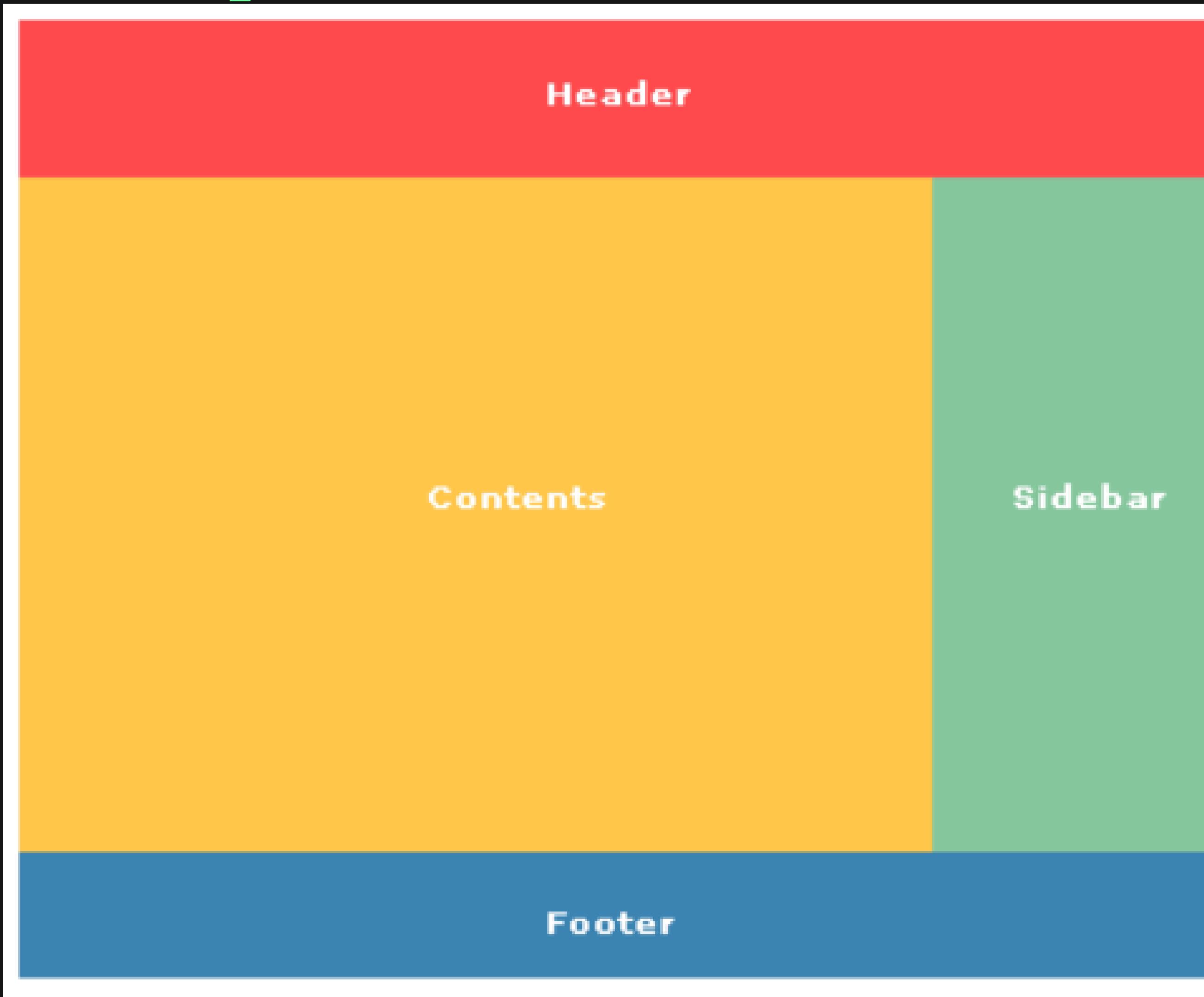
STARTANGULAR2_5

- e2e
- node_modules
- src
 - app
 - header
 - # header.component.css U
 - header.component.html U
 - TS header.component.spec.ts U
 - TS header.component.ts U

```
<> app.component.html x
1 | <app-header> </app-header>
```

```
> app.component.html      TS header.component.ts x
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 }
```

Bài tập:



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.

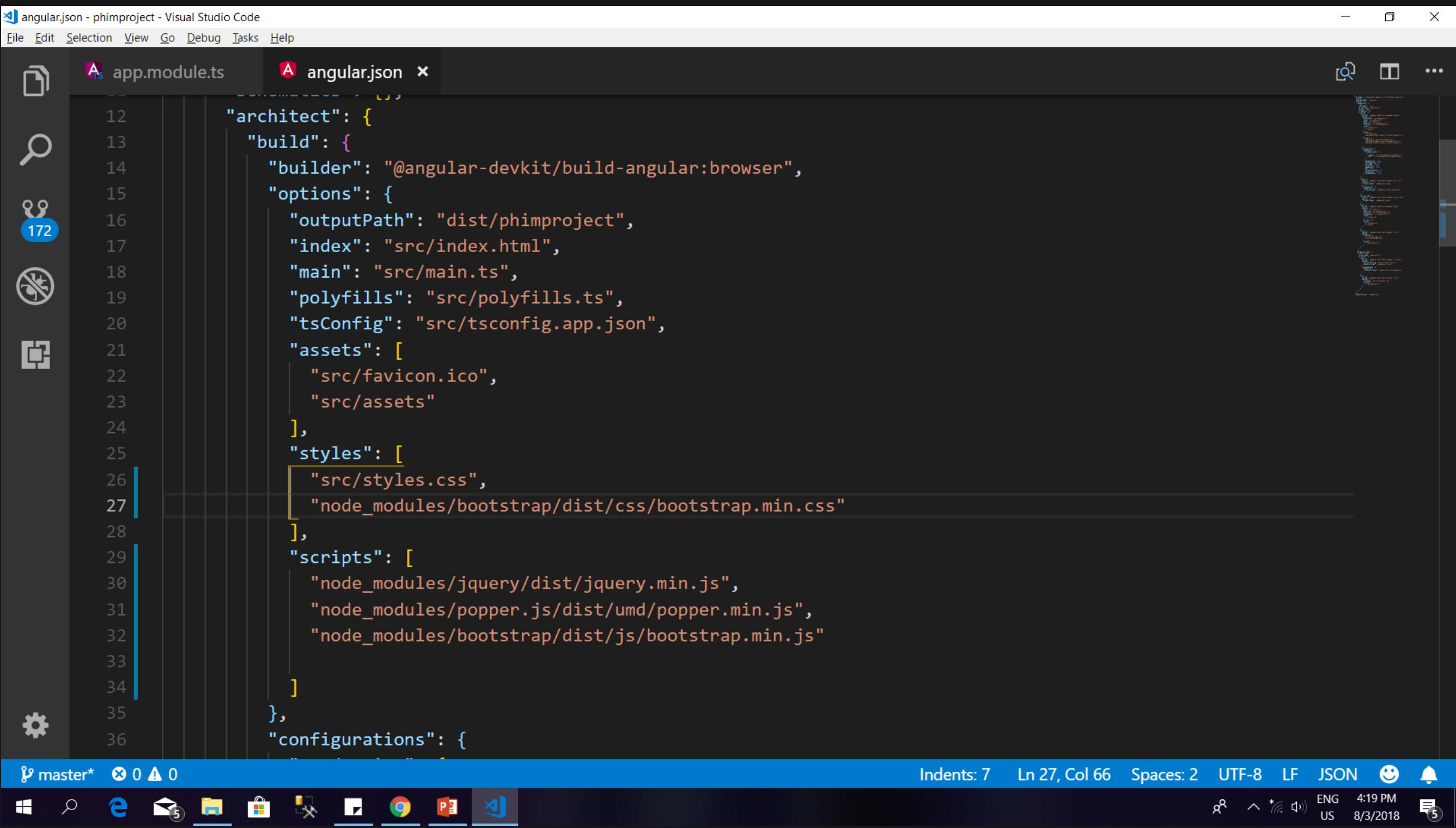
Hướng dẫn cài đặt bootstrap 4

❖ 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



```

{
  "architect": {
    "build": {
      "builder": "@angular-devkit/build-angular:browser",
      "options": {
        "outputPath": "dist/phimproject",
        "index": "src/index.html",
        "main": "src/main.ts",
        "polyfills": "src/polyfills.ts",
        "tsConfig": "src/tsconfig.app.json",
        "assets": [
          "src/favicon.ico",
          "src/assets"
        ],
        "styles": [
          "src/styles.css",
          "node_modules/bootstrap/dist/css/bootstrap.min.css"
        ],
        "scripts": [
          "node_modules/jquery/dist/jquery.min.js",
          "node_modules/popper.js/dist/umd/popper.min.js",
          "node_modules/bootstrap/dist/js/bootstrap.min.js"
        ]
      },
      "configurations": {}
    }
  }
}

```

Bài tập :

headercomponent

IndexComponent (viền vàng)

Business Name or Tagline
1920 x 400

SliderComponent (Viền tím)

What We Do
Lorem ipsum dolor sit amet, consectetur adipisicing elit. A deserunt neque tempore recusandae animi soluta quasi? Asperiores rem dolore eaque vel, porro, soluta unde debitis aliquam laboriosam. Repellat explicabo, maiores!

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Omnis optio neque consectetur consequatur magni in nisi, natus beatae quidem quam odit commodi ducimus totam eum, alias, adipisci nesciunt voluptate. Voluptatum.

Call to Action »

ItemComponent (Lưu ý thiết kế 1 component sau đó copy ra 3 component)

300 x 200

Card title
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sapiente esse necessitatibus neque sequi doloribus.

Find Out More!

300 x 200

Card title
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sapiente esse necessitatibus neque sequi doloribus totam ut praesentium aut.

Find Out More!

300 x 200

Card title
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sapiente esse necessitatibus neque.

Find Out More!

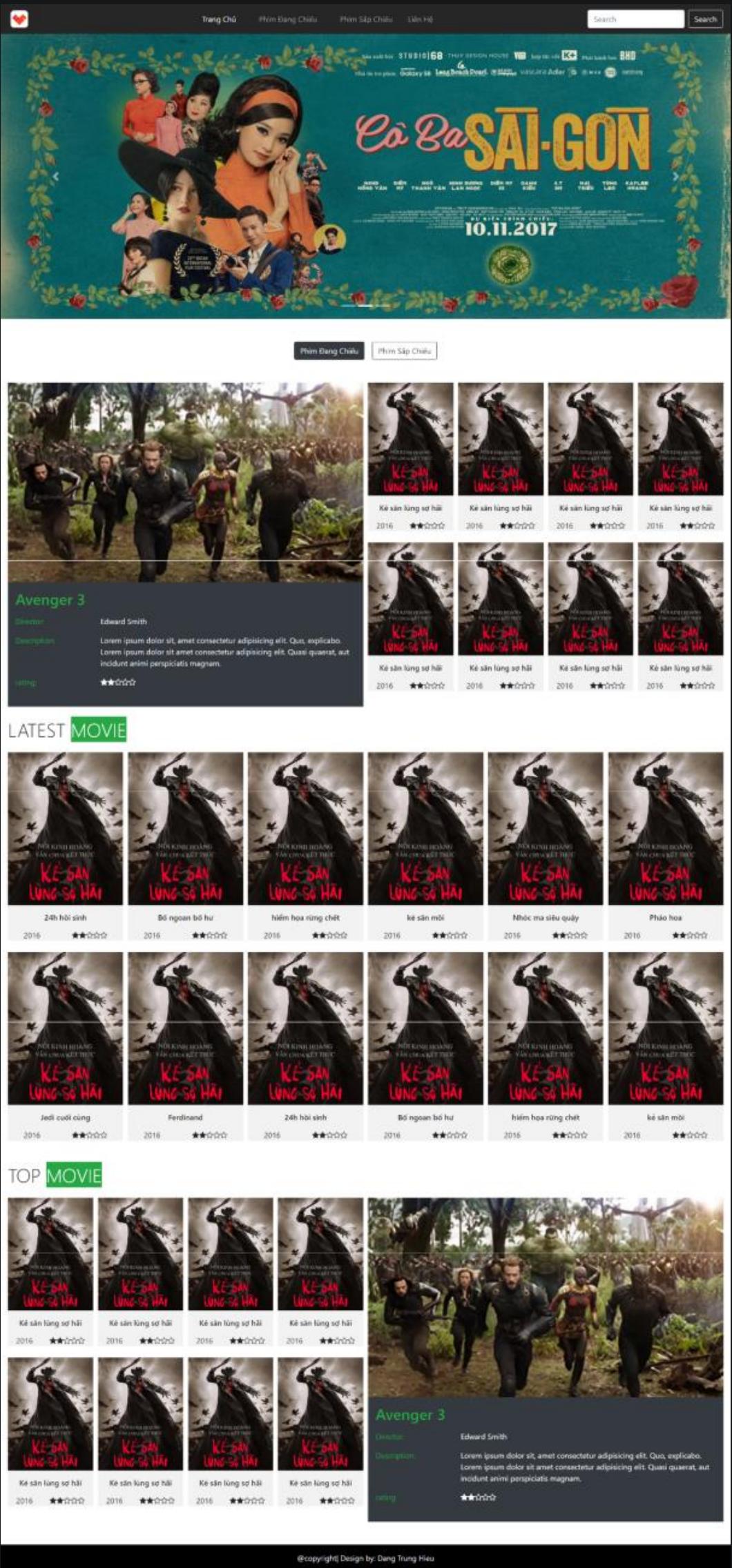
FooterComponent

Copyright © Your Website 2017

Yêu cầu:

- + Tạo và thiết kế component tương ứng.
 - + Bổ 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 dàn layout có nội dung:



- Yêu cầu phân tích xem trang này có bao nhiêu component.
- Xây dựng các component tương ứng.

3. Phân tách module – Nhóm các component

Module TrangChu

Component TrangChu

Component
SanPham

Component
SanPham

Component
SanPham

Component
SanPham

- ❖ Module là gì ?
 - ❖ Tại sao phải phân tách module?
 - ❖ Phân tách module như thế nào ?
- ✓ **Module** là **1 nhóm chứa** các **component** để **thực hiện** **1 chức năng** nào đó của ứng dụng.
- ✓ 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**.

AppModule

AppComponent

Module Home

Component TrangChu

Component
Slider

Component
header

Component
Content

Component
Footer ...

Module AdminQLSP

Component Them, Sua, XemChiTiet

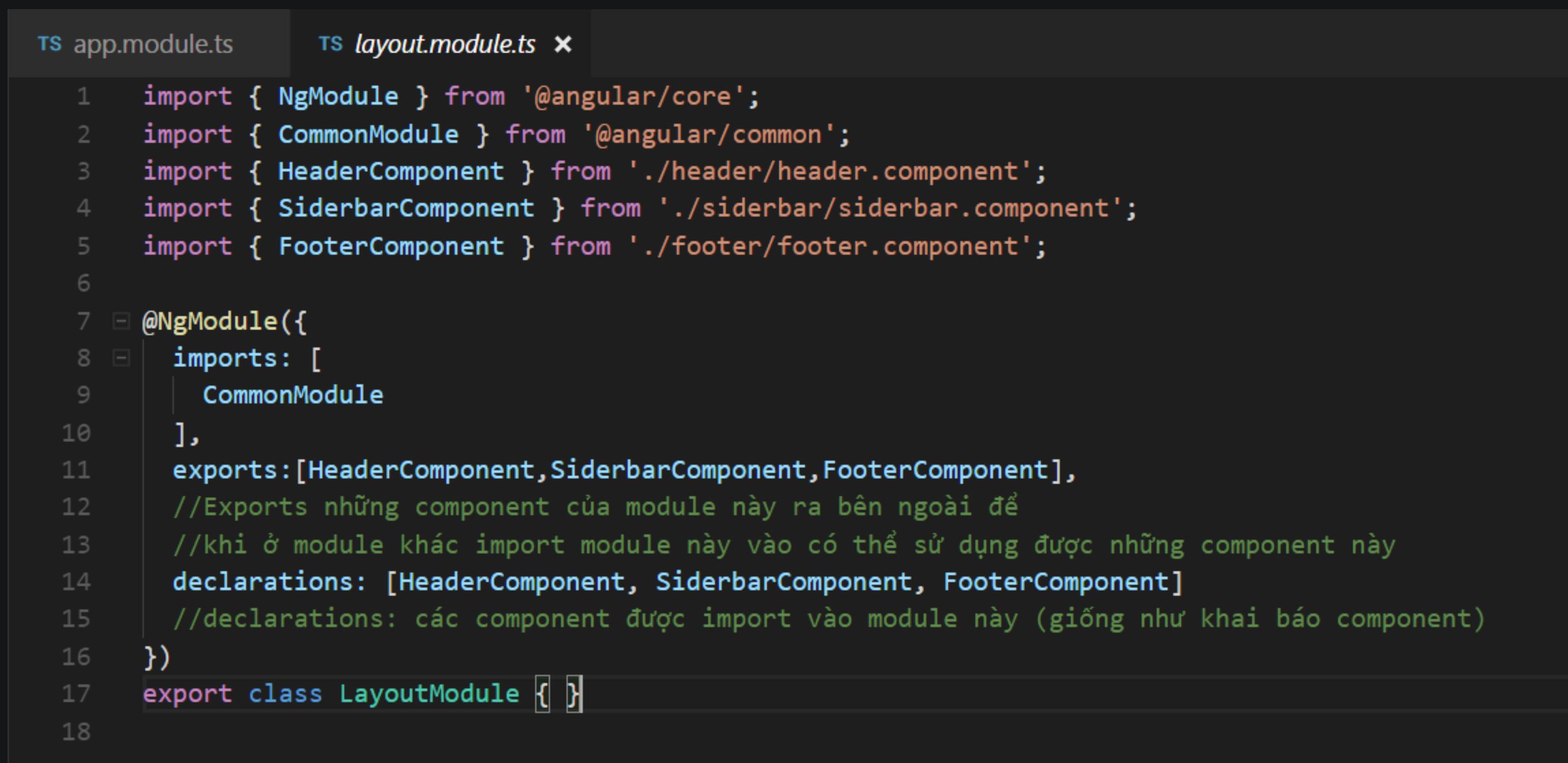
Component
ThemSP

Component
SuaSP

Component
XemChiTiet

Component
...

Lấy ví dụ về Layout module (Một module con)



```
TS app.module.ts      TS layout.module.ts ×

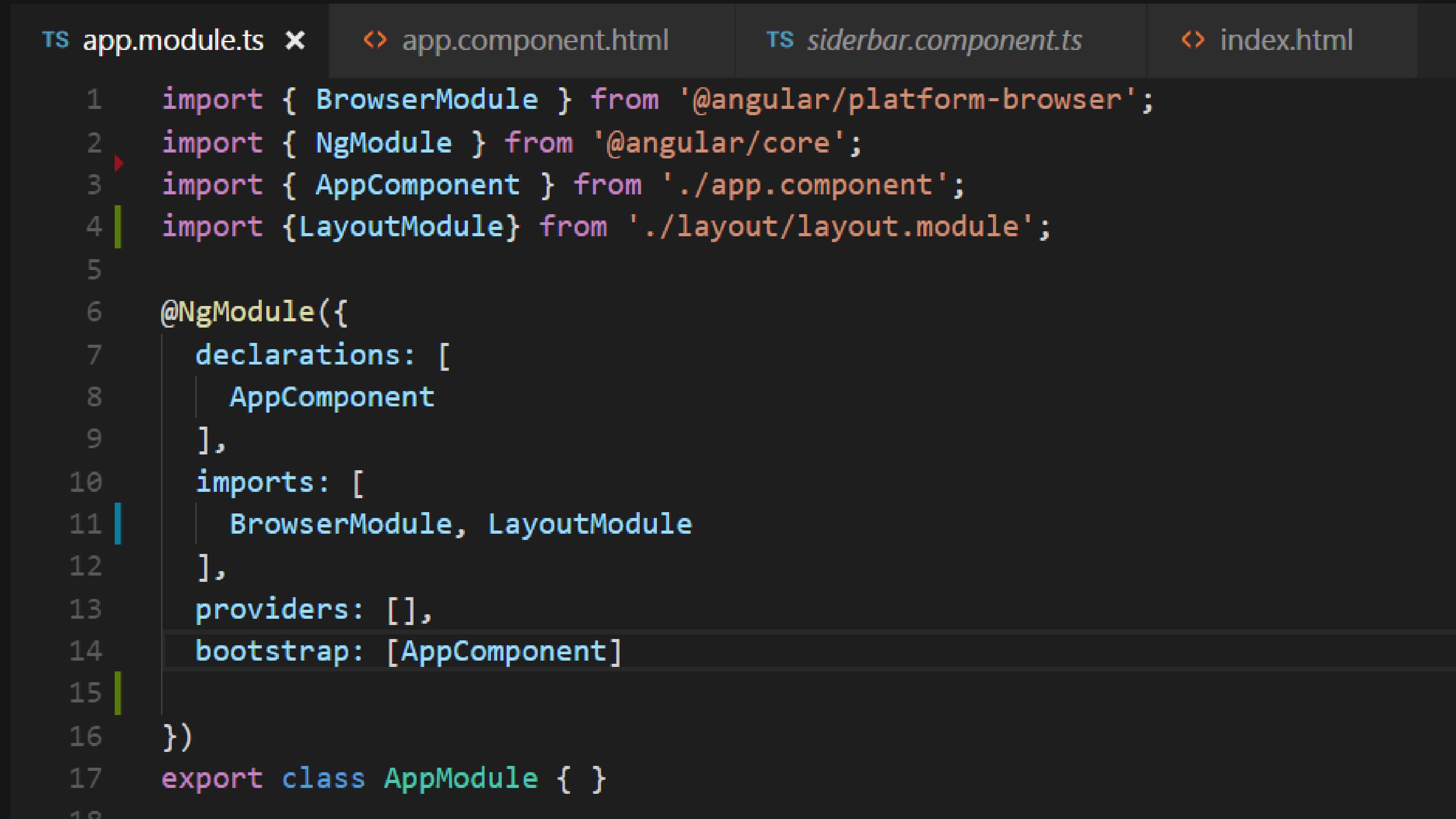
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { HeaderComponent } from './header/header.component';
4 import { SiderbarComponent } from './siderbar/siderbar.component';
5 import { FooterComponent } from './footer/footer.component';
6
7 @NgModule({
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**

Câu hỏi đặt ra 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

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



```
TS app.module.ts ×  ◊ app.component.html      TS sidebar.component.ts      ◊ index.html
  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 )
 17 export class AppModule { }
```

Khái niệm về Template và Styles

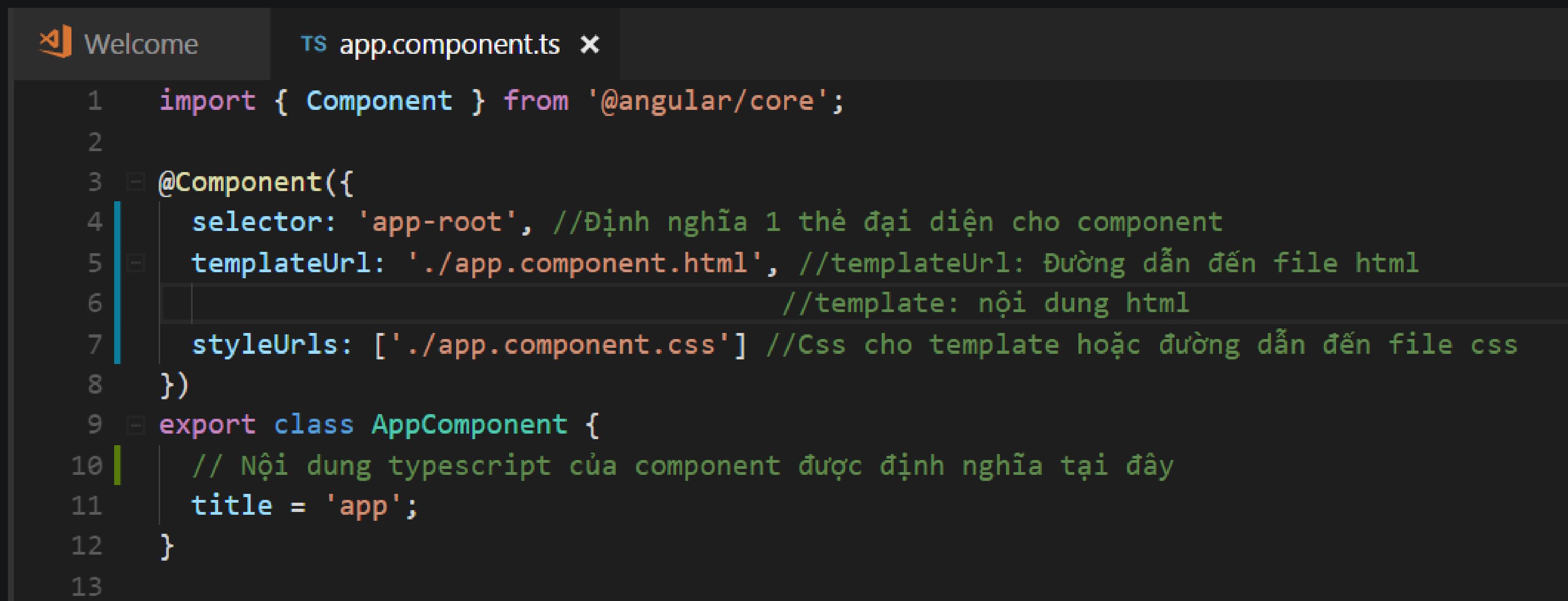
Template là 1 thành phần html của component.

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**



```
VS Code Editor showing app.component.ts file. The code defines an Angular component with the following properties:  
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: nội dung html  
7   styleUrls: ['./app.component.css'] //Css cho template hoặc đường dẫn đến file css  
8 })  
9 export class AppComponent {  
10   // Nội dung typescript của component được định nghĩa tại đây  
11   title = 'app';  
12 }  
13
```

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
```

4. Data-binding

- Data binding là 1 tính năng 1 đặc tính 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

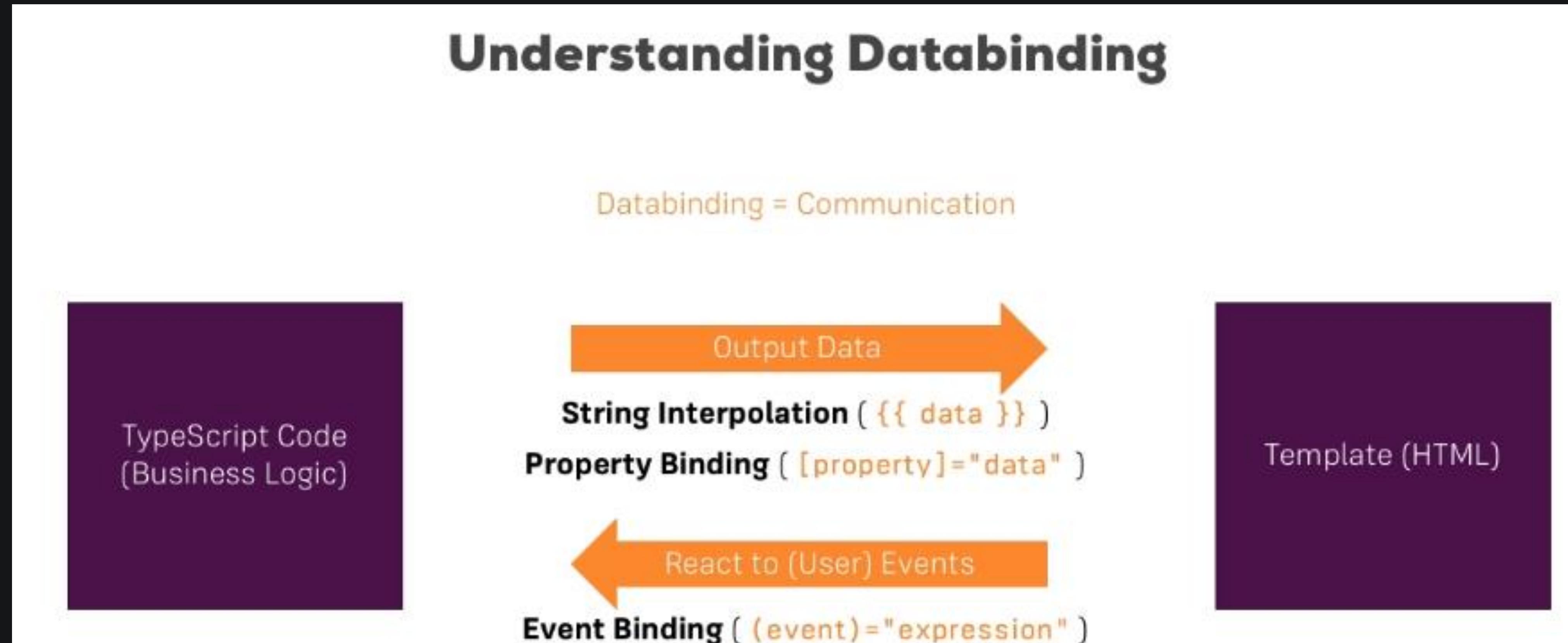
❑ Các cơ chế binding dữ liệu angular 2

❖ One-waybinding

- Interpolation
- Property binding
- Event binding

❖ Two-waybinding

❖ Oneway binding



❖ Oneway binding (Interpolation)

```
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 }  
14
```

Thế nào là one-way binding **one-way binding sử dụng các biến trong class component** binding ra view hay còn gọi là template (**Interpolation Binding**) hoặc ngược lại thông qua sự kiện (**eventbinding**). **Interpolation Binding** được viết với cú pháp **{{tenbien}}** hoặc **{{tenham()}}**. Nếu dữ liệu model thay đổi => view thay đổi.

❖ Oneway 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 []

Ví dụ:

```
TS propertybinding.component.ts ×  
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]**

❖ 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 ×
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 }
21
```

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

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

❖ Event Binding(tt)

Event binding với tham số

```
ts eventbinding.component.ts ×

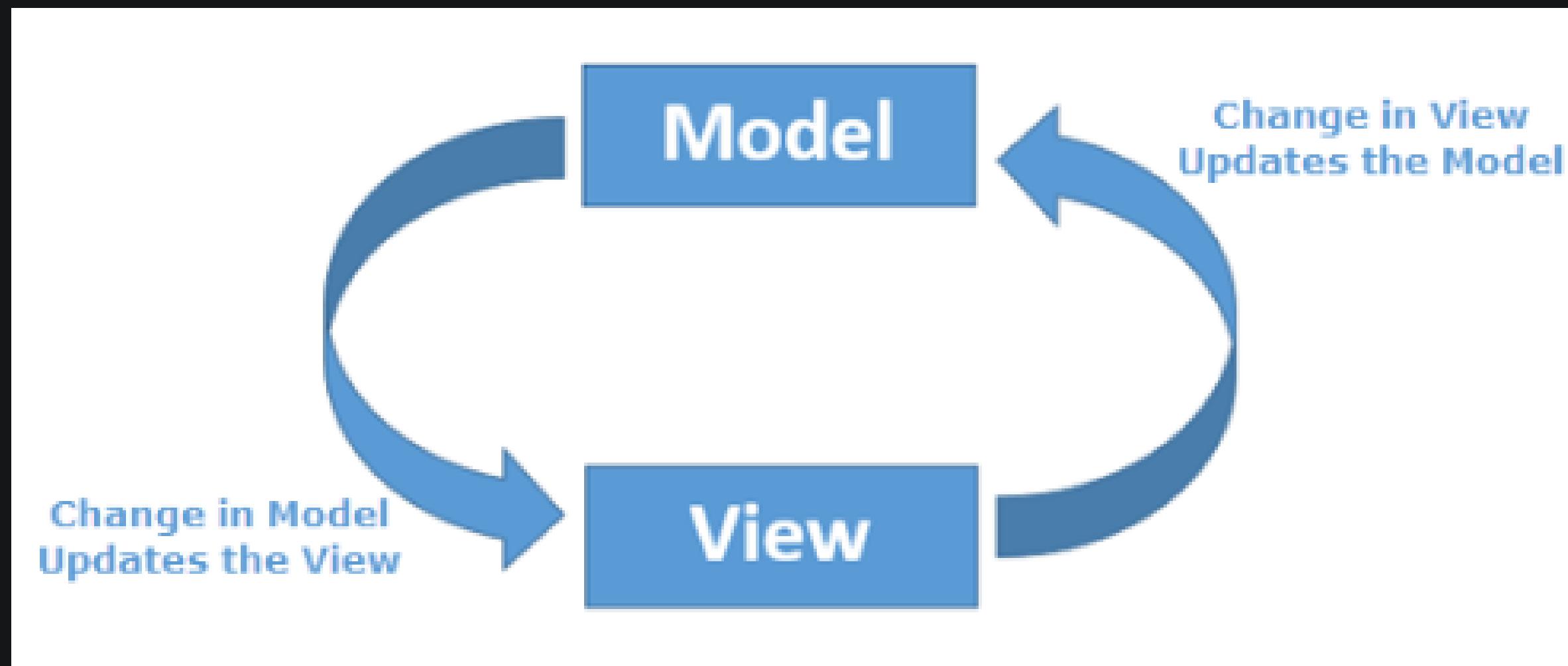
1 import { Component, OnInit } from '@angular/core';
2 @Component({
3   selector: 'app-eventbinding',
4   template: `
5     <input type="text" #thamso index='12' />
6     <button (click)="DisplayName(thamso)">Submit</button>
7   `,
8   styleUrls: ['./eventbinding.component.css']
9 })
10 export class 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 }
22
```

#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**

❖ 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:

Register form

Email:

FullName:

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

❖ Structural – directives

Dùng để thay đổi diện mạo của dom ví dụ như ẩn hiện, load dữ liệu ...

- *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**

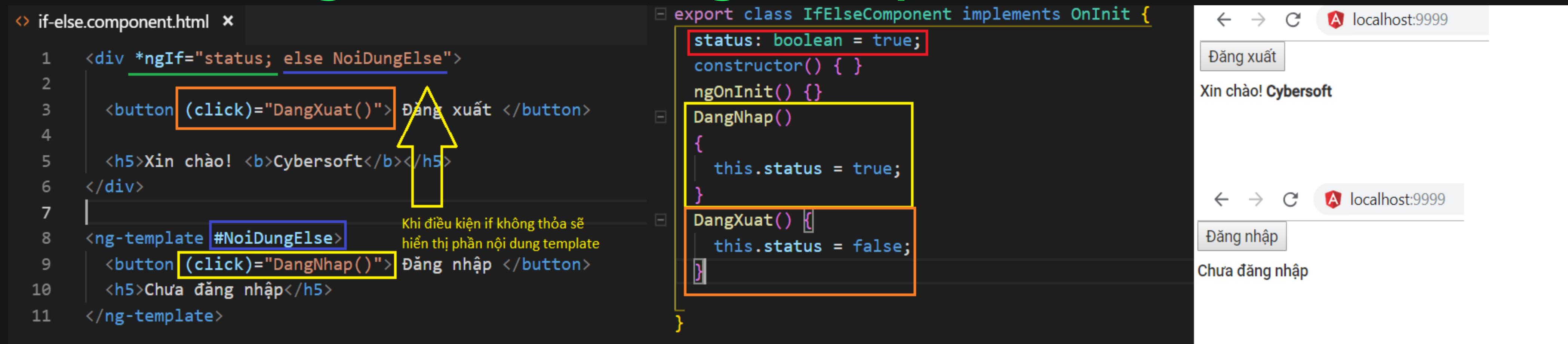
- **ng-template:**

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).

Ví dụ về ngIf else và ng-template:



```

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

```

```

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

```

Đăng xuất

Xin chào! Cybersoft

Đăng nhập

Chưa 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 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.

*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"> Màu đỏ </div>
8       <div *ngSwitchCase="'blue'" style="color:blue"> Màu xanh </div>
9       <div *ngSwitchCase="'green'" style="color:green"> Màu xanh lá </div>
10      <div *ngSwitchDefault style="color:orange"> Màu cam mặc định </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   }
26 }
```

[ngSwitch] = "điều kiện"
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 }
```

*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 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 ?

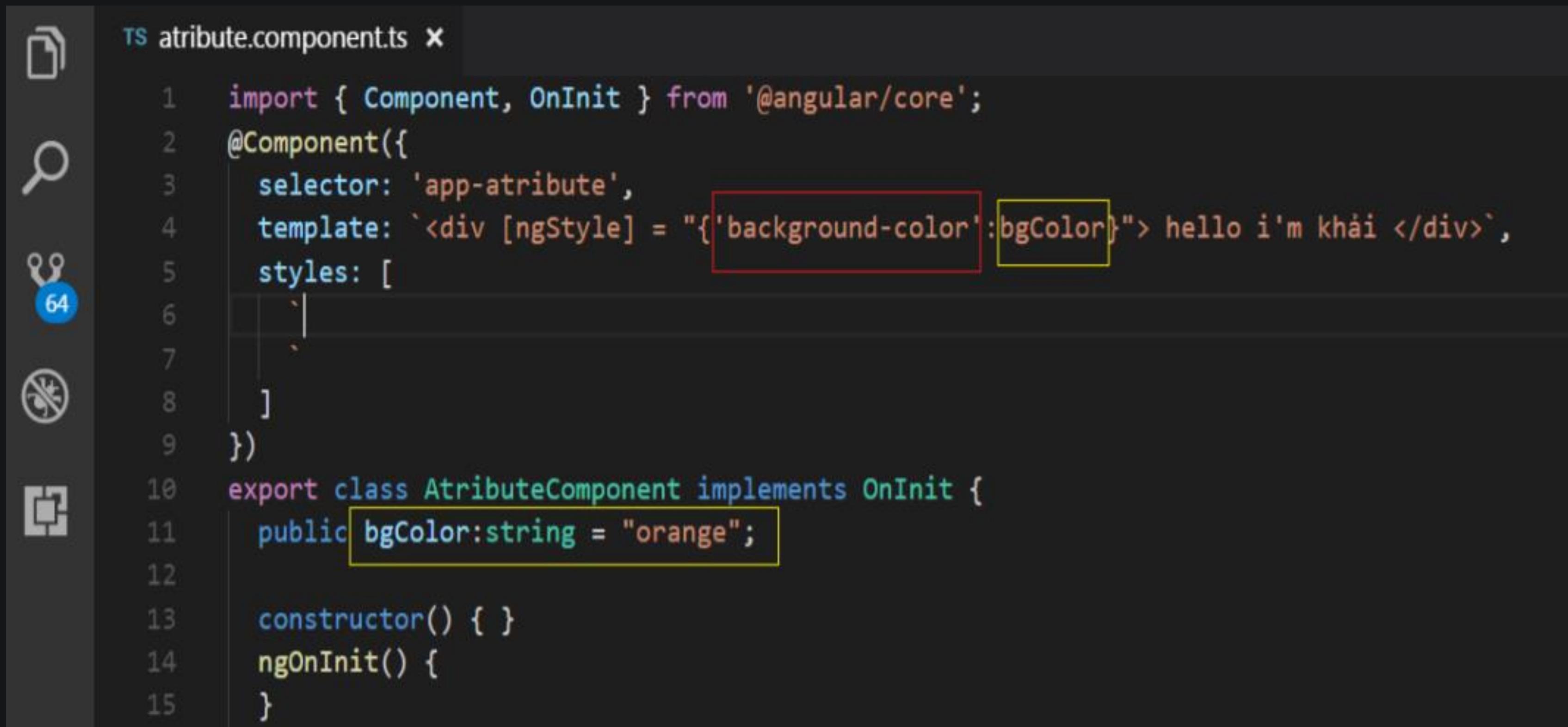
❖ Attribute – directives

***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() {
  }
}

```

app.component.html baitap2.component.html baitap2.component.ts

NewProject > src > app > baitap2 > baitap2.component.html > span

View

2 {{name}}

Browser

localhost:4200

{{name}}

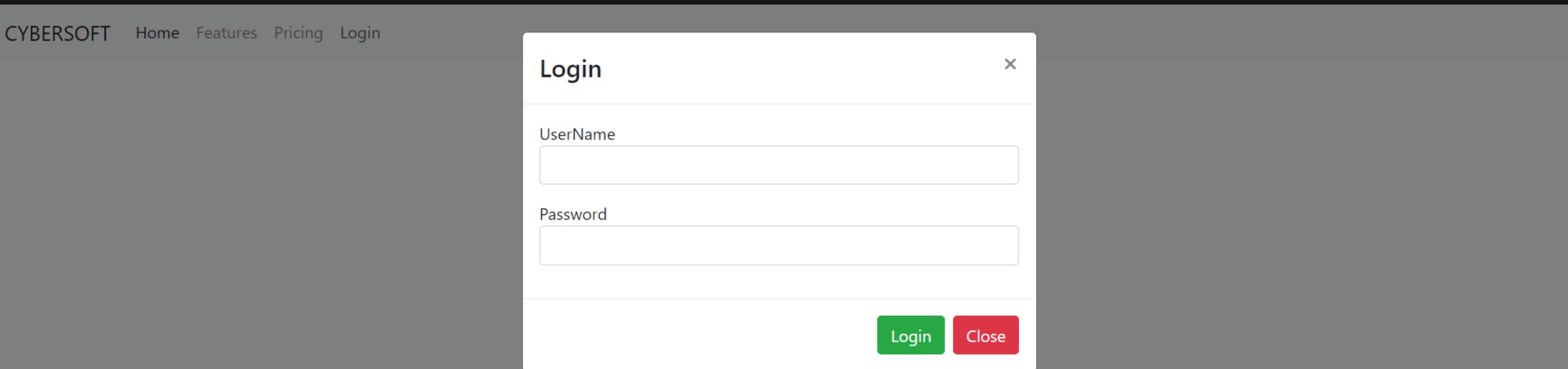
❖ Bài tập

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.



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.

Quản lý danh mục sản phẩm

Mã SP	Tên SP	Giá
MTB	Máy tính bảng	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

5. Directive (Attribute directive)

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

Có 3 loại directive:

- ❖ Components – directives
- ❖ Structural – directives
- ❖ Attribute – directives

❖ Attribute - Directive

*Attribute directive (Tự định nghĩa): Để định nghĩa 1 attribute directive cũng giống như định nghĩa 1 component tuy nhiên directive thì không có thành phần template hay css.

```
TS high-light-directive.directive.ts ×

1 import { Directive, ElementRef, OnInit } from '@angular/core';
2
3 @Directive({
4   selector: '[appHighLightDirective]' ↑
5 })                                Lớp đối tượng dùng để DOM
6 export class HighLightDirective implements OnInit{
7
8   constructor(private elementRef: ElementRef) { →
9     Khai báo đối tượng đại diện cho thành phần chứa directive
10 }
11 ngOnInit(){
12   this.elementRef.nativeElement.style.backgroundColor = 'green';
13 }                                     Cú pháp dùng để DOM tương tự JS : Gán background = màu xanh lục
14 }
```

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 – Directive (DOM)

***Renderer2:** 1 tính năng của angular 4 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 – Directive (Event)

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

```
export class HighLightDirectiveDirective 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 – Directive (Event)

***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';

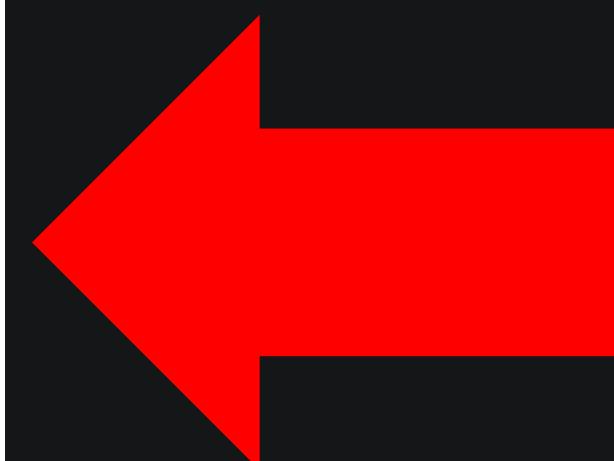
@HostListener('mouseenter') SuKienHover(eventData:Event)
{
    this.bgColor = 'green' ;
}

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

❖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>
```



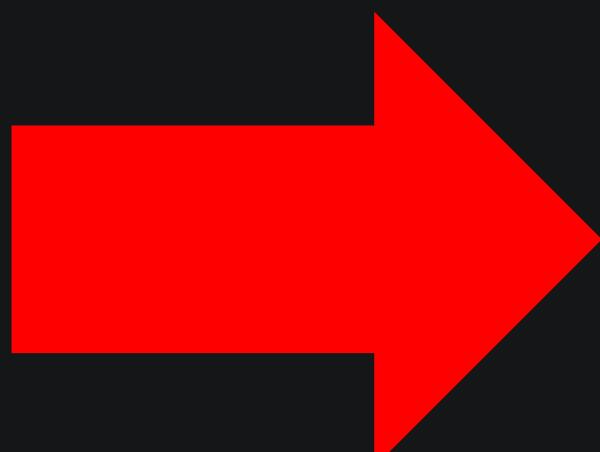
Lỗi

❖ 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

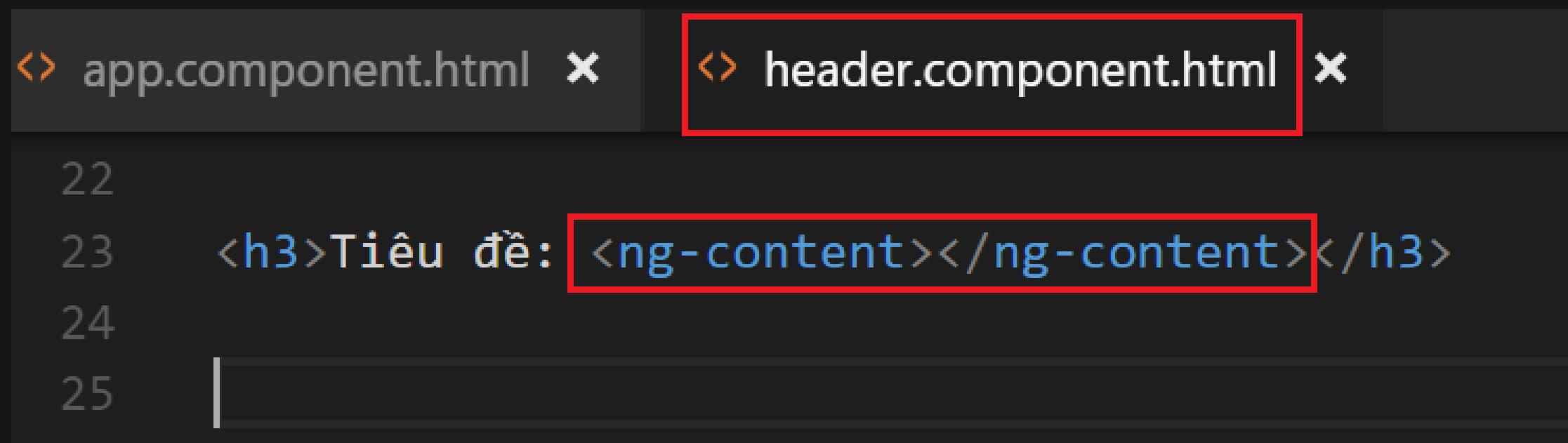
```
<ng-container *ngTemplateOutlet="template"> </ng-container>
<ng-template #template>
    <div> Nội dung template </div>
</ng-template>
```

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

❖ ng-content

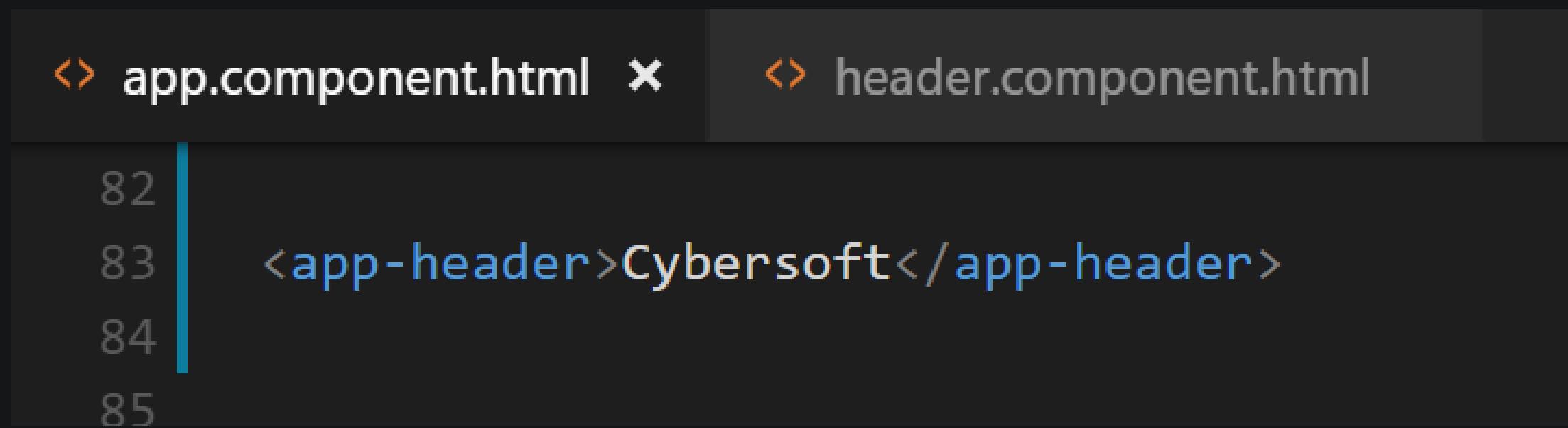
Sử dụng khi ta muốn chèn nội dung từ component cha đến component con tại vị trí chỉ định.

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
```

❖ Bài tập:

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

MaSP	TenSP	Gia	Thêm sản phẩm
------	-------	-----	----------------------

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 »

❖ 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>

6. Input, Output, ViewChild, ViewChildren

- **Input:** là thao tác truyền dữ liệu từ component cha vào component con thông qua thuộc tính Input của component đó (**component con là component khai báo input**).
- **Output:** Dùng để **đẩy giá trị component con ra** component cha chứa nó thông qua sự kiện (keyup, click, change, ...) đại diện bởi đối tượng EventEmitter.
- **ViewChild:** Dùng để **DOM đến component hoặc thẻ trên thành phần html của đối tượng đó**, để sử dụng các thuộc tính, phương thức của class component đó.
- **ViewChildren:** Dùng để **DOM đến danh sách các thẻ hoặc danh sách các component** có cùng **selector**. (Tương tự each bên Jquery nhưng thể hiện dưới dạng hướng đối tượng hơn)

Input

Là thao tác truyền dữ liệu từ component này (cha) sang component khác.

```
TS sinh-vien.component.ts ×
1 import { Component, OnInit, Input } from '@angular/core';
2
3 @Component({
4   selector: 'app-sinh-vien',
5   template: `
6     <div class="SinhVien">
7       Họ tên: {{HoTen}} <br />
8       Tuổi: {{Tuoi}}
9     </div>
10   `,
11   styleUrls: ['./sinh-vien.component.css']
12 })
13 export class SinhVienComponent implements OnInit {
14   @Input() HoTen:string;
15   @Input() Tuoi:string;
16   constructor() { }
17
18   ngOnInit() {
19   }
20 }
```

```
sinh-vien.component.ts TS danh-sach-sinh-vien.component.ts ×
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-danh-sach-sinh-vien',
5   template: `<app-sinh-vien *ngFor="let sv of DanhSachSinhVien"
6   [HoTen]="sv.Ho_Ten"
7   [Tuoi]="sv.Tuoi"></app-sinh-vien>`,
8   styleUrls: ['./danh-sach-sinh-vien.component.css']
9 })
10 export class DanhSachSinhVienComponent implements OnInit {
11
12   private DanhSachSinhVien = [
13     {Ho_Ten:"Nguyễn văn a",Tuoi:15},
14     {Ho_Ten:"Nguyễn văn b",Tuoi:18},
15     {Ho_Ten:"Nguyễn văn c",Tuoi:19}
16   ]
17   constructor() { }
18
19   ngOnInit() {
20   }
21 }
22 }
```

Dữ liệu được truyền từ component a => b thông qua tham số **name** và **age**

Output

Component con (GheNgoicomponent)

```

@Component({
  selector: 'ghe-ngoi',
  template: `
    <button class="ghe chuatad" [ngClass]="{chuadat:!trangthai,dadat:trangthai}" (click)="DatGhe(trangthai)"></button>
    ,
    styles: [
      `.ghe{
        width:50px;
        height:50px;
      }
      `.dadat{
        background-color:red;
      }
      `.chuadat{
        background-color:green;
      }
    ]
  )
}

```

Component xác định class đã đặt hay chưa đặt dựa vào giá trị biến của biến **trangthai**

```

export class GhengoiComponent implements OnInit {
  public trangthai = false;

  @Output () eventDatGhe = new EventEmitter();
  DatGhe(value:boolean) Để đẩy được giá trị biến(khi thay đổi)
  {
    if(value == true) ra ngoài ta phải thông qua 1 sự kiện
    {
      this.trangthai =false;
    }else
    {
      this.trangthai =true;
    }
    //Đẩy biến trạng thái ra component cha khi có sự thay đổi
    this.eventDatGhe.emit(this.trangthai);
  }
  public setname(name:string) Giá trị biến đẩy ra được truyền
  {
    console.log(name);
    {
      background-color:orange;
    }
  }
}

```

↑
Giá trị biến đẩy ra được truyền vào phương thức **emit()**

- Ta định nghĩa 1 component ghế ngồi.
- Trong component ghế ngồi ta định nghĩa trạng thái của ghế:
 - + Nếu trạng thái ghế là true: ghế chưa đặt (áp dụng css class chưa đặt)
 - + Nếu trạng thái ghế là false: ghế đã được đặt (áp dụng css class ghế đã đặt)
- Thông qua sự kiện click ta thay đổi giá trị trạng thái của ghế. Và ta muốn để giá trị này ra ngoài component sach-ghe. Để làm gì để component đó tính được bao nhiêu ghế đã đặt và bao nhiêu ghế chưa đặt

Component cha (DanhSachGheNgoiComponent)

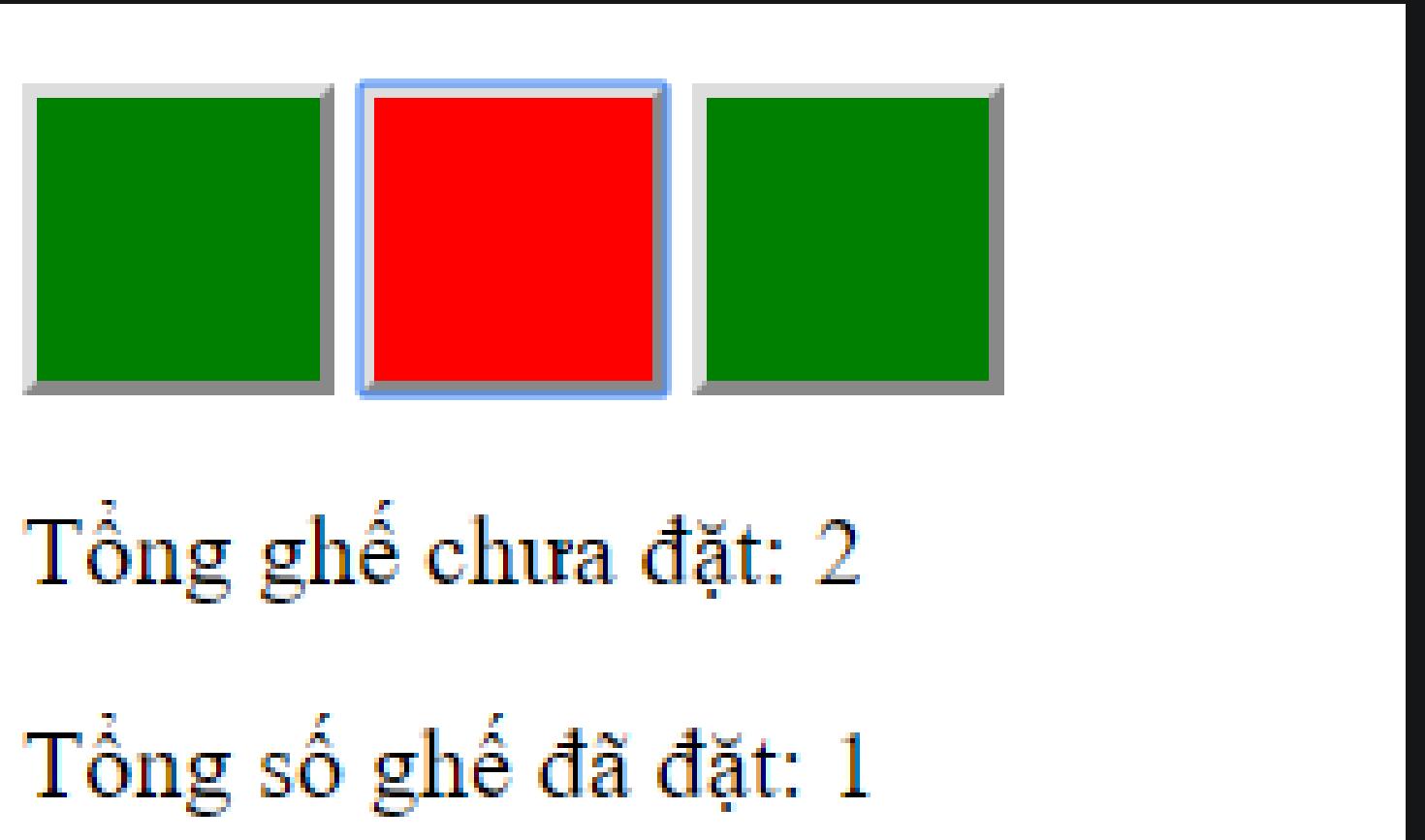
```
import { Component, OnInit, ViewChild, ViewChildren, QueryList } from '@angular/core';
import { GhengoiComponent } from '../ghengoi/ghengoi.component';

@Component({
  selector: 'danh-sach-ghe-ngoi',
  template: `
    <ghe-ngoi #t *ngFor="let ghe of DanhSachGhe" (eventDatGhe) = 'Dat_Ghe_Parent($event,ghe.maghe)'></ghe-ngoi>
    <p>Tổng ghế chưa đặt: {{SoGheChuaDat}}</p>
    <p>Tổng số ghế đã đặt: {{SoGheDaDat}}</p>
  `,
  styleUrls: ['./c.component.css']
})
export class DanhsachghengoiComponent implements OnInit {

  public DanhSachGhe:Array<any> = [{maghe:"ghe1",trangthai:false},{maghe:"ghe2",trangthai:false},{maghe:"ghe3",trangthai:false}];
  public SoGheDaDat:number = 0;
  public SoGheChuaDat:number = this.DanhSachGhe.length;
  Dat_Ghe_Parent(trangthaighe:boolean,maghe) //Biến được gửi từ component con sang
{
  alert(maghe);
  if(trangthaighe == true)
  {
    this.SoGheDaDat++;
    this.SoGheChuaDat--;
  }
  else
  {
    this.SoGheDaDat--;
    this.SoGheChuaDat++;
  }
}
```

Đặc tả:

Tại sao ta lại dùng Output. Bởi vì khi ta tạo ra component ghế từ danh sách ghế ta đâu thể biết trước và định nghĩa bao nhiêu sự kiện click trên bao nhiêu cái ghế ? Một điều nữa 2 component này hiện tại code TypeScript đang độc lập với nhau thuộc 2 class đối tượng khác nhau. Do vậy không thể lấy được giá trị từ component kia khi nó click được. Nên cách duy nhất ta sử dụng là thông qua binding = Output().



View Child (DOM đến component)

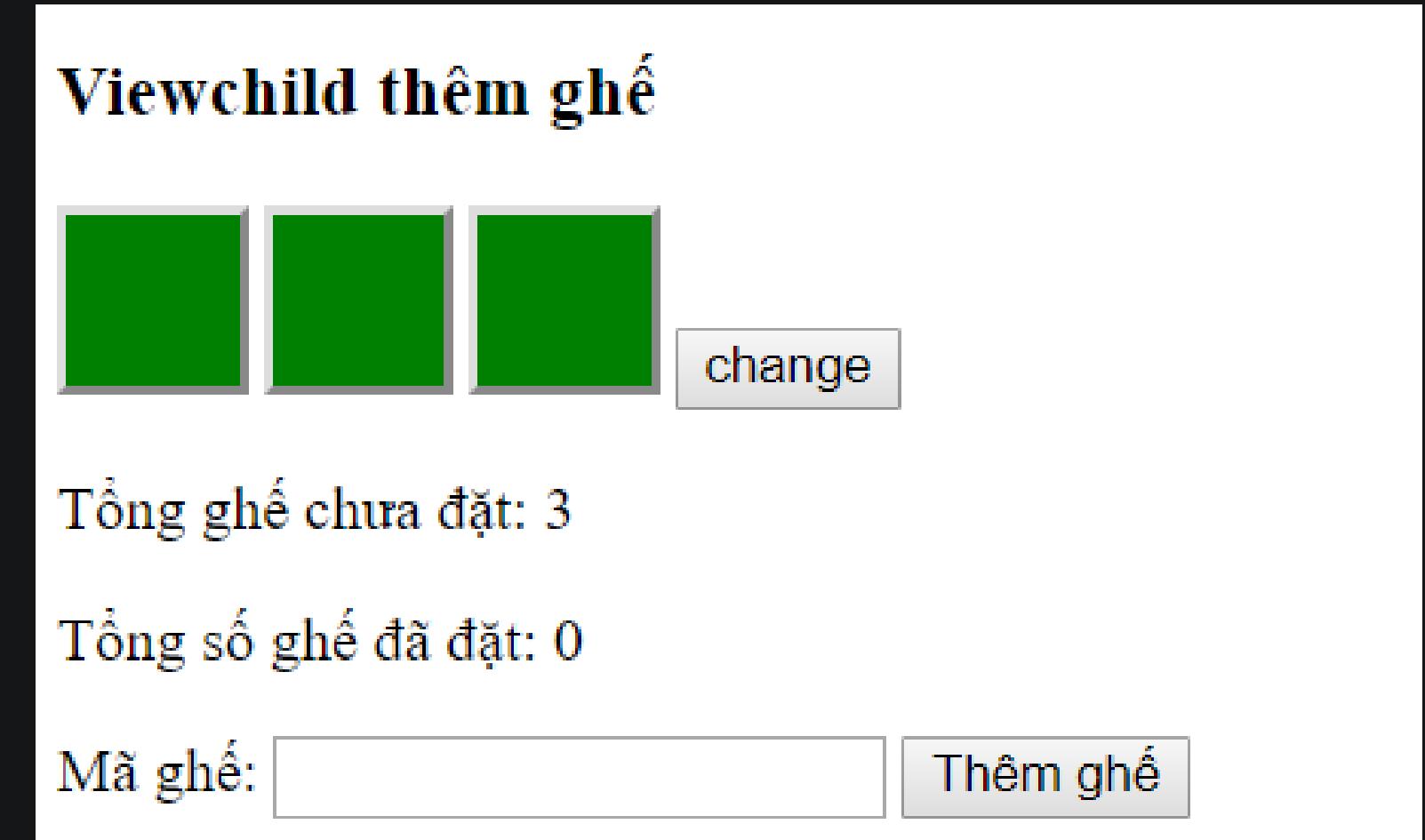
Component con (DanhSachGheNgoiComponent)

```

export class DanhsachghengoiComponent implements OnInit {
    public DanhSachGhe:Array<any> = [{maghe:"ghe1",trangthai:false},{maghe:"ghe2",trangthai:false},{maghe:"ghe3",trangthai:false}];
    public SoGheDaDat:number = 0;
    public SoGheChuaDat:number = this.DanhSachGhe.length;
    Dat_Ghe_Parent(trangthaighe:boolean,maghe) //Biến được gửi từ component con sang
    {
        alert(maghe);
        if(trangthaighe == true)
        {
            this.SoGheDaDat++;
            this.SoGheChuaDat--;
        }
        else
        {
            this.SoGheDaDat--;
            this.SoGheChuaDat++;
        }
    }
    themdsdge(maghe:string,trangthai:boolean)
    {
        var ob = {maghe:maghe,trangthai:trangthai};
        this.DanhSachGhe.push(ob);
    }
}

```

Viewchild thêm ghế



Tổng ghế chưa đặt: 3

Tổng số ghế đã đặt: 0

Mã ghế: Thêm ghế

Tạo thêm 1 textbox và 1 button
thêm ghế
+ Hàm xử lý sự kiện

View Child (DOM đến component)

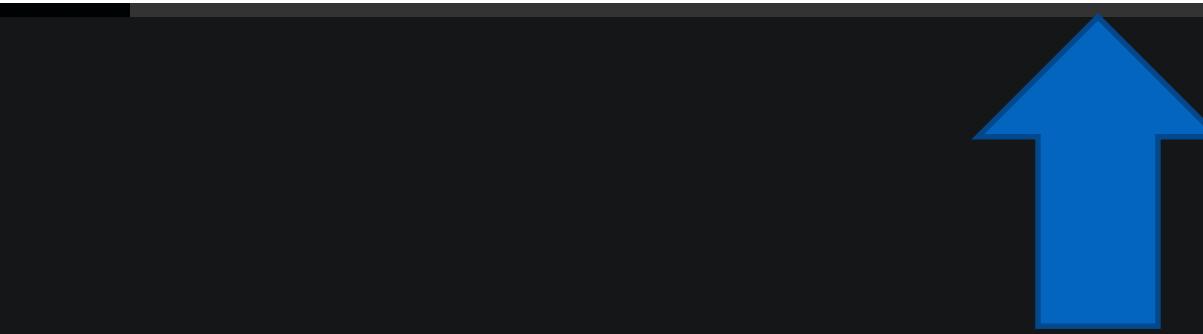
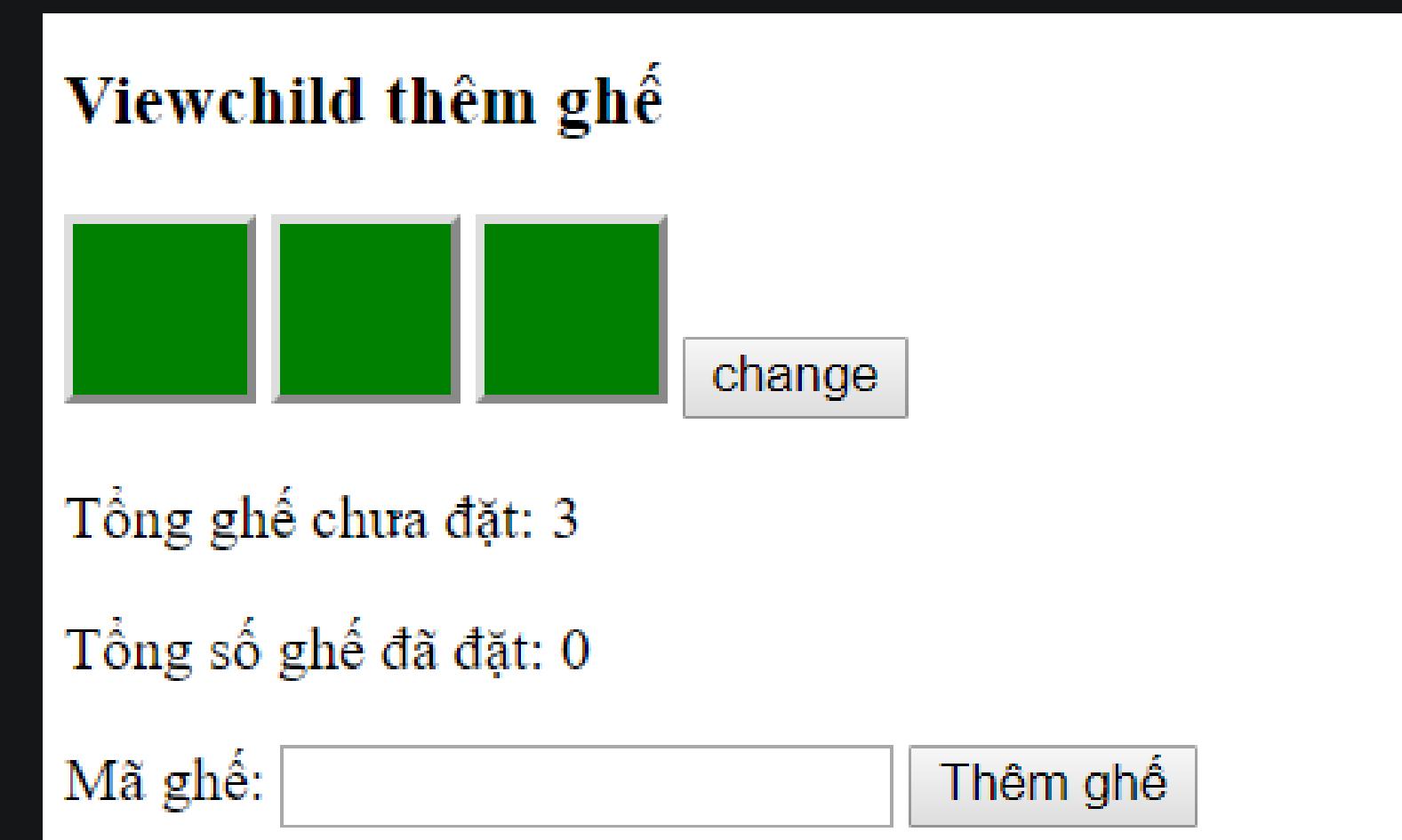
Component cha (QuanLyDanhSachGheNgoiComponent)

```
ts interactivecomponent.module.ts      ts quanlydanhsachghengoi.component.ts x  ts danhsachghengoi.com
1 import { Component, OnInit, ViewChild } from '@angular/core';
2 import { DanhsachghengoiComponent } from '../danhsachghengoi/danhsachghengoi.component';
3 @Component({
4   selector: 'quanly-danhsachghe',
5   template:
6     <h3>Viewchild thêm ghế</h3>
7     <danh-sach-ghe-ngozi></danh-sach-ghe-ngozi> ← component con
8     Mã ghế: <input #maghe>
9     <button (click)="themghe(maghe.value)">Thêm ghế</button>
10    ,
11    styleUrls: ['./quanlydanhsachghengoi.component.css']
12  )
13 export class QuanlydanhsachghengoiComponent implements OnInit {
14
15   constructor() { }
16
17   @ViewChild (DanhsachghengoiComponent) dsgheCom:DanhsachghengoiComponent;
18   themghe(maghe:string)
19   {
20     console.log(this.dsgheCom);
21     this.dsgheCom.themdsghe(maghe,true);
22   }
23   ngOnInit() { ↑
24   }
25
26 }
```

Tại đây thông qua **viewchild** để điều hướng đến **component con** gọi phương thức **themghe**(phương thức của component con)

Kiểu dữ liệu component con

Giao diện của component con khi được component cha gọi



Phương thức thêm ghế tại component cha làm nhiệm vụ gọi đến phương thức them ghế vào component con

☐ View Child (DOM đến tag html)

Ví Dụ 1: Dùng ViewChild Dom đến 1 thẻ thông thường có đánh #name

Import vào 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 /> |
```

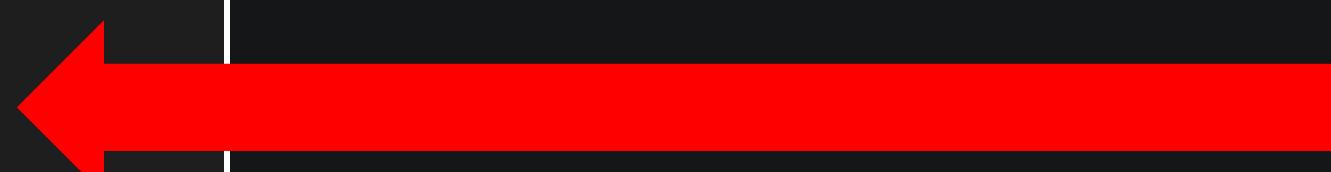
Khai báo kiểu dữ liệu cho thuộc tính DOM từ #name là ElementRef

```
@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



□ View Children (DOM danh sách các selector)

Ví Dụ 1: Dùng ViewChildren dom đến danh sách các đối tượng component

Cũng tương tự như các tag html tag component cũng có các thuộc tính và phương thức. Để dom đến các thuộc tính và phương thức của 1 tập các thẻ component ta dùng đối tượng Viewchildren để thao tác.

```
<app-item [item]="sp" *ngFor="let sp of mangDienThoai"></app-item>
```

Thể hiện của các đối tượng component dưới dạng tag html

```
//Dom đến danh sách các thẻ app-item => 1 list đối tượng ItemComponent
@ViewChildren(ItemComponent) listTagAppItem: QueryList<ItemComponent>;
    Tên lớp đối tượng của component đó

ThemSP(MaSP,TenSP,MoTa){
    let sp = {MaSP:MaSP,TenSP:TenSP,MoTa:MoTa};
    //Dom đến thuộc tính DSSP của thẻ <app-list-item>
    //Push sp vào thuộc tính DSSP
    // this.tagAppListItem.DSSP.push(sp);

    for(let i = 0; i<this.listTagAppItem.length;i++)
    {
        console.log(this.listTagAppItem.toArray()[i]);
    }
    Phương thức để duyệt component (Vì kết quả trả về là 1 danh sách nên ta sẽ duyệt tuần tự danh sách từng đối tượng (Thẻ) một qua phương thức .forEach hoặc map
    this.listTagAppItem.forEach((appItem:ItemComponent)=>{
        console.log(appItem);
    })
}
```

Thao tác dom bằng ViewChildren => Trả về 1 danh sách (Có thể hiểu như 1 mảng) các đối tượng là các thẻ component với các thuộc tính và phương thức tương ứng như hình bên phải

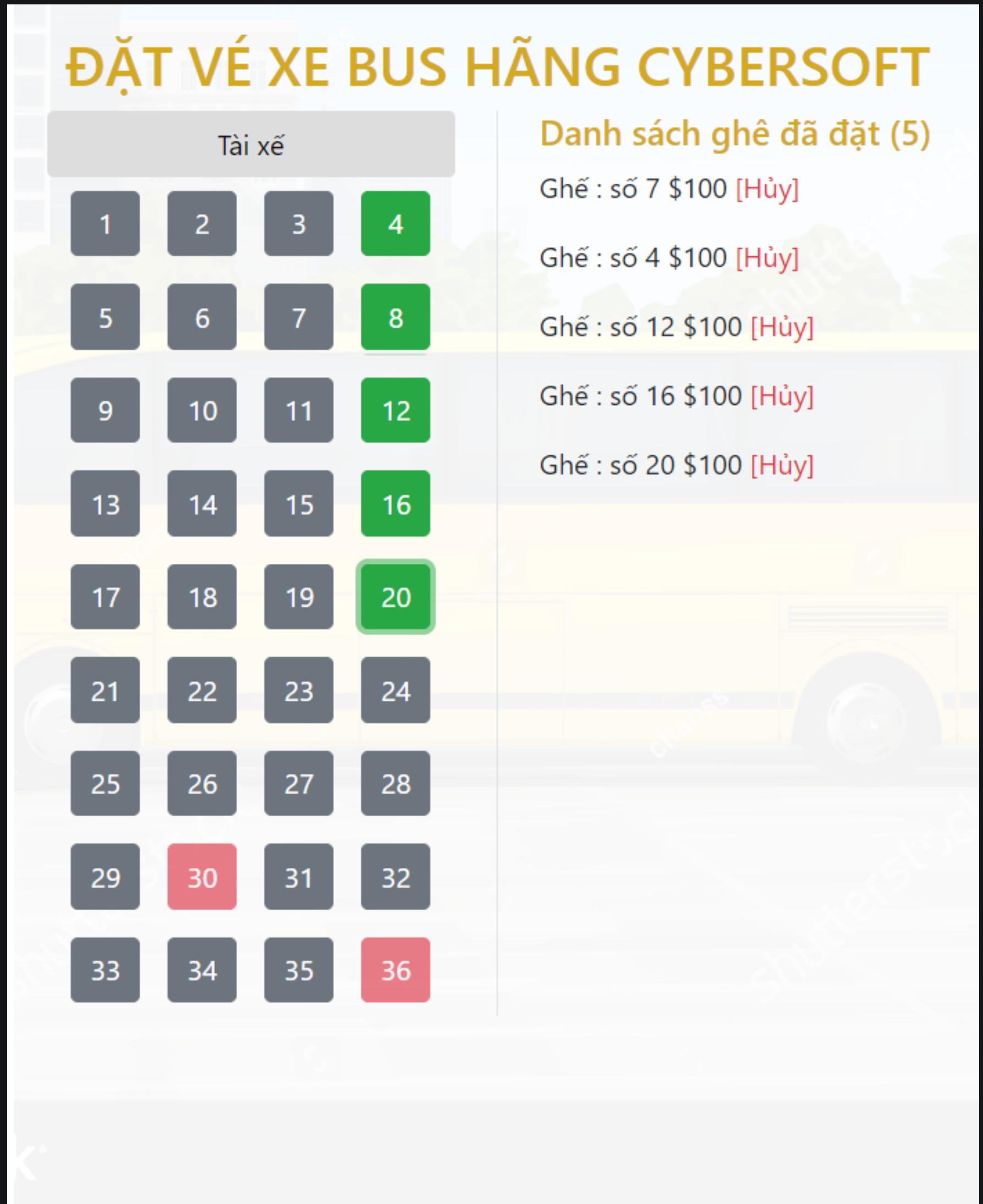
TS item.component.ts ×

```
1 import { Component, OnInit, Input, Output,EventEmitter } from '@angular/core';
2
3 @Component({
4     selector: 'app-item',
5     templateUrl: './item.component.html',
6     styleUrls: ['./item.component.css']
7 })
8 export class ItemComponent implements OnInit {
9
10     @Input() item: any = {};
11     @Output() eventChonSanPham = new EventEmitter();
12     constructor() { }
13
14     ngOnInit() {
15     }
16     ChonSanPham(){
17         //Đưa giá trị item ra ngoài thông qua eventChonSanPham
18         this.eventChonSanPham.emit(this.item);
19     }
20     XemChiTiet(){
21     }
22     {
23         console.log("Xem chi tiết:" + this.item);
24     }
}
```

Lớp đối tượng của component

Tương tự như các <tag> html [component] cũng có các thuộc tính và phương thức

Bài tập



✓ Yêu cầu cơ bản

- **Phân tích component** dựa vào hình bên trái.
- **Cho mảng các ghế phía dưới (có thể copy)**. Yêu cầu hiện thực các ghế trên xe buýt bằng control button. (Lưu ý không cần phải load đẹp như hình 1 hàng 4 ghế là được).
- Phân tích lớp đối tượng và các phương thức xử lý. (Tạo folder model trong thư mục app tạo folder model tạo các class Object tương ứng)
- 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},
```

7. Hướng dẫn cài đặt primeNG

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

Bước 1: Cài đặt thư viện `npm install primeng --save`

Bước 2: Cài đặt thư viện `npm install primeicons --save`

Bước 3: Cài đặt thư viện animation `npm install @angular/animations --save`

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

```
projectPrimeNG › src › app › shared › Modules › prime-ng › 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 { }
15
```

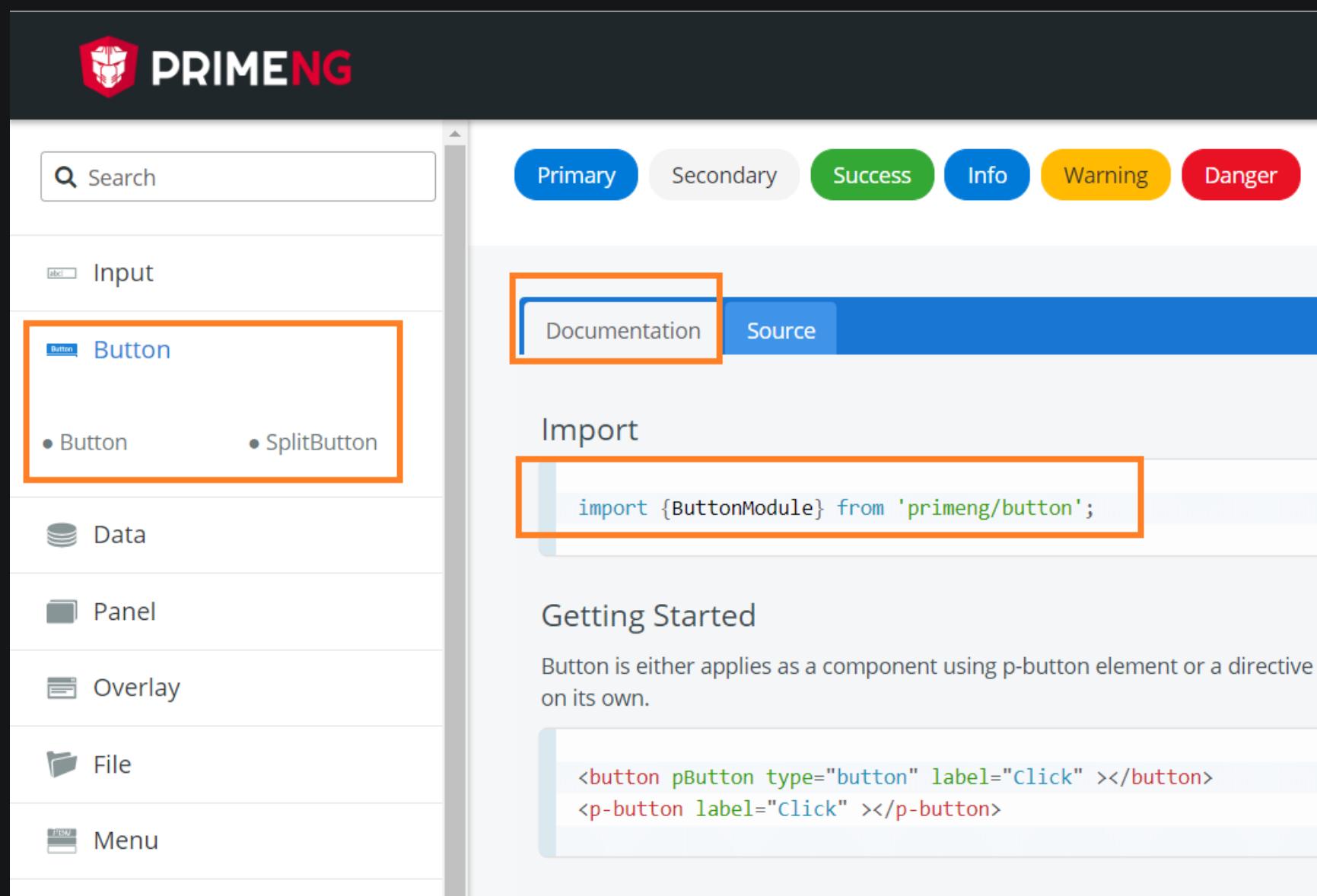
Bước 5: Đóng gói các thư viện css của primeNG (thuộc tính styles của file angular.json)

```
{} angular.json x

projectPrimeNG > {} angular.json > {} projects > {} projectPrimeNG > {} architect > {} build > {} options > [ ]assets
  20   "esconfig": "src/esconfig.app.json",
  21   "assets": [
  22     "src/favicon.ico",
  23     "src/assets"
  24   ],
  25   "styles": [
  26     "src/styles.css",
  27     "node_modules/primeicons/primeicons.css",
  28     "node_modules/primeng/resources/themes/nova-light/theme.css",
  29     "node_modules/primeng/resources/primeng.min.css"
  30   ],
  31   "scripts": []
  32 },
  33   "configurations": {
  34     "production": {
  35       "fileReplacements": [
  36         {
  37           "replace": "src/environments/environment.ts",
  38           "with": "src/environments/environment.prod.ts"
  39         }
  40       ]
  41     }
  42   }
}
```

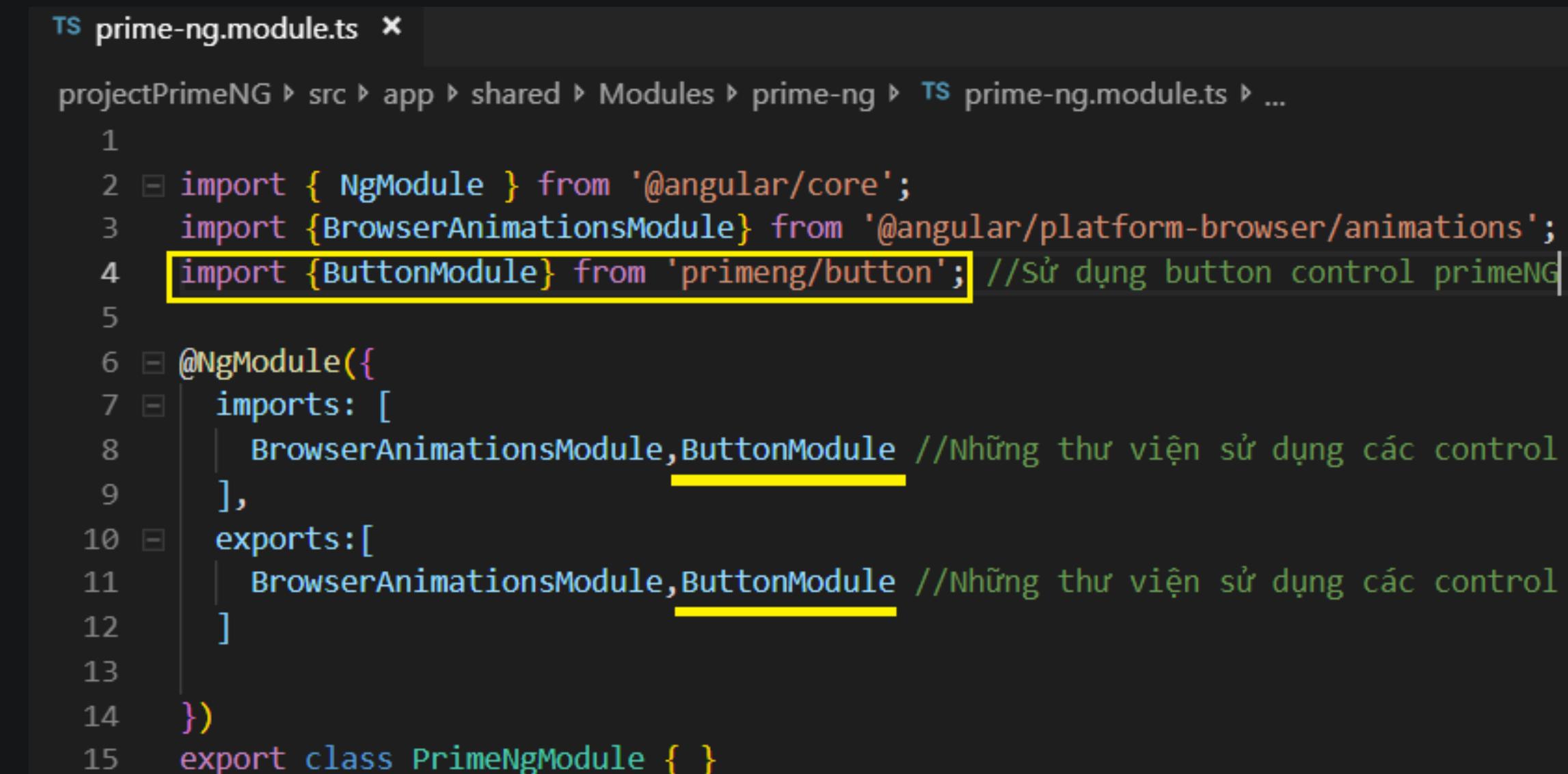
Bước 6: Cách sử dụng

Để sử dụng 1 số control primeNG tương tự như material ta vào phần control phía bên phải trên trang primeNG xem cần import thư viện nào thì ta import thư viện đó vào module PrimeNG



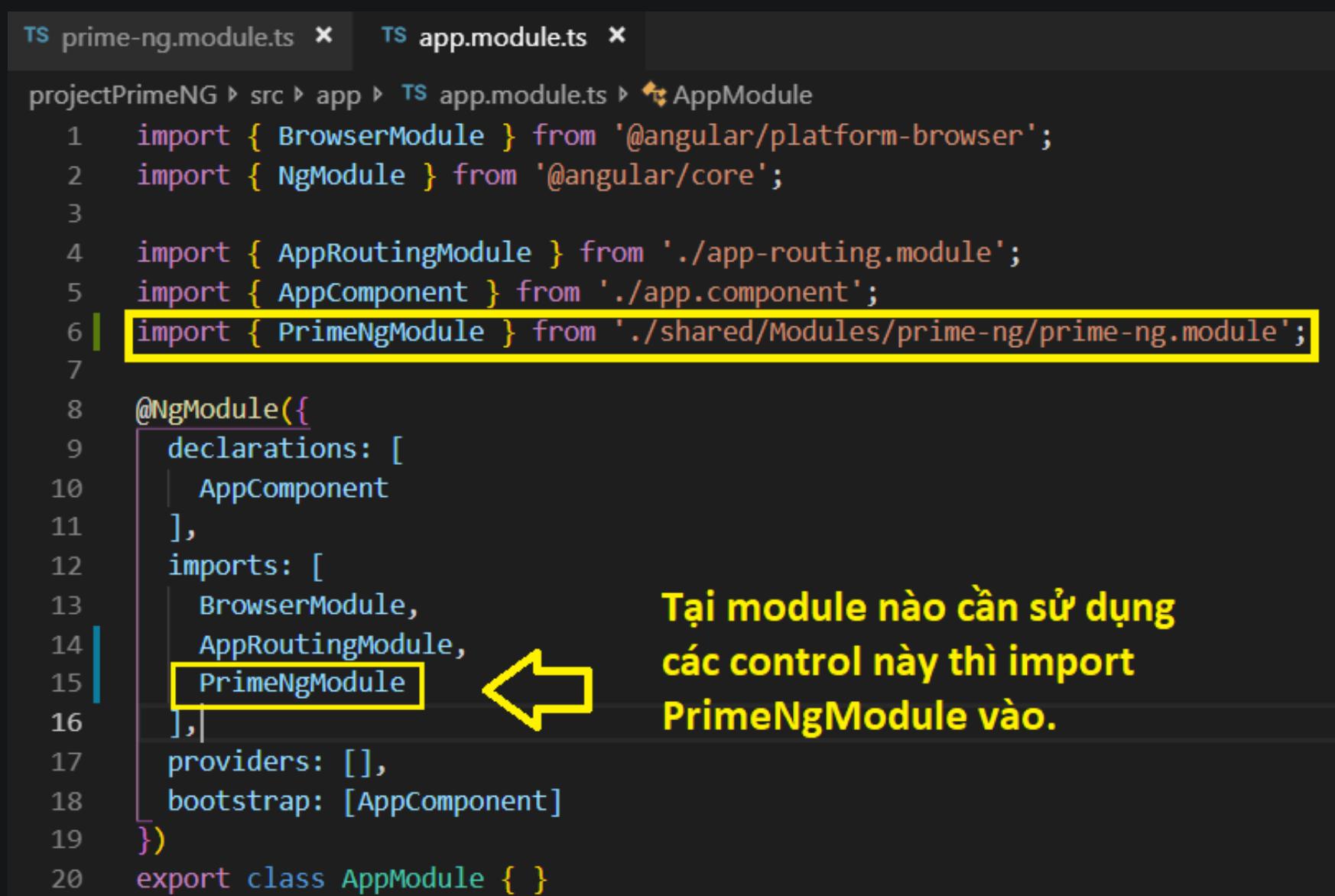
The screenshot shows the PrimeNG documentation for the 'Button' component. The 'Documentation' tab is selected. Below it, the 'Import' section contains the code: `import {ButtonModule} from 'primeng/button';`. The sidebar on the left lists other components like Input, Data, Panel, Overlay, File, and Menu.

Bước 1



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

Bước 2



```
TS prime-ng.module.ts x  TS app.module.ts x
projectPrimeNG > src > app > TS app.module.ts > AppModule
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
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
```

A callout arrow points from the text 'Tại module nào cần sử dụng các control này thì import PrimeNgModule vào.' to the line `PrimeNgModule` in the imports array of the `AppModule`.

Bước 3

Bước 4: Dùng tên directive sử dụng như trên document (Lưu ý phần document có phần properties của directive này dựa vào đó có thể tùy chỉnh)

```
<p-button label="Click" icon="pi pi-check" (onClick)="Print()"></p-button>
```

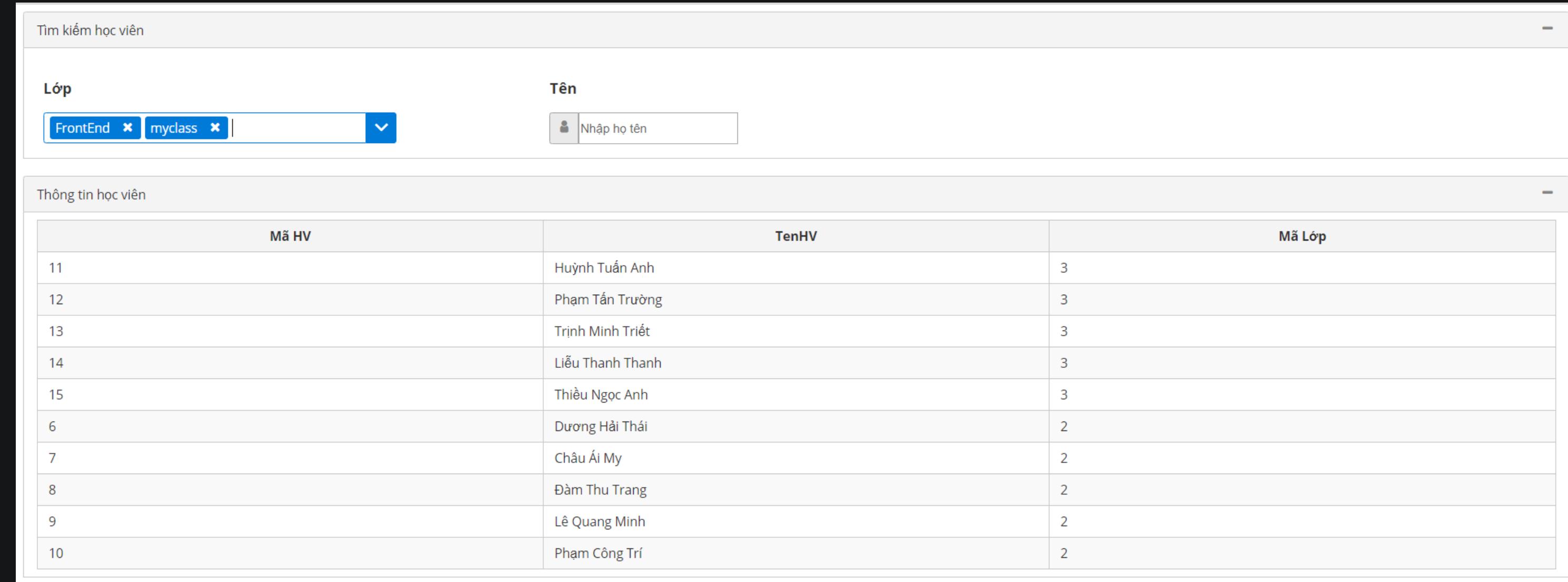
Tại module nào cần sử dụng các control này thì import PrimeNgModule vào.

Bài tập

Cho mảng dữ liệu

```
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 My", 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 Tấn 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 Phương Châu", MaLop:4},
]
```

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



The screenshot shows a user interface for managing student data. At the top, there is a search bar labeled 'Tìm kiếm học viên' and two dropdown menus: 'Lớp' containing 'FrontEnd' and 'myclass', and 'Tên' with a placeholder 'Nhập họ tên'. Below these are two tables. The first table, 'Thông tin học viên', lists student IDs (MaHV) from 11 to 10, their names (TenHV), and their class numbers (Mã Lớp). The second table, 'Lớp', lists class names (TenLop) and their corresponding student counts (Mã Lớp).

Mã HV	TenHV	Mã Lớp
11	Huỳnh Tuấn Anh	3
12	Phạm Tấn 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 My	2
8	Đàm Thu Trang	2
9	Lê Quang Minh	2
10	Phạm Công Trí	2

TenLop	Mã Lớp
FrontEnd	1
myclass	2

- **Yêu cầu:**
 - Dùng primeNG control thiết kế giao diện và chức năng như hình.
 - Sử dụng PrimeNG Button thiết kế cho các nút button
 - Sử dụng PrimeNG AutoComplete để thiết kế cho filter search
 - Sử dụng PrimeTable và PrimePanel để thiết kế view hiển thị dữ liệu

7. Hướng dẫn cài đặt material angular

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

Bước 1: Cài đặt thư viện npm i --save @angular/material @angular/cdk @angular/animations

Bước 2: Cài đặt thư viện ng add @angular/material

Bước 3: Tạo một module với tên material.module.ts (Module chứa thư viện animation)

```
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 { }
```

TS 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 { }

```

Import module Material vào module cần sử dụng control material

Web tham khảo cách sử dụng các control

Material Components CDK Guides

Tab này dùng để xem module cần import vào material.module.ts
để sử dụng được control material này

Checkbox

OVERVIEW API EXAMPLES

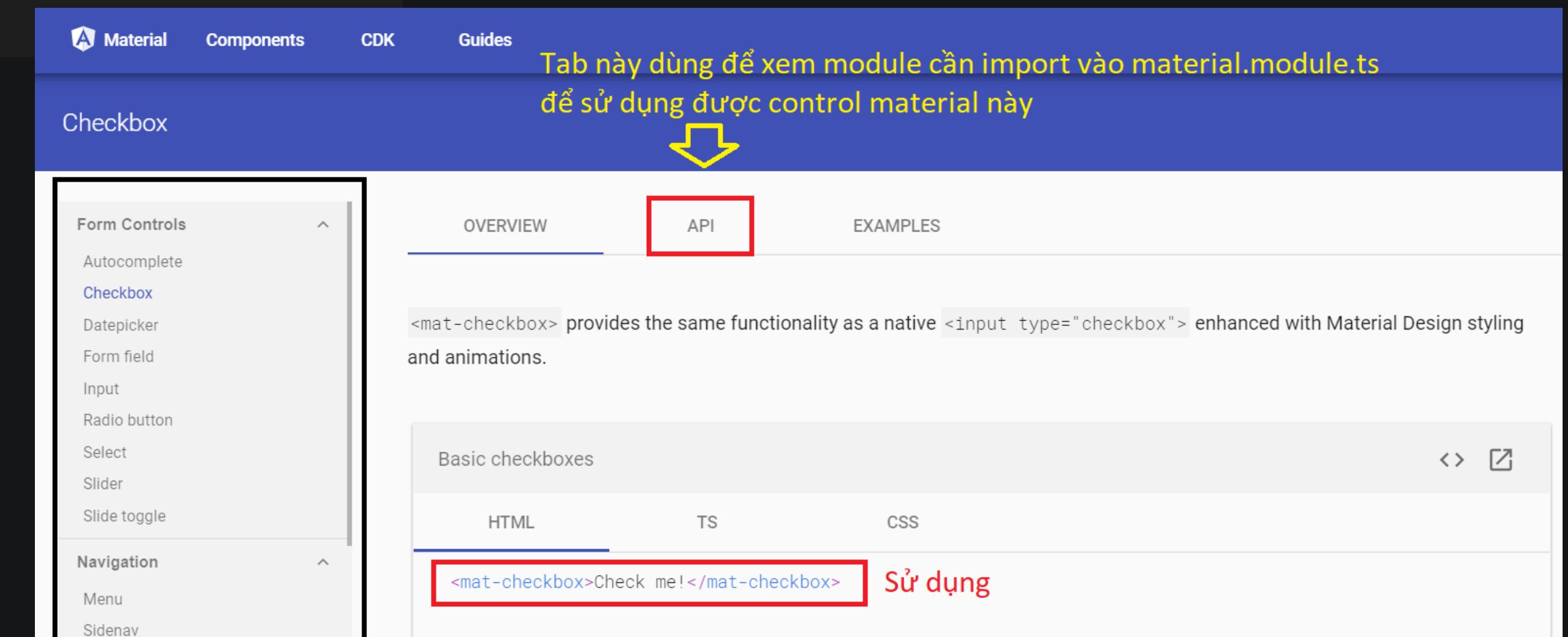
<mat-checkbox> provides the same functionality as a native <input type="checkbox"> enhanced with Material Design styling and animations.

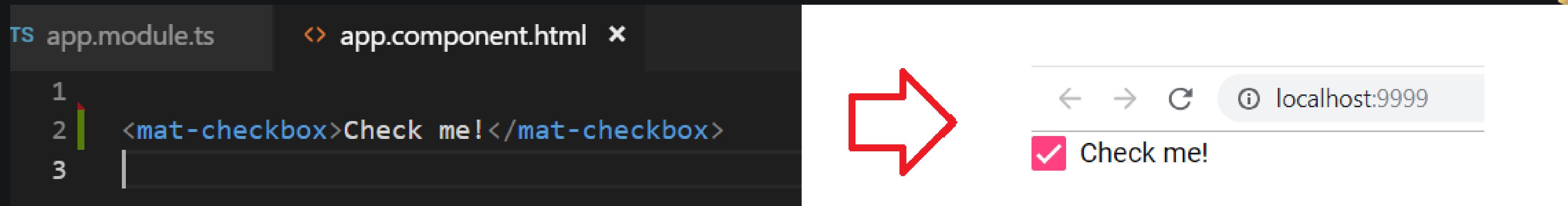
Basic checkboxes

HTML TS CSS

<mat-checkbox>Check me!</mat-checkbox>

Sử dụng





TS app.module.ts <> app.component.html x

```
1 <mat-checkbox>Check me!</mat-checkbox>
2
3
```

localhost:9999

Check me!

Sử dụng control checkbox

```
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 )
```

Đối với một số control có

8. Template - Routing

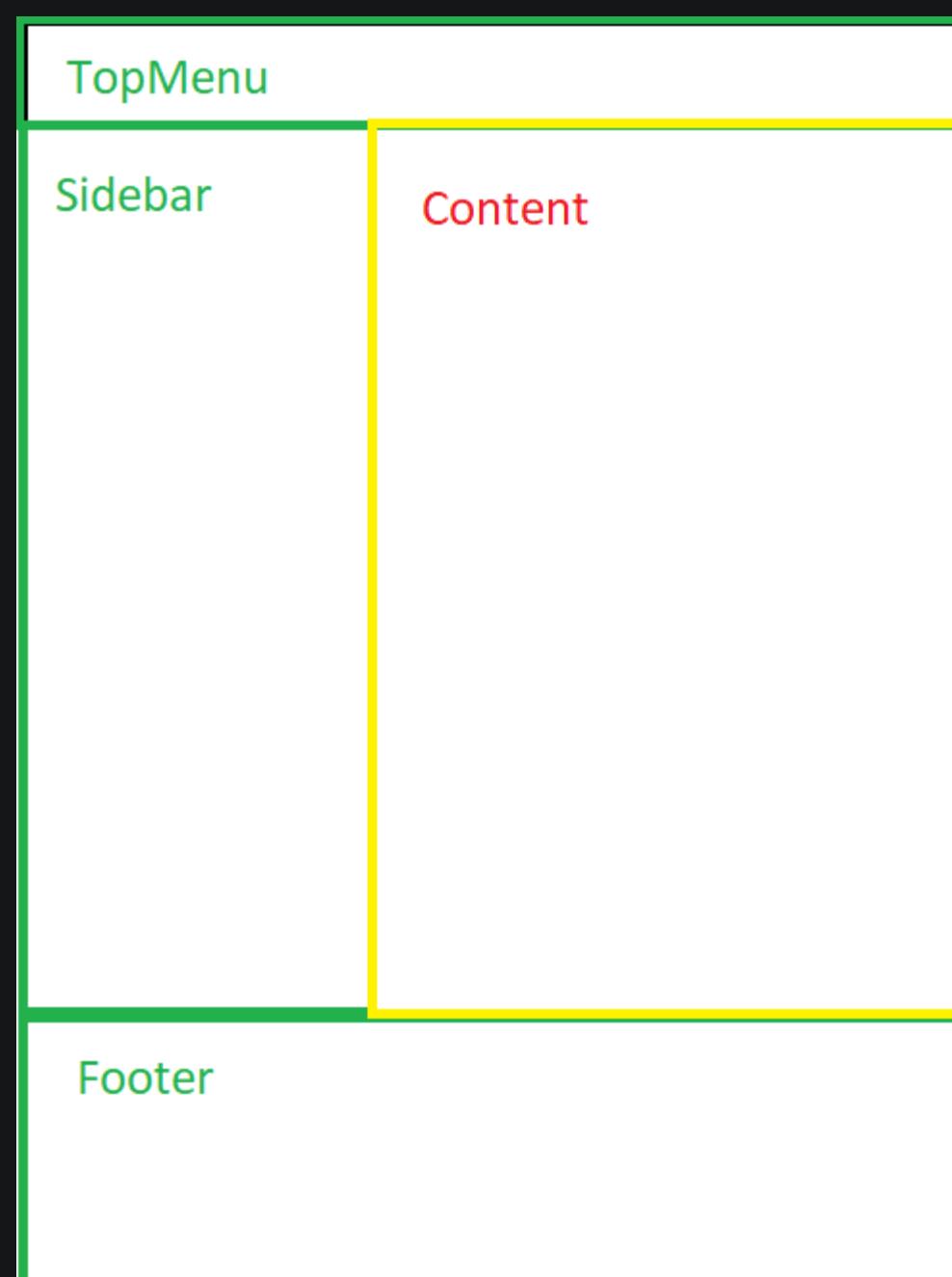
Template là gì ?

Template là những mẫu layout dàn trang, bố cục được thiết kế sẵn, khi sử dụng chỉ việc chỉnh sửa lại phần nội dung cho phù hợp giúp người dùng tiết kiệm được thời gian chi phí.

Thay vì 1 web site ta phải xây dựng layout cho các trang từ đầu đến cuối thì ta chỉ cần dụng 1 template cho các trang khác kế thừa phần nội dung.

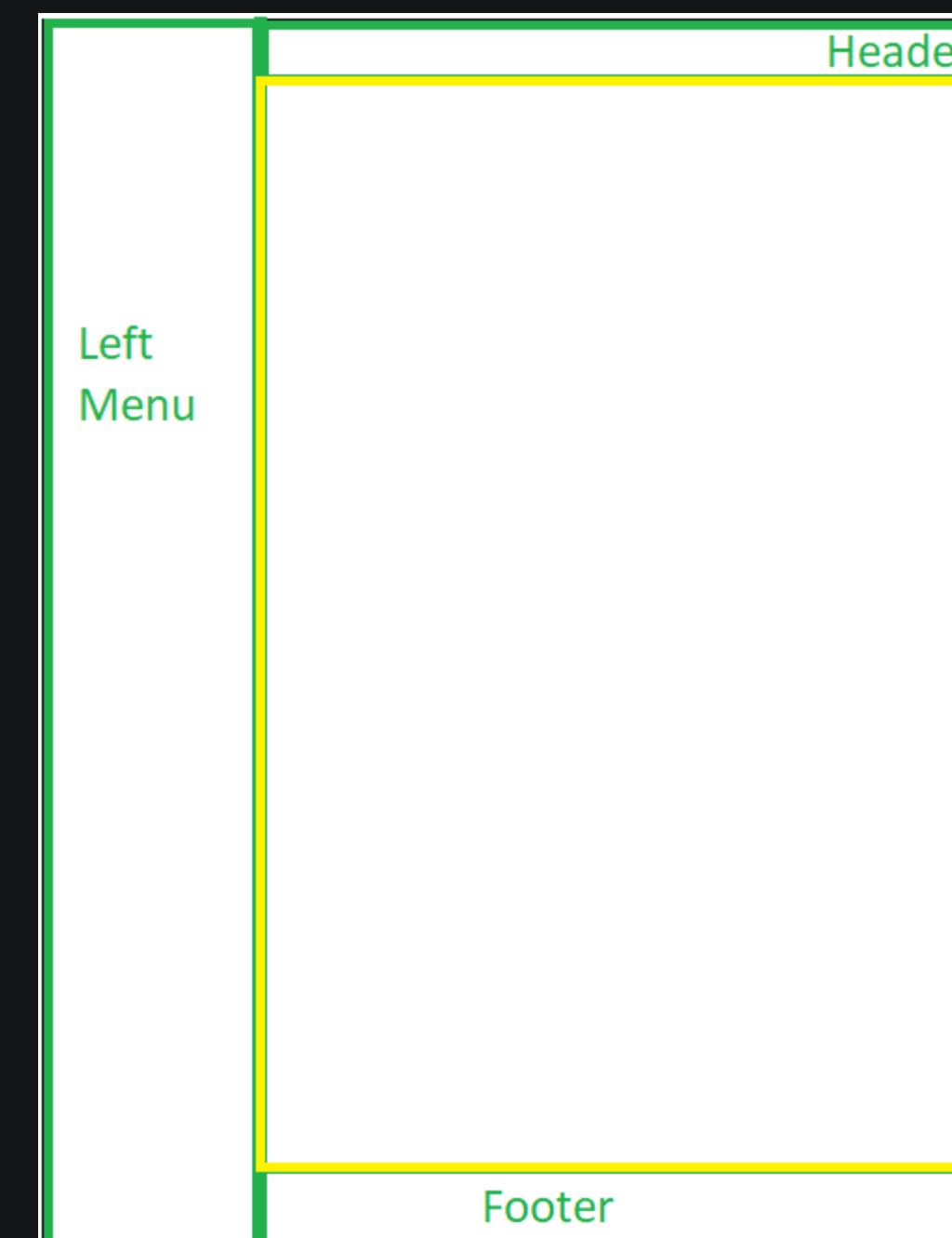
Một ứng dụng web có thể có nhiều template (Home, Admin, User).

Ví dụ:



Template dành cho giao diện chính (Template Home).

Các thành phần như TopMenu, Sidebar, Footer được áp dụng cho tất cả các trang sử dụng template này (Trang chủ, trang chi tiết phim, trang, đăng ký, đăng nhập ...)



Template dành cho giao diện quản trị (Template admin).

Các thành phần như Header, LeftMenu, Footer được áp dụng cho tất cả các trang sử dụng template này ví dụ như (Quản lý người dùng, Quản lý danh sách phim, Quản lý ...).

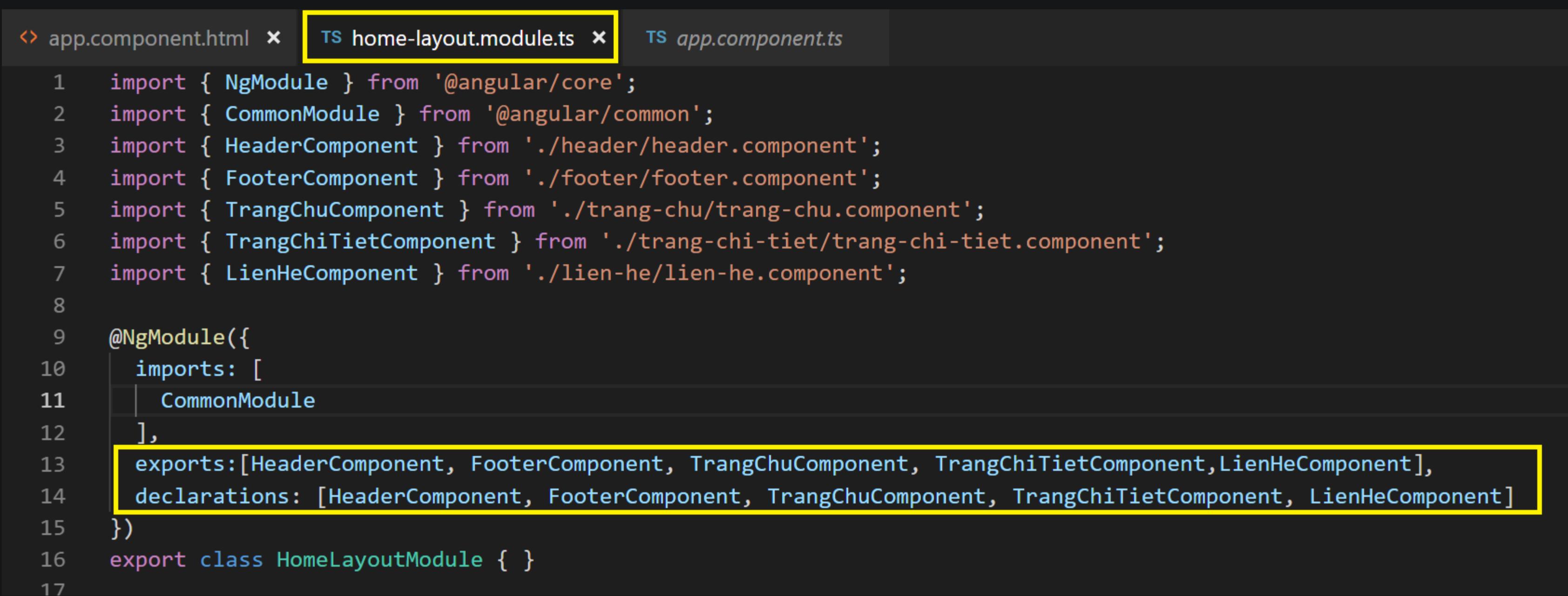
Routing là gì?

Routing là cơ chế quản lý các component, module dựa trên các url được định nghĩa sẵn đối với thành phần template. Nói 1 cách đơn giản mỗi routes quản lý một template và mỗi template được xác định bởi một module – hoặc component.

Có nhiều cách để chia 1 ứng dụng thành nhiều routes.

Cách 1: Chia routes theo component

Ta lần lượt có các component là: Header, Footer, TrangChu, TrangChiTiet, LienHe



The screenshot shows a code editor with three tabs: 'app.component.html', 'home-layout.module.ts', and 'app.component.ts'. The 'home-layout.module.ts' tab is active and highlighted with a yellow box. The code in the file is:

```
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 { }
17
```

The 'exports' and 'declarations' sections of the module are highlighted with a yellow box.

Cách 1: Chia routes theo component

Nội dung của app.module.ts

```
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) // Gắn 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()**

Cách 1: Chia routes theo component

Nội dung của app.component.ts

```
app.component.html x
1 <app-header></app-header>
2 <router-outlet></router-outlet> ← Phản nội dung sẽ được
3 <app-footer></app-footer> thay thế khi ta thay đổi
4 url của website tương
5 ứng với routes
```

Nội dung của **HomeLayoutModule.ts** – Vì chứa component header
(có chứa các thẻ a có routerLink nên tại Module này phải import
RouterModule để hiểu cú pháp routerLink trong thẻ a)

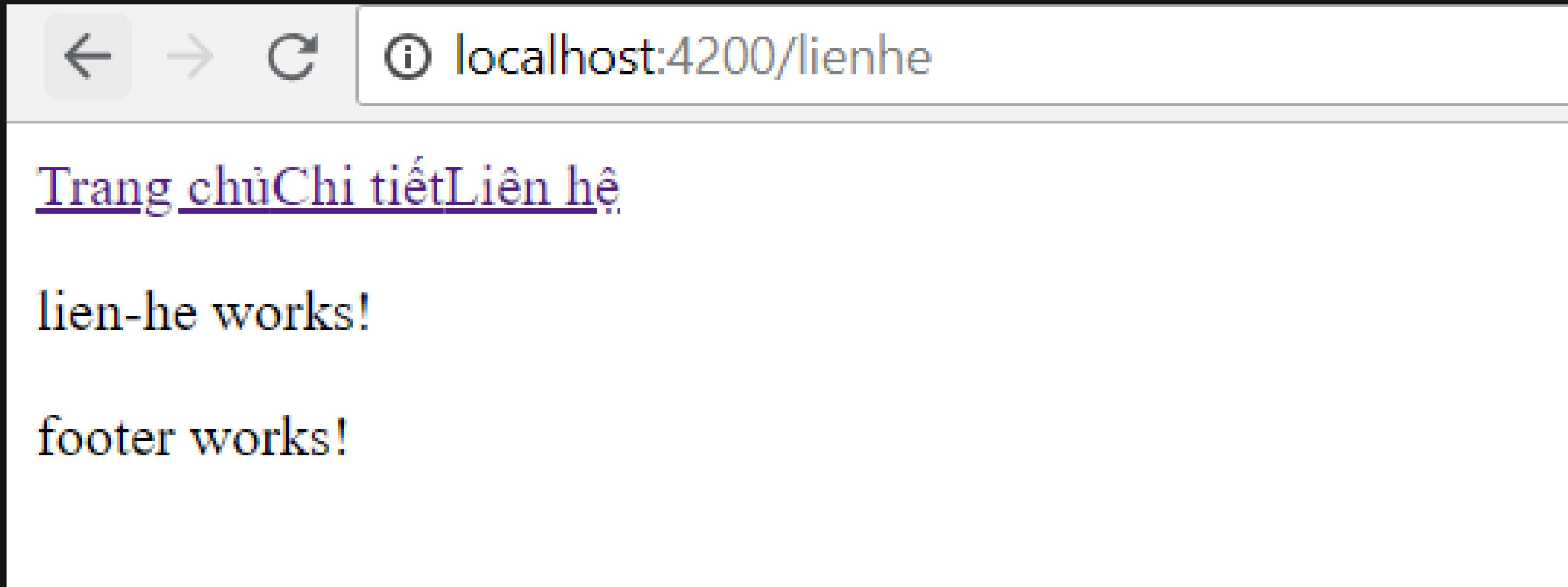
```
home-layout.module.ts x
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { HeaderComponent } from './header/header.component';
4 import { FooterComponent } from './footer/footer.component';
5 import { TrangChuComponent } from './trang-chu/trang-chu.component';
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 { }
```

Các đường link giúp người dùng
thao tác qua lại giữa các trang

```
header.component.html x
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

Cách 1: Chia routes theo component

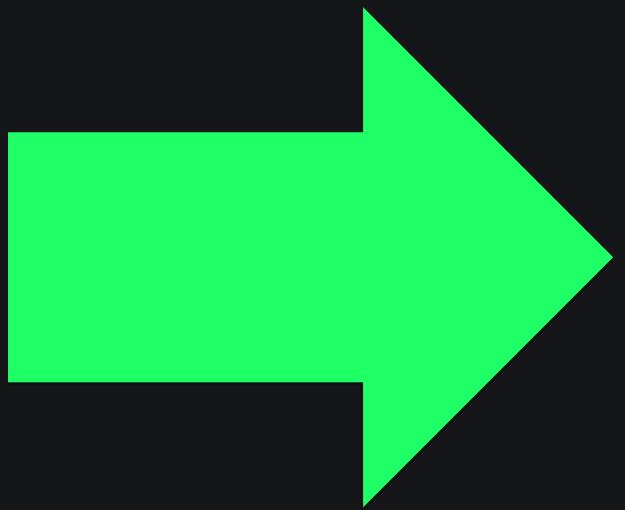


Kết quả: khi người dùng click vào link nào thì chỉ duy nhất phần tại nơi chứa thẻ router-outlet bị thay đổi mọi thành phần khác được giữ nguyên, vẫn phát sinh link url được nhưng không phải load lại trang -> Điều này khắc phục được việc giúp cho google index website thông qua link url đồng thời cải thiện tốc độ load trang của ứng dụng.

Cách 2: Chia routes theo module và template

Tuy nhiên khi việc phát triển 1 ứng dụng khá lớn với nhiều module cũng như component thì việc quản lý các routes không hề đơn giản. 1 app.module phải khai báo rất nhiều module hoặc component để định tuyến các đường truyền (url).

Giải pháp



Chia nhỏ các routes => quản lý routes theo từng module. Các module được chia sẻ được nhóm theo chức năng

Cách 2: Chia routes theo module và template

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

Ví dụ hệ thống ta có 2 phân hệ module chính là HomeModule và AdminModule

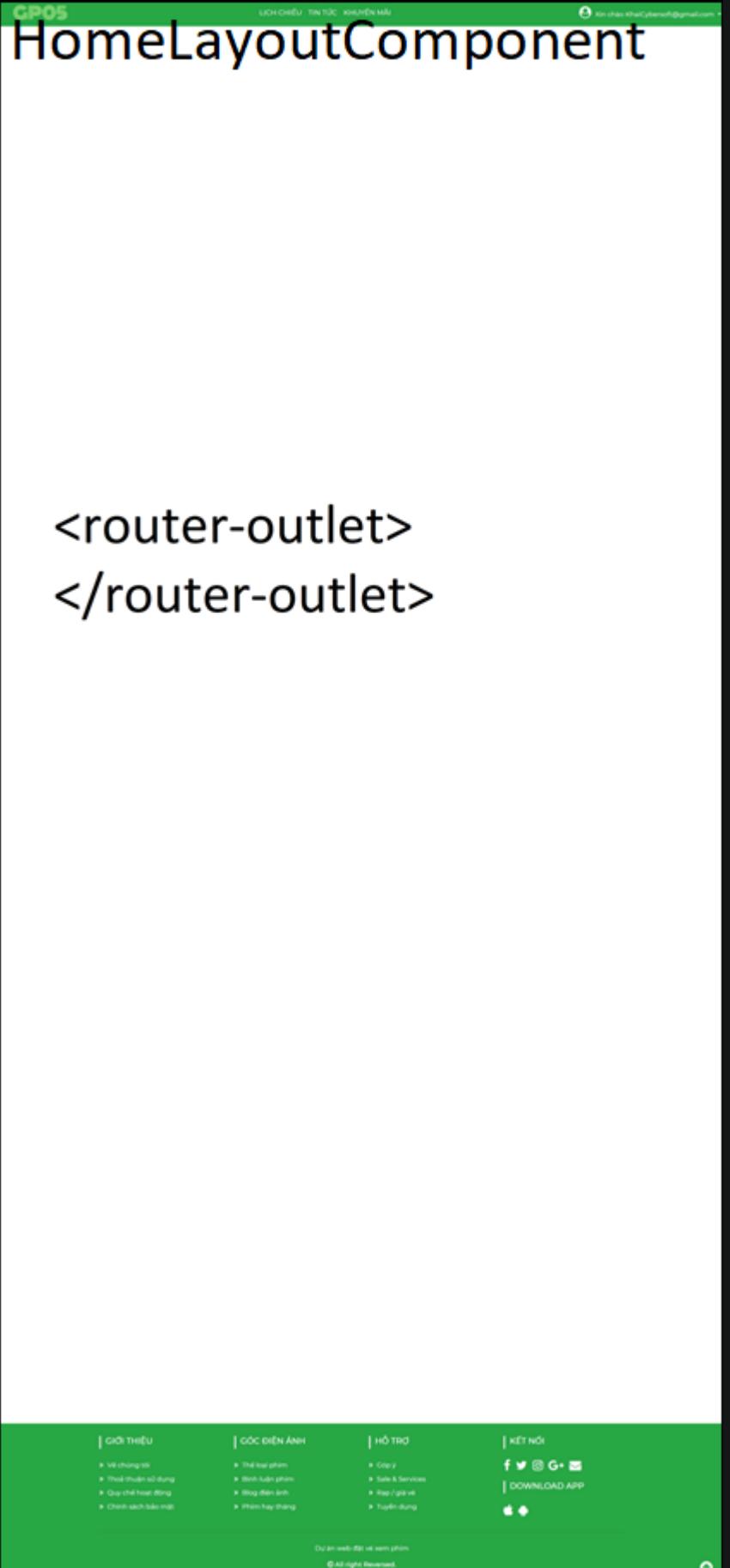
Trong HomeModule chứa các trang: TrangChu, TrangChiTiet, TrangDangNhap ...

Trong Adminmodule chứa các trang: QuanLyPhim, QuanLyNguoiDung ...

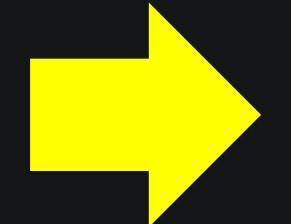
=> Và mỗi module đó ta sẽ tổ chức mỗi template khác nhau => Vì vậy ta không thể làm như các 1 được vì cách 1 chúng ta chỉ có duy nhất 1 phần <router-outlet> để định nghĩa (1 template duy nhất ở đây ta cần 2 template), chúng ta đã cố định header và footer nên 2 component này sẽ có mặt ở tất cả các link

□ Home template (Home module)

HomeModule (home template)



<router-outlet>
</router-outlet>



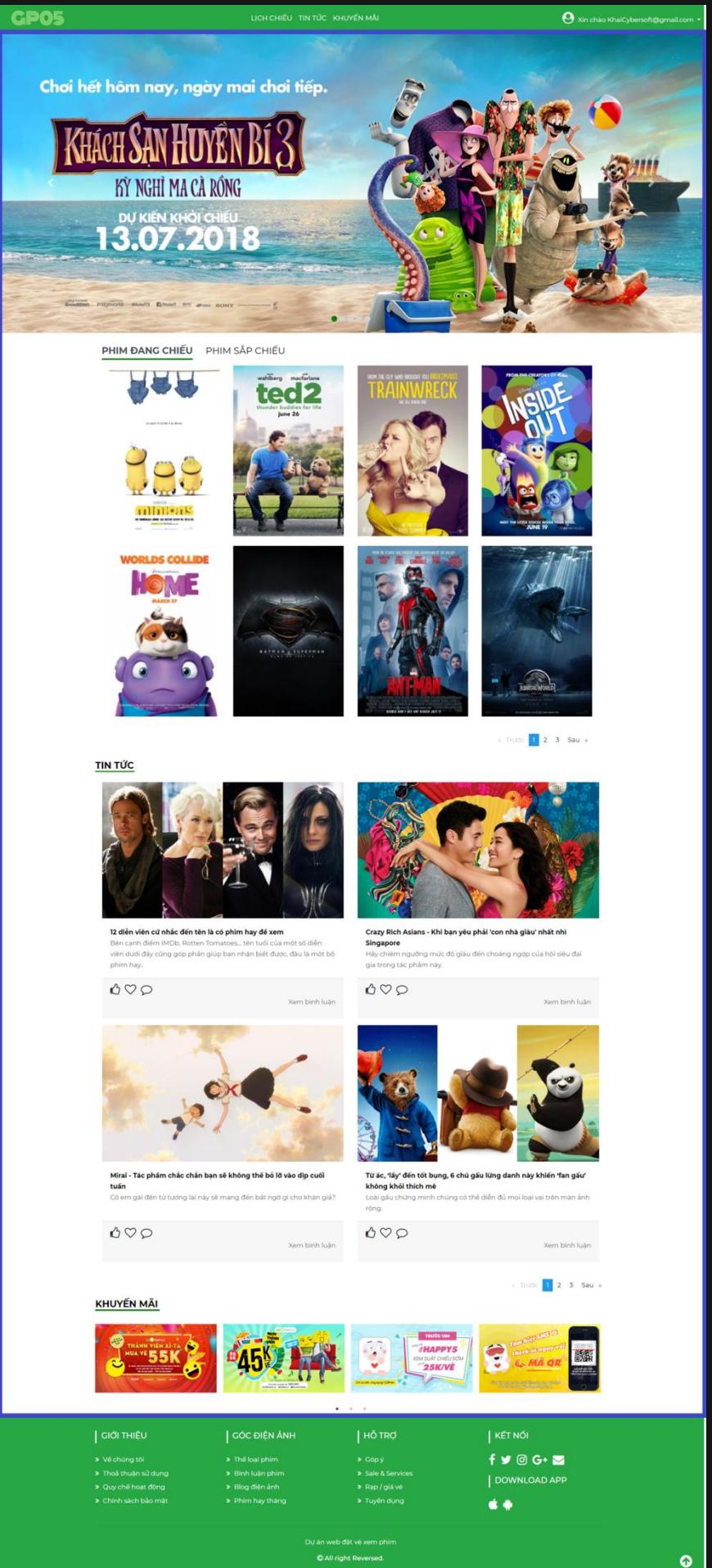
Kế thừa từ router-outlet
Home template

Link: localhost:4200/home

hoặc

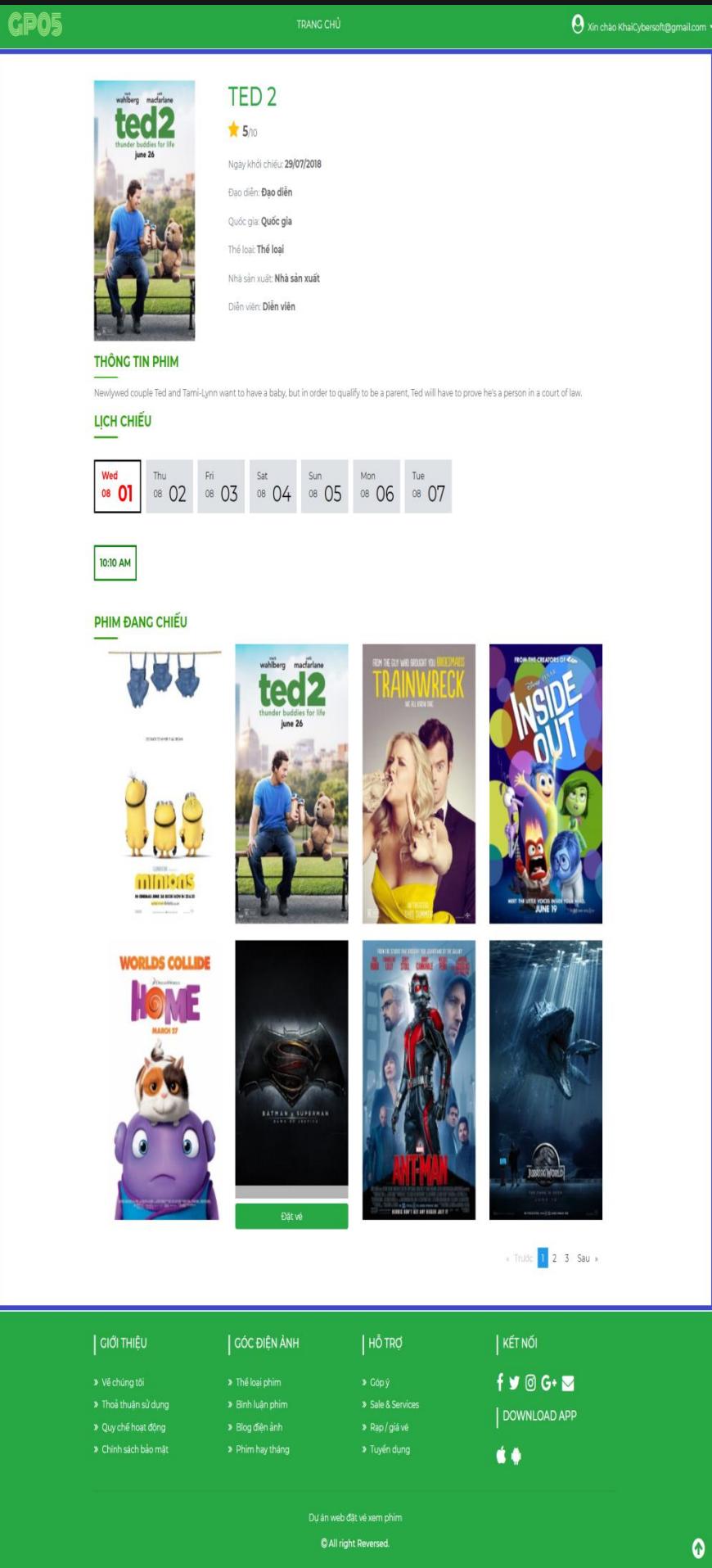
Link: localhost:4200 (thường thì
người ta sẽ có 1 path bỏ trống
để vào module mặc định)

Trangchu: kế thừa từ home template
định nghĩa lại phần khung màu tím

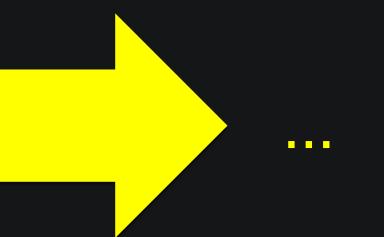


Link: localhost:4200/home/trangchu
hoặc
Link: localhost:4200/trangchu

TrangChiTiet: cũng được kế thừa từ
home template



Link: localhost:4200/home/trangchitiet/1
hoặc
Link: localhost:4200/trangchitiet/1



Cấu hình thực tế (HomeModule -> Template Home)

```
ts home.module.ts      ts app.module.ts x   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 tương 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) // Gắn đổi tương appRoutes (khai báo các đường dẫn) vào đổi tương Router (vì là app nêu
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

home.module.ts

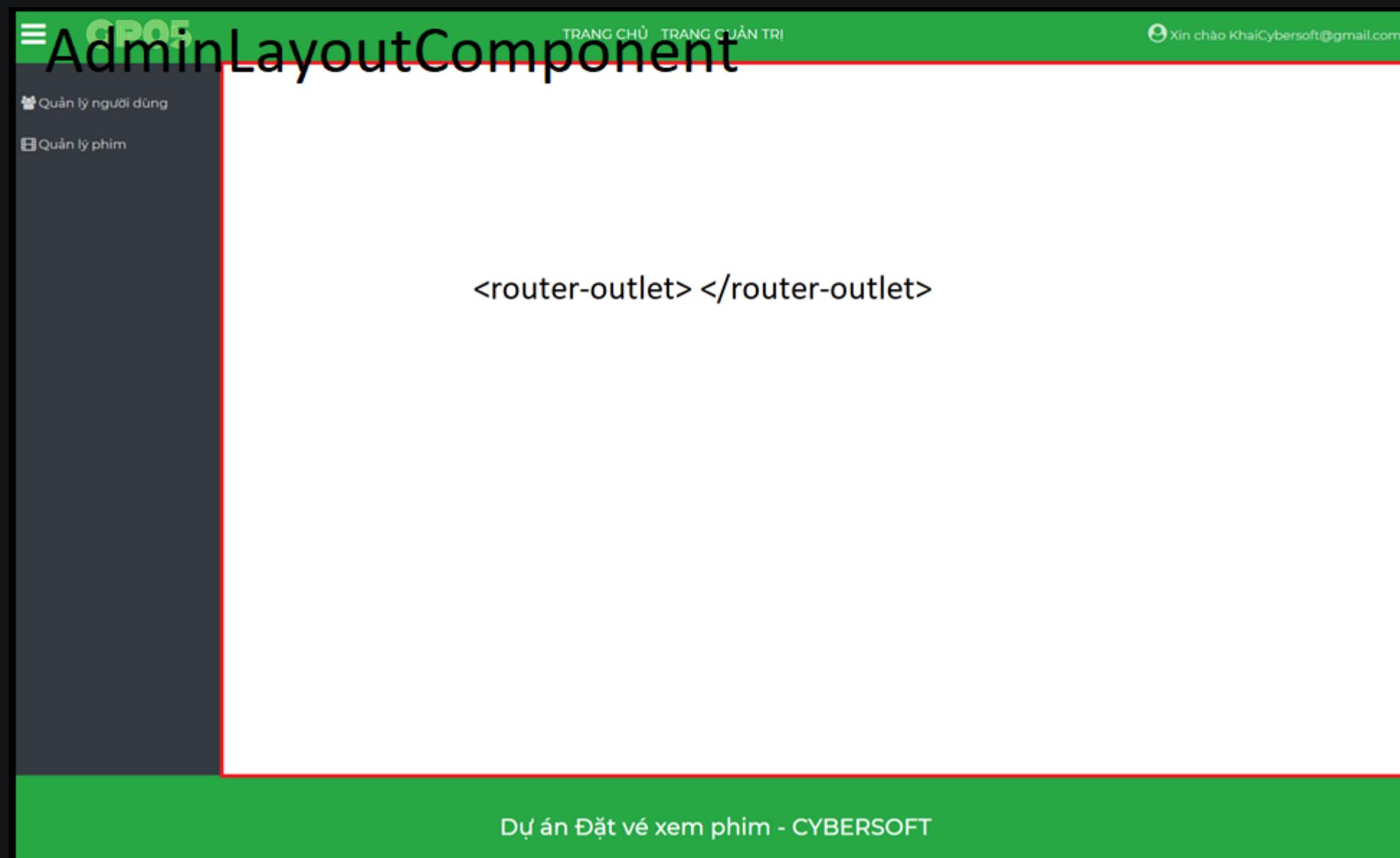
app.module.ts

```
ts home.module.ts x   ts app.module.ts      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         //http://localhost:4200/home/theo nhánh module Home hoặc http://localhost:4200/ (theo nhánh module '')
16         // Nếu người dùng /home và không / gì cả thì nó sẽ load tại router-outlet của thẻ HomeLayoutComponent là nội dung component TrangChu
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/trangchu(theo nhánh module Home) hoặc http://localhost:4200/trangchu (theo nhánh module '')
22         path: 'trangchu', component: TrangChuComponent
23       },
24       {
25         //Tương tự load http://localhost:4200/home/trangchitiet (theo nhánh module home)
26         //http://localhost:4200/trangchitiet (theo nhánh module '')
27         path: 'trangchitiet', component: TrangChiTietComponent
28       },
29     ]
30   }
31 ]
32
33 @NgModule({
34   imports: [
35     CommonModule, RouterModule.forChild(homeRoutes)
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 {}
```

Định tuyến trên
HomeModule

Gán homeRoutes vào RouterModule

Admin template (Admin module)



AdminLayoutComponent

TRANG CHỦ TRANG QUẢN TRỊ

Xin chào KhaiCybersoft@gmail.com

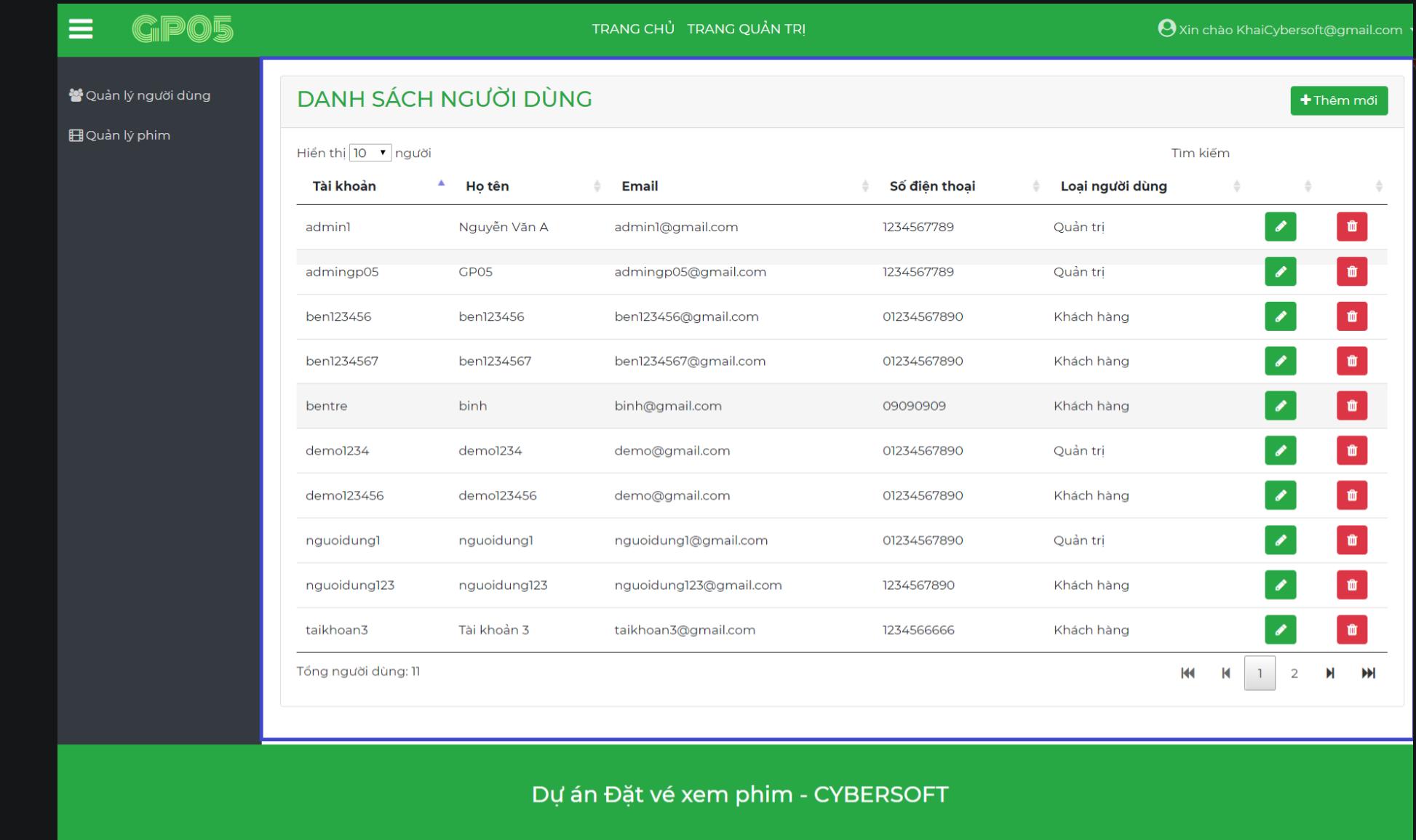
Quản lý người dùng

Quản lý phim

<router-outlet> </router-outlet>

Dự án Đặt vé xem phim - CYBERSOFT

Link: localhost:4200/admin



GPO5

TRANG CHỦ TRANG QUẢN TRỊ

Xin chào KhaiCybersoft@gmail.com

+Thêm mới

Quản lý người dùng

Quản lý phim

DANH SÁCH NGƯỜI DÙNG

Hiển thị 10 người

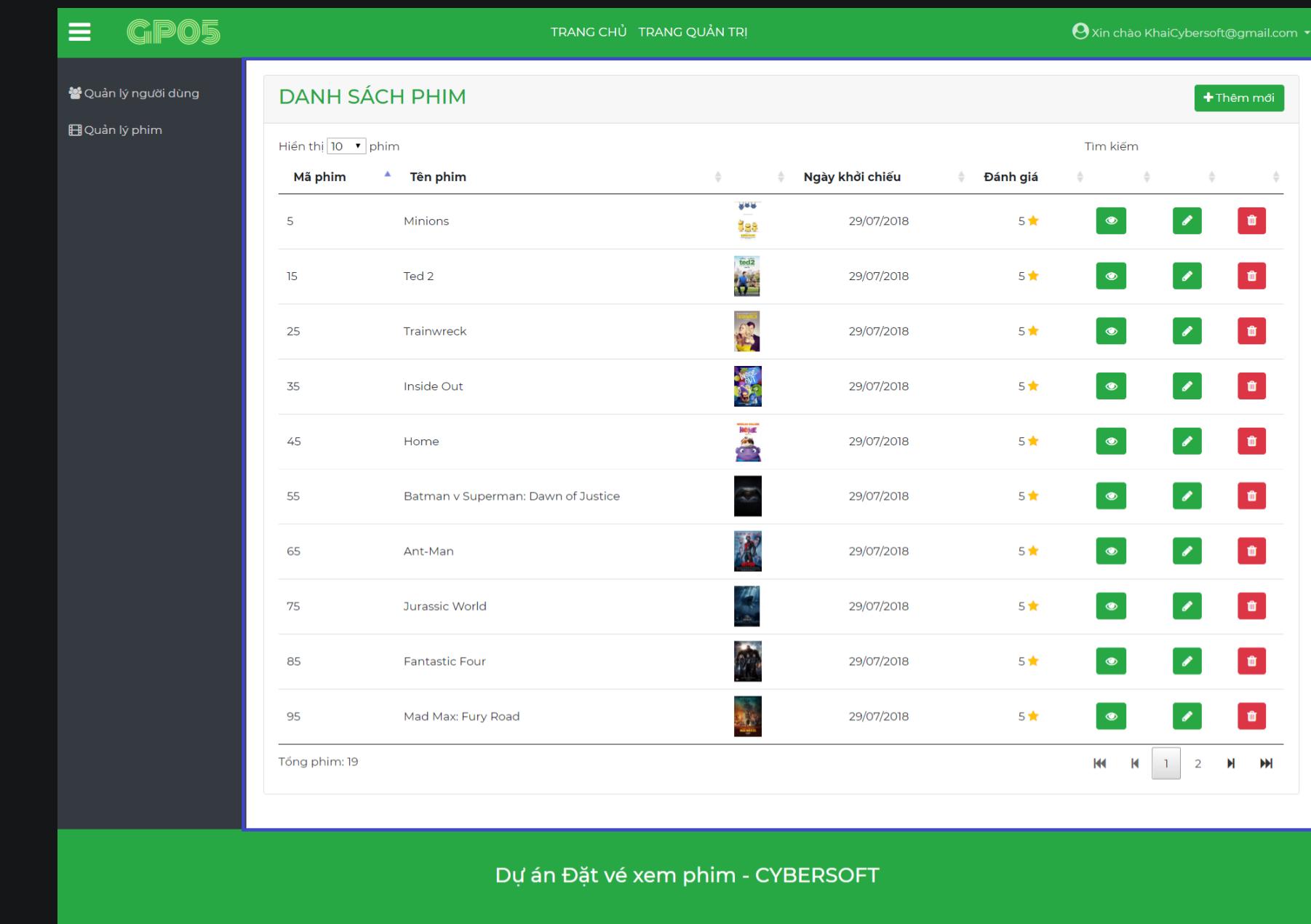
Tài khoản Họ tên Email Số điện thoại Loại người dùng

Tài khoản	Họ tên	Email	Số điện thoại	Loại người dùng
admin1	Nguyễn Văn A	admin1@gmail.com	1234567789	Quản trị
admiringp05	GPO5	admiringp05@gmail.com	1234567789	Quản trị
ben123456	ben123456	ben123456@gmail.com	01234567890	Khách hàng
ben1234567	ben1234567	ben1234567@gmail.com	01234567890	Khách hàng
bentre	binh	binh@gmail.com	09090909	Khách hàng
demo1234	demo1234	demo@gmail.com	01234567890	Quản trị
demo123456	demo123456	demo@gmail.com	01234567890	Khách hàng
nguoidung1	nguoidung1	nguoidung1@gmail.com	01234567890	Quản trị
nguoidung123	nguoidung123	nguoidung123@gmail.com	1234567890	Khách hàng
taikhoan3	Tài khoản 3	taikhoan3@gmail.com	1234566666	Khách hàng

Tổng người dùng: 11

Dự án Đặt vé xem phim - CYBERSOFT

Link: localhost:4200/admin/quanlynguoidung



GPO5

TRANG CHỦ TRANG QUẢN TRỊ

Xin chào KhaiCybersoft@gmail.com

+Thêm mới

Quản lý người dùng

Quản lý phim

DANH SÁCH PHIM

Hiển thị 10 phim

Mã phim Tên phim Ngày khởi chiếu Đánh giá

Mã phim	Tên phim	Ngày khởi chiếu	Đánh giá
5	Minions	29/07/2018	5 ★
15	Ted 2	29/07/2018	5 ★
25	Trainwreck	29/07/2018	5 ★
35	Inside Out	29/07/2018	5 ★
45	Home	29/07/2018	5 ★
55	Batman v Superman: Dawn of Justice	29/07/2018	5 ★
65	Ant-Man	29/07/2018	5 ★
75	Jurassic World	29/07/2018	5 ★
85	Fantastic Four	29/07/2018	5 ★
95	Mad Max: Fury Road	29/07/2018	5 ★

Tổng phim: 19

Dự án Đặt vé xem phim - CYBERSOFT

Link: localhost:4200/admin/quanlyphim

Cấu hình thực tế (HomeModule -> Template Home)

```
ts home.module.ts ts app.module.ts x 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
14   { path: 'admin', loadChildren: () => AdminModule} //localhost:4200/admin => load module admin
15 ]
16
17
18 @NgModule({
19   declarations: [
20     AppComponent
21   ],
22   imports: [
23     BrowserModule, RouterModule.forRoot(appRoutes) // Gắn đổi tượng appRoutes (khai báo các đường dẫn) vào đổi tượng Router (vì lâ
24   ],
25
26   bootstrap: [AppComponent]
27 })
28 export class AppModule { }
```

app.module.ts



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

admin.module.ts

```
ts admin.module.ts x
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: '', component: AdminLayoutComponent, children: [ //Trong AdminLayoutComponent ta cũng bố trí 1 thẻ <router-outlet>
11         { path: 'quanlynguoidung', component: QuanLyNguoiDungComponent }, //localhost:4200/admin/quanlynguoidung => load ra component QuanLyNguoiDungComponent
12         { path: 'quanlyphim', component: QuanLyPhimComponent } //localhost:4200/admin/quanlyphim => load ra component QuanLyPhimComponent
13       ]
14     }
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>



Kết quả

AdminLayoutComponent

TRANG CHỦ TRANG QUẢN TRỊ

Xin chào KhaiCybersoft@gmail.com

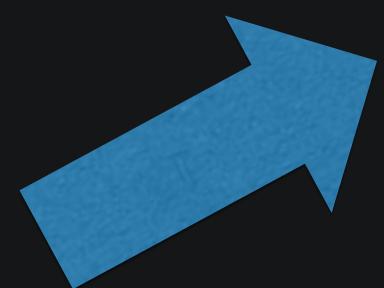
Quản lý người dùng

Quản lý phim

<router-outlet> </router-outlet>

Dự án Đặt vé xem phim - CYBERSOFT

Link: localhost:4200/admin



GPO5

TRANG CHỦ TRANG QUẢN TRỊ

Xin chào KhaiCybersoft@gmail.com

Quản lý người dùng

Quản lý phim

DANH SÁCH NGƯỜI DÙNG

+Thêm mới

Hiển thị 10 người

Tìm kiếm

Tài khoản	Họ tên	Email	Số điện thoại	Loại người dùng	Chỉnh sửa	Xoá
admin1	Nguyễn Văn A	admin1@gmail.com	1234567789	Quản trị		
admingroup05	GPO5	admingroup05@gmail.com	1234567789	Quản trị		
ben123456	ben123456	ben123456@gmail.com	01234567890	Khách hàng		
ben1234567	ben1234567	ben1234567@gmail.com	01234567890	Khách hàng		
bentre	binh	binh@gmail.com	09090909	Khách hàng		
demo1234	demo1234	demo@gmail.com	01234567890	Quản trị		
demo123456	demo123456	demo@gmail.com	01234567890	Khách hàng		
nguoidung1	nguoidung1	nguoidung1@gmail.com	01234567890	Quản trị		
nguoidung123	nguoidung123	nguoidung123@gmail.com	1234567890	Khách hàng		
taikhoan3	Tài khoản 3	taikhoan3@gmail.com	1234566666	Khách hàng		

Tổng người dùng: 11

Dự án Đặt vé xem phim - CYBERSOFT

Link: localhost:4200/admin/quanlynguoidung



GPO5

TRANG CHỦ TRANG QUẢN TRỊ

Xin chào KhaiCybersoft@gmail.com

Quản lý người dùng

Quản lý phim

DANH SÁCH PHIM

+Thêm mới

Hiển thị 10 phim

Tìm kiếm

Mã phim	Tên phim	Ngày khởi chiếu	Đánh giá	Chỉnh sửa	Xoá
5	Minions	29/07/2018	5 ★		
15	Ted 2	29/07/2018	5 ★		
25	Trainwreck	29/07/2018	5 ★		
35	Inside Out	29/07/2018	5 ★		
45	Home	29/07/2018	5 ★		
55	Batman v Superman: Dawn of Justice	29/07/2018	5 ★		
65	Ant-Man	29/07/2018	5 ★		
75	Jurassic World	29/07/2018	5 ★		
85	Fantastic Four	29/07/2018	5 ★		
95	Mad Max: Fury Road	29/07/2018	5 ★		

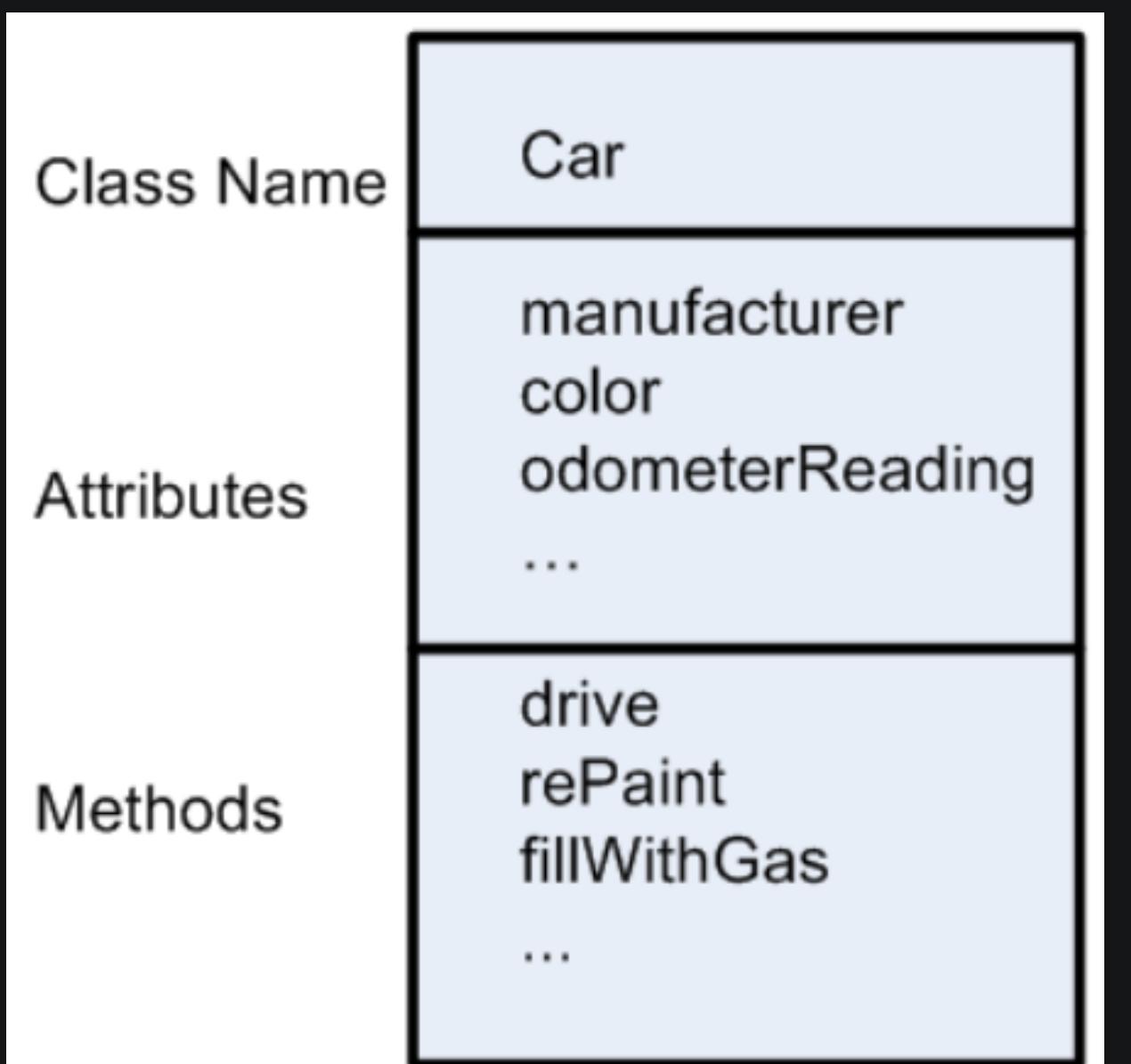
Tổng phim: 19

Dự án Đặt vé xem phim - CYBERSOFT

Link: localhost:4200/admin/quanlyphim

Object Class

- ❖ Object Class (TypeScript đã học) là prototype



Cú pháp tạo Class

ng g class [tên class]

Observable

Observable là 1 tính năng mới của ES7. Trước khi nhắc đến observable ta cần tìm hiểu 1 tí về callback function và promise(ES6)

Ôn tập 1 tí về khái niệm bất đồng bộ trong javascript hay còn gọi là ajax (Asynchronous JavaScript and XML)

```
Function ShowMessage()
{
    var message = 'hello';
    $.ajax({
        url : "url",
        data : {}
        success : function(result){
            message = result;
        },
        error:function(error){
            message = error;
        }
    });
    alert(message);
}
```

Theo các ban đoán code trên sẽ trả về giá trị message là gì ?
a. hello
b. Giá trị result trong hàm success
c. Giá trị error trong hàm error

Nói về callback function.

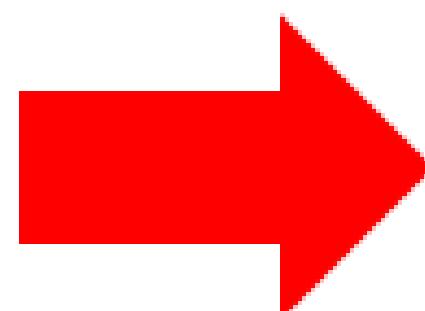
-Khái niệm: 1 function A được truyền vào 1 function B thông qua danh sách tham số của B. Lúc này tại hàm B sẽ gọi hàm A thực hiện một chức năng mà A đang nắm giữ. Đó là lý thuyết
Ví dụ: jQuery dùng rất nhiều callback function

```
$( 'selector' ).click(function(){
    alert('hello cybersoft');
});
```

Cách viết khác hàm click nhận vào tham số là 1 function, function xử lý chức năng cho sự kiện click.

```
$( 'selector' ).click(ShowMe
ssage);
```

```
function ShowMessage(){
    alert('hello cybersoft');
};
```



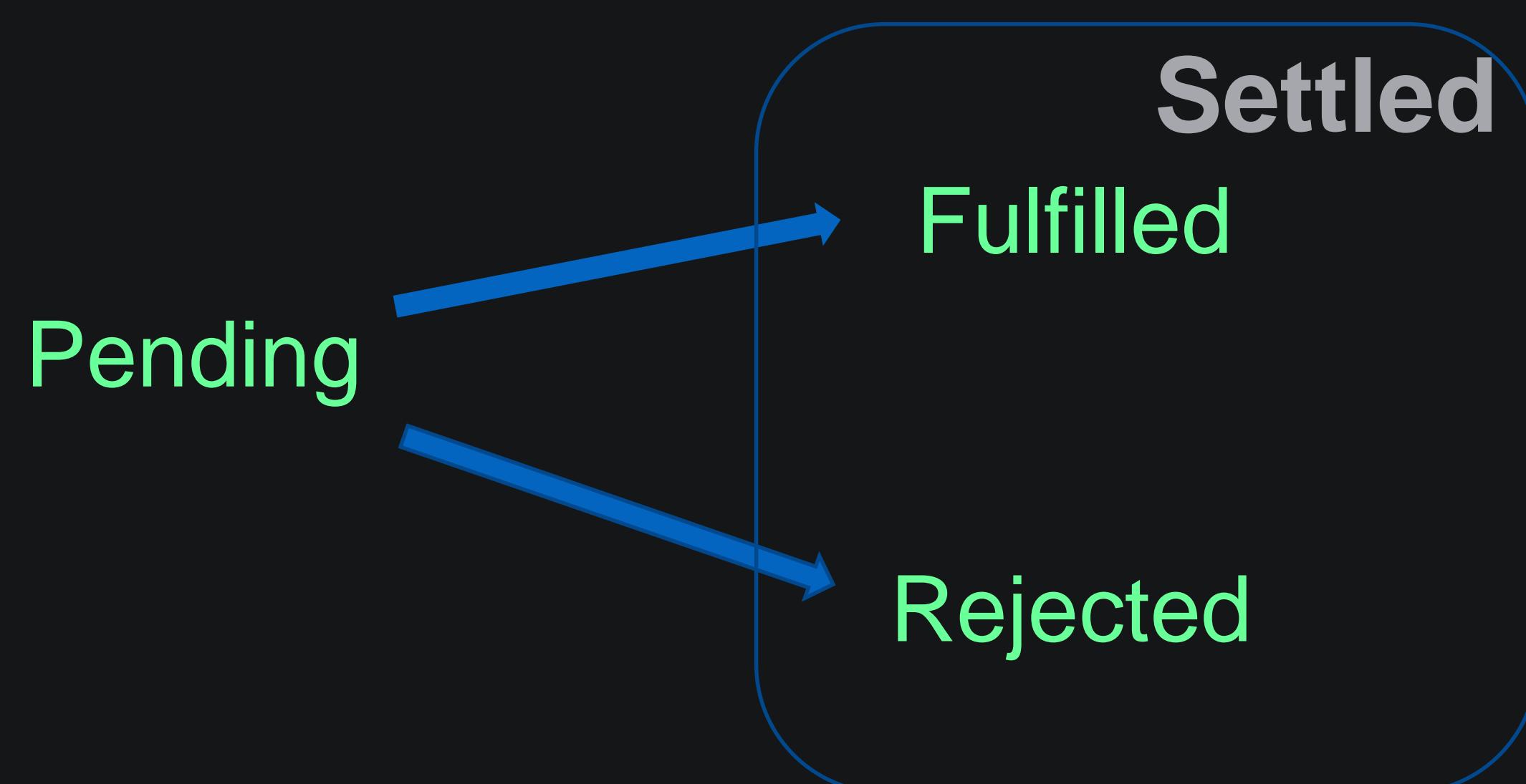
Đảm bảo mọi việc diễn ra theo đúng trình tự ví dụ khi click vào sự kiện 1 nút thì nút đó sẽ thực thi hành vi của 1 hàm khác. Jquery ứng dụng rất nhiều. Trong đó có ajax

Promise

Promise sinh ra để xử lý một kết quả của một hành động cụ thể, Kết quả của mỗi hành động sẽ có các trạng thái như

- + Thành Công làm gì (trạng thái Fulfilled)
- + Thất bại làm gì (trạng thái Rejected)
- + Chờ xử lý hoặc bị từ chối làm gì (Pending)

Nói cách khác nó cũng tựa như ajax khi này



Fulfilled Rejected gọi là **settled** tức là đã xử lý xong.

```

function ShowMessage()
{
    var promise = new promise(successData, faildData);
    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)
            faildData(request.statusText,request.status);
        }
    };
    //Gọi promise:
    promise.then(successData,faildData).cache(function(){
        console.log("error")
    });
}

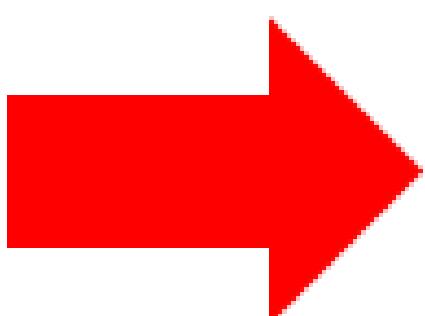
```

```

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

function faildData(error,status)
{
    message = error; //(2)
    alert(error)
}

```



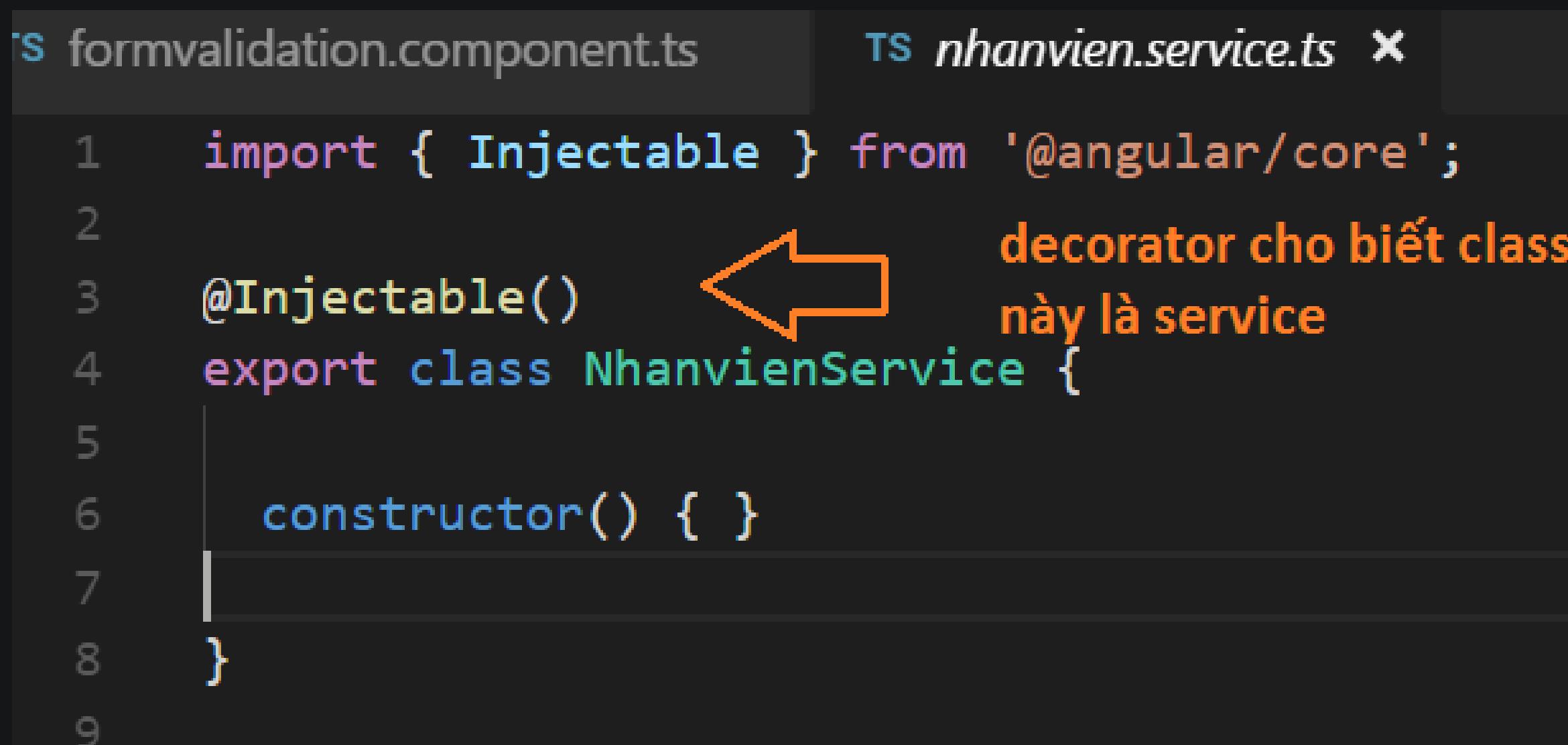
Kết quả của 1 hàm promise sẽ trả về 1 đối tượng promise nên setup tất cả các hàm xử lý chỉ cần .then() để quản lý các promise.

Observable

- ❖ Observable cũng tương tự như promise nhưng promise nhận giá trị tại **1 thời điểm** là thành công nhảy vào hàm thành công làm trả ra kết quả hoặc thất bại cũng vậy.
- ❖ Observable nó lấy **nhiều giá trị** tại **nhiều thời điểm**.
- ❖ 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.

□ Service - Observable - Object - HTTPService

- Service là chứa các phương thức tương tác với backend(server) để lấy hoặc load dữ liệu từ backend về thông qua các phương thức như GET, POST, PUT, DELETE và thư viện hỗ trợ từ angular 2 gọi là **Observable** để đổ vào đối tượng (Object).
- Ngoài ra ta còn có thể trung chuyển dữ liệu thông qua service giữa các component
- Nói nôm na service chứa các hàm gọi ajax để lấy dữ liệu về



TS formvalidation.component.ts

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable()
4 export class NhanvienService {
5
6   constructor() { }
7
8 }
```

TS nhanvien.service.ts

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable()
4 export class NhanvienService {
5
6   constructor() { }
7
8 }
```

decorator cho biết class
này là service

ng g service [tên service]

Sử dụng services

- Để sử dụng service cho component nào thì import service cho component đó. Nếu muốn sử dụng service cho toàn ứng dụng thì import service đó 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

```
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() { }
```

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 { }
```

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();
    }
}
```

Service

- ☐ Từ phiên bản 6 trở đi, angular tự động thêm thuộc tính providedIn:'root' vào file service, giúp chúng ta lược bỏ bước import và provider service ở module
- ☐ Sử dụng service ở component nào, truyền vào hàm khai tạo của component đó

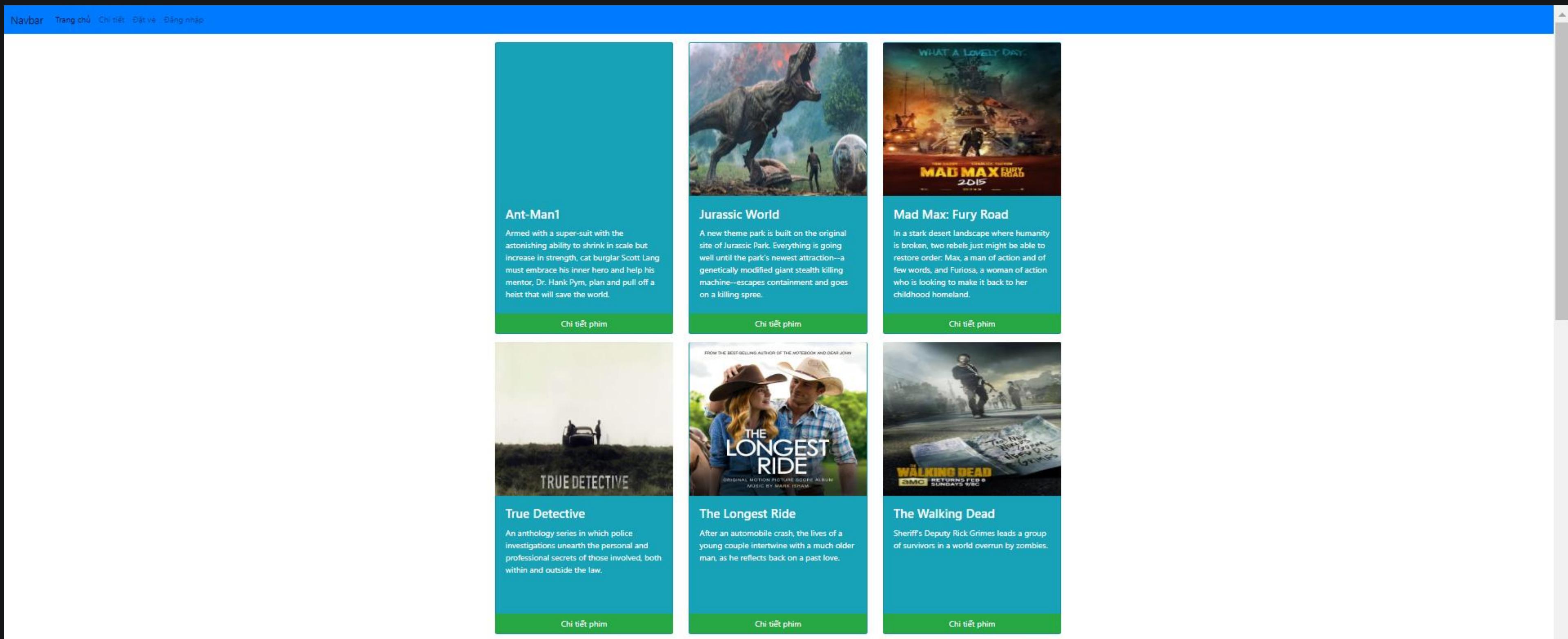
```
import { Injectable } from '@angular/core';

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

  constructor() { }
}
```

Bài tập

- 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



HTTP (Http Module)

Http module cung cấp cho chúng ta service http dung để giao tiếp với backend(server) thông qua một số phương thức như get, post, put, delete ...

```
import {HttpModule} from '@angular/http';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule, LayoutModule, DatabindingModule, DirectivesModule,
    TestserviceModule, MovieModule, HttpModule
  ],
  bootstrap: [AppComponent]
})
```

Để dùng được http service ta cần import **HttpModule** vào toàn ứng dụng thông qua **appmodule**

Get Post Put Delete(http service) – Lưu trữ cục bộ

Tương tự như jQuery angular 2 cũng hỗ trợ một số phương thức get post put ... Đã giới thiệu ở phần HTTP Module.

Đối với Backend các giao thức thường được sử dụng

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 xóa dữ liệu

Thực Hành: HTTP - GET

Bước 1: Tạo class service QuanLyPhim với nội dung

```
TS quan-ly-phim.service.ts ×

angularfe10 ▶ src ▶ app ▶ core ▶ Services ▶ TS quan-ly-phim.service.ts ▶ ...

1 import { Injectable } from '@angular/core';
2 import { Http, Response, Headers } from '@angular/http';
3 import { Observable } from 'rxjs';
4 import { map } from 'rxjs/operators';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class QuanLyPhimService {
10   readonly api: any = {
11     LayDanhSachPhim: "http://sv2.myclass.vn/api/QuanLyPhim/LayDanhSachPhim?MaNhom=GP01",
12   } //Đối tượng quản lý các chuỗi api
13   constructor(private http: Http) {
14     ↑          Đối tượng http dùng để kết nối với backend thông qua các
15   }                      phương thức get, post, put, delete, ...
16   LayDanhSachPhim(): Observable<any> {
17     let observable = this.http.get(this.api.LayDanhSachPhim)
18     .pipe(map(res: Response) => res.json());
19     return observable;
20   }
21 }
```

Đối tượng quản lý các chuỗi api từ backend

↑ Đối tượng http dùng để kết nối với backend thông qua các phương thức get, post, put, delete, ...

↑ Kết quả trả về sẽ được chứa trong hàm pipe. là một observable => chứa 2 phương thức thành công và không thành công. Tượng tự promise

Thực Hành: HTTP - GET

Bước 2: Tại HomeModule>QuanLyPhimComponent. Khai báo service và một đối tượng nhận giá trị kết quả từ service như hình bên dưới

```

  TS quan-ly-phim.service.ts    TS quan-ly-phim.component.ts ×

angularfe10 ▶ src ▶ app ▶ home ▶ quan-ly-phim ▶ TS quan-ly-phim.component.ts ▶ ...
1  import { Component, OnInit, OnDestroy } from '@angular/core';
2  import { QuanLyPhimService } from 'src/app/core/Services/quan-ly-phim.service';
3  import { DanhSachPhim } from 'src/app/core/Models/danh-sach-phim';
4  import { Subscription } from 'rxjs';
5  @Component({
6    selector: 'app-quan-ly-phim',
7    templateUrl: './quan-ly-phim.component.html',
8    styleUrls: ['./quan-ly-phim.component.css']
9  })
10 export class QuanLyPhimComponent implements OnInit,OnDestroy {
11
12   public dsPhim:DanhSachPhim = new DanhSachPhim();
13   private subDanhSachPhim = new Subscription(); //Đối tượng nhận giá trị là 1 observable
14   constructor(private svQuanLyPhim:QuanLyPhimService) { }
15
16   ngOnInit() {
17     this.subDanhSachPhim = this.svQuanLyPhim.LayDanhSachPhim().subscribe((ketqua)=>{
18       console.log(ketqua);
19       this.dsPhim.MangPhim = ketqua;
20     })
21   }
22   ngOnDestroy(){
23     // Phương thức này được thực thi khi component bị hủy ( ví dụ như chuyển sang trang khác là component khác)
24     // Khi chuyển sang component khác sự kiện ngOnDestroy sẽ chạy, hủy theo dõi biến svQuanLyPhim (chứa observable thuộc tính subscribe)
25     this.subDanhSachPhim.unsubscribe();
26   }
}

```

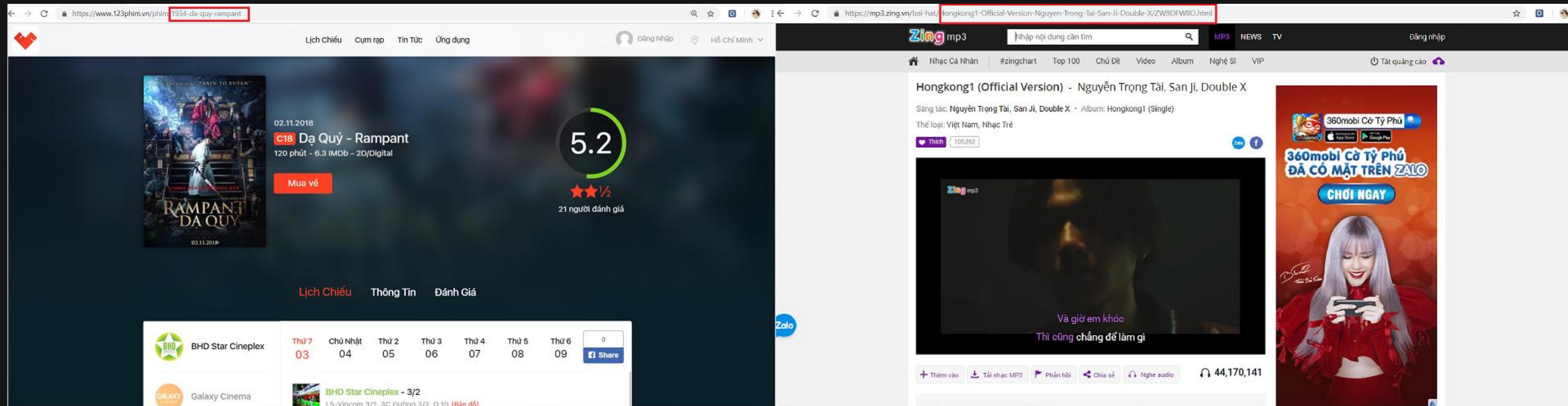
phương thức subscribe đăng ký theo dõi mỗi khi đối tượng observable có sự thay đổi sẽ tự gọi lại hàm (callback function) bên trong nó. (hàm (ketqua) => {});

☐ 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.



Parameter (tham số trên URL) trong angular

Parameter dùng để truyền nhận giá trị thông qua link (Url). Từ đó ta có thể dựa trên giá trị tham số đó để truy vấn trích xuất dữ liệu.

www.123phim.vn/phim/841-thor-ragnarok **tham số, parameter**

Lịch Chiếu Cụm rạp Tin Tức Ứng dụng Hồ Chí Minh

27.10.2017

C13 Thor: Tận Thế Ragnarok
(C13)
130 phút - 8.2 IMDb - IMAX/3D/2D

Mua vé

9

★★★★★½

270 người đánh giá

Lịch Chiếu Thông Tin Đánh Giá

☐ **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

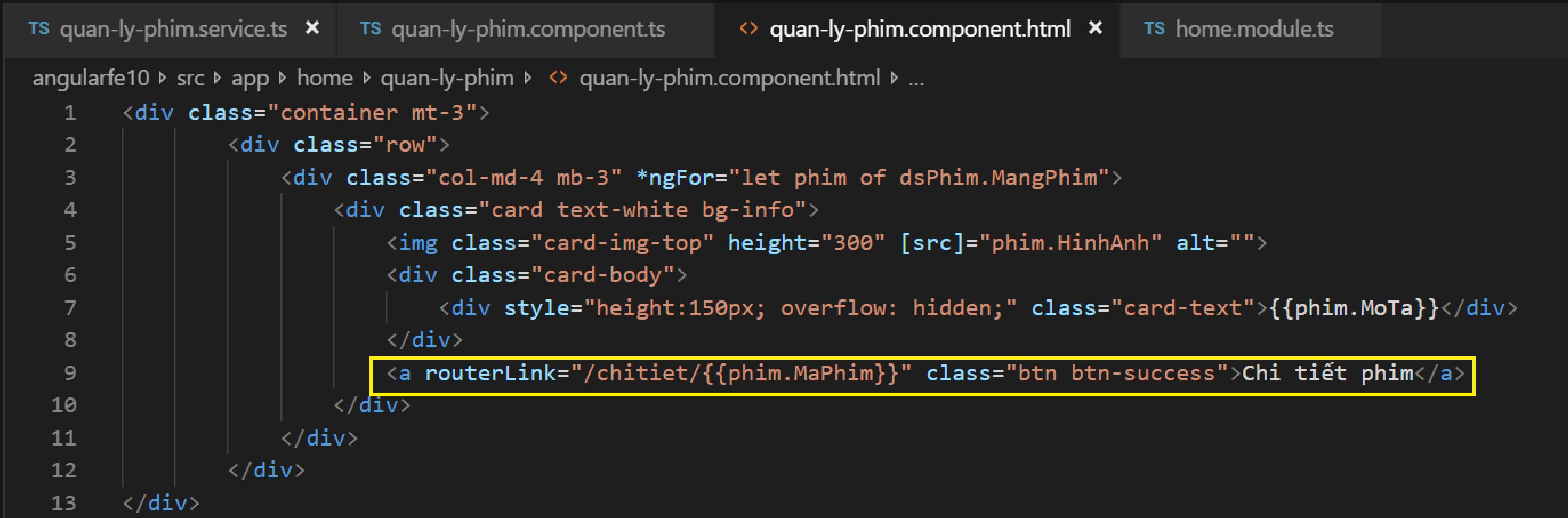
Thực hành truyền tham số

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       {path: 'chitiet/:id', component: ChiTietComponent},
23       {path: 'datve', component: DatVeComponent},
24       {path: 'pipe', component: PipeComponent},
25       {path: 'quanlynv', component: QuanLyNhanVienComponent},
26       {path: 'quanlyphim', component: QuanLyPhimComponent},
27     ]
28   }
29 ]
30 }
31 ]
32 |
33 @NgModule({
34   imports: [
35     CommonModule, RouterModule.forChild(homeRoutes), FormsModule
36   ],
37   declarations: [TrangChuComponent, ChiTietComponent, DatVeComponent, LayoutComponent, PipeComponent, Shortcu
```

Thực hành truyền tham số

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



The screenshot shows a code editor with four tabs at the top: 'TS quan-ly-phim.service.ts' (closed), 'TS quan-ly-phim.component.ts' (closed), '<> quan-ly-phim.component.html' (selected), and 'TS home.module.ts'. The code in the selected tab is:

```
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>
```

The line containing the router link is highlighted with a yellow box.

Thực hành truyền tham số

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



The screenshot shows a code editor with four tabs: quan-ly-phim.service.ts, quan-ly-phim.component.ts, quan-ly-phim.component.html, and home.module.ts. The quan-ly-phim.component.html tab is active, displaying the following code:

```
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>
```

The line containing the routerLink directive is highlighted with a yellow box.

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    ◁ quan-ly-phim.component.html    TS home.module.ts
angularfe10 › src › app › home › chi-tiet › 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   }
31   ngOnDestroy(){
32     this.subParam.unsubscribe();
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

Thực hành truyền tham số

Bước 4: Tạo service lấy chi tiết phim từ api của backend

```

's quan-ly-phim.service.ts ✘  TS chi-tiet.component.ts  TS quan-ly-phim.component.ts  ⌂ quan-ly-phim.component.html
angularfe10 ▶ src ▶ app ▶ core ▶ Services ▶ TS quan-ly-phim.service.ts ▶ ...
1 import { Injectable } from '@angular/core';
2 import { Http, Response, Headers } from '@angular/http';
3 import { Observable } from 'rxjs';
4 import { map } from 'rxjs/operators';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class QuanLyPhimService {
10   readonly api:any = {
11     LayDanhSachPhim:"http://sv2.myclass.vn/api/QuanLyPhim/LayDanhSachPhim?MaNhom=GP01",
12
13     LayChiTietPhim:"http://sv2.myclass.vn/api/QuanLyPhim/LayChiTietPhim?MaPhim="
14   } //Đối tượng quản lý các chuỗi api
15   constructor(private http:Http) {
16
17 }
18
19   LayDanhSachPhim(): Observable<any> {
20     let observable = this.http.get(this.api.LayDanhSachPhim)
21       .pipe(map((res: Response) => res.json()));
22     return observable;
23   }
24
25   LayChiTietPhim(id: string): Observable<any> {
26     let observable = this.http.get(this.api.LayChiTietPhim + id)
27       .pipe(map((res: Response) => res.json()));
28     return observable;
29   }
30
31 }
32

```

Service lấy chi tiết phim

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},
    ]
  }
]
```

+ 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                  </div>
10                 <!-- <a routerLink="/chitiet?MaPhim={{phim.MaPhim}}&TenPhim={{phim.TenPhim}}" class="btn btn-success">Chi tiết phim</a>
11                 <a [routerLink]=["/chitiet"]
12                   [queryParams]={MaPhim:phim.MaPhim,TenPhim:phim.TenPhim}"
13                   class="btn btn-success">Chi tiết phim</a>
14             </div>
15         </div>
16     </div>
17 </div>
```

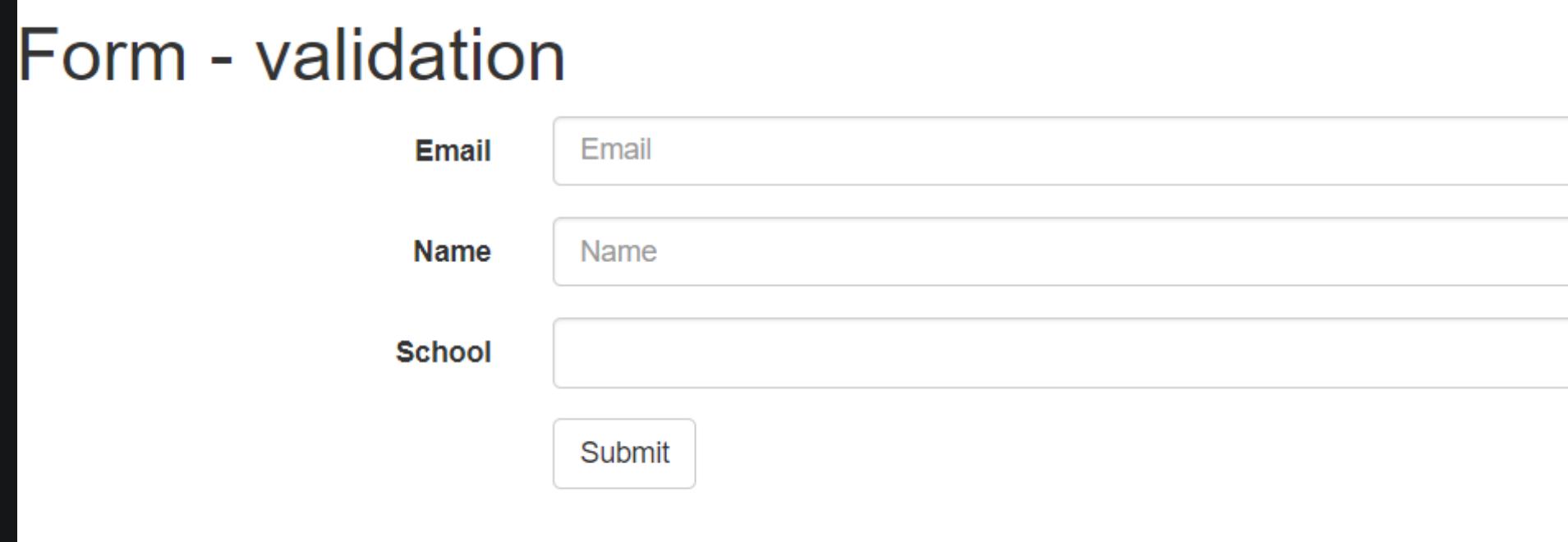
+ Bước 3: Để lấy thông tin các tham số từ routink ta cũng dùng đối tượng activateRouter thông qua thuộc tính queryparam.

```
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



```
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: gửi vào tham số là đối tượng **ngForm** (biến **registerForm.value**)

□ Form – Form Validation

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 – Validation (Validation)

❖ 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 – Validation (Validation)

- ❖ Kiểm tra rỗng với **required**

Lưu ý: Ví dụ trên chụp còn thiếu ngModel trong the input

```
<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">
    Email không được rỗng !
  </div>
</div>
```

Xét trong lỗi đó có required không nếu có thì hiển thị div báo lỗi này.

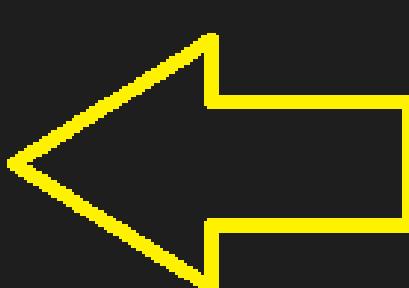
Nếu có sự thay đổi của thẻ input#email hoặc #email được chạm vào. && #email errors (required, không đúng định dạng, quá ký tự ...) thì sẽ hiển thị div này.
Ví dụ (true && true = true) nên hiển thị

Form – Validation (Validation)

❖ 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.errorsmaxlength)">
        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 – Validation (Validation)

❖ Kiểm tra định dạng chuỗi nhập vào với pattern

```
<input type="email" pattern="[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-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

□ 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>
</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 – Validation (Validation)

❖ 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-default">Submit</button>
  </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 – Validation (Validation)

❖ Ngoài ra angular 2 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

➤ Ví dụ:

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

- ✓ 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

➤ Kết quả:

Form - validation

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

Name:

Name:

➤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 #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: gửi vào tham số là đối tượng `ngForm` (biến `registerForm.value`)

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

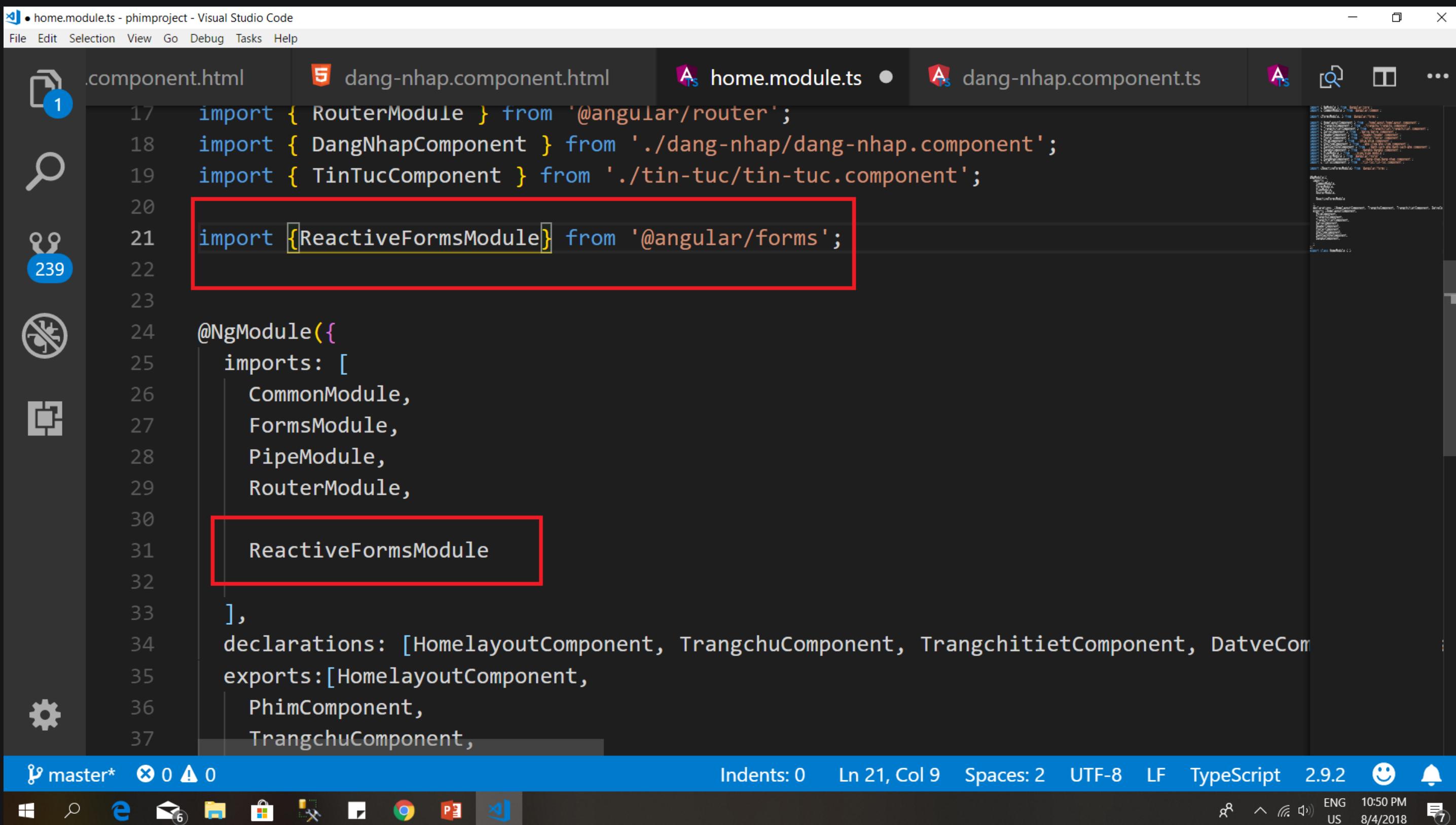
Register

☐ **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.**
- ✓ **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 6 hỗ trợ thêm các module để handle form, trong đó có ReactiveForm.
- ❖ Cách sử dụng ReactiveFormModule :
 - ❑ 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

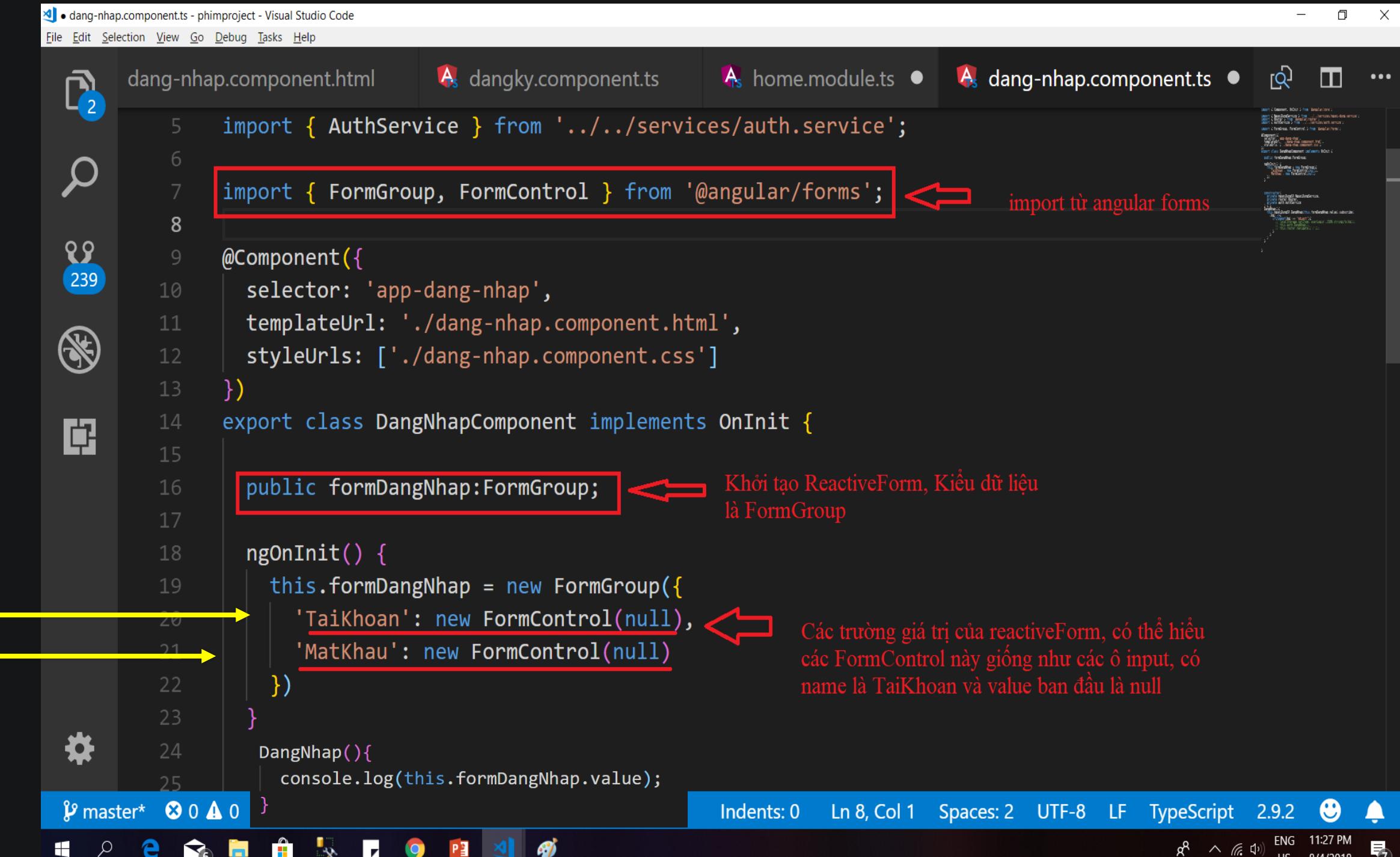
.component.html      5 dang-nhap.component.html    A home.module.ts ●   A dang-nhap.component.ts
A 1
17 import { RouterModule } from '@angular/router';
18 import { DangNhapComponent } from './dang-nhap/dang-nhap.component';
19 import { TinTucComponent } from './tin-tuc/tin-tuc.component';
20
21 import { 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,
```

The screenshot shows the Visual Studio Code interface with the file 'home.module.ts' open. The code imports several modules and includes the 'ReactiveFormsModule' import statement at line 21, which is highlighted with a red rectangle. The code editor has a dark theme with light-colored syntax highlighting. The bottom status bar shows the file path as 'home.module.ts - phimproject - Visual Studio Code', the line and column as 'Ln 21, Col 9', and the TypeScript version as '2.9.2'. The system tray at the bottom right shows the date and time as '8/4/2018 10:50 PM'.

□ 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.ts - phimproject - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
dang-nhap.component.html dangky.component.ts home.module.ts dang-nhap.component.ts ...
2 5 import { AuthService } from '../../../../../services/auth.service';
6
7 import { FormGroup, FormControl } from '@angular/forms'; ← import từ angular forms
8
9 @Component({
10   selector: 'app-dang-nhap',
11   templateUrl: './dang-nhap.component.html',
12   styleUrls: ['./dang-nhap.component.css']
13 })
14 export class DangNhapComponent implements OnInit {
15
16   public formDangNhap: FormGroup; ← Khởi tạo ReactiveForm, Kiểu dữ liệu
17   là FormGroup
17
18   ngOnInit() {
19     this.formDangNhap = new FormGroup({
20       'TaiKhoan': new FormControl(null), ← Các trường giá trị của reactiveForm, có thể hiểu
21       'MatKhau': new FormControl(null)       là các FormControl này giống như các ô input, có
22     })
23   }
24
25   DangNhap(){
26     console.log(this.formDangNhap.value);
27   }
}
master* 0 0 0 0 Indents: 0 Ln 8, Col 1 Spaces: 2 UTF-8 LF TypeScript 2.9.2 ENG 11:27 PM US 8/4/2018
```

☐ ReactiveForm

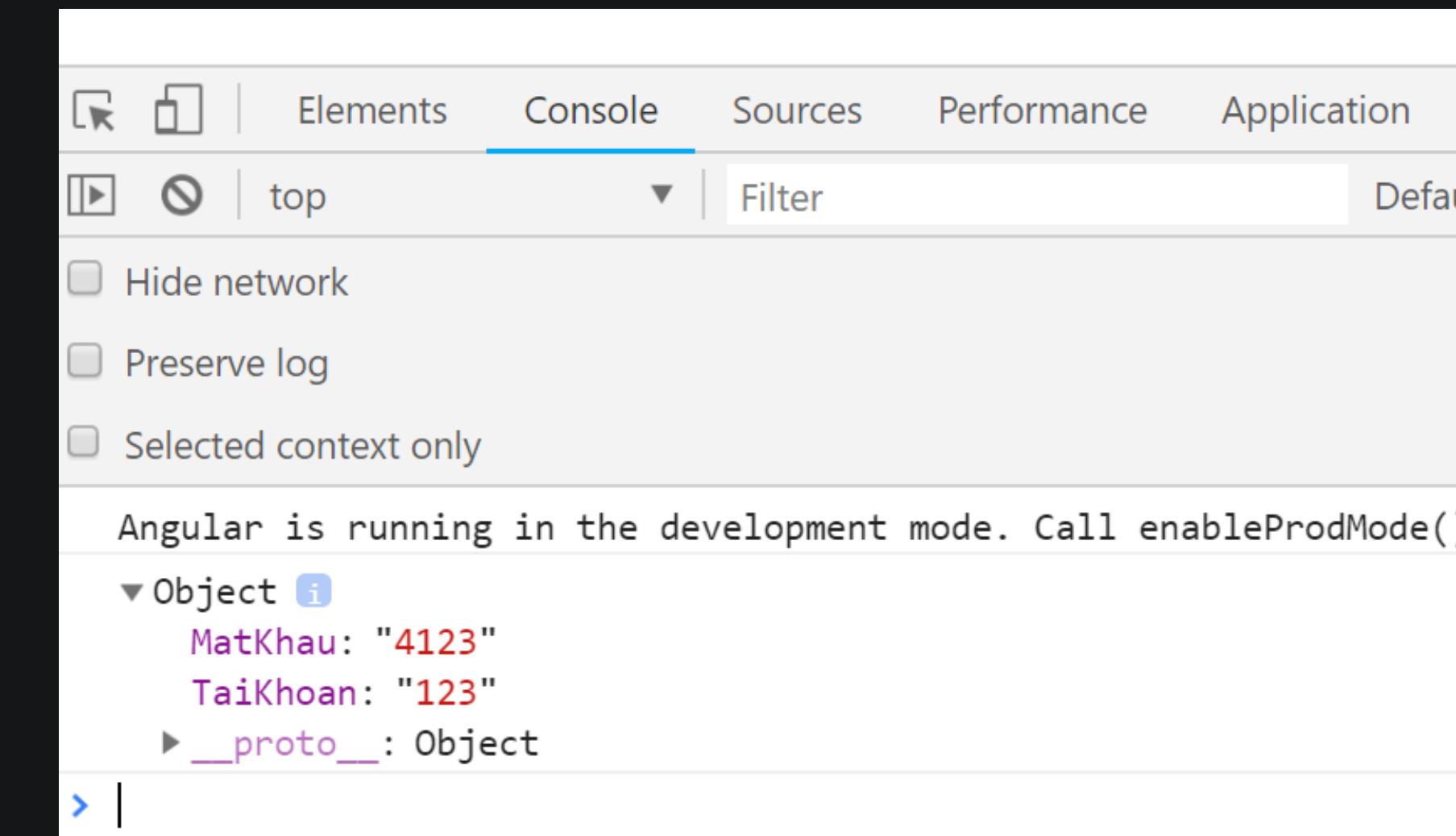
☐ Kết quả:

Đăng Nhập

Tài Khoản

Mật Khẩu

Đăng nhập

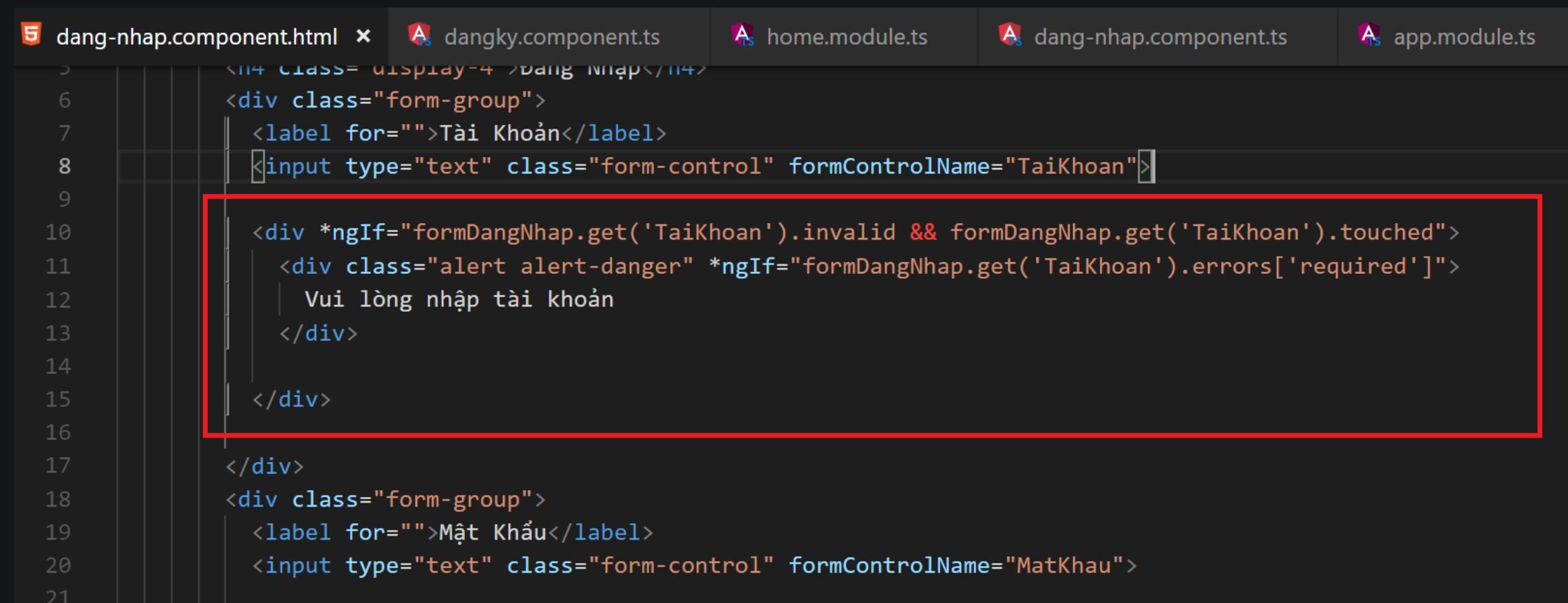


The screenshot shows the browser's developer tools open to the 'Console' tab. The console output displays the following information:

```
Angular is running in the development mode. Call enableProdMode() to enable the production mode.
Object
  MatKhau: "4123"
  TaiKhoan: "123"
  __proto__: Object
```

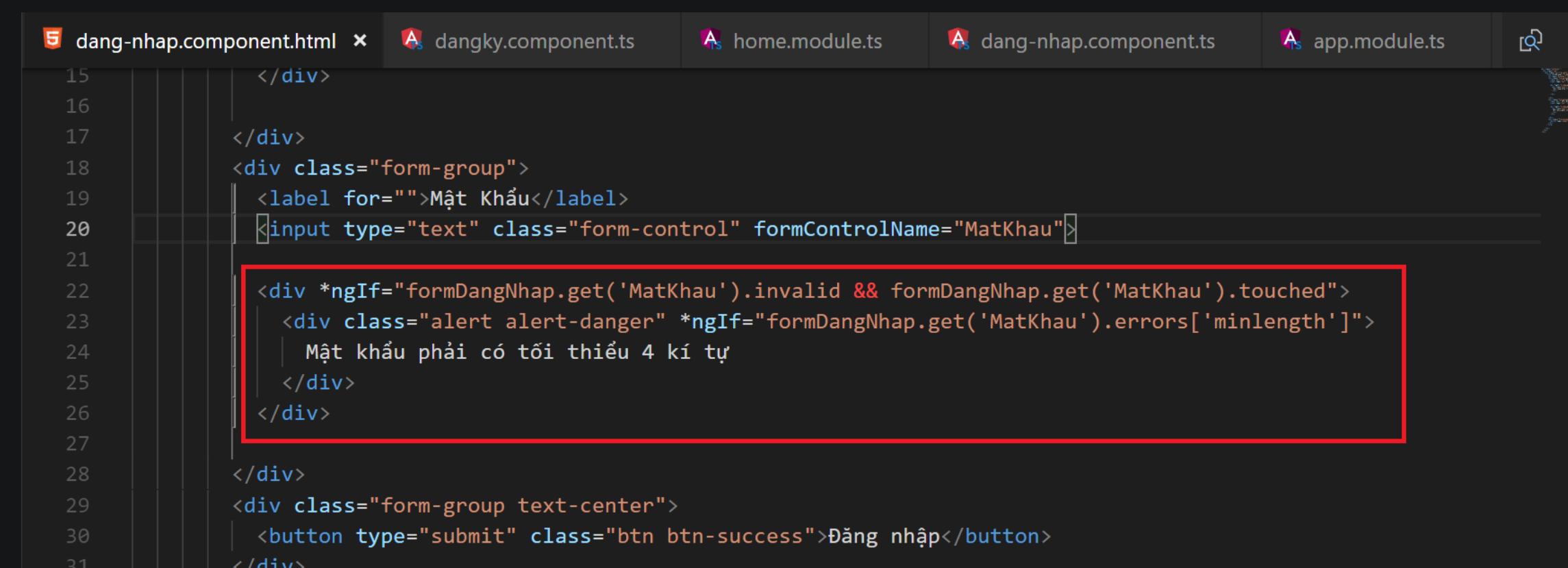
☐ Validation với ReactiveForm

- ☐ Ở HTML, kiểm tra không hợp lệ thì hiện alert



```
dang-nhap.component.html
<div class="form-group">
  <label for="">>Tài Khoản</label>
  <input type="text" class="form-control" formControlName="TaiKhoan">

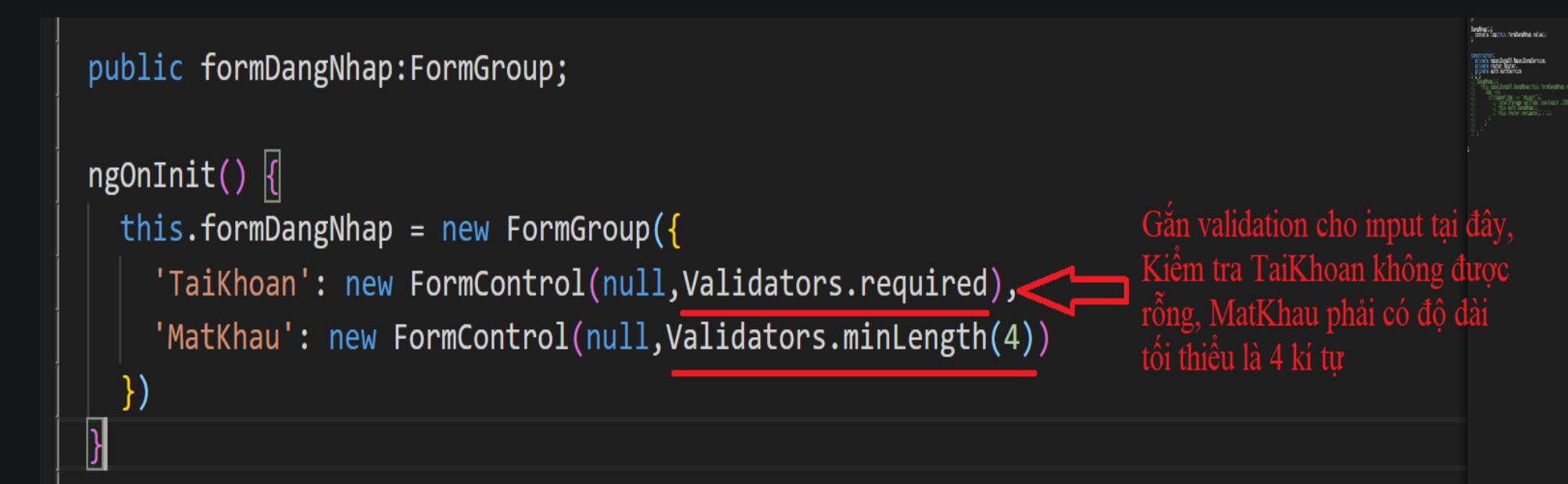
  <div *ngIf="formDangNhap.get('TaiKhoan').invalid && formDangNhap.get('TaiKhoan').touched">
    <div class="alert alert-danger" *ngIf="formDangNhap.get('TaiKhoan').errors['required']">
      | Vui lòng nhập tài khoản
    </div>
  </div>
</div>
```



```
dang-nhap.component.html
<div class="form-group">
  <label for="">>Mật Khẩu</label>
  <input type="text" class="form-control" formControlName="MatKhau">

  <div *ngIf="formDangNhap.get('MatKhau').invalid && formDangNhap.get('MatKhau').touched">
    <div class="alert alert-danger" *ngIf="formDangNhap.get('MatKhau').errors['minlength']">
      | Mật khẩu phải có tối thiểu 4 kí tự
    </div>
  </div>
</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

☐ 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
31  </div>

```

☐ 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

#	No.	Hình ảnh	Tên Phim	Sản xuất	Ngày công chiếu	Trailer
<input type="checkbox"/>	0		jkhakjhajkshjkdas	fgsdfgsdfg	2018-06-12T00:00:00	dsfgsdfgsdfg
<input type="checkbox"/>	1		Minions	Kyle Bald	2015-07-10T00:00:00	https://www.youtube.com/watch?v=Wfql_DoHRKc
 Activate Windows Go to Settings to activate Windows.						
<input type="text" value="ID"/>		<input type="text" value="Title"/>				
<input type="text" value="Description"/>						
<input type="text" value="Rating"/>		<input type="text" value="Director"/>				
<input type="text" value="Release Date"/>		<input type="text" value="Trailer"/>				
<input type="text" value="Image Url"/>						
<input type="button" value="ADD MOVIE"/>						

☐ Tao 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 kí tự, lớn hơn 10 kí tự thì cắt bớt và them dấu “...”
- ✓ Ngày chiếu hiển thị theo định dạng Ngày-Tháng-Năm Giờ : phút : giây
- ✓ ID hiển hiện ra bảng phải là chữ in hoa.
- ✓ Thêm button Cập nhật, sửa lại phim.

Pipes

Là một filter của angular giúp format định dạng dữ liệu khi hiển thị ra màn hình

LOẠI PIPE	MÔ TẢ	CÚ PHÁP
LOWERCASEPIPE	CHUYỂN ĐỔI TỪ HOA SANG THƯỜNG	<code>{{[GIA TRỊ] LOWERCASE}}</code>
UPPERCASEPIPE	CHUYỂN ĐỔI TỪ CHỮ THƯỜNG => HOA	<code>{{[GIA TRỊ] UPPERCASE}}</code>
DATEPIPE	DÙNG ĐỂ ĐỊNH DẠNG NGÀY THÁNG NĂM	<code>{{TODAY DATE:'FULLDATE'}}</code>
PERCENTPIPE	ĐỊNH DẠNG PHẦN TRĂM (SỐ ĐÓ * 100)	<code>{{VALUE PERCENT}}, {{VALUE A.B-C}}</code>
DECIMALPIPE	ĐỊNH DẠNG TIỀN TỆ, SỐ LIỆU	<code>{{VALUE PERCENT}}, {{VALUE A.B-C}}</code>
JSONPIPE	CHUYỂN TỪ 1 MẢNG JSON => CHUỖI JSON	<code>{{OBJECTJSON JSON }}</code>
JSONSLICE	LẤY SỐ PHẦN TỬ TỪ X->Y-1 TRONG MẢNG	<code>{{OBJECTJSON SLICE:2:3}}</code>

❖ Ví dụ:

```
TS pipe.component.ts x
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 ShortcutPipe 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 **ShortcutPipe**

```
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, Trangdat
  exports: [TrangchuComponent, TrangchitietComponent, TrangdatgheCo
})
export class HomeModule { }
```

Cơ chế Guard (CanActivate – CanDeactive)

❖ CanActivated

CanActivated là cơ chế xác nhận quyền có được truy cập vào trang đó không.

❖ Khởi tạo và định nghĩa

```
ng g guard [Ten_Guard]
```

❖ Định nghĩa

TS is-login.guard.ts ✘

projectPrimeNG ▶ src ▶ app ▶ _core ▶ guards ▶ TS is-login.guard.ts ▶ ...

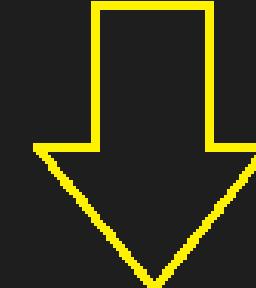
```
1 import { Injectable } from '@angular/core';
2 import { CanActivate} from '@angular/router';
3
4
5 @Injectable({
6   providedIn: 'root'
7 })
8 export class IsLoginGuard implements CanActivate {
9   canActivate():boolean {
10     //Biểu thức
11     return true; //Cho phép get vào link Route
12     //giá trị false là không cho get vào
13   }
14 }
```

phương thức chứa
xử lý trả về giá trị
true (cho phép được vào
router đó) hoặc false
(không cho phép vào
route đó)

Tại những nơi cấu hình routerLink ta gắn các đối tượng Guard vào để dựa vào đối tượng này website có cho phép get vào link này hay không.

```
const homeRoute:Routes = [
  {path: '', component:LayoutHomeComponent, children:[
    {path: '', component:TrangChuComponent},
    {path:'trangchu', component:TrangChuComponent},
    {path:'baitap', component:BaitapComponent},
    {path:'quanlyphim', component:QuanlyphimComponent},
    // {path:'chitietphim/:id', component:ChiTietPhimComponent},
    {path:'chitietphim', component:ChiTietPhimComponent},
    {path:'datve/:id', component:DatveComponent, canActivate: [ IsLoginGuard ] },
    {path:'dangnhap', component:DangNhapComponent},
    {path:'dangky', component:DangKyComponent, canDeactivate: [ IsSaveRegisterFormGuard ] },
  ]}
]
```

Truyền vào đối tượng Guard để xét quyền truy cập vào link bên dưới



Cách khởi tạo cũng tương tự CanActivate là cơ chế ràng buộc trước khi rời khỏi trang đó (route link đó)

❖ Định nghĩa

```
TS is-save-register-form.guard.ts ✘  
  
projectPrimeNG › src › app › _core › guards › TS is-save-register-form.guard.ts › IsSaveRegisterFormGuard  
1 import { Injectable } from '@angular/core';  
2 import { CanDeactivate } from '@angular/router';  
3 import { DangKyComponent } from 'src/app/home/dang-ky/dang-ky.component';  
4 @Injectable({  
  providedIn: 'root'  
})  
export class IsSaveRegisterFormGuard implements CanDeactivate<DangKyComponent> {  
  //Component nằm trong Canactivate là component mình sẽ thao tác đến để dựa vào  
  // 1 trong các thuộc tính đó xác định giá trị true (Cho phép rời khỏi trang) false (Không cho phép)  
  canDeactivate(component: DangKyComponent):boolean{  
    if(component.checkSaveForm)  
    {  
      return true;  
    }  
    alert("Bạn chưa lưu");  
    return false;  
  }  
}
```

Dựa vào 1 thuộc tính mà mình cài đặt tại component đó để xác định cho phép rời khỏi trang hay không. Giá trị trả về là true (Được phép rời khỏi trang) False (Không được phép rời khỏi trang)

Tại những nơi cấu hình routerLink ta gắn các đối tượng Guard vào để dựa vào đối tượng này website có cho phép rời khỏi trang này theo điều kiện đã định nghĩa trong hàm CanDeactivate trả về true hay không.

```
const homeRoute:Routes = [
  {path: '', component:Layout HomeComponent, children:[
    {path: '', component:TrangChuComponent},
    {path: 'trangchu', component:TrangChuComponent},
    {path: 'baitap', component:BaitapComponent},
    {path: 'quanlyphim', component:QuanlyphimComponent},
    // {path: 'chitietphim/:id', component:ChiTietPhimComponent},
    {path: 'chitietphim', component:ChiTietPhimComponent},
    {path: 'datve/:id', component:DatveComponent, canActivate:[IsLoginGuard]},
    {path: 'dangnhap', component:DangNhapComponent},
    {path: 'dangky', component:DangKyComponent, canDeactivate:[IsSaveRegisterFormGuard]}, 
  ]}
]
```

❖ Thực hành với POST kết hợp cơ chế canActivate canDeactivate

❖ CanDeactivate:

+B1: Xây dựng trang (component) đăng ký (thuộc homeModule), trang này sẽ chứa form đăng ký và 1 thuộc tính `checkInputForm` kiểu `Boolean` mang giá trị mặc định là `false` → xử lý khi điền đầy đủ các trường và bấm submit trả về `true`

+B2: Xây dựng 1 Guard implement thư viện `CanDeactivate<[TenComponentDangKy]>`

+B3: Tại hàm `CanDeactivate` (Thuộc Guard `CanDeactivate`) ta sẽ dựa vào giá trị của thuộc tính `checkInputForm` thuộc component đăng nhập nếu thuộc tính đó mang giá trị `true` ta cho phép hàm `CanDeactivate` trả về `true` nghĩa là được phép rời khỏi trang này. Ngược lại = `false`.

+B4: Gắn Guard này vào `homeRoute` (thuộc `HomeModule`) tại thuộc tính của object link (`CanDeactivate`). Ở đây ta gắn vào trang đăng ký

+B4: Thực hành với **POST**. Xây dựng service `QuanLyNguoiDung`, tại service này xây dựng phương thức đăng ký dựa vào link api cung cấp:

+ api: `http://sv3.myclass.vn/api/quanlynguoidung/dangky`

+ header: “Content-Type” “application/json;charset=UTF-8”

+ ObjectType: {`TaiKhoan:`”,`MatKhau:`”,`HoTen:`”,`Email:`”,`SoDT:`”} (có thể tạo đối tượng `modelClass`)

+B5: Xử lý sự kiện cho nút submit Form gọi service để push dữ liệu từ người dùng nhập vào về webserver lưu trữ.

❖ Thực hành với POST kết hợp cơ chế canActivate deactivate

CYBERSOFT Trang chủ Bài tập Quản lý phim Đăng nhập Đăng ký

Đăng ký

Tài khoản

Mật khẩu

Họ tên

Email

Số điện thoại

Phân service đăng ký

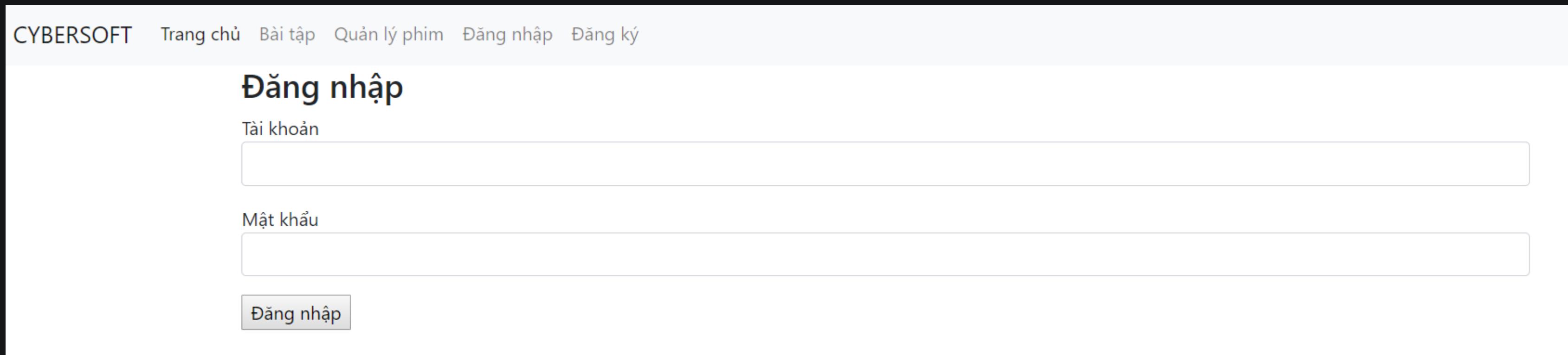
```
public DangKy/nguoIDung:NguoiDung):Observable<any[]>{
    let linkApi = `http://sv3.myclass.vn/api/quanlynguoidung/DangKy`;
    let header = new Headers(); //Class header được import từ @angular/http
    header.append("Content-Type", "application/json; charset=UTF-8");
    let observable = this.http.post(linkApi, nguoIDung, {headers:header})
        .pipe(map((result: Response) => { return result.json() }));
    return observable;
}
```

❖ Thực hành với POST kết hợp cơ chế canActivate canDeactivate

❖ CanDeactivate:

- +B1: Xây dựng trang (component) đăng nhập (thuộc homeModule), trang này sẽ chứa form đăng nhập.
- +B2: Thực hành với POST. Tại service QuanLyNguoiDung xây dựng phương thức DangNhap
 - + api: <http://sv3.myclass.vn/api/quanlynguoidung/dangnhap>
 - + header: "Content-Type" "application/json; charset=UTF-8"
 - + ObjectType: {TaiKhoan:"", MatKhau:""} (có thể sử dụng modelClass cùng với đăng ký)
- +B3: Tại component DangNhap gọi service đăng nhập truyền vào đối tượng người dùng, xử lý kết quả trả về thành công thì lưu kết quả vào localStorage.
- +B4: Xây dựng 1 Guard isLogin (Kiểm tra user login chưa)
- +B5: Tại hàm CanActivate (Thuộc Guard CanActivate) ta sẽ kiểm tra localStorage có giá trị chưa. Nếu đã có giá trị thì => đã đăng nhập rồi => return về true hoặc ngược lại.
- +B6: Gắn Guard này vào homeRoute (thuộc HomeModule) tại thuộc tính CanActivate của object link của những link nào cần verify đăng nhập.

❖Hình ảnh minh họa bước 1:



CYBERSOFT Trang chủ Bài tập Quản lý phim Đăng nhập Đăng ký

Đăng nhập

Tài khoản

Mật khẩu

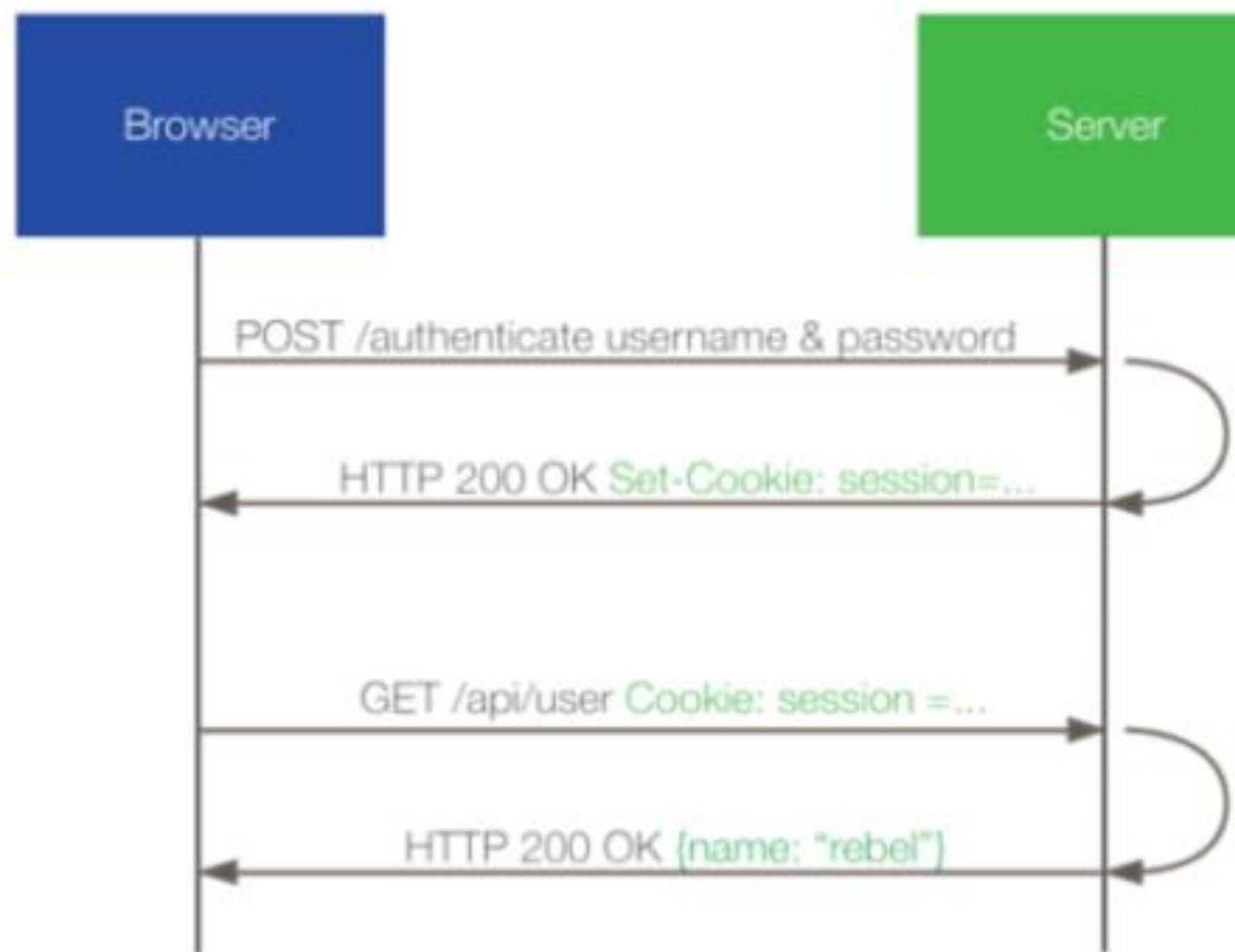
Đăng nhập

Service đăng nhập

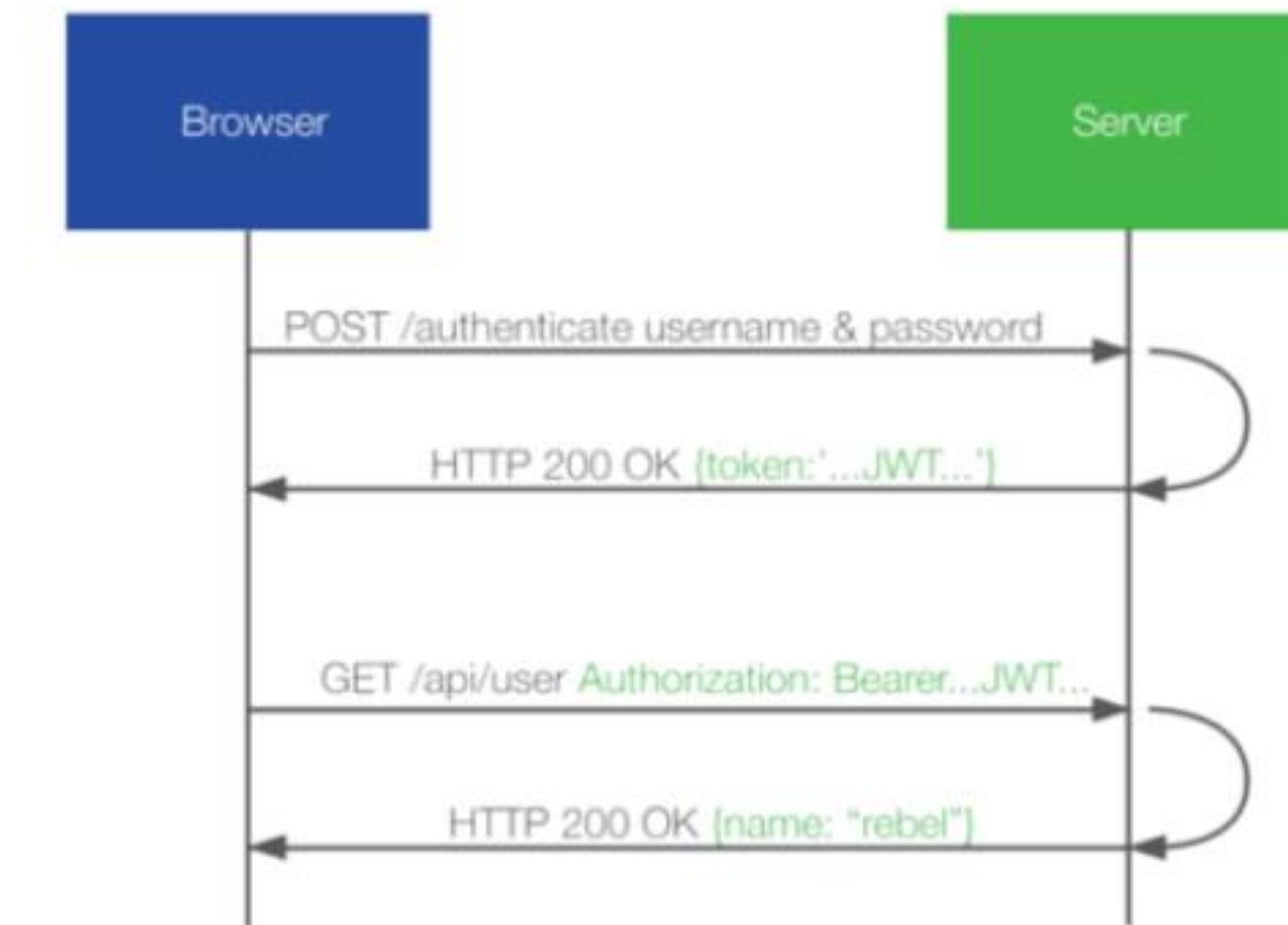
```
public DangNhap.nguoIDung:NguoiDung):Observable<any[]>{  
  
let linkApi = `http://sv3.myclass.vn/api/quanlynguoidung/DangNhap`;  
let header = new Headers();  
header.append("Content-Type", "application/json;charset=UTF-8");  
let observable = this.http.post(linkApi,nguoIDung,{headers:header})  
| .pipe(map((result: Response) => { return result.json() }));  
return observable;  
}
```

❖ Demo về cơ chế token – Hướng dẫn thực hiện chức năng đặt vé

Traditional Cookie-based Authentication

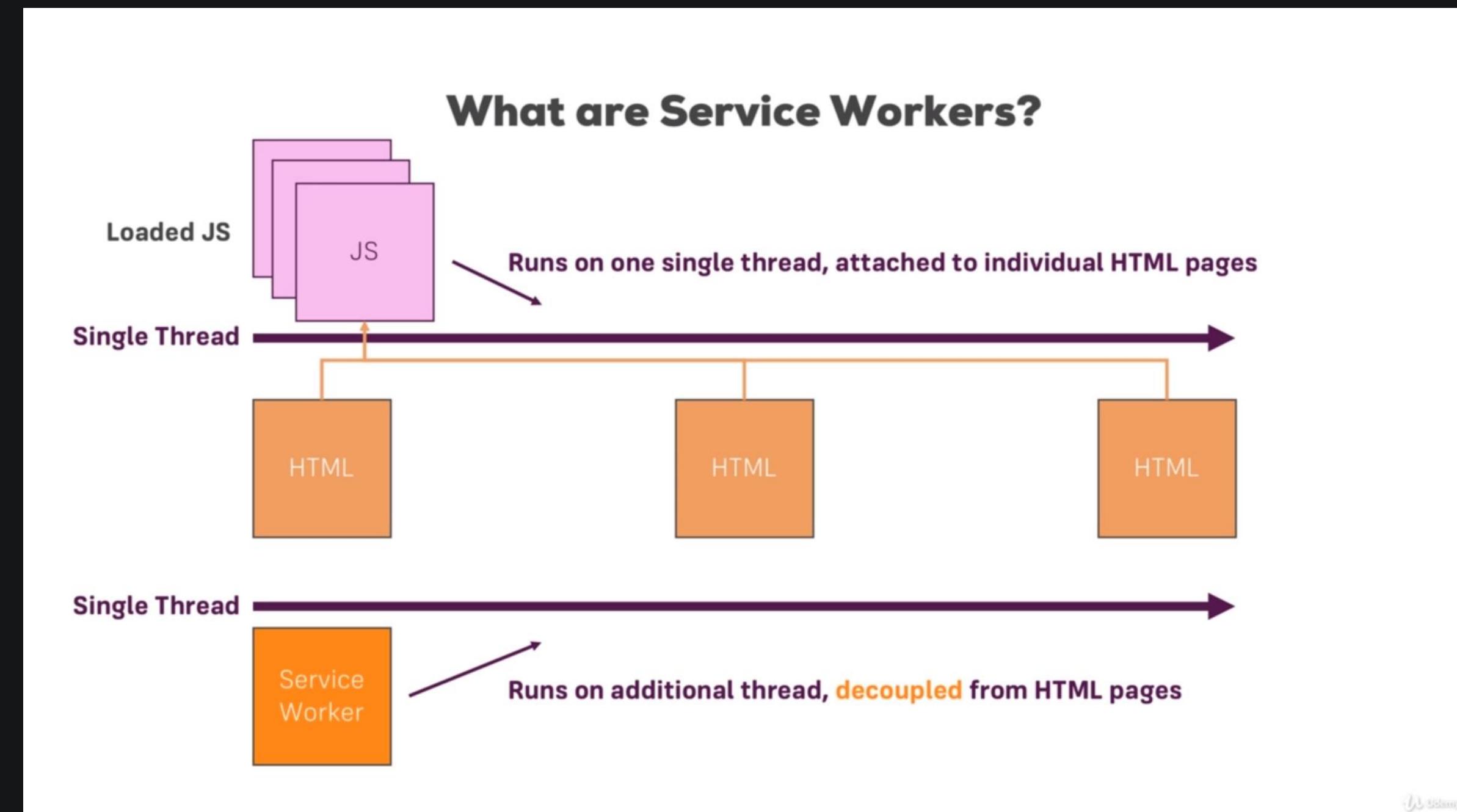


Modern Token-based Authentication



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:

```
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         "name": "assets",
17         "installMode": "lazy",
18         "updateMode": "prefetch",
19         "resources": {
20           "files": [
21             "/assets/**"
22           ]
23         }
24     }
25   }
```

```
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:

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

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

- File Menu:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- Title Bar:** app.module.ts - demoAngular - Visual Studio Code.
- Left Sidebar:** Document icon (1), Search icon, Find icon (47), and Settings icon.
- Central Area:** The code editor displays the `app.module.ts` file. The code is as follows:

```
src ▶ app ▶ TS app.module.ts ▶ AppModule
24     IndexComponent,
25     SliderComponent,
26     IndexContentComponent,
27     ItemComponent,
28
29 ],
30 imports: [
31     BrowserModule,
32
33     ServiceWorkerModule.register('ngsw-worker.js', { enabled: environment.production })
34 ],
35 providers: [],
36 bootstrap: [AppComponent]
37 }
38 export class AppModule { }
39
```
- Right Sidebar:** Includes a tree view of the project structure and a status bar at the bottom.
- Bottom Status Bar:** Shows the current branch is "master*", with 0 changes, 0 issues, and a lightning bolt icon. It also shows "Go Live", line 29, column 5, spaces: 2, UTF-8, LF, TypeScript 3.1.4, and icons for smiley face, bell, and file.

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

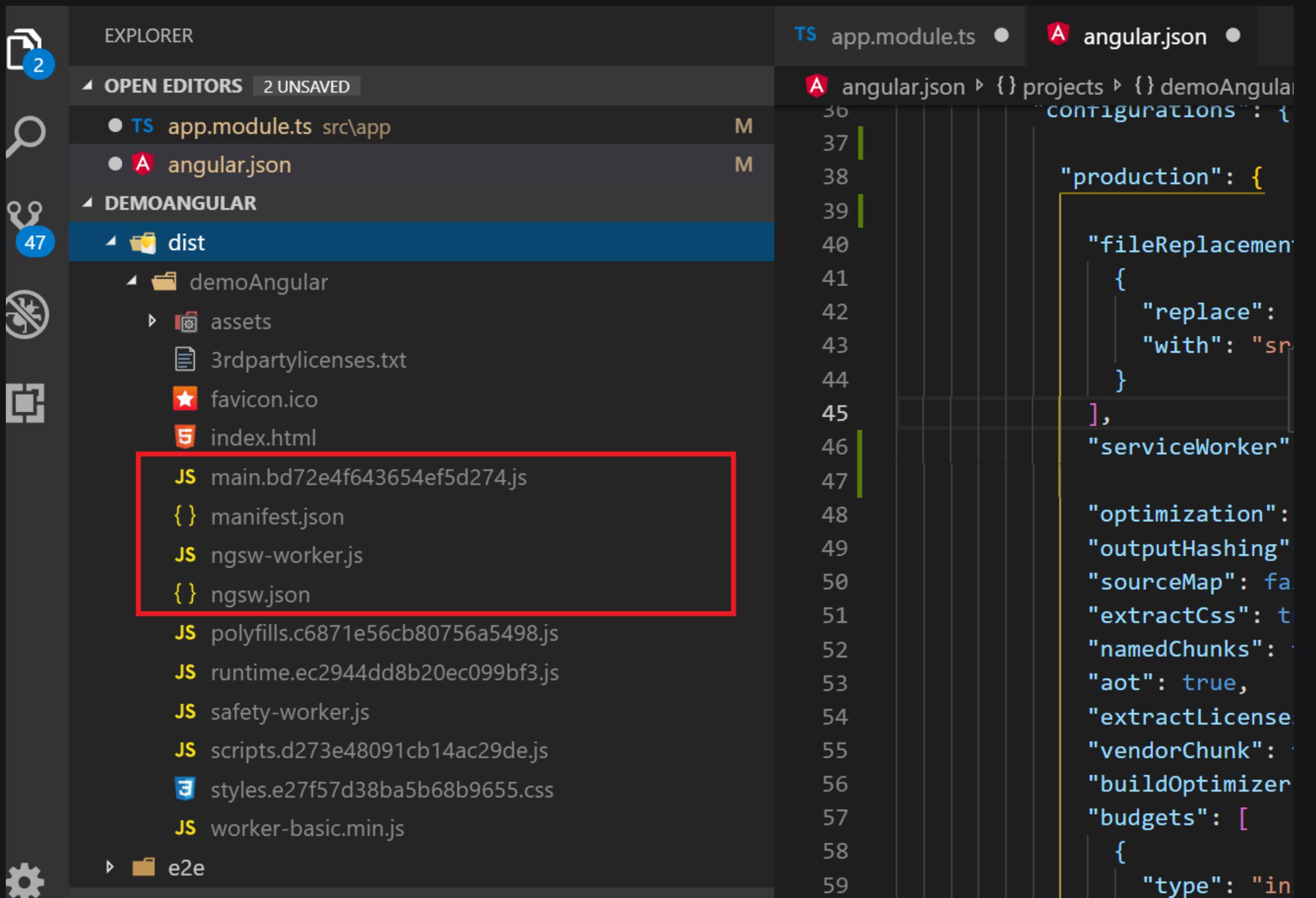
- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- Title Bar:** angular.json - demoAngular - Visual Studio Code.
- Sidebar:** Shows icons for Document (2), Search, Find in Current File, Find in Project, Find in Workspace, and Settings (47).
- Editor:** The angular.json file is open. The "production" configuration block is highlighted with a red box. The code snippet is as follows:

```
    "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.
    }
```

- Bottom Status Bar:** master*, 0 ▲ 0 ⚡, Go Live, Ln 45, Col 17, Spaces: 2, UTF-8, LF, JSON, smiley face, notification bell.
- Taskbar:** Icons for File Explorer, Search, Task List, Home, Microsoft Edge, Microsoft Store, Mail, File Explorer, Microsoft Word, Microsoft Excel, Microsoft Powerpoint, and Paint.
- System Tray:** Icons for Network, Battery, Volume, and Date/Time (11:56 AM, 11/15/2018, ENG).

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 is the Explorer sidebar with a tree view of the project structure. The 'dist' folder under 'DEMOANGULAR' is selected and highlighted in blue. Inside 'dist', there are several files and folders: 'demoAngular', 'assets', '3rdpartylicenses.txt', 'favicon.ico', and 'index.html'. Below these, four JavaScript files are listed and highlighted with a red box: 'main.bd72e4f643654ef5d274.js', 'manifest.json', 'ngsw-worker.js', and 'ngsw.json'. In the center-right is the code editor showing the 'angular.json' file. The 'production' configuration section is expanded, showing settings for file replacement and service workers.

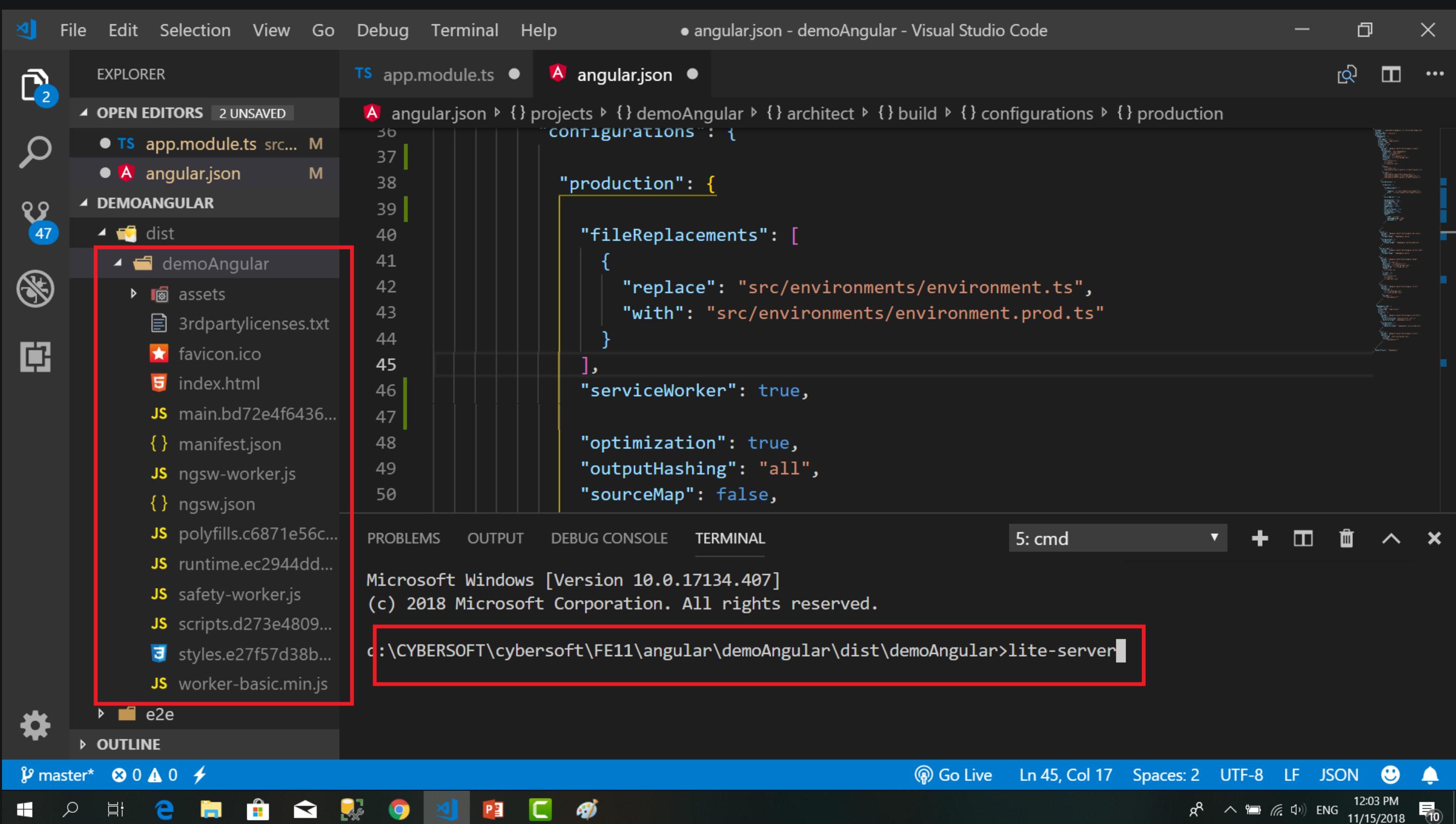
```

{
  "projects": {
    "demoAngular": {
      "configurations": {
        "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": [
    {
      "type": "initial"
    }
  ]
}

```

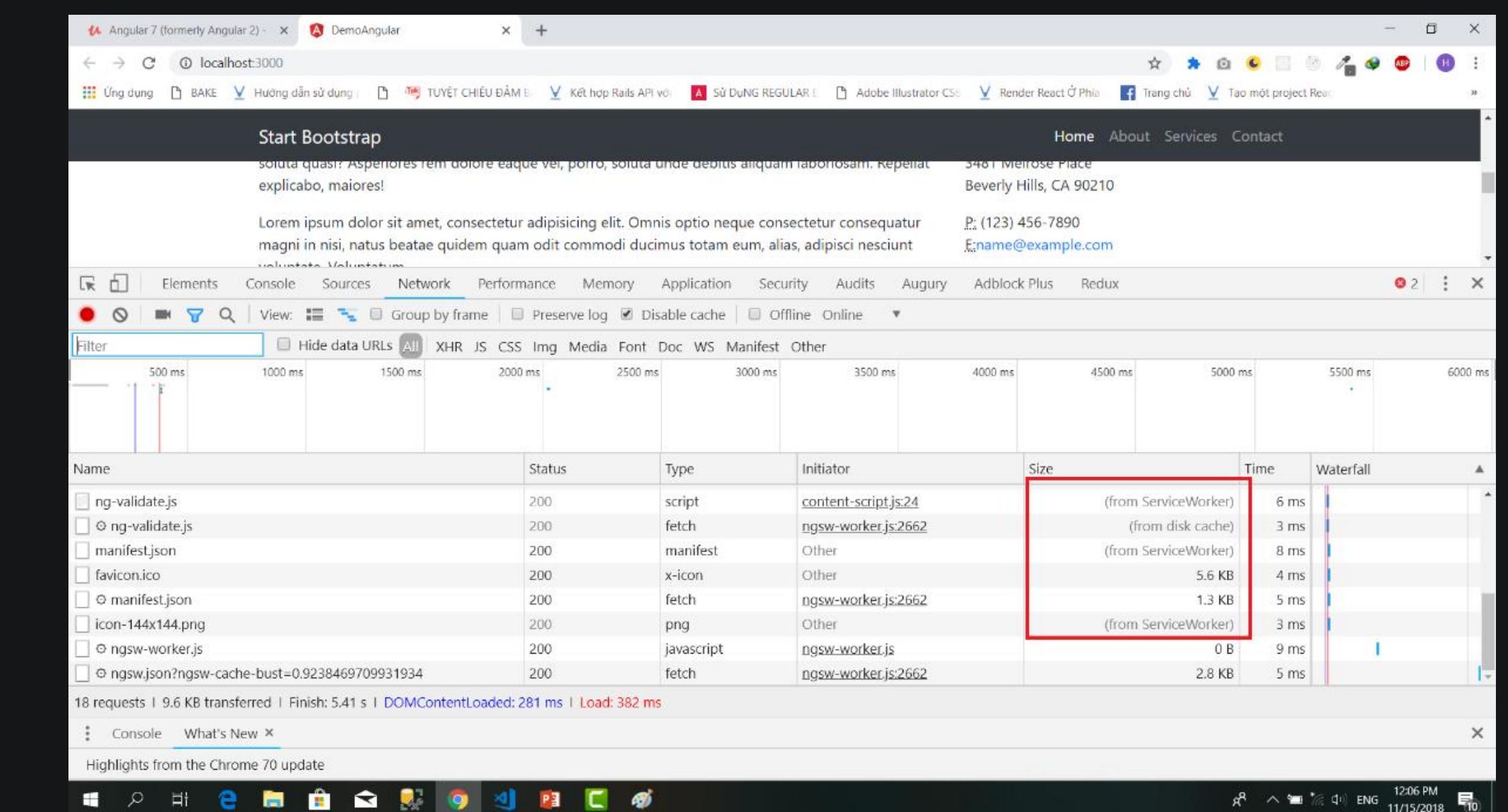
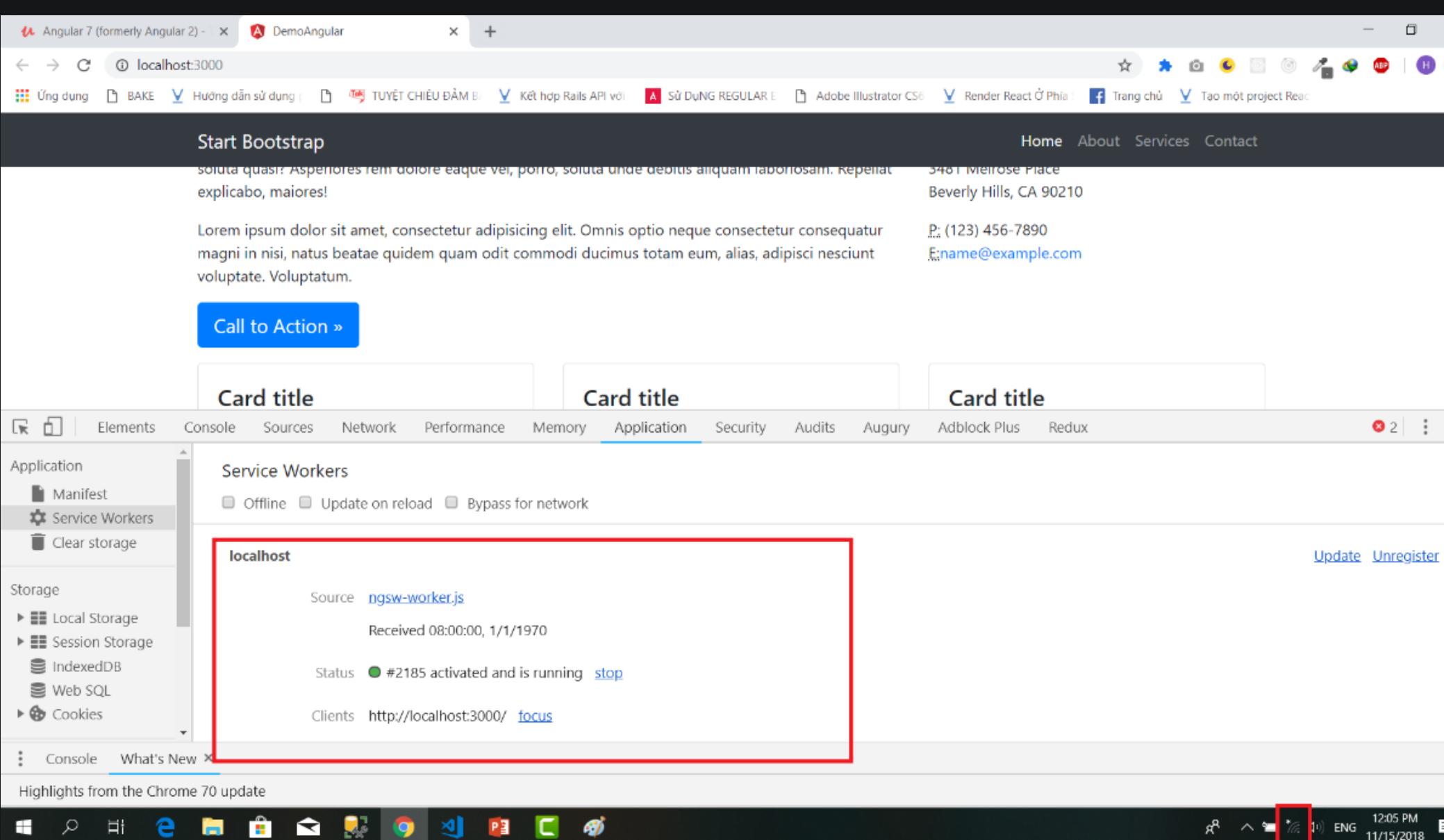
Sử dụng service worker?

- 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



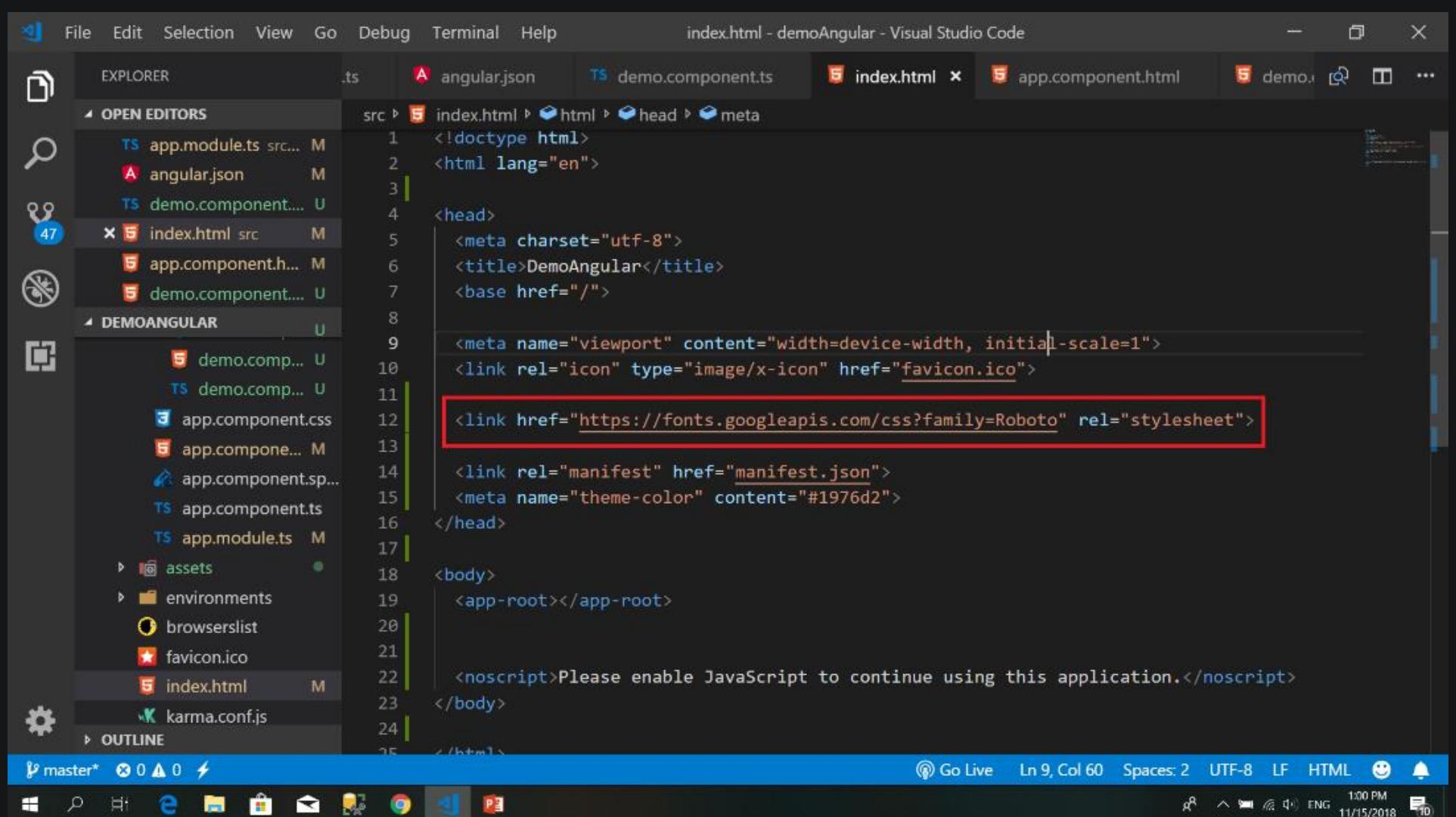
Sử dụng service worker?

- 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



Sử dụng service worker?

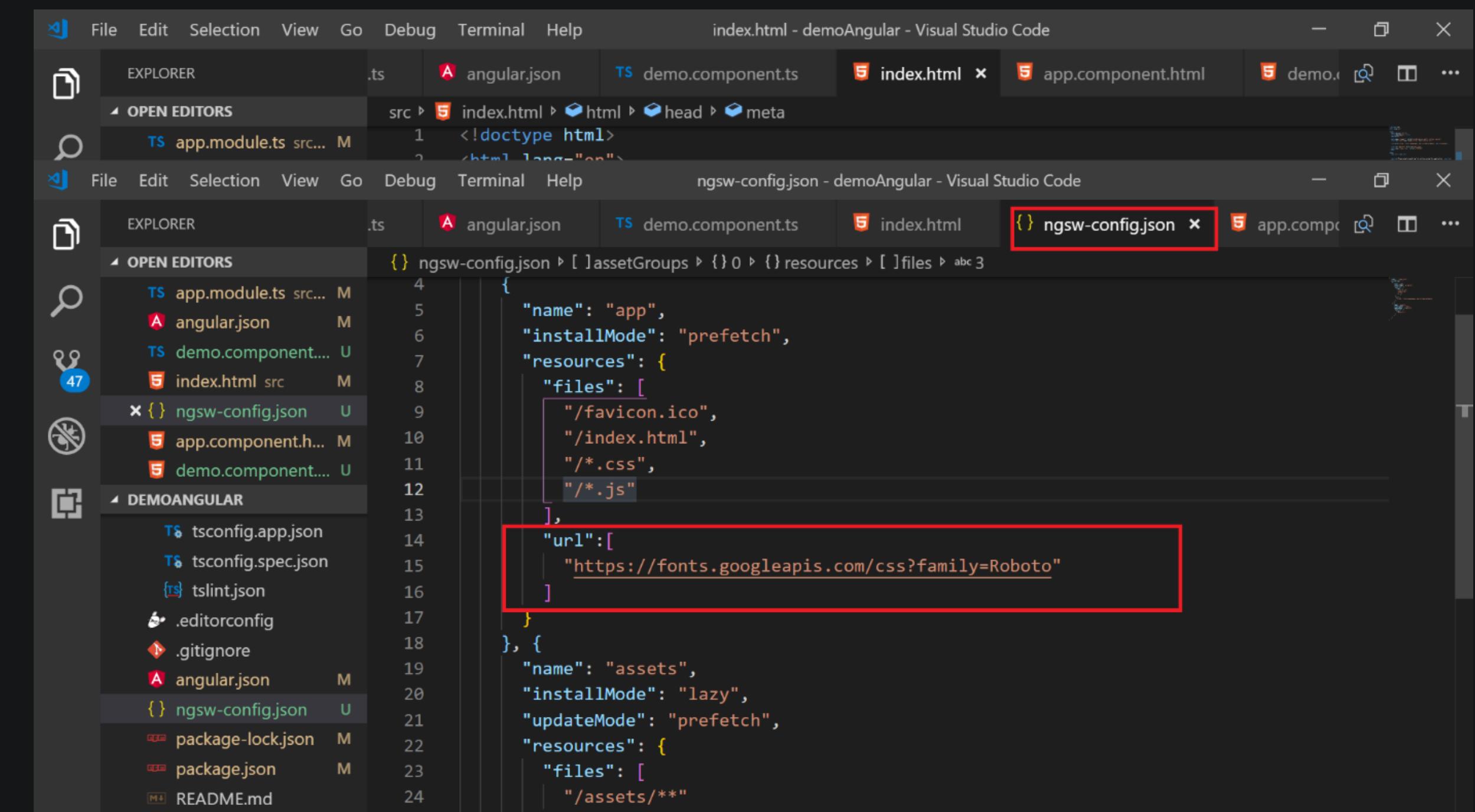
Nếu sử dụng cdn ở index.html, chẳng hạn như sử dụng font từ google font, ta cần thêm một vài set up



```

<!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">
    <link rel="manifest" href="manifest.json">
    <meta name="theme-color" content="#1976d2">
</head>
<body>
    <app-root></app-root>
    <noscript>Please enable JavaScript to continue using this application.</noscript>
</body>

```



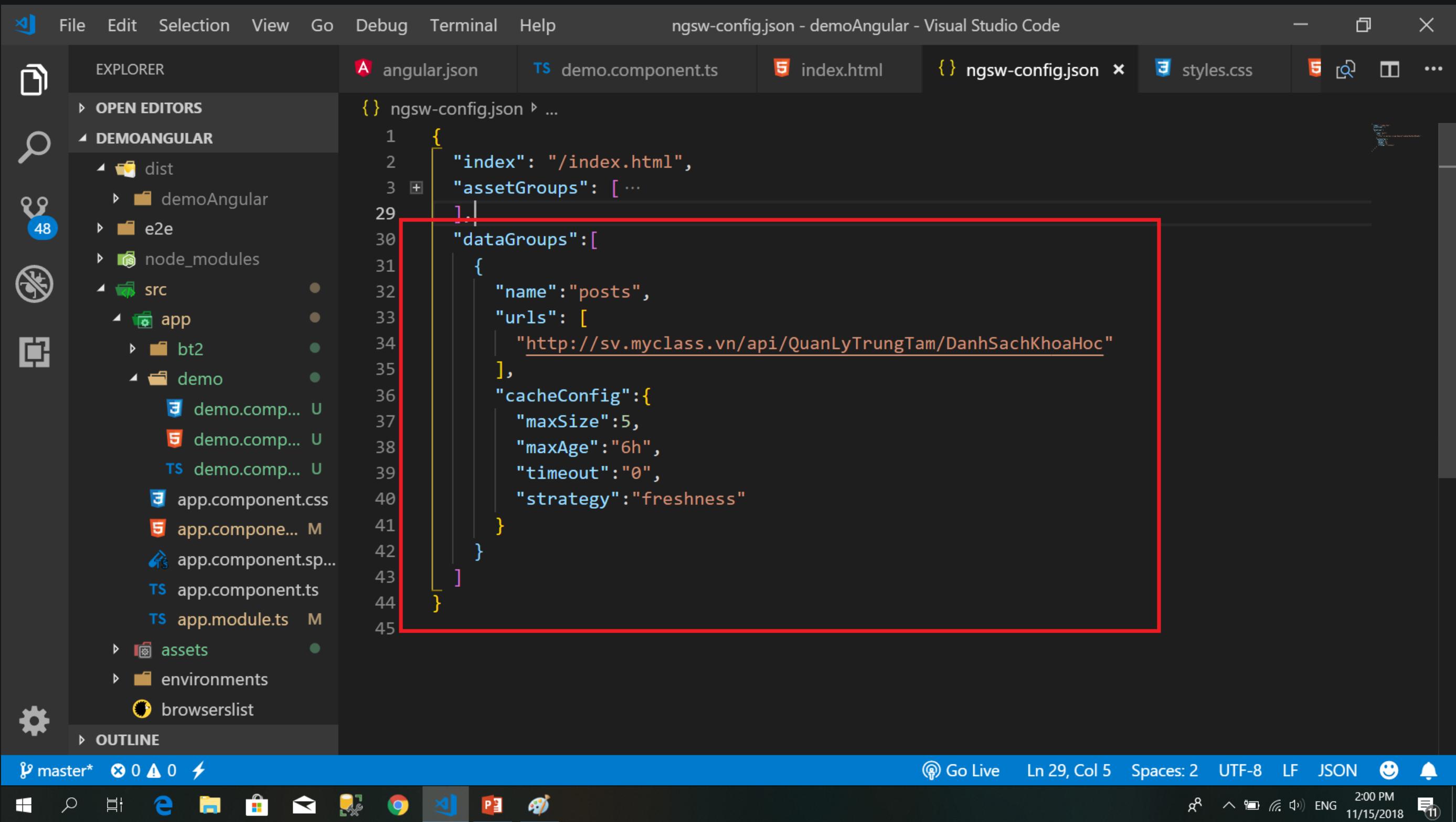
```

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

```

Sử dụng service worker?

- Load dữ liệu động với service worker:
 - Thêm một group vào file ngsw-config.json như sau



```
{} ngsw-config.json > ...
1  {
2    "index": "/index.html",
3    "assetGroups": [ ...
4    ],
5    "dataGroups": [
6      {
7        "name": "posts",
8        "urls": [
9          "http://sv.myclass.vn/api/QuanLyTrungTam/DanhSachKhoaHoc"
10         ],
11        "cacheConfig": {
12          "maxSize": 5,
13          "maxAge": "6h",
14          "timeout": "0",
15          "strategy": "freshness"
16        }
17      }
18    ]
19  }
```

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