**Một số lệnh trong Angular Cli**

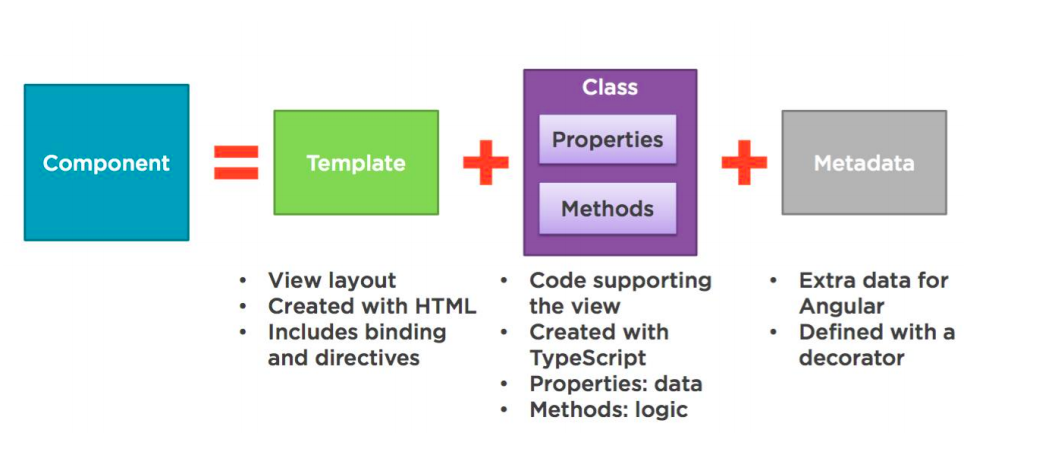
|  |  |
| --- | --- |
| **Lệnh** | **Mô tả** |
| ng g component new-component | tạo component |
| ng g directive new-directive | tạo directive |
| ng g pipe new-pipe | tạo pipe |
| ng g service new-service | tạo service |
| ng g class new-class | tạo class |
| ng g guard new-guard | tạo guard |
| ng g interface new-interface | tạo interface |
| ng g enum new-enum | tạo enum |
| ng g module my-module | tạo module |
| ng test | test |
| ng build --configuration=production | thiết lập môi trường production |
| ng build --configuration=staging | thiết lập môi trường staging |
| ng serve --port 3006 --open | khởi động dự án với cổng 3006 |

**Component**

- Component cơ bản là các lớp tương tác với temple HTML, giúp tạo ra các thẻ tag html do mình tự định nghĩa ra.

- Một component thành lập bởi 3 thành phần: code html gọi là temple, một class typescript xử lí dữ liệu và tương tác với với temple, cuối cùng là CSS.

- Một dự án có nhiều component



**Data Binding**

**- Compnent to View**

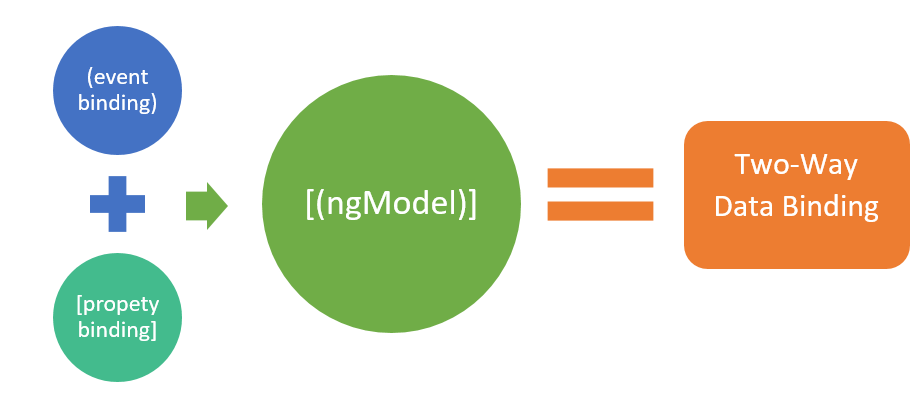
|  |  |  |
| --- | --- | --- |
| Tên | Cú pháp | Mô tả |
| Interpolation binding | {{value}} |  |
| Property binding | [src]=”value” |  |
| Class binding (class) | [class.class\_name]=”value” |  |
| Style binding (style) | [style.color]=”’red’” |  |

**- View to Component**

|  |  |  |
| --- | --- | --- |
| Tên | Cú pháp | Mô tả |
| Event binding | (event-name)=”function()” | click, enter, change,... |

**- To way binding**

|  |  |  |
| --- | --- | --- |
| Tên | Cú pháp | Mô tả |
| To way binding | [(ngModel)]=”value” | đồng bộ giữa view và component, cần phải import FormsModule |



**Directive** là một đối tượng giúp chúng ta dễ dàng thay đổi một đối tượng khác và cách áp dụng rất đơn giản và linh hoạt. Là một phần mở rộng được Angular định nghĩa thêm

**Component Directive**

- <component-name></component-name>

**Structural Directive**

- ngIf

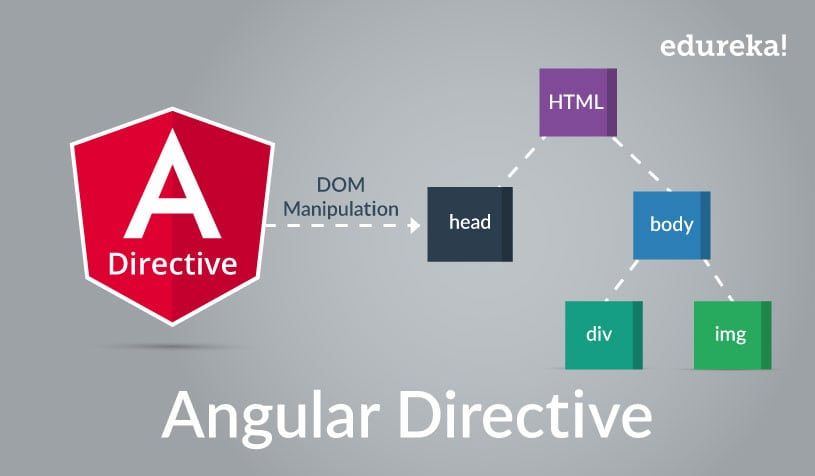
- ngFor

- ngSwitch

**Attribute Directive**

- ngClass

- ngStyle



**Component Interaction**

- @Input()

+ cần import Input từ @angular/core

+ truyền dữ liệu từ component cha vào component con

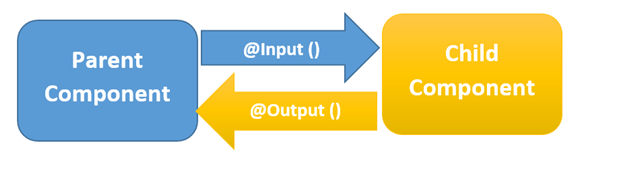
+ cú pháp:

- @Output()

+ cần import Output, EventEmitter từ @angular/core

+ truyền dữ liệu từ component con ra ngoài component cha

+ cú pháp:



**Pipe**

- Dùng để biến đổi dữ liệu trước khi hiển thị ra cho người dùng

- Nhận vào 1 giá trị và trả ra ngoài 1 giá trị khác

- Cần import @angular/common

- Các pipe có sẵn trong angular:

+ Lowercase pipe: toàn bộ in thường

+ Uppercase pipe: toàn bộ in hoa

+ Date pipe: ngày tháng

+ Currency pipe: tiền tệ

+ Json pipe: json

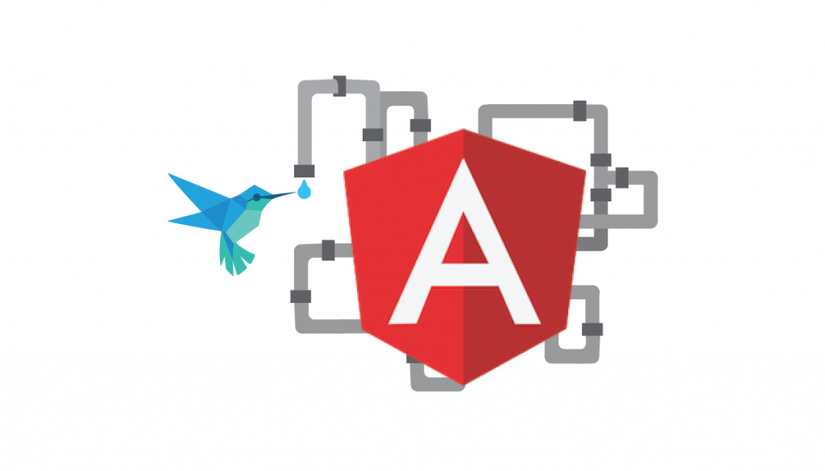
+ TitleCase: viết hoa chữ đầu của mỗi từ

+ Percent pipe: sử lí phần trăm

+ Decimal pipe: sử lí số thực

+ Slice pipe: cắt ký tự

- Có thể kết hợp các pipe lại với nhau và custom pipe



**Template Reference Variables**

- Một TRV là một tham chiếu đến một phần tử DOM hoặc directive trong template. Sử dụng biến tham chiếu, ta có thể truy cập vào các giá trị thuộc tính phần tử DOM

- Để thao tác với Template variable có 2 cách:

+ tại template: #name hoặc ref-name

+ tại component: @ViewChild(“name”) name2: ElementRef

- [@ViewChild](https://viblo.asia/u/ViewChild" \t "/home/sonnguyen/Documents\\x/_blank) decorator không thể nhìn qua 1 component bị bao đóng. Điều này có nghĩa là ViewChild chỉ có thể xem được thành phần nằm bên trong template của chính nó

- [@ViewChild](https://viblo.asia/u/ViewChild" \t "/home/sonnguyen/Documents\\x/_blank) để truy vấn tới DOM element

- [@ViewChild](https://viblo.asia/u/ViewChild" \t "/home/sonnguyen/Documents\\x/_blank) để truy vấn tới DOM component

- [@ViewChild](https://viblo.asia/u/ViewChild" \t "/home/sonnguyen/Documents\\x/_blank)  với thuộc tính metadata

- [@ViewChild](https://viblo.asia/u/ViewChild" \t "/home/sonnguyen/Documents\\x/_blank) truy vấn directive

**Life Cycle**

- Contructor: Được gọi trước tất cả các lifecycle thường dùng để tiêm các DI như các service. Đây không phải là một hook method

- ngOnInit:

+ Chỉ gọi một lần duy nhất sau khi ngOnChanges() được gọi lần đầu tiên

+ Dùng để khởi tạo các giá trị

- ngOnDestroy:

+ Được gọi khi component bị hủy, dùng để hủy các kết nối và giải phóng bộ nhớ

- ngOnChanges:

+ Đươc thực hiện khi @Input có sự thay đổi

+ Gọi trước ngOnInit

+ Cho ta một đối tượng kiểu SimpleChanges

- ngDoCheck

+ Gọi mỗi lần khi nhưng thứ có liên quan đến @Input thay đổi

+ Được gọi sau ngOnChanges và ngOnInit

- ngContent

- ngAfterContentInit

+ Sử dụng ng-content để kiểm tra, sử dụng @ContentChild (kiểu ElementRef) - Template Reference Variable

+ Chỉ gọi 1 lần duy nhất khi ngContent được tạo

- ngAfterContentChecked:

+ Được gọi nhiều lần, khi giá trị của ngContent hoặc những giá trị liên quan đến @ContentChild có sự thay đổi

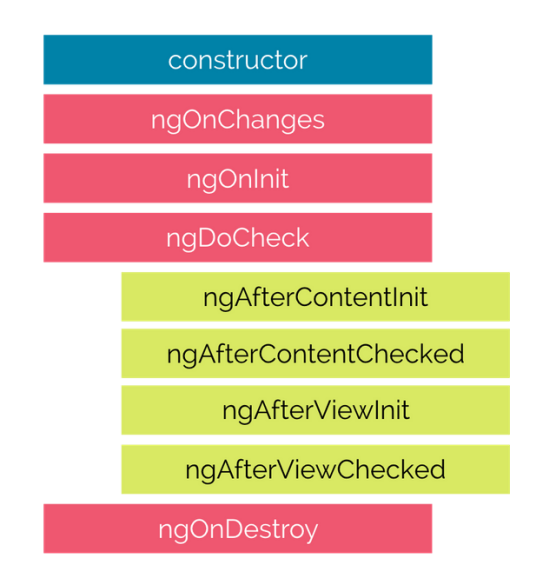
- ngAfterViewInit:

+ Được gọi sau khi angular khởi tạo view của component và các viewchild

+ xử lí template + template reference variable

+ Sử dụng @ViewChild

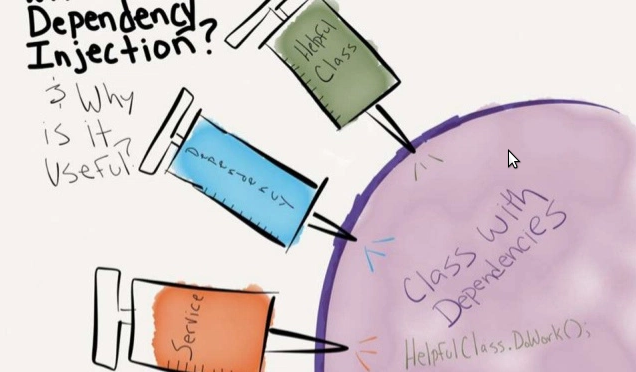
- ngAfterViewChecked:

+ Được gọi nhiều lần, khi trên view có sự thay đổi

**Service**

- Service trong angular là một class đóng gói một vài chức năng và cung cấp nó cho phần còn lại của ứng dụng

- Dependency Injection là chức năng cho phép chúng ta “nhúng” các lớp vào các lớp khác, giống như dùng một thư viện vậy, và chúng ta có thể dùng các lớp được nhúng vào đó giống như dùng một thuộc tính bình thường mà không cần phải thực hiện các công đoạn khai báo, khởi tạo…v.v



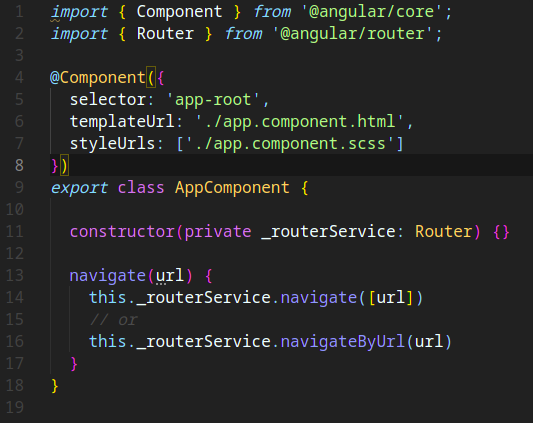
**Router**

1. **Chuyển trang**

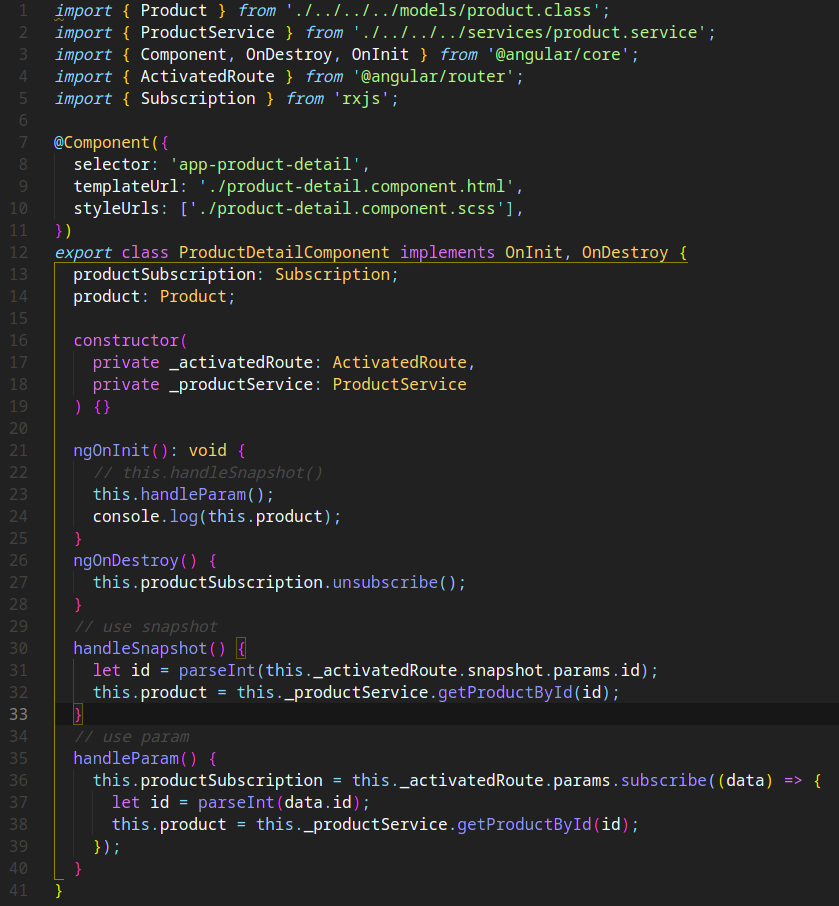
- Chuyển trang bằng template



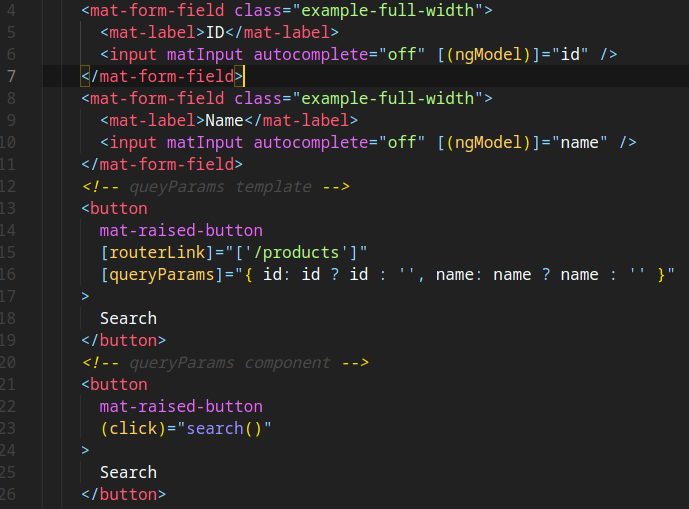
- Chuyển trang bằng component:

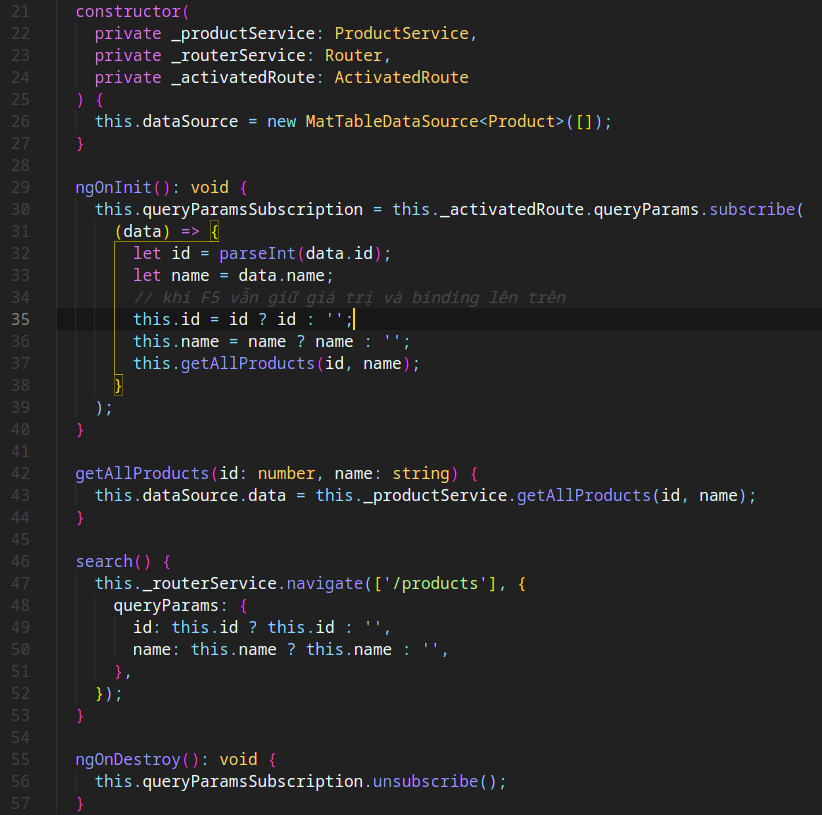


1. **Nhận tham số trên URL (snapshot - params)**

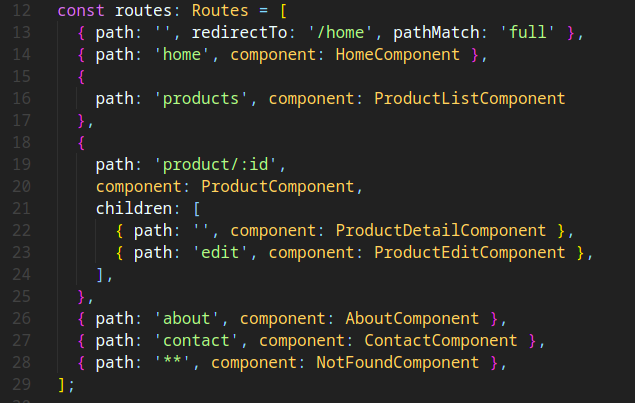


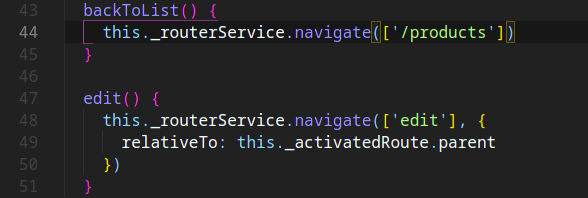
**3, Nhận tham số trên URL (Query param)**

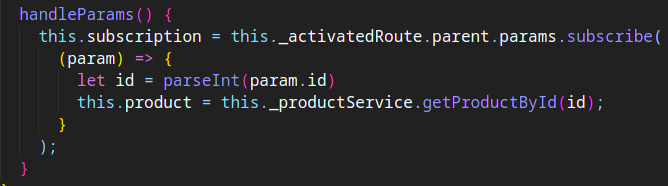




**4. Child Router**

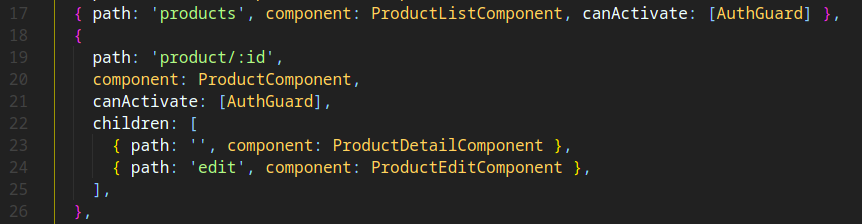


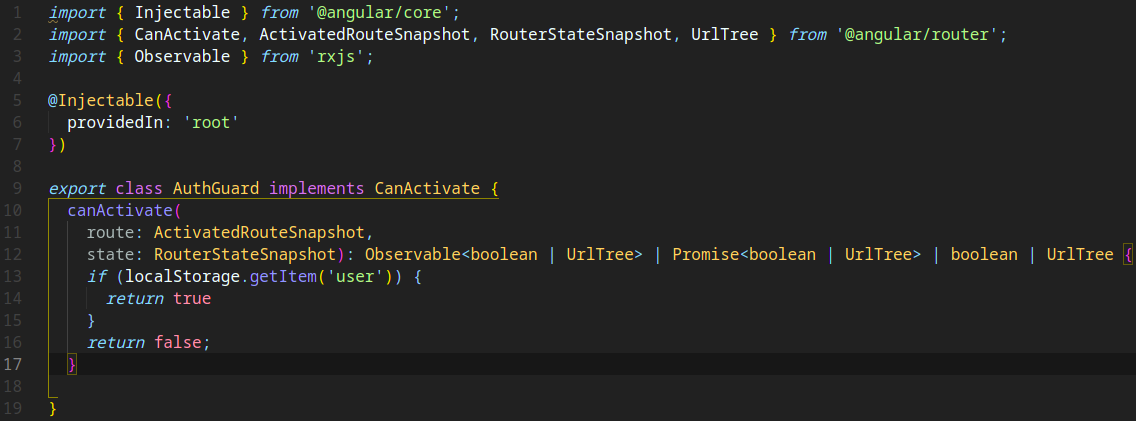




1. **CanActive**

- Là cách để **chặn** người dùng truy cập vào một router nào đó

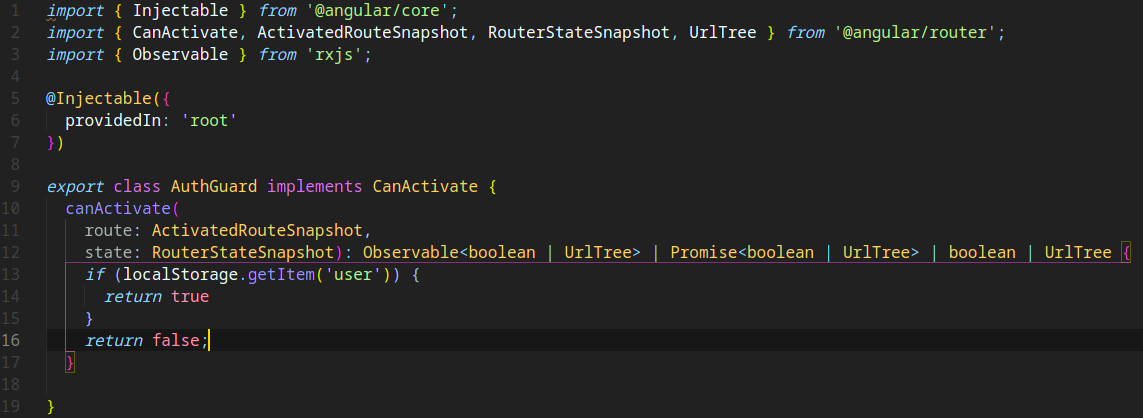




1. **CanDeActive**

- Là cách để không cho người dùng **thoát** khỏi 1 router nào đó





**Module**

- Module chức năng

- Module share

**HTTP Client**