# I. Binding:

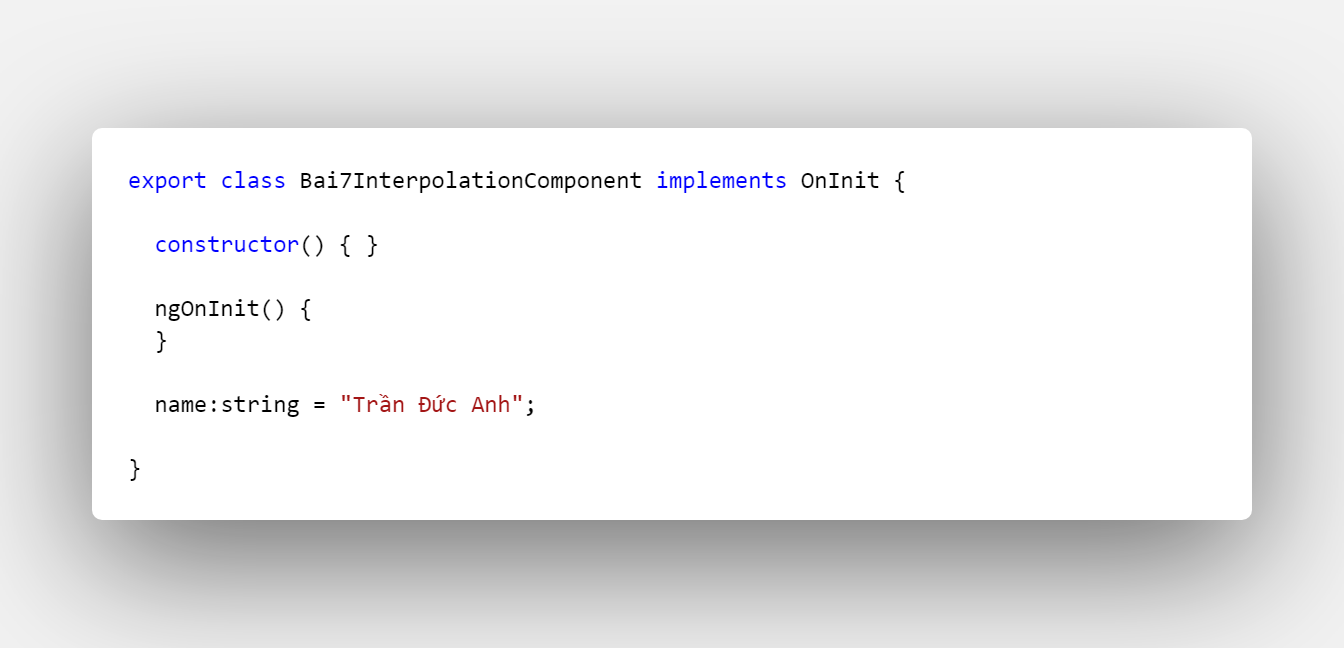
## 1. One way binding:

### a. Interpolation:

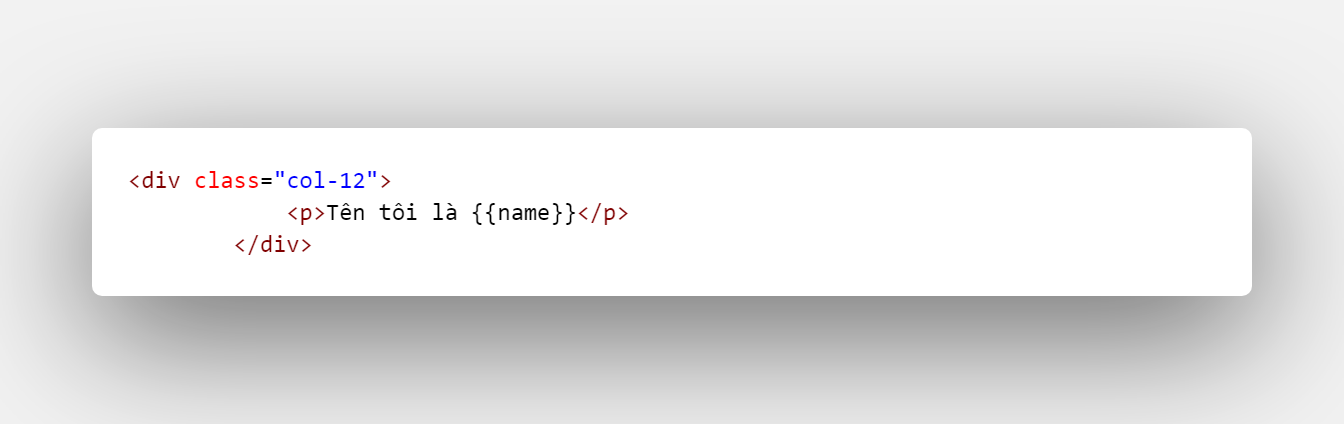
Interpolation là cơ chế buộc dữ liệu một chiều, dùng để hiển thị dữ liệu từ Model lên View.

*\* Cách sử dụng:*

- Bước 1: Trong file component.ts khai báo biến.



- Bước 2: Tại file component.html, sử dụng interpolation để lấy ra giá trị của biến đó.



### b. Property Binding:

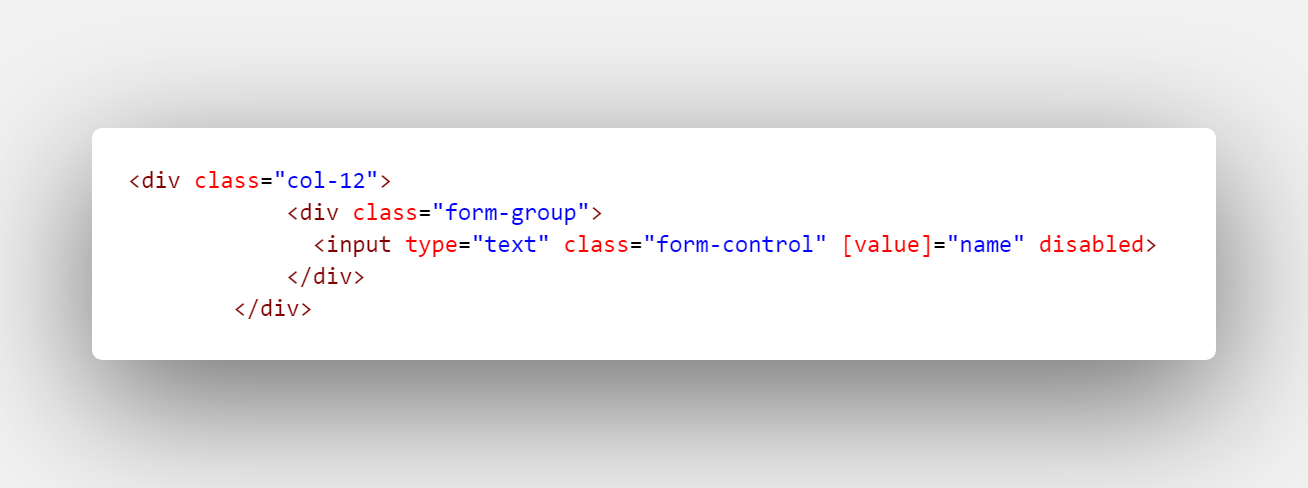
Property binding cũng là cơ chế buộc dữ liệu 1 chiều nhưng thông qua attributes của thẻ html.

*\* Cách sử dụng:*

- Bước 1: Trong component.ts khai báo biến chứa dữ liệu muốn chuyển sang file html.



- Bước 2: Trong thẻ html, sử dụng [attr-name]="var-name".

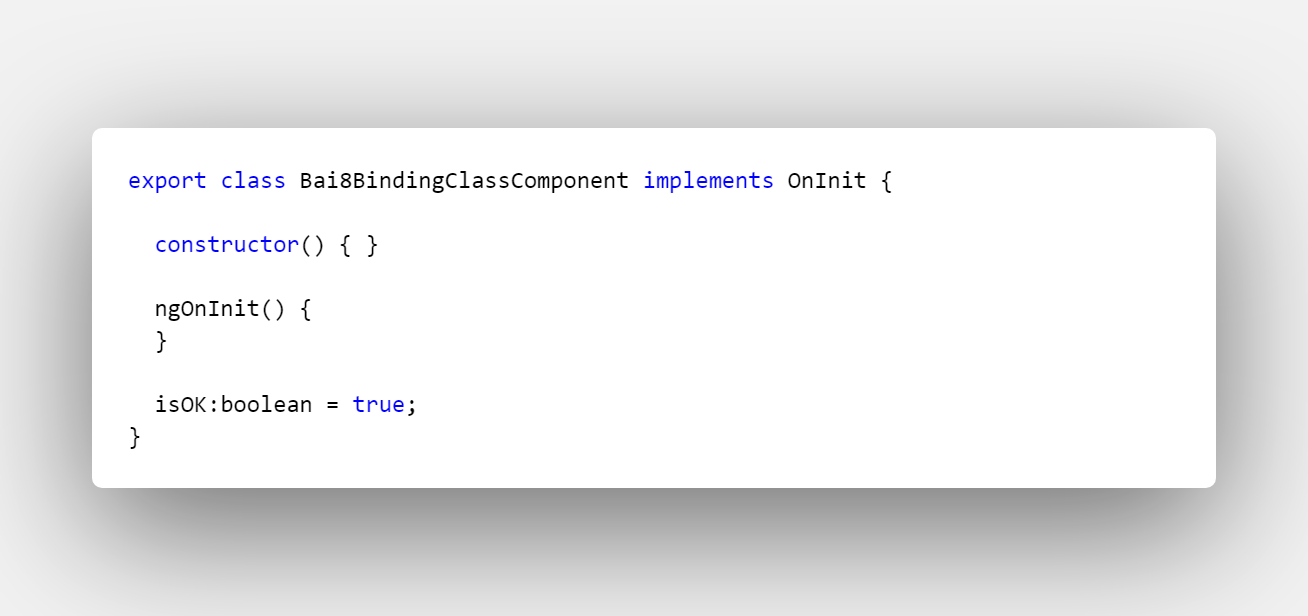


### c. Binding Class:

Binding class sử dụng để áp dụng class cho thẻ html theo điều kiện.

*\* Cách sử dụng:*

- Bước 1: Khai báo biến điều kiện trong file ts.



- Bước 2: Sử dụng [class.class-name]="var-name". Lưu ý: tại đây có thể sử dụng phủ định !var-name.



### d. Binding Style:

Binding style dùng để gắn style đã khai báo thành các biến vào thẻ html.

*\* Cách sử dụng:*

- Bước 1: Khai báo biến điều kiện và biến chứa dữ liệu style.



- Bước 2: Sử dụng [style.style-name]="var-condition?var-style-if-true:var-style-if-false" (Toán tử 3 ngôi).

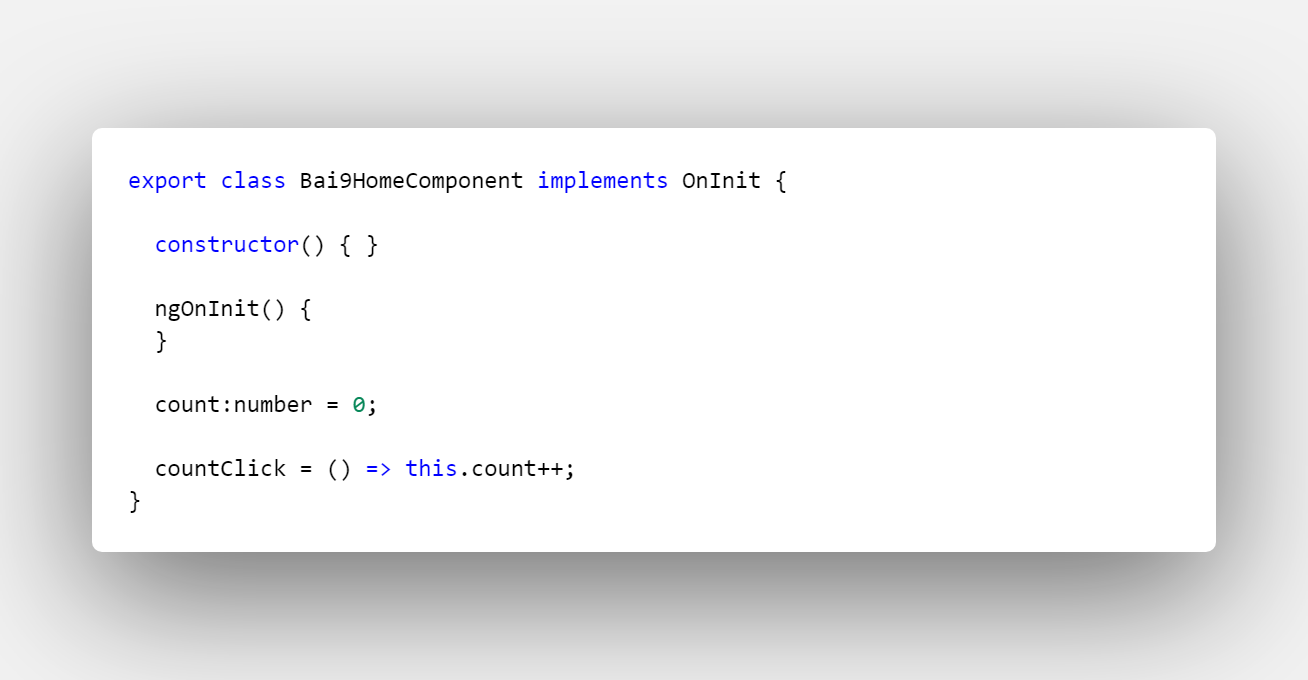


### e. Event binding:

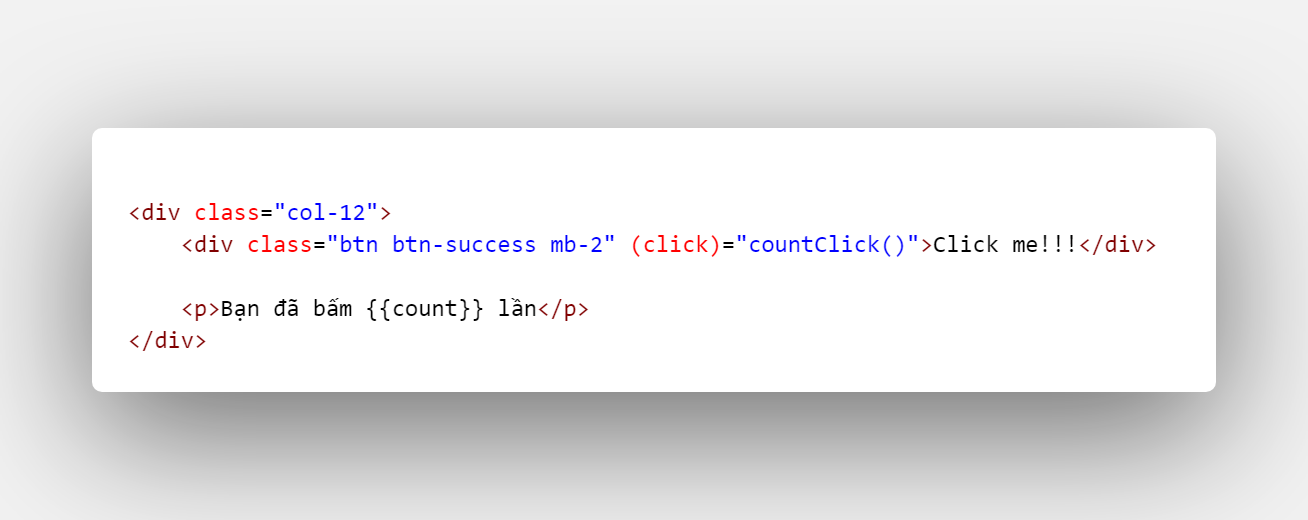
Event binding sử dụng để gọi các method theo sự kiện tương tác vào thẻ html.

*\* Cách sử dụng:*

- Bước 1: Trong file ts, tạo method chức năng.



- Bước 2: Trong thẻ html, sử dụng (event-name)="method-name()".

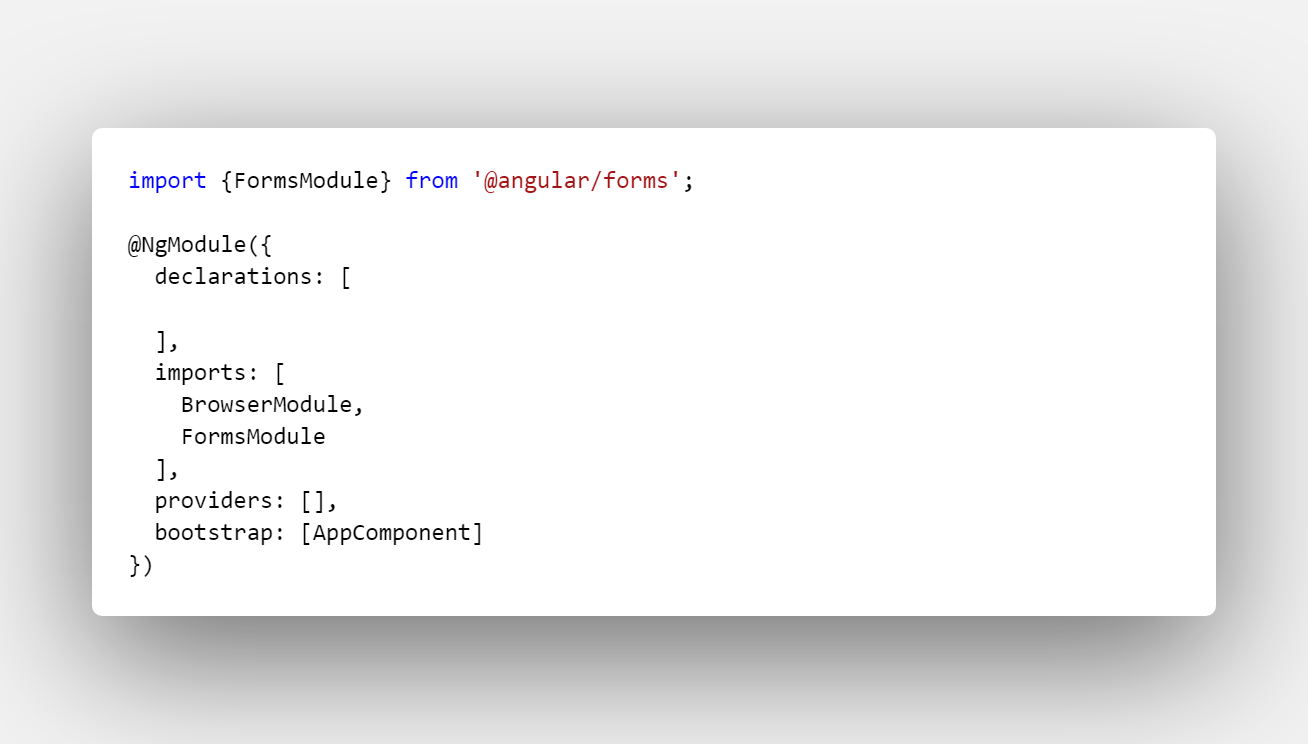


## 2. Two way binding:

Two way binding là cơ chế buộc dữ liệu 2 chiều. Khi dữ liệu ở View thay đổi thì dữ liệu ở Model cũng thay đổi theo và ngược lại.

*\* Cách sử dụng:*

- Bước 1: Trong app.module.ts khai báo FormsModule.



- Bước 2: Nếu muốn dùng dữ liệu trong code logic, khai báo biến var-name chứa dữ liệu trong file ts.



- Bước 3: Trong thẻ nhập dữ liệu vào, sử dụng [(ngModel)]="var-name".



- Bước 4: Sử dụng interpolation để hiển thị dữ liệu.



# II. Directive:

## 1. Structural directive:

Structural directive là những thẻ chỉ thị làm thay đổi cấu trúc DOM ví dụ như ẩn hiện.

### a. ngIf:

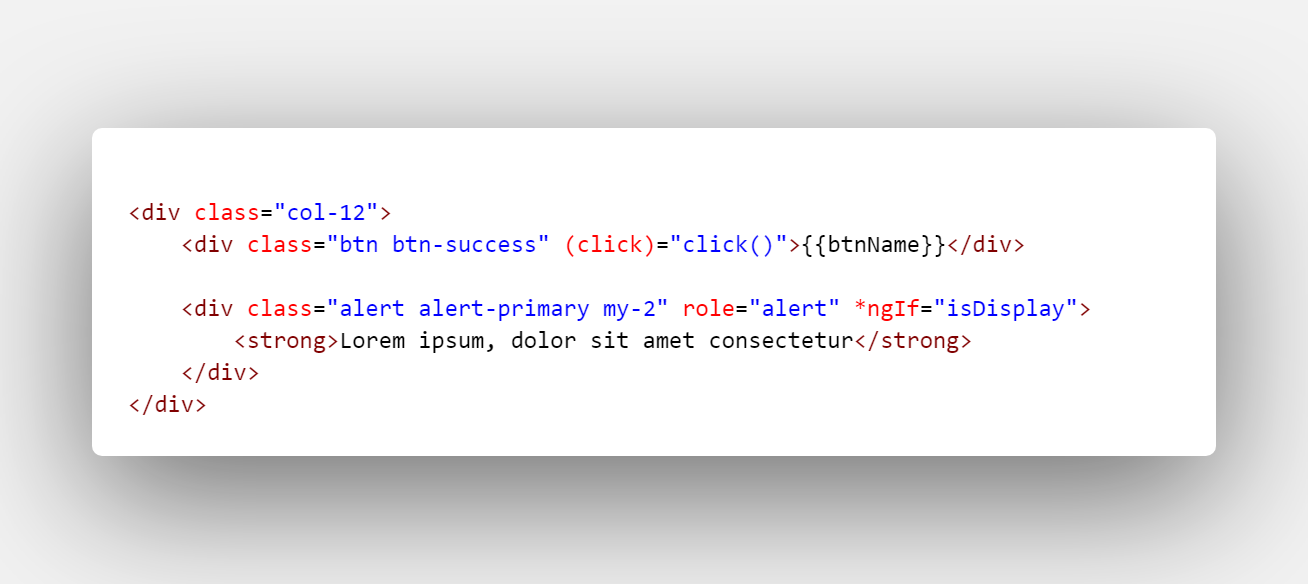
ngIf dùng để ẩn hoặc hiện thẻ theo điều kiện.

*\* Cách sử dụng:*

- Bước 1: Trong file ts tạo 1 biến làm điều kiện.



- Bước 2: Trong thẻ html, sử dụng \*ngIf="condition-var". *(condition-var cũng có thể là 1 biểu thức điều kiện)*. Thẻ sẽ hiện ra nếu điều kiện đúng và ẩn đi nếu điều kiện sai.



### b. ngSwitch:

ngSwitch dùng để hiển thị các thẻ con của nó theo điều kiện đã được định sẵn.

*\* Cách sử dụng:*

- Bước 1: Tạo biến điều kiện trong file ts.



- Bước 2: Trong thẻ cha, sử dụng [ngSwitch]='conditon-var'.



- Bước 3: Trong thẻ con, sử dụng \*ngSwitchCase="condition-value".



- Lưu ý: Nếu giá trị của điều kiện là chuỗi thì trong \*ngSwitchCase, giá trị phải đặt thêm trong cặp nháy đơn.

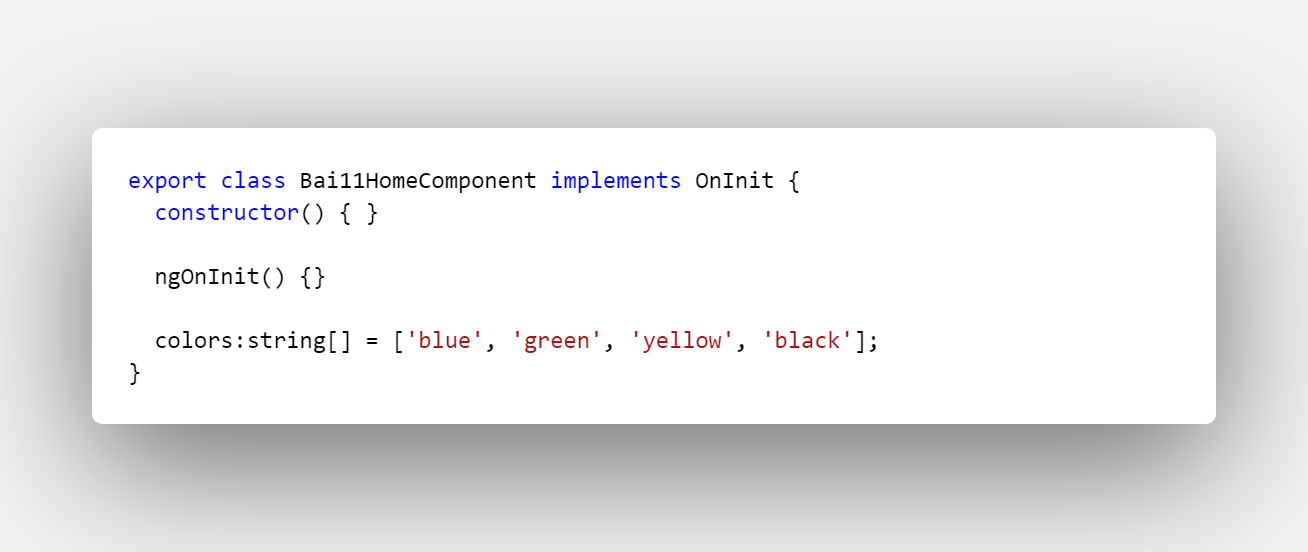


### c. ngFor:

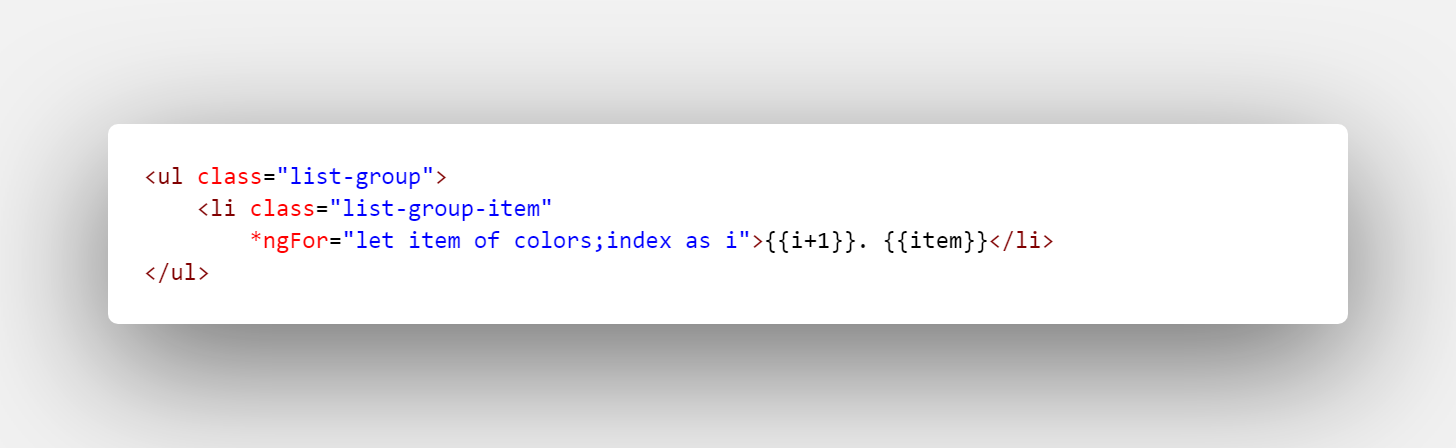
ngFor được sử dụng để lặp thẻ chứa nó.

*\* Cách sử dụng:*

- Bước 1: Trong file ts, tạo biến chứa dữ liệu. Ví dụ 1 mảng.



- Bước 2: Trong thẻ html muốn lặp, sử dụng \*ngFor="let item of arrs;index as i". Với item là đối tượng hiện tại, arrs là biến chứa mảng dữ liệu, gán chỉ số của item trong arrs vào i.



## 2. Attributes directive:

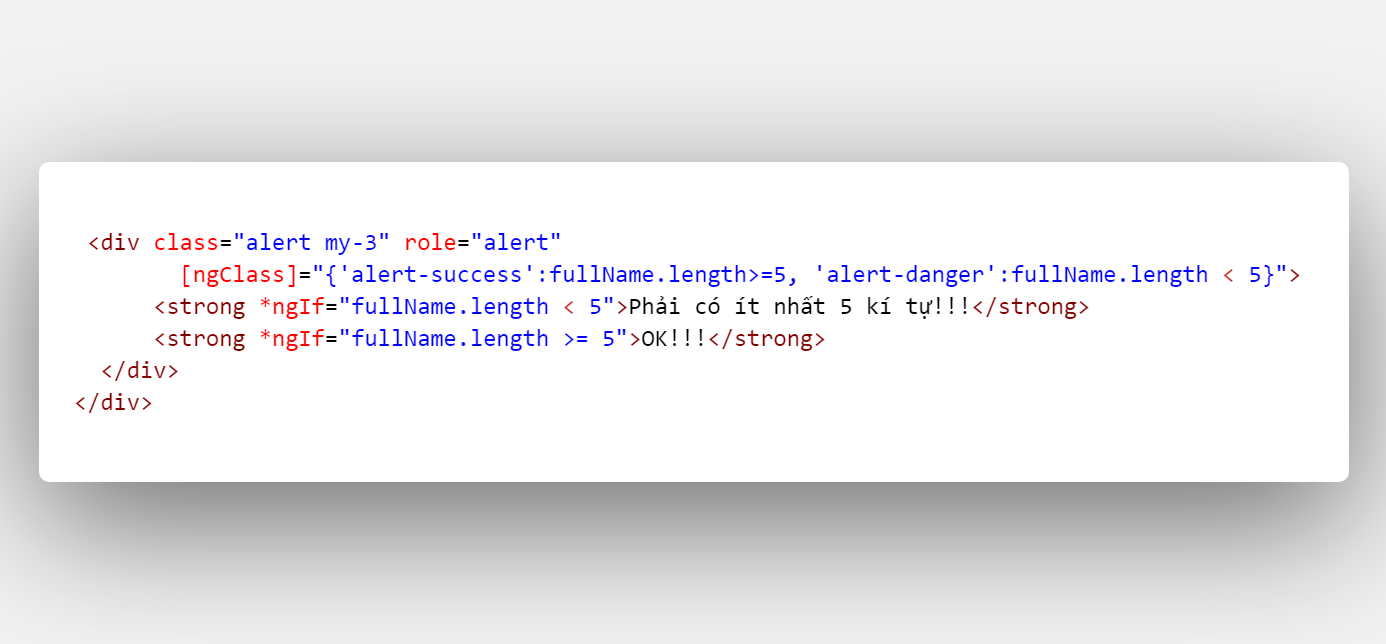
Attribute Directive là những chỉ thị làm thay đổi style của thẻ html. Đối với các structural directive (ngoại trừ ngSwitch) thì sẽ dùng kí tự \* trước directive. Còn attribute directive thì sẽ dùng [ ] lồng ra ngoài directive.

### a. ngClass:

ngClass dùng để thêm class vào thẻ html nếu thỏa mãn điều kiện.

*\* Cách sử dụng:*

Trong thẻ html, sử dụng [ngClass]="'class-name1':condition1"



Lưu ý:

- Condition có thể là biến điều kiện được tạo sẵn trong ts hoặc là biểu thức điều kiện.

- Có thể sử dụng nhiều cặp class-condition

- Nếu có nhiều cặp class-condition thì phải đặt trong cặp ngoặc móc.

### b. ngStyle:

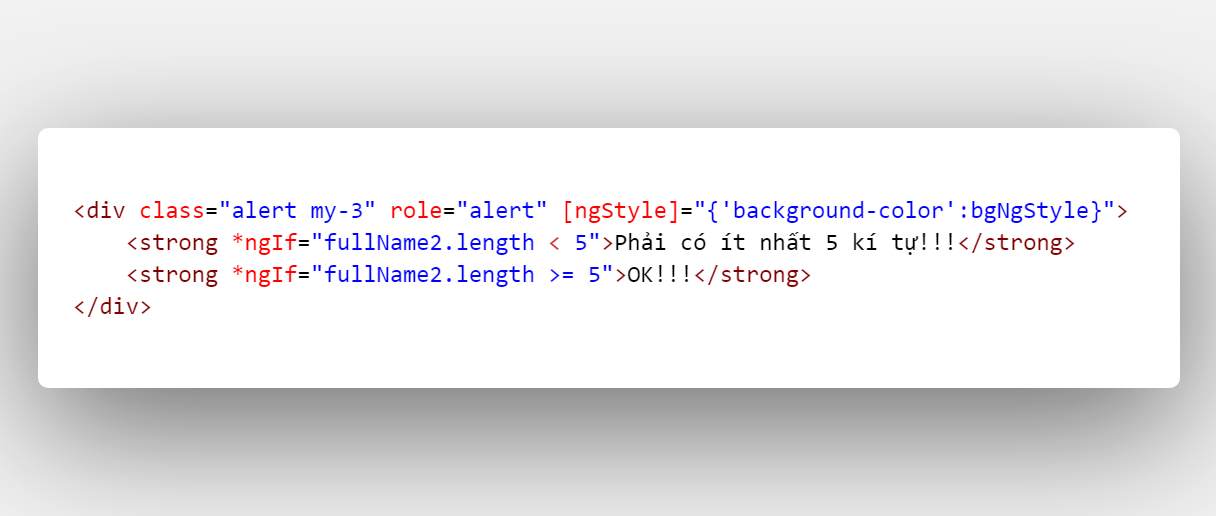
ngStyle dùng để áp dụng những style cho thẻ html theo giá trị được khai báo trong file ts.

*\* Cách sử dụng:*

- Bước 1: Trong file ts, khai báo biến chứa giá trị cho style.



- Bước 2: Trong thẻ html sử dụng [ngStyle].



# III. Tương tác giữa các component:

## 1. Input:

@Input dùng để chuyền giá trị từ component cha vào component con.

*\* Cách sử dụng:*

- Bước 1: Trong directive của component con tại component cha, sử dụng [key] = "value".



- Bước 2: Trong component-child.ts:

+ Import Input.

+ Khai báo @Input('key') varName:DataType;



Lưu ý:

- Nếu không muốn khai báo key, ta có thể sử dụng key mặc định là varName khi khai báo @Input tại component-child.ts.

- Nếu value là chuỗi thì phải lồng vào trong cặp nháy đơn [varNameInput] = " 'value' ".

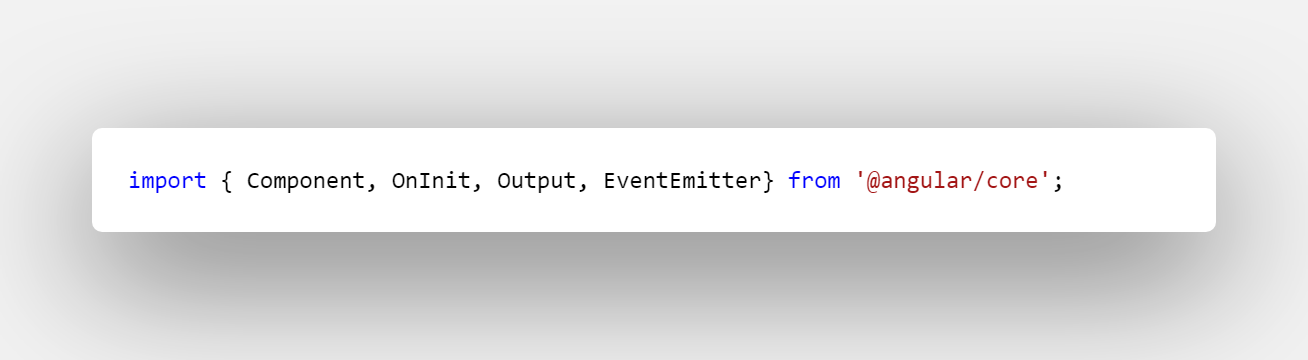
- Nếu value là 1 biến thì phải sử dụng [varNameInput] = (var-name);

### b. Output:

@Output dùng để gửi dữ liệu từ component con sang component cha.

*\* Cách sử dụng:*

- Bước 1: Tại component-child.ts, import Output và EventEmitter from @angular/core.



- Bước 2: Tại component-child.ts

+ Khai báo @Output('key') sender = new EventEmitter<"datatype return">();

+ Sử dụng this.sender.emit(data) để gửi dữ liệu là data sang component cha.



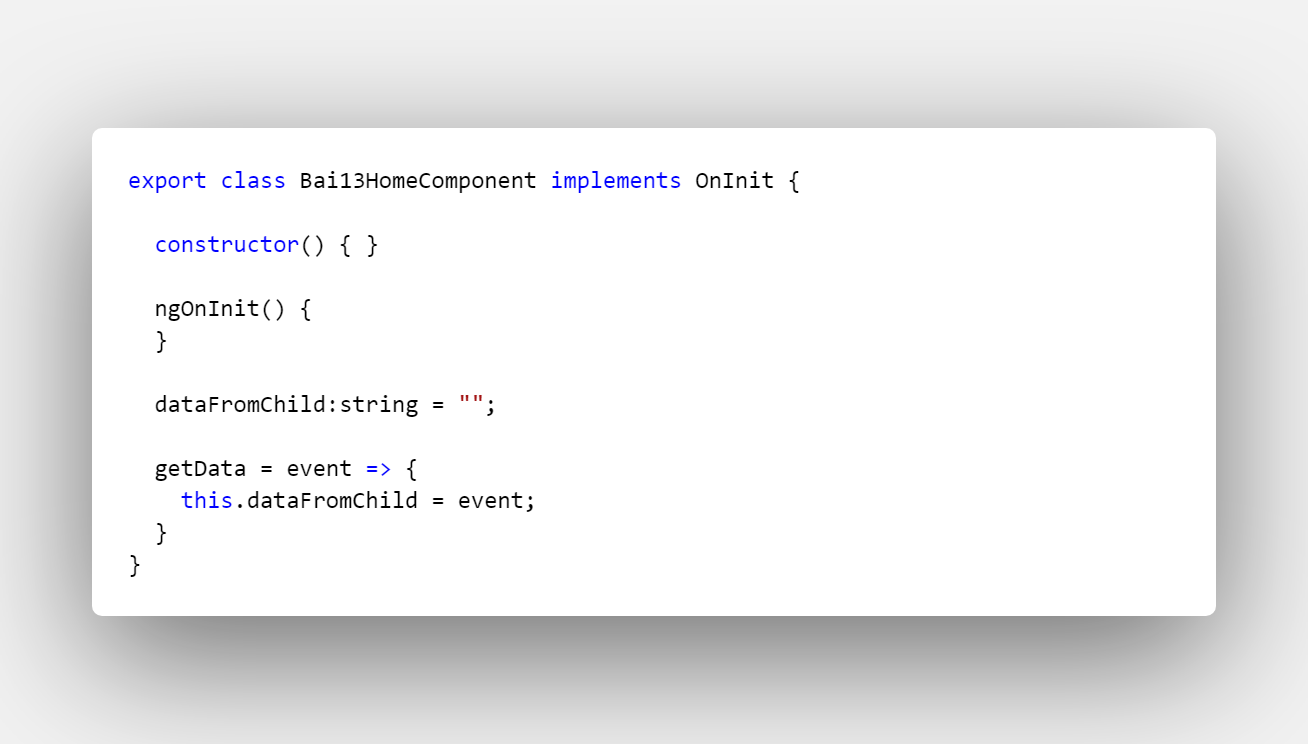
- Bước 3: Trong parent-component.html, tại directive component con, sử dụng (key)="method($event)"

+ method sự kiện của component cha nơi nhận dữ liệu gửi tới từ component con.

+ $event đại diện cho dữ liệu được gửi sang từ component cha.



- Bước 4: Trong parent-component.ts, lấy ra event trong method được sử dụng ở trên.



### c. ViewChild:

@ViewChild dùng để gọi đến 1 phương thức của component con, thông qua phương thức đó, ta có thể thực hiện hành động trong component con hoặc lấy được giá trị của các biến của component con.

*\* Cách sử dụng:*

- Bước 1: Trong component-child.ts, tạo phương thức để component cha gọi. Phương thức này có thể thực hiện hành động hoặc trả về giá trị.



- Bước 2: Trong parent-component.html, viết directive của component con.



- Bước 3: Trong parent-component.ts:

+ Import component con.

+ Khai báo @ViewChild(ChildComponent) private child-var: ChildComponent.

+ Gọi đến phương thức của component con thông qua this.child-var.



# IV. Pipes:

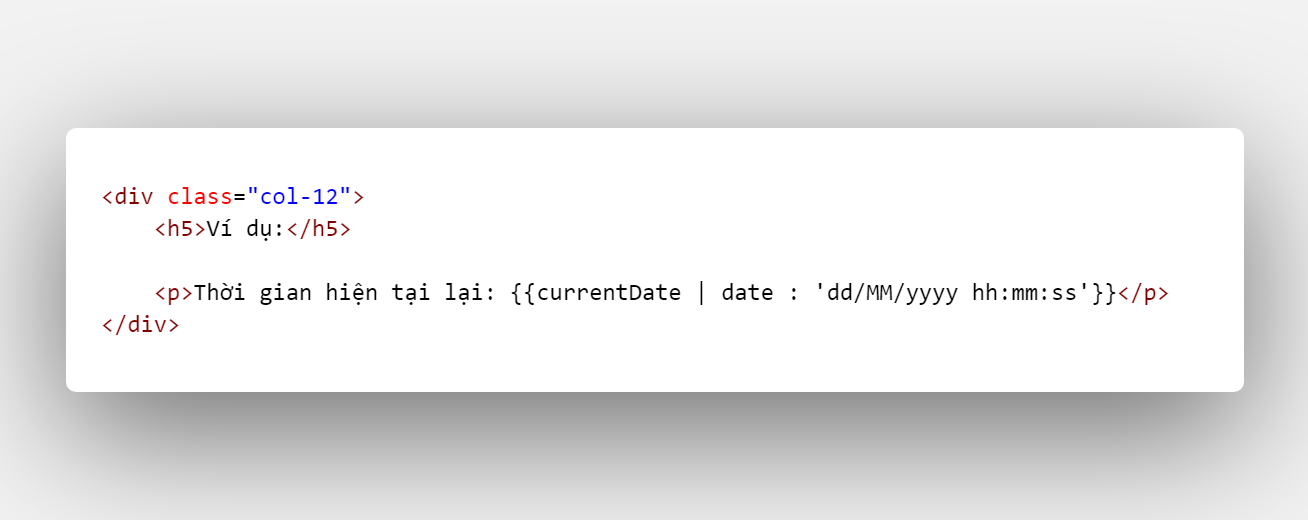
Pipes dùng để format lại dữ liệu, giống như Filter trong AngularJS.

*\* Cách sử dụng:*

- Bước 1: Tạo biến dữ liệu trong file ts.



- Bước 2: Trong thẻ html sử dụng {{varName | pipe }}. Có thể sử dụng nhiều pipe, các pipe cách nhau bởi ký tự "|".



Có thể tham khảo các pipes hiện có tại: [Pipes Angular](https://angular.io/api?type=pipe)

# V. Form:

Dùng để lấy dữ liệu của người dùng nhập vào.

*\* Cách sử dụng:*

- Bước 1: import FormsModule trong app.module.ts.



- Bước 2: Trong code html, tạo form:

+ Đặt id cho form và gán với giá trị ngForm: #idForm="ngForm".

+ Trong thẻ form sử dụng (ngSubmit)="method(idForm.value)" để khi bấm submit, tất cả dữ liệu của form sẽ được truyền vào trong method.

+ Trong từng thẻ input, khai báo name và ngModel.

+ Sử dụng thẻ input hoặc button với type là submit để submit form.



- Bước 3: Trong file ts, gọi đến method ở trên và lấy data để sử dụng.



# VI. HttpClient:

HttpClient dùng để gọi dữ liệu từ api.

## 1. get:

- Bước 1: Import HttpClientModule vào app.module.ts



- Bước 2: Tạo service bằng cmd 'ng g s service\_name'.

+ Import HttpClient, Observable.

+ Trong contructor, khai báo 1 biến private http.

+ Tạo method có kiểu dữ liệu trả về là Observable< any[] >

+ Trong method này, sử dụng this.http.get< any[]>('path\_api') để lấy dữ liệu từ api về.



- Bước 3: Tại nơi muốn lấy dữ liệu, import service và gọi đến method của service.



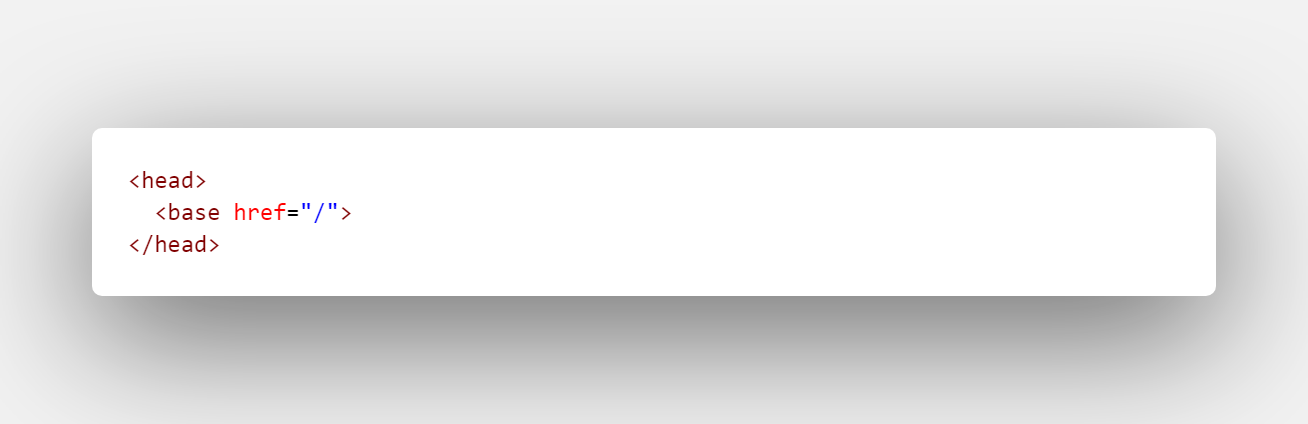
# VII. Routing:

## 1. Routing:

Routing dùng để định tuyến cho trang web, giúp chuyển trang mà không cần phải load lại trang web.

*\* Cách sử dụng:*

- Bước 1: Trong index.html, khai báo < base href="/"> trong cặp thẻ head để trang web nhận biết path gốc.



- Bước 2: Tạo file router:

+ Import Routes và RouterModule.

+ Khai báo routes có kiểu Routes là 1 mảng các path+component.

+ Ở cuối mảng routes, khai báo:

path:'\*\*', component: NotFoundComponent

Để chuyển đến trang Not Found khi link không tồn tại.

+ export appRouter.

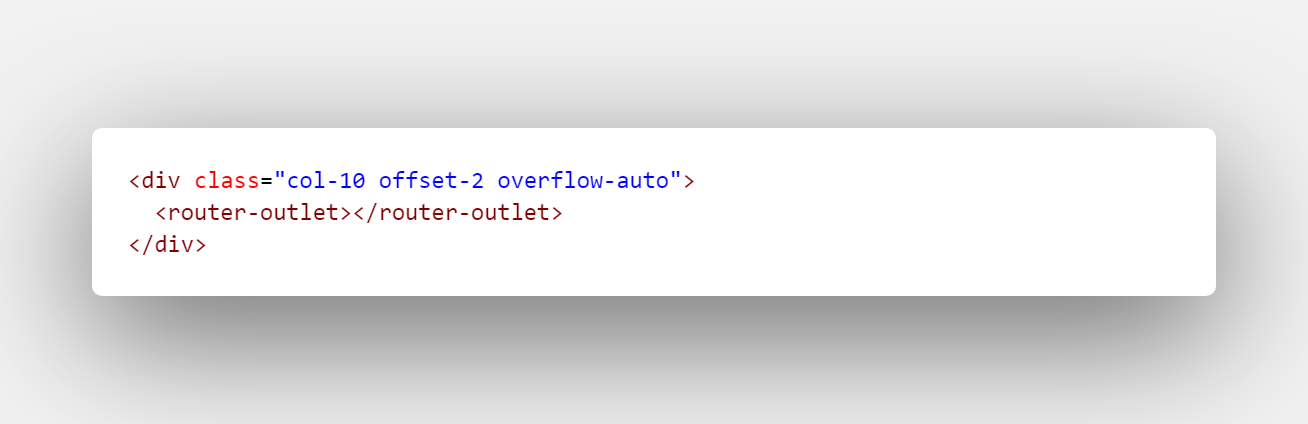


- Bước 3: Import appRouter ở trên vào app.module.ts



- Bước 4: Tại nơi muốn định tuyến component, ta đặt thẻ

< router-outlet> < /router-outlet>.



- Bước 5: Trong thẻ a, thay href bằng routerLink.

+ Thêm routerLinkActive="classActive" khi muốn thêm class cho thẻ a được click.



## 2. Redirect trong code ts:

- Bước 1: Trong code ts:

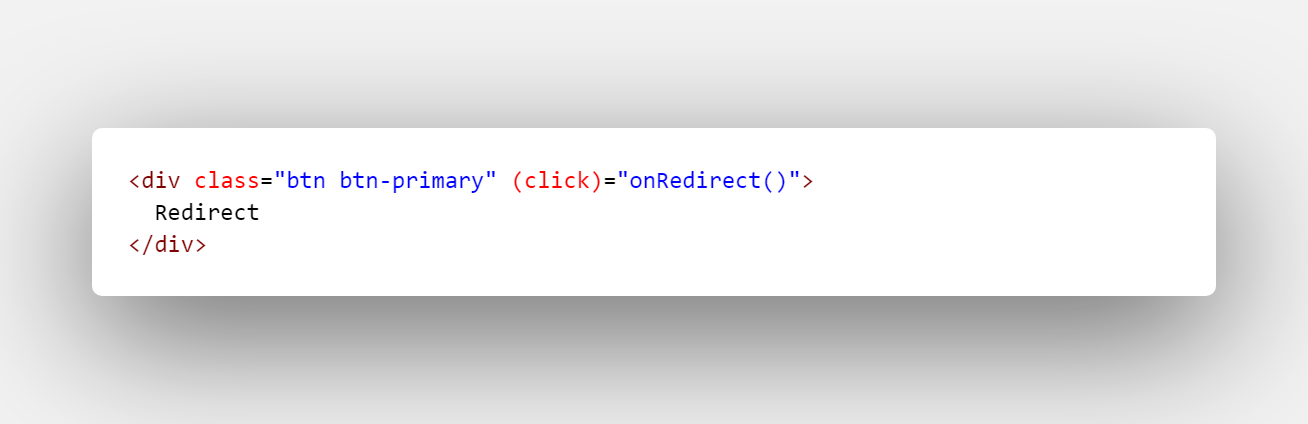
+ Import Router.

+ Khởi tạo 1 biến private router: Router trong contructor.

+ Tạo method, sử dụng this.router.navigate(['path\_muốn\_chuyển\_ đến]);



- Bước 2: Trong html, gọi đến hàm vừa khai báo ở trên.



## 3. Gửi và nhận tham số trên URL:

- Bước 1: Khai báo path cho component nơi nhận param.



- Bước 2: Tromg file component.ts nơi nhận param:

+ Import ActivatedRoute.

+ Khai báo biến activatedRoute trong contructor.

+ Sử dụng activatedRoute để lấy ra các param.



## 4. QueryParams:

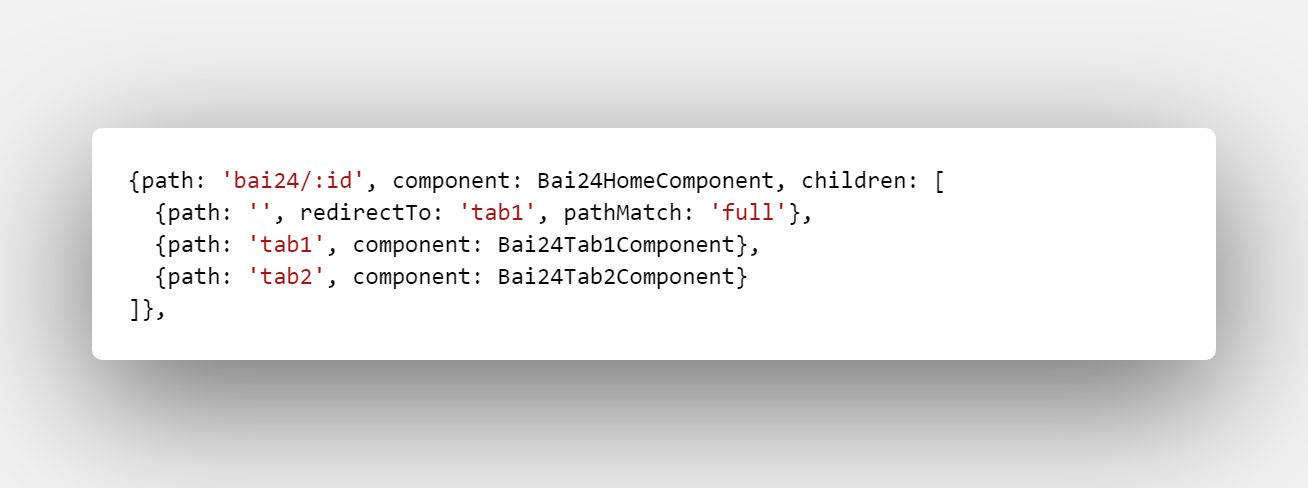
Đối với cách truyền tham số như ở trên thì ta phải khai báo param trong router. Đối với queryParam thì ta không phải khai báo trong router, các tham số queryParam có hay không có không ảnh hưởng đến URL.

## 5. Children Route:

Children Route sử dụng để lồng router vào trong một router khác.

*\* Cách sử dụng:*

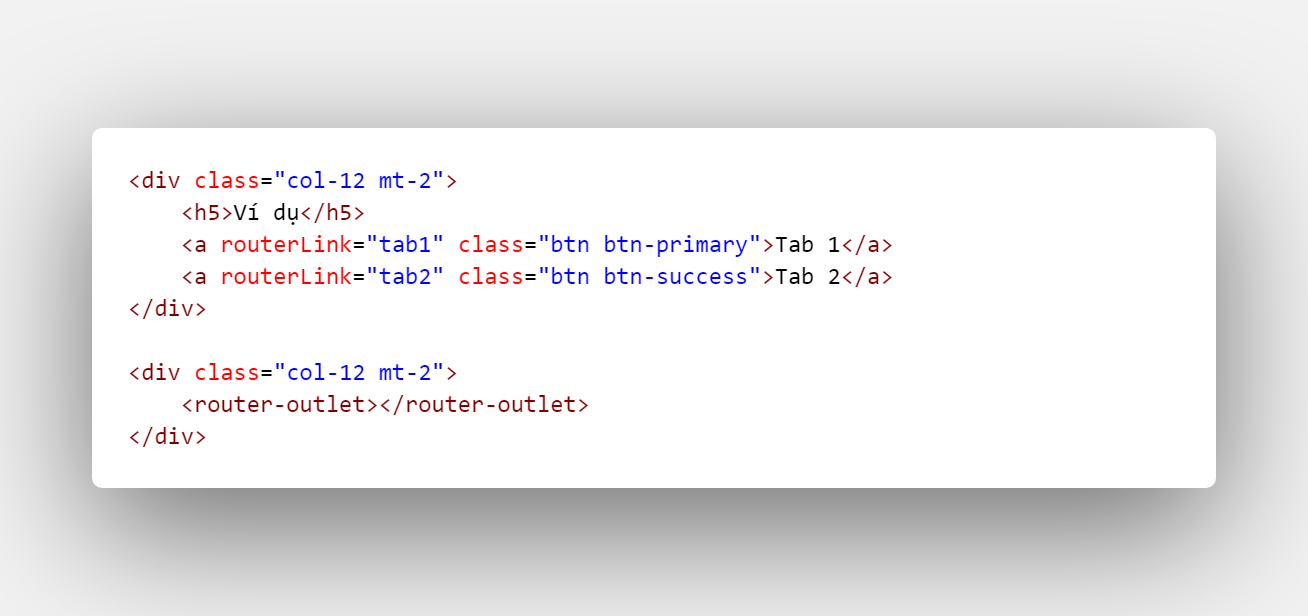
- Bước 1: Trong file router.ts, khai báo child trong path của component cha. Child sẽ chứa path của các component con.



- Bước 2: Trong component cha:

+ Tạo các thẻ a dùng routerLink để làm link cho các component con.

+ Đặt cặp thẻ router-outlet.



\* Nhận tham số từ URL của component cha

- Trong file ts của component con:

+ Import ActivatedRouter.

+ Khai báo 1 biến private của ActivatedRouter trong contructor.

+ Sử dụng activatedRouter truy cập đến parent và lấy ra các param theo param name.



## 6. Route Guard:

Route guard được sử dụng dùng để xác thực quyền truy cập trang web của người dùng hoặc bắt sự kiện người dùng rời khỏi trang web khi đang nhập thông tin trong 1 form. Có thể hiểu Route Guard tương tực như Interceptor trong Java.

*\* Cách sử dụng:*

- Bước 1: Tạo file guard.ts:

+ Phải khai báo @Injectable để thực hiện cơ chế DI.

+ Implement CanActivate và overite lại phương thức canActivate.

+ Trong phương thức canActivate sẽ return ra true hoặc false. Nếu là true thì sẽ cho đi đến path được chỉ đinh guard này. false ngược lại.



- Bước 2: Thêm canActivate vào trong path. Giá trị của canActivate là một mảng các guard. Guard này cũng sẽ có tác dụng với các path child của path đó.

