* **Angular** là 1 single page application (đều có index.html)
* **Angular CLI** là một công cụ được phát triển đề chạy trên giao diện cửa sổ ứng dụng (command line interface hay CLI) nhằm giúp đỡ việc tạo dự án, quản lý tập tin trong dự án và thực hiện nhiều tác vụ khác nhau như test (kiểm thử), bundle và deploy dự án một cách nhanh chóng và hiệu quả.
* Tất cả các cấu hình trong đồ án khi khởi tạo trong Angular đều được chứa trong file **angular.json**
* Chạy chương trình bằng: ng serve or ng s or ng s –open
* 1 trang trong angular được cấu thành từ nhiều component
* Main.ts là file đầu tiên chạy khi khởi động 1 dự án
* **Main.ts** => ts module => tập hợp của nhiều module khác nhau trong dự án (từ các thư viện bên ngoài hoặc từ local)
* Main.ts sẽ gọi hàm

**platformBrowserDynamic()**.bootstrapModule(AppModule)

* .catch(*err* => console.error(*err*));
* platformBrowserDynamic() : nếu làm việc với web
* **bootstrapModule**(AppModule): Khởi động 1 cái module vd này cho thấy nó đang khởi động AppModule

**app.module.ts:** là 1 typescript module

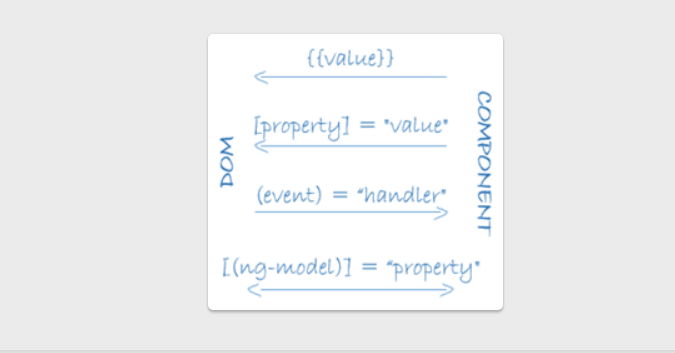
* Có thể inport những thứ bên ngoài vào sử dụng hoặc export những thứ tồn tại bên trong nó ra để bên ngoài sử dụng
* Có dấu @ đằng trước ex: @**NgModule** gọi là typescript declarator
* Angular sử dụng typescript declarator để cung cấp cho Metadata: data about data
* Data: đại diện cho **AppModule**
* Data : là những phần bên trong @**NgModule** có liên hệ mật thiết đến **AppModule**
* App.component.ts là typescript module
* @Component({
* selector: 'app-root', //
* templateUrl: './app.component.html', // hiểu thị view
* styleUrls: ['./app.component.css'] // cung cấp css or scss
* })

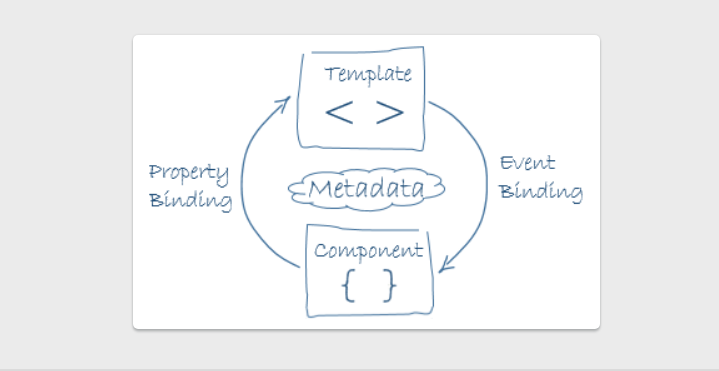
Lệnh tạo component: ng generate component <name conponent>

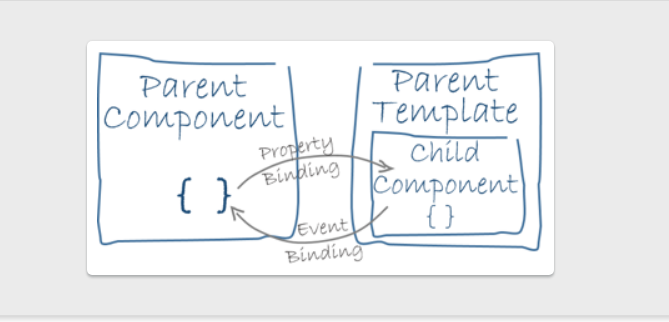
**Data binding:** (tạm dịch ) ràng buộc dữ liệu

**DOM là gì?**

Khi browser load trang web của bạn, nó sẽ parse phần HTML và xây dựng nên một cây Document Object Model (DOM) từ đó để biểu diễn tương ứng những gì HTML đang được dựng, cho phép chúng ta có thể tương tác với phần HTML như đọc, sửa HTML bằng JavaScript.







Có 1 số ways để cho component.html giao tiếp với class Appcomponent

## Muốn hiển thị giá trị của 1 object nào đó (INTERPOLATION): {{ expresstion (cách diễn tả) }} trong expresstion evaluate (Tìm 1 biểu thức) nó return giá trị gì thì display giá trị đó giá. Giá trị trả ra sẽ có datatypes: string nó sẽ gọi hàm .toString() để chuyển đổi giá trị và display ra html

Example: <input type = “text” value = “Thien”/> // browser chuyển thành dưới dạng Document Object Model (DOM) giúp cho chúng ta tương tác đc vs html = js

Attribute được để chỉ những gì các bạn đặt vào phần opening tag của một tag sẽ là attributes, còn những gì là property của object (node) sẽ được gọi là property

Syntax of Data Binding: when by data binding example:

<input [type] = “ ’text’ ” value = “Thien”/> : ’text’ là một expresstion

App.component.ts: class Appcomponent

inputType = ‘text’

.html

<input [type] = “ inputType ” value = “Thien”/> có giá trị tương tự

Nếu chúng ta biết cái property không thay đổi

<input type = “text” value = “Thien”/>

Event binding:

<input type = “text” (click)= “handler()” value = “Thien”/>

Dùng để nhận sự kiện từ template và sử lý trong app.component.ts

## TWO-WAY BINDING

Sử dụng import app.module: FormModule

<input type = “text” [(ngModel)] = “name”/>

Mở rộng:

<input type = “text” [ngModel] = “name” (ngModelChange) = “$event”/> ngModelChange sẽ bắt sự kiện đang diễn ra và có giá trị tại thời điểm sẩy ra sự kiện

If – angular

Cú pháp có chứa dấu \* (asterisk) như sau \*ngIf="expression"

Why do we call structural directive structural?

structural directive là một đoạn directive có thể thay đổi cấu trúc DOM có thể thêm vào cũng có thể bỏ ra.

Khi expression bằng truthy thì giá trị mà ta set up cho nó đc gọi và gắn vào DOM nếu expression bằng falsy thì all các thẻ, giá trị đc set up trong đk truthy sẽ bị loại bỏ ra khỏi DOM thay vào đó là các thẻ, giá trị đc set up falsy.

Trong \*ngIf="expression" người ta thường sử dụng thêm

<ng-template></ng-template>

Đây là một nhóm các temlate được định nghĩ trc hiển thị khi thỏa điều kiện gì đó

Example:

<app-hello *\*ngIf*="user.age >= 18;else under18" *[name]*="user.name"></app-hello>

<ng-template *#under18*>

  <div>

    <h1>Young</h1>

  </div>

</ng-template>

So sánh cách viết thông thường vs cách viết đầy đủ

Example:

<app-hello *\*ngIf*="user.age >= 18;else under18" *[name]*="user.name"></app-hello>

<!--Viết đúng không thu gọn-->

<ng-template *[ngIf]*="user.age >= 18" *[ngIfElse]*="under18">

  <app-hello *[name]*="user.name"></app-hello>

</ng-template>

<ng-template *#under18*>

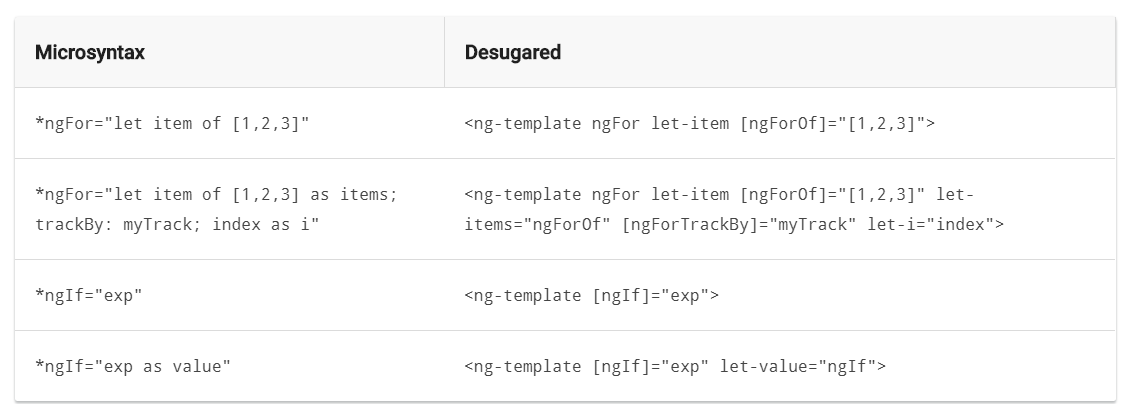
  <div>

    <h1>Young</h1>

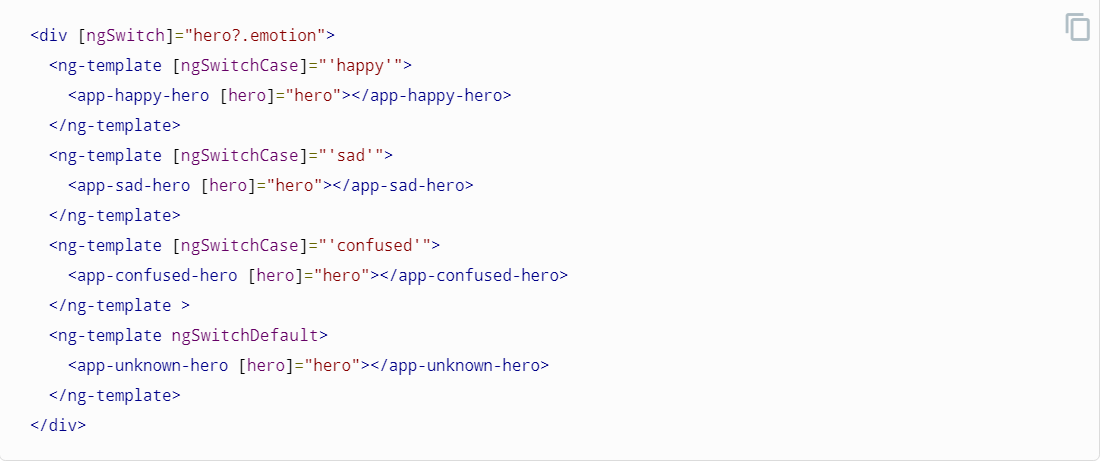
  </div>

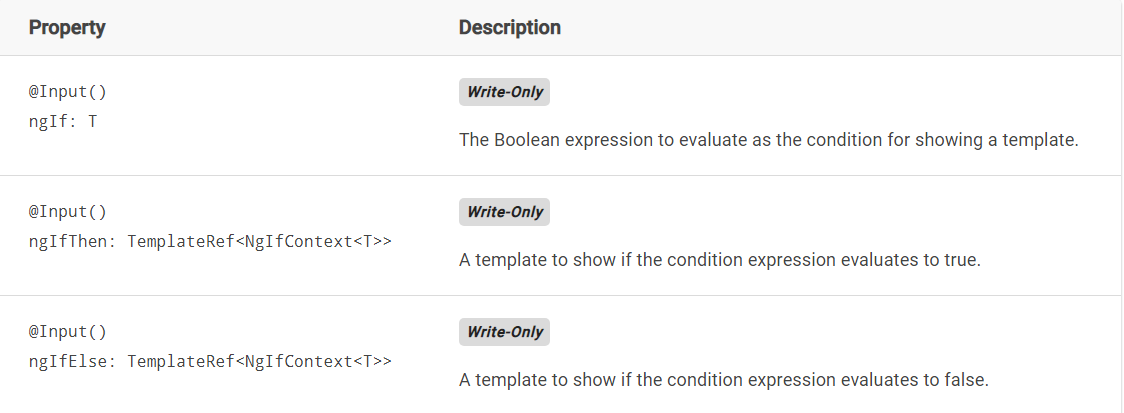
</ng-template>

Sự khác biệt giữa [hidden] với structural directive khi expression có giá trị là falsy all các thẻ và giá trị được gắng trong truthy không bị bỏ đi nó chỉ ẩn xuống và vẫn còn trong DOM còn structural directive thì bỏ ra khỏi DOM



ngSwitch



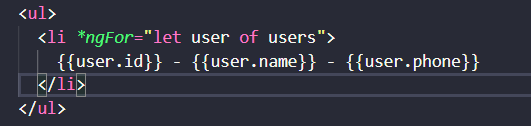


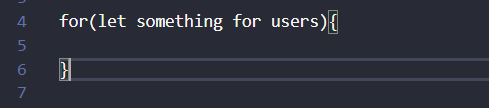
<https://angular.io/api/common/NgIf>

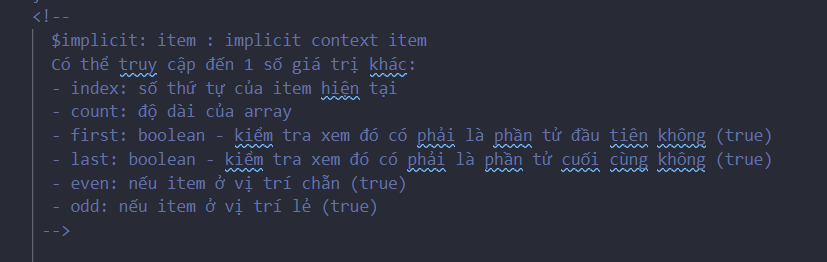
<https://angular.io/guide/structural-directives>

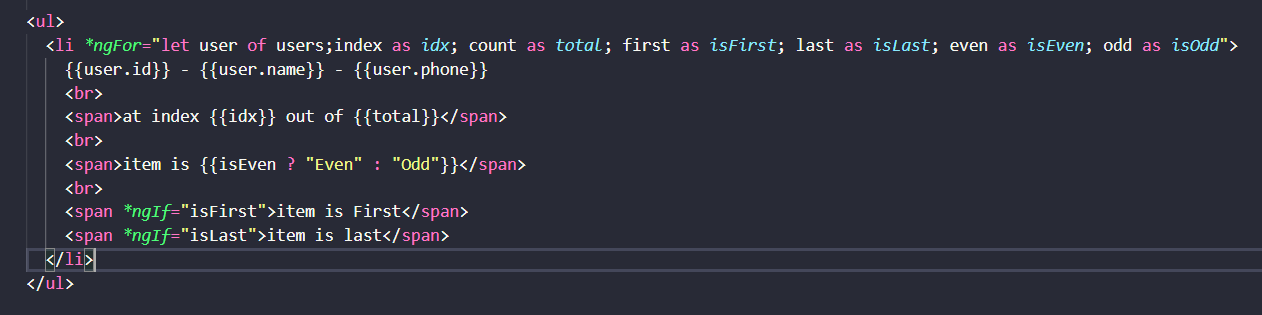
ngFor directive

Khi muốn in ra các tag thẻ giống nhau từ 1 mảng vd table, list,….

Syntax: 

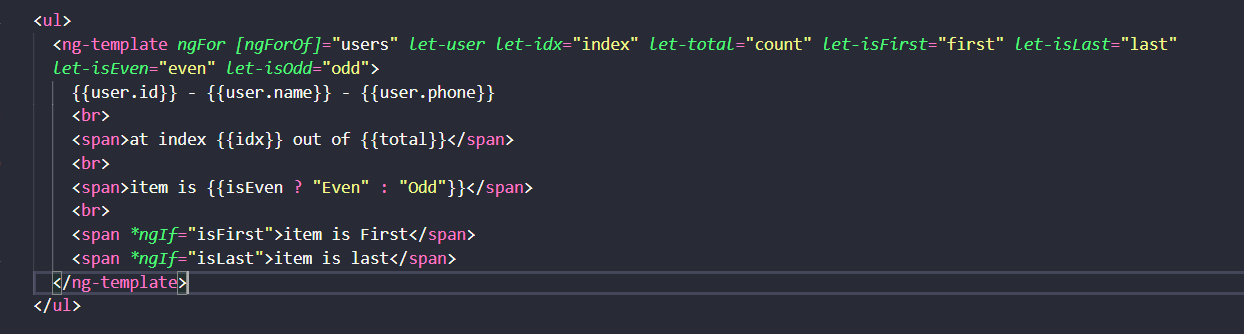
Tương tự cú pháp trong ts: 

NgFor cung cấp: 

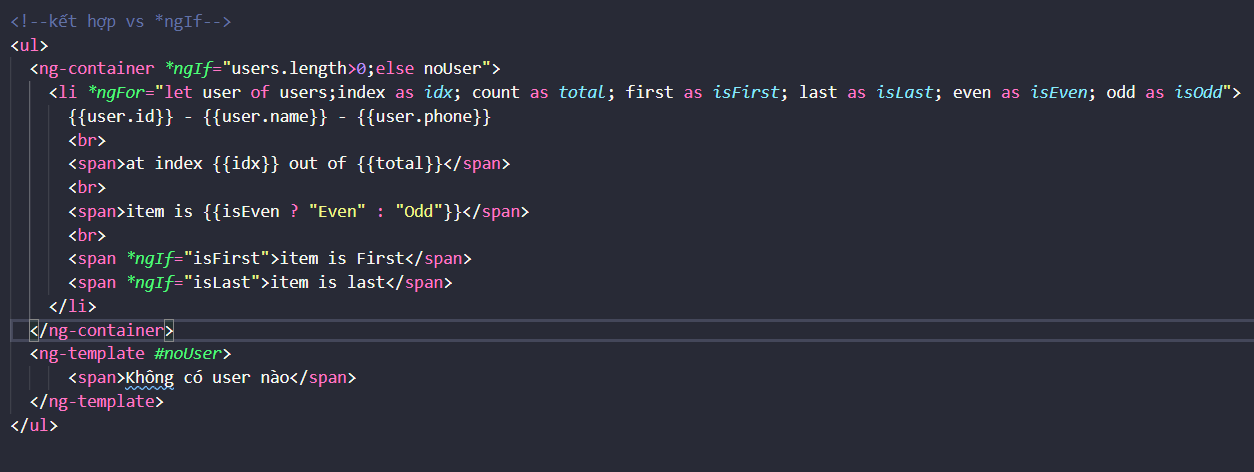




Viết đầy đủ:



Kết hợp vs \*ngIf



<https://angular.io/guide/structural-directives>

<https://angular.io/api/common/NgForOf>

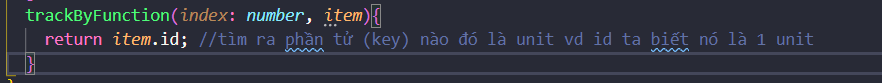
<https://www.tiepphan.com/thu-nghiem-voi-angular-2-built-in-directives-ngif-ngfor-ngswitchcase/>

<https://www.tiepphan.com/thu-nghiem-voi-angular-2-ngfor-index-first-last-even-odd-trackby/>

Trong trường hợp ta add hay delete item sẽ xẩy ra 1 lỗi là list ban đầu khi ta thêm/ delete item nào đó thì nó sẽ rander lại toàn bộ list ban đầu của chúng ta. Để khắc phục ta sử dụng 1 thủ thuật gọi là Track by function



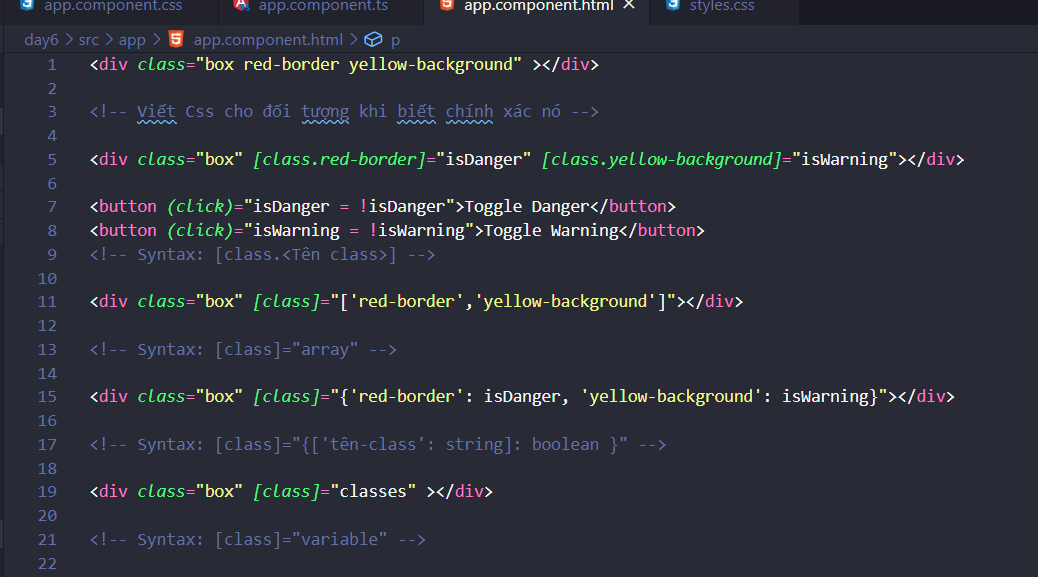
TS



<https://www.youtube.com/watch?v=dXDC-4KGIGI>

# Day 6: ATTRIBUTE DIRECTIVE

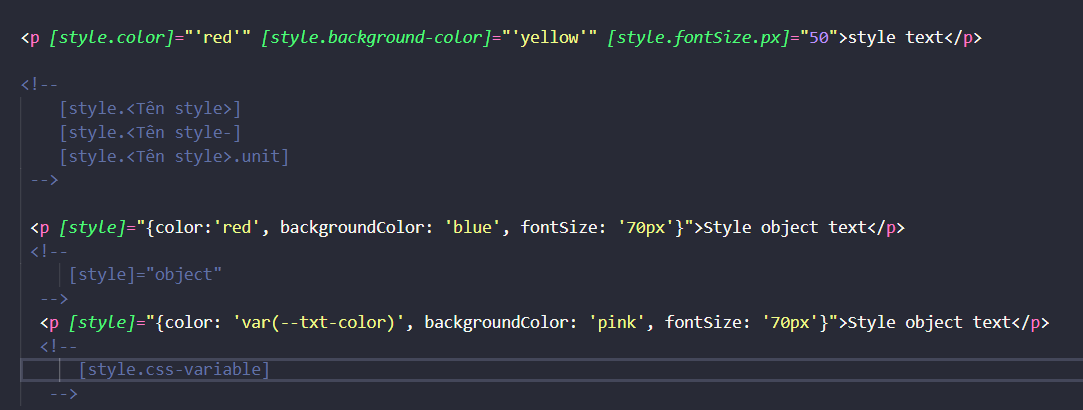
Class Binding



Syntax:

* [class.<Tên class>]
* [class]=”array”
* [class]=”{[‘Tên class’: string]: boolean}” (object)
* [class]=”variable’

Style Binding



Syntax:

* [style.<Tên style>]
* [style.<Tên style>-]
* [style.<Tên style>.unit]
* [style]=”object’
* [style.css.variable]

Data down: parent component tương tác với child component

Event up: child component tương tác với parent component