Angular4.x

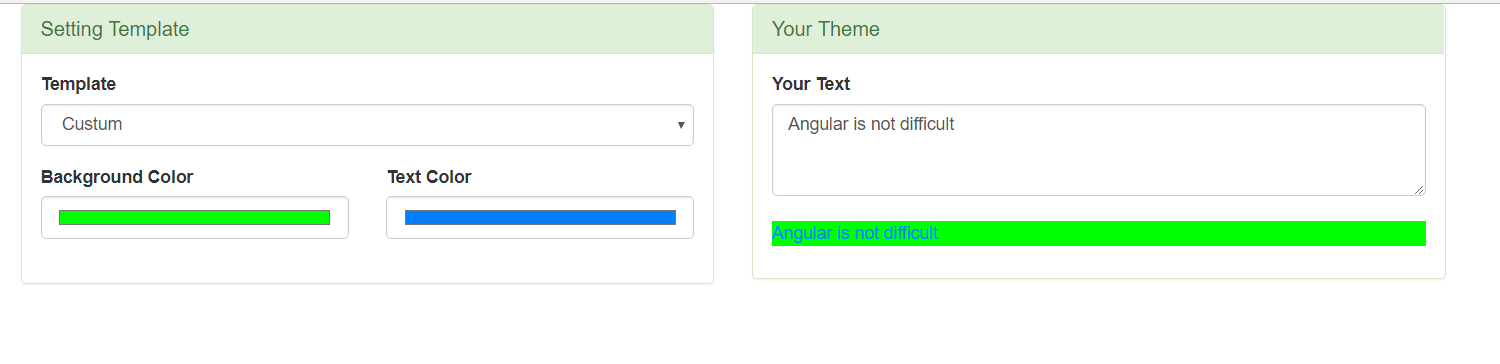
Phần 2 :

1. Data binding : giúp truyền các thuộc tính ra ngoài template
   1. Sử dụng interpolation {{}} :
      1. có thể dùng được với thuộc tính,
      2. phương thức ,
      3. đối tượng
      4. Dùng được cả các thuộc tính trong thẻ như src = {{}}
   2. Khi đưa dữ liệu cho :
      1. Property
         * Dùng
           1. {{expression}}
           2. [target]="expression"
           3. bind-target="expression"
      2. Attribute
         * Dùng
           1. [attr.attr\_name]=”” : vi du [attr.colspan]=”1+1”
      3. Class
      4. Style
         * [style.color]= “isActive ? ‘red’ : ‘yellow”
2. Event – binding : giúp truyền các thuộc tính từ view-> data sourc
   1. Cú pháp :
      1. (target)="statement" : phương thức mà chúng ta gọi
      2. on-target="statement"
   2. giúp bắt sự kiện:
      1. <button (click)="onSave()">Save</button>
   3. Nếu truyền vào $event thì cho phép lấy thông tin cụ thể hơn nữa:
      1. (click)="onSave($event)”
      2. Để tránh trường hợp vừa gọi sự kiện click của button và click của thẻ div thì : $event.stopPropagation()
   4. Ví dụ nhỏ : jQuery Tag
      1. Khi nhập nội dung abc vào ô input thì giá trị ở input bị xóa và nhẩy xuống vị trí thẻ p
      2. Giải quyết:
         * Bắt sự kiện nhập từ bàn phím : (keyup)=”onKeup($event)”
         * Method onKeyUp(): do người dùng tự định nghĩa
         * $event : cho biết khi nào người dùng nhập vào phím enter
         * Kiểm tra nếu $event.which == 13 : nghĩa là người dùng đã ấn vào phím enter
         * Lấy nội dung ra bằng cách truy cập vào phần tử target => value
         * Rồi đặt nó vào sử dụng jquery
         * Thêm vào rồi thì đặt lại value = “ “ để loại bỏ giá trị
         * Khắc phục dấu $ cảnh báo : declare var $ :any;
3. Two-way binding:
   1. Chức năng đổ dữ liệu 2 chiều
   2. ngModel : là câu cầu để kết nối 2 chỉ thị này nghĩa là phải import FormsModule
      1. import { FormsModule } from '@angular/forms';
4. cách đặt tên biến : #number
5. Tìm hiểu về Built-in directives:
   1. ngIf:
      1. cú pháp : \*ngIf="isActive"
   2. ngFor
      1. cú pháp : \*ngFor="let course of courses;let i=index "
      2. biến i cho biết chỉ số
   3. ngSwitch
      1. chú ý bên trong phải để dấu : \*ngSwitchCase = "'ConGa1'"
      2. phần ví dụ về các tab khi kích vào tab home thì tab đó hiện nội dung lên
         * sử dụng ng Switch
         * Sử dụng data-bindind để add cho class active
         * Sử dụng sự kiện click
   4. ngClass : thêm hoặc loại bỏ nhiều class css cùng 1 lúc
      1. [class. glyphicon]="isSpecial" : cach này sử dụng trong derective thôi
      2. [ngClass] = "{

'glyphicon-th' : !isSpecial,

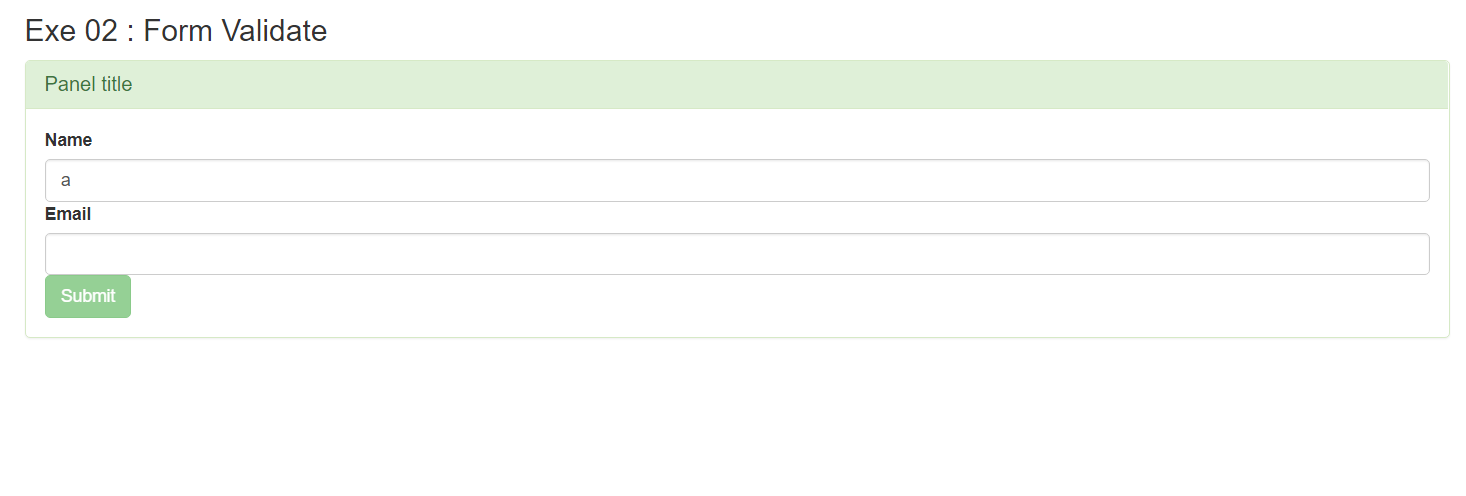
'glyphicon-list': isSpecial}"

* + 1. Viết ngắn gọn hơn nữa [ngClass] = “method()”
  1. ngStyle :
     1. có thể viết theo kiểm style-bingding nhưng để cú pháp ngắn hơn thì viết ngStyle
     2. cú pháp:

1. Bài tập Setting Template:
   1. Hình ảnh :
   2. Các chức năng :
      1. Khi thay đổi giá trị trong text aria thì giá trị trong ô bên dưới cũng thay đổi => two-way-binding là ra
      2. Khi thay đổi trong giá trị select thì phần dao diện cũng thay đổi theo => hướng làm khi thay đổi thì thay đổi class của nó là xong
         * Thực hiện đặt cho cái select = 1 biến là #theme và nhớ thêm sự kiện (change)=0 : thể hiện sự thay đổi
         * Đặt luôn giá trị của option rồi khi người dùng thay đổi thì lấy luôn giá trị đó gắng vào bên dưới : theme.value
      3. Khi lựa chọn vào custum thì hiển thị 2 ô backgroung Color vs text Color hiển thị
      4. Khi xây dựng sẽ gặp lỗi : control must be defined as 'standalone' in ngModelOptions

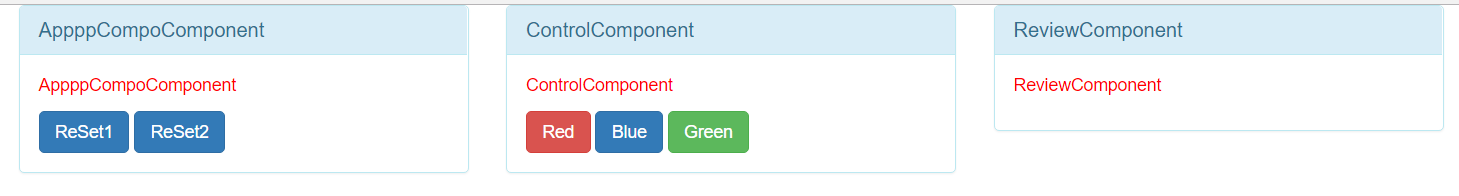
Các fix : [ngModelOptions]="{standalone: true}"

* + 1. Dùng biến đó .value để lấy giá trị lựa chọn ở phần color và dùng ngStyle để thêm thuộc tính đó vào ô nhưng chú ý những từ có 2 từ trở lên thì bỏ dấu – và viết hoa chữ cái đầu của từ thứ 2
    2. Khi xây dựng xong thì thấy khi ấn vào các lựa chọn blue, black thì nó lại k thay đổi nguyên nhân là nó lấy định dạng của phần style đè lên rồi => hướng giải quyết là tạo 1 thẻ p nữa cho 2 trường hợp khác nhau 1 thẻ p dùng cho trường hợp class 1 thẻ p dùng trong trường hợp người dùng chọn theme là custom
  1. Xây dựng giao diện cần chú ý :
     1. Thẻ form dùng để bao bọc phần bên ngoài khi nhận và gửi dữ liệu đi
     2. Để hiển thị phần lựa chọn mầu sắc thì để input type=”color”
     3. Đùng div với class form-control để bao bọc phần select

1. Bài tập Vote-Status
   1. Hình ảnh demo:
   2. Phân tích:
      1. Showvote : cho biết khi nào nó hiển thị
      2. Method voting(): truyền vào tên của cảm xúc và thay đổi để tắt show đi
      3. Sự kiện chuột di chuyển vào : mouseover
      4. Sự kiện di chuyển chuột ra : mouseleave
      5. Khi thẻ span được hover vào thì showVote == true và đoạn nội dung được hiển thị
2. Bài tập Form validate
   1. Hình ảnh đề mo: 
   2. Yêu cầu :
      1. Khi vừa vào thì k button submit : k được ấn, khi nhập tên và email thì có thể ấn button đó
   3. Thực hiện:
      1. Giao diện :
         * Sử dụng pannetl
         * Bên trong là 1 form
         * Cấu trúc 1 phần tử là 1 lable , 1 input, 1 thẻ span để hiện thị khi người dùng ok và 1 thẻ span để hiển thị khi người dùng nhập sai
      2. File ts
         * Định nghĩa 1 interface Person để tý nữa tạo 1 đối tượng personInfo thuộc nó
         * Tạo biến validName, validEmail : kiểm tra name,email có hợp lệ
         * onSubmit : kiểm tra người dùng đã click vào button đó hay chưa , nếu chưa click thì = false và nó là mặc định
         * elements : string[] = [“name”,”email”] : mục đích sử dụng directive ng-for để sinh ra tập tin giao diện
         * button có sử dụng method setDisableButton() :trả về kiểu true hoặc false : bằng cách kiểm tra 1 trong 2 cái name hoặc email là rỗng là trả về false
         * method isEmty(): viết để kiểm tra độ dài thôi : và phương thức chỉ sử dụng trong class này lên quyền truy cập là private
         * method showLable() : chuyển kí tự đầu thành in hoa kí tự sau chuyển thành chữ thường
         * onInputChange($evnent, element) : truyền vào ô name và email nếu name thì cập nhập cho name còn nếu nhập vào email thì cập nhập cho email
         * onSubmitForm() : khi click vào thì …

Phần 3 :

1. Tempalate reference variable : #
   * + - 1. Đặt 1 nội dung là 1 biến rồi truy cập vào nó
2. Input : từ Component Cha truyền cho component Con thì dùng Input : chú ý phân biệt cha con rõ ràng
   1. Cách nhớ Input => []=””
   2. Truyền vào tk nào thì đặt selector vào tk đấy truyền thông qua giá trị mtitle(tự đặt) = title (giá trị muốn truyền)
   3. Truyền đi cần cái hứng để nhận :
      1. Import Input
      2. @Input(“truyền thông qua cái gì ”) đặt lại tên : kiểu dữ liệu
      3. Truyền nhiều dữ liệu thì trong component thêm inputs:[“mtitle”,”aaa”]
3. Output : từ con truyền ra cha thì dùng Output
   1. Cách nhớ O : ()=””
   2. Vị trí khai báo ở selector của nó
   3. Output cần :
      1. Impor Output, EventEmitter
      2. @Output(“answerRequest”) myRequest = new EventEmitter<string>();
      3. <string> : là genergit của nó cho phéo truyền kiểu dữ liệu chúng ta muốn
      4. Tại component con truyền đi cho cha thì truyền cho biến myRequest : this. myRequest.emit
      5. answerRequest=”method” là cái xe để chuyển biến đi
      6. và trong tk cha khai báo method để hiển thị nội dung đã nhận được từ tk con
4. template:
   1. cú pháp <template></template> : cứ hiểu ngầm nó như 1 seletor nhỏ thôi và giúp mã HTML k bị lồng quá nhiều mã khác
5. Bài Tập : Setting Color
   1. Demo



* 1. Xây dựng giao diện
     1. Chia làm 3 khối ứng với 3 component
     2. Sử dụng pannnel và button để thực hiện
  2. Yêu cầu bài tập như sau
     1. Khi ấn vào nút reset thì tất cả 3 component phải chuyển hết về mầu đen
        1. Tạo 1 biến có nội dung là mầu đen , sử dụng databinding hoặc ng-Style để định nghĩa rồi đổ nó ra chính component đó rồi truyền biến đó đi 2 component kia
        2. Truyền nó đi thì sử dụng input
        3. Khi click vào button thì truyền mầu sắc đó đi 2 component kia :
           1. Sử dụng sự kiện click và truyền vào mầu sắc trong đó => in ra được mầu sắc khi click vào
           2. Sử dụng output để truyền cho tk cha. Thực hiện nhiệm vụ đẩy output ra thì đặt tại selector đó
           3. Giải thích sự kiện click output : khi changeColor được click thì nó đẩy giá trị vào emit theo biến selectColor rồi truyền ra ngoài selector của nó rồi truyền tiếp ra ngoài tk cha của nó là xong
        4. Thực hiện chức năng khi ấn vào nút reset thì truyền vào cho mầu đen ban đầu
           1. Cách 1 sử dụng sự kiện click là 1 function để truyền giá trị mầu đen vào
           2. Cách 2: sử dụng template variable chính là cái control component và nó sẽ sử dụng được phương thức của control.ts

1. Bài tập ListColor
   1. Demo
   2. Xây dựng giao diện
      1. Dùng table để hiển thị
      2. Tạo 1 arr kiểu string chứa danh sách các khóa học và dùng arr đó đổ ra view => sử dụng ng-for
   3. Chức năng dòng chẵn , lẻ , dòng đầu tiên , cuối cùng khác nhau phần mầu sắc
      1. Sử dụng [ngClass]=”function(i)” truyền vào biến i nó chạy từ 0 đến phần tử cuối cùng và trả về giá trị đầu tiên , cuối cùng , chẵn , lẻ
2. Bài tập Setting Tab
   1. Demo
   2. Xây dựng giao diện
      1. Chia làm 2 phần control và result
   3. Thực hiện chức năng
      1. Truyền từ cha vào con thì dùng input và truyền biến spaceInOneTab
      2. valueTab : là 1 arr truyền ngược lại bên ngoài
      3. giá trị mặc định chính là giá trị truyền vào và để [selected] = “dk”
      4. khi lựa chọn các giá trị muốn lấy được giá trị đó thì dùng template variable rồi truyền = cách

(change) = “settingTab(selectTabs.value)” thực hiện cập nhập lại giá trị ban đầu và tiến hành gửi ra cho tk cha

* 1. thực hiện phần coppy thì có từ khóa là js clipboard

**Phần 4.**

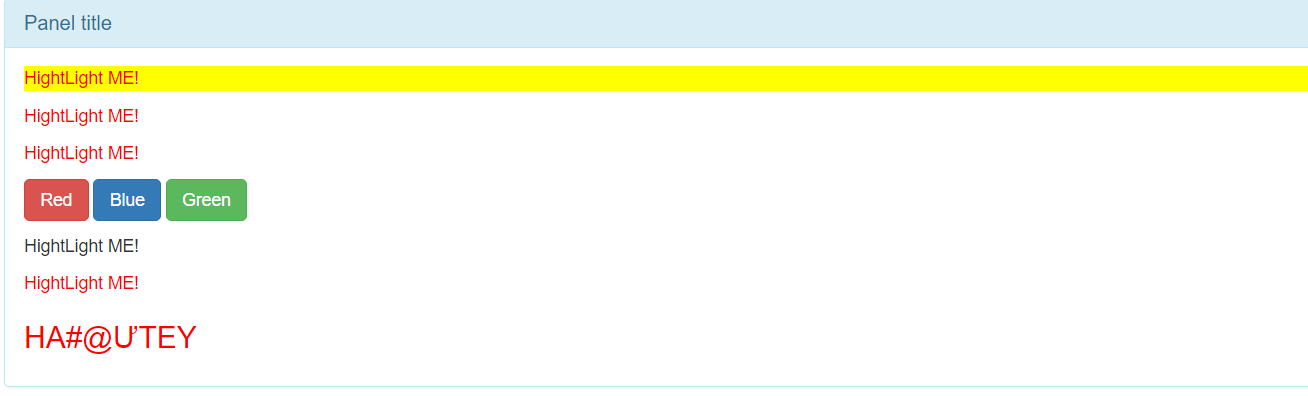
1. Tìm hiểu về các loại pipe
   1. **LowerCasePipe :** chuyển chuỗi sang dạng chữ thường
   2. **Uppercase :** chuyển chuỗi sang dạng chữ hoa
   3. **slicePipe :** slice:0:4 => lấy giá trị từ 0 đến 4 (ko lấy kí tự 4)
   4. **DecimalPipe :** dùng để lọc dữ liệu kiểu số
      1. Ví dụ: number:’1.8-12’
      2. 12 là số chữ số thập phân tối đa chúng ta muốn lấy
      3. 8 là số chữ số thập phân tối thiểu nhất muốn lấy
      4. Số con số ở phần nguyên mà chúng ta muốn hiển thị
   5. **PercentPipe :** giúp format những con số theo dạng %
   6. **CurrencyPipe :** định dạng tiền tệ truyền vào **currencyCode is the ISO 4217**
   7. **DatePipe :**  Định dạng ngày đọc trong doc
   8. **Json Pipe :** định dạng chuỗi json
   9. **Kết hợp nhiều pipe | pipe1 | pipe2**
2. **Custom Pipe**
   1. **Capitalize :**  giúp chuyển đổi từ đầu tiên trong mỗi từ thành chữ in hoa
      1. **Chú ý :** value trong custumpir : chính là nội dung text ,args :là giá trị truyền vào ở bên giao diện nhé
   2. **Array-**pipe : khi viết pipe thì return cái gì thì nó trả về cái đó
      1. **Array.**filter(function(value){});
      2. **Filter là để lọc các phần tử ví dụ lọc các phần tử chẵn thì đơn giản return về phần tử chẵn %2 == 0 là ok**
      3. **Cách 2 có thể dùng arrow function đó là bỏ function (value) và chuyển thành => là ok**
      4. Giả sử muốn chuyển thành pipe lẻ k lẽ lại viết 1 pipe nữa thì ko hay cho lắm thì ta sẽ sử dụng đến param để giải quyết bài toán này
         1. Cần truyền vào tham số ở pipe ngoài giao diện trước ví dụ : | array : ‘odd’
         2. Phần args : string
         3. Kiểm tra nếu người dùng truyền ‘odd ’ thì trả về chẵn ko thì trả về lẻ
   3. **Sort –pipe :**  thứ tự các phần tử sẽ sếp theo chiều anpha b
      1. **Theo kiểu tăng dần thì truyền biến**  : ASC
      2. **Theo giảm dần thì truyền** : DESC
      3. **Chỉ cần sử dụng phương thức sẵn có để sắp xếp** : SORT() là giải quyết được nhưng lại cả input và output đều thay đổi
      4. **Hướng giải quyết** : **tạo ra 1 mảng tạm =** array ban đầu gọi method slice()
      5. **Muốn sếp giảm dần thì chỉ cần đảo ngược lại cái tăng là ok sử dụng hàm reverse()**;

**d. Summary-pipe :**  hiển thị 1 đoạn miêu tả ngắn gồm 10 ký tự

* + 1. Truyền vào số kí tự muốn hiểu thị ví dụ 20 : |summary :20
    2. Sử dụng substring để cắt lấy từ kí tự 0 đến kí tự args truyền vào
    3. Nâng cấp : nếu truyền vào là 1 đống các thứ thì cần thay đổi args thành any[] và tạo let param1 = args[0] ; param2 = args[1] và thường tạo giá trị mặc định cho nó nếu nó khác 0 và rỗng tthif lấy chính nó còn nó k khác thì lấy giá trị mặc định
    4. Nhưng sẽ bị phần tử 0 chưa được định nghĩa => hãy vào phần truyền nhiều tham số ở pipe đó là currentcy pipe
       - Phần iii là sai cần gán mặc định ngay trên phần transform truyền vào
  1. Category –pipe : lọc cái danh mục trùng nhau thì k hiển thị
     1. **Tạo ra 1 kho chứa** là 1 array rỗng tên là result
     2. **Duyệt qua các phần tử trong category => đổ các phần tử đó vào 1 mảng mà chúng ta hứng trước**
     3. **Trước khi đổ kiểm tra xem nó tồn tại hay chưa nếu chưa thì đổ vào còn rồi thì k đổ**
     4. **Kiểm tra cái biến mới cate tồn tại trong result hay chưa thì sử dụng indexOf <=-1 chứng tỏ chưa tồn tại => thực hiện thêm push : result.push(cate)**

1. **Project 04** 
   1. **Demo :**
   2. **Thực hiện thiết kế giao diện:**
      1. **Chia ra thành control component và list component**
      2. **Chú ý khi dùng class pull left hoặc pull right thì cần có class clearfix**
      3. **Thiết kế list component sử dụng thumbai trong bootstrap là ok**
   3. **Thực hiện .ts**
      1. Vì chưa học về cách kết hợp với cơ sở dữ liêu lên có thể khai báo thành courses : any[] = []; cũng được nhưng lần này dùng cái hay hơn đó là định nghĩa các interface hay cac class cho nó

**Phần 5 :**

1. **Directives :**  được chia thành 3 loại
   1. **Components : thường thao tác với các loại template**
   2. **Attribute directives : nhóm sử dụng để thực hiện project 05 : được sử dụng để thay đổi các thuộc tính của các đối tượng như ngStyle , ngClass ,**
   3. **Structural directives : thêm vào hoặc xóa đi các thành phần Dom của elements nào đó : ngFor , NgIf, NgSwitch**
   4. **Derectives : giúp nhận thêm hành động từ phía người dùng**
2. **Bài tập hightlingth :**
   1. **Demo : **
   2. **Hướng làm:**
      1. Khi click vào button thì truyền mầu của nó vào cho method và thay đổi mầu sắc đó
      2. Tạo mới 1 tập tin derective : muốn sử dụng derective đó ở đâu thì lấy luôn selector đó đặt vào vị trí mong muốn
      3. Tạo 1 contructor ở directive đó chú ý cần để là private el:ElementRef **, private renderer: Renderer Chú ý là để sử dụng được nó cần import nó vào**
      4. **el.nativeElement.style.backgroundColor = 'yellow'; hãy in từng phần 1 ra sẽ thấy những điều đặc biệt**
      5. **nếu muốn lấy nội dung thì innerText , innerHTML**
      6. **renderer.setElementStyle(el.nativeElement,'cursor','pointer'); cái này cũng thay đổi được**
      7. **thay vì viết trong constructor thì mình có thể viết ở ngoài thành 1 hàm**
      8. **đối với các sự kiện di chuyển chuột vào và di chuyển chuột ra thì cần có @HostListener(‘mouseenter’) onMouseEnter(){}**
      9. **di chuyển chuột ra @HostListener(‘mouseleave’) onMouseLeve(){}**
      10. **tìm hiểu thêm về HostListener : từ khóa angular2 HostListener** 
          * **thấy :**

**@HostListener('click',['$event.target']) onclick(btn){**

**console.log(btn);**

**}**

**Nó chính là thẻ mà directive gắn vào**

* + - * **truy cập được các nội dung khi click ví dụ lấy innerHTML hoặc innerText**
  1. **Tìm hiểu sâu hơn** 
     1. **Từ**  khóa use angular2 directive in host of another
     2. **Thấy cái này hay hay :**

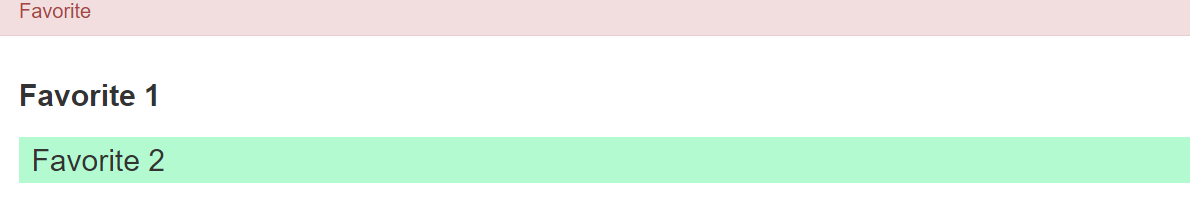
**host: {**

**'** **(click) ': 'onClick($event)'**

**}**

Nó chính là thuộc tính và bên trong là đối tượng biểu thị sự kiện click và phương thưcs

* 1. Truyền tham số vào cho các directive để nó mềm dẻo hơn thì thay cái selector ở được gắn thành [seletor]=”’Biến’”
     1. Biến này thường là giá trị mình muốn thay đổi ví dụ như mã mầu
     2. Truyền nó vào giống Input thì cần import Input , @input(“selector”) name(đặt lại tên) : string;
     3. Truyền từ component.HTml => derective
     4. Thường tạo 1 biến lưu trữ mầu mặc định
     5. Khi click vào button mầu đỏ thì hover chuot vào nó sẽ là mầu đỏ thì click vào truyền vào mầu sắc của nó mà mã mầu thay đổi ở component và chính là biến color vậy chỉ cần truyền vào biến color ở selector của directive là xong
  2. DefaultColor k lên để cứng như vậy mà nếu truyền vào nó khác rỗng thì lấy giá trị đó còn nếu = rỗng thì lấy giá trị mặc định ban đầu

1. Project : Faforice
   1. Demo 
   2. Decription :

Khi di chuyển vào thì thẻ H3 thứ 1 sẽ có thêm class ishover

Khi di chuyển vào thì thẻ h3 thứ 2 sẽ có thêm class là ishover và có thêm class là isfavorite

* 1. Làm
     1. Định nghĩa 2 class mới là ishover với isfavorite ở trong phần style
     2. Vấn đề làm sao để gắn các class này vào thì xây dựng directive
     3. Gắn selector của directive đó và truyền tham số cho nó hay gọi nó là attribute directive
     4. Truyền rồi thì phải viết hàm để nhận giá trị truyền vào Input
     5. Sử dụng **hostbiding** để tiêm vào các class
     6. Sử dụng HostListener để lắng nghe khi hover và khi k hover

1. Project Expand and collapse contents
   1. Demo
   2. Dicripttion:
      1. Có 3 khối giao diện
      2. Khối thứ 1 và 2 ở trạng thái hiển thị toàn bộ nội dung
      3. Khối thứ 3 ở trạng thái đóng nội dung
      4. Khi click vào các khối thì nó sẽ show ra giao diện đó
   3. Hướng giải quyết
      * 1. Cần tạo các class để phù hợp với đề bài
        2. Để selector của directive vào phần giao diện phù hợp truyền các tham số vào vào cho nó , nếu = true thì hiện ra còn = false thì ẩn đi
        3. Hoặc k truyền và truyền false truyền thì phần directive phải khai báo input để nhận
        4. Phần **hostbinding** để thêm vào các class cần thiết và class đó là thể hiện việc ẩn và hiện vậy là nhét nó vào là ok . hay hiểu là nếu iscollap == true thì class được gắn vào . => do đó thiết lập this.is-collap == true thì được truyền vào
        5. Đặt thêm if để kiểm tra nếu k truyền giá trị gì vào thì cho nó = true
        6. Để bắt sự kiện click thì gọi đến hostListener gọi đến phương thức do ta xây dựng : nhìn thì thấy khi click thì giá trị true , false thay đổi
        7. Khi gắn selector của directive và truyền vào tham số thì đó chính là input bây giờ muốn nó thay đổi gì đó khi thực hiện click thì nó lại là output đó
        8. Khi có sự tương tác (sự kiện click ) thì chúng ta có isCollap là chúng ta lấy nó và đẩy về cho components , component đọc được và đưa ra những icon phù hợp
        9. K phải gắn cố định như này : *colappded :boolean = true;* mà phải dựa vào kết quả của directive hay nói cách khác là phải nhận kết quả trả về từ sự kiện click
        10. Phải thiết lập output (collapsed) = “onCollapsedChange($event)”
        11. Truyền di theo EventEmitter và đẩy kết quả của tk ở phía directive - click đó đi *this.collapsed.emit(this.isCollapsed);*
        12. *In thử ra cái hàm onCollapsedChange()*  ở phía .ts sẽ thấy truyền được giữa directive và component rồi
   4. K chơi kiểu như trên mà xóa hết phần output và sử dụng cách thực hiện mới
      1. Chỉ cần biết vùng mình đang sử dụng ở trạng thái nào
      2. Mà nó dựa vào giá trị trong directive nếu lấy được giá trị đó trong HTML là vấn đề đã được giải quyết => vậy đặt biến cho nó là #zvnColapp1 nó tương đương với toàn bộ directive và có thể truy cập vào được giá trị , thuộc tính trong đó
      3. Và phải đặt biến tạm đúng = directive và chú ý cần khai báo thêm thuộc tính exportAs: “tên của directive”

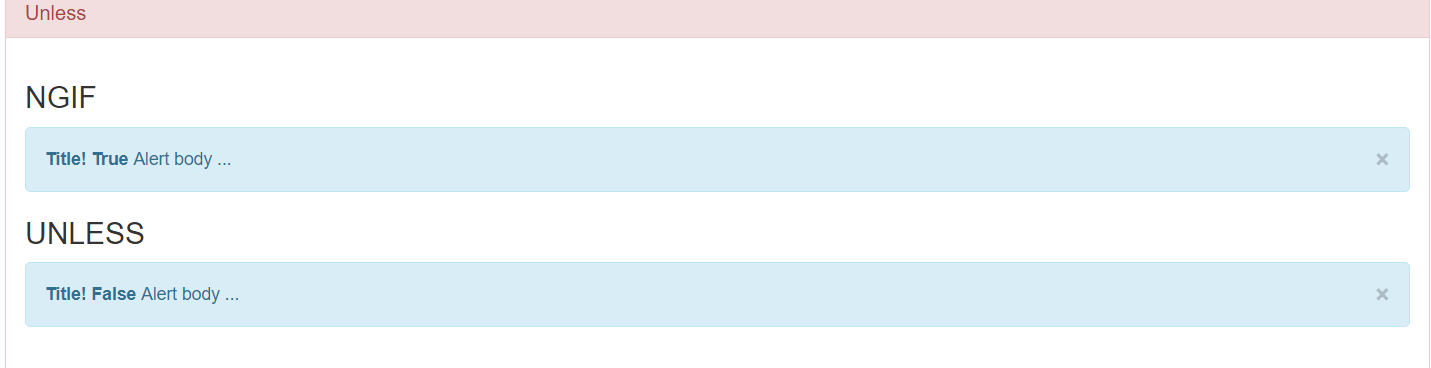
**Phần 6 :**

1. **Structural Directive :**  thuộc về phần cấu trúc bên trong của Element nào đó. Liên tưởng đến ngIf, ngFor, ngSwitch
   1. **Đưa ra 3 cách viết duyệt mảng :**
      1. ***<li class="list-group-item" \*ngFor="let student of students">{{ student }}</li*>**
      2. **<li class="list-group-item" template="ngFor let student of students">{{ student }}</li>**
      3. <ng-template ngFor let-student [ngForOf]="students" >

<li class="list-group-item" >{{ student }}</li>

</ng-template>

* 1. **Từ** 1 dịch ra 2 dịch ra 3
  2. **Tìm hiểu thêm nhé hihi**

1. **Directive UnLess**  : phủ định lại 1 cái gì đó
   1. **Demo : **
   2. **ngIf = true**  thì hiển thị . Hãy xây dựng 1 structuar directive phủ định lại cái ngIf kia
   3. **thực hiện**
      1. đặt selector của directive vào vị trí giao diện cần tùy biến có điều là phải có dấu \* đó
      2. chưa biết viết ntn và viết từ đâu thì vào doc đọc xem API=>

ngIF rồi xem contructor của nó có những cái gì

* + 1. sử dụng tài liệu structual directive thì thấy cái contructor của nó :

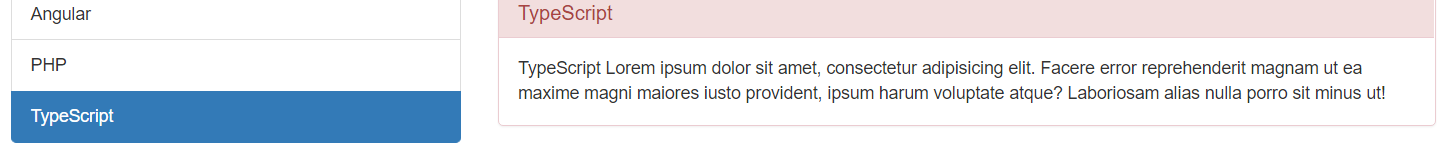
constructor(

private templateRef: TemplateRef<any>,

private viewContainer: ViewContainerRef) { }

**hay hay đúng k in nó ra xem là cái gì nhé có 1 đống linh tinh cách thứ**

* + 1. chính là chúng ta tiêm vào 2 cái tk bên trên giờ làm sao để cho nó hiển thị thì :
       1. this.viewContainer.createEmbeddedView(this.templateRef);

1. Project Delay
   1. Demo
   2. Decription : cứ sau khoảng 1s thì các phần tử sẽ đổ ra
   3. Giải quyết xây dựng delay directive là ok
      1. Gắn selector của direct tive vào vị trí giao diện truyền số time hiển thị
      2. Thiết lập hiển thị dựa vào 2 đối tượng TemplateRef, ViewContainerRef
      3. Sử dụng hàm setTimeout() theo kiểu arrow function để hiển thị theo sau bao nhiêu giây nhưng nó lại hiển thị cả 1 đống cùng 1 lúc
      4. Để hiển thị từng phần tử thì cần lấy số giây nhân với chỉ số của phần tử ở bên ngoài giao diện ấy
2. Tab components
   1. Demo : 
   2. Decription : click vào đâu thì nó sẽ hiển nội dung của phần đấy ra
   3. Giải quyết
      1. Chia nhỏ phần list thành 1 directive và truyền vào 1 tham số tương tự như id để khi click thì nhận biết được nó vd : zvnselector = “php” và truyền thêm 1 cái nữa để nhận biết khi nào thì active [active]=”true”
      2. Truyền vào thì phải khai báo input và sử dụng **hostbingding** để gắn vào các class cho nó (chính là class active)
      3. Sử dụng **hostlistenner** để lắng nghe sự kiện click
      4. Sử dụng attribute directive (directive zvnContain)để gắng vào các nội dung bên phải và để biết khi nào cái nào được chọn thì dựa vào id để chọn =>
      5. Và cũng phải truyền tk active để nó biết nó còn hiển thị
      6. Sử dụng hostbinding để gắn thêm hidden vào cho cái có nội dung tương ứng vì hidden nó có thêm dislay = none ở bootstrap nên sử dụng nó
         * 2 mục mới độc lập với nhau => cần xây dựng 1 cầu nối giữa 2 directive này (input-output k dùng cach này mà chơi cách xây dựng 1 directive cơ )
         * Xây dựng zvnTabContainer directive bao bọc bên ngoài 2 directive con kia vậy thì phần định nghĩa directive cha cần phải có phần chứa 2 directive đó serlecto : thuộc kiểu directive 1 , contrain :thuộc kiểu directive thứ 2
         * Khi tạo ra thì 2 cái selectos và contains đều = rỗng và phải gắn các directive loại 1 ,directive loại 2 vào bằng cách tạo method addSelector ở tk directive cha
         * Vậy khi gọi tk directive 1 thì làm sao có thể truyền vào được . Vậy ta phải dùng cách tạo contructor ở tk directive 1 đó truyền vào kiểu cha của nó và trong method gọi phương thức của tk cha ok
         * Tương tự với phần directive contain cũng thực hiện thêm mới vào cho nó
         * Đến đây có được danh sách các selector và danh sách các contain và giờ cho nó hiện lên qua tk click mà click thì nằm ở directive selector (bên trái) mà tk quản lí lại là tk cha của nó và nó lại lằm trong phần contructor nhớ thêm private cho nó để coi nó là 1 thuộc tính và dùng this gọi được
         * Và trong method click cần phải 1 method (show )nữa để nó hiển thị vì nó quản lí cả selector và contain nên khi đưa id vào thì nó biết được cần hiển thị tk nào ra
         * Phương thức show() phải được viết ở chỗ tk cha nhiệm vụ xử lí 2 công việc : active tk bên trái và hiện nội dung ở phía bên phải
           1. Để active thì duyệt qua các phần tử trong selectors rồi xem phần tử nào có id = giá trị click thì thay đổi active cho nó = true còn k = thì thay = false thực hiện sử dụng foreach
           2. Tương tự với phần contains là ok

**=) đến đây là giải quyết xong rồi nhưng muốn nâng cấp cao hơn : nghĩa là bỏ phần constructor của tk directive thứ 2 (contain) đồng ngh**ĩa với việc bỏ hết các hàm addContain , giá trị contains ,

**Sử dụng khái niệm mới @ContainChildrent(tên direcive thứ 2) tên mới : kiểu dữ liệu mới < direcive thứ 2>(querylist)**

**KẾT Luận : đến đây mới thấy hóa ra là nó dùng các thuộc tính của css để làm nó ẩn và hiện đi thôi chứ nếu mà muốn nó hoàn toàn k được load vào cơ thì phải dùng đến tructual directive mời xem bài phía dưới**

1. **Tabs Sử dụng** Structual directive
   1. **Trong phần directive thứ 2** : contructor truyền vào 2 cái TemolateRef và ViewContainerRef. Và thiết lập phần giá trị nhận vào
   2. **Chuỗi thì phải đặt trong dấu nháy đơn** : "'typescript'"
   3. **Khi thay đổi giá trị thì phải xuất hiện nội dung xây dựng 2 hàm hiện và ẩn (đọc lại câu lệnh phần struc để hiểu kĩ hơn)**
   4. **Trong phần quản lí cần chỉnh lại 1 chút là nếu như contain.id = id truyền vào thì cho hiển thị (phương thức hiển thị ấy)**
   5. **Để giá trị mặc định khi hiển thị**
      1. **Duyệt qua các selector và xem cái nào có active == true thì lấy id của nó dựa vào id đó hiển thị nó ra**

**Phần 7 :**

1. **Lifecycle Hooks : vòng đời**
   1. Đối với các lập trình viên thì dựa vào các hooks để đưa ra các xử lí phù hợp
   2. Chúng ta k cần dùng tất cả các gián điệp (interface) , địa điểm (hook) , Cách dụ dỗ (hook method)
   3. Tìm hiểu các hooks
      1. Constructor được gọi đầu tiên gọi trước tất cả các hooks
      2. **ngOnChanges () được** gọi đầu tiên: và nó được thực hiện khi có sự thay đổi. và quản lí thông qua đối tượng  **SimpleChanges > cứ**  mỗi lần gọi thì ngDoCheck được gọi theo
      3. ngOnInit() : dùng trong quá trình khởi tạo các giá trị
      4. **ngDoCheck() : thường ít sử dụng**  và kiểm tra xem có sự thay đổi gì ko
      5. **ngAfterContentInit() :**  chỉ dành cho components :
      6. **ngAfterContentChecked() :** chỉ dành cho components : in ra được nội dung
      7. **ngAfterViewInit(): chỉ gọi 1 lần**
      8. **ngAfterViewChecked()**
      9. **ngOnDestroy() : dùng trong khi hủy các giá trị nếu như cần thiết**

**Muốn lấy 1 nội dung của 1 thẻ gì đó thì cần đặt tên cho nó nếu nó là con của component thì phải khai báo @ContentChild(‘myChild’) myChild : ElementRef**

1. **Service và DI**
   1. **Dependency Injection :** 
      1. giúp giảm sự phụ thuộc giữa các lớp, tạo ra sự linh hoạt cao và tính dễ bảo trì.
      2. Lợi ích : Giảm sự phụ thuộc giữa các class, tăng khả năng dùng lại và khả năng dễ đọc của mã nguồn, tạo cho ứng dụng
      3. Ví dụ khi xây dựng ứng dụng game :playtaition
         1. Khai báo class
            1. Trong class phải có phương thức chơi play(){}
            2. **Mỗi** trò chơi thì định nghĩa 1 class
            3. **Cần tạo 1 interface rồi tạp class rồi implement từ class đó**
            4. **Phần contructror () truyền vào thực tiếp. đối tượng xây dựng ở bên ngoài**
   2. **Service**
      1. **Cùng 1 xử lí mà khai báo 2 lần thì k ổn mà dùng service sẽ ok hơn**
      2. **Để** gọi được đến service vừa tạo thì cần import service đó và tạo 1 biến kiểu đó
      3. **Và** contructor sử dụng DI : khai báo 1 biến thuộc kiểu đó và this.bien = bien truyen vao
      4. **Cách viết ngắn :**

**constructor(private logService : LogServiveService) { }**

* + 1. **Khi khai báo xong thì** method chỉ cần gọi trực tiếp trong service là xong
    2. **Và nhớ phải đăng kí nó ở trong app.module.ts**
    3. **Chú ý 2 component cùng sử dụng 1 service thì nó là 1 thể hiện khi khai báo ở app.modul.ts**
  1. **Bt Caculartor**
     1. **Constructor dùng dể tiêm vào các service => vậy cần tiêm bao nhiêu servive thì tạo bấy nhiêu ở constructor là ok**
  2. **Bài tập**  Course
     1. **Demo**
     2. **Thực hiện**

**Khởi tạo 2 mảng từ ngonInit thông qua services**

* + - * 1. **Service thực hiện công việc**  lấy ra danh sách các khóa học : cần 1 biến course : string[];
        2. **Construc tor thực hiện gán các phần tử**
        3. **1 method trả về danh sách các khóa học**
        4. **1 method thêm mới các khóa học**
  1. **Bài tập : Project View video programming on Youtube**
     1. **Demo :**
     2. **Xây dựng giao diện**
        1. **Chia làm 2 phần phần bên trái chiếm 5 phần bên phải chiếm 7**
        2. **Cấu trúc bên trái :** div bên trong là 1 thẻ a có input = checkbox và khi xem rồi thì thẻ a có thêm class là seen và 1 thẻ a nữa khi click vào thì có thêm class là active
        3. **Chia thành 2 component nhỏ**
     3. **Cần xây dựng 1 services để tạo 1 dịch vụ cung cấp dữ liệu cho components**
        1. **Xây d**ựng dịch vụ dành cho video: sẽ cung cấp danh sách các video cho componetts vì vậy tại service sẽ lấy các danh sách video về và mỗi phần tử trong class đó là 1 thể hiện của video
        2. **Xây dựng class video**
        3. **Xây dựng video-service** :
           1. **Cần chứa các danh sách các video thì tạo nó ở arr**
           2. **Trong constructor push video đó vào push theo kiểu id , name giá trị thứ 3 k cần vì để mặc định là false**
           3. **Tuy nhiên k tao trong constuctor kiểu vậy mà tạo 1 method riêng rồi trong constructor gọi lại nó thôi**
           4. **Method lấy ra các video hiện có getVideos(){}**
           5. **Đã xong và muốn sử dụng được nó cần xác định xem dùng nó ở đâu- khai báo thành phần proviteder trong youtobe-component**
        4. **Để sử dụng được video-service ở youtobe cần** khai báo providers và tiêm nó vào chỗ constructor
        5. **Để chỉnh chu hơn thì tạo video mặc định thì tạo 1 method getcurrentVideo ở phần services và lấy phần tử nào tùy ý**
        6. **Tại youtobe-components thì lấy phần tử hiện hành Currenvideo đó ra và truyền ra bên ngoài sử dụng input – output khi có rồi thì chỉ cần gán lại ra ngoài giao diện nữa là ok**
           1. **Nhưng nếu currentVideo = null thì lấy gì ra mà truyền vậy dùng ngIf để kiểm tra cho kĩ**
        7. **Phần đổ giao diện cho layer thì chỉ cần chỉnh : [src]="'https://www.youtube.com/embed/' + currentVideo.id"**
           1. **Nhưng lại xuất hiện lỗi : unsafe value used in a resource URL context coppy vất lên gg tìm ra xây dựng 1 pipe nhé**
        8. **Phần outline components : lấy ra các danh sách**
           1. **Cần sử dụng video services:**

**Import service**

**Tiêm vào constructor**

* + - * 1. **File ts cần lấy ra tất cả các video thông qua services**
        2. **Lấy được rồi thì đổ nó ra phía ngoài là ok**
        3. **Xử lý trường hợp có thêm class seen : seen khi video.seen == true**

**Sử dụng ngClass để gắn thêm class cho nó**

* + - * 1. **Trường hợp acc tive khi video nằm ở bên trái == video mà chúng ta đang xem**

**Khi người dùng click vào 1 video nào đó thì video đó là video chúng ta đang xem**

**Lấy ví dụ là tk curren video thì ở youtobe-component có rồi truyền cho tk outline**

**Mục đích để lấy video.id == currentVideo.id**

* + - * 1. **Xử lý** sự kiện click ở bên tay trái thì nội dung ở bên tay phải cũng phải thay đổi theo

Nghĩa là phải bắt được sự kiện click thông qua method

Truyền vào input ở phía trên rồi giờ trả về cho nó 1 output truyền cái video vừa click nó ra ngoài cho tk cha và cập nhập lại giá trị hiện hành current video cho nó là nó tự thay đổi theo

Xử lí trường hợp xem rồi thì nó thay đổi mầu sắc rất đơn giản khi thẻ a được click vào thì thay đổi thuộc tính seen cho nó : video.seen == true

Xử lí video nào xem rồi thì cho nó cái nút check thì xem hay chưa nó thuộc vào video.seen

Xử lí khi click vào check box k ảnh hưởng đến thẻ a ở bên ngoài

**event.stopPropagation();**

**cập nhập lại giá trị check box : video.seen nếu là true thì cập nhập thành false và ngược lại**

* + - 1. **xử lý cookies**
         1. **khi xem rồi mà f5 lại thì thông tin k còn được lưu trữ nữa mà nó lại đưa về trạng thái ban đầu**
         2. **vậy dùng cookies để lưu lại**

**sử dụng npm –cookies**

**câu lệnh npm install ng2-cookies sẽ thấy có thêm tập tin ng2-cookie ở node module**

**chưa thành công gg tiếp angular2 cookie ra trang này :**

[**https://www.npmjs.com/package/angular2-cookie**](https://www.npmjs.com/package/angular2-cookie)

**Xóa cái mà mình lỡ cài vào rồi : gg npm uninstall package**

**Đến đây thì cũng chả dùng được và phải dùng cái thư viện mới là ngx-cookie**

**Việc sử dụng cookie chỉ ở outline thôi**

**Sử dụng get, put , set để lưu trữ cookies**

**Khai báo nó thành hằng số để sau này có gì chỉnh sửa cho nó dễ**

**Phương thức put sẽ chứa danh sách id của những video đã xem mà danh sách lại ở trạng thái array thì chuyển nó sang json là ok JSON.stringify(videoIDs)**

**Method getCookie sẽ lấy ra các chuỗi đó bằng cách biến nó thành đối tượng**

**Khi vừa vào có được danh sách các ID mà chúng ta đã xem**

**Nghĩa là lấy các id mà đã get được ra rồi đem so sánh với các id cái nào trùng thì cho nó thêm class = seen**

**Method sync(){**

**Lấy ra danh sách các cookies**

**Nếu chuỗi = rỗng sẽ có lỗi vì thế cần if( khac null)**

**if(this.historuIDs.indexOf(this.videos[i].id) != -1){**

**this.videos[i].seen = true ;**

**}**

**Kiểm tra xem nó có tồn tại hay k nếu có thì cho nó == true**

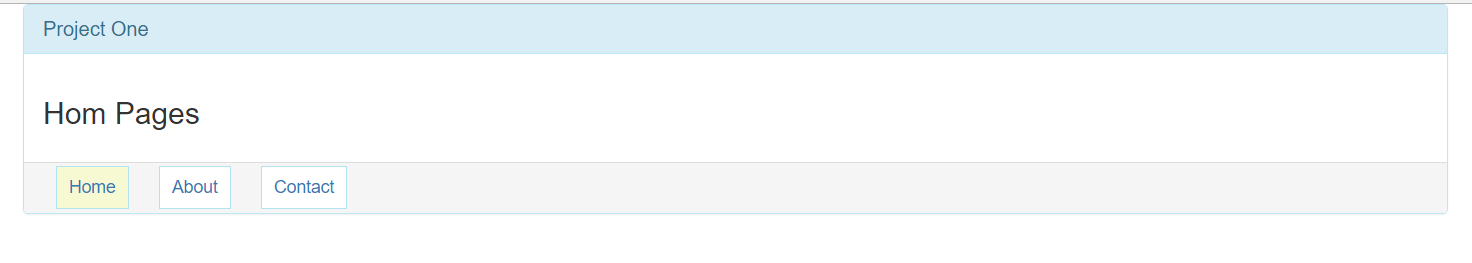
**Để lưu vào trong cookie thì phải lưu vào trong cookieIDs: method aClick thẻ a được click thì có video rồi => có id của nó và thực hiện push vào cookiesID là ok nhưng để chặt chẽ hơn khi click vào 1 lần r mà click thêm 1 lần nữa thì nó push 2 lần mất vì thế cần kiểm tra tồn tai hay k tồn tại : this.cookieIDs.indexOf(video.id)**

**Tạo 1 method lưu cookies vào cookies của trình duyệt**

* + - 1. **Xử lí click và bỏ click ở ô check box nghĩa là thêm và xóa cookies theo ô check box đó thì xem method ckbClick()**
         1. **Nếu video.seen == true đã xem mà click vào thì phải xóa cookie đi và ngược lại thì phải add vào mà add chính là ohaanf ở thẻ aClick rồi thì phải xây dựng 1 method addVideoCookies**
         2. **Xây dựng method xóa cookies : removeVideoFromCookieArr**
      2. **Nâng cấp ứng dụng :** 
         1. **K dùng iframe để nhúng các video youtobe nữa**
         2. [**https://www.npmjs.com/package/ngx-youtube**](https://www.npmjs.com/package/ngx-youtube)

**Phần 8 :**

**Bài tập 1 :** xây dựng chức năng home-about-contact Page

1. De mo : 
2. Thực hiện
   1. Để sử dụng được nó cần có giá trị : <base href="/"> ở index.html đặt trong head
   2. Phần Define router có cái gì đó hay hay thì đọc và config nó

const appRoutes: Routes = [

{ path: '', component: HomeComponent },

{ path: 'about', component: AboutComponent },

];

Giải thích : path “” là đường dẫn mặc định cái path2 là đường dẫn con

* 1. Khi khai báo xong đường dẫn thì

<router-outlet></router-outlet> : sẽ hiển thị nội dung

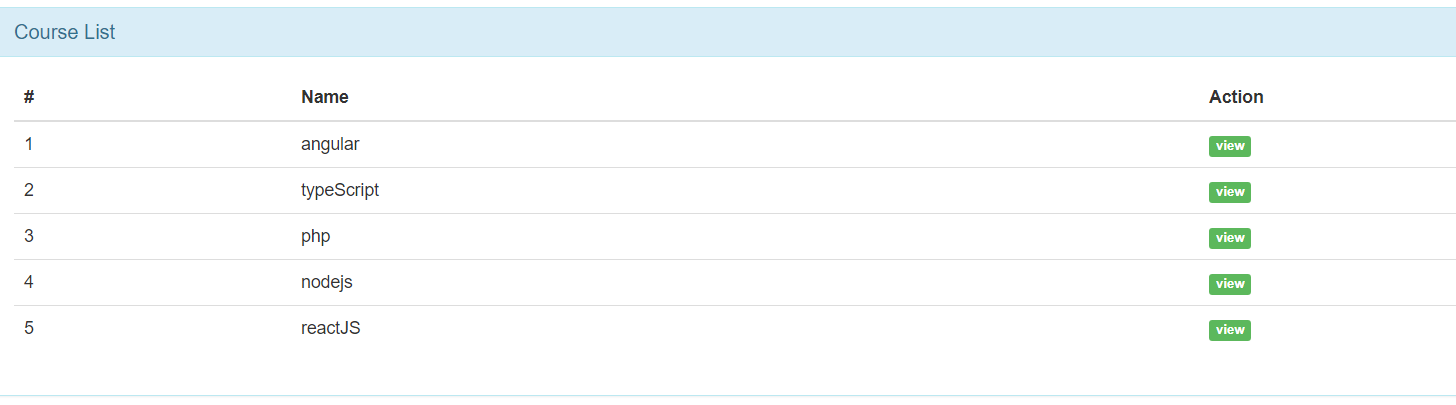
* 1. Trong trường hợp url k tồn tại thì path:’\*\*’ , component : NotFoud
  2. Khi xây dựng menu thì thường dùng thẻ a vs thuộc tính href nhưng bị load lại trang để đảm bảo angular thì người ta dùng router link : routerLink="/crisis-center"
  3. Khi đang ở trang contact thì phải có nội dung active cho nó thì dùng : [routerLinkActive](https://angular.io/api/router/RouterLinkActive)="active" nhưng sẽ có lỗi để fix lỗi này : [routerLinkActiveOptions] = "{exact:true}"
  4. Thực hiện click vào các menu con thì sử dụng routerLink là xong nhưng cách khác click vào thì phải tiêm vào và sử dụng như DI :

goContact(){

this.router.navigate(['/contact']);

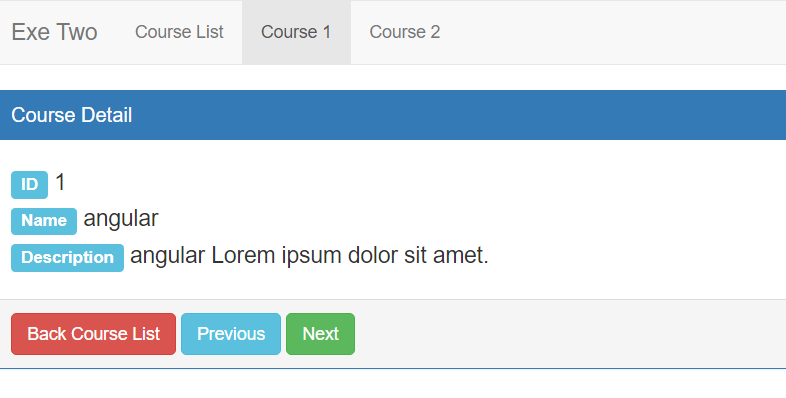
}

Bài tập 2 :

1. Đemo : 
2. Các bược thực hiện
   1. Service : thực hiện theo đúng tiêu chuẩn DI
      1. Method getCourse() : trả về tất cả các khóa học
      2. Method getCourseByID() : lấy thông tin của 1 khóa học dựa vào id của khóa học đó . Tại sao cần method này vì khi click vào view thì giá trị lấy ra được là id của khóa học , tên , phần miêu tả
   2. Cấu hình router để khi vào trang nó sẽ tự động chuyển đến trang con : từ khóa **redirectTo**
   3. Cấu hình course/c01 thì phải khai báo lại chú ý đặt thêm tham số cho nó : { path: 'course/:id', component: CourseDetailComponent },  
      : chú ý thành phần nào càng chi tiết thì đặt ở đường dẫn đầu tiên, phần nào càng chung chung thì đặt ở phía dưới
   4. Thực hiện gắn link **[routerLink]="['/courses'] ,** để menu active :

**routerLinkActive="active" gắn ở phía ngoài**

* 1. Truyền tham số cho routerLink **[routerLink]="['/course', 'C01']**
  2. Thực hiện click vào view thì nó chuyển đến trang thông tin chi tiết của nó . Vậy xác định chỗ nào xuất hiện nút view đó couse-list có method onSelect(course ) : để biết chuyển hướng đến trang nào
  3. Thực hiện việc chuyển hướng bằng method **this.\_routerServices.navigate(['/course',courseID])** mà router là 1 router-sevices thì phải import và tiêm vào constructor của nó giống DI
  4. Từ id nằm ở URL lấy ra thông tin của khóa học và đổ vào vị trí từ khóa : **actived Router : là**  1 services : thì phải
     1. Import [ActivatedRoute](https://angular.io/api/router/ActivatedRoute)
     2. tiêm vào constructor
     3. khởi tạo thử giá trị ở thần ngOnInit(){
     4. **this.\_activatedRouteService.snapshot.params['id'];** lấy ra được id nhưng mà params phần tử id hay nói cách khác là cái tham số truyền vào ở
     5. { path: 'course/:id', component: CourseDetailComponent },
     6. Chú Ý nếu sử dụng snapshot thì id khi sử dụng sẽ k được cập nhập lại
  5. CanActive : bảo mật gì đó cứ tài liệu angular mà đọc : giới hạn phần id : : nghĩa là k cho phép xem thông tin khóa học có id = “C01”
     1. Hoc theo cách viết của họ method canActive : tra ve true có quyền truy cập , k quyền truy câp trả về false
     2. Tạo 1 services : course-detail-guard.service.ts
     3. Đọc thêm ở API : canActivate
     4. Khai báo thêm routing

1. **CÁCh CHIA NHỎ MODULE** 
   1. 1 Angular module bản chất là 1 class mà có thêm NgModule trong đó cần khai báo import, declarations, bootstrap, providers
      1. Declarations : components, directives , pipe muốn sử dụng được thì phải đăng kí trong Declarations
      2. Import : để cấu hình ,xây dựng được ứng dụng- 1 hoặc nhiều module phụ thuộc vào thành phần khác , import khi nào cần thiết và dùng nó thì mới import
      3. Providers : khai báo các services
      4. Bootstrap : khi components chạy thì thành phần nào mà muốn nó định nghĩa đầu tiên thì định nghĩa trong phần bootstrap
      5. Export: components , directives pipe ko bao giờ được dùng services
   2. **Feature Module :** 
      1. **Từ 1 module chạy chính kéo vào các module nhỏ**
      2. **Từ 1 tập tin chạy chính router chạy chính : RouterModule.forRoot()**
      3. **Phần con forChild()**
   3. **Các chia sẻ các module**
      1. **Tạo 1 thư mục chứa nhiều foder**
      2. **Tạo tập tin shared.module.ts :** 
         1. **Import pipe. Diective cần thiết**
         2. **Khai báo declartions**
         3. **Muốn chia sẻ phần nào thì , exports :[] nó ra**
      3. **Giờ ở 1 module khác muốn dùng nó thì import nó vào là ok rồi hihi**
2. **Bài tập 3 là nâng cấp của bài tập 2** 
   1. **Demo **
   2. **Thay đổi ID của router thành kiểu number => định nghĩa lại phần course interface**
   3. **Tìm hiểu về navigateByUrl : this.\_routerServices.navigate(['course/' + courseID]);**
   4. **Tìm hiểu về Subscription: lắng nghe sự thay đổi để khi next thì sẽ hiển thị dúng thông tin** 
      1. **Cần import *import { Subscription } from 'rxjs/Rx';***
      2. ***Nhớ tạo ngOnDestroy() để tăng tốc độ cho web***
      3. ***Đọc thêm để hiểu hơn***
      4. ***Tránh lỗi = cách giới hạn lại id vì nếu next tới ngoài khoảng id cho phép là lỗi ngay***
   5. **Tìm hiểu về Query Param**
      1. **Gắn nó vào [queryParams]="{text : 'navbar', page : 2}" phần menu**
      2. **Cách 2 truyền qua navigate :**

**this.\_routerServices.navigate(['/course',courseID],{queryParams: {from : 'course-list'}});**

* + 1. **Cách lấy thông tin param trên url ra *this.\_activeRouterService.queryParams.subscribe***

**Lấy ra rồi nhưng cũng lên tạo ra subscrition để hủy**

* 1. **Tìm hiểu về** CanDeActivate : khi thực hiện 1 hành động gì đó thì nó mới cho vào hoặc ra khỏi router . VÀ tương tự CanActive thì cần tạo 1 services implement từ CanDeActive và gg thêm để hiểu thêm cách viết : Chú ý phần genergit là phần đứng trước của CanDeactivate : **{ path: 'course/:id', component: CourseDetailComponent, canDeactivate: [CourseDetailDeactivateService], },**

1. **Bai tap 4 : Tim Hieu Child Router : Bài tập 4 là bản năng cấp bài tập 3** 
   1. **Viết url dạng** 
      1. [**http://localhost**](http://localhost) **: 3000/courses List =>**

**ulr dạng courses + “ “ List CourseListComponent**

* + 1. **i. http://localhost : 3000/courses/2 Info =>**

**url dạng courrses + “/:id” Info CourseComponent**

* + 1. **Sử dụng ChildRouter:**

**{**

**Path : ‘courses’,**

**Children : [**

**{path : ‘’ , component : CourseListComponent},**

**{path : ‘:id’ , component : CourseComponent}**

**]**

**}**

***Phần nâng cấp***

***url : course/2/edit Edit EditComponent***

***: course/2/ DetailComponent***

***Course/2/detail DetailComponent***

***Course/2/delete DeleteComponent***

**Hướng giải quyết :**

**/\* course + id + “”**

**course + id + “detail”**

**course + id + “edit”**

**course + id + “delete**

**\*/**

**=> nhận thấy có sự phân cấp cha con rõ ràng thì tạo childrouter là ok**

**{**

**path: 'course/:id',**

**component: CourseComponent,**

**children: [**

**{ path: '' , redirectTo: 'detail' },**

**{ path: 'detail', component: CourseDetailComponent },**

**{ path: 'edit' , component: CourseEditComponent },**

**{ path: 'delete', component: CourseDeleteComponent },**

**]**

**},**

**Chú ý đưa thêm router-outlet để đổ dữ liệu thích hợp**

**Click vào edit thi sử dụng navigate để gắn link**

**onSelectDelete(courseID : number) {**

**this.\_routerService.navigate([‘course’, courseID , ‘delete’])**

**}**

***Chuyển đến trang khác bằng cách mới :***

**this.\_routerService.navigate([‘edit’],{ relativeTo: this.\_activeRouteService.parent}) => nó lấy router cha + edit**

**Phần 8 :**

**1. Study HTTP:**

**Gửi hành động lên webservice thì nó trả về nội dung mà mình mong muốn**

**Sẽ k xây dựng phía backend mà sẽ sử dụng luôn dịch vụ cung cấp miễn phí**

1. **Mock API : cung cấp các dịch vụ : đọc kĩ phần doc**
2. **Tìm hiểu Mock API : tạo dữ liệu mẫu và in ra url : http://5a900bded515b50012005367.mockapi.io/api/angular2/v2/courses**
3. **Tim hieu ve postman :** gửi yêu cầu từ post man đến moc api
   1. **Cach mở :**
   2. **gửi url ở trên vào thì nó in ra hay ko**
   3. **Lấy ra thông tin cụ thể : http://5a900bded515b50012005367.mockapi.io/api/angular2/v2/courses/5**

**2. observable và Promise :**

**- thuộc về đồng bộ và bất đồng bộ**

**(ví dụ hành động A -> lấy từ get abc.com -> x)**

**Hành động B -> x -> y**

**Hành động C ->y ->finish**

**Giải thích x lấy 10s mới xong nhưng chưa hết 10s thì y lấy -> bất đồng bộ xảy ra**

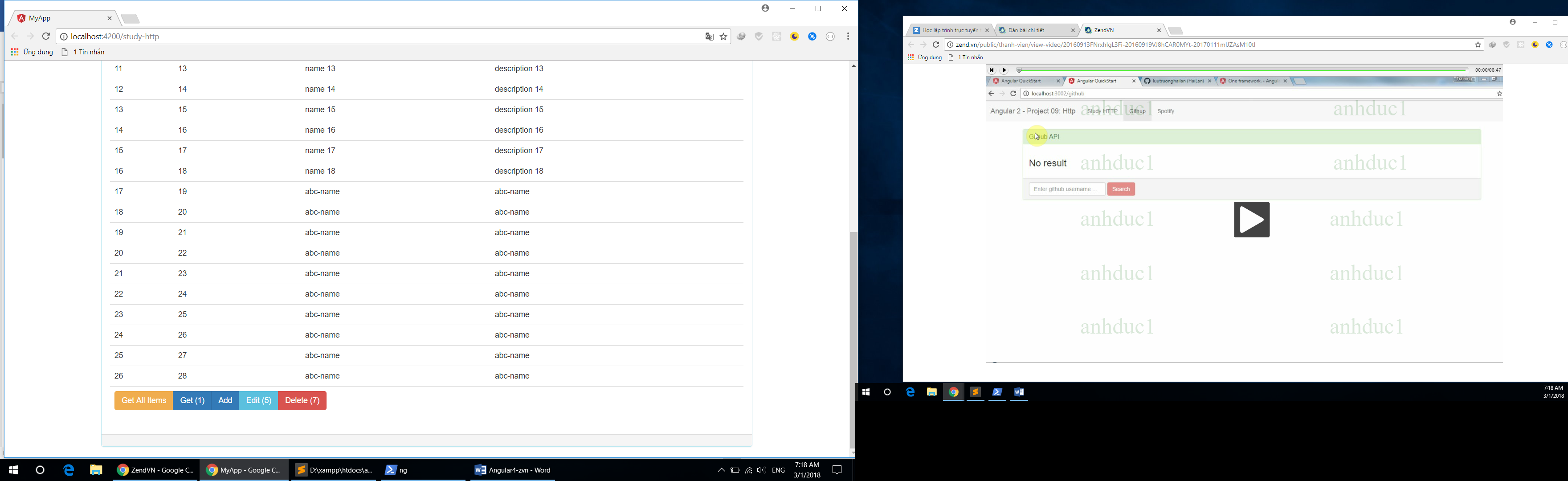
**- observable : là 1 arr lưu trữ giá trị vào tất cả các thời điểm , cái này sinh ra ES7 , muốn sử dụng được thì phải import Reactive Extendsion (RxJS)**

**- observable : dùng để giải quyết vấn đề về đồng bộ và bất đồng bộ, nó tổng quát hơn lên được dùng nhiều hơn về promise**

**- observable : nó truy cập đến 1 service và lấy giá trị trả về**

**3. Project 01**

**01. Demo**



**02. thực hiện**

**a. ấn vào nút getall thì lấy toàn bộ dữ liệu đổ ra**

ý tưởng sẽ là : từ component sẽ gọi đến service từ service sẽ gọi đến web service để lấy dữ liệu ra

* + - * + ở http.service sẽ có 1 method lấy dữ liệu từ URL : *getItems() : Observable<ICourse[]>{*

*return this.\_httpService.get('http://5a900bded515b50012005367.mockapi.io/api/angular2/v2/courses');}*

Nhận xét : để trả về kiểu **Observable<ICourse[]> : cần import :**

**import { Observable } from 'rxjs/Observable';**

**import { ICourse } from './../defines/course';**

**để** return this.\_httpService.get('http://5a900bded515b50012005367.mockapi.io/api/angular2/v2/courses' thì cần

import { HttpClient, HttpHeaders } from '@angular/common/http';

**còn nữa :**  ở phần module chạy chính cần :

**import { HttpClientModule } from '@angular/common/http';**

**trong component CourseListComponent method có method**

**//trả về all khóa học**

**getItems(){**

**this.\_httpService.getItems().subscribe(**

**(res : any) => console.log(res)**

**);**

**}**

**Trong service sẽ cũng có method**

**getItems() : any{**

**return this.\_httpService.get('http://5a900bded515b50012005367.mockapi.io/api/angular2/v2/courses')**

**Nhưng sau sẽ k để url dài quá như kia và cần sẽ có sự điều chỉnh nhận thấy các dạng url :**

// Method Action

// GET Get Items http://586fc8ad0474f212000b02c5.mockapi.io/api/angular2/v2/courses'

// GET Get Item http://586fc8ad0474f212000b02c5.mockapi.io/api/angular2/v2/courses/id

// POST Add Item http://586fc8ad0474f212000b02c5.mockapi.io/api/angular2/v2/courses/

// PUT Edit Item http://586fc8ad0474f212000b02c5.mockapi.io/api/angular2/v2/courses/:id

// DELETE Delete Item <http://586fc8ad0474f212000b02c5.mockapi.io/api/angular2/v2/courses/:id>

* + - * **nó** chỉ thay đổi ở đoạng cuối thay vì viết lặp đi lặp lại thì ta định nghĩa url ở phía ngoài
      * **private apiUrl = ‘’;**
  1. **ấn vào nút get(2) : lấy id của khóa học có id = 2** ra thì giống như ở bên trên thôi nó chỉ khác nhau ở cái id phía sau thì nối cho nó cái id vào là ok
  2. **Ấn vào nút Add thì sẽ thêm mới 1 khóa học** :

**Vậy**  sẽ phụ thuộc vào phương thức add Item và chúng ta sẽ truyền vào thông tin của 1 khóa học chứ k phải là id mà để id cho nó tự tăng

**Thì component**  có method thì service cũng phải có method như thế

**Theo tài liệu về mock API** thì dạng url để thêm mới 1 khóa học dùng post() nó truyền vào 3 tham số url – body (tham số truyền vào của hàm) – httpOptions

Chú ý : Đọc trong tài liệu ấy

const httpOptions = {

headers: new HttpHeaders({ 'Content-Type': 'application/json' })

};

Để có nó nhớ import :

import { HttpClient, HttpHeaders } from '@angular/common/http';

trong component thì thực hiển đẩy khóa học lấy được vào danh sách các khóa học lúc đầu

khi vừa vào trang thì hiển thị danh sách các khóa học hiện có thì nó hay hơn vậy chỉ cần khởi tạo bên ngOnInit

* 1. Xây dựng chức năng cho button Edit :
     1. Truyền vào 2 giá trị name , dicription ở bên component
     2. Bên service cũng phải có method này và tham số truyền vào phải có thêm id để biết edit cái nào chứ
     3. Để cập nhập dữ liệu ngay thì sau khi lấy được dữ liệu ra gọi lại method lấy ra các khoahs học . getItems(); nhưng nó lại phải truy cập vào serveice và cập nhập lại toàn bộ trang web và tốn tài nguyên
        + Vì thế chỉ cần so sánh giữa .
        + Tương tự với nút delete

4. BT 02 – GitHub : được chi thành 1 module

01. Demo

02. Thuc Hien

a. gg github api để xem nó có hỗ trợ gì k : muốn lấy cái gì thì chú ý phần đấy : ví dụ muốn lấy thông tin thì vùng user

b. muốn lấy repo thì quan sát vùng repo

- đổ dữ liệu 2 chiều thì dùng two-way-binding : => sử dụng ngmodel => cần import form=module

- > = 4 kí tự thì button sẽ cho phép ấn

- vừa vào cho flagDissable = true

- khi thay đổi giá trị ở ô in put thì method setActiveButton sẽ thực hiện kiểm tra xem nó > 5 kí tự chưa nếu rồi thì đặt lại thành false => để tránh trường hợp ghi đủ 5 kí tự rồi xóa đi mà nó vẫn active thì để trong hàm đó giá trị luôn là true

- nhận thấy có 2 vùng để đổ dữ liệu ra:

Profie : thông tin

Repos : các repository

**Phân tích tiếp :**

Vào api của github : từ url [*https://api.github.com*](https://api.github.com) lấy ra được 1 chuỗi json = > tiếp theo cụ thể hơn : là lấy 1 user cụ thể [*https://api.github.com/users/luutruonghailan*](https://api.github.com/users/luutruonghailan)=> hiện ra 1 chuỗi json và chúng ta muốn lấy thông tin gì thì lấy thôi vậy tôi muốn lấy : "name": "HaiLan","company": "ZendVN",

"blog": "www.zend.vn","location": "VietNam",

"email": null,

Và để tiện quản lí và phát triền hơn thì tôi xây dựng interface cho nó thực ra thì cũng k cần nhưng ngại xóa

Tiếp theo cần truy cập đến url đó để lấy về thông tin nhưng phải lấy thông qua service => method getUser() : giống http mà thôi

* + - * Thì ở bên component cần có method loadUser() ra
      * Thực ra lên nhận giá trị từ phần input và đẩy nó vào trong service rồi mới thực hiện tìm kiếm : xử lý :
        1. Tạo biến username
        2. Thiết lập giá trị cho username : setUsername(value : string)
      * Bên component : trước khi gọi getUser thì gọi setUsername trước và truyền vào giá trị cần tìm là this.username
    1. Lấy repos ra từ service
       - Method getRepos() dạng url : *GET /orgs/:org/repos*
    2. Lấy requyets nhiều hơn : để tránh limitted
       - * Vào github phần setting -> developer setting -> đăng kí 1 cái mới
         * Lấy được :

Client ID:

edddf807668d4306a575

Client Secret:

4a7d6e6f568e4e2e32b66ae3a3fa96b495694e3a

* + - * + Bổ xung vào phần services
        + Trong tài liệu github thì kết thúc yêu cầu nó có : *?client\_id=xxxx&client\_secret=yyyy*
        + Thì thêm nó vào method get

5. Spotify : tài liệu chính Spotify api

01. demo :

02. thực hiện:

Album.component.ts : hiển thị các bài track trong 1 album nào đó

Artist.component.ts : hiển thị các album của 1 tác giả nào đó

Search.component.ts : tìm kiếm tên của các tác giả

* + - * Luồng đi từ search -> Artist.component.ts sẽ hiển thị ra các thông tin -> album.component.ts : hiển thị xem album này có những track gì

1. Xây dựng chức năng search

Khi nhập vào ô input thì giá trị của nó phải được gửi đi theo link như thế nào thì phải vào trang : *https://developer.spotify.com/*

**Vào phần web api** :

* + - * + **Ser**vice thì cần tạo 1 apiURL

Phần 9 : Form

* + - * + Cần import vào FormModule
        + Sử dụng ngModel ngoài kiểu two-way data bingding thì nó còn thêm 1 vài class : ng
        + Lấy ra thuộc tính class : .className
        + Thuộc tính required : bắt buộc phải nhập
        + => dựa vào sự thay đổi class ở file note.txt => bổ xung css

input.ng-invalid : border 2px red ;

validate : kiểm tra hợp lệ hay ko

* gg : html input attributes : =>: có 1 thuộc tính **pattern (regexp**) => sử dụng email
* sử dụng minlength=’’ và maxlength=’’ cho fb
  + - * + ReSetFrom : tìm thẻ form : đặt cho nó 1 cái tên #formProfile=”ngForm”
        + Nút reset : truyền vào chính đối tượng của form đó (click)= “onResetForm(formProfile)”
        + Trong file .ts cũng có method onResetForm(formProfile : NgForm){

Log(formProfile) => in ra NgForm  
các control của nó

formProfile.reset

* + - * + }

**Phần 10 : Project Cuối khóa**

1. **Xây dựng backend:**
   1. **Dựa vào toàn bộ URL của playlist - > lấy được thông tin của Playlist -> cung cấp dữ liệu cho fire base -> ở ngoài angular tương tác với fire base và đổ ra dữ liệu**
      1. **Input** [**https://www.youtube.com/watch?v=xt5gQoKnAcU&list=PLv6GftO355AucJ4Td8\_6h007nQuVJQsPN**](https://www.youtube.com/watch?v=xt5gQoKnAcU&list=PLv6GftO355AucJ4Td8_6h007nQuVJQsPN)
      2. **Output : file Json chứa thông tin của playlist đó (danh sách video của play list đó )**
         * + **Làm sao lấy thông tin của playlist -> youtube Data API**

**B1 : lấy thông tin của video dựa vào video ID**

**B2 : từ thông tin lấy được phải biết cách ghi vào file json**

**B3 : kiểm tra video đã lưu chưa, chưa lưu thì lưu vào mà lưu rồi thì thôi k lưu nữa**

**B4 : lấy thông tin cơ bản của playlist dựa vào id của nó**

**B5 : lấy videos từ playlist ID**

**B6: lưu vào json**

**B7 : kiểm tra lưu được hay chưa**

1. **sao lấy thông tin của playlist -> youtube Data API**
   1. **gg rồi vào lập tài khoản rồi tạo key**
   2. **key đó là để lưu lại và gắn vào url**
   3. **vào phần tham chiếu (referent) => vào videos => tổng quan => list (cần lấy danh sách)**

[**https://www.googleapis.com/youtube/v3**](https://www.googleapis.com/youtube/v3)**/ => phần url cung cấp cố định => định nghĩa thành 1 hằng số**

**ZENDTUBE Frontend**

1. **phân tích module**
   1. **zendtube : module chính của ứng dụng**
   2. **Elements : module chứa các element/widget dùng chung cho toàn bộ ứng dụng**
   3. **Admin : module dành cho vùng chức năng admin**
   4. **Shear : module chia sẻ tài nguyên dùng chung**
2. **Phân chia giao diện và liệt kê chức năng cần thực hiện**
   1. **Element linh động hơn widget : title, Footer … những thành phần dùng chung**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Component** | **Module** | **Description** |
| **1** | widgetAds *zvn-widget-ads* | **Elements** | Component – hiển thị ra phần quảng cáo |
| **2** | widgetMenuTop *zvn-widget-menu-top* | **Elements** | Phần menu-top |
| **3** | widgetMenuMain  *zvn-widget-menu-main* | **Elements** | Phần menu-main |
| **4** | widgetMenuBottom  *zvn-widget-menu-bottom* | **Elements** | Phần menu-bottom |
| **5** | widgetBanner  *zvn-widget-banner* | **Elements** | **Phần** banner |
| **6** | WidgetSocial  *zvn-widget-social* | **Element** | **Phần** góc phải- fb-you |
| **7** | **Header**  **Zvn-widget-header** | **Element** | **Phần gồm 6 cái component trên** |

**Phần bên phải**

|  |  |  |  |
| --- | --- | --- | --- |
| **8** | **WidgetPlaylist**  ***zvn-widget-category*** | **Element**  Phần bên trên trước phần quảng cáo | **Thumbnail – Title - link** |
| **9** | **WidgetFeaturedVideo**  ***zvn-widget-featured*** | **Element phần dưới phần quảng cáo** | **Tên-hình ảnh –số lượng người bình luận – số lượng người xem – số lượng người đánh giá (rating video này) -> phân tích từ trang home mà ra** |
|  |  |  |  |

**Phần footer-top**

|  |  |  |  |
| --- | --- | --- | --- |
| **10** | **WidgetPopularVideo**  ***zvn-popular-video*** | **Element** | **Show contacht HTML** |
| **11** | **WidgetContact**  ***zvn-contact*** | **Element** |  |
| **12** | **WidgetFacebook**  ***zvn-facebook*** | **Element** |  |
| **13** | **ElmFooterTop *zvn-elm-footer-top*** | **Element** | **Là component của 10 11 12** |

**Phần footer-bottom**

|  |  |  |  |
| --- | --- | --- | --- |
| **14** | **ElmFooterBottom *zvn-elm-footer-bottom*** | **Element** | **Coppyright- Menu** |

**Phần nội dung chính :**

**Chia phần lập trình viên codeer-developer - > view more video**

|  |  |  |  |
| --- | --- | --- | --- |
| **15** | **ElmPlaylist *zvn-elm-video*** | **Element** | **Gồm panelTitle - Video** |
| **16** | **ElmPanneltitle**  ***Zvn-elm-title*** | **Element** | **title** |
| **17** | **ElmVideo**  **Zvn-elm-video** | **Element** |  |

**Xác đinh URL có thể có**

|  |  |  |
| --- | --- | --- |
| **URL** | **Component** | **DESCRiption** |
| **/** | **HomeComponent** | **Trang chủ** |
| **/video?page=1** | **VideosComponent** | **Thông tin chi tiết của video dựa vào id** |
| **/video/:id** | **VideosComponent** | **Thông tin chi tiết của video dựa vào ID** |
| **/playlists?page=1** | **PlaylistComponent** | **Danh sách tất cả playlist có phân trang** |
| **/playlist/:id** | **PlaylistComponent** | **Thông tin chi tiết của playlist dựa vào id** |
| **\*\*** | **ErrorCompont** | **Trang 404** |

**Key : để tìm hiểu angualr 4 có gì hay k : angular4 new features**

**Chia components xong rồi giờ thực hiện :**

**Phần quảng cáo widgets-ads:**

* + - * + **Có 3 phần quảng cáo : cấu trúc HTML khác nhau nhưng chỉ cần dùng 1 widgets : để nhận biết nó thì chỉ cần bổ sung cho nó cái input để nhận biết => component của nó cần khai báo input**
        + **Sử dụng cách viết input kiểu mới trong selector; [zvn-widget-ads]**
        + **Trong html**
        + **<div class="header-banner" zvn-widget-ads ="'top'"></div>**
        + **Tránh sinh code thừa**
        + **tương tự với phần**

**Phần quảng cáo widgets-baner: Giống hệt cái trên**

* + - * + **Muốn biết giao diện nó cần nội dung gì quay lại tập tin html của no và kiểm tra**

**Phần youtobe bên góc trái :**

**Để tránh việc sinh code thừa thì thay đổi cách viết của selector là ok [name] và thay đổi cách viết phía giao diện**

**Phần menu main : menu ở giữa thì tươn tự như trên**

* + - * **Xây dưng elementHeader => để chứa các component phía trên**

**Phần playlist- bên phải**

**Phần box like fb**

**Phần định dạng lưới tùy chọn các video thì truyền từ tập tin app.component.html**

**Khi gọi playlist ở app.component thì truyền vào cho nó title và layout => trong elmplaylist nhận 2 tk vừa truyền vào**

**Tk layout thì đổ luôn ở tập tin giao diện thì dùng interprotion**

**Và sử dụng Input để truyền vào trong phần định dạng lưới**