1. Components

- Để tạo component mới, ta sử dụng lệnh:
ng generate component <component-name>
- Vòng đời của Component Angular hỗ trợ các Interface để tạo nên sự tuần tự trong quá trình hoạt động. Một vòng đời của Component sẽ có thứ tự như sau: OnChanges, OnInit, DoCheck AfterContentInit, AfterContentChecked, AfterViewInit, AfterViewChecked, OnDestroy.
- Cách thêm lifecycle vào trong Component: Nếu muốn sử dụng method nào thì ta chỉ cần Implements nó vào trong Class
- Hiển thị dữ liệu trên View: Để hiển thị dữ liệu ra màn hình thì trong file template (tức là file html khai báo trong Component) ta sẽ sử dụng cú pháp {{ }}
- Truyền dữ liệu giữa các Component: Angular cung cấp cho chúng ta 2 decorator để truyền dữ liệu giữa các Component là @Input và @Output.
Decorator là một cú pháp khai báo đặc biệt, không bao giờ đứng độc lập mà luôn được gắn kèm với một khai báo class, method, property hoặc accessor.

COMPONENT INTERACTION - PASS DATA FROM PARENT TO CHILD WITH INPUT BINDING

@INPUT DECORATOR

Để thêm một property cho phép thiết lập progress hiện tại của thanh progress-bar, chúng ta có thể khai báo một property cho class và thêm một decorator

NGONINIT VS CONSTRUCTOR

- Constructor là hàm tạo của một class, nó là một function đặc biệt mà khi bạn khởi tạo một instance của class thì nó sẽ được tự động chạy, và chỉ chạy duy nhất một lần.
- ngOninit là một life-cycle method, nó sẽ được Angular tự động gọi khi component được khởi tạo, sau khi constructor chạy và sau khi các input đã được binding
- Angular khuyến cáo hạn chế code ở constructor, constructor làm càng ít nhiệm vụ càng tôt, hãy để ngOnInit lo tiếp phần việc còn lại.

2. Templates

- Trong angularjs có nhiều cách sử dụng template nhưng có 2 cách được dùng nhiều đó là sử dụng ng-include và ng-routes

```
<div ng-include src="template"></div>
```

hoặc được khai báo trong phần routes của module

3. Directives

- Directives 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. Directives có thể hiểu như là các đoạn mã typescript (hoặc javascript) kèm theo cả HTML và khi gọi thì gọi như là HTML luôn.

ví dụ:

```
<div *ngIf="time">
  Time: {{ time }}
</div>
```

- Phân loại:
    1. Components directives: Không có nghi ngờ gì khi gọi component là directive cũng được, vì rõ ràng là component cho phép định nghĩa selector và gọi ra như một thẻ html tag `(<component-name></component-name>)`
    2. Structural directives: Là directive cấu trúc, dùng để vẽ html, hiển thị data lên giao diện html, và thay đổi cấu trúc DOM bằng việc thêm bớt các phần tử trong DOM. Các structural directive thường có dấu '*' ở trước của directive. Ví dụ `*ngFor`, `*ngIf`
    3. Attribute directives: Thay đổi giao diện, tương tác của các đối tượng hoặc thay đổi directive khác hoặc thêm các thuộc tính động cho element html. ví dụ *`ngStyle`

STRUCTURE DIRECTIVE - NGIF

CẤU TRÚC IF-ELSE

Để hiển thị một phần view (template) theo một điều kiện, chúng ta sẽ gắn thêm một property đặc biệt vào một tag, với cú pháp có chứa dấu `* (asterisk)` như sau `*ngIf="expression".` Chỉ cần có thế là chúng ta có thể hiển thị view tùy thuộc vào dữ liệu mà expression trả về. Truthy thì hiển thị, Falsy thì không hiển thị.

NG-TEMPLATE

Với cú pháp sử dụng dấu `*` ở trên, có thể các bạn sẽ thấy nó khác lạ, nhưng thực tế, nó được gọi là Syntactic sugar (giúp nhìn code dễ hiểu, dễ đọc hơn chẳng hạn) được chuyển đổi sang dạng property binding

```
<div *ngIf="user.age >= 13">Bạn có thể xem nội dung PG-13</div>
<div *ngIf="user.age < 13">Bạn không thể xem nội dung PG-13</div>
```

hoặc

```
<ng-template [ngIf]="user.age >= 13" [ngIfElse]="noPG13">
  <div>Bạn có thể xem nội dung PG-13</div>
</ng-template>
```

STRUCTURE DIRECTIVE NGFOROF

MỘT SỐ LOCAL VARIABLE TRONG MỘT NGFOROF TEMPLATE

Terms	Description
$implicit: T	Giá trị của phần tử trong danh sách ở lần lặp hiện tại
index: number	index của lần lặp hiện tại
count: number	số lượng phần tử trong danh sách
first: boolean	True nếu đây là phần tử đầu tiên trong danh sách
last: boolean	True nếu đây là phần tử cuối cùng trong danh sách
even: boolean	True nếu đây là phần tử ở index chẵn
odd: boolean	True nếu đây là phần tử ở index lẻ

ví dụ

```
<div *ngFor="let author of authors; index as idx; count as total">
  ({{idx}})/({{total}}): {{author.id}} - {{author.firstName}}
  {{author.lastName}}
</div>
```

3. Dependency injection

- Dependency Injection là một phần quan trọng trong bộ core của Angular. Sử dụng cơ chế Dependency Injection giúp chúng ta có thể nhúng service vào các component hoặc các service với nhau.

Khởi tạo bên trong ProductComponent

Để khởi tạo instance của một class bên trong một class khác, chúng ta có thể chỉ cần `new` là được.
ví dụ

```
class ProductComponent {
  cartService: CartService = new CartService();
}
```

DI trong Angular

DI bao gồm ba thành phần sau đây:

- Injector: là một object có chứa các API để chúng ta có thể lấy về các instances đã tạo hoặc tạo các instances của các phụ thuộc.
- Provider: giống như một công thức để Injector có thể biết làm thế nào để tạo một instance của một phụ thuộc.
- Dependency: là một object (có thể là function, một value thông thường) của một kiểu dữ liệu cần phải khởi tạo.

Có thể cung cấp `injectors` với `providers` ở nhiều levels khác nhau trong app, bằng một trong ba cách sau:

- Trong `@Injectable()` decorator cho service đó.
- Trong `@NgModule()` decorator (providers array) đối với NgModule.
- Trong `@Component()` decorator (providers array) đối với component hoặc directive

4. Routing

- Thực hiện nhiệm vụ chính là chuyển trang, thay đổi một số thành phần mà không cần phải tải lại trang.
- Để sử dụng Router trong Angular ta cần :
    Tại index.html ta cần thêm <base href="/">
    Cần có một khu vực khai báo directive: <router-outlet></router-outlet> nơi này là phần các nội dung cần thay đổi
    Cần phải import RouterModule, Routes từ @angular/router
    Khai báo router root cho ứng dụng : RouterModule.forRoot(array_routes : Routes)
- Phần tử trong array_routes sẽ là một object bao gồm tối thiểu :
    path : Khai báo đường dẫn đến một component. VD: 'index', 'home' …
    Nếu khai báo path : '**' thì nếu không tìm thấy router thì sẽ load ra component tương ứng mà mình định nghĩa cho path này
    component : Khai báo component
- Lấy tham số trên router
    - Để lấy được tham số trên router ta sử dụng ActivateRoute từ @angular/router
    - Cú pháp: `.snapshot.params['tên_param_khai_báo_trong_router']`