**Angular Action**

1. Explore
2. **Data binding & Event binding**

* **Data binding: (Ràng bược dữ liệu)**
* Có 2 loại ràng buộc dữ liệu:
* One – way binding: Có 2 loại:

1. **Interpolation**: dữ liệu được chèn vào template bằng các sử dụng cặp dấu ngoặc nhọn

Ví dụ: {{ user.name }}

1. **Property binding**: dữ liệu được ràng buộc với một thuộc tính của một phần tử html thông qua dấu ngoặc vuông ‘[]’

Ví dụ: [src] = ‘imagleurl’

* **Two – way binding**: dữ liệu được chuyển đổi và đồng bộ hóa giữa template và thành phần. Điều này cho phép cập nhật dữ liệu trong cả 2 hướng. Two – way binding được thực hiện bằng dấu ngoặc trong và dấu ngoặc vuông ‘[(ngModel)]’.
* **Event binding (Ràng buộc sự kiện)**
* **Event binding**: xử lí các sự kiện từ giao diện người dùng như click, change, submit, hover và nhiều sự kiện khác. Khi sự kiện xảy ra, một hàm hoặc một phương thức trong thành phần angular được gọi để xử lí sự kiện đó.
* Cú pháp của event binding là ‘(event) = “expression”’

1. **Ngif Structural Directive**

* Ngif là một directive cấu trúc trong angular, nghĩa là nó thay đổi cấu trúc của DOM bằng cách thêm hoặc loại bỏ các phần tử DOM dựa trên một điều kiện.
* **Cú pháp**: <element \*ngIf="condition"></element>
* **Trong đó**:

+ ‘element’: phần tử html mà bạn muốn kiểm soát hiển thị dựa trên điều kiện.

+ ‘condition’: Biểu thức điều kiện. Nếu biểu thức này trả về true, phần tử sẽ được hiển thị, ngược lại sẽ bị ẩn.

1. **Ngfor Structural Directive**

* Ngfor cho phép lặp qua một mảng hoặc một đối tượng và tạo ra một phiên bản của một template cho mỗi phần tử trong mảng hoặc cho mỗi cặp key – value trong đối tượng.
* **Cú pháp**: <element \*ngFor="let item of items; let i = index; trackBy: trackByFn"></element>
* **Trong đó:**
* **‘element’:**  phần tử html mà bạn muốn lặp
* **‘items’:**  Mảng hoặc đối tượng mà bạn muốn lặp qua
* **‘item’:**  Biến được sử dụng để đại diện cho mỗi phần tử trong mảng hoặc đối tượng.
* **‘index’:** Biến index (tùy chọn) được sử dụng để lấy chỉ mục của mỗi phần tử.
* **‘trackByFn’:** Hàm (tùy chọn) được sử dụng để xác định các theo dõi các phần tử trong mảng khi thay đổi.

1. **Class & style binding**

* **Class binding:** Cho phép thêm hoặc loại bỏ các lớp CSS từ một phần tử html dựa trên điều kiện mã typescript.

Ví dụ: <div [class.myClass]="isMyClass">Some Content</div>

Ở đây `**myClass**` sẽ được thêm vào ` div ` nếu `**isMyclass**` là true và sẽ bị loại bỏ nếu `**isMyclass**` là false

* **Style binding**: Cho phép thay đổi các thuộc tính css trực tiếp từ mã TypeScript

**Ví dụ**: <div [style.color]="'red'">Some Content</div>

* **Kết hợp điều kiện**: Có thể kết hợp class và style binding với một biểu thức điều kiện để điều khiển các phần tử được hiển thị

**Ví dụ**: <div [class.myClass]="isMyClass" [style.color]="isRed ? 'red' : 'blue'">Some Content</div>

Ở đây: `.**myClass**` sẽ được thêm vào nếu `**isMyclass**` là true và màu sẽ được đặt thành đỏ `**isRed**` là true, ngược lại sẽ là màu xanh.

1. **Input binding**

* Là cơ chế cho phép truyền dữ liệu từ component cha xuống component con thông qua một thuộc tính của component con. Điều này cho phép các component con nhận dữ liệu từ component cha và tái sử dụng chúng trong quá trình hiển thị và xử lí.

1. **Output binding**

* Là cơ chế cho phép truyền dữ liệu từ component con lên component cha thông qua sự kiện. Điều này cho phép các component con thông báo cho component cha về các sự kiện xảy ra trong chúng và truyền dữ liệu đi kèm với sự kiện đó.

1. **Custom two-way Data binding**
2. **ViewChild/ ViewChildren**
3. **Components (Thành phần)**
4. **Module (Mô-đun):**
5. **Templates và Data Binding (Mẫu và Databinding)**