

* Có thể chia Angular thành 7 khối cơ bản bao gồm:
  + Module
  + Component
  + Metadata
  + Data binding
  + Directive
  + Service
  + Dependency Injection

1. Module

* Là container chứa các thành phần liên quan của ứng dụng
* Mỗi ứng dụng Angular version < 14 sẽ có một module gốc là AppModule
* Từ Angular > 14 tiếp cận Angular có thể không cần sử dụng module nữa mà sử dụng standalone

1. Component

* Là phần điều khiển một phần view của ứng dụng
* Component thông thường bao gồm ba phần: Template, Style, TS Class. Thông thường component hay được cấu trúc theo mô hình MVC
  + Template định nghĩa file bằng HTML
  + Style là css/scss/sass cho template
  + Class là file ts viết các logic để control phần template như cập nhật UI, data

1. Metadata

* Thông tin bổ sung giúp Angular xử lý class
* Được định nghĩa bằng decorator như @Component, @Injectable

1. Data binding

* Angular có 4 cơ chế dữ liệu binding từ ts class sang template bao gồm:
  + Intepolation: Binding 1 chiều từ template sang class

//component.html

<p>{{ name }}</p>;

// component.ts

@Component({

  templateUrl: "component.html",

  selector: "app-component",

})

export class Component {

  name = "Peter";

}

* + Property binding: Gán giá trị cho properties của DOM Element

// component.html

<button [disabled]="buttonDisabled"></button>

 // component.ts

 @Component({

    templateUrl: 'component.html',

    selector: 'app-component',

  })

  export class Component {

    buttonDisabled = true;

  }

* + Event binding: Xử lý các sự kiện từ user

// component.html

<p>My name is {{name}}</p>

<button (click)="updateName()">Update button</button>

// component.ts

@Component({

    templateUrl: 'component.html',

    selector: 'app-component',

  })

  export class Component {

    name = 'Peter';

    updateName() {

      this.name = 'John';

    }

  }

* + Two way binding: Kêt hợp event binding và property binding

<div>

  <input type="text"

    (keyup.enter)="updateMesssages()"

    [(ngModel)]="message"

  >

  <button (click)="updateMesssages()">Button...</button>

</div>

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.scss']

})

export class AppComponent {

  messages: string[] = [];

  message = '';

  updateMesssages() {

    this.messages.push(this.message);

    this.message = '';

  }

}

1. Directive

* Là một đối tượng giúp ta thay đổi, áp dụng hành vi lên một đối tượng khác theo một logic, điều kiện nhất định
* Có 4 loại directive chính: Component, Structural, Attribute, Custom
  + Component: Các template để định nghĩa view, chính là các component được mô tả trên
  + Structural: Lệnh điều khiển cấu trúc DOM, thường dùng để vẽ HTML, thay đổi cấu trúc DOM. Các lệnh phổ biến hay dùng gồm có: \*ngFor, \*ngIf, \*ngSwitch
  + Attribute: Thay đổi style bằng [ngStyle], [ngClass]
  + Custom: Là các directive tự custom, sử dụng @Input/@Output để tương tác với directive. Ví dụ về tự tạo custom directive có mục đích đóng context menu khi user click ở ngoài vùng context menu đó

import { Directive, ElementRef, EventEmitter, HostListener, Output } from '@angular/core';

@Directive({

  selector: '[appClickOutside]'

})

export class ClickOutsideDirective {

  @Output() clickOutside = new EventEmitter<void>();

  constructor(private el: ElementRef) {}

  @HostListener('document:click', ['$event'])

  onClick(event: MouseEvent): void {

    const targetElement = this.el.nativeElement;

    if (!targetElement.contains(event.target)) {

      this.clickOutside.emit();

    }

  }

}

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  showMenu = false;

  toggleMenu() {

    this.showMenu = !this.showMenu;

  }

  closeMenu() {

    this.showMenu = false;

  }

}

<div>

  <button (click)="toggleMenu()">Toggle Context Menu</button>

  <div \*ngIf="showMenu" appClickOutside (clickOutside)="closeMenu()">

    <div class="context-menu">

      <p>Context Menu Item 1</p>

      <p>Context Menu Item 2</p>

      <p>Context Menu Item 3</p>

    </div>

  </div>

</div>

1. Service

* Chứa logic nghiệp vụ, thường được dùng để gọi API, xử lý logic chung, chia sẻ dữ liệu giữa các component

1. Dependency Injection

* Là các pattern design giúp các class nhận dependencies của chúng