

## ChatGPT

Chào bạn, tôi đang theo lộ trình học Angular từ cơ bản tới nâng cao. Bạn đã dạy tôi các bài sau đây: Bài 1: Giới thiệu Angular và Cài đặt Môi trường Bài 2: Data Binding trong Angular Bài 3: Component Communication trong Angular Bài 4: Angular Routing (Điều hướng giữa các Component) Bài 5: Reactive Forms trong Angular Bài 6: Gọi API trong Angular với HttpClient Bài 7: Quản lý trạng thái trong Angular với NgRx (State Management) Bài 8: Angular Services và Dependency Injection Hãy tiếp tục lộ trình trên.

Chào bạn! Hôm nay, chúng ta sẽ tiếp tục với **Bài 9: Lifecycle Hooks trong Angular**. Đây là một khái niệm quan trọng giúp bạn kiểm soát vòng đời của một component trong Angular.

## Bài 9: Lifecycle Hooks trong Angular

### 1. Giới thiệu về Lifecycle Hooks

Trong Angular, các component có một vòng đời từ khi được tạo ra đến khi bị hủy. Angular cung cấp các **lifecycle hooks** (móc vòng đời) để bạn có thể thực hiện các hành động tại các giai đoạn khác nhau của component.

Dưới đây là các lifecycle hooks quan trọng:

Hook	Khi nào chạy?	Mục đích chính
------	---------------	----------------

Hook	Khi nào chạy?	Mục đích chính
<code>ngOnChanges()</code>	Khi giá trị của input property thay đổi	Phản ứng với sự thay đổi của <code>@Input()</code>
<code>ngOnInit()</code>	Khi component được khởi tạo	Khởi tạo dữ liệu, gọi API lần đầu
<code>ngDoCheck()</code>	Khi Angular kiểm tra các thay đổi trong component	Tùy chỉnh logic kiểm tra thay đổi
<code>ngAfterContentInit()</code>	Khi nội dung <code>&lt;ng-content&gt;</code> được chèn vào	Thực hiện hành động sau khi nội dung được chèn
<code>ngAfterContentChecked()</code>	Sau mỗi lần kiểm tra nội dung <code>&lt;ng-content&gt;</code>	Theo dõi nội dung có thay đổi không
<code>ngAfterViewInit()</code>	Khi view của component được khởi tạo xong	Truy cập phần tử DOM hoặc <code>@ViewChild()</code>
<code>ngAfterViewChecked()</code>	Sau mỗi lần kiểm tra view của component	Theo dõi thay đổi trong view
<code>ngOnDestroy()</code>	Trước khi component bị hủy	Dọn dẹp tài nguyên như subscription, event listener

## 2. Ví dụ về Lifecycle Hooks

Dưới đây là một component đơn giản sử dụng lifecycle hooks để kiểm tra vòng đời của component.

Tạo component `lifecycle-demo.component.ts`

typescript

```
import { Component, Input, OnChanges, OnInit, DoCheck, AfterContentInit,
        AfterContentChecked, AfterViewInit, AfterViewChecked, OnDestroy }
from '@angular/core';

@Component({
  selector: 'app-lifecycle-demo',
  template: `<p>Lifecycle Demo: {{ message }}</p>`,
})
export class LifecycleDemoComponent
  implements OnChanges, OnInit, DoCheck, AfterContentInit,
  AfterContentChecked,
    AfterViewInit, AfterViewChecked, OnDestroy {

  @Input() message: string = '';

  constructor() {
    console.log('Constructor: Component is being created');
  }

  ngOnChanges() {
    console.log('ngOnChanges: Input property changed');
  }

  ngOnInit() {
    console.log('ngOnInit: Component initialized');
  }

  ngDoCheck() {
    console.log('ngDoCheck: Change detection running');
  }

  ngAfterContentInit() {
    console.log('ngAfterContentInit: ng-content projected');
  }

  ngAfterContentChecked() {
    console.log('ngAfterContentChecked: ng-content checked');
  }

  ngAfterViewInit() {
    console.log('ngAfterViewInit: View initialized');
  }

  ngAfterViewChecked() {
    console.log('ngAfterViewChecked: View checked');
  }

  ngOnDestroy() {
```

```
        console.log('ngOnDestroy: Component is about to be destroyed');
    }
}
```

## Sử dụng component trong `app.component.html`

---

html

```
<button (click)="toggleComponent()">Toggle Component</button>
<app-lifecycle-demo *ngIf="showComponent" [message]="parentMessage"></app-
lifecycle-demo>
```

## Thêm logic trong `app.component.ts`

---

typescript

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
})
export class AppComponent {
  showComponent = true;
  parentMessage = 'Hello from parent';

  toggleComponent() {
    this.showComponent = !this.showComponent;
  }
}
```

---

## 3. Khi nào sử dụng Lifecycle Hooks?

---

- `ngOnInit()`: Dùng để gọi API hoặc khởi tạo giá trị khi component được tạo.
- `ngOnChanges()`: Dùng để phản ứng khi giá trị của `@Input()` thay đổi.
- `ngDoCheck()`: Dùng để thực hiện các thay đổi thủ công không được phát hiện tự động.
- `ngAfterViewInit()`: Dùng để truy cập vào DOM sau khi component được hiển thị.
- `ngOnDestroy()`: Dùng để hủy subscription hoặc dọn dẹp bộ nhớ.

---

## 4. Kết luận

---

Lifecycle Hooks giúp bạn kiểm soát quá trình hoạt động của component trong Angular. Biết cách sử dụng các hooks này sẽ giúp bạn tối ưu hóa hiệu suất và quản lý tài nguyên trong ứng dụng.

---

Bạn hãy thử chạy code trên và quan sát các log trong console để hiểu cách các hooks này hoạt động nhé! Sau khi bạn hoàn thành bài này, tôi sẽ hướng dẫn tiếp **Bài 10: Angular Directives (Directive trong Angular)**. 🚀

NG0303: Can't bind to 'ngIf' since it isn't a known property of 'app-lifecycle-demo' (used in the '\_AppComponent' component template).

Lỗi **NG0303: Can't bind to 'ngIf' since it isn't a known property** thường xảy ra khi bạn sử dụng `*ngIf` nhưng quên import **CommonModule** vào `@NgModule` trong Angular. Dưới đây là cách khắc phục:

### Cách 1: Import `CommonModule` trong `app.module.ts`

---

Mặc định, các directive như `*ngIf`, `*ngFor`, ... thuộc `CommonModule`, nhưng `CommonModule` không tự động được import trong mọi module. Nếu bạn đang sử dụng `*ngIf` trong `AppComponent`, hãy đảm bảo `CommonModule` đã được import trong `app.module.ts`.

**Mở `app.module.ts` và kiểm tra:**

---

typescript

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { LifecycleDemoComponent } from './lifecycle-demo.component';
import { CommonModule } from '@angular/common'; // Thêm dòng này

@NgModule({
  declarations: [
    AppComponent,
    LifecycleDemoComponent
  ],
  imports: [
    BrowserModule,
    CommonModule // Đảm bảo import module này
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

---

## Cách 2: Nếu **LifecycleDemoComponent** thuộc một module khác

---

Nếu bạn đã tạo **LifecycleDemoComponent** trong một module khác (**LifecycleModule** chẳng hạn), bạn cần làm như sau:

1. Mở **lifecycle.module.ts** và import **CommonModule**

typescript

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { LifecycleDemoComponent } from './lifecycle-demo.component';

@NgModule({
  declarations: [LifecycleDemoComponent],
  imports: [CommonModule], // Đảm bảo đã import CommonModule
  exports: [LifecycleDemoComponent] // Xuất component để module khác có thể
    sử dụng
})
export class LifecycleModule { }
```

2. Import **LifecycleModule** trong **app.module.ts** Nếu **LifecycleDemoComponent** thuộc module riêng, bạn cần import module đó vào **app.module.ts**:

typescript

```
import { LifecycleModule } from './lifecycle/lifecycle.module';

@NgModule({
  imports: [
    BrowserModule,
    LifecycleModule // Đảm bảo module chứa LifecycleDemoComponent được
import
  ]
})
export class AppModule { }
```

---

### Cách 3: Nếu đang dùng **Standalone Components** (Angular 15+)

---

Nếu bạn dùng **Standalone Components** (Angular 15 trở lên), bạn không cần module, nhưng phải import **CommonModule** trực tiếp trong component.

typescript

```
import { Component, Input } from '@angular/core';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-lifecycle-demo',
  standalone: true, // Định nghĩa component này là Standalone
  imports: [CommonModule], // Import CommonModule
  template: `<p>Lifecycle Demo: {{ message }}</p>`,
})
export class LifecycleDemoComponent {
  @Input() message: string = '';
}
```

---

Sau khi thực hiện một trong các cách trên, lỗi **NG0303: Can't bind to 'ngIf' since it isn't a known property** sẽ được khắc phục. Hãy thử lại và cho tôi biết nếu bạn còn gặp lỗi nào khác nhé! 🚀

tôi đang dùng standalone component

Nếu bạn đang sử dụng **Standalone Component** trong Angular, bạn cần đảm bảo **CommonModule** được **import trực tiếp** vào component đó, vì các directive như **\*ngIf** thuộc về **CommonModule**.

---

### 🔧 Cách khắc phục lỗi NG0303 trong Standalone Component

---

Bạn chỉ cần **import CommonModule** vào **AppComponent** hoặc bất kỳ component nào sử dụng **\*ngIf**.

**1** Cập nhật **app.component.ts**

---



typescript

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { LifecycleDemoComponent } from './lifecycle-demo.component';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [CommonModule, LifecycleDemoComponent], // Import CommonModule
  template: `
    <button (click)="toggleComponent()">Toggle Component</button>
    <app-lifecycle-demo *ngIf="showComponent" [message]="parentMessage">
  </app-lifecycle-demo>
  `,
})
export class AppComponent {
  showComponent = true;
  parentMessage = 'Hello from parent';

  toggleComponent() {
    this.showComponent = !this.showComponent;
  }
}
```

## Kiểm tra lại `lifecycle-demo.component.ts`

---

typescript

```
import { Component, Input } from '@angular/core';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-lifecycle-demo',
  standalone: true,
  imports: [CommonModule], // Đảm bảo đã import CommonModule
  template: `<p>Lifecycle Demo: {{ message }}</p>`,
})
export class LifecycleDemoComponent {
  @Input() message: string = '';
}
```

---

### ♦ Giải thích

---

- **Standalone Components** không thuộc bất kỳ module nào (**NgModule**), vì vậy chúng không tự động có quyền truy cập vào **CommonModule**.
  - **CommonModule** cần được import vào **mọi component** nếu chúng sử dụng directive như **\*ngIf**, **\*ngFor**, v.v.
  - Trong **AppComponent**, bạn cũng phải **import LifecycleDemoComponent** để có thể sử dụng nó.
- 

Sau khi thực hiện các bước trên, lỗi **NG0303: Can't bind to 'ngIf'** sẽ được khắc phục. Bạn thử lại và cho tôi biết nếu vẫn gặp lỗi nhé! 🚀

ChatGPT can make mistakes. Check important info.