Here's a **complete working Angular app** that demonstrates the **Component Lifecycle**, **template access**, and **change tracking** with lifecycle hooks.

**🧪 App Overview**

* AppComponent: The root component that toggles a child.
* LifecycleDemoComponent: Implements all lifecycle hooks, tracks changes, accesses template using @ViewChild.

**📁 File Structure**

src/

app/

app.component.ts

app.component.html

lifecycle-demo/

lifecycle-demo.component.ts

lifecycle-demo.component.html

lifecycle-demo.component.css

**🧩 1. app.component.ts**

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

@Component({

selector: 'app-root',

templateUrl: './app.component.html'

})

export class AppComponent {

showChild = true;

name = 'Angular';

toggleChild() {

this.showChild = !this.showChild;

}

changeName() {

this.name = 'Angular ' + Math.floor(Math.random() \* 100);

}

}

**🧩 2. app.component.html**

<h2>Angular Lifecycle Demo</h2>

<button (click)="toggleChild()">Toggle Child</button>

<button (click)="changeName()">Change Name</button>

<div \*ngIf="showChild">

<app-lifecycle-demo [name]="name"></app-lifecycle-demo>

</div>

**🧩 3. lifecycle-demo.component.ts**

import {

Component, Input, OnInit, OnChanges, DoCheck,

AfterContentInit, AfterContentChecked,

AfterViewInit, AfterViewChecked, OnDestroy,

SimpleChanges, ViewChild, ElementRef

} from '@angular/core';

@Component({

selector: 'app-lifecycle-demo',

templateUrl: './lifecycle-demo.component.html',

styleUrls: ['./lifecycle-demo.component.css']

})

export class LifecycleDemoComponent implements

OnChanges, OnInit, DoCheck,

AfterContentInit, AfterContentChecked,

AfterViewInit, AfterViewChecked,

OnDestroy {

@Input() name: string = '';

@ViewChild('templateElement') para!: ElementRef;

constructor() {

console.log('constructor');

}

ngOnChanges(changes: SimpleChanges) {

console.log('ngOnChanges', changes);

}

ngOnInit() {

console.log('ngOnInit');

}

ngDoCheck() {

console.log('ngDoCheck');

}

ngAfterContentInit() {

console.log('ngAfterContentInit');

}

ngAfterContentChecked() {

console.log('ngAfterContentChecked');

}

ngAfterViewInit() {

console.log('ngAfterViewInit');

console.log('Template Text:', this.para.nativeElement.textContent);

}

ngAfterViewChecked() {

console.log('ngAfterViewChecked');

}

ngOnDestroy() {

console.log('ngOnDestroy');

}

}

**🧩 4. lifecycle-demo.component.html**

<div class="child-box">

<p #templateElement>Hello, {{ name }}!</p>

</div>

**🎨 5. lifecycle-demo.component.css**

.child-box {

padding: 10px;

margin: 10px;

border: 2px solid green;

}

**▶️ How to Run**

1. Create a new Angular project using CLI:

ng new lifecycle-demo-app

cd lifecycle-demo-app

1. Replace src/app contents with the files above.
2. Declare the component in AppModule:

// app.module.ts

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

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

import { LifecycleDemoComponent } from './lifecycle-demo/lifecycle-demo.component';

@NgModule({

declarations: [

AppComponent,

LifecycleDemoComponent

],

imports: [BrowserModule],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

1. Run the app:

ng serve

**💡 Experiment Ideas**

* Toggle the child to see ngOnDestroy in action.
* Use ChangeDetectorRef to manually detect changes.
* Add a timer in ngOnInit and clear it in ngOnDestroy.

Would you like me to **send this as a zip file**, or **upload to StackBlitz** so you can run it online instantly?

**You said:**

@ViewChild() • Getting Access to the Template & DOM with @ViewChild

• Projecting Content into Components with ng-content

• @ContentChild() decorator • Getting Access to ng-content with @ContentChild

**ChatGPT said:**

Great! You're diving into **advanced Angular component interaction** and DOM access. Let’s walk through:

**🧩 @ViewChild()**

**➤ Access elements or child components inside the component’s own template.**

**✅ Syntax:**

@ViewChild('templateRef') elementRef!: ElementRef;

**✅ Example:**

<!-- lifecycle-demo.component.html -->

<p #para>Hello from ViewChild!</p>

@ViewChild('para') para!: ElementRef;

ngAfterViewInit() {

console.log('ViewChild Text:', this.para.nativeElement.textContent);

}

Use ngAfterViewInit() to access it safely.

**🎯 Use Cases for @ViewChild()**

* Access DOM elements directly.
* Call methods on child components.
* Focus input elements, scroll to view, etc.

**🎁 Projecting Content with ng-content**

<ng-content> allows you to project HTML **into a component from the outside**.

**✅ Example:**

// card.component.ts

@Component({

selector: 'app-card',

template: `

<div class="card">

<ng-content></ng-content>

</div>

`

})

export class CardComponent {}

<!-- parent.component.html -->

<app-card>

<p>This is projected into the card!</p>

</app-card>

**🔍 @ContentChild()**

While @ViewChild accesses template elements, @ContentChild() accesses **projected content** via ng-content.

**✅ Syntax:**

@ContentChild('projectedRef') projected!: ElementRef;

**✅ Example:**

<!-- parent.component.html -->

<app-card>

<p #innerText>Projected paragraph</p>

</app-card>

// card.component.ts

@ContentChild('innerText') inner!: ElementRef;

ngAfterContentInit() {

console.log('ContentChild Text:', this.inner.nativeElement.textContent);

}

Use ngAfterContentInit() for projected content.

**🧪 Full Working Example**

**🔧 1. card.component.ts**

import { Component, ContentChild, AfterContentInit, ElementRef } from '@angular/core';

@Component({

selector: 'app-card',

template: `

<div class="card">

<ng-content></ng-content>

</div>

`,

styles: [`.card { padding: 10px; border: 1px solid #ccc; }`]

})

export class CardComponent implements AfterContentInit {

@ContentChild('innerText') projectedContent!: ElementRef;

ngAfterContentInit() {

console.log('Projected content:', this.projectedContent.nativeElement.textContent);

}

}

**📄 2. app.component.html**

<h2>Content Projection Demo</h2>

<app-card>

<p #innerText>This text is projected</p>

</app-card>

**✅ Summary**

| **Decorator** | **Accesses...** | **When Available** |
| --- | --- | --- |
| @ViewChild() | Component's own template | ngAfterViewInit() |
| @ContentChild() | Projected ng-content | ngAfterContentInit() |