**Angular Lifecycle Hooks**

In Angular, every component goes through a series of stages from **creation to destruction.**

These stages are called the **component lifecycle**, and Angular provides special methods called **lifecycle hooks** to let you run code at each stage.

Lifecycle hooks are very useful for:

* Initializing data
* Watching for input changes
* Interacting with the DOM
* Cleaning up before a component is destroyed

**ORDER OF EXECUTION**

 constructor

 ngOnChanges (if inputs are present)

 ngOnInit

 ngDoCheck

 ngAfterContentInit

 ngAfterContentChecked

 ngAfterViewInit

 ngAfterViewChecked

 ngOnDestroy

### 1. **constructor()**

This is a standard TypeScript method — called when the component class is created in memory. It’s used to **inject services** or do very light setup.

❌ Don't write API calls or complex logic here — component is not ready yet.

### 2. **ngOnInit()**

This hook is called **once**, after the component has been initialized and all the input properties are set.

✅ Use this to:

* Fetch data from APIs
* Run initialization logic
* Set default values

💡 One of the **most commonly used hooks** in Angular.

### 3. **ngOnChanges()**

This hook runs **whenever the value of an @Input() property changes**. It runs before ngOnInit() and also any time the parent updates the data.

✅ Use this to:

* React to changes in input values coming from the parent component

### 4. **ngOnDestroy()**

This hook is called **just before the component is removed from the screen**.

✅ Use this to:

* Unsubscribe from observables
* Clear timers or intervals
* Clean up resources to prevent memory leaks

This is especially useful when working with subscriptions, WebSockets, or setInterval.

### 5. **ngDoCheck()**

This is called during every Angular change detection cycle, even if no data has changed.

✅ Use this when:

* You want to write **custom change detection logic**
* Angular’s default detection doesn’t catch something

Usually not needed unless working on advanced performance tweaks.

### 6. **ngAfterContentInit()**

Called once after Angular **inserts external content** into the component using <ng-content>.

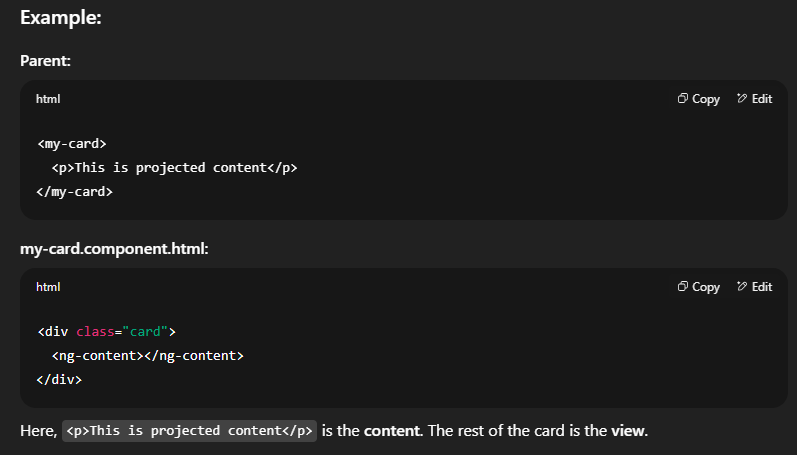
✅ Use this if:

* Your component uses **content projection** and you need to access or react to the projected content.

🧠 **What is Content?**  
Content is **HTML that comes from a parent component** and is placed into this component using <ng-content>.

**What is content?**

External HTML that a parent sends into this component using <ng-content>.

****

### 7. **ngAfterContentChecked()**

Called every time Angular **checks the projected content** for **changes.**

✅ Useful when:

* You need to perform actions after every update of the projected content.

Not commonly used in beginner-level apps.

### 8. **ngAfterViewInit()**

Called once after Angular initializes the component's **own view** (i.e., its template and child components).

✅ Use this to:

* Access DOM elements using @ViewChild
* Do something only after the full view is rendered

🧠 **What is View?**  
The component’s own HTML structure and anything declared inside its template.

### 9. **ngAfterViewChecked()**

Called after the view and child views have been checked by Angular’s change detection.

✅ Useful when:

* You want to react after the view has been updated (usually for fine-tuning or debugging)

Rarely used in beginner-level projects.

**THANK YOU**