**Component Life Cycle Hooks in Angular**

The life cycle hooks are the methods that angular invokes on the [directives](https://www.tektutorialshub.com/angular/angular-directives/) and [components](https://www.tektutorialshub.com/angular/angular-component/) as it creates, changes, and destroys them.

## What is Angular Component lifecycle hooks

When the angular application starts it creates and renders the root component. It then creates and renders its Childrens & their children. It forms a [tree of components](https://www.tektutorialshub.com/angular/angular-architecture-overview-concepts/#a-typical-angular-application).

Once Angular loads the components, it starts the process of rendering the view.  To do that it needs to check the input properties, evaluate the data bindings & expressions, render the projected content etc. Angular also removes the component from the DOM, when it is no longer needs it.

Angular lets us know when these events happen using lifecycle hooks

The Angular life cycle hooks are nothing but callback function, which angular invokes when a certain event occurs during the component’s life cycle.

## Angular lifecycle hooks

angular invokes during the component life cycle. Angular invokes them when a certain event occurs.

* ngOnChanges
* ngOnInit
* ngDoCheck
* ngAfterContentInit
* ngAfterContentChecked
* ngAfterViewInit
* ngAfterViewChecked
* ngOnDestroy

Note that the constructor event is fired before the OnInit hook.

## The Order of Execution of Life Cycle Hooks

The Angular executes the hooks in the following order

On Component Creation

1. OnChanges
2. OnInit
3. DoCheck
4. AfterContentInit
5. AfterContentChecked
6. AfterViewInit
7. AfterViewChecked

When the Component with Child Component is created

1. OnChanges
2. OnInit
3. DoCheck
4. AfterContentInit
5. AfterContentChecked
   1. Child Component -> OnChanges
   2. Child Component -> OnInit
   3. Child Component -> DoCheck
   4. Child Component -> AfterContentInit
   5. Child Component -> AfterContentChecked
   6. Child Component -> AfterViewInit
   7. Child Component -> AfterViewChecked
6. AfterViewInit
7. AfterViewChecked

After The Component is Created

1. OnChanges
2. DoCheck
3. AfterContentChecked
4. AfterViewChecked

The OnChanges hook is fired only if there is an input property defined in the component and it changes. Otherwise, it will never fire

## How to Use Lifecycle Hooks

1. Import Hook interfaces
2. Declare that Component/directive Implements lifecycle hook interface
3. Create the hook method

### Import Hook interfaces

Import hook interfaces from the core module. The name of the Interface is hook name without ng.

import { Component,OnInit } from '@angular/core'

### Component Implements lifecycle hook interface

## export class AppComponent implements OnInit {

The life cycle hook methods must use the same name as the hook.

|  |  |
| --- | --- |
| 1  2  3  4  5 | ngOnInit() {      console.log("AppComponent:OnInit");    } |

## Change detection Cycle

Before diving into the lifecycle hooks, we need to understand the change detection cycle.

Change detection is the mechanism by which angular keeps the template in sync with the component

How does angular know when the value of name changes?. It does so by running a change detection cycle on every event that may result in a change. It runs it on every input changes, DOM events, timer events like setTimeout() and setInterval() , http requests etc.

During the change detection cycle angular checks each and every bound property in the template, with that of the component class. If it detects any changes it updates the DOM.

Angular raises the life cycle hooks during the important stages of the change detection  mechanism.

## Constructor

Life Cycle of a component begins, when Angular creates the component class. First method that gets invoked is class Constructor.

Constructor is neither a life cycle hook nor it is specific to Angular.  It is a Javascript feature. It is a method which is invoked, when a class is created.

Angular makes use of a constructor to [inject dependencies](https://www.tektutorialshub.com/angular/angular-dependency-injection/).

At this point, none of the components input properties are available to use. Neither its child components are constructed. Projected contents are also not available.

Once Angular instantiates the class, It kick-start the first change detection cycle of the component.

## ngOnChanges

The Angular invokes [ngOnChanges](https://www.tektutorialshub.com/angular/angular-ngonchanges-life-cycle-hook/) life cycle hook whenever any data-bound input property(@Input -> one of the ways by which a parent communicates with the child component) of the component or directive changes. Initializing the Input properties is the first task that angular carries during the change detection cycle. And if it detects any change in property, then it raises the [ngOnChanges](https://www.tektutorialshub.com/angular/angular-ngonchanges-life-cycle-hook/) hook. It does so during every change detection cycle. This hook is not raised if change detection does not detect any changes.

The change detector checks if such input properties of a component are changed by the parent component. If it is then it raises the [ngOnChanges](https://www.tektutorialshub.com/angular/angular-ngonchanges-life-cycle-hook/) hook.

The change detector uses the === [strict equality operator](https://www.tektutorialshub.com/typescript/strict-equality-loose-equality-in-typescript/) for detecting changes. Hence for objects, the hook is fired only if the references are changed

## ngOnInit

The Angular raises the [ngOnInit](https://www.tektutorialshub.com/angular/angular-ngoninit-and-ngondestroy/) hook, after it creates the component and updates its input properties. It raises it after the [ngOnChanges](https://www.tektutorialshub.com/angular/angular-ngonchanges-life-cycle-hook/) hook.

This hook is fired **only once** and immediately after its creation (during the first change detection).

This is a perfect place where you want to add any initialisation logic for your component.  Here you have access to every input property of the component. You can use them in  http get requests to get the data from the back end server or run some initialization logic etc.

But note that none of child components or projected content are available at this point. Hence any properties we decorate with [@ViewChild](https://www.tektutorialshub.com/angular/understanding-viewchild-viewchildren-querylist-in-angular/), [@ViewChildren](https://www.tektutorialshub.com/angular/understanding-viewchild-viewchildren-querylist-in-angular/) , [@ContentChild](https://www.tektutorialshub.com/angular/contentchild-and-contentchildren-in-angular/) & [@ContentChildren](https://www.tektutorialshub.com/angular/contentchild-and-contentchildren-in-angular/) will not be available to use.

## ngDoCheck

The Angular invokes the [ngDoCheck](https://www.tektutorialshub.com/angular/angular-ngdocheck-life-cycle-hook/) hook event during every change detection cycle. This hook is invoked even if there is no change in any of the properties.

Angular invokes it after the [ngOnChanges](https://www.tektutorialshub.com/angular/angular-ngonchanges-life-cycle-hook/) & [ngOnInit](https://www.tektutorialshub.com/angular/angular-ngoninit-and-ngondestroy/) hooks.

Use this hook to Implement a custom change detection, whenever Angular fails to detect the changes made to Input properties. This hook is particularly useful when you opt for the Onpush change detection strategy.

The Angular [ngOnChanges](https://www.tektutorialshub.com/angular/angular-ngonchanges-life-cycle-hook/) hook [does not detect all the changes made to the input properties](https://www.tektutorialshub.com/angular/angular-ngonchanges-life-cycle-hook/#onchanges-does-not-fire-always).

## ngAfterContentInit

ngAfterContentInit Life cycle hook is called after the Component’s [projected content](https://www.tektutorialshub.com/angular/ng-content-content-projection-in-angular/) has been fully initialized. Angular also updates the properties decorated with the [ContentChild and ContentChildren](https://www.tektutorialshub.com/angular/contentchild-and-contentchildren-in-angular/) before raising this hook. This hook is also raised, even if there is no content to project.

The content here refers to the external content injected from the parent component via [Content Projection](https://www.tektutorialshub.com/angular/ng-content-content-projection-in-angular/).

The [Angular Component](https://www.tektutorialshub.com/angular/angular-component/)s can include the [ng-content](https://www.tektutorialshub.com/angular/ng-content-content-projection-in-angular/) element, which acts as a placeholder for the content from the parent as shown below

|  |  |
| --- | --- |
| 1  2  3  4  5 | <h2>Child Component</h2>  <ng-content></ng-content>   <!-- placehodler for content from parent --> |

Parent injects the content between the opening & closing element.  Angular passes this content to the child component

|  |  |
| --- | --- |
| 1  2  3  4  5 | <h1>Parent Component</h1>  <app-child> This <b>content</b> is injected from parent</app-child> |

During the change detection cycle, Angular checks if the injected content has changed and updates the DOM.

This is a component only hook.

## ngAfterContentChecked

ngAfterContentChecked Life cycle hook is called during every change detection cycle after Angular finishes checking of component’s projected content. Angular also updates the properties decorated with the [ContentChild and ContentChildren](https://www.tektutorialshub.com/angular/contentchild-and-contentchildren-in-angular/) before raising this hook. Angular calls this hook even if there is no projected content in the component

This hook is very similar to the ngAfterContentInit hook. Both are called after the external content is initialized, checked & updated. Only difference is that ngAfterContentChecked is raised after every change detection cycle. While ngAfterContentInit during the first change detection cycle.

This is a component only hook.

## ngAfterViewInit

ngAfterViewInit hook is called after the Component’s View & all its child views and directives are fully initialized. Angular also updates the properties decorated with the [ViewChild](https://www.tektutorialshub.com/angular/understanding-viewchild-viewchildren-querylist-in-angular/) & [ViewChildren](https://www.tektutorialshub.com/angular/understanding-viewchild-viewchildren-querylist-in-angular/) properties before raising this hook.

The View here refers to the template of the current component and all its child components & directives.

This hook is called during the first change detection cycle, where angular initializes the view for the first time

At this point all the lifecycle hook methods & change detection  of all child components & directives are processed & Component is completely ready

This is a component only hook.

## ngAfterViewChecked

The Angular fires this hook after it checks & updates the component’s views and child views. This event is fired after the ngAfterViewInit and after that during every change detection cycle

This hook is very similar to the ngAfterViewInit hook. Both are called after all the child components & directives are initialized and updated. Only difference is that ngAfterViewChecked is raised during every change detection cycle. While ngAfterViewInit during the first change detection cycle.

This is a component only hook.

## ngOnDestroy

The ngOnDestroy or OnDestroy hook is called just before the Component/Directive instance is destroyed by Angular

Use this hook to Perform any cleanup logic for the Component. This is the correct place where you would like to Unsubscribe Observables and detach event handlers to avoid memory leaks.