**Life Cycle Hook**

The life cycle hooks are the methods that angular invokes on directives and components as it creates, changes, and destroys them.

Using life-cycle hooks we can fine-tune the behavior of our components during creation, update, and destruction.

## Angular lifecycle hooks

Here is the complete list of life cycle hooks provided by the Angular

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

Each hook defined above is invoked by Angular when certain events occur.

## Life Cycle of a Component with Hooks

Angular components have their own life cycle events that are mainly maintained by Angular itself. Below is the list of life cycle events of any Angular components. In Angular, every component has a life-cycle, a number of different stages it goes through from initialization to destruction. There are eight different stages in the component lifecycle. Every stage is called a life cycle hook event. So, we can use these component lifecycle events in different stages of our application to obtains complete controls on the components.

* **ngOnChanges** – This event executes every time a value of an input control within the component has been changed. This event activates first when the value of a bound property has been changed.
* **ngOnInit** – This event executed at the time of Component initialization. This event is called only once, just after the ngOnChanges() events. This event is mainly used to initialize data in a component.
* **ngDoCheck** – This event is executed every time when the input properties of a component are checked. We can use this life cycle method to implement the checking on the input values as per our own logic check.
* **ngAfterContentInit** – This lifecycle method is executed when Angular performs any content projection within the component views. This method executes only once when all the bindings of the component need to be checked for the first time. This event executes just after the ngDoCheck() method.
* **ngAfterContentChecked** - This life cycle hook method executes every time the content of the component has been checked by the change detection mechanism of Angular. This method is called after the ngAfterContentInit() method. This method is can also be executed on every execution of ngDoCheck() event.
* **ngAfterViewInit** – This life cycle method executes when the component completes the rendering of its view full. This life cycle method is used to initialize the component’s view and child views. It is called only once, after ngAfterContentChecked(). This lifecycle hook method only applies to components.
* **ngAfterViewChecked** – – This method is always executed after the ngAterViewInit() method. Basically, this life cycle method is executed when the change detection algorithm of the angular component occurs. This method automatically executed every execution time of the ngAfterContentChecked().
* **ngOnDestroy** – This method will be executed when we want to destroy the Angular components. This method is very useful for unsubscribing the observables and detaching the event handlers to avoid memory leaks. It is called just before the instance of the component being destroyed. This method is called only once, just before the component is removed from the DOM.

### Component Life Cycle Demo

Now in this demo, we will demonstrate the life cycle events of a component. For that, add the below code in the app.component.ts file –

1. **import** { Component } from '@angular/core';
3. @Component({
4. selector: 'app-root',
5. templateUrl: './app.component.html',
6. styleUrls : ['./custom.css']
7. })
8. **export** **class** AppComponent {
9. data:number=100;
10. constructor() {
11. console.log(`**new** - data is ${**this**.data}`);
12. }

    ngOnChanges() {

        console.log(`ngOnChanges - data is ${**this**.data}`);

    }

    ngOnInit() {

        console.log(`ngOnInit  - data is ${**this**.data}`);

    }

1. }

Now add the below code in app.component.html file –

1. <span **class**="setup">Given Number</span>
2. <h1 **class**="punchline">{{ data }}</h1>

Now refresh the browser to check the output –