|  |  |
| --- | --- |
| ngOnChanges() | This event called every time when a value of an input control within the component has been changed. The method receives a [SimpleChanges](https://angular.io/api/core/SimpleChanges) object of current and previous property values.  Called before ngOnInit() and whenever one or more data-bound input properties change. |
| ngOnInit() | Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties.  Called *once*, after the *first*ngOnChanges(). |
| ngDoCheck() | Detect changes when Angular can't detect on its own.  Called during every change detection run, immediately after ngOnChanges()and ngOnInit(). |
| [ngAfterContentInit()](https://angular.io/api/router/RouterLinkActive#ngAfterContentInit) | Called after Angular projects external content into the component's view / the view that a directive is in.  Called *once* after the first ngDoCheck(). |
| ngAfterContentChecked() | Called after Angular checks the content projected into the directive/component.  Called after the [ngAfterContentInit()](https://angular.io/api/router/RouterLinkActive#ngAfterContentInit) and every subsequent ngDoCheck(). |
| [ngAfterViewInit()](https://angular.io/api/forms/NgForm#ngAfterViewInit) | Called after Angular initializes the component's views and child views / the view that a directive is in.  Called *once* after the first ngAfterContentChecked(). |
| ngAfterViewChecked() | Called after Angular checks the component's views and child views / the view that a directive is in.  Called after the [ngAfterViewInit()](https://angular.io/api/forms/NgForm#ngAfterViewInit)and every subsequent ngAfterContentChecked(). |
| ngOnDestroy() | Cleanup just before Angular destroys the directive/component. Unsubscribe Observables and detach event handlers to avoid memory leaks.  Called *just before* Angular destroys the directive/component. |

<https://dzone.com/articles/angular-6-part-3-life-cycle-of-a-component>

**ngOnChanges**– This event executes every time when a value of an input control within the component has been changed.

It always receives a change data map, containing the current and previous value of the bound property wrapped in a SimpleChange.

Actually, this event is fired first when a value of a bound property has been changed.

**ngOnInit** – This event called after the component has been initialized. This event is mainly used for the initialize data in a component.

This event is basically called only after the ngOnChanges()events

**ngDoCheck**– This event is triggered every time the input properties of a component are checked.

Basically, this method allows us to implement our own custom change detection logic or algorithm for any component.

**ngAfterContentInit-** This method executes when all the bindings of the component need to be checked for the first time.

This event executes just after the ngDoCheck() method. This method is basically linked with the child component initializations.

**ngAfterContentChecked** – This lifecycle 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 also called on every subsequent execution of ngDoCheck().

This method is also mainly linked with the child component initializations.

**ngAfterViewInit** – This lifecycle hook method executes when the component’s view has been fully initialized and child views.

It is called after ngAfterContentChecked(). This lifecycle hook method only applies to components.

**ngAfterViewChecked**–It is executed every time the view of the given component has been checked by the change detection algorithm of Angular.

This method executes after every subsequent execution of the ngAfterContentChecked().

This method also executes when any binding of the children directives has been changed.

So this method is very useful when the component waits for some value which is coming from its child components.

This method is called after the ngAterViewInit() method

**ngOnDestroy** – This method will be executed just before Angular destroys the components.

This method is very useful for unsubscribing from the observables and detaching the event handlers to avoid memory leaks.

Actually, it is called just before the instance of the component is finally destroyed.

This method is called just before the component is removed from the DOM.