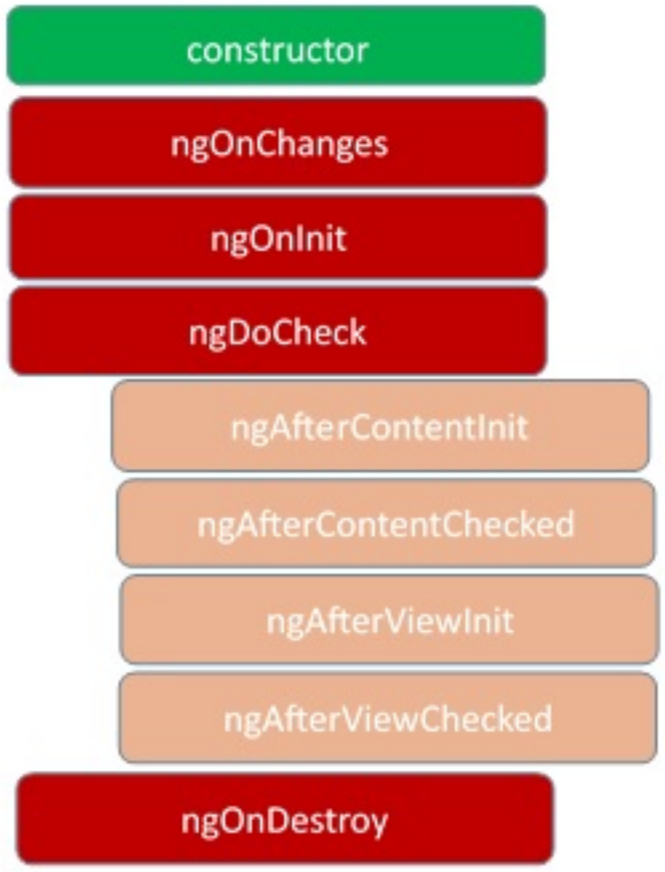
Lifecycle Method

In Angular, every component has a life-cycle, a number of different stages it goes through. There are 8 different stages in the component lifecycle. Every stage is called as lifecycle hook event. So, we can use these hook events in different phases of our application to obtain control of the components. Since a component is a TypeScript class, every component must have a constructor method. The constructor of the component class executes, first, before the execution of any other lifecycle hook events. If we need to inject any dependencies into the component, then the constructor is the best place to inject those dependencies. After executing the constructor, Angular executes its lifecycle hook methods in a specific order.



These stages are mainly divided into two phases – one is linked to the component itself and another is linked to the children of that component.

- **ngOnChanges** – This event executes every time when a value of an input control within the component has been changed. Actually, this event is fired first when a value of a bound property has been changed. It always receives a change data map, containing the current and previous value of the bound property wrapped in a `SimpleChange`.
- **ngOnInit** – This event initializes after Angular first displays the data-bound properties or when the component has been initialized. This event is basically called only after the `ngOnChanges()`events. This event is mainly used for the initialize data in a component.
- **ngDoCheck** – This event is triggered every time the input properties of a component are checked. We can use this hook method to implement the check with our own logic check. Basically, this method allows us to implement our own custom change detection logic or algorithm for any component.
- **ngAfterContentInit** – This lifecycle method is executed when Angular performs any content projection within the component views. 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. This method is

initialized after Angular initializes the component's view and child views. It is called after `ngAfterContentChecked()`. This lifecycle hook method only applies to components.

- **ngAfterViewChecked** – This method is called after the `ngAterViewInit()` method. 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.
- **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.