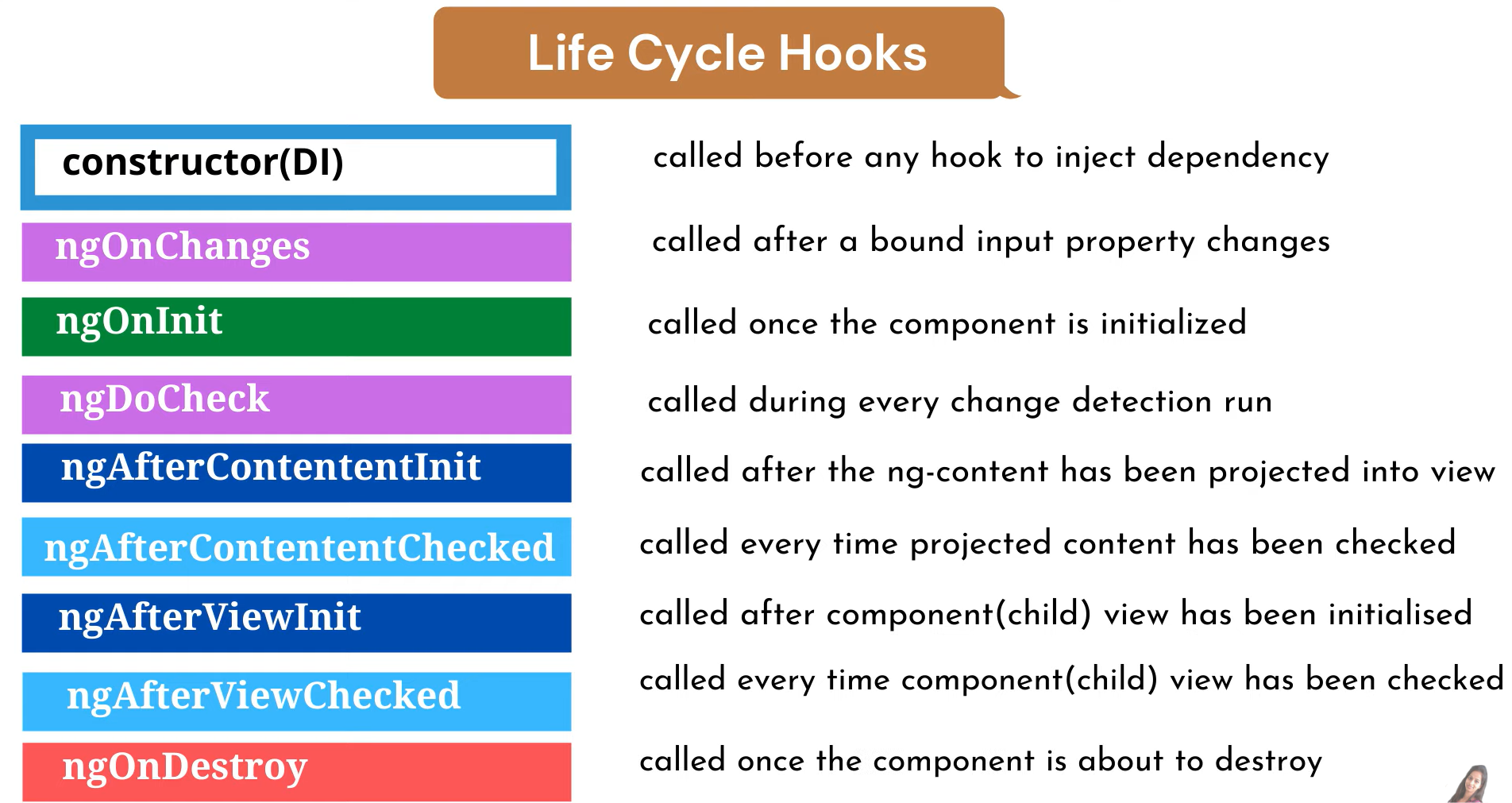
16

**Component Lifecycle/Hooks**

* Constructor
* ngOnchange
* ngOnInit
* ngDoCheck
  + ngAfterContentInit
  + ngAfterContentChecked
  + ngAfterViewInit
  + ngAfterViewChecked
* ngOnDestroy

****

**What are Component Hooks/Lifecycle?**

* Every component in angular has a life-cycle , a series of stages that goes through from initialization to destruction
* Life-cycle hooks events occur at each stage. As a result, we can use these hook events various stages of our application to achieve fine control over the components.
* Hooks can implement by using interfaces

what is constructor?

* it is special method which will invoked automatically whenever object is created
* it is typescript feature not angular
* it is used to dependency injection (di)
* it executes in first of hooks after that start hooks

example;



What is ngOnInit?

* Called on initialization
* OnInit is a lifecycle hook that is called after Angular has initialized all data-bound properties of a directive.
* ngOnInit() method to handle any additional initialization tasks.
* This hook is called when the first change detection is run on the component.

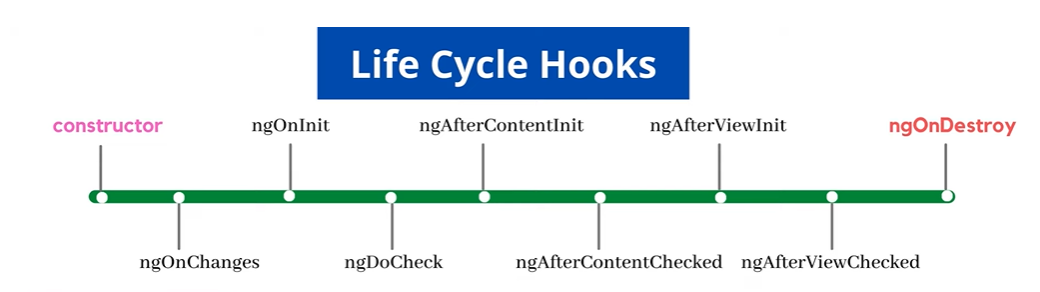
Example

Graphical user interface

Description automatically generated with medium confidence

Difference between constructor and ngOnInit

|  |  |
| --- | --- |
| constructor | ngOnInIt |
| * Should use only for dependency injection * It is part of typescript * Binding not happened till we can access only variables | * Should use for handle any initial logic that need to be executed * It is part of angular framework * Binding with the UI is done * So can access all data bound properties |



**ngOnChanges**

* this method is called once on components creation and then every time changes are detected in one of the component input properties
* It Receives a Simplechanges Object as a parameter which contains information regarding the which of the input properties has changed- in case we have more than one – current and previous value

**ngDoCheck**

* During the change detection , when angular checks components input properties for change, it uses === for dirty checking.
* For arrays/object this means the references only are the dirty checked, since product array isn’t change. Hence ngOnchange will not execute for that the solution we can use ngDocheck
* Detect and act upon changes that angular cant or wont detects its own
* Called during the every change detection run, immediately after ngOnchanges and ngOninit

**ngAfterContentInit**

* ngAfterContentInit fires after the component’s content DOM initializes (loads for the first time).
* Waiting on @ContentChild(ren) queries is the hook’s primary use-case.

**ngAfterContentChecked**

* ngAfterContentChecked fires after every cycle of change detection targeting the content DOM.
* This lets developers facilitate how the content DOM reacts to change detection.
* ngAfterContentChecked can fire frequently and cause performance issues if poorly implemented.

**ngAfterViewInit**

* ngAfterViewInit fires once after the view DOM finishes initializing.
* The view always loads right after the content.
* ngAfterViewInit waits on @ViewChild(ren) queries to resolve. These elements are queried from within the same view of the component.

### ngAfterViewChecked

* ngAfterViewChecked fires after any change detection cycle targeting the component’s view.
* The ngAfterViewChecked hook lets developers facilitate how change detection affects the view DOM.