Lifecycle event sequence

After your application instantiates a component or directive by calling its constructor, Angular calls the hook methods you have implemented at the appropriate point in the lifecycle of that instance.

Angular executes hook methods in the following sequence. Use them to perform the following kinds of operations.

HOOK METHOD	PURPOSE	TIMING
`ngOnChanges()`	Respond when Angular sets or resets data-bound input properties. The method receives a SimpleChanges object of current and previous property values. NOTE: This happens frequently, so any operation you perform here impacts performance significantly. See details in Using change detection hooks in this document.	Called before `ngOnInit()` (if the component has bound inputs) and whenever one or more data-bound input properties change. NOTE: If your component has no inputs or you use it without providing any inputs, the framework will not call `ngOnChanges()`.
`ngOnInit()`	Initialize the directive or component after Angular first	Called once, after the first `ngOnChanges()`. `ngOnInit()` is still called even when `ngOnChanges()` is not

HOOK METHOD	PURPOSE	TIMING
	displays the data-bound properties and sets the directive or component's input properties. See details in Initializing a component or directive in this document.	(which is the case when there are no template-bound inputs).
`ngDoCheck()`	Detect and act upon changes that Angular can't or won't detect on its own. See details and example in Defining custom change detection in this document.	Called immediately after `ngOnChanges()` on every change detection run, and immediately after `ngOnInit()` on the first run.
`ngAfterContentInit()`	Respond after Angular projects external content into the component's view, or into the view that a directive is in. See details and example in Responding to changes in	Called *once* after the first `ngDoCheck()`.

HOOK METHOD	PURPOSE	TIMING
	content in this document.	
`ngAfterContentChecked()`	Respond after Angular checks the content projected into the directive or component. See details and example in Responding to projected content changes in this document.	Called after `ngAfterContentInit()` and every subsequent `ngDoCheck()`.
`ngAfterViewInit()`	Respond after Angular initializes the component's views and child views, or the view that contains the directive. See details and example in Responding to view changes in this document.	Called *once* after the first `ngAfterContentChecked()`.
`ngAfterViewChecked()`	Respond after Angular checks the component's views and child views, or the view that	Called after the `ngAfterViewInit()` and every

HOOK METHOD	PURPOSE	TIMING
	contains the directive.	subsequent `ngAfterContentChecked()`.
`ngOnDestroy()`	Cleanup just before Angular destroys the directive or component. Unsubscribe Observables and detach event handlers to avoid memory leaks. See details in Cleaning up on instance destruction in this document.	Called immediately before Angular dest