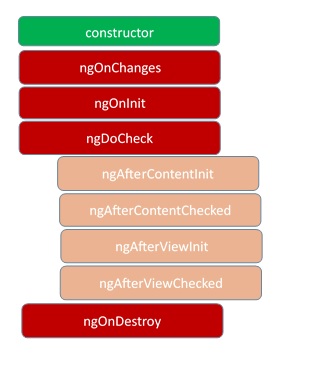
**Lifecycle hooks**

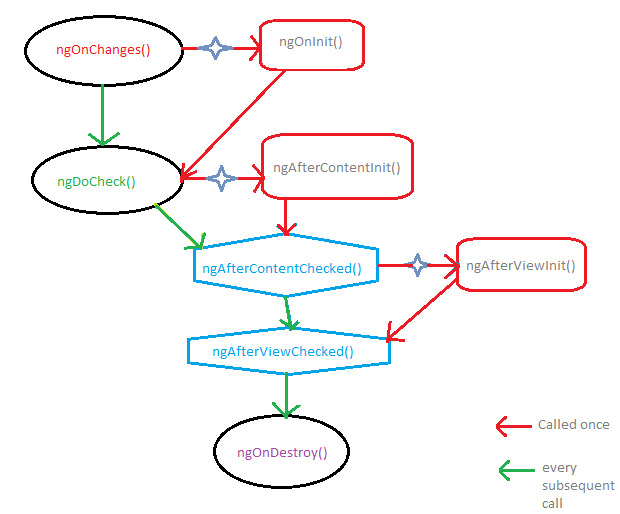
Every Angular component goes through the process of creation, then Angular goes on to execute all the functions it was created to execute and then go to possible destruction, this is called the lifecycle of a component.

Angular does this by creating the component, rendering it as well as creating and rendering all its children. Then Angular checks for changes in the data properties or in the DOM, and makes the appropriate updates and finally when it is done, destroys it then removes it from the DOM.

**There are eight lifecycle hooks in Angular:**

* ngOnChanges()
* ngOnInit()
* ngDoCheck()
* ngAfterContentInit()
* ngAfterContentChecked()
* ngAfterViewInit()
* ngAfterViewChecked()
* ngonDestroy()





## ngOnChanges()

This is the very first lifecycle hook, it is called right after your class gets initialized and the component is created the ngOnChanges() is called. You might want to wonder why ngOnInit hook is not called first, but that is because Angular counts that very first class initialization as a data property change. So the hook that gets called once a data property change (like resetting values) occurs is ngOnChanges(). This hook is basically called after the constructor is called and any other time there is a property change inside your component.

Open up the starter project you downloaded in VS Code and open the app.component.ts file inside the app folder. It should look like this:

*// src/app/app.component.ts*

import { Component } from '@angular/core';

**@Component**({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'ngcanvas';

}

To test out the ngOnChanges hook, copy in the code below inside this file:

*// src/app/app.component.ts*

import { Component, OnChanges } from '@angular/core';

**@Component**({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent implements OnChanges {

ngOnChanges(changes: import("@angular/core").SimpleChanges): void {

}

title = 'ngcanvas';

constructor(){

alert("1. on changes called");

}

}

Here we have brought in the ngOnChanges hook from Angular and have initialized it with an alert statement in the constructor. If you run the application, the alert will pop up before the component will be loaded.

## ngOnInit()

This is the second lifecycle hook called by Angular, it is called right after the very first ngOnChanges hook is called. It is only called once, it initializes the component, sets and displays component input properties. It is the most important lifecycle hook in Angular as it signals the activation of the created component. For the fact that this hook is called only once, it is therefore great for fetching data from external sources like servers and APIs.

To test this out, add the ngOnInit hook syntax to the class with the code below:

*// src/app/app.component.ts*

import { Component, OnChanges, OnInit } from '@angular/core';

**@Component**({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent implements OnChanges, OnInit {

ngOnInit(): void {

alert("2. on init is called");

}

ngOnChanges(changes: import("@angular/core").SimpleChanges): void {

}

title = 'ngcanvas';

constructor(){

alert("1. on changes called");

}

}

You will see the alerts pop up in the right sequence as it is called, this will be true even if ngOnInit comes first from a top-down analysis.

## ngDoCheck()

This is the third Angular lifecycle hook that gets called on a component. It is called during every change detection run, Angular has an internal system that goes around the component processes every so often looking for changes that the compiler cannot detect on its own. This hook is called at every change detection run, usually after the ngOnInit hook is called.

To test this, copy in the code below into the app.component.ts file:

*// src/app/app.component.ts*

import { Component, OnChanges, OnInit, DoCheck } from '@angular/core';

**@Component**({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent implements OnChanges, OnInit, DoCheck {

ngDoCheck(): void {

alert("3. do check is called");

}

ngOnInit(): void {

alert("2. on init is called");

}

ngOnChanges(changes: import("@angular/core").SimpleChanges): void {

}

title = 'ngcanvas';

constructor(){

alert("1. on changes called");

}

}

You will see the alerts pop up in the right sequence here again, not minding the hierarchy of presentation.

## ngAfterContentInit()

This is the fourth lifecycle hook Angular calls after a component has been initialized. This hook is called only once immediately after the first ngDoCheck hook is called, it is a kind of ngDoCheck but for content projected into the component view with ng-content. You may refer to the brief content projection summary at the beginning of this post again.

At this point you must have noticed the amazing power of Angular with intellisense in VS Code to handle imports and also initialization of these hooks perfectly.

*// src/app/app.component.ts*

import { Component, OnChanges, OnInit, DoCheck, AfterContentInit } from '@angular/core';

**@Component**({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent implements OnChanges, OnInit, DoCheck, AfterContentInit {

ngAfterContentInit(): void {

alert("4. after content init called");

}

ngDoCheck(): void {

alert("3. do check is called");

}

ngOnInit(): void {

alert("2. on init is called");

}

ngOnChanges(changes: import("@angular/core").SimpleChanges): void {

}

title = 'ngcanvas';

constructor(){

alert("1. on changes called");

}

}

## ngAfterContentChecked()

This is the fifth lifecycle hook Angular calls after a component has been initialized. It is called after the content projected into a component view is initialized, after the ngAfterContentInit hook and every subsequent ngDoCheck hook is called.

Exercise: To test this hook, follow the pattern in the previous hooks above and achieve create the alert for this.

## ngAfterViewInit()

This is the sixth lifecycle hook Angular calls after a component has been initialized. It is called only once after the very first ngAfterContentChecked hook is called. It is called after Angular initializes component views and the subsequent child views under each component, this will have to include the views displayed through content projection too and that is why it is called after the ngAfterContentChecked hook.

Exercise: To test this hook, follow the pattern in the previous hooks above and achieve create the alert for this.

## ngAfterViewChecked()

This is the seventh lifecycle hook Angular calls after a component has been initialized. It is called after Angular checks the component views and the subsequent child views under each component for changes, this includes the views displayed through content projection too. It is called after the ngAfterViewInit hook and every subsequent ngAfterContentChecked hook.

Exercise: To test this hook, follow the pattern in the previous hooks above and achieve create the alert for this.

## ngOnDestroy()

This is the last Angular lifecycle hook, it is called just before the component is removed from the DOM. Inside it clean up of the component is done, from detaching event handlers to unsubscribing from observables.

Exercise: To test this hook, follow the pattern in the previous hooks above and achieve create the alert for this. You will also notice that ngOnDestroy hook is not called, that is because the DOM has not been removed, if you have an unsubscribe statement inside it for instance, it will get called. Your final app.component.ts file should look like this:

*// src/app/app.component.ts*

import { Component, OnChanges, OnInit, DoCheck, AfterContentInit, AfterContentChecked, AfterViewInit, AfterViewChecked, OnDestroy } from '@angular/core';

**@Component**({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent implements OnChanges, OnInit,

DoCheck, AfterContentInit, AfterContentChecked, AfterViewInit,

AfterViewChecked, OnDestroy{

ngOnChanges(changes: import("@angular/core").SimpleChanges): void {

}

ngOnInit(): void {

alert("2. on init is called");

}

ngDoCheck(): void {

alert("3. do check is called");

}

ngAfterContentInit(): void {

alert("4. after content init called");

}

ngAfterContentChecked(): void {

alert("5. after content check called");

}

ngAfterViewInit(): void {

alert('6. after view init called');

}

ngAfterViewChecked(): void {

alert('7. after view init checked');

}

ngOnDestroy(): void {

alert('8. on destroy called');

}

title = 'ngcanvas';

constructor(){

alert("1. on changes called");

}

}

The ideal behavior is for the ngDoCheck, ngAfterContentChecked and ngAfterViewChecked hooks to be called multiple times as changes occur so do not be surprised they get called more than once.