**Lesson21 Life Cycle in Angular**

**Notes:-**

**1-A component has a lifecycle managed by Angular:-**

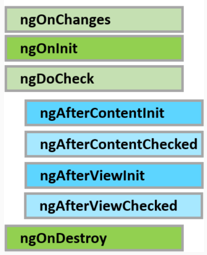
**1-Creates the component**

**2-Renders the component**

**3-Creates and renders the component children**

**4-Checks when the component data-bound properties change, and**

**5-Destroys the component before removing it from the DOM**



**The 3 most commonly used hooks are**

|  |  |
| --- | --- |
| **Life Cycle Hook** | **Purpose** |
| ngOnChanges | Executes, every time the value of an input property changes. The hook method receives a SimpleChanges object containing current and previous property values. This is called before ngOnInit |
| ngOnInit | Executes after the constructor and after ngOnChange hook for the first time. It is most commonly used for component initialisation and retrieving data from a database |
| ngOnDestroy | Executes just before angular destroys the component and generally used for performing cleanup |

**There are 3 simple steps to use the Life Cycle Hooks**

**Step 1: Import the Life Cycle Hook interface. For example, to use ngOnInit() life cycle hook, import OnInit interface.  
  
import { OnInit } from '@angular/core';  
  
Step 2 : Make the component class implement the Life Cycle Hook interface, using the implements keyword as shown below. This step is optional, but good to have so you will get editor support and flags errors at compile time if you incorrectly implement the interface method or make any typographical errors.  
  
export class SimpleComponent implements OnInit { }**

**//Note:- the benefit of the Implementation is that the sensitivity of writing the method that implement from the interface to be sensitive   
  
Step 3: Write the implementation code for the life cycle interface method. Each interface has a single hook method whose name is the interface name prefixed with ng.  
  
ngOnInit() {  
    console.log('OnInit Life Cycle Hook');}**

**Steps:-**

**1-on the inner component we apply OnChanges and apply the input variable and apply the onNgChange**

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

**@Component({**

**selector: 'app-simple-comp',**

**//templateUrl: './simple-comp.component.html',**

**template:'You Entered : {{simpleInput}}',**

**styleUrls: ['./simple-comp.component.css']**

**})**

**export class SimpleCompComponent implements OnChanges {**

**// Input property. As and when this property changes**

**// ngOnChanges life cycle hook method is called**

**@Input() simpleInput: string;**

**// Step 3 : Implementation for the hook method**

**// This code logs the current and previous value**

**// to the console.**

**//This variaable called changes contain the current and previous values and the property name of Input type**

**ngOnChanges(changes: SimpleChanges) {**

**for (let propertyName in changes) {**

**let change = changes[propertyName];**

**let current = JSON.stringify(change.currentValue);**

**let previous = JSON.stringify(change.previousValue);**

**console.log(propertyName + ': currentValue = '**

**+ current + ', previousValue = ' + previous);**

**// The above line can be rewritten using**

**// placeholder syntax as shown below**

**// console.log(`${propertyName}: currentValue**

**// = ${current }, previousValue = ${previous }`);**

**}}}**

**2-we link the input variable and the local variable and we see that the each time the input variable change , the ngOnChange is execute**

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

@Component({

selector: 'app-root',

//we apply the inner component and link the input variable from the inner component and the local variable

🡺 so the method called ngOnChange is called when every time the input variable change

template: `Your Text : <input type='text' [(ngModel)]='userText'/>

<br/><br/>

<app-simple-comp [simpleInput]='userText'></app-simple-comp>`,

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

})

export class AppComponent {

title = 'Mohammed Enbah';

userText: string = 'Pragim';

}