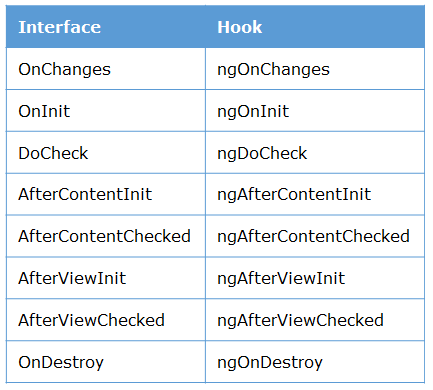
Till now we have learned to share data between components. Now let us learn about the life cycle of a component.

A component has a life cycle managed by Angular which consists of creating a component, rendering it, creating and rendering its child components, checks when its data-bound properties change and destroys it before removing it from the DOM

Angular has some methods or hooks which provide visibility into these key life moments of a component and the ability to act when they occur.

Following are the life-cycle hooks of a component. The methods will be invoked in the same order as mentioned in the below table



In this course, we will cover only two life-cycle hooks, namely **ngOnInit**and **ngOnDestroy**

**Lifecycle Hooks**

* **ngOnInit** – It will be invoked when Angular initializes the directive or component. This will be called **after**the *constructor()*. Typically constructor should not have any code and the setup code should be inside *ngOnInit().*
* **ngOnDestroy** – It will be invoked before Angular destroys directive or component

**Syntax**:

1. import { Component, OnInit, OnDestroy } from '@angular/core';
2. ...
3. export class AppComponent implements OnInit, OnDestroy {
5. ngOnInit() { }
7. ngOnDestroy() { }
8. }

**Line 1:** Import the interfaces of life-cycle hooks

**Line 4:** Inherit interfaces which have life-cycle methods to override

**Line 6,8:** Override the specific life-cycle hooks

For more information on [Life-cycle hooks](https://angular.io/guide/lifecycle-hooks), refer here.

Let us learn to implement life-cycle hooks through a demo.

**Highlights:**

* Understanding component life-cycle
* Exploring and overriding various life-cycle hooks

**Demo Steps:**

**Problem Statement:**Overriding component life-cycle hooks and logging the corresponding messages to understand the flow.

Output should be as shown below :

**Inside parent component : AppComponent**

**Inside child component : LoginComponent**

1. Write the below code in **app.component.ts**

1. import { Component, OnInit, OnDestroy } from '@angular/core';
2. @Component({
3. selector: 'app-root',
4. styleUrls: ['./app.component.css'],
5. templateUrl: './app.component.html'
6. })
7. export class AppComponent implements OnInit{
9. ngOnInit() {
10. console.log('Inside parent component : AppComponent')
11. }
13. }

**Line 4:** Inherit all life-cycle interfaces

**Line 8-10:** Overriding required life-cycle methods and logging a message

**Note:**ngOnInit() is the first method to be invoked for AppComponent.

2. Create a new component name Login, and have the below code in **LoginComponent**

1. @Component({
2. selector: 'app-login',
3. templateUrl: './login.component.html',
4. styleUrls: ['./login.component.css']
5. })
6. export class LoginComponent implements OnInit {
7. ngOnInit(){
8. console.log('Inside child component : LoginComponent')
9. }
10. }

3. Save the files and check the output in the browser and refer to the messages logged in the VS Code terminal.