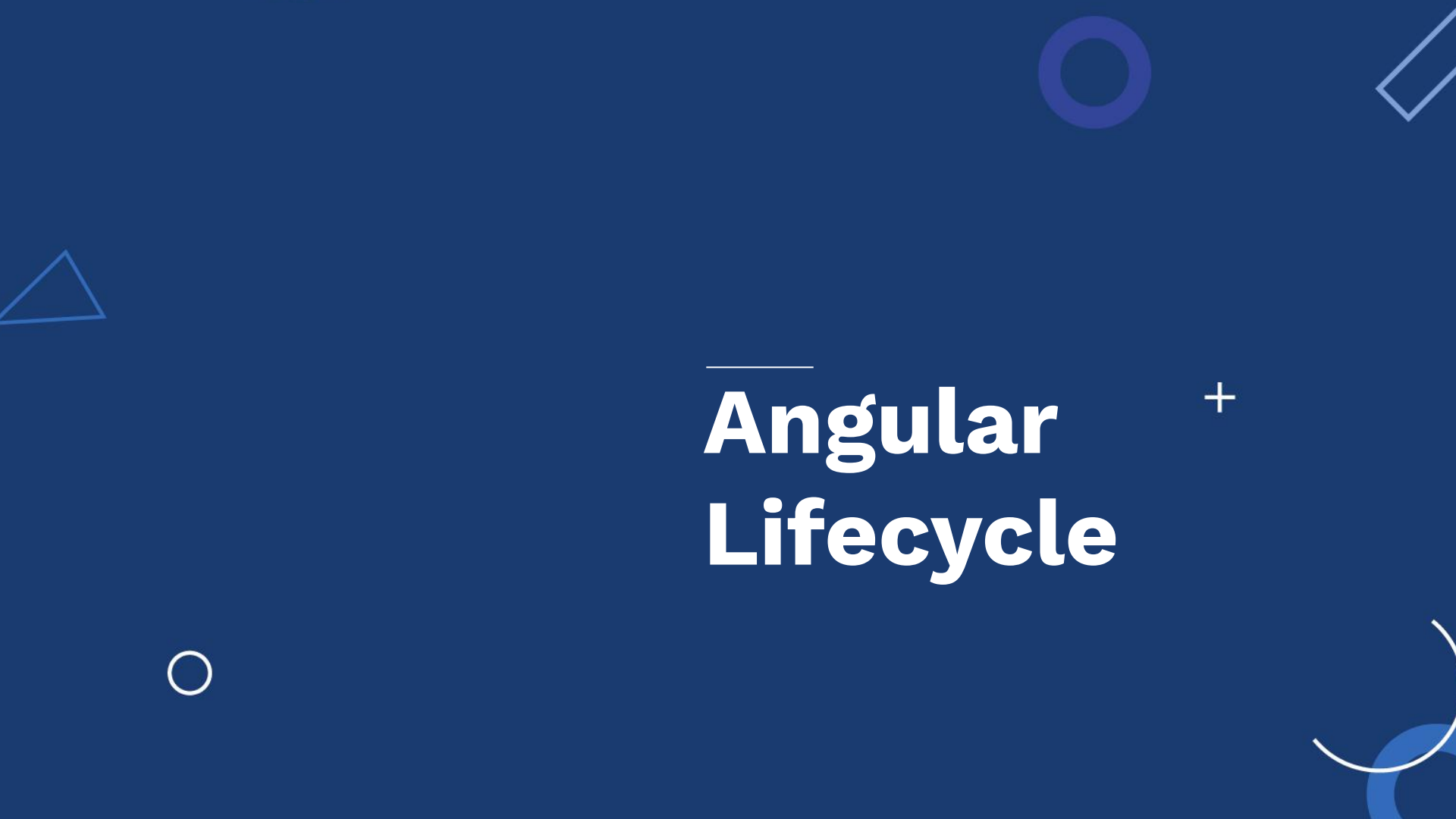




# Front End Development 1

## Sesi 20

---



# Angular Lifecycle

+

# Lifecycle Hooks

Lifecycle hooks adalah fungsionalitas dalam Angular yang memperbolehkan kita untuk “menyelipkan” proses/code kemudian menjalankannya di siklus (lifecycle) spesifik dari suatu komponen atau directives.

Angular manage “siklus hidup” dari suatu komponen, mulai dari pembuatan, update, atau ketika menghapusnya. Dengan lifecycle hooks ini, kita bisa mengontrol siklus tersebut.

Untuk dapat menggunakan lifecycle hooks, kita harus menambahkan hooks method, yang dimulai dengan prefix **ng** dalam komponen atau directives.

# Hooks Method: Event Sequence

constructor

**ngOnChanges**

**ngOnInit**

**ngDoCheck**

**ngAfterContentInit**

**ngAfterContentChecked**

**ngAfterViewInit**

**ngAfterViewChecked**

**ngOnDestroy**

Hooks Methods ini dibagi menjadi 2 kategori

1. Hooks untuk component atau directives (berwarna hijau di gambar)
2. Hooks untuk child component (berwarna biru di gambar)

# Hooks Method: Penjelasan

Hooks	Penjelasan
<code>ngOnChanges()</code>	Dipanggil sebelum dilakukan <code>ngOnInit()</code> dan ketika terjadinya pergantian data pada <i>input properties</i> .
<code>ngOnInit()</code>	Dipanggil saat pertama kali melakukan inisiasi pada komponen atau directive. <code>NgOnInit()</code> akan dipanggil setelah <code>ngOnChanges()</code>
<code>ngDoCheck()</code>	Dipanggil setelah <code>ngOnInit()</code> . Hooks ini mengidentifikasi perubahan yang tidak dapat dihandle oleh angular.
<code>ngAfterContentInit()</code>	Merupakan hooks yang mengakomodasi event saat terdapat konten external(selain dari angular) masuk pada component view. Hook ini hanya terdapat pada komponen.

Hooks	Penjelasan
<code>ngAfterContentChecked()</code>	Angular akan melakukan <i>check</i> (perubahan) pada external component. Hook ini hanya terdapat pada komponen.
<code>ngAfterViewInit()</code>	Hook yang dipanggil ketika menginisiasi <i>component views</i> dan <i>child views</i> . Hook ini hanya terdapat pada komponen
<code>ngAfterViewChecked()</code>	Hook yang dipanggil ketika angular telah melakukan check pada <i>component views</i> dan <i>child views</i> . Hook ini hanya terdapat pada komponen
<code>ngOnDestroy()</code>	Hook yang dipanggil sebelum angular menghancurkan( <i>destroy</i> ) komponen atau directive.

# Lifecycle Hooks: Interface

```
1  import {
2    Component,
3    OnInit,
4    OnChanges,
5    SimpleChanges,
6    Input,
7    AfterViewInit,
8    DoCheck,
9    AfterViewChecked,
10   AfterContentChecked,
11   AfterContentInit,
12   OnDestroy
13 } from '@angular/core';
14
15 @Component({
16   selector: 'app-component-overview',
17   templateUrl: './component-overview.component.html',
18   styleUrls: ['./component-overview.component.css']
19 })
20
21 export class ComponentOverviewComponent implements
22   OnChanges,
23   OnInit,
24   DoCheck,
25   AfterViewInit,
26   AfterViewChecked,
27   AfterContentChecked,
28   AfterContentInit,
29   OnDestroy {
30   // kode disini
31 }
```

Interface di lifecycle berfungsi sebagai abstraksi. Dan bersifat OPTIONAL. Kita akan tetap bisa menggunakan hooks method, walau tidak mencantumkan interface ini.

Namun, penggunaan interface ini akan berguna untuk:

1. Abstraksi/overview untuk memperjelas event lifecycle mana yang digunakan.
2. Kompiler Typescript akan memberikan warning apabila kita tidak mengimplementasikan hooks method walau sudah mencantumkan interface ini.

# Hooks Method: Cara Penggunaan di Komponen

```
component-overview.component.ts x
src > app > component-overview > component-overview.component.ts > ComponentOverview
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-component-overview',
5    templateUrl: './component-overview.component.html',
6    styleUrls: ['./component-overview.component.css']
7  })
8
9  export class ComponentOverviewComponent implements OnInit {
10     constructor() { }
11     ngOnInit() {
12       console.log('ngOnInit');
13     }
14     ngOnChanges(){
15       console.log('ngOnChanges');
16     }
17     ngDoCheck(){
18       console.log('ngDoCheck');
19     }
20     ngAfterContentInit(){
21       console.log('ngAfterContentInit');
22     }
23     ngAfterContentChecked(){
24       console.log('ngAfterContentChecked');
25     }
26     ngAfterViewInit(){
27       console.log('ngAfterViewInit');
28     }
29     ngAfterViewChecked(){
30       console.log('ngAfterViewChecked');
31     }
32     ngOnDestroy(){
33       console.log('ngOnDestroy');
34     }
35   }
```

- **OnInit**, merupakan interface lifecycle yang dicantumkan agar bisa menggunakan hooks method **ngOnInit()**. (OPTIONAL)
- **constructor** dan **ngOnInit** hanya dipanggil sekali.
- **ngOnInit()** dapat digunakan sebagai tempat untuk fetch initial data.
- **ngOnChanges** akan dipanggil berulang kali bila ada perubahan.

\*Hooks method hanya digunakan seperlunya saja, tidak perlu mendefinisikan semua hooks dalam komponen.