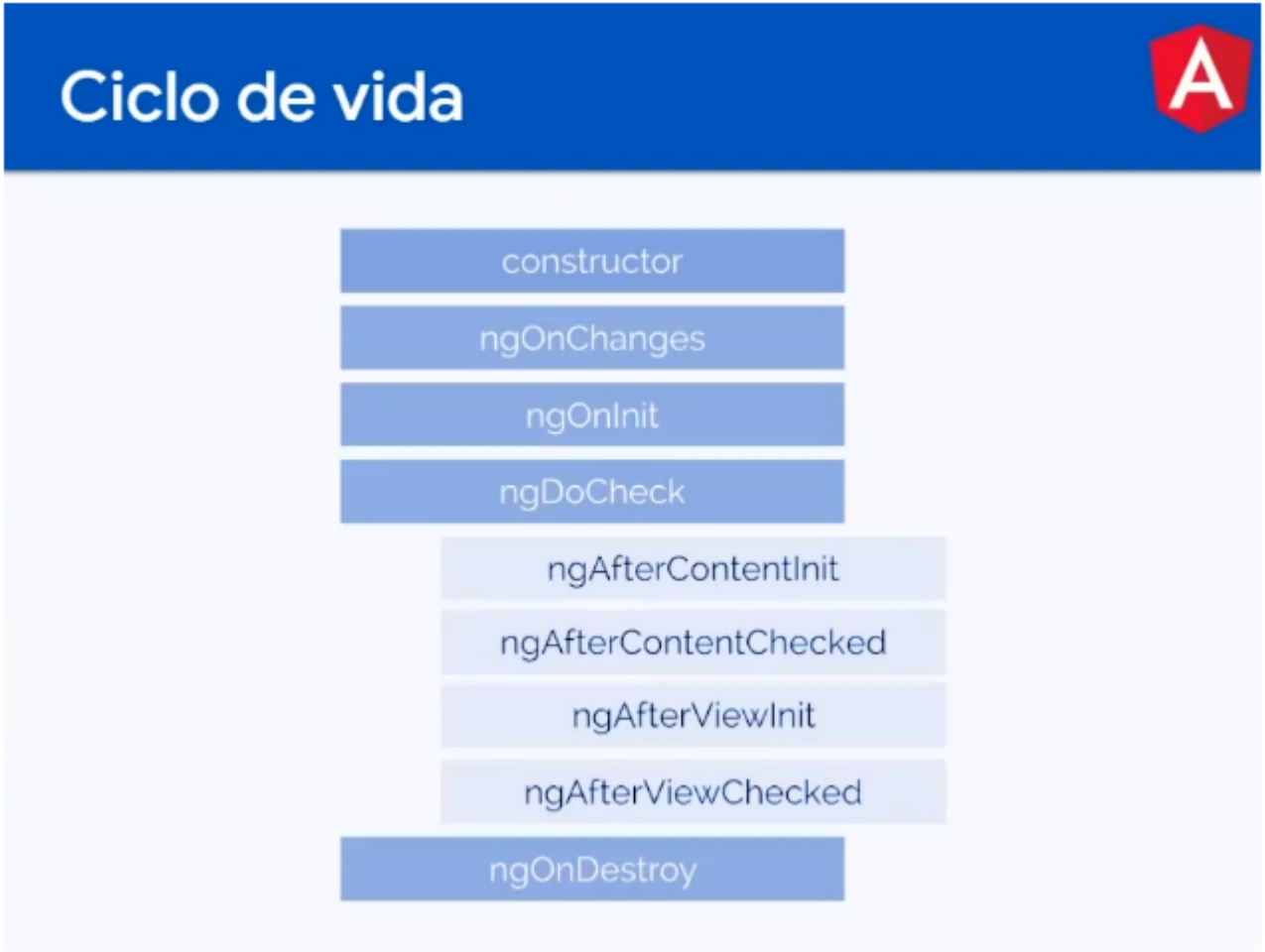
Ciclo de vida de componentes 6/20

RECURSOS MARCADORES

Un componente pasa por varias etapas en su ciclo de vida. A través de hooks, **puedes realizar una determinada acción** cuando el componente es inicializado, cuando se dispara un evento, cuando se detecta un cambio, cuando el componente es destruido, etc.

A continuación, se detalla la secuencia de eventos y el orden de los mismos:



Hooks más utilizados

Constructor

Como en toda clase en la programación orientada a objetos, el `constructor` es quien crea la instancia del objeto y sus dependencias.

- Solo se ejecuta una vez antes del render del componente.
- No tiene que utilizarse para procesos asincrónicos.

ngOnChanges

El hook `ngOnChanges()` se dispara cada vez que se produce un cambio de estado en el componente. Cuando una variable cambia de valor, por ejemplo o ante el cambio de valor de un Input.

- Se ejecuta N cantidad de veces antes y durante el render del componente.
- Puede emplearse para procesos asincrónicos.

ngOnInit

Es el hook más usado, `ngOnInit()` es ideal para cualquier solicitud de datos asincrónicos a una API para preparar el componente antes de renderizarlo.

- Únicamente se ejecuta una vez, antes del render del componente.
- Puede usarse para procesos asincrónicos.

ngAfterViewInit

Este hook únicamente se ejecuta una vez cuando el render del componente haya finalizado. Puede serte útil para realizar acciones programáticas que requieran que todo el HTML del componente ya este preparado.

- Únicamente se ejecuta una vez después del render del componente.

ngOnDestroy

Finalmente, `ngOnDestroy()` se ejecutará cuando el componente es destruido, o sea, cuando ya no existe en la interfaz del navegador. Suele utilizarse para liberar espacios de memoria que el componente requiera.

Usando hook

Los hooks de ciclo de vida de Angular, son interfaces que tienen que importarse desde `@angular/core` para implementarlos en la clase y así detectar los cambios en cada evento.

```
import { Component, OnInit, AfterContentInit, OnDestroy } from '@angular/core';

@Component({
  selector: 'app-test-name',
  templateUrl: './test-name.component.html',
  styleUrls: ['./test-name.component.less']
})
export class TestNameComponent implements OnInit, AfterContentInit, OnDestroy {

  constructor() {
    console.log('1. Primero sucederá esto');
  }

  ngOnInit(): void {
    console.log('2. Luego esto');
  }

  ngAfterViewInit(): void {
    console.log('3. Seguido de esto');
  }

  ngOnDestroy(): void {
    console.log('4. Finalmente esto (cuando el componente sea destruido)');
  }

}
```

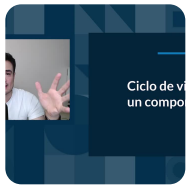
Cada hook tiene sus características y utilidades recomendadas dependiendo lo que necesitas ayer. Es importante seguir estas recomendaciones para buscar optimizar el rendimiento de tu aplicación.

Contribución creada con los aportes de Kevin Fiorentino.

Lecturas recomendadas

GitHub - platzi/angular-componentes at 5-step
https://github.com/platzi/angular-componentes/tree/5-step

Clases relacionadas



Ciclo de vida de un componente

Curso Básico de Web Components con JavaScript

Entendiendo el ciclo de vida de los componentes

Curso de Vue.js: Componentes y Composition API

Ciclos de vida de un componente

Curso de Reactividad con Vue.js 3