

CICLO DE VIDA DE

ANGULAR

ONINIT

Descripción:

Esta es la fase en la que Angular inicializa el componente o la directiva después de crearlo. Es el momento ideal para inicializar propiedades y cargar datos que el componente necesita.

Analogía:

Es como cuando te mudas a una nueva casa y desempacas tus cosas por primera vez. Configuras tus muebles y decoras para que todo esté listo para vivir.

```
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-ejemplo',
5   template: `<h1>{${titulo}}</h1>`,
6 })
7 export class EjemploComponent implements OnInit {
8   titulo: string;
9
10 constructor() {
11   this.titulo = 'Cargando...';
12 }
13
14 ngOnInit() {
15   this.titulo = 'Componente Inicializado';
16 }
17 }
```

```
import { Component, Input, OnChanges, SimpleChanges } from
@Directive({
  selector: 'app-hijo',
  template: `<p>${datos}</p>`,
})
export class HijoComponent implements OnChanges {
  @Input() datos: string;

  ngOnChanges(changes: SimpleChanges) {
    console.log('Cambios detectados:', changes);
  }
}
```

ONCHANGES

Descripción:

Esta fase se activa cada vez que se detectan cambios en las propiedades vinculadas con un componente hijo. Es útil para reaccionar a los cambios de los datos de entrada.

Analogía:

Imagina que tienes una pizarra donde escribes y borras información. Cada vez que algo cambia en la pizarra, lo notas inmediatamente y actúas en consecuencia.

AFTERCONTENTINIT

Descripción:

Este hook se ejecuta después de que Angular ha proyectado el contenido en el componente. Es útil para inicializar lógica que depende del contenido proyectado.

Analogía:

Es como cuando preparas una fiesta en tu casa y esperas a que lleguen todos los invitados para empezar las actividades principales.

```
import { Component, AfterContentInit, ContentChild } from
@Directive({
  selector: 'app-padre',
  template: `<ng-content></ng-content>`,
})
export class PadreComponent implements AfterContentInit {
  @ContentChild('contenidoHijo') contenidoHijo: any;

  ngAfterContentInit() {
    console.log('Contenido hijo:', this.contenidoHijo);
  }
}
```

AFTERCONTENTCHECKED

Descripción:

Este hook se llama después de que el contenido proyectado se ha verificado. Es útil para realizar tareas adicionales después de que Angular ha verificado el contenido.

Analogía:

Es como revisar una vez más el contenido de un documento después de haberlo escrito para asegurarte de que todo está correcto.

AFTERVIEWINIT

Descripción:

Este hook se llama después de que Angular ha inicializado las vistas del componente. Es útil para inicializar lógica que depende de la vista del componente.

Analogía:

Es como cuando instalas una nueva televisión en tu sala y te aseguras de que la imagen y el sonido estén configurados correctamente una vez encendida.

```
import { Component, AfterViewInit, ViewChild } from
@Directive({
  selector: 'app-vista',
  template: `<p>#miParrafo>Texto en la vista</p>`,
})
export class VistaComponent implements AfterViewInit {
  @ViewChild('miParrafo') parrafo: any;

  ngAfterViewInit() {
    console.log('Parrafo en la vista:', this.parrafo);
  }
}
```

AFTERVIEWCHECKED

Descripción:

Este hook se llama después de que Angular ha verificado la vista del componente y de sus hijos. Es útil para ejecutar lógica adicional después de que las vistas han sido verificadas.

Analogía:

Es como hacer un último chequeo de seguridad en un avión antes de despegar para asegurarse de que todo está en orden.

ONDESTROY

Descripción:

Este hook se llama justo antes de que Angular destruya el componente o la directiva. Es ideal para realizar limpieza como la desuscripción de observables o la liberación de recursos.

Analogía:

Es como cuando te mudas de una casa, te aseguras de apagar todas las luces, cerrar las puertas y llevarte todas tus pertenencias.

```
import { Component, OnDestroy } from
@Directive({
  selector: 'app-destruccion',
  template: `<p>Destruyendo componente...</p>`,
})
export class DestruccionComponent implements OnDestroy {
  ngOnDestroy() {
    console.log('Componente destruido');
  }
}
```

DOCHECK

Descripción:

Este hook se llama durante cada ciclo de detección de cambios, permitiendo ejecutar lógica personalizada de verificación de cambios.

Analogía:

Es como un guardia de seguridad que realiza rondas periódicas para asegurarse de que todo está en orden

y detectar cualquier irregularidad.

```
import { Component, DoCheck } from
@Directive({
  selector: 'app-dock-check',
  template: `<p>DoCheck ejecutado</p>`,
})
export class DoCheckComponent implements DoCheck {
  ngDoCheck() {
    console.log('Detección de cambios personalizada');
  }
}
```