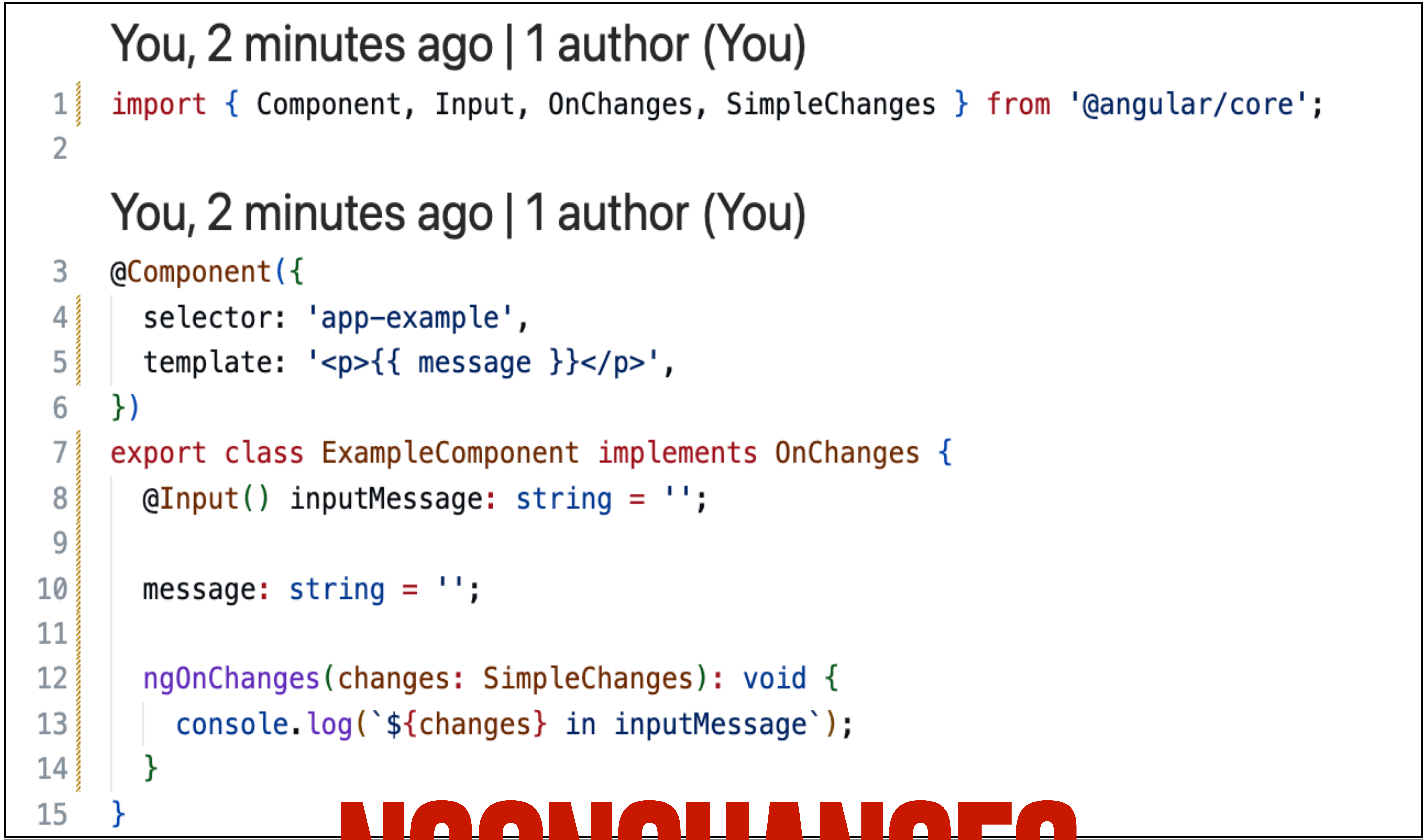
ANGULAR LIFECYCLE HOOKS

SOMIL SHARMA

INTRODUCTION

- In Angular, components are the building blocks of the application, and they go through a lifecycle that consists of several phases.
- The lifecycle hooks in Angular provide developers with the ability to tap into various stages of a component's lifecycle, allowing for the execution of custom logic at specific points.
- These hooks are crucial for managing the state, performing initialisation, and handling cleanup tasks.
- Angular follows a specific order when executing lifecycle hooks during the creation, change detection, and destruction of a component. The order is as follows:

  ngOnChanges → ngOnInit → ngDoCheck → ngAfterContentInit → ngAfterContentChecked → ngAfterViewInit → ngAfterViewChecked → ngOnDestroy



You, 2 minutes ago | 1 author (You)

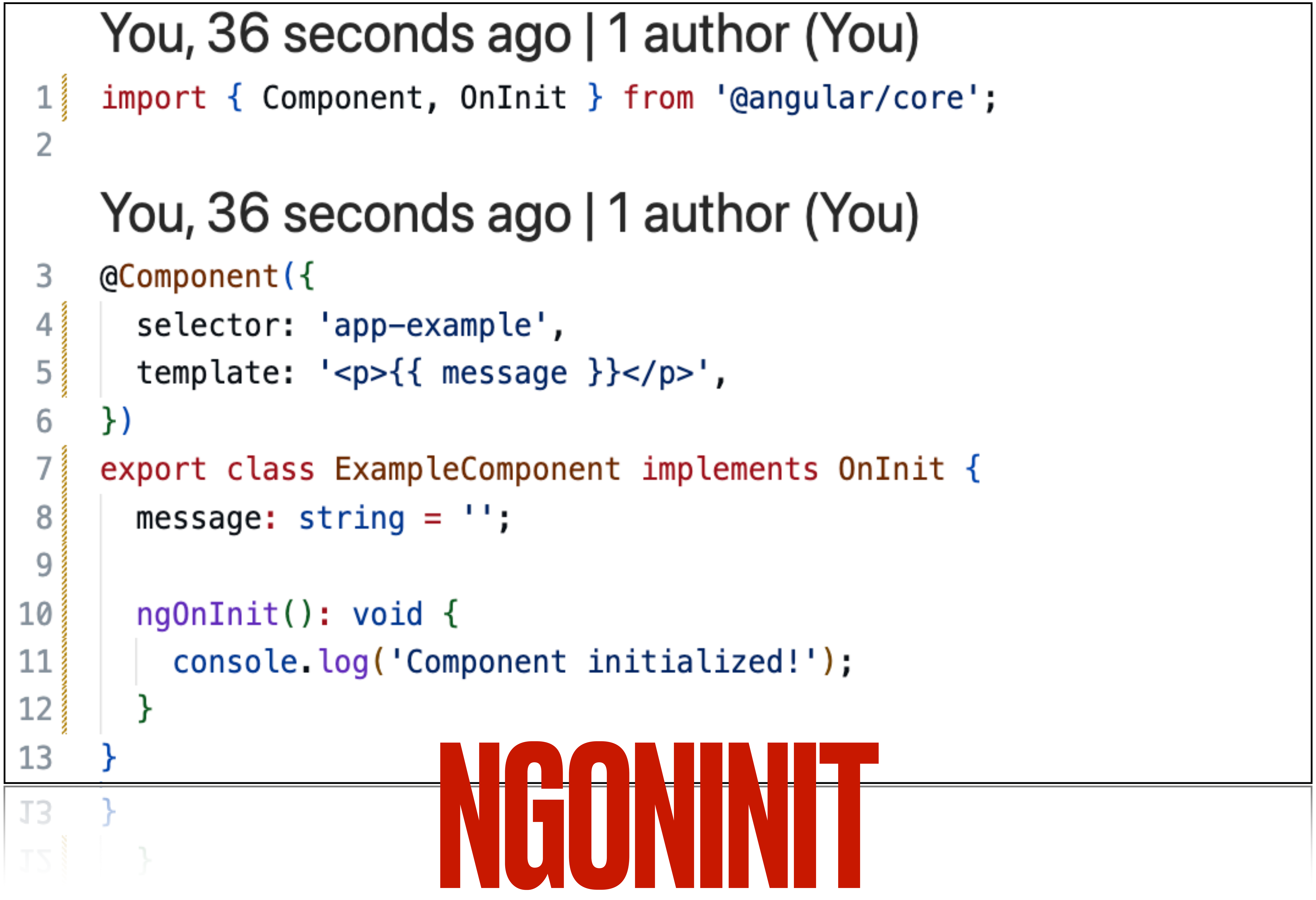
```
import { Component, Input, OnChanges, SimpleChanges } from '@angular/core';

```

You, 2 minutes ago | 1 author (You)

```
@Component({
  selector: 'app-example',
  template: '<p>{{ message }}</p>',
})
export class ExampleComponent implements OnChanges {
  @Input() inputMessage: string = '';

  message: string = '';

  ngOnChanges(changes: SimpleChanges): void {
    console.log(`${changes} in inputMessage`);
  }
}
```

NGONCHANGES

- The primary purpose of ngOnChanges is to allow the component to respond to changes in its input properties and perform necessary actions.
- The SimpleChanges object passed to ngOnChanges contains the names of the input properties as keys.
- For each property, it provides a SimpleChange object with previousValue and currentValue properties.



You, 36 seconds ago | 1 author (You)

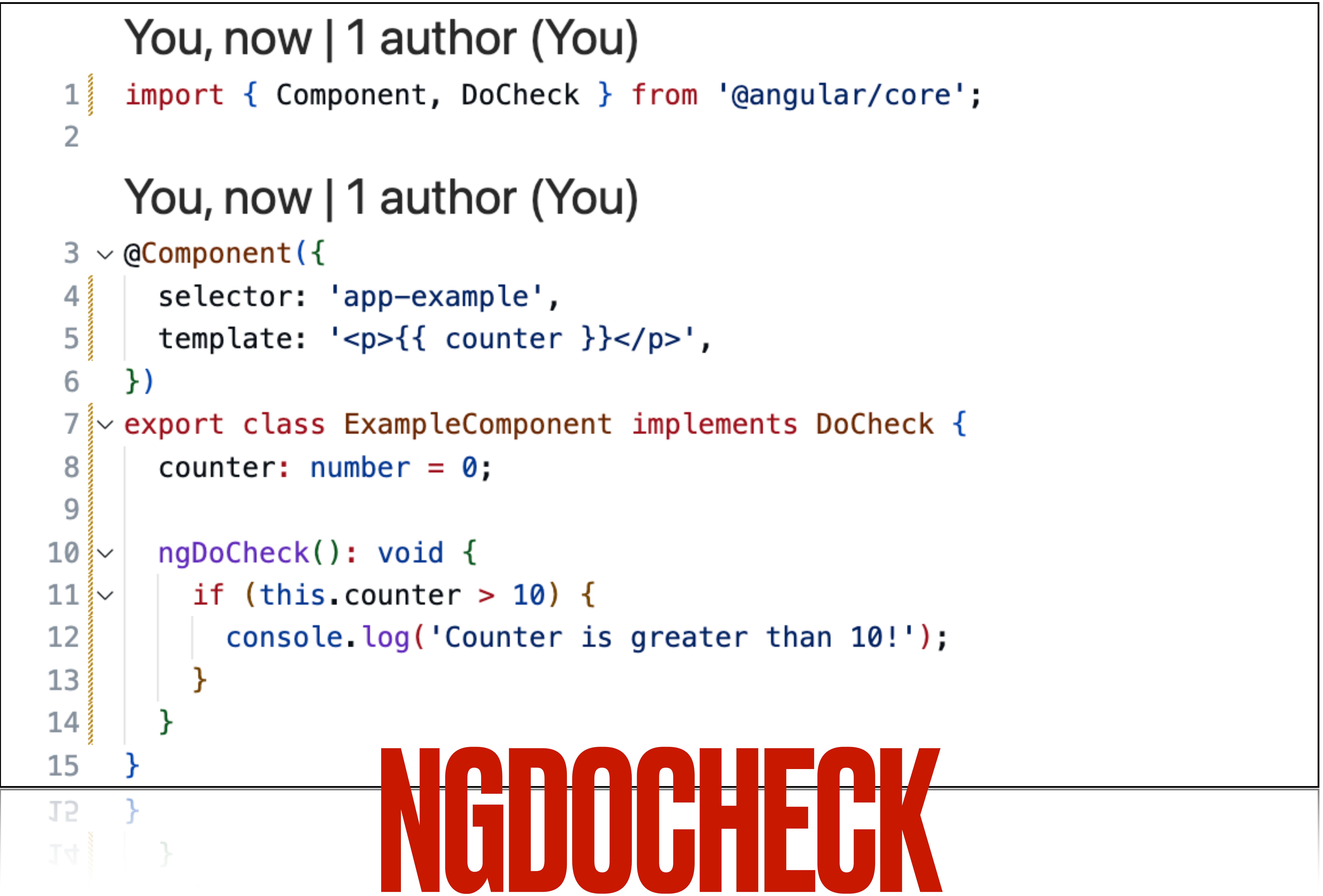
```
import { Component, OnInit } from '@angular/core';

```

You, 36 seconds ago | 1 author (You)

```
@Component({
  selector: 'app-example',
  template: '<p>{{ message }}</p>',
})
export class ExampleComponent implements OnInit {
  message: string = '';

  ngOnInit(): void {
    console.log('Component initialized!');
  }
}
```

NGONINIT

- The ngOnInit lifecycle hook in Angular is called after the component has been initialised, and its input properties have been bound.
- It is a one-time initialisation hook that provides a place for setting up the component, initialising data, and performing any tasks that need to occur once at the beginning of the component's lifecycle.


You, now | 1 author (You)

```
import { Component, DoCheck } from '@angular/core';

```

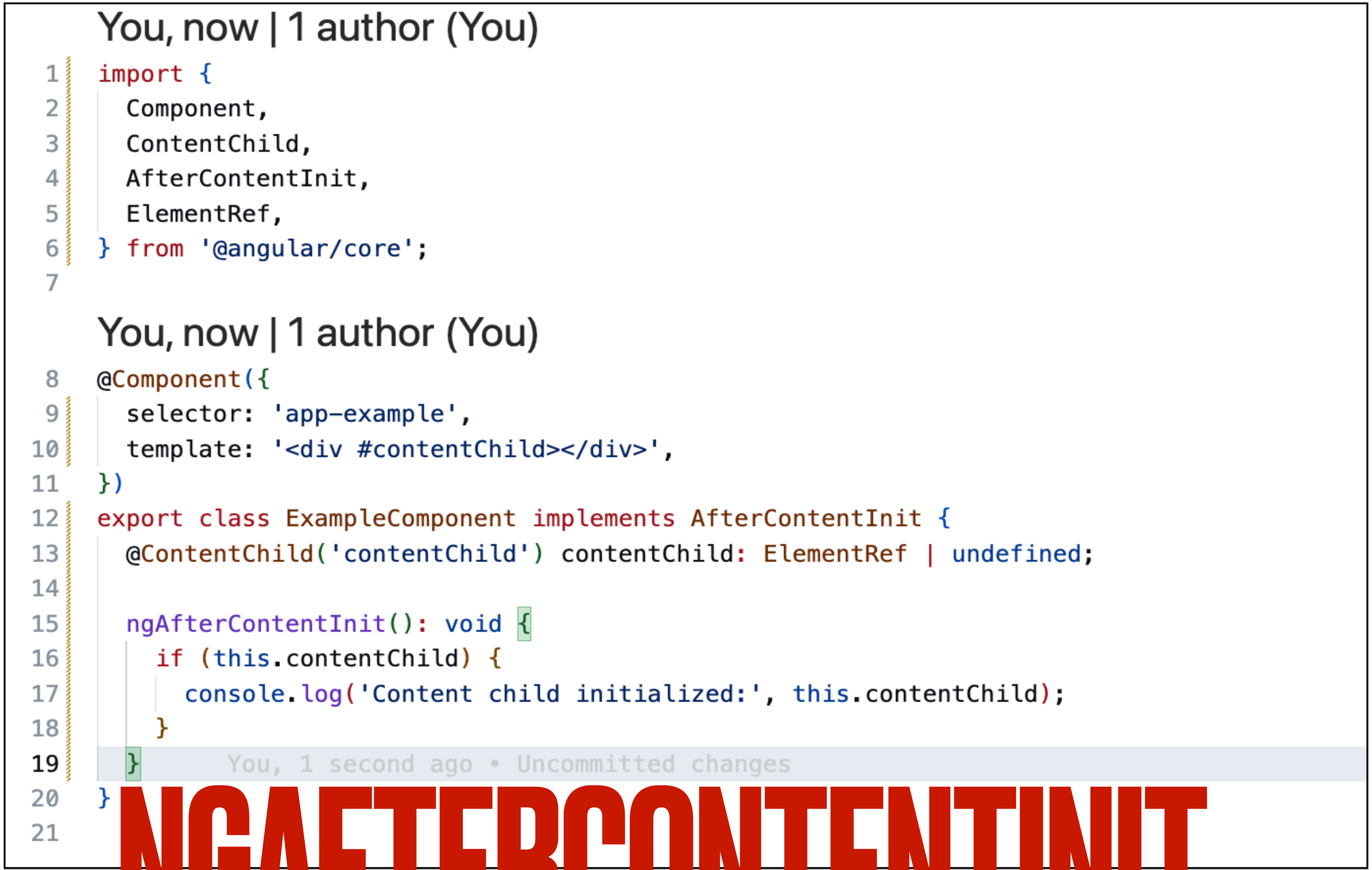
You, now | 1 author (You)

```
@Component({
  selector: 'app-example',
  template: '<p>{{ counter }}</p>',
})
export class ExampleComponent implements DoCheck {
  counter: number = 0;

  ngDoCheck(): void {
    if (this.counter > 10) {
      console.log('Counter is greater than 10!');
    }
  }
}
```

NGDOCHECK

- The ngDoCheck lifecycle hook in Angular provides a mechanism for custom change detection.
- This hook gives more fine-grained control over when change detection is triggered, which can be useful for optimising performance in specific scenarios.

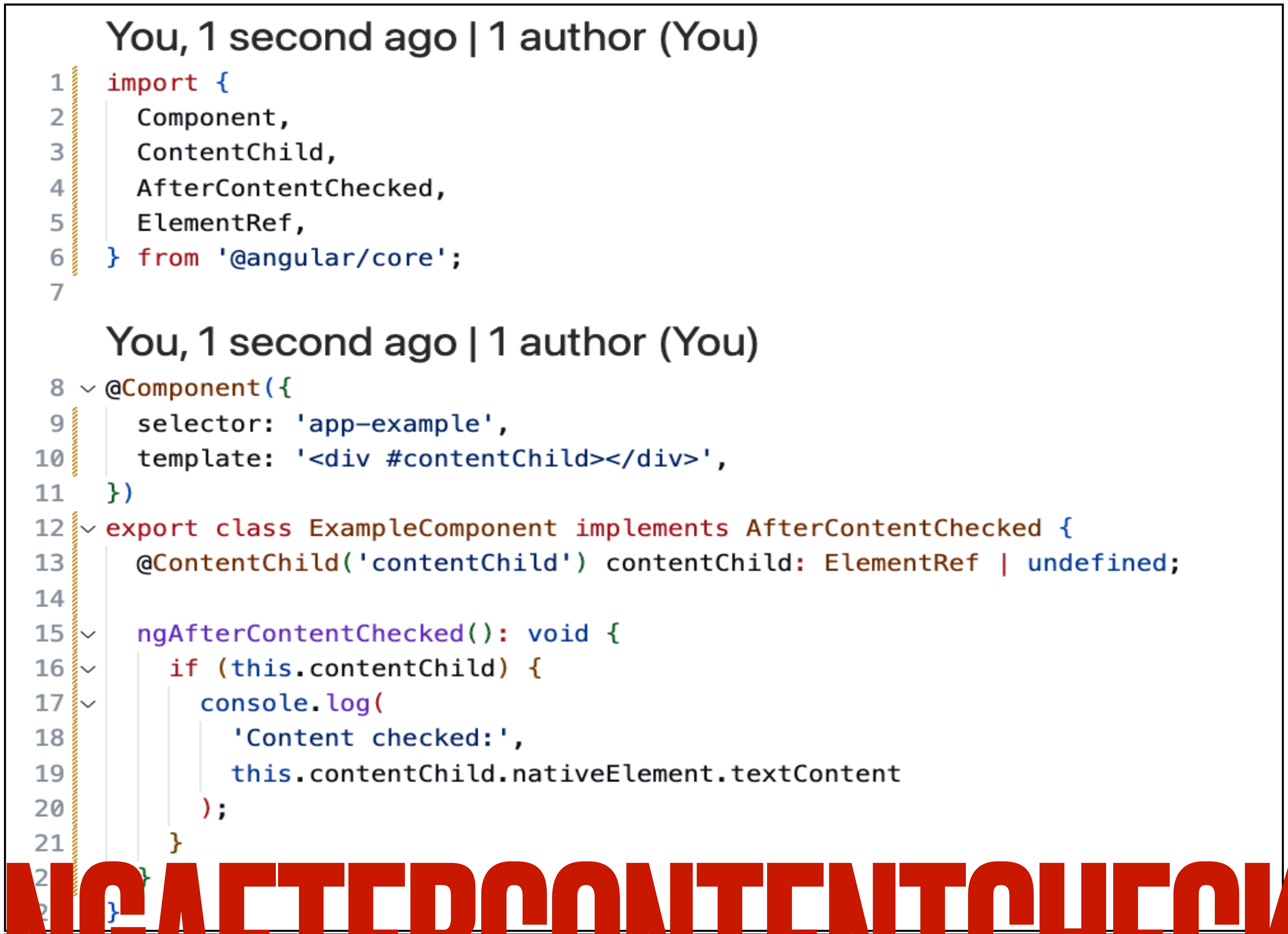
NGAFTERCONTENTINIT



```
You, now | 1 author (You)
import {
  Component,
  ContentChild,
  AfterContentInit,
  ElementRef,
} from '@angular/core';

You, now | 1 author (You)
@Component({
  selector: 'app-example',
  template: '<div #contentChild></div>',
})
export class ExampleComponent implements AfterContentInit {
  @ContentChild('contentChild') contentChild: ElementRef | undefined;

  ngAfterContentInit(): void {
    if (this.contentChild) {
      console.log('Content child initialized:', this.contentChild);
    }
  }        You, 1 second ago • Uncommitted changes
}
```

- The ngAfterContentInit lifecycle hook in Angular is called after the component's content has been projected into its view and the initialisation of the content is complete.
- It is a good place to perform tasks that rely on the presence and initialisation of content children.

NGAFTERCONTENTCHECKED



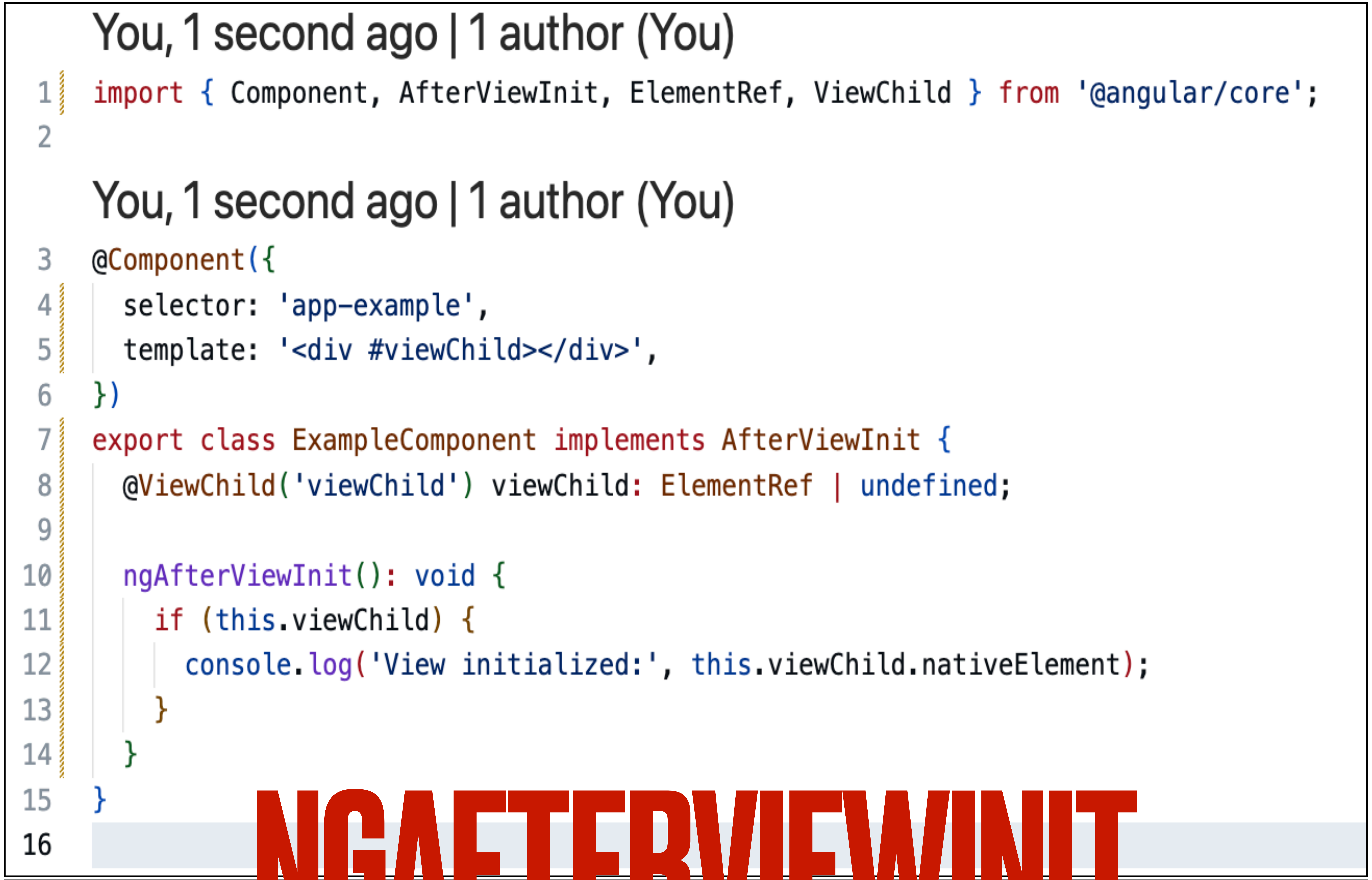
You, 1 second ago | 1 author (You)

```typescript
import {
  Component,
  ContentChild,
  AfterContentChecked,
  ElementRef,
} from '@angular/core';

```

You, 1 second ago | 1 author (You)

```typescript
@Component({
  selector: 'app-example',
  template: '<div #contentChild></div>',
})
export class ExampleComponent implements AfterContentChecked {
  @ContentChild('contentChild') contentChild: ElementRef | undefined;

  ngAfterContentChecked(): void {
    if (this.contentChild) {
      console.log(
        'Content checked:',
        this.contentChild.nativeElement.textContent
      );
    }
  }
}
```

- This hook is part of the Angular component lifecycle and provides a way to perform additional checks or tasks related to the component's content after each change detection cycle.
- One should avoid heavy computations or operations that could impact performance since this hook is called frequently.



You, 1 second ago | 1 author (You)

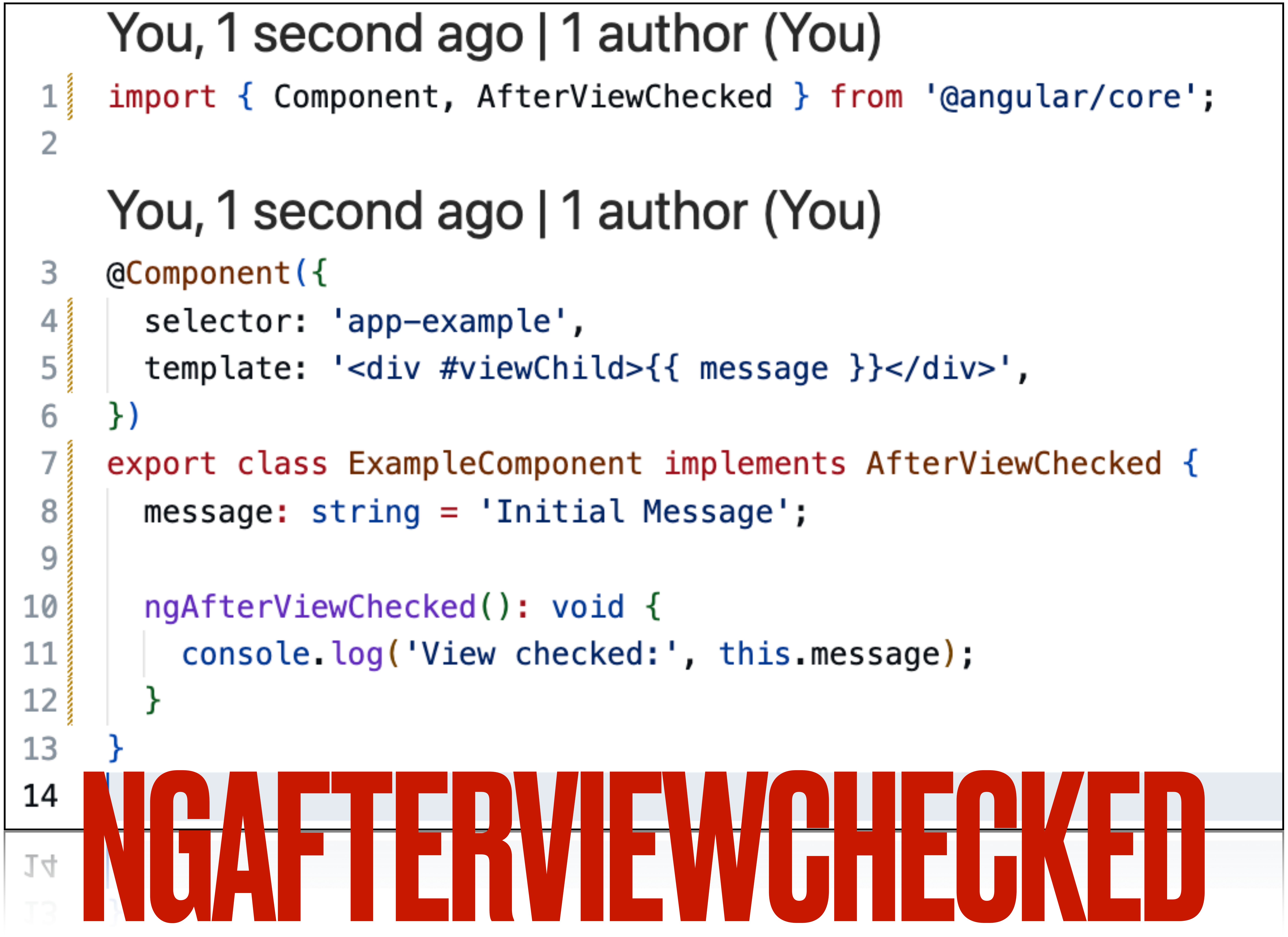
```
import { Component, AfterViewInit, ElementRef, ViewChild } from '@angular/core';

```

You, 1 second ago | 1 author (You)

```
@Component({
  selector: 'app-example',
  template: '<div #viewChild></div>',
})
export class ExampleComponent implements AfterViewInit {
  @ViewChild('viewChild') viewChild: ElementRef | undefined;

  ngAfterViewInit(): void {
    if (this.viewChild) {
      console.log('View initialized:', this.viewChild.nativeElement);
    }
  }
}

```

NGAFTERVIEWINIT

- The ngAfterViewInit lifecycle hook in Angular is called after the component's view, including its child views, has been initialised.
- This hook is part of the Angular component lifecycle and provides a way to perform initialisation or additional setup tasks that rely on the component's view being fully initialised and rendered.



You, 1 second ago | 1 author (You)

```typescript
import { Component, AfterViewChecked } from '@angular/core';

```

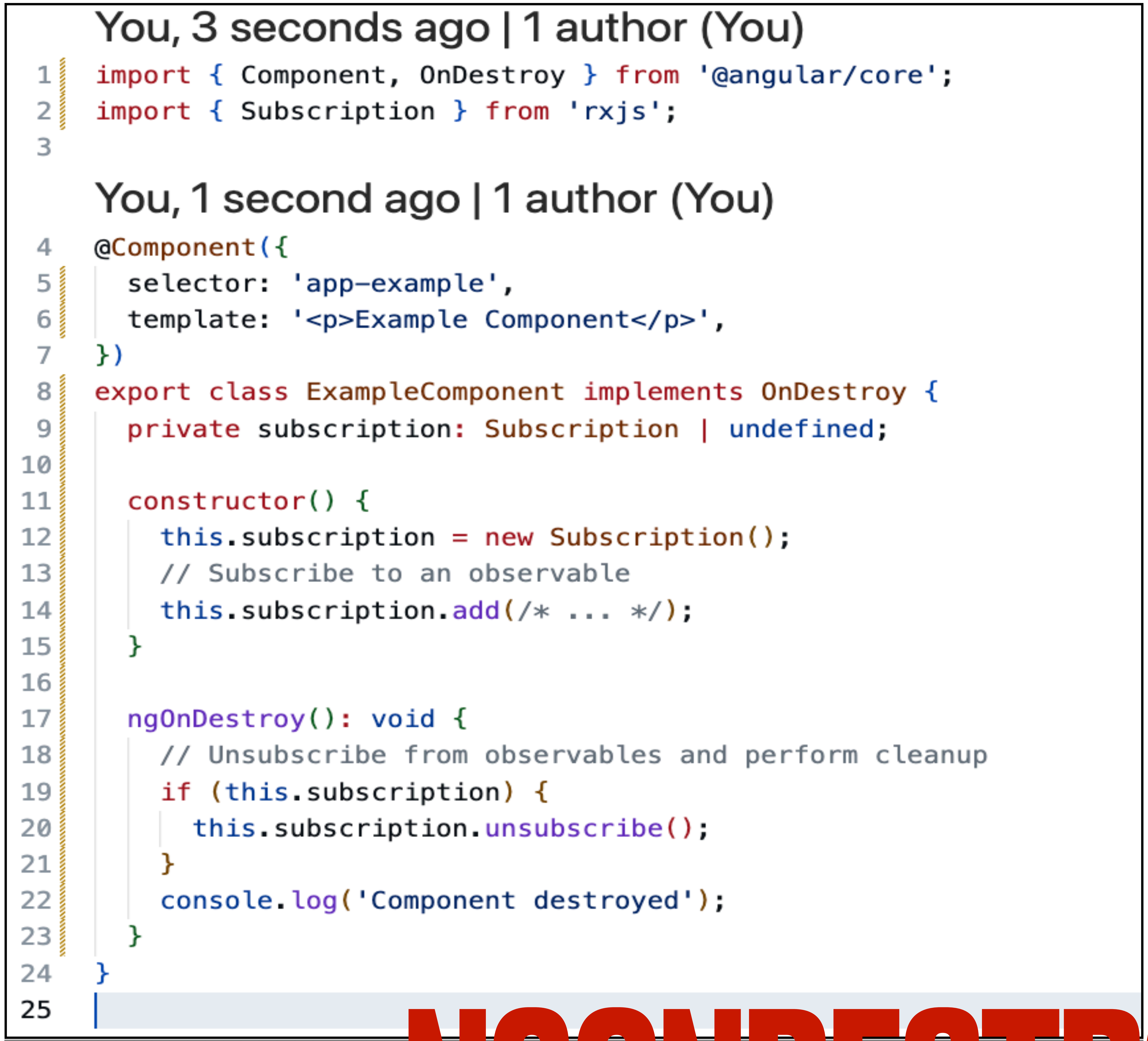
You, 1 second ago | 1 author (You)

```typescript
@Component({
  selector: 'app-example',
  template: '<div #viewChild>{{ message }}</div>',
})
export class ExampleComponent implements AfterViewChecked {
  message: string = 'Initial Message';

  ngAfterViewChecked(): void {
    console.log('View checked:', this.message);
  }
}

```

NGAFTERVIEWCHECKED

- The ngAfterViewChecked lifecycle hook in Angular is called after Angular has checked the component's view and its child views for changes.
- This hook is part of the Angular component lifecycle and provides a way to perform additional checks or tasks related to the component's view after each change detection cycle.

NGONDESTROY



You, 3 seconds ago | 1 author (You)

```
import { Component, OnDestroy } from '@angular/core';
import { Subscription } from 'rxjs';

```

You, 1 second ago | 1 author (You)

```
@Component({
  selector: 'app-example',
  template: '<p>Example Component</p>',
})
export class ExampleComponent implements OnDestroy {
  private subscription: Subscription | undefined;

  constructor() {
    this.subscription = new Subscription();
    // Subscribe to an observable
    this.subscription.add(/* ... */);
  }

  ngOnDestroy(): void {
    // Unsubscribe from observables and perform cleanup
    if (this.subscription) {
      this.subscription.unsubscribe();
    }
    console.log('Component destroyed');
  }
}
```

- The ngOnDestroy lifecycle hook in Angular is called just before a component is destroyed or removed from the DOM.
- It provides developers with an opportunity to perform cleanup tasks, unsubscribe from observables, and release any resources associated with the component to prevent memory leaks.

THANKS.