Save it

Like it

# ngOnChanges Vs ngDoCheck

@coder_aishya

## Triggering Mechanism

### NgOnChanges

Called only when an input property bound with @Input changes

### NgDoCheck

called during every change detection cycle, which allows for custom change detection logic.

## Purpose

### NgOnChanges

To perform actions based on changes to input properties.

### NgDoCheck

To implement custom change detection logic.

@coder_aishya

## Parameters

### NgOnChanges

Receives a SimpleChanges object containing previous and current values of input properties.

**SimpleChange is a simple class with 3 properties:**

**previousValue:any**

Previous value of the input property.

**currentValue:any**

Does not receive any parameters, allowing for more general custom change detection logic.

**FirstChange():boolean**

Boolean value, which tells us whether it was the first time the change has taken place

### NgDoCheck

Does not receive any parameters.

@coder_aishya

## Performance

### NgOnChanges

Generally **efficient** as it's only triggered for input property changes.

### NgDoCheck

Can be performance-intensive if not used carefully, as it's called frequently.

@coder_aishya

## When to Use Which

### NgOnChanges

Ideal for handling changes in input properties and performing actions based on those changes.

### NgDoCheck

Used when Angular's default change detection isn't sufficient and you need custom logic to detect changes. However, use it cautiously due to performance implications

## ngOnChange Example

childcomponent.ts

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

@Component({
  selector: 'app-child-on-changes',
  template: `<p>{{ name }}</p>`,
})
export class ChildOnChangesComponent implements OnChanges {
  @Input() name: string;

  ngOnChanges(changes: SimpleChanges): void {
    if (changes['name']) {
      const prevValue = changes['name'].previousValue;
      const currValue = changes['name'].currentValue;
      console.log(`name changed from ${prevValue} to ${currValue}`);
    }
  }
}
```

@coder_aishya

## ngOnChanges Example

**parentcomponent.ts**

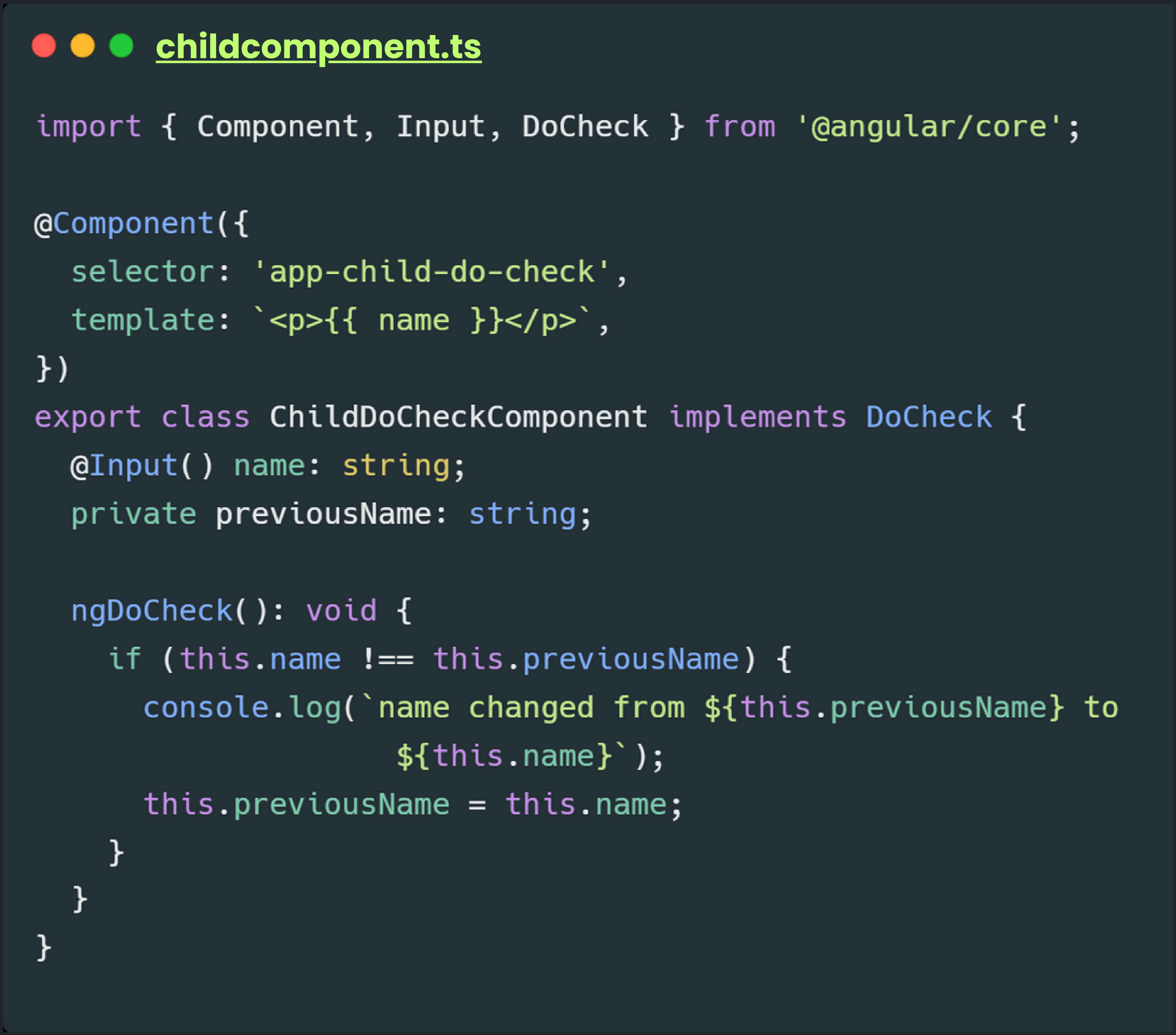
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-parent',
  template: `
    <button (click)="changeName( )">Change Name</button>
    <app-child-on-changes [name]="parentName"></app-child-on-changes>
  `,
})
export class ParentComponent {
  parentName: string = 'Initial Name';

  changeName( ): void {
    this.parentName = 'Updated Name';
  }
}
```

@coder_aishya

## ngDoCheck Example



childcomponent.ts

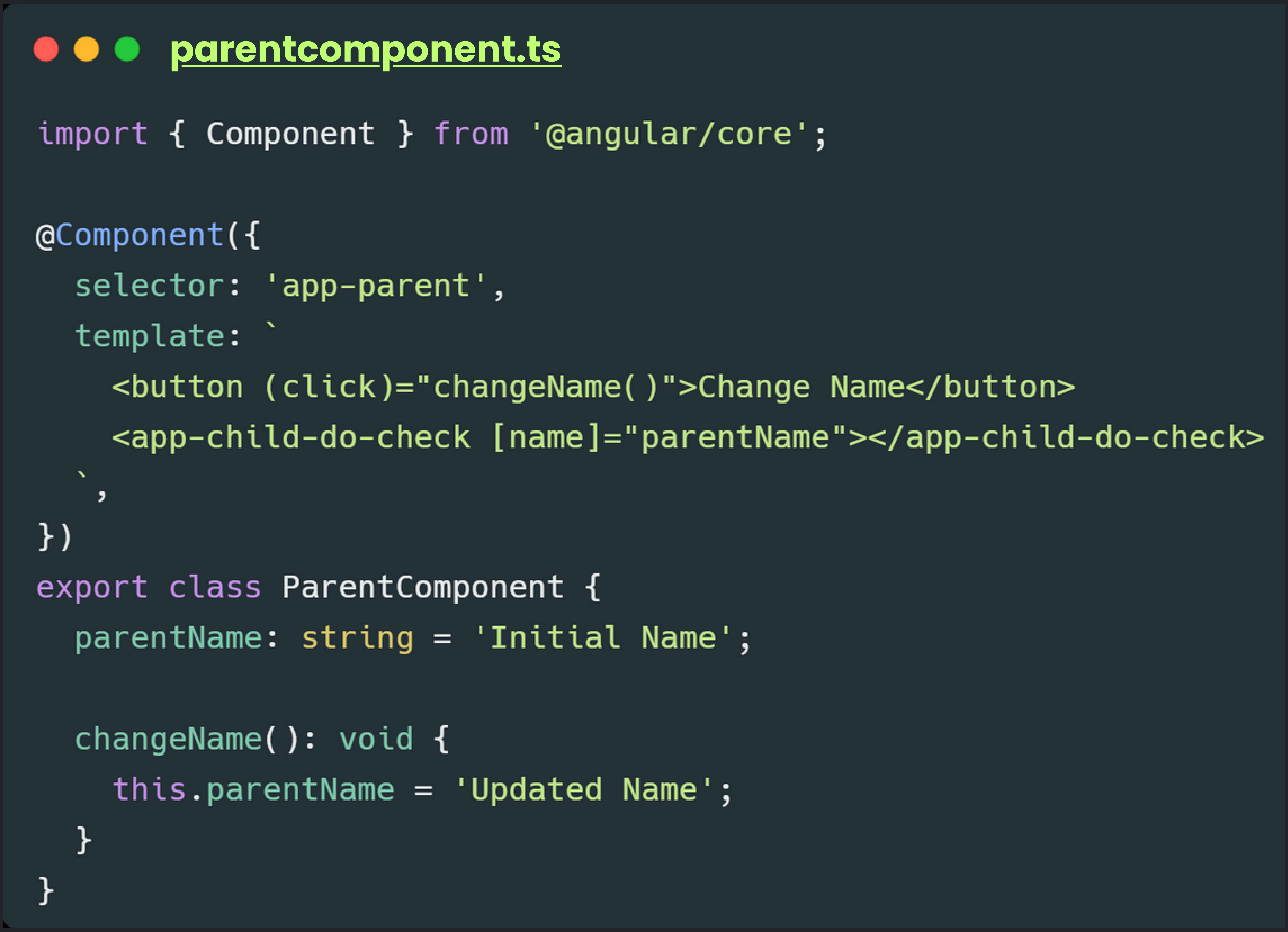
```typescript
import { Component, Input, DoCheck } from '@angular/core';

@Component({
  selector: 'app-child-do-check',
  template: `<p>{{ name }}</p>`,
})
export class ChildDoCheckComponent implements DoCheck {
  @Input() name: string;
  private previousName: string;

  ngDoCheck(): void {
    if (this.name !== this.previousName) {
      console.log(`name changed from ${this.previousName} to
              ${this.name}`);
      this.previousName = this.name;
    }
  }
}
```

@coder_aishya

## ngDoCheck Example



parentcomponent.ts

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

@Component({
  selector: 'app-parent',
  template: `
    <button (click)="changeName()">Change Name</button>
    <app-child-do-check [name]="parentName"></app-child-do-check>
  `,
})
export class ParentComponent {
  parentName: string = 'Initial Name';

  changeName(): void {
    this.parentName = 'Updated Name';
  }
}
```

## Follow me for more

aishwarya-dhuri

coder_aishya