What's the difference between ngOnInit and ngAfterViewInit of Angular2?

Asked 2 years, 9 months ago Active 1 year, 1 month ago Viewed 58k times

42

7

I can not understand what the difference between `ngOnInit` and `ngAfterViewInit` .

I found the only difference between them is `@ViewChild` . According to the following code, the `elementRef.nativeElement` in them are the same.

What scene should we use `ngAfterViewInit` ?

```
@Component({
  selector: 'my-child-view',
  template: `
  <div id="my-child-view-id">{{hero}}</div>
  `
})
export class ChildViewComponent {
  @Input() hero: string = 'Jack';
}

//////////////////////
@Component({
  selector: 'after-view',
  template: `
    <div id="after-view-id">-- child view begins --</div>
      <my-child-view [hero]="heroName"></my-child-view>
    <div>-- child view ends --</div>`
   + `
    <p *ngIf="comment" class="comment">
      {{comment}}
    </p>
  `
})
export class AfterViewComponent implements AfterViewInit, OnInit {
  private prevHero = '';
  public heroName = 'Tom';
  public comment = '';

  // Query for a VIEW child of type `ChildViewComponent`
```

```
    }

    ngOnInit(){
      console.log('OnInit');
      console.log(this.elementRef.nativeElement.querySelector('#my-child-view-id'));
      console.log(this.elementRef.nativeElement.querySelector('#after-view-id'));
      console.log(this.viewChild);
      console.log(this.elementRef.nativeElement.querySelector('p'));
    }

    ngAfterViewInit() {
      console.log('AfterViewInit');
      console.log(this.elementRef.nativeElement.querySelector('#my-child-view-id'));
      console.log(this.elementRef.nativeElement.querySelector('#after-view-id'));
      console.log(this.viewChild);
      console.log(this.elementRef.nativeElement.querySelector('p'));
    }
  }
```

angular ngoninit

edited Jul 16 '18 at 13:49
Sujatha Girijala
638 6 19

asked Nov 26 '16 at 10:27
Zhiyuan Sun
251 1 3 5

3 Answers

50

`ngOnInit()` is called after `ngOnChanges()` was called the first time. `ngOnChanges()` is called every time inputs are updated by change detection.

`ngAfterViewInit()` is called after the view is initially rendered. This is why `@ViewChild()` depends on it. You can't access view members before they are rendered.

edited May 12 '17 at 8:15

answered Nov 26 '16 at 12:34
Günter Zöchbauer

2 When it's added to the DOM. If you set `display: hidden` it's till rendered, but not visible on the screen. But if you investigate the DOM using the browsers devtools, you'll be able to see the markup. – Günter Zöchbauer Jun 22 '17 at 19:12

3 "*you can't access view members before they are rendered*" - So how do you explain that the `ViewChild` (vc) is available on `onNgInit` ? plnkr.co/edit/AzhRe6bjnuPLKJWEJGwp?p=preview , Can you please explain ? – Royi Namir Jun 27 '17 at 6:45

1 @Royi I can't open your Plunker on my phone and it will tke a few days until I'm back to my computer. Statically added elements are already available in `ngOnInit` . If you have content that is rendered for examply by `*ngFor` from data passes to an `@Input` , this content won't yet be available in `ngOnInit` – Günter Zöchbauer Jun 27 '17 at 18:14

2 Thank you very much for the response. That's exactly the scenario. So I guess this is it. i.imgur.com/Vbajl4F.jpg . Enjoy your vacation. – Royi Namir Jun 27 '17 at 18:29

1 see also angular.io/guide/lifecycle-hooks – peter70 Nov 15 '17 at 8:39

17

`ngOnInit()` is called right after the directive's data-bound properties have been checked for the first time, and before any of its children have been checked. It is invoked only once when the directive is instantiated.

`ngAfterViewInit()` is called after a component's view, and its children's views, are created. Its a lifecycle hook that is called after a component's view has been fully initialized.

answered Nov 26 '16 at 12:34
Vishal Gulati
4,840 8 48 73

1

Content is what is passed as children. View is the template of the current component.

The view is initialized before the content and `ngAfterViewInit()` is therefore called before `ngAfterContentInit()` .

** `ngAfterViewInit()` is called when the bindings of the children directives (or components) have been checked for the first time. Hence its perfect for accessing and manipulating DOM with Angular 2 components. As @Günter Zöchbauer mentioned before is correct `@ViewChild()` hence runs fine inside it.

Example:

```
})
export class WidgetThree{
    @ViewChild('input1') input1;

    constructor(private renderer:Renderer){}

    ngAfterViewInit(){
        this.renderer.invokeElementMethod(
            this.input1.nativeElement,
            'focus',
            []
        )
    }
}
```

answered May 12 '17 at 8:15
STEEL
3,014 4 36 64

I think you are wrong here. ngAfterViewInit() executes only after ngAfterContentChecked() and ngAfterContentChecked() executes only after the ngAfterContentInit() and every subsequent ngDoCheck(). Please refer angular lifecycle hooks details for more details angular.io/guide/lifecycle-hooks – Suneet Bansal Feb 16 at 8:49