

Using @ViewChild in Angular



Nishu Goel Sep 8, 2018 · 2 min read



A ViewChild is a component, directive, or element as a part of a template. If we want to access a child component, directive, DOM element inside the parent component, we use the decorator @ViewChild() in Angular.

To understand this, suppose we have two components, a child and a parent one. Since the child component can be located inside the parent component, it can be accessed as @ViewChild.

```
1 export class AppComponent implements OnInit, AfterViewInit {
2   message: any;
3   @ViewChild(ChildComponent) chViewChild: ChildComponent;
4
5   ngAfterViewInit() {
6     console.log(this.chViewChild);
7   }
8
9   ngOnInit() {
10    this.message = 'Hello World !';
11  }
12 }
```

app.component.ts hosted with ❤ by GitHub

[view raw](#)

In the above code, we imported @ViewChild decorator and then imported the lifecycle hook AfterViewInit and implemented it. Now to change the

Related



Angular Introduction
Angular is a JavaScript...



Queuing CSS animations in
Angular with RXJS
CSS Animations

No provider for HttpClient!

Add HttpClientModule to imports in...

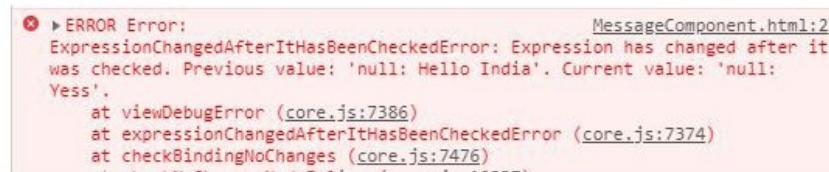


Move your angular cli app to
Nx
Help me improve these...

the `message` property value, we can make use of the `ViewChild` we just create like:

```
ngAfterViewInit(){
  this.chviewChild.message = 'Changed value of View Child'
}
```

This will output the value **Changed value of View Child**, but will throw an error saying “**Expression has changed after it was last checked**”.



The screenshot shows a red error message in the console. The message is: "ERROR Error: ExpressionChangedAfterItHasBeenCheckedError: Expression has changed after it was checked. Previous value: 'null: Hello India'. Current value: 'null: Yess'." The error is located in `MessageComponent.html:2`. The stack trace shows the error was thrown at `viewDebugError (core.js:7386)`, `expressionChangedAfterItHasBeenCheckedError (core.js:7374)`, and `checkBindingNoChanges (core.js:7476)`.

To handle this error, we use Change detection using ***ChangeDetectorRef***

```
1 constructor(private cd: ChangeDetectorRef) {}
2
3 ngAfterViewInit() {
4   console.log(this.chviewChild);
5   this.chviewChild.message = 'Changed value of View Child';
6   this.cd.detectChanges();
7 }
```

handle.ts hosted with ❤ by GitHub

[view raw](#)

Here, we have imported ***ChangeDetectorRef*** from ‘@angular/core’ and injected it to the constructor. We have also called the method ***detectChanges()***.

This was for one child element located inside the component template, but what in the case when we have multiple references to the child element?

That is when @ViewChild comes into play.

```
1  export class AppComponent implements OnInit {
2      messages: any;
3      ngOnInit() {
4          this.messages = this.getMessage();
5      }
6      getMessage() {
7          return [
8              'Hi',
9              'Hello',
10             'How are you?'
11         ];
12     }
13 }
```

compp.ts hosted with ❤ by GitHub

[view raw](#)

Modifying the template,

```
1  @Component({
2      selector: 'app-root',
3      template: `
4      <div>
5      <app-child *ngFor="let i of messages" [message]='i'></app-child>
6      </div>`
7  })
```

com.ts hosted with ❤ by GitHub

[view raw](#)

Finally, marking the reference of ViewChildren with type QueryList,

```
1  @ViewChild(ChildComponent) chviewChildren: QueryList<ChildComponent>;
2  ngAfterViewInit() {
3      console.log(this.chviewChildren);
4  }
```

comp1.ts hosted with ❤ by GitHub

[view raw](#)

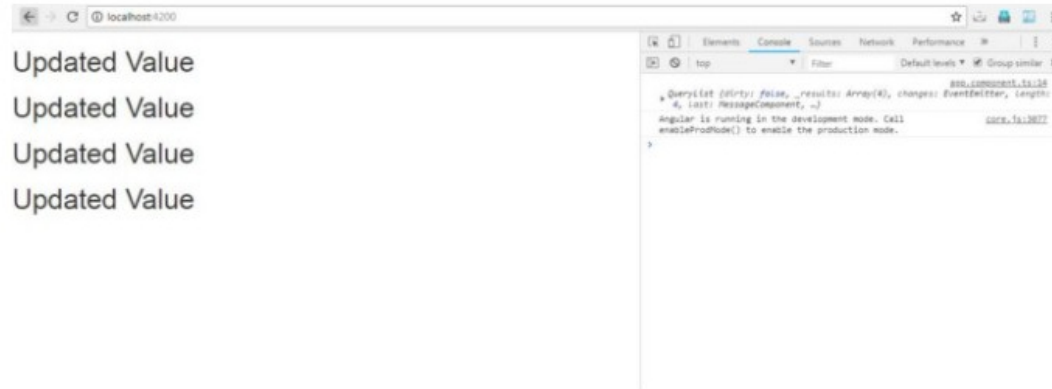
Now, we can update the value of the message property using Viewchildren inside the ngAfterViewInit() life cycle hook like this:

```
this.chviewChildren.forEach((item) => {item.message = 'Updated Value';});
```

Let us look at the final code now:

```
1  import { Component, Input, OnInit, ViewChild, AfterViewInit, ChangeDetectorRef, ViewChildren } from '@angular/core';
2  import { ChildComponent } from './child.component';
3  import { QueryList } from '@angular/core';
4
5  @Component({
6    selector: 'app-root',
7    template: `
8      <app-child *ngFor = 'let i of message' [message] = 'i'></app-child>
9    `
10  })
11  export class AppComponent implements OnInit, AfterViewInit{
12    constructor(private cd: ChangeDetectorRef){}
13    ngAfterViewInit() {
14      console.log(this.chviewChildren);
15      this.chviewChildren.forEach((item) =>{item.message = "Updated Value"; });
16      this.cd.detectChanges();
17    }
18    message: any;
19    @ViewChildren(ChildComponent) chviewChildren: QueryList<ChildComponent>;
20
21    ngOnInit(){
22      this.message = this.getMessage();
23    }
24    getMessage(){
25      return[
26        'Hi',
27        'Hello',
28        'How are you?'
29      ];
30    }
31  }
```

This gives us the output as:



JavaScript Angular Angular Cli Angular2



78



More from Nishu Goel

Follow



Engineer @ The DataWorks, Google GDE, Microsoft MVP, @angular ❤, Author —
<https://amzn.to/2Zd9ADd>

Sep 5, 2018

Reusing components with Content Projection in Angular

Want to change the content inside your component and re-use them? Well, That is what we use Content Projection for, in Angular.

Let us start by defining what Content Projection is. A feature with the help of which we can have a component with a kind of visual wrapper, and...

[Read more · 2 min read](#)



Share your ideas with millions of readers.

Write on Medium

Sep 3, 2018

Beginning with Unit Testing in Angular

Before going on to test our angular code, we need to understand what is a unit test.

A Unit test is defined to be a test on a single unit, where unit can be a single class or a group of related classes.

There are 2 types of tests we often...

[Read more · 2 min read](#)



Aug 25, 2018

Binding Array to a Table

In this article, we will learn how to bind an array to a HTML table. This means that we want the elements in our array and their attributes to be displayed on the view.

For this, we will mainly use some hard-coded data and structure directives like `*ngIf` and `*ngFor`.

...

[Read more · 1 min read](#)



Aug 24, 2018

What is a Shared Module?

In Angular, we can create different feature modules for every feature. Each one of these features would require importing the `CommonModule` and other imports too. Do we really want to repeat all this for every feature module? Or do we have an alternative?

Yes, We can define a Shared Module...

[Read more · 2 min read](#)

Aug 23, 2018

Modules in Angular

What is an Angular Module?

A class with an `@NgModule` decorator which organizes the pieces of an application and extend our application with capabilities from the external libraries.

An Angular Module:

- Declares
- Imports
- Exports
- Bootstraps
- Provides

To start with, every angular application has a root application module

To start with, every angular application has a root application module, called AppModule and a root application component, called AppComponent. The AppModule...

[Read more](#) · 2 min read

