@ViewChild Decorator

- @View Child decorator helps in accessing properties/ methods of a child component, directive, or DOM element.

- ViewChild decorator returns the first element or directive matching the Selector from the DOM.

- ViewChild decorator creates an Instance of a component/ directive class in the parent component to access the properties or methods of that component/directive.

— @iam_frontendev

Accessing a child component using @ViewChild

- @ViewChild allows the parent component to access the properties and methods of the child component.

- @ViewChild decorator creates an instance of a child components in the parent component in the parent component and selector of child component should be used in the parent component's template.

STEP1 - Create a Timer Component in app folder

```
ng g c Timer
```

Step 2 - Add the below code in app.module.ts

```
...
@NgModule({
   declarations: [
        AppComponent,
        TimerComponent ],
  imports : [ BrowserModule ],
  providers: [ ],
  bootstrap : [AppComponent]
})
```

Added → the timer component.

STEP 3 — Add the following code to timer.component.ts

```
...

export class TimerComponent {
constructor () { }
flag = false;
 count = 1;
 begin () {
      this.flag = true;
      ~~const~~ const start = setInterval (()=>{
    if (this.flag === false) {
     clearInterval (start);
    }
     this.count +=1;
     }, 1000);
   }

 end () {
    this.flag = false;
  }
}
```

-@iam_frontendev

STEP-4 - Now instantiate timer.component.ts

```
   ...
   import {TimerComponent} from 'timer.
                                 component.ts
@Component({
     ...
  })
export class AppComponent {

@ViewChild (TimerComponent)timerComponent
          : TimerComponent ;

  StartTimer ( ) {
      this.timerComponent.begin( );
    }

   stopTimer ( ) {
      this.timer Component.end ( );
    }
  }
```

Create instance of timer component

App component now access the properties/ methods of Timer Component.

STEP-5 Add the following code in timer.component.html

```
<p> {{count}} </p>
```

STEP-6 Add the following code in app.component.html.

```
<h3> Accessing component using
        @ViewChild </h3>
 Timer Example :
 <button type="button" (click)="startTimer()">
       Begin </button>
<button type="button" (click)="stopTimer()">
       End </button>

<app-timer> </app-timer>
```

-@iam_frontendev

ACCESSING A DIRECTIVE USING @View Child

- @ViewChild creates an instance of a directive within a component and in this way the component can access the methods of the directive class.

STEP-1 : Add the following in app Module

// app. module.ts

```
...
@NgModule({
  declarations: [
    AppComponent, ColorDirective ],
  imports: [ BrowserModule ],
  providers: [ ],
  bootstrap: [AppComponent ]
})
export class AppModule { }
```

Create a directive and add here.

—@iam_frontendev

STEP-2 :- Add the below in the color.directive .ts file.

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

@Directive ({
    Selector :'[appColor]'
    })
export class ColorDirective implements
                              AfterViewInit {
constructor (private elementRef : ElementRef)
                                         { }
ngAfterViewInit () {
 this.elementRef.nativeElement.style.color=
                  'green';

 modify ( color: string) {
  this.elementRef.nativeElement.style.color=
          color;
  }
}
```

used to execute statement after component fully initialized.

—@iam_frontdev

● STEP 3 :- Add the code in app.component .ts file and access the directive methods.

—@iam-frontendev

```
...
import {ColorDirective} from './color.directive';

@Component({
  ...
 })
 export class AppComponent {
 @ViewChild(colorDirective) colorDirective!:
             colorDirective;

 modifyColor(color: string) {
   this.colorDirective.modify(color);
  }
 }
```

↓ Color directive class can be now accessed from AppComponent.

@ViewChild decorator creates an instance of a color directive in App Component.

STE4: Add the below code in app.component.html

```
<h3> Accessing Directive using @ViewChild </h3>
 <br/>
<div appColor> Modify Color </div>
<br/>
<div>
   Modify Color :
  <input type="radio" name="color"
   (click)= "modifyColor('blue')" /> Blue
 <input type= "radio' name= "color"
  (click) = "modify Color ('yellow')"/> Yellow
 <input type = "radio" name = "color"
  (click) = "modifyColor ('cyan')"/> Cyan
</div>
```

-@iam_frontendev.

OUTPUT :

Accessing directive using @ViewChild.

$\boxed{\text{Modify Color}}$ → The text color will change based on radio button.

Modify Color: O Blue O Yellow O Cyan

ACCESSING A NATIVE ELEMENT USING @View Child

→ @ViewChild requires the template variable name to be passed as its argument and allows the component to change the appearance or behavior of a given template element.

STEP 1 :- // app.component.html

```
<h3> Accessing Template variable using
       @ViewChild </h3>
 <div>
     Employee Name :
 <input type = "text" #empName>
  <br/> Employee Number :
  <input type="number" #empnumber>
</div>
```

Two Input box

Two input boxes in the template with 'empname' and 'empnumber' as their respective template reference variable.

-@iam_frontender

STEP 2 : // app.component.ts

```
import {component, ViewChild, AfterViewInit,
        ElementRef } from '@angular/core';
...
export class AppComponent implements
              AfterViewInit {
@ViewChild ('empname') empName: ElementRef;
@ViewChild ('empnumber') empNumber: ElementRef;
```

↳ ElementRef needs to be instantiated using @ViewChild

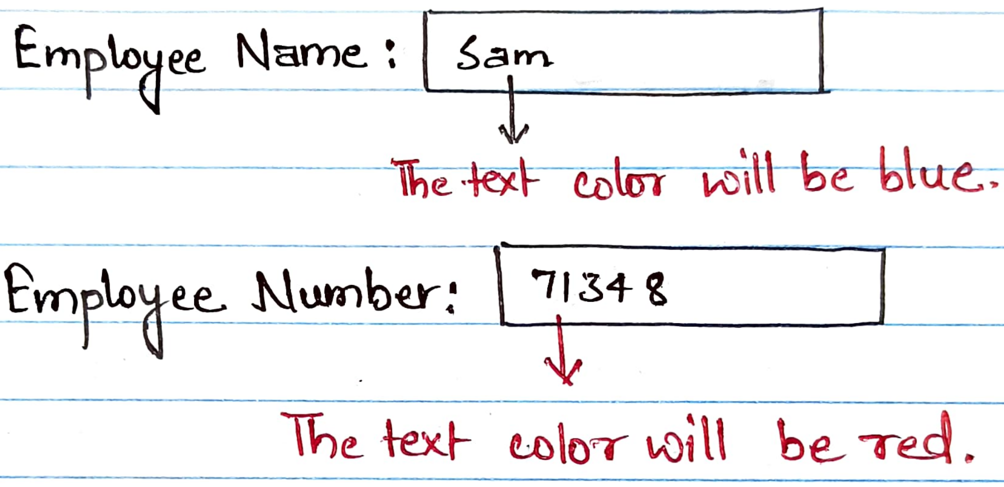
```
ngAfterViewInit() {
this.empName.nativeElement.style.color='blue';
this.empNumber.nativeElement.style.color='red';
 }
}
```

→ AfterViewInit hook is used to execute statements after a component view is fully initialized.

-@iam_frontendev

OUTPUT :-

Accessing Template variable using @View Child.



Employee Name : Sam

The text color will be blue.

Employee Number: 71348

The text color will be red.

-@iam_frontender