ViewChild:

- In Angular, ViewChild is a decorator that allows you to obtain a reference to the following things:
    1. Child component
    2. Directive or element from a parent component.
- It's a way to interact with elements or components within a template.

- First Learn how we work with ElementRef

ElementRef

—---

- ElementRef represents a specific normal tag (not component) in the template / content.
- You can create the reference name by writing "#" and followed by reference name in the template.
- You can access the element by specifying its reference name in the @ViewChild or @ContentChild decorator

Syntax:

I. Create the reference name:

- <tag #referenceName></tag>

II. Access the element

```
class parentcomponent
{
@ViewChild("referencename") propertyname: ElementRef;
@ContentChild("referencename") propertyname: ElementRef;
…
}
```

Example :

Step1: Create the parent component

- Inside the parent.component.html the following code

```
<div class="container">
    <div class="row">
        <div>
            <h2 #id1>UserName:{{uname}}</h2>
            <input type="button" value="Click Me" (click)="onClickMe()"/>
        </div>
    </div>
    <div class="row">

    </div>
</div>
```

- Inside the parent.component.ts the following code

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

@Component({
  selector: 'app-parent',
  templateUrl: './parent.component.html',
  styleUrls: ['./parent.component.css']
})
export class ParentComponent{
  uname:string="Imtiaz";
  @ViewChild("id1")id1:ElementRef={} as ElementRef;


  onClickMe(){
     this.id1.nativeElement.innerHTML="Mohammed Imtiaz";
     this.id1.nativeElement.style.color='green';
  }
}
```

Another way:
Step1:

- Create parent and child component

1. app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {


}
```

2. App.component.html

```
<div class="container">
    <!-- <app-addproduct></app-addproduct>
     -->
     <!-- <app-addanddelete></app-addanddelete>
      -->
      <app-parent></app-parent>
</div>
```

3. parent.component.html

```
<div class="container">
    <div class="row">
        <!-- <div class="box"> -->
            <div #box>
            <h2>Welcome to Parent Component</h2>
        </div>

    </div>
    <div class="row">
        <div class="col-md-6">
```

```
            <button class="btn btn-success mt-2 ml-2" (click)="accessProperty()">Access Child Property</button>
            <button class="btn btn-warning mt-2 ml-2" (click)="accessMethod()">Access Child Method</button>
        </div>
    </div>
    <div class="row">

        <app-child>
            <p #p1>This is child content!</p>
        </app-child>
    </div>
</div>
```

4. parent.component.ts

```
import { AfterViewInit, Component, ElementRef, ViewChild,Renderer2 } from '@angular/core';
import { ChildComponent } from '../child/child.component';

@Component({
  selector: 'app-parent',
  templateUrl: './parent.component.html',
  styleUrls: ['./parent.component.css']
})
export class ParentComponent implements AfterViewInit {
   @ViewChild('box')box:ElementRef={} as ElementRef;
   @ViewChild(ChildComponent)child:ChildComponent={} as ChildComponent;

   constructor(private renderer:Renderer2){

  }
   ngAfterViewInit(): void {
       /*this.box.nativeElement.style.color='red';
       this.box.nativeElement.style.backgroundColor='black';
       this.box.nativeElement.innerHTML='WELCOME TO MY HOME PAGE';
       this.box.nativeElement.style.fontSize='30px';
       this.box.nativeElement.style.textalign='center';*/
       this.renderer.setStyle(this.box.nativeElement,"backgroundColor","black");
       this.renderer.setStyle(this.box.nativeElement,"color","white");
       this.renderer.setStyle(this.box.nativeElement,"textAlign","center");
       this.renderer.setAttribute(this.box.nativeElement,'class','boxClass');
       console.log(this.box.nativeElement)

    }
    accessProperty():void{
      this.child.uname='Change Child Data By Parent';
    }
    accessMethod():void{
      this.child.clickMe();

    }
}
```

5. Child.component.html

```
<div class="container" appTestdirective>
    <h2>UserName:{{uname}}</h2>
    <button (click)="clickMe()">Click Me</button>
    <hr/>
    <ng-content></ng-content>
    <hr/>

</div>
```

6. Child.component.ts

```
import { Component, ElementRef, ContentChild, Renderer2, OnInit,AfterContentInit
} from '@angular/core';

@Component({
  selector: 'app-child',
  templateUrl: './child.component.html',
  styleUrls: ['./child.component.css']
})
export class ChildComponent implements OnInit,AfterContentInit {
 uname:string='From Child';
@ContentChild('p1') child:ElementRef={} as ElementRef;

constructor(private renderer:Renderer2){

}
ngOnInit(): void {

}
clickMe():void{
  alert(this.uname)
 }

 ngAfterContentInit(): void {
     this.child.nativeElement.style.color='red';
 }
}
```

Css

1. Parent.component.css

```
.container {
    width:700px;
    height:700px;
    background-color: antiquewhite;
    border:1px solid black;
}

.boxClass{
    width:690px;
    height:100px;
    background-color: brown;
    color:white;
    margin-left:10px;
    padding-left:20px;
}
```

2. Child.component.css

```
.container{
    width:560px;
    height:200px;
    border:1px solid red;
    margin-top:20px;
    margin-left:50px;
}
```

2. Working with directives
> create the directive
> ng g d testdirective

```
import { Directive, ElementRef, Renderer2 } from '@angular/core';

@Directive({
  selector: '[appTestdirective]'
})
export class TestdirectiveDirective {

  constructor(private e1:ElementRef,private rendere:Renderer2) {
        this.rendere.setStyle(this.e1.nativeElement,"backgroundColor","yellow");
   }


}
```

use the directive inside the child component

```
<div class="container" appTestdirective>
    <h2>UserName:{{uname}}</h2>
    <button (click)="clickMe()">Click Me</button>
    <hr/>
    <ng-content></ng-content>
    <hr/>

</div>
```