**Using ViewChild with Directives**

ViewChild makes it possible to access directives.

Let’s say you have a SharkDirective. This directive will look for elements with the attribute appShark and prepend the text in the element with the word "Shark".

Ideally, you will use @angular/cli to generate your directive:

1. ng generate directive shark --skip-tests

Copy

This command will create a shark.directive.ts file. And adds the directive to app.module.ts:

app.module.ts

import { SharkDirective } from './shark.directive';

...

@NgModule({

declarations: [

AppComponent,

SharkDirective

],

...

})

Copy

Then, use ElementRef and Renderer2 to rewrite the text. Replace the contents of shark.directive.ts with the following:

shark.directive.ts

import {

Directive,

ElementRef,

Renderer2

} from '@angular/core';

@Directive(

{ selector: '[appShark]' }

)

export class SharkDirective {

creature = 'Dolphin';

constructor(elem: ElementRef, renderer: Renderer2) {

let shark = renderer.createText('Shark ');

renderer.appendChild(elem.nativeElement, shark);

}

}

Copy

Next, add an appShark attribute to a span containing text in the component template. Replace the contents of app.component.html with the following:

app.component.html

<span appShark>Fin!</span>

Copy

When viewing the application in a browser, it will render the word "Shark" before the contents of the element:

Output

Shark Fin!

Now, you can also access the creature instance variable of SharkDirective and set an extraCreature instance variable with its value. Replace the contents of app.component.ts with the following:

app.component.ts

import {

Component,

ViewChild,

AfterViewInit

} from '@angular/core';

import { SharkDirective } from './shark.directive';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent implements AfterViewInit {

extraCreature!: string;

@ViewChild(SharkDirective)

set appShark(directive: SharkDirective) {

this.extraCreature = directive.creature;

};

ngAfterViewInit() {

console.log(this.extraCreature); // Dolphin

}

}

Copy

This code used a setter to set the extraCreature variable. Notice that it waits for the AfterViewInit [lifecycle hook](https://www.digitalocean.com/community/tutorials/angular-lifecycle-hooks) to access the variable, as this is when child components and directives become available.

When viewing the application in a browser, you will still see the "Shark Fin!" message. However, in the console log, it will display:

Output

Dolphin

The parent component was able to access the value from the directive.

**Using ViewChild with DOM Elements**

ViewChild makes it possible to access native DOM elements that have a template reference variable.

Let’s say you have an <input> in the template with the #someInput reference variable. Replace the contents of app.component.html with the following:

app.component.html

<input #someInput placeholder="Your favorite sea creature">

Copy

Now, you can access the <input> with ViewChild and set the value. Replace the contents of app.component.ts with the following:

app.component.ts

import {

Component,

ViewChild,

AfterViewInit,

ElementRef

} from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent implements AfterViewInit {

@ViewChild('someInput') someInput!: ElementRef;

ngAfterViewInit() {

this.someInput.nativeElement.value = 'Whale!';

}

}

Copy

When ngAfterViewInit fires the value of the <input> will be set to:

Output

Whale!

The parent component was able to set the value of the child DOM Element.

**Using ViewChild with Child Components**

ViewChild makes it possible to access a child component and call methods or access instance variables that are available to the child.

Let’s say you have a PupComponent.

Ideally, you will use @angular/cli to generate your component:

1. ng generate component pup --flat --skip-tests

Copy

This command will create pup.component.ts, pup.component.css, and pup.component.html files. And adds the component to app.module.ts:

app.module.ts

import { PupComponent } from './pup.component';

...

@NgModule({

declarations: [

AppComponent,

PupComponent

],

...

})

Copy

Then, add a whoAmI method to PupComponent which returns a message:

pup.component.ts

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-pup',

templateUrl: './pup.component.html',

styleUrs: ['./pup/component.css']

})

export class PupComponent implements OnInit {

constructor() { }

whoAmI() {

return 'I am a pup component!';

}

ngOnInit(): void {

}

}

Copy

Next, reference the child component in the app template. Replace the contents of app.component.html with the following:

app.component.html

<app-pup>pup works!</app-pup>

Copy

Now, you can call the whoAmI method from within the parent component class with ViewChild. Replace the contents of app.component.ts with the following:

app.component.ts

import {

Component,

ViewChild,

AfterViewInit

} from '@angular/core';

import { PupComponent } from './pup.component';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css'],

})

export class AppComponent implements AfterViewInit {

@ViewChild(PupComponent) pup!: PupComponent;

ngAfterViewInit() {

console.log(this.pup.whoAmI()); // I am a pup component!

}

}

Copy

When viewing the application in a browser, the console log will display:

Output

I am a pup component!

The parent component was able to call the child component’s whoAmI method.

**Conclusion**

In this tutorial, you used ViewChild to access a directive, child component, and a DOM element from a parent component class.

If the reference changes to a new element dynamically, ViewChild will automatically update its reference.

In cases where you want to access multiple children, you would use [ViewChildren](https://angular.io/api/core/ViewChildren) instead.

If you’d like to learn more about Angular, check out [our Angular topic page](https://www.digitalocean.com/community/tags/angularjs) for exercises and programming projects.