## Testing Directives

**Learning Objectives**

• Understand how to test directives using wrapper components.

**Test setup**

We are going to test a directive called the HoverFocusDirective. It has an attribute selector of hoverfocus and if its attached to an element hovering over that element sets the background color to blue.

import {

Directive,

HostListener,

HostBinding

}

from '@angular/core';

@Directive({

selector: '[hoverfocus]'

})

export class HoverFocusDirective {

@HostBinding("style.background-color") backgroundColor: string;

@HostListener('mouseover') onHover() {

this.backgroundColor = 'blue';

}

@HostListener('mouseout') onLeave() {

this.backgroundColor = 'inherit';

}

}

It uses @HostListener to listen to mouseover and mouseout events on it’s host element and it also uses @HostBinding to set the style property of it’s host element.

The starting point for our test suite is very similar to our previous examples:

import {TestBed} from '@angular/core/testing';

import {HoverFocusDirective} from './hoverfocus.directive';

describe('Directive: HoverFocus', () => {

beforeEach(() => {

TestBed.configureTestingModule({

declarations: [HoverFocusDirective]

});

});

})

;

**Test wrapper component**

To test a directive we typically create a *dummy* testing component so we can interact with the directive and test it’s effect on the components view, like so:

@Component({

template: `<input type="text" hoverfocus>`

①

})

class TestHoverFocusComponent {

}

① The directive is associated with an input control in the components view.

Now we have a component to work with we can configure the test bed and get the required references for the tests, like so:

describe('Directive: HoverFocus', () => {

let component: TestHoverFocusComponent;

let fixture: ComponentFixture<TestHoverFocusComponent>;

let inputEl: DebugElement;

beforeEach(() => {

TestBed.configureTestingModule({

declarations: [TestHoverFocusComponent, HoverFocusDirective] ① });

fixture = TestBed.createComponent(TestHoverFocusComponent); ②

component = fixture.componentInstance;

inputEl = fixture.debugElement.query(By.css('input'));

});

});

1. We declare *both* the directive we want to test and the dummy test component.
2. We grab a reference to the component fixture as well as the component and a the input DebugElement from the component view.

**Interacting and inspecting the the view**

We now have all the pieces we need to create a test spec for the directive itself:

it('hovering over input', () => {

inputEl.triggerEventHandler('mouseover', null); ①

fixture.detectChanges();

expect(inputEl.nativeElement.style.backgroundColor).toBe('blue'); ②

inputEl.triggerEventHandler('mouseout', null);

fixture.detectChanges();

console.log(inputEl.nativeElement.style.backgroundColor);

expect(inputEl.nativeElement.style.backgroundColor).toBe('inherit');

});

1. We use triggerEventHandler to simulate events.
2. The style property on the nativeElement is what we can inspect to see the current style applied to an element.

**Summary**

To test directives we use dummy test components which we can create using the Angular Test Bed and which we can interact with by using a component fixture.

We can trigger events on DebugElements by using the triggerEventHandler function and if we want to see what styles are applied to it we can find it via the nativeElement.style property.

**Listing**

<http://plnkr.co/edit/snp12WgkVCemGf8s5XKW?p=preview>

*hoverfocus.directive.ts*

import {

Directive,

HostListener,

HostBinding

}

from '@angular/core';

@Directive({

selector: '[hoverfocus]'

})

export class HoverFocusDirective {

@HostBinding("style.background-color") backgroundColor: string;

@HostListener('mouseover') onHover() {

this.backgroundColor = 'blue';

}

@HostListener('mouseout') onLeave() {

this.backgroundColor = 'inherit';

}

}

*hoverfocus.directive.spec.ts*

/\* tslint:disable:no-unused-variable \*/

import {TestBed, ComponentFixture} from '@angular/core/testing'; import {Component, DebugElement} from "@angular/core"; import {By} from "@angular/platform-browser"; import {TestBed} from '@angular/core/testing'; import {HoverFocusDirective} from './hoverfocus.directive';

@Component({

template: `<input type="text" hoverfocus>`

})

class TestHoverFocusComponent { }

describe('Directive: HoverFocus', () => {

let component: TestHoverFocusComponent;

let fixture: ComponentFixture<TestHoverFocusComponent>;

let inputEl: DebugElement;

beforeEach(() => {

TestBed.configureTestingModule({

declarations: [TestHoverFocusComponent, HoverFocusDirective]

});

fixture = TestBed.createComponent(TestHoverFocusComponent);

component = fixture.componentInstance;

inputEl = fixture.debugElement.query(By.css('input'));

});

it('hovering over input', () => {

inputEl.triggerEventHandler('mouseover', null);

fixture.detectChanges();

expect(inputEl.nativeElement.style.backgroundColor).toBe('blue');

inputEl.triggerEventHandler('mouseout', null);

fixture.detectChanges();

expect(inputEl.nativeElement.style.backgroundColor).toBe('inherit');

});

});