Testing Attribute Directives

Contents

An *attribute directive* modifies the behavior of an element, component or another directive. Its name reflects the way the directive is applied: as an attribute on a host element.

> For the sample app that the testing guides describe, see the sample app.
>
> For the tests features in the testing guides, see tests.

Testing the `HighlightDirective`

The sample application's `HighlightDirective` sets the background color of an element based on either a data bound color or a default color (lightgray). It also sets a custom property of the element (`customProperty`) to `true` for no reason other than to show that it can.

app/shared/highlight.directive.ts

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

@Directive({ selector: '[highlight]' })
/**
 * Set backgroundColor for the attached element to highlight color
 * and set the element's customProperty to true
 */
export class HighlightDirective implements OnChanges {

  defaultColor =  'rgb(211, 211, 211)'; // lightgray

  @Input('highlight') bgColor: string;

  constructor(private el: ElementRef) {
    el.nativeElement.style.customProperty = true;
  }

  ngOnChanges() {
    this.el.nativeElement.style.backgroundColor = this.bgColor || this.defaultColor;
```

```
  }
}
```

It's used throughout the application, perhaps most simply in the `AboutComponent`:

app/about/about.component.ts

```
import { Component } from '@angular/core';
@Component({
  template: `
  <h2 highlight="skyblue">About</h2>
  <h3>Quote of the day:</h3>
  <twain-quote></twain-quote>
  `
})
export class AboutComponent { }
```

Testing the specific use of the `HighlightDirective` within the `AboutComponent` requires only the techniques explored in the "Nested component tests" section of Component testing scenarios.

app/about/about.component.spec.ts

```
beforeEach(() => {
  fixture = TestBed.configureTestingModule({
    declarations: [ AboutComponent, HighlightDirective ],
    schemas:      [ NO_ERRORS_SCHEMA ]
  })
  .createComponent(AboutComponent);
  fixture.detectChanges(); // initial binding
});

it('should have skyblue <h2>', () => {
  const h2: HTMLElement = fixture.nativeElement.querySelector('h2');
  const bgColor = h2.style.backgroundColor;
  expect(bgColor).toBe('skyblue');
});
```

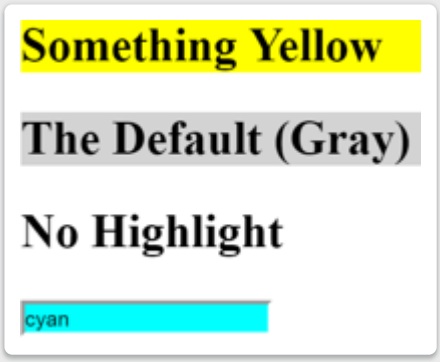
However, testing a single use case is unlikely to explore the full range of a directive's capabilities. Finding and testing all components that use the directive is tedious, brittle, and almost as unlikely to afford full coverage.

Class-only tests might be helpful, but attribute directives like this one tend to manipulate the DOM. Isolated unit tests don't touch the DOM and, therefore, do not inspire confidence in the directive's efficacy.

A better solution is to create an artificial test component that demonstrates all ways to apply the directive.

app/shared/highlight.directive.spec.ts (TestComponent)

```
@Component({
  template: `
  <h2 highlight="yellow">Something Yellow</h2>
  <h2 highlight>The Default (Gray)</h2>
  <h2>No Highlight</h2>
  <input #box [highlight]="box.value" value="cyan"/>`
})
class TestComponent { }
```



> The `<input>` case binds the `HighlightDirective` to the name of a color value in the input box. The initial value is the word "cyan" which should be the background color of the input box.

Here are some tests of this component:

app/shared/highlight.directive.spec.ts (selected tests)

```
beforeEach(() => {
  fixture = TestBed.configureTestingModule({
    declarations: [ HighlightDirective, TestComponent ]
  })
  .createComponent(TestComponent);

  fixture.detectChanges(); // initial binding

  // all elements with an attached HighlightDirective
  des = fixture.debugElement.queryAll(By.directive(HighlightDirective));

  // the h2 without the HighlightDirective
  bareH2 = fixture.debugElement.query(By.css('h2:not([highlight])'));
});

// color tests
it('should have three highlighted elements', () => {
```

```
  expect(des.length).toBe(3);
});

it('should color 1st <h2> background "yellow"', () => {
  const bgColor = des[0].nativeElement.style.backgroundColor;
  expect(bgColor).toBe('yellow');
});

it('should color 2nd <h2> background w/ default color', () => {
  const dir = des[1].injector.get(HighlightDirective) as HighlightDirective;
  const bgColor = des[1].nativeElement.style.backgroundColor;
  expect(bgColor).toBe(dir.defaultColor);
});

it('should bind <input> background to value color', () => {
  // easier to work with nativeElement
  const input = des[2].nativeElement as HTMLInputElement;
  expect(input.style.backgroundColor).toBe('cyan', 'initial backgroundColor');

  // dispatch a DOM event so that Angular responds to the input value change.
  input.value = 'green';
  input.dispatchEvent(newEvent('input'));
  fixture.detectChanges();

  expect(input.style.backgroundColor).toBe('green', 'changed backgroundColor');
});



it('bare <h2> should not have a customProperty', () => {
  expect(bareH2.properties.customProperty).toBeUndefined();
});
```

A few techniques are noteworthy:

- The `By.directive` predicate is a great way to get the elements that have this directive *when their element types are unknown*.
- The `:not` pseudo-class in `By.css('h2:not([highlight])')` helps find `<h2>` elements that *do not* have the directive. `By.css('*:not([highlight])')` finds *any* element that does not have the directive.
- `DebugElement.styles` affords access to element styles even in the absence of a real browser, thanks to the `DebugElement` abstraction. But feel free to exploit the `nativeElement` when that seems easier or more clear than the abstraction.
- Angular adds a directive to the injector of the element to which it is applied. The test for the default color uses the injector of the second `<h2>` to get its `HighlightDirective` instance and its `defaultColor`.
- `DebugElement.properties` affords access to the artificial custom property that is set by the directive.

RESOURCES

About

Resource Listing

Press Kit

Blog

Usage Analytics

HELP

Stack Overflow

Gitter

Report Issues

Code of Conduct

COMMUNITY

Events

Meetups

Twitter

GitHub

Contribute

LANGUAGES

简体中文版

正體中文版

日本語版

한국어

Super-powered by Google ©2010-2020. Code licensed under an MIT-style License. Documentation licensed under CC BY 4.0.

Version 10.0.10-local+sha.84d1ba792b.