{@a attribute-directive}

# Testing Attribute Directives

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.

If you'd like to experiment with the application that this guide describes, run it in your browser or download and run it locally.

## 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.

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

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.

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:

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.