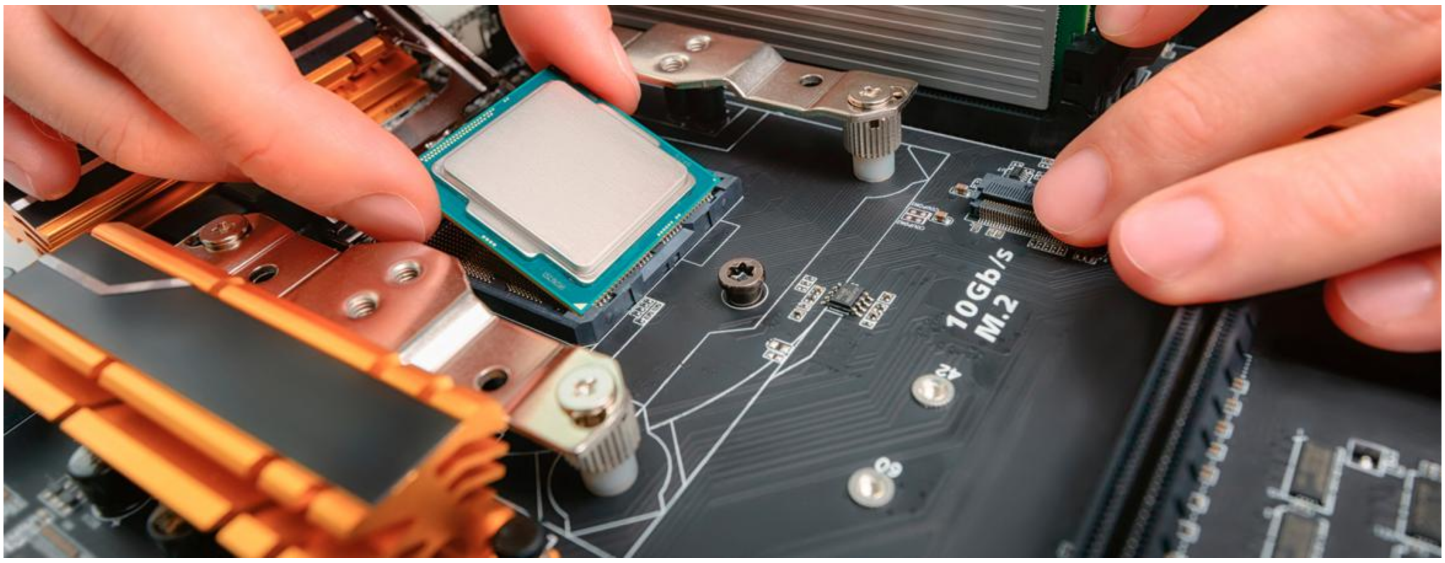

Nov 10 2016 03:30pm | **AngularJS (/taxonomy/term/10)** | **Share ()**

Angular 2: The Difference Between Components and Directives

By James McGeachie

One of the first things Angular 2 developers will learn is the new syntax for creating a Component using a class with the @Component decorator to provide metadata. At the same time they will also learn that a component is a fundamental building block of an Angular 2 web application. However, many Angular 1.X developers moving over have utilized 'directives' in a similar manner (creating custom html elements) and in Angular 2 we have the @Directive decorator. So the question arises - **what's the difference between components and directives in Angular 2?**

We’re going to answer that question in this post, but we’ll go further than just discussing syntax. To best answer this question and provide you with a full understanding, we’ll start by demystifying this terminology and clarifying the underlying concepts.

Directives

What does this mysterious word ‘Directive’ actually mean? Well, let’s first look at the English definition (http://www.merriam-webster.com/dictionary/directive) (Merriam-Webster):

> *“Something that serves to direct, guide, and usually impel toward an action or goal; especially : an authoritative instrument issued by a high-level body or official”*

Okay, it’s an instrument that directs towards an action/goal. Are we done? Not quite. In Computer Science we have a specific concept of a ‘Directive Pragma’. From the Wiki: (https://en.wikipedia.org/wiki/Directive_(programming))

> *“In computer programming, a directive pragma (from "pragmatic") is a language construct that specifies how a compiler (or assembler or interpreter) should process its input. Directives are not part of the language proper”*

So in a programming context, directives provide guidance to the compiler to alter how it would otherwise process input, i.e change some behaviour.

The Angular 1.X Directive

Now that we have a better picture of what a ‘Directive’ is in abstract, let’s look at the Angular 1.X implementation of this concept. This description comes from the Angular 1.X docs:
(https://docs.angularjs.org/guide/directive)

> *“At a high level, directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tell AngularJS's HTML compiler ($compile) to attach a specified behavior to that DOM element (e.g. via event listeners), or even to transform the DOM element and its children.”*

This description fits in with our understanding of the computer science directive concept from the previous section. We are giving instructions to the compiler in order to alter behaviour. Specifically in the context of Angular 1.X, this is to alter behaviour of, or transform, DOM elements. We see this in implemented through the built-in Angular directives we commonly such as ng-if, ng-repeat, ng-class, etc, but also through custom directives we can create ourselves as custom DOM elements, attributes, classes or comments.

We'll talk about what directives are in Angular 2 shortly, but let's talk about components first.

Web Components

A simple way of describing Web Components comes from the Mozilla Developer Network (https://developer.mozilla.org/en-US/docs/Web/Web_Components)

> *"Web Components consists of several separate technologies.You can think of Web Components as reusable user interface widgets that are created using open Web technology."*

So Web Components are a conceptual construct, the idea of creating reusable user interface elements, which can be accomplished by grouping technologies together. This specific group of technologies, per the Web Component spec, is:

- Custom Elements
- HTML Templates
- Shadow DOM
- HTML Imports

With these 4 technologies, you can create a custom element with encapsulated state that renders its template at run-time can and be packaged and distributed via exports and imports. The sum total of this functionality is a Web Component.

Angular Components

In Angular 2,. we can create a component like so:

```
@Component({

        selector: ‘my-component’,

        template: `<p>This is my component template</p>`,

        encapsulation: ViewEncapsulation.Native,

})

export class MyComponent {}
```

Here we define our selector (Giving us a custom element), specify our template, set our encapsulation type and finally export our component class to be accessible throughout our application.

Sound familiar? That’s because Components in Angular 2 are an implementation of the Web Component concept. They are a construct within the Angular 2 framework that is specifically designed to meet the required functionality of a Web Component, as per the evolving standard.

However, they’re not just implemented in Angular 2. In Angular 1.5, the ‘component’ keyword was introduced allowing you to create custom directives that conform with the Web Component spec. These were an ‘after-thought’ in the regard that they weren’t a planned feature of the framework from the beginning like in Angular 2, but they do illustrate an important point - directives can be a mechanism by which components are implemented.

Angular 2 Directives & Components

As promised earlier, let’s look at what directives are in an Angular 2 context. From the API docs for @Directive (https://angular.io/docs/ts/latest/api/core/index/Directive-decorator.html):

> *“Directives allow you to attach behavior to elements in the DOM.”*

Sounds like the same thing as before, and in fact it is. However in Angular 2, directives are split into the following 3 categories:

- Attribute
- Structural
- and… *Component*.

Yes, in Angular 2, **Components are a type of Directive**. For further confirmation on this, let’s look at the API documentation for @Component (https://angular.io/docs/ts/latest/api/core/index/Component-decorator.html), where we see the following description:

> *"Angular components are a subset of directives. Unlike directives, components always have a template and only one component can be instantiated per an element in a template."*

So in closing we can now say this:

Directives in Angular 2 are: The mechanism by which we attach behaviour to elements in the DOM, consisting of Structural, Attribute and Component types.

Components in Angular 2 are: The specific type of directive that allows us to utilize web component functionality - encapsulated, reusable elements available throughout our application.

Further Reading.

We hope this article helps everyone gain a better understanding of the building blocks of an Angular application. If you'd like to go into more detail on the specific types of directive offered in Angular 2, including Components, please see our previous blog post on **The 3 Types of Directive.** (http://dev6.com/angular-2-the-three-types-of-directives)