@Directive v/s @Component in Angular

Asked 4 years ago Active 13 days ago Viewed 99k times

386

83

What is the difference between `@Component` and `@Directive` in Angular? Both of them seem to do the same task and have the same attributes.

What are the use cases and when to prefer one over another?

angular

edited Sep 9 at 6:32
Pato Vargas
2,009 1 12 36

asked Sep 20 '15 at 14:03
Prasanjit Dey
2,224 2 8 15

12 A component is *a directive with a template* and the `@Component` decorator is actually a `@Directive` decorator extended with template-oriented features - source. – Cosmin Ababei Apr 13 '16 at 12:20

2 Directive vs Component is the new Service vs Factory. The confusion is also increased because when actually requiring other components from a component definition you specify them in the `directives` array... maybe Lida Weng comment below helps a bit clarifying that the component "it's actually an extended 'Directive' " – Nobita Jul 10 '16 at 0:25

1 components actually extend directive, they just require you to have a template (HTML) as opposed to directives.. So you'd use directive to modify existing html element, and component makes html elements – Marko Niciforovic Nov 3 '16 at 14:17

8 Answers

491

A @Component requires a view whereas a @Directive does not.

Directives

~~I liken a @Directive to an Angular 1.0 directive with the option~~ ~~`restrict: 'A'`~~ (Directives aren't limited to attribute usage.) Directives add

```
import {Directive} from '@angular/core';

@Directive({
    selector: "[logOnClick]",
    hostListeners: {
        'click': 'onClick()',
    },
})
class LogOnClick {
    constructor() {}
    onClick() { console.log('Element clicked!'); }
}
```

Which would be used like so:

```
<button logOnClick>I log when clicked!</button>
```

Components

A component, rather than adding/modifying behaviour, actually creates its own view (hierarchy of DOM elements) with attached behaviour. An example use case for this might be a contact card component:

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

@Component({
  selector: 'contact-card',
  template: `
    <div>
      <h1>{{name}}</h1>
      <p>{{city}}</p>
    </div>
  `
})
class ContactCard {
  @Input() name: string
  @Input() city: string
  constructor() {}
}
```

Which would be used like so:

`ContactCard` is a reusable UI component that we could use anywhere in our application, even within other components. These basically make up the UI building blocks of our applications.

In summary

Write a component when you want to create a reusable set of DOM elements of UI with custom behaviour. Write a directive when you want to write reusable behaviour to supplement existing DOM elements.

Sources:

- @Directive documentation
- @Component documentation
- Helpful blog post

edited Aug 27 '17 at 19:37
Lazar Ljubenović
11.4k 4 30 58

answered Sep 20 '15 at 16:39
jaker
5,764 1 15 18

2 does @directive annotation has template/templateUrl property ? – Pardeep jain Oct 28 '15 at 7:59

7 Is this answer still true? The angular2 tutorial itself creates a component without a view – Tamas Hegedus Jan 3 '16 at 1:10

it's without a view, but templateurl or template are mandatory in the component – Luca Trazzi Jan 7 '16 at 16:54

3 I like this kind of answers, but I would really appreciate an update when crucial changes happen to the framework. – Memet Olsen May 9 '16 at 8:02

First two points links are dead now. – Shashank Vivek Sep 14 '16 at 10:06

71

Components

1. To register a component we use `@Component` meta-data annotation.
2. Component is a directive which uses shadow DOM to create encapsulated visual behavior called components. Components are typically used to create UI widgets.
3. Component is used to break up the application into smaller components.

Directive

1. To register directives we use `@Directive` meta-data annotation.
2. Directive is used to add behavior to an existing DOM element.
3. Directive is use to design re-usable components.
4. Many directives can be used per DOM element.
5. Directive doesn't use View.

Sources:

http://www.codeandyou.com/2016/01/difference-between-component-and-directive-in-Angular2.html

edited Sep 28 '17 at 20:20
Aniruddha Das
7,084 9 57 71

answered Jan 8 '16 at 10:20
virender
2,557 3 23 24

You can use as many component instances as you want – Mihai Răducanu May 26 '16 at 20:52

5 Components - point 4. I thinks it's wrong - it can be used multiple times. it's actually an extended 'Directive' – Lida Weng Jul 3 '16 at 10:31

Could have expanded this with examples. – Mukus Oct 10 '16 at 0:35

55

A component is a directive-with-a-template and the `@Component` decorator is actually a `@Directive` decorator extended with template-oriented features.

edited Apr 21 '16 at 22:02
madhead
17.1k 13 99 139

answered Apr 21 '16 at 12:06
yusuf tezel
691 7 15

3 Not sure why you've got downvoted too much. It seems that @Component is a Directive with a template (to generate view) for me. – Harry Ninh Jun 1 '16 at 2:48

22

> page, through both custom elements and attributes that add functionality to our existing components.

http://learnangular2.com/components/

But what directives do then in Angular2+ ?

> Attribute directives attach behaviour to elements.
>
> There are three kinds of directives in Angular:
>
> 1. Components—directives with a template.
> 2. Structural directives—change the DOM layout by adding and removing DOM elements.
> 3. Attribute directives—change the appearance or behaviour of an element, component, or another directive.

https://angular.io/docs/ts/latest/guide/attribute-directives.html

So what's happening in Angular2 and above is **Directives** are attributes which add functionalities to **elements** and **components**.

Look at the sample below from Angular.io:

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

@Directive({ selector: '[myHighlight]' })
export class HighlightDirective {
    constructor(el: ElementRef) {
       el.nativeElement.style.backgroundColor = 'yellow';
    }
}
```

So what it does, it will extends you components and HTML elements with adding yellow background and you can use it as below:

```
<p myHighlight>Highlight me!</p>
```

But components will create full elements with all functionalities like below:

```
import { Component } from '@angular/core';
```

```
  template: `
    <div>Hello my name is {{name}}.
      <button (click)="sayMyName()">Say my name</button>
    </div>
   `
})
export class MyComponent {
  name: string;
  constructor() {
    this.name = 'Alireza'
  }
  sayMyName() {
    console.log('My name is', this.name)
  }
}
```

and you can use it as below:

```
<my-component></my-component>
```

When we use the tag in the HTML, this component will be created and the constructor get called and rendered.

edited May 16 '18 at 0:04

answered Apr 16 '17 at 2:16

Alireza
61.6k 15 197 133

6

Change detection

Only `@Component` can be a node in the change detection tree. This means that you cannot set `ChangeDetectionStrategy.OnPush` in a `@Directive` . Despite this fact, a Directive can have `@Input` and `@Output` properties and you can inject and manipulate host component's `ChangeDetectorRef` from it. So use Components when you need a granular control over your change detection tree.

answered Mar 30 '18 at 4:51

Evgeniy Malyutin
765 6 12

5

behaviour.

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

directives are split into the 3 categories:

- Attribute
- Structural
- Component

Yes, in Angular 2, Components are a type of Directive. According to the Doc,

> “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.”

Angular 2 Components are an implementation of the **Web Component** concept. 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 in summary directives The mechanism by which we attach **behavior** to elements in the DOM, consisting of Structural, Attribute and Component types.
- Components are the specific type of directive that allows us to utilize **web component functionality** AKA reusability - encapsulated, reusable elements available throughout our application.

answered Jul 3 '18 at 12:49
Sachila Ranawaka
22.3k 4 35 59

2

If you refer the official angular docs

```
https://angular.io/guide/attribute-directives
```

There are three kinds of directives in Angular:

3. Attribute directives—change the appearance or behavior of an element, component, or another directive. e.g [ngClass].

As the Application grows we find difficulty in maintaining all these codes. For reusability purpose, we separate our logic in smart components and dumb components and we use directives (structural or attribute) to make changes in the DOM.

answered Dec 19 '18 at 11:38

Akshay Rajput
926 2 9

0

Components

Components are the most basic UI building block of an Angular app. An Angular app contains a tree of Angular components. Our application in Angular is built on a **component tree**. Every component should have its template, styling, life cycle, selector, etc. **So, every component has its structure** You can treat them as an apart standalone small web application with own template and logic and a possibility to communicate and be used together with other components.

Sample .ts file for Component:

```
import { Component } from '@angular/core';

@Component({
    // component attributes
    selector: 'app-training',
    templateUrl: './app-training.component.html',
    styleUrls: ['./app-training.component.less']
})

export class AppTrainingComponent {
    title = 'my-app-training';
}
```

and its ./app.component.html template view:

```
Hello {{title}}
```

Then you can render AppTrainingComponent template with its logic in other components (after adding it into module)

```
<div>
   <app-training></app-training>
</div>
```

and the result will be

```
<div>
   my-app-training
</div>
```

as AppTrainingComponent was rendered here

See more about Components

Directives

Directive changes the appearance or behavior of an existing DOM element. For example [ngStyle] is a directive. Directives **can extend components** (can be used inside them) but they **don't build a whole application**. Let's say they just support components. **They don't have its own template** (but of course, you can manipulate template with them).

Sample directive:

```
@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {

  constructor(private el: ElementRef) { }

  @Input('appHighlight') highlightColor: string;

  @HostListener('mouseenter') onMouseEnter() {
    this.highlight(this.highlightColor || 'red');
  }

  private highlight(color: string) {
    this.el.nativeElement.style.backgroundColor = color;
  }
}
```

```
<p [appHighlight]="color" [otherPar]="someValue">Highlight me!</p>
```

See more about directives

edited Aug 30 at 7:08

answered Aug 30 at 7:02

Przemek Struciński
713 5 8

protected by Community ♦ Apr 6 '17 at 12:57

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?