## Structural Directives

**Learning Objectives**

* Understand what a structural directive is.
* Know why we use the \* character for some directives only.

**Long form structural directives**

*Structural Directives* are directives which *change* the structure of the DOM by adding or removing elements.

There are three built in structural directives, NgIf, NgFor and NgSwitch.

These directives work by using the HTML5 <template> tag. This is a new tag in HTML which is specifically designed to hold *template* code. It can sit under the body element but any content inside it is not shown in the browser.

Using template we can write an ngIf expression as:

<

template [ngIf]='condition'

>

<p>I am the content to show</p>

<

/template

>

If we go back to our joke app example and replace the hiding and showing of a joke with this *template* version of ngIf we would end up with:

<

template [ngIf]="!data.hide"

>

<p class="card-text">

{{ data.punchline }}

</p>

<

/template

>

The

NgFor

version is slightly more complex:

<

template ngFor

①

let-j

②

[ngForOf]="jokes">

③

<joke [joke]="j"></joke>

<

/template

>

1. This is the NgFor directive itself.
2. This is another way of declaring a template local reference variable, equivalent to #j.

③[ngForOf] is an *input property* of the NgFor directive.

**Syntax sugar and** \*

So if we can write ngIf with template what is al the fuss about \*.

When we *prepend* a directive with \* we are telling it to use the element it’s attached to *as* the template.

Looking at the NgIf example from above, these two snippets of code are equivalent:

<

template [ngIf]="!data.hide"

>

<p class="card-text">

{{ data.punchline }}

</p>

<

/template

>

<

p class="card-text"

\*ngIf="!data.hide">

{{ data.punchline }}

<

/p

>

Finally looking at the more complex NgFor example from above, these two snippets of code are also equivalent:

<

template ngFor

let-j

[ngForOf]="jokes">

<joke [joke]="j"></joke>

<

/template

>

<

joke \*ngFor="let j of jokes"

[joke]="j">

<

/joke

>

**Summary**

Structural directives are a type of directive which changes the structure of the DOM.

They take advantage of the HTML5 <template> tag to define the element they want to insert into the

DOM.

We can prepend the directive name with \* to skip having to define a <template> and have the directive use the element it’s attached to as the template.

**Listing**

*script.ts*

import {platformBrowserDynamic} from '@angular/platform-browser-dynamic';

import {

Component,

Directive,

NgModule,

Input,

Output,

EventEmitter,

TemplateRef,

ViewContainerRef

}

from '@angular/core';

import {BrowserModule} from '@angular/platform-browser';

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

//

// Domain Model

//

class Joke {

public hide: boolean;

constructor(public setup: string, public punchline: string) {

this.hide = true;

}

toggle() {

this.hide = !this.hide;

}

}

//

// Structural Directives

//

@Directive({

selector: '[ccIf]'

})

export class CodeCraftIfDirective {

constructor(private templateRef: TemplateRef<any>,

private viewContainer: ViewContainerRef) {

}

@Input() set ccIf(condition: boolean) {

if (condition) {

this.viewContainer.createEmbeddedView(this.templateRef);

} else {

this.viewContainer.clear();

}

}

}

@Directive({

selector: '[ccFor]'

})

export class CodeCraftForOfDirective {

constructor(private templateRef: TemplateRef<any>,

private viewContainer: ViewContainerRef) {

}

@Input() set ccForOf(collection: any) {

if (condition) {

this.viewContainer.createEmbeddedView(this.templateRef);

} else {

this.viewContainer.clear();

}

}

}

//

// Components

//

//

@Component({

selector: 'joke',

template: `

<

div class="card card-block"

>

<h4 class="card-title">

{{ data.setup }}

</h4>

<ng-template [ngIf]="!data.hide">

<p class="card-text">

{{ data.punchline }}

</p>

</ng-template>

<button class="btn btn-primary"

(click)="data.toggle()">Tell Me

</button>

<

/div

>

`

})

class JokeComponent implements OnInit {

@Input('joke') data: Joke;

}

@Component({

selector: 'joke-list',

template: `

<

ng-template ngFor

let-j

[ngForOf]="jokes">

<joke [joke]="j"></joke>

<

/ng-template

>

`

})

class JokeListComponent {

jokes: Joke[] = [];

constructor() {

this.jokes = [

new Joke("What did the cheese say when it looked in the mirror?", "Hello-me

(

Halloumi)"),

new Joke("What kind of cheese do you use to disguise a small horse?", "Mask-a-

pony (Mascarpone)"),

new Joke("A kid threw a lump of cheddar at me", "I thought

‘

That

’

s not very

mature

’

"),

];

}

}

@Component({

selector: 'app',

template: `

<

joke-list></joke-list

>

`

})

class AppComponent {

}

//

// Bootstrap

//

@NgModule({

imports: [BrowserModule],

declarations: [

AppComponent,

JokeComponent,

JokeListComponent,

CodeCraftIfDirective

],

bootstrap: [AppComponent]

})

export class AppModule {

}

platformBrowserDynamic().bootstrapModule(AppModule);

## Wrapping Up

Directives are just components but without views.

We *attach* directives to elements by adding them as attributes.

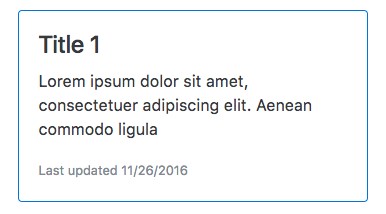
Angular provides a small number of built-in directives and in this section we covered all of them, including:

* NgFor
* NgIf
* NgSwitch
* NgClass
* NgNonBindable

In the next section we will create our own custom directives.

## Activity

Create a component which shows the articles from a list. If the article is a text article show a component like this:

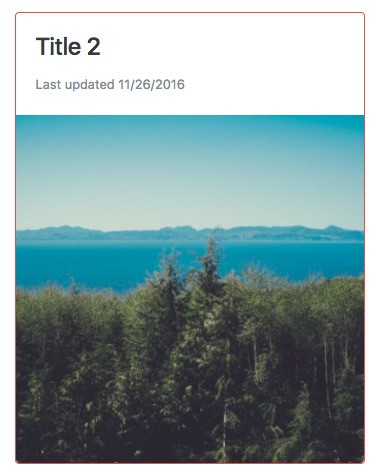




To get the blue border use the class

card-outline-primary

If the article is an image article show a component like this:





To get the red border use the class

card-outline-danger



Use the

shortDate

date format string to get the date formatted as we want.

**Steps**

Fork this plunker:

<http://plnkr.co/edit/i6Lc92Of0DHjFy5xnmCX?p=preview>

Use the built-in directives you’ve just learned to finish off the RecentArticlesComponent.

**Solution**

When you are ready compare your answer to the solution in this plunker:

<http://plnkr.co/edit/aNeU1SVei02wYfjdNefY?p=preview>