# Built-in Directives

## Overview

We’ve covered Components in the last section, in this section and the next we’ll cover the concept of *Directives*.

We’ve touched on a few directives already, such as NgFor.

Directives are components *without* a view. They are components without a template. Or to put it another way, components are directives *with* a view.

Everything you can do with a directive you can also do with a component. But not everything you can do with a component you can do with a directive.

We typically *associate* directives to existing elements by use *attribute* selectors, like so:

<elemenent aDirective></element>

We capitalise the name of directives when we are talking about the directive *class*. For example when we say NgFor we mean the class which defines the NgFor directive.

 When we are talking about either an *instance* of a directive or the *attribute* we use to associate a directive to an element we lowercase the first letter. So ngFor refers to both the *instance* of a directive and the \_attribut\_e name used to associate a directive with an element.

In this section we are going to cover the built in directives that come bundled with Angular.

## NgFor

**Learning Objectives**

* Know how to use the NgFor directive in your application.
* Know how to get the index in the array of the item you are looping over.
* Know how to nest multiple NgFor directives together.

**Basics**

We’ve covered this directive before in the quickstart.

NgFor is a structural directive, meaning that it changes the structure of the DOM.

It’s point is to repeat a given HTML template once for each value in an array, each time passing it the array value as context for string interpolation or binding.

 This directive is the successor of Angular 1s ng-repeat directive.

Let’s take a look at an example:

@Component({

selector: 'ngfor-example',

template: `

<ul>

<li \*ngFor="let person of people">

①

{{ person.name }}

</li>

</ul>

`

})

class NgForExampleComponent {

people: any[] = [

{

"name": "Douglas Pace"

},

{

"name": "Mcleod Mueller"

},

{

"name": "Day Meyers"

},

{

"name": "Aguirre Ellis"

},

{

"name": "Cook Tyson"

}

];

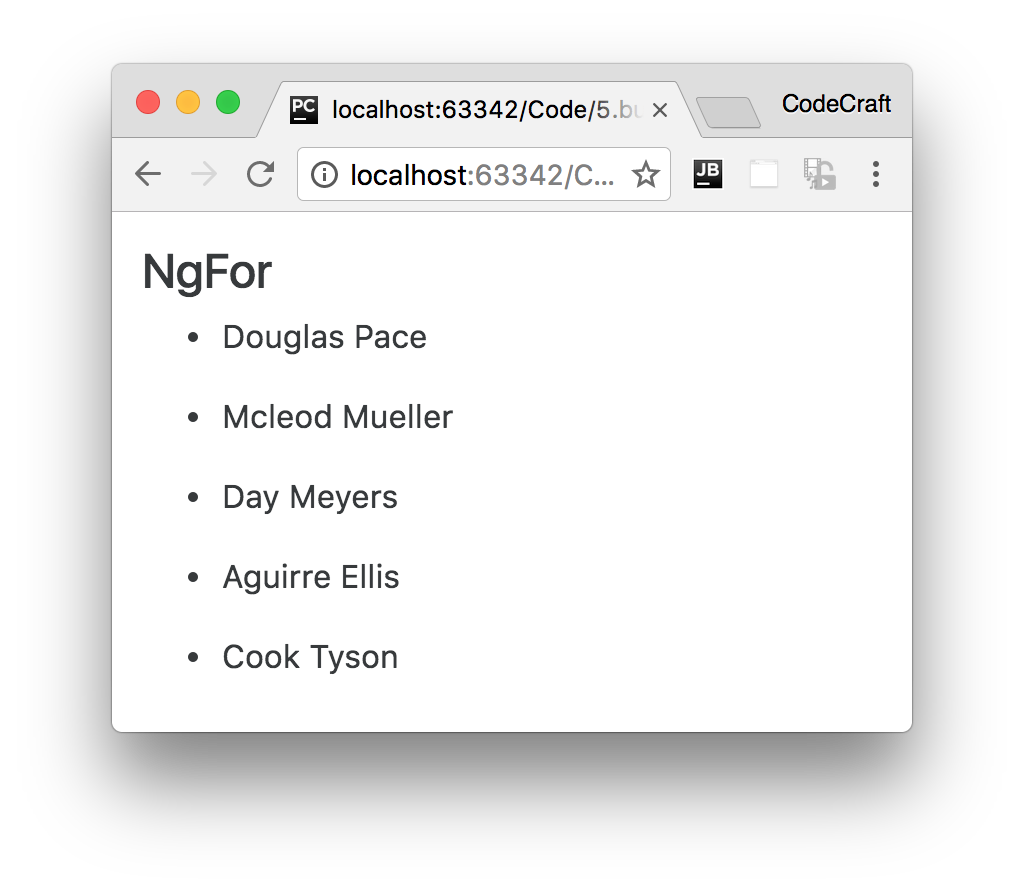
}

① We loop over each person in the people array and print out the persons name.

The syntax is \*ngFor="let <value> of <collection>".

*<value>* is a variable name of your choosing, *<collection>* is a property on your component which holds a collection, usually an array but anything that can be iterated over in a for-of loop.

If we ran the above we would see this:



**Index**

Sometimes we also want to get the *index* of the item in the array we are iterating over.

We can do this by adding another variable to our ngFor expression and making it equal to index, like so:

<

ul>

①

<li \*ngFor="let person of people; let i = index">

①

{{ i + 1 }} - {{ person.name }}

②

</li>

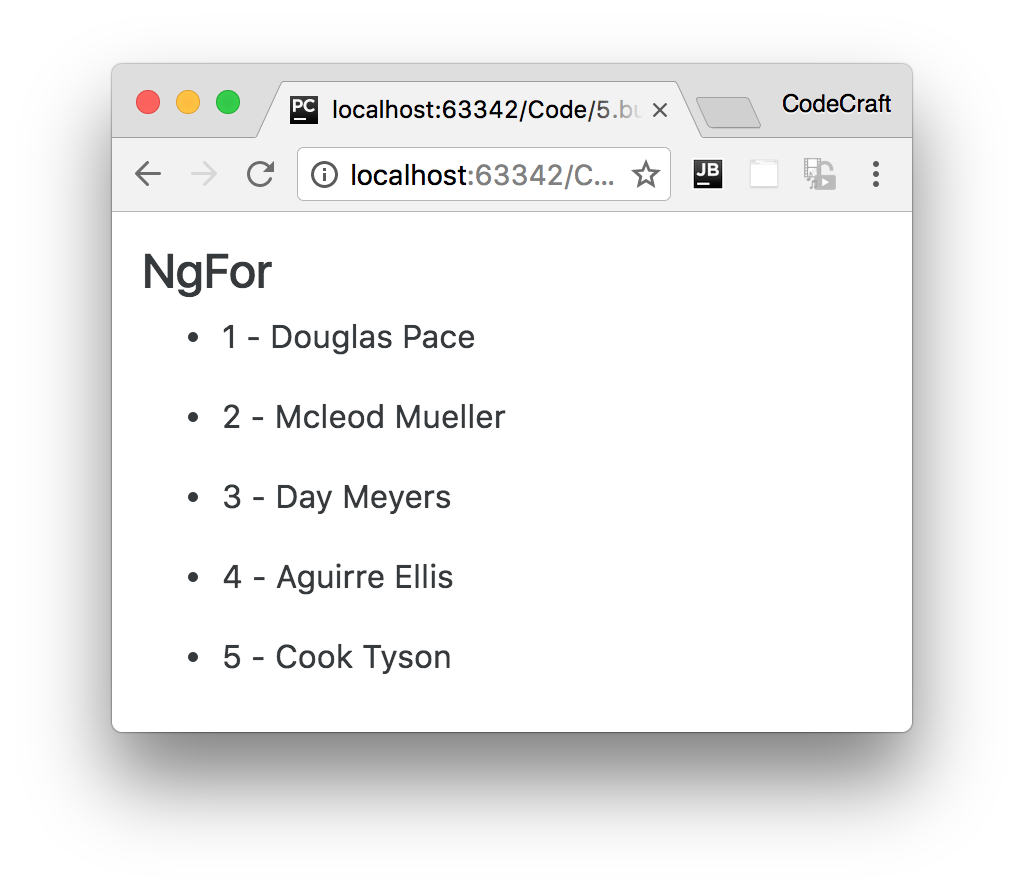
<

/ul

>

1. We create another variable called i and make it equal to the special keyword index.
2. We can use the variable i just like we can use the variable person in our template.

If we ran the above we would now see this:





The index is always zero based, so starts at 0 then 1,2,3,4 etc..



In Angular 1 the variable

$index

would automatically be available for us to use in

an

ng-repeat

directive. In Angular we now need to provide this variable explicitly.

**Grouping**

If our data structure was in fact grouped by country we can use two ngFor directives, like so:

*script.ts*

@Component({

selector: 'ngfor-grouped-example',

template: `

<h4>NgFor (grouped)</h4>

<ul \*ngFor="let group of peopleByCountry">

①

<li>{{ group.country }}</li>

<ul>

<li \*ngFor="let person of group.people">

②

{{ person.name }}

</li>

</ul>

</ul>

`

})

class NgForGroupedExampleComponent {

peopleByCountry: any[] = [

{

'country': 'UK',

'people': [

{

"name": "Douglas Pace"

},

{

"name": "Mcleod Mueller"

},

]

},

{

'country': 'US',

'people': [

{

"name": "Day Meyers"

},

{

"name": "Aguirre Ellis"

},

{

"name": "Cook Tyson"

}

]

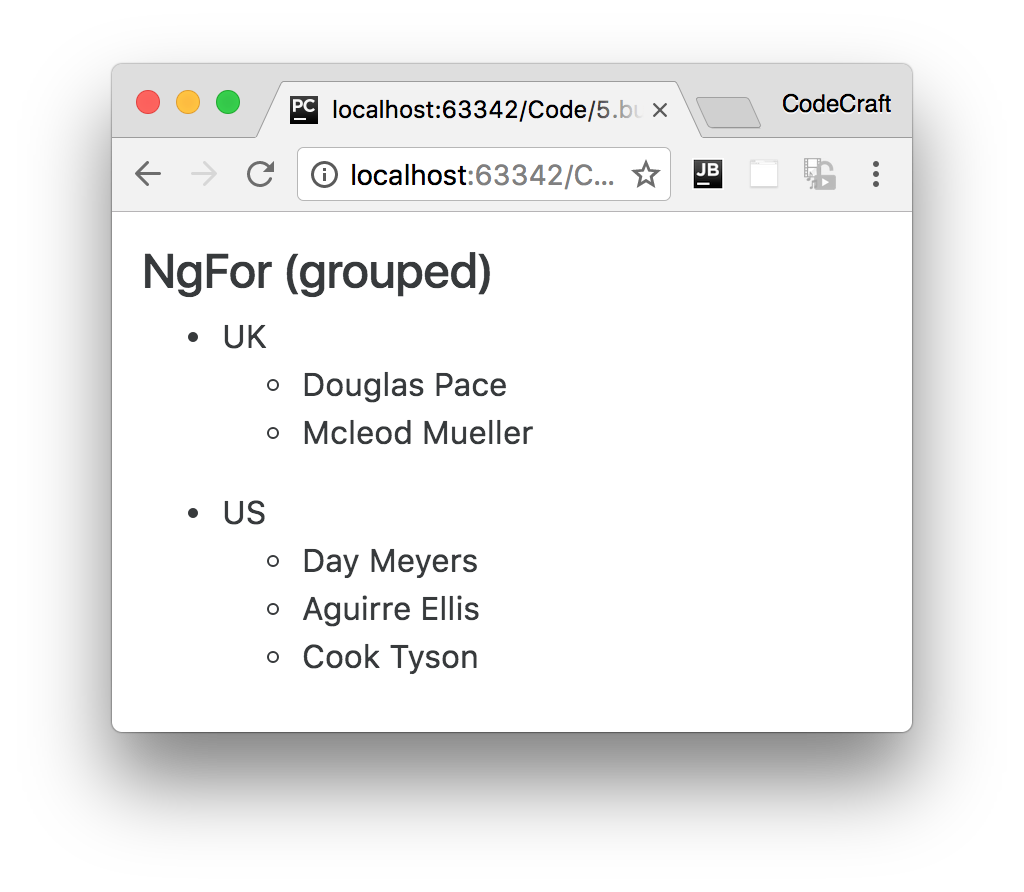
}

];

}

1. The first ngFor loops over the groups, each grop contains a country property which we render out on the next line and a people array property.
2. To loop over the people array we create a second *nested* ngFor directive.

If we ran the above we would see:



**Summary**

We use the NgFor directive to loop over an array of items and create multiple elements dynamically from a template element.

The *template* element is the element the directive is attached to.

We can nest muliple NgFor directives together.

We can get the index of the item we are looping over by assigning index to a variable in the NgFor expression.

**Listing**

<http://plnkr.co/edit/n76eLP4d1SubfA2u9FeD?p=preview>

*script.ts*

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

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

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

@Component({

selector: 'ngfor-example',

template: `<h4>NgFor</h4>

<

ul

>

<li \*ngFor="let person of people; let i = index">

{{ i + 1 }} - {{ person.name }}

</li>

<

/ul

>

`

})

class NgForExampleComponent {

people: any[] = [

{

"name": "Douglas Pace"

},

{

"name": "Mcleod Mueller"

},

{

"name": "Day Meyers"

},

{

"name": "Aguirre Ellis"

},

{

"name": "Cook Tyson"

}

];

}

@Component({

selector: 'ngfor-grouped-example',

template: `<h4>NgFor (grouped)</h4>

<

ul \*ngFor="let group of peopleByCountry"

>

<li>{{ group.country }}</li>

<ul>

<li \*ngFor="let person of group.people">

{{ person.name }}

</li>

</ul>

<

/ul

>

`

})

class NgForGroupedExampleComponent {

peopleByCountry: any[] = [

{

'country': 'UK',

'people': [

{

"name": "Douglas Pace"

},

{

"name": "Mcleod Mueller"

},

]

},

{

'country': 'US',

'people': [

{

"name": "Day Meyers"

},

{

"name": "Aguirre Ellis"

},

{

"name": "Cook Tyson"

}

]

}

];

}

@Component({

selector: 'directives-app',

template: `

<ngfor-grouped-example></ngfor-grouped-example>

<ngfor-example></ngfor-example>

`

})

class DirectivesAppComponent {

}

@NgModule({

imports: [BrowserModule],

declarations: [

NgForExampleComponent,

NgForGroupedExampleComponent,

DirectivesAppComponent],

bootstrap: [DirectivesAppComponent],

})

class AppModule {

}

platformBrowserDynamic().bootstrapModule(AppModule);