

Angular 2+

Notes for Professionals

Chapter 2: Components

Angular components are elements composed by a template that will render your application.

Section 2.1: A simple component

To create a component we add @component decorator in a class passing some parameters:

- **providers:** Resources that will be injected into the component constructor
- **selector:** The query selector that will find the element in the HTML and replace by the
- **styles:** inline styles. NOTE: DO NOT use this parameter with require, it works on dev
- **builds the application in production** all your styles are lost
- **stylesUrl:** Array of path to style files
- **templateUrl:** String that contains your HTML
- **template:** Path to a HTML file

There are other parameters you can configure, but the listed ones are what you will use

A simple example:

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

@Component({
  selector: 'app-required',
  styles: ['required.component.scss'],
  templateUrl: 'required.component.html',
})
export class RequiredComponent { }
```

Section 2.2: Templates & Styles

Templates are HTML files that may contain logic.

You can specify a template in two ways:

Passing template as a file path

```
@Component({
  templateUrl: 'my-component.html',
})
```

Passing a template as an inline code

```
@Component({
  template: '<div>My template here</div>',
})
```

Templates may contain styles. The styles declared in @Component are different from anything applied in the component will be restricted to this scope. For example:

```
div { background: red; }
```

All divs inside the component will be red, but if you have other components, other divs in your HTML they will not be changed at all.

Angular 2+ Notes for Professionals

Chapter 4: Directives

Section 4.1: *ngFor

format: components:

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

// Defines example component and associated template
@Component({
  selector: 'example',
  template: '<div>*ngFor: let f of fruit > {{f}} </div>',
  styleUrls: ['example.component.css'],
})
export class ExampleComponent {
  // Array of fruit to be iterated by *ngFor
  fruit = ['Apples', 'Oranges', 'Bananas', 'Limes', 'Lemons'];
}
```

Output:

```
<div>Apples</div>
<div>Oranges</div>
<div>Bananas</div>
<div>Limes</div>
<div>Lemons</div>
```

In its most simple form, *ngFor has two parts: let variableName of object/array

In the case of fruit = ['Apples', 'Oranges', 'Bananas', 'Limes', 'Lemons'],

Apples, Oranges, and so on are the values inside the array fruit.

[value]="f" will be equal to each current fruit (f) that *ngFor has iterated over.

Unlike AngularJS, Angular2 has not continued with the use of ng-options for <select> and ng-repeat for <table> general repetitions.

*ngFor is very similar to ng-repeat with slightly varied syntax.

References:

Angular2 | [Displaying Data](#)

Angular2 | [*ngFor](#)

Angular 2+ Notes for Professionals

Chapter 11: How to use ngfor

The ngfor directive is used by Angular2 to instantiate a template once for every item in an iterable object. This directive binds the iterable to the DOM, so if the content of the iterable changes, the content of the DOM will be also changed.

Section 11.1: *ngFor with pipe

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'sum'
})
```

```
export class ExamplePipe implements PipeTransform {
  transform(value: string): string {
    if (value && value != 0) {
      return value;
    }
  }
}
```

```
@Component({
  selector: 'example-component',
  template: '<div>
    <div *ngFor="let number of numbers | sum" >
      {{number}}
    </div>
  </div>'
})
```

```
export class ExampleComponent {
  let numbers = List<number> = Array<number>.of(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
}
```

Section 11.2: Unordered list example

```
<ul>
  <li *ngFor="let item of items" >{{item.name}}</li>
</ul>
```

Section 11.3: More complex template example

```
<div *ngFor="let item of items" >
  <div [class.name]="item.name">
    <div [class.description]="item.description">
      {{item.description}}
    </div>
  </div>
</div>
```

Section 11.4: Tracking current interaction example

```
<div *ngFor="let item of items; let i = index" >
  <div [class.number]="i">{{i}}</div>
</div>
```

In this case, i will take the value of index, which is the current loop iteration.

Angular 2+ Notes for Professionals

200+ pages
of professional hints and tricks

Contents

About	1
Chapter 1: Getting started with Angular 2+	2
Section 1.1: Getting started with Angular 2 with node.js/expressjs backend (http example included)	2
Section 1.2: Install angular2 with angular-cli	7
Section 1.3: Getting started with Angular 2 without angular-cli	10
Section 1.4: Getting through that pesky company proxy	14
Section 1.5: Keeping Visual Studios in sync with NPM and NODE Updates	15
Section 1.6: Let's dive into Angular 4!	16
Chapter 2: Components	20
Section 2.1: A simple component	20
Section 2.2: Templates & Styles	20
Section 2.3: Testing a Component	21
Section 2.4: Nesting components	22
Chapter 3: Component interactions	24
Section 3.1: Pass data from parent to child with input binding	24
Section 3.2: Parent - Child interaction using @Input & @Output properties	30
Section 3.3: Parent - Child interaction using ViewChild	31
Section 3.4: Bidirectional parent-child interaction through a service	32
Chapter 4: Directives	35
Section 4.1: *ngFor	35
Section 4.2: Attribute directive	36
Section 4.3: Component is a directive with template	36
Section 4.4: Structural directives	36
Section 4.5: Custom directive	36
Section 4.6: Copy to Clipboard directive	36
Section 4.7: Testing a custom directive	38
Chapter 5: Page title	40
Section 5.1: changing the page title	40
Chapter 6: Templates	41
Section 6.1: Angular 2 Templates	41
Chapter 7: Commonly built-in directives and services	42
Section 7.1: Location Class	42
Section 7.2: AsyncPipe	42
Section 7.3: Displaying current Angular 2 version used in your project	43
Section 7.4: Currency Pipe	43
Chapter 8: Directives & components : @Input @Output	44
Section 8.1: Angular 2 @Input and @Output in a nested component	44
Section 8.2: Input example	45
Section 8.3: Angular 2 @Input with asynchronous data	46
Chapter 9: Attribute directives to affect the value of properties on the host node by using the @HostBinding decorator.	48
Section 9.1: @HostBinding	48
Chapter 10: How to Use ngIf	49
Section 10.1: To run a function at the start or end of *ngFor loop Using *ngIf	49
Section 10.2: Display a loading message	49
Section 10.3: Show Alert Message on a condition	49

Section 10.4: Use *ngIf with*ngFor	50
Chapter 11: How to use ngfor	51
Section 11.1: *ngFor with pipe	51
Section 11.2: Unordered list example	51
Section 11.3: More complex template example	51
Section 11.4: Tracking current interaction example	51
Section 11.5: Angular 2 aliased exported values	52
Chapter 12: Angular - ForLoop	53
Section 12.1: NgFor - Markup For Loop	53
Section 12.2: *ngFor with component	53
Section 12.3: Angular 2 for-loop	53
Section 12.4: *ngFor X amount of items per row	54
Section 12.5: *ngFor in the Table Rows	54
Chapter 13: Modules	55
Section 13.1: A simple module	55
Section 13.2: Nesting modules	55
Chapter 14: Pipes	57
Section 14.1: Custom Pipes	57
Section 14.2: Built-in Pipes	58
Section 14.3: Chaining Pipes	58
Section 14.4: Debugging With JsonPipe	59
Section 14.5: Dynamic Pipe	59
Section 14.6: Unwrap async values with async pipe	60
Section 14.7: Stateful Pipes	61
Section 14.8: Creating Custom Pipe	62
Section 14.9: Globally Available Custom Pipe	63
Section 14.10: Extending an Existing Pipe	63
Section 14.11: Testing a pipe	63
Chapter 15: OrderBy Pipe	65
Section 15.1: The Pipe	65
Chapter 16: Angular 2 Custom Validations	68
Section 16.1: get/set FormBuilder controls parameters	68
Section 16.2: Custom validator examples:	68
Section 16.3: Using validators in the FormBuilder	69
Chapter 17: Routing	70
Section 17.1: ResolveData	70
Section 17.2: Routing with Children	72
Section 17.3: Basic Routing	73
Section 17.4: Child Routes	76
Chapter 18: Routing (3.0.0+)	78
Section 18.1: Controlling Access to or from a Route	78
Section 18.2: Add guard to route configuration	79
Section 18.3: Using Resolvers and Guards	80
Section 18.4: Use Guard in app bootstrap	81
Section 18.5: Bootstrapping	81
Section 18.6: Configuring router-outlet	82
Section 18.7: Changing routes (using templates & directives)	82
Section 18.8: Setting the Routes	83
Chapter 19: Dynamically add components using ViewContainerRef.createComponent	85

Section 19.1: A wrapper component that adds dynamic components declaratively	85
Section 19.2: Dynamically add component on specific event(click)	86
Section 19.3: Rendered dynamically created component array on template HTML in Angular 2	87
Chapter 20: Installing 3rd party plugins with angular-cli@1.0.0-beta.10	91
Section 20.1: Add 3rd party library that does not have typings	91
Section 20.2: Adding jquery library in angular-cli project	91
Chapter 21: Lifecycle Hooks	94
Section 21.1: OnChanges	94
Section 21.2: OnInit	94
Section 21.3: OnDestroy	94
Section 21.4: AfterContentInit	95
Section 21.5: AfterContentChecked	95
Section 21.6: AfterViewInit	95
Section 21.7: AfterViewChecked	96
Section 21.8: DoCheck	96
Chapter 22: Angular RXJS Subjects and Observables with API requests	98
Section 22.1: Wait for multiple requests	98
Section 22.2: Basic request	98
Section 22.3: Encapsulating API requests	98
Chapter 23: Services and Dependency Injection	100
Section 23.1: Example service	100
Section 23.2: Example with Promise.resolve	101
Section 23.3: Testing a Service	102
Chapter 24: Service Worker	105
Section 24.1: Add Service Worker to our app	105
Chapter 25: EventEmitter Service	108
Section 25.1: Catching the event	108
Section 25.2: Live example	109
Section 25.3: Class Component	109
Section 25.4: Class Overview	109
Section 25.5: Emmiting Events	109
Chapter 26: Optimizing rendering using ChangeDetectionStrategy	110
Section 26.1: Default vs OnPush	110
Chapter 27: Angular 2 Forms Update	111
Section 27.1: Angular 2 : Template Driven Forms	111
Section 27.2: Angular 2 Form - Custom Email/Password Validation	111
Section 27.3: Simple Password Change Form with Multi Control Validation	113
Section 27.4: Angular 2 Forms (Reactive Forms) with registration form and confirm password validation	114
Section 27.5: Angular 2: Reactive Forms (a.k.a Model-driven Forms)	116
Section 27.6: Angular 2 - Form Builder	117
Chapter 28: Detecting resize events	119
Section 28.1: A component listening in on the window resize event	119
Chapter 29: Testing ngModel	120
Section 29.1: Basic test	120
Chapter 30: Feature Modules	122
Section 30.1: A Feature Module	122
Chapter 31: Bootstrap Empty module in angular 2	123

Section 31.1: An empty module	123
Section 31.2: Application Root Module	123
Section 31.3: Bootstrapping your module	123
Section 31.4: A module with networking on the web browser	123
Section 31.5: Static bootstrapping with factory classes	124
Chapter 32: Lazy loading a module	125
Section 32.1: Lazy loading example	125
Chapter 33: Advanced Component Examples	127
Section 33.1: Image Picker with Preview	127
Section 33.2: Filter out table values by the input	128
Chapter 34: Bypassing Sanitizing for trusted values	130
Section 34.1: Bypassing Sanitizing with pipes (for code re-use)	130
Chapter 35: Angular 2 Data Driven Forms	133
Section 35.1: Data driven form	133
Chapter 36: Angular 2 In Memory Web API	135
Section 36.1: Setting Up Multiple Test API Routes	135
Section 36.2: Basic Setup	135
Chapter 37: Ahead-of-time (AOT) compilation with Angular 2	137
Section 37.1: Why we need compilation, Flow of events compilation and example?	137
Section 37.2: Using AoT Compilation with Angular CLI	138
Section 37.3: Install Angular 2 dependencies with compiler	138
Section 37.4: Add `angularCompilerOptions` to your `tsconfig.json` file	138
Section 37.5: Run ngc, the angular compiler	138
Section 37.6: Modify `main.ts` file to use NgFactory and static platform browser	139
Chapter 38: CRUD in Angular 2 with Restful API	140
Section 38.1: Read from an Restful API in Angular 2	140
Chapter 39: Use native webcomponents in Angular 2	141
Section 39.1: Include custom elements schema in your module	141
Section 39.2: Use your webcomponent in a template	141
Chapter 40: Update typings	142
Section 40.1: Update typings when: typings WARN deprecated	142
Chapter 41: Mocking @ngrx/Store	143
Section 41.1: Unit Test For Component With Mock Store	143
Section 41.2: Angular 2 - Mock Observable (service + component)	144
Section 41.3: Observer Mock	147
Section 41.4: Unit Test For Component Spying On Store	147
Section 41.5: Simple Store	148
Chapter 42: ngrx	151
Section 42.1: Complete example : Login/logout a user	151
Chapter 43: Http Interceptor	157
Section 43.1: Using our class instead of Angular's Http	157
Section 43.2: Simple Class Extending angular's Http class	157
Section 43.3: Simple HttpClient AuthToken Interceptor (Angular 4.3+)	158
Chapter 44: Animation	160
Section 44.1: Transition between null states	160
Section 44.2: Animating between multiple states	160
Chapter 45: Zone.js	162
Section 45.1: Getting reference to NgZone	162

[Click here to download full PDF material](#)