Contents

[Introduction 1](#_Toc148108729)

[Chapter 02 1](#_Toc148108730)

# Introduction

We followed Pro Angular5th edition book by Adam Freeman.

## Chapter 02

|  |  |  |
| --- | --- | --- |
| Concept | Page |  |
| **Angular Feature\ngModel** | 32 | These brackets denote a two-way data binding, and ngModel is an Angular feature and is used to set up two-way bindings on form elements, such as checkboxes.  <mat-checkbox [(ngModel)]="item.complete" color="primary">            <!--{{ item.complete }}-->          </mat-checkbox> |
| **Angular Material** | 24 | To style the HTML content produced by the application, I am going to use the Angular Material package:  ng add @angular/material@13.0.2 –defaults |
| **Angular Material Components** | 25 | Listing 2-13. Applying Components in the app.components.html File in the src/app Folder:  <mat-toolbar color="primary" class="mat-elevation-z3">    <span class="spacer">{{ username }}'s Todo List</span>    <span class="spacer"></span>    <mat-icon matBadge="{{ itemCount }}" matBadgeColor="accent">checklist</mat-icon>  </mat-toolbar> |
| **Angular Material Table Component** | 28 | <table mat-table [dataSource]="items" class="mat-elevation-z3 fullWidth">  The Angular Material table component is applied by adding the **mat-table attribute** to a standard HTML table element, and the data the table will contain is specified using the dataSource attribute |
| **Component** | 20 | An Angular component is responsible for managing a template and providing it with the data and logic it needs.  …  app.component.ts is a file that defines a component. |
| **Data binding** | 19 | Displaying data in a template is done using double braces—{{ and }}—and Angular evaluates whatever you put between the double braces to get the value to display  …  The {{ and }} characters are an example of a **data binding**, which means they create a relationship between the template and a data value |
| **Data binding\Attribute Binding** | 29 | The square brackets (the [ and ] characters) denote an **attribute binding**, which is a data binding that is used to set an element attribute, providing the Angular Material table component with the data that it will display  <div class="tableContainer">    <table mat-table [dataSource]="items" class="mat-elevation-z3 fullWidth">  **items** is a read-only property defined in the component. |
| **Data binding\One way binding** | 31 | At the moment, the template contains only one-way data bindings, which means they are used to display a data value but are unable to change it. |
| **Data binding\Two way binding** | 32 | Angular also supports two-way data bindings, which can be used to display a data value and change it, too.  The combination of brackets is known as the banana-in-a-box because the round brackets look like a banana contained in a box made by the square brackets. These brackets denote a two-way data binding, and **ngModel** is an Angular feature and is used to set up two-way bindings on form elements, such as checkboxes.  <mat-checkbox [(ngModel)]="item.complete" color="primary">            <!--{{ item.complete }}-->          </mat-checkbox> |
| **Data model** | 17 |  |
| Decorator | 22 | ... @Component({ selector: 'app-root', templateUrl: './app.component.html', styleUrls: ['./app.component.css'] })  ...  @Component({    selector: 'app-root',    templateUrl: './app.component.html',    styleUrls: ['./app.component.css']  })  export class AppComponent {  This is an example of a decorator, which provides metadata about a class. This is the @Component decorator, and, as its name suggests, it tells Angular that this is a component |
| Event Binding | 36 | This is an example of an event binding, and it tells Angular to invoke a component method called addItem, using the value property of the input element as the method argument, and then to clear the input element by setting its value property to the empty string.  <button matSuffix mat-raised-button color="accent" class="addButton"      (click)="addItem(todoText.value); todoText.value = ''">          Add      </button> |
| Feature | 25 | AKA (as know as) Component |
| File extensions | 27 | app.component.css  app.component.html  app.component.ts |
| Html attributes | 29 | The Angular Material table component is applied by adding the **mat-table attribute** to a standard HTML table element, and the data the table will contain is specified using the dataSource attribute:  <table mat-table [dataSource]="items" class="mat-elevation-z3 fullWidth">  …  \*matHeaderCellDef is another example of an attribute  <ng-container matColumnDef="id">        <th mat-header-cell \*matHeaderCellDef>#</th> |
| Html attributes\Cardinal | 36 | The input element has an attribute whose name starts with the # character, which is used to define a variable to refer to the element in the template’s data bindings:  <input matInput placeholder="Enter to-do description" #todoText > |
| Html code | 30 | Alberto note: “html code” is my concept. We already saw this technique several times, a mixture of html with code and code can be in C#, JavaScript or another language.  Normally this code helps to generate Html elements.  In this example we see the java script code *let i = index*.  <td mat-cell \*matCellDef="let i = index"> {{ i + 1 }} </td> |
| keywords | 18 | export, class, and constructor keywords |
|  | 21 | The import keyword declares dependencies on JavaScript modules  ...  import { Component } from '@angular/core';  …  When working with modules, the **import** statement specifies the types that are imported between curly braces  …  Import statements can also be used to declare dependencies on other types like data model classes. |
| modules | 21 | Modules contain types. When we use import statement with modules, within curly braces we have types. |
|  | 24 | Once the package has been installed, open the **app.module.ts** file in the src folder and make the changes shown in Listing 2-12 . These changes declare dependencies on the Angular Material features that are used in this chapter. Confusingly, this file is also called a **module**, which means that there are two types of modules in an Angular project: **JavaScript modules** and **Angular modules**. |
| properties | 23 | Example of read-only properties, username and itemCount:  export class AppComponent {    private list = new TodoList("Bob", [      new TodoItem("Go for run", true),      new TodoItem("Get flowers"),      new TodoItem("Collect tickets"),    ]);    get username(): string {      return this.list.user;    }    get itemCount(): number {      return this.list.items.filter(item => !item.complete).length;    }    get items(): readonly TodoItem[] {      return this.list.items.filter(item => this.showComplete || !item.complete);    } |
| template | 19 | a fragment of HTML that contains expressions that are evaluated by Angular  …  app.component.html is a template file |
|  |  |  |
| **static data types** | 17 | Angular applications are written in TypeScript, which is a superset of JavaScript. I introduce TypeScript in Chapters 3 and 4, but its main advantage is that it supports **static data types**, which makes JavaScript development more familiar to C# and Java developers. (JavaScript has a **prototype-based type system** that many developers find confusing.) |
|  |  |  |

## Aide Memory

Let assume you see **app.component.html** code.

Explain these lines of code:

<mat-toolbar color="primary" class="mat-elevation-z3">

*Mat-toolbar is an Angular Material feature or in other words a Angular component.*

<span class="spacer">{{ username }}'s Todo List</span>

*Curly brackets denotes data binding. The username property is defined in the AppComponent that is defined in app.component.ts file. Looking into AppComponent we see that username is a readonly property:*

get username(): string {…

<mat-icon matBadge="{{ itemCount }}" matBadgeColor="accent">checklist</mat-icon>

The same as previous, we see an example of one-way databinding. The AppComponent readonly itemCount property is used to set matBadge attribute.

<input matInput placeholder="Enter to-do description" #todoText >

We see here an example of an cardinal html attribute #todoText. This is a variable that will refer to input element.

<button matSuffix mat-raised-button color="accent" class="addButton"

    (click)="addItem(todoText.value); todoText.value = ''">

        Add

    </button>

We can see how we defined #todoText variable because we want to get the input value. Here we see an example of event binding (click) and the event handler is addItem, a method of AppComponent.

<table mat-table [dataSource]="items" class="mat-elevation-z3 fullWidth">

Here we see another example of data binding, this binding we call it attribute binding and we use brackets. The items is an AppComponent readonly property and its an array. We are setting dataSource attribute to this array.

<td mat-cell \*matCellDef="let item"> {{ item.task }} </td>

Here we see some html code *let item*. We are creating a variable with the name item. We need to see here the context, this expression let item will be evaluated for each element for the dataSource.

<mat-checkbox [(ngModel)]="item.complete" color="primary">

Here is an example of a two way data binding, using an Angular feature called ngModel. ngModel will set item.Complete property to the value of the checkbox. If we open TodoItem file, you will that in the constructor we have defined the complete property. And yes, we can define class properties directory in the parameters of a constructor.

export class TodoItem {

    constructor(public task: string, public complete: boolean = false)

    {

        // no statements required

    }

}