How to Read This Book

This book aims to be the single most useful resource on learning Angular. By the time you're done reading this book, you (and your team) will have everything you need to build reliable, powerful Angular apps.

Angular is a rich and feature-filled framework, but that also means it can be tricky to understand all of its parts. In this book, we'll walk through everything from installing the tools, writing components, using forms, routing between pages, and calling APIs.

But before we dig in, there are a few guidelines I want to give you **in order to get the most out of this book**. Briefly, I want to tell you:

- how to approach **the code examples** and
- **how to get help** if something goes wrong

Running Code Examples

This book comes with a library of runnable code examples. The code is available to download from the same place where you downloaded this book.

We use the program `npm`[6] to run **every example** in this book. This means you can type the following commands to run any example:

```
npm install
npm start
```

If you're unfamiliar with `npm`, we cover how to get it installed in the Getting Started section in the first chapter.

After running `npm start`, you will see some output on your screen that will tell you what URL to open to view your app.

If you're ever unclear on how to run a particular sample app, checkout the `README.md` in that project's directory. Every sample project contains a `README.md` that will give you the instructions you need to run each app.

[6]https://www.npmjs.com/

Angular CLI

With a couple of minor exceptions, every project in this book was built on Angular CLI[7]. Unless specified otherwise, you can use the `ng` commands in each project.

For instance, to run an example you can run `ng serve` (this is, generally, what is run when you type `npm start`). For most projects you can compile them to JavaScript with `ng build` (we'll talk about this more in the first chapter). And you can run end-to-end tests with `ng e2e`, etc.

Without getting too far into the details, Angular CLI is based on Webpack, a tool which helps process and bundle our various TypeScript, JavaScript, CSS, HTML, and image files. **Angular CLI is not a requirement** for using Angular. It's simply a wrapper around Webpack (and some other tooling) that makes it easy to get started.

Code Blocks and Context

Nearly every code block in this book is pulled from a **runnable code example**, which you can find in the sample code. For example, here is a code block pulled from the first chapter:

code/first-app/angular-hello-world/src/app/app.component.ts

```
export class AppComponent {
  title = 'app works!';
}
```

Notice that the header of this code block states the path to the file which contains this code: `code/first-app/angular-hello-world/src/app/app.component.ts`.

If you ever feel like you're missing the context for a code example, open up the full code file using your favorite text editor. **This book is written with the expectation that you'll also be looking at the example code alongside the manuscript**.

For example, we often need to `import` libraries to get our code to run. In the early chapters of the book we show these `import` statements, because it's not clear where the libraries are coming from otherwise. However, the later chapters of the book are more advanced and they focus on *key concepts* instead of repeating boilerplate code that was covered earlier in the book. **If at any point you're not clear on the context, open up the code example on disk**.

Code Block Numbering

In this book, we sometimes build up a larger example in steps. If you see a file being loaded that has a numeric suffix, that generally means we're building up to something bigger.

[7] https://github.com/angular/angular-cli

For instance, in the Dependency Injection chapter you may see a code block with the filename: `price.service.1.ts`. When you see the `.N.ts` syntax that means we're building up to the ultimate file, which will **not** have a number. So, in this case, the final version would be: `price.service.ts`. We do it this way so that a) we can unit test the intermediate code and b) you can see the whole file in context at a particular stage.

A Word on Versioning

As you may know, the Angular covered in this book is a descendant of an earlier framework called "AngularJS". This can sometimes be confusing, particularly when reading supplementary blogs or documentation.

The official branding guidelines state that "*AngularJS*" is a term reserved for AngularJS 1.x, that is, the early versions of "Angular".

Because the new version of Angular used TypeScript (instead of JavaScript) as the primary language, the 'JS' was dropped, leaving us with just *Angular*. For a long time the only consistent way to distinguish the two was folks referred to the *new* Angular as *Angular 2*.

However, the Angular team in 2017 switched to *semantic versioning* with a new major-release upgrade slated for every 6 months. Instead of calling the next versions *Angular 4*, *Angular 5*, and so on, the number is also dropped and it's just *Angular*.

In this book, when we're referring to *Angular* we'll just say *Angular* or sometimes *Angular 4*, just to avoid confusion. When we're talking about "the old-style JavaScript Angular" we'll use the term *AngularJS* or *AngularJS 1.x*.

Getting Help

While we've made every effort to be clear, precise, and accurate you may find that when you're writing your code you run into a problem.

Generally, there are three types of problems:

- A "bug" in the book (e.g. how we describe something is wrong)
- A "bug" in our code
- A "bug" in your code

If you find an inaccuracy in how we describe something, or you feel a concept isn't clear, email us! We want to make sure that the book is both accurate and clear.

Similarly, if you've found a bug in our *code* we definitely want to hear about it.

If you're having trouble getting your own app working (and it isn't *our* example code), this case is a bit harder for us to handle.

Your first line of defense, when getting help with your custom app, should be our unofficial community chat room[8]. We (the authors) are there from time-to-time, but there are hundreds of other readers there who may be able to help you faster than we can.

If you're still stuck, we'd still love to hear from you, and here some tips for getting a clear, timely response.

Emailing Us

If you're emailing us asking for technical help, here's what we'd like to know:

- What revision of the book are you referring to?
- What operating system are you on? (e.g. Mac OS X 10.8, Windows 95)
- Which chapter and which example project are you on?
- What were you trying to accomplish?
- What have you tried[9] already?
- What output did you expect?
- What actually happened? (Including relevant log output.)

The **absolute best way to get technical support** is to send us a short, self-contained example of the problem. Our preferred way to receive this would be for you to send us a Plunkr link by using this URL[10].

That URL contains a runnable, boilerplate Angular app. If you can copy and paste your code into that project, reproduce your error, and send it to us **you'll greatly increase the likelihood of a prompt, helpful response**.

When you've written down these things, email us at us@fullstack.io[11]. We look forward to hearing from you.

Technical Support Response Time

We perform our free, technical support **once per week**.

If you need a faster response time, and help getting **any** of your team's questions answered, then you may consider our premium support option[12].

[8]https://gitter.im/ng-book/ng-book

[9]http://mattgemmell.com/what-have-you-tried/

[10]https://angular.io/resources/live-examples/quickstart/ts/eplnkr.html

[11]mailto:us@fullstack.io

[12]mailto:us@fullstack.io?Subject=Angular%20Premium%20Support&Body=Hello%21%20I%27m%20interested%20in%20premium%20Angular%20support%20for%20our%20team

Chapter Overview

Before we dive in, I want to give you a feel for the rest of the book and what you can expect inside.

The first few chapters provide the **foundation** you need to get up and running with Angular. You'll create your **first apps**, use **the built-in components**, and start **creating your components**.

Next we'll move into intermediate concepts such as using **forms**, using **APIs**, **routing** to different pages, and using *Dependency Injection* to organize our code.

After that, we'll move into more **advanced concepts**. We spend a good part of the book talking about *data architectures*. Managing state in client/server applications is hard and we dive deep into two popular approaches: using **RxJS Observables** and using **Redux**. In these chapters, we'll show how to build the same app, two different ways, so you can compare and contrast and evaluate what's best for you and your team.

After that, we'll discuss how to write complex, **advanced components** using Angular's most powerful features. Then we talk about how to write **tests** for our app and how we can **upgrade our Angular 1 apps** to Angular 4+. Finally, we close with a chapter on writing **native mobile apps** with Angular using **NativeScript**.

By using this book, **you're going to learn how to build real Angular apps** faster than spending hours parsing out-dated blog posts.

So hold on tight - you're about to become an Angular expert, and have a lot of fun along the way. Let's dig in!

- Nate (@eigenjoy[13])

[13]https://twitter.com/eigenjoy