BEGINNING FLUTTER®

BEGINNING

Flutter®

BEGINNING

Flutter®

A HANDS ON GUIDE TO APP DEVELOPMENT

Marco L. Napoli

wrox™
A Wiley Brand

Beginning Flutter®: A Hands On Guide To App Development

Published by
John Wiley & Sons, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2020 by John Wiley & Sons, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-1-119-55082-2
ISBN: 978-1-119-55087-7 (ebk)
ISBN: 978-1-119-55085-3 (ebk)

Manufactured in the United States of America

Library of Congress Control Number: 2019940772

To God; my wife Carla; my children, Michael, Timothy, and Joseph; and you, the reader.

ABOUT THE AUTHOR

Marco L. Napoli is the CEO of Pixolini, Inc., and an experienced mobile, web, and desktop app developer. He has a strong proven record in developing visually elegant and simple-to-use systems. He wrote his first native iOS app in 2008. His work and published apps can be seen at `www.pixolini.com`.

He has loved computers from an early age. His dad noticed and bought him a PC, and he has been developing software since. He attended the University of Miami for an architecture degree, but he had already started his own business, and after four years he decided architecture was not for him. He developed systems for a diverse mix of industries including banking, healthcare, real estate, education, trucking, entertainment, and horizontal markets. Later, a leading banking software company acquired his MLN Enterprises, Inc., company. The main products were mortgage banking, processing, and marketing software.

Next, he started consulting and later created IdeaBlocks, Inc., with the purpose of software development consulting. He developed for a client that sold hospitality software for mobile, desktop, and web platforms. The main product focus was on hotel sales, catering, webspace, guest service, and maintenance software. The products synced via cloud servers using Microsoft SQL Server with encryption applied to sensitive data. His client's customers included Hyatt Place and Summerfield, Hilton Hotel, Holiday Inn, Hampton Inn, Marriott, Best Western, Radisson Hotel, Sheraton Hotels, Howard Johnson, Embassy Suites, and many more. Once his contract was done, he closed IdeaBlocks.

Today, his focus is running Pixolini. He develops mobile, desktop, and web apps for iOS, Mac, Android, Windows, and the Web. He also teaches a course at Udemy using a web app that he developed for analyzing real estate investment calculations. He has developed and published more than 10 apps in each respective store.

He was interviewed by Hillel Coren for the "It's All Widgets Flutter Podcast" on November 27, 2018, and the episode can be found at `https://itsallwidgets.com/podcast/episodes/1/marco-napoli`.

"I cannot code without espresso, cappuccino, or coffee, and I love martial arts."

Marco is married to Carla, and they have three amazing children.

ABOUT THE TECHNICAL EDITOR

Zeeshan Chawdhary is an avid technologist, with 14 years of experience in the industry. Having started his career with mobile development with J2ME, he soon ventured into web development, creating robust and scalable web applications. As a chief technology officer, he has led teams to build web and mobile apps for companies such as Nokia, Motorola, Mercedes, GM, American Airlines, and Marriott. He is currently director of development on an international team, serving clients with technologies like Magento, WordPress, WooCommerce, Laravel, NodeJS, Google Puppeteer, ExpressJS, ReactJS, and .NET. He has also authored books on iOS, Windows Phone, and iBooks.

Zeeshan is based in Mumbai, India. He can be reached at `imzeeshanc@gmail.com` or on Twitter @imzeeshan, and he maintains a Medium publication at `https://medium.com/@imzeeshan`.

ACKNOWLEDGMENTS

I want to thank the talented team at Wiley, including all of the editors, managers, and many people behind the scenes who helped to get this book published. My thanks to Devon Lewis for recognizing early on that Flutter is having a major impact on the industry, to Candace Cunningham for her project editing skills and her insights, to Zeeshan Chawdhary for his technical input and suggestions, to Barath Kumar Rajasekaran and his team for getting the production of the book ready, and to Pete Gaughan for always being available.

A special thanks to the Flutter team at Google, especially Tim Sneath, Ray Rischpater, and Filip Hráček, for their kindness and invaluable feedback.

A thank-you to my wife and children who have patiently listened and given feedback for the projects created in this book.

CONTENTS

INTRODUCTION

Flutter was unveiled at the 2015 Dart Developer Summit under the name Sky. Eric Seidel (engineer director for Flutter at Google) opened his talk by saying that he was there to speak about Sky, which was an experimental project presented as "Dart on mobile." He had built and published a demo on the Android Play Store, and he started the demo by stating that there was no Java drawing this application, meaning it was native. The first feature Eric showed was a square spinning. Driving the device at 60 Hertz was Dart, which was the first goal for the system: to be fast and responsive. (He wanted to go much faster [i.e., 120 Hertz], but he was restricted by the capability of the device he was using.) Eric went on to show multitouch, fast scrolling, and other features. Sky provided the best mobile experience (for users and developers); the developers took lessons from working on the Web, and they thought they could do better. The user interface (UI) and the business logic were both written in Dart. The goal was to be platform-agnostic.

Fast-forward to 2019, and Flutter now is powering Google's smart display platform including the Google Home Hub and is the first step toward supporting desktop apps with Chrome OS. The result is that Flutter supports desktop apps running on Mac, Windows, and Linux. Flutter is described as a portable UI framework for all screens like mobile, web, desktop, and embedded devices from a single codebase.

This book teaches you how to develop mobile applications for iOS and Android from a single codebase by using the Flutter framework and Dart as the programming language. As Flutter is expanding beyond mobile, you can take the knowledge that you learn in this book and apply it to other platforms. You don't need to have previous programming experience; the book starts with the basics and progresses to developing production-ready applications.

I wrote this book in a simple, down-to-earth style to teach you each concept. You can follow the "Try It Out" practice-style exercises to implement what you learn and create feature-focused applications.

Each chapter builds upon the previous ones and adds new concepts to advance your knowledge for building fast, beautiful, animated, and functional apps. By the end of this book, you'll be able to take the knowledge and techniques you have learned and apply them to develop your own applications. In the last four chapters of the book, you'll create a journal app with the ability to save data locally and a second journal app that adds mood tracking with state management, authentication, and multidevice data cloud syncing capabilities including offline sync, which is a must for today's mobile applications. I have made every effort to teach you the techniques using a friendly and commonsense approach so you can learn the basics all the way to advanced concepts needed for the workplace.

From the first time I saw Google presenting Flutter, it has captured my attention. What especially attracted me to Flutter was the widgets concept. You take widgets and nest (*composition*) them together to create the UI needed, and best of all, you can easily create your own custom widgets. The other major item that attracted me to Flutter was the ability to develop for iOS and Android from a single codebase; this is something I had been needing for a long time and never found a great solution until Flutter. Flutter is declarative; it's a modern reactive framework where widgets handle what the UI should look like according to their current state.

My passion for developing with Flutter and Dart keeps growing, and I decided to write this book to share my experiences and expertise with others. I firmly believe the book teaches everyone from beginners to knowledgeable developers, giving them the tools and knowledge to build and advance as a multiplatform developer. This book is full of tips, insights, what-if scenarios, diagrams, screenshots, sample code, and exercises. All of the project source code is available for download on this book's web page at `www.wiley.com/go/beginningflutter`.

WHO THIS BOOK IS FOR

This book is for everyone who wants to learn how to program mobile, multiplatform applications by using Flutter and Dart. It is for absolute beginners who want to learn to develop modern, fast native performance, and reactive mobile applications for iOS and Android. However, it also takes you from absolute beginner to learning the advanced concepts required to develop production-ready applications. It's also for people who are familiar with programming who want to learn the Flutter framework and the Dart language.

This book is written with the assumption of having no prior programming, Flutter, or Dart experience. If you have programmed in other languages or are familiar with Flutter and Dart, you'll simply get a deeper understanding of each concept and technique.

WHAT THIS BOOK COVERS

The early chapters introduce and cover the architecture of the Flutter framework, the Dart language, and steps for creating a new project. You'll use that knowledge to create new projects for each exercise in the book. Each chapter is written to advance your knowledge by focusing on new concepts. The chapters are also written to be reference material to refresh your knowledge of each concept.

Starting in Chapter 2, as you learn each concept and technique, you'll follow the "Try It Out" practice-style exercises and create new application projects to put into practice what you've learned. As you move forward, each chapter is designed to teach you more advanced topics. The last four chapters focus on creating two production-ready applications by applying previously learned materials and implementing new advanced concepts. You can also find these exercises as part of the code downloads on this book's page at `www.wiley.com/go/beginningflutter`.

HOW THIS BOOK IS STRUCTURED

This book is divided into 16 chapters. Although each chapter builds upon the previous concepts, they are also self-contained and written for you to be able to jump to a particular interest to learn or refresh that topic.

Part I: The Foundations of Flutter Programming

In the first part of the book, you'll get to know the core aspects of Flutter so you have a solid foundation to build on.

> **Chapter 1: Introducing Flutter and Getting Started—**You'll learn how the Flutter framework works behind the scenes and about the benefits of the Dart language. You'll see how `Widget`, `Element`, and `RenderObject` are related, and you'll get an understanding of how they form the widget tree, element tree, and render tree. You'll get an introduction to `StatelessWidget` and `StatefulWidget` and their lifecycle events. You'll learn that Flutter is declarative, meaning Flutter builds the UI to reflect the state of the app. You'll learn how to install the Flutter framework, Dart, editor, and plugins on macOS, Windows, or Linux.
>
> **Chapter 2: Creating a Hello World App—**You'll learn how to create your first Flutter project to get familiarized with the process. By writing this minimal example, you'll learn the basic structure of an application, how to run the app on the iOS simulator and the Android emulator, and how to make changes to the code. At this point, do not worry about understanding the code yet; I'll walk you through it step-by-step in later chapters.
>
> **Chapter 3: Learning Dart Basics—**Dart is the foundation of learning to develop Flutter applications, and in this chapter you'll understand Dart's basic structure. You'll learn how to comment your code, how the `main()` function starts the app, how to declare variables, and how to use the `List` to store an array of values. You'll learn about the operator symbols and how to use them to perform arithmetic, equality, logical, conditional, and cascade notation. You'll learn how to use external packages and classes and how to use the `import` statement. You'll learn how to implement asynchronous programming by using a `Future` object. You'll learn how to create classes to group code logic and use variables to hold data and how to define functions to execute logic.
>
> **Chapter 4: Creating a Starter Project Template—**You'll learn the steps to create a new project that you'll use and replicate to create all of the exercises in this book. You'll learn how to organize files and folders in your project. You'll create the most commonly used names to group your widgets, classes, and files by the type of action needed. You'll learn how to structure widgets and import external packages and libraries.
>
> **Chapter 5: Understanding the Widget Tree—**The widget tree is the result of *composing* (nesting) widgets to create simple and complex layouts. As you start nesting widgets, the code can become harder to follow, so good practice is to try to keep the widget tree as shallow as possible. You'll get an introduction to the widgets that you'll use in this chapter. You'll get an understanding of the effects of a deep widget tree, and you'll learn how to refactor it into a shallow widget tree, resulting in more manageable code. You'll learn three ways to create a shallow widget tree by refactoring with a constant, with a method, and with a widget class. You'll learn the benefits and cons of each technique.

Part II: Intermediate Flutter: Fleshing Out an App

In Part II of the book, you'll get your hands dirty, stepping through how to add functionality that creates great user experiences.

Chapter 6: Using Common Widgets—You'll learn how to use the most common widgets, which are the base building blocks for creating beautiful UI and user experience (UX). You'll learn how to load images from the application's asset bundle and over the Web via a uniform resource locator (URL). You'll learn how to use the included Material Components icons and how to apply decorators to enhance the look and feel of widgets or use them as input guides to entry fields. You'll learn how to use the `Form` widget to validate text field entry widgets as a group. You'll learn different ways to detect orientation to lay out widgets accordingly depending on whether the device is in portrait or landscape mode.

Chapter 7: Adding Animation to an App—You'll learn how to add animation to an app to convey action. When animation is used appropriately, it improves the UX, but too many or unnecessary animations can make the UX worse. You'll learn how to create `Tween` animations. You'll learn how to use the built-in animations by using the `AnimatedContainer`, `AnimatedCrossFade`, and `AnimatedOpacity` widgets. You'll learn how to create custom animations by using the `AnimationController` and `AnimatedBuilder` classes. You'll learn how to create staggered animations by using multiple `Animation` classes. You'll learn how to use the `CurvedAnimation` class for nonlinear effects.

Chapter 8: Creating an App's Navigation—You'll learn that good navigation creates a great UX, making it easy to access information. You'll learn that adding animation while navigating to another page can also improve the UX as long as it conveys an action, rather than being a distraction. You'll learn how to use the `Navigator` widget to manage a stack of routes to move between pages. You'll learn how to use the `Hero` widget to convey a navigation animation to move and size a widget from one page to another. You'll learn different ways to add navigation by using the `BottomNavigationBar`, `BottomAppBar`, `TabBar`, `TabBarView`, and `Drawer` widgets. You'll also learn how to use the `ListView` widget together with the `Drawer` widget to create a list of navigational menu items.

Chapter 9: Creating Scrolling Lists and Effects—You'll learn how to use different widgets to create a scrolling list to help users view and select information. You'll learn how to use the `Card` widget to group information used in conjunction with the scrolling list widgets. You'll learn how to use the `ListView` widget to build a linear list of scrollable widgets. You'll learn how to use the `GridView` to display tiles of scrollable widgets in a grid format. You'll learn how to use the `Stack` widget to overlap, position, and align its children widgets in conjunction with a scrolling list. You'll learn how to implement the `CustomScrollView` to create custom scrolling effects like parallax animation by using sliver widgets like `SliverSafeArea`, `SliverAppBar`, `SliverList`, `SliverGrid`, and more.

Chapter 10: Building Layouts—You'll learn how to nest widgets to build professional layouts. This concept is a major part of creating beautiful layouts, and it's known as *composition*. Basic and complex layouts are mostly based on vertical or horizontal widgets or a combination of both. This chapter's goal is to create a journal entry page displaying details like a header image, title, diary detail, weather, (journal location) address, tags, and footer images. To lay out the page, you'll use widgets such as `SingleChildScrollView`, `SafeArea`, `Padding`, `Column`, `Row`, `Image`, `Divider`, `Text`, `Icon`, `SizedBox`, `Wrap`, `Chip`, and `CircleAvatar`.

Chapter 11: Applying Interactivity—You'll learn how to add interactivity to an app by using gestures. In mobile applications, gestures are the heart of listening to user interaction, and making use of gestures can result in an app with a great UX. However, overusing gestures without adding value by conveying an action can create a poor UX. You'll learn how to use the `GestureDetector` gestures such as the tap, double tap, long press, pan, vertical drag, horizontal drag, and scale. You'll learn how to use the `Draggable` widget to drag over the `DragTarget` widget to create a drag-and-drop effect to change the widget color. You'll learn how to implement the `InkWell` and `InkResponse` widgets that respond to touch and visually show a splash animation. You'll learn how to implement the `Dismissible` widget that is dismissed by dragging. You'll learn how to use the `Transform` widget and the `Matrix4` class to scale and move widgets.

Chapter 12: Writing Platform-Native Code—In some cases, you need to access specific iOS or Android API functionality, and you'll learn how to use platform channels to send and receive messages between the Flutter app and the host platform. You'll learn how to use the `MethodChannel` to send messages from the Flutter app (client side) and the `FlutterMethodChannel` on iOS and `MethodChannel` on Android to receive calls (host side) and send back results.

Part III: Creating Production-Ready Apps

For the final four chapters of the book, you'll move into more advanced territory and prepare to release your sample apps to production.

Chapter 13: Saving Data with Local Persistence—You'll learn how to build a journal app. You'll learn how to persist data over app launches by using the JSON file format and saving the file to the local iOS and Android filesystem. JavaScript Object Notation (JSON) is a common open-standard and language-independent file data format with the benefits of being human-readable text. You'll learn how to create database classes to write, read, and serialize JSON files. You'll learn how to format a list and sort it by date.

In a mobile application, it's important not to block the UI while processing, and you'll learn how to use the `Future` class and the `FutureBuilder` widget. You'll learn how to present a date-selection calendar, validate user entry data, and move focus between entry fields.

You'll also learn how to delete records using the `Dismissible` widget by dragging or flinging on an entry. To sort entries by date, you'll learn how to use the `List().sort` method and the `Comparator` function. To navigate between pages, you'll use the `Navigator` widget, and you'll learn how to use the `CircularProgressIndicator` widget to show that an action is running.

Chapter 14: Adding the Firebase and Firestore Backend —Spanning this chapter, Chapter 15, and Chapter 16, you'll use techniques that you have learned in previous chapters along with new concepts and tie them together to create a production-level mood journaling app. In a production-level app, how would you combine what you learned, improve performance by redrawing only the widgets with data changes, pass state between pages and up the widget tree, handle the user authentication credentials, sync data between devices and the cloud, and create classes that handle platform-independent logic between mobile and web apps? These are the reasons why these last three chapters will teach you how to apply the previous techniques you learned along with new important ones to develop a production-level mobile app.

In these last three chapters, you'll learn how to implement app-wide and local-state management and maximize platform code sharing by implementing the Business Logic Component (BLoC) pattern.

You'll learn how to use authentication and persist data to a cloud database by using Google's Firebase backend server infrastructure, Firebase Authentication, and Cloud Firestore. You'll learn that Cloud Firestore is a NoSQL document database to store, query, and sync data with offline support for mobile and web apps. You'll be able to synchronize data between multiple devices. You'll learn how to set up and build serverless applications.

Chapter 15: Adding State Management to the Firestore Client App—You'll continue to edit the mood journaling app created in Chapter 14. You'll learn how to create app-wide and local-state management that uses the `InheritedWidget` class as a provider to manage and pass `State` between widgets and pages.

You'll learn how to use the BLoC pattern to create BLoC classes, for example, managing access to the Firebase Authentication and Cloud Firestore database service classes. You'll learn how to use the `InheritedWidget` class to pass a reference between the BLoC and pages. You'll learn how to use the reactive approach by using `StreamBuilder`, `StreamController`, and `Stream` to populate and refresh data.

You'll learn how to create service classes to manage the Firebase Authentication API and the Cloud Firestore database API. You'll create and take advantage of an `abstract` class to manage user credentials. You'll learn how to create a data model class to handle the mapping of the Cloud Firestore `QuerySnapshot` to individual records. You'll learn how to create a class to manage a list of mood icons, description, and icon rotation position according to the selected mood. You'll use the `intl` package and learn how to create a date formatting class.

Chapter 16: Adding BLoCs to Firestore Client App Pages—You'll continue to edit the mood journaling app created in Chapter 14 with the additions from Chapter 15.

You'll learn how to apply the BLoC, service, provider, model, and utility classes to the UI widget pages. The benefit of using the BLoC pattern allows for separation of the UI widgets from the business logic. You'll learn how to use dependency injection to inject service classes into the BLoC classes. By using dependency injection, the BLoCs remain platform-agnostic. This concept is extremely important since the Flutter framework is expanding beyond mobile and onto web, desktop, and embedded devices.

You'll learn how to apply app-wide authentication state management by implementing classes that apply the BLoC pattern. You'll learn how to create a login page that implements the BLoC pattern class to validate emails, passwords, and user credentials. You'll learn how to pass state between pages and the widget tree by implementing provider classes (`InheritedWidget`). You'll learn how to modify the home page to implement and create BLoC pattern classes to handle login credentials validation, create a journal entries list, and add and delete individual entries. You'll learn how to create the journal edit page that implements the BLoC pattern classes to add, modify, and save existing entries.

WHAT YOU NEED TO USE THIS BOOK

You'll need to install the Flutter framework and Dart to create the example projects. This book uses Android Studio as its primary development tool, and all projects compile for iOS and Android. For compiling iOS applications, you'll need a Mac computer with Xcode installed. You can also use other editors like Microsoft Visual Studio Code or IntelliJ IDEA. For the last major project, you'll need to create a free Google Firebase account to take advantage of cloud authentication and data syncing, including offline support.

The source code for the samples is available for download from `www.wiley.com/go/beginningflutter`.

CONVENTIONS

To help you get the most from the text and keep track of what's happening, we've used a number of conventions throughout the book.

TRY IT OUT You should work through all the *Try It Out* exercises in the book.

1. These exercises consist of a set of numbered steps.
2. Follow the steps with your copy of the database.

HOW IT WORKS

At the end of each *Try It Out,* the code you've typed will be explained in detail.

As for styles in the text:

- We *italicize* new terms and important words when we introduce them.
- We show keyboard strokes like this: Ctrl+A.
- We show filenames, URLs, and code within the text like so:

  `persistence.properties`

We present code in two different ways:

```
We use a monofont type with no highlighting for most code examples.
We use bold to emphasize code that is particularly important in the present
 context or to show changes from a previous code snippet.
```

ERRATA

We make every effort to ensure that there are no errors in the text or in the code. However, no one is perfect, and mistakes do occur. If you find an error in one of our books, like a spelling mistake or faulty piece of code, we would be very grateful for your feedback. By sending in errata, you may save another reader hours of frustration, and at the same time, you will be helping us provide even higher-quality information.

To find the errata page for this book, go to this the book's page at `www.wiley.com/go/beginning-flutter` and click the Errata link. On this page, you can view all errata that have been submitted for this book and posted by Wrox editors.

If you don't spot "your" error on the Book Errata page, please email our customer service team at `wileysupport@wiley.com` with the subject line "Possible Book Errata Submission." We'll check the information and, if appropriate, post a message to the book's errata page and fix the problem in subsequent editions of the book.