Mastering Flutter

Learn to develop Flutter apps for iOS, Android, desktop and web

Kevin Moore

www.bpbonline.com

First Edition 2025

ISBN: 978-93-65899-177

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

Dedicated to

My wife, two sons and two cats

About the Author

Kevin Moore is currently working as a Flutter developer for two companies and has been an Android developer for over 14 years and a Flutter developer for over 4 years. He has written several books on mobile development, including Android, Flutter, and Kotlin Multi-platform. He is a Google Developer Expert in Flutter and speaks at conferences around the country.

About the Reviewers

- **Randal Schwartz** is a self-taught programmer, writer, trainer, and new media host with a passion for technology and creative pursuits.

  Throughout his career, Randal has honed his skills in various programming languages, including Perl, Dart, and Flutter, and has become a recognized expert in the field. Notably, he has authored several influential books on Perl programming, including "Programming Perl," "Learning Perl," and "Effective Perl Programming."

  He is currently recognized as a Google Developer Expert in the areas of Dart and Flutter (one of 10 in the United States and 150 in the world).

  Randal's professional journey has taken him through diverse roles, from software developer and system administrator to consultant and technical writer. He has contributed his expertise to numerous organizations, including Stonehenge Consulting Services, Inc., O'Reilly & Associates, and TWiT.tv, where he hosted the popular show "FLOSS Weekly."

  Beyond his technical prowess, Randal is also a gifted communicator and educator. He has lectured at conferences, provided technical training, and shared his insights through magazine articles and online platforms. Randal's unique blend of technical expertise, writing talent, and engaging personality has made him a sought-after speaker, author, and consultant in the tech industry.

- **Roman Jaquez** is a Google Developer Expert in Flutter as well as a Google Certified Cloud Architect who loves sharing his passion for Flutter with the wider developer community. He is also the lead organizer at GDG Lawrence and has 10+ years of experience as a software engineer. He enjoys spreading the word about best practices in the industry, ranging from mobile, web, and cloud.

Acknowledgement

I would like to express my gratitude to all those who contributed to the completion of this book.

First and foremost, I extend my heartfelt appreciation to my family and friends for their unwavering support and encouragement throughout this journey. Their love and encouragement have been a constant source of motivation.

I would like to extend a special thanks to the following individuals for their valuable input and contributions to this project: Simon Lightfoot, Randal Schwartz, Roman Jaquez, and Scott Stoll. Thank you for your invaluable support and all the things you have taught me.

I would also like to thank Yiru Gan for the incredible design on the Movie App.

I am immensely grateful to BPB Publications for their guidance and expertise in bringing this book to fruition. Their support and assistance were invaluable in navigating the complexities of the publishing process.

Last but not least, I want to express our gratitude to the readers who have shown interest in the book. Your support and encouragement have been deeply appreciated.

Thank you to everyone who has played a part in making this book a reality.

Preface

Flutter has become one of the most popular multi-platform frameworks available. While others have died out, Flutter continues to grow stronger and have a larger community. It has done so because it is easy to use and easy to create apps for every major platform. Instead of learning each platform and language, you can just learn one.

Comprising nineteen chapters, this book covers a wide range of topics essential for learning Flutter. From the Dart programming language needed to write Flutter apps, to the widget system provided by Flutter you will learn everything you need to start creating apps for every platform.

Chapter 1: **Introduction to Flutter** - Learn all about Flutter and why you should use it. Learn how to install Flutter and get started. You will also choose which IDE you will use to develop apps. You will learn about the Flutter architecture and the basics of how it is put together.

Chapter 2: **Dart Essentials** - Learn about the Dart programming language that is used to write Flutter apps. Dart is a modern programming language with a lot of great features. You will need to know this language if you are to write Flutter apps. You will learn about basic programming fundamentals like variables, control flow, functions, and classes, as well as some more advanced topics like null safety.

Chapter 3: **Building the Movie App** - Learn about the app you will build with this book. This is a beautiful app that showcases popular movies and allows users to learn all about the movie and the cast. You will start the app from scratch, learning how to create your own apps. You will also learn about the different types of widgets Flutter uses. Finally, you will learn about the amazing hot reload feature of Flutter that allows you to keep changing your app while it is running and not have to rebuild each time.

Chapter 4: **Basic Widgets** - Get started with learning some of the most important widgets that Flutter offers. Mastering these building block widgets will help you build screens.

Chapter 5: **Themes, Colors and Fonts** - An app is boring and ugly unless you have a good set of consistent colors and fonts. Learn how to create a theme that has the colors and fonts you need to build the movie app.

Chapter 6: **State Management Fundamentals** - How you manage the state of your data will either make your programming life difficult or easy. Learn some of the state management packages available to Flutter developers.

Chapter 7: **Advanced Widgets** - Now that you know the basic widgets, learn more advanced widgets like ListViews, Grids, Cards, and Slivers.

Chapter 8: **Navigation and Routing** - Learn an easy way to transition from one page to another. Learn how to push and pop pages with ease.

Chapter 9: **Animations and Transitions** - Let your app look professional with engaging animations.

Chapter 10: **Futures and Async/Await** - Prevent your UI from slowing down by performing background tasks asynchronously. Learn all about Dart's Futures.

Chapter 11: **Networking** - Learn how to retrieve data from the cloud. This is really important as there is a lot of great information out there to show your users.

Chapter 12: **Local Storage and Databases** - Learn how to save data for both simple and complex data needs. Learn about shared preferences for simple data and databases for more complex needs.

Chapter 13: **Web and Desktop** - Now that you have created mobile apps, learn how to develop for the desktop (both the Mac and Windows) and the web.

Chapter 14: **Handling User Input and Gestures** - Learn how to handle input from the user, from text fields, and gestures to focus management.

Chapter 15: **Firebase** - Learn how to use one of the most popular cloud databases to store your data remotely.

Chapter 16: **Packages** - Learn how to create your own Dart package. This is really useful to create shared code, either for yourself, your team, or others in the community.

Chapter 17: **Platform Channels and Plugins** - Learn about how you can write code that runs natively on each platform. This is useful if you cannot find a plugin that does what you need. You will build a plugin to save and retrieve native preferences.

Chapter 18: **Testing and Performance** - Learn about the different types of testing, from unit to widget and then to integration. This will allow you to feel confident in your code and will show you any problems that may crop up when you are changing your code. You will also learn about the tools available to measure your app's performance and find and fix those problems.

Chapter 19: **Building and Publishing** - In this final chapter you will learn how to build release versions of your app and publish them to both the Google Play Store and the Apple App Store. This will allow you to reach your users on both platforms.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/d0ff3b

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Mastering-Flutter.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Introduction to Flutter

Introduction

In this chapter, you will take your first steps towards mastering Flutter, *Google's UI toolkit* for crafting beautiful, natively compiled applications for mobile, web, and desktop from a single codebase. You will learn about and install Flutter. You will also learn about the Movie application that you will be creating in this book.

Structure

The chapter covers the following topics:

- Overview of Flutter
- Flutter architecture
- Benefits of Flutter
- Flutter's language: Dart
- Installing Flutter SDK
- Development application

Objectives

By the end of this chapter, you will have an understanding of what Flutter is and the incredible possibilities it unlocks. You will be eager to dive into the next chapters and embark on your journey to become a Flutter master!

Overview of Flutter

Flutter is a UI toolkit developed by *Google*. It has the ability to run on mobile (both Android and iOS), desktop (macOS, Windows, and Linux), and the web. The Flutter toolkit uses a declarative UI built of Widgets. In fact, *Google* likes to say that everything is a widget. There are widgets for almost every task you need, and if not, you just need to create your own, by building a widget with other widgets inside or going all the way and creating a custom widget with its own rendering code. You will learn more in later chapters.

As mentioned above, Flutter works on almost every platform using almost the same code (some separate code needs to be written for platform-specific areas like menus). In fact, *Google* used Flutter on their Nest displays.

History

Flutter has an interesting history; the first version was introduced in 2015 and was known as **Sky**. It ran on Android. It used Dart as the development language. Then in 2018 Flutter 1.0 was released. In 2021, Flutter 2 was released at the *Flutter Engage event*. This version had a canvas-based engine for the web and early desktop support. This was a very important version as it introduced Dart 2.0 with null safety. This made development safer (but broke existing code). In 2021, Flutter 2.5 was released with Material Design support. In 2022 Flutter 3 was introduced with full desktop support and iOS Objective-C and Swift interop.

There have been many attempts at cross-platform frameworks, many of which have failed. Java was supposed to be the one language to rule them all but never quite achieved it. Since then, there have been other attempts such as React Native (by *Facebook*) and Xamarin (by *Microsoft*). React Native is still in use but Microsoft has ended support for Xamarin and now uses .Net MAUI. React Native uses HTML web pages and Xamarin uses C#.

Flutter architecture

The Flutter framework is made up of several layers. Each layer can be replaced if needed but does not have any special access to the layer below it. The top-level layer is the Framework. This is written in Dart. The entire Flutter framework looks like the following figure:

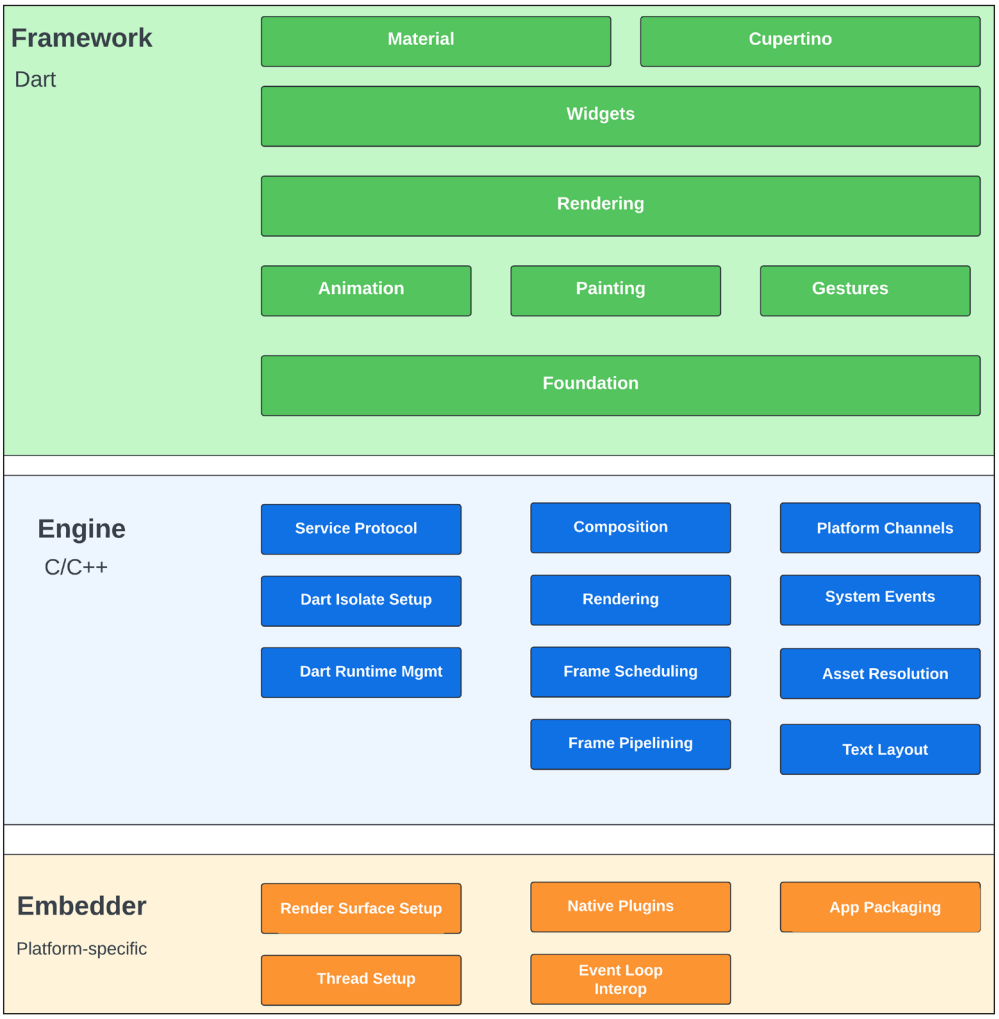

***Figure 1.1**: Flutter architecture*

This layer contains the widgets you will be using/creating. The Framework layer contains many different common widgets that you will use as well as the rendering code needed to display those widgets. The **`Material`** and **`Cupertino`** libraries are for *Google* and *Apple's UI* look and feel. The Rendering, Animation, Painting, and Foundation are lower-level Dart codes that form the essential drawing framework.

The next layer is the Engine and is written in C/C++. This is not a layer you need to use. The final layer is the Embedder and is different for each platform. One of the nice features of Flutter is that each application is compiled into native code just like any other application on that platform. This means that applications run as fast as native applications (or faster).

Note: That there is a small overhead for the built-in Flutter code.

For the engine, Flutter was originally built using the Skia graphics drawing system. The Flutter team is moving this system to a newer version called **Impeller**. It has been written for both iOS and Android and will be coming to other platforms later.

If you do not find the functionality you need in the Flutter framework, you can use many third-party plugins and packages. Packages are code written in Dart that provide extra functionality. Plugins are built with lower-level code for each platform. For example, a plugin would need separate code for Android, iOS, macOS, Windows, and the web.

Benefits of Flutter

One of the main benefits of Flutter is its rapid development time. This happens in several different areas. The first one is the declarative UI that Flutter uses, which allows you to use widgets that contain other widgets and only update those that have changed. The killer feature of Flutter is its hot reload feature. This allows you to start the application on the platform you are working on and make almost any change. Click the **Hot reload** button, and your application will be updated. You can add new packages, add new screens, change your logic, and just have the application instantly update. Adding a new plugin will have you restart your program, but you can normally do most of your application development in one session.

Unlike many other cross-platform systems, Flutter has native performance as it is compiled into native code.

Flutter is also open source, so it can be examined and even improved. The entire source code is available to see how everything works.

More importantly, Flutter can build for any platform. This makes development faster as you only have one code base. It also reduces the number of engineers needed on a project. Flutter can also be developed for the web (although that is still a work in progress).

For the web, you can output your application using HTML, a canvas-based system, or **WebAssembly** (**Wasm**). The canvas-based system is faster but makes it so that systems like *Google search* cannot index your web pages. This is bad for **search engine optimization** (**SEO**). This means *Google* will not be able to find and index individual pages. This problem is currently being worked on.

Flutter's language: Dart

Flutter uses the Dart language which was developed by *Google*. Since it was originally developed for the web, it is similar to JavaScript. The syntax is very similar with variables being defined with the var keyword and both having async/await keywords. *Google* has made a lot of improvements to the language, especially the 2.0 version with null safety. Null safety solves many inadvertent crashes. The language is object-oriented, uses classes, and is garbage-collected (which just means that it handles all allocation/disposal of classes in memory). The language was released in 2013 and Flutter 2.0 was released in 2018. They then introduced the dart2native tool to create native code. This means that you do not

have to have the Dart SDK installed on a machine to run applications (unlike Java, which needs a JDK or JRE to run). Dart 3.0 was released in May 2023 with null safety, records, patterns, and class modifiers. There is also work being done for Wasm, but it is not finished yet. Wasm is a binary instruction format designed to run code at near-native speed in web browsers, complementing JavaScript. It enables developers to compile high-performance applications written in various languages (like C++, Rust, or Go) and run them efficiently on the web, expanding the possibilities for web applications beyond what JavaScript alone could traditionally handle.

Flutter is a single-threaded toolkit with a main thread. Dart provides a way to create asynchronous methods using the async and await keywords. This is for running methods in a semi-asynchronous way (they are put in a queue to be run on the main thread). To get true multi-threading, you need to use Dart's isolates. This is the way to truly run things on another thread. They are called **isolates** because they are totally isolated from any other thread. You cannot share variables or anything in memory. You can pass data back and forth with send and receive ports, which are part of the Dart language. Isolates are pretty advanced. There are a few methods that will allow you to write simple isolates but to pass data back and forth is difficult.

Installing Flutter SDK

To get started with Flutter you will want to download the Flutter SDK.

1. Go to **https://docs.flutter.dev/get-started/install** and follow the instructions for the desktop platform you are on.
2. Find the Install the Flutter SDK section of the instructions and download the zip.
3. Unzip the files and put them in an easy-to-find location on your computer. Make sure the path is simple and does not contain any spaces or special characters. A good example is *flutter*. Next, you will need to add the flutter installation to your path.

Windows

For Windows, go to **Advanced System Settings | Advanced | Environment Variables.** In the **User variables** section edit the **Path** entry. Double-click on an empty row and type the path to Flutter. For example: **`C:\flutter`**. Click on the **OK** buttons until you are out of all dialogs.

macOS

For macOS, you will need to update your path by opening or creating a file called **`~/.zshenv`**. Add the following line:

```
export PATH=$HOME/flutter/bin:$PATH
```

When you have installed Flutter in a folder called **`flutter`** you can save the file.

CocoaPods

iOS and MacOS applications usually use CocoaPods for library management (although that is changing). This tool is used for managing iOS and MacOS libraries. For Flutter you will need this tool as it generates a library needed to run Flutter on these platforms. CocoaPods is located at: **https://cocoapods.org/**. There are two ways to install it. Since the Mac has several tools already installed (ruby and gems), you can simply install it from the command line with this line (**you will need your password for this)**:

```
sudo gem install cocopods
```

Another alternative is the Homebrew package manager. This is a very popular package manager that allows you to easily install other command line programs. This makes it very simple to install command line software on a Mac. You can find this at: **https://brew.sh/**. After installing it, use this:

```
brew install cocopods
```

Flutter Doctor

A really good tool to use is Flutter itself. There are several flutter tools available. The most important one is Flutter doctor. This tool does a diagnostic on Flutter and will let you know if you need to install any additional tools or fix a problem. From a command line type:

```
flutter doctor
```

You should see something like this (refer to the following figure):

```
flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.19.6, on macOS 14.2.1 23C71 darwin-arm64, locale en-US)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Xcode - develop for iOS and macOS (Xcode 15.1)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2022.3)
[✓] VS Code (version 1.88.1)
[✓] Connected device (4 available)
[✓] Network resources
```

***Figure 1.2**: Flutter doctor*

If there are any errors, follow the directions for fixing them.

Development application

When developing a Flutter application, you can use any text editor you want, but if you want to be more productive, you can use an **integrated development environment** (**IDE**). It will help you organize your code and make it easy to run and test. There are several IDEs that can run Flutter applications.