Building Cross-Platform Apps with Flutter and Dart

Build scalable apps for Android, iOS, and web from a single codebase

Deven Joshi

www.bpbonline.com

First published: 2023

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-89423-570

www.bpbonline.com

Dedicated to

My beloved family & friends,
who keep pushing me on.

About the Author

Deven Joshi is an avid app developer who loves all things mobile. Tinkering with Flutter since its early days, he is heavily involved in the Flutter community and has open-source contributions including apps, articles, videos, and packages. He has worked with several startups and companies across multiple domains helping them build out their Flutter products. He is an active Flutter evangelist speaking about the topic at various local and international conferences. He is currently a Developer Advocate at Stream and specializes in their Flutter SDKs. Based on all his contributions to Flutter, he is recognized by Google as a Google Developer Expert (GDE) in Flutter and Dart.

About the Reviewers

- **Amadi Promise** is a highly skilled Android and Flutter developer with a track record of success in building robust and scalable mobile applications. Passionate about problem-solving through coding, technical writing, and sharing knowledge through public speaking engagements. Committed to helping individuals succeed in the rapidly evolving digital landscape through education and training in digital skills."

- **Santosh Das** is an experienced software developer who provided valuable feedback on this book. With expertise in Flutter technology and 5 years of experience in software development, he ensured that the book met high standards for technical accuracy, clarity, and relevance. He holds a bachelor's degree in computer science and technology from Gujarat Technological University, and this book marks his debut as a technical reviewer. Santosh's commitment to staying up to date with the latest developments in Flutter makes him a valuable resource for the technical community.

Acknowledgement

I want to express my deepest gratitude to my family and friends for their unwavering support and encouragement throughout this book's writing.

I am also grateful to BPB Publications for their guidance and expertise in bringing this book to fruition. It was a long journey of revising this book, with valuable participation and collaboration of reviewers, technical experts, and editors.

I would also like to acknowledge the unwavering support and guidance of everyone in the Flutter community who were pivotal in my growth.

Finally, I would like to thank all the readers who have taken an interest in my book and for their support in making it a reality.

Preface

Flutter, the open-source UI toolkit developed by Google has gained immense popularity among developers due to its ability to create stunning, fast, and fluid user interfaces. By using a single codebase, Flutter enables you to build applications that feel native to each platform, offering a consistent user experience regardless of the device. With its powerful features and extensive Widget library, Flutter provides an efficient and productive environment for building complex applications in record time.

Accompanying Flutter is Dart, a modern, object-oriented programming language designed to optimize the development process. Dart combines the best aspects of familiar programming languages, offering a concise syntax, strong type system, and advanced features such as asynchronous programming and reactive programming patterns. As the primary language for Flutter development, Dart brings efficiency and elegance to your codebase, allowing you to write expressive and maintainable code.

This book curates a comprehensive journey through the world of cross-platform development with Flutter and Dart. Whether you are a beginner or an experienced developer, the aim is to provide you with the knowledge and practical skills necessary to build robust, scalable, and visually appealing applications. From setting up your development environment to exploring advanced topics such as state management, animations, and testing, each chapter is crafted to deliver hands-on insights and real-world examples.

Chapter 1: An Overview of Dart- provides a general introduction to the Dart language. It also provides the first look into the language and talks about its history. The evolution of the language over time is also shown.

Chapter 2: Data Types- introduces the various built-in data types offered by Dart, such as numbers, strings, booleans, lists, and maps. It also explores the characteristics and usage of each data type, highlights their strengths and will guide you on when and how to employ them in your code. The chapter also talks about null-safety and deals with nullable and non-nullable variables.

Chapter 3: Conditionals and Loops- explores conditional statements (if, else if, else) and loops (while, do-while, for, for-each) in Dart. These constructs provide powerful tools for controlling program flow and executing code based on conditions or for a specific number of iterations.

Chapter 4: Functions and Classes- explores Functions and Classes in Dart. Goes into various kinds of functions with respect to parameters as well as extension functions. The chapter also dives into the nuances of classes and creating a hierarchy.

Chapter 5: Operators- talks about using operators effectively in your Dart code, enabling you to manipulate and evaluate data with precision and efficiency. The chapter goes through the operators available in the Dart language.

Chapter 6: Asynchronous Programming- explores the power of asynchronous programming in Dart. Asynchronous programming allows you to execute concurrent and non-blocking operations, enabling your applications to handle time-consuming tasks without blocking the user interface. This chapter dives into concepts such as futures, async, await, and streams, equipping you with the knowledge and techniques to write efficient and responsive code.

Chapter 7: Why Flutter?- delves into the unique features and advantages of Flutter that makes it stand out among other frameworks. Additionally, we discuss the extensive widget library, the vibrant Flutter community, and the ability to build high-performance applications with a single codebase. By the end, you will have a clear understanding of why Flutter is a powerful framework for creating stunning cross-platform applications.

Chapter 8: Installing Flutter- guides you through the process of installing Flutter, the open-source UI framework for cross-platform app development. This chapter provides step-by-step instructions for setting up Flutter on your preferred operating system. It also covers the installation of necessary dependencies, configuring the Flutter SDK, and setting up the development environment.

Chapter 9: Flutter Project Structure and Package Ecosystem- explores the project structure and package ecosystem in Flutter, providing insights into organizing and managing your Flutter projects effectively. This chapter examines the essential directories and files that make up a Flutter project. It also delves into the Flutter package ecosystem, which offers a wide range of pre-built packages and libraries to enhance your app development process and talks about leveraging packages from the official Flutter package repository, as well as how to manage dependencies using Flutter's package manager, pub.dev.

Chapter 10: Diving into Widgets- explores the different types of Widgets, including stateless and stateful Widgets, and their role in creating interactive user interfaces. This chapter discusses the Widget tree and how Widgets are composed hierarchically to form the UI structure. You will also learn about the Widget lifecycle, handling user interactions, and updating UI elements based on state changes.

Chapter 11: Basic Widgets and Layouts- delves into Flutter's extensive Widget library, showcasing commonly used widgets and demonstrating how to customize and combine them to create visually appealing and functional interfaces.

Chapter 12: Networking in Flutter- covers the various techniques and tools available in Flutter for making HTTP requests, handling responses, and managing network connectivity. This chapter also discusses techniques for handling asynchronous operations during network requests, ensuring your application remains responsive while data is being fetched or uploaded.

Chapter 13: Local Data Persistence- delves into the realm of local data persistence in Flutter, enabling you to store and retrieve data locally on the user's device. This chapter explores various techniques and mechanisms for persisting data, ensuring that your application can retain and access information even when offline or between app sessions. We will also look into the usage of local databases, such as SQLite, in Flutter for managing structured data and performing advanced data manipulation operations.

Chapter 14: Theming, Navigation, and State Management- begins by diving into theming, which allows you to customize the visual appearance and styling of your Flutter application. This chapter delves into navigation techniques in Flutter, enabling you to create intuitive app flows and handle user interactions. It also discusses state management - a crucial aspect of building scalable and maintainable Flutter applications. We explore different state management approaches, including Provider and BLoC, empowering you to manage and update app state efficiently, resulting in responsive and interactive user experiences.

Chapter 15: Advanced Flutter-Animations- explores the realm of advanced Flutter animations, empowering you to bring your applications to life with fluid and captivating motion. This chapter talks about the fundamentals of animations in Flutter and goes into various kinds of animations such as implicit and explicit animations.

Chapter 16: Advanced Flutter – Under the Hood- breaks down Widgets to their fundamental building blocks including Elements and RenderObjects. This chapter talks about the various trees of Flutter and how Widgets work internally.

Chapter 17: Writing Tests in Flutter- discusses the different types of tests in Flutter, including unit tests, widget tests, and golden tests, and provide guidance on when and how to use each type effectively.

Chapter 18: Popular Flutter Packages- explores a selection of popular Flutter packages that extend the capabilities of the framework and enable you to build feature-rich and robust applications. This chapter highlights a variety of packages across different categories, each offering unique functionalities and solutions to common development challenges.

Chapter 19: Deploying Applications- explores the process of deploying Flutter applications, ensuring that your creations reach users on various platforms. This chapter goes into the deployment options available for different target platforms, including iOS, Android, and the web. We will also discuss platform-specific deployment considerations, such as submitting apps to the Apple App Store and Google Play Store, as well as the other respective 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/szxebry

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Building-Cross-Platform-Apps-with-Flutter-and-Dart**. 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.

Table of Contents

Kup książkę

Kup książkę

CHAPTER 1
An Overview of Dart

The plethora of programming languages available in the world is often the reason potential developers are confounded when a framework opts to go for a relatively unknown language in the mainstream development world. However, Dart was chosen for a very specific feature set that gives Flutter an edge over existing application development frameworks, and features that we will explore in detail in due time. As I often see it, Dart is a language that offers features from all over the programming sphere without seeming unfamiliar. If you are familiar with any major programming language, Dart should not seem very new, albeit with a few surprises here and there.

Structure

In this chapter, the following topics will be learned:

- History of Dart
- About Dart
- Working of Dart in Flutter
- Hello, World!
- The evolution of the Dart language
- What will we learn about Dart?

Objectives

After studying this chapter, you should gain a greater understanding of Dart's origin and why it is used in Flutter.

History of Dart

Dart is a comparatively recent language, debuting in 2011 and 1.0 released in 2013. However, people often point out correctly that newer languages like Swift came out after Dart and are widely used, whereas Dart is not. The reason for that pertains more to the original purpose of the language than its feature set. Dart originally served a very different purpose, with Google pitching it as a replacement for JavaScript. Dart can be directly compiled to JavaScript using the **`dart2js`** compiler.

However, Dart failed to catch on to its original purpose and became neglected, even going to the top of lists such as *Worst programming languages to learn this year* on blogging sites. Again, not because of the language and semantics, but how little it was used outside Google. Then, when very few expected the Dart language to take off, Flutter arrived and brought a storm of developers to try it out (fast-forward to now, and Dart and Flutter are some of the fastest-growing requirements on the market). This influx of new developers brought more rapid changes and new features to the language. Dart was not explicitly designed with Flutter in mind, however, it was a good fit when several languages were compared. *Figure 1.1* illustrates the logo of Dart:

***Figure 1.1**: The Dart logo*

I remember giving Dart a try in 2013/14 (my memory is a bit foggy on this one) – when Flutter likely existed in a conceptual stage, if at all. They had examples like creating a pirate badge with Dart on the website, which I followed along just for fun - not expecting I would have anything to do with the language in the future since I was into native Android development, which was something geared towards the web. I did not even faintly expect that a language I did just because of an obscure blog post would be my primary programming language almost a decade later.

Funnily enough, the next time I heard of Dart was when I happened to run across the alpha version of Flutter almost 2-3 years later. Even then, Flutter was not nearly as developed as it is today. The tooling is the most annoying thing because the code structure of Flutter was quite different from the Java/XML I was used to. But after that, I just found a flow that I had not in other mobile frameworks – and Dart felt instantly familiar, obviously partly from my earlier exposure, but more because it was designed to be. Almost overnight, I could develop ideas in hours instead of days or weeks. Mobile development was not the same for me – and I clung to Flutter and Dart because I knew they were the next big things.

Moving on from the memory lane to something more objective.

About Dart

Dart is open-source, object-oriented, and statically typed (2.x). Most features of the language should be familiar to most people, with some added features like mixins, and syntactic sugar designed to make common tasks in development easier.

Here are a few reasons Flutter chose Dart in particular:

- **Easier to learn**: Dart does not require an exorbitant time required to learn. Therefore, developers can focus on the Flutter framework instead of spending time adapting their existing knowledge to fit the new language and semantic constraints.

- **Can be Ahead-Of-Time (AOT) or Just-In-Time (JIT) compiled according to need**: AOT compilation eliminates the need for code to be compiled every time it is run, leading to faster start-up times. This is effective when apps are installed on a user's device, leading to much quicker app launches.

  AOT, however, leads to slower development times when the code is changed or updated. This is when JIT compilation makes for a pleasant development experience offering easy and quick code changes. Dart uses JIT in development and AOT in production apps, the best of both worlds. The JIT technique allows for the stateful hot reload Flutter is famous for. (More to come in later chapters).

  Here is a small tidbit for Android lovers: If you ever had a phone with Android Jellybean and, subsequently, Android KitKat, you may have noticed that KitKat apps opened way quicker but required more time for installation compared to Jellybean. This was due to the new **Android Runtime** (**ART**) being implemented. ART leveraged AOT compilation – hence taking extra time but didn't need to do any extra work when opening an app, therefore the much faster loading times. If you are young enough not to know Android KitKat and Jellybean, you sincerely make me feel old.

- **Eliminates the need for a declarative language similar to XML (Android) or JSX (React)**: Declarative layout languages often add a lot of code to the application by defining a separate language for UI and code. For example, in Android, before Kotlin came along, developers needed to get references to views before using them, leading to unnecessary steps that can be attributed to context-switching. However, Dart allows Flutter to declare UI alongside normal code, making several common tasks like creating lists easier than equivalents on Android. *Figure 1.2* describes the contents of a Flutter app when it is in development:

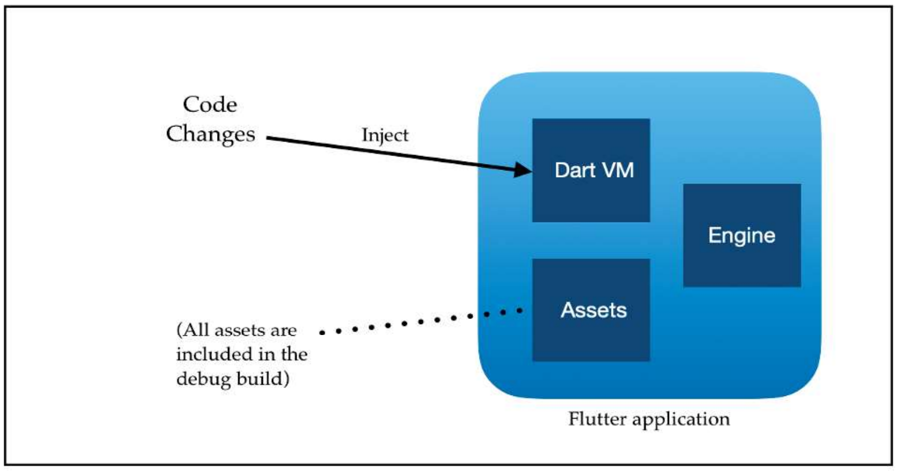

***Figure 1.2**: Running a Flutter app in development.*

- **Null-safe**: Null-safety is a favourite feature of mobile development folks since it drastically reduces errors when creating a mobile application. While we talk about this a bit more later, it means that you explicitly need to tell Dart what data in your app can and cannot be null. Dart will enforce that you do not allow null values or explicitly check for them. Over time, at least in my experience, this saves hundreds of hours on a single project.

  While null safety is now implemented in Dart, this was not the case when it originally chose Dart. Implementing null safety later was a significant effort by the Dart team and the entire community in unison since all Dart and Flutter packages needed to be upgraded. Not having null-safety was one of, if not the largest complaint that Android/iOS developers had coming to Flutter since languages like Kotlin provided immense relief to the **`NullPointerException`** plagued Java developers. They did not want to give up that luxury – which, if you can completely understand even if you have developed a single Java Android project to completion.

Alongside Flutter, **AngularDart** also allowed the creation of web apps using Dart. In 2019, Flutter also announced its move to the web with a project named *Hummingbird* — Flutter for the web. Flutter Web is now stable alongside many desktop platforms such as Windows, MacOS, and Linux. After Flutter Web's development, Google's focus on the web seems to have shifted a bit away from AngularDart and now recommends Flutter for developing web projects.