Flutter and Dart

Up and Running

Build native apps for both iOS and Android using a single codebase

Dr. Deepti Chopra
Roopal Khurana

bpb

Table of Contents

1. Introduction to Flutter
- Introduction
- Structure
- Objectives
- Introduction to Flutter
- Features of Flutter
- Architecture of Flutter
- Advantages of Flutter
- Disadvantages of Flutter
- Capabilities that make Flutter more powerful over other tools
- Conclusion
- Questions

2. Installing the Flutter SDK
- Introduction
- Structure
- Objectives
- Setting up environment for Flutter
  - *Installation of Flutter SDK on Windows*
    - *Update the Path*
    - *Run Flutter Doctor*
    - *Setting up Android Device*
    - *Set up Android Emulator-*
    - *Licenses of Android SDK Platform-*
    - *Setting up an Editor*
  - *Installing Flutter and Dart Plugins*
    - *Linux or Windows*
  - *Validating the setup with the Flutter Doctor*
- Tools for Building Flutter Based Application
- Conclusion
- Questions

3. Introduction to Dart
- Introduction
- Structure
- Objectives
- Introduction to Dart
- Need for Dart Coding for Flutter
- Declaring and referencing variables
- Operators in Dart
  - *Arithmetic operators*
  - *Relational Operators*
  - *Type test operators*
  - *Bitwise operators*
  - *Assignment Operators*
  - *Logical operators*
  - *Conditional operators*
  - *Cascade Notation Operator*
- Control statements in Dart
  - *Break statement*
  - *Continue statement*
  - *Decision making statements*
    - *if Statement*
    - *if…else Statement*
    - *else…if… Statement*
    - *Nested if Statement*
    - *Switch*
    - *For Loop*
    - *While Loop*
    - *Do-while loop*
- Conclusion
- Questions

4. Classes and Functions in Dart
- Introduction
- Structure
- Objectives
- Classes and Functions in Dart
  - *Declaration of class*

Creating an instance of the class
Dart Constructors
Named Constructors
The 'this' Keyword
Getters and Setters in Dart class
Class Inheritance
Types of Inheritance
Class Inheritance and Method Overriding
The static keyword
The Super Keyword
Import Packages
Dart Package Manager
Dart Libraries
Importing a library
Name alias of Library
Implementing Asynchronous Programming
Conclusion
Questions

5. Introduction to Widgets
Introduction
Structure
Objectives
Creating files and folders using widgets
Installation of File Manager
Structuring Widgets
Understanding Widget Tree
Inherited Widget
Conclusion
Questions

6. Using Common Widgets
Introduction
Structure
Objectives
Using common widgets
Adding Animation to App

AnimatedAlign Widget
AnimatedBuilder Widget
Performance optimizations
AnimatedBuilder
AnimatedContainer Widget
AnimatedCrossFade Widget
AnimatedDefaultTextStyle Widget
AnimatedListState class Null safety
AnimatedModalBarrier class Null safety
AnimatedPhysicalModel Widget
AnimatedPositioned Widget
Code-based animations
Drawing-based animations
Creating an App's Navigation
Build two routes
Use Navigator.push() to navigate to second route
Use Navigator.pop() method to return to the first route
Conclusion
Questions

7. Building Flutter Application
Introduction
Structure
Objectives
Building Flutter application using Android specific code
Install Flutter and set up an editor
Create a new Flutter project
Designing the app's UI
Implement app logic
Test the app
Deploy the app
Implement the platform channel on Android
Invoke the platform channel from Flutter
Handle the method call on Android
Return a result from Android
Building Flutter application using iOS specific code
Conclusion

Questions

8. Introduction to Packages
Introduction
Structure
Objectives
Type of packages using Dart Packages
Types of Packages
Library packages
Application packages
Command-line packages
Plugin packages
Experimental packages
Web
Mobile
Desktop
Testing
Utilities
Dart Package
Generic dart code
Flutter Plugin
Develop a Flutter Plugin Package
Create a new package
Define the plugin's API
Add platform-specific code
Test the plugin
Publish the plugin
Dart Package Manager
Installing a Package
Flutter Plugin Package
Steps for developing Dart packages
Creating a package
Implementing the package
Publishing packages
Conclusion
Questions

9. Building Layouts
- Introduction
- Structure
- Objectives
- Introduction to Layout
  - *Container*
  - *Row and Column*
  - *Expanded*
  - *ListView*
  - *Stack*
  - *Card*
  - *Creating a text, image or icon*
    - *Selection of a layout widget*
    - *Creation of a Visible Widget*
    - *Addition of layout widget to page*
    - *Material apps*
- Type of Layout Widgets
  - *Single Child Widgets*
- Layout Application
  - *Padding in Flutter*
  - *Align Widget*
  - *Container*
  - *Stacks*
  - *GridView*
  - *Scaffold*
- Conclusion
- Questions

10. Flutter Database Concepts
- Introduction
- Structure
- Objectives
- SQLite
- Adding Firebase
  - *Adding Firestore Backend*
- JSON, XML, HTML
- Hive

Moor
Conclusion
Questions

Index

CHAPTER 1

Introduction to Flutter

Introduction

Flutter Framework is used to create simple and efficient mobile applications. Only a single codebase is required to develop a mobile application across different platforms such as Android and iOS. This chapter will give an introduction to Flutter. It will discuss in detail why Flutter is required to develop sophisticated and elegant mobile applications. It would give insight into the architecture of Flutter Framework, its pros and cons and why Flutter proves to be better option for designing mobile applications compared to the other mobile application frameworks.

Structure

In this chapter, we will cover the following topics:

- Introduction to Flutter
- Features of Flutter
- Architecture of Flutter Framework
- Advantages of Flutter
- Disadvantages of Flutter
- Capabilities that make Flutter more preferred compared to other frameworks

Objectives

This chapter aims to give a basic introduction to Flutter Framework that may be used to develop exciting mobile applications. From this chapter, readers will be able to know about the architecture of Flutter Framework in detail, the characteristics of Flutter, its positive and negative aspects and also know reasons what makes it one of the most preferable mobile application frameworks for developing mobile applications compared to the other competitive frameworks. We shall conclude the chapter with some examples that can be implemented with Flutter Framework in later chapters.

Introduction to Flutter

A Flutter may be referred to as a high performance and simple framework that is used along with Dart language to develop mobile application. Flutter framework supports both Android and iOS. In order to create modern applications, Flutter provides many useful and simple widgets that can be used for designing mobile applications easily. Gestures and Animations are supported by these widgets. Building Mobile Applications using Flutter is based on Reactive programming. Widget may be represented using state. If there are some changes made in Widget, then as a part of Reactive Programming, there is a comparison between the old and the new state. Then, instead of re-rendering the whole mobile application, only the changes that have been made in new state as compared to the old state are rendered in the mobile application. The logo of Flutter is given in *Figure 1.1*:

Figure 1.1: *Logo of Flutter*

Figure 1.2 shows logo of Flutter and Dart. Dart is a programming language that helps to build applications using Flutter framework. An example of Dart programming for building mobile application is shown in *Figure 1.2*:

Figure 1.2: *Logo of Dart and Flutter*

In 2019, Google launched Dart 2.5 and Flutter 1.9. Flutter 1.9 helps in building web, desktop and mobile applications using single codebase. Please refer to the following *Figure 1.3*:

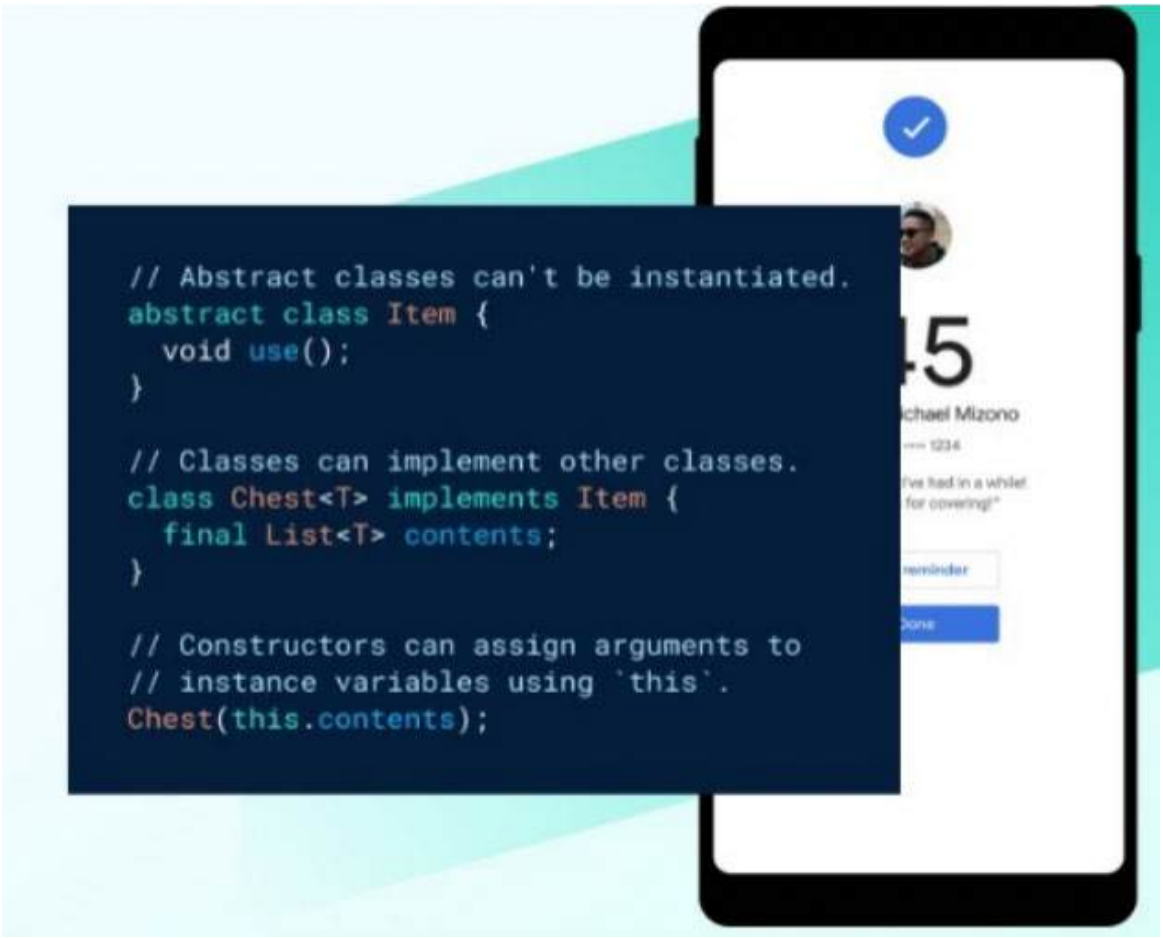

Figure 1.3: *Dart Programming used for developing Mobile App*

Features of Flutter

Flutter is a mobile application framework that offers many features to the developers. These include the following:

- **Reactive and Modern Framework**

  During development of Mobile application, the Flutter Mobile Application run in VM environment that provides hot reload of stateful changes without the need to recompile the entire code. Developers are able to interact with Flutter with the help of Flutter framework and they provide a proper mapping between a given application state to the interface state. When there are some changes in the application state, Flutter performs the task of modification of Interface. Flutter is a reactive and modern framework written in Dart language.

- **Simple and High-Performance Application**

  Flutter based applications are simple to write and they deliver high level performance when common pitfalls are avoided. The following pitfalls may be avoided while building Flutter application:

  1. While working on building Animation based application, we should avoid using Opacity widget.
  2. Clipping during building Animation based application should be avoided.

- **Supports easy to understand Dart Language**

  Dart is a programming language whose constructs are similar to C and is used for making mobile applications using Flutter. Dart can also be used for developing web applications that will run on all the web browsers. Dart has predefined libraries which are present in Dart SDK. Some of the commonly used Dart libraries include the following:

  1. dart:core-

     This library is imported in all the files. It defines the core functionality.
  2. dart:async-

     This library is used for performing asynchronous programming.
  3. dart:math-

     This library defines mathematical constants and functions.

4. dart:convert

   This library is used for conversion between different data representations.

- **Fast Development**

  Flutter uses Dart language for creating fast mobile, web and desktop applications on any platform using a single codebase. This is referred to as hot reload feature. Flutter's hot reload feature helps to quickly do updates, remove errors and add new features.

- **Supports large Widget Catalog**

  Flutter is used to create fast and pleasing apps with the help of its large collection of interactive, structural, platform and visual widget catalog.

- **Supports same UI for different platforms**
- **Generates beautiful UI**

Architecture of Flutter

Flutter framework is designed in such a way that single code may be used to develop mobile application in both Android and iOS. Flutter architecture is displayed in *Figure 1.4*. It may be explained in the following sections:

- **The layer model**: It comprises of the components from which flutter is created.
- **Reactive user interfaces**: It involves user interface development in Flutter.
- **Widgets**: Widgets are the building blocks of user interfaces in Flutter.
- **Rendering steps**: This process involves conversion of UI process into pixels.
- **Platform embedders**: It is the code that allows execution of flutter apps on operating system of desktop and mobile.
- **Combining other code and Flutter**: Other techniques may be added in a Flutter app.
- **Web Support**: It involves behavior of Flutter in browser environment.

Flutter comprises of a layered system in which independent and individual libraries depend on the underlying layer. No layer can have privileged access to the other layers present under it. Please refer to the following figure: