



Beginning App Development with Flutter

Create Cross-Platform Mobile Apps

Rap Payne

Apress®



Beginning App Development with Flutter

**Create Cross-Platform
Mobile Apps**

Rap Payne

Apress®

Beginning App Development with Flutter: Create Cross-Platform Mobile Apps

Rap Payne
Dallas, TX, USA

ISBN-13 (pbk): 978-1-4842-5180-5
<https://doi.org/10.1007/978-1-4842-5181-2>

ISBN-13 (electronic): 978-1-4842-5181-2

Copyright © 2019 by Rap Payne

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Aaron Black
Development Editor: James Markham
Coordinating Editor: Jessica Vakili

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit <http://www.apress.com/rights-permissions>.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-5180-5. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

This book is dedicated to the men and women of the Flutter Community. I've never seen a group more devoted to the success of others. You're an inspiration and example to me.

*Particular thanks to these members of the community who've helped me with Flutter issues.
This Texan owes y'all!*

Andrew "Red" Brogdon (Columbus, Ohio),

Brian Egan (Montana),

Emily Fortuna (San Francisco),

Frederik Schwieger (Düsseldorf, Germany),

Jeroen "Jay" Meijer (Rotterdam, Netherlands),

Martin Rybak (New York), Martin Jeret (Estonia),

Nash Ramdial (Trinidad), Nilay Yenner (San Francisco),

Norbert Kozsir (Karlsruhe, Germany), Pooja Bhaumik (Bengaluru, India), Raouf Rahiche (Casablanca by way of Algeria), Remi Rousselet (Paris), Rohan Tanaja (Berlin),

Scott Stoll (Cleveland, Ohio),

But especially Simon Lightfoot (London), who we all call "The Flutter Whisperer" He taught me much of what I know about Flutter.

Praise for Beginning App Development with Flutter

“Rap has written a great starting guide full of information for those who are new to developing multi-platform apps with Flutter.”

—Frederik Schwieger (Düsseldorf, Germany), Organizer of the International Flutter Hackathon and creator of flutter school

“A great read! This covers everything a beginner might want to know, and more. It explains not only what Flutter is but why it exists works the way it does. It also provides great tips for common pitfalls along the way. Definitely recommended.”

—Jeroen “Jay” Meijer (Rotterdam, Netherlands),
Leader of Flutter Community Github

“Rap’s book is a great book to get started with Flutter. It covers every important topic to write your very first app but also contains valuable information for more seasoned developers.”

—Norbert Kozsir (Karlsruhe, Germany)
Flutter Community Editor

“As a non-native English speaker, I’m totally impressed by the simplicity of this book and how much I can read and understand without getting bored.”

—Raouf Rahiche (Algeria) Flutter speaker,
developer, and instructor

PRAISE FOR BEGINNING APP DEVELOPMENT WITH FLUTTER

“As an early adopter and one of the original members of the Flutter Community, Rap is one of the world’s foremost authorities on Flutter. Where documentation is written for Engineers, by Engineers, Rap is a human who (thankfully!) writes in an enjoyable style that can easily be understood by other humans.”

—Scott Stoll (Cleveland, Ohio), Contributor to the Flutter codebase and Co-founder of the Flutter Study Group

Table of Contents

About the Authorxvii

About the Technical Reviewerxix

Who is this book for?.....xxi

Part I: Introduction to Flutter..... 1

Chapter 1: Hello Flutter.....3

 What is Flutter?.....4

 Why Flutter?.....5

 The other options5

 Native solutions7

 Conclusion8

Chapter 2: Developing in Flutter9

 The Flutter toolchain10

 The Flutter SDK.....10

 IDEs10

 IDE DevTools12

 Emulators13

 Keeping the tools up to date.....15

 The Flutter development process18

 Scaffolding the app and files.....18

 Running your app21

 Conclusion27

Part II: Foundational Flutter.....29

Chapter 3: Everything Is Widgets31

 UI as code33

 Built-in Flutter widgets35

 Value widgets.....36

 Layout widgets.....36

 Navigation widgets37

 Other widgets.....38

 How to create your own stateless widgets38

 Widgets have keys.....41

 Passing a value into your widget.....42

 Stateless and Stateful widgets45

 So which one should I create?45

 Conclusion46

Chapter 4: Value Widgets.....47

 The Text widget.....47

 The Icon widget48

 The Image widget49

 Embedded images.....50

 Network images51

 Sizing an image51

 Input widgets54

 Text fields55

 Putting the form widgets together65

 Form widget65

FormField widget.....	67
One big Form example	71
Conclusion	76
Chapter 5: Responding to Gestures	77
Meet the button family	78
RaisedButton	80
FlatButton and IconButton.....	81
FloatingActionButton	81
CupertinoButton.....	82
Dismissible.....	83
Custom gestures for your custom widgets	83
Step 1: Decide on your gestures and behaviors	84
Step 2: Create your custom widget	85
Step 3: Add a GestureDetector widget.....	86
Step 4: Associate your gesture with its behavior	87
Example 1: Reacting to a long press.....	87
Example 2: Pinching to add a new item.....	89
Example 3: Swiping left or right.....	90
What if there are two or more gestures happening at the same time?	92
Conclusion	92
Chapter 6: Laying Out Your Widgets	93
Laying out the whole scene	100
MaterialApp widget	100
The Scaffold widget.....	101
The AppBar widget	102
SafeArea widget	104
SnackBar widget	105

TABLE OF CONTENTS

How Flutter decides on a widget’s size 106

 The dreaded “unbounded height” error..... 107

 Flutter’s layout algorithm 108

Putting widgets next to or below others 110

Your widgets will never fit! 113

What if there’s extra space left over? 113

 mainAxisAlignment..... 113

 crossAxisAlignment 115

 Expanded widget..... 117

What if there’s not enough space? 121

 The ListView widget 121

Container widget and the box model 124

 Alignment and positioning within a Container..... 126

 So how do you determine the size of a Container? 128

Special layout widgets 130

 Stack widget..... 130

 GridView widget 131

 The Table widget 134

Conclusion 137

Chapter 7: Navigation and Routing 139

Stack navigation 140

 Navigating forward and back 141

 Get result after a scene is closed 143

Drawer navigation..... 144

 The Drawer widget 146

 Filling the drawer 148

Tab Navigation	150
TabController	151
TabBarView	151
TabBar and Tabs	152
TabBar at the bottom	153
The Dialog widget	153
showDialog() and AlertDialog	154
Responses with a Dialog	155
Navigation methods can be combined	157
Chapter 8: Styling Your Widgets	159
Thinking in Flutter Styles	160
A word about colors	161
Styling Text	163
TextStyle	163
Custom fonts	165
Container decorations	168
Border	170
BorderRadius	172
BoxShape	173
Stacking widgets	176
Positioned widget	178
Card widget	180
Themes	181
Applying theme properties	183
Conclusion	186

TABLE OF CONTENTS

Chapter 9: Managing State187

What is state? 187

What goes in a StatefulWidget? 189

The most important rule about state! 190

Passing state down..... 191

Lifting state back up 192

An example of state management 193

When should we use state? 198

Advanced state management 200

 InheritedWidget 200

 BLoC 200

 ScopedModel..... 201

 Hooks..... 201

 Provider 202

 Redux 202

 Whoa! That’s a lot of packages! 203

Conclusion 203

Part III: Above and Beyond205

Chapter 10: Your Flutter App Can Work with Files207

Including libraries in your Flutter app 208

 Finding a library 208

 Adding it to pubspec.yaml 210

 Importing the library 210

 Using the library 211

Futures, async, and await 211

 Why would it wait? 212

 How do we get the data from a Future? 213

await.....	214
async	215
Including a file with your app.....	216
Writing a file.....	218
And reading it!	219
Using JSON	220
Writing your app's memory to JSON	221
Reading JSON into your app's memory	222
Shared preferences.....	223
To write preferences.....	224
To read preferences.....	224
Conclusion	225
Chapter 11: Making RESTful API Calls with HTTP	227
What is an <i>API</i> call?	228
The flavors of API requests	228
Making an HTTP GET or DELETE request	230
Making an HTTP PUT, POST, or PATCH request.....	231
HTTP responses to widgets.....	232
Brute force – The easy way	233
FutureBuilder – The clean way	234
Strongly typed classes.....	238
Create a business class.....	238
Write a <code>.fromJson()</code> method	239
Use <code>.fromJson()</code> to hydrate the object.....	240
One big example	240
Setting up	242
Create the Flutter app.....	243

TABLE OF CONTENTS

Making a strongly typed business class.....	243
PeopleList.dart.....	244
A GET request in Flutter.....	247
A DELETE request in Flutter.....	247
PeopleUpsert.dart.....	248
A POST and PUT request in Flutter	252
Conclusion	254
Chapter 12: Using Firebase with Flutter	255
Introducing Firebase	256
Cloud Firestore	257
Cloud Functions.....	258
Authentication	259
Setting up Firebase itself	259
(1) Creating a Firebase project	260
(2) Creating the database	263
(3) Creating an iOS app.....	267
(4) Creating an Android app.....	273
(5) Adding FlutterFire plugins	277
Using Firestore.....	278
To get a collection	279
To query.....	281
To upsert.....	281
To delete	282
Where to go from here	283

Appendix A: Dart Language Overview	287
What is Dart?	287
Expected features – Dart Cheatsheet	288
Data types	288
Arrays/lists	289
Conditional expressions	289
Looping.....	290
Classes.....	290
Class constructors.....	291
Unexpected things about Dart.....	291
Type inference	292
final and const.....	292
Variables are initialized to null	293
String interpolation with \$.....	294
Multiline strings.....	294
Spread operator.....	294
Map<foo, bar>	295
Functions are objects	295
Big arrow/Fat arrow	296
Named function parameters.....	296
Omitting “new” and “this.”	297
Class constructor parameter shorthand.....	298
Private class members.....	299
Mixins	299
The cascade operator (..).....	300
No overloading	301
Named constructors.....	301
Index.....	303

About the Author



Rap Payne has focused on mobile development since he started Agile Gadgets, a mobile app development company, in 2003. He is a consultant, trainer, and entrepreneur who has written apps, mentored developers, and taught software development classes for Fortune 500 companies like Boeing, Walmart, Coca-Cola, Wells Fargo, Honda, CVS, GE, Chase, HP, Lockheed, ExxonMobil, Lowe's, Nike, J.C. Penney, USAA, and Walgreens;

government agencies like the NSA, the US Air Force, Navy, Army, NASA, Britain's GCHQ, and Canada's postal service; and several provincial governments, to name a few.

As a professional mentor and trainer, Rap has developed a talent for communicating highly complex ideas in easy-to-understand ways. And as a real-world developer, he understands the need to teach these topics using practical and realistic examples and exercises.

About the Technical Reviewer



Massimo Nardone has more than 22 years of experience in Security, Web/Mobile development, Cloud, and IT Architecture. His true IT passions are Security and Android.

He has been programming and teaching how to program with Android, Perl, PHP, Java, VB, Python, C/C++, and MySQL for more than 20 years.

He holds a Master of Science in Computing Science from the University of Salerno, Italy.

He has worked as a Project Manager, Software Engineer, Research Engineer, Chief Security Architect, Information Security Manager, PCI/SCADA Auditor, and Senior Lead IT Security/Cloud/SCADA Architect for many years.

His technical skills include Security, Android, Cloud, Java, MySQL, Drupal, Cobol, Perl, Web/Mobile development, MongoDB, D3, Joomla, Couchbase, C/C++, WebGL, Python, Pro Rails, django CMS, Jekyll, Scratch, and so on.

He works as Chief Information Security Officer (CISO) for Cargotec Oyj.

He worked as visiting lecturer and supervisor for exercises at the Networking Laboratory of the Helsinki University of Technology (Aalto University). He holds four international patents (PKI, SIP, SAML, and Proxy areas).

Who is this book for?

If you're a developer with experience in some object-oriented language like Java, C#, C++, or Objective-C and you want to create Android apps, iOS apps, or web apps with Flutter, this book is for you. It is especially important for you if you want to create an app that runs on multiple platforms and if you are new to Flutter.

If you've got some experience already with Flutter, you'll undoubtedly learn something, but we're not expecting that you have any prerequisite knowledge or experience with Flutter. All of our chapters are written with the assumption that everything in Flutter is completely new to you.

If you know anything about iOS development, Android development, or web development, that will certainly help with understanding the topics because there are lots of analogies in them for Flutter. The more you know about those things, the better, especially JavaScript and React. But if you know none of them, don't fret. They're by no means necessary.

Knowledge of the Dart language also will help. We've found that Dart has got its unique features for sure, but it is extremely easy to pick up if you understand object-oriented concepts. Heck, if you know Java or C#, most code snippets are understandable without any explanation of the language. Read a few and you'll be writing your own in no time.

At the same time, there are some unique but very cool Dart features that we consider best practices. We could have "simplified" the code for Java devs by not using these best practices, but in the long run that's not doing you any favors. Instead, we go ahead and use them, but we do explain those things in "Appendix A: Dart Language Overview." In there, we give you a cheat sheet with just enough detail to write code, followed

WHO IS THIS BOOK FOR?

by a more in-depth explanation of the features that will be unexpected by developers of other languages. Pay special attention to the section called “Unexpected things about Dart.”

What is covered?

This book teaches you how to create fully functioning and feature-rich apps that run on iOS, Android, and the Web. We do this in three sections.

Part I: Introduction to Flutter

1. **Hello Flutter** – We’re setting the stage for the book. Giving you a feel for why you’re here. What problems does Flutter solve? Why the boss would choose Flutter vs. some other solution.
2. **Developing in Flutter** – Flutter has a unique set of tools, but it isn’t always straightforward what each tool does and how to use it. This chapter guides you through the process of write-debug-test-run. We get an understanding of the tooling including installation and maintenance.

Part II: Foundational Flutter

3. **Everything Is Widgets** – Widgets are super important to Flutter since they’re the building blocks of every Flutter app. We show why and provide the motivation and basic tools to create widgets. Topics include composition, UI as code, widget types, keys, and stateless vs. stateful widgets.

4. **Value Widgets** – A deep dive into widgets that hold a value, especially user-input fields. Topics include the `pubspec.yaml` file; Text, Image, and Icon widgets; and how to create forms in Flutter.
5. **Responding to Gestures** – How to make your program do things in response to user actions like taps, swiping, pinching, and the like. We'll show you the button family and the `GestureDetector` widget.
6. **Laying Out Your Widgets** – We'll learn how to lay out a view, controlling how widgets are placed side by side and/or above and below, defining the amount of space between widgets, and aligning them vertically and horizontally.
7. **Navigation and Routing** – Navigation is making the app hide one widget and show another in response to user actions. This makes them feel like they're moving from one scene to another. We'll cover stack navigation, tab navigation, and drawer navigation.
8. **Styling Your Widgets** – Then we'll look at how to control each widget's color, borders, decorations, shapes, and other presentational characteristics. We handled light styling as we introduced each widget earlier, but this is where we answer all the questions needed to get a real-world app looking good and staying consistent throughout with themes.
9. **Managing State** – How to get data from one widget to another and how to change that data. We cover how to create `StatefulWidget`s and design them in the best way. We also provide a high-level overview of tools to handle real-world complex state management.

Part III: Above and Beyond

10. **Your Flutter App Can Work with Files** – Using libraries. Futures, async, await. Bundling files with your app. Reading and writing a file. JSON serialization.
11. **Making RESTful API Calls with Ajax** – How to read from and write to an HTTP API server. This is where we show how to make GET, POST, PUT, DELETE, and PATCH requests.
12. **Using Firebase with Flutter** – We will show you a real-world, robust cloud solution that works like a dream with Flutter. No surprise that it is also a Google offering.

What is not covered and where can I find it?

As importantly, you should know what not to expect in the book. We will not give you a primer on the Dart programming language beyond the aforementioned appendix. We simply didn't think it was the best use of your time and wanted to dive right into Flutter. If you feel you need a primer later on, go here: <https://dart.dev/guides/language/language-tour> followed by <https://dart.dev/tutorials>. We chose not to discuss deploying to the app stores. The stores already do a fine job of explaining how to submit an app. That, and the process, changes so frequently that your definitive resource ought to be the stores themselves. You'll find

instructions at <https://developer.apple.com/ios/submit/> and here: <https://play.google.com/apps/publish>. And we aren't going to cover certain advanced topics like device-specific development in iOS and Android or adding Flutter to an existing iOS/Android project. This is a beginner's book and we didn't want to overwhelm you. These and so many other topics can be found on the Web by searching and through some of the other resources we'll point you to in the last chapter of book.

PART I

Introduction to Flutter

CHAPTER 1

Hello Flutter

Picture this in your mind's eye. You are the superintelligent and capable CEO of a new business. Obviously your mission is to maximize sales while minimizing expenses. “Hmmm,” you think. “I can really increase sales if I make our products available on the Web.” So you ask your friends how to create a web app and they say ...

“You need to hire a web developer. They should know HTML, CSS, JavaScript, and probably some framework like React, Vue, or Angular.”

It's expensive but you do it and your gamble pays off. Sales increase markedly. Trying to keep on top of demand, you monitor social media and engage your customers. You hear them say that this web app is great and all but “We'd have been here earlier if you had an app in the App Store.” So you talk to your team who, while being experts in the Web, are not iOS developers. They tell you ...

“You need to hire an iOS expert. They should know iOS, Swift or Objective-C, Xcode, macOS, and CocoaPods for development.”

Your research shows that this person is even more specialized and therefore expensive than your web devs. But again, it seems to be the right thing to do, so you bite the bullet and hire them. But even while this app is being developed, you see that the feedback was not isolated to iOS apps, but instead was looking at all mobile devices. And – oh, snap! – 85% of devices worldwide run Android, not iOS. You bury your head in your hands as you ponder whether or not you can afford to ignore 85% of your potential customers. Your advisors tell you ...

“You need to hire an Android expert. They should know the Android OS, Gradle, Android SDK, XML, Android Studio, and Java or Kotlin.”

“Really?!? Another developer?”, you say. “Yes. And one just as expensive as your iOS developer,” they respond.

Isn’t there one person who can do all three things? Some way to share the code between all of those environments? Then you could hire just one person. In fact, they could write the code one time and deploy it to the Web, to the App Store, and to the Google Play Store. One codebase to maintain. One place to make improvements and upgrades. One place to squash bugs.

Ladies and gentlemen, allow me to introduce you to Flutter!

What is Flutter?

Flutter is a set of tooling that allows us to create beautiful apps that run on iOS, Android, the Web, and desktop.¹

Flutter is ...

- Free (as in free beer. No cost)
- Open source (that’s the other sense of the word “free”)
- Backed by and originated at Google
- Being enhanced and maintained by a team of developers at Google and hundreds of non-Google contributors around the globe
- Currently being used by thousands of developers in organizations across the world for production apps
- Fast because it compiles to truly native apps that don’t use crutches like WebViews and JavaScript bridges

¹Desktop is coming soon. Flutter will work on Windows, macOS, Chromebooks, and Linux.

- Written one place and compiled to a web app for billions of browsers, an iOS app for iPhones and iPads, and an Android app for all of the rest of the phones and tablets out there

Why Flutter?

Google's mission with Flutter is ...

To build a better way to develop for mobile

Notice what is not in that mission. There's no mention of Android (which is also owned by Google) nor of iOS nor of the Web. Flutter's goal is to create a better way to develop for all devices. In other words, Flutter should be better to create iOS apps than Swift. It should be better to create Android apps than Kotlin. It should be better to create web apps than HTML/JavaScript. And if you get all of those things simultaneously with one codebase, all the better.

The Flutter team has succeeded spectacularly with this mission.

As proof, Eric Seidel offers this example.² The Google CRM team used Flutter to build an internal Android app and did it **three times** faster than with their traditional Android toolchain!

But it turns out that Flutter isn't the only game in town for cross-platform. You have other options.

The other options

Cross-platform development comes in three general flavors listed in Table 1-1.

²http://bit.ly/eric_seidel_flutter_keynote_video at 21:47 in.

Table 1-1. *Cross-platform development categories*





	Some technologies	Cons	Pros
Progressive Web Apps (PWA)	HTML/CSS, React, Angular, Vue	Not a real app. Runs in a web browser. Not available in app stores. Hard to create a desktop shortcut. Cannot access many of the device’s resources like accelerometer, compass, proximity sensor, Bluetooth, NFC, and more	Easy to write
Hybrid	PhoneGap, Cordova, Sencha, Ionic	Runs in a WebView so it can be slow. Nearly impossible to share code with the web app	Easier for web devs to learn because it uses HTML and JavaScript as its language and structure
Compile-to-native solutions	React Native, NativeScript, Flutter, Xamarin	Learning a framework may be difficult. Mastering the toolchain definitely is	Real apps that can be found in the stores and run fast

If you have a captive audience, one where users value your app so much that they’re willing to accept a poorer user experience, the cheapest solution is to create a PWA. If your app is extremely naive and speed is not expected to be an issue, a hybrid solution might be appropriate. But if speed, smoothness, and sophisticated capability are important, you will need to go with a native solution.

Native solutions

As of today, there are four fairly popular compile-to-native solutions (Table 1-2).

Table 1-2. *Compile-to-native cross-platform frameworks*

	 Xamarin	 NativeScript	 React Native	 Flutter
Year introduced	2011	2014	2015	2018
Backed by	Microsoft	Telerik	Facebook	Google
Presentation language	XAML and/or xamarin.forms	Proprietary but looks like XML	Proprietary but looks like JSX	Dart
Procedural language	C#	JavaScript	JavaScript	Dart

These are all decent options. All are free to develop in and are well-tested, having many production applications created. All have been used in large organizations.

But only one has an option to create a web application in addition to the iOS and Android apps that will be deployed to the app stores – Flutter.

Flutter is the latest of these frameworks to be released. As such it has a distinct advantage of observing those that had come before. The Flutter team took note of what worked well with other frameworks and what failed. In addition, Flutter added new innovations and ideas – all baked in from the start rather than being bolted on as improvements are made.

But I suspect that if you've bought this book, you don't need much convincing so I'll stop. Suffice it to say that Flutter is amazing! It is easy to write, elegant, well-designed – an absolute pleasure to code in.³

Conclusion

Now, if you're the kind of developer I hope you are, you're chomping at the bit to get your hands dirty writing some code! So let's get to it. We'll start by installing and learning the Flutter development toolchain.

³But if you do want to read more, here's a deeper discussion of Flutter vs. some other frameworks: <http://bit.ly/2HC9Khm>

CHAPTER 2

Developing in Flutter

As we saw in the last chapter, Flutter enables us to create apps that run on the Web, on desktop computers, and on mobile devices (which seems to be the main draw). But wait a second, how exactly do we create these apps? What editor should we use? What is needed in the Flutter project? How do you compile the Dart source code? Do we need any other tools to support the project? How do you get it into a browser or on a device in order to test it out? Good questions, right?

Let's answer those questions and more in this chapter. Let's cover two significant topics:

1. Tools needed – How to install and maintain them
2. The development process – How to create the app, run it, and debug it

Caution By its nature, cross-platform app development tooling involves an awful lot of moving parts from various organizations, few of whom consult with the others before making changes. And since we're dealing with boundary-pushing and young technology, changes happen frequently. We've tried in this chapter to stick with timeless information but even it is likely to become stale eventually. Please check with the authors of these tools for the latest and greatest information.
