

Flutter開発環境 Macで作るお～！編

やる夫が
異世界へ転生
したら
モバイルアプリ
が
作れるようになつた
話

Flutter で Android、iPhone アプリ開発

— 開発環境作成 Mac 編 —

[著] 今北産業

技術書典 13（2022年夏）新刊
2022年8月29日 ver 1.0

■免責

本書は情報の提供のみを目的としています。

本書の内容を実行・適用・運用したことで何が起きようとも、それは実行・適用・運用した人自身の責任であり、著者や関係者はいかなる責任も負いません。

■商標

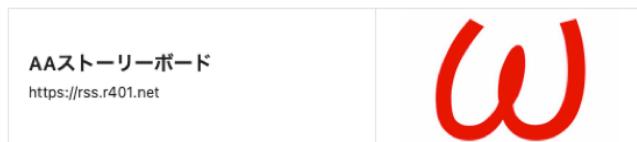
本書に登場するシステム名や製品名は、関係各社の商標または登録商標です。

また本書では、™、®、©などのマークは省略しています。

はじめに

このたびは、「Flutter 開発環境の作成-Mac 版」を、手にしていただき誠にありがとうございます。

本書で使用しているイラストは、



にて作成しています。作者の tgk 様、2ch へ AA を投稿された皆様へ感謝します。

本書は、Mac での Flutter 開発環境の作成を解説しています。CPU が Intel、Apple Silicon のどちらでも問題ありません。本書にて Flutter へ興味を持っていただけ、ひとりでも Flutter 使いが増えることを願っています。

もし、本書に間違い、誤字脱字がございましたら [サポートサイト^{*1}](#) にて issue を作成していただけると助かります。

♣ モバイルアプリケーション

あなたが毎日使っているスマートフォンですが、アプリケーションがなければなにもできません。

動画を観たり、

- ゲームをしたり、
- SNS で友人とつながったり、
- 支払い

なども、すべてアプリケーションを使います。

そのモバイルアプリケーション（以下アプリ）を無料で簡単に開発できるとしたら、いかがで

^{*1} https://github.com/risingforce9zz/book02_flutter-dev-starter-mac

しょうか？

しかも、Android 用、iPhone 用とも、ひとつの開発環境で。

その魔法のようなツールは、「Flutter」と言います。Google が 2018 年に正式版を発表したフリーでオープンソースなクロスプラットフォームアプリケーションを開発できる SDK（Software Development Kit）です。

この Flutter 開発環境を、あなたの PC に導入してみませんか？

♣ クロスプラットフォームとは？

それぞれのプラットフォームでアプリケーションを開発する場合には、ネイティブ言語と呼ばれるプログラミング言語が使用されます。

Android アプリ

Java または Kotlin

iOS アプリ、Mac 用デスクトップアプリ

Swift または ObjectiveC

Windows デスクトップアプリ

C++、C#

Web アプリ

HTML、CSS、JavaScript

しかし、Flutter ではプログラミング言語 Dart を使うことで、ひとつのソースコードから各プログラミング言語へ変換し、それぞれのプラットフォームでアプリケーションをビルドできます。

つまり、あなたは Dart が使えるようになるだけで、Android アプリ、iOS アプリ（iPhone、iPad）、Windows デスクトップアプリ、Mac デスクトップアプリ、Web アプリを開発できます。

♣ なぜ、クロスプラットフォーム開発が主流になるのか？

たとえば、iPhone 用アプリを開発する方法としては、Mac を使い、Xcode（統合開発ツール）で、プログラミング言語 ObjectiveC、または、Swift でコードを書きます。

同じアプリを Android 用としてリリースするには、Android Studio（統合開発ツール）やエディ

タを使い、プログラミング言語 Java、または、Kotlin でコードを書きます。

エミュレータか実機（Android、iPhone）でデバッグを行い、機能の実装に問題がなければリースします。

同じアプリですが、ソースコードは別々に管理しなければなりません。また、アップデート、修正もそれぞれのソースコードを変更しなければなりません。

クロスプラットフォーム開発環境では、ひとつのソースコードから、複数のプラットフォーム用のアプリケーションを開発できます。

クロスプラットフォーム開発環境で使われているのは、

- React Native -- JavaScript
- Xamarin -- C# <--本書執筆中に Xamarin のサポート終了が発表されました。
- Flutter -- Dart

などがあります。

♣ なぜ Flutter なのか？

Flutter では作成したアプリケーションが、それぞれのネイティブ言語に変換されてからビルドされるので高いパフォーマンスと信頼性が得られます。

Flutter を使えば、

- Android Studio か VSCode でプログラミング言語 Dart で、
- ひとつのソースコードで、Android、iOS のアプリケーションに対応

できます。

リリース時には、プロジェクトフォルダに Flutter から変換された、

Android、

Android Studio 上でビルド

iPhone、

iOS フォルダ（objective-C、Kotlin の選択）を Xcode を使ってビルド

Windows 用、Mac 用の各デスクトップ、Web アプリ

それぞれのプラットフォーム上で Flutter コマンドでビルド

驚くほど簡単にアプリが作成できますので、手持ちの PC に開発環境を作成しましょう。

すべて無料で始めることができます。

目次

はじめに	i
第1章 Flutter とは？	1
第2章 Flutter 開発環境の作成手順	3
2.1 本家サイト	3
第3章 Homebrew	5
3.1 Homebrew のインストール	5
第4章 バージョン管理ツール asdf	9
4.1 asdf のインストール	9
第5章 Flutter のインストール	14
第6章 Xcod	20
6.1 Xcodes のインストール	21
6.2 Xcode のインストール	21
6.2.1 Xcodes を使用する場合	21
6.2.2 Apple App Store からインストールする場合	23
第7章 Xcode のインストール後の確認	24
第8章 CocoaPods	26
8.1 CocoaPods のインストール	26
8.1.1 gem でインストール	26
8.1.2 Homebrew でインストール	27
8.2 flutter doctor での確認	27
第9章 AndroidStudio	29
9.1 Android Studio のインストール	29
9.2 Android Studio のセットアップ	31
9.3 flutter doctor の確認	32
第10章 command-line tools のインストール	37
10.1 cmdline-tools のインストール	37

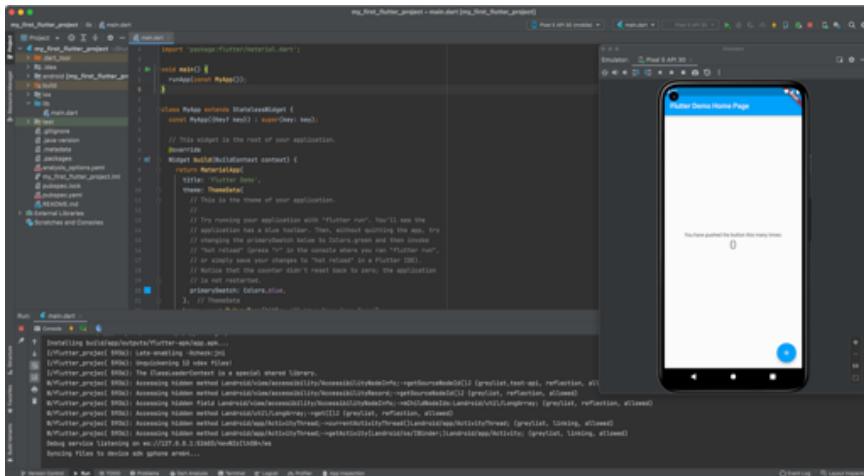
10.2	Android ライセンス	40
10.3	Android Studio へのプラグインのインストール	41
第 11 章 プロジェクトのスタート		44
11.1	スタートアッププロジェクトを作る	44
第 12 章 Android エミュレータ		48
12.1	Android Platform	48
12.2	エミュレータの作成	49
第 13 章 iPhone エミュレータ		66
13.1	iOS エミュレータ	66
13.2	以前の iOS バージョンの機種を作成	66
第 14 章 エミュレータでデバッグ		74
14.1	Android エミュレータ	74
14.2	iPhone エミュレータ	74
第 15 章 おわりに		78

第 1 章

Flutter とは？

それでは、「Flutter とは何なのか？」をもう少し詳しく紹介します。

「はじめに」でも紹介しましたように、開発環境（Android Studio 含む）が作成できさえすれば、数クリックするだけで以下のようなスタートアップアプリがエミュレータ上で動作します。



▲ 図 1.1: Android Studio でスタートアッププロジェクトのデバッグ

Google Play や Apple Store で公開するためには、それぞれに開発者登録したり、リリース作業が必要になります。

しかし、モバイルアプリケーションを作成するためには、

- iOS アプリ開発のためプログラミング言語 Swift の学習
- Xcode（統合開発環境）の使い方を取得
- Android アプリ開発のためプログラミング言語 Java、または、Kotlin の学習
- Android Studio（統合開発環境）の使い方を取得

など、膨大な作業が必要だと諦めていたのではありませんか？

しかし、Flutterを使えば、モバイルアプリ開発へのハードルはグッと下がります。その理由は、

- UIデザインが簡単
- 開発環境が充実
- Dart（プログラミング言語）の習得が楽

だからです。

もう少し詳しく説明すると、

UIは、Widgetを組み合わせて作成

アプリケーションのUI（ユーザー・インターフェース）部分は、見える部分・見えない部分もWidget（ウィジェット）と呼ばれる部品を組み合わせて作ります。

そのWidgetは、テキスト入力用のテキストフィールド、ボタンなどの基本的なものから、有料・無料で世界中の開発者が公開しているものまで豊富にあります。

もちろん、あなた自身で作ることもできます。

さらに、Adobe XDやFigmaなどのデザインツールからUIコードの生成もできます。

開発環境の充実

Android Studio（統合開発環境）、Visual Studio Codeでのコード編集やデバッグ環境もあります。

そして、Flutterはオープンソースとはいえ、Googleが開発の中心のためGoogle製品のAndroidやFirebaseとは非常に親和性が高くドキュメントも充実しています。

最近は、QiitaやZennなどの技術情報サイトでもFlutterに関する記事が増えています。

Dart

Flutterで使われるプログラミング言語は、「Dart」です。オブジェクト指向、クラスベース、ガーベージコレクションを備えたGoogleの開発したものです。

Google社内では、開発当初はJavaScriptを置き換えるものとしていましたので、JavaScript経験者なら学習コストは低いでしょうし、JavaやC#経験していれば、さらに取得は簡単でしょう。

つまり、Flutterを使えば、モバイルアプリが簡単に作れるのです。

第 2 章

Flutter 開発環境の作成手順

macOSへのFlutter開発環境の構築は、以下の手順で行います。

本家サイトで紹介されている方法と少し違いますが、Flutterのアップデートなどに対応できる
ようにします。

2.1 本家サイト

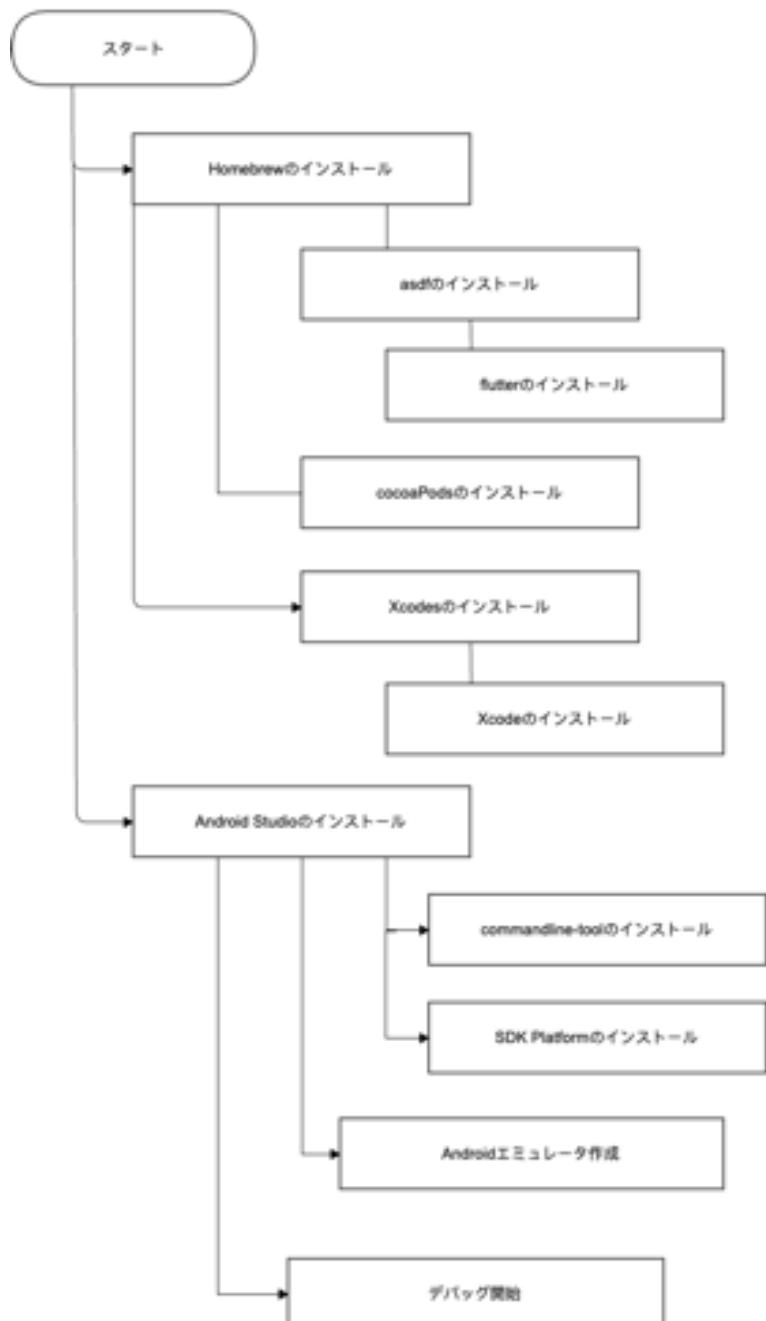
本家サイトのmacOS^{*1}向けのインストール方法とは少し違います。

こちらでは、Xcode、gitがインストールされていることが前提になっていますが、本書では、
Homebrewのインストール方法も解説し、Homebrewを使って必要なものをインストールしてい
きます。

手順は、以下のようになります。

^{*1} <https://docs.flutter.dev/get-started/install/macos>

Flutter開発環境の作成 Mac編



▲図 2.1: macOSへのFlutter開発環境の作成手順

第3章

Homebrew

Homebrew (Mac、Linux 用のパッケージマネージャー) をインストールします。すでにインストール済みの方はスキップしてください。

パッケージマネージャーとは、対象のプログラム・コマンド（たとえば、git）をインストールするときに、必要なライブラリなどを一括でインストールしてくれます。また、パッケージマネージャーでインストールしておけば、アップデートも一括で行えます。マシーンの引越し時にも、リストをコピーするだけで一括インストールができます。

今回の開発環境の作成では、

- git --- Flutter の必須、
- asdf --- Flutter のバージョン管理、
- CocoaPods --- Swift、Objective-C での Cocoa プロジェクトの依存関係を管理、
- Xcodes --- Xcode のバージョン管理、

をインストールするために使います。

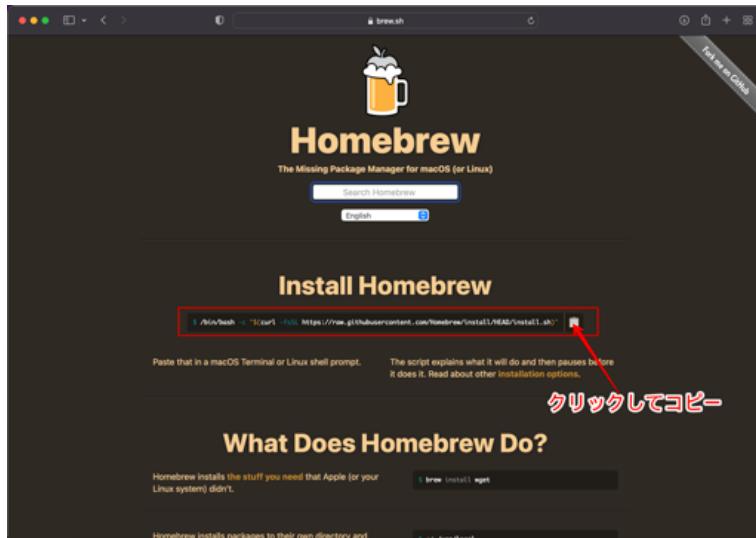
ここでのバージョン管理とは、複数のバージョン (Flutter、Xcode) をインストールし、バージョンを切り替えて使えるようにすることです。

3.1 Homebrew のインストール

Homebrew 本家^{*1}のページへアクセスします。

1. 上図の赤ワク部分を、アイコンをクリックしてコピーする。
2. ターミナルを開き、貼り付けます。
3. エンターキーで実行します。

^{*1} https://brew.sh/index_ja



ターミナルについて

ターミナルは、macOS 付属のもの^{*2}でも良いですし、ご自分の好みで選択してください。私は、「iTerm2」を使っています。

こちらに、macOS で使えるターミナルの比較をされた良記事 [Mac の端末をいろいろ検討した結果 iTerm2 になった^{*3}](#) があります。

実行すると、以下のようになります。

- インストールするものを表示
- 作成するディレクトリを表示
- Xcode Command Line Tools (Xcode インストール済みの場合はスキップ) インストールを表示
- 確認のため、「実行にはエンターキー」、「キャンセルは、ほかのキー」を押す。

ターミナル画面について

少し薄い文字でコメントをターミナル画面上に、「←←コメント」のように入っています。

▼ homebrew インストールコマンドの実行

```
> /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"< エンターキー  
==> Checking for `sudo` access (which may request your password)...  
Password:< Macのパスワードを入力しエンターキー  
==> This script will install:< 以下をインストールしますよ～
```

```
/opt/homebrew/bin/brew
/opt/homebrew/share/doc/homebrew
/opt/homebrew/share/man/man1/brew.1
.
• 中略
.

/opt/homebrew
==> The following new directories will be created: ←以下のフォルダが作成されるよ～
/opt/homebrew/bin
/opt/homebrew/etc
.
• 中略
.

/opt/homebrew/Frameworks
==> The Xcode Command Line Tools will be installed. ← XCodeのコマンドラインツールがインストールされるよ～
Press RETURN/ENTER to continue or any other key to abort: ←実行するならエンターキー、止めないのであれば他のキーを押して
```

ここで、エンターキーを押します。

▼ エンターキーを押して続行

```
--> /usr/bin/sudo /usr/bin/install -d -o root -g wheel -m 0755 /opt/homebrew
--> /usr/bin/sudo /bin/mkdir -p /opt/homebrew/bin ←先ほど表示したディレクトリ作成
←作成したディレクトリのパーミッション変更
← AppleよりXCode Command Line Toolsのインストール
← Homebrewのダウンロードとインストール
==> Installation successful! ←インストール完了
==> Next steps: ←終わったら以下を実行してね
- Run these two commands in your terminal to add Homebrew to your PATH:
echo 'eval "$( /opt/homebrew/bin/brew shellenv )"' >> /Users/ユーザー名/.zprofile
eval "$( /opt/homebrew/bin/brew shellenv )"

- Run brew help to get started
- Further documentation: https://docs.brew.sh
```

Homebrew のインストールが完了しましたので、パスを通すために「`~/.zshrc`」に以下を追加してください。上記では「`.zprofile`」に追加するようにありますが、「`.zshrc`」に書込みます。

以下のコマンドを実行すると、「`~/.zshrc`」の最後の行へ「`eval "$(/opt/homebrew/bin/brew shellenv)"`」を追加します。「`~/.zshrc`」がない場合にはファイルを作成して追加します。

.zprofile と.zshrc の違い

こちらに、「[zsh の設定ファイルの読み込み順序と使い方 Tips まとめ^{*4}](#)」良記事があります。

▼ homebrew のパスを通す

```
> echo 'eval "$( /opt/homebrew/bin/brew shellenv )"' >> ~/.zshrc

> source ~/.zshrc ← 設定ファイル.zshrcの読み込み
> brew help ← homebrewが動くかヘルプを表示
Example usage:
brew search TEXT|/REGEX/
brew info [FORMULA|CASK...]
brew install FORMULA|CASK...
brew update
.
.
.
brew help [COMMAND]
man brew
https://docs.brew.sh
```

以上で、Homebrew のインストールが完了しました。

最初に、「git」をインストールしてみましょう。ターミナルに「brew install git」と入力しエンターキーを押します。

▼ git のインストール

```
> brew install git
```

これだけで「git」がインストールできます。

^{*4} <https://qiita.com/muran001/items/7b104d33f5ea3f75353f>

第 4 章

バージョン管理ツール asdf

そもそも、バージョン管理ツールとは？、また、なぜ必要なの？

Flutter もそうですが、Python、Java、Ruby、Node.js などのプログラミング言語やフレームワークは日々更新され、ある程度の修正・機能追加が終わった時点で新バージョンとしてリリースされます。

新バージョンがリリースされる度に開発環境を更新すれば良いのですが、開発中のプロジェクトに影響がないとはいえません。

また、以前のバージョンで開発したプロジェクトの保守を異なるバージョンの開発環境で行うと、

- 動かなくなったり、
 - 使用していた Widget などが非推奨（deprecation）になった。
- など、予期せぬことが起こります。
そのため、
- PC 全体で使えるデフォルトバージョンの指定、
 - プロジェクトフォルダ毎でのバージョンの指定、

ができるように、バージョン管理ツールをインストールします。

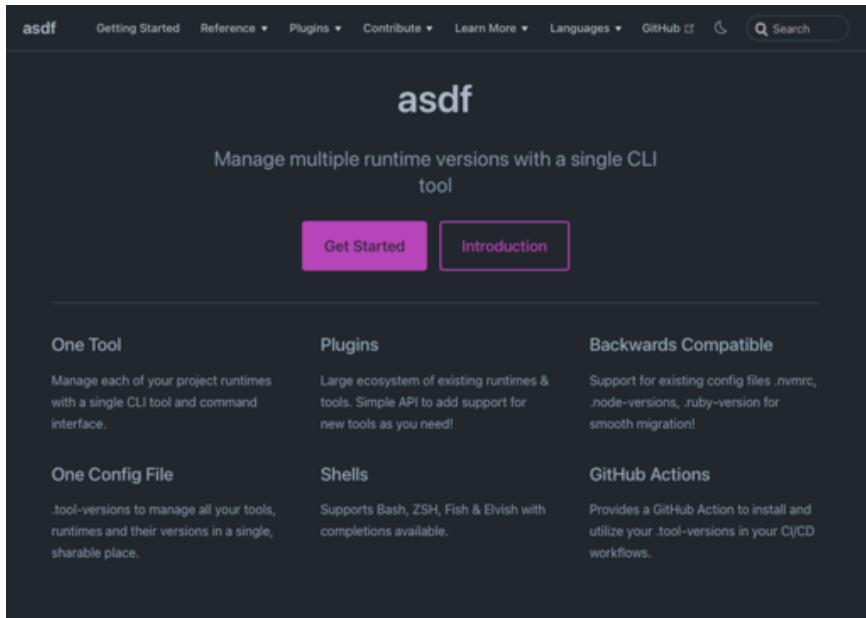
4.1 asdf のインストール

asdf 本家サイト^{*1}

バージョン管理ツールは、

- Java のバージョン管理ツール jenv など
- Ruby のバージョン管理ツール rbenv など
- python のバージョン管理ツール pyenv、anaconda など
- Node.js のバージョン管理 nvm など

^{*1} <https://asdf-vm.com>



プログラミング言語、フレームワーク毎にたくさんあります。

Flutter にもバージョン管理ツールとして、「asdf」、「fvm」などがあります。「fvm」は、「flutter コマンド」を直接入力できないので、今回は「asdf」を使います。

「asdf」は、flutter だけではなく上記のものや「php」などたくさんのプログラミング言語・フレームワークのバージョン管理が行えます。これは、各プログラミング言語・フレームワークをプラグインとして扱い、asdf にインストールするプラグイン毎にバージョン管理ができるためです。

それでは、asdf 本体のインストールを行います。asdf のインストールは、Homebrew を使用します。

ターミナルに「brew install asdf」と入力しエンターキーを押します。

▼ asdf のインストール

```
> brew install asdf

To use asdf, add the following line to your ~/.zshrc:
. /opt/homebrew/opt/asdf/libexec/asdf.sh
Restart your terminal for the settings to take effect.
zsh completions have been installed to:
/opt/homebrew/share/zsh/site-functions
==> Summary
☒  /opt/homebrew/Cellar/asdf/0.10.2: 168 files, 717.0KB
==> Running `brew cleanup asdf`...
```

```
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.  
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).  
==> Caveats  
==> asdf  
To use asdf, add the following line to your ~/.zshrc:  
  
. /opt/homebrew/opt/asdf/libexec/asdf.sh  
  
Restart your terminal for the settings to take effect.  
  
zsh completions have been installed to:  
/opt/homebrew/share/zsh/site-functions
```

インストールが完了しましたので、「Caveats」にある指示に従い「`~/.zshrc`」に asdf のスクリプトが実行されるように、

```
. /opt/homebrew/opt/asdf/libexec/asdf.sh
```

を追加します。

現時点での「`~/.zshrc`」は、

▼リスト 4.1:

```
# Homebrew  
eval "$( /opt/homebrew/bin/brew shellenv )"  
  
# asdf  
. /opt/Homebrew/opt/asdf/libexec/asdf.sh
```

があれば大丈夫です。

設定ファイルを再度読み込みます。

▼.zshrc の再読み込み

```
> source ~/.zshrc
```

これで、asdf が動作します。ターミナルに「`asdf`」と入力しエンターキーを押すと、ヘルプが表示されます。

▼asdf のヘルプ

```
☒ asdf  
version: v0.10.2  
  
MANAGE PLUGINS  
asdf plugin add <name> [<git-url>] Add a plugin from the plugin repo OR,  
add a Git repo as a plugin by  
specifying the name and repo url  
  
asdf plugin list [--urls] [--refs] List installed plugins. Optionally show  
git urls and git-ref
```

```
asdf plugin list all           List plugins registered on asdf-plugins
                             repository with URLs
asdf plugin remove <name>    Remove plugin and package versions
                             Update asdf to the latest on the master branch
.
.
.
RESOURCES
GitHub: https://github.com/asdf-vm/asdf
Docs:   https://asdf-vm.com

"Late but latest"
-- Rajinikanth
```

Flutter のバージョン管理ができるように、「flutter プラグイン」をインストールします。どんなプログラミング言語・フレームワークのバージョン管理ができるのかは、「asdf plugin list all」で確認できます。

ターミナルに「asdf plugin add flutter」と入力しエンターキーを押します。

▼ flutter プラグインのインストール

```
> asdf plugin add Flutter
updating plugin repository...HEAD is now at 08fb6a9 fix: recent erroneous PRs (#658)
```

Flutter のインストールは、asdf 経由で行います。asdf 経由でインストール可能な Flutter のバージョンを確認します。

ターミナルに「asdf list all flutter」と入力しエンターキーを押します。

▼ asdf でインストール可能な Flutter のバージョン表示

```
> asdf list all flutter
0.1.6-dev
0.2.1-dev
0.1.8-dev
0.1.9-dev
1.26.0-12.0.pre-dev
1.22.6-stable
.
.
.
2.13.0-0.1.pre-beta
2.10.5-stable ← 2.x系の最新版
.
.
.
3.0.5-stable ← 最新の安定版
```

```
•  
• 中略  
•  
3.3.0-0.5.pre-beta ←最新のベータ版
```

過去のバージョンも含めインストール可能なバージョンが表示されます。

..... インストール可能バージョンが表示されない場合

Mac 用の flutter は、「intel chip」、「Apple Silicon」版と分かれています。バージョン表示されない場合は、

▼ Rosetta2 使用許可

```
> sudo softwareupdate --install-rosetta --agree-to-license
```

を行ってください。

以上で、バージョン管理ツール「asdf」のインストールが完了し、Flutter インストールの準備ができました。

第 5 章

Flutter のインストール

Flutter の最新版をインストールし、PC 全体で最新版を使えるようにします。
また、2.x 系の最新版もインストールし、バージョンを切り替えて使えることを確認します。

Flutter は、バージョンの切替ができるように asdf 経由でインストールします。先ほどインストール可能なバージョンを確認したところ最新版は「3.0.5-stable」でしたので、このバージョンを指定してインストールします。

ターミナルに「asdf install flutter インストールしたいバージョン」を入力しエンターキーを押します。

▼ Flutter のインストール

```
> asdf install flutter 3.0.5-stable
/Users/ユーザー名/.asdf/plugins/flutter/bin/install: line 25: jq: command not found
/Users/ユーザー名/.asdf/plugins/flutter/bin/install: line 26: jq: command not found
/Users/ユーザー名/.asdf/plugins/flutter/bin/install: line 27: [: -gt: unary operator expected
Cannot find the download url for the version: 3.0.5-stable
```

「jq: command not found」と表示されエラーとなりました。

「jq」は何かを調べると、「Lightweight and flexible command-line JSON processor」とあり、インストール先のシステムが「intel chip」、「Apple Silicon」で処理を分ける処理で使われていました。

「jq」は Homebrew でインストールできます。ターミナルに「brew intall jq」と入力しエンターキーを押します。

▼jq のインストール

```
> brew install jq
```

jq のインストールが完了しましたら、再度 Flutter 最新版のインストールを行います。

▼Flutter 最新版のインストール

```
> asdf install flutter 3.0.5-stable
% Total      % Received % Xferd  Average Speed   Time      Time      Time  Current
Dload  Upload   Total   Spent    Left  Speed
100 1188M  100 1188M     0       0  38.7M          0  0:00:30  0:00:30 --::-- 41.1M
```

最新版（3.0.5-stable）のインストールが完了しました。インストール完了の確認のため、インストールしたバージョンを確認します。

ターミナルに「flutter --version」と入力しエンターキーを押します。

▼Flutter バージョンの確認

```
> flutter --version
No version is set for command flutter
Consider adding one of the following versions in your config file at
flutter 3.0.5-stable
```

設定ファイルに「flutter 3.0.5-stable」を追加しなさい。と怒られました。

2.x 系の最新版もインストールします。

▼Flutter 2.x 系のインストール

```
> asdf install flutter 2.10.5-stable
% Total      % Received % Xferd  Average Speed   Time      Time      Time  Current
Dload  Upload   Total   Spent    Left  Speed
100 1126M  100 1126M     0       0  30.3M          0  0:00:37  0:00:37 --::-- 28.7M
```

同じくバージョンを確認してみます。

▼Flutter バージョンの確認

```
> flutter --version
No version is set for command flutter
Consider adding one of the following versions in your config file at
flutter 3.0.5-stable
flutter 2.10.5-stable
```

今回も設定ファイルにバージョンを追加しなさいと怒られました。

PC 全体で最新版が使えるように設定します。

ターミナルに「asdf global flutter 3.0.5-stable」と入力しエンターキーを押します。その後、バージョン確認も行います。

▼ asdf で使用する Flutter バージョンの指定

```
> asdf global flutter 3.0.5-stable
> flutter --version
Flutter 3.0.5 • channel stable • https://github.com/flutter/flutter.git
Framework • revision f1875d570e (4 weeks ago) • 2022-07-13 11:24:16 -0700
Engine • revision e85ea0e79c
Tools • Dart 2.17.6 • DevTools 2.12.2

[✓]
> [✓] Welcome to Flutter! - https://flutter.dev [✓]
> [✓]
> [✓]
> [✓]
> [✓] The Flutter tool uses Google Analytics to anonymously report feature usage [✓]
> [✓] statistics and basic crash reports. This data is used to help improve [✓]
> [✓] Flutter tools over time. [✓]
> [✓]
> [✓]
> [✓] Flutter tool analytics are not sent on the very first run. To disable [✓]
> [✓] reporting, type 'flutter config --no-analytics'. To display the current [✓]
> [✓] setting, type 'flutter config'. If you opt out of analytics, an opt-out [✓]
> [✓] event will be sent, and then no further information will be sent by the [✓]
> [✓] Flutter tool. [✓]
> [✓]
> [✓]
> [✓] By downloading the Flutter SDK, you agree to the Google Terms of Service. [✓]
> [✓] Note: The Google Privacy Policy describes how data is handled in this [✓]
> [✓] service. [✓]
> [✓]
> [✓]
> [✓] Moreover, Flutter includes the Dart SDK, which may send usage metrics and [✓]
> [✓] crash reports to Google. [✓]
> [✓]
> [✓]
> [✓] Read about data we send with crash reports: [✓]
> [✓]
> [✓] https://flutter.dev/docs/reference/crash-reporting [✓]
> [✓]
```

- >
- > See Google's privacy policy:
 - >
 - > <https://policies.google.com/privacy>
 - >
 - >
- >

asdf コマンドで「global」指定をしたので、PC のどのフォルダでも「3.0.5-stable」が使われます。プロジェクト毎に別なバージョンを使い対場合には、「global」の代わりにプロジェクトフォルダへ移動し「asdf local flutter 使用するバージョン」をターミナルに入力しエンターキーを押します。

全体を「2.10.5-stable」に切り替えて確認します。

▼ Flutter 2.10.5-stable へ切替

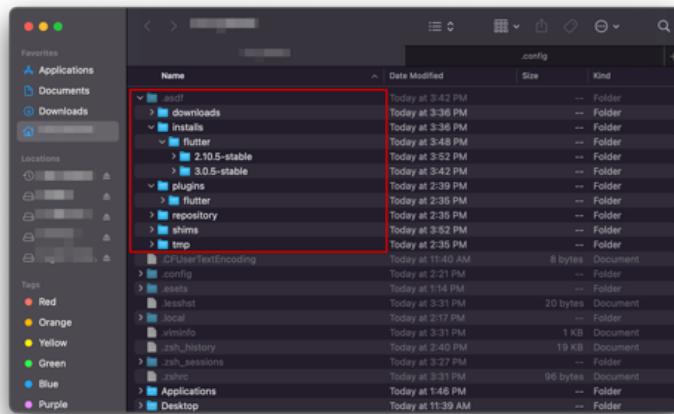
バージョンが「2.10.5」と表示され、新しいバージョンがあることも表示しています。

最新版を使うよう戻します。

▼ Flutter バージョン切り替え

```
> asdf global flutter 3.0.5-stable
> flutter --version
Flutter 3.0.5 • channel stable • https://github.com/flutter/flutter.git
Framework • revision f1875d570e (6 weeks ago) • 2022-07-13 11:24:16 -0700
Engine • revision e85ea0e79c
Tools • Dart 2.17.6 • DevTools 2.12.2
```

複数の Flutter バージョンをインストールしましたが、いったいどこのフォルダへ、どのようにインストールされているのでしょうか？Finderで確認すると、以下のようにインストールされています。



▲図 5.1: flutter の複数バージョンのインストール状況

flutter をインストールできましたので、flutter 開発環境の内容を、flutter コマンドで確認します。ターミナルに「flutter doctor」と入力しエンターキーを押すのですが、より詳しく表示されるように「-v」オプションを付けて確認してみます。

▼ flutter doctor -v

```
> flutter doctor -v
Running "flutter pub get" in flutter_tools...                                5.1s
[✓] Flutter (Channel stable, 3.0.5, on macOS 12.5 21G72 darwin-arm, locale en-CA)
  • Flutter version 3.0.5 at /Users/ユーザー名/.asdf/installs/flutter/3.0.5-stable
  • Upstream repository https://github.com/flutter/flutter.git
  • Framework revision f1875d570e (5 weeks ago), 2022-07-13 11:24:16 -0700
  • Engine revision e85ea0e79c
  • Dart version 2.17.6
  • DevTools version 2.12.2

[!] Android toolchain - develop for Android devices
    ✘ Unable to locate Android SDK.
    Install Android Studio from: https://developer.android.com/studio/index.html
    On first launch it will assist you in installing the Android SDK components.
    (or visit https://flutter.dev/docs/get-started/install/macos#android-setup for detailed instructions).
    If the Android SDK has been installed to a custom location, please use
    `flutter config --android-sdk` to update to that location.
```

```
[✗] Xcode - develop for iOS and macOS
  ✗ Xcode installation is incomplete; a full installation is necessary for iOS development.
    Download at: https://developer.apple.com/xcode/download/
    Or install Xcode via the App Store.
    Once installed, run:
      sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer
      sudo xcodebuild -runFirstLaunch
  ✗ CocoaPods not installed.
    CocoaPods is used to retrieve the iOS and macOS platform side's plugin code that responds to your plugin usage on the Dart side.
    Without CocoaPods, plugins will not work on iOS or macOS.
    For more info, see https://flutter.dev/platform-plugins
    To install see https://guides.cocoapods.org/using/getting-started.html#installation
    for instructions.

[✓] Chrome - develop for the web
  • Chrome at /Applications/Google Chrome.app/Contents/MacOS/Google Chrome

[!] Android Studio (not installed)
  • Android Studio not found; download from https://developer.android.com/studio/index.html
    (or visit https://flutter.dev/docs/get-started/install/macos#android-setup for detailed instructions).

[✓] Connected device (2 available)
  • macOS (desktop) • macos • darwin-arm64 • macOS 12.5 21G72 darwin-arm
  • Chrome (web)   • chrome • web-javascript • Google Chrome 104.0.5112.79

[✓] HTTP Host Availability
  • All required HTTP hosts are available

! Doctor found issues in 3 categories.
```

現時点では、3つのカテゴリに問題があると表示されています。内容は、バツが付いているものです。

1. Android toolchain
2. Xcode
3. Android studio

となっていきます。

これを、順番に解決すれば Flutter 開発環境の作成が完了となります。

第 6 章

Xcod

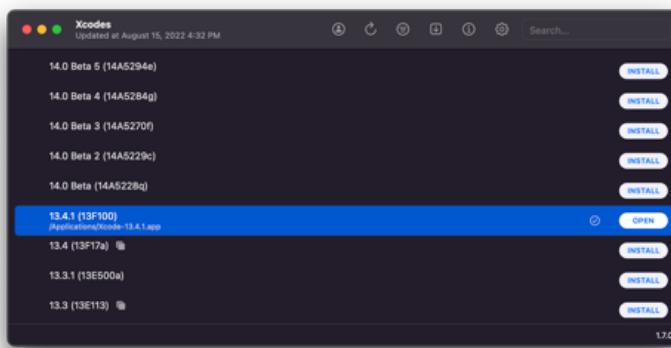
Xcode は Apple が提供するシステム統合開発環境です。統合開発環境も Flutter などのフレームワークやプログラミング言語と同じように macOS と共にバージョンアップされます。

Xcode も同じようにバージョン管理の対象にし、「Xcodes」と言うツールを使い、複数のバージョンを切り替えることができるようになります。



▲図 6.1: Xcode のバージョン管理ツール

Xcodes とは、複数の Xcode バージョンをインストールでき、それを簡単に切り替えることができるツールです。ダウンロードも独自で行うため、Apple Store からダウンロードするよりも断然早くダウンロードできます。



▲図 6.2: Xcodes

6.1 Xcodes のインストール

Apple App Store からインストールされる方は、ここはスキップしてください。

Xcodes は Homebrew からインストールします。ターミナルに「brew install --cask xcodes」と入力しエンターキーを押します。

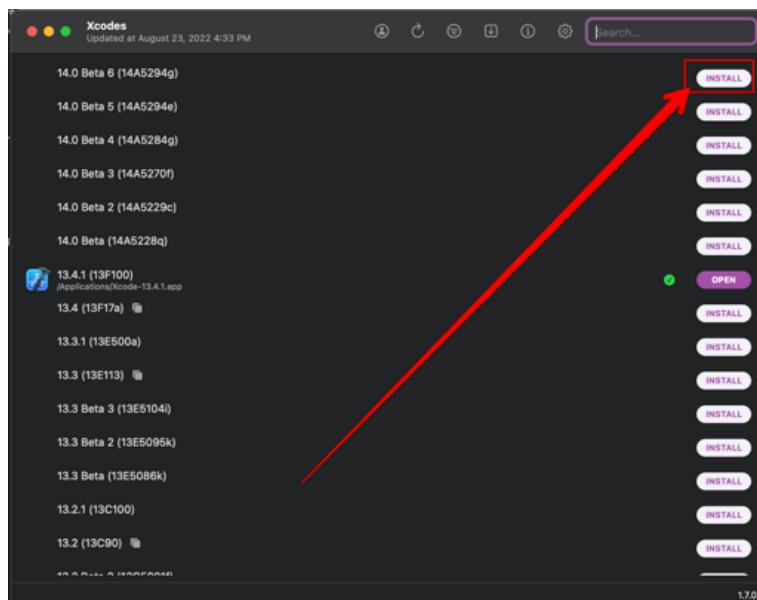
▼ Xcodes のインストール

```
> brew install --cask xcodes
```

6.2 Xcode のインストール

♣ 6.2.1 Xcodes を使用する場合

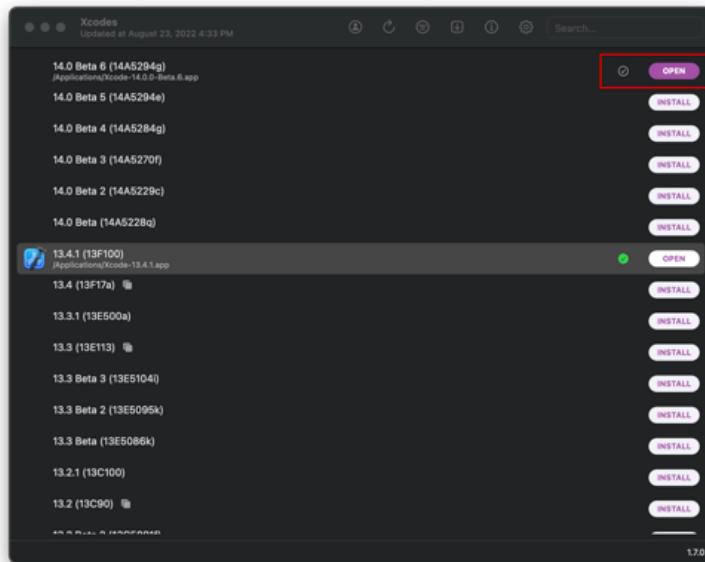
インストールが完了したら、Launchpad などから起動します。通常の GUI アプリケーションですので Application フォルダにインストールされています。



▲ 図 6.3: Xcodes を起動

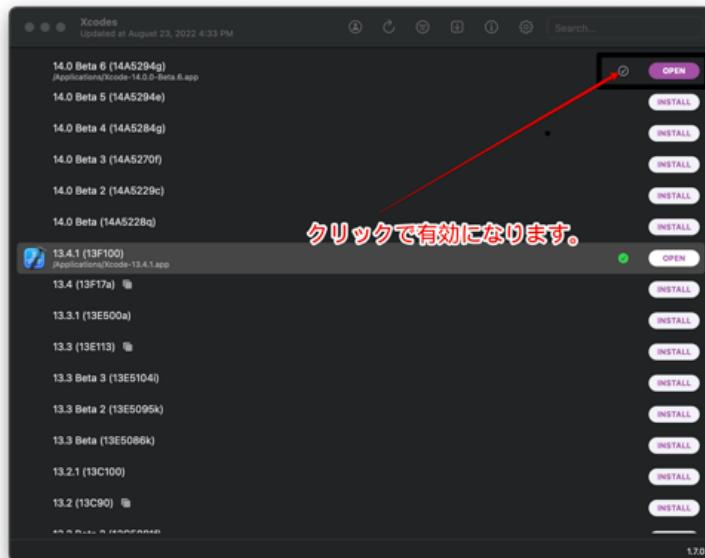
インストール可能なバージョンが表示されますので、インストールするバージョンの右側の「INSTALL」ボタンをクリックします。ダウンロードが完了しましたら、「INSTALL」ボタンが「OPEN」ボタンに変わります。

インストール先は、「/Application フォルダ」にバージョン付きの名前です。



▲図 6.4: Xcode のインストール完了

「Open」ボタンの左にある「チェックマーク」をクリックすると、対象のバージョンがアクティブになります。アクティブにすると、Xcode インストール後に行う以下のコマンドが自動で実行されます。



このコマンドは、Xcodes で選択したバージョンをアクティブにすると自動で実行されます。

▼ Xcode インストール後に必要なコマンド

```
> sudo xcode-select --switch /Applications/XCode+バージョン.app/Contents/developer  
> sudo xcodebuild -runFirstLaunch
```

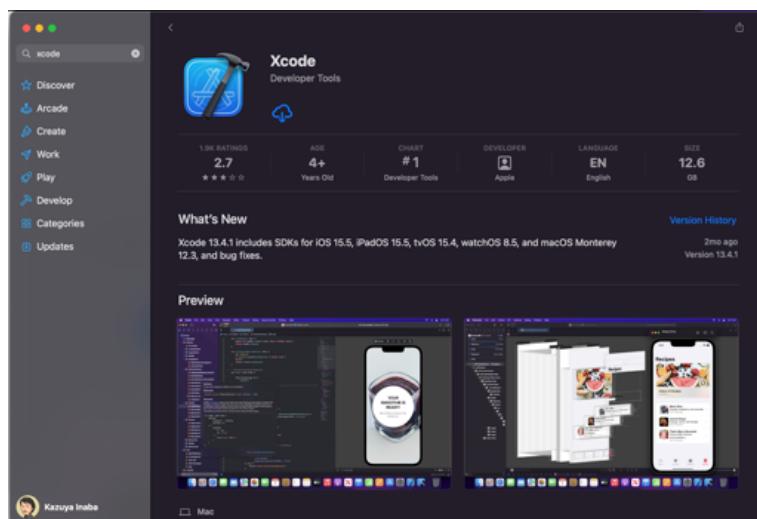
現在の Xcode のバージョンの確認は、以下のコマンドで行います。

▼ Xcode バージョン確認

```
☒ xcodebuild -version  
Xcode 14.0  
Build version 14A5294g  
☒ xcodebuild -version  
Xcode 13.4.1  
Build version 13F100
```

♣ 6.2.2 Apple App Store からインストールする場合

Launchpad から「App Store」を起動します。



▲ 図 6.5: Mac App Store

「入手」、「インストール」をクリックしてインストールします。Xcode は 13GB 近くありますので時間がかかります。

以上で、Xcode のインストールが完了しました。

第 7 章

Xcode のインストール後の確認

Xcode をインストールしましたので、再度、「flutter doctor -v」で確認します。

ターミナルに「flutter doctor -v」と入力しエンターキーを押します。

▼ Xcode インストール後の flutter doctor の結果

```
> flutter doctor -v
[✓] Flutter (Channel stable, 3.0.5, on macOS 12.5 21G72 darwin-arm, locale en-CA)
  • Flutter version 3.0.5 at /Users/ユーザー名/.asdf/install/flutter/3.0.5-stable
  • Upstream repository https://github.com/flutter/flutter.git
  • Framework revision f1875d570e (5 weeks ago), 2022-07-13 11:24:16 -0700
  • Engine revision e85ea0e79c
  • Dart version 2.17.6
  • DevTools version 2.12.2

[✗] Android toolchain - develop for Android devices
    ✗ Unable to locate Android SDK.
      Install Android Studio from: https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK components.
      (or visit https://flutter.dev/docs/get-started/install/macos#android-setup for detailed instructions).
      If the Android SDK has been installed to a custom location, please use
      `flutter config --android-sdk` to update to that location.

[!] Xcode - develop for iOS and macOS (Xcode 13.4.1)
  • Xcode at /Applications/Xcode-13.4.1.app/Contents/Developer
  ✗ CocoaPods not installed.
    CocoaPods is used to retrieve the iOS and macOS platform side's plugin code that responds to your plugin usage on the Dart side.
    Without CocoaPods, plugins will not work on iOS or macOS.
    For more info, see https://flutter.dev/platform-plugins
    To install see https://guides.cocoapods.org/using/getting-started.html#installation for instructions.

[✓] Chrome - develop for the web
  • Chrome at /Applications/Google Chrome.app/Contents/MacOS/Google Chrome
```

```
[!] Android Studio (not installed)
  • Android Studio not found; download from https://developer.android.com/studio/index.html
    (or visit https://flutter.dev/docs/get-started/install/macos#android-setup for detailed instructions).

[✓] Connected device (2 available)
  • macOS (desktop) • macos • darwin-arm64 • macOS 12.5 21G72 darwin-arm
  • Chrome (web) • chrome • web-javascript • Google Chrome 104.0.5112.79

[✓] HTTP Host Availability
  • All required HTTP hosts are available

! Doctor found issues in 3 categories.
```

Xcode の問題はなくなりましたが、

1. CocoaPods がインストールされていない。
2. Android toolchain
3. Android studio

と問題が 3 カテゴリにあると表示されています。

次に、「CocoaPods」をインストールします。

第8章

CocoaPods

「CocoaPods」とは、Swift、Objective-C プロジェクトの依存関係を管理してくれるものです。Homebrew と同じように、Swift や Objective-C プロジェクトに必要なプログラムをインストールでき、そのプログラムが依存するライブラリを自動的にインストールしてくれるものです。

8.1 CocoaPods のインストール

CocoaPods は、プログラミング言語「Ruby」の依存関係を管理する「gem」を使ってインストールします。

しかし、gem は Ruby のバージョンに影響されます。macOS Monterey でのデフォルトは、「ruby-2.6.8」で、最新版は「ruby-3.1.2」です。

メジャーバージョンも異なるため、どんな影響ができるのか不明です（影響なしの場合もあります）。これがバージョンの違いによる依存関係の問題です。

それを避ける方法としては、依存関係を自動で解決してくれる Homebrew を使います。

♣ 8.1.1 gem でインストール

自分でトラブルを解決できる方は、gem を使ってインストールできます。

▼ gem を使って CocoaPods をインストール

```
> sudo gem update --system ← gemをさいしんの状態にする  
> sudo gem install cocoapods ← CocoaPodsのインストール
```

♣ 8.1.2 Homebrew でインストール

Homebrew を使ってインストールする場合は、ターミナルに「brew install cocoapods」と入力しエンターキーを押します。

▼ Homebrew で CocoaPods をインストール

```
> brew install cocoapods
Warning: Treating cocoapods as a formula. For the cask, use homebrew/cask/cocoapods
=> Downloading https://ghcr.io/v2/homebrew/core/ruby/manifests/3.1.2
#####
100.0%
=> Downloading https://ghcr.io/v2/homebrew/core/ruby/blobs/sha256: •
=> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256 • >
> •
#####
100.0%
=> Downloading https://ghcr.io/v2/homebrew/core/cocoapods/manifests/1.11.3
#####
100.0%
=> Downloading https://ghcr.io/v2/homebrew/core/cocoapods/blobs/sha256 •
=> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256 • >
> •
#####
100.0%
=> Installing dependencies for cocoapods: ruby
=> Installing cocoapods dependency: ruby
=> Pouring ruby--3.1.2.arm64_monterey.bottle.tar.gz
[X] /opt/homebrew/Cellar/ruby/3.1.2: 15,996 files, 42.8MB
=> Installing cocoapods
=> Pouring cocoapods--1.11.3.arm64_monterey.bottle.tar.gz
[X] /opt/homebrew/Cellar/cocoapods/1.11.3: 13,496 files, 28.3MB
=> Running `brew cleanup cocoapods`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
```

8.2 flutter doctor での確認

CocoaPods のインストールが完了しましたので、「flutter doctor -v」で確認します。

▼ flutter doctor -v で確認

```
> flutter doctor -v
[✓] Flutter (Channel stable, 3.0.5, on macOS 12.5 21G72 darwin-arm, locale en-CA)
  • Flutter version 3.0.5 at /Users/ユーザー名8/.asdf/install/flutter/3.0.5-stable
  • Upstream repository https://github.com/flutter/flutter.git
  • Framework revision f1875d570e (5 weeks ago), 2022-07-13 11:24:16 -0700
  • Engine revision e85ea0e79c
  • Dart version 2.17.6
  • DevTools version 2.12.2

[X] Android toolchain - develop for Android devices
  [X] Unable to locate Android SDK.
    Install Android Studio from: https://developer.android.com/studio/index.html
    On first launch it will assist you in installing the Android SDK components.
```

```
(or visit https://flutter.dev/docs/get-started/install/macos#android-setup for detailed instructions).
If the Android SDK has been installed to a custom location, please use
`flutter config --android-sdk` to update to that location.

[✓] Xcode - develop for iOS and macOS (Xcode 13.4.1)
    • Xcode at /Applications/Xcode-13.4.1.app/Contents/Developer
    • CocoaPods version 1.11.3

[✓] Chrome - develop for the web
    • Chrome at /Applications/Google Chrome.app/Contents/MacOS/Google Chrome

[!] Android Studio (not installed)
    • Android Studio not found; download from https://developer.android.com/studio/index.html
(or visit https://flutter.dev/docs/get-started/install/macos#android-setup for detailed instructions).

[✓] Connected device (2 available)
    • macOS (desktop) • macos • darwin-arm64 • macOS 12.5 21G72 darwin-arm
    • Chrome (web)   • chrome • web-javascript • Google Chrome 104.0.5112.79

[✓] HTTP Host Availability
    • All required HTTP hosts are available

! Doctor found issues in 2 categories.
```

Xcode の欄の問題が解決され、2つのカテゴリとなりました。

1. Android toolchain
2. Android studio

の2点が問題として表示されています。

次に、Android Studio をインストールします。

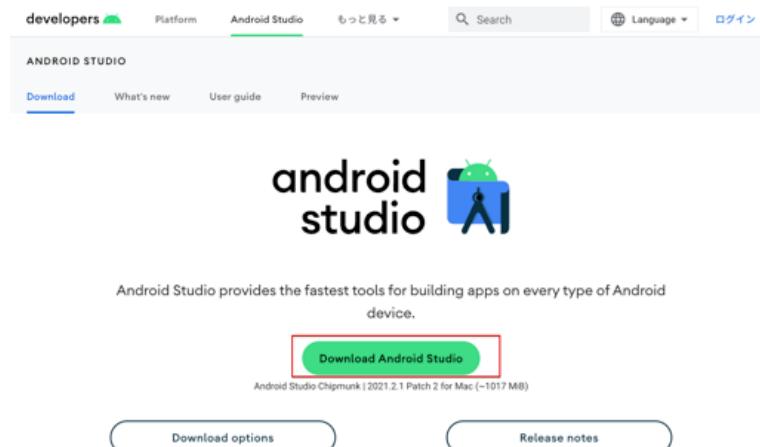
第9章

Android Studio

Android Studio は、Jetbarains の IDE を Google が Android アプリ開発用に提供しています。Android SDK のバージョン毎のインストールや Android Emulator の作成などができる、devTools なども含まれていますので、Android アプリ開発では必須といえます。

9.1 Android Studio のインストール

Android Studio のインストールは、Android Studio^{*1}サイトからインストーラをダウンロードできます。

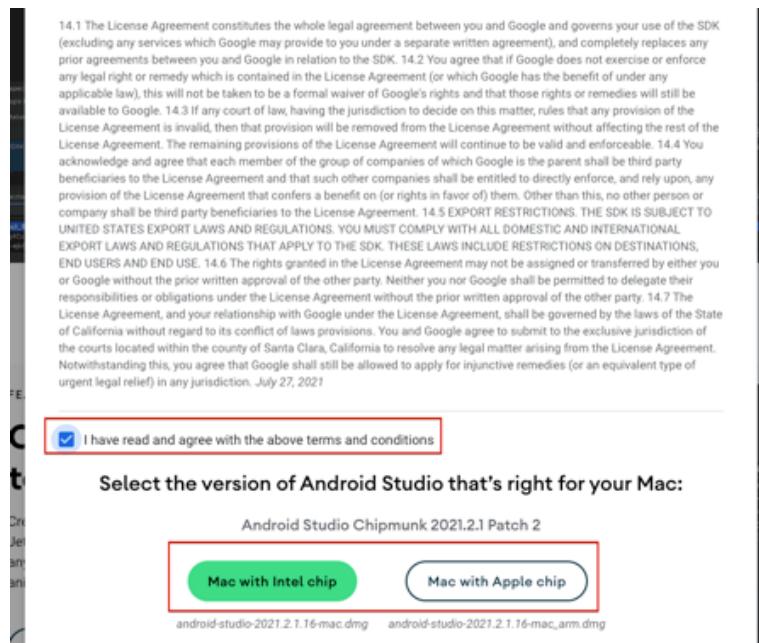


▲図 9.1: Android Studio のサイト

「Download Android Studio」ボタンをクリックすると「ライセンス許諾」が開きます。下まで

*1 <https://developer.android.com/studio>

スクロールし「同意する」チェックボックスにチェックをします。自分の Mac のチップに合わせ 「Intel chip」、「Apple chip」のボタンをクリックするとダウンロードが開始します。



▲図 9.2: ライセンス許諾

ダウンロードした「android-studio-2021.2.1.16-mac_arm.dmg」を開きます。



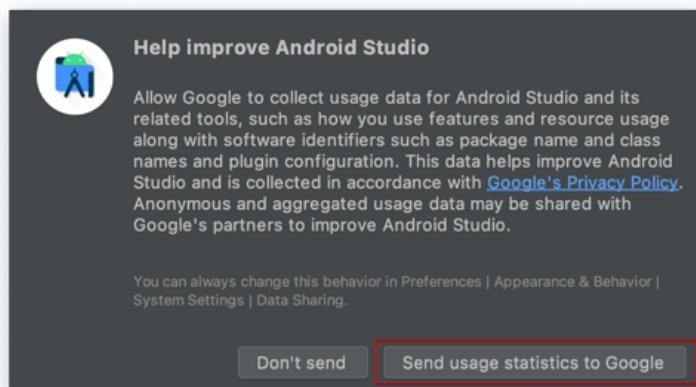
▲図 9.3: ダウンロードしたパッケージを開く

9.2

Android Studio のセットアップ

Launchpad などからインストールした「Android Studio」を開きます。

初回はセットアップ画面が起動します。最初に「使用状況を Google に送る」かを聞かれます。どちらを選択されてもかまいません。



▲図 9.4: Android Studio セットアップ

ここからセットアップの開始です。

インストールタイプ「スタンダード」、「カスタム」の選択があります。

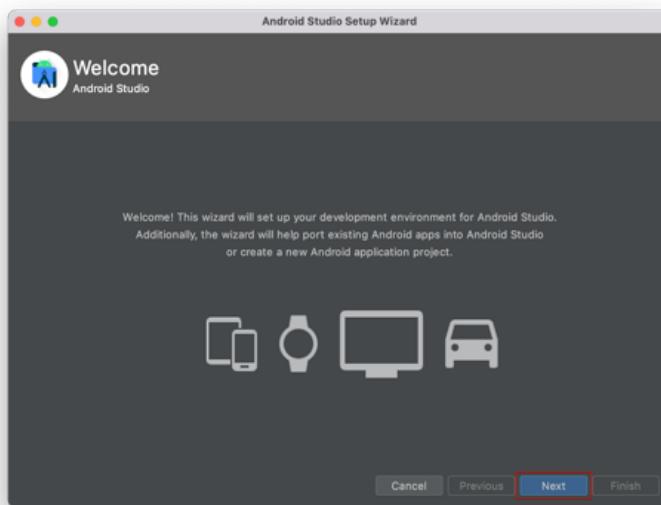
テーマの選択です。「ダーク」「ライト」がありますが、あとからでも変更できます。

確認画面になります。

「Andoloid-SDK-Licence」、「Android-SDK-arm-dbt-license」の確認画面です。赤ワクを選択し、それぞれ「Accept」を選択すると「Next」ボタンが有効になります。

必要なものがダウンロードされます。

セットアップ完了です。



▲図 9.5: Android Studio セットアップ

9.3 flutter doctor の確認

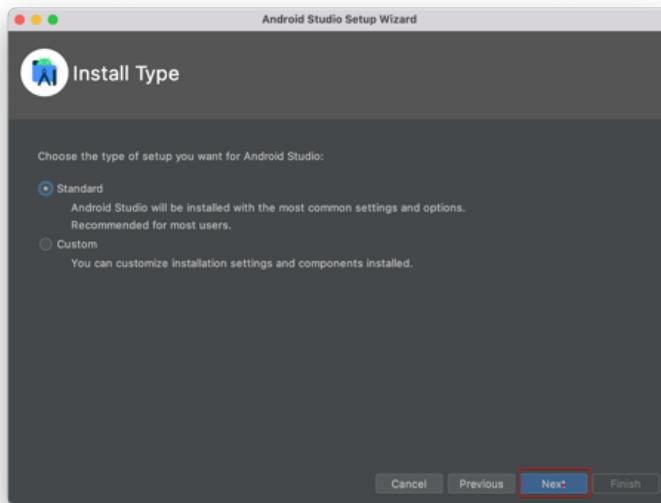
Android studio のインストールが完了しましたので、「flutter doctor -v」で確認します。

▼ flutter doctor -v

```
☒ flutter doctor -v
[✓] Flutter (Channel stable, 3.0.5, on macOS 12.5 21G72 darwin-arm, locale en-CA)
  • Flutter version 3.0.5 at /Users/ユーザー名/.asdf/installs/flutter/3.0.5-stable
  • Upstream repository [https://github.com/flutter/flutter.git](https://github.com/>/flutter/flutter.git)
  • Framework revision f1875d570e (5 weeks ago), 2022-07-13 11:24:16 -0700
  • Engine revision e85ea0e79c
  • Dart version 2.17.6
  • DevTools version 2.12.2

[!] Android toolchain - develop for Android devices (Android SDK version 33.0.0)
  • Android SDK at /Users/ユーザー名/Library/Android/sdk
 ☒ cmdline-tools component is missing
    Run `path/to/sdkmanager --install "cmdline-tools;latest"`
    See [https://developer.android.com/studio/command-line](https://developer.android.>com/studio/command-line) for more details.
  ☒ Android license status unknown.
    Run `flutter doctor --android-licenses` to accept the SDK licenses.
    See [https://flutter.dev/docs/get-started/install/macos#android-setup](https://flutte>ter.dev/docs/get-started/install/macos#android-setup) for more details.

[✓] Xcode - develop for iOS and macOS (Xcode 13.4.1)
```



▲図 9.6: Android Studio セットアップ

```
• Xcode at /Applications/Xcode-13.4.1.app/Contents/Developer
• CocoaPods version 1.11.3

[✓] Chrome - develop for the web
    • Chrome at /Applications/Google Chrome.app/Contents/MacOS/Google Chrome

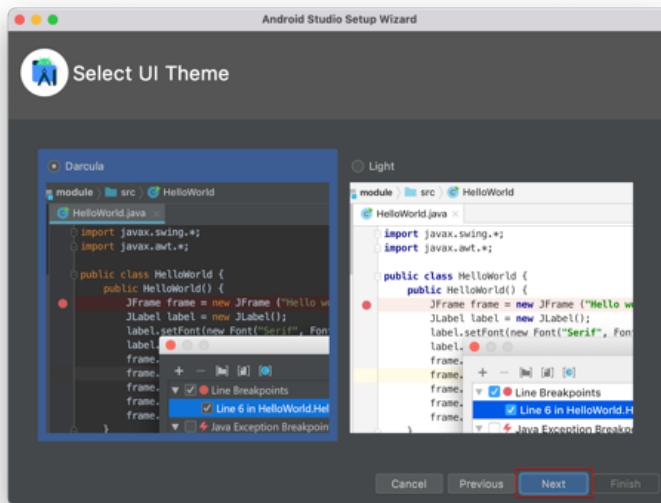
[✓] Android Studio (version 2021.2)
    • Android Studio at /Applications/Android Studio.app/Contents
    • Flutter plugin can be installed from:
      ☒ [https://plugins.jetbrains.com/plugin/9212-flutter](https://plugins.jetbrains.com/plugin/9212-flutter)
        • Dart plugin can be installed from:
          ☒ [https://plugins.jetbrains.com/plugin/6351-dart](https://plugins.jetbrains.com/plugin/6351-dart)
            • Java version OpenJDK Runtime Environment (build 11.0.12+0-b1504.28-7817840)

[✓] VS Code (version 1.70.1)
    • VS Code at /Applications/Visual Studio Code.app/Contents
    • Flutter extension version 3.46.0

[✓] Connected device (2 available)
    • macOS (desktop) • macos • darwin-arm64 • macOS 12.5 21G72 darwin-arm
    • Chrome (web)    • chrome • web-javascript • Google Chrome 104.0.5112.79

[✓] HTTP Host Availability
    • All required HTTP hosts are available

! Doctor found issues in 1 category.
```



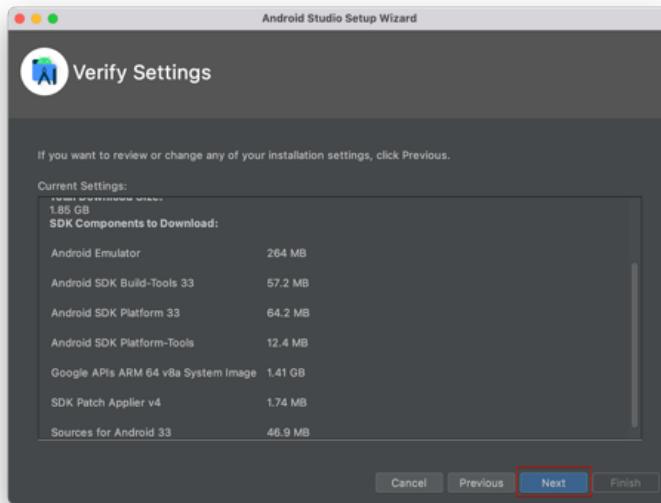
▲図 9.7: Android Studio セットアップ

Android Studio 欄の問題が解消されました。残りは、

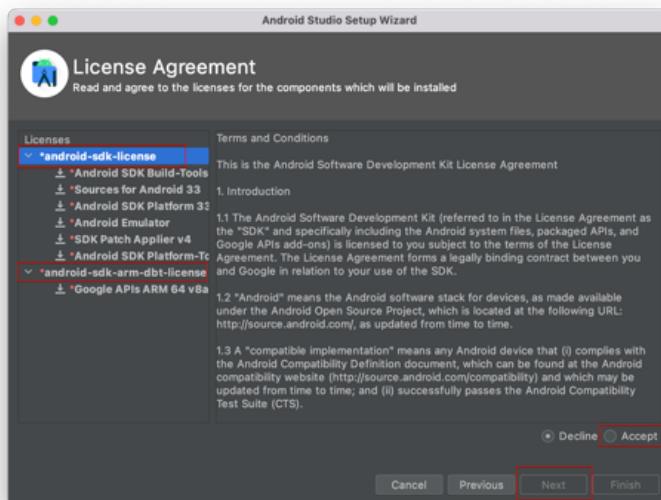
1. Android toolchain での cmdline-tools がない。
2. Android のライセンス状況が不明

と、2つ残されています。

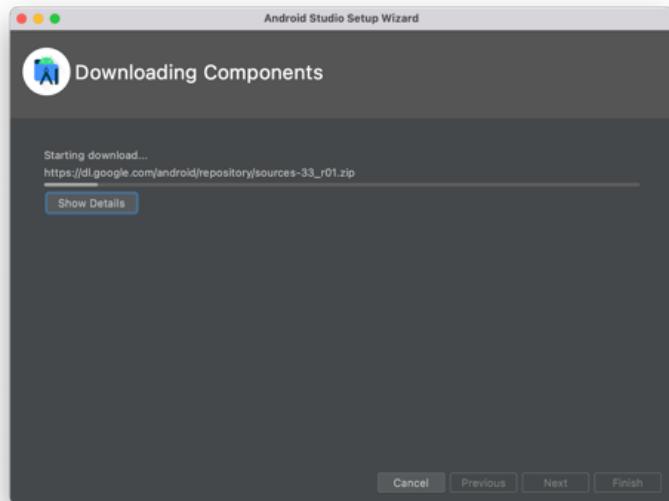
それでは、「cmdline-tools」をインストールします。



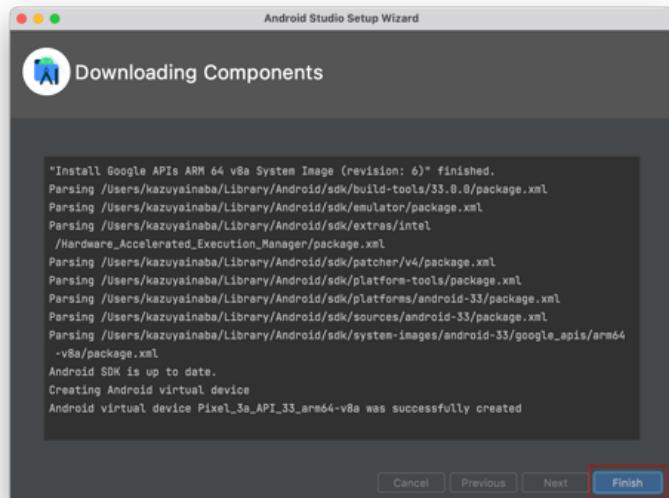
▲図 9.8: Android Studio セットアップ



▲図 9.9: Android Studio セットアップ



▲図 9.10: Android Studio セットアップ



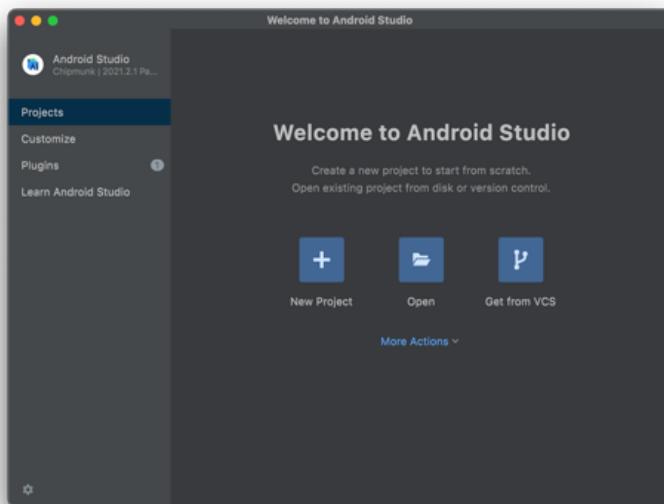
▲図 9.11: Android Studio セットアップ

第 10 章

command-line tools のインストール

10.1 cmdline-tools のインストール

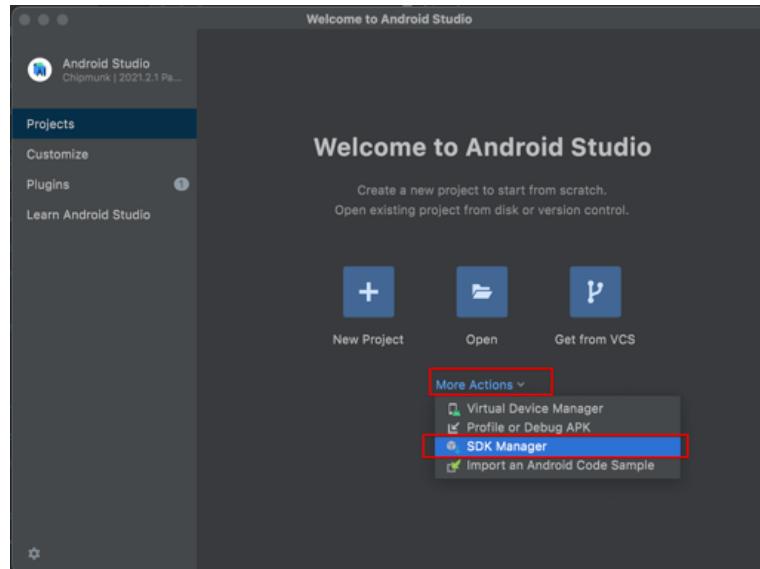
Android Studio を起動しますと初期画面が開きます。



▲図 10.1: cmdline-tools のインストール

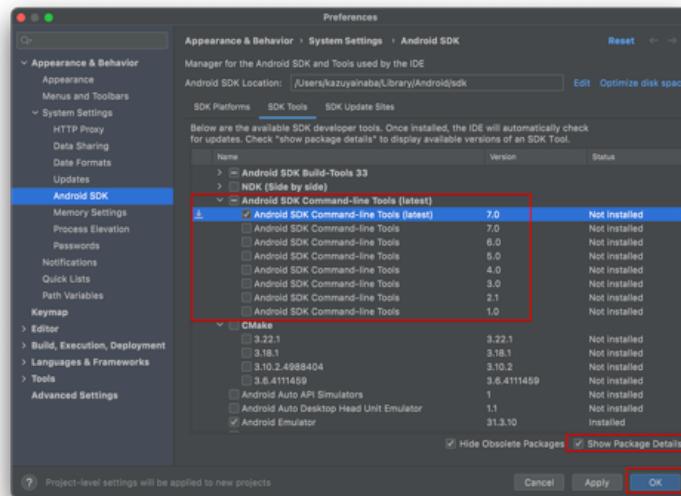
中央にある「More Actions」をクリックし、「SDK-Manager」を選択します。

右下の「OK」ボタンの上に「Show Package Details」チェックボックスがありますので、チェックします。「Android SDK Command-line Tools」の欄をクリックすると、インストール可能な



▲図 10.2: cmdline-tools のインストール

「cmdline-tools」のバージョンが表示されます。一番上の（latest）をチェックし、右下の「OK」ボタンをクリックするとインストールされます。



▲図 10.3: cmdline-tools のインストール

cmdline-tools のインストールが完了しましたら、パスを通します。「.zshrc」に以下を追加し、再読み込みします。

▼ .zshrc へ cmdline-tools のパスを通す

```
# Android
export ANDROID_HOME=$HOME/Library/Android/sdk
export PATH="$ANDROID_HOME/platform-tools:$ANDROID_HOME/emulator:$PATH"
export PATH="$ANDROID_HOME/cmdline-tools/latest/bin:$PATH"
```

▼ .zshrc の再読み込み

```
> source ~/.zshrc
```

「flutter doctor -v」で確認します。

▼ flutter doctor -v

```
✗ flutter doctor -v
[✓] Flutter (Channel stable, 3.0.5, on macOS 12.5 21G72 darwin-arm, locale en-CA)
  • Flutter version 3.0.5 at /Users/ユーザー名/.asdf/installs/flutter/3.0.5-stable
  • Upstream repository [https://github.com/flutter/flutter.git](https://github.com/>/flutter/flutter.git)
  • Framework revision f1875d570e (5 weeks ago), 2022-07-13 11:24:16 -0700
  • Engine revision e85ea0e79c
  • Dart version 2.17.6
  • DevTools version 2.12.2

[!] Android toolchain - develop for Android devices (Android SDK version 33.0.0)
  • Android SDK at /Users/ユーザー名/Library/Android/sdk
  • Platform android-33, build-tools 33.0.0
  • ANDROID_SDK_ROOT = /Users/ユーザー名/Library/Android/sdk
  • Java binary at: /Applications/Android Studio.app/Contents/jre/Contents/Home/bin/>
>/java
  • Java version OpenJDK Runtime Environment (build 11.0.12+0-b1504.28-7817840)
    ! Some Android licenses not accepted. To resolve this, run: flutter doctor --android-licenses

[✓] Xcode - develop for iOS and macOS (Xcode 13.4.1)
  • Xcode at /Applications/Xcode-13.4.1.app/Contents/Developer
  • CocoaPods version 1.11.3

[✓] Chrome - develop for the web
  • Chrome at /Applications/Google Chrome.app/Contents/MacOS/Google Chrome

[✓] Android Studio (version 2021.2)
  • Android Studio at /Applications/Android Studio.app/Contents
  • Flutter plugin can be installed from:
    ✗ [https://plugins.jetbrains.com/plugin/9212-flutter](https://plugins.jetbrains.c>om/plugin/9212-flutter)
  • Dart plugin can be installed from:
    ✗ [https://plugins.jetbrains.com/plugin/6351-dart](https://plugins.jetbrains.com/>plugin/6351-dart)
  • Java version OpenJDK Runtime Environment (build 11.0.12+0-b1504.28-7817840)

[✓] VS Code (version 1.70.1)
  • VS Code at /Applications/Visual Studio Code.app/Contents
  • Flutter extension version 3.46.0
```

```
[✓] Connected device (2 available)
  • macOS (desktop) • macos • darwin-arm64 • macOS 12.5 21G72 darwin-arm
  • Chrome (web)   • chrome • web-javascript • Google Chrome 104.0.5112.79

[✓] HTTP Host Availability
  • All required HTTP hosts are available

! Doctor found issues in 1 category.
```

あとひとつ、Android のライセンスに関しての問題が解消されていません。

10.2 Android ライセンス

「flutter doctor -v」で表示された通りターミナルに「flutter doctor --android-licenses」と入力しエンターキーを押します。

ライセンス許諾条項が表示されます。「accepted (y/N)」と聞かれますので、「y」を押します。すべてのライセンス許諾に「y」と対応して完了です。

▼ Android ライセンス

```
> flutter doctor --android-licenses
accepted (y/N)?
```

完了しましたら、再び「flutter doctor -v」で確認します。

▼ flutter doctor -v

```
☒ flutter doctor -v
[✓] Flutter (Channel stable, 3.0.5, on macOS 12.5 21G72 darwin-arm, locale en-CA)
  • Flutter version 3.0.5 at /Users/ユーザー名/.asdf/installs/flutter/3.0.5-stable
  • Upstream repository [https://github.com/flutter/flutter.git](https://github.com/flutter/flutter.git)
  • Framework revision f1875d570e (5 weeks ago), 2022-07-13 11:24:16 -0700
  • Engine revision e85ea0e79c
  • Dart version 2.17.6
  • DevTools version 2.12.2

[✓] Android toolchain - develop for Android devices (Android SDK version 33.0.0)
  • Android SDK at /Users/ユーザー名/Library/Android/sdk
  • Platform android-33, build-tools 33.0.0
  • ANDROID_SDK_ROOT = /Users/ユーザー名/Library/Android/sdk
  • Java binary at: /Applications/Android Studio.app/Contents/jre/Contents/Home/bin/java
  • Java version OpenJDK Runtime Environment (build 11.0.12+0-b1504.28-7817840)
  • All Android licenses accepted.

[✓] Xcode - develop for iOS and macOS (Xcode 13.4.1)
```

```
• Xcode at /Applications/Xcode-13.4.1.app/Contents/Developer
• CocoaPods version 1.11.3

[✓] Chrome - develop for the web
    • Chrome at /Applications/Google Chrome.app/Contents/MacOS/Google Chrome

[✓] Android Studio (version 2021.2)
    • Android Studio at /Applications/Android Studio.app/Contents
    • Flutter plugin can be installed from:
      [✗ [https://plugins.jetbrains.com/plugin/9212-flutter](https://plugins.jetbrains.com/plugin/9212-flutter)]
        • Dart plugin can be installed from:
          [✗ [https://plugins.jetbrains.com/plugin/6351-dart](https://plugins.jetbrains.com/plugin/6351-dart)]
            • Java version OpenJDK Runtime Environment (build 11.0.12+0-b1504.28-7817840)

[✓] VS Code (version 1.70.1)
    • VS Code at /Applications/Visual Studio Code.app/Contents
    • Flutter extension version 3.46.0

[✓] Connected device (2 available)
    • macOS (desktop) • macos • darwin-arm64 • macOS 12.5 21G72 darwin-arm
    • Chrome (web)   • chrome • web-javascript • Google Chrome 104.0.5112.79

[✓] HTTP Host Availability
    • All required HTTP hosts are available

- No issues found!
```

これで問題は、すべて解消されました。

次に、上記の「Android Studio」欄にある、

- Flutter plugin can be installed from …
- Dart plugin can be installed from …

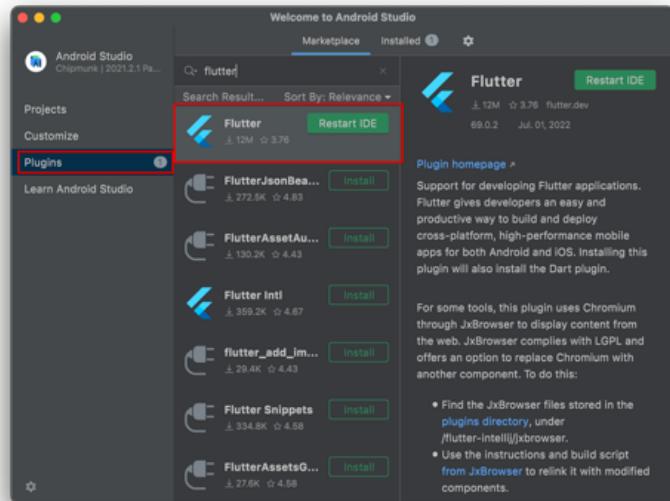
とありますので、Android Studio に 2 つのプラグインをインストールします。

10.3 Android Studioへのプラグインのインストール

Android Studio を起動し、初期画面の左ペインにある「plugin」をクリックします。検索欄に、「Flutter」、「Dart」と入力し検索します。

表示されたプラグインの「Install」ボタンをクリックします。

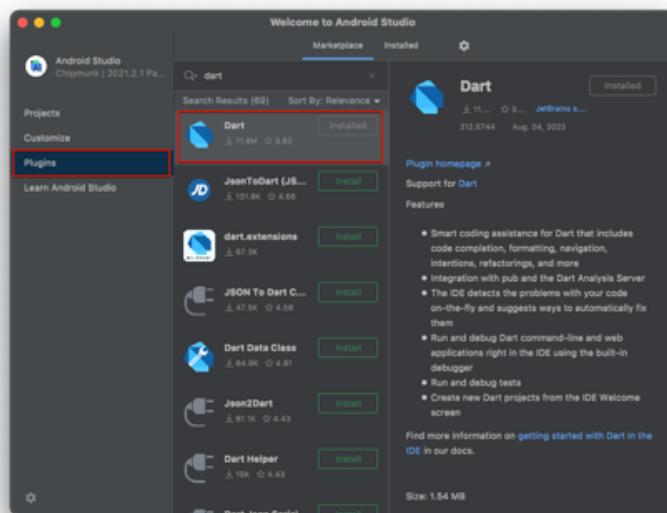
すべてのプラグインをインストールが完了すると、Android Studio を再起動します。



▲図 10.4: Flutter プラグインのインストール

以上で、Flutter 開発環境の作成が完了しました。

スタートアッププロジェクトを作成してみましょう。



▲図 10.5: Dart プラグインのインストール

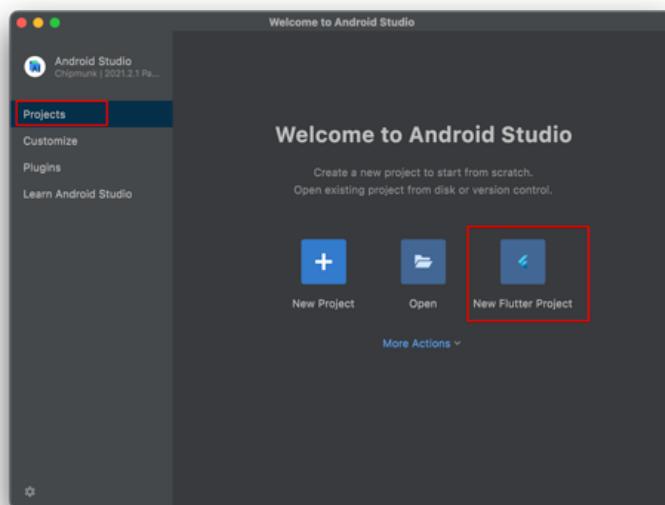
第 11 章

プロジェクトのスタート

Android Studio でスタートアッププロジェクトを作成します。
ウィザードに沿って作成するだけで、あっと言う間にモバイルアプリができあがります。

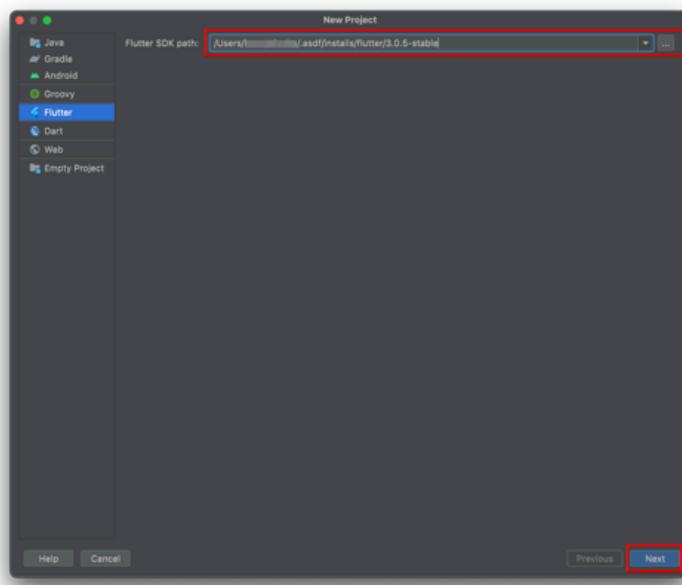
11.1 スタートアッププロジェクトを作る

Android Studio を起動します。Flutter プラグインをインストールしましたので、「New Flutter Project」を選択できるようになっています。



▲ 図 11.1: Android Studio 起動

「New Flutter Project」をクリックすると、「flutter SDK path」を入力、または選択します。



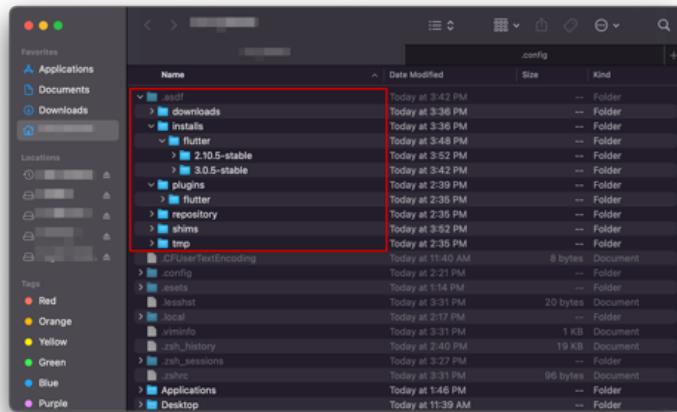
▲図 11.2: Flutter SDK path の指定

asdf を使用して Flutter をインストールした場合には、ユーザーホーム（/Users/ユーザー名/.asdf）以下のフォルダにあります。

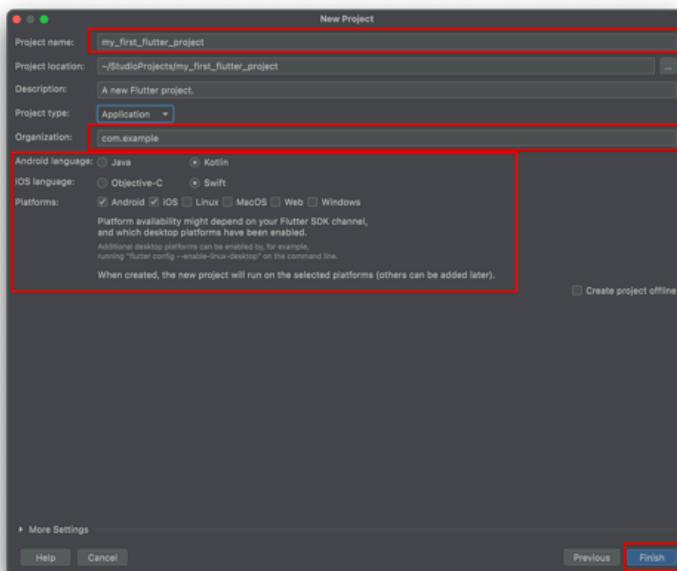
プロジェクトに関する情報を入力します。

- Project name: プロジェクトの名前（flutter は原則単語をアンダースコアでつなぐ）
 - Project location: プロジェクト保存先
 - Description: プロジェクトの詳細
 - Project type: Application/Plugin/Package/Module の選択があります。
 - Organization: Apple Developper に登録した Identifire
 - Android language: Android 用に使用するプログラミング言語 Java/Kotlin
 - iOS language: iOS 用に使用するプログラミング言語 Objective-C/Swift
 - Publication: 出力するプラットフォーム Android/iOS/Linux/MacOS/Web/Windows
- 以上の入力が完了しましたら、「Finish」ボタンをクリックします。

flutter コマンドが実行され、スタートアッププロジェクトが作成されます。後は、魔改造するだけです。



▲図 11.3: Flutter SDK のインストール場所



▲図 11.4: desc

次に、作成されたプロジェクトをエミュレータで動かしてみましょう。



▲図 11.5: desc

第 12 章

Android エミュレータ

Android OS で動作するスマートフォン・タブレットは、毎年たくさんの製品がリリースされます。それらは画面サイズが違います。

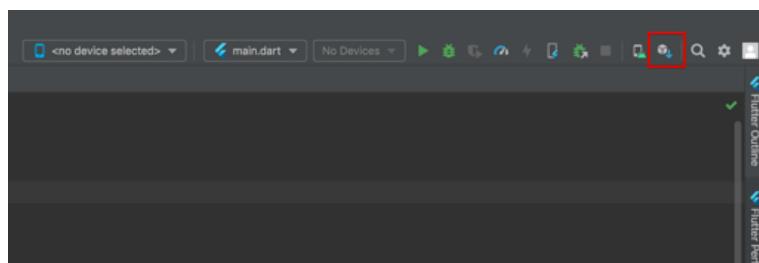
また、Android OS も毎年アップデートされます。しかし、すべての端末機の OS がアップデートされるわけではありません。

そのため、動作確認は、すべての Android OS で動作を確認し、すべての画面解像度で UI のチェックを行うべきですが、すべてを実機で検証するのは不可能でしょう。

しかし、仕様（OS のバージョン、画面解像度）に即して、できる限りの端末機でチェックできるように、エミュレータ（疑似端末）を使いテストします。

12.1 Android Platform

Android OS がエミュレータで実行できるように、各プラットフォームの SDK や image をダウンロードします。Andoroid Studio で右上のアイコンをクリックして SDK マネージャーを開きます。



▲ 図 12.1: desc

ダウンロードするものは、

Android SDK Platform

各バージョン用にアプリをコンパイルするために必要

System Image

Android エミュレータ上で実行するために必要

Sources for Android

プラットフォームのソースファイルがふくまれています。

各 Android のバージョンと API レベルは、以下の通りです。

Android 13

コードネーム:Tiramisu API レベル:33 リリース日:2022/8/15

Android 12、12L

コードネーム: S API レベル:31 リリース日:2021/10/4

コードネーム: Sv2 API レベル:32 リリース日:2022/3/7

Android 11

コードネーム: R API レベル:30 リリース日:2020/9/8

Android 10

コードネーム: Q API レベル:29 リリース日:2019/9/3

.....
Chip が Intel か Apple Silicon かでダウンロードする Image が違います。

エミュレータを動作させる、リリース用バイナリを作成する PC に搭載されている Chip に合わせてダウンロードします。

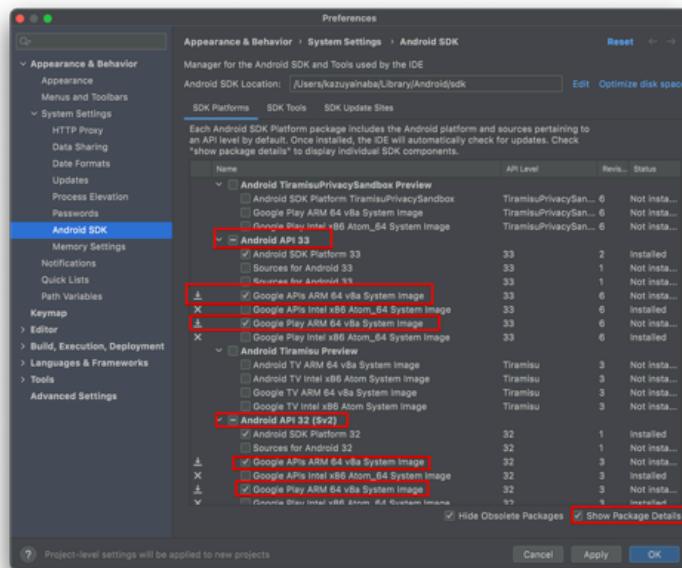
12.2 エミュレータの作成

エミュレータを作成するために、Android Studio の右上のデバイスマネージャーのアイコン、または、右横にあるデバイスマネージャータブをクリックします。

デフォルトで作成されているエミュレータを削除します。一番右の下向き矢印をクリックし、「Delete」を選択します。

「Create device」ボタンをクリックします。

新規デバイス作成画面が開きます。適当なデバイスを選択してください。今回は、Pixel5 を選択しました。「Next」ボタンをクリックします。



▲図 12.2: desc

デバイスに使用するイメージを選択します。

Pixel5 の販売時の仕様は、

- Android 11 -> API レベル 30
- 6.0inch フルスクリーン 1,080 x 2,340
- 8GB メモリ、128GB ストレージ

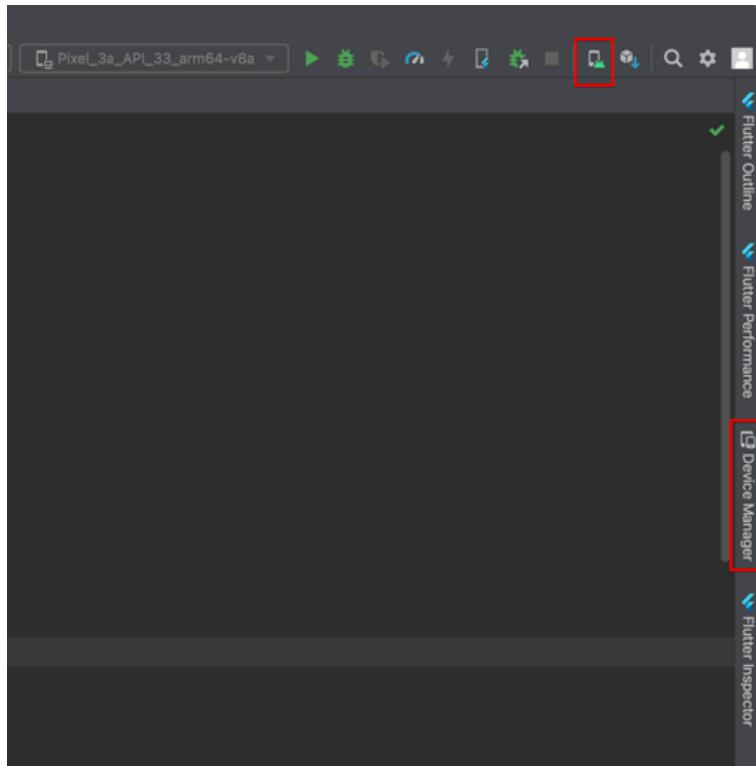
そのため、API レベル 30 を選択しています。「Next」ボタンをクリックします。

確認画面になりますが、「Show Advanced Settings」ボタンをクリックします。

メモリ、ストレージを 4GB に増やします。エミュレータの動作速度、ホスト PC のスペックなどのトレードオフがありますので自由に決めてください。

「Finish」ボタンをクリックします。

デバイスマネージャーに作成したエミュレータが表示されます。右向き三角アイコンで実行します。



▲図 12.3: デバイスマネージャーを開く

エミュレータウィンドウでエミュレータが起動します。歯車アイコンをクリックし、「View Mode」>「Float」を選択すると、エミュレータウィンドウを自由に移動させることができます。

エミュレータは、作成された時点では英語となっていますので日本語化します。マウスで下から上にフリックします。

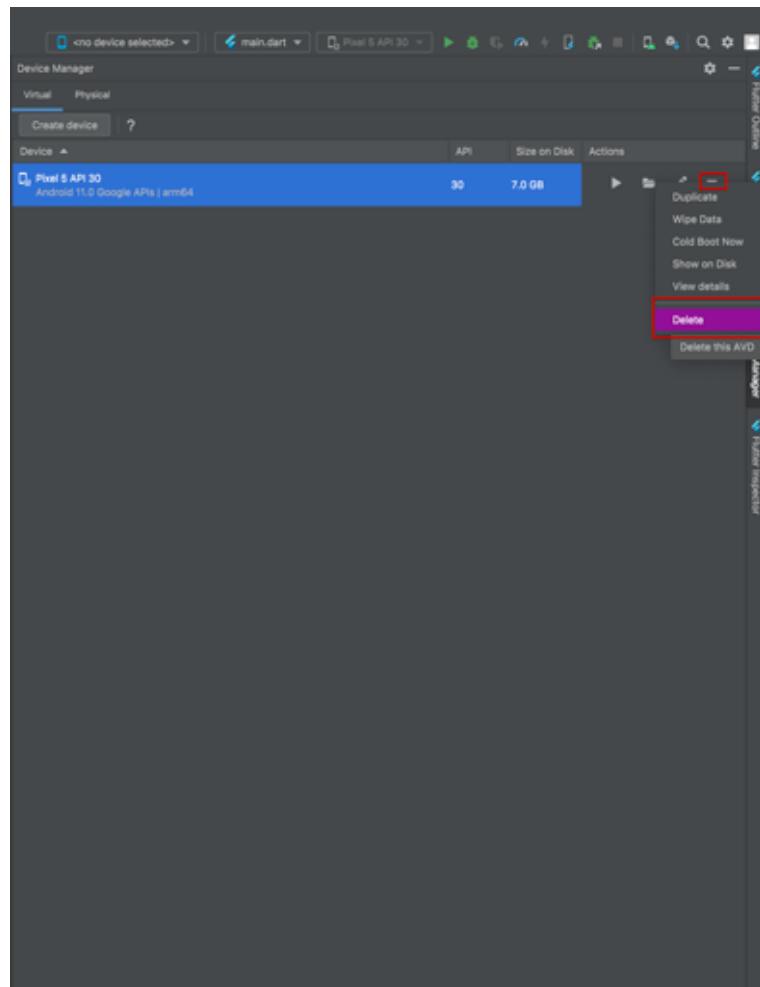
「Settings」を選択します。

「System」を選択します。

「Languages&input」を選択します。

「Languages」を選択します。

「Add a language」を選択します。

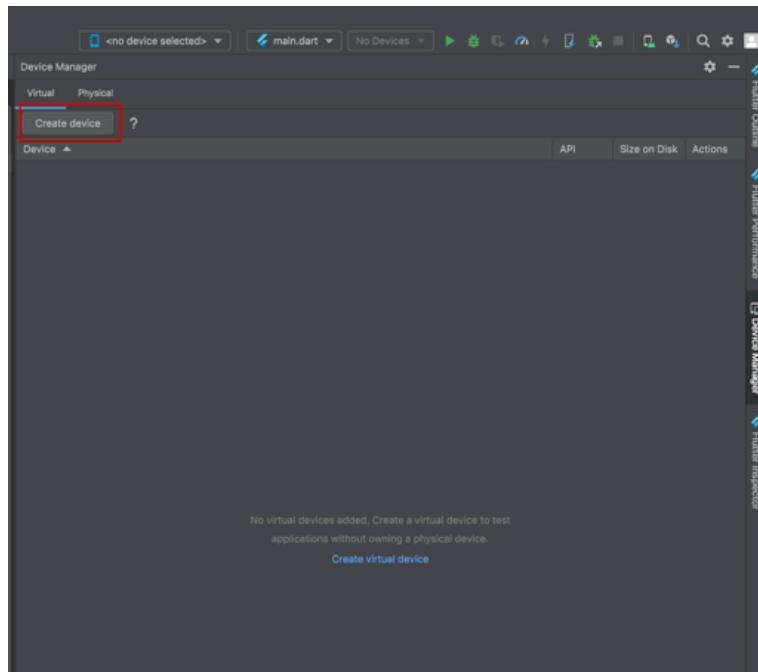


▲図 12.4: デフォルトのエミュレータを削除

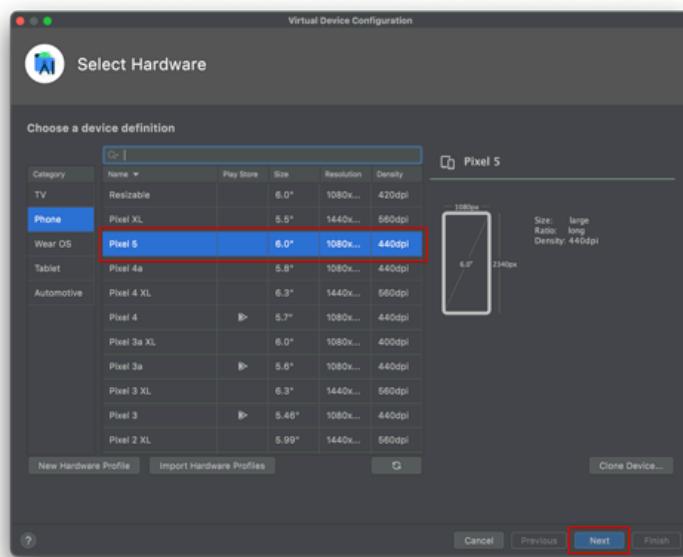
「日本語」を選択します。

追加された「日本語」を「English」の上に移動させます。

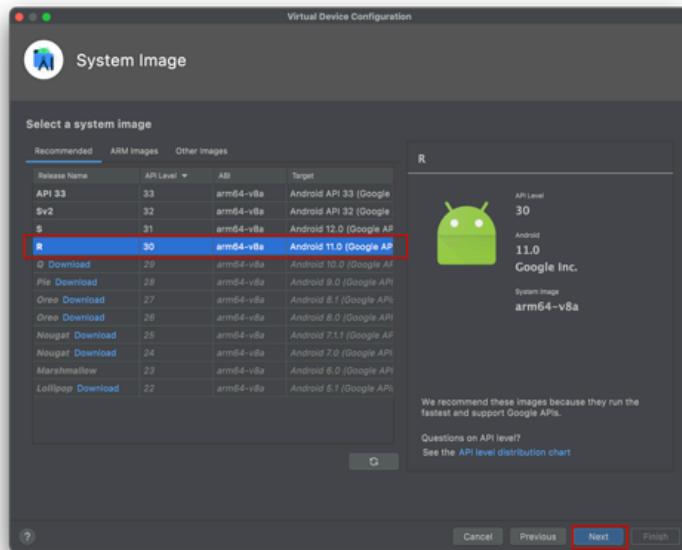
日本語化できました。



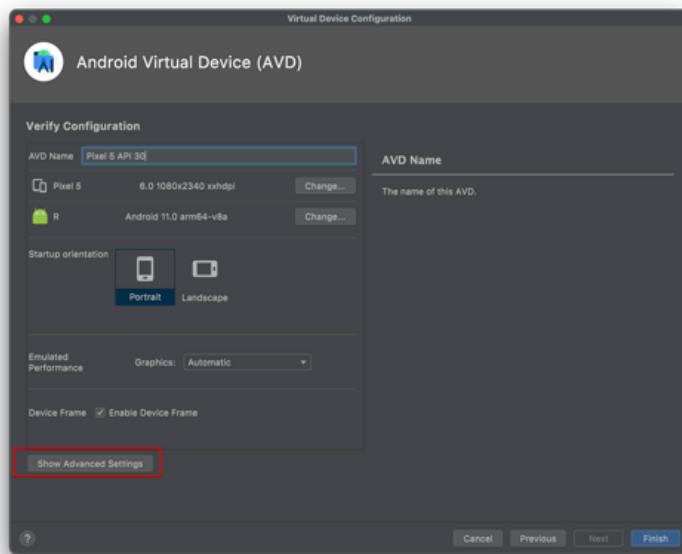
▲図 12.5: 新規エミュレータの作成



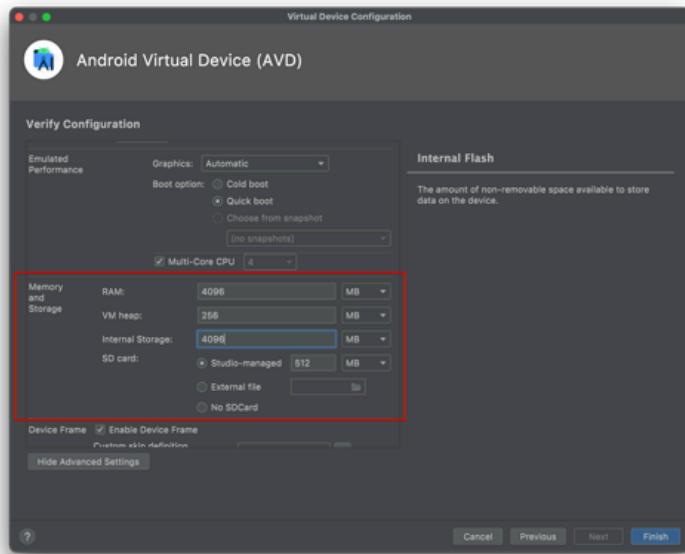
▲図 12.6: デバイスマネージャーを開く



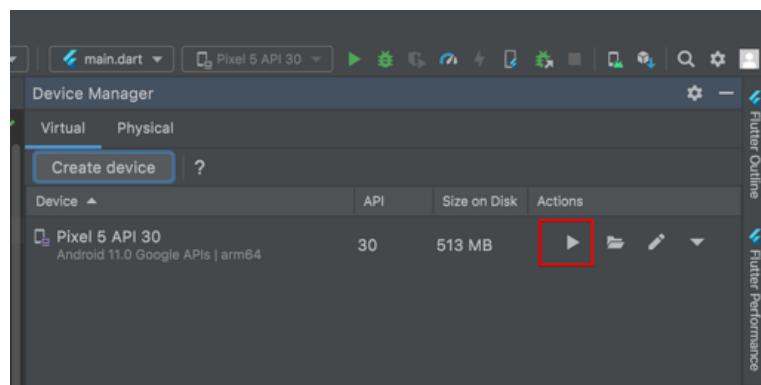
▲図 12.7: API レベルの選択



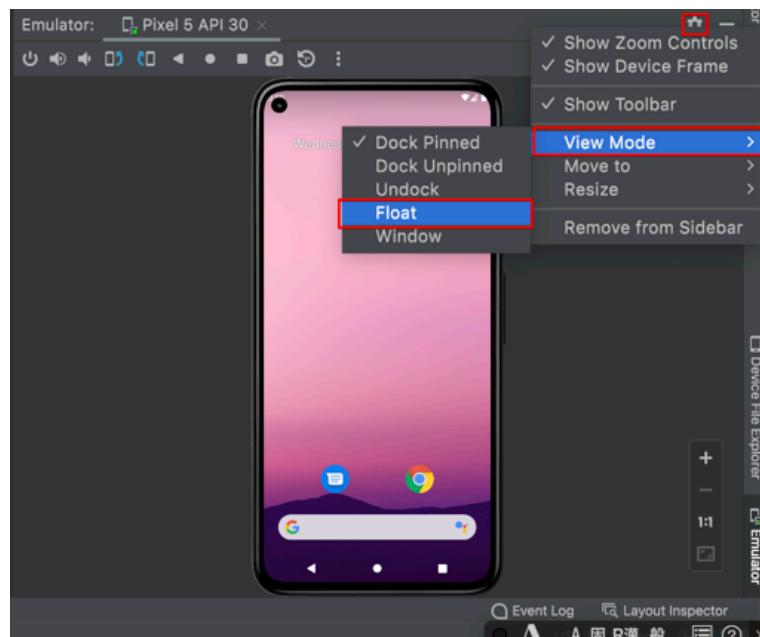
▲図 12.8: 設定を変更する



▲図 12.9: メモリ、ストレージを増やす



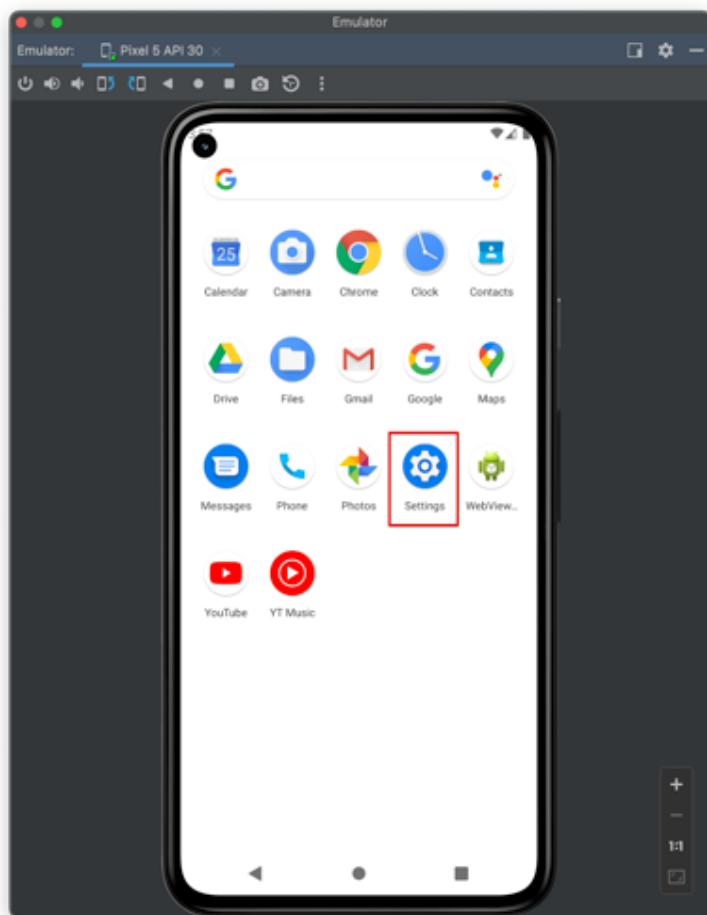
▲図 12.10: エミュレータを起動



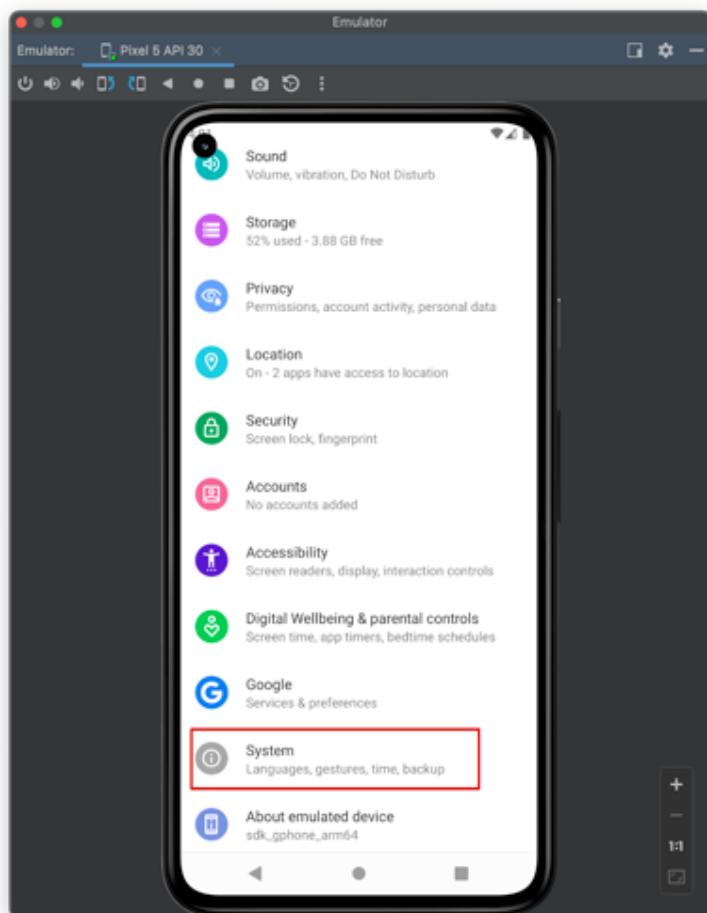
▲図 12.11: エミュレータウィンドウを FloatView で



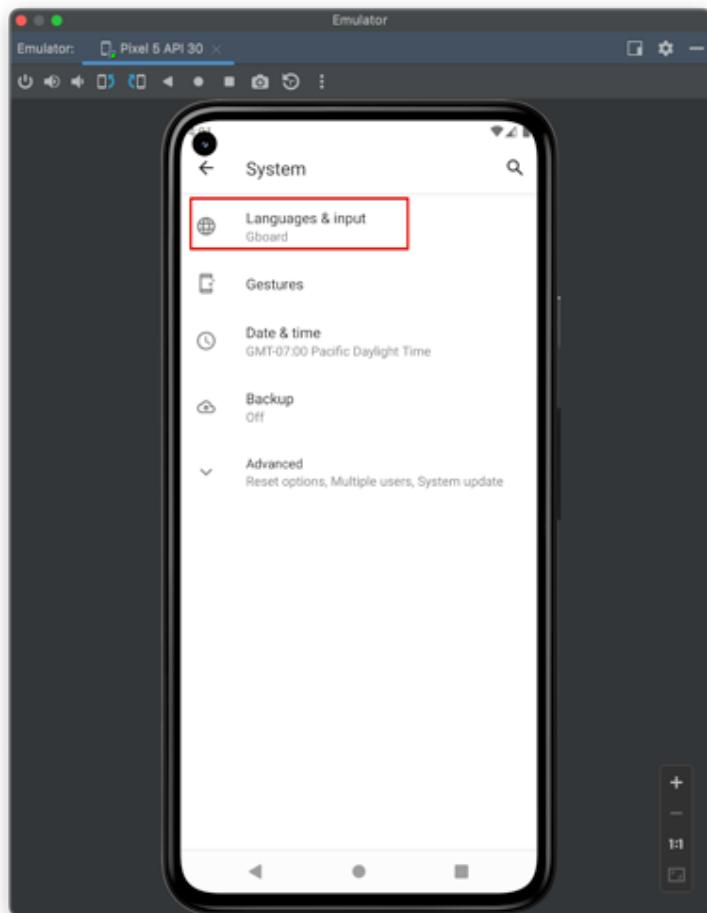
▲図 12.12: エミュレータを日本語化



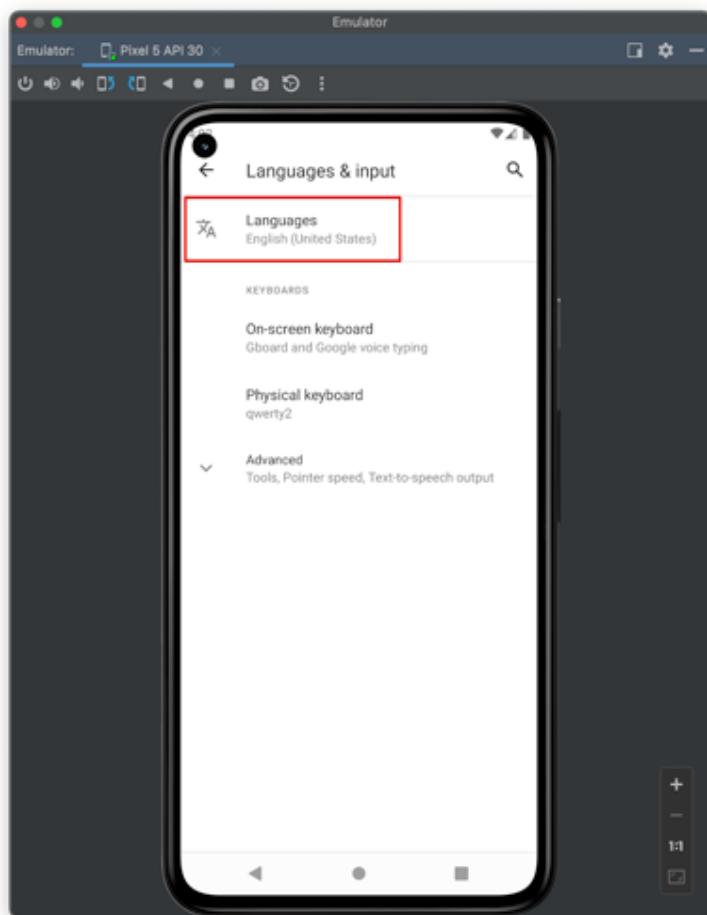
▲図 12.13: 設定アプリを選択



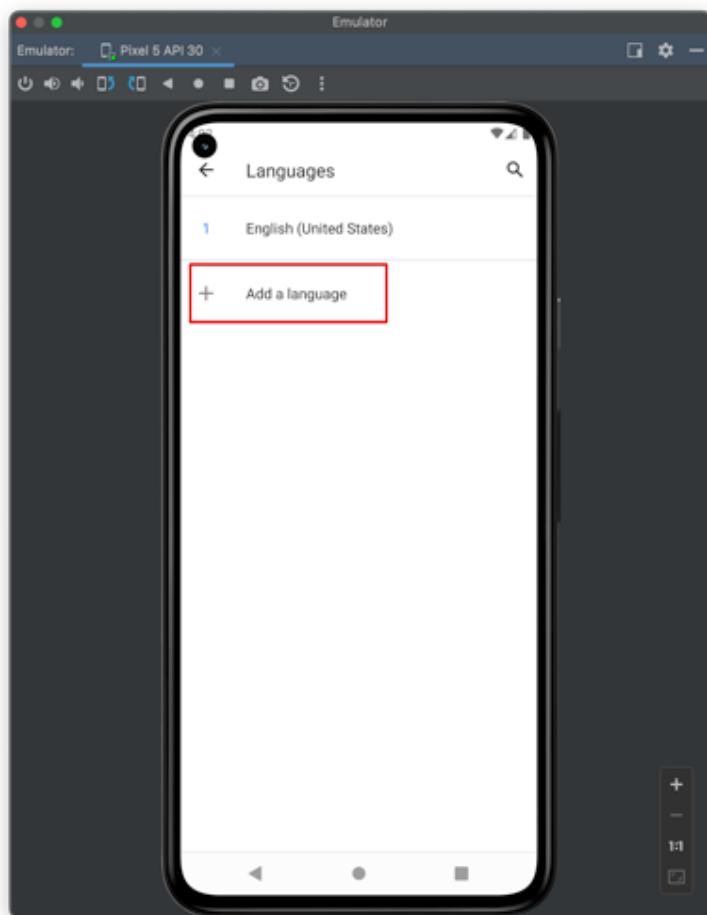
▲図 12.14: 設定からシステムを選択



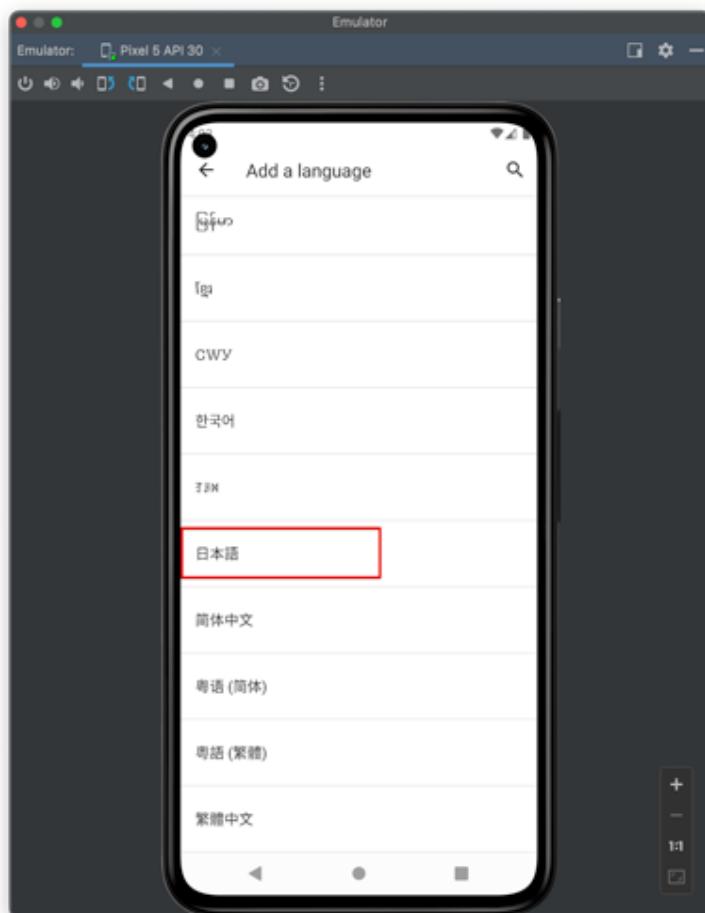
▲図 12.15: 言語と入力を選択



▲図 12.16: 言語を選択



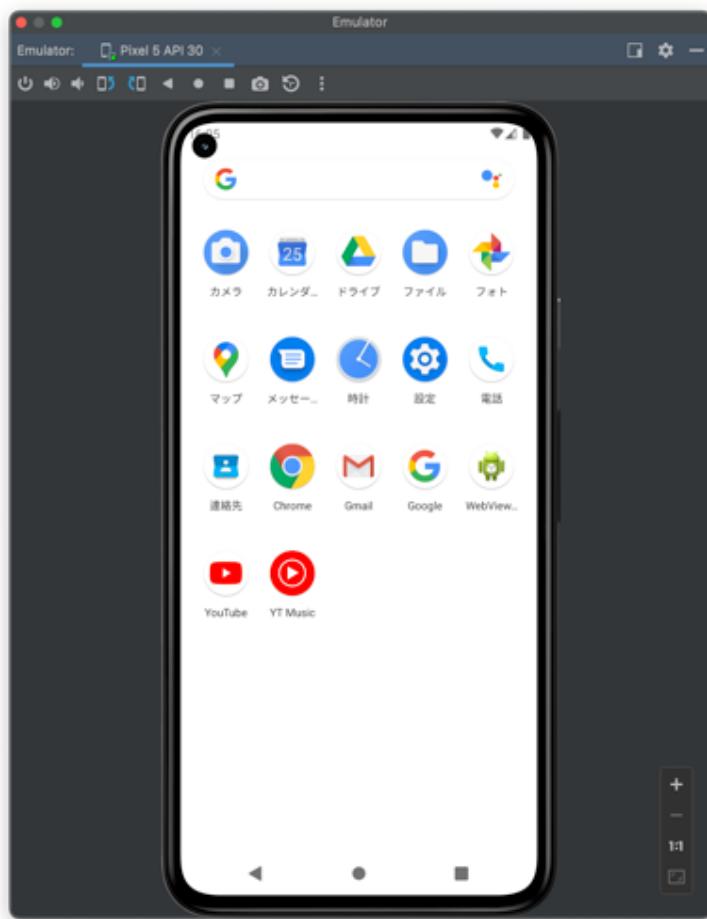
▲図 12.17: 言語を追加を選択



▲図 12.18: 日本語を選択



▲図 12.19: 日本語を優先



▲図 12.20: 日本語化完了

第 13 章

iPhone エミュレータ

iOS 用のデバイスも毎年リリースされ、複数の画面サイズが当たり前になりました。

これらもすべて実機でテストできれば良いのですが、仕様の範囲内でエミュレータを使って動作確認、デザイン確認を行います。

13.1 iOS エミュレータ

iOS エミュレータは、Xcode に含まれています。ターミナルに「open -a Simulator.app」を入力しエンターキーを押します。

▼ iOS エミュレータの起動

```
> open -a Simulator
```

コマンドが実行されると、エミュレータ（iOS ではシミュレータ？でも、本書ではエミュレータで統一します。）が起動します。Dock にあるエミュレータアイコンを Option で「Dock に追加」を行えば、次回からはワン・クリックで起動できます。

機種の切替は、メニューの「File」>「Open Simulator」>「iOS XX.X」>機種名で選択します。

13.2 以前の iOS バージョンの機種を作成

Xcode は、ダウンロードした時点での最新の iOS しか持っていません。旧バージョンのエミュレータを作成したい場合は、Xcode を起動し「Window」>「Devices and Simulators」を選択します。



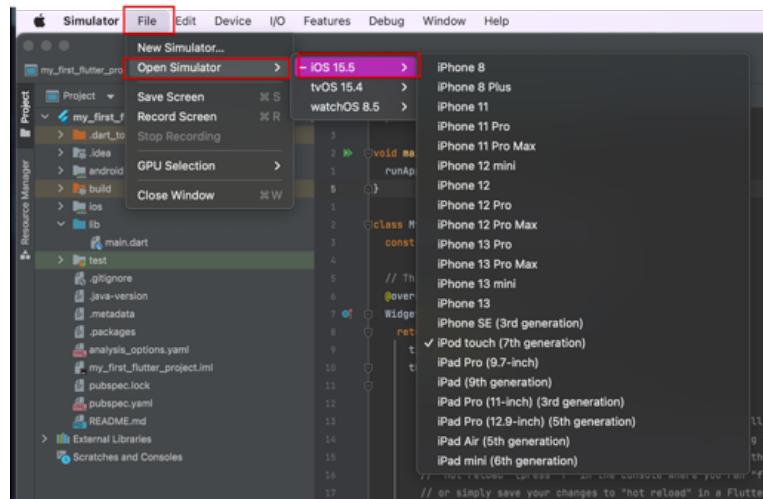
▲図 13.1: iOS エミュレータ

エミュレータウィンドウが開きますので、下の「+」をクリックします。

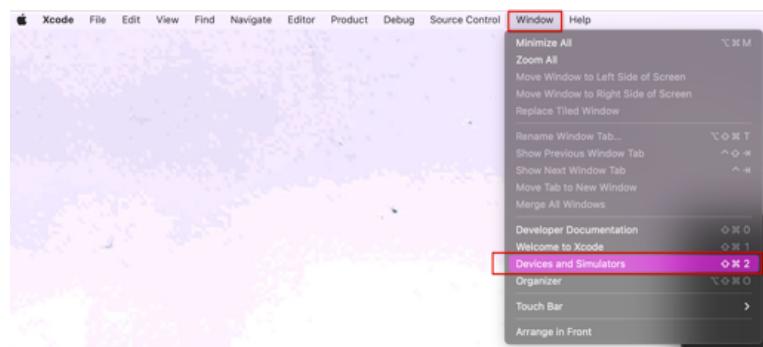
エミュレータの新規作成ウィンドウが開きますので、「Simulator Name」に作成する機種名を入力します。

「Device Type」のドロップダウンリストから作成する機種を選択します。

使用したい iOS のバージョンがない場合には、「Download more simulator runtimes」をクリッ



▲図 13.2: iOS エミュレータの機種変更



▲図 13.3: iOS エミュレータの旧バージョン機種作成

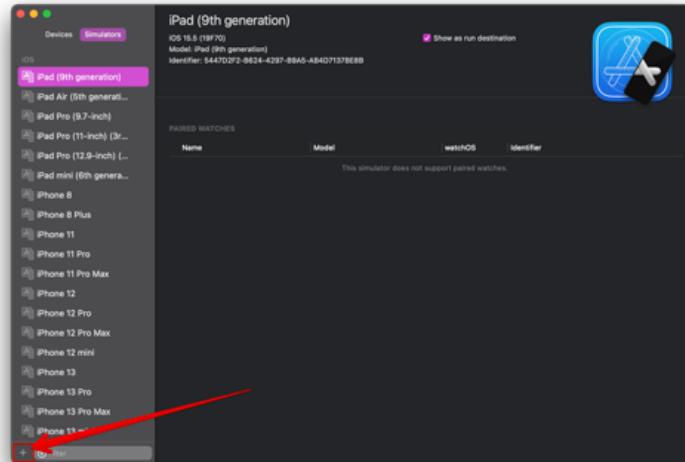
クします。

Components ウィンドウが開きます。使用したい iOS Simulator 左のアイコンをクリックするとダウンロードが開始されます。

ダウンロードは、結構時間がかかります。

ダウンロードが完了すると、機種が作成されています。

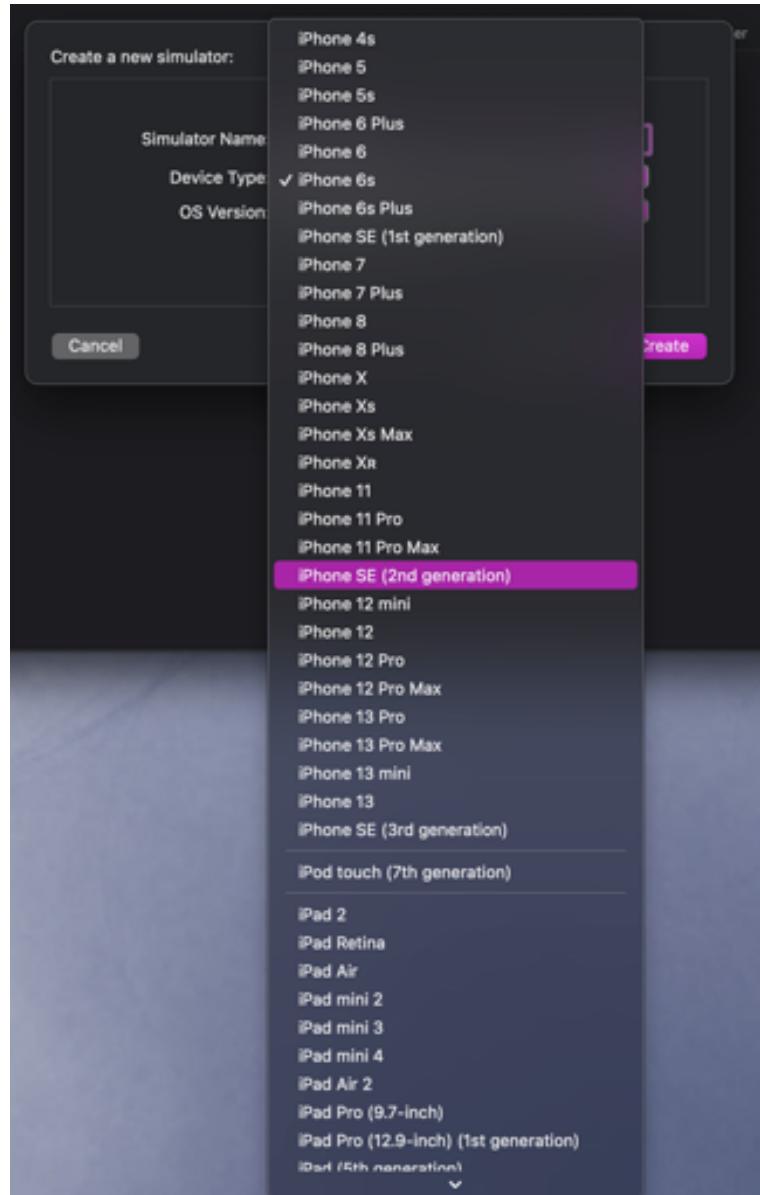
ダウンロードした旧バージョンは、次回からは選択できます。



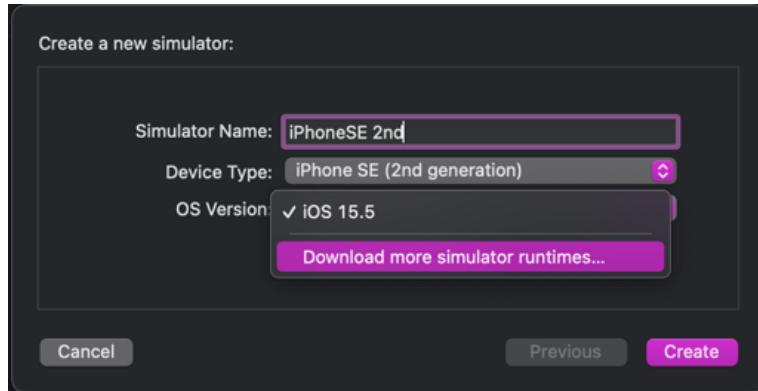
▲図 13.4: iOS エミュレータウィンドウ



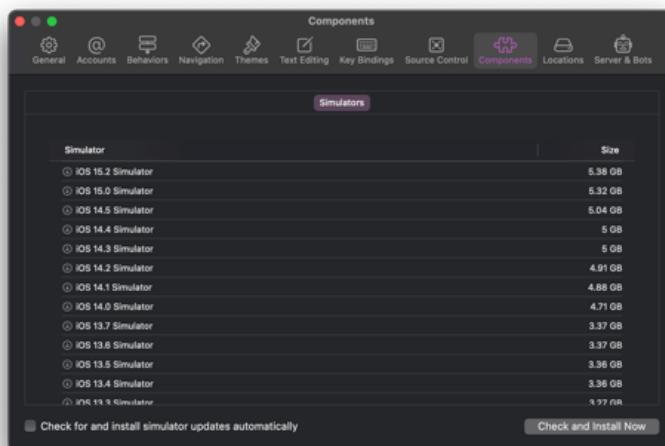
▲図 13.5: 新規 iOS エミュレータの作成



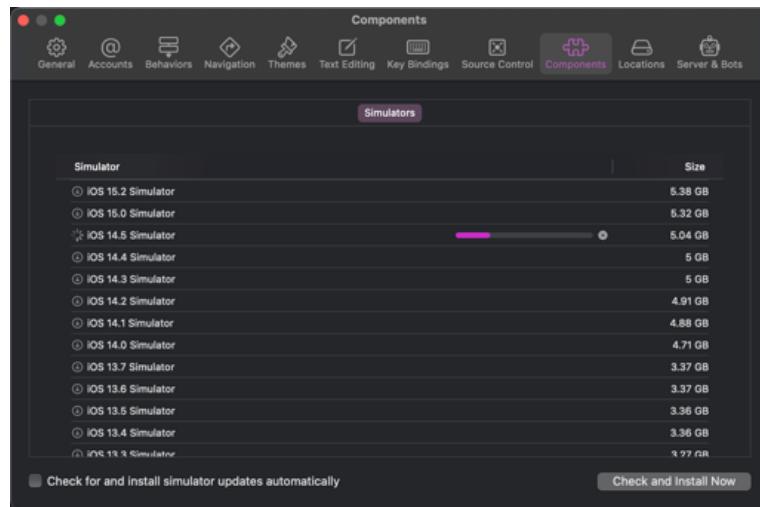
▲図 13.6: 新規 iOS エミュレータのデバイスタイプ



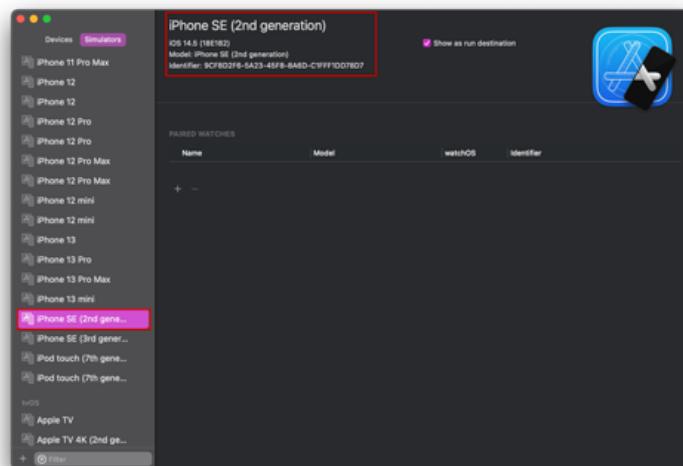
▲図 13.7: 新規 iOS エミュレータでの OS のバージョン選択



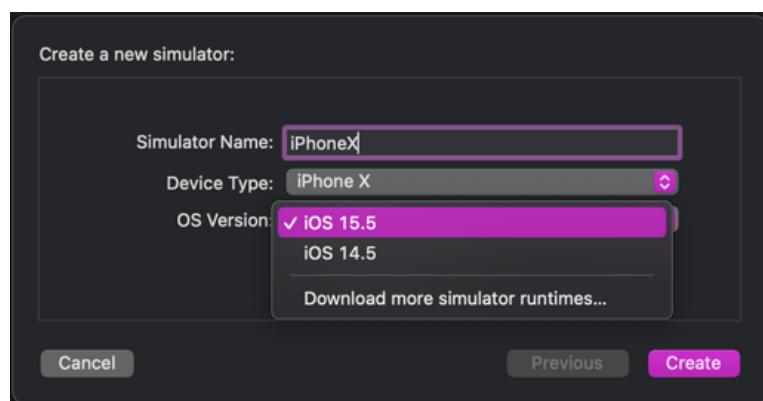
▲図 13.8: iOS 旧バージョンダウンロード



▲図 13.9: 旧バージョンダウンロード中



▲図 13.10: 旧バージョンでの機種作成完了



▲図 13.11: iOS バージョンを指定しての機種作成

第 14 章

エミュレータでデバッグ

それでは、エミュレータを使ってスタートアッププロジェクトを動かしてみましょう。
実機やエミュレータを使い、動作確認し不具合を見つけて修正することをデバッグと言います。
ボタンクリックひとつで簡単に開始できます。

14.1 Android エミュレータ

デバイスマネージャーからエミュレータを起動します。

Android Studio の右上にあるエミュレータから対象を選択し、右向き三角のアイコンをクリックします。

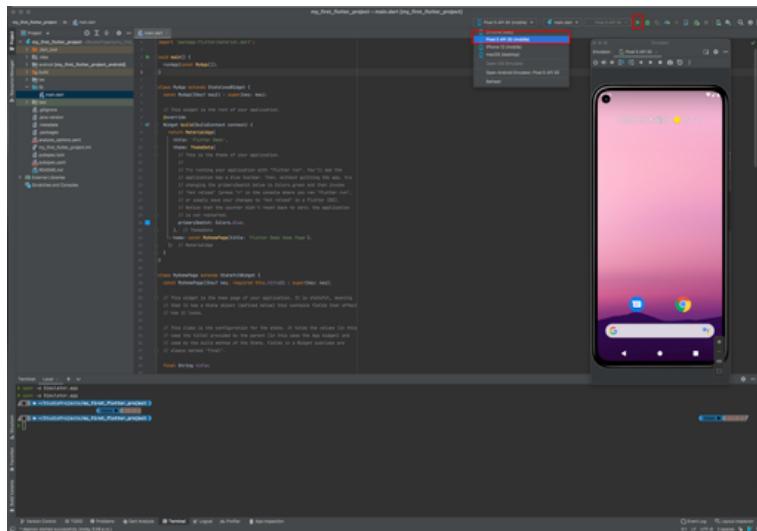
プログラムの解析が行われ、エラーがない場合にはプログラムがスタートします。エミュレータに表示された「+」ボタンをクリックすると表示されている数字が増えています。

デバッグの終了は、先ほどデバッグ開始時にクリックした右向き三角付近の「赤い四角」アイコンをクリックします。

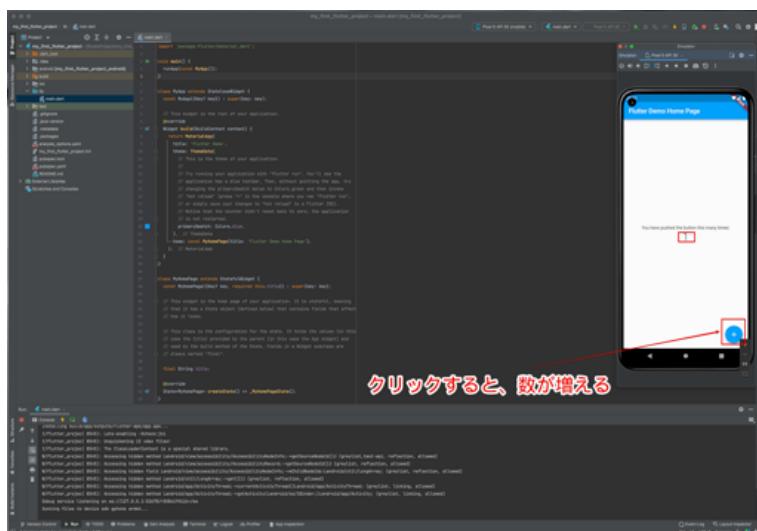
14.2 iPhone エミュレータ

Android Studio で、デバッグ対象に iPhone を選択します。

iPhone でプログラムが開始され表示されます。同じように「+」ボタンをクリックすると表示している数が増えます。



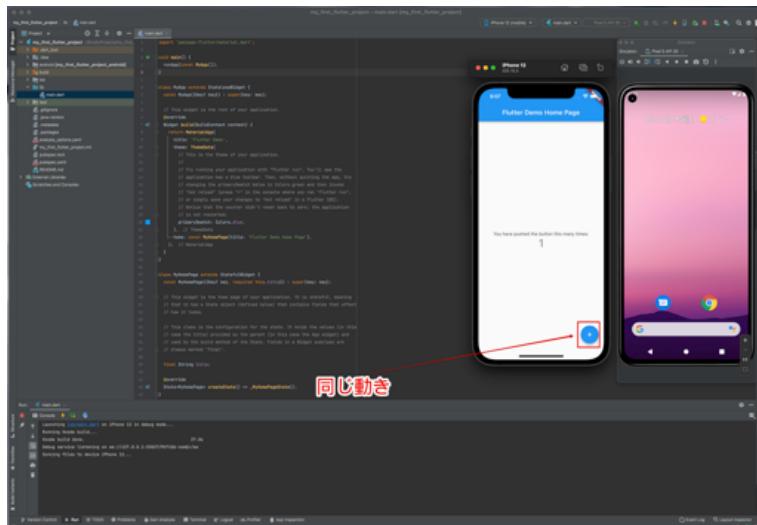
▲図 14.1: Android エミュレータでデバッグ開始



▲図 14.2: Android エミュレータでデバッグ中



▲図 14.3: iPhone エミュレータでデバッグ開始



▲図 14.4: iPhone エミュレータでデバッグ中

第 15 章

おわりに

いかがでしたでしょうか？

開発環境の作成ができれば、スタートアッププロジェクトをエミュレータで動作させるまでは、数クリックです。

しかも、ひとつのソースコードから、Android アプリ、iOS アプリが同じように動きました。さらに、

- Linux デスクトップ
- macOS デスクトップ
- Windows デスクトップ
- Web アプリケーション

も、同じソースコードから出力できます。

もし、Macをお持ちでしたら、あなたも Flutter でモバイルアプリ、デスクトップアプリを作成してみませんか？ すべては無料で始めることができます。

Flutter で Android、iPhone アプリ開発

開発環境作成 Mac 編

2022 年 8 月 29 日 ver 1.0 (技術書典 13)

著 者 今北産業

© 2022 今北産業