SwiftUI 対応

たった２日でマスターできる

iPhone アプリ開発 集中講座

Xcode 12 | iOS 14 | 対応

藤治仁・小林加奈子・小林由憲［著］

ソシム

はじめに

この書籍は、**「iOS アプリを作ってみたい、すべての初心者が、体験から学べる入門書」**です。

iOS とは、Apple が macOS をベースに開発した、iPhone、iPad、iPod touch 向けのモバイル OS です。本書籍では、プログラム言語の Swift（スウィフト）を用いて開発を行います。

執筆陣は、2014 年 11 月 1 日から**「Swift ビギナーズ倶楽部」**というコミュニティを開催していました。プログラミング自体がはじめての方、iOS アプリを作ってみたい他分野のエンジニアの方、そして高校生から、定年退職された方まで、たくさんの方々にご参加いただきました。最初は試行錯誤しながらも、勉強会に参加をして参加者の方どうしで教え合い、アプリ開発を楽しまれている方々が多くいらっしゃいます。

その経験を通して執筆陣が理解したことは、はじめての分野を学習する際には、経験者にとってはどんな些細なことも、初学者にとってはつまずくポイントになりえるということです。ただ、どんな分野もそうですが、最初は基礎的なことを、まねごとでよいので丁寧に繰り返し練習すれば、少しずつ自分だけのオリジナルの発想が出てきます。

本書では、構成段階からハンズオンセミナーを開催し、たくさんの参加者の方々の声をまとめ、各レッスンを構成しました。また、継続してセミナーを開催しながら、どの操作でつまずくのか、どのコードの説明が理解しがたいのかをフィードバックを受けて、教材を改善し続けています。

そうした調査に基づき、プログラミングを通して、**「モノづくり」の楽しさを体験**していただけるように、少しずつ階段を上っていく体験を重視した構成にしました。初心者が最初の一歩を踏み出す書籍を目指しています。

本書のコンセプトは**「まずは体験してみて、その経験を生かして学んでいく」**です。難しいプログラミング文法の説明は極力最小限にまとめ、多くのサンプルアプリの開発を体験してもらうことで、最短距離でアプリ開発の「勘所」がつかめるように工夫しました。

これから iOS アプリを制作される方々の一助となれば幸いです。ようこそ、iOS アプリ開発の世界へ。

目次 CONTENTS

Day 1 Day 2

Lesson 1 タイマーアプリを作ろう
―画面遷移とデータの永続化― 257

いままで単一画面で完結するアプリを作ってきましたが、この章では画面遷移の方法を学びます。
また、設定画面で選択した秒数をタイマー画面で利用できるように、設定画面でデータを保持しておく必要があるので、その方法も学びましょう。

Lesson 4 お菓子検索アプリを作ろう
—Web API と JSON の使い方を学ぶ— 405

iOS アプリでインターネットを通してデータを取得できると、作りたいアプリの可能性が広がります。
お菓子に関するキーワードが入力されたら、インターネットからお菓子の情報を取得し、アプリの画面に表示するお菓子検索アプリを作ります。

この本の読み方と使い方

本書が対象とする方

- プログラムを書いたことはないけれど、iPhone/iPad アプリを作ってみたい方
- iPhone アプリをよく利用していて、自分でも作ってみたいと思った方
- 中高生、大学生で iPhone アプリ開発を学んでみたい方
- シルバー世代や中高年の方で再学習を実施したい方
- 企業で入社前研修や企業導入研修での教材を検討している方

そんな iOS アプリを作ってみたい、すべての初心者が対象です。

アプリを作ることを「開発」するともいいます。開発といっても「難しいことをする！」と身構える必要はありません。プログラミングを楽しみながら、リラックスして読み進めてください。

本書でできるようになること

初心者の方もサンプルアプリを作ることにより、動く体験と基本の知識が身につくようになります。

この書籍を終えるころには、他の入門書やプログラミング文法書を読む力もついていると思います。そして、作りたいアプリや学習したい分野も見えてくると思いますので、ぜひ、次の書籍を購入してステップアップを目指してください。

本書の特徴

とにかく**「体験」**すること、そしてあとから**「理解」**することに重点を置いています。

本書では、プログラミングの文法説明は最小限にして、iPhone/iPad アプリを作って動かしていくことを目的として構成しています。

プログラミング文法書のように文法を理解して覚えるのではなく、どんどんアプリを作って体験していくことに比重を置いています。プログラミングがはじめての人でも楽しみながら iOS アプリが作れるという体験ができるように工夫しました。

学習が進めやすいように、学校の授業のように時限制（レッスン）で区切っています。各レッスンごとに独立したサンプルアプリが作れるように配慮していますので、制作したいサンプルアプリがあれば、途中からでも学習できます。

まったくの初心者の方は、読み飛ばさずに最初からじっくりと取り組んでみてください。少しでも経験のある方は、作りたいサンプルアプリのレッスンからはじめるのもよいでしょう。

本書の構成

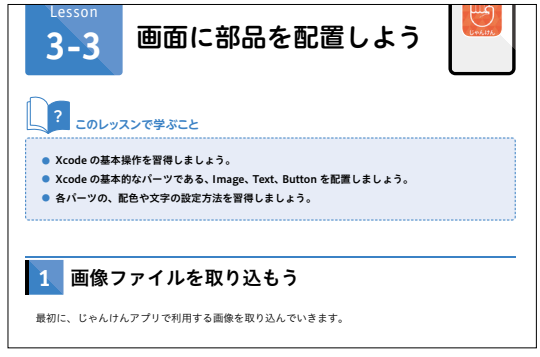
Day1（1 日目）はレッスン 5 まであり、iPhone アプリ制作の概論と開発の準備から入ります。そして、ここで「じゃんけんアプリ」「楽器アプリ」「マップ検索アプリ」の 3 つのアプリを作ります。「アプリを作って動かすことができた！」という体験を得てください。

Day2（2 日目）はレッスン 4 まであります。サンプルアプリは「タイマーアプリ」「カメラアプリ（前半）」「カメラアプリ（後半）」「お菓子検索アプリ」を作ります。

本書の読み方とページ構成

① このレッスンで学ぶこと

レッスンの中で学べることをピックアップして予測できるようにわかりやすくしています。

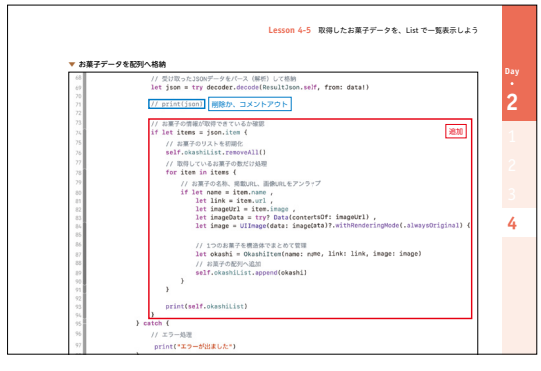


② Xcode 画面

Xcode（エックスコード）は、iOS アプリを視覚的に開発するための統合開発環境です。Xcode 画面は、操作が理解しやすいようにナンバリングやコメントを入れています。

③ プログラムコードの追加

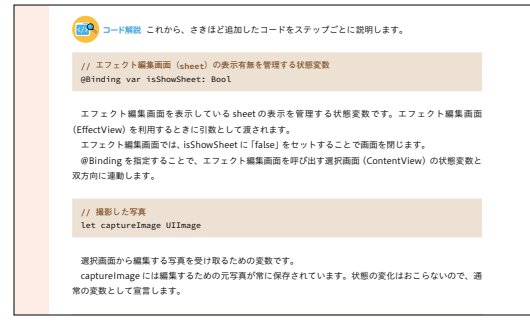
プログラムのコードを追加する場所は、わかりやすいように赤枠で囲っています。また、削除の場合は青枠で囲っています。



④ 説明用のプログラムコード

Swift のプログラムコードを掲載しています。コードは理解しやすいように、1 行から数行の塊で説明します。

※ Swift（スウィフト）は、iOS アプリを作るためのプログラミング言語です。


コード解説 これから、さきほど追加したコードをステップごとに説明します。

```
// エフェクト編集画面（sheet）の表示有無を管理する状態変数
@Binding var isShowSheet: Bool
```

エフェクト編集画面を表示している sheet の表示を管理する状態変数です。エフェクト編集画面（EffectView）を利用するときに引数として渡されます。
エフェクト編集画面では、isShowSheet に「false」をセットすることで画面を閉じます。
@Binding を指定することで、エフェクト編集画面を呼び出す選択画面（ContentView）の状態変数と双方向に連動します。

```
// 撮影した写真
let captureImage UIImage
```

選択画面から編集する写真を受け取るための変数です。
captureImage には編集するための元写真が常に保存されています。状態の変化はおこらないので、通常の変数として宣言します。

⑤ Point（ポイント）、Tips（ティップス）、Column（技術コラム）

Point

Point（ポイント）は、いまの学習の中で知っておいたほうがよいことや覚えるべき箇所を記載しています。

Tips

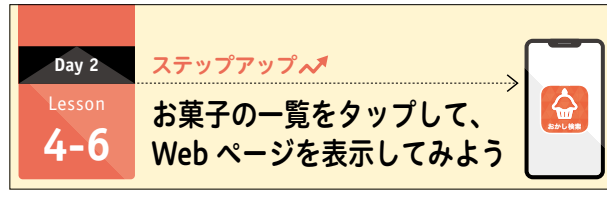
Tips（ティップス）は、Xcode の操作やコードの補足、または技術の事例を紹介します。

COLUMN

Column（技術コラム）は、本文の流れからは少しそれますが、技術の背景や、抽象的な技術の解説を記載しています。

さらに一歩踏み込んで学習できる「ステップアップ」

本書では、もう一歩進んで学習したい方のためにレッスンごとに「ステップアップ」を設けています。



サンプルアプリをさらにカスタマイズして、機能アップを行いながら学びます。「ステップアップ」が難しいと感じられた方は、最初は飛ばして学習を進めてください。2 回目以降からは「ステップアップ」も含めて学習していくことで、効果的に学ぶことができます。

本書の公式サポートサイトの紹介とサンプルアプリダウンロード

公式サポートサイトでは、本書の内容に関するサポートや、本書内で掲載されているサンプルアプリ、プログラムコードなどが提供されています。

完成したサンプルアプリの動きを確認したり、自分で打ち込んだプログラムコードの確認などでご使用ください。

本書の公式サポートサイト

https://ticklecode.com/swiftbook2020

ご利用の前に必ずお読みください

必要なパソコン機器

iOS アプリ開発には、Mac が必要です。

本当にはじめての方は、Windows パソコンでもアプリ開発ができると思いがちですが、**iOS アプリ開発には、Mac が必須**になります。Mac であれば、MacBook、iMac、Mac mini のどれであっても大丈夫です。macOS を搭載したパソコンがないと、iOS アプリ開発のための環境を作ることができません。ぜひ、自分に合った Mac の購入を検討してみてください。

本書で必要な各ソフトウェアのバージョンについて

アプリ開発を行う前に、**必要なソフトウェアのバージョンを確認**して、それを満たす必要があります。バージョンはそのソフトウェアがいつ提供されたものであるのかを示す番号です。

iOS は Apple が macOS をベースに開発した iPhone、iPad、iPod touch 向けのモバイル OS です。この iOS のバージョンも確認する必要があります。

サンプルプログラムや本書で記載されているプログラムコード、画面掲載は、以下の環境に対応しています。

- Intel Mac（執筆 2020 年 9 月時点では、Apple Silicon での動作は未確認）
- macOS Catalina 10.15.4 以降または、macOS Big Sur
- Xcode 12.0 以上
- iOS 14.0 以上

上記のバージョンに満たない場合は、バージョンアップを行う必要があります。

バージョンアップに関しては、Apple サポートページをご確認ください。

Apple サポートページ
https://support.apple.com/ja-jp

Day 1 Day 2

Lesson 1

はじめてのアプリを開発する前に知っておこう

START
Lesson 1-1
プログラミングを体験から学んでいこう

Lesson 1-2
あらかじめ挫折しそうなポイントを押さえておこう

Lesson 1-3
アプリ開発をするなら知っておこう！
～ WWDC、手数料、課金方法～

Lesson 1-4
Swift（スウィフト）を知ろう

GOAL
Lesson 1-5
SwiftUI（スウィフトユーアイ）を知ろう

iOS アプリの開発を始める前に、知っておいた方がよいことを学びます。

書籍を読み進めていく上で、最初の心構えを知っておくことで、心理的なハードルが下がり、学習が行いやすくなります。

Swift と SwiftUI についての概要を解説しているので役割を理解してください。

Day 1

Lesson 1-1

プログラミングを体験から学んでいこう

このレッスンで学ぶこと

- プログラミングに対する気持ちを整理します。プログラミングを体験して学んでいくという考え方を理解します。

1 この本の使い方と前提知識

1-1 体験から学習する

あなたの心の中に、「プログラミングは頭で学ぶもの」という気持ちがあるなら、すぐに切り替えましょう！

プログラミングは「体験しながら体で学ぶ」という方法もあります。プログラミングの文法や仕組みの理解は、あとからついてきます。あなたが体験したことが糧となって、徐々に理解できるようになります。

1-2　予習や事前学習について

本書を学習するにあたって、予習のための時間は必要ありません。

はじめてプログラミングを学習する方でも、楽しみながら、段階的に知識を習得していけるように目指して構成しています。

予習で最初に知識を詰め込んでも、その知識が必要になるとは限りません。最初にアプリを作っていき、必要なときに必要な量を学習する方が効率的です。

実際に作ってみて「動いている！」という成功イメージを持つことが大切です。

1-3　基礎知識、前提知識について

本書では、事前の基礎知識、前提知識はとくに必要ありません。「動いた！」という体験をしてから、気になること、疑問に思うことを調べて、「なるほど！」と思えたときに基礎知識が身についたといえるでしょう。

「まずは体験してみる」「基礎知識はあとから学習する」ことを念頭において読み進めていくことが大切です。

まずは、自分で書いたコードが動く楽しさを実感してください。

1-4　反復学習について

最初は、新しい知識の情報量が多いため、一度学んで理解できたと感じても定着していないことが多々あります。学校の学習や、スポーツの練習と同じで、プログラミングも反復学習、反復練習を行うことで、学びが深くなります。

本書では、一度解説した内容を、あとのレッスンでも解説しています。思い出せるように簡略して解説したり、詳細に解説しているページを明記したりと、繰り返し復習していただける構成にしています。

また、最初は本書の通りに設定をしたりコードを書いたりするのも一苦労されると思います。最後までやり遂げられたら、2 周目 3 周目と繰り返していただくことをお薦めします。最初には読み漏らしていたり理解できなかったりした解説も、新たな気づきがあり理解が進みます。

1-5　オリジナルアプリについて

本書のサンプルアプリは、シンプルで理解しやすいように設計しています。これをもとに組み合わせることで本格的なアプリを作ることができます。

本書のサンプルアプリを作れたら、機能を少しずつ変更したり、読者の皆さまが実装してみたい機能を追加したりしてみましょう。

Day 1

Lesson
1-2

あらかじめ挫折しそうなポイントを押さえておこう

このレッスンで学ぶこと

- アプリ開発で挫折しそうなポイントを事前に理解します。本書で集中して学習する、エラーや警告に対する考え方、Xcode について学びます。

1 学習ポイントを押さえよう

ポイント①：まずは一冊の本に取り組む

入門書をたくさん購入することで満足してしまいがちですが、最初は、1 つの書籍を最後まで学習することが大切です。まずは、じっくりと本書だけに取り組んでみてください。あせることはありません。すべてが理解できなくても、気にせず前に進んでください。

そして、本書を最後まで終えたら、また最初から取り組むことをお薦めします。最初はよくわからなかった箇所も、理解が進んでいることが実感できるはずです。

本書を読み終えるころには、次の新しい入門書や文法書も読み進めていくことができるようになります。

ポイント②：アプリ開発をする前の準備

アプリ開発を行う上で、事前の準備が必要になります。事前の設定ではトラブルも多く、ここで諦めてしまう方も多いと思います。

本書では、Day 1 Lesson 2 でアプリ開発の準備にも十分な時間を割いています。事前の準備が終われば、いよいよアプリを作ります！

ポイント③：アプリ開発で表示される警告やエラー

アプリの開発を進めていく過程で、**「警告」**や**「エラー」**と言われるものに遭遇することがあります。

最初は、警告やエラーの意味がよくわからず戸惑うことと思いますが、安心してください。警告やエラーはあなたを責めているのではなく、**よりよい方法を教えてくれている**のです。

警告やエラーに遭遇したときには、「アドバイスしてくれてありがとう！」というぐらいの気持ちを持って、開発に取り組みましょう。エラーメッセージを読んでわからない場合でも、メッセージをそのままインターネットで検索すると対処法がわかる場合も多くあります。調べながら解決をすることを覚えてください。

ポイント④：まずは、Xcode を体験して慣れていこう

アプリ開発には、Xcode という統合開発環境を利用します。

最初は、Xcode の操作がよくわからなくてなかなか思うように作業が進みません。でも、Xcode でのメニュー配置が理解できて、目的の作業をするための手順がわかるようになると、効率的に制作できるようになります。

Xcode も理解しようとするよりも体験をして、慣れていくことが大切です。繰り返し操作を体験し、どんどん慣れてください。

Day 1

Lesson 1-3

アプリ開発をするなら知っておこう！

～WWDC、手数料、課金方法～

このレッスンで学ぶこと

- **アプリ開発をする上で知っておきたい、WWDC の概要、ダウンロード数などの数値、手数料、課金について概要をつかみます。**

1 WWDC での公開情報をもとに考察してみよう

1-1 WWDC とは

WWDC（Worldwide Developers Conference）は、iPhone の開発元である Apple が毎年開催している、開発者向けのイベントです。

WWDC で Apple の新製品や新機能が発表されたり、市場動向や開発者への支払額が公開されたりします。そのため、アプリ開発者にとっては、とても関心が高いイベントです。

WWDC - Apple Developer
https://developer.apple.com/wwdc20/

2020 年に開催された WWDC20 は、6 月 22 日～26 日に開催されました。世界的なウイルス感染拡大を考慮して、初めての完全オンラインで開催されました。参加料は無料で 2,200 万人が参加しました。

WWDC は、Apple の製品・サービスが発表されるイベントですが、WWDC 2019 までは、参加するためのチケットは 1,599 ドル（WWDC 2019 チケット価格）で販売されていて、1 ドル 108 円換算で約 17 万 2 千円の参加料でした。その人気の高さがわかります。

▼ WWDC での製品発表

発表年	製品
2010 年	iPhone 4、iPad
2012 年	MacBook Pro（Retina ディスプレイ搭載）
2013 年	Mac Pro（円筒形デザイン）
2014 年	Swift
2015 年	Swift 2.0、Swift オープンソース化
2016 年	Swift 3.0、Sirikit、iMessage Apps
2017 年	Swift 4.0、iPad Pro、HomePod
2018 年	ARKit 2.0、CreateML
2019 年	Sign in with Apple、SwiftUI、Combine
2020 年	Apple Silicon への移行、Widget、App Clips

WWDC では、表のような製品・サービスが発表されています。

1-2　Apple がアプリ開発者に支払った金額

アプリ開発者は平均でどのぐらいの収入を得ているのでしょうか？　WWDC 2019 以降は開発者への支払い総額や登録アプリ数の公式発表は行われていません。公式発表がされた、WWDC 2018 での数値から試算してみたいと思います。

WWDC 2018 での公式発表によると、2008～2018 年までの 10 年間で Apple が開発者に支払った金額は累計 1000 億ドルを超えています。支払った金額を 1 ドル 110 円換算すると、約 11 兆円にもなります。また、登録アプリ数は約 220 万本（2017 年 1 月現在）であるとされるため、この 11 兆円を 220 万本のアプリ数で割ると、1 つのアプリの平均収入は約 500 万円になります。

2008 年からアプリを 1 つリリースしていたとすると、年間平均で 50 万円、月平均で約 4 万円の収入になります。そして、登録アプリ数の 220 万本には無料アプリも含まれていますので、平均は約 4 万円以上になると思われます。

1-3　Apple の手数料

Apple Store で販売した売上のうちの、30% を Apple が手数料として差し引きます。これが、Apple の収益になります。残りの 70% が開発者への支払額です。支払いは、指定した銀行口座に振り込まれます。

たとえば、120 円のアプリであれば、84 円が振り込まれることになります。

1-4　ユーザーへの課金方法

iPhone アプリを、App Store に公開してユーザーに課金する方法は 3 種類あります。

▼ ユーザーへの課金方法

有料ダウンロード	アプリをダウンロードするときに課金する方法。ユーザーは 1 度ダウンロードすれば、その後のアップデートなどは無料で実行可能。アプリを削除して再びダウンロードするときも無料
広告	アプリの中に広告を表示させて収益を得る方法。いろいろな会社がアプリ向けの広告配信サービスを提供している
In App Purchase（アプリ内課金）	アプリ内で課金する方法。有料もしくは無料でアプリを提供し、そのアプリ内でアイテムや機能追加を販売し、収益を得る。アプリ内課金には、次の 4 つの方法がある ①消耗型：アプリの実行に伴い消費されていく。消費アイテムなど ②非消耗型：1 度購入するとユーザーのすべてのデバイスで使用可。書籍など ③自動更新購読：期間を決めて販売、自動で更新。新聞、雑誌など ④非更新購読：期間を決めて販売、自動で更新されない。1ヶ月購読など

Day 1

Lesson 1-4

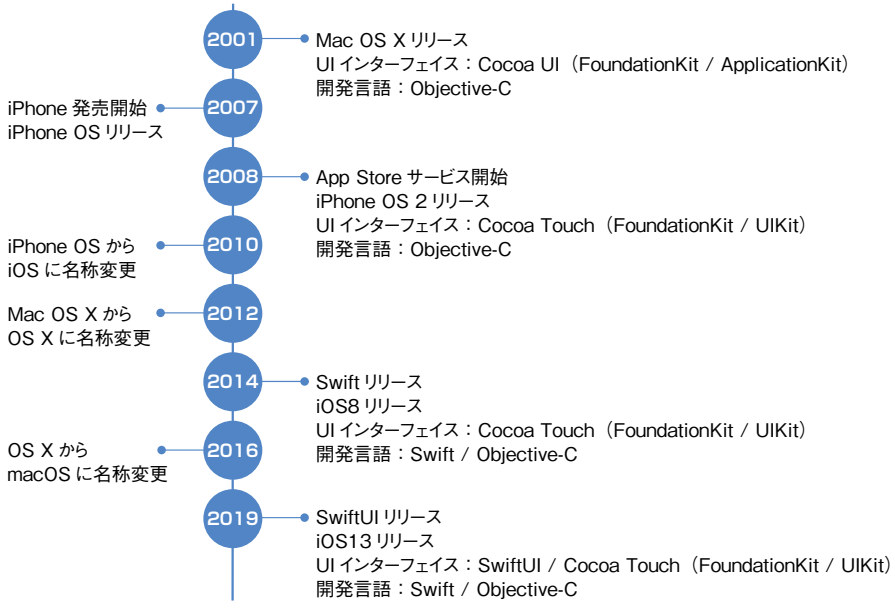
Swift（スウィフト）を知ろう

このレッスンで学ぶこと

- **iOS 開発の歴史を学びます。**
- **iOS アプリ開発で使う、Swift（スウィフト）というプログラミング言語について概要を学びましょう。**

1 iOS 開発の歴史を振り返ろう

みなさんが、本書で iOS アプリを作るために利用するプログラミング言語が、Swift（スウィフト）です。
さらに、画面を作る仕組みの SwiftUI（スウィフトユーアイ）も併せて学びます。
Swift は 2014 年、SwiftUI は 2019 の WWDC ではじめて発表されました。
Swift と、SwiftUI の概要をお伝えする前に、先に iOS 開発の歴史を簡単に振り返ってみましょう。



2001 年に、Mac の OS（オペレーティングシステム）として、Mac OS X（マック オーエス テン）がリリースされました。

アプリ開発の基礎として、画面を作るための UI インターフェイス開発キットの、Cocoa UI フレームワーク（FoundationKit と ApplicationKit）と Objective-C というプログラミング言語が使用されていました。

UI は、User Interface（ユーザーインターフェイス）の略称で、ユーザ（利用者）とアプリの接点を意味し、アプリの画面やレイアウト、使い勝手のことを指します。

2007 年には、iPhone が発売開始されました。翌年の 2008 年には、App Store サービスが開始され、Apple 以外に開発されたアプリのダウンロードが可能になりました。

同時に、Cocoa Touch（FoundationKit と UIKit）と Objective-C を用いたモバイルプラットフォームが iPhone 用に開放されました。後に、iPad も対応されました。

2014 年には、Objective-C の後継となる開発言語、Swift がリリースされました。Swift は、従来の iOS 開発ツール、Xcode、Objective-C、および Cocoa フレームワークのすべてと互換性があるように設計されています。

そして、2019 年に、新しい UI インターフェイス開発キットの SwiftUI がリリースされました。Swift や SwiftUI は、従来の開発で課題になっていた開発の難しさ、学習のしづらさを改善し提供してくれています。

この流れからもわかるように、常に開発環境や言語は改善され進化を遂げていきます。開発者はより良い開発を行うためには、常に新しい技術をキャッチアップして学習する必要があるということが分かります。

これは、iOS 開発以外のソフトウェア開発のどの分野でも同じで、常に進化した技術が公開され開発の仕方も変化します。変化を楽しみながら学習を続けましょう。

Tips

Swift が発表される前は、「Objective-C」（オブジェクティブシー）というプログラミング言語を使って、アプリ開発が行われていました。

Objective-C は 1980 年代から開発がはじまり、機能拡張を経て現在も使用されています。Objective-C は、記述が長くなり複雑であったり、プログラムを書いていく効率があまりよくなかったりと、初心者には難しい言語です。

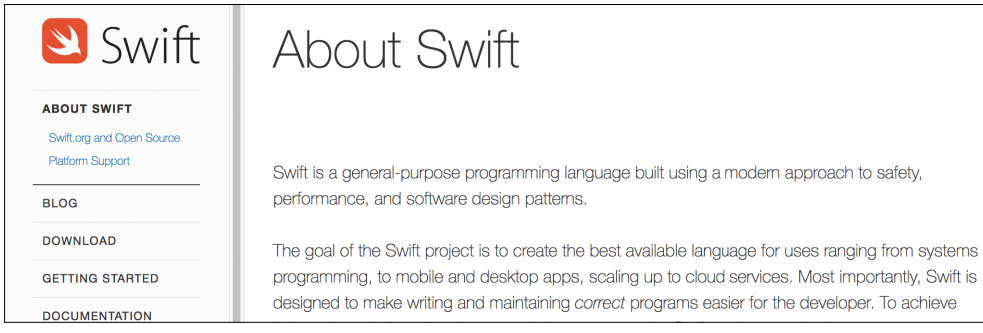
Swift では、学びやすいような工夫がされていて、新しい考え方や機能を積極的に取り入れています。

Swift もバージョンアップを重ねて、十分に開発が行える言語として成長しています。また、過去の資産を有効活用できるように、Objective-C で作成されたプログラムを Swift から呼び出すこともできます。本書では、これからの開発言語である「Swift」でプログラムを記述します。

2 Swift の特徴を押さえよう

Swift は、現代的なプログラミング機能を備え、開発者にとっても学習がしやすい言語です。

▼ Swift.org



Swift.org - Welcome to Swift.org
https://swift.org/

Apple の公式サイトでは、Swift とは「誰もが圧倒的に優れたアプリケーションを作れる、パワフルなオープンソースの言語」として紹介されています。

Swift を使うことでアプリ開発者はより安全で、より信頼性の高いコードを書くことができ、時間を節約しながら、より豊かなアプリを作ることができます。

特徴①：Swift は高速である

Swift は日本語訳で「迅速」という意味で、鳥の「あまつばめ」を示す意味でも使われます。Swift の迅速さを強調するために、ロゴマークにも「あまつばめ」が採用されています。

Swift は他のプログラミング言語と比較して、検索アルゴリズム（データを探し出す方法）が高速だと言われています。

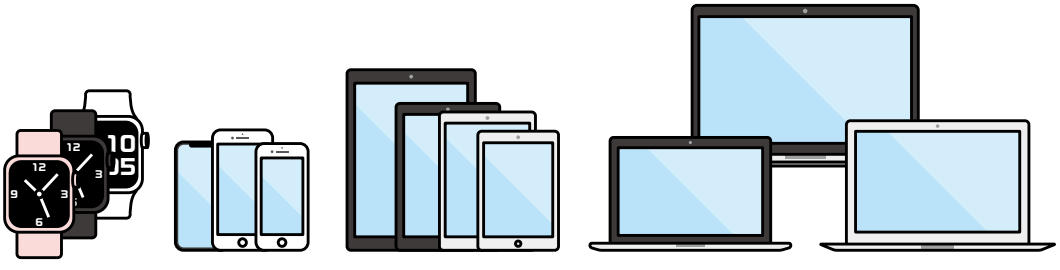
特徴②：Swift はモダンである

モダン（modern）とは「現代風」という意味です。Swift では、他のプログラミングでも採用されている新しい機能を、積極的に取り入れています。

特徴③：Swift は安全である

Swift はプログラミングの記述ミスやバグ（不具合）が起こりにくい仕組みを採用しています。

特徴④：たくさんの Apple 製品で動くアプリが作れる



Swift で作れるアプリは、iPhone や iPad 上で動くアプリだけではありません。Mac、Apple Watch などたくさんの Apple 製品で動くアプリを作ることができます。

特徴⑤：Swift はオープンソースである

オープンソース（Open Source）とは、プログラムソースを一般に公開して、誰もが使ってよいとする考え方です。

オープンソースであれば、一般人も Swift の改良に参加することができますので、ネット上では日々意見交換され、さらなる改良・進化していくことが期待されています。

Day 1

Lesson 1-5

SwiftUI（スウィフトユーアイ）を知ろう

このレッスンで学ぶこと

- 新しい開発方法の、SwiftUI（スウィフトユーアイ）という仕組みについて概要を学びます。

1 SwiftUI の特徴を押さえよう

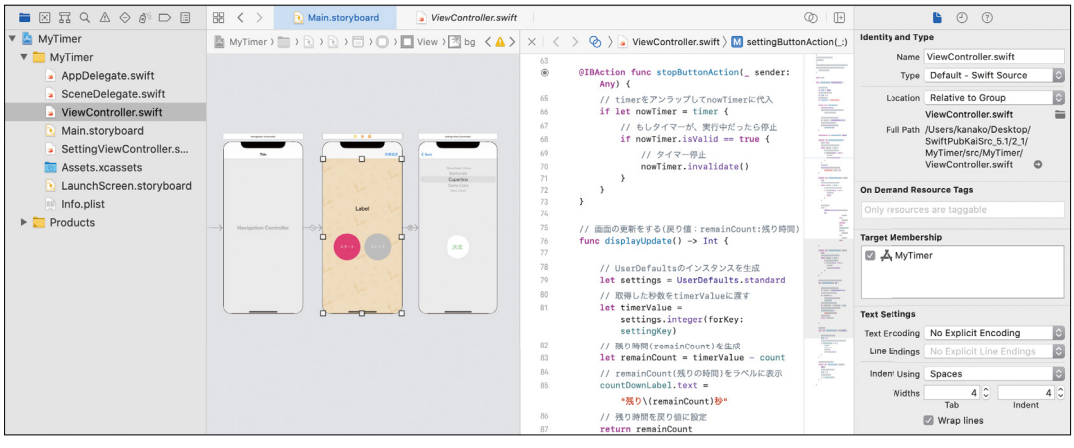
WWDC 2019 にて、新しい開発プラットフォーム SwiftUI が発表されました。

SwiftUI は、iOS や他の Apple プラットフォーム上で、宣言型な構文を用いて、プログラムコードとデザインを完全に同期しながら、アプリをデザインできるユーザーインターフェイスのツールキットです。

SwiftUI

SwiftUI での画面作成方法と、SwiftUI の前に行われていたストーリーボード（Storyboard）の画面作成方法を見比べてみましょう。

▼ ストーリーボードの画面



ストーリーボードでは、GUI（Graphical User Interface）で画面を作ります。

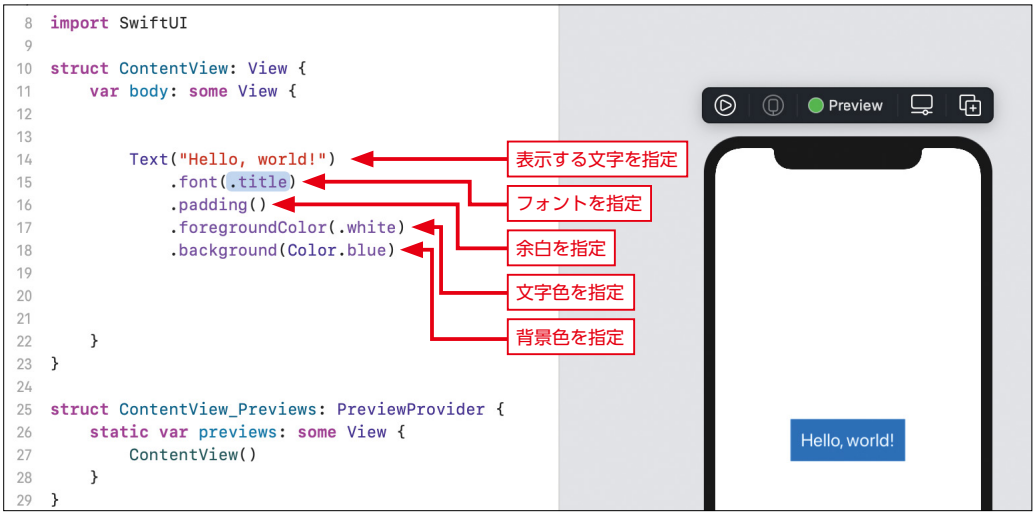
GUI（ジーユーアイ）とは、アイコン、メニュー、スクロールバー等をマウスや指の操作で視覚的に操作をしてコンピューターに命令をする仕組みのことを言います。

画面の移動の設定や、画面の部品をドラッグ & ドロップで配置をして、色や文字の大きさ等の設定を画面で行っていきます。そして画面の部品と、プログラムコードを関連付ける設定を行いアプリを作っていきます。

もちろん、画面のデザインに関することもコードで書けますが、開発者によって作り方がまちまちで、大規模な開発ではルールの統一が必要で管理が難しく工夫が必要でした。

▼ SwiftUI の画面

SwiftUI は、宣言型な構文を用いて、プログラミングコードとデザインを同期しながら、アプリのデザインができます。同期しながらアプリのデザインが出来るとはどういうことでしょう？



例えば、文字を表示できる Text という部品に対して、デザインを適用する場合は、このように指定をします。

Text に表示する文字を指定してから、フォント、余白、文字色、背景色を段階的に適用しています。これらは、GUI からも設定が行なえますが、設定と同時に自動的にプログラムコードに同期され反映がされます。

このように、何をしたいのかを指定することを宣言的プログラミング（Declarative programming）と言います。対義語としては、どのようにするのかの処理を記述する命令型プログラミング（Imperative programming）があり、iOS 開発の中では、SwiftUI より前の開発方法にあたります。

本書では、Swift と SwiftUI を用いたアプリ開発を解説します。様々なサンプルアプリを通して、実践的に学びつつ SwifUI の開発方法を学んでいきましょう。

SwiftUI については、次の Apple 公式ページをご参照ください。

SwiftUI - Apple Developer
https://developer.apple.com/jp/xcode/swiftui/

Day 1 Day 2

Lesson 2

アプリ開発の環境を整えて、Xcode の使い方を学ぼう

START
- Lesson 2-1 開発をするために必要な準備をしよう
- Lesson 2-2 Apple ID を取得しよう
- Lesson 2-3 Xcode をインストールしよう
- Lesson 2-4 Xcode を起動して、プロジェクトを作成しよう
- Lesson 2-5 Xcode をより使いやすくするための設定をしよう
- Lesson 2-6 ボタンをタップして「Hello, World!」から「Hi, Swift!」に切り替えてみよう

GOAL
- Lesson 2-7 アプリの動きを確認する方法を学ぼう

iOS アプリを開発するために、必要なものを学びましょう。

最初に、まだ Apple ID を取得していない方のために、Apple ID の基礎知識と取得方法についても解説します。

Day 1

Lesson 2-1

開発をするために必要な準備をしよう

このレッスンで学ぶこと

- アプリ開発をするために用意しなければならないことを学びます。
- Mac や Apple ID、Xcode の概要について理解します。

1 アプリ開発に必要な 3 つのものを準備しよう

iOS アプリを開発するためには、次の 3 つが必要になります。

- **Mac**
- **Apple ID アカウント**
- **Xcode**

上記の 3 つに加え、開発したアプリを全世界の人たちに利用してもらうには、別途、「Apple Developer Program」への登録も必要です。

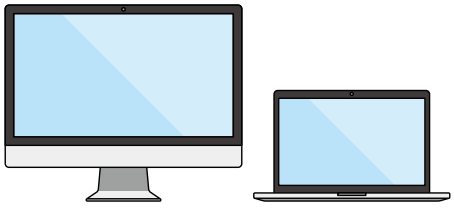
このレッスンでは開発したアプリを、iPhone/iPad に転送（実機転送）して利用するところまでをゴールにしています。

では、必要な 3 つのものを確認しましょう。

1-1 Mac

2015 年 12 月に、Swift は誰でも利用・改変できるオープンソースとしてソースコードが公開されました。そのため、将来 Mac 以外のパソコンでも iOS アプリ開発ができるようになる可能性もありますが、現時点では、Mac を準備してください。

▼ Mac



1-2 Apple ID

iOS アプリを開発するためには、Apple ID の作成が必要です。Apple ID は、Apple が提供しているオンラインサービスを利用するために必要なアカウントです。オンラインストアの「iTunes Store」では、音楽・映画やオーディオブックを購入できます。また、「App Store」というサービスでは、iPhone や iPad、Mac で利用できるアプリがダウンロードできます。さらに、有料で販売されているアプリも購入することができます。iPhone や iPad をお持ちの方は、すでに Apple ID を利用していると思います。それぞれのサービスを利用するのに必要な Apple ID ですが、Xcode のダウンロードでも必要になるため、事前に作成する必要があります。

▼ App Store と iTunes

1-3 Xcode（エックスコード）

Apple ID が作成できたら、「App Store」で、iOS アプリ開発に必要な Xcode をインストールします。Xcode は、Mac、iPhone、iPad、Apple Watch、Apple TV 向けのアプリを開発できる環境を提供してくれます。Xcode を利用して、画面やコードの作成、デバッグ、App Store へのアプリの提出ができます。一般にこのようなツールは、**統合開発環境、もしくは IDE（Integrated Development Environment）**と呼ばれています。IDE は、ソフトウェアを効率よく開発できるように、さまざまな機能を提供してくれている開発ツールです。Swift や Objective-C を用いて開発ができる Xcode 以外にも、さまざまな言語（Java、Ruby、PHP など）で、有料無料問わずにたくさんの IDE があります。Xcode 以外にも、iOS アプリ開発ができる IDE はありますが、一般的に多く利用されているのは Xcode です。

▼ Xcode

Day 1

Lesson 2-2

Apple ID を取得しよう

このレッスンで学ぶこと

- Apple ID の作成手順を学び、Apple ID を取得します。
- Apple ID の調べ方や、メールアドレスの変更方法も確認します。

1 Apple ID をすでに取得されている方

すでに Apple ID を取得済みの方は、その Apple ID が 1 つあれば、Apple のすべてのサービスが利用できます。

App Store から Xcode をダウンロードできますので、このレッスンは読み飛ばしていただいて大丈夫です。

1-1 Apple ID を持っているか不明な方

取得しているご自身の Apple ID がよくわからない場合があります。
その際は、下記の URL から調べることができます。

Apple ID を忘れた場合 - Apple サポート
https://support.apple.com/ja-jp/HT201354

1-2 Apple ID のメールアドレスを変更したい方

すでにお持ちの Apple ID のメールアドレスを変更できます。
メールアドレスを変更しても、いままで利用してきた Apple ID をそのまま使い続けることができます。
下記の URL に、メールアドレスの変更方法が記載されています。

Apple ID を別のメールアドレスに変更 - Apple サポート
https://support.apple.com/ja-jp/HT202667

2 Apple ID をまだ取得されていない方

次の手順に沿って、一緒に作成しましょう。

2-1 Apple ID のサイトにアクセス

Apple ID のサイトにアクセスします。
［Apple ID を作成］をクリックします。

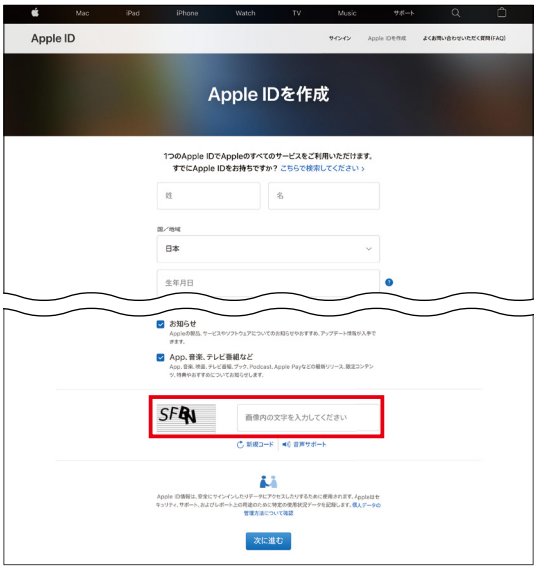
Apple ID を管理 -Apple
https://appleid.apple.com/jp

▼ Apple ID ログイン画面

2-2 Apple ID を作成

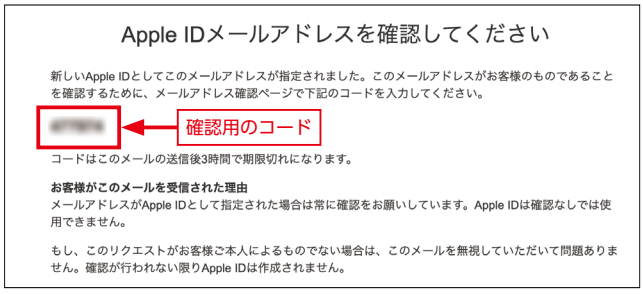
各項目を入力しましょう。個人の情報を記入します。赤枠の入力は、文字認証と呼ばれている仕組みです。左に表示されている画像の英数字を読み取って、右の入力欄に入力します。読みづらい英数字の場合は、「新規コード」をクリックして、読める英数字を表示させましょう。これは、悪意のあるプログラムから自動的にアクセスされ不正に利用されないようにするためです。人間にしか読めない読みにくい画像で英数字を読んで入力させることで、悪意のあるプログラムからは利用できないようにしています。よく利用される認証方法ですが、Apple の場合は音声で読みあげるサポートもしてくれています。「音声サポート」をクリックすると、表示されている英数字を読みあげてくれます［次に進む］をクリックすると、確認用のコードを入力する画面が表示されます。

▼ Apple ID の作成



入力したメールアドレスに確認用のコードが送信されます。メールを確認すると、確認用のコードが記載されています。

▼ メールで送信された確認コード



確認用のコードが、登録した E メールアドレスと電話番号にそれぞれ送信されます。確認をして確認コードを入力してください。

▼ 確認コードの入力

これで、Apple ID の作成は完了です。Xcode のダウンロードが行える準備が整いました。
続いて、Xcode のダウンロードとインストールを行います。

Point

Apple ID は、App Store、iTunes Store、iCloud をはじめとした Apple サービスへのアクセスに使う個人アカウントです。
Apple すべてのサービスで使う連絡先、支払情報などが含まれています。
アプリ開発においても、Xcode から iPhone へアプリを転送する際に Apple ID が必要になりますし、アプリを App Store へ申請登録する際にも必要となります。
とても重要な ID とパスワードですので、大切に管理しておきましょう。

Xcode をインストールしよう

このレッスンで学ぶこと

- App Store から、Xcode のインストール方法を学びます。
- Xcode でアプリ開発が行える環境を整えます。

1 Xcode をダウンロードしよう

1-1 Xcode をダウンロードする前に

本書では、Xcode を利用して開発を進めますので、最初に App Store から Xcode のダウンロードを行います。

インターネット回線の速さによって変わりますが、Xcode のダウンロードには数十分かかります。

MacBook などの場合は、ダウンロードとインストール実行中にバッテリーが切れないように気をつけてください。

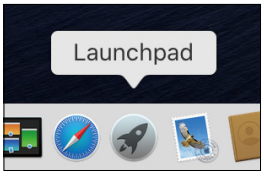
また、Xcode 自体のファイルサイズが数 GB あり大きいので、インストールする Mac のストレージ容量も充分かどうか確認してください。

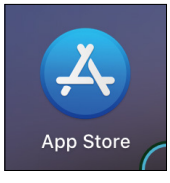
1-2 App Store からのインストール

App Store から Xcode をインストールしましょう。

Mac の Dock にある「Launchpad」（ランチパッド）アイコンをクリックして起動します。Launchpad の中にある App Store を起動します。

▼ Launchpad（ランチパッド）から App Store を起動





App Store が起動するので、画面の左上の検索ボックスに「Xcode」と入力して、「enter」キーを押します。

▼ App Store から Xcode を検索

Xcode のダウンロードとインストールがはじまります。Xcode のファイルサイズはとても大きいため、ダウンロードには時間がかかることがあります。本書を読み進めながら、気長にお待ちください。

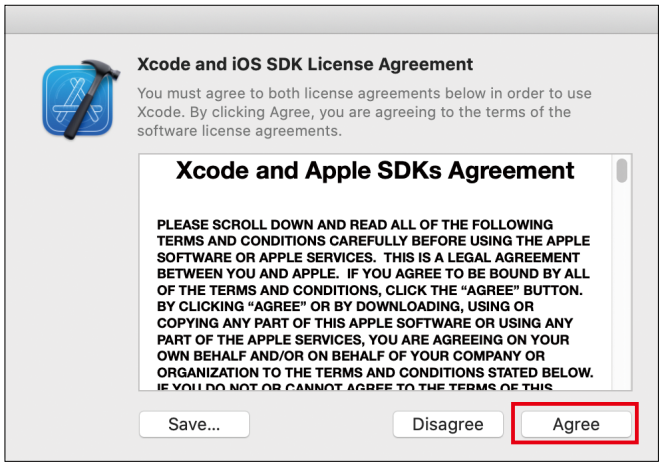
▼ 赤枠をクリックしてダウンロード

ダウンロードが完了すると「開く」アイコンが表示されるのでクリックします。

▼ 開くをクリック

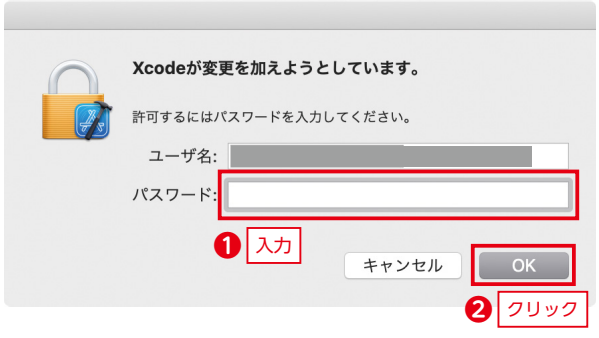
Xcode の利用規約が表示されます。利用規約の内容を確認して、「Agree」（同意）をクリックします。

▼「Agree」をクリック



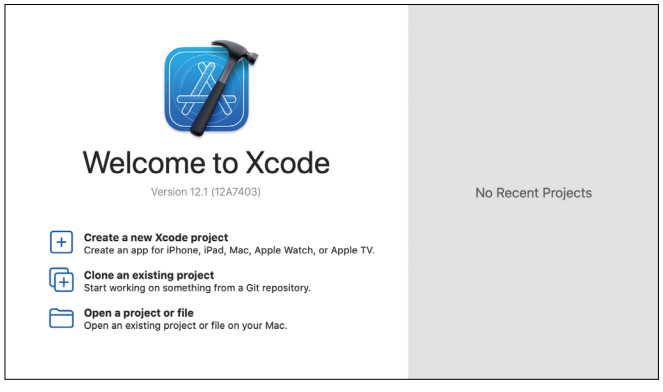
❶ Mac のアカウントのパスワードを入力して、❷「OK」をクリックします。

▼ パスワードを入力



Xcode が起動します。アプリ開発の準備は完了しました。おめでとうございます！

▼ Xcode の起動



Day 1

Lesson
2-4

Xcode を起動して、プロジェクトを作成しよう

このレッスンで学ぶこと

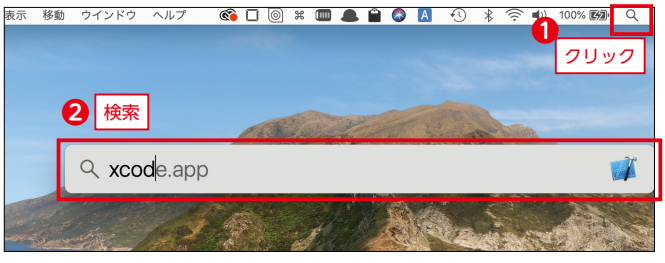
- Xcode を起動して、プロジェクトの作成を行います。
- Xcode の基本的な画面構成について学習します。

1 Xcode を起動しよう

1-1 Xcode を起動

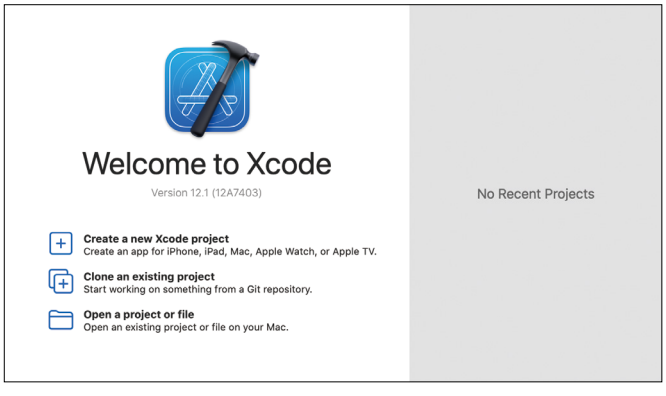
Xcode の起動方法はいくつかあります。Launchpad から、Xcode のアイコンをクリックして起動しても良いですし、❶画面右上の Spotlight（スポットライト）をクリックして、Xcode を検索して起動もできます。

▼ Spotlight を起動して Xcode を検索



「Welcome to Xcode」と書かれた画面が表示されます。画面には、「Create a new Xcode project」、「Clone an existing project」、「Open a project or file」の 3 つのメニューが表示されています。次のページより、メニューを 1 つずつ確認しましょう。

▼ Xcode 起動画面



- **Create a new Xcode project**
  新規のプロジェクトを作成します。本書では、すべてこのメニューから新規にプロジェクトを作成して、アプリを開発します。
- **Clone an existing project**
  バージョン管理システムを使ってプロジェクトを作成する方法です。
  バージョン管理とは、ファイルの変更履歴を管理してくれるシステムです。本書では解説の対象外になります。
- **Open a project or file**
  Mac で既存のプロジェクトやファイルを開きます。

2 プロジェクトを作成しよう

2-1 プロジェクトを作成

赤枠の［Create a new Xcode project］をクリックして、はじめてのプロジェクトを作成してみましょう。

プロジェクトはアプリ開発に必要なプログラムコードや様々なデータを束ねるものです。

1 つのプロジェクトで 1 つのアプリを開発します。

プロジェクトを作成すると、雛形（ひながた）のアプリ画面や開発に必要なプログラムコードが作成されます。

新規のプロジェクトを作成したいので、赤枠の［Create a new Xcode project］をクリックします。

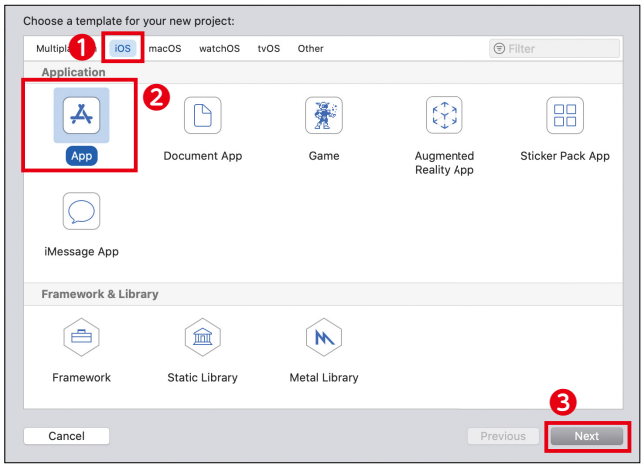
メニューの下に表示されている「Show this window when Xcode launches」のチェックボックスは、Xcode 起動時にこのメニューを表示するか否かの設定項目です。

▼ Xcode プロジェクトの作成

［Choose a template for your new project:］ダイアログが表示されます。これは、プロジェクトのテンプレート（雛形）を選ぶ画面です。この画面で、プロジェクトのテンプレートを選ぶことができます。次の手順に沿って選びましょう。

❶ 画面の左上部の［iOS］を選択します。
❷［App］を選択します。
❸［Next］を選択しましょう。

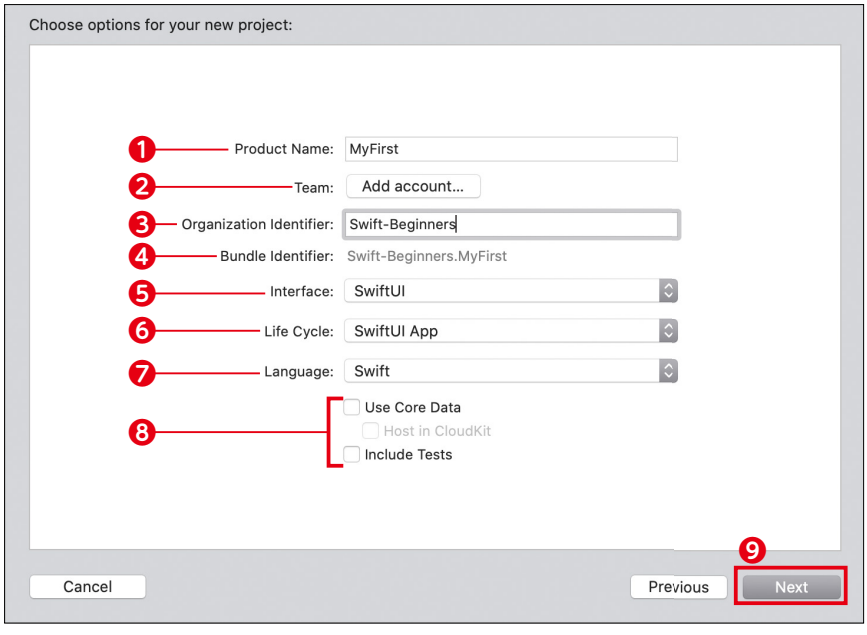
▼ プロジェクトテンプレートの選択



2-2　プロジェクトの情報を入力

［Choose options for your new project］と記載された画面が表示されます。
この画面では、プロジェクトの情報を設定します。設定する情報を一つ一つ確認しましょう。

▼ プロジェクトの新規作成画面



❶ Product Name（製品名）

製品名とは、プロジェクトの名前です。製品名は、好きなものが入力できます。ここでは、「MyFirst」と入力します。

製品名は 2 文字以上で 255 バイト以下である必要があります。

❷ Team（チーム）

Xcode に登録済みの「Apple ID」を選択します。シミュレータで動作確認をする場合には設定の必要はないですが、実機転送（じっきてんそう）を行う際には設定が必要です。

本レッスンの最後で実機転送を説明していますので、いまは設定をしなくても大丈夫です。

❸ Organization Identifier（組織識別名）

［Organization Identifier］は、❶［Product Name］と組み合わされて、❹［Bundle Identifier］に設定されます。

［Bundle Identifier］は世界中のアプリの中でユニーク（一意）にする必要があり、重複が許されません。

［Organization Identifier］は、アプリが所属するグループ（組織や個人）を識別します。

識別として利用されるのは、ドメイン表記を逆にした記述（逆ドメイン）です。

ドメインが「swift-beginners.jp」の場合は、「jp.swift-beginners」と入力します。

ドメインを持っていない方は、メールアドレスを逆から入力する方法もあります。

なぜなら、メールアドレスも世界に一つだけのものなので、ユニークになるからです。

ここでは、個人のメールアドレスを入力してみましょう。

そのときに、メールアドレスは逆から入力し、@（アットマーク）は .（ピリオド）に置き換えます。

例えば、「swift-beginners@example.jp」の場合は、「jp.example.swift-beginners」と入力します。

Point

本書では、［Organization Identifier］を、「Swift-Beginners」と入力しています。
ですが、学習の際には、別の ID にしてください。たとえば、ご自身のメールアドレスのように、重複しにくい ID を入力してください。

❹ Bundle Identifier

この項目は、入力することはできません。［Product Name］と［Organization Identifier］を組み合わせて自動で生成されます。

今回の設定では、「Swift-Beginners.MyFirst」と表記されています。

❺ Interface（インターフェース）

［SwiftUI］（スウィフトユーアイ）と［Storyboard］（ストーリーボード）が選べます。

［SwiftUI］は、Xcode11 から提供された、アプリの UI（ユーアイ）を直感的かつ、少ないコードで開発できる新しいフレームワークです。

UI は、User Interface の略称で、ユーザ（利用者）とアプリの接点を意味し、アプリの画面やレイアウト、使い勝手のことを指します。

今回は、［SwiftUI］を選択してアプリ開発に挑戦しましょう。

❻ Life Cycle（ライフサイクル）

［SwiftUI App］（スウィフトユーアイ アップ）と［UIKit App Delegate］（ユーアイキット アップ デリゲート）を選べます。ライフサイクルとは、iOS アプリの起動、終了、プッシュ通知等の管理方法です。［UIKit App Delegate］を選択すると SwiftUI 以前のアプリ構成でテンプレートが出力されます。SwiftUI と Storyboard の開発を併用する場合に利用します。［SwiftUI App］が SwiftUI に純粋に準拠したライフサイクルです。本書では、［SwiftUI App］を利用していきます。

❼ Language（プログラミング言語）

アプリを作るプログラミング言語を選択します。

［Swift］（スウィフト）と［Objective-C］（オブジェクティブシー）が選べます。「Swift」を選択します。

❽ Use Core Data、Host in CloudKit、Include Tests

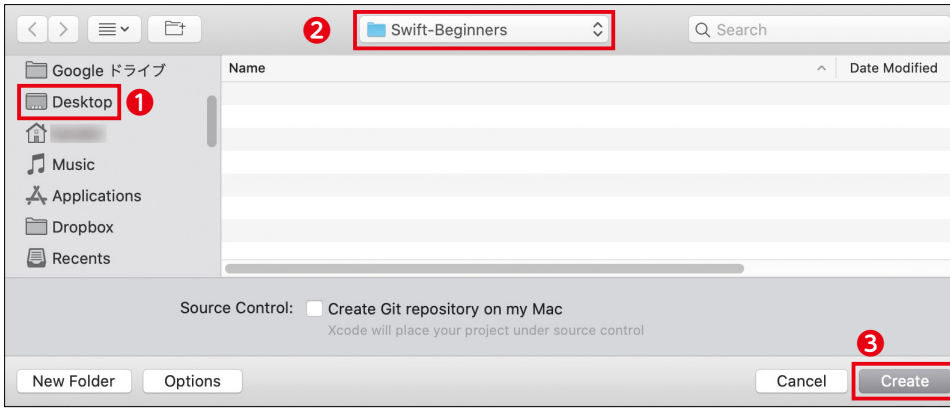
これらの項目にチェックをすると、高度な開発で必要になるデータベースやテストコードの雛形が生成されます。

本書では解説の対象外のため、すべてチェックを外します。

2-3　プロジェクトを保存

プロジェクトの保存場所を聞かれるので、適当な場所を選択します。本書では、❶［Desktop］（デスクトップ）に❷「Swift-Beginners」というフォルダを作成して、そこにプロジェクトを保存します。❸［Create］（クリエイト）ボタンをクリックします。Create Git repository on my Mac のチェックは、ソースの履歴を管理できる、バージョン管理システムを利用するか否かの設定です。本書では対象外にしているので割愛をさせていただきます。

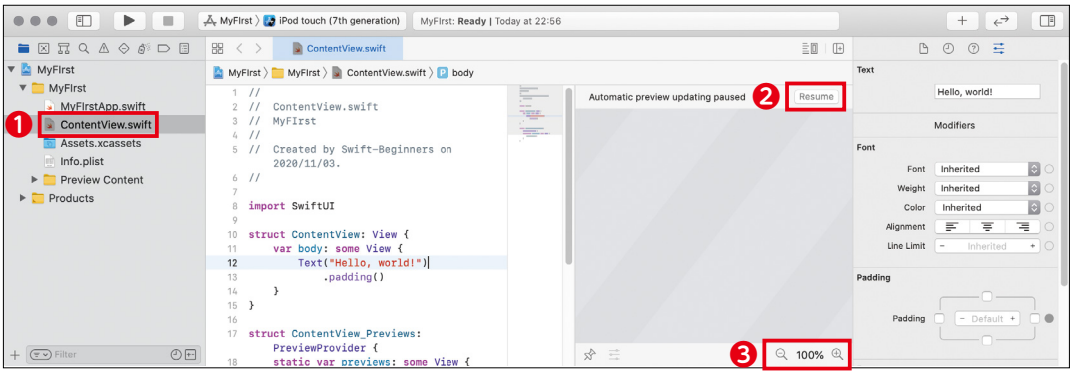
▼ プロジェクトの保存場所



3 Xcode 画面構成を確認しよう

プロジェクトが作成できたので、Xcode の画面の名称を確認していきましょう。

▼ Xcode プロジェクト画面



プロジェクトが作成されると、❶ ContentView.swift が画面に表示された状態になります。 ContentView.swift 以外にもファイルやディレクトリが作成されていることが確認できます。このように、プロジェクトを作成すると、アプリ画面のテンプレート（雛形）が作成されます。

ContentView.swift で、画面に表示する View（ビュー）やレイアウトを実装（じっそう）していきます。

View とは、アプリの UI の基本的な構成要素である、Button（ボタン）や Text（テキスト）のことです。

画面に配置（はいち）される、部品やレイアウトのことを、**View** と呼びます。

❷［Resume］（レジューム）をクリックすると、ContentView.swift のプレビューが表示されます。

クリックしてみましょう。

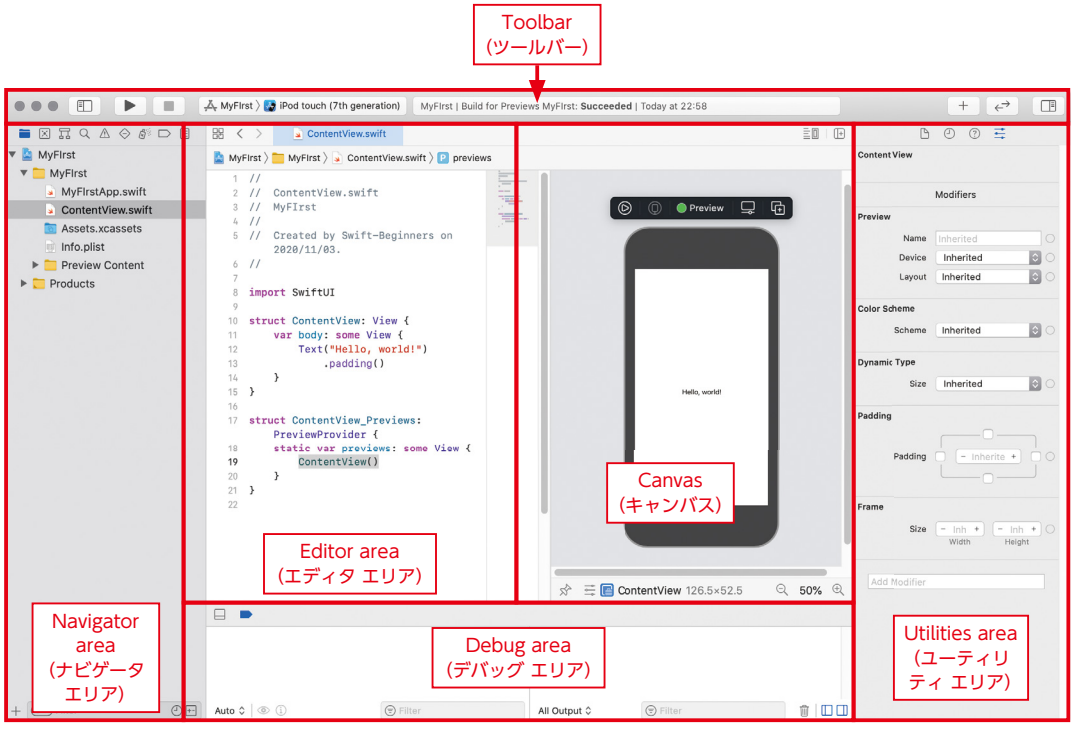
画面が表示されますが、画面がはみ出て表示されている場合は、❸「＋」「-」アイコンで表示比率を調整できます。

3-1 エリアの名称を確認しよう

実際の利用方法は、サンプリアプリの開発を通して繰り返し操作して慣れていきますので、今は、こんな画面の区切りがあるという認識をしていただければ大丈夫です。

Debug area だけは、メニューから表示させる必要があります。メニューの［View］>［Debug Area］>［Show Debug Area］をクリックすることで表示することができます。

Xcode 画面構成



Tips

Xcode は英語表記のため、最初は戸惑うと思います。本書でも、Xcode の画面項目を示すときには英語表記のまま説明しています。これにはいくつか理由があります。

まず、Xcode はバージョンアップが早いため、英語表記でそのまま使えるようになったほうが早く対応できます。そして、アプリを作って世界へ公開したい（リリース）となったときに、審査の手続きも英語で行います。その審査がリジェクト（申請却下）となったときの理由や対応方法も英語のメールで届くため、普段から英語での操作環境に慣れていると、対処の仕方も見えてきます。

また、公式ドキュメントも英語の方が圧倒的に多くあります。近年は翻訳ツールの性能も向上しているので、翻訳ツールを活用するのも良い方法です。

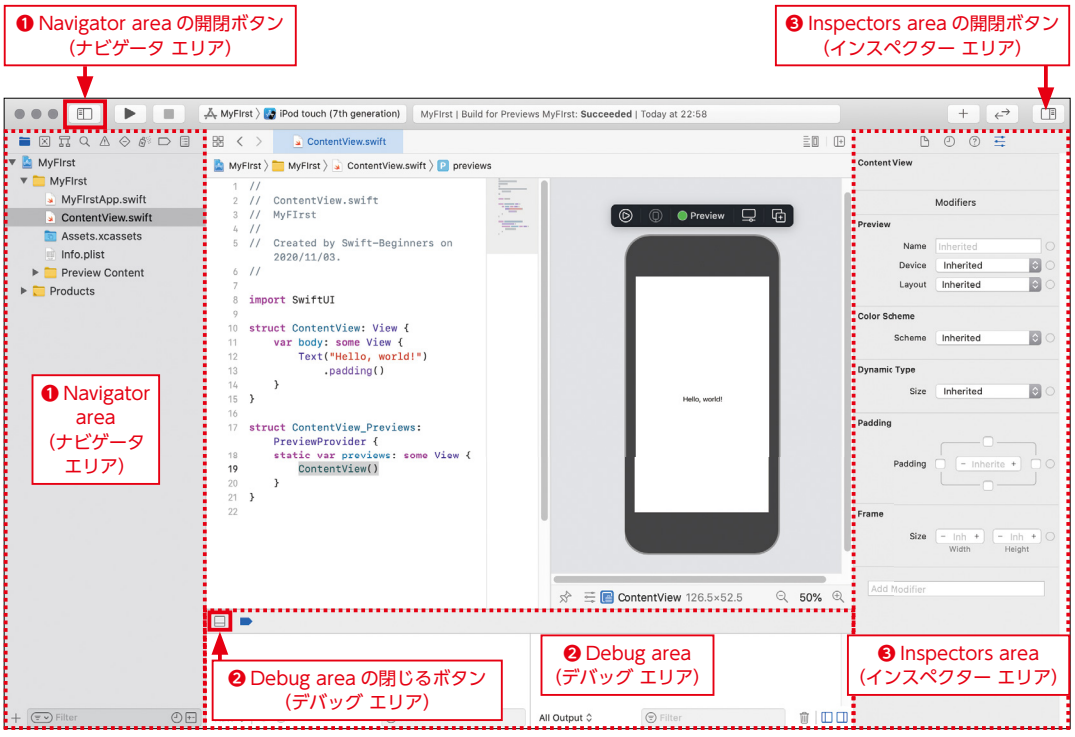
上記のような理由から、Xcode はそのまま英語表記で、最初はなるべくフリガナを記載して解説を進めていきます。

3-2　エリアの開閉方法

［Editor area］は表示されたままですが、❶［Navigator area］❷［Debug area］❸［Inspectors area］の 3 つのエリアは、❶［Navigator area］の開閉ボタン ❸［Inspectors area］の開閉ボタンのアイコンをクリックすることで開閉することができます。［Debug area］は、❷［Debug area］の閉じるボタンで閉じることができます。表示させる場合は、メニューの［View］>［Debug Area］>［Show Debug Area］をクリックすることで表示することができます。

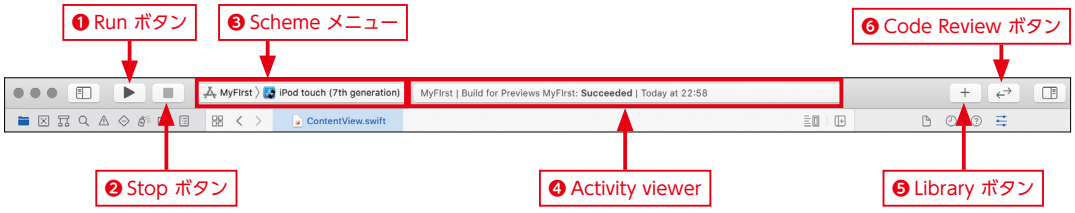
開発中は、限られた画面スペースで作業をしていくため、状況に応じて各エリアを開閉しましょう。

▼ 3 つのエリアの開閉方法



3-3　［Toolbar］（ツールバー）の機能を確認しよう

▼［Toolbar］の構成



❶ ［Run］（実行）ボタン

プログラムを、**ビルド・実行**できます。

ビルドとは何なのでしょう？　実は、私達が書く Swift コードは直接コンピュータが理解することはできません。

ビルドとは、コードをコンピュータが理解できる最終的な実行ファイルとして作成することです。

プログラミング関連の書籍にはよく出てくる用語なので、覚えておきましょう。

❷ ［Stop］（ストップ）ボタン

実行中のプログラムを停止します。

❸ ［Scheme］（スキーム）メニュー

プロジェクトで実行する、「iOS シミュレータ」を選択できます。

iOS シミュレータは、Xcode で利用できる仮想デバイスです。実際に iPhone、iPad などを Mac につながなくても、Xcode 上でアプリの動作を確認できます。

たとえば iPhone 11 を持っていなくても、iOS シミュレータを起動することで、iPhone 11 ではどのようにアプリが表示されるのかの確認が行えます。

❹ ［Activity viewer］（アクティビティビューワ）

実行中のプログラムの状態や、ビルドの進捗状況が表示されます。

❺ ［Library］（ライブラリ）ボタン

画面の View パーツ（部品）や、プログラムのコードスニペット（テンプレート）が収められています。

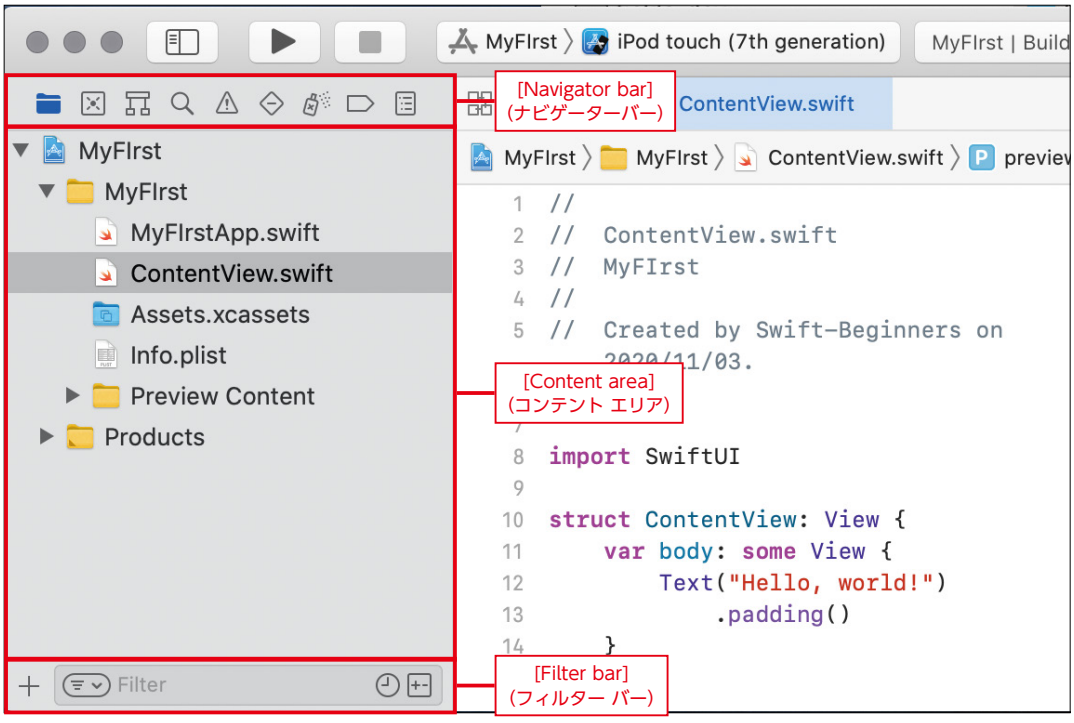
❻ ［Code Review］（コードレビュー）ボタン

プログラムコードの変更履歴を記録・追跡できる Git というバージョン管理システムを活用する際に使用します。本書では使用致しません。

3-4　Navigator area（ナビゲータエリア）を理解しよう

ウィンドウの左側にある［Navigator area］（ナビゲータエリア）で、プロジェクトに含まれるファイル選択や操作ができます。

▼［Navigator area］の各名称



［Navigator area］の中は更に 3 つのエリアがあり、様々な機能を提供しています。それぞれの役割を見ていきましょう。

- **［Navigator bar］（ナビゲータバー）**
  ［Navigator bar］には、各種ナビゲータが用意されています。よく利用するナビゲータに絞って、説明します。

▼［Navigator bar］のよく利用する機能

ナビゲータ	説明
❶［Project navigator］（プロジェクトナビゲータ）	プロジェクトのファイルの一覧が表示されます。ファイルの追加・削除が行えます。この一覧で、ファイルを選択すると、エディタエリアにファイルの中身が表示され、編集ができるようになります。
❷［Find navigator］（検索ナビゲータ）	検索オプションとフィルタを利用して、プロジェクト内の文字列を検索できます。
❸［Issue navigator］（問題ナビゲータ）	プロジェクトのコードを分析し、診断・警告・エラーなどの問題を表示します。

- **［Content area］（コンテントエリア）**
  プロジェクトの各フォルダやファイルが表示されます。ファイルを選択して、エディタエリアに表示、または編集をします。
- **［Filter bar］（フィルタバー）**
  ［Filter bar］に文字を入力して、［Content area］に表示されるファイルを絞り込むことができます。

Point

本書では、操作する画面のメニュー項目をわかりやすくするために文章内にもアイコンを設置しています。
たとえば、［Attributes Inspector］、［Run］などです。このアイコンを頼りにメニューやボタンを見つけて操作を覚えてください。

Xcode をより使いやすくするための設定をしよう

このレッスンで学ぶこと

- Xcode をより使いやすくするための設定方法をいくつか確認します。

1 Xcode の環境を設定しよう

Xcode の［Preferences］（プリファレンス：環境設定）を、自分好みに設定することでより使いやすくなります。

環境設定以降は、再度設定するまでその設定が継続されるので、最初に Xcode の環境設定をしましょう。

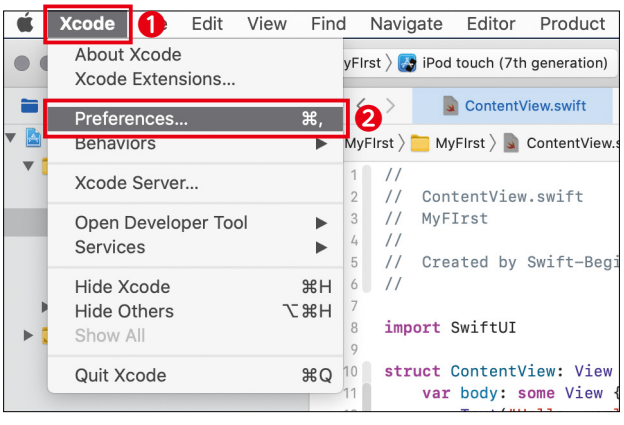
ここでは、行数番号の表示設定、フォント & カラーの設定、コード折りたたみ機能の設定、Minimap（ミニマップ）の設定方法を説明します。

❶ アプリケーションメニューから［Xcode］を選択します。

❷［Preferences］をクリックします。

［Preferences］（環境設定）画面が表示されます。ここでは、Xcode をより使いやすくするための設定が行えます。

▼ プロジェクトテンプレートの選択

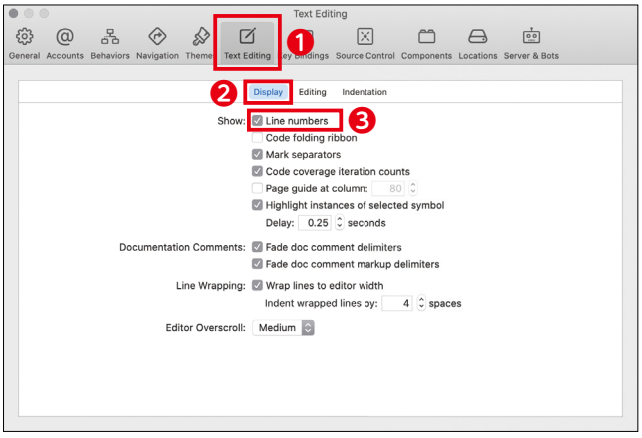


1-1　［Line numbers］（ラインナンバー：行番号）の表示

❶［TextEditing］を選択します。［Text Editing］では、プログラムを書いていくためのエディタの設定ができます。

❷［Display］をクリックして、❸［Line numbers］（行番号）にチェックを入れます。この設定でエディタに行番号が入り、コードが探しやすくなります。

▼ [Text Editing]-[Display]



▼ 行番号なし、行番号あり

行番号なし

MyFIrst 〉 MyFIrst 〉 ContentView.swift 〉 No Selection

```
//
//  ContentView.swift
//  MyFIrst
//
//  Created by Swift-Beginners.
//

import SwiftUI

struct ContentView: View {
    var body: some View {
        Text("Hello, world!")
            .padding()
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

行番号あり

MyFIrst 〉 MyFIrst 〉 ContentView.swift 〉 No Selection

```
1  //
2  //  ContentView.swift
3  //  MyFIrst
4  //
5  //  Created by Swift-Beginners.
6  //
7
8  import SwiftUI
9
10 struct ContentView: View {
11     var body: some View {
12         Text("Hello, world!")
13             .padding()
14     }
15 }
16
17 struct ContentView_Previews: PreviewProvider {
18     static var previews: some View {
19         ContentView()
20     }
21 }
```

Point

通常の文書を書くときは、行番号を意識することは少ないですが、プログラムを書いていくエディタでは、行番号はとても重要です。

行番号を頼りに、コードのボリュームを見積もったり、コードの位置を特定することができたりと、とても便利です。

デフォルトでは、表示されるように設定されていますが、表示されていない場合は、この設定を確認してください。