これからつくる

iPhoneアプリ開発入門

Swiftではじめるプログラミングの第一歩

藤治仁　徳弘佑衣　小林加奈子　小林由憲 =著

SB Creative

本書に関するお問い合わせ

この度は小社書籍をご購入いただき誠にありがとうございます。小社では本書の内容に関するご質問を受け付けております。
本書を読み進めていただく際に不明な箇所がございましたらお問い合わせください。
なお、お問い合わせに関しましては以下のガイドラインを設けております。恐れ入りますが、ご質問の際には最初に下記ガイドラインをご確認ください。

ご質問の前に

小社ウェブサイトで「正誤表」をご確認ください。最新の正誤情報を下記のウェブページに掲載しております。

本書サポートページ：http://isbn.sbcr.jp/87148/

上記ページの「正誤情報」のリンクをクリックしてください。なお、正誤情報がない場合、リンクをクリックすることはできません。

ご質問の際の注意点

●ご質問はメール、または郵便など、必ず文書にてお願いいたします。お電話では承っておりません。
●ご質問は本書の記述に関することのみとさせていただいております。
　従いまして、○○ページの○○行目というように記述箇所をはっきりお書き添えください。
　記述箇所が明記されていない場合、ご質問を承れないことがございます。
●小社出版物の著作権は著者に帰属いたします。従いまして、ご質問に関する回答も基本的に著者に確認の上回答しております。
　これに伴い返信は数日ないしそれ以上かかる場合がございます。あらかじめご了承ください。

ご質問送付先

ご質問については下記のいずれかの方法をご利用ください。

ウェブページより
上記のサポートページ内にある「この商品に関する問い合わせはこちら」をクリックすると、メールフォームが開きます。
要綱に従ってご質問を記入の上、送信ボタンを押してください。

郵送
郵送の場合は下記までお願いいたします。

〒106-0032
東京都港区六本木2-4-5
SBクリエイティブ　読者サポート係

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、体験しながら学べる入門書」です。

iOSはAppleがMac OSをベースに開発した、iPhone、iPad、iPod Touch向けのモバイルOSです。

執筆陣は本書の構成段階からハンズオンセミナーを開催し、たくさんの初心者の方々の声をまとめ、各レッスンを構成しました。

レッスンごとに、ハンズオンセミナー開催後に、参加者の方々からフィードバックをいただく時間を設け、iOS開発の初心者の方々がわからないポイントはどこで、どの操作でつまずくのかを丁寧に調査しました。

また、執筆陣は「Swiftビギナーズ倶楽部」というコミュニティを継続開催しており、プログラミング自体がはじめての方からたくさんの意見を取り入れています。

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、少しずつ階段を登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。

執筆陣は、iOSアプリ開発者や勉強会の主宰者、セミナーの講師、イベントでのスピーカーなど、多彩な経験者で構成されています。

これからiOSアプリを制作される方のために、思いを込めて執筆しました。

ようこそ、iOSアプリ開発の世界へ。

目　次

Contents

Lesson 4

楽器アプリを作ろう　～音の扱い方を学ぶ～

音もアプリのクオリティを高める大切な要素になります。
ここでは、iOSアプリ開発で音をどのように実装していけばよいのかを体験していきましょう！

Lesson 5

マップ検索アプリを作ろう ～UIパーツの扱いとdelegateを学ぶ～

テキストエリアにキーワードを入力すると、
該当する場所にピンが立つ、マップアプリを開発していきます。
文字を入力できるUIパーツ Text Fieldは、キーボードのリターンキーなどを
カスタマイズすることができますので、その設定方法も学んでいきます。

2日目

Lesson 1

タイマーアプリを作ろう ～画面遷移とデータの保持について学ぶ～

ここでは、画面遷移の方法を学びます。タイマー画面と設定画面の2つの画面がある、
タイマーアプリを作ります。タイマー画面で秒数のカウントダウンを行い、
設定画面でタイマーの秒数を選ぶことができます。

Lesson 2

SNS投稿ができるカメラアプリを作ろう ～前半～

カメラを利用することができる「UIImagePickerController」と、カメラロールに保存したりSNS投稿ができる「UIActivityViewController」を利用して、カメラアプリを開発していきます。画面のレイアウトを設定するAutoLayoutの機能や、UIパーツとプログラムを関連付けする方法も学んでいきましょう。

Lesson 3

SNS投稿ができるカメラアプリを作ろう ～後半～

カメラアプリには、エフェクト機能が定番機能になっていることが多いです。エフェクト機能を利用できる「Core Image」を利用して開発していきます。また、画面遷移を実現する「Segue」の応用として、画面遷移するときに情報を渡すことを学びます。

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～

ネットからお菓子の情報を「JSON」形式で取得し、iPhoneに表示してみましょう。ここでは、インターネットのデータを有効活用するために、WebAPIとJSONについて学んでいきます。そして、取得したデータをTable Viewを使って、リストで表示します。

0：この本の読み方と使い方

0-1 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。

0-2 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。また、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していきましょう。

0-3 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、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。

そして、本書を読み終えた人向けに、本書の公式サイトにさらなるサンプルアプリを用意しています。ぜひ、チャンレンジしてみてください。

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。

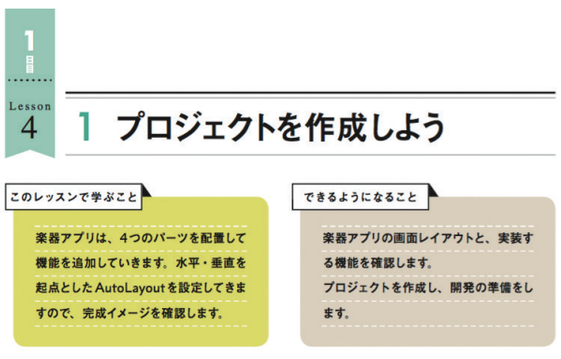
0-4 本書の構成

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。

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。

0-5 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プログラムソースコード

1-4 Playボタンの処理を、1-1で作成したsoundPlayerメソッドに差し替え

```
@IBAction func play(sender: AnyObject) {
  soundPlayer(&backmusicPlayer ,path:guitarPath, count: -1)
}
```

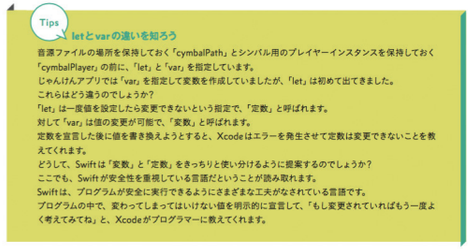
シンバルやギターのコードとの違いは、ループ回数を「count: -1」と指定している点です。
「-1」を指定することで、エンドレスでループ再生をしています。

Swiftのソースコードを掲載しています。ソース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ていきます。

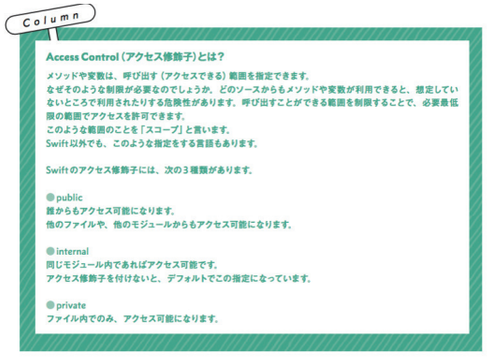
※Swift（スウィフト）は、iOSアプリを作るためのプログラミング言語です。
※Xcode（エックスコード）は、iOSアプリを視覚的に開発するための統合開発環境です。

③Tips（ワンポイント）・Column（技術コラム）

Tips（ワンポイント）は、いまの学習の中での補助的な情報や、知っておいたほうがよいことを記載しています。さらに小さな情報は「MEMO」として掲載しています。



Column（技術コラム）は、本文の流れからは少しそれますが、重要な技術や使い方を記載しています。



0-6 さらなるステップアップ！　～カスタマイズ編について～

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「カスタマイズ編」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、機能アップさせながら学んでいきます。

カスタマイズ編が難しいと感じられた方は、最初の学習ではカスタマイズ編を飛ばして学習を進めてみてください。2回目以降からはカスタマズ編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。

0-7 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

公式サイトでは、本書の内容に関するサポートや、書籍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://swiftbg.github.io/swiftbook/

本書のサンプルアプリダウンロードと使い方

https://swiftbg.github.io/swiftbook/sample/

0-8 本書を読了された方のための応用編講座

本書を最後まで読了された方には、応用編として、さらなる学習教材を用意しています。少し難易度が上がりますが、やりがいがあると思いますので、ぜひ、チャレンジしてみてください！

応用編テキスト・サンプルアプリのダウンロード

https://swiftbg.github.io/swiftbook/advanced/

1：ご利用の前に必ずお読みください

1-1 必要なパソコン機器

iOSアプリ開発には、Macが必要です。

Apple Mac製品

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。

Mac OS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。

1-2 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Mac OS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。

OS X El CapitanまたはMac OS Sierra
Xcode 8.0以上
iOS10.0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

Tips

アプリ作りの勉強をはじめたときに、最初のハードルとなるのが、Xcode（エックスコード）の操作です。目的の画面を作るためにはXcodeを操作する必要がありますが、最初は、「どの場所にどんな機能のボタンがあるのか」がよくわからないため、なかなか学習が進みません。
そういった問題を克服するために、本書では、どんどん体験して、「体感」で覚えていく構成にしました。ぜひ、サンプルアプリを作りながら「体感」してください！

Lesson

1

はじめてのアプリを開発する前に知っておこう

このレッスンでできるようになること

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んでいきましょう。
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。
Appleが開催している開発者向けの大きなイベント「WWDC」についても触れていきます。WWDCでは、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などが公開されています。
「Swift」（スウィフト）言語の概要も学んでいきます。Swiftは2014年に発表された新しいプログラミング言語で、Appleが作りました。Swiftは、iPhoneだけでなくMac、Apple Watchでも動くアプリを作ることができます。

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ていきます。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ていきましょう。

できるようになること

プログラミングは難しいものではなく、楽しんで作るものだと理解すると、気負いがなくなります。

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。

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。

まずは、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。

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。

できるようになること

最初に挫折しそうなポイントに触れているので、心理的なハードルが低くなります。

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、まずは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。

焦ることはありません。

すべてが理解できなくても、前に進んでいきましょう。

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。

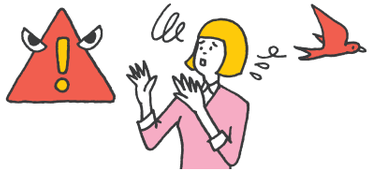
本書では、1日目 Lesson2でアプリ開発の準備にも十分な時間を割いています。事前の準備が終われば、いよいよアプリを作っていきます！

ポイント③：アプリ開発で表示される警告やエラー

アプリを開発していく過程で、「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もって、開発に取り組みましょう。



ポイント④：まずは、Xcodeを体験して慣れていこう

アプリ開発はXcodeという統合開発環境を利用して制作を行います。

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して、慣れていくことが大切です。繰り返し操作を体験し、ぜひ慣れていきましょう。

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。

できるようになること

WWDCの情報をもとに、アプリ開発の環境を理解していきます。開発者がとても気になる課金や報酬の概要についても知っておきましょう。

1：WWDC2016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るため、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/wwdc/

WWDC2016は6月13日～17日にサンフランシスコで開催されました。

WWDCはAppleの製品・サービスが発表されるイベントですが、参加するためのチケットは1,599ドル（WWDC2016チケット価格）で販売されていて、1ドル105円換算で約16万7,000円と、その人気の高さがわかります。

WWDCでは、過去に右のような製品・サービスが発表されています。

WWDCでの製品発表

発表年	製品
1998年	iMac
2006年	Intel XserveとMac Pro
2008年	iPhone3G、MacBook Air
2010年	iPhone4、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	Swift2.0、Swiftオープンソース化
2016年	Swift3.0、Sirikit、iMessage Apps

1-2 App Storeのダウンロード数と開発者への支払額

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？

WWDC2016での公式発表の数値から試算してみたいと思います。

WWDC2016での公式発表によると、App Storeからのアプリダウンロード数は1,300億本を超えており、2008～2016年までの8年間でAppleが開発者に支払った金額は累計500億ドルです。累計額を1ドル105円換算すると、約5.3兆円にもなります。また、登録アプリ数は約200万本であるとされるため、この5.3兆円を200万本のアプリ数で割ると、1つのアプリの平均収入は約265万円になります。

2008年からアプリを1つリリースしていたとすると、年間平均で約33万円、月平均で約2万7千円の収入になります。

そして、登録アプリ数の200万本には無料アプリも含まれていますので、平均は約2万7千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%が開発者への支払い額です。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

課金方法	説明
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

4 Swift（スウィフト）を知ろう

このレッスンで学ぶこと

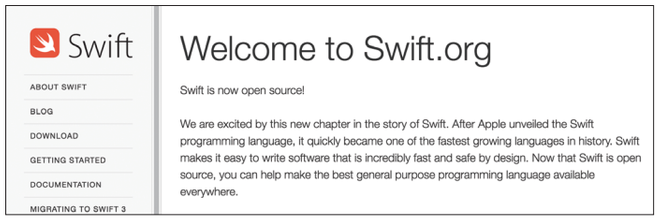
iOSアプリ開発では、Swift（スウィフト）というプログラミング言語を使います。2014年に登場した新しい言語です。Swiftの概要を理解していきます。

できるようになること

Swiftには高速、モダン、安全、インタラクティブといった特徴があります。Swiftを学ぶにつれてこれらのことがよくわかるようになります。

1：Swiftの特徴を押さえよう

みなさんが、iOSアプリを作るために使うプログラミング言語がSwift（スウィフト）と呼ばれるものです。Swiftは2014年のWWDCではじめて発表されました。



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

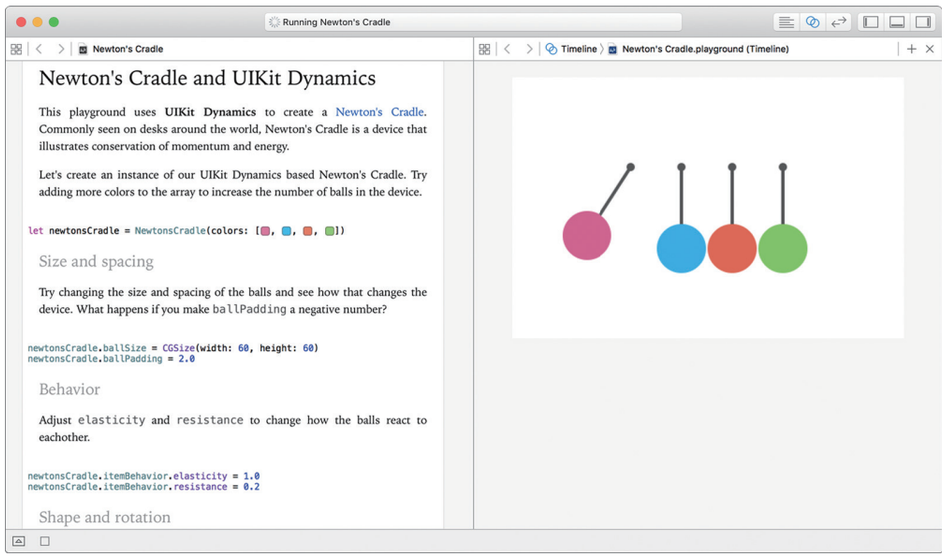
モダン（modern）とは「現代風」という意味です。Swiftでは、他のプログラミングでも採用されている新しい機能を、積極的に取り入れています。

●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：Swiftはインタラクティブである

Swiftはインタラクティブ（対話形式）に動かすことができます。



Playgrounds

Playgrounds（プレイグラウンド）と呼ばれる、Swiftを記述するとリアルタイムで実行結果がわかる機能があり、Swiftの動作を確認する際にとても便利です。

●特徴⑤：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⑥：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。

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、アプリ開発を行っていました。
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。
本書は、これからの開発言語である「Swift」でプログラムを記述していきます。

Lesson

2

アプリ開発の環境を整えて、Xcodeの使い方を学ぼう

このレッスンでできるようになること

iOSアプリを開発するために、必要なものを学んでいきます。
Apple IDを取得されていない方のために、Apple IDの説明と取得方法についても解説します。
そして、アプリ開発をするために必要なXcode（エックスコード）のインストールを学び、iPhoneの画面に「Hello Swift!」と表示できるようなアプリを作ってみます。
最初の簡単なアプリを作りながら、Xcodeにどのような機能があるのか体験していきましょう。
基本の操作を行いながら、iPhone（実機）がなくてもテストが行えるシミュレータの操作や、iPhone本体へのアプリ転送（実機転送）も学びます。

日目

1 iOSアプリを開発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

iOSアプリを開発するために、用意しなければならないものや、それぞれの役割について学んでいきます。

できるようになること

iOSアプリ開発をはじめる前に必要な知識である、MacやApple ID、Xcodeの概要について理解していきます。

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

また、開発したアプリを全世界の人たちに利用してもらえるようにするには、別途、**「Apple Developer Program」**への登録も必要です。

この章では開発したアプリを、iPhoneに転送（実機転送）を行って利用するところまでをゴールにしています。

アプリを開発して世界に公開したいという方は、本書の公式サイトで公開手順を掲載していますので、確認してみてください。

アプリの公開手順
https://swiftbg.github.io/swiftbook/release

それでは、必要な3つのものを確認していきましょう。

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、Mac以外のパソコンでもiOSアプリ開発ができる可能性もありますが、本書を執筆している2016年9月時点ではMacが必要になります。

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。

App StoreとiTunes

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、事前に作成しておく必要があります。

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。

一般的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。

Xcode

では、**IDE**とはなんなのでしょう？

IDEと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を使って開発ができる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。

また、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。

本書でも、Xcodeをインストールして開発を進めていきます。

2 Apple IDを取得しよう

このレッスンで学ぶこと

Apple IDを作成する手順を学びます。
また、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。

できるようになること

Apple IDが取得できるようになります。
また、Apple IDのメールアドレスの変更も行えるようになります。

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がよくわからないときがあります。

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を変更する - 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

次の手順にそって、一緒に作成していきましょう。

2-1 Apple IDのサイトにアクセス

Apple IDを管理 - Apple

https://appleid.apple.com/

［Apple IDを作成］をクリックします。

Apple IDログイン画面

2-2 Apple IDを作成

各項目を入力していきましょう。

個人の情報と、個人確認をするときのセキュリティ質問などを設定します。

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。

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。

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。

［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を作成

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。

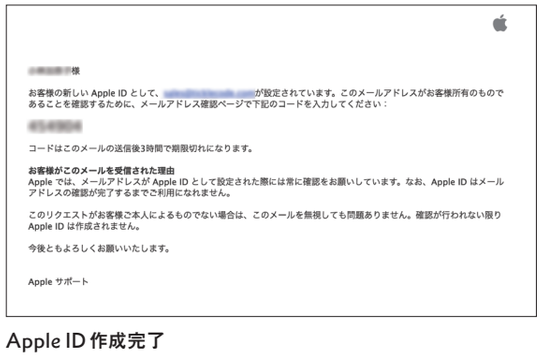
メールを確認すると、赤枠の中に確認用のコードが記載されていますので、その数字を右の画面に入力します。

Apple IDコード入力

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。



お客様の新しい Apple ID として、が設定されています。このメールアドレスがお客様所有のものであることを確認するために、メールアドレス確認ページで下記のコードを入力してください：

コードはこのメールの送信後3時間で期限切れになります。

お客様がこのメールを受信された理由
Apple では、メールアドレスが Apple ID として設定された際には常に確認をお願いしています。なお、Apple ID はメールアドレスの確認が完了するまでご利用になれません。

このリクエストがお客様ご本人によるものでない場合は、このメールを無視しても問題ありません。確認が行われない限り Apple ID は作成されません。

今後ともよろしくお願いいたします。

Apple サポート

Apple ID作成完了

Tips

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。

3 Xcodeをインストールしよう

このレッスンで学ぶこと

Xcodeを使って開発を進めていくために、App Storeから、Xcodeをダウンロードしてインストールする手順を学びます。

できるようになること

App Storeから、Xcodeのダウンロードとインストールができるようになります。iOS開発を行える環境が整います。

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

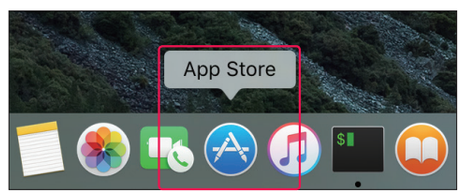
本書では、Xcodeを利用して開発を進めていきますので、最初にApp StoreからXcodeのダウンロードを行います。

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てしまわ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約5GBほどあり大きいので、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おいてください。

1-2 App Storeからのインストール

MacのDocにある「App Store」アイコンをクリックして起動します。



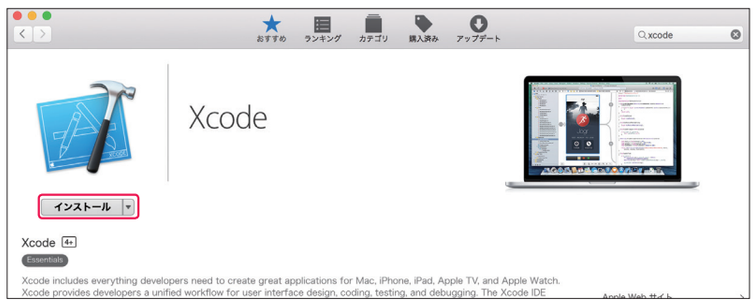
Docから App Storeを起動

App StoreからXcodeを検索

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

検索結果からXcodeを選択

検索結果から「Xcode」をクリックします。



Xcodeインストール

Xcodeの詳細ページが表示されます。［インストール］をクリックすると、Xcodeのダウンロードとインストールがはじまります。

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方法やXcodeの基本的な画面構成について学びます。使用頻度の高いナビゲーションやボタンを学習します。

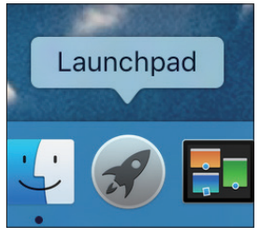
できるようになること

プロジェクトの作成ができるようになります。Xcodeの画面の配置やボタンの概要について知っておくと、あとの作業の効率がよくなります。

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が、今回はLaunchpadから、Xcodeのアイコンをクリックして起動します。

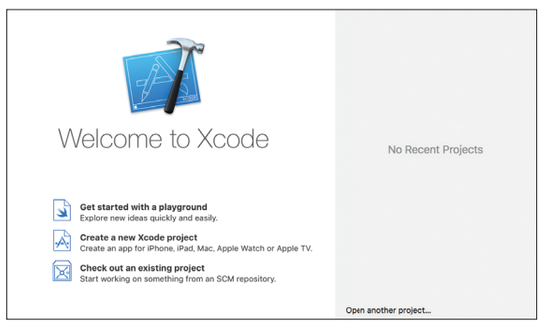


Launchpadから Xcodeを起動

1-2 Xcodeの起動画面

「Welcome to Xcode」と表示されている下に3つのメニューが表示されています。

1つずつ見ていきましょう。



Xcode起動画面

Get started with a playground

Xcode6から、Playground（プレイグラウンド）という機能が追加されました。

Playgroundとは「遊び場」という意味で、Swiftのコードを書きながら結果がすぐに確認できます。Swiftを書きながら、動きを確認するのに適しています。

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、アプリを開発していきます。

Check out an existing project

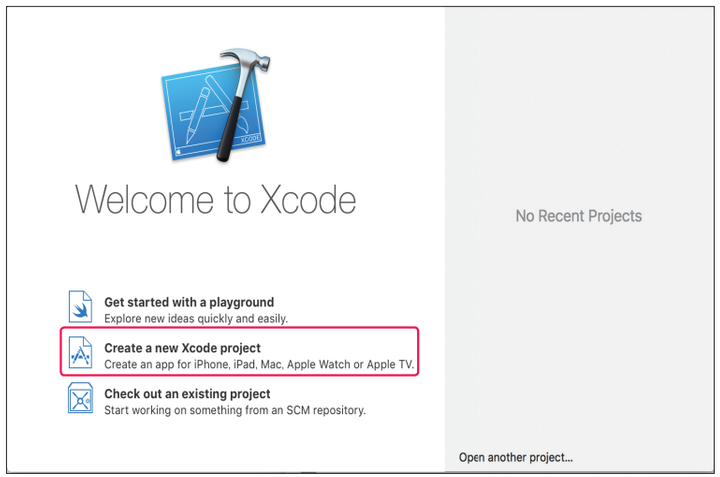
バージョン管理システムを使ってプロジェクトを作成する方法です。

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。

高度な機能になりますので、本書では取り扱いません。

2：プロジェクトを作成しよう

2-1 プロジェクトを作成



Xcode起動画面

では、[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。

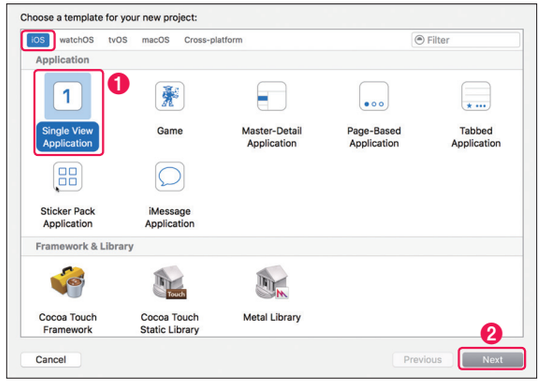
プロジェクトはアプリ開発を束ねるものです。1つのプロジェクトで1つのアプリを開発します。

プロジェクトの中に、アプリの画面設計や画像ファイル、プログラムが入っています。

[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）を選ぶ画面になります。

この画面でプロジェクト、テンプレートを選ぶことができます。ではさっそく、❶「Single View Application」を選択して、❷ [Next] をクリックしましょう。



プロジェクトテンプレートの選択

Tips

開発するアプリの構成に近いテンプレートを選んで、プロジェクトを生成する方法もあります。どのようなテンプレートがあるのかをいくつか見てみましょう。

Master-Detail Application

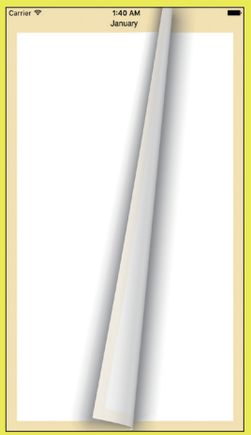
[Table View] という行で、情報を表示できるパーツが搭載されたテンプレートを生成してくれます。プラスマークをタップすると日時が追加されるという、シンプルな動きです。

Master-Detail Application

Page-Based Application

これは、ページをめくるような動きをしてくれるテンプレートです。

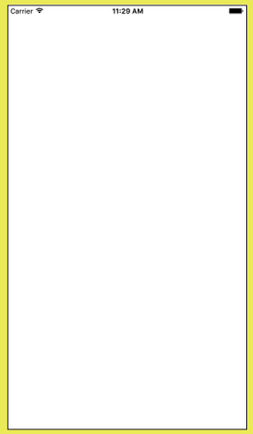
簡単にこのようなアニメーションが使えると、とても便利ですね。



Page-Based Application

● Single View Application

いちばんシンプルなテンプレートです。
いままでのテンプレートのように、機能やアニメーションは実装されていません。
本書では、一からアプリの作り方を学んでいきますので、なにも実装されていないこのシンプルなテンプレートを選択して開発を進めていきます。

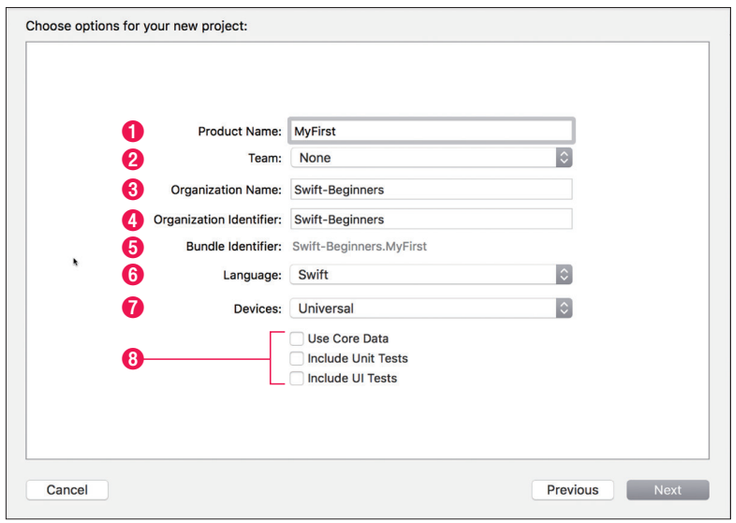


Single View Application

その他にもさまざまなテンプレートが用意されていますので、興味がある方は実際にテンプレートからプロジェクトを作成して確認してみてください。

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。



プロジェクトの新規作成

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見ていきましょう。

❶ Product Name（製品名）

プロジェクトの名前です。アプリの初期段階での名前になります。プロジェクトの名前には好きなものが入力できます。ここでは、「MyFirst」と入力します。

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が必要ありませんが、実機転送を行う際には設定が必要になります。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 Name（組織名）

会社名、組織名を入力します。本書では「Swift-Beginners」と入力していますが、自由に入力していただいて大丈夫です。

❹ Organization Identifier（組織識別名）

この項目は［Bundle Identifier］の設定にかかわります。
［Bundle Identifier］は世界中のアプリの中でユニーク（一意）にする必要があるので、よく利用される方法としては、ドメイン表記を逆にした記述（逆ドメイン）を入力します。
ドメインが「swift-beginners.jp」の場合は、「jp.swift-beginners」と設定します。
ドメインを持っていない方は、メールアドレスを逆から設定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、ユニークになります。
例）swift@example.comの場合は、com.example.swiftなど
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、ご自身のホームページのドメインのように、重複しにくいIDを入力してください。

❺ Bundle Identifier

ここに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。
世界中のアプリの中で、開発するアプリを識別する必要があります。

❻ 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の選択です。「Swift」と「Objective-C」が選べます。
「Swift」を選択します。

❼ Devices（デバイス）

開発するアプリが、どのデバイス向けか指定をします。

iPhone、iPad両方で使えるアプリであれば、「Universal」を選択します。

今回は「Universal」を選択します。

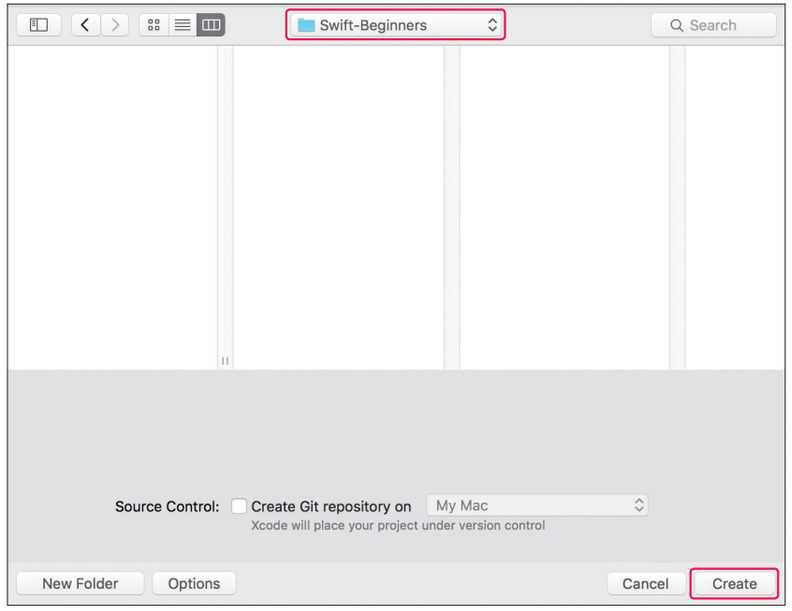
❽ Use Core Data、Include Unit Tests、Include UI Tests

すべてチェックを外してください。これらの項目は、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指定です。

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、「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ていきます。

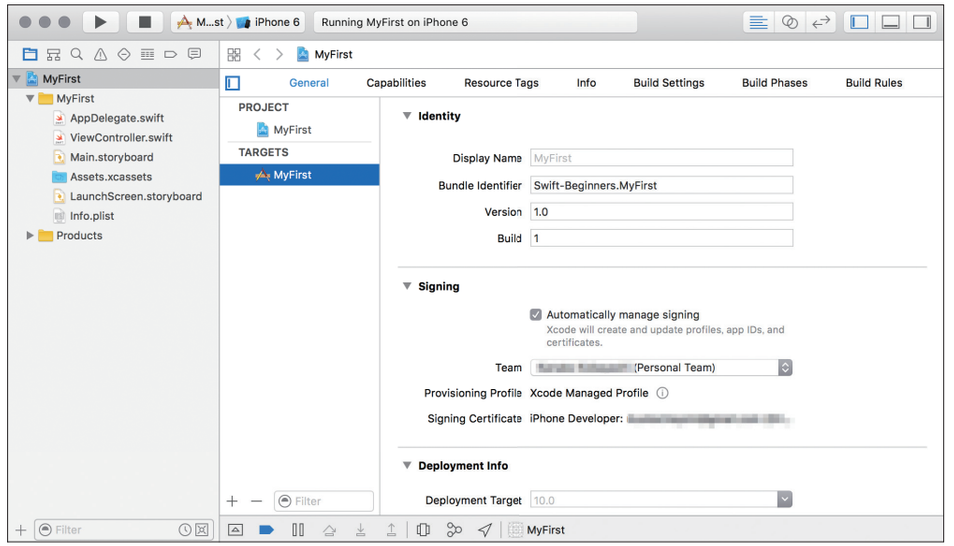
[Create] ボタンをクリックします。



プロジェクトの保存場所

プロジェクトが作成され、Xcodeプロジェクト画面が表示されます。

このXcodeプロジェクト画面でアプリの画面を作成したり、プログラムを書いていきます。

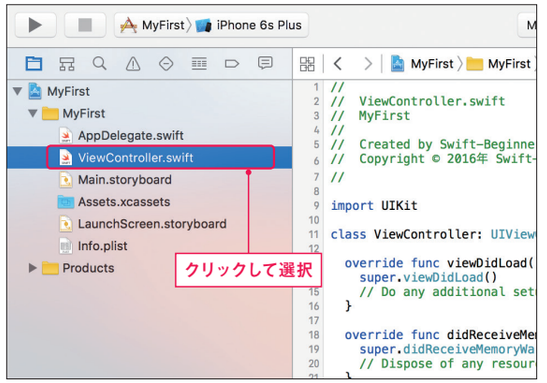


Xcodeプロジェクト画面

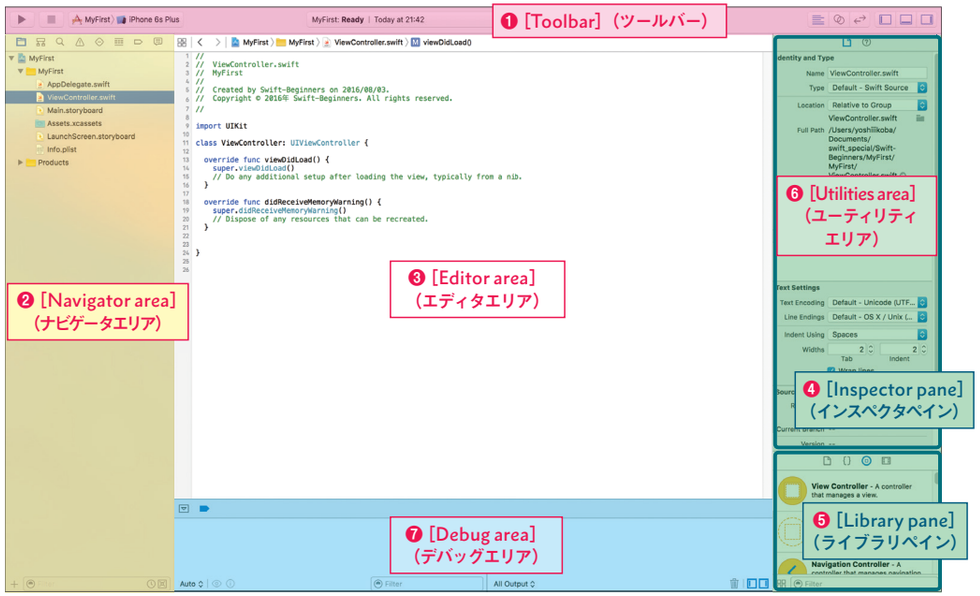
3：Xcode画面のナビゲーションを理解しよう

最初は、Xcodeプロジェクト画面が表示されています。

まずは、[ViewController.swift] を選択してみてください。ここからは、Xcodeのナビゲーションや画面配置を見ていきます。



[ViewController.swift]の選択



Xcode画面構成

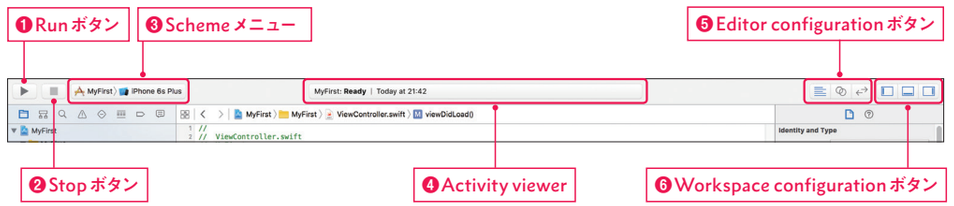
Xcodeはいくつかの画面エリアに分かれています。

Xcodeの画面エリア

画面のエリア	役割
❶ [Toolbar]（ツールバー）	画面上部のメニューでアプリを実行したり、画面を切り替えたりする
❷ [Navigator area]（ナビゲータエリア）	左サイドのナビゲータエリアで、ファイルを選択する
❸ [Editor area]（エディタエリア）	プログラムを書いたり、画面を作成したりする
❹ [Inspector pane]（インスペクタペイン）	エディタエリアで、表示または選択されているパーツ、ソースファイルに対応する情報を、表示、編集できる
❺ [Library pane]（ライブラリペイン）	テキスト、ラベルといったUIパーツなどが集まっている
❻ [Utilities area]（ユーティリティエリア）	[Inspector pane] [Library pane] などが配置されているエリアを指す
❼ [Debug area]（デバッグエリア）	シミュレータ実行時などに状況が出力される場所。自分でプログラムを書いて出力できるので、動作確認やテストを行うときにも使用する

まずは、ツールバーの機能から見ていきましょう。

3-1 [Toolbar]（ツールバー）



[Toolbar]の構成

❶ [Run]ボタン（実行ボタン）

プログラムを、ビルド・実行できます。

ところで、**「ビルド」**とはなんなのでしょう？

「ビルド」は、私たちが書いたコードをコンピューター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]ボタン（ストップボタン）

実行中のプログラムを停止します。

❸ [Scheme]メニュー（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iPhone6s Plusを持っていなくても、iOSシミュレータを起動することで、iPhone6s Plus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ー）

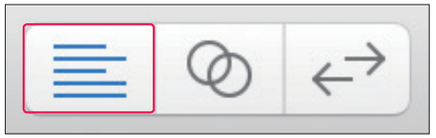
実行中のプラグラムの状態や、ビルドの進捗状況が表示されます。

❺ [Editor configuration]ボタン

Xcodeの画面中央にあるプログラムコードの表示方法を指定します。

[Standard editor]（左のボタン）

選択したファイルの内容が表示されます。

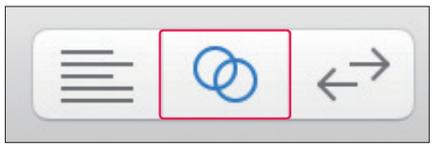


[Standard editor]

[Assistant editor]（中央のボタン）

［Assistant editor］を選択すると、画面が2つに分割されます。

編集中のファイルの画面（View）出力結果が表示されます。

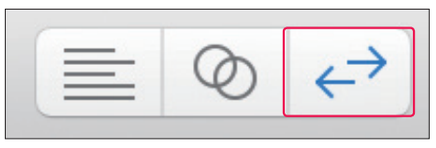


[Assistant editor]

[Version editor]（右のボタン）

選択しているファイルが修正されている場合に、差分を表示してくれます。

この機能は、プロジェクトがバージョン管理の対象になっている場合に利用できます。



[Version editor]

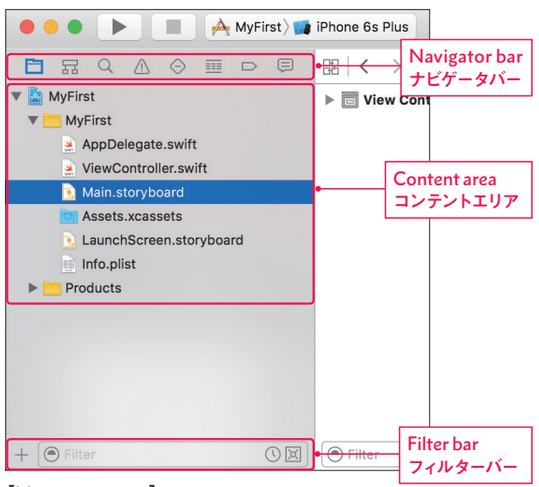
❻ [Workspace configuration]ボタン

左のアイコンから① [Navigator area]、② [Debug area]、③ [Utilities area] のそれぞれを表示・非表示に切り替えることができます。

[Workspace configuration]ボタン

3-2 [Navigator area]（ナビゲータエリア）

［Navigator area］はXcodeの左端に表示されているナビゲーションで、［Navigator bar］［Content area］［Filter bar］で構成されています。



[Navigator area]

[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]

ナビゲータバー

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る。ファイルの追加・削除が行える。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る
❷［Find navigator］（検索ナビゲータ）	プロジェクト内の文字列を検索できる
❸［Issue navigator］（問題ナビゲータ）	プロジェクトのコードを解析し、診断・警告・エラーなどを表示

[Content area]（コンテントエリア）

プロジェクトの各ファイルが表示されます。

[Filter bar]（フィルターバー）

［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。

3-3 [Editor area]（エディタエリア）

開発作業の多くは、［Editor area］で行います。

［Project navigator］で選択したファイルの内容によって、エディタの表示が切り替わります。

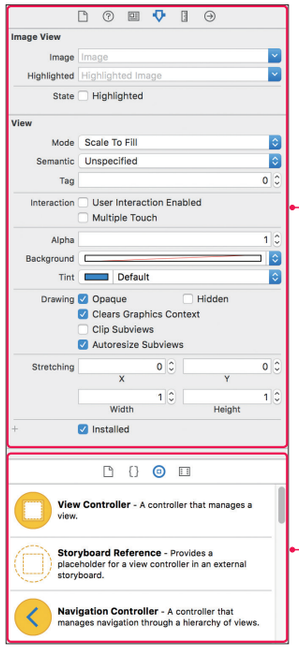
3-4 [Utilities area]（ユーティリティエリア）

このエリアでは、おもに［Storyboard］（ストーリーボード）に関する設定を行います。

［Storyboard］は、アプリのレイアウトや画面遷移を視覚的に作成できます。

ボタンやテキストなどを、ドラッグ＆ドロップの操作で配置したり、画面遷移も簡単に設定できます。

よく利用するものとして、［Inspector pane］と［Library pane］があります。



Inspector pane

Library pane

[Utilities area]

［Inspector pane］は、パーツの文字を大きくしたり色を変えたりと、パーツのカスタマイズが行えます。

［Library pane］にはテキストやボタンなどのたくさんのパーツがあります。このパーツを配置して画面を組み立てていきます。

次からは、実際にプロジェクトを作成して、Xcodeの操作に慣れていきます。

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きますが、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いです。

上記のような理由から、Xcodeはそのまま英語表記で使っていくことが望ましいです。

5 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 Preferences（環境設定）を学んでいきます。

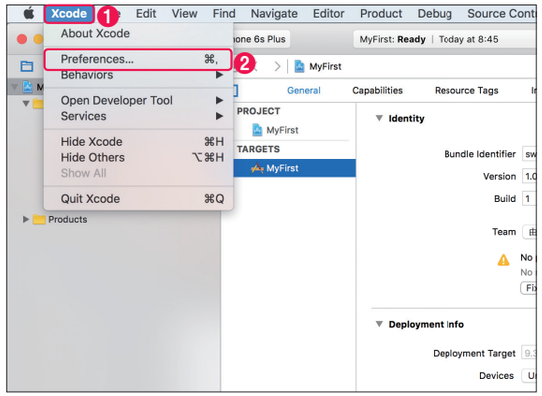
できるようになること

［Preferences］（環境設定）メニューからエディタの［Line numbers］（行番号）、TabやIndent（インデント）や、その他の設定ができるようになります。

1：Xcodeの環境を設定しよう

❶メニューから［Xcode］を選択します。

❷［Preferences］をクリックします。



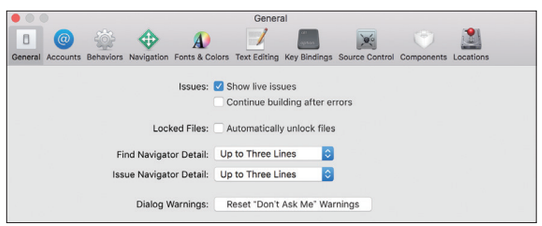
［Preferences］を選択

Memo

［Preferences］では、Xcodeの環境設定ができます。

「環境設定」とは、操作がしやすいように、表示や動作を設定することです。

［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。

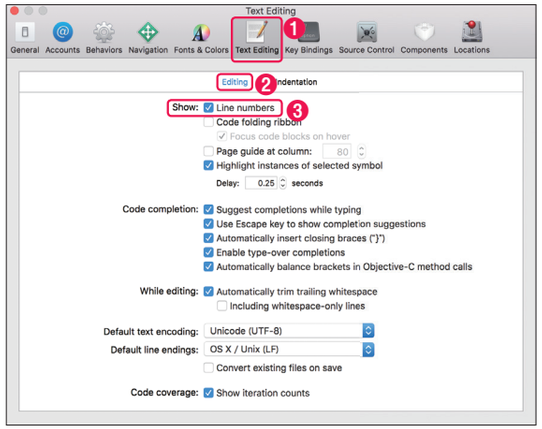


［Preferences］（環境設定）画面

1-1 [Line numbers]（行番号）の表示

Swiftなどのプログラムコードを書いていく場所を**「エディタ」**と呼びます。エディタに行番号があると探しやすく、とても便利です。

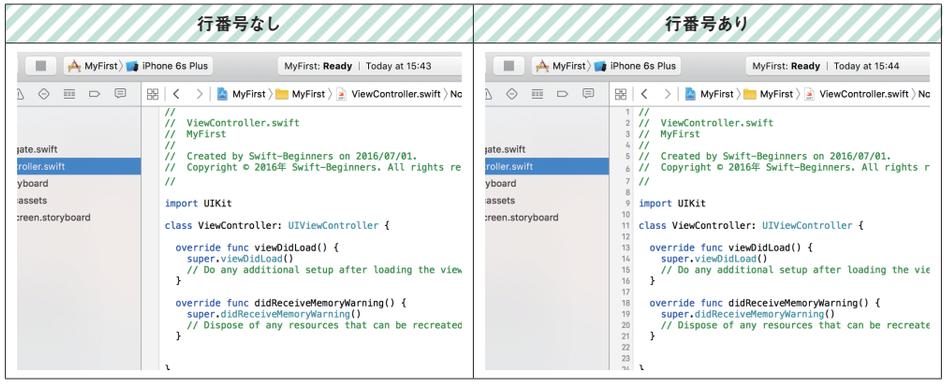
❶ [Text Editing] を選択します。[Text Editing] では、プログラムを書いていくためのエディタの設定が行えます。
❷ [Editing] をクリックします。
❸ [Line numbers]（行番号）にチェックを入れます。この設定でエディタに行番号が入り、コードが探しやすくなります。



[Text Editing] - [Editing]

次のように、行頭に行番号が付加されます。

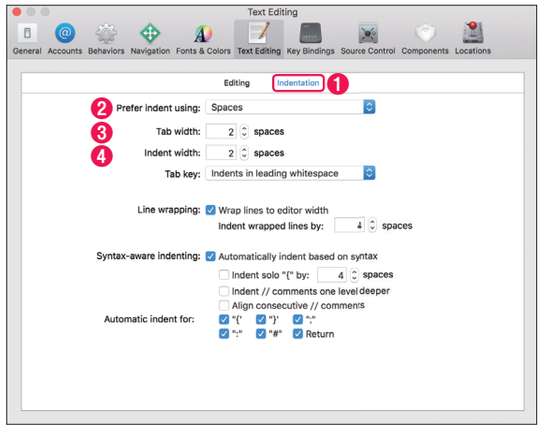
行番号なし、行番号あり



1-2 エディタのTab（タブ）やIndent（インデント）の設定

エディタのTabやIndentを本書と同じ設定にしておくと、学習が進めやすいです。

[Text Editing] の設定で、
❶ [Indentation] をクリックします。
❷ [Prefer indent using] は、インデントに使う文字を指定します。本書では「Spaces」を設定して、インデントには空白文字を使います。
❸ [Tab width] は、「Tab」キーを1回入力したときの文字数です。本書では、「2」spacesで2文字を設定します。
❹ [Indent width] は、インデント階層の文字数を指定します。本書では、「2」spacesを設定します。

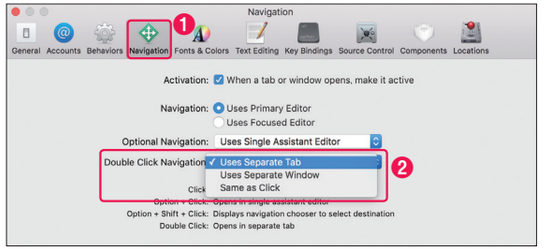


[Text Editing] - [Indentation]

1-3 [Double Click Navigation]で新しいタブを開くように設定

Xcodeのデフォルト設定では、[Navigator] のファイルをダブルクリックすると新しいウィンドウが開いてしまいますが、慣れないうちには作業が混乱しやすいです。

ダブルクリックしたときに、新しいウィンドウを開くのではなく、新しいタブを開くように変更していきます。

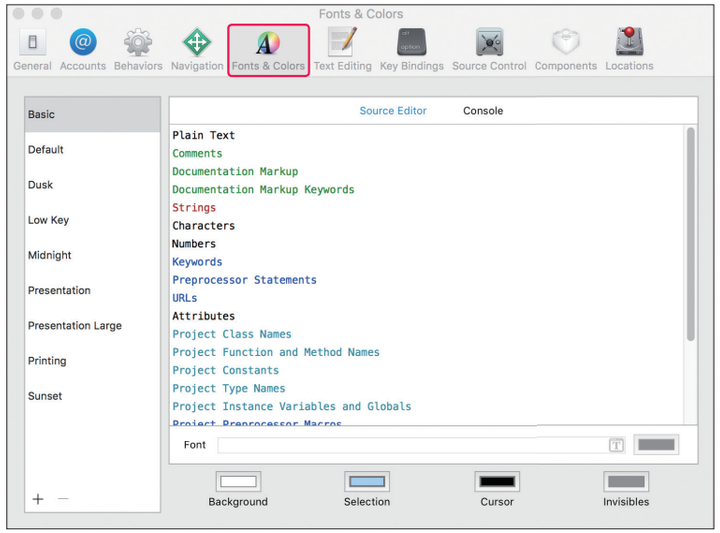


[Navigation]

❶ [Navigation] を選択します。
❷ [Double Click Navigation] で、「Uses Separate Tab」を選択します。

1-4 [Fonts & Colors]の設定

エディタのカラーやフォントサイズを設定できます。好みの設定を選ぶことができます。



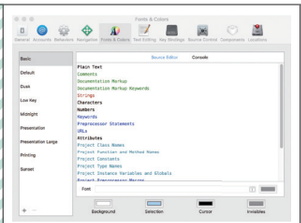
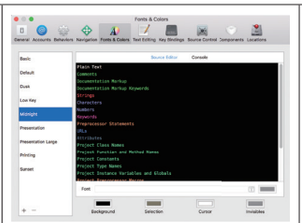
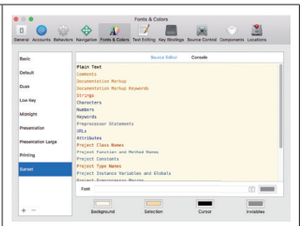
[Fonts & Colors]

Memo

エディタはプログラムコードを書いていく場所になります。Swiftの文法や重要な箇所をハイライトしてくれます。

[Fonts & Colors]を選択して、左から好みの設定を選びます。よく使用されている、代表的な[Fonts & Colors]をご紹介します。

代表的な[Fonts & Colors]

[Fonts & Colors]			
設定の内容	[Basic] デフォルトのカラー設定。本書でも使用	[Midnight] 背景がブラックになる。基本の文字がホワイトになる	[Sunset] 背景が薄いイエローになり、基本の文字がブラックになる

6 「Hello Swift!」と表示してみよう

このレッスンで学ぶこと

画面に「Hello Swift!」と表示する簡単なアプリを作っていきます。Xcodeの基本操作、シミュレータなどの操作を体験して学んでいきます。

できるようになること

Labelパーツの使い方、レイアウト方法、シミュレータ、実際のiPhone（実機）にアプリを転送する実機転送も体験していきます。

1：完成をイメージしよう

「どのようなものを作るのか」を知っておくと、1つ1つの手順が理解しやすくなります。

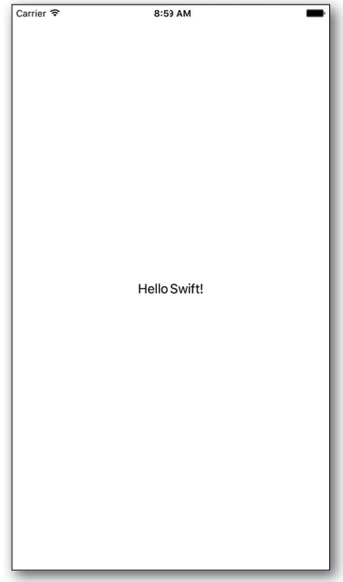
アプリの完成イメージを確認してみましょう。

アプリが起動すると、画面に「Hello Swift!」と表示されます。

1フレーズを表示するシンプルなアプリを作りながら、アプリ開発に必要な機能を体験していきましょう。

Memo

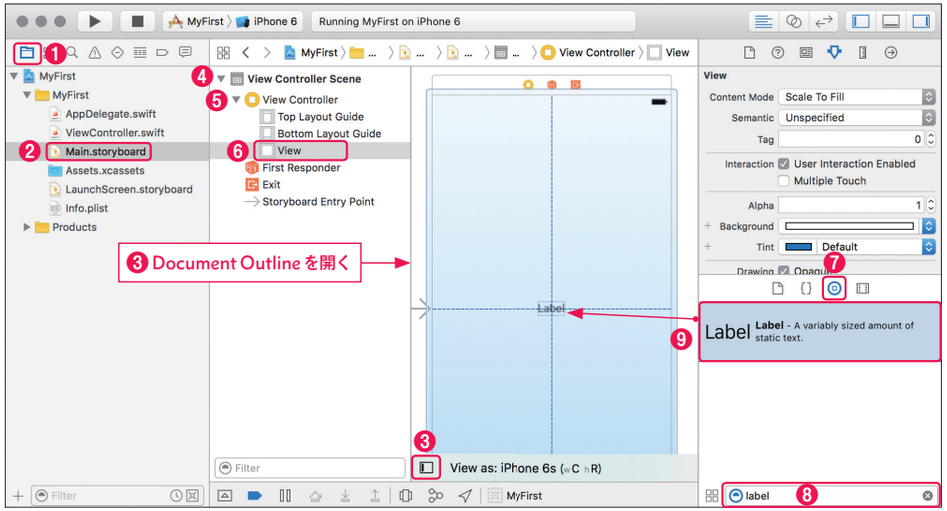
本書では最初に、これから制作するアプリの完成イメージを確認します。完成イメージを念頭に置きながら学習することで、より理解しやすくなります。



完成イメージ

2：Labelを配置して、AutoLayoutを使ってみよう

2-1 Labelを配置



Labelの検索と配置

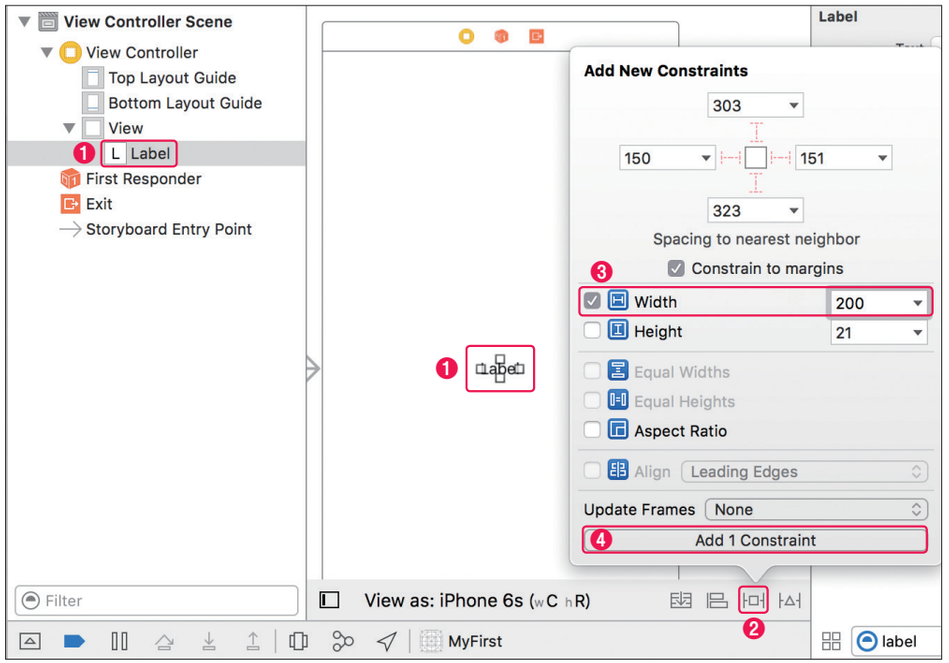
まずは、最初に［Storyboard］へLabelを配置して、Xcodeでの操作を体験してみましょう。

❶［Project navigator］からファイルの一覧を表示します。
❷［Main.storyboard］を選択します。
❸［Document Outline］ボタンをクリックすることで、Document Outlineエリアが開閉します。
今回は、Document Outlineを開きます。
❹［View Controller Scene］の▼マークをクリックします。
❺［View Controller］の▼マークをクリックします。
❻［View］をクリックして選択します。
❼［Object library］を選択すると、パーツの一覧が表示されます。
パーツは「UIパーツ」と呼ばれ、テキストや画像、ラベルなどの画面構成部品のことを示します。
詳しくは、「1日目 Lesson3-2 アプリに必要なパーツの配置と設定をしてみよう」（P.57）で学習します。
❽検索窓に「label」と入力します。**パーツは沢山ありますので、検索窓を使って探すと便利です。**
❾検索されたLabelをStoryboardの中央にドラッグ＆ドロップします。縦と横に青い線が現れたら、中央に位置しているということがわかります。

2-2 AutoLayout（オートレイアウト）を使う

ここでは、**AutoLayout（オートレイアウト）**という機能を使って設定していきます。AutoLayoutは条件を設定していくことで、画面やパーツのレイアウトを整えていきます。この条件のことを「制約」（Constrain）と呼びます。

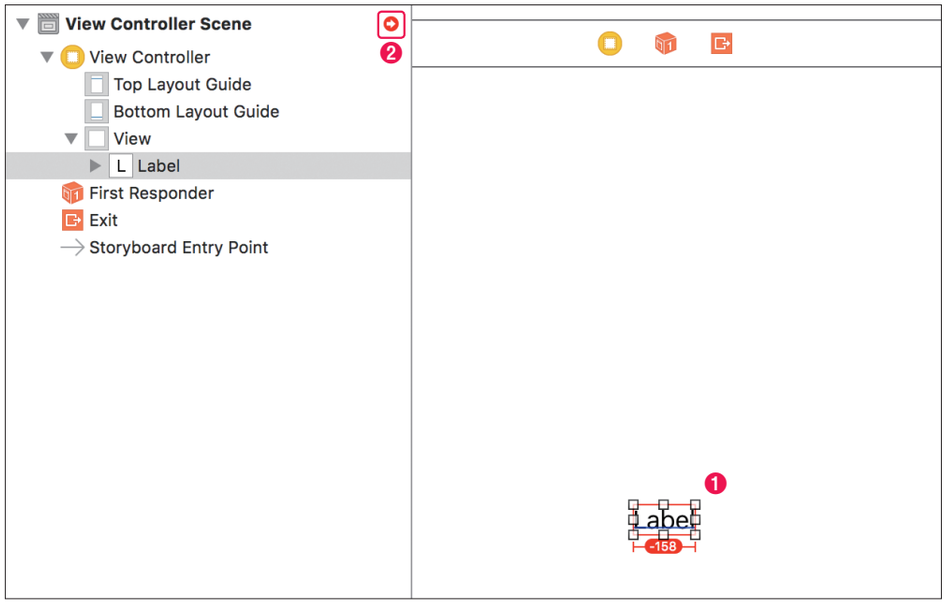
のちほど、「1日目 Lesson3-3 各パーツの表示位置、幅や高さを設定しよう」（P.69）で詳しい解説を行いますので、まずは、Xcodeに触れていきましょう。



AutoLayoutで横幅の指定

Labelの横幅を設定していきます。

❶［Document Outline］から、先ほど配置したLabelを選択します。もしくは、Labelパーツを直接選択します。どちらの方法でも大丈夫です。
❷［Pin］をクリックします。
❸［Width］（横幅）にチェックを入れて、「200」と入力します。
❹［Add 1 Constraint］をクリックして、Labelの横幅に関する制約を追加します。

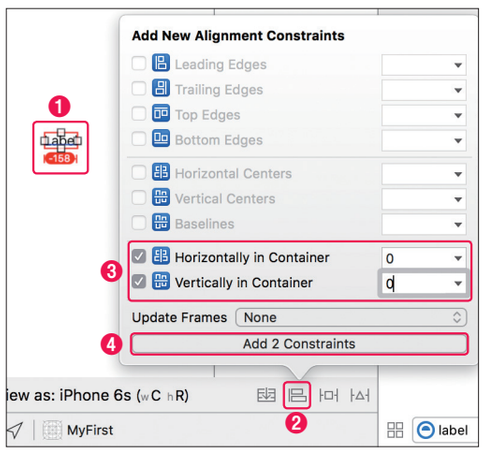


AutoLayoutで中央配置

❶Labelを選択すると、赤い線で囲われていることが確認できます。
❷［View Controller Scene］の横にも赤いマークが表示されています。

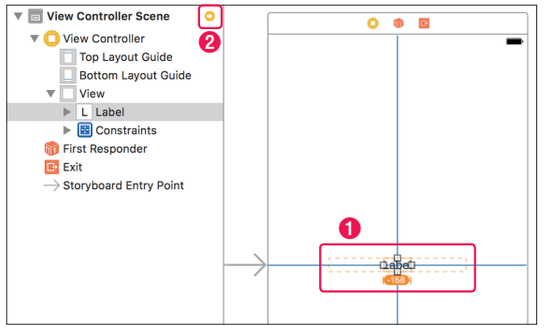
これは、パーツに充分なAutoLayoutの設定ができていないことを表しています。
この状態を解消するために、縦と横の配置を設定して、Labelが中央に配置されるように設定していきます。

❶Labelを選択します。
❷［Align］を選択します。
❸［Horizontally in Container］と［Vertically in Container］にチェックをつけます。
［Horizontally in Container］は、画面中央から水平方向の距離を入力します。「0」であれば、左右の中央に配置されます。ここでは、「0」と入力します。
［Vertically in Container］には、画面中央から垂直方向の距離を入力します。「0」であれば、上下の中央に配置されます。ここでは、「0」と入力します。
❹［Add 2 Constraints］をクリックして、制約を反映します。
先ほどは、［Add 1 Constraint］と表示されていましたが、今回は［Add 2 Constraints］と表示され、数字が変わっています。［Add 2 Constraints］は、「2つの制約を追加する」という意味です。



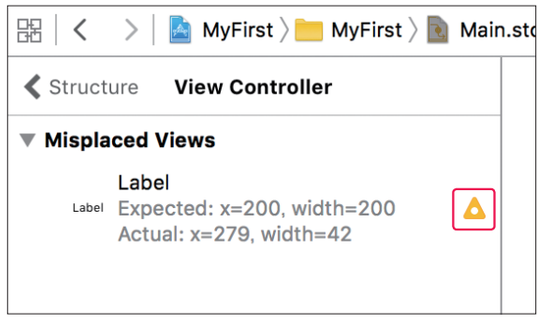
Labelが中央に配置されるよう設定

❶Labelを囲む線が、赤色からオレンジ色に変わりました。これは警告を示していて、AutoLayoutで設定した制約と、実際のパーツの大きさや配置が一致していないことを知らせてくれています。
❷［View Controller Scene］の横の をクリックします。



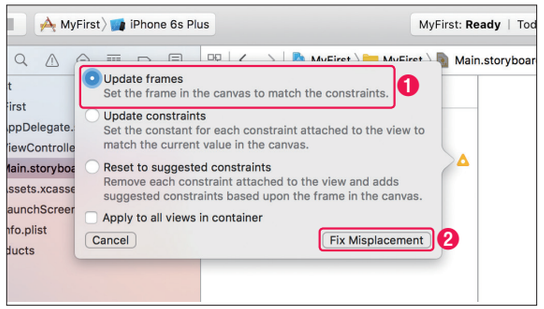
AutoLayoutでの警告

［Misplaced Views］の一覧で、Labelに警告が表示されています。
この状態を解消するために、 をクリックします。



警告マークを選択

❶［Update frames］を選択します。その下にある文言「Set the frame in the canvas to match the constraints.」の意味は、「画面の表示を設定した制約に合わせます」という意味です。
❷［Fix Misplacement］（誤った配置を修正）をクリックして、位置のずれを解消します。

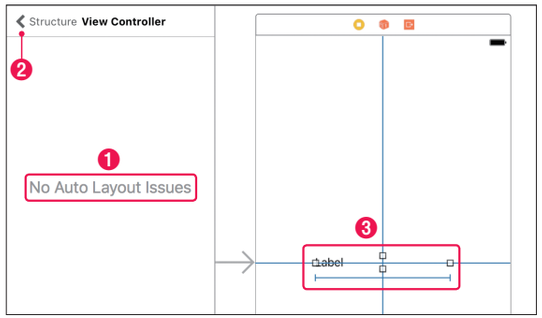


［Update frames］で位置のずれを解消

❶「No Auto Layout Issues」が表示されて、レイアウトに関する問題は解消されたことが確認できます。

❷［Structure］の左横にある矢印 ◀ をクリックして、［Document Outline］に戻ります。

❸LabelでAutoLayoutの設定と差異があることを意味するオレンジ色の線や数値が、消えていることを確認します。また、Labelの横幅と高さが変化していることも確認します。これで、Labelが正しく配置されました。



レイアウトに関する問題は解消

2-3 Labelの文字を中央に配置

Labelが画面の中央に配置されたので、Labelに表示する文字も中央揃えに設定していきます。

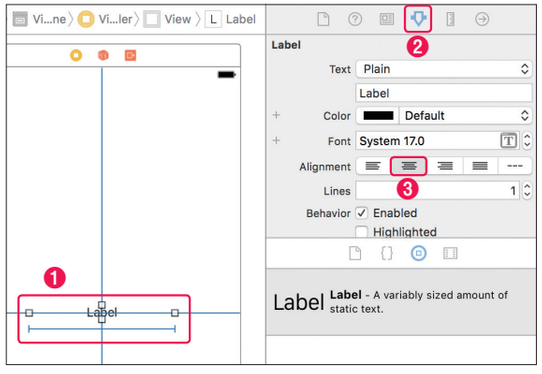
❶Labelを選択した状態で、

❷［Attributes Inspector］をクリックします。

❸［Alignment］で中央揃えを選択すると、Labelの文字が、中央配置になります。

Memo

Xcodeは変更内容を自動的に保存します。ファイルの保存を気にせず、操作を進めてください。



文字の中央配置

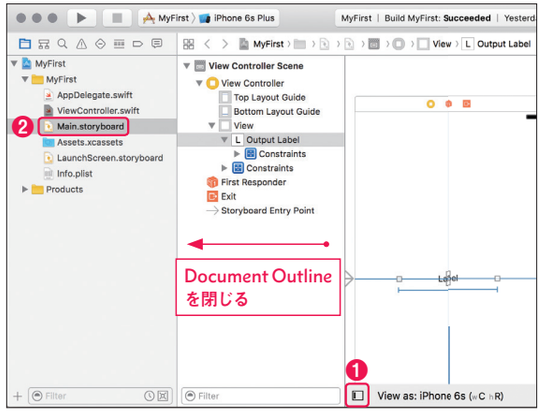
3：プログラムを書いてみよう

3-1 Labelとプログラムの関連付け

Xcodeでは、配置したLabelにプログラムを関連付けする作業を行います。

❶画面の表示を大きくしたいので、［Document Outline］ボタンをクリックしてDocument Outlineを閉じます。

❷［Main.storyboard］を選択してください。

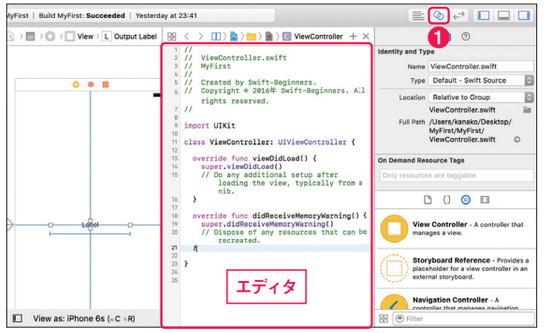


［Document Outline］を閉じる

［Main.storyboard］を選択している状態で、❶［Assistant editor］をクリックすると画面が分割され、エディタが表示されます。

この状態で、［Main.storyboard］に配置されたパーツとプログラムのコードを関連付けます。

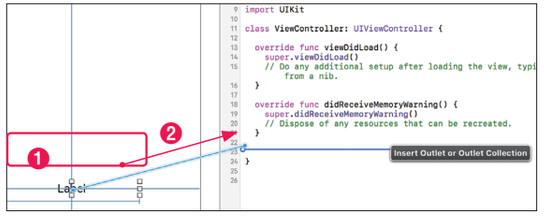
関連付けの詳しい説明は「1日目 Lesson3 じゃんけんアプリを作ろう」(P.53)で行いますので、ここでは簡単に体験してみましょう。



エディタの表示

❶Labelを選択した状態で、

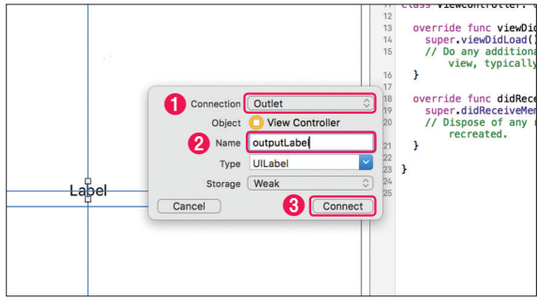
❷「control」キーを押しながら、エディタの最後の「}」(波括弧)上にドラッグ&ドロップします。



Labelからエディタへドラッグ&ドロップ

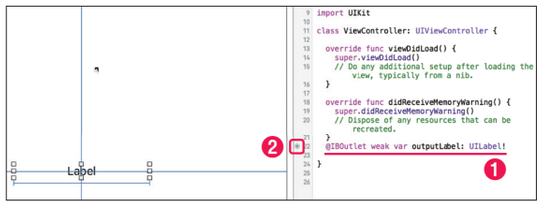
ダイアログが表示されます。Labelとプログラムをどのような名前で関連付けするのかを設定します。

❶ [Connection] の「Outlet」を選択します。
❷ [Name] には、「outputLabel」と設定します。
❸ [Connect] をクリックします。



関連付けの設定

❶「@IBOutlet weak var outputLabel: UILabel!」というコードが追加されています。これが、Labelとの関連付けを示すプログラムコードになります。
❷黒い丸が、Labelと関連付けがされているというマークになります。



@IBOutletの追加

3-2 Labelに文字をセット

```
override func viewDidLoad() {
  super.viewDidLoad()
  // Do any additional setup after loading the view, typically from a nib.

  // (1) ラベルに文字をいれる
  outputLabel.text = "Hello Swift!"
}
```
追加

Labelへ文字を設定

本書では上の図のように、プログラムコードの追加を示していきます。コードを追加する場所がわかるように、追加する周辺の情報も掲載しています。

今回は、「override func viewDidLoad() {」から「}」の中に、「outputLabel.text = "Hello Swift!"」と入力してください。

はじめてのプログラムコードを書きました！

これは、Labelに「Hello Swift!」という文字を表示させるコードです。

Tips
「override func viewDidLoad(){ ... }」の中に書いたコードは、アプリ起動時に1度だけ動きます。

4：シミュレータを起動してみよう

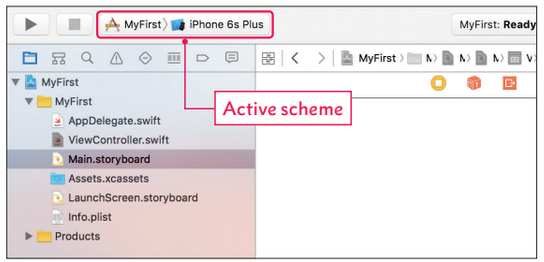
Xcodeには**「シミュレータ」**というツールがあります。

シミュレータを使うと、XcodeからiPhoneやiPadの画面が起動して、動作確認が行えます。実際のデバイスが手元になくてもシミュレータで確認できるので大変便利です。

実際にシミュレータで「Hello Swift!」の表示を確認しながら、シミュレータの起動方法と操作方法を学習していきましょう。

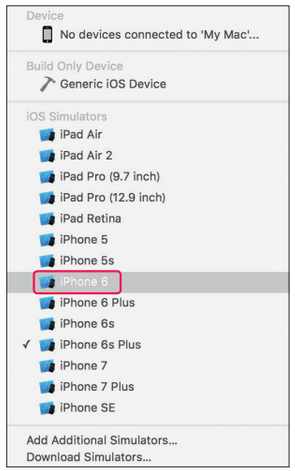
4-1 デバイスの選択と種類

Xcodeの左上に[Active scheme]（アクティブ スキーム）があります。ここでは、シミュレータで起動できるiOSデバイスを指定できます。クリックすると、選択できるデバイス（機種）が一覧表示されます。



デバイスの選択

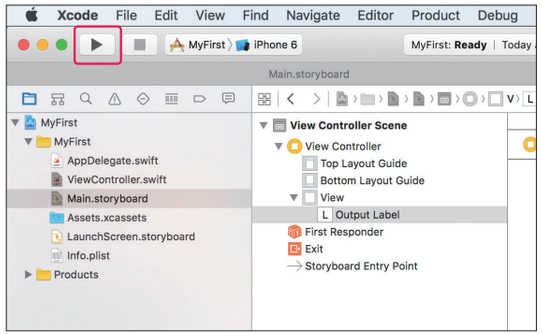
いろいろとデバイスが選択できますが、今回は「iPhone 6」を選択してみます。お好きなデバイスを選択してみてください。



「iPhone 6」を選択

4-2 シミュレータの起動

Xcodeの左上に配置されている ▶ [Run] をクリックします。



[Run] をクリック

Memo

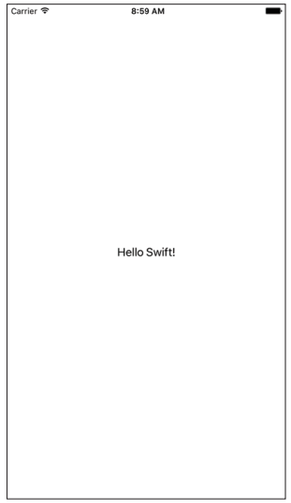
シミュレータの起動には、時間がかかることがあります。アプリが起動すると自動的にiPhoneの画面が表示されますので、気長に待ちましょう。

Memo

シミュレータはキーボードの「command ⌘」キーを押しながら「R」キーを押すことでも起動できます。操作が行えるキーの組み合わせを、「ショートカットキー」といいます。

シミュレータが起動されます。

「Hello Swift!」と表示されていれば、はじめてのアプリ開発は成功です！



シミュレータ起動

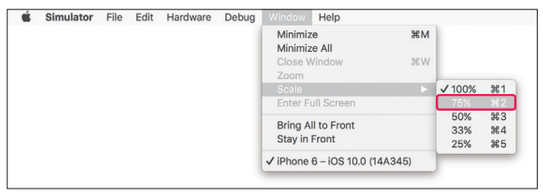
Memo

今回のように、アプリの開発はXcodeで画面を作り、プログラムを書いて、シミュレータで動作確認を行なう、という手順を繰り返していきます。

4-3 シミュレータの表示サイズの拡大縮小

シミュレータの表示サイズを変更できます。

シミュレータのデバイスを前面に起動した状態で、[Window] → [Scale] → [75%] とすると、75%に縮小されて表示されます。



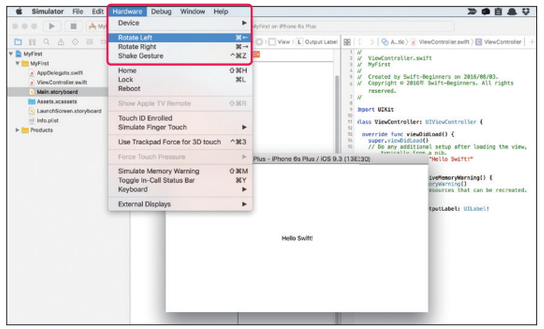
表示サイズの指定

Memo

シミュレータを選択している状態で「command ⌘」キー と一緒に数字の「1」を押してみてください。
画面表示が、100%の拡大率に切り替わります。
「1」が100%、「2」が75%、「3」が50%、「4」が33%、「5」が25%

4-4 シミュレータの向き(横画面、縦画面)を変える

シミュレータを選択している状態で [Hardware] → [Rotate Left] または [Rotate Right] を選択することで、縦画面・横画面の切り替えができます。



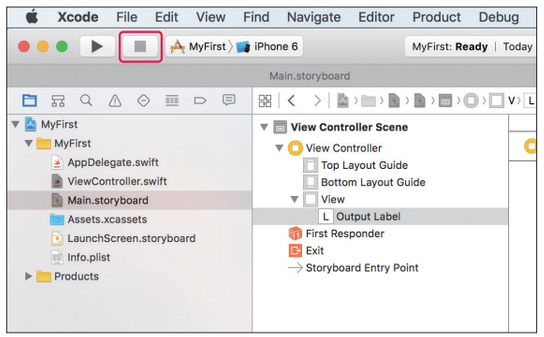
シミュレータの向き

Memo

ショートカットは、「command ⌘」キーと「矢印」キー
左回転は、「command ⌘」キーと「←」キー
右回転は、「command ⌘」キーと「→」キー

4-5 シミュレータの終了方法

[Stop] ボタンをクリックして、シミュレータを停止します。



[Stop] ボタンをクリック

Memo

シミュレータを終了するためのショートカットは、「command⌘」キー＋「Q」キーです。

5：実機転送を行い、iPhoneで確認してみよう

制作途中のアプリならシミュレータで確認することは十分可能です。ですが、実機で確認することでより完成度の高いアプリになります。

実機であれば、実際の指での感触や滑らかさ、アプリの動くスピードなども確認することができます。とくにカメラの撮影は実機で実施したいところです。

iPhoneにアプリを転送して確認する方法を学んでいきましょう。

5-1 MacとiPhoneを接続します

実機（iPhone本体）にアプリを転送するには、MacとiPhoneをLightningケーブルで接続する必要があります。実機転送を行うときのLightningケーブルは正規品が望ましいです。

Lightningケーブル（正規品）は、Appleのサイト（http://www.apple.com/jp/）から購入できます。

MacとiPhone

5-2 Apple IDをXcodeに登録

Xcodeに登録しているApple IDとiPhoneで設定されているApple IDが同じであれば、アプリを転送することができます。

XcodeにApple IDを登録していきます。

メニューから［Xcode］→［Preferences...］を選択します。

Memo

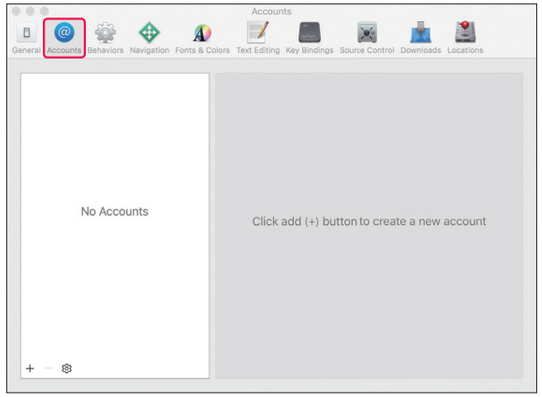
［Preferences］は、Xcodeの環境設定が行える画面です。重要ですので、以降のレッスンで詳しく説明します。

［Preferences...］を選択

環境設定の画面が表示されますので、［Accounts］タブを選択します。

Memo

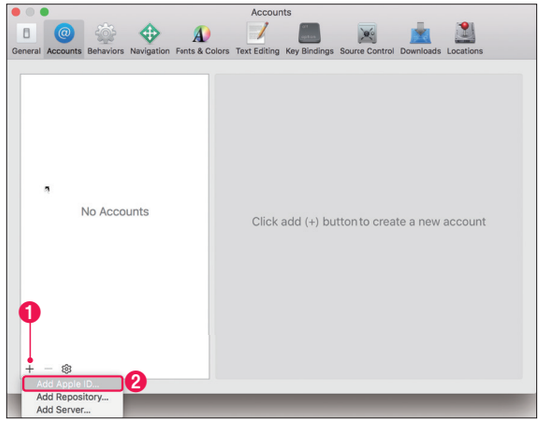
［Accounts］がApple IDを管理しているタブです。最初ですので、「No Accounts」と表示されています。



［Accounts］タブ

Apple IDでログインします。

❶「+」（プラス）マークをクリックします。
❷［Add Apple ID］を選択します。

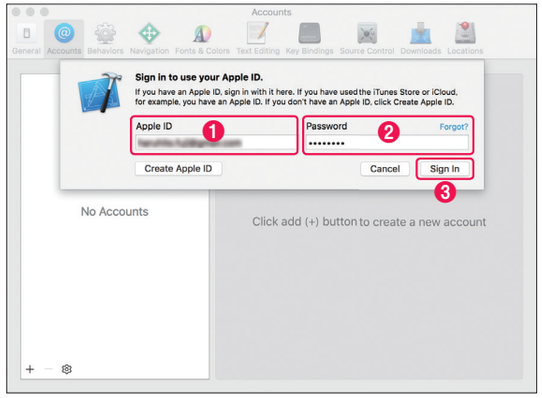


Add Apple IDを選択

❶［Apple ID］（メールアドレス）を入力します。
❷［Password］を入力します。
❸［Sign In］をクリックします。

Memo

Apple IDのサインインにはネット環境が必要です。インターネットに接続されているか確認してください。

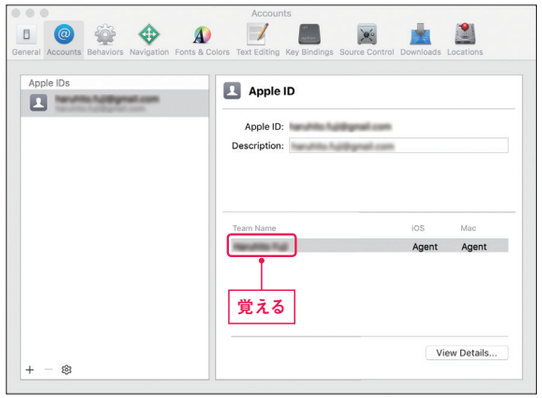


Apple IDの入力

Apple IDのログインが完了すると、Xcodeに登録されます。

あとで**Team Name**を選択するので、登録名を覚えておいてください。

左上の赤いボタンをクリックして［Preferences］の画面を閉じます。



Team Name

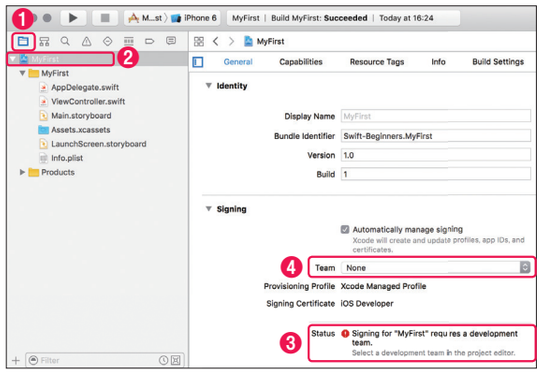
5-3 Team（チーム）を設定します

❶［Project Navigator］を選択して、

❷「MyFirst」を選択します。

❸エラーが発生していのが確認できます。

エラーの内容は、［Team］（チーム）を設定することで解消ができます。



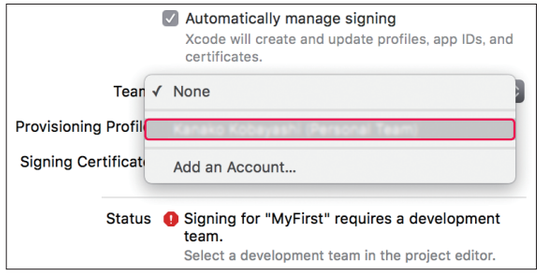
プロジェクトのエラー確認

シミュレータの起動時には問題ありませんが、実機転送を行う場合には、［Team］が設定されていないとエラーとなり、転送できません。

❹［Team］を確認するとなにも選択されていないことを意味する「None」が選択されています。

こちらを設定する必要があるので、リストボックスをクリックします。

先ほど登録したアカウントが表示されているので、選択してください。



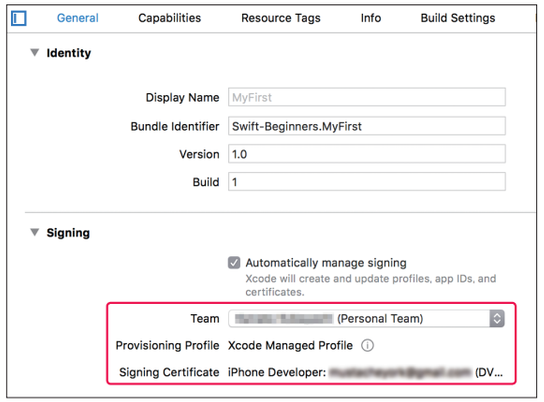
Apple ID選択画面

［Team］を選択すると、エラーが解消されます。

Memo

［Team］を適正に選択しても、［Status］に「Failed to create provisioning profile.」とエラーが表示される場合は、［Bundle Identifier］の設定に問題があります。
この場合は、次のWebページを参考にしてください。

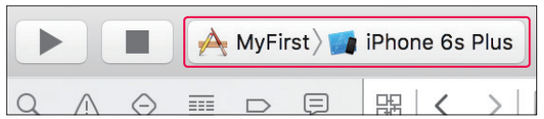
本書サポートサイト：シミュレータ
https://swiftbg.github.io/swiftbook/simulator



［Team］の設定

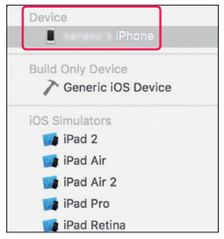
5-4 iPhoneをXcodeから起動します

iPhoneをMacに接続した状態で、デバイスを選択します。枠の中をクリックします。



デバイスの選択

［Device］の欄に、先ほど接続したiPhoneが表示されているので、クリックして選択します。



実機の選択

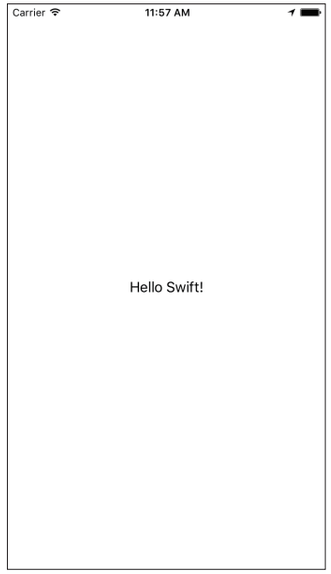
▶ マークをクリックすると、実機転送が行われます。

実機転送

iPhoneにアプリが転送されると、アプリが起動され、「Hello Swift!」と表示されます。

Memo

実機転送に慣れないときは、Xcodeにいろいろなメッセージが表示されると思います。その場合は、次のTipsを参考に解決してください。



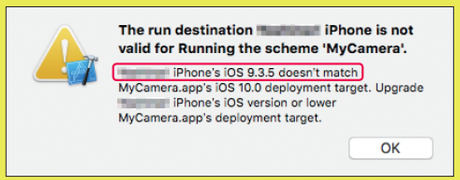
アプリの完成イメージ

Tips

実機転送時にXcodeにメッセージが表示された場合の対応について

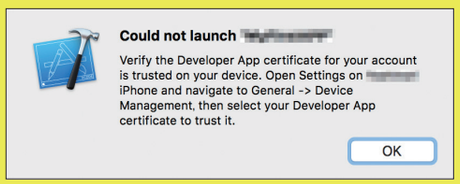
●Xcodeで「The run destination ○○○ iPhone is not valid ...」が表示されたとき

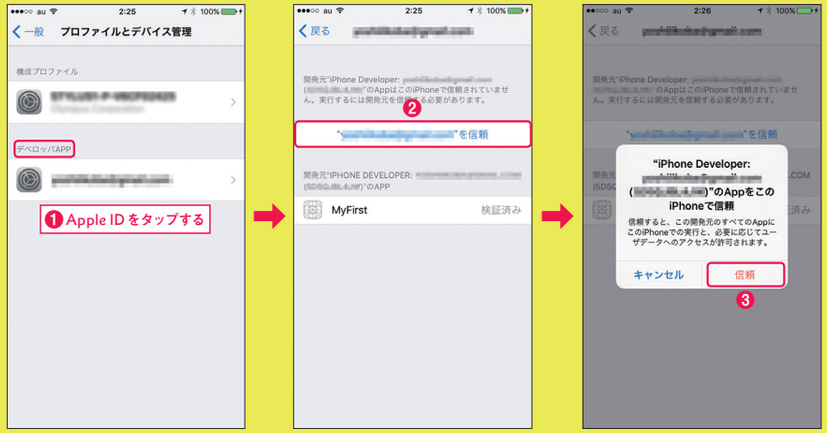
メッセージの本文に「○○○ iPhone's iOS X.X.X doesn't match ...」（Xは数字）の記載があるときは、Xcodeで生成されたアプリの対応バージョンとiPhoneのiOSバージョンが合っていません。iPhoneのiOSバージョンを最新にしてから実機転送を行ってください。



●Xcodeで「Could not launch ○○○」と表示されたとき

初回の実機転送では、セキュリティのエラーメッセージが表示されます。この場合、iPhone（実機）で「信頼」するための設定を行う必要があります。
iPhoneで［設定］をタップして開き、［一般］→［プロファイルとデバイス管理］を開きます。

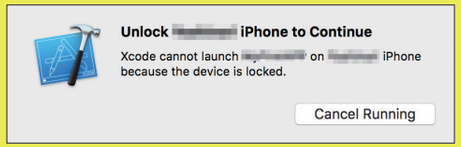




［プロファイルとデバイス管理］画面で、［デベロッパAPP］欄に表示されている、❶Apple IDをタップします。
詳細画面が開きますので、❷「" (Apple ID) "を信頼」をタップします。
信頼するかどうかを問われますので、❸「信頼」をタップします。
この操作で、iPhone側で開発者（Apple ID）が信頼されました。
再度実機転送を行うと、アプリが転送されて実行されるようになります。

●Xcodeで「Unlock ○○○ iPhone to Continue」と表示されたとき

iPhoneがロックされていると転送できないため、メッセージが表示されます。
メッセージが表示されている状態でiPhoneのロック解除を行うとメッセージが消え、引き続きアプリの転送が行われます。



シミュレータに関するトラブルやエラーは、まだ他にもあります。その他の情報に関しては、下記のページにまとめています。
本書のサンプルを実際に開発していくなかで、シミュレータに関する困ったことがあればこちらを参照してください。

本書サポートサイト：シミュレータ
https://swiftbg.github.io/swiftbook/simulator