



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

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ものとさせていただきます。従いまして、〇〇ページの〇〇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

- 本書内に記載されている会社名、商品名、製品名などは一般に各社の登録商標または商標です。本書中では®、™マークは明記しておりません。
- 本書の出版にあたっては正確な記述に努めました。が、本書の内容に基づく運用結果について、著者およびSBクリエイティブ株式会社は一切の責任を負いかねます。ご了承ください。

©2016 Haruhito Fuji, Yui Tokuhito, Kanako Kobayashi, Yoshinori Kobayashi

本書の内容は著作権法上の保護を受けています。著作権者・出版権者の文書による許諾を得ずに、本書の一部または全部を無断で複写・複製・転載することは禁じられております。

はじめに

この書籍は、「iOS アプリを作りたい、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。初心者が最初の一步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

はじめに iii

- 0 この本の読み方と使い方 x
- 1 ご利用の前に必ずお読みください xiii

1 目目

Lesson

1

はじめてのアプリを
開発する前に知っておこう

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。

- 1 プログラミングを体験から
学んでいこう 2

1 この本の使い方と前提知識 2
- 2 あらかじめ挫折しそうなポイントを
押さえておこう 3

1 学習ポイントを押さえよう 3
- 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～ 5

1 WWDC2016での
公開情報をもとに考察してみよう 5
- 4 Swift（スウィフト）を知ろう 7

1 Swiftの特徴を押さえよう 7

Lesson

2

アプリ開発の環境を整えて、
Xcodeの使い方を学ぼう

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。
アプリ開発するために必要な 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。

- 1 iOSアプリを開発するために
必要な準備をしよう 12

1 アプリ開発に必要な
3つのものを準備しよう 12
- 2 Apple IDを取得しよう 14

1 Apple IDをすでに取得されている方 14

2 Apple IDをまだ取得されていない方 15
- 3 Xcodeをインストールしよう 17

1 Xcodeをダウンロードしよう 17
- 4 Xcodeを起動して、
プロジェクトを作成しよう 19

1 Xcodeを起動しよう 19

2 プロジェクトを作成しよう 20

3 Xcode画面のナビゲーションを
理解しよう 25
- 5 Xcodeをより使いやすくなるための
設定をしよう 31

1 Xcodeの環境を設定しよう 31

6 「Hello Swift!」と表示してみよう 35

- 1 完成をイメージしよう 35
- 2 Labelを配置して、
AutoLayoutを使ってみよう 36

3 プログラムを書いてみよう 40

- 4 シミュレータを起動してみよう 43
- 5 実機転送を行い、
iPhoneで確認してみよう 46

Lesson

3

じゃんけんアプリを作ろう
～Swiftの基本を学ぶ～

じゃんけんアプリを作りながら、アプリ開発の基本を学んでいきます。
具体的には、プロジェクトの作成方法、画面へのパーツ配置、プログラムとの連結、
Swiftの基本文法を学びます。そして、最後には実際に iPhoneで
じゃんけんアプリを動かしてみましょう！

1 プロジェクトを作成しよう 54

- 1 完成をイメージしよう 54
- 2 プロジェクトを作成しよう 55

2 アプリに必要なパーツの
配置と設定をしてみよう 57

- 1 Storyboardにパーツを配置してみよう 57
- 2 パーツを装飾してみよう 60
- 3 Image Viewパーツで使う
画像ファイルを設定しよう 67

3 各パーツの表示位置、幅や高さを
設定しよう 69

- 1 AutoLayout（オートレイアウト）の
概要を理解しよう 69
- 2 作業しやすいように画面を整えよう 69
- 3 Buttonの位置と幅・高さを設定しよう 70
- 4 Labelの位置と幅・高さを設定しよう 76
- 5 Image Viewの位置と幅・高さを
設定しよう 78

4 画面のパーツとプログラムを
関連付けしてみよう 80

- 1 コードを書きやすいように
Xcode画面を整理しよう 80
- 2 Image Viewをコードと関連付けしよう 82
- 3 Labelをコードと関連付けしよう 84
- 4 Buttonをコードと関連付けしよう 85

5 プログラムを書いて
じゃんけんアプリを動かそう 90

- 1 プログラムを書く前に、
大きく作り方を把握しよう 90

6 カスタマイズ編①
～起動画面（LaunchScreen）を
設定しよう～ 105

- 1 完成をイメージしよう 105
- 2 起動画面に使うロゴ画像を設定しよう 106

7 カスタマイズ編②
～アイコンを設定しよう～ 111

- 1 完成をイメージしよう 111
- 2 アイコンのサイズや
使用用途を確認してみよう 112
- 3 アイコンの作成と設定をしよう 113

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！

1	プロジェクトを作成しよう	118	3	「AVFoundation」を読み込もう	138
1	完成をイメージしよう	118	4	シンバルとプログラムをつなごう	138
2	プロジェクトを作成しよう	118	5	メソッドをコーディングしよう	139
2	楽器を配置しよう	119	6	ギターとプログラムをつなごう	145
1	画像ファイルを取り込んで配置しよう	119	7	「Play」「Stop」とプログラムをつなごう	146
3	タップで、音を鳴らしてみよう	136	4	カスタマイズ編 ～リファクタリングで コードをすっきり！～	150
1	それぞれの音源を鳴らすように 実装しよう	136	1	リファクタリング (refactoring) を 押さえよう	150
2	音を扱う準備をしよう	136			

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。

1	プロジェクトを作成しよう	156	3	マップ検索アプリを作っていこう	165
1	完成をイメージしよう	156	1	Text Fieldを使ってみよう	165
2	プロジェクトを作成しよう	157	2	キーワードから 緯度経度を取得してみよう	170
2	画面を作成しよう	158	3	delegateやクロージャについて 理解しよう	175
1	使用するUIパーツを配置しよう	158	4	カスタマイズ編 ～マップの表示方法を 切り替えてみよう～	178
2	AutoLayoutで、レイアウトを整えよう	160	1	完成をイメージしよう	178
3	関連付けをしよう	162	2	地図種別の「切替」ボタンを配置しよう	179
			3	地図種別切り替え処理を実装しよう	183

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。

1	プロジェクトを作成しよう	190	4	画面を更新するメソッドを作成しよう	229
1	完成をイメージしよう	190	5	経過時間を処理する メソッドを作成しよう	232
2	プロジェクトを作成しよう	191	6	カウントダウンの処理を実装しよう	233
2	Storyboardでレイアウトを作ろう	192	7	タイマーを停止する処理を実装しよう	236
1	[Main.storyboard] を Navigation Controllerに変更しよう	192	8	秒数の設定画面に、画面遷移をしよう	236
2	Navigation Controllerの 仕組みを知ろう	194	9	秒数設定画面から戻ってきたら、 設定した秒数で画面を更新しよう	238
3	タイマー画面のUIパーツを配置しよう	195	10	シミュレータでカウントダウンを 確認しよう	239
4	タイマー画面の AutoLayoutを 指定しよう	198	4	設定画面を作ろう	241
5	設定画面を追加しよう	208	1	PickerViewを利用できるようにしよう	241
6	画面遷移を行うSegueを追加しよう	209	2	変数を宣言しよう	242
7	設定画面のUIパーツを配置しよう	211	3	設定画面起動時に、PickerViewに 保存している秒数をセットしよう	244
8	設定画面の各パーツに AutoLayoutを設定しよう	214	4	for文とは？	245
9	設定画面の SettingViewControllerを 追加しよう	216	5	UIPickerViewの DataSourceの delegateメソッドを作成しよう	246
10	[Assistant editor] で確認しよう	219	6	UIPickerViewの内容を表示する delegateメソッドを作成しよう	247
11	タイマー画面 (View Controller) を 関連付けよう	221	7	画面で秒数を選択したときの delegateメソッドを作成しよう	248
12	設定画面 (SettingViewController) を 関連付けよう	224	8	「決定」ボタンがタップされたら、 タイマー画面に遷移しよう	248
3	タイマー画面を作ろう	226	5	カスタマイズしてみよう	253
1	コードを書きやすいように準備しよう	226	1	完成をイメージしよう	253
2	変数の宣言を追加しよう	227	2	UIAlertControllerを追加しよう	254
3	起動時に初期値を設定しよう	228			

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。

1 プロジェクトを作成しよう 258

- 1 完成をイメージしよう 258
- 2 プロジェクトを作成しよう 259

2 画面を作ろう 260

- 1 使用する UIパーツを配置しよう260
- 2 AutoLayoutで、レイアウトを整えよう263
- 3 UIパーツの詳細を設定しよう265
- 4 UIパーツをプログラムと関連付けしよう...267

3 カメラの起動と SNSシェアを作ろう 271

- 1 カメラの使用をユーザーに許可してもらおう271
- 2 カメラが使用できる機種、できない機種を判定しよう273
- 3 カメラを起動するコードを追加しよう275
- 4 カメラで撮影した写真を表示させよう277
- 5 SNSシェア機能を追加しよう278

4 カスタマイズ編
～フォトライブラリーから写真を
取り込んでみよう～ 283

- 1 完成をイメージしよう 283
- 2 フォトライブラリーの使用をユーザーに許可してもらおう284
- 3 カスタマイズしてみよう 284

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。

1 画面を修正しよう 292

- 1 完成をイメージしよう 292
- 2 「SNSに投稿する」ボタンを削除しよう 293

2 画面を追加しよう 296

- 1 新しい ViewControllerを追加しよう296
- 2 ViewControllerとコードを結びつけよう297
- 3 いままでの ViewControllerから画面遷移できるようにしよう300
- 4 使用する UIパーツを配置しよう302

3 エフェクト画面を作っていこう 310

- 1 画面遷移をしてみよう310
- 2 画面遷移して画像を表示してみよう314
- 3 画面を閉じてみよう315
- 4 画像をエフェクトしてみよう317
- 5 画像をシェアしてみよう321

4 カスタマイズ編
～エフェクトの種類を増やしてみよう～ 323

- 1 完成をイメージしよう323
- 2 カスタマイズしてみよう324
- 3 フィルタ名の効果を確認しよう327

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。

1 プロジェクトを作成しよう 330

- 1 完成をイメージしよう 330
- 2 プロジェクトを作成しよう 330

2 Web APIとJSONについて学ぼう 331

- 1 Web APIの基本的な仕組みを学ぼう331
- 2 JSONとXMLについて学ぼう332
- 3 ブラウザで Web APIを使ってデータを取得してみよう334

3 Search Bar, Table Viewを配置しよう 338

- 1 Search Barを配置しよう338
- 2 Table Viewを配置しよう340
- 3 各パーツに AutoLayoutを設定しよう342
- 4 Table View Cellの [Style] と [Identifier] を設定344

4 キーワードを入力して
お菓子データを取得しよう 345

- 1 画面のパーツとプログラムの関連付けをしよう345
- 2 Search Barで入力されたキーワードをデバッグエリアに出力しよう347
- 3 Web APIのリクエストURLを組み立てよう351
- 4 リクエストを生成して、JSONを取得しよう355

5 取得したお菓子データを
Table Viewで一覧表示してみよう 362

- 1 お菓子データをタプルに格納してみよう362
- 2 お菓子データを Table Viewで一覧表示してみよう367

6 カスタマイズ編
～お菓子の一覧をタップして
Webページを表示してみよう～ 373

- 1 完成イメージを確認しよう373
- 2 SFViewControllerを使ってWebページを表示しよう374

索引 379

Swiftビギナーズ倶楽部について 383

謝辞 384

執筆陣プロフィール 385

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

1
Lesson
4

1 プロジェクトを作成しよう

このレッスンで学ぶこと

楽器アプリは、4つのパーツを配置して機能を追加していきます。水平・垂直を起点としたAutoLayoutを設定してきますので、完成イメージを確認します。

できるようになること

楽器アプリの画面レイアウトと、実装する機能を確認します。プロジェクトを作成し、開発の準備をします。

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。

③ Tips (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。

Tips letとvarの違いを知ろう

音楽ファイルの場所を保持しておく「cymbalPath」とシンバル用のプレイヤーインスタンスを保持しておく「cymbalPlayer」の前に、「let」と「var」を指定しています。
じゃんけんアプリでは「var」を指定して変数を作成していましたが、「let」は初めて出てきました。
これらはどう違うのでしょうか？
「let」は一度値を設定したら変更できないという指定で、「定数」と呼ばれます。
対して「var」は値の変更が可能で、「変数」と呼ばれます。
定数を宣言した後に値を書き換えようとすると、Xcodeはエラーを発生させて定数は変更できないことを教えてくれます。
どうして、Swiftは「定数」と「変数」をきちんと使い分けるように提案するのでしょうか？
ここでも、Swiftが安全性を重視している言語だということが読み取れます。
Swiftは、プログラムが安全に実行できるようにさまざまな工夫がなされている言語です。
プログラムの中で、変わってしまったらいけない値を明示的に宣言して、「もし変更されていたらもう一度よく考えてみてね」と、Xcodeがプログラマーに教えてくれます。

Column

Access Control (アクセス修飾子)とは？

メソッドや変数は、呼び出す(アクセスできる)範囲を指定できます。
なぜそのような制限が必要なのでしょう。どのソースからもメソッドや変数を利用できると、想定していないところで利用されたりする危険性があります。呼び出すことができる範囲を制限することで、必要最低限の範囲でアクセスを許可できます。
このような範囲のことを「スコープ」と言います。
Swift以外にも、このような指定をする言語もあります。

Swiftのアクセス修飾子には、次の3種類があります。

- public
誰からもアクセス可能になります。
他のファイルや、他のモジュールからもアクセス可能になります。
- internal
同じモジュール内であればアクセス可能です。
アクセス修飾子を付けないと、デフォルトでこの指定になっています。
- private
ファイル内でのみ、アクセス可能になります。

0-6 さらにステップアップ！ ～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イ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ますが、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 本書に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

iOS 10.0 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！

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。



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

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の方が効率的です。

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考えられます。

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

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。

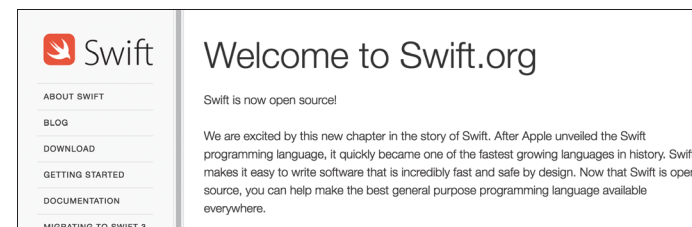
できるようになること

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/>

Lesson 1

はじめてのアプリを開発する前に知っておこう

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

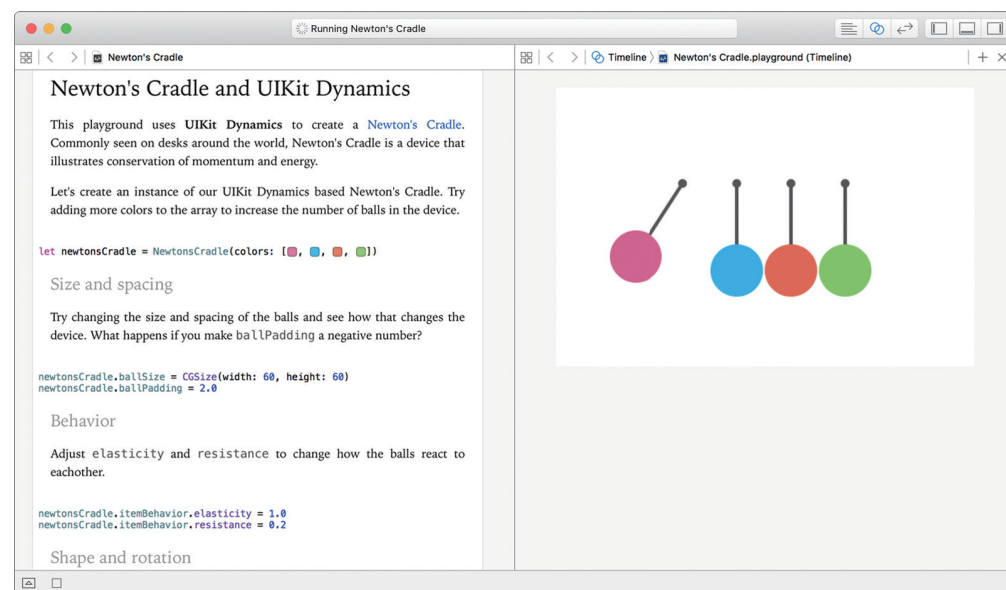
モダン（modern）とは「現代風」という意味です。Swiftでは、他のプログラミングでも採用されている新しい機能を、積極的に取り入れています。

●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なく、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。

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。

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、事前に作成しておく必要があります。

1-3 Xcode (エクスコード)

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。

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。



Mac



App StoreとiTunes



Xcode