親切すぎる iPhone アプリ開発の本

Xcode 8（Swift 3）への対応

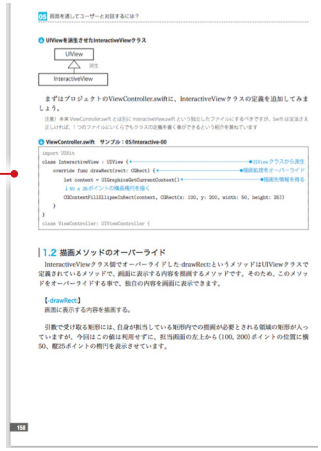
2016.09.19

目的

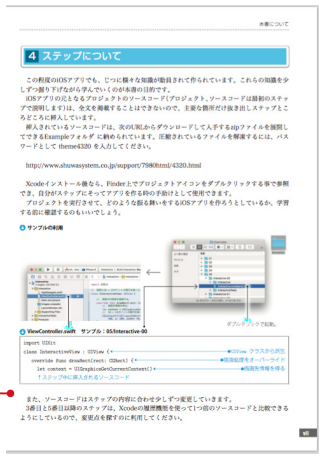
Xcode 8 で引き続き「親切すぎる iPhone アプリ開発の本」を利用する方法を説明する。

Xcode 8 では、Swift 言語が 2 から 3 にバージョンアップされました。
そのため、本書の各ステップで紹介されている Swift 用プログラムコードを、そのまま書き写して Xcode 上で実践することはできなくなりました。
この本ではその対処法を説明します。

158 ページ



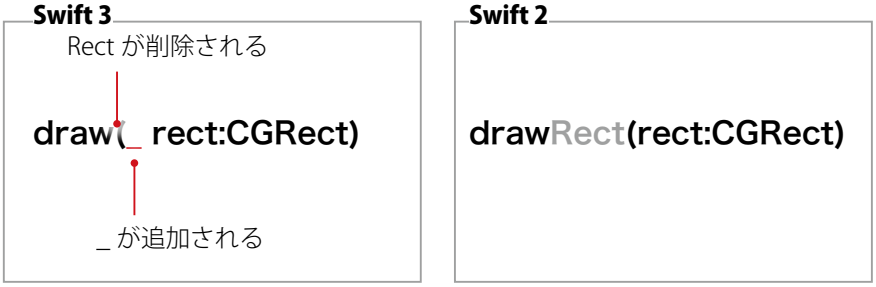
viii ページ



サンプルプロジェクト群「Examples」の利用

Swift 言語のバージョンアップにより、例えば 158 ページで紹介した -drawRect: は以下のように変更されます。

例）Swift 3、Swift 2での-drawRect:のプログラムコードの違い




```
class InteractiveView: UIView {

    //  画面の内容部を描画する。
    override func draw(_ rect: CGRect) {
        //  描画先情報を得る。
        let context = UIGraphicsGetCurrentContext();
```

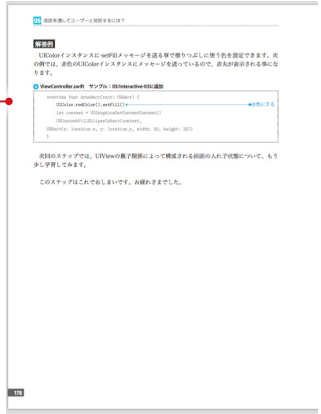
```
class InteractiveView: UIView {

    //  画面の内容部を描画する。
    override func drawRect(rect: CGRect) {
        //  描画先情報を得る。
        let context = UIGraphicsGetCurrentCo
```

ただし、記述さえ Swift 3 用に変更すれば、これまでどおりアプリを作ることができます。

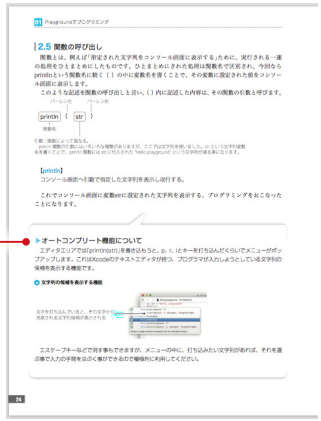
サポートページにあるサンプルプロジェクト群「Examples」は Xcode 8 に対応済みなので、これをダウンロードし viii ページで紹介してるソースコードの比較方法を利用して Swift 2 用のコードと Swift 3 用のコードの違いを確認しつつ読み進めていってください。

170 ページ



243 ページ

24 ページ



Examples のサンプルソースを加工する場合

170 ページのように Examples のソースを加工する場合は、いったん本文中で例示された記述をそのまま入力し、そこから 243 ページで紹介しているエラーマークの特典を活用し Swift 3 に対応させてください。

例)エラーマークの特典を活用

本文通りに書き込むと、エラーマークが表示される

```
// 画面の内容部を描画する。
override func draw(_ rect: CGRect) {
    UIColor.redColor().setFill()
    // 描画先情報を得る。
```

エラーマークをクリックすると、修正候補が表示される

```
// 画面の内容部を描画する。
override func draw(_ rect: CGRect) {
    UIColor.redColor.setFill()                    Cannot call value of non-function type 'UIColor'
    // 描画先情報を得る。
    Cannot call value of non-function type 'UIColor'
    Fix-it  Delete "()"
    ext();
    f.location.x, y: self.location.y, width: 50, height: 25));
}
```

表示された修正候補を選ぶと、選んだ修正候補の内容にソースが修正される
この時、もし追加の修正が必要な場合は引き続きエラーマークが表示される

```
// 画面の内容部を描画する。
override func draw(_ rect: CGRect) {
    UIColor.redColor.setFill()
    // 描画先情報を得る。
```

再度エラーマークをクリックすると、修正候補が表示される

```
// 画面の内容部を描画する。
override func draw(_ rect: CGRect) {
    UIColor.red.setFill()                         'redColor' has been renamed to 'red'
    // 描画先情報を得る。
    'redColor' has been renamed to 'red'
    Fix-it  Replace "redColor" with "red"
    tCurrentContext();
    CGRect(x: self.location.x, y: self.location.y, width: 50, height: 25));
}
```

修正が完了するとエラーマークは無くなる

```
// 画面の内容部を描画する。
override func draw(_ rect: CGRect) {
    UIColor.red.setFill()
    // 描画先情報を得る。
```

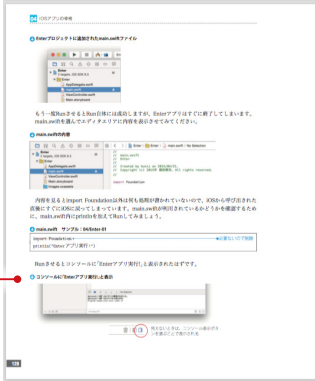
24 ページで紹介したオートコンプリート機能の段階で Swift 3 の書き方が推測できる人は、そのままそれを選んでもらってもかまいません。

多くの場合 Swift 3 への変更は、先頭文字を大文字から小文字にしたり、文字列の一部を省略したりといった Swift 2 のコードから連想しやすいものとなっています。

その他に、Edit → Convert → ToCurrent Swift Syntax…メニューで一気に変換させる方法もあります。

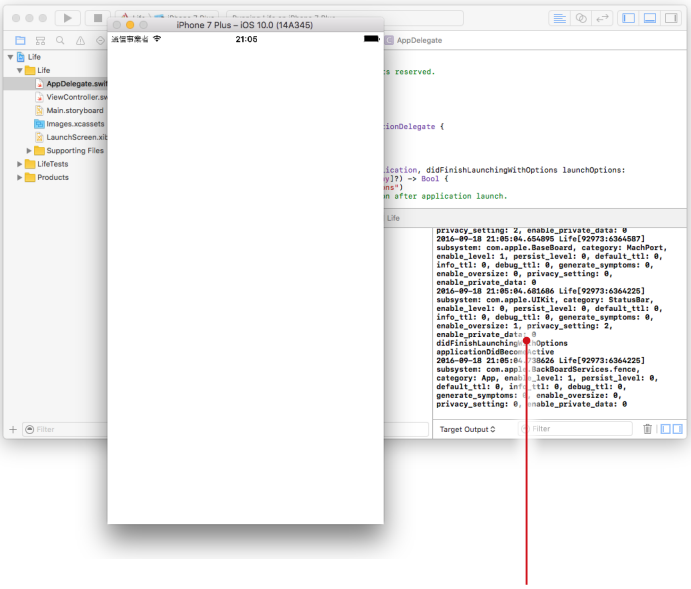
デバッグ出力を抑える

128 ページ



修正されるかもしれませんが、現時点の Xcode 8（8A218a 版）は iOS アプリ実行時にコンソール（128 ページ参照）に非常にたくさんのメッセージが出力されます。

例）コンソール出力

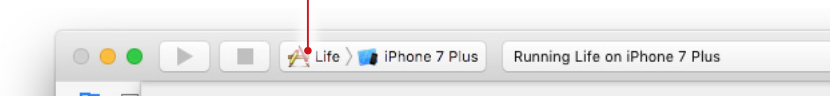


たくさんのメッセージが出力される

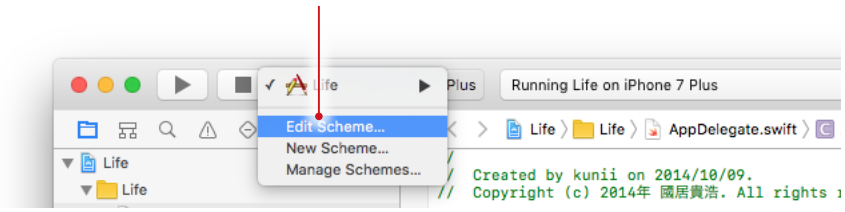
このせいで自分が出力させたメッセージが読みにくい場合は、次の方法で出力を抑えることができます。

例）コンソールメッセージ出力を抑えるためのScheme設定

Set the active scheme をクリック

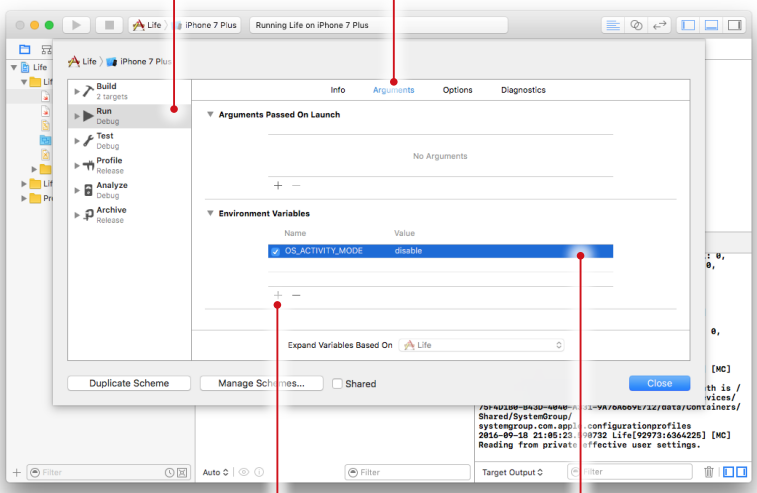


ポップアップしたメニューから Edit Scheme を選択



❶表示された画面で Run を選択

❷ Arguments を選択



❸ Environment Variables グループの + ボタンをクリックする

❹新しい項目が追加されるので、Name に OS_ACTIVITY_MODE、Value に disable を設定する

サンプル 21/Life-00 が、この設定になっているので参考にしてください。

Swift 3 への変更点について

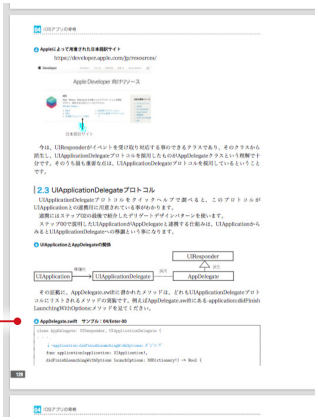
各変更点をページごとに解説していこうとすると、丸々一冊の新刊になってしまう関係から「Xcode 7(Swift 2) への対応」でおこなったような、ページごとの変更点解説は割愛させて頂きます。

代わりに全体に関係する大きな変更点を以下に示します。Examples で変更点を調べるときの参考にしてください。

なお、本書で初めて Swift にふれる方は、先にサポートページに置かれている「Xcode 7(Swift 2) への対応」を読んで Swift 1.2 から Swift 2 への変更を把握しておくことをお勧めします。また Swift 3 では「Xcode 7(Swift 2) への対応」で説明した変更から、さらに変更されたものもある点に注意してください。

変更点

120 ページ



- メソッド・関数とも、第 1 引数か第 2 引数以降かの区別なく、ラベルを省略すると自動的に仮引数名がラベルに使用されるようになりました。ラベルを利用したくない場合は、ラベル名の代わりに _ を書く必要があります。

AppDelegate.swift サンプル :04/Enter-00

iOS 提供のプロトコルやクラスのメソッドは
第 1 引数にラベルをつけない。そのため第 1 引数の
仮引数名の前に _ が付けられることになった

```
func application(_ application: UIApplication, didFinishLaunching・・・
```

- ++、-- 演算子は廃止されました。
- クラスオブジェクトを指定する場合、クラス名に追加して .self が必要になりました。

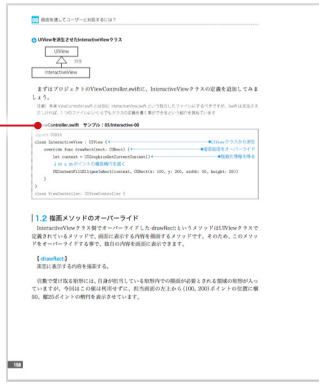
129 ページ

main.swift サンプル :04/Enter-03

クラスオブジェクトを指定する場合、クラス名に追加して
.self が必要になった

```
UIApplicationMain(・・・, nil, NSStringFromClass(AppDelegate.self))
```

180 ページ



・ CGContextFillEllipseInRect といった CGContext に関係する関数は CGContext のメソッドとなりました。

ViewController.swift サンプル :05/Interactive-00

```
let context = UIGraphicsGetCurrentContext();
// 50 x 25 ポイントの楕円を描く。
context?.fillEllipse(in: CGRect(x: 100, y: 200, width: 50, height: 25))
```

CGContext のメソッドとなった

224 ページ

・ ターゲット・アクションなどのメソッド登録に、メソッド名を文字列を指定する方法は推奨されなくなりました。代わりに #selector を使います。

ViewController.swift サンプル :09/DelegateButton-08

書き方がわからない時は、最初に文字列で指定しておき
243 ページで紹介しているエラーマーク（ここでは注意
マークとなる）の特典を活用すればよい

```
bt.addTarget(self, action: #selector(ViewController:touchDown(_:)), ・・・
```

292 ページ

・ 291 ページで説明した CALayerDelegate プロトコルが公開されたので、CALayer の delegate に指定されるオブジェクトのクラスはこれを採用する必要があります。

ViewController.swift サンプル :13/CustomLayer-01

```
class ViewController: UIViewController, CALayerDelegate {
```

・ NSURL、NSData、NSDate、NSIndexPath 型は残されていますが、Swift では基本的にそれぞれ URL、Data、Date、IndexPath 型を使うようになります。

ちなみに、サンプルソースでは Swift 3 への自動変換により IndexPath への扱いは次のようになっています。

662 ページ

ViewController.swift サンプル :34/Table-01

```
func tableView(_ tableView: UITableView,
        cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    ・・・
// 内容を設定。
cell.textLabel?.text = " 項目 \((indexPath as NSIndexPath).section)・・・
```

これを NSIndexPath の時と同じように書いても問題ありません。

```
// 内容を設定。
cell.textLabel?.text = " 項目 \(indexPath.section)・・・
```

- NSString と String 間等の暗黙の変換は廃止されました。as を使った明示が必要になります。
- NSDictionary の Swift での表現は [NSObject : AnyObject] から [AnyHashable : Any] となります。
- private キーワードの意味が変更されました。Swift 2 での private キーワードと同じ意味を持つ Swift 3 のキーワードは「fileprivate」です。

以上が Xcode 8（Swift 3）への対応の説明です。
お疲れさまでした。