モバイル(iPhone)
アプリケーションの作り方

Programing Language Swift for iOS Applications

株式会社あいはら 研究開発チーム
山田 泰司 <taiji@aihara.co.jp>
Apple ID: taiji_yamada@mac.com

リソースの配布元 - http://www.aihara.co.jp/~taiji/lecture-2019/#Xcode+iOS+Swift

私について、
簡単な自己紹介

1. 1995年、東京電機大学 大学院 修士課程修了、(株)あいはら入社
2. 1997年、Unix向けカオス時系列解析システム「ChaosTimes」を開発・販売
3. 2002年、Windows向けカオス時系列解析システム「Sunday ChaosTimes」を開発・販売
4. 2003年、埼玉大学 工学部 情報工学科にて、**ページ記述言語PostScriptとgs-cjkプロジェクトに関する特別講義**
5. 2006年〜2016年、埼玉大学 同上にて**「情報システム工学 I (システム開発工学)」の講義と演習**
6. 2008年〜2016年、埼玉大学 同上にて**「情報システム工学 II (システム工学概論)」の講義と演習**
7. 2010年、埼玉大学 同上にて、**Objective-CによるiPhoneアプリ開発の特別講義**
8. 2017〜2018年、埼玉大学 同上にて、**SwiftよるiPhoneアプリ開発の講義と演習**
9. 2018年、東京理科大学 工学部 情報工学科にて、**SwiftよるiPhoneアプリ開発の講義と演習**
10. 通して、非線形工学に関するCLIプログラム、Unixソフトウェア、Windowsソフトウェア、Macアプリ、iPhoneアプリ、Webアプリの研究開発、受託開発、受託研究、及び、ネットワークシステム管理に従事

講義で使用する資料

1. 諏訪 悠紀 著
   「iPhoneアプリ開発講座 はじめてのSwift」2016/12/1
   … とてもバランスのとれた良著。但し、Swiftの技術的な疑問には答えてくれないので、特に初心者以外には、これだけでは不足

2. 荻原 剛志 著
   「詳解Swift 第4版」2017/12/26
   … アプリ開発を直接的に目的としていない。Java/C++等のみの、他言語の学習者等が自ずと抱くであろうSwiftの疑問に答える良著。※ 正誤表が配布されているので注意

3. 大重 美幸 著
   「詳細！Swift iPhoneアプリ開発入門ノート」2018/11/3
   … 以上ではアプリ開発には物足りないので、アニメーション、フィンガーアクション、グラフィックス描画、データストレージ、デバイスの機能(各種センサー等)、仮想現実を本書で自習書として活用のこと。※ 正誤表が配布されているので注意 ※ 公式サンプルコードが動かない場合は Bundle Identifier を自分のものにせよ。

4. Apple 公式の開発文書 … アプリ＆言語について学習者が参照すべき(大量にある)一次情報源。但し、プログラミング言語Swiftで解説されているか、対象のSwiftのサンプルコードがあるかは個別に探してみる必要あり。

macOSを使う際の助言

1. **今使っているアプリケーションのメニューは、画面最上部**
   ウィンドウの「赤の●×」ボタンは「終了」ではなく「閉じる」
   ウィンドウの「緑の●」ボタンは「フルスクリーン」、オプション＋「緑の●」ボタンで「拡大/縮小」
   日本語キーボードの場合：バックスラッシュ「\」は、オプション＋「¥」キー

2. キーボードやトラックパッドが使いにくかったら、最上部のメニューバー - アップルメニュー - システム環境設定 - トラックパッドやキーボードで調整可能。但し、アクセシビリティにも関連する設定項目が存在するので注意
   - 特に、システム環境設定 - キーボード - ショートカット - フルキーボードアクセスを「すべてのコントロール」にすることを推奨 → **マウスやトラックパッド無しで、タブキーやカーソルキーで操作可能になる！**

3. ⌘はコマンドキー、⌥はオプションキー、⇧はシフトキー、^はコントロールキー　スペース　リターンキー
   - ⌘タブ … アプリケーション切り替え、⇧⌘タブ …同左の逆順、
     ⌘F1 … アプリケーション内ウィンドウ切り替え、⇧⌘F1 … 同左の逆順
   - ⌘C … コピー、⌘V … ペースト、⌘X … カット、⌘Z … アンドゥ、⇧⌘Z … リドゥ、等々
   - **⌘S … ファイルの保存、⌘W … ファイルを閉じる、⌘O … ファイルを開く、⌘Q … 終了、⌘N … 新規〜**
   - **その他、Emacsライクなキーバインディング**

macOS: GNU Emacsライクなキーバインディング

文字入力するエリアやXcodeのエディタ、プレイグラウンドは以下のショートカットが可能

^f　… 前へ、　　^b　… 後へ、

^n　… 下へ、　　^p　… 上へ、

^a　… 行頭へ、　　^e　… 行末へ、

^h　… ⌫後方削除、　　^d　… ⌦前方削除、

^o　… 次の行に行かずに改行挿入、

^k　… 行末までをバッファに保持して削除、

^y　… カーソル位置にバッファをヤンク、

^t　… 前後の交換、等々

macOS特有の技術用記号

⌘	… command	^	… control
⇧	… shift	⇪	… caps lock
⌥	… option	⎇	… alt
⎋	… esc	fn	… Fn
↩	… return	⌅	… enter
⇥	… tab	⇤	… back tab
⏏	… イジェクトボタン	⏻	… 電源ボタン
⌦	… forward deleteキー	⌫	… deleteキー
fn+⌫	… forward deleteキー	␣	… space
⌧	… clear	⇭	… num lock
↖	… home	↘	… end
fn+←	… home	fn+→	… end
⇞	… page up	⇟	… page down
fn+↑	… page up	fn+↓	… page down
⚙	… setting		… apple menu

iPhoneで何ができるのか？

Apple公式の開発文書目録(1/8)

App Frameworks(1/2)

1.AppKit - macOS
2.Bundle Resources - *(リソース保管)
3.Foundation - *（基礎フレームワーク）
- Fundamentals
  - Numbers, Data, and Basic Values
  - Strings and Text
  - Collections
  - Dates and Times
  - Units and Measurement
  - Data Formatting
  - Filters and Sorting
- App Support
  - Task Management
  - Resources
  - Notifications（通知）
  - App Extension Support
  - Errors and Exceptions
  - Scripting Support
- Files and Data Persistence
  - File System
  - Archives and Serialization
  - Preferences
  - Spotlight
  - iCloud
- Networking（ネットワーキング）
  - URL Loading System
  - Bonjour
- Low-Level Utilities
  - XPC（プロセス間通信）
  - Object Runtime
  - Processes and Threads
  - Streams, Sockets, and Ports

（iOSもしくは* と添えたテーマが iOS 関連）

iPhoneで何ができるのか？

Apple公式の開発文書目録(2/8)

App Frameworks(2/2)

3.Swift - *（プログラミング言語Swift公式ドキュメント）
4.TVML - tvOS
5.TVMLKit - tvOS
6.TVMLKit JS - tvOS
7.TVUIKit ~~[beta]~~ - tvOS
8.UIKit - iOS（基礎的なユーザインターフェースキット）
- App Structure
- Core App
- Resource Management
- App Extensions
- User Interface
- Views and Controls
- View Controllers
- View Layout
- Animation and Haptics（アニメーションと感覚フィードバック）
- Windows and Screens
- User Interactions
- Touches, Presses, and Gestures（タッチ、プレス、ジェスチャー）
- Drag and Drop（ドラッグ＆ドロップ）
- Focus Interactions（フォーカス）
- Peek and Pop（ピーク、ポップ）
- Keyboard and Menus（キーボードとメニュー）
- Accessibility（アクセシビリティ）
- Graphics, Drawing, and Printing
- Images and PDF（画像とPDF）
- Drawing（描画）
- Printing（印刷）
- Text
- Text Display and Fonts（テキスト表示とフォント）
- Text Storage（テキストのストレージ）
- Keyboards and Input（キーボードと入力）

9.WatchKit - iOS

（iOSもしくは* と添えたテーマが iOS 関連）

iPhoneで何ができるのか？

Apple公式の開発文書目録(3/8)

App Services(1/2)

1.Accounts - *（外部アカウントによるアクセスと管理）
2.AddressBook - * ※deprecated by Contacts
3.AddressBookUI - iOS ※deprecated by ContactsUI
4.AdSupport - *（広告サポート）
5.ApplicationServices - macOS
6.BusinessChat - *(Message.appによる顧客とのチャット)
7.CallKit - iOS（電話キット）
8.CarPlay - iOS [beta](カーナビゲーションフレームワーク)
9.ClassKit - iOS（教室キット）
10.ClockKit - watchOS
11.CloudKit - *（クラウドキット）
12.Contacts - *（コンタクト）
13.ContactsUI - *（コンタクトユーザインタフェース）
14.Core Data - *（コアデータフレームワーク）
15.Core Foundation - *
- Utilities
  - Base Utilities
  - Byte-Order Utilities
  - Core Foundation URL Access Utilities
  - Preferences Utilities
  - Socket Name Server Utilities
  - Time Utilities
- Opaque Types

16.Core Location - *（コア位置情報）
17.Core ML - * ~~[beta]~~（コア機械学習）
18.Core Motion - iOS（コアモーション）
- Device Motion(デバイスモーション)
- Getting Processed Device Motion Data(処理されたデバイスモーションデータの取得)
- Accelerometers(加速度センサー)
- Gyroscopes(ジャイロスコープ)
- Pedometer(歩数計)
- Magnetometer(磁気センサー)
- Altitude(高度計)
- Historical Data(これらの履歴)

19.Core Spotlight - *（検索のためのフレームワーク）
20.Core Text - *（コアテキスト）
21.Create ML - macOS ~~[beta]~~（機械学習のためのフレームワーク）
22.DeviceCheck - iOS ~~[beta]~~（デバイス識別API）

（iOSもしくは* と添えたテーマが iOS 関連）

iPhoneで何ができるのか？

Apple公式の開発文書目録(4/8)

App Services(2/2)

23.EventKit - *（予定・行事キット）
24.EventKitUI - iOS（予定・行事キットユーザインタフェース）
25.FileProvider - iOS ~~[beta]~~（ファイルプロバイダ）
26.FileProviderUI - iOS ~~[beta]~~（ファイルプロバイダユーザインタフェース）
27.HealthKit、~~HealthKitUI~~ - iOS（ヘルスキット）
28.HomeKit - iOS(ホームアプリケーションのためのキット)
29.iAd - iOS（旧版の広告フレームワーク）
30.JavaScriptCore - *(Javascript実行環境フレームワーク)
31.~~IdentityLookup~~ - iOS [beta]
31.MapKit - *（地図キット）
32.Messages - iOS（メッセージフレームワーク）
33.MessageUI - iOS（メッセージユーザインタフェース）
34.MultipeerConnectivity - *（マルチP2Pフレームワーク）
35.Natural Language - * ~~[beta]~~（自然言語処理のためのフレームワーク）
36.NewsstandKit - iOS（新聞雑誌棚キット）
37.NotificationCenter - *（通知センター）
38.PassKit - iOS（決済アプリキット）
39.PreferencePanes - iOS（アプリのシステム設定）
40.PushKit - iOS（VoIPプッシュ通知キット）
41.QuickLook - *（プレビューサポートのためのフレームワーク）
42.SafariServices - *（ブラウザSafariを利用するためのフレームワーク）
43.SiriKit - *（音声認識サービスためのキット）
44.SMS and Call Reporting - iOS（メッセージフィルタリング関連）
45.Social - *（ソーシャルネットワークのためのフレームワーク）
46.Speech - *（音声会話フレームワーク）
47.StoreKit - *（Appストア、Apple Musicサービスのためのキット）
48.TVServices - tvOS
49.UserNotifications - *（ユーザ通知のためのフレームワーク）
50.UserNotificationsUI - iOS（ユーザ通知ユーザインタフェース）
51.WatchConnectivity - iOS
52.WebKit - *（ウェブコンテンツや閲覧履歴等のためのキット）

（iOSもしくは* と添えたテーマが iOS 関連）

iPhoneで何ができるのか？

Apple公式の開発文書目録(5/8)

Graphics and Games

1.AGL - macOS
2.ARKit - iOS ~~[beta]~~（仮想現実キット）
3.ColorSync - macOS ~~[beta]~~
4.Core Animation - *（コアアニメーション）
Quartz Coreフレームワークの一部
5.Core Graphics - *（コアグラフィックス）
6.Core Image - *（コアイメージ）
7.Game Controller - *（ゲームコントローラペリフェラル）
8.GameKit - iOS（ソーシャルゲームキット）
9.GameplayKit - *（ゲームプレイキット）
10.GLKit - *（モバイル向けオープングラフィックスライブラリ用ビュー）
11.Image I/O - *（画像入出力）
12.Metal - *(増速コンピュータグラフィックスAPI)
13.Metal Performance Shaders - *（Metalパフォーマンスシェーダー）
14.MetalKit - *（Metalキット）
15.Model I/O - *（3Dモデル入出力）
16.OpenGL ES - iOS（モバイル向けオープングラフィックスライブラリ）
17.PDFKit - *(ポータブル文書フォーマットキット)
18.Quartz - macOS
- Viewing and Transforming Images (ImageKit)
- Displaying PDFs(PDFKit)
- Quartz Composer
- Using Quick Look
- Conversion Filters(Quartz Filter)

19.ReplayKit - iOS（リプレイキット）
20.SceneKit - *（3Dグラフィックスと物理計算）
21.SpriteKit - *（スプライト画像キット）
22.Vision - *（コンピュータビジョンフレームワーク）

（iOSもしくは* と添えたテーマが iOS 関連）

iPhoneで何ができるのか？

Apple公式の開発文書目録(6/8)

Media

1.Apple News - *（AppleニュースAPIとフォーマット）
2.AssetsLibrary - iOS ※deprecated by Photos
3.AudioToolbox - *（音響ツールボックス）
4.AudioUnit - *（音響ユニット）
5.AVFoundation - *（音響映像基盤）
- Playback and Editing
- Media Capture
- Audio
- Speech

6.AVKit - iOS（音響映像キット）
7.Core Audio - *（コアオーディオ）
8.Core Audio Kit - *（コアオーディオキット）
9.Core Media - *（コアメディア）
- Sample Processing
- Time Representation
- Media Synchronization
- Text Markup
- Metadata
- Queues

10.Core MIDI - *（コアMIDI）
11.Core Video - *（コアビデオ）
- Data Processing
- Time Management
- Metal
- OpenGL/OpenGL ES

12.FxPlug - Objective-C, Final Cut Pro
13.HTTP Live Streaming - *(動画ライブ配信フレームワーク)
14.iTunesLibrary - macOS, Objective-C
15.JavaScriptCore - *（Javascript実行環境フレームワーク）
16.Media Player - *（メディア再生のためのフレームワーク）
17.MediaAccessibility - *（メディアのクローズドキャプション表示のためのフレームワーク）
18.MediaLibrary - macOS
19.~~Photos - *（写真のためのフレームワーク）~~
20.~~PhotosUI - *（写真ユーザインタフェース）~~
21.PhotoKit - *（写真のためのフレームワーク）
22.Professional Video Applications - macOS, Final Cut Pro
23.QTKit - macOS
24.SafariServices - *（Safariブラウザサービス）
25.ScreenSaver - macOS
26.TVMLKit - tvOS
27.TVMLKit JS - tvOS
28.VideoToolbox - *（ビデオツールボックス）
29.WebKit - *（ウェブキット）

（Objective-C と添えたテーマはその言語の知識が必要）
（iOSもしくは* と添えたテーマが iOS 関連）
その他、オーディオAPIの OpenAL は独立したオープンソースプロダクトのためか、ここには未掲載

iPhoneで何ができるのか？

Apple公式の開発文書目録(7/8)

Web/Developer Tools

Web

1.Apple Store Connect API - www
2.Apple Music API - www
3.Apple Pay on the Web - Safari
4.CloudKit JS - www
5.LivePhotosKit JS - www
6.MapKit JS - www [beta]
7.MusicKit JS - www [beta]
8.Safari Extensions JS - www
9.WebKit JS - www

Developer Tools

1.Automator - macOS
2.Code Diagnostics - Xcode
3.InstallerJS - macOS
4.Playground Support - macOS
5.PlaygroundBluetooth - macOS
6.ScriptingBridge - macOS
7.XcodeKit - macOS
8.XCTest - Xcode

（iOS, Xcode, wwwもしくは* と添えた
テーマが iOS 関連）

iPhoneで何ができるのか？

Apple公式の開発文書目録(8/8)

System

1.Accelerate - *（高速化のためのフレームワーク）
・BNNS（基礎的ニューラルネットワークサブルーチン群）
・Quadrature（求積法）
・BLAS（基礎的線形代数サブルーチン群）
・Sparse Solvers（疎な行列の解法）
・vDSP（ベクトル化されたデジタル信号処理）
・vecLib（ベクトル高速演算）
・vImage（高速画像処理ライブラリ）
・simd(シングルインストラクションマルチプルデータライブラリ)
2.AuthenticationServices [beta]
3.CFNetwork - *（コアファウンデーションのネットワーキング）
4.Collaboration - macOS
5.Compression - *（データ圧縮と伸長）
6.Core Bluetooth - *（コアブルートゥース）
7.Core NFC - iOS [~~beta~~]（コア近距離無線通信）
8.Core Services - macOS
9.Core Telephony - *（コアテレフォニー）
10.Core WLAN - macOS
11.CryptoTokenKit - macOS
12.~~DarwinNotify~~ - * [objc]
13.DiskArbitration - macOS
14.Dispatch - *（GCDによる並列処理のためのフレームワーク）
15.~~dnssd~~ - * [objc]（DNSによるサービス検出）
16.ExceptionHandling - macOS
17.ExternalAccessory - *（外部装置のためのフレームワーク）
18.FinderSync - macOS（Finderとの連携のためのフレームワーク）
19.ForceFeedback - macOS
20.FWAUserLib - macOS
21.GSS - *（汎用セキュリティサービスのためのフレームワーク）
22.Hypervisor - macOS
23.InputMethodKit - macOS
24.IOBluetooth - macOS
25.IOBluetoothUI - macOS
26.IOKit - * [~~private~~]（I/Oキット）
27.IOSurface - * [~~beta~~]（フレームバッファとテクスチャのためのフレームワーク）
28.~~Kernel~~ - macOS
29.LatentSemanticMapping - macOS(文章の分類のためのフレームワーク)
30.LocalAuthentication - *（ローカル認証フレームワーク）
31.MobileCoreServices - iOS（モバイルコアサービス）
32.Network - * [beta]（セキュアネットワーキング）
33.NetworkExtension - *（VPNネットワーク）
34.Objective-C Runtime - *（Objective-Cランタイムの低レベルAPI）
35.OpenDirectory - macOS
36.os - *
37.Security - *（セキュアなデータ管理とアクセス制御）
38.SecurityFoundation - macOS
39.SecurityInterface - macOS
40.ServiceManagement - macOS（launchd API）
41.simd - *(シングルインストラクションマルチプルデータライブラリ)
42.SystemConfiguration - *（システム環境設定に関するフレームワーク）
43.vmnet - macOS
44.XPC - macOS

（[objc] と添えたテーマはObjective-Cの知識が必要）
（iOSもしくは* と添えたテーマが iOS 関連）

Swift、目下の注意事項

1. Swift3 で言語仕様やAPIが大幅に整理されて、以前のコードでは動かないので注意
   → よって、古いネット情報はそのままでは動かない。

2. もし必要なら、XcodeのEditメニュー - Convert - To Current Swift Syntaxで変換できると思われるので参考にされたし。

3. Swift4 では、これほどの後方互換性を失う改変は行われなかったが、細かなAPI等の整理がなされているので、Swift3からの多少の変更は必要 → Appleの一次情報に当たるかXcodeの補完機能、上記のマイグレーション機能、警告やヘルプを活用するとよい

4. Swift5 でも同様

Swift on Xcode 事始め

1. Xcodeのインストール（P.010）
   … メニューバー - アップルメニュー - App StoreのApple Appから、… **要Apple ID**
   … 30分程度かかる模様、… 2019/08/27時点で最新はバージョン10.3
   … そして、App Storeの「開く」からは開かない(かも)
   … 管理者でもある利用者のログインパスワードが必要
   … Finder のアプリケーションからXcodeを開く
   (注) 実機のための署名は１０アプリ/７日！

2. 「Get started with a playground」でプログラミング言語Swiftを試してみましょう！
   … 下部コンソールの「▶」ボタンで実行、だがリアルタイムに実行される。
   … エラーの箇所がコードの左「●」が出なくなる不具合、もしくは、やたら実行が遅い！
   → **保存して、開き直す。それでも駄目なら、Xcodeを立ち上げ直して、開き直す。始めから作り直す。**
   … いずれもリアルタイム実行を下部コンソールの「▶」ボタンの長押しでオフにしておけば多少緩和
   … ~~間違って ^Z と押してしまうと**見えない**制御文字「^z」が意図せず入って**謎のエラー！**~~
   … プロジェクトでstoryboardの**部品が表示されないバグ！**→ マシンの再起動！？

・よく使うショートカット：⌘F … 検索、⌥⌘F … 検索と置換、^スペース … 候補、^⌘スペース … 特殊文字
(アプリ制作のときは、⌘R … Run、⇧⌘K … Clean、⌘B … Build、⌘I … Profile)

macOSを使う際の助言２

`MyPlayground.playground` は**実はフォルダ**

1. Finder から `MyPlayground.playground` を「右クリック」もしくは「２本指でタップ」
2. メニューの「パッケージの内容を表示」でフォルダの中に入れます。
3. すると、以下のファイルがあることが解る。

`Contents.swift`	← **Xcodeで編集しているファイルの実体はコレ！**
`contents.xcplayground`	… これは上記の設定ファイル(XML形式)
`playground.xcworkspace/`	… これはさらにフォルダで、その他、設定情報等

4. 例えば、**Emacs.app** のようなエディタでこれ(`Contents.swift`)を編集することも可能です。

Emacs.app - `https://emacsformacosx.com/`

歴史的な用語のまとめ

1. 接頭辞「NS」… オペレーティングシステム「NeXTSTEP」の頭文字。macOSの起源
   - NSURL、NSXMLParser など、Swift3からその多くの「NS」が取り払われたが、取り払われていないものも~~依然として多い~~少なからずある。
2. 接頭辞「IB」… NeXTSTEP開発者向けアプリケーション「Interface Builder」の頭文字。現在は、かつての同「Project Builder」とともに統合開発環境Xcodeに統合されているので、直接目に触れることはない。
   - Objective-C … 上記の主たる開発言語。Smalltalk型のオブジェクト指向をプログラミング言語Cにハイブリッド化した言語
3. 接頭辞「CF」 … フレームワーク「Core Foundation」の頭文字。Objective-C向けのFoundationフレームワークとほぼ同等の機能をCで提供するもの
   - Objective-C++ … Smalltalk型のオブジェクト指向をプログラミング言語C++にハイブリッド化した言語。**双方のオブジェクト指向は完全に独立しているので、当然のことながら他方を継承したりはできない。**

プログラミング言語Swiftの非常に重要な特徴

1. 2016年リリース。コンパイラによる最適化が効きやすい言語設計
   よって、**Objective-C よりも高速に動作**する。

2. C/C++のような**ヘッダファイル(`*.h`/`*.hpp`)が不要**

3. **参照型はクラス(とクロージャ)のみで、他はすべて値型**という、かなり思い切った言語仕様

4. **C/C++のポインタという概念は(基本的には)無い。**→ 安全に寄与

5. C/C++, Java, Javascriptにあるような**暗黙的型変換は無い**(些か不便…)。

6. (弁別型)共用体としても使える**高機能な enum 型**(C/C++等の列挙型 enum と共用体 union とは**一線を画す**)と**高機能な switch 文**(C/C++等の switch 文とは**一線を画す**)

7. 自動参照カウンティング(ARC: Automatic References Counting)によるメモリ管理

8. **構文はC/C++, Java, Javascriptなどとは異なり**、引数ラベルなど、むしろScalaにとてもよく似ている。さらに、コードを簡潔に書くための工夫がふんだんに取り入れられている。

資料における用語の補足と Swiftらしいコードの例

1. **外部引数名 → 引数ラベル**

```
func f(x: Double) -> Double { return x }
let y = f(x: 0) // ここでコロン「:」の左にあるx
```

2. **暗黙的開示オプショナル型 = 有値オプショナル型**

```
var x: Double! = nil // xの型は「Double!」
```

3. **失敗のあるイニシャライザ = エラーになり得るイニシャライザ**

```
class foo {
    let x: Double
    init?(_ x: Double) {
        if x < 0 { return nil }
        self.x = x
    }
}
x = foo(-1)?.x //「?」と「nil」
```

4. **エラーを投げる関数 = エラー通報関数 = エラーを投げ得る関数**

```
enum SomethingError : Error { case negativeValue }
func g(_ x: Double) throws -> Double {
    if !(x < 0) { return x }
    else { throw SomethingError.negativeValue }
}
do {
    let y = try g(-1)
}
catch SomethingError.negativeValue { print(“Error") }
//「throws, throw」と「try, catch」
```

プログラミング言語Swift

「諏訪 著: iPhoneアプリ開発講座 - はじめてのSwift」の前半

1. iPhoneアプリ開発に必要なもの(P.001)

2. XcodeのPlaygroundで学ぶSwift(P.009)

3. Swiftの基本的な文法(P.023)
   変数と定数(P.024)、値の型(P.031)、コレクション(P.039)、演算子(P.053)、制御構文(P.060)、関数(P.074)、オプショナル(P.087)

4. Swiftとオブジェクト指向プログラミング(P.099)
   型(P.101)、クラス(P.102)、プロパティ(P.104)、メソッド(P.113)、サブスクリプト(P.119)、参照カウント(P.123)、イニシャライザとデイニシャライザ(P.130)、継承(P.134)、ストラクチャ(P.141)、列挙型(P.148)、プロトコル(P.152)、エクステンション(P.163)

5. Swiftの発展的な文法(P.167)
   タプル(P.168)、関数オブジェクト(P.173)、クロージャ(P.180)、型のキャスト(P.192)

※使用する配布リソース：`MyPlaygrounds-*.zip`

プログラミング言語SwiftによるiPhoneアプリ開発

「諏訪 著: iPhoneアプリ開発講座 - はじめてのSwift」の後半

6. iOSアプリ開発入門(P.205)とサンプルアプリ(P.220)

7. iOSアプリ開発レシピ(P.245)

    A. (整数)電卓アプリ(P.246)

    B. (簡易)タイマーアプリ(P.266)

    C. 地図(マップ)アプリ(P.278)

    D. 図鑑アプリ(P.291)、但し「図鑑」と言っても写真は後述

    E. 写真ビューアアプリ(P.313)

※使用する配布リソース：

`Sample-*.zip`
`Calculator-*.zip`
`Timer-*.zip`
`Map-*.zip`
`Animals-*.zip`
`PhotoViewer-*.zip`

~~※書籍公式配布リソース(Swift4)：~~

~~`Swift4Sample.zip`~~

詳解Swift第4版の内容(1/2)

1. Swiftでプログラミング(荻原P.002)
   データ型と変数、制御構文、簡単な実行方法
2. 関数(荻原P.038)
   関数定義の基本、関数定義におけるさまざまな設定、オーバーロード、タプル、演算子、演算子の定義(優先度グループ)
3. 構造体(ストラクチャ)(荻原P.068)
   構造体の定義、メソッド、プロパティ、添え字付け(サブスクリプト)~~、プロトコル~~
4. オプショナル(荻原P.094)
   オプショナル型、オプショナル束縛構文、有値オプショナル型、失敗のあるイニシャライザ
5. プロトコル(荻原P.110)
   プロトコルとは(プロトコル指向プログラミング言語Swift)、プロトコルの宣言、プロトコルと付属型、代表的なプロトコルの例、値型データの共有
6. 基本的なデータ型(荻原P.136)
   ~~整数と実数、~~範囲型とStride型、配列、文字列と文字、辞書
7. パターン(荻原P.168)
   タプルとswitch文、列挙型、共用体の列挙型、パターンマッチ
8. クラスと継承(荻原P.190)
   クラス定義、イニシャライザ、継承とサブクラスの定義、クラスと型(キャスト、Selfほか)、解放時処理~~、遅延格納型プロパティ~~
9. メモリ管理(荻原P.226)
   参照型データとARC(Automatic Reference Counting)、強い参照の循環、オプショナルチェーン、キーパス

※使用する配布リソース：`Playunderground-2019*.zip`

詳解Swift第4版の内容(2/2)

10. 拡張(エクステンション)(荻原P.246)
    拡張の宣言、拡張定義とプロトコルへの適合(準拠)、プロトコル拡張、集合とプロトコル
11. エラー処理(荻原P.266)
    エラー処理構文、処理の中断と後始末(defer文)、アクセス制御、アサーションとテスト、利用可能条件とコンパイラ制御文、インスタンスのシリアライズ
12. クロージャ(荻原P.306)
    クロージャの宣言、変数のキャプチャ、クロージャの使い方と記法、クロージャと強い参照の循環、クロージャの応用
    (@autoclosure、短絡評価をする演算子の定義ほか、遅延格納型プロパティほか)
13. ジェネリクス(荻原P.342)
    ジェネリクスの概要、ジェネリック関数、ジェネリクスによる型定義、**シークエンス操作**
14. C/Objective-Cとのデータの受け渡し(荻原P.374)
    互換なデータ型とポインタ、Data型とポインタ、対応づけられたデータ型、列挙型とその他データ型
15. Objective-Cとの連携(荻原P.414)
    SwiftとObjective-Cをビルドする、Objective-Cのオブジェクトを使う、SwiftのクラスをObjective-Cで使えるようにする、GUI部品との連携、Playground向け言語仕様
16. コーディングとデバッグ(荻原P.413)
    Playground向けの言語仕様、文書化コメント、デバッガLLDBの簡単な使い方、プロトコル指向
17. Appendix(荻原P.429)
    Swiftの標準ライブラリ(プロトコル、型、ジェネリック型、ポインタ型、関数、属性)、~~Swiftの構文図(識別子・演算子で使える文字、構文図)~~、文書化コメント、デバッガLLDBの簡単な使い方

※使用する配布リソース：`Playunderground-2019*.zip`

詳解Swift第3版の内容(1/2)

😌旧版ですがそのまま載せておきます。新版への大きな変更はありません😌

@objc識別子とUnicode.Scalar型、Subsequence型などに関する仕様変更

1. Swiftでプログラミング(荻原P.001)
   データ型と変数、制御構文、簡単な実行方法
2. 関数(荻原P.037)
   関数定義の基本、関数定義におけるさまざまな設定、オーバーロード、タプル
3. 構造体(ストラクチャ)(荻原P.057)
   構造体の定義、メソッド、プロパティ、添え字付け(サブスクリプト)、プロトコル
4. オプショナル(荻原P.085)
   オプショナル型、オプショナル束縛構文、有値オプショナル型、失敗のあるイニシャライザ
5. 基本的なデータ型(荻原P.101)
   整数と実数、範囲型とStride型、配列、文字列と文字、辞書
6. パターン(荻原P.133)
   タプルとswitch文、列挙型、共用体の列挙型、パターンマッチ
7. 演算子(荻原P.155)
   Swiftの演算子、演算子の定義、優先度グループ、プロトコルと演算子の定義
8. クラスと継承(荻原P.169)
   クラス定義、イニシャライザ、継承とサブクラスの定義、解放時処理、遅延格納型プロパティ
9. メモリ管理(荻原P.207)
   参照型データとARC(Automatic Reference Counting)、強い参照の循環、オプショナルチェーン

※使用する配布リソース：Playunderground-2017*.zip

詳解Swift第3版の内容(2/2)

😌旧版ですがそのまま載せておきます。新版への大きな変更はありません😌

@objc識別子とUnicode.Scalar型、Subsequence型などに関する仕様変更

10. プロトコル(荻原P.227)
    プロトコルの宣言、プロトコルと型、プロトコルと付属型
11. 拡張(エクステンション)(荻原P.243)
    拡張の宣言、拡張定義とプロトコルへの適合(準拠)、プロトコル拡張、集合とプロトコル
12. エラー処理(荻原P.263)
    エラー処理構文、処理の中断と後始末(defer文)、アクセス制御、アサーションとテスト、利用可能条件とコンパイラ制御文
13. クロージャ(荻原P.293)
    クロージャの宣言、変数のキャプチャ、クロージャの使い方と記法、クロージャと強い参照の循環
14. ジェネリクス(荻原P.323)
    ジェネリクスの概要、ジェネリック関数、ジェネリクスによる型定義
15. C/Objective-Cとのデータの受け渡し(荻原P.343)
    互換なデータ型とポインタ、Data型とポインタ、対応づけられたデータ型、列挙型とその他データ型
16. Objective-Cとの連携(荻原P.381)
    SwiftとObjective-Cをビルドする、Objective-Cのオブジェクトを使う、SwiftのクラスをObjective-Cで使えるようにする、GUI部品との連携
17. コーディングとデバッグ(荻原P.413)
    Playground向けの言語仕様、文書化コメント、デバッガLLDBの簡単な使い方、プロトコル指向
18. Appendix(荻原P.429)
    Swiftの標準ライブラリ(プロトコル、型、ジェネリック型、ポインタ型、関数、属性)、Swiftの構文図(識別子・演算子で使える文字、構文図)

※使用する配布リソース：Playunderground-2017*.zip

上記に載っていない有益な技術 in Swift

1. 書式付き出力printf, snprintf代替

```
print(String(format: "%6.3g, %6.3g", x, y))
let string = String(format: "%6.3g, %6.3g", x, y)
```

2. 正規表現による文字列の検索と置換
   - 正規表現そのものについては以下を参照：
     http://www.aihara.co.jp/~taiji/lecture-2019/#regex
3. システムのファンクションコール＝C関数を呼ぶ
4. 特殊文字の入力方法 - ^⌘スペース

   メニューの編集(Edit) - 絵文字と記号の入力(Emoji & Symbols) - 文字ウィンドウの「⚙」メニューで「リストをカスタマイズ」して「Unicode」等を選べるようにしておくことを推奨
5. コマンドラインプログラム

※使用する配布リソース：

MyPlaygrounds-*.zip
ModuloOperation-*.zip, URLcat-*.zip, SampleXMLParser-*.zip

コマンドラインプログラム in Swift

1. Xcode の File - New - Projects - macOS - Command Line Tool で作成したプロジェクト「ModuloOperation」

2. File - New - File で「DivideAndModulo.swift」をモジュールとして追加、編集

3. File - Add Files で「DivideAndModulo.swift」をプロジェクトに追加

```
# 端末使いのための makefile の編集→と使い方↓

$ make
swiftc -O -o ModuloOperation main.swift DivideAndModulo.swift

$ ./ModuloOperation # 実行可能！

$ make clean
rm -f ModuloOperation
```

```
# makefile
TARGET=ModuloOperation
SWIFTCFLAGS=-O

all: $(TARGET)

$(TARGET): main.swift DivideAndModulo.swift
	swiftc $(SWIFTCFLAGS) -o $@ $^

clean:
	rm -f $(TARGET)
```

4. そのような例において、ModuloOperation/ の中：
   ./ModuloOperation/
   ./ModuloOperation/DivideAndModulo.swift
   ./ModuloOperation/main.swift
   ./ModuloOperation/makefile … 例えばこんなファイルを置けば『端末使い』には便利！
   ./ModuloOperation/ModuloOperation … ⌘Rで他所に生成される実行ファイルをここに生成

コマンドライン引数 in Swift

1. Xcode の File - New - Projects - macOS - Command Line Tool で作成したプロジェクト「URLcat」

```
// main.swift
import Foundation

var a = 1 // コマンドライン引数の解析例

for i in a..<Int(CommandLine.argc) {
    a = i
    if CommandLine.arguments[i] == "--help" {
        print("usage: \(CommandLine.arguments[0]) [URL ...]")
        exit(0)
    }
    else { break }
}
for i in a..<Int(CommandLine.argc) {
    print("\(i)\t\(CommandLine.arguments[i])")
    if let url = URL(string: CommandLine.arguments[i]),
       let htmlString = try? String(contentsOf: url) {
        print(htmlString)
    }
}
```

```
# makefile
TARGET=URLcat
SWIFTCFLAGS=-O

all: $(TARGET)

$(TARGET): main.swift
	swiftc $(SWIFTCFLAGS) -o $@ $^

clean:
	rm -f $(TARGET)
```

2. そのような例において、URLcat/ の中：
   ./URLcat/
   ./URLcat/main.swift
   ./URLcat/makefile … 例えばこんなファイルを置けば『端末使い』には便利！
   ./URLcat/URLcat … ⌘Rで他所に生成される実行ファイルをここに生成

コマンドラインプログラムにおける XMLParser in Swift

1. Xcode の File - New - Projects - macOS - Command Line Tool で作成したプロジェクト「SampleXMLParser」

```
// main.swift
import Foundation
class SampleXMLParser : NSObject, XMLParserDelegate {
    // NSObject型を継承し、プロトコルXMLParserDelegateに準拠したクラスSampleXMLParser型の定義

    var parser: XMLParser?
    func parse(location: String) {
        if let url = URL(string: location) {
            parser = XMLParser(contentsOf: url)
        }
        parser?.delegate = self;
        parser?.parse()
    }
      : // ここに様々なXMLParserDelegateに準拠したメソッドを定義すればXMLパースが可能となる！

}
var a = 1
      : // 先の例と同様につき省略

for i in a..<Int(CommandLine.argc) {
    print("\(i)\t\(CommandLine.arguments[i])")
    let sampleXMLParser = SampleXMLParser()
    sampleXMLParser.parse(location: CommandLine.arguments[i])
}
```

```
# makefile
TARGET=SampleXMLParser
SWIFTCFLAGS=-O

all: $(TARGET)

$(TARGET): main.swift
      swiftc $(SWIFTCFLAGS) -o $@ $^

clean:
      rm -f $(TARGET)
```

よくある間違い

1. ~~「^Z」のせいで制御文字が混ざってしまっている。~~
2. 「_␣」の空白「␣」がない。
3. 大文字小文字の区別のミス、スペルミス、または、意図せず空白が入っている。
4. デリゲート(移譲)に準拠すべきクラスのメソッドが足りてない。
5. レイアウトの制約が不適切
6. 各種の関連付けがダブっている、または、不適切
7. `Identifier`の名付けを忘れている。
8. 「Info」の設定し忘れ
9. 補完で違うのを選んでいる(例：`destination` -> `description`)。
10. `Tag`の番号を設定し忘れている、または、違うものに設定している。
11. ブレークポイントが意図せず設定されている。
12. コピー＆ペースト由来の意図しない文字コード(例: 「〃」のつもりが「”〃〝”“”〝〃」)。

発展と課題について(1/3)

1. 前述のApple公式開発文書(https://developer.apple.com/documentation)の目録にて、特に日本語を添えたテーマについて、

    - **Swiftのサンプルコードが掲載されていたり、配布されていたりするもの**

    これらを発展のテーマとして、挑戦してみるのが良い。

2. それでも敷居が高いという人は、Apple公式開発文書の日本語訳(https://developer.apple.com/jp/documentation/)もお薦めする。テーマは網羅されていない上に Swift 向けも少ないことに留意

3. やはり、それでも探すのに手間取ったり、適度なテーマはかつての Objective-C 向けしかなかったりするので、他の適当な「書籍」に当たってみることも検討されたい。ネット情報にあたるもの良いが、いずれにしても、Swift2 のコードのままであることが多いので、前述の Xcode の Convert も検討されたい。

発展と課題について(2/3)

1. 前述のApple公式開発ライブラリ(https://developer.apple.com/library/content/navigation/)の Resource Types - Sample Code にて、**2015年以降に Swift 版が登場し始める**が、

    - 探すのは大変労力がかかる！いくつも試すときは実機はなるべく外しておくこと(７アプリ/１０日制限！)。

    - AppleWatch との連携**も**配慮したサンプルには watchOS のコードも同梱されており、iOS にも関係がある。

    - プラットフォームが macOS, tvOS のみのサンプルは、iOSつまりiPhoneには無関係。

2. WWDC Sample Code をウェブで探すと、興味あるものが見つかるかもしれない。

発展と課題について(3/3)

書籍を活用を検討：以下に一部紹介

1. **「本気ではじめるiPhoneアプリ作り」西 磨翁、SBクリエイティブ、416頁、2016/02/27 ＊Swift4対応改訂版が出来＊Xcode10対応改訂版が出来**
   - 永続的なデータを扱うアプリ
   - サウンドとアニメーションのアプリ
   - 通信アプリ
   - アプリ公開について
2. **「これからつくる iPhoneアプリ開発入門」藤 治仁ほか、SBクリエイティブ、386頁、2016/10/26**
   - 楽器アプリ
   - マップアプリ、タイマーアプリ
   - カメラとSNS投稿アプリ
   - Web APIとJSONを利用したテーブルビューアプリ
3. **「詳細！Swift3 iPhoneアプリ開発入門ノート」大重 美幸、ソーテック社、640頁、2016/11/5 ＊Swift4対応改訂版が出来＊Xcode10対応改訂版が出来**（←自習書）
   - アニメーション
   - フィンガーアクション
   - グラフィックス描画
   - データストレージ
   - デバイスの機能(各種センサー等)
4. **「絶対に挫折しない iPhoneアプリ開発『超』入門」高橋 京介、400頁、2016/11/30 ＊Swift4対応改訂版が出来＊Xcode10対応改訂版が出来**
   - SNSアプリ開発
   - カメラアプリ
   - アプリ公開で収益を上げるために
5. **「Swift 実践入門」石川 洋資、西山勇世、464頁、2017/02/07 ＊Swift4対応改訂版が出来**
   - 本格的なWeb APIクライアント