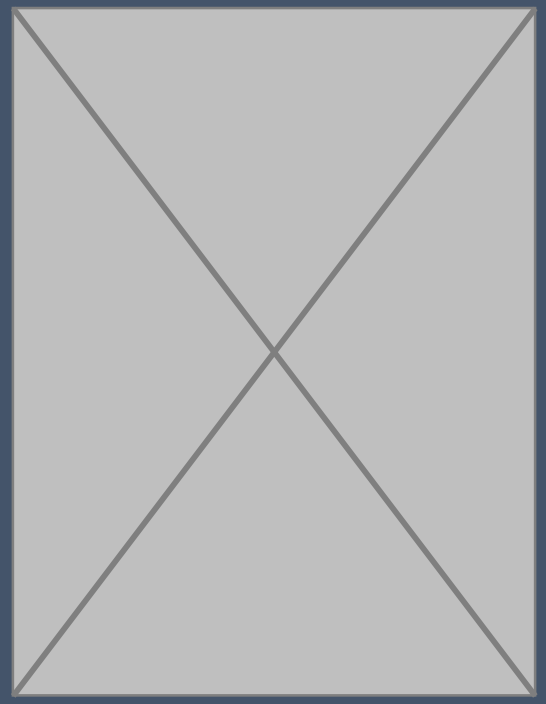
OSS 開発者向け
iOS アプリ開発のはじめかた

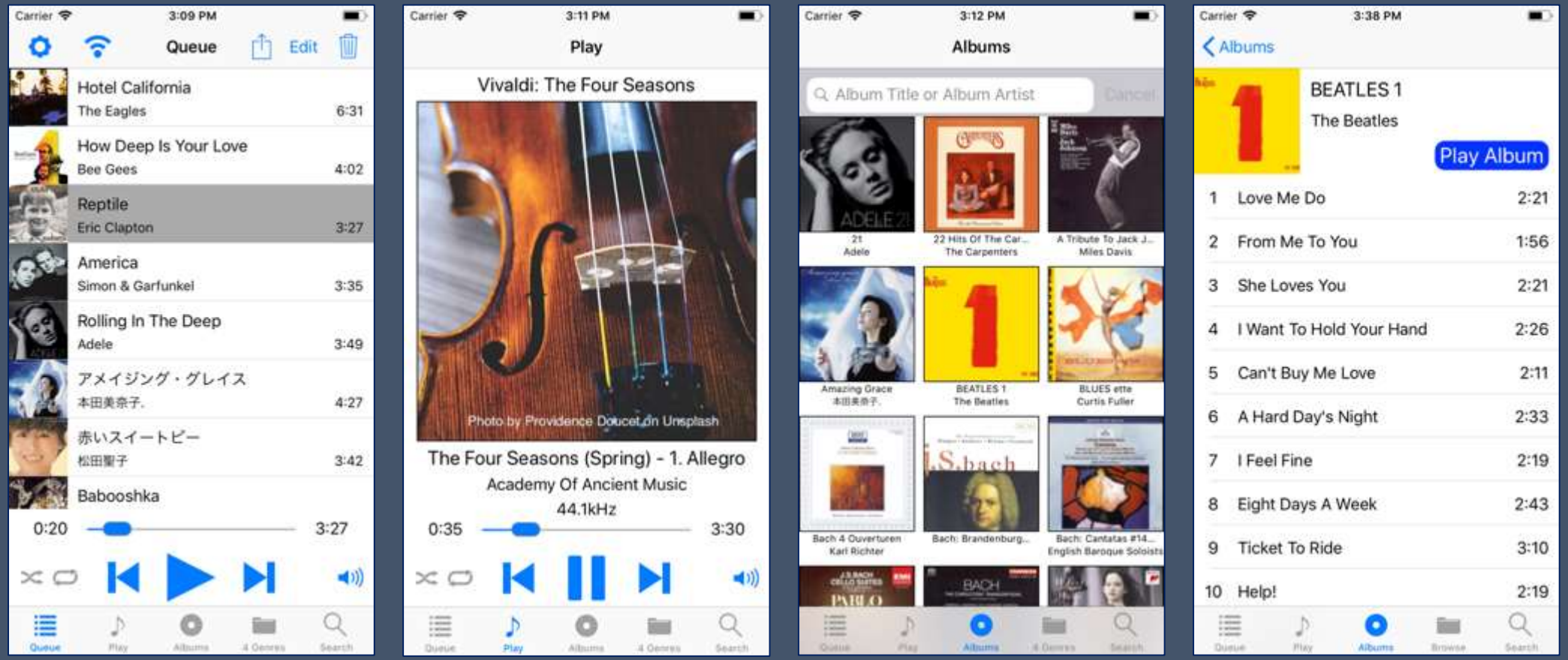
2019.08.24

ラズパイオーディオの会
鈴木康之
openaudiolab.com/jp



配布にあたり一部画像は非表示にしております

副業で作っているプログラム



yaMPC – yet another Music Player Client
MPD（Music Player Daemon 用コントールアプリ）
＊ オープンソースを使って開発していますが、yaMPC自体は（まだ）オープンソースではありません。

本日のTarget Audience

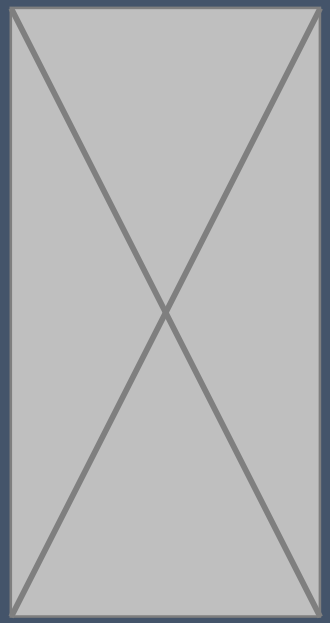
- レベル：入門編
- 対象者：iOSアプリ開発をこれから始めてみたい方
- 前提知識：オブジェクト指向言語での開発経験

Agenda

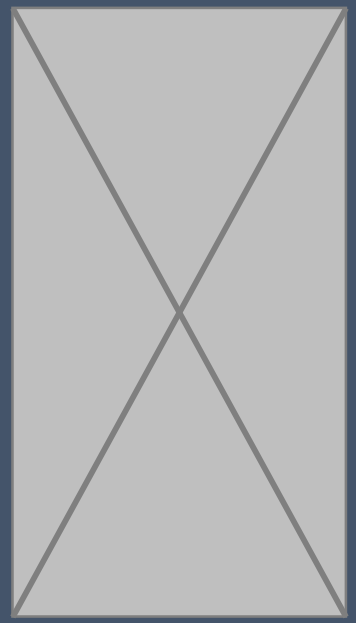
- Introduction
- Swift, UIKit教材紹介
- Swift言語の特徴
- GitとGitHub
- デバッグ
- Beta Test
- クラッシュログの解析
- WWDC19トピック　Xcode 11 / iOS 13
  - iPad Apps for Mac
  - SwiftUI
  - SF Symbols
  - iOS 13 Dark Mode

Why Apps?

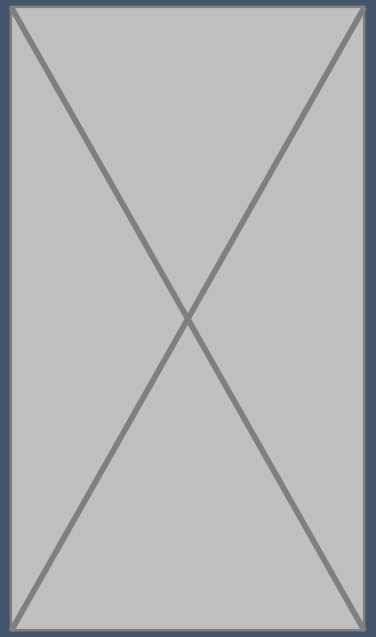
- Webアプリ主流の時代に、なぜiOSアプリ？
- Webアプリ＋スマホアプリの時代
- スマホアプリファーストの時代？



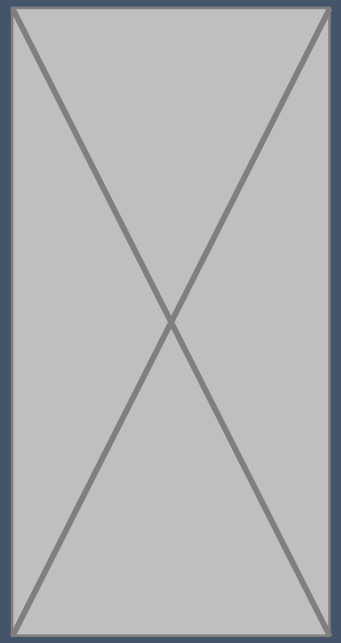
Amazon.co.jp



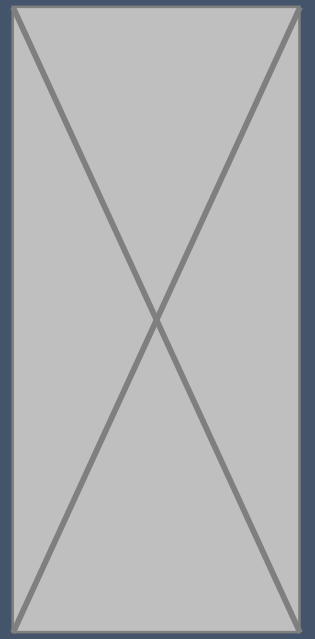
Timesカーシェア



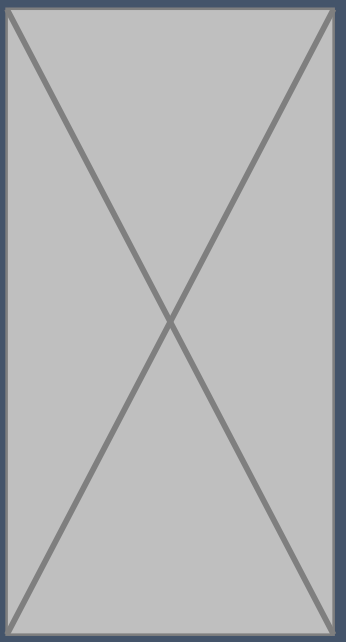
YouTube



新幹線EX予約



Google 翻訳



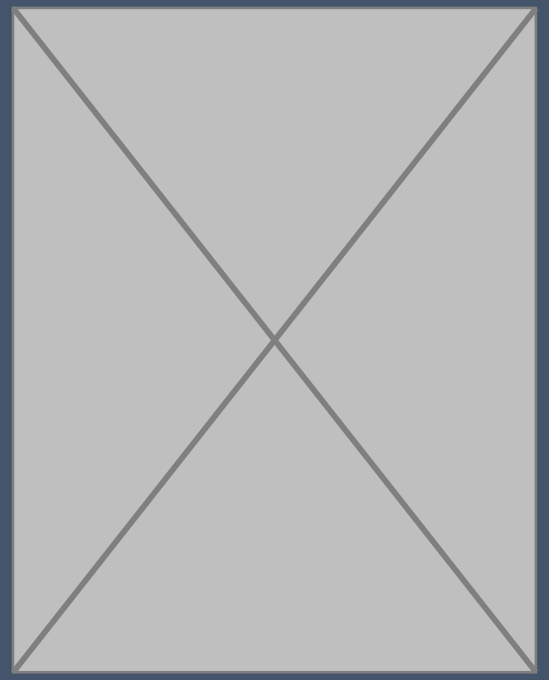
Google Maps

iOS開発に必要なもの

- iOS開発に必要なもの
  - Mac
  - Xcode (App Storeから無償ダウンロード)
  - Apple Developer Program メンバーシップ
    実機テストだけなら無償、App Storeでの配布は有償 (税抜 ￥11,800/年)
  - iPhone/iPadの実機

- iOS開発に必要な知識
  - 言語：Swift
  - フレームワーク：Foundation, UIKit
  - IDE：Xcode

で、なにから始めたらよいのか？
市販本のサンプルコードの写経から？



- いちおうiOSアプリ作れるようになるけど…
- 疑問点がいっぱい
  - Xcodeの詳細な使い方
    - デバッガー
    - プロファイラー
    - 単体テストの自動化（JUnitみたいなの）
  - オブジェクトのパーシステント化（シリアライズ、iPhoneに保存、デシリアライズ）
  - Beta Testのためのアプリ配布方法
  - アプリのクラッシュログの見方
  - Table Viewの作り方詳細
  - など…

UIKit - Table View

もっとも柔軟性があり、
もっとも利用範囲の広いUI部品

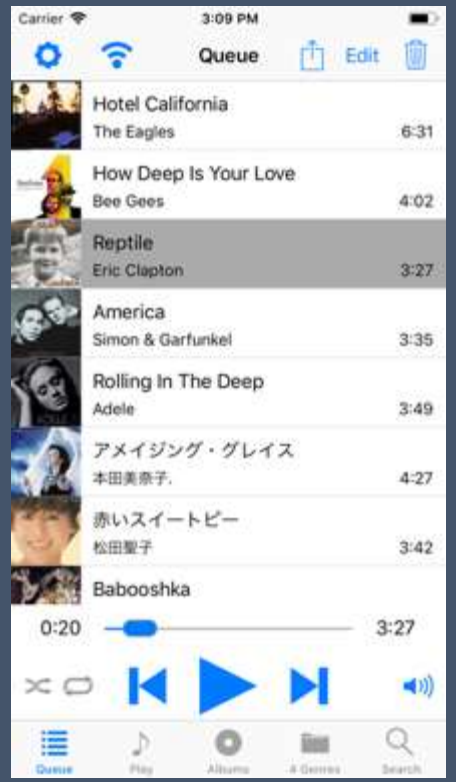
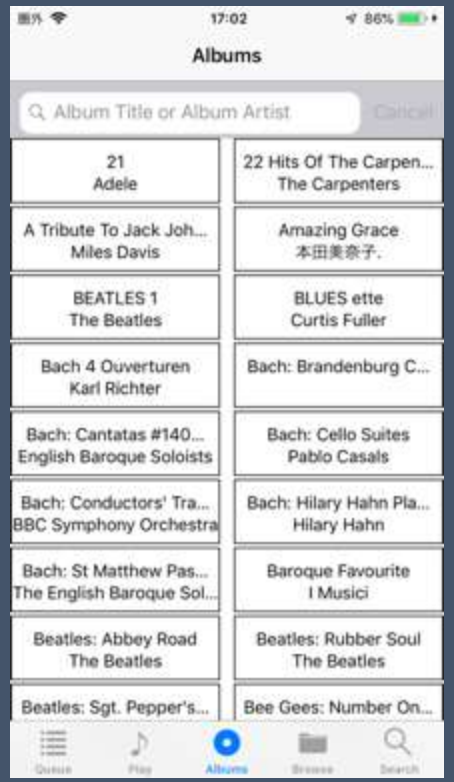
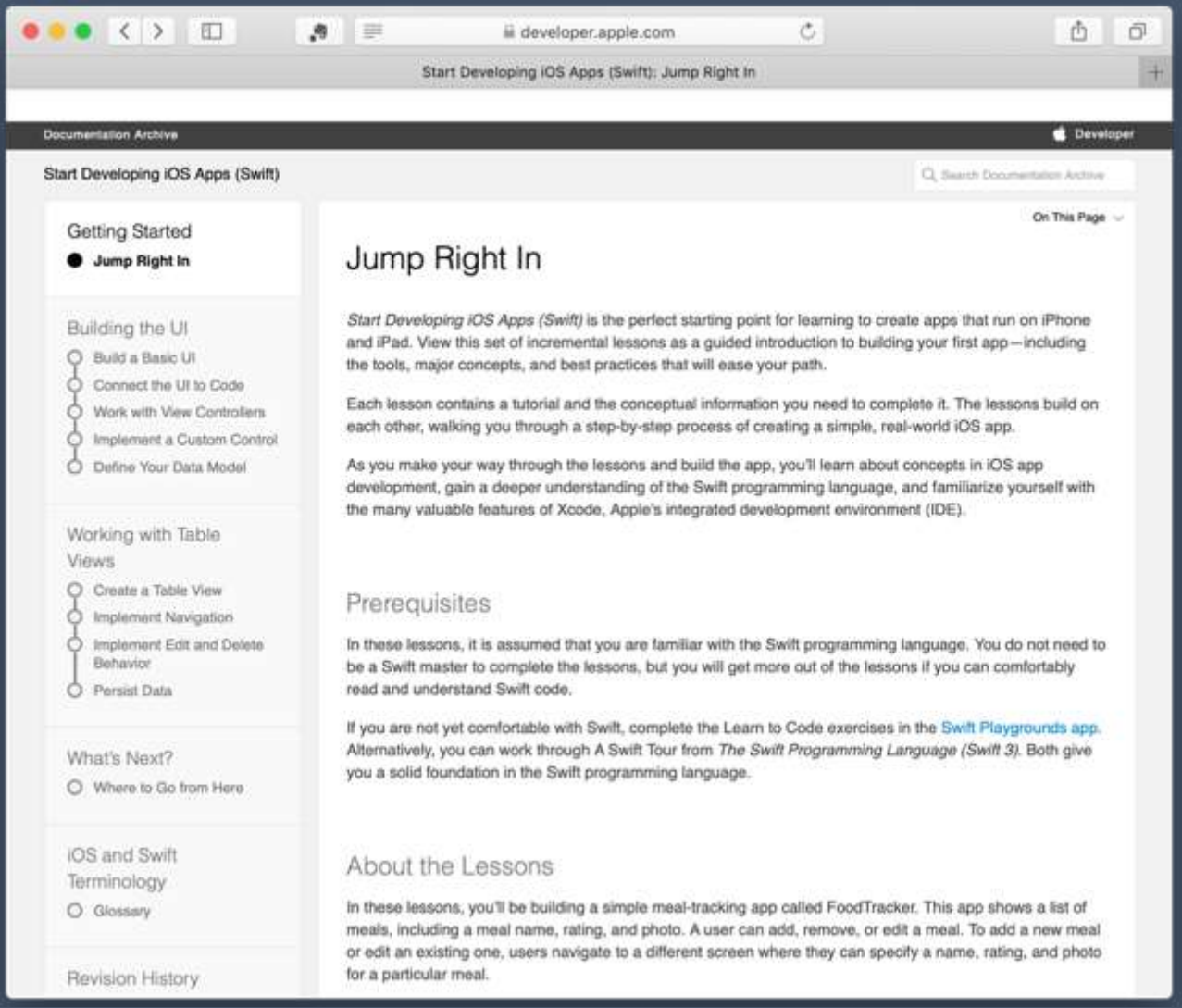


Table View



Collection View
（複数行表示・横スクロール）

Swift言語をざっくりマスターしたら Start Developing iOS Apps (Swift)



- 基本的なiOSアプリ開発のお作法
- Table Viewの作り方詳細
- 単体テストの自動化
- オブジェクトのパーシステント化（シリアライズ、iPhoneに保存、読込み、デシリアライズ）

英語ですが...

https://developer.apple.com/library/archive/referencelibrary/GettingStarted/DevelopiOSAppsSwift/

以下、日本語での解説

http://yataiblue.hatenablog.com/entry/2015/11/05/000000

Swift言語について

- Swift 無償教材
  - Swift Playgrounds iPadアプリ（日本語）
  - Apple Books: Swiftによるアプリケーション開発：入門編（日本語）
  - Apple Books: The Swift Programming Language（英語）
- PHP, JavaScript, Pythonな方
  - Swiftは**強い**静的型付け言語です
  - 型推論も行いますが、型の曖昧さがあるとコンパイルできません
- Java, C++な方
  - オブジェクト指向言語ではクラスのインスタンスはオブジェクト
  - Swiftでは、すべてのインスタンスはオブジェクト

Swift言語の特徴

Swift言語について
Int型の定数・変数はオブジェクト

```
extension Int {
    func whoami() {print("I am", self.description)}
}
let c = 5
c.whoami()
var v = 10
v = v * c
v.whoami()
```

[実行結果]

```
I am 5
I am 50
```

- Intは拡張してメソッドを追加できる
- Intにはdescriptionというプロパティがある
- したがってIntのインスタンスはオブジェクト

Intの正体はstructです

```
public struct Int : FixedWidthInteger, SignedInteger {
                    …
}
```

Swift言語について
３つの基本型

- Swift言語の３つの基本型
  - class　　参照型
  - struct　　値型
  - enum　　値型
- いずれも
  - イニシャライザ
  - メソッド
  - プロパティ

  を持てる。
- いずれのインスタンスもオブジェクトである。
- Swiftでは class ≒ struct。違いは、
  - classは参照型、structは値型
  - classは継承できる、structは継承できない

Swift言語について
Class(参照型) vs Struct(値型) func()

```swift
class Song {
    var title: String
    var artist: String
    var artistSort: String
    init(title:String, artist: String) {
        self.title = title
        self.artist = artist
        self.artistSort = artist
    }
    func setArtistSort(artistSort: String) {
        self.artistSort = artistSort
    }
}
let mySong = Song(title: "もし君を許せたら", artist: "家入レオ")
mySong.setArtistSort(artistSort: "いえいりれお")
print("Artist Sort = ", mySong.artistSort)
```

[出力結果]

```
Artist Sort = いえいりれお
```

```swift
struct Song {
    var title: String
    var artist: String
    var artistSort: String
    init(title:String, artist: String) {
        self.title = title
        self.artist = artist
        self.artistSort = artist
    }
    mutating func setArtistSort(artistSort: String) {
        self.artistSort = artistSort
    }
}
var mySong = Song(title: "もし君を許せたら", artist: "家入レオ")
mySong.setArtistSort(artistSort: "いえいりれお")
print("Artist Sort = ", mySong.artistSort)
```

[出力結果]

```
Artist Sort = いえいりれお
```

Swift言語について
Class(参照型) vs Struct(値型) 代入

```swift
class Song {
    var title: String
    var artist: String
    var artistSort: String
    init(title:String, artist: String) {
        self.title = title
        self.artist = artist
        self.artistSort = artist
    }
}

let song1 = Song(title: "もし君を許せたら", artist: "家入レオ")
let song2 = song1
song2.artistSort = "いえいりれお"
print("song1.artistSort = ", song1.artistSort)
print("song2.artistSort = ", song2.artistSort)
```

[出力結果]

```
song1.artistSort =  いえいりれお
song2.artistSort =  いえいりれお
```

```swift
struct Song {
    var title: String
    var artist: String
    var artistSort: String
    init(title:String, artist: String) {
        self.title = title
        self.artist = artist
        self.artistSort = artist
    }
}

let song1 = Song(title: "もし君を許せたら", artist: "家入レオ")
var song2 = song1
song2.artistSort = "いえいりれお"
print("song1.artistSort = ", song1.artistSort)
print("song2.artistSort = ", song2.artistSort)
```

[出力結果]

```
song1.artistSort =  家入レオ
song2.artistSort =  いえいりれお
```

Swift言語について
Class(参照型) vs Struct(値型) 引数として

```swift
class Song {
    var title: String
    var artist: String
    var artistSort: String
    init(title:String, artist: String) {
        self.title = title
        self.artist = artist
        self.artistSort = artist
    }
}
func setArtistSort(song: Song, artistSort: String) {
    song.artistSort = artistSort
}
let mySong = Song(title: "もし君を許せたら", artist: "家入レオ")
setArtistSort(song: mySong, artistSort: "いえいりれお")
print("Artist Sort = ", mySong.artistSort)
```

[出力結果]

Artist Sort = いえいりれお

```swift
struct Song {
    var title: String
    var artist: String
    var artistSort: String
    init(title:String, artist: String) {
        self.title = title
        self.artist = artist
        self.artistSort = artist
    }
}
func setArtistSort(song: Song, artistSort: String) {
    var song = song
    song.artistSort = artistSort
}
var mySong = Song(title: "もし君を許せたら", artist: "家入レオ")
setArtistSort(song: mySong, artistSort: "いえいりれお")
print("Artist Sort = ", mySong.artistSort)
```

[出力結果]

Artist Sort = 家入レオ

Swift言語について
Class(参照型) vs Struct(値型)の参照渡し

```
class Song {
    var title: String
    var artist: String
    var artistSort: String
    init(title:String, artist: String) {
        self.title = title
        self.artist = artist
        self.artistSort = artist
    }
}
func setArtistSort(song: Song, artistSort: String) {
    song.artistSort = artistSort
}
let mySong = Song(title: "もし君を許せたら", artist: "家入レオ")
setArtistSort(song: mySong, artistSort: "いえいりれお")
print("Artist Sort = ", mySong.artistSort)
```

[出力結果]

```
Artist Sort =  いえいりれお
```

```
struct Song {
    var title: String
    var artist: String
    var artistSort: String
    init(title:String, artist: String) {
        self.title = title
        self.artist = artist
        self.artistSort = artist
    }
}
func setArtistSort(song: inout Song, artistSort: String) {
    song.artistSort = artistSort
}
var mySong = Song(title: "もし君を許せたら", artist: "家入レオ")
setArtistSort(song: &mySong, artistSort: "いえいりれお")
print("Artist Sort = ", mySong.artistSort)
```

[出力結果]

```
Artist Sort = いえいりれお
```

Swift言語について
Struct 値型の参照渡し vs 値型を戻す

```
func setArtistSort(song: inout Song, artistSort: String) {
    song.artistSort = artistSort
}
var mySong = Song(title: "もし君を許せたら", artist: "家入レオ")
setArtistSort(song: &mySong, artistSort: "いえいりれお")
print("Artist Sort = ", mySong.artistSort)
```

[出力結果]

```
Artist Sort = いえいりれお
```

```
func setArtistSort(song: Song, artistSort: String) -> Song {
    var song = song
    song.artistSort = artistSort
    return song
}
var mySong = Song(title: "もし君を許せたら", artist: "家入レオ")
mySong = setArtistSort(song: mySong, artistSort: "いえいりれお")
print("Artist Sort = ", mySong.artistSort)
```

[出力結果]

```
Artist Sort = いえいりれお
```

WWDC19 “Modern Swift API Design”
Prefer structs over classes

- Only choose classes when reference semantics are important

https://developer.apple.com/videos/play/wwdc2019/415/

Swift言語について
Optional型：nilかも知れないオブジェクト

```
func printTrackNum(trackNum: Int?) {
    if let i:Int = trackNum {
        print("Track Number is", i)
    } else {
        print("Track Number is nil")
    }
}

var track1,track2 :Int?
track1 = Int("10")
printTrackNum(trackNum: track1)
track2 = Int("a")
printTrackNum(trackNum: track2)
```

[出力結果]

```
Track Number is 10
Track Number is nil
```

```
func printTrackNum(trackNum: Int?) {
    guard trackNum != nil else {
        print(“Track Number is nil”)
        return
    }
    print("Track Number is", trackNum!)
}

var track1,track2 :Int?
track1 = Int("10")
printTrackNum(trackNum: track1)
track2 = Int("a")
printTrackNum(trackNum: track2)
```

[出力結果]

```
Track Number is 10
Track Number is nil
```

```
var track1,track2 :Int
track1 = Int("10") ?? 0
print("Track Number = ", track1)
track2 = Int("a") ?? 0
print("Track Number = ", track2)
```

[出力結果]

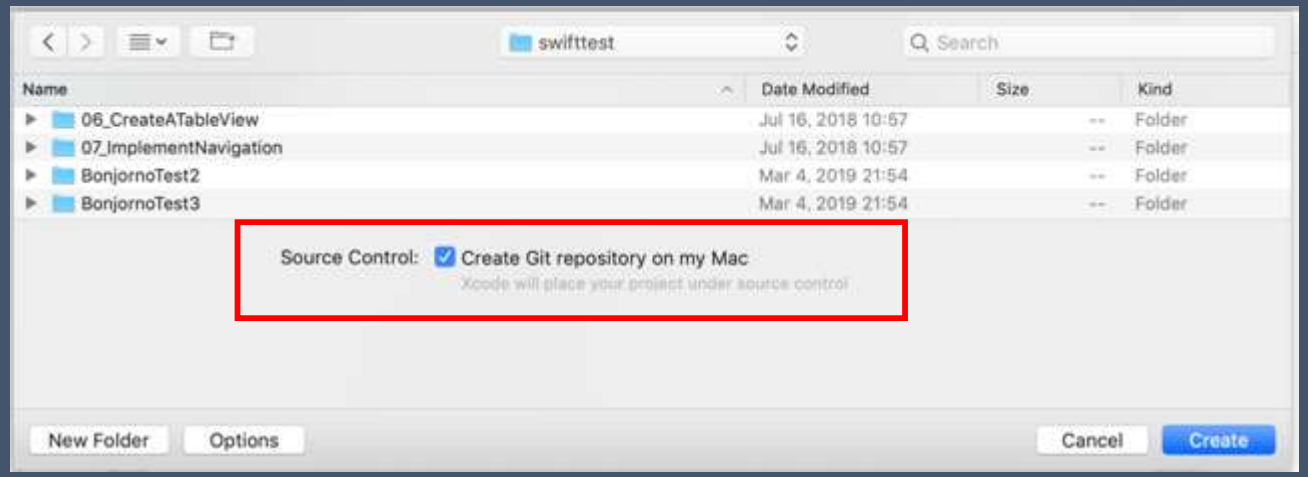
```
Track Number is 10
Track Number is 0
```

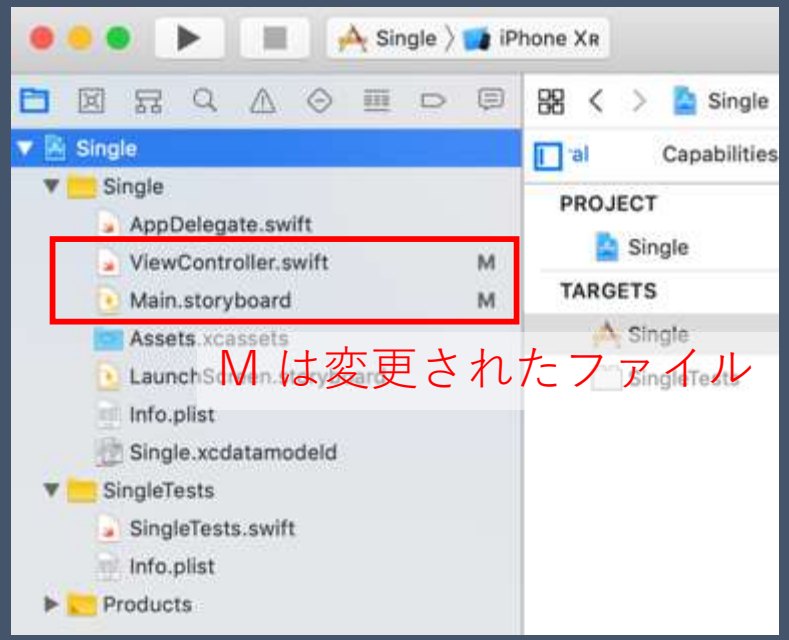
GitとGitHub

Xcodeによるソースコード管理

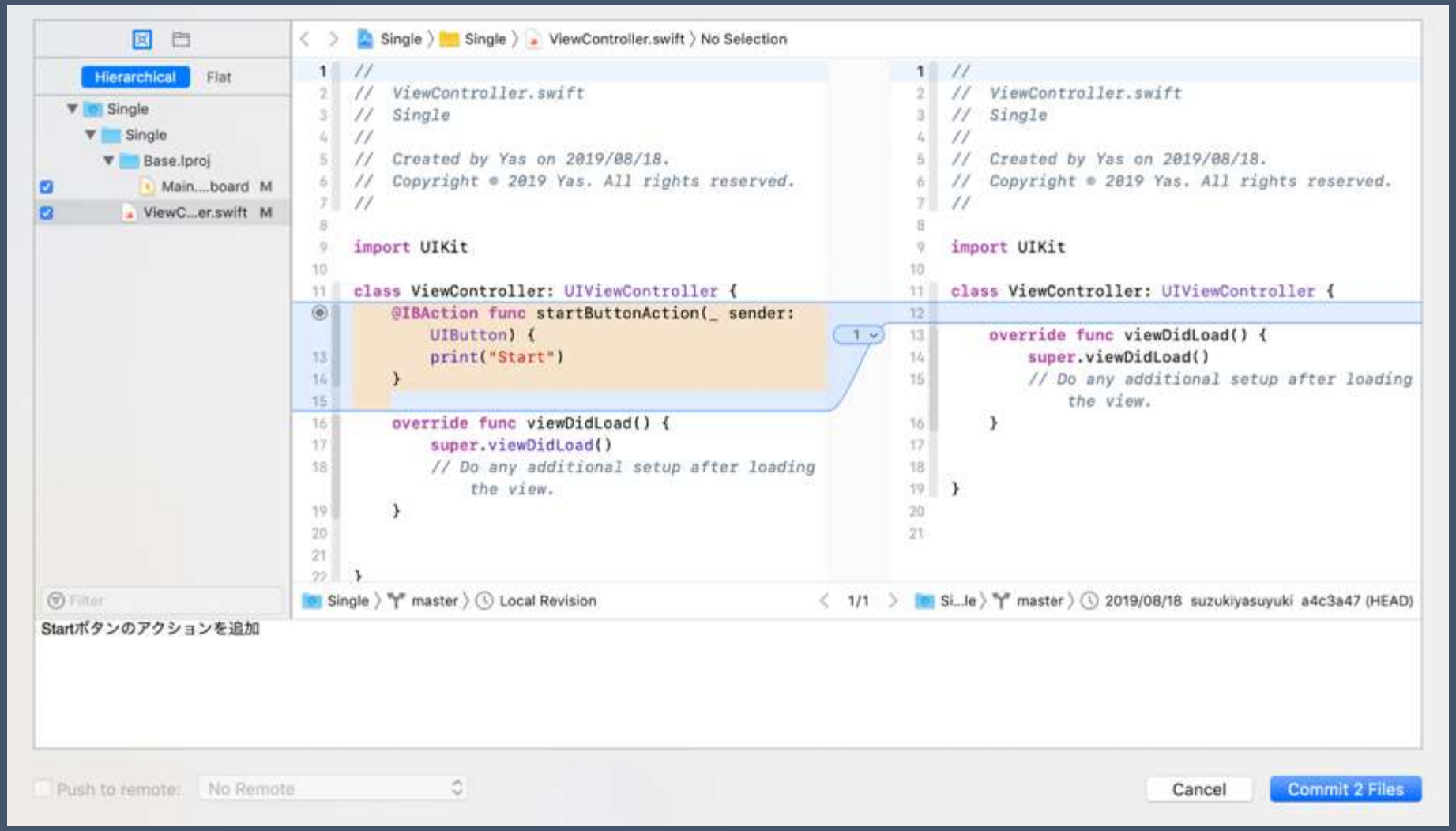
Git

- Xcodeでプロジェクト作成時に



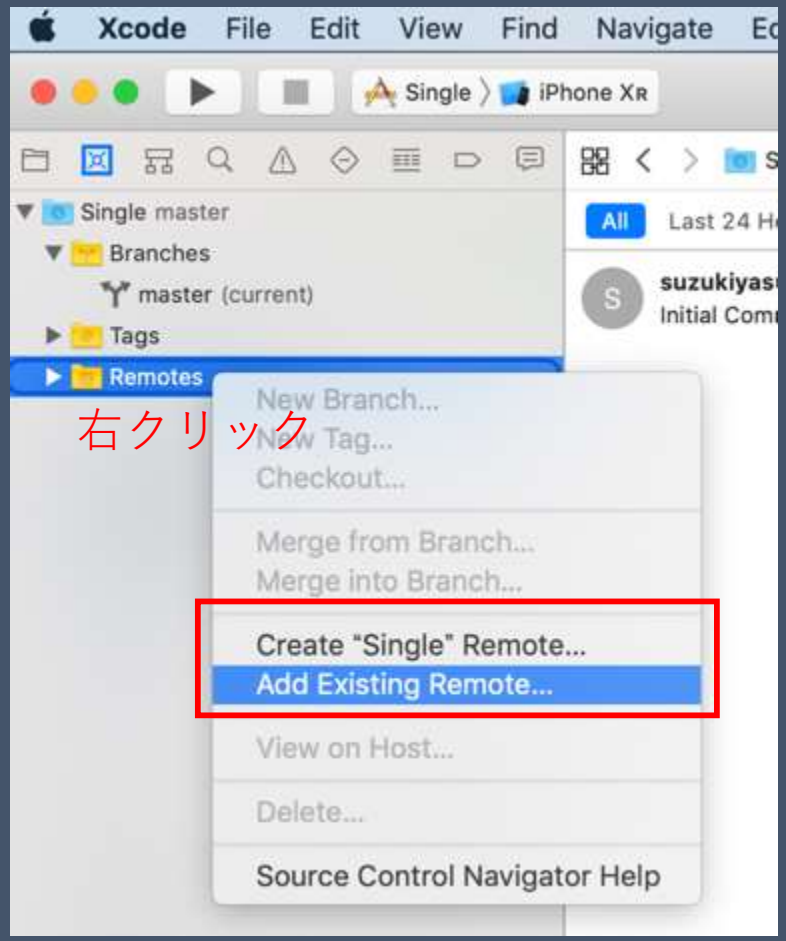


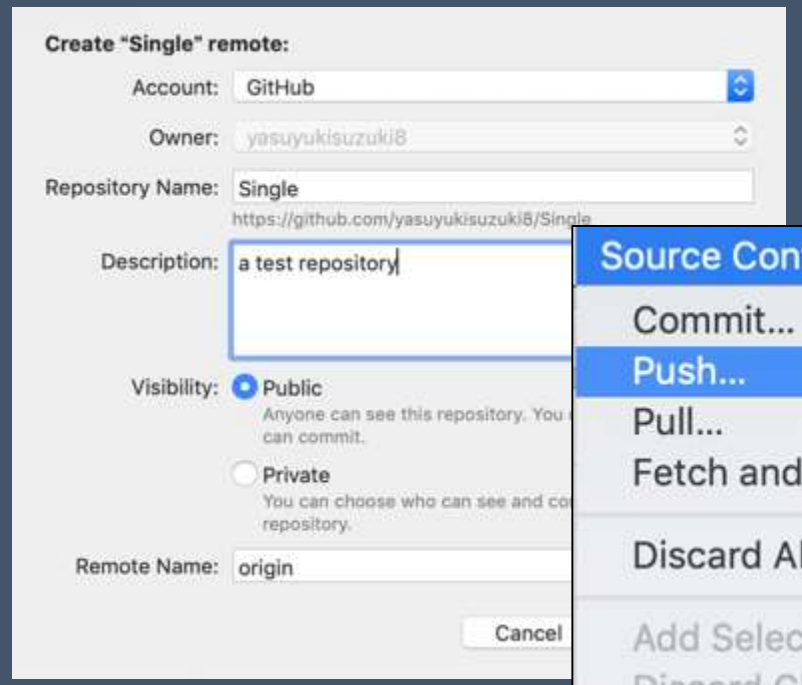
Git
変更箇所の確認、変更コメントの記入、Commit

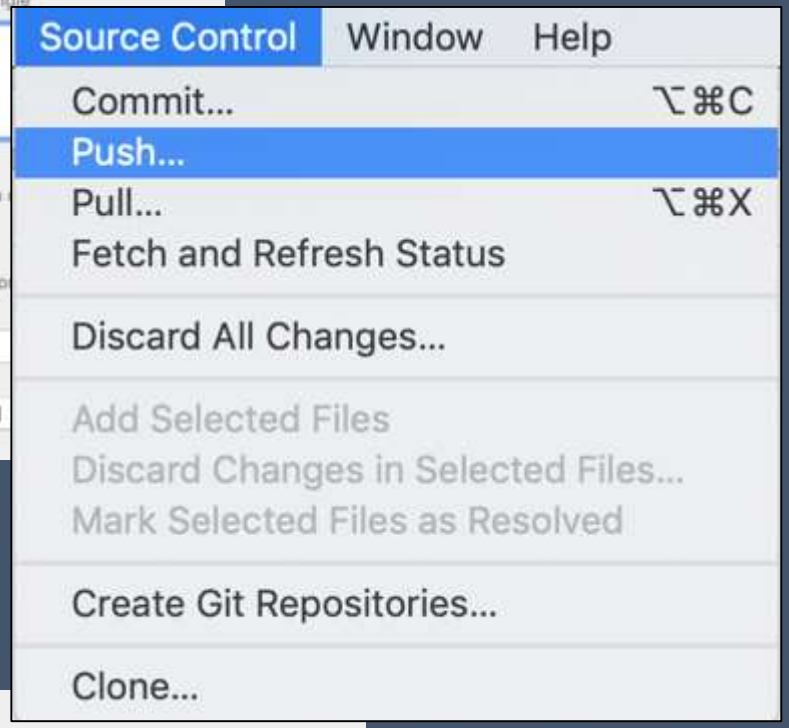


GitHub

Create “プロジェクト名” Remote... でGitHubリポジトリを作成







GitHubでリポジトリを作成済なら、
Add Existing Remote... でリポジト

デバッグ

Xcodeによるデバッグ

Xcode デバッグ基本

プログラム実行継続
Step over (1行実行)
Step into (関数に入る)
Step out (関数から出る)

変数・定数マウスオーバーで値を表示

行番号クリックで
ブレークポント設定

実行中の行 ⟶

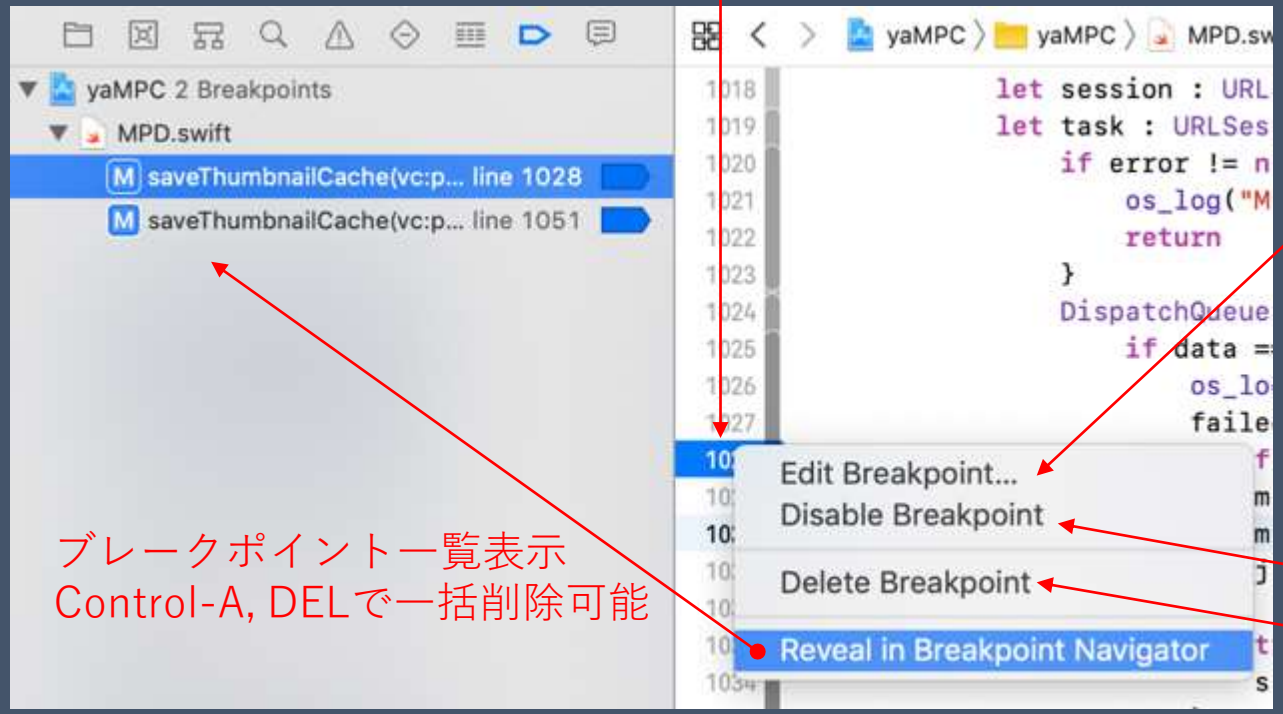
オブジェクトの値

os_log(), print()によるデバッグ出力

ブレークポイントの設定

指定回数だけ無視、ログ出力などの詳細設定

ブレークポイントをクリックで
ON/OFFを切り替え



ブレークポイント一覧表示
Control-A, DELで一括削除可能

ブレークポイントをOFF
ブレークポントを削除

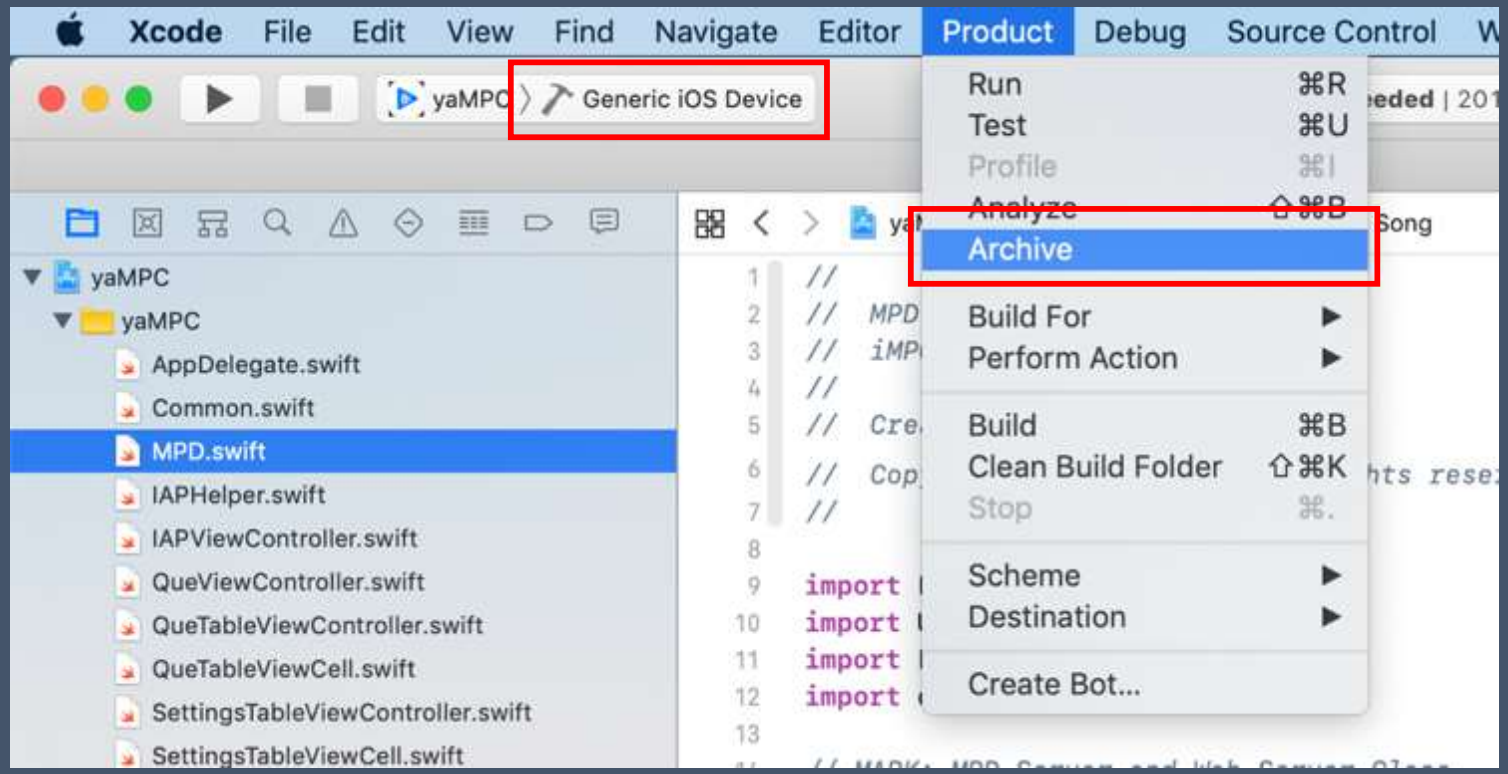
ブレークポイントを右クリックでメニュー

Beta Test

TestFlight

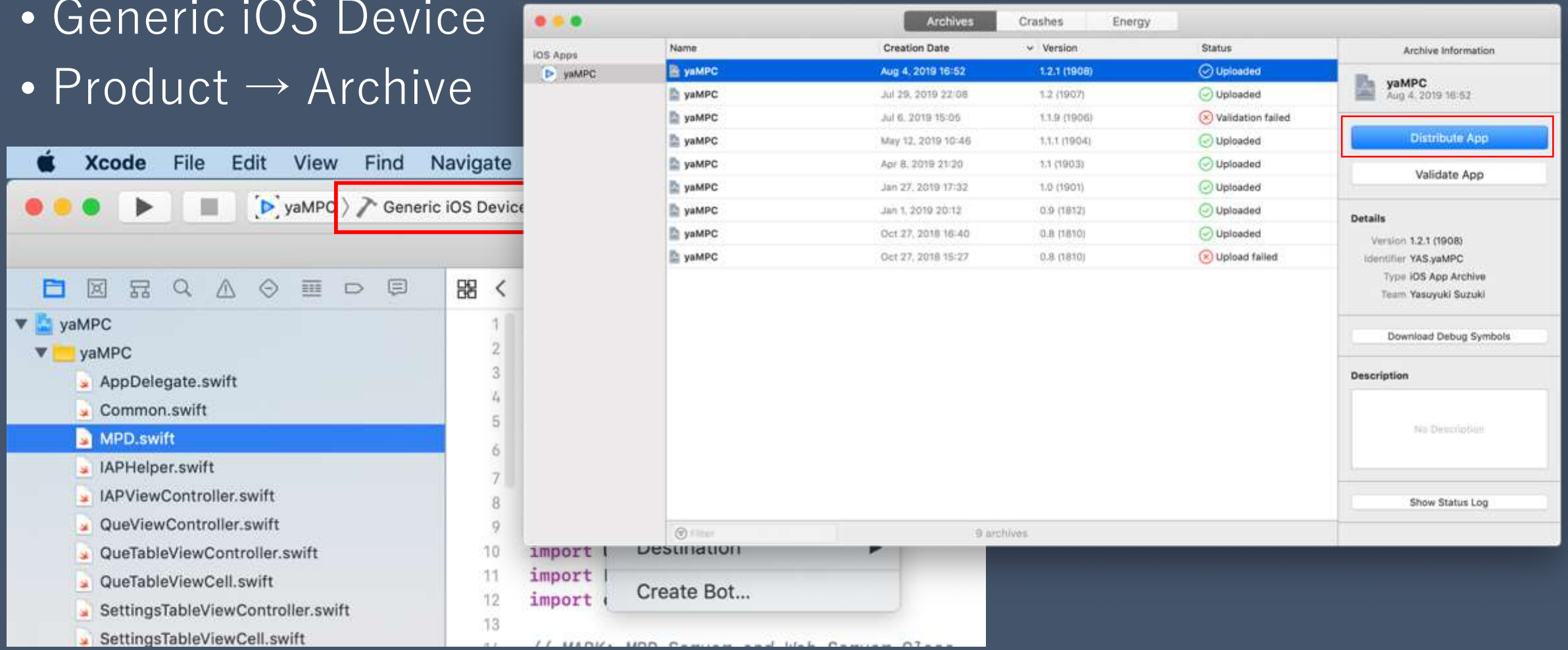
App Store Connectへのアップロード

- Generic iOS Device
- Product → Archive



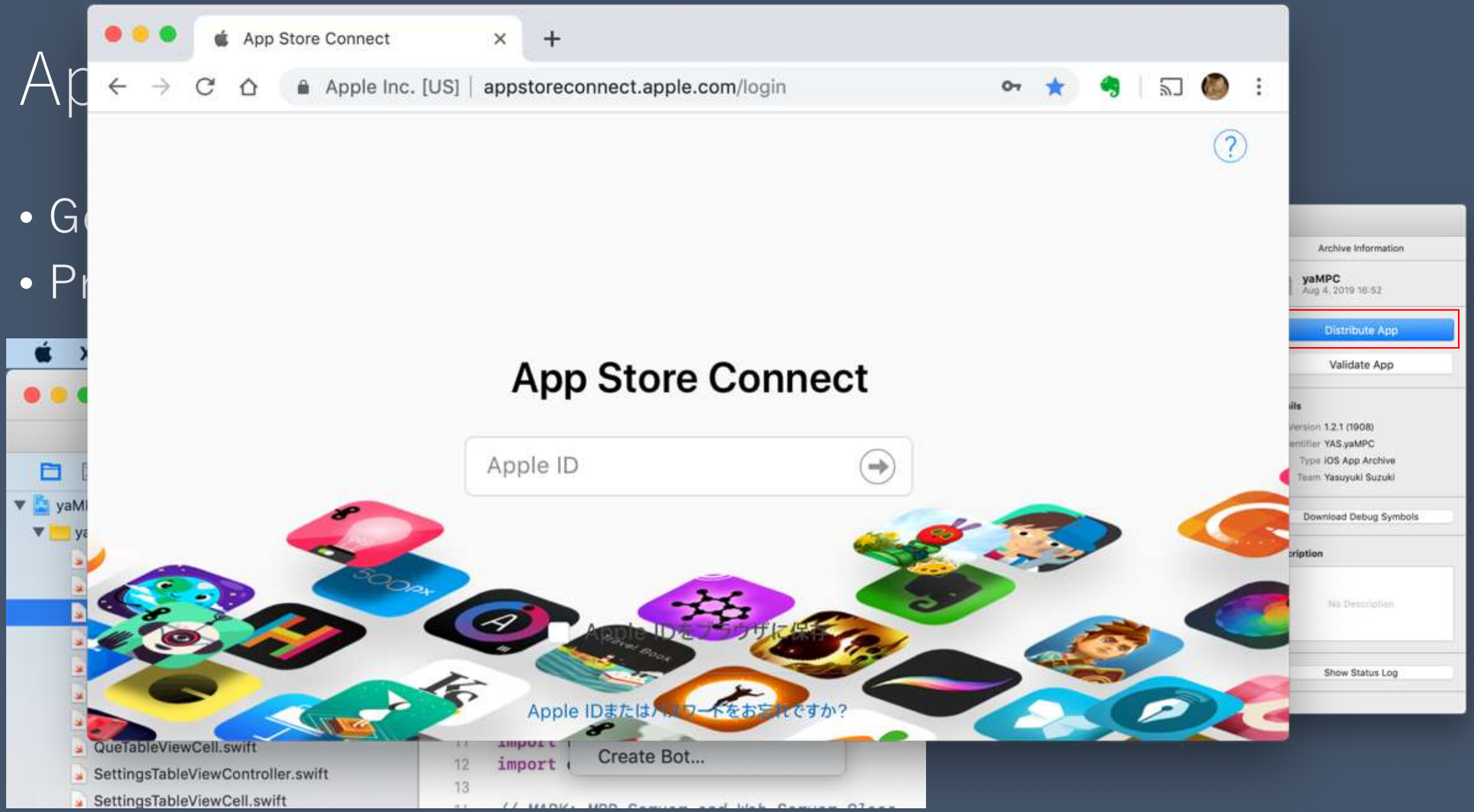
App Store Connectへのアップロード

- Generic iOS Device
- Product → Archive



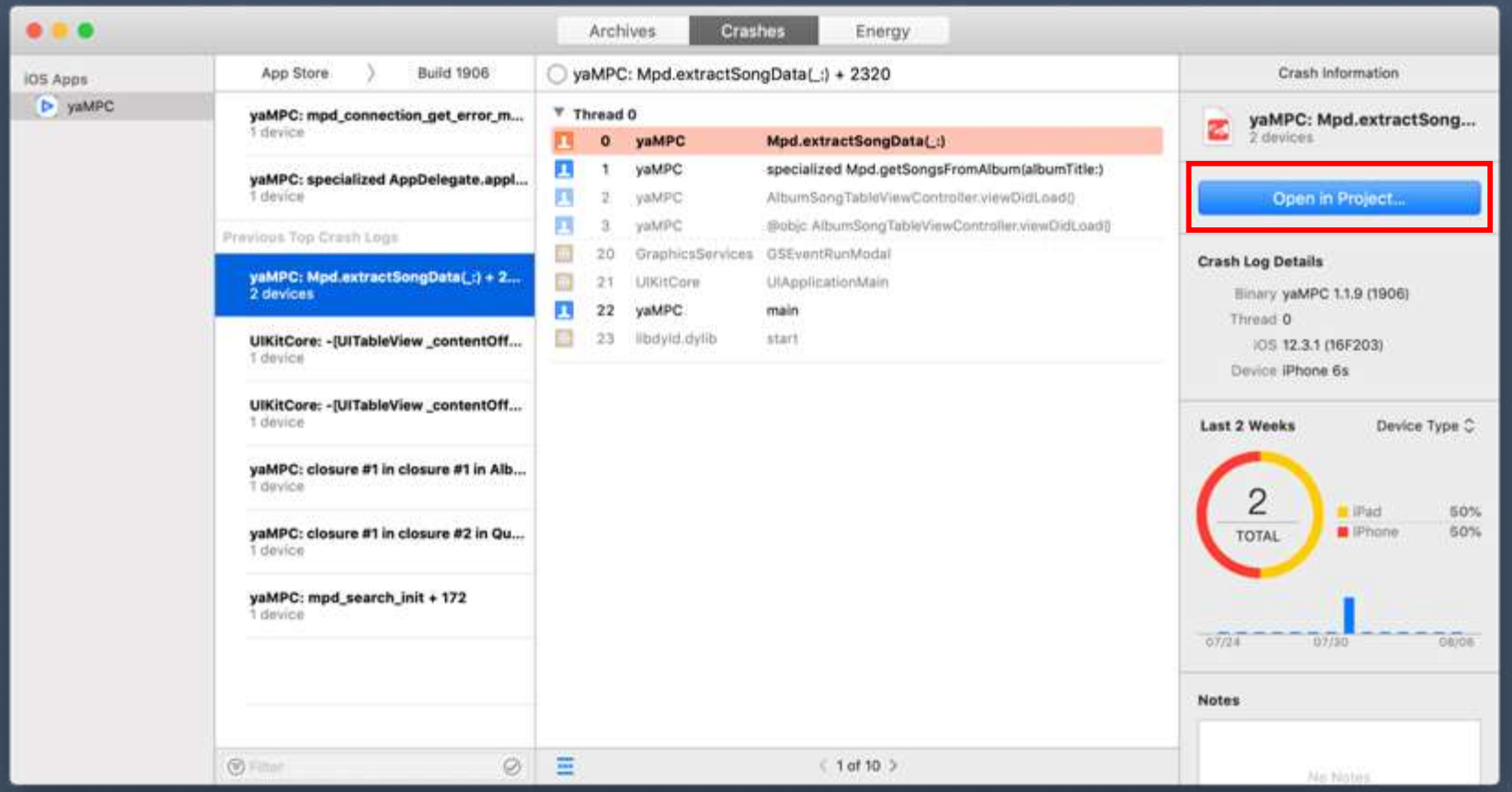
Ap

- G
- Pr


App Store Connect
Apple Inc. [US] | appstoreconnect.apple.com/login
App Store Connect
Apple ID
Apple IDまたはパスワードをお忘れですか?
Archive Information
yaMPC
Aug 4, 2019 16:52
Distribute App
Validate App
Version 1.2.1 (1908)
Identifier YAS.yaMPC
Type iOS App Archive
Team Yasuyuki Suzuki
Download Debug Symbols
Show Status Log
QueTableViewCell.swift
SettingsTableViewController.swift
SettingsTableViewCell.swift
Create Bot...

Betaテスターのクラッシュ解析

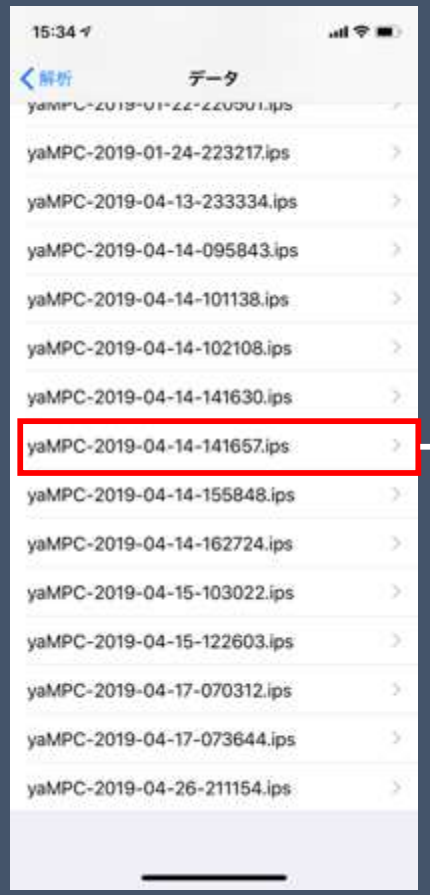
- Window → Organizer → Crashesタブ



クラッシュ箇所の
ソースコードを表示

一般ユーザーからのクラッシュログ

- iPhone 設定 → プライバシー → 解析 → 解析データ



メール等で送信可能だが…

意味不明

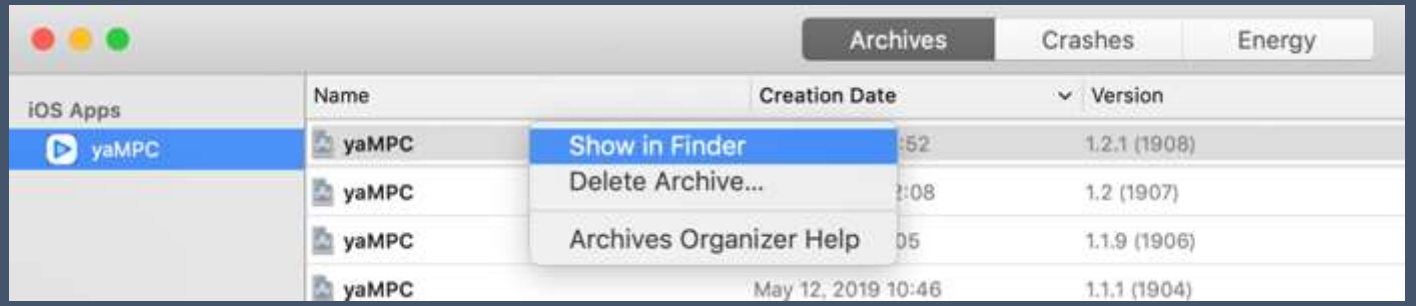
```
Exception Type:  EXC_BREAKPOINT (SIGTRAP)
Exception Codes: 0x0000000000000001, 0x0000000101065740
Termination Signal: Trace/BPT trap: 5
Termination Reason: Namespace SIGNAL, Code 0x5
Terminating Process: exc handler [262]
Triggered by Thread:  0


Thread 0 name:  Dispatch queue: com.apple.main-thread
Thread 0 Crashed:
0   yaMPC                         0x0000000101065740 0x101040000 + 153408
1   yaMPC                         0x0000000101065b0c 0x101040000 + 154380
2   yaMPC                         0x00000001010838a8 0x101040000 + 276648
3   yaMPC                         0x0000000101087368 0x101040000 + 291688
4   yaMPC                         0x0000000101083400 0x101040000 + 275456
5   UIKitCore                     0x00000001e6c9a7c4 0x1e6982000 + 3246020
6   UIKitCore                     0x00000001e6bfbe90 0x1e6982000 + 2596496
7   UIKitCore                     0x00000001e6c9a7c4 0x1e6982000 + 3246020
8   UIKitCore                     0x00000001e6bd1714 0x1e6982000 + 2422548
9   UIKitCore                     0x00000001e6c9a7c4 0x1e6982000 + 3246020
```

クラッシュログの “Symbolicate”

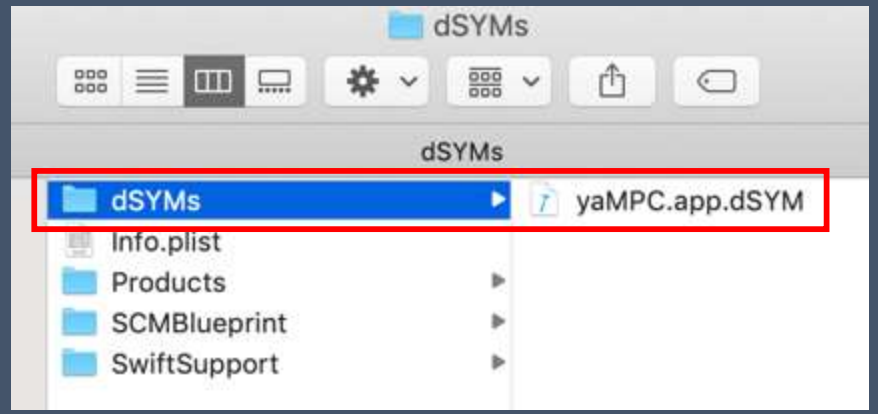
- 必要なもの
  - クラッシュログ（ipsファイル）
  - ビルドの際のシンボル情報（dSYM）
  - symbolicatecrashプログラム
  - Xcode Developer path: /Applications/Xcode.app/Contents/Developer

dSYMの入手方法

- Xcode Window → Organizer
- Archivesタブで対象バージョンを右クリック → Show in Finder



- .xarchiveファイルを右クリック→ パッケージの内容を表示



クラッシュログの “Symbolicate”方法

```
$ ls
yaMPC-2019-07-11-221332.ips yaMPC.app.dSYM
$ export DEVELOPER_DIR="/Applications/Xcode.app/Contents/Developer”
$ PATH=$PATH:/Applications/Xcode.app/Contents/SharedFrameworks/DVTFoundation.framework/Versions/A/Resources
$ symbolicatecrash -v yaMPC-2019-07-11-221332.ips yaMPC.app.dSYM > symbolicated.txt
$ ls
symbolicated.txt            yaMPC.app.dSYM
yaMPC-2019-07-11-221332.ips
```

Crash箇所判明

MPD.swift 627行目　Mpd.extractSongData()にてクラッシュ

```
Exception Type:  EXC_BREAKPOINT (SIGTRAP)
Exception Codes: 0x0000000000000001, 0x0000000100bca894
Termination Signal: Trace/BPT trap: 5
Termination Reason: Namespace SIGNAL, Code 0x5
Terminating Process: exc handler [237]
Triggered by Thread:  0


Thread 0 name:  Dispatch queue: com.apple.main-thread
Thread 0 Crashed:
0   yaMPC                           0x0000000100bca894 Mpd.extractSongData(_:) + 174228 (MPD.swift:627)
1   yaMPC                           0x0000000100bcac40 Mpd.getQue() + 175168 (MPD.swift:653)
2   yaMPC                           0x0000000100be9420 QueViewController.updateView() + 300064 (<compiler-generated>:0)
```

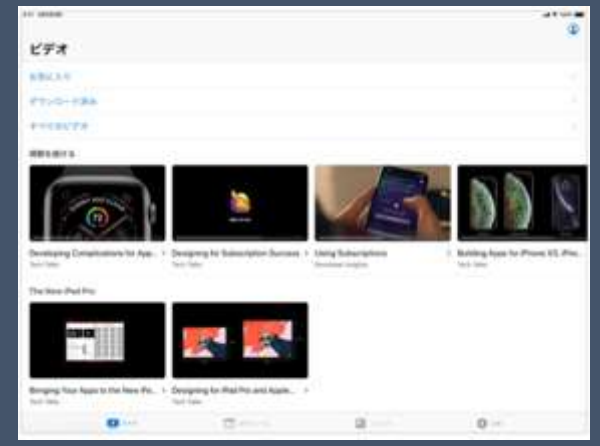
```
626
627         let song = Song(id: id, track: Int(trackNumber)!, uri: uri, title: title, artist: artist,
                albumArtist: albumArtist, album: albumTitle, composer: composer, genre: genre,
                thumbnail: thumbnail, time: Int(duration))
628
629         return song
630     }
```

WWDC19 トピックス

- iPad Apps for Mac
- SwiftUI
- SF Symbols
- Dark Mode

WWDC19のセッション動画
多くの動画に日本語字幕が付いています

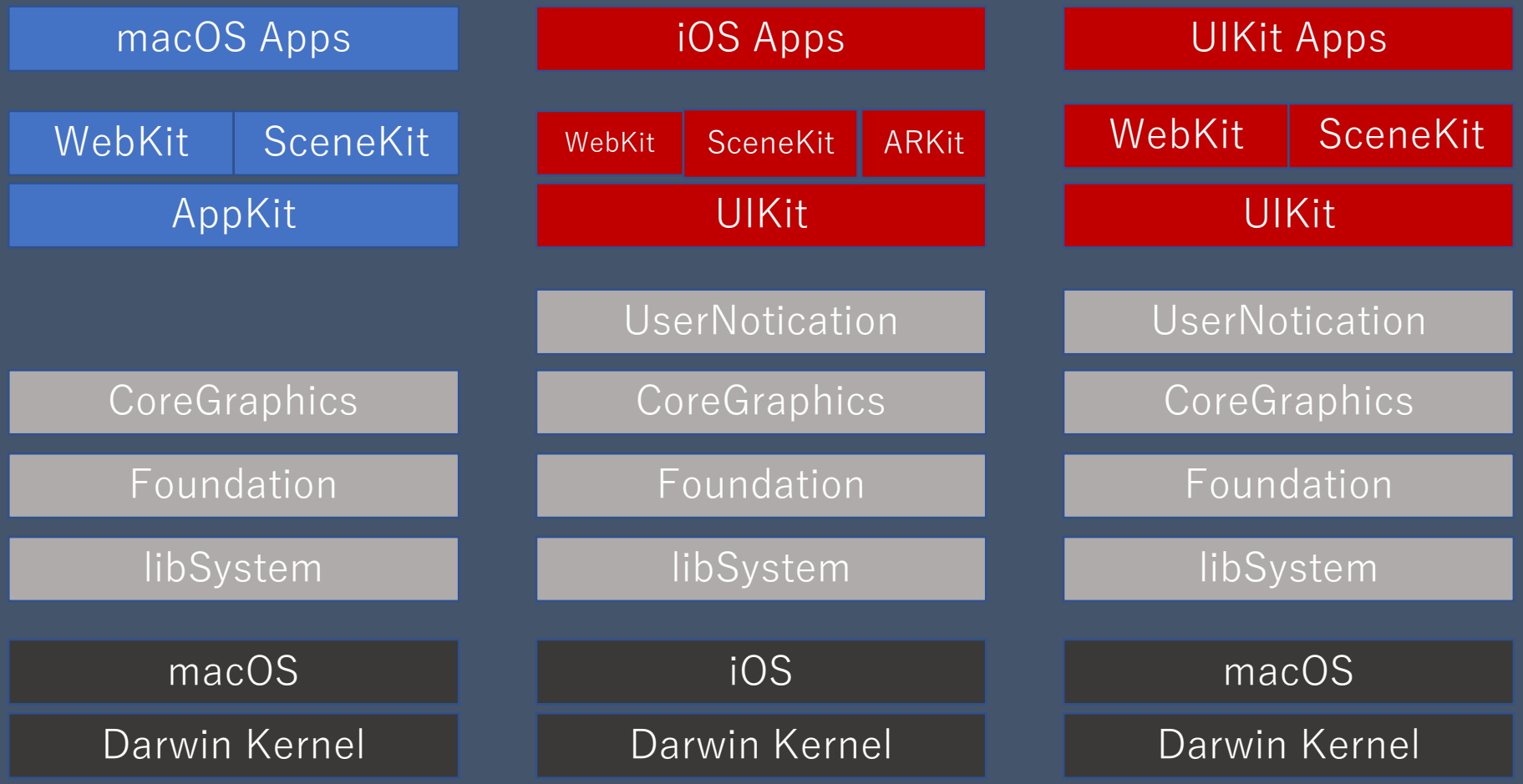
- iPad用　WWDCアプリ
- WWDC19 Session Videos
  https://developer.apple.com/videos/wwdc2019/
  - iPad Apps for Mac
    https://developer.apple.com/videos/play/wwdc2019/205/
  - SwiftUI
    https://developer.apple.com/videos/play/wwdc2019/204/
  - SF Symbols
    https://developer.apple.com/videos/play/wwdc2019/206/
  - Dark Mode
    https://developer.apple.com/videos/play/wwdc2019/214/



iPad WWDC App

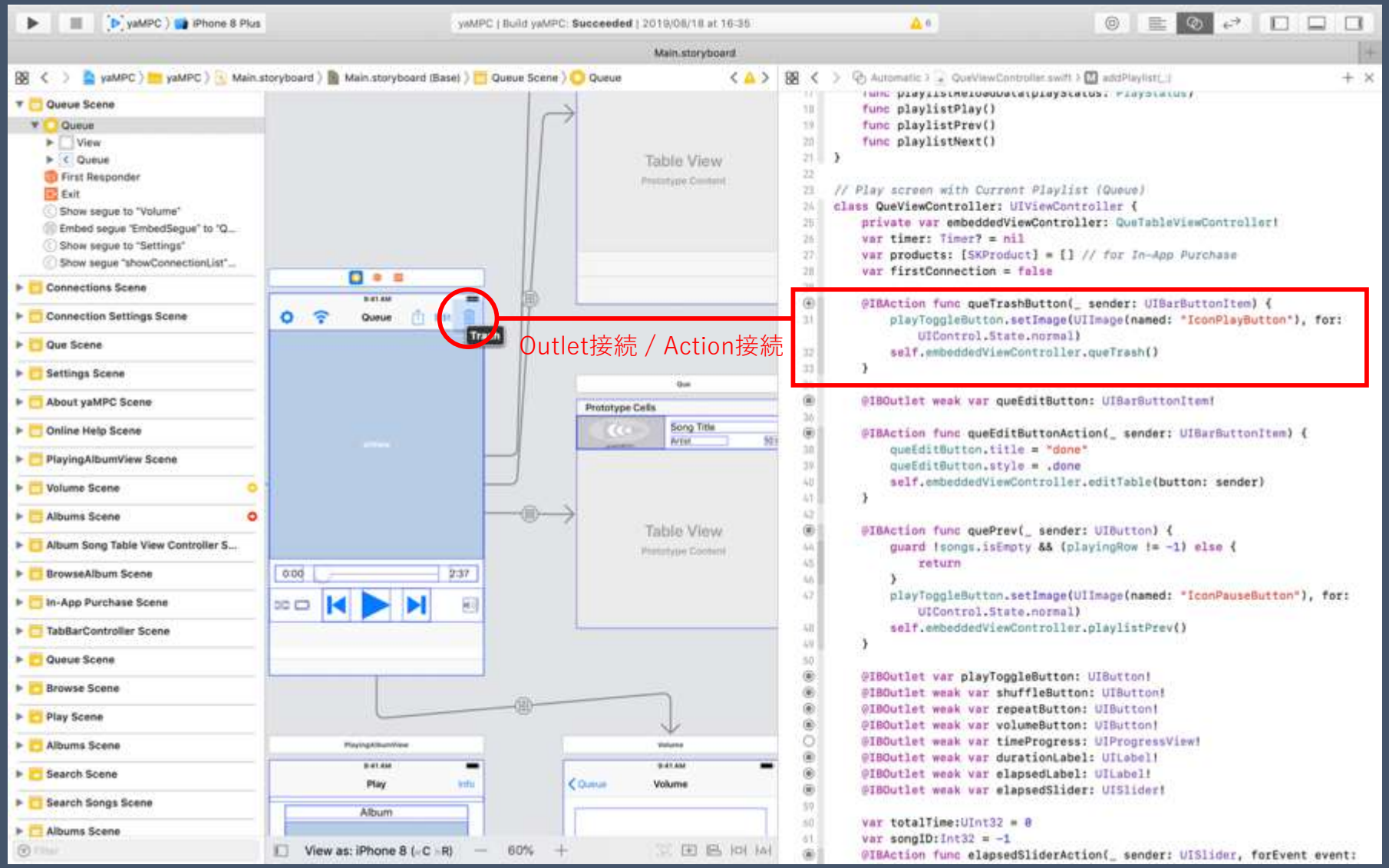
iPad App for Mac

https://developer.apple.com/videos/play/wwdc2019/205/



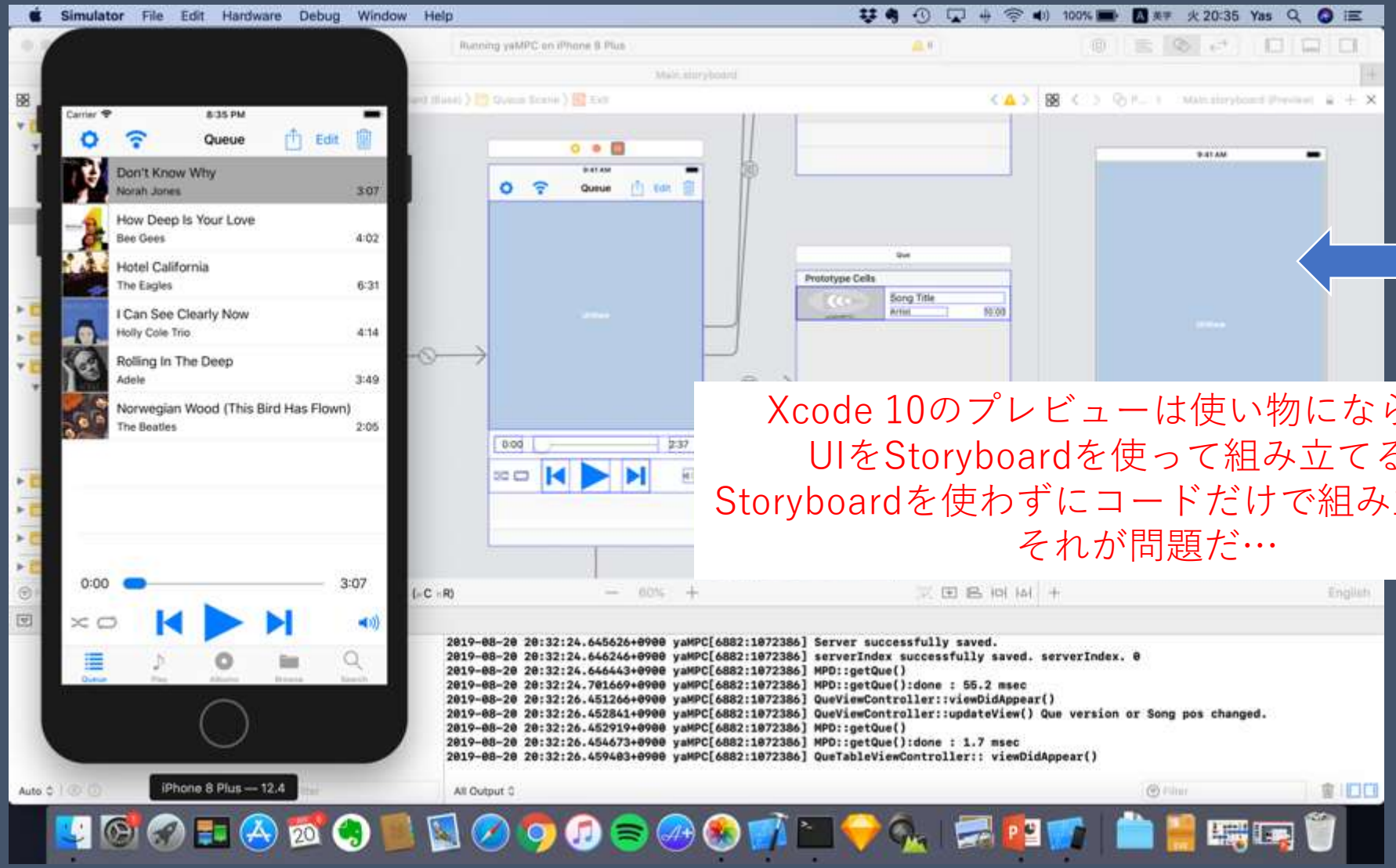
- UIKitをmacOS上でサポートすることでiPad AppをMacに移植しやすくするもの
- ARKitなどmacOSでサポートされないものもある
- StoreKitなどmacOSとiOSで互換性がないものもある

Xcode 10 Storyboard



Xcode 10 Preview & iPhone Simulator

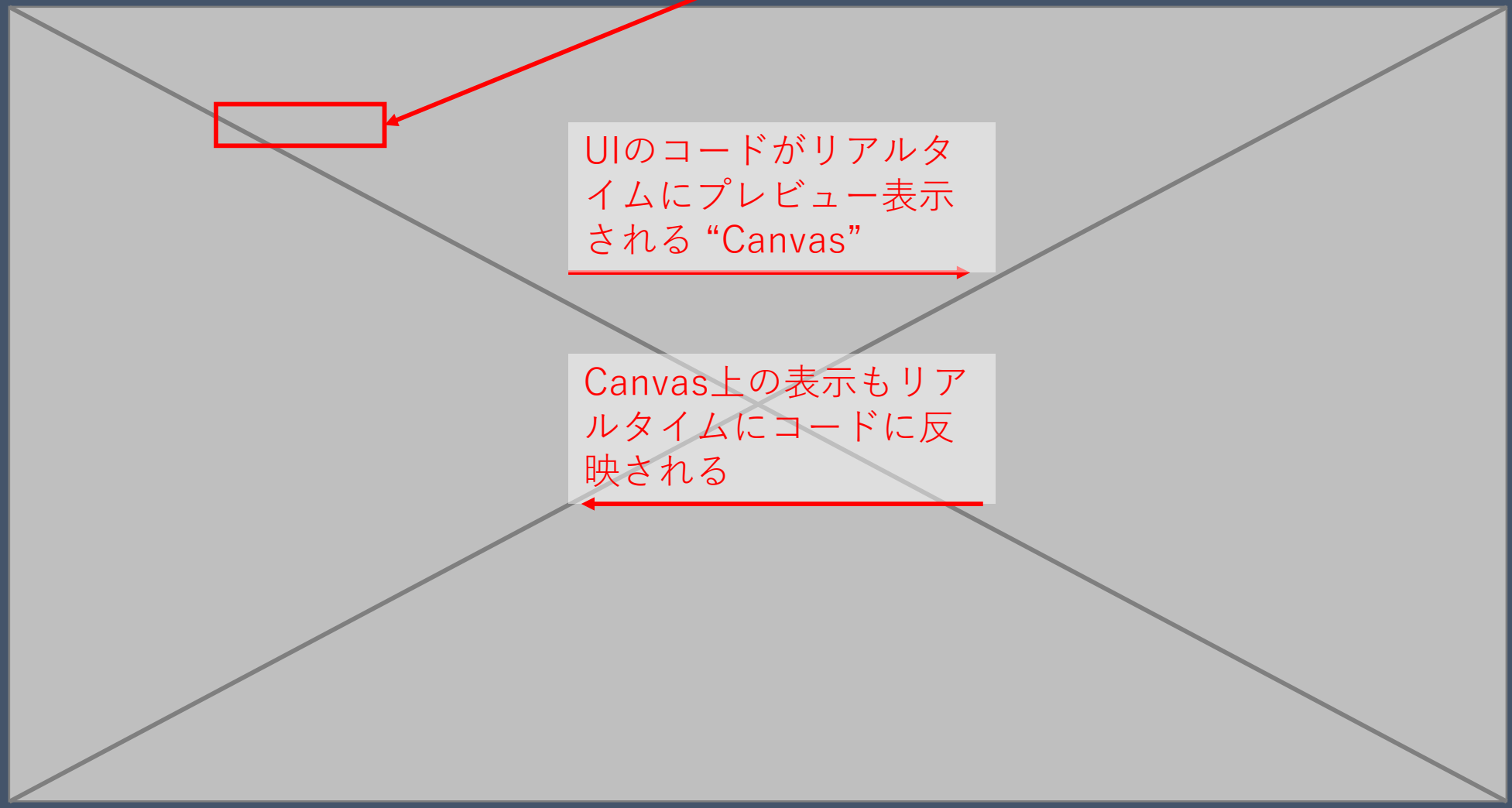
正確にチェックしたければ、iPhoneシミュレーター



限定的なプレビュー表示

Xcode 10のプレビューは使い物にならない。
UIをStoryboardを使って組み立てるか？
Storyboardを使わずにコードだけで組み立てるか？
それが問題だ…

Xcode 11 SwiftUI



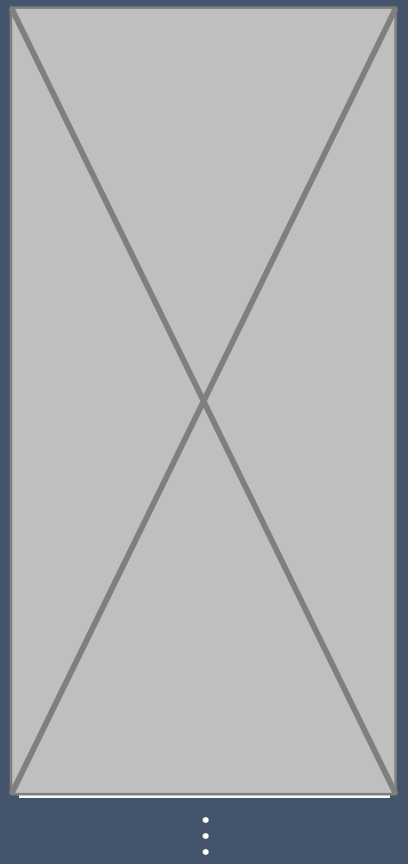
https://developer.apple.com/videos/play/wwdc2019/204/

Xcode 11 SwiftUIの注意点

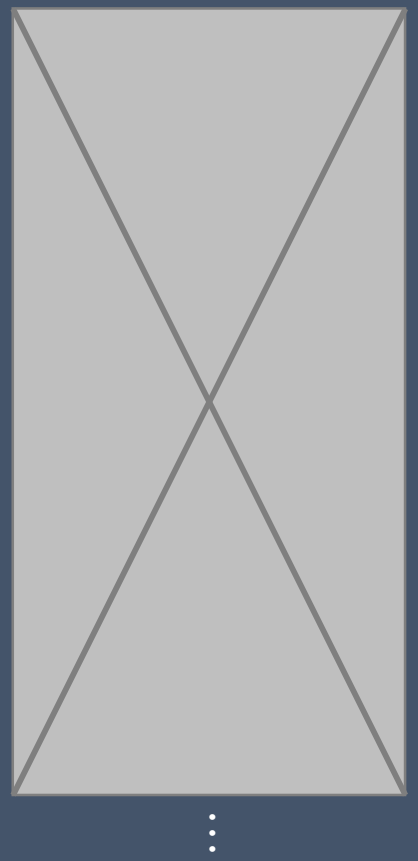
- SwiftUIは、現時点ではUIKitのUI部品をすべてサポートしているわけではない。
- SwiftUIとUIKitは混在できる。従来のアプリの一部画面にSwiftUIを使ったり、その逆もできる。
- SwiftUIはUIKitとは別物なので、UIKitをmacOSでサポートする“iPad Apps for Mac”は利用できない。
- SwiftUIで、iPhoneアプリ、iPadアプリ、Macアプリ、Apple Watchアプリ、Apple TVアプリを開発できる。
  しかし“Write once, run anywhere (WORA)”ではない。
  SwiftUIを使って、それぞれのデバイス毎に適したUIでアプリを開発する必要がある。

iOS System Icons
種類が少ないし、使える場所が限られる

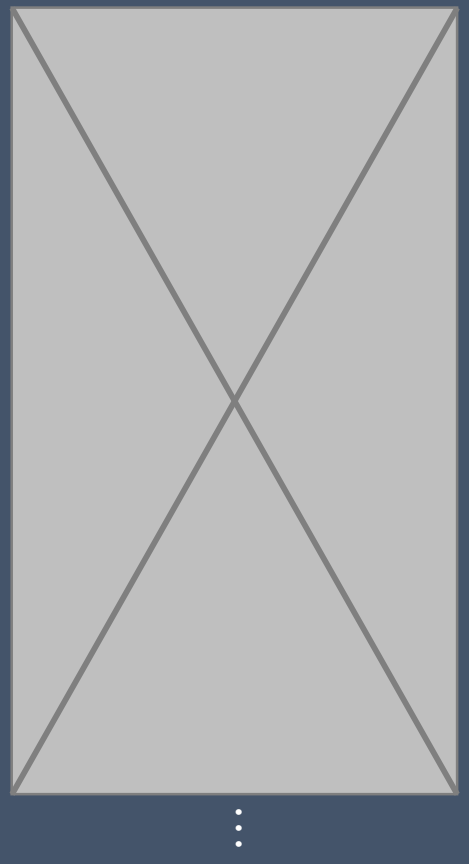
Navigation Bar
Toolbar



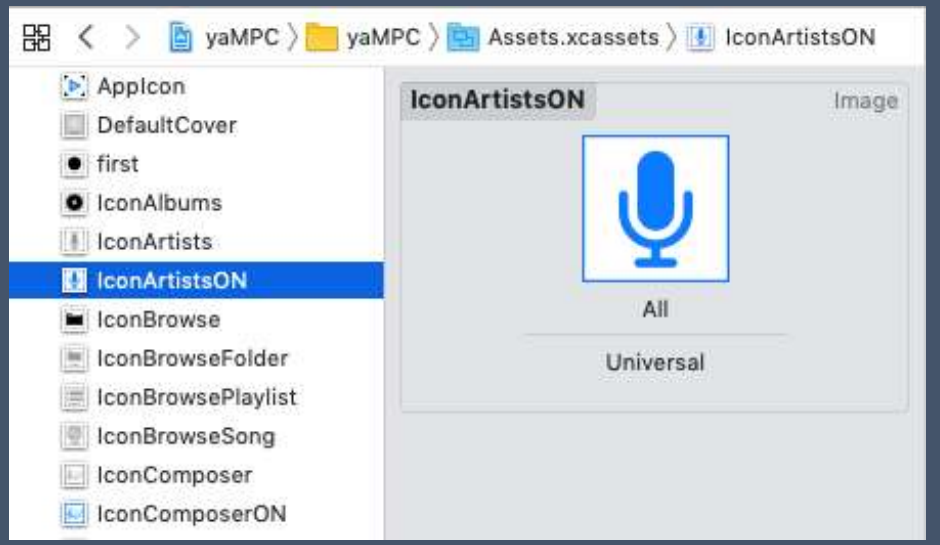
Tab Bar



Home Screen
Quick Action



適当なアイコンがないと
自分で作らないといけない

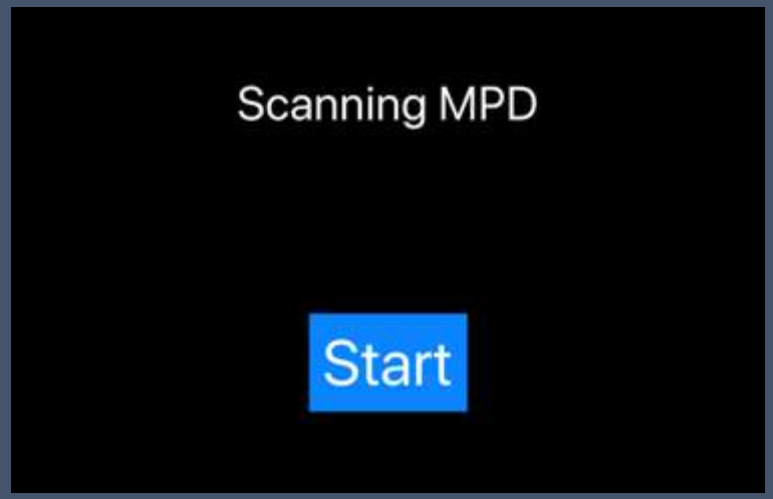


SF Symbols

San Francisco system fontとバランスが良い。フォントと同様にウェイト等を調整できる。

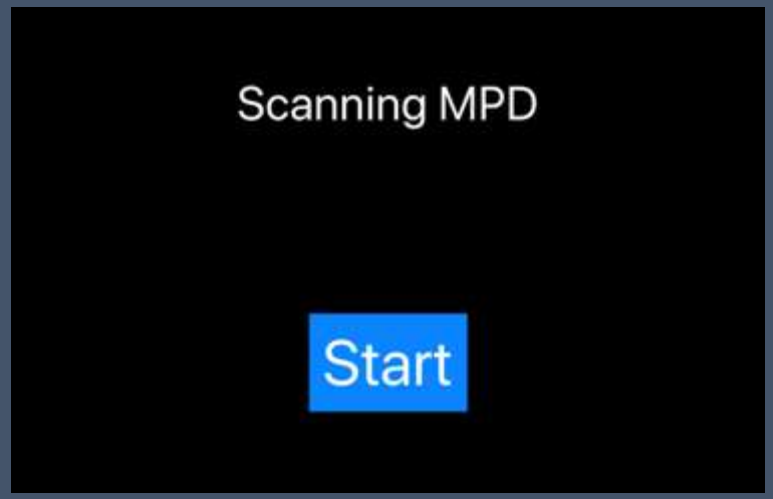
https://developer.apple.com/design/human-interface-guidelines/sf-symbols/overview/

iOS 12 Dark Mode




```
if darkMode {
        view.backgroundColor = .black
        Label.textColor = .white
        Button.setTitleColor(.white, for: .normal)
        Button.backgroundColor = UIColor(red: 10/255, green: 132/255, blue: 255/255, alpha:1.0)
} else {
        view.backgroundColor = .white
        Label.textColor = .black
        Button.setTitleColor(.white, for: .normal)
        Button.backgroundColor = UIColor(red: 0/255, green: 122/255, blue: 255/255, alpha:1.0)
}
```

iOS 13 Dark Mode https://developer.apple.com/videos/play/wwdc2019/214/



Semantic Dynamic Colors

view.backgroundColor = .systemGroundBackground
Label.textColor = .label
Button.setTitleColor(.systemWhite, for: .normal)
Button.backgroundColor = .systemBlue

iOSアプリのはじめかた　まとめ

1. PlaygroundでSwift言語をざっくり覚える
   教材の例
   - 詳細！Swift iPhoneアプリ開発 入門ノート Part 2
   - Apple Books: Swiftによるアプリケーション開発：入門編
2. 基本的なiOSアプリ開発のお作法を学ぶ
   推奨教材
   - Start Developing iOS Apps (Swift)
3. Foundation, UIKitについて学ぶ
   教材の例
   - 詳細！Swift iPhoneアプリ開発 入門ノート Part 3
4. アプリを作りながらXcodeに詳しくなる
   - 最低限必要なソースコード管理(Git)とデバッグは今日紹介しました

Follow-up

- 講演後、iOSアプリのバイナリ互換性に関する質問がありました。例えば「Swift 3で書かれたプログラムをXcode 9でビルドし、それをiOS 13で動作させて問題ないか？」という問いです。
- Swift 4以前はアプリと同じバージョンのstandard libraryとruntime libraryを同梱して配布していたのでiOS13でも動作するはずです。
- Swift 5でバイナリ安定化（ABI Stability）が達成されました。このためSwift 5で書かれたプログラムをXcode 10.2でビルドしたアプリは、iOS 13のstandard libraryとruntime libraryを用いて動作します。
- 概要は下記参照（日本語）：
  https://www.softantenna.com/wp/software/swift-5-released/
- ABI Stabilityの詳細は下記参照（英語）：
  https://swift.org/blog/abi-stability-and-more/

openaudiolab.com/jp