iOSアプリ制作のための Objective-C入門

中川　聡

nakasen_20th

ねらい

- 最近 iOS アプリ開発を「やりたくなった」
- 最近 iOS アプリ開発を「やらなければならなくなった」

「とにかく慣れていただきましょう」

本日やること

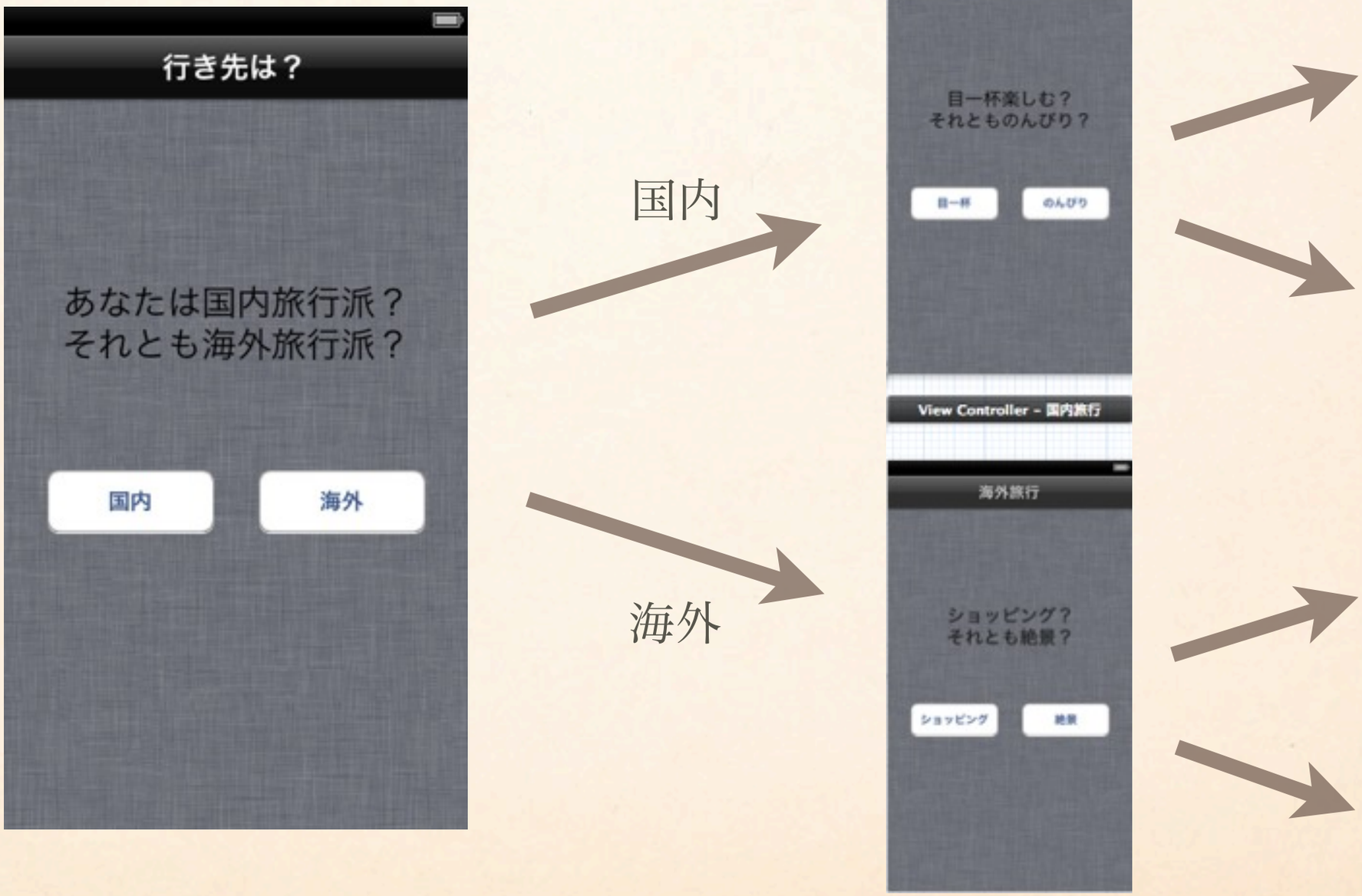
- 質問シート型のアプリを作成

1. まずは1行もプログラムを書かずに
2. 次にGUIツールを全く使わずに

✴ 本日のリソースは全てGitHubに上がってます！

https://github.com/nakasen/iOS_Seminar_for_beginners

どんなアプリ？


行き先は？
あなたは国内旅行派？
それとも海外旅行派？
国内
海外
国内
海外
国内旅行
目一杯楽しむ？
それとものんびり？
目一杯
のんびり
View Controller – 国内旅行
海外旅行
ショッピング？
それとも絶景？
ショッピング
絶景

Storyboardのみで作る

赤線の部分
重要

テンプレート選び

- 新規プロジェクトの際テンプレート選びが重要
- 今回は（たいていの場合は）「iOS」→「Application」
  →「Single View Application」
- 間違えて選んだことにあとで気付いたら、ジタバタする
  より作り直した方が早い（初心者のうちは）

Storyboardでのキーワード

説明に必要なのでこれだけは。（赤枠内から）

Navigator Area　Utilities Area

Standard Editor　Assistant Editor

Project Navigator

Attributes inspector　Size inspector　Connections inspector

Object library　Media library

Storyboardでの作業

- Object library から UIパーツを選んで View 上に貼る

  （これは簡単。失敗したら Command + Z）

- オブジェクトから Control + ドラッグでひもづけ

  （これは・・・間違えないように。ミスは後々響く）

  （見て覚えて体で覚える。忘れたらGitHubの「Manual.txt」を見ましょう）

Storyboardを一切使わない

- 「iOS」→「Application」→「Empty Application」
- 必要なキーワードが増えます。

Debug Area

Search Navigator　Issue Navigator　Debug Navigator

File inspector　Quick Help inspector

何を書かないといけないの？

- とりあえず動かしてみる
- Debug Area になんと書いてある？
- 足りないものをコードで生成してみる

やっとObjective-C

- アプリケーションの起動プロセスを知る

  A Day In The Life

  「iOS アプリの構造がどのようになっているか紐解いてみる」の記事が素晴らしい！　（こんなにわかりやすく書けない）

- AppDelegate.m で UIWindow オブジェクトを生成

- Root View Controller の役割を果たすものが必要

Objective-Cの特徴

- Cとオブジェクト指向のハイブリッド言語
- メソッドの定義、呼び出し方が独特
- あとは普通に動的オブジェクト指向言語です
- 元がC言語なので、書く側のモラルは当然必要

Objective-Cのメソッド

- 定義

```
- (NSString *)addString:(NSString *)string1 nextString:(NSString *)string2
{
    NSString *compositeString = string1;
    compositeString = [compositeString stringByAppendingString:string2];
    return compositeString;
}
```

- 呼び出し

```
NSString *displayString = [self addString:@"ABC" nextString:@"DEF"];
```

メソッド呼び出し

- メソッドの引数をメソッドで呼び出し

```
int number = 123;
NSString *displayString =
[self addString:@"ABC" nextString:[NSString stringWithFormat:@"%d", number]];
```

View Controller

- Window には Root View Controller が必要
- View Controller には最低一枚の View が必要
- UI パーツは View に addSubview

UIKit フレームワーク

- UI 処理は UIKit フレームワークにおまかせ
- UIWindow　UIViewController　UINavigationController
  UILabel　UIButton　・・・
- Apple 公式ドキュメント
  「iOS ヒューマンインターフェースガイドライン」
  は膨大なユーザテストの成果物

Xcodeの強みを活かす

- Jump to Definition は使える！
  （ソースコードを右クリック）
- 補完もかなりイケる！
- Quick Help も親切！
- UIKit オブジェクトを Jump to Definition と Quick Help で追いかけて行けば、Objective-C の習得にもつながる

プロパティ

- インスタンスのアクセサメソッドを自動生成
- getter メソッド名はインスタンス名そのまま
- setter メソッド名は set + 大文字で始まるインスタンス名
- ドットシンタックスでアクセスできる
  → メソッド呼び出しソースの読みにくさを緩和

NSLog とブレークポイント

- デバッグ時のダンプには NSLog() 関数

```
NSLog(@"displayString = %@", displayString);
```

- ブレークポイント設定時、設定したこと忘れないで

スコープなど

- @private、@protected、@public
  （デフォルトは@protected）
- ある方法を使うと隠蔽は破られる　気をつけましょう
- 外部クラスのオブジェクトへのアクセスは通知が基本
- 通知の方法はおよそ３通り　次回以降の勉強会で