Introduction to iPhone Apps using Swift

October 2019

Jens Nerup

@barkoded at Twitter

jens@makecph.com

Prerequisite

You'll need a Mac capable of running latest version of Xcode

Agenda

- Languages & Platform
- Cocoa Design Patterns
- View Controller
- Application Launch
- Lets get started…
- Resources

Languages & Platform

Apple Supported Languages

	Swift	**Objective C**
Typing discipline	Static, Strong, Inferred	Static, Dynamic, Weak
Influenced by	Objective-C, Rust, Haskell, Ruby, Python, C#...	C, Smalltalk
Paradigm	Multi-paradigm[1]	Reflective, class-based object-oriented
First appeared	2014	1984
Popularization	Apple	NeXT

[1] Protocol-oriented, object-oriented, functional, imperative, block structured

Objective-C

- Thin layer atop C, and is a "strict superset" of C
- All of the syntax for non-object-oriented operations are identical to those of C
- All of the object-oriented features is an implementation of Smalltalk-style messaging.
- Objective-C++ files are denoted with a .mm file extension. (Combination of C++ and Objective-C syntax)

Objective-C - continued

- Blocks is a nonstandard extension for Objective-C that uses special syntax to create closures[2].
- Usable through out all the platforms - macOS, iOS, iPadOS, watchOS and tvOS.
- Caution - As of iOS 13 some frameworks are only available using Swift.

---

[2] A syntax that is very hard to remember unless you work with it every day http://goshdarnblocksyntax.com

Swift

- Publicly announced during WWDC 2014 - June 2014
- Version 1.0 released with iOS 8 on **September 17, 2014**
- Latest version (5.1) released on **September 19, 2019**
- Builds on the best of C and Objective-C and many other languages
- Seamless access to all existing Cocoa frameworks

Swift - continued

- Safe programming patterns and "modern" features
- Mix-and-match interoperability with C and Objective-C (but not C++)
- Reference types (classes & closures) and value types (structures & enumerations)
- Actively developed by Apple Inc. and others
- Open Source - http://swift.org & https://github.com/apple/swift

User Interface Frameworks

	UIKit	**SwiftUI**
Language	Swift, Objective C	Swift
Platforms	iOS/iPadOS/watchOS/ tvOS	All (incl. macOS)
Stable	Yes	Kind of stable
Type	Imperative[3]	Declarative

[3] Mostly works on delegates and callback blocks

Additional App, Graphics & Games Frameworks

- **App Frameworks**
  Swift Standard Library, Foundation, UIKit
- **Graphics and Games Frameworks**
  Metal, Core Graphics, GLKit, …

Additional App Services, Media and Web Frameworks

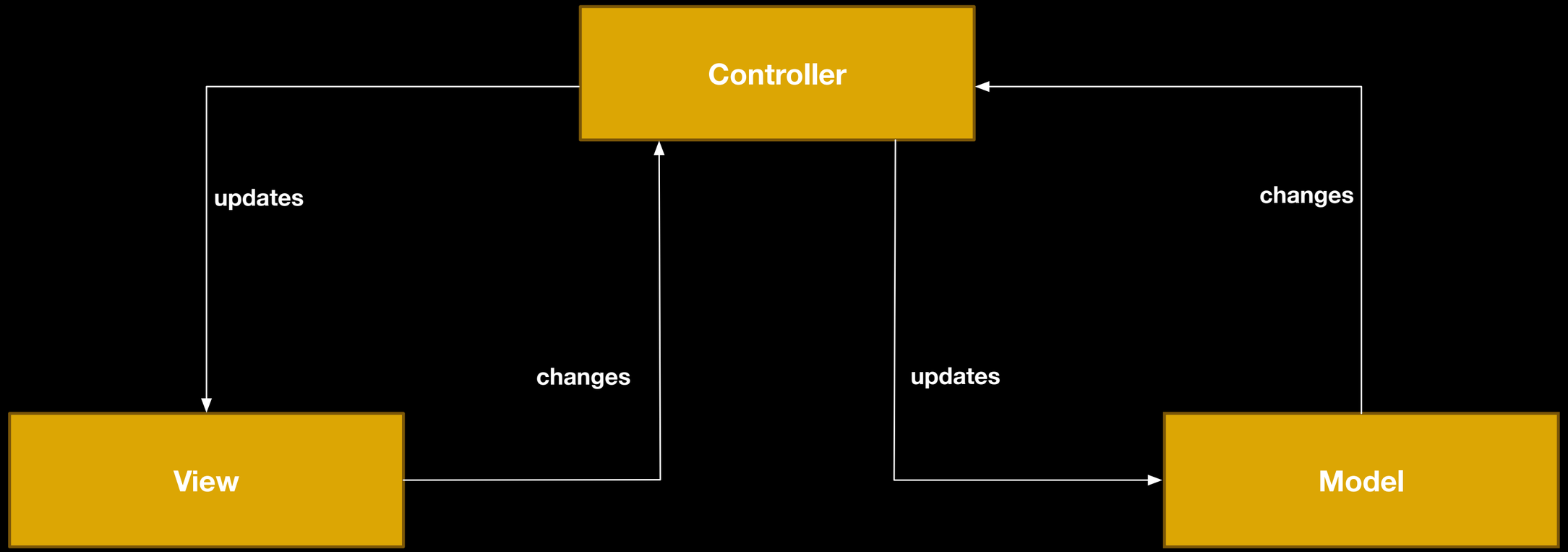
- **App Services**
  MapKit, Core Location, Core Data, …
- **Media and Web**
  AVKit, WebKit, Safari Services, …

Cocoa Design Patterns

3 Essential Design Patterns

- Model View Controller - MVC
- Delegate Pattern
- Notification (Observer Pattern)

Model View Controller



Delegate in Cocoa

Purpose: *Object expresses certain behaviour to the outside but in reality delegates responsibility for implementing that behaviour to an associated object*.

- Defined using a **protocol**
- Defining both required and **optional** methods. If Swift only protocol then **optional** isn't supported
- Mostly assigned on the delegating **class**

Delegate

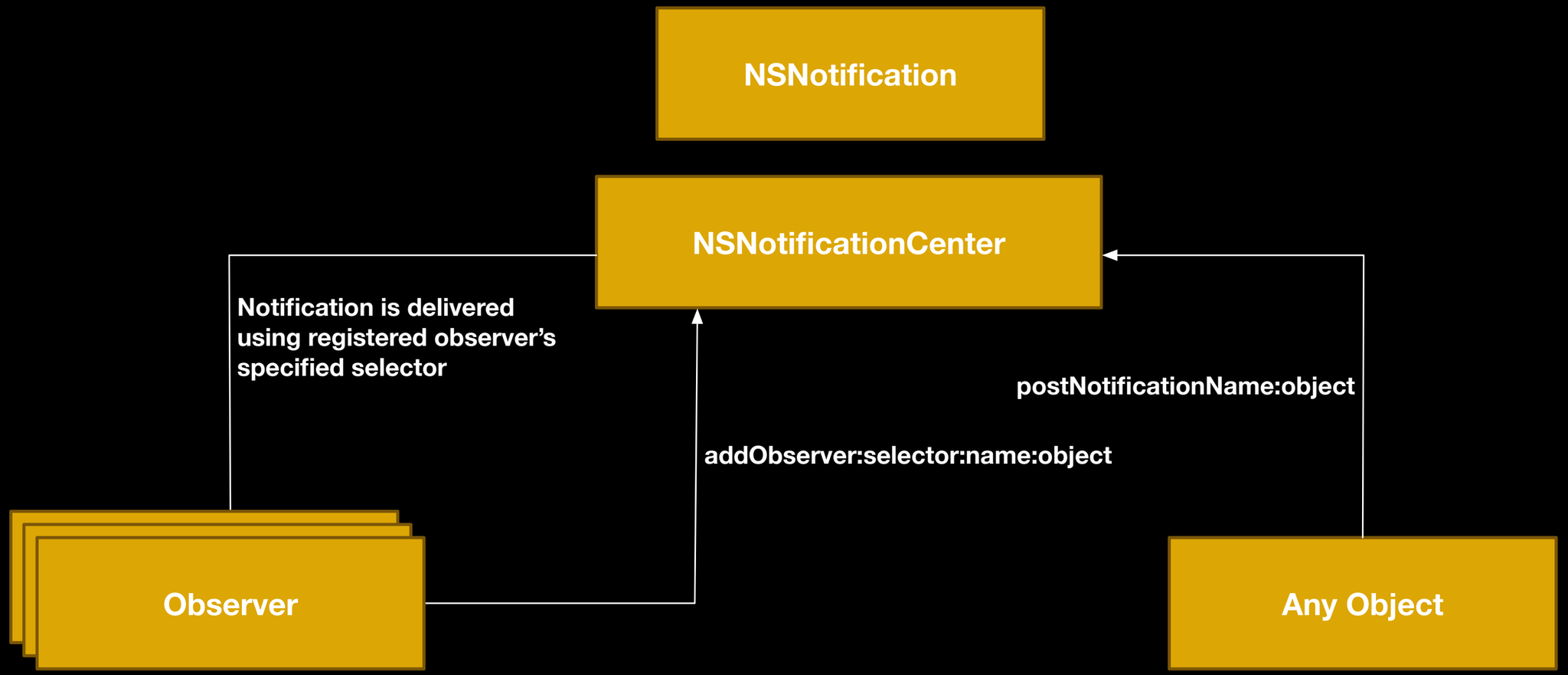
```
protocol PlaygroundServiceDelegate: class {
  func didUpdate(playground: Playground)
}

class PlaygroundService {
...
// The delegate should always be weak for class protocols to avoid retain cycles
weak var delegate: PlaygroundServiceDelegate?
...
}
```

Cocoa Delegate Naming

- Usually include on of three verbs: **should**, **will** or **did**
- **should** methods should return a value.
- **will** and **did** are not expected to return values
- **will** and **did** are primary informative before and after an occurrence - *think of it as a one to one notification.*

Notification



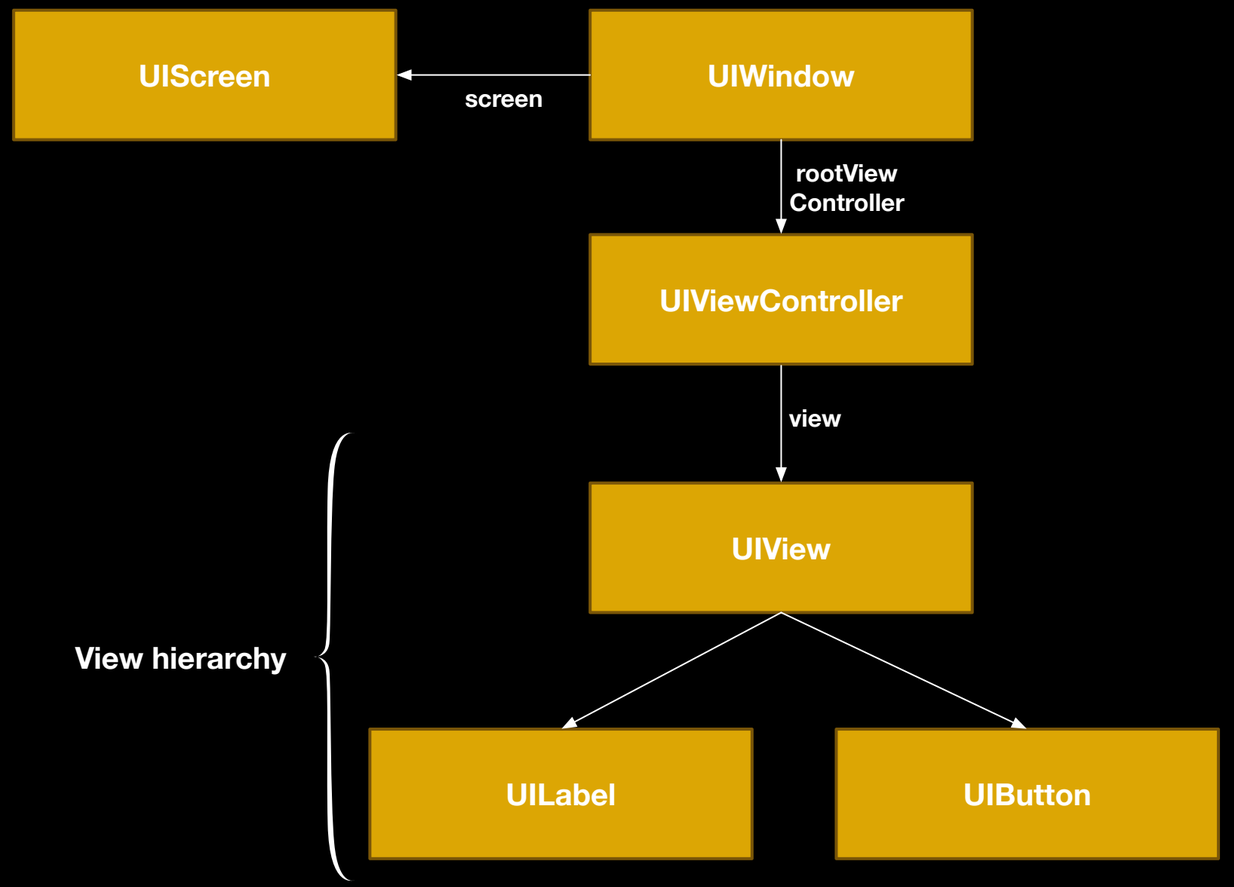
UIViewController

View Controller

… manages a set of views that make up a portion of your app's user interface. It is responsible for loading and disposing of those views, for managing interactions with those views, and for coordinating responses with any appropriate data objects. View controllers also coordinate their efforts with other controller objects—including other view controllers—and help manage your app's overall interface.

Root View Controller

- The *RootViewController* set on **UIApplication** via your **UIApplicationDelegate**
- Each **UIViewController** has an associated **UIView** (with zero or more children - the view hierarchy)
- Defines the initial visual starting point



"Content" View Controller

- Presents content on the screen
- Should be reusable and self-contained entities.
- Knows of the model layer and manages the view hierarchy.
  Common tasks:
  - Show data to the user (e.g. details)
  - Collect data from the user (e.g. forms)

MVC ≠ Massive View Controller

Should be avoided. Use the SOLID principal

SOLID

- **S**: a class should have only a single responsibility
- **O**: open for extension, but closed for modification
- **L**: objects should be replaceable with instances of their subtypes
- **I**: many client-specific interfaces are better than one general-purpose interface.
- **D**: one should “Depend upon Abstractions. Do not depend upon concretions.

The Light View Controller

Separate concerns to:

- Avoid becoming the place for all tasks
- Help you maintain readability, maintainability, and testability.

The Lighter View Controller

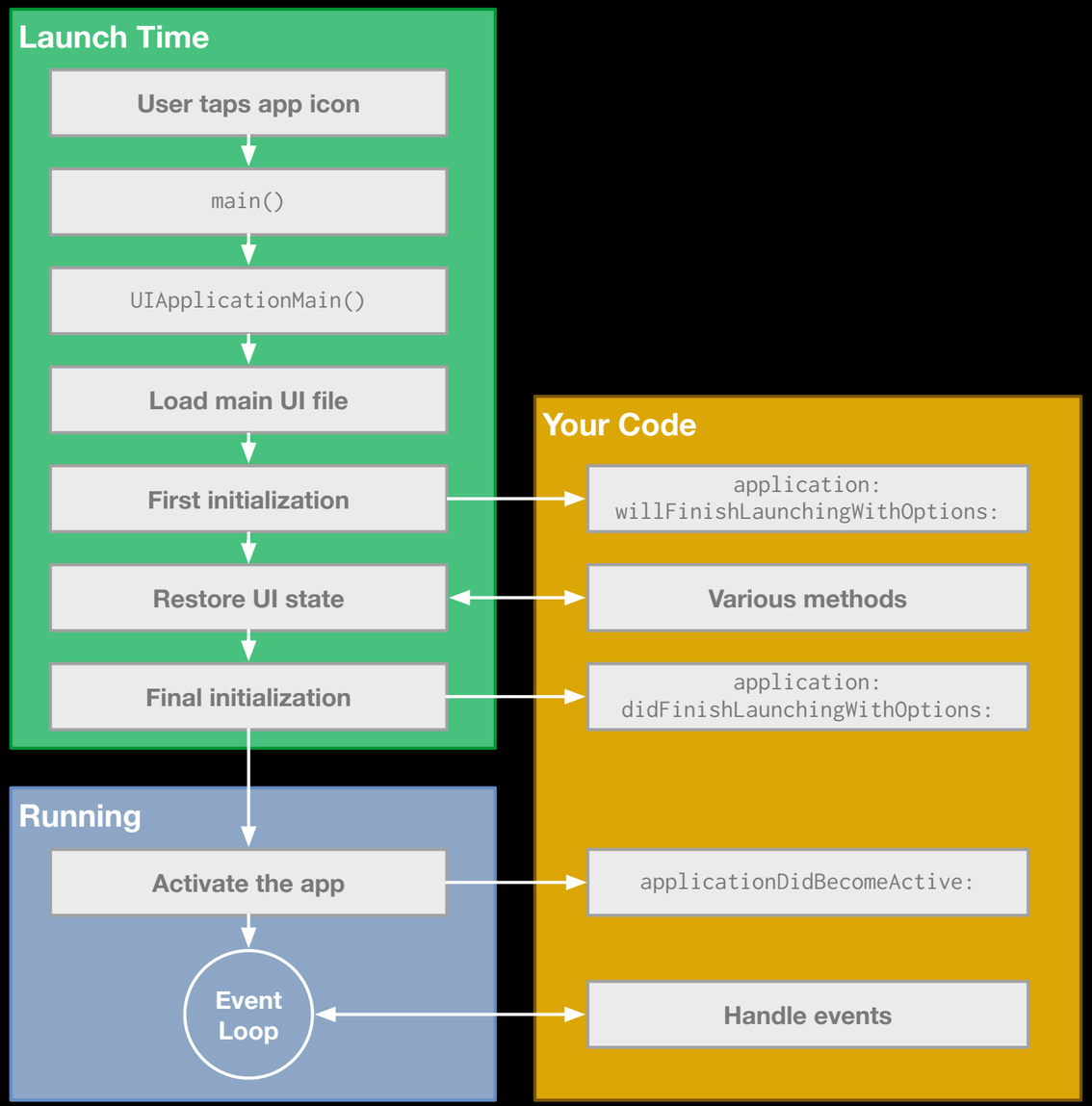
Easy steps:

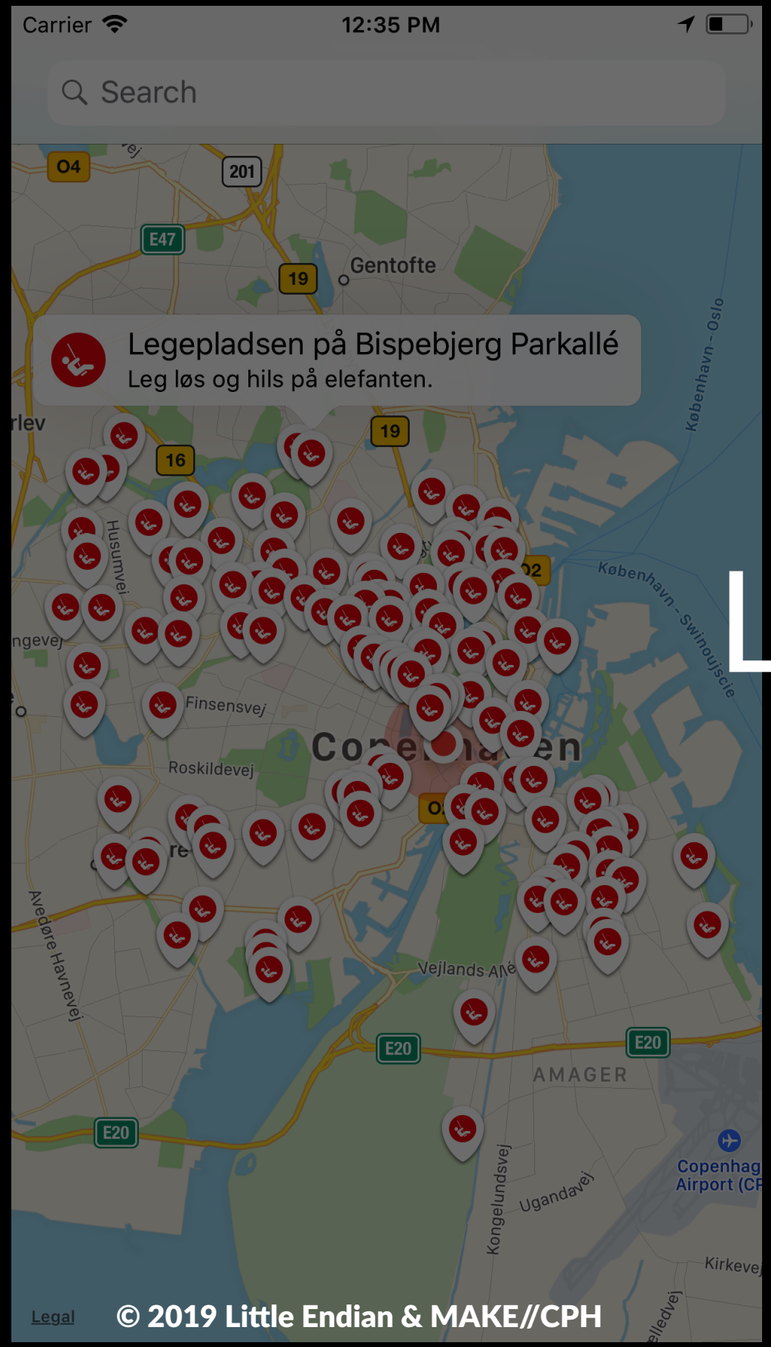
- Separate the **datasources** and **delegates** from the **ViewController**
- Move networking to separate classes
- Move domain display formatting to separate classes - presenters/view-model.
- Use categories on e.g. cells to set domain information - try to

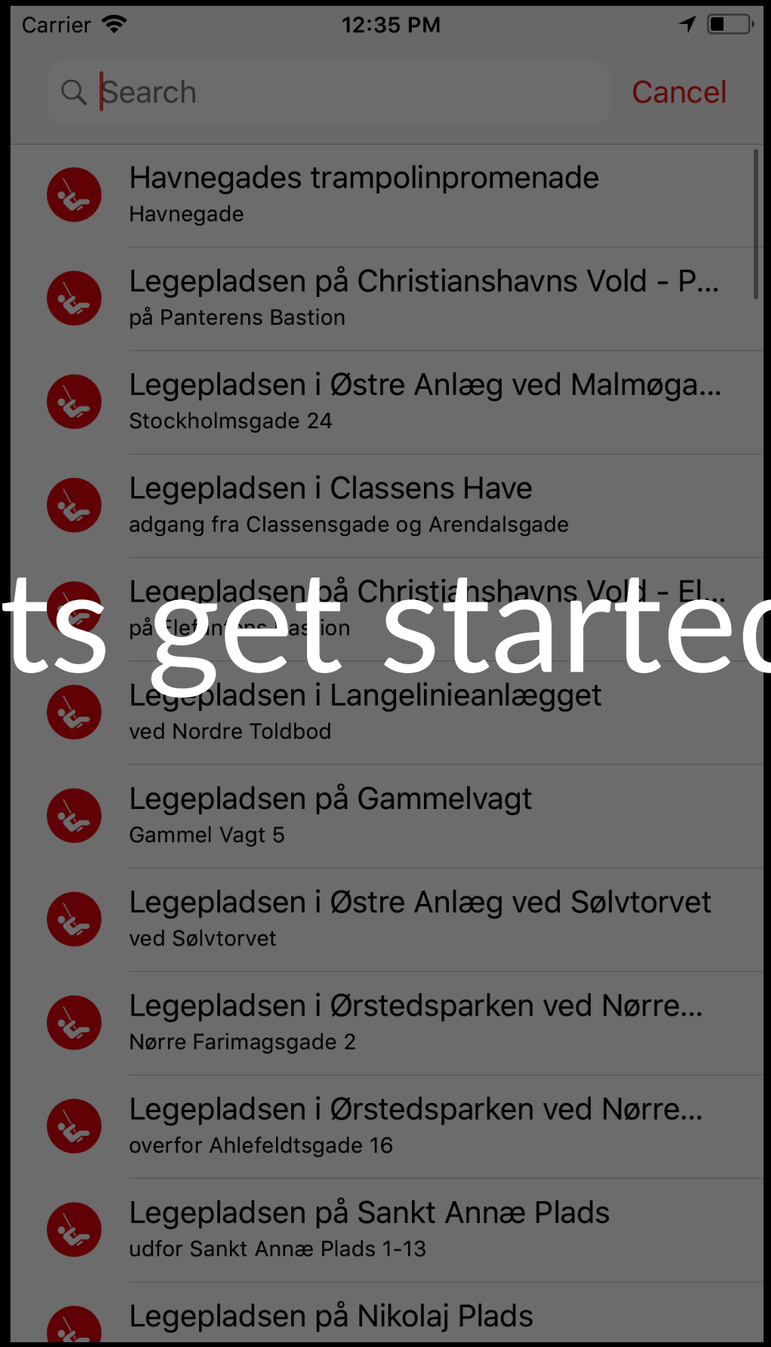
Application Launch

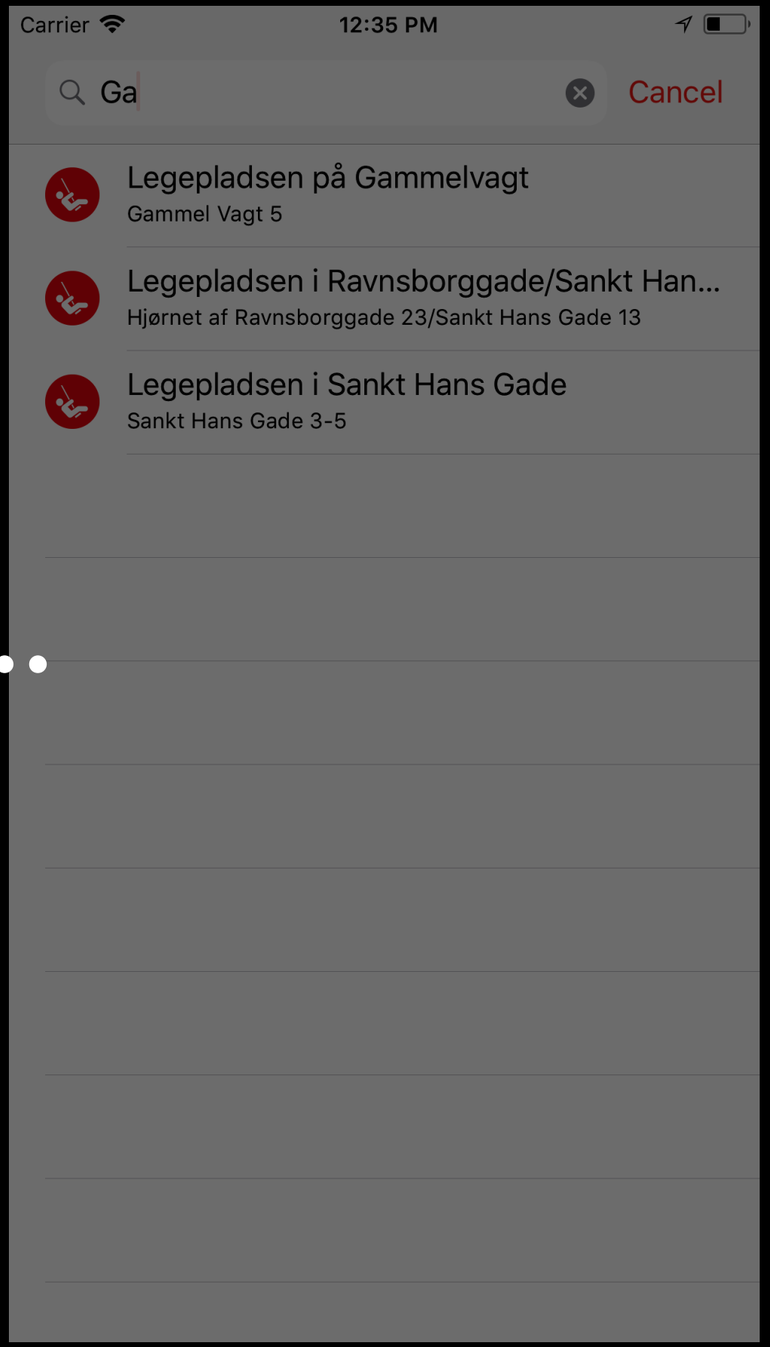
What to Do at Launch Time

- Check the contents of the launch options dictionary for information about why the app was launched, and respond appropriately.
- Initialise the app's most critical data structures.
- Prepare your app's window and views for display.
- Be as lightweight as possible to reduce your app's launch time.
- Start handling events in less than 5 seconds









Lets get started...

Resources

Online

- Source code for this presentation
- The Swift Programming Language - Swift 5.1
- Developing iOS 10 Apps with Swift
- API Reference
- SwiftUI
- Swift Playgrounds for iPad

Online - continued

- Hacking With Swift
- 8 Patterns to Help You Destroy Massive View Controller
- Swift Package Manager - Official Swift Package Manager
- Cocoapods - Open Source Package Manager
- Carthage - Open Source Package Manager

Oldies but Goodies

- **The C Programming Language (2nd Edition)**
  by Brian W. Kernighan and Dennis Ritchie
- **Programming in Objective-C**
  by Stephen Kochan

Selected Extra Tools

- AppCode
  Alternative IDE to Xcode but not a full replacement yet.
- Kaleidoscope
  Great diff and merge tool but sadly not in active development.
- Tower
  Superb commercial = git client.
- GitUp - (free)
  Another great open source git client for the Mac.

Selected Extra Tools - continued

- SimPholders
  A lovely little tool to manage your simulators.
- Charles Proxy
  When working with REST API's this is a must as a proxy.
- Paw
  Working with REST API? Then consider this to explore the API.
- Sketch
  Vector graphics editor - suberb alternative to Adobe Illustrator.