Introduction to iOS

using Swift

Jens Nerup

@barkoded at Twitter

Prerequisite

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

Agenda

- Swift & iOS Platform
- Cocoa Design Patterns
- Application Launch
- View Controller
- Lets get started...

Swift & the iOS Platform

Swift

- Publicly announced during WWDC 2014 - June 2014
- Version 1.0 released with iOS 8 on **September 17, 2014**
- Latest version (4.0.3) released on **December 5, 2017**
- Builds on the best of C and Objective-C and many other languages
- Seamless access to all existing Cocoa frameworks

Swift

- Safe programming patterns and "modern" features
- Mix-and-match interoperability with C and Objective-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

Swift - Memory

- **A**utomatic **R**eference **C**ounting aka **ARC**
- Reference counting applies only to instances of classes.
- Watch out for *Retain Cycles* and *Closure captures*

iOS - App, Graphics & Games Frameworks

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

iOS - App Services, Media and Web Frameworks

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

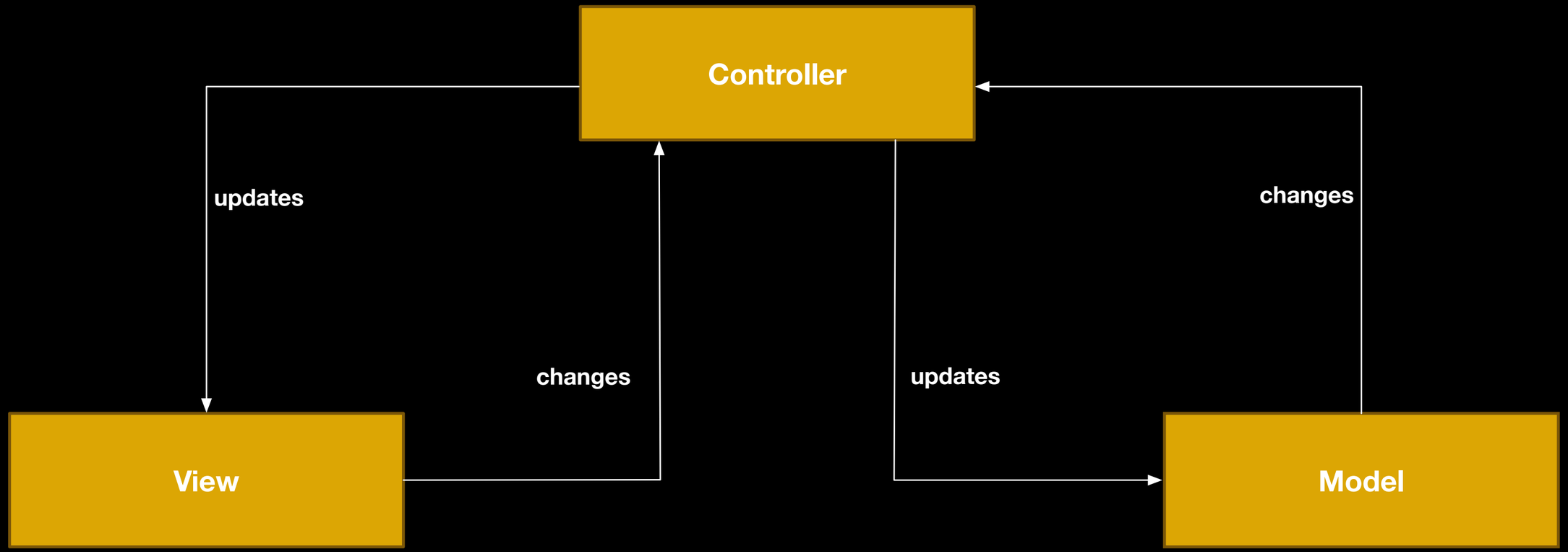
Essential Cocoa Design Patterns

Essential Cocoa Design Patterns

3 essential 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.
- Mostly assigned on the delegating **class**

Delegate

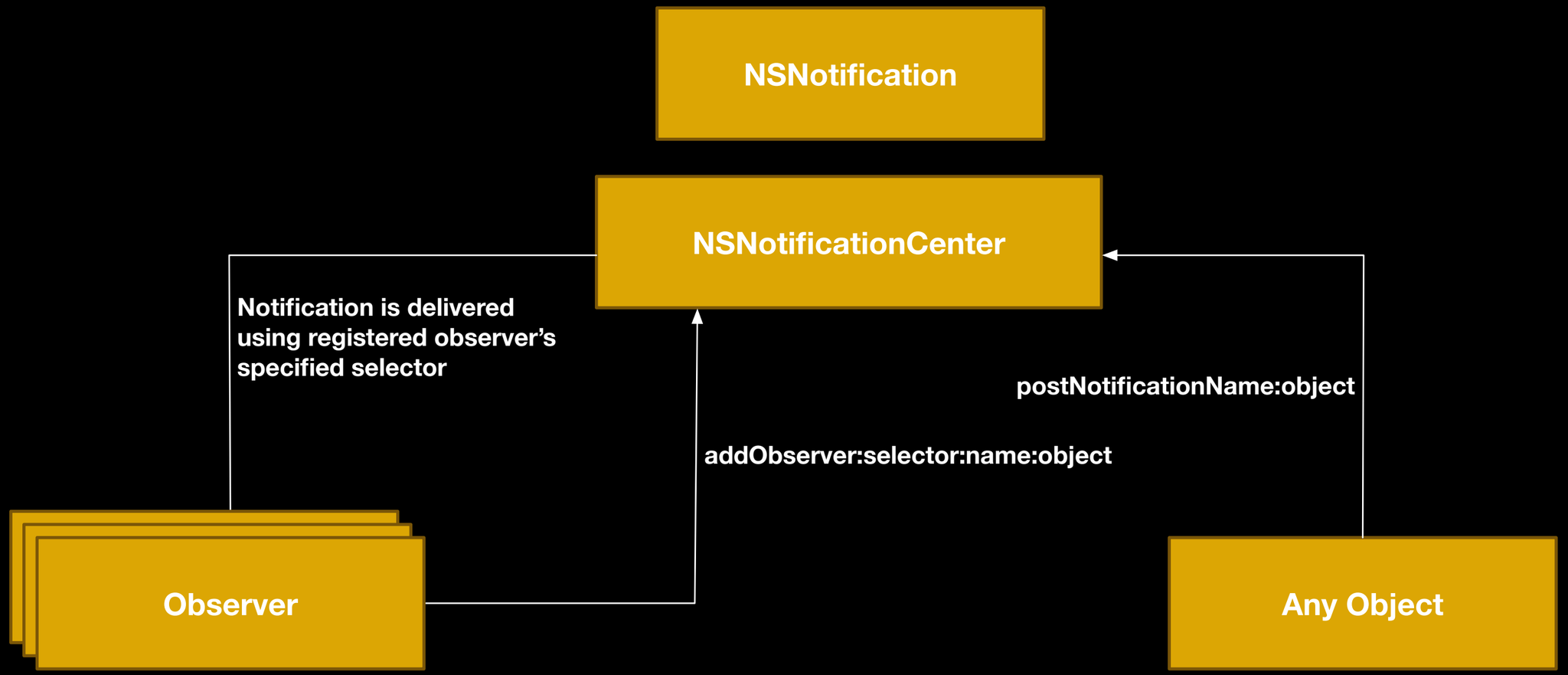
```
protocol PlaygroundServiceDelegate: class {
  func didUpdate(playground: Playground)
}

class PlaygroundService {
...
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.*

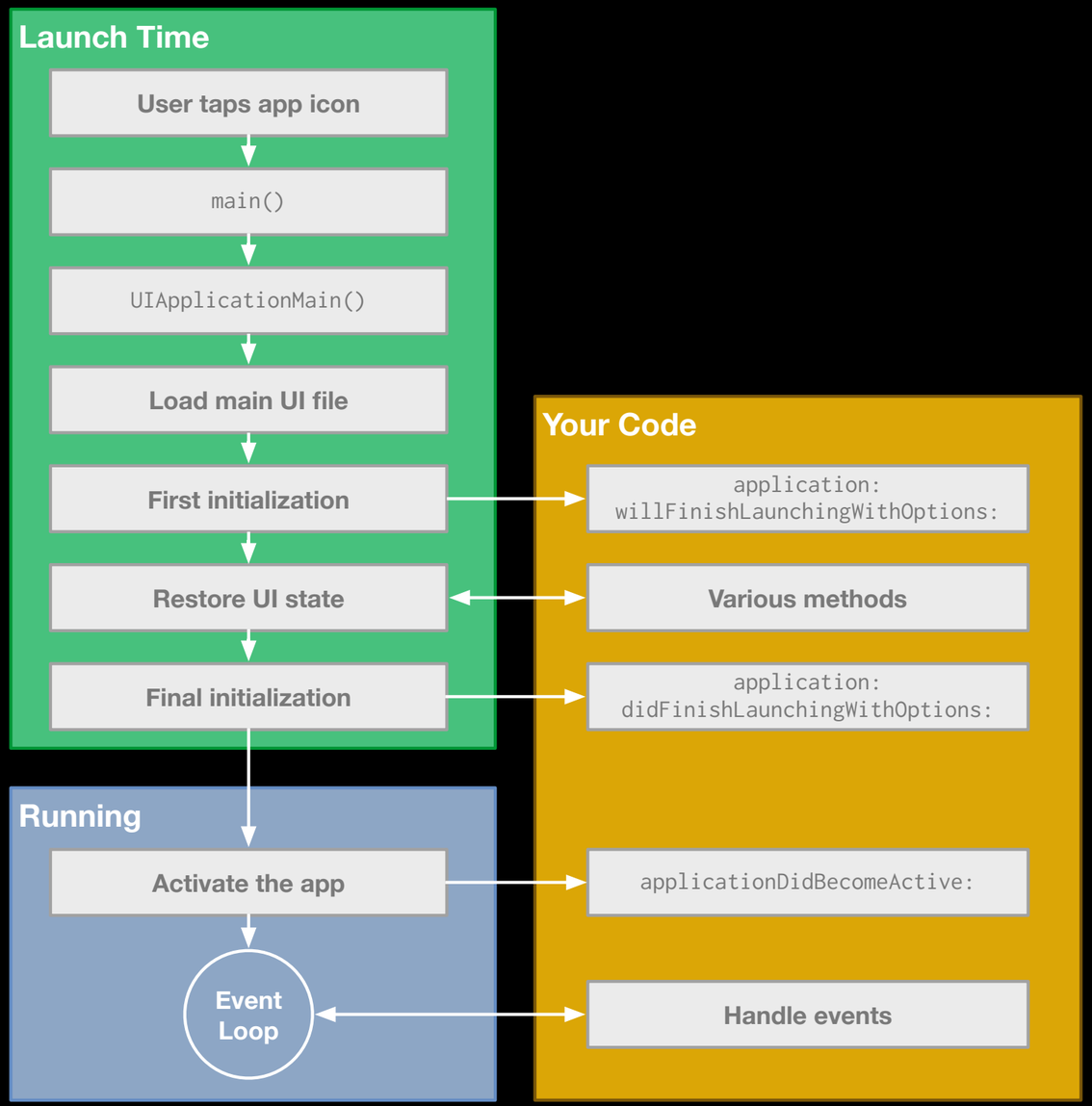
Notification



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



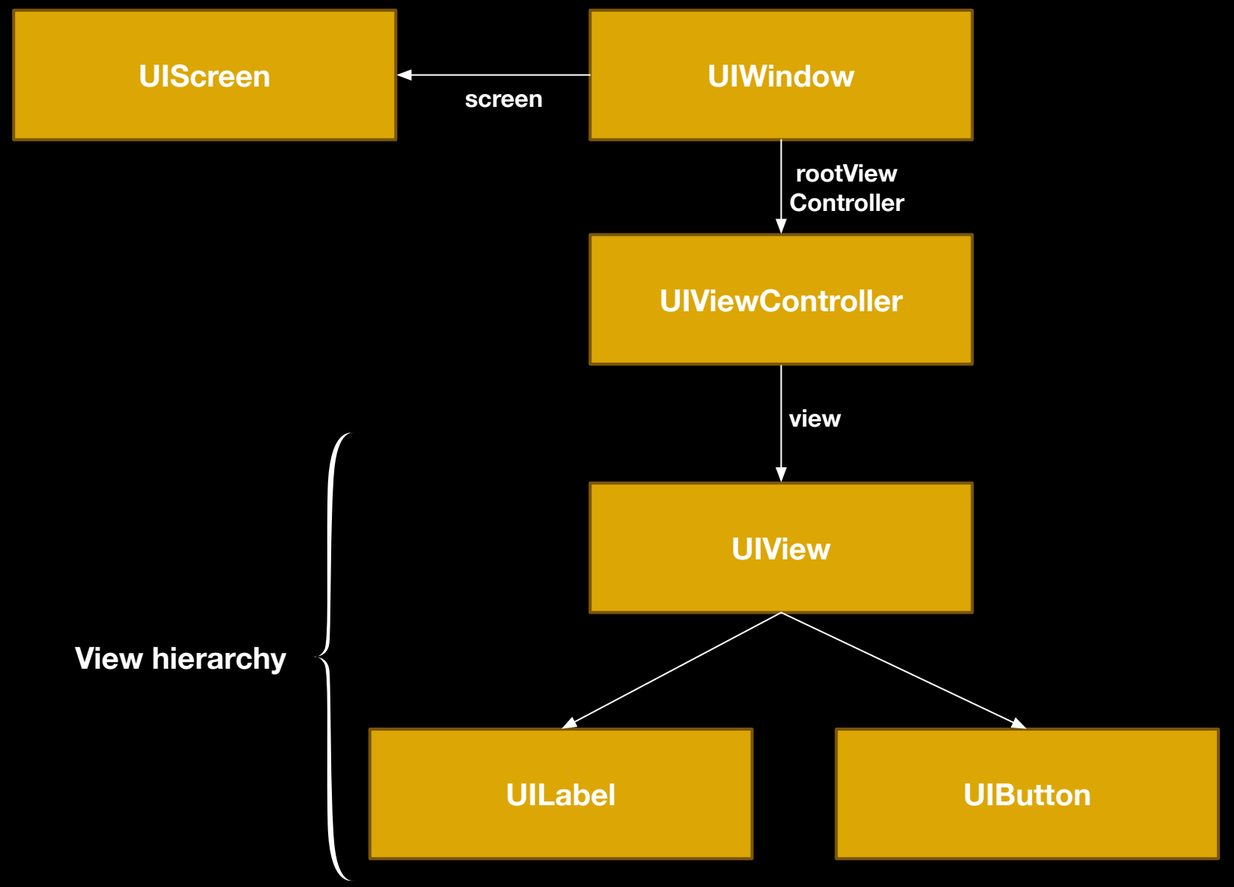
View Controller

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)

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

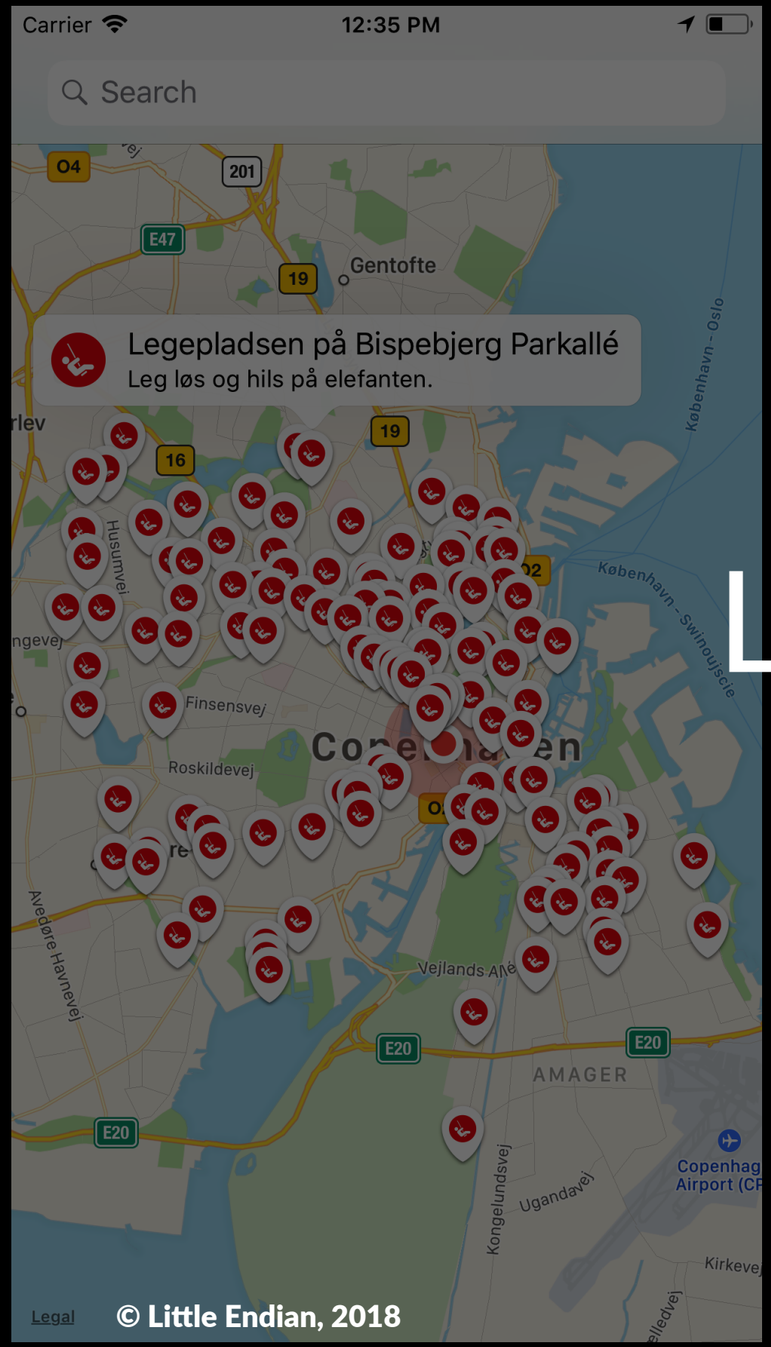
Separate concerns to:

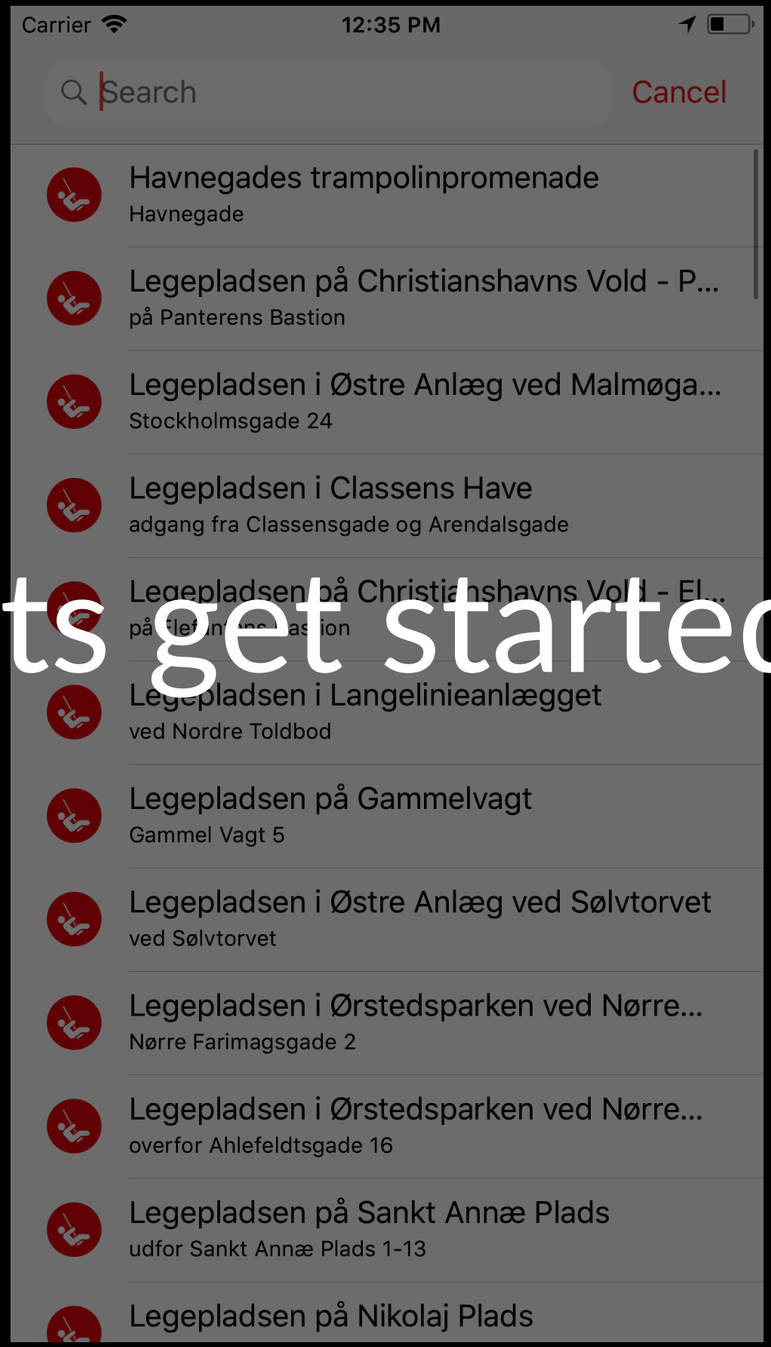
- Avoid becoming the place for all tasks
- Help you maintain readability, maintainability, and testability.

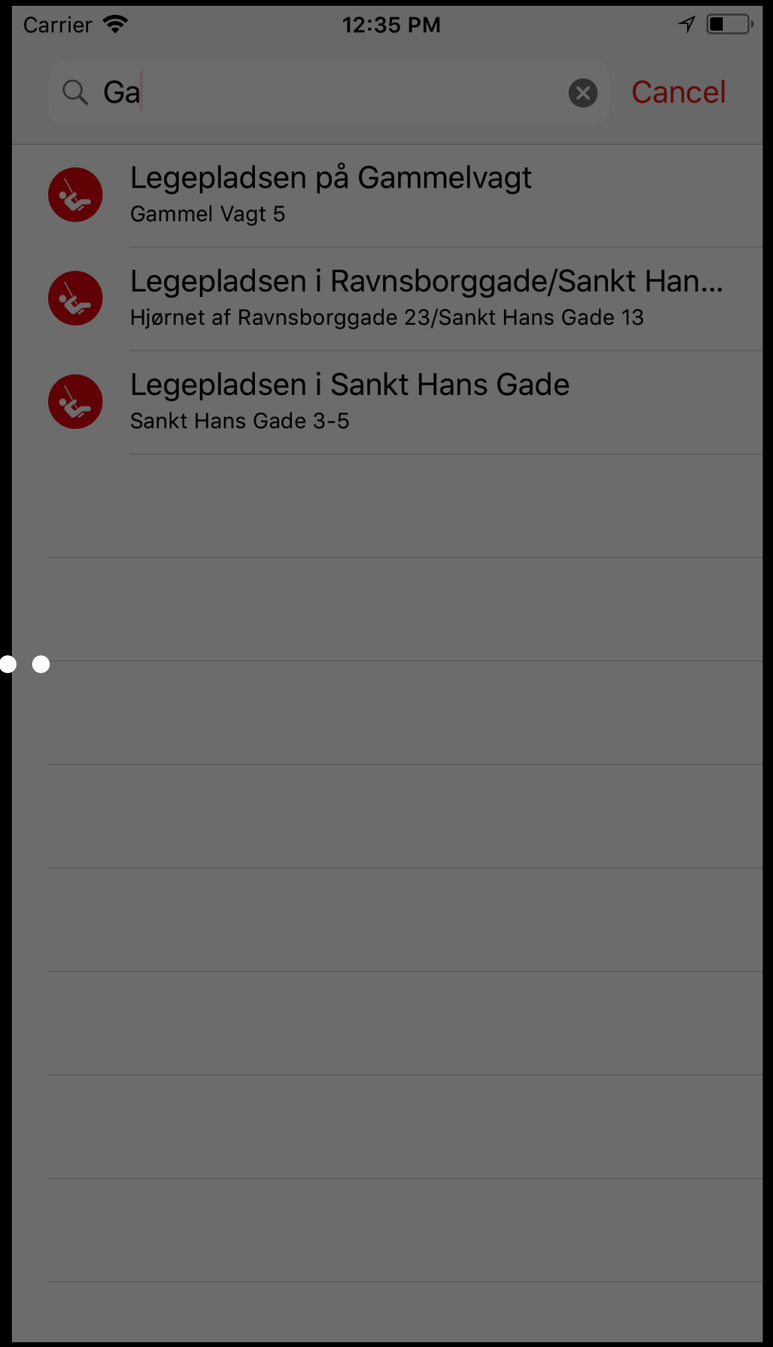
The Lighter View Controller

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.







Lets get started...

Resources

Online

- The Swift Programming Language (Swift 4.0.3)
- Intro to App Development with Swift (iBooks)
- App Development with Swift (iBooks)
- API Reference
- Swift Playgrounds for iPad
- This week in Swift (newsletter)

Online - continued

- 8 Patterns to Help You Destroy Massive View Controller
- SOLID
- Swift Package Manager - Package Manager
- Cocoapods - Package Manager
- Carthage - Package Manager

Offline

- **iOS Programming: The Big Nerd Ranch Guide 5th Edition**
  by Christian Keur and Aaron Hillegass
- **Programming in Objective-C**
  by Stephen Kochan
- **The C Programming Language (2nd Edition)**
  by Brian W. Kernighan and Dennis Ritchie

Selected Extra Tools

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

Selected Extra Tools

- 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 - alternative to Adobe Illustrator.

Little endian