Modern iOS Application Development

Rastislav Mirek

UIKit Demo

How did you like it?

Auto Layout

- Best tool for building UI, offers declarative syntax and visual editor to replace imperative code.
- Adaptive layout - different layout for different screen sizes.
- Required vs. optional layout constraints.
- Alternatives: E.g. Async Display Kit

Animating Layout Changes

```
UIView.animate(withDuration:  0.2) {
    self.view.layoutIfNeeded()
}
```

SwiftUI Demo

What are advantages of Storyboards and what are advantages of SwiftUI?

UIKit vs. Swift UI

UIKit	SwiftUI
MVC	Reactive, MVVM
Has UICollectionView	No UICollectionView
More mature, well known	New, modern & “hot”
More visual, navigation visualization	Hot reloading, multiple previews
Only targeting one platform	One UI for iOS, Mac an Apple Watch
Better tooling, more 3rd party libraries	Code only, no “black box” storyboard code
Classes & Inheritance	Structs & Composition
Wide OS version support	One iOS 13
Still dominant	Great for new apps

Swift

Swift’s Most Inovative Feature

First popular language to solve Tony Hoares Billion Dollar Mistake.

“I call it my billion-dollar mistake. It was the invention of the null reference in 1965. At that time, I was designing the first comprehensive type system for references in an object-oriented language (ALGOL W).”

Nulls are replaced by explicit optionals that can be chained e.g.

```
let newOptional = optional?.transformMethod()?.transformMethod()
```

- Optionals are in fact generic enums.
- By not force unwrapping them but using if let unwrapped = optional or guard let unwrapped = optional a lot of errors can be avoided.

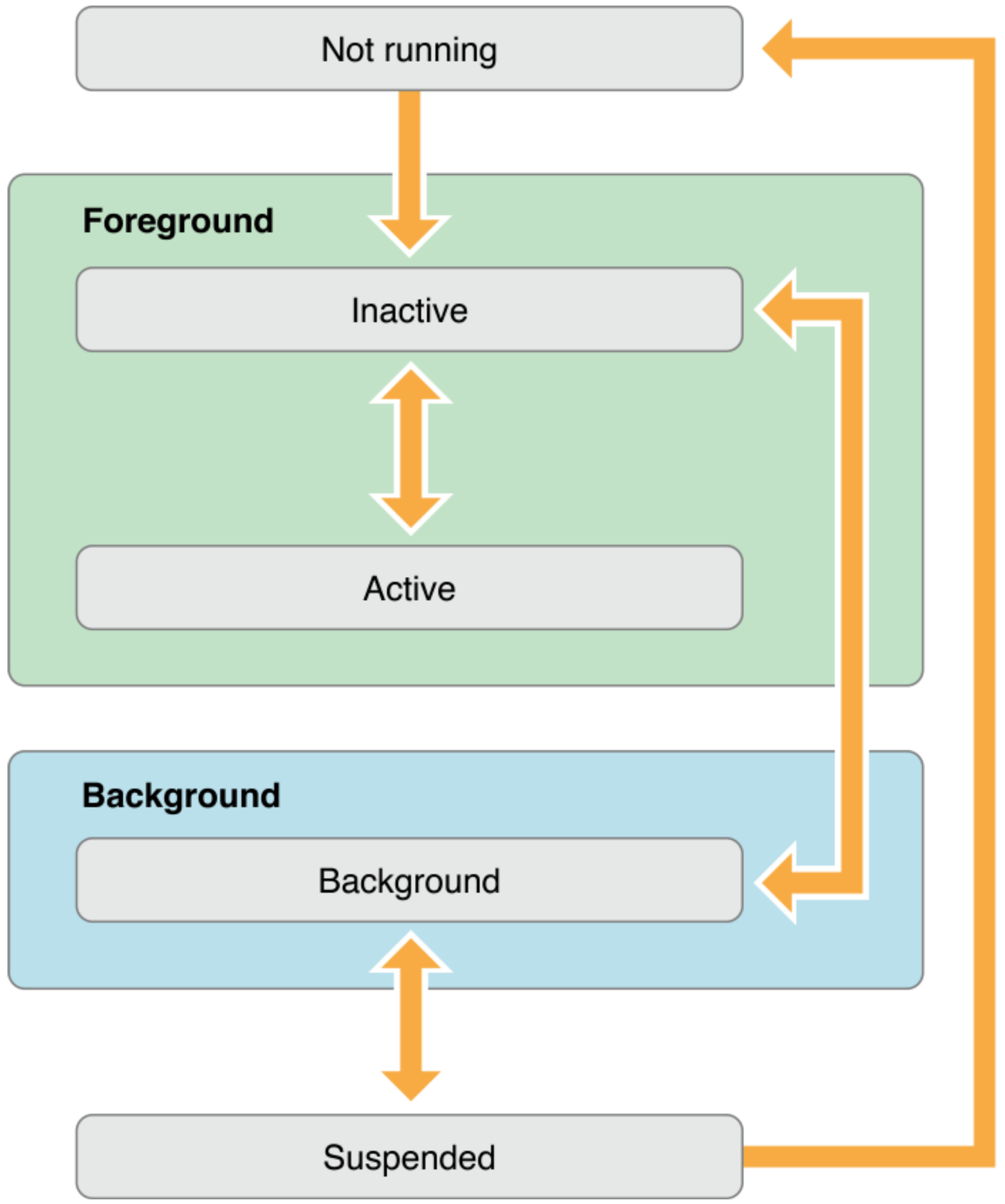
What do you think is a biggest gain of not having implicit nulls? Can you quantify it?

Swift Key Characteristics

- Verbose: All function parameters have labels unless otherwise specified.
- Functional as well as imperative.
- Modern features: Optionals, custom operators, imutability, extensions on protocols, minimalistic syntax, enums with associated values, code in unicode, ...
- What you would expect: Generics, exceptions, functional methods (map, reduce, ...), structs, inheritance, …
- Stable, fast, open source

iOS App Architecture

App Lifecycle



Architecture Tips

- Model-View-Controller (only for UIKit):
    - Keep logic out of both Model and View
    - Extract logic from controllers to separate services layer
    - Wrap data persistence into separate layer with API, independent of persistence technology used
- If you are building complex App, split your code into several XCode projects
    - Utilise Dynamic Frameworks
    - Build your own reusable libraries
- As your controllers start to grow consider MVVM or VIPER architecture.

Persistance

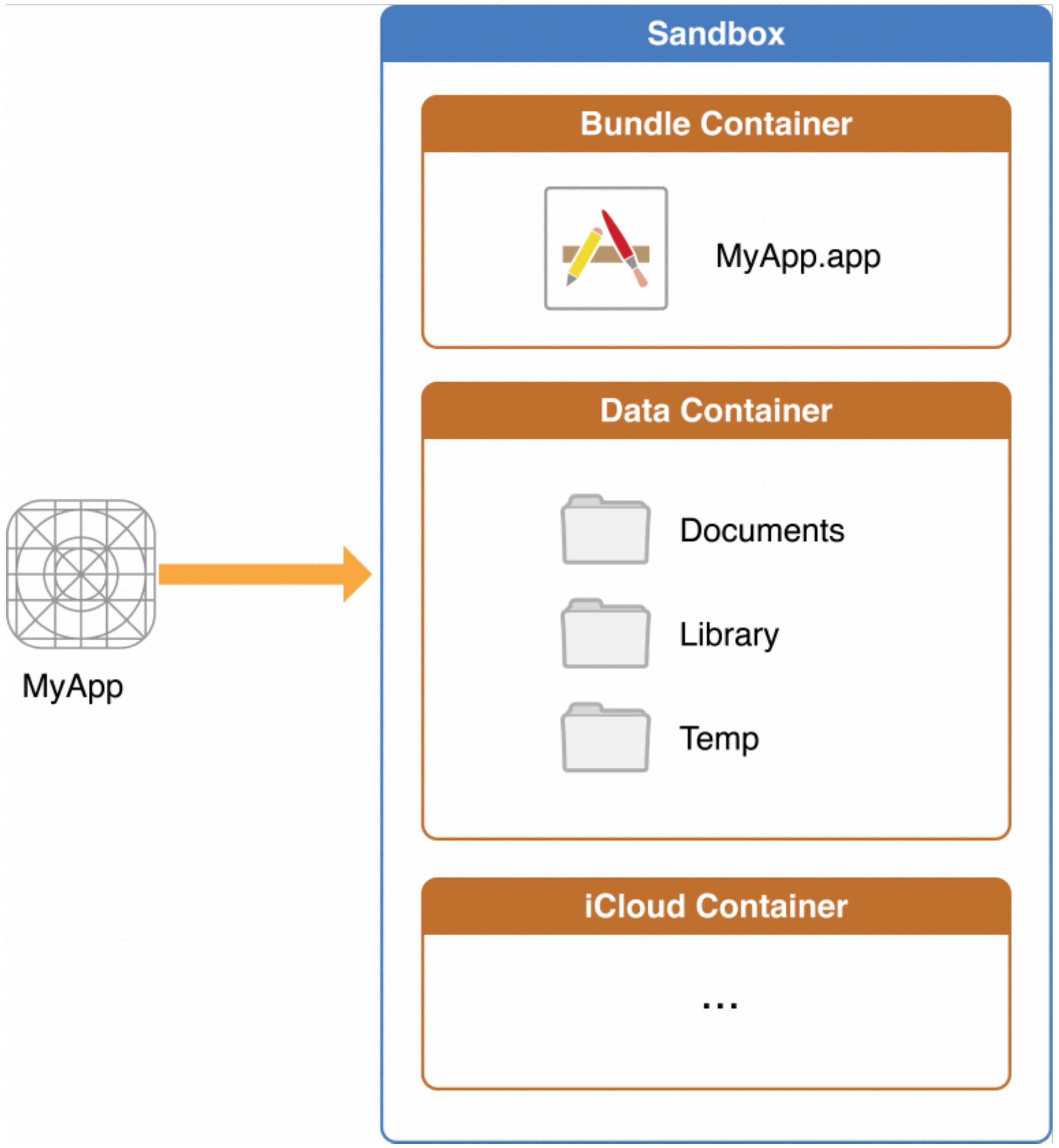
How do you plan to persist data for your projects?

Local Data Persistance

Commonly used local persistence technology:

- File system
- CoreData
- Realm
- Keychain
- UserDefaults (in combination with serializers)

App Container



Backend & Remote Persistence

Common backend choices:

- Firebase
- CouldKit
- Realm
- Custom server

Move long running tasks to background threads.

- GCD (DispatchQueue)
- OperationQueue

Development Tips

Development Tools

- Xcode
  - no good alternatives
  - works well with storyboards, XIBs, localisation, assets
- XCode Instruments
  - advanced debugging and profiling, some pretty cool
- Dependency Managers
  - Swift Package Manager (SPM)
  - CocoaPods
  - Carthage

What would you do to make your mobile app stand out from the crowd?

Any cool tech/visual things planned for your projects?

UICollectionView

`UICollectionView` might be the most flexible component of UIKit thanks to custom layouts.

- Custom layouts are not dependent on collection view nor data source.
- They extend `UICollectionViewLayout`.
- They can define arbitrary items layout as well as position of supplementary and decoration views.
- If having performance issues override `invalidationContext(forBoundsChange:)` and invalidate just the views that have been repositioned.
- Collection view supports interactive layout transitions and layout animations.

Tips

- Consider using Facebook API, Google API, Firebase for: User tracking, bug reporting, login, notification delivery, etc.
- Work with your designer(s); do not hesitate to tell them if any design is hard to implement.
- IB live, reusable views can be created with `@IBDesignable` and `@IBInspectable`.
- Interesting blur effect can easily be created with `UIVisualEffectView`.
- There is no performance penalty for concatenating strings in Swift.

- Icons, images, string files and data files can easily be organised with Asset Catalogs and then read in code.
- High app download size can be significantly decreased using on demand resources.
- Multiproject XCode workspaces
- There are ready-to-use controllers for camera, image library, email, sharing, . . .
- 2 things to avoid: Allowing rotation for just some screens of the app and accessing private APIs or properties.

Questions?

Thank you