

Developing Modern iOS Applications

Mgr. Rastislav Mirek

CEO, TypeSoft

Consultant, Touch4IT

PV239 Mobile Application Development, Faculty of Informatics,
Masaryk University

Organization

- Feel free to ask questions anytime.
- Breaks?
- Have something to add? Differences to Android? Feel free to discuss.
- On the following slides 'App' means Application :)

Outline

- 1 Introduction
 - Types of Mobile Applications for Apple Devices
- 2 Pros and Cons of Being iOS Developer
- 3 Developing Successful Applications
- 4 Architecture and Technology Choices
 - Technology Choices
 - Architecture
- 5 Effective Development
- 6 Summary

Types of Mobile Applications for Apple Devices

- iOS Apps
 - both iPhone and iPad or just one device family?
 - must support all form factors for chosen device family
- watchOS Apps - iWatch
 - need iPhone App for a container

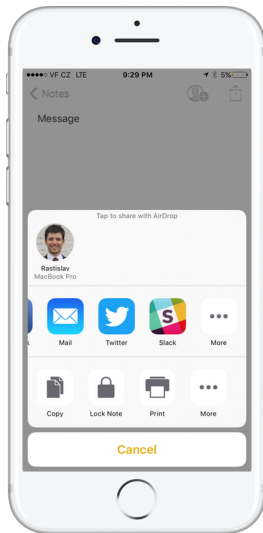
Types of iOS Applications

- standard Apps
- game Apps (out of lecture scope)
- Apps containing App extensions - 20 types e.g.:
 - Action
 - Custom Keyboard
 - Photo Editing
 - Today (Widgets)
 - iMessage, Sticker Packs
 - Call

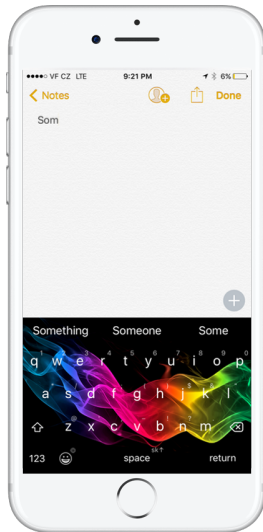
Today Extension Example



Action Extension Example



Keyboard Extension Example



iMessage Extension Example



Cons of Being iOS Developer

- tied to Apple ecosystem
- restricted access to device capabilities and APIs
- constraints on how should App work (iOS HIG, etc.)
- Swift is open source but Objective-C and frameworks are not
- XCode
- higher development setup costs

App Reviews

- detailed, several days
- testing for frauds, crashes, usage of private APIs, breaking rules

Requirements for iOS/watchOS Application Development

- Hardware requirements:
 - Mac computer
 - Phone or iPad device for testing
- Apple Developer account (\$99 per year).
- Will to explore the unknown and exciting :)

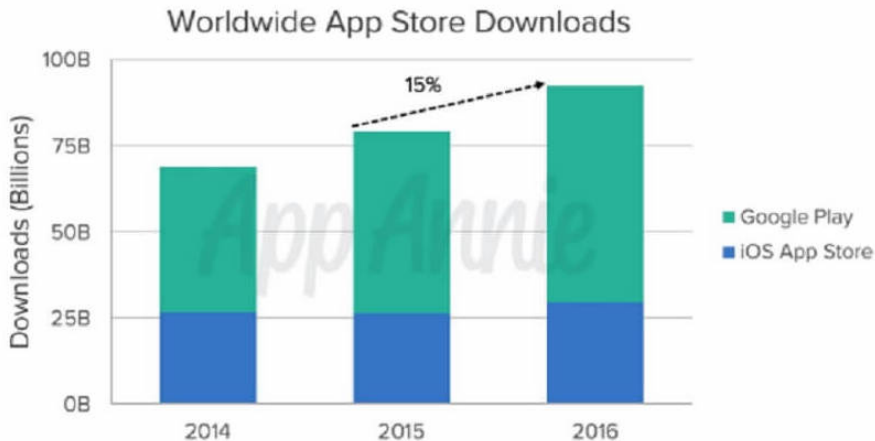
Example

- Mac mini 15 490 Kč
- iPhone SE 12 990 Kč
- approx. 31 000 Kč with Apple Developer account

How to Lower setup costs

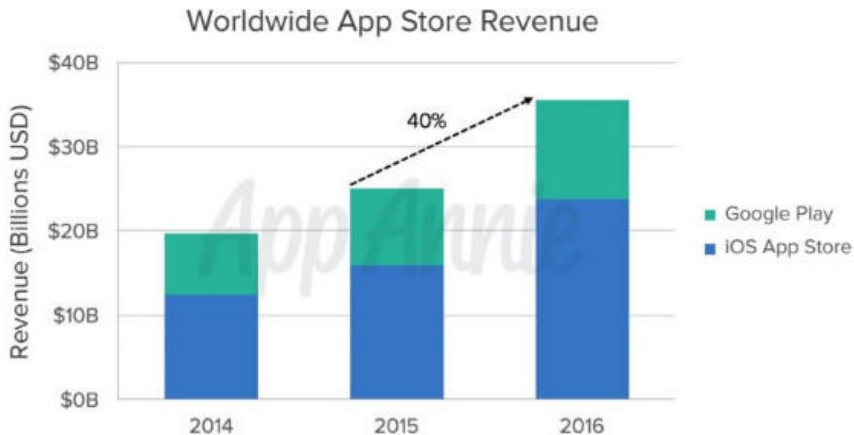
- computers at the Faculty
- XCode simulator
 - no camera, GPS, some third-party APIs do not work e.g. Facebook login
- second-hand testing devices
- short-term device rental
- apply for student developer account

Estimated Worldwide Mobile App Downloads Growth



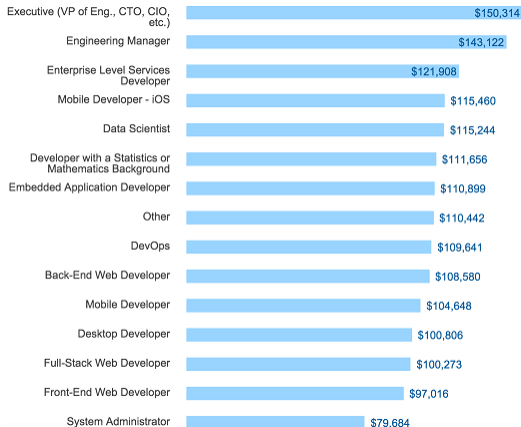
Source: App Annie

Estimated Worldwide Mobile Apps Revenue Growth



Source: App Annie

Estimated US Software Engineering Salaries in 2016



Source: Stack Overflow

Pros of Being iOS Developer

- still fast growing market
- strong wave of mobile-first startups ensures future growth
- number 1 mobile platform in US
- higher income audience, higher revenue, higher salaries
- mobile development is growing fast in ČR/SR, resulting in a shortage of developers
- Swift

App Store Environment

- excluded content, ranked content (e.g. gambling), granted by Apple
- good support, no scam or duplicated apps, rights infringements, etc.

Key Differences to Android

Abstracting from specific technological differences:

- users' behaviour: fewer users, but spending approx. 2x much on Apps
- consistent hardware - 8 devices vs. thousands
- typically need to support just last 2 iOS versions
- these factors result in approx. 30% shorter development time

Common Application Success Factors

- Apps with native UI have higher success ratio
- Startups often aim to deliver too many features in first version
- Focus on releasing the first version quickly then iterate frequently

Important Factors

- 1 App quality (stability, responsiveness) and features
- 2 Getting on the market early
- 3 User experience
- 4 Market Research and Marketing
 - adding Facebook API to App helps to reduce Facebook Ads cost

Planning Monetisation

Apple typically has 30% margin. What about Google Play?

- 1 Paid Apps
 - fixed price tiers, same price in all countries
- 2 In-App Purchases (freemium model)
 - selling via website is possible but it may not be linked from App
- 3 Subscriptions
- 4 Auto-Renewable Subscriptions
 - Apple's share drops to 15% after first subscription year
- 5 Apple Pay
 - real world products only
 - Apple's margin is approx. 2%
- 6 In App Ads: iAds, Facebook Ads, Google Ads, ...
- 7 Affiliates, B2B (most profitable model)

Taking Advantage of Device and iOS Capabilities

- Touch ID - fingerprint sensor
- Force Touch - tap force sensor
- SiriKit - voice commands
- Search Kit
- Health Kit - fitness Apps, health Apps
- Push Notifications, Proactive Suggestions (notifications based on user behaviour patterns)
- Accessibility, Dynamic Type, App Extensions, ...

Releasing to the App Store

Required by App Store

- 1 support web page
- 2 screenshots, description, keywords
- 3 privacy policy in case of App for children
- 4 age rating
- 5 export compliance for Apps using cryptography

Common App Refusal Reasons

- not possible to report user-generated content in the App
- critical bugs, crashes
- not complying with HIG, iOS marketing guidelines

Releasing to the App Store

Beta Testing

- via TestFlight (by Apple), Fabric and similar
- almost no restrictions on tested Apps

Native Development Alternatives

Xamarin C#

React Native Javascript, HTML, CSS

Ionic Angular, Javascript

Advantages and Disadvantages

- write once, run everywhere
- some components might not be available
- another layer between your code and OS - might introduce bugs, lower performance

Swift vs. Objective-C

Swift Pros

- much newer, modern syntax and features, less verbose, no header files
- build with functional programming in mind: Closures (lambda expressions) are first-class members
- has generics, strongly typed language
- does not have null pointer
- syntax similar to major languages, such as Java or C#
- faster development

Objective-C Pros

- more stable, Swift changes syntax a little with every major release
- still has more libraries
- XCode's support for Swift is still very limited

Swift vs. Objective-C

Rule of thumb: Use Swift for new projects, Objective-C for legacy projects already written in it.

Example (Objective-C Method Signature)

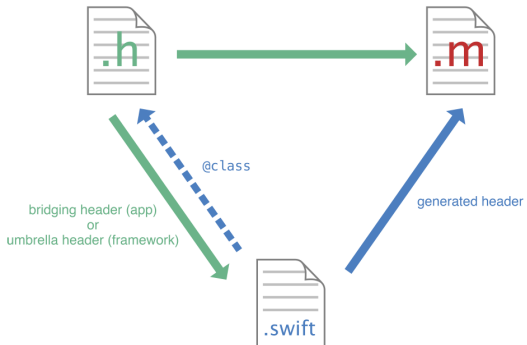
```
+ (NSString *)myMethod:(NSString *)firstArg and:(NSArray *)secondArg { ... }
```

Example (Swift Method Signature)

```
func myMethod(firstArg: String, secondArg: [Int]) -> String {  
    ...  
}
```

Swift Demo

Swift and Objective-C Interoperability



Source: Apple

Swift's Most Innovative Feature

First popular language to solve Tony Hoares Billion Dollar Mistake.

Tony Hoare's 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()
```

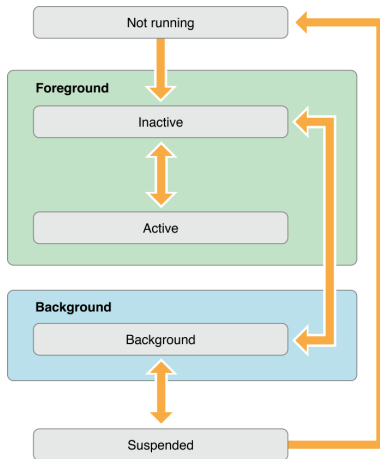
- Optionals are in fact generic enums.
- By not force unwrapping them but using `if let unwrapped = optional` or `guard let unwrapped = optional` a lot of errors can be avoided.

Swift 2 vs. Swift 3

Highest impact changes:

- 1 All function parameters have labels unless otherwise specified.
- 2 Removed for each cycle.
- 3 Removed ++ and -- operators.
- 4 lowerCamelCase for enums and properties.
- 5 Better import of C functions.

App Lifecycle



Source: Apple

App Lifecycle

App state changes can be tracked using `UIApplicationDelegate`'s methods:

- `application:willFinishLaunchingWithOptions:`
- `application:didFinishLaunchingWithOptions:`
- `applicationDidBecomeActive:`
- `applicationWillResignActive:`
- etc.

General Architecture Tips

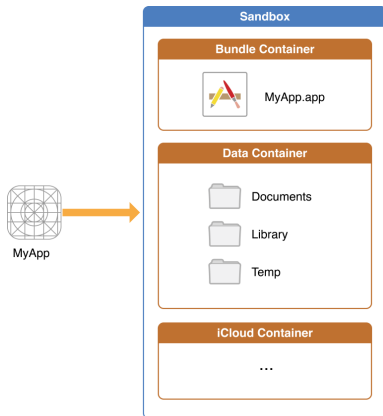
- Model-View-Controller:
 - Keep logic out of both Model and View
 - Extract logic from controllers to separate services layer
 - Wrap data persistence into separate layer with API, independent of persistence technology used
- If you are building complex App, split your code into several XCode projects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

Data Persistence

Commonly used persistence technology:

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

Security Aspects of Data Persistence



Application sandbox. Source: Apple

Security Aspects of Data Persistence

For security purposes, an iOS app's interactions with the file system are limited to the directories inside the app's sandbox directory.

- App typically cannot access data outside of its sandbox.
- App's sandbox is encrypted. Other apps cannot manipulate app's data.
- App can request App Group permission.
 - Apps in the same group have access to shared sandbox.
 - They need to have the same group ID.
 - Useful for app extensions, apps from the same developer, ...

Talking to Backend

- Common backend choices:
 - Firebase
 - Parse
 - CouldKit
 - Realm
 - custom server
- Move long running tasks to background threads.
 - GCD (DispatchQueue) vs. NSOperation/NSOperationQueue
 - QoS

Development Tools

- Xcode
 - no good alternatives
 - limited Swift support
 - no refactor, no advanced features
 - unstable code highlighting
 - works well with storyboards, XIBs, localisation, assets
- XCode Instruments
 - advanced debugging and profiling
- CocoaPods
 - dependency manager for Objective-C and Swift
 - reliable, standard

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.

Animating Layout Changes

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

Adaptive Auto Layout Demo

Advanced UICollectionView usage

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.

Other Development Tips

- 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.

Other Development Tips

- Work with your designer(s); do not hesitate to tell them if any design is hard to implement.
- There are ready-to-use controllers for camera, image library, email, sharing, ...
- Consider using Facebook API, Google API, Firebase for: User tracking, bug reporting, login, notification delivery, etc.
- 2 things to avoid: Allowing rotation for just for screens of the app and accessing private APIs or properties (e.g. via KVC).

Questions?

Thank you