307: Architecting for Multiple Platforms

iPhone, iPad, Watch, TV

raywenderlich.com

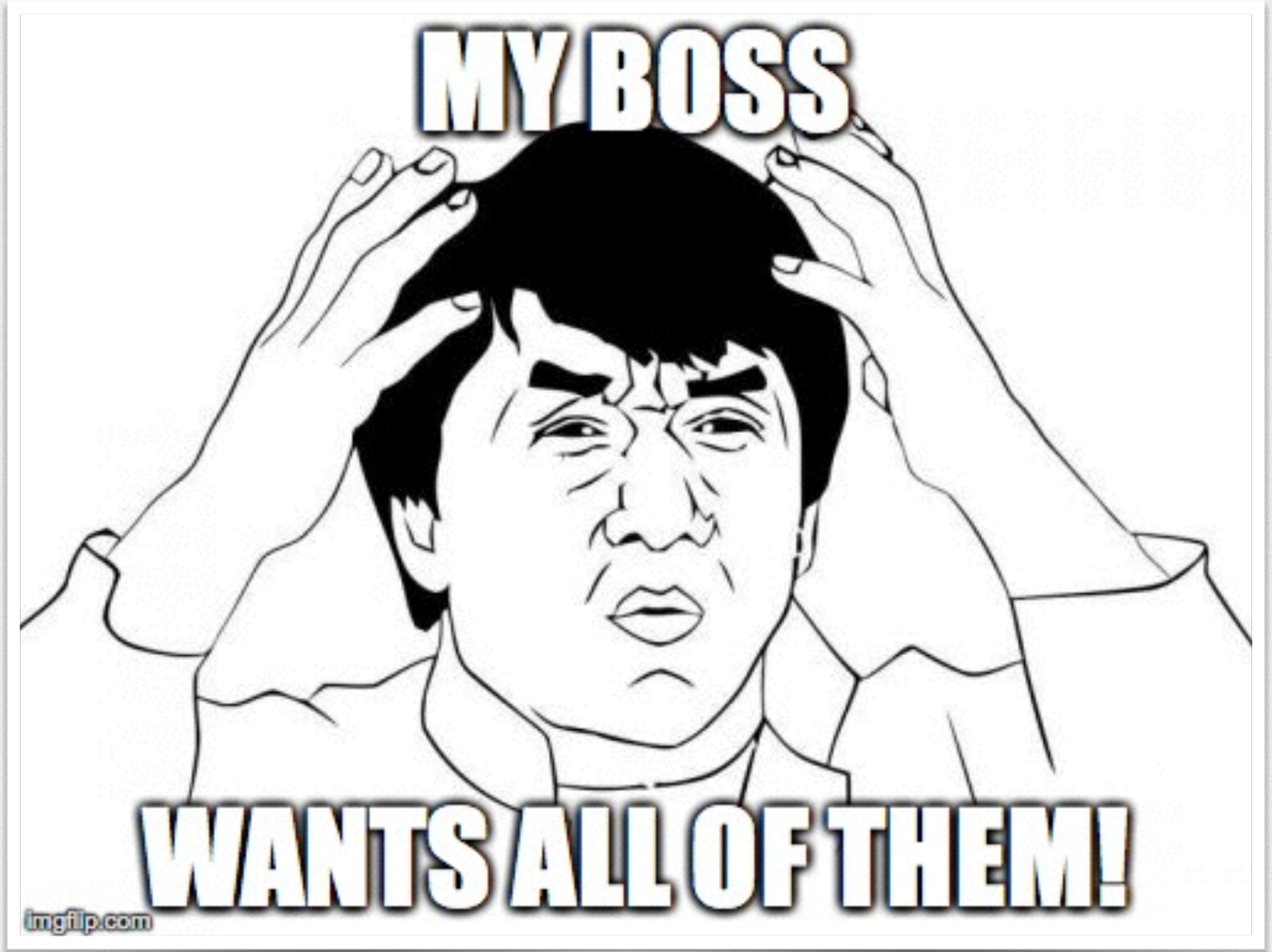
2010-2015

- You probably programmed apps for iPhone.
- You might have programmed apps for iPad.
- Life was simple.

Today

Rented
Top Films
LOOPER
breaking dawn
LAWLESS
SEARCHING FOR SUGAR MAN
THE IMPOSTER
Films
TV Programmes
Music
Computers
Settings
NETFLIX
MLB.com
sky NEWS

Activity
MOVE
EXERCISE
STAND
15:00


MY BOSS
WANTS ALL OF THEM!
imgflip.com

Cross Platform Themes
* What you'll learn today

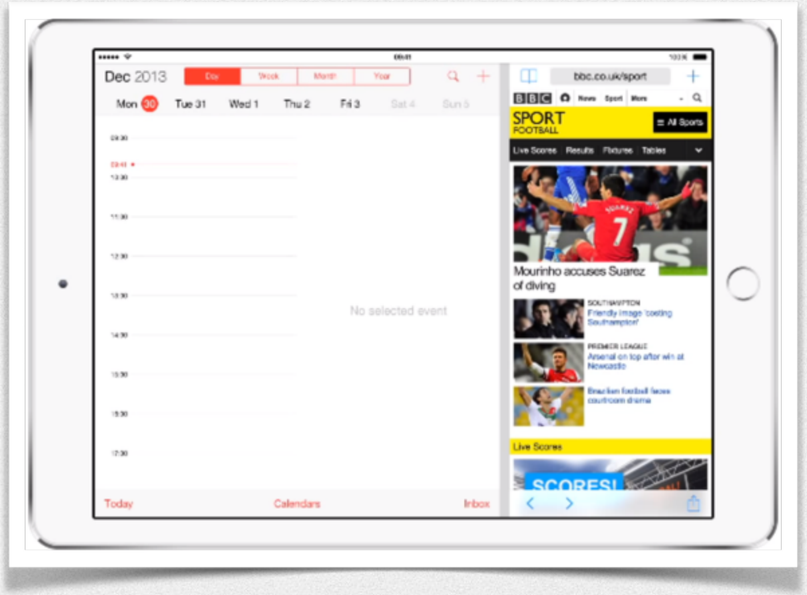
- Sharing Code
- Sharing Bits (i.e. Frameworks)
- Don't Repeat Yourself (DRY)
- Minimize Code/Debug Cycles

Cross Platform Concerns

- Screen Sizes
- Shared Code and Consistency - DRY (Don't Repeat Yourself)
- App vs Extension
- Code/Debug Cycle

Cross Platform Concerns
Screen Sizes

- Current generation iPhones and iPads now come in many sizes
- Some support landscape (iPhone 6+)
- Some iPads support split-screen



Cross Platform Concerns
Sharing Code

- Your apps may share **very** similar behaviors
- Sometimes difficult when platform **X** has slightly different behavior than platform **Y**
- Temptations to “copy/paste/tweak”.

```
class VoteViewController: UIViewController {
  override func viewDidLoad() {
    button.backgroundColor = .greenColor()
  }
}
```

```
class TVVoteViewController: UIViewController {
  override func viewDidLoad() {
    button.backgroundColor = .greenColor()
  }
}
```

```
class VoteInterfaceController: WKInterfaceController {
  override func awakeWithContext(context: AnyObject?) {
    button.setBackgroundColor(.greenColor())
  }
}
```

Cross Platform Concerns
Platform Capabilities

- Limited API's for app extensions
  - `sharedApplication`
  - Camera APIs
- iPhones can't use an Apple Pencil.
- TV's can't use the camera or notifications.
- ... But they can all (except watch!) use `UIViewController`!

Cross Platform Concerns
Code Debug Cycle

- Write once, ~~run everywhere~~ break everything!
- Building iPhone in a debug environment won't build TV or Watch.
- Code/debug cycles scale up with number of targets.
- May code for hours on one platform without realizing what broke on another.

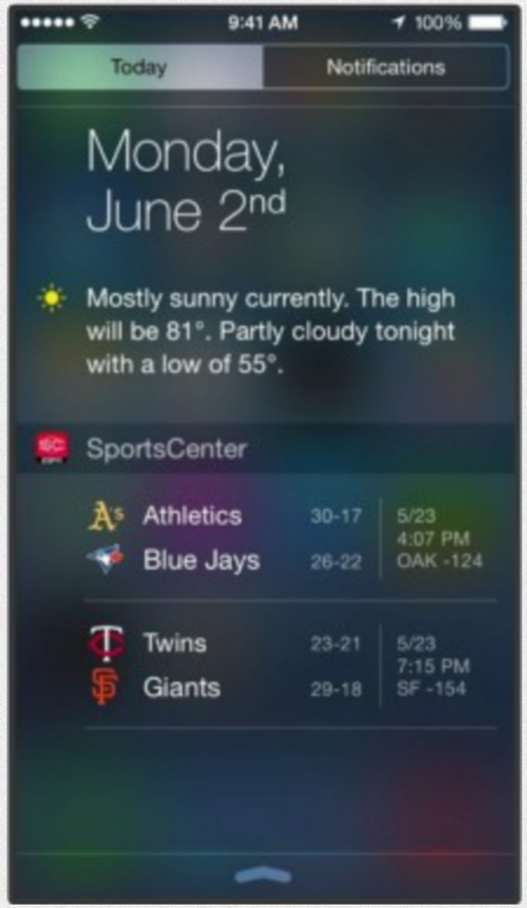
A Tour of the iOS Family

Start with iPhone and iPad

- What you're (likely) used to!
- Shared views, view controllers, models, networking etc.
- Single **target** and **bundle ID.**
- Use AutoLayout and Size Classes

App Extensions

- iOS and UIKit
- Packaged with “containing” app
- Limited APIs
- Code sharing with iPhone/iPad



Apple TV

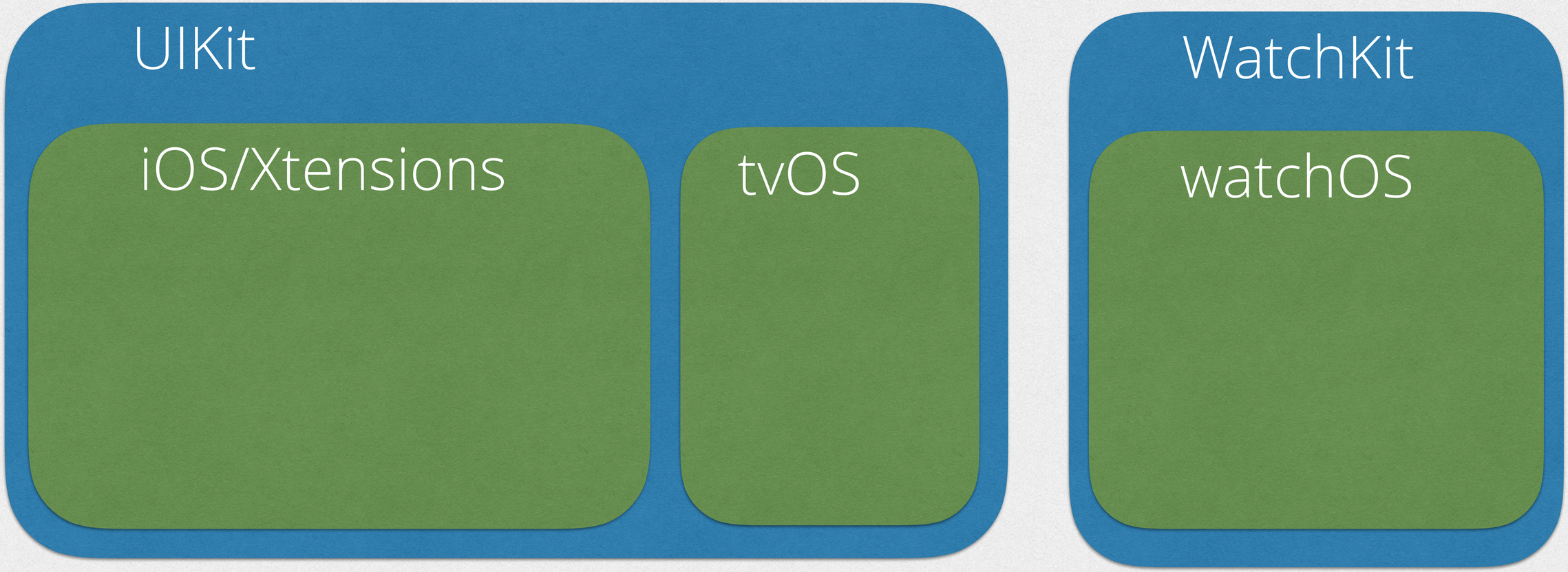
- **tvOS** and **UIKit**
- **Very large** overlap between **iOS** and **tvOS**.
- tvOS ≠ iOS
- Binary Compatibility

Apple Watch

- **watchOS and WatchKit**
- The least-like every other platform
- No UIKit
- Not binary compatible with iOS

Quick Recap

Compatibility Map



Binary Compatible

Code Compatible

Code Compatibility
Plain-Old-Swift-Objects (POSO's!)

- iOS - ✅
- iOS Extensions - ✅
- tvOS - ✅
- watchOS - ✅

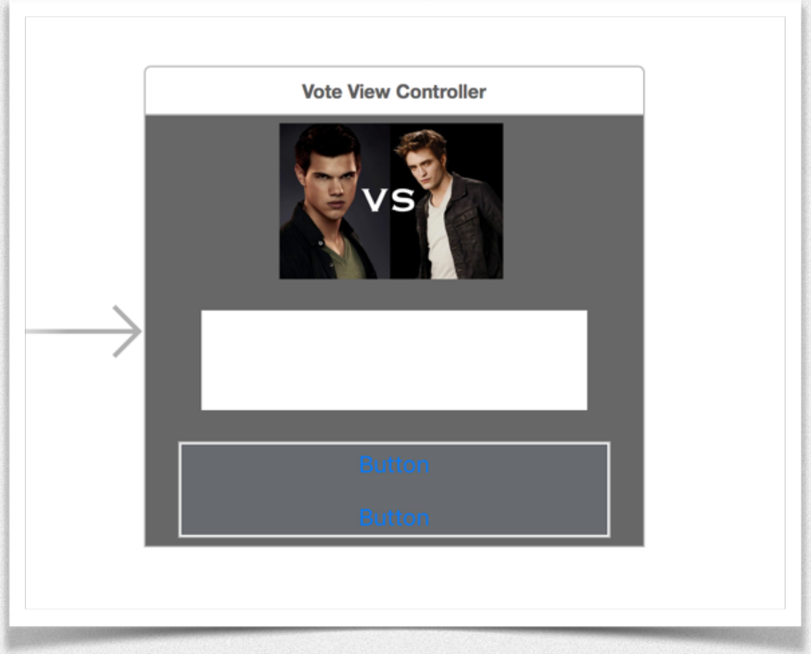
```
import Foundation

struct Answer {
  let answer: String
  let answerId: Int
}
```

Code Compatibility
UIKit

- iOS - ✅
- iOS Extensions - ✅
- tvOS - ✅
- watchOS - ❌
  - (Some UIKit namespace overlap)

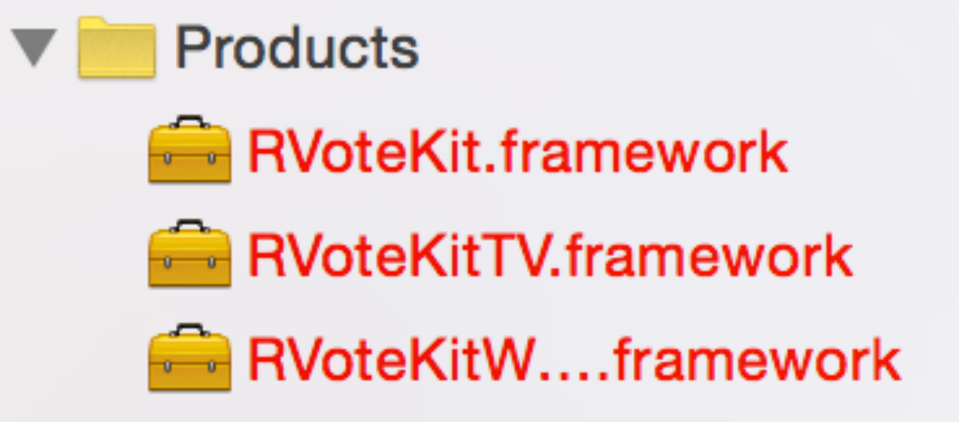
Storyboard Compatibility

- iOS - ❌
- iOS Extensions - ❌
- tvOS - ❌
- watchOS - ❌
- **Storyboards cannot be shared across any platform :[**



Binary Compatibility

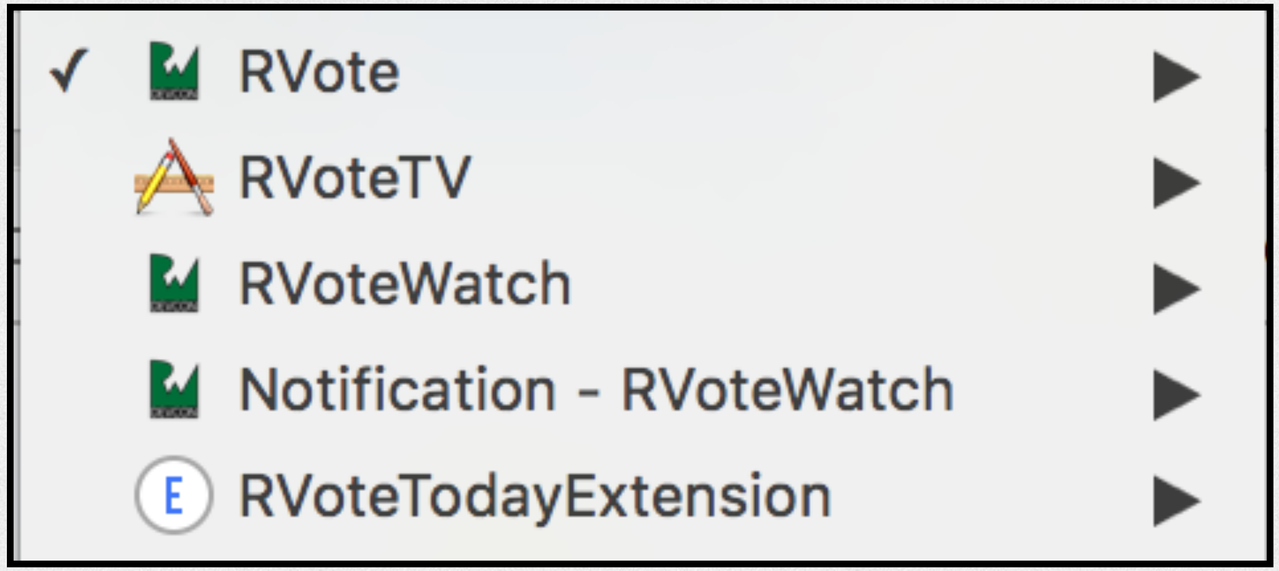
- iOS -
- iOS Extensions -
- tvOS -
- watchOS -



Consequences

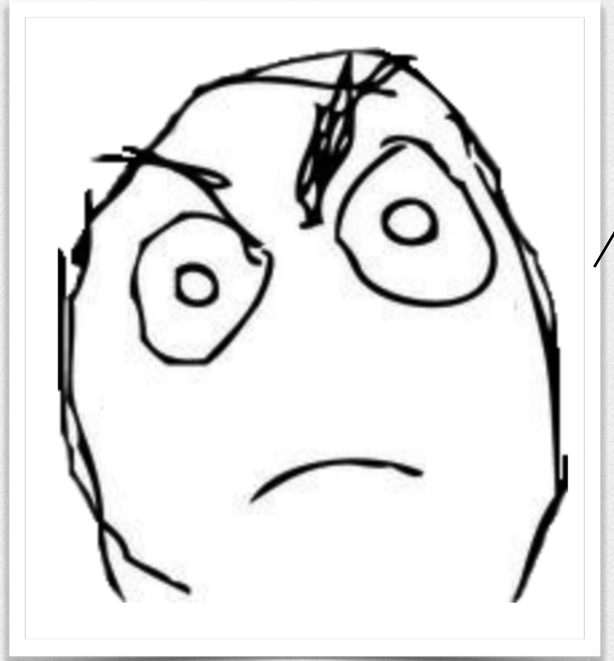
Consequences

- Hit “Play” = ~~*Runs your app*~~ Runs one of your 5 targets
- “Surprise” build errors :[



Consequences

- Hit “Play” = ~~*Runs your app*~~ Runs one of your 5 targets



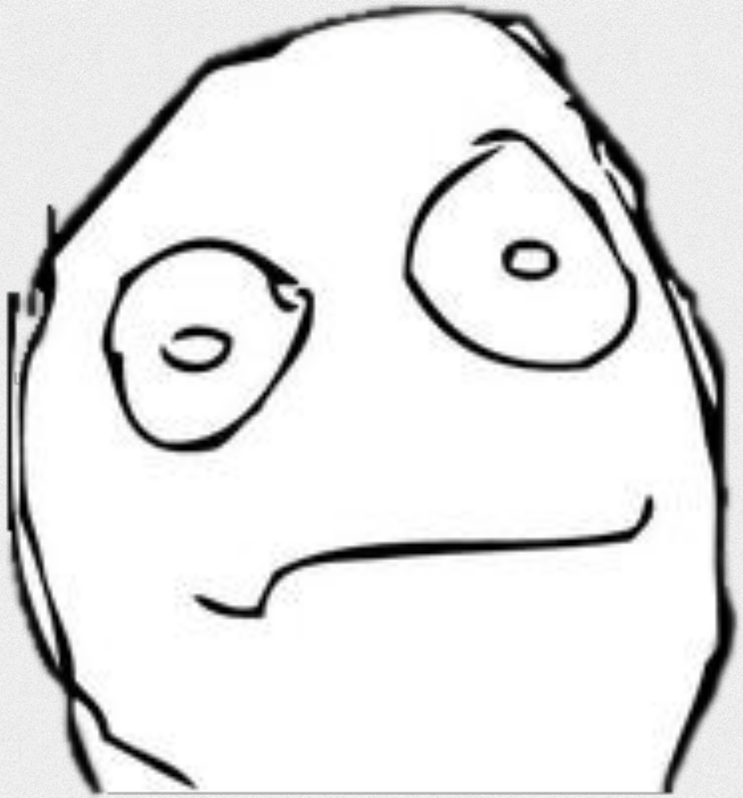
~~*Bill, you broke the build!*~~

Bill, you broke 4 builds!

Consequences
UI

- Some **common** language and frameworks (Foundation, Swift/Obj-C)
- Some **somewhat common** frameworks (UIKit)
- Multiple UI paradigms (touch, focus, watch)

Yikes!



How To Manage?

How to Manage?

- How to maintain **consistent visual design**?
- How to **maximize code reuse**?
  - ...while leveraging **platform-specific features**?
- How to **safely** write reusable code?

Maximize Code Reuse

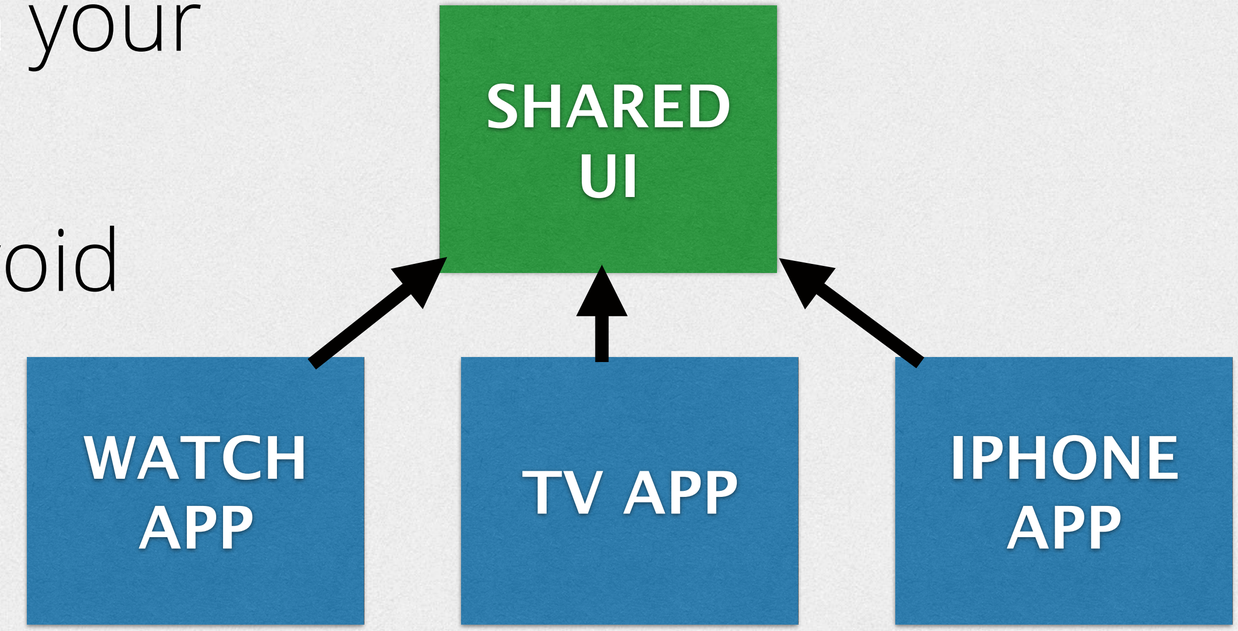
Maximize Code Reuse

- All of **Foundation** can be used across all platforms.
  - Share networking and model code!
- **View Controllers** can be shared across iOS, tvOS, and (some) extensions.
- Of course, ordinary Swift / Objective-C code can be shared across all platforms.

Visual Consistency

Visual Consistency

- Leverage a shared UI framework
- Build UI separate from your core app
- Minimize repeating, avoid bugs!



Playgrounds

- Playgrounds allow you to render your interface in a (mostly) platform-agnostic environment.
- Preview controls, color themes, or even entire view controllers.
- Minimal code/debug cycles. Just edit code and watch the updates!
- (BONUS!) Forces you to write decoupled UI code.

Playgrounds!

```
import UIKit
import RVoteKit
import XCPlayground

let button = NSBundle(forClass: VoteButton.self).
    loadNibNamed("VoteButton", owner: nil, options: nil).
    first as! UIButton

button.frame = CGRect(x: 0, y: 0, width: 100, height: 50)

let view = UIView(frame: CGRect(x: 0, y: 0, width: 100,
    height: 50))
view.backgroundColor = UIColor.whiteColor()

view.addSubview(button)

XCPlaygroundPage.currentPage.liveView = view
```

Playgrounds!

```
8  Clay.dark                    r 0.051 g 0.076 b 0.13 a 1.0
9  Clay.cream                   r 1.0 g 0.926 b 0.87 a 1.0
10 Clay.green                   r 0.772 g 0.85 b 0.425 a 1.0
11 Clay.brown                   r 0.53 g 0.385 b 0.366 a 1.0
12 Clay.teal                    r 0.686 g 0.88 b 0.809 a 1.0
13
14 UIColor.text                 r 0.051 g 0.076 b 0.13 a 1.0
15 UIColor.background           r 1.0 g 0.926 b 0.87 a 1.0
16 UIColor.highlight            r 0.772 g 0.85 b 0.425 a 1.0
17 UIColor.buttonEnabled        r 0.772 g 0.85 b 0.425 a 1.0
```

Safely Writing Reusable Code

Carefully Decorate with Platform Specific Capabilities

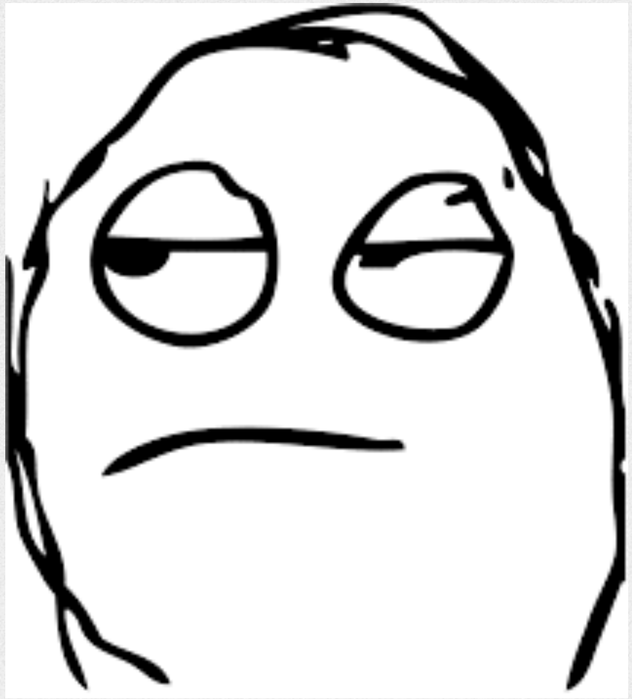
- Reference **sharedApplication()** - **Bad :[**

```
func scheduleVoteReminder() {
  let notification = UILocalNotification()
  notification.fireDate = NSDate()
  notification.alertBody = "Check out your new polls!"
  UIApplication.sharedApplication()
    .scheduleLocalNotification(notification)
}
```

Carefully Decorate with Platform Specific Capabilities

- Guard with preprocessor - **Better**

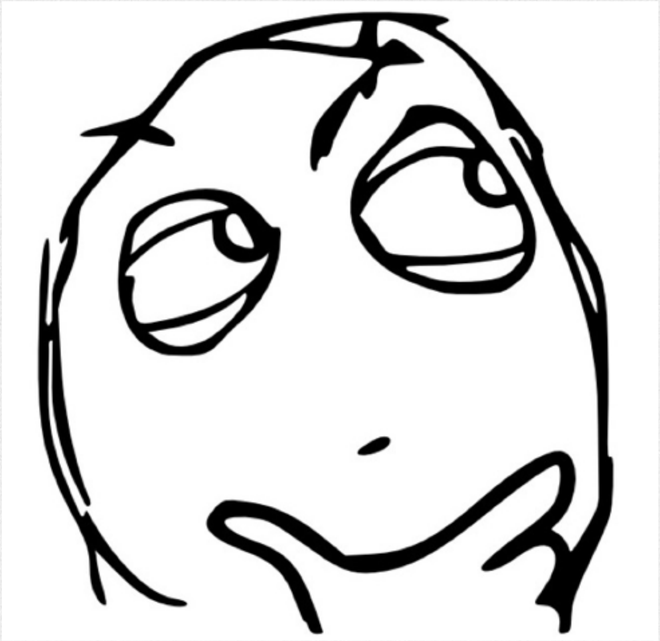
```
  func scheduleVoteReminder() {
#if IOS
    let notification = UILocalNotification()
    notification.fireDate = NSDate()
    notification.alertBody = "Check out your new polls!"
    UIApplication.sharedApplication()
      .scheduleLocalNotification(notification)
#endif
  }
```



Carefully Decorate with Platform Specific Capabilities

- What about protocol extensions?

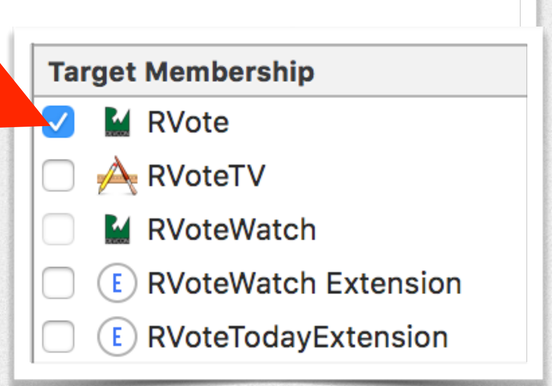
```
protocol VoteSchedulable {
  func scheduleVoteReminder()
}

extension VoteSchedulable {
  func scheduleVoteReminder() {
    // No action by default
  }
}

class MyViewController: VoteSchedulable {
  func vote() {
    // Code to record vote
    // ...

    scheduleVoteReminder()
  }
}
```

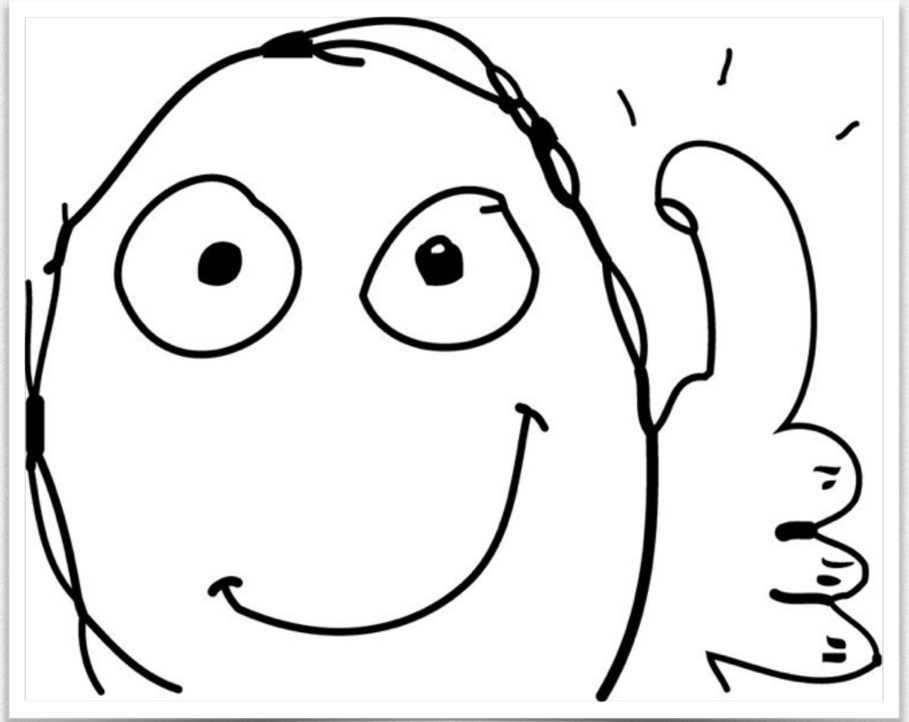


Carefully Decorate with Platform Specific Capabilities

- Implement on an extension - **Best!**

```
extension MyViewController {
  func scheduleVoteReminder() {
      let notification = UILocalNotification()
      notification.fireDate = NSDate()
      notification.alertBody = "Check out your new polls!"
      UIApplication.sharedApplication()
        .scheduleLocalNotification(notification)
  }
}
```

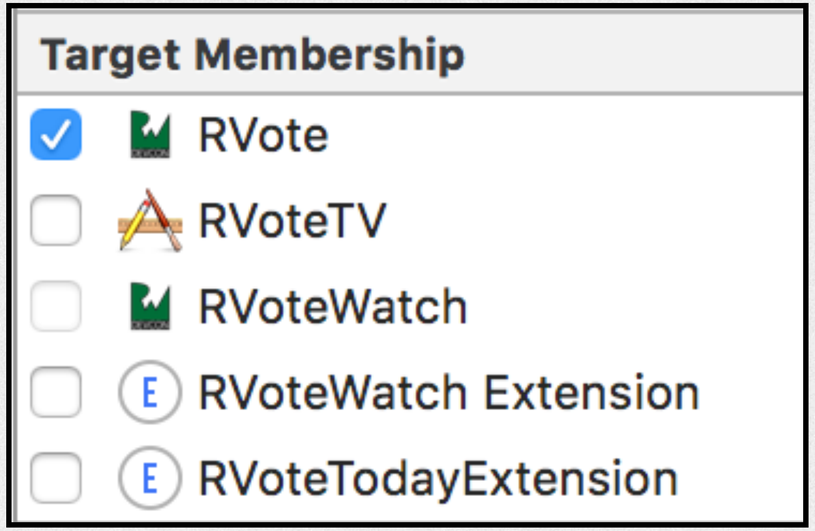




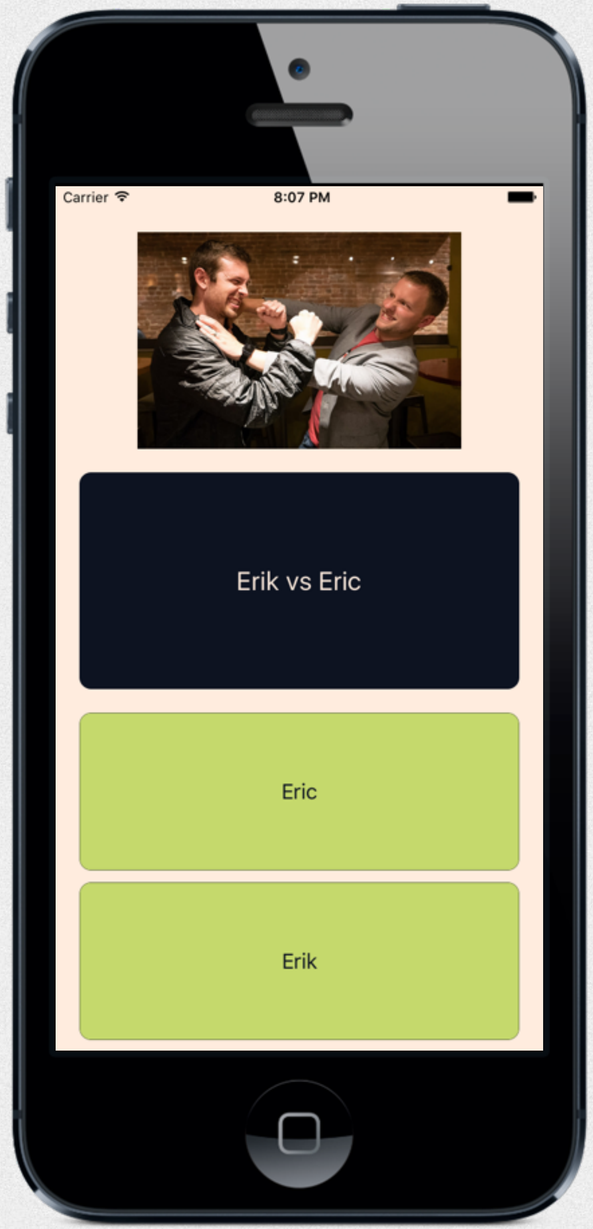
Protocol Oriented Programming To The Rescue!

- Protocol extensions - and Protocol-Oriented Programming - allow you to “decorate” a target with specific behaviors.

- Write an extension, and include it only in the targets you want to use that extension in!



Demo


Carrier
8:07 PM
Erik vs Eric
Eric
Erik

307: Architecting for Multiple Platforms

Conclusion

What You Learned

- That there were probably more targets than you thought!
- How to share code between many targets.
- How to minimize code/debug time with playgrounds.

Where To Go From Here?

- Twitter: @erikkerber
- RWDevCon
  - 305: App Architecture
  - 301: Advanced Auto Layout & Adaptive Layout
- App Extension Guide: http://apple.co/1Ffdhqe
- Watchkit By Tutorials: http://www.raywenderlich.com/store/watchkit-by-tutorials
- tvOS Apprentice: http://www.raywenderlich.com/store/tvos-apprentice