Professional iOS App Architecture

Justin Williams

Justin Williams

I write code for money in Colorado.

I don't do demos.

But I'll give you a full sample project at the end for sticking around.

Why This Talk?

The Auto Layout Trilogy

- 2013: Achieving Zen With Auto Layout
- 2014: Stupid Auto Layout Tricks
- 2015: Mastering Auto Layout
- 2016: ???


Carrier
1:21 PM
TED
特集
人気
日本語
フィリップ・ローズデール
セカンドライフでの暮らし
サラ・グレイ
短命だった息子は新しい未来への道を開く
ジル・ファラント
干ばつに耐えられる農作物の作り方
ウリ・ハッソン
コミュニケーション中の脳の反応
ケネス・ラコバラ
恐竜の化石探しを通して知った、宇宙における人類
1:22 PM
100%
人気
ドイツ語
Jedidah Isler
Wie ungenutztes Talent die Wissenschaft verbessern k...
Sanford Biggers
Der unbeirrbare Blick eines Künstlers auf rassistische...
Raffaello D'Andrea
Entdecken Sie die glanzvollen Fluggeräte der...
Trevor Timm
Wie frei ist die US-Pressefreiheit?

TED 1.0

- 100% Objective-C
- Manual memory management
- English only
- Completely separate UI (and launch sequence!) for iPhone and iPad
- Designed for the iPhone 3Gs and iPad 1.0

Ch-Ch-Changes

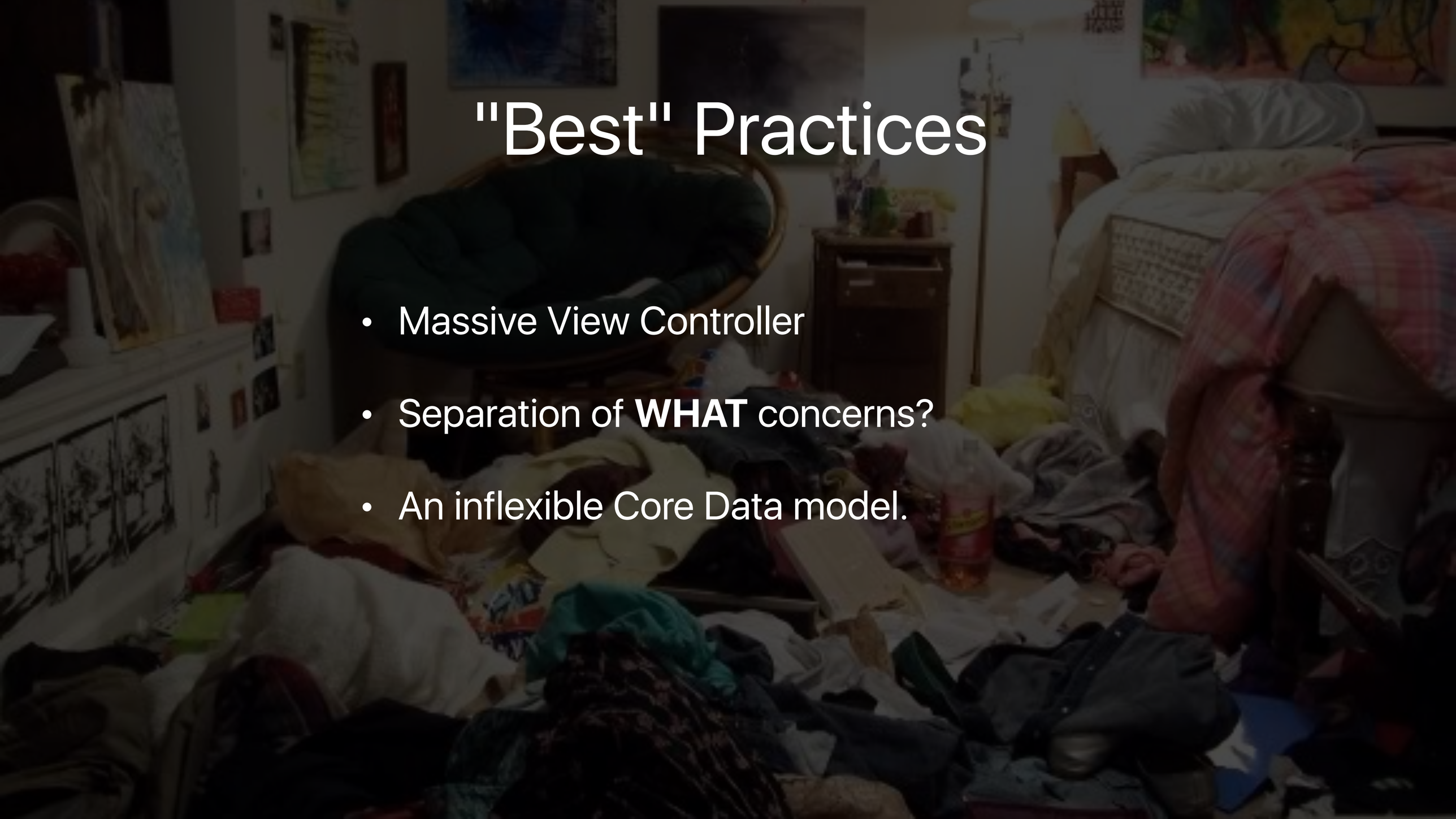
2010	2011	2012	2013	2014	2015	2016
				Extensibility		
		iPhone 5		Size Classes	Obj-C Nullability	
iPhone 4		Auto Layout	iOS 7 Redesign	iPhone 6 / Plus	Swift 2	
2x Graphics	ARC	Switching to HLS / AVFoundation	XCTest	3x Graphics	iPad Pro 12.9	Swift 3

TED 3.0

- 50% Swift 2.2 / 50 % Objective-C
- 100% ARC
- 100% Auto Layout and size classes
- Localized and deployed in 22 languages
- Designed for iPhone 4S up to iPad Pro 12.9"

Not Just Device Capabilities

Best Practices Have Changed Too



"Best" Practices

- Massive View Controller
- Separation of **WHAT** concerns?
- An inflexible Core Data model.

WARNING

Opinionated Developer Incoming!

What We'll Cover

- Framework-first development
- Dependency Management
- Data Parsing & Persistence
- Network Access

Framework Based Design

Dynamic Frameworks

- First introduced on iOS with iOS 8
- First introduced on macOS back when you still used Windows.
- Enables runtime linking of your code.
- Encapsulates resources as well
- Necessary if you are using Swift (ABI Stability)

Why Frameworks?

- Better separation of concerns
- Better testability
- Faster incremental compiliation

Application	Core	Network	Data
View Controllers	Logging	Network Object	Persistence
App Specific Helpers	Settings	API Requests	JSON Parsing
App Extensions	Class Extensions	Operations	Business Logic

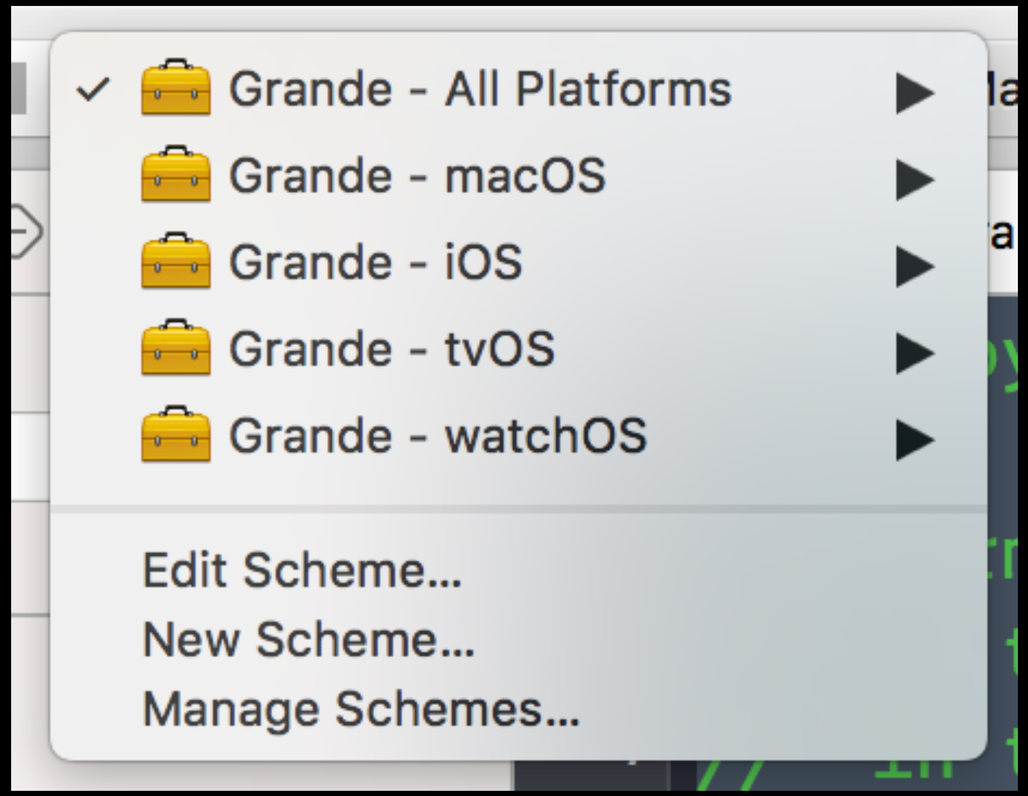
Build For All Platforms

iOS, macOS, tvOS, and watchOS (it’s still a thing!)

Use Schemes

One scheme per target / platform

One catch-all for all platforms on a target.



Precompiled Frameworks

Enable faster compilation by not recompiling rarely changing code constantly.

Dependency Management

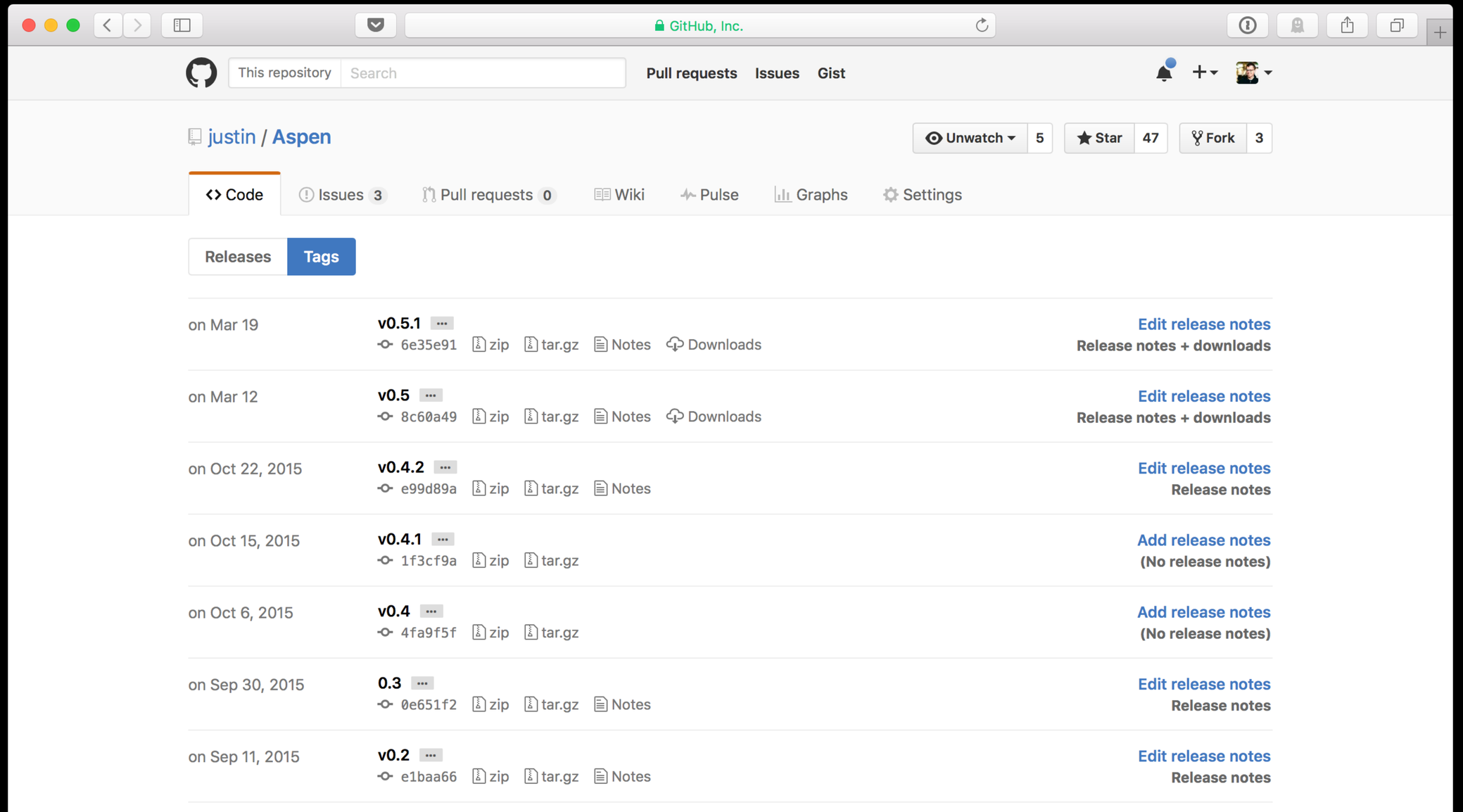
Dependency Management

- Cocoapods
- Carthage
- Living in the stone age

Carthage

- Decentralized
- Based off Git and your Xcode projects.
- Builds dynamic frameworks only
- No “podspec” type files.

Semantic Carthaging



```
carthage build --no-skip-current
carthage archive iDevFramework
```

Dependency Lecture

I’m required to do this by Crusty the Grey Beard.

Things I Use And Like

- Freddy https://github.com/bignerdranch/Freddy
- PINRemoteImage https://github.com/pinterest/PINRemoteImage
- Valet https://github.com/square/Valet

Parsing & Persistence

Core Data

Realm

I like Realm. I use Realm. Usually.

Yay Realm

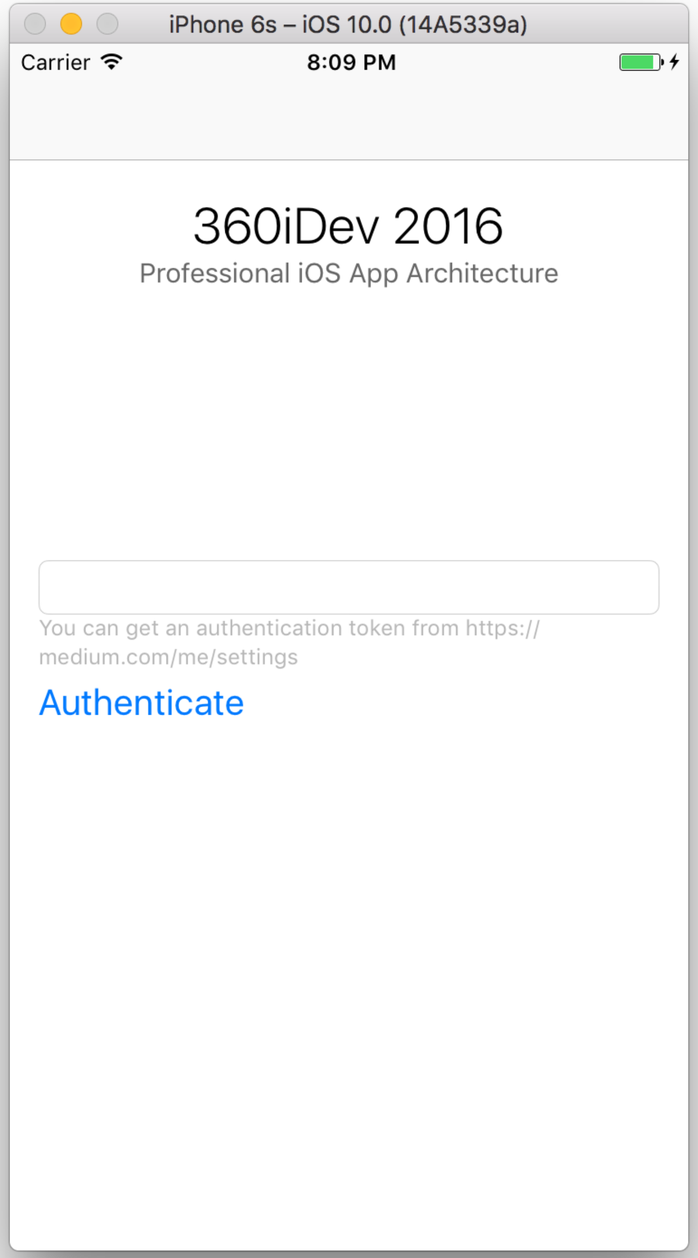
- Easy to adopt
- Native bindings in both Objective-C and Swift
- Easy to work with
- Stable

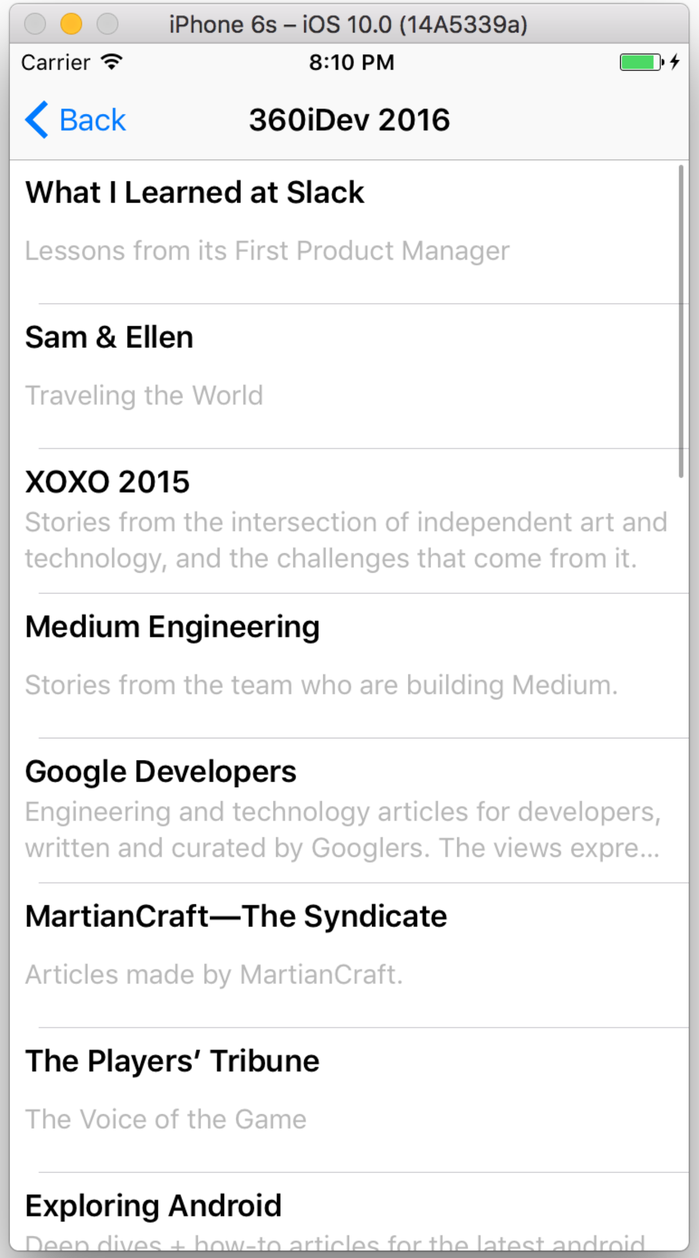
Boo Realm

- Swift ABI stability pain
- It’s a startup!

Let’s Write Some Code

Finally™


iPhone 6s – iOS 10.0 (14A5339a)
Carrier
8:09 PM
360iDev 2016
Professional iOS App Architecture
You can get an authentication token from https://medium.com/me/settings
Authenticate


iPhone 6s – iOS 10.0 (14A5339a)
Carrier
8:10 PM
Back
360iDev 2016
What I Learned at Slack
Lessons from its First Product Manager
Sam & Ellen
Traveling the World
XOXO 2015
Stories from the intersection of independent art and technology, and the challenges that come from it.
Medium Engineering
Stories from the team who are building Medium.
Google Developers
Engineering and technology articles for developers, written and curated by Googlers. The views expre...
MartianCraft—The Syndicate
Articles made by MartianCraft.
The Players' Tribune
The Voice of the Game
Exploring Android

```
/// Used to represent an authenticated Medium account.
public final class User: Object, MediumContent {
    /// A unique identifier for the user.
    public dynamic var id: String = ""
    /// The user's username on Medium.
    public dynamic var username: String = ""
    /// The user's name on Medium.
    public dynamic var name: String = ""
    /// The URL to the user's profile on Medium.
    public var url: URL? {
        return URL(string: urlString)
    }
    fileprivate dynamic var urlString: String = ""
    /// The URL to the user's avatar on Medium.
    public var imageUrl: URL? {
        return URL(string: imageUrlString)
    }
    fileprivate dynamic var imageUrlString: String = ""

    override public static func primaryKey() -> String? {
        return "id"
    }
}
```

```
/// A collection of thoughts, from a Thought Leader (or `User`).
public final class Publication: Object, MediumContent {
    /// A unique identifier for the post.
    public dynamic var id: String = ""
    /// The publication's title.
    public dynamic var title: String = ""
    /// The publication's description.
    public dynamic var synopsis: String = ""
    /// The URL of the publication on Medium.
    public var url: URL? {
        return URL(string: urlString)
    }
    fileprivate dynamic var urlString: String = ""
    /// The URL to the publication's image on Medium.
    public var imageURL: URL? {
        return URL(string: imageUrlString)
    }
    fileprivate dynamic var imageUrlString: String = ""
    override public static func primaryKey() -> String? {
        return "id"
    }
}
```

```swift
do {
    let realm = try Realm()
    try realm.write {
        realm.add(objects, update: true)
    }

    logInfo("Successfully wrote \(objects.count) objects to Realm.")
} catch {
    logError("Error writing to Realm: \(error)")
}
```

```
{
  "data": {
    "id": "asdfasdfasdfasdfasdfasdfasfd",
    "username": "justin",
    "name": "Justin Williams",
    "url": "https://medium.com/@justin",
    "imageUrl": "https://cdn-images-1.medium.com/fit/c/200/200/1*YWgIErDhmCgBtuAiHVkOmg.jpeg"
  }
}
```

```
extension User: JSONDecodable {
    public convenience init(json: JSON) throws {
        self.init()
        id = try json.string("data", "id")
        username = try json.string("data", "username")
        name = try json.string("data", "name")
        urlString = try json.string("data", "url")
        imageUrlString = try json.string("data", "imageUrl")
    }
}


extension Publication: JSONDecodable {
    public convenience init(json: JSON) throws {
        self.init()
        id = try json.string("id")
        title = try json.string("name")
        synopsis = try json.string("description")
        urlString = try json.string("url")
        imageUrlString = try json.string("imageUrl")
    }
}
```

```
do {
    // Check if we are getting a dictionary or an array.
    if let data = try? json.array("data", alongPath: .NullBecomesNil) {
        parsedObjects = try data!.map(T.init)
    } else if (try? json.dictionary("data", alongPath: .NullBecomesNil)) !=
        nil {
        let result = try T(json: json)
        parsedObjects = [result]
    }
} catch {
    logError("Error parsing JSON: \(error)")
    if let completion = completionHandler {
        completion(Result.failure(error))
    }
    isFinished = true
    return
}
```

Network Requests & Operations

Network Access

- Result based
- Protocol oriented
- Powered by operation queues

Result Based

```
public enum Result<T> {
    case success(T)
    case failure(Error)
}
```

```
guard let token = tokenTextField.text else { return }

let config = URLSessionConfiguration.default
config.allowsCellularAccess = true
config.networkServic[Go Forward].default
let session = URLSession(configuration: config)

network = NetworkAPI(accessToken: token, session: session)
_ = network.getProfile() { (result) in
    switch (result) {
    case .success(_):
        self.showPublications()
        break
    case .failure(let error):
        print(error)
        break
    }
}
```

Protocol Oriented

- Each request is built off a APIRequest
- GeneratedRequest adds construction of a URLRequest
- Pass the request into your network operation.

```
protocol APIRequest {
    var baseURL: NSURL? { get }
    var method: String { get }
    var path: String { get }
    var parameters: Dictionary<String, String> { get }
    var headers: Dictionary<String, String> { get }
    var httpBody: HTTPBody? { get }
    var accessToken: String { get }
}
```

```swift
protocol GeneratedRequest: APIRequest {
    func constructRequest() -> URLRequest?
}

extension GeneratedRequest {
    func constructRequest() -> URLRequest? {
        guard let baseURL = baseURL else { return nil }
        guard let URLComponents = NSURLComponents(url: baseURL as URL,
            resolvingAgainstBaseURL: true) else { return nil }
        URLComponents.path = (URLComponents.path ?? "") + path
        URLComponents.queryItems = parameters.map { key, value in
            return URLQueryItem(name: key, value: value)
        }

        guard let URL = URLComponents.url else { return nil }
        let request = NSMutableURLRequest(url: URL)

        if let body = httpBody {
            request.httpBody = body.encoded()
        }

        request.addValue("Bearer \(self.accessToken)", forHTTPHeaderField:
            Constants.authorization)
        request.addValue(Constants.applicationJSON, forHTTPHeaderField: Constants.
            accept)
        request.addValue(Constants.applicationJSON, forHTTPHeaderField: Constants.
            contentType)
        request.httpMethod = method
        return request as URLRequest
    }
}
```

```swift
import Foundation

struct GetPublicationsRequest: GeneratedRequest {
    internal let userID: String

    // MARK: APIRequest Overrides
    // ====================================
    // APIRequest Overrides
    // ====================================
    var path: String
    var accessToken: String

    // MARK: Initialization
    // ====================================
    // Initialization
    // ====================================
    init(userID: String, accessToken: String) {
        self.accessToken = accessToken
        self.userID = userID

        self.path = "/users/\(userID)/publications"
    }
}
```

```swift
/**
 Retrieve the publications a user has access to.

 - parameter userID: The user's unique identifier.
 - parameter completionHandler: The optional completion handler that will return
     a `Result` instance with either a `User` object or an error.
 */
public func getPublications(userID: String, completionHandler:
    MediumAPIOperationCompletionHandler?) -> Operation {
    let template = GetPublicationsRequest(userID: userID, accessToken:
        accessToken)
    let operation = MediumAPIOperation<Publication>(requestTemplate: template,
        session: session, completionHandler: completionHandler)
    addToQueue(operation: operation)
    return operation
}
```

MediumAPIOperation

- Combines three different sub-operations.
- Network Access
- JSON Parsing
- Persistence

More On Operations

- One queue for user interactive requests and one for background operations.
- Queue choice is determined by NSQualityOfService
- An internal queue manages compound operations
- Operations can return values, or just persist and forget it

Even More On Operations

- Adjust the concurrency based on network quality (if network goes bad, ensure that the most important stuff goes through)
- Listen to the operation's isExecuting, isFinished and isCancelled state to show UI to the user.

I told you there would be a GitHub repo.

https://github.com/justin/iDev2016