

Open Source

An overview of why open source is key to building mobile software.

By: Conrad Stoll, @conradstoll

Architect, Mutual Mobile

How do you build an application?

How do I participate in the community?

Learning about Dependencies

Shared Problems Deserve

Shared Solutions

What sort of problems?

Communication over a network

Responsive User Interface

Data Persistence

Testing

Dependency Management

**Great Open Source
software**

**Very applicable to other
people**

**Sometimes it takes
multiple people**

To solve a hard problem

What can you do with Open Source?

1. You can use it

2. You can contribute to it

3. You can make your own

1. Using Open Source

Open Source tools help you focus on the real work you need to do.

Focus on solving the key problems for your project.

Web Frameworks

Django

Rails

Node

Angular

HTTP Networking



AFNetworking

AFNETWORKING

OkHttp

OkHttp

User Interface

Challenges Building User Interfaces

Animation

Animation conveys meaning and makes your app more delightful.

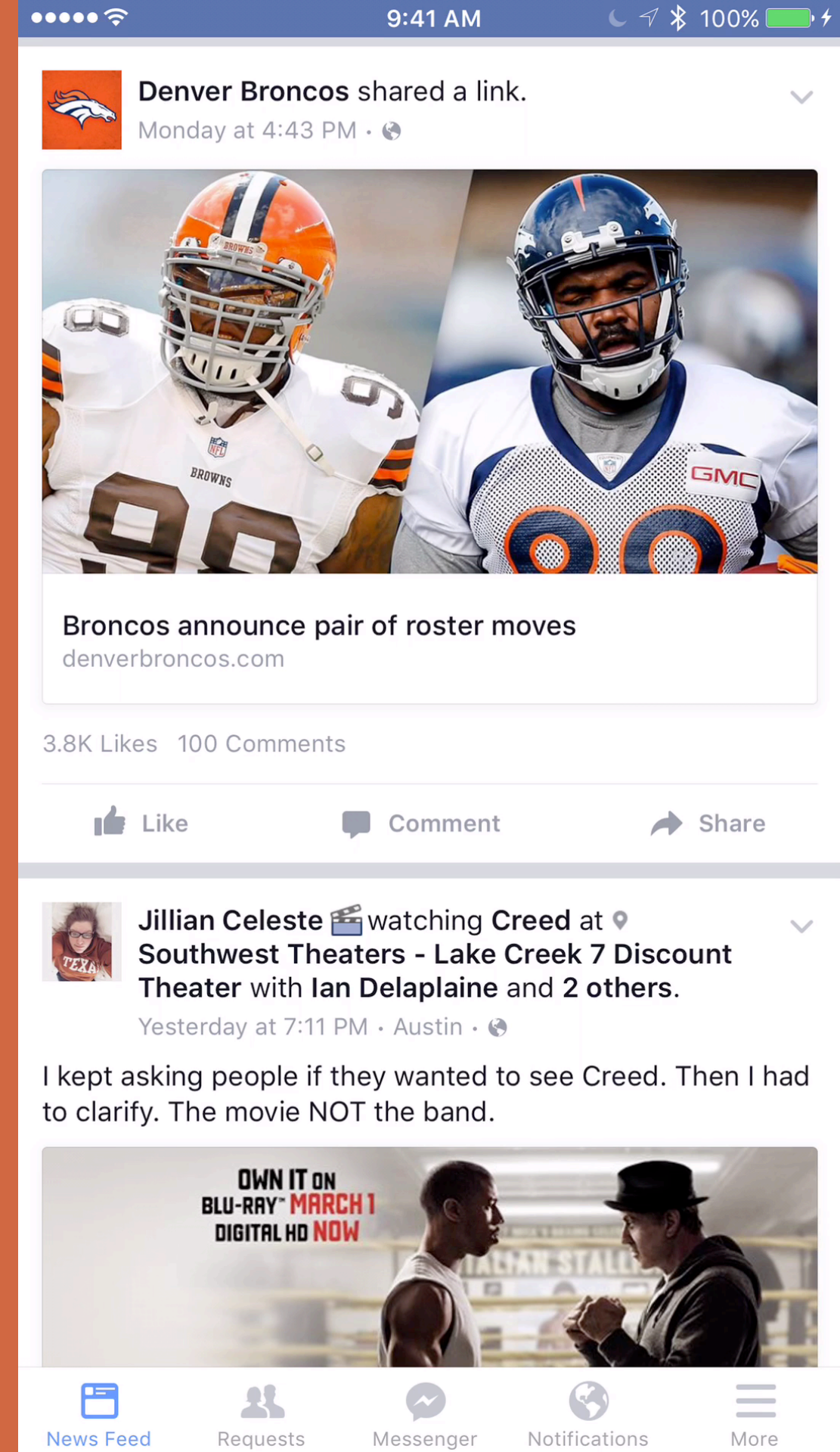
Performance

Interfaces should be fast and get out of the users way.

Facebook

Immensely complex app

Polished UI



Tools

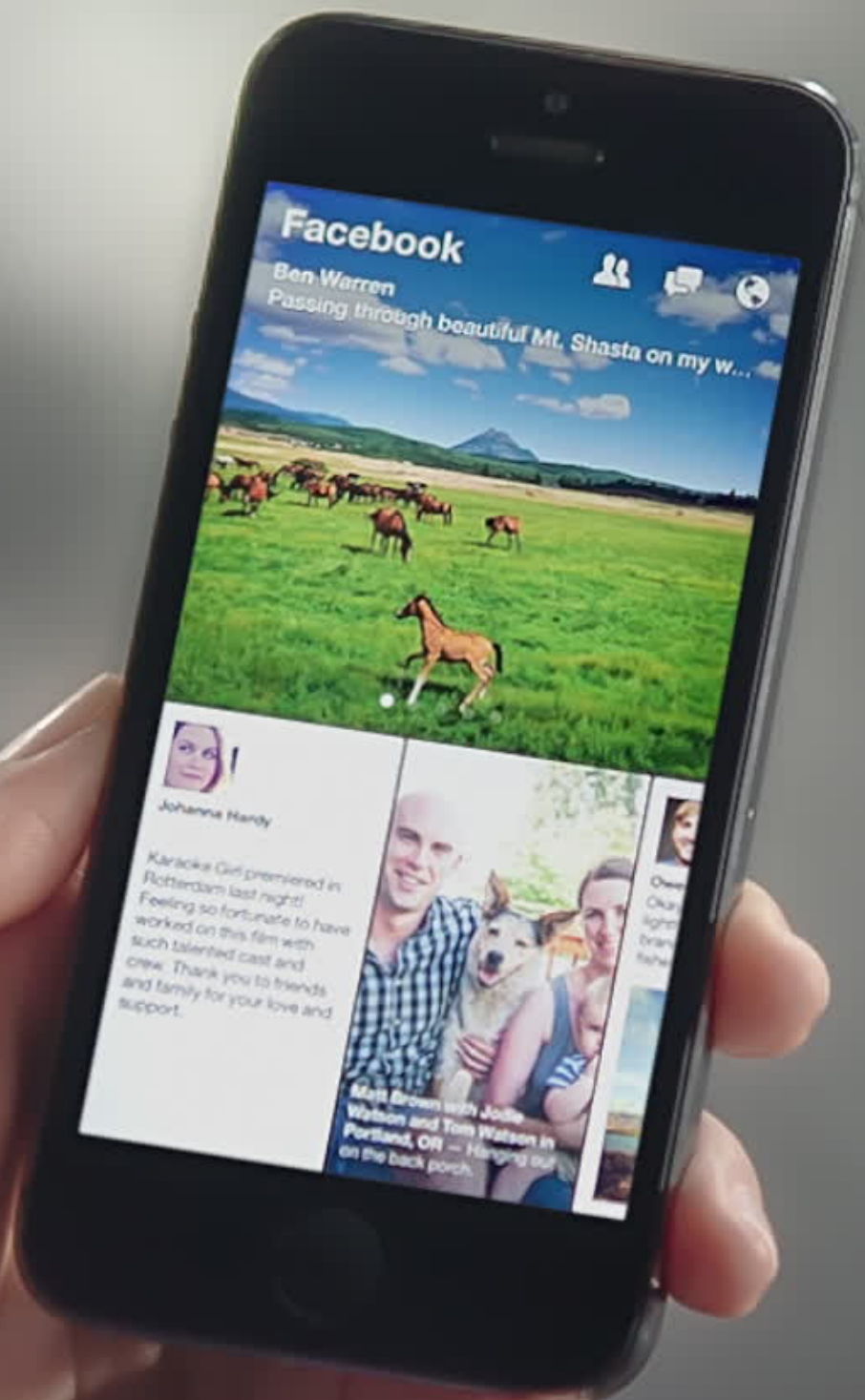
React

ReactNative

ComponentKit

Paper

**Different take on Facebook, with
an amazing interface**



Facebook

Ben Warren

Passing through beautiful Mt. Shasta on my w...



Johanna Hardy

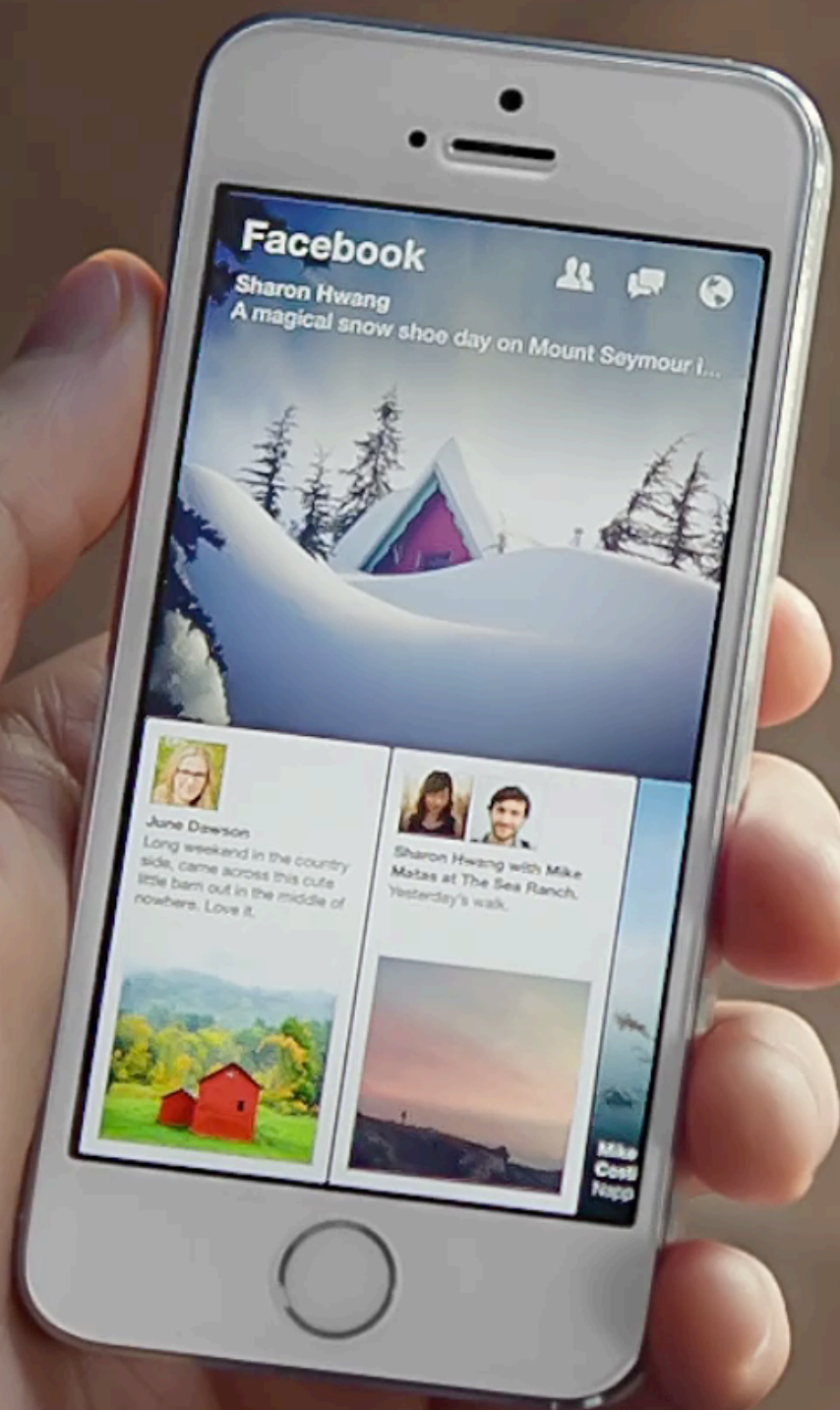
Karaoke Girl premiered in Rotterdam last night! Feeling so fortunate to have worked on this film with such talented cast and crew. Thank you to friends and family for your love and support.



Matt Brown with Jodie Watson and Tom Watson in Portland, OR — Hanging out on the back porch.



Over Our light train fish



Tools

AsyncDisplayKit

Pop

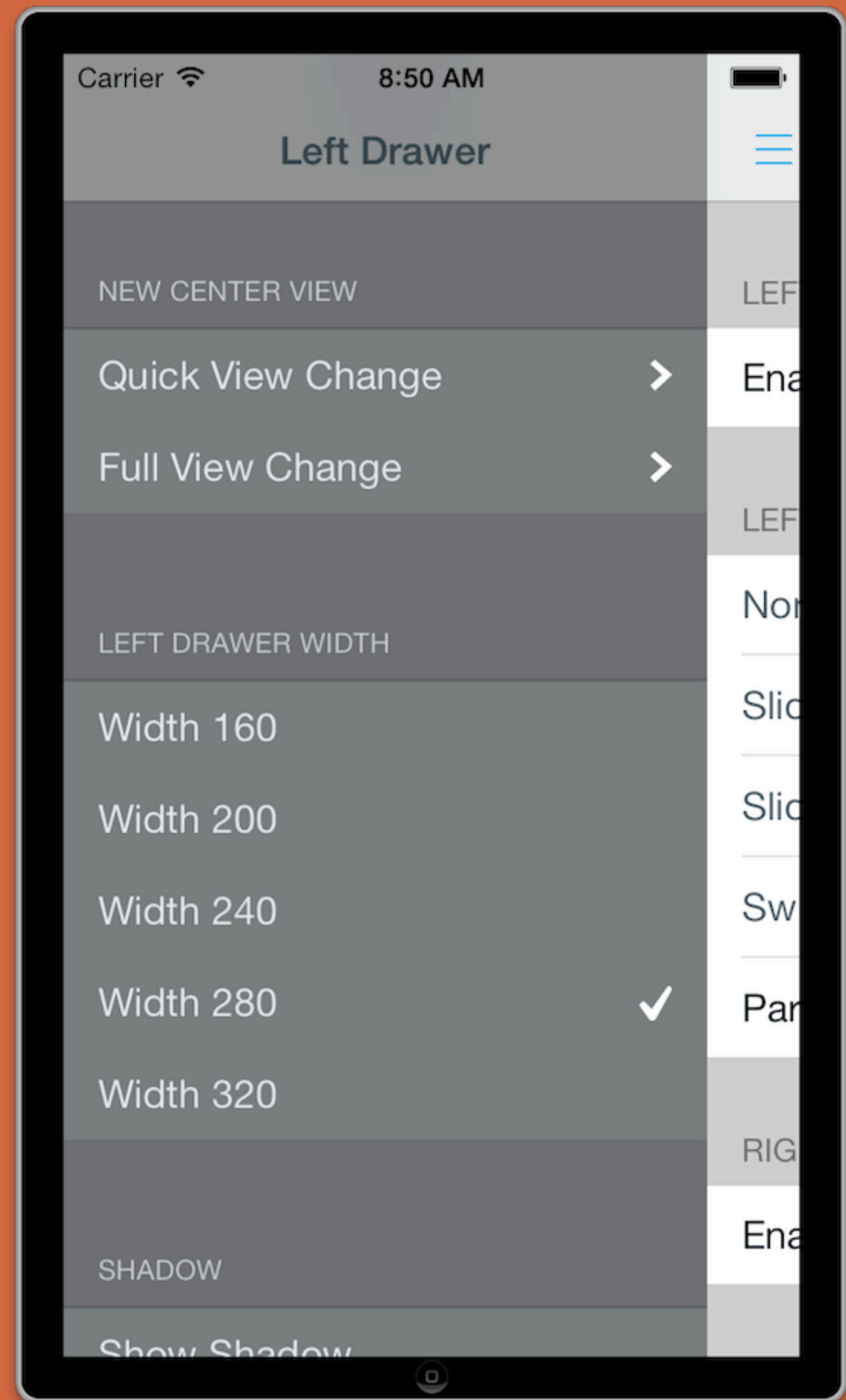
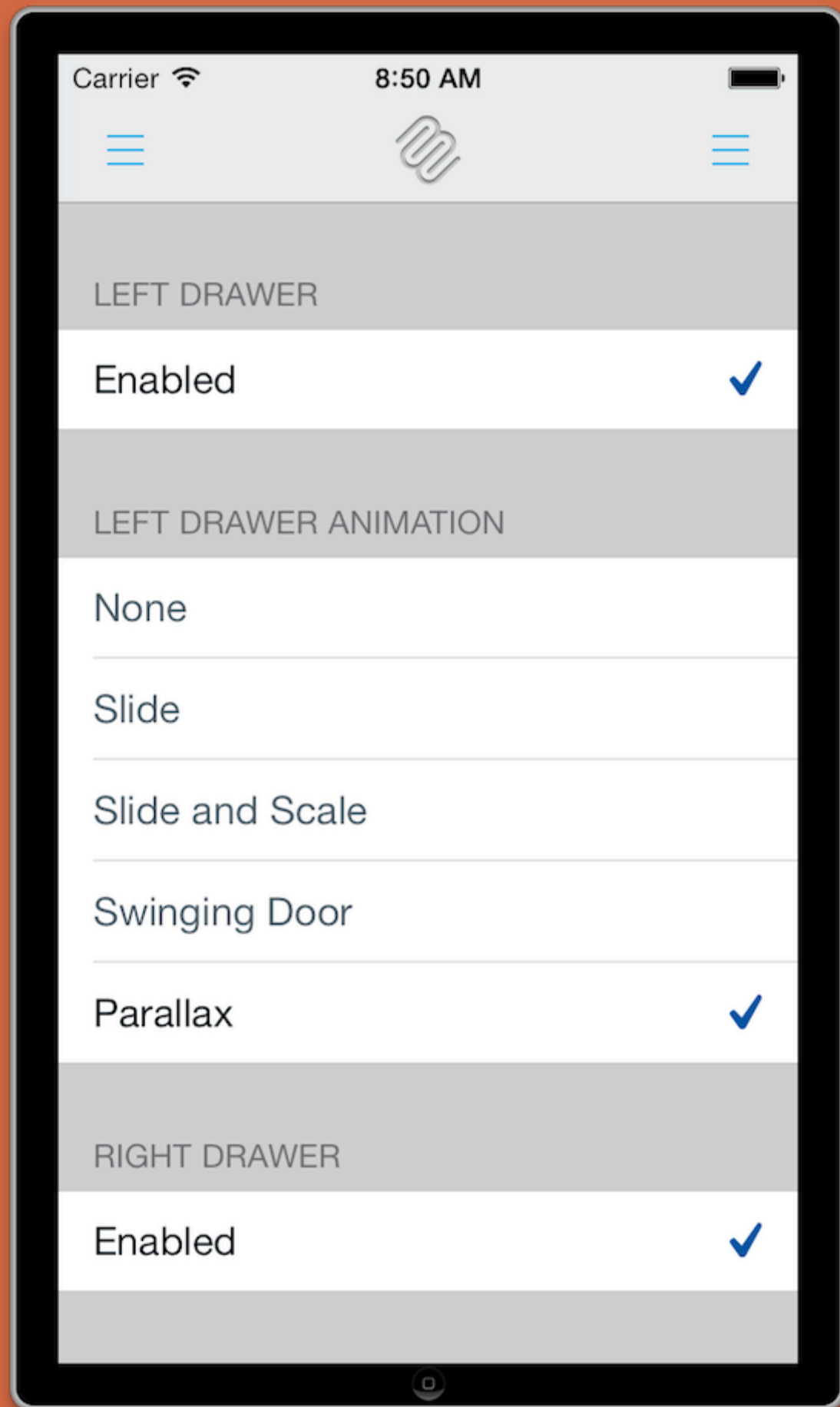
Tweaks

Consistency

Making an app feel familiar to users on the platform

MMDrawerController

You know those side menus on iOS apps?



Super common problem

Thousands of apps do this.

**Super easy to use library
to build a side menu**

Over 4000 stars on GitHub

Data Persistence

SQLite

Used for...just about everything.

Realm

Competitor to SQLite

Partially open source, with a full company around it.

Designed for Mobile!

MMRecord

**Serialization, between REST and
Core Data**

**We were solving the same
problem in all of our
projects**

**Ridiculously easy to
connect a web service to
Core Data**

**Used across many projects at
Mutual Mobile and hundreds of
other projects in the community.**

Consider *how* you use
Open Source
dependencies

Dependency Injection

Don't allow your database code to seep into your views.

Dependency Management

**Know what versions you're using
and if they're maintained.**

2. Contributing to Open Source

Are you using Open Source?

What could you do to make it better?

**Look at the libraries you
are using.**

Do they have any problems?

XMPPFramework

**Multithreading and crashing
problems**

Submitted updates to fix them

Urban Airship

**Super popular push notification
provider**

**Fixed several issues with
their SDK for iOS**

How to contribute back?

STOP!

Don't write code yet.

First Step:

Identify the Problem

Is the problem in the library, or
with your implementation of it?

Second Step:

Ask for Feedback

(Before you write code!)

Third Step:

**Make your contribution
easy to accept**



ABOUT SWIFT

BLOG

DOWNLOAD

GETTING STARTED

DOCUMENTATION

SOURCE CODE

COMMUNITY

CONTRIBUTING

Answering Questions

Reporting Bugs

[Triaging Bugs](#)

Contributing Code

Participating in the Swift
Evolution Process

LLVM and Swift

CONTINUOUS
INTEGRATION

PROJECTS

COMPILER AND
STANDARD LIBRARY

PACKAGE MANAGER

CORE LIBRARIES

REPL AND DEBUGGER

Contributing Code

Incremental Development

The Swift project uses *small, incremental changes* as its preferred development model. Sometimes these changes are small bug fixes. Other times, these changes are small steps along the path to reaching larger stated goals. In contrast, long-term development branches can leave the community without a voice during development. Some additional problems with long-term branches include:

- Resolving merge conflicts can take a lot of time if branch development and mainline development occur in the same pieces of code.
- People in the community tend to ignore work on branches.
- Very large changes are difficult to code review.
- Branches are not routinely tested by the continuous integration infrastructure.

To address these problems, Swift uses an incremental development style. Small changes are preferred whenever possible. We require contributors to follow this practice when making large or otherwise invasive changes. Some tips follow:

- Large or invasive changes usually have secondary changes that must be made before the large change (for example, API cleanup or addition). Commit these changes before the major change, independently of that work.
- If possible, decompose the remaining interrelated work into unrelated sets of changes. Next, define the first increment and get consensus on the development goal of the change.
- Make each change in the set either stand alone (for example, to fix a bug) or part of a planned series of changes that work toward the development goal. Explaining these relationships to the community can be helpful.

If you are interested in making a large change and feel unsure about its overall effect, please make sure to first discuss the change and reach a consensus through the [developer mailing list](#). Then ask about the best way to go about making the change.

Submit clean code

**Match the style of the library
you're contributing to**

**Write tests if you're fixing
a bug**

**And of course, manually test the
code yourself**

Create beautiful
programmer interfaces

Think of developers as your users

Eliminate typos, write
documentation, think about your
method names

**So why should you
contribute?**

**Improve the tools that you
already use**

**Part of being a good citizen in the
developer community**

**Learn about how things
work**

**Open a library up and read it.
Figure out how it works.**

Learn something new.

Build your Portfolio

Contributing to open source is great practice.

It's good for your career!

3. Releasing Open Source

Share openly

Solve important problems

**Have you seen the same
problem twice?**

Do you have a solution for it?

Bingo!

Make it Easy to Use

The key to great open source is ease of use.

API

Make it Intuitive

Keep it Simple

Documentation

Write detailed header documentation.

Clearly explain how to use your library.

Examples

Include examples of how to use your library.

Solve an obvious problem

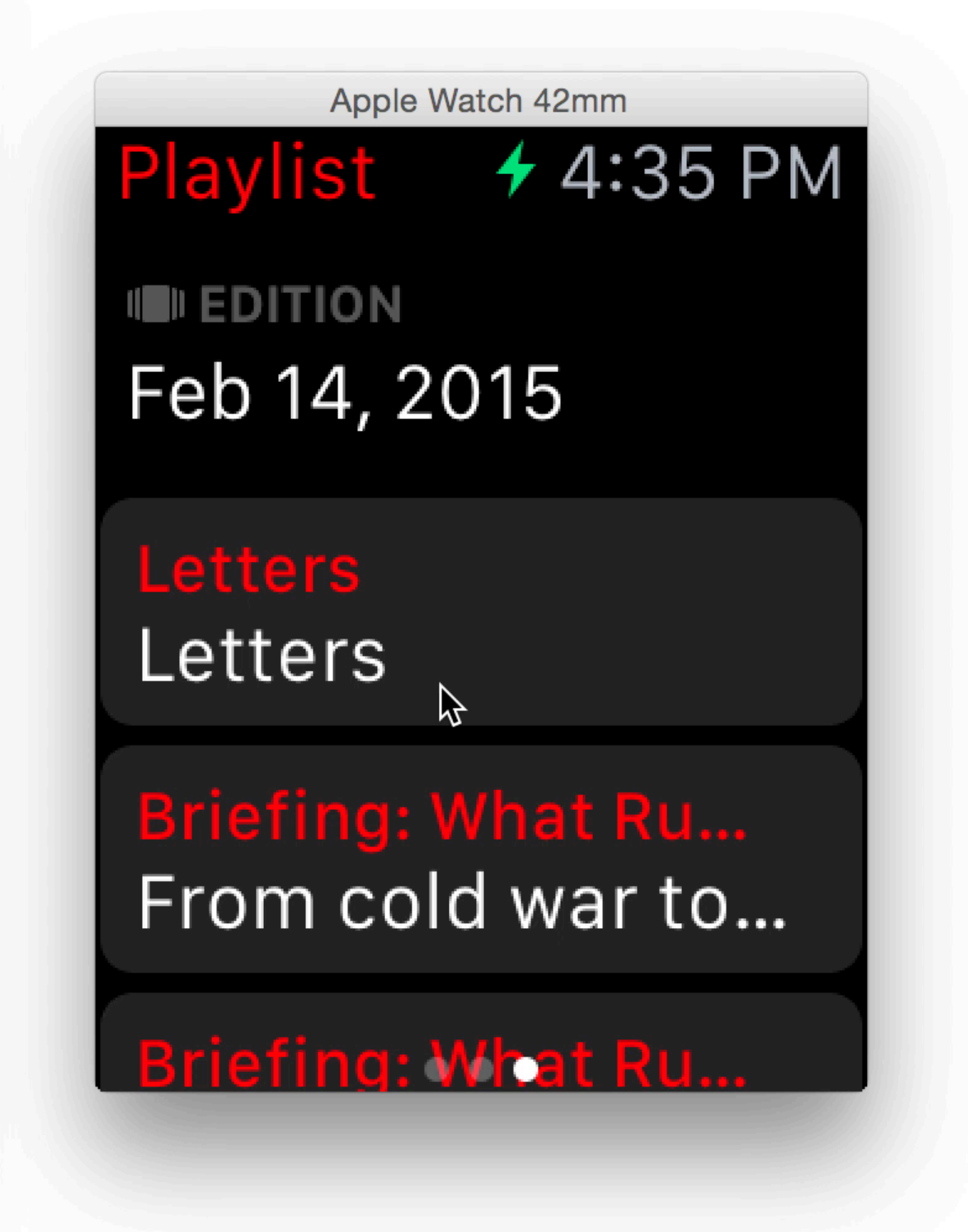
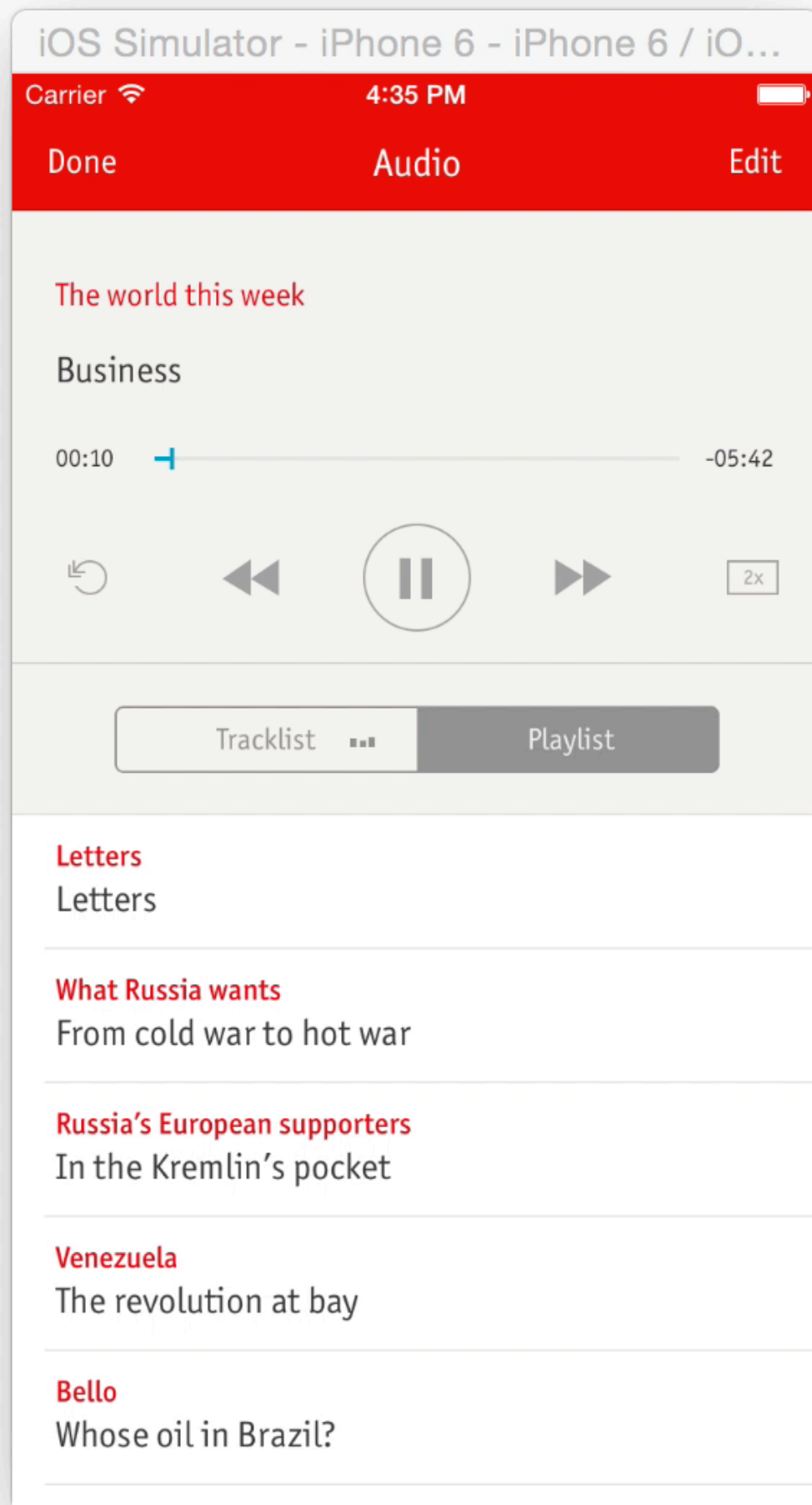
Make your library something that resonates with people.

MMWormhole



Inter Process Communication

**How do you share information
between the watch app and the
iPhone app?**



**Apple didn't provide a
solution for this**

**Left to developers to figure out
how to do it**

We Solved It!

And we made it SUPER easy to use.

Shared with everyone

Posted on developer forums

Posted on Twitter

Used in Thousands of Apps

Best way to communicate between iPhone and Watch app

Easily Extendable

Updated to support watchOS 2

Final Thoughts

Living Reference

Don't forget about your project

Challenges with Swift

Water Cooler

**Provide a place for people to
discuss common problems**

Adapt and Evolve

AFNetworking - Objective-C

Alamofire - Swift

```
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
[manager GET:@"http://example.com/resources.json" parameters:nil success:^(AFHTTPRequestOperation *operation, id responseObject) {
    NSLog(@"JSON: %@", responseObject);
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    NSLog(@"Error: %@", error);
}];
```

`Alamofire.request(.GET, "http://httpbin.org/get")`

Summary:

Use excellent tools

**Give back to the
community**

Start something new

Thanks!

conrad.stoll@mutualmobile.com

@conradstoll