Real World ReactiveCocoa

By Ole Gammelgaard Poulsen / Shape

ReactiveCocoa Basics

- Functional Reactive Programming Framework for Obj-C
- Developed by @jspahrsummers and @joshaber from Github.

APIs for composing and transforming streams of values.

Motivation

Traditional imperative programming implies a lot of state resulting in bugs and hard-to-change code.


TRICK
KHAO CHIẾN THẮNG

RACSignal

- Propagates events: `next`, `error` and, `completed`.
- After an `error` or `completed` event the signal is disposed.
- `nexts` can contain a value (an object or nil).
- Signals can be operated on or subscribed to; extensive library of operators and hooks.

Example

```
RACSignal *signal = [RACSignal return:@"hey"];
[signal subscibeNext:^(NSString *value) {
    NSLog(@"Signal sent value: %@", value);
}];
```

Composing and transforming signals is the fun part

Creating two signals

```
RACSignal *dateSignal = [RACSignal interval:1
                                onScheduler:[RACScheduler mainThreadScheduler]];

NSDate *startDate = [NSDate date];
RACSignal *secondsSignal = [dateSignal map:^id(NSDate *date) {
    return  @((int)[date timeIntervalSinceDate:startDate]);
}];

[secondsSignal subscribeNext:^(id x) {
    NSLog(@"x = %@", x);
}];

x = 1
x = 2
x = 3
x = 4
```

Merge

```
RACSignal *mergedSignal = [secondsSignal merge:signalA];

[mergedSignal subscribeNext:^(id x) {
    NSLog(@"x = %@", x);
}];
```

```
x = 2
x = 1
x = 2
x = 3
```

combineLatest:reduce:

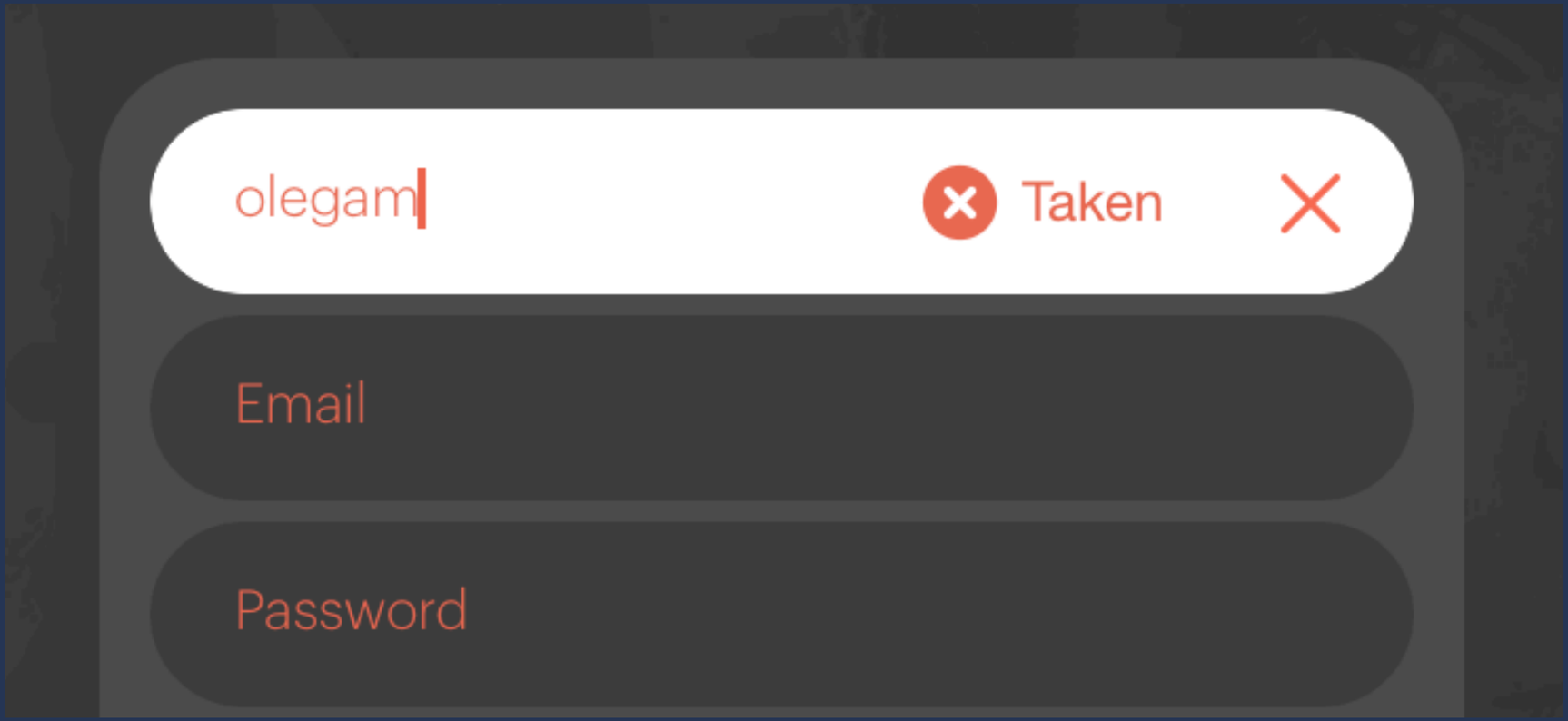
```
RACSignal *signalA = [RACSignal return:@(2)];

RACSignal *sumOfLatestSignal = [RACSignal combineLatest:@[signalA, secondsSignal]
                                                 reduce:^id(NSNumber *a, NSNumber *b) {
    return @([a integerValue] + [b integerValue]);
}];

[sumOfLatestSignal subscribeNext:^(id x) {
    NSLog(@"x = %@", x);
}];

x = 3
x = 4
x = 5
x = 6
```

Bindings

```
RACSignal *weekdaySignal = RACObserve(self.viewModel, currentWeekday);
RAC(self.label, text) = weekdaySignal;
```

- RACObserve uses KVO so properties needs to be compliant.
- For non-compliant properties use spcial RAC categories.
- RACObserve sends current value immediately. If undesired use -skip:1.

Real world example: Username availability checker



Requirements

- Should check if entered name is empty, available, taken, too short, or invalid.
- The UI should also reflect when the name is being checked and if it failed.
- It should throttle network requests when the user changes the text quickly.

```
typedef NS_ENUM(NSUInteger, CheckStatus) {
    CheckStatusEmpty = 0,
    CheckStatusChecking,
    CheckStatusAvailable,
    CheckStatusTaken,
    CheckStatusInvalid,
    CheckStatusTooShort,
    CheckStatusFailed,
};

@interface UsernameAvailabilityChecker : NSObject

- (instancetype)initWithCurrentUsername:(NSString *)currentUsername;

@property (nonatomic, copy) NSString *username;
@property (readonly) CheckStatus availabilityStatus;

@end
```

```
- (instancetype)initWithCurrentUsername:(NSString *)currentUsername {
    if (!(self = [super init])) return nil;
    RACSignal *availabilityStatusSignal = [[[RACObserve(self, username)
        throttleForInterval:0.4 afterAllowing:1]
        map:^(NSString *name) {
            if (name.length == 0) return [RACSignal return:@(CheckStatusEmpty)];
            if ([name isEqualToString:currentUsername])
                return [RACSignal return:@(CheckStatusAvailable)];
            if (name.length < 4) return [RACSignal return:@(CheckStatusTooShort)];
            if (![name validName]) return [RACSignal return:@(CheckStatusInvalid)];

            return [[[[APIClient sharedInstance] getAvailabilityForUsername:name]
                catch:^(NSError *error) {
                    return [RACSignal return:@(CheckStatusFailed)];
                }] startWith:@(CheckStatusChecking)];
        }]  switchToLatest];
    RAC(self, availabilityStatus) = availabilityStatusSignal;
    return self;
}
```

One of the challenges...

subscribeNext: is bad for you

Solution?

`@weakify(self)` / `@strongify(self)`

NO

- Avoid leaving RAC. Return signals from methods if they can fail or complete later.
- Use `RAC()` to bind to properties, but remember to catch errors.
- Use `rac_liftSelector:withSignals:`

Also

- Decompose big chunks of code by extracting variables and methods.
- Use long describing names as always.
- Create categories on `RACSignal`.

The future of ReactiveCocoa

RxSwift

Questions?