# ReactiveCocoa

## or: How I Learned to Stop Worrying and Learn FRP

# Abstraction

```
#import <Foundation/Foundation.h>

@interface Person : NSObject

@property (nonatomic, copy) NSString *name;

- (void)say:(NSString *)message withExclaim:(BOOL)exclaim;

@end

@implementation Person

- (void)say:(NSString *)message withExclaim:(BOOL)exclaim
{
    if (exclaim) {
        message = [message uppercaseString];
    }
    NSLog(@"%@", message);
}

@end
```

# Objective-C

## OOP

# Declarative Programming

# Describes how to compute

```
NSArray *array = @[@"foo", @"bar", @"baz"];
NSMutableArray *uppercaseArray = [NSMutableArray array];

for (int i = 0; i < array.count; i++ ) {
    uppercaseArray[i] = [array[i] uppercaseString];
}
// uppercaseArray: @[ @"FOO", @"BAR", @"BAZ" ]
```

```
id u = [@[@"foo", @"bar", @"baz"] map:^id(NSString *s) {
    return [s uppercaseString];
}];
// u: @[ @"FOO", @"BAR", @"BAZ" ]
```

# Describes what to compute

# Declarative Programming

# =

# Good thing

# The State of Our Apps

*Lots of state*

*Poor code locality*

*Boilerplate*

# Complexity

## INPUT

Taps
Keyboard events
GPS events
Web service responses
...


## OUTPUT

UI updates
File on disk
New DB record
Web service request
...

# Input and Output

by Josh Abernathy
http://j.mp/joshaberio

# *Is there a better abstraction?*

# Reactive Programming

"Programming around data flows and the propagation of change"

```
a = 2
b = 2
c = a + b # c is 4

b = 3
# now what's the value of c?
```

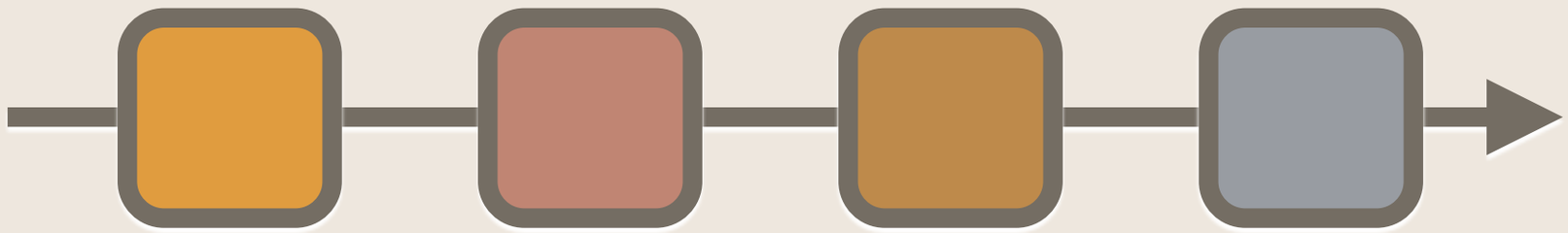
=SUM( A1 , B1 )

| | A | B | C | D |
|---|---|---|---|---|
| 1 | 2 | 2 | 4 | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | | |
| 9 | | | | |
| 10 | | | | |

# Data types that represent a value over time

# Streams

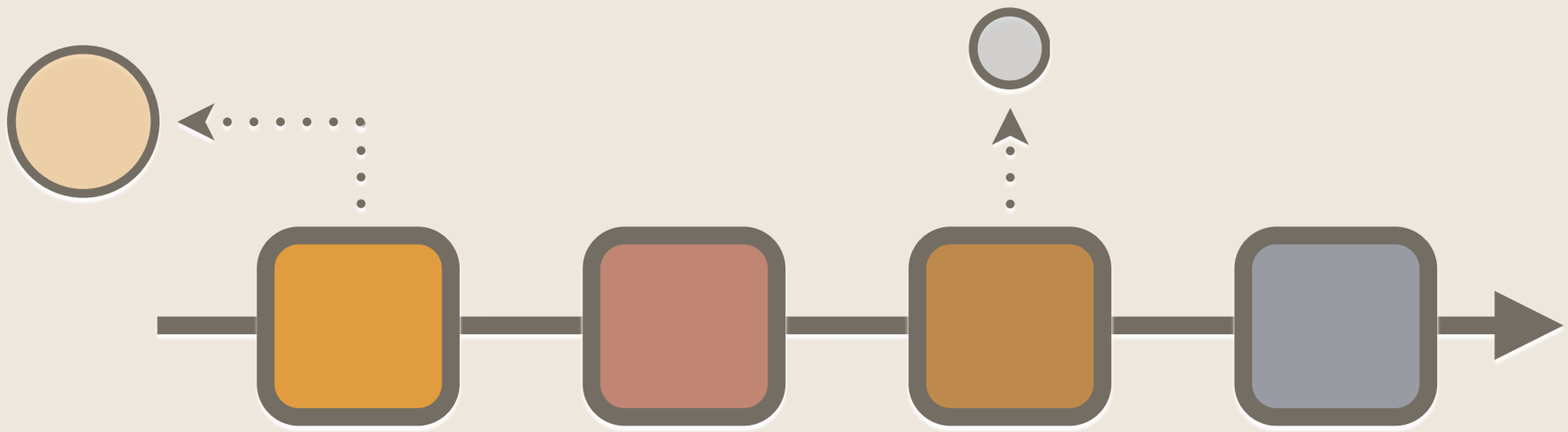
sequences of values over time



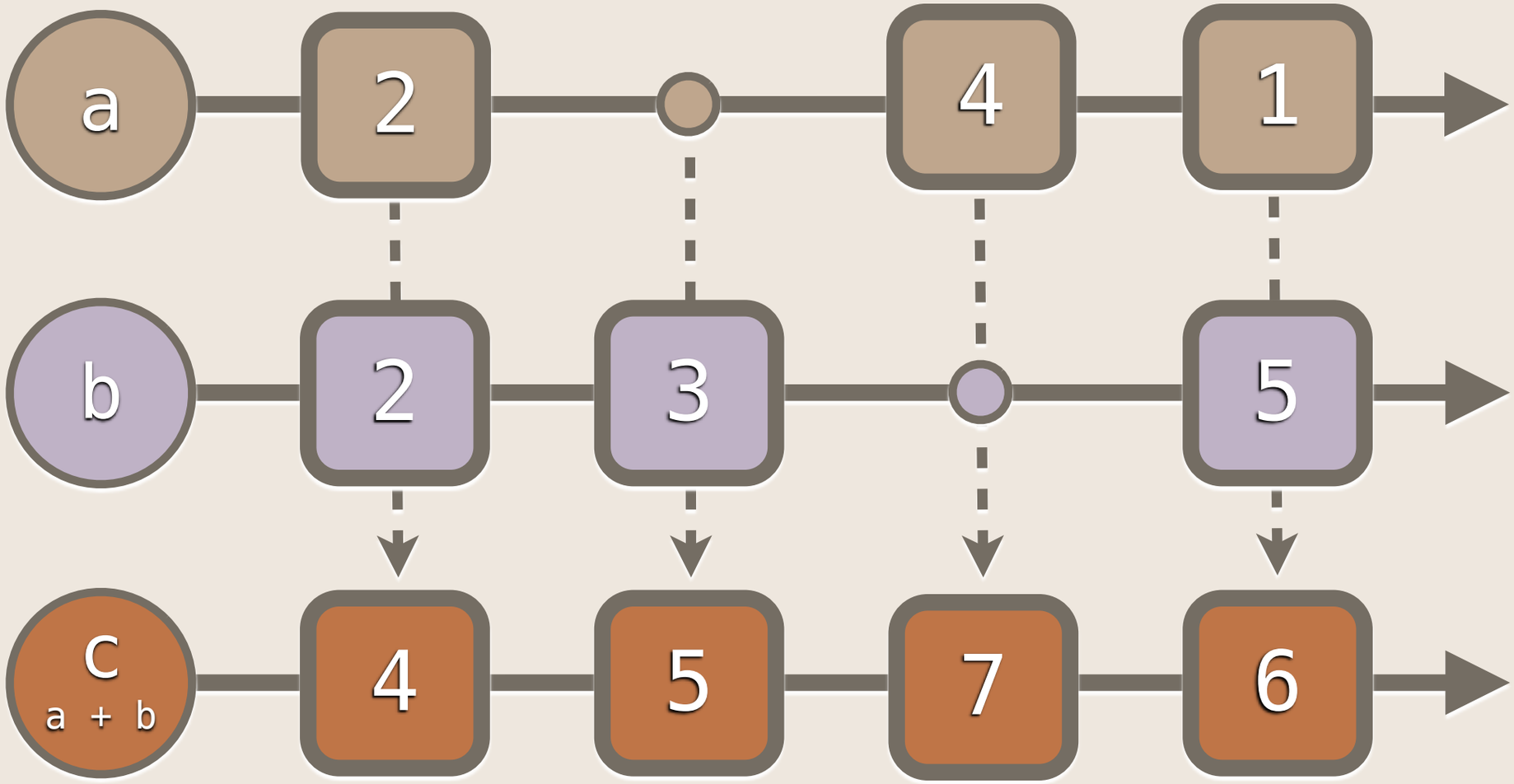
# Signals

observable streams

affords reaction to past, present, and future values



```
a := 2          # signal
b := 2          # signal
c := a + b # signal binding
           #   other signals
```



# ReactiveCocoa

"A [Cocoa] framework for composing and transforming streams of values"

# KVO property
# UI events
# Network request
# Async work
# ...
# Values over time

https://speakerdeck.com/joshaber/reactivecocoa-for-a-better-world

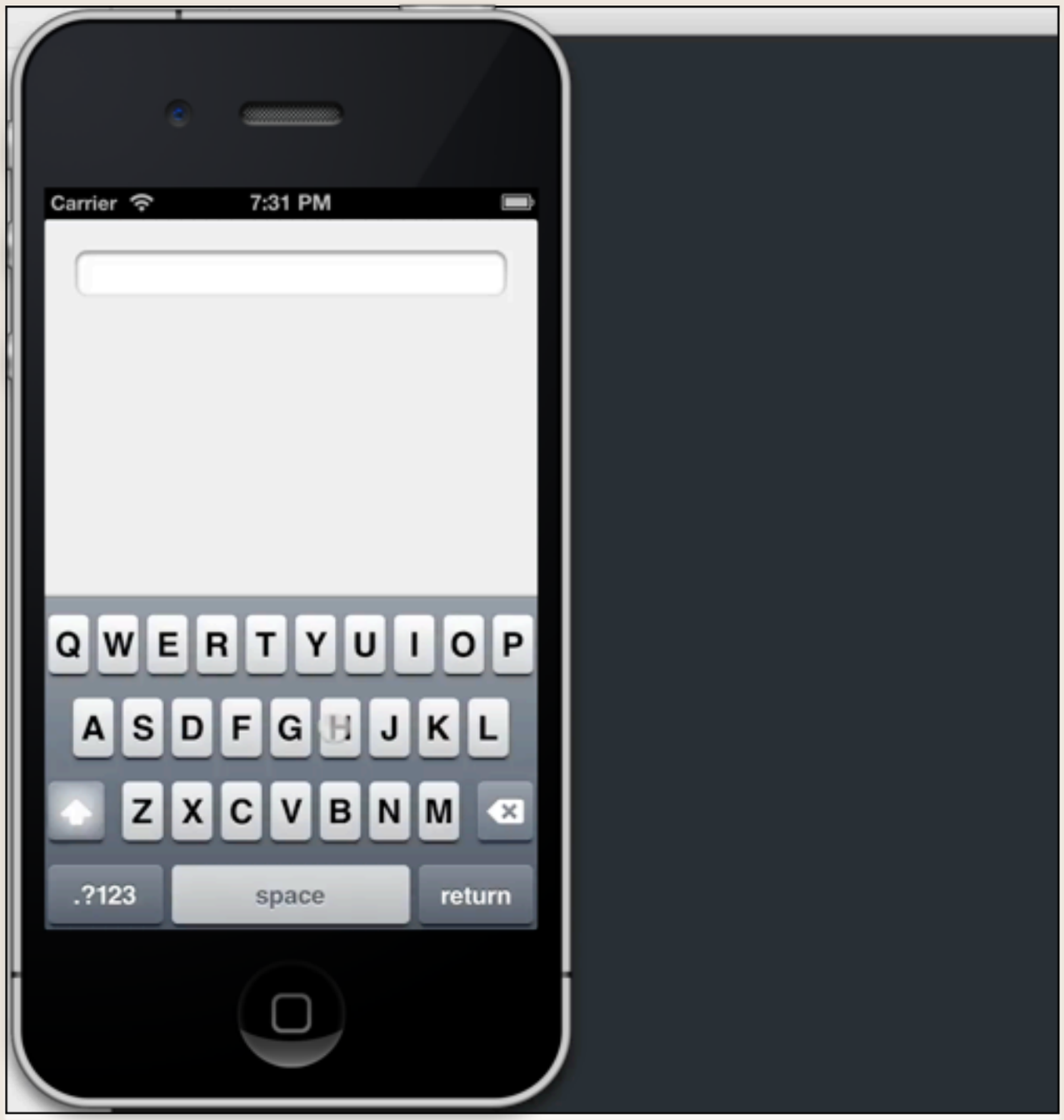
```
@interface RACSignal : RACStream
```

| next | completed | error |
| --- | --- | --- |
| sending a single value | done sending all values & successful | sending values but terminated in error |

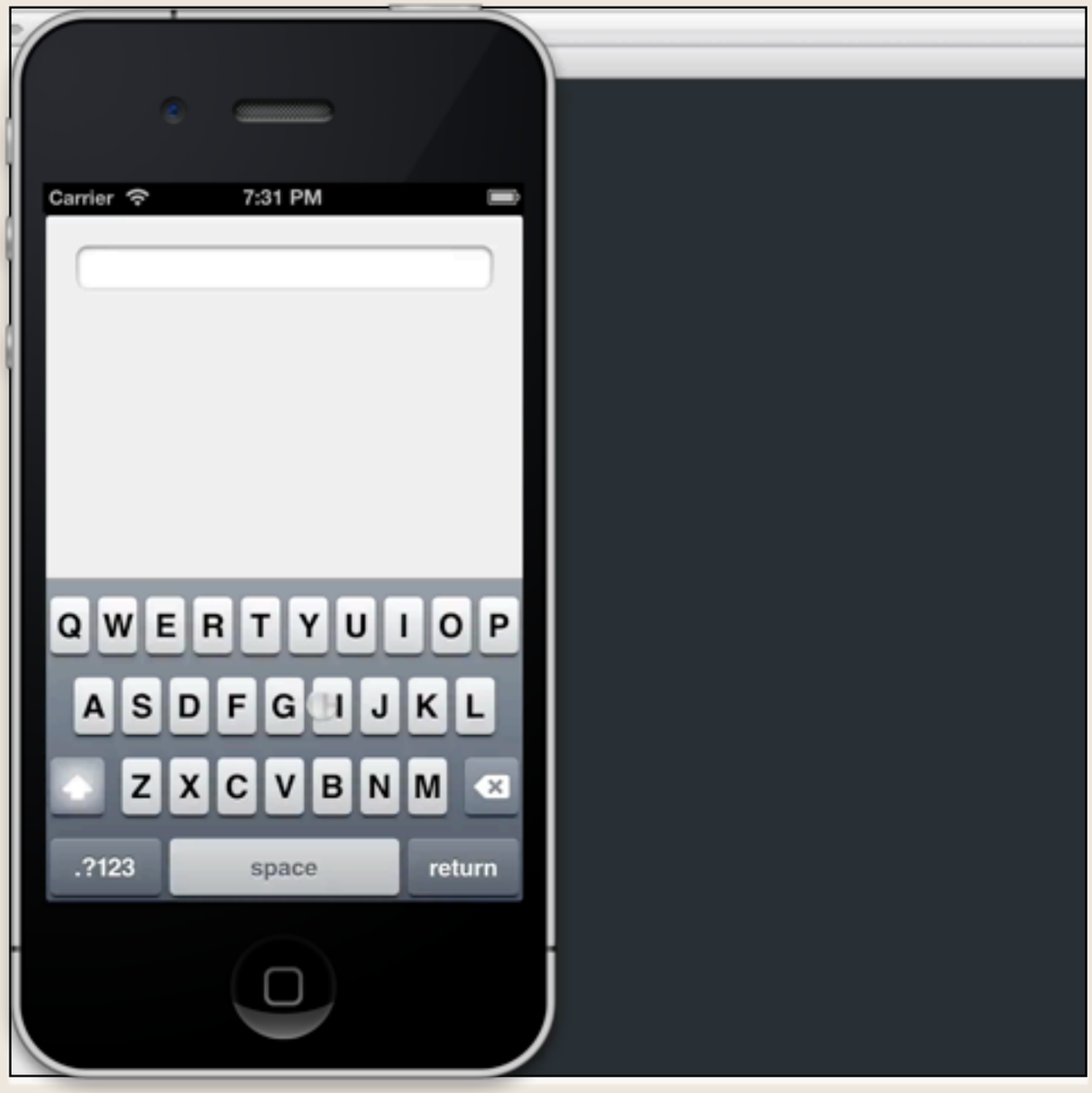
```
[RACAble(self.username) subscribeNext:^(NSString *newName) {
    NSLog(@"%@", newName);
}];
```

https://github.com/ReactiveCocoa/ReactiveCocoa/blob/master/README.md

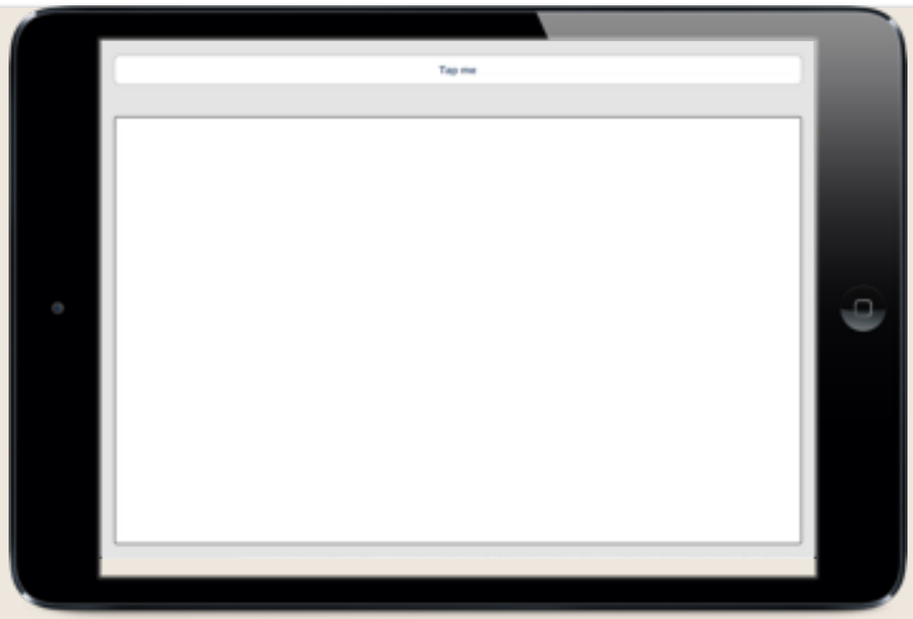
```objc
[self.textField.rac_textSignal subscribeNext:^(NSString *text) {
    NSLog(@"Incoming text signal: %@", text);
}];
```



```
[[self.textField.rac_textSignal
    map:^id(id text) {
        return [text stringByAppendingString:@" — map'd it"];
    }]
    subscribeNext:^(NSString *text) {
        NSLog(@"Incoming text signal: %@", text);
    }];
```



```objc
RACSignal *url = [[[button rac_signalForControlEvents:UIControlEventTouchUpInside]
    map:^id(id _) {
        return NSString *s = [@[
            [NSURL URLWithString:@"http://nutshell.com"],
            [NSURL URLWithString:@"http://google.com"],
            [NSURL URLWithString:@"http://wikipedia.org"],
            [NSURL URLWithString:@"http://apple.com"],
            [NSURL URLWithString:@"http://github.com/ReactiveCocoa/ReactiveCocoa"],
        ] aps_randomObject];
    }]
    publish]
    autoconnect];

RACSignal *html = [[url flattenMap:^RACStream *(NSURL *url) {
    NSStringEncoding encoding = NSUTF8StringEncoding;
    return [NSString rac_readContentsOfURL:url
                              usedEncoding:&encoding
                                 scheduler:[RACScheduler scheduler]];
    }]
    deliverOn:RACScheduler.mainThreadScheduler];

RAC(label, text) = [url map:^id(NSURL *u) { return u.absoluteString; }];
[webView rac_liftSelector:@selector(loadHTMLString:baseURL:)
              withObjects:html, url];
```



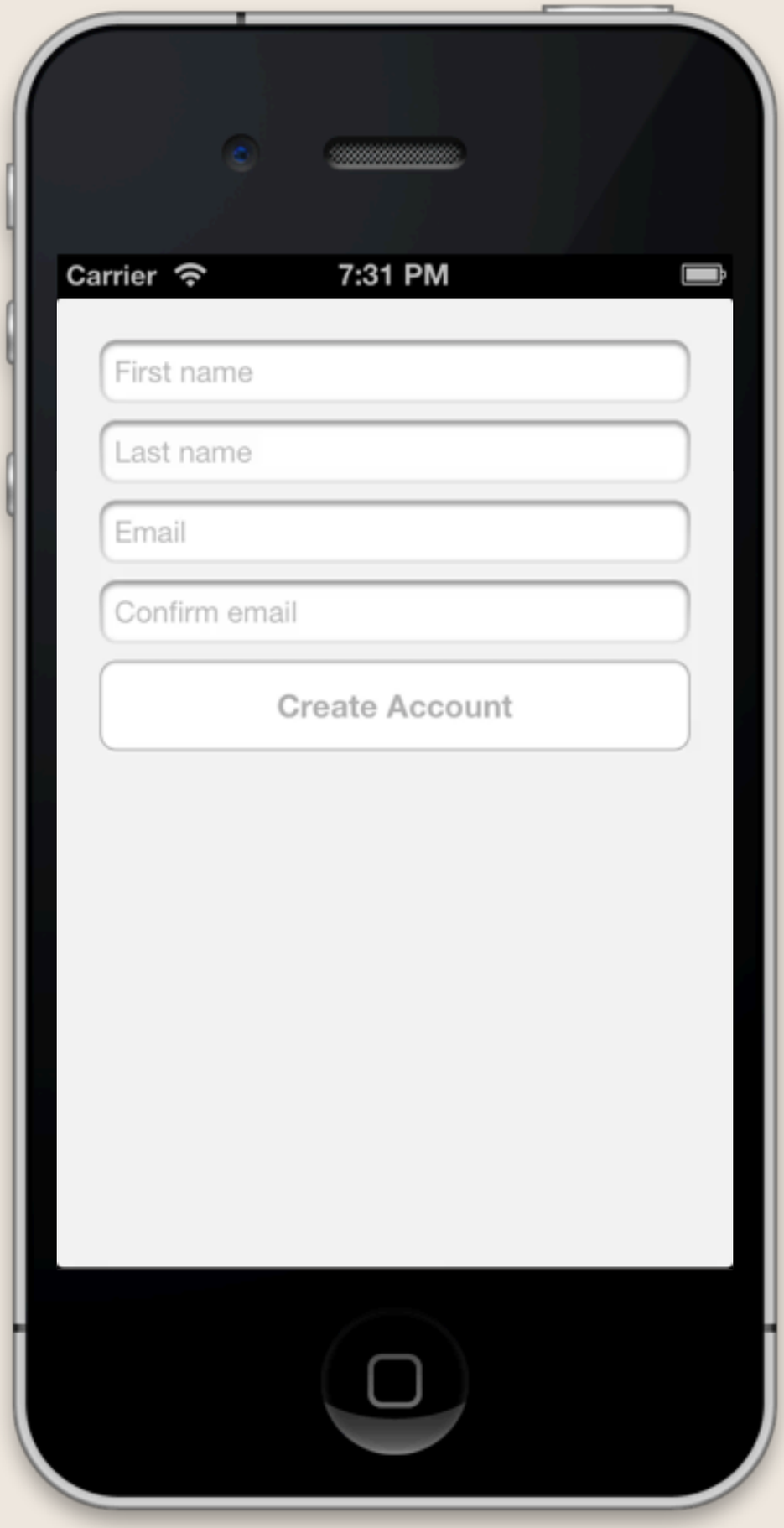
# … less imperative

```
field.enabled = NO;

[worker doWorkWithCompletionBlock:^(id result, NSError *error) {
    field.enabled = YES;
    if (error) { /* handle error */ }
    else /* handle result */
}];
```

# … more declarative

```
RACCommand *doWork = [worker doWorkAsCommand];
RAC(field, enabled) = [RACAble(doWork, executing) not];

// later, when we want to do the work
// our UI pipes are plugged together so we don't need to
// worry about that
[doWork execute:sender];

// handle result of doWork somewhere else where it matters
```

# Demo


Carrier
7:31 PM
First name
Last name
Email
Confirm email
Create Account

```objc
#import "SignUpViewController.h"
#import "APIClient.h"

@interface SignUpViewController ()
@property (weak, nonatomic) IBOutlet UITextField *firstNameField;
@property (weak, nonatomic) IBOutlet UITextField *lastNameField;
@property (weak, nonatomic) IBOutlet UITextField *emailField;
@property (weak, nonatomic) IBOutlet UITextField *confirmEmailField;
@property (weak, nonatomic) IBOutlet UIButton *createButton;
@property (weak, nonatomic) IBOutlet UILabel *statusLabel;
@property (weak, nonatomic) IBOutlet UIActivityIndicatorView *loadingIndicator;

@property (strong, nonatomic) UIColor *enabledButtonColor;
@property (strong, nonatomic) UIColor *disabledButtonColor;

@property (strong, nonatomic) UIColor *defaultTextColor;
@property (strong, nonatomic) UIColor *loadingTextColor;
@end

@implementation SignUpViewController

- (id)init
{
    self = [super initWithNibName:NSStringFromClass(self.class) bundle:nil];
    return self;
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    return [self init];
}

- (void)viewDidLoad
```

```objc
#import "SignUpViewController.h"
#import "APIClient.h"

@interface SignUpViewController ()
@property (weak, nonatomic) IBOutlet UITextField *firstNameField;
@property (weak, nonatomic) IBOutlet UITextField *lastNameField;
@property (weak, nonatomic) IBOutlet UITextField *emailField;
@property (weak, nonatomic) IBOutlet UITextField *confirmEmailField;
@property (weak, nonatomic) IBOutlet UIButton *createButton;
@property (weak, nonatomic) IBOutlet UILabel *statusLabel;
@property (weak, nonatomic) IBOutlet UIActivityIndicatorView *loadingIndicator;

@property (strong, nonatomic) UIColor *enabledButtonColor;
@property (strong, nonatomic) UIColor *disabledButtonColor;

@property (strong, nonatomic) UIColor *defaultTextColor;
@property (strong, nonatomic) UIColor *loadingTextColor;
@end

@implementation SignUpViewController

- (id)init
{
    self = [super initWithNibName:NSStringFromClass(self.class) bundle:nil];
    return self;
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    return [self init];
}

- (void)viewDidLoad
```

```objc
#import "SignUpViewController.h"
#import "APIClient.h"

@interface SignUpViewController ()
@property (weak, nonatomic) IBOutlet UITextField *firstNameField;
@property (weak, nonatomic) IBOutlet UITextField *lastNameField;
@property (weak, nonatomic) IBOutlet UITextField *emailField;
@property (weak, nonatomic) IBOutlet UITextField *confirmEmailField;
@property (weak, nonatomic) IBOutlet UIButton *createButton;
@property (weak, nonatomic) IBOutlet UILabel *statusLabel;
@property (weak, nonatomic) IBOutlet UIActivityIndicatorView *loadingIndicator;

@property (strong, nonatomic) UIColor *enabledButtonColor;
@property (strong, nonatomic) UIColor *disabledButtonColor;

@property (strong, nonatomic) UIColor *defaultTextColor;
@property (strong, nonatomic) UIColor *loadingTextColor;
@end

@implementation SignUpViewController

- (id)init
{
    self = [super initWithNibName:NSStringFromClass(self.class) bundle:nil];
    return self;
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    return [self init];
}

- (void)viewDidLoad
```

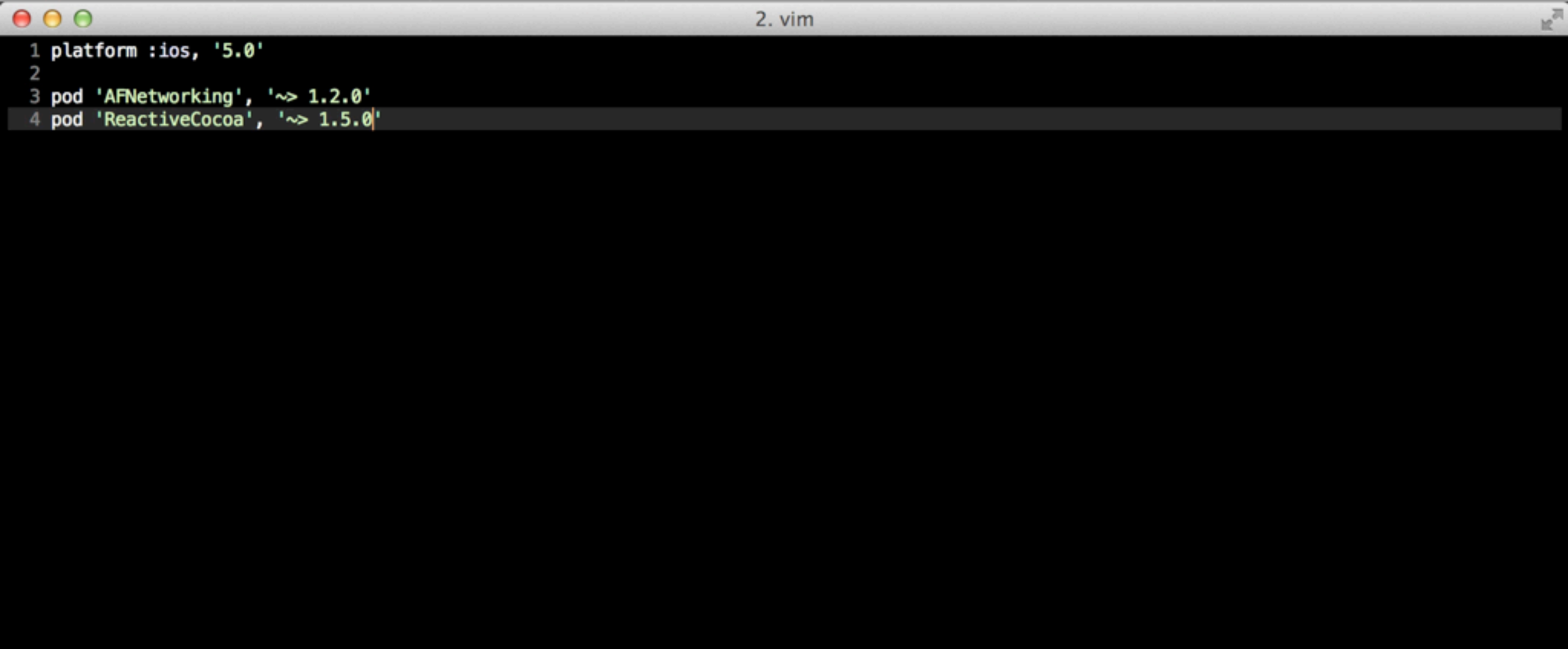
```objc
- (void)viewDidLoad
{
    [super viewDidLoad];

    // initial button setup
    self.enabledButtonColor = self.createButton.titleLabel.textColor;
    self.disabledButtonColor = UIColor.lightGrayColor;
    [self.createButton setTitleColor:self.disabledButtonColor
                            forState:UIControlStateNormal];
    self.createButton.titleLabel.textColor = UIColor.lightGrayColor;
    self.createButton.enabled = NO;
    [self.createButton addTarget:self
                          action:@selector(createButtonTouched:)
                forControlEvents:UIControlEventTouchUpInside];

    // initial text field setup
    self.defaultTextColor = self.firstNameField.textColor;
    self.loadingTextColor = UIColor.lightGrayColor;

    // listen on text editing changes
    [self.firstNameField addTarget:self
                            action:@selector(textFieldChanged:)
                  forControlEvents:UIControlEventEditingChanged];
    [self.lastNameField addTarget:self
                           action:@selector(textFieldChanged:)
                 forControlEvents:UIControlEventEditingChanged];
    [self.emailField addTarget:self
                        action:@selector(textFieldChanged:)
              forControlEvents:UIControlEventEditingChanged];
    [self.confirmEmailField addTarget:self
                               action:@selector(textFieldChanged:)
                     forControlEvents:UIControlEventEditingChanged];
}
```

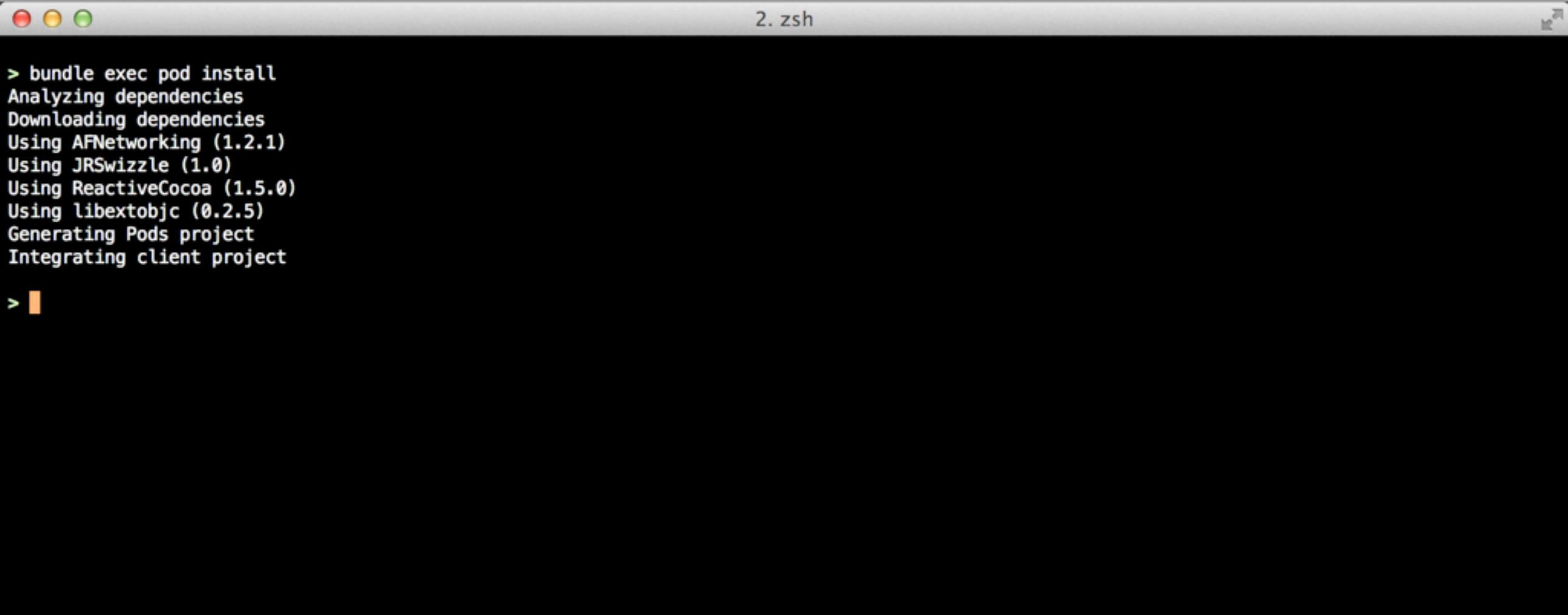
```objc
- (void)textFieldChanged:(UITextField *)textField
{
    self.createButton.enabled = self.isFormValid;
    [self.createButton setTitleColor:(self.createButton.isEnabled
                                       ? self.enabledButtonColor
                                       : self.disabledButtonColor)
                            forState:UIControlStateNormal];
}

- (void)createButtonTouched:(UIButton *)createButton
{
    [self updateUIForNetworkActivity:YES];

    self.statusLabel.textColor = UIColor.lightGrayColor;

    [APIClient.sharedClient createAccountForEmail:self.emailField.text
                                        firstName:self.firstNameField.text
                                         lastName:self.lastNameField.text
                                          success:^(id account) {
                                              self.statusLabel.textColor = [UIColor colorWi
                                              self.statusLabel.text = NSLocalizedString(@"T

                                              [self updateUIForNetworkActivity:NO];
                                          }
                                          failure:^(NSError *error) {
                                              self.statusLabel.textColor = [UIColor colorWi
                                              self.statusLabel.text = error.localizedFailur

                                              [self updateUIForNetworkActivity:NO];
                                          }];
}

#pragma mark Validation
```

```objc
- (BOOL)isFormValid
{
    NSString *firstName = self.firstNameField.text;
    NSString *lastName = self.lastNameField.text;
    NSString *email = self.emailField.text;
    NSString *confirmEmail = self.confirmEmailField.text;

    return firstName.length > 0
           && lastName.length > 0
           && email.length > 0
           && confirmEmail.length > 0
           && [email isEqualToString:confirmEmail];
}

#pragma mark UI Updates

- (void)updateUIForNetworkActivity:(BOOL)isNetworkActivity
{
    self.createButton.enabled = !isNetworkActivity;
    [self.createButton setTitleColor:(isNetworkActivity
                                      ? self.disabledButtonColor
                                      : self.enabledButtonColor)
                            forState:UIControlStateNormal];

    self.firstNameField.enabled
        = self.lastNameField.enabled
        = self.emailField.enabled
        = self.confirmEmailField.enabled = !isNetworkActivity;

    self.firstNameField.textColor
        = self.lastNameField.textColor
        = self.emailField.textColor
        = self.confirmEmailField.textColor
```

```objc
- (void)createButtonTouched:(UIButton *)createButton
{
    [self updateUIForNetworkActivity:YES];

    self.statusLabel.textColor = UIColor.lightGrayColor;

    [APIClient.sharedClient createAccountForEmail:self.emailField.text
                                        firstName:self.firstNameField.text
                                         lastName:self.lastNameField.text
                                          success:^(id account) {
                                              self.statusLabel.textColor = [UIColor colorWi
                                              self.statusLabel.text = NSLocalizedString(@"T

                                              [self updateUIForNetworkActivity:NO];
                                          }
                                          failure:^(NSError *error) {
                                              self.statusLabel.textColor = [UIColor colorWi
                                              self.statusLabel.text = error.localizedFailur

                                              [self updateUIForNetworkActivity:NO];
                                          }];
}

#pragma mark Validation

- (BOOL)isFormValid
{
    NSString *firstName = self.firstNameField.text;
    NSString *lastName = self.lastNameField.text;
    NSString *email = self.emailField.text;
    NSString *confirmEmail = self.confirmEmailField.text;

    return firstName.length > 0
```

```objc
- (void)updateUIForNetworkActivity:(BOOL)isNetworkActivity
{
    self.createButton.enabled = !isNetworkActivity;
    [self.createButton setTitleColor:(isNetworkActivity
                                       ? self.disabledButtonColor
                                       : self.enabledButtonColor)
                            forState:UIControlStateNormal];

    self.firstNameField.enabled
        = self.lastNameField.enabled
        = self.emailField.enabled
        = self.confirmEmailField.enabled = !isNetworkActivity;

    self.firstNameField.textColor
        = self.lastNameField.textColor
        = self.emailField.textColor
        = self.confirmEmailField.textColor
        = (isNetworkActivity ? self.loadingTextColor : self.defaultTextColor);

    isNetworkActivity ? [self.loadingIndicator startAnimating]
                      : [self.loadingIndicator stopAnimating];
}

@end
```


2. vim
1 platform :ios, '5.0'
2
3 pod 'AFNetworking', '~> 1.2.0'
4 pod 'ReactiveCocoa', '~> 1.5.0'
"Podfile" 3L, 53C


2. zsh
> bundle exec pod install
Analyzing dependencies
Downloading dependencies
Using AFNetworking (1.2.1)
Using JRSwizzle (1.0)
Using ReactiveCocoa (1.5.0)
Using libextobjc (0.2.5)
Generating Pods project
Integrating client project
>

```objc
#import "SignUpViewController.h"
#import "APIClient.h"

@interface SignUpViewController ()
@property (weak, nonatomic) IBOutlet UITextField *firstNameField;
@property (weak, nonatomic) IBOutlet UITextField *lastNameField;
@property (weak, nonatomic) IBOutlet UITextField *emailField;
@property (weak, nonatomic) IBOutlet UITextField *confirmEmailField;
@property (weak, nonatomic) IBOutlet UIButton *createButton;
@property (weak, nonatomic) IBOutlet UILabel *statusLabel;
@property (weak, nonatomic) IBOutlet UIActivityIndicatorView *loadingIndicator;

@property (strong, nonatomic) UIColor *enabledButtonColor;
@property (strong, nonatomic) UIColor *disabledButtonColor;

@property (strong, nonatomic) UIColor *defaultTextColor;
@property (strong, nonatomic) UIColor *loadingTextColor;
@end

@implementation SignUpViewController

- (id)init
{
    self = [super initWithNibName:NSStringFromClass(self.class) bundle:nil];
    return self;
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    return [self init];
}

- (void)viewDidLoad
```

```objc
#import "SignUpViewController.h"
#import "APIClient.h"
#import <ReactiveCocoa/ReactiveCocoa.h>

@interface SignUpViewController ()
@property (weak, nonatomic) IBOutlet UITextField *firstNameField;
@property (weak, nonatomic) IBOutlet UITextField *lastNameField;
@property (weak, nonatomic) IBOutlet UITextField *emailField;
@property (weak, nonatomic) IBOutlet UITextField *confirmEmailField;
@property (weak, nonatomic) IBOutlet UIButton *createButton;
@property (weak, nonatomic) IBOutlet UILabel *statusLabel;
@property (weak, nonatomic) IBOutlet UIActivityIndicatorView *loadingIndicator;

@property (strong, nonatomic) UIColor *enabledButtonColor;
@property (strong, nonatomic) UIColor *disabledButtonColor;

@property (strong, nonatomic) UIColor *defaultTextColor;
@property (strong, nonatomic) UIColor *loadingTextColor;
@end

@implementation SignUpViewController

- (id)init
{
    self = [super initWithNibName:NSStringFromClass(self.class) bundle:nil];
    return self;
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    return [self init];
}
```

```objc
- (BOOL)isFormValid
{
    NSString *firstName = self.firstNameField.text;
    NSString *lastName = self.lastNameField.text;
    NSString *email = self.emailField.text;
    NSString *confirmEmail = self.confirmEmailField.text;

    return firstName.length > 0
           && lastName.length > 0
           && email.length > 0
           && confirmEmail.length > 0
           && [email isEqualToString:confirmEmail];
}

#pragma mark UI Updates

- (void)updateUIForNetworkActivity:(BOOL)isNetworkActivity
{
    self.createButton.enabled = !isNetworkActivity;
    [self.createButton setTitleColor:(isNetworkActivity
                                      ? self.disabledButtonColor
                                      : self.enabledButtonColor)
                            forState:UIControlStateNormal];

    self.firstNameField.enabled
        = self.lastNameField.enabled
        = self.emailField.enabled
        = self.confirmEmailField.enabled = !isNetworkActivity;

    self.firstNameField.textColor
        = self.lastNameField.textColor
        = self.emailField.textColor
        = self.confirmEmailField.textColor
```

```
- (void)viewDidLoad
{
    [super viewDidLoad];



#pragma mark Validation

- (BOOL)isFormValid
{
    NSString *firstName = self.firstNameField.text;
    NSString *lastName = self.lastNameField.text;
    NSString *email = self.emailField.text;
    NSString *confirmEmail = self.confirmEmailField.text;

    return firstName.length > 0
           && lastName.length > 0
           && email.length > 0
           && confirmEmail.length > 0
           && [email isEqualToString:confirmEmail];
}


    // initial button setup
    self.enabledButtonColor = self.createButton.titleLabel.textColor;
    self.disabledButtonColor = UIColor.lightGrayColor;
    [self.createButton setTitleColor:self.disabledButtonColor
                            forState:UIControlStateNormal];
    self.createButton.titleLabel.textColor = UIColor.lightGrayColor;
    self.createButton.enabled = NO;
    [self.createButton addTarget:self
```

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    RACSignal *formValid = [RACSignal combineLatest:@[] reduce:nil];




#pragma mark Validation

- (BOOL)isFormValid
{
    NSString *firstName = self.firstNameField.text;
    NSString *lastName = self.lastNameField.text;
    NSString *email = self.emailField.text;
    NSString *confirmEmail = self.confirmEmailField.text;

    return firstName.length > 0
           && lastName.length > 0
           && email.length > 0
           && confirmEmail.length > 0
           && [email isEqualToString:confirmEmail];
}


    // initial button setup
    self.enabledButtonColor = self.createButton.titleLabel.textColor;
    self.disabledButtonColor = UIColor.lightGrayColor;
    [self.createButton setTitleColor:self.disabledButtonColor
                            forState:UIControlStateNormal];
    self.createButton.titleLabel.textColor = UIColor.lightGrayColor;
    self.createButton.enabled = NO;
```

```objc
- (void)viewDidLoad
{
    [super viewDidLoad];

    RACSignal *formValid = [RACSignal
        combineLatest:@[
            self.firstNameField.rac_textSignal,
            self.lastNameField.rac_textSignal,
            self.emailField.rac_textSignal,
            self.confirmEmailField.rac_textSignal,
        ]
        reduce:nil];




#pragma mark Validation

- (BOOL)isFormValid
{
    NSString *firstName = self.firstNameField.text;
    NSString *lastName = self.lastNameField.text;
    NSString *email = self.emailField.text;
    NSString *confirmEmail = self.confirmEmailField.text;

    return firstName.length > 0
            && lastName.length > 0
            && email.length > 0
            && confirmEmail.length > 0
            && [email isEqualToString:confirmEmail];
}
```

```objc
- (void)viewDidLoad
{
    [super viewDidLoad];

    RACSignal *formValid = [RACSignal
        combineLatest:@[
            self.firstNameField.rac_textSignal,
            self.lastNameField.rac_textSignal,
            self.emailField.rac_textSignal,
            self.confirmEmailField.rac_textSignal,
        ]
        reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confir
            return @(firstName.length > 0
                     && lastName.length > 0
                     && email.length > 0
                     && confirmEmail.length > 0
                     && [email isEqualToString:confirmEmail]);
        }];



#pragma mark Validation

- (BOOL)isFormValid
{
    NSString *firstName = self.firstNameField.text;
    NSString *lastName = self.lastNameField.text;
    NSString *email = self.emailField.text;
    NSString *confirmEmail = self.confirmEmailField.text;

    return firstName.length > 0
           && lastName.length > 0
```

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    RACSignal *formValid = [RACSignal
        combineLatest:@[
            self.firstNameField.rac_textSignal,
            self.lastNameField.rac_textSignal,
            self.emailField.rac_textSignal,
            self.confirmEmailField.rac_textSignal,
        ]
        reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confir
            return @(firstName.length > 0
                     && lastName.length > 0
                     && email.length > 0
                     && confirmEmail.length > 0
                     && [email isEqualToString:confirmEmail]);
        }];



    // initial button setup
    self.enabledButtonColor = self.createButton.titleLabel.textColor;
    self.disabledButtonColor = UIColor.lightGrayColor;
    [self.createButton setTitleColor:self.disabledButtonColor
                            forState:UIControlStateNormal];
    self.createButton.titleLabel.textColor = UIColor.lightGrayColor;
    self.createButton.enabled = NO;
    [self.createButton addTarget:self
                          action:@selector(createButtonTouched:)
                forControlEvents:UIControlEventTouchUpInside];
```

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    RACSignal *formValid = [RACSignal
        combineLatest:@[
            self.firstNameField.rac_textSignal,
            self.lastNameField.rac_textSignal,
            self.emailField.rac_textSignal,
            self.confirmEmailField.rac_textSignal,
        ]
        reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confir
            return @(firstName.length > 0
                     && lastName.length > 0
                     && email.length > 0
                     && confirmEmail.length > 0
                     && [email isEqualToString:confirmEmail]);
        }];

    RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];



    // initial button setup
    self.enabledButtonColor = self.createButton.titleLabel.textColor;
    self.disabledButtonColor = UIColor.lightGrayColor;
    [self.createButton setTitleColor:self.disabledButtonColor
                            forState:UIControlStateNormal];
    self.createButton.titleLabel.textColor = UIColor.lightGrayColor;
    self.createButton.enabled = NO;
    [self.createButton addTarget:self
                          action:@selector(createButtonTouched:)
                forControlEvents:UIControlEventTouchUpInside];
```

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    RACSignal *formValid = [RACSignal
        combineLatest:@[
            self.firstNameField.rac_textSignal,
            self.lastNameField.rac_textSignal,
            self.emailField.rac_textSignal,
            self.confirmEmailField.rac_textSignal,
        ]
        reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confir
            return @(firstName.length > 0
                     && lastName.length > 0
                     && email.length > 0
                     && confirmEmail.length > 0
                     && [email isEqualToString:confirmEmail]);
        }];

    RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
    RACSignal *networkResults = [createAccountCommand addSignalBlock:^RACSignal *(id value)
        // TODO: Create signal from APIClient's create account method
        return [RACSignal empty];
    }];



    // initial button setup
    self.enabledButtonColor = self.createButton.titleLabel.textColor;
    self.disabledButtonColor = UIColor.lightGrayColor;
    [self.createButton setTitleColor:self.disabledButtonColor
                            forState:UIControlStateNormal];
    self.createButton.titleLabel.textColor = UIColor.lightGrayColor;
```

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    RACSignal *formValid = [RACSignal
        combineLatest:@[
            self.firstNameField.rac_textSignal,
            self.lastNameField.rac_textSignal,
            self.emailField.rac_textSignal,
            self.confirmEmailField.rac_textSignal,
        ]
        reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confir
            return @(firstName.length > 0
                     && lastName.length > 0
                     && email.length > 0
                     && confirmEmail.length > 0
                     && [email isEqualToString:confirmEmail]);
        }];

    RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
    RACSignal *networkResults = [[[createAccountCommand
        addSignalBlock:^RACSignal *(id value) {
            // TODO: Create signal from APIClient's create account method
            return [RACSignal empty];
        }]
        switchToLatest]
        deliverOn:[RACScheduler mainThreadScheduler]];



    // initial button setup
    self.enabledButtonColor = self.createButton.titleLabel.textColor;
    self.disabledButtonColor = UIColor.lightGrayColor;
```

```
        return @(firstName.length > 0
                 && lastName.length > 0
                 && email.length > 0
                 && confirmEmail.length > 0
                 && [email isEqualToString:confirmEmail]);
    }];

RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
RACSignal *networkResults = [[[createAccountCommand
    addSignalBlock:^RACSignal *(id value) {
        // TODO: Create signal from APIClient's create account method
        return [RACSignal empty];
    }]
    switchToLatest]
    deliverOn:[RACScheduler mainThreadScheduler]];

    // bind create button's UI state and touch action









    // initial button setup
    self.enabledButtonColor = self.createButton.titleLabel.textColor;
    self.disabledButtonColor = UIColor.lightGrayColor;
    [self.createButton setTitleColor:self.disabledButtonColor
                            forState:UIControlStateNormal];
```

```
        return @(firstName.length > 0
                 && lastName.length > 0
                 && email.length > 0
                 && confirmEmail.length > 0
                 && [email isEqualToString:confirmEmail]);
    }];

    RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
    RACSignal *networkResults = [[[createAccountCommand
        addSignalBlock:^RACSignal *(id value) {
            // TODO: Create signal from APIClient's create account method
            return [RACSignal empty];
        }]
        switchToLatest]
        deliverOn:[RACScheduler mainThreadScheduler]];


    // bind create button's UI state and touch action
    [[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNe
        [createAccountCommand execute:sender];
    }];









    // initial button setup
    self.enabledButtonColor = self.createButton.titleLabel.textColor;
```

```
        return @(firstName.length > 0
                 && lastName.length > 0
                 && email.length > 0
                 && confirmEmail.length > 0
                 && [email isEqualToString:confirmEmail]);
    }];

    RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
    RACSignal *networkResults = [[[createAccountCommand
        addSignalBlock:^RACSignal *(id value) {
            // TODO: Create signal from APIClient's create account method
            return [RACSignal empty];
        }]
        switchToLatest]
        deliverOn:[RACScheduler mainThreadScheduler]];

    // bind create button's UI state and touch action
    [[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNe
        [createAccountCommand execute:sender];
    }];

    RACSignal *buttonEnabled = RACAbleWithStart(createAccountCommand, canExecute);
```

```
        return @(firstName.length > 0
                 && lastName.length > 0
                 && email.length > 0
                 && confirmEmail.length > 0
                 && [email isEqualToString:confirmEmail]);
    }];

    RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
    RACSignal *networkResults = [[[createAccountCommand
        addSignalBlock:^RACSignal *(id value) {
            // TODO: Create signal from APIClient's create account method
            return [RACSignal empty];
        }]
        switchToLatest]
        deliverOn:[RACScheduler mainThreadScheduler]];

    // bind create button's UI state and touch action
    [[self.createButton rac_signalForControlEventsUIControlEventTouchUpInside] subscribeNe
        [createAccountCommand execute:sender];
    }];

    RACSignal *buttonEnabled = RACAbleWithStart(createAccountCommand, canExecute);
    RAC(self.createButton.enabled) = buttonEnabled;
```

```
52
53      RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
54      RACSignal *networkResults = [[[createAccountCommand
55          addSignalBlock:^RACSignal *(id value) {
56              // TODO: Create signal from APIClient's create account method
57              return [RACSignal empty];
58          }]
59          switchToLatest]
60          deliverOn:[RACScheduler mainThreadScheduler]];
61
62      // bind create button's UI state and touch action
63      [[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNe
64          [createAccountCommand execute:sender];
65      }];
66
67      RACSignal *buttonEnabled = RACAbleWithStart(createAccountCommand, canExecute);
68      RAC(self.createButton.enabled) = buttonEnabled;
69
70      UIColor *defaultButtonTitleColor = self.createButton.titleLabel.textColor;
71
72
73
74
75
76
77
78
79
80
81
82
83
```

```
52
53      RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
54      RACSignal *networkResults = [[[createAccountCommand
55          addSignalBlock:^RACSignal *(id value) {
56              // TODO: Create signal from APIClient's create account method
57              return [RACSignal empty];
58          }]
59          switchToLatest]
60          deliverOn:[RACScheduler mainThreadScheduler]];
61
62      // bind create button's UI state and touch action
63      [[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNe
64          [createAccountCommand execute:sender];
65      }];
66
67      RACSignal *buttonEnabled = RACAbleWithStart(createAccountCommand, canExecute);
68      RAC(self.createButton.enabled) = buttonEnabled;
69
70      UIColor *defaultButtonTitleColor = self.createButton.titleLabel.textColor;
71      RACSignal *buttonTextColor = [buttonEnabled map:^id(NSNumber *x) {
72          return x.boolValue ? defaultButtonTitleColor : [UIColor lightGrayColor];
73      }];
74
75
76
77
78
79
80
81
82
83
```

```
        return [RACSignal empty];
    }]
    switchToLatest]
    deliverOn:[RACScheduler mainThreadScheduler]];

// bind create button's UI state and touch action
[[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNe
    [createAccountCommand execute:sender];
}];

RACSignal *buttonEnabled = RACAbleWithStart(createAccountCommand, canExecute);
RAC(self.createButton.enabled) = buttonEnabled;

UIColor *defaultButtonTitleColor = self.createButton.titleLabel.textColor;
RACSignal *buttonTextColor = [buttonEnabled map:^id(NSNumber *x) {
    return x.boolValue ? defaultButtonTitleColor : [UIColor lightGrayColor];
}];

[self.createButton rac_liftSelector:@selector(setTitleColor:forState:)
                        withObjects:buttonEnabled, @(UIControlStateNormal)];
```

```
    deliverOn:[RACScheduler mainThreadScheduler]];

// bind create button's UI state and touch action
[[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNe
    [createAccountCommand execute:sender];
}];

RACSignal *buttonEnabled = RACAbleWithStart(createAccountCommand, canExecute);
RAC(self.createButton.enabled) = buttonEnabled;

UIColor *defaultButtonTitleColor = self.createButton.titleLabel.textColor;
RACSignal *buttonTextColor = [buttonEnabled map:^id(NSNumber *x) {
    return x.boolValue ? defaultButtonTitleColor : [UIColor lightGrayColor];
}];

[self.createButton rac_liftSelector:@selector(setTitleColor:forState:)
                        withObjects:buttonEnabled, @(UIControlStateNormal)];

    // bind button and text field state to create account command executing state
```

```
    deliverOn:[RACScheduler mainThreadScheduler]];

// bind create button's UI state and touch action
[[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNe
    [createAccountCommand execute:sender];
}];

RACSignal *buttonEnabled = RACAbleWithStart(createAccountCommand, canExecute);
RAC(self.createButton.enabled) = buttonEnabled;

UIColor *defaultButtonTitleColor = self.createButton.titleLabel.textColor;
RACSignal *buttonTextColor = [buttonEnabled map:^id(NSNumber *x) {
    return x.boolValue ? defaultButtonTitleColor : [UIColor lightGrayColor];
}];

[self.createButton rac_liftSelector:@selector(setTitleColor:forState:)
                        withObjects:buttonEnabled, @(UIControlStateNormal)];

// bind button and text field state to create account command executing state

RACSignal *executing = RACAble(createAccountCommand, executing);
```

```
64      [createAccountCommand execute:sender];
65  }];
66
67  RACSignal *buttonEnabled = RACAbleWithStart(createAccountCommand, canExecute);
68  RAC(self.createButton.enabled) = buttonEnabled;
69
70  UIColor *defaultButtonTitleColor = self.createButton.titleLabel.textColor;
71  RACSignal *buttonTextColor = [buttonEnabled map:^id(NSNumber *x) {
72      return x.boolValue ? defaultButtonTitleColor : [UIColor lightGrayColor];
73  }];
74
75  [self.createButton rac_liftSelector:@selector(setTitleColor:forState:)
76                          withObjects:buttonEnabled, @(UIControlStateNormal)];
77
78  // bind button and text field state to create account command executing state
79
80  RACSignal *executing = RACAble(createAccountCommand, executing);
81
82  RACSignal *fieldTextColor = [executing map:^id(NSNumber *x) {
83      return x.boolValue ? [UIColor lightGrayColor] : [UIColor blackColor];
84  }];
85
86
87
88
89
90
91
92
93
94
95
```

```
68      RAC(self.createButton.enabled) = buttonEnabled;
69
70      UIColor *defaultButtonTitleColor = self.createButton.titleLabel.textColor;
71      RACSignal *buttonTextColor = [buttonEnabled map:^id(NSNumber *x) {
72          return x.boolValue ? defaultButtonTitleColor : [UIColor lightGrayColor];
73      }];
74
75      [self.createButton rac_liftSelector:@selector(setTitleColor:forState:)
76                              withObjects:buttonEnabled, @(UIControlStateNormal)];
77
78      // bind button and text field state to create account command executing state
79
80      RACSignal *executing = RACAble(createAccountCommand, executing);
81
82      RACSignal *fieldTextColor = [executing map:^id(NSNumber *x) {
83          return x.boolValue ? [UIColor lightGrayColor] : [UIColor blackColor];
84      }];
85
86      RAC(self.firstNameField.textColor) = fieldTextColor;
87      RAC(self.lastNameField.textColor) = fieldTextColor;
88      RAC(self.emailField.textColor) = fieldTextColor;
89      RAC(self.confirmEmailField.textColor) = fieldTextColor;
90
91
92
93
94
95
96
97
98
99
```

```
    }];

    [self.createButton rac_liftSelector:@selector(setTitleColor:forState:)
                            withObjects:buttonEnabled, @(UIControlStateNormal)];

    // bind button and text field state to create account command executing state

    RACSignal *executing = RACAble(createAccountCommand, executing);

    RACSignal *fieldTextColor = [executing map:^id(NSNumber *x) {
        return x.boolValue ? [UIColor lightGrayColor] : [UIColor blackColor];
    }];

    RAC(self.firstNameField.textColor) = fieldTextColor;
    RAC(self.lastNameField.textColor) = fieldTextColor;
    RAC(self.emailField.textColor) = fieldTextColor;
    RAC(self.confirmEmailField.textColor) = fieldTextColor;

    RACSignal *notExecuting = [executing not];
```

```
75      [self.createButton rac_liftSelector:@selector(setTitleColor:forState:)
76                              withObjects:buttonEnabled, @(UIControlStateNormal)];
77
78      // bind button and text field state to create account command executing state
79
80      RACSignal *executing = RACAble(createAccountCommand, executing);
81
82      RACSignal *fieldTextColor = [executing map:^id(NSNumber *x) {
83          return x.boolValue ? [UIColor lightGrayColor] : [UIColor blackColor];
84      }];
85
86      RAC(self.firstNameField.textColor) = fieldTextColor;
87      RAC(self.lastNameField.textColor) = fieldTextColor;
88      RAC(self.emailField.textColor) = fieldTextColor;
89      RAC(self.confirmEmailField.textColor) = fieldTextColor;
90
91      RACSignal *notExecuting = [executing not];
92
93      RAC(self.firstNameField.enabled) = notExecuting;
94      RAC(self.lastNameField.enabled) = notExecuting;
95      RAC(self.emailField.enabled) = notExecuting;
96      RAC(self.confirmEmailField.enabled) = notExecuting;
97
98
99
100
101
102
103
104
105
106
```

```
80      RACSignal *executing = RACAble(createAccountCommand, executing);
81
82      RACSignal *fieldTextColor = [executing map:^id(NSNumber *x) {
83          return x.boolValue ? [UIColor lightGrayColor] : [UIColor blackColor];
84      }];
85
86      RAC(self.firstNameField.textColor) = fieldTextColor;
87      RAC(self.lastNameField.textColor) = fieldTextColor;
88      RAC(self.emailField.textColor) = fieldTextColor;
89      RAC(self.confirmEmailField.textColor) = fieldTextColor;
90
91      RACSignal *notExecuting = [executing not];
92
93      RAC(self.firstNameField.enabled) = notExecuting;
94      RAC(self.lastNameField.enabled) = notExecuting;
95      RAC(self.emailField.enabled) = notExecuting;
96      RAC(self.confirmEmailField.enabled) = notExecuting;
97
98      [executing subscribeNext:^(NSNumber *x) {
99          x.boolValue ? [self.loadingIndicator startAnimating] : [self.loadingIndicator stopA
100         self.statusLabel.textColor = [UIColor lightGrayColor];
101     }];
102
103
104
105
106
107
108
109
110
111
```

```
    }];

    RAC(self.firstNameField.textColor) = fieldTextColor;
    RAC(self.lastNameField.textColor) = fieldTextColor;
    RAC(self.emailField.textColor) = fieldTextColor;
    RAC(self.confirmEmailField.textColor) = fieldTextColor;

    RACSignal *notExecuting = [executing not];

    RAC(self.firstNameField.enabled) = notExecuting;
    RAC(self.lastNameField.enabled) = notExecuting;
    RAC(self.emailField.enabled) = notExecuting;
    RAC(self.confirmEmailField.enabled) = notExecuting;

    [executing subscribeNext:^(NSNumber *x) {
        x.boolValue ? [self.loadingIndicator startAnimating] : [self.loadingIndicator stopA
        self.statusLabel.textColor = [UIColor lightGrayColor];
    }];

    // Derive the status label's text and color from our network result
```

```
87  RAC(self.lastNameField.textColor) = fieldTextColor;
88  RAC(self.emailField.textColor) = fieldTextColor;
89  RAC(self.confirmEmailField.textColor) = fieldTextColor;
90
91  RACSignal *notExecuting = [executing not];
92
93  RAC(self.firstNameField.enabled) = notExecuting;
94  RAC(self.lastNameField.enabled) = notExecuting;
95  RAC(self.emailField.enabled) = notExecuting;
96  RAC(self.confirmEmailField.enabled) = notExecuting;
97
98  [executing subscribeNext:^(NSNumber *x) {
99      x.boolValue ? [self.loadingIndicator startAnimating] : [self.loadingIndicator stopA
100     self.statusLabel.textColor = [UIColor lightGrayColor];
101 }];
102
103 // Derive the status label's text and color from our network result
104 RAC(self.statusLabel.text) = [networkResults map:^id(id value) {
105 }];
106
107
108
109
110
111
112
113
114
115
116
117
118
```

```
87  RAC(self.lastNameField.textColor) = fieldTextColor;
88  RAC(self.emailField.textColor) = fieldTextColor;
89  RAC(self.confirmEmailField.textColor) = fieldTextColor;
90
91  RACSignal *notExecuting = [executing not];
92
93  RAC(self.firstNameField.enabled) = notExecuting;
94  RAC(self.lastNameField.enabled) = notExecuting;
95  RAC(self.emailField.enabled) = notExecuting;
96  RAC(self.confirmEmailField.enabled) = notExecuting;
97
98  [executing subscribeNext:^(NSNumber *x) {
99      x.boolValue ? [self.loadingIndicator startAnimating] : [self.loadingIndicator stopA
100     self.statusLabel.textColor = [UIColor lightGrayColor];
101 }];
102
103 // Derive the status label's text and color from our network result
104 RAC(self.statusLabel.text) = [networkResults map:^id(RACEvent *x) {
105     return x.eventType == RACEventTypeError ? x.error.localizedFailureReason
106                                             : NSLocalizedString(@"Thanks for signing up
107 }];
108
109
110
111
112
113
114
115
116
117
118
```

```
87  RAC(self.lastNameField.textColor) = fieldTextColor;
88  RAC(self.emailField.textColor) = fieldTextColor;
89  RAC(self.confirmEmailField.textColor) = fieldTextColor;
90
91  RACSignal *notExecuting = [executing not];
92
93  RAC(self.firstNameField.enabled) = notExecuting;
94  RAC(self.lastNameField.enabled) = notExecuting;
95  RAC(self.emailField.enabled) = notExecuting;
96  RAC(self.confirmEmailField.enabled) = notExecuting;
97
98  [executing subscribeNext:^(NSNumber *x) {
99      x.boolValue ? [self.loadingIndicator startAnimating] : [self.loadingIndicator stopA
100     self.statusLabel.textColor = [UIColor lightGrayColor];
101 }];
102
103 // Derive the status label's text and color from our network result
104 RAC(self.statusLabel.text) = [networkResults map:^id(RACEvent *x) {
105     return x.eventType == RACEventTypeError ? x.error.localizedFailureReason
106                                             : NSLocalizedString(@"Thanks for signing up
107 }];
108 RAC(self.statusLabel.textColor) = [networkResults map:^id(RACEvent *x) {
109     return x.eventType == RACEventTypeError ? [UIColor redColor]
110                                             : [UIColor greenColor];
111 }];
112
113
114
115
116
117
118
```

```
    RACSignal *fieldTextColor = [executing map:^id(NSNumber *x) {
        return x.boolValue ? [UIColor lightGrayColor] : [UIColor blackColor];
    }];

    RAC(self.firstNameField.textColor) = fieldTextColor;
    RAC(self.lastNameField.textColor) = fieldTextColor;
    RAC(self.emailField.textColor) = fieldTextColor;
    RAC(self.confirmEmailField.textColor) = fieldTextColor;

    RACSignal *notExecuting = [executing not];

    RAC(self.firstNameField.enabled) = notExecuting;
    RAC(self.lastNameField.enabled) = notExecuting;
    RAC(self.emailField.enabled) = notExecuting;
    RAC(self.confirmEmailField.enabled) = notExecuting;

    [executing subscribeNext:^(NSNumber *x) {
        x.boolValue ? [self.loadingIndicator startAnimating] : [self.loadingIndicator stopA
        self.statusLabel.textColor = [UIColor lightGrayColor];
    }];

    // Derive the status label's text and color from our network result
    RAC(self.statusLabel.text) = [networkResults map:^id(RACEvent *x) {
        return x.eventType == RACEventTypeError ? x.error.localizedFailureReason
                                                : NSLocalizedString(@"Thanks for signing up
    }];
    RAC(self.statusLabel.textColor) = [networkResults map:^id(RACEvent *x) {
        return x.eventType == RACEventTypeError ? [UIColor redColor]
                                                : [UIColor greenColor];
    }];
}

@end
```

```
    RACSignal *buttonTextColor = [buttonEnabled map:^id(NSNumber *x) {
        return x.boolValue ? defaultButtonTitleColor : [UIColor lightGrayColor];
    }];

    [self.createButton rac_liftSelector:@selector(setTitleColor:forState:)
                            withObjects:buttonTextColor, @(UIControlStateNormal)];

    // bind button and text field state to create account command executing state

    RACSignal *executing = RACAble(createAccountCommand, executing);

    RACSignal *fieldTextColor = [executing map:^id(NSNumber *x) {
        return x.boolValue ? [UIColor lightGrayColor] : [UIColor blackColor];
    }];

    RAC(self.firstNameField.textColor) = fieldTextColor;
    RAC(self.lastNameField.textColor) = fieldTextColor;
    RAC(self.emailField.textColor) = fieldTextColor;
    RAC(self.confirmEmailField.textColor) = fieldTextColor;

    RACSignal *notExecuting = [executing not];

    RAC(self.firstNameField.enabled) = notExecuting;
    RAC(self.lastNameField.enabled) = notExecuting;
    RAC(self.emailField.enabled) = notExecuting;
    RAC(self.confirmEmailField.enabled) = notExecuting;

    [executing subscribeNext:^(NSNumber *x) {
        x.boolValue ? [self.loadingIndicator startAnimating] : [self.loadingIndicator stopA
        self.statusLabel.textColor = [UIColor lightGrayColor];
    }];

    // Derive the status label's text and color from our network result
```

```
        }];

    RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
    RACSignal *networkResults = [[[createAccountCommand
        addSignalBlock:^RACSignal *(id value) {
            // TODO: Create signal from APIClient's create account method
            return [RACSignal empty];
        }]
        switchToLatest]
        deliverOn:[RACScheduler mainThreadScheduler]];

    // bind create button's UI state and touch action
    [[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNe
        [createAccountCommand execute:sender];
    }];

    RACSignal *buttonEnabled = RACAbleWithStart(createAccountCommand, canExecute);
    RAC(self.createButton.enabled) = buttonEnabled;

    UIColor *defaultButtonTitleColor = self.createButton.titleLabel.textColor;
    RACSignal *buttonTextColor = [buttonEnabled map:^id(NSNumber *x) {
        return x.boolValue ? defaultButtonTitleColor : [UIColor lightGrayColor];
    }];

    [self.createButton rac_liftSelector:@selector(setTitleColor:forState:)
                            withObjects:buttonTextColor, @(UIControlStateNormal)];

    // bind button and text field state to create account command executing state

    RACSignal *executing = RACAble(createAccountCommand, executing);

    RACSignal *fieldTextColor = [executing map:^id(NSNumber *x) {
        return x.boolValue ? [UIColor lightGrayColor] : [UIColor blackColor];
```

```objc
#import <AFNetworking/AFNetworking.h>

@interface APIClient : AFHTTPClient

+ (instancetype)sharedClient;

- (void)createAccountForEmail:(NSString *)email
                    firstName:(NSString *)firstName
                     lastName:(NSString *)lastName
                      success:(void (^)(id account))successBlock
                      failure:(void (^)(NSError *error))failureBlock;

@end
```

```objc
#import <AFNetworking/AFNetworking.h>

@class RACSignal;

@interface APIClient : AFHTTPClient

+ (instancetype)sharedClient;

- (RACSignal *)createAccountForEmail:(NSString *)email
                           firstName:(NSString *)firstName
                            lastName:(NSString *)lastName
                             success:(void (^)(id account))successBlock
                             failure:(void (^)(NSError *error))failureBlock;

@end
```

```objc
#import <AFNetworking/AFNetworking.h>

@class RACSignal;

@interface APIClient : AFHTTPClient

+ (instancetype)sharedClient;

- (RACSignal *)createAccountForEmail:(NSString *)email
                           firstName:(NSString *)firstName
                            lastName:(NSString *)lastName;

@end
```

```
        apiClient = [APIClient new];
    });
    return apiClient;
}

- (id)init
{
    self = [super initWithBaseURL:[NSURL URLWithString:APIClientDefaultEndpoint]];
    if (self) {
        AFNetworkActivityIndicatorManager.sharedManager.enabled = YES;
        [self registerHTTPOperationClass:AFJSONRequestOperation.class];
        [self setDefaultHeader:@"Accept" value:@"application/json"];
    }
    return self;
}

- (void)createAccountForEmail:(NSString *)email
                    firstName:(NSString *)firstName
                     lastName:(NSString *)lastName
                      success:(void (^)(id account))successBlock
                      failure:(void (^)(NSError *error))failureBlock
{
    [self postPath:@"/accounts"
        parameters:@{ @"first_name": firstName, @"last_name": lastName, @"email": email, }
           success:^(AFHTTPRequestOperation *operation, id responseObject) {
               successBlock(responseObject);
           }
           failure:^(AFHTTPRequestOperation *operation, NSError *error) {
               id responseJSON = nil;
               if ([operation respondsToSelector:@selector(responseJSON)]) {
                   responseJSON = [(id)operation responseJSON];
               }
```

```objc
- (RACSignal *)createAccountForEmail:(NSString *)email
                           firstName:(NSString *)firstName
                            lastName:(NSString *)lastName
{
    [self postPath:@"/accounts"
        parameters:@{ @"first_name": firstName, @"last_name": lastName, @"email": email, }
           success:^(AFHTTPRequestOperation *operation, id responseObject) {
               successBlock(responseObject);
           }
           failure:^(AFHTTPRequestOperation *operation, NSError *error) {
               id responseJSON = nil;
               if ([operation respondsToSelector:@selector(responseJSON)]) {
                   responseJSON = [(id)operation responseJSON];
               }
               failureBlock([NSError errorWithDomain:@"com.example"
                                                code:error.code
                                            userInfo:@{
                                                NSLocalizedFailureReasonErrorKey: [response
                                                ?: @"Fail

                                            }]);
           }];
}

@end
```

```
        AFNetworkActivityIndicatorManager.sharedManager.enabled = YES;
        [self registerHTTPOperationClass:AFJSONRequestOperation.class];
        [self setDefaultHeader:@"Accept" value:@"application/json"];
    }
    return self;
}

- (RACSignal *)createAccountForEmail:(NSString *)email
                           firstName:(NSString *)firstName
                            lastName:(NSString *)lastName
{
    RACSubject *subject = [RACSubject subject];
    [self postPath:@"/accounts"
        parameters:@{ @"first_name": firstName, @"last_name": lastName, @"email": email, }
           success:^(AFHTTPRequestOperation *operation, id responseObject) {
               successBlock(responseObject);
           }
           failure:^(AFHTTPRequestOperation *operation, NSError *error) {
               id responseJSON = nil;
               if ([operation respondsToSelector:@selector(responseJSON)]) {
                   responseJSON = [(id)operation responseJSON];
               }
               failureBlock([NSError errorWithDomain:@"com.example"
                                                code:error.code
                                            userInfo:@{
                                                NSLocalizedFailureReasonErrorKey: [response
                                                ?: @"Fail

                                            }]);
           }];
    return subject;
}

@end
```

```
        AFNetworkActivityIndicatorManager.sharedManager.enabled = YES;
        [self registerHTTPOperationClass:AFJSONRequestOperation.class];
        [self setDefaultHeader:@"Accept" value:@"application/json"];
    }
    return self;
}

- (RACSignal *)createAccountForEmail:(NSString *)email
                           firstName:(NSString *)firstName
                            lastName:(NSString *)lastName
{
    RACSubject *subject = [RACSubject subject];

    [self postPath:@"/accounts"
        parameters:@{ @"first_name": firstName, @"last_name": lastName, @"email": email, }
           success:^(AFHTTPRequestOperation *operation, id responseObject) {
               [subject sendNext:responseObject];
               [subject sendCompleted];
           }
           failure:^(AFHTTPRequestOperation *operation, NSError *error) {
               id responseJSON = nil;
               if ([operation respondsToSelector:@selector(responseJSON)]) {
                   responseJSON = [(id)operation responseJSON];
               }
               failureBlock([NSError errorWithDomain:@"com.example"
                                                code:error.code
                                            userInfo:@{
                                                NSLocalizedFailureReasonErrorKey: [response
                                                ?: @"Fail

                                            }]);
           }];

    return subject;
```

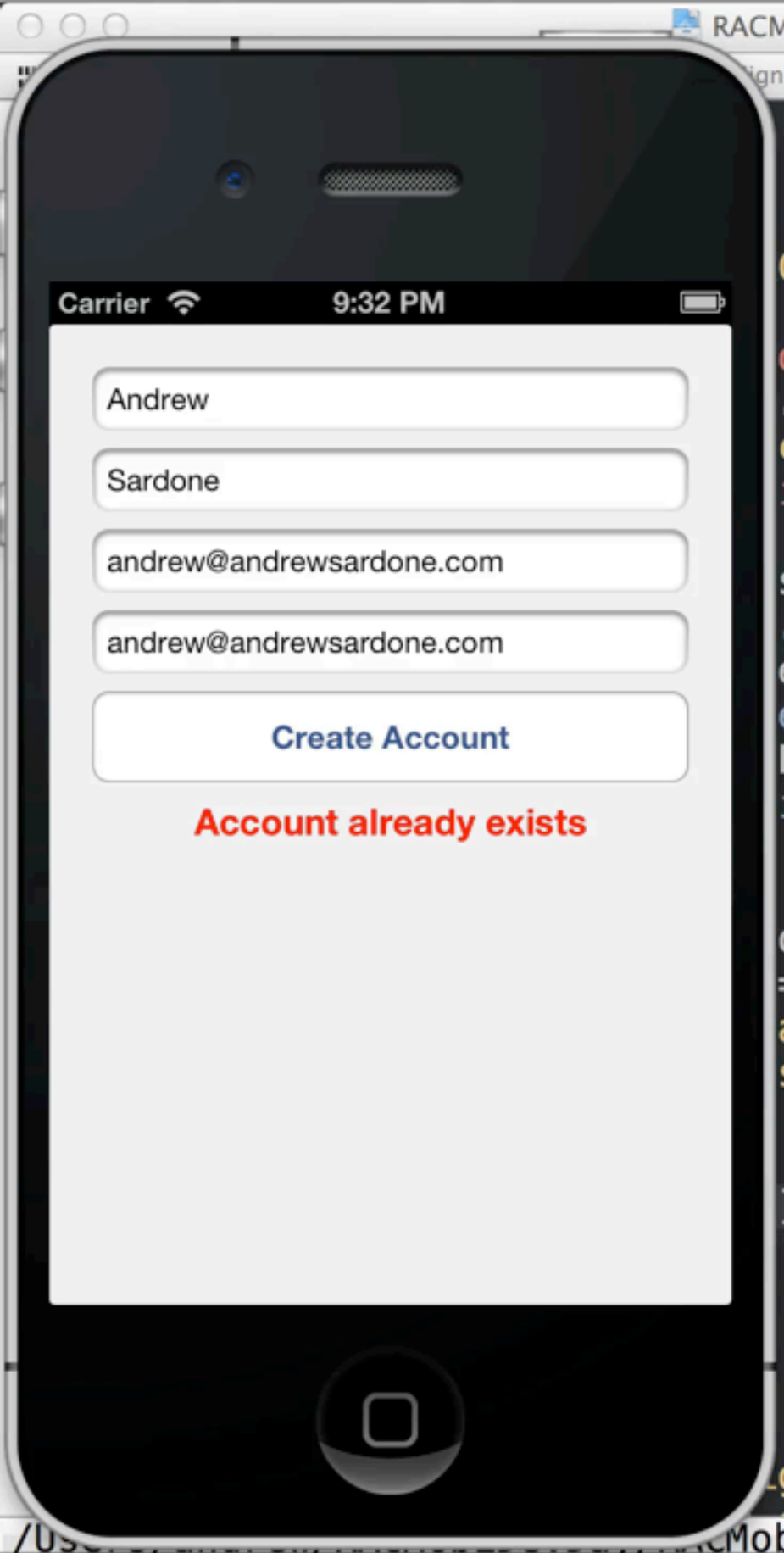
```
        AFNetworkActivityIndicatorManager.sharedManager.enabled = YES;
        [self registerHTTPOperationClass:AFJSONRequestOperation.class];
        [self setDefaultHeader:@"Accept" value:@"application/json"];
    }
    return self;
}

- (RACSignal *)createAccountForEmail:(NSString *)email
                           firstName:(NSString *)firstName
                            lastName:(NSString *)lastName
{
    RACSubject *subject = [RACSubject subject];

    [self postPath:@"/accounts"
        parameters:@{ @"first_name": firstName, @"last_name": lastName, @"email": email, }
           success:^(AFHTTPRequestOperation *operation, id responseObject) {
               [subject sendNext:responseObject];
               [subject sendCompleted];
           }
           failure:^(AFHTTPRequestOperation *operation, NSError *error) {
               id responseJSON = nil;
               if ([operation respondsToSelector:@selector(responseJSON)]) {
                   responseJSON = [(id)operation responseJSON];
               }
               [subject sendError:[NSError errorWithDomain:@"com.example"
                                                      code:error.code
                                                  userInfo:@{
                                                      NSLocalizedFailureReasonErrorKey: [response
                                                      ?: @"Fail

                                                  }]];
           }];

    return subject;
```

```
        self.firstNameField.rac_textSignal,
        self.lastNameField.rac_textSignal,
        self.emailField.rac_textSignal,
        self.confirmEmailField.rac_textSignal,
    ]
    reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confir
        return @(firstName.length > 0
                 && lastName.length > 0
                 && email.length > 0
                 && confirmEmail.length > 0
                 && [email isEqualToString:confirmEmail]);
    }];

RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
RACSignal *networkResults = [[[createAccountCommand
   addSignalBlock:^RACSignal *(id value) {
       // TODO: Create signal from APIClient's create account method
       return [RACSignal empty];
   }]
   switchToLatest]
   deliverOn:[RACScheduler mainThreadScheduler]];

// bind create button's UI state and touch action
[[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNe
    [createAccountCommand execute:sender];
}];

RACSignal *buttonEnabled = RACAbleWithStart(createAccountCommand, canExecute);
RAC(self.createButton.enabled) = buttonEnabled;

UIColor *defaultButtonTitleColor = self.createButton.titleLabel.textColor;
RACSignal *buttonTextColor = [buttonEnabled map:^id(NSNumber *x) {
```

```
        self.firstNameField.rac_textSignal,
        self.lastNameField.rac_textSignal,
        self.emailField.rac_textSignal,
        self.confirmEmailField.rac_textSignal,
    ]
    reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confir
        return @(firstName.length > 0
                 && lastName.length > 0
                 && email.length > 0
                 && confirmEmail.length > 0
                 && [email isEqualToString:confirmEmail]);
    }];

RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
RACSignal *networkResults = [[[createAccountCommand
   addSignalBlock:^RACSignal *(id value) {
       return [APIClient.sharedClient createAccountForEmail:self.emailField.text
                                                   firstName:self.firstNameField.text
                                                    lastName:self.lastNameField.text]
   }]
   switchToLatest]
   deliverOn:[RACScheduler mainThreadScheduler]];

// bind create button's UI state and touch action
[[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNe
    [createAccountCommand execute:sender];
}];

RACSignal *buttonEnabled = RACAbleWithStart(createAccountCommand, canExecute);
RAC(self.createButton.enabled) = buttonEnabled;

UIColor *defaultButtonTitleColor = self.createButton.titleLabel.textColor;
```

```
        self.firstNameField.rac_textSignal,
        self.lastNameField.rac_textSignal,
        self.emailField.rac_textSignal,
        self.confirmEmailField.rac_textSignal,
    ]
    reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confir
        return @(firstName.length > 0
                 && lastName.length > 0
                 && email.length > 0
                 && confirmEmail.length > 0
                 && [email isEqualToString:confirmEmail]);
    }];

RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
RACSignal *networkResults = [[[createAccountCommand
   addSignalBlock:^RACSignal *(id value) {
       return [[APIClient.sharedClient createAccountForEmail:self.emailField.text
                                                    firstName:self.firstNameField.text
                                                     lastName:self.lastNameField.text]
                materialize];
    }]
    switchToLatest]
    deliverOn:[RACScheduler mainThreadScheduler]];

// bind create button's UI state and touch action
[[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNe
    [createAccountCommand execute:sender];
}];

RACSignal *buttonEnabled = RACAbleWithStart(createAccountCommand, canExecute);
RAC(self.createButton.enabled) = buttonEnabled;
```

```
{
    [super viewDidLoad];

    RACSignal *formValid = [RACSignal
        combineLatest:@[
            self.firstNameField.rac_textSignal,
            self.lastNameField.rac_textSignal,
            self.emailField.rac_textSignal,
            self.confirmEmailField.rac_textSignal,
        ]
        reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confir
            return @(firstName.length > 0
                     && lastName.length > 0
                     && email.length > 0
                     && confirmEmail.length > 0
                     && [email isEqualToString:confirmEmail]);
        }];

    RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
    RACSignal *networkResults = [[[createAccountCommand
        addSignalBlock:^RACSignal *(id value) {
            return [[APIClient.sharedClient createAccountForEmail:self.emailField.text
                                                         firstName:self.firstNameField.text
                                                          lastName:self.lastNameField.text]
                    materialize];
        }]
        switchToLatest]
        deliverOn:[RACScheduler mainThreadScheduler]];

    // bind create button's UI state and touch action
    [[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNe
        [createAccountCommand execute:sender];
```




RACMobiDevDay.xcworkspace — SignUpViewController.m

SignUpViewController.m 〉 M -viewDidLoad

```
CSignal

d.rac_textSignal,
.rac_textSignal,
c_textSignal,
ield.rac_textSignal,

stName, NSString *lastName, NSString *email, NSString *confir
.length > 0
e.length > 0
ength > 0
Email.length > 0
isEqualToString:confirmEmail]);


ommand = [RACCommand commandWithCanExecuteSignal:formValid];
= [[[createAccountCommand
al *(id value) {
sharedClient createAccountForEmail:self.emailField.text
                         firstName:self.firstNameField.text
                          lastName:self.lastNameField.text]
];


 mainThreadScheduler]];

 state and touch action
gnalForControlEvents:UIControlEventTouchUpInside] subscribeNe
```

/RACMobiDevDay/SignUpViewController.m

RACMobiDevDay

*Reduce unnecessary state*
*Better code locality*
*Fluent interface*
*Simple data structures*
*Composing functions*

# Thanks!

github.com/ReactiveCocoa

pinboard.in/u:andrewsardone/t:ReactiveCocoa

github.com/andrewsardone/RACMobiDevDay

nsbrief.com/81-justin-spahr-summers/

Input and Output – j.mp/joshaberio

Reactive Extensions (Rx) – j.mp/reactiveextensions

**Andrew Sardone**

twitter: @andrewa2 app.net: @andrewsardone github.com/andrewsardone