

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/joshabero>

*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?
```

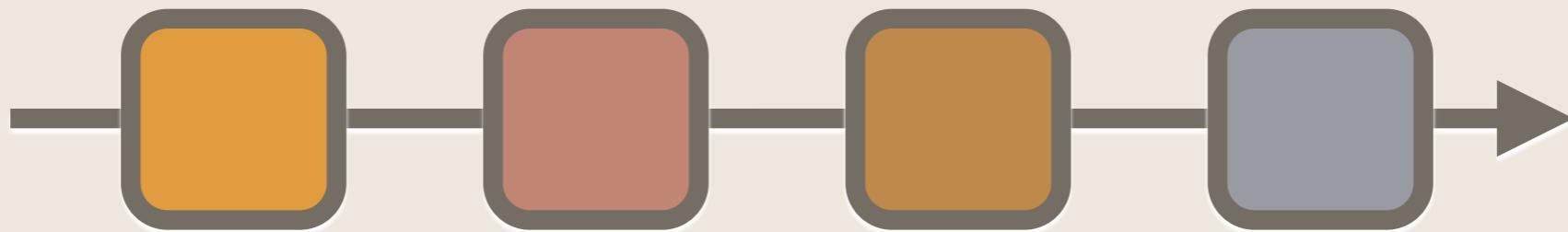
=SUM(A1, B1)

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

*Data types that
represent a value
over time*

Streams

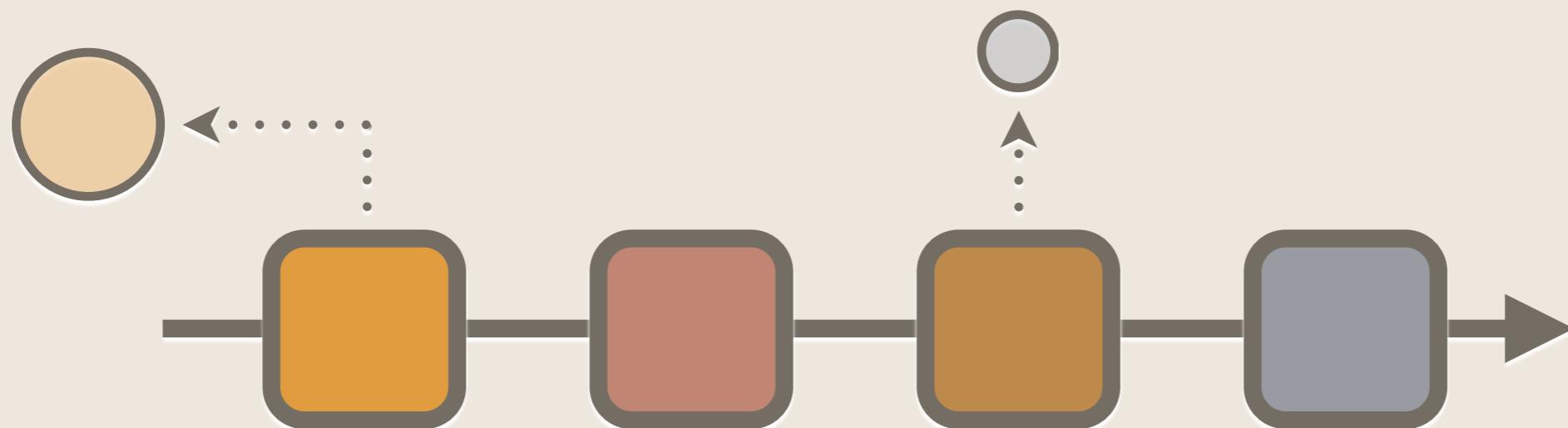
sequences of values over time



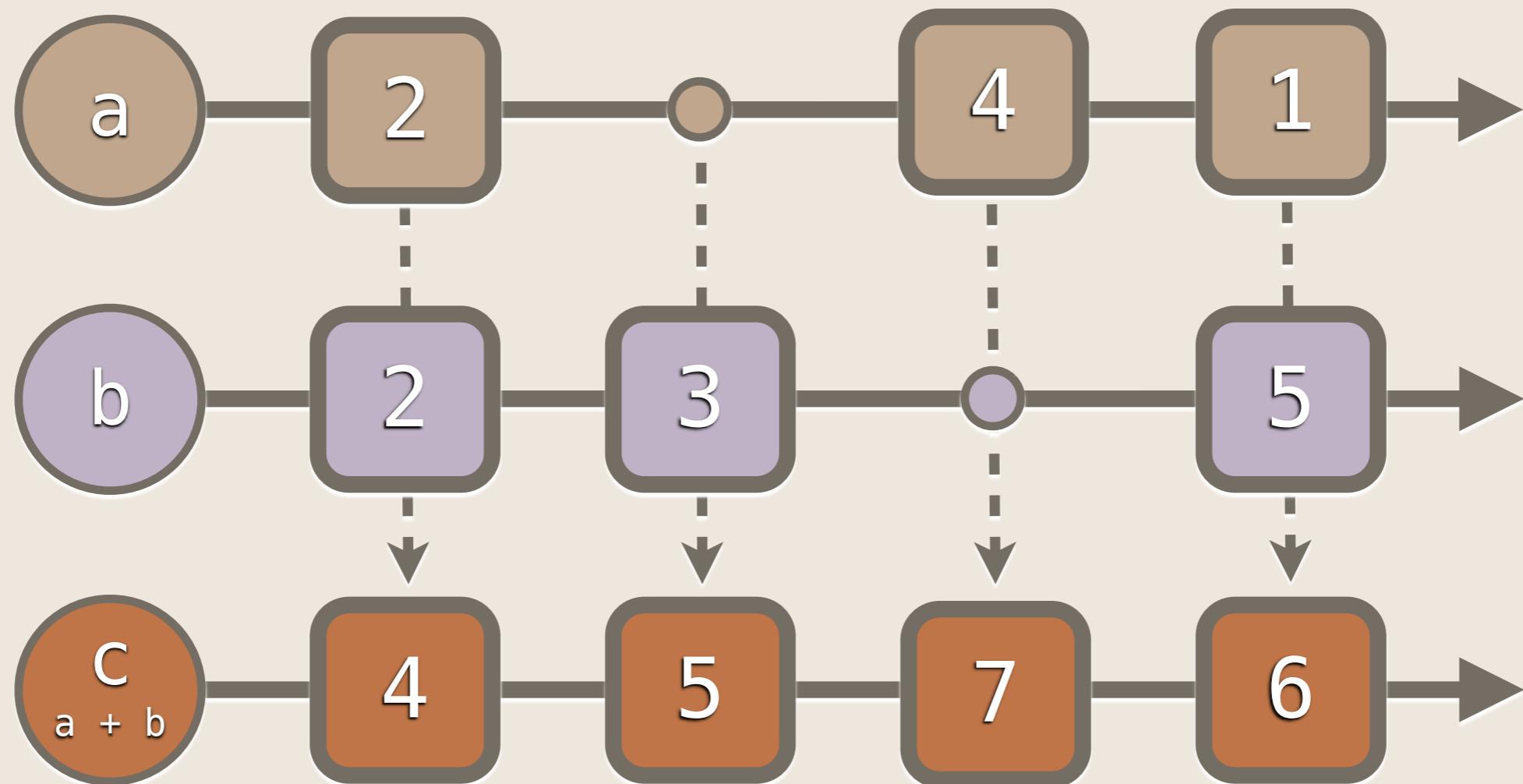
Signals

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

@interface RACSignal : RACStream

next

completed

error

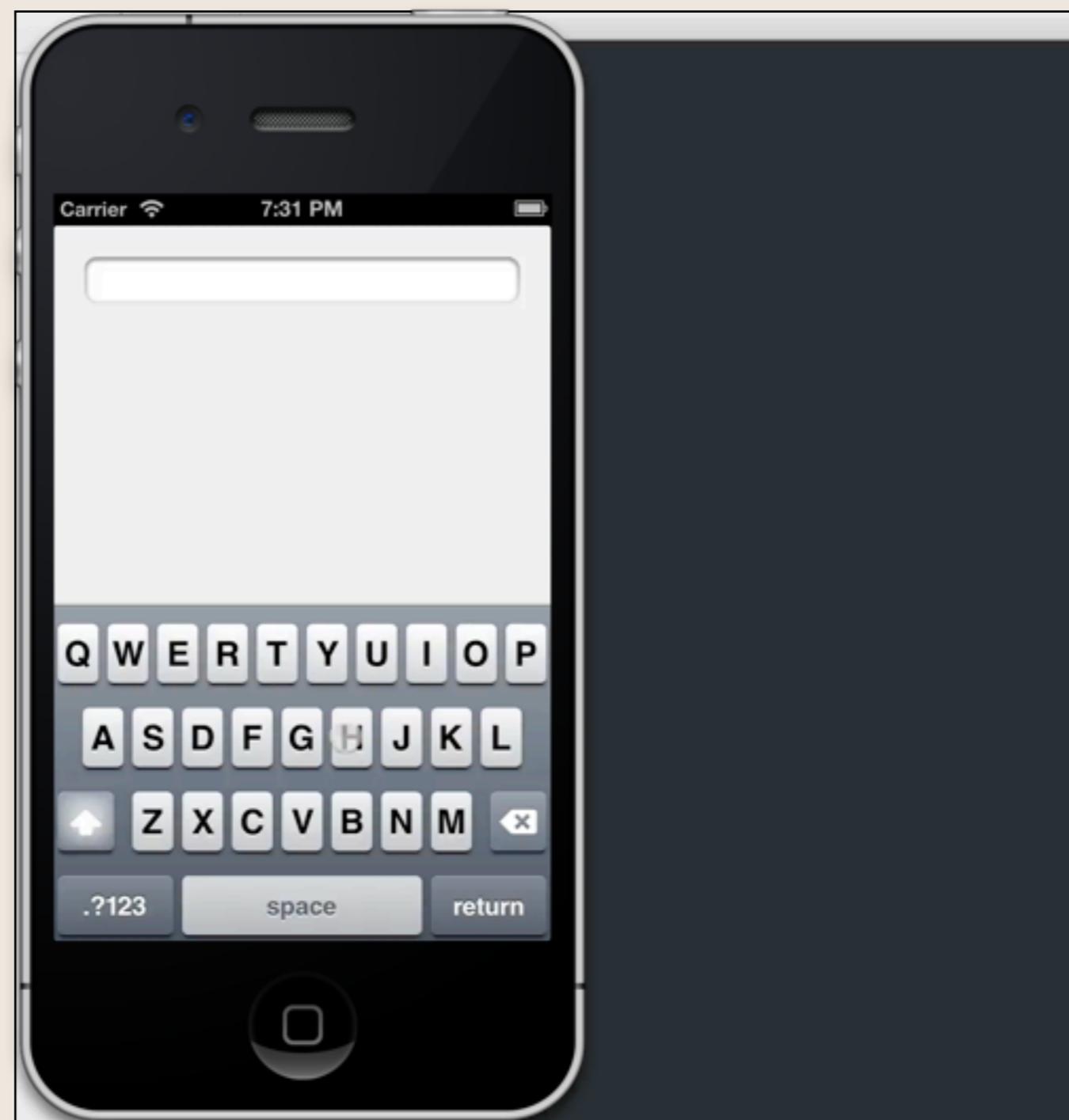
**sending a single
value**

**done sending all
values &
successful**

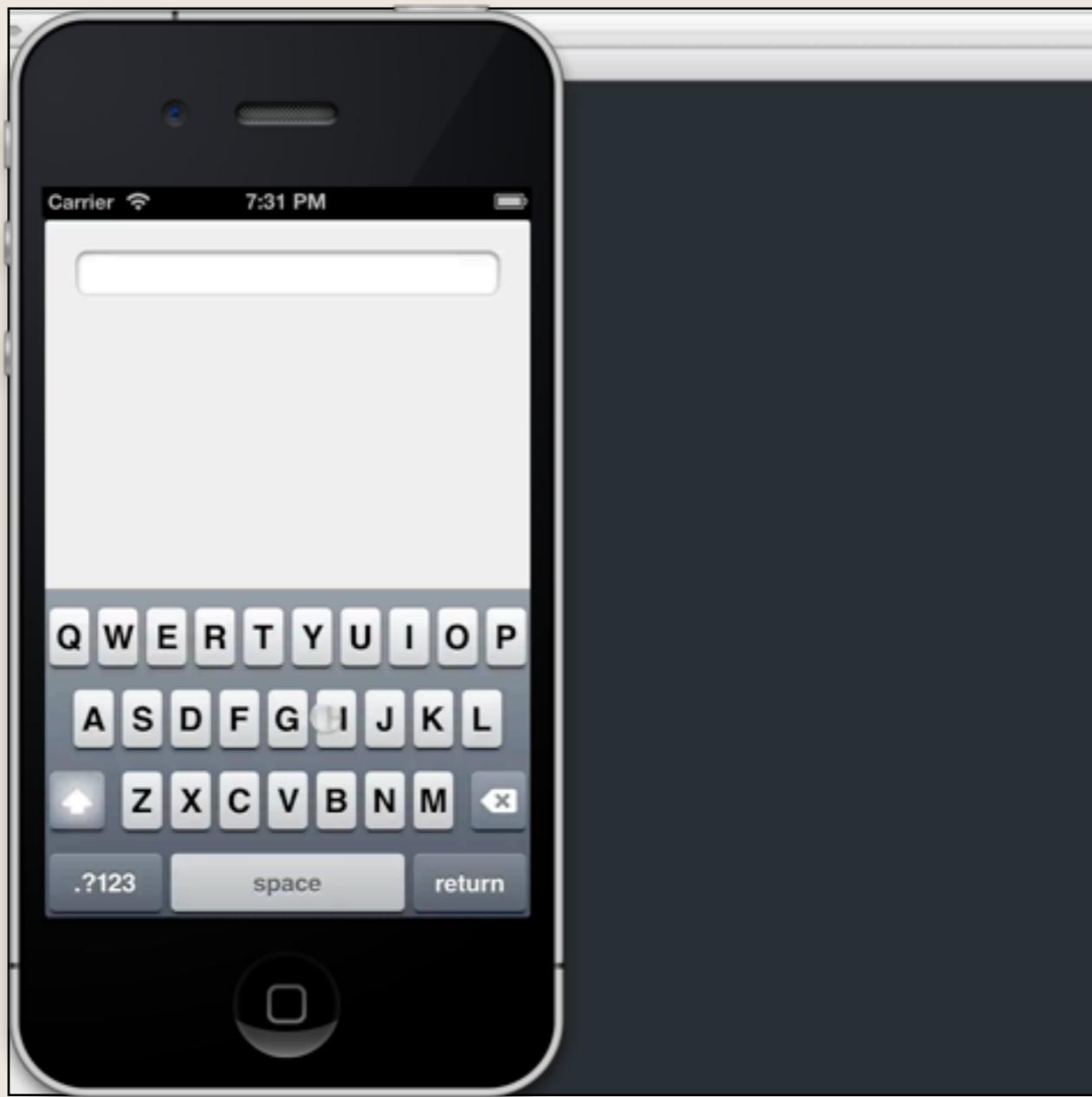
**sending values
but terminated in
error**

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

```
[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);  
}];
```



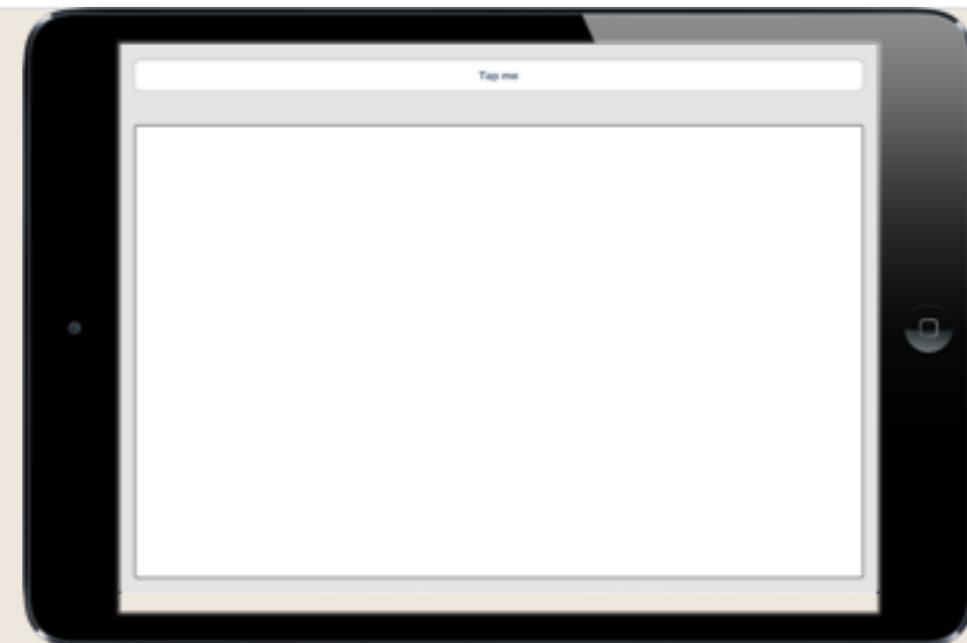
```

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:)
    withObject:html, url];

```



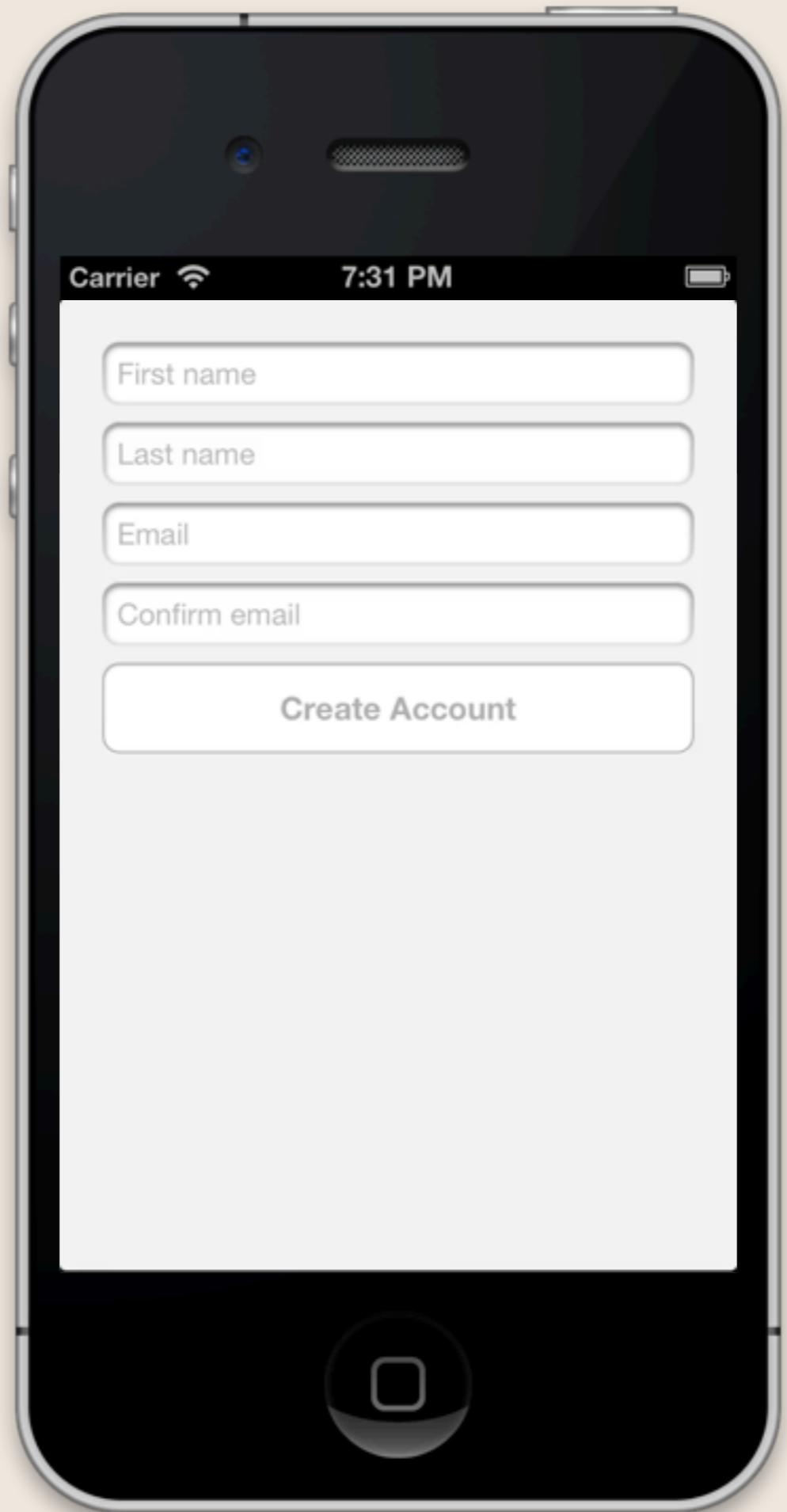
... less imperative

```
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



```
RACMobiDevDay.xcworkspace — SignUpViewController.m
RACMobiDevDay > RACMobiDevDay > SignUpViewController.m > No Selection
1 #import "SignUpViewController.h"
2 #import "APIClient.h"
3
4 @interface SignUpViewController () {
5     @property (weak, nonatomic) IBOutlet UITextField *firstNameField;
6     @property (weak, nonatomic) IBOutlet UITextField *lastNameField;
7     @property (weak, nonatomic) IBOutlet UITextField *emailField;
8     @property (weak, nonatomic) IBOutlet UITextField *confirmEmailField;
9     @property (weak, nonatomic) IBOutlet UIButton *createButton;
10    @property (weak, nonatomic) IBOutlet UILabel *statusLabel;
11    @property (weak, nonatomic) IBOutlet UIActivityIndicatorView *loadingIndicator;
12
13    @property (strong, nonatomic) UIColor *enabledButtonColor;
14    @property (strong, nonatomic) UIColor *disabledButtonColor;
15
16    @property (strong, nonatomic) UIColor *defaultTextColor;
17    @property (strong, nonatomic) UIColor *loadingTextColor;
18 }
19
20 @implementation SignUpViewController
21
22 - (id)init
23 {
24     self = [super initWithNibName:NSStringFromClass(self.class) bundle:nil];
25     return self;
26 }
27
28 - (id)initWithNibName:(NSString *)nibNameOrNilOrNil bundle:(NSBundle *)nibBundleOrNilOrNil
29 {
30     return [self init];
31 }
32
33 - (void)viewDidLoad
34 /Users/andrew/RACMobiDevDay/RACMobiDevDay/SignUpViewController.m
```

RACMobiDevDay.xcworkspace — SignUpViewController.m

RACMobiDevDay > RACMobiDevDay > SignUpViewController.m > P firstNameField

```
1 #import "SignUpViewController.h"
2 #import "APIClient.h"
3
4 @interface SignUpViewController ()
5 @property (weak, nonatomic) IBOutlet UITextField *firstNameField;
6 @property (weak, nonatomic) IBOutlet UITextField *lastNameField;
7 @property (weak, nonatomic) IBOutlet UITextField *emailField;
8 @property (weak, nonatomic) IBOutlet UITextField *confirmEmailField;
9 @property (weak, nonatomic) IBOutlet UIButton *createButton;
10 @property (weak, nonatomic) IBOutlet UILabel *statusLabel;
11 @property (weak, nonatomic) IBOutlet UIActivityIndicatorView *loadingIndicator;
12
13 @property (strong, nonatomic) UIColor *enabledButtonColor;
14 @property (strong, nonatomic) UIColor *disabledButtonColor;
15
16 @property (strong, nonatomic) UIColor *defaultTextColor;
17 @property (strong, nonatomic) UIColor *loadingTextColor;
18 @end
19
20 @implementation SignUpViewController
21
22 - (id)init
23 {
24     self = [super initWithNibName:NSStringFromClass(self.class) bundle:nil];
25     return self;
26 }
27
28 - (id)initWithNibName:(NSString *)nibNameOrNilOrNil bundle:(NSBundle *)nibBundleOrNilOrNil
29 {
30     return [self init];
31 }
32
33 - (void)viewDidLoad
34 /Users/andrew/RACMobiDevDay/RACMobiDevDay/SignUpViewController.m
-- VISIBLE LINE --
```

RACMobiDevDay.xcworkspace — SignUpViewController.m

RACMobiDevDay > RACMobiDevDay > SignUpViewController.m > @interface SignUpViewController()

```
1 #import "SignUpViewController.h"
2 #import "APIClient.h"

3
4 @interface SignUpViewController () {
5     @property (weak, nonatomic) IBOutlet UITextField *firstNameField;
6     @property (weak, nonatomic) IBOutlet UITextField *lastNameField;
7     @property (weak, nonatomic) IBOutlet UITextField *emailField;
8     @property (weak, nonatomic) IBOutlet UITextField *confirmEmailField;
9     @property (weak, nonatomic) IBOutlet UIButton *createButton;
10    @property (weak, nonatomic) IBOutlet UILabel *statusLabel;
11    @property (weak, nonatomic) IBOutlet UIActivityIndicatorView *loadingIndicator;
12
13    @property (strong, nonatomic) UIColor *enabledButtonColor;
14    @property (strong, nonatomic) UIColor *disabledButtonColor;
15
16    @property (strong, nonatomic) UIColor *defaultTextColor;
17    @property (strong, nonatomic) UIColor *loadingTextColor;
18 }

19
20 @implementation SignUpViewController
21
22 - (id)init
23 {
24     self = [super initWithNibName:NSStringFromClass(self.class) bundle:nil];
25     return self;
26 }
27
28 - (id)initWithNibName:(NSString *)nibNameOrNilOrNil bundle:(NSBundle *)nibBundleOrNilOrNil
29 {
30     return [self init];
31 }
32
33 - (void)viewDidLoad
34 /Users/andrew/RACMobiDevDay/RACMobiDevDay/SignUpViewController.m
-- VISIBLE LINE --
```

```
33 - (void)viewDidLoad
34 {
35     [super viewDidLoad];
36
37     // initial button setup
38     self.enabledButtonColor = self.createButton.titleLabel.textColor;
39     self.disabledButtonColor = UIColor.lightGrayColor;
40     [self.createButton setTitleColor:self.disabledButtonColor
41                             forState:UIControlStateNormal];
42     self.createButton.titleLabel.textColor = UIColor.lightGrayColor;
43     self.createButton.enabled = NO;
44     [self.createButton addTarget:self
45                           action:@selector(createButtonTouched:)
46                         forControlEvents:UIControlEventTouchUpInside];
47
48     // initial text field setup
49     self.defaultTextColor = self.firstNameField.textColor;
50     self.loadingTextColor = UIColor.lightGrayColor;
51
52     // listen on text editing changes
53     [self.firstNameField addTarget:self
54                               action:@selector(textFieldChanged:)
55                         forControlEvents:UIControlEventEditingChanged];
56     [self.lastNameField addTarget:self
57                               action:@selector(textFieldChanged:)
58                         forControlEvents:UIControlEventEditingChanged];
59     [self.emailField addTarget:self
60                               action:@selector(textFieldChanged:)
61                         forControlEvents:UIControlEventEditingChanged];
62     [self.confirmEmailField addTarget:self
63                               action:@selector(textFieldChanged:)
64                         forControlEvents:UIControlEventEditingChanged];
65 }
```

```
RACMobiDevDay.xcworkspace — SignUpViewController.m
RACMobiDevDay > RACMobiDevDay > SignUpViewController.m - textFieldChanged:
70 - (void)textFieldChanged:(UITextField *)textField
71 {
72     self.createButton.enabled = self.isFormValid;
73     [self.createButton setTitleColor:(self.createButton.isEnabled
74                                 ? self.enabledButtonColor
75                                 : self.disabledButtonColor)
76      forState:UIControlStateNormal];
77 }
78
79 - (void)createButtonTouched:(UIButton *)createButton
80 {
81     [self updateUIForNetworkActivity:YES];
82
83     self.statusLabel.textColor = UIColor.lightGrayColor;
84
85     [APIClient.sharedClient createAccountForEmail:self.emailField.text
86                                              firstName:self.firstNameField.text
87                                              lastName:self.lastNameField.text
88                                              success:^(id account) {
89         self.statusLabel.textColor = [UIColor colorWithRed:0.9 green:0.9 blue:0.9];
90         self.statusLabel.text = NSLocalizedString(@"Account created successfully!");
91
92         [self updateUIForNetworkActivity:NO];
93     }
94     failure:^(NSError *error) {
95         self.statusLabel.textColor = [UIColor colorWithRed:0.9 green:0.9 blue:0.9];
96         self.statusLabel.text = error.localizedDescription;
97
98         [self updateUIForNetworkActivity:NO];
99     }];
100 }
101
102 #pragma mark Validation
103 /Users/andrew/RACMobiDevDay/RACMobiDevDay/SignUpViewController.m
```

```
RACMobiDevDay.xcworkspace — SignUpViewController.m
RACMobiDevDay > RACMobiDevDay > SignUpViewController.m -isFormValid
104 - (BOOL)isFormValid
105 {
106     NSString *firstName = self.firstNameField.text;
107     NSString *lastName = self.lastNameField.text;
108     NSString *email = self.emailField.text;
109     NSString *confirmEmail = self.confirmEmailField.text;
110
111     return firstName.length > 0
112         && lastName.length > 0
113         && email.length > 0
114         && confirmEmail.length > 0
115         && [email isEqualToString:confirmEmail];
116 }
117
118 #pragma mark UI Updates
119
120 - (void)updateUIForNetworkActivity:(BOOL)isNetworkActivity
121 {
122     self.createButton.enabled = !isNetworkActivity;
123     [self.createButton setTitleColor:(isNetworkActivity
124                                 ? self.disabledButtonColor
125                                 : self.enabledButtonColor)
126      forState:UIControlStateNormal];
127
128     self.firstNameField.enabled
129         = self.lastNameField.enabled
130         = self.emailField.enabled
131         = self.confirmEmailField.enabled = !isNetworkActivity;
132
133     self.firstNameField.textColor
134         = self.lastNameField.textColor
135         = self.emailField.textColor
136         = self.confirmEmailField.textColor

```

```
79 - (void)createButtonTouched:(UIButton *)createButton
80 {
81     [self updateUIForNetworkActivity:YES];
82
83     self.statusLabel.textColor = UIColor.lightGrayColor;
84
85     [APIClient.sharedClient createAccountForEmail:self.emailField.text
86                                              firstName:self.firstNameField.text
87                                              lastName:self.lastNameField.text
88                                              success:^(id account) {
89         self.statusLabel.textColor = [UIColor colorWithRed:0.9 green:0.9 blue:0.9];
90         self.statusLabel.text = NSLocalizedString(@"Account created successfully!");
91
92         [self updateUIForNetworkActivity:NO];
93     }
94     failure:^(NSError *error) {
95         self.statusLabel.textColor = [UIColor colorWithRed:0.9 green:0.9 blue:0.9];
96         self.statusLabel.text = error.localizedDescription;
97     }];
98
99 }
100
101 #pragma mark Validation
102
103 - (BOOL)isValidForm
104 {
105
106     NSString *firstName = self.firstNameField.text;
107     NSString *lastName = self.lastNameField.text;
108     NSString *email = self.emailField.text;
109     NSString *confirmEmail = self.confirmEmailField.text;
110
111     return firstName.length > 0
112 }
```

```
120 - (void)updateUIForNetworkActivity:(BOOL)isNetworkActivity
121 {
122     self.createButton.enabled = !isNetworkActivity;
123     [self.createButton setTitleColor:(isNetworkActivity
124                             ? self.disabledButtonColor
125                             : self.enabledButtonColor)
126      forState:UIControlStateNormal];
127
128     self.firstNameField.enabled
129         = self.lastNameField.enabled
130         = self.emailField.enabled
131         = self.confirmEmailField.enabled = !isNetworkActivity;
132
133     self.firstNameField.textColor
134         = self.lastNameField.textColor
135         = self.emailField.textColor
136         = self.confirmEmailField.textColor
137         = (isNetworkActivity ? self.loadingTextColor : self.defaultTextColor);
138
139     isNetworkActivity ? [self.loadingIndicator startAnimating]
140                       : [self.loadingIndicator stopAnimating];
141 }
142
143 @end
144
```

```
1 platform :ios, '5.0'  
2  
3 pod 'AFNetworking', '~> 1.2.0'  
4 pod 'ReactiveCocoa', '~> 1.5.0'|
```

```
> 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
```

```
> |
```

```
RACMobiDevDay.xcworkspace — SignUpViewController.m
RACMobiDevDay > RACMobiDevDay > SignUpViewController.m > No Selection
1 #import "SignUpViewController.h"
2 #import "APIClient.h"
3
4 @interface SignUpViewController () {
5     @property (weak, nonatomic) IBOutlet UITextField *firstNameField;
6     @property (weak, nonatomic) IBOutlet UITextField *lastNameField;
7     @property (weak, nonatomic) IBOutlet UITextField *emailField;
8     @property (weak, nonatomic) IBOutlet UITextField *confirmEmailField;
9     @property (weak, nonatomic) IBOutlet UIButton *createButton;
10    @property (weak, nonatomic) IBOutlet UILabel *statusLabel;
11    @property (weak, nonatomic) IBOutlet UIActivityIndicatorView *loadingIndicator;
12
13    @property (strong, nonatomic) UIColor *enabledButtonColor;
14    @property (strong, nonatomic) UIColor *disabledButtonColor;
15
16    @property (strong, nonatomic) UIColor *defaultTextColor;
17    @property (strong, nonatomic) UIColor *loadingTextColor;
18 }
19
20 @implementation SignUpViewController
21
22 - (id)init
23 {
24     self = [super initWithNibName:NSStringFromClass(self.class) bundle:nil];
25     return self;
26 }
27
28 - (id)initWithNibName:(NSString *)nibNameOrNilOrNil bundle:(NSBundle *)nibBundleOrNilOrNil
29 {
30     return [self init];
31 }
32
33 - (void)viewDidLoad
34 /Users/andrew/RACMobiDevDay/RACMobiDevDay/SignUpViewController.m
```

```
RACMobiDevDay.xcworkspace — SignUpViewController.m
RACMobiDevDay > RACMobiDevDay > SignUpViewController.m > No Selection
1 #import "SignUpViewController.h"
2 #import "APIClient.h"
3 #import <ReactiveCocoa/ReactiveCocoa.h>
4
5 @interface SignUpViewController ()
6 @property (weak, nonatomic) IBOutlet UITextField *firstNameField;
7 @property (weak, nonatomic) IBOutlet UITextField *lastNameField;
8 @property (weak, nonatomic) IBOutlet UITextField *emailField;
9 @property (weak, nonatomic) IBOutlet UITextField *confirmEmailField;
10 @property (weak, nonatomic) IBOutlet UIButton *createButton;
11 @property (weak, nonatomic) IBOutlet UILabel *statusLabel;
12 @property (weak, nonatomic) IBOutlet UIActivityIndicatorView *loadingIndicator;
13
14 @property (strong, nonatomic) UIColor *enabledButtonColor;
15 @property (strong, nonatomic) UIColor *disabledButtonColor;
16
17 @property (strong, nonatomic) UIColor *defaultTextColor;
18 @property (strong, nonatomic) UIColor *loadingTextColor;
19 @end
20
21 @implementation SignUpViewController
22
23 - (id)init
24 {
25     self = [super initWithNibName:NSStringFromClass(self.class) bundle:nil];
26     return self;
27 }
28
29 - (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
30 {
31     return [self init];
32 }
33
```

```
105 - (BOOL)isValidForm
106 {
107     NSString *firstName = self.firstNameField.text;
108     NSString *lastName = self.lastNameField.text;
109     NSString *email = self.emailField.text;
110     NSString *confirmEmail = self.confirmEmailField.text;
111
112     return firstName.length > 0
113         && lastName.length > 0
114         && email.length > 0
115         && confirmEmail.length > 0
116         && [email isEqualToString:confirmEmail];
117 }
118
119 #pragma mark UI Updates
120
121 - (void)updateUIForNetworkActivity:(BOOL)isNetworkActivity
122 {
123     self.createButton.enabled = !isNetworkActivity;
124     [self.createButton setTitleColor:(isNetworkActivity
125                                 ? self.disabledButtonColor
126                                 : self.enabledButtonColor)
127      forState:UIControlStateNormal];
128
129     self.firstNameField.enabled
130         = self.lastNameField.enabled
131         = self.emailField.enabled
132         = self.confirmEmailField.enabled = !isNetworkActivity;
133
134     self.firstNameField.textColor
135         = self.lastNameField.textColor
136         = self.emailField.textColor
137         = self.confirmEmailField.textColor
```

```
34 - (void)viewDidLoad
35 {
36     [super viewDidLoad];
37
38
39
40
41
42 #pragma mark Validation
43
44 - (BOOL)isValidForm
45 {
46     NSString *firstName = self.firstNameField.text;
47     NSString *lastName = self.lastNameField.text;
48     NSString *email = self.emailField.text;
49     NSString *confirmEmail = self.confirmEmailField.text;
50
51     return firstName.length > 0
52         && lastName.length > 0
53         && email.length > 0
54         && confirmEmail.length > 0
55         && [email isEqualToString:confirmEmail];
56 }
57
58 // initial button setup
59 self.enabledButtonColor = self.createButton.titleLabel.textColor;
60 self.disabledButtonColor = UIColor.lightGrayColor;
61 [self.createButton setTitleColor:self.disabledButtonColor
62                             forState:UIControlStateNormal];
63 self.createButton.titleLabel.textColor = UIColor.lightGrayColor;
64 self.createButton.enabled = NO;
65 [self.createButton addTarget:self
66                         action:@selector(buttonTapped)
67               forControlEvents:UIControlEventTouchUpInside];
```

```
RACMobiDevDay.xcworkspace — SignUpViewController.m
RACMobiDevDay > RACMobiDevDay > SignUpViewController.m - viewDidLoad
34 - (void)viewDidLoad
35 {
36     [super viewDidLoad];
37
38     RACSignal *formValid = [RACSignal combineLatest:@[] reduce:nil];
39
40
41
42
43 #pragma mark Validation
44
45 - (BOOL)isValid
46 {
47     NSString *firstName = self.firstNameField.text;
48     NSString *lastName = self.lastNameField.text;
49     NSString *email = self.emailField.text;
50     NSString *confirmEmail = self.confirmEmailField.text;
51
52     return firstName.length > 0
53         && lastName.length > 0
54         && email.length > 0
55         && confirmEmail.length > 0
56         && [email isEqualToString:confirmEmail];
57 }
58
59
60 // initial button setup
61 self.enabledButtonColor = self.createButton.titleLabel.textColor;
62 self.disabledButtonColor = UIColor.lightGrayColor;
63 [self.createButton setTitleColor:self.disabledButtonColor
64                         forState:UIControlStateNormal];
65 self.createButton.titleLabel.textColor = UIColor.lightGrayColor;
66 self.createButton.enabled = NO;

```

```
34 - (void)viewDidLoad
35 {
36     [super viewDidLoad];
37
38     RACSignal *formValid = [RACSignal
39         combineLatest:@[
40             self.firstNameField.rac_textSignal,
41             self.lastNameField.rac_textSignal,
42             self.emailField.rac_textSignal,
43             self.confirmEmailField.rac_textSignal,
44         ]
45         reduce:nil];
46
47
48
49
50 #pragma mark Validation
51
52 - (BOOL)isValidForm
53 {
54     NSString *firstName = self.firstNameField.text;
55     NSString *lastName = self.lastNameField.text;
56     NSString *email = self.emailField.text;
57     NSString *confirmEmail = self.confirmEmailField.text;
58
59     return firstName.length > 0
60         && lastName.length > 0
61         && email.length > 0
62         && confirmEmail.length > 0
63         && [email isEqualToString:confirmEmail];
64 }
65
66
```

RACMobiDevDay.xcworkspace — SignUpViewController.m

RACMobiDevDay > RACMobiDevDay > SignUpViewController.m - viewDidLoad

```
34 - (void)viewDidLoad
35 {
36     [super viewDidLoad];
37
38     RACSignal *formValid = [RACSignal
39         combineLatest:@[
40             self.firstNameField.rac_textSignal,
41             self.lastNameField.rac_textSignal,
42             self.emailField.rac_textSignal,
43             self.confirmEmailField.rac_textSignal,
44         ]
45         reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confirmEmail) {
46             return @(firstName.length > 0
47                     && lastName.length > 0
48                     && email.length > 0
49                     && confirmEmail.length > 0
50                     && [email isEqualToString:confirmEmail]);
51     }];
52
53
54
55
56 #pragma mark Validation
57
58 - (BOOL)isValidForm
59 {
60     NSString *firstName = self.firstNameField.text;
61     NSString *lastName = self.lastNameField.text;
62     NSString *email = self.emailField.text;
63     NSString *confirmEmail = self.confirmEmailField.text;
64
65     return firstName.length > 0
66         && lastName.length > 0
67         && email.length > 0
68         && confirmEmail.length > 0
69         && [email isEqualToString:confirmEmail];
70 }
```

```
34 - (void)viewDidLoad
35 {
36     [super viewDidLoad];
37
38     RACSignal *formValid = [RACSignal
39         combineLatest:@[
40             self.firstNameField.rac_textSignal,
41             self.lastNameField.rac_textSignal,
42             self.emailField.rac_textSignal,
43             self.confirmEmailField.rac_textSignal,
44         ]
45         reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confirmEmail) {
46             return @(firstName.length > 0
47                     && lastName.length > 0
48                     && email.length > 0
49                     && confirmEmail.length > 0
50                     && [email isEqualToString:confirmEmail]);
51         }];
52
53
54
55
56     // initial button setup
57     self.enabledButtonColor = self.createButton.titleLabel.textColor;
58     self.disabledButtonColor = UIColor.lightGrayColor;
59     [self.createButton setTitleColor:self.disabledButtonColor
60                             forState:UIControlStateNormal];
61     self.createButton.titleLabel.textColor = UIColor.lightGrayColor;
62     self.createButton.enabled = NO;
63     [self.createButton addTarget:self
64                           action:@selector(createButtonTouched:)
65                           forControlEvents:UIControlEventTouchUpInside];
66 }
```

```
34 - (void)viewDidLoad
35 {
36     [super viewDidLoad];
37
38     RACSignal *formValid = [RACSignal
39         combineLatest:@[
40             self.firstNameField.rac_textSignal,
41             self.lastNameField.rac_textSignal,
42             self.emailField.rac_textSignal,
43             self.confirmEmailField.rac_textSignal,
44         ]
45         reduce:^^(NSString *firstName, NSString *lastName, NSString *email, NSString *confirmEmail)
46         {
47             return @(firstName.length > 0
48                     && lastName.length > 0
49                     && email.length > 0
50                     && confirmPassword.length > 0
51                     && [email isEqualToString:confirmEmail]);
52         };
53
54
55
56
57     RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
58
59
60
61
62
63
64
65
66
// 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];
```

```
34 - (void)viewDidLoad
35 {
36     [super viewDidLoad];
37
38     RACSignal *formValid = [RACSignal
39         combineLatest:@[
40             self.firstNameField.rac_textSignal,
41             self.lastNameField.rac_textSignal,
42             self.emailField.rac_textSignal,
43             self.confirmEmailField.rac_textSignal,
44         ]
45         reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confirmEmail) {
46             return @(firstName.length > 0
47                     && lastName.length > 0
48                     && email.length > 0
49                     && confirmEmail.length > 0
50                     && [email isEqualToString:confirmEmail]);
51         }];
52
53     RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
54     RACSignal *networkResults = [createAccountCommand addSignalBlock:^RACSignal *(id value)
55         // TODO: Create signal from APIClient's create account method
56         return [RACSignal empty];
57     }];
58
59
60
61     // initial button setup
62     self.enabledButtonColor = self.createButton.titleLabel.textColor;
63     self.disabledButtonColor = UIColor.lightGrayColor;
64     [self.createButton setTitleColor:self.disabledButtonColor
65         forState:UIControlStateNormal];
66     self.createButton.titleLabel.textColor = UIColor.lightGrayColor;

```

```
RACMobiDevDay.xcworkspace — SignUpViewController.m
RACMobiDevDay > RACMobiDevDay > SignUpViewController.m - viewDidLoad
34 - (void)viewDidLoad
35 {
36     [super viewDidLoad];
37
38     RACSignal *formValid = [RACSignal
39         combineLatest:@[
40             self.firstNameField.rac_textSignal,
41             self.lastNameField.rac_textSignal,
42             self.emailField.rac_textSignal,
43             self.confirmEmailField.rac_textSignal,
44         ]
45         reduce:^NSString *(NSString *firstName, NSString *lastName, NSString *email, NSString *confirmEmail) {
46             return @(
47                 firstName.length > 0
48                 && lastName.length > 0
49                 && email.length > 0
50                 && confirmPassword.length > 0
51                 && [email isEqualToString:confirmEmail]);
52         };
53
54     RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
55     RACSignal *networkResults = [[[createAccountCommand
56         addSignalBlock:^RACSignal *(id value) {
57             // TODO: Create signal from APIClient's create account method
58             return [RACSignal empty];
59         }]
60         switchToLatest]
61         deliverOn:[RACScheduler mainThreadScheduler]];
62
63
64     // initial button setup
65     self.enabledButtonColor = self.createButton.titleLabel.textColor;
66     self.disabledButtonColor = UIColor.lightGrayColor;

```

```
RACMobiDevDay.xcworkspace — SignUpViewController.m
RACMobiDevDay > RACMobiDevDay > SignUpViewController.m -viewDidLoad
    return @(firstName.length > 0
        && lastName.length > 0
        && email.length > 0
        && confirmPassword.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];
```

```
reduced:^(NSString *firstName, NSString *lastName, NSString *email, NSString *compli-  
return @(firstName.length > 0  
    && lastName.length > 0  
    && email.length > 0  
    && confirmPassword.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] subscribeNext:  
    [createAccountCommand execute:sender];  
};  
  
// initial button setup  
self.enabledButtonColor = self.createButton.titleLabel.textColor;
```

```
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] subscribeNext:^(createAccountCommand execute:sender];
}];

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

```
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] subscribeNext:^(RACSignal *buttonEnabled) {
    [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] subscribeNext:^(id value) {
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;
```

```
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] subscribeNext:^(RACSignal *buttonEnabled) {
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
```

```
57 // TODO: Create signal from selector to create account method
58     return [RACSignal empty];
59 }
60 switchToLatest]
61 deliverOn:[RACScheduler mainThreadScheduler]];
62
63 // bind create button's UI state and touch action
64 [[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNext:^(id<RACTuple> value) {
65     [createAccountCommand execute:sender];
66 }];
67
68 RACSignal *buttonEnabled = RACableWithStart(createAccountCommand, canExecute);
69 RAC(self.createButton.enabled) = buttonEnabled;
70
71 UIColor *defaultButtonTitleColor = self.createButton.titleLabel.textColor;
72 RACSignal *buttonTextColor = [buttonEnabled map:^id(NSNumber *x) {
73     return x.boolValue ? defaultButtonTitleColor : [UIColor lightGrayColor];
74 }];
75
76 [self.createButton rac_liftSelector:@selector(setTitleColor:forState:)
77                         withObject:buttonEnabled, @(UIControlStateNormal)];
```



```
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
```

```
switch (state) {
    case RACCommandExecutingState:
        deliverOn: [RACScheduler mainThreadScheduler]];
    }
}

// bind create button's UI state and touch action
[[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNext:^(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
```

```
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
```

```
switch (state) {
    case RACCommandExecuting:
        deliverOn: [RACScheduler mainThreadScheduler]];
    }
}

// bind create button's UI state and touch action
[[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNext:^(id<RACTuple> value) {
    [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             withObject: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  
69  
70    RAC(self.createButton.enabled) = buttonEnabled;  
71  
72    UIColor *defaultButtonTitleColor = self.createButton.titleLabel.textColor;  
73    RACSignal *buttonTextColor = [buttonEnabled map:^id(NSNumber *x) {  
74        return x.boolValue ? defaultButtonTitleColor : [UIColor lightGrayColor];  
75    }];  
76  
77    [self.createButton rac_liftSelector:@selector(setTitleColor:forState:)  
78     withObjects:buttonEnabled, @(UIControlStateNormal)];  
79  
80 // bind button and text field state to create account command executing state  
81  
82    RACSignal *executing = RACable(createAccountCommand, executing);  
83  
84    RACSignal *fieldTextColor = [executing map:^id(NSNumber *x) {  
85        return x.boolValue ? [UIColor lightGrayColor] : [UIColor blackColor];  
86    }];  
87  
88    RAC(self.firstNameField.textColor) = fieldTextColor;  
89    RAC(self.lastNameField.textColor) = fieldTextColor;  
90    RAC(self.emailField.textColor) = fieldTextColor;  
91    RAC(self.confirmEmailField.textColor) = fieldTextColor;
```

```
73 }];  
74  
75 [self.createButton rac_liftSelector:@selector(setTitleColor:forState:)  
    withObjects:buttonEnabled, @(UIControlStateNormal)];  
76  
77 // bind button and text field state to create account command executing state  
78  
79 RACSignal *executing = RACable(createAccountCommand, executing);  
80  
81 RACSignal *fieldTextColor = [executing map:^id(NSNumber *x) {  
82     return x.boolValue ? [UIColor lightGrayColor] : [UIColor blackColor];  
83 }];  
84  
85 RAC(self.firstNameField.textColor) = fieldTextColor;  
86 RAC(self.lastNameField.textColor) = fieldTextColor;  
87 RAC(self.emailField.textColor) = fieldTextColor;  
88 RAC(self.confirmEmailField.textColor) = fieldTextColor;  
89  
90 RACSignal *notExecuting = [executing not];  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104
```

```
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 stopAnimating];  
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 stopAnimating];
    self.statusLabel.textColor = [UIColor lightGrayColor];
}];

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

```
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 stopAnimating];
    self.statusLabel.textColor = [UIColor lightGrayColor];
}];

// Derive the status label's text and color from our network result
RAC(self.statusLabel.text) = [networkResults map:^id(id value) {
}];

!
```



```
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 stopAnimating];
    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.localizedDescription
                                              : NSLocalizedString(@"Thanks for signing up!", nil);
}];
```

```
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 stopAnimating];
    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.localizedDescription
                                              : NSLocalizedString(@"Thanks for signing up!");
};
RAC(self.statusLabel.textColor) = [networkResults map:^id(RACEvent **x) {
    return x.eventType == RACEventTypeError ? [UIColor redColor]
                                              : [UIColor greenColor];
}];


```

```
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 stopAnimating];
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.localizedDescription
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 @end
```

```
RACMobiDevDay.xcworkspace — SignUpViewController.m
RACMobiDevDay > RACMobiDevDay > SignUpViewController.m - viewDidLoad
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             withObject:buttonTextColor, @(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 [executing subscribeNext:^(NSNumber *x) {
99     x.boolValue ? [self.loadingIndicator startAnimating] : [self.loadingIndicator stopAnimating];
100    self.statusLabel.textColor = [UIColor lightGrayColor];
101 }];
102
103 // Derive the status label's text and color from our network result
/Users/andrew/RACMobiDevDay/RACMobiDevDay/SignUpViewController.m
```

```
RACMobiDevDay.xcworkspace — SignUpViewController.m
RACMobiDevDay > RACMobiDevDay > SignUpViewController.m -viewDidLoad
51 }
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] subscribeNext:^(RACCommand *createAccountCommand execute:sender);
64 ]
65
66 RACSignal *buttonEnabled = RACableWithStart(createAccountCommand, canExecute);
67 RAC(self.createButton.enabled) = buttonEnabled;
68
69 UIColor *defaultButtonTitleColor = self.createButton.titleLabel.textColor;
70 RACSignal *buttonTextColor = [buttonEnabled map:^id(NSNumber *x) {
71     return x.boolValue ? defaultButtonTitleColor : [UIColor lightGrayColor];
72 }];
73
74 [self.createButton rac_liftSelector:@selector(setTitleColor:forState:)
75                         withObject:buttonTextColor, @(UIControlStateNormal)];
76
77 // bind button and text field state to create account command executing state
78
79 RACSignal *executing = RACable(createAccountCommand, executing);
80
81 RACSignal *fieldTextColor = [executing map:^id(NSNumber *x) {
82     return x.boolValue ? [UIColor lightGrayColor] : [UIColor blackColor];
83 }];
84
85 /Users/andrew/RACMobiDevDay/RACMobiDevDay/SignUpViewController.m
```

```
1 #import <AFNetworking/AFNetworking.h>
2
3 @interface APIClient : AFHTTPClient
4
5 + (instancetype)sharedClient;
6
7 - (void)createAccountForEmail:(NSString *)email
8     firstName:(NSString *)firstName
9     lastName:(NSString *)lastName
10    success:(void (^)(id account))successBlock
11   failure:(void (^)(NSError *error))failureBlock;
12
13 @end
14
```

```
1 #import <AFNetworking/AFNetworking.h>
2
3 @class RACSignal;
4
5 @interface APIClient : AFHTTPClient
6
7 + (instancetype)sharedClient;
8
9 - (RACSignal *)createAccountForEmail:(NSString *)email
10    firstName:(NSString *)firstName
11    lastName:(NSString *)lastName
12    success:(void (^)(id account))successBlock
13    failure:(void (^)(NSError *error))failureBlock;
14
15 @end
16
```

```
1 #import <AFNetworking/AFNetworking.h>
2
3 @class RACSignal;
4
5 @interface APIClient : AFHTTPClient
6
7 + (instancetype)sharedClient;
8
9 - (RACSignal *)createAccountForEmail:(NSString *)email
10                           firstName:(NSString *)firstName
11                           lastName:(NSString *)lastName;
12
13 @end
14
```

```
13     apiClient = [APIClient new];
14 }
15 return apiClient;
16 }
17
18 - (id)init
19 {
20     self = [super initWithBaseURL:[NSURL URLWithString:APIClientDefaultEndpoint]];
21     if (self) {
22         AFNetworkActivityIndicatorManager.sharedManager.enabled = YES;
23         [self registerHTTPOperationClass:AFJSONRequestOperation.class];
24         [self setDefaultHeader:@"Accept" value:@"application/json"];
25     }
26     return self;
27 }
28
29 - (void)createAccountForEmail:(NSString *)email
30                         firstName:(NSString *)firstName
31                         lastName:(NSString *)lastName
32                         success:(void (^)(id account))successBlock
33                         failure:(void (^)(NSError *error))failureBlock
34 {
35     [self postPath:@"/accounts"
36      parameters:@{ @"first_name": firstName, @"last_name": lastName, @"email": email, }
37      success:^(AFHTTPRequestOperation *operation, id responseObject) {
38         successBlock(responseObject);
39     }
40      failure:^(AFHTTPRequestOperation *operation, NSError *error) {
41         id responseJSON = nil;
42         if ([operation respondsToSelector:@selector(responseJSON)])
43             responseJSON = [(id)operation responseJSON];
44     }
45 }
```

```
28
29 - (RACSignal *)createAccountForEmail:(NSString *)email
30             firstName:(NSString *)firstName
31             lastName:(NSString *)lastName
32 {
33     [self postPath:@"/accounts"
34      parameters:@{ @"first_name": firstName, @"last_name": lastName, @"email": email, }
35      success:^(AFHTTPRequestOperation *operation, id responseObject) {
36         successBlock(responseObject);
37     }
38     failure:^(AFHTTPRequestOperation *operation, NSError *error) {
39         id responseJSON = nil;
40         if ([operation respondsToSelector:@selector(responseJSON)]) {
41             responseJSON = [(id)operation responseJSON];
42         }
43         failureBlock([NSError errorWithDomain:@"com.example"
44                         code:error.code
45                         userInfo:@{
46                             NSLocalizedFailureReasonErrorKey: [response
47                                         ?: @"Fail
48                                         }];
49     }];
50 }
51
52 @end
53
```

```
22 AFNetworkActivityIndicatorManager.sharedManager.enabled = YES;
23 [self registerHTTPOperationClass:AFJSONRequestOperation.class];
24 [self setDefaultHeader:@"Accept" value:@"application/json"];
25 }
26 return self;
27 }

28 - (RACSignal *)createAccountForEmail:(NSString *)email
29             firstName:(NSString *)firstName
30             lastName:(NSString *)lastName
31 {
32     RACSubject *subject = [RACSubject subject];
33     [self postPath:@"/accounts"
34         parameters:@{ @"first_name": firstName, @"last_name": lastName, @"email": email, }
35         success:^(AFHTTPRequestOperation *operation, id responseObject) {
36             successBlock(responseObject);
37         }
38         failure:^(AFHTTPRequestOperation *operation, NSError *error) {
39             id responseJSON = nil;
40             if ([operation respondsToSelector:@selector(responseJSON)])
41                 responseJSON = [(id)operation responseJSON];
42         }
43         failureBlock([NSError errorWithDomain:@"com.example"
44                         code:error.code
45                         userInfo:@{
46                             NSLocalizedFailureReasonErrorKey: [responseJSON
47                                         ?: @"Failure"]
48                         }]);
49     ];
50 }
51 return subject;
52 }

53 @end
```

```
22 AFNetworkActivityIndicatorManager.sharedManager.enabled = YES;
23 [self registerHTTPOperationClass:AFJSONRequestOperation.class];
24 [self setDefaultHeader:@"Accept" value:@"application/json"];
25 }
26 return self;
27 }

28 - (RACSignal *)createAccountForEmail:(NSString *)email
29             firstName:(NSString *)firstName
30             lastName:(NSString *)lastName
31 {
32     RACSubject *subject = [RACSubject subject];
33
34     [self postPath:@"/accounts"
35      parameters:@{ @"first_name": firstName, @"last_name": lastName, @"email": email, }
36      success:^(AFHTTPRequestOperation *operation, id responseObject) {
37         [subject sendNext:responseObject];
38         [subject sendCompleted];
39     }
40     failure:^(AFHTTPRequestOperation *operation, NSError *error) {
41         id responseJSON = nil;
42         if ([operation respondsToSelector:@selector(responseJSON)])
43             responseJSON = [(id)operation responseJSON];
44     }
45     failureBlock([NSError errorWithDomain:@"com.example"
46                  code:error.code
47                  userInfo:@{
48                      NSLocalizedFailureReasonErrorKey: [responseJSON objectForKey:@"error"]
49                  }];
50
51 ];
52
53 return subject;
54 }
```

```
22 AFNetworkActivityIndicatorManager.sharedManager.enabled = YES;
23 [self registerHTTPOperationClass:AFJSONRequestOperation.class];
24 [self setDefaultHeader:@"Accept" value:@"application/json"];
25 }
26 return self;
27 }

28 - (RACSignal *)createAccountForEmail:(NSString *)email
29                               firstName:(NSString *)firstName
30                               lastName:(NSString *)lastName
31 {
32     RACSubject *subject = [RACSubject subject];
33
34     [self postPath:@"/accounts"
35      parameters:@{ @"first_name": firstName, @"last_name": lastName, @"email": email, }
36      success:^(AFHTTPRequestOperation *operation, id responseObject) {
37         [subject sendNext:responseObject];
38         [subject sendCompleted];
39     }
40     failure:^(AFHTTPRequestOperation *operation, NSError *error) {
41         id responseJSON = nil;
42         if ([operation respondsToSelector:@selector(responseJSON)])
43             responseJSON = [(id)operation responseJSON];
44     }
45     [subject sendError:[NSError errorWithDomain:@"com.example"
46                               code:error.code
47                               userInfo:@{
48                                   NSLocalizedFailureReasonErrorKey: [responseJSON
49                                         ?: @"Failure"]
50                               }]];
51 ];
52 }

53 return subject;
54
```

```
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
```

```
        self.firstNameField.rac_textSignal,
        self.lastNameField.rac_textSignal,
        self.emailField.rac_textSignal,
        self.confirmEmailField.rac_textSignal,
    ]
reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confirmEmail)
    return @(firstName.length > 0
            && lastName.length > 0
            && email.length > 0
            && confirmPassword.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] subscribeNext:^(id value) {
    [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 {
    if (x) {
        return [UIColor colorWithRed:(x.floatValue / 255.0) green:(x.floatValue / 255.0) blue:(x.floatValue / 255.0) alpha:1.0];
    } else {
        return defaultButtonTitleColor;
    }
}];
```

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

Command *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];
Signal *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] subscribeNext:^(createAccountCommand execute:sender);

Signal *buttonEnabled = RACableWithStart(createAccountCommand, canExecute);
(self.createButton.enabled) = buttonEnabled;

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

```
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
```

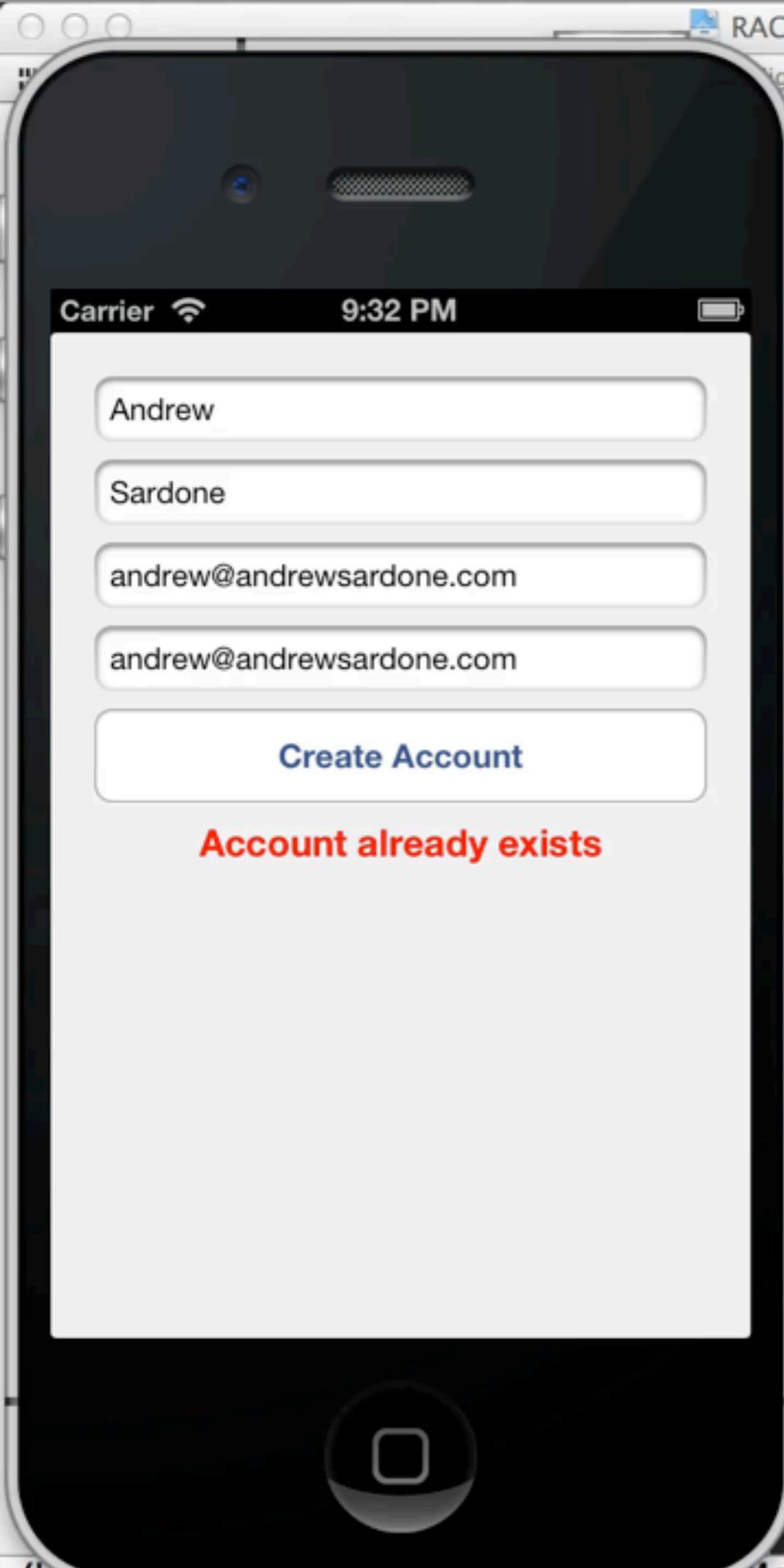
```
        self.firstNameField.rac_textSignal,
        self.lastNameField.rac_textSignal,
        self.emailField.rac_textSignal,
        self.confirmEmailField.rac_textSignal,
    ]
reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confirmEmail)
{
    return @(
        firstName.length > 0
        && lastName.length > 0
        && email.length > 0
        && confirmPassword.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] subscribeNext:^(id value) {
    [createAccountCommand execute:sender];
}];

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

```
35 {  
36     [super viewDidLoad];  
37  
38     RACSignal *formValid = [RACSignal  
39         combineLatest:@[  
40             self.firstNameField.rac_textSignal,  
41             self.lastNameField.rac_textSignal,  
42             self.emailField.rac_textSignal,  
43             self.confirmEmailField.rac_textSignal,  
44         ]  
45         reduce:^(NSString *firstName, NSString *lastName, NSString *email, NSString *confirmEmail)  
46             return @(firstName.length > 0  
47                     && lastName.length > 0  
48                     && email.length > 0  
49                     && confirmEmail.length > 0  
50                     && [email isEqualToString:confirmEmail]);  
51     };  
52  
53     RACCommand *createAccountCommand = [RACCommand commandWithCanExecuteSignal:formValid];  
54     RACSignal *networkResults = [[[createAccountCommand  
55         addSignalBlock:^RACSignal *(id value) {  
56             return [[APIClient.sharedClient createAccountForEmail:self.emailField.text  
57                         firstName:self.firstNameField.text  
58                         lastName:self.lastNameField.text]  
59                         materialize];  
60         }]  
61         switchToLatest]  
62         deliverOn:[RACScheduler mainThreadScheduler]]];  
63  
64     // bind create button's UI state and touch action  
65     [[self.createButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNext:  
66         [createAccountCommand execute:sender];  
67     ]  
68 }
```



CSignal

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

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

```
mainThreadScheduler]];
```

state and touch action

`SignalForControlEvents:UIControlEventTouchUpInside] subscribeNext`

objDevDay/SignUpViewController.m

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](https://twitter.com/andrewa2)

app.net: [@andrewsardone](https://app.net/@andrewsardone)

github.com/[andrewsardone](https://github.com/andrewsardone)