ReactiveCocoa and Swift Better Together

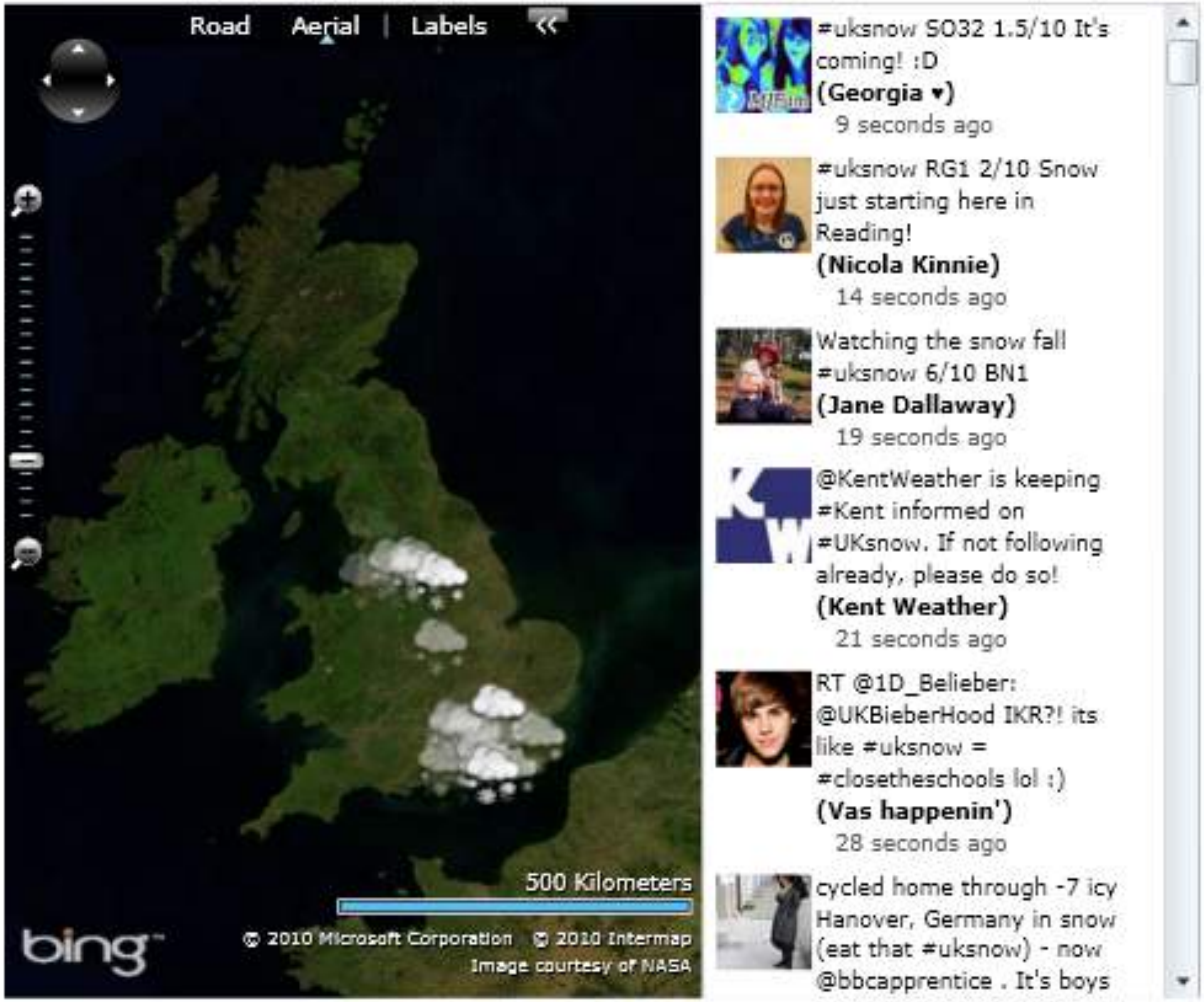
@ColinEberhardt
Technology Director, Scott Logic Ltd


NEWCASTLE
BROWN ALE

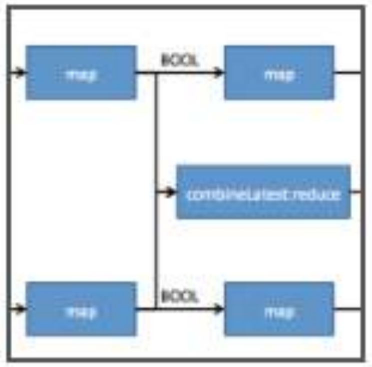
ReactiveCocoa and Swift Better Together

@ColinEberhardt
Technology Director, Scott Logic Ltd

#uksnow



Road
Aerial
Labels
500 Kilometers
bing
© 2010 Microsoft Corporation © 2010 Intermap
Image courtesy of NASA
#uksnow SO32 1.5/10 It's coming! :D
(Georgia ♥)
9 seconds ago
#uksnow RG1 2/10 Snow just starting here in Reading!
(Nicola Kinnie)
14 seconds ago
Watching the snow fall #uksnow 6/10 BN1
(Jane Dallaway)
19 seconds ago
@KentWeather is keeping #Kent informed on #UKsnow. If not following already, please do so!
(Kent Weather)
21 seconds ago
RT @1D_Belieber: @UKBieberHood IKR?! its like #uksnow = #closetheschools lol :)
(Vas happenin')
28 seconds ago
cycled home through -7 icy Hanover, Germany in snow (eat that #uksnow) - now @bbcapprentice . It's boys

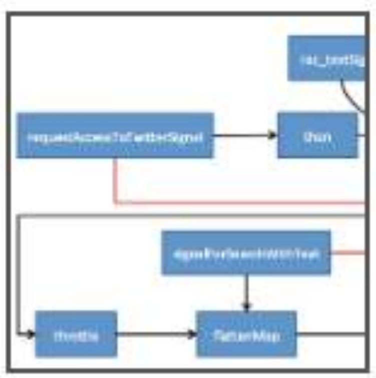


ReactiveCocoa Tutorial – The Definitive Introduction: Part 1/2

Get to grips with ReactiveCocoa in this 2-part tutorial series. Put the paradigms to one-side, and understand the practical value with work-through examples.

Colin Eberhardt on March 10, 2014

iOS Design Skill:



ReactiveCocoa Tutorial – The Definitive Introduction: Part 2/2

Get to grips with ReactiveCocoa in this 2-part tutorial series. Put the paradigms to one-side, and understand the practical value with work-through examples

Colin Eberhardt on March 11, 2014

iOS Design Skill:

[[[[[[[[[[[[

f*****gblocksyntax.com

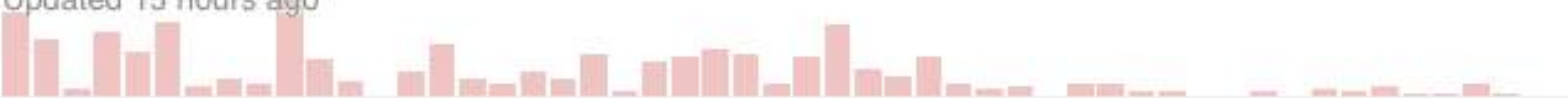
ReactiveCocoa Core Concepts

ReactiveCocoa

Objective-C ★ 5,755 744

A framework for composing and transforming streams of values

Updated 15 hours ago



Functional Programming

+

Reactive Programming

=

Functional **R**eactive **P**rogramming

ReactiveCocoa
In my own words

Every line of code we write
is executed in reaction to
an event

But … these events come in many different forms

ReactiveCocoa provides a common interface for all events

… this allows us to define a language for manipulating, transforming and coordinating events

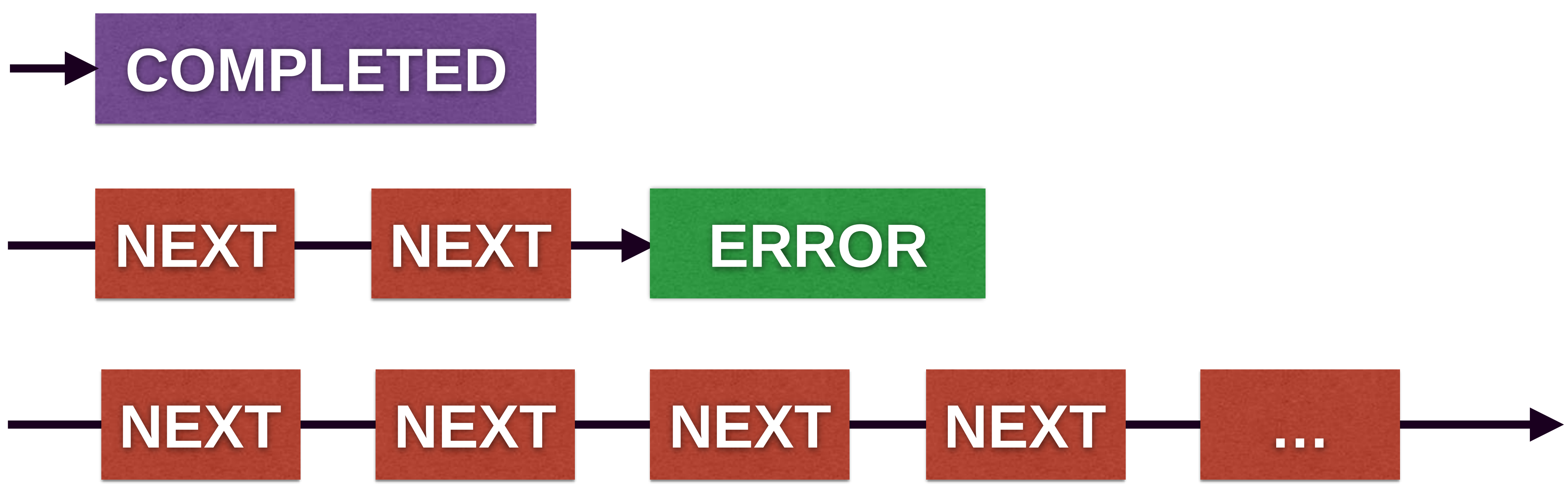
ReactiveCocoa in action

```
RACSignal *textSignal = [self.usernameTextField rac_textSignal];

[textSignal subscribeNext:^(id x) {
  NSLog(x);
}];
```


iOS Simulator - iPhone 4s - iPhone 4s / iOS 8....
Carrier
8:37 AM

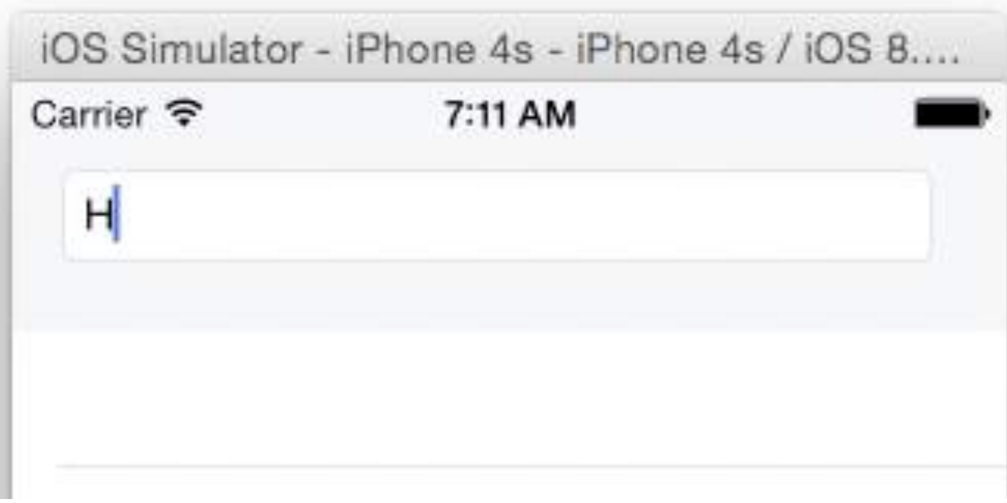
Signals emit events
to their subscribers

… they emit none, one or more **next** events, optionally followed by either an **error** or **completed**



Signal All Things!

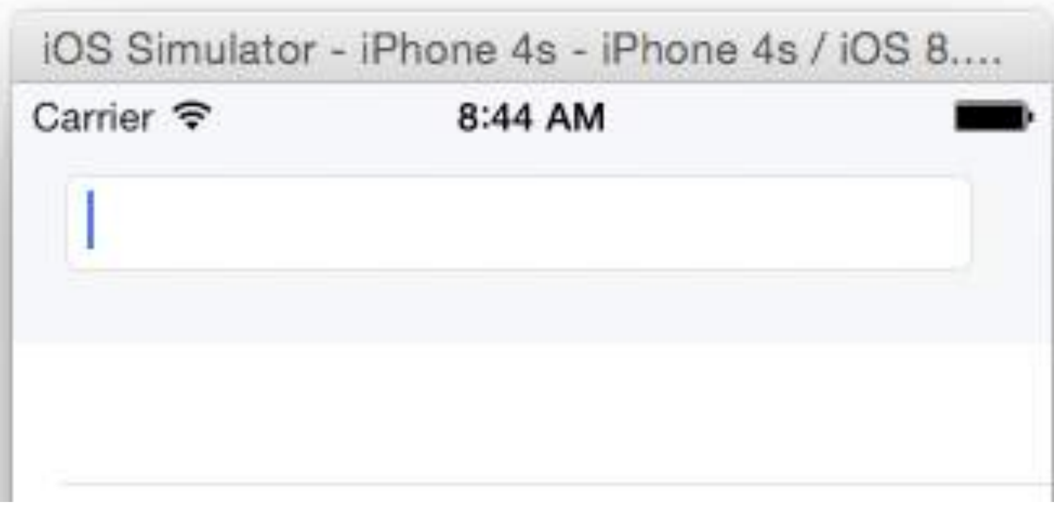
```
RACSignal *textSignal = [self.usernameTextField rac_textSignal];

RACSignal *filteredText = [textSignal filter:^BOOL(NSString *text) {
  return text.length > 3;
}];

[filteredText subscribeNext:^(id x) {
  NSLog(x);
}];
```


iOS Simulator - iPhone 4s - iPhone 4s / iOS 8....
Carrier
7:11 AM
H

What are events? and
what do they look like?

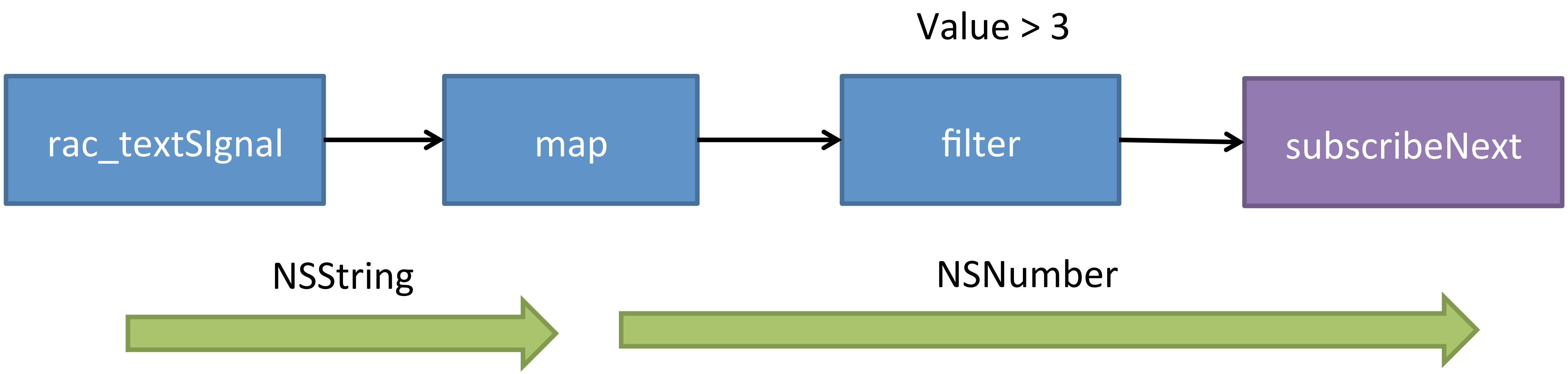
anything!

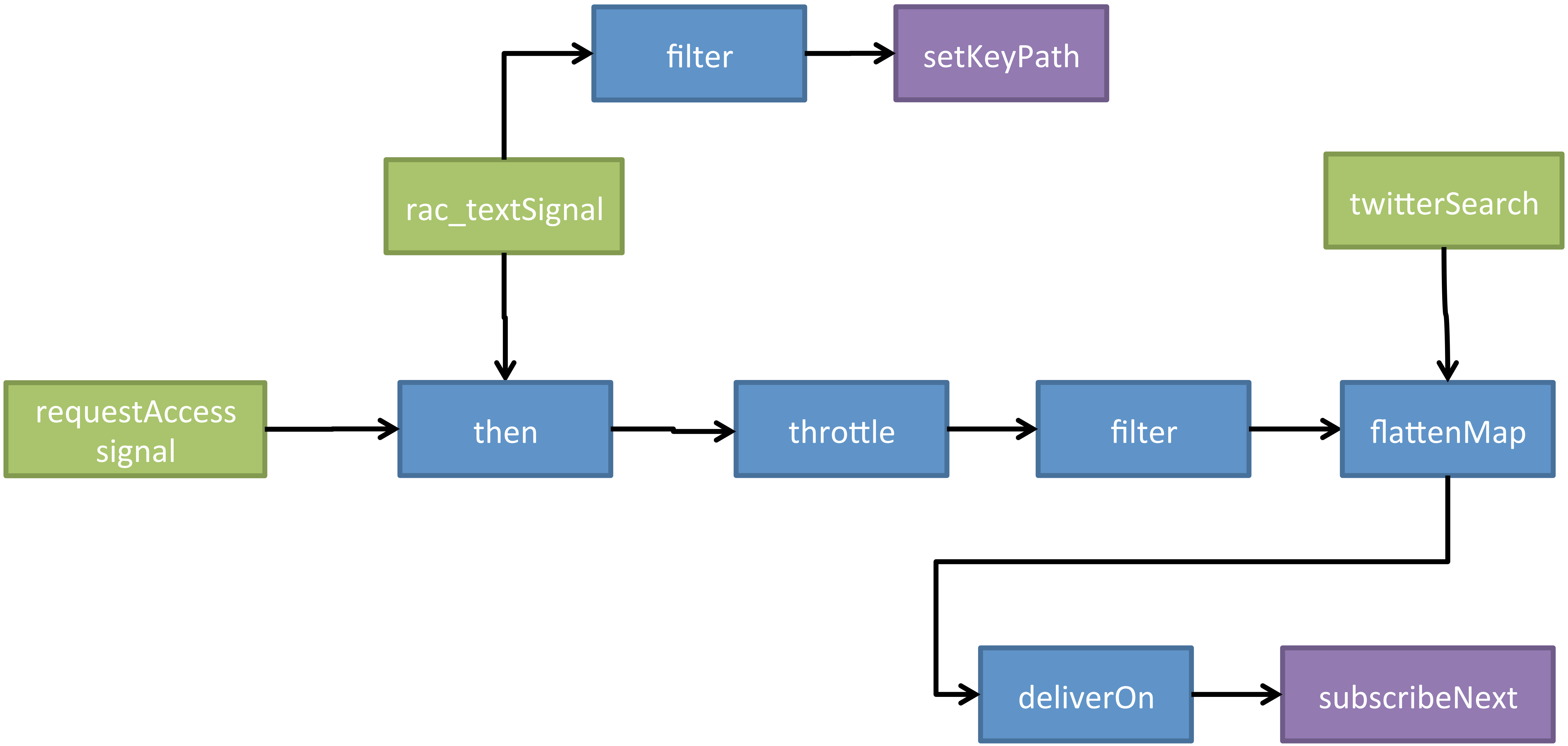
```
RACSignal *textSignal = [self.usernameTextField rac_textSignal];

RACSignal *textLength = [textSignal map:^id(NSString *text) {
  return @(text.length);
}];

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


iOS Simulator - iPhone 4s - iPhone 4s / iOS 8....
Carrier
8:44 AM

```
RACSignal *textSignal = [self.usernameTextField rac_textSignal];

RACSignal *textLength = [textSignal map:^id(NSString *text) {
  return @(text.length);
}];

RACSignal *filteredText = [textLength filter:^BOOL(NSNumber *length) {
  return [length intValue] > 3;
}];

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

```objc
[[[[self.usernameTextField rac_textSignal]
  map:^id(NSString *text) {
    return @(text.length);
  }]
  filter:^BOOL(NSNumber *length) {
    return [length intValue] > 3;
  }]
  subscribeNext:^(id x) {
   NSLog(@"%@", x);
  }];
```


Value > 3
rac_textSIgnal
map
filter
subscribeNext
NSString
NSNumber


filter
setKeyPath
rac_textSignal
twitterSearch
requestAccess signal
then
throttle
filter
flattenMap
deliverOn
subscribeNext

```objc
[[[[[[[self requestAccessToTwitterSignal]
  then:^RACSignal *{
    @strongify(self)
    return self.searchText.rac_textSignal;
  }]
  filter:^BOOL(NSString *text) {
    @strongify(self)
    return [self isValidSearchText:text];
  }]
  throttle:0.5]
  flattenMap:^RACStream *(NSString *text) {
    @strongify(self)
    return [self signalForSearchWithText:text];
  }]
  map:^id(NSDictionary *jsonSearchResult) {
    NSArray *statuses = jsonSearchResult[@"statuses"];
    NSArray *tweets = [statuses linq_select:^id(id tweet) {
      return [RWTweet tweetWithStatus:tweet];
    }];
    return tweets;
  }]
  deliverOn:[RACScheduler mainThreadScheduler]]
  subscribeNext:^(NSArray *tweets) {
    [self.resultsViewController displayTweets:tweets];
  } error:^(NSError *error) {
    NSLog(@"An error occurred: %@", error);
  }];
```

stackoverflow Questions Tags Users Badges Unanswered

Fluent interface pattern in Objective-C

8

2

I am a newbie in Objective-c and I would like to implement fluent interface pattern in my OC class. Here is my updated and simplified case from my project:

```
 ...
}
- (MyLogger*) indent:(BOOL)indent;
- (MyLogger*) debug:(NSString*)message, ...;
- (id) warning:(NSString*)message, ...;
```

18 `[[[[[[[[[[[[[[[[I dont] think] fluent] interface] patterns] will] be] useful] unless] you] want] to] write] methods] like] this];` – KennyTM Jun 26 '10 at 15:53

ReactiveCocoa

Swift

```objc
RACSignal *textSignal = [self.usernameTextField rac_textSignal];

[textSignal subscribeNext:^(id x) {
  NSLog(x);
}];
```

```swift
let textSignal = self.usernameTextField.rac_textSignal()

textSignal.subscribeNext { (text: AnyObject!) -> Void in
  let textString = text as String
  println(textString)
}
```

Objective-C Friction

```
let textSignal: RACSignal =
   usernameTextField.rac_textSignal()

textSignal.subscribeNext {
  (text: AnyObject!) -> Void in
  let textString = text as String
  println(textString)
}
```

Simplified with Swift

```swift
func subscribeNextAs<T>(nextClosure:(T) -> ()) -> () {
  self.subscribeNext {
    (next: AnyObject!) -> () in
    let nextAsT = next! as T
    nextClosure(nextAsT)
  }
}
```

```swift
textSignal.subscribeNext {
  (text: AnyObject!) -> Void in
  let textString = text as String
  println(textString)
}
```

```swift
textSignal.subscribeNextAs {
  (text: String) -> () in
  println(text)
}
```

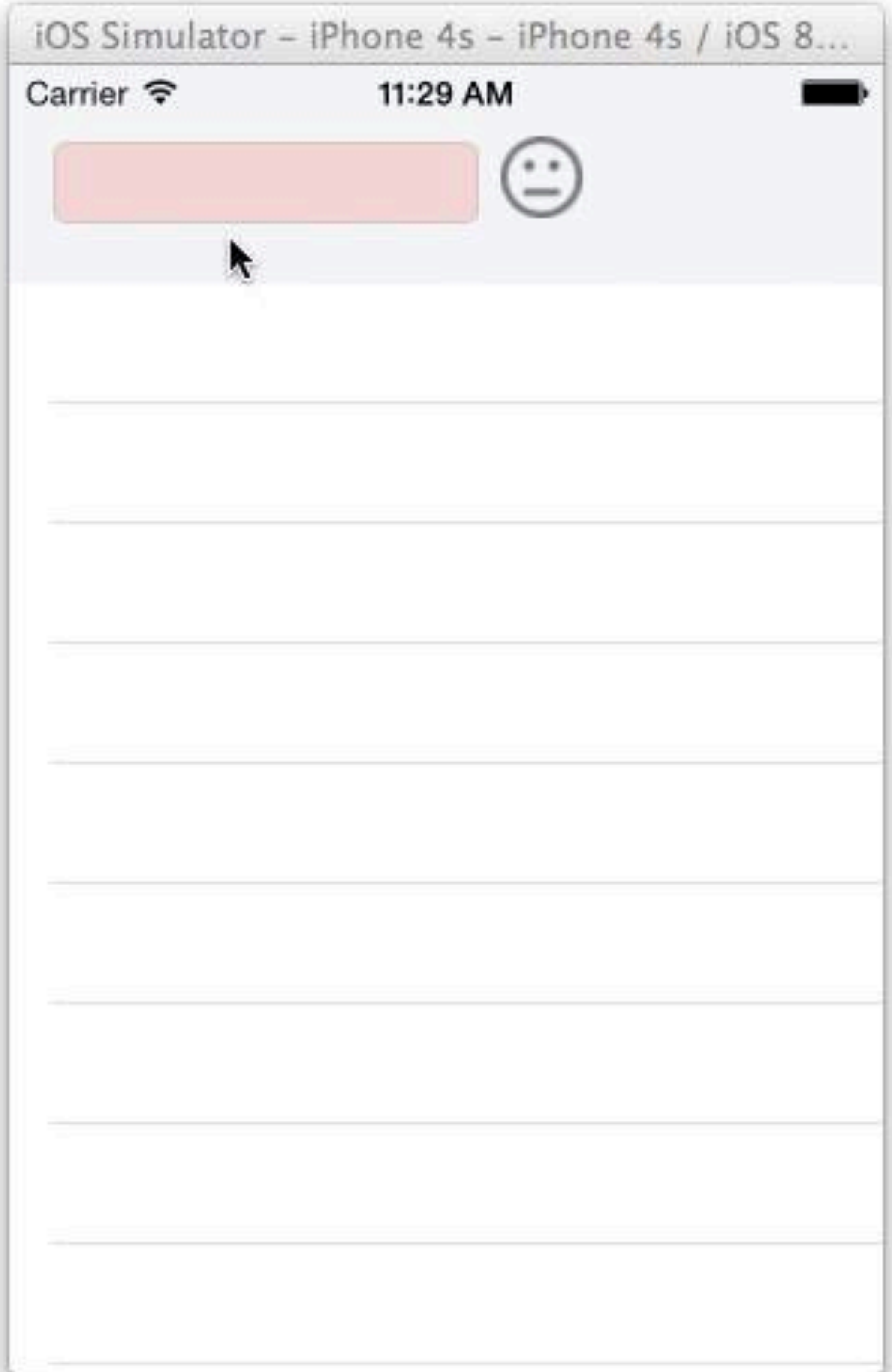
```objc
[[[[[[[self requestAccessToTwitterSignal]
  then:^RACSignal *{
    @strongify(self)
    return self.searchText.rac_textSignal;
  }]
  filter:^BOOL(NSString *text) {
    @strongify(self)
    return [self isValidSearchText:text];
  }]
  throttle:0.5]
  flattenMap:^RACStream *(NSString *text) {
    @strongify(self)
    return [self signalForSearchWithText:text];
  }]
  map:^id(NSDictionary *jsonSearchResult) {
    NSArray *statuses = jsonSearchResult[@"statuses"];
    NSArray *tweets = [statuses linq_select:^id(id tweet) {
      return [RWTweet tweetWithStatus:tweet];
    }];
    return tweets;
  }]
  deliverOn:[RACScheduler mainThreadScheduler]]
  subscribeNext:^(NSArray *tweets) {
    [self.resultsViewController displayTweets:tweets];
  } error:^(NSError *error) {
    NSLog(@"An error occurred: %@", error);
  }];
```

[[[[[[[[[[[[

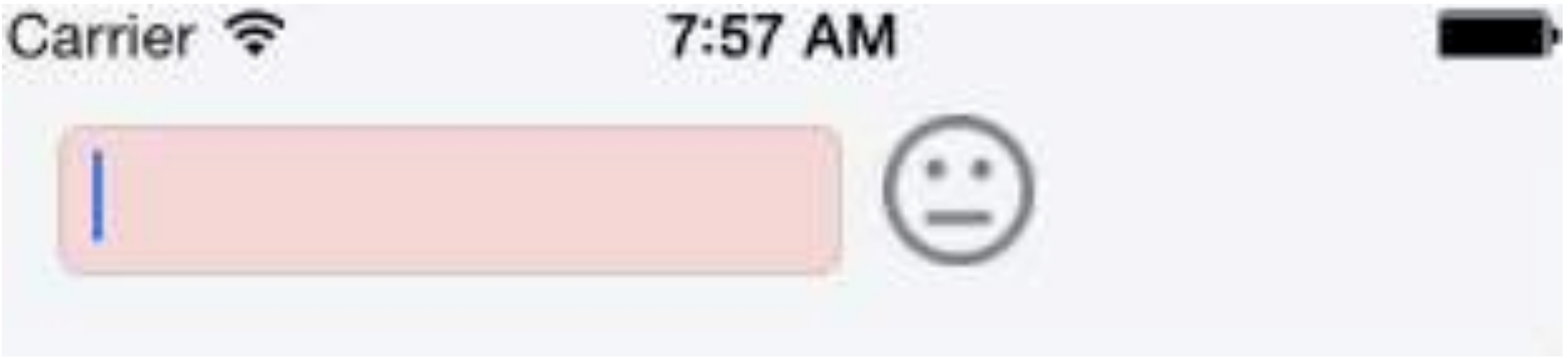
```swift
requestAccessToTwitterSignal()
  .then {
    self.searchTextField.rac_textSignal()
  }
  .filterAs {
    (text: NSString) -> Bool in
    text.length > 3
  }
  .doNext {
    (any) in
    self.tweetsTableView.alpha = 0.5
  }
  .throttle(0.5)
  .flattenMapAs {
    (text: NSString) -> RACStream in
    self.signalForSearchWithText(text)
  }
  .deliverOn(RACScheduler.mainThreadScheduler())
  .subscribeNextAs ({
    (tweets: NSDictionary) in
    let statuses = tweets["statuses"] as [NSDictionary]
    self.tweets = statuses.map { Tweet(json: $0) }
    self.tweetsTableView.reloadData()
    self.tweetsTableView.scrollToTop()
    self.tweetsTableView.alpha = 1.0
  }, {
    (error) in
    println(error)
  })
```

```
requestAccessToTwitterSignal()
  .then {
    self.searchTextField.rac_textSignal()
  }
  .filterAs {
    (text: NSString) -> Bool in
    text.length > 3
  }
  .doNext {
    (any) in
    self.tweetsTableView.alpha = 0.5
  }
  .throttle(0.5)
  .flattenMapAs {
    (text: NSString) -> RACStream in
    self.signalForSearchWithText(text)
  }
  .deliverOn(RACScheduler.mainThreadScheduler())
  .subscribeNextAs ({
    (tweets: NSDictionary) in
    let statuses = tweets["statuses"] as [NSDictionary]
    self.tweets = statuses.map { Tweet(json: $0) }
    self.tweetsTableView.reloadData()
    self.tweetsTableView.scrollToTop()
    self.tweetsTableView.alpha = 1.0
  }, {
    (error) in
    println(error)
  })
```

TweetSentiment


iOS Simulator – iPhone 4s – iPhone 4s / iOS 8...
Carrier
11:29 AM

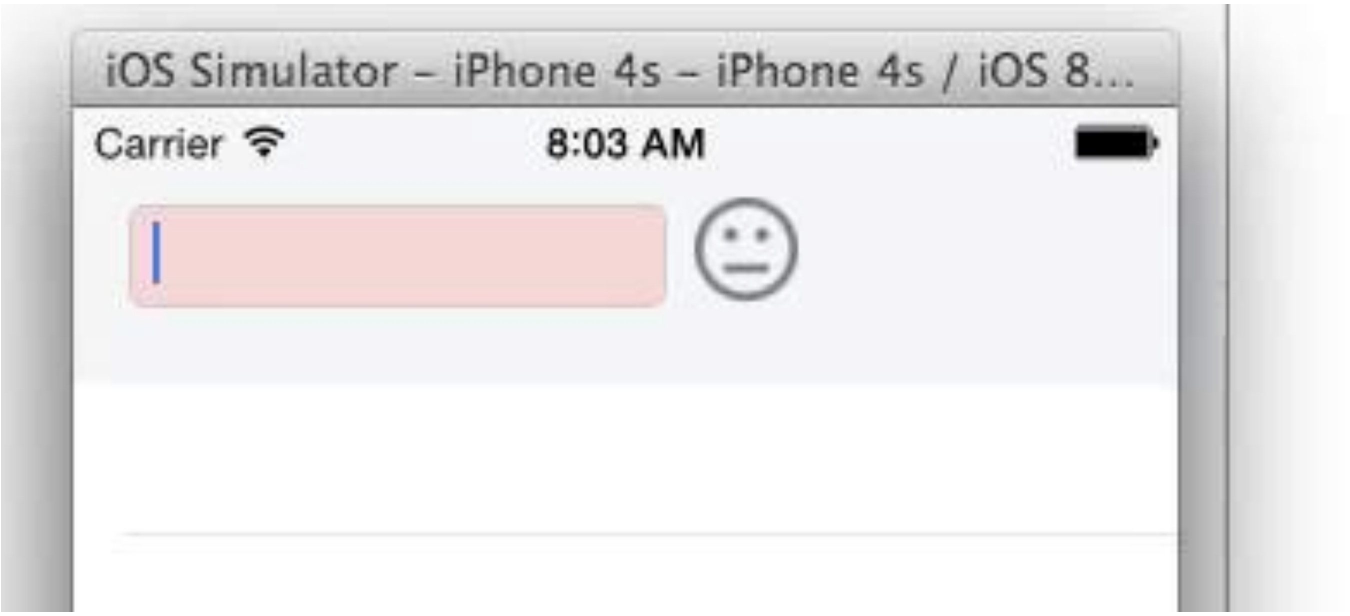
```
searchTextField
  .rac_textSignal()
  .mapAs {
    (text: NSString) -> UIColor in
    text.length <= 3
      ? UIColor.lightRedColor()
      : UIColor.whiteColor()
  }
  .setKeyPath("backgroundColor", onObject: searchTextField)
```

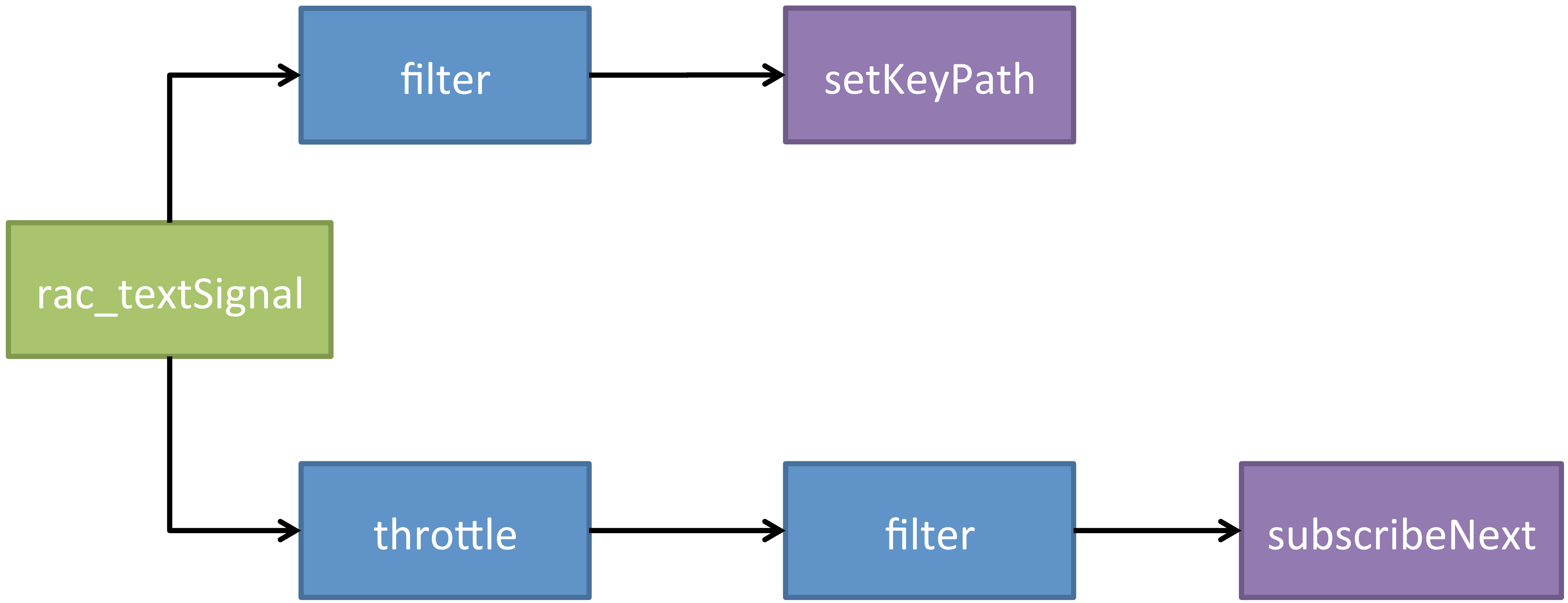


Don’t search for very short search terms

Don’t issue a new query on each key press

```
searchTextField
  .rac_textSignal()
  .throttle(0.5)
  .filterAs {
    (text: NSString) -> Bool in
    text.length > 3
  }
  .subscribeNextAs {
    (text: NSString) -> () in
    println(text)
  }
```



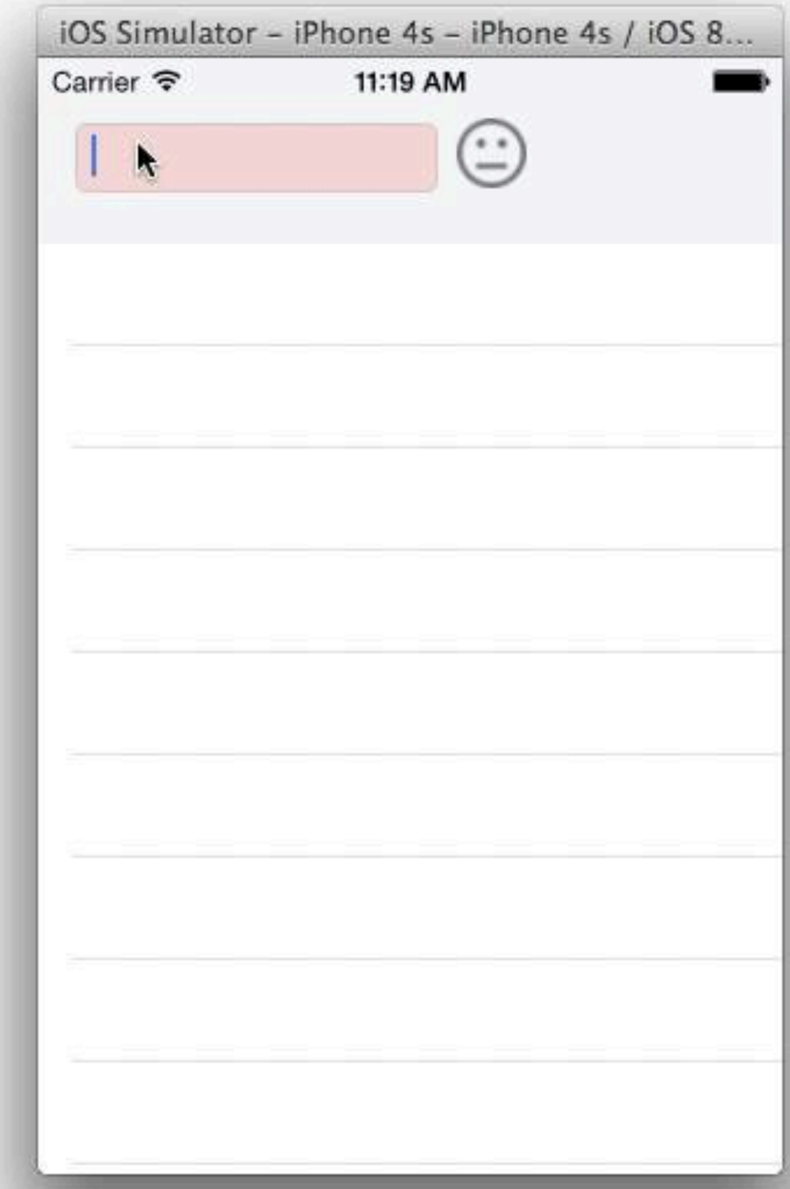

filter
setKeyPath
rac_textSignal
throttle
filter
subscribeNext

Creating Signals

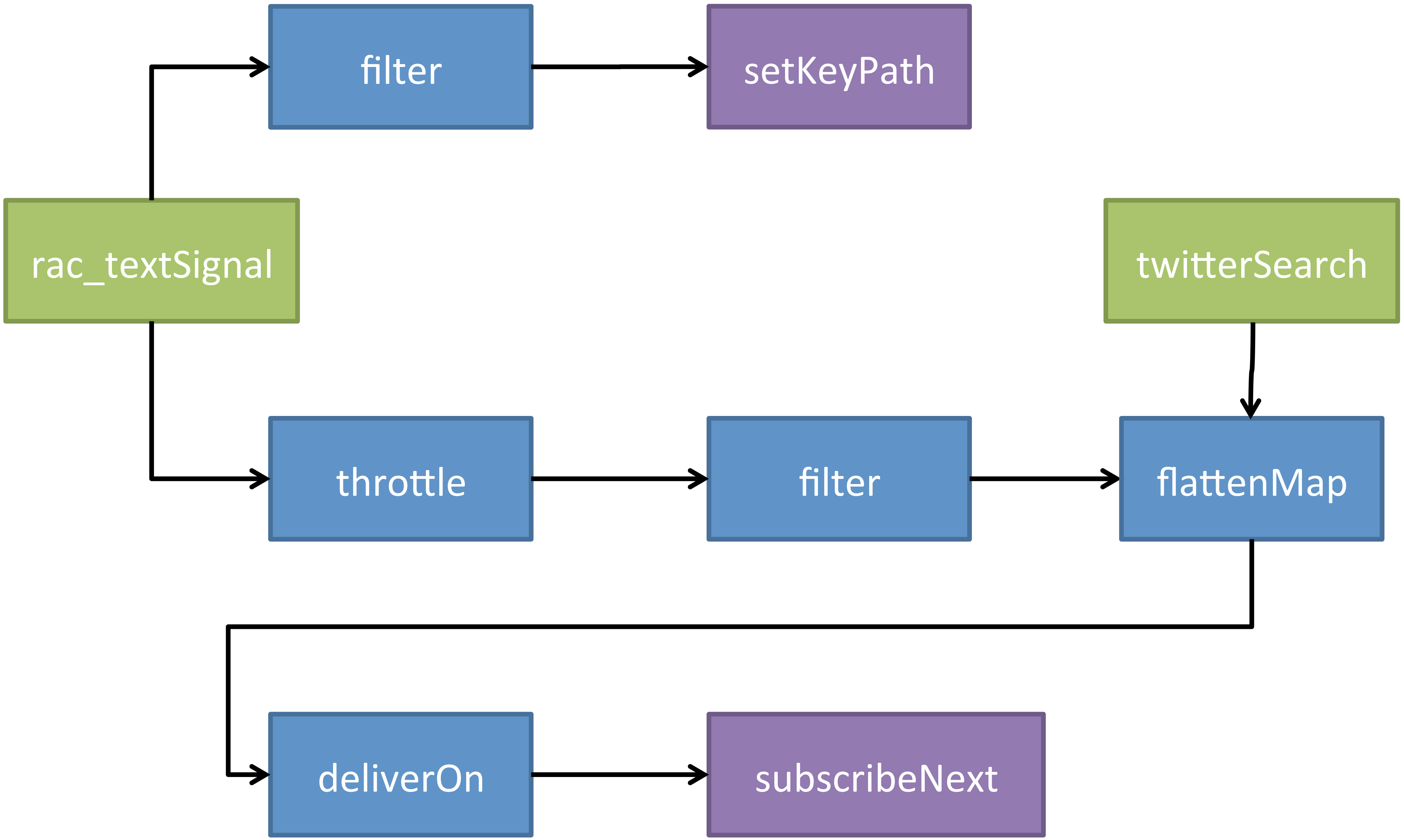
```swift
private func signalForSearchWithText(text: String) -> RACSignal {

  func requestforSearchText(text: String) -> SLRequest {
    return ...
  }

  return RACSignal.createSignal {
    subscriber -> RACDisposable! in

    let request = requestforSearchText(text)
    let maybeTwitterAccount = self.getTwitterAccount()

    if let twitterAccount = maybeTwitterAccount {
      request.account = twitterAccount
      request.performRequestWithHandler {
        (data, response, _) -> Void in
        if response != nil && response.statusCode == 200 {
          let timelineData = NSJSONSerialization.parseJSONToDictionary(data)
          subscriber.sendNext(timelineData)
          subscriber.sendCompleted()
        } else {
          subscriber.sendError(TwitterInstantError.InvalidResponse.toError())
        }
      }
    } else {
      subscriber.sendError(TwitterInstantError.NoTwitterAccounts.toError())
    }

    return nil
  }
}
```

```swift
searchTextField
  .rac_textSignal()
  .throttle(0.5)
  .filterAs {
    (text: NSString) -> Bool in
    text.length > 3
  }
  .mapAs {
    (text: NSString) -> RACStream in
    self.signalForSearchWithText(text)
  }
  .subscribeNextAs {
    ...
  }
```

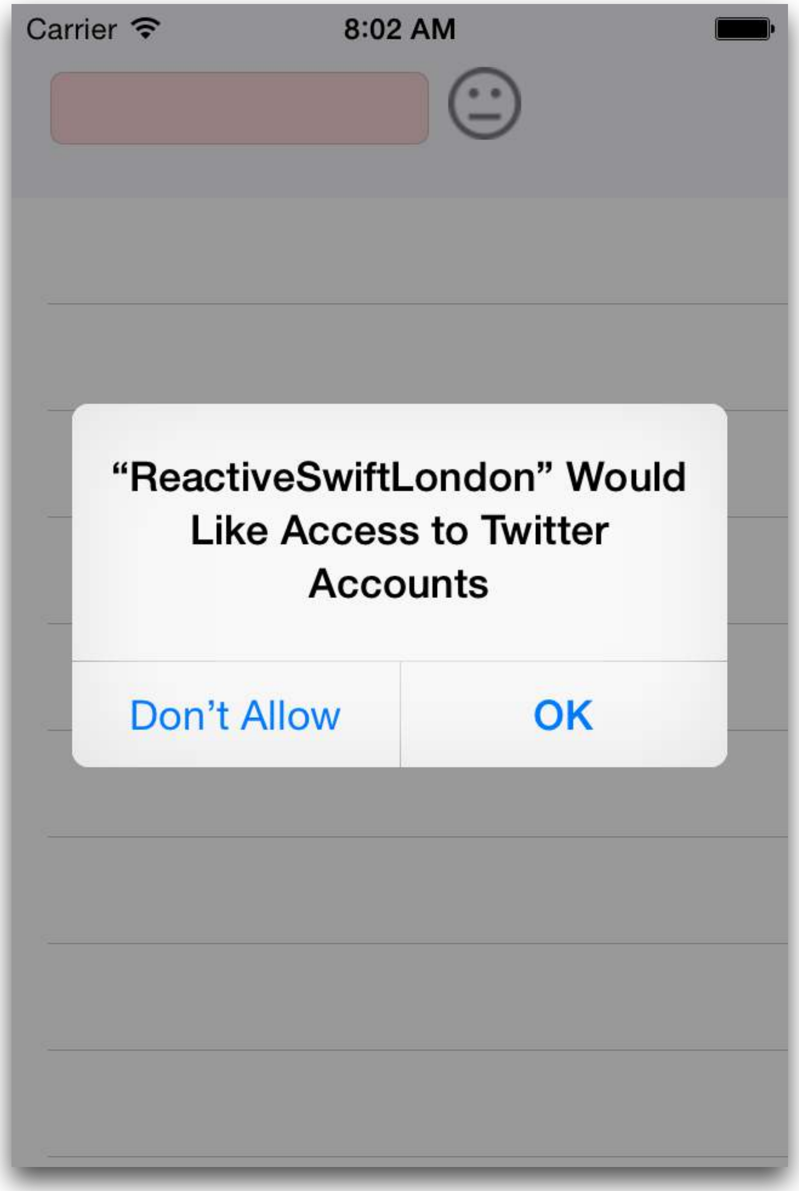
```swift
searchTextField
  .rac_textSignal()
  .throttle(0.5)
  .filterAs {
    (text: NSString) -> Bool in
    text.length > 3
  }
  .flattenMapAs {
    (text: NSString) -> RACStream in
    self.signalForSearchWithText(text)
  }
  .subscribeNextAs {
    ...
  }
```



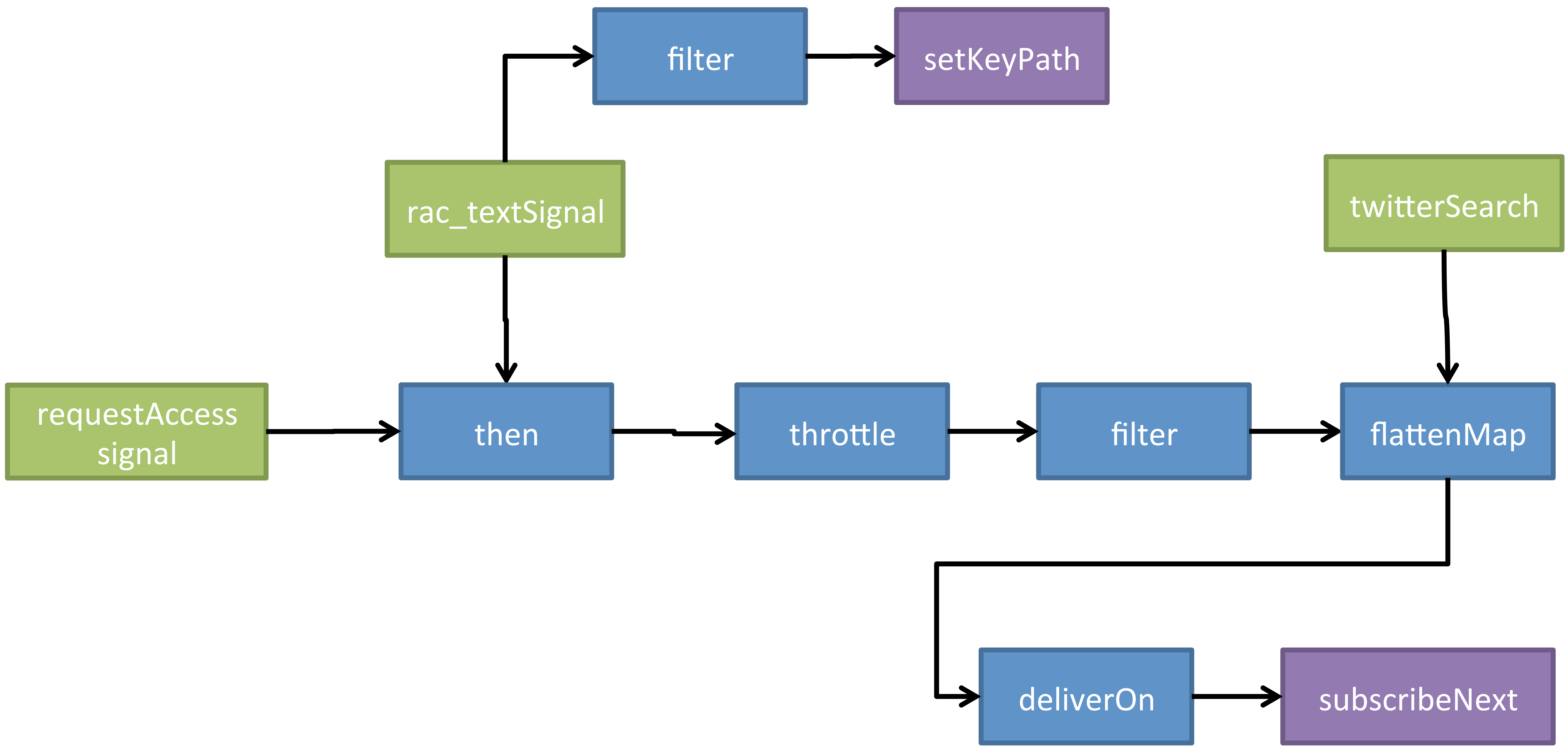

```
searchTextField
  .rac_textSignal()
  .throttle(0.5)
  .filterAs {
    (text: NSString) -> Bool in
    text.length > 3
  }
  .flattenMapAs {
    (text: NSString) -> RACStream in
    self.signalForSearchWithText(text)
  }
  .deliverOn(RACScheduler.mainThreadScheduler())
  .subscribeNextAs {
    (tweets: NSDictionary) in
    let statuses = tweets["statuses"] as [NSDictionary]
    self.tweets = statuses.map { Tweet(json: $0) }
    self.tweetsTableView.reloadData()
    self.tweetsTableView.scrollToTop()
  }
```



filter
setKeyPath
rac_textSignal
twitterSearch
throttle
filter
flattenMap
deliverOn
subscribeNext




```
requestAccessToTwitterSignal()
  .then {
    self.searchTextField.rac_textSignal()
  }
  .filterAs {
    ...
  }
  .throttle(0.5)
  ...
```


filter
setKeyPath
rac_textSignal
twitterSearch
requestAccess
signal
then
throttle
filter
flattenMap
deliverOn
subscribeNext

Error Handling

```
requestAccessToTwitterSignal()
  .then {
    self.searchTextField.rac_textSignal()
  }
  .filterAs {
    ...
  }
  .throttle(0.5)
  .doNext {
    ...
  }
  .flattenMapAs {
    ...
    self.signalForSearchWithText(text)
  }
  .deliverOn(RACScheduler.mainThreadScheduler())
  .subscribeNextAs({
    ...
  }, {
    (error) in
    println(error)
  })
```

Fetching Sentiment Data

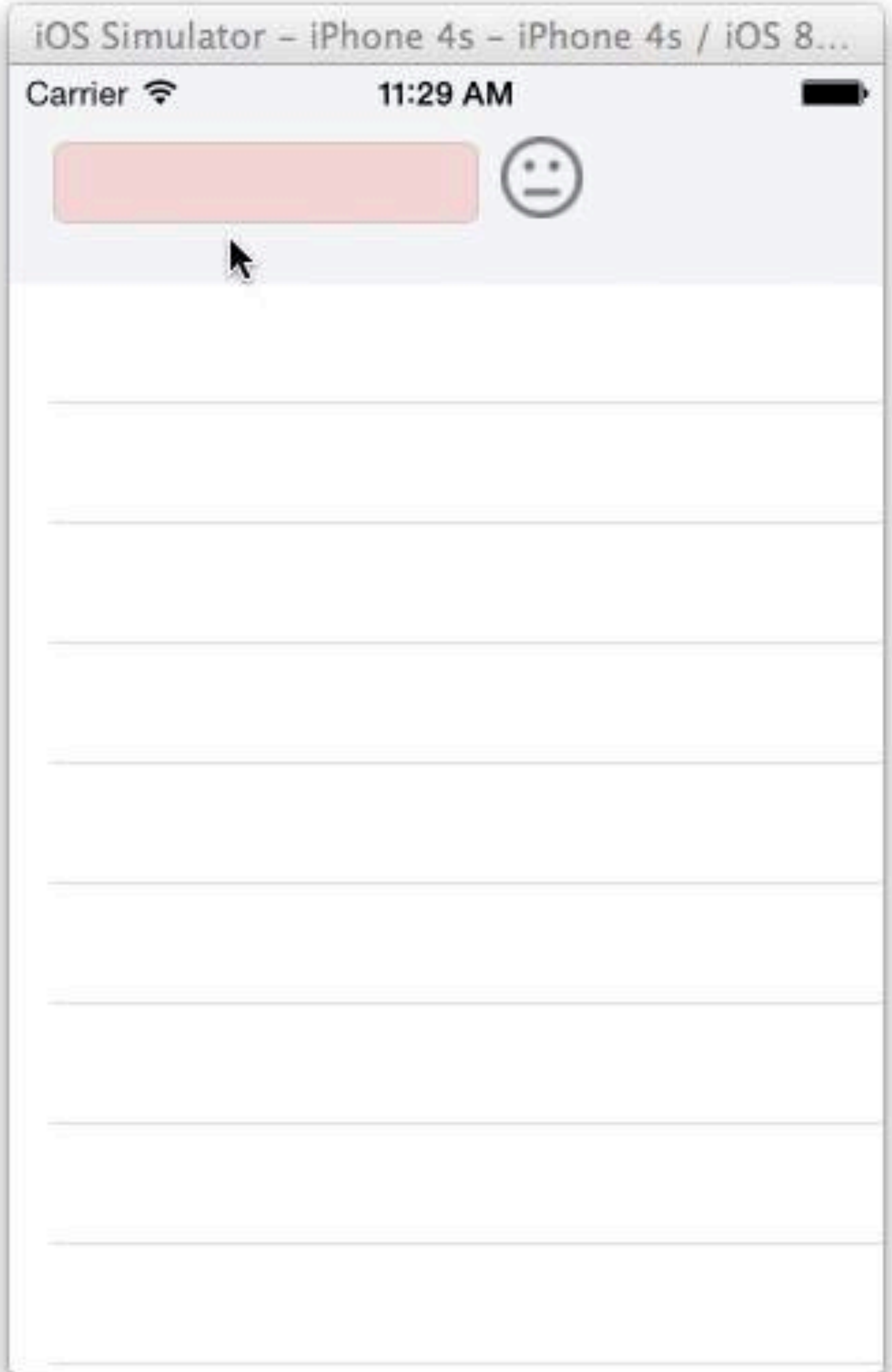
Fire a sentiment API request for each tweet and merge the signals?

Fetch sentiment data when cells become visible?

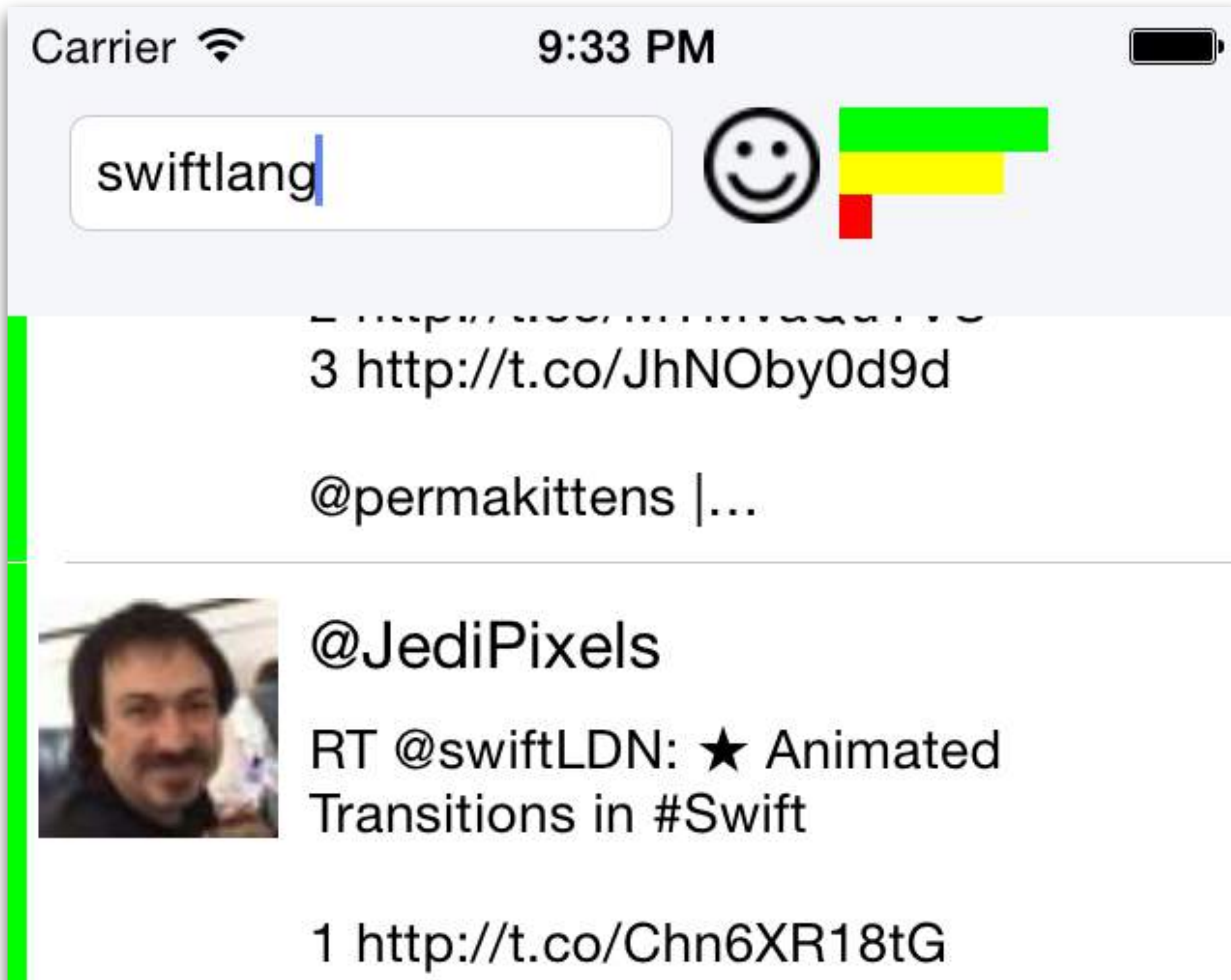
Fetch sentiment data when a cell has been visible for a short duration?

Signal All Things!

```
RACSignal
  .interval(0.5, onScheduler:
        RACScheduler(priority: RACSchedulerPriorityBackground))
  .take(1)
  .takeUntil(rac_prepareForReuseSignal)
  .flattenMap {
    (next) -> RACStream in
    self.obtainSentimentSignal(hasTweet)
  }
  .deliverOn(RACScheduler.mainThreadScheduler())
  .subscribeNextAs {
    (sentiment: String) in
    NSNotificationCenter.defaultCenter()
        .postNotificationName("sentiment", object: sentiment)
    self.sentimentIndicator.backgroundColor =
                    self.sentimentToColor(sentiment)
  }
```


iOS Simulator – iPhone 4s – iPhone 4s / iOS 8...
Carrier
11:29 AM


I curried
a function!


Carrier
9:33 PM
swiftlang
3 http://t.co/JhNOby0d9d
@permakittens |...
@JediPixels
RT @swiftLDN: ★ Animated Transitions in #Swift
1 http://t.co/Chn6XR18tG

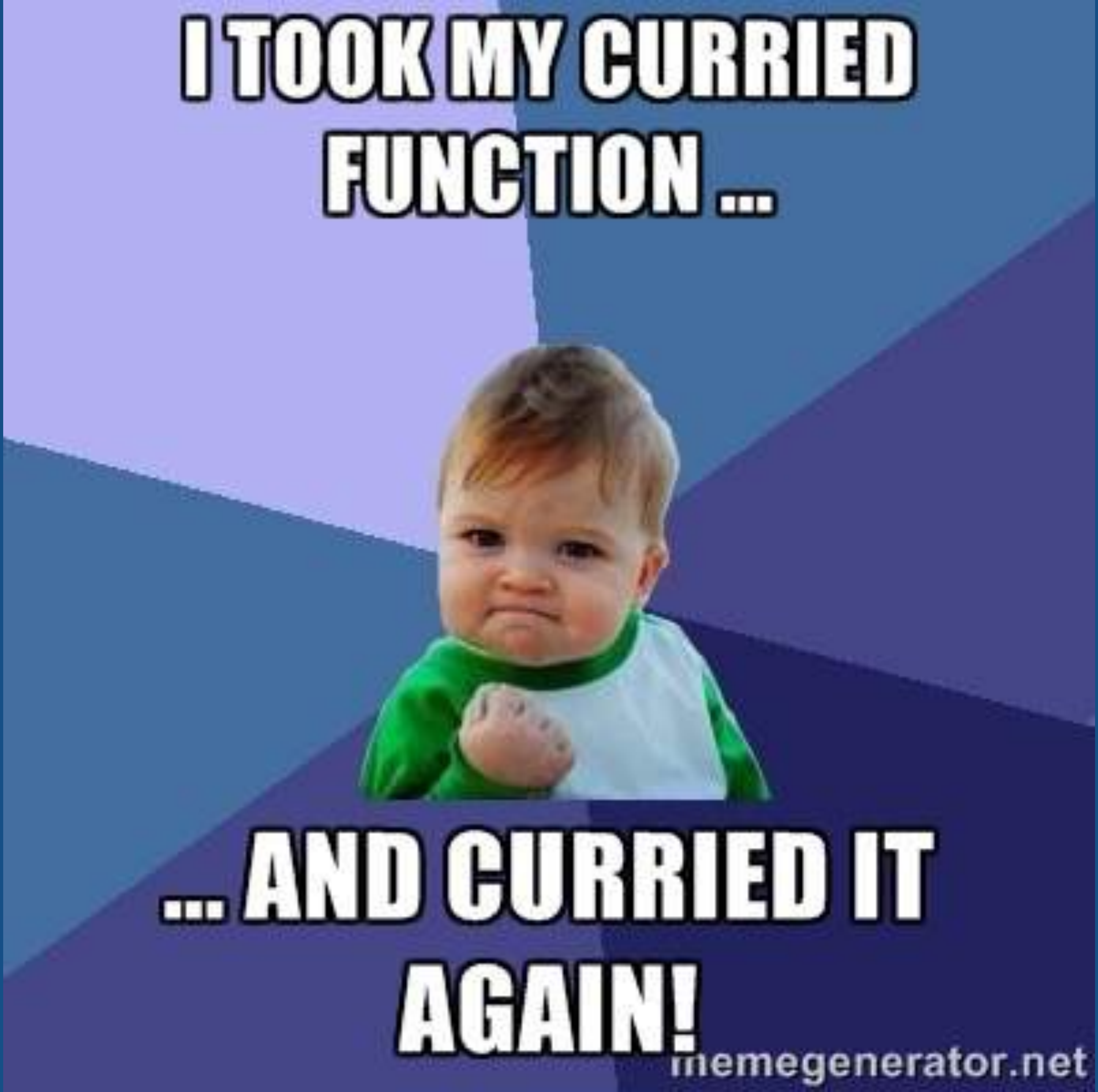
```swift
func scale(domainMin: Double, domainMax: Double, screenMin: Double,
                                  screenMax: Double, pt: Double) -> CGFloat {

  let value = ((pt - domainMin) / (domainMax - domainMin))
                                       * (screenMax - screenMin) + screenMin
  return CGFloat(value)
}
```

```
func scale(domainMin: Double, domainMax: Double, screenMin: Double,
                                  screenMax: Double)(pt: Double) -> CGFloat {

  let value = ((pt - domainMin) / (domainMax - domainMin))
                                        * (screenMax - screenMin) + screenMin
  return CGFloat(value)
}
```

```swift
let xscale = scale(0, Double(maxValue), 35, Double(self.bounds.width - 35))
let yscale = scale(0, 3, 0, Double(self.bounds.height))
```

```swift
positiveLayer.frame = CGRect(x: xscale(pt: 0), y: yscale(pt: 1),
      width: xdelta(pt1: Double(positive), pt2: 0.0), height: yscale(pt: 1))
neutralLayer.frame = ...
```

```swift
func delta(scale:(Double) -> CGFloat)(pt1: Double, pt2: Double) -> CGFloat {
  return scale(pt1) - scale(pt2)
}

let xdelta = delta(xscale)
```


I TOOK MY CURRIED FUNCTION ...
... AND CURRIED IT AGAIN!
memegenerator.net

Swift

ReactiveCocoa

Swift encourages immutability

51 variables

39 constants

12 variables

- 6 outlets (not my fault!)
- 1 UIWindow
- 5 UI state variables

ReactiveCocoa 3

```
public final class Signal<T, E: ErrorType> {
  ...
}
```

```swift
requestAccessToTwitterSignal()
  .then {
    self.searchTextField.rac_textSignal()
  }
  .filterAs {
    (text: NSString) -> Bool in
    text.length > 3
  }
  .doNext {
    (any) in
    self.tweetsTableView.alpha = 0.5
  }
  .throttle(0.5)
  .flattenMapAs {
    (text: NSString) -> RACStream in
    self.signalForSearchWithText(text)
  }
  .deliverOn(RACScheduler.mainThreadScheduler())
  .subscribeNextAs ({
    (tweets: NSDictionary) in
    let statuses = tweets["statuses"] as [NSDictionary]
    self.tweets = statuses.map { Tweet(json: $0) }
    self.tweetsTableView.reloadData()
    self.tweetsTableView.scrollToTop()
    self.tweetsTableView.alpha = 1.0
  }, {
    (error) in
    println(error)
  })
```

```swift
requestAccessToTwitterSignal()
  |> then (textField.rac3_textSignal())
  |> filter {
    countElements($0) > 3
  }
  |> throttle(0.5, onScheduler: QueueScheduler.mainQueueScheduler)
  |> doNext {
    text in self.tweetsTableView.alpha = 0.5
  }
  |> flattenMap {
    self.signalForSearchWithText($0)
  }
  |> observeOn(QueueScheduler.mainQueueScheduler)
  |> observe(next: {
    tweetsDictionary in
    let statuses = tweetsDictionary["statuses"] as [NSDictionary]
    self.tweets = statuses.map { Tweet(json: $0) }
    self.tweetsTableView.reloadData()
    self.tweetsTableView.alpha = 1.0
  })
```

pipe forward operator

|>

```swift
public final class Signal<T, E: ErrorType> {
    func map (…) -> Signal
}
```

```
public final class Signal<T, E: ErrorType> {

}

func map (…) -> Signal
```

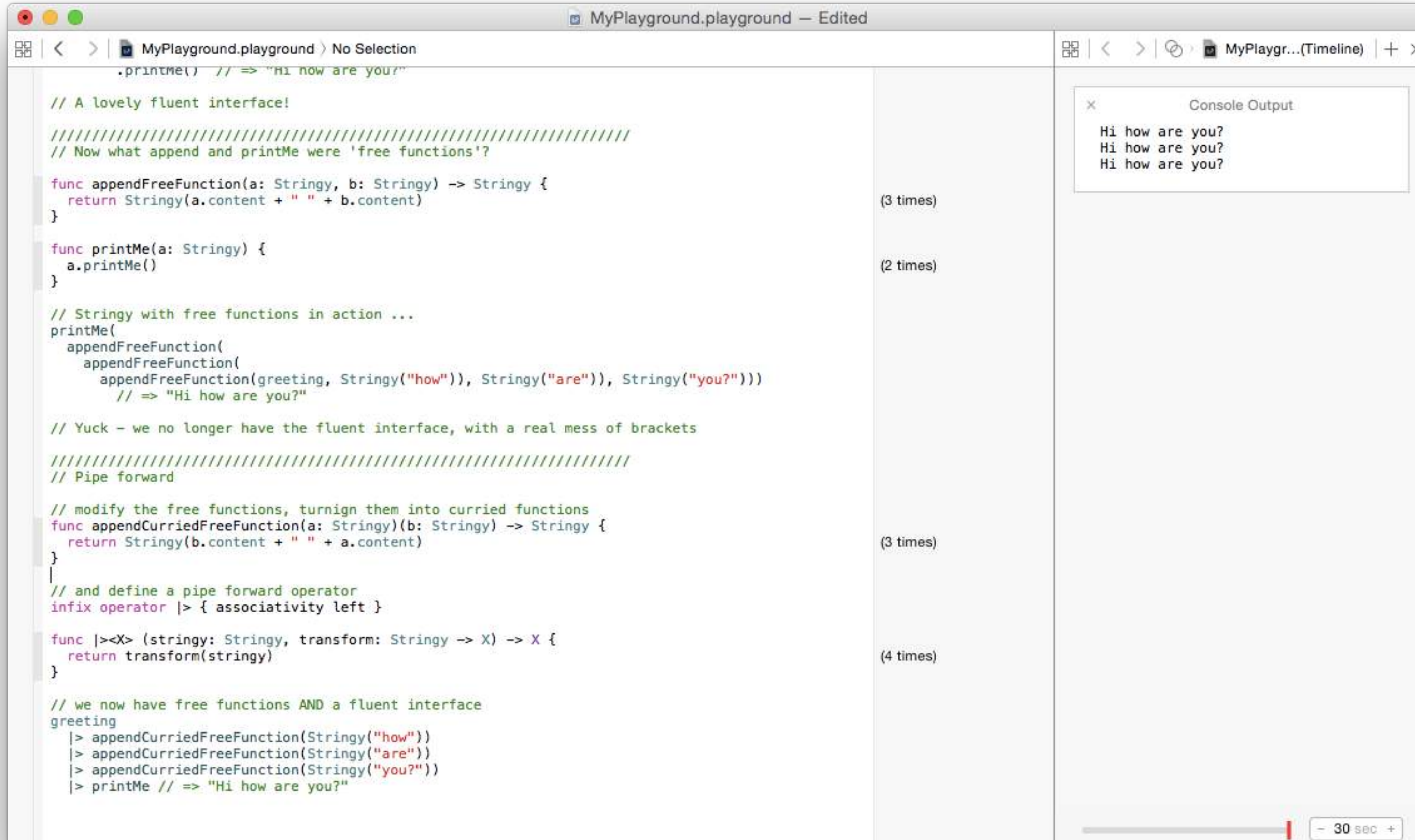
```
let mapped = map(signal, { $0.foo })
```

```
let mapped = map(map(signal, { $0.foo }), { $0.bar })
```

```
let mapped = map(map(map(signal, { $0.foo }), { $0.bar }), { $0.sadface })
```

```
func map<T, U, E>(transform: T -> U)
                 (signal: Signal<T, E>) -> Signal<U, E>
```

```swift
public func |> <T, E, X>
    (signal: Signal<T, E>, transform: Signal<T, E> -> X) -> X {
  return transform(signal)
}
```


MyPlayground.playground — Edited
MyPlayground.playground
No Selection
MyPlaygr...(Timeline)
.printMe() // => "Hi how are you?"
// A lovely fluent interface!
///
// Now what append and printMe were 'free functions'?
func appendFreeFunction(a: Stringy, b: Stringy) -> Stringy {
return Stringy(a.content + " " + b.content)
}
(3 times)
func printMe(a: Stringy) {
a.printMe()
}
(2 times)
// Stringy with free functions in action ...
printMe(
appendFreeFunction(
appendFreeFunction(
appendFreeFunction(greeting, Stringy("how")), Stringy("are")), Stringy("you?")))
// => "Hi how are you?"
// Yuck - we no longer have the fluent interface, with a real mess of brackets
///
// Pipe forward
// modify the free functions, turnign them into curried functions
func appendCurriedFreeFunction(a: Stringy)(b: Stringy) -> Stringy {
return Stringy(b.content + " " + a.content)
}
(3 times)
// and define a pipe forward operator
infix operator |> { associativity left }
func |><X> (stringy: Stringy, transform: Stringy -> X) -> X {
return transform(stringy)
}
(4 times)
// we now have free functions AND a fluent interface
greeting
|> appendCurriedFreeFunction(Stringy("how"))
|> appendCurriedFreeFunction(Stringy("are"))
|> appendCurriedFreeFunction(Stringy("you?"))
|> printMe // => "Hi how are you?"
Console Output
Hi how are you?
Hi how are you?
Hi how are you?
30 sec

```swift
func |><X> (stringy: Stringy, transform: Stringy -> X) -> X {
  return transform(stringy)
}
```

```
func |><X>
  return
}
```

ReactiveCocoa

Swift

ReactiveCocoa

@ColinEberhardt

http://tiny.cc/reactive-swift