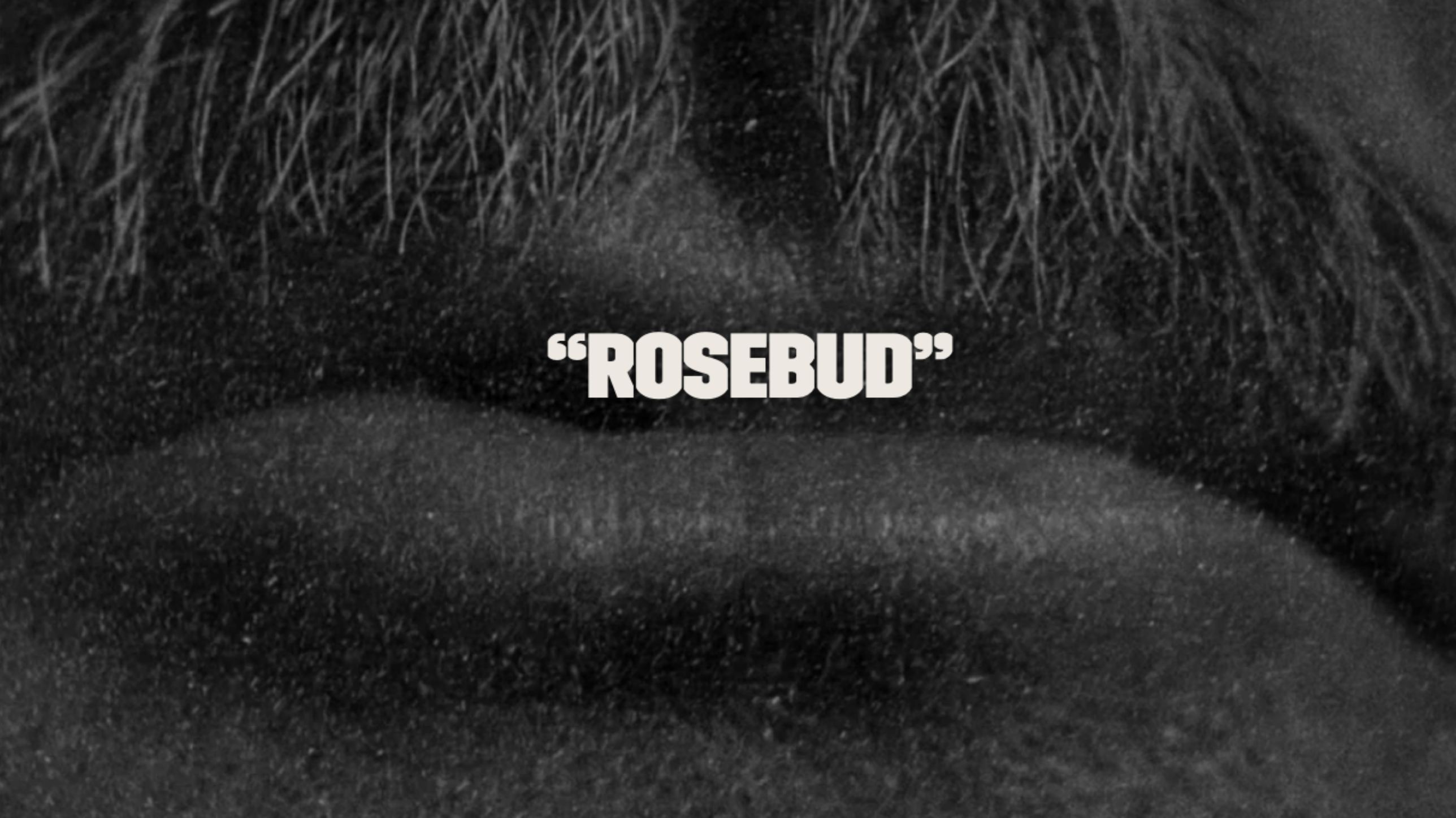
THE REACTIVE CODE THAT BROKE MY BRAIN

AND CHANGED MY MIND (JUST IN TIME FOR COMBINE)

Lou Franco
@loufranco


“ROSEBUD”


"I LOVE YOU
HONEY BUNNY"

THE CODE THAT CHANGED MY MIND

```
let output = viewModel.transform(input)
```

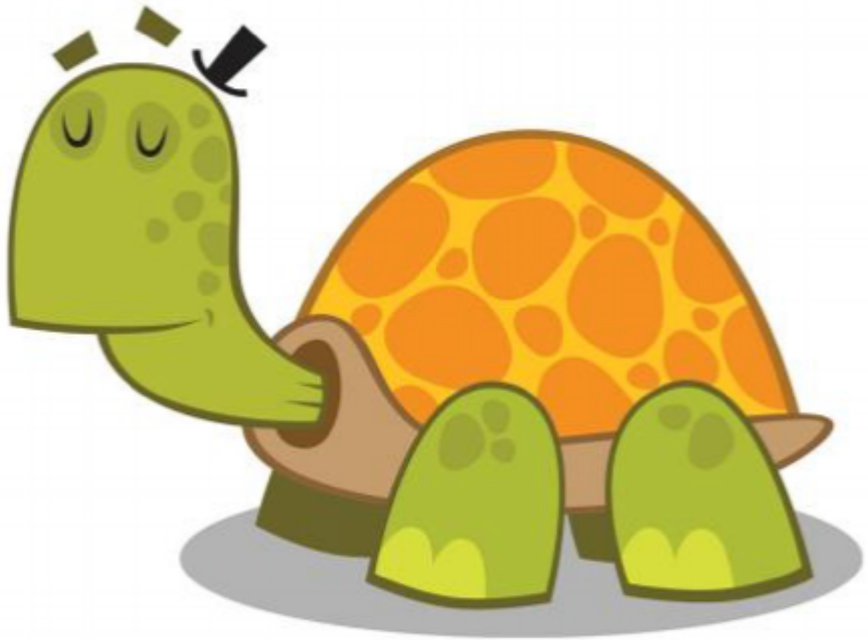
```swift
protocol IOViewModelType {
    associatedtype Input
    associatedtype Output

    func transform(_ input: Input) -> Output
}
```

FIVE YEARS EARLIER ...

The Pragmatic Programmers

Swift Style

An Opinionated Guide to an Opinionated Language



Erica Sadun

edited by Brian MacDonald

Copyrighted Material

Google

swift style

All | Images | Videos | News | Shopping | More | Settings | Tools

About 298,000,000 results (0.63 seconds)

Taylor Swift - Style - YouTube

https://www.youtube.com/watch?v=-CmadmM5cOk


“WE USE REACTIVE COCOA HERE”

MY INTRO TO REACTIVE PROGRAMMING

```
RACSignal *countSignal =
  [self.textField.rac_textSignal map:^id(NSString *text) {
    return [NSString stringWithFormat:@"%i", text.length];
  }];
RAC(self.countLabel, text) = countSignal;
```

REACTIVE PROGRAMMING

SEQUENCE TYPES

» Sequence: source of values

» Iterator: pull the values

SEQUENCE TYPES

» Sequence: source of values

» Iterator: pull the values

```
// values: [Int]

var sumOfOdds = 0
for v in values {
  if v % 2 == 1 {
    sumOfOdds += v
  }
}
print(sumOfOdds)
```

REACTIVE TYPES

» Publisher/Observable: source of values

» Subscriber/Observer: asynchronously pushed values

REACTIVE TYPES

» Publisher/Observable: source of values

» Subscriber/Observer: asynchronously pushed values

```
// values: AnyPublisher<Int, Never>

var sumOfOdds = 0
values.sink(receiveCompletion: { (_) in
    print(sumOfOdds)
}, receiveValue: { v in
    if v % 2 == 1 {
        sumOfOdds += v
    }
})
```

CONSEQUENCES OF ASYNCHRONY

- Time
- Order
- Threads
- Errors
- Cancellation

SEQUENCE EXAMPLE

```
// values: [Int]

var sumOfOdds = 0
for v in values {
  if v % 2 == 1 {
    sumOfOdds += v
  }
}
print(sumOfOdds)
```

SEQUENCE OPERATORS

```
sumOfOdds = values
  .filter { $0 % 2 == 1 }
  .reduce(0, +)
```

REACTIVE EXAMPLE

```
// values: AnyPublisher<Int, Never>

var sumOfOdds = 0
values.sink(receiveCompletion: { (_) in
    print(sumOfOdds)
}, receiveValue: { v in
    if v % 2 == 1 {
        sumOfOdds += v
    }
})
```

REACTIVE OPERATORS (COMBINE)

```
// values: AnyPublisher<Int, Never>
values
    .filter { $0 % 2 == 1 }
    .reduce(0, +)
    .sink { sumOfOdds = $0 }
```

REACTIVE OPERATORS (RXSWIFT)

```
// values: Observable<Int>
values
    .filter { $0 % 2 == 1 }
    .reduce(0, +)
    .subscribe { sumOfOdds = $0 }
```

OPERATORS ARE FAMILIAR

SEQUENCES	PUBLISHERS
map	map
flatMap	flatMap
filter	filter
reduce	reduce / scan
first	first

COMBINE AND RXSWIFT

COMBINE	RXSWIFT
Publisher	Observable
Subscriber	Observer
Subject	Subject
map/filter/reduce/scan	map/filter/reduce/scan
merge/flatMap/combineLatest	merge/flatMap/combineLatest
debounce/delay/throttle	debounce/delay/throttle

REACTIVE PROGRAMMING REPLACES

» Completion Closures

» KVO

» Target/Action calling

» Notification Center

» Delegates

» DataSources

with one model (using Subjects)

SO...

- » Publishers send values asynchronously
  - » Introducing time, order, threads, errors, and cancellation
- » Operators combine and transform Publishers
- » Rx replaces KVO, delegates, closures, etc
- » Subjects can adapt other models to Rx

MY INTRO TO REACTIVE PROGRAMMING

```
RACSignal *countSignal =
  [self.textField.rac_textSignal map:^id(NSString *text) {
    return [NSString stringWithFormat:@"%i", text.length];
  }];
RAC(self.countLabel, text) = countSignal;
```

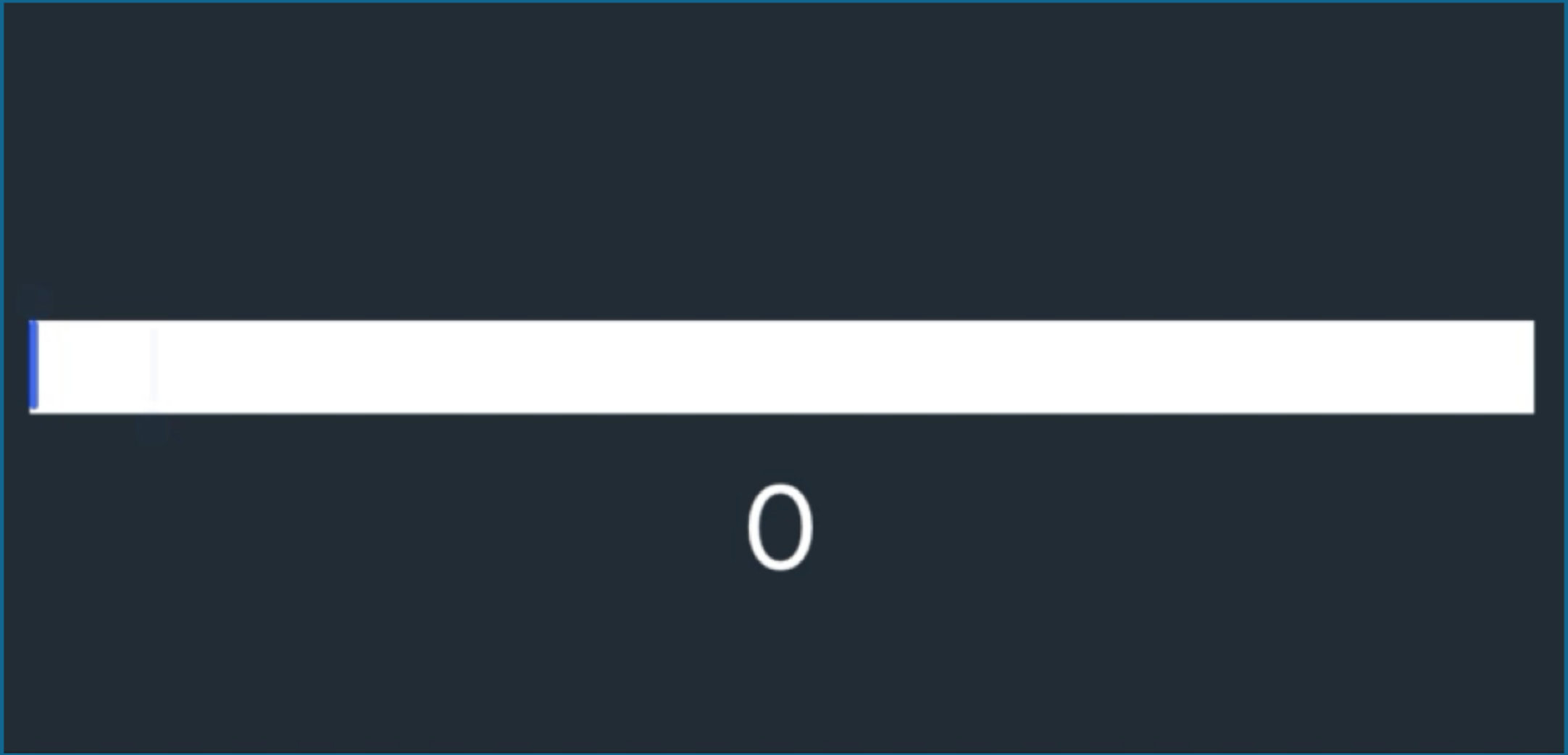
MY INTRO TO REACTIVE PROGRAMMING

```
RACSignal *countSignal =
  [self.textField.rac_textSignal map:^id(NSString *text) {
    return [NSString stringWithFormat:@"%i", text.length];
  }];
RAC(self.countLabel, text) = countSignal;
```

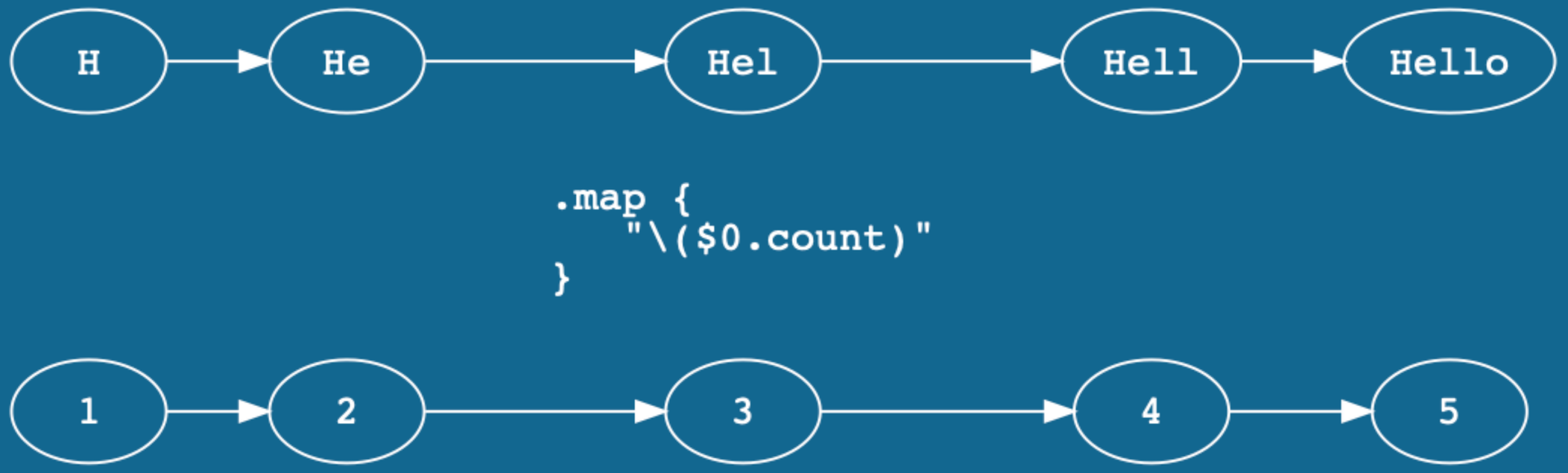
RXSWIFT EQUIVALENT

```
self.textField.rx.text
  .map { "\($0.count)" }
  .bind(to: self.countLabel.rx.text)
```

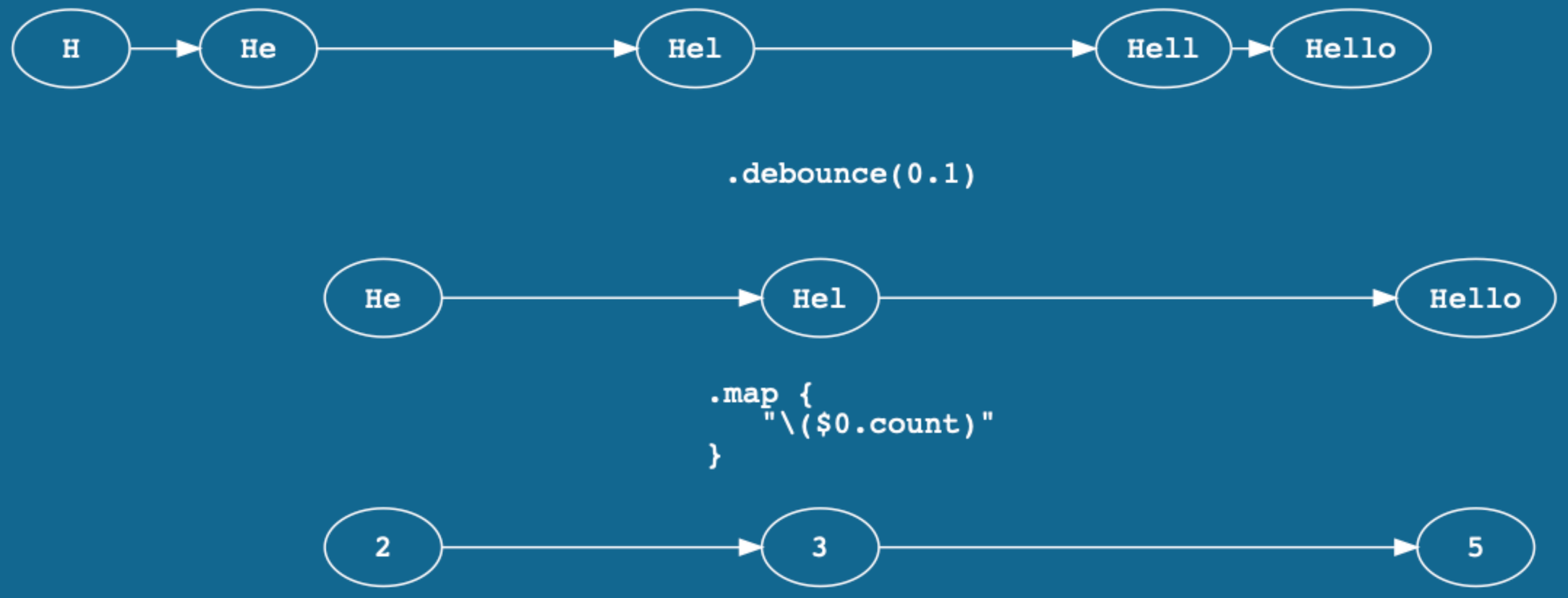
DEMO



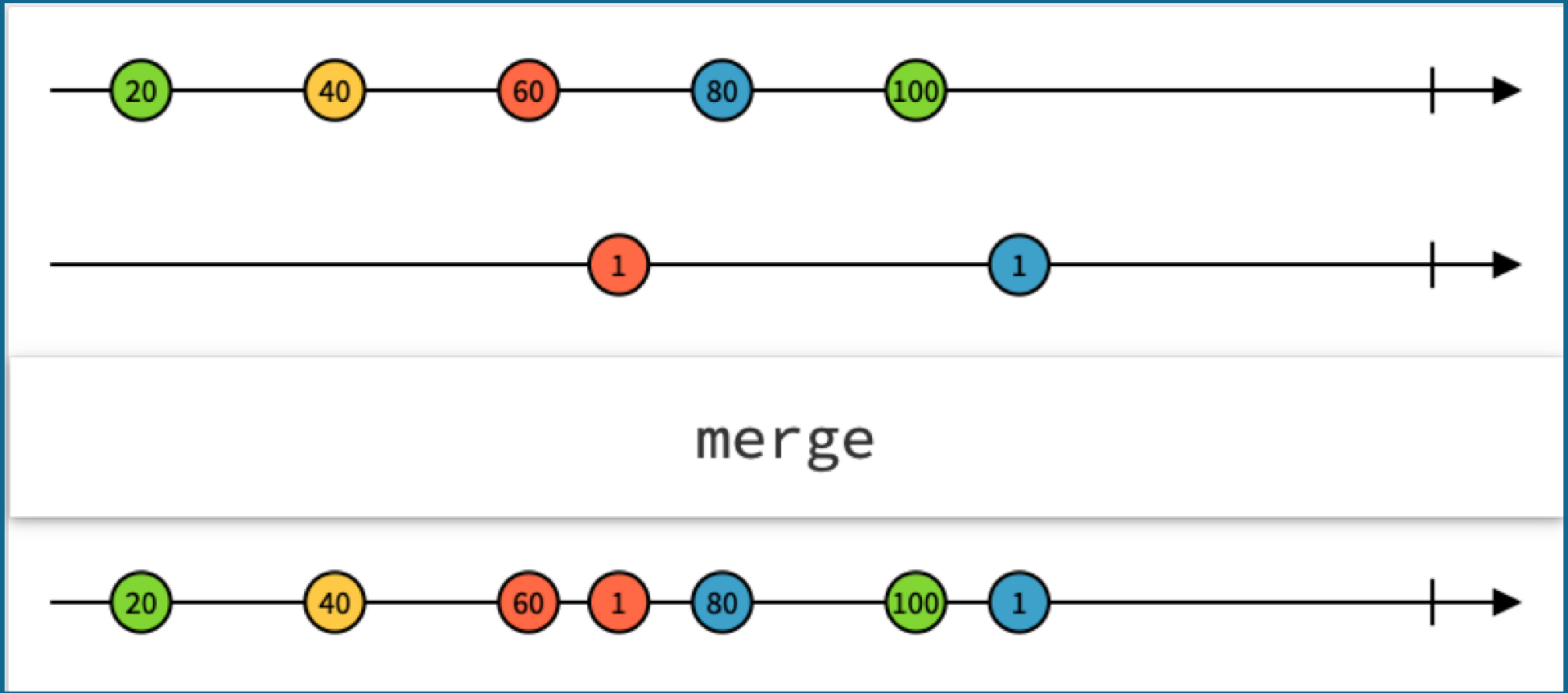
MARBLE DIAGRAM: MAP



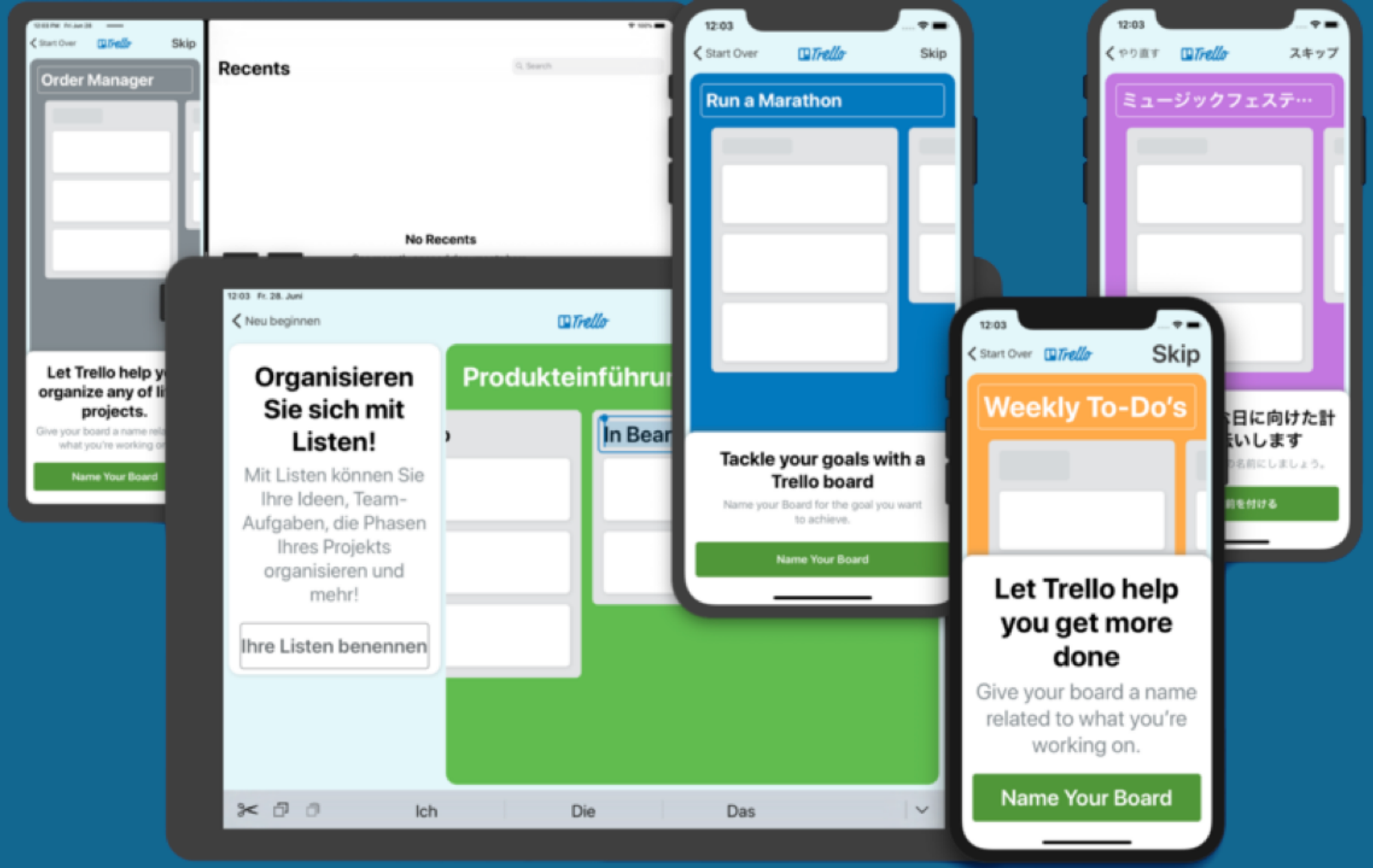
MARBLE DIAGRAM: DEBOUNCE

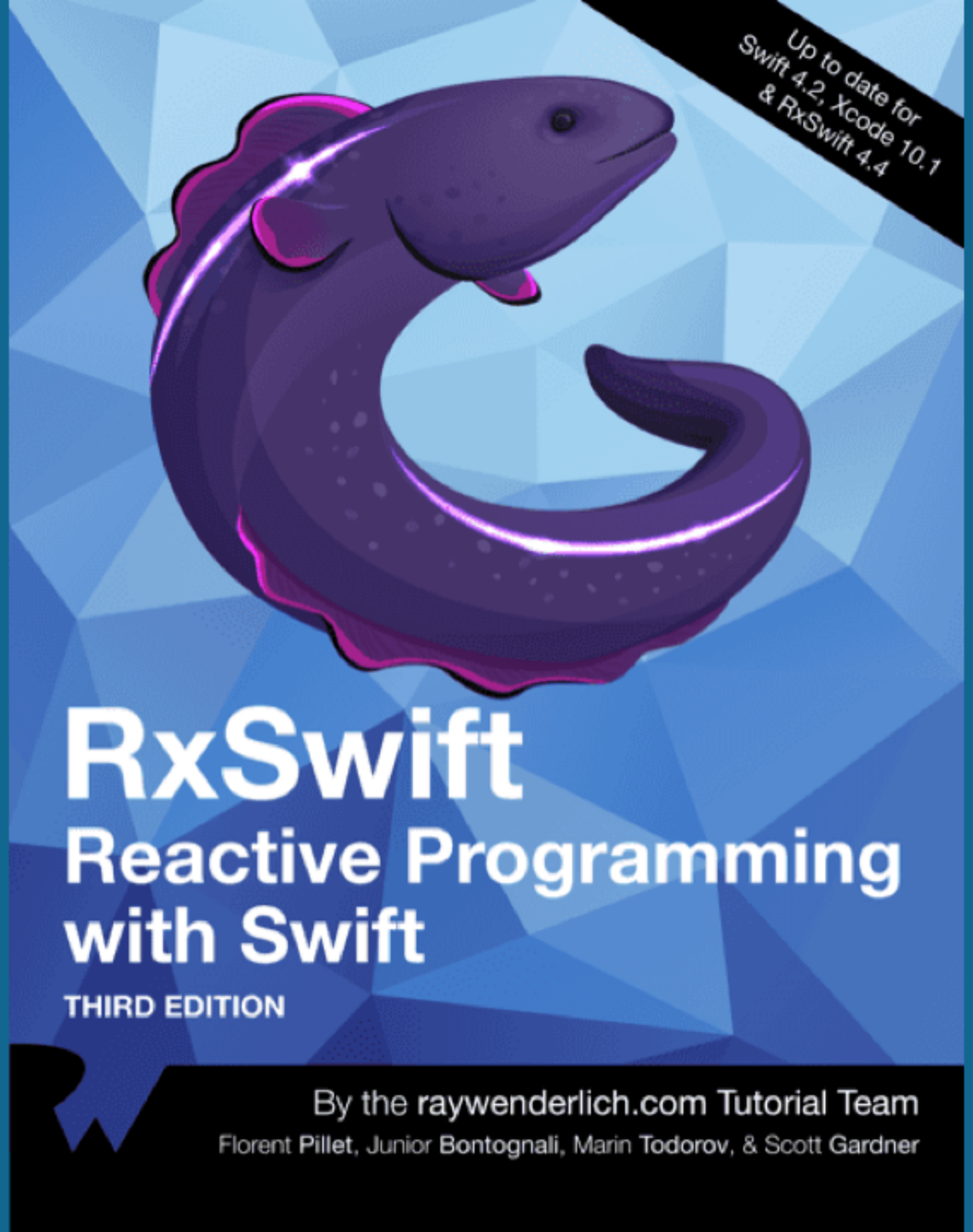


RXMARBLES.COM



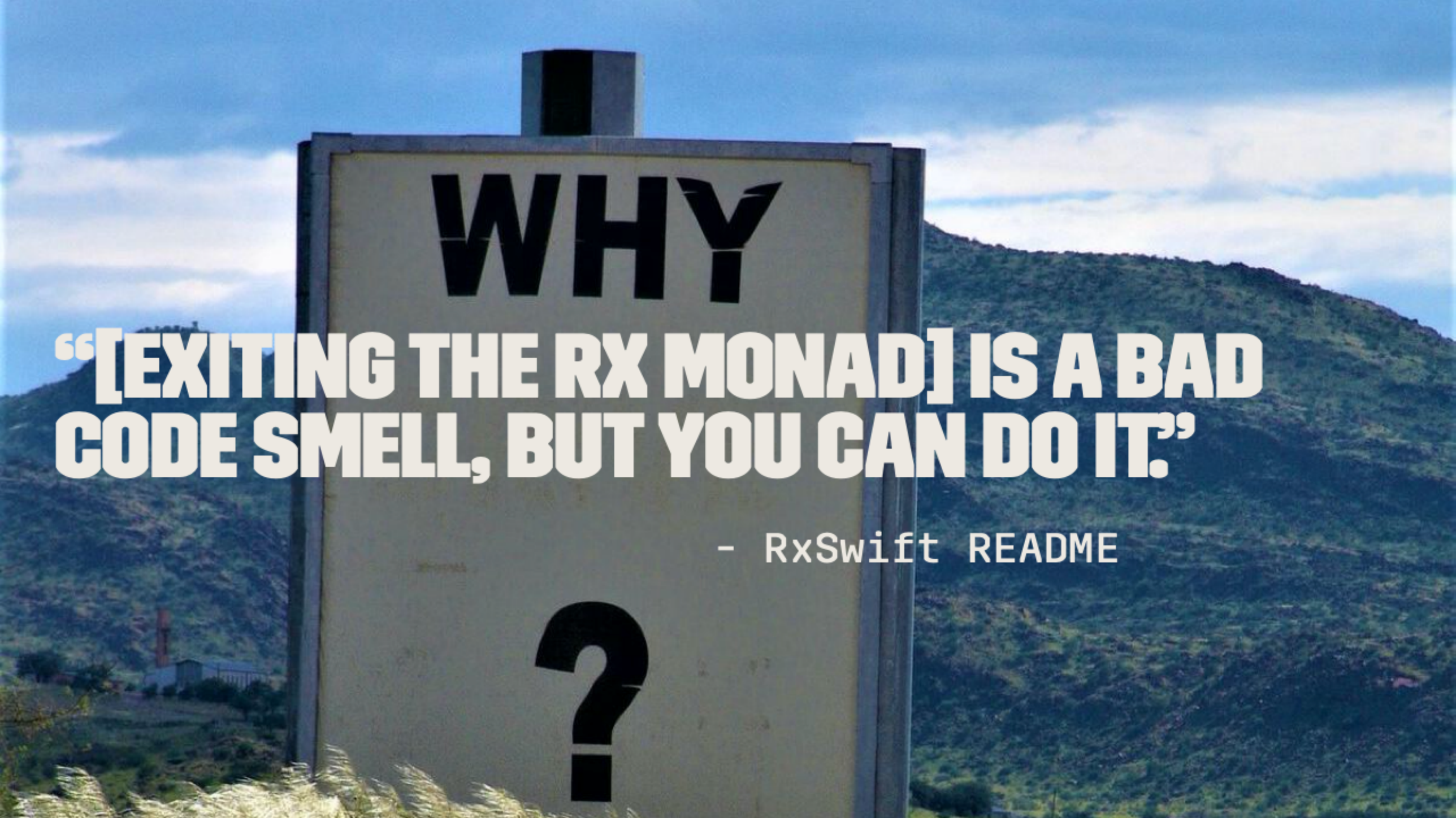

MY RX USAGE UP UNTIL 2019


Start Over
Trello
Skip
Order Manager
Let Trello help y
organize any of li
projects.
Name Your Board
Recents
No Recents
12:03 Fr. 28. Juni
Neu beginnen
Trello
Organisieren Sie sich mit Listen!
Mit Listen können Sie Ihre Ideen, Team-Aufgaben, die Phasen Ihres Projekts organisieren und mehr!
Ihre Listen benennen
Produkteinführu
In Bear
Ich
Die
Das
12:03
Start Over
Trello
Skip
Run a Marathon
Tackle your goals with a Trello board
Name your Board for the goal you want to achieve.
Name Your Board
12:03
やり直す
Trello
スキップ
ミュージックフェステ…
日に向けた計
いします
前を付ける
12:03
Start Over
Trello
Skip
Weekly To-Do's
Let Trello help you get more done
Give your board a name related to what you're working on.
Name Your Board


Up to date for
Swift 4.2, Xcode 10.1
& RxSwift 4.4
RxSwift
Reactive Programming
with Swift
THIRD EDITION
By the raywenderlich.com Tutorial Team
Florent Pillet, Junior Bontognali, Marin Todorov, & Scott Gardner

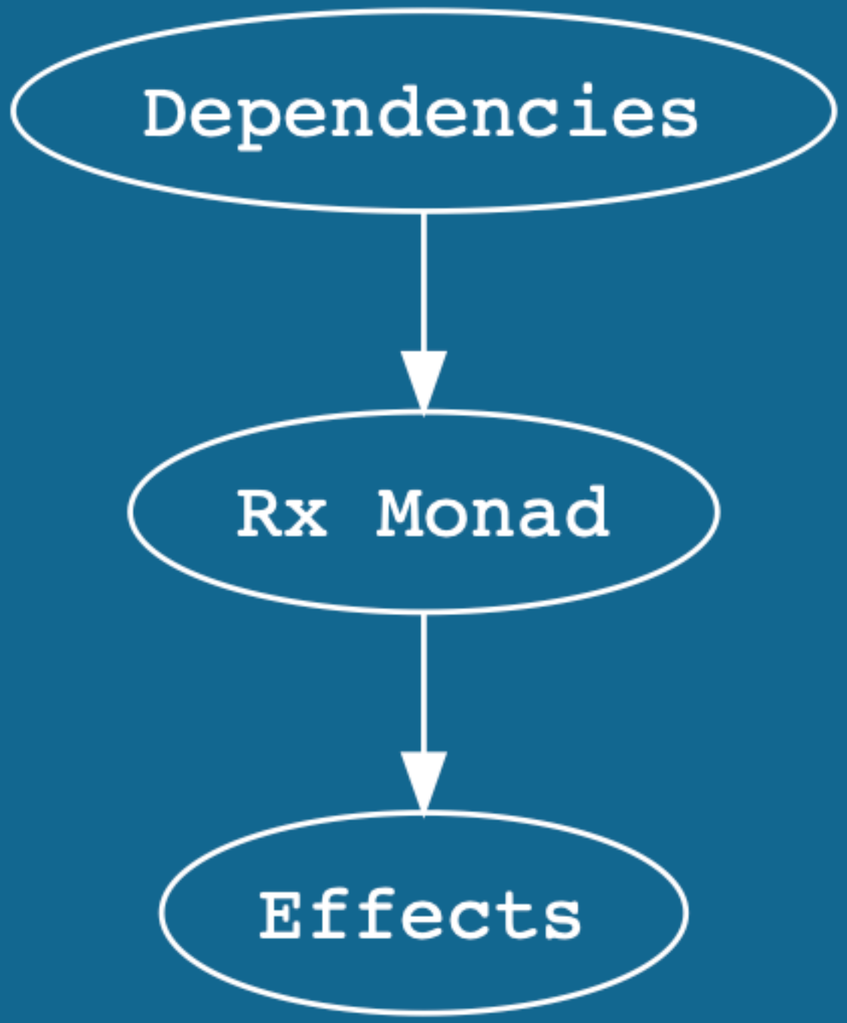

objc ↑↓
App
Architecture
iOS Application Design Patterns in Swift
By Chris Eidhof, Matt Gallagher, and Florian Kugler

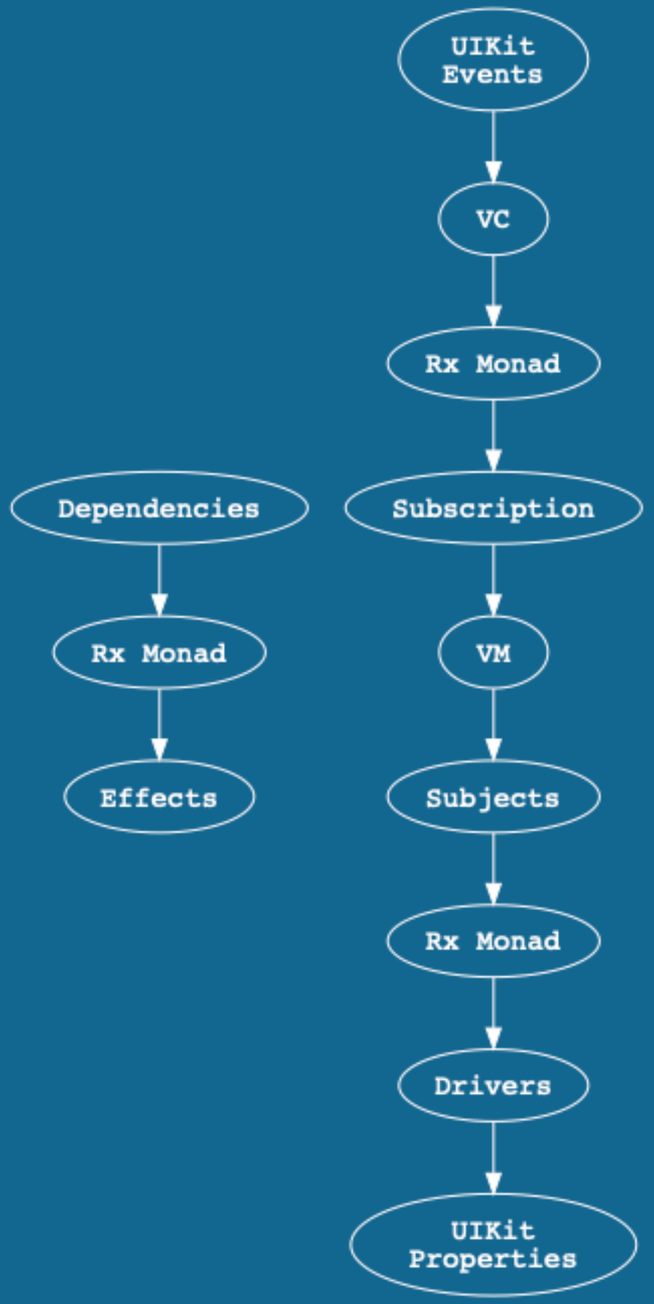
REACTIVE MVVM

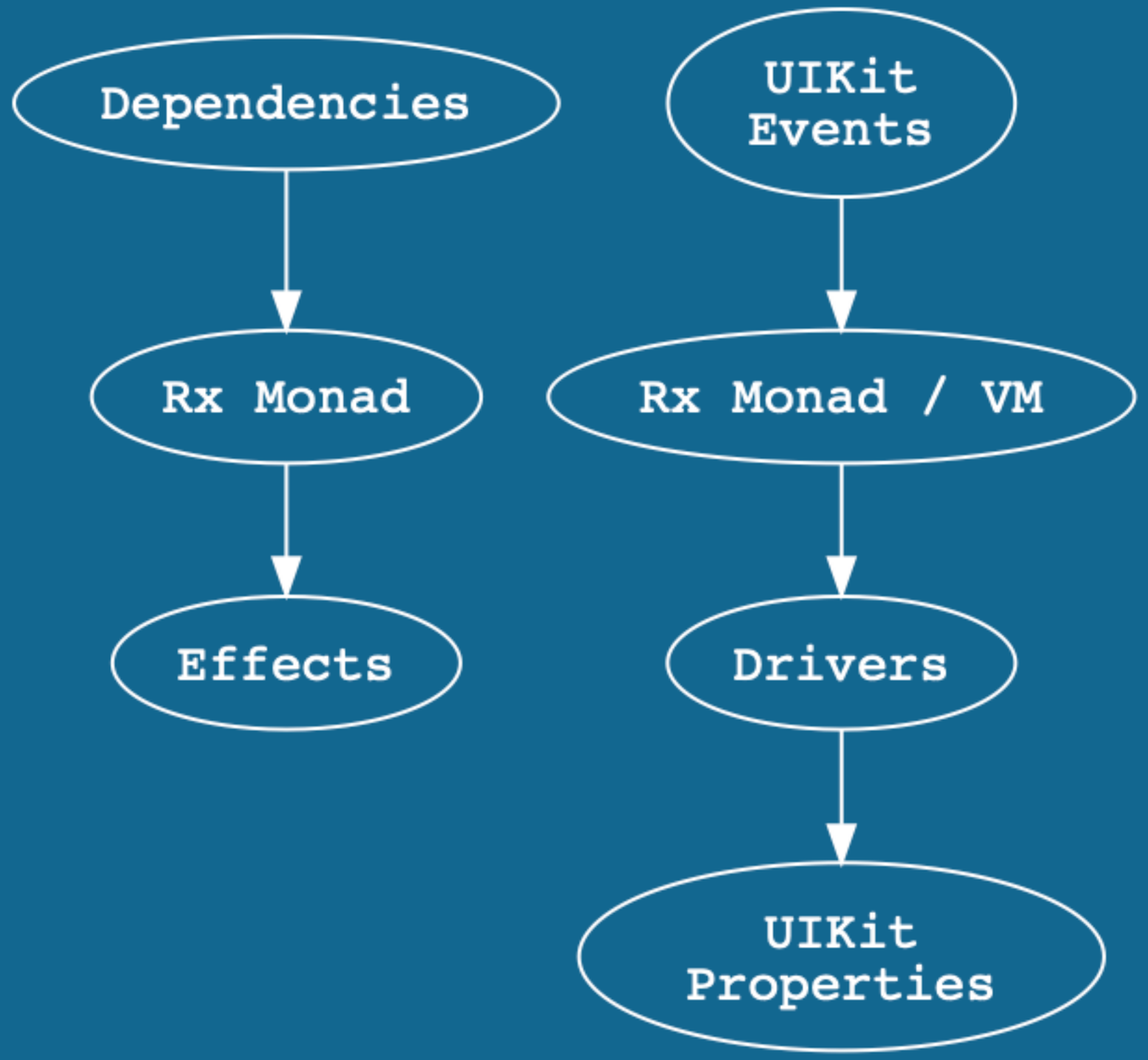


“[EXITING THE RX MONAD] IS A BAD CODE SMELL, BUT YOU CAN DO IT.”

- RxSwift README


Dependencies
Rx Monad
Effects

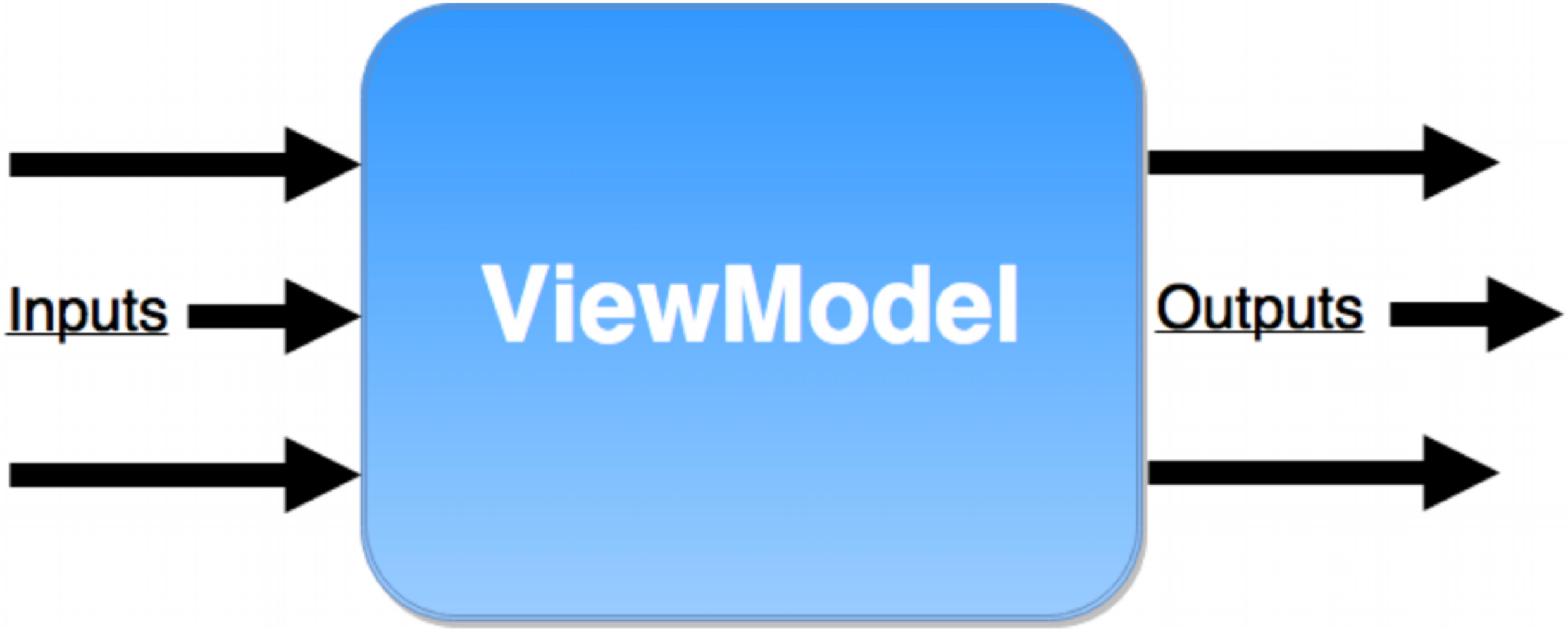

UIKit Events
VC
Rx Monad
Subscription
VM
Subjects
Rx Monad
Drivers
UIKit Properties
Dependencies
Rx Monad
Effects


Dependencies
Rx Monad
Effects
UIKit Events
Rx Monad / VM
Drivers
UIKit Properties

RxSwift + MVVM: how to feed ViewModels

Martin Moizard Follow

Sep 27, 2017 · 6 min read


Inputs
ViewModel
Outputs

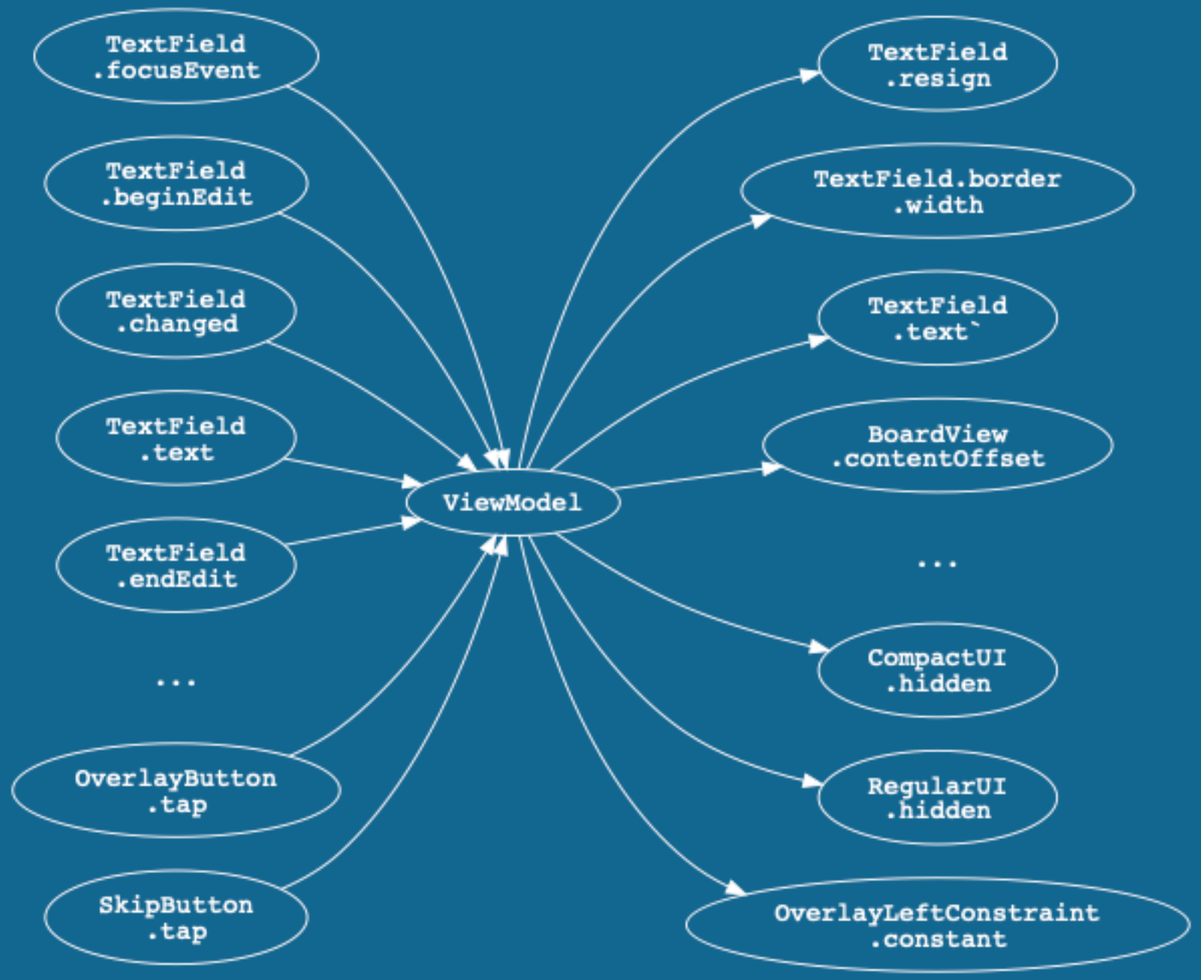
VIEWMODEL PUBLIC INTERFACE

```
protocol IOViewModelType {
    associatedtype Input
    associatedtype Output

    func transform(_ input: Input) -> Output
}
```

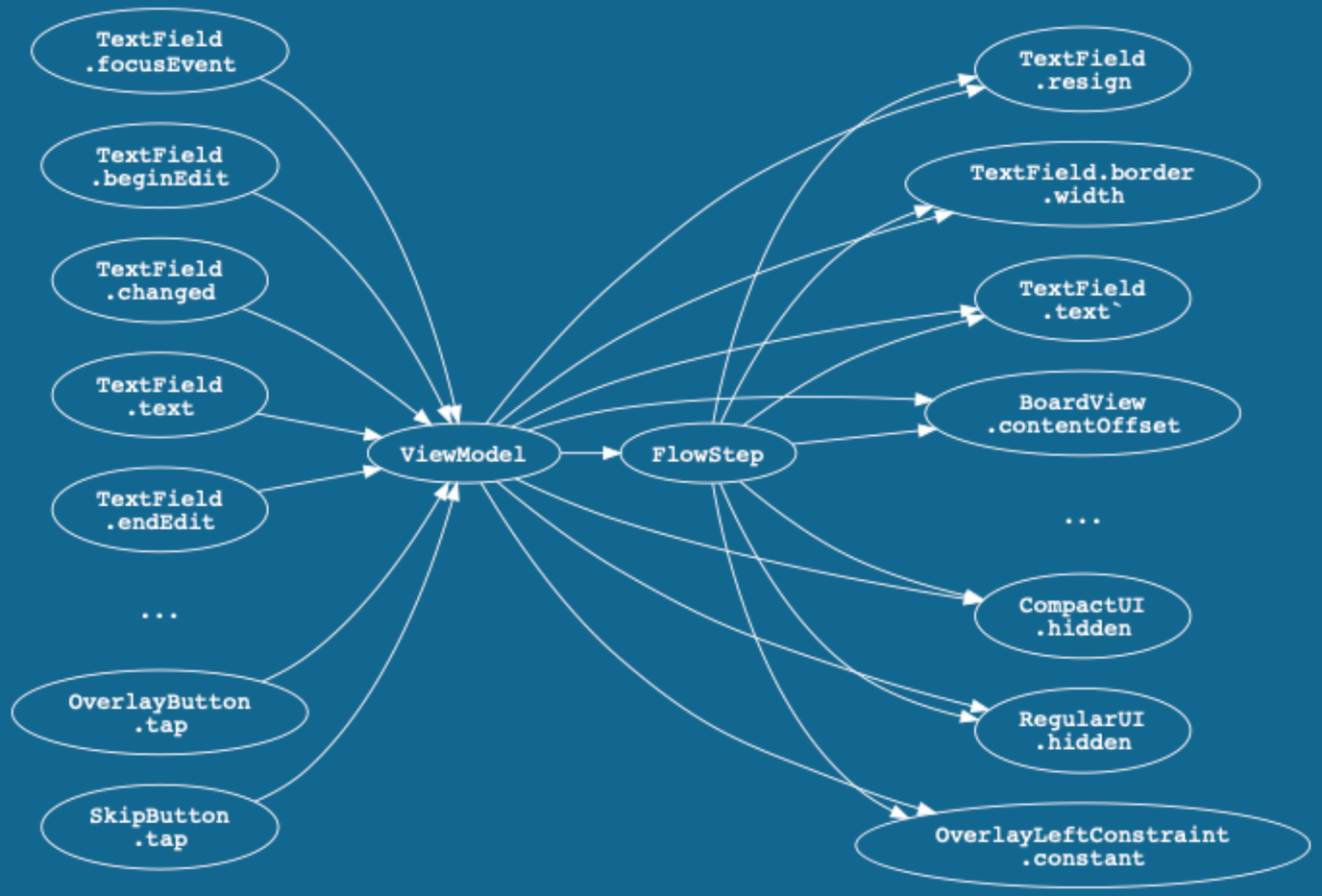
VIEWCONTROLLER USE

```
let output = viewModel.transform(input)
```

```
let output = viewModel.transform(input)
```



```
let flowStep = flowStep(input)
let textFieldResign = textFieldResign(input, flowStep)
```



FLOWSTEP

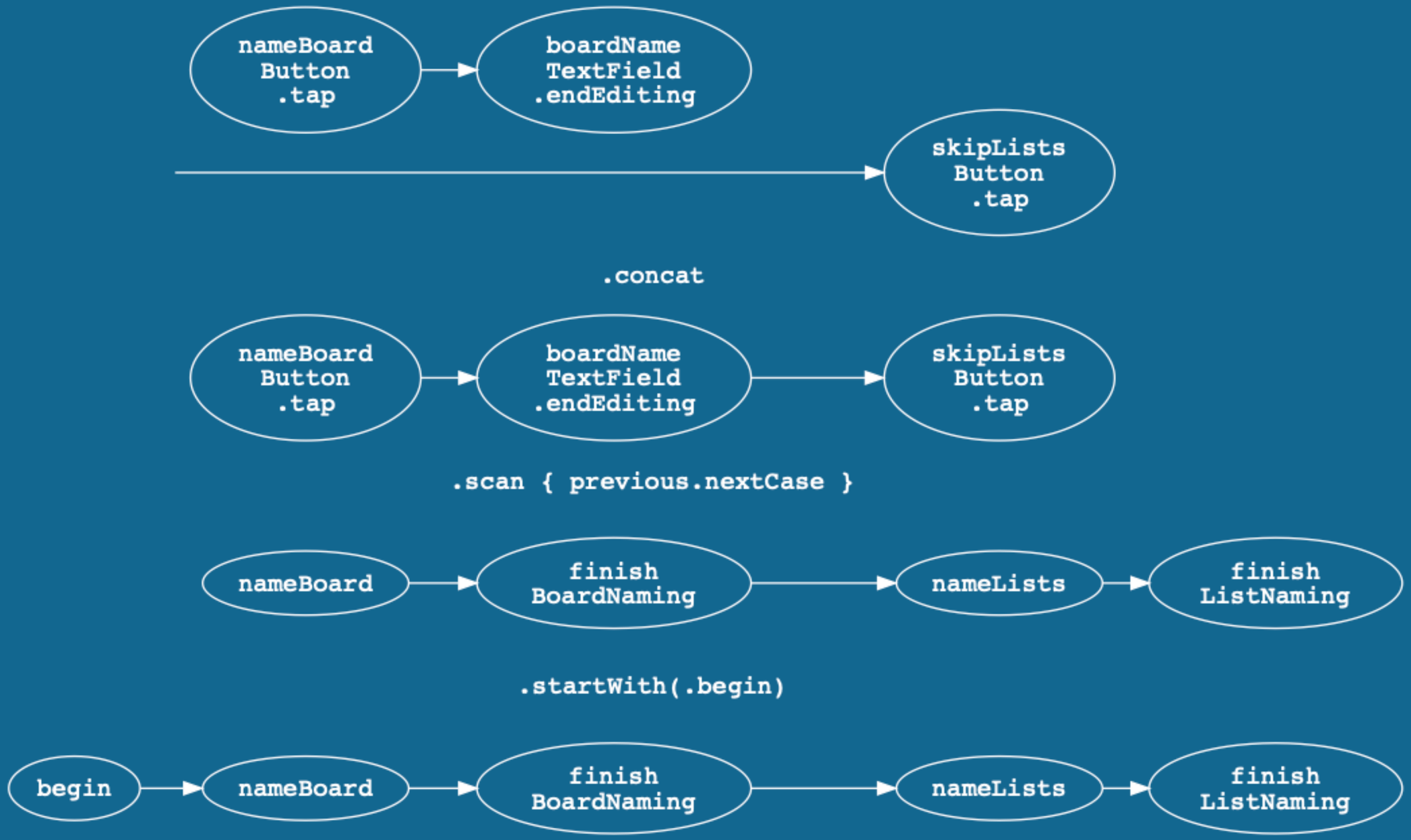

begin
name Board
finish BoardNaming
name Lists
finish ListNaming
name Cards
finish CardNaming
create Board
END

STATE MACHINE IN RX

```
enum FlowStep: Int, CaseIterable, Comparable {
    case begin
    case nameBoard
    case finishBoardNaming
    case nameLists
    case finishListNaming
    case nameCards
    case finishCardNaming
    case createBoard
}
```

STATE MACHINE IN RX

```
let stateMachine: Observable<FlowStep> =
  boardNameSteps
  .concat(listNameSteps)
  .concat(cardNameSteps)
  .concat(createBoardSteps)
  .scan(FlowStep.begin) { (previous, _) -> FlowStep in
      return previous.nextCase
  }
  .startWith(.begin)
```


nameBoard Button .tap
boardName TextField .endEditing
skipLists Button .tap
.concat
nameBoard Button .tap
boardName TextField .endEditing
skipLists Button .tap
.scan { previous.nextCase }
nameBoard
finish BoardNaming
nameLists
finish ListNaming
.startWith(.begin)
begin
nameBoard
finish BoardNaming
nameLists
finish ListNaming

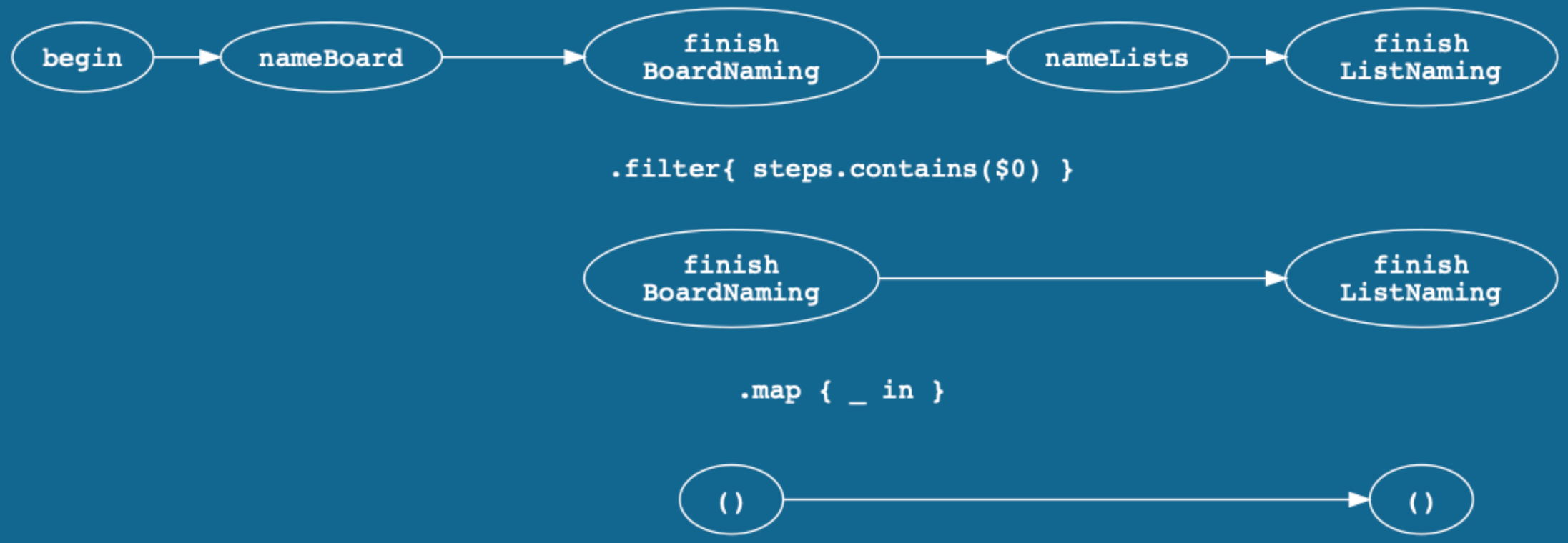
EXAMPLE: BUTTON ENABLING

```
let boardNamingButtonEnabled: Driver<Bool> =
    flowStep
        .filter { $0 > .begin }
        .map { _ in false }
        .take(1)
```

EXAMPLE: SCROLL BOARD

```
let scrollBoard: Driver<Void> =
  flowStep
    .filter {
      return [.finishBoardNaming,
              .finishListNaming,
              .finishCardNaming].contains($0)
    }
    .map { _ in }
```

SCROLL BOARD



EXAMPLE: SIZE ADAPTATION

```
func compactUIHidden(_ input: Input) -> Driver<Bool> {
    return input
      .traitCollection
      .map { $0.horizontalSizeClass != .compact }
}

func regularUIHidden(_ compactUIHidden: Driver<Bool>) -> Driver<Bool> {
    return compactUIHidden.not()
}

func boardLeadingConstraint(_ compactUIHidden: Driver<Bool>) -> Driver<CGFloat> {
    return compactUIHidden.map { [weak self] in
       $0 ? self.overlayWidth : 0
    }
}
```

INPUT

```
let input = OnboardingViewModel.Input(
    boardNameText: boardNameTextField.rx.text.orEmpty,
    boardNameEditingDidBegin: boardNameTextField.rx.controlEvent(.editingDidBegin),
    boardNameEditingChanged: boardNameTextField.rx.controlEvent(.editingChanged),
    boardNameEditingDidEnd: boardNameTextField.rx.controlEvent(.editingDidEnd),
    boardNameEditingDidEndOnExit: boardNameTextField.rx.controlEvent(.editingDidEndOnExit),
    ...
```

OUTPUT

```
let output = viewModel.transform(input)
```

TRANSFORM

```
let flowStep: Observable<FlowStep> = self.flowStep(input)

let boardNameDisplayText: Driver<String> = self.boardNameDisplayText(input, flowStep)
let focusBoardNameTextField: Driver<Bool> = self.focusBoardNameTextField(input)
let boardNameTextFieldSelectAllText: Driver<Void> = self.boardNameTextFieldSelectAllText(input)
let boardNameTextFieldShowActiveBorder: Driver<Bool> = self.boardNameTextFieldShowActiveBorder(input)
let boardNameTextFieldShowHintBorder: Driver<Bool> = self.boardNameTextFieldShowHintBorder(flowStep)
...
return Output(
  boardZeroContentOffset: boardZeroContentOffset,
  boardViewZoomOut: boardViewZoomOut,
  overlayLeadingConstraintConstant: overlayLeadingConstraintConstant,
  overlayFadeAnimations: overlayFadeAnimations,
  boardNamingOverlayButtonEnabled: boardNamingOverlayButtonEnabled,
  ...
```

DRIVE

```
...
/* Board name */
output.boardNameDisplayText.drive(boardNameTextField.rx.text)
output.focusBoardNameTextField.drive(boardNameTextField.rx.isFirstResponder)
output.boardNameTextFieldShowActiveBorder.drive(boardNameTextField.rx.showBorder)
output.boardNameTextFieldShowHintBorder.drive(boardNameTextField.rx.showHintBorder)
output.boardNameTextFieldSelectAllText.drive(boardNameTextField.rx.selectAll())
...
/* Compact/Regular UI */
output.compactUIHidden.drive(self.compactUI.rx.isHidden)
output.regularUIHidden.drive(self.regularUI.rx.isHidden)
output.boardLeadingConstant.drive(self.boardViewLeadingConstraint.rx.constant)
output.boardBottomConstant.drive(self.boardViewBottomConstraint.rx.constant)
...
```

BENEFITS

» Stateless ViewModel

» Declarative ViewModel / ViewController

» Very easy to make UX Tweaks

» Bugs very localized

» Testing

TEST: MOCK UI WITH SUBJECTS

```
override func setUp() {
  super.setUp()
  self.boardNameText = PublishSubject<String>()
  self.boardNameEditingDidBegin = PublishSubject<Void>()
  self.boardNameEditingChanged = PublishSubject<Void>()
  self.boardNameEditingDidEnd = PublishSubject<Void>()
  self.boardNameEditingDidEndOnExit = PublishSubject<Void>()
  ...
  self.output = self.viewModel.transform(input)
}
```

TEST

```
func testHintBorderShownOnBoardNameTextField() {
    var shown = true

    output.boardNameTextFieldShowHintBorder.drive(onNext: { (isShown) in
        shown = isShown
    }).disposed(by: disposeBag)

    self.boardNameOverlayGoButtonTap.onNext(())
    self.boardNameEditingDidBegin.onNext(())

    XCTAssertFalse(shown)
}
```

THE CODE THAT CHANGED MY MIND

```
let output = viewModel.transform(input)

func transform(_ input: Input) -> Output {
    let flowStep: Observable<FlowStep> = self.flowStep(input)
    // ...
}
```

THANKS / Q&A

» Slides and links at loufranco.com/rx-presentation

» RxSwift + MVVM: how to feed ViewModels
by Martin Moizard
https://medium.com/blablacar-tech/rxswift-mvvm-66827b8b3f10

» Trello iOS Assisted Onboarding
https://github.com/trello/trello-ios-assisted-onboarding

» RxMarbles

PHOTO CREDITS

» Bank Phrom (photo)

» Waldemar Brandt (photo)

» Ken Treloar (photo)