Demystifying Reactive Programming

Let's Build a Reactive Programming Library

Curt Clifton
The Omni Group
Twitter: @curtclifton
Web: curtclifton.net

What is Reactive Programming?

Arrays with **map** and **filt**

Array<Int>

[0, 1, 2, 1, 1, 0, 3]

```
enum Emoji: In {
map { i in Emoji.init(rawValue: i) }
    case 🐮 = 0
    case 🐶 = 1
    cas    = 2
}
```

(Int) -> Emoji?

[🐮, 🐶, 🐋, 🐶, 🐶, 🐮, nil]

Array<Emoji?>

Arrays with map and filt

```
enum Emoji: Int {
    case 🐮 = 0
    case 🐶 = 1
    case 🐋 = 2
}
```

Array<Int>

[0, 1, 2, 1, 1, 0, 3]

filter { i in i <= 2 }

(Int) -> Bool

[0, 1, 2, 1, 1, 0]

Array<Int>

map { i in Emoji.init(rawValue: i) }

Array<Emoji?>

[🐮, 🐶, 🐋, 🐶, 🐶, 🐮]

flatmap is your Friend

```
enum Emoji: Int {
    case 🐮 = 0
    case 🐶 = 1
    case 🐋 = 2
}
```

[0, 1, 2, 1, 1, 0, 3]

Array<Int>

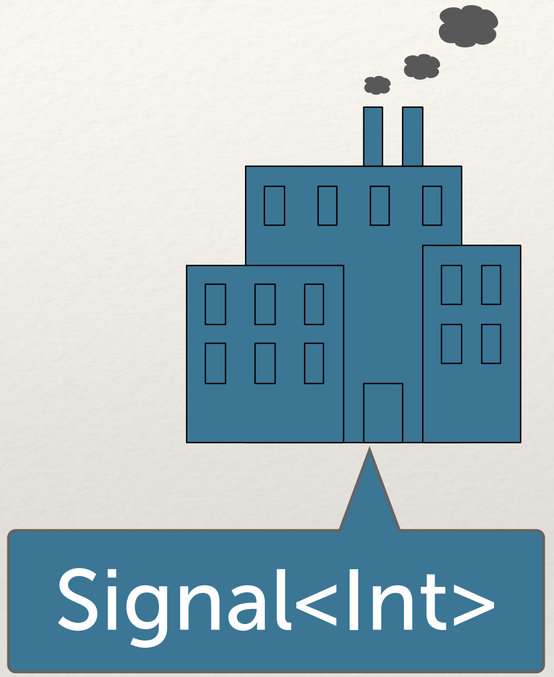
flatmap { i in Emoji.init(rawValue: i) }

(Int) -> Emoji?

[🐮, 🐶, 🐋, 🐶, 🐶, 🐮]

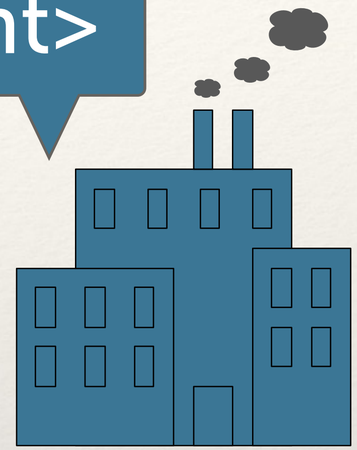
Array<Emoji>

Signals



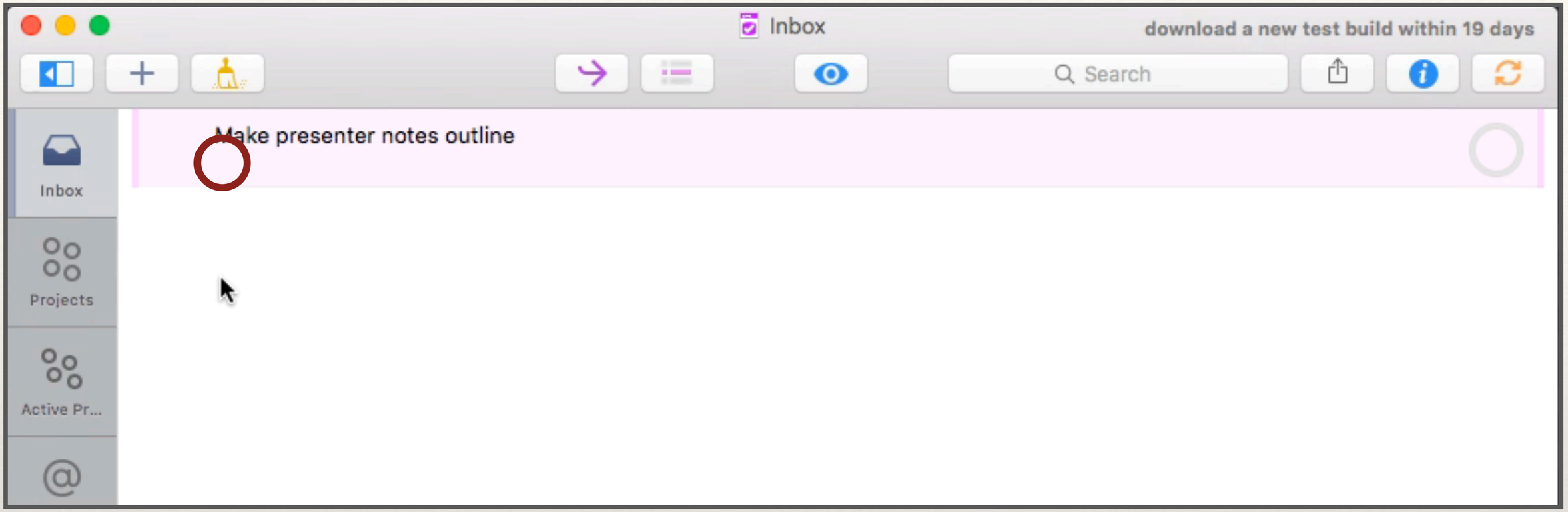
Signals with **flatmap**

Signal<Int>



flatmap { i in Emoji.init(rawValue: i) }

Signal<Emoji>



OmniFocus for Mac

You want the fields to do what?

What are the Inputs for Note Button Color?

- isPressingNoteIcon
- isWindowKey
- delayedIsMouseOverRow
- isMouseOverNoteIcon
- hasNote
- isNoteExpanded
- isRowEditing
- hasMetadataNextToNoteIcon
- isInitialUpdateComplete

Note Button Color in OmniFocus

```
ReactivePatternMatch *noteIconConditionalDisplayPattern =
    [ReactivePatternMatch patternForEnvironment:conditionEnvironment];
[noteIconConditionalDisplayPattern
    addPattern:@"!self.isInitialUpdateCellContentsComplete"
    result:@(OFIConditionalDisplayStyleHidden)];
[noteIconConditionalDisplayPattern
    addPattern:@"isPressingNoteIcon"
    result:@(OFIConditionalDisplayStylePressed)];
[noteIconConditionalDisplayPattern
    addPattern:@"self.isWindowKey && delayedIsMouseOverRow && isMouseOverNoteIcon"
    result:@(OFIConditionalDisplayStyleHoveredView)];
[noteIconConditionalDisplayPattern
    addPattern:@"self.hasNote || self.isNoteExpanded"
    result:@(OFIConditionalDisplayStyleMetadata)];
[noteIconConditionalDisplayPattern
    addPattern:@"(self.isWindowKey && delayedIsMouseOverRow) || isRowEditing || hasMetadataNextToNoteIcon"
    result:@(OFIConditionalDisplayStylePlaceholder)];
[noteIconConditionalDisplayPattern
    setDefaultResult:@(OFIConditionalDisplayStyleHidden)];
ReactiveObjectCoalescer *noteIconConditionalDisplaySource =
    [ReactiveObjectCoalescer coalescerForObjectValueSource:noteIconConditionalDisplayPattern];
[self.noteDisclosureAccessoryButton
    setConditionalDisplayStyleConditionSource:noteIconConditionalDisplaySource];
```

Goals

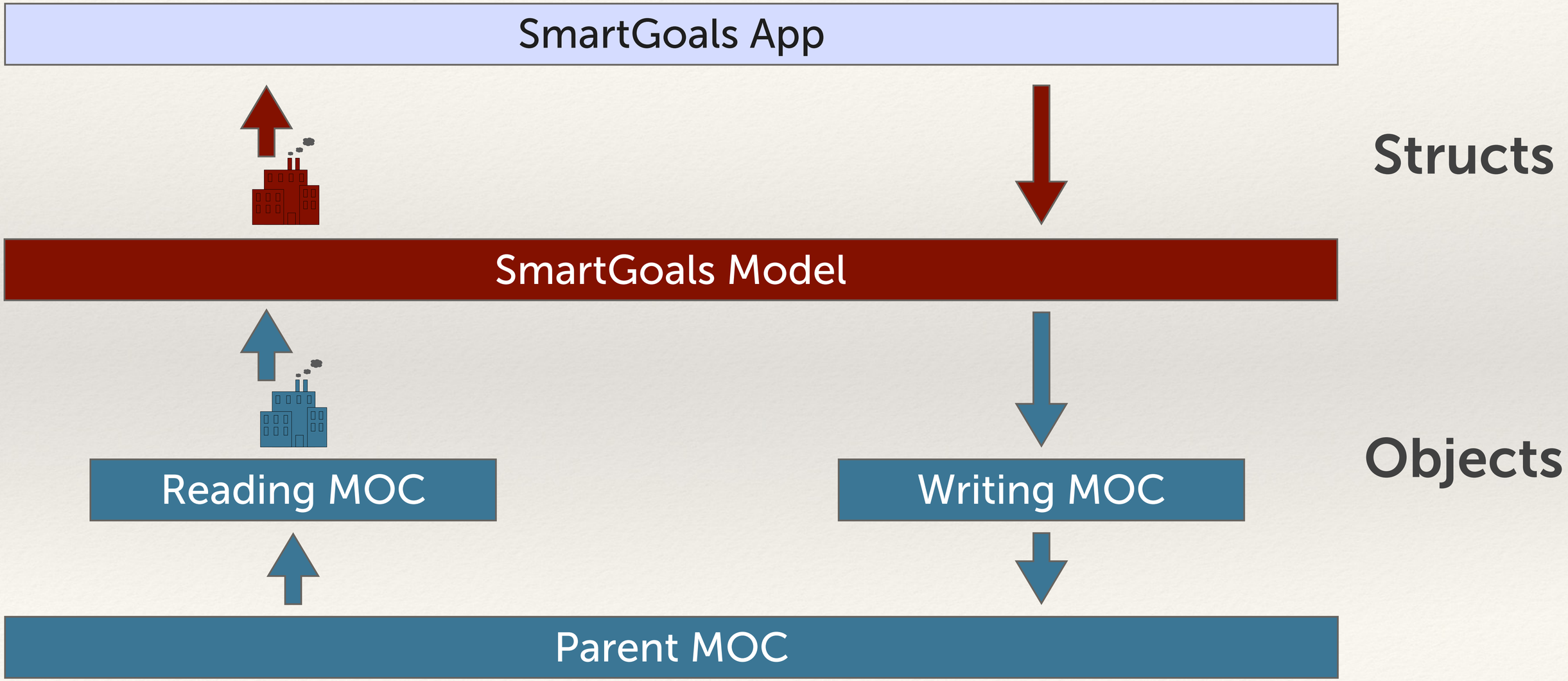
- Demonstrate that Reactive Programming is not magic
- Help you think about the risks and rewards of Reactive Programming

Sample App

Let's Build It

- Basic Signals
- Mapping
- Reactive Core Data
- Queues
- Controls

Reactive Core Data



Let's Build It

- Basic Signals
- Mapping
- Reactive Core Data
- Queues
- Controls

Why Avoid Reactive Programming?

- Learning curve (initially and for new team members)
- Dependency on third party framework
- “Don’t fight the frameworks”
- Challenging to debug
- Widely perceived as unreadable

“I’ve yet to see reactive code that didn’t look like Perl shoved in a blender, even though I like the idea in theory.” — @pilky

Why Use Reactive Programming?

- Declarative description of data flow
- Reduces *explicit* state
- Create the future?
- Cross platform APIs with RxSwift and friends
- Don’t have to go all-in

Reactive Programming

- Reactive Programming is not magic
- Comes with some risk
- A useful tool in your kit

Where Next?

- ReactiveCocoa vs. RxSwift (Ash Furrow)
- ReactiveCocoa vs. RxSwift (Rui Peres)
- The Introduction to Reactive Programming You've been Missing (Andre Staltz)
- The Reactive Revolution of Swift (Junior Bontognali)
- The Non-Reactive Solution and related follow-up (Brent Simmons)

Thanks!

Slides & code available from curtclifton.net

Curt Clifton
The Omni Group
Twitter: @curtclifton
Web: curtclifton.net