ALL! THE! NEW! THINGS!

USING MVVM, FLOW-COORDINATORS, AND RXSWIFT TO CREATE A COMPLETELY DYNAMIC UI

@EL_IS_FOR_ELLEN - SHE/HER

WHAT LIES AHEAD...

- BRIEF INTRODUCTION - WHO AM I? WHY AM I HERE?
- OVERVIEW OF MVVM, RXSWIFT, AND COORDINATORS
- USE CASES + AVOID CASES
- UPS AND DOWNS
- QUESTIONS AND DISCUSSION

@EL_IS_FOR_ELLEN - SHE/HER

WHO AM I, ANYWAYS?

- IOS DEVELOPER
- BACKGROUND IN ART AND EDUCATION
- HOCKEY PLAYER
- CAT ENTHUSIAST

@EL_IS_FOR_ELLEN - SHE/HER

I'm Ellen. In a former life I taught high school students art and design. I switched careers through an apprenticeship that lead to a full time iOS role and ended up working on a project that used MVVM, RxSwift, and Coordinators. I then worked at a major automotive company which also utilizes RxSwift and MVVM. And now I work at Thomson Reuters where for the most part none of these are implemented in the projects I work on.

I'm going to talk about my experience working on projects that use these tools as well as on boarding to a more vanilla MVC iOS project and the perspective these experiences have given me. Everything I say here is my own and not tied to TR.

THE PROBLEM

@EL_IS_FOR_ELLEN - SHE/HER

Make an app that helps users walk their customers through an application. The app will guide them through the questions required based on the specific situation so our user can focus on good customer service.

TRICKY REQUIREMENTS ✅

- FORMS WITH DYNAMIC FIELDS
- NAVIGATION CHANGES BASED ON SERVER RESPONSE
- NAVIGATION DEPENDS ON USER INPUT
- SAVE AS YOU GO
- ALLOW FOR SWITCHING BETWEEN PLATFORMS
- DUMBEST FRONT ENDS POSSIBLE

@EL_IS_FOR_ELLEN - SHE/HER

Aside from a basic tabbed view controller, we didn't know what we'd be presenting or when. Everything depended on what we received from the back end. Dumb UI actually means sophisticated logic.

MVVM

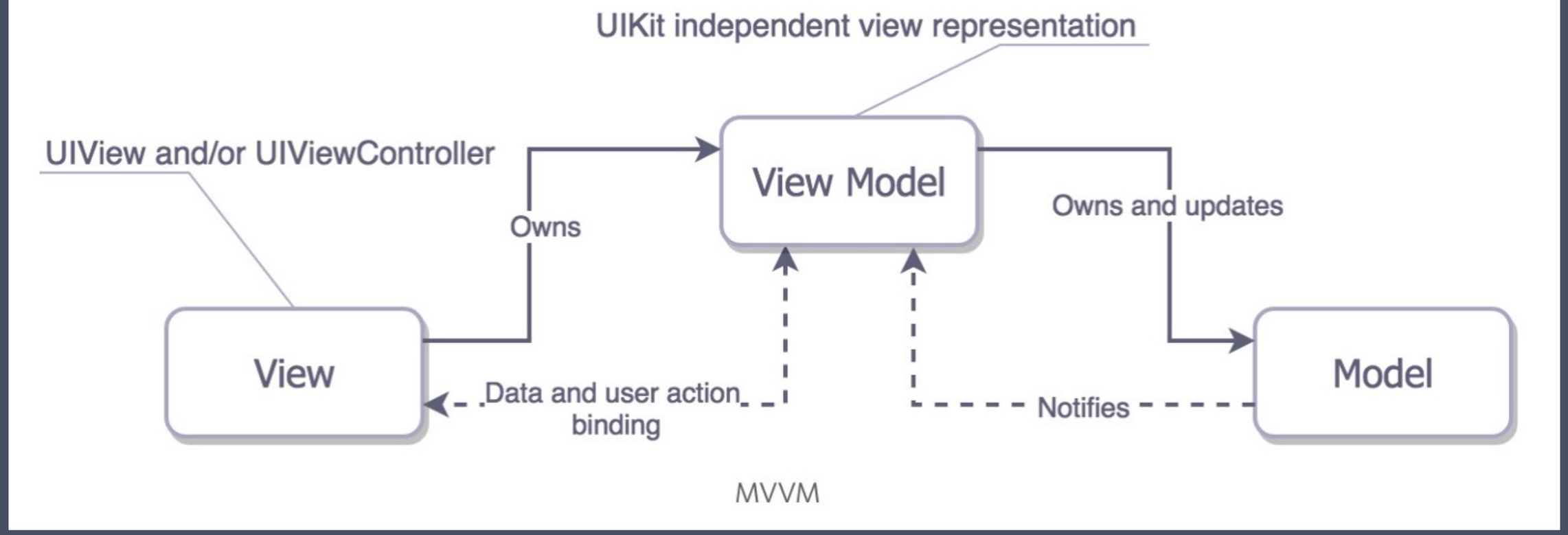
@EL_IS_FOR_ELLEN - SHE/HER

Let's jump in with the architectural pattern MVVM

MODEL VIEW VIEWMODEL

@EL_IS_FOR_ELLEN - SHE/HER

MVVM stands for Model View ViewModel, but what does that mean?



MVVM

FOR A MORE IN DEPTH LOOK AT MVVM AND RXSWIFT CHECKOUT THIS TUTORIAL FROM NAVDEEP SINGH.

MVVM keeps all the pieces of your code decoupled which makes them easier to test.

PROS

- LOW OVERHEAD FOR IMPLEMENTATION
- TESTABILITY
- LEARNING CURVE IS NOT EXTREME
- DECOUPLED CODE ALSO IMPROVES SAFETY
- LOTS OF RESOURCES

CONS

- CAN BE OVERKILL
- VIEWMODEL CAN DO TOO MANY THINGS
- NO BUILT IN SOLUTION FOR DATA BINDING

Many of examples of different implementations. ViewModel becomes the new MVC.

RXSWIFT

@EL_IS_FOR_ELLEN - SHE/HER

One of many Reactive Programming options. A whole new way of thinking.

STATE

RX IS THAT SWEET SPOT BETWEEN FUNCTIONAL AND IMPERATIVE WORLDS. IT ENABLES YOU TO USE IMMUTABLE DEFINITIONS AND PURE FUNCTIONS TO PROCESS SNAPSHOTS OF MUTABLE STATE IN A RELIABLE COMPOSABLE WAY.

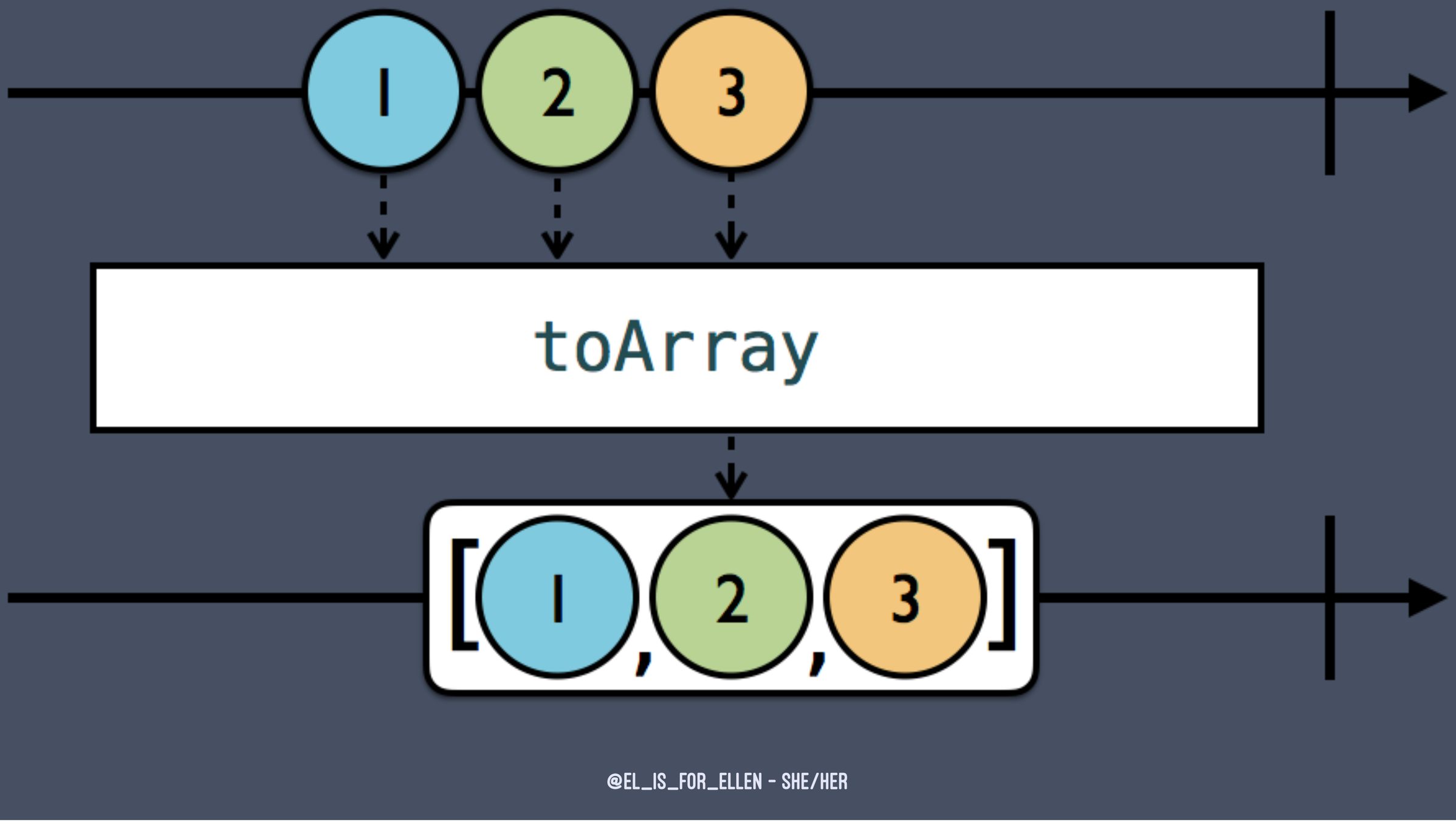
– RXSWIFT DOCUMENTATION

@EL_IS_FOR_ELLEN - SHE/HER

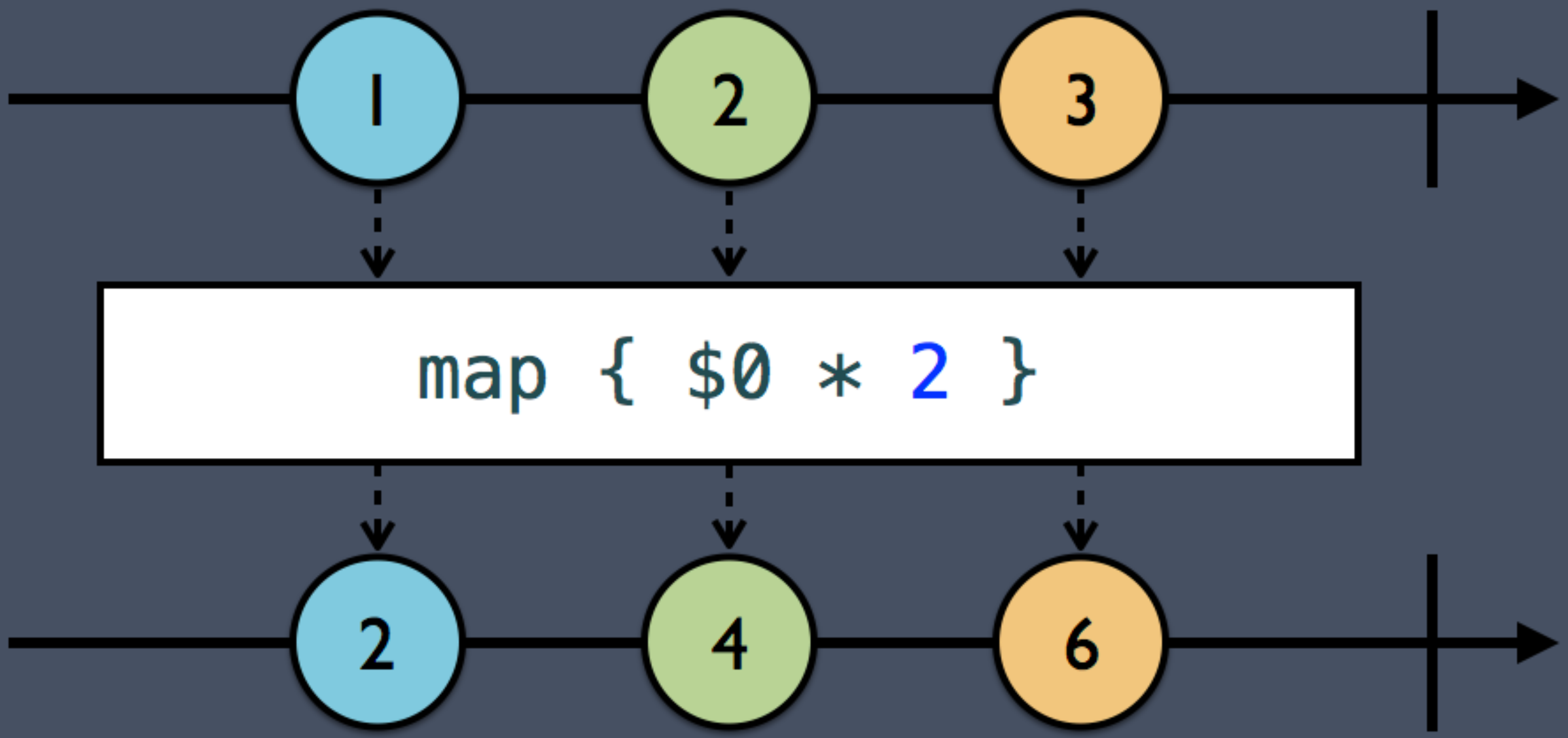
OBSERVE 👀 TRANSFORM 🔀

@EL_IS_FOR_ELLEN - SHE/HER

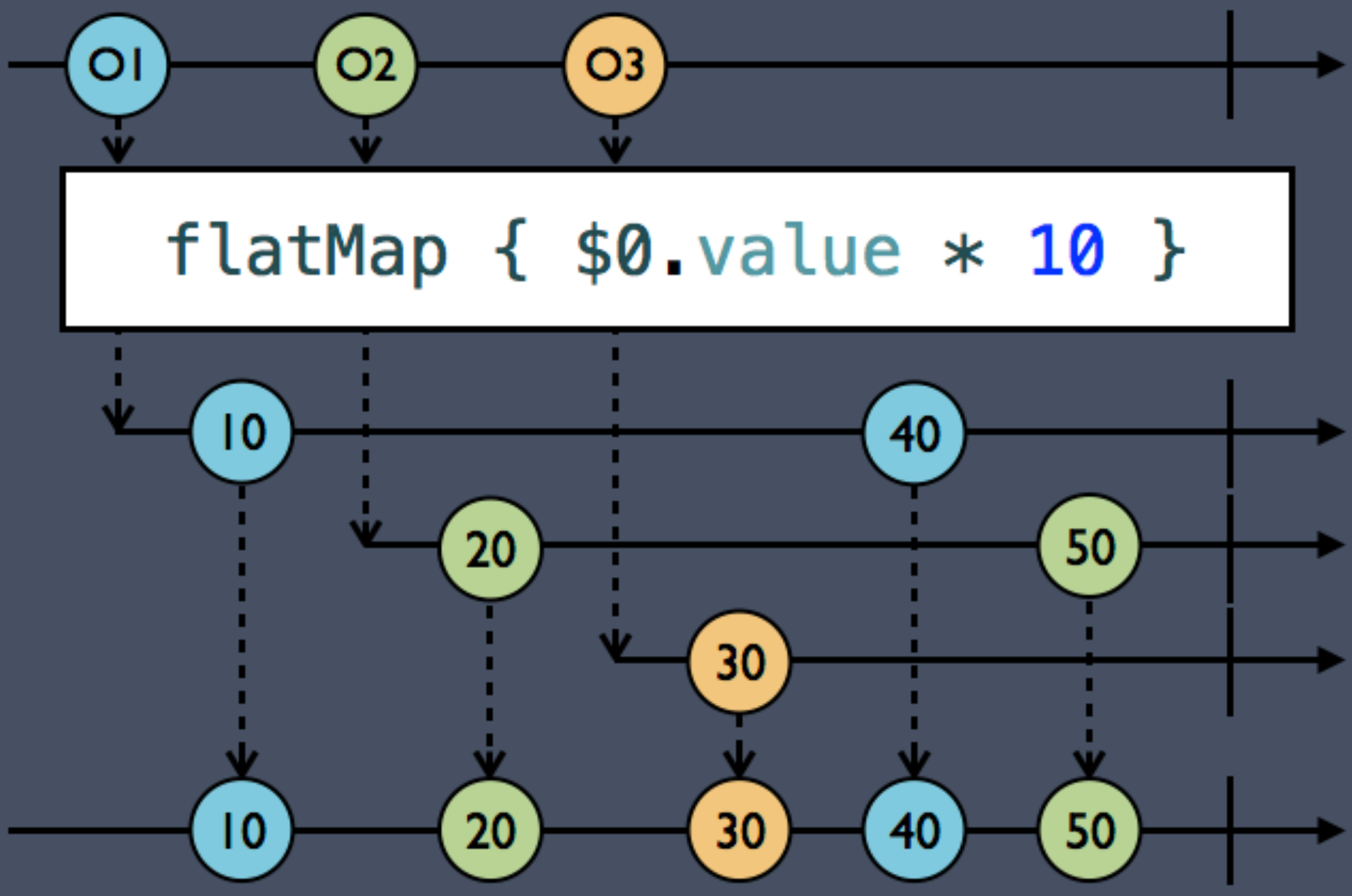
The bulk of Rx can be summed up with observing and transforming. Everything is a sequence. KVO and Notifications.



Observe what comes in. Grab everything that comes in and transform it to an array.



Map everything we observe and transform it.



Source observable on top, target observables in the middle, and the subscriber at the bottom. RxSwift allows for receiving and transforming data in complicated ways. Marble diagrams are super handy to help understand the sequences.

INSTEAD OF UITABLEVIEWDELEGATE METHODS

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) → UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "UITableViewCell", for: indexPath)
    let item = items[indexPath.item]
    cell.textLabel.text = item.name
    cell.image = item.image
    cell.subtitle.text = item.subtitle

    return cell
}
```

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) → Int {
    return items.count
}
```

RXSWIFT COMBINES AND SIMPLIFIES

```
viewModel
    .rows
    .bind(to: resultsTableView.rx.items(cellIdentifier: "SearchCell", cellType: SearchCell.self)) { (_, viewModel, cell) in
        cell.title = viewModel.title
        }
    .disposed(by: disposeBag)
```


Data binding is one of the most powerful aspects of RxSwift. Here's an example of code we've all written numerous times but in Rx.

PROS

- DATA BINDING
- ASYNCH SOLUTION
- CROSS PLATFORM KNOWLEDGE
- MARBLE DIAGRAMS!

@EL_IS_FOR_ELLEN - SHE/HER

Legit solves some problems if you need to create a dynamic app. Like learning OOP or other paradigm the learning you do with RxSwift can be taken with you and applied to reactive projects in other languages.

CONS

👎

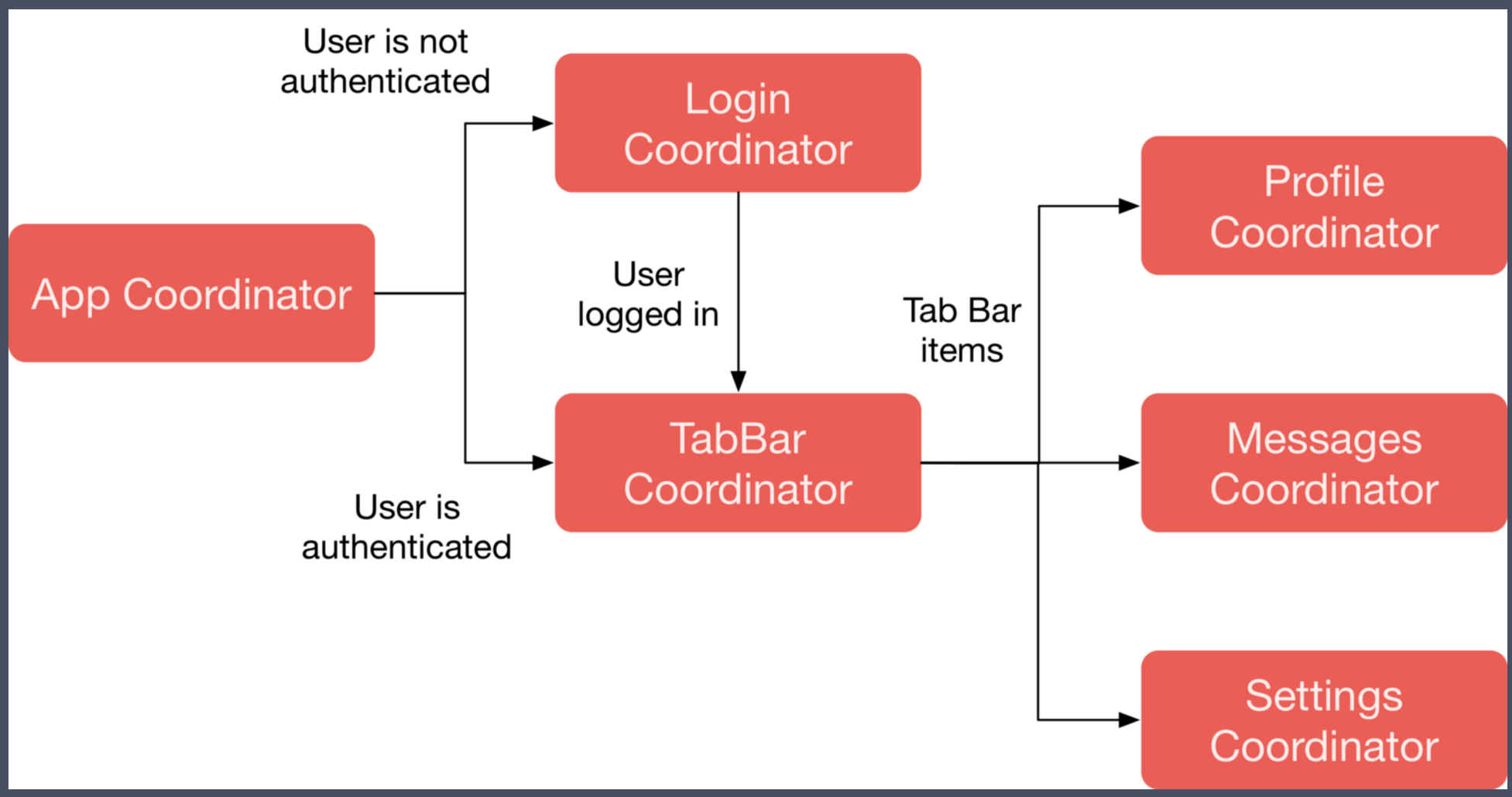
- ONBOARDING IS COMPLICATED AND LENGTHY 😭
- LIMITED RESOURCES
- HARD TO READ
- TESTING IS MORE COMPLICATED

Many of examples of different implementations. ViewModel becomes the new MVC. It's the dumbest you've felt in a long time. Plus Combine is coming.

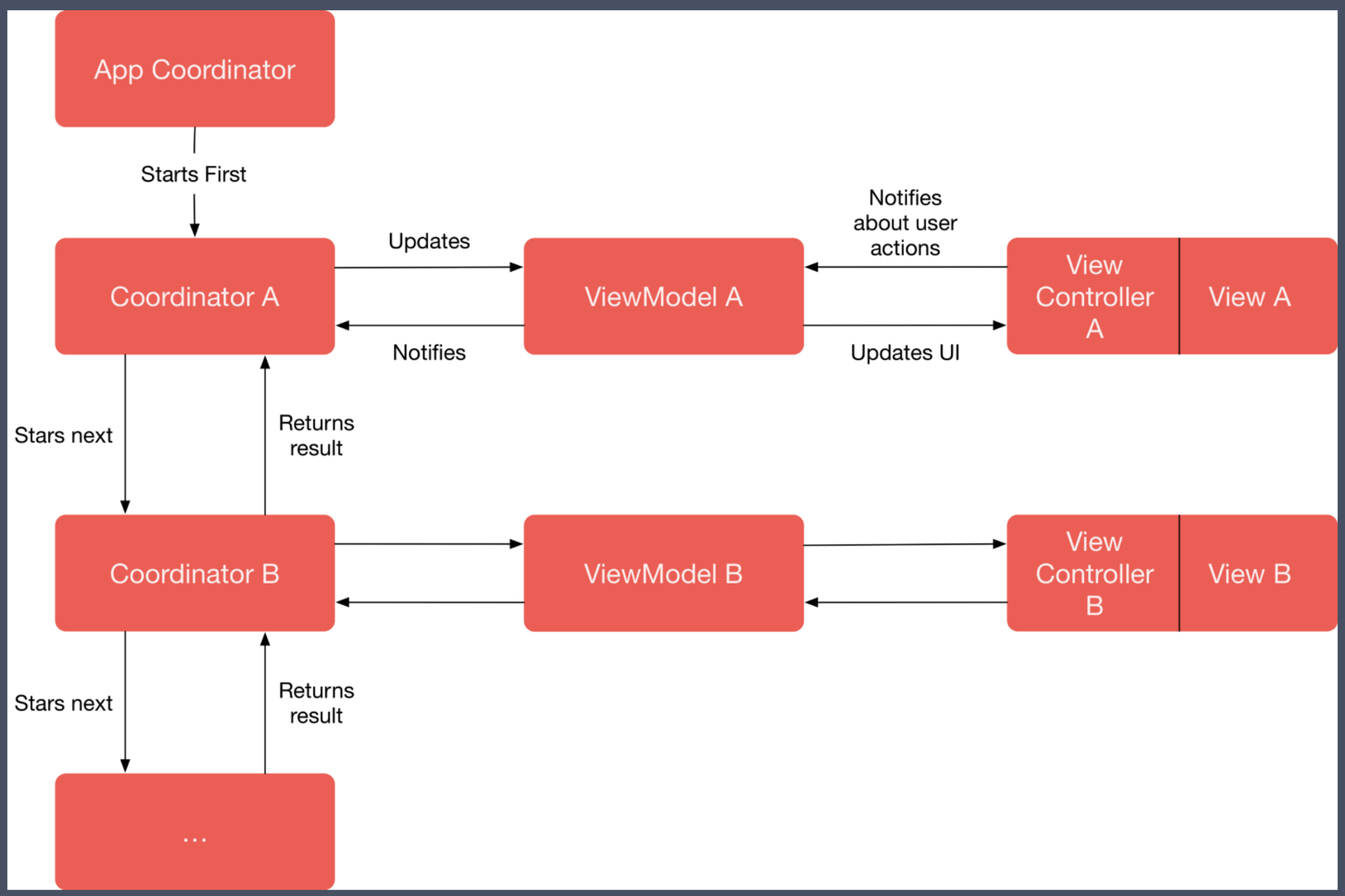
COORDINATORS

@EL_IS_FOR_ELLEN - SHE/HER

Design pattern that controls the flow of the app. Controls the presentation and dismissal of all view controllers.



FOR A MORE IN DEPTH LOOK AT COORDINATORS, MVVM, AND RXSWIFT CHECKOUT THIS BLOG POST ON HACKERNOON.



FOR A MORE IN DEPTH LOOK AT COORDINATORS, MVVM, AND RXSWIFT CHECKOUT THIS BLOG POST ON HACKERNOON.

Helps define the use cases and flow of the app. Handles dependency injection.

PROS

- CAN BE USED WITH MVC OR MVVM
- DEPENDENCY INJECTION
- MAKES VIEW CONTROLLERS MORE REUSABLE
- MOVING A VIEW CONTROLLER IN THE FLOW IS PAINLESS

@EL_IS_FOR_ELLEN - SHE/HER

Versatile, had to move a view controller from one flow to another did it by moving 2 lines of code.

CONS

- ADDED COMPLEXITY - DELEGATING UP AND DOWN TREES
- TONS OF OVERHEAD FOR A SMALL APP

Some ramp up for first looking at a code base to understand flow when delegating up and down a lot. Way too much overhead for a simple app with a simple flow that doesn't change.

USE CASES ABUSE CASES

- WHAT DOES MY APP DO?
- WHAT WILL IT BECOME?
- DOES THIS NEED TO BE REACTIVE?

@EL_IS_FOR_ELLEN - SHE/HER

Apps that are not dynamic will not benefit from RxSwift. Flows that are simple and linear may not need coordinators, though you could benefit dependency injection.

MY EXPERIENCE

- GREENFIELD PROJECT
- ONBOARDING TO EXISTING PROJECT
- VANILLA MVC PROJECT

SOURCES AND RESOURCES 💻

- GITHUB.COM/REACTIVEX/RXSWIFT
- RAYWENDERLICH.COM/688-INTRODUCING-RXSWIFT-REACTIVE-PROGRAMMING-WITH-SWIFT
- RAYWENDERLICH.COM/158-COORDINATOR-TUTORIAL-FOR-IOS-GETTING-STARTED
- ACADEMY.REALM.IO/POSTS/MOBILIZATION-LUKASZ-MROZ-MVVM-COORDINATORS-RXSWIFT/