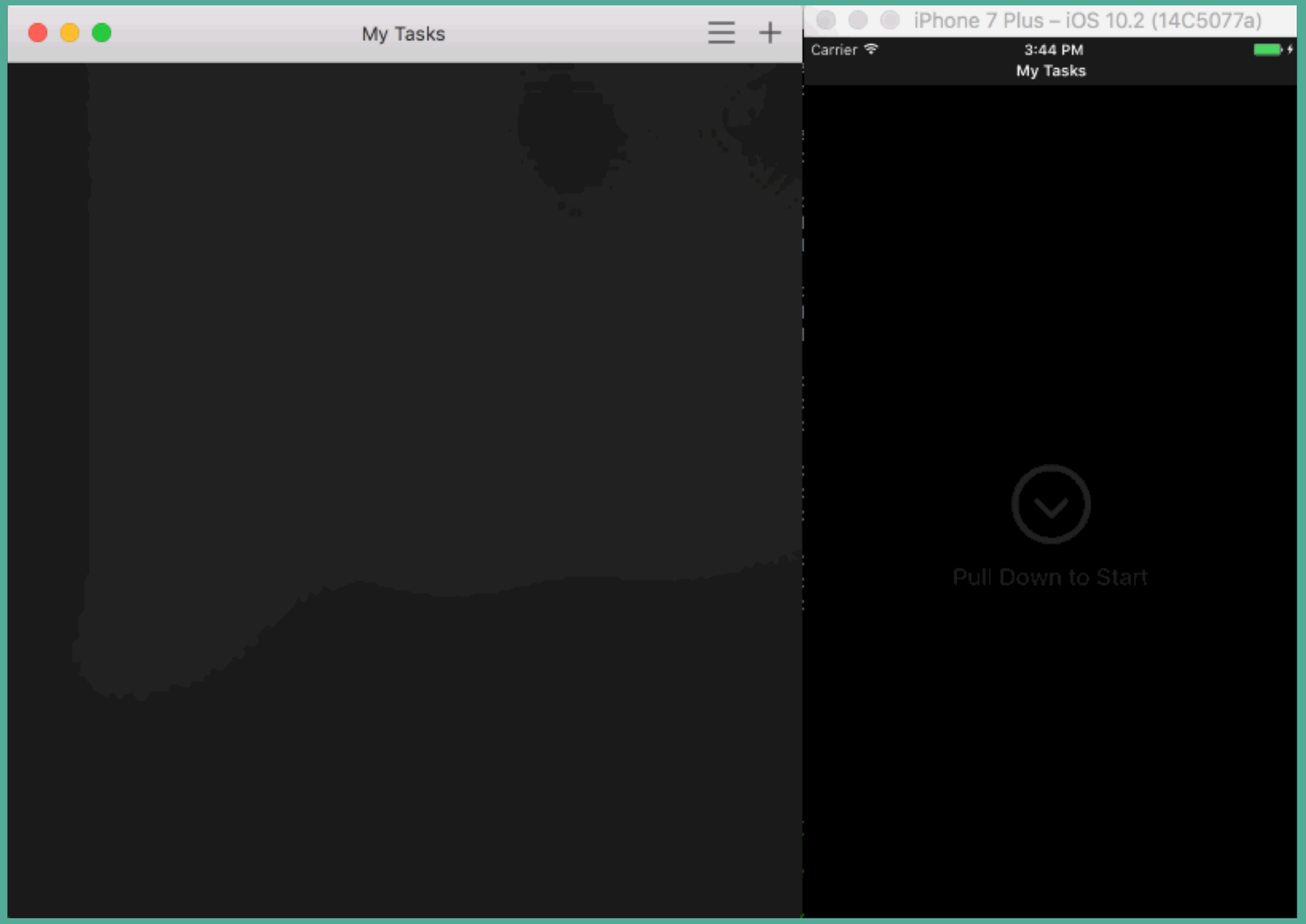
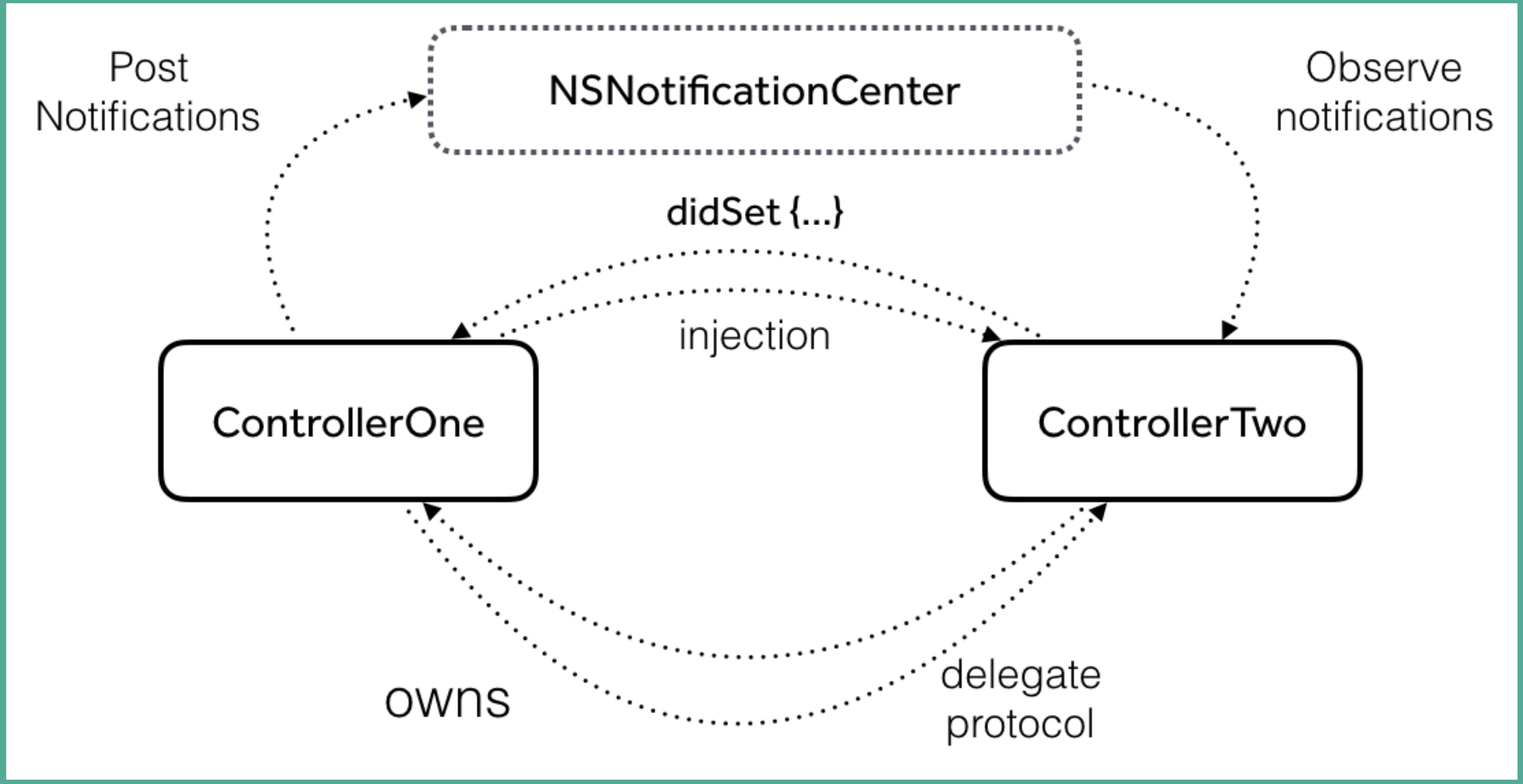
Mastering Realm Notifications

JP Simard, @simjp, realm.io


My Tasks
iPhone 7 Plus – iOS 10.2 (14C5077a)
Carrier
3:44 PM
My Tasks
Pull Down to Start

Cocoa

Messaging


Post
Notifications
NSNotificationCenter
Observe
notifications
didSet {...}
injection
ControllerOne
ControllerTwo
owns
delegate
protocol

Reactive Programming

Reactive

!= Functional

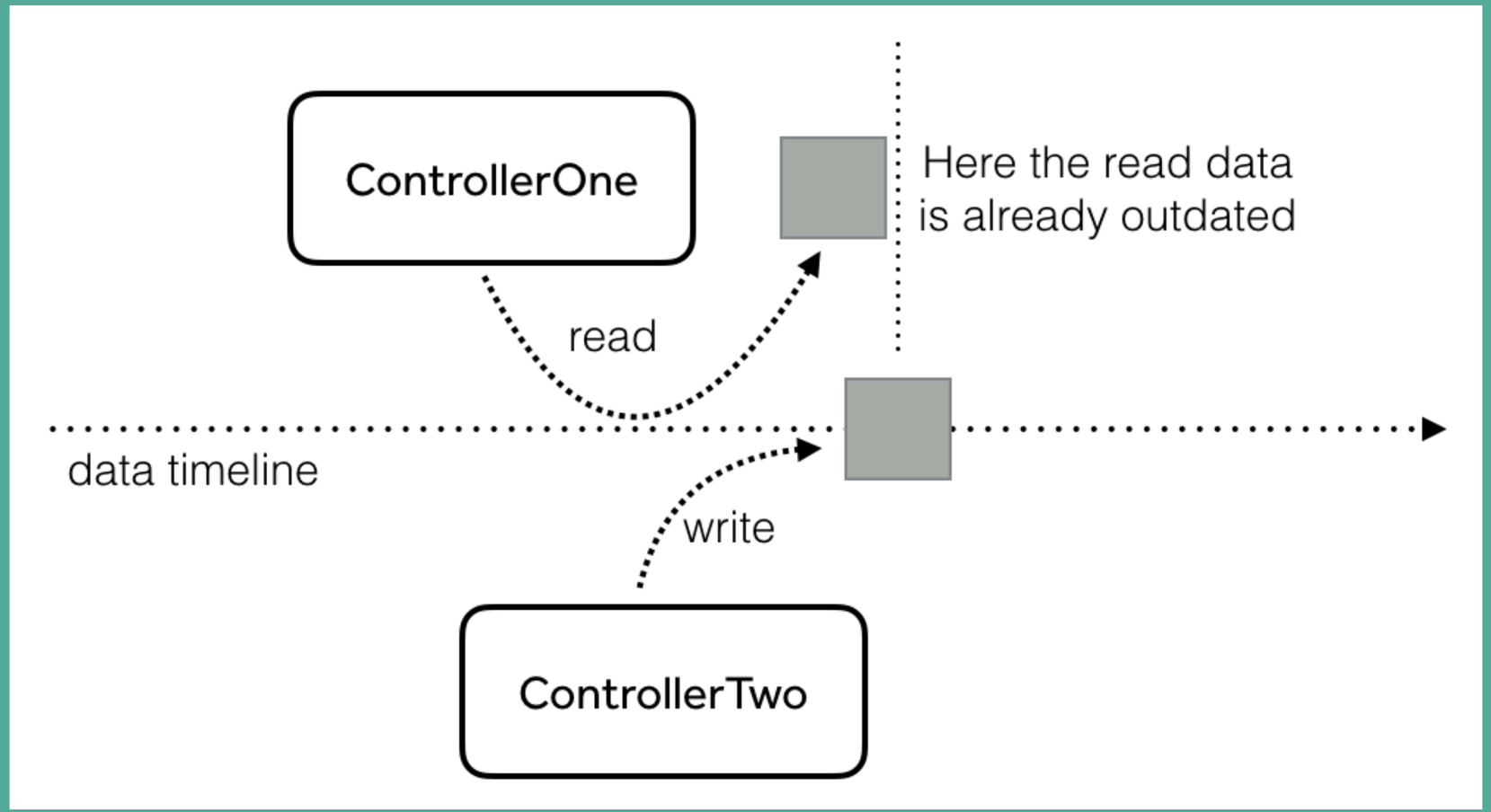
Peter Steinberger @steipete · 12 Aug 2015

Tried GitHub Desktop. Freezes after setup complete, stack trace somewhere deeeeeeeeep in ReactiveCocoa.

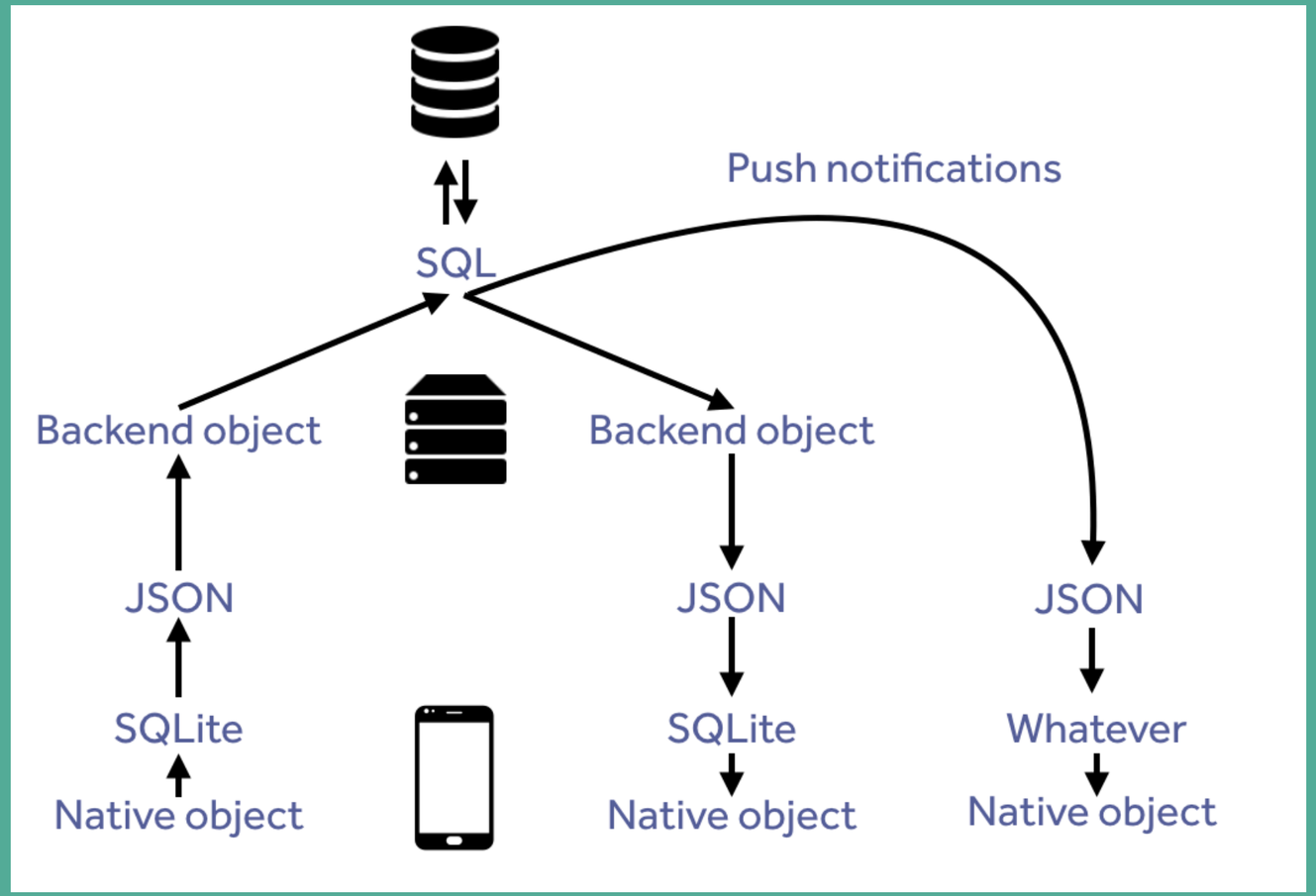
13 60 68 •••

Reactive Programming just means reacting to changes & minimizing the explicit messaging about changes.

It's about avoiding stale data

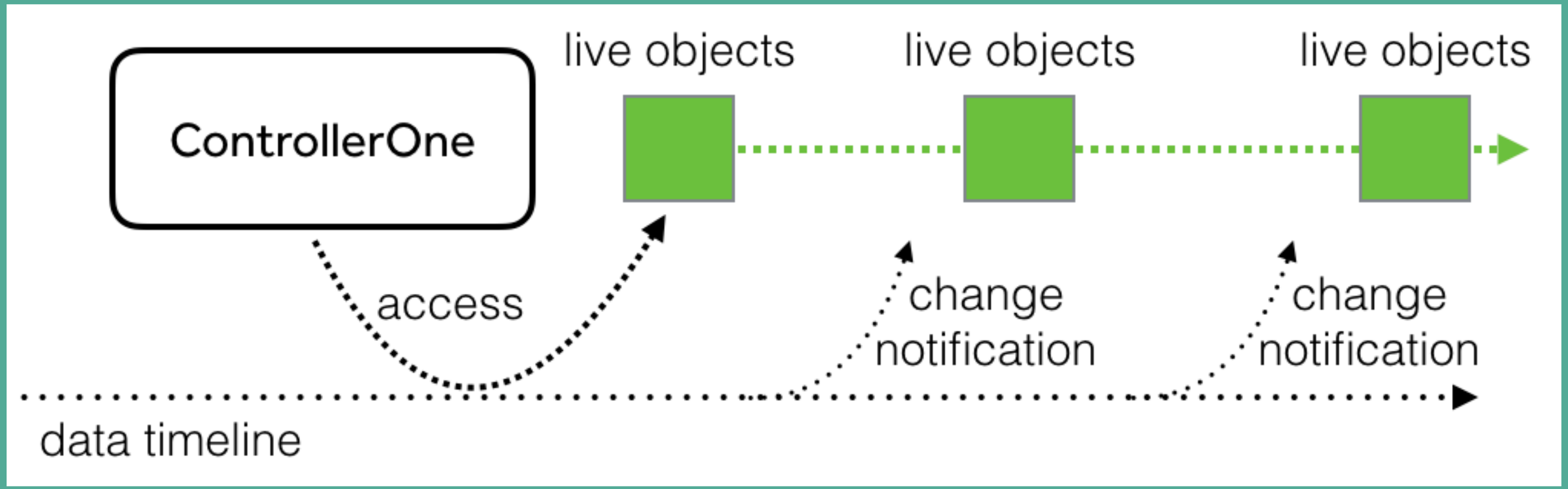


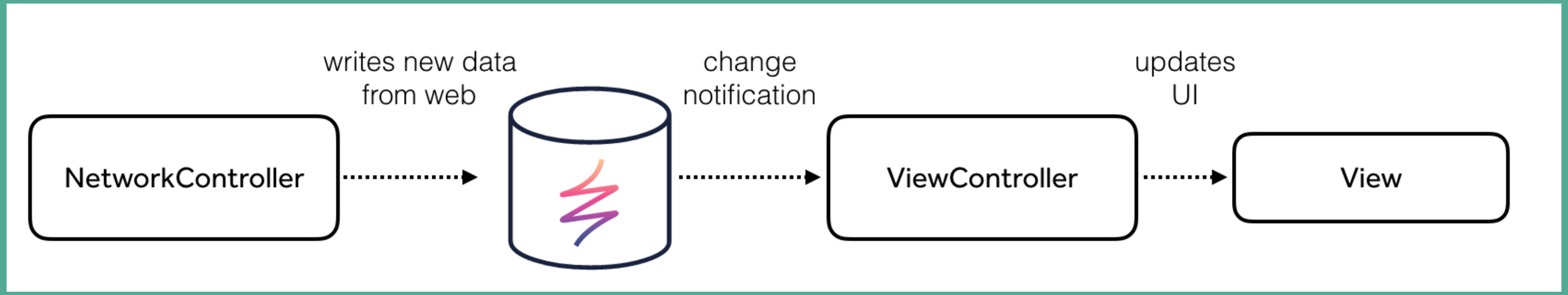
It's about avoiding duplicate code paths



Avoid different code paths

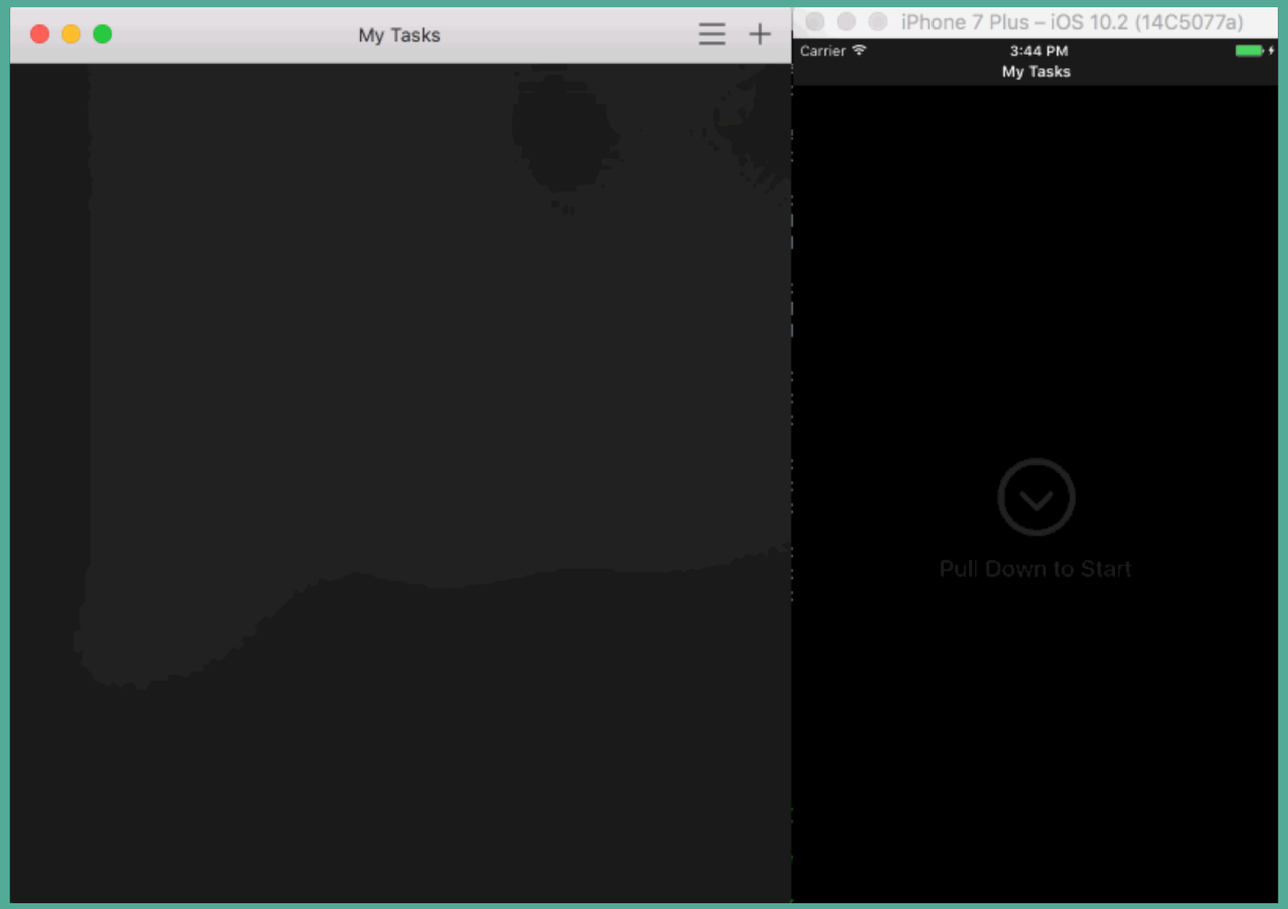
- network
- database
- user action
- push notification
- system alerts
- other processes


live objects
live objects
live objects
ControllerOne
access
change
notification
change
notification
data timeline


writes new data
from web
change
notification
updates
UI
NetworkController
ViewController
View

Intro to Realm Notifications

RealmTasks


My Tasks
iPhone 7 Plus – iOS 10.2 (14C5077a)
Carrier
3:44 PM
My Tasks
Pull Down to Start

Types of Realm Notifications

- Realm Notifications
- Collection Notifications
- KVO
- Upcoming APIs...

Realm

Notifications

Realm Notifications

```
let realm = try Realm()
let token = realm.addNotificationBlock { notification, realm in
    viewController.updateUI()
}

// Later...
token.stop()
```

Realm Notifications

- **Use when you want to react to *everything* that happens**
- **Asynchronously delivered**
- **`Realm.refresh()` waits for these notifications to be delivered**
- **There's usually a better way**

Notification Token

- How Realm knows how long to deliver notifications for.
- Must hold a strong reference.
- Best to call `stop()` when you no longer want to be notified.
- Logs "released without unregistering a notification" if deallocated without calling `stop()`. Harmless but the best way we know to inform

Collection

Notifications

Collection Notifications

```
let realm = try Realm()
let results = realm.objects(Item.self) // Auto-Updating Results
let token = results.addNotificationBlock(tableView.applyChanges)
```

Collection Notifications

```
extension UITableView {
  func applyChanges<T>(changes: RealmCollectionChange<T>) {
    switch changes {
      case .initial: reloadData()
      case .update(let results, let deletions, let insertions, let updates):
        let fromRow = { (row: Int) in return IndexPath(row: row, section: 0) }

        beginUpdates()
        insertRows(at: insertions.map(fromRow), with: .automatic)
        reloadRows(at: updates.map(fromRow), with: .automatic)
        deleteRows(at: deletions.map(fromRow), with: .automatic)
        endUpdates()
      case .error(let error): fatalError("\(error)")
    }
  }
}
```

Notifications may be coalesced! (grouped)

Notifications may be coalesced (grouped)

```
switch changes {
  case .initial: reloadData()
  case .update(let results, let deletions, let insertions, let updates):
    let fromRow = { (row: Int) in return IndexPath(row: row, section: 0) }

    beginUpdates()
    insertRows(at: insertions.map(fromRow), with: .automatic)
    reloadRows(at: updates.map(fromRow), with: .automatic)
    deleteRows(at: deletions.map(fromRow), with: .automatic)
    endUpdates()
  case .error(let error): fatalError("\(error)")
}
```

Synchronous vs Asynchronous Queries

`Results` **are queried synchronously if requested, asynchronously otherwise.**

Asynchronous Query

```
let realm = try Realm()
// Query never performed on the current thread
let results = realm.objects(Item.self)
let token = results.addNotificationBlock { _ in
  /* results available asynchronously here */
}
```

Synchronous Query

```
let realm = try Realm()
// Query performed synchronously to return first result
_ = results.first // <- query performed here
let token = results.addNotificationBlock { _ in
  /* results available asynchronously here */
}
```

Collection Notifications

- May be coalesced
- Generally delivered on runloop iterations
- Query upkeep always performed on background thread
- Notifications delivered on the source thread
- Can avoid *all* source thread overhead if used correctly

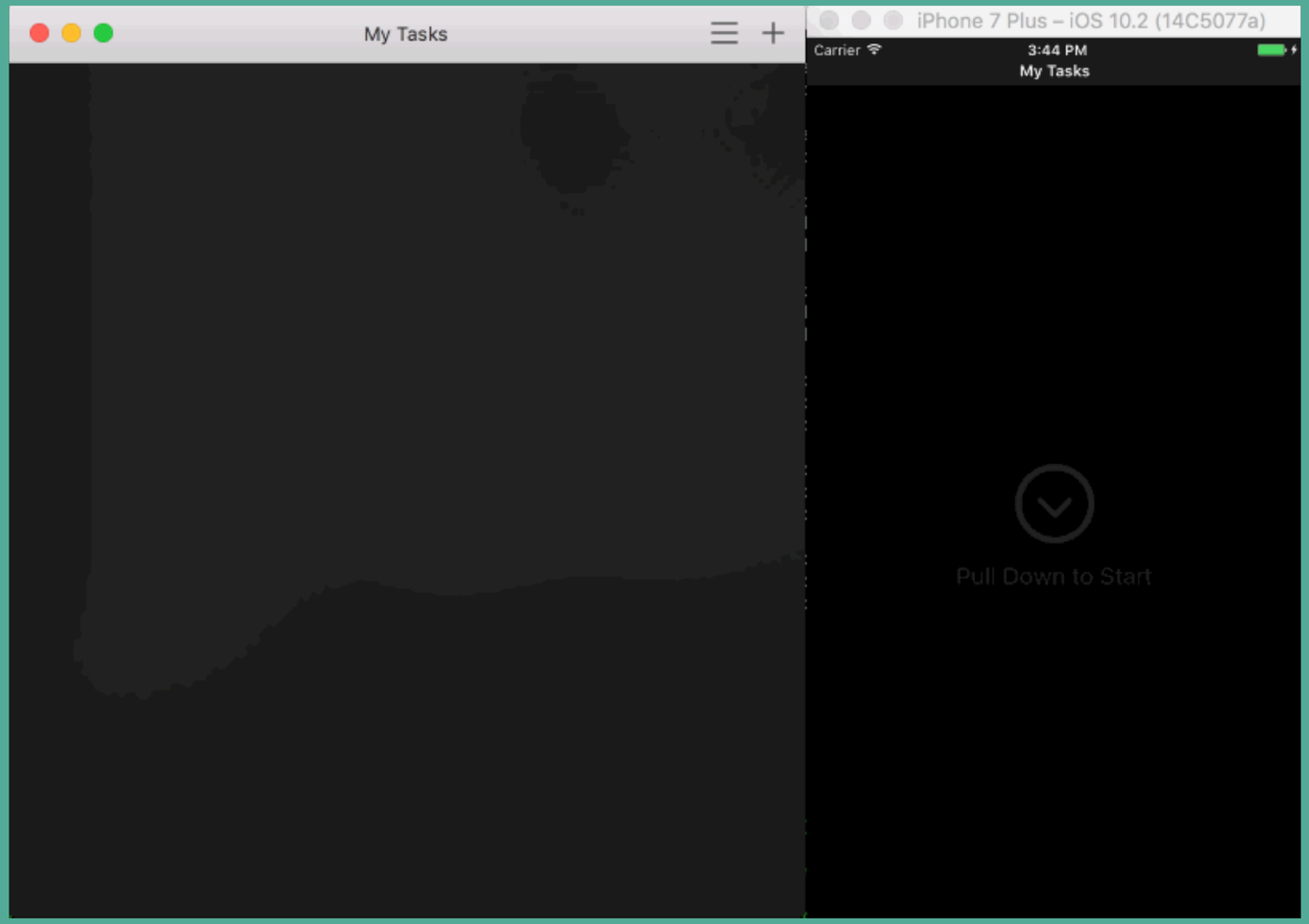
Interface-Driven

Writes

Interface-Driven Writes

- Changeset notifications are all about applying state diffs.
- State *must* be up to date for the diff to make sense.
- Some changes should be applied to the UI synchronously.

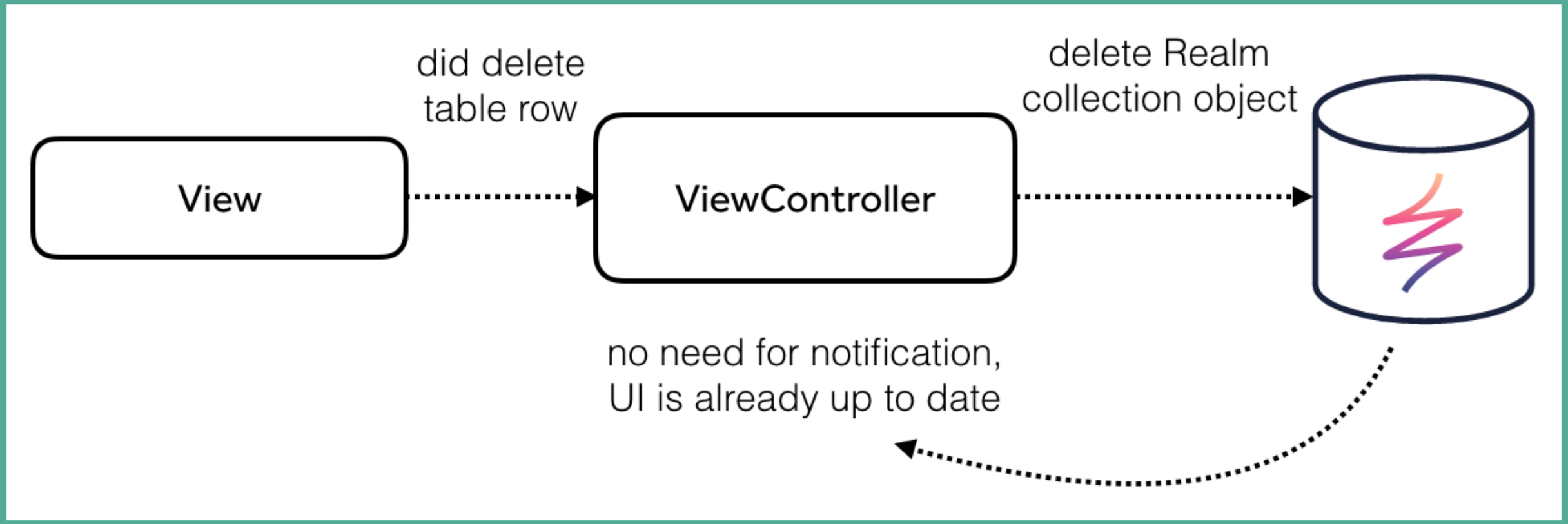
Like reordering rows.



Interface-Driven Writes

```
func viewDidLoad() {
  messages = realm.objects(Message.self)
  self.token = messages.addNotificationBlock(tableView.applyChanges)
}

func tableView(_ tableView: UITableView,
               commit editingStyle: UITableViewCellEditingStyle,
               forRowAt indexPath: IndexPath) {
  guard editingStyle == .delete else { return }
  realm.beginWrite()
  messages.removeAtIndex(indexPath.row)
  tableView.deleteRows(at: [indexPath], with: .automatic)
  realm.commitWrite(withoutNotifying: [self.token])
}
```


did delete
table row
delete Realm
collection object
View
ViewController
no need for notification,
UI is already up to date

Key-Value Observation

Key-Value Observation

- Enables Realm to work well with Cocoa APIs
- ReactiveCocoa
- Synchronously delivered
- Single property at a time

Objects as Messages

Inter-process data sharing/message passing

- Realm already has robust interprocess data sharing
- Could use to share data or notify main iOS app from app extension

Not limited to single-device

— Also for Realm Object Server communication
— And massively distributed apps

Objects as Messages

```
let managementRealm = try user.managementRealm()
try managementRealm.write {
  let permissionChange = SyncPermissionChange(realmURL: realmURL,
                                              userID: anotherUserID,
                                              mayRead: true,
                                              mayWrite: true,
                                              mayManage: false)
  managementRealm.add(permissionChange)
}
```

Objects as Messages

```
let collectionOfOne = managementRealm.objects(SyncPermissionChange.self)
                                     .filter("id = %@", permissionChange.id)
token = collectionOfOne.addNotificationBlock { notification in
  if case .update(let changes, _, _, _) = notification,
    let change = changes.first {
    // Object Server processed the permission change operation
    switch change.status {
    case .notProcessed: break // handle case
    case .success: break // handle case
    case .error: break // handle case
    }
    print(change.statusMessage) // contains error or message
  }
}
```

Object-Level Notifications

Object-Level Notifications

```
let realm = try Realm()
let jane = realm.objects(Person.self).filter("name == 'Jane'").first!
let token = jane.addNotificationBlock { change in
  switch change {
  case .change(let propertyChanges):
    for propChange in propertyChanges {
      print("'\(propChange.name)': \(propChange.oldValue) -> \(propChange.newValue)")
    }
  case .deleted: print("object was deleted")
  case .error(let error): print("notification couldn't be delivered: \(error)")
  }
}

// Later...
token.stop()
```

Object-Level Notifications

```
let realm = try Realm()
let state = realm.objects(AppState.self).first!
let token1 = state.addNotificationBlock(observingProperties: ["unreadCount", "status"],
                                        block: { change in
  // handle change and restrict to properties we care about
})

let token2 = state.addNotificationBlock(ignoringProperties: ["noisyProp"],
                                        block: { change in
  // handle change and avoid being notified for properties we don't care about
})
```

Object-Level Notifications

- Coming early 2017
- Asynchronously delivered
- Great for Objects-as-APIs/Messages and event handling

Resources

- Realm's docs on Notifications
  - realm.io/docs/swift#notifications
- Live Objects and Fine-Grained Notifications
  - realm.io/news
- RealmTasks
  - github.com/realm/RealmTasks
  - Removing workaround PR: RealmTasks#352
- RxRealm: RxSwiftCommunity/RxRealm

Thank You!

Questions?

JP Simard, @simjp, realm.io