探究响应式编程在 iOS 开发中的优势

iDev
靛青K

DevLink iDev全平台开发者大会

自我介绍

- 宋旭陶/靛青K
- SwiftGG 真灌水成员
- ENJOY iOS 工程师
- RxSwift & FRP 爱好者

今天的目的

- 让没有尝试过 RxSwift 的您产生兴趣
- 让已经有所尝试的您有新的灵感

以 RxSwift 为例

摘要

- 解决 Callback Hell
- RxSwift 原理
- 非网络请求形式的异步实践

Callback Hell

回调地狱

```
Alamofire.request(.POST, "login", parameters: ["username": "max",
"password": "insanity"])
    .responseJSON(completionHandler: { (firedResponse) -> Void in
        Alamofire.request(.GET, "myUserInfo" + firedResponse.result.value)
            .responseJSON(completionHandler: { myUserInfoResponse in
                Alamofire.request(.GET, "friendList" +
myUserInfoResponse.result.value)
                    .responseJSON(completionHandler: { friendListResponse
in
                        Alamofire.request(.GET, "blockedUsers" +
friendListResponse.result.value)
                            .responseJSON(completionHandler: {

                            })
                    })
            })
    })
```

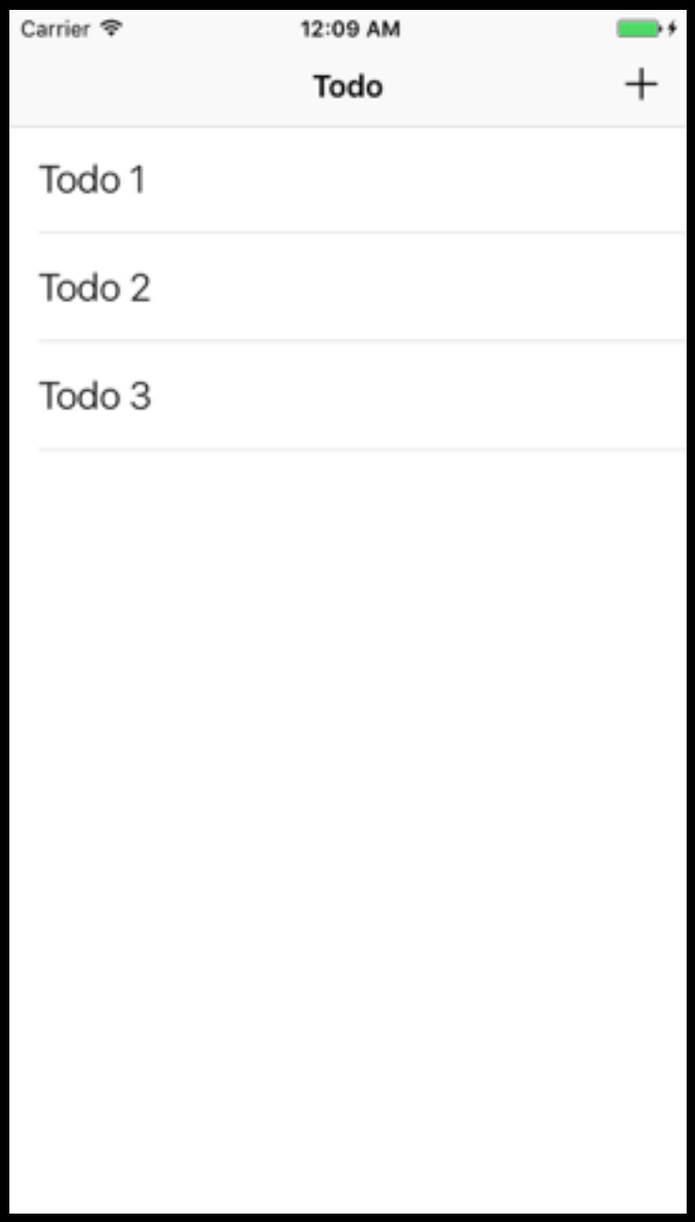
使用 RxSwift 解决回调地狱

```
Alamofire
    .request(.POST, "login", parameters: ["username": "max", "password":
"insanity"])
.flatMap { firedResponse in
    Alamofire.request(.GET, "myUserInfo" + firedResponse.result.value)
}
.flatMap { myUserInfoResponse in
    Alamofire.request(.GET, "friendList" + myUserInfoResponse.result.value)
}
.flatMap { friendListResponse in
    Alamofire.request(.GET, "blockedUsers" + friendListResponse.result.value)
}
```

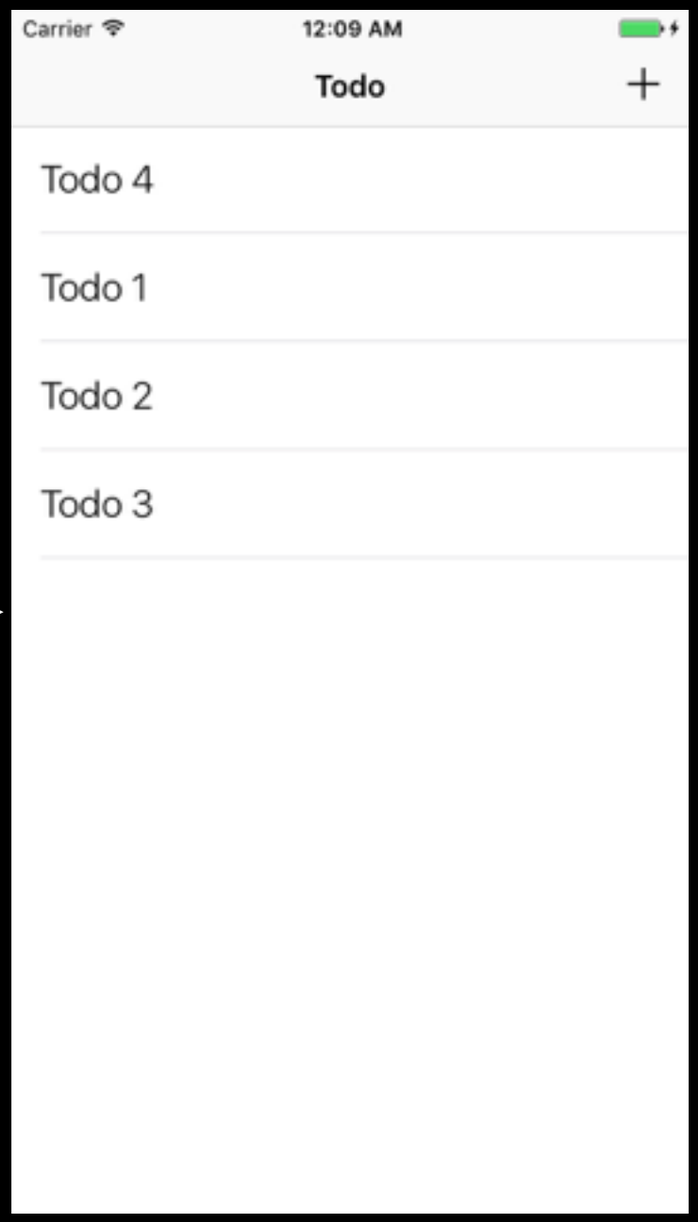
只用 RxSwift 解决网络回调吗?

RxSwift 基本原理

以 Todo 为例


Carrier
12:09 AM
Todo
Todo 1
Todo 2
Todo 3


Carrier
12:09 AM
Todo
Todo 1
Todo 2
Todo 3
添加新的 Todo
Todo 4
取消
好


Carrier
12:09 AM
Todo
Todo 4
Todo 1
Todo 2
Todo 3

一个数组的添加

Todo 1	Todo 2	Todo 3

Todo 4	+	Todo 1	Todo 2	Todo 3

Todo 4	Todo 1	Todo 2	Todo 3

```
var todoList = ["Todo 1", "Todo 2", "Todo 3"]
todoList.insert("Todo 4", at: 0)
```

真实场景

维护数据

```
var todoList = ["Todo 1", "Todo 2", "Todo 3"]
todoList.insert("Todo 4", at: 0)
```

手动 维护视图

```
tableView.beginUpdates()
tableView.insertRows(at: [IndexPath(row: 0, section: 0)],
                     with: .automatic)
tableView.endUpdates()
```

自动维护/响应式/观察者模式
数据驱动

```swift
var todoList = ["Todo 1", "Todo 2", "Todo 3"]
todoList.insert("Todo 4", at: 0)
```

数据

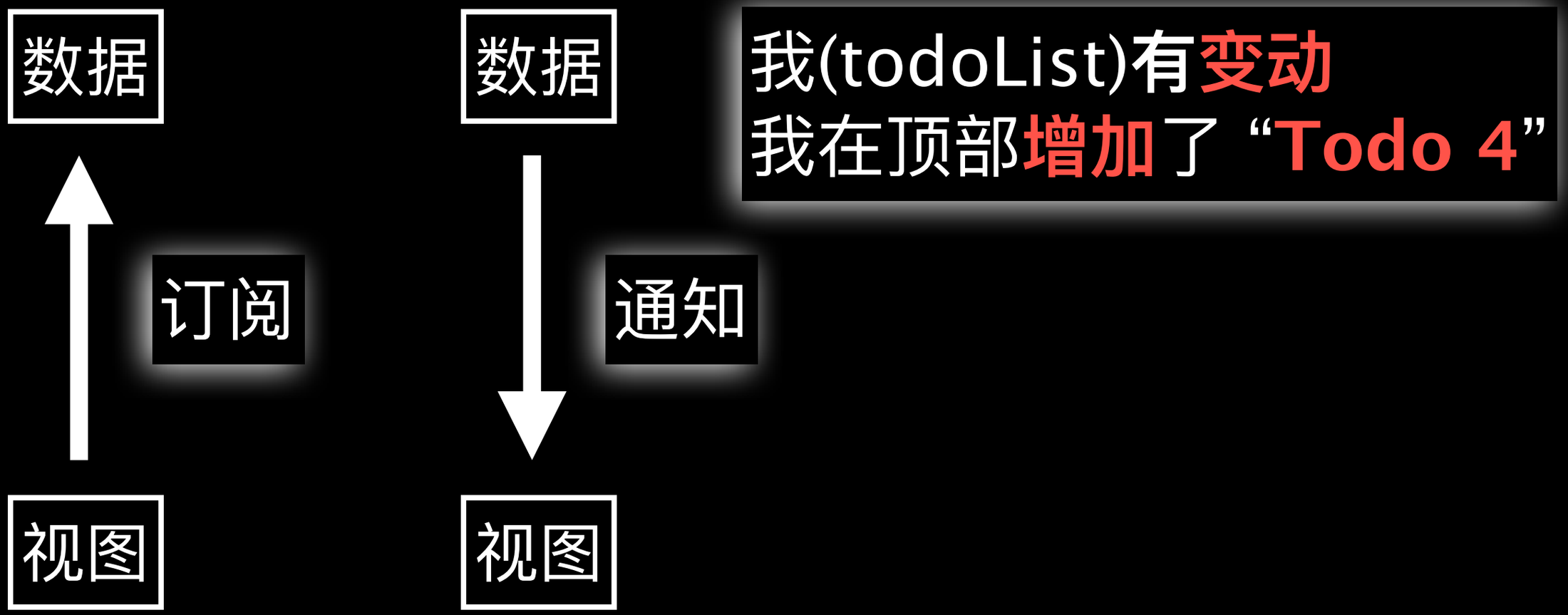
我(todoList)有**变动**
我在顶部**增加**了 “**Todo 4**”

视图

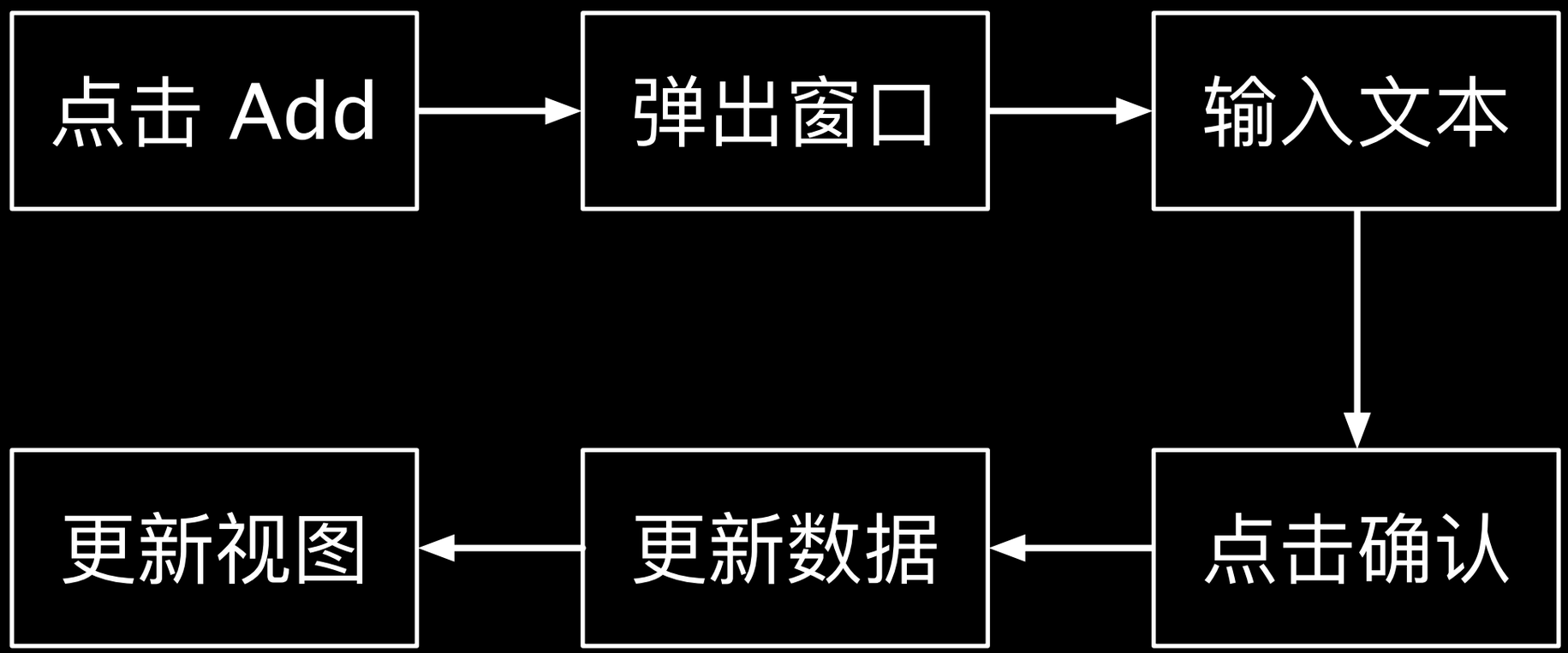
```swift
tableView.beginUpdates()
tableView.insertRows(at: [IndexPath(row: 0, section: 0)],
                     with: .automatic)
tableView.endUpdates()
```

咦？数据变了，增加了 “Todo 4”
我对应做一些更新

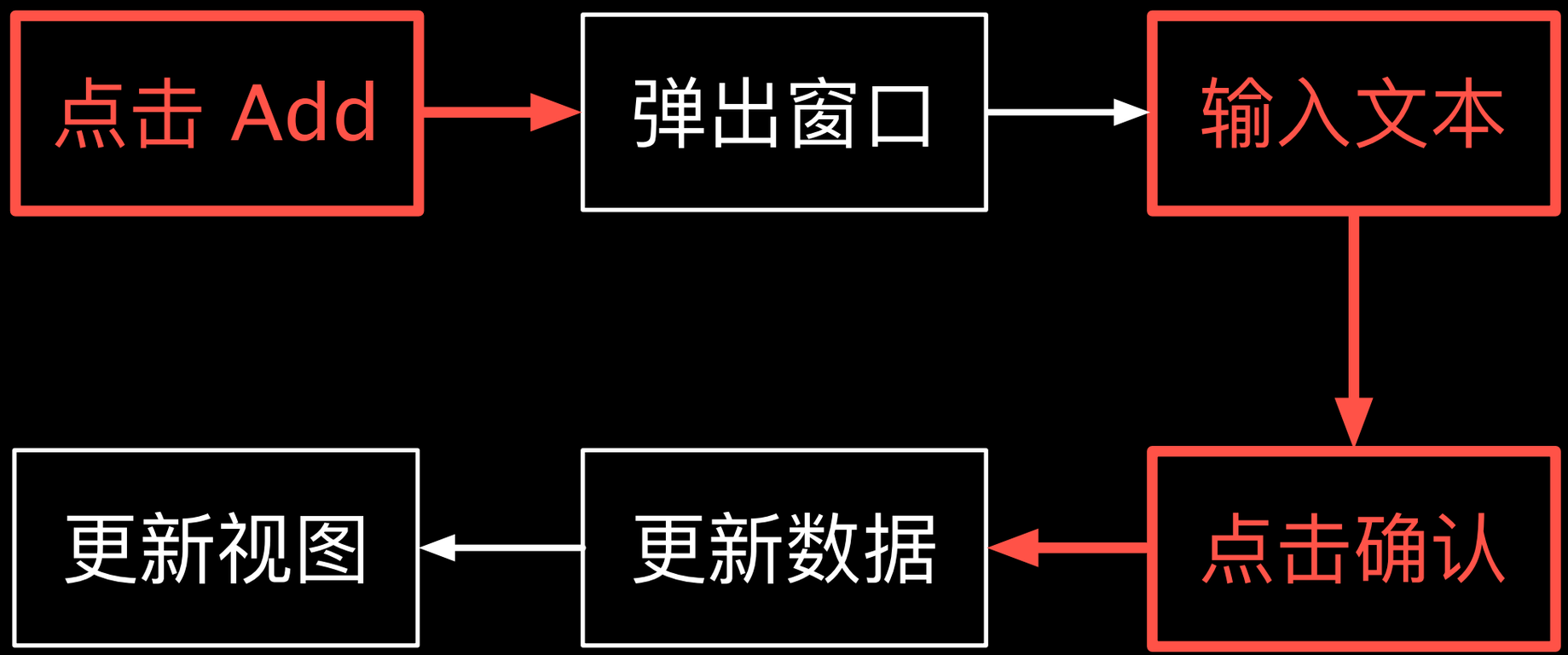
自动维护/响应式/观察者模式



回到真实场景



用户行为



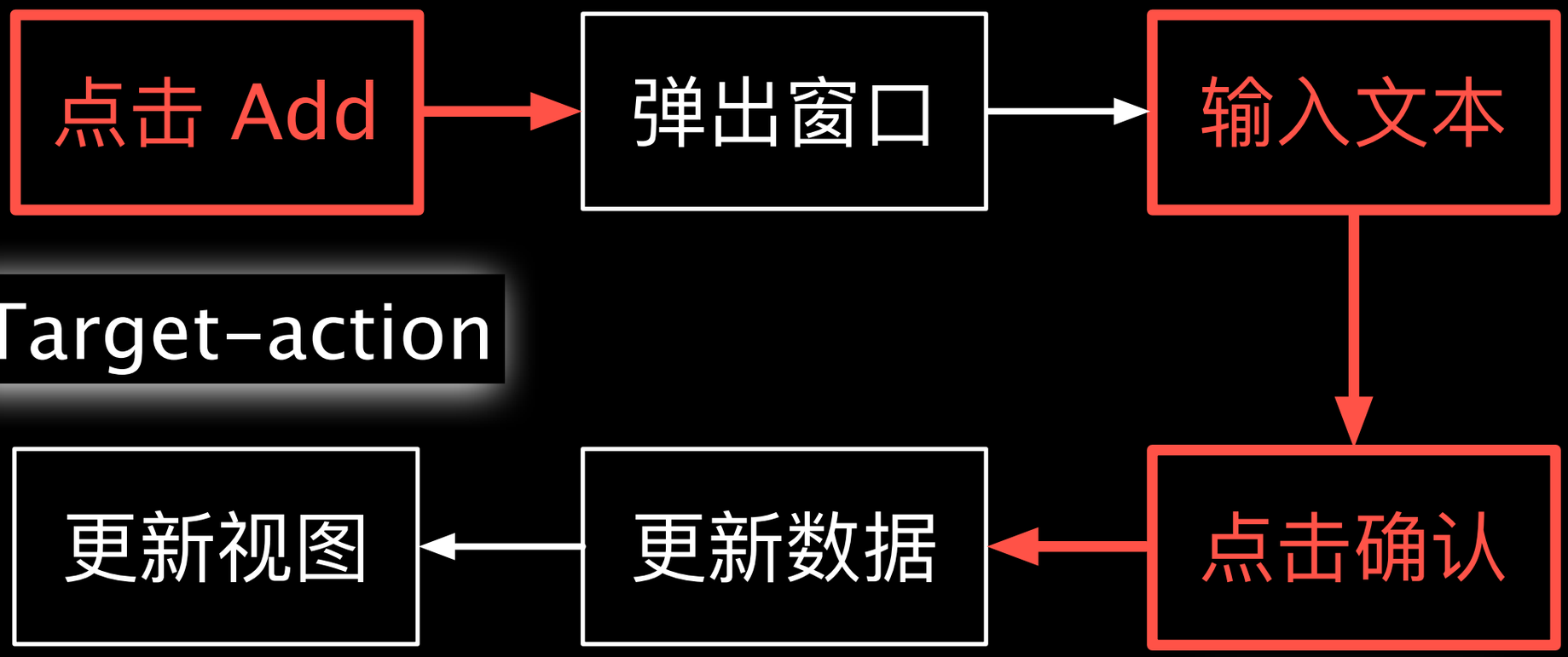
用户触发才进行下一步

等待用户行为

等待用户行为 = 等待网络回调

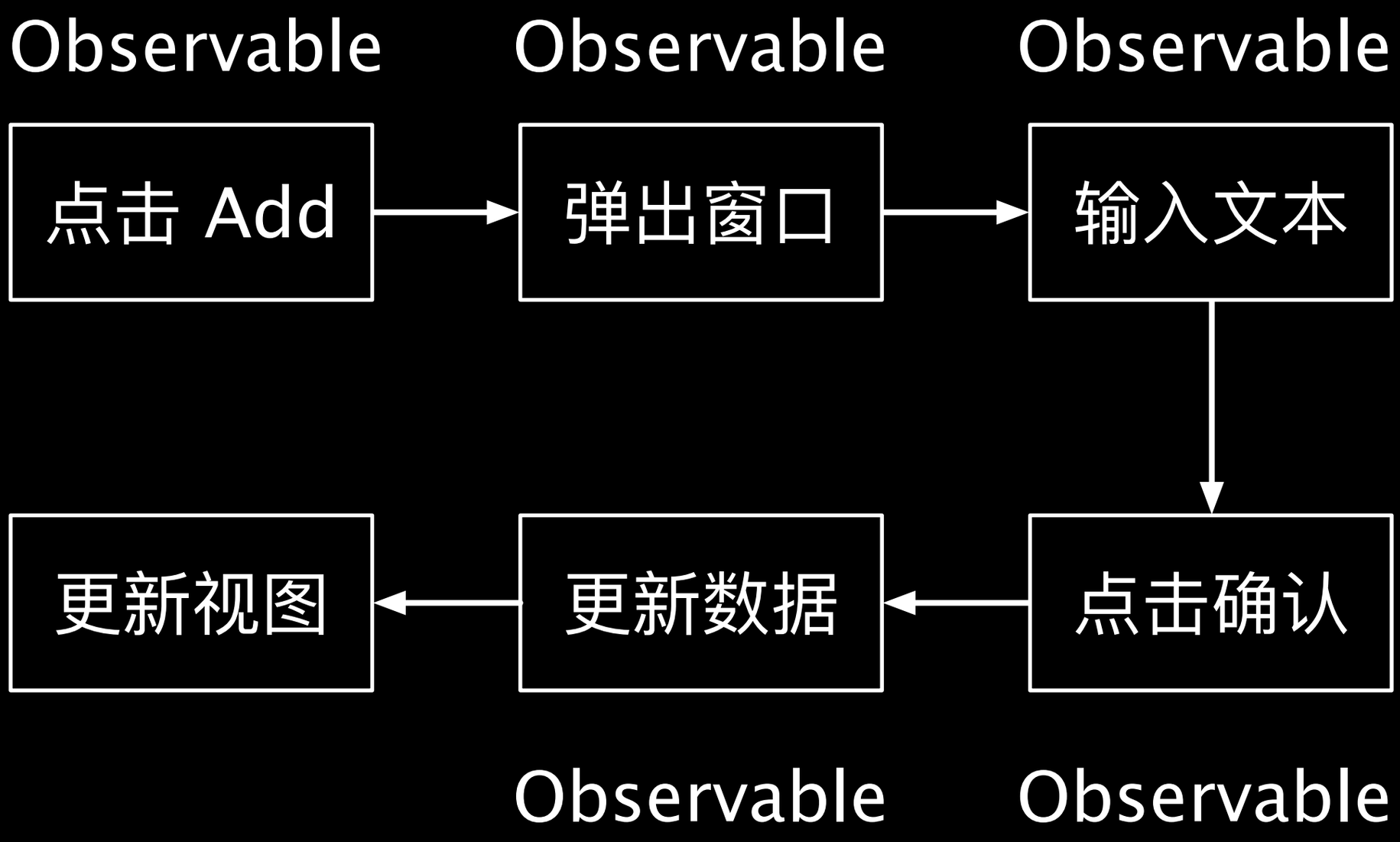
- 回调触发时间未知
- 可以采用 Closure / delegate / target-action

利用原生 API 实践



代码很容易变得很乱
易读性很差

Observable


Observable
Observable
Observable
点击 Add
弹出窗口
输入文本
更新视图
更新数据
点击确认
Observable
Observable

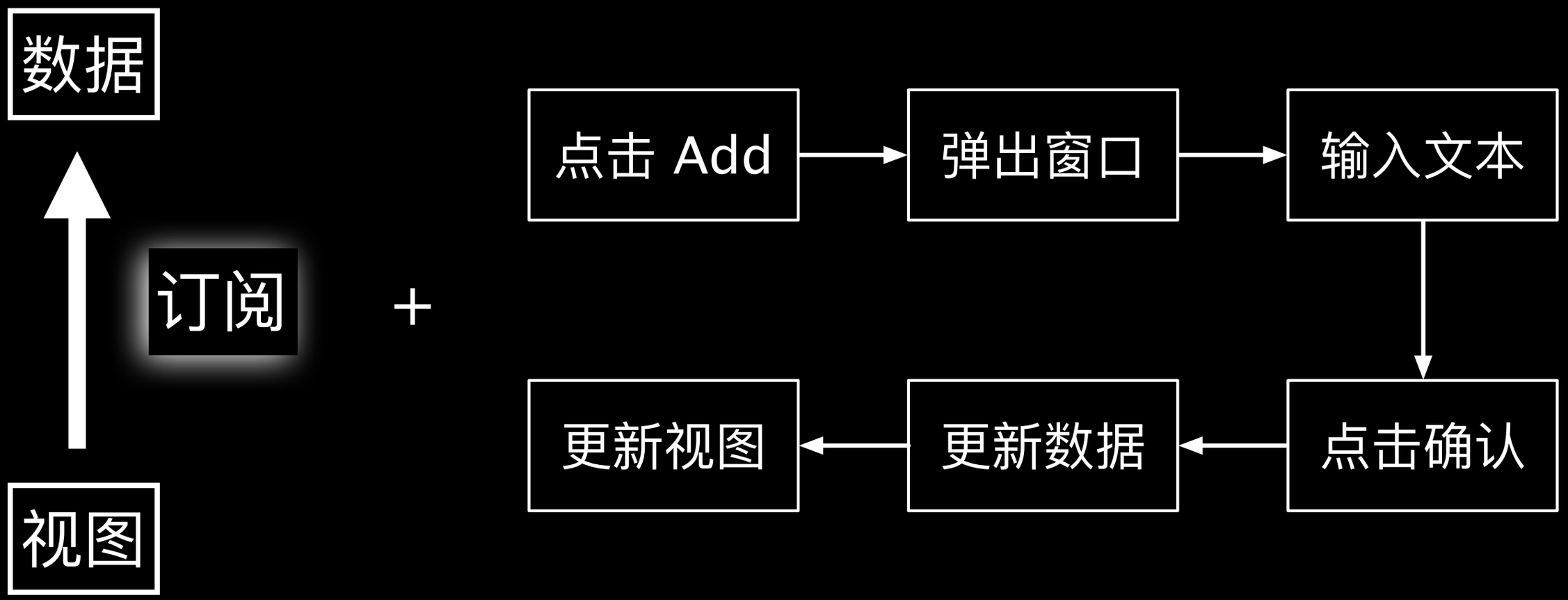
一切都是“同步”的

```
 (Click Add)
.(Show Alert)
.(Input Text)
.(Click Ensure)
.(Update Database)
.(Update View)
```

一切都是“同步”的

```
(Click Add)
addBarButtonItem.rx
    .tap
    .flatMap(showTextField)
    .map { [$0] }
    .scan([String](), accumulator: +)
    .map { [TodoSectionModel(model: "", items: $0)] }
    .bindTo(tableView.rx.items(dataSource: dataSource))
(Update View)
```

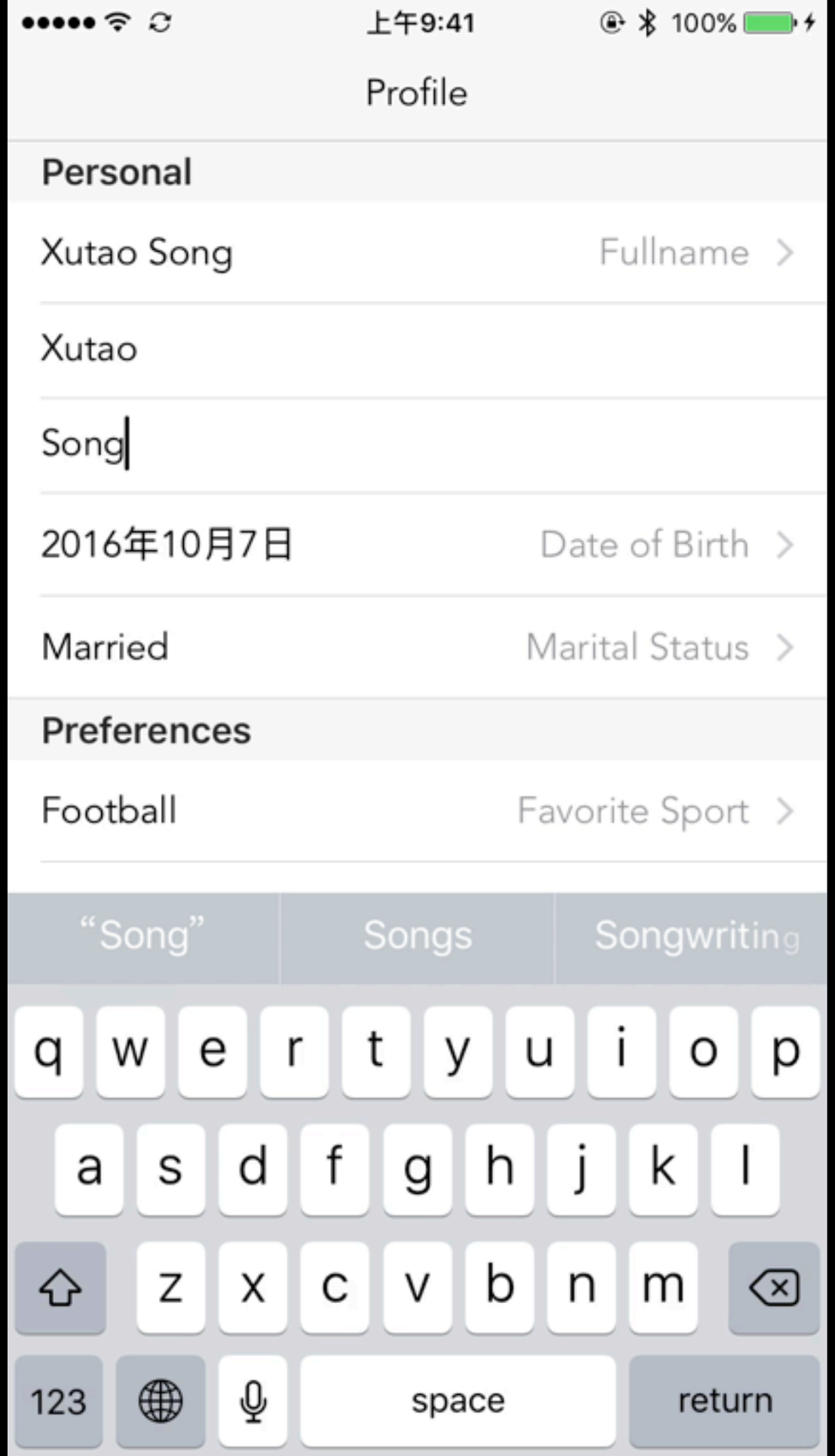

响应式



填写信息

对“如何在 iOS 中实现一个可展开的 Table View”的实践

http://swift.gg/2015/12/03/expandable-table-view/



Expandable 〉 DianQK | Running Expandable on DianQK

Expandable 〉 Expandable 〉 ViewController.swift 〉 M dateWasSelected(_:)

```
164     func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
165         let indexOfTappedRow = visibleRowsPerSection[indexPath.section][indexPath.row]
166
167         if cellDescriptors[indexPath.section][indexOfTappedRow]["isExpandable"] as! Bool == true {
168             var shouldExpandAndShowSubRows = false
169             if cellDescriptors[indexPath.section][indexOfTappedRow]["isExpanded"] as! Bool == false {
170                 // In this case the cell should expand.
171                 shouldExpandAndShowSubRows = true
172             }
173
174             cellDescriptors[indexPath.section][indexOfTappedRow]["isExpanded"] = shouldExpandAndShowSubRows
175
176             for i in (indexOfTappedRow + 1)...(indexOfTappedRow + (cellDescriptors[indexPath.section][indexOfTappedRo
                    ["additionalRows"] as! Int)) {
177                 cellDescriptors[indexPath.section][i]["isVisible"] = shouldExpandAndShowSubRows
178             }
179         } else {
180             if cellDescriptors[indexPath.section][indexOfTappedRow]["cellIdentifier"] as! String == "idCellValuePicke
181                 var indexOfParentCell: Int!
182
183                 for i in stride(from: indexOfTappedRow - 1, to: 0, by: -1) {
184                     if cellDescriptors[indexPath.section][i]["isExpandable"] as! Bool == true {
185                         indexOfParentCell = i
186                         break
187                     }
188                 }
189
190                 cellDescriptors[indexPath.section][indexOfParentCell]["primaryTitle"] = (tblExpandable.cellForRow(at:
                        as! CustomCell).textLabel?.text
191                 cellDescriptors[indexPath.section][indexOfParentCell][
192
193                 for i in (indexOfParentCell + 1)...(indexOfParentCell                    exPath.section][indexOfPa
                        ["additionalRows"] as! Int)) {
194                     cellDescriptors[indexPath.section][i]["isVisible"]
195                 }
196             }
197         }
198
199         getIndicesOfVisibleRows()
200         tblExpandable.reloadSections(IndexSet(integer: indexPath.section), with: UITableViewRowAnimation.fade)
201     }
202
```

Thread 1: EXC_BREAKPOINT (code=1, su

Expandable 〉 Thread 1 〉 1 ViewController.tableView(UITableView, didSelectRowAt : IndexPath) -> ()

- tableView = (UITableView) 0x0000000013682e400
- indexPath = (IndexPath) 2 indices
- self = (Expandable.ViewController) 0x0000000135d0e3c0
- indexOfTappedRow = (Int) 1
- indexOfParentCell = (Int!) nil

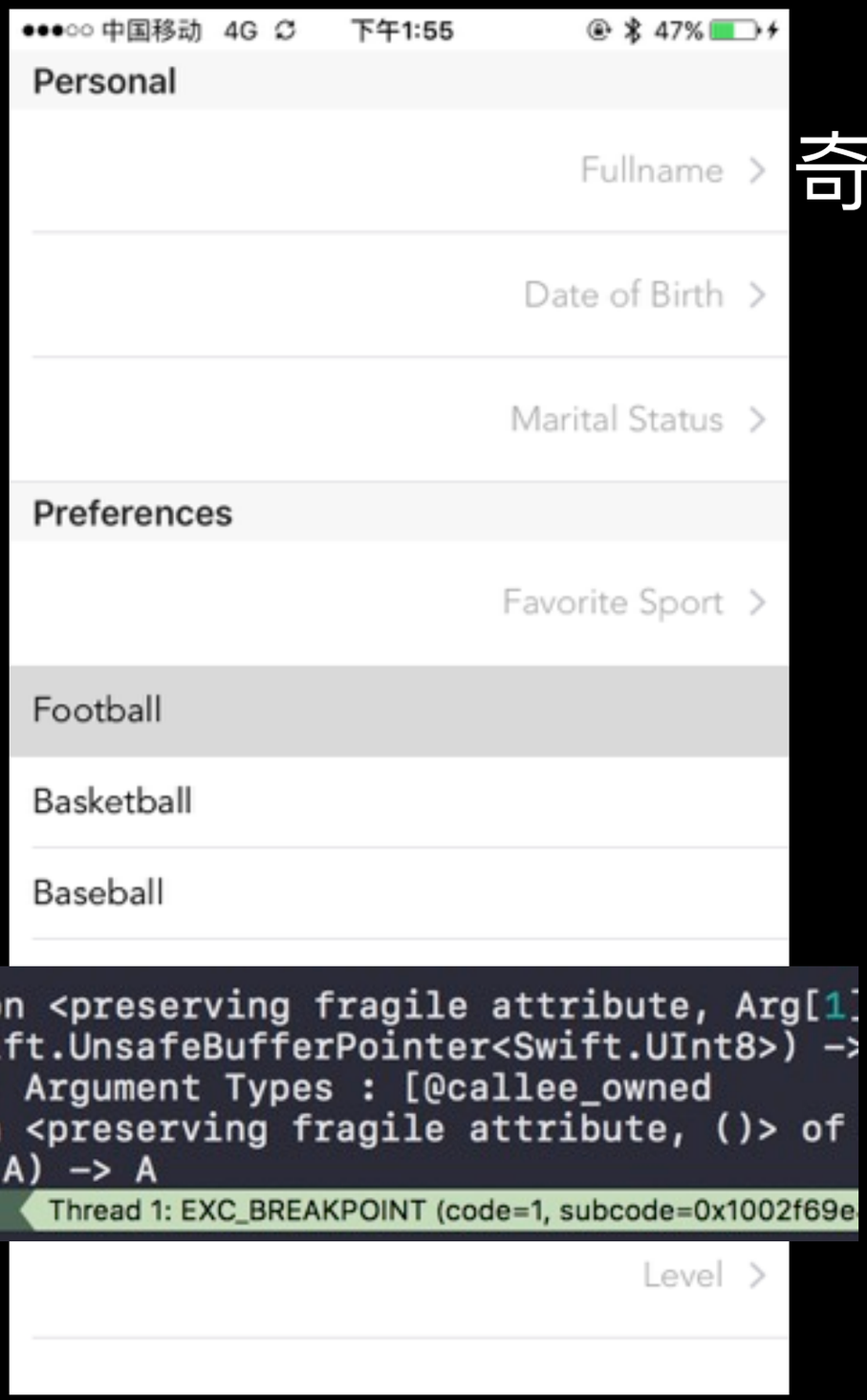
```
(void *) $0 = 0x0000000000000000
fatal error: unexpectedly found nil while unwrapping an Optional val
2016-10-07 13:55:07.889097 Expandable[804:131922] fatal error: unexpe
while unwrapping an Optional value
(lldb)
```

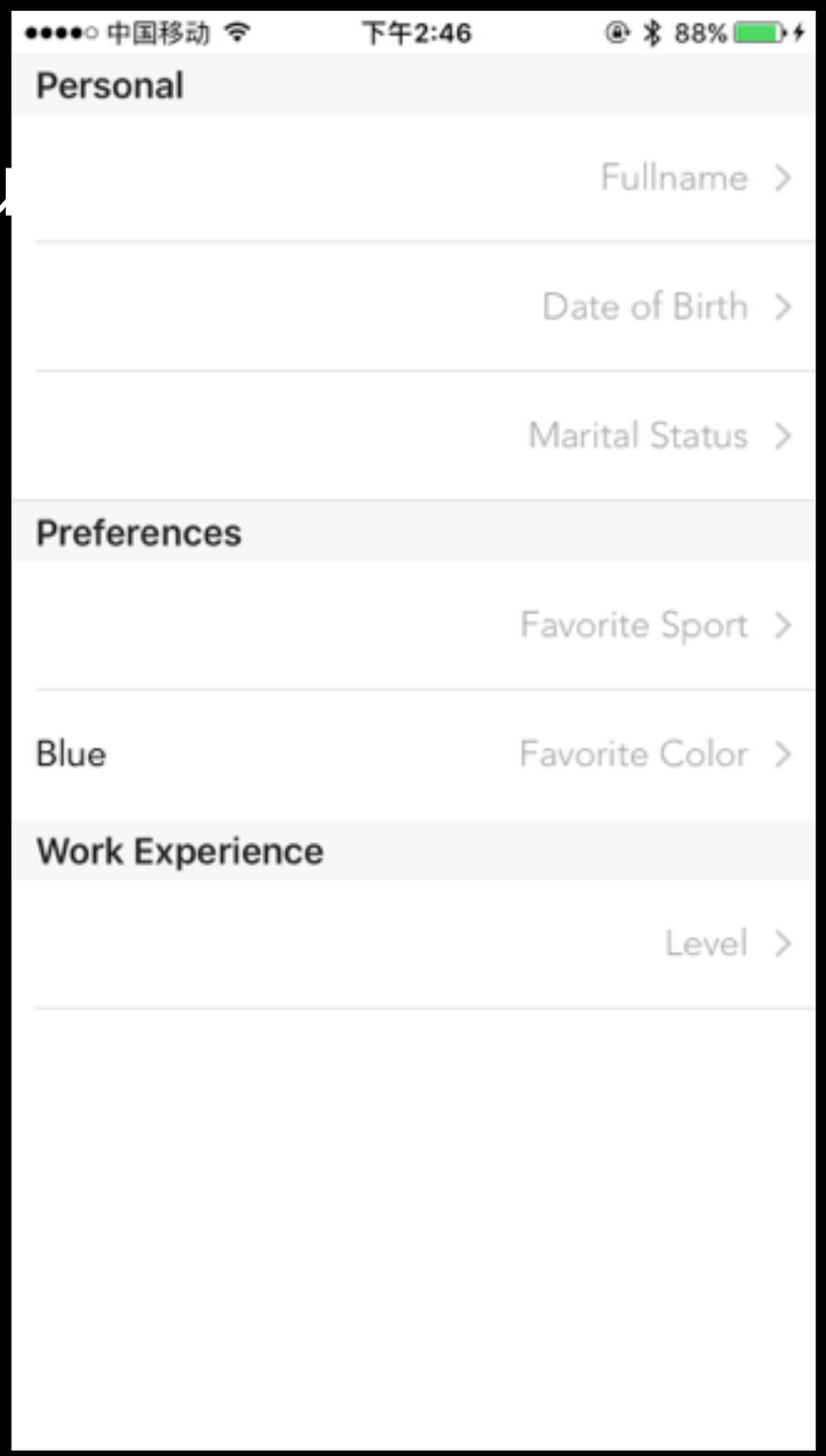
Auto | Filter | All Output | Filter

804 2% 12.6 MB Low 212 KB/s Zero KB/s

ue: com....hread (serial)
ed _fatalErrorMessage...
troller.tableView(UITa...
ewController.tableVie...
eView _selectRowAtInd...
eView _userSelectRow...
rCACommitDeferredBl...
AfterCAFlushAndRun...
CommitHandler
LOOP_IS_CALLING_O...
LoopDoObservers
nLoopRun
oopRunSpecific
tRunModal
lication _run]
cationMain
kit.eventfetch-thread (...

奇怪的 Bu


●●●○○ 中国移动 4G 下午1:55 47%
Personal
Fullname
Date of Birth
Marital Status
Preferences
Favorite Sport
Football
Basketball
Baseball
tion <preserving fragile attribute, Arg[1
Swift.UnsafeBufferPointer<Swift.UInt8>) ->
)), Argument Types : [@callee_owned
ion <preserving fragile attribute, ()> of
-> A) -> A
Thread 1: EXC_BREAKPOINT (code=1, subcode=0x1002f69e
Level


●●●●○ 中国移动 下午2:46 88%
Personal
Fullname
Date of Birth
Marital Status
Preferences
Favorite Sport
Blue
Favorite Color
Work Experience
Level

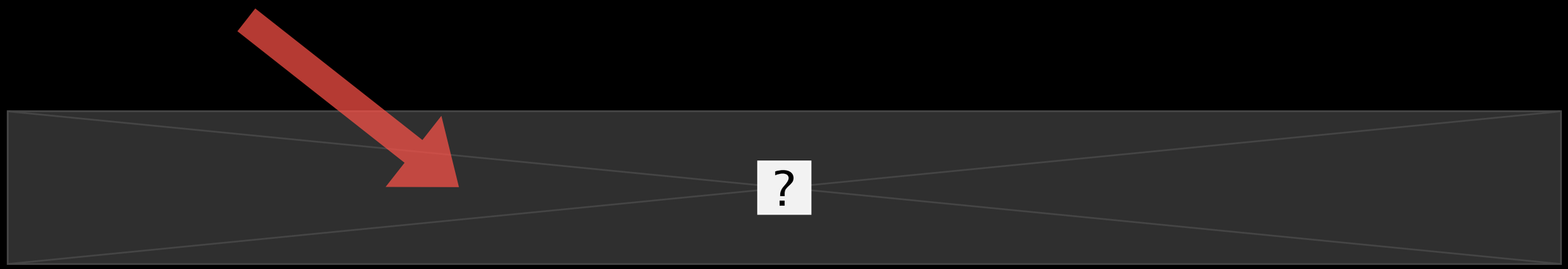
查找父级单元格

在 `if` 分支内，我们将执行四种不同的任务：

1. 首先，我们需要找到顶级单元格的行索引值，即你点击选中的单元格的“父母”。事实上，我们采用自下而上（即从点击选中的单元格开始向上遍历）的方式对单元格描述数组执行一次搜索，首个属性 `isExpandable = true` 的单元格就是我们想要的家伙。
2. 接着，将顶级单元格中的 textLabel 标签值设置为选中单元格的值。
3. 然后，设置顶级单元格的 isExpanded 等于 false ，即折叠状态。
4. 最后，标记顶级单元格下的所有子单元格为不可见状态。

```
var indexOfParentCell: Int!

// 任务一
for var i=indexOfTappedRow - 1; i>=0; --i {
    if cellDescriptors[indexPath.section][i]["isExpandable"] as! Bool == true {
        indexOfParentCell = i
        break
    }
}
// 任务二
```

理解错了函数



硬编码

```
tblExpandable.register(UINib(nibName: "NormalCell", bundle: nil), forCellReuseIdentifier: "idCellNormal")
tblExpandable.register(UINib(nibName: "TextfieldCell", bundle: nil), forCellReuseIdentifier: "idCellTextfield")
tblExpandable.register(UINib(nibName: "DatePickerCell", bundle: nil), forCellReuseIdentifier: "idCellDatePicker")
tblExpandable.register(UINib(nibName: "SwitchCell", bundle: nil), forCellReuseIdentifier: "idCellSwitch")
tblExpandable.register(UINib(nibName: "ValuePickerCell", bundle: nil), forCellReuseIdentifier: "idCellValuePicker")
tblExpandable.register(UINib(nibName: "SliderCell", bundle: nil), forCellReuseIdentifier: "idCellSlider")
```

```
guard let path = Bundle.main.path(forResource: "CellDescriptor", ofType: "plist")
    , let cellDescriptors = NSArray(contentsOfFile: path) as? Array<Array<[String : Any]>> else { return }
self.cellDescriptors = cellDescriptors
```

```
switch section {
case 0:
    return "Personal"

case 1:
    return "Preferences"

default:
    return "Work Experience"
}
```

```
if parentCellIndexPath?.row == 1 {
    if fullnameParts.count == 2 {
        newFullname = "\(newText) \(fu
    } else {
        newFullname = newText
    }
} else {
    newFullname = "\(fullnameParts[0]) \(newText)"
}
```

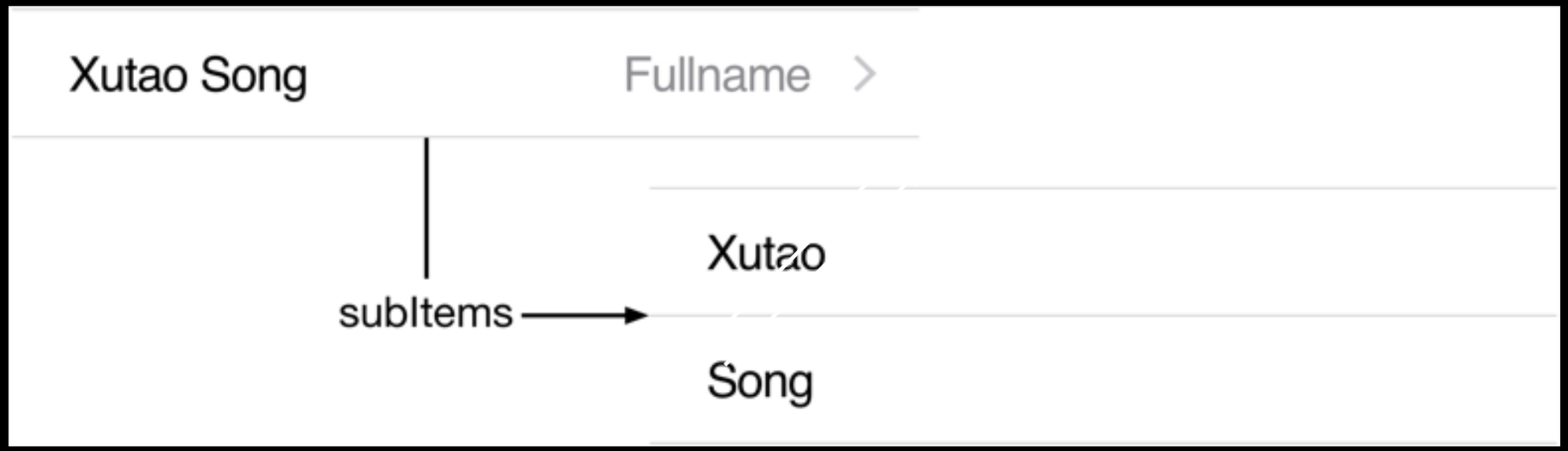
复杂的 if else

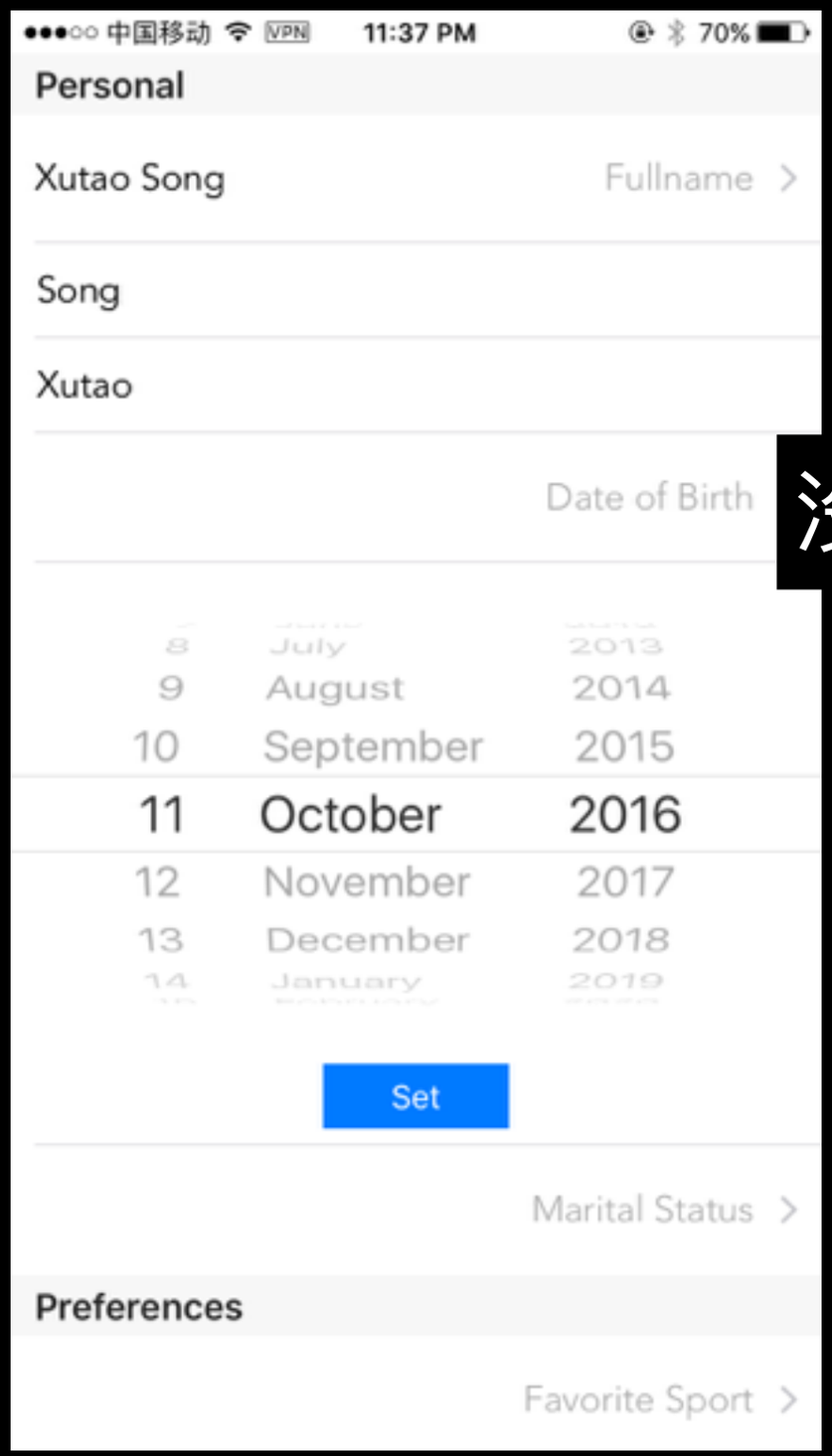
```
if cellDescriptors[indexPath.section][indexOfTappedRow]["isExpandable"] as! Bool == true {
    var shouldExpandAndShowSubRows = false
    if cellDescriptors[indexPath.section][indexOfTappedRow]["isExpanded"] as! Bool == false {
        // In this case the cell should expand.
        shouldExpandAndShowSubRows = true
    }
    cellDescriptors[indexPath.section][indexOfTappedRow]["isExpanded"] = shouldExpandAndShowSubRows

    for i in (indexOfTappedRow + 1)...(indexOfTappedRow + (cellDescriptors[indexPath.section][indexOfTappedRow]
        ["additionalRows"] as! Int)) {
        cellDescriptors[indexPath.section][i]["isVisible"] = shouldExpandAndShowSubRows
    }
} else {
    if cellDescriptors[indexPath.section][indexOfTappedRow]["cellIdentifier"] as! String == "idCellValuePicker" {
        var indexOfParentCell: Int!

        for i in stride(from: indexOfTappedRow - 1, to: -1, by: -1) {
            if cellDescriptors[indexPath.section][i]["isExpandable"] as! Bool == true {
                indexOfParentCell = i
                break
```

如何优雅的展开 Cell

- Diff 处理
  - 判断那些是新增、删除数据
  - 调用相关 API
- 应用响应式

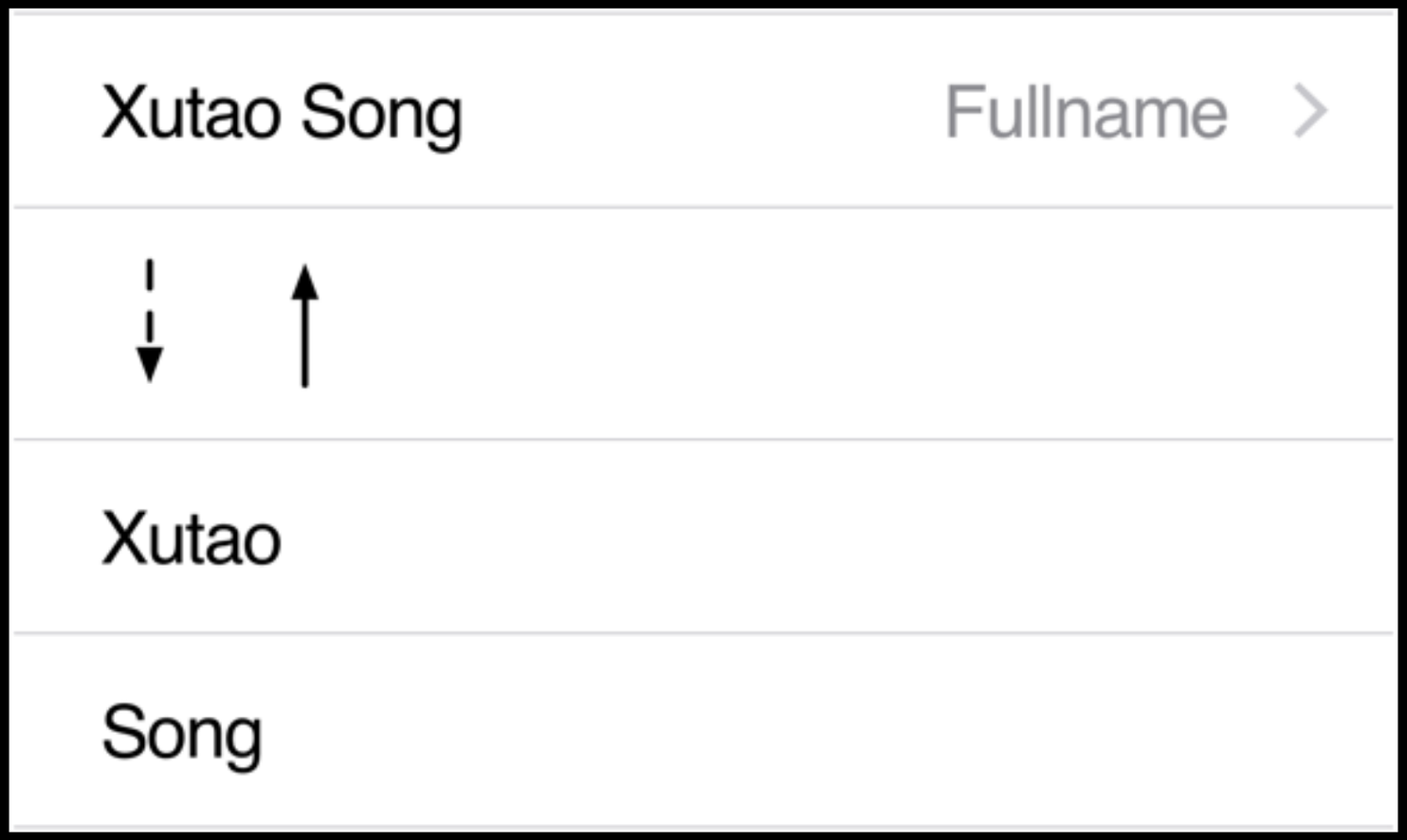
建立 SubItems





没有显示已经输入的内容

绑定



(firstName + lastName) => title

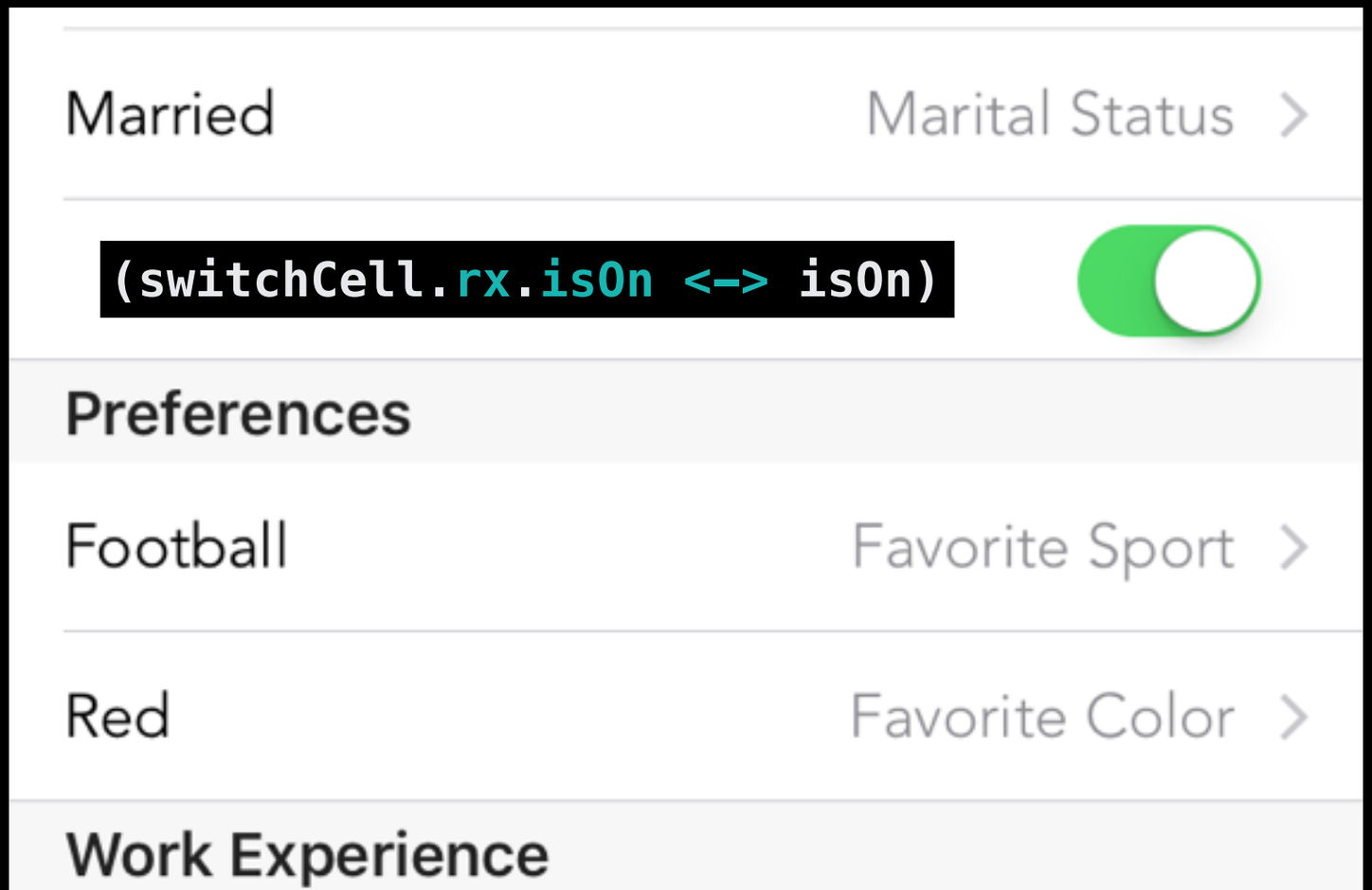
```
Observable
    .combineLatest(
        firstName,
        lastName
    ) { $0 + " " + $1 }
    .bindTo(title)
```

date.map(format) => title

```
date.asObservable()
    .map(DateFormatter().config.longStyle.string)
    .bindTo(title)
```

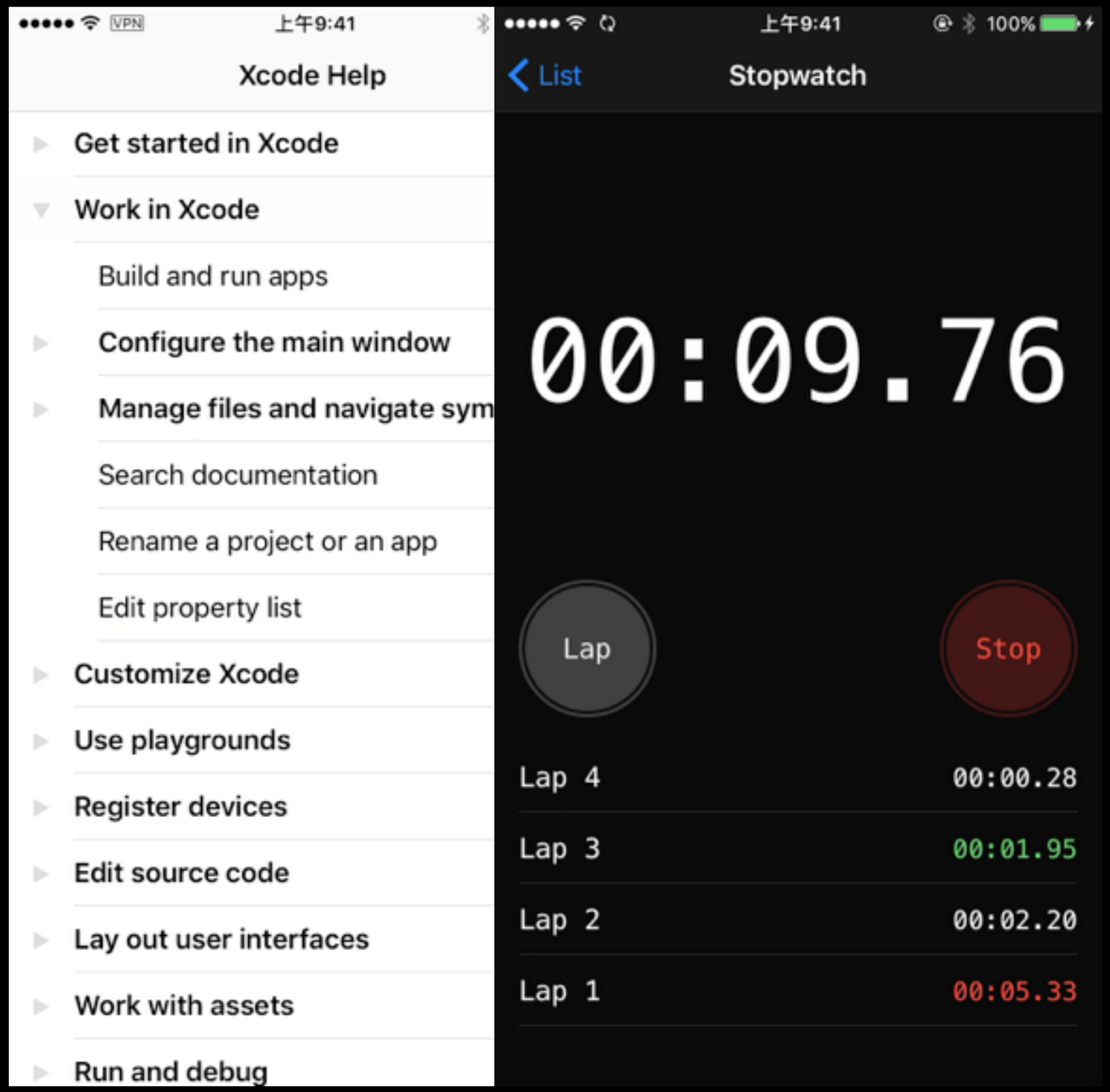
isMarried.map(format) => title

```
isMarried.asObservable()
    .map { $0 ? "Married" : "Married" }
    .bindTo(title)
```

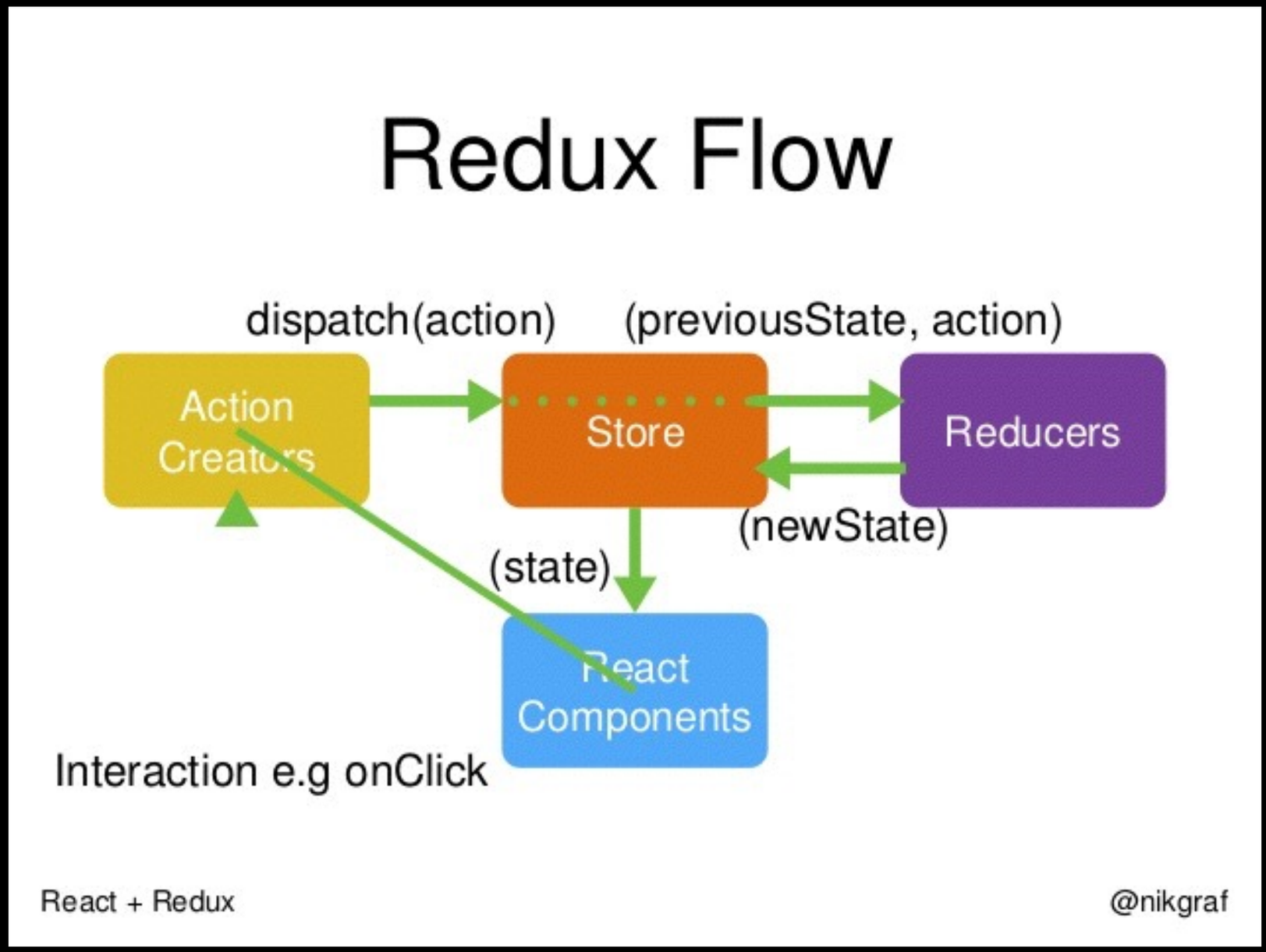


其他例子

- 书籍目录
- Apple 计时器



结合 Redux ?


Redux Flow
dispatch(action)
(previousState, action)
Action Creators
Store
Reducers
(newState)
(state)
React Components
Interaction e.g onClick
React + Redux
@nikgraf

总结

- RxSwift = Observable + LINQ
- RxSwift 是数据/事件驱动框架
- RxSwift 可以解决**各种异步**问题

相关内容

- RxSwift vs PromiseKit
- DianQK/rx-sample-code
- DianQK/rx-redux-sample-code
- RxDataSources
- 如何在 iOS 中实现一个可展开的 Table View
- devxoul/RxTodo

提问?

- dianqk@icloud.com
- Weibo: 靛青K
- Twitter: Songxut