Architecture made for SwiftUI

Matěj Kašpar Jirásek

Lead iOS developer @ The Funtasty

BaiOS WWDC Special Meetup

Binárium, 25. 6. 2019

Architecture made for declarative user interface

```
import SwiftUI

struct LandmarkList: View {
    var body: some View {
        NavigationView {
            List(landmarkData) { landmark in
                NavigationButton(destination: LandmarkDetail(landmark: landmark)) {
                    LandmarkRow(landmark: landmark)
                }
            }
            .navigationBarTitle(Text("Landmarks"), displayMode: .large)
        }
    }
}
```

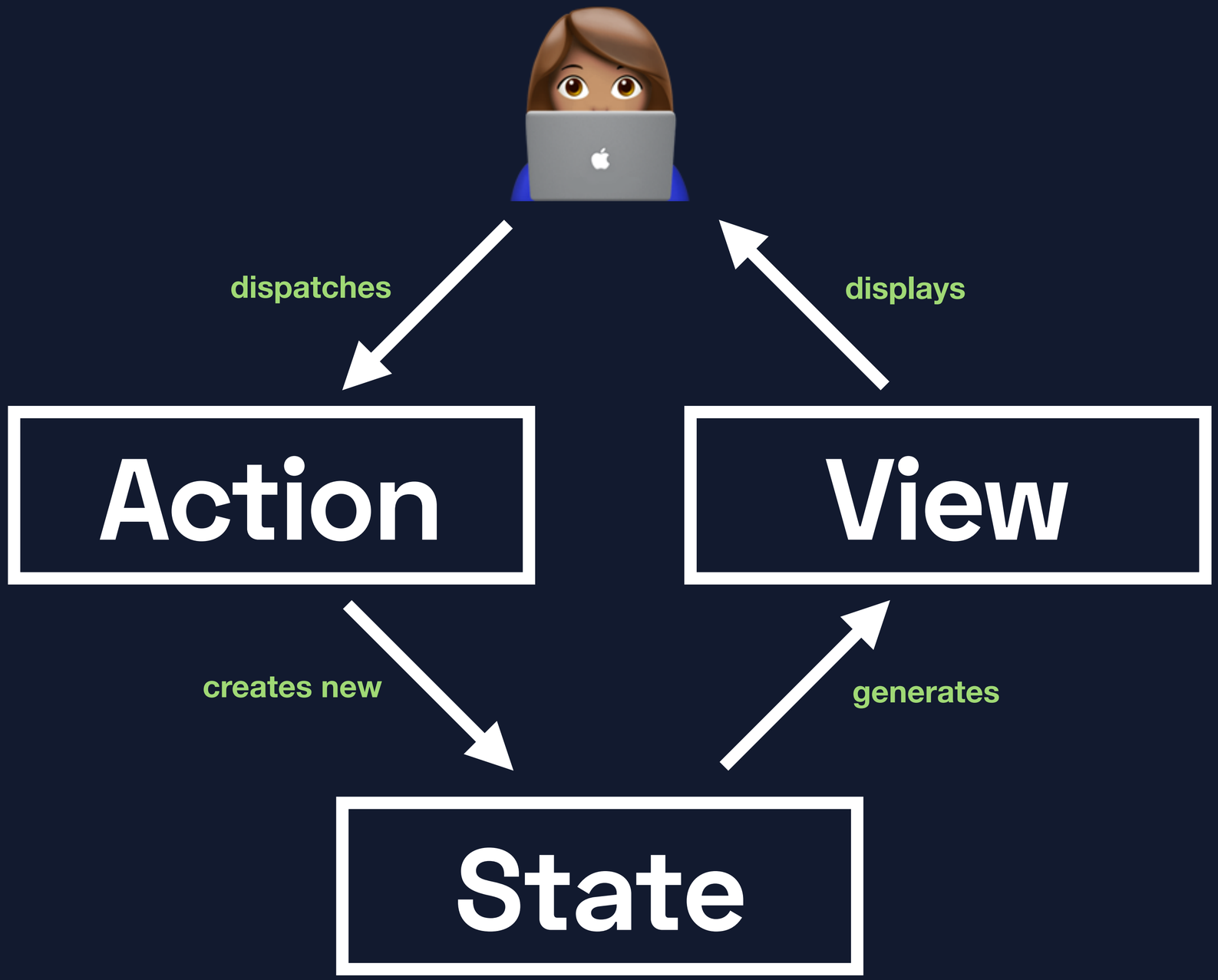

MVC

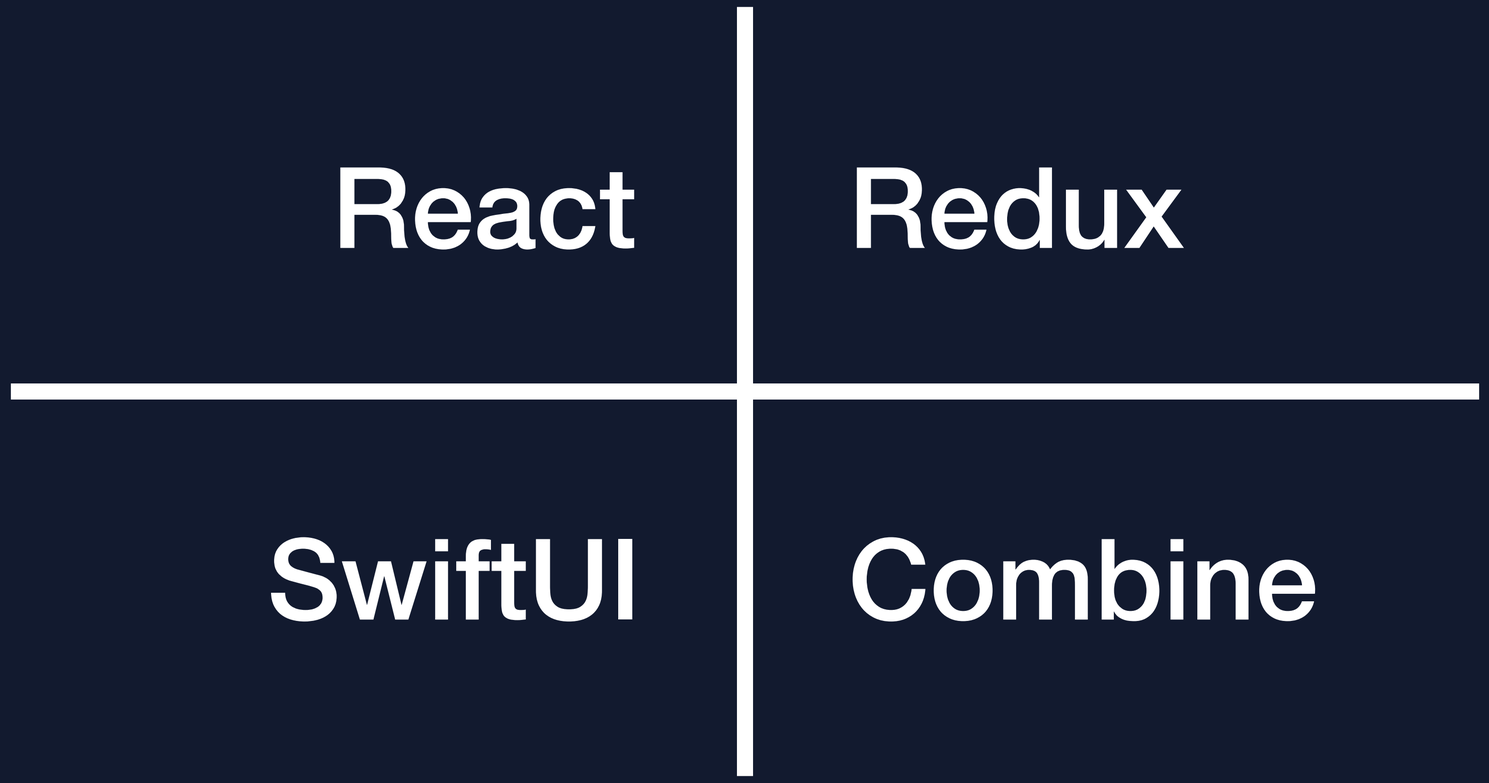
Bi-directional architecture?


Store
View
State

Let’s start from scratch

Uni-directional architecture


dispatches
displays
Action
View
creates new
generates
State


React
Redux
SwiftUI
Combine

```
protocol Action {}
typealias Reducer<State> = (State, Action) -> State
```

```
typealias Call = () -> Void
typealias Dispatch = (Action) -> Call
```


+
Counter app
Example
-

```swift
typealias AppState = Int

enum AppAction: Action {
    case increment
    case decrement
    case reset
}
```

```swift
func appReducer(state: AppState, action: Action) -> AppState {
    switch action {
    case AppAction.increment:
        return state + 1
    case AppAction.decrement:
        return max(state - 1, 0)
    case AppAction.reset:
        return 0
    default:
        return state
    }
}
```

```swift
final class ViewController: UIViewController {

    @objc func increment() {
        store.dispatch(action: AppAction.increment)
    }

    @objc func decrement() {
        store.dispatch(action: AppAction.decrement)
    }

    @objc func reset() {
        store.dispatch(action: AppAction.reset)
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = .white

        let label = UILabel()
        label.textAlignment = .center

        let incrementButton = UIButton(type: .system)
        incrementButton.setTitle("+", for: .normal)
        incrementButton.addTarget(self, action: #selector(increment), for: .touchUpInside)

        let decrementButton = UIButton(type: .system)
        decrementButton.setTitle("-", for: .normal)
        decrementButton.addTarget(self, action: #selector(decrement), for: .touchUpInside)

        let resetButton = UIButton(type: .system)
        resetButton.setTitle("Reset", for: .normal)
        resetButton.addTarget(self, action: #selector(reset), for: .touchUpInside)

        let stackView = UIStackView(arrangedSubviews: [label, incrementButton, decrementButton, resetButton])
        stackView.distribution = .fillEqually
        stackView.axis = .vertical
        stackView.frame = CGRect(x: 0, y: 0, width: 300, height: 300)
        view.addSubview(stackView)

        store.subscribe { [weak label] state in
            label?.text = String(state)
        }
    }
}
```

```swift
struct CounterScene: View {

    let dispatch: Dispatch
    let count: Int

    var body: some View {
        VStack {
            Text("Count: \(count)")
            Button(
                action: dispatch(AppAction.increment),
                label: { Text("+") }
            )
            Button(
                action: dispatch(AppAction.decrement),
                label: { Text("-") }
            )
            Button(
                action: dispatch(AppAction.reset),
                label: { Text("Reset") }
            )
        }
    }
}
```

```
Button(action: {
    […]
}, label: {
    Text("SomeLabel")
})
```

```
Button(action: doSomeAction, label: {
    Text("SomeLabel")
})
```

```
(Calls, State) -> View
```

Demo

“There’s beauty in going slowly, in learning things as we come across them.”

–Pedro Piñera

More information

- WWDC 2019 — Session 226 — Data Flow Through SwiftUI
- WWDC 2019 — Session 204 — Introducing SwiftUI: Build Your First App
- ELM, Flux, Redux, ReSwift

Q&A