Declarative and functional iOS apps with SwiftUI and Combine

Agenda

1. About me
2. Swift
3. SwiftUI: Declarative vs. Imperative
4. Combine
5. Coding

About me

- Max Tharr
- Software developer at Mayflower GmbH
- Coming from an Augmented Reality/ Games Engineering background
- Last year, I started an open source app (PayForMe) in order to learn SwiftUI and Combine
- Afterwards, I worked on an iOS app in the FinTech sector for 8 months, where we introduced SwiftUI and Combine

Swift

- First released in 2014 by Apple
- Built on top of the LLVM toolchain
- Invented by Chris Lattner, evolved by a Core Team of 8 people
- Static and strongly typed language
- Multi-Paradigm
- Runs on every major operating system except BSD

Swift principles

Safe

The most obvious way to write code must behave in a safe manner.

Swift principles

Fast

Intended to compete with C++ in performance

Swift principles

Expressive

Readability of code is in focus

Declarative vs. Imperative

- With imperative UI frameworks you have to define the control flow
- With declarative UI frameworks you just define the UI depending on its state
- SwiftUI is a declarative UI framework (like React), with the goal to reduce side effects

Combine

- Combine is a reactive framework heavily inspired by RxSwift
- It combines functional programming principles with asynchronous event handling
- Greatly reduces complexity and side effects on user or network events

Live coding

- Simple app using a graphql backend
- Navigation problem example from production project
- Everything else is up to you!

Outlook

- If you want to build a real (slightly larger) app with SwiftUI and Combine, I highly recommend The Composable Architecture
- It's a framework providing a Redux-like store system
- https://www.hackingwithswift.com/ is a great resource for SwiftUI and Combine
- Using Combine is a great (digital) book providing detailed information about Combine