(Modern)
Swift Concurrency

A journey

Alex Manarpies, October 13, 2021


Intro

tvOS

KBC

iOS

DPG Media

Video

Who am I?

(Clean)
Architecture

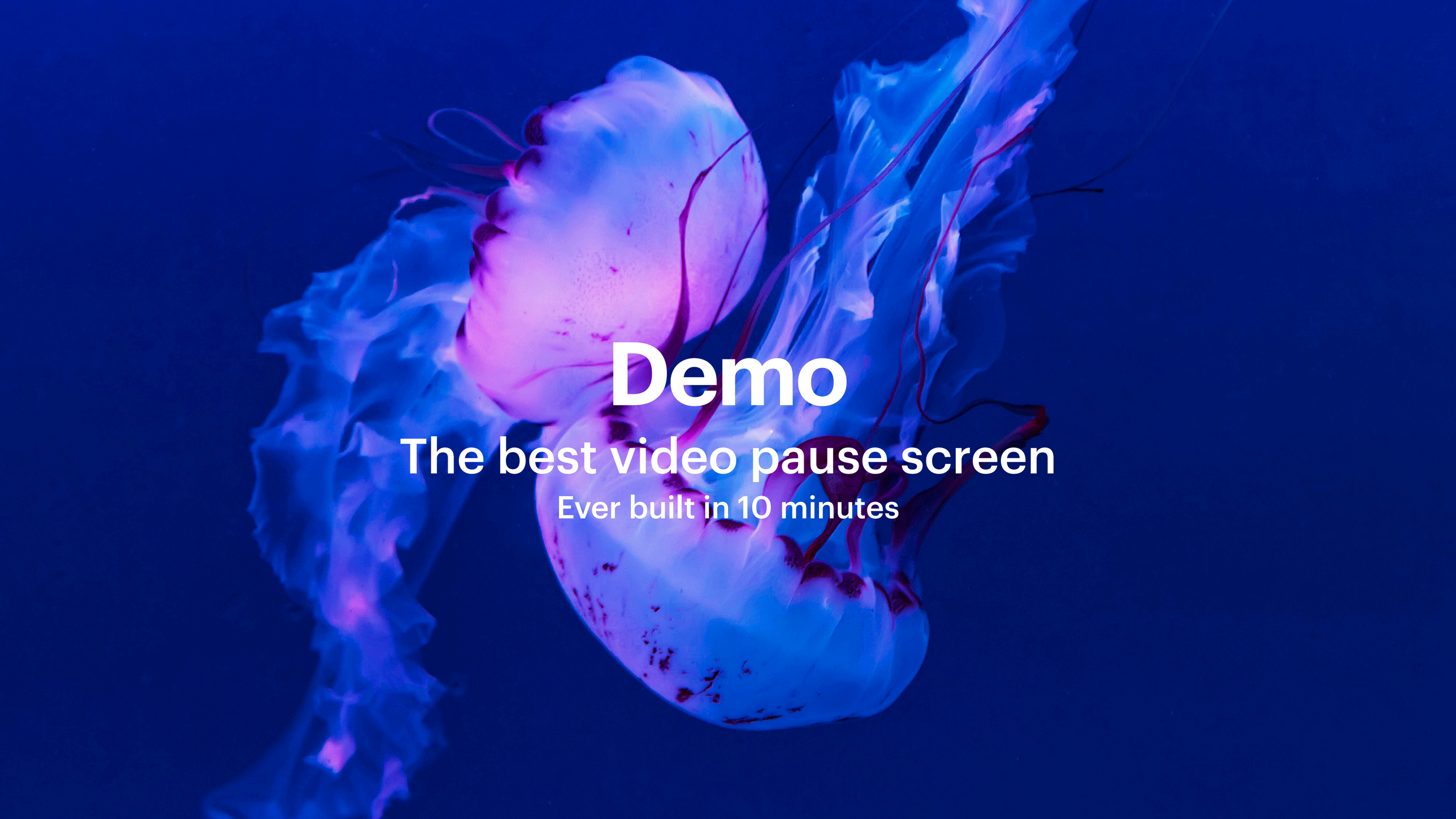
VTM GO

Streamz

Rx

Today

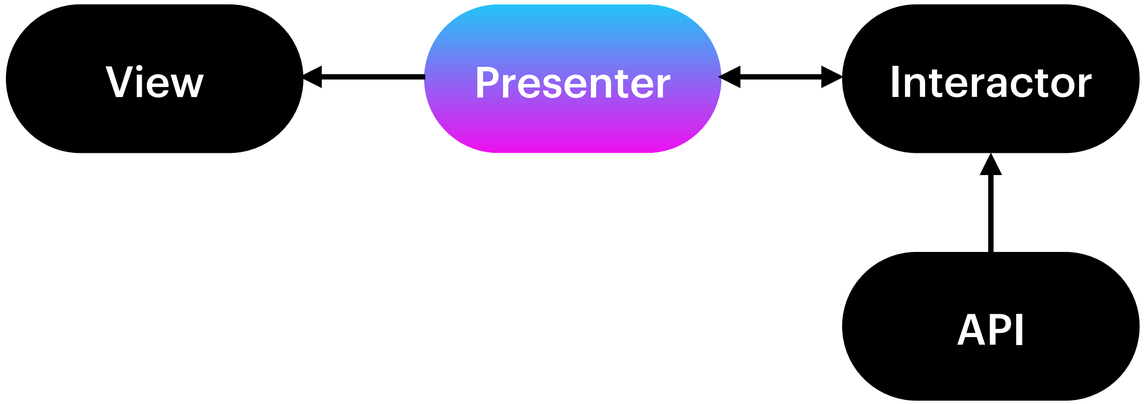
1. Intro to sample project
   “PauseFlix” 😜
2. Closures → RxSwift → Combine
   → Async/Await 🚀
3. Q&A 🎙️


Demo
The best video pause screen
Ever built in 10 minutes


Code

- UIKit with vanilla AutoLayout-in-code
- TMDB REST calls with URLSession
- Decodable models

- Basic architecture:



Closures

What else?

@escaping (Result<Output, Failure>) → Void

- Nesting
- Notify results over @escaping closures
- What about errors?
- Result<Value, Failure>
- Waiting and collecting multiple results?

- Dispatch background work back to main queue with **DispatchQueue.main.async {}**

RxSwift

Fancy, but with a learning curve

@escaping (Result<Output, Failure>) → Void

↓

Observable<Output>

- Learning curve
- Linear “pipeline” architecture
- Error handling, but untyped
- Built-in primitives for sequential (flatMap, concat, andThen operations) and parallel execution (combineLatest, merge)

- Observe on MainScheduler if work was dispatched to background, with **.observe(on: MainScheduler.instance)**

Combine

Kinda-Rx, with some type-safe Apple flair

Observable<Output>

↓

AnyPublisher<Output, Failure>

- Maps to RxSwift in many ways
- Explicit typing everywhere, error types included
- Cancellable = inverse of Rx Disposable
- Sink = Subscribe

- Receive background dispatched output on main queue with **.receive(on: DispatchQueue.main)**

Structured Concurrency and async/await

Ready for prime time?

AnyPublisher<Output, Failure>

↓

async throws → Output

- iOS 15 only, required Swift 5.5
  For now. Backport to iOS 13+ is under discussion
- Production-ready? Still a serious **stack corruption** (read: crash) issue present in Swift 5.5 😱
  forums.swift.org/t/swift-5-5-has-serious-stack-corruption-bugs/52344

- Language-level support: explicit **async** and **await** keywords
- Error handling with **do-try-catch**
- Errors untyped, as opposed to Combine
- Annotate with **@MainActor** to automatically dispatch results back to the main queue

Advanced Concurrency Techniques

The almighty Task

- Async-let tasks: statically known number of concurrencies, cancellation
  WWDC session around 04:52
- Group tasks: dynamic number of concurrencies, data races, **@Sendable**, Actors, **for try await**
  WWDC session around 12:55
- Unstructured tasks: launch async work from non-async code, **Task {}**, detached tasks
  WWDC session around 19:05


Q&A

Resources

- RxMarbles: How observable operators work
  rxmarbles.com
- RxSwift to Combine Cheat Sheet
  github.com/CombineCommunity/rxswift-to-combine-cheatsheet
- WWDC session on Structured Concurrency and async/await
  developer.apple.com/videos/play/wwdc2021/10134
- Book: Clean Architecture by “Uncle Bob” Robert C. Martin
  amazon.com/Clean-Architecture-Craftsmans-Software-Structure-ebook-dp-B075LRM681/dp/B075LRM681

Thank you

You can find me on linkedin.com/in/alexmanarpies