PHASE 2 LUNCH & LEARN - TIM GOURLEY

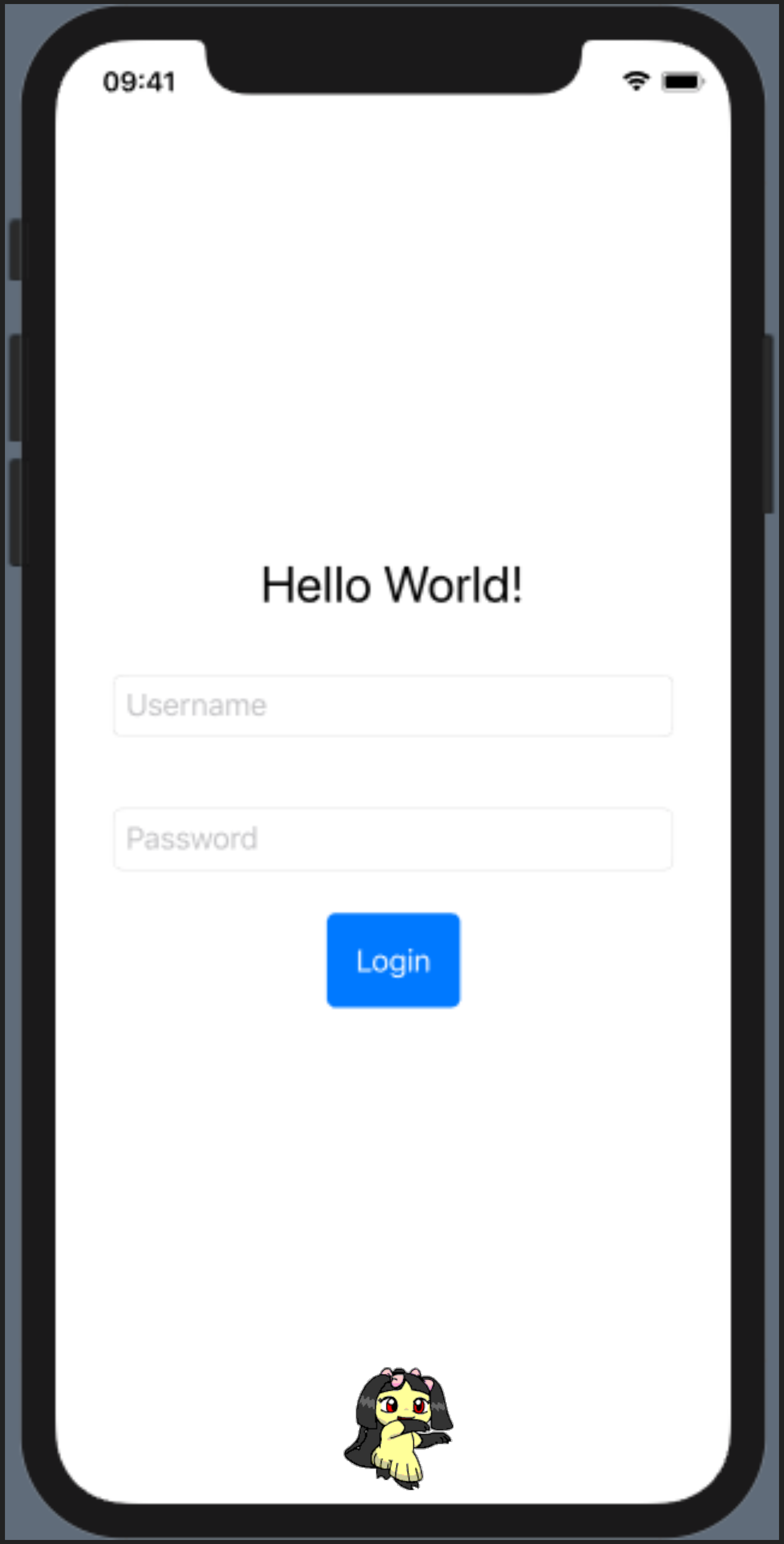
SWIFTUI + COMBINE

SWIFTUI

- Declarative user interfaces using the **builder** pattern.
- It's time we see other people, `UIViewController`. I'm with Combine now.
- Ok, maybe not *just yet*. `UIKit` isn't going anywhere for a while.
- Good editor support in Xcode with live previews, poor and misleading errors in the UI currently.
- Not easy to model complex user interfaces just yet.

```swift
struct BasicView: View {
    @State var name: String = ""
    @State var password: String = ""

    var body: some View {
        VStack {
            Text("Hello World!")
                .font(.title)
            // $name is the wrapped binding of the State var
            // and is of type Binding<Bool>. Name is Bool.
            TextField("Username", text: $name)
                .textFieldStyle(RoundedBorderTextFieldStyle())
                .padding()
            SecureField("Password", text: $password)
                .textFieldStyle(RoundedBorderTextFieldStyle())
                .padding()
            Button(action: {
                // Logging in…
            }, label: { Text("Login") })
                .padding()
                .foregroundColor(Color.white)
                .background(Color.blue)
                .cornerRadius(5)
        }.padding()
    }
}
```

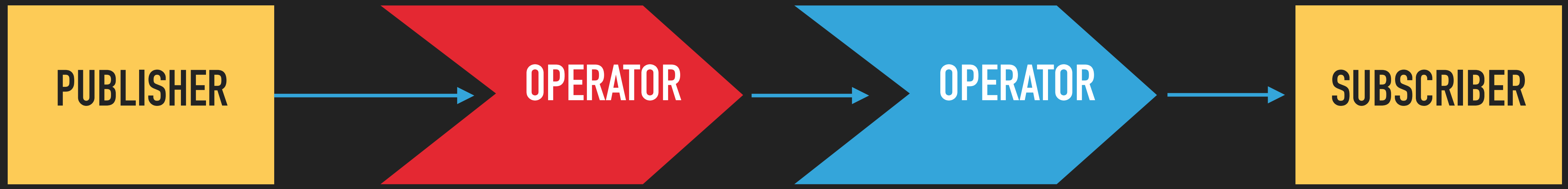

09:41
Hello World!
Username
Password
Login

@STATE, @OBSERVEDOBJECT, & @ENVIRONMENTOBJECT

- `@State var name: String = “John Doe”`
  Reactive variable belonging to a View. The compiler manages the memory so it is persistent as long as the view exists.
- `@ObservedObject var unsplash = UnsplashViewModel()`
  Similar to @State, but external to the View. You instantiate it, populate it, etc by implementing ObservableObject. Important data can be exposed and made reactive by the @Published annotation.
- `@EnvironmentObject var order: Order`
  Similar to @ObservedObject, but provided for the entire application. You will need to pass in an instance of this object when instantiating the view. Makes global state more convenient.

COMBINE

- Framework providing a declarative API for dealing with asynchronous operations. Think of it as functional reactive programming.
- Declares both **publishers** and **subscribers**.
- `Publisher` protocol declares a type that can deliver a sequence of values over time, with **operators** to act on values and republish.
- `Subscriber` protocol acts on elements as it receives them. Publishers only emit values when explicitly requested by the Subscriber.
- Comparable to **RxSwift** and **ReactiveSwift**.


PUBLISHER
OPERATOR
OPERATOR
SUBSCRIBER

CONNECTING A SUBSCRIBER TO A PUBLISHER

- `Subscriber.Sink`
  - Handles a new element/completion event in a closure
  - Can be cancellable; handy for volatile publishers
- `Subscriber.Assign`
  - Write a new element to a property, like on a UI control/ view.
  - Not supposed to fail, so you must handle errors before calling assign.

```swift
// Build a 2D array [[1,2], [3,4], [5,6] .. [17,18]]
var images: [[Int]] = []

// Create a publisher from the sequence
// This is a variable that is a Publisher.Sequence
_ = (1...18).publisher
    // Collect 2 values from the sequence
    // And return a single array. So [1,2], then [3,4] ...
    .collect(2)
    // Now return a single array of all the sequences
    // So [[1,2], [3,4], ... [17,18]]
    .collect()
    // sink is our subscriber function, designating the
    // end of the chain. We will store the 2D array
    // inside the images variable. The other argument for
    // sink is “receiveCompletion” to determine success/
    // failure for the operation.
    .sink(receiveValue: { images = $0 })
```

LINKS FOR MORE INFORMATION

- Introducing Combine WWDC Video
  https://developer.apple.com/videos/play/wwdc2019/722/
- Combine in Practice WWDC Video
  https://developer.apple.com/videos/play/wwdc2019/721/
- Using Combine (book in progress)
  https://heckj.github.io/swiftui-notes/
- https://fuckingswiftui.com/
- CocoaHeads User Group - This Thursday (12/19/2019) @ 11:30
  "100 Days of SwiftUI"