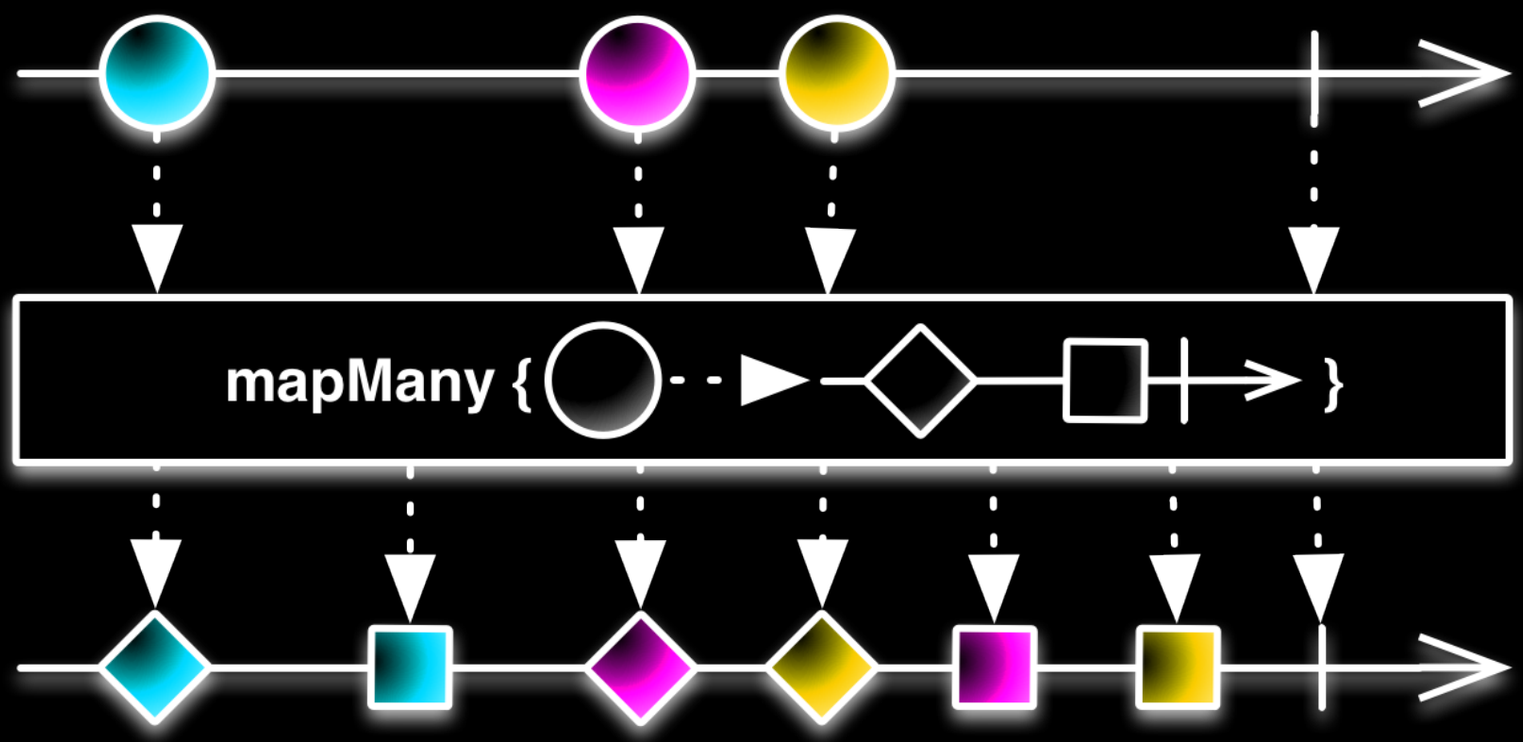
Yet Another Reactive Framework for iOS

Viktor Belenyesi (Prezi)
@bvic23

Native Development Meetup
2015.09.15.

What is RP?

?



?

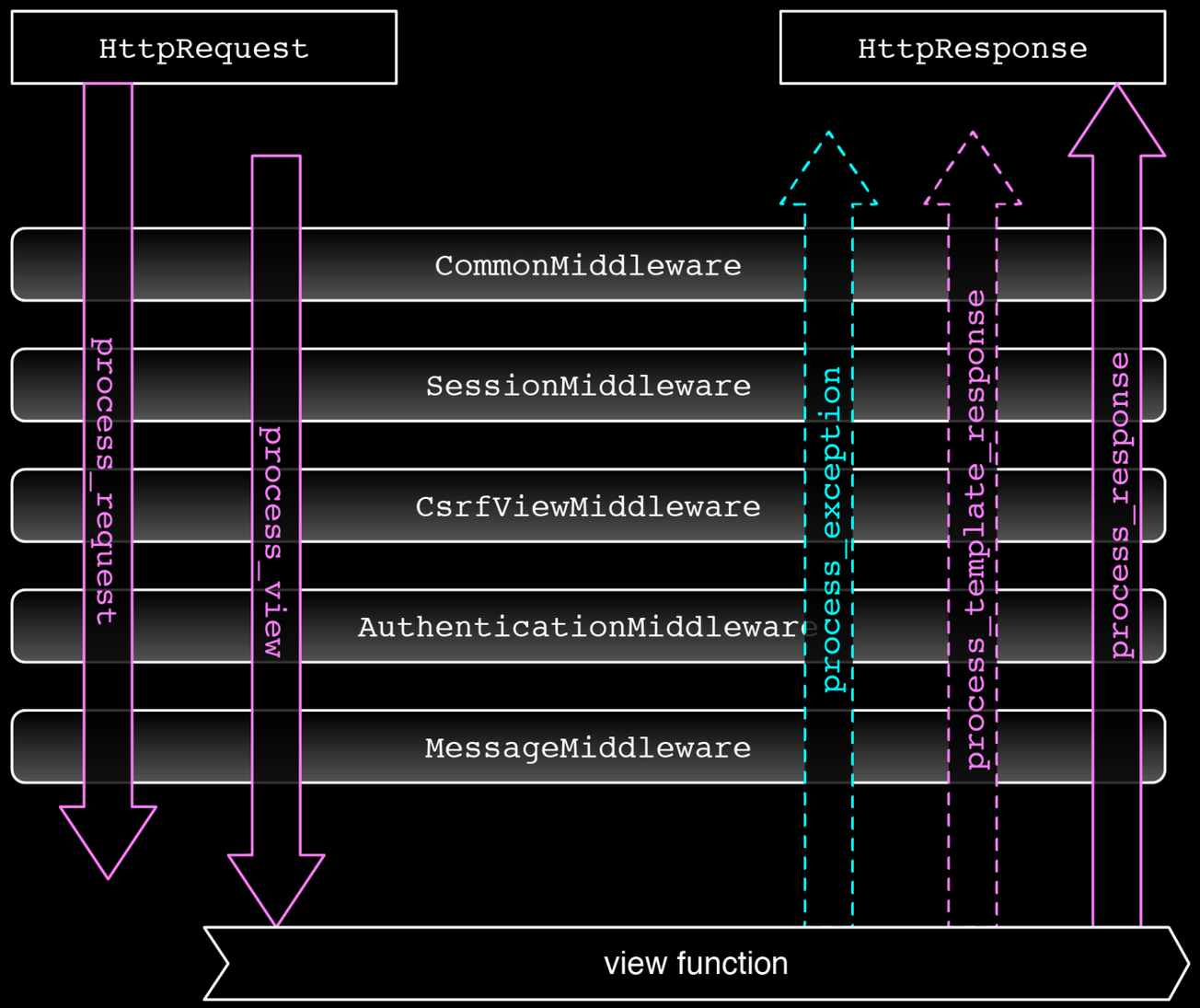
http://rxmarbles.com/

Keep it simple

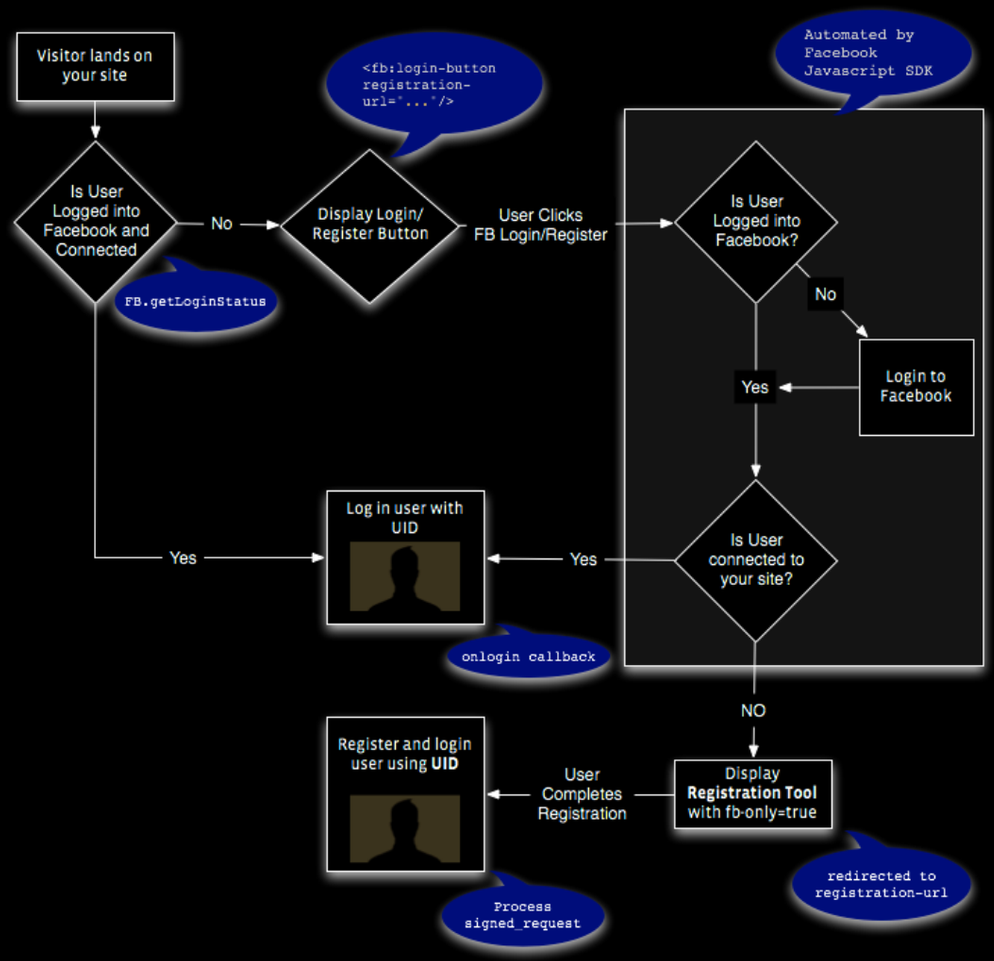
✓

Why?

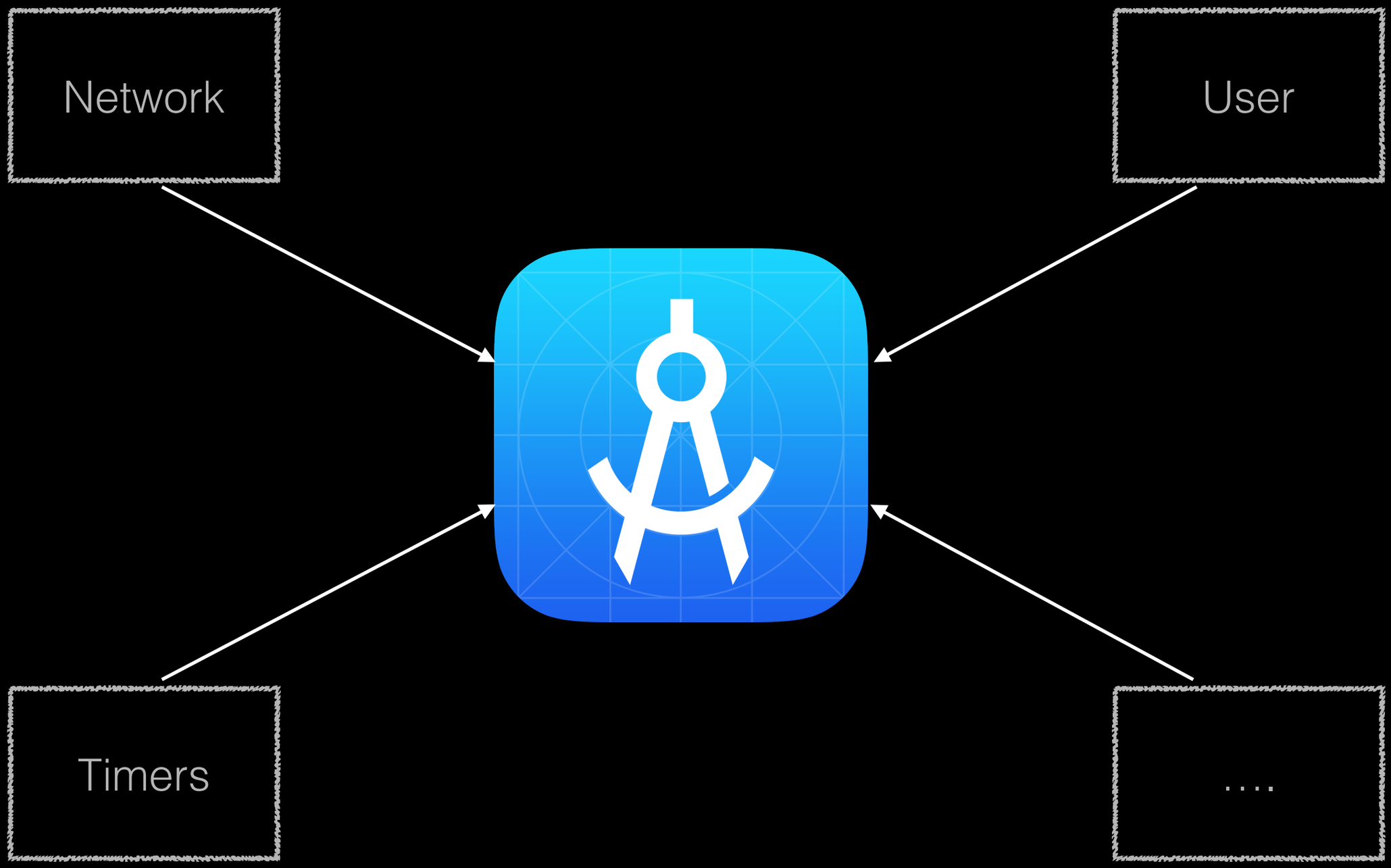
Backend



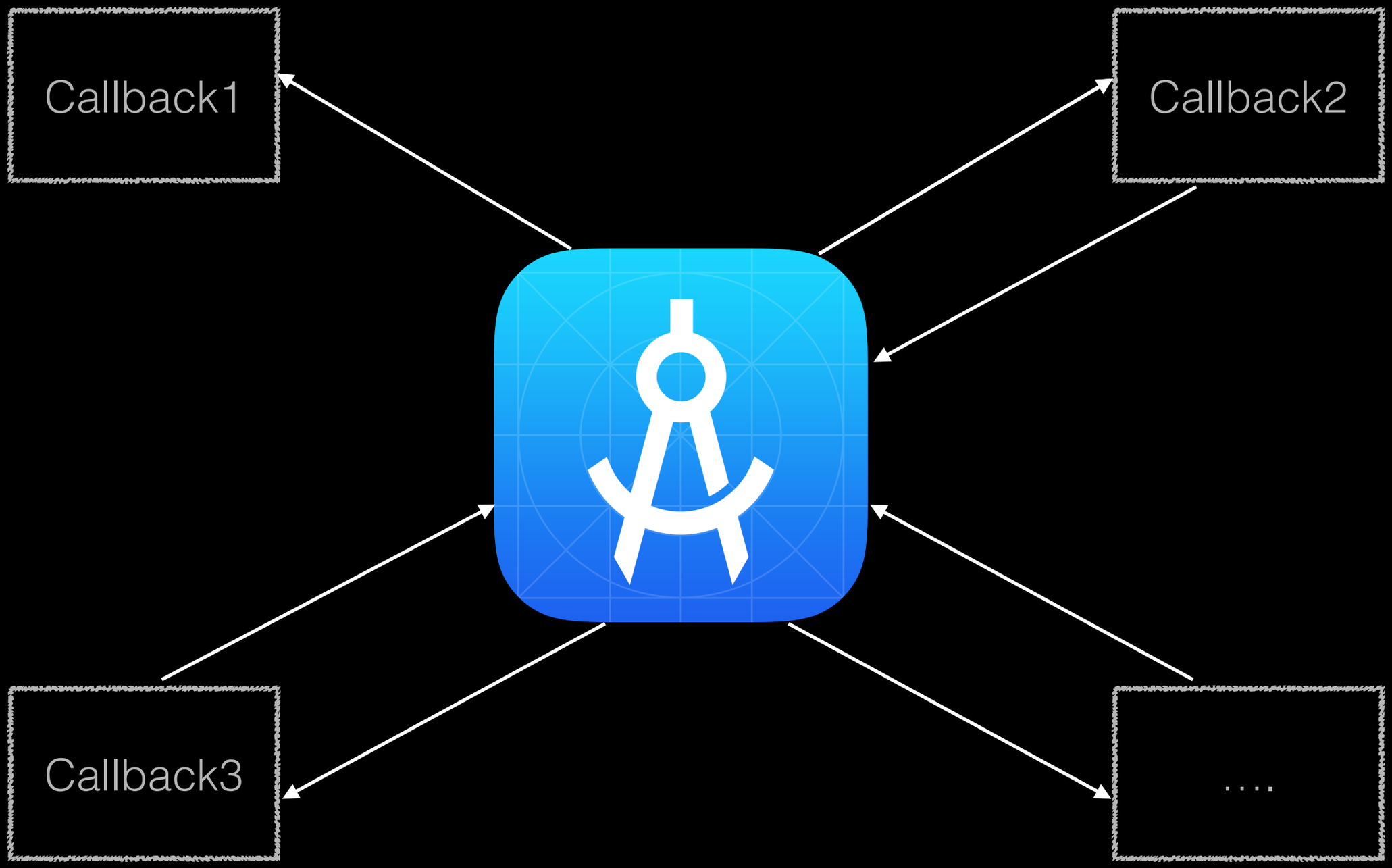
UI



User Inter[face/action]



User Inter[face/action] II



How to handle?

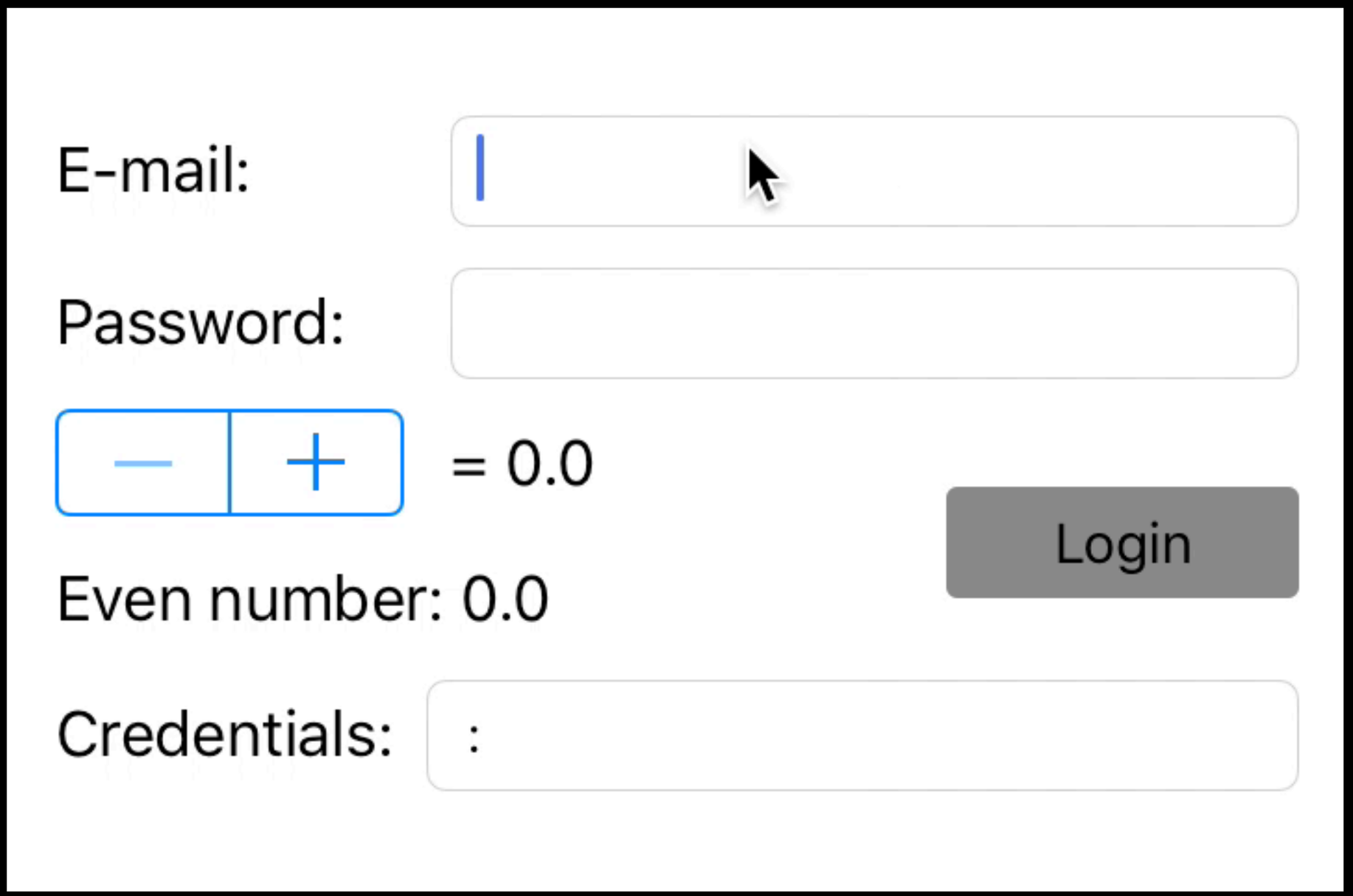
Tests

Design Patterns

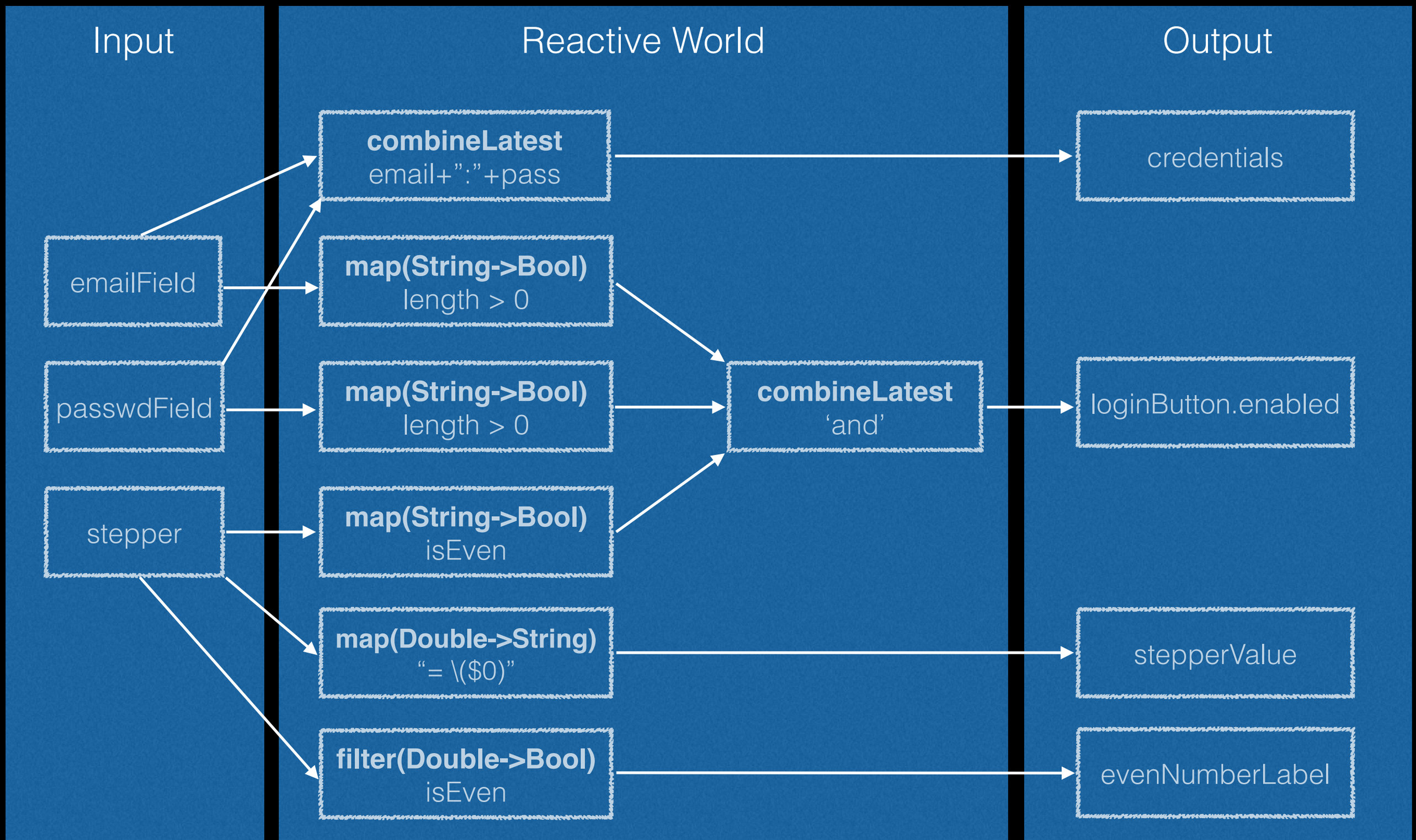
Reactive Programming

Functional Programming

Example - Login UI



Example: Login UI II.



Example: Login UI III.

```
self.loginButton.reactiveEnabled = definedAs {
    self.emailField.reactiveText*.length > 0 &&
    self.passwordField.reactiveText*.length > 0 &&
    self.stepper.reactiveValue.map(isEven)*
}

self.credentialsField.reactiveText = definedAs {
    "\(self.emailField.reactiveText*) : \(self.passwordField.reactiveText*)"
}

self.evenNumberLabel.reactiveText = self.stepper.reactiveValue.filter(isEven).map {
    "Even number: \($0)"
}

self.stepperValueLabel.reactiveText = definedAs {
    " = \(self.stepper.reactiveValue.value())"
}
```

What is *?

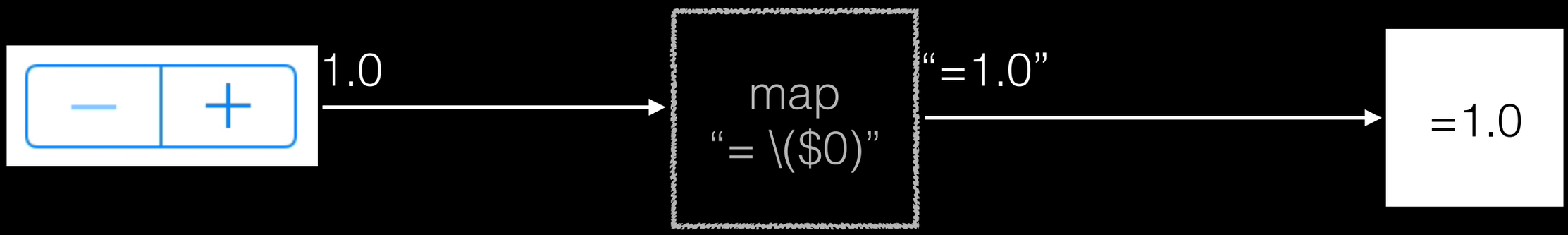
```
self.stepperValueLabel.reactiveText = definedAs {
    " = \(self.stepper.reactiveValue.value())"
}
```

Syntactic sugar

What is definedAs?

```
self.stepperValueLabel.reactiveText = definedAs {
    " = \(self.stepper.reactiveValue*)"
}
```

Map + CombineLatest = Reactive context



What is reactiveX?

```
self.stepperValueLabel.reactiveText = definedAs {
    " = \(self.stepper.reactiveValue*)"
}
```

UIKit extension Reactive property

Left hand side ⟶ Observer

Right hand side ⟶ Emitter (Signal)

Side effects

```
onChangeDo(self.credentialsField.reactiveText) {
    print("credentials: \($0*)")
}
```

```
self.credentialsField.reactiveText.onChange {
    print("credentials: \($0*)")
}
```

Error handling

```
onErrorDo(self.credentialsField.reactiveText) {
    print("errors: \($0*)")
}
```

```
self.credentialsField.reactiveText.onError {
    print("errors: \($0*)")
}
```

Custom emitter / subject

```
// given
let a = reactive(1)
let b = reactive(2)
let c = definedAs {
    a* + b*
}

// when
a <- 2

// then
expect(c*) == 4
```

What is VinceRP

Swift based

Easy to use

Declarative

Reactive Framework

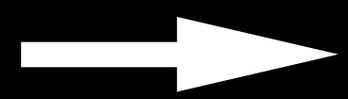
Ingredients

Type safety	Runtime fw, compiler helps
Generics	Support several types
Type inference	Less code is the best code
Extensions	Avoid subclassing
ObjC runtime	Properties + swizzling
FP support	Operators (filter, map, reduce)
Custom operators	*, <-

Ingredients II.

Trailing closure syntax sugar

```
let c = definedAs({ () -> () in
    a* + b*
})
```



```
let c = definedAs {
    a* + b*
}
```

Swift Wishlist

Traits (like in Scala)

No more

```
func ping(incoming: Set<Node>) -> Set<Node> {
    fatalError(ABSTRACT_METHOD)
}
```

Properties to extensions

No more

```
private var textEmitter: Var<String?>? {
    get {
        return objc_getAssociatedObject(self, &AssociatedKeys.textKey) as? Var<String?>
    }
    set {
        objc_setAssociatedObject(self, &AssociatedKeys.textKey, newValue as Var<String?>?,
    }
}
```

Goals

DRY

```
let c = definedAs(a, b) {
    "\(a* + b*)"
}
```

```
let c = definedAs {
    "\(a* + b*)"
}
```

Easy to use/learn

Read the code as a spec

Let’s read together :-)

```
self.loginButton.reactiveEnabled = definedAs {
    self.emailField.reactiveText*.length > 0 &&
    self.passwordField.reactiveText*.length > 0 &&
    self.stepper.reactiveValue.map(isEven)*
}

self.credentialsField.reactiveText = definedAs {
    "\(self.emailField.reactiveText*) : \(self.passwordField.reactiveText*)"
}

self.evenNumberLabel.reactiveText = self.stepper.reactiveValue.filter(isEven).map {
    "Even number: \($0)"
}

self.stepperValueLabel.reactiveText = definedAs {
    " = \(self.stepper.reactiveValue.value())"
}
```

Future

OpenSource on GitHub

Async extension

Timer, debounce, inspection

Bugfixes

Optimize

Who is Vince?

Thank You!

@nativedevmeetup, @bvic23

Q & A