#WWDC19

Introducing SwiftUI

Building Your First App

Jacob Xiao, SwiftUI Engineer
Kyle Macomber, SwiftUI Engineer

Demo

The Way Views Work

Kyle Macomber, SwiftUI Engineer

```
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

```
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

```
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

```
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

A View Defines a Piece of UI

```
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

```
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

```swift
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

```
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

```swift
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

```
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

```
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

```
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

```
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

```
struct RoomDetail : View {
    let room: Room

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

A View Defines its Dependencies

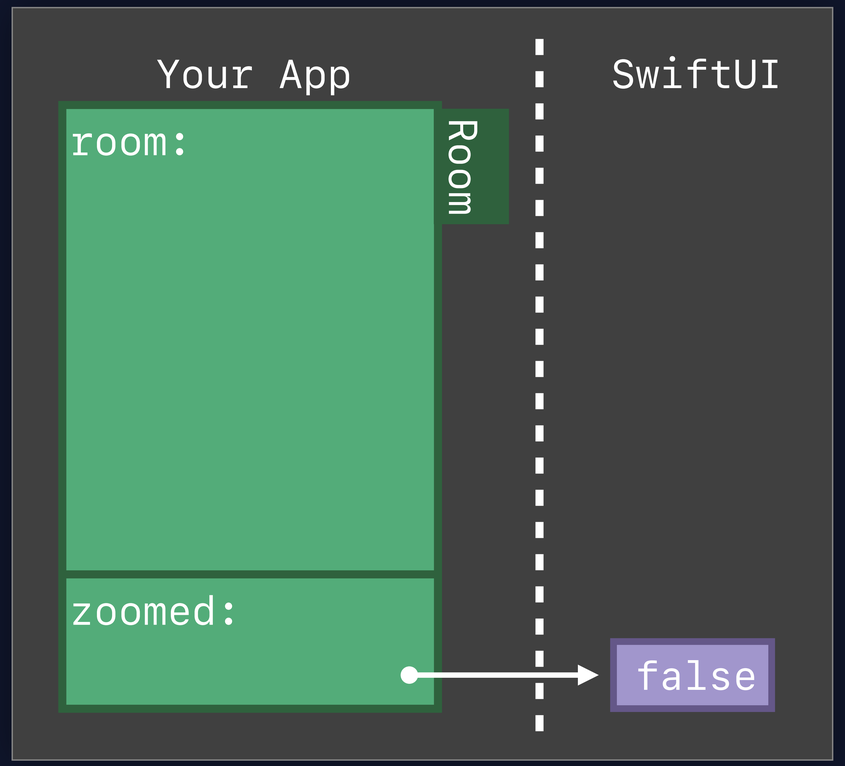

9:41
Rooms
Observation Deck


9:41
Rooms
Observation Deck


9:41
Rooms
Observation Deck

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```



```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: .fit)
    }
}
```

false

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
    }
}
```

false

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false                    true

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
    }
}
```

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false                    true

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
    }
}
```

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false
    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
    }
}
```

false

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

false

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false                    false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

false

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

false

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false                    true

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false                    true

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false                    true

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false                    false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

false

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

false

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

false

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

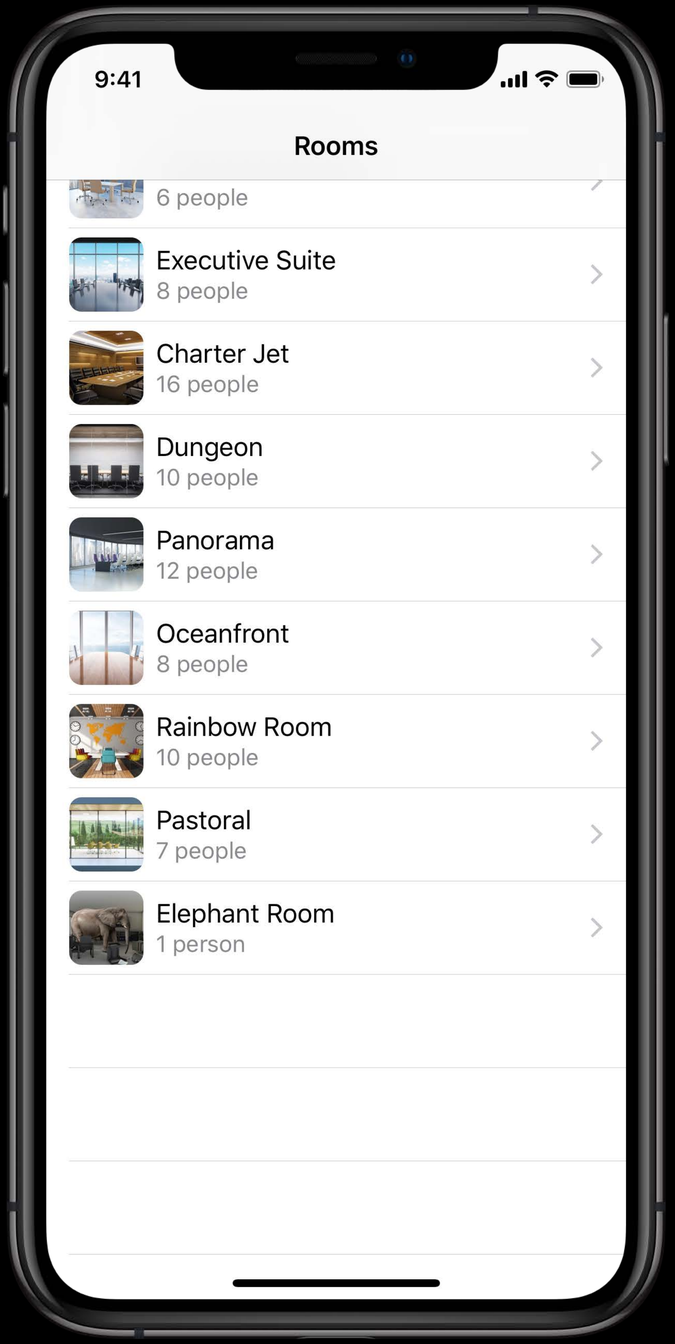
true

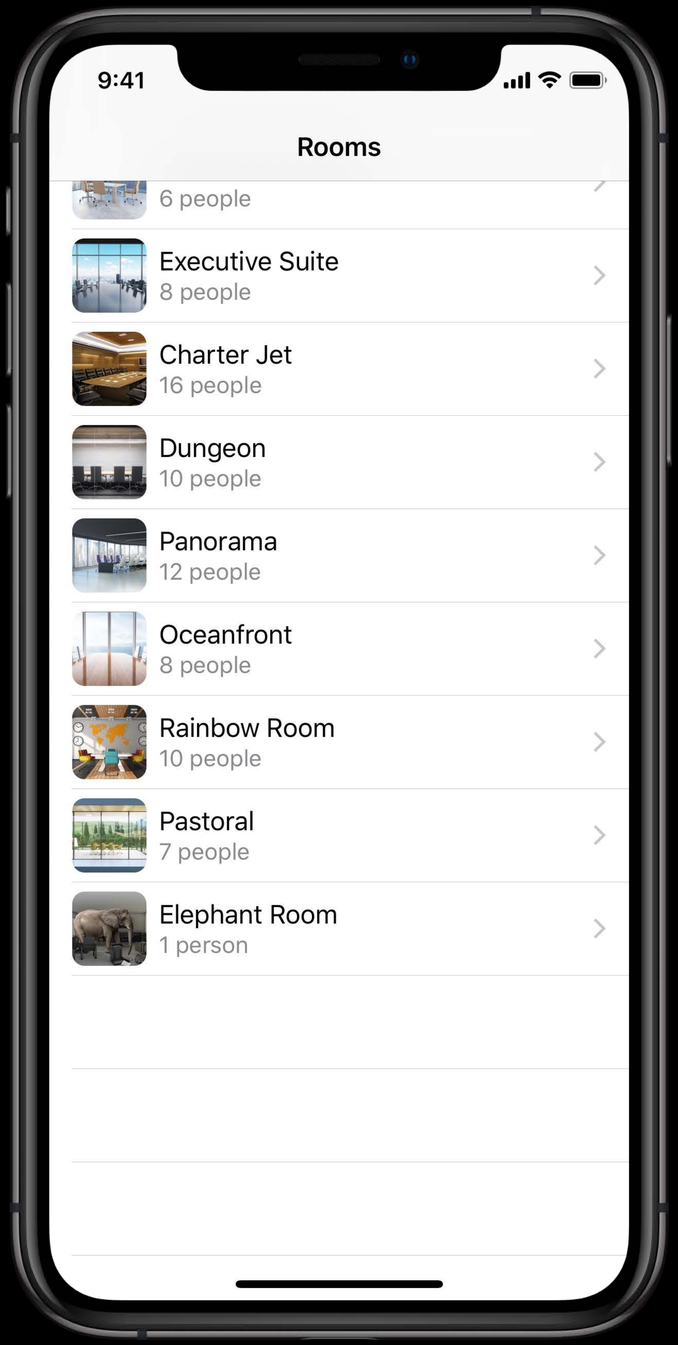
```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

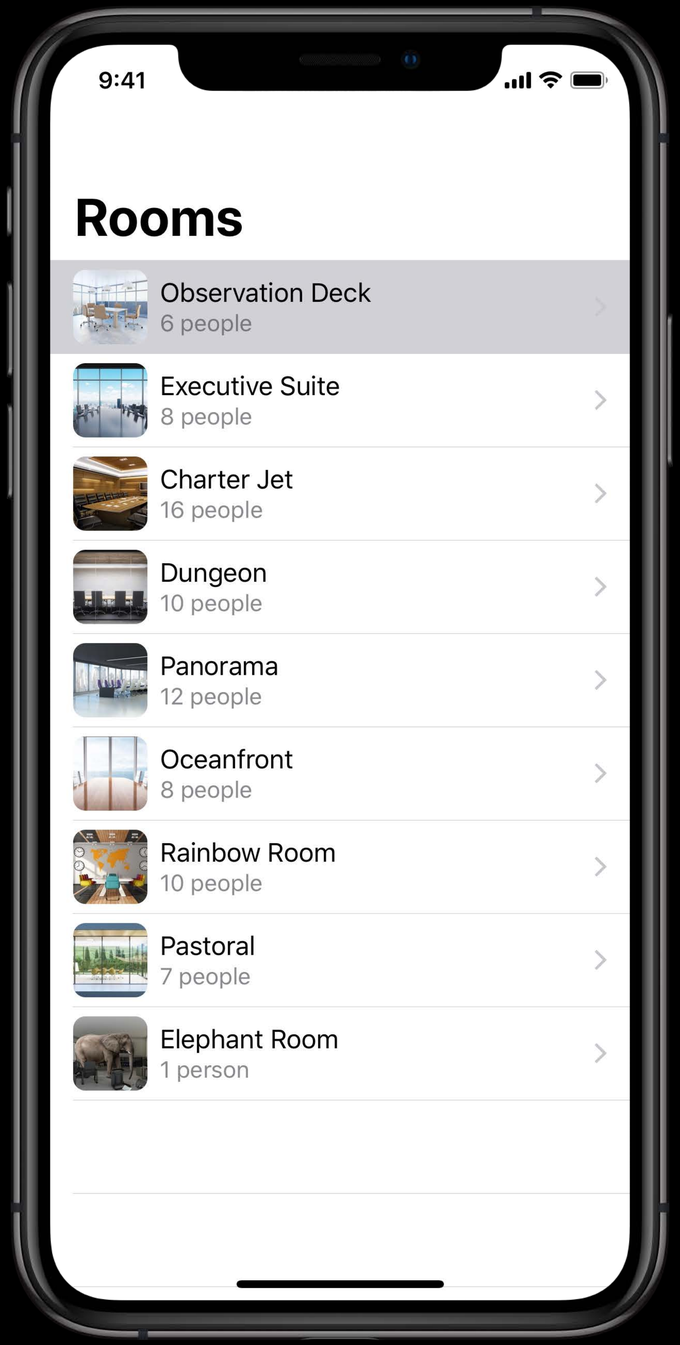
true

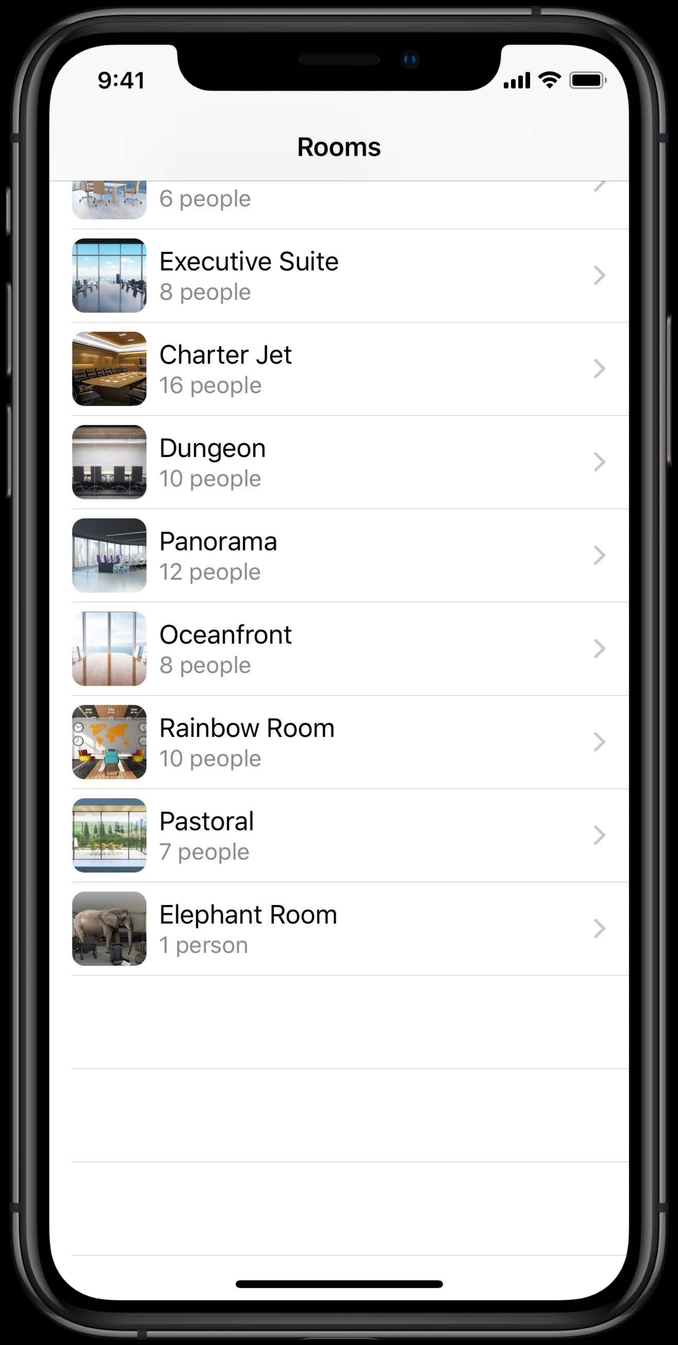
```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false                    true

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

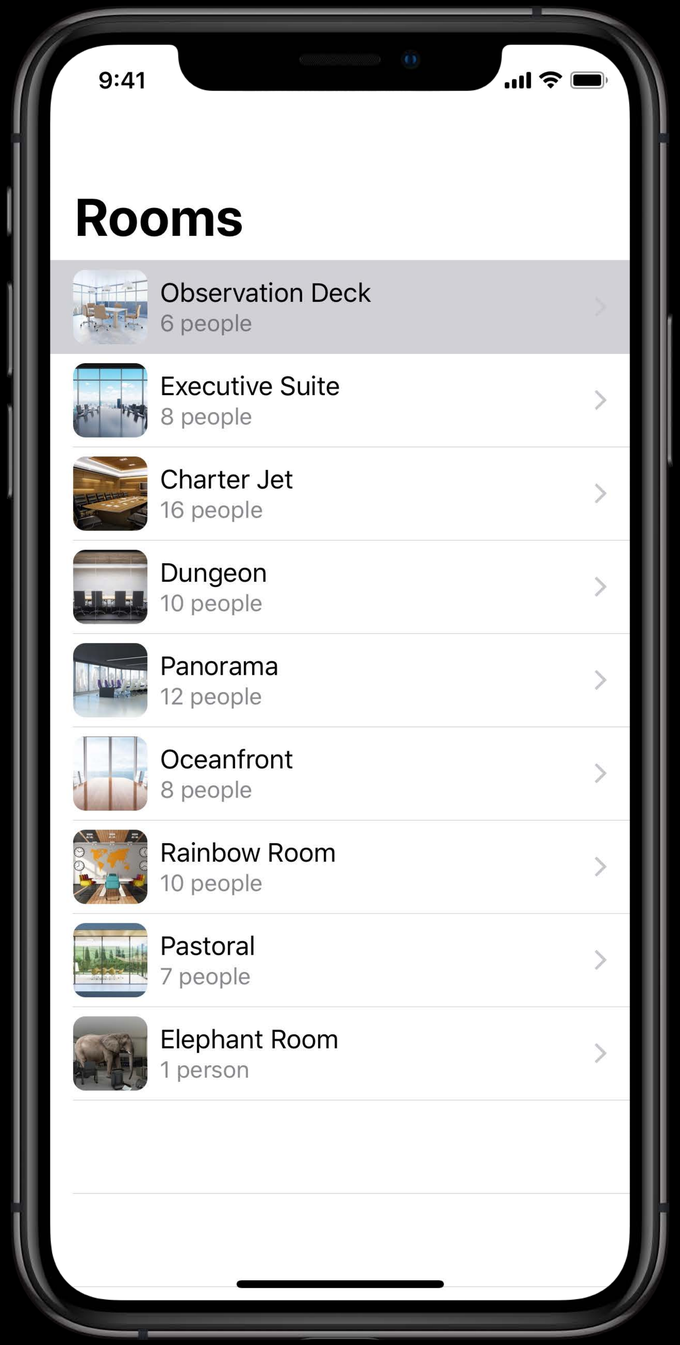
Where Is Truth?


9:41
Rooms
6 people
Executive Suite
8 people
Charter Jet
16 people
Dungeon
10 people
Panorama
12 people
Oceanfront
8 people
Rainbow Room
10 people
Pastoral
7 people
Elephant Room
1 person


9:41
Rooms
6 people
Executive Suite
8 people
Charter Jet
16 people
Dungeon
10 people
Panorama
12 people
Oceanfront
8 people
Rainbow Room
10 people
Pastoral
7 people
Elephant Room
1 person


9:41
Rooms
Observation Deck
6 people
Executive Suite
8 people
Charter Jet
16 people
Dungeon
10 people
Panorama
12 people
Oceanfront
8 people
Rainbow Room
10 people
Pastoral
7 people
Elephant Room
1 person


9:41
Rooms
6 people
Executive Suite
8 people
Charter Jet
16 people
Dungeon
10 people
Panorama
12 people
Oceanfront
8 people
Rainbow Room
10 people
Pastoral
7 people
Elephant Room
1 person


9:41
Rooms
Observation Deck
6 people
Executive Suite
8 people
Charter Jet
16 people
Dungeon
10 people
Panorama
12 people
Oceanfront
8 people
Rainbow Room
10 people
Pastoral
7 people
Elephant Room
1 person


9:41
Rooms
Observation Deck

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}
```

```
struct AspectRatioView : View {
    let contentMode: ContentMode
    var body: some View { ... }
}
```

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}

struct AspectRatioView : View {
    let contentMode: ContentMode
    var body: some View { ... }
}
```

Source of Truth

Derived Value

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}

struct AspectRatioView : View {
    let contentMode: ContentMode
    var body: some View { ... }
}
```

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}

struct AspectRatioView : View {
    let contentMode: ContentMode
    var body: some View { ... }
}
```

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}

struct AspectRatioView : View {
    let contentMode: ContentMode
    var body: some View { ... }
}
```

false

.fit

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}

struct AspectRatioView : View {
    let contentMode: ContentMode
    var body: some View { ... }
}
```

false

.fit

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}

struct AspectRatioView : View {
    let contentMode: ContentMode
    var body: some View { ... }
}
```

false

.fit

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}

struct AspectRatioView : View {
    let contentMode: ContentMode
    var body: some View { ... }
}
```

true

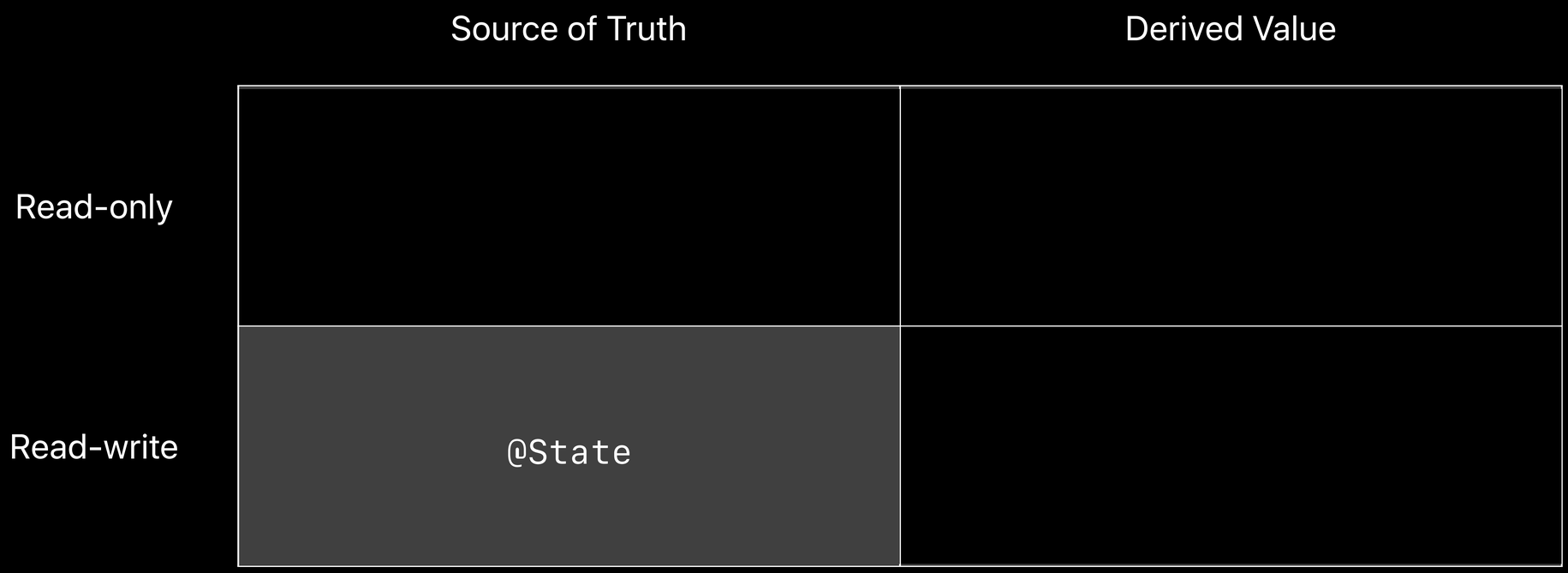
.fit

```
struct RoomDetail : View {
    let room: Room
    @State private var zoomed = false
    
    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction { self.zoomed.toggle() }
    }
}

struct AspectRatioView : View {
    let contentMode: ContentMode
    var body: some View { ... }
}
```

true

.fill

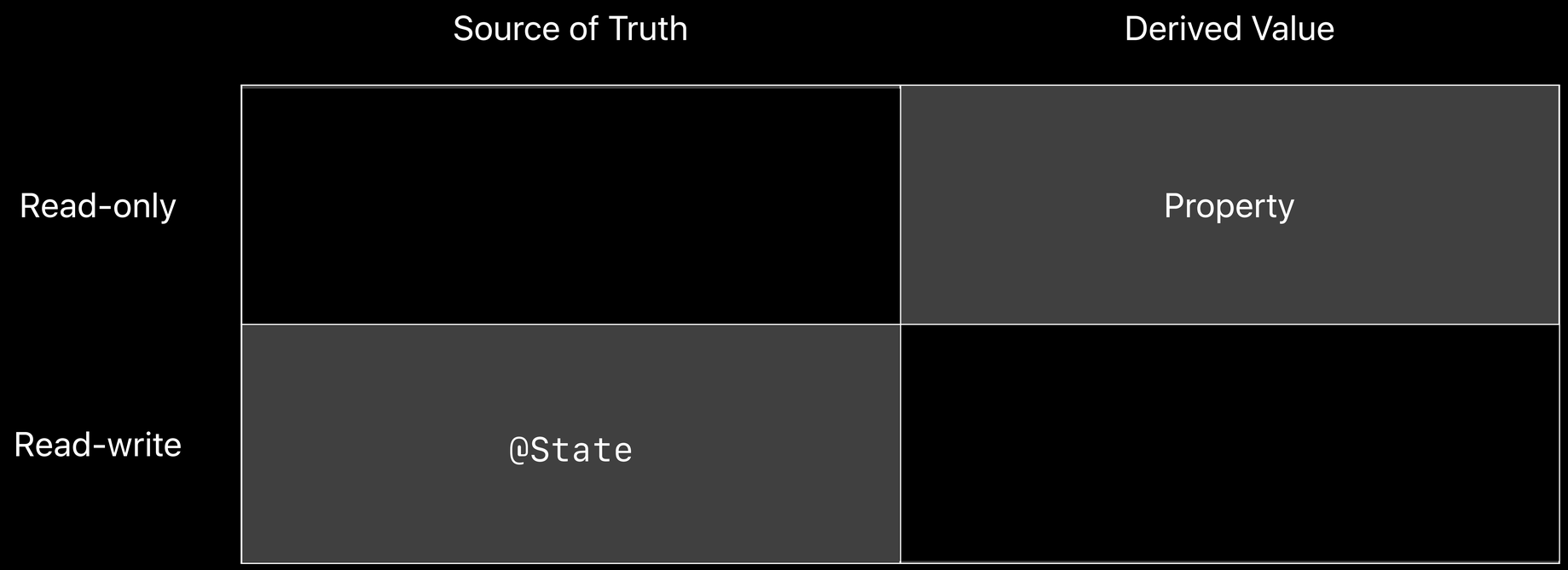
Data Flow Primitives



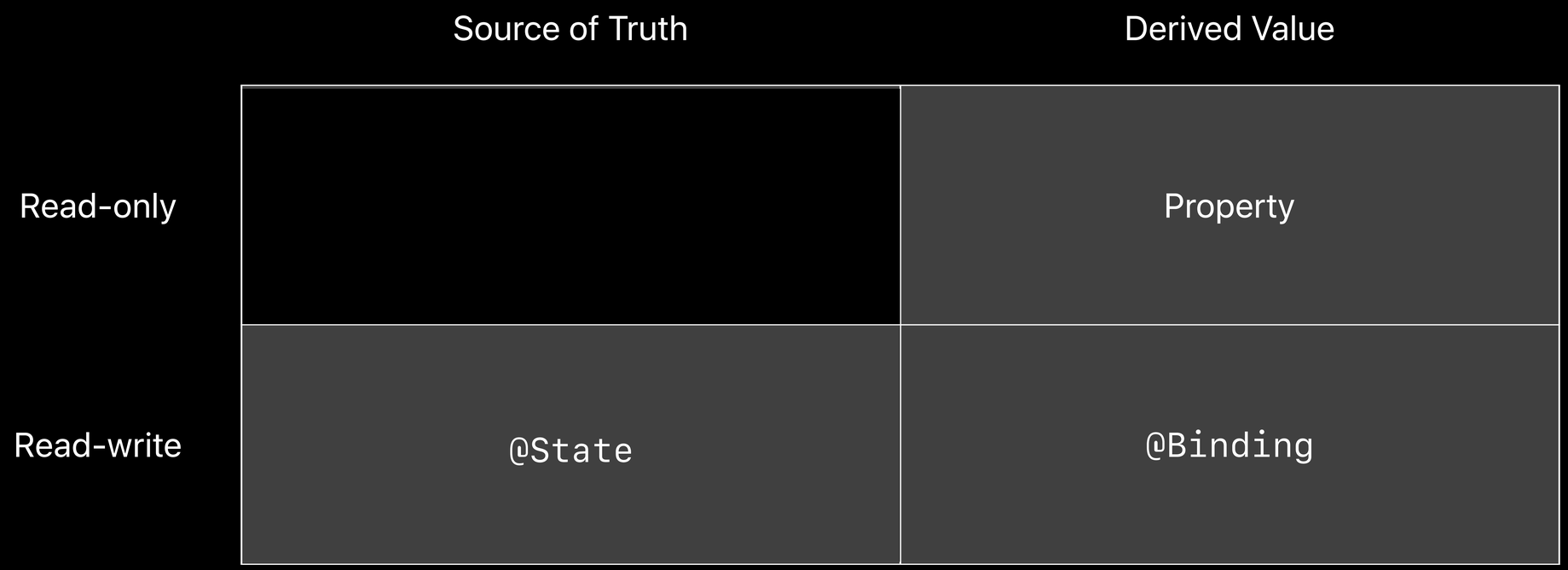
Data Flow Through SwiftUI

Thursday, 9:00

Data Flow Primitives



Data Flow Primitives



Data Flow Through SwiftUI

Thursday, 9:00

Data Flow Primitives

	Source of Truth	Derived Value
Read-only	Constant	Property
Read-write	`@State`	`@Binding`

Data Flow Through SwiftUI

Thursday, 9:00

Data Flow Primitives

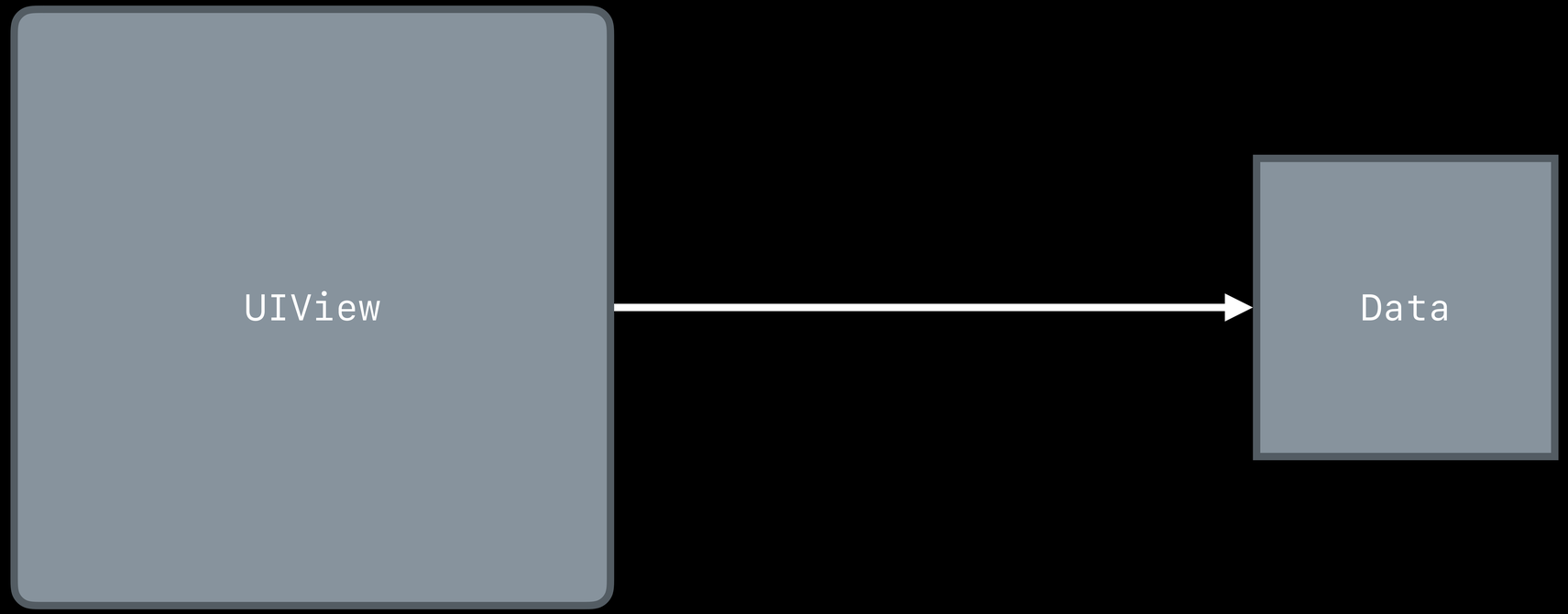
	Source of Truth	Derived Value
Read-only	Constant	Property
Read-write	`@State` `BindableObject`	`@Binding`

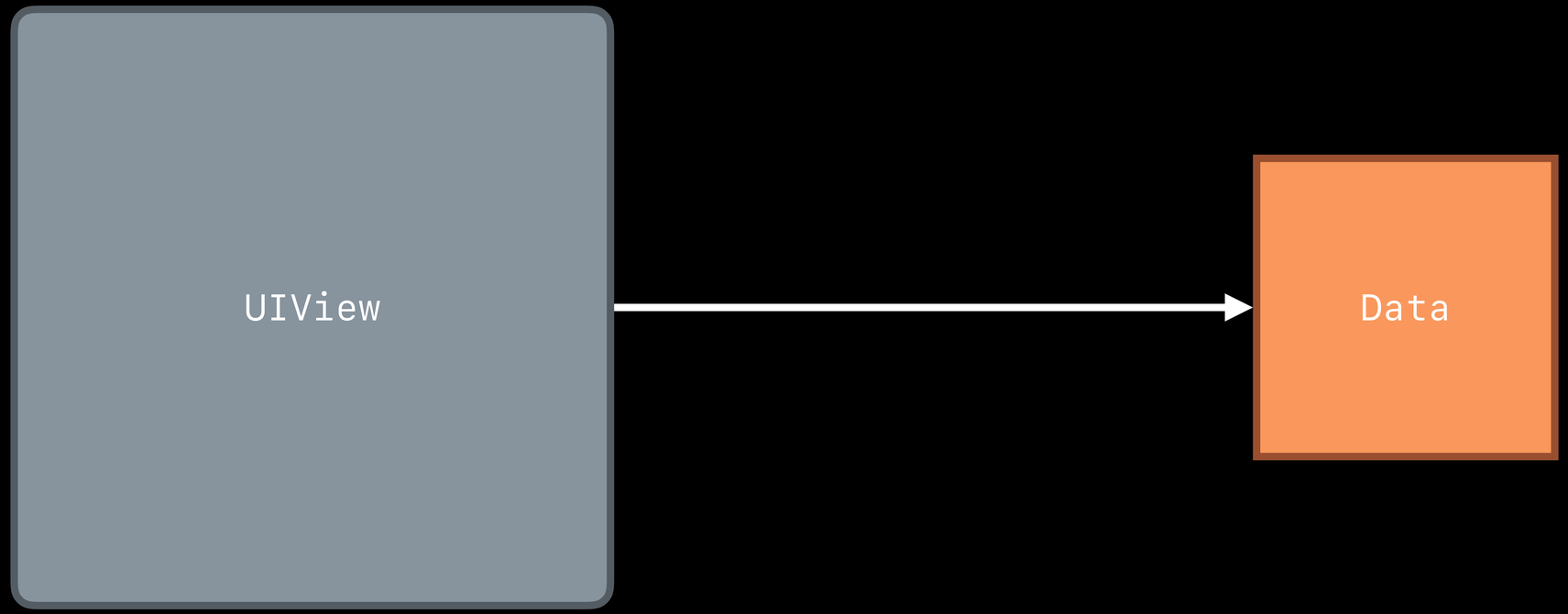
Data Flow Through SwiftUI

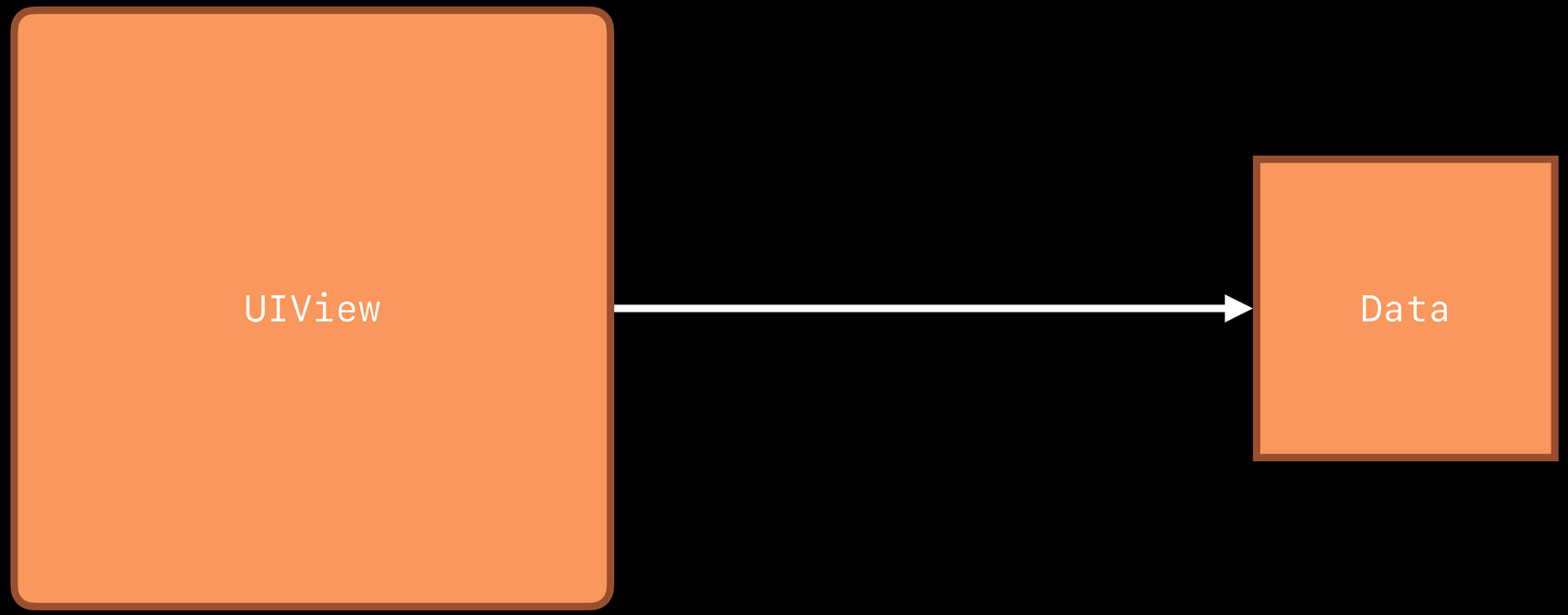
Thursday, 9:00

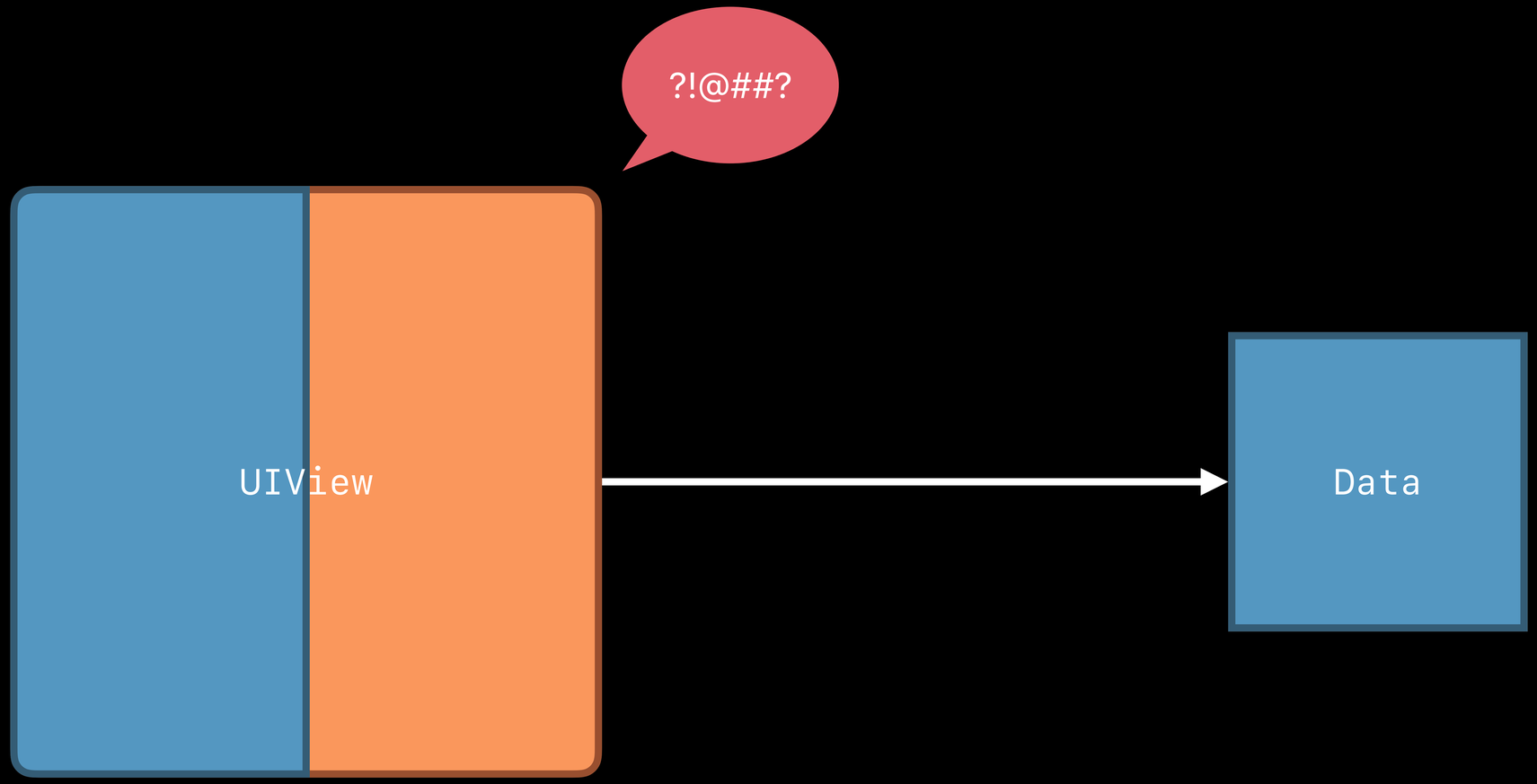
Managing Dependencies Is Hard

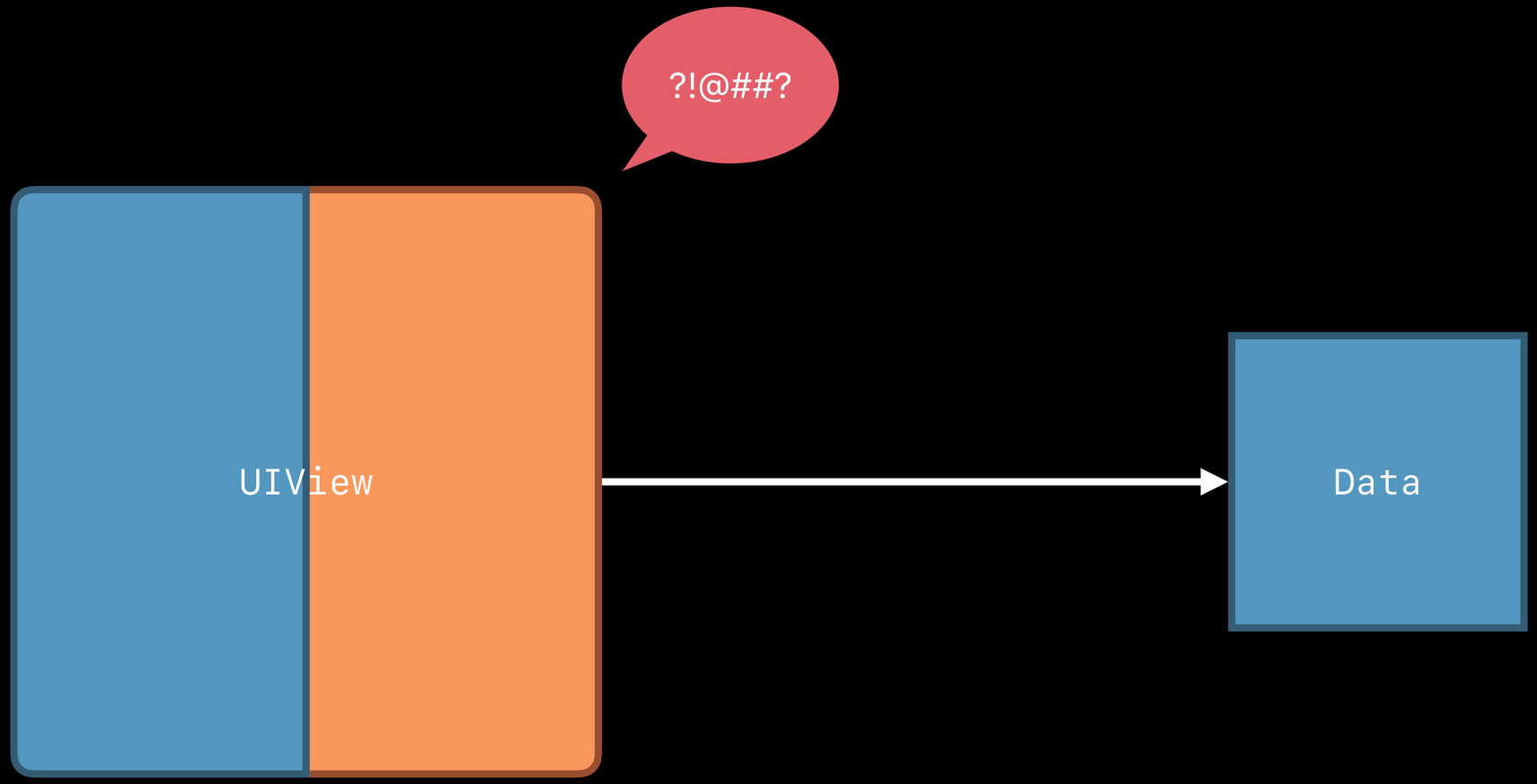

UIView
Data

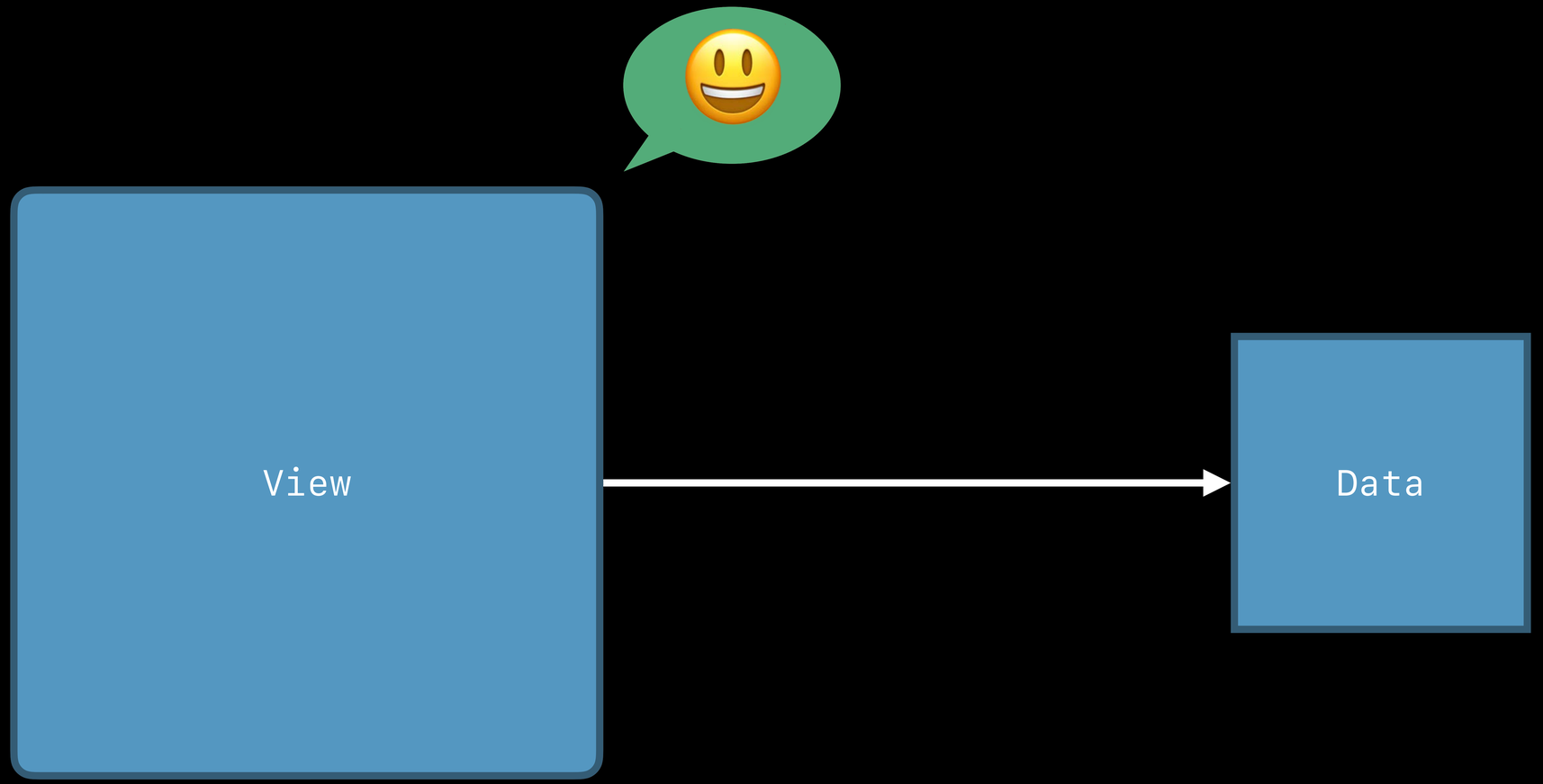

UIView
Data


UIView
Data


UIView
Data

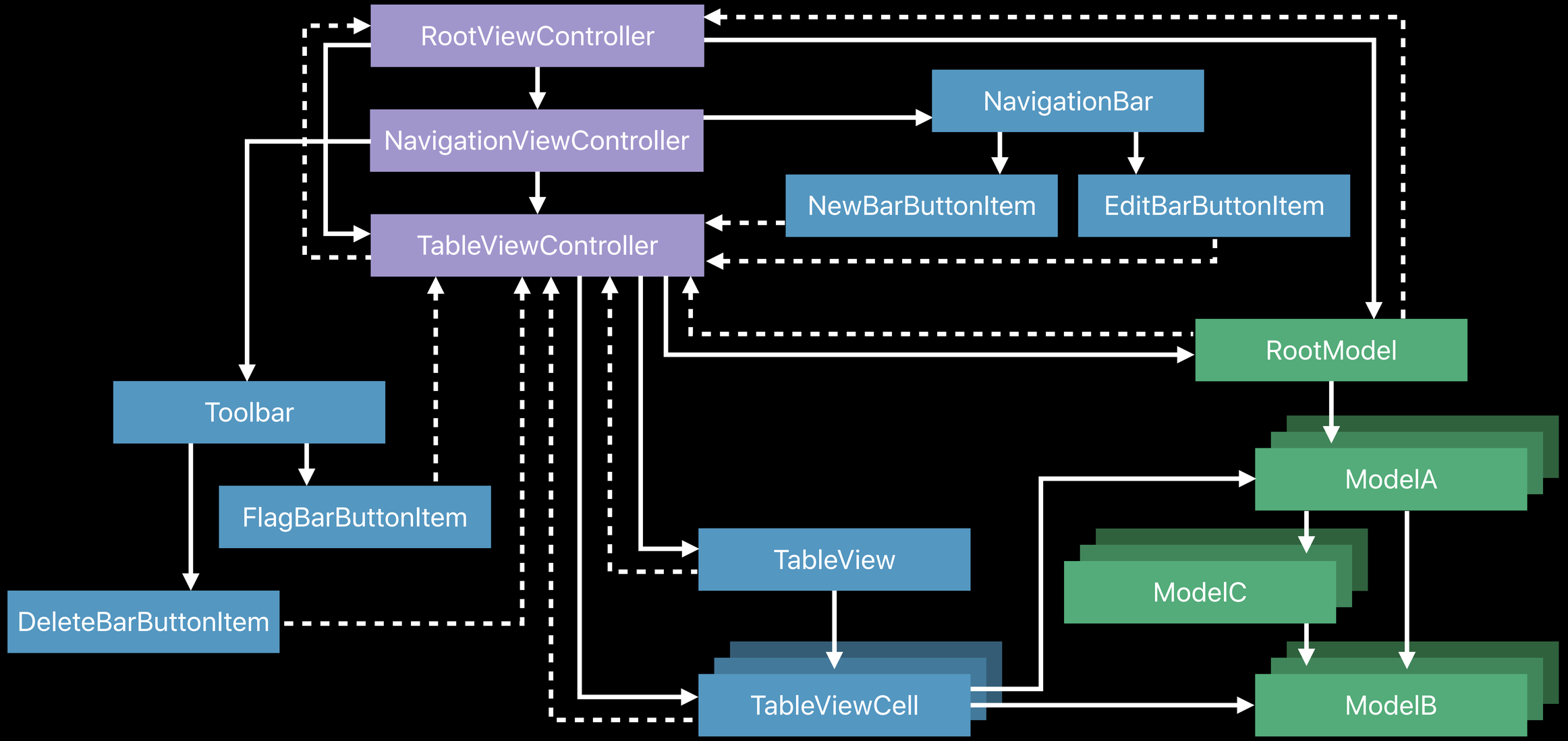

?!@##?
UIView
Data


?!@##?
UIView
Data

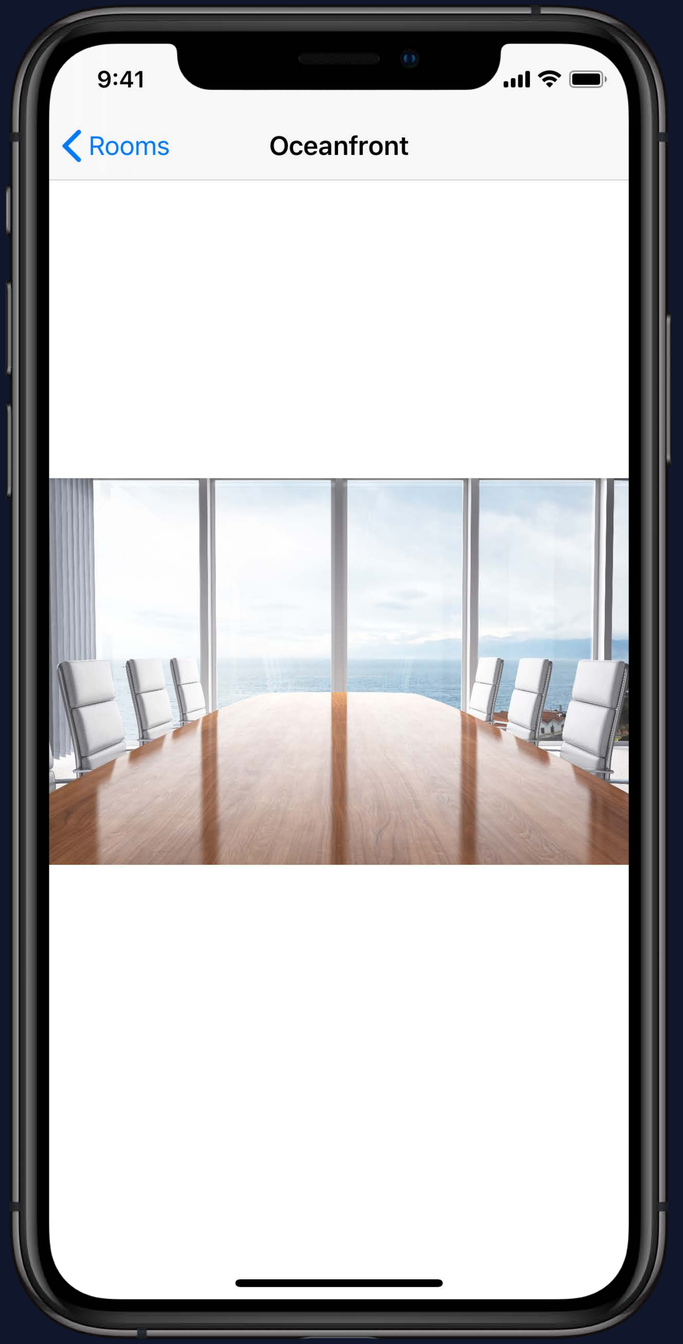

View
Data

Managing Dependencies Is Hard

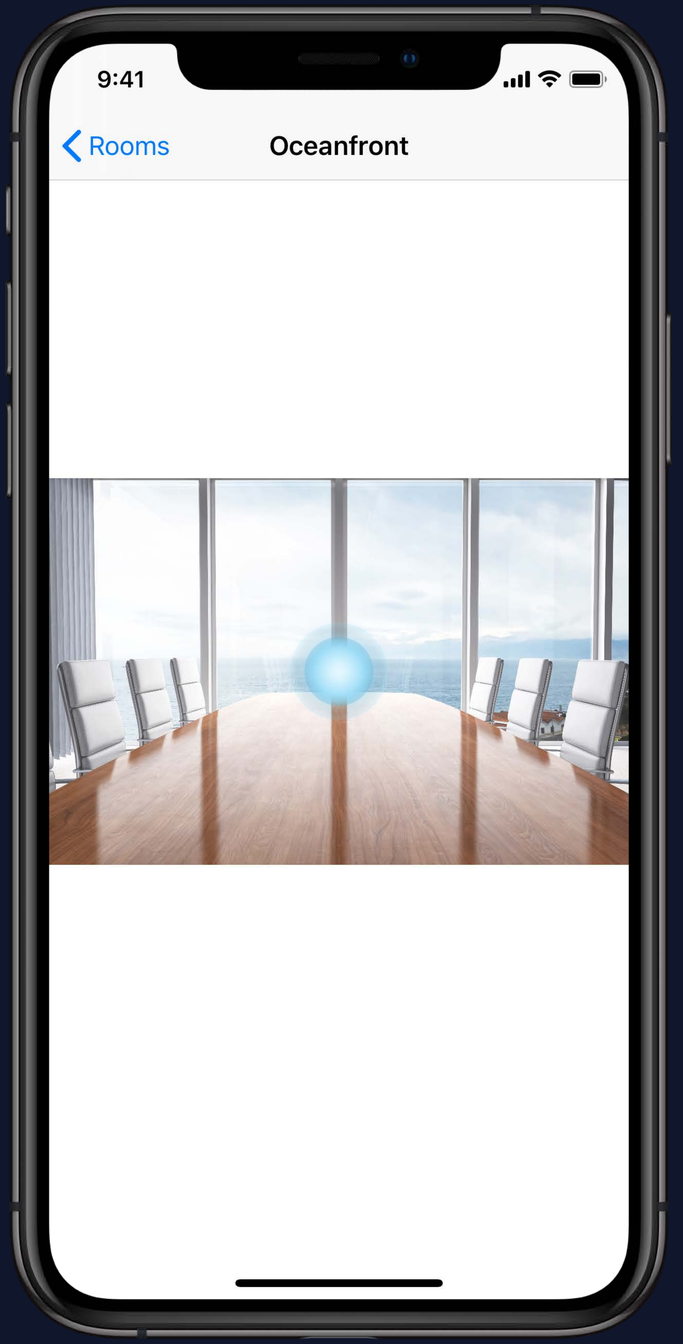
Managing Dependencies Is Hard



```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```



```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

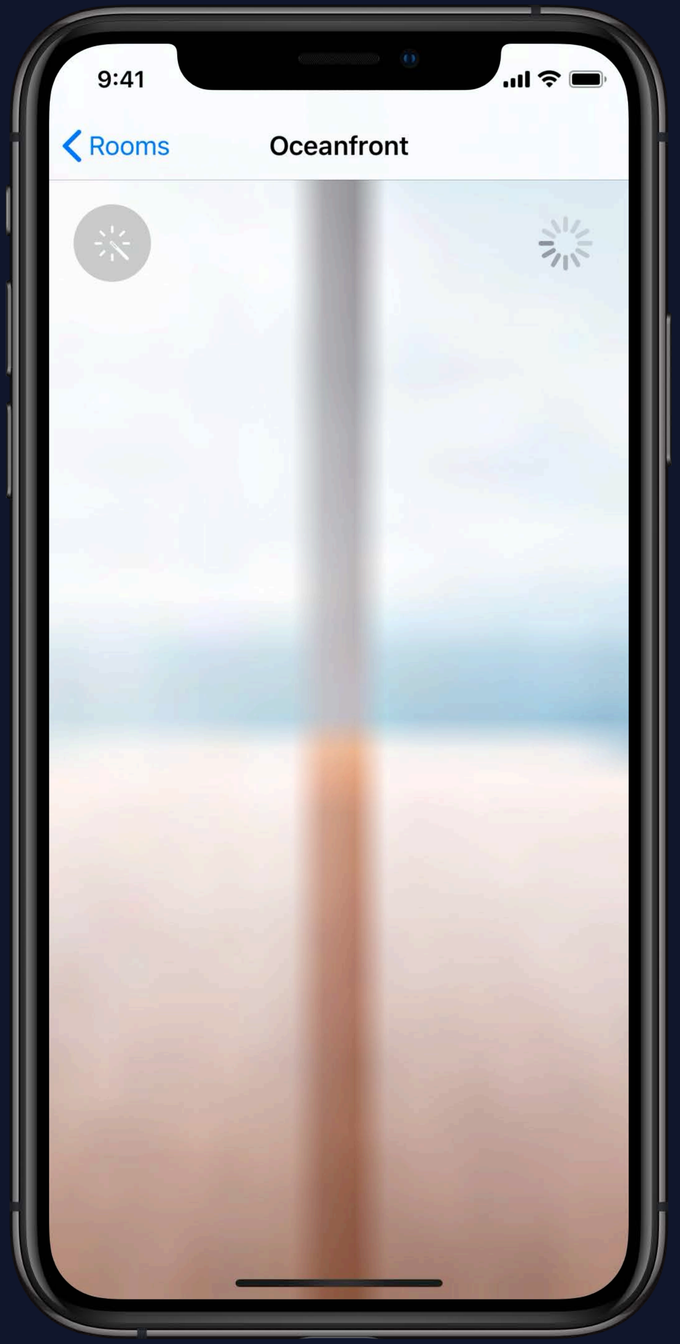


```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

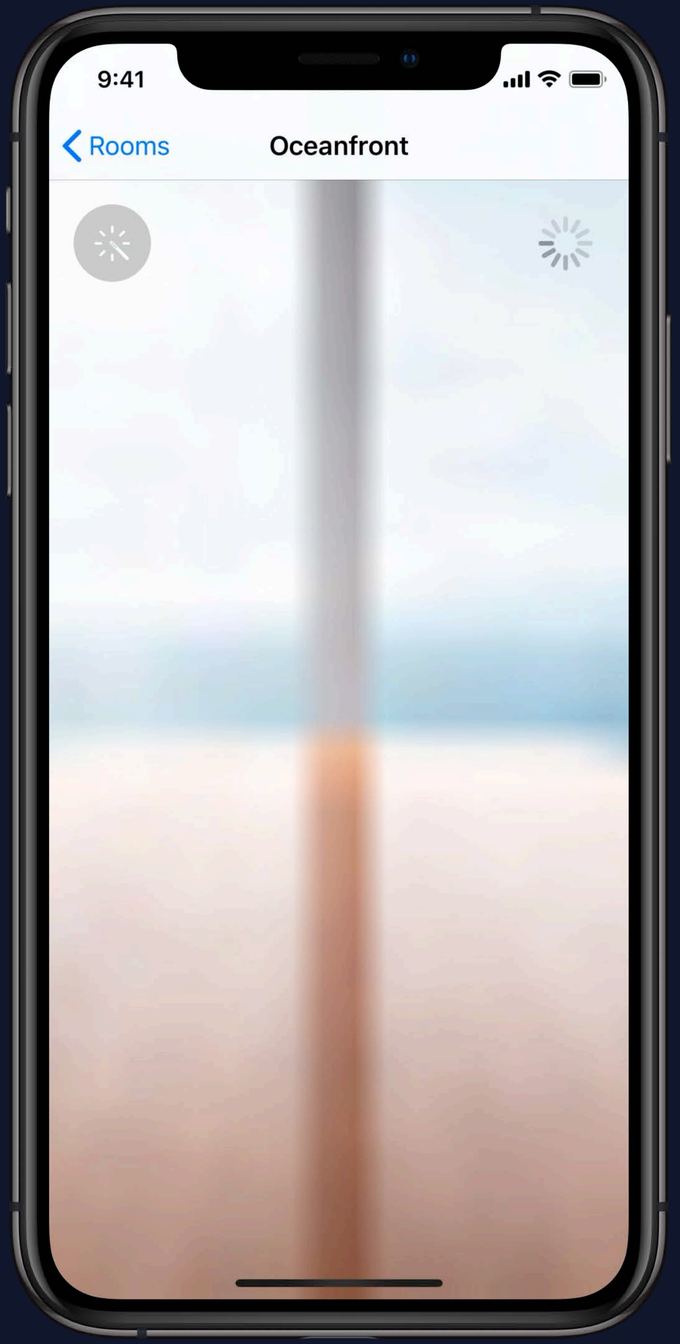
```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

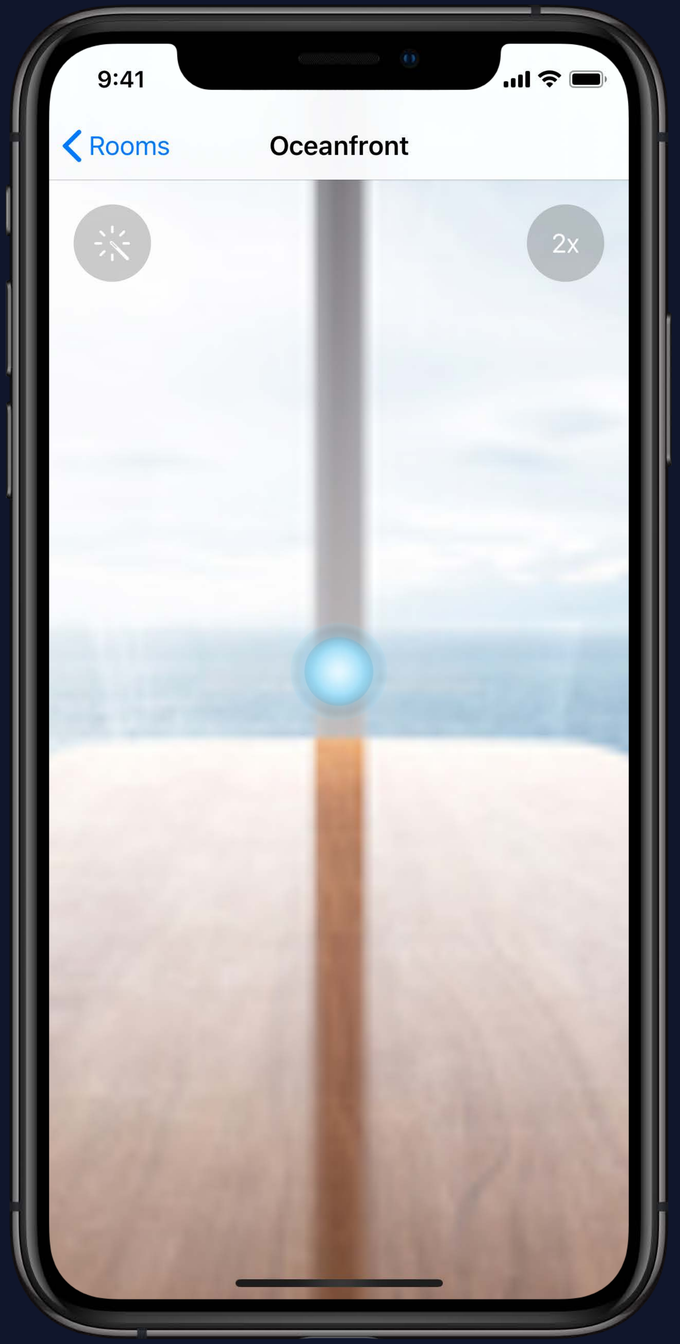


```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

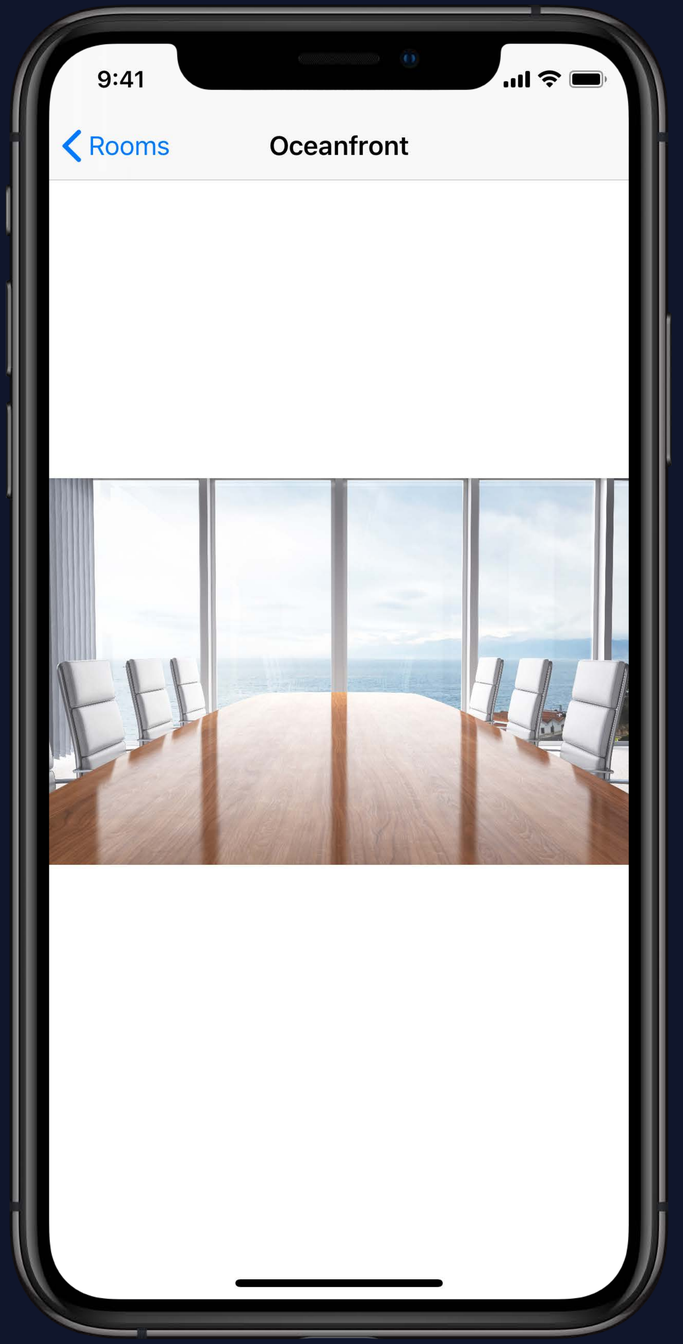


```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

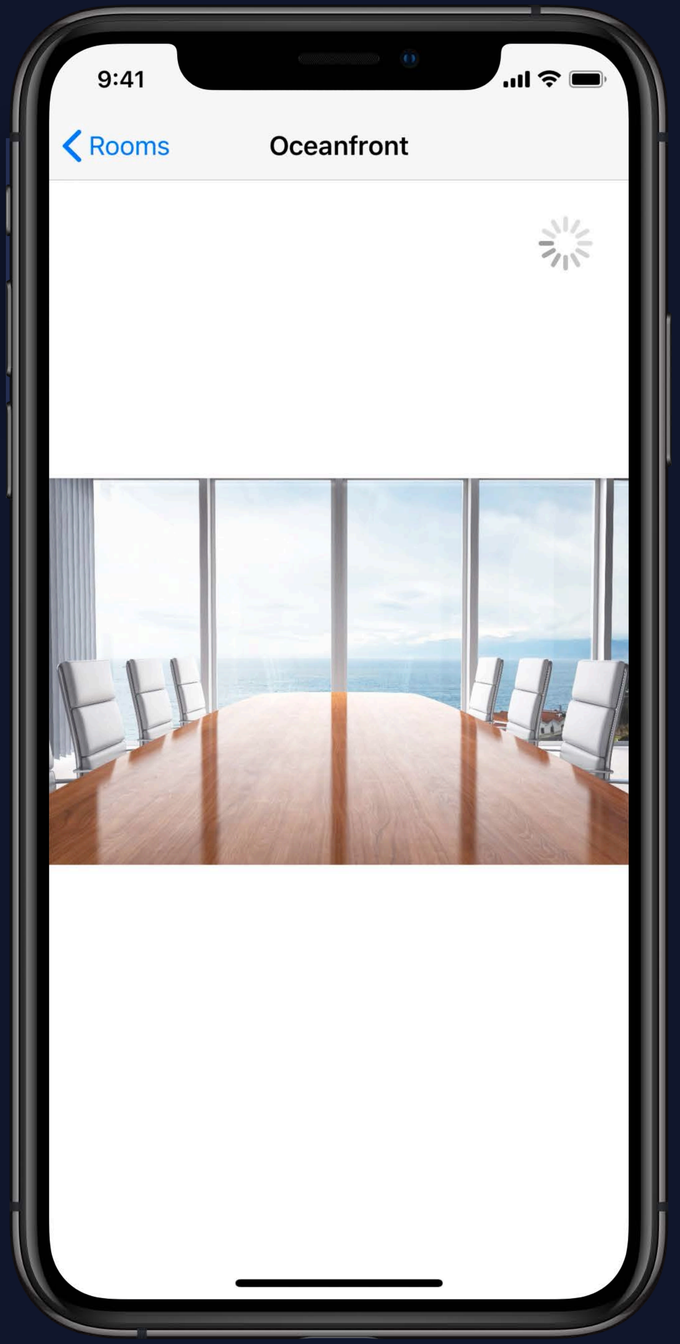
```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```



```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

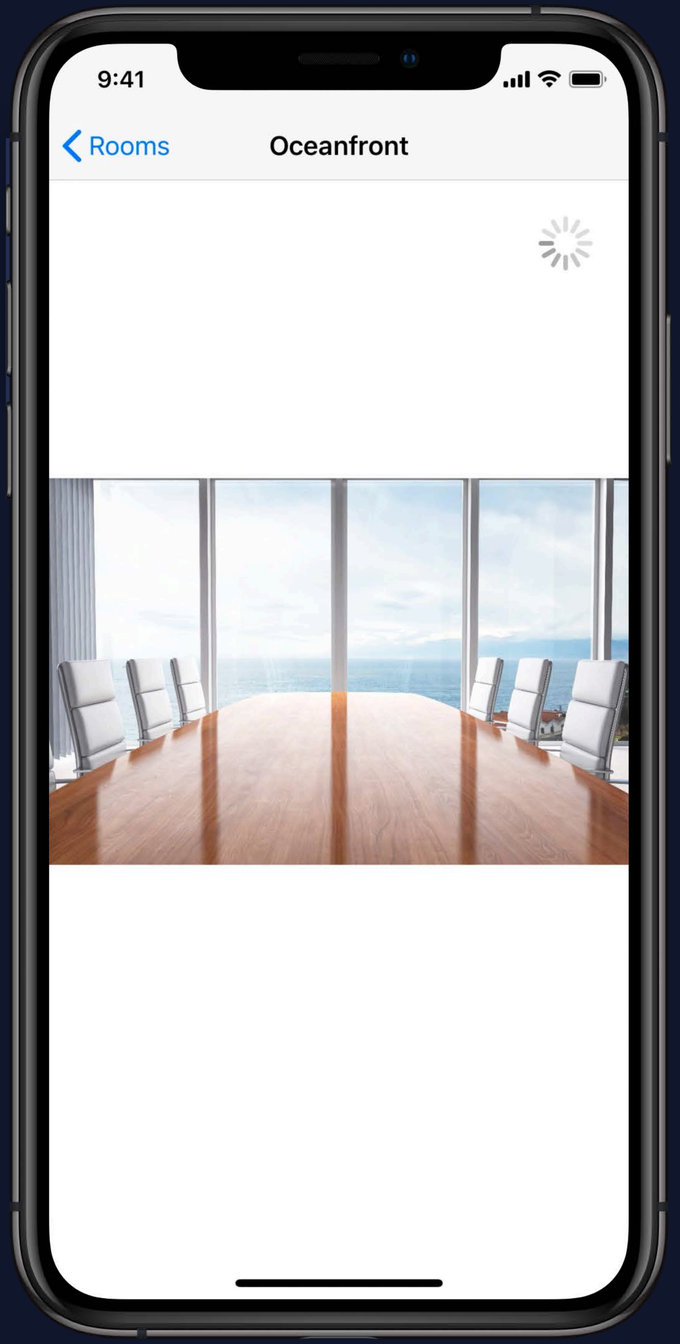


```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```



```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

?!@##?



```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
           // if zoomed hide activity indicator, show scale badge
           // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

```
zoomIn
zoomOut
enhance
completion
```

```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
            // if zoomed hide activity indicator, show scale badge
            // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

```
zoomIn
zoomOut
enhance
completion
```

```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
           // if zoomed hide activity indicator, show scale badge
           // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

```
zoomIn
enhance
completion
zoomOut
```

```
class RoomDetailViewController : UIViewController {
    func zoomIn() {
        // scale image to fill, show enhance button
        // if enhancing show activity indicator
        // if enhanced show scale badge
    }
    func enhance() {
        // show activity indicator
        enhancer.enhance(room.image, scale: scale) {
           // if zoomed hide activity indicator, show scale badge
           // set enhanced image
        }
    }
    func zoomOut() {
        // scale image to fit, hide enhance button
        // hide activity indicator, hide scale badge
    }
}
```

```
enhance
zoomOut
zoomIn
completion
```

zoomOut
zoomIn
completion
enhance

zoomIn
zoomOut
completion
enhance

completion
zoomOut
zoomIn
enhance

enhance
zoomOut
zoomIn
completion

zoomOut
zoomIn
enhance
completion

zoomIn
enhance
zoomOut
completion

completion
zoomOut
enhance
zoomIn

enhance
zoomOut
completion
zoomIn

zoomOut
completion
zoomIn
enhance

zoomIn
completion
zoomOut
enhance

completion
zoomIn
zoomOut
enhance

enhance
zoomIn
zoomOut
completion

zoomOut
zoomIn
completion
enhance

zoomIn
zoomOut
completion
enhance

completion
zoomOut
zoomIn
enhance

enhance
zoomOut
completion
zoomIn

zoomOut
zoomIn
enhance
completion

zoomIn
enhance
zoomOut
completion

completion
zoomOut
enhance
zoomIn

enhance
zoomOut
completion
zoomIn

zoomOut
completion
zoomIn
enhance

zoomIn
completion
zoomOut
enhance

completion
zoomIn
zoomOut
enhance

enhance
zoomIn
zoomOut
completion

zoomOut
completion
zoomIn
enhance

zoomIn
completion
zoomOut
enhance

completion
zoomIn
zoomOut
enhance

enhance
zoomIn
zoomOut
completion

zoomOut
completion
enhance
zoomIn

zoomIn
completion
enhance
zoomOut

completion
zoomIn
enhance
zoomOut

enhance
completion
zoomOut
zoomIn

zoomOut
enhance
completion
zoomIn

zoomIn
enhance
completion
zoomOut

completion
enhance
zoomIn
zoomOut

enhance
completion
zoomIn
zoomOut

zoomOut
enhance
zoomIn
completion

enhance
completion

enhance
enhance
completion
completion
enhance
completion

zoomIn
enhance
completion
zoomOut

enhance
completion
enhance
completion

enhance
enhance
completion
enhance
completion
completion

completion
enhance
completion
zoomIn
zoomOut

enhance
enhance
completion
completion

enhance
completion
enhance
completion
enhance
completion

enhance
completion
zoomIn
zoomOut

enhance
completion
enhance
completion
enhance
completion

zoomOut
enhance
zoomIn
completion

enhance
completion

enhance
enhance
completion
completion
enhance
completion

zoomIn
enhance
completion
zoomOut

enhance
completion
enhance
completion

enhance
enhance
completion
enhance
completion
completion

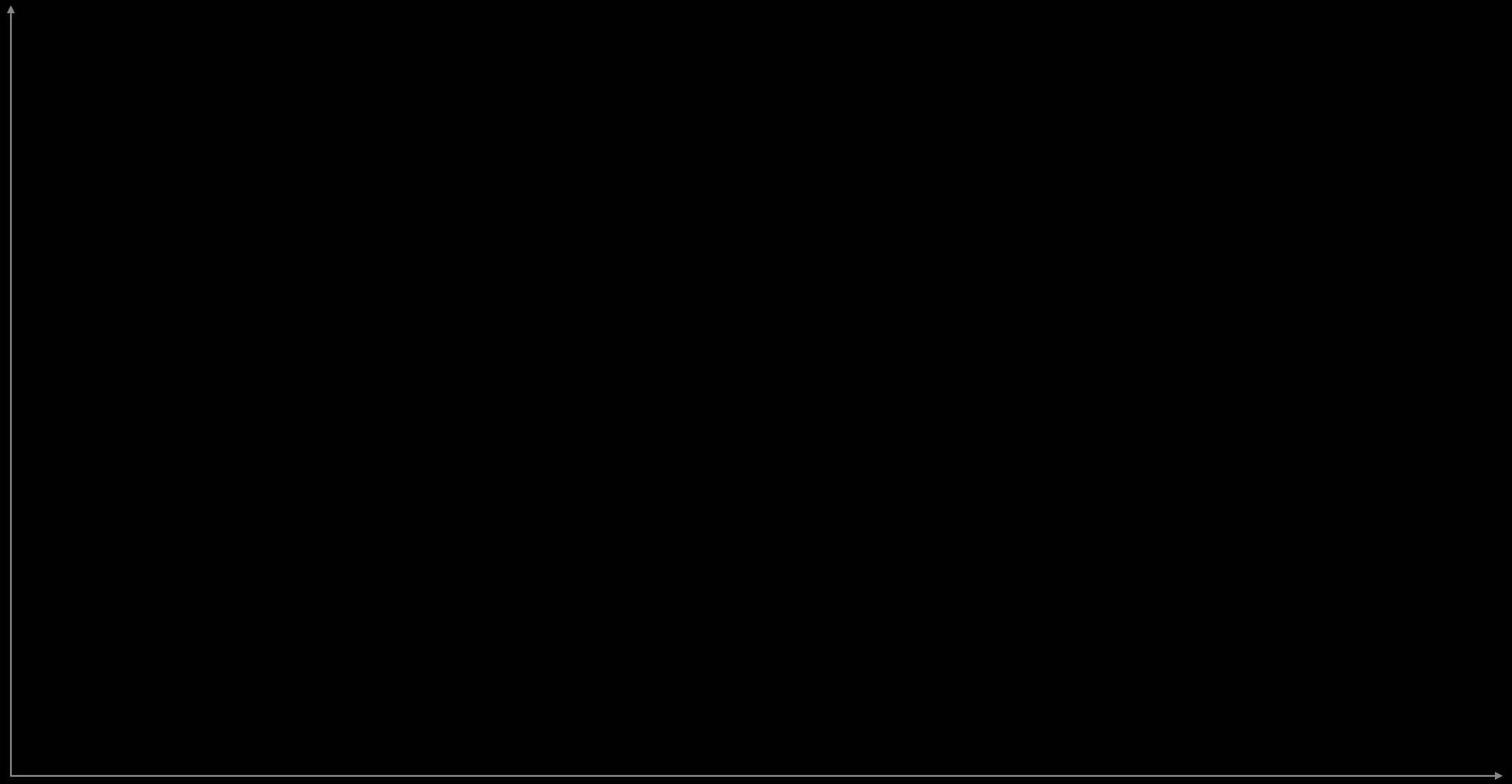
completion
enhance
completion
zoomIn
zoomOut

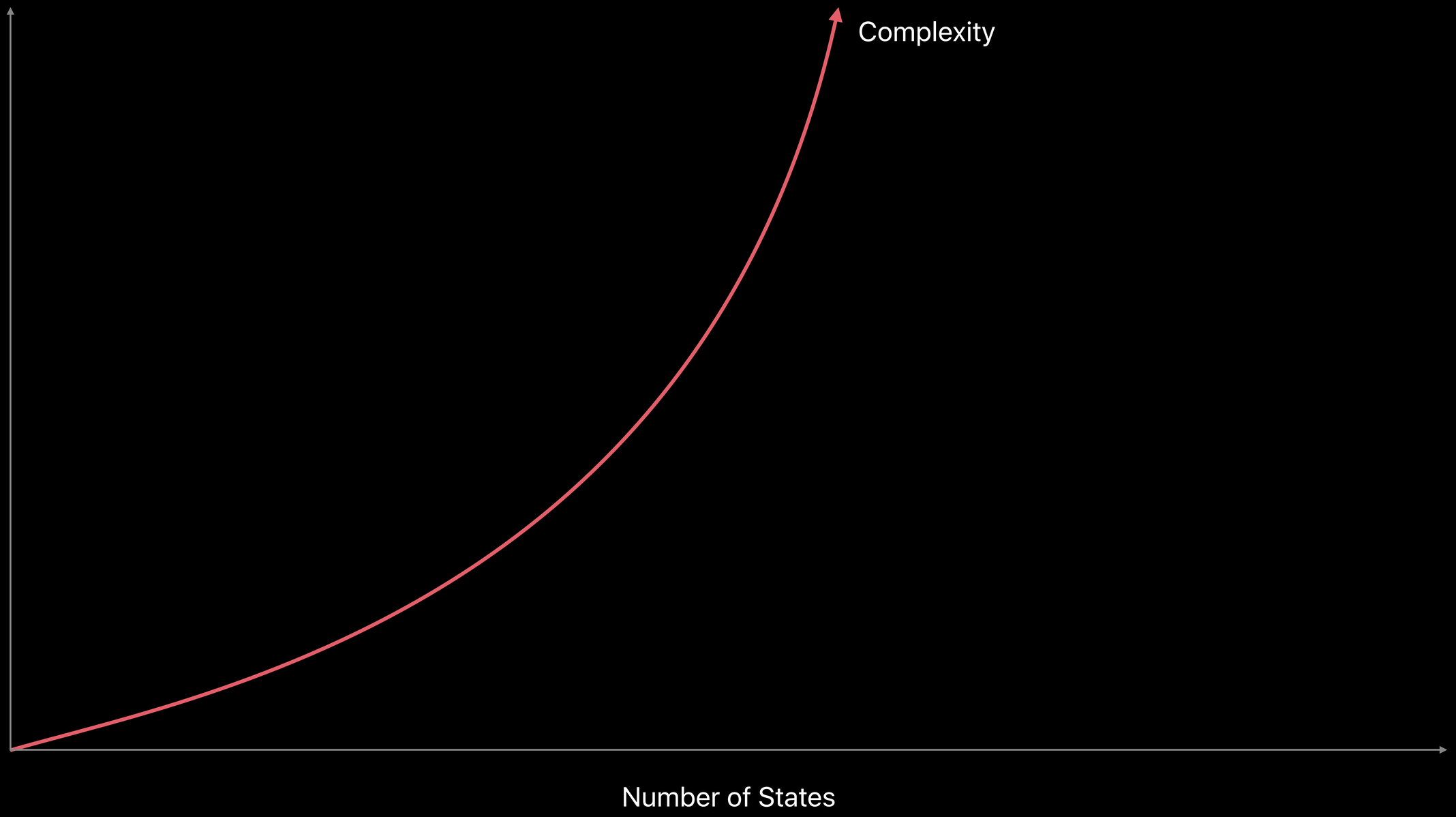
enhance
enhance
completion
completion

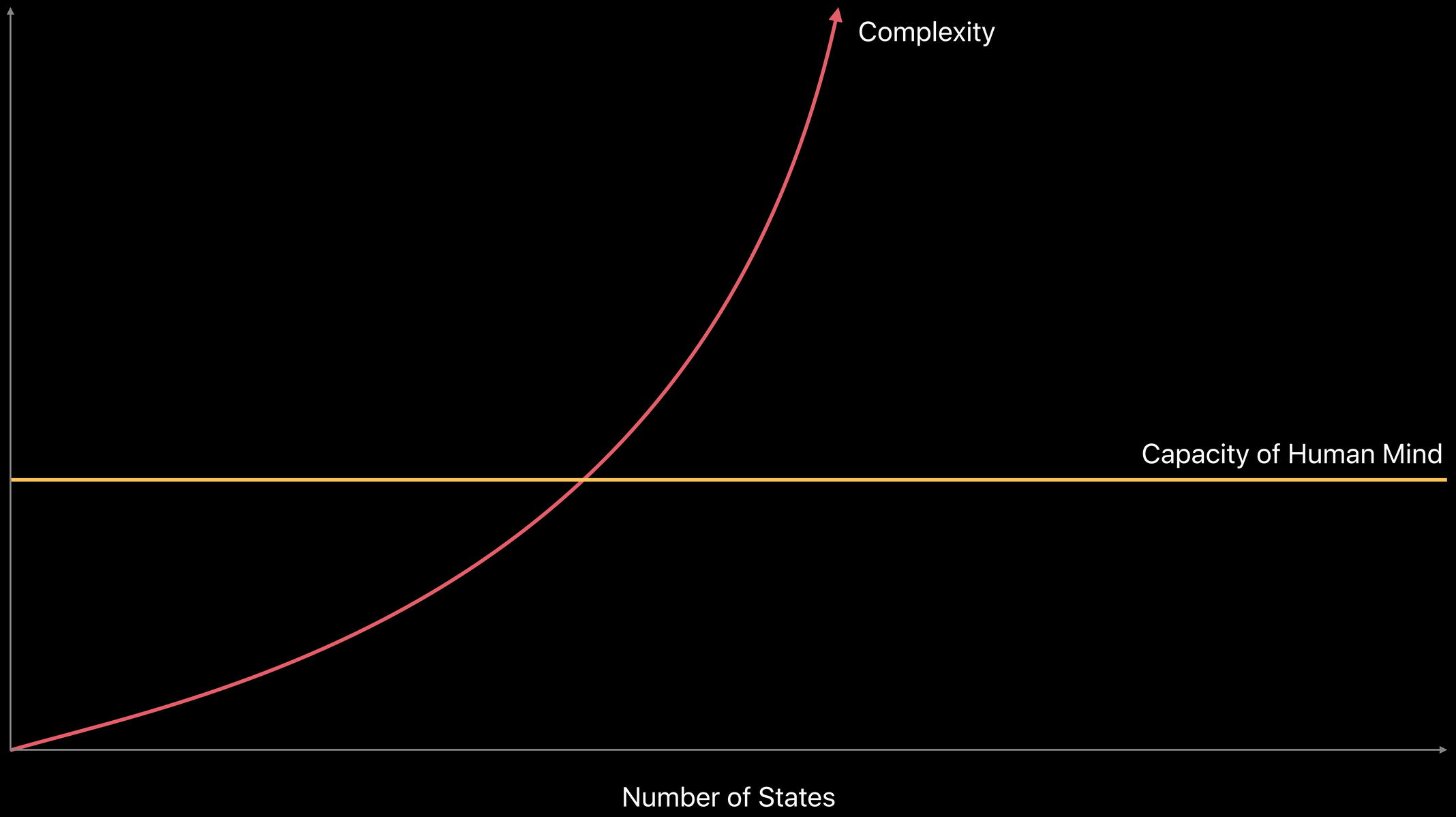
enhance
completion
enhance
completion
enhance
completion

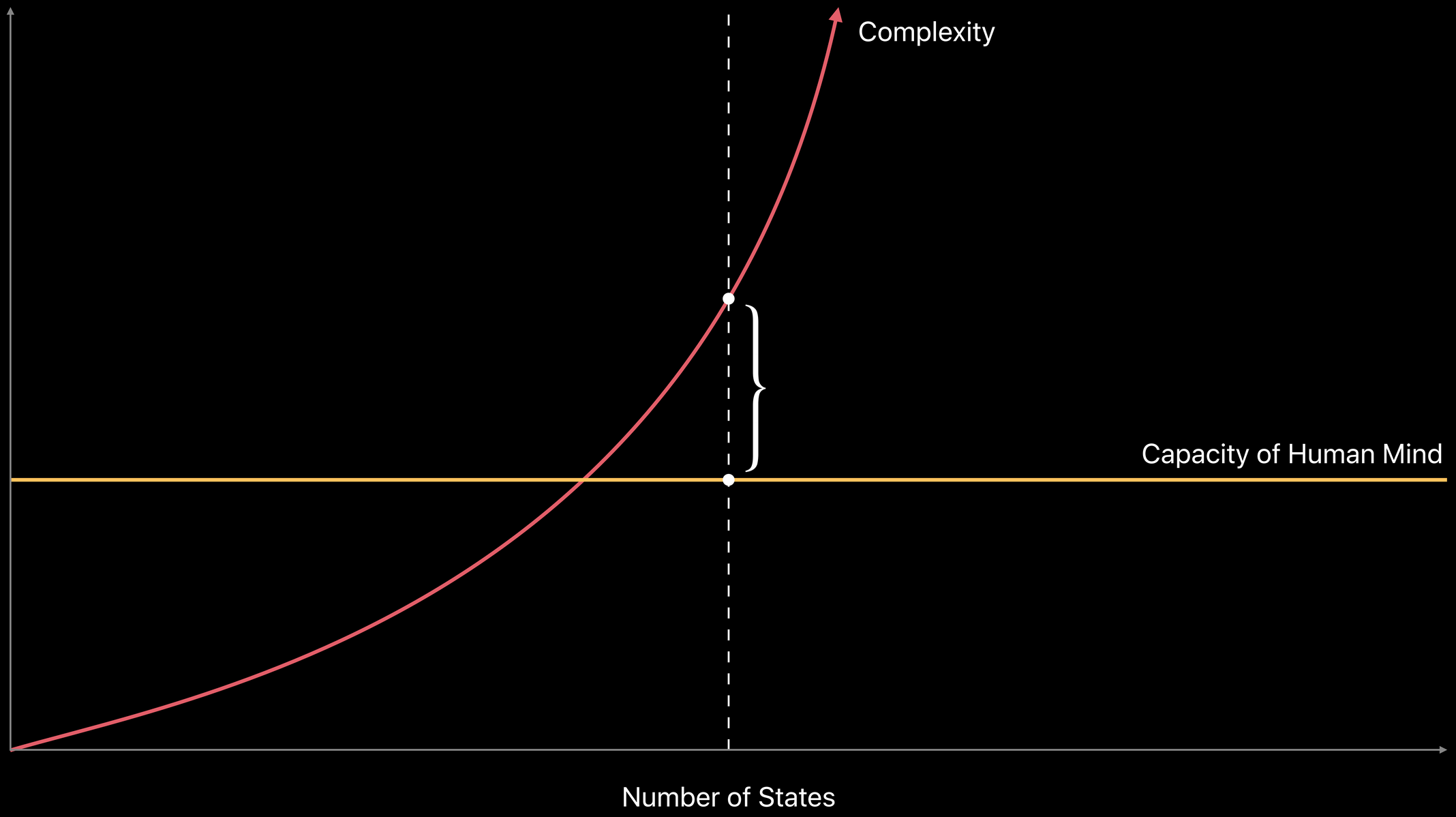
enhance
completion
zoomIn
zoomOut

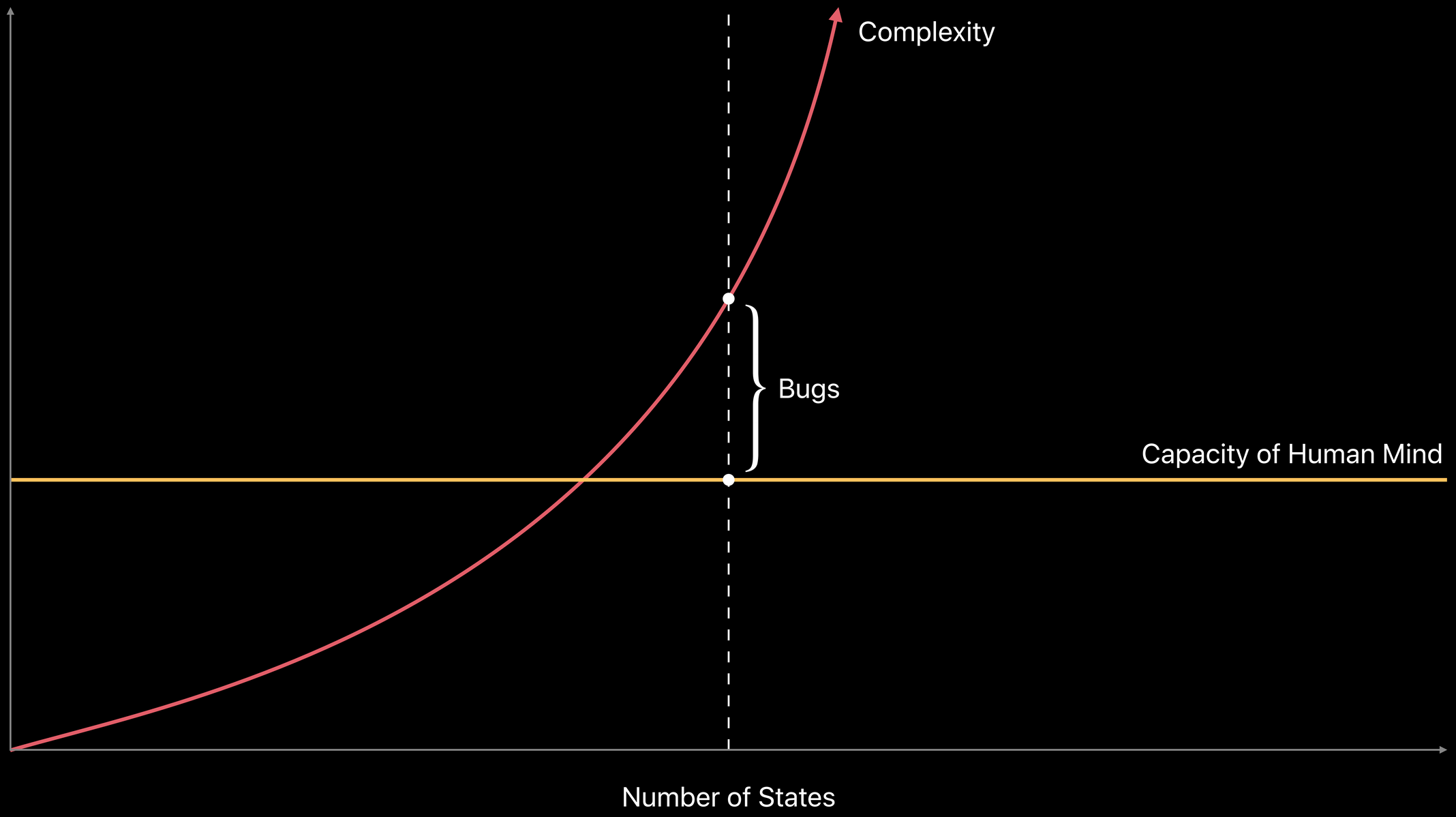
enhance
completion
enhance
completion
enhance
completion


Number of States


Complexity
Number of States


Complexity
Capacity of Human Mind
Number of States


Complexity
Capacity of Human Mind
Number of States


Complexity
Bugs
Capacity of Human Mind
Number of States

zoomOut
zoomIn
completion
enhance

zoomIn
zoomOut
completion
enhance

completion
zoomOut
zoomIn
enhance

enhance
zoomOut
zoomIn
completion

zoomOut
zoomIn
enhance
completion

zoomIn
zoomOut
enhance
completion

completion
zoomOut
enhance
zoomIn

enhance
zoomOut
completion
zoomIn

zoomOut
completion
zoomIn
enhance

zoomIn
completion
zoomOut
enhance

completion
zoomIn
zoomOut
enhance

enhance
zoomIn
zoomOut
completion

zoomOut

zoomIn

completion

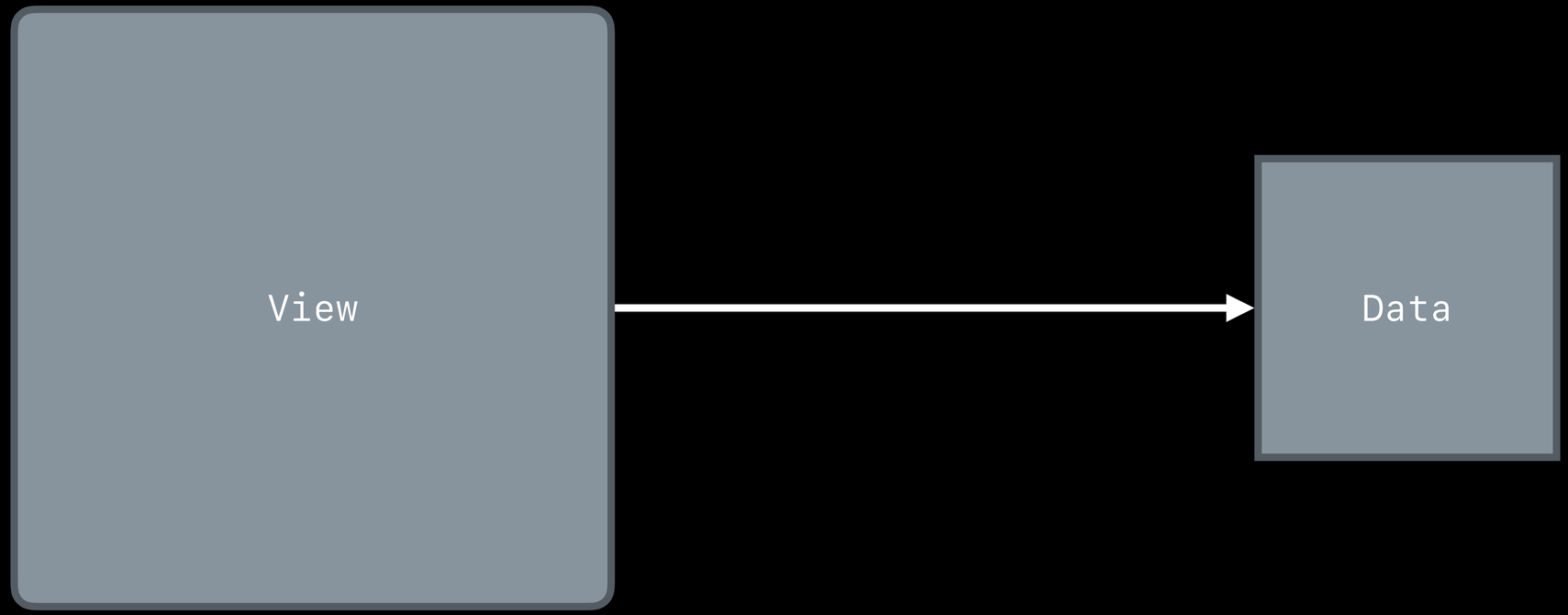
enhance

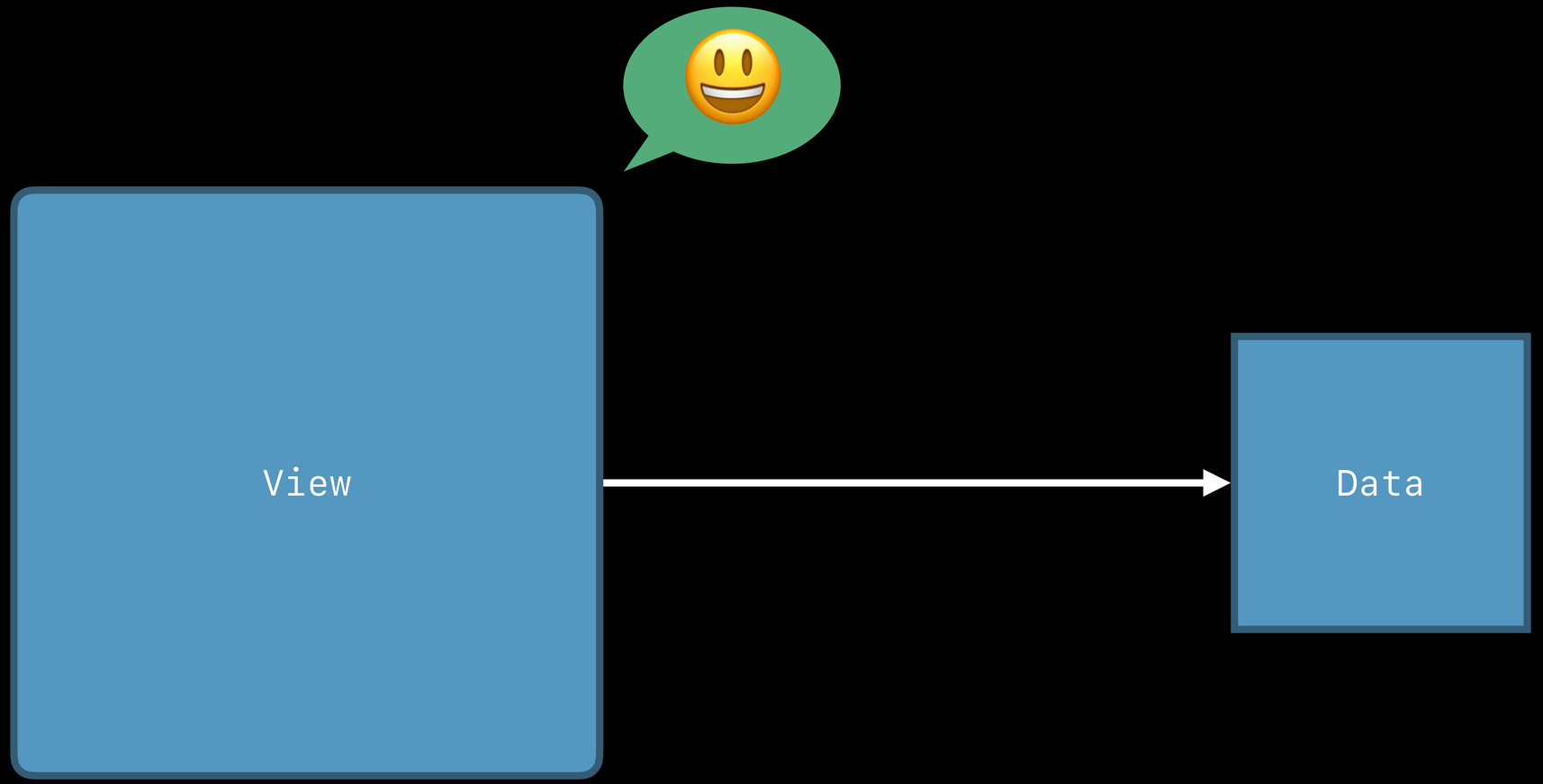
body body body body
body body body body
body body body body
body body body body

body body body body
body body body body
body body body body
body body body body

body body body body
body body body body
body body body body
body body body body

body body body body


View
Data


View
Data

Demo

Jacob Xiao, SwiftUI Engineer

The Shortest Path to a Great App

Kyle Macomber, SwiftUI Engineer

The Shortest Path to a Great App

Declarative

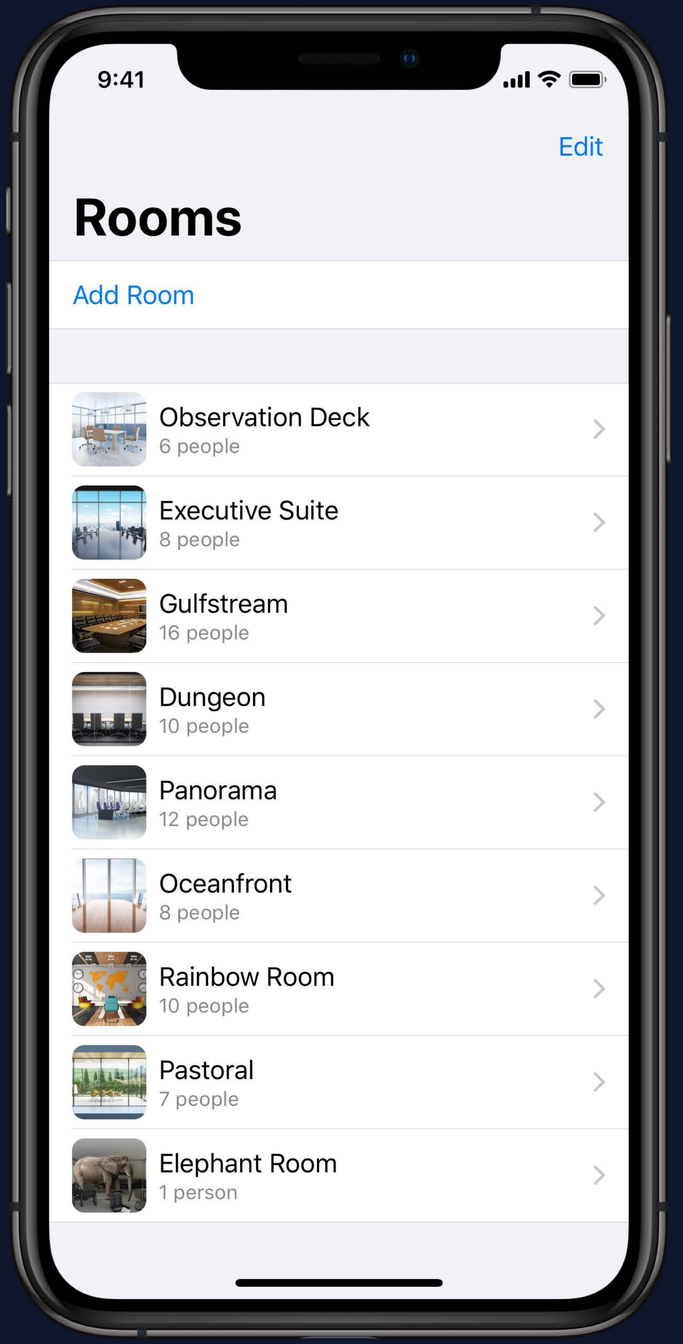
Compositional

Automatic

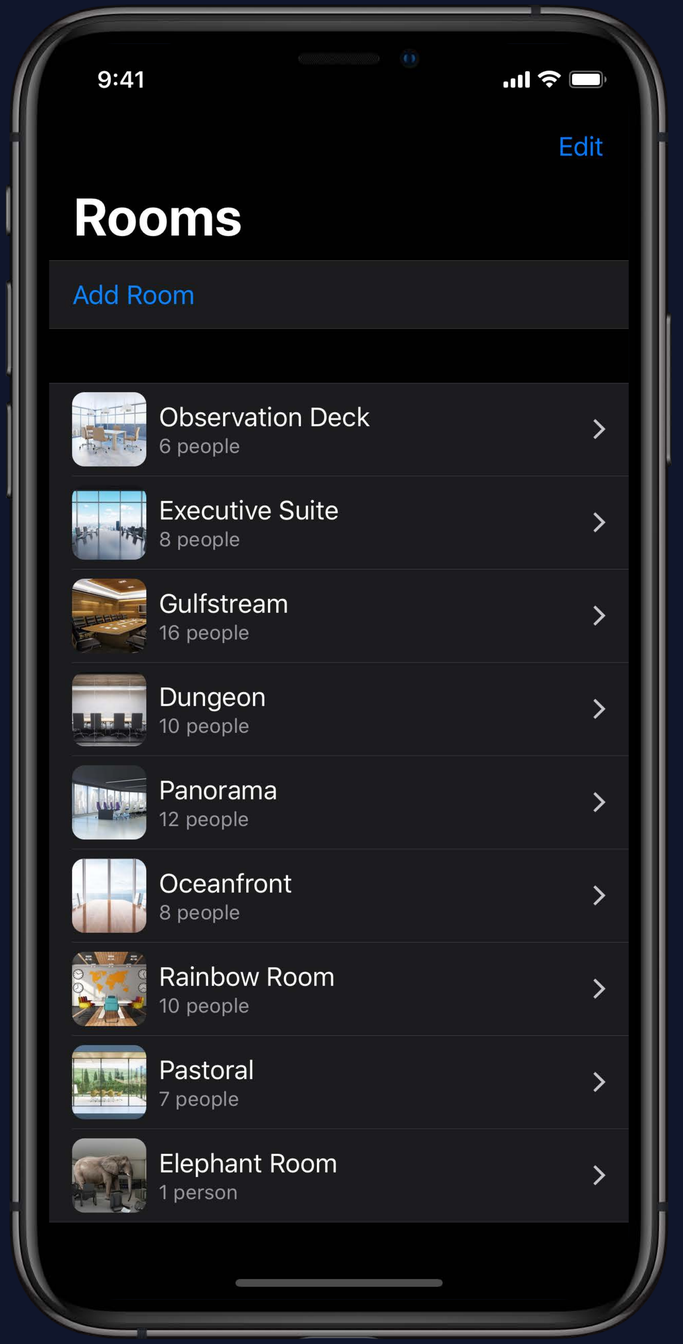
Consistent

```
// Declarative Syntax

ZStack(alignment: .topLeading) {
    Image(room.imageName)
        .resizable()
        .aspectRatio(contentMode: zoomed ? .fill : .fit)
        .tapAction {
            withAnimation { self.zoomed.toggle() }
        }

    if room.hasVideo && !zoomed {
        Image(systemName: "video.fill")
            .font(.title)
            .padding()
            .transition(.move(edge: .leading))
    }
}
```

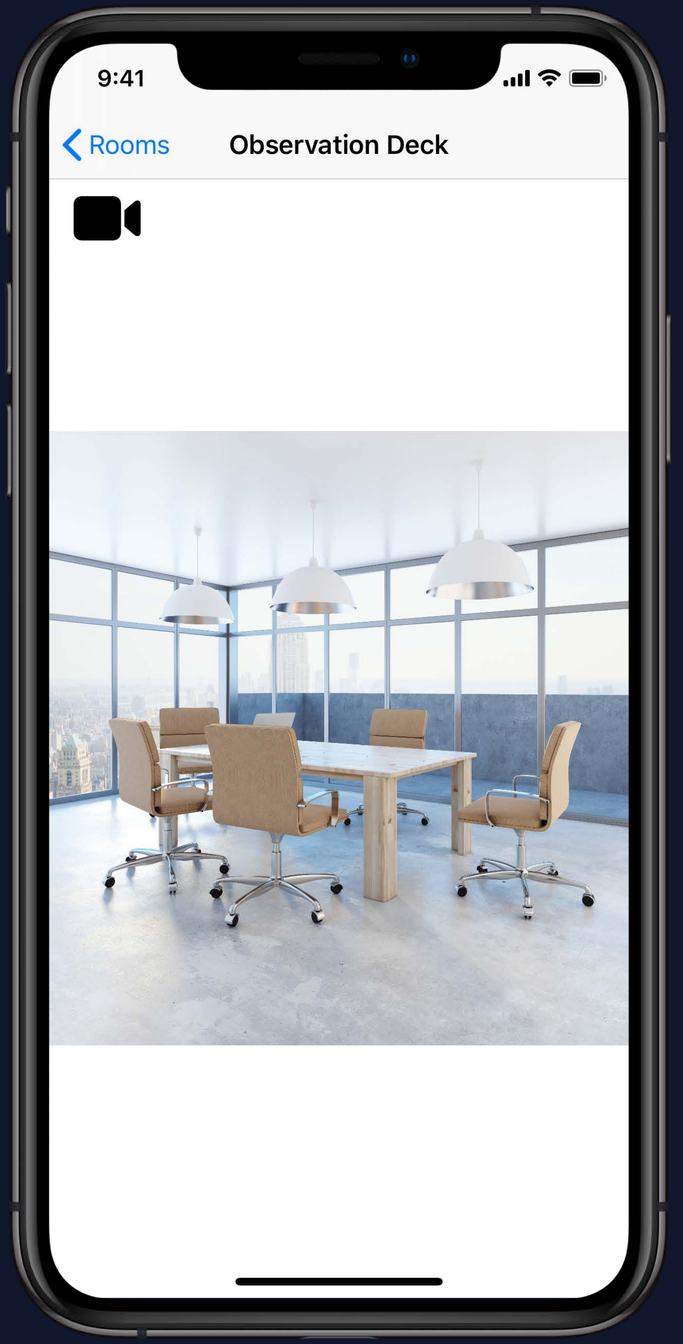
```
// Compositional Pieces

NavigationButton(destination: RoomDetail(room: room)) {
    Image(room.thumbnailName)
        .cornerRadius(8)

    VStack(alignment: .leading) {
        Text(room.name)

        Text("\(room.capacity) people")
            .font(.footnote)
            .foregroundColor(.secondary)
    }
}
```



```
// Automatic Behaviors

NavigationButton(destination: RoomDetail(room: room)) {
    Image(room.thumbnailName)
        .cornerRadius(8)

    VStack(alignment: .leading) {
        Text(room.name)

        Text("\(room.capacity) people")
            .font(.footnote)
            .foregroundColor(.secondary)
    }
}
```



```
// Consistent State

struct RoomDetail : View {
    var room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction {
                withAnimation { self.zoomed.toggle() }
            }
    }
}
```



```
// Interruptible Animations

struct RoomDetail : View {
    var room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction {
                withAnimation { self.zoomed.toggle() }
            }
    }
}
```

```
// Interruptible Animations

struct RoomDetail : View {
    var room: Room
    @State private var zoomed = false

    var body: some View {
        Image(room.imageName)
            .resizable()
            .aspectRatio(contentMode: zoomed ? .fill : .fit)
            .tapAction {
                withAnimation { self.zoomed.toggle() }
            }
    }
}
```

Basic Features

Basic
Features

Exciting, Custom
Features!

Basic Features

Exciting, Custom Features!

More Information

developer.apple.com/wwdc19/204

SwiftUI Essentials	Wednesday, 11:00
Data Flow Through SwiftUI	Thursday, 9:00
Building Custom Views with SwiftUI	Friday, 9:00


WWDC19