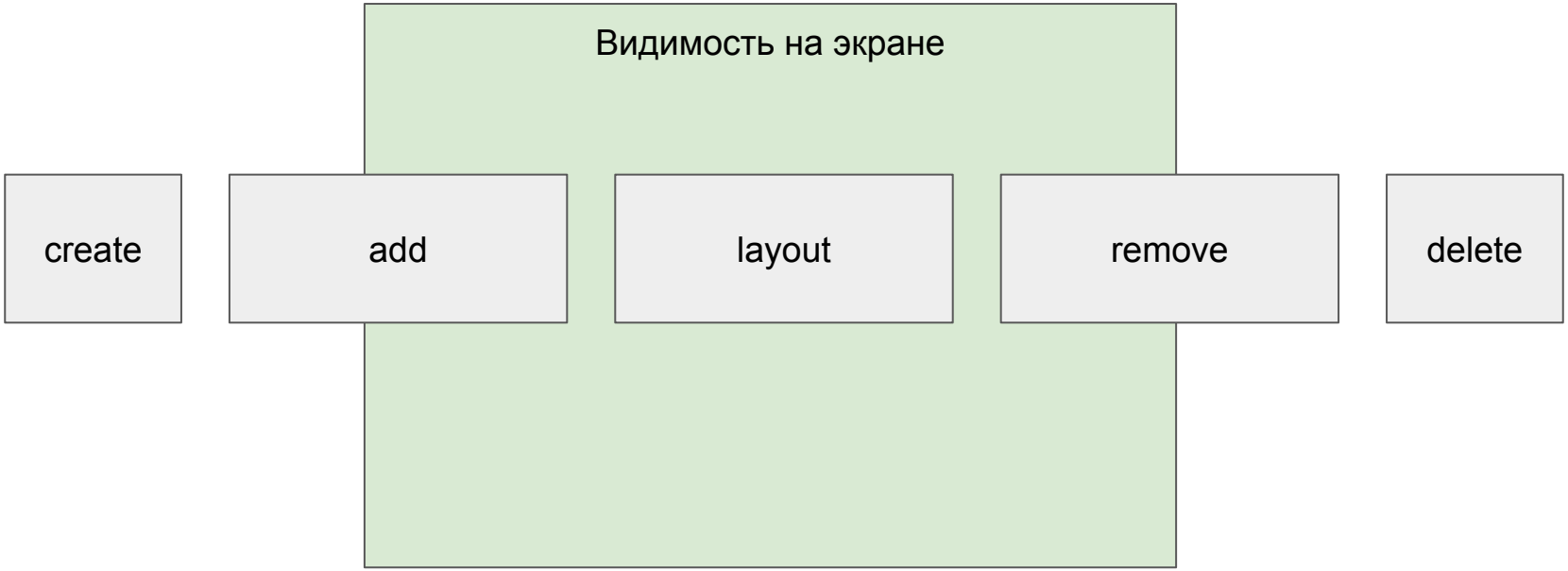
Жизненный цикл объектов SwiftUI

Dmitry Kurkin
Spark Networks

План

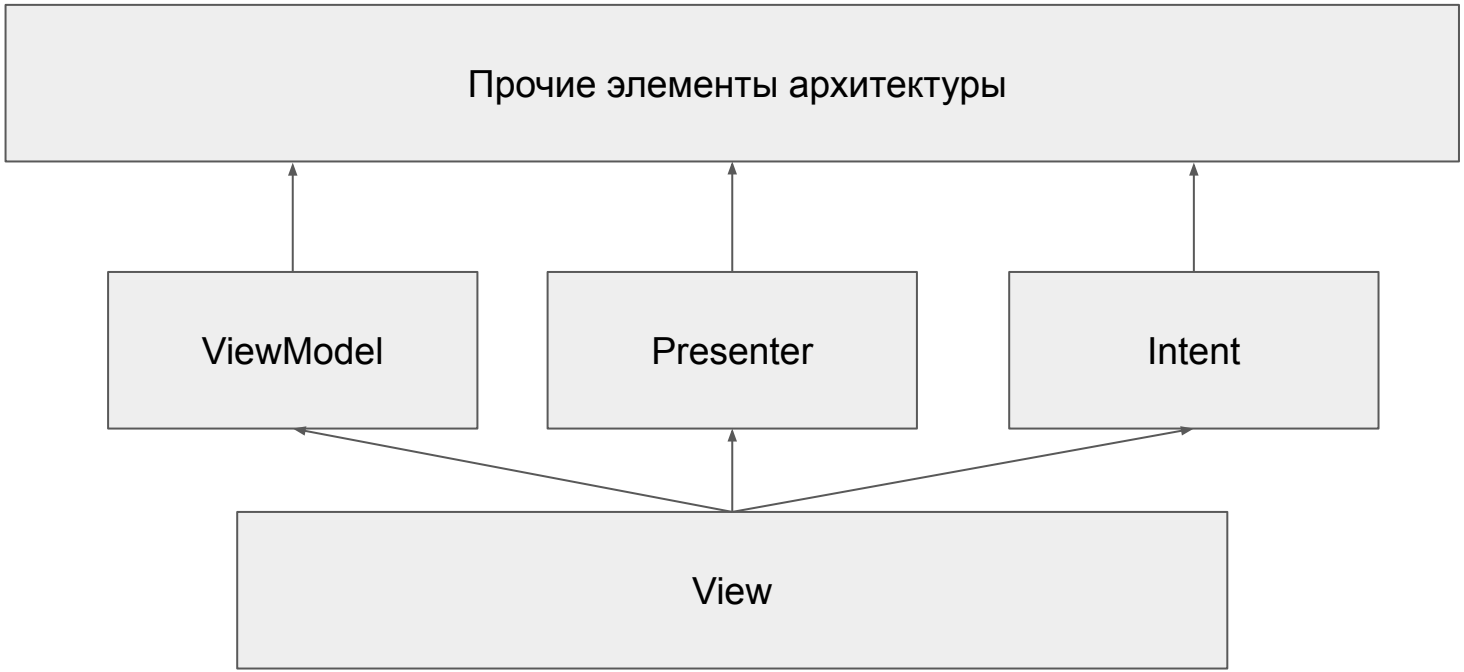
- Почему это важно
- UIView и View
- State
- Environment

Жизненный цикл UIView

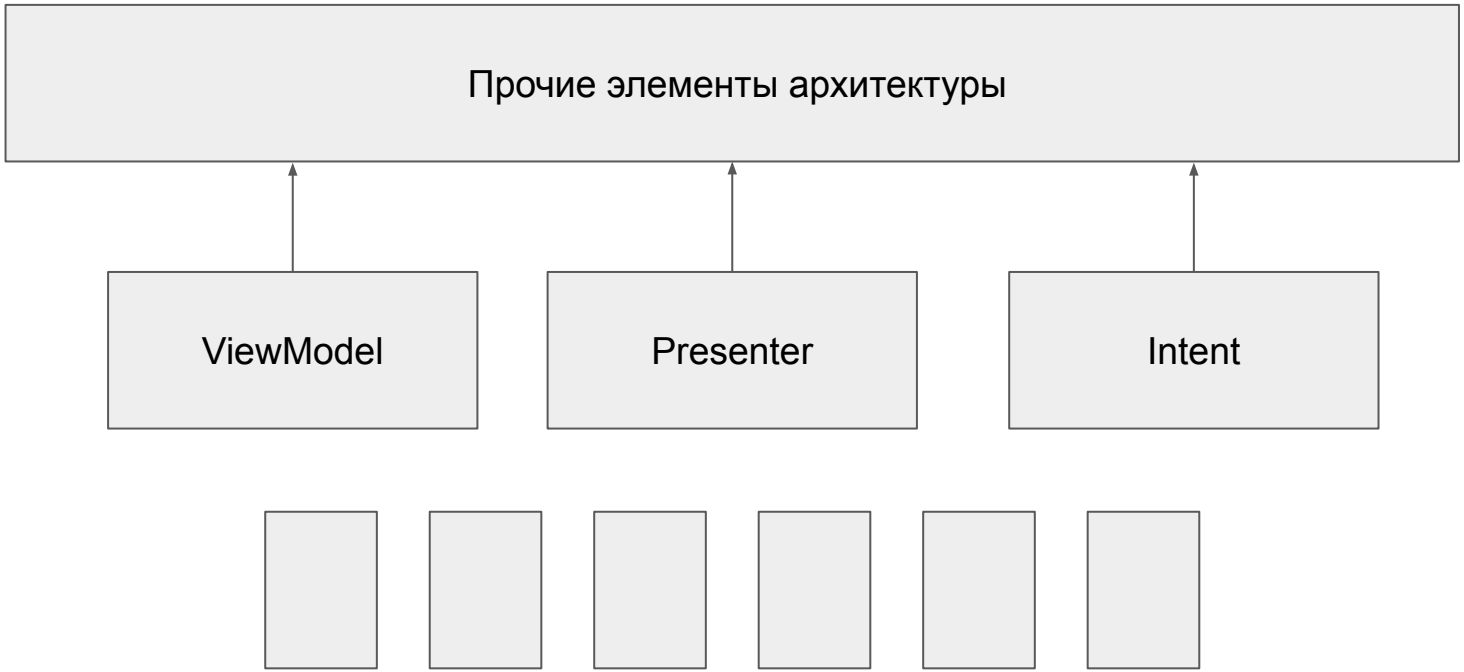


Жизненный цикл UIView

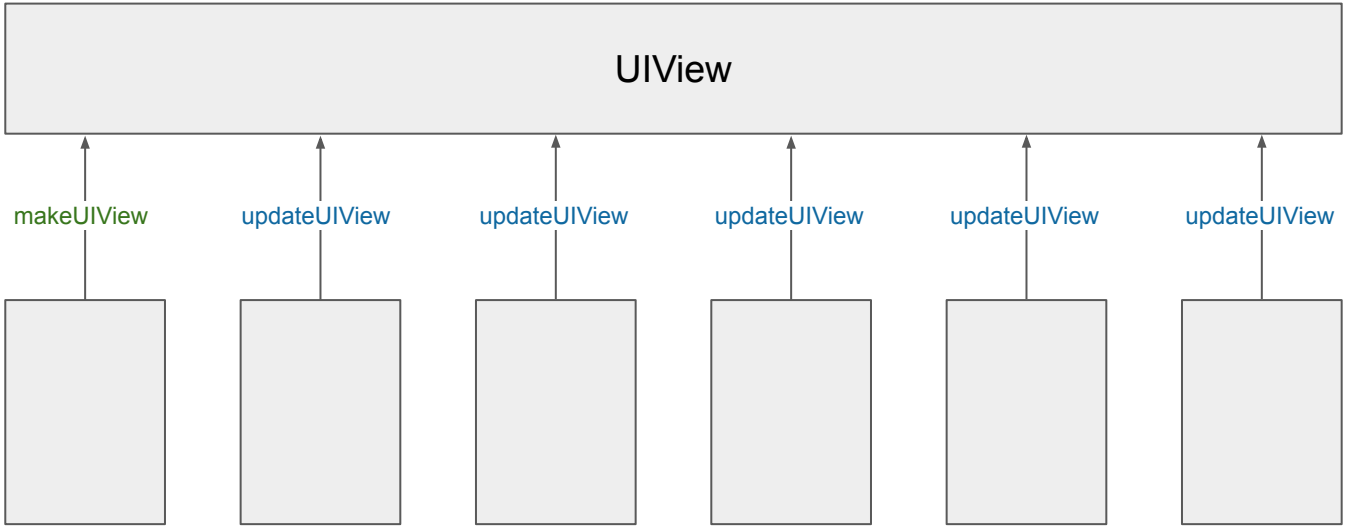
Жизненный цикл и архитектура



Жизненный цикл Swift UI View



UIViewRepresentable



Данные View

Данные View

Swift

- let
- var

State

@State
@StateObject

Environment

@Environment
@EnvironmentObject

@ObservedObject

let

```
struct CustomView: View {
    private let text: String

    init(_ text: String) {
        self.text = text
    }

    var body: some View {
        Text(text)
    }
}
```

var

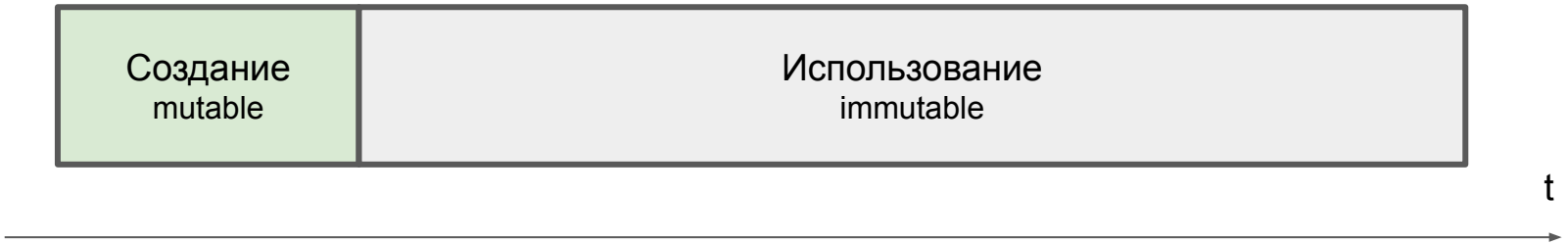
```
struct CustomView: View {
    private var text: String

    init(_ text: String) {
        self.text = text
    }

    var body: some View {
        Text(text)
    }
}
```

var

```
CustomView("Will it work?")
    .onTapGesture {
        custom.text = "Nope"
    }
```

var

- Struct
- Не “mutable”
- Нет доступа



var

```
CustomView("Details")
    .placeHolder(text: "Nice placeholder")
    .onClose { remove() }
```

var

```
CustomView("Details")
    .placeHolder(text: "Nice placeholder")
    .onClose { remove() }


mutating func placeHolder(text: String) -> Self {
    self.text = text
    return self
}
```

var

```
CustomView("Details")
    .placeHolder(text: "Nice placeholder")
```

Cannot use mutating member on immutable value: function call returns immutable value

```
    .onClose { remove() }


mutating func placeHolder(text: String) -> Self {
    self.text = text
    return self
}
```

var

```
var body: some View {
    var view = CustomView("Detail")
    return  view.placeHolder(text: "Nice placeholder")
}
```

var

```
func placeHolder(text: String) -> Self {
    var mutableSelf = self
    mutableSelf.text = text
    return mutableSelf
}
```

var в структуре View - это очень сомнительная затея

State

State

```
struct CustomView: View {
    @State private var text: String

    init(_ text: String = "Touch Me") {
        self.text = text
    }

    var body: some View {
        Text(text)
            .onTapGesture {
                text = "Satisfaction"
            }
    }
}
```

State. Проблема? Все работает!

```
struct CustomView: View {
    @State private var text: String

    init(_ text: String = "Touch Me") {
        self.text = text
    }

    var body: some View {
        Text(text)
            .onTapGesture {
                text = "Satisfaction"
            }
    }
}
```

Загадка State

```
var body: some View {
    CustomView(viewModel.inputText)
}

inputText = "Push me"
  ...
inputText = "Touch me"
  ...
inputText = "Satisfaction"
```

Загадка State

```
struct CustomView: View {
    @State private var text: String = "Nope"

    init(_ text: String?) {
        text.map { self.text = $0 }
        print("\(self.text) - \(text)")
    }

    var body: some View {
        let _ = print("-- \(text)")
        Text(text)
            .onTapGesture {
                text = "Hi-Hi"
            }
    }
}
```

Загадка State

```
struct CustomView: View {
    @State private var text: String = "Nope"

    init(_ text: String?) {
        text.map { self.text = $0 }
        print("\(self.text) - \(text)")
    }

    var body: some View {
        let _ = print("-- \(text)")
        Text(text)
            .onTapGesture {
                text = "Hi-Hi"
            }
    }
}
```

Nope - Optional("Nope")

-- Nope

Nope - Optional("Touch me")

Nope - Optional("Satisfaction")

Nope - Optional("Push me")

Nope - Optional("Touch me")

Nope - Optional("Satisfaction")

Загадка State

```
@State private var viewModel: ViewModel

init(_ text: String) {
    viewModel = ViewModel(text)
}

var body: some View {
    let _ = print("-- \(viewModel.objectId)")
    Text(viewModel.text)
}
```

-> ViewModel init ObjectIdentifier(**0x0000600001ee8280**)

-- ObjectIdentifier(**0x0000600001ee8280**)

-> ViewModel init ObjectIdentifier(0x0000600001ee8e80)

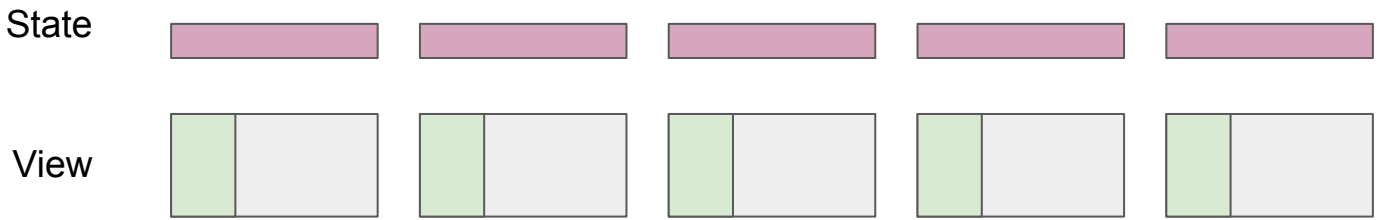
-- ObjectIdentifier(**0x0000600001ee8280**)

-> ViewModel init ObjectIdentifier(0x0000600001e142c0)

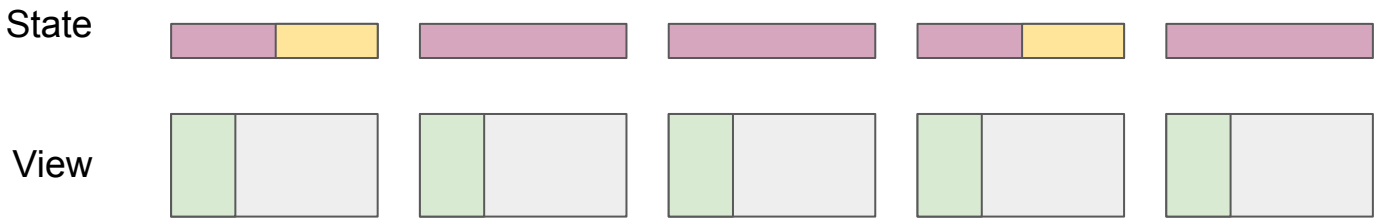
-> ViewModel deinit ObjectIdentifier(0x0000600001ee8e80)

-- ObjectIdentifier(**0x0000600001ee8280**)

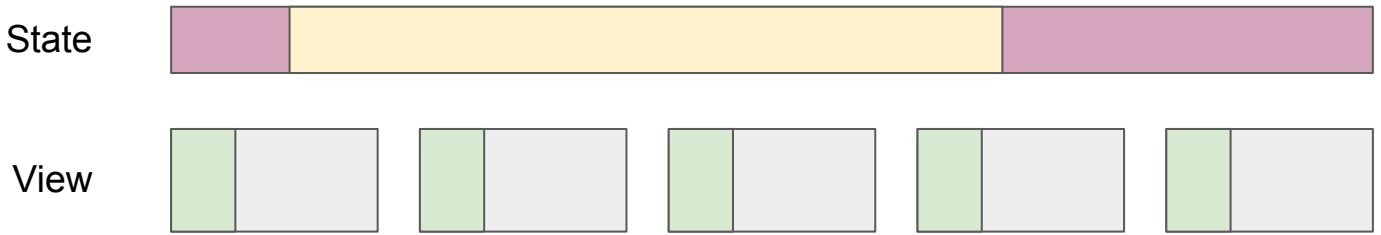
State. Время жизни



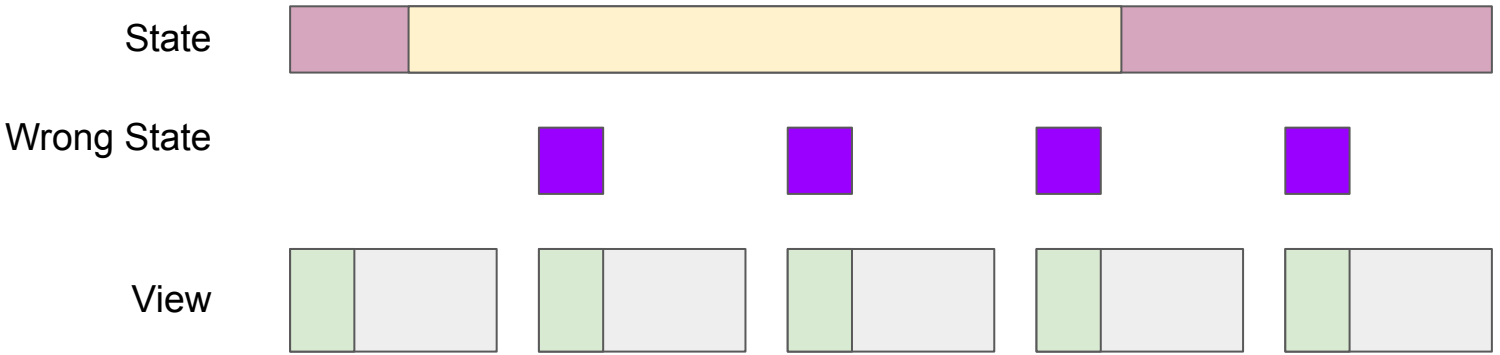
State. Время жизни



State. Время жизни



State. Время жизни



State. Проблема? Все работает!

```
struct CustomView: View {
    @State private var text: String

    init(_ text: String = "Touch Me") {
        self.text = text
    }

    var body: some View {
        Text(text)
            .onTapGesture {
                text = "Satisfaction"
            }
    }
}
```

State снаружи не доступен

```
struct CustomView: View {
    @State private var text: String = "Touch Me"

    init() {
    }

    var body: some View {
        Text(text)
            .onTapGesture {
                text = "Satisfaction"
            }
    }
}
```

StateObject

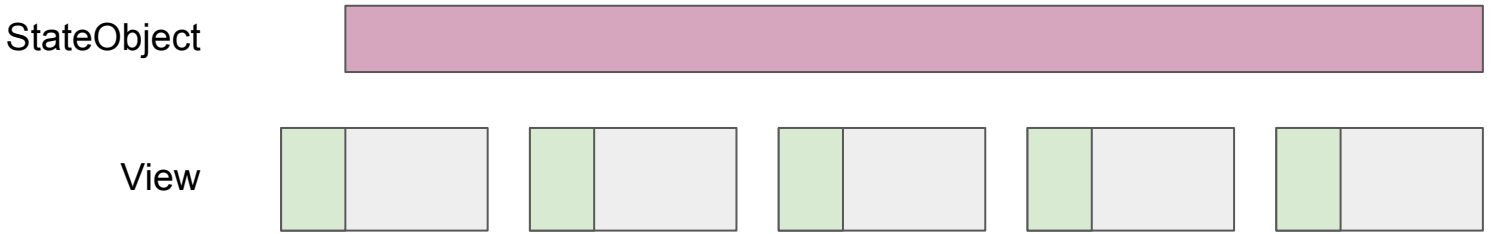
```
@StateObject private var viewModel: ViewModel

init(_ text: String) {
    viewModel = ViewModel(text)
Cannot assign to property: 'viewModel' is a get-only property
}
```

StateObject

```
@StateObject private var viewModel = ViewModel("Nope")

init(_ text: String) {
    viewModel.text = text
Accessing StateObject's object without being installed on a View. This will
create a new instance each time.
}
```

2022-10-30 21:02:20.481521+0100 Counter[95839:12059255] [SwiftUI] Accessing StateObject's object without being installed on a View. This will create a new instance each time.
-> ViewModel init ObjectIdentifier(0x0000600000c90c20)
-> ViewModel deinit ObjectIdentifier(0x0000600000c90c20)
-> ViewModel init ObjectIdentifier(**0x0000600000c82d20**)
-- ObjectIdentifier(**0x0000600000c82d20**)

StateObject. Время жизни

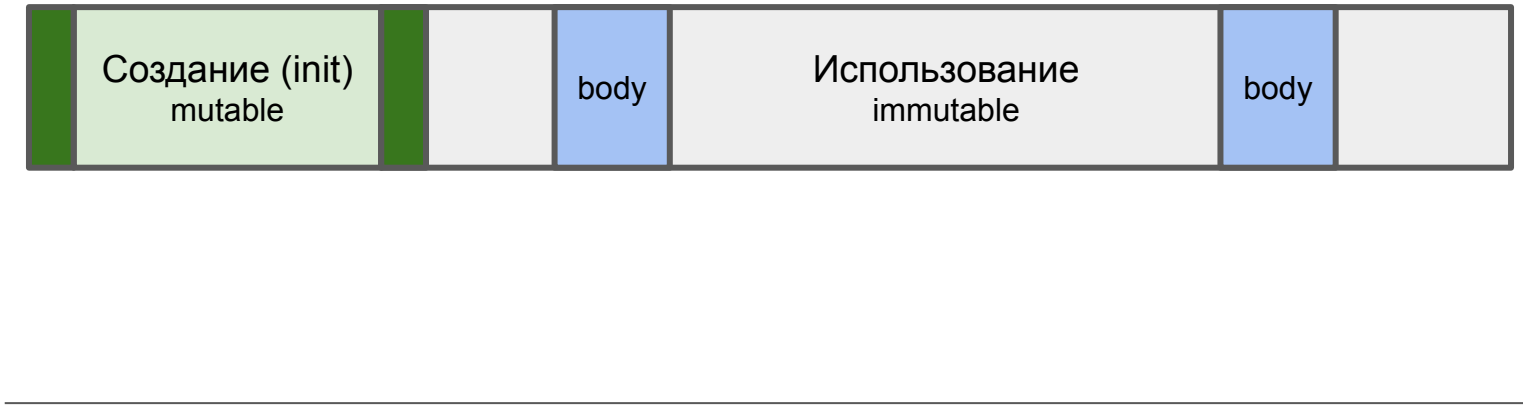


Документация

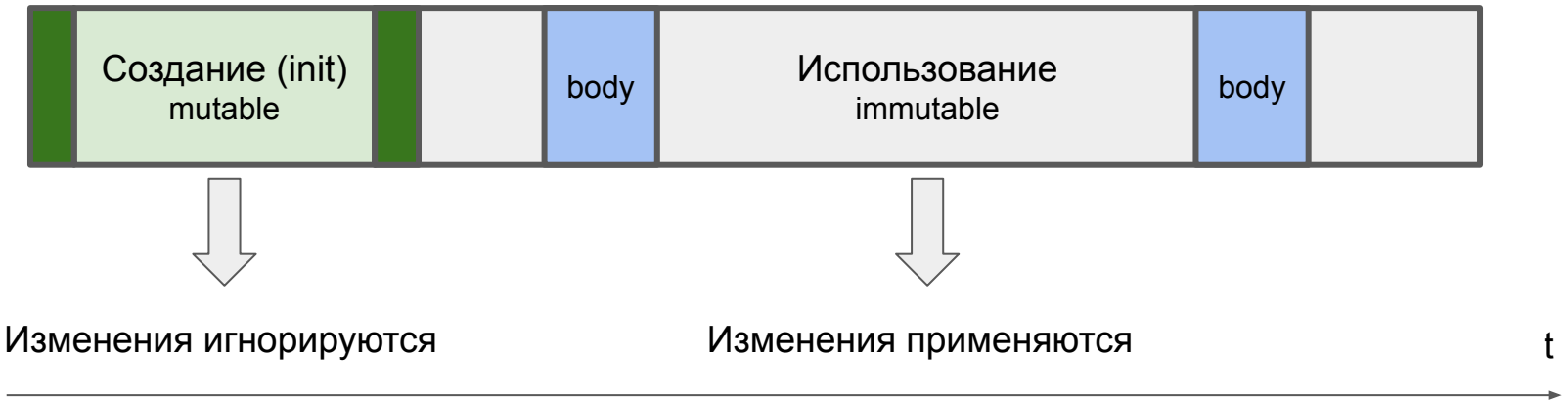
State:
Don’t initialize a state property of a view at the point in the view hierarchy where you instantiate the view, because this can conflict with the storage management that SwiftUI provides.

StateObject:
SwiftUI creates a new instance of the object only once for each instance of the structure that declares the object.

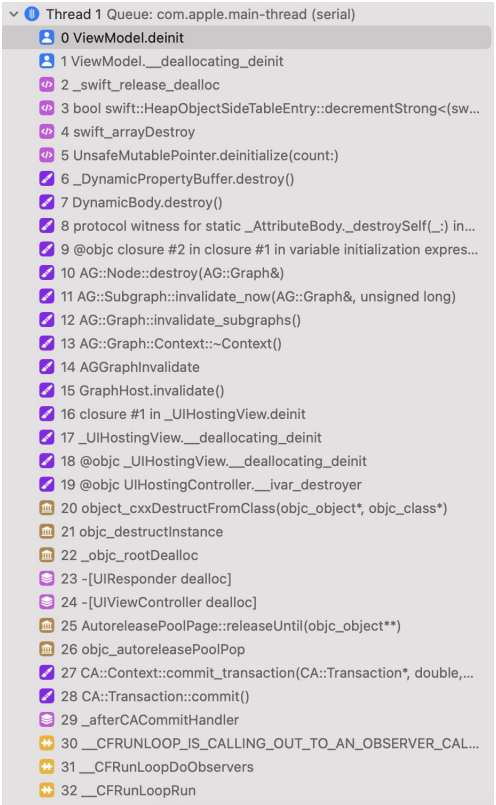
Изменение


Создание (init)
mutable
body
Использование
immutable
body
t

Изменение


Создание (init)
mutable
body
Использование
immutable
body
Изменения игнорируются
Изменения применяются
t

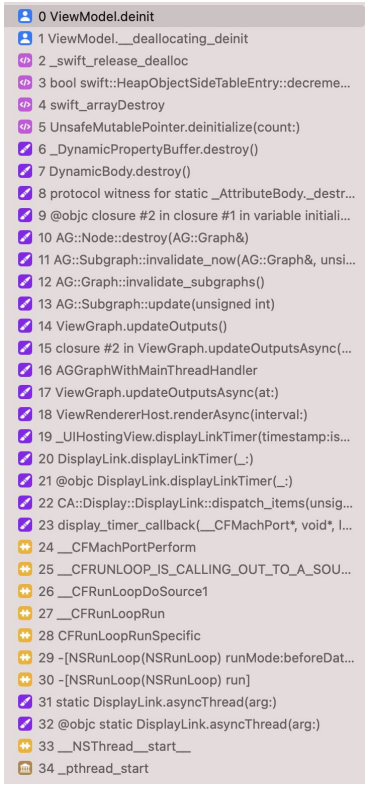
Удаление State



Удаление State

```
#0    ViewModel.deinit
…
#7    DynamicBody.destroy()
…
#10   AG::Node::destroy(AG::Graph&)
…
#13   AG::Graph::Context::~Context() ()
…
#17   _UIHostingView.__deallocating_deinit ()
…
#24   -[UIViewController dealloc] ()
```

Удаление State. Смена идентификатора

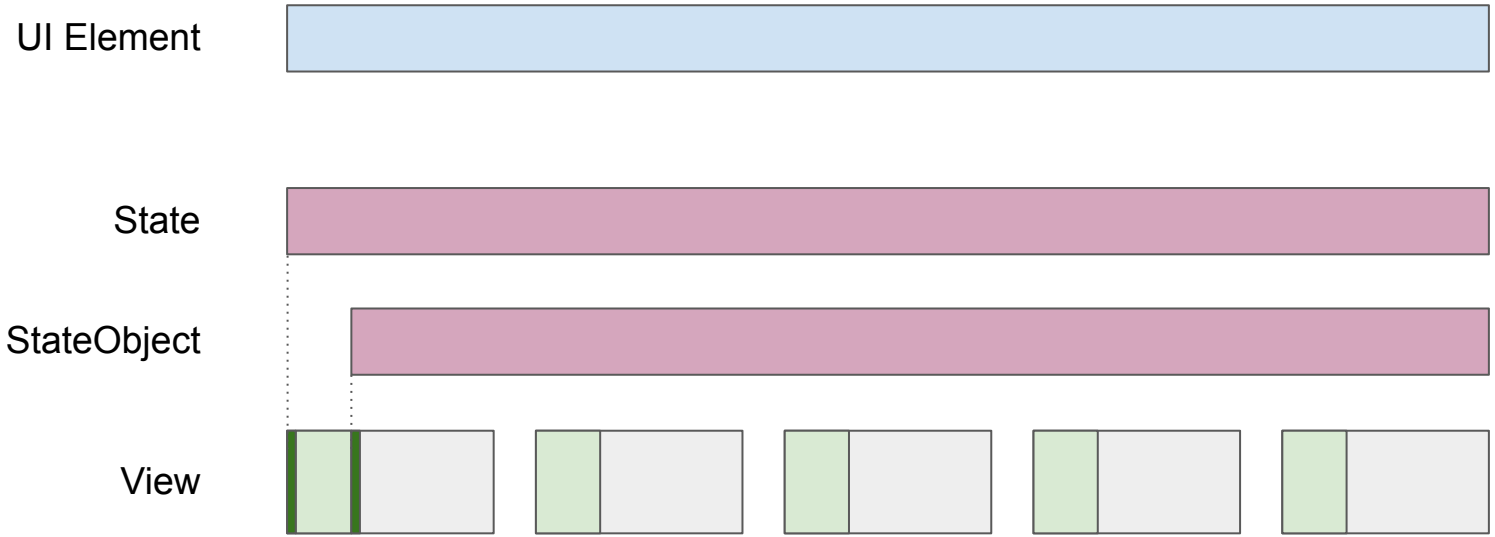


Удаление State. Смена идентификатора

Thread 10

#0 ViewModel.deinit
…
#7 DynamicBody.destroy()
…
#10 AG::Node::destroy(AG::Graph&)
…
#31 static DisplayLink.asyncThread(arg:)

Время жизни View & Co



Environment

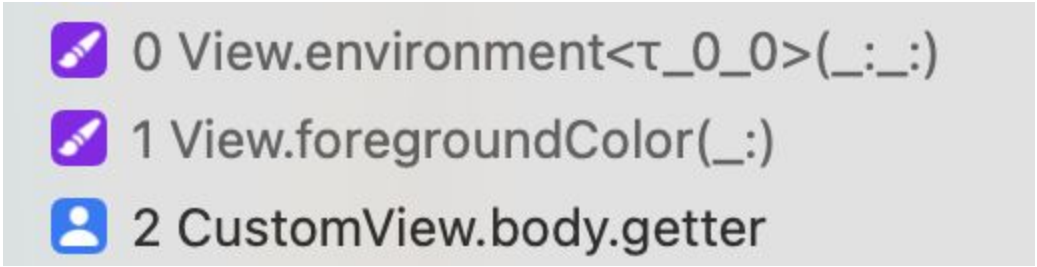
Изменение View

```
Text(text)
    .font(.system(.title))
    .foregroundColor(.brown)
    .onAppear {
        print("Do something")
    }
    .onTapGesture {
        text = "Hi-Hi"
    }
```

Environment value

```
@Environment(\.font) var font: Font?
@Environment(\.lineLimit) var lineLimit: Int?
```

Environment value



Environment value

```
private struct MyEnvironmentKey: EnvironmentKey {
    static let defaultValue: String = "Default value"
}

extension EnvironmentValues {
    var myCustomValue: String {
        get { self[MyEnvironmentKey.self] }
        set { self[MyEnvironmentKey.self] = newValue }
    }
}

extension View {
    func myCustomValue(_ myCustomValue: String) -> some View {
        environment(\.myCustomValue, myCustomValue)
    }
}
```

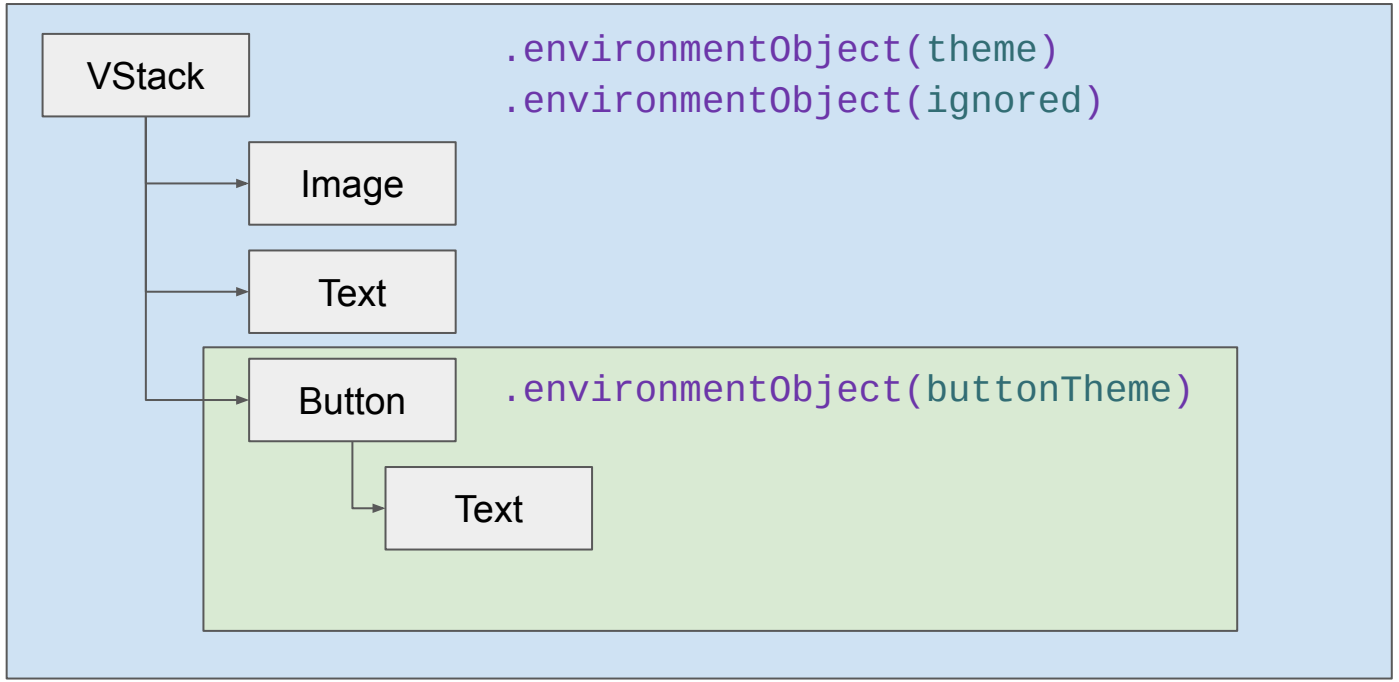
Environment value

```
private struct MyEnvironmentKey: EnvironmentKey {
    static let defaultValue: MyObject? = nil
}

extension EnvironmentValues {
    var myCustomValue: MyObject? {
        get { self[MyEnvironmentKey.self] }
        set { self[MyEnvironmentKey.self] = newValue }
    }
}

extension View {
    func myCustomValue(_ myCustomValue: MyObject) -> some View {
        environment(\.myCustomValue, myCustomValue)
    }
}
```

Environment Object

```
CustomView(viewModel.inputText)
    .environmentObject(theme)

…
@EnvironmentObject var theme: ColorTheme
```

Environment. Область видимости



Разрыв иерархии

```
VStack {
    CustomView(viewModel.inputText)
        .id(viewModel.itemId)
    Button("Next") {
        viewModel.next()
    }
    .myWidget(isPresented: viewModel.widget) {
        Button("Next") {
            viewModel.next()
        }
    }
}
```

Разрыв иерархии

```
VStack {
    CustomView(viewModel.inputText)
        .id(viewModel.itemId)
    Button("Next") {
        viewModel.next()
    }
    .myWidget(isPresented: viewModel.widget) {
        Button("Next") {
            viewModel.next()
        }
    }
}
```

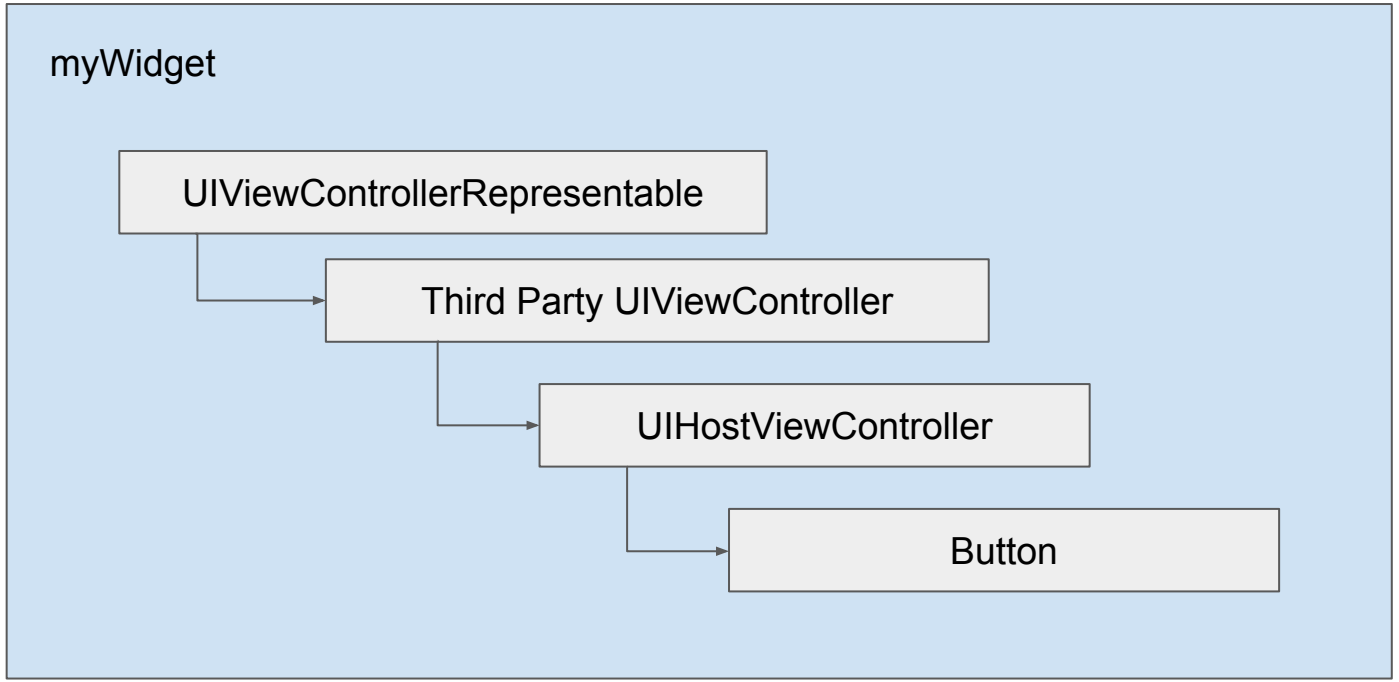
Разрыв иерархии

```
VStack {
    CustomView(viewModel.inputText)
        .id(viewModel.itemId)
    Button("Next") {
        viewModel.next()
    }
    .myWidget(isPresented: viewModel.widget) {
        Button("Next") {
            viewModel.next()
        }
    }
}
```

Разрыв иерархии

```
VStack {
    CustomView(viewModel.inputText)
        .id(viewModel.itemId)
    Button("Next") {
        viewModel.next()
    }
    .myWidget(isPresented: viewModel.widget) {
        Button("Next") {
            viewModel.next()
        }
    }
}
```

Environment. Разрыв



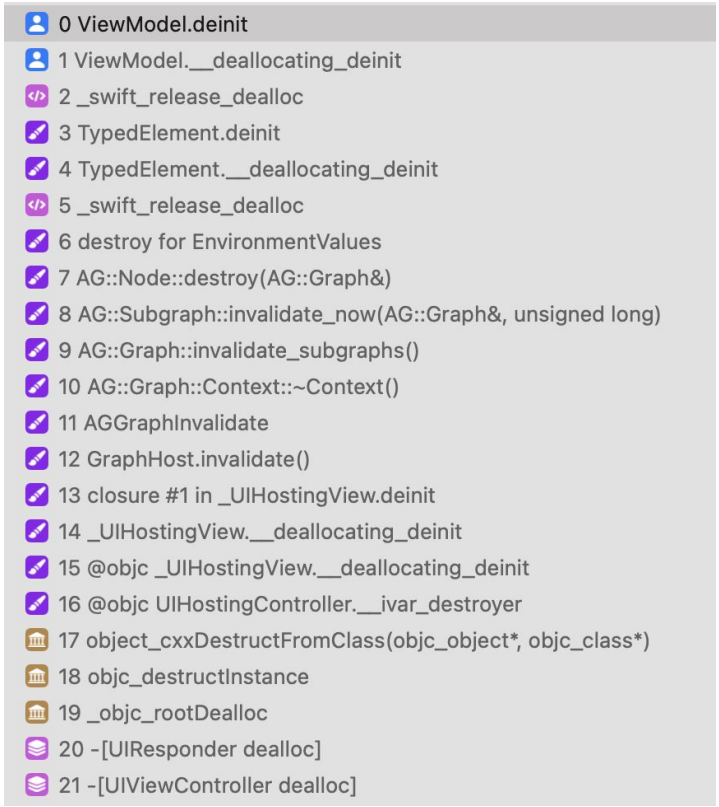
Разрыв иерархии на изоленте

```
VStack {
    CustomView(viewModel.inputText)
        .id(viewModel.itemId)
    Button("Next") {
        viewModel.next()
    }
    .myWidget(isPresented: viewModel.widget) {
        Button("Next") {
            viewModel.next()
        }
        .environmentObject(knownObject)
        .environment(KnownEnvironmentKey, knownValue)
    }
}
```

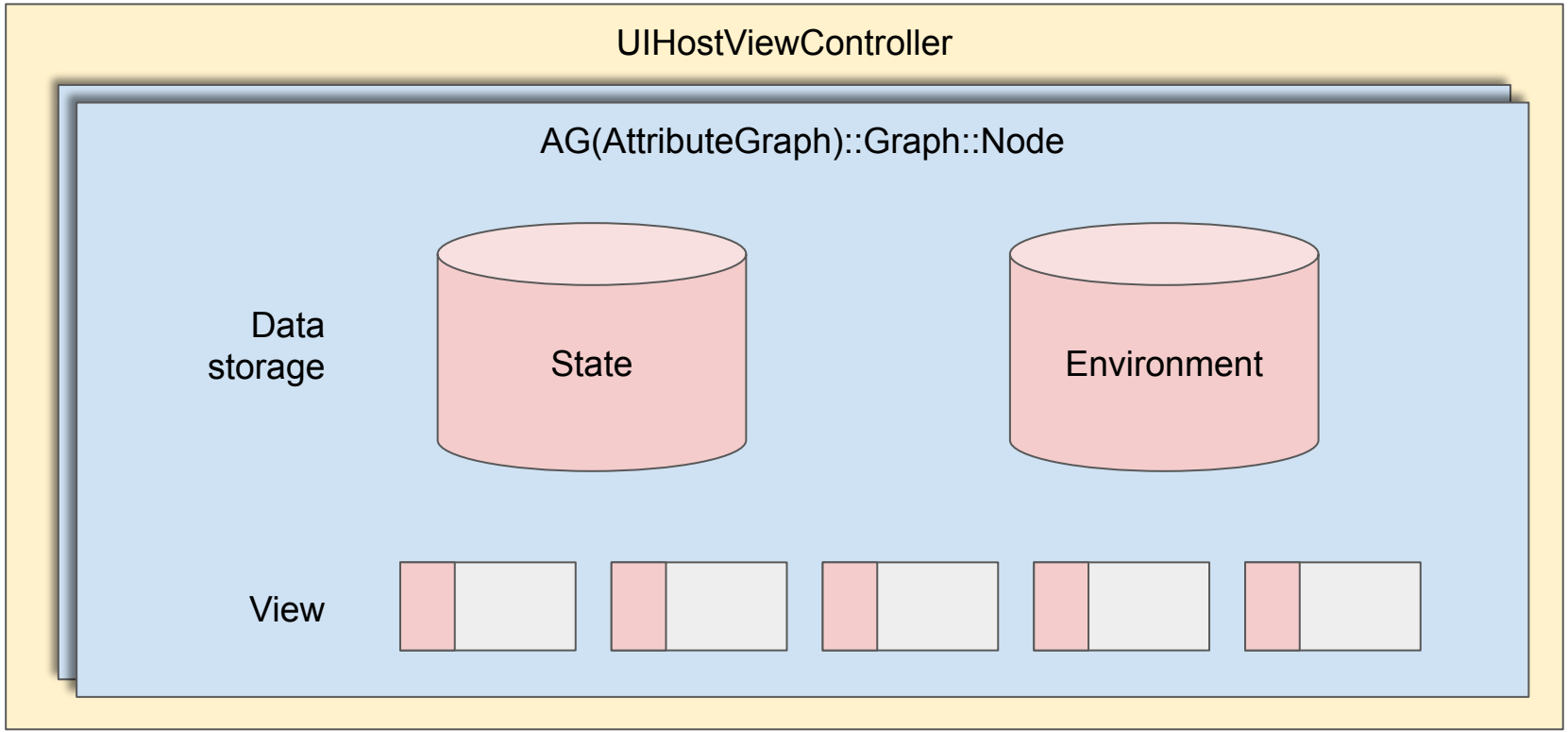
Разрыв иерархии на изоленте

```
func makeUIViewController(context: Context) -> PresentingViewController<Content> {
    PresentingViewController(isPresented: $present, content: content)
        .with(environment: [
            KnownEnvironmentKey.self = context.environment[KnownEnvironmentValue.self]
            KnownEnvironmentObject.self = context.environment[KnownEnvironmentObject.self]
        ])
}

let controller = UIHostingController(
    rootView: content()
        .environment(KnownEnvironmentKey.self, storage[KnownEnvironmentKey.self])
        .environmentObject(storage[KnownEnvironmentObject.self])
)
```

Удаление Environment



Итоги


UIHostViewController
AG(AttributeGraph)::Graph::Node
Data storage
State
Environment
View

Вопросы

Куркин Дмитрий
dmitry.kurkin@spark.net
Telegram: @kurkinm
https://github.com/sclown