SwiftUI Views
Lifetime and Identity

26 February 2024

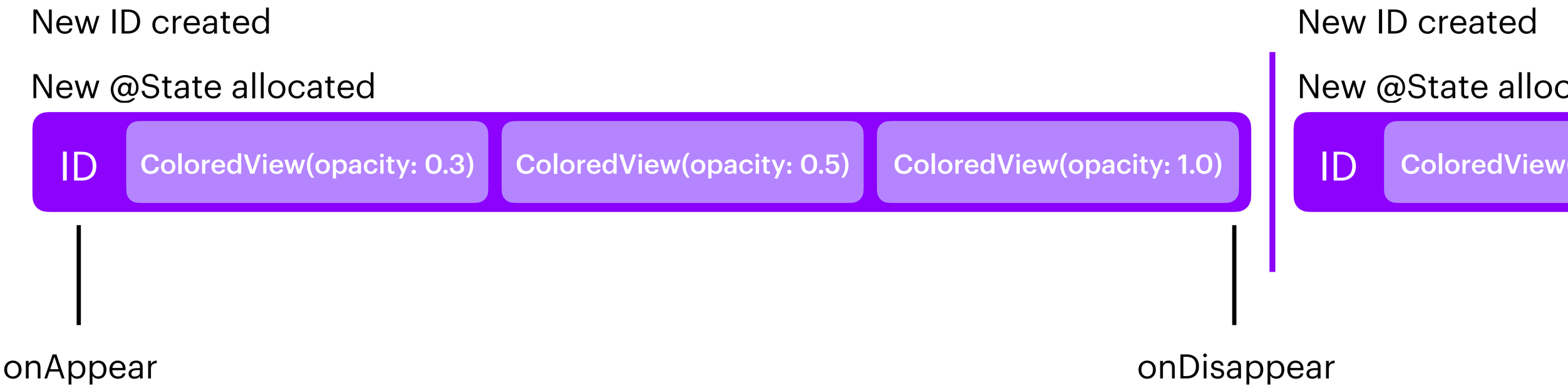
View Lifetime

View Identity

Identity in List

View Lifetime

- A view's value is short-lived
- A view's lifetime is the duration of its identity
- Persistence of state is tied to lifetime



Types of Identity

Explicit Identity

Structural Identity

Explicit Identity

```
struct ForEach<Data, ID, Content> where Data : RandomAccessCollection, ID : Hashable
```

```
struct FruitListView: View {
    let fruits = [
        Fruit(name: "Banana", color: .yellow),
        Fruit(name: "Cherry", color: .red)
    ]

    var body: some View {
        ScrollView {
            ForEach(fruits, id: \.name) { fruit in
                FruitView(fruit: fruit)
            }
        }
    }
}
```

```
func id<ID>(_ id: ID) -> some View where ID : Hashable
```

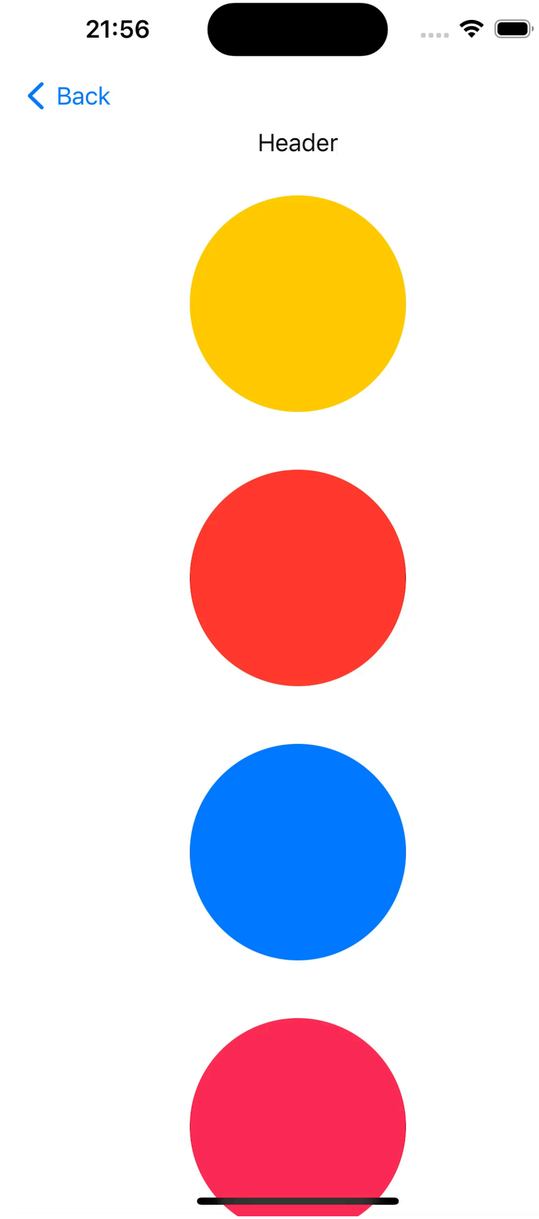
```
struct ExplicitIdentityScrollTo: View {
    let headerID = "header"

    let fruits = [...]

    var body: some View {
        ScrollView {
            ScrollViewReader { proxy in
                Text("Header")
                    .id(headerID)

                ForEach(fruits, id: \.name) { data in
                    CircleView(data: data)
                }

                Button("Scroll to top") {
                    withAnimation {
                        proxy.scrollTo(headerID)
                    }
                }
            }
        }
        .scrollIndicators(.hidden)
    }
}
```



How to choose stable identifier

✅ An identifier shouldn't change over time

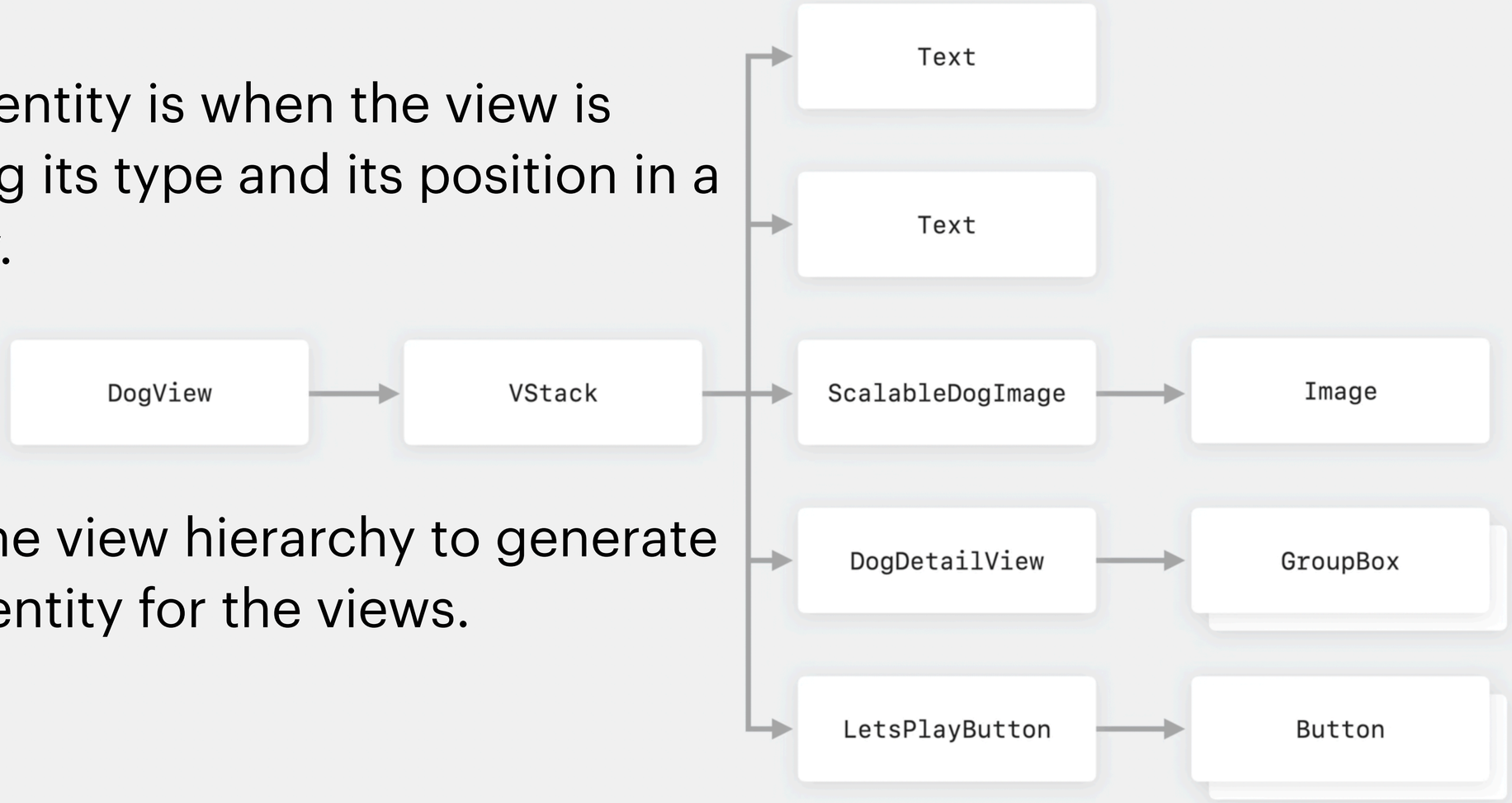
✅ Identifiers need to be unique.

Example

Structural Identity

- A structural identity is when the view is identified using its type and its position in a view hierarchy.

- SwiftUI uses the view hierarchy to generate the implicit identity for the views.



Example

Conditional View Modifiers
if statement
switch

⚠️ Broke Animations

⚠️ Reset @State @StateObject to its initial value

⚠️ Reduce performance

⚠️ Remember that you should use branching via if or switch statement only when you need to present different views.

Inert modifiers

✅ SwiftUI modifiers are cheap, so there is little inherent cost with this pattern.

```
.opacity(isEnabled ? 1 : 0)
.padding(isEnabled ? 8 : 0)
```

What type of identity do Lists use?

What type of identity do Lists use?

Both Explicit and Structural

Example

There are a few things to keep in mind to achieve a better performance:

✅ Maintain the view’s identity. If you can, don’t use conditional statements to preserve the identity.

✅ Use stable identifiers for your view if they are explicitly provided.

✅ Avoid using AnyView if possible

Q & A

Links

Demystify SwiftUI

Demystify SwiftUI Performance

How the SwiftUI View Lifecycle and Identity work

Why Conditional View Modifiers are a Bad Idea

Avoiding SwiftUI's AnyView

SwiftUI: Understanding identity via transitions