iOS Bootcamp - Meeting 2

Hosted by App Team Carolina

Agenda

What can you expect this meeting?

1. Icebreaker
2. Code Modularity
3. Structs + Subviews
4. Instagram refactor

Attendance!

Please fill this out!

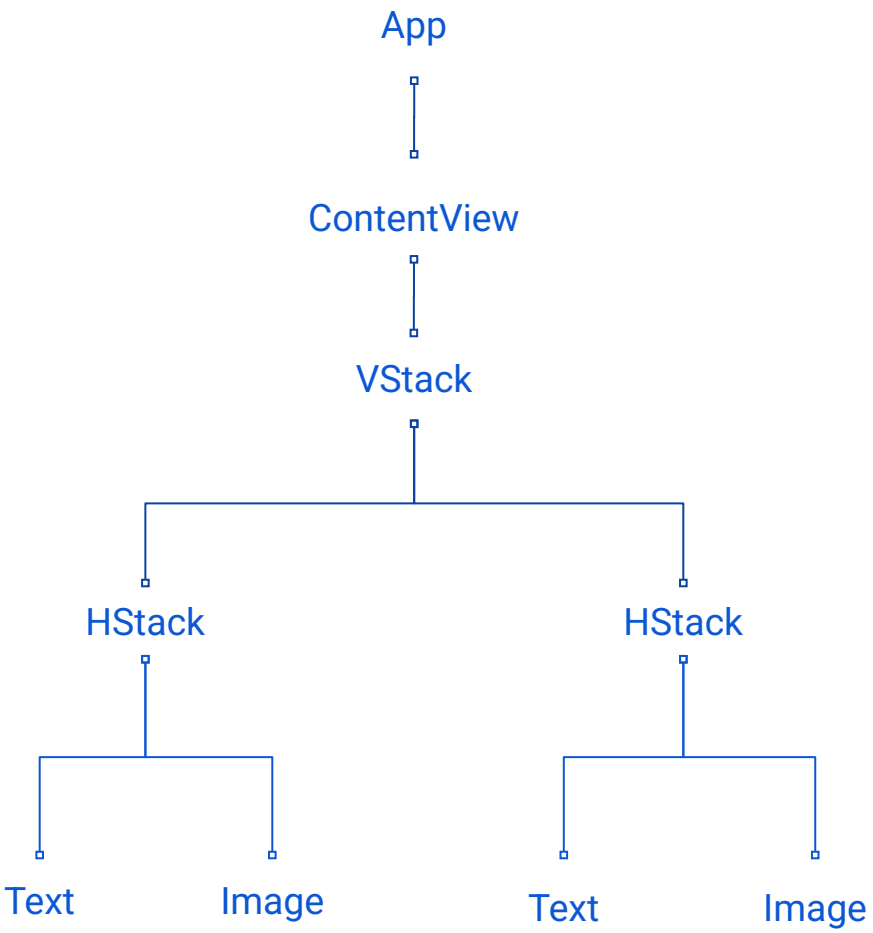
Code Modularity

Code Modularity

App Hierarchy

- SwiftUI apps → **nested tree of Views**
- Each View can contain **subviews**
- Easy to manage when small

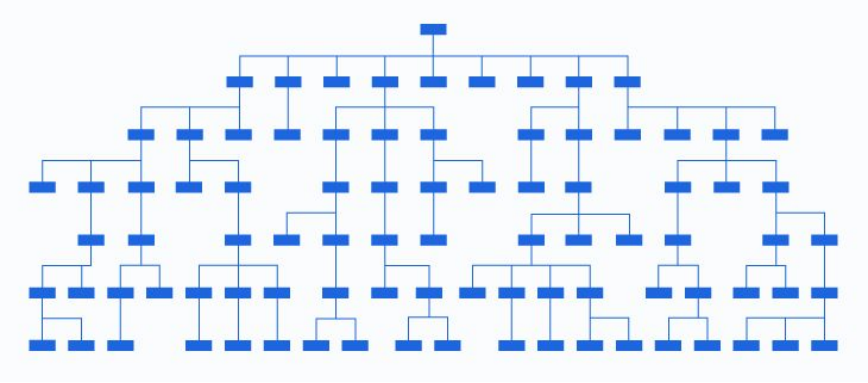
How would this diagram change as app complexity increases?



Code Modularity

App Hierarchy

Large apps (i.e. Instagram, TikTok, YouTube) have a **huge tree of Views.**



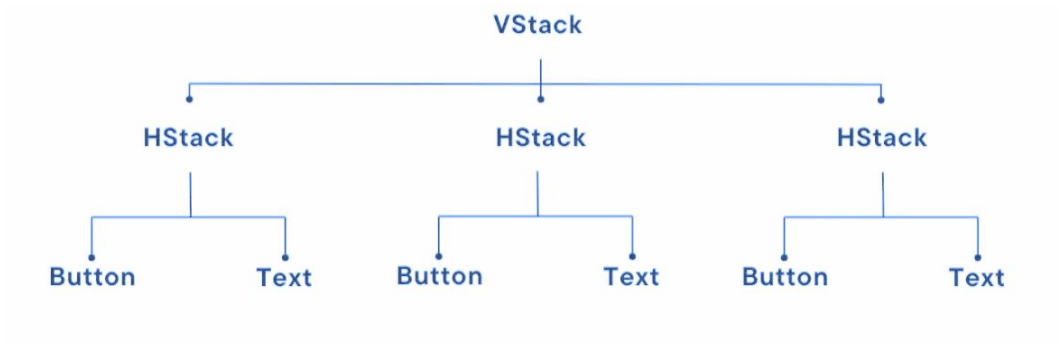
The more screens and nested components in an app, the harder to read and navigate it becomes.

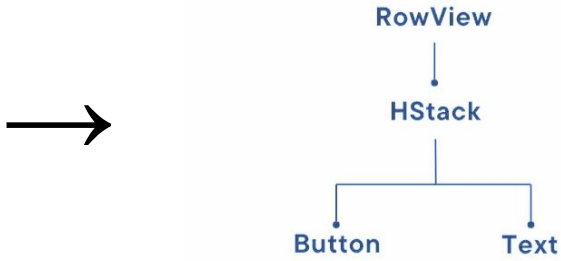
How do we solve this problem?

Code Modularity

Motivating Subviews

Code modularity in SwiftUI is the practice of breaking complex Views into smaller, reusable subviews.



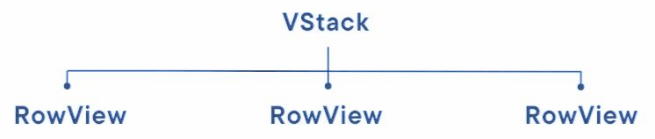


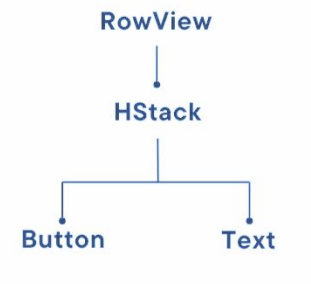
Code Modularity

Motivating Subviews

Why create subviews?

- **Reuse:** Write once, use anywhere
- **Maintain:** Fix a bug in one place
- **Clarity:** Descriptive names (e.g. RowView vs. generic HStack)





Structs and Subviews

Structs

What is a struct?

- A **struct** is a blueprint for a data object
- Groups **related properties** and optional behaviors (methods)
- Allows you to create many instances with the same structure.

```
struct User {
    var name: String
    var email: String
    var role: String
}
```

```
struct Rectangle {
    var width: Int
    var height: Int

    func area() -> Int {
        return width * height
    }
}
```

Structs

Struct vs. Class

While both Structs and Classes store related data, they have their differences:

- Structs are **value types** – each instance is copied when assigned or passed
- Classes are **reference types** – multiple variables can share the same instance

```
struct Point { var x = 0 }

func testPoints() {
    var a = Point()
    var b = a        // b is a copy
    b.x = 5          // a.x is still 0
}
```

Practice Creating Structs

Return to Notion

Structs

View structs

- Every SwiftUI View is a **struct**
- View structs **must** provide a body property that returns a View
- Can hold additional properties–including custom structs

```
struct ContentView: View {
    var body: some View {
        Text("Hello!")
    }
}


struct ProfileCardView: View {
    var user: User

    var body: some View {
        VStack {
            Text(user.email)
            Text(user.name)
        }
    }
}
```

Subviews

Example

```
struct ContentView: View {
    var body: some View {
        VStack(alignment: .leading) {
            HStack {
                Image(systemName: "person.circle")
                Text("Alexandra")
            }
            HStack {
                Image(systemName: "person.circle")
                Text("Hussain")
            }
            HStack {
                Image(systemName: "person.circle")
                Text("Alex")
            }
            HStack {
                Image(systemName: "person.circle")
                Text("Tri")
            }
        }
        .padding()
    }
}
```

→

```
struct ContentView: View {
    var body: some View {
        VStack(alignment: .leading) {
            ProfileCard(name: "Alexandra")
            ProfileCard(name: "Hussain")
            ProfileCard(name: "Alex")
            ProfileCard(name: "Tri")
        }
        .padding()
    }
}

struct ProfileCard: View {
    var name: String

    var body: some View {
        HStack {
            Image(systemName: "person.circle")
            Text(name)
        }
    }
}
```

Practice Creating Subviews

Return to Notion

Subviews

Dynamic reuse with ForEach

ForEach:

- Is a UI version of a for-loop
- Iterates over any **collection** (array, range, etc.)
- Creates one instance of the view per element

```
struct ForEachView: View {
    let names = ["Hussain", "Alex", "Tri"]

    var body: some View {
        ForEach(names, id: \.self) { name in
            Text(name)
        }
    }
}
```

Hussain
Alex
Tri

The **id:** param tells SwiftUI how to uniquely identify each view.

Subviews

Using ForEach with custom structs

- When you use ForEach with simple types (like String or Int), you can rely on id: \.self.
- **Custom structs** require a different way to uniquely identify each item

```
var body: some View {
    ForEach(profiles, id: \.self) { profile in
        ProfileCardView(profile: profile)
    }
```

Solution: Make your struct conform to **Identifiable**

Subviews

Using ForEach with custom structs

Identifiable tells SwiftUI: *"Each instance can be uniquely tracked by an **id property.**"*

This prevents glitches when adding, removing, or reordering items.

```
struct Profile: Identifiable {
    var id = UUID()
    var name: String
    var age: Int
}
```

```
var body: some View {
    ForEach(profiles) { profile in
        ProfileCardView(profile: profile)
    }
}
```

Use **UUID()** to generate a unique identifier whenever a new object is created.

Practice Using ForEach

Return to Notion

Instagram 2.0

Return to Notion