I've got style, miles and miles

So much style that it's wasted

iOSDevUK 2023: Richard Turton

Why should I listen to this idiot?

SwiftUI in a perfect world

Does your desk look like this?

SwiftUI in a perfect world

Does your code look like this?

```
Form {
    TextField("User Name", text: $username)
    SecureField("Password", text: $password)
    Button("Log In") { login() }
}
```

SwiftUI in the actual world

My desk looks like this

SwiftUI in the actual world

When design happens

```
VStack {
  Text("Name")
  Text(name)
}
```

Name
Richard

SwiftUI in the actual world

When design happens

```
VStack {
  Text("Name")
    .foregroundColor(.secondary)
  Text(name)
}
```

SwiftUI in the actual world

When design happens

```
VStack(alignment: .leading) {
  Text("Name")
    .foregroundColor(.secondary)
  Text(name)
}
```

SwiftUI in the actual world

When design happens

```
VStack(alignment: .leading) {
  Text("Name")
    .foregroundColor(.secondary)
    .font(.caption.smallCaps())
  Text(name)
}
```

NAME

Richard

SwiftUI in the actual world

When design happens

```
VStack(alignment: .leading) {
  Text("Name")
    .foregroundColor(.secondary)
    .font(.caption.smallCaps())
  Divider()
  Text(name)
}
```

SwiftUI in the actual world

When design happens

```
HStack {
  VStack(alignment: .leading) {
    Text("Name")
      .foregroundColor(.secondary)
      .font(.caption.smallCaps())
    Divider()
    Text(name)
  }
  VStack(alignment: .leading) {
    Text("Name")
      .foregroundColor(.secondary)
      .font(.caption.smallCaps())
    Divider()
    Text(score)
  }
}
```

NAME	SCORE
Richard	-3

SwiftUI in the actual world

When design happens

```
HStack {
  VStack(alignment: .leading) {
    Text("Name")
      .foregroundColor(.secondary)
      .font(.caption.smallCaps())
    Divider()
    Text(name)
  }
  VStack(alignment: .leading) {
    Text("Status")
      .foregroundColor(.secondary)
      .font(.caption.smallCaps())
    Divider()
    status
  }
  VStack(alignment: .leading) {
    Text("Score")
      .foregroundColor(.secondary)
      .font(.caption.smallCaps())
    Divider()
    Text(score)
  }
}
```

NAME	STATUS	SCORE
Richard	☀	-3

Let’s fix that with custom modifiers

There are a million ways to do anything in SwiftUI and they’re all super easy

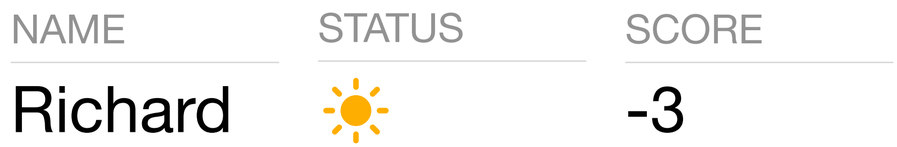
```
HStack {
  VStack(alignment: .leading) {
    Text("Name")
      .foregroundColor(.secondary)
      .font(.caption.smallCaps())
    Divider()
    Text(name)
  }
  VStack(alignment: .leading) {
    Text("Status")
      .foregroundColor(.secondary)
      .font(.caption.smallCaps())
    Divider()
    status
  }
  VStack(alignment: .leading) {
    Text("Score")
      .foregroundColor(.secondary)
      .font(.caption.smallCaps())
    Divider()
    Text(score)
  }
}
```

NAME	STATUS	SCORE
Richard	☀	-3

Let’s fix that with custom modifiers

There are a million ways to do anything in SwiftUI and they’re all super easy

```
HStack {
  VStack(alignment: .leading) {
    Text("Name")
      .titleFormatted
    Divider()
    Text(name)
  }
  VStack(alignment: .leading) {
    Text("Status")
      .titleFormatted
    Divider()
    status
  }
  VStack(alignment: .leading) {
    Text("Score")
      .titleFormatted
    Divider()
    Text(score)
  }
}
```



Let's fix that with specialist views

There are a million ways to do anything in SwiftUI and they’re all super easy

```
struct Field: View {
  let label: String
  let value: String

  var body: some View {
    VStack(alignment: .leading) {
      Text(label)
        .foregroundColor(.secondary)
        .font(.caption.smallCaps())
      Divider()
      Text(value)
    }
  }
}
```

Let's fix that with specialist views

There are a million ways to do anything in SwiftUI and they’re all super easy

```
Field(label: “Name”, value: name)
```

Let's fix that with specialist views

There are a million ways to do anything in SwiftUI and they're all super easy … but they're not all the best

```
Field(label: “Name”, value: name)
```

- Localization
- Accessibility
- What about that one case where you gave it an image and not some text

Let's actually fix that with LabeledContent

This is the way

~~Field(label: “Name”, value: name)~~

```
HStack {
  LabeledContent("Name", value: name)
  LabeledContent("Status") { status }
  LabeledContent("Score", value: score, format: .number)
}
```

LabeledContent doesn’t look right!

```
LabeledContent("Name", value: name)
```


```
func labeledContentStyle<S>(_ style: S) -> some View
where S : LabeledContentStyle
```

```
AutomaticLabeledContentStyle
```

Implementing LabeledContentStyle

```
struct StackedLabeledContentStyle: LabeledContentStyle {
    …
}
```

Implementing LabeledContentStyle

```
struct StackedLabeledContentStyle: LabeledContentStyle {
  func makeBody(configuration: Configuration) -> some View {
    VStack(alignment: .leading) {
      configuration.label
        .foregroundColor(.secondary)
        .font(.caption)
      Divider()
      configuration.content
      }
  }
}
```

NAME

Richard

Using a custom LabeledContentStyle

```
HStack {
  LabeledContent("Name", value: name)
    .labeledContentStyle(StackedLabeledContentStyle())
  LabeledContent("Status") { status }
    .labeledContentStyle(StackedLabeledContentStyle())
  LabeledContent("Score", value: score, format: .number)
    .labeledContentStyle(StackedLabeledContentStyle())
}
```

Using a custom LabeledContentStyle

```
HStack {
  LabeledContent("Name", value: name)
  LabeledContent("Status") { status }
  LabeledContent("Score", value: score, format: .number)
}
.labeledContentStyle(StackedLabeledContentStyle())
```

Using a custom LabeledContentStyle

```
HStack {
  LabeledContent("Name", value: name)
  LabeledContent("Status") { status }
  LabeledContent("Score", value: score, format: .number)
}
```

…

```
.labeledContentStyle(StackedLabeledContentStyle())
```

Using a custom LabeledContentStyle

```
HStack {
  LabeledContent("Name", value: name)
  LabeledContent("Status") { status }
  LabeledContent("Score", value: score, format: .number)
}
```

…

```
.labeledContentStyle(.stacked)
```

Other ~~Styleable Stylable~~ Views You Can Style

- Button
- ControlGroup
- DatePicker
- DisclosureGroup
- Form
- Gauge
- GroupBox
- IndexView
- Label
- LabeledContent
- List
- Menu
- NavigationSplitView
- Picker
- ProgressView
- TabView
- Table
- TextField
- Toggle

Other ~~Styleable Stylable~~ Views You Can Style

- Button
- ControlGroup
- DatePicker
- DisclosureGroup
- Form
- Gauge
- GroupBox
- IndexView
- Label
- LabeledContent
- List
- Menu
- NavigationSplitView
- Picker
- ProgressView
- TabView
- Table
- TextField
- Toggle

Other ~~Styleable Stylable~~ Views You Can Style

- Button
- ControlGroup
- DatePicker
- DisclosureGroup
- Form
- Gauge
- GroupBox
- IndexView
- Label
- LabeledContent
- List
- Menu
- NavigationSplitView
- Picker
- ProgressView
- TabView
- Table
- TextField
- Toggle

Other ~~Styleable Stylable~~ Views You Can Style

- Button
- ControlGroup
- DatePicker
- DisclosureGroup
- Form
- Gauge
- GroupBox
- IndexView
- Label
- LabeledContent
- List
- Menu
- NavigationSplitView
- Picker
- ProgressView
- TabView
- Table
- TextField
- Toggle

Other ~~Styleable Stylable~~ Views You Can Style

- Button
- ControlGroup
- DatePicker
- DisclosureGroup
- Form
- Gauge
- GroupBox
- IndexView
- Label
- LabeledContent
- List
- Menu
- NavigationSplitView
- Picker
- ProgressView
- TabView
- Table
- TextField
- Toggle

Other ~~Styleable Stylable~~ Views You Can Style

- Button
- ControlGroup
- DatePicker
- DisclosureGroup
- Form
- Gauge
- GroupBox
- IndexView
- Label
- LabeledContent
- List
- Menu
- NavigationSplitView
- Picker
- ProgressView
- TabView
- Table
- TextField
- Toggle

Other ~~Styleable Stylable~~ Views You Can Style

- Button
- ControlGroup
- DatePicker
- DisclosureGroup
- Form
- Gauge
- GroupBox
- IndexView
- Label
- LabeledContent
- List
- Menu
- NavigationSplitView
- Picker
- ProgressView
- TabView
- Table
- TextField
- Toggle

Other ~~Styleable Stylable~~ Views You Can Style

- Button
- ControlGroup
- DatePicker
- DisclosureGroup
- Form
- Gauge
- GroupBox
- IndexView
- Label
- LabeledContent
- List
- Menu
- NavigationSplitView
- Picker
- ProgressView
- TabView
- Table
- TextField
- Toggle

Common patterns in styles

Protocol Configuration Environment Modifier Implementation

```
protocol XXStyle {

    associatedtype Body: View

    @ViewBuilder
    func makeBody(configuration: Self.Configuration) ->
      Self.Body

    typealias Configuration = XXStyleConfiguration

}
```

Common patterns in styles

Protocol **Configuration** Environment Modifier Implementation

```
struct XXStyleConfiguration {

    let viewContent: AnyView

    let modelContent: ??

}
```

AnyInterlude

```
public struct LabeledContentStyleConfiguration {

    /// A type-erased label of a labeled content instance.
    public struct Label : View {
        public typealias Body = Never
    }

    …

}

@frozen public struct AnyView : View {

    /// Create an instance that type-erases `view`.
    public init<V>(_ view: V) where V : View

    public init<V>(erasing view: V) where V : View
    public typealias Body = Never
}
```

AnyInterlude

```
LabeledContent("Status") { status }
```

```
.labeledContentStyle(.stacked)
```

Inside LabeledContent...

```
let style: any LabeledContentStyle
```

Inside StackedLabeledContentStyle...

```
func makeBody(configuration: Configuration) -> some View {
  configuration.content
}
```

AnyInterlude

```
struct Anyhow: View {

    let somethingElse: any View

    var body: some View {  Type 'any View' cannot conform to 'View'
        somethingElse
    }
}
```

AnyInterlude

```
struct Anyhow: View {

    let somethingElse: any View

    var body: some View {
        AnyView(somethingElse)
    }
}
```

Common patterns in styles

Protocol **Configuration** Environment Modifier Implementation

```
struct XXStyleConfiguration {

    let viewContent: AnyView

    let modelContent: ??

}
```

Common patterns in styles

Protocol Configuration **Environment** Modifier Implementation

```
struct XXStyleKey: EnvironmentKey {
    static let defaultValue: any XXStyle = DefaultXXStyle()
}

extension EnvironmentValues {
    var xxStyle: any XXStyle {
        get { self[XXStyleKey.self] }
        set { self[XXStyleKey.self] = newValue }
    }
}
```

Common patterns in styles

Protocol Configuration Environment **Modifier** Implementation

```
extension View {
    func xxStyle(_ style: any xxStyle) -> some View {
        self.environment(\.xxStyle, style)
    }
}

extension XXStyle where Self == ThisXXStyle {
    static var this: Self { ThisXXStyle() }
}

.xxStyle(.this)
```

Common patterns in styles

Protocol Configuration Environment Modifier **Implementation**

```
struct XX<ViewContent: View>: View {

    @Environment(\.xxStyle) private var style
    let configuration: XXStyleConfiguration
    init(_ modelStuff: ??, @ViewBuilder content: () -> ViewContent) {
        configuration = .init(
            viewContent: .init(content()),
            modelContent: modelStuff)
    }

    var body: some View {
        style.makeBody(configuration: configuration)
    }
}
```

Type 'any View' cannot conform to 'View'

Common patterns in styles

Protocol Configuration Environment Modifier **Implementation**

```
struct XX<ViewContent: View>: View {

    @Environment(\.xxStyle) private var style
    let configuration: XXStyleConfiguration
    init(_ modelStuff: ??, @ViewBuilder content: () -> ViewContent) {
        configuration = .init(
            viewContent: .init(content()),
            modelContent: modelStuff)
    }

    var body: some View {
        AnyView(style.makeBody(configuration: configuration))
    }
}
```

Making your own view with styles

User Profile View

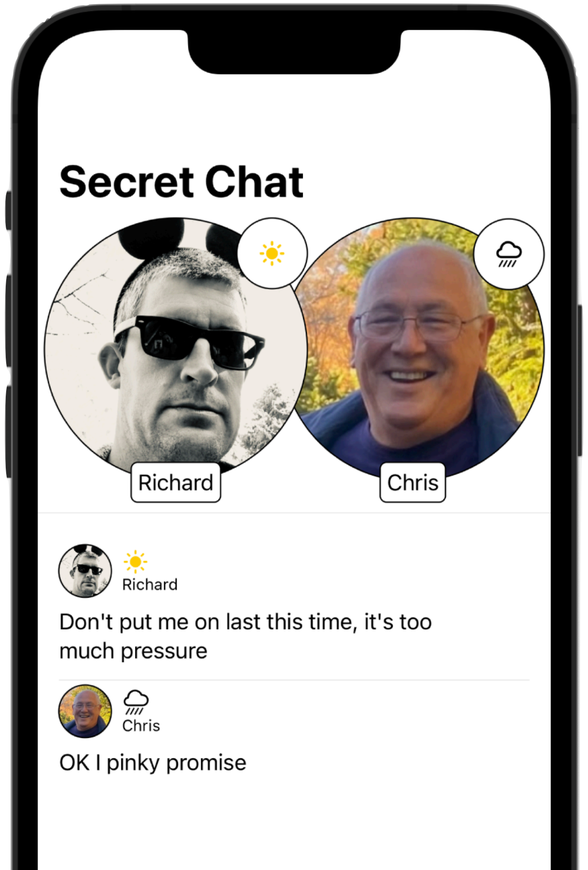
Protocol Configuration Environment Modifier Implementation

```
protocol UserProfileStyle {

    associatedtype Body: View

    @ViewBuilder
    func makeBody(configuration:
        Self.Configuration) -> Self.Body

    typealias Configuration =
        UserProfileStyleConfiguration
}
```

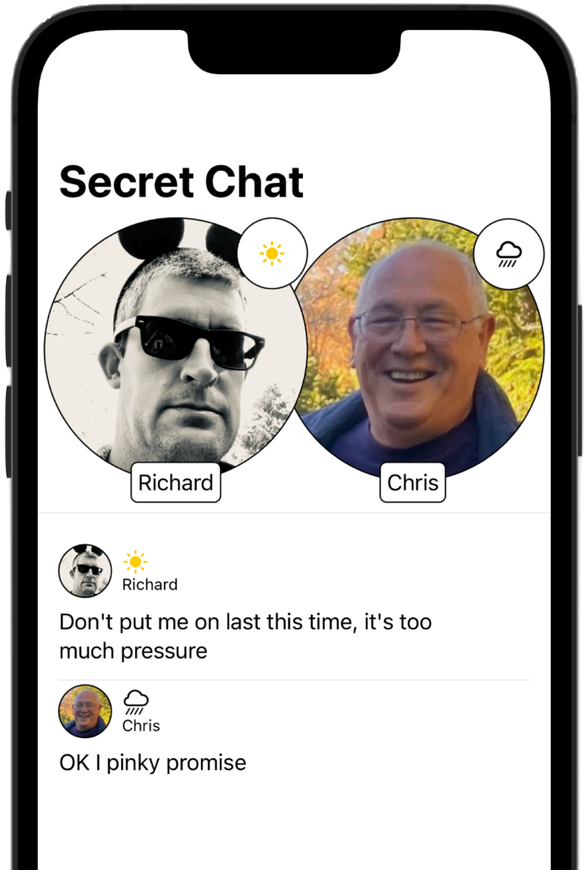


Making your own view with styles

User Profile View

Protocol **Configuration** Environment Modifier Implementation

```
struct UserProfileStyleConfiguration {
    let picture: AnyView
    let badge: AnyView
    let name: String
}
```



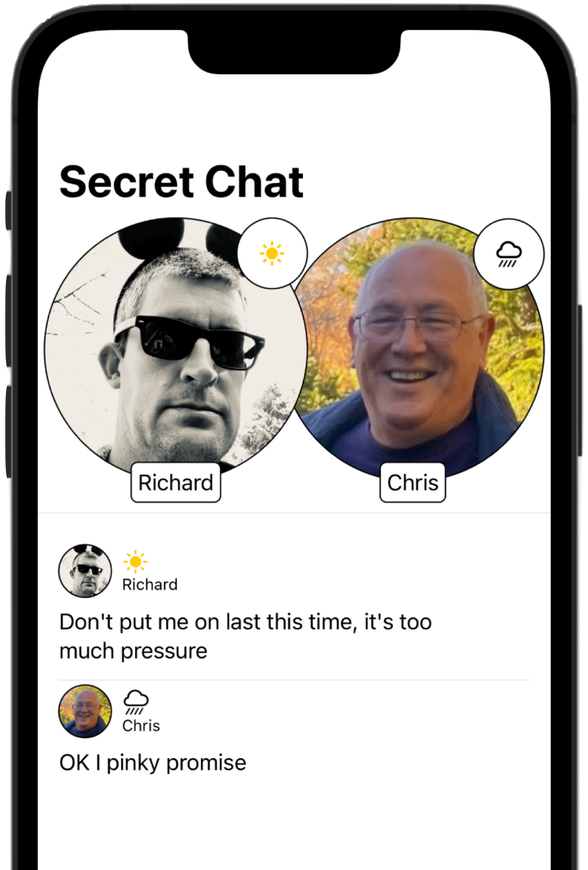
Making your own view with styles

User Profile View

Protocol Configuration **Environment** Modifier Implementation

```
struct UserProfileStyleKey: EnvironmentKey {
  static let defaultValue:
    any UserProfileStyle = HeroUserProfileStyle()
}

extension EnvironmentValues {
  var userProfileStyle: any UserProfileStyle {
    get { self[UserProfileStyleKey.self] }
    set { self[UserProfileStyleKey.self] = newValue }
  }
}
```

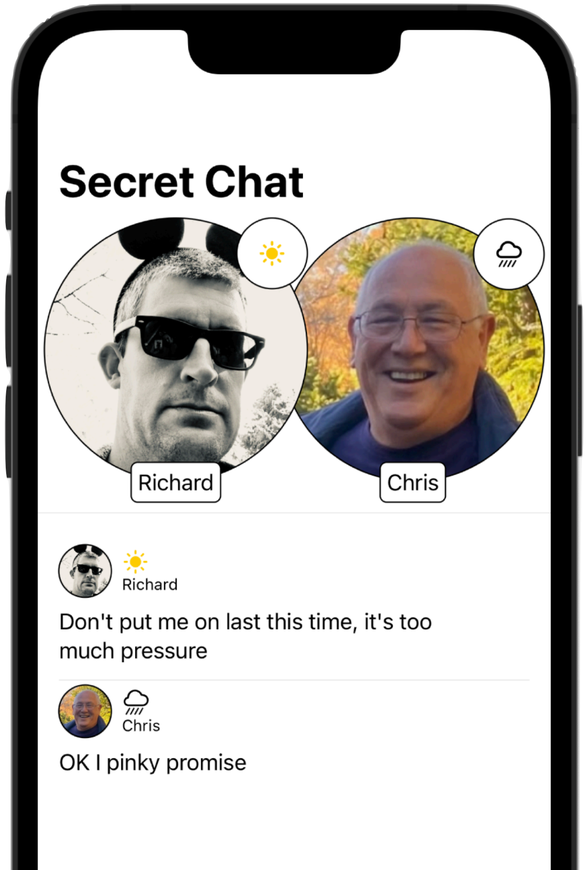


Making your own view with styles

User Profile View

Protocol Configuration Environment **Modifier** Implementation

```
extension View {
   func playerProfileStyle(
      _ style: any UserProfileStyle) -> some View {
         self.environment(\.userProfileStyle, style)
      }
}
```



Making your own view with styles

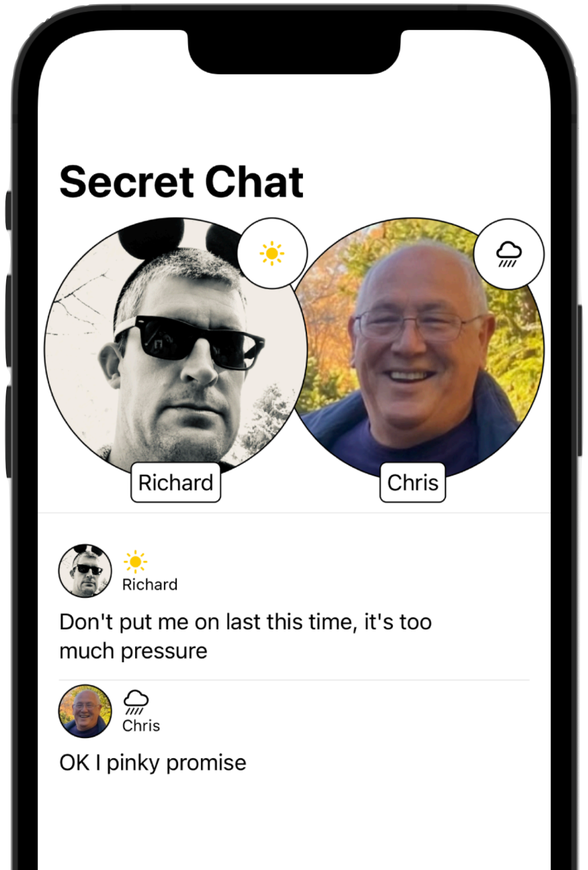
User Profile View

Protocol Configuration Environment Modifier **Implementation**

```
struct UserProfile<Picture: View, Badge: View>: View {

  @Environment(\.userProfileStyle) private var style
  let configuration: UserProfileStyleConfiguration
  init(
    _ name: String,
    @ViewBuilder picture: () -> Picture,
    @ViewBuilder badge: () -> Badge) {
      configuration = .init(
        picture: .init(picture()),
        badge: .init(badge()),
        name: name)
    }

    var body: some View {
      AnyView(style.makeBody(configuration: configuration))
    }
}
```

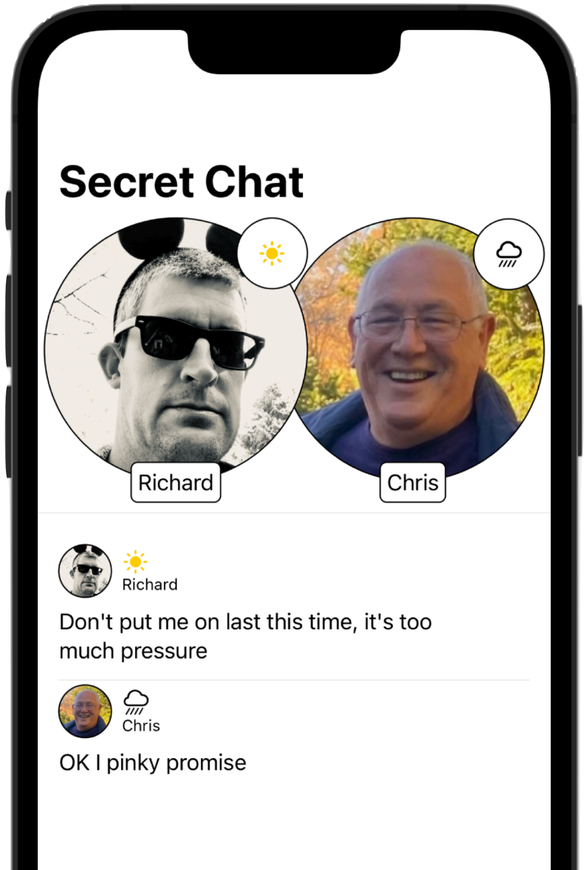


Making your own view with styles

User Profile View

Protocol Configuration Environment Modifier **Implementation**

```
struct HeroUserProfileStyle: UserProfileStyle {
    func makeBody(configuration: Configuration) -> some View {
        configuration.picture
            .frame(width: 200, height: 200)
            .clipShape(Circle())
            .overlay {
                Circle()
                    .stroke()
            }
            .overlay(alignment: .topTrailing) {
                configuration.badge
                    .font(.headline)
                    .padding()
                    .background(in: Circle())
                    .overlay { Circle().stroke() }
            }
            .padding(.bottom)
            .overlay(alignment: .bottom) {
                Text(configuration.name)
                    .padding(5)
                    .background(in: RoundedRectangle(cornerRadius: 5))
                    .overlay { RoundedRectangle(cornerRadius: 5).stroke()}
            }
    }
}
```



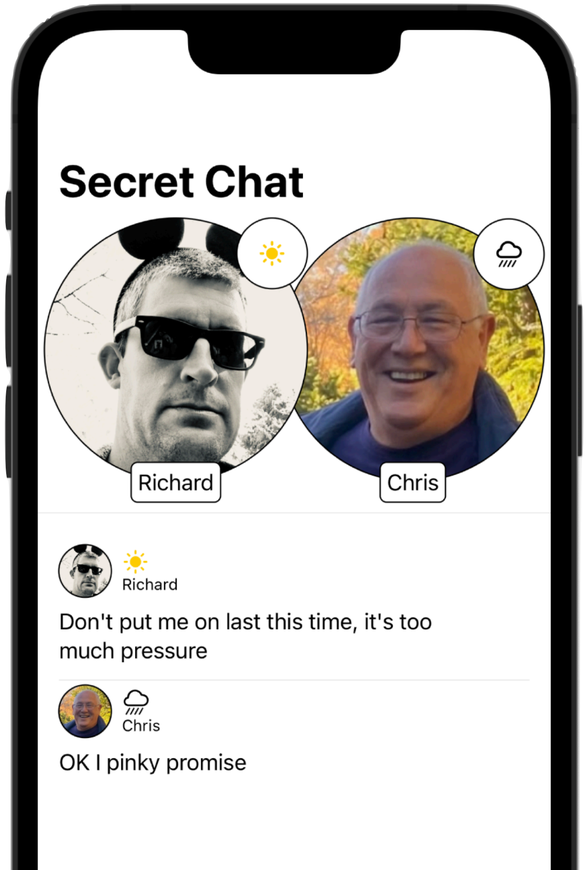
Making your own view with styles

User Profile View

Protocol Configuration Environment Modifier **Implementation**

```
struct ChatAvatarUserProfileStyle: UserProfileStyle {

    func makeBody(configuration: Configuration) -> some View {
        HStack {
            configuration.picture
                .frame(width: 40, height: 40)
                .clipShape(Circle())
                .overlay {
                    Circle()
                        .stroke()
                }
            VStack(alignment: .leading) {
                configuration.badge
                Text(configuration.name)
                    .font(.caption)
            }
        }
    }
}
```

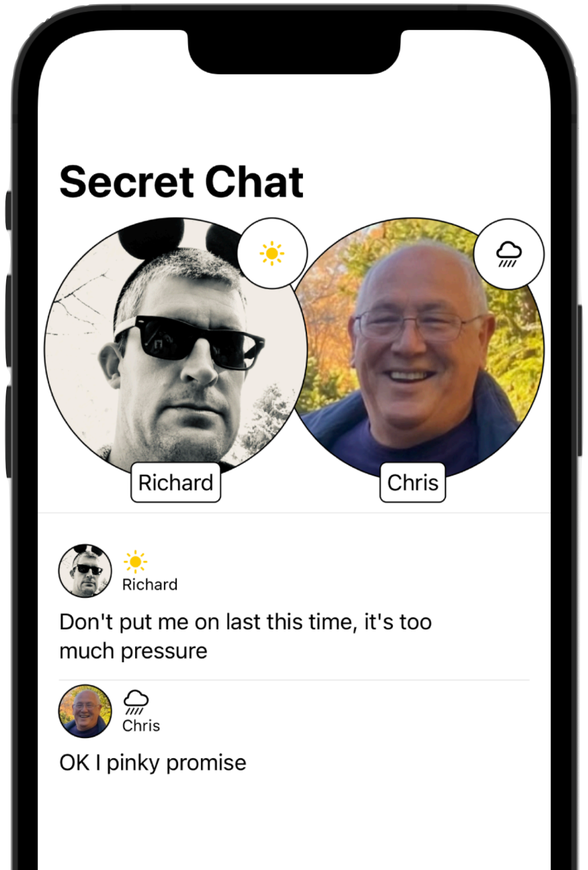


Making your own view with styles

User Profile View

Protocol Configuration Environment Modifier **Implementation**

```
VStack {
  HStack(spacing: -20) {
    profile
      .zIndex(1)
    chrisProfile
  }
  .playerProfileStyle(.hero)
  Divider()
  VStack(alignment: .leading) {
    profile
    Text("Don't put me on last this time, it's too much pressure")
    Divider()
    chrisProfile
    Text("OK I pinky promise")
  }
  .playerProfileStyle(.chat)
  .padding()
}
```



Conclusions

- Use first party views and style them where possible
- Making your own styled views is possible
- Don’t pick fights with the compiler
- Clean your desk, you filthy animal

Thankyou!