SwiftUI

Under the Hood

Layouts and Updates

March 11, 2020

github.com/qmchenry/SwiftUI-Presentations

Topics

- Simple layouts
- Representation
- Updating
- Opaque types

```
Text("Hello Planet!")
```

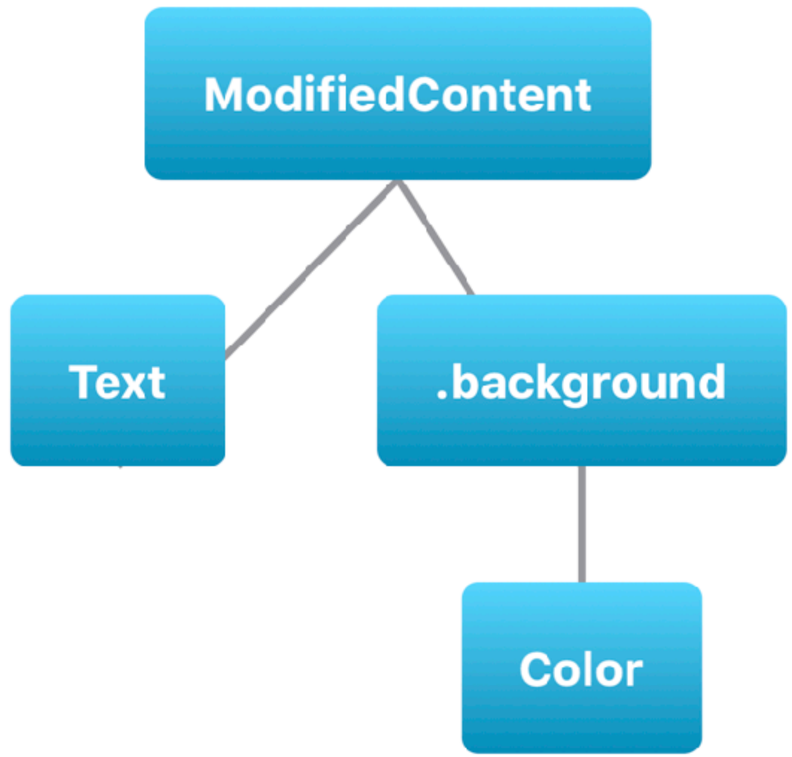
Hello Planet!

Text

Text

```
Text("Hello Planet!")
  .background(Color.blue)
```

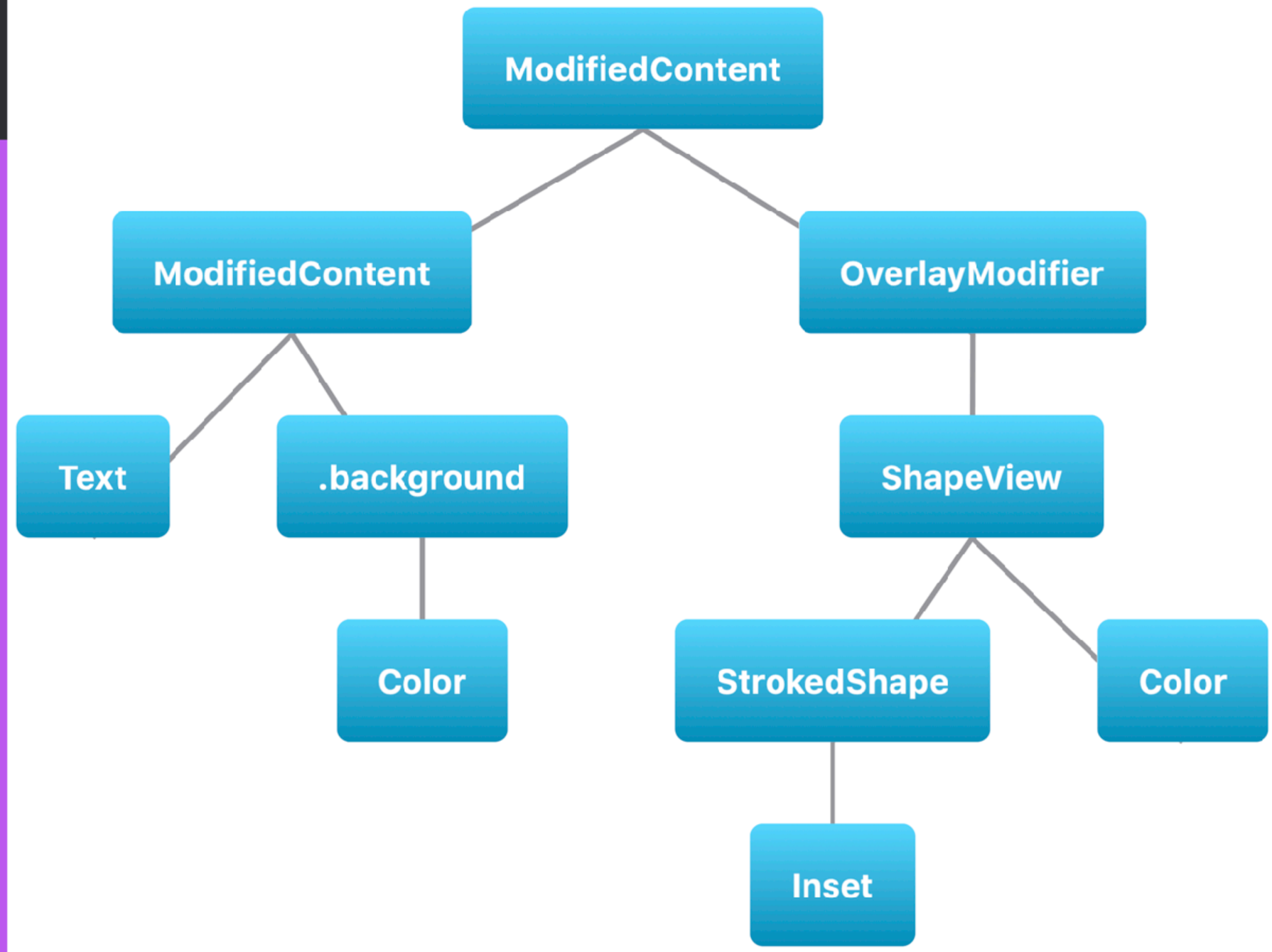
Hello Planet!




```
ModifiedContent<Text, _BackgroundModifier<Color>>
```

```
Text("Hello Planet!")
  .background(Color.blue)
  .border(Color.white)
```

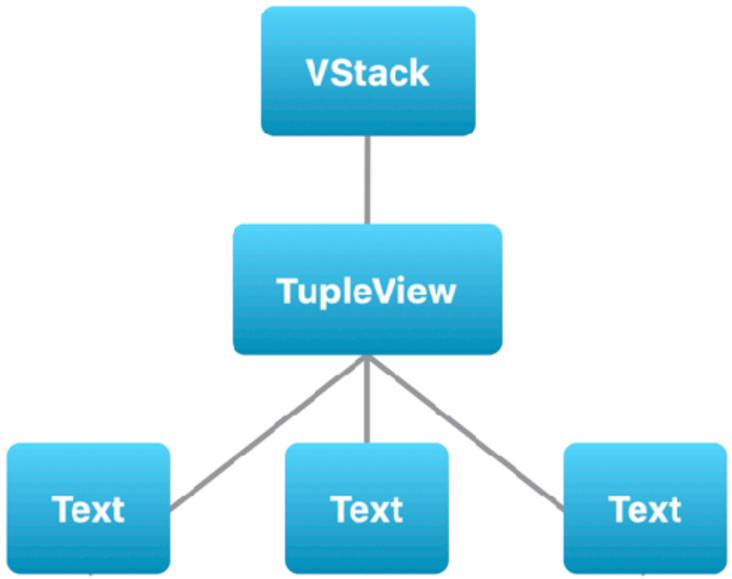
Hello Planet!




```
ModifiedContent<ModifiedContent<Text, _BackgroundModifier<Color>>,
_OverlayModifier<_ShapeView<_StrokedShape<_Inset>, Color>>>
```

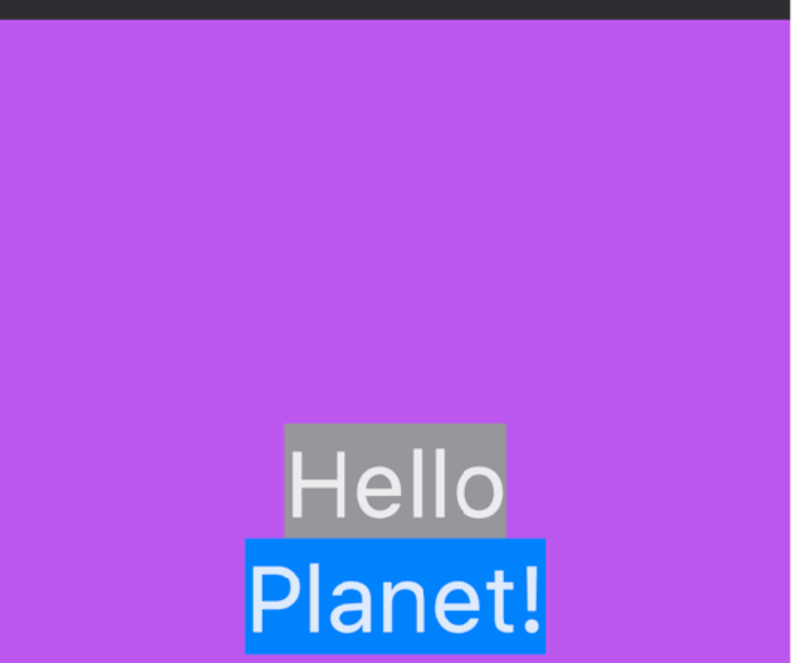
```
VStack {
    Text("Hello")
    Text("Planet!")
}
```

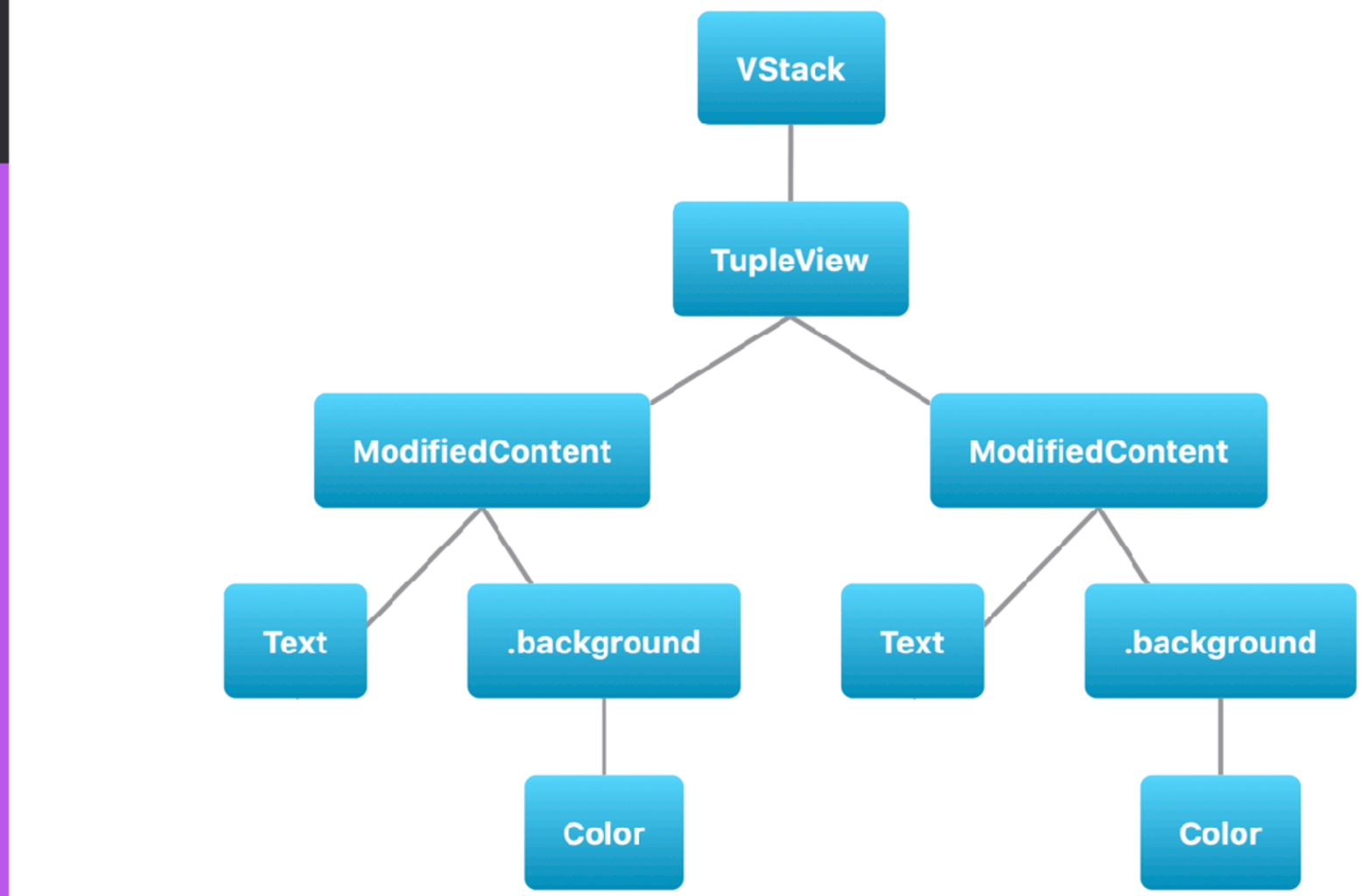
Hello
Planet!



VStack<TupleView<(Text, Text, Text)>>

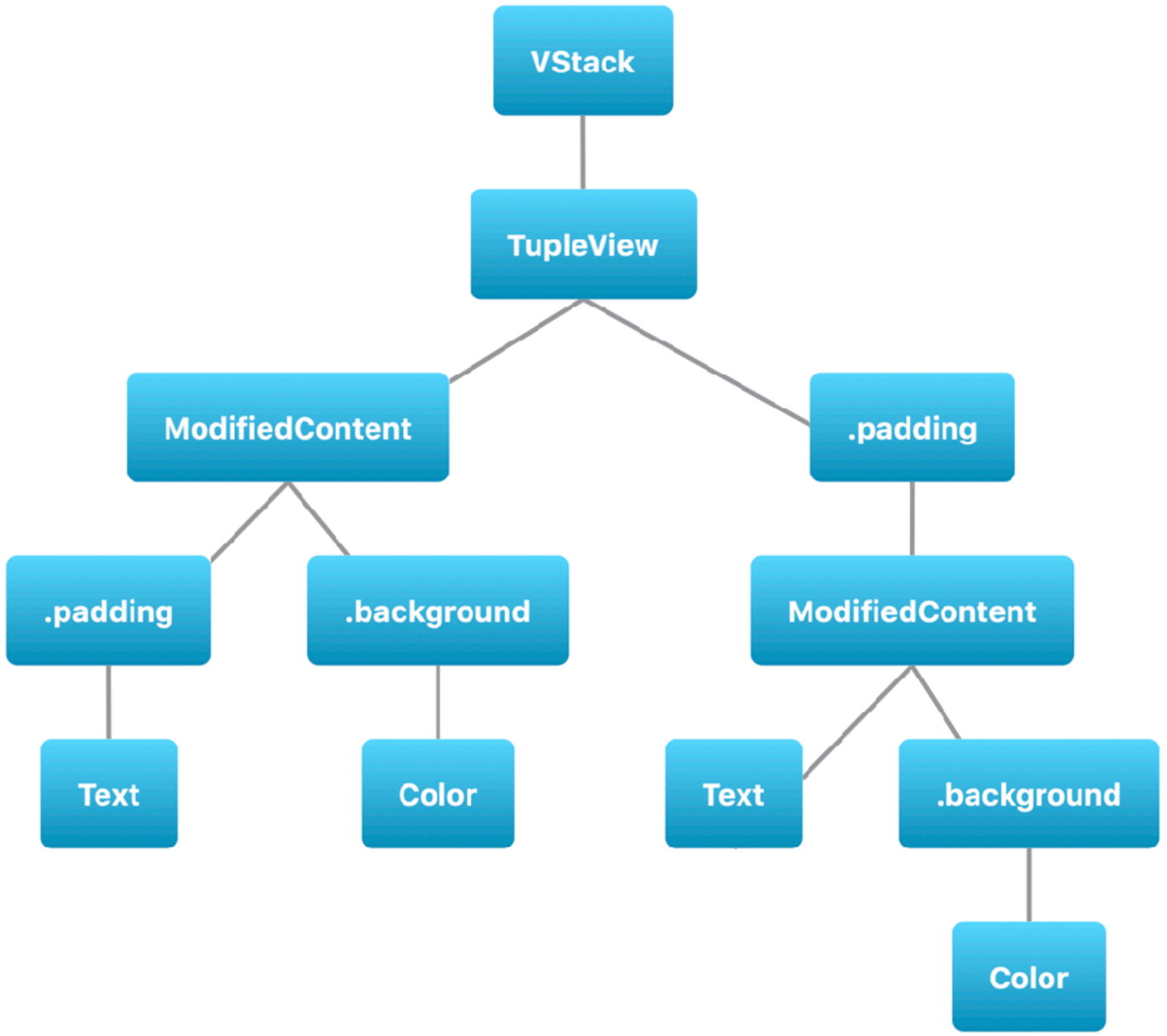
```
VStack {
  Text("Hello")
    .background(Color.gray)
  Text("Planet!")
    .background(Color.blue)
}
```






```
VStack<TupleView<(ModifiedContent<Text, _BackgroundModifier<Color>>, ModifiedContent<Text,
_BackgroundModifier<Color>>)>>
```

```
VStack {
  Text("Hello")
    .padding()
    .background(Color.gray)
  Text("Planet!")
    .background(Color.blue)
    .padding()
}
```

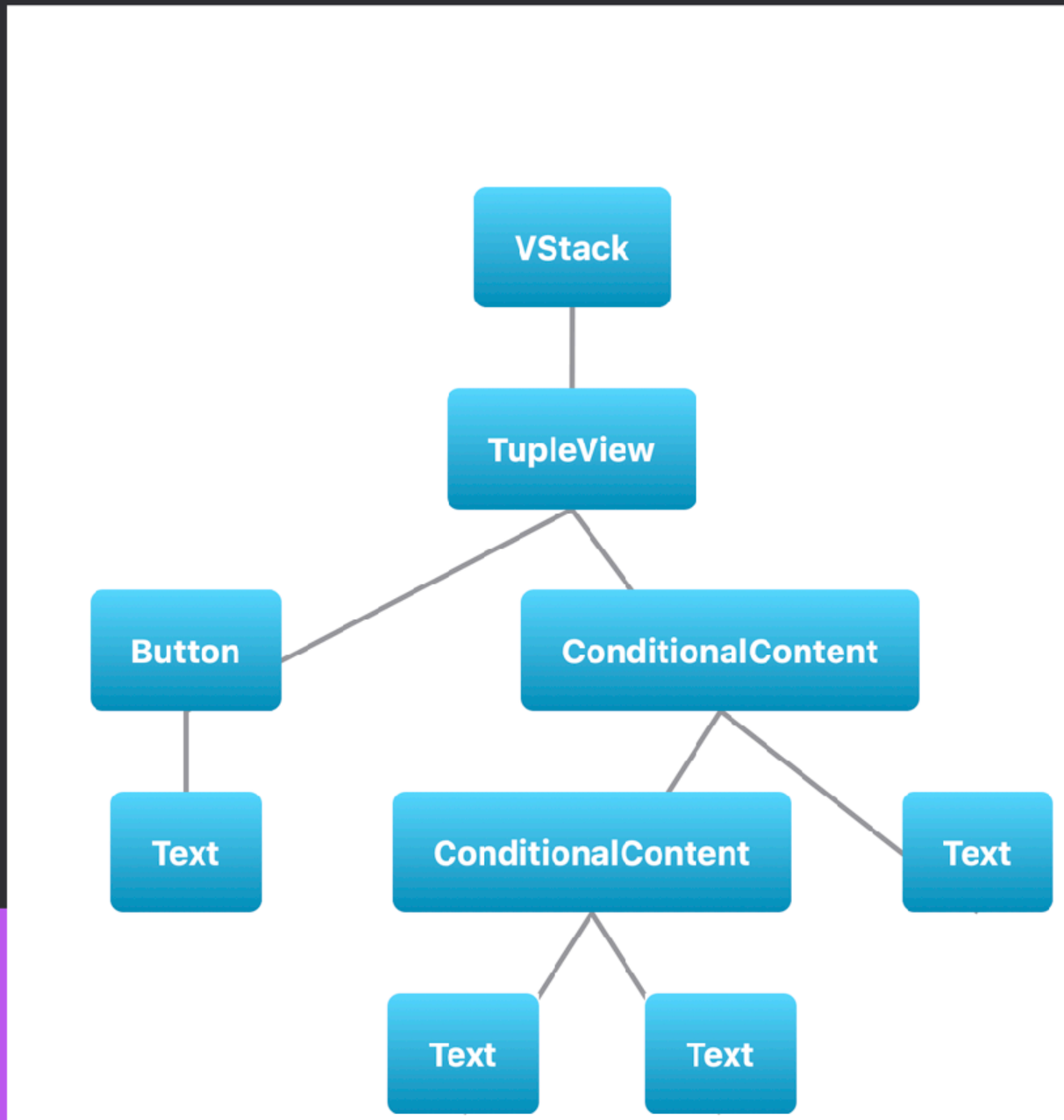



```
VStack<TupleView<(ModifiedContent<ModifiedContent<Text, _PaddingLayout>, _BackgroundModifier<Color>>,
ModifiedContent<ModifiedContent<Text, _BackgroundModifier<Color>>, _PaddingLayout>)>>
```

```
@State var counter = 0
var body: some View {
  VStack {
    Button("Click here") {
      self.counter += 1
    }
    if counter == 0 {
      Text("No clicks, weirdo!")
    } else if counter == 1 {
      Text("\(counter) click, weirdo!")
    } else {
      Text("\(counter) clicks, weirdo!")
    }
  }
}
```

Click here

No clicks, weirdo!

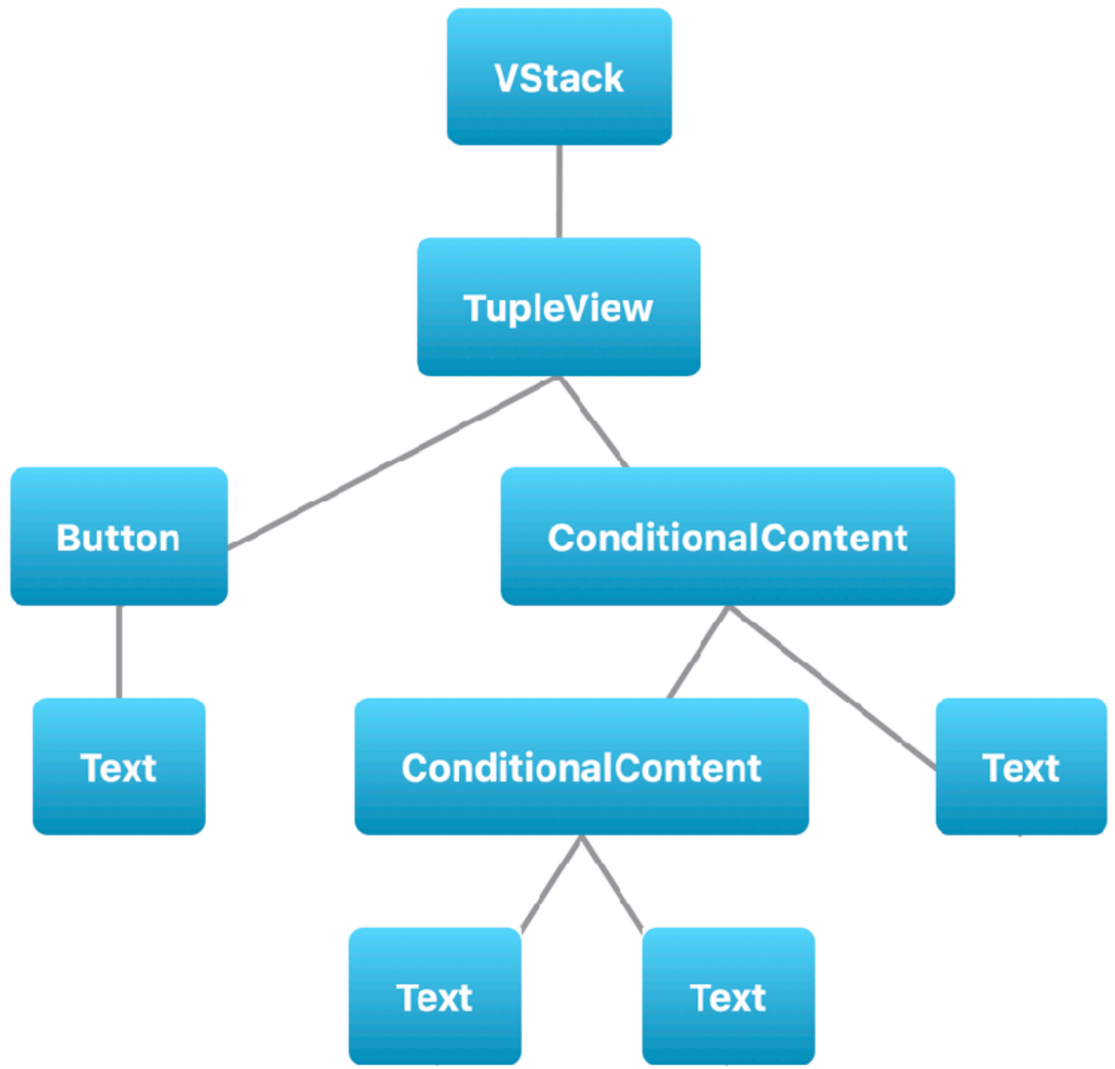



```
VStack<TupleView<(Button<Text>, _ConditionalContent<_ConditionalContent<Text, Text>, Text>)>>
```

```
@State var counter = 0
var body: some View {
  VStack {
    Button("Click here") {
      self.counter += 1
    }
    if counter == 0 {
      Text("No clicks, weirdo!")
    } else if counter == 1 {
      Text("\(counter) click, weirdo!")
    } else {
      Text("\(counter) clicks, weirdo!")
    }
  }
}
```

Click here

1 click, weirdo!

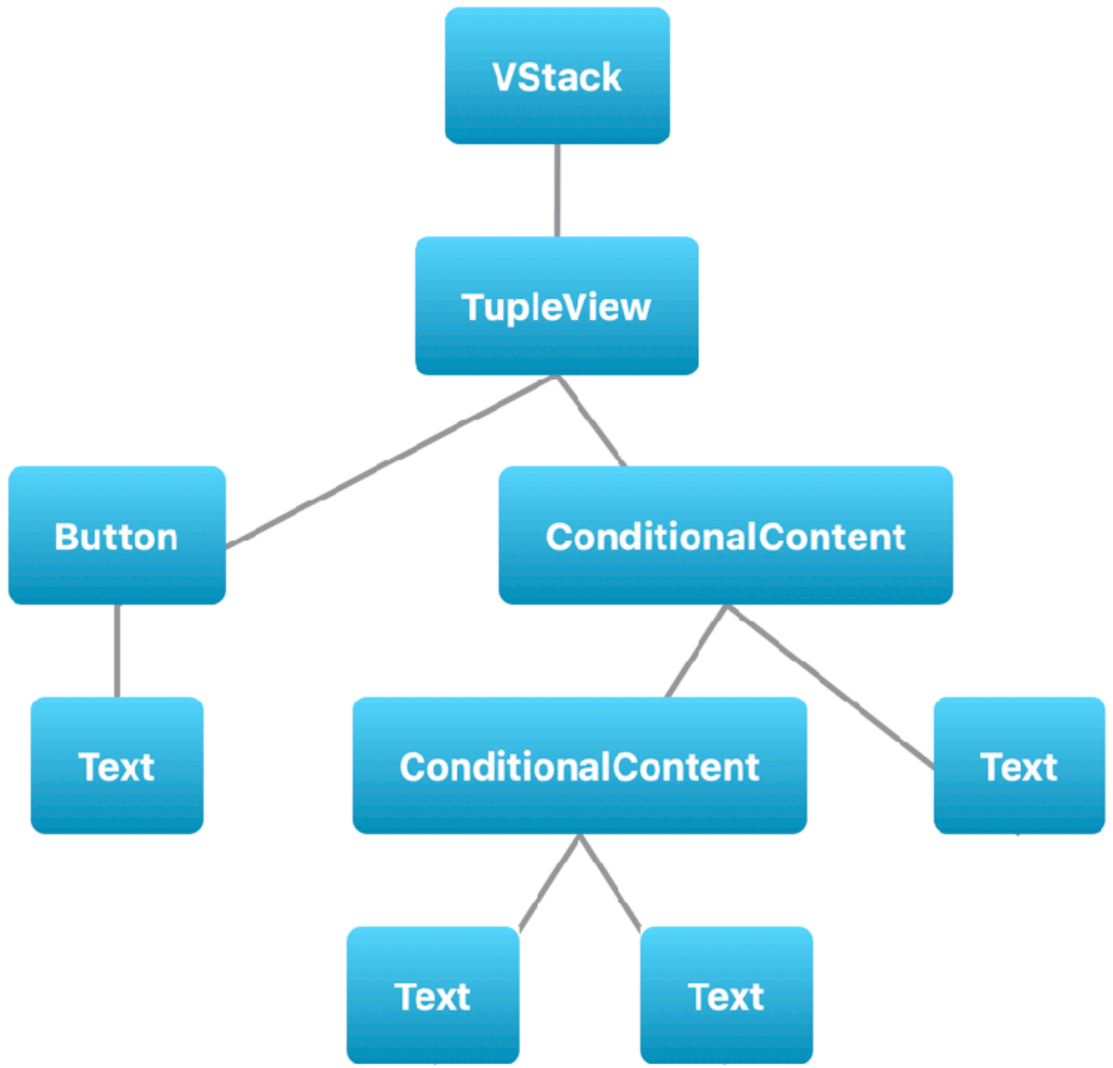



```
VStack<TupleView<(Button<Text>, _ConditionalContent<_ConditionalContent<Text, Text>, Text>)>>
```

```
@State var counter = 0
var body: some View {
  VStack {
    Button("Click here") {
      self.counter += 1
    }
    if counter == 0 {
      Text("No clicks, weirdo!")
    } else if counter == 1 {
      Text("\(counter) click, weirdo!")
    } else {
      Text("\(counter) clicks, weirdo!")
    }
  }
}
```

Click here

15 clicks, weirdo!




```
VStack<TupleView<(Button<Text>, _ConditionalContent<_ConditionalContent<Text, Text>, Text>)>>
```

Opaque types

```
var body: VStack<TupleView<(ModifiedContent<ModifiedContent<Text, _PaddingLayout>,
_BackgroundModifier<Color>>, ModifiedContent<ModifiedContent<Text, _BackgroundModifier<Color>>,
_PaddingLayout>)>> {
    ...
}
```

Opaque types

```
var body: some View {
    ...
}
```

Opaque types

```swift
var dishOkay: some View {
    if thing == 0 {
        return Text("This is")
    } else {
        return Text("fine 🔥")
    }
}

var dishBzzzt: some View {
    if thing == 0 {
        return Text("This isn't")
    } else {
        return Circle()
    }
}

var dish: some View {
    HStack {
        Text("Ohai")
        if self.thing == 0 {
            Text("This is")
        } else {
            Circle()
        }
    }
}
```

Function declares an opaque return type, but the return statements in its body do not have matching underlying types