Understandable flex

Product Engineer 소기현

담고 있는 내용

- 애플 접근성 소개
- SwiftUI Accessibility API
- Accessibility Tree
- 플렉스의 접근성 사례 조사
- 우리의 전략과 앞으로의 계획

애플 접근성 소개

- Make your app usable by **all** of your customers
- Apple's built in accessibility features
  - VoiceOver, Full Keyboard Access ...
- UI
  - Visual User Interface (우리가 일반적으로 생각하는 UI)
  - Accesibility User Interface (접근성도 UI다. 즉, 사용성을 고려한 UI 설계 필요)

애플 접근성 소개

접근성 요소

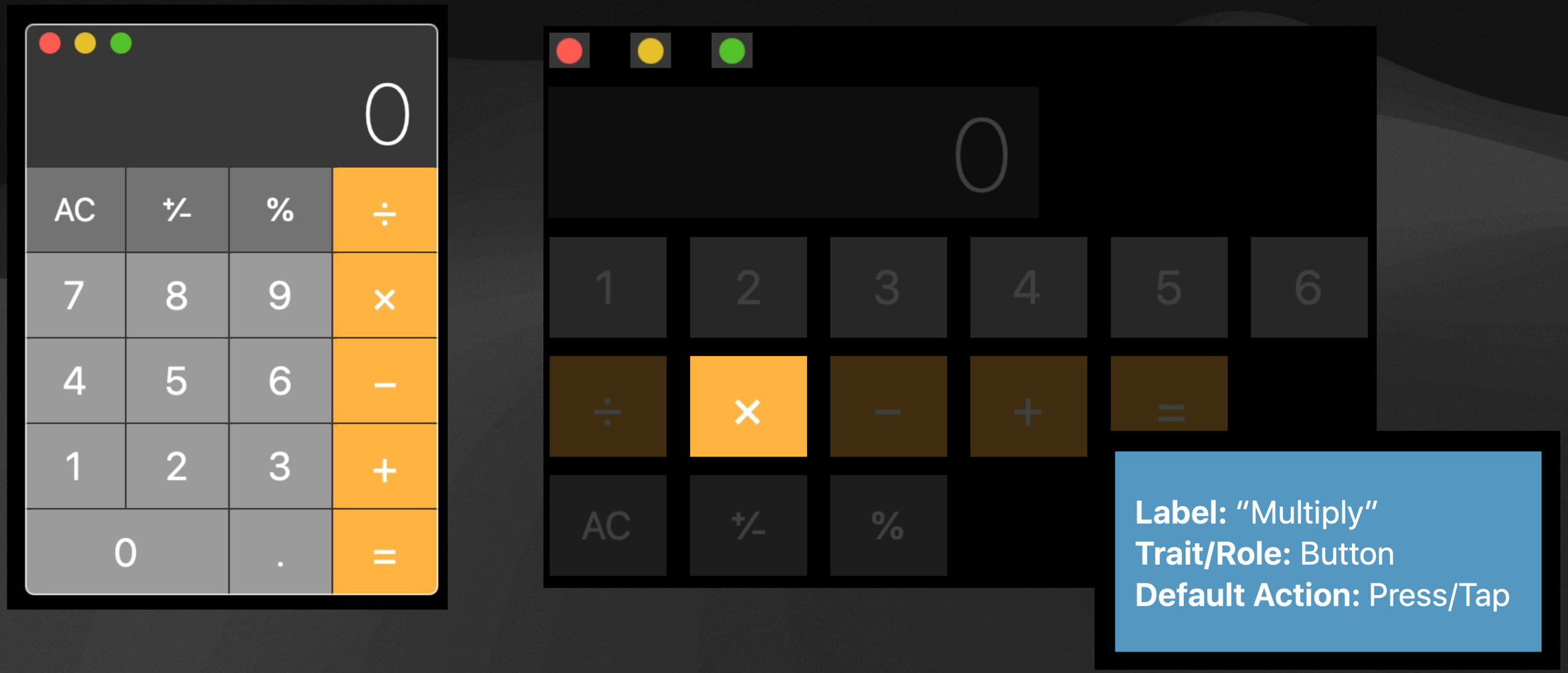
Visual User Interface가 View로 구성되는 것처럼

Accessibility User Interface는 Accessibility Element로 구성되어 있으며.

Label, Value, Trait, Role, Action 등 세부 항목을 포함하고 있습니다.

애플 접근성 소개

접근성 요소 예시



애플 접근성 소개

Automatic Accessibility

SwiftUI로 개발을 하면 상당 부분을 **자동으로** 접근성 요소를 생성해준다.

```
// SwiftUI makes accessibility elements
automatically for you

var body: some View {
    VStack {
        Text("Top Text")
        Button(action: { print("Pressed!") }) {
            Text("Middle Button")
        }
        Text("Bottom Text")
    }
}
```

Label: “Top Text”
Trait/Role: Static Text

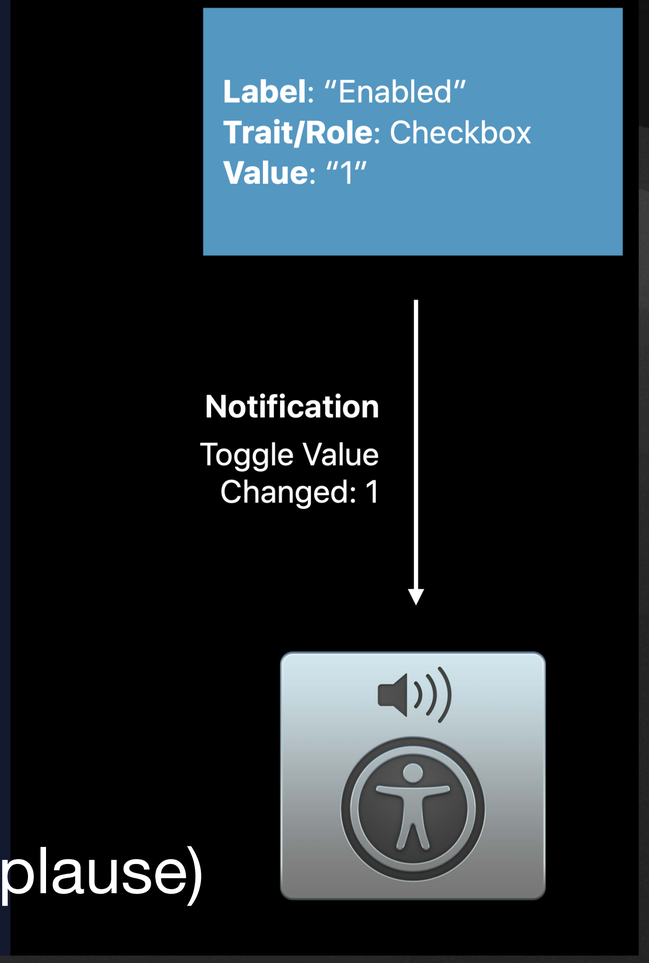
Label: “Middle Button”
Trait/Role: Button
Default Action: Press/Tap

Label: “Bottom Text”

애플 접근성 소개

Automatic Accessibility - Notifications

```
// SwiftUI automatically sends
accessibility notifications

@State private var enabled = false

var body: some View {
    VStack {
        Toggle(isOn: $enabled) {
            Text("Enabled")
        }
        Button(action: { enabled.toggle() }) {
            Text("Flip")
        }
    }
}
```



@State로 정의된 변수는 값이 변경되면 VoiceOver 발생한다. (applause)

애플 접근성 소개

Automatic Accessibility - Custom Control

CustomButton에서 기본 Button의 automatic accessibility를 제공하는 방법

```
var body: some View {
    Button(action: {}) { Text("Custom UI") }
        .buttonStyle(.init(CustomButtonStyle()))
}
```

Label: “Custom UI”
Trait/Role: Button
Default Action: Press/Tap

애플 접근성 소개

Automatic Accessibility - Image

```
Image("CheckmarkGlyph",
    label: Text("Signup Complete!"))
```

Trait/Role: Image
Label: “Signup Complete!”

```
Image("CheckmarkGlyph")
```

Trait/Role: Image
Label: None

```
Image(decorative: "CheckmarkGlyph")
```

애플 접근성 소개

Automatic Accessibility - 정리

- Standard controls accessible by default
- Accessibility Notifications are automatic (@State)
- Custom controls are automatically accessible (ButtonStyle)
- Accessible and decorative images
- Built-in, accessible labels for all controls (Picker...)

SwiftUI Accessibility API

자동 제공되는 Accessiblity 만으로는

Understandable / Interactable / Navigable 하지 않다면

추가적으로 Accessibility API를 사용할 수 있습니다

예를 들어 계산기 앱에서 곱하기 버튼을 VoiceOver에서 "X"로 읽어버린다면

의미가 잘 전달되지 않을것 입니다

SwiftUI Accessibility API

Label

accessibilityLabel(_:)

Adds a label to the view that describes its contents.

iOS 14.0+ | iPadOS 14.0+ | Mac Catalyst 14.0+ | macOS 11.0+ | tvOS 14.0+ | visionOS 1.0+ | watchOS 7.0+

```
nonisolated
func accessibilityLabel(_ label: Text) -> ModifiedContent<Self, AccessibilityAttachmentModifier>
```

Show all declarations

Discussion

Use this method to provide an accessibility label for a view that doesn't display text, like an icon. For example, you could use this method to label a button that plays music with the text "Play". Don't include text in the label that repeats information that users already have. For example, don't use the label "Play button" because a button already has a trait that identifies it as a button.

SwiftUI Accessibility API

Trait

accessibilityAddTraits(_:)

Adds the given traits to the view.

iOS 14.0+ | iPadOS 14.0+ | Mac Catalyst 14.0+ | macOS 11.0+ | tvOS 14.0+ | visionOS 1.0+ | watchOS 7.0+

```
nonisolated
func accessibilityAddTraits(_ traits: AccessibilityTraits) -> Modified
Content<Self, AccessibilityAttachmentModifier>
```

SwiftUI Accessibility API

Trait

```
static let allowsDirectInteraction: AccessibilityTraits
```
The accessibility element allows direct touch interaction for VoiceOver users.

```
static let causesPageTurn: AccessibilityTraits
```
The accessibility element causes an automatic page turn when VoiceOver finishes reading the text within it.

```
static let isButton: AccessibilityTraits
```
The accessibility element is a button.

```
static let isHeader: AccessibilityTraits
```
The accessibility element is a header that divides content into sections, like the title of a navigation bar.

```
static let isImage: AccessibilityTraits
```
The accessibility element is an image.

```
static let isKeyboardKey: AccessibilityTraits
```
The accessibility element behaves as a keyboard key.

```
static let isLink: AccessibilityTraits
```
The accessibility element is a link.

```
static let isModal: AccessibilityTraits
```
The accessibility element is modal.

```
static let isSearchField: AccessibilityTraits
```
The accessibility element is a search field.

```
static let isSelected: AccessibilityTraits
```
The accessibility element is currently selected.

```
static let isStaticText: AccessibilityTraits
```
The accessibility element is a static text that cannot be modified by the user.

```
static let isSummaryElement: AccessibilityTraits
```
The accessibility element provides summary information when the application starts.

```
static let isToggle: AccessibilityTraits
```
The accessibility element is a toggle.

```
static let playsSound: AccessibilityTraits
```
The accessibility element plays its own sound when activated.

```
static let startsMediaSession: AccessibilityTraits
```
The accessibility element starts a media session when it is activated.

```
static let updatesFrequently: AccessibilityTraits
```
The accessibility element frequently updates its label or value.

SwiftUI Accessibility API

Value

accessibilityValue(_:)

Adds a textual description of the value that the view contains.

iOS 14.0+ | iPadOS 14.0+ | Mac Catalyst 14.0+ | macOS 11.0+ | tvOS 14.0+ | visionOS 1.0+ | watchOS 7.0+

```
nonisolated
func accessibilityValue(_ valueDescription: Text) -> Modified
Content<Self, AccessibilityAttachmentModifier>
```

Show all declarations ⊕

Discussion

Use this method to describe the value represented by a view, but only if that's different than the view's label. For example, for a slider that you label as "Volume" using accessibilityLabel(), you can provide the current volume setting, like "60%", using accessibilityValue().

SwiftUI Accessibility API

Action

accessibilityAction(named:_:)

Adds an accessibility action to the view. Actions allow assistive technologies, such as the VoiceOver, to interact with the view by invoking the action.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | tvOS 13.0+ | visionOS 1.0+ | watchOS 6.0+

```
nonisolated
func accessibilityAction(
    named name: Text,
    _ handler: @escaping () -> Void
) -> ModifiedContent<Self, AccessibilityAttachmentModifier>
```

Show all declarations ⊕

Discussion

For example, this is how a custom action to compose a new email could be added to a view.

```
var body: some View {
    ContentView()
        .accessibilityAction(named: Text("New Message")) {
            // Handle action
        }
}
```

SwiftUI Accessibility API

Understandable

- Do the displayed strings provide enough information?

Interactable

- Will a custom action simplify the interaction?

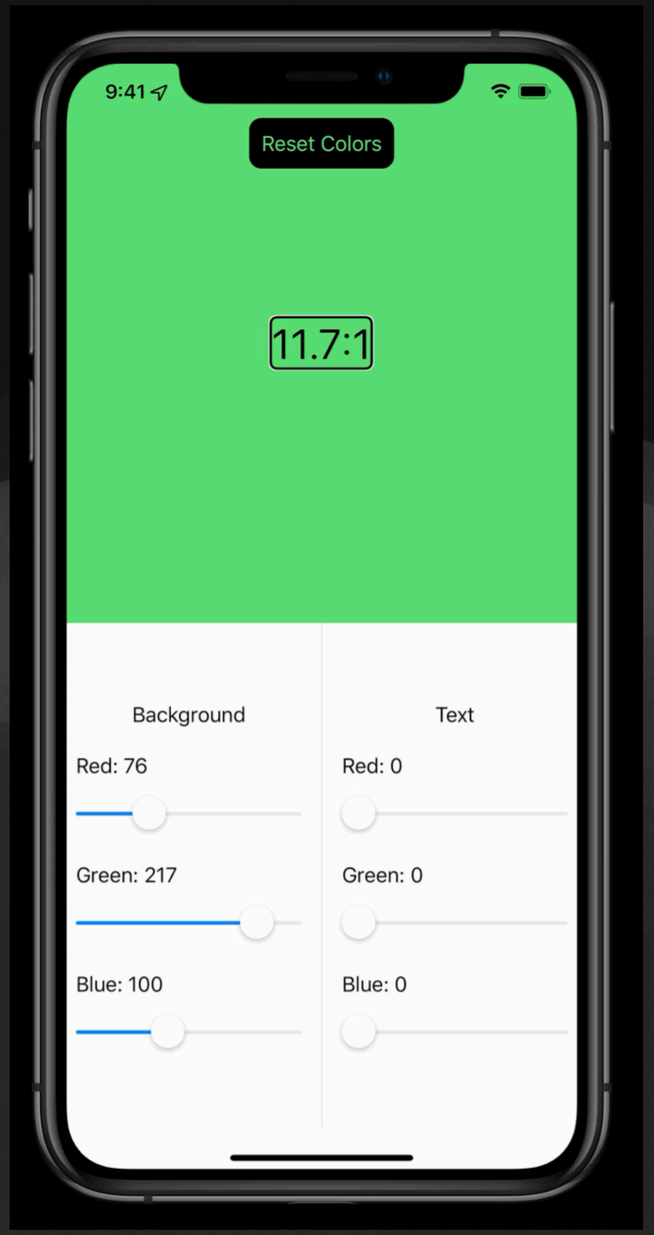
Navigable

- Can you speed up navigation?

SwiftUI Accessibility API

Case Study

- 배경을 더블탭 하면 배경색과 글자색이 switch
- 배경색과 글자색의 명암비가 글자로 표현
- 배경색과 글자색을 RGB 슬라이더를 사용하여 조정



SwiftUI Accessibility API

Case Study

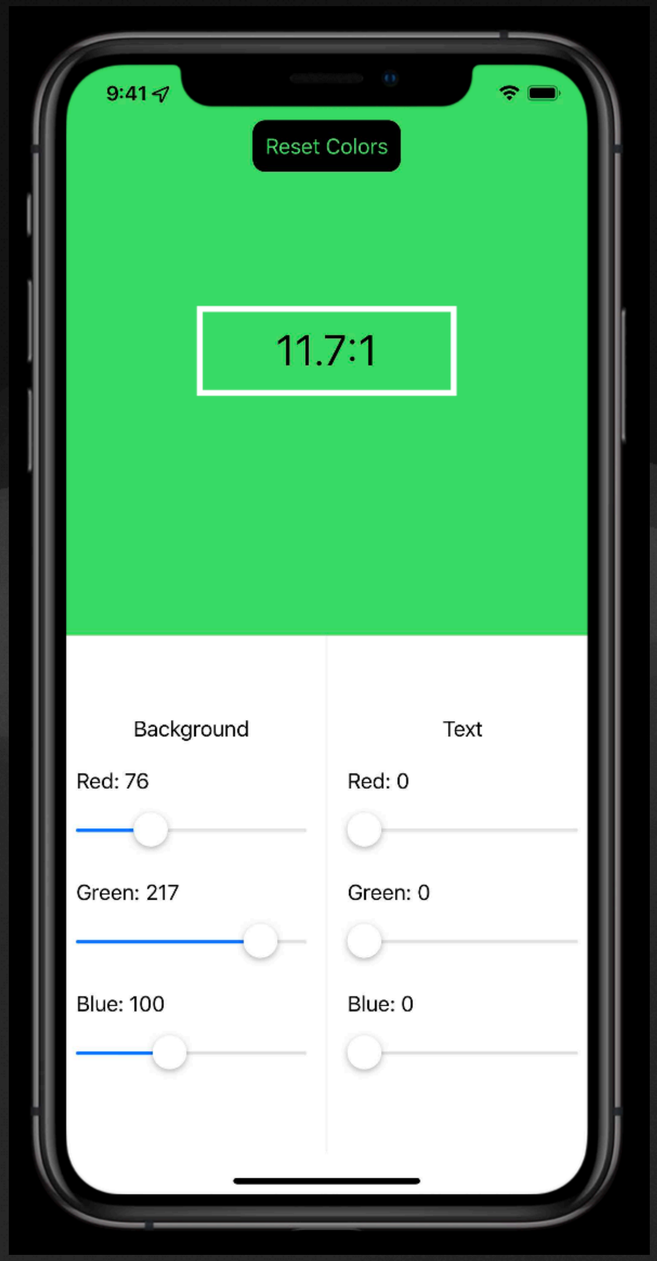
🔊 "11.7, 1"

- Not understandable

🔊 "Contrast Ratio, 11.7 to 1"

- Understandable 👍

```
ContrastRatioView()
.accessibility(label: Text("Contrast Ratio"))
.accessibility(value: Text("\(ratio) to 1"))
```

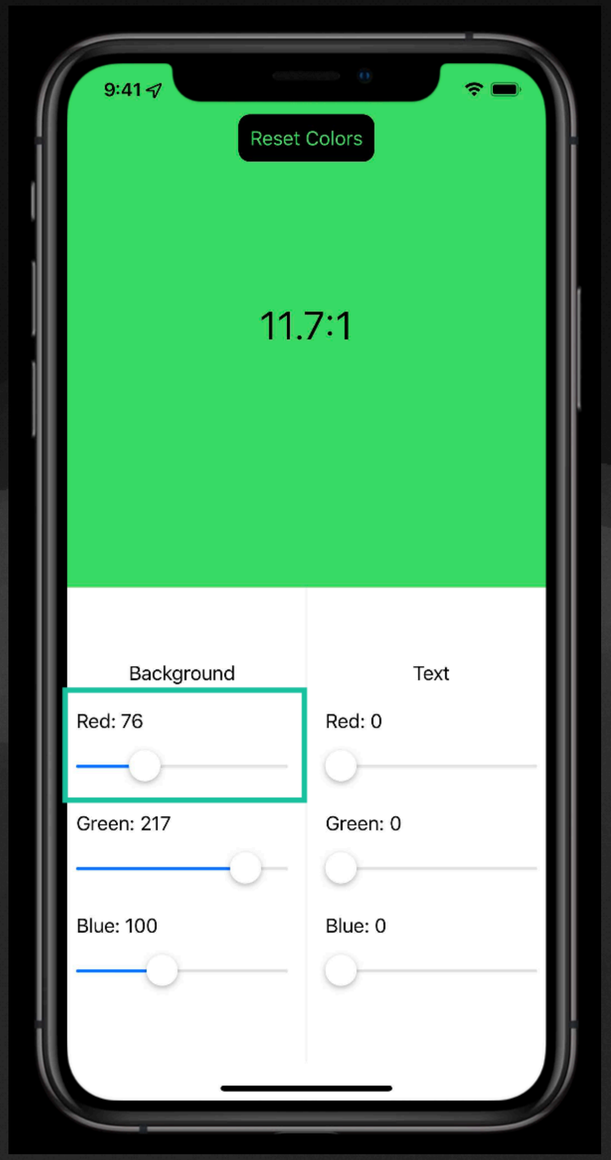


SwiftUI Accessibility API

Case Study

- 🔊 "27%"
  - Not understandable
- 🔊 "Red,76"
  - Understandable 👍

```
VStack(alignment: .leading) {
    Text(verbatim: String(format: "Red: %.0f", red))
    .accessibility(visibility: .hidden)

    Slider(value: $red, from: 0, through: 255.0)
    .accessibility(label: Text("Red"))
    .accessibility(value:
        Text(verbatim: String(format: "%.0f", red)))
}
```

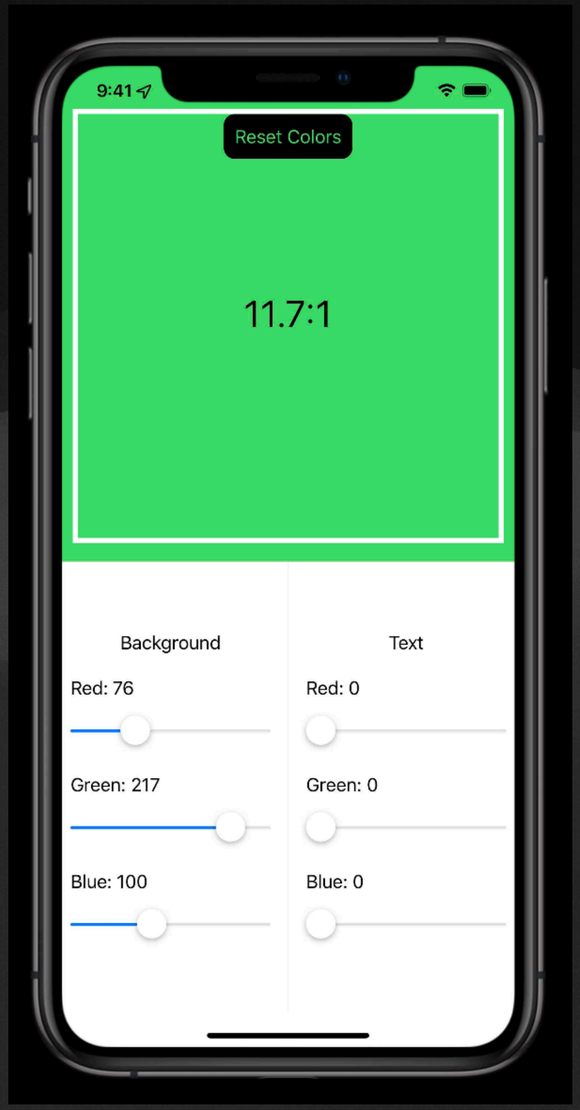


SwiftUI Accessibility API

Case Study

- 배경을 더블탭 하면 글자색과 배경색이 switch
- VoiceOver가 켜져있는 경우 더블탭 제스쳐를 사용할 수 없다 not Interactable 👎
- custom action을 add 해준다
  - focus 되었을 때 custom action이 있음을 알려주고, 스와이프 다운 or 업으로 실행할 수 있게 된다.

```
ContrastRatioView()
...
.accessibilityAction(named: Text(verbatim: "Swap Colors")) {
    /* swap the colors */
}
```



SwiftUI Accessibility API

Case Study

이 화면의 주요 요소를 세 부분으로 나눠 보면

- preview section
- background color control section
- text color control section

마지막 요소 까지 가려면 거쳐야 하는 요소가 너무 많다

not navigable 👎

SwiftUI Accessibility API

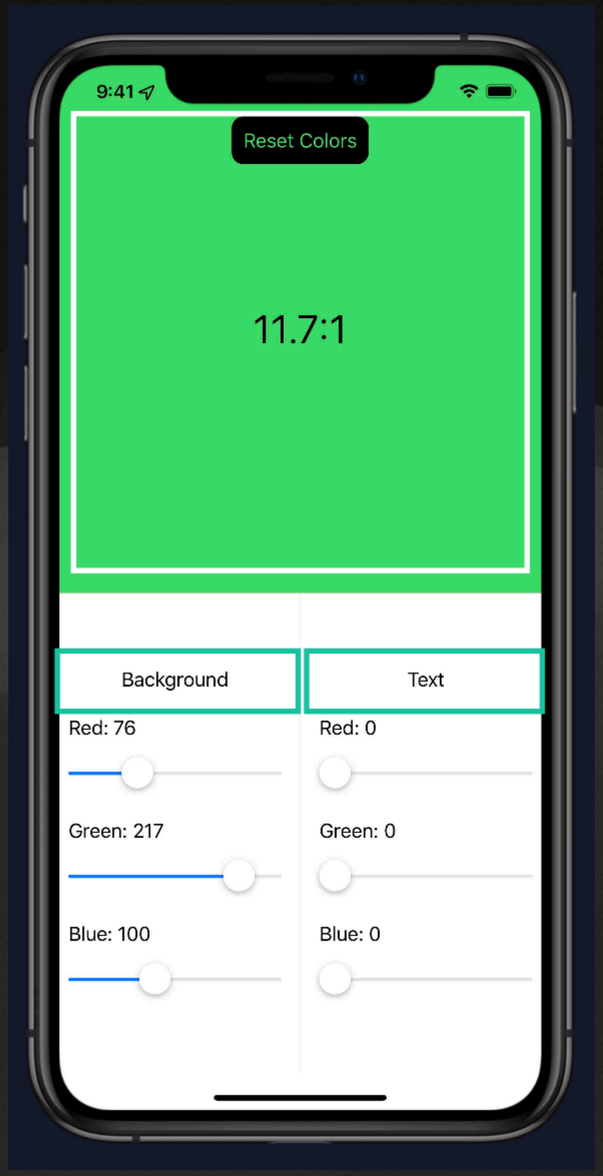
Case Study

Section을 구분할 수 있도록 isHeader trait를 추가해 주자

-> 헤더가 지정되면 rotor를 켜고 다음 header로 바로 이동 할 수 있다

-> 👍 navigable

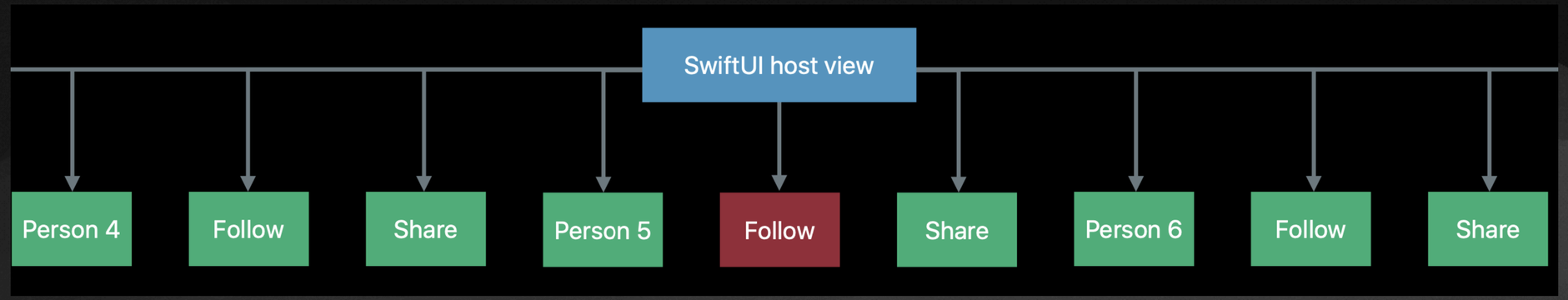
```
ContrastRatioView()
...
.accessibility(addTraits: .isHeader)

Text("Background")
.accessibility(addTraits: .isHeader)

Text("Text")
.accessibility(addTraits: .isHeader)
```



Accessibility Tree

- View 구조가 복잡해서 세부 요소가 너무 많으면?
  - 중복되는 정보가 많아서 not understandable
  - 요소가 너무 많아서 not navigable

Accessibility Tree



- 하나의 요소로 합치거나
- 특정 요소만 남기거나
- 통합된 하나의 요소로 만들자

Accessibility Tree

accessibilityElement(children:)

accessibilityElement(children:)

Creates a new accessibility element, or modifies the AccessibilityChildBehavior of the existing accessibility element.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | tvOS 13.0+ | visionOS 1.0+ | watchOS 6.0+

```
nonisolated
func accessibilityElement(children: AccessibilityChildBehavior = .ignore) -> some View
```

Accessibility Tree

accessibilityElement(children:)

- ignore

  하위 뷰의 접근성 요소를 모두 무시(hidden)하고 새로운 접근성 요소를 만든다.

ignore

Any child accessibility elements become hidden.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | tvOS 13.0+ | visionOS 1.0+ | watchOS 6.0+

```
static let ignore: AccessibilityChildBehavior
```

Discussion

Use this behavior when you want a view represented by a single accessibility element. The new accessibility element has no initial properties. So you will need to use other accessibility modifiers, such as accessibilityLabel(:), to begin making it accessible.

```
var body: some View {
    VStack {
        Button("Previous Page", action: goBack)
        Text("\(pageNumber)")
        Button("Next Page", action: goForward)
    }
    .accessibilityElement(children: .ignore)
    .accessibilityValue("Page \(pageNumber) of \(pages.count)")
    .accessibilityAdjustableAction { action in
        if action == .increment {
            goForward()
        } else {
            goBack()
        }
    }
}
```

Before using the ignorebehavior, consider using the combine behavior.

> **Note**
>
> A new accessibility element is always created.

Accessibility Tree

accessibilityElement(children:)

- combine

  하위 뷰의 접근성 요소의 속성들을 합쳐서 새로운 접근성 요소로 만든다.

combine

Any child accessibility element's properties are merged into the new accessibility element.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | tvOS 13.0+ | visionOS 1.0+ | watchOS 6.0+

```
static let combine: AccessibilityChildBehavior
```

Discussion

Use this behavior when you want a view represented by a single accessibility element. The new accessibility element merges properties from all non-hidden children. Some properties may be transformed or ignored to achieve the ideal combined result. For example, not all of `AccessibilityTraits` are merged and a `default` action may become a named action (`init(named:)`).

```
struct UserCell: View {
    var user: User

    var body: some View {
        VStack {
            Image(user.image)
            Text(user.name)
            Button("Options", action: showOptions)
        }
        .accessibilityElement(children: .combine)
    }
}
```

Accessibility Tree

Ordering

- ZStack orders from back to front

accessibilitySortPriority(_:)

Sets the sort priority order for this view's accessibility element, relative to other elements at the same level.

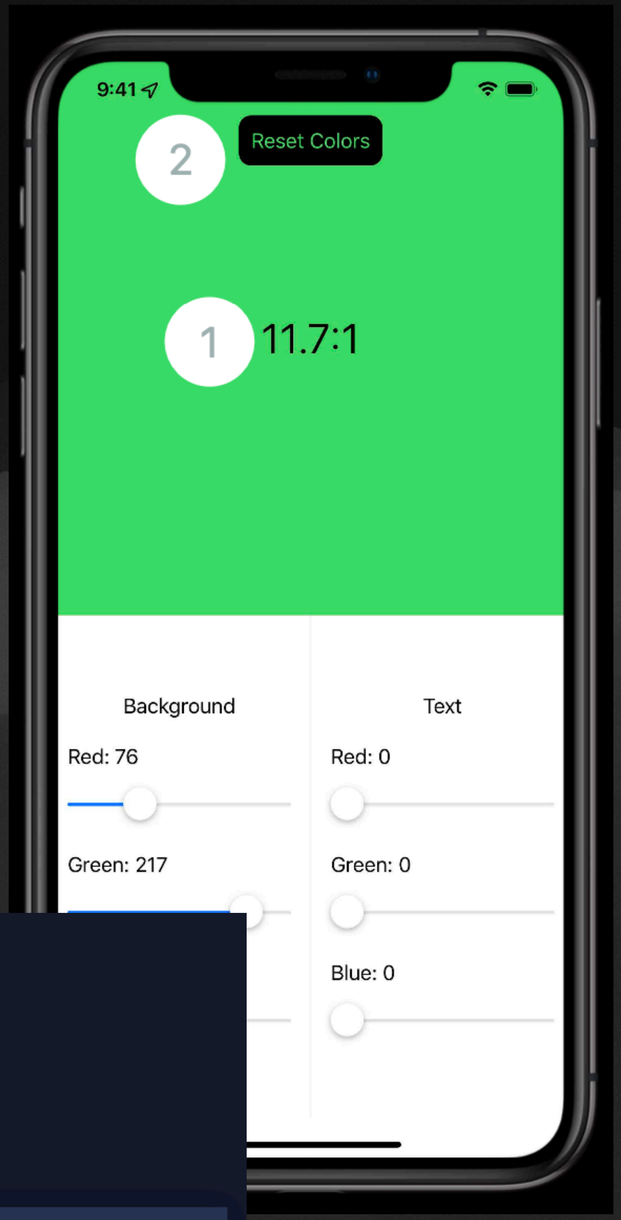
iOS 14.0+ | iPadOS 14.0+ | Mac Catalyst 14.0+ | macOS 11.0+ | tvOS 14.0+ | visionOS 1.0+ | watchOS 7.0+

```
nonisolated
func accessibilitySortPriority(_ sortPriority: Double) -> Modified
Content<Self, AccessibilityAttachmentModifier>
```

Discussion

Higher numbers are sorted first. The default sort priority is zero.

```
Button(action: {
    resetColors()
}) {
    Text("Reset Colors")
    .foregroundColor(backgroundStorage.color)
}
.accessibility(sortPriority: 1)
```



Summary

- 자동으로 최소한은 해준다
- 추가로 최적화 필요할 때 고려 할 점
  - understandable -> Label, Trait, Value..
  - interactable -> custom action...
  - navigable -> grouping...