Accessibility in SwiftUI

Building the next generation of apps for everyone

John Nefulda, Accessibility Engineer
Michael Gorbach, Accessibility Engineer
Introduction to Accessibility
Introduction to Accessibility

Automatic Accessibility with SwiftUI
Introduction to Accessibility

Automatic Accessibility with SwiftUI

SwiftUI Accessibility API
Introduction to Accessibility
Automatic Accessibility with SwiftUI
SwiftUI Accessibility API
Accessibility Tree
Intro to Accessibility
Make your app usable by all of your customers
Voice Control

VoiceOver
Word Prediction
AssistiveTouch
Reduce Transparency
On/Off Labels

Speak Screen
Mono Audio
Button Shapes
Dwell Support
Increase Contrast

Software TTY
Gliding Cursor Speed
Siri
Sticky Keys
Zoom

Switch Control
Larger Text
Audio Descriptions
Dictation
Slow Keys

Gestures
Braille Support
Cursor Color
Invert Colors
Grayscale

Vibrating Alerts
Closed Captions
Hover Text
Display Filters
Hearing Aids

Larger Cursor
Magnifier
Speak Screen
Reduce Transparency

Reduce Motion
Full Keyboard Access
Word Prediction
Sticky Keys

Siri
Braille Support
Software TTY

Reduce Motion
Full Keyboard Access
Word Prediction
Sticky Keys

Siri
Braille Support
Software TTY
Full Keyboard Access
Accessibility User Interface
Accessibility User Interface
Accessibility User Interface

0

AC  ÷  
0

7  8  9  x

4  5  6  -

1  2  3  +

0

.
Accessibility User Interface
Accessibility User Interface

Label: “Multiply”
Trait/Role: Button
Default Action: Press/Tap
Accessibility User Interface

Label: “Main Display”
Value: “0”
Trait/Role: Static Text
Default Action: None
Custom Action: “Clear”
Accessibility User Interface
Accessibility User Interface
Accessibility User Interface

Make sure your app’s Accessibility is:
Accessibility User Interface

Make sure your app’s Accessibility is:

1. Understandable: Labels
Accessibility User Interface

Make sure your app’s Accessibility is:

1. Understandable: Labels
2. Interactable: Actions
Accessibility User Interface

Make sure your app’s Accessibility is:

1. Understandable: Labels
2. Interactable: Actions
3. Navigable: Ordering and Grouping
## Accessibility User Interface

<table>
<thead>
<tr>
<th>Title</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessibility Lessons</td>
<td>WWDC 2019</td>
</tr>
<tr>
<td>Deliver an Exceptional Accessibility Experience</td>
<td>WWDC 2018</td>
</tr>
<tr>
<td>What's New in Accessibility</td>
<td>WWDC 2017</td>
</tr>
</tbody>
</table>
Automatic Accessibility with SwiftUI
// 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")
    }
}
// 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")
    }
}
// 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")
    }
}
// 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")
    }
}
// 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")
    }
}
// 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")
    }
}
// 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")
    }
}
// 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")
    }
}
Accessibility Notifications

Elements?
Accessibility Notifications

Elements:
1, 2, 3, 4, 5, ...

Accessibility Logo with a person and volume symbol.
## Accessibility Notifications

The main display value is 5.

| 5 | 1 2 3 | 4 5 6 | 7 8 9 | C +/- % ÷ × − + . | Elements: 1, 2, 3, 4, 5 ... |

The image shows a calculator with keys for basic arithmetic operations and a display showing the main value of 5. The accessibility notification icon indicates that elements 1, 2, 3, 4, 5, etc., are being displayed.
Accessibility Notifications
Accessibility Notifications

Main Display Value Changed: 10
// 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")
        }
    }
}
// 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")
        }
    }
}
// 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")
        }
    }
}
// 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")
        }
    }
}
// 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")
        }
    }
}
Accessible Custom Controls

Custom UI  Custom UI
With SwiftUI, you can easily customize button drawing

```swift
struct CustomButtonStyle : ButtonStyle {
    func body(configuration: Button<Label>, isPressed: Bool) -> some View {
        configuration.label
            .font(size: 18)
            .foregroundColor(isPressed ? .black : .white)
            .padding(8)
            .background(RoundedRectangle(cornerRadius: 5)
                            .fill(isPressed ? Color.red : Color.blue))
    }
}
```
struct CustomButtonStyle : ButtonStyle {

    func body(configuration: Button<Label>, isPressed: Bool) -> some View {

        configuration.label
            .font(size: 18)
            .foregroundColor(isPressed ? .black : .white)
            .padding(8)
            .background(
                RoundedRectangle(cornerRadius: 5)
                    .fill(isPressed ? Color.red : Color.blue)
            )
    }
}
// With SwiftUI, you can easily customize button drawing

```swift
struct CustomButtonStyle : ButtonStyle {
    func body(configuration: Button<Label>, isPressed: Bool) -> some View {
        configuration.label
            .font(size: 18)
            .foregroundColor(isPressed ? .black : .white)
            .padding(8)
            .background(RoundedRectangle(cornerRadius: 5)
                .fill(isPressed ? Color.red : Color.blue))
    }
}
```
// With SwiftUI, you can easily customize button drawing

struct CustomButtonStyle : ButtonStyle {
  func body(configuration: Button<Label>, isPressed: Bool) -> some View {
    configuration.label
      .font(size: 18)
      .foregroundColor(isPressed ? .black : .white)
      .padding(8)
      .background(
        RoundedRectangle(cornerRadius: 5)
        .fill(isPressed ? Color.red : Color.blue)
      )
  }
}
// Create a custom-styled button with SwiftUI

var body: some View {
    Button(action: {}) { Text("Custom UI") }
    .buttonStyle(.init(CustomButtonStyle()))
}
// Create a custom-styled button with SwiftUI

var body: some View {
    Button(action: {}) { Text("Custom UI") }
    .buttonStyle(.init(CustomButtonStyle()))
}
// Create a custom-styled button with SwiftUI

var body: some View {
    Button(action: {}) { Text("Custom UI") }
        .buttonStyle(.init(CustomButtonStyle()))
}
// Image accessibility in SwiftUI

struct SignupCompleteView : View {
    var body: some View {
        VStack {
            Image("CheckmarkGlyph")
        }
    }
}
// Image accessibility in SwiftUI

struct SignupCompleteView: View {
    var body: some View {
        VStack {
            Image("CheckmarkGlyph")
        }
    }
}
// Image accessibility in SwiftUI

struct SignupCompleteView: View {
    var body: some View {
        VStack {
            Image("CheckmarkGlyph")
        }
    }
}
// Image accessibility in SwiftUI

struct SignupCompleteView: View {
    var body: some View {
        VStack {
            Image("CheckmarkGlyph",
                label: Text("Signup Complete!"))
        }
    }
}
struct SignupCompleteView: View {
    var body: some View {
        VStack {
            Image("CheckmarkGlyph", label: Text("Signup Complete!"))
        }
    }
}
// Image accessibility in SwiftUI

struct SignupCompleteView : View {
    var body: some View {
        VStack {
            Image(decorative: "CheckmarkGlyph")
            Text("Signup Complete!")
            Text("Thank you for signing up!")
        }
    }
}
// Image accessibility in SwiftUI

struct SignupCompleteView : View {
    var body: some View {
        VStack {
            Image(decorative: "CheckmarkGlyph")
            Text("Signup Complete!")
            Text("Thank you for signing up!")
        }
    }
}
// Image accessibility in SwiftUI

struct SignupCompleteView: View {
    var body: some View {
        VStack {
            Image(decorative: "CheckmarkGlyph")
            Text("Signup Complete!")
            Text("Thank you for signing up!")
        }
    }
}
Label: None
Value: “Alex”
Trait/Role: PopUp Button
// Control labels for accessibility in SwiftUI

static let voices = [ "Alex", "Fred", "Victoria" ]

@State var selectedVoice = Self.voices.first!

var body: some View {
    Picker(selection: $selectedVoice,
           label: Text("System Voice")) {
        ForEach(Self.voices.identified(by: \.self)) {
            Text(verbatim: $0)
        }
    }
}
static let voices = [ "Alex", "Fred", "Victoria" ]

@State var selectedVoice = Self.voices.first!

var body: some View {
    Picker(selection: $selectedVoice, label: Text("System Voice")) {
        ForEach(Self.voices.identified(by:\.self)) {
            Text(verbatim: $0)
        }
    }
}
// Control labels for accessibility in SwiftUI

static let voices = [ "Alex", "Fred", "Victoria" ]

@State var selectedVoice = Self.voices.first!

var body: some View {
    Picker(selection: $selectedVoice,
        label: Text("System Voice")) {
            ForEach(Self.voices.identified(by: \.self)) {
                Text(verbatim: $0)
            }
        }
}
// Control labels for accessibility in SwiftUI

static let voices = [ "Alex", "Fred", "Victoria" ]

@State var selectedVoice = Self.voices.first!

var body: some View {
    Picker(selection: $selectedVoice,
           label: Text("System Voice")) {
        ForEach(Self.voices.identified(by: \.self)) {
            Text(verbatim: $0)
        }
    }
}
Label: “System Voice”
Trait/Role: Static Text

Label: Linked
Value: “Alex”
Trait/Role: PopUp Button

System Voice: Alex
Alex, System Voice, pop up button

System Voice: Alex
Automatic Accessibility with SwiftUI

- Standard controls accessible by default
- Accessibility Notifications are automatic
- Custom controls are automatically accessible
- Accessible and decorative images
- Built-in, accessible labels for all controls
SwiftUI Accessibility API

John Nefulda, Accessibility Engineer
SwiftUI Accessibility API

CalculatorButton(.multiply)
SwiftUI Accessibility API

```
CalculatorButton(.multiply)
.accessibility(label: Text("Multiply"))
```
SwiftUI Accessibility API

```swift
CalculatorButton(.multiply).accessibility(label: Text("Multiply"))
```
SwiftUI Accessibility API

```swift
CalculatorButton(.multiply)
.accessibility(label: Text("Multiply"))
.accessibility(addTraits: selected ? .isSelected : [])
```
SwiftUI Accessibility API

ResultView()
SwiftUI Accessibility API

ResultView()

.accessibility(label: Text("Result"))
.accessibility(value: Text("\(value)"))
SwiftUI Accessibility API

```swift
ResultView()
    .accessibility(label: Text("Result"))
    .accessibility(value: Text("\(value)"))
    .accessibilityAction(named: Text("Clear")) {
        clear()
    }
```
Accessibility API
Accessibility API

Understandable
Accessibility API

Understandable

Interactable
Accessibility API

Understandable

Interactable

Navigable
Do the displayed strings provide enough information?
Will a custom action simplify the interaction?
Can you speed up navigation?
Accessibility API

Understandable
• Do the displayed strings provide enough information?

Interactable
• Will a custom action simplify the interaction?

Navigable
• Can you speed up navigation?
Understandable
Do the displayed strings provide enough information?
Symbols should be spoken correctly

Understandable
Do the displayed strings provide enough information?
Understandable
Do the displayed strings provide enough information?

Symbols should be spoken correctly
• “11.7, 1” → “11.7 to 1”
Symbols should be spoken correctly
• “11.7, 1” → “11.7 to 1”
Context can be spoken if appropriate
Symbols should be spoken correctly
• “11.7, 1” → “11.7 to 1”

Context can be spoken if appropriate
• “11.7, 1” → “Contrast Ratio, 11.7 to 1”
// Setting a label and a value

ContrastRatioView()
.accessibility(label: Text("Contrast Ratio"))
.accessibility(value: Text("\(ratio\) to 1"))
Understandable
Do the displayed strings provide enough information?
Understandable
Do the displayed strings provide enough information?

Color slider doesn’t convey the right value
Understandable

Do the displayed strings provide enough information?

- "27%" → "Red, 76"
// Setting a label and a value

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

    Slider(value: $red, from: 0, through: 255.0)
}

// Setting a label and a value

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)))
}

Interactable
Does a custom action simplify the interaction?

• “Swap Colors” custom action
// Creating custom actions

ContrastRatioView()
...

.accessibilityAction(named: Text(verbatim: "Swap Colors")) {
    /* swap the colors */
}
Navigable
Can you speed up navigation?
Navigable

Can you speed up navigation?

This app can be divided into three main spaces
Navigable
Can you speed up navigation?

This app can be divided into three main spaces
• Contrast ratio
This app can be divided into three main spaces:

- Contrast ratio
- Background color

Navigable
Can you speed up navigation?
Navigable
Can you speed up navigation?

This app can be divided into three main spaces
• Contrast ratio
• Background color
• Text color
// Setting a header trait

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

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

Text("Text")
.accessibility(addTraits: .isHeader)
Accessibility Tree
// Accessibility Tree

struct TableCell : View {
    let person: Person
    var body: some View {
        HStack {
            Text(verbatim: person.name)
            Spacer()
            Button(action: { /* Follow */ }) {
                Text("Follow")
            }
            Button(action: { /* Share */ }) {
                Text("Share")
            }
        }
    }
}
struct TableCell: View {
    let person: Person
    var body: some View {
        HStack {
            Text(verbatim: person.name)
            Spacer()
            Button(action: { /* Follow */ }) {
                Text("Follow")
            }
            Button(action: { /* Share */ }) {
                Text("Share")
            }
        }
    }
}
Accessibility Tree

Accessibility

SwiftUI host view

Text

Button
Accessibility Tree

- Accessibility
  - UI/AppKit Accessibility Tree
    - SwiftUI host view
      - UI/NSViewRepresentable
      - Text
      - Button
      - Button
Accessibility Tree

SwiftUI host view

- Text
- Button
- Button
Accessibility Tree

- Person 4
- Follow
- Share
- Person 5
- Follow
- Share
- Person 6
- Follow
- Share
struct TableCell: View {
    let person: Person
    var body: some View {
        HStack {
            Text(verbatim: person.name)
            Spacer()
            Button(action: { /* Follow */ }) {
                Text("Follow")
            }
            Button(action: { /* Share */ }) {
                Text("Share")
            }
        }
    }
}
struct TableCell : View {
    let person: Person
    var body: some View {
        HStack {
            Text(verbatim: person.name)
            Spacer()
            Button(action: { /* Follow */ }) {
                Text("Follow")
            }
            Button(action: { /* Share */ }) {
                Text("Share")
            }
        }
        .accessibilityElement(children: .combine)
    }
}
structTableCell: View {
    let person: Person
    var body: some View {
        HStack {
            Text(verbatim: person.name)
            Spacer()
            Button(action: { /* Follow */ }) {
                Text("Follow")
            }
            Button(action: { /* Share */ }) {
                Text("Share")
            }
        }
        .accessibilityElement(children: .combine)
    }
}
Accessibility Tree

- Person 5
- Person 6
- Person 7
- Person 8
- Person 9
- Person 10
- Person 11
- Person 12
- Person 13

SwiftUI host view
Accessibility Tree
Ordering

Accessibility Tree
Ordering
Accessibility Tree

Ordering
Accessibility Tree
Ordering
Accessibility Tree

ZStack orders from back to front
Accessibility Tree

Ordering

- ZStack orders from back to front
- Customize ordering using sortPriority
Button(action: {
    resetColors()
}) {
    Text("Reset Colors")
    .foregroundColor(backgroundStorage.color)
}

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

Understandable
• Provide context by adding labels, values, and hints
Accessibility API Summary

- Understandable
  • Provide context by adding labels, values, and hints

- Interactable
  • Simplify by adding custom actions
Accessibility API Summary

Understandable
- Provide context by adding labels, values, and hints

Interactable
- Simplify by adding custom actions

Navigable
- Speed up elements by grouping
Accessibility API Summary

Understandable
- Provide context by adding labels, values, and hints

Interactable
- Simplify by adding custom actions

Navigable
- Speed up elements by grouping
Summary
Evaluating Accessibility

Use your app with
• VoiceOver
• Full Keyboard Access
• Voice Control
Accessibility Inspector

Explore, test, and debug your app’s Accessibility

Accessibility Lessons: Inspector  
WWDC 2019

Auditing Your Apps for Accessibility  
WWDC 2016
Summary
Summary

Automatically accessible SwiftUI apps
Summary

Automatically accessible SwiftUI apps
Understandable, interactable, and navigable
Summary

Automatically accessible SwiftUI apps

Understandable, interactable, and navigable

Powerful SwiftUI Accessibility API
Summary

Automatically accessible SwiftUI apps

Understandable, interactable, and navigable

Powerful SwiftUI Accessibility API

Unified across all platforms
More Information

developer.apple.com/wwdc19/238

Accessibility Lab

Friday, 11:00
Apple WWDC19