Accessibility

Which of these statements are true?

Click on any answer to show more detail.

1. Correct

    Option 1: Accessibility hints are read first, followed the accessibility label.

    Option 2: Accessibility labels are read first, followed the accessibility hint.

    You selected Option 2.

2. Correct

    Option 1: Accessibility labels should usually be short.

    Option 2: SF Symbols don't have any default accessibility labels.

    You selected Option 1.

3. Correct

    Option 1: It takes a lot of work to add accessibility to our apps.

    Option 2: SwiftUI provides a lot of accessibility support automatically.

    You selected Option 2.

4. Correct

    Option 1: Without a custom label, SwiftUI will read an image's name as its VoiceOver description.

    Option 2: You should test the way VoiceOver reads labels using the simulator.

    You selected Option 1.

5. Correct

Option 1: Image views automatically have the `isImage` accessibility trait.

Option 2: By default, SwiftUI uses a scale of 1 to 10 for the accessibility values of sliders.

You selected Option 1.

6. Correct

Option 1: You can add an accessibility label or an accessibility hint, but not both.

Option 2: We can use conditions inside accessibility labels to decide which text to use.

You selected Option 2.

7. Correct

Option 1: Decorative images are images that are merely there to make the UI look nicer.

Option 2: Each view can have either one or zero accessibility traits.

You selected Option 1.

8. Correct

Option 1: Three text fields in the same `VStack` are considered to be separate elements if they aren't specifically combined.

Option 2: If we use `.accessibilityElement(children: .ignore)` the entire view becomes invisible to VoiceOver.

You selected Option 1.

9. Correct

Option 1: It is possible to hide views from the accessibility system.

Option 2: It is recommended to hide views from the accessibility system.

You selected Option 1.

10. Correct

Option 1: We can control how SwiftUI reads out the value of UI controls such as steppers and sliders.

Option 2: As it's a container, `VStack` can't have accessibility data.

You selected Option 1.

11. Correct

Option 1: SwiftUI allows us to group views into a single accessibility element.

Option 2: Views with an `onTapGesture()` modifier automatically have the `isButton` trait.

You selected Option 1.

12. Correct

Option 1: Accessibility labels must always be a single hard-coded string.

Option 2: We should aim to make all our apps accessible to everyone.

You selected Option 2.

Total score: 12/12

Back to Review menu

Note: if you're following the 100 Days of Swift or the 100 Days of SwiftUI, just close this window and return to where you were.

@twostraws

paul@hackingwithswift.com

About Glossary Privacy Policy Refund Policy Update Policy

Hacking with Swift is sponsored by Gold Supporters on Patreon – click here to find out more

Thanks for your support, Dean Moore!