

In one line: Accessibility: VoiceOver walks a tree of *elements* and speaks each one's **label, value, traits, hint**; text must scale with **Dynamic Type**; honour user settings (Reduce Motion, contrast). **Localization:** every user-visible string is a *whole sentence* in a **String Catalog**, arguments are **positional**, plurals follow **CLDR** rules, layout is **leading/trailing**, and numbers/dates go through **formatters**.

How it works — accessibility

- **Label** = what it is (“Volume” — never “Volume button”, the trait says button). **Value** = current state (“70%”). **Traits** = role/state (.button .header .selected .adjustable .notEnabled). **Hint** = result of acting, spoken last, user can turn hints off.
- `isAccessibilityElement = true` makes a view a leaf; on a container it *hides the children*. Order children with `accessibilityElements = [...]`. Decorative: `.accessibilityHidden(true)`.
- **Grouping** (SwiftUI): `.accessibilityElement(children:)` .combine (one swipe, merged) .ignore (you label it) .contain (keep children, group them).
- `.adjustable` needs `accessibilityIncrement()` / `Decrement()` (SwiftUI `.accessibilityAdjustableAction`).
- **Custom actions** (`UIAccessibilityCustomAction`, `.accessibilityAction(named:)`) expose swipe/long-press actions via the **Actions rotor**; custom **rotors** (`.accessibilityRotor`) jump between headings/links. Announce: `UIAccessibility.post(notification: .announcement, ...)`.
- `accessibilityIdentifier` is for **UI tests**: stable, not localized, never spoken.
- **Dynamic Type**: `UIFont.preferredFont(forTextStyle:)` or `UIFontMetrics(forTextStyle:).scaledFont(for:)` + `adjustsFontForContentSizeCategory = true` (default false) + `numberOfLines = 0`. SwiftUI text styles scale by themselves; `@ScaledMetric`. AX sizes → stack vertically.
- **Settings**: Reduce Motion (`\.accessibilityReduceMotion`: swap zoom for fades), Reduce Transparency, Increase Contrast, Bold Text. Contrast $\geq 4.5:1$ (3:1 large text); never colour alone.
- Targets $\geq 44 \times 44$ pt. Audit: **Accessibility Inspector** + `XCUUIApplication().performAccessibilityAudit()` (iOS 17).

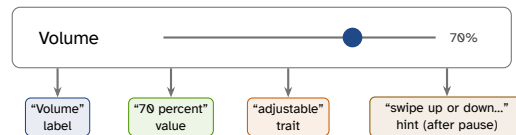
How it works — localization

- **String Catalog** .xcstrings (Xcode 15, JSON): keys extracted at build, per-language state, plural + device variants inline. Replaces `.strings + .stringsdict`.
- Code: `String(localized: "key", defaultValue:, comment:)` (iOS 15), `LocalizedStringResource`; SwiftUI `Text("literal")` is a `LocalizedStringKey`. Packages: bundle: `.module`. Always a comment:.
- **Plurals**: “Vary by plural” — categories zero one two few many other per language (Polish: one / few / many). Never `count == 1 ? :`.
- **RTL**: leading/trailing + `.natural` alignment mirror; left/right never do. Opt out per view with `semanticContentAttribute(.playback for media controls)`; SwiftUI `\.layoutDirection`.
- **Formatters**: `date.formatted(...)`, `.currency(code:)`, `Measurement(km↔mi)`. `Locale.current = format` region, not UI language.
- **Test**: pseudolanguages (Accented, Bounded, Double-Length, RTL); vendors get *Export Localizations* (`.xcloc/XLIFF`).

Example

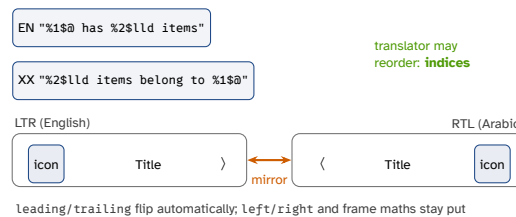
```
HStack {
    // a volume row
    Image(systemName: "speaker.wave.2").accessibilityHidden(true)
    Text("Volume") // LocalizedStringKey
    Slider(value: $level)
}
.accessibilityElement(children: .ignore)
.accessibilityLabel("Volume") // no "slider" - trait adds it
.accessibilityValue(Text(level, format: .percent))
.accessibilityAdjustableAction { // swipe up / down
    level += $0 == .increment ? 0.1 : -0.1
}
.accessibilityIdentifier("settings.volume") // tests only
// catalog key "%lld files deleted", varied by plural
let msg = String(localized: "%(count) files deleted",
    comment: "Toast after bulk delete")
```

Picture — what VoiceOver says



identifier "settings.volume" — never spoken, XCTest only

Picture — positional arguments & mirroring



Interview traps

- ! “Play button” as label → “Play button, button”.
- ! `isAccessibilityElement` on a container hides its children.
- ! `preferredFont` without the `adjustsFont...` flag → never re-scales.
- ! UI tests matching localized labels — use `accessibilityIdentifier`.
- ! `"\n" + "items"` — breaks word order *and* plurals.
- ! `Text(aStringVar)` is *not* localized — only literals / keys.

Remember

“L-V-T-H” — Label, Value, Trait, Hint (the order VoiceOver speaks). **Localize sentences, not words; lead, don’t left.**

Likely questions

1. Label vs identifier? — spoken + localized vs silent test hook.
2. Scale a custom font? — `UIFontMetrics.scaledFont(for:)`.
3. Why `%1$@`? — lets translators reorder arguments.
4. Catch truncation / hard-coded text early? — pseudolanguages.
5. Swipe-to-delete for VoiceOver? — accessibility custom action.