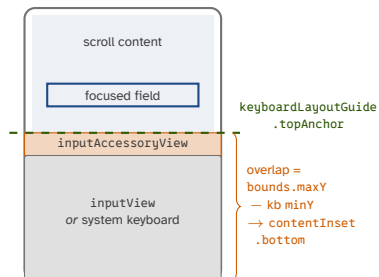
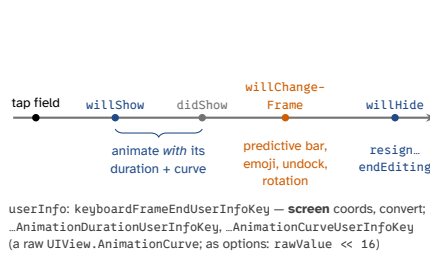


In one line: The keyboard belongs to the **first responder**: iOS shows its `inputView` (default: the system keyboard) + `inputAccessoryView`, and **your** layout moves — with `keyboardLayoutGuide` (iOS 15) or from the **real end frame** in the notifications. Input is shaped by **traits**, and is **provisional** while an IME holds marked text.

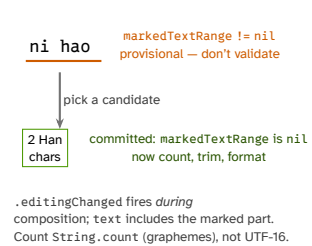
Geometry



Notifications (UIResponder.keyboard...)



Marked text (IME, dictation)



How it works

- First responder:** `becomeFirstResponder()` returns `Bool` — `false` if not in a window yet or `canBecomeFirstResponder` is `false`. `Dismiss: resignFirstResponder(), view.endEditing(true)`, or a `nil`-targeted `resignFirstResponder` action.
- `inputView` **replaces** the keyboard (a picker); `inputAccessoryView` rides **above** it. Chat composer: the VC returns `true` from `canBecomeFirstResponder` and vends the bar as its `inputAccessoryView`; with `keyboardDismissMode = .interactive` it follows the finger.
- keyboardLayoutGuide** (iOS 15): pin to its `topAnchor`; show/hide, height changes and animation come free; follows `UndockedKeyboard` for the floating iPad keyboard. With no keyboard it sits at the safe-area bottom.
- Scroll view:** inset content + indicators by the overlap *minus the bottom safe area* (already in the adjusted inset), `scrollRectToVisible`, reset on hide. `UITableViewController` does it for you.
- Traits:** `keyboardType` (`.emailAddress`, `.numberPad`, `.decimalPad`, `.URL`); `textContentType` drives `AutoFill` — `.username`, `.password`, `.newPassword` (+ `passwordRules`), `.oneTimeCode` (SMS code offered above the keyboard, no permission); `returnKeyType`; `autocorrectionType`, `smartQuotesType`, `smartDashesType`.
- UITextField:** the delegate asks (`shouldBeginEditing`, `shouldChangeCharactersIn` with an `NSRange` in UTF-16, `textFieldShouldReturn`); control events *report* (`.editingChanged` incl. `paste/AutoFill`, `.editingDidEndOnExit`; `addAction(_:for:)`). A programmatic text = triggers neither.
- Hardware keyboard:** `UIKeyCommand(title:action:input:modifierFlags:)` in a responder's `keyCommands` (or the `buildMenu(with:)` menu) — titled ones appear when `Cmd` is held; `wantsPriorityOverSystemBehavior` (iOS 15) wins keys the system uses. Raw keys: `pressesBegan` → `UIPress.key`. The software keyboard shrinks to a shortcut bar.
- SwiftUI:** avoids the keyboard itself (iOS 14; opt out `.ignoresSafeArea(.keyboard)`). iOS 15: `@FocusState` + `.focused`, `.submitLabel` + `.onSubmit`, a `ToolbarItemGroup(placement: .keyboard)`; iOS 16 `.scrollDismissesKeyboard`; iOS 17 `.onKeyPress`.

Example

```
// iOS 15+: pin the composer to the keyboard - no observers
composer.bottomAnchor.constraint(
    equalTo: view.keyboardLayoutGuide.topAnchor).isActive = true
// scroll view, from keyboardWillChangeFrame:
@objc func kbChanged(_ n: Notification) {
    guard let end = n.userInfo?[UIResponder
        .keyboardFrameEndUserInfoKey] as? CGRect,
        let screen = view.window?.screen else { return }
    let kb = view.convert(end, from: screen.coordinateSpace)
    let overlap = max(0, view.bounds.maxY - kb.minY
        - view.safeAreaInsets.bottom)
    scroll.contentInset.bottom = overlap
    scroll.verticalScrollIndicatorInsets.bottom = overlap }
// SwiftUI
TextField("Email", text: $email).focused($field, equals: .email)
.textContentType(.emailAddress).keyboardType(.emailAddress)
.submitLabel(.next).onSubmit { field = .password }
```

Interview traps

- ! A hard-coded keyboard height, or only `willShow`: the predictive bar, emoji, hardware or floating keyboard change the frame — handle `willChangeFrame`, or use the guide.
- ! Keyboard frame is in **screen** coordinates; in Split View / Stage Manager the window is not the screen — convert.
- ! `becomeFirstResponder()` in `viewDidLoad` returns `false` (no window); do it in `viewDidAppear`. SwiftUI's twin: setting `@FocusState` in `onAppear` can be dropped — defer it.
- ! Max length in `shouldChangeCharactersIn`: the range is **UTF-16**, emoji are several units, and marked text is not final — compute the result with `Range(range, in:)`, skip while `markedTextRange != nil`.
- ! Smart quotes / dashes / autocorrect in username, code or search fields silently change " into " — set `.no`.
- ! `.numberPad` has no Return key: add a Done in the accessory view / keyboard toolbar.

Remember

Responder owns the keyboard · guide beats notifications · read the real frame · marked text is not text yet.

Likely questions

- Composer glued to the keyboard? — `keyboardLayoutGuide`, or `inputAccessoryView + .interactive`.
- SMS code `AutoFill`? — `textContentType = .oneTimeCode`.
- Next field on Return? — `returnKeyType = .next`; in `textFieldShouldReturn` make the next one first responder.
- `Cmd-S` on iPad? — `UIKeyCommand / .keyboardShortcut("s")`.