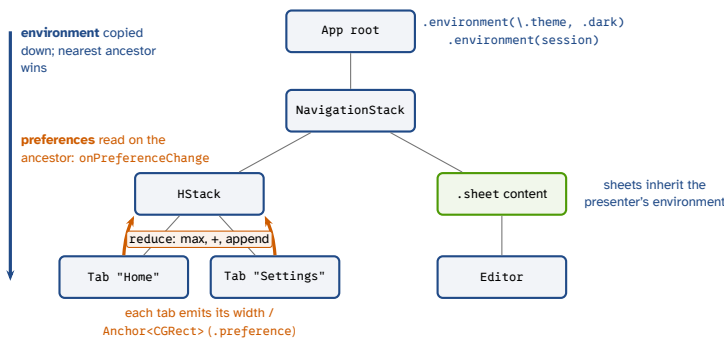
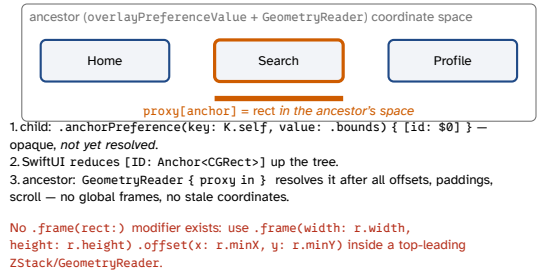


In one line: Two **implicit** channels through the view tree: the **environment flows down** (an ancestor sets a value, every descendant can read it), **preferences flow up** (descendants emit values, SwiftUI **reduces** them, an ancestor reads the result). Explicit `init` parameters remain the default for a view's own data.



Anchor preference — a custom tab bar underline



How it works — down

- EnvironmentValues** — value-typed bag keyed by `EnvironmentKey` types. Read `@Environment(\.colorScheme)`; set `.environment(\.key, v)`; augment the inherited value with `.transformEnvironment`.
- Custom key:** type with static `let defaultValue + computed property` on `EnvironmentValues` over `self[Key.self]`. `@Entry` (Xcode 16) writes both. A keyed read **never crashes** — it falls back to `defaultValue`.
- Objects:** `ObservableObject` → `.environmentObject(o) + @EnvironmentObject` (any `@Published` invalidates every reader). `@Observable` (iOS 17) → `.environment(o) + @Environment(Session.self)`: per-property tracking. Missing object = **runtime crash**; type it `Session?` to make it optional.
- Re-render:** only views whose body *reads* the changed value — read churning values low.
- DI:** `@Entry var api: APIClient = .live`; previews/tests inject `.mock`. Ambient concerns only; the item a view shows stays in `init`.

How it works — up

- PreferenceKey:** `defaultValue + reduce(value: inout V, nextValue: () -> V)`. Emit `.preference(key:value:)`; read `.onPreferenceChange(Equatable)`, or build UI with `overlayPreferenceValue / backgroundPreferenceValue`.
- reduce folds siblings **in tree order**; non-emitting views may contribute `defaultValue`, so it **must be the identity of reduce** (`0` for max, `[]` for append). Only descendants *below the reader* are aggregated.
- Uses: equal-width labels, child size (`GeometryReader` in `.background` — no layout effect), scroll offset, custom tab bars, tooltips, connector lines.

Example — custom key + child-size preference

```
extension EnvironmentValues { @Entry var theme: Theme = .light }
struct MaxWidth: PreferenceKey {
    static let defaultValue: CGFloat = 0 // identity of max
    static func reduce(value: inout CGFloat,
        nextValue: () -> CGFloat) {
        value = max(value, nextValue()) } }
Text(label).background(GeometryReader { g in Color.clear
    .preference(key: MaxWidth.self, value: g.size.width) })
.onPreferenceChange(MaxWidth.self) { w in // on the ancestor
    if w != labelWidth { labelWidth = w } } // only on change
```

Interview traps

- Not injected:** `@EnvironmentObject` / non-optional `@Environment(T.self)` compiles, then crashes at render. Culprits: `#Preview`, a new `UIHostingController` (fresh environment), injecting on a sibling branch or inside a `NavigationLink` destination.
- Down only** — a modifier *below* the reader is invisible to it; inject at a common ancestor.
- Reference type in an EnvironmentKey:** mutating its insides invalidates nobody. Keys = immutable config (theme, formatter, flags, clients); mutable shared state = `@Observable`.
- Preference loop:** size → `@State` → child resizes → new size... “Bound preference ...tried to update multiple times per frame”. Store only on change, measure in `.background`; newer SDKs: `onGeometryChange`.
- reduce that overwrites (`value = nextValue()`) or ignores `nextValue()` keeps one child.
- Swift 6:** static `var defaultValue` is global mutable state — use static `let`.
- `\.dismiss` in a root with no presenter is a silent no-op.

Choosing the channel

the view's own data	<code>init param / @Binding</code>
ambient config, services	<code>EnvironmentKey / @Entry</code>
shared mutable model	<code>@Observable + .environment(obj)</code>
child → parent value	<code>PreferenceKey + onPreferenceChange</code>
child rect for drawing	<code>anchorPreference + overlayPreferenceValue</code>

Remember

“Config rains down, measurements bubble up — and the default is the identity.”

Likely questions

- Why can `@Environment(\.x)` never crash? — keys resolve to `defaultValue`.
- `@EnvironmentObject` vs `@Environment(T.self)`? — whole-object vs per-property invalidation (17).
- Why anchors, not global frames? — resolved in the reader's space after every transform.
- DI through the environment? — `.live` default, `.mock` in previews/tests.