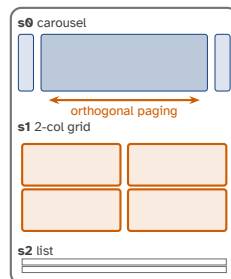
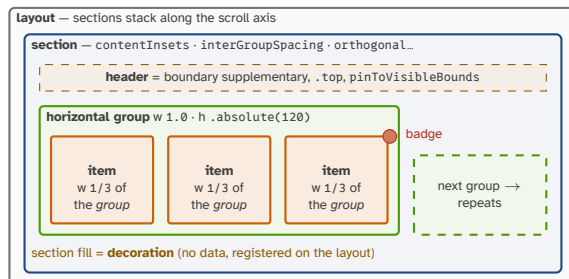


In one line: A compositional layout (iOS 13) is **declared**, not computed in delegate callbacks: **items** sit in **groups** (the repeating unit), a group repeats along a **section**, sections stack into the **layout**. Every size is `.absolute`, `.estimated` or `.fractional` of *its container*. Cells are dequeued from **registrations** and rendered from **content configurations** (iOS 14).



UICollectionViewLayoutDimension

`.absolute(120)` — fixed points
`.estimated(44)` — a guess; the cell's Auto Layout measures the real size
`.fractionalWidth(0.5) / Height` — of the **container**: an item's container is its **group**; a group's is the section's content area (after insets). Height may be `.fractionalWidth` — squares.
orthogonal: `.none` · `.continuous` · `.continuousGroupLeadingBoundary` · `.paging` (container width) · `.groupPaging` · `.groupPagingCentered` (one group)
s0 = group width 0.9 so neighbours peek; at 1.0 it looks like `.paging`.

How it works

- Group** — `.horizontal/.vertical(layoutSize:subitems:)`, `repeatingSubitem:count:` (iOS 16), groups nest (big tile + 2×2); `.custom` returns explicit frames. `interItemSpacing` (`.fixed/.flexible`) between items only.
- Section provider** — the closure `init { index, env in ... }` builds each section; `env.container` (effective content size) + `env.traitCollection` pick 2 vs 4 columns. Map index → your section via `sectionIdentifier(for:)` (iOS 15), not raw indices.
- List** (iOS 14) — `UICollectionViewLayoutListConfiguration`: `.plain/.grouped/.insetGrouped/.sidebar`, headerMode, swipe-action providers; `.list(using:layoutEnvironment:)` per section mixes lists with grids — the modern UITableView.
- Supplementary** — headers/footers = boundary items (`elementKind`, `alignment`, `pinToVisibleBounds`); badges = item-anchored (`containerAnchor`); data-backed, via `SupplementaryRegistration`. **Decoration** = visual only, class registered on the layout.
- Registration** (iOS 14) — `CellRegistration<Cell, Item>` + `dequeueReusableCellReusableCell(using:for:item:)`: no string ids, typed item. Create it **once**.
- Content configuration** — `defaultContentConfiguration()` → `UIListContentConfiguration` (`text`, `secondaryText`, `image`) → `cell.contentConfiguration`: a value describes the look, the content view applies it. Custom: `UIContentConfiguration` (`makeContentView()`, `updated(for:)`) + a `UIContentView`. States: `configurationUpdateHandler` (iOS 15). SwiftUI: `UIHostingConfiguration` (iOS 16).
- Self-sizing** — `.estimated` on **both** item and group for that axis + a fully constrained cell; estimates near reality or content jumps.
- Invalidation** — a bounds change re-runs the provider (rotation for free); `invalidateLayout()` for other inputs; `setCollectionViewLayout(_:animated:)` swaps layouts.

Example

```
let layout = UICollectionViewCompositionalLayout { i, env in
    if i > 0 { return .list(using: .init(appearance: .insetGrouped),
                               layoutEnvironment: env) } // a list
    let item = NSCollectionViewLayoutItem(layoutSize: .init(
        widthDimension: .fractionalWidth(1),
        heightDimension: .fractionalHeight(1))) // fill the group
    let group = NSCollectionViewLayoutGroup.horizontal(layoutSize:
        .init(widthDimension: .fractionalWidth(0.9), // peek
              heightDimension: .absolute(180)), subitems: [item])
    let s = NSCollectionViewLayoutSection(group: group)
    s.orthogonalScrollingBehavior = .groupPagingCentered
    return s }
let reg = UICollectionView.CellRegistration< // ONCE
    UICollectionViewCell, Item> { cell, _, item in
    var c = cell.defaultContentConfiguration()
    c.text = item.title; cell.contentConfiguration = c }
```

Interview traps

- !** `sizeForItemAt` does nothing — only the flow layout reads `UICollectionViewDelegateFlowLayout`. The silent migration bug.
- !** `.fractionalHeight(1)` on an item = fill the **group**, not the screen.
- !** Item content insets (shrink each item, gaps double between neighbours) + `interItemSpacing` = surprise gutters. Pick one; section insets for edges.
- !** `.estimated` on the item but `.absolute` on the group — no self-sizing.
- !** Creating a `CellRegistration` inside the cell provider — re-registers per cell; iOS 15+ raises an exception.
- !** An orthogonal section scrolls in its own internal scroll view — `scrollViewDidScroll` does not fire; use `visibleItemsInvalidationHandler` (per frame: keep it cheap).

Remember

Item in group in section in layout; fractions of the container; estimate both levels; register once, configure by value.

Likely questions

- App Store-style shelves? — section provider + orthogonal sections.
- Adaptive columns? — read `env.container` in the provider.
- Supplementary vs decoration? — data-backed vs pure visual on the layout.
- Flow → compositional? — `sectionInset` → `contentInsets`, line spacing → `interGroupSpacing`.