Building Modern CollectionViews with CompositionalLayout

CocoaHeads Aachen

Jonas Schlabertz, 28.7.2022

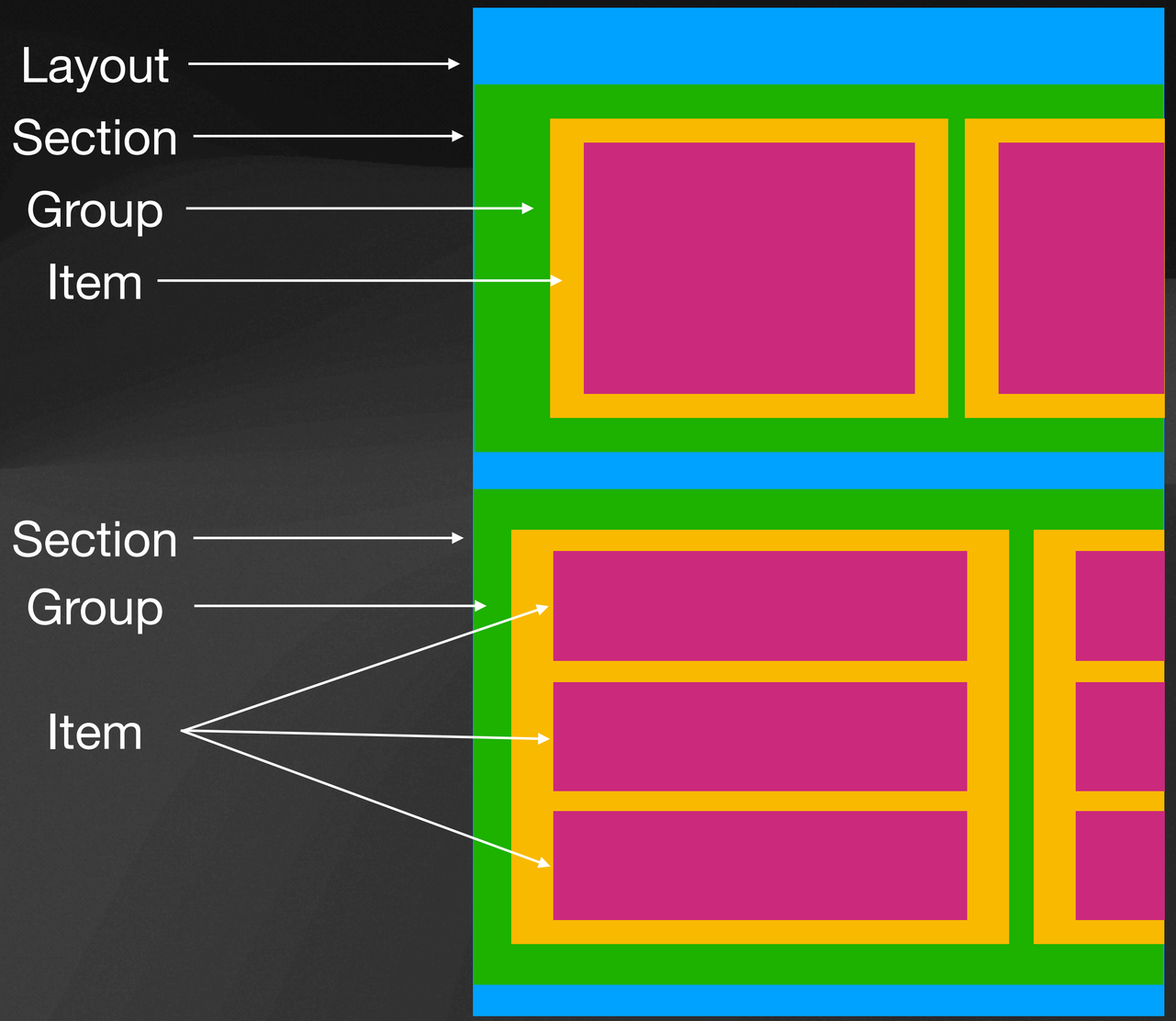
A Treat for our SwiftUI Fans

- UICollectionViewCells now support SwiftUI directly (iOS 16+)

```
cell.contentConfiguration = UIHostingConfiguration {
    Text("Hello World!")
}
```

- Now let’s talk about a serious UI framework 😊

CompositionalLayout Basics



LayoutSize and LayoutDimension

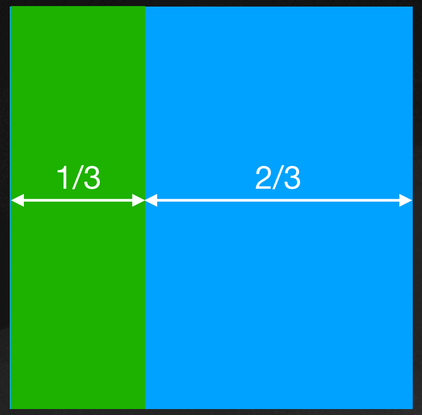
- NSCollectionLayoutSize describes the sizes of items and groups

```
class NSCollectionLayoutSize {
    init(
        widthDimension width: NSCollectionLayoutDimension,
        heightDimension height: NSCollectionLayoutDimension
    )
}
```

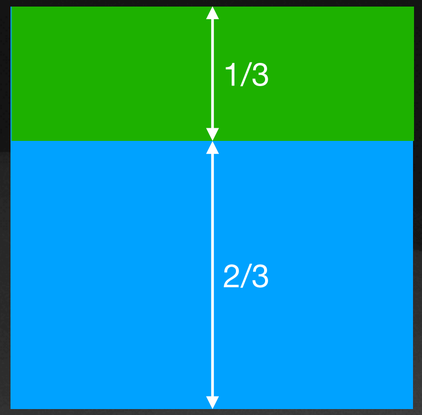
- NSCollectionLayoutDimension describes the individual dimensions of items and groups in relation to their surrounding container

```
class NSCollectionLayoutDimension {
    static func fractionalHeight(_ fractionalHeight: CGFloat) -> Self
    static func fractionalWidth(_ fractionalWidth: CGFloat) -> Self
    static func absolute(_ absoluteDimension: CGFloat) -> Self
    static func estimated(_ estimatedDimension: CGFloat) -> Self
}
```

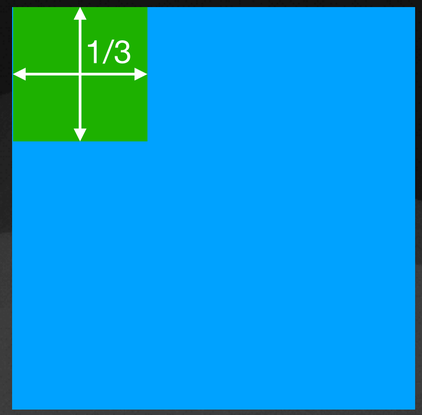
NSCollectionLayoutDimension Examples



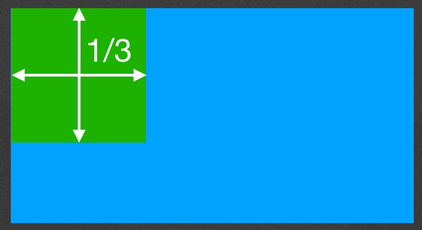

```
.fractionalWidth(1/3)
```



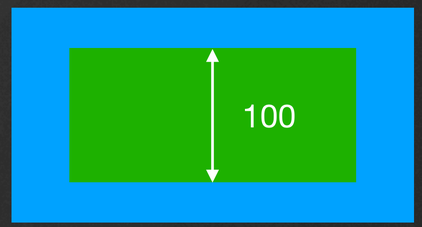

```
.fractionalHeight(1/3)
```



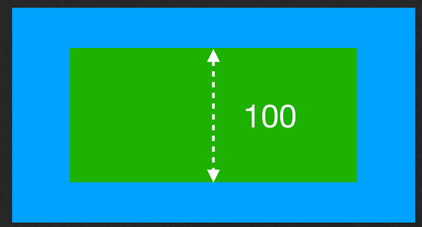

```
.fractionalWidth(1/3)
.fractionalHeight(1/3)
```




```
widthDimension: .fractionalWidth(1/3)
heightDimension: .fractionalWidth(1/3)
```




```
.absolute(100)
```




```
.estimated(100)
AutoLayout sets the final size
```

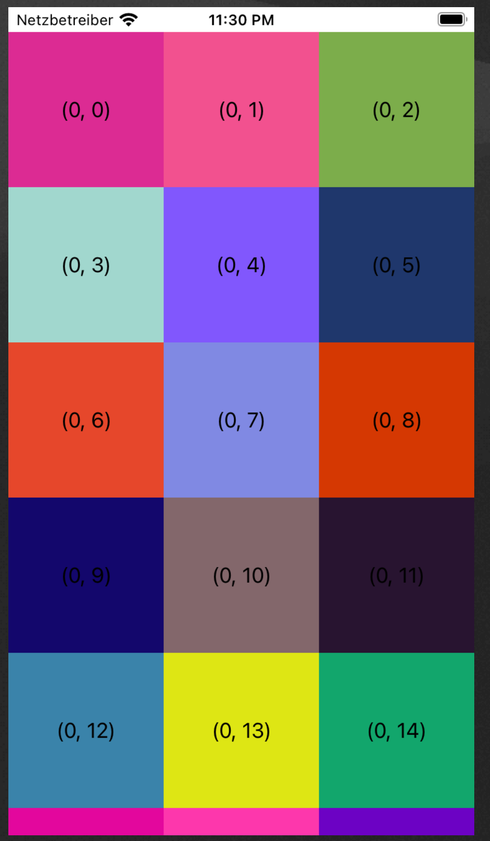
Initializing CompositionalLayout

- Example: Create a layout with TableView Sections

```
import UIKit

class Layout: UICollectionViewCompositionalLayout {
    init() {
        super.init(sectionProvider: { section, environment in
            let listConfig = UICollectionLayoutListConfiguration(appearance: .plain)
            let list: NSCollectionLayoutSection =  .list(using: listConfig,
                                                         layoutEnvironment: environment)

            return list
        })
    }
}
```

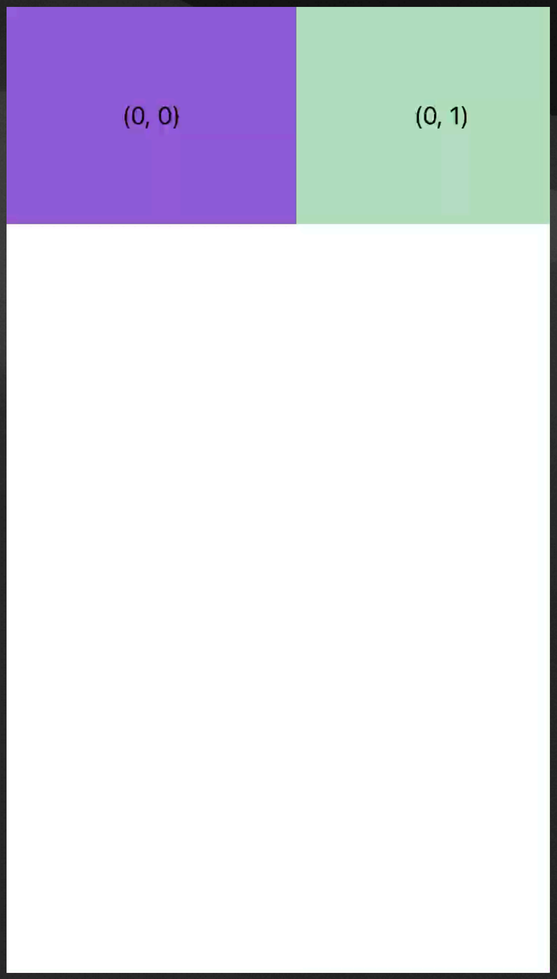
Replicating Flow Layout

```
let itemSize = NSCollectionLayoutItem(widthDimension: .fractionalWidth(1/3),
                                      heightDimension: .fractionalHeight(1))
let item = NSCollectionLayoutItem(layoutSize: itemSize)

let groupSize = NSCollectionLayoutSize(widthDimension: .fractionalWidth(1),
                                       heightDimension: .fractionalWidth(1/3))
let group: NSCollectionLayoutGroup = .horizontal(layoutSize: groupSize,
                                                 repeatingSubitem: item,
                                                 count: 3)

let section = NSCollectionLayoutSection(group: group)
```



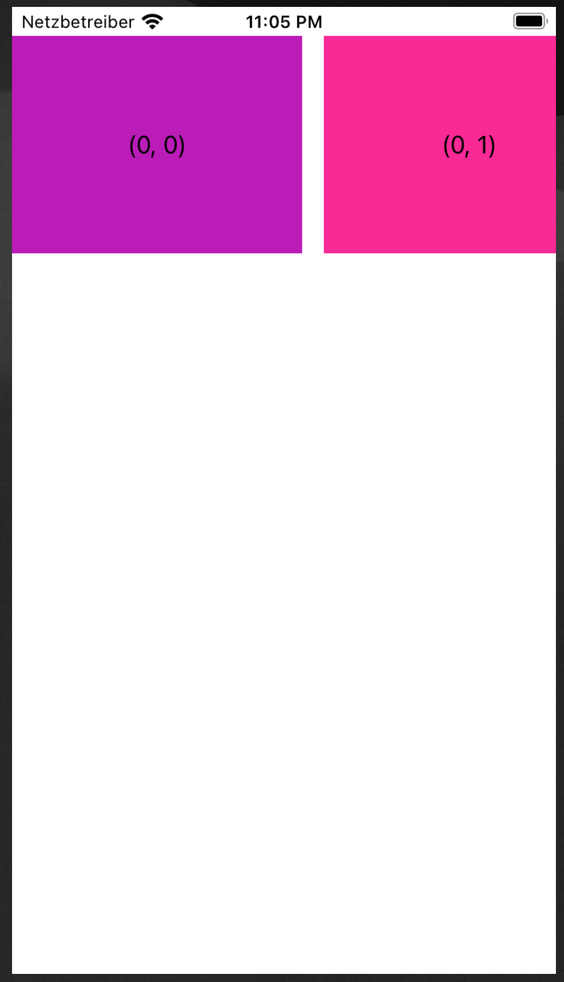
Horizontal Slider

```
let itemSize = NSCollectionLayoutSize(widthDimension: .fractionalWidth(1),
                                      heightDimension: .fractionalHeight(1))
let item = NSCollectionLayoutItem(layoutSize: itemSize)

let groupSize = NSCollectionLayoutSize(widthDimension: .absolute(200),
                                       heightDimension: .absolute(150))
let group: NSCollectionLayoutGroup = .horizontal(layoutSize: groupSize,
                                                 subitems: [item])

let section = NSCollectionLayoutSection(group: group)
```



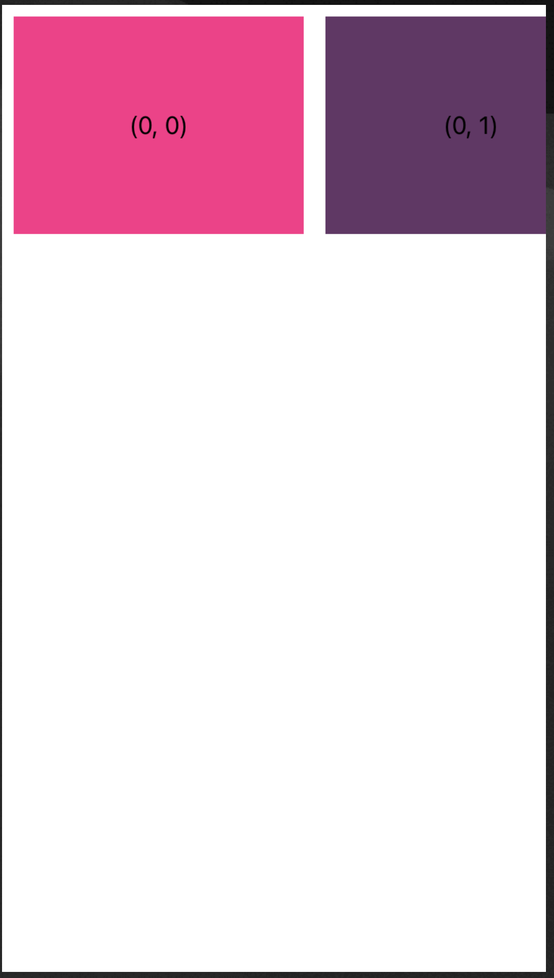
Horizontal Slider

```
let itemSize = NSCollectionLayoutSize(widthDimension: .fractionalWidth(1),
                                      heightDimension: .fractionalHeight(1))
let item = NSCollectionLayoutItem(layoutSize: itemSize)

let groupSize = NSCollectionLayoutSize(widthDimension: .absolute(200),
                                       heightDimension: .absolute(150))
let group: NSCollectionLayoutGroup = .horizontal(layoutSize: groupSize,
                                                 subitems: [item])

let section = NSCollectionLayoutSection(group: group)
section.interGroupSpacing = 15
```



Horizontal Slider

```
let itemSize = NSCollectionLayoutSize(widthDimension: .fractionalWidth(1),
                                      heightDimension: .fractionalHeight(1))
let item = NSCollectionLayoutItem(layoutSize: itemSize)

let groupSize = NSCollectionLayoutSize(widthDimension: .absolute(200),
                                       heightDimension: .absolute(150))
let group: NSCollectionLayoutGroup = .horizontal(layoutSize: groupSize,
                                                 subitems: [item])

let section = NSCollectionLayoutSection(group: group)
section.interGroupSpacing = 15
section.contentInsets = NSDirectionalEdgeInsets(top: 8,
                                                leading: 8,
                                                bottom: 8,
                                                trailing: 8)
```



Horizontal Slider

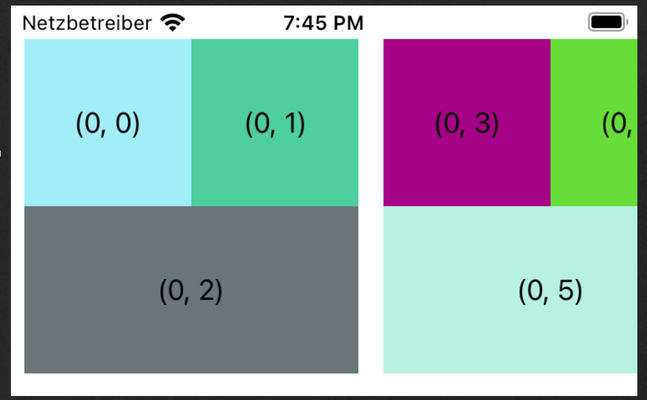
```
let itemSize = NSCollectionLayoutSize(widthDimension: .fractionalWidth(1),
                                      heightDimension: .fractionalHeight(1))
let item = NSCollectionLayoutItem(layoutSize: itemSize)

let groupSize = NSCollectionLayoutSize(widthDimension: .absolute(200),
                                       heightDimension: .absolute(150))
let group: NSCollectionLayoutGroup = .horizontal(layoutSize: groupSize,
                                                 subitems: [item])

let section = NSCollectionLayoutSection(group: group)
section.interGroupSpacing = 15
section.contentInsets = NSDirectionalEdgeInsets(top: 8,
                                                leading: 8,
                                                bottom: 8,
                                                trailing: 8)
```

- Sections support modifiers on horizontal scroll behavior

```
section.orthogonalScrollingBehavior = .continuous
section.orthogonalScrollingBehavior = .groupPaging
section.orthogonalScrollingBehavior = .groupPagingCentered
```

Combining Groups

```
let itemSize = NSCollectionLayoutSize(widthDimension: .fractionalWidth(1/2),
                                      heightDimension: .fractionalHeight(1))
let item = NSCollectionLayoutItem(layoutSize: itemSize)

let doubleItemGroupSize = NSCollectionLayoutSize(widthDimension: .fractionalWidth(1),
                                                 heightDimension: .fractionalHeight(1/2))
let doubleItemGroup: NSCollectionLayoutGroup = .horizontal(layoutSize: doubleItemGroupSize,
                                                           repeatingSubitem: item,
                                                           count: 2)

let fullWidthItemSize = NSCollectionLayoutSize(widthDimension: .fractionalWidth(1),
                                               heightDimension: .fractionalHeight(1/2))
let fullWidthItem = NSCollectionLayoutItem(layoutSize: fullWidthItemSize)

let groupSize = NSCollectionLayoutSize(widthDimension: .absolute(200),
                                       heightDimension: .absolute(200))
let group: NSCollectionLayoutGroup = .vertical(layoutSize: groupSize,
                                               subitems: [doubleItemGroup,
                                                          fullWidthItem])

let section = NSCollectionLayoutSection(group: group)
```



Adding Supplementary Views

- Sizes of supplementary views (e.g. section headers, decoration views for cells, …) need to be defined in the CompositionalLayout
- Example: Declaring the size of a section header

```
let size  = NSCollectionLayoutSize(widthDimension: .fractionalWidth(1),
                                   heightDimension: .estimated(40))

let kind = UICollectionView.elementKindSectionHeader
let item = NSCollectionLayoutBoundarySupplementaryItem(layoutSize: size,
                                                       elementKind: kind,
                                                       alignment: .top)

section.boundarySupplementaryItems = [item]
```

Observing User Interaction

- Sections support observation of the visible items within the section to react to user interaction
- ScrollOffset (CGPoint) also provided in the observation handler

```
section.visibleItemsInvalidationHandler = { visibleItems, scrollOffset, layoutEnvironment in
    // Handle new scroll position here
    // - visibleItems includes visible supplementary views and cells
    // - scrollOffset is a CGPoint
    // - layoutEnvironment contains information about content size and insets
}
```

- Can be set independently for each section instance

Any Questions?

Example App

https://github.com/Schlabbi/CompositionalLayout-Example

Jonas Schlabertz
jonas@schlabertz.de
CocoaHeads Aachen, 28.7.2022