#WWDC19

Advances in UI Data Sources

Steve Breen, UIKit Framework Engineer
Troy Stephens, AppKit Framework Engineer
Jacob Klapper, Share Sheet Engineer

Current state-of-the-art

A new approach

Demos

Considerations

Current State-of-the-Art

```swift
// MARK: UICollectionViewDataSource

func numberOfSections(in collectionView: UICollectionView) -> Int {
    return models.count
}
func collectionView(_ collectionView: UICollectionView,
                    numberOfItemsInSection section: Int) -> Int {
    return models[section].count
}
func collectionView(_ collectionView: UICollectionView,
                    cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {
    let cell = collectionView.dequeueReusableCell(withReuseIdentifier: reuseIdentifier,
                                                  for:indexPath)
    // configure cell
    return cell
}
```

UICollectionViewDataSource

Simple

Flexible

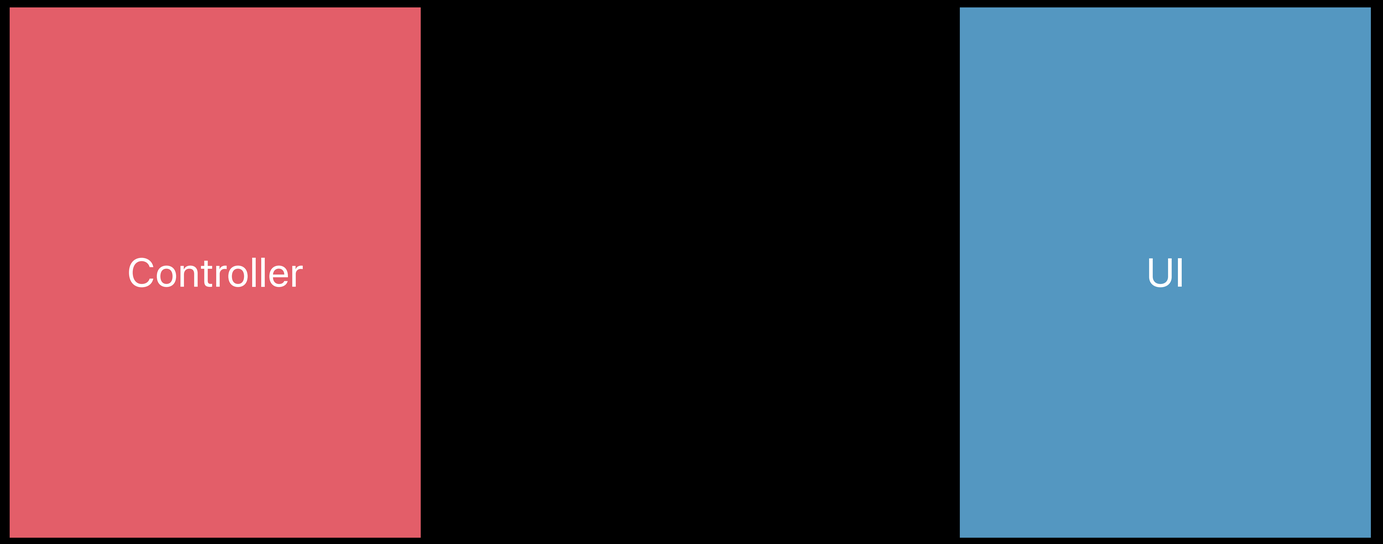
Apps Are Often Complicated

UI data sources backed by controllers

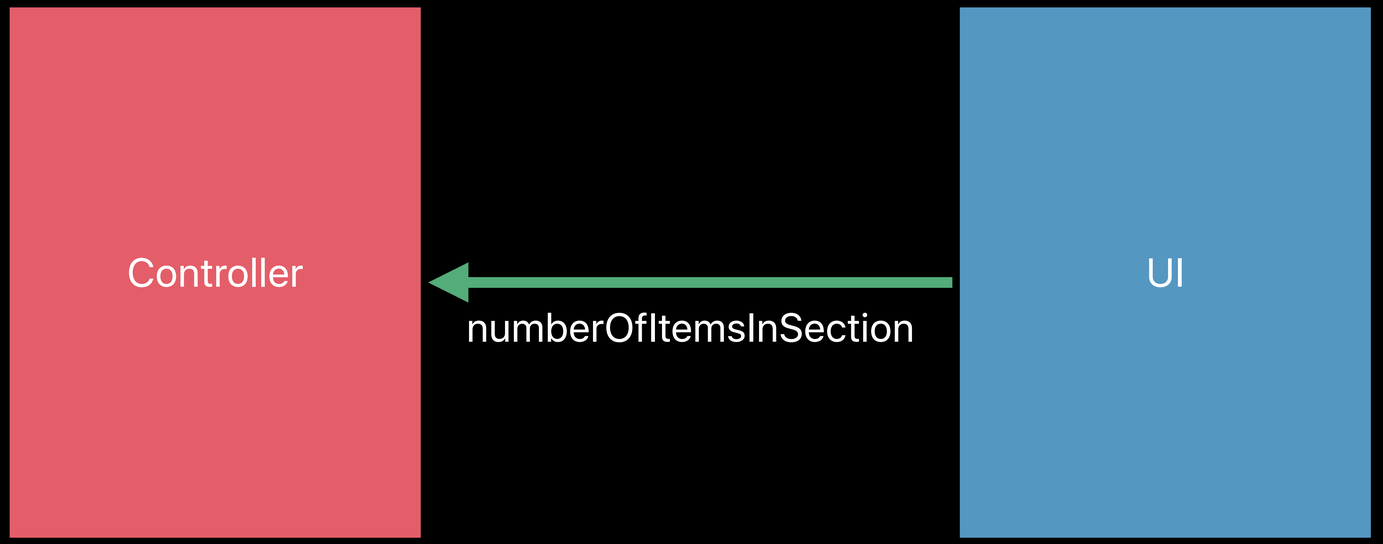
Core Data

Web services

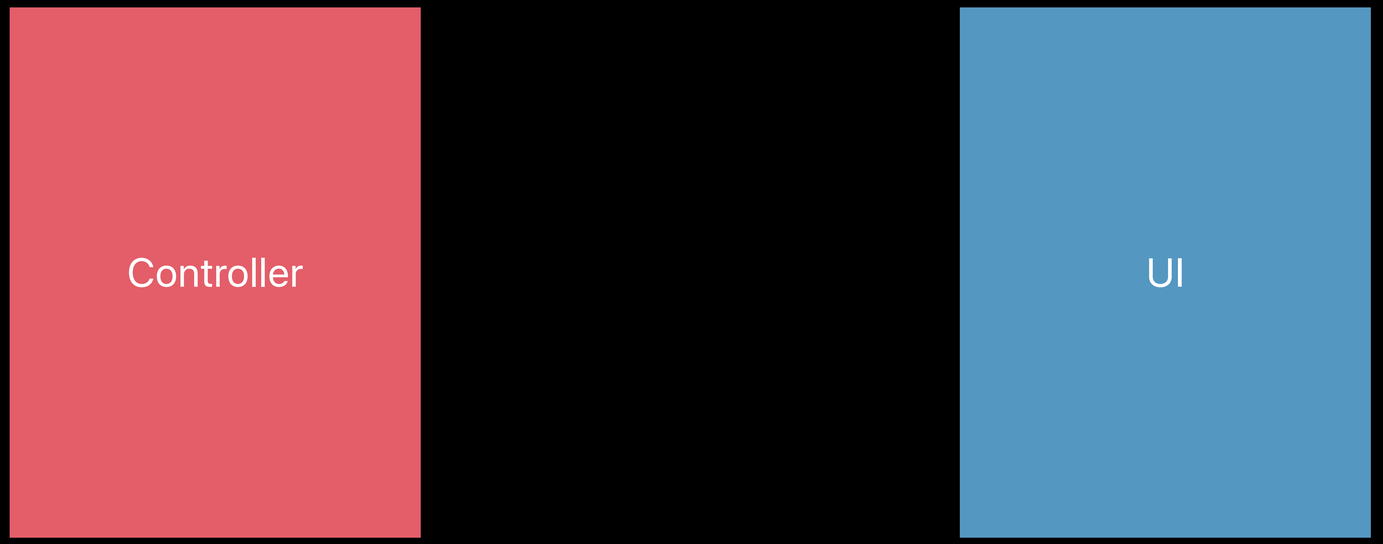
Generating UI Updates Can Be Challenging



Generating UI Updates Can Be Challenging

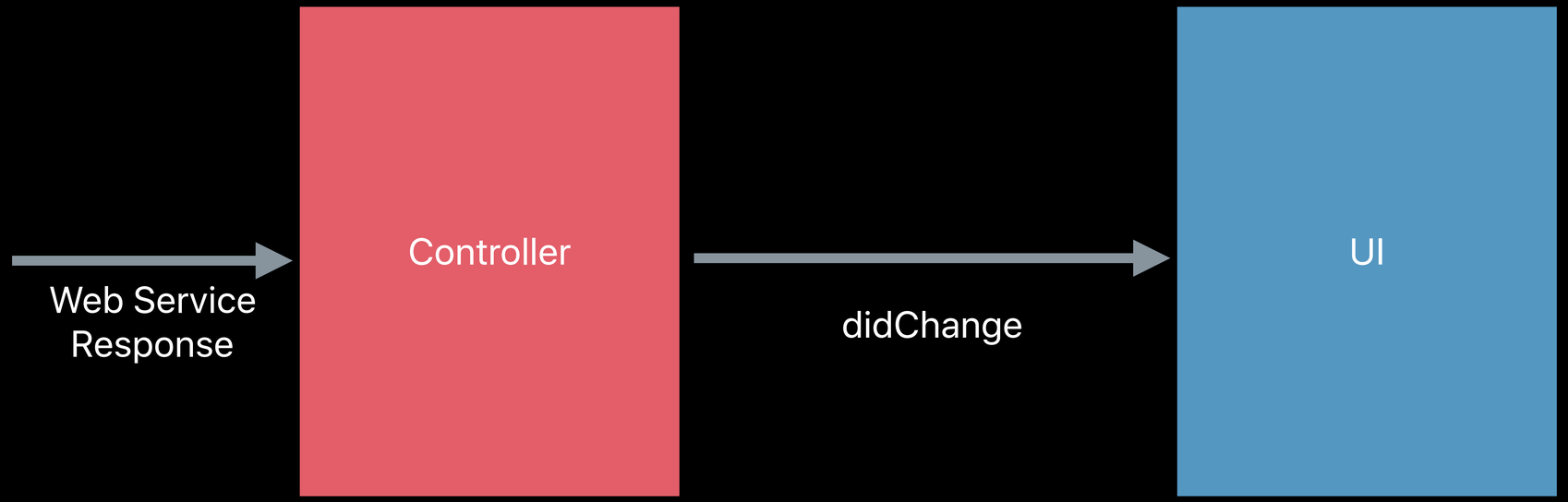


Generating UI Updates Can Be Challenging



Generating UI Updates Can Be Challenging

Generating UI Updates Can Be Challenging



When Updates Go Wrong 🔥

```
*** Terminating app due to uncaught exception 'NSInternalInconsistencyException', reason:
'Invalid update: invalid number of sections.  The number of sections contained in the
collection view after the update (10) must be equal to the number of sections contained in the
collection view before the update (10), plus or minus the number of sections inserted or
deleted (0 inserted, 1 deleted).'
***
```

Where Is Our Truth?

Our data source and current UI state must always agree

Current approach is error prone

No centralized truth

A New Approach

NEW

Diffable Data Source

A Declarative Approach to UI State

`performBatchUpdates()`

Crashing, hassles, complexity

`apply()`

Simple, automatic diffing

Snapshots

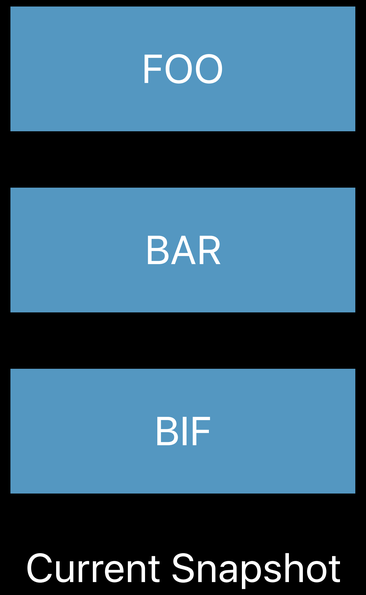
Truth of UI state

Unique identifiers for sections and items

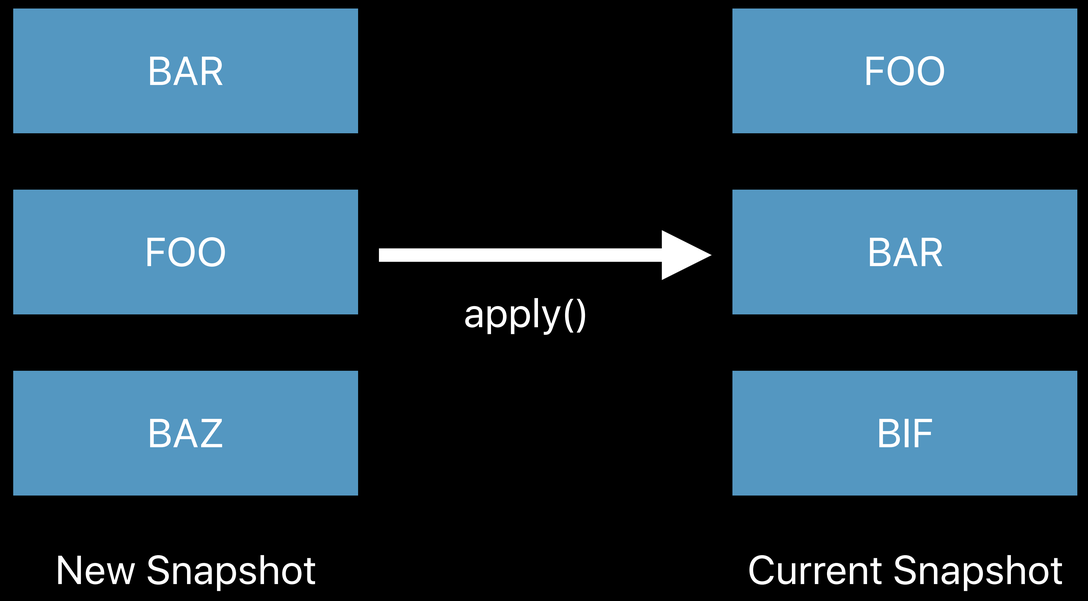
No more IndexPaths

Snapshot

Applying a Snapshot



Applying a Snapshot



Applying a Snapshot

Final Snapshot

Diffable Data Source

UICollectionViewDiffableDataSource

UITableViewDiffableDataSource

NSCollectionViewDiffableDataSource

NSDiffableDataSourceSnapshot

Demo

Show me some code already!

Considerations

Diffable Data Source Updates

Always call `apply()`

Never call `performBatchUpdates()`, `insertItems()`, etc

Constructing Snapshots

Empty snapshot

```
let snapshot = NSDiffableDataSourceSnapshot<Section, UUID>()
```

Current data source snapshot copy

```
let snapshot = dataSource.snapshot()
```

```
// Snapshot State

var numberOfItems: Int { get }
var numberOfSections: Int { get }
var sectionIdentifiers: [SectionIdentifierType] { get }
var itemIdentifiers: [ItemIdentifierType] { get }
```

```
// Configuring Snapshots

func insertItems(_ identifiers: [ItemIdentifierType],
                 beforeItem beforeIdentifier: ItemIdentifierType)

func moveItem(_ identifier: ItemIdentifierType,
              afterItem toIdentifier: ItemIdentifierType)

func appendItems(_ identifiers: [ItemIdentifierType],
                 toSection sectionIdentifier: SectionIdentifierType? = nil)

func appendSections(_ identifiers: [SectionIdentifierType])
```

Identifiers

Must be unique

Conforms to Hashable

Data model or identifier

```swift
// Custom Identifiers

struct MyModel: Hashable {
    let identifier = UUID()
    func hash(into hasher: inout Hasher) {
        hasher.combine(identifier)
    }
    static func == (lhs: MyModel, rhs: MyModel) -> Bool {
        return lhs.identifier == rhs.identifier
    }
}
```

```
// What About IndexPath-based APIs?

func collectionView(_ collectionView: UICollectionView,
                    didSelectItemAt indexPath: IndexPath) {
    if let identifier = dataSource.itemIdentifier(for: indexPath) {
        // Do something
    }
}
```

Performance

Fast

Measure your app

Safe to call `apply()` from a background queue

apply() From a Background Queue

Always call exclusively from the main queue or a background queue

Framework will log or assert

Demo

Share Sheet adoption

performBatchUpdates()

apply()

iOS, tvOS, and macOS

Automates animation

Easy, fast, and robust

More Information

developer.apple.com/wwdc19/220

Advances in Collection View Layout	WWDC 2019
UIKit and Collection Lab	Thursday, 9:00


WWDC19