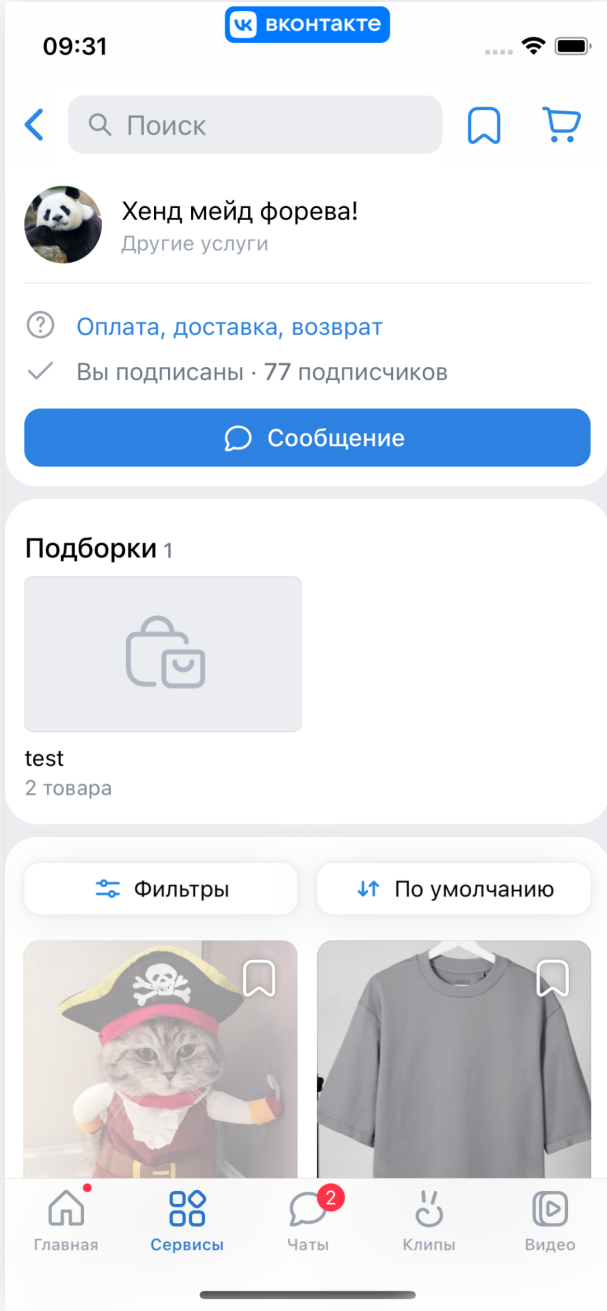
UICollectionViewLayout from scratch

Евгений JonFir Елчев

О чем доклад?

О том как, добиться полного контроля над UICollectionView

Пример


09:31
вконтакте
Поиск
Хенд мейд форева!
Другие услуги
Оплата, доставка, возврат
Вы подписаны · 77 подписчиков
Сообщение
Подборки 1
test
2 товара
Фильтры
По умолчанию
Главная
Сервисы
Чаты
2
Клипы
Видео

1

Плавающий
заголовок

1

Плавающий заголовок

2

Фон у секции

1

Плавающий заголовок

2

Фон у секции

3

Фиксированное количество столбцов

1

Плавающий заголовок

2

Фон у секции

3

Фиксированное количество столбцов

4

Адаптивная верстка

1

Плавающий заголовок

2

Фон у секции

3

Фиксированное количество столбцов

4

Адаптивная верстка

5

Анимации

1

Плавающий заголовок

2

Фон у секции

3

Фиксированное количество столбцов

4

Адаптивная верстка

5

Анимации

6

Контролируемый скрол

1

Плавающий заголовок

2

Фон у секции

3

Фиксированное количество столбцов

4

Адаптивная верстка

5

Анимации

6

Контролируемый скрол

7

120 fps

1

Плавающий заголовок

2

Фон у секции

3

Фиксированное количество столбцов

4

Адаптивная верстка

5

Анимации

6

Контролируемый скрол

7

120 fps

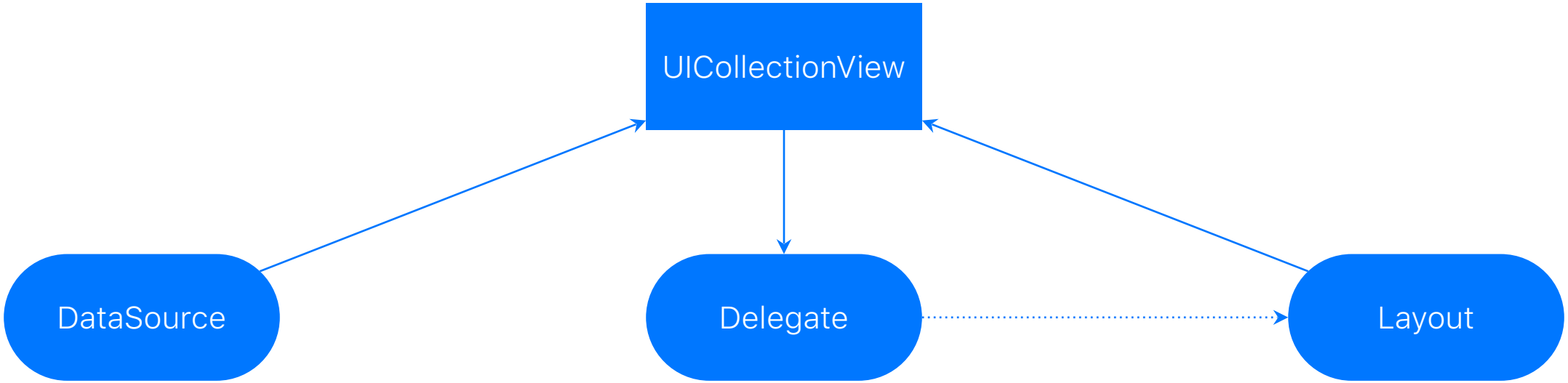
8

Без компромиссов с figma

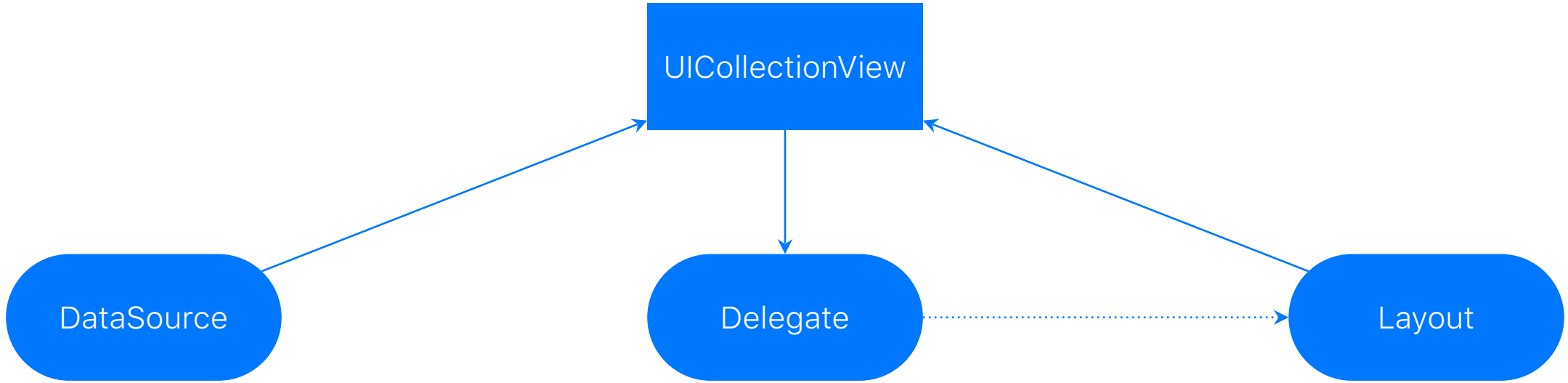
План

Как работает UICollectionView	13
UICollectionViewFlowLayout	16
UICollectionViewCompositionalLayout	31
UICollectionViewLayout	44

Как работает UICollectionView


UICollectionView
DataSource
Delegate
Layout

Отображает ячейки
Отвечает за реюз
Главный элемент



Количество секций, ячеек
Вью для ячеек и
дополнительных элементов

Обрабатывает действия пользователя
Предоставляет информацию для Layout

Информация о
фреймах ячеек

UICollectionViewFlowLayout

UICollectionViewFlowLayout позволяет

+ Можно задать размер ячейки

UICollectionViewFlowLayout позволяет

- Можно задать размер ячейки
- Установить минимальные отступы между ячейками

UICollectionViewFlowLayout позволяет

- Можно задать размер ячейки
- Установить минимальные отступы между ячейками
- Установить отступы между секциями

UICollectionViewFlowLayout позволяет

- Можно задать размер ячейки
- Установить минимальные отступы между ячейками
- Установить отступы между секциями
- Добавить header и footer для секции

UICollectionViewFlowLayout позволяет

- Можно задать размер ячейки
- Установить минимальные отступы между ячейками
- Установить отступы между секциями
- Добавить header и footer для секции
- Сделать header и footer плавающими

UICollectionViewFlowLayout позволяет

- Можно задать размер ячейки
- Установить минимальные отступы между ячейками
- Установить отступы между секциями
- Добавить header и footer для секции
- Сделать header и footer плавающими
- Расставляет ячейки по главной оси, отдавая все свободное место отступам

UICollectionViewFlowLayout не позволяет

- Задать количество строк или колонок

UICollectionViewFlowLayout не позволяет

- Задать количество строк или колонок

- Добавить декоративные элементы

UICollectionViewFlowLayout не позволяет

- Задать количество строк или колонок
- Добавить декоративные элементы
- Добавлять что либо кроме header и footer

UICollectionViewFlowLayout не позволяет

- Задать количество строк или колонок
- Задать свое расположение для ячеек
- Добавить декоративные элементы
- Добавлять что либо кроме header и footer

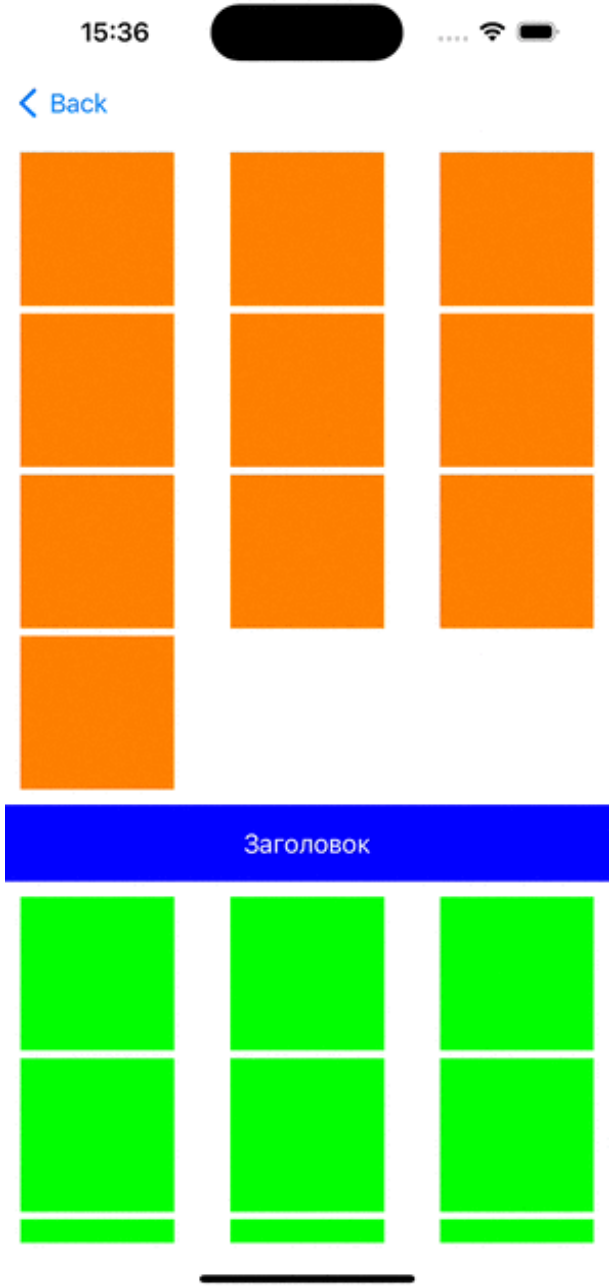
UICollectionViewFlowLayout не позволяет

- Задать количество строк или колонок
- Задать свое расположение для ячеек
- Добавить декоративные элементы
- Управлять плавающими элементами
- Добавлять что либо кроме header и footer

UICollectionViewFlowLayout не позволяет

- Задать количество строк или колонок
- Задать свое расположение для ячеек
- Добавить декоративные элементы
- Управлять плавающими элементами
- Добавлять что либо кроме header и footer
- Управлять анимациями и скролом

Результат



Дефолтный layout

UICollectionViewCompositional Layout

UICollectionViewCompositionalLayout позволяет

+ Задать размер ячейки

UICollectionViewCompositionalLayout позволяет

- Задать размер ячейки
- Задать отступы между секциями, ячейками и группами

UICollectionViewCompositionalLayout позволяет

- Задать размер ячейки
- Задать отступы между секциями, ячейками и группами
- Добавить header и footer для секции, сделать их плавающими

UICollectionViewCompositionalLayout позволяет

- Задать размер ячейки
- Задать количество строк и колонок
- Задать отступы между секциями, ячейками и группами
- Добавить header и footer для секции, сделать их плавающими

UICollectionViewCompositionalLayout позволяет

- Задать размер ячейки
- Задать количество строк и колонок
- Задать отступы между секциями, ячейками и группами
- Добавить декоративные элементы
- Добавить header и footer для секции, сделать их плавающими

UICollectionViewCompositionalLayout позволяет

- Задать размер ячейки
- Задать количество строк и колонок
- Задать отступы между секциями, ячейками и группами
- Добавить декоративные элементы
- Добавить header и footer для секции, сделать их плавающими
- Добавить дополнительные элементы

UICollectionViewCompositionalLayout не позволяет

- Гибко управлять декоративными элементами

UICollectionViewCompositionalLayout не позволяет

- Гибко управлять декоративными элементами

- Управлять плавающими элементами

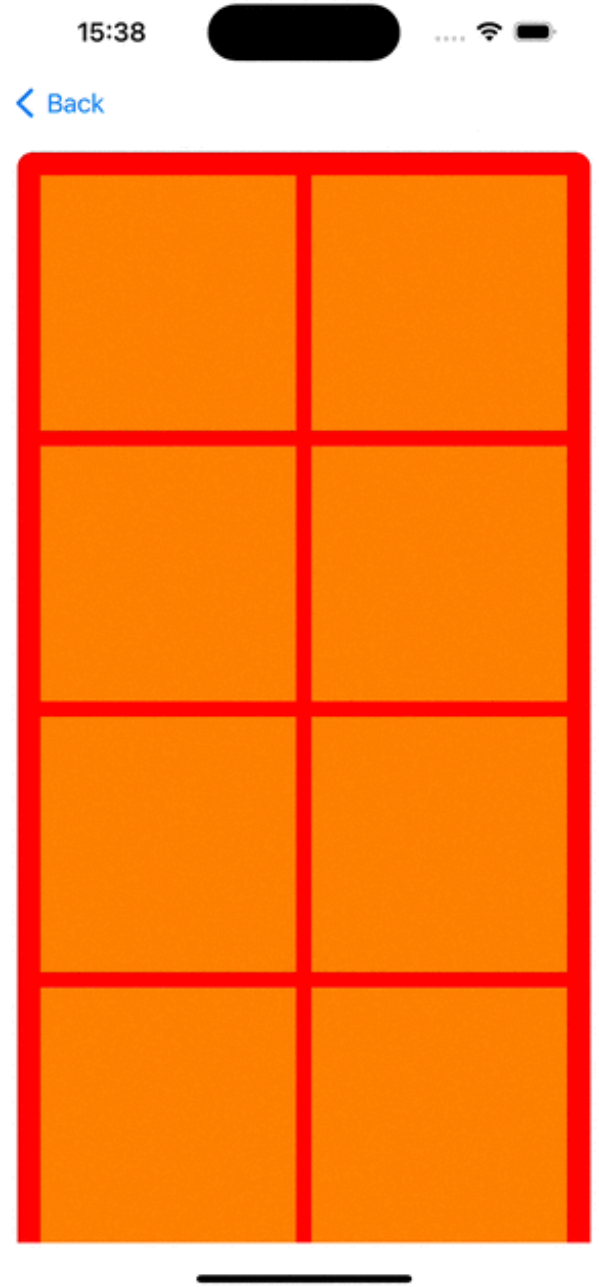
UICollectionViewCompositionalLayout не позволяет

- Гибко управлять декоративными элементами
- Управлять плавающими элементами
- Управлять анимациями

UICollectionViewCompositionalLayout не позволяет

- Гибко управлять декоративными элементами
- Управлять скролом
- Управлять плавающими элементами
- Управлять анимациями

Результат



Layout с гибкой расстановкой элементов

UICollectionViewLayout

UICollectionViewLayout позволяет

Все что угодно

UICollectionViewLayout сложный

- Все нужно делать самостоятельно

UICollectionViewLayout сложный

- Все нужно делать самостоятельно

- Нет внятной документации

UICollectionViewLayout сложный

- Все нужно делать самостоятельно
- Нет внятной документации
- Нет внятных примеров

UICollectionViewLayout сложный

- Все нужно делать самостоятельно
- Нет внятной документации
- Нет внятных примеров
- Нет хороших материалов в интернете

Есть есть несколько референсов на GitHub

Смотрим в доку

Methods to override

Every layout object should implement the following methods:

- `collectionViewContentSize`
- `layoutAttributesForElements(in:)`
- `layoutAttributesForItem(at:)`
- `layoutAttributesForSupplementaryView(ofKind:at:)` (if your layout supports supplementary views)
- `layoutAttributesForDecorationView(ofKind:at:)` (if your layout supports decoration views)
- `shouldInvalidateLayout(forBoundsChange:)`

These methods provide the fundamental layout information that the collection view needs to place contents on the screen. If your layout doesn't support supplementary or decoration views, don't implement the corresponding methods.

When the data in the collection view changes and items are to be inserted or deleted, the collection view asks its layout object to update the layout information. Specifically, any item that's moved, added, or deleted must have its layout information updated to reflect its new location. For moved items, the collection view uses the standard methods to retrieve the item's updated layout attributes. For items being inserted or deleted, the collection view calls some different methods, which you should override to provide the appropriate layout information:

- `initialLayoutAttributesForAppearingItem(at:)`
- `initialLayoutAttributesForAppearingSupplementaryElement(ofKind:at:)`
- `initialLayoutAttributesForAppearingDecorationElement(ofKind:at:)`
- `finalLayoutAttributesForDisappearingItem(at:)`
- `finalLayoutAttributesForDisappearingSupplementaryElement(ofKind:at:)`
- `finalLayoutAttributesForDisappearingDecorationElement(ofKind:at:)`

In addition to these methods, you can also override the `prepare(forCollectionViewUpdates:)` to handle any layout-related preparation. You can also override the `finalizeCollectionViewUpdates()` method and use it to add animations to the overall animation block or to implement any final layout-related tasks.

Смотрим в доку

Methods to override

Every layout object should implement the following methods:

- `collectionViewContentSize`
- `layoutAttributesForElements(in:)`
- `layoutAttributesForItem(at:)`

Пример галереи от Apple

Пример галереи от Apple

```
var cachedAttributes = [UICollectionViewLayoutAttributes]()

override func prepare() {
    super.prepare()

    cachedAttributes.removeAll()

    let count = collectionView.numberOfItems(inSection: 0)
    var currentIndex = 0
    var lastFrame: CGRect = .zero

    while currentIndex < count {
        for rect in segmentRects {
            let attributes = UICollectionViewLayoutAttributes(forCellWith: IndexPath(item: currentIndex, section: 0))
            attributes.frame = rect

            cachedAttributes.append(attributes)
            contentBounds = contentBounds.union(lastFrame)

            currentIndex += 1
            lastFrame = rect
        }
    }
}
```

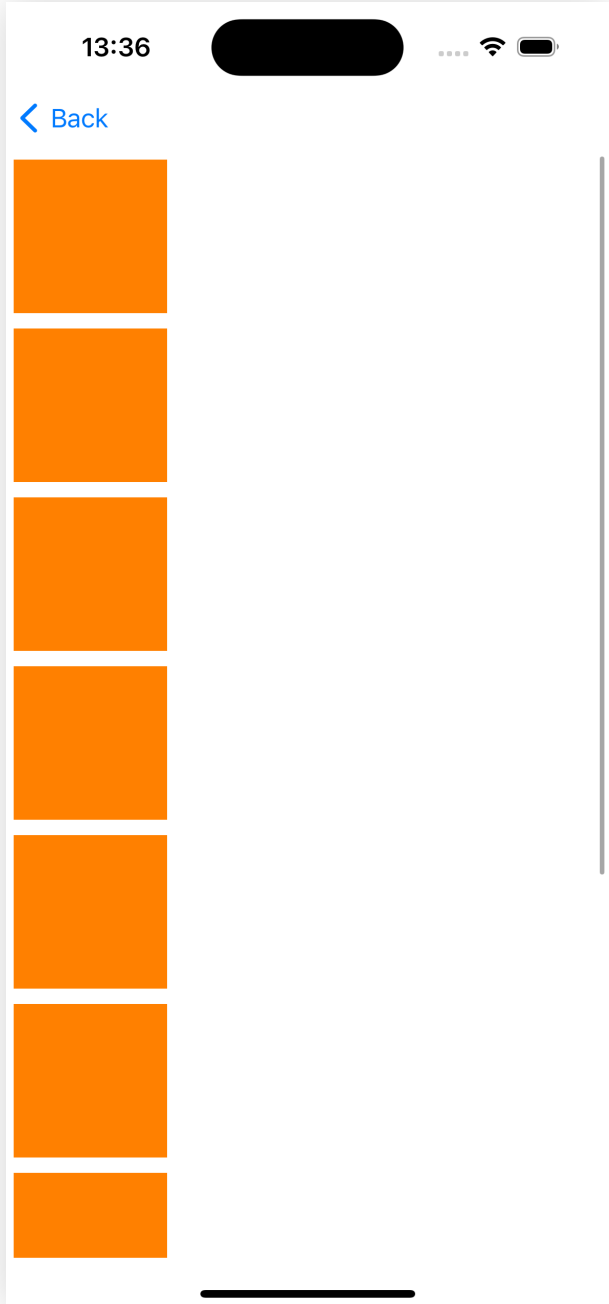
Пример галереи от Apple

```
let attributes = UICollectionViewLayoutAttributes(forCellWith: IndexPath(item: currentIndex, section: 0))
attributes.frame = rect
```

Причем тут prepare?

Давайте разбираться

Составим минимальный Layout



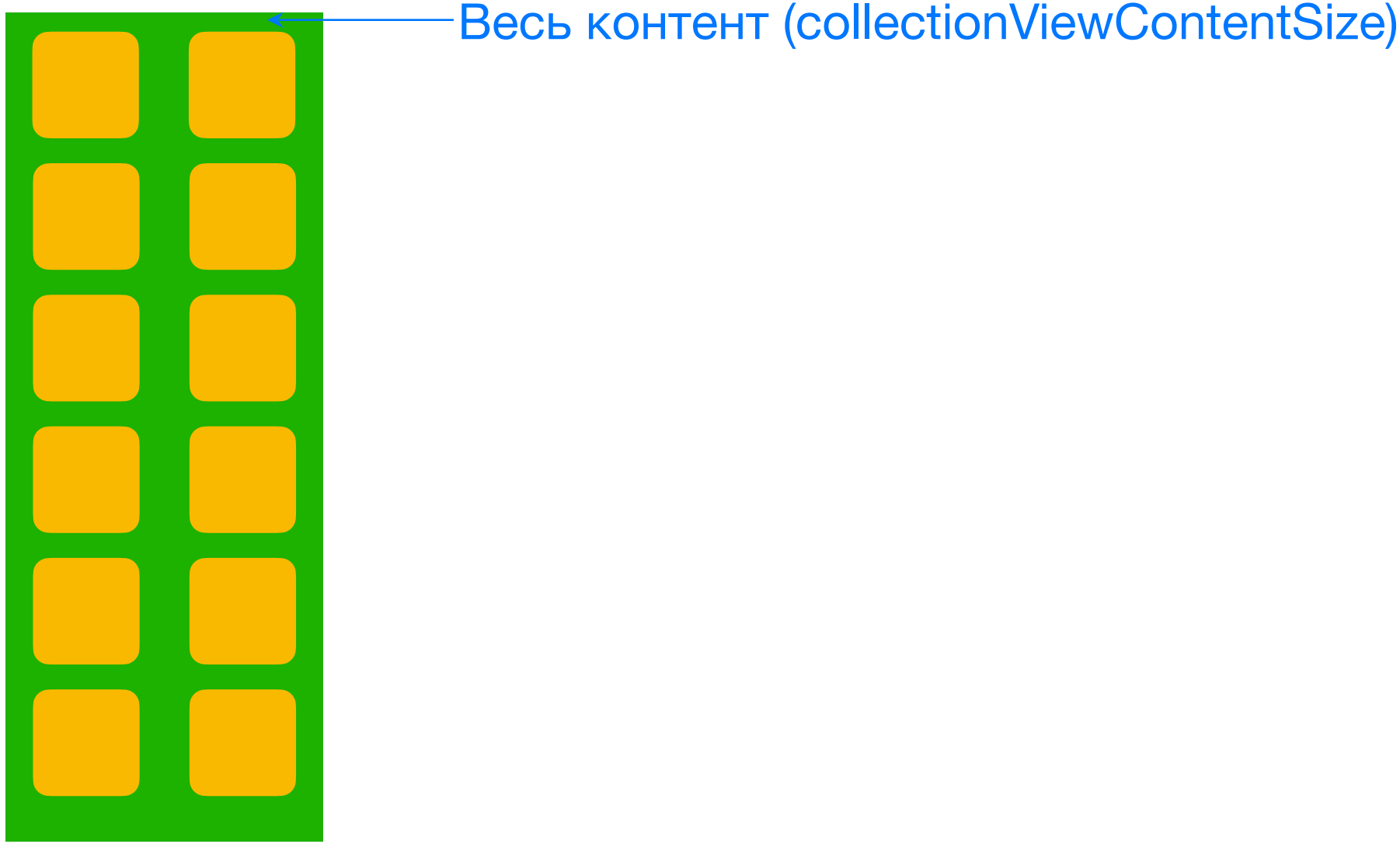
Вспомин аем доку

Methods to override

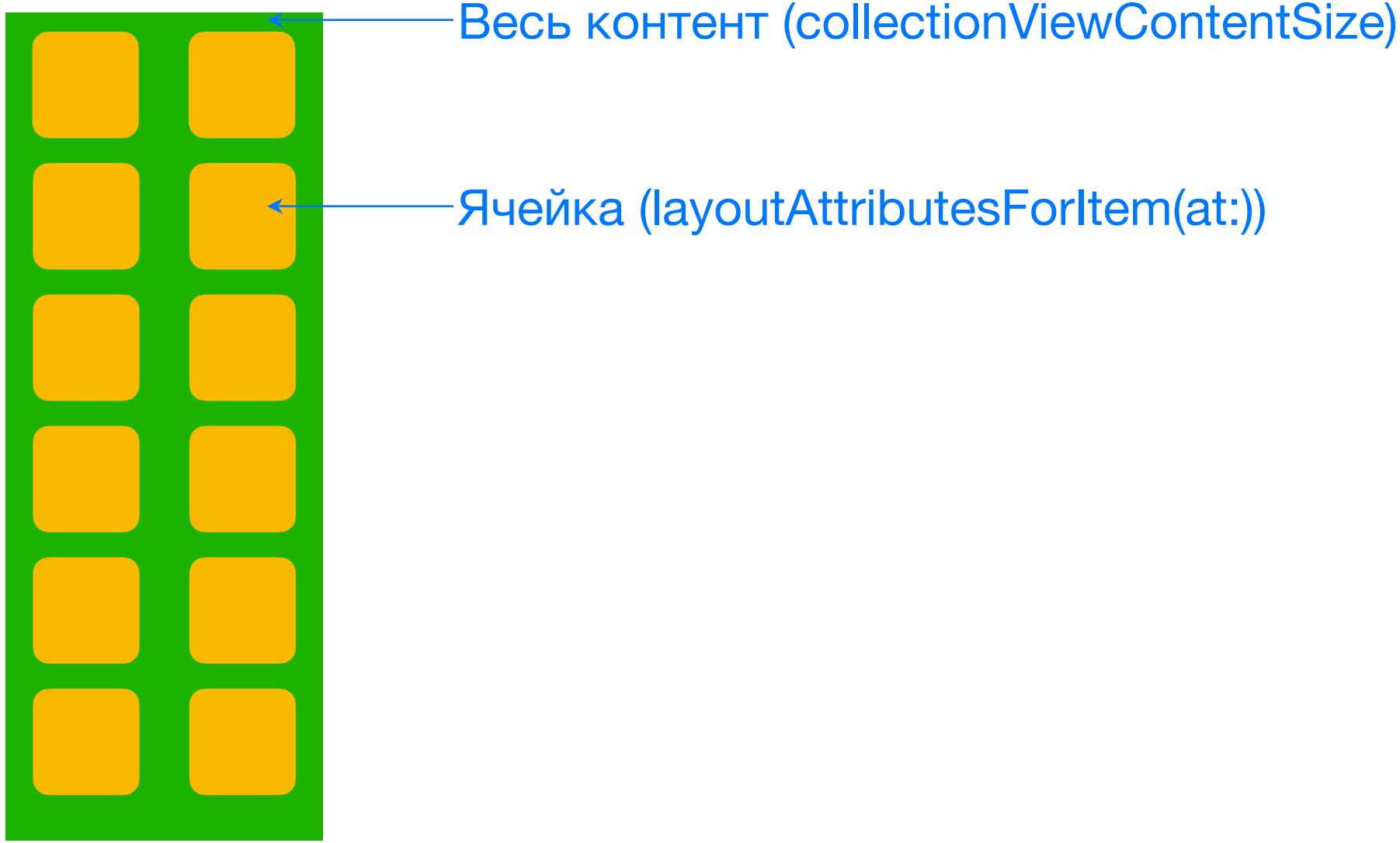
Every layout object should implement the following methods:

- `collectionViewContentSize`
- `layoutAttributesForElements(in:)`
- `layoutAttributesForItem(at:)`

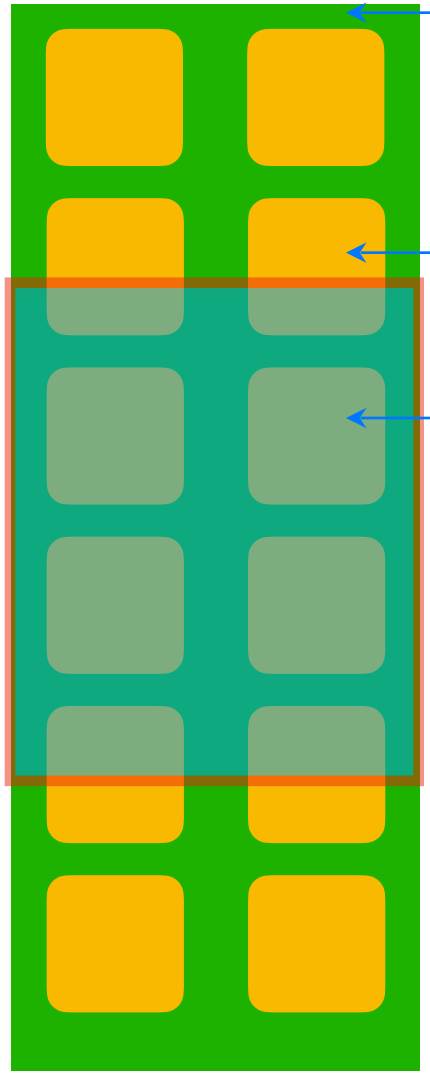
UICollectionViewLayout



UICollectionViewLayout



UICollectionViewLayout

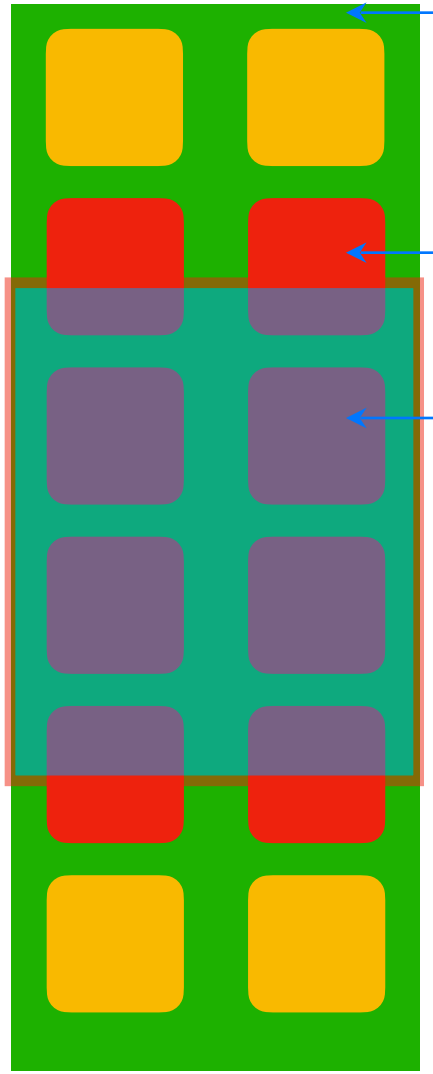


Весь контент (collectionViewContentSize)

Ячейка (layoutAttributesForItem(at:))

Видимая область (layoutAttributesForElements(in:))

UICollectionViewLayout

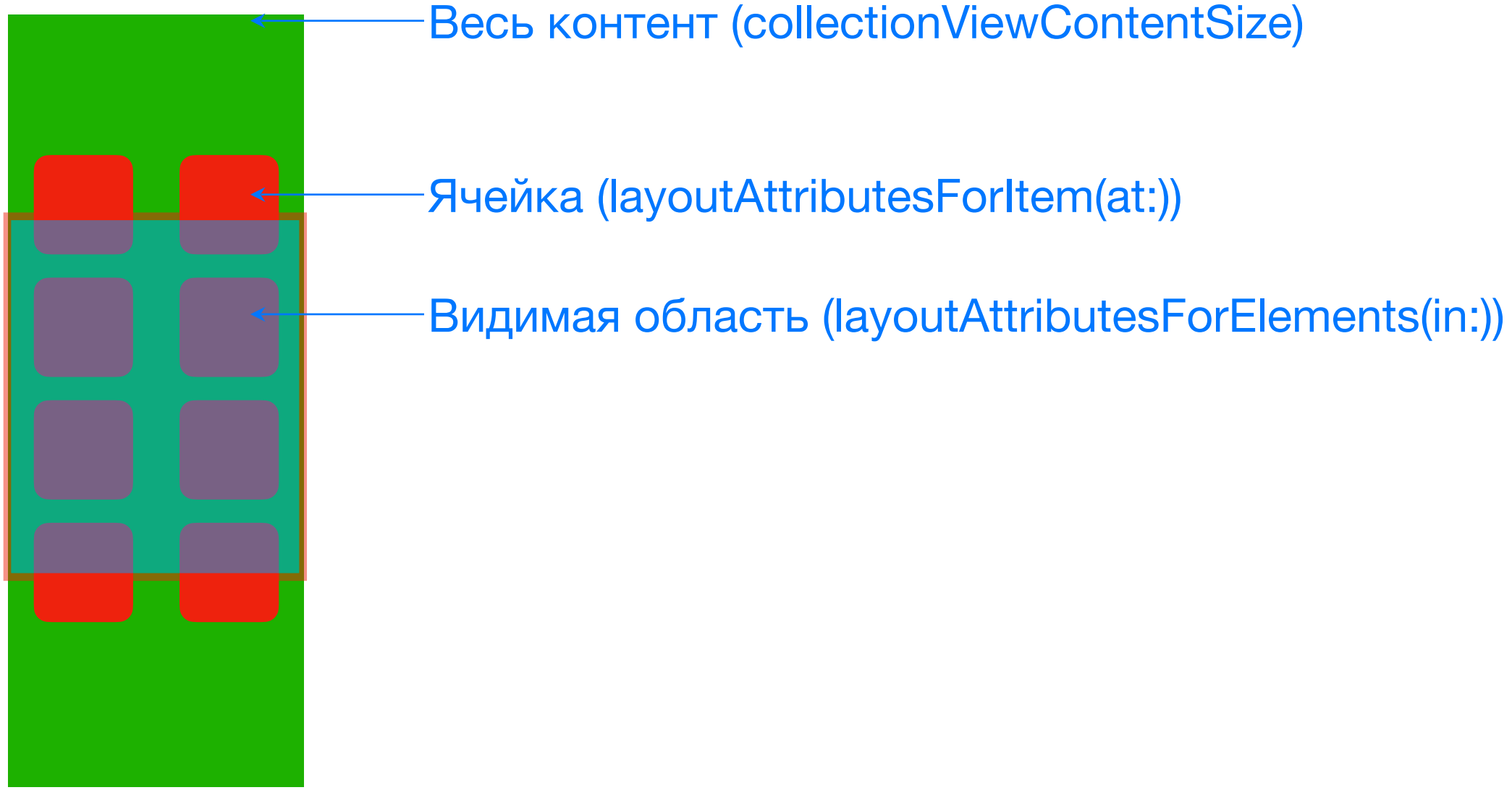


Весь контент (collectionViewContentSize)

Ячейка (layoutAttributesForItem(at:))

Видимая область (layoutAttributesForElements(in:))

UICollectionViewLayout



Размер ячейки фиксирован

Как получить фрейм?

collectionViewContentSize

```swift
override var collectionViewContentSize: CGSize {
    return CGSize(
        width: itemFullSize.width,
        height: itemFullSize.height * Double(numberOfItems)
    )
}
```

layoutAttributesForItem(at:)

```swift
override func layoutAttributesForItem(at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    let itemsPadding = Double(indexPath.item) * itemFullSize
    let attributes = UICollectionViewLayoutAttributes(forCellWith: indexPath)
    attributes.frame = CGRect(
        origin: .init(x: , y: itemsPadding),
        size: itemSize
    )
    return attributes
}
```

layoutAttributesForElements(in:)

```swift
override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    let firstItem = Int(rect.minY) / Int(itemFullSize.height)
    let lastItem = Int(rect.maxY) / Int(itemFullSize.height)

    return (firstItem...lastItem)
        .map { IndexPath(item: $0, section: 0) }
        .compactMap(layoutAttributesForItem)
}
```

layoutAttributesForElements(in:)

```swift
override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    let firstItem = Int(rect.minY) / Int(itemFullSize.height)
    let lastItem = Int(rect.maxY) / Int(itemFullSize.height)

    return (firstItem...lastItem)
        .map { IndexPath(item: $0, section: 0) }
        .compactMap(layoutAttributesForItem)
}
```

layoutAttributesForElements(in:)

```swift
override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    let firstItem = Int(rect.minY) / Int(itemFullSize.height)
    let lastItem = Int(rect.maxY) / Int(itemFullSize.height)

    return (firstItem...lastItem)
        .map { IndexPath(item: $0, section: 0) }
        .compactMap(layoutAttributesForItem)
}
```

layoutAttributesForElements(in:)

```swift
override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    let firstItem = Int(rect.minY) / Int(itemFullSize.height)
    let lastItem = Int(rect.maxY) / Int(itemFullSize.height)

    return (firstItem...lastItem)
        .map { IndexPath(item: $0, section: 0) }
        .compactMap(layoutAttributesForItem)
}
```

layoutAttributesForElements(in:)

```swift
override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    let firstItem = Int(rect.minY) / Int(itemFullSize.height)
    let lastItem = Int(rect.maxY) / Int(itemFullSize.height)

    return (firstItem...lastItem)
        .map { IndexPath(item: $0, section: 0) }
        .compactMap(layoutAttributesForItem)
}
```

layoutAttributesForElements(in:)

```swift
override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    let firstItem = Int(rect.minY) / Int(itemFullSize.height)
    let lastItem = Int(rect.maxY) / Int(itemFullSize.height)

    return (firstItem...lastItem)
        .map { IndexPath(item: $0, section: 0) }
        .compactMap(layoutAttributesForItem)
}
```

Что за UICollectionView LayoutAttributes ?

Frame

Frame

zIndex


1
2
3
Frame
zIndex
Alpha

Frame

zIndex

Alpha

Transform3D

1

Frame

2

zIndex

3

Alpha

4

Transform3D

5

Любые другие параметры для Layout

Что делать, если нельзя рассчитать фрейм ячейки, не зная фрейм предыдущей ячейки?

Пример


15:42
Back
rerum tempore illo omnis dolor quia
est expedita nobis doloremque reprehenderit et nulla ea veritatis
consectetur voluptatem hic sit odit numquam tempora
animi minus ratione dicta temporibus commodi
omnis
inventore hic dolor ab accusamus aspernatur facere
libero id rerum voluptatem commodi molestiae
quidem molestias ullam quibusdam quisquam iure voluptatem ipsa
itaque ut est vitae incidunt cum
ea ut ut
necessitatibus nisi accusamus dolorum qui unde
a consequatur nam
dolorem ut sit ut facere pariatur
voluptatum et iste voluptatem maxime excepturi voluptas facilis

Переопределяем методы так, чтобы считались фреймы на основе предыдущих фреймов

Расчет фреймов за раз

```
func makeAttributes() -> [UICollectionViewLayoutAttributes] {
    var attributesList: [UICollectionViewLayoutAttributes] = []

    for itemIndex in 0..<collectionView.numberOfItems(inSection: sectionIndex) {
        // смотри frame предыдущей ячейки
        // считаем от нее отсуп
        // считаем фрейм текущей ячейки
    }
    return attributesList
}
```

collectionViewContentSize

```swift
override var collectionViewContentSize: CGSize {
    let attributes = makeAttributes()

    let heigth = attributes.last?.frame.maxY ?? 0
    let with = collectionView.frame.width
    return CGSize(width: with, height: heigth)
}
```

collectionViewContentSize

```swift
override var collectionViewContentSize: CGSize {
    let attributes = makeAttributes()

    let heigth = attributes.last?.frame.maxY ?? 0
    let with = collectionView.frame.width
    return CGSize(width: with, height: heigth)
}
```

collectionViewContentSize

```swift
override var collectionViewContentSize: CGSize {
    let attributes = makeAttributes()

    let heigth = attributes.last?.frame.maxY ?? 0
    let with = collectionView.frame.width
    return CGSize(width: with, height: heigth)
}
```

layoutAttributesForItem(at:)

```swift
override func layoutAttributesForItem(at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    let attributes = makeAttributes()
    return attributes[indexPath.item]
}
```

layoutAttributesForElements(in:)

```swift
override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    let attributes = makeAttributes()
    return attributes.filter { $0.frame.intersects(rect) }
}
```

Результат


15:40
Back
rerum tempore illo omnis dolor quia
est expedita nobis doloremque reprehenderit et nulla ea veritatis
consectetur voluptatem hic sit odit numquam tempora
animi minus ratione dicta temporibus commodi
omnis
inventore hic dolor ab accusamus aspernatur facere
libero id rerum voluptatem commodi molestiae
quidem molestias ullam quibusdam quisquam iure voluptatem ipsa
itaque ut est vitae incidunt cum
ea ut ut
necessitatibus nisi accusamus dolorum qui unde
a consequatur nam
dolorem ut sit ut facere pariatur
voluptatum et iste voluptatem maxime excepturi voluptas facilis

Перформанс

Нельзя считать все фреймы на каждый скролл

Встречайте prepare

1

Вызывается на
каждое изменение
Layout

1

Вызывается на каждое изменение Layout

2

По умолчанию ничего не делает

Идеальное место для того, чтобы считать атрибуты

prepare

```swift
var attributesList: [UICollectionViewLayoutAttributes] = []

override func prepare() {
    super.prepare()

    attributesList.removeAll(keepingCapacity: true)

    for itemIndex in 0..<collectionView.numberOfItems(inSection: sectionIndex) {
        // смотрим на frame предыдущей ячейки
        // считаем от нее отступ
        // считаем фрейм текущей ячейки
    }

}
```

prepare

```swift
var attributesList: [UICollectionViewLayoutAttributes] = []

override func prepare() {
    super.prepare()

    attributesList.removeAll(keepingCapacity: true)

    for itemIndex in 0..<collectionView.numberOfItems(inSection: sectionIndex) {
        // смотрим на frame предыдущей ячейки
        // считаем от нее отступ
        // считаем фрейм текущей ячейки
    }

}
```

prepare

```swift
var attributesList: [UICollectionViewLayoutAttributes] = []

override func prepare() {
    super.prepare()

    attributesList.removeAll(keepingCapacity: true)

    for itemIndex in 0..<collectionView.numberOfItems(inSection: sectionIndex) {
        // смотрим на frame предыдущей ячейки
        // считаем от нее отступ
        // считаем фрейм текущей ячейки
    }

}
```

prepare

```swift
var attributesList: [UICollectionViewLayoutAttributes] = []

override func prepare() {
    super.prepare()

    attributesList.removeAll(keepingCapacity: true)

    for itemIndex in 0..<collectionView.numberOfItems(inSection: sectionIndex) {
        // смотрим на frame предыдущей ячейки
        // считаем от нее отступ
        // считаем фрейм текущей ячейки
    }
}
```

collectionViewContentSize

```swift
override var collectionViewContentSize: CGSize {
    let heigth = attributesList.last?.frame.maxY ?? 0
    let with = collectionView.frame.width
    return CGSize(width: with, height: heigth)
}
```

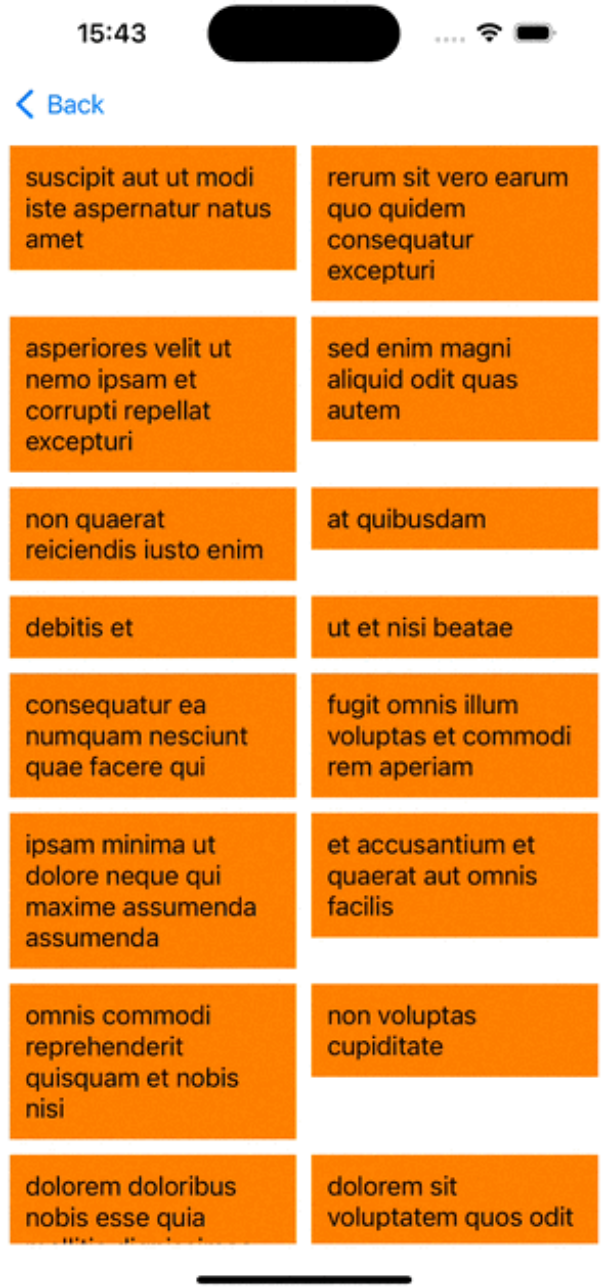
layoutAttributesForItem(at:)

```swift
override func layoutAttributesForItem(at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    return attributesList[indexPath.item]
}
```

layoutAttributesForElements(in:)

```
 override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    return attributesList.filter { $0.frame.intersects(rect) }
}
```

Результат


15:43
Back
suscipit aut ut modi iste aspernatur natus amet
rerum sit vero earum quo quidem consequatur excepturi
asperiores velit ut nemo ipsam et corrupti repellat excepturi
sed enim magni aliquid odit quas autem
non quaerat reiciendis iusto enim
at quibusdam
debitis et
ut et nisi beatae
consequatur ea numquam nesciunt quae facere qui
fugit omnis illum voluptas et commodi rem aperiam
ipsam minima ut dolore neque qui maxime assumenda assumenda
et accusantium et quaerat aut omnis facilis
omnis commodi reprehenderit quisquam et nobis nisi
non voluptas cupiditate
dolorem doloribus nobis esse quia
dolorem sit voluptatem quos odit

Prepare решает проблемы?

Продолжаем разбираться

Добавим фон для секций

AttributesForDecorationView

```swift
var decorationsList: [UICollectionViewLayoutAttributes] = []

func prepare() {
  // Считаем атрибуты для фона
}

override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    return attributesList.filter { $0.frame.intersects(rect) } + decorationsList.filter { $0.frame.intersects(rect) }
}

override func layoutAttributesForDecorationView(ofKind elementKind: String, at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    return decorationsList[indexPath.item]
}
```

AttributesForDecorationView

```swift
var decorationsList: [UICollectionViewLayoutAttributes] = []

func prepare() {
  // Считаем атрибуты для фона
}

override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    return attributesList.filter { $0.frame.intersects(rect) } + decorationsList.filter { $0.frame.intersects(rect) }
}

override func layoutAttributesForDecorationView(ofKind elementKind: String, at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    return decorationsList[indexPath.item]
}
```

AttributesForDecorationView

```swift
var decorationsList: [UICollectionViewLayoutAttributes] = []

func prepare() {
  // Считаем атрибуты для фона
}

override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    return attributesList.filter { $0.frame.intersects(rect) } + decorationsList.filter { $0.frame.intersects(rect) }
}

override func layoutAttributesForDecorationView(ofKind elementKind: String, at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    return decorationsList[indexPath.item]
}
```

AttributesForDecorationView

```swift
var decorationsList: [UICollectionViewLayoutAttributes] = []

func prepare() {
  // Считаем атрибуты для фона
}

override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    return attributesList.filter { $0.frame.intersects(rect) } + decorationsList.filter { $0.frame.intersects(rect) }
}

override func layoutAttributesForDecorationView(ofKind elementKind: String, at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    return decorationsList[indexPath.item]
}
```

Результат


15:45
Back
quibusdam blanditiis error a voluptatem minus ullam soluta qui
inventore sed atque unde porro itaque
ducimus voluptatem explicabo ex fuga impedit
placeat quibusdam odit excepturi voluptates quo sed placeat
architecto quidem
voluptatibus id et et qui
sit
voluptas et consequatur ipsam deleniti sit esse
est exercitationem in aut
dignissimos
aut
reiciendis alias perspiciatis veniam fugiat ab numquam eveniet adipisci
omnis asperiores
quam quo

Добавим плавающий заголовок

AttributesForSupplementaryView

```swift
var supplementariesList: [UICollectionViewLayoutAttributes] = []

override func prepare() {
    // считаем атрибуты для плавающего заголовка
    // смещаем его так, что бы он смещался (плавал
}

override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    attributesList.filter { $0.frame.intersects(rect) }
        + decorationsList.filter { $0.frame.intersects(rect) }
        + supplementariesList.filter { $0.frame.intersects(rect) }
}

override func layoutAttributesForSupplementaryView(ofKind elementKind: String, at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    return supplementariesList[indexPath.item]
}
```

AttributesForSupplementaryView

```swift
var supplementariesList: [UICollectionViewLayoutAttributes] = []

override func prepare() {
    // считаем атрибуты для плавающего заголовка
    // смещаем его так, что бы он смещался (плавал
}

override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    attributesList.filter { $0.frame.intersects(rect) }
        + decorationsList.filter { $0.frame.intersects(rect) }
        + supplementariesList.filter { $0.frame.intersects(rect) }
}

override func layoutAttributesForSupplementaryView(ofKind elementKind: String, at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    return supplementariesList[indexPath.item]
}
```

AttributesForSupplementaryView

```
var supplementariesList: [UICollectionViewLayoutAttributes] = []

override func prepare() {
    // считаем атрибуты для плавающего заголовка
    // смещаем его так, что бы он смещался (плавал
}

override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    attributesList.filter { $0.frame.intersects(rect) }
        + decorationsList.filter { $0.frame.intersects(rect) }
        + supplementariesList.filter { $0.frame.intersects(rect) }
}

override func layoutAttributesForSupplementaryView(ofKind elementKind: String, at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    return supplementariesList[indexPath.item]
}
```

AttributesForSupplementaryView

```swift
var supplementariesList: [UICollectionViewLayoutAttributes] = []

override func prepare() {
    // считаем атрибуты для плавающего заголовка
    // смещаем его так, что бы он смещался (плавал
}

override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    attributesList.filter { $0.frame.intersects(rect) }
        + decorationsList.filter { $0.frame.intersects(rect) }
        + supplementariesList.filter { $0.frame.intersects(rect) }
}

override func layoutAttributesForSupplementaryView(ofKind elementKind: String, at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    return supplementariesList[indexPath.item]
}
```

AttributesForSupplementaryView

```swift
var supplementariesList: [UICollectionViewLayoutAttributes] = []

override func prepare() {
    // считаем атрибуты для плавающего заголовка
    // смещаем его так, что бы он смещался (плавал
}

override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    attributesList.filter { $0.frame.intersects(rect) }
        + decorationsList.filter { $0.frame.intersects(rect) }
        + supplementariesList.filter { $0.frame.intersects(rect) }
}

override func layoutAttributesForSupplementaryView(ofKind elementKind: String, at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    return supplementariesList[indexPath.item]
}
```

Результат


15:47
Back
Заголовок
enim reiciendis aliquid sint sequi qui dignissimos libero
recusandae non sequi voluptas voluptatibus
provident et quisquam quaerat
tempora sit iure fugiat et quibusdam ut
cumque quia consequatur saepe
dolores cum est
quo laborum molestias quas ullam sint corrupti doloribus assumenda
rerum minus
et dolor amet facilis
soluta
quos labore
a tempore iure nihil porro voluptatem

Заголовок есть, но не плавает

prepare
вызывается только
на обновление
layout

Нужно обновлять при скролле

Сброс layout при скролле

```swift
override func shouldInvalidateLayout(forBoundsChange newBounds: CGRect) -> Bool {
    true
}
```

Результат


15:48
Back
Заголовок
et sint dolores quibusdam
eius esse ut doloribus sequi ea
ut minus maxime doloribus dolores sed
similique similique dolor quia vitae ut recusandae inventore
qui error aut doloremque nemo nihil
consectetur repellat aut commodi expedita sint libero delectus sapiente
fugit facere atque velit ducimus voluptatem incidunt molestiae
deleniti facere totam
ex cumque dolore
et magnam ut error qui
nam accusamus
odio

Заголовок плавает

Перформанс тоже плавает

Почему перформанс 💩 ?

prepare
вызывается
постоянно

Все атрибуты постоянно пересчитываются

Как этого избежать?

Смотрим в доку

Optimizing layout performance using invalidation contexts

When designing your custom layouts, you can improve performance by invalidating only those parts of your layout that actually changed. When you change items, calling the `invalidateLayout()` method forces the collection view to recompute all of its layout information and reapply it. A better solution is to recompute only the layout information that changed, which is exactly what invalidation contexts allow you to do. An invalidation context lets you specify which parts of the layout changed. The layout object can then use that information to minimize the amount of data it recomputes.

To define a custom invalidation context for your layout, subclass the `UICollectionViewLayoutInvalidationContext` class. In your subclass, define custom properties that represent the parts of your layout data that can be recomputed independently. When you need to invalidate your layout at runtime, create an instance of your invalidation context subclass, configure the custom properties based on what layout information changed, and pass that object to your layout's `invalidateLayout(with:)` method. Your custom implementation of that method can use the information in the invalidation context to recompute only the portions of your layout that changed.

If you define a custom invalidation context class for your layout object, you should also override the `invalidationContextClass` method and return your custom class. The collection view always creates an instance of the class you specify when it needs an invalidation context. Returning your custom subclass from this method ensures that your layout object always has the invalidation context it expects.

Смотрим в доку

Optimizing layout performance using invalidation contexts

When designing your custom layouts, you can improve performance by invalidating only those parts of your layout that actually changed. When you change items, calling the `invalidateLayout()` method forces the collection view to

Apple подтверждает, считать все каждый раз не надо

UICollectionView LayoutInvalidation Context

Смотрим в доку

`var invalidateEverything: Bool`

A Boolean that indicates that all layout data should be marked as invalid.

`var invalidateDataSourceCounts: Bool`

A Boolean that indicates whether the layout should ask for new section and item counts.

Invalidating the Content Area

`var contentOffsetAdjustment: CGPoint`

The delta value to be applied to the collection view's content offset.

`var contentSizeAdjustment: CGSize`

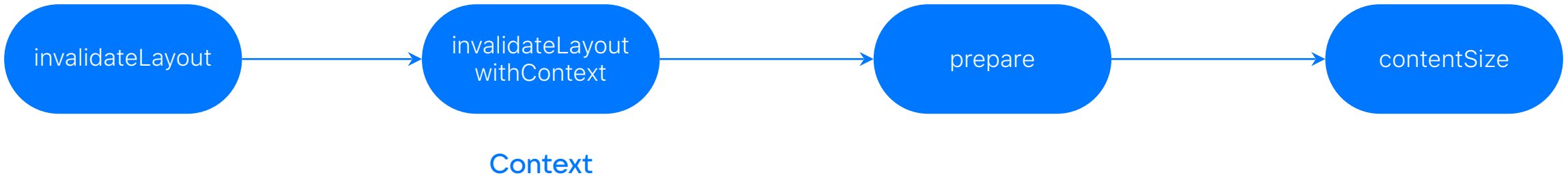
The delta value to be applied to the collection view's content size.

Как использовать контекст в prepare?

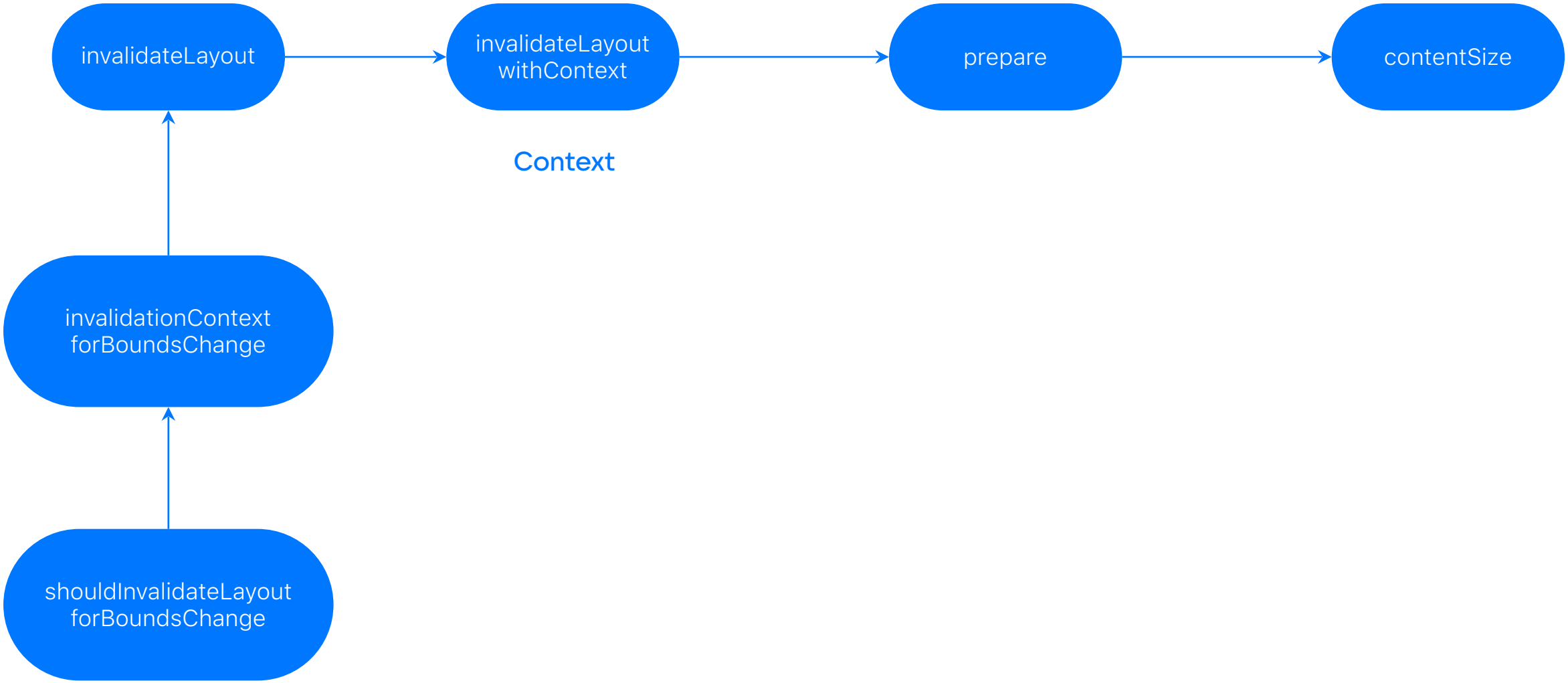
Никак

Как работает Layout?

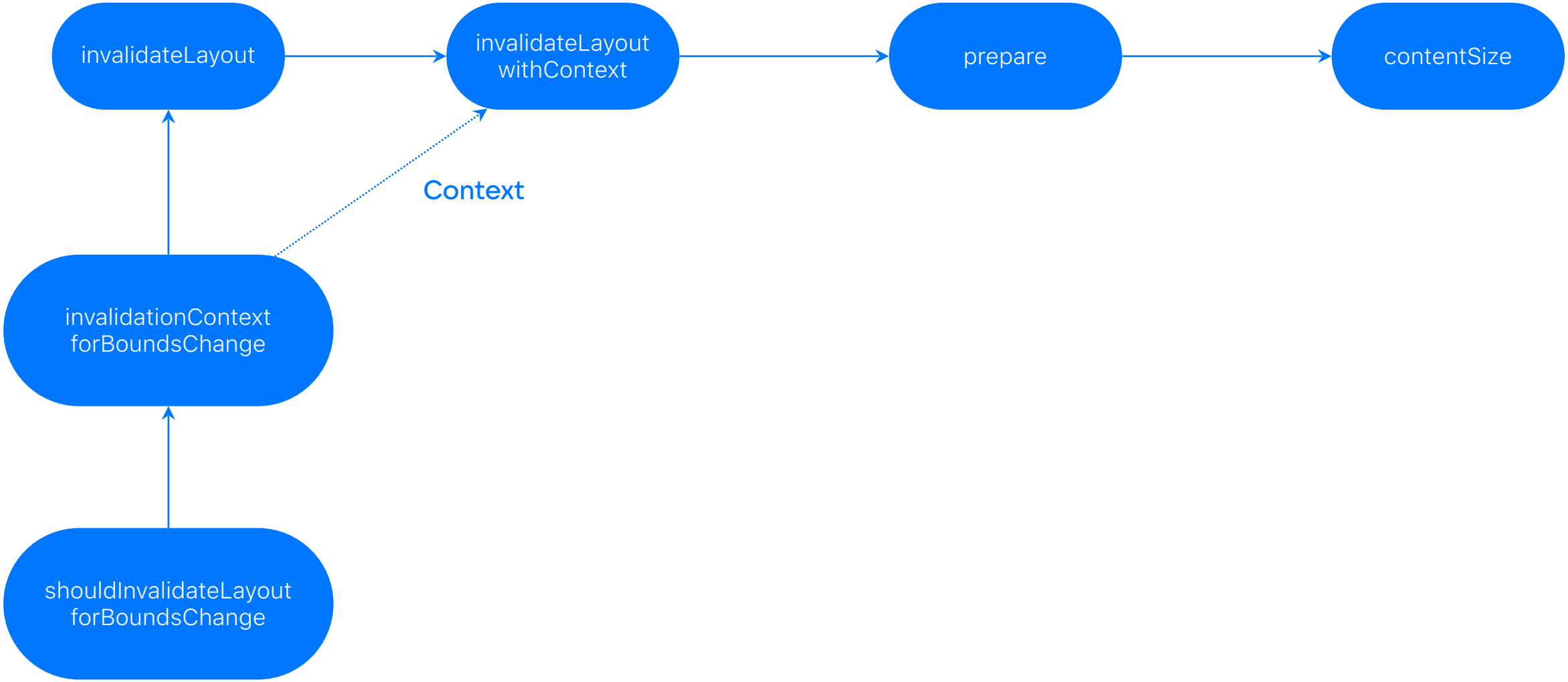
ReloadData


invalidateLayout
invalidateLayout withContext
prepare
contentSize
Context

Scroll


invalidateLayout
invalidateLayout withContext
prepare
contentSize
Context
invalidationContext forBoundsChange
shouldInvalidateLayout forBoundsChange

Scroll


invalidateLayout
invalidateLayout
withContext
prepare
contentSize
Context
invalidationContext
forBoundsChange
shouldInvalidateLayout
forBoundsChange

Заводим свой контекст

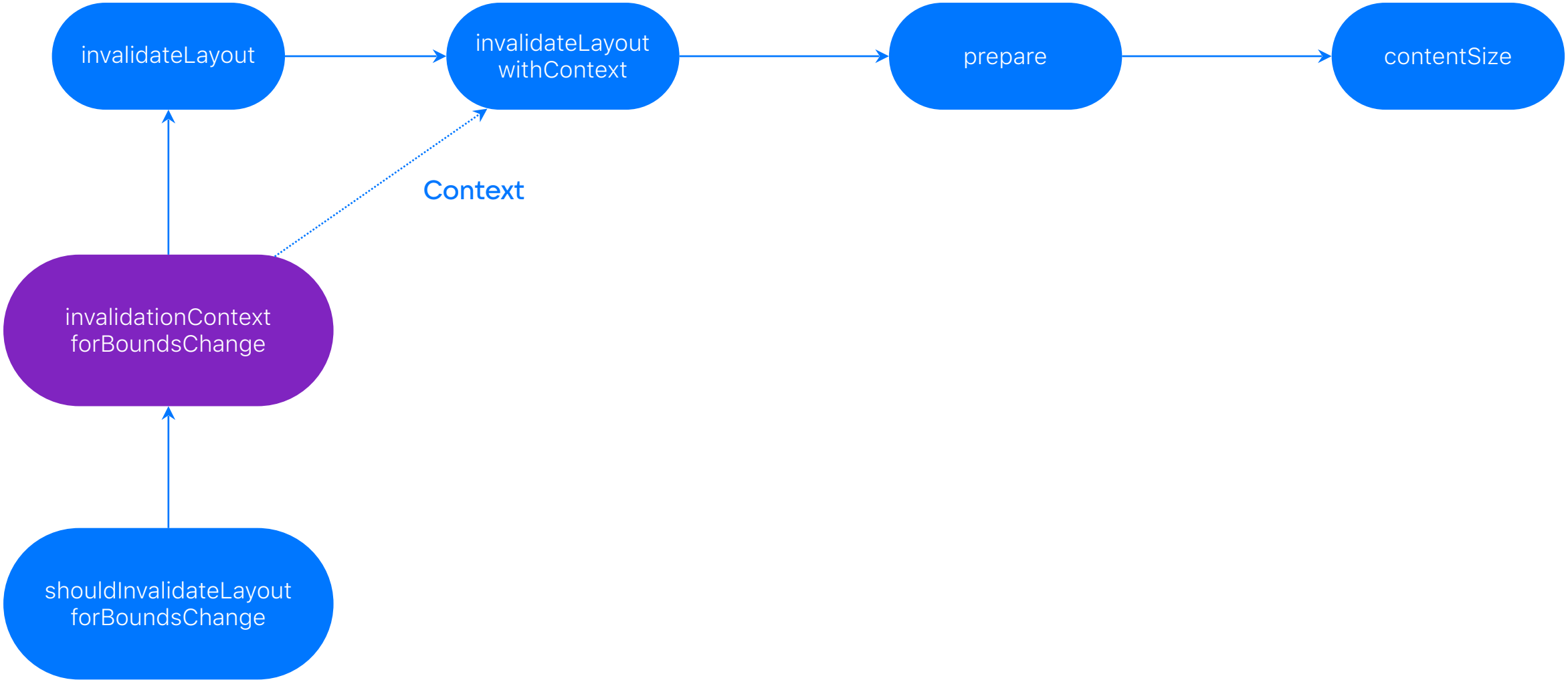
CustomInvalidationContext

```
private class CustomInvalidationContext: UICollectionViewLayoutInvalidationContext {
    var invalidateLayoutMetrics = true
}
```

CustomInvalidationContext

```
override class var invalidationContextClass: AnyClass {
  CustomInvalidationContext.self
}
```

Scroll


invalidateLayout
invalidateLayout withContext
prepare
contentSize
Context
invalidationContext forBoundsChange
shouldInvalidateLayout forBoundsChange

invalidationContext(forBoundsChange:)

```swift
override func invalidationContext(forBoundsChange newBounds: CGRect) -> UICollectionViewLayoutInvalidationContext {
    let defaultContext = super.invalidationContext(forBoundsChange: newBounds)

    guard let invalidationContext = defaultContext as? CustomInvalidationContext else {
        fatalError("`context` must be an instance of `CustomInvalidationContext`")
    }

    invalidationContext.invalidateLayoutMetrics = false

    return invalidationContext
}
```

invalidationContext(forBoundsChange:)

```
override func invalidationContext(forBoundsChange newBounds: CGRect) -> UICollectionViewLayoutInvalidationContext {
    let defaultContext = super.invalidationContext(forBoundsChange: newBounds)

    guard let invalidationContext = defaultContext as? CustomInvalidationContext else {
        fatalError("`context` must be an instance of `CustomInvalidationContext`")
    }

    invalidationContext.invalidateLayoutMetrics = false

    return invalidationContext
}
```

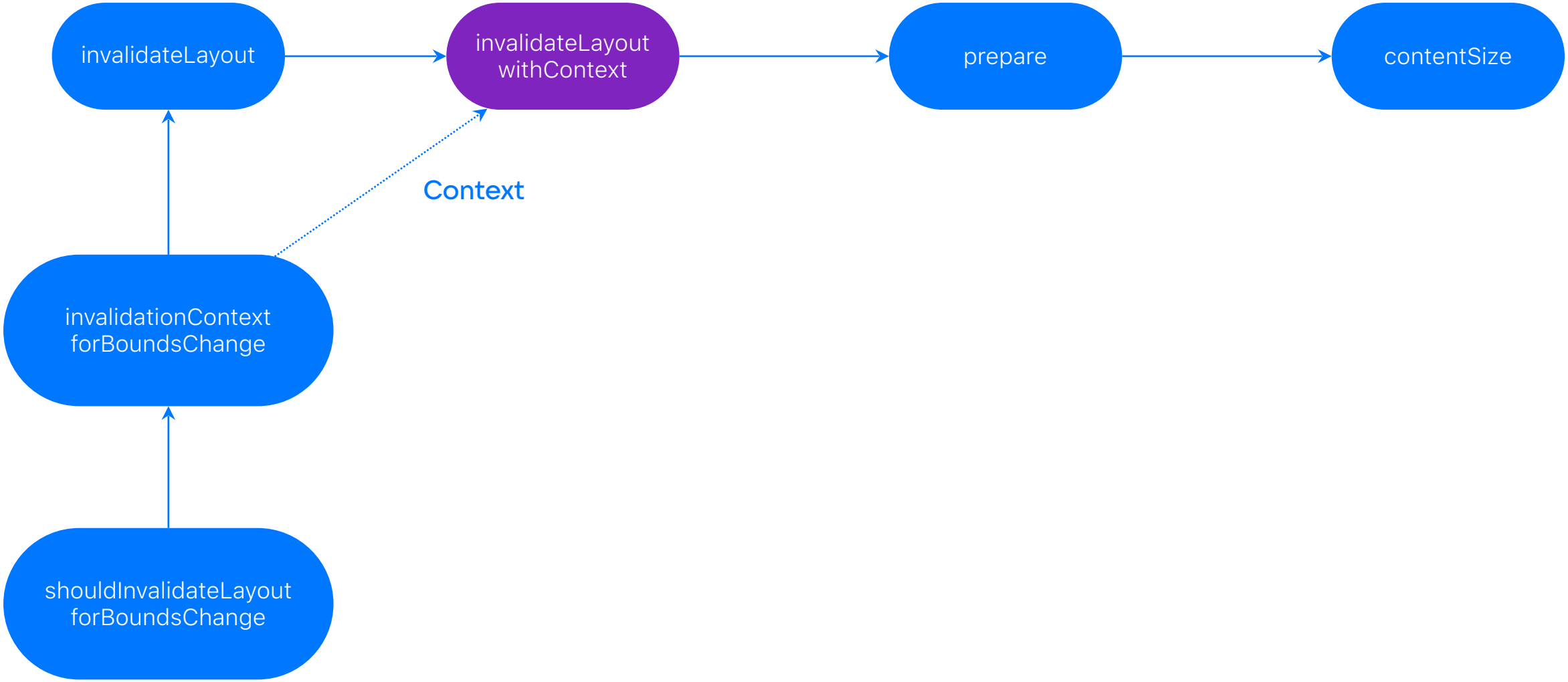
invalidationContext(forBoundsChange:)

```swift
override func invalidationContext(forBoundsChange newBounds: CGRect) -> UICollectionViewLayoutInvalidationContext {
    let defaultContext = super.invalidationContext(forBoundsChange: newBounds)

    guard let invalidationContext = defaultContext as? CustomInvalidationContext else {
        fatalError("`context` must be an instance of `CustomInvalidationContext`")
    }

    invalidationContext.invalidateLayoutMetrics = false

    return invalidationContext
}
```

invalidationContext(forBoundsChange:)

```swift
override func invalidationContext(forBoundsChange newBounds: CGRect) -> UICollectionViewLayoutInvalidationContext {
    let defaultContext = super.invalidationContext(forBoundsChange: newBounds)

    guard let invalidationContext = defaultContext as? CustomInvalidationContext else {
        fatalError("`context` must be an instance of `CustomInvalidationContext`")
    }

    invalidationContext.invalidateLayoutMetrics = false

    return invalidationContext
}
```

Scroll


invalidateLayout
invalidateLayout
withContext
prepare
contentSize
Context
invalidationContext
forBoundsChange
shouldInvalidateLayout
forBoundsChange

invalidateLayout(with:)

```swift
override func invalidateLayout(with context: UICollectionViewLayoutInvalidationContext) {
    guard let context = context as? CustomInvalidationContext else {
        fatalError("`context` must be an instance of `CustomInvalidationContext`")
    }

    let invalidateFromOurContext = context.invalidateLayoutMetrics && !context.invalidateEverything && !context.invalidateDataSourceCounts
    let invalidateFromContext = context.invalidateEverything

    if invalidateFromOurContext || invalidateFromContext {

        attributesList.removeAll()
        decorationsList.removeAll()
        supplementariesList.removeAll()

    }

    super.invalidateLayout(with: context)
}
```

invalidateLayout(with:)

```
override func invalidateLayout(with context: UICollectionViewLayoutInvalidationContext) {
    guard let context = context as? CustomInvalidationContext else {
        fatalError("`context` must be an instance of `CustomInvalidationContext`")
    }

    let invalidateFromOurContext = context.invalidateLayoutMetrics && !context.invalidateEverything && !context.invalidateDataSourceCounts
    let invalidateFromContext = context.invalidateEverything

    if invalidateFromOurContext || invalidateFromContext {

        attributesList.removeAll()
        decorationsList.removeAll()
        supplementariesList.removeAll()

    }

    super.invalidateLayout(with: context)
}
```

invalidateLayout(with:)

```
override func invalidateLayout(with context: UICollectionViewLayoutInvalidationContext) {
    guard let context = context as? CustomInvalidationContext else {
        fatalError("`context` must be an instance of `CustomInvalidationContext`")
    }

    let invalidateFromOurContext = context.invalidateLayoutMetrics && !context.invalidateEverything && !context.invalidateDataSourceCounts
    let invalidateFromContext = context.invalidateEverything

    if invalidateFromOurContext || invalidateFromContext {

        attributesList.removeAll()
        decorationsList.removeAll()
        supplementariesList.removeAll()

    }

    super.invalidateLayout(with: context)
}
```

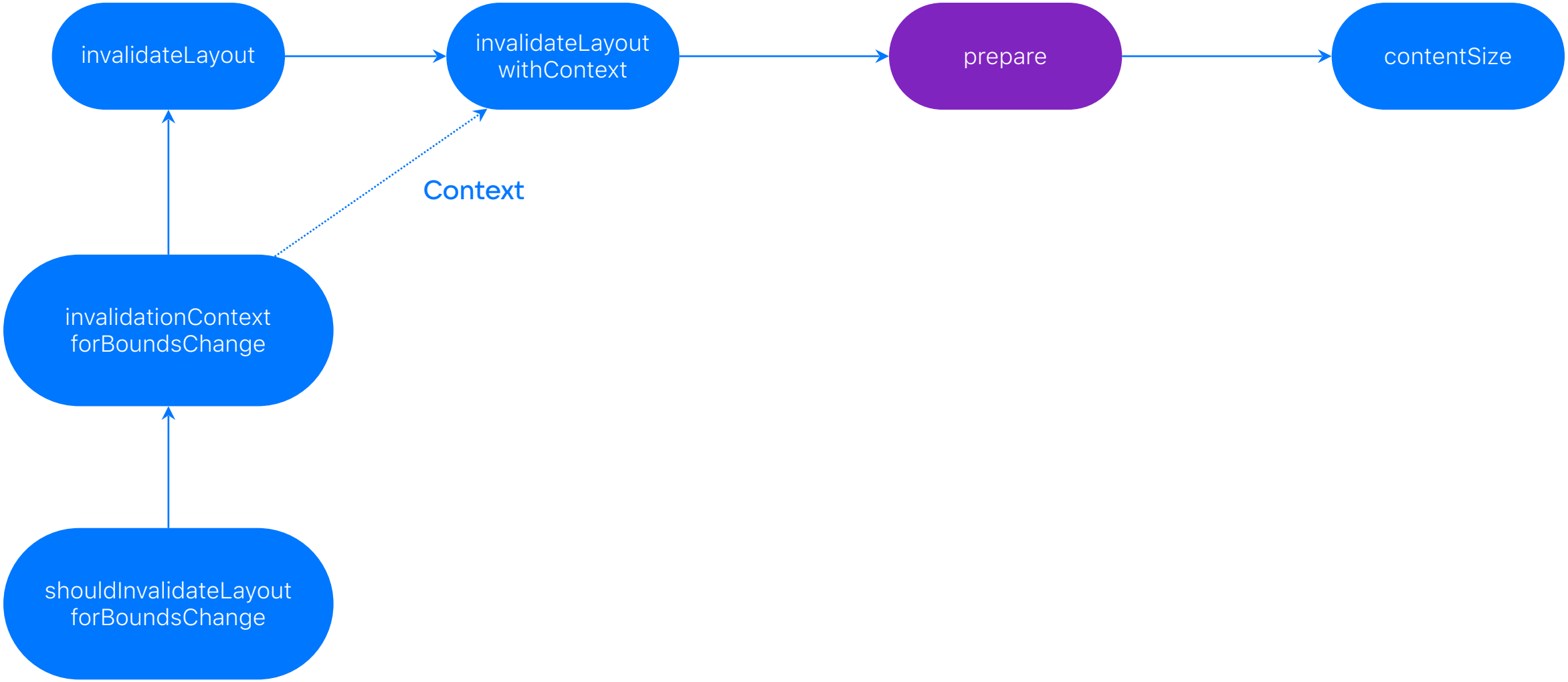
invalidateLayout(with:)

```
override func invalidateLayout(with context: UICollectionViewLayoutInvalidationContext) {
    guard let context = context as? CustomInvalidationContext else {
        fatalError("`context` must be an instance of `CustomInvalidationContext`")
    }

    let invalidateFromOurContext = context.invalidateLayoutMetrics && !context.invalidateEverything && !context.invalidateDataSourceCounts
    let invalidateFromContext = context.invalidateEverything

    if invalidateFromOurContext || invalidateFromContext {

        attributesList.removeAll()
        decorationsList.removeAll()
        supplementariesList.removeAll()

    }

    super.invalidateLayout(with: context)
}
```

invalidateLayout(with:)

```
override func invalidateLayout(with context: UICollectionViewLayoutInvalidationContext) {
    guard let context = context as? CustomInvalidationContext else {
        fatalError("`context` must be an instance of `CustomInvalidationContext`")
    }

    let invalidateFromOurContext = context.invalidateLayoutMetrics && !context.invalidateEverything && !context.invalidateDataSourceCounts
    let invalidateFromContext = context.invalidateEverything

    if invalidateFromOurContext || invalidateFromContext {
        needRebuildLayout = true
    }

    hasDataSourceCountInvalidationBeforeReceivingUpdateItems = context.invalidateDataSourceCounts && !context.invalidateEverything

    super.invalidateLayout(with: context)
}
```

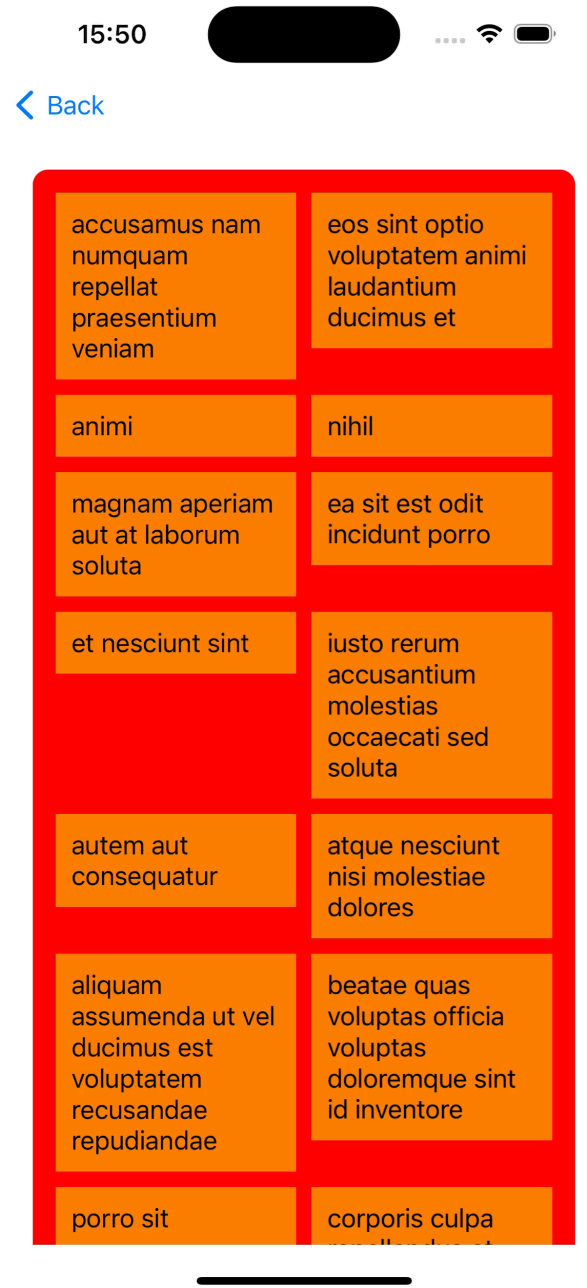
Scroll


invalidateLayout
invalidateLayout
withContext
prepare
contentSize
Context
invalidationContext
forBoundsChange
shouldInvalidateLayout
forBoundsChange

invalidateLayout(with:)

```
override func prepare() {
    super.prepare()

    if needRebuildLayout {
        // обновляем атрибуты
    }

    // обновляем позицию заголовка

    needRebuildLayout = false
}
```

Результат


15:50
Back
accusamus nam numquam repellat praesentium veniam
eos sint optio voluptatem animi laudantium ducimus et
animi
nihil
magnam aperiam aut at laborum soluta
ea sit est odit incidunt porro
et nesciunt sint
iusto rerum accusantium molestias occaecati sed soluta
autem aut consequatur
atque nesciunt nisi molestiae dolores
aliquam assumenda ut vel ducimus est voluptatem recusandae repudiandae
beatae quas voluptas officia voluptas doloremque sint id inventore
porro sit
corporis culpa

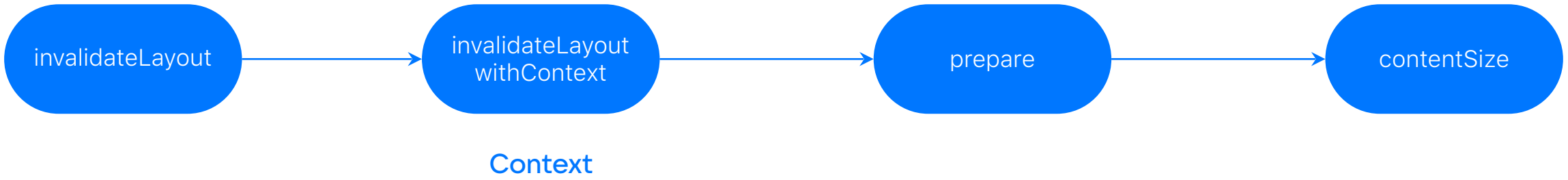
Заголовок плавает

Перформанс стабилен

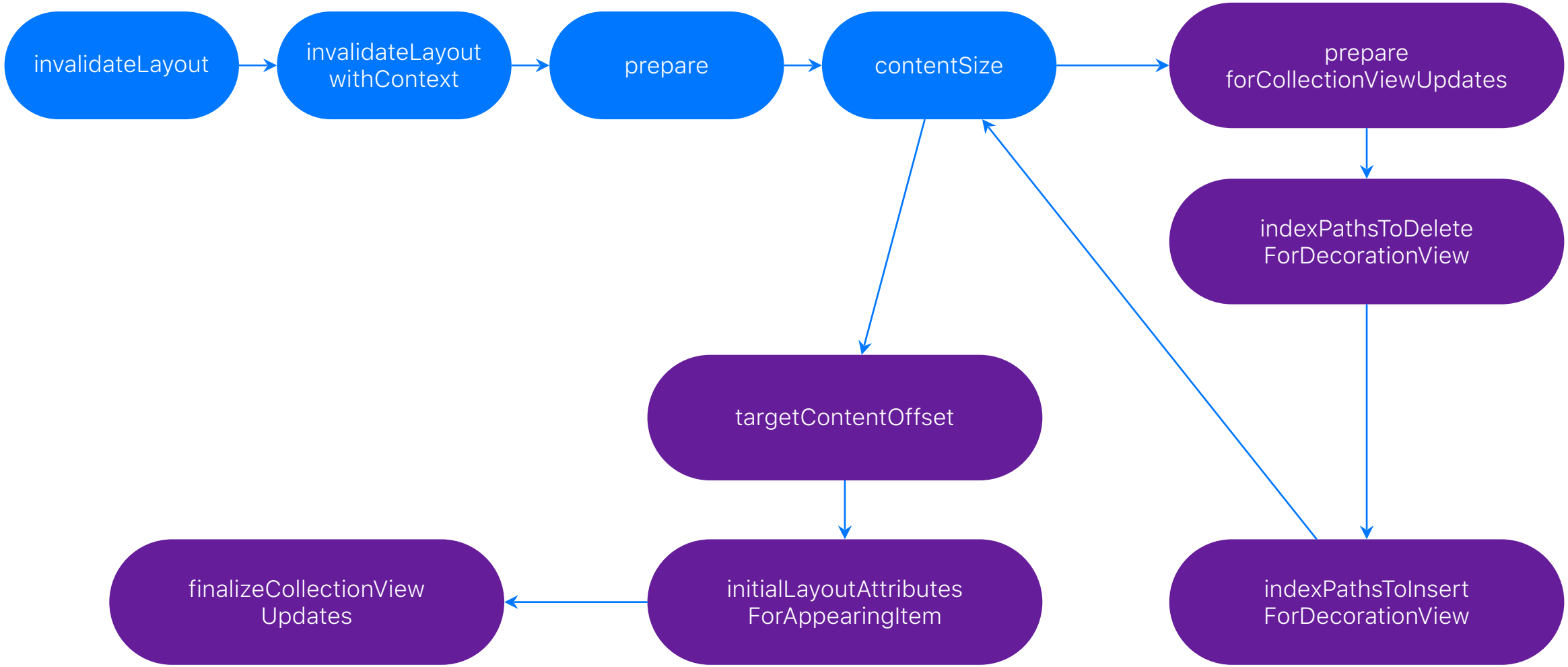
Что будет, если мы вставим или удалим элемент?

Все пересчитает заново

ReloadData


invalidateLayout
invalidateLayout withContext
prepare
contentSize
Context

Update


invalidateLayout
invalidateLayout withContext
prepare
contentSize
prepare forCollectionViewUpdates
indexPathsToDelete ForDecorationView
targetContentOffset
finalizeCollectionView Updates
initialLayoutAttributes ForAppearingItem
indexPathsToInsert ForDecorationView

Как пересчитать только то, что нужно?

Поможет ли контекст?

Смотрим в доку

```
var invalidatedItemIndexPaths: [IndexPath]?
```

An array of index paths representing the cells that were invalidated.

```
var invalidatedSupplementaryIndexPaths: [String : [IndexPath]]?
```

A dictionary that identifies the supplementary views that were invalidated.

```
var invalidatedDecorationIndexPaths: [String : [IndexPath]]?
```

A dictionary that identifies the decoration views that were invalidated.

Информация есть

Но ее нет

Смотрим в доку

```
var invalidatedItemIndexPaths: [IndexPath]?
```

An array of index paths representing the cells that were invalidated.

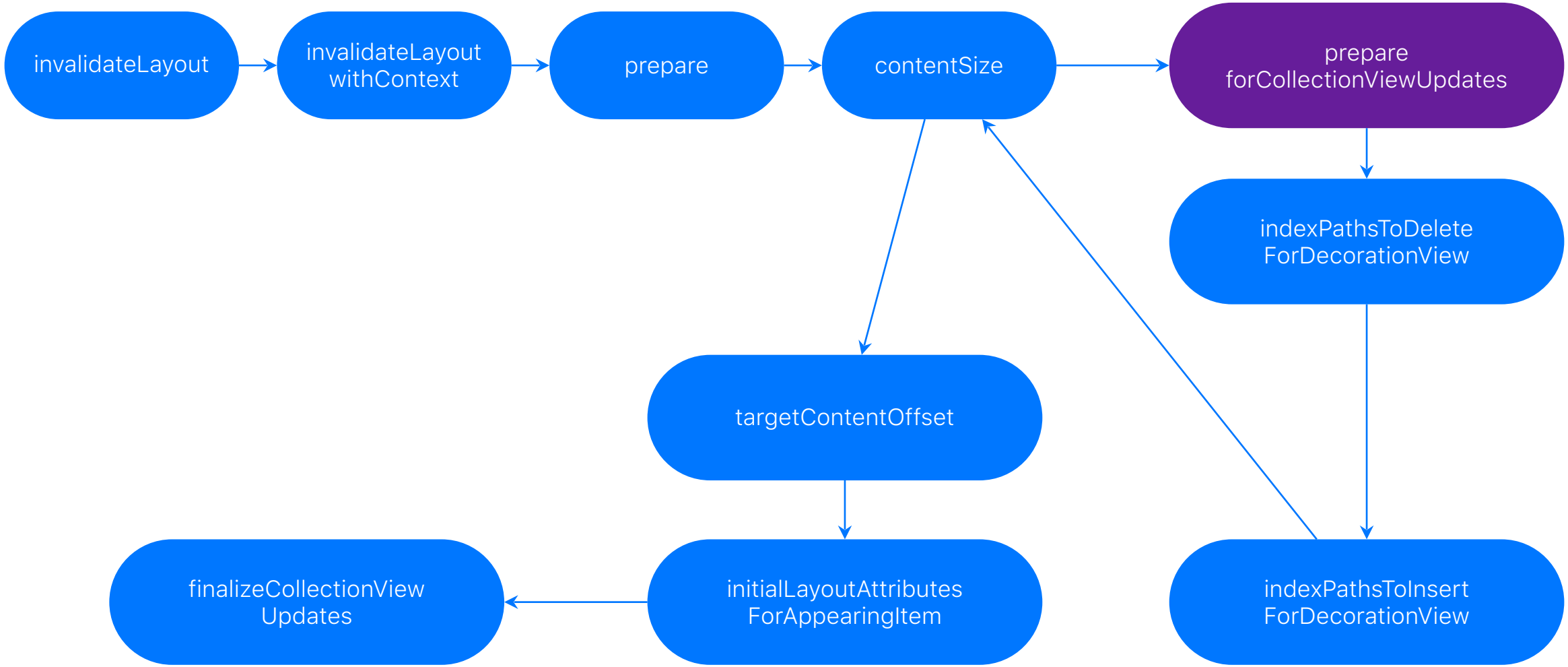
```
var invalidatedSupplementaryIndexPaths: [String : [IndexPath]]?
```

A dictionary that identifies the supplementary views that were invalidated.

```
var invalidatedDecorationIndexPaths: [String : [IndexPath]]?
```

A dictionary that identifies the decoration views that were invalidated.

Update


invalidateLayout
invalidateLayout withContext
prepare
contentSize
prepare forCollectionViewUpdates
indexPathsToDelete ForDecorationView
targetContentOffset
finalizeCollectionView Updates
initialLayoutAttributes ForAppearingItem
indexPathsToInsert ForDecorationView

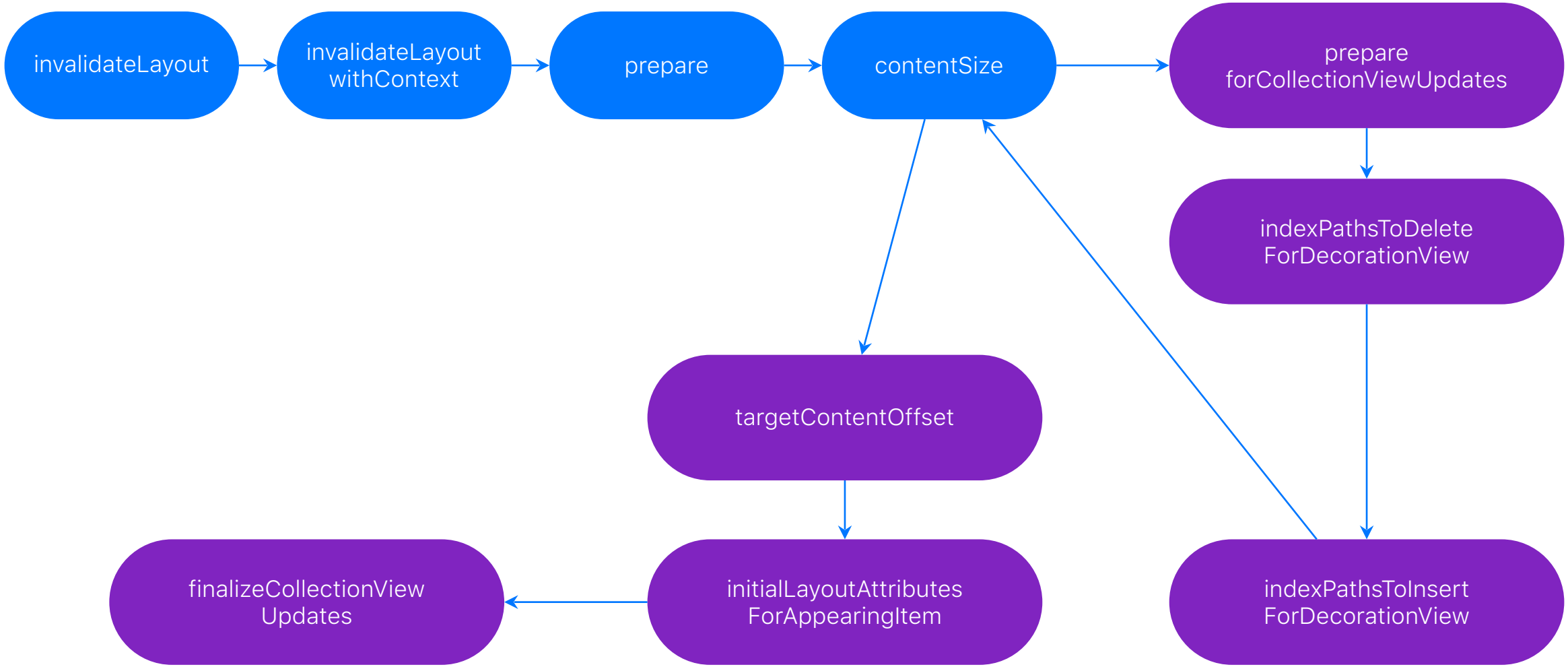
prepare(forCollectionViewUpdates:)

```swift
override func prepare(forCollectionViewUpdates updateItems: [UICollectionViewUpdateItem]) {
  // обновляем атрибуты только для изменившися элементов

  super.prepare(forCollectionViewUpdates: updateItems)
}
```

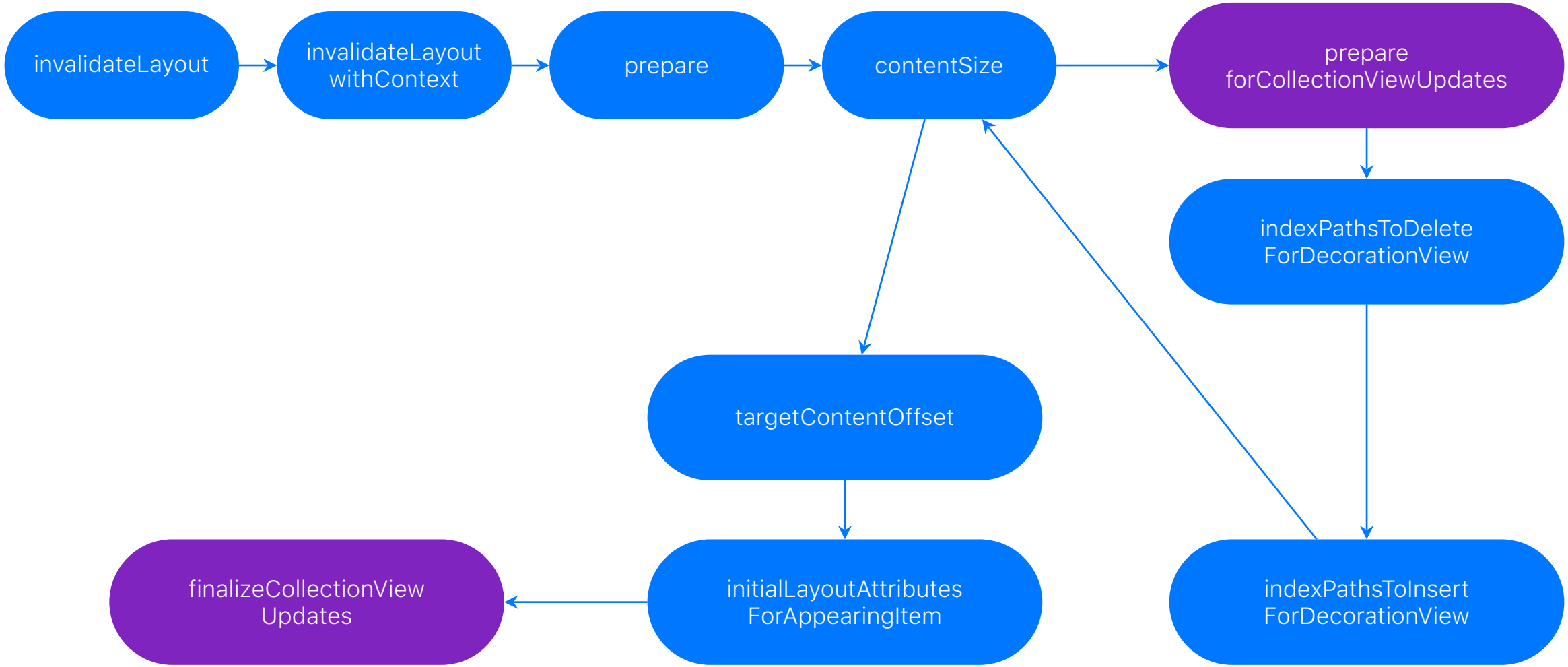
Все!

Не все!

Update


invalidateLayout
invalidateLayout withContext
prepare
contentSize
prepare forCollectionViewUpdates
indexPathsToDelete ForDecorationView
targetContentOffset
finalizeCollectionView Updates
initialLayoutAttributes ForAppearingItem
indexPathsToInsert ForDecorationView

Update


invalidateLayout
invalidateLayout withContext
prepare
contentSize
prepare forCollectionViewUpdates
indexPathsToDelete ForDecorationView
targetContentOffset
finalizeCollectionView Updates
initialLayoutAttributes ForAppearingItem
indexPathsToInsert ForDecorationView

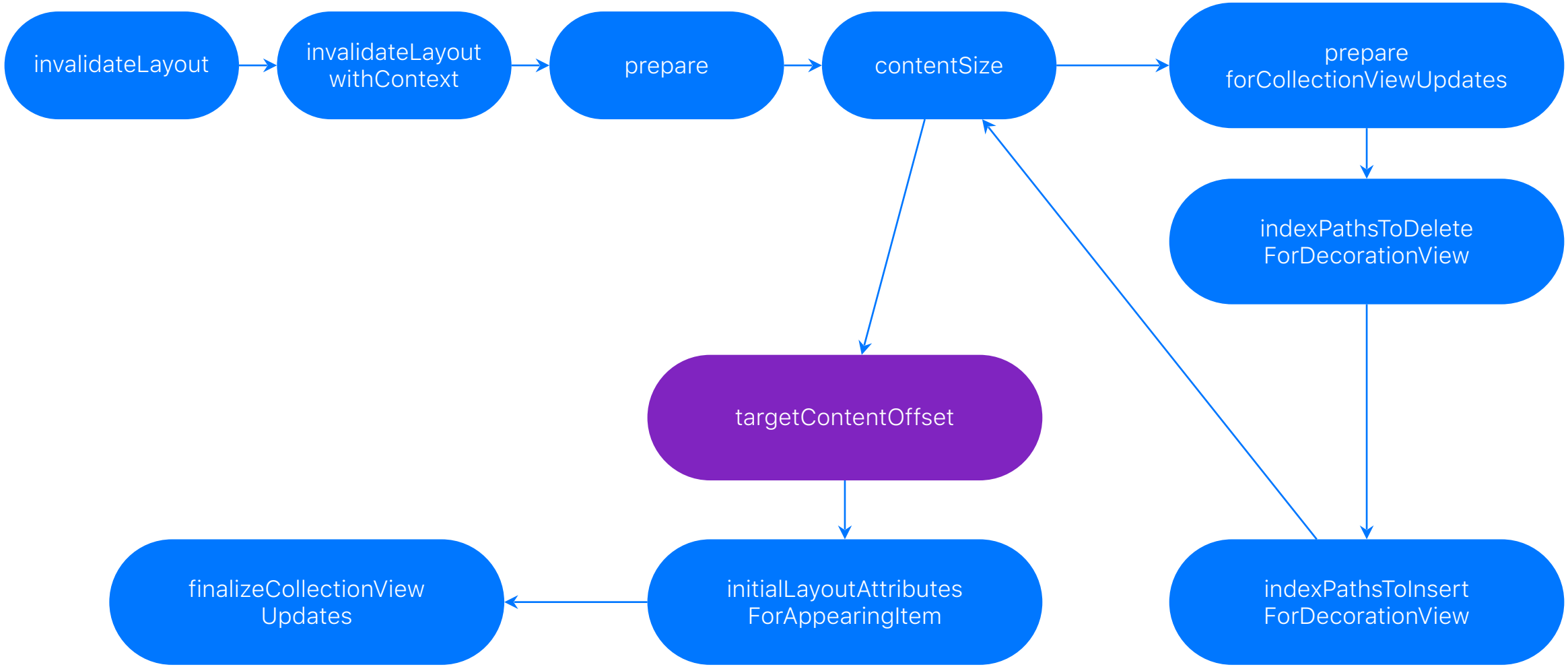
Мы должны помнить изменения

Например что мы меняем

```
var itemIndexPathsToInsert = Set<IndexPath>()
var itemIndexPathsToDelete = Set<IndexPath>()
```

Теперь почти все

Scroll


invalidateLayout
invalidateLayout withContext
prepare
contentSize
prepare forCollectionViewUpdates
indexPathsToDelete ForDecorationView
targetContentOffset
finalizeCollectionView Updates
initialLayoutAttributes ForAppearingItem
indexPathsToInsert ForDecorationView

initialLayoutAttributesForAppearingItem

```
override func targetContentOffset(forProposedContentOffset proposedContentOffset: CGPoint) -> CGPoint {
    let defualtOffset = super.targetContentOffset(forProposedContentOffset: proposedContentOffset)
    guard let collectionView, let indexPath = state.itemIndexPathsToInsert.sorted().first else { return defualtOffset }

    let offsetY = currentSections[indexPath.section].items[indexPath.item].frame.maxY

    return CGPoint(x: defualtOffset.x, y: offsetY)
}
```

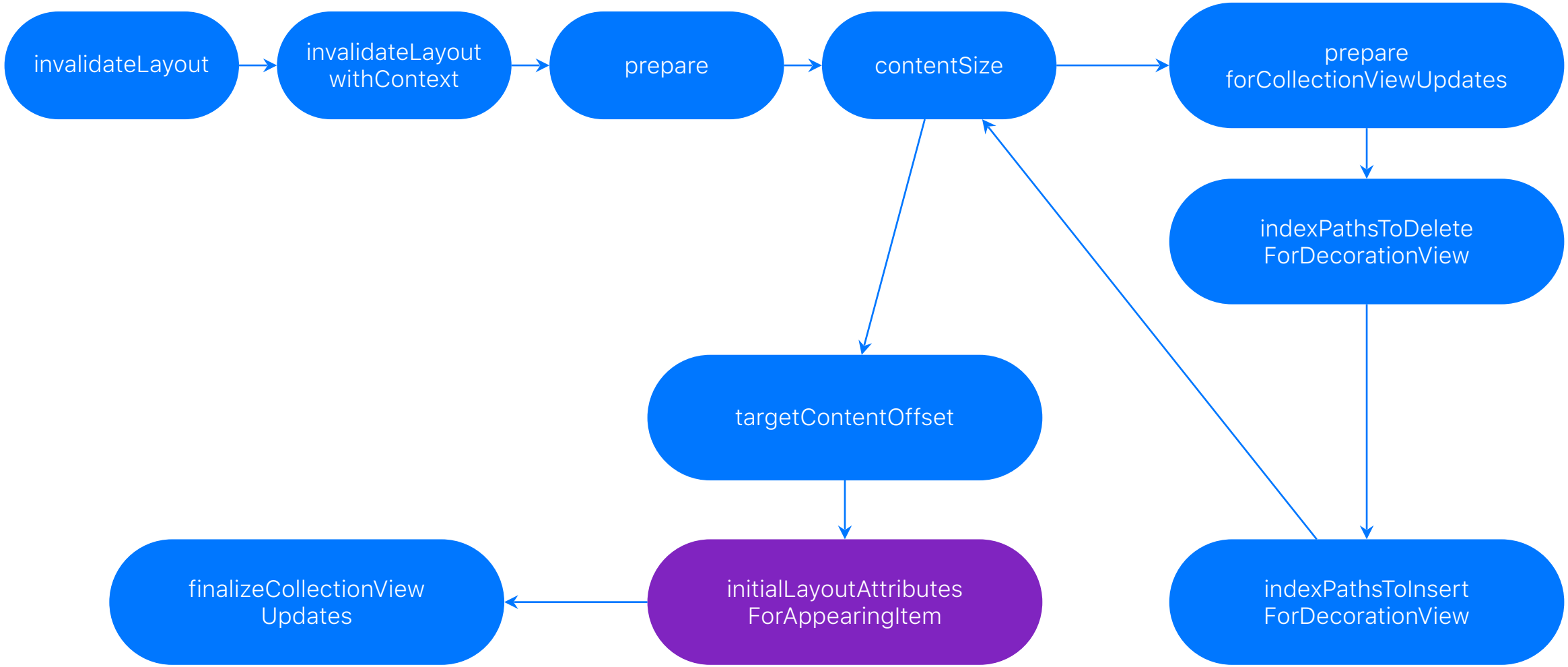
initialLayoutAttributesForAppearingItem

```
override func targetContentOffset(forProposedContentOffset proposedContentOffset: CGPoint) -> CGPoint {
    let defualtOffset = super.targetContentOffset(forProposedContentOffset: proposedContentOffset)
    guard let collectionView, let indexPath = state.itemIndexPathsToInsert.sorted().first else { return defualtOffset }

    let offsetY = currentSections[indexPath.section].items[indexPath.item].frame.maxY

    return CGPoint(x: defualtOffset.x, y: offsetY)
}
```

initialLayoutAttributesForAppearingItem

```
override func targetContentOffset(forProposedContentOffset proposedContentOffset: CGPoint) -> CGPoint {
    let defualtOffset = super.targetContentOffset(forProposedContentOffset: proposedContentOffset)
    guard let collectionView, let indexPath = state.itemIndexPathsToInsert.sorted().first else { return defualtOffset }

    let offsetY = currentSections[indexPath.section].items[indexPath.item].frame.maxY

    return CGPoint(x: defualtOffset.x, y: offsetY)
}
```

Scroll


invalidateLayout
invalidateLayout withContext
prepare
contentSize
prepare forCollectionViewUpdates
indexPathsToDelete ForDecorationView
targetContentOffset
finalizeCollectionView Updates
initialLayoutAttributes ForAppearingItem
indexPathsToInsert ForDecorationView

initialLayoutAttributesForAppearingItem

```swift
override func initialLayoutAttributesForAppearingItem(at itemIndexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    guard state.itemIndexPathsToInsert.contains(itemIndexPath) else {
        return super.layoutAttributesForItem(at: itemIndexPath)
    }

    let attributes = layoutAttributesForItem(at: itemIndexPath)?.copy() as? CustomLayoutAttributes
    attributes?.frame.size.height = 0.0
    return attributes
}
```

initialLayoutAttributesForAppearingItem

```swift
override func initialLayoutAttributesForAppearingItem(at itemIndexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    guard state.itemIndexPathsToInsert.contains(itemIndexPath) else {
        return super.layoutAttributesForItem(at: itemIndexPath)
    }

    let attributes = layoutAttributesForItem(at: itemIndexPath)?.copy() as? CustomLayoutAttributes
    attributes?.frame.size.height = 0.0
    return attributes
}
```

initialLayoutAttributesForAppearingItem

```swift
override func initialLayoutAttributesForAppearingItem(at itemIndexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    guard state.itemIndexPathsToInsert.contains(itemIndexPath) else {
        return super.layoutAttributesForItem(at: itemIndexPath)
    }

    let attributes = layoutAttributesForItem(at: itemIndexPath)?.copy() as? CustomLayoutAttributes
    attributes?.frame.size.height = 0.0
    return attributes
}
```

initialLayoutAttributesForAppearingItem

```swift
override func initialLayoutAttributesForAppearingItem(at itemIndexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    guard state.itemIndexPathsToInsert.contains(itemIndexPath) else {
        return super.layoutAttributesForItem(at: itemIndexPath)
    }

    let attributes = layoutAttributesForItem(at: itemIndexPath)?.copy() as? CustomLayoutAttributes
    attributes?.frame.size.height = 0.0
    return attributes
}
```

Результат


15:51
Back
magni
sit et quibusdam inventore et velit consectetur nulla pariatur
at ex qui provident maxime
eveniet aut ut vero omnis aut
eos non dolorem natus quia tempora vero
reiciendis est corporis illo dolorem voluptatum in
quia velit ut
voluptatum dolores natus non illo error provident nam
dolor omnis consequatur quaerat qui voluptatem excepturi dolore aut
esse ea
Заголовок

Все

Материалы

- https://habr.com/ru/articles/523492/ - статья о том, как парень мучался с этим самым лейаутом
- https://github.com/ekazaev/ChatLayout - репа этого парня, где он реализовал лейаут для чатов
- https://github.com/airbnb/MagazineLayout - репа airbnb, в которую он, а потом и я смотрели для референса
- https://github.com/JonFir/UICollectionViewLayoutExample - моя репа с примерами из доклада

Ваши вопросы