

ios开发-定制自己的瀑布流



作者 sindri的小巢 (/u/0cf7d455eb9e) + 关注

2015.09.18 20:13* 字数 3139 阅读 5054 评论 18 喜欢 22

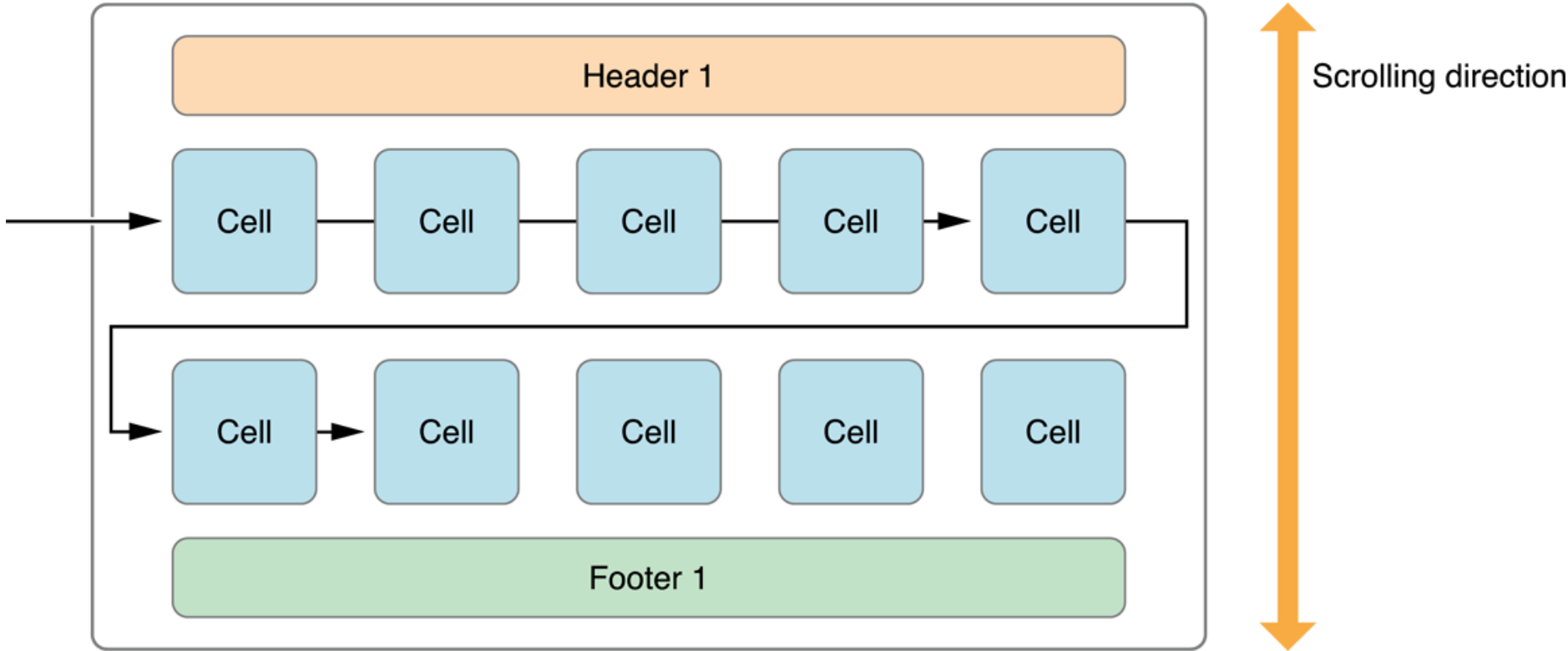
(/u/0cf7d455eb9e)

序言

前段时间开发的时候，需要在tableView上拉的时候实现最底下的cell随着滑动从左边移动出来的效果（淘宝客户端在上拉加载的时候从左边滑动出现的效果）。苦思了很久，最终通过在scrollView的代理中通过判断偏移量来改变当前最下面的cell的frame实现这种效果，但是这样的实现却远远达不到我想要的目标。同时，在滑动tableView时进行大量繁杂的计算还造成了上拉时轻微卡顿的现象，于是我导出寻找另外的解决方案。终于，被我忽视了很久的UICollectionView成为了解决这一问题的最佳选择。

关于UICollectionView

UICollectionView在iOS6之后第一次被引入，它和tableView共享一套API设计，但是功能却远比tableView强大，其最大的特点在于完美的灵活性和可定制化。它对于子视图显示的过程而言仅仅扮演了容器的对象，它不在乎子视图内真正的内容。由于它将决定子视图位置、大小以及外观等属性的任务委托给单独的一个布局对象（*UICollectionViewLayout*），我们可以通过继承这个布局类来实现自定义化的collectionView。



通过上图，我们可以看到UICollectionView不同于tableView的一个特点是前者每一个item并不是单独的一行，官方文档中提及

During layout, the flow layout object adds items to the current line until there is not enough space left to fit an entire item.

在把新的item添加到当前行上的时候，flowLayout对象会计算当前行上剩余的宽度是否足以容纳下这个item，如果无法容纳，那么就换行。更多关于collectionViewLayout特性可以查看这篇文章

(https://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/CollectionViewPGforIOS/UsingtheFlowLayout/UsingtheFlowLayout.html#//apple_ref/doc/uid/TP40012334-CH3-SW4)。

对于自定义collectionView来说，瀑布流可能是最为基本的自定义方案。因此，我们今天的例子将从定制瀑布流开始。苹果官方文档对于UICollectionViewLayout的使用有以下说明：

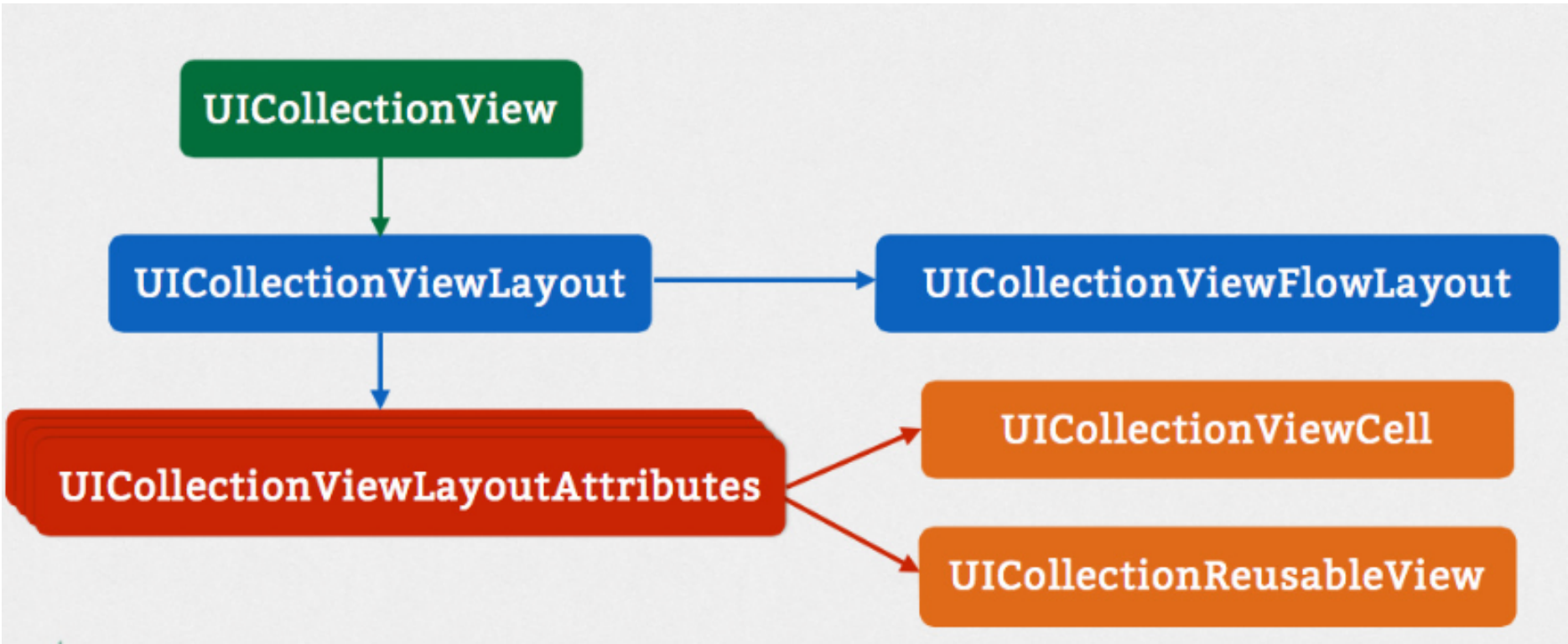
You can configure the flow layout either programmatically or using Interface Builder in Xcode. The steps for configuring the flow layout are as follows:

- 1、 Create a flow layout object and assign it to your collection view.**
- 2、 Configure the width and height of cells.**
- 3、 Set the spacing options (as needed) for the lines and items.**
- 4、 If you want section headers or section footers, specify their size.**
- 5、 Set the scroll direction for the layout.**

大意是通过创建UICollectionViewLayout对象来创建我们的collectionView，然后配置cell的尺寸、间距行距等，有必要的时候还能对组头组尾视图进行设置。

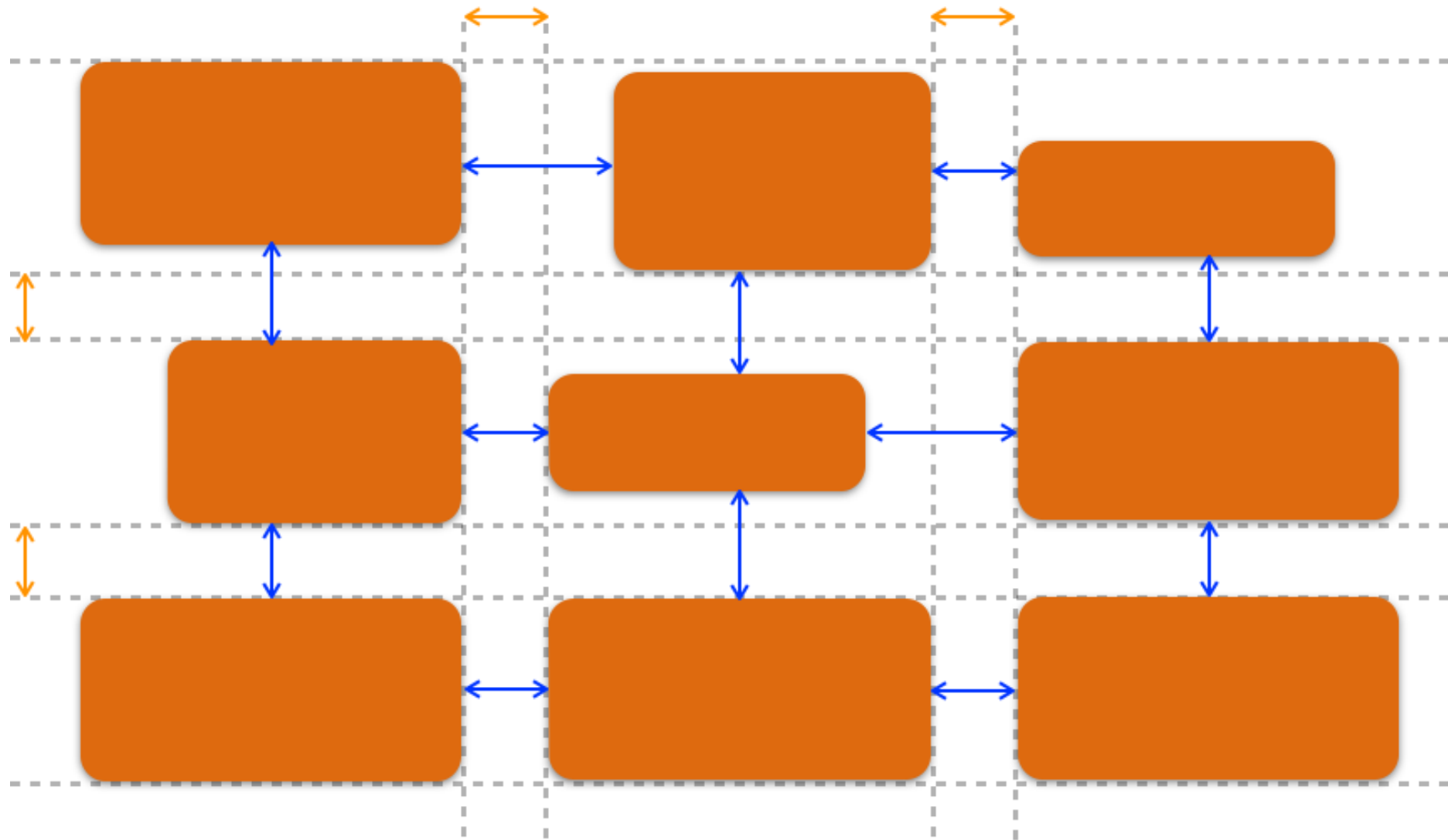
UICollectionViewLayout

在学习如何自定义之前，我们需要了解一下UICollectionView中不同类的依赖关系



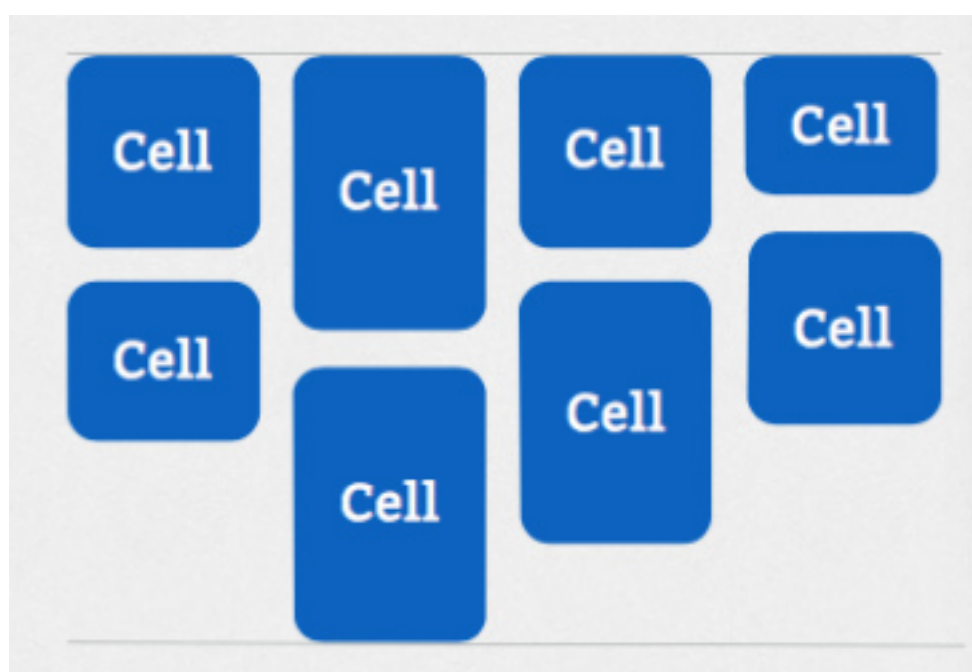
在显示cell的时候，collectionView会向UICollectionViewLayout询问布局属性。这时候，我们可以通过重载方法创建UICollectionViewLayoutAttributes对象，每个对象保存一个item的布局属性。接下来，我们会通过创建UICollectionViewFlowLayout的子类来实现瀑布流，之所以选择这个类的原因在于它的定制要比定制继承自UICollectionViewLayout的子类简单，因为它包括了itemSize、minimumLineSpacing等重要的布局属性。

瀑布流最大的特点在于不同尺寸的cell之间进行紧密的缝合连接，但是如果我们使用的是默认的布局对象，那么显示的效果就会跟下面的图一样不堪入目：



↑↓ minimumLineSpacing ↔ minimumInteritemSpacing ↑↓ 实际距离

我们可以看到，系统计算下一行行高的时候是基于当前本行中y坐标和高度和的最大值加上行距就是下一行的y坐标起始点 $nextLineY = \text{MAX}(\text{cell.y} + \text{cell.height}) + \text{lineSpacing}$ 。所以我们想要实现瀑布流的做法就是通过保存每一列当前的高度，然后用来修改这一列上下一个item的起始坐标来实现每一个cell之间紧凑缝合的效果。



相关方法

- (void)prepareLayout

系统在准备对item进行布局前会调用这个方法，我们重写这个方法之后可以在方法里面预先设置好需要用到的变量属性等。比如在瀑布流开始布局前，我们可以对存储瀑布流高度的数组进行初始化。有时我们还需要将布局属性对象进行存储，比如卡片动画式的定制，也可以在这个方法里面进行初始化数组。切记要调用[super prepareLayout];

- (CGSize)collectionViewContentSize

由于collectionView将item的布局任务委托给layout对象，那么滚动区域的大小对于它而言是不可知的。自定义的布局对象必须在这个方法里面计算出显示内容的大小，包括supplementaryView和decorationView在内。

- (NSArray *)layoutAttributesForElementsInRect:(CGRect)rect

个人觉得完成定制布局最核心的方法，没有之一。collectionView调用这个方法并将自身坐标系统中的矩形传过来，这个矩形代表着当前collectionView可视的范围。我们需要在这个方法里面返回一个包括UICollectionViewLayoutAttributes对象的数组，这个布局属性对象决定了当前显示的item的大小、层次、可视属性在内的布局属性。同时，这个方法还可以设置supplementaryView和decorationView的布局属性。合理使用这个方法的前提是不要随便返回所有的属性，除非这个view处在当前collectionView的可视范围内，又或者大量额外的计算造成的用户体验下降——你加班的原因。

- *(UICollectionViewLayoutAttributes *)layoutAttributesForItemAtIndex:(NSIndexPath *)indexPath*

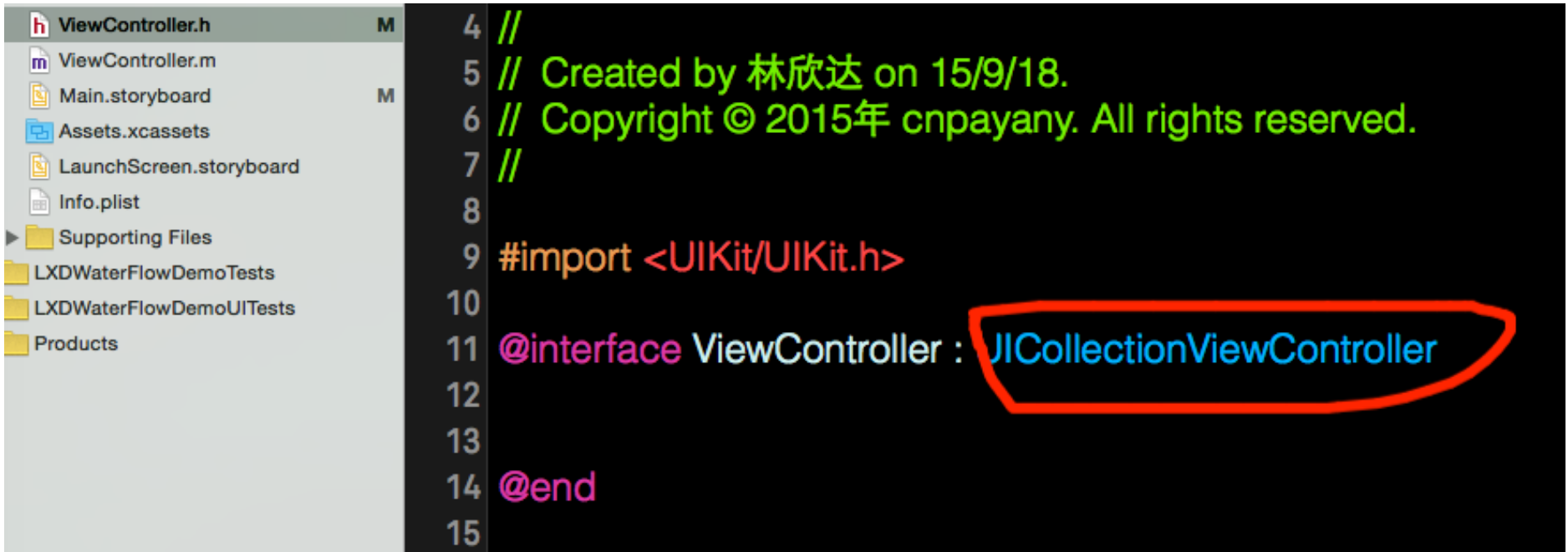
相当重要的方法。collectionView可能会为了某些特殊的item请求特殊的布局属性，我们可以在这个方法中创建并且返回特别定制的布局属性。根据传入的indexPath调用 [UICollectionViewLayoutAttributes layoutAttributesWithIndexPath:]方法来创建属性对象，然后设置创建好的属性，包括定制形变、位移等动画效果在内

- *(BOOL)shouldInvalidateLayoutForBoundsChange:(CGRect)newBounds*

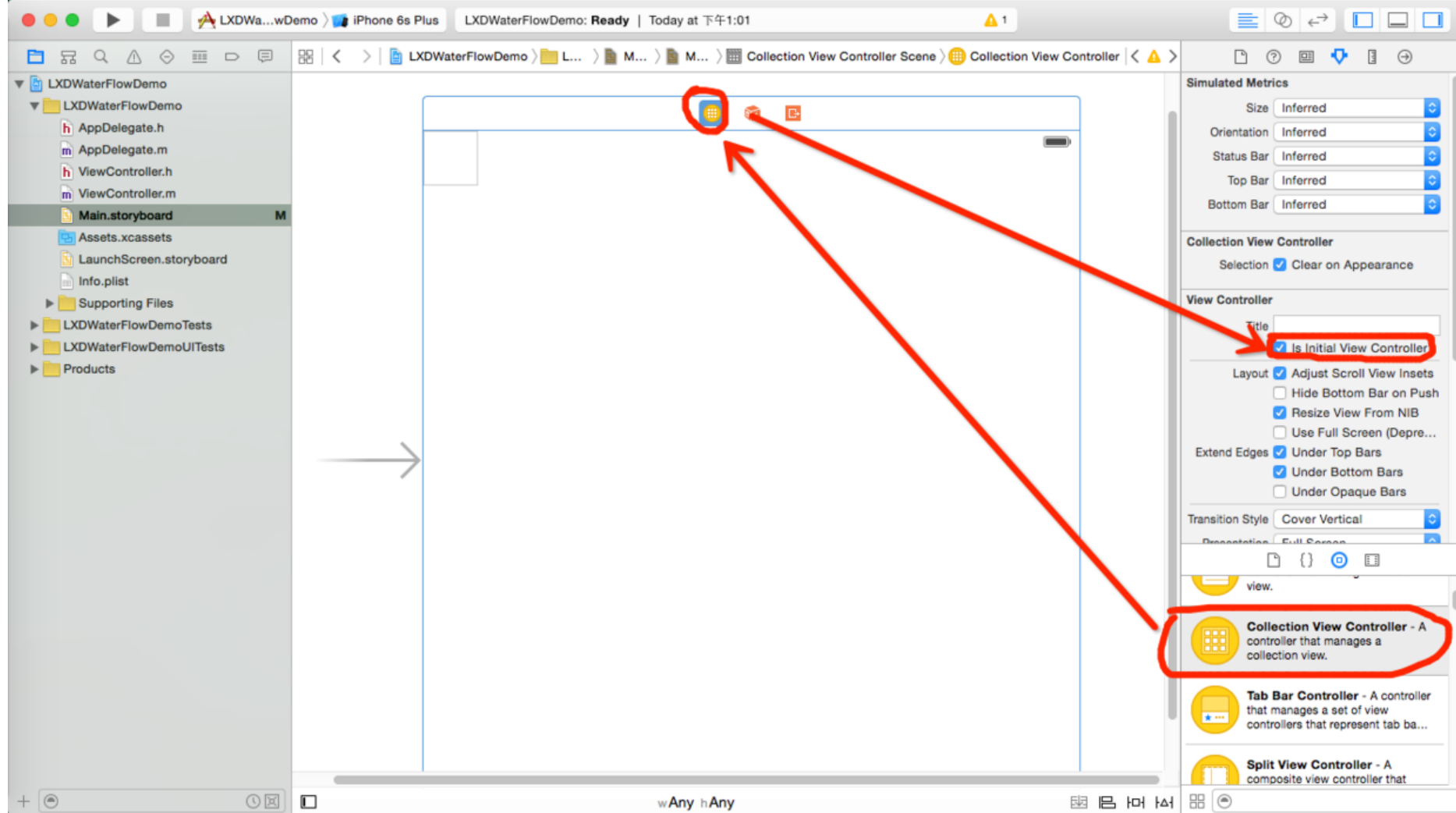
当collectionView的bounds改变的时候，我们需要告诉collectionView是否需要重新计算布局属性，通过这个方法返回是否需要重新计算的结果。简单的返回YES会导致我们的布局在每一秒都在进行不断的重绘布局，造成额外的计算任务。

准备工作

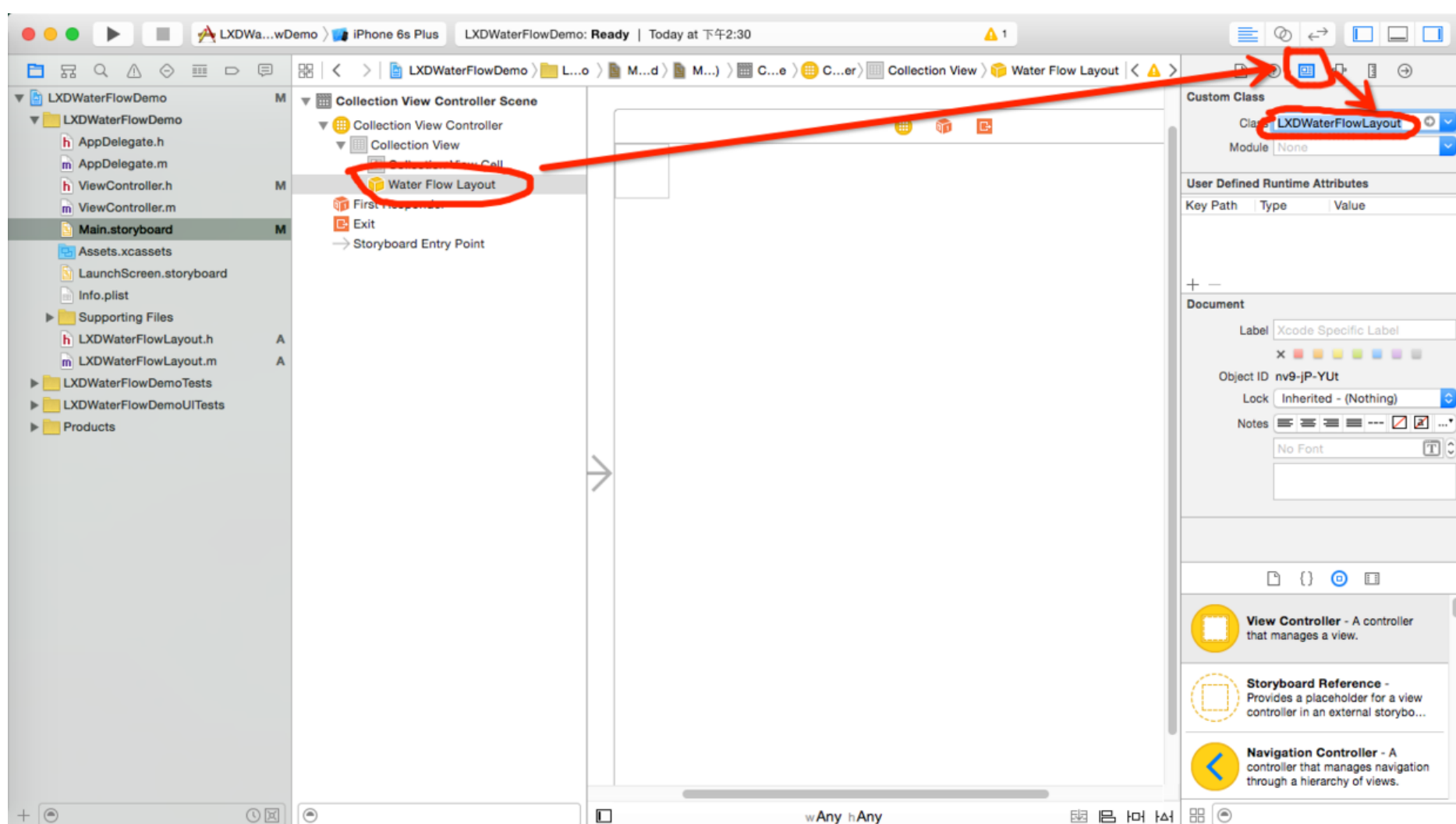
打开Xcode创建一个新项目，命名为名字前缀+WaterFlowDemo。创建好项目之后，选择ViewController.h，然后修改父类为UICollectionViewController



打开Main故事板，然后删除已经存在的ViewController，显示右侧控件栏拉进来一个 UICollectionViewController。然后选中新增进来的控制器，设置为故事板的初始化控制器。



然后command+N新建文件，选择父类为UICollectionViewFlowLayout，命名为WaterFlowLayout，创建布局类。接着，在故事板的控制器里面选择collectionView的布局对象。在开始重写方法实现瀑布流之前，我们要思考好瀑布流的实现思路：



首先，由于瀑布流的item尺寸长宽不定，正常而言分为等宽（竖向）、等高（横向）两种（ps：如果不等宽也不等高，先不说代码上实现起来的复杂程度，单单是视觉上就不合格了）。每一个item都是紧凑连接的，因此我们需要一个容器来存储每一列/行当前的最大长/宽值。这里我们将使用一个存储不同列高度（等宽）的数组来实现。

其次，每一个item的尺寸在第一次展示的时候就应该确定好。虽然瀑布流的尺寸是随机的（实际应用中经常是由图片尺寸决定的），但是我们并不希望在下拉出一大截位置后回头滚动回来的时候，这些item的尺寸再次发生变化。这不符合逻辑，也会导致高度计算上的巨大偏差。所以我们还需要把这些坐标尺寸存储起来，并和item对应的indexPath成对存储。因此我们用NSStringFromCGRect()方法将item的位置信息转换成字符串后和indexPath成对存储在字典中，而且由于frame可能会出现相同值，所以我们将frame转换成字符串存储并让indexPath作为key。

综合上面的考虑，我们的layout类当中应该包括两个成员属性

```
@property (nonatomic, strong) NSMutableDictionary * attributes;
```

```
@property (nonatomic, strong) NSMutableArray * colArray;
```

除此之外，我们还需要几个宏定义来表示包括item间距、行距、列数以及每一个item的宽度：

```
#define COLUMNCOUNT 3

#define SCREENWIDTH [UIScreen mainScreen].bounds.size.width

#define INTERITEMSPACING 10.0f

#define LINESPACING 10.0f 10.0f

#define ITEMWIDTH (SCREENWIDTH - (COLUMNCOUNT - 1)*INTERITEMSPACING) / 3
```

代码实现

```
/**

 * 准备布局item前调用，我们要在这里面完成必要属性的初始化

 */

- (void)prepareLayout

{

    [super prepareLayout];

    //初始化行距间距

    self.minimumLineSpacing = LINESPACING;

    self.minimumInteritemSpacing = INTERITEMSPACING;

    //初始化存储容器

    _attributes = [NSMutableDictionary dictionary];

    _colArray = [NSMutableArray arrayWithCapacity: COLUMNCOUNT];

    for (int i = 0; i < COLUMNCOUNT; i++) {

        [_colArray addObject: @(.0f)];

    }

    //遍历所有item获取位置信息并进行存储

    NSInteger sectionCount = [self.collectionView numberOfSections];

    for (int section = 0; section < sectionCount; section++) {
```

```
        NSInteger itemCount = [self.collectionView numberOfSection: section];

        for (int item = 0; item < itemCount; item++) {

            [self layoutItemFrameAtIndex: [NSIndexPath indexPathWithItem:
item section: section]];

        }

    }

}

/**

 * 用来设置每一个item的尺寸，然后和indexPath存储起来

 */

- (void)layoutItemFrameAtIndex: (NSIndexPath *)indexPath

{

    CGSize itemSize = CGSizeMake(ITEMWIDTH, 100+arc4random%101);

    //获取当前三列高度中高度最低的一列

    NSInteger smallestCol = 0;

    CGFloat lessHeight = [_colArray[smallestCol] doubleValue];

    for (int col = 1; col < _colArray.count; col++) {

        if (lessHeight < [_colArray[col] doubleValue]) {

            shortHeight = [_colArray[col] doubleValue];

            smallestCol = col;

        }

    }

    //在当前高度最低的列上面追加item并且存储位置信息

    UIEdgeInsets insets = self.collectionView.contentInset;

    CGFloat x = insets.left + smallestCol * (INTERITEMSPACING + ITEMWIDTH);

    CGRect frame = {x, insets.top + shortHeight, itemSize};

    [_attributes setValue: indexPath forKey: NSStringFromCGRect(frame)];

    [_colArray replaceObjectAtIndex: smallestCol withObject:
@CGRectGetMaxY(frame)];

}
```

```
/**

* 返回所有当前在可视范围内的item的布局属性

*/

- (NSArray *)layoutAttributesForElementsInRect: (CGRect)rect

{

    //获取当前所有可视item的indexPath。通过调用父类获取的布局属性数组会缺失一部分可视item的布局属性

    NSMutableArray * indexPaths = [NSMutableArray array];

    for (NSString * rectStr in _attributes) {

        CGRect cellRect = CGRectFromString(rectStr);

        if (CGRectIntersectsRect(cellRect, rect)) {

            NSIndexPath * indexPath = _attributes[rectStr];

            [indexPaths addObject: indexPath];

        }

    }

    //获取当前要显示的所有item的布局属性并返回

    NSMutableArray * layoutAttributes = [NSMutableArrayWithCapacity: indexPaths.count];

    [indexPaths enumerateObjectsUsingBlock: ^(NSIndexPath * indexPath, NSUInteger idx, BOOL * stop) {

        UICollectionViewLayoutAttributes * attributes = [self layoutAttributesForItemAtIndexPath: indexPath];

        [layoutAttributes addObject: attributes];

    }];

    return layoutAttributes;

}

/**

* 返回对应indexPath的布局属性

*/
```


- (UICollectionViewLayoutAttributes *)layoutAttributesForItemAtIndexPath:
(NSIndexPath *)indexPath {

UICollectionViewLayoutAttributes * attributes =
[UICollectionViewLayoutAttributes layoutAttributesForCellWithIndexPath:
indexPath];

for (NSString * frame in _attributes) {

if (_attributes[frame] == indexPath) {

attributes.frame = CGRectFromString(frame);

break;

}

}

return attributes;

}

/**

*** 设置collectionView的可滚动范围（瀑布流必要实现）**

***/**

- (CGSize)collectionViewContentSize

{

__block CGFloat maxHeight = [_colArray[0] floatValue];

[_colArray enumerateObjectsUsingBlock: ^(NSNumber * height, NSUInteger
idx, BOOL *stop) {

if (height.floatValue > maxHeight) {

maxHeight = height.floatValue;

}

}

return CGSizeMake(CGRectGetWidth(self.collectionView.frame), maxHeight +
self.collectionView.contentInset.bottom);

}

/**

*** 在collectionView的bounds发生改变的时候刷新布局**

```

*/

- (BOOL)shouldInvalidateLayoutForBoundsChange: (CGRect)newBounds

{

    return !CGRectEqualToRect(self.collectionView.bounds, newBounds);

}

```

多说几句

使用collectionView完成业务需求之后，它几乎成为了我最喜爱的控件，极高的可定制性决定了它的重要地位。虽然从代码实现的角度上来说，合适的布局几乎可以实现tableView，但它不是为了取代后者而出现的。collectionView相较tableView而言，并不那么的大众化，毕竟常规的数据展示使用tableView就能完美显示。

上面瀑布流的代码中，item的高度是在layout里面随机生成的，但在实际开发中，高度的生成不该由布局对象来完成。为了解决这个问题，我们可以在自定义的布局对象中增加一个遵循UICollectionViewDelegateFlowLayout的代理人属性：

```

@property (nonatomic, weak) id<UICollectionViewDelegateFlowLayout>
delegate;

```

然后在prepareLayout方法中加上一句self.delegate = self.collectionView.delegate。这样我们就可以通过向代理对象发送协议方法消息来获取itemSize（将item的尺寸交给controller来完成）。

除了上面提到的属性之外，UICollectionViewLayoutAttributes还有center、zIndex、transform3D等属性能让我们定制滑动的形变等动画效果，例如卡片动画就是我非常喜爱的效果之一。此外还有下面六个方法帮助我们在对item进行删除和添加操作的时候定制动画效果

initialLayoutAttributesForAppearingItemAtIndexPath:

initialLayoutAttributesForAppearingSupplementaryElementOfKind:atIndexPath:

:

initialLayoutAttributesForAppearingDecorationElementOfKind:atIndexPath:

finalLayoutAttributesForDisappearingItemAtIndexPath:

finalLayoutAttributesForDisappearingSupplementaryElementOfKind:atIndexPath:

finalLayoutAttributesForDisappearingDecorationElementOfKind:atIndexPath:

关于这些方法的使用可以学习这篇文章

(<http://www.cocoachina.com/industry/20140725/9247.html>)。



sindri的小巢 (/u/0cf7d455eb9e)

写了 102229 字，被 2525 人关注，获得了 1636 个喜欢 (/u/0cf7d455eb9e)




+ 关注

iOS小码农一枚

如果觉得我的文章对您有用，请随意打赏。您的支持将鼓励我继续创作！


赞赏支持

喜欢 | 22



更多分享

(http://cwb.assets.jianshu.io/notes/images/2109891/v




写下你的评论...

18条评论

只看作者

按喜欢排序 按时间正序 按时间倒序



低吟浅唱1990 (/u/ddba270c0490)

2楼 · 2015.10.29 16:04


(/u/ddba270c0490)


代码好像有点问题哦


 赞

 回复

sindri的小巢 (/u/0cf7d455eb9e): @低吟浅唱1990 (/users/ddba270c0490) 哪里出问题了?

2015.10.29 17:57  回复

 添加新评论



低吟浅唱1990 (/u/ddba270c0490)

3楼 · 2015.10.29 18:01

(/u/ddba270c0490)

不知道是不是你没有写全的原因。自我感觉

NSInteger smallestCol = 0;

```
CGFloat lessHeight = [_colArray[smallestCol] doubleValue];
```

```
for (int col = 1; col < _colArray.count; col++) {
```

```
if (lessHeight < [_colArray[col] doubleValue]) {
```

```
shortHeight = [_colArray[col] doubleValue];
```

```
smallestCol = col
```

这一部分有点问题。你看看

 赞

 回复

sindri的小巢 (/u/0cf7d455eb9e): @低吟浅唱1990 (/users/ddba270c0490) 这段代码是获取当前列数中最短的一个插入新的cell。

2015.10.29 19:47

回复

低吟浅唱1990 (/u/ddba270c0490):

@Sindri的小巢 (/users/0cf7d455eb9e) 我知道，但是你好像写反了。比较的时候

2015.10.29 20:00

回复

sindri的小巢 (/u/0cf7d455eb9e):

@低吟浅唱1990 (/users/ddba270c0490) 你运行下看看，每个cell输出一下自己的item位置，没有写错

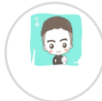
2015.10.29 20:17

回复

添加新评论

还有2条评论，

展开查看



fallrainy (/u/83d05575aa1b)

4楼 · 2015.10.30 00:51

(/u/83d05575aa1b)

不知你说的效果是淘宝哪个页面的效果？

赞

回复


sindri的小巢 (/u/0cf7d455eb9e):

@fallrainy (/users/83d05575aa1b) 你往上拉看商品的时候，那些cell会有一个左移的动画

2015.10.30 08:08

回复

添加新评论



旧夏2014 (/u/54cc1a5fc8b8)

5楼 · 2016.03.03 16:47

(/u/54cc1a5fc8b8)

这个能改下适用加上头尾视图么

赞

回复

sindri的小巢 (/u/0cf7d455eb9e):

@旧夏2014 (/users/54cc1a5fc8b8) 可以，那你就需要多增加保存头尾视图高度的变量，来计算cell的正确位置

2016.03.03 16:48

回复

旧夏2014 (/u/54cc1a5fc8b8):

@Sindri的小巢 (/users/0cf7d455eb9e) 那你最后返回NSArray 那个方法里怎么知道是头尾还是cell呢

如果是头尾需要掉这个方法

UICollectionViewLayoutAttributes * attributes = [self layoutAttributesForSupplementaryViewOfKind: atIndexPath:];

可是kind参数怎么传呢

2016.03.03 16:56

回复

旧夏2014 (/u/54cc1a5fc8b8):

@旧夏2014 (/users/54cc1a5fc8b8) 另外就是为什么保存的是 indexPath而不直接保存UICollectionViewLayoutAttributes * attributes呢


2016.03.03 16:57

回复

添加新评论

还有2条评论，

展开查看



faee2f946885 (/u/faee2f946885)

6楼 · 2016.07.14 14:29

(/u/faee2f946885)

学习思路还是挺不错的，但是你这个排版实在难看，代码框都没有，还有代码错了几处。

如果你想展现Demo我建议你放在GitHub上吧

赞

回复

sindri的小巢 (/u/0cf7d455eb9e):

@faee2f946885 (/users/faee2f946885) 这是早期的博客了，那时候不是用md语法，我看看过段时间用md重写吧

2016.07.14 18:18

回复

添加新评论

被以下专题收入，发现更多相似内容

+

我的专题

ios

程序员

我说技术

iOS

iOS开发技巧进阶

iOS Pro...

iOS 开发

iOS点滴

IOS

iOS开发技术分享

iOS开发记

collect...

Swift U...

iOS进阶

技术

加载更多...