


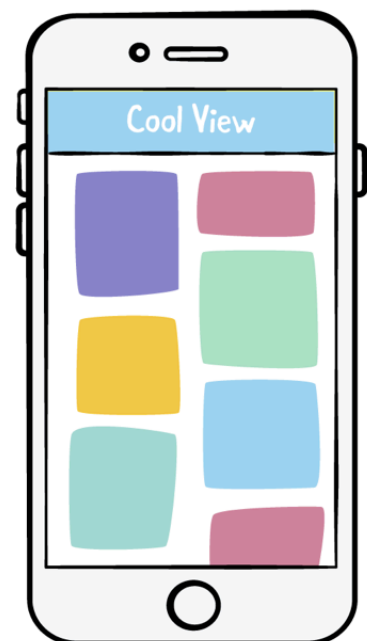


CUSTOM 
COLLECTION
 VIEW
LAYOUT 



 HANDS-ON CHALLENGES

Custom Collection View Layout

Michael Briscoe

Copyright ©2017 Razeware LLC.

Notice of Rights

All rights reserved. No part of this book or corresponding materials (such as text, images, or source code) may be reproduced or distributed by any means without prior written permission of the copyright owner.

Notice of Liability

This challenge and all corresponding materials (such as source code) are provided on an "as is" basis, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in action of contract, tort or otherwise, arising from, out of or in connection with the software or the use of other dealing in the software.

Trademarks

All trademarks and registered trademarks appearing in this book are the property of their own respective owners.

Insets and Cell Padding

With the basics of the Mosaic layout setup, how about you give the cells a little elbow room? Even though there are no predefined spacing properties in `UICollectionViewLayout`, we can implement our own.

Hints:

1. Maybe you can add a new `cellPadding` property to `MosaicViewLayout`?
2. You could also inset the content of the collection view.
3. Don't forget to take the inset into account when making calculations.

Before you turn the page for our solution, be sure to give it a try for yourself first!

Solution

Open **MosaicViewLayout.swift** and add the following variable to the top of the class with the other properties:

```
var cellPadding: CGFloat = 0
```

The layout will use this property to hold the amount of desired cell padding.

Next, replace the width variable definition with this:

```
fileprivate var width: CGFloat {  
    get {  
        let insets = collectionView!.contentInset  
        return collectionView!.bounds.width - (insets.left + insets.right)  
    }  
}
```

This takes the collection view's contentInset into account when returning the width of the collection view.

Now add this line to prepare() within the block that is stepping through each item, just after let frame =...:

```
let insetFrame = frame.insetBy(dx: cellPadding, dy: cellPadding)
```

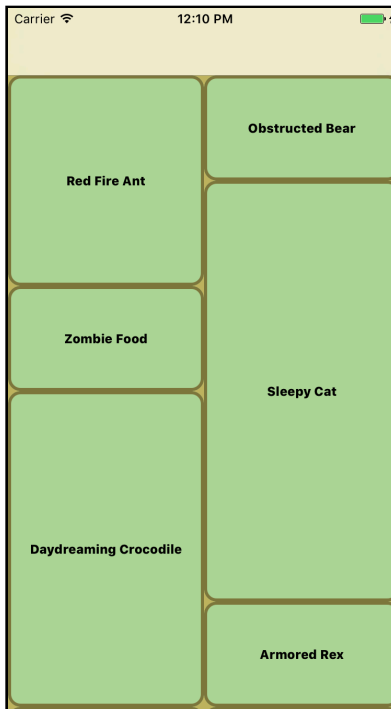
Here you are taking the calculated frame, and adding an inset by the amount of cellPadding.

Next, within the same block of code, change the line that sets attributes.frame to the following:

```
attributes.frame = insetFrame
```

This passes the new frame defined in insetFrame to the attributes frame property.

Build and run.



Oops! We forgot to set the insets and cell padding!

Open **MasterViewController.swift** and add these lines to `viewDidLoad()`:

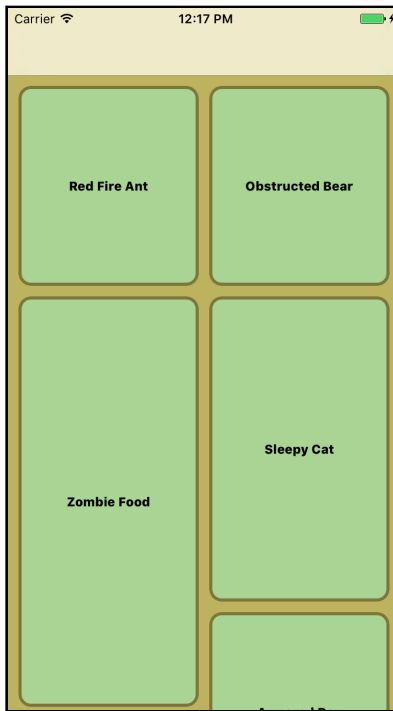
```
collectionView!.contentInset = UIEdgeInsets(top: 5, left: 5, bottom: 10,
right: 5)
layout.cellPadding = 5
```

The first line insets the overall content so that it appears 5 pixels away from the edges of the screen.

The second line adds 5 pixels of padding so that each cell isn't touching each other.

Now do a build and run.

Custom Collection View Layouts



Now that's more like it!