

 IvanCaldwell / **CollectionViewChallenge**
forked from LambdaSchool/CollectionViewChallenge

No description, website, or topics provided.


Edit

[Manage topics](#)

1 commit1 branch0 releases1 contributor


Branch: masterNew pull requestCreate new fileUpload filesFind fileClone or download

This branch is even with LambdaSchool:master.

 erica initial commit


CollectionViewChallenge.xcodeproj

initial commit

 CollectionViewChallenge


initial commit

7 hours ago

 images

initial commit

7 hours ago

 README.md

initial commit

7 hours ago

README.md

Module Challenge: Collection Views - Collected

This challenge allows you to practice the concepts and techniques learned in today's guided lesson and apply them in a concrete project. Your lesson explored collection view controllers. You demonstrate proficiency by creating an application that showcases the same features you learned in class.

Instructions

Read these instructions carefully. Understand exactly what is expected *before* starting this Challenge.

This is an individual assessment but you are permitted to consult with and support other members of your cohort. You are encouraged to follow the twenty-minute rule and seek support from your PM and Instructor in your cohort help channel on Slack.

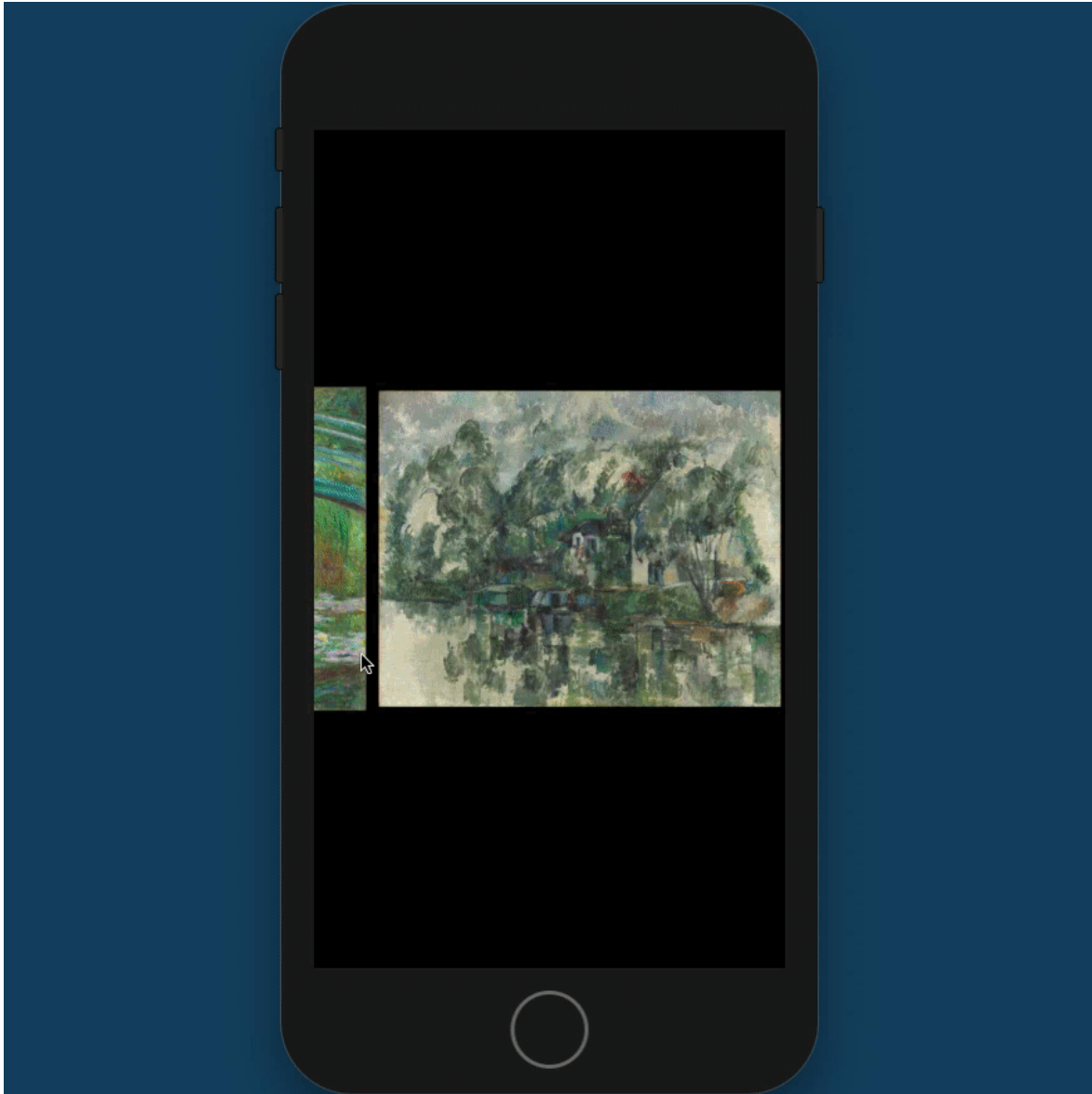
Commits

Commit your code regularly and meaningfully. This helps both you (in case you ever need to return to old code for any number of reasons) and your project manager.

Description

In this challenge, you build a scrolling art gallery. This gallery respects and preserves image aspect ratios during layout rather than embedding them into same-size views. Narrow images will sit directly next to wide ones and vice versa, without padding. To accomplish this you'll use an advanced collection view feature that lets you customize the size of each collection view cell.

Your completed application will look like the following video:



The final high-hanging image in landscape mode is acceptable.

Questions

Demonstrate your understanding of today's concepts by answering the following free-form questions. Submit them as a text file along with your project.

- What is a collection view controller (CVC)?
- When do you use a table view controller (TVC) and when a collection view controller (CVC)?
- What are the differences in the CVC vs TVC data source methods?
- What kinds of interfaces can you build with CVCs that you cannot with TVCs? Under what circumstances are TVCs preferable?
- When do you use sections to organize table and collection view controllers? What advantages do sections offer?

Project Setup

Follow these steps to set up your project:

- Fork this project.
- Create `CollectionViewController.swift`, `CollectionViewCell.swift`, `CollectionViewCell.xib`
- Create skeleton implementations for `CollectionViewController` and `CollectionViewCell`

```
class CollectionViewController: UICollectionViewController {  
}  
  
class CollectionViewCell: UICollectionViewCell {  
}
```

- Add a new collection view controller to `Main.storyboard`.
- Set it as the entry point.
- Delete its prototype cell.
- Set its background color to black.
- Set its identity to `CollectionViewController`.

Set up the view cell

These steps design your view cell in a xib file before copying it to your main storyboard:

- In `CollectionViewCell.xib`, delete the default view and drag in a collection view cell.
- Set the view's identity to `CollectionViewCell`.
- Add an image view and constrain it to stretch to all sides of its parent.
- Connect the image view to the an `imageView` IBOutlet.
- Set the image view's vertical and horizontal content hugging priorities to 1000.
- Set the image view's contentMode to scale aspect fit.
- In the collection view cell implementation, create a static property called `reuseIdentifier`. Set it to `"cell"`.
- Copy your cell and paste it into the main storyboard's collection view controller.
- Set the cell's re-use identifier in your main storyboard.

Load the images

Follow these steps to load the images into an array to power your collection view:

- Create an `images` variable property for the collection view controller. Set the `images` property type to `[UIImage]` and its initial value to `[]`.
- In the collection view controller's `viewDidLoad` method, load all 12 images into the image array. Don't forget to use string interpolation and check whether the image loaded properly using `guard let image = UIImage(named: name)`.

Set up the layout

Establish a flow layout with these steps:

- Conform the collection view to `UICollectionViewDelegateFlowLayout`
- Add two `CGFloat` properties called `targetDimension` (set it to 320) and `insetAmount`, set it to 32.
- Implement `viewWillAppear` and within it, retrieve the collection view's `collectionViewLayout`, casting it to `UICollectionViewFlowLayout`.
- Set the layout's `sectionInset` using the `insetAmount` for each inset.
- Set the layout's `minimumLineSpacing` to `.greatestFiniteMagnitude`.
- Set the scroll direction to horizontal.

Set up the data source

Build your data source to provide images to your collection view:

- Implement the collection view's `numberOfItemsInSection` (`images.count`) method.
- Implement the `cellForItemAt` method. In it, set the image view's image. Don't forget to conditionally cast the cell (using `as?`) to the right type.

Control the view size for your layout

Here's where flexible sizing comes to play. Add the following method to your collection view:

```
func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout,
sizeForItemAt indexPath: IndexPath) -> CGSize {

    // Fetch image
    let image = images[indexPath.row]

    // Fetch largest dimension of the image, whether width or height
    let maxDimension = max(image.size.width, image.size.height)

    // Calculate how to scale that largest dimension into `targetDimension`
    let scale = targetDimension / maxDimension

    // Return scaled dimensions
    return CGSize(width: image.size.width * scale, height: image.size.height * scale)
}
```

This method uses the largest dimension of each image to scale to your `targetDimension` extent.

Minimum Viable Product

Your finished project must include all of the following requirements:

- Your collection view looks like the sample in the preview, with flexible cell sizing.
- You correctly include all the images provided to you.

Stretch Problems

After finishing your required elements, push your work further. These goals may or may not be things you learned in this module but they build on the material you just studied. Time allowing, stretch your limits and see if you can deliver on the following optional goals:

- Add more art pieces to your gallery.
- Implement single selection by creating a yellow border around the selected cells. Don't forget about cell re-use.
- Implement multiple selection.
- Add a label to each cell. Use Google's reverse image search to find the name for each work of art.
- Embed the collection view in a Navigation Controller and set the collection view controller's `title` property to reflect the number of images selected or the name of the work that has been selected.

Progress Dashboard

-

Download checked items