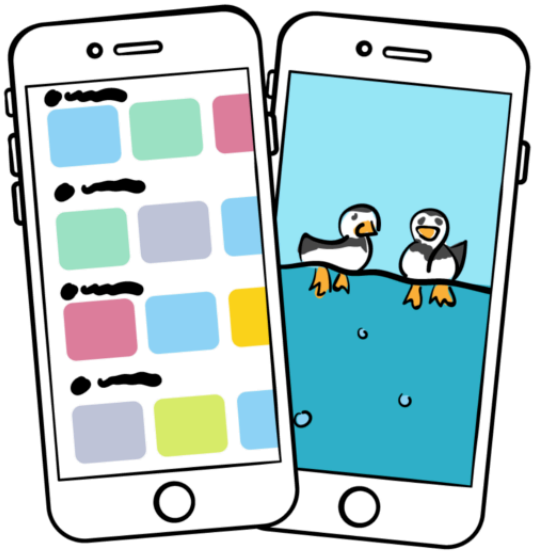
BEGINNING

COLLECTION VIEWS



HANDS-ON CHALLENGES

Beginning Collection Views

Michael Briscoe

Notice of Rights

Notice of Liability

Trademarks

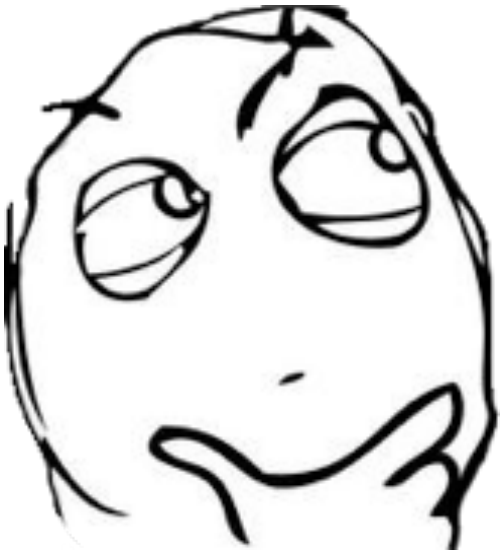
2 Challenge #2: Cell Selection

By Michael Briscoe

With your collection view all setup, and hooked up to the parks data source via the appropriate `UICollectionViewDataSource` methods, it's time to think about cell selection.

The collection view as it stands lacks any sort of user interaction, even though you have everything that you need to make it possible. Ideally, tapping on a cell should push the detail view controller that's already setup in the storyboard onto the navigation stack and display the corresponding national park.

But, just like UITableView, there are two different ways you can handle cell selection: using a Selection Segue in Interface Builder, or using the appropriate `UICollectionViewDelegate` method. So, which one should you choose?



How about both! :]

Challenge A: Using a Selection Segue

Your first challenge is to implement cell selection using a Selection Segue.

Hints:

1. Within Interface Builder, You'll need to setup a selection segue between the **CollectionViewCell** and the **Detail View Controller**.
2. Override `prepare(for:sender)` in the `UICollectionViewController` subclass to display the park data.

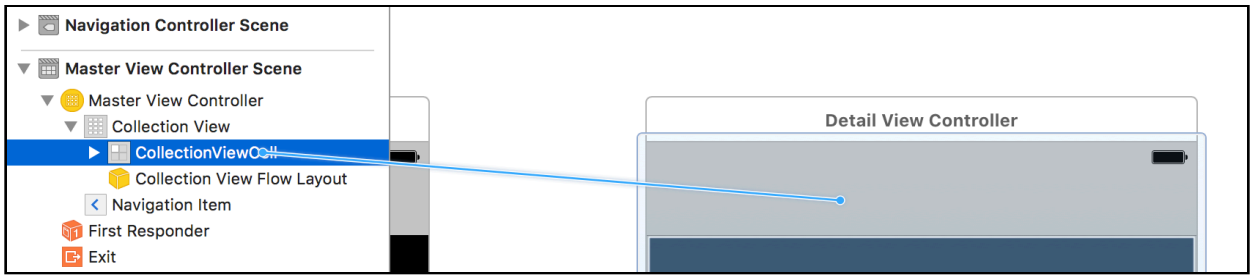
Before you turn the page for our solution, be sure to give it a try for yourself first!

Solution

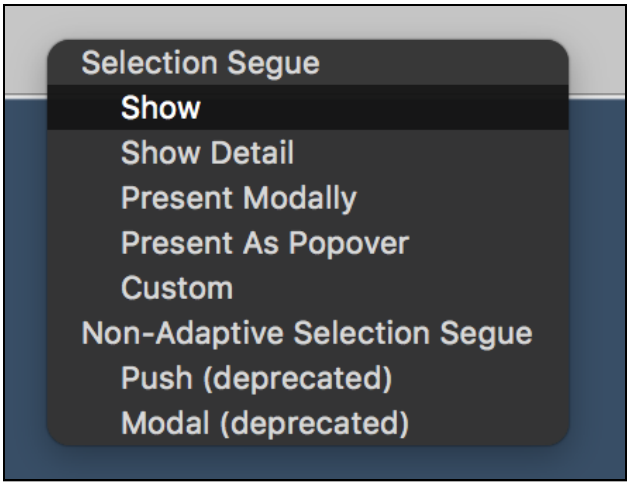
Using a Selection Segue

Open **Main.storyboard** from the Layout group and, using the disclosure triangles, expand Master View Controller Scene in the Document Outline until you find the **CollectionViewCell** object.

Next, ctrl+drag from **CollectionViewCell** to the **Detail View Controller** on the storyboard canvas to create a segue:

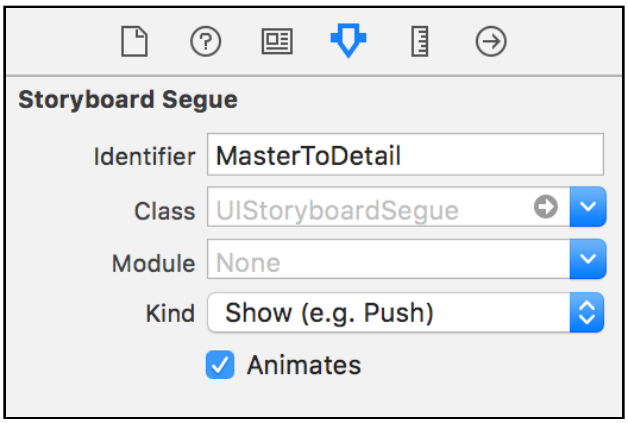


From the resulting popup menu, from the **Selection Segue** section, choose **show**:



The final step to creating a selection segue in Interface Builder is to give the segue an identifier so that it can be referenced from code.

Select the segue you just created, either in the Document Outline or in the storyboard canvas, and then in the Attributes Inspector enter **MasterToDetail** in the **Identifier** text box, like so:



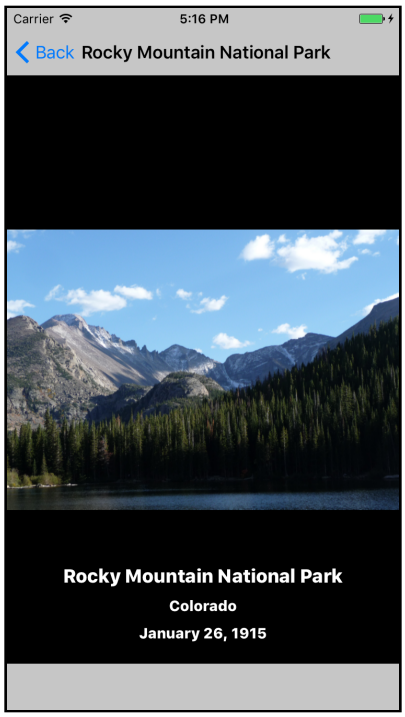
Build and run. Tapping on a cell now pushes the detail view controller onto the navigation stack, but you'll notice the selected park isn't being displayed. You're going to fix that now.

Open **MasterViewController.swift** from the **Controllers** group, and just below `viewDidLoad()`, add the following method:

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
  // 1
  if segue.identifier == "MasterToDetail" {
    // 2
    if let indexPath = collectionView!.indexPathsForSelectedItems!
      .first {
      // 3
      if let park = parksDataSource.parkForItemAtIndexPath(indexPath) {
        let detailViewController = segue.destination as!
          DetailViewController
        detailViewController.park = park
      }
    }
  }
}
```

1. First make sure you're working with the correct segue by checking its identifier.
2. Get the index path of the selected cell. Since collection views support multiple cell selection, `indexPathsForSelectedItems` returns an array, so you simply take the first index path.
3. Next, you ask the parks data source for the park corresponding to that index path, before finally setting it as the park on the detail view controller.

Build and run. You can now tap any cell and the corresponding park will be displayed by the detail view controller.



Challenge B: Using The Delegate

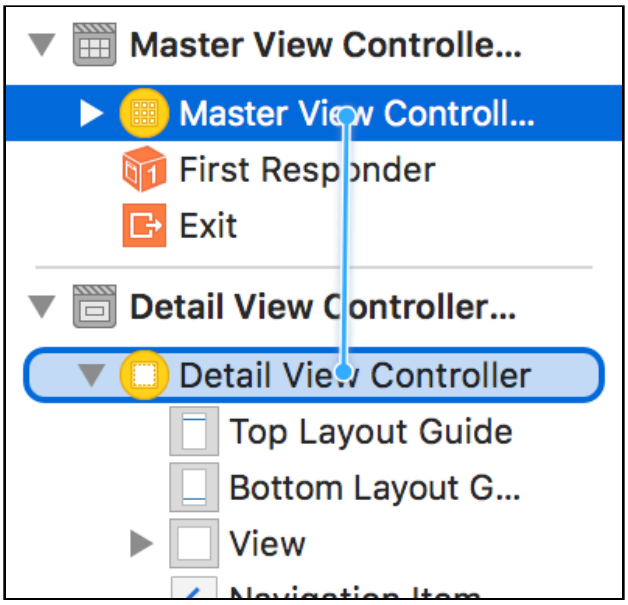
Now that you've used a selection segue to view the detail view, your next challenge —should you choose to accept it—is to implement cell selection with the `UICollectionViewDelegate` protocol.

Take a look at the `UICollectionViewDelegate` documentation and try for yourself, before turning the page. :]

Solution

Using The Delegate

Open **Main.storyboard** and delete the segue you created earlier. Next, ctrl+drag from **Master View Controller** to **Detail View Controller** in the Document Outline to create a manual segue:



From the resulting popup menu, from the **Manual Segue** section, choose **show**. Then, select the segue you just created, either in the Document Outline or in the storyboard canvas, and in the Attributes Inspector enter **MasterToDetail** in the **Identifier** text box.

Open **MasterViewController.swift** from the **Controllers** group, and at the bottom of the class add the following:

```
// MARK: UICollectionViewDelegate
extension MasterViewController {

  override func collectionView(_ collectionView: UICollectionView,
    didSelectItemAt indexPath: IndexPath) {
    if let nationalPark =
      parksDataSource.parkForItemAtIndexPath(indexPath) {
        performSegue(withIdentifier: "MasterToDetail", sender:
          nationalPark)
    }
  }
}
```

Here you implement the delegate method `collectionView(_:didSelectItemAt:)` which is called by the collection view whenever a user selects a cell. You simply ask the parks data source for the park corresponding to the given index path, and then

trigger the segue manually, passing the park as the sender. This is a neat little trick if your `prepareFor(_:sender:)` implementation needs access to the selected object, like it does here, and saves you having to track that object using a property if it's not used anywhere else.

Finally, replace the exisiting implementation of `prepareFor(_:sender:)` with the following:

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
  if segue.identifier == "MasterToDetail" {
    let detailViewController = segue.destination as! DetailViewController
    detailViewController.park = sender as? Park
  }
}
```

Here you've simplified the implementation somewhat by removing the code that asks the collection view for the selected index path, and already have access to selected park since it's passed from `collectionView(_:didSelectItemAt:)`. The final step is to cast the sender variable to an instance of Park so it can be set on the detail view controller without throwing a compiler error.

Build and run. Like before, you can tap any cell and the corresponding national park will be displayed by the detail view controller.

Using the selection delegate method as opposed to a selection segue offers a lot more control over how selection is handled, as you'll see later in the video series.