Using Swift Enums for safer UICollectionViews and UITableViews

swift.berlin meetup
November 30, 2015
Ariel Elkin

PROJECT A

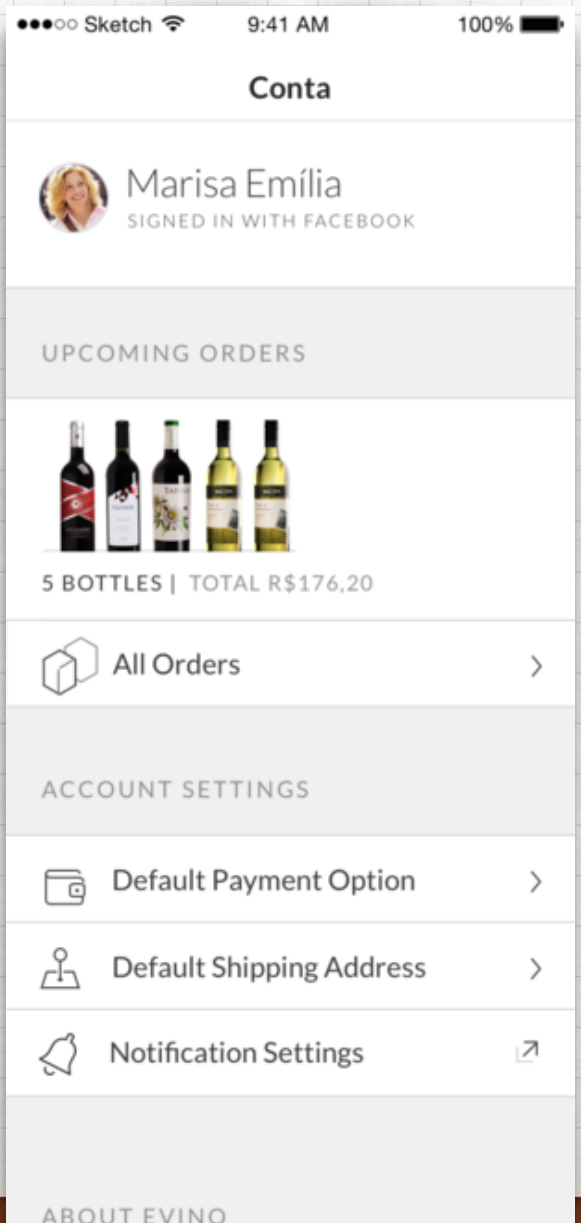
- Early-stage investor and company builder
- Marketplaces, E-commerce, Saas
- 20+ companies in our portfolio

ME

- iOS Developer in Project A's Mobile Team
- 10+ apps in the App Store
- Puts his dishes in the dishwasher

COLLECTION VIEWS ARE COMPLEX

- section numbers
- item sizes
- section count
- reuse identifiers
- etc…



MANAGING THE LAYOUT STRUCTURE

- Typically involves hard-coding integer values

```
28      let amountPriceSectionNumber = 0
29      let bottleCollectionViewSectionNumber = 1
30      let postcodeCellSectionNumber = 2
31      var summarySectionNumber = Int()
32      var freeShippingSectionNumber = Int()
33      var subscriptionSectionNumber = Int()
34      var checkoutSectionNumber = Int()
```

MANAGING THE LAYOUT STRUCTURE

DISADVANTAGES

- Layout information is fragmented

```
28      let amountPriceSectionNumber = 0
29      let bottleCollectionViewSectionNumber = 1
30      let postcodeCellSectionNumber = 2
31      var summarySectionNumber = Int()
32      var freeShippingSectionNumber = Int()
33      var subscriptionSectionNumber = Int()
34      var checkoutSectionNumber = Int()
```

MANAGING THE LAYOUT STRUCTURE

DISADVANTAGES

- We have to manually ensure that checks are exhaustive

```
switch indexPath.section {

case amountPriceSectionNumber:
    return CGSize(width: UIScreen.mainScreen().bounds.width, height: 50)

case bottleCollectionViewSectionNumber:
    return CGSize(width: UIScreen.mainScreen().bounds.width, height: 120)
```

MANAGING THE LAYOUT STRUCTURE

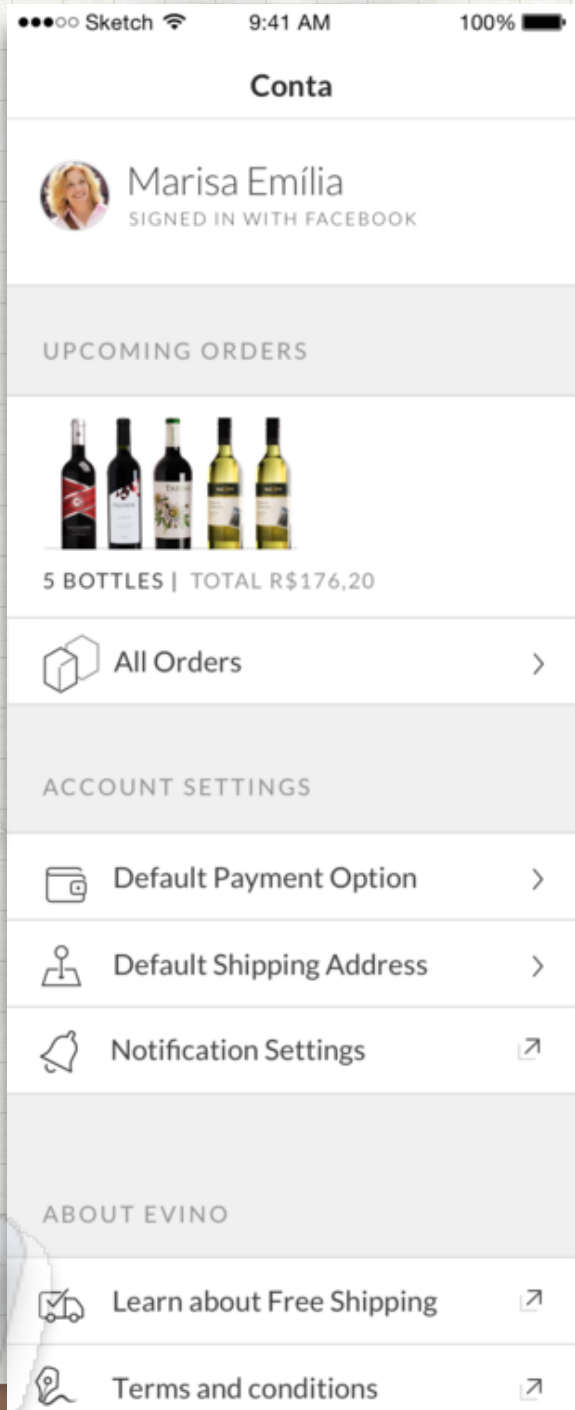
DISADVANTAGES

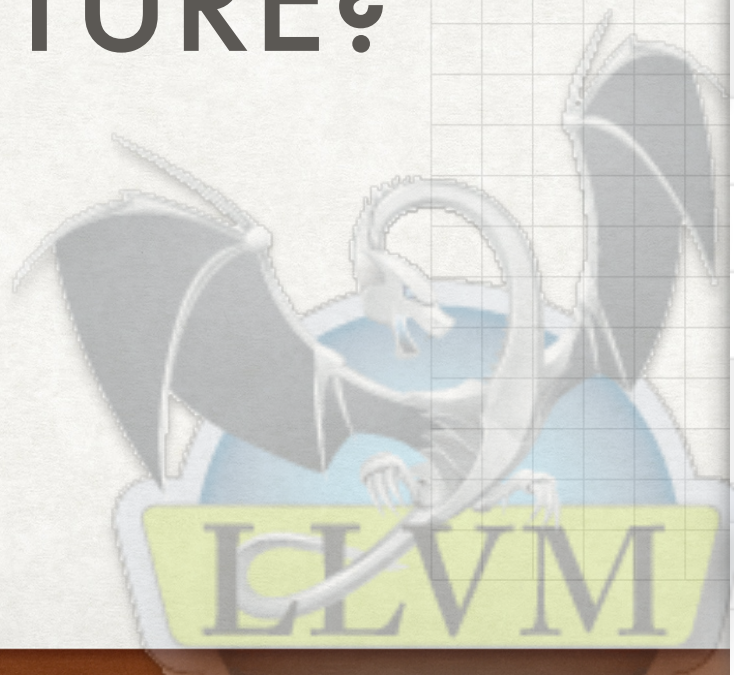
- Error handling not included

```
switch indexPath.section {

case amountPriceSectionNumber:
    return CGSize(width: UIScreen.mainScreen().bounds.width, height: 50)

case bottleCollectionViewSectionNumber:
    return CGSize(width: UIScreen.mainScreen().bounds.width, height: 120)
```

HOW CAN THE
COMPILER HELP US WITH

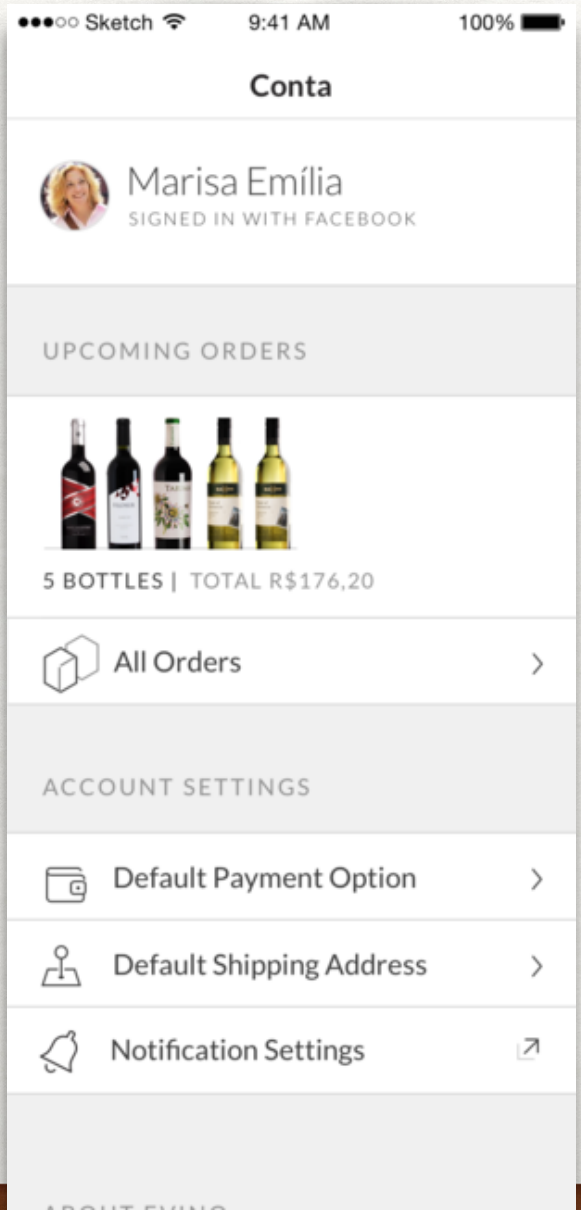
THE LAYOUT STRUCTURE?





ENUMS

```swift
/// Represents the collection view's
/// sections.
private enum Section: Int {
    case UserDetails
    case Orders
    case AccountSettings
}
```



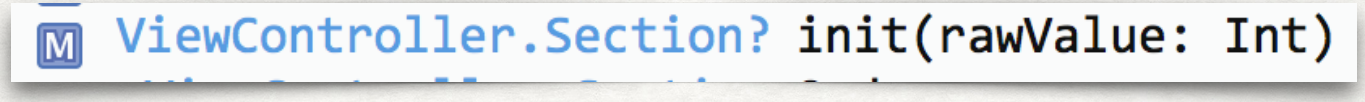
ENUMS

- turn the layout structure into a type

```
/// Represents the collection view's
/// sections.
private enum Section: Int {
    case UserDetails
    case Orders
    case AccountSettings
}
```

ENUMS

NATURAL ERROR-HANDLING




```
//MARK:
//MARK: UICollectionViewDataSource
extension ViewController: UICollectionViewDataSource {
    func collectionView(collectionView: UICollectionView, cellForItemAtIndexPath indexPath: NSIndexPath) ->
        UICollectionViewCell {

        guard let section = Section(rawValue: indexPath.section) else {
            assertionFailure()
            return UICollectionViewCell()
        }
```

ENUMS

EXHAUSTIVITY

```
func collectionView(collectionView: UICollectionView, cellForItemAtIndexPath indexPath: NSIndexPath) ->
    UICollectionViewCell {

    guard let section = Section(rawValue: indexPath.section) else {
        assertionFailure()
        return UICollectionViewCell()
    }

    var cell = collectionView.dequeueReusableCellWithReuseIdentifier(section.reuseIdentifier, forIndexPath: indexPath)

    switch section {

    case .UserDetails:
        cell = cell as! UserDetailsCell
        //configure cell...
        return cell

    case .Orders:
        cell = cell as! OrderCell
        return cell

    case .AccountSettings:
        cell = cell as! AccountSettingsCell
        return cell
    }
}
```

ENUMS

EXHAUSTIVITY

```
switch section {

case .UserDetails:
    let cell = collectionView.dequeueReusableCellWithReuseIdentifier("UserDetailsCell",
        forIndexPath: indexPath)
    return cell

case .Orders:
    let cell = collectionView.dequeueReusableCellWithReuseIdentifier("OrdersCell",
        forIndexPath: indexPath)
    return cell
}
```

Switch must be exhaustive, consider adding a default clause

ENUMS

ENUM FUNCTIONS AND PROPERTIES

```swift
/// Represents the collection view's
/// sections.
private enum Section: Int {
    case UserDetails
    case Orders
    case AccountSettings

    /// The reuse identifier string associated
    /// with this section
    var reuseIdentifier: String {
        switch self {
        case .UserDetails:
            return "UserDetails"
        case .Orders:
            return "Orders"
        case .AccountSettings:
            return "AccountSettings"
        }
    }

    /// The `UICollectionViewCell` subclass associated
    /// with this section
    var cellClass: UICollectionViewCell.Type {
        switch self {
        case .UserDetails:
            return UserDetailsCell.self
        case .Orders:
            return OrderCell.self
        case .AccountSettings:
            return AccountSettingsCell.self
        }
    }
```

```swift
// Look! No hardcoded nothing!
collectionView.registerClass(Section.UserDetails.cellClass, forCellWithReuseIdentifier: Section.UserDetails.reuseIdentifier)
collectionView.registerClass(Section.Orders.cellClass, forCellWithReuseIdentifier: Section.Orders.reuseIdentifier)
collectionView.registerClass(Section.AccountSettings.cellClass, forCellWithReuseIdentifier: Section.AccountSettings.reuseIdentifier)
```

ENUMS

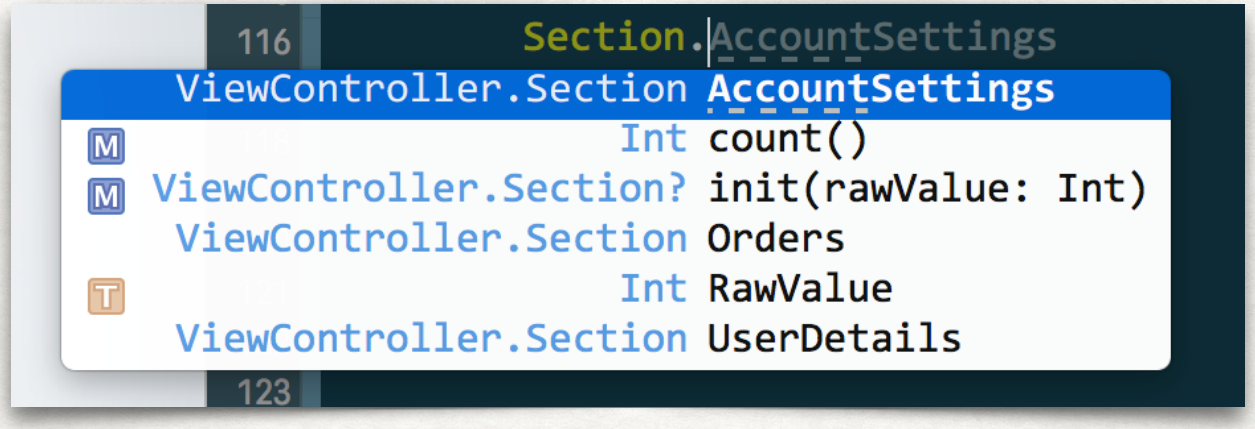
COUNT

```
/// Represents the collection view's
/// sections.
private enum Section: Int {
    case UserDetails
    case Orders
    case AccountSettings

    static func count() -> Int {
        return Section.AccountSettings.rawValue + 1
    }
```

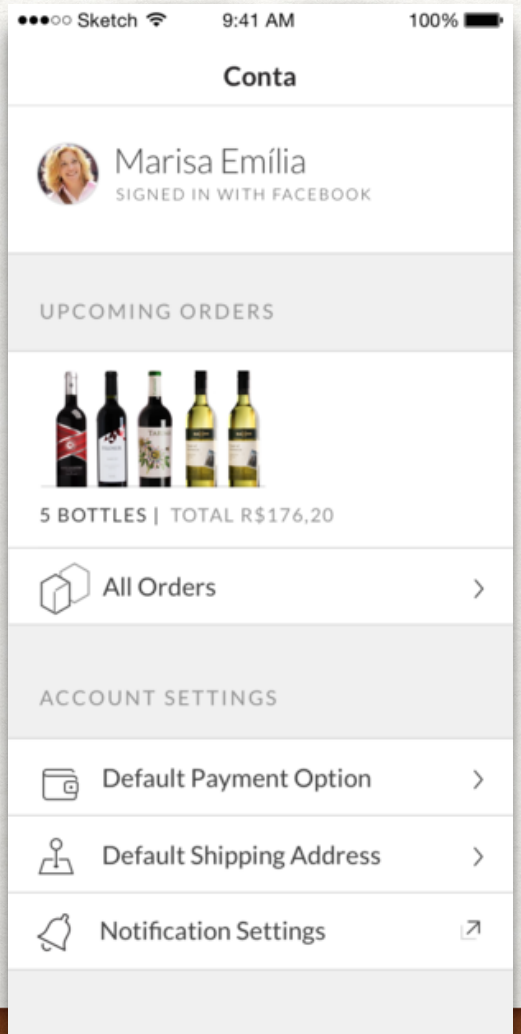
```
func numberOfSectionsInCollectionView(collectionView: UICollectionView) -> Int {
    return Section.count()
}
```

ENUMS

AUTOCOMPLETION GOODNESS



DEMO


Sketch
9:41 AM
100%
Conta
Marisa Emília
SIGNED IN WITH FACEBOOK
UPCOMING ORDERS
5 BOTTLES | TOTAL R$176,20
All Orders
ACCOUNT SETTINGS
Default Payment Option
Default Shipping Address
Notification Settings

ENUMS

FAVOUR DECOUPLING

```
enum Section : Int {
    case Dough
    case Ingredients

    func title() -> String? {
        switch self {
        case .Dough: return NSLocalizedString("section_title_dough", comment: "")
        case .Ingredients: return NSLocalizedString("section_title_ingredients", comment: "")
        }
    }
}
```

THANK YOU!

```
enum ThankYou {
    case VeryMuch
}
```

QUESTIONS?

```
enum Question {
    case Easy
    case Hard
    case WhereIsTheSampleCode
    case WhereCanIGetTheSlides

    var url: NSURL {
        switch self {

        case .WhereIsTheSampleCode, .WhereCanIGetTheSlides :
            return NSURL(string: "http://bit.ly/enums_cv_samplecode")!

        default:
            return NSURL(string: "http://bit.ly/enums_collectionviews")!
        }
    }
}
```