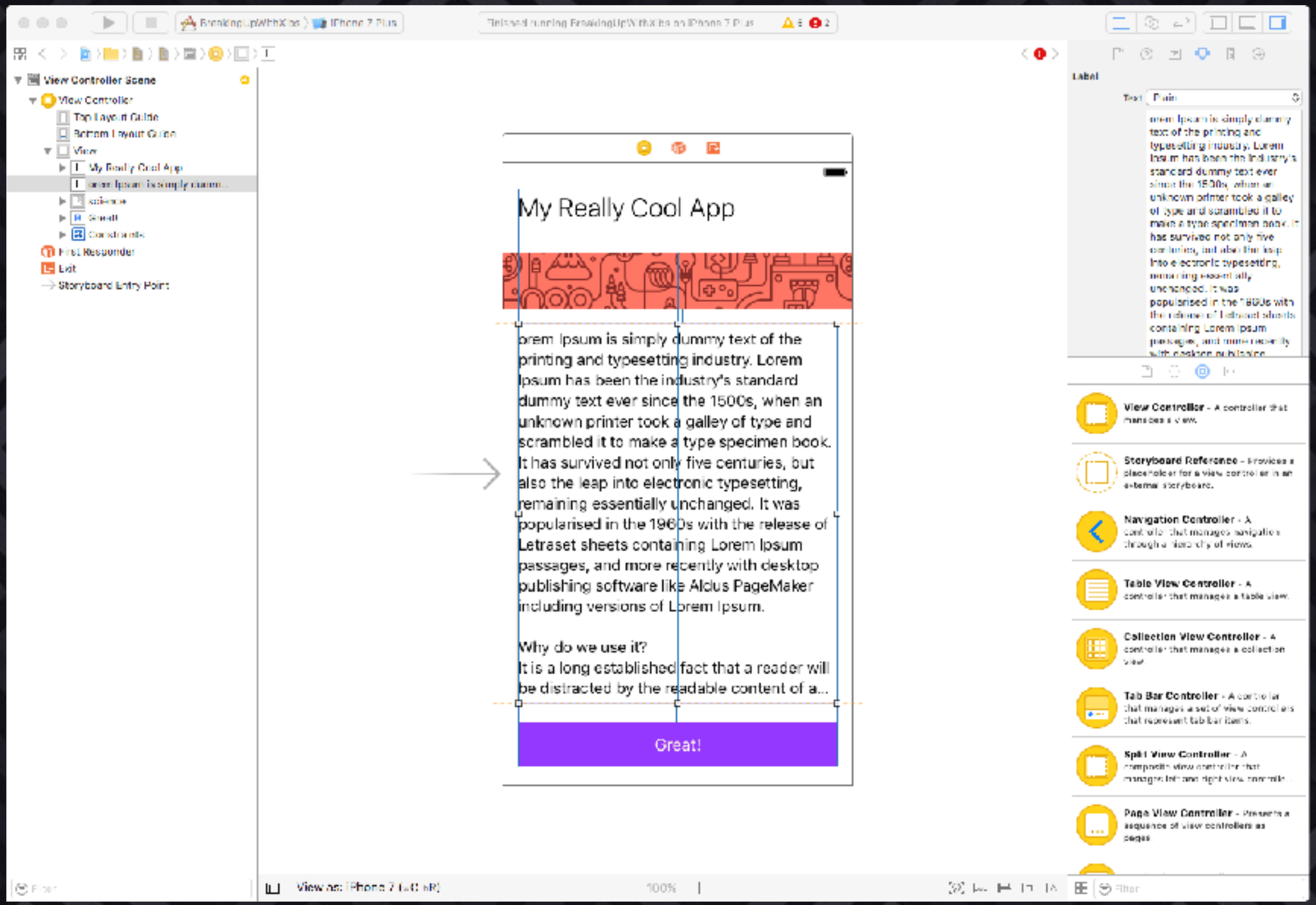
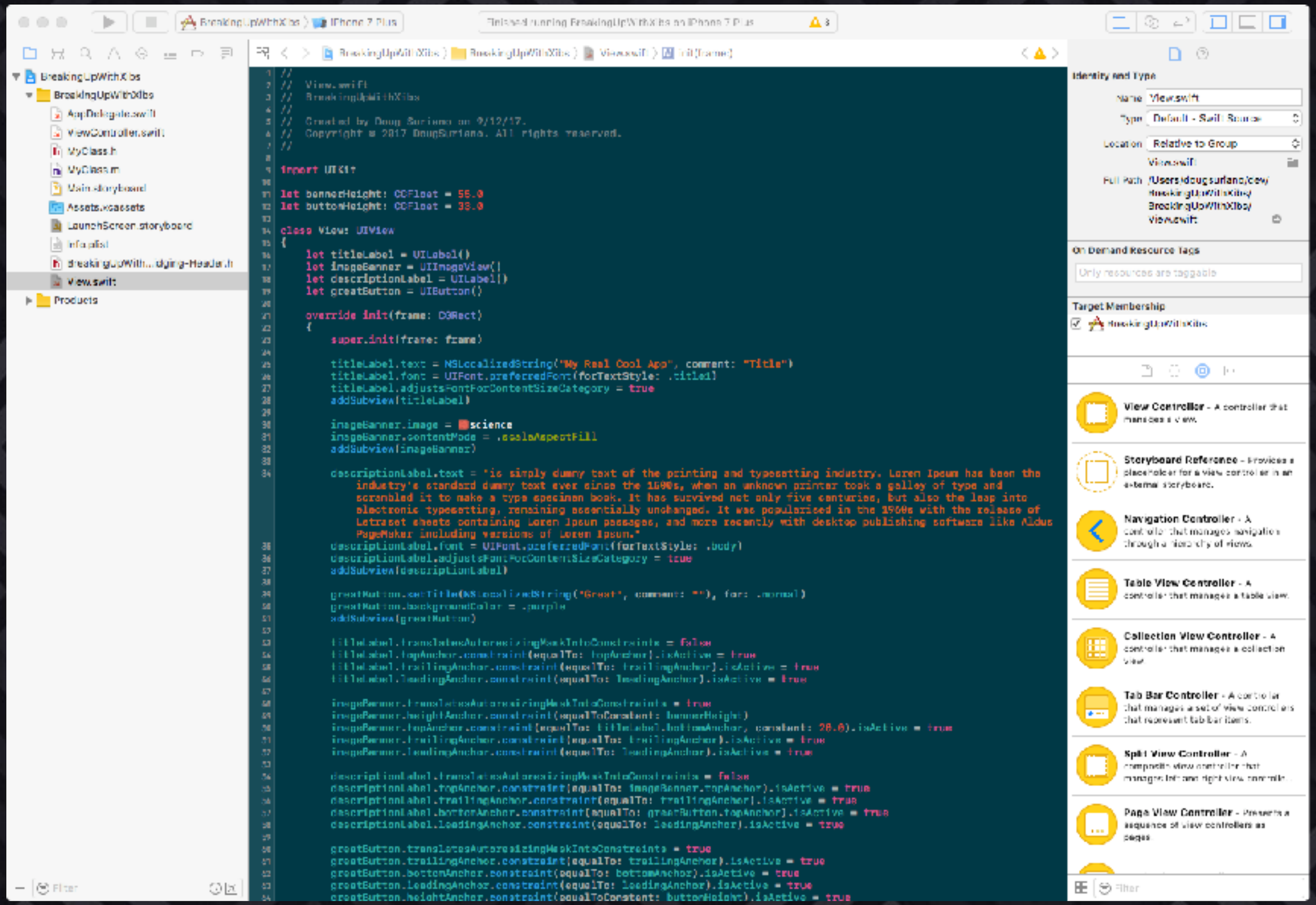
It's Not You, It's Me.

Breaking up with Xibs and Storyboards.


View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
My Really Cool App
First Responder
Exit
Storyboard Entry Point
My Really Cool App
orem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.
Why do we use it?
It is a long established fact that a reader will be distracted by the readable content of a...
Great!
Label
Text
View Controller – A controller that manages a view.
Storyboard Reference – Provides a placeholder for a view controller in an external storyboard.
Navigation Controller – A controller that manages navigation through a hierarchy of views.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Tab Bar Controller – A controller that manages a set of view controllers that represent tab bar items.
Page View Controller – Presents a sequence of view controllers as pages.


BreakingUpWithXibs
BreakingUpWithXibs
AppDelegate.swift
ViewController.swift
MyClass.h
MyClass.m
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
BreakingUpWith...dging-Header.h
View.swift
Products
//
// View.swift
// BreakingUpWithXibs
//
// Created by Doug Sartiano on 9/12/17.
// Copyright © 2017 DougSartiano. All rights reserved.
//
import UIKit
let bannerHeight: CGFloat = 50.0
let buttonHeight: CGFloat = 30.0
class View: UIView
{
let titleLabel = UILabel()
let imageBanner = UIImageView()
let descriptionLabel = UILabel()
let greatButton = UIButton()
override init(frame: CGRect)
{
super.init(frame: frame)
titleLabel.text = NSLocalizedString("My Real Cool App", comment: "Title")
titleLabel.font = UIFont.preferredFont(forTextStyle: .title1)
titleLabel.adjustsFontForContentSizeCategory = true
addSubview(titleLabel)
imageBanner.image = science
imageBanner.contentMode = .scaleAspectFill
addSubview(imageBanner)
descriptionLabel.text = "is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularized in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum."
descriptionLabel.font = UIFont.preferredFont(forTextStyle: .body)
descriptionLabel.adjustsFontForContentSizeCategory = true
addSubview(descriptionLabel)
greatButton.setTitle(NSLocalizedString("Great", comment: ""), for: .normal)
greatButton.backgroundColor = .purple
addSubview(greatButton)
titleLabel.translatesAutoresizingMaskIntoConstraints = false
titleLabel.topAnchor.constraint(equalTo: topAnchor).isActive = true
titleLabel.trailingAnchor.constraint(equalTo: trailingAnchor).isActive = true
titleLabel.leadingAnchor.constraint(equalTo: leadingAnchor).isActive = true
imageBanner.translatesAutoresizingMaskIntoConstraints = true
imageBanner.heightAnchor.constraint(equalToConstant: bannerHeight)
imageBanner.topAnchor.constraint(equalTo: titleLabel.bottomAnchor, constant: 20.0).isActive = true
imageBanner.trailingAnchor.constraint(equalTo: trailingAnchor).isActive = true
imageBanner.leadingAnchor.constraint(equalTo: leadingAnchor).isActive = true
descriptionLabel.translatesAutoresizingMaskIntoConstraints = false
descriptionLabel.topAnchor.constraint(equalTo: imageBanner.topAnchor).isActive = true
descriptionLabel.trailingAnchor.constraint(equalTo: trailingAnchor).isActive = true
descriptionLabel.bottomAnchor.constraint(equalTo: greatButton.topAnchor).isActive = true
descriptionLabel.leadingAnchor.constraint(equalTo: leadingAnchor).isActive = true
greatButton.translatesAutoresizingMaskIntoConstraints = true
greatButton.trailingAnchor.constraint(equalTo: trailingAnchor).isActive = true
greatButton.bottomAnchor.constraint(equalTo: bottomAnchor).isActive = true
greatButton.leadingAnchor.constraint(equalTo: leadingAnchor).isActive = true
greatButton.heightAnchor.constraint(equalToConstant: buttonHeight).isActive = true
Identity and Type
Name View.swift
Type Default - Swift Source
Location Relative to Group
View.swift
Full Path /Users/doug.sartiano/dev/BreakingUpWithXibs/BreakingUpWithXibs/View.swift
On Demand Resource Tags
Only resources are taggable
Target Membership
BreakingUpWithXibs
View Controller - A controller that manages a view.
Storyboard Reference - Provides a placeholder for a view controller in an external storyboard.
Navigation Controller - A controller that manages navigation through a hierarchy of views.
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a collection view.
Tab Bar Controller - A controller that manages a set of view controllers that represent tab bar items.
Split View Controller - A composite view controller that manages left and right view controllers.
Page View Controller - Presents a sequence of view controllers as pages.
Filter

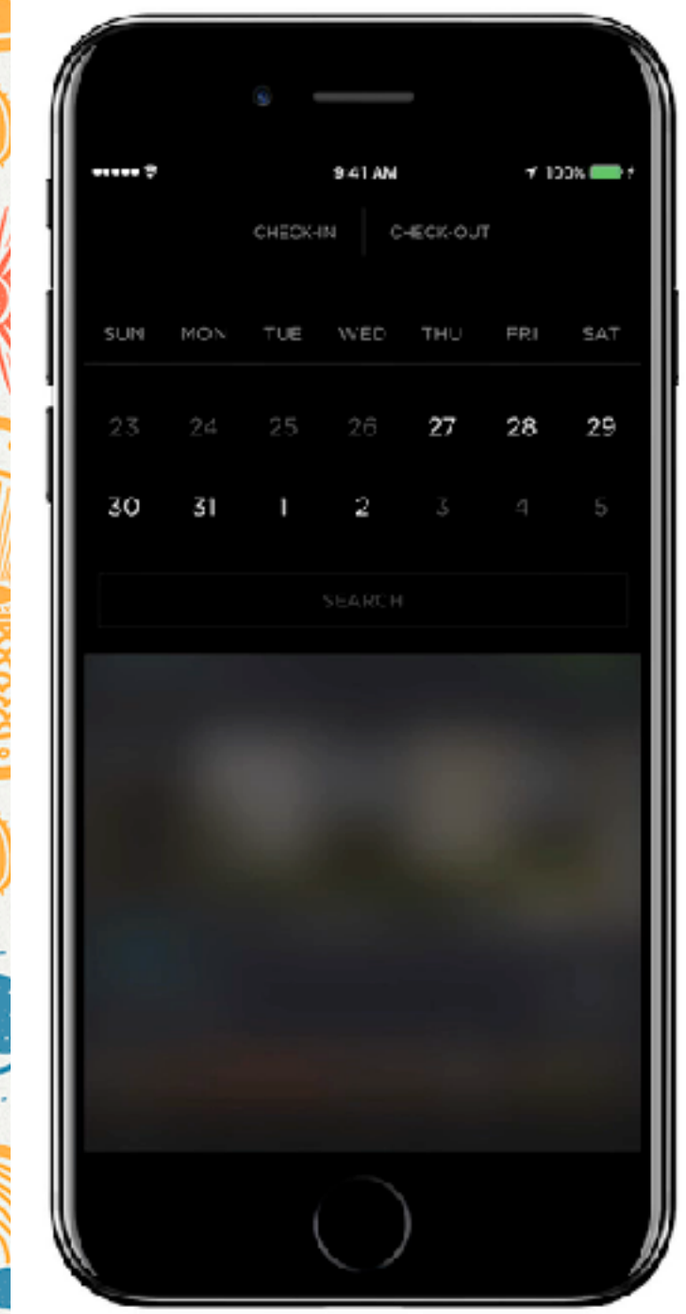
Tonight

- Some advantages of using programmatic layout.
- Available Layout Systems
- Put it together - Creating a view controller with programatic layout.
- Making your life easier.
- Question Time!

Doug

Suriano.

Doug Suriano.



Advantages

Advantages

1. Constants!
2. Code Reviewability
3. Swift!
4. Project Scalability

Constants

- Magic Numbers are a code smell.

```
if error.code == 612
{
    // Handle the error
}
```

Constants

- Magic Numbers are a code smell.

```
if error.code == kInvalidStartDate
{
    // Handle the error
}
```

Constants

- Spacing, Padding, Inset, Sizing float values.
- Colors
- Fonts

```
myConstraint.constant = 12.0

UIColor(red: 192.0 / 255.0, green: 57.0/255.0, blue: 43.0 / 255, alpha: 1.0)

UIFont(name: "Gotham-Book", size: 14.0)
```

Constants

- Spacing, Padding, Inset, Sizing float values.
- Colors
- Fonts

```
myConstraint.constant = kInset

UIColor.pomegranate

UIFont.gothamBook
```

Constants in Xibs/ Storyboards

Values for spacing, sizings, insets.	**Nope.**
Colors	**In Swift #colorLiteral**
Fonts	**If you use Apple's standard Fonts**

Code Review

- If you work with at least one other iOS engineer, you should require that every change to your codebase go through code review.
- Getting your code reviewed makes you a better engineer.
- Reviewing other engineer's code also makes you a better engineer.

Code Review

- Swift and Objective-C are easier to review than XML.
- Code Review can catch problematic constraints and structure earlier.

Swift

The complier is looking out for us!

```
188 {
189     NSString *string = [NSString stringWithString:@42];
190
```

```
51
52     let string = String(stringLiteral: NSNumber(integerLiteral: 42))
53
```

Swift

- Interface builder was built for an Objective-C Dynamic World.
- The Swift Complier won’t help make sure your storyboard/xibs files are bug free.

Scalability

- As your project and team grows, your Programmatic layout code will scale better with your growth.
- More easily enforce separation of concerns
- Merge conflicts easier to resolve.

Choosing a layout method

Layout Methods.

1. Manual Layout
2. Auto Layout using UIKit
3. Auto Layout using 3rd party SDK
4. Custom Layout outside of UIKit.

Manual Layout

```
let myView = UIView()
view.addSubview(myView)

myView.frame = CGRect(x: 10.0, y: 10.0, width: 100.0, height: 70.0)
```

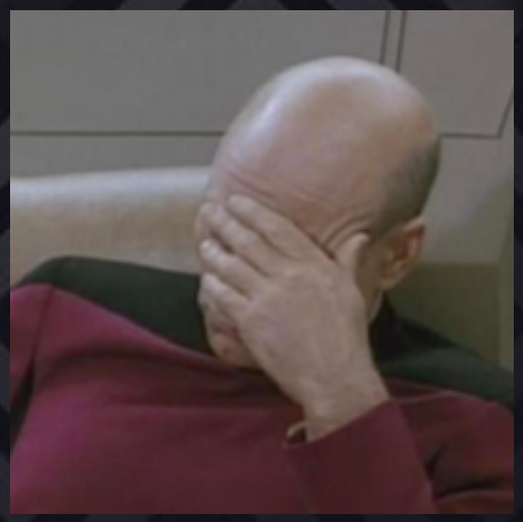
Manual Layout - Advantages

- Easy to implement basic views.
- Fast Performance.
- Matches the language designers use.
- Lots of flexibility.

Manual Layout - Disadvantages

- More work to make code work:
  - in multiple screen sizes
  - Dynamic Text
  - iPad multi-tasking
- Your sizing logic (`sizeThatFits:`) is different from your layout logic (`layoutSubviews`)
- Your layout code can quickly become very complicated.
- Extremely easy to write bad layout code.

UIKit Programmatic Auto Layout



Visual Format Language

NSLayoutConstraint

NSLayoutAnchor

VFL

```
UIView *myView = [[UIView alloc] initWithFrame:CGRectZero];
myView.translatesAutoresizingMaskIntoConstraints = NO;

[self addSubview:myView];

NSDictionary *views = NSDictionaryOfVariableBindings(myView);

NSArray *hConstraints;
NSArray *vConstraints;

hConstraints = [NSLayoutConstraint constraintsWithVisualFormat:@"H:|-10-[myView(100)]"
                                                       options:0
                                                       metrics:nil
                                                         views:views];

vConstraints = [NSLayoutConstraint constraintsWithVisualFormat:@“V:|-10-[myView(70)]"
                                                       options:0
                                                       metrics:nil
                                                         views:views];

[self addConstraints:hConstraints];
[self addConstraints:vConstraints];
```

(Objective-C for dramatic effect.)

NSLayoutConstraint

```
let myView = UIView()
view.addSubview(myView)
view.translatesAutoresizingMaskIntoConstraints = false

let constraints = [
    NSLayoutConstraint(item: myView,
                       attribute: .leading,
                       relatedBy: .equal,
                       toItem: view,
                       attribute: .leading,
                       multiplier: 1.0,
                       constant: 10.0),
    NSLayoutConstraint(item: myView,
                       attribute: .top,
                       relatedBy: .equal,
                       toItem: view,
                       attribute: .top,
                       multiplier: 1.0,
                       constant: 10.0),
    NSLayoutConstraint(item: myView,
                       attribute: .width,
                       relatedBy: .equal,
                       toItem: nil,
                       attribute: .notAnAttribute,
                       multiplier: 1.0,
                       constant: 100.0),
    NSLayoutConstraint(item: myView,
                       attribute: .height,
```

```
let constraints = [
    NSLayoutConstraint(item: myView,
                       attribute: .leading,
                       relatedBy: .equal,
                       toItem: view,
                       attribute: .leading,
                       multiplier: 1.0,
                       constant: 10.0),
    NSLayoutConstraint(item: myView,
                       attribute: .top,
                       relatedBy: .equal,
                       toItem: view,
                       attribute: .top,
                       multiplier: 1.0,
                       constant: 10.0),
    NSLayoutConstraint(item: myView,
                       attribute: .width,
                       relatedBy: .equal,
                       toItem: nil,
                       attribute: .notAnAttribute,
                       multiplier: 1.0,
                       constant: 100.0),
    NSLayoutConstraint(item: myView,
                       attribute: .height,
                       relatedBy: .equal,
                       toItem: nil,
                       attribute: .notAnAttribute,
                       multiplier: 1.0,
                       constant: 70.0),
]

view.addConstraints(constraints)
```

Layout Anchors

```
let myView = UIView()
view.addSubview(myView)
myView.translatesAutoresizingMaskIntoConstraints = false

myView.leadingAnchor.constraint(equalTo: view.leadingAnchor, constant: 10.0)
myView.topAnchor.constraint(equalTo: view.topAnchor, constant: 10.0)
myView.widthAnchor.constraint(equalToConstant: 100.0)
myView.heightAnchor.constraint(equalToConstant: 70.0)
```

Auto Layout - Advantages

- "Set it and forget it" - Your constraints take care of both layout and sizing.
- Resizing, dynamic text, multiple screen sizes comes out of the box.
- Syntax with layout anchors….not bad

Auto Layout - Disadvantages

- Frustrating - the DMV of UIKit.
- Requires you to shift your thinking.
- Confusing error handling
- Lots of constraints == Poor Performance.

Auto Layout - 3rd party

- Examples include Masonry, SnapKit, and Cartography.
- Most of these libraries aim to “fix” auto layout, especially the VFL and NSLayoutConstraint Approach.

```objc
[view1 mas_makeConstraints:^(MASConstraintMaker *make) {
    make.edges.equalTo(superview).with.insets(padding);
}];
```

```swift
self.view.addSubview(box)
box.snp.makeConstraints { (make) -> Void in
   make.width.height.equalTo(50)
   make.center.equalTo(self.view)
}
```

```swift
constrain(view1, view2) { view1, view2 in
    view1.width   == (view1.superview!.width - 50) * 0.5
    view2.width   == view1.width - 50
    view1.height  == 40
    view2.height  == view1.height
    view1.centerX == view1.superview!.centerX
    view2.centerX == view1.centerX

    view1.top >= view1.superview!.top + 20
    view2.top == view1.bottom + 20
}
```

Custom Layout

- Replace UIKit
- Examples include Async Display Kit, LayoutKit.
- Typically backed by large corporation.

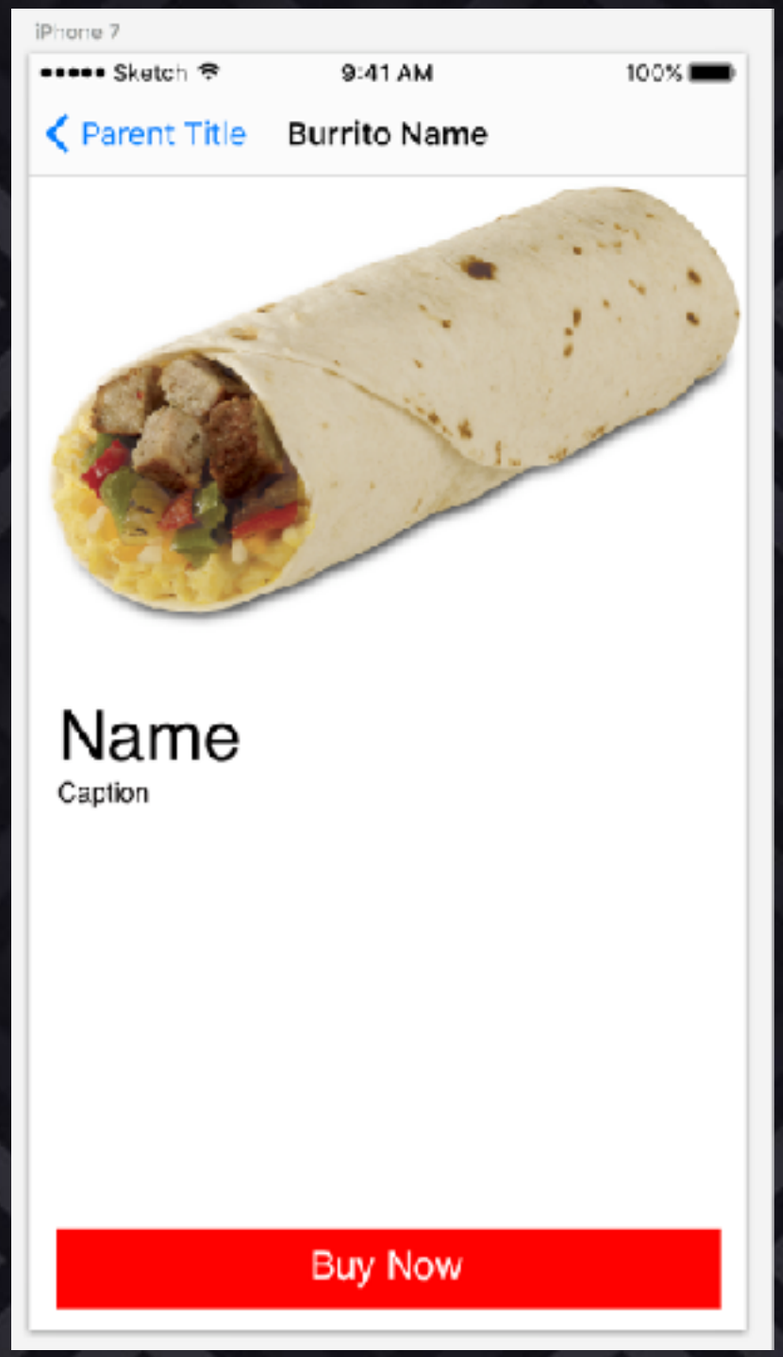
What I Recommend

Auto Layout - UIKit	Yes - (With Layout Anchors!)
Manual Layout	Yes - When needed.
Auto Layout - 3rd party library	Not anymore - Layout Anchors saved the day.
Custom Layout	No - 99% of apps are better suited with sticking with UIKit.

Let’s build a View Controller!

What are we building?

- **Details page for a shopping app.**
  - `BRTNDetailsViewController`
- **We need:**
  - **1 x** `UIImageView`
  - **2 x** `UILabel`
  - **1 x** `UIButton`
- **Let's use layout anchors.**



```swift
//

import UIKit

fileprivate struct Constants
{
    static let cellReuseIdentifier = "cellReuseIdentifier"
}

class BRTNMenuViewController: UITableViewController
{
    fileprivate var burritos = [BRTNBurrito]()

    override func viewDidLoad()
    {
        super.viewDidLoad()

        title = "BurritoTonight"

        burritos = BRTNBurritoProvider.mockBurritos

        tableView.register(UITableViewCell.self, forCellReuseIdentifier: Constants.cellReuseIdentifier)

        tableView.reloadData()
    }

    override func numberOfSections(in tableView: UITableView) -> Int
    {
        return 1
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int
    {
        return burritos.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
    {
        guard let cell = tableView.dequeueReusableCell(withIdentifier: Constants.cellReuseIdentifier) else
        {
            fatalError("This should never happen")
        }

        let burrito = burritos[indexPath.row]
        cell.textLabel?.text = burrito.name
        cell.imageView?.image = burrito.image

        return cell
    }
}
```

```swift
//
//  BRTNDetailsViewController.swift
//  BurritoTonight
//
//  Created by Doug Suriano on 9/12/17.
//  Copyright © 2017 DougSuriano. All rights reserved.
//

import UIKit

class BRTNDetailsViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }


    /*
    // MARK: - Navigation

    // In a storyboard-based application, you will often want to do a little preparation before navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // Get the new view controller using segue.destinationViewController.
        // Pass the selected object to the new view controller.
    }
    */

}
```

```swift
//
//  BRTNDetailsViewController.swift
//  BurritoTonight
//
//  Created by Doug Suriano on 9/12/17.
//  Copyright © 2017 DougSuriano. All rights reserved.
//

import UIKit

class BRTNDetailsViewController: UIViewController
{
    let burrito: BRTNBurrito

    required init(burrito: BRTNBurrito)
    {
        self.burrito = burrito

        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder)
    {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        view.backgroundColor = .white
    }

}
```

```swift
//
//  BRTNDetailsViewController.swift
//  BurritoTonight
//
//  Created by Doug Suriano on 9/12/17.
//  Copyright © 2017 DougSuriano. All rights reserved.
//

import UIKit

class BRTNDetailsViewController: UIViewController
{
    let burrito: BRTNBurrito

    required init(burrito: BRTNBurrito)
    {
        self.burrito = burrito

        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder)
    {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        view.backgroundColor = .white
    }

}
```


Xcode File Edit View Find Navigate Editor
Tue Sep 12 8:07:52 PM Doug Suriano
BurritoTonight
iPhone 7
BurritoTonight
BRTNDetailsViewController
BurritoTonight
Products
BurritoTonight.app
//
// BRTNDetailsV
// BurritoTonigh
//
// Created by Doug
// Copyright © 2017
served.
//
import UIKit
class BRTNDetailsViewCont
{
let burrito: BRTNBurrit
fileprivate var imageView
fileprivate var titleLabel
fileprivate var captionLabe
fileprivate var buyButton: U
required init(burrito: BRTNBurr
{
self.burrito = burrito
super.init(nibName: nil, bun
}
required init?(coder aDecoder
{
fatalError("init(coder:
}
override func viewDidLo
super.viewDidLoad()
view.backgroundC
}
}
TONIGHT BELONGS TO YOU

“Friends don’t let friends force unwrap properties.”

```swift
//
//  BRTNDetailsViewController.swift
//  BurritoTonight
//
//  Created by Doug Suriano on 9/12/17.
//  Copyright © 2017 DougSuriano. All rights reserved.
//

import UIKit

class BRTNDetailsViewController: UIViewController
{
    let burrito: BRTNBurrito

    fileprivate var imageView: UIImageView!
    fileprivate var titleLabel: UILabel!
    fileprivate var captionLabel: UILabel!
    fileprivate var buyButton: UIButton!

    required init(burrito: BRTNBurrito)
    {
        self.burrito = burrito

        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder)
    {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        view.backgroundColor = .white
    }

}
```

```swift
//
//  BRTNDetailsViewController.swift
//  BurritoTonight
//
//  Created by Doug Suriano on 9/12/17.
//  Copyright © 2017 DougSuriano. All rights reserved.
//

import UIKit

class BRTNDetailsViewController: UIViewController
{
    let burrito: BRTNBurrito

    fileprivate lazy var imageView = UIImageView()
    fileprivate lazy var titleLabel = UILabel()
    fileprivate lazy var captionLabel = UILabel()
    fileprivate lazy var buyButton = UIButton()

    required init(burrito: BRTNBurrito)
    {
        self.burrito = burrito

        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder)
    {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        view.backgroundColor = .white
    }

}
```

takes care of sizing, spacing and other attributes to make the text legible for all text styles and sizes. For somebody like me who is font challenged this is most

```swift

class BRTNDetailsViewController: UIViewController
{
    let burrito: BRTNBurrito

    fileprivate lazy var imageView = UIImageView()
    fileprivate lazy var titleLabel = UILabel()
    fileprivate lazy var captionLabel = UILabel()
    fileprivate lazy var buyButton = UIButton()

    required init(burrito: BRTNBurrito)
    {
        self.burrito = burrito

        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder)
    {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        view.backgroundColor = .white

        imageView.image = burrito.image
        imageView.contentMode = .scaleAspectFill
        view.addSubview(imageView)

        titleLabel.text = burrito.name
        titleLabel.font = UIFont.preferredFont(forTextStyle: .title1)
        titleLabel.adjustsFontForContentSizeCategory = true
        view.addSubview(titleLabel)

        captionLabel.text = burrito.caption
        captionLabel.font = UIFont.preferredFont(forTextStyle: .caption1)
        captionLabel.adjustsFontForContentSizeCategory = true
        captionLabel.numberOfLines = 0
        view.addSubview(captionLabel)

        buyButton.setTitle("Buy Now", for: .normal)
        buyButton.setTitleColor(.white, for: .normal)
        buyButton.backgroundColor = .red
        view.addSubview(buyButton)

    }

}
```

```
        titleLabel.text = burrito.name
        titleLabel.font = UIFont.preferredFont(forTextStyle: .title1)
        titleLabel.adjustsFontForContentSizeCategory = true
        view.addSubview(titleLabel)

        captionLabel.text = burrito.caption
        captionLabel.font = UIFont.preferredFont(forTextStyle: .caption1)
        captionLabel.adjustsFontForContentSizeCategory = true
        captionLabel.numberOfLines = 0
        view.addSubview(captionLabel)

        view.addSubview(spacerView)

        buyButton.setTitle("Buy Now", for: .normal)
        buyButton.setTitleColor(.white, for: .normal)
        buyButton.backgroundColor = .red
        view.addSubview(buyButton)

        imageView.translatesAutoresizingMaskIntoConstraints = false
        imageView.topAnchor.constraint(equalTo: view.topAnchor).isActive = true
        imageView.trailingAnchor.constraint(equalTo: view.trailingAnchor).isActive = true
        imageView.leadingAnchor.constraint(equalTo: view.leadingAnchor).isActive = true
        imageView.heightAnchor.constraint(equalTo: view.heightAnchor, multiplier: 0.5).isActive = true

        titleLabel.translatesAutoresizingMaskIntoConstraints = false
        titleLabel.topAnchor.constraint(equalTo: imageView.bottomAnchor, constant: 10.0).isActive = true
        titleLabel.trailingAnchor.constraint(equalTo: view.trailingAnchor, constant: 10.0).isActive = true
        titleLabel.leadingAnchor.constraint(equalTo: view.leadingAnchor, constant: 10.0).isActive = true

        captionLabel.translatesAutoresizingMaskIntoConstraints = false
        captionLabel.topAnchor.constraint(equalTo: titleLabel.bottomAnchor, constant: 10.0).isActive = true
        captionLabel.trailingAnchor.constraint(equalTo: view.trailingAnchor, constant: -10.0).isActive = true
        captionLabel.bottomAnchor.constraint(equalTo: spacerView.topAnchor, constant: -10.0).isActive = true
        captionLabel.leadingAnchor.constraint(equalTo: view.leadingAnchor, constant: 10.0).isActive = true

        spacerView.translatesAutoresizingMaskIntoConstraints = false
        spacerView.trailingAnchor.constraint(equalTo: view.trailingAnchor).isActive = true
        spacerView.bottomAnchor.constraint(equalTo: buyButton.topAnchor).isActive = true
        spacerView.leadingAnchor.constraint(equalTo: view.leadingAnchor).isActive = true

        buyButton.translatesAutoresizingMaskIntoConstraints = false
        buyButton.heightAnchor.constraint(equalToConstant: 55.0).isActive = true
        buyButton.trailingAnchor.constraint(equalTo: view.trailingAnchor, constant: -10.0).isActive = true
        buyButton.bottomAnchor.constraint(equalTo: view.bottomAnchor, constant: -10.0).isActive = true
        buyButton.leadingAnchor.constraint(equalTo: view.leadingAnchor, constant: 10.0).isActive = true

    }

}
```

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help Internal

Tue Sep 12 9:14:01 PM Doug Suriano

BurritoTonight

Show in Finder
Open with External Editor
Open As
Show File Inspector
New File...
Add Files to "BurritoTonight"...
Delete
New Group
New Group from Selection
Sort by Name
Sort by Type
Find in Selected Groups...
Source Control
Project Navigator Help

```swift
//
//  BRTNDetailsViewController.swift
//  BurritoTonight
//
//  Created by Doug Suriano on 9/12/17.
//  Copyright © 2017 DougSuriano. All rights reserved.
//

import UIKit

fileprivate struct Constants

    static let imageHeightRatio: CGFloat = 0.5
    static let inset: CGFloat = 10.0
    static let buttonHeight: CGFloat = 55.0


class BRTNDetailsViewController: UIViewController
{
    let burrito: BRTNBurrito

    fileprivate lazy var imageView = UIImageView()
    fileprivate lazy var titleLabel = UILabel()
    fileprivate lazy var captionLabel = UILabel()
    fileprivate lazy var spacerView = UIView()
    fileprivate lazy var buyButton = UIButton()

    required init(burrito: BRTNBurrito)
    {
        self.burrito = burrito

        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder)
    {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        view.backgroundColor = .white

        imageView.image = burrito.image
        imageView.contentMode = .scaleAspectFill
        view.addSubview(imageView)

        titleLabel.text = burrito.name
        titleLabel.font = UIFont.preferredFont(forTextStyle: .title1)
```

Making your life easier.

Extensions (And Categories) are your friend!

- I believe strongly in sticking close to UIKit.
- …but UIKit is built in Objective-C which allows you to extend functionality of classes without subclassing easily in both Swift and Obj-C!
- Best of both worlds; close to the metal but optimized for your project's needs.

autoresizingMask

```
myView.translatesAutoresizingMaskIntoConstraints = false
```

```
extension UIView
{
    func enableAutoLayout()
    {
        translatesAutoresizingMaskIntoConstraints = false
    }
}
```

```
myView.enableAutoLayout()
```

Activate

```
v.leadingAnchor.constraint(equalTo: view.topAnchor, constant: 10.0).isActive = true
```

```
extension NSLayoutConstraint
{
    @discardableResult func activate() -> NSLayoutConstraint
    {
        isActive = true
        return self
    }
}
```

```
v.leadingAnchor.constraint(equalTo: view.topAnchor, constant: 10.0).activate()
```

Pin to Top of View

```
v.enableAutoLayout()
v.topAnchor.constraint(equalTo: view.topAnchor).activate()
v.trailingAnchor.constraint(equalTo: view.trailingAnchor).activate()
v.bottomAnchor.constraint(equalTo: otherView.bottomAnchor).activate()
v.leadingAnchor.constraint(equalTo: view.leadingAnchor).activate()
```

```
extension UIView
{
    func fixToTopEdges(of view: UIView)
    {
        view.enableAutoLayout()
        view.topAnchor.constraint(equalTo: view.topAnchor).activate()
        view.trailingAnchor.constraint(equalTo: view.trailingAnchor).activate()
        view.leadingAnchor.constraint(equalTo: view.leadingAnchor).activate()
    }
}
```

```
v.fixToTopEdges(of: view)
v.bottomAnchor.constraint(equalTo: otherView.bottomAnchor).activate()
```

Invest in a view debugger!

- Xcode's?
- Commercial Options
  - Reveal
  - Spark Inspector

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help Internal

Tue Sep 12 9:31:27 PM Doug Surfano

iPhone 7 – iOS 10.3 (14E269)

Carrier 9:31 PM

BurritoTonight

Chicken

Steak

Beef

Veggie

BurritoTonight on iPhone Simulator (iOS 10.3)

UIScreen

UIScreen
UIWindow
UILayoutContainerView
UINavigationTransitionView
UIViewControllerWrapperView
UITableView
UITableViewWrapperView
UITableViewCell
UITableViewCellContentView

Carrier 9:30 PM

BurritoTonight

No Selection

BurritoTonight > iPhone 7

Finished running BurritoTonight on iPhone 7

BurritoTonight > BurritoTonight > BRTNDetailsViewController.swift > viewDidLoad()

```swift
    lazy var detailView: BRTNDetailsView = {
        let view = BRTNDetailsView()
        return view
    }()

    required init(burrito: BRTNBurrito)
    {
        self.burrito = burrito

        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder aDecoder: NSCoder)
    {
        fatalError("init(coder:) has not been implemented")
    }

    override func loadView()
    {
        self.view = detailView
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        view.backgroundColor = .white

        detailView.titleLabel.text = burrito.name
        detailView.captionLabel.text = burrito.caption
        detailView.imageView.image = burrito.image
    }

}
```

Identity and Type

Name: BRTNDetailsViewController.swift

Type: Default - Swift Source

Location: Relative to Group

BRTNDetailsViewController.swift

Full Path: /Users/dougsurfano/Desktop/breaking-up-with-the-storyboard/Burritotonight/BurritoTonight/BRTNDetailsViewController.swift

On Demand Resource Tags

Only resources are taggable

Target Membership

BurritoTonight

Text Settings

Text Encoding: Unicode (UTF-8)

Line Endings: Default - macOS / Unix (LF)

Indent Using: Spaces

Widths: Tab 4 Indent 4

Wrap lines

No Matches

BurritoTonight[2154:115073] [MC] Reading from private effective user settings.

Auto

All Output

Don’t forget UIStackView

- Easy as hell!
- Saves time
- Less lines of code.

```
let stackView = UIStackView()
stackView.axis = .horizontal
stackView.spacing = kPadding
view.addSubview(stackView)

stackView.addArrangedSubview(titleLabel)
stackView.addArrangedSubview(descriptionLabel)
stackView.addArrangedSubview(button)
```

That's it!

github.com/dougsuriano/
breakinguptalk

@dougsuriano

HotelTonight Promo Code: PIZZAPARTY

Questions?