



AUTO LAYOUT BY EXAMPLE

Nick Chen
Fall 2015
talentspark.io



AUTO LAYOUT BY EXAMPLE

Nick Chen
Fall 2015
talentspark.io

“ANY SUFFICIENTLY ADVANCED
TECHNOLOGY IS
INDISTINGUISHABLE FROM MAGIC”

– Arthur C. Clarke

HOW DOES IT WORK?

The screenshot shows a web browser window with the URL `overconstrained.io` in the address bar. The page itself has a dark header with the word **OVERCONSTRAINED** in white. In the top right corner of the header, there is a white button with the word **ABOUT** in yellow. To the right of this button are links for **PROJECTS** and **SLACK**. The main content area features a large title **CASSOWARY** in bold black letters, centered above a horizontal line with a star in the middle. Below the title, there is a paragraph of text describing what Cassowary is and how it works. At the bottom, another paragraph provides information about who developed the toolkit and its applications.

CASSOWARY

Cassowary is an incremental constraint solving toolkit that efficiently solves systems of linear equalities and inequalities. Constraints may be either requirements or preferences. Client code specifies the constraints to be maintained, and the solver updates the constrained variables to have values that satisfy the constraints.

Cassowary was developed by Greg Badros and Alan Borning, and was optimized for user interface applications. Badros used Cassowary amongst others for implementing Constraint Cascading Style Sheets (CCSS), an extension to Cascading Style Sheets (CSS).

HOW DOES IT WORK?

The screenshot shows a web browser window with the URL `overconstrained.io` in the address bar. The page has a dark header with the word **OVERCONSTRAINED** on the left, and **ABOUT**, **PROJECTS**, and **SLACK** links on the right. Below the header, the word **CASSOWARY** is centered above a horizontal line with a star in the middle. The text below describes Cassowary as an incremental constraint solving toolkit. A blue callout box at the bottom right contains the text: "This is the engine behind auto layout." The word "Cassowary" in the main text is highlighted with a blue rectangle.

CASSOWARY

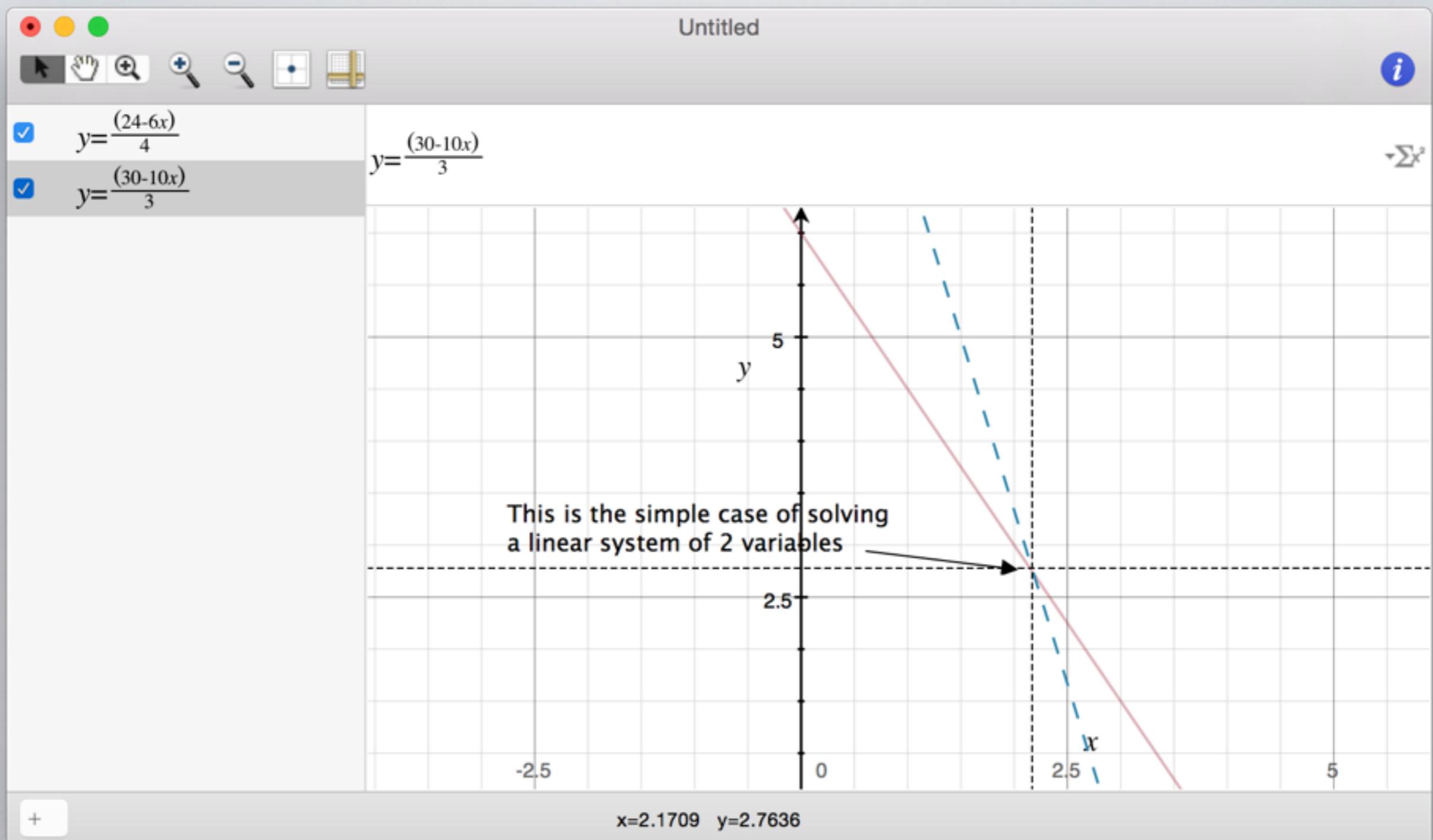
— ★ —

Cassowary is an incremental constraint solving toolkit that efficiently solves systems of linear equalities and inequalities. Constraints may be either requirements or preferences. Client code specifies the constraints to be maintained, and the solver updates the constrained variables to have values that satisfy the constraints.

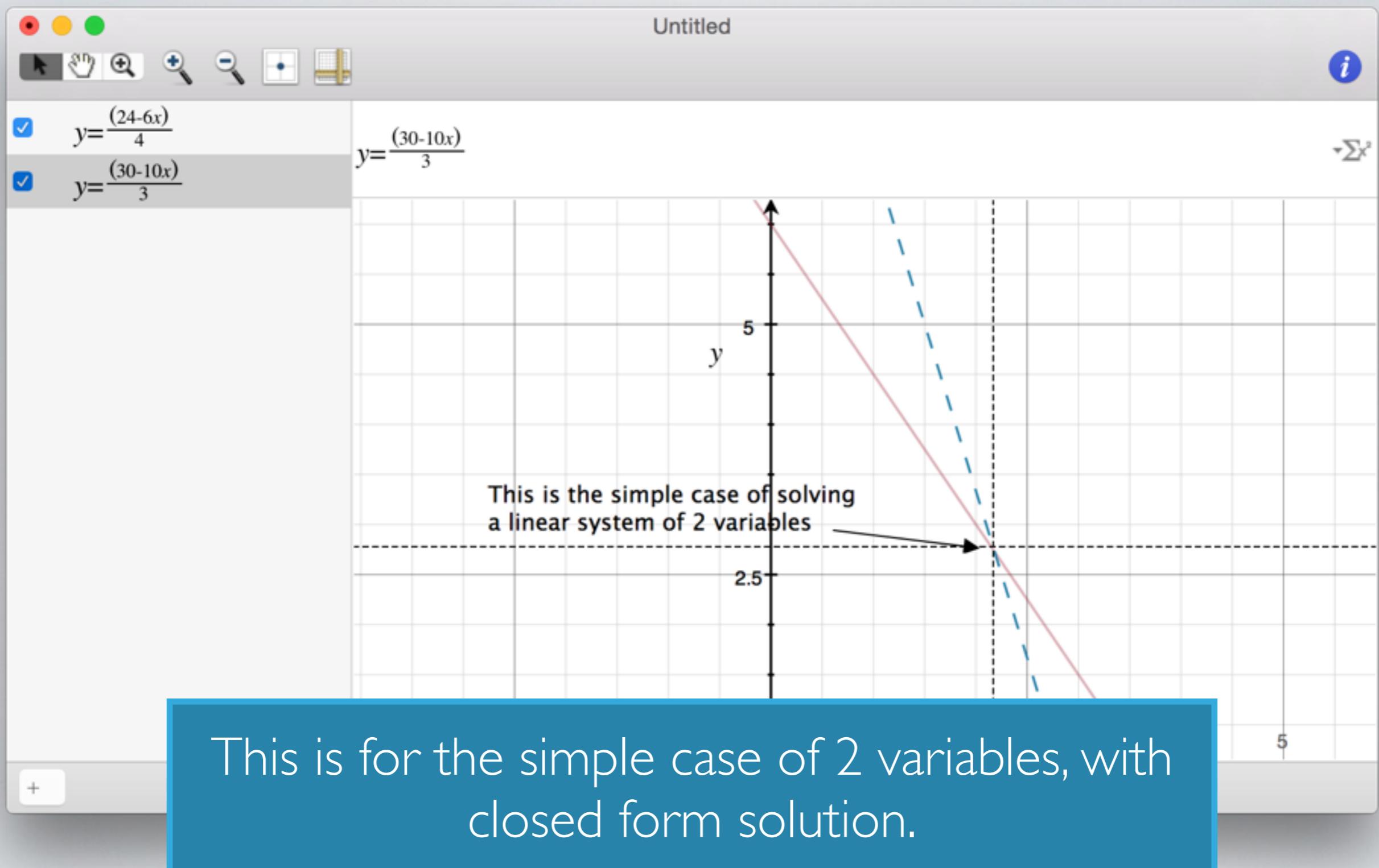
Cassowary was developed by Greg Badros and Alan Borning, and was optimized for user interface applications. Badros used Cassowary amongst others for implementing Constraint Casc

This is the engine behind auto layout.

HOW DOES IT WORK?



HOW DOES IT WORK?

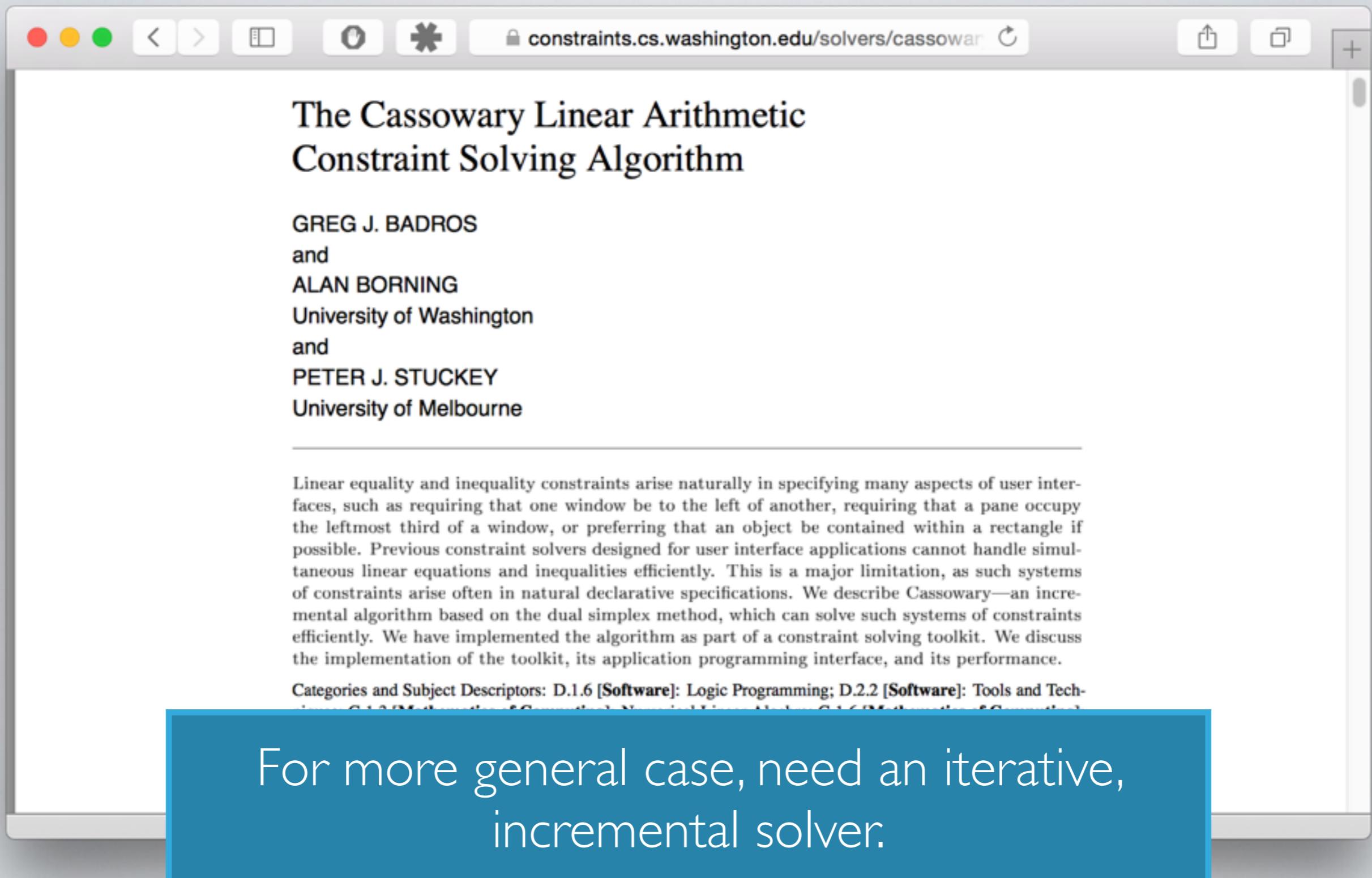


HOW DOES IT WORK?

The screenshot shows a web browser window with the following details:

- Address Bar:** constraints.cs.washington.edu/solvers/cassowar
- Content Area:**
 - Title:** The Cassowary Linear Arithmetic Constraint Solving Algorithm
 - Authors:** GREG J. BADROS
and
ALAN BORNING
University of Washington
and
PETER J. STUCKEY
University of Melbourne
 - Text:** Linear equality and inequality constraints arise naturally in specifying many aspects of user interfaces, such as requiring that one window be to the left of another, requiring that a pane occupy the leftmost third of a window, or preferring that an object be contained within a rectangle if possible. Previous constraint solvers designed for user interface applications cannot handle simultaneous linear equations and inequalities efficiently. This is a major limitation, as such systems of constraints arise often in natural declarative specifications. We describe Cassowary—an incremental algorithm based on the dual simplex method, which can solve such systems of constraints efficiently. We have implemented the algorithm as part of a constraint solving toolkit. We discuss the implementation of the toolkit, its application programming interface, and its performance.
 - Categories and Subject Descriptors:** D.1.6 [Software]: Logic Programming; D.2.2 [Software]: Tools and Techniques; G.1.3 [Mathematics of Computing]: Numerical Linear Algebra; G.1.6 [Mathematics of Computing]: Optimization
 - General Terms:** Constraints, User Interface
 - Additional Key Words and Phrases:** Cassowary, constraint-solving toolkit

HOW DOES IT WORK?



The screenshot shows a web browser window with the following details:

- Address Bar:** constraints.cs.washington.edu/solvers/cassowar
- Content Area:**
 - Title:** The Cassowary Linear Arithmetic Constraint Solving Algorithm
 - Authors:** GREG J. BADROS
and
ALAN BORNING
University of Washington
and
PETER J. STUCKEY
University of Melbourne
 - Abstract:** A detailed description of the Cassowary algorithm, mentioning its incremental nature based on the dual simplex method for solving systems of linear equality and inequality constraints.
 - Categories and Subject Descriptors:** D.1.6 [Software]: Logic Programming; D.2.2 [Software]: Tools and Techniques; G.1.3 [Mathematics of Computing]: Numerical Analysis; G.1.6 [Mathematics of Computing]: Game Theory, Programming Languages, and Computational Complexity
 - Text Box:** A blue box at the bottom contains the text: "For more general case, need an iterative, incremental solver."

HOW DOES IT WORK?

```
● ● ● 2. vazexqi@daigo: ~ (zsh)
vazexqi@daigo:~|⇒ brew install python 22:00:05
vazexqi@daigo:~|⇒ pip install cassowary 22:00:19
Collecting cassowary
  Using cached cassowary-0.5.1-py2.py3-none-any.whl
Installing collected packages: cassowary
Successfully installed cassowary-0.5.1
vazexqi@daigo:~|⇒ 22:00:21
```

HOW DOES IT WORK?

```
vazexqi@daigo:~|~ brew install python                                22:00:05
vazexqi@daigo:~|~ pip install cassowary                            22:00:19
Collecting cassowary
  Using cached cassowary-0.5.1-py2.py3-none-any.whl
Installing collected packages: cassowary
Successfully installed cassowary-0.5.1
vazexqi@daigo:~|~ | 22:00:21
```

It's unfortunate but I can't access the iOS engine directly so use the python port of cassowary.

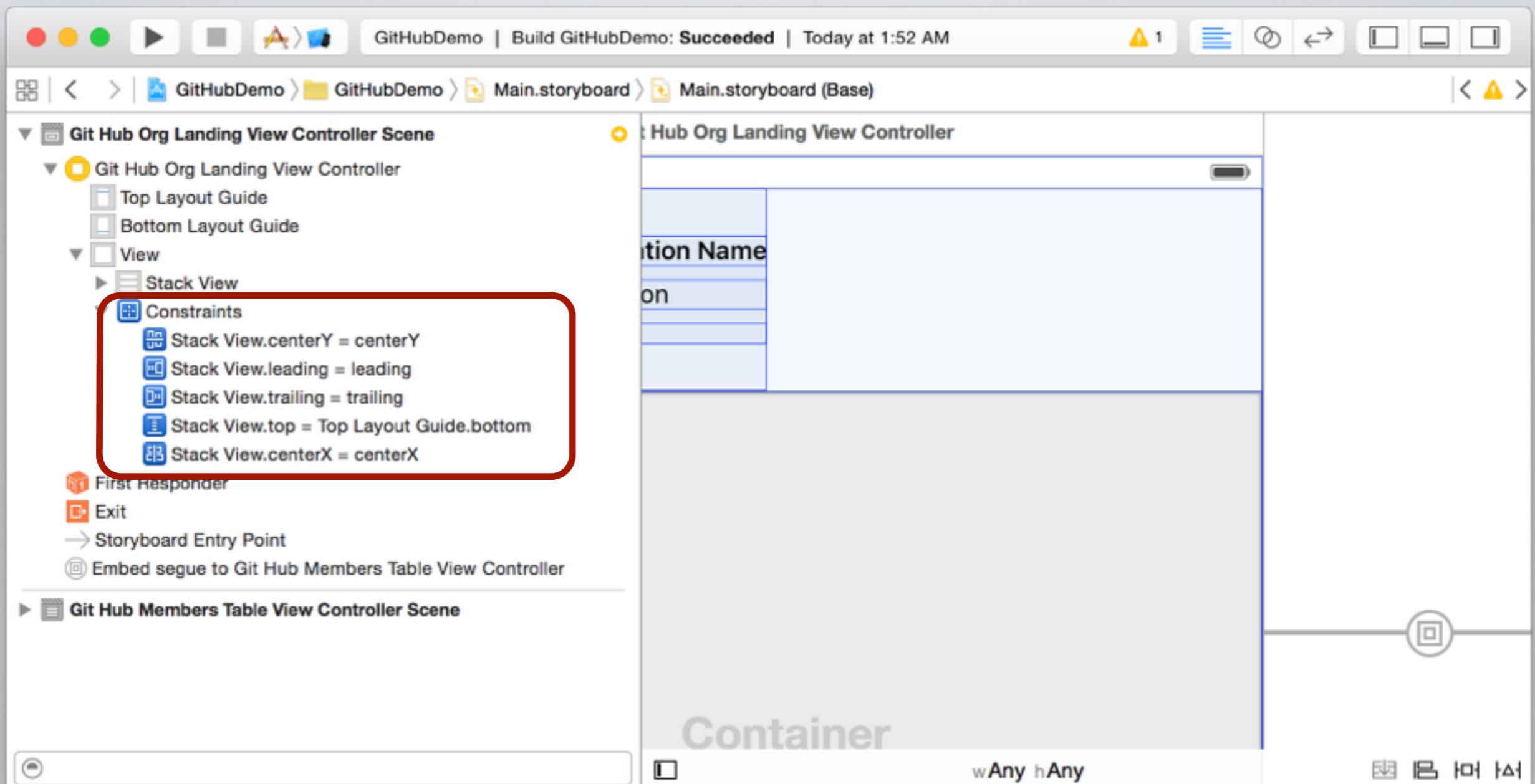
HOW DOES IT WORK?

```
vazexqi@daigo:~|~ python 2. python (Python) 22:02:43
Python 2.7.6 (default, Sep  9 2014, 15:04:36)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.39)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from cassowary import SimplexSolver, Variable
>>> solver = SimplexSolver()
>>> left = Variable('left')
>>> middle = Variable('middle')
>>> right = Variable('right')
>>> solver.add_constraint(middle == (left + right) / 2)
Required:{1.0}{0.5*left[0.0] + -1.0*middle[0.0] + 0.5*right[-0.0]}
>>> solver.add_constraint(right == left + 10)
Required:{1.0}{10.0 + left[-5.0] + -1.0*right[5.0]}
>>> solver.add_constraint(right <= 100)
Required:{1.0}{100.0 + -1.0*right[100.0]}
>>> solver.add_constraint(left >= 0)
Required:{1.0}{left[90.0]}
>>> left.value
90.0
>>> middle.value
95.0
>>> right.value
100.0
>>> █
```

SCENARIOS

- Under-constrained - Ambiguous, more than one solution. Could pick either solution at run time 😞
- Perfectly-constrained 😎
- Over-constrained - Conflicting constraints, have to break one of them. Again unpredictable behavior at run time 😞

CONSTRAINTS IN IOS



NSLAYOUTCONSTRAINT

self.contentView.Left = <multiplier> * self.view.Left + <constant>



```
let leftConstraint = NSLayoutConstraint(item: self.contentView,  
                                         attribute: .Left,  
                                         relatedBy: .Equal,  
                                         toItem: self.view,  
                                         attribute: .Left,  
                                         multiplier: 1.0,  
                                         constant: 0.0)  
self.view.addConstraint(leftConstraint)
```



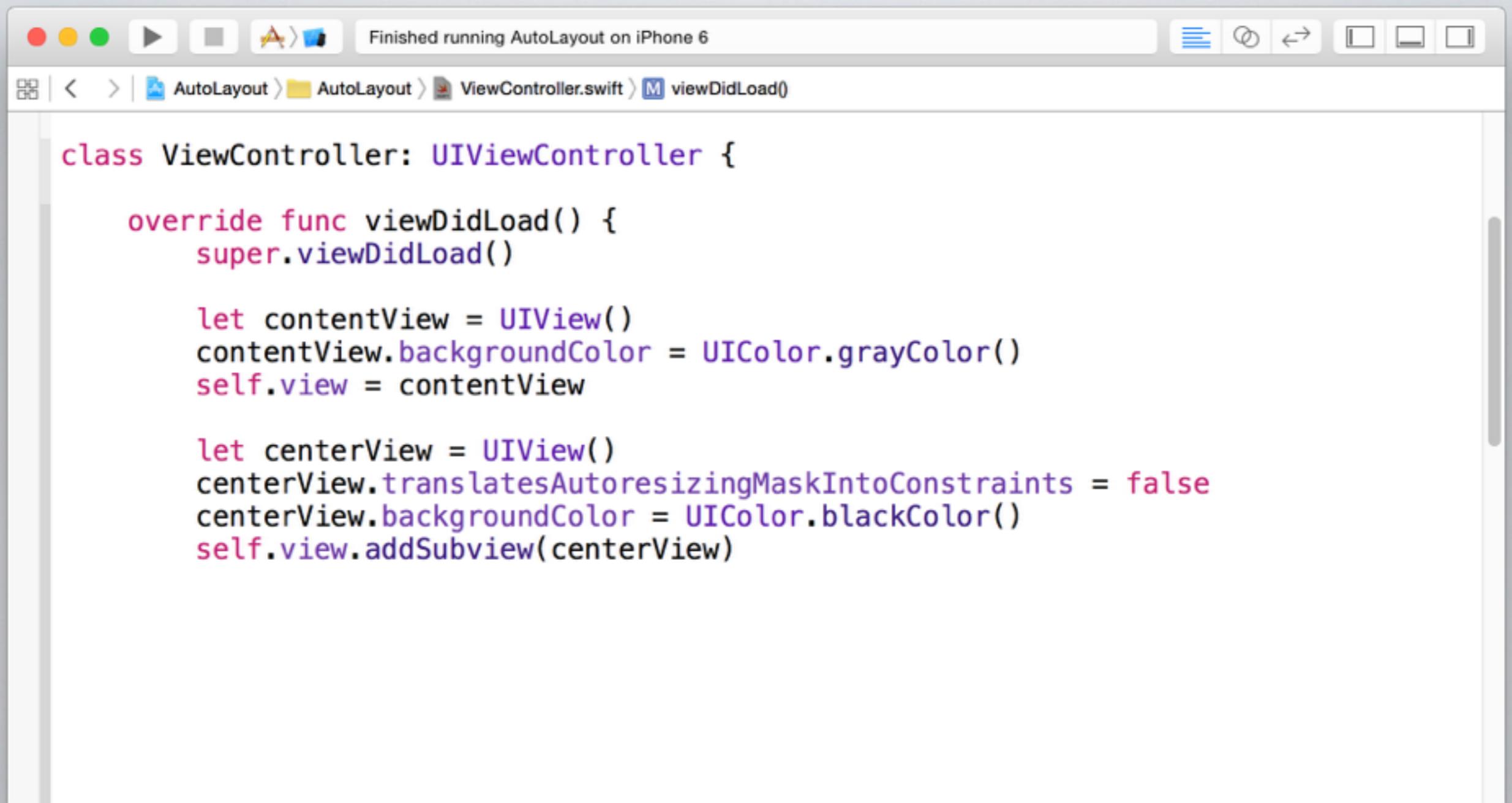
NSLAYOUTCONSTRAINT

self.contentView.Left = <multiplier> * self.view.Left + <constant>



```
let leftConstraint = self.contentView.leftAnchor.constraintEqualToAnchor(  
    self.view.leftAnchor,  
    multiplier:1.0)  
  
self.view.addConstraint(leftConstraint)
```

PROGRAMMATICALLY



A screenshot of the Xcode IDE showing a Swift file named ViewController.swift. The code implements a view controller that adds a black center view to a gray content view.

```
class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        let contentView = UIView()
        contentView.backgroundColor = UIColor.grayColor()
        self.view = contentView

        let centerView = UIView()
        centerView.translatesAutoresizingMaskIntoConstraints = false
        centerView.backgroundColor = UIColor.blackColor()
        self.view.addSubview(centerView)
    }
}
```

PROGRAMMATICALLY

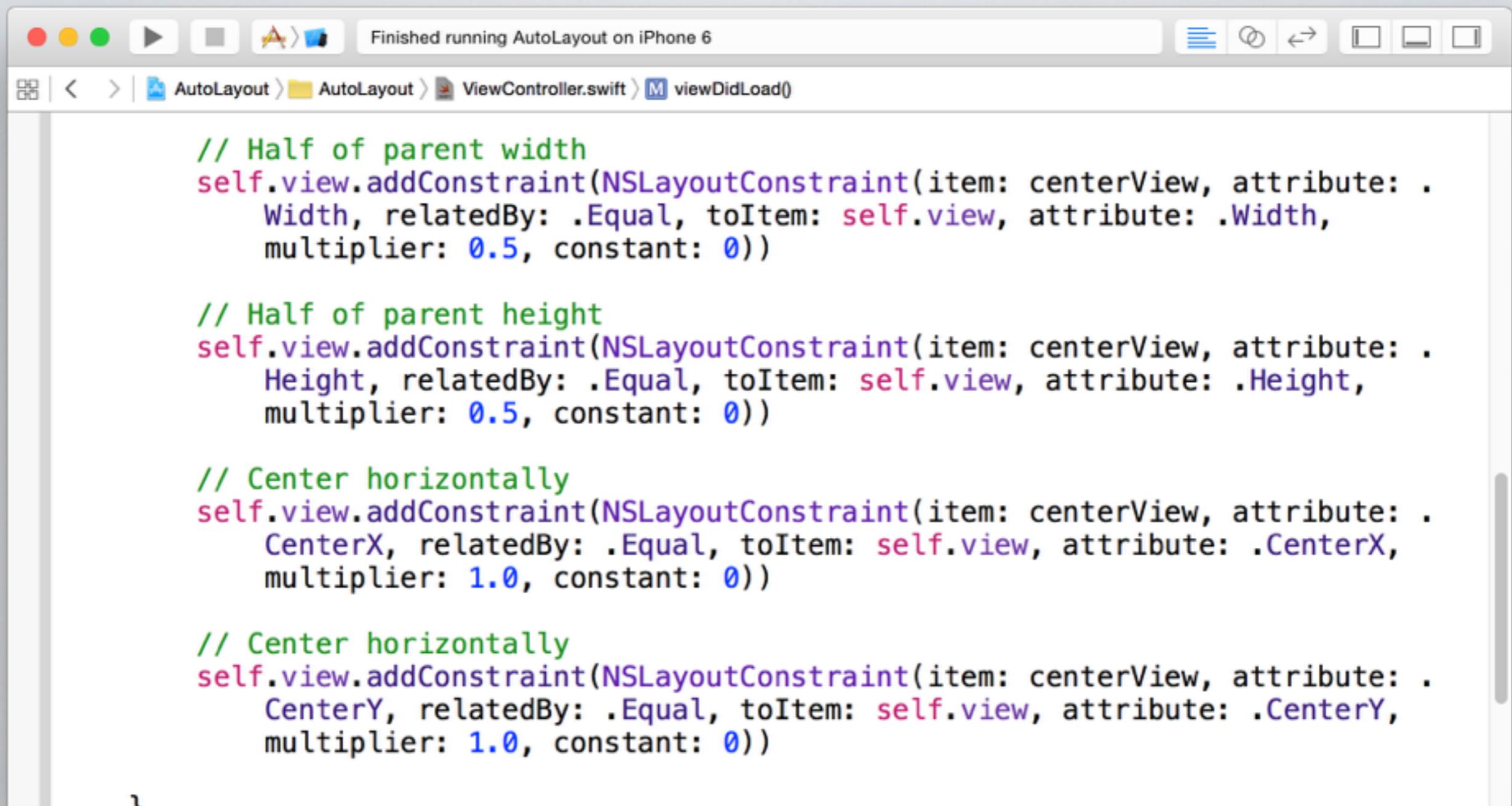
The screenshot shows a Xcode interface with a Swift file named ViewController.swift. The code defines a contentView and a centerView, and sets the translatesAutoresizingMaskIntoConstraints property of centerView to false. A callout bubble is open over the line of code `centerView.translatesAutoresizingMaskIntoConstraints = false`, providing documentation for the `translatesAutoresizingMaskIntoConstraints` property.

```
let contentView = UIView()
contentView.backgroundColor = UIColor.grayColor()
self.view = contentView

let centerView = UIView()
centerView.translatesAutoresizingMaskIntoConstraints = false
centerView.b
    Declaration var translatesAutoresizingMaskIntoConstraints: Bool { get set }
    Description A Boolean value that determines whether the view's autoresizing mask is translated into Auto Layout constraints.
        If this property's value is true, the system creates a set of constraints that duplicate the behavior specified by the view's autoresizing mask. This also lets you modify the view's size and location using the view's frame, bounds, or center properties, allowing you to create a static, frame-based layout within Auto Layout.
        Note that the autoresizing mask constraints fully specify the view's size and position; therefore, you cannot add additional constraints to modify this size or position without introducing conflicts. If you want to use Auto Layout to dynamically calculate the size and position of your view, you must set this property to false, and then provide a nonambiguous, nonconflicting set of constraints for the view.
        By default, the property is set to true for any view you programmatically create. If you add views in Interface Builder, the system automatically sets this property to false.
    Availability iOS (6.0 and later)
    Declared In UIKit
    Reference UIView Class Reference

```

PROGRAMMATICALLY



A screenshot of an Xcode interface. The title bar says "Finished running AutoLayout on iPhone 6". The navigation bar shows the file path: "AutoLayout > AutoLayout > ViewController.swift > viewDidLoad()". The main area contains the following Swift code:

```
// Half of parent width
self.view.addConstraint(NSLayoutConstraint(item: centerView, attribute: .Width, relatedBy: .Equal, toItem: self.view, attribute: .Width, multiplier: 0.5, constant: 0))

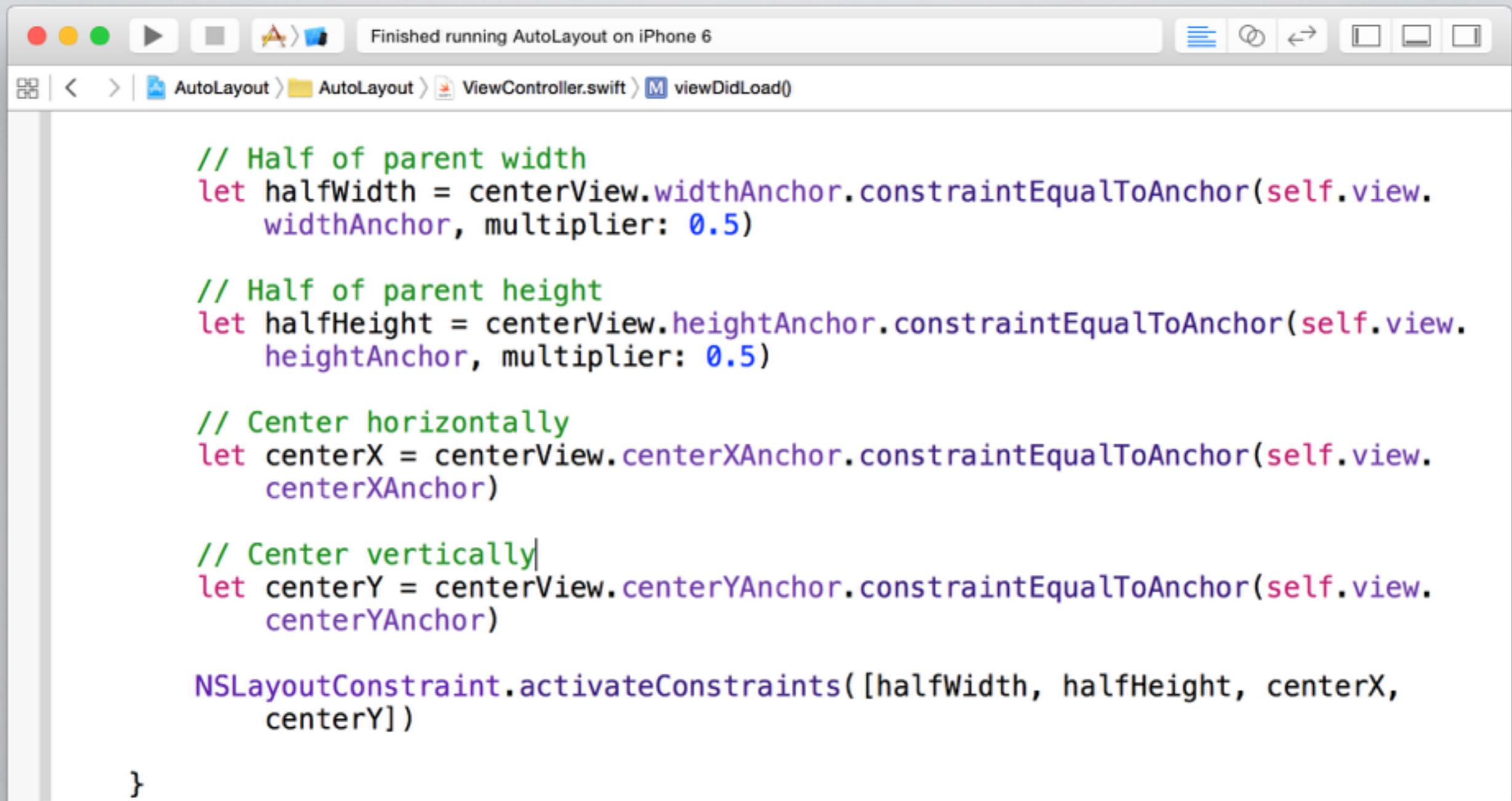
// Half of parent height
self.view.addConstraint(NSLayoutConstraint(item: centerView, attribute: .Height, relatedBy: .Equal, toItem: self.view, attribute: .Height, multiplier: 0.5, constant: 0))

// Center horizontally
self.view.addConstraint(NSLayoutConstraint(item: centerView, attribute: .CenterX, relatedBy: .Equal, toItem: self.view, attribute: .CenterX, multiplier: 1.0, constant: 0))

// Center vertically
self.view.addConstraint(NSLayoutConstraint(item: centerView, attribute: .CenterY, relatedBy: .Equal, toItem: self.view, attribute: .CenterY, multiplier: 1.0, constant: 0))
```

The code defines four NSLayoutConstraint objects. The first two set the width and height of a view named "centerView" to half of its parent's width and height respectively, using a multiplier of 0.5. The last two constraints center the "centerView" both horizontally and vertically within its parent view.

PROGRAMMATICALLY



The screenshot shows a Mac OS X desktop with an Xcode window open. The title bar says "Finished running AutoLayout on iPhone 6". The file path in the toolbar is "AutoLayout > AutoLayout > ViewController.swift > viewDidLoad()". The main editor area contains the following Swift code:

```
// Half of parent width
let halfWidth = centerView.widthAnchor.constraintEqualToAnchor(self.view.
    widthAnchor, multiplier: 0.5)

// Half of parent height
let halfHeight = centerView.heightAnchor.constraintEqualToAnchor(self.view.
    heightAnchor, multiplier: 0.5)

// Center horizontally
let centerX = centerView.centerXAnchor.constraintEqualToAnchor(self.view.
    centerXAnchor)

// Center vertically
let centerY = centerView.centerYAnchor.constraintEqualToAnchor(self.view.
    centerYAnchor)

NSLayoutConstraint.activateConstraints([halfWidth, halfHeight, centerX,
    centerY])

}
```

PROGRAMMATICALLY

```
Finished running AutoLayout on iPhone 6
AutoLayout > AutoLayout > ViewController.swift > viewDidLoad()

// Half of parent width
let halfWidth = centerView.widthAnchor.constraintEqualToAnchor(self.view.
    widthAnchor, multiplier: 0.5)

// Half of parent height
let halfHeight = centerView.heightAnchor.constraintEqualToAnchor(self.view.
    heightAnchor, multiplier: 0.5)

// Center horizontally
let centerX = centerView.centerXAnchor.constraintEqualToAnchor(self.view.
    centerXAnchor)

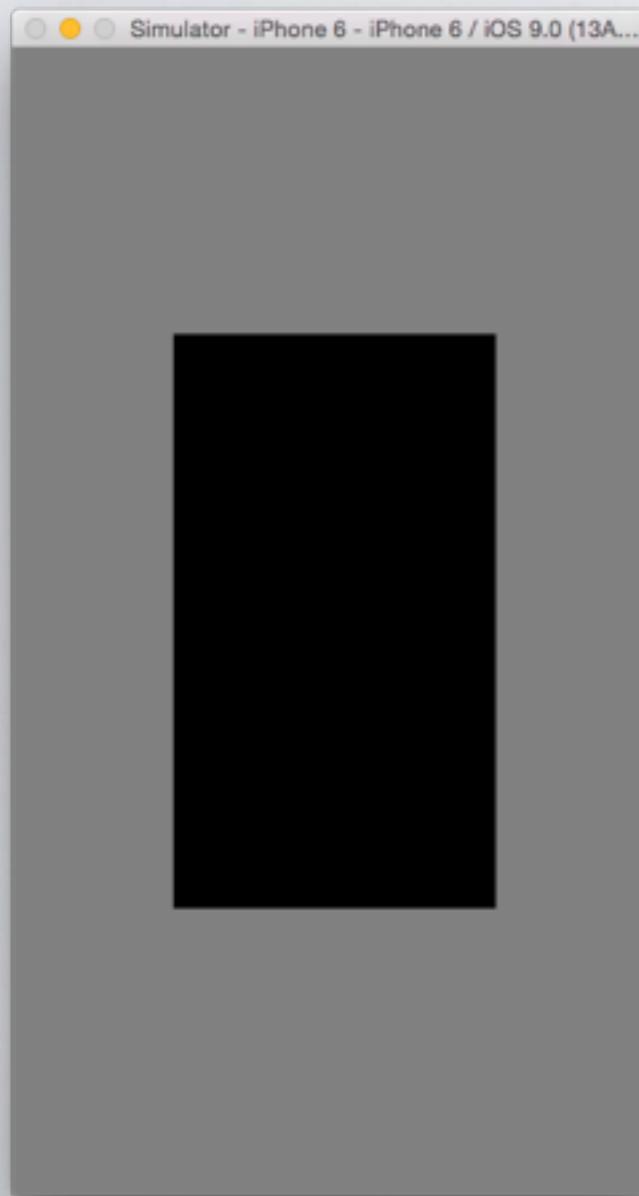
// Center vertically
let centerY = centerView.centerYAnchor.constraintEqualToAnchor(self.view.
    centerYAnchor)

NSLayoutConstraint.activateConstraints([halfWidth, halfHeight, centerX,
    centerY])

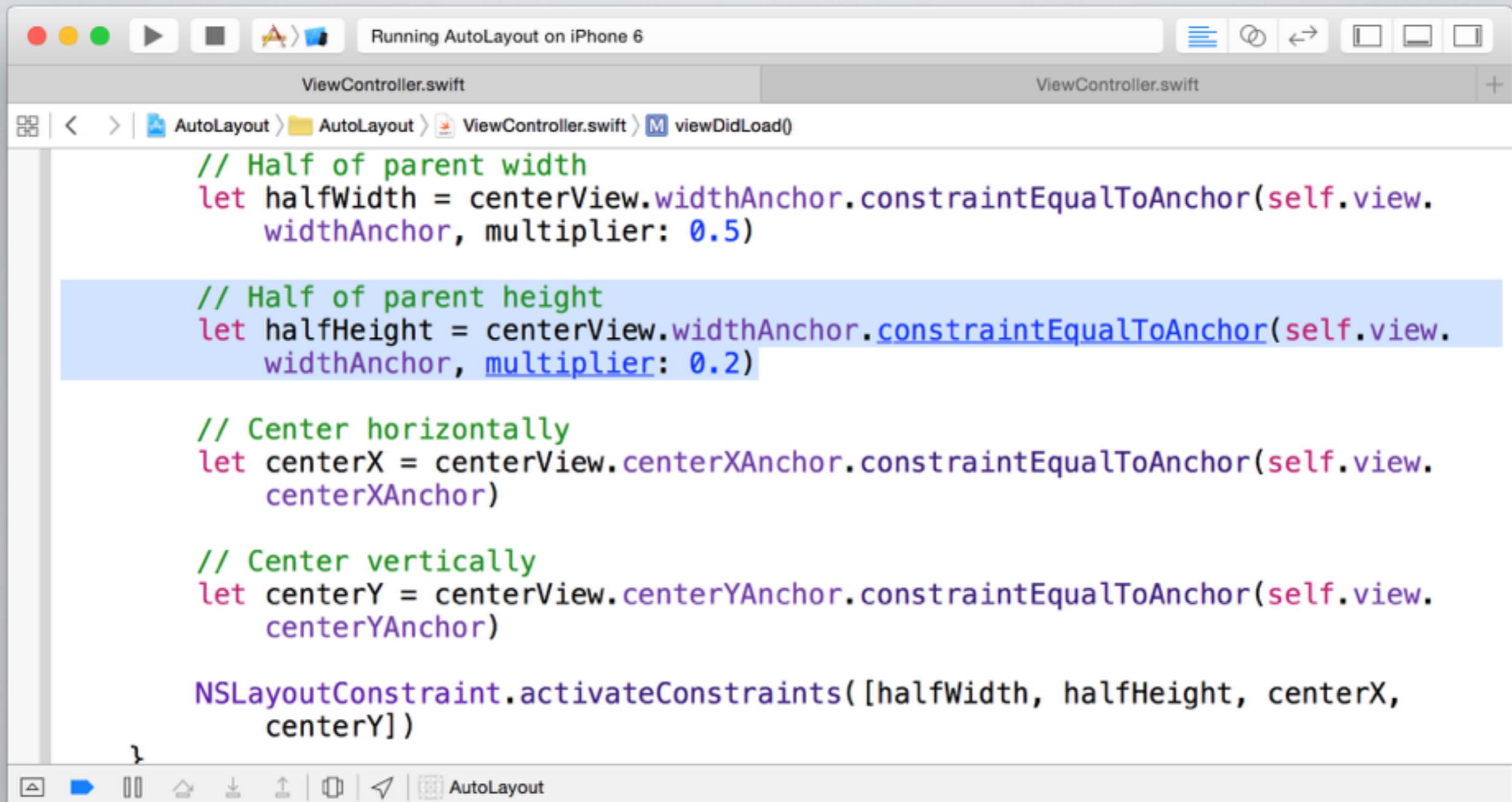
}
```

Always remember to
activate the constraints

PROGRAMMATICALLY



PROGRAMMATICALLY



The screenshot shows the Xcode interface with a project titled "AutoLayout" running on an iPhone 6 simulator. The code in ViewController.swift demonstrates how to set up AutoLayout constraints programmatically within the viewDidLoad() method.

```
// Half of parent width
let halfWidth = centerView.widthAnchor.constraintEqualToAnchor(self.view.widthAnchor, multiplier: 0.5)

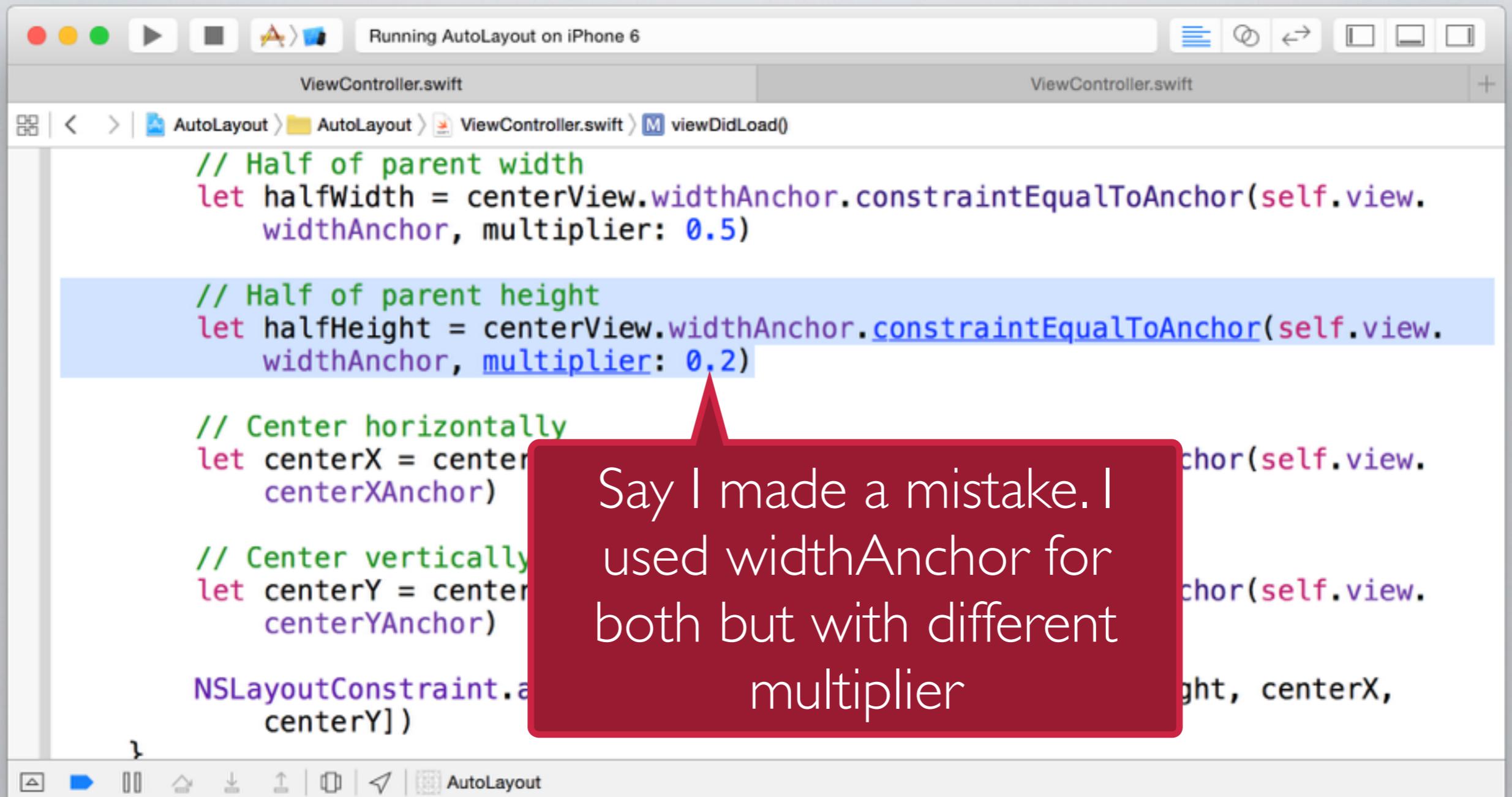
// Half of parent height
let halfHeight = centerView.widthAnchor.constraintEqualToAnchor(self.view.widthAnchor, multiplier: 0.2)

// Center horizontally
let centerX = centerView.centerXAnchor.constraintEqualToAnchor(self.view.centerXAnchor)

// Center vertically
let centerY = centerView.centerYAnchor.constraintEqualToAnchor(self.view.centerYAnchor)

NSLayoutConstraint.activateConstraints([halfWidth, halfHeight, centerX, centerY])
```

PROGRAMMATICALLY



The screenshot shows the Xcode interface with a storyboard file open. The top bar displays "Running AutoLayout on iPhone 6". The storyboard preview shows a single view controller with a central yellow square. The code editor shows the ViewController.swift file with the following content:

```
// Half of parent width
let halfWidth = centerView.widthAnchor.constraintEqualToAnchor(self.view.widthAnchor, multiplier: 0.5)

// Half of parent height
let halfHeight = centerView.widthAnchor.constraintEqualToAnchor(self.view.widthAnchor, multiplier: 0.2)

// Center horizontally
let centerX = centerView.centerXAnchor.constraintEqualToAnchor(self.view.centerXAnchor, multiplier: 1.0)

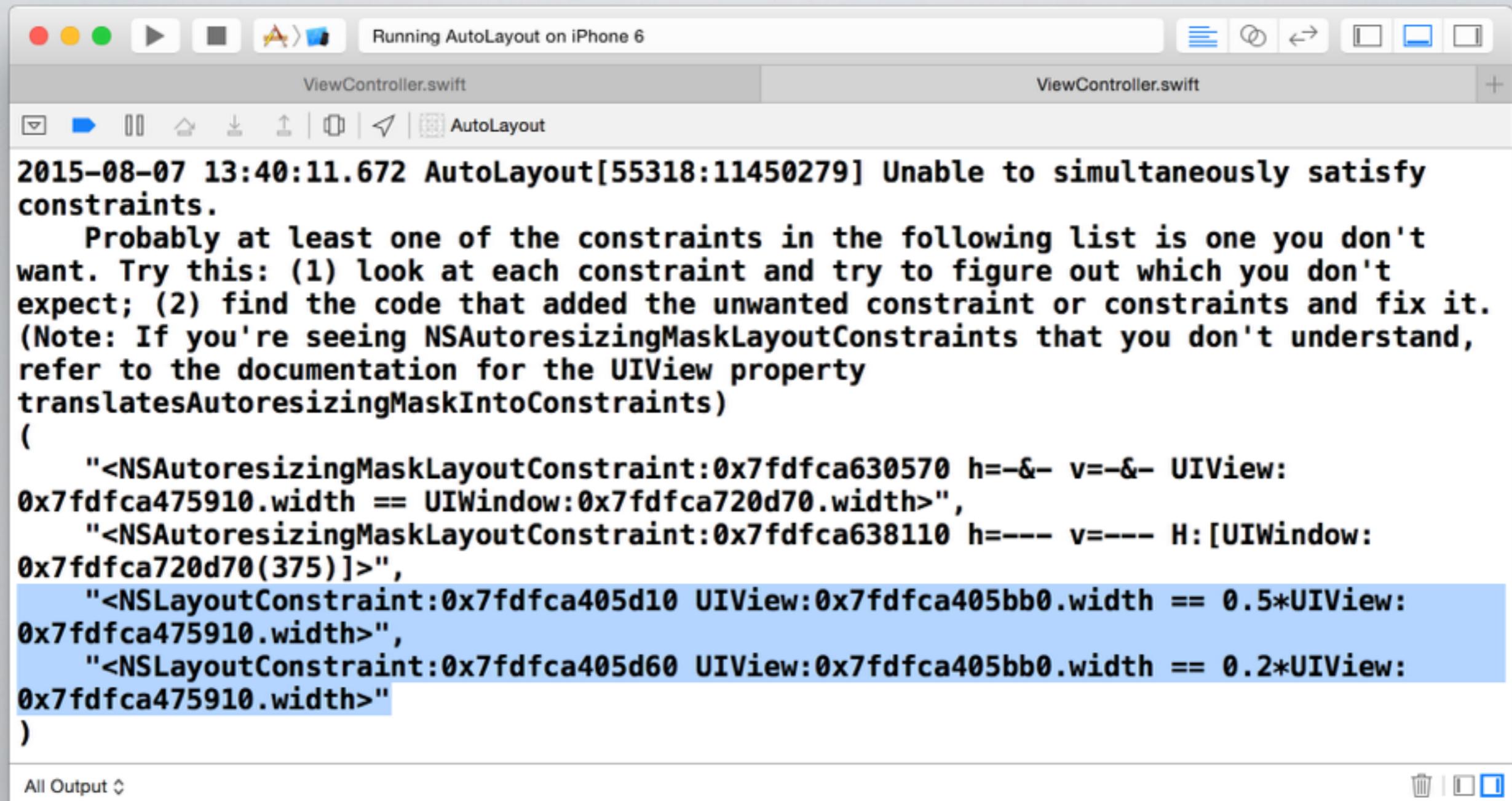
// Center vertically
let centerY = centerView.centerYAnchor.constraintEqualToAnchor(self.view.centerYAnchor, multiplier: 1.0)

NSLayoutConstraint.activateConstraints([centerX, centerY])
```

A red callout bubble with a black outline and a black arrow points from the text "Say I made a mistake. I used widthAnchor for both but with different multiplier" to the line of code "let halfHeight = centerView.widthAnchor.constraintEqualToAnchor(self.view.widthAnchor, multiplier: 0.2)". The background of the callout is red.

Say I made a mistake. I used widthAnchor for both but with different multiplier

PROGRAMMATICALLY



The screenshot shows a Xcode interface with a toolbar at the top and two tabs labeled "ViewController.swift". Below the tabs is a toolbar with various icons. The main area displays a log message:

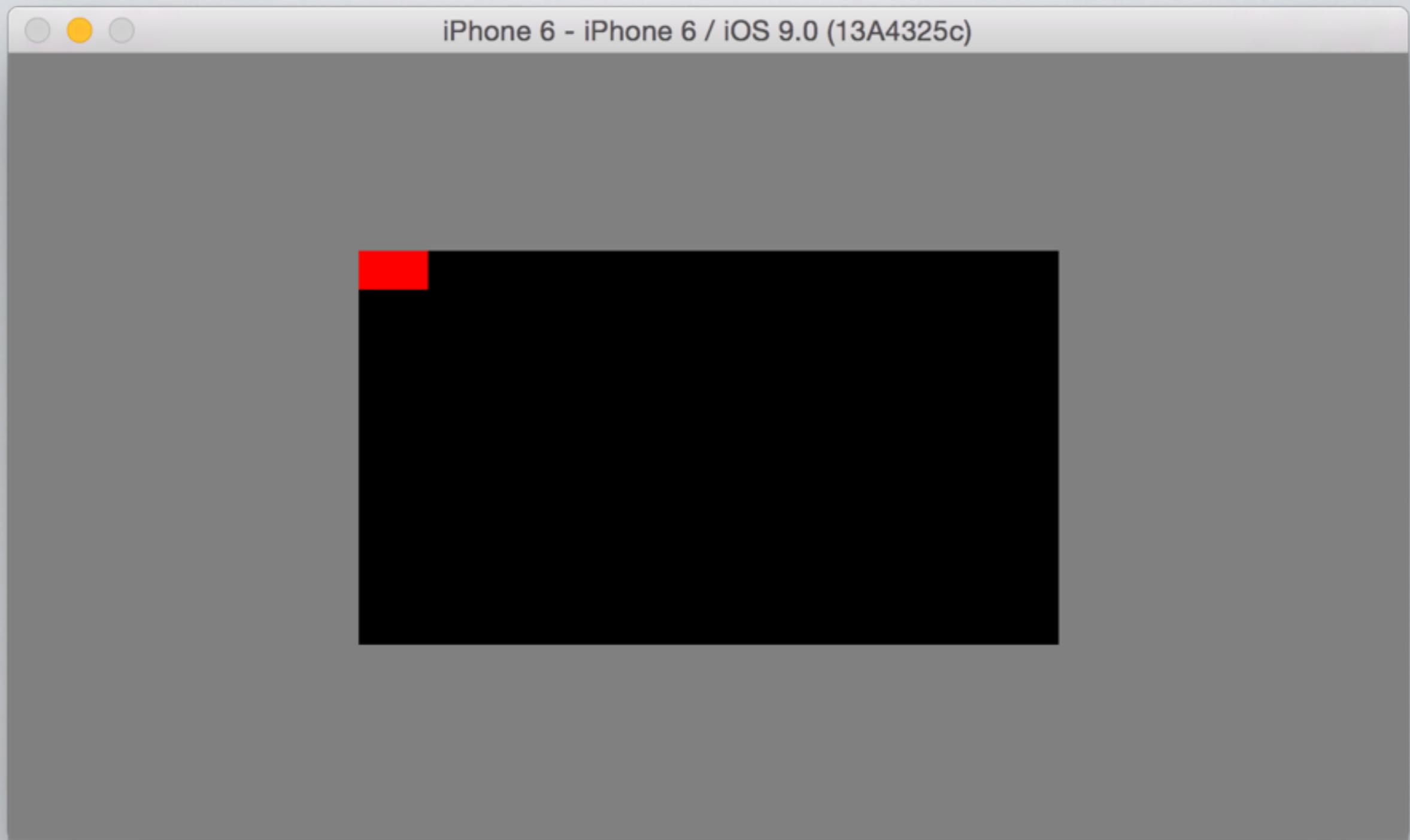
```
2015-08-07 13:40:11.672 AutoLayout[55318:11450279] Unable to simultaneously satisfy constraints.
```

Probably at least one of the constraints in the following list is one you don't want. Try this: (1) look at each constraint and try to figure out which you don't expect; (2) find the code that added the unwanted constraint or constraints and fix it. (Note: If you're seeing `NSAutoresizingMaskLayoutConstraint`s that you don't understand, refer to the documentation for the `UIView` property `translatesAutoresizingMaskIntoConstraints`)

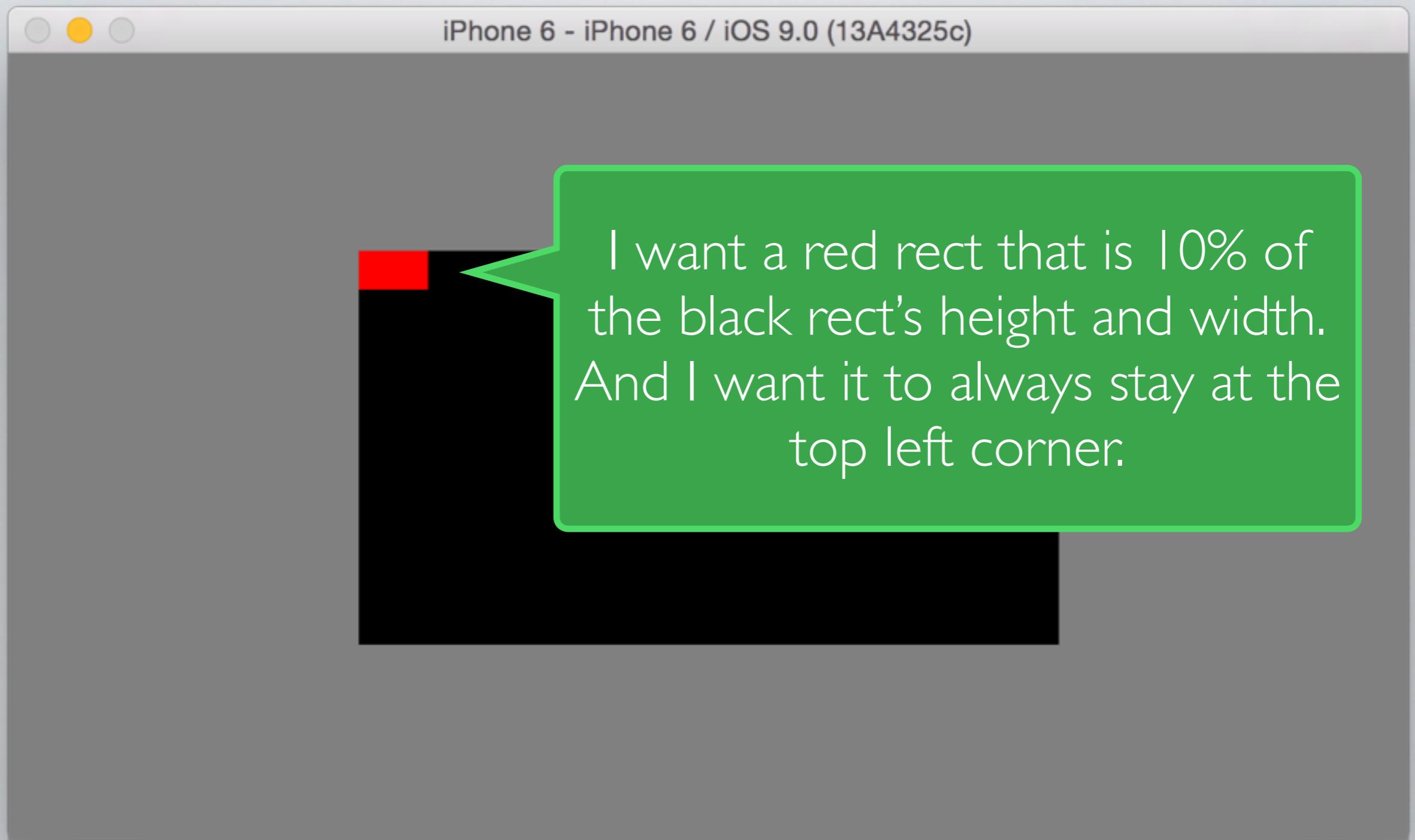
```
(  
    "<NSAutoresizingMaskLayoutConstraint:0x7fdfca630570 h=-&- v=-&- UIView:  
0x7fdfca475910.width == UIWindow:0x7fdfca720d70.width>",  
    "<NSAutoresizingMaskLayoutConstraint:0x7fdfca638110 h=---- v=---- H:[UIWindow:  
0x7fdfca720d70(375)]>",  
    "<NSLayoutConstraint:0x7fdfca405d10 UIView:0x7fdfca405bb0.width == 0.5*UIView:  
0x7fdfca475910.width>",  
    "<NSLayoutConstraint:0x7fdfca405d60 UIView:0x7fdfca405bb0.width == 0.2*UIView:  
0x7fdfca475910.width>"  
)
```

At the bottom left, there is a "All Output" dropdown, and at the bottom right, there are three small icons: a trash can, a square, and a blue square.

EXERCISE



EXERCISE



EXERCISE

The screenshot shows a GitHub repository page for 'talentsparkio / AutoLayout'. The page includes a navigation bar with icons for file operations, a search bar, and links for Pull requests, Issues, and Gist. Below the header, there's a sidebar with repository statistics: 3 commits, 1 branch, 0 releases, and 1 contributor. The main content area displays a commit history with the latest commit by 'vazexqi' (39 seconds ago) and other recent commits by 'aitianore' (38 seconds ago and 15 hours ago). The interface has a light gray background with blue and white text.

This repository Search Pull requests Issues Gist

talentsparkio / AutoLayout Unwatch 4 Star

Description Website

Short description of this repository Website for this repository (optional) Save or Cancel

3 commits 1 branch 0 releases 1 contributor

Branch: master + AutoLayout / +

Use new NSLayoutAnchor API
vazexqi authored 39 seconds ago latest commit 37689a915a

AutoLayout.xcodeproj Programmatic example 15 hours ago

AutoLayout Use new NSLayoutAnchor API 38 seconds ago

aitianore Programmatic example 15 hours ago

EXERCISE

The screenshot shows a GitHub repository page for 'talentsparkio / AutoLayout'. The page includes a navigation bar with icons for file operations, a search bar, and links for Pull requests, Issues, and Gist. Below the header, the repository name 'talentsparkio / AutoLayout' is displayed, along with an 'Unwatch' button, a '4' badge, and a star icon. A 'Description' section contains fields for a short description and a website URL, with a 'Save' button. Summary statistics show 3 commits, 1 branch, 0 releases, and 1 contributor. The 'master' branch is selected. The repository's contents are listed, including 'AutoLayout.xcodeproj', 'Programmatic example', and 'Skeleton code to get started'. The 'Skeleton code to get started' file is highlighted with a blue box.

GitHub, Inc. github.com/talentsparkio/AutoLayout

This repository Search

Pull requests Issues Gist

talentsparkio / AutoLayout

Unwatch 4 Star

Description

Short description of this repository

Website

Website for this repository (optional)

Save or Cancel

3 commits 1 branch 0 releases 1 contributor

Branch: master + AutoLayout / +

Use new NSLayoutAnchor API

vazexqi authored 39 seconds ago latest commit 37689a915a

AutoLayout.xcodeproj Programmatic example 15 hours ago

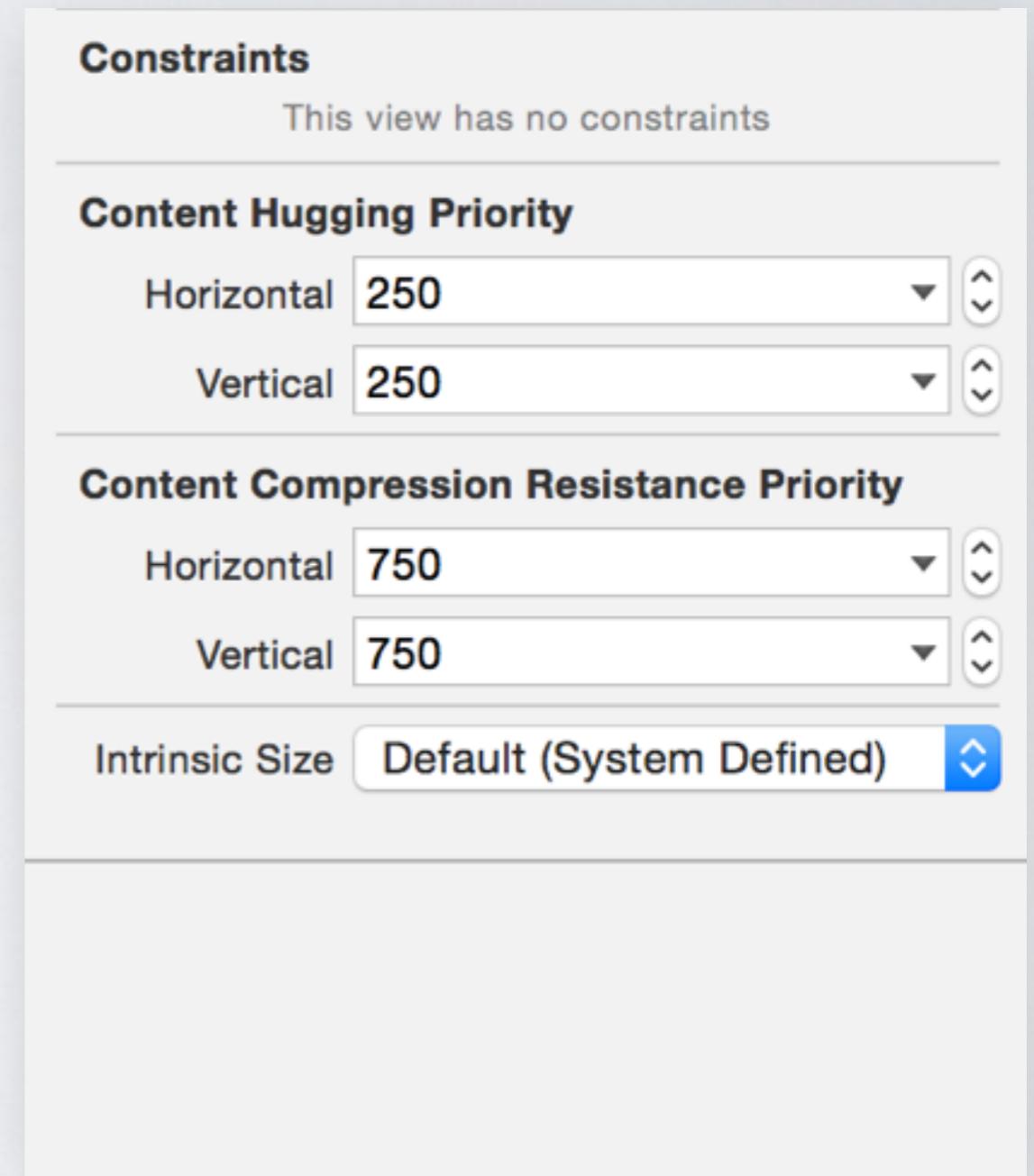
AutoLayout Skeleton code to get started

CONSTRAINTS WE'VE SEEN

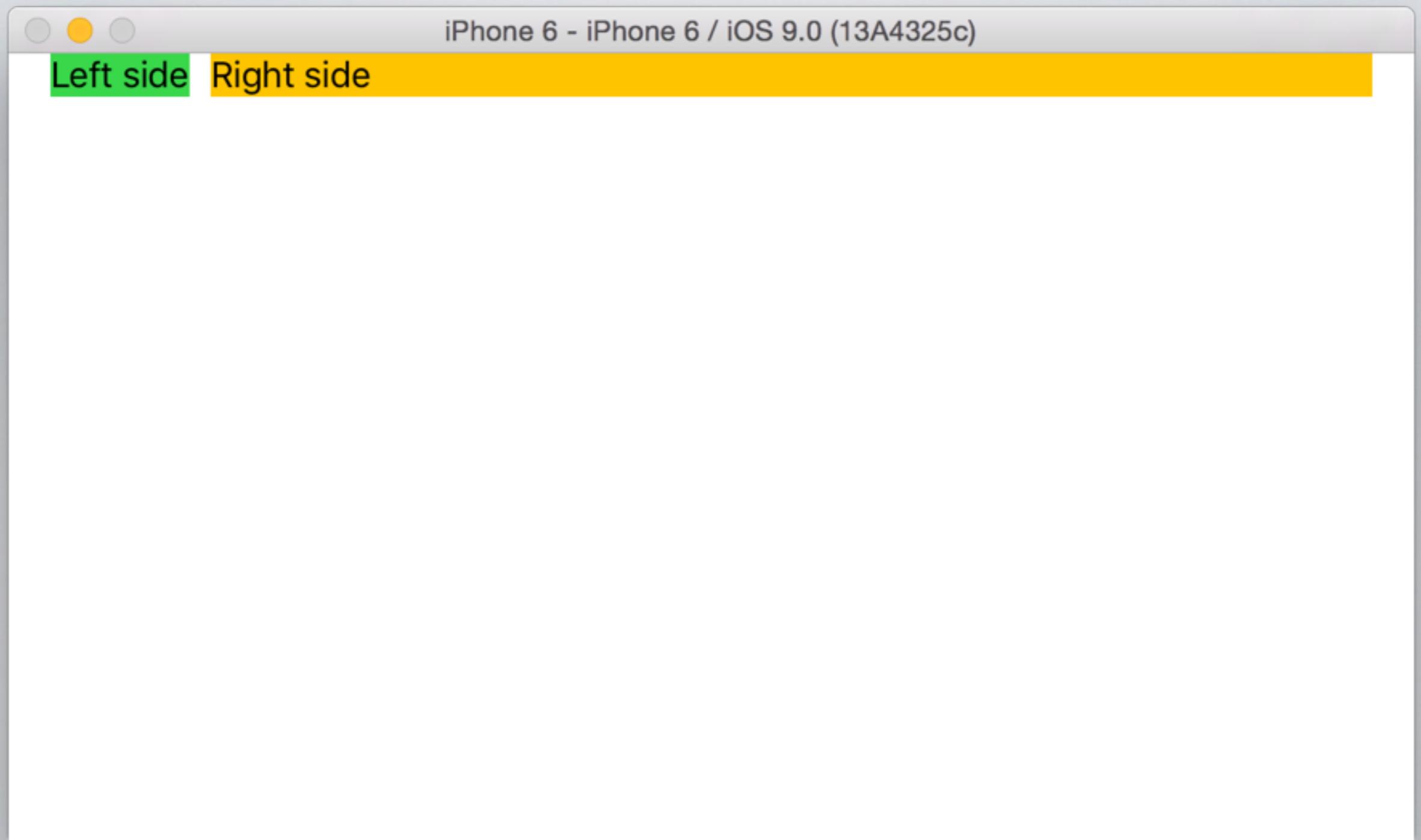
- Alignment constraints (top, left, bottom, right, spacing)
- Size constraints (width, height)

TWO OTHER CONSTRAINTS

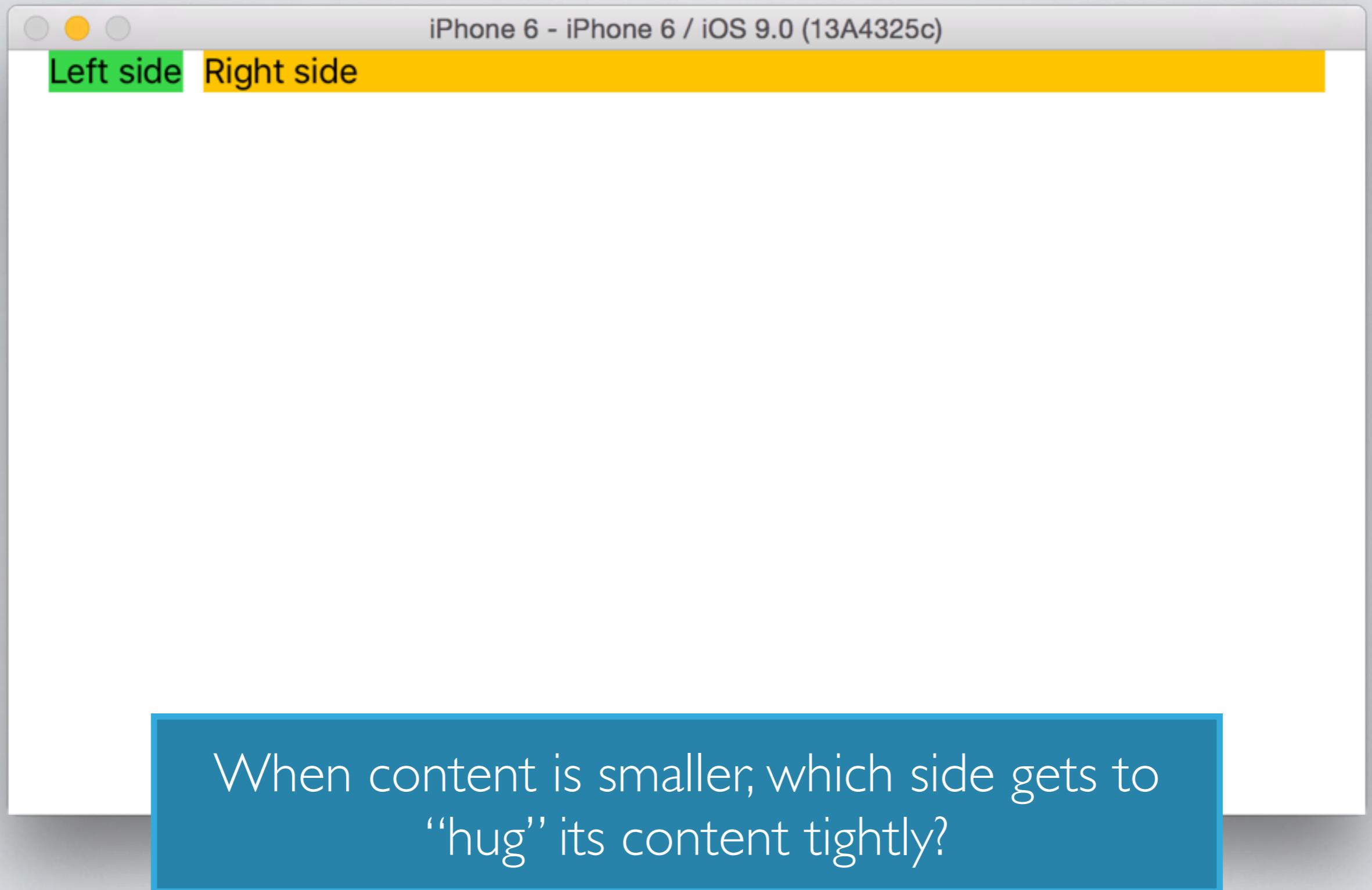
- Content Hugging Priority
- Compression Resistance Priority



CONTENT HUGGING

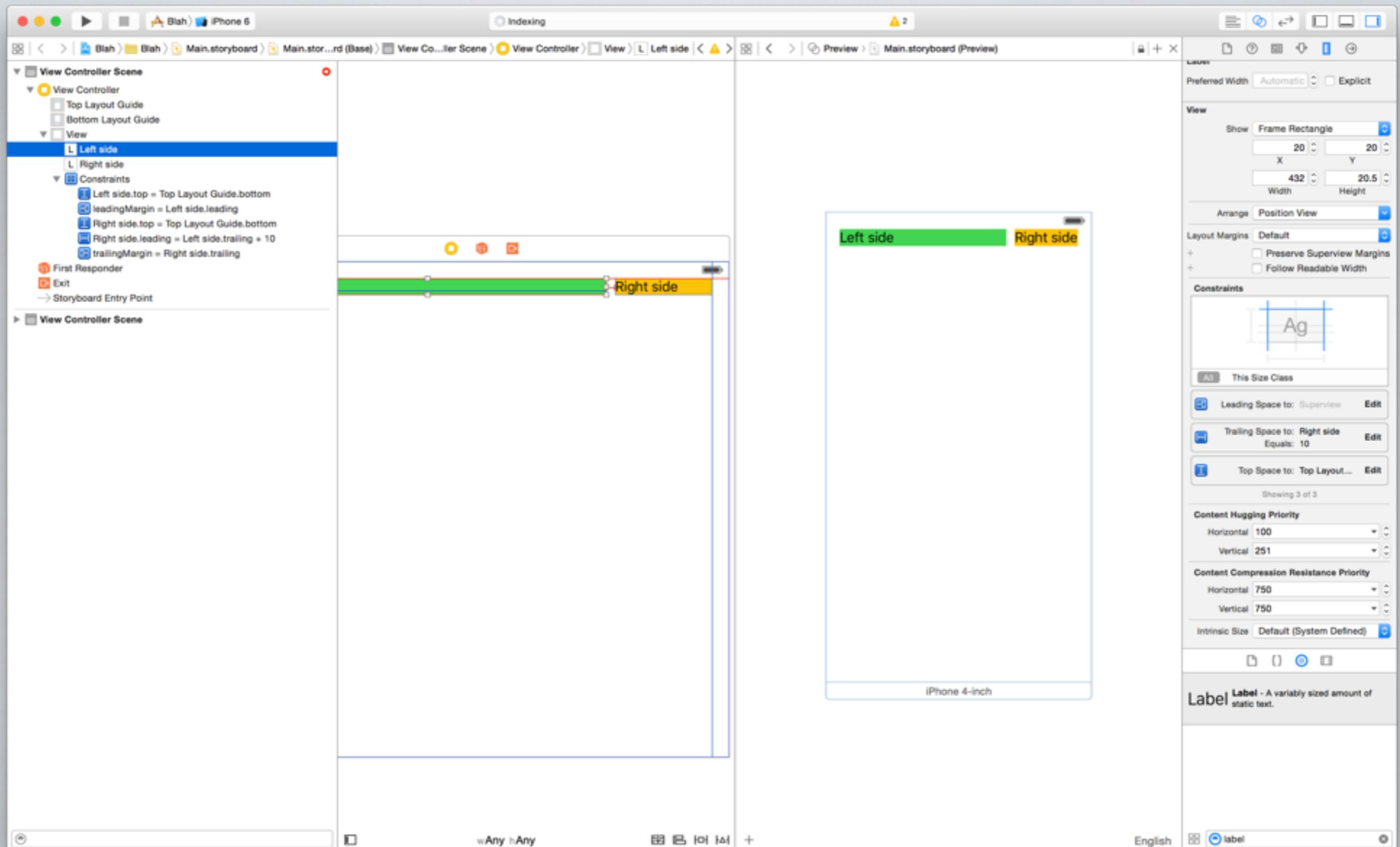


CONTENT HUGGING

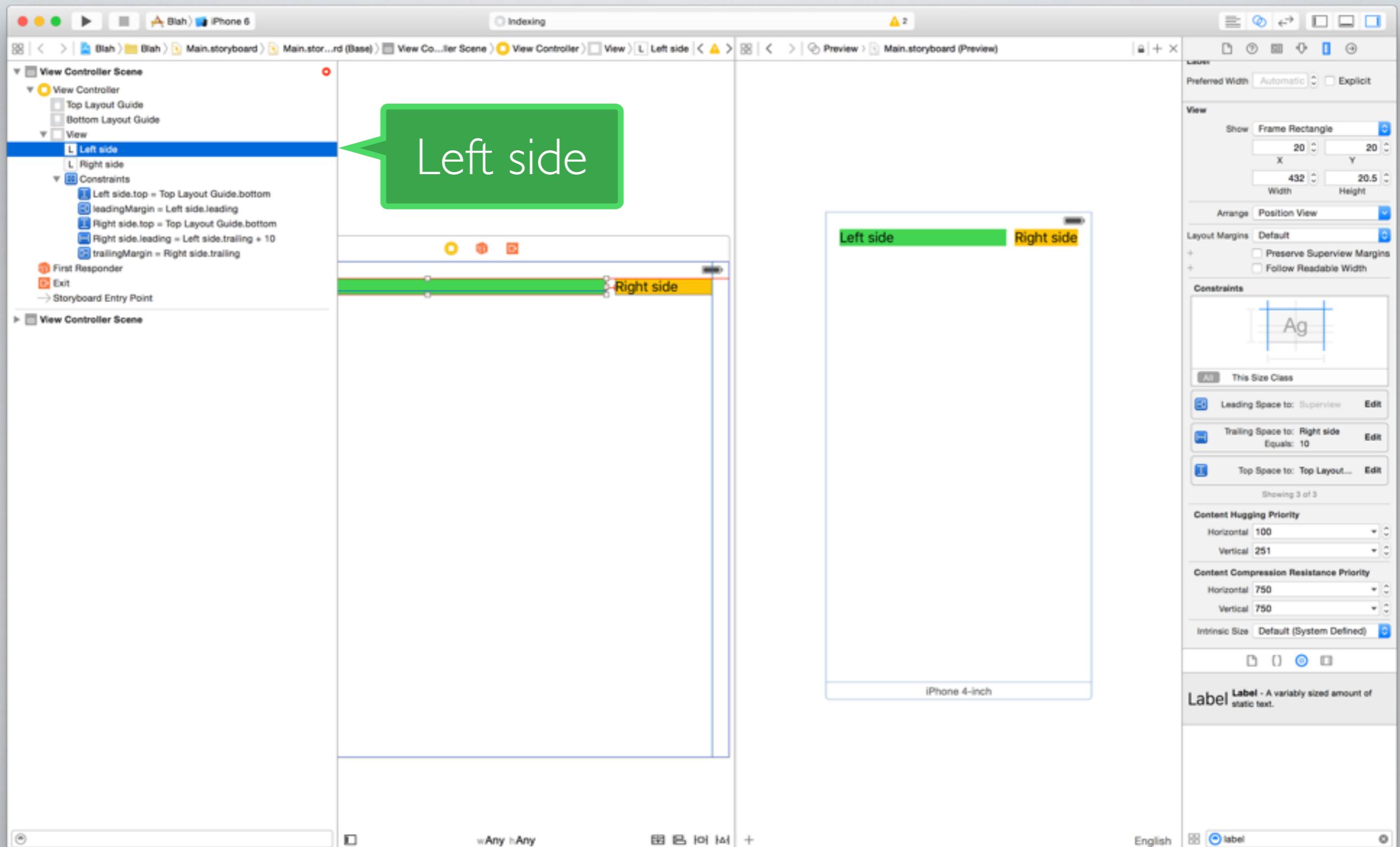


When content is smaller, which side gets to
“hug” its content tightly?

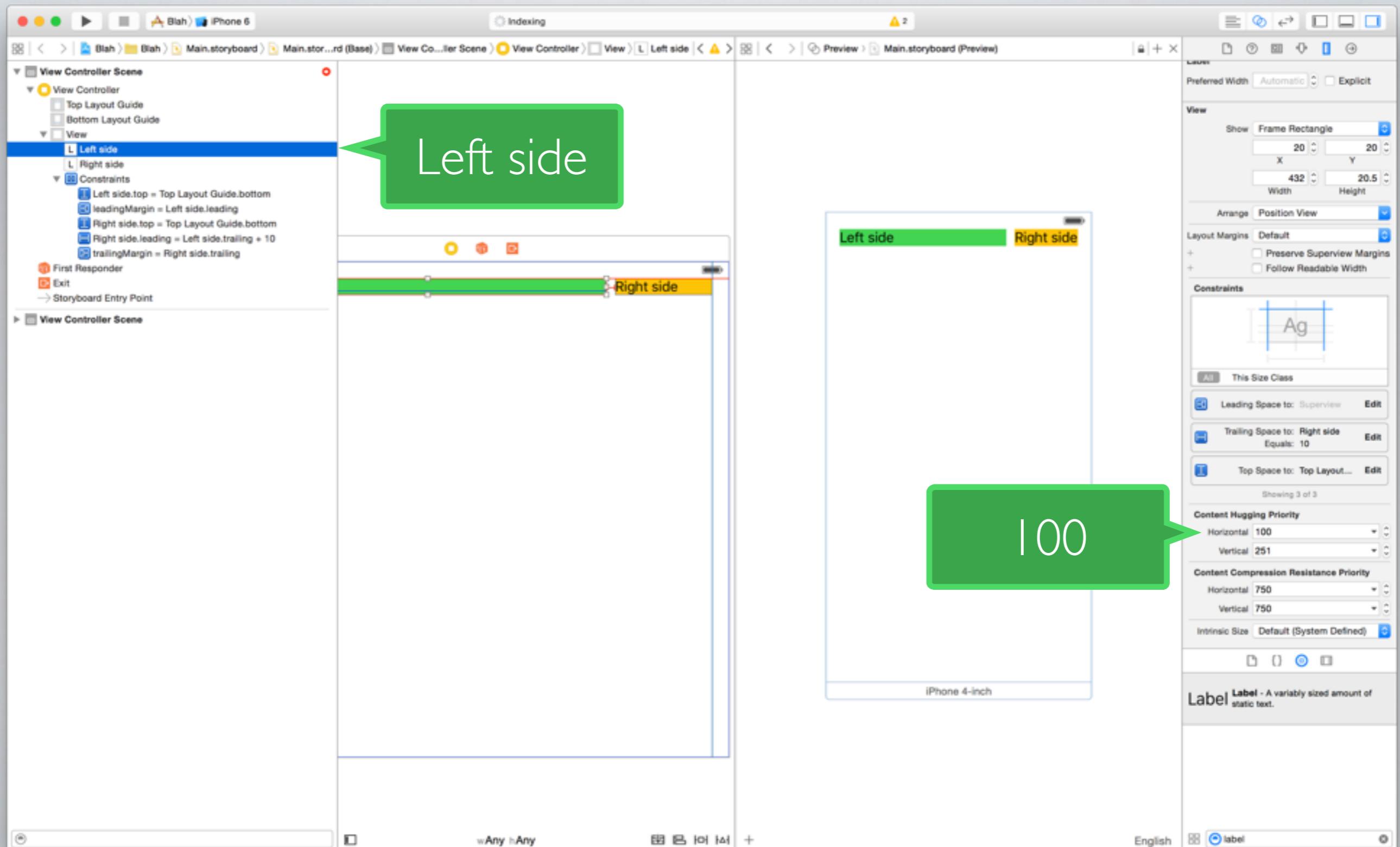
CONTENT HUGGING



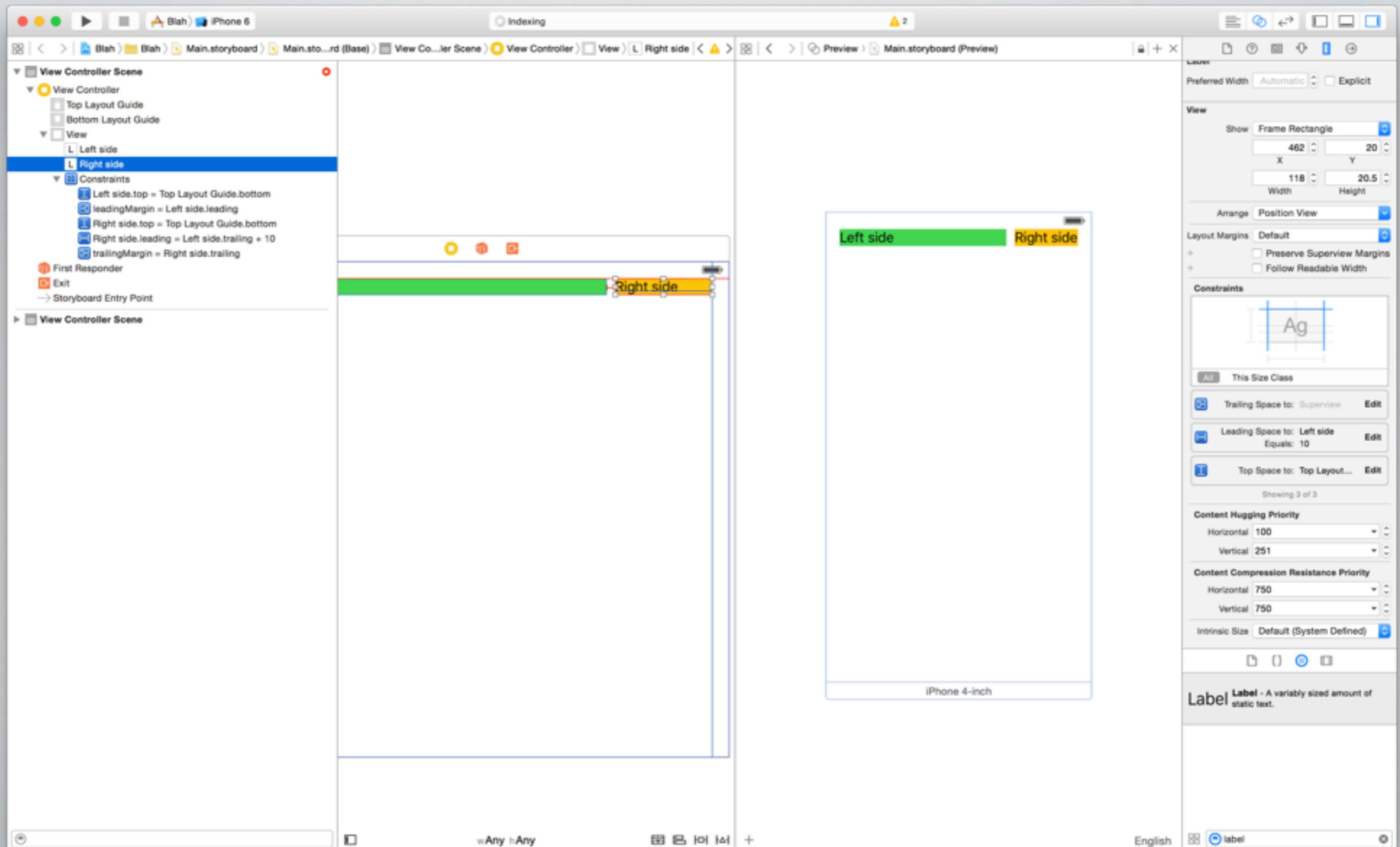
CONTENT HUGGING



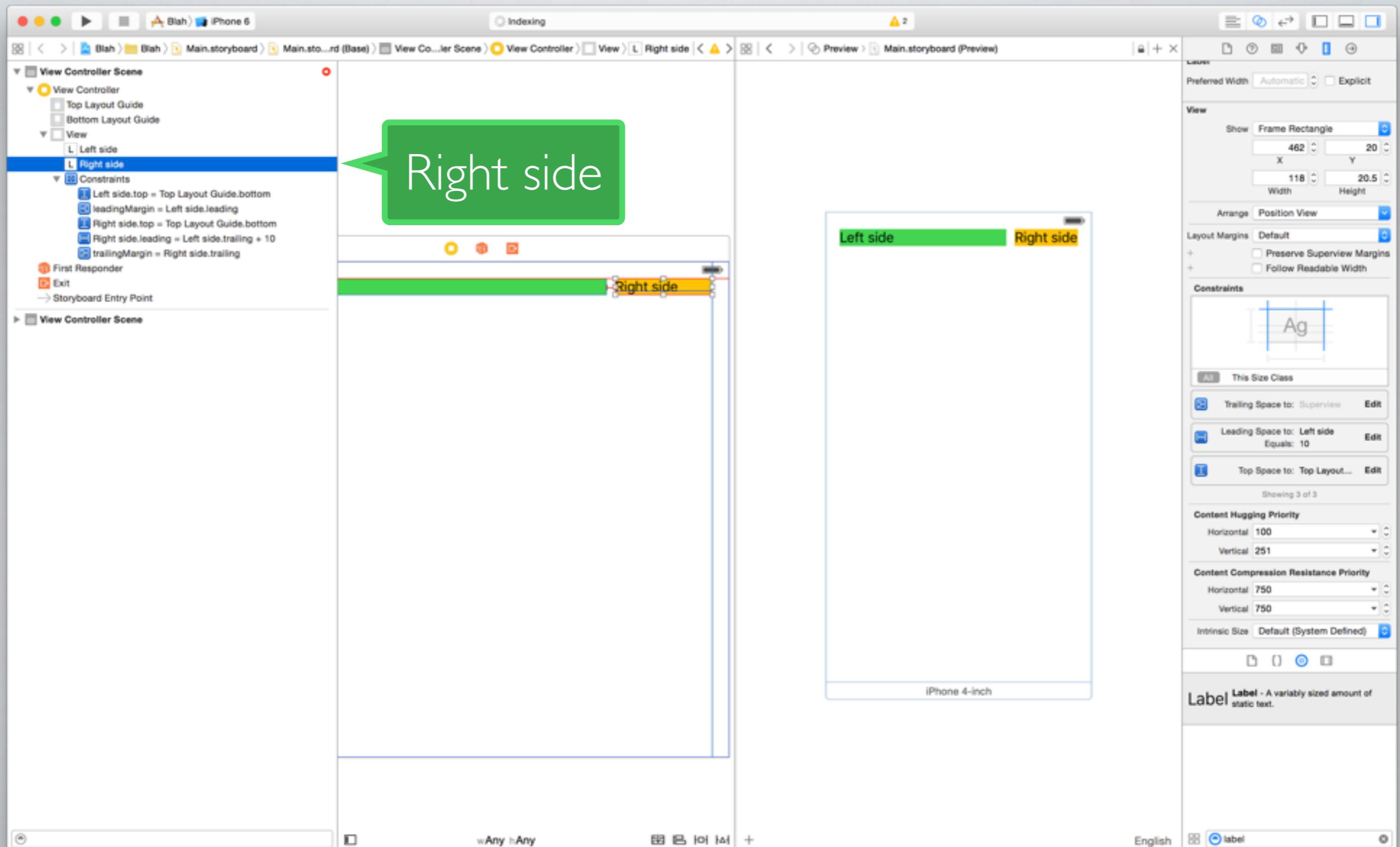
CONTENT HUGGING



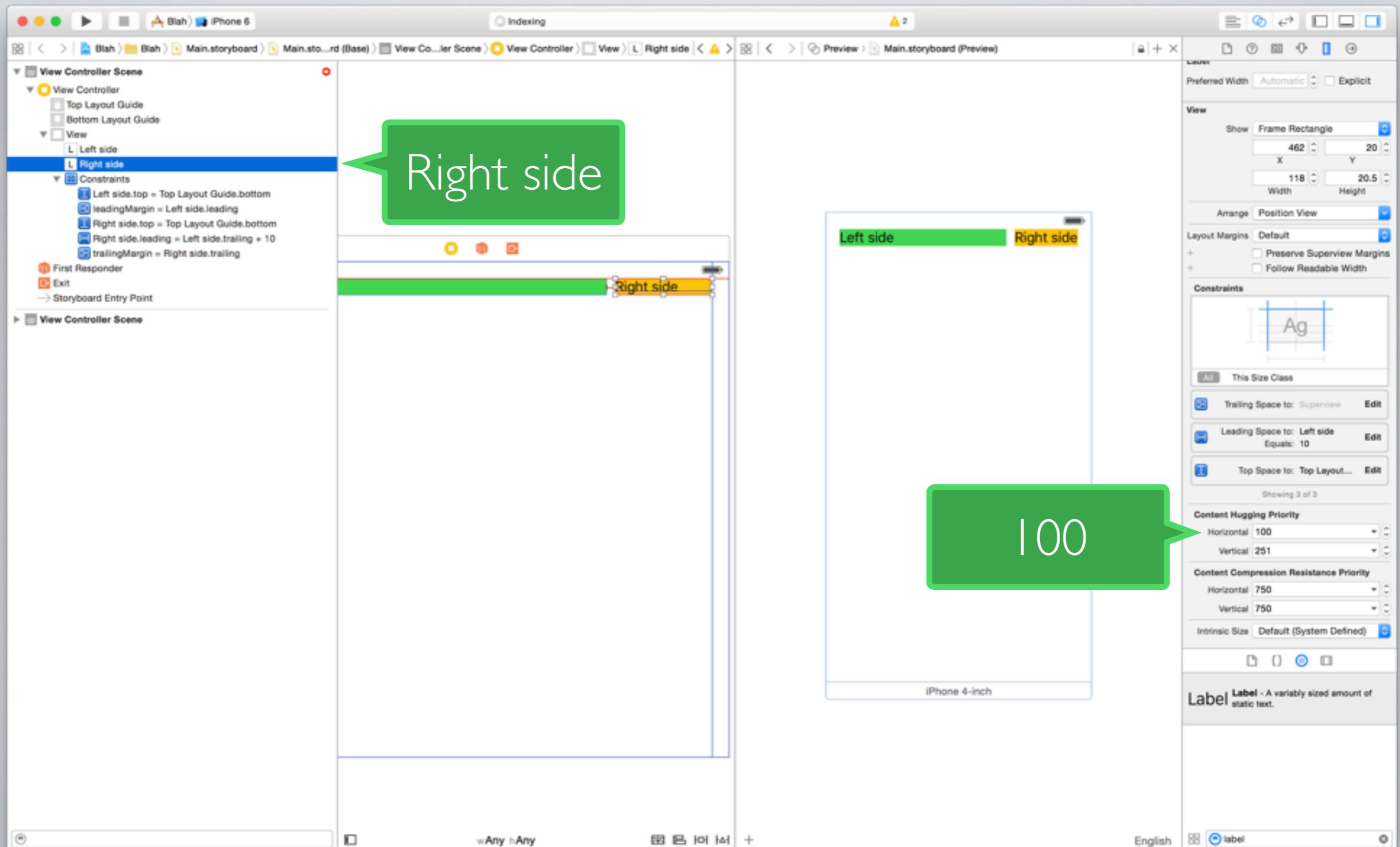
CONTENT HUGGING



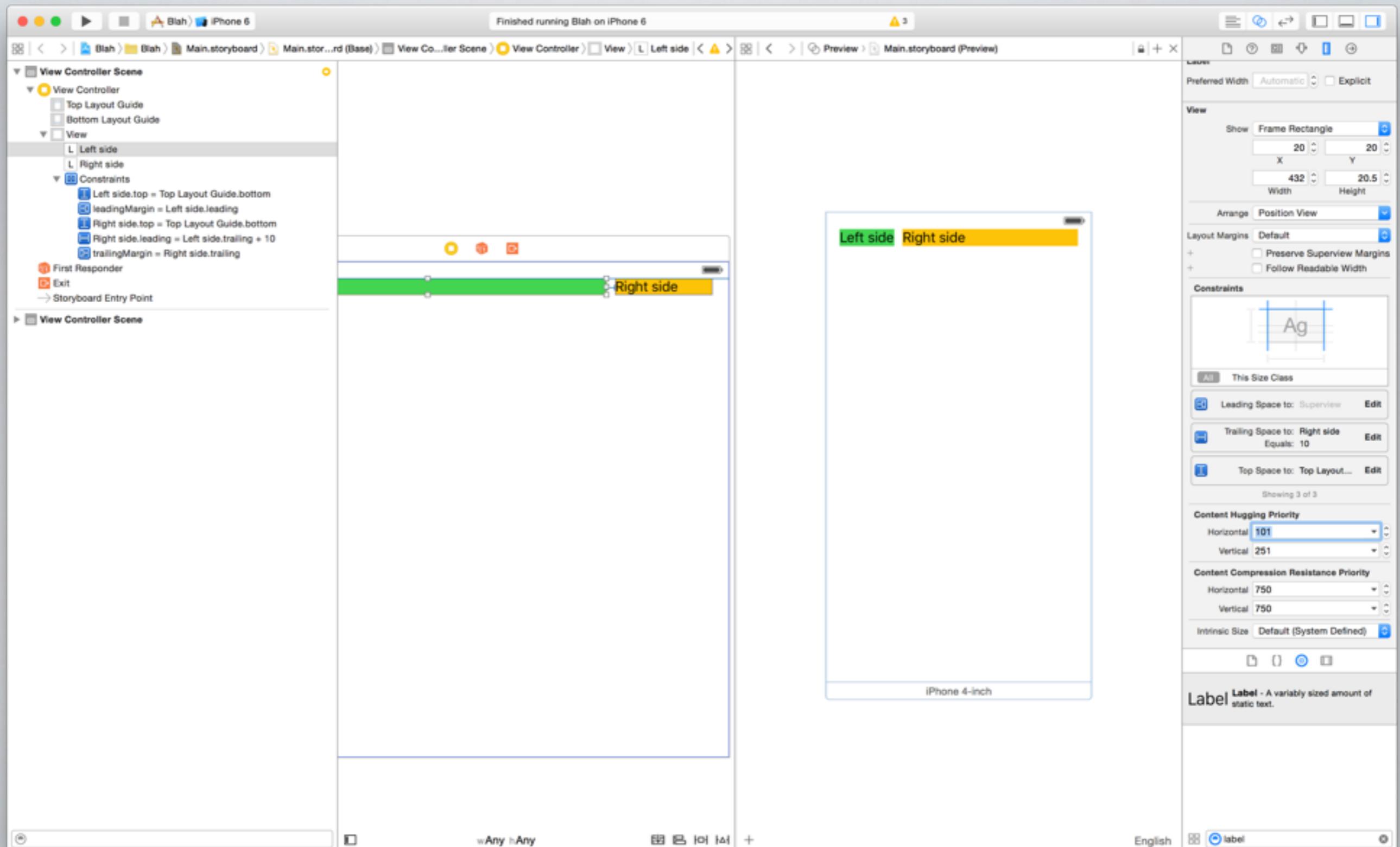
CONTENT HUGGING



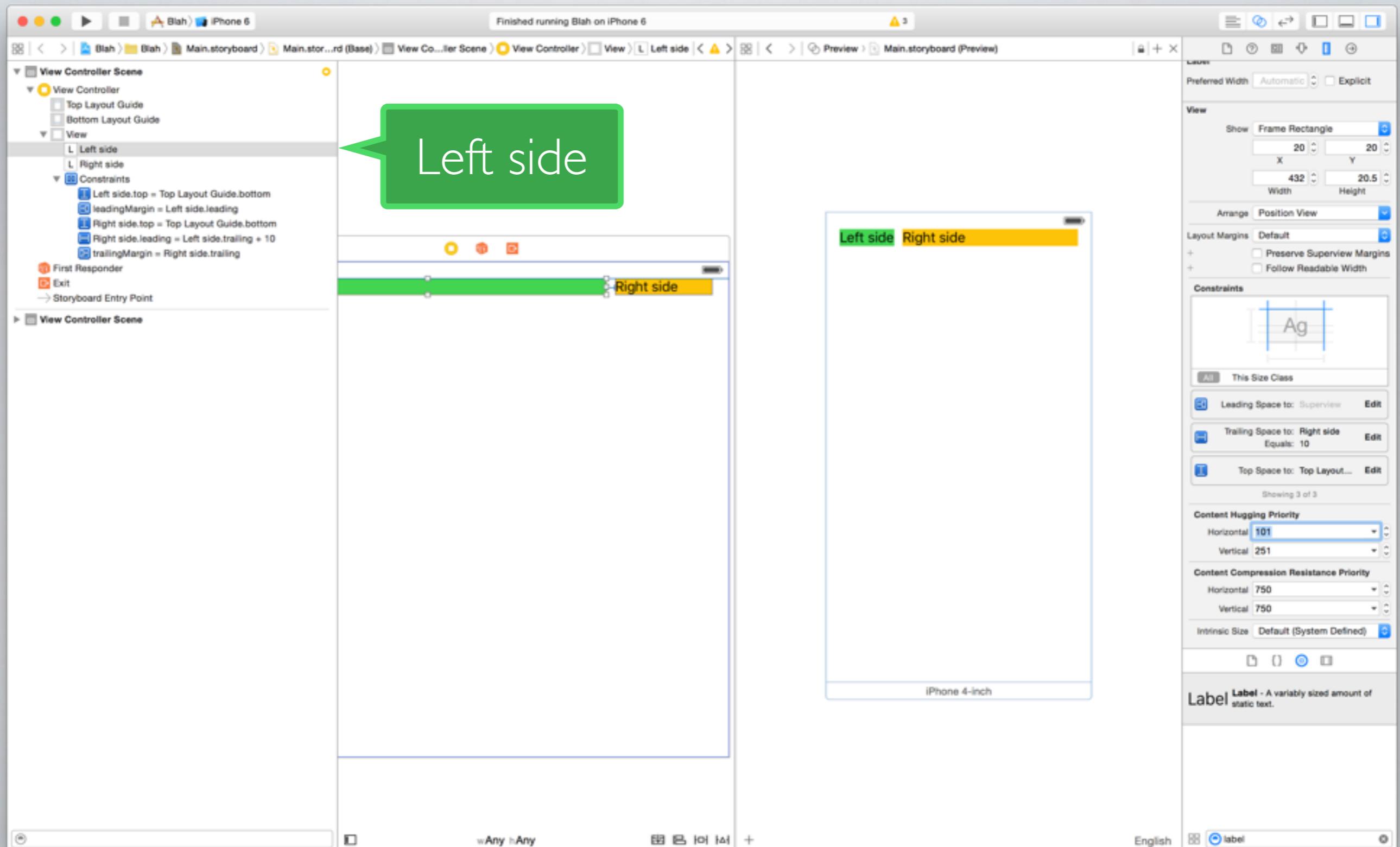
CONTENT HUGGING



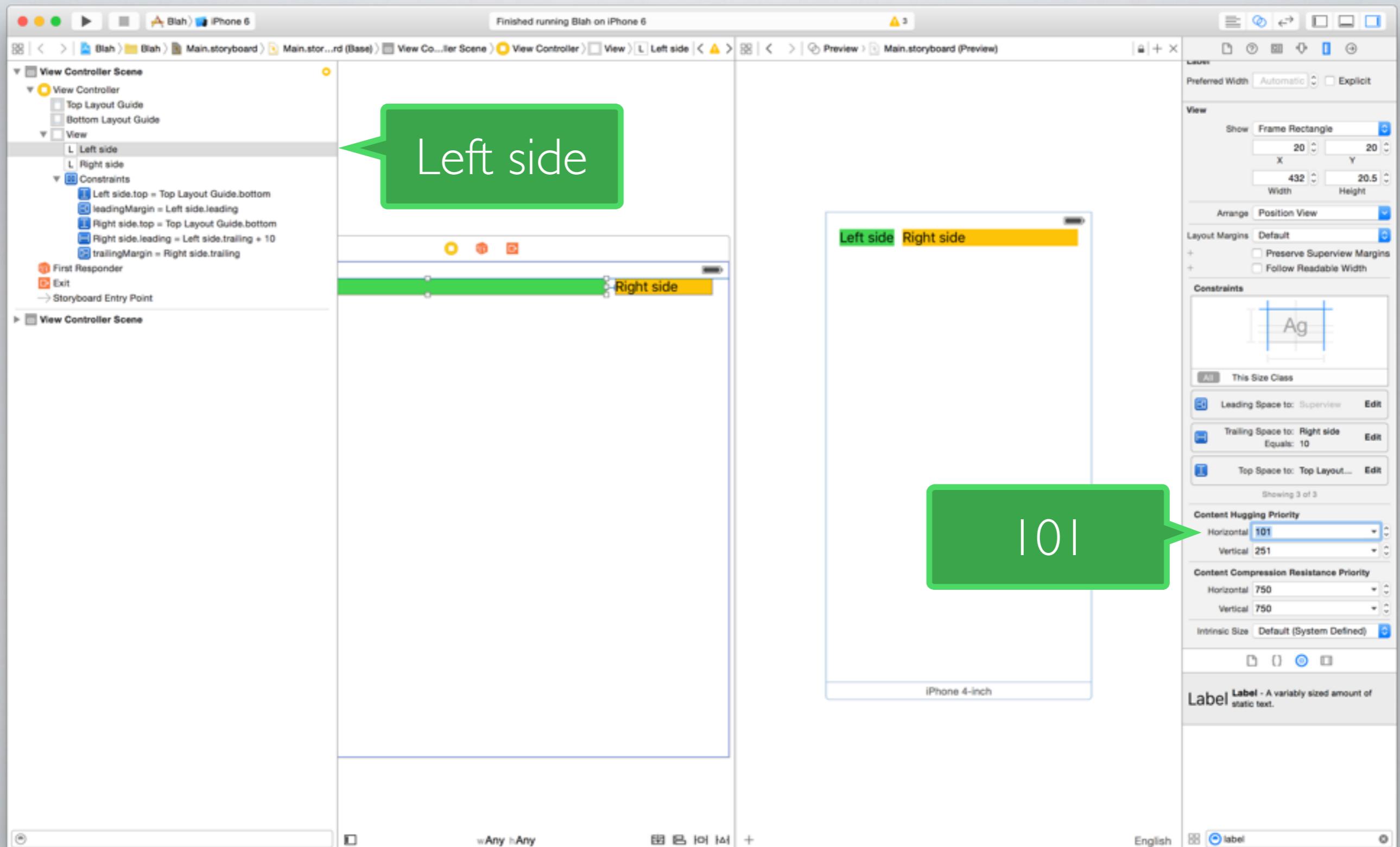
CONTENT HUGGING



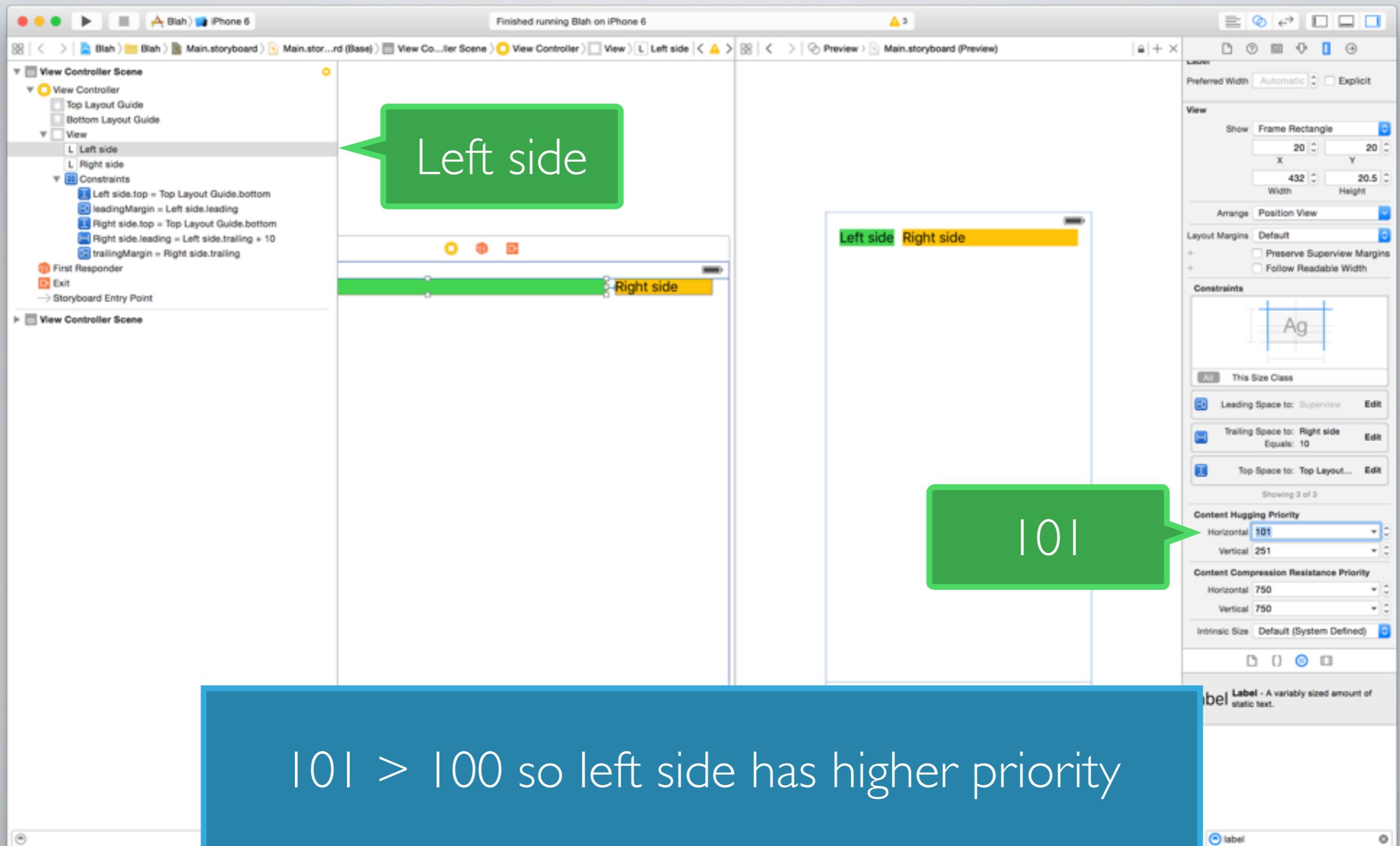
CONTENT HUGGING



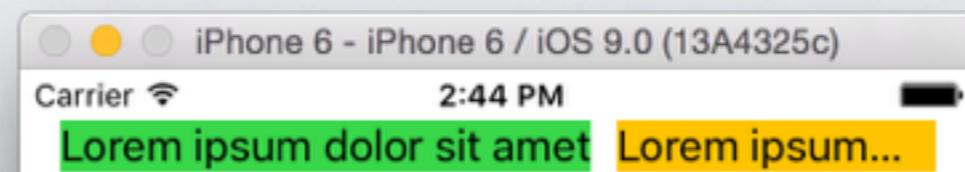
CONTENT HUGGING



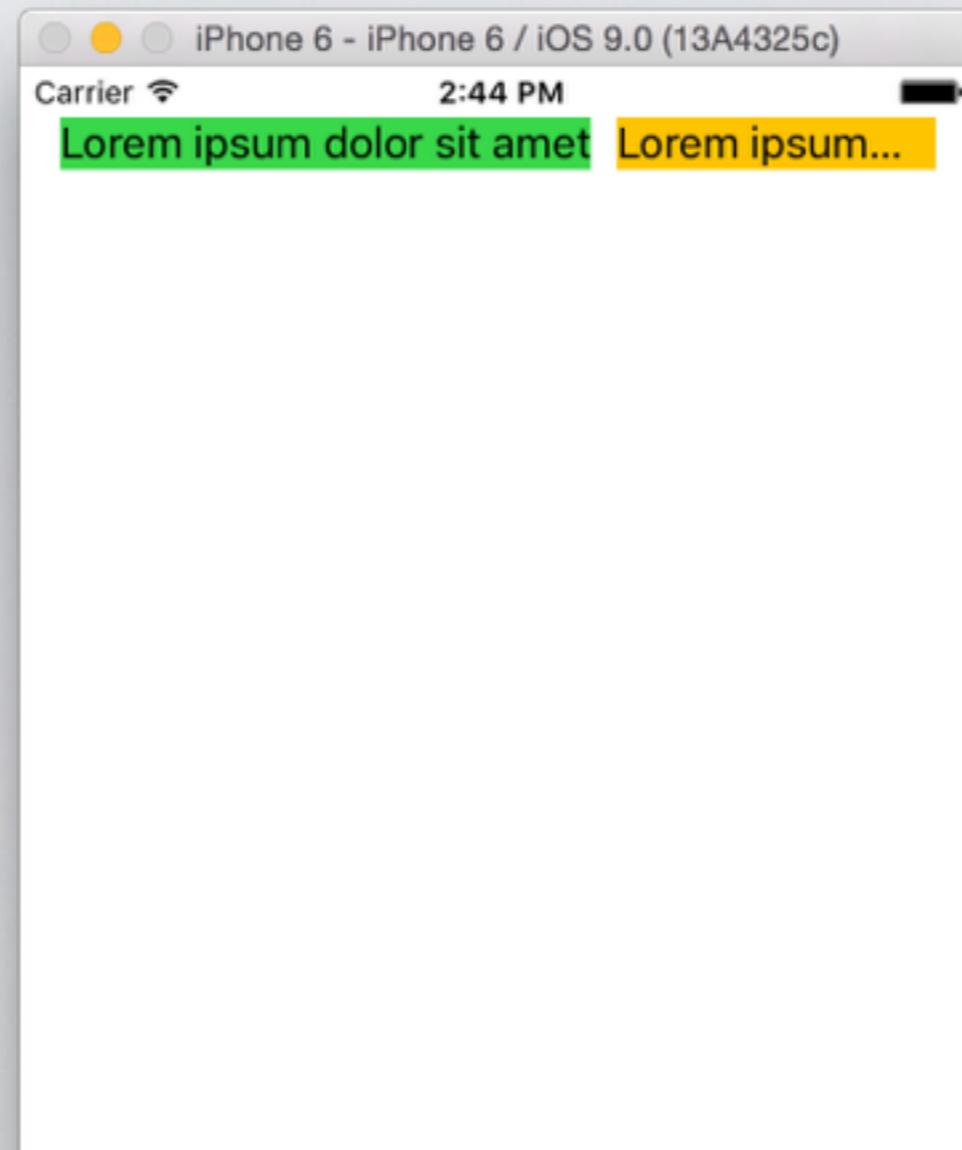
CONTENT HUGGING



COMPRESSION RESISTANCE

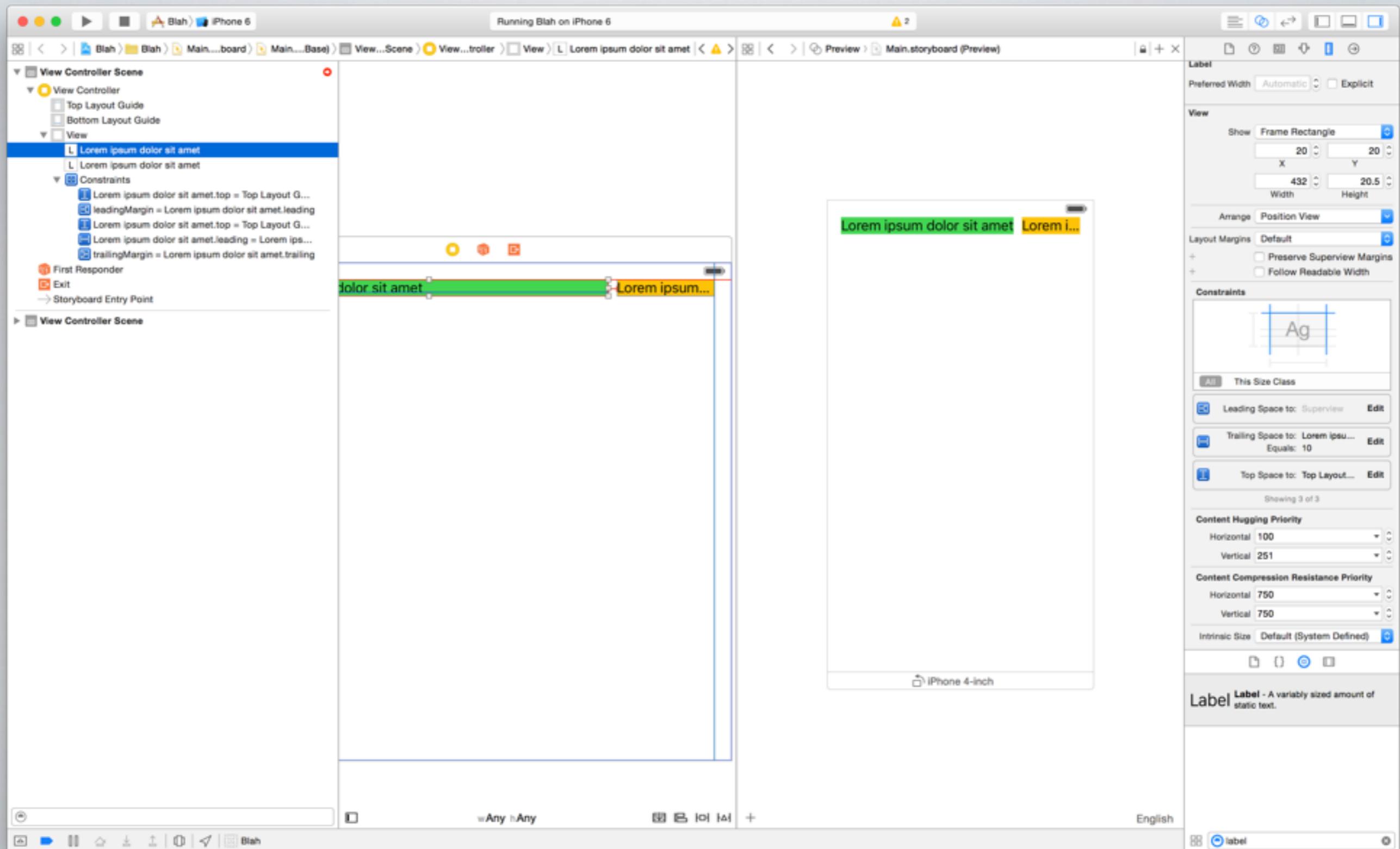


COMPRESSION RESISTANCE

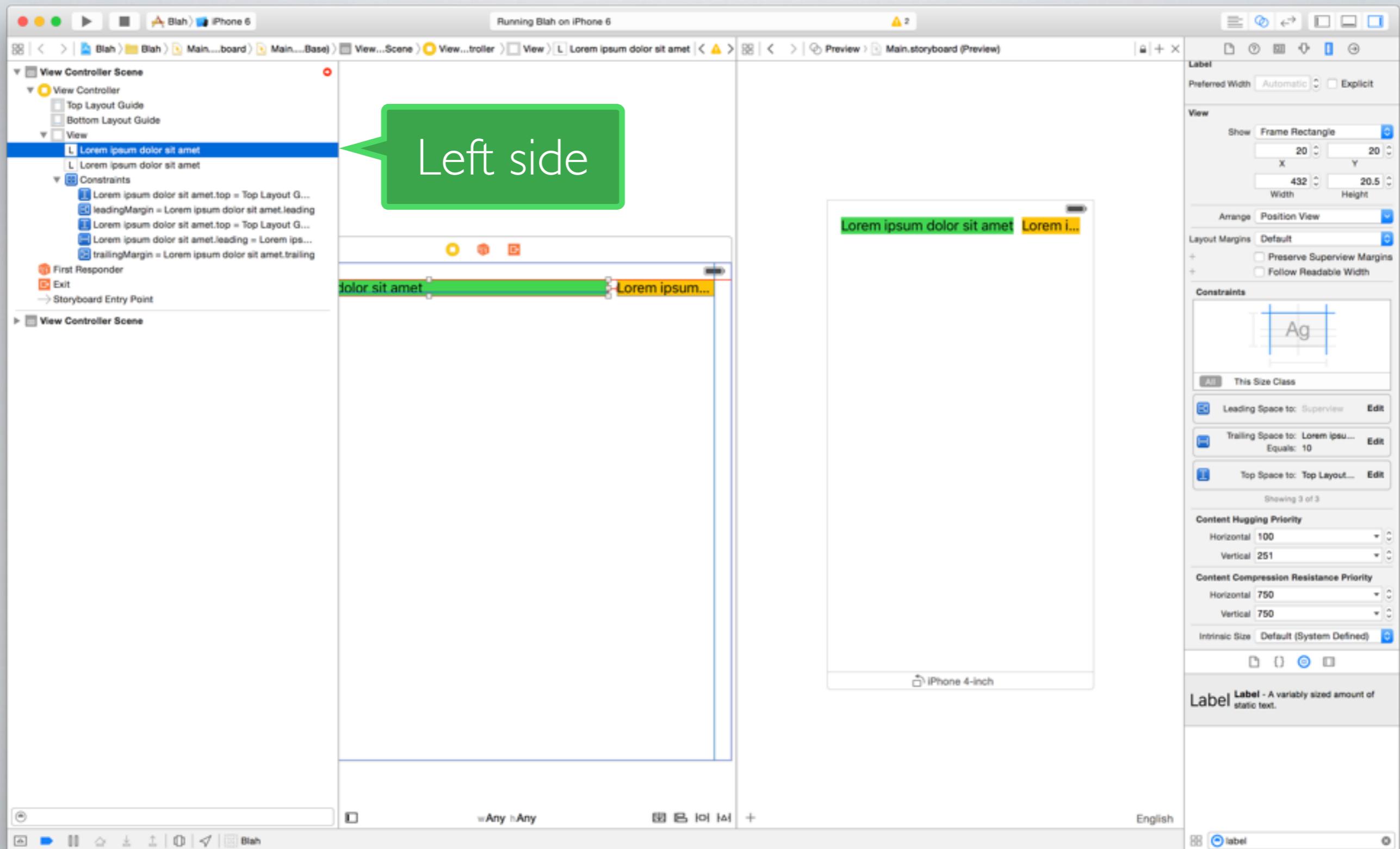


When content is larger, which side gets to “hug” its content tightly?

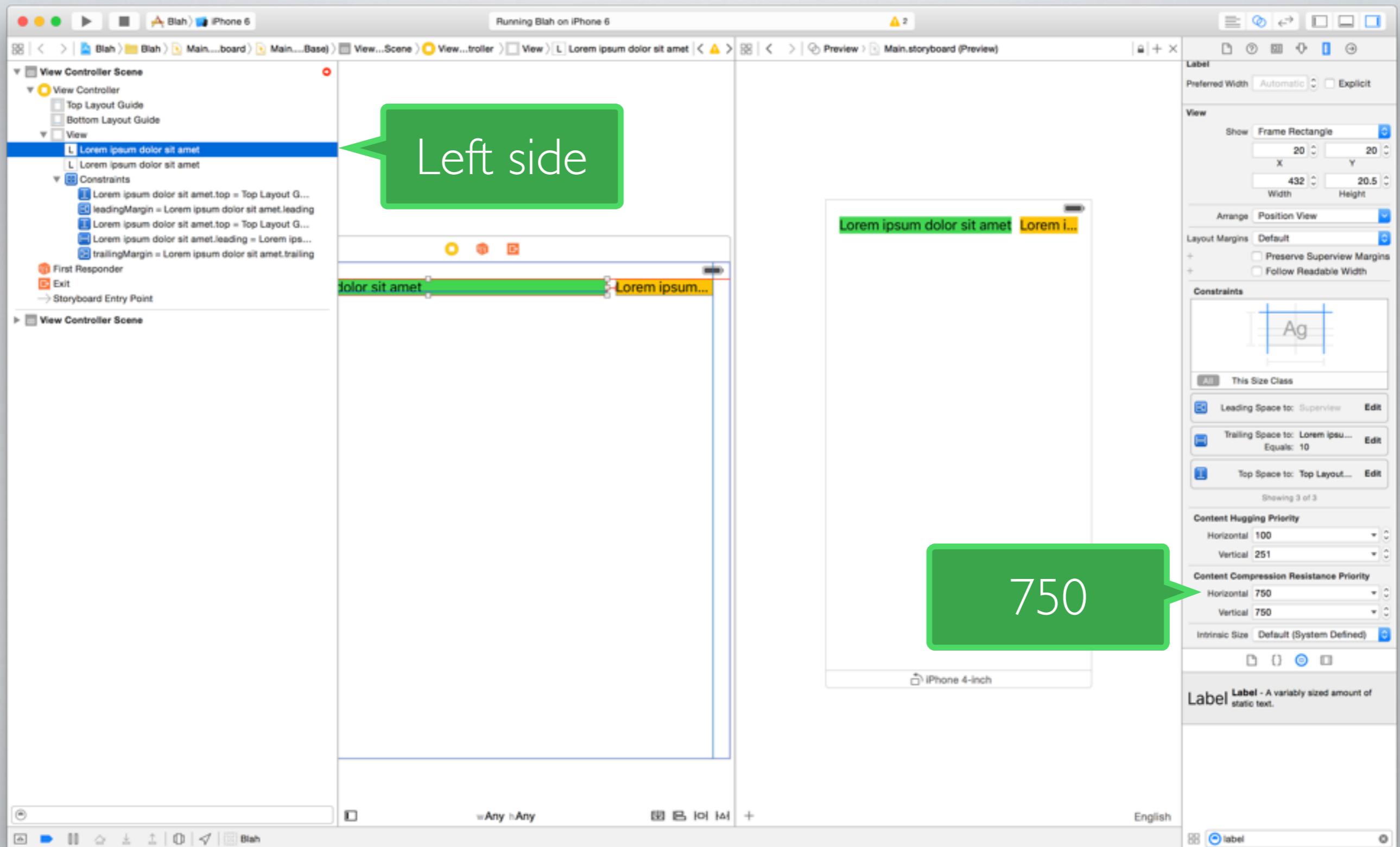
COMPRESSION RESISTANCE



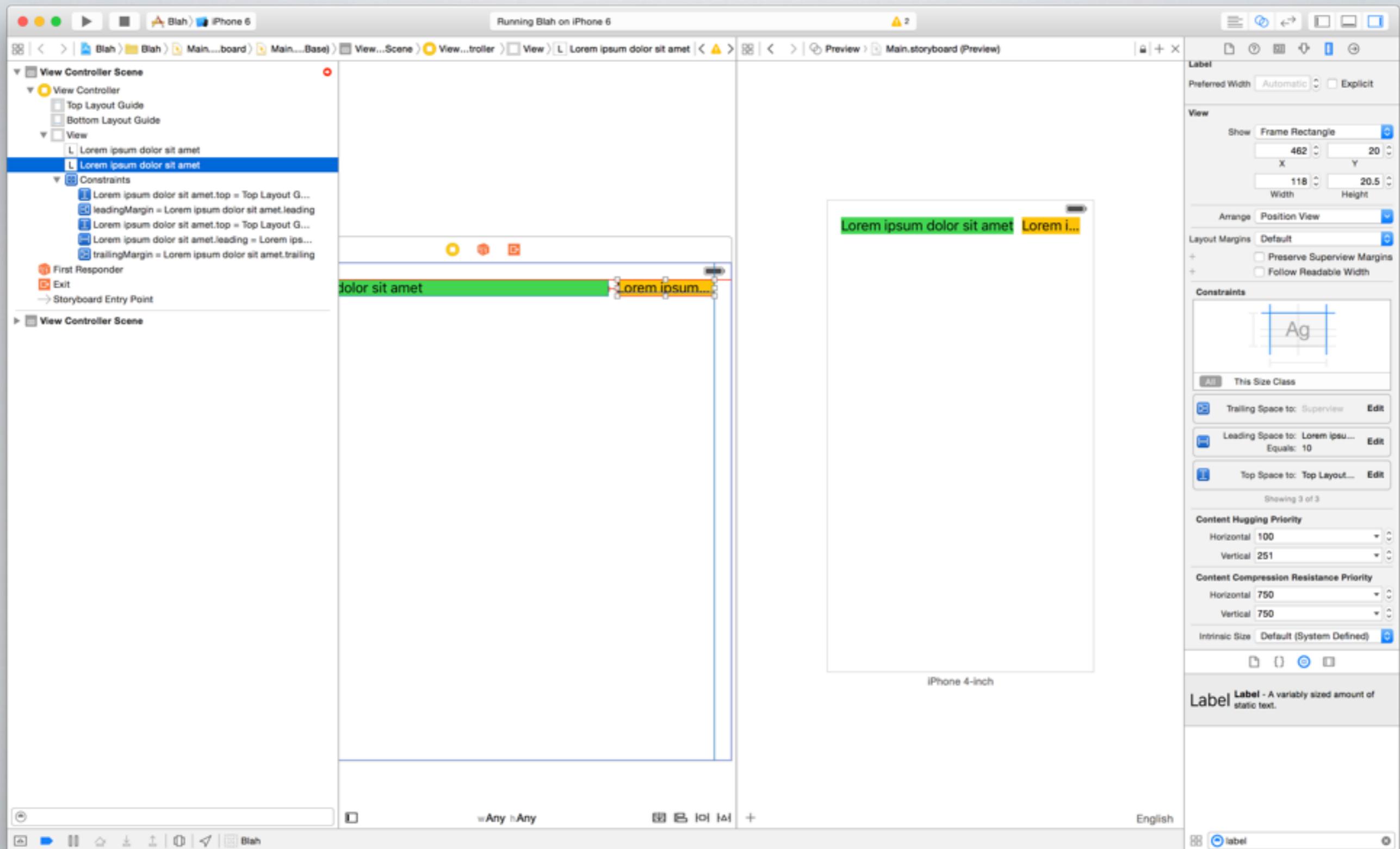
COMPRESSION RESISTANCE



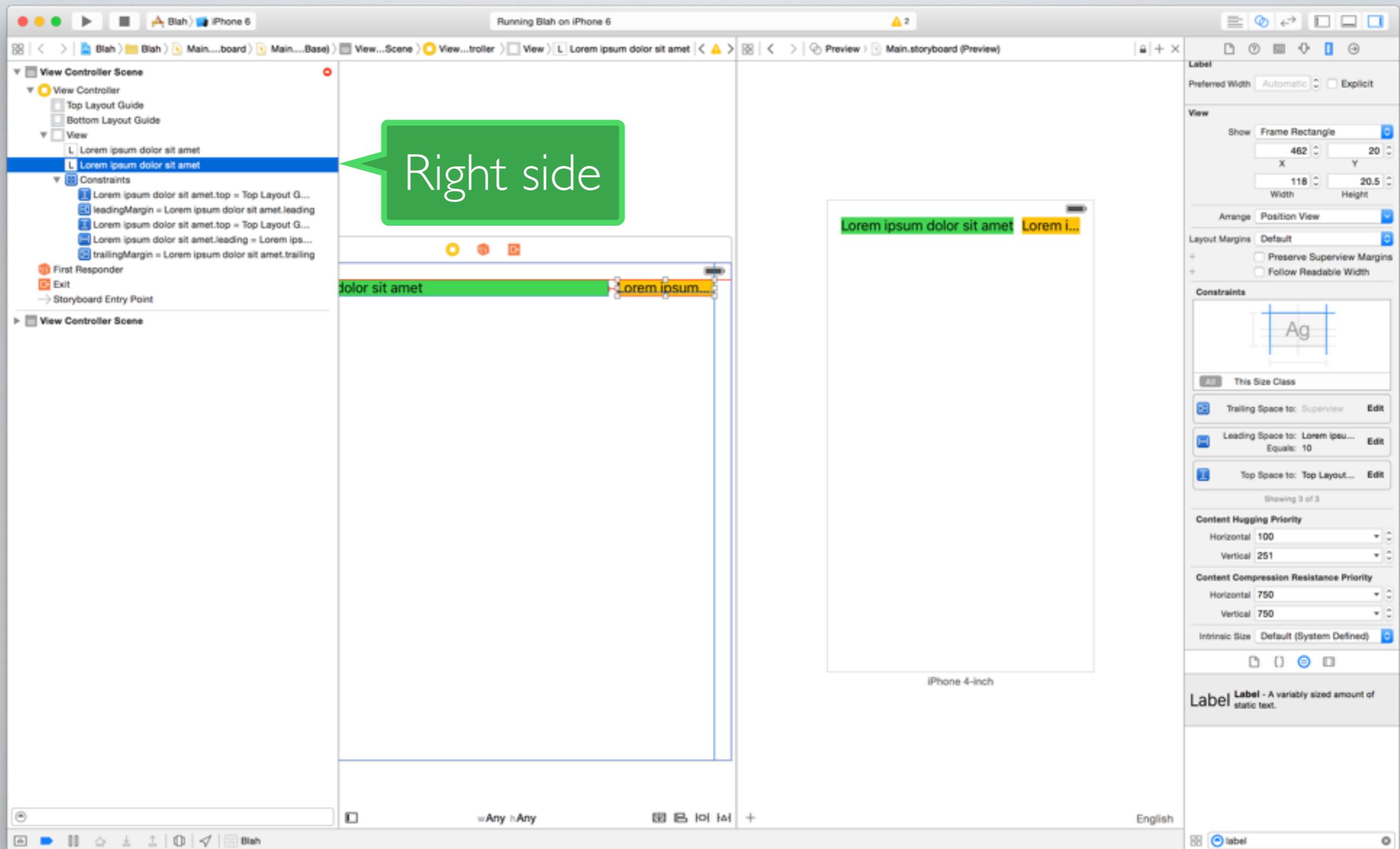
COMPRESSION RESISTANCE



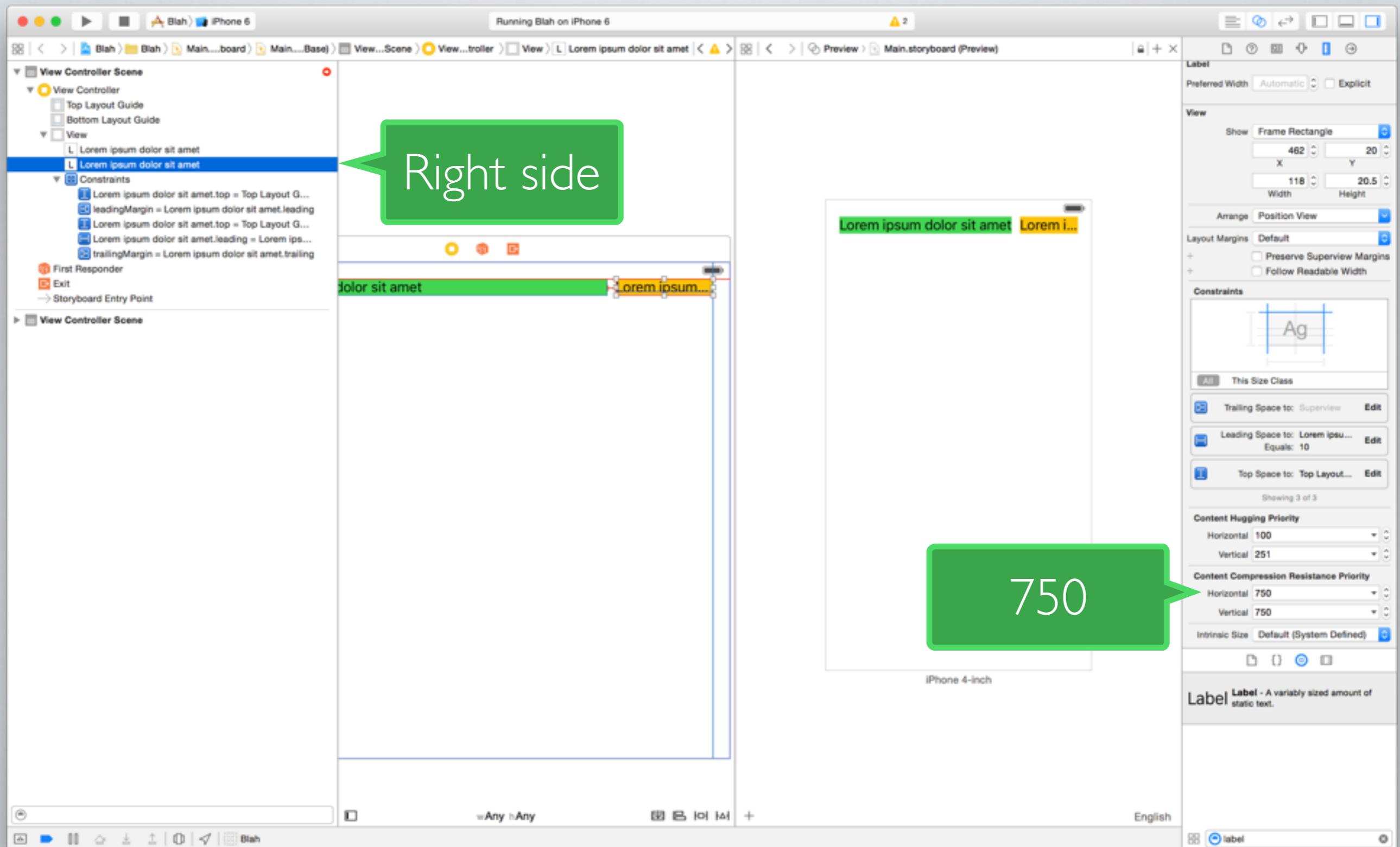
COMPRESSION RESISTANCE



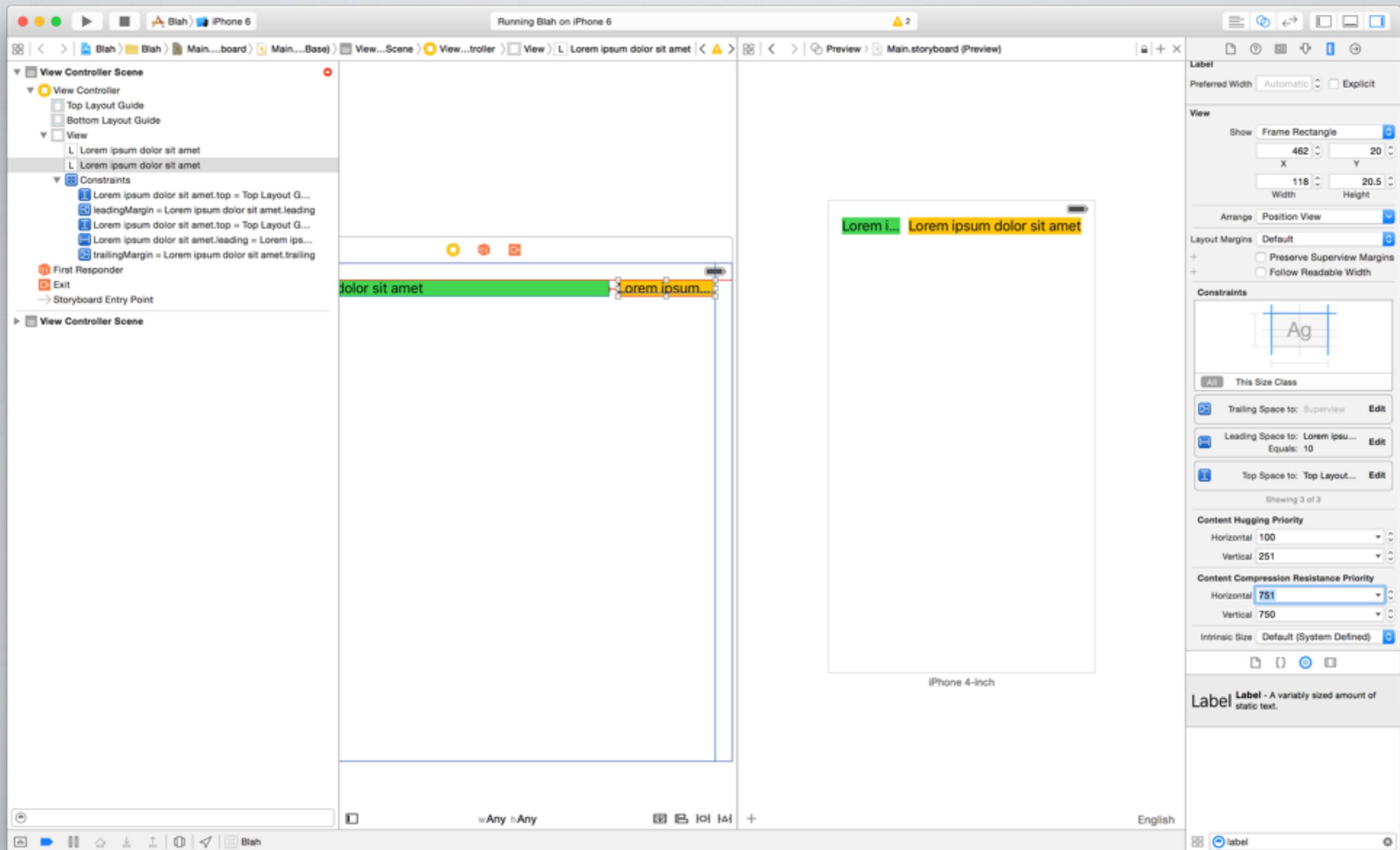
COMPRESSION RESISTANCE



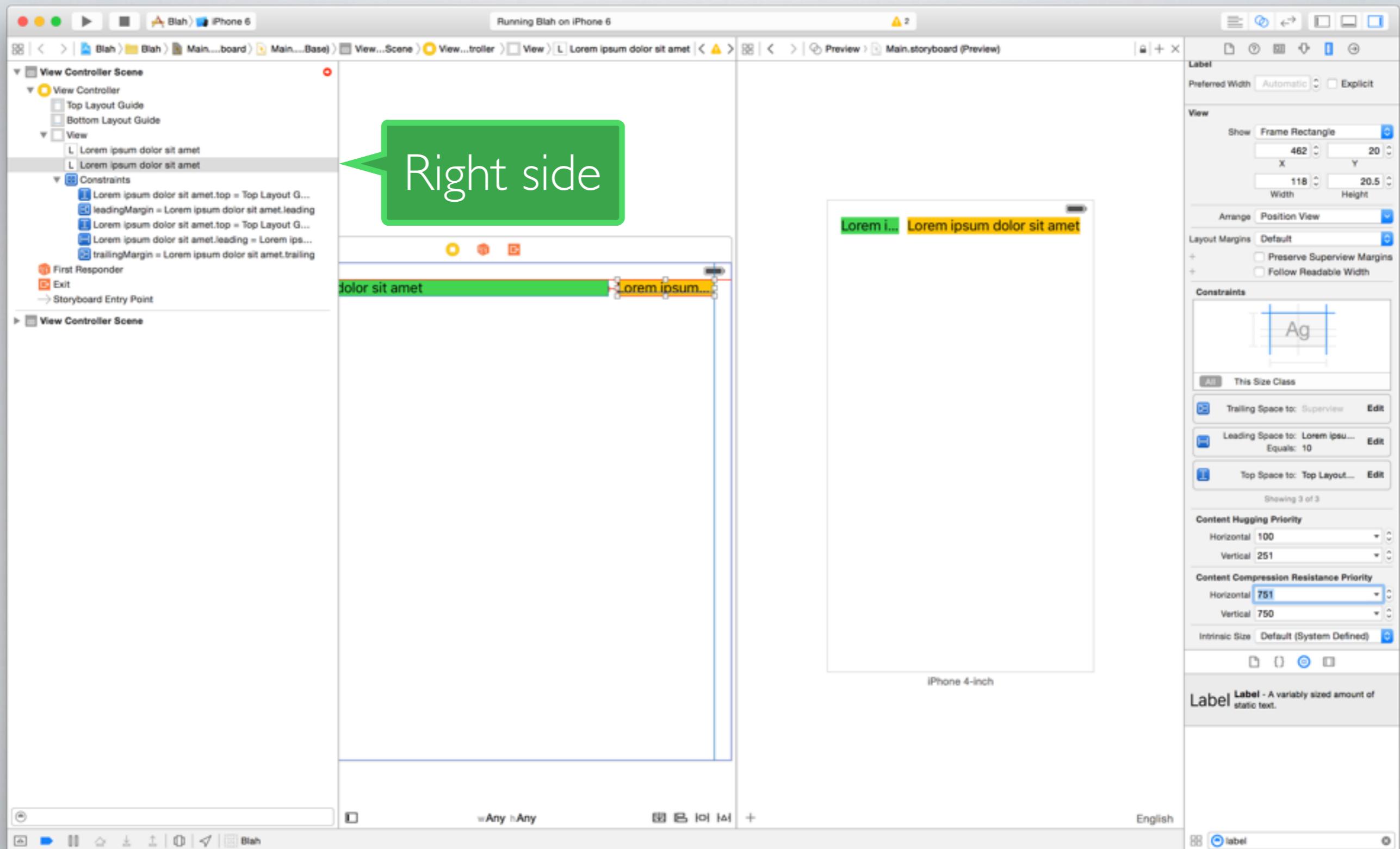
COMPRESSION RESISTANCE



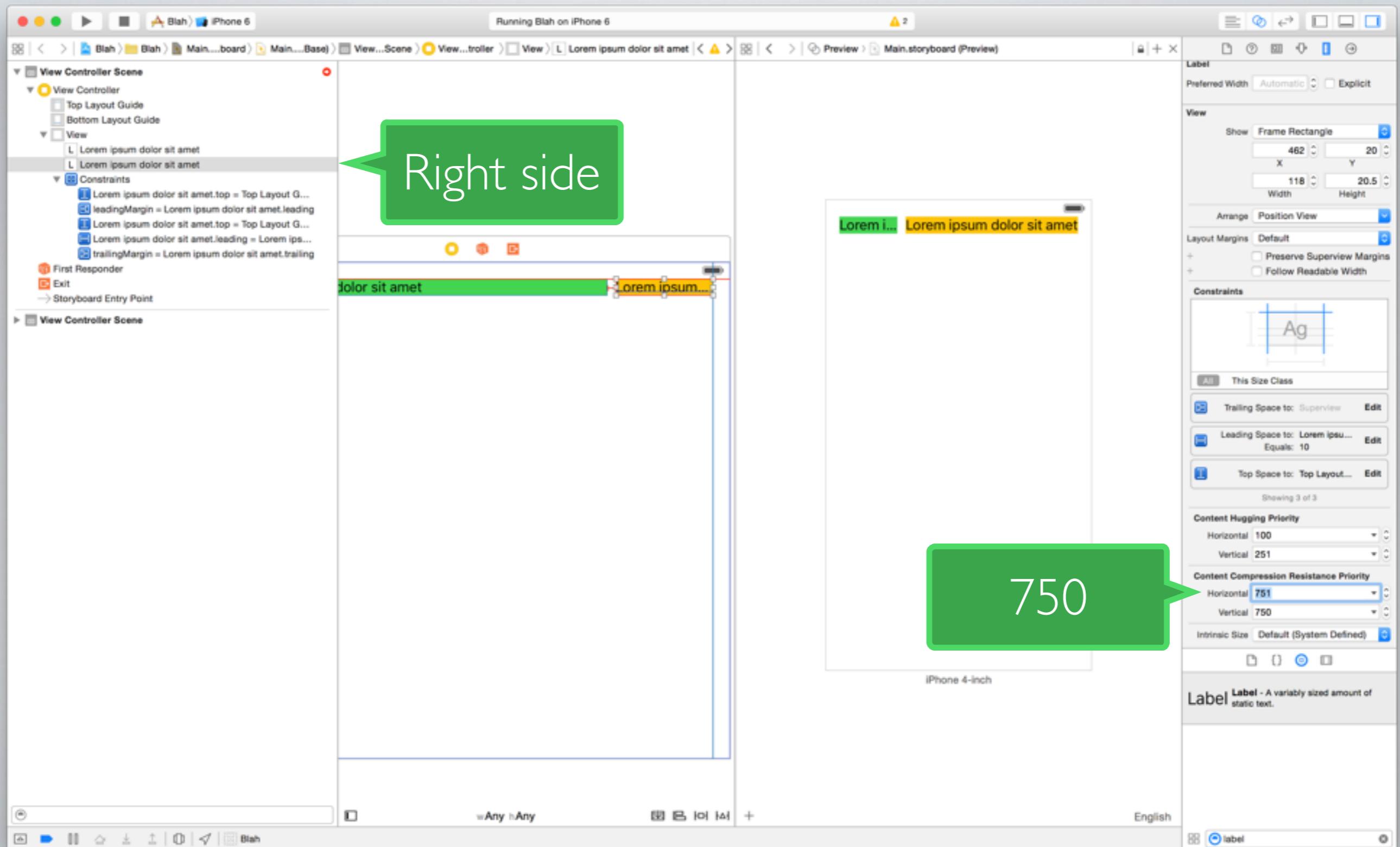
COMPRESSION RESISTANCE



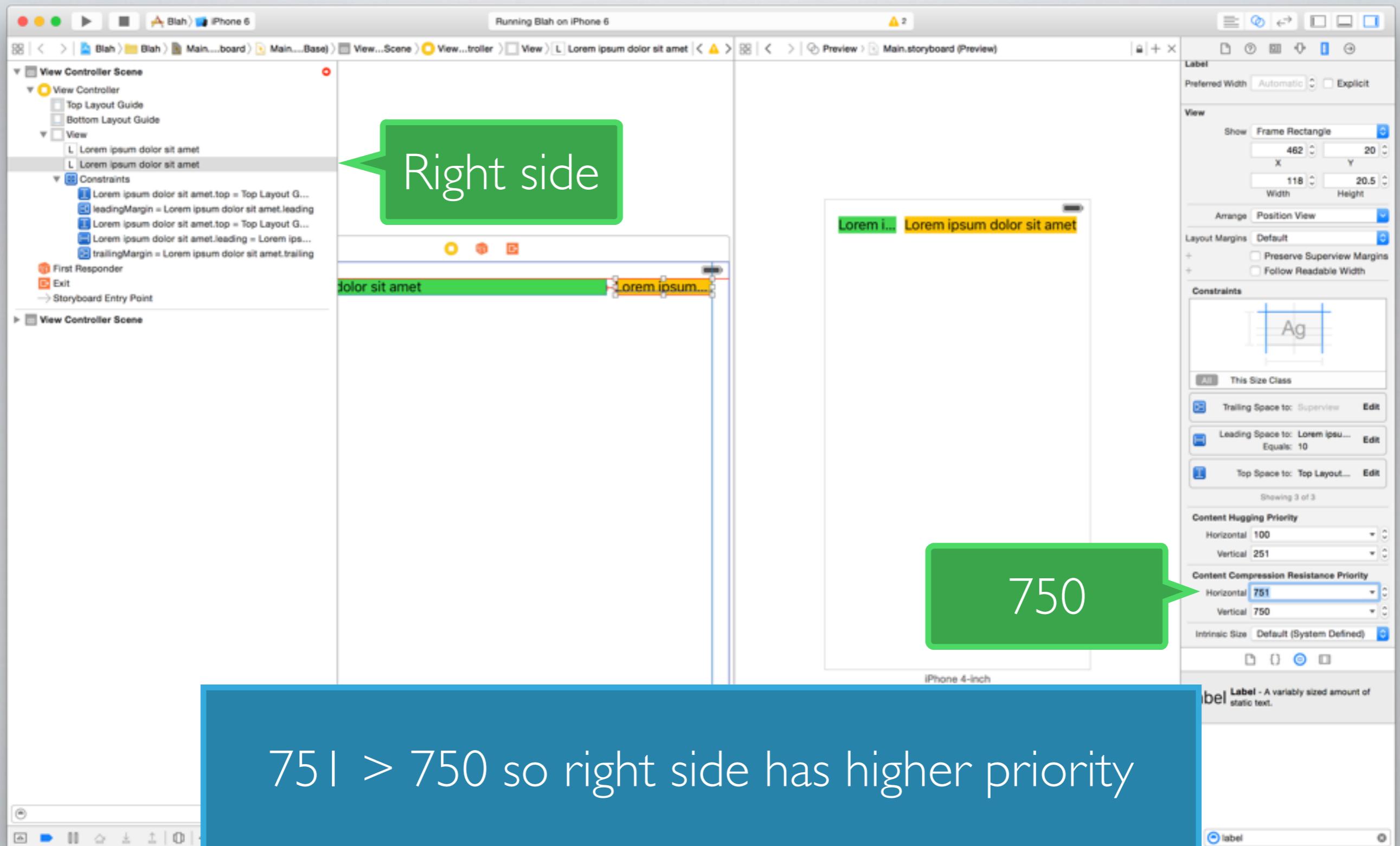
COMPRESSION RESISTANCE



COMPRESSION RESISTANCE



COMPRESSION RESISTANCE



SIZE CLASSES



SIZE CLASSES



We know how to make things look good on
different orientations

SIZE CLASSES



We know how to make things look good on different orientations

SIZE CLASSES



We know how to make things look good on
different orientations

SIZE CLASSES



SIZE CLASSES



Can we make things look *different* based on the orientation?

SIZE CLASSES



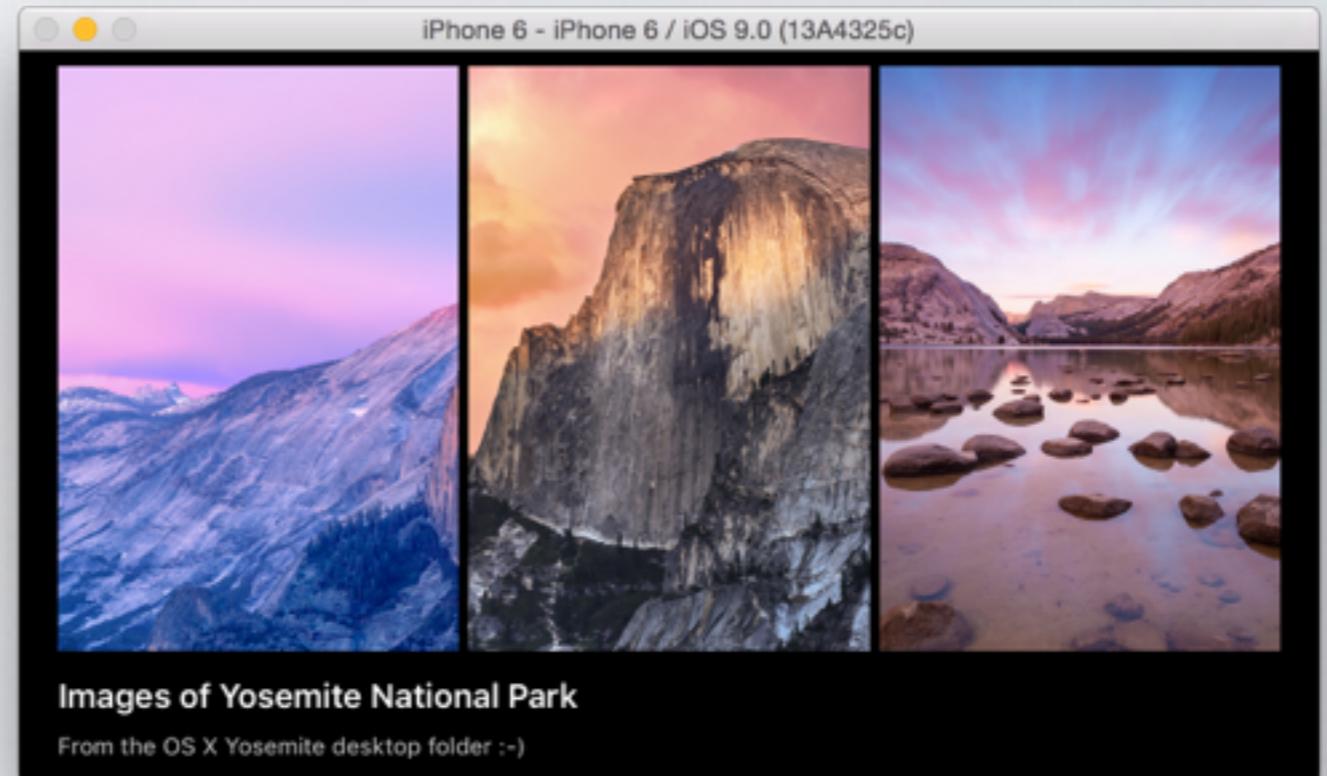
Can we make things look *different* based on the orientation?

SIZE CLASSES



Can we make things look *different* based on the orientation?

SIZE CLASSES



Images of

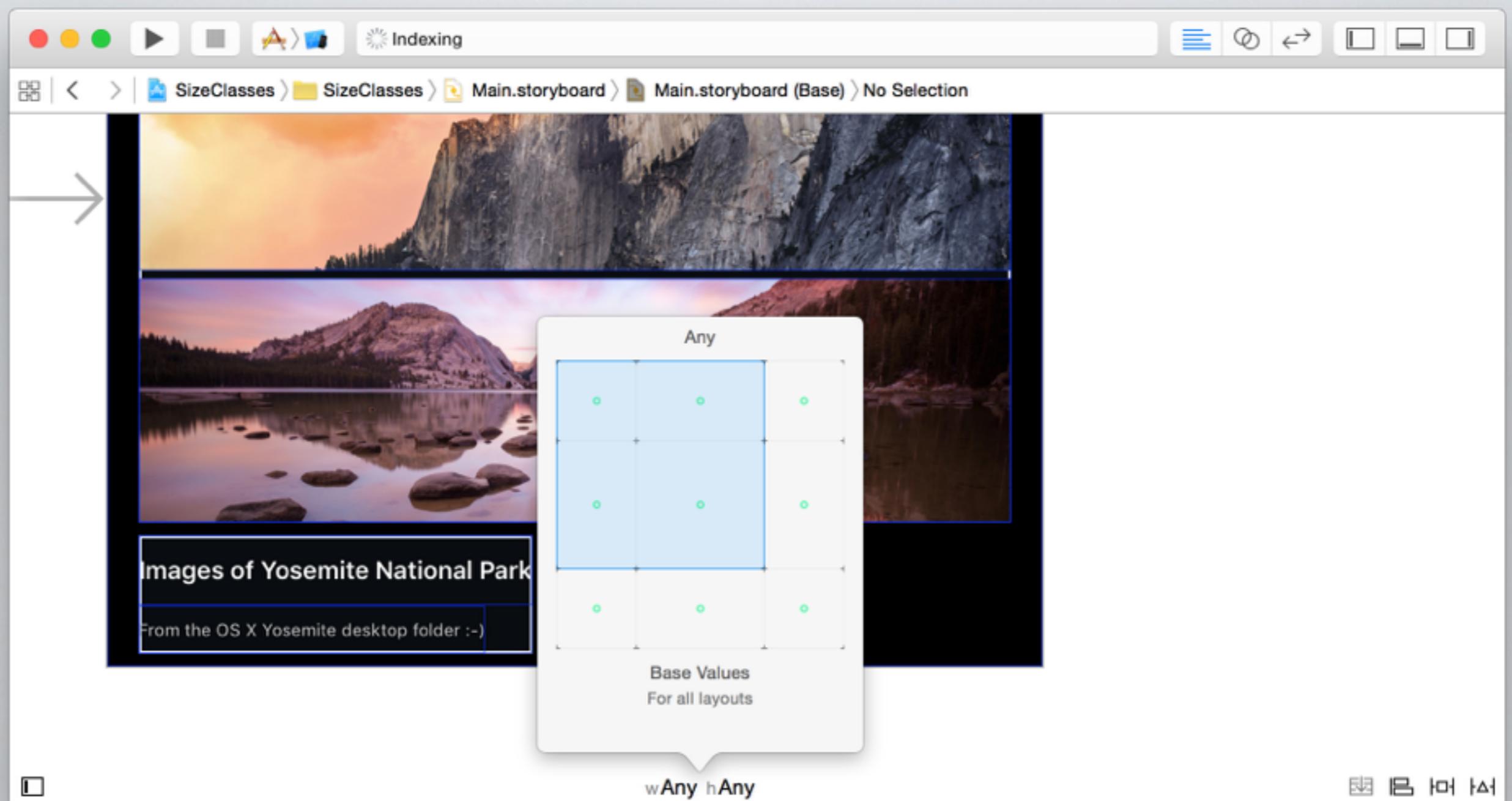
From the OS X

Can we make things look *different* based on the orientation?

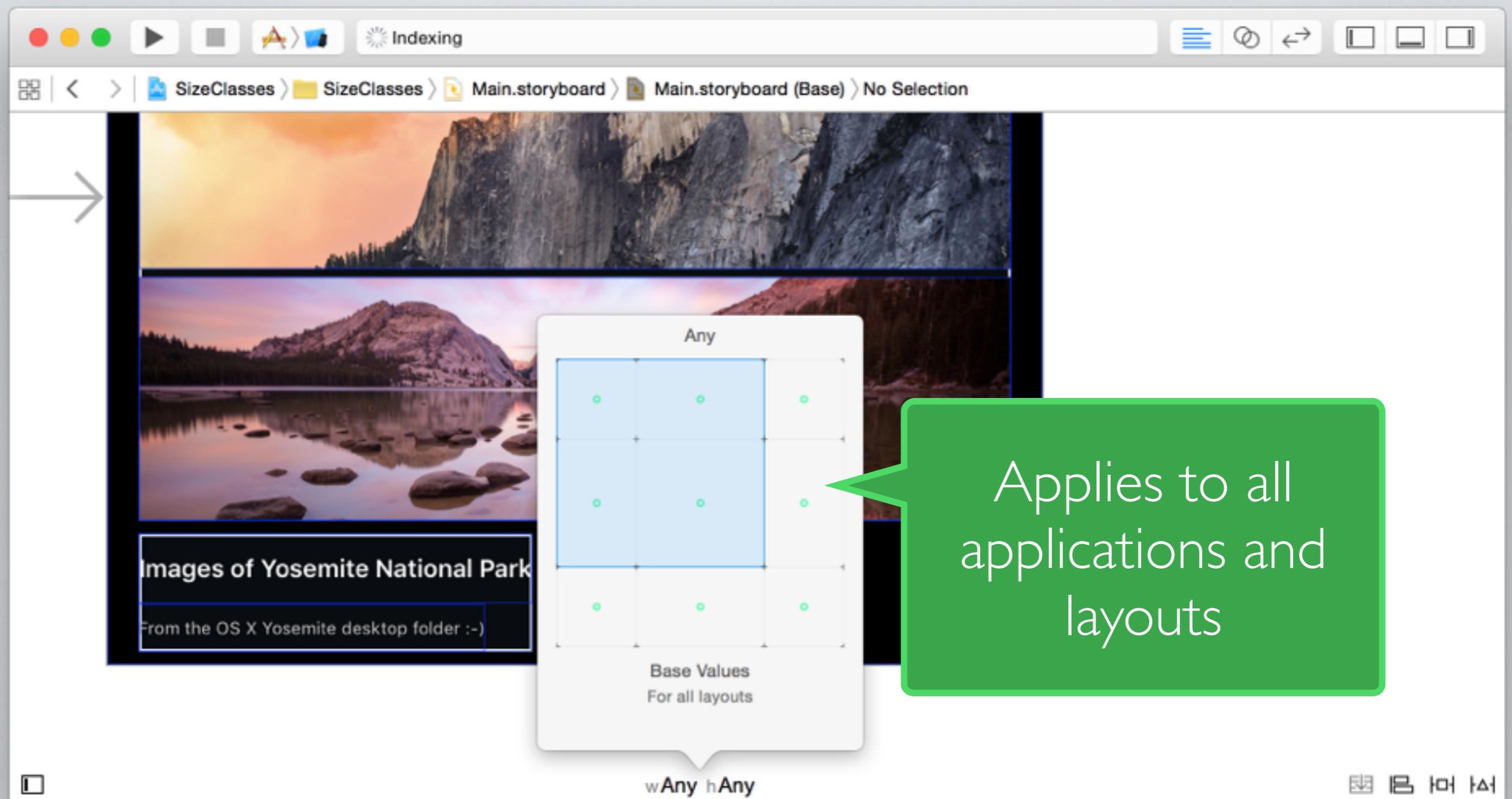
EXERCISE

- Recreate the example using only Storyboards and auto layout
- You can find the images in /Library/Desktop Pictures/
- Scale the images down first – they are huge!

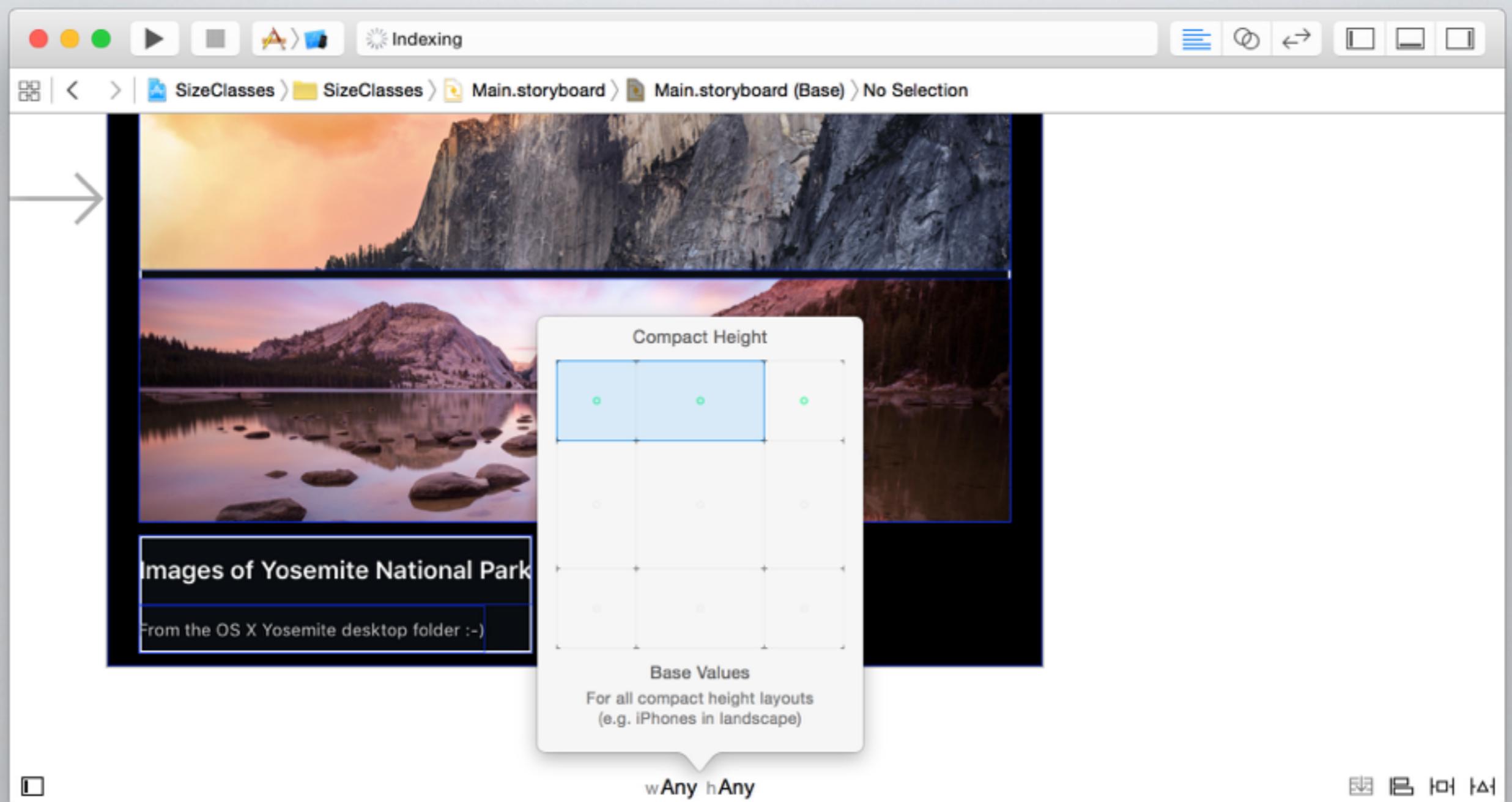
SIZE CLASSES



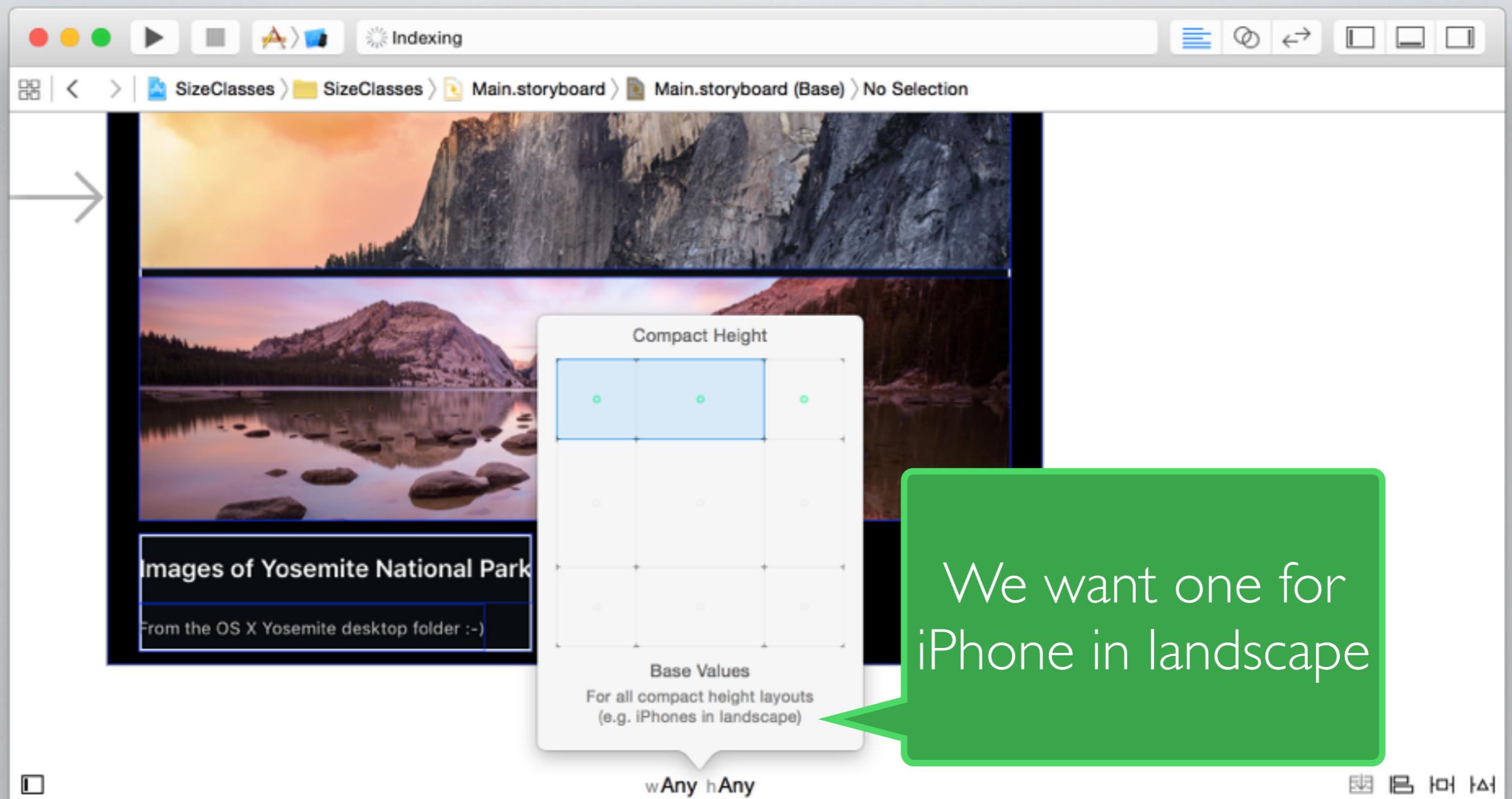
SIZE CLASSES



SIZE CLASSES



SIZE CLASSES



Regular Height
Compact Width

iPhone
Portrait

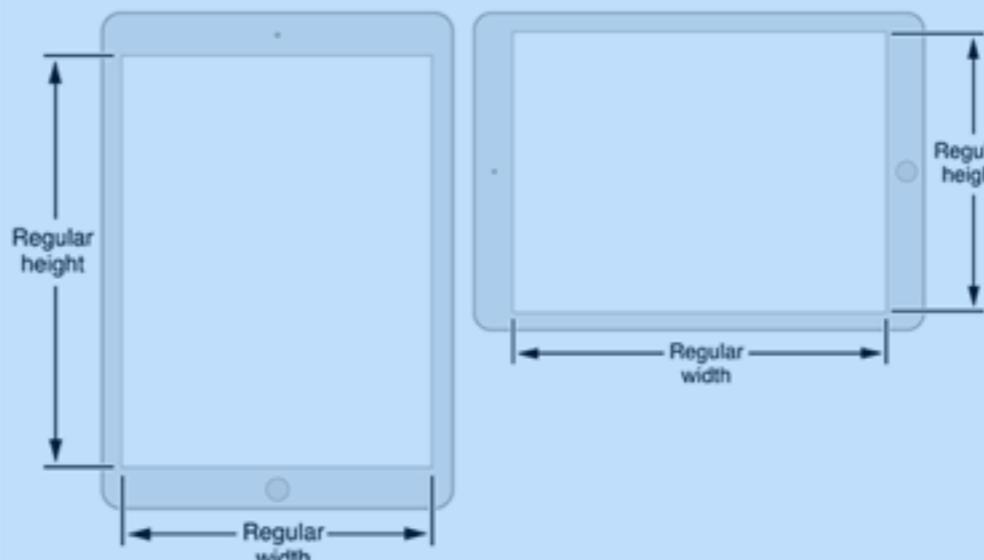
Compact width

Regular height

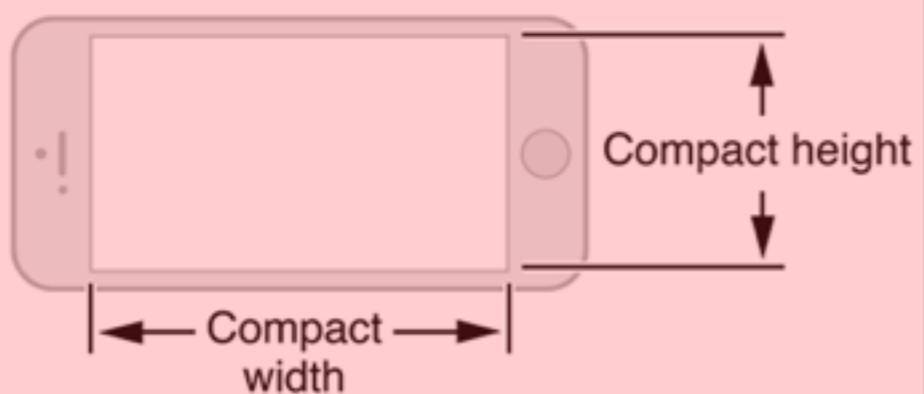


Regular Height
Regular Width

iPad Portrait & Landscape

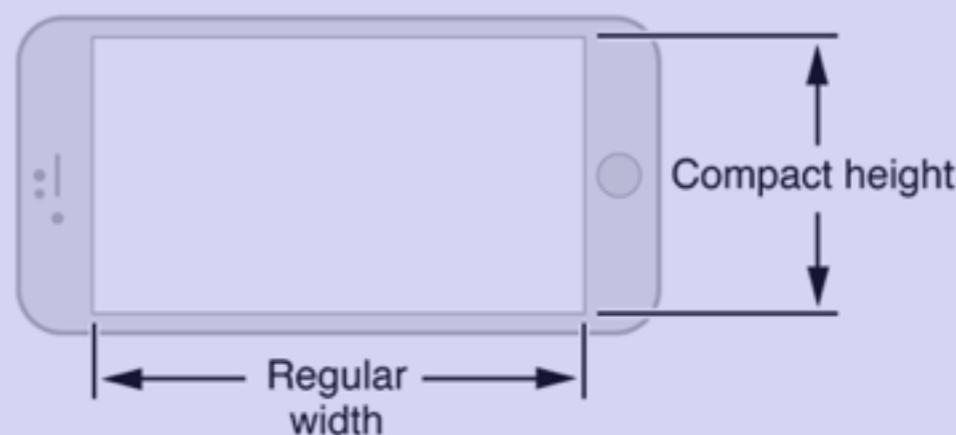


iPhone 4, 5, 6 Landscape



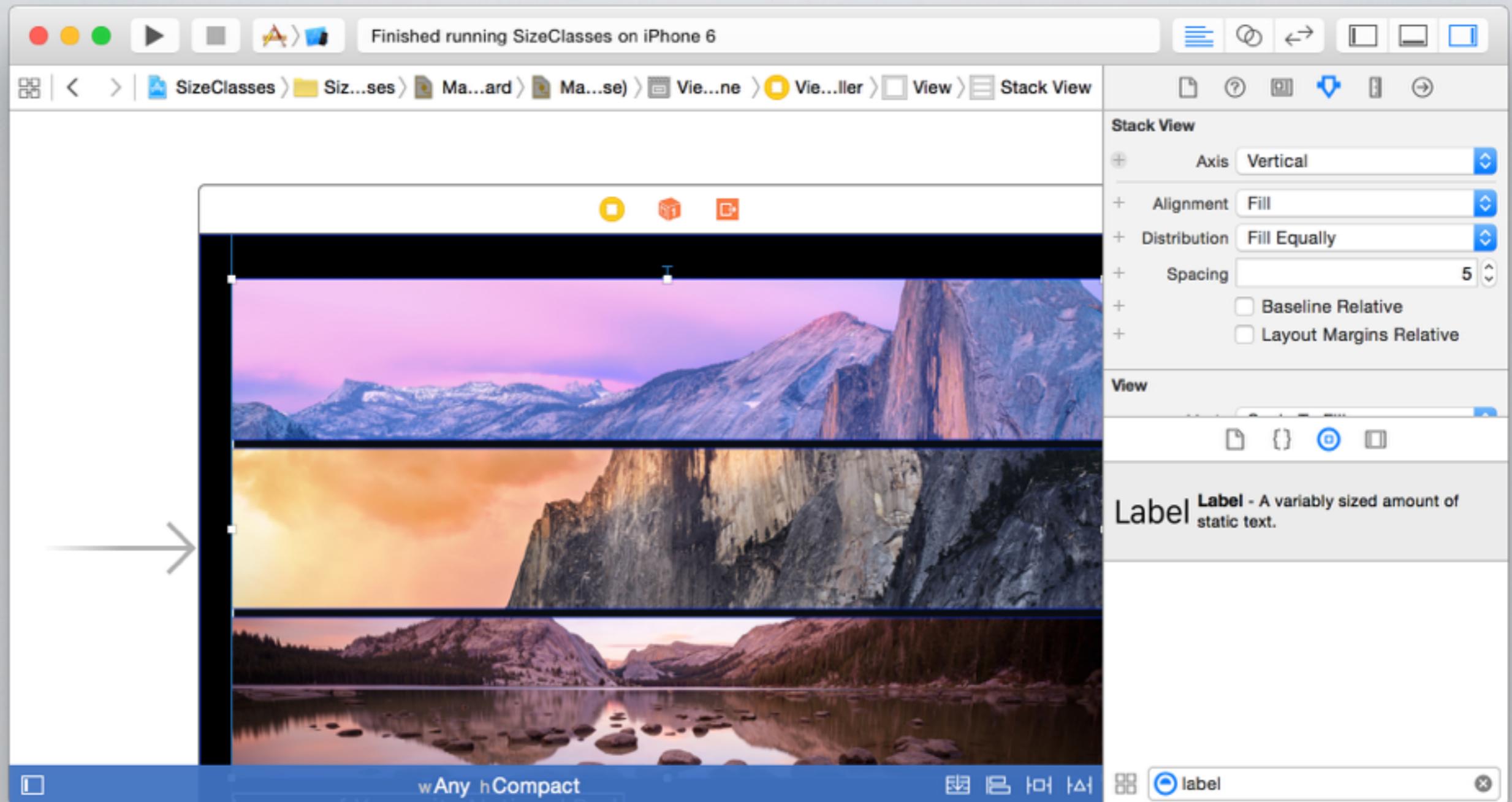
Compact Height
Compact Width

iPhone 6 Plus Landscape

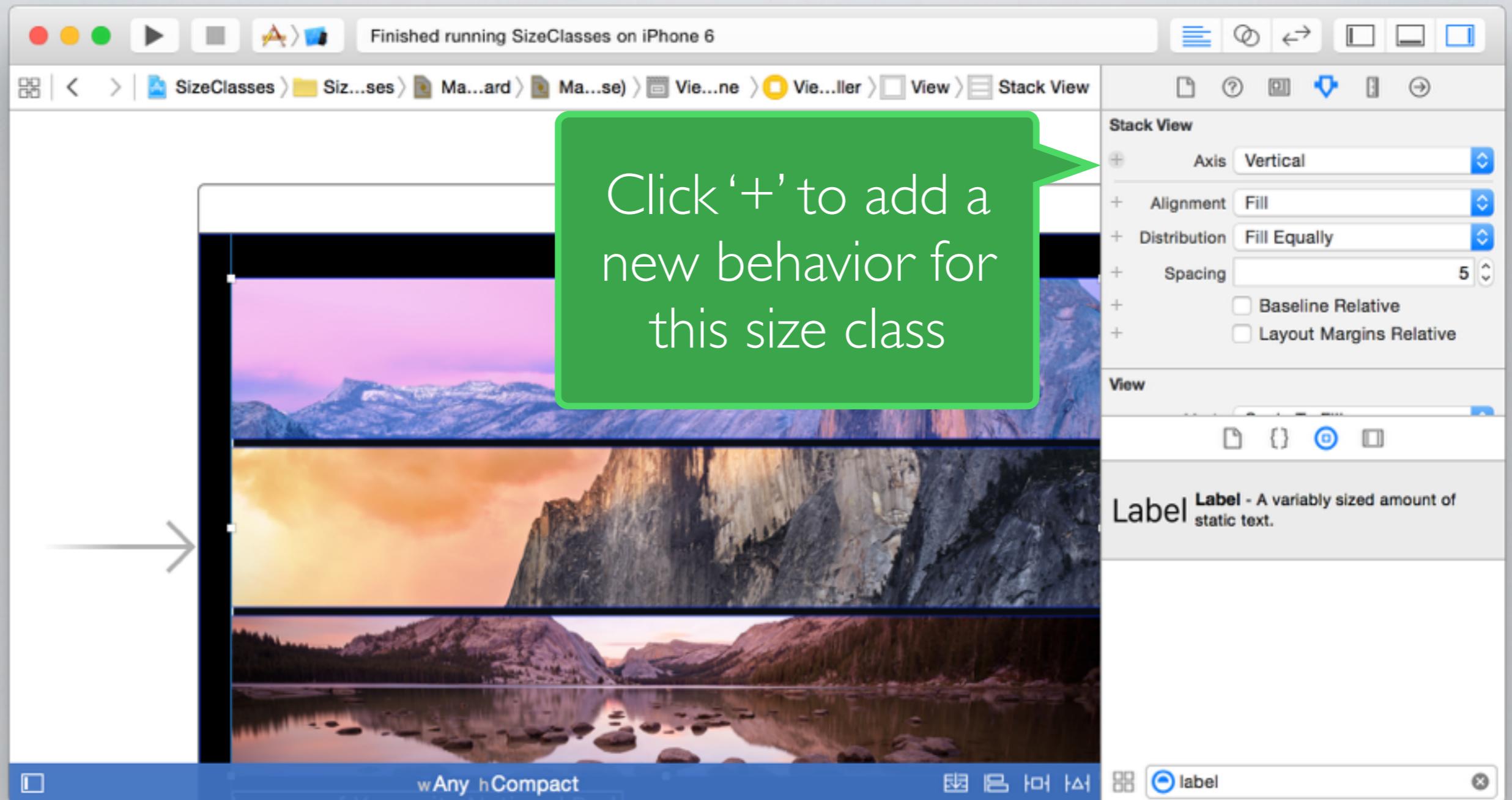


Compact Height
Regular Width

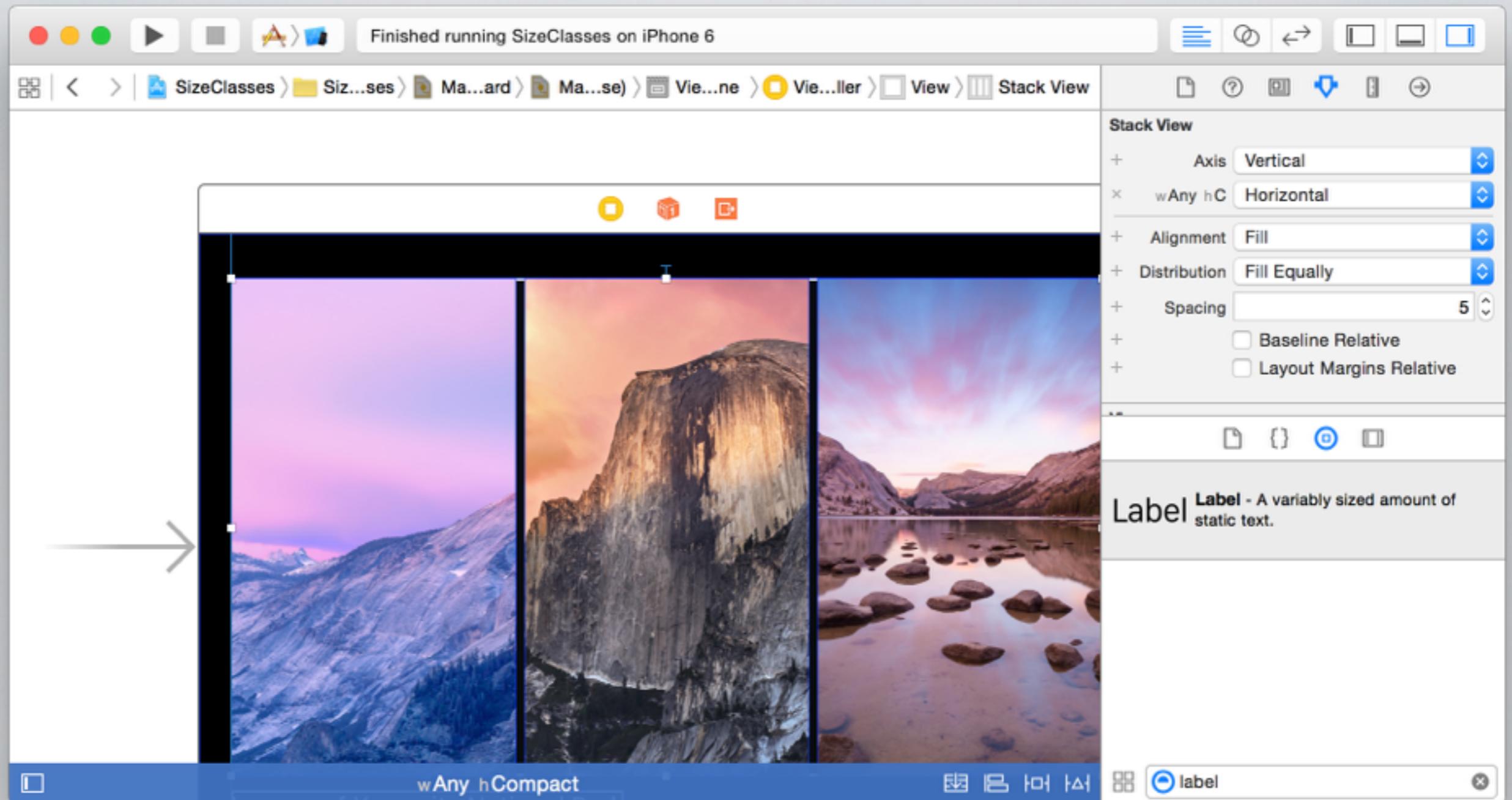
SIZE CLASSES



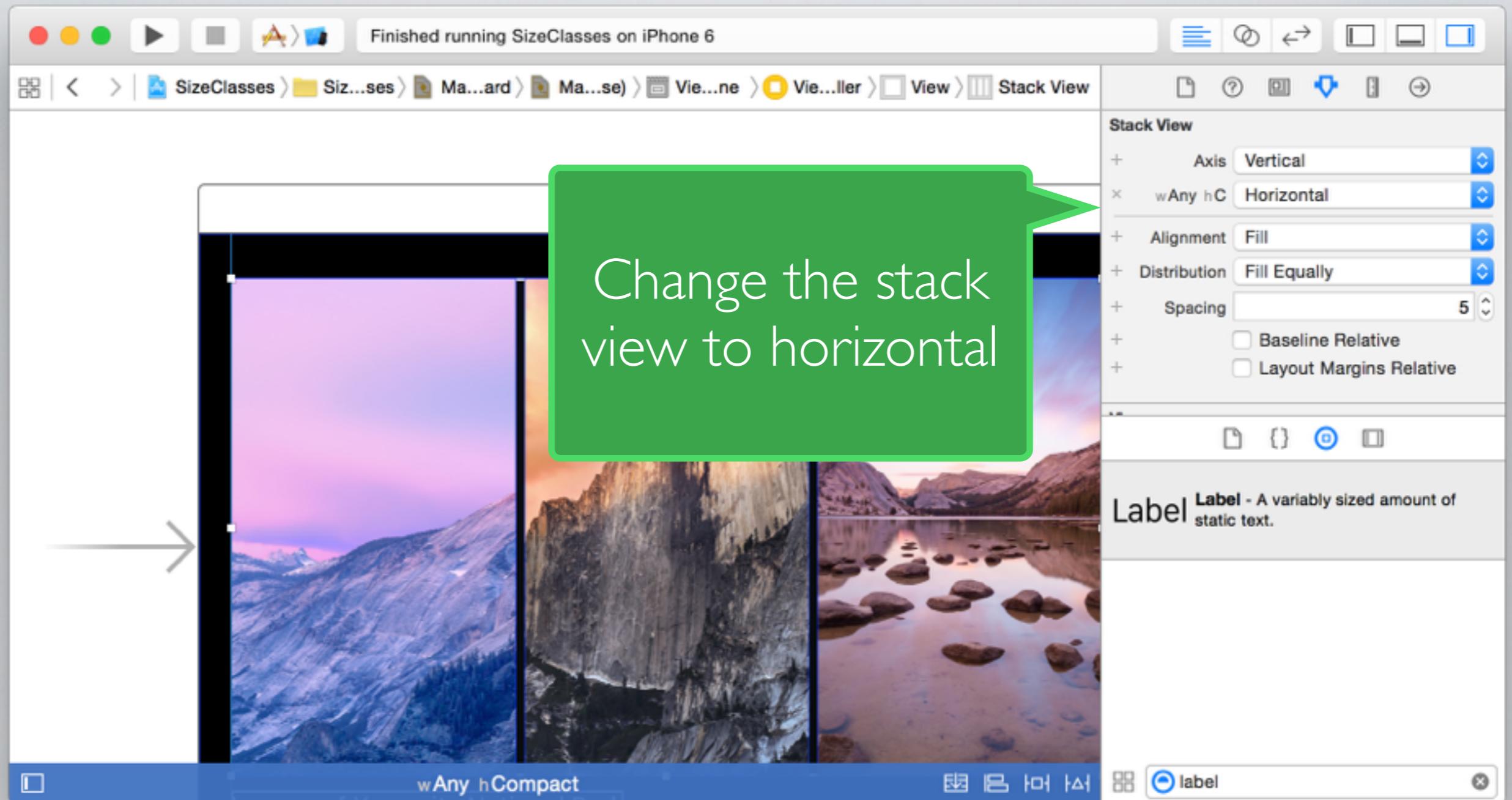
SIZE CLASSES



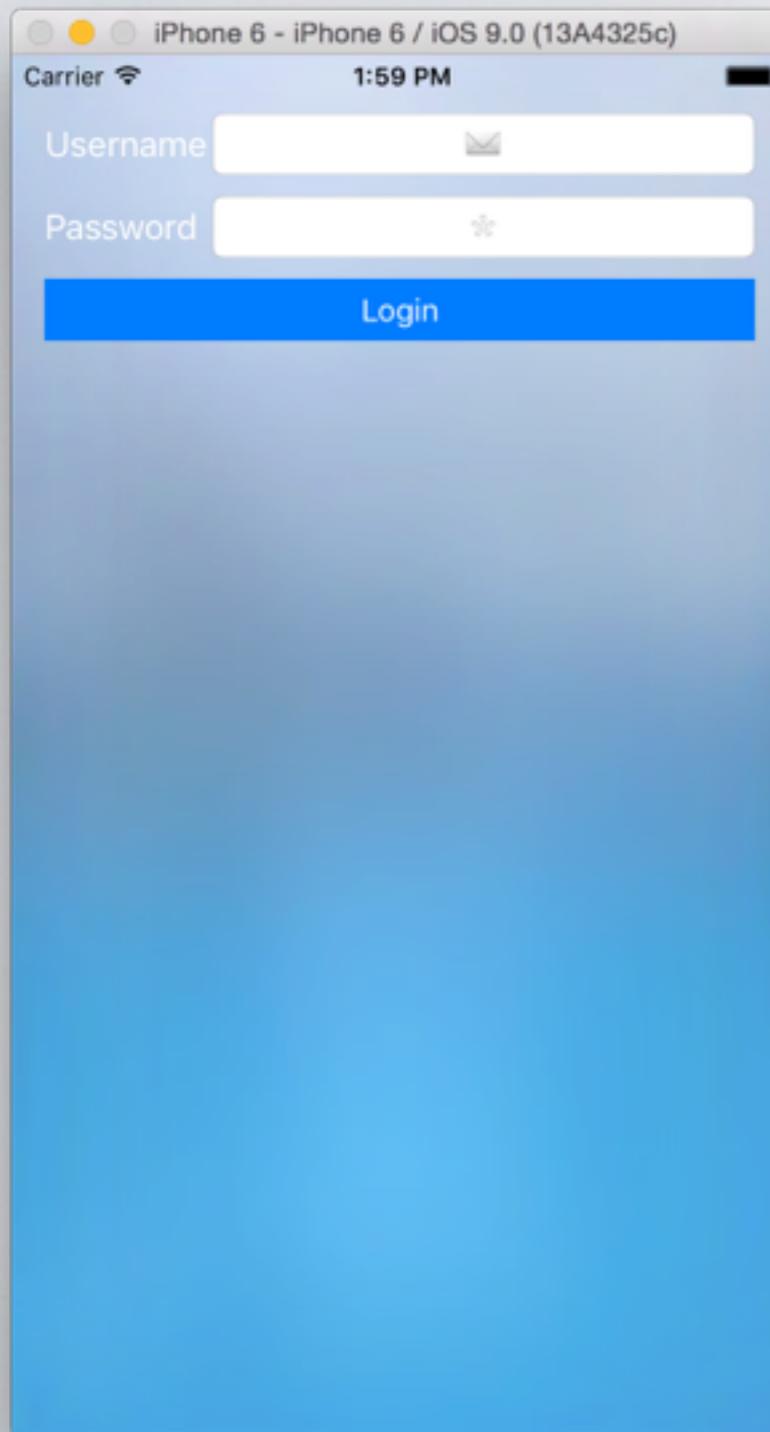
SIZE CLASSES



SIZE CLASSES



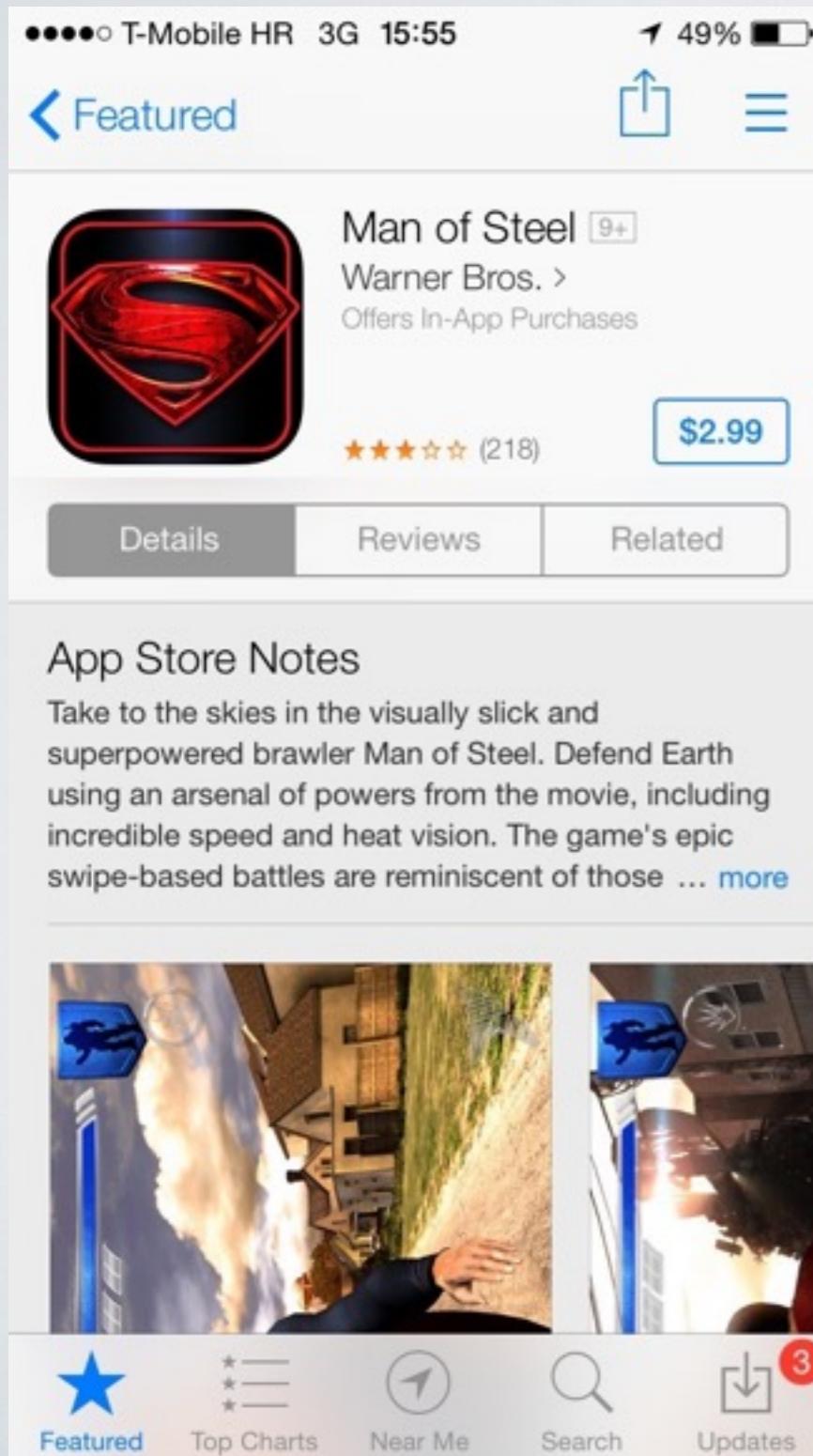
EXERCISE



Try to recreate this

- This does **not** follow iOS guidelines but it's fine as an exercise
- In general take a look at <http://pttrns.com/?did=1&scid=9> for inspiration of signup screens

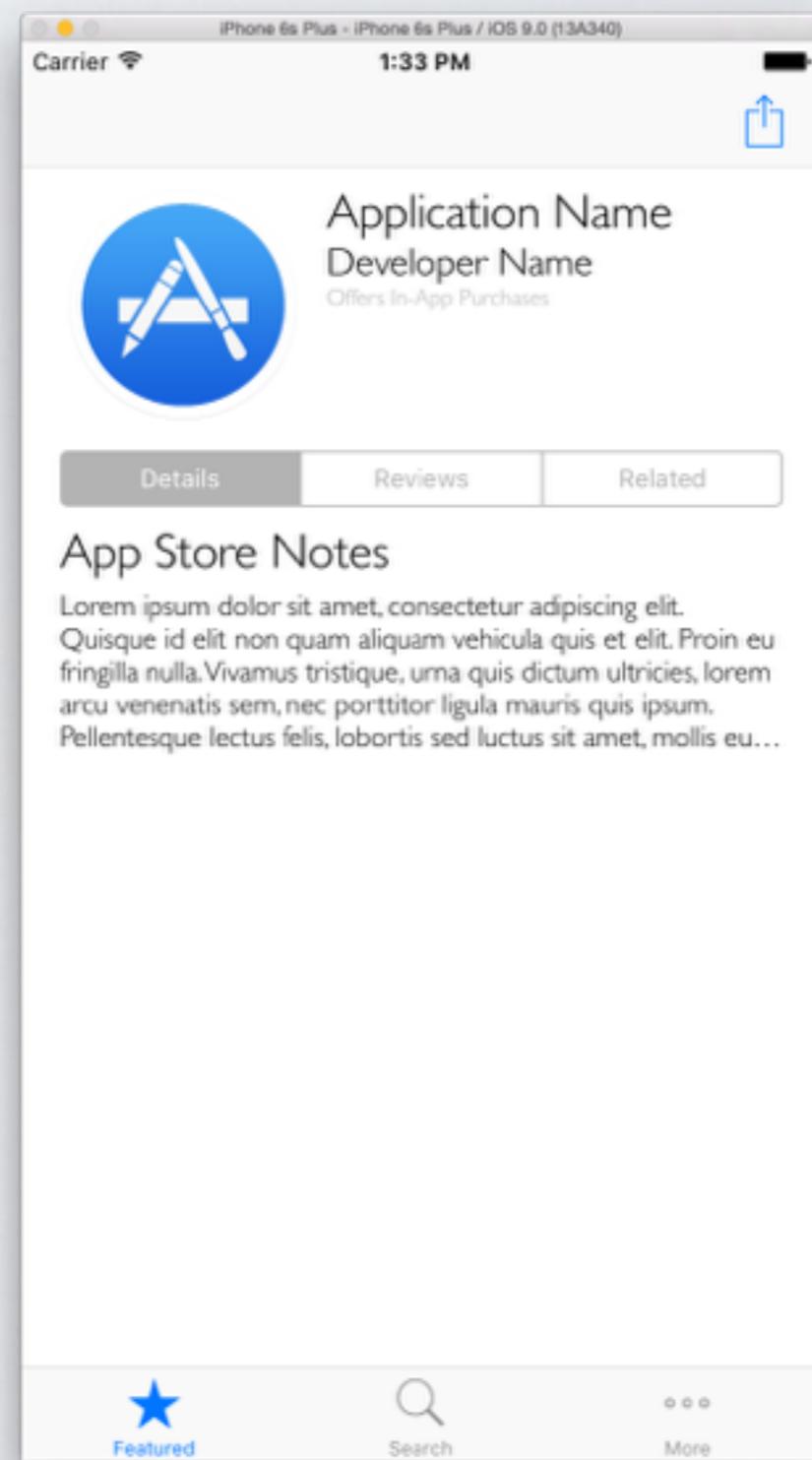
EXERCISE



Try to recreate this

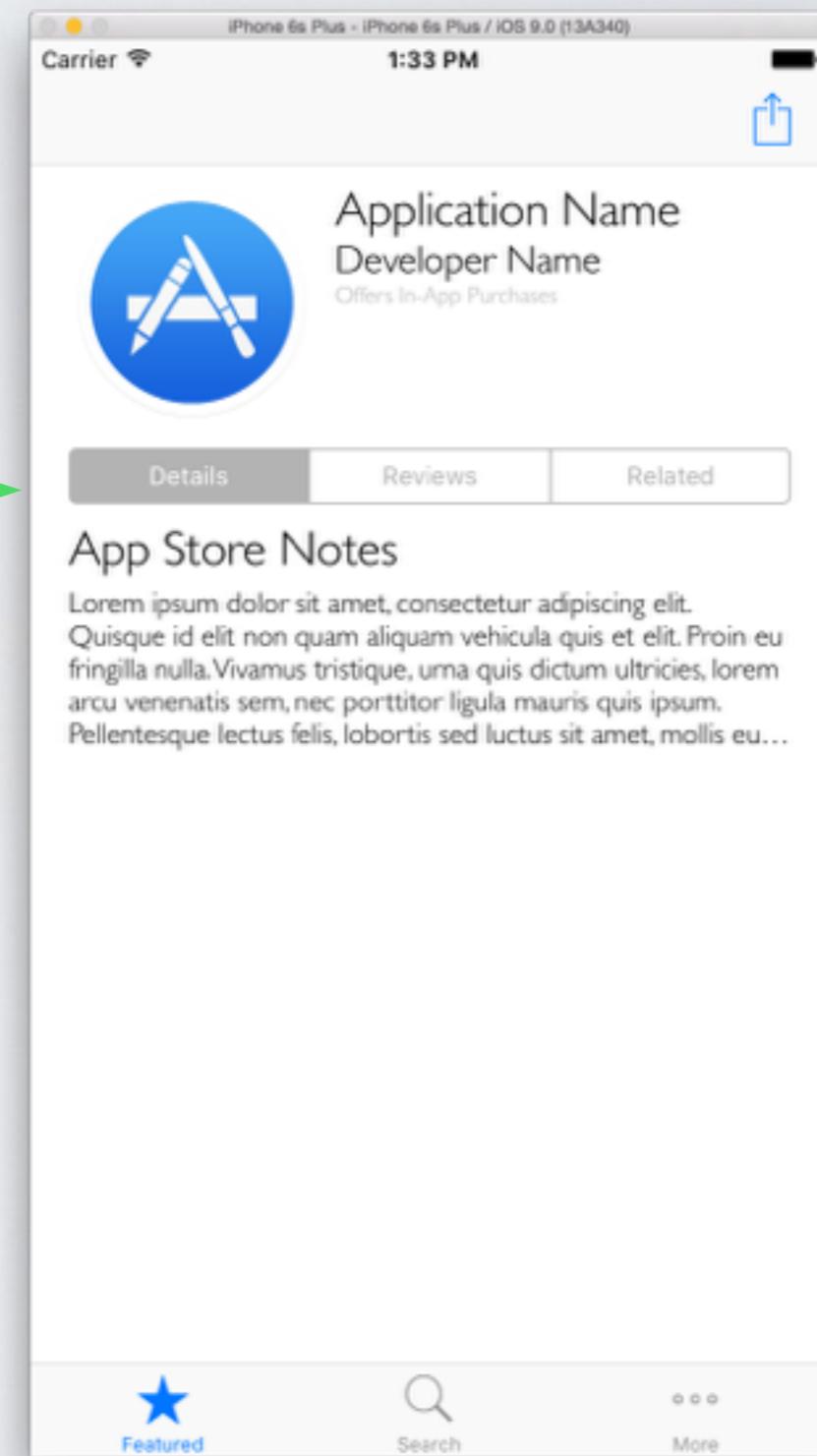
- Don't worry about the top and bottom bar first
- Add a UIImageView for the logo
- Add some text for app name, developer, details

EXERCISE



EXERCISE

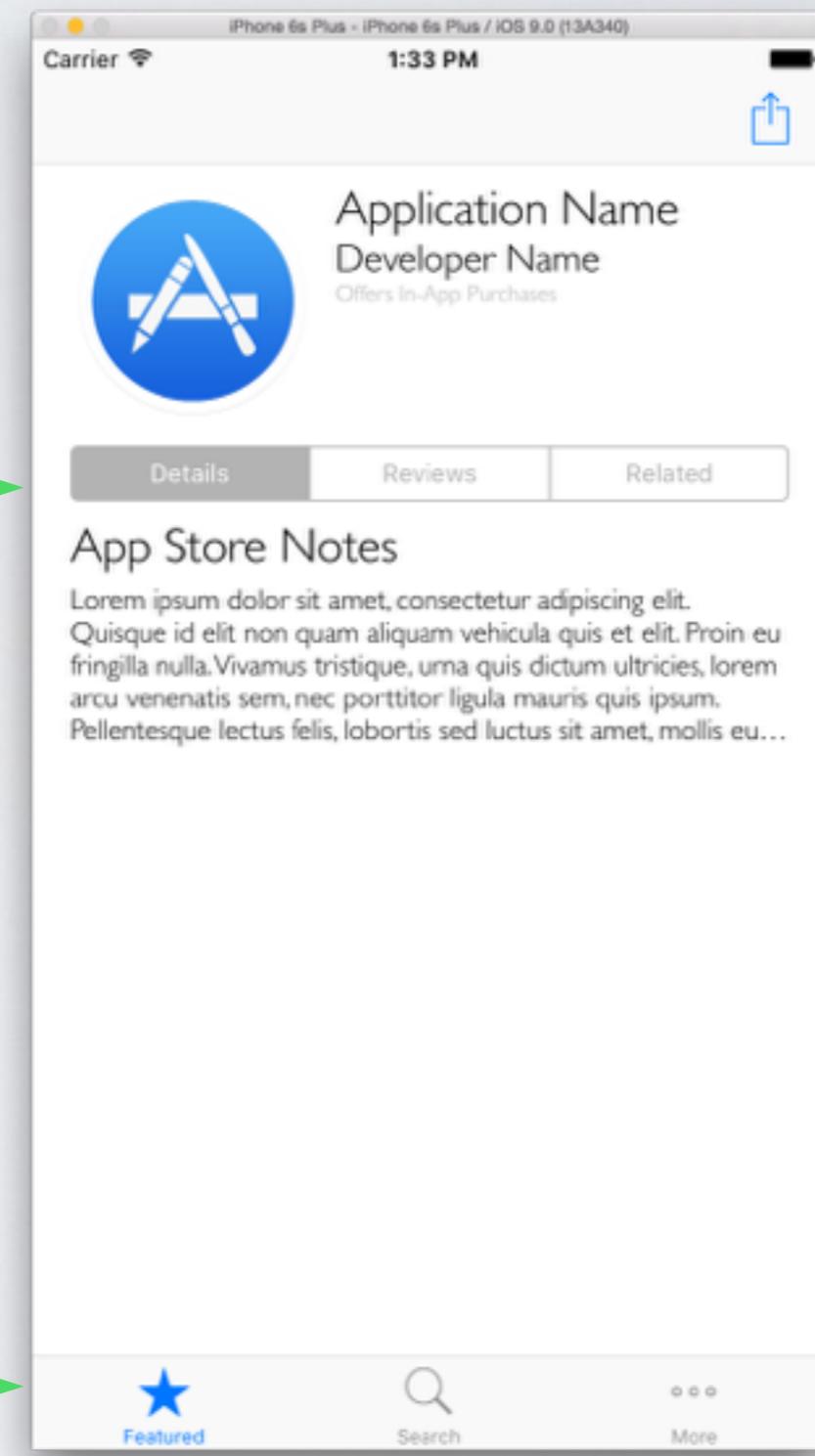
UISegmentedControl



EXERCISE

UISegmentedControl

TabBarController

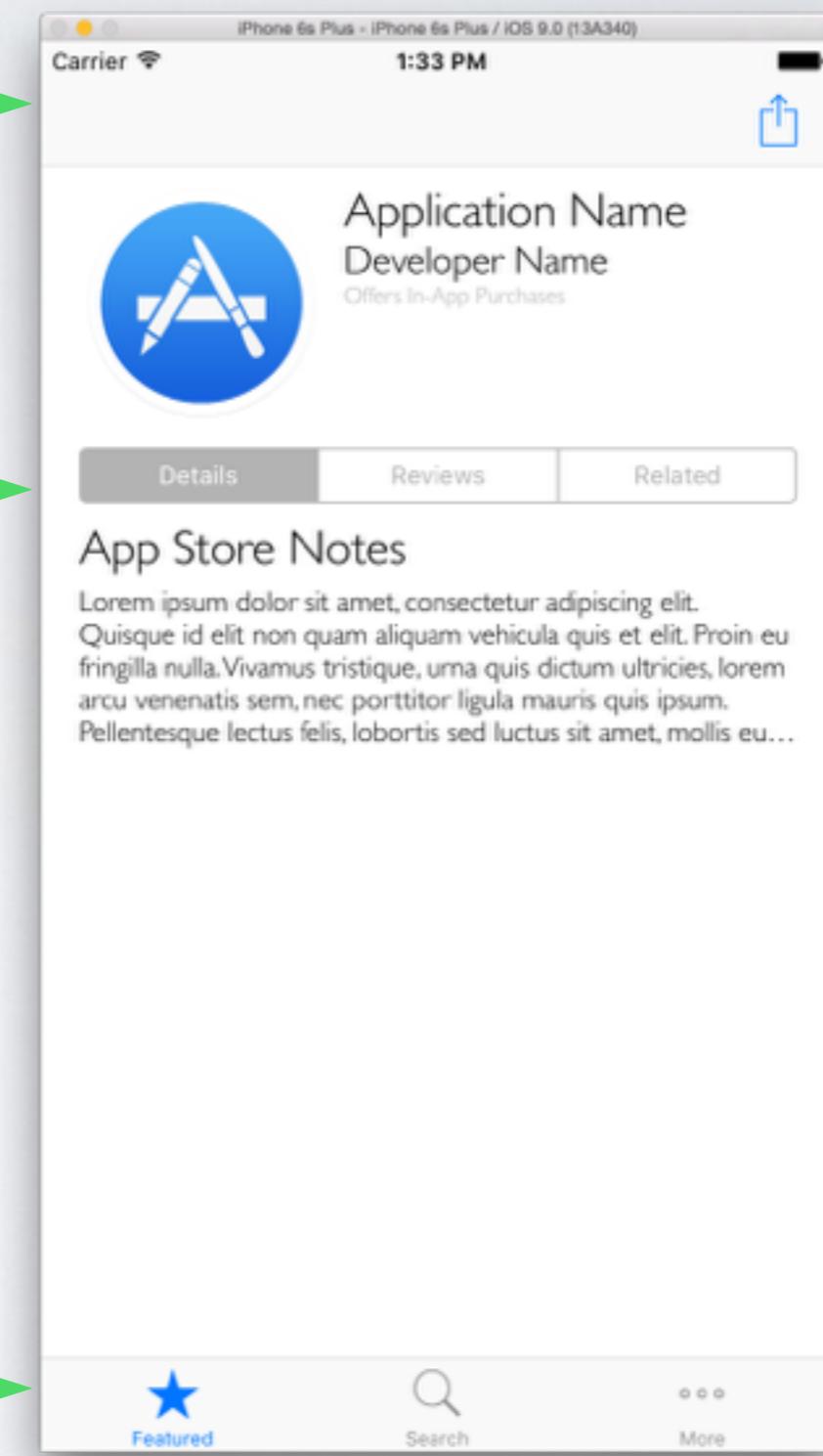


EXERCISE

NavigationController

UISegmentedControl

TabBarController



NEXT TIME

- Editing in a table view and custom table view cells