

iOS
DeCal

lecture 3

AutoLayout

cs198-001 : spring 2018

today's lecture

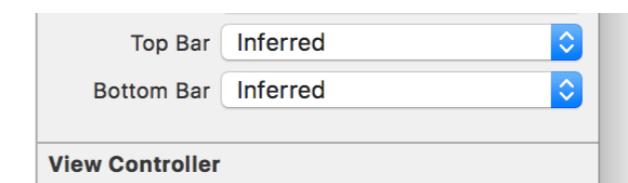
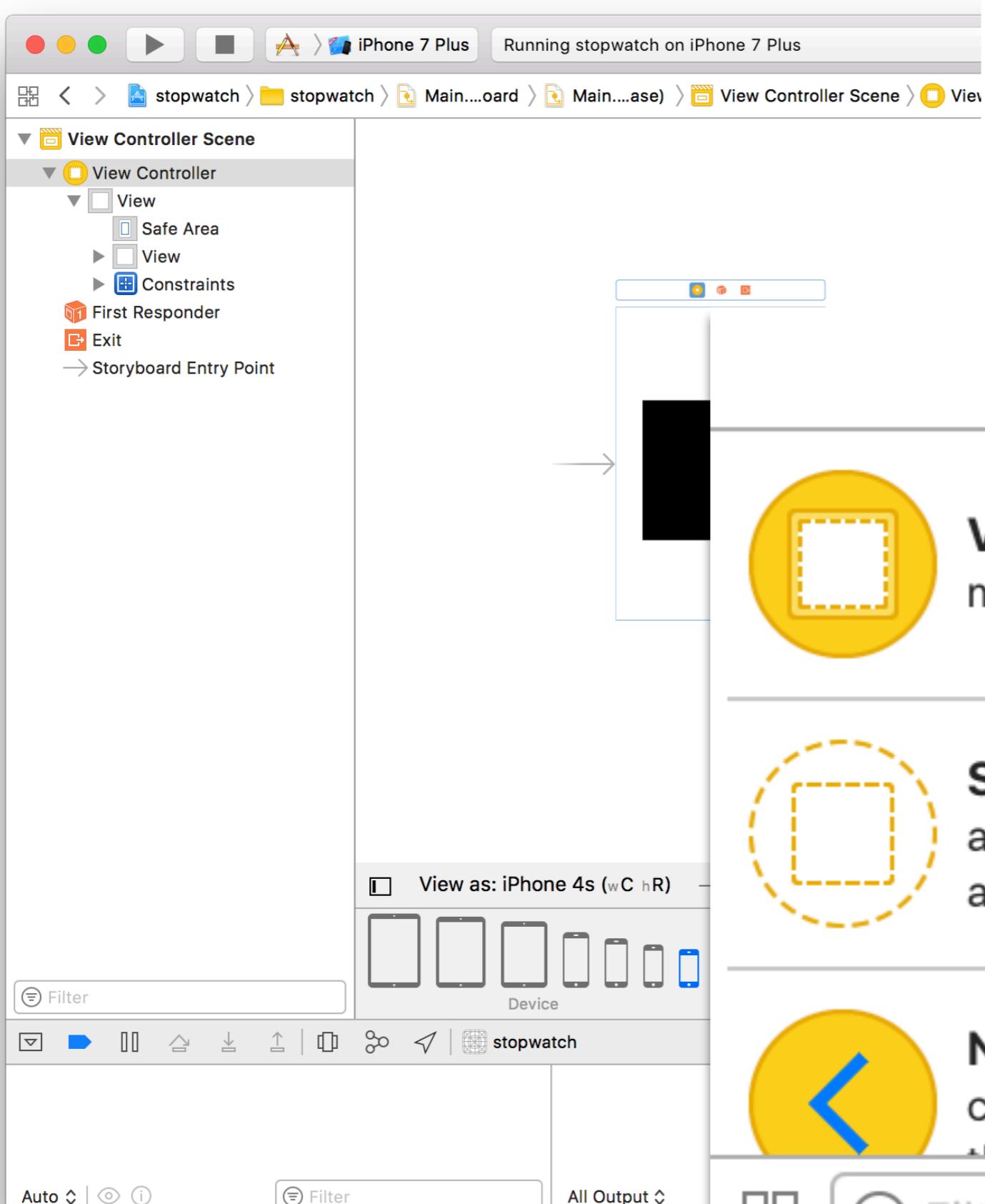
- announcements
- UI toolkit / Object library
- laying out views using Auto layout + Storyboard
- check in

announcements

- hw2 (hangman) released??? tonight.
- due next Monday at 11:59pm
- if you didn't check off for lab last week, you can check off at the beginning of next lab

iOS storyboard objects

object library



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

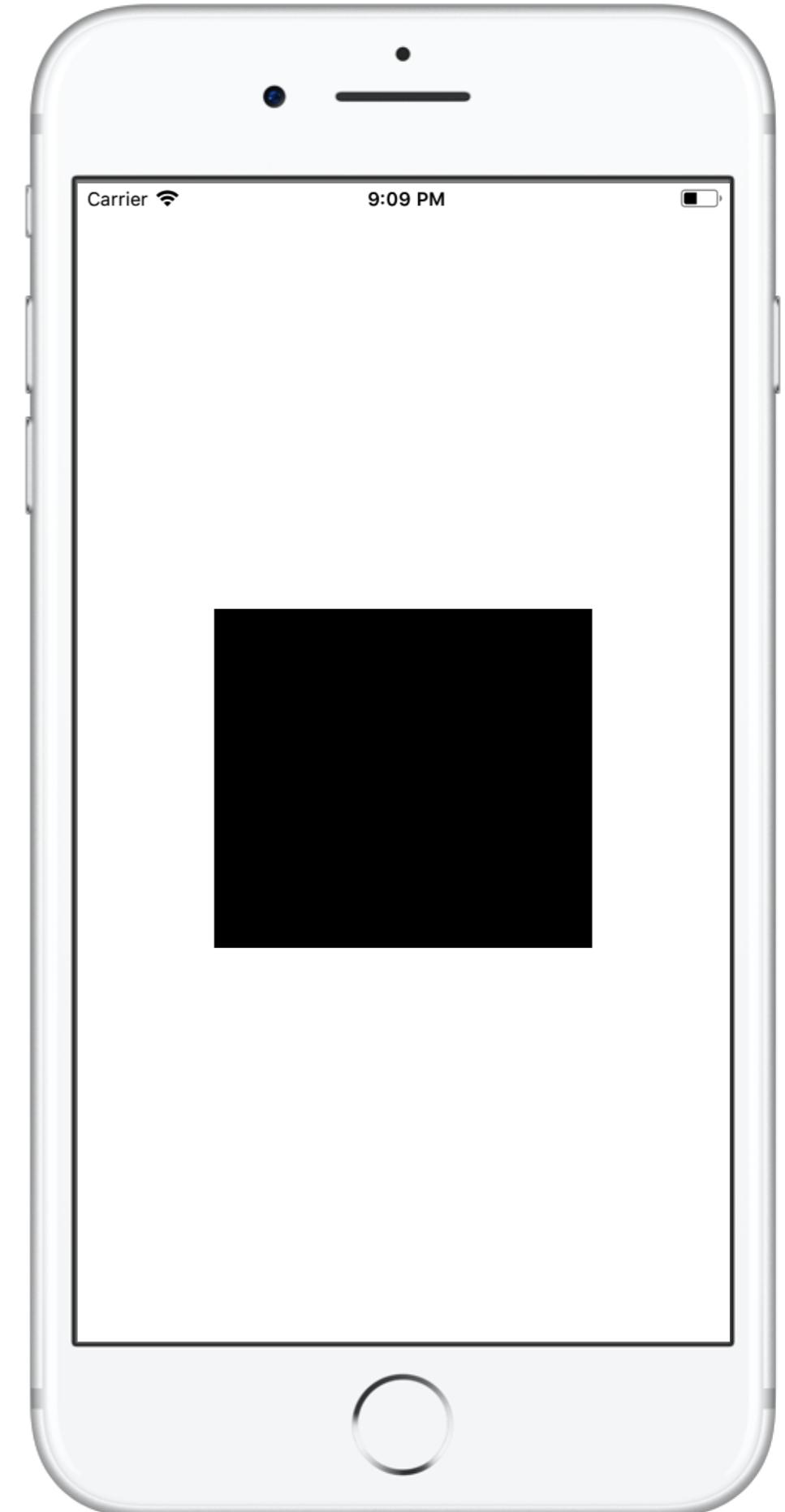


views



View - Represents a rectangular region in which it draws and receives events.

- just a box
- all UI elements subclass **UIView**
- can respond to touches / other gestures

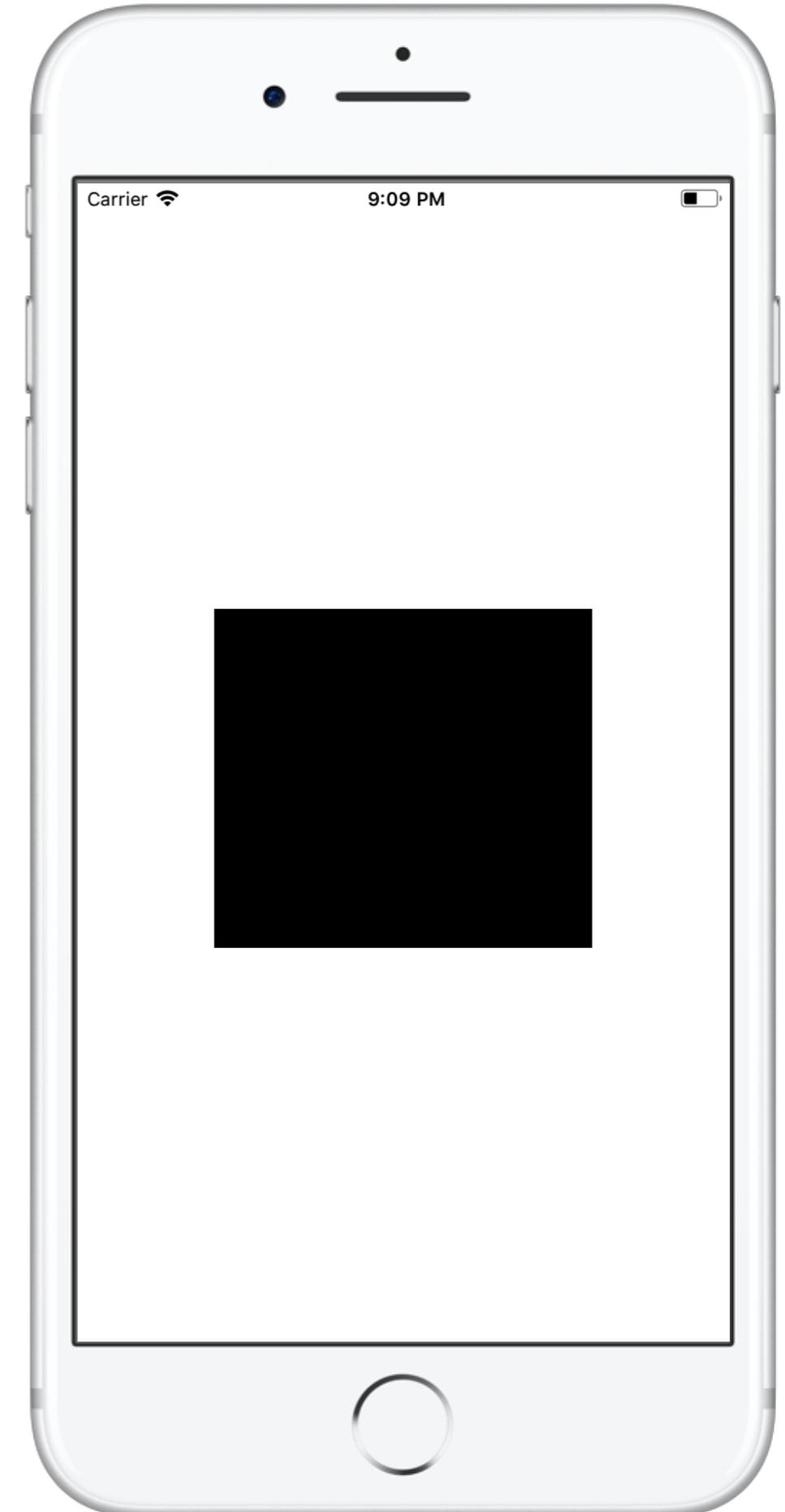


views



View - Represents a rectangular region in which it draws and receives events.

- just a box
- all UI elements subclass **UIView**
- can respond to touches / other gestures
- *how many views are in this image?*

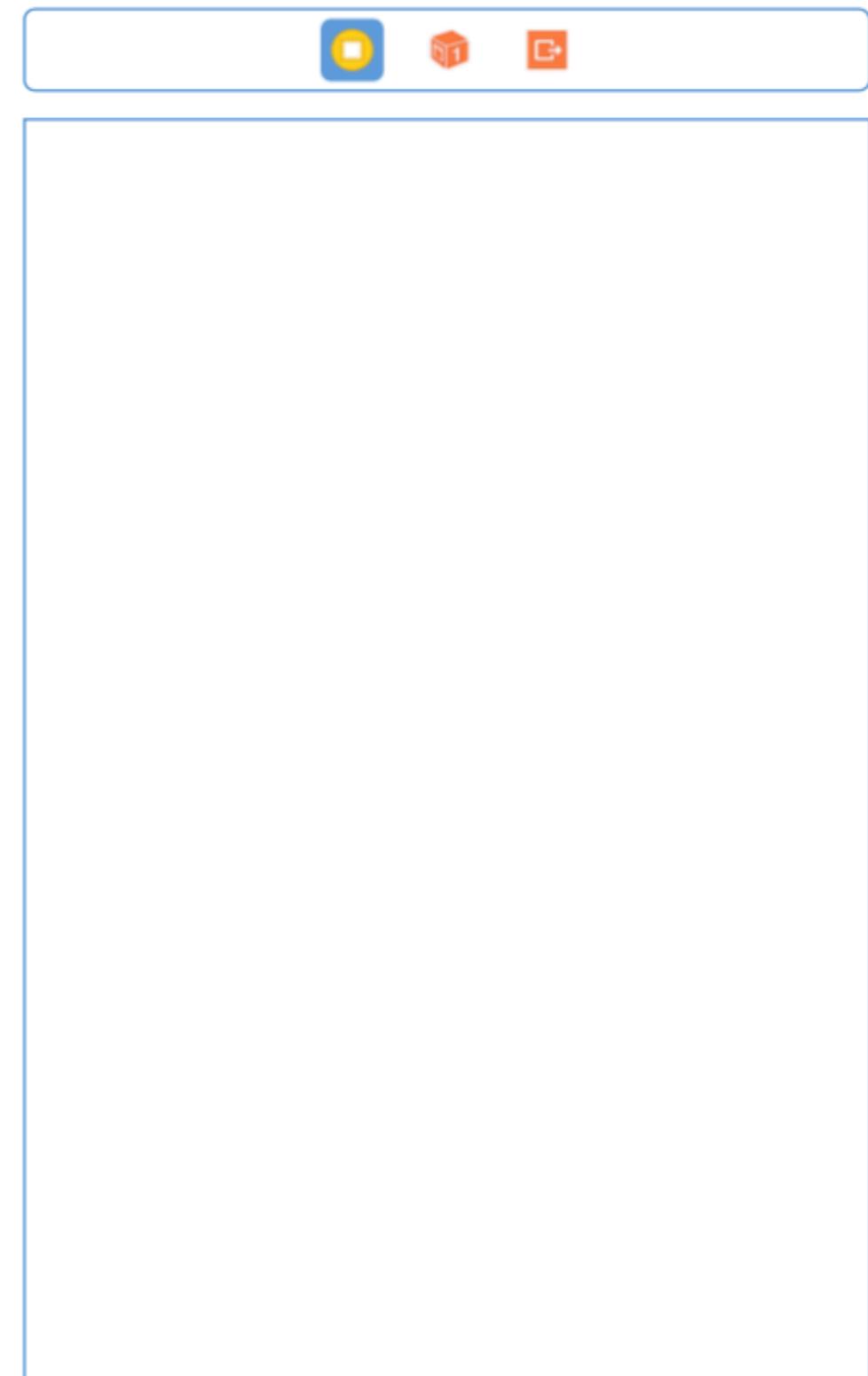


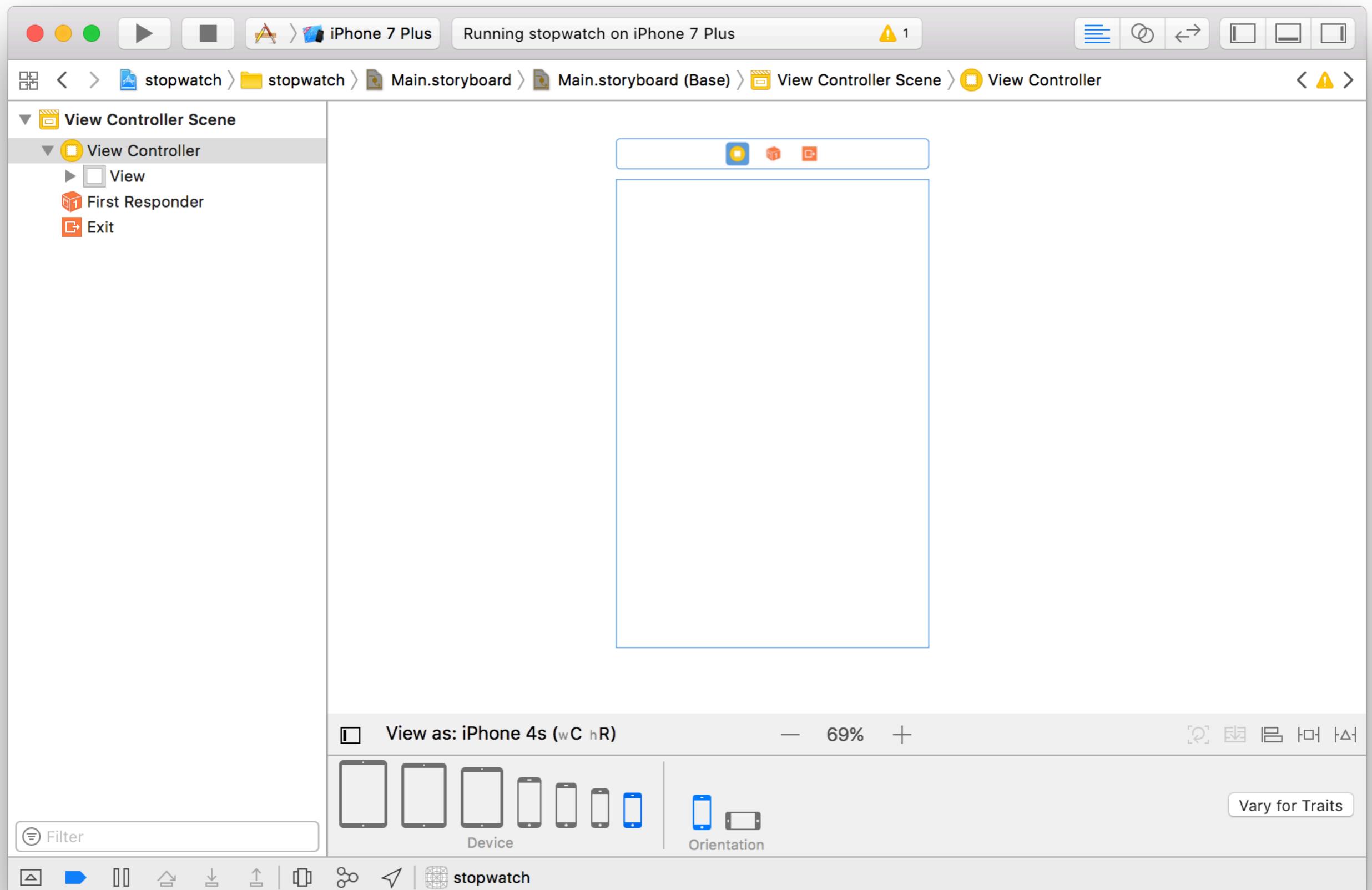
view controllers



View Controller - A controller that manages a view.

- not really a UI object
- adds a new “screen” to your storyboard
- comes with a blank view
- view controller ≠ view





view controller vs. view

labels

Label **Label** - A variably sized amount of static text.

- used to display text
- not editable by user



text fields

Text

Text Field - Displays editable text and sends an action message to a target object when Return is tapped.

- **editable text**
- **limited to one line**
- **use to get user input**



text views



Text View - Displays multiple lines of editable text and sends an action message to a target object when Ret...

- **editable text**
- **multi-line**
- **use to get lengthy user text input**
- **scrollable**



scroll views

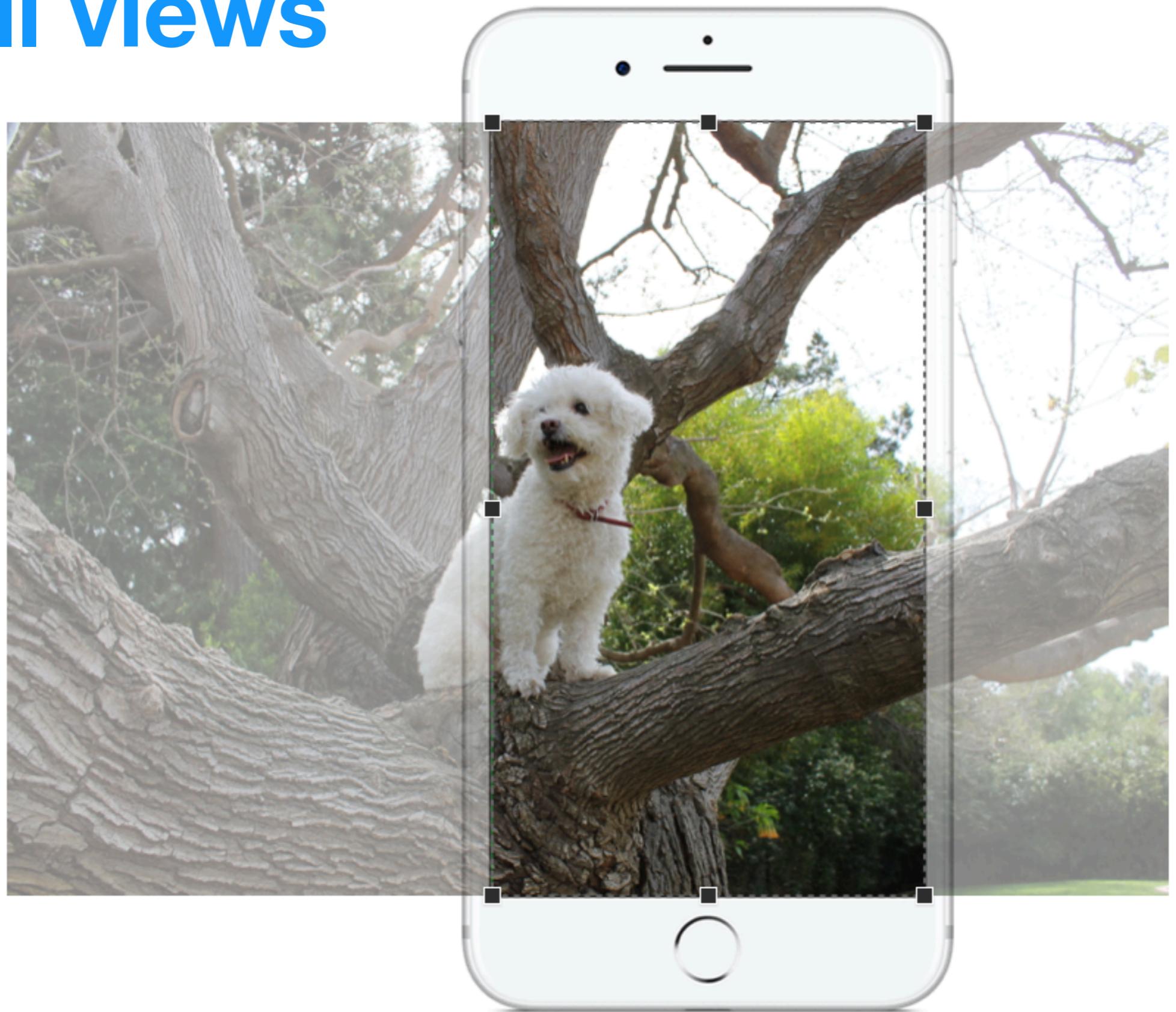


ScrollView - Provides a mechanism to display content that is larger than the size of the application's window.

- accepts user swipes to scroll embedded content
- typically, you'll use a subclass of UIScrollView
- examples: UITextView, UICollectionView, UITableView



scroll views



buttons

Button - Intercepts touch events and sends an action message to a target object when it's tapped.

- use to detect touch events
- useless unless you connect to an IBAction or register a selector

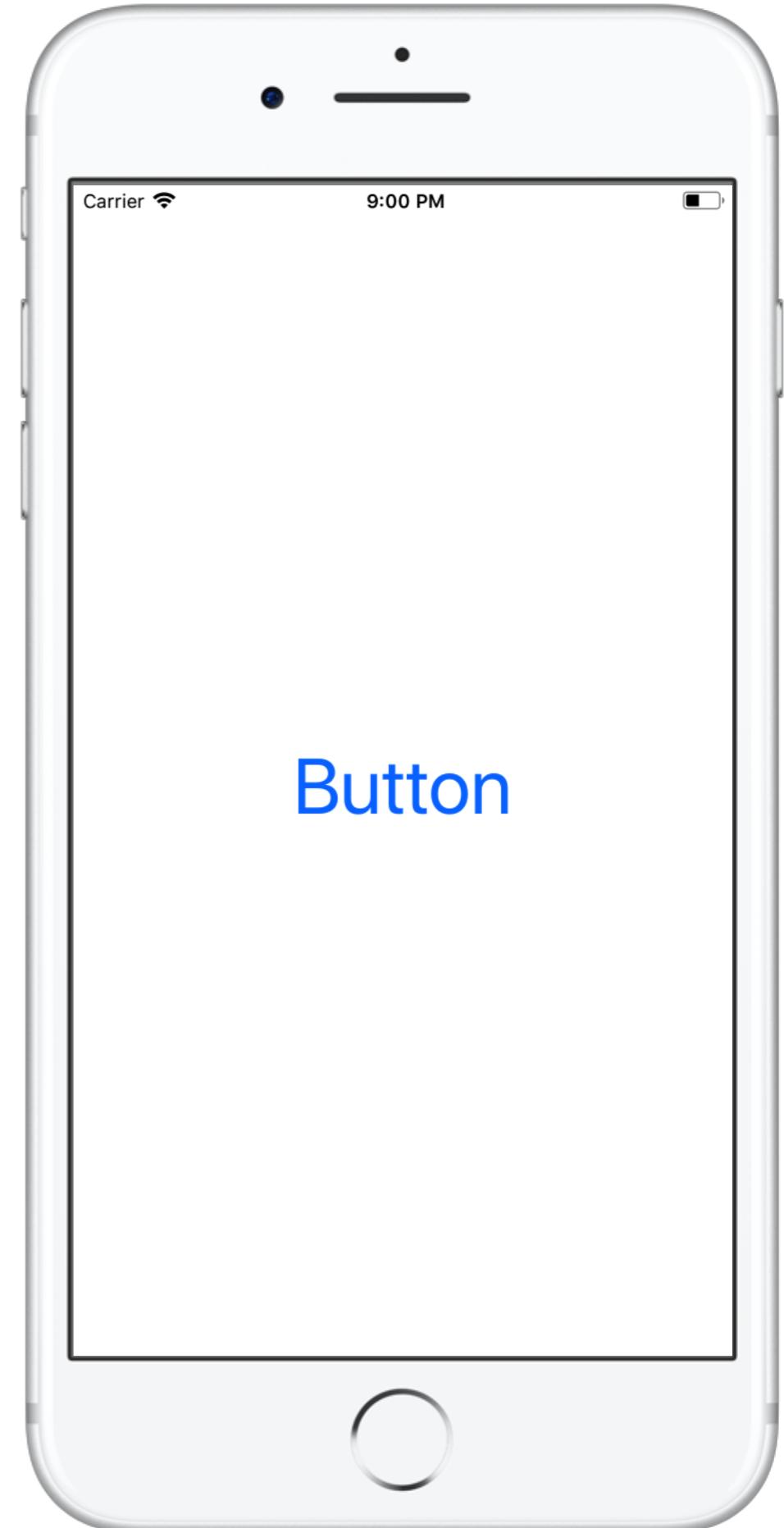
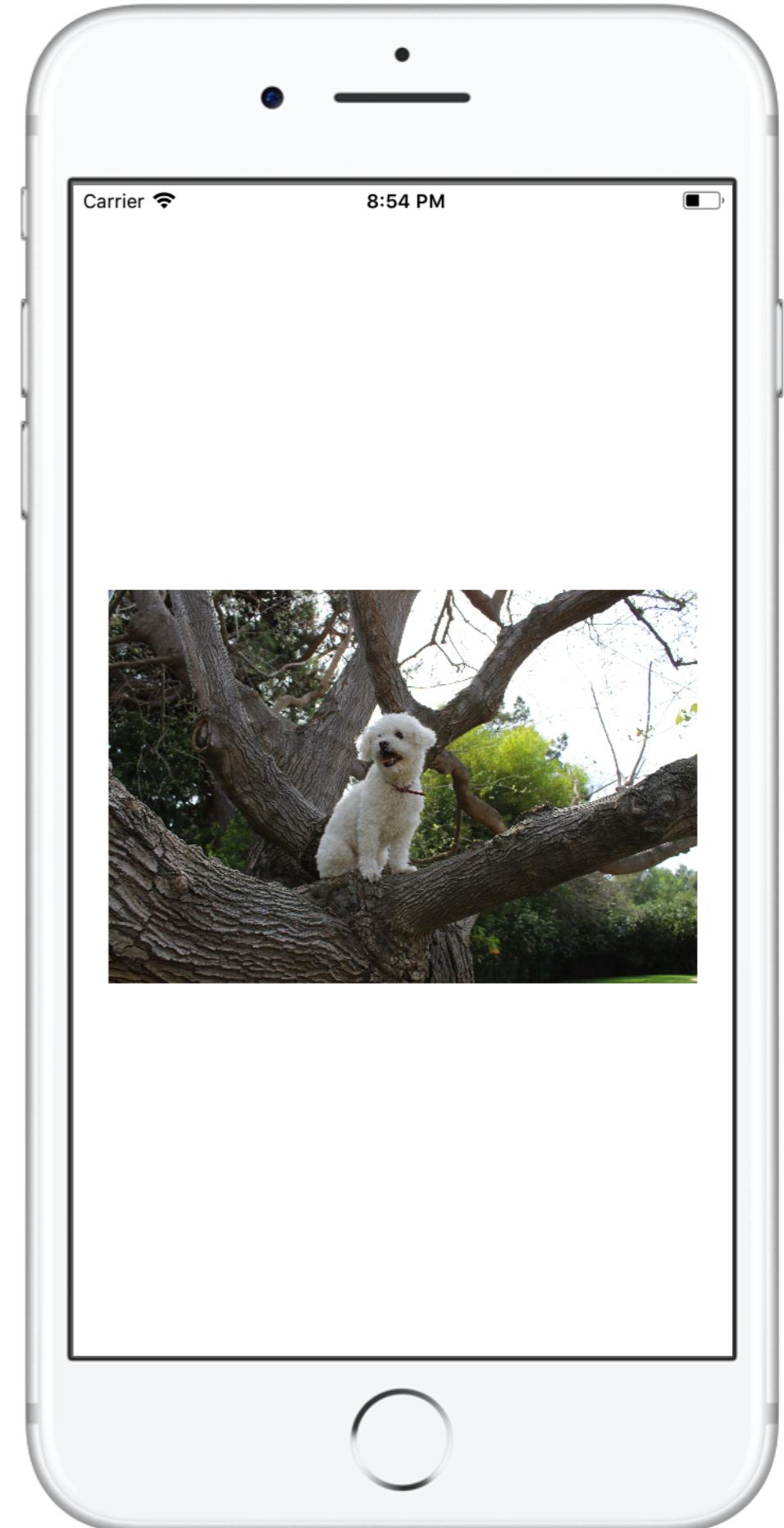


image views



Image View - Displays a single image, or an animation described by an array of images.

- used to display images
- image view ≠ image
- think of it like a picture frame



Storyboard references



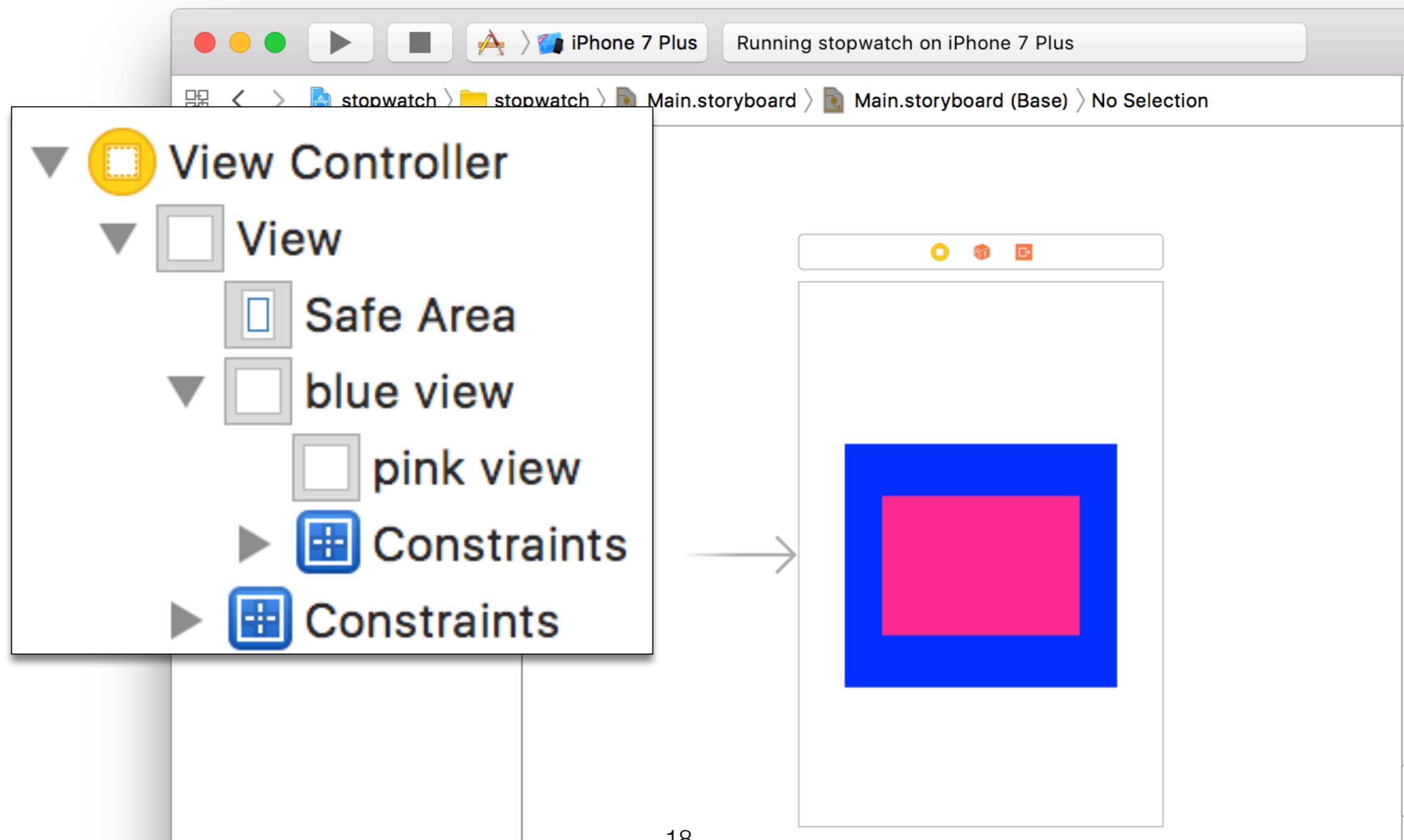
Storyboard Reference - Provides a placeholder for a view controller in an external storyboard.

- more than one storyboard file? no problem!
- *these will make more sense when we cover segues (next lecture)*



Storyboard Reference

superview / subview



stack views

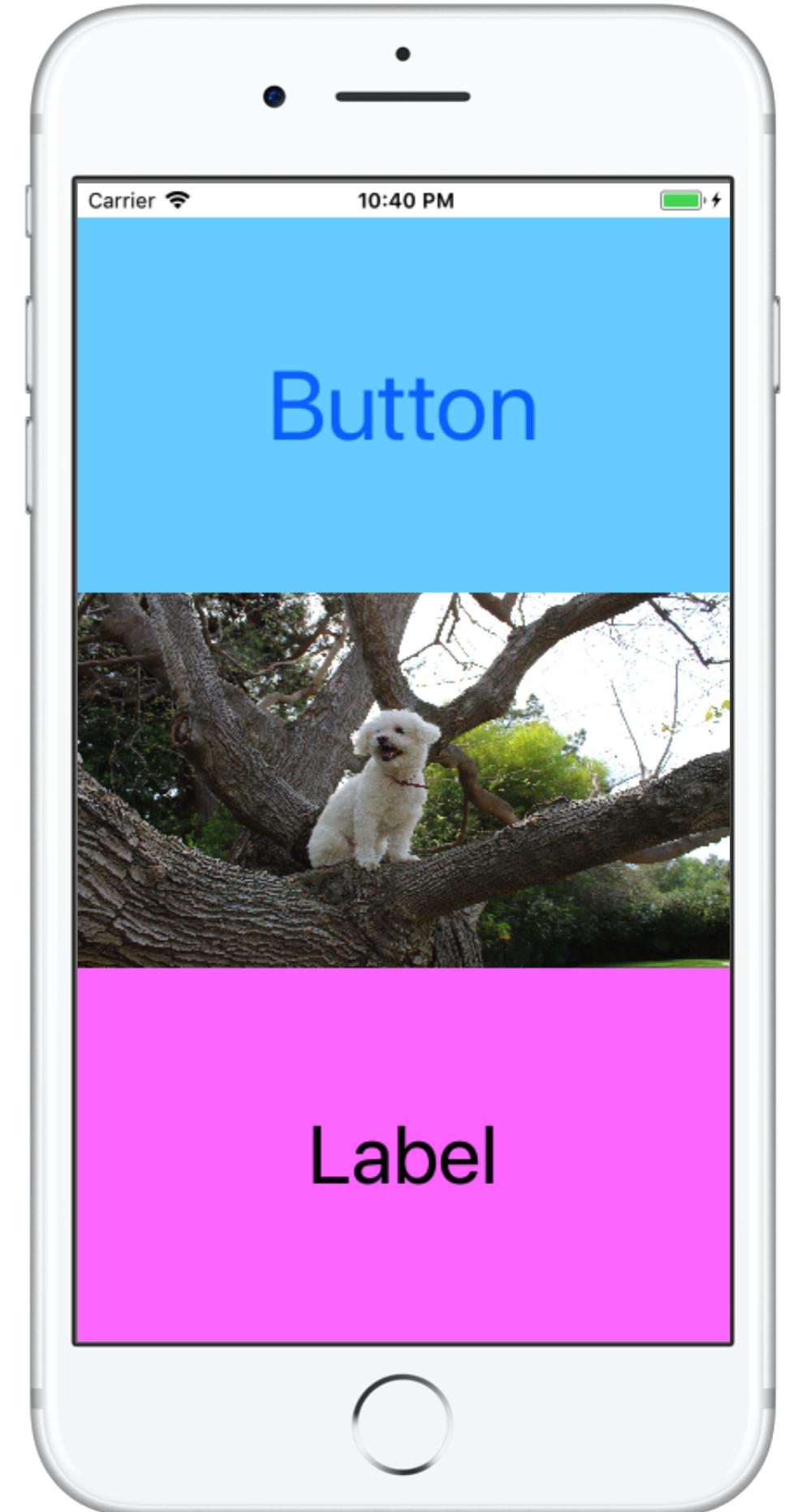


Horizontal Stack View - Arranges views linearly.



Vertical Stack View - Arranges views linearly.

- used to arrange other subviews (static)
- AutoLayout's best friend



stack views



Horizontal Stack View - Arranges views linearly.



Vertical Stack View - Arranges views linearly.

- used to arrange other subviews (static)
- AutoLayout's best friend
- nested stackviews

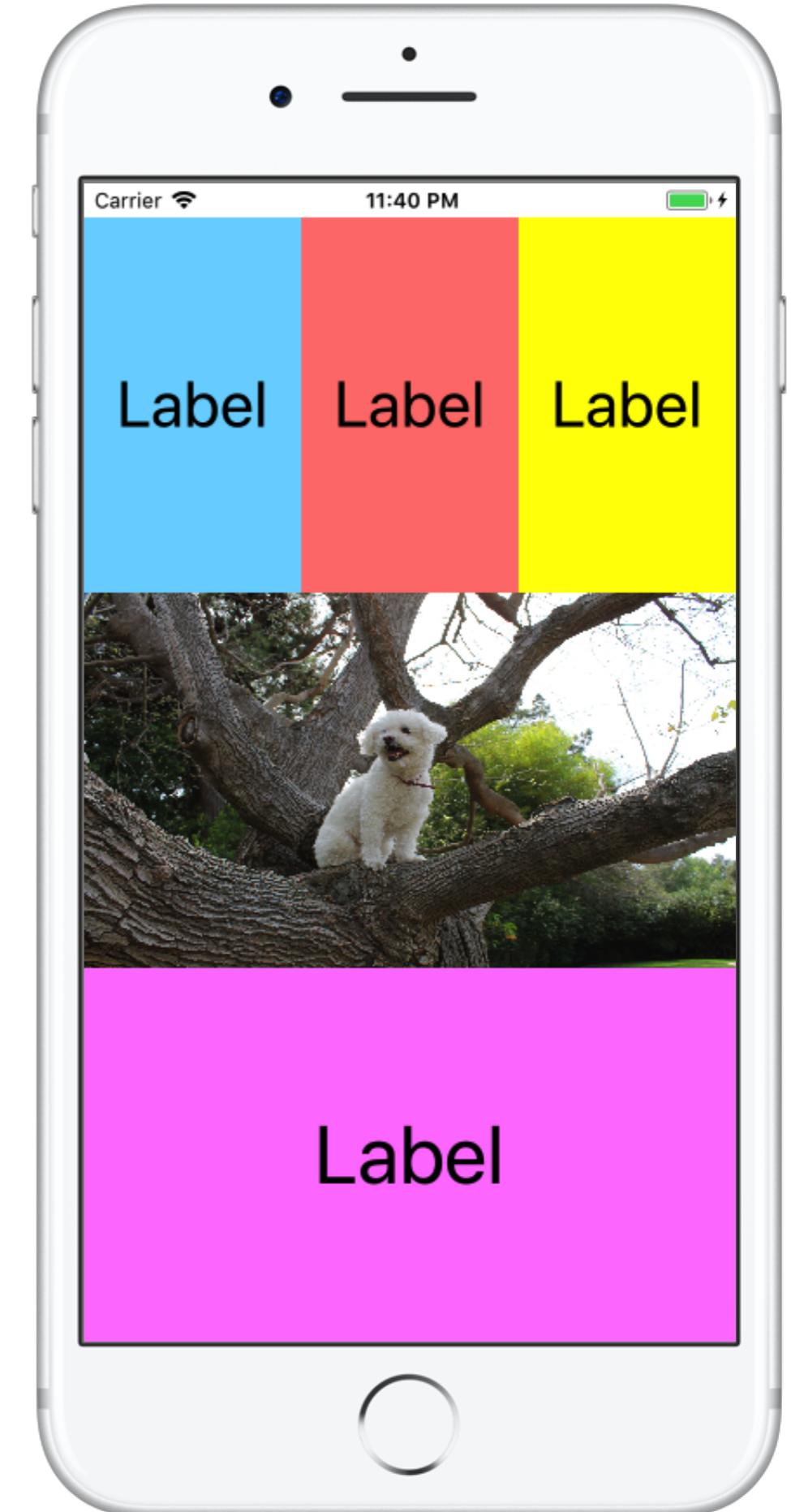


table views

(future lecture)



Table View Controller - A controller that manages a table view.

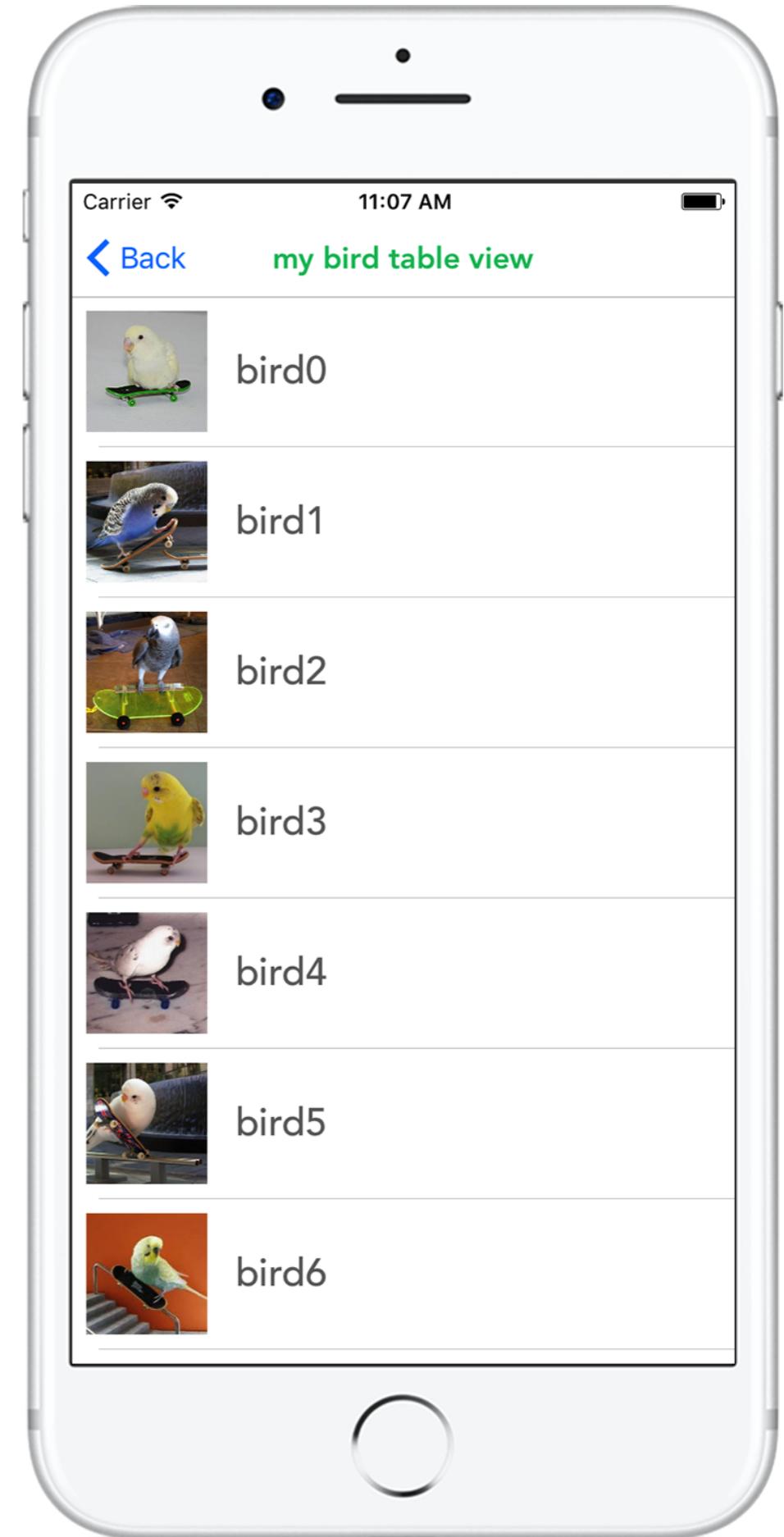


Table View - Displays data in a list of plain, sectioned, or grouped rows.



Table View Cell - Defines the attributes and behavior of cells (rows) in a table view.

- displays cells of data
- scrollable, static or dynamic, view reuse



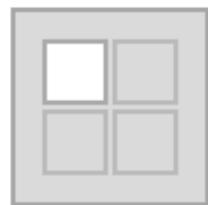
collection views (future lecture)



Collection View Controller - A controller that manages a collection view.

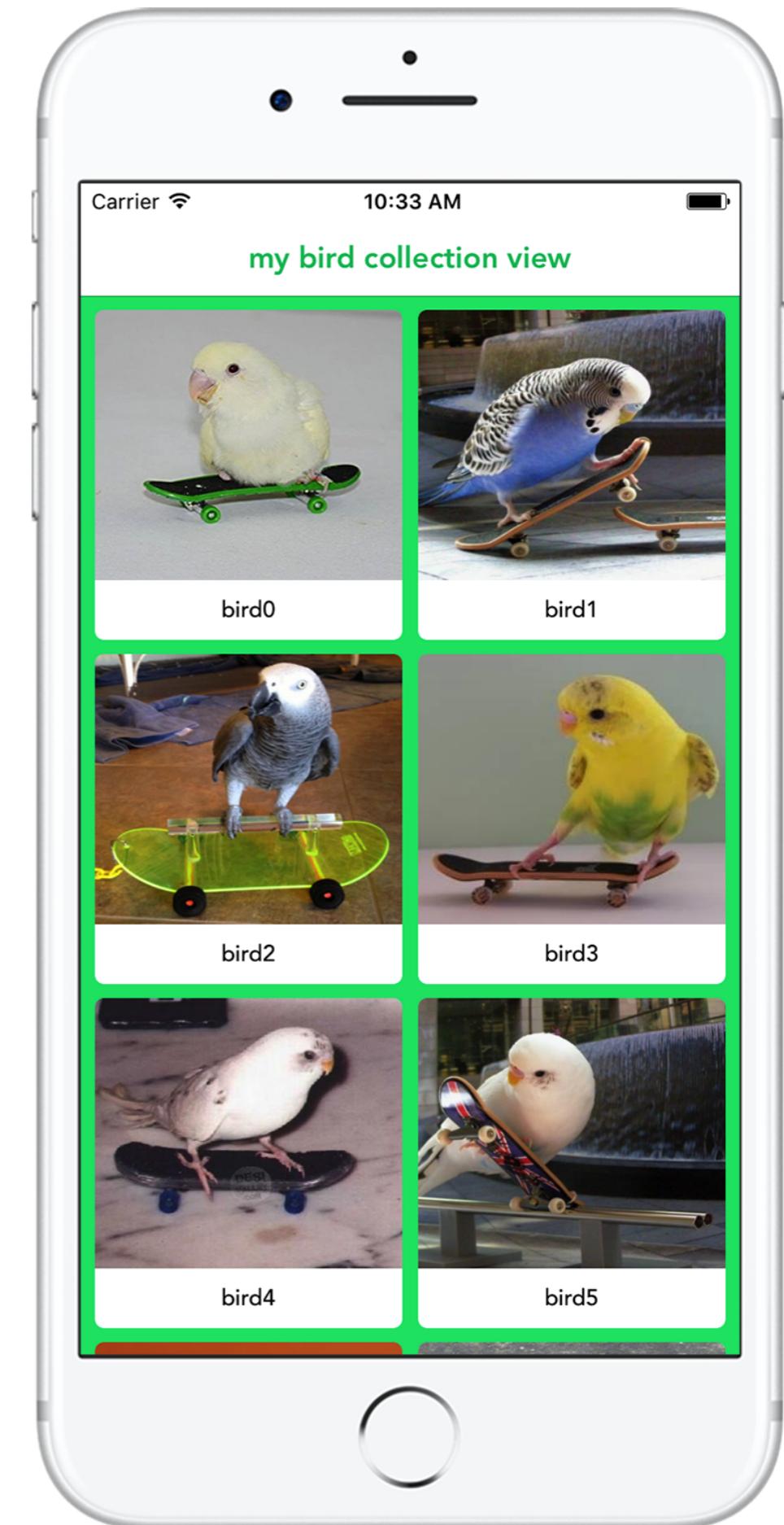


Collection View - Displays data in a collection of cells.



Collection View Cell - Defines the attributes and behavior of cells in a collection view.

- like table views, but displays data in a grid



**laying out your user
interface**

creating your user interface

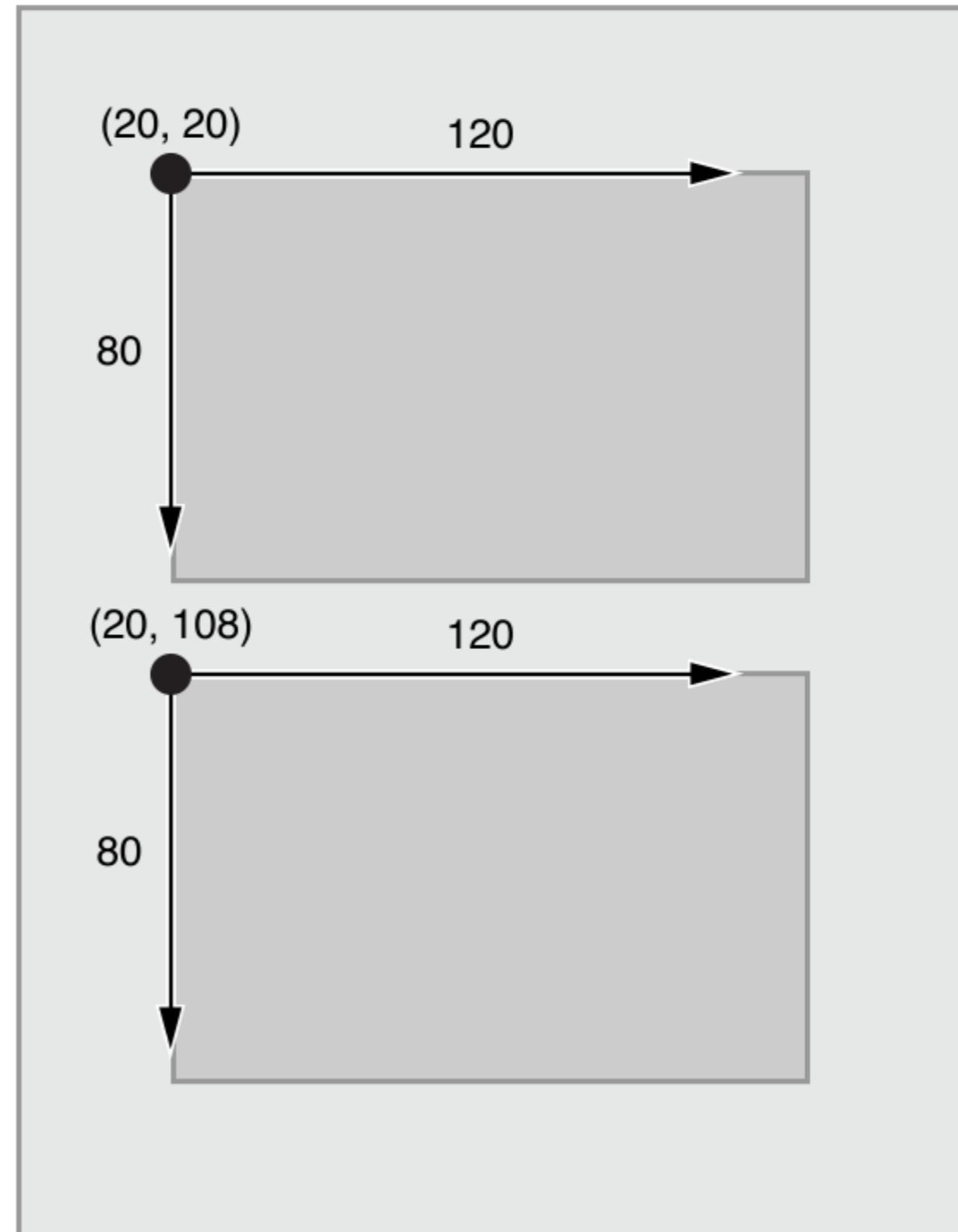
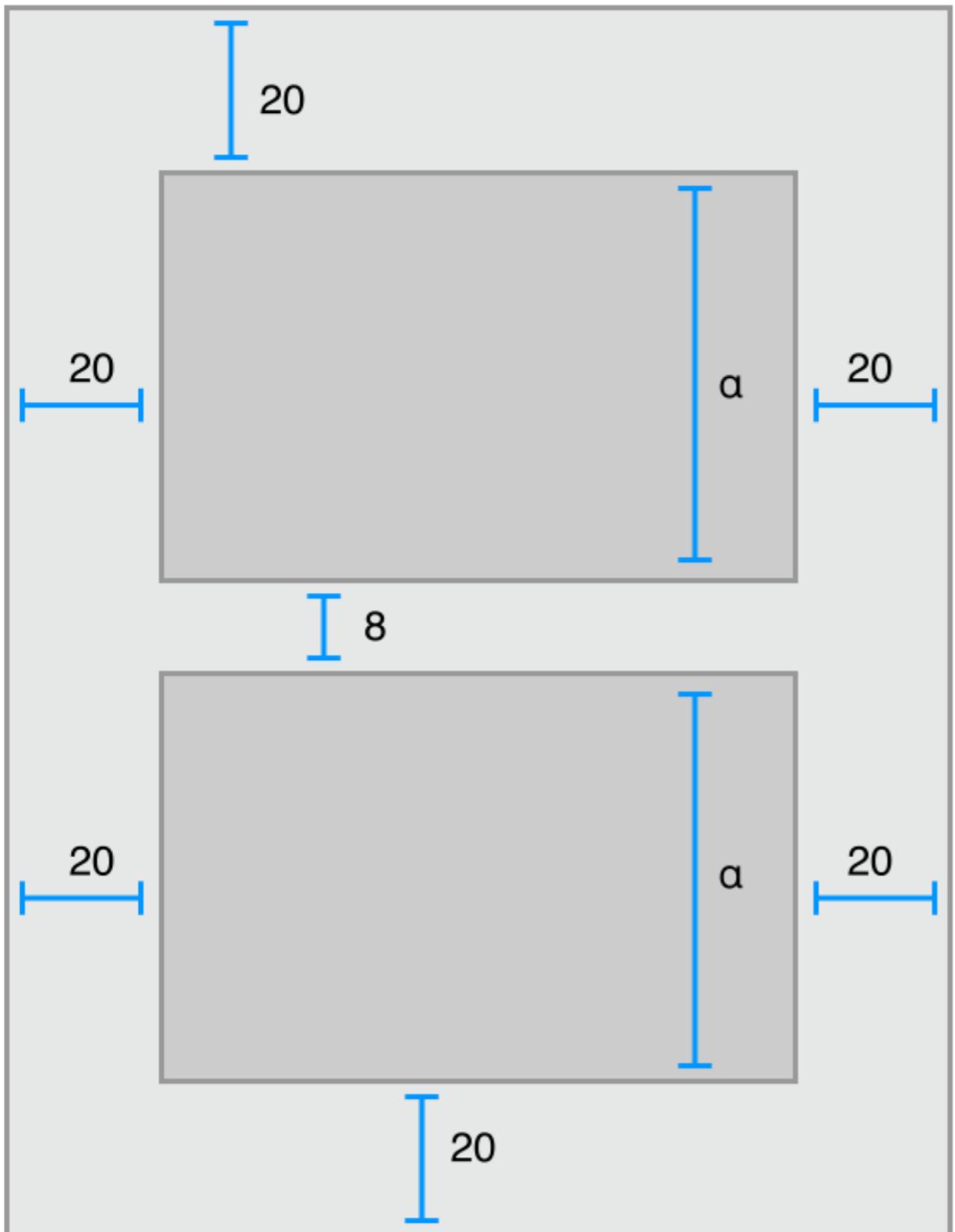
what are our options?

frame-based layout

- define the origin / width / height (frame) for each view
- recalculate frame on layout changes
- implement programmatically

AutoLayout

- define sizing / layout through relational constraints
- implement either in Storyboard or programmatically
- what we'll go over in this class



why autolayout (with storyboard)

- visual (readability pro)
- positioning / frame code is lengthy
- less steep learning curve
- recommended + constantly being updated with new Xcode releases

Auto Layout

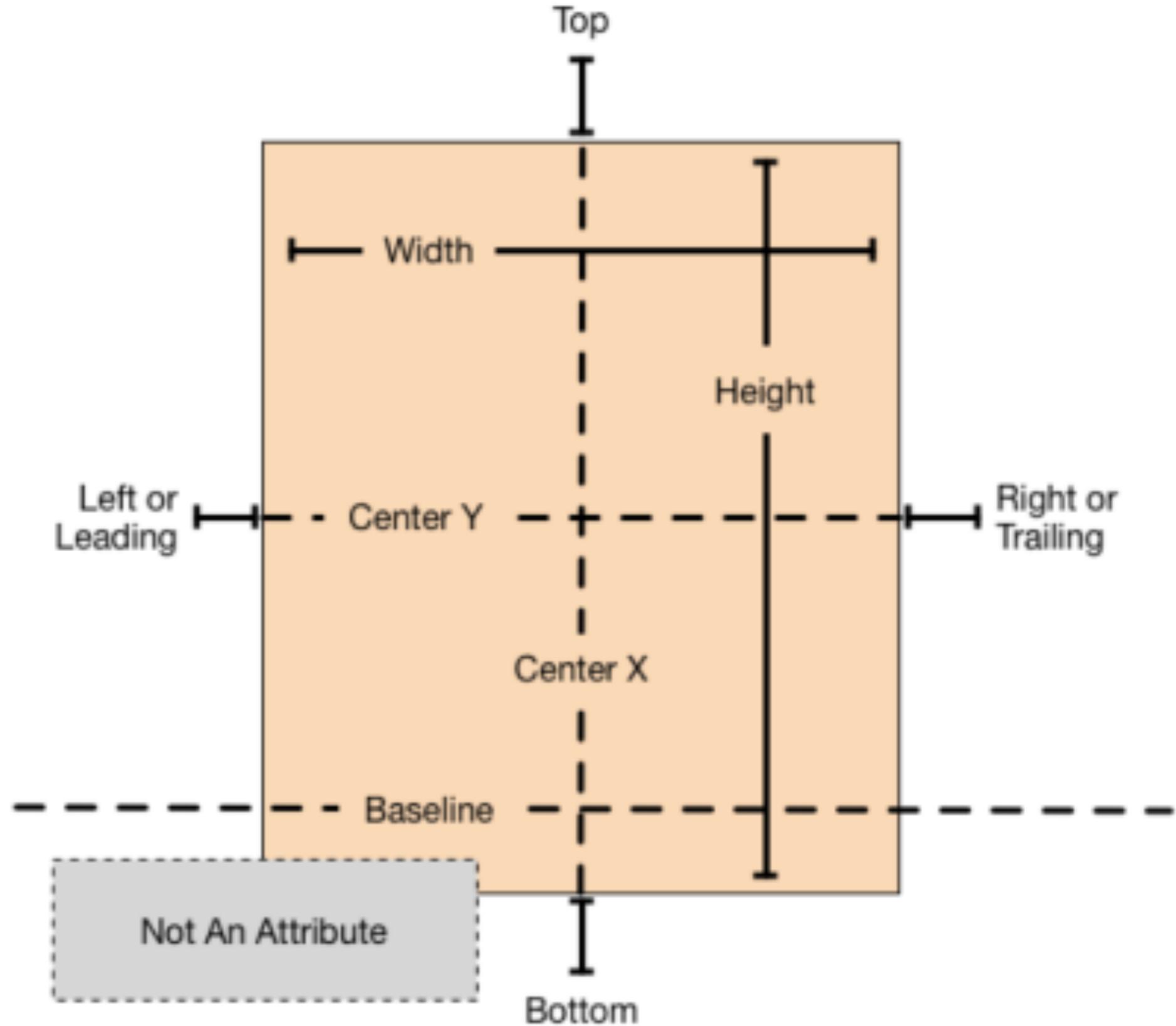
what is Auto Layout?

- constraint based, descriptive layout system
- create adaptive interfaces that responds to changes in screen size and device orientation



layout anchors

Use these properties to create relationships between views



list of constraint types

- **height** - height of view
- **width** - width of view
- **top** - vertical spacing to top view
- **bottom** - vertical spacing to bottom view
- **baseline** - align baseline
- **leading** - spacing to left view
- **trailing** - spacing to right view
- **center x** - center align horizontally
- **center y** - center align vertically

a clarification

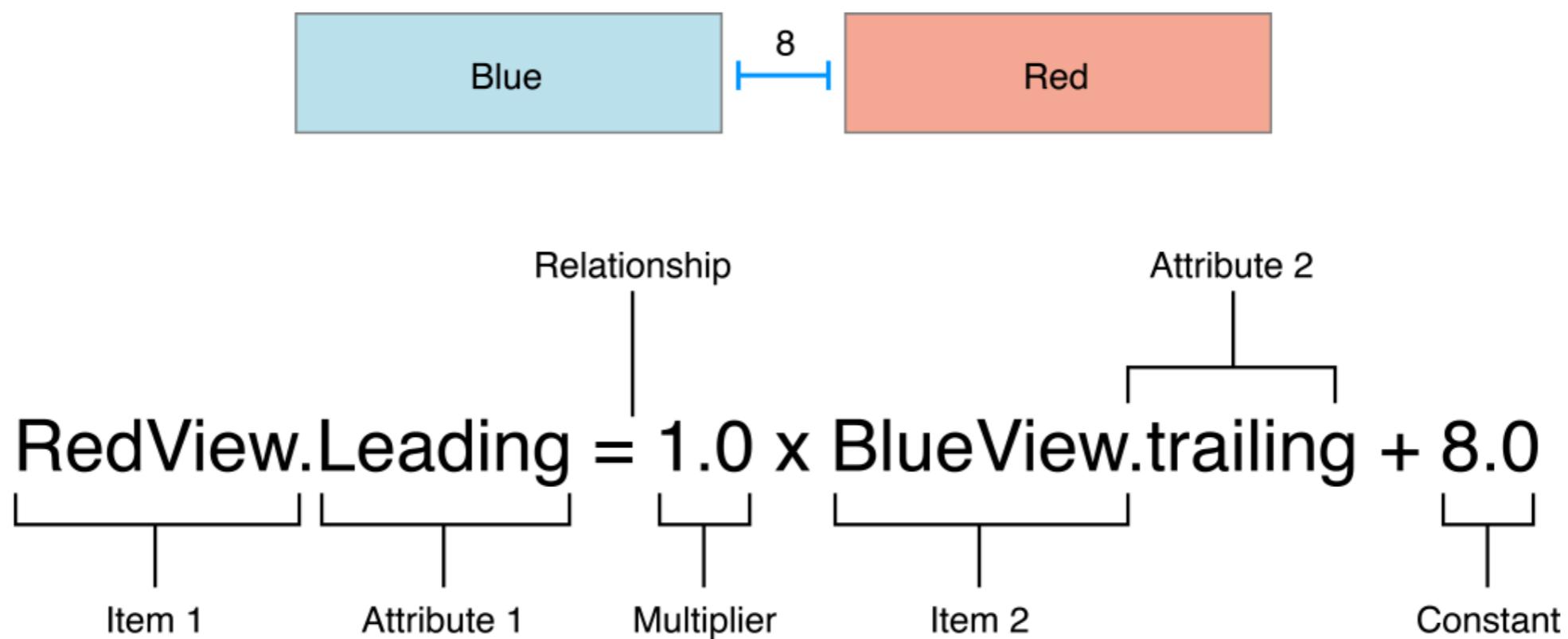
bottom



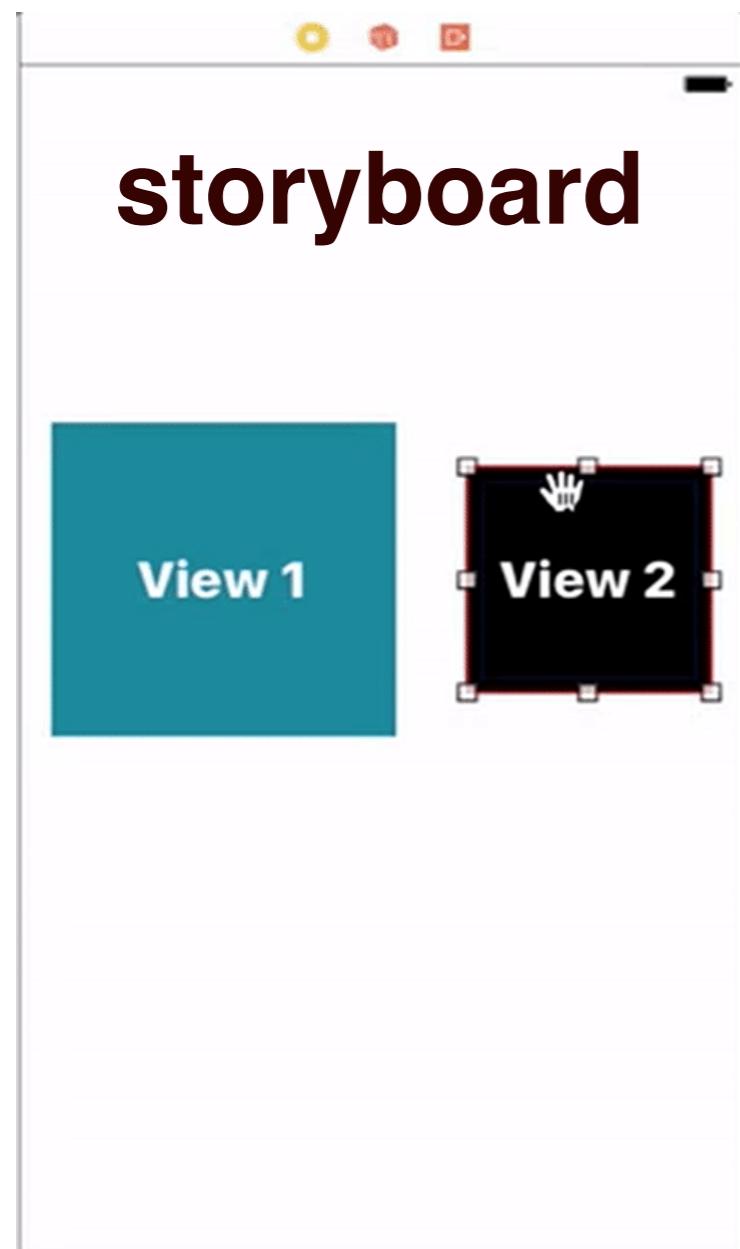
baseline



constraints (high level)



implementing AutoLayout

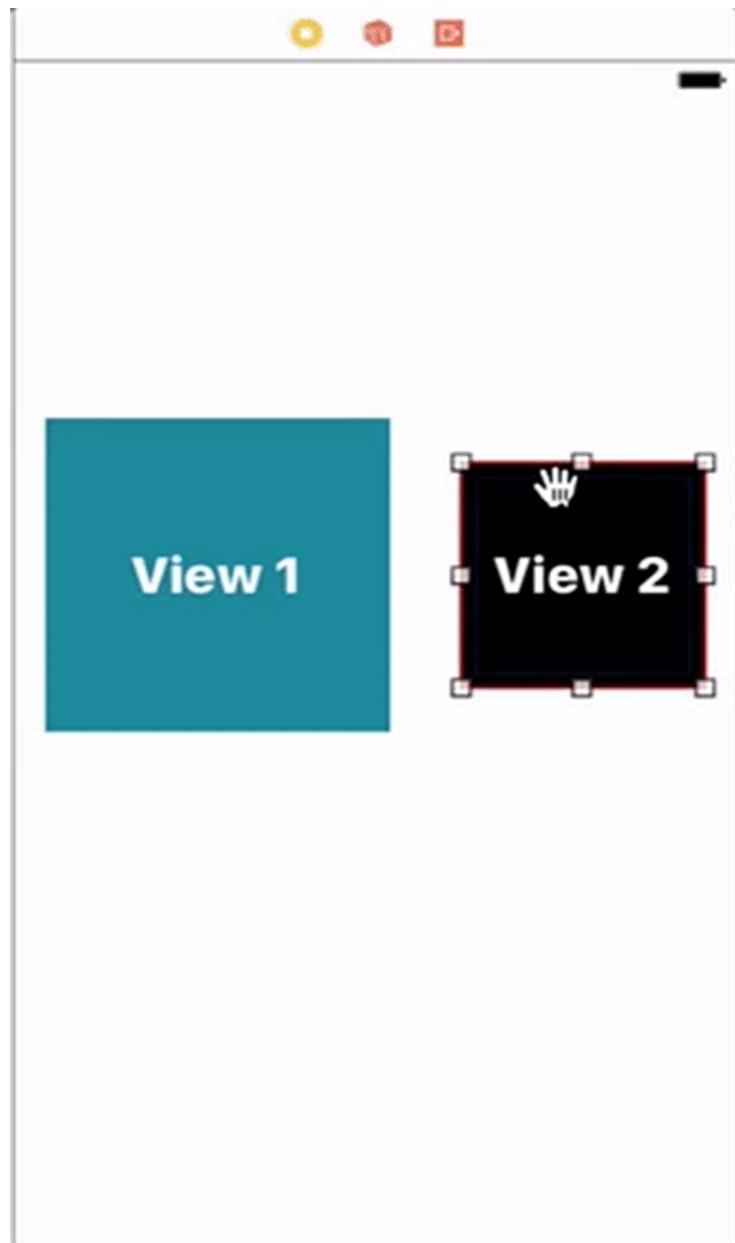


programmatically

```
let constraint =  
    view2.leadingAnchor.constraint(  
        equalTo: view1.trailingAnchor,  
        constant: 8)  
  
constraint.isActive = true
```

in both of these examples, the spacing between view's is set to 8 points

implementing AutoLayout (storyboard)



to create a constraint between two views in Storyboard, you can either...

- **control + drag** between the two views
- **control + drag** between view names in the document outline
- use align + add new constraints menu's

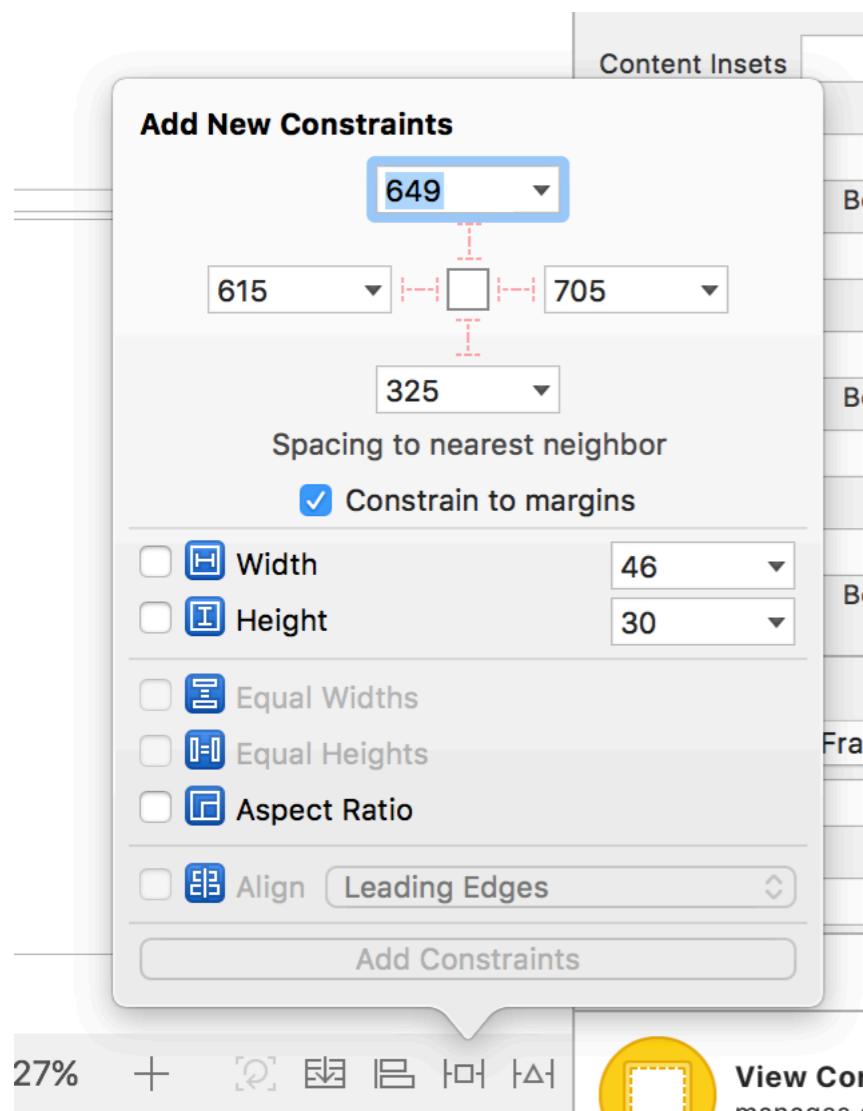
implementing AutoLayout (storyboard)

to create a constraint between two views in Storyboard, you can either...

- **control + drag** between the two views
- **control + drag** between view names in the document outline
- use align + add new constraints menu's



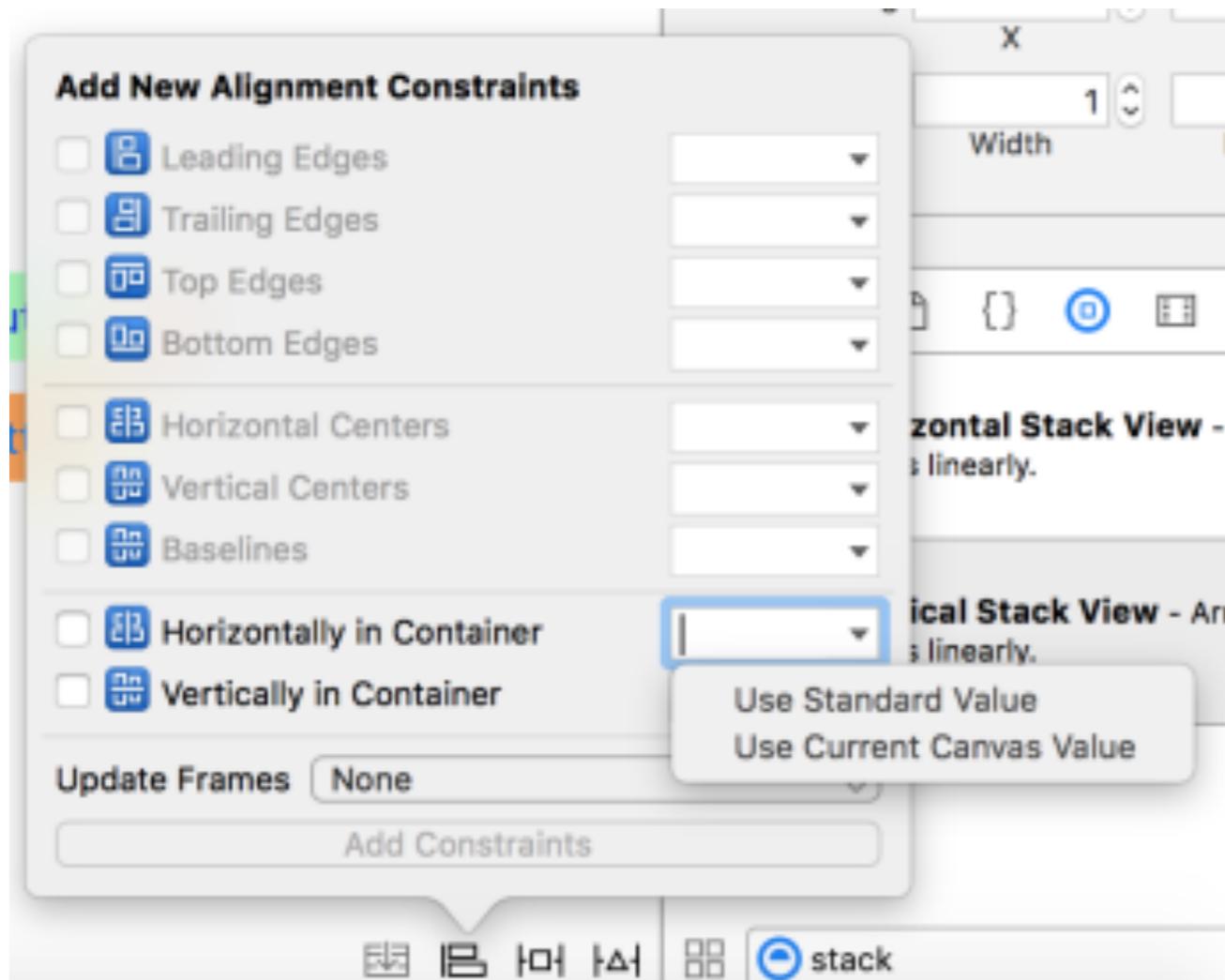
implementing AutoLayout (storyboard)



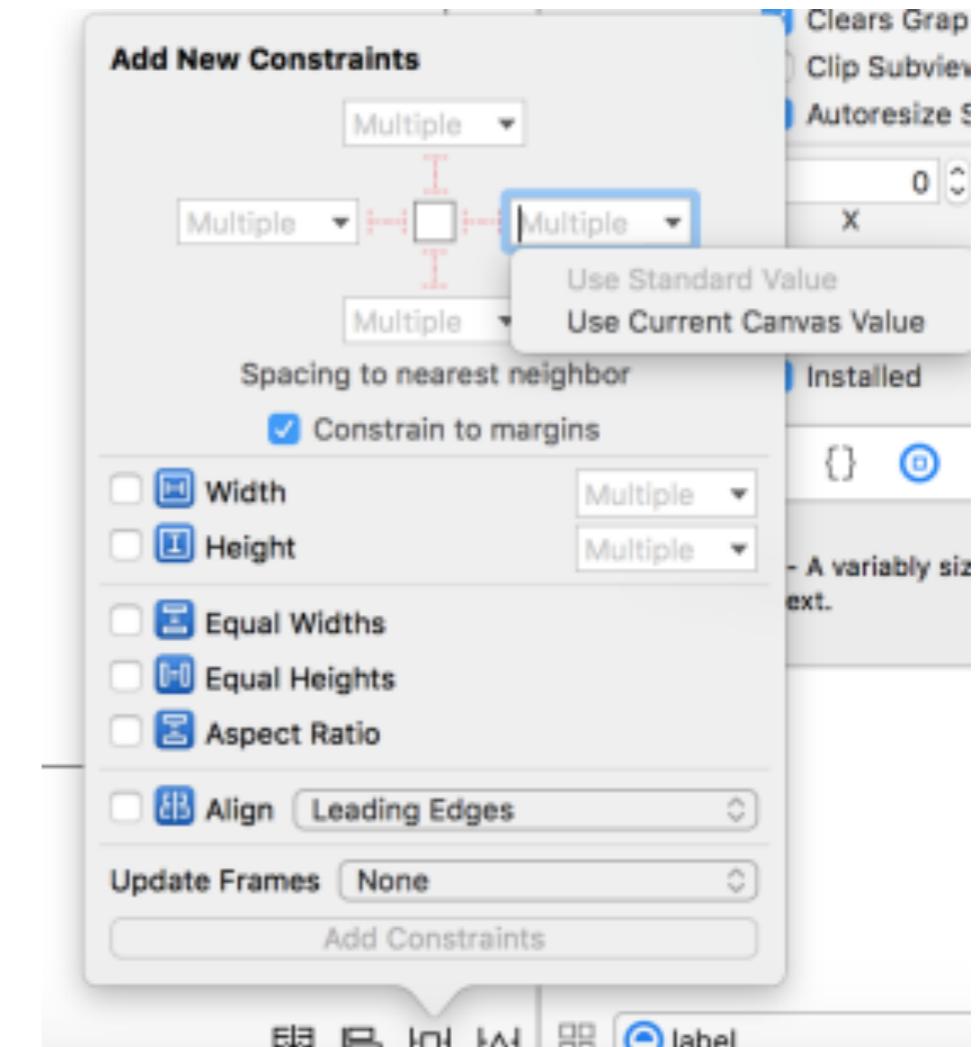
to create a constraint between two views in Storyboard, you can either...

- **control + drag** between the two views
- **control + drag** between view names in the document outline
- **use align + add new constraints** menu's

types of constraints

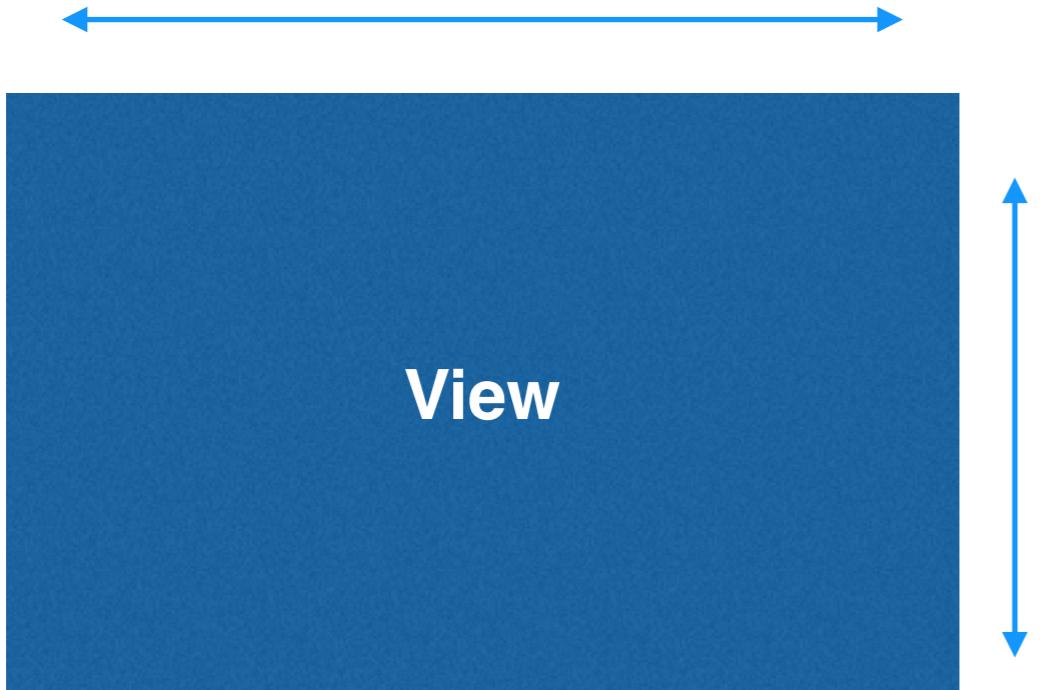


alignment: align objects with each other

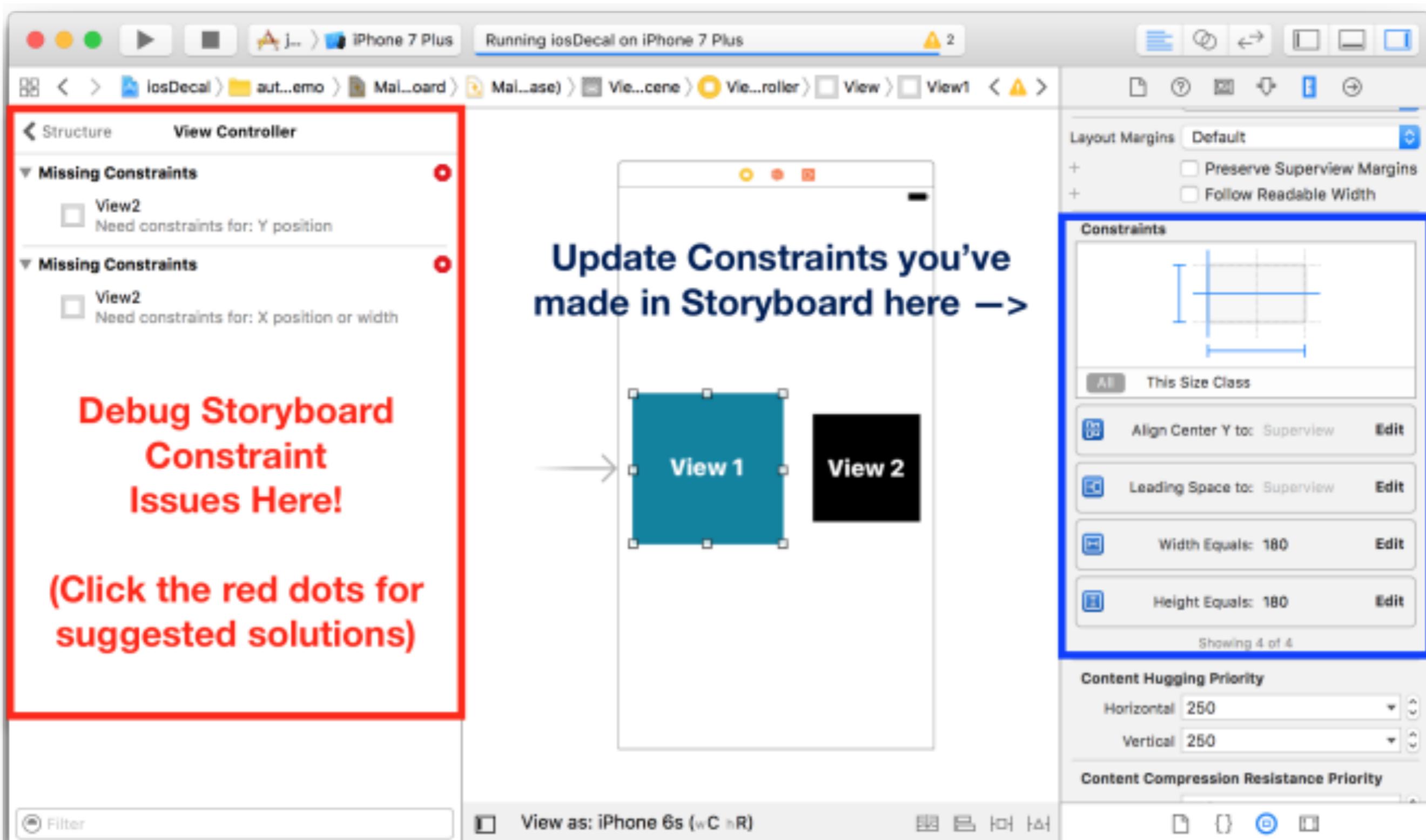


pin: adds space to nearest neighbor (can be a superview or itself)

creating constraints - the philosophy



1. x position
2. y position
3. height
4. width

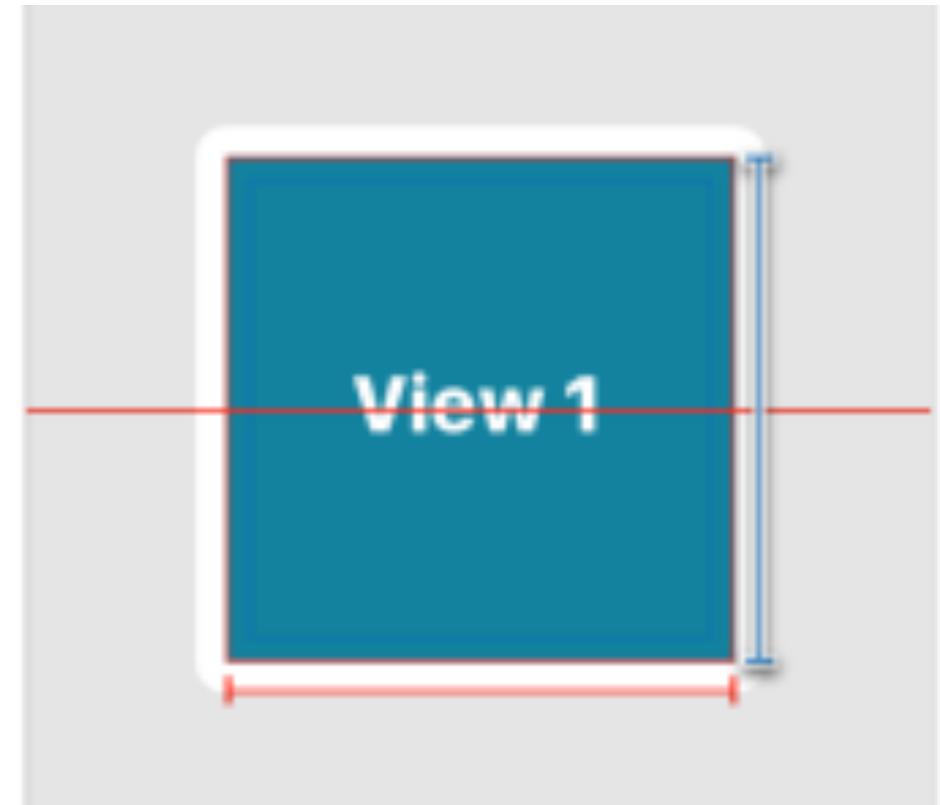


Auto Layout debugging

red lines - Interface Builder

If you see red lines in interface builder, that means you either have:

- too few constraints
(ambiguous)
- too many constraints
(conflicting)

A screenshot of the Xcode Problems list. The title bar says "View Controller". There are two entries under "Missing Constraints":

- "Image View Need constraints for: X position, width" (with a red dot icon)
- "Image View Need constraints for: height" (with a red dot icon)

conflicts



Structure

View Controller

▼ **Conflicting Constraints**



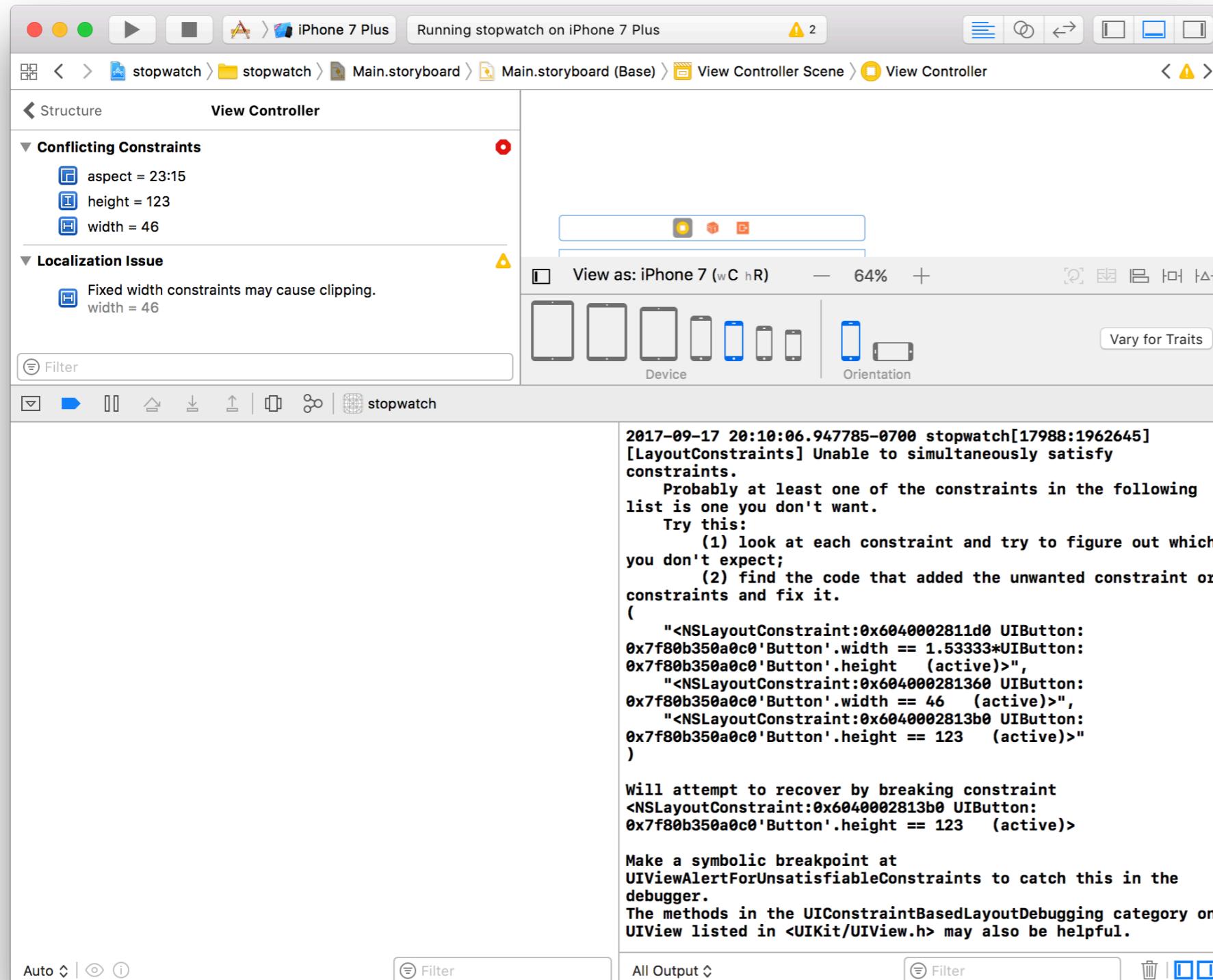
aspect = 23:15

height = 400

width = 46

Xcode will warn you when create multiple conflicting constraints

conflicts



conflicts

The screenshot shows the Xcode interface with a storyboard file open. The top bar indicates "Running stopwatch on iPhone 7 Plus". The storyboard navigation path is: stopwatch > stopwatch > Main.storyboard > Main.storyboard (Base) > View Controller Scene > View Controller. The left sidebar shows "View Controller" selected. Under "Conflicting Constraints", there are three entries: aspect = 23:15, height = 123, and width = 46. Under "Localization Issue", it says "Fixed width constraints may cause clipping." with a note about width = 46. A red warning icon is visible in the top right corner of the storyboard area.

2017-09-17 20:10:06.947785-0700 stopwatch[17988:1962645] [LayoutConstraints] 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.

(

```
<> 2017-09-17 20:10:06.947785-0700 stopwatch[17988:1962645] [LayoutConstraints] 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. (
```

"<NSLayoutConstraint:0x6040002811d0 UIButton:0x7f80b350a0c0'Button'.width == 1.53333*UIButton:0x7f80b350a0c0'Button'.height (active)>",<NSLayoutConstraint:0x604000281360 UIButton:0x7f80b350a0c0'Button'.width == 46 (active)>,<NSLayoutConstraint:0x6040002813b0 UIButton:0x7f80b350a0c0'Button'.height == 123 (active)>")

Will attempt to recover by breaking constraint<NSLayoutConstraint:0x6040002813b0 UIButton:0x7f80b350a0c0'Button'.height == 123 (active)>

Make a symbolic breakpoint at
UIViewAlertForUnsatisfiableConstraints to catch this in the debugger.
The methods in the UIConstraintBasedLayoutDebugging category on
UIView listed in <UIKit/UIKit.h> may also be helpful.

44

localization issues

▼ Localization Issue

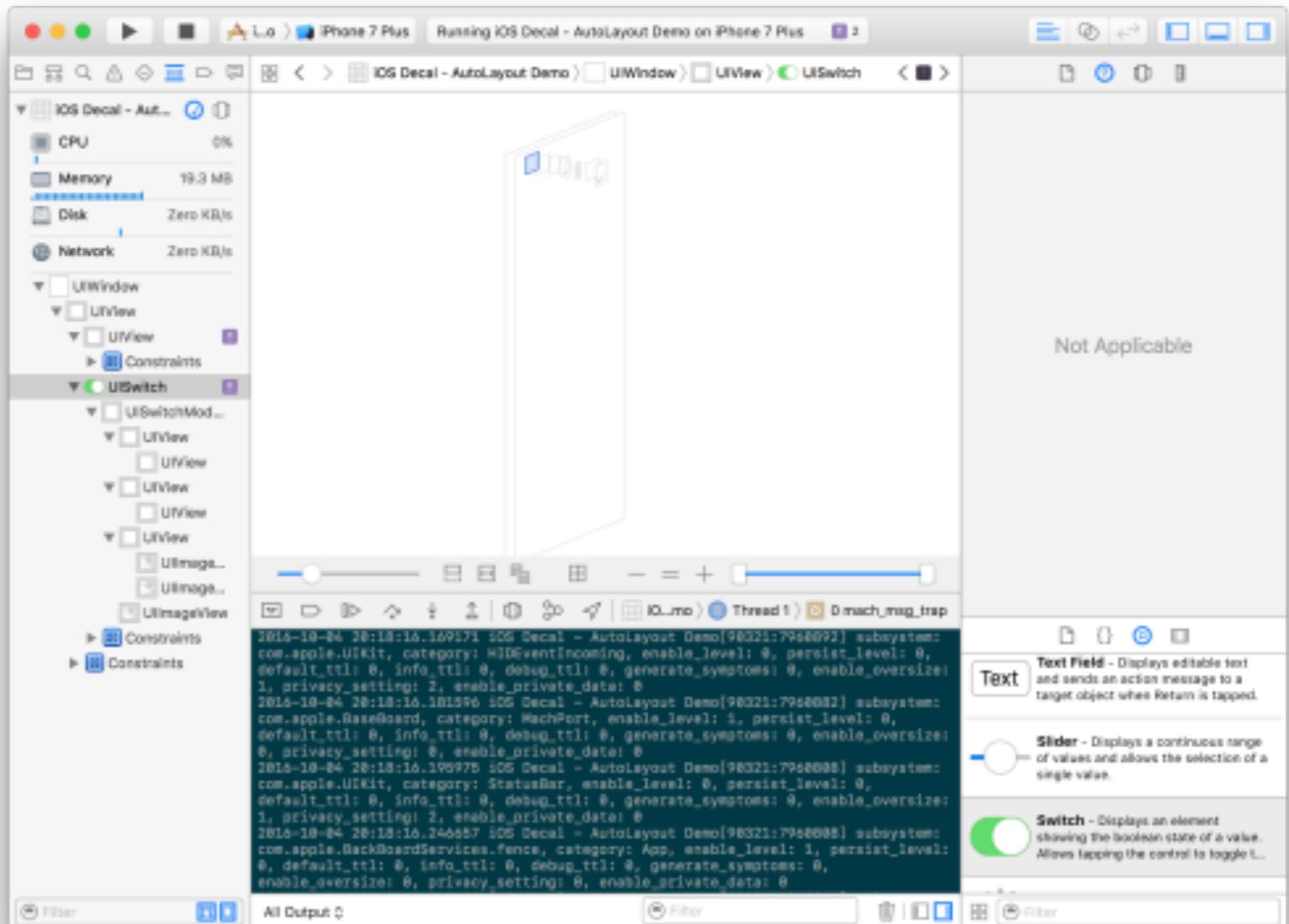
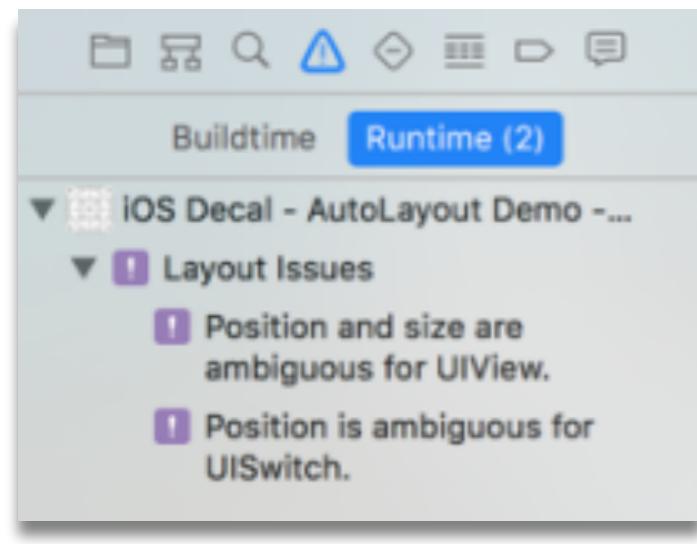


Fixed width constraints may cause clipping.

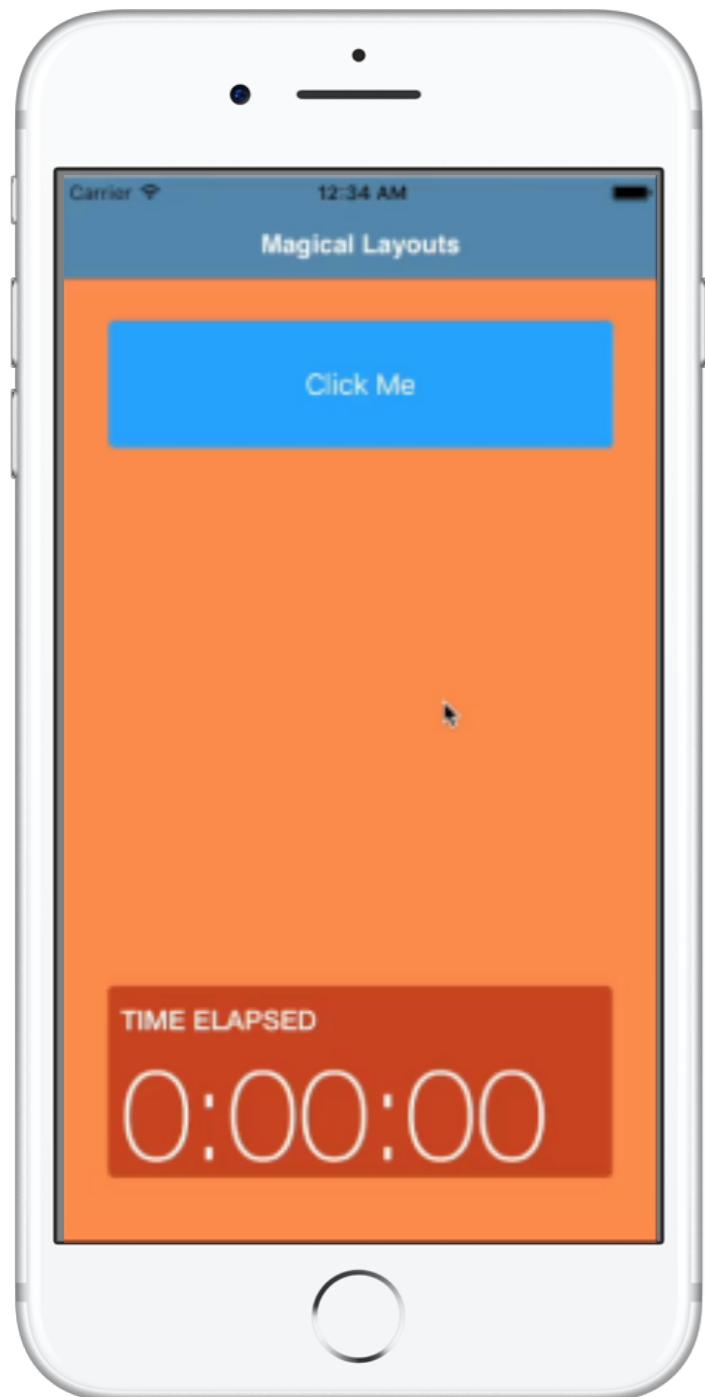
width = 46

fixed width constraints for text elements generate localization warnings

Debug View Hierarchy



Auto Layout with stack views



**use stack views to cut down
on the amount of layout work
you need to do**

side note: setting a stack
subview's hidden property to
true animates beautifully

stack view properties

arrangedSubviews - the views inside the stack

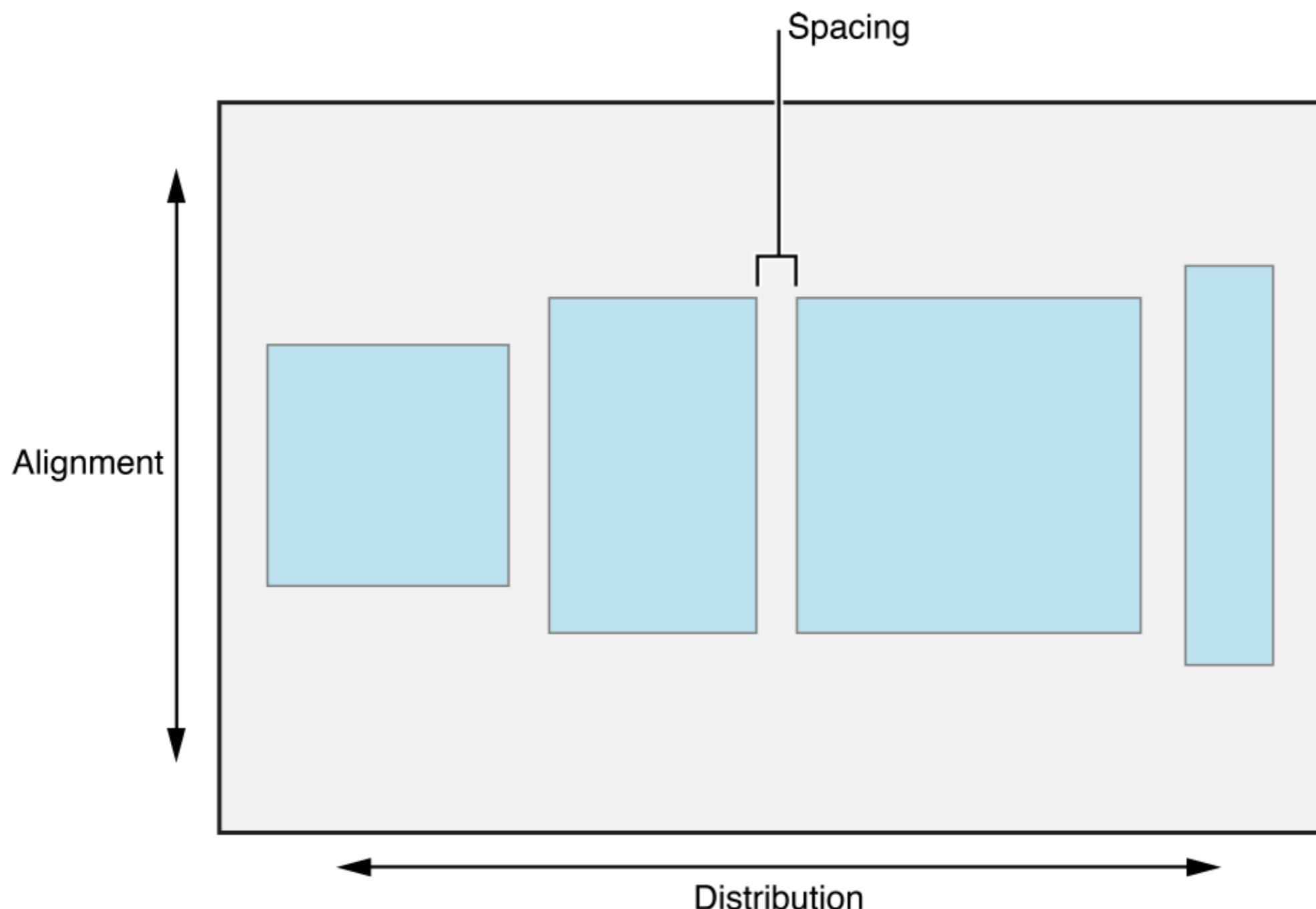
distribution - the distribution of the arranged views **along the stack view's axis**

alignment - The alignment of the arranged subviews
perpendicular to the stack view's axis

axis - horizontal or vertical

spacing - space between subviews

stack view properties

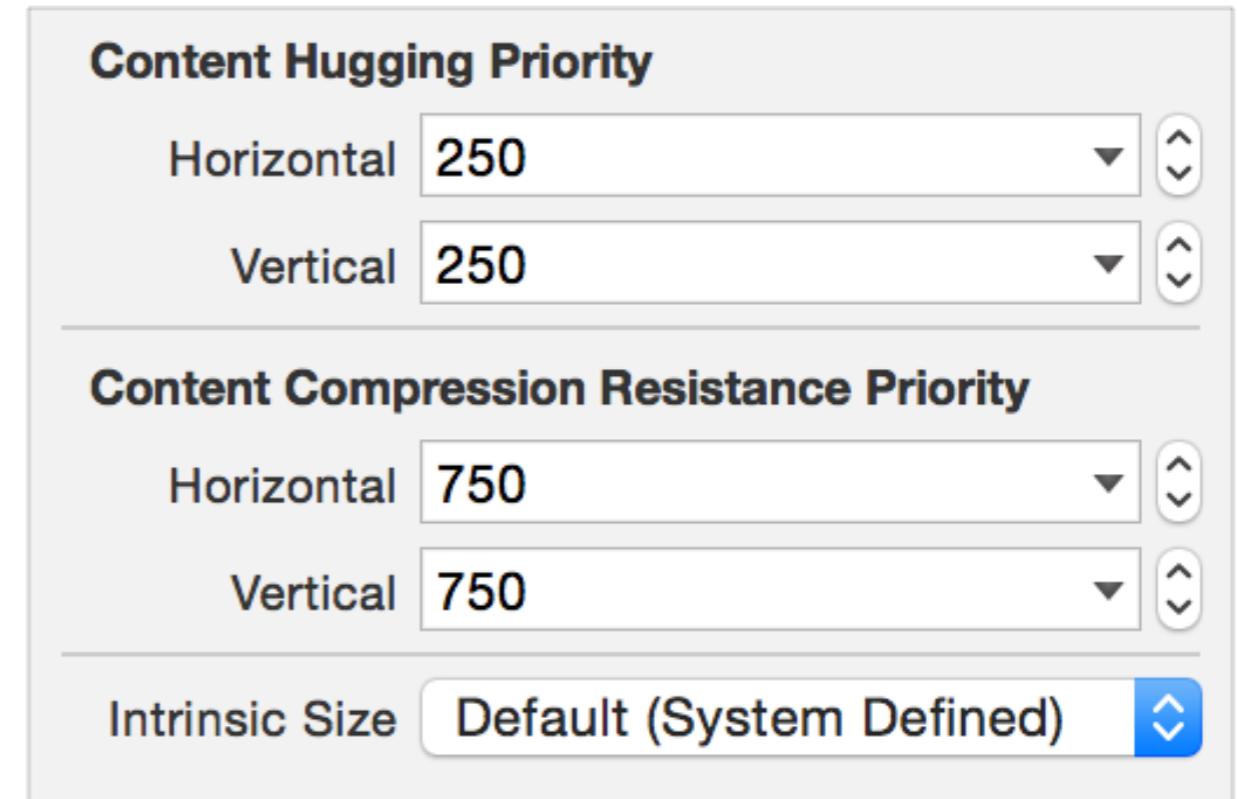


hugging / compression

most of the time - you can avoid setting this

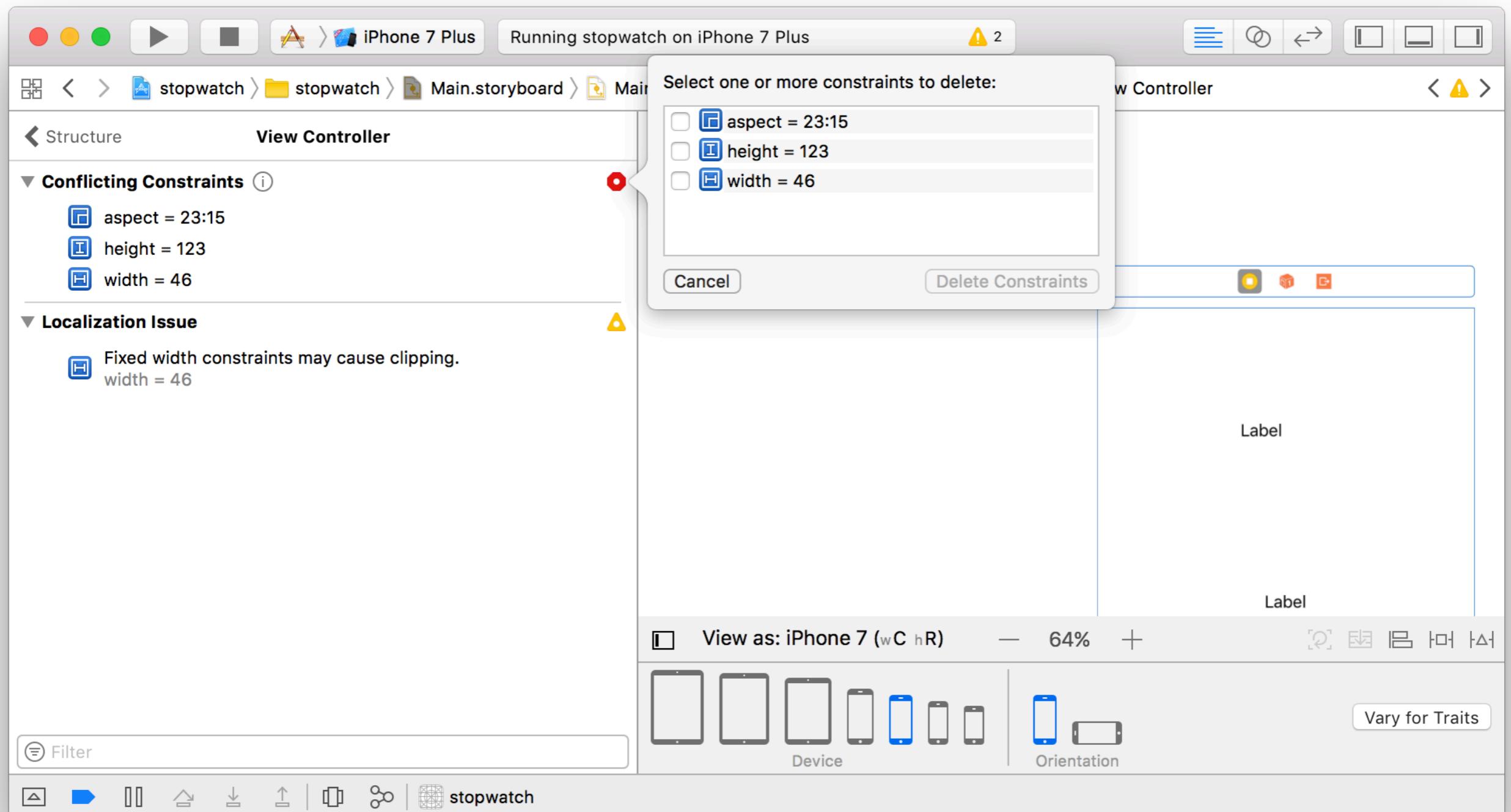
hugging - how much view *does not* want to grow

compression - how much view *does not* want to shrink



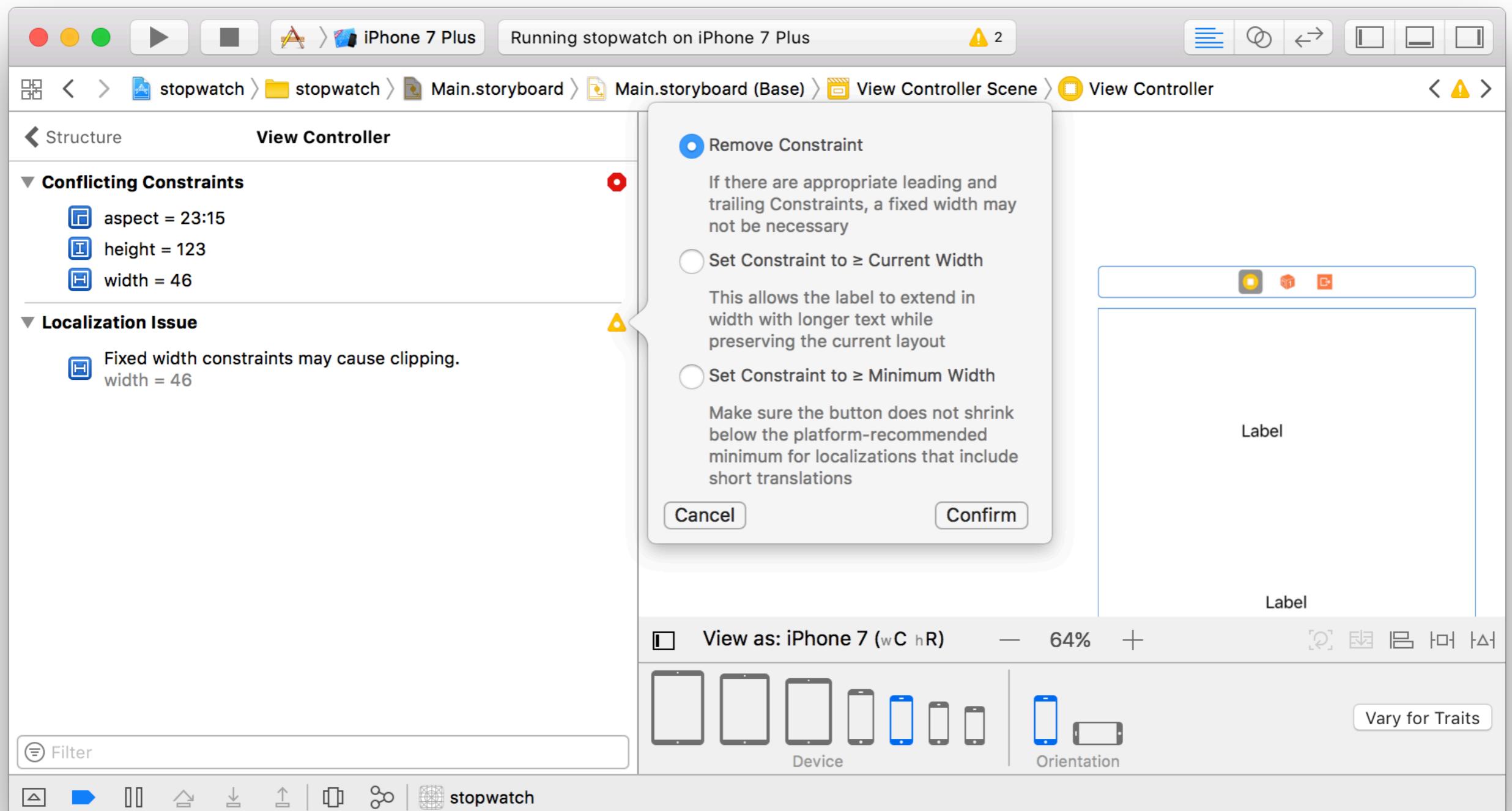
**some AutoLayout
tricks**

resolving conflicts



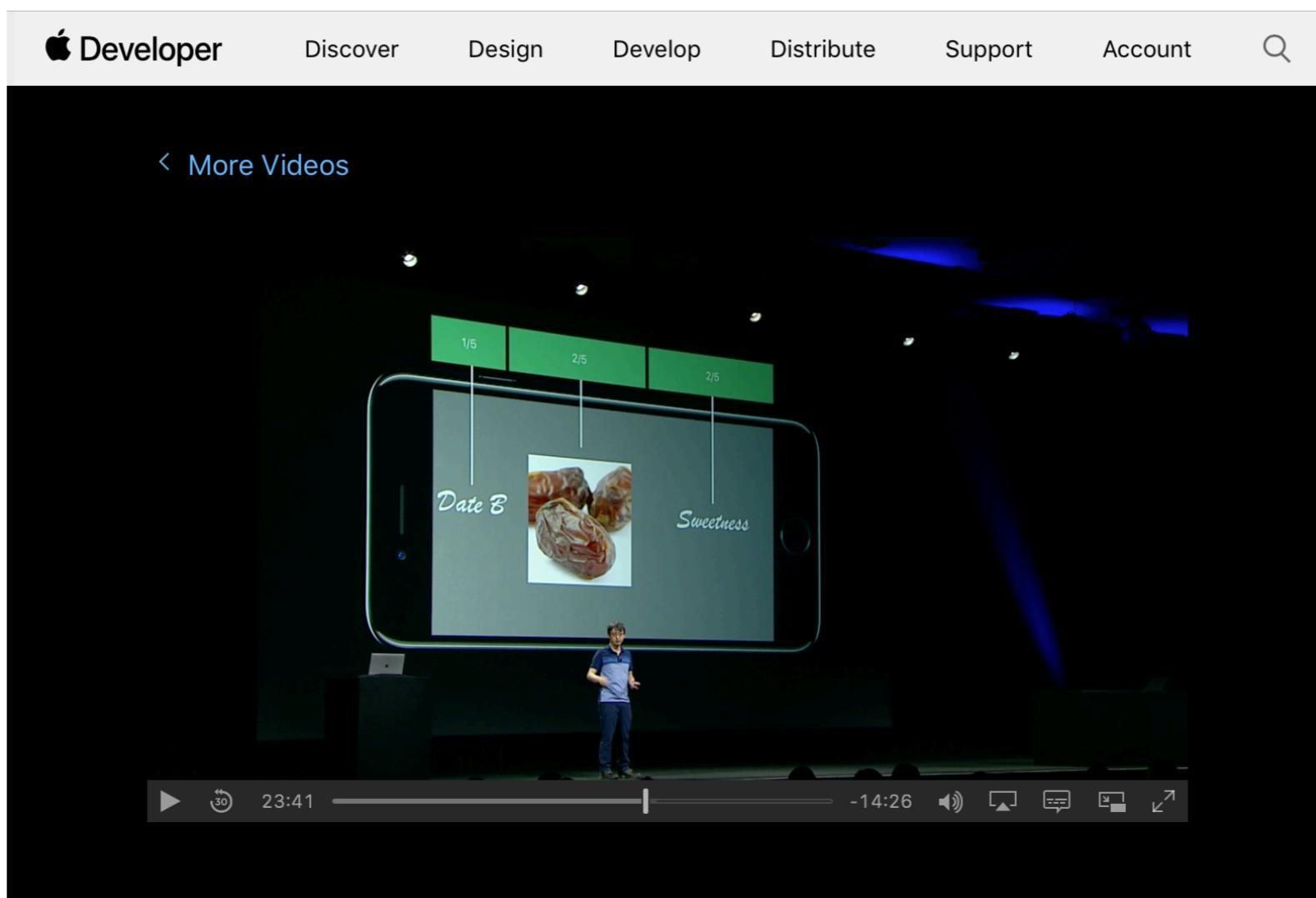
tap on warning icon to reveal tips to resolve your issue

resolving localization issues



spacer views

spacer views - hidden UIViews to enforce proportions



check piazza for a
auto-layout tutorial
video

check-in

hw 2 : hangman
due Monday (2/19) at 11:59 pm

Next Lab : Auto Layout