GENERAL ASSEMBLY

MOBILE DEVELOPMENT

AUTO LAYOUT USING INTERFACE BUILDER

William Martin
Head of Product, Floored

LEARNING OBJECTIVES

- Distinguish between Springs-and-Struts and Auto Layout.
- Define what a “constraint” is and how we apply them to views.
- Devise layouts using Auto Layout.
- Describe the differences between frame and constraint bounds.

REVIEWING VIEWS

UIVIEW

- A *class* that represents drawable, interactive widgets on a screen.
- Typically a rectangular region with a position and size.
- Subclass of UIResponder, that manages events and interactivity.
- Useful by itself, but subclassed into many, more-useful types:
    - UILabel
    - UITextField
    - UITextView
    - etc.

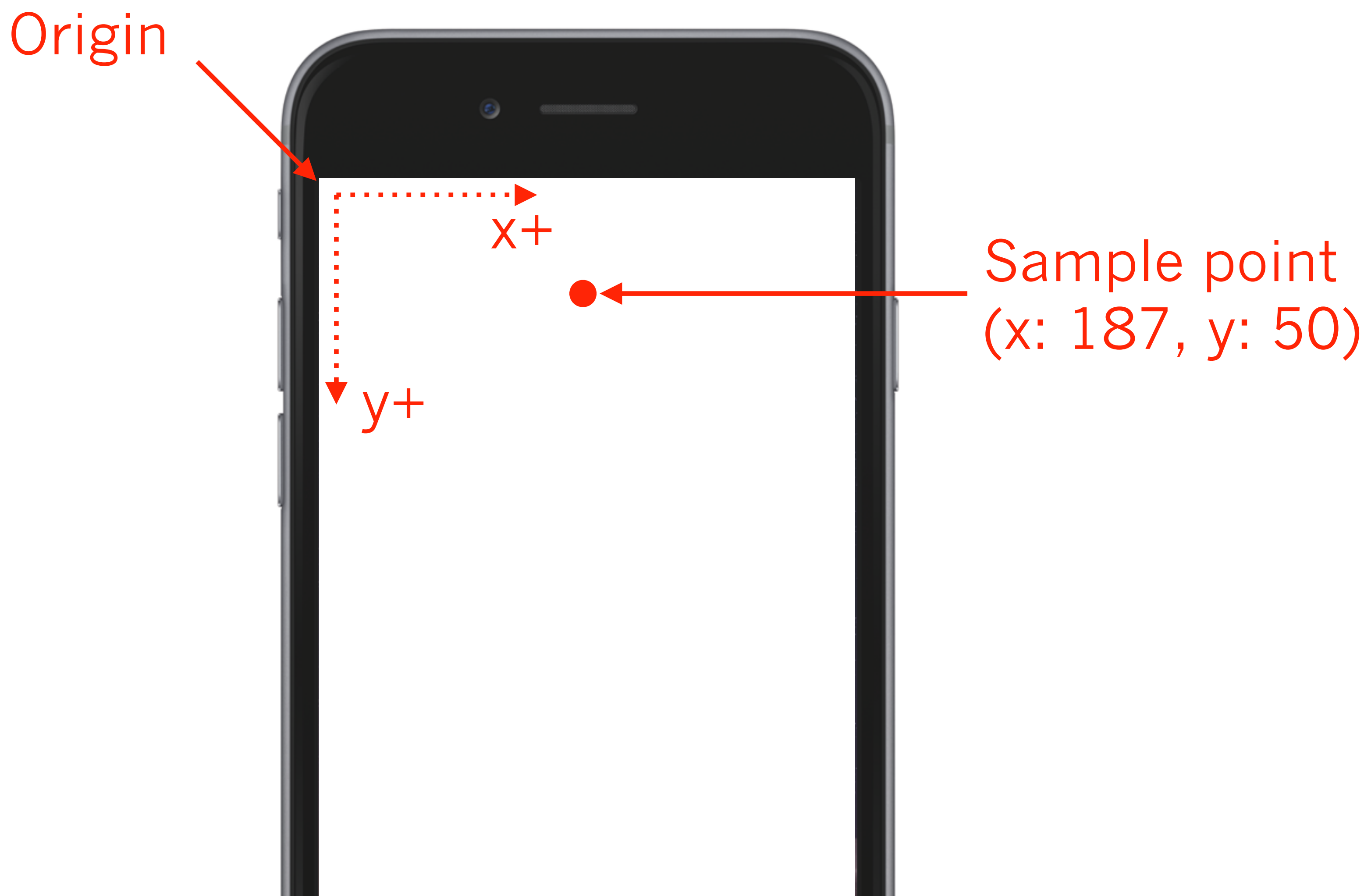
UIVIEW

- superview: The UIView that contains a view in question.
- subviews: The UIViews a view contains. A view can contain multiple views. Sometimes they're called "child" views.
- frame: The position and size of the view within its superview (the *external* coordinate system).
  - Has: origin (x and y coordinates), size (width and height)
  - Property used most often.
- bounds: The view's *internal* coordinates system.
  - Usually, just the frame but with (0, 0) as the origin.

POINTS, NOT PIXELS

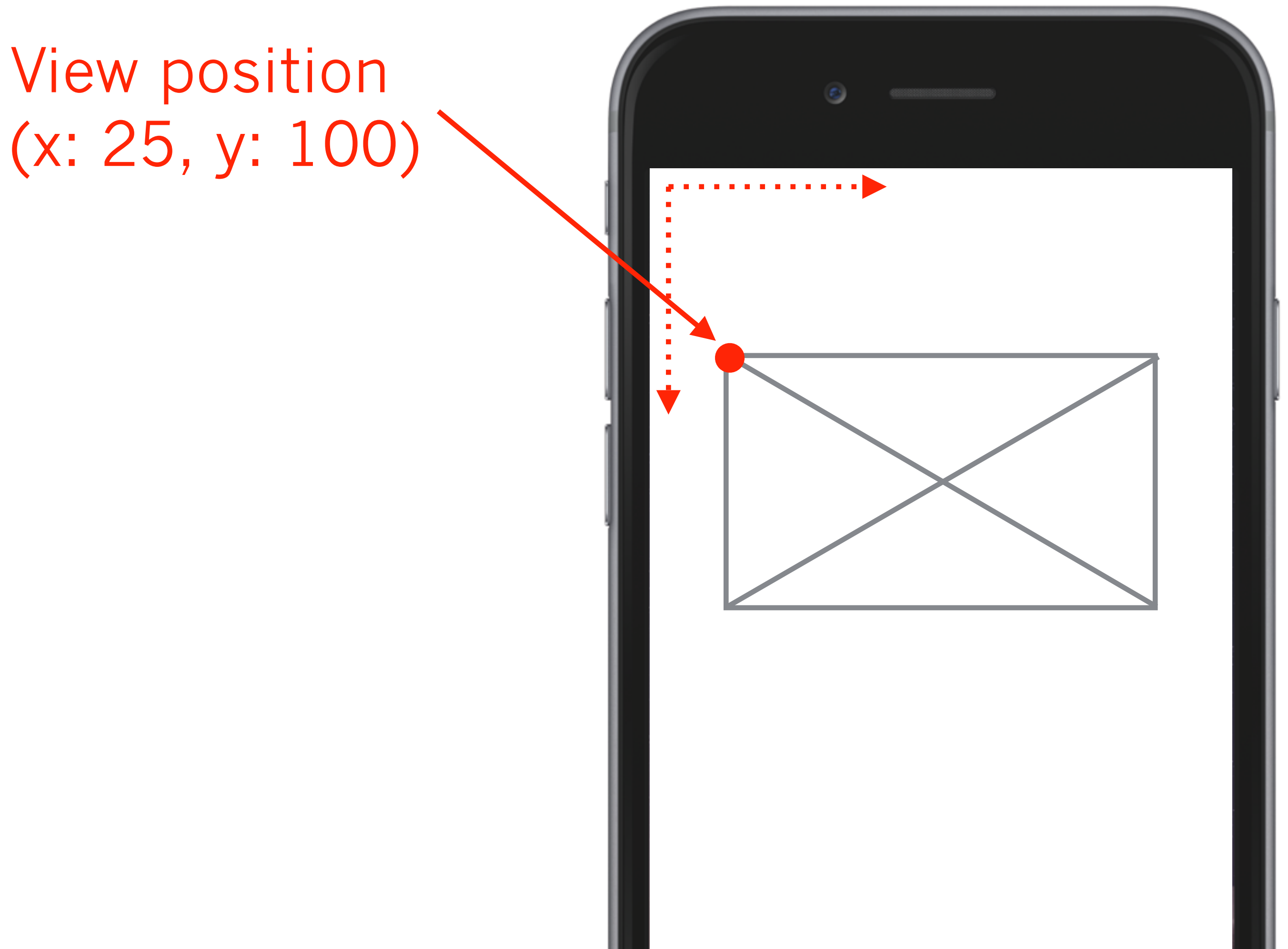
- All UIView work we do in iOS uses **points,** *not* **pixels.**
    - A “point” is a virtual unit that may actually be rendered by multiple physical pixels.
    - http://www.paintcodeapp.com/news/ultimate-guide-to-iphone-resolutions

FRAME = POSITION + SIZE



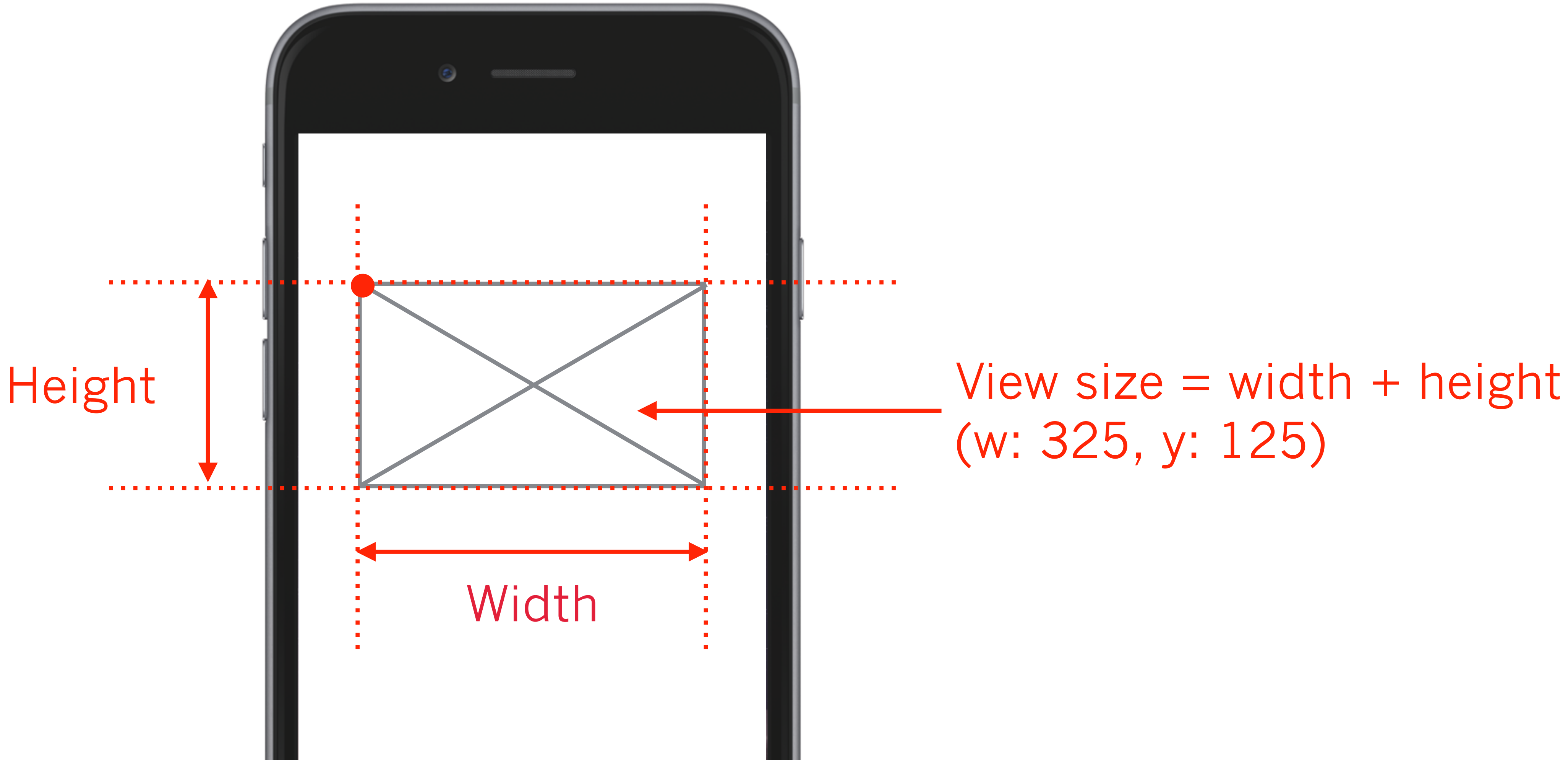
AUTO LAYOUT

FRAME = POSITION + SIZE



AUTO LAYOUT

FRAME = POSITION + SIZE



MANAGING VIEWS IN IOS

- Two ways to manage view hierarchies:
  - Interface Builder (Storyboards and NIBs)
  - Code
- Three ways to lay out views:
  - Springs & struts (the older way)
  - Auto Layout (the newer way, what we'll cover)
  - Manually, with code.

BEGINNING AUTO LAYOUT

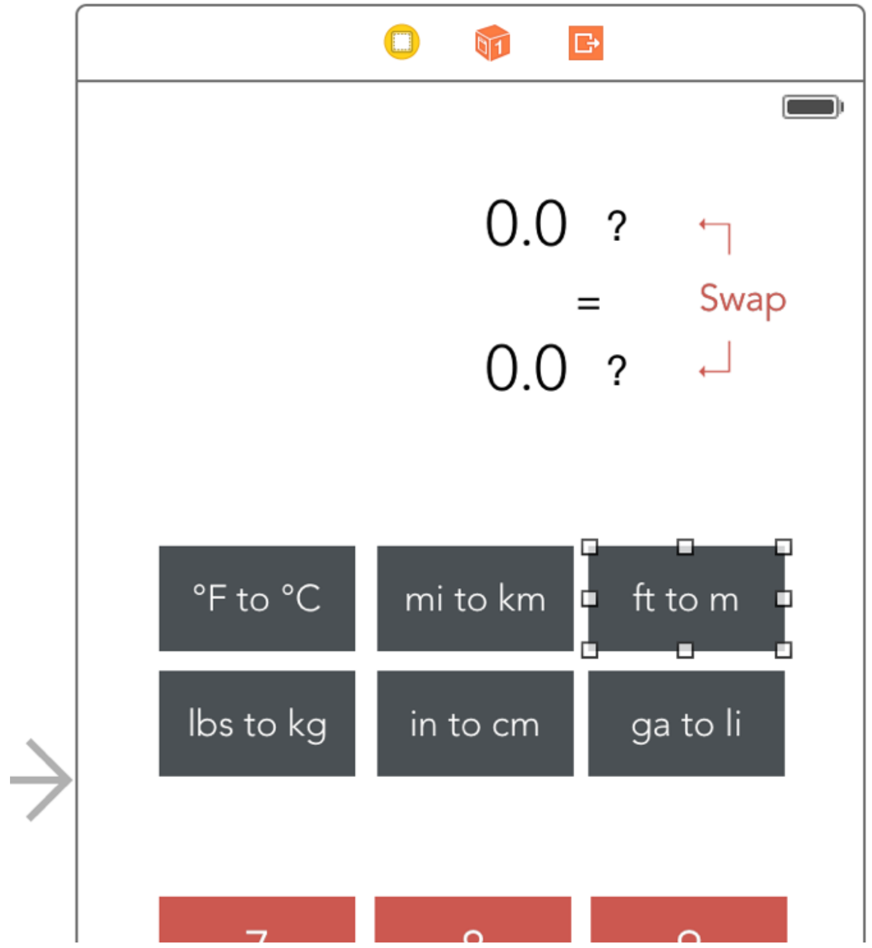
SPRINGS + STRUTS

Lays out views with respect to their superviews (only).

- Springs are flexible and can scale with the superview (width + height).
- Struts are inflexible, used for margins.

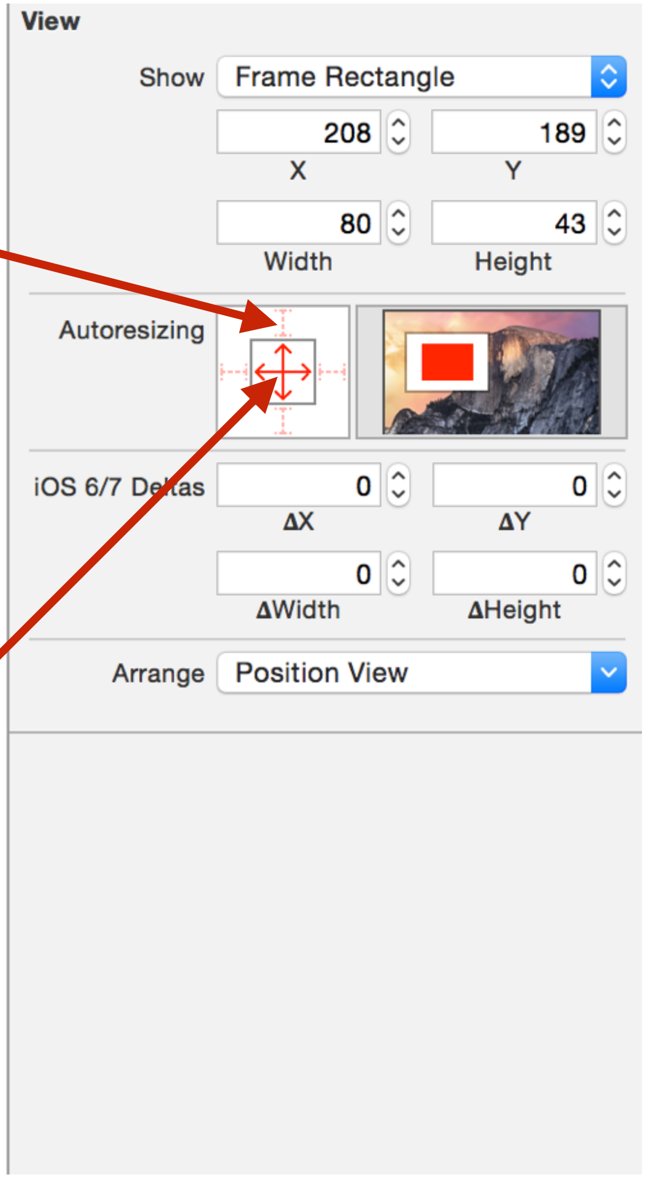
AUTO LAYOUT

SPRINGS + STRUTS



Struts

Springs


View

Show: Frame Rectangle

X: 208	Y: 189
Width: 80	Height: 43

Autoresizing

iOS 6/7 Deltas

ΔX: 0	ΔY: 0
ΔWidth: 0	ΔHeight: 0

Arrange: Position View

SPRINGS + STRUTS

Pros:

- Much simpler to understand and implement than Auto Layout.
- Good for a large set of cases.

Cons:

- Can’t describe relationships between "sibling" views explicitly.
- Can become problematic when views start changing size (e.g. text views that change size when a user adds text).

WHAT IS AUTO LAYOUT?

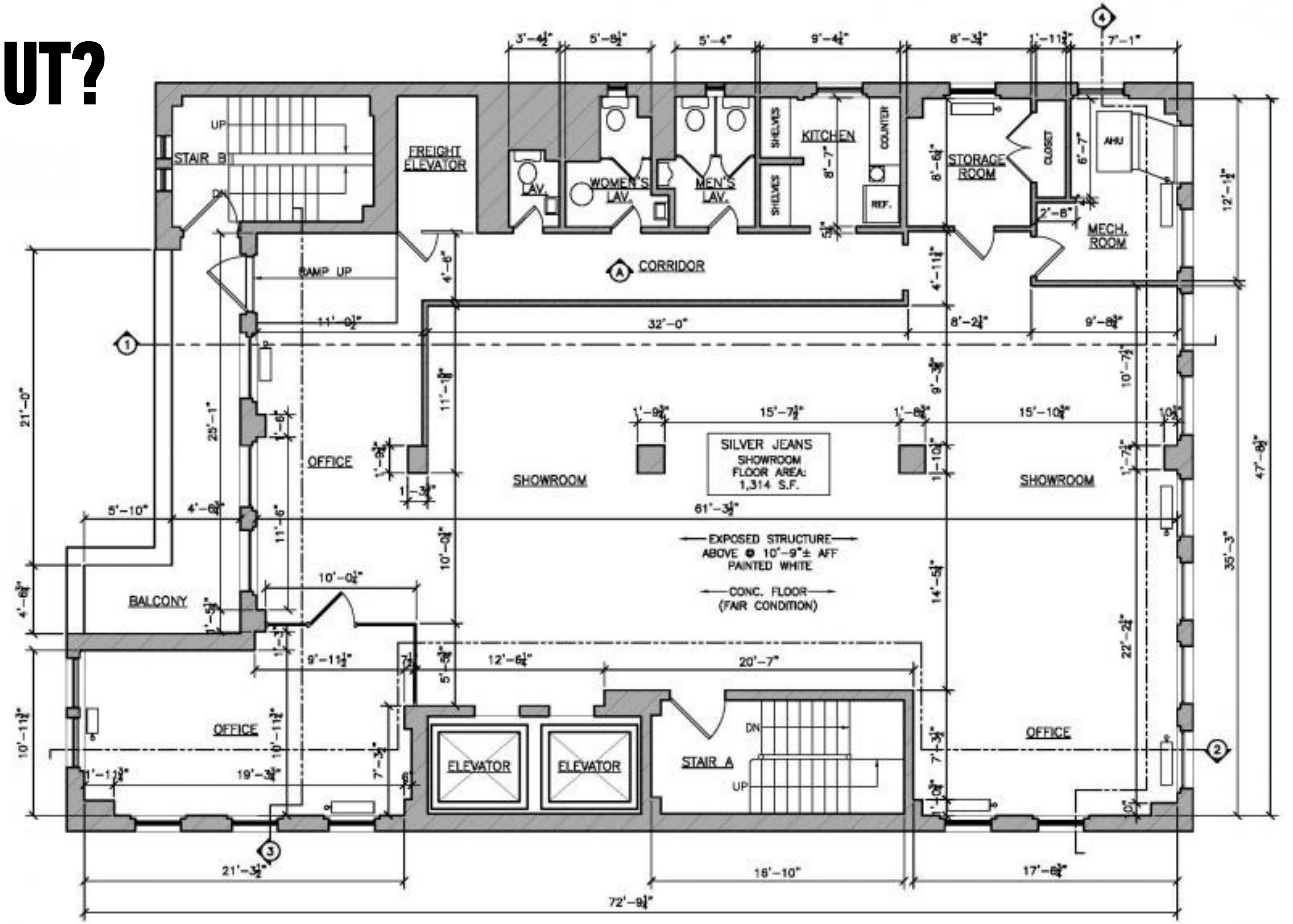
- A newer, complex, more general system to lay out views.
- Uses “constraints” that describe relationships between views.
- Uses a "solver" that updates view positions and sizes when:
    - view sizes change,
    - the device is reoriented, or
    - the app is run on different devices.

WHAT IS AUTO LAYOUT?

Similar to dimensions in an architectural plan.

They act to describe to a contractor where things should go during construction.

In Auto Layout, these dimensions are prescriptive and dynamic.

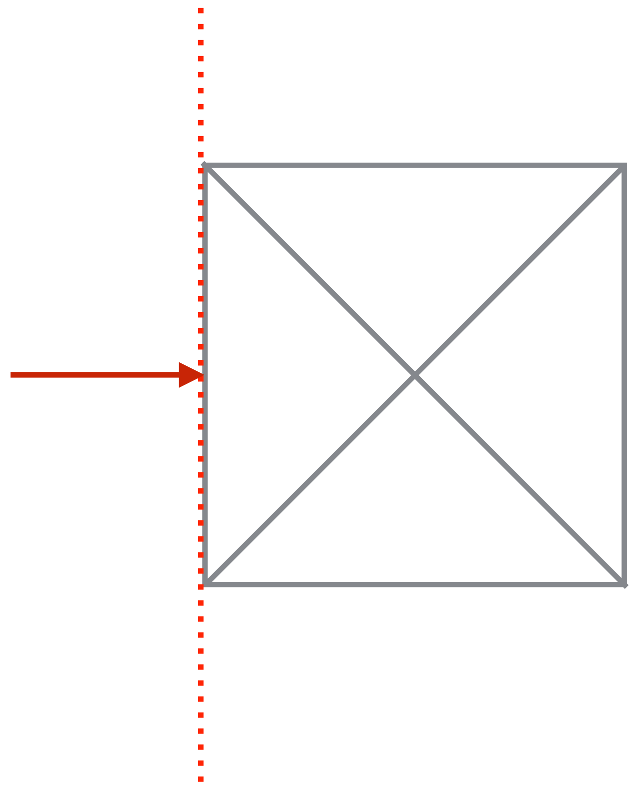


AUTO LAYOUT GOTCHYAS:

- First, “Auto Layout” is a misnomer. It’s quite manual and tedious to set up and use. The "automatic" part comes from the fact that layouts are computed in an automated way, but springs-and-struts are automatic, too.
- Auto Layout is counter-intuitive; the terminology is not obvious; and the tools are non-deterministic.
- Auto Layout is hard. Every time you hear or read “easily” or “as simple as” with regards to Auto Layout itself or the tools IB gives you to manage it, *it’s a lie.*

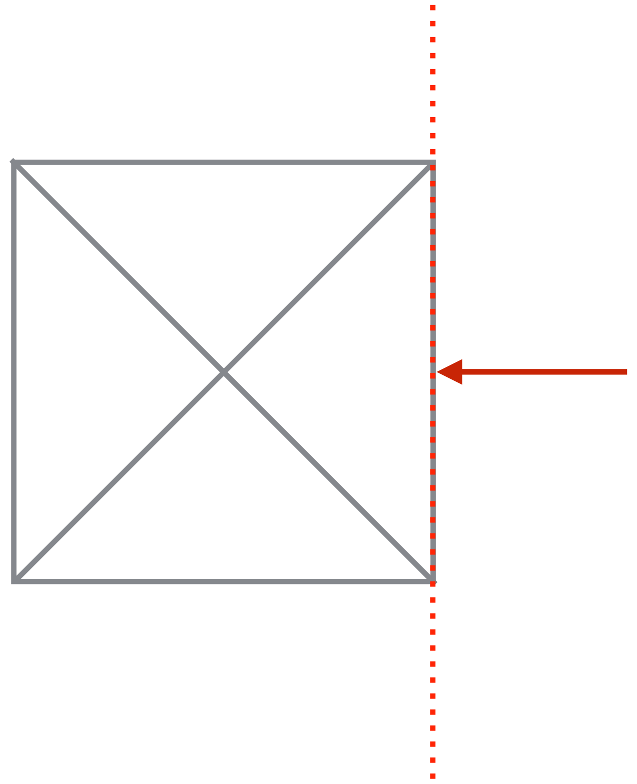
TERMINOLOGY

- "Leading edge" – For languages that read left-to-right, it's the x-coordinate of the left edge.



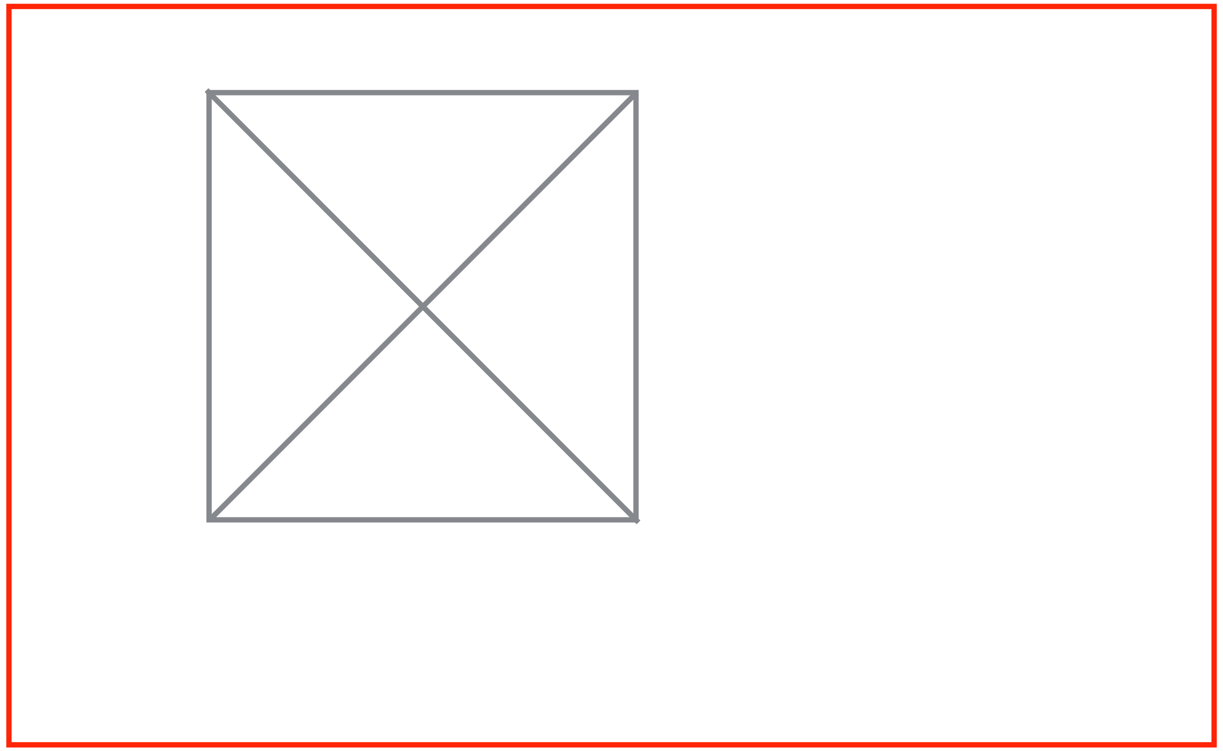
TERMINOLOGY

- "Trailing edge" – For languages that read left-to-right, it's the x-coordinate of the right edge.



TERMINOLOGY

- "Container" – Synonym for superview: the view that contains a particular view.

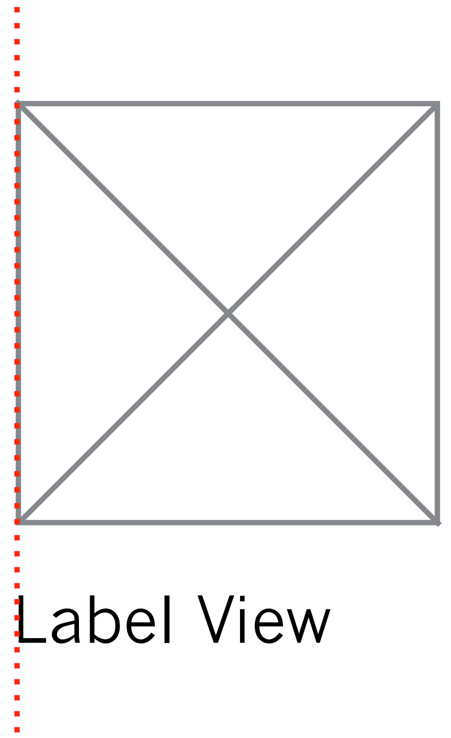


CONSTRAINTS

- A single "constraint" describes a one-dimensional relationship about a single view or between two views.
- E.g. "The width of the sidebar should be 100 points."
- E.g. “The height of the body should equal the height of the sidebar.”
- Constraints consist of:
  - A “first” view and attribute.
  - A “relation.”
  - A “second” view and attribute.
  - Multiplier
  - Constant
  - Priority

CONSTRAINT EXAMPLES

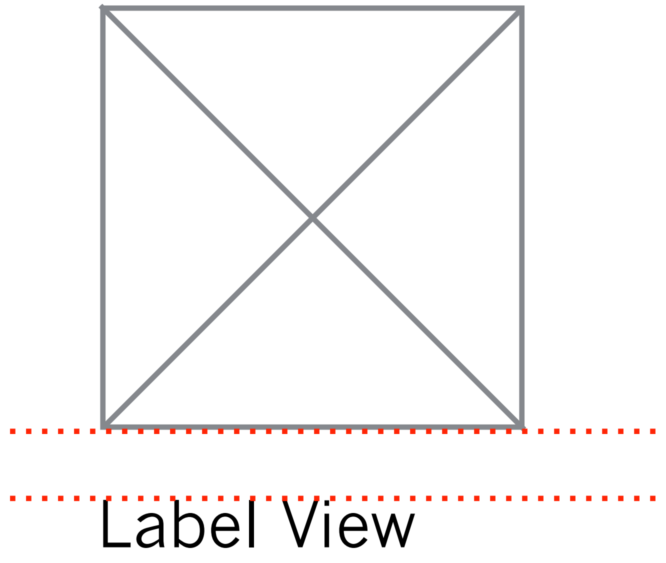
- First view + attribute: MyLabelView.Leading
- Relation: Equal
- Second view + attribute: MyImageView.Leading
- Multiplier: 1
- Constant: 0



- What does it mean?
    - The left coordinate of a Label should equal the left coordinate of an Image.

CONSTRAINT EXAMPLES

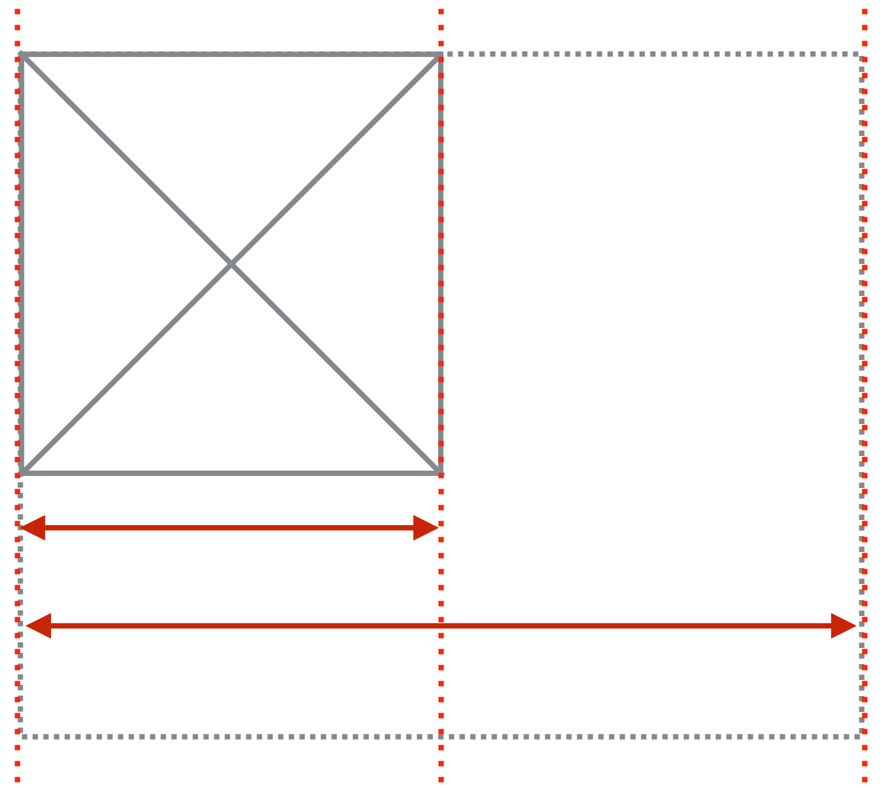
- First view + attribute: MyLabelView.Top
- Relation: Equal
- Second view + attribute: MyImageView.Bottom
- Multiplier: 1
- Constant: 10



- What does it mean?
  - The top of the Label should be 10 points below the bottom of the Image.

CONSTRAINT EXAMPLES

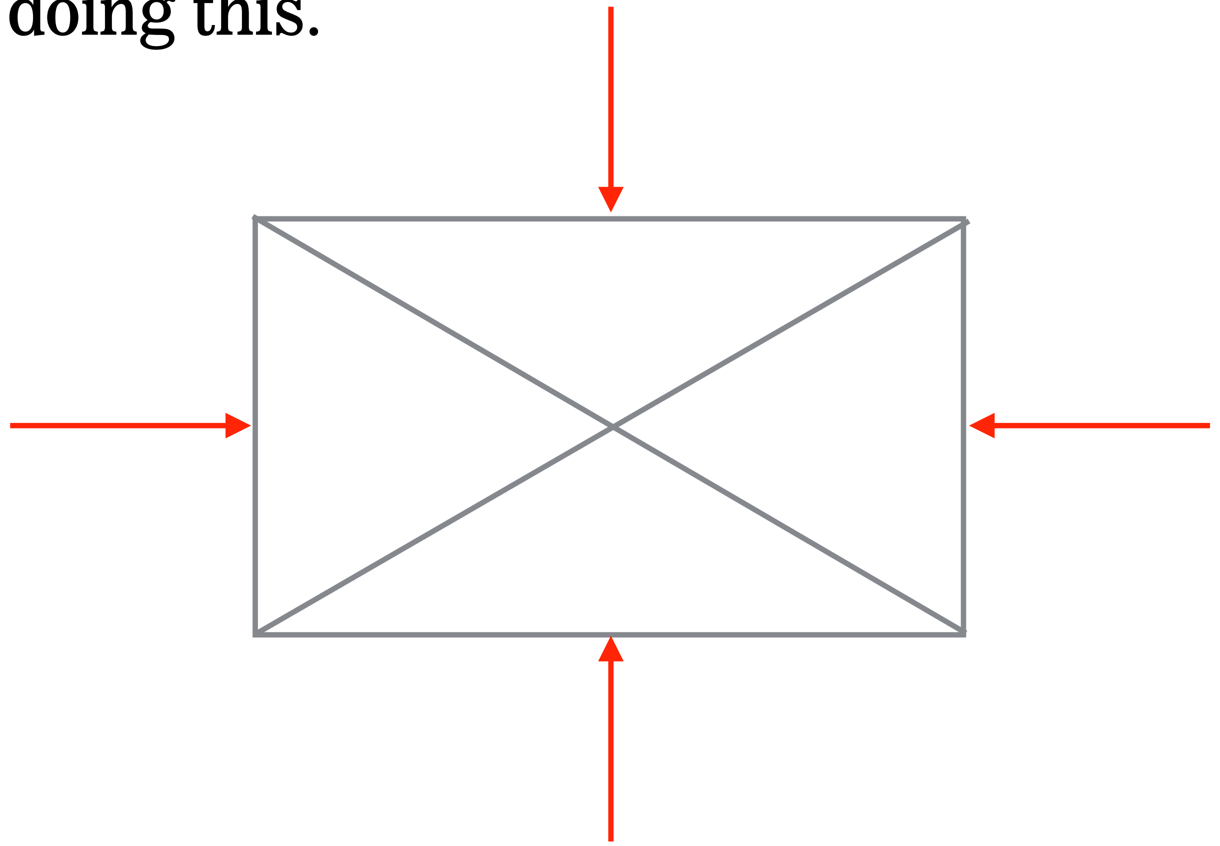
- First view + attribute: MyImageView.Width
- Relation: Equal
- Second view + attribute: MySuperView.Width
- Multiplier: 1:2
- Constant: 0



- What does it mean?
  - The width of the image should be 50% of the width of its container.

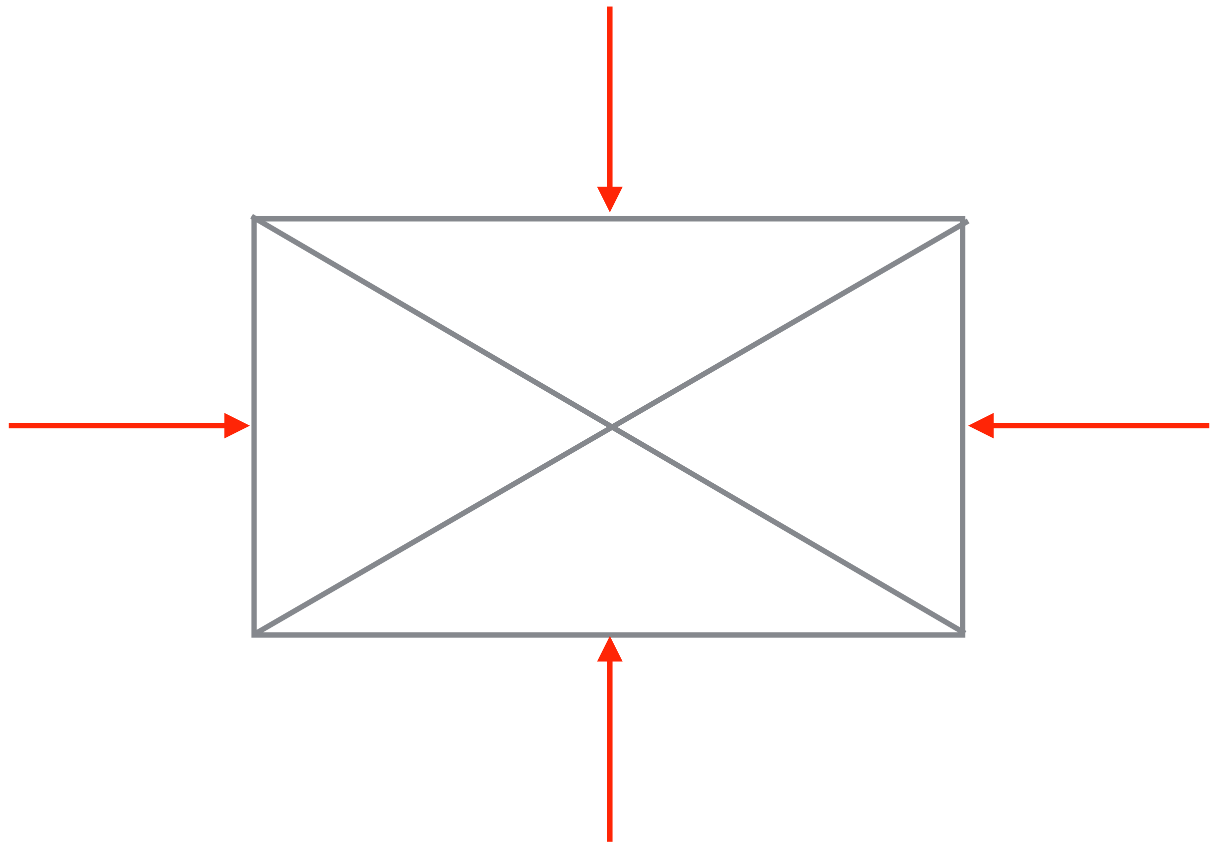
SUFFICIENT CONSTRAINT

- In order to describe the position and size of a view fully, you need to provide sufficient constraints to describe the position of every edge of the view. There are a few ways of doing this.



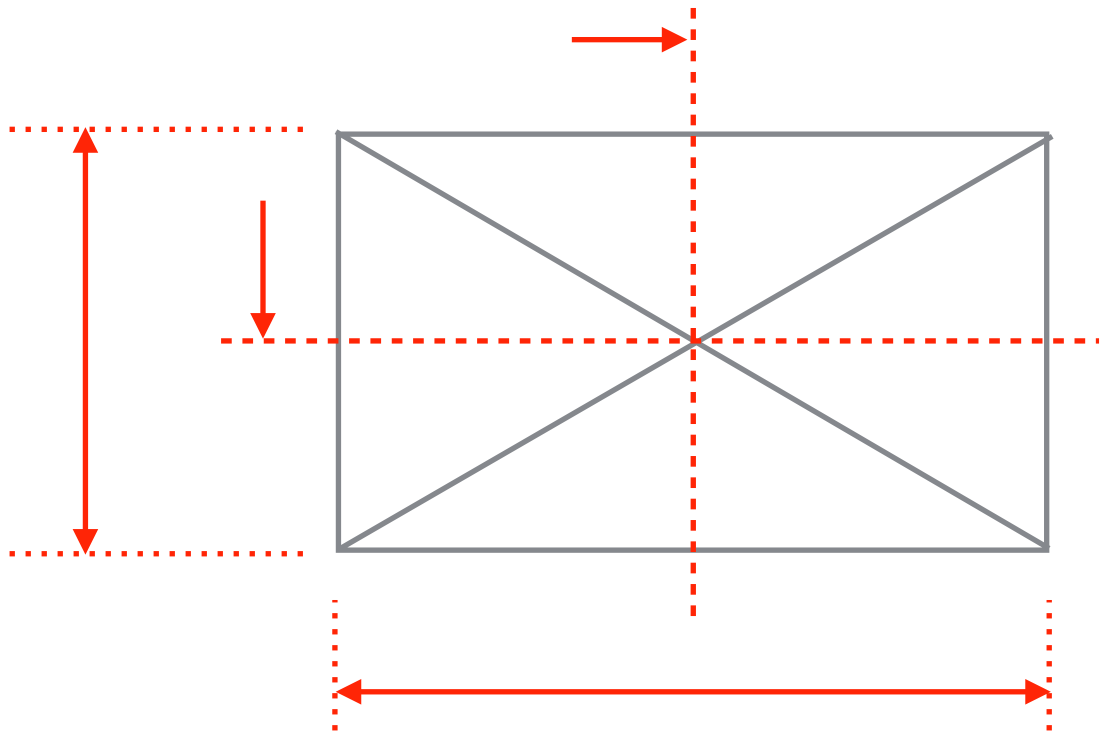
CONSTRAINT EXAMPLES

- Leading edge – x-position
- Trailing edge – x-position
- Top edge – y-position
- Bottom edge – y-position



CONSTRAINT EXAMPLES

- Horizontal center – x-position
- Vertical center – x-position
- Width
- Height

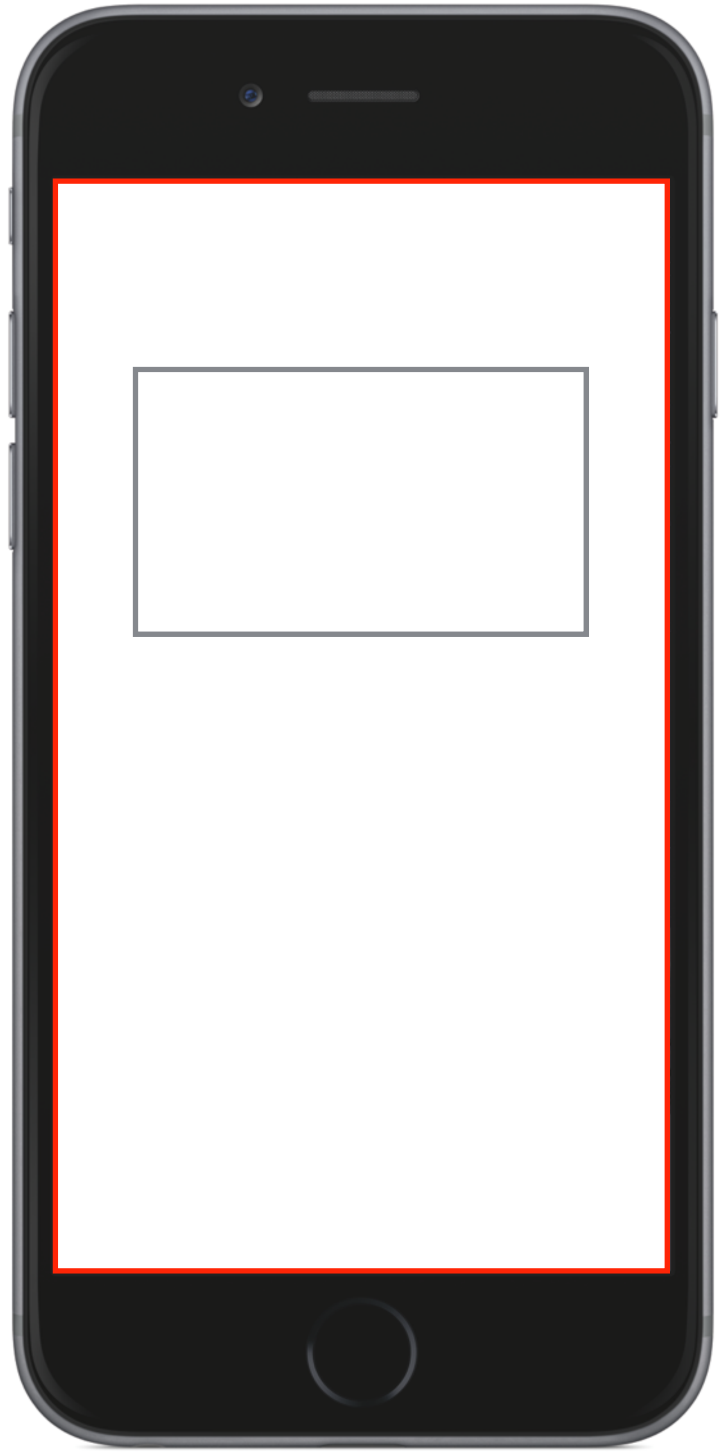


CONSTRAINT DEPENDENCIES

We need some frame of reference to start describing constraints. We can't just have them all relative to some other view that's also relative to other views.

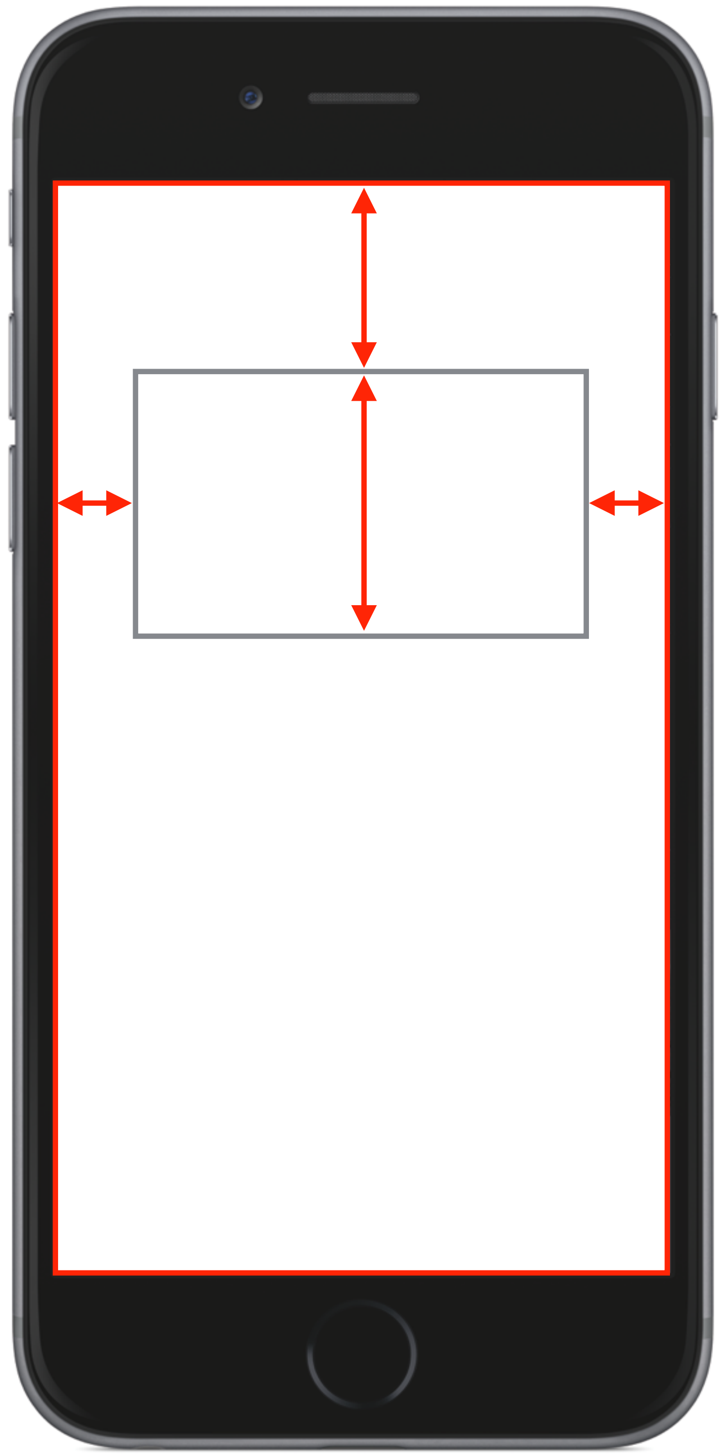
There are two places where we get this frame of reference:

- The top-most view of the View Controller.
- Absolute constraints (e.g. width = 300 points)



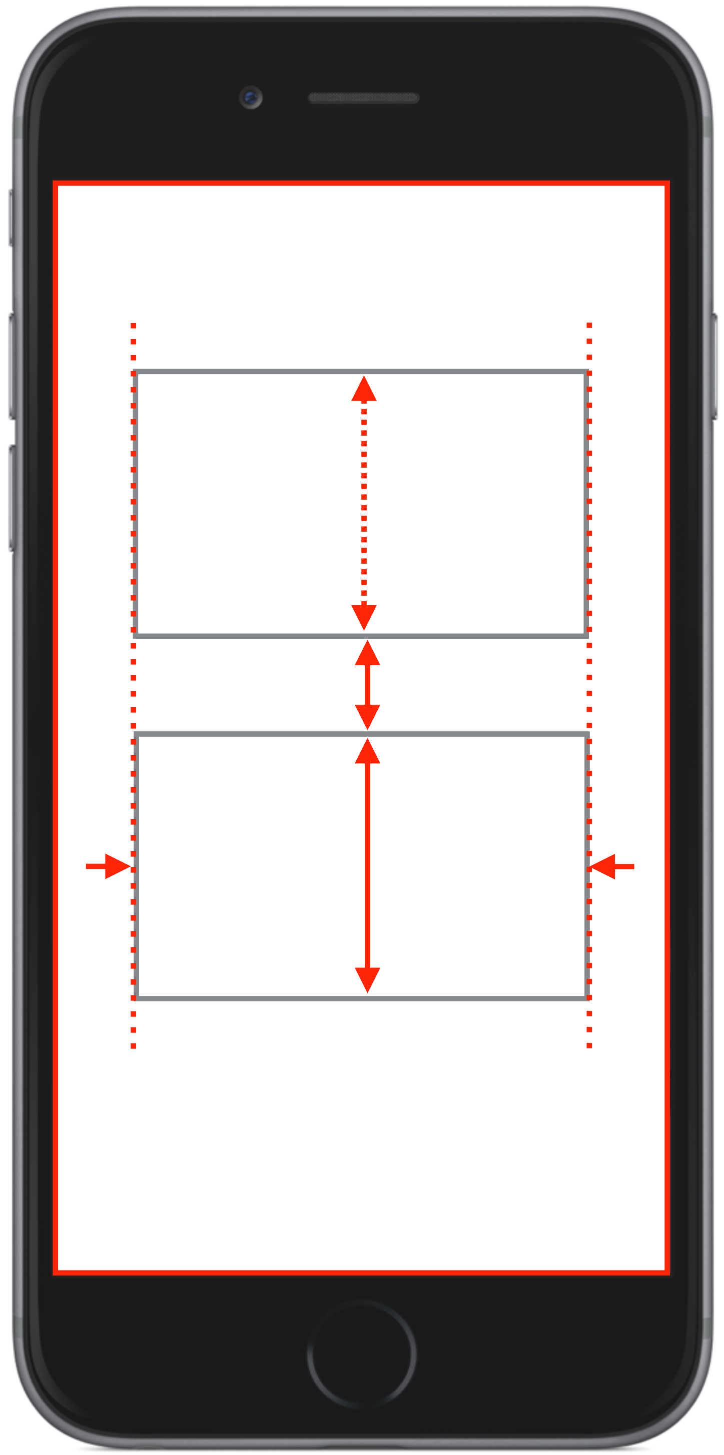
CONSTRAINT DEPENDENCIES

Thus, the first views we place generally are relative to the View Controller's bounds and other absolute ones, if necessary.



CONSTRAINT DEPENDENCIES

Then, once the first views are placed and have sufficient constraints, we can place more views relative to the first ones.

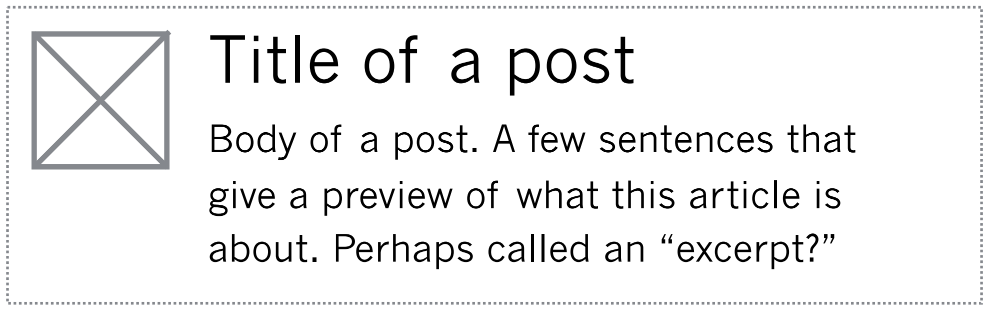


CONSTRAINTS EXERCISE

DESCRIBE WITH CONSTRAINTS

.Left, .Right, .Top, .Bottom
.Width, .Height
.CenterX, .CenterY

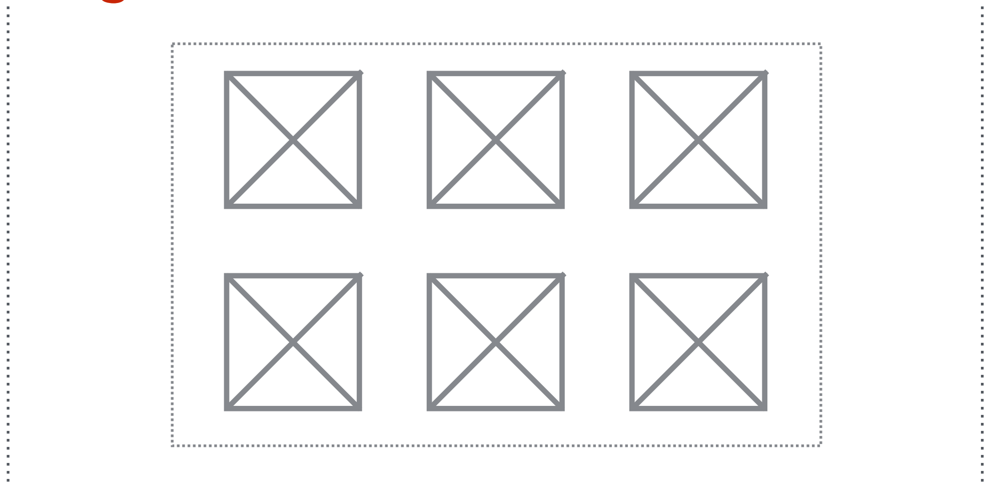
Custom Table Cell View



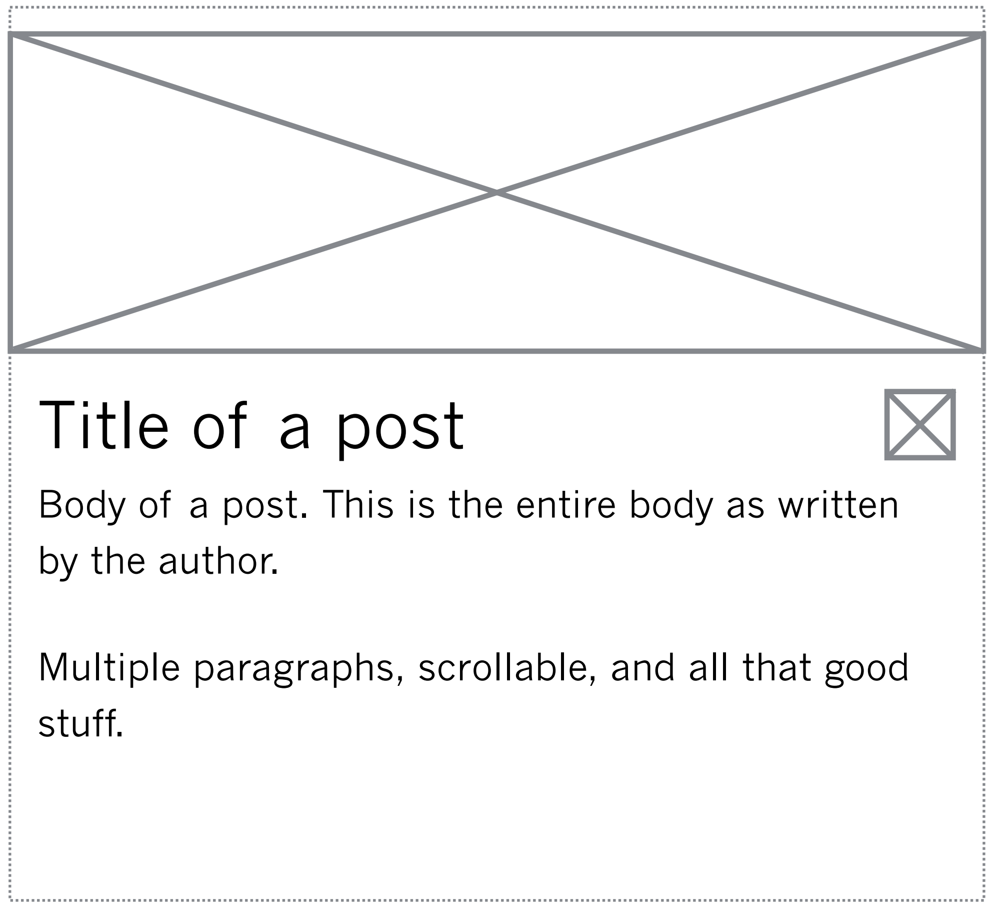
Title of a post

Body of a post. A few sentences that give a preview of what this article is about. Perhaps called an “excerpt?”

Tag Picker View



Detail View



Title of a post

Body of a post. This is the entire body as written by the author.

Multiple paragraphs, scrollable, and all that good stuff.

AUTO LAYOUT IN IB

CREATING CONSTRAINTS

We can create and adjust *constraints* in Interface Builder to create these dynamic layouts without using code.

WARNINGS

Some warnings about IB and setting constraints:

- IB is dumb, and its behaviors are sometimes counterintuitive. *So every time you create a constraint, check it and double-check it!*
- IB will warn about conflicting/insufficient constraints while you author constraints, but this isn’t always 100% comprehensive. It's still possible to create layout logic that doesn't work correctly.

CONSTRAINT ERRORS

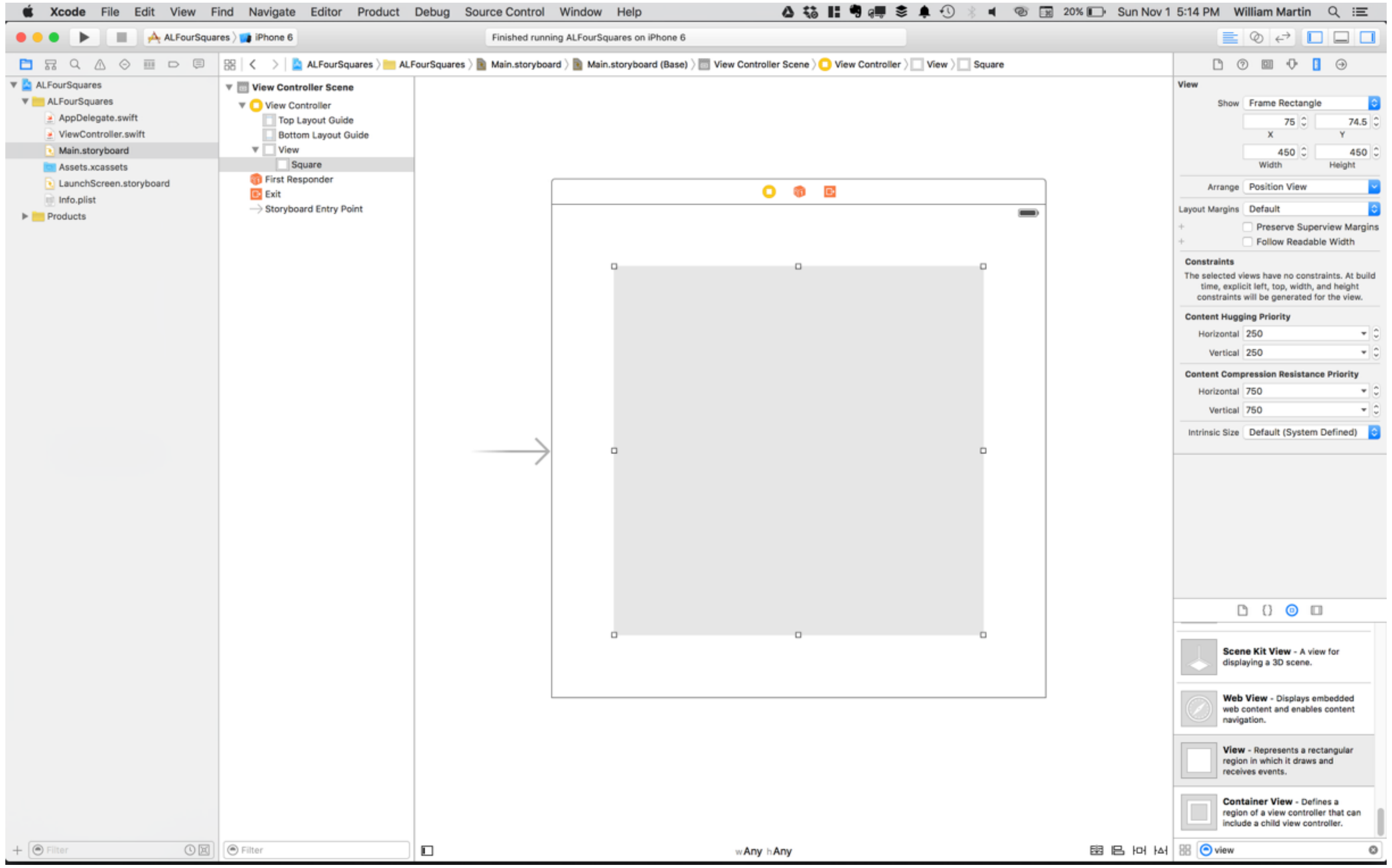
- Two common types of errors:
  - *Insufficient* constraints: We have too few constraints on at least one view, so the solver cannot figure out X, Y, width, or height.
  - *Conflicting* constraints: We have constraints on at least one view that result in *different* solutions for X, Y, width, or height.
- You should fix both of these when you see them, even if visual bugs to do not result from them.

AUTO LAYOUT IN IB WALKTHROUGH

AUTO LAYOUT

SETTING A SINGLE CONSTRAINT

A View Controller in IB typically starts as a square. We'll place Views in a general, template-like way that doesn't necessarily reflect the actual proportions that we want.


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Sun Nov 1 5:14 PM William Martin
ALFourSquares › iPhone 6
Finished running ALFourSquares on iPhone 6
ALFourSquares › ALFourSquares › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Square
ALFourSquares
ALFourSquares
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Square
First Responder
Exit
Storyboard Entry Point
wAny hAny
Scene Kit View - A view for displaying a 3D scene.
Web View - Displays embedded web content and enables content navigation.
View - Represents a rectangular region in which it draws and receives events.
Container View - Defines a region of a view controller that can include a child view controller.
view

View

Show: Frame Rectangle
X: 75 Y: 74.5
Width: 450 Height: 450
Arrange: Position View
Layout Margins: Default
- [ ] Preserve Superview Margins
- [ ] Follow Readable Width

Constraints

The selected views have no constraints. At build time, explicit left, top, width, and height constraints will be generated for the view.

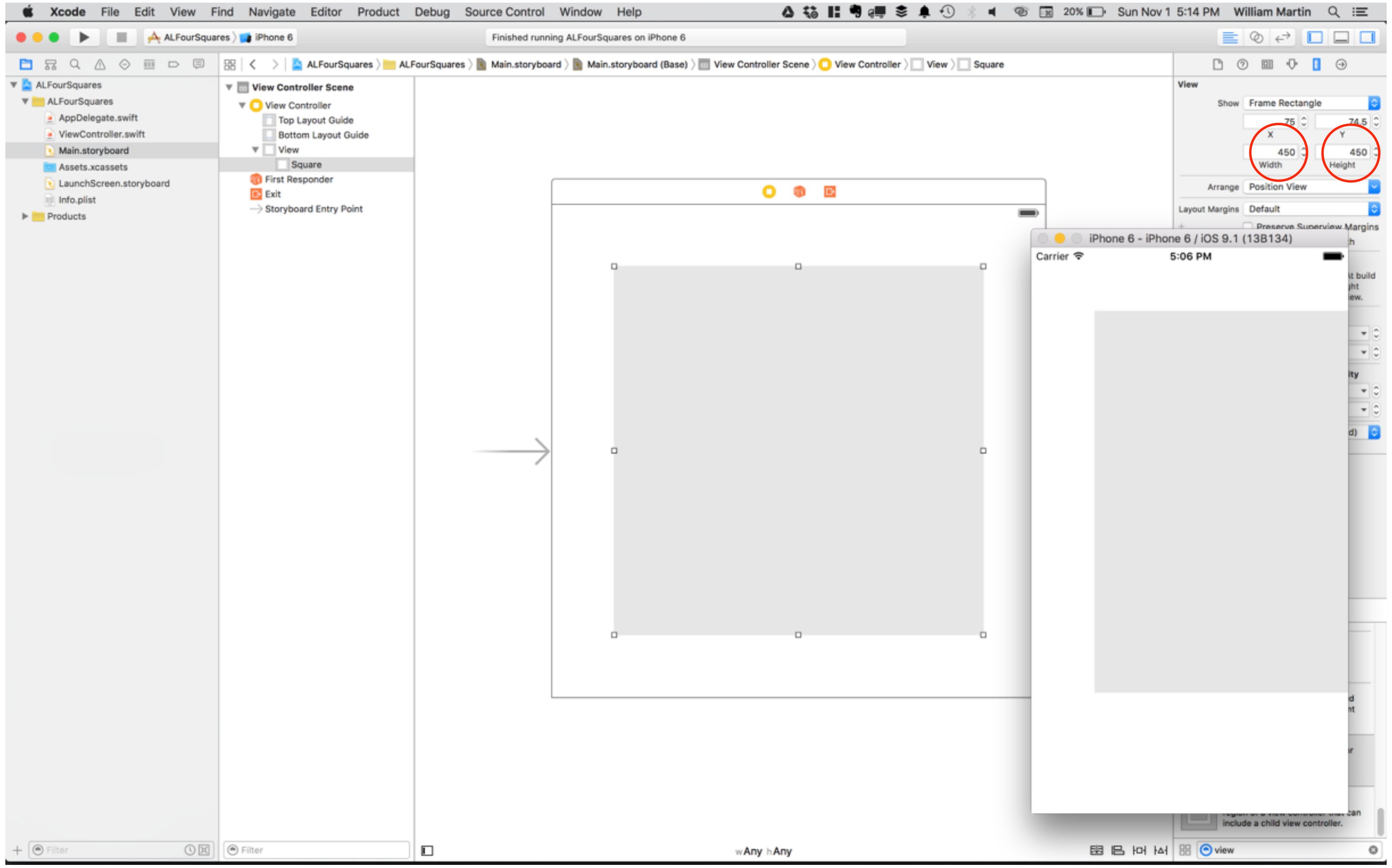
Content Hugging Priority
Horizontal: 250
Vertical: 250

Content Compression Resistance Priority
Horizontal: 750
Vertical: 750

Intrinsic Size: Default (System Defined)

SETTING A SINGLE CONSTRAINT

Place a single UIView as a square, but note that when the app is run, the actual point dimensions of the View are maintained but don't respect the proportions of the simulator.

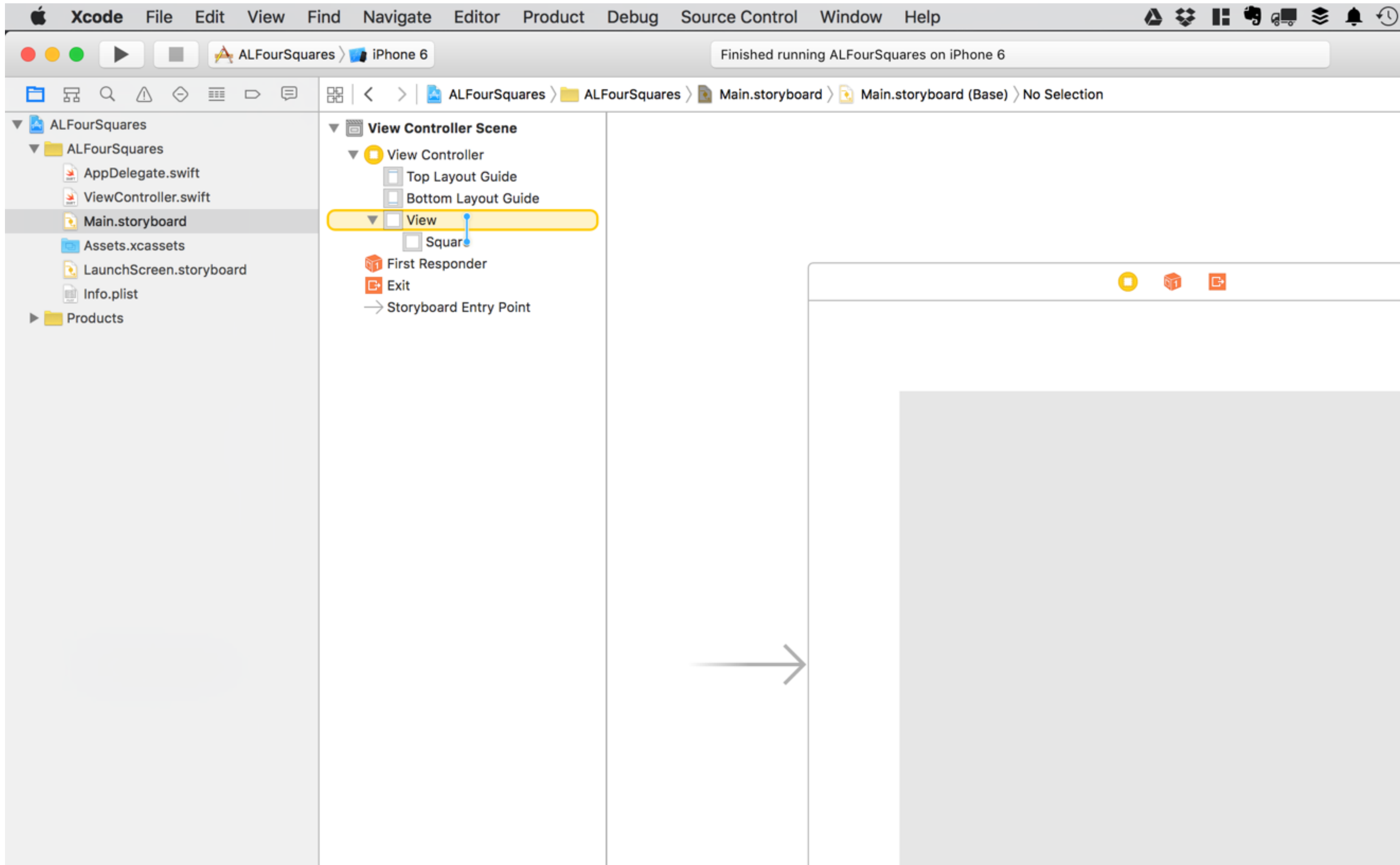

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Sun Nov 1 5:14 PM William Martin
ALFourSquares iPhone 6
Finished running ALFourSquares on iPhone 6
ALFourSquares > ALFourSquares > Main.storyboard > Main.storyboard (Base) > View Controller Scene > View Controller > View > Square
ALFourSquares
ALFourSquares
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Square
First Responder
Exit
Storyboard Entry Point
iPhone 6 - iPhone 6 / iOS 9.1 (13B134)
Carrier
5:06 PM
wAny hAny
Filter
view

View
Show: Frame Rectangle
X: 75
Y: 74.5
Width: 450
Height: 450
Arrange: Position View
Layout Margins: Default
Preserve Superview Margins

SETTING A SINGLE CONSTRAINT

There are several ways to set a constraint in IB.

1. Ctrl+Drag from one view to another in the Document Outline.

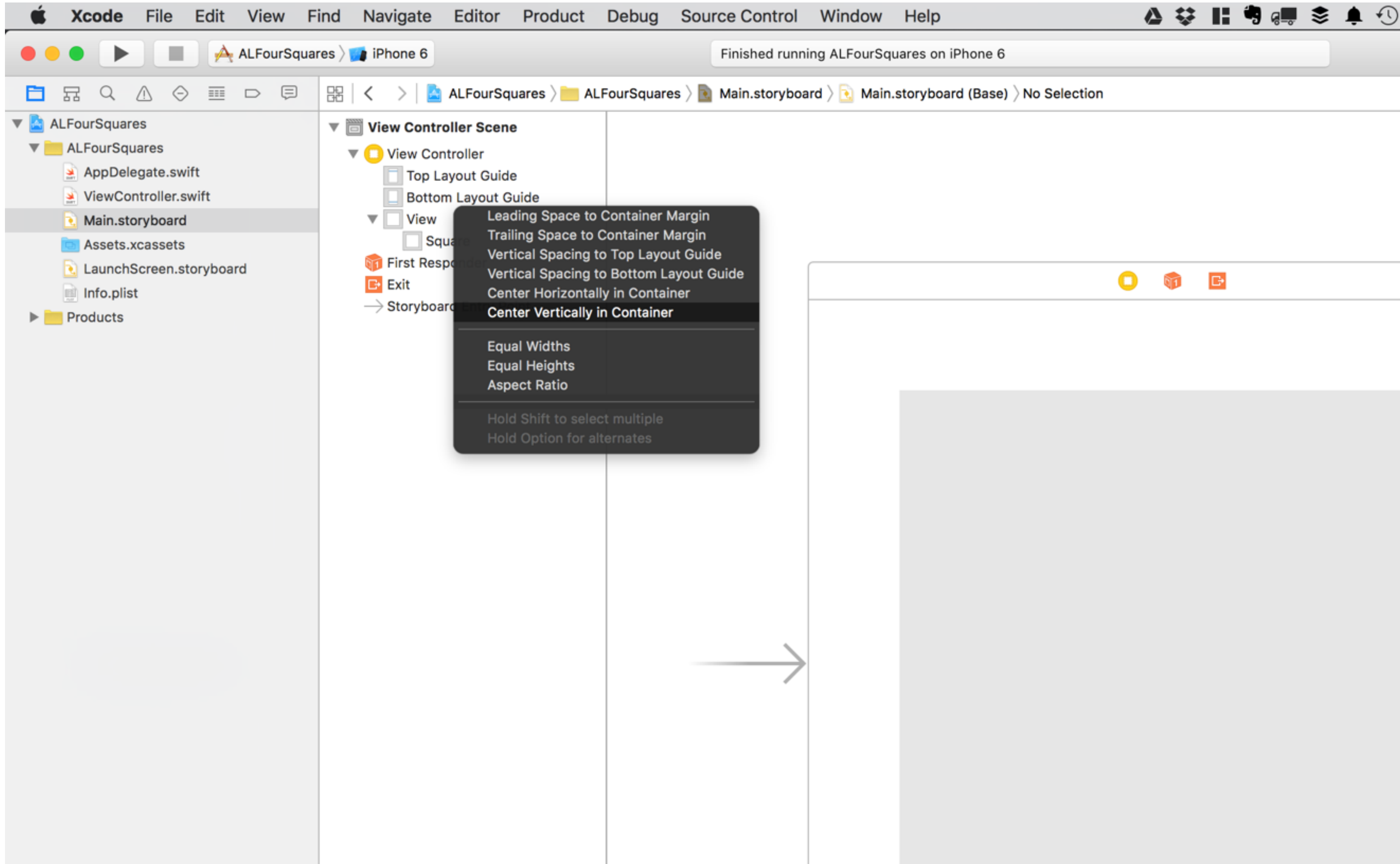


SETTING A SINGLE CONSTRAINT

Right after, a black popover will appear asking you for the relation you want to set for the constraint.

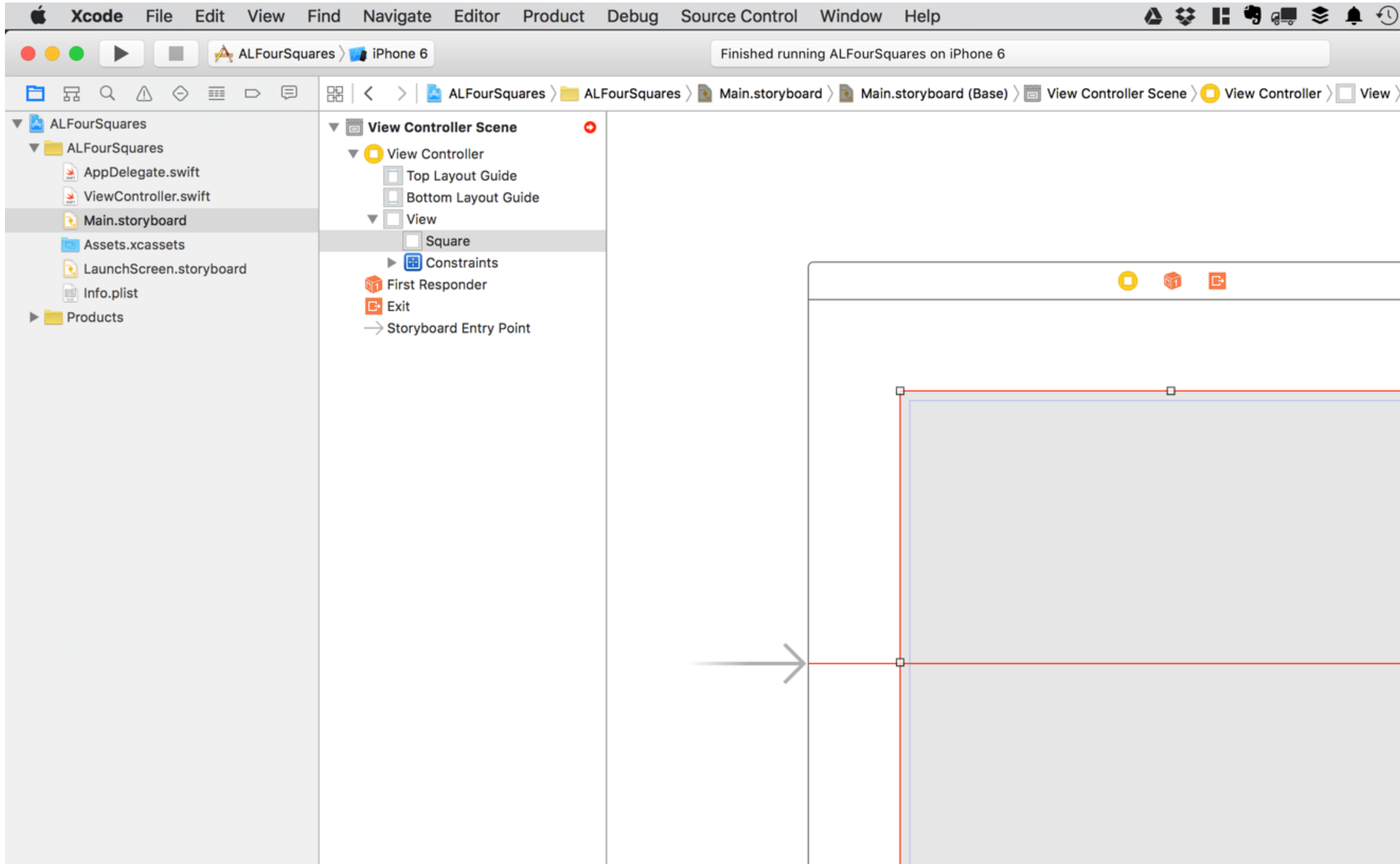
Remember, a constraint consists of two views, a relation, and some parameters.

Here, we select "Center Vertically in Container."



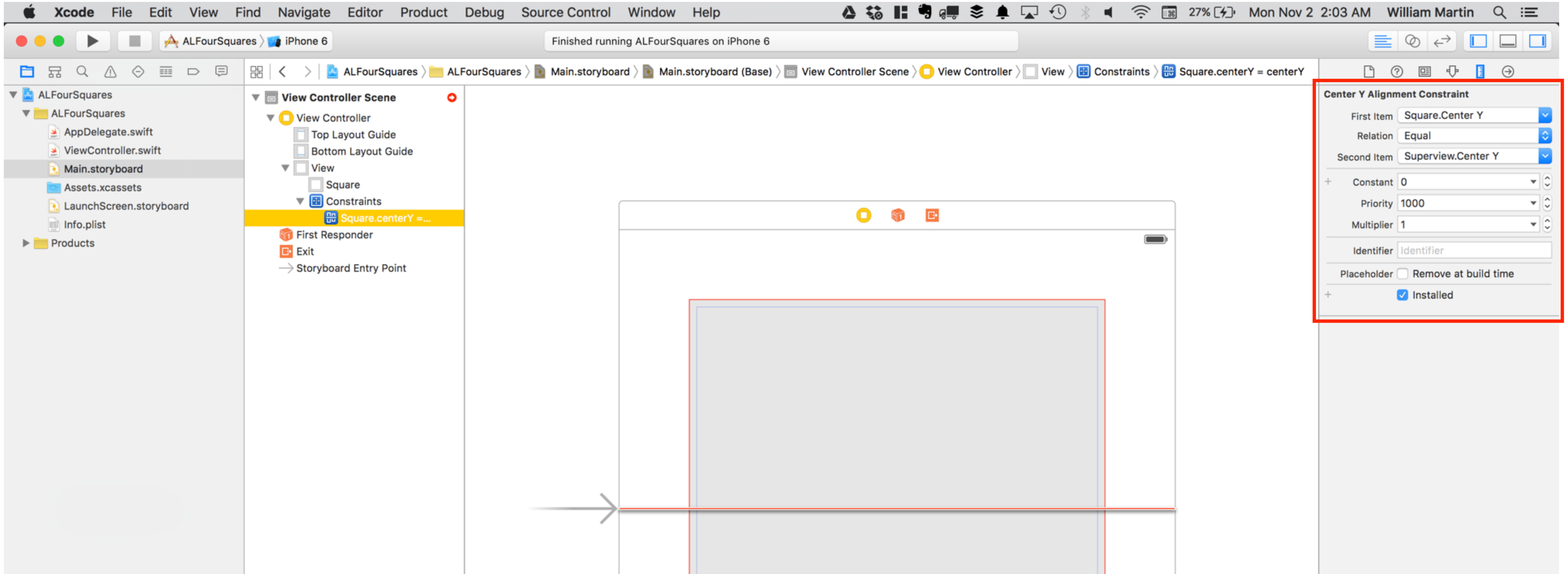
SETTING A SINGLE CONSTRAINT

The constraint is set, and selecting the view will reveal graphics showing the constraint on the canvas.



SETTING A SINGLE CONSTRAINT

Select the constraint, and the Size Inspector will show its parameters.


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
27% Mon Nov 2 2:03 AM William Martin
ALFourSquares > iPhone 6
Finished running ALFourSquares on iPhone 6
ALFourSquares > ALFourSquares > Main.storyboard > Main.storyboard (Base) > View Controller Scene > View Controller > View > Constraints > Square.centerY = centerY
ALFourSquares
ALFourSquares
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Square
Constraints
Square.centerY =...
First Responder
Exit
Storyboard Entry Point

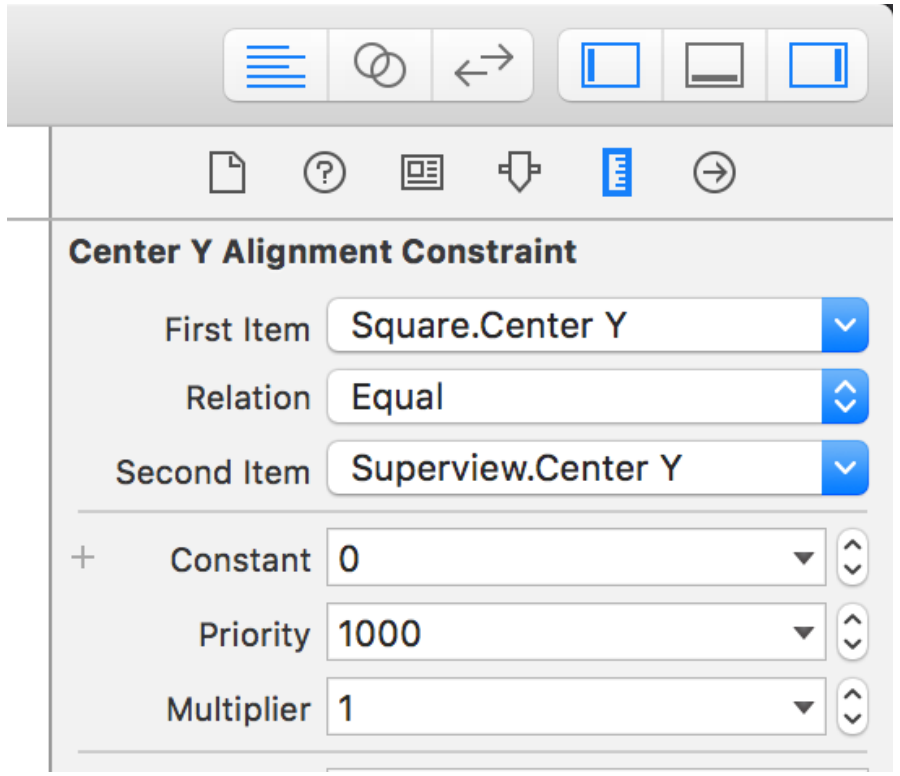
Center Y Alignment Constraint

First Item	Square.Center Y
Relation	Equal
Second Item	Superview.Center Y
Constant	0
Priority	1000
Multiplier	1
Identifier	Identifier
Placeholder	☐ Remove at build time
	☑ Installed

SETTING A SINGLE CONSTRAINT

The parameters describe the nature of the constraint. In this case, we're talking about centering the gray square vertically on the available screen.

Thus, the y-position of the centers of both views should be equal, with an offset of 0 points.



Center Y Alignment Constraint

First Item	Square.Center Y
Relation	Equal
Second Item	Superview.Center Y
Constant	0
Priority	1000
Multiplier	1