ITM103 iOS Application Development

Topic 7: Adaptive UIs

Objectives

- By the end of the lesson, you will be able to:
  - Understand Auto Layout
  - Understand Stack Views
  - Understand Trait Variations

Adaptive UIs

Auto Layout

Auto Layout

- Auto Layout is a mechanism for automatically positioning and resizing the UI elements in your views.

- Allows developers to create an adaptive interface that responds appropriately to changes in screen size and device orientation

Auto Layout

- Fundamental building block in Auto Layout is the constraint.

- Constraint express rules for the layout of elements in your interface.

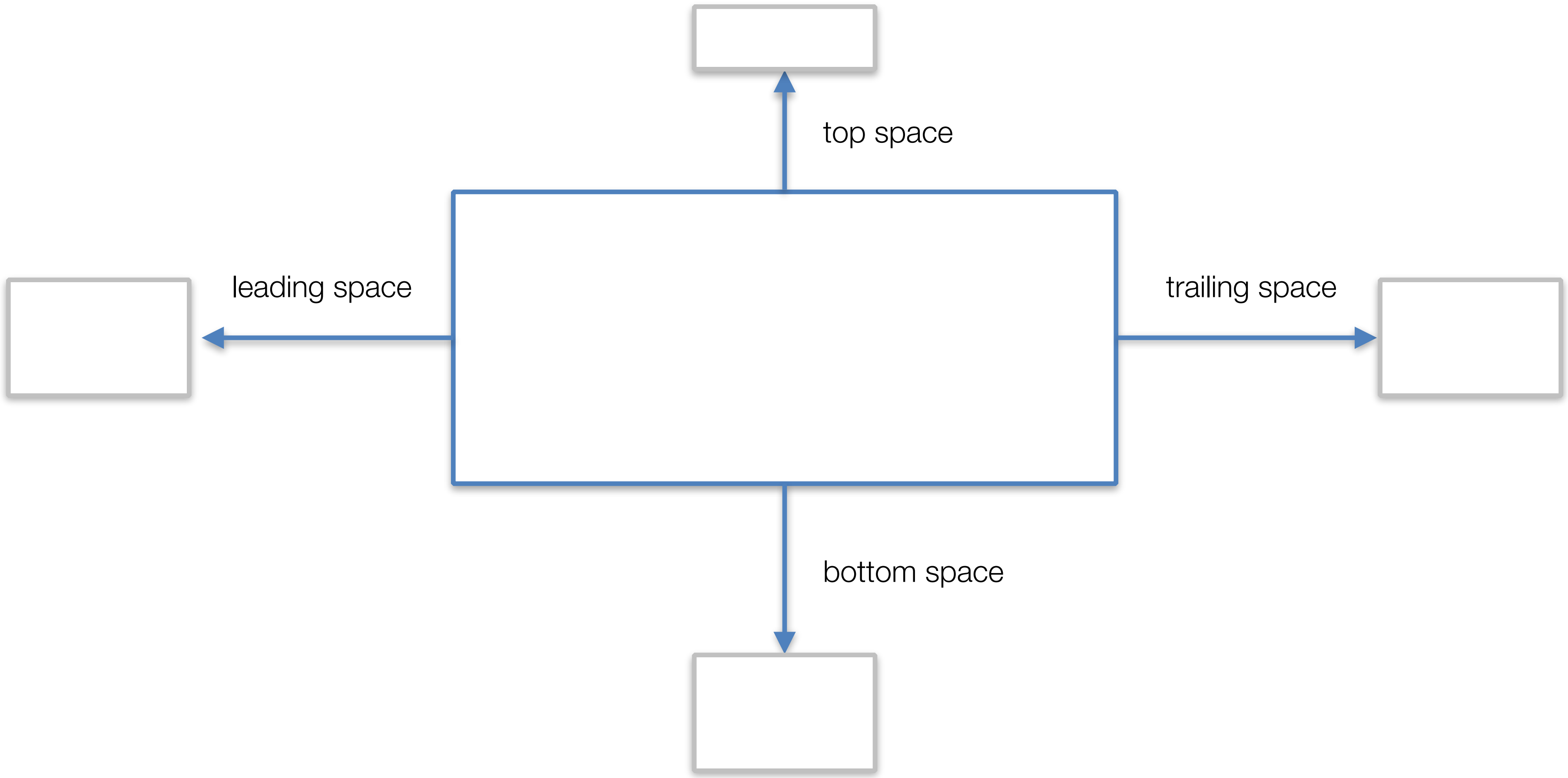
- Usually: Relationship between **views**

  One of them is usually the parent container or a same-level view

- Or: Fixed width / height

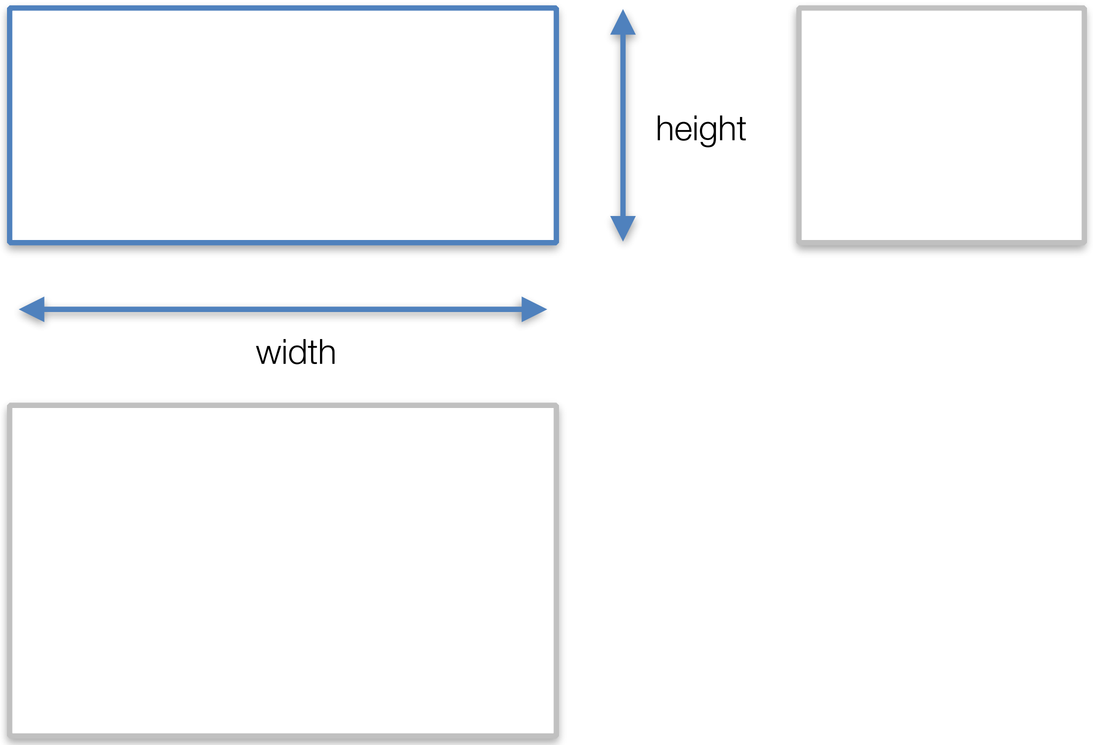
Common Constraints

- Distance to any view or edge of screen.



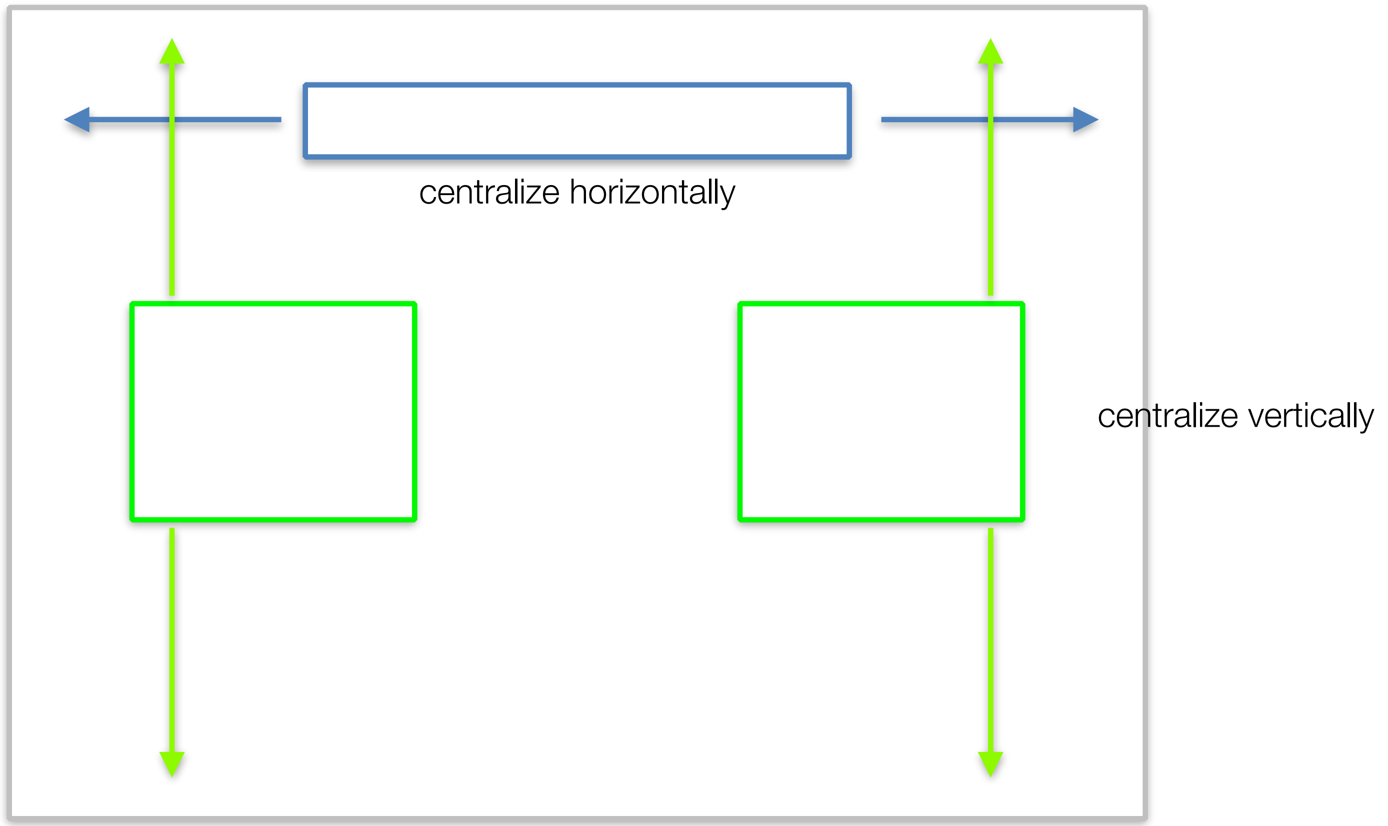
Common Constraints

- Equal width / height to another control
    - Usually still need additional constrain to position x, y coordinates.



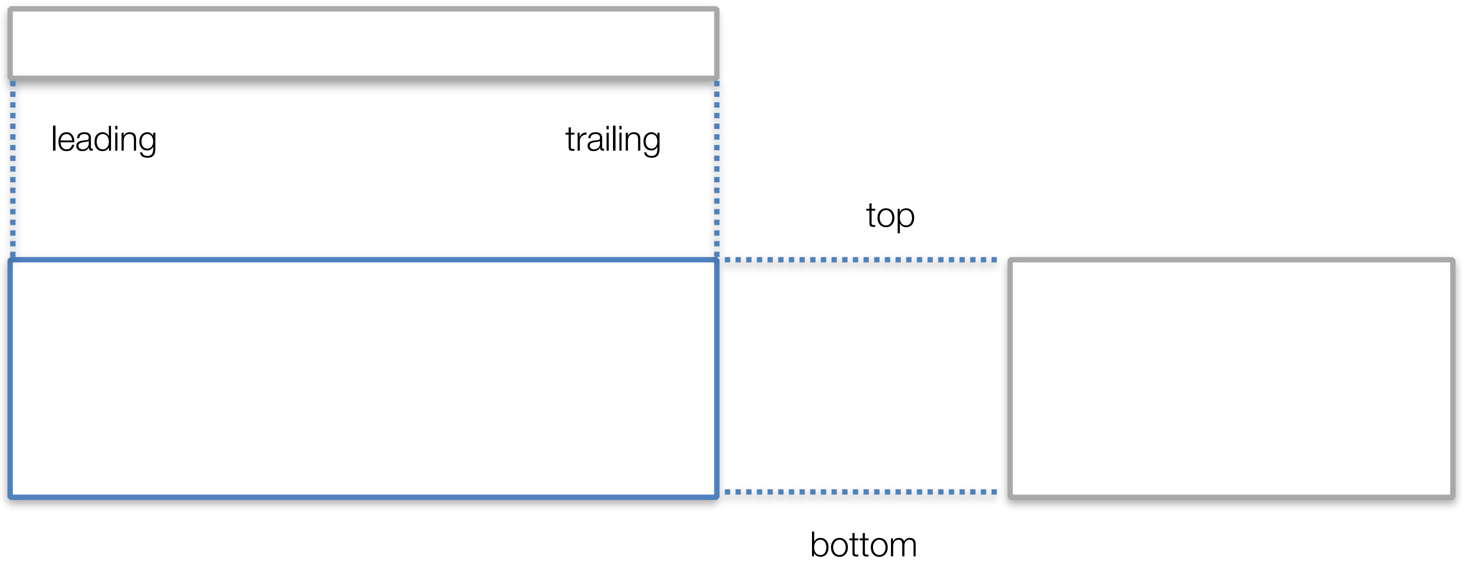
Common Constraints

- Centre-align horizontally / vertically inside container

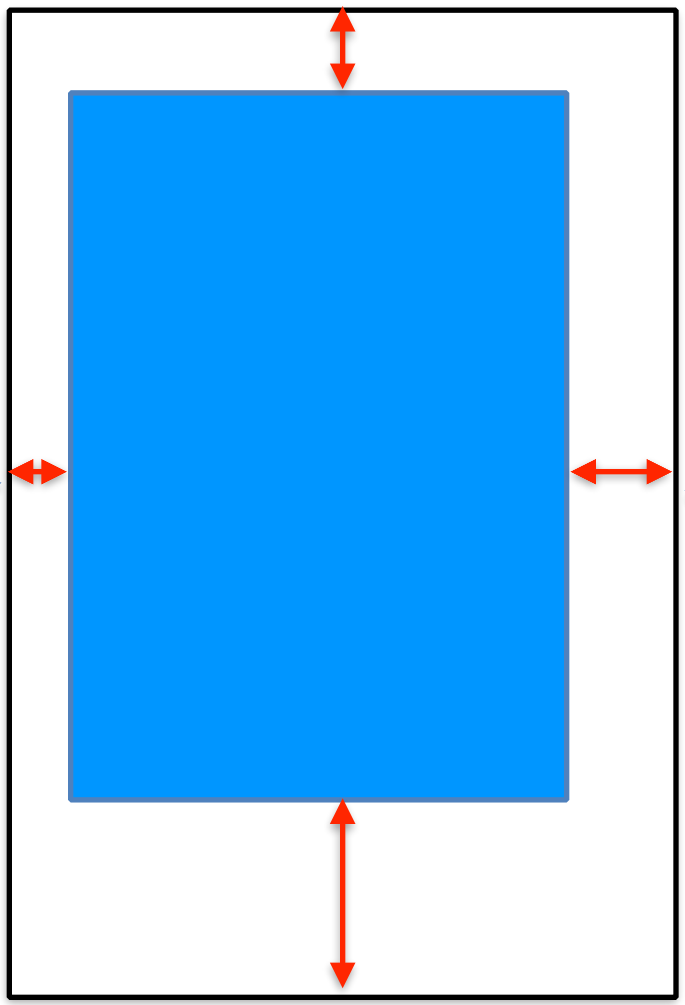


Common Constraints

- Align edge to another view



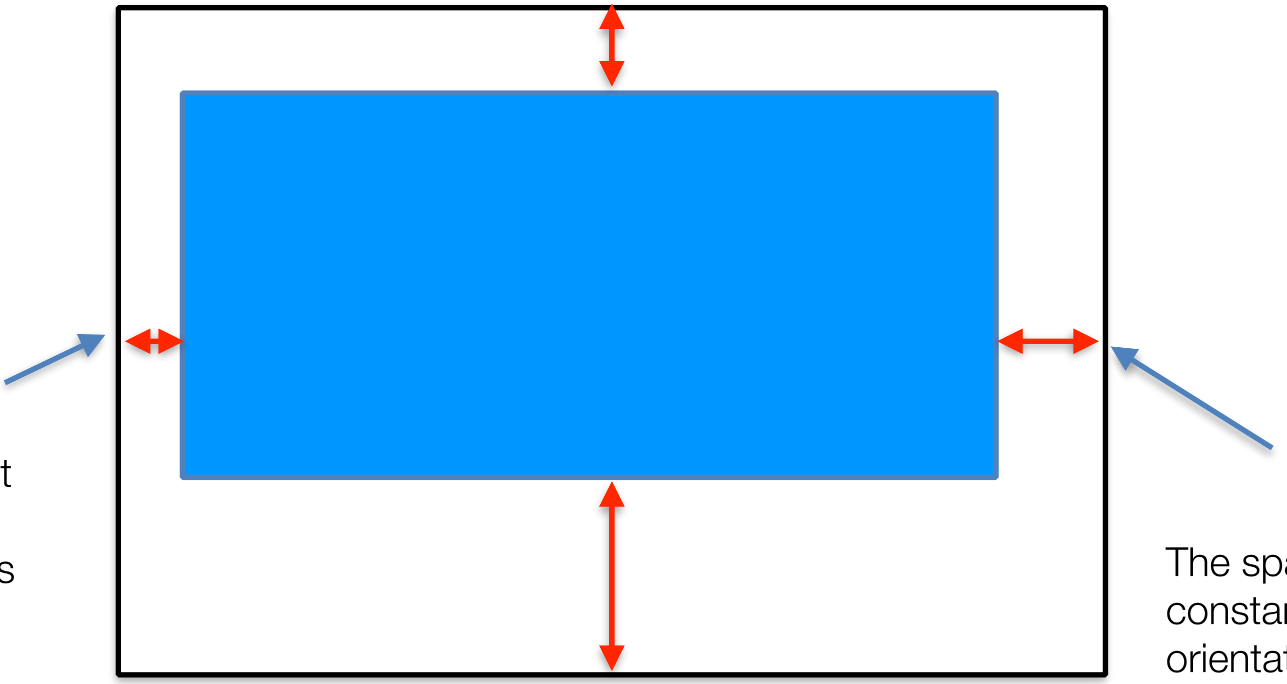
Examples



The spacing is kept constant as the orientation changes

The spacing is kept constant as the orientation changes

Examples

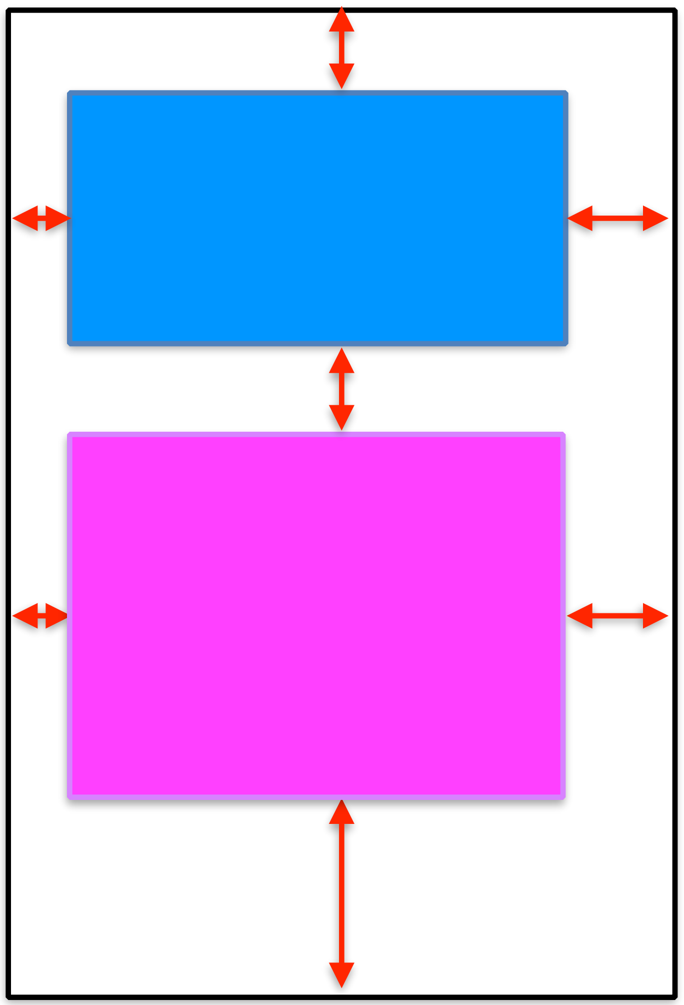


The spacing is kept constant as the orientation changes

The spacing is kept constant as the orientation changes

Examples - Ambiguous Case

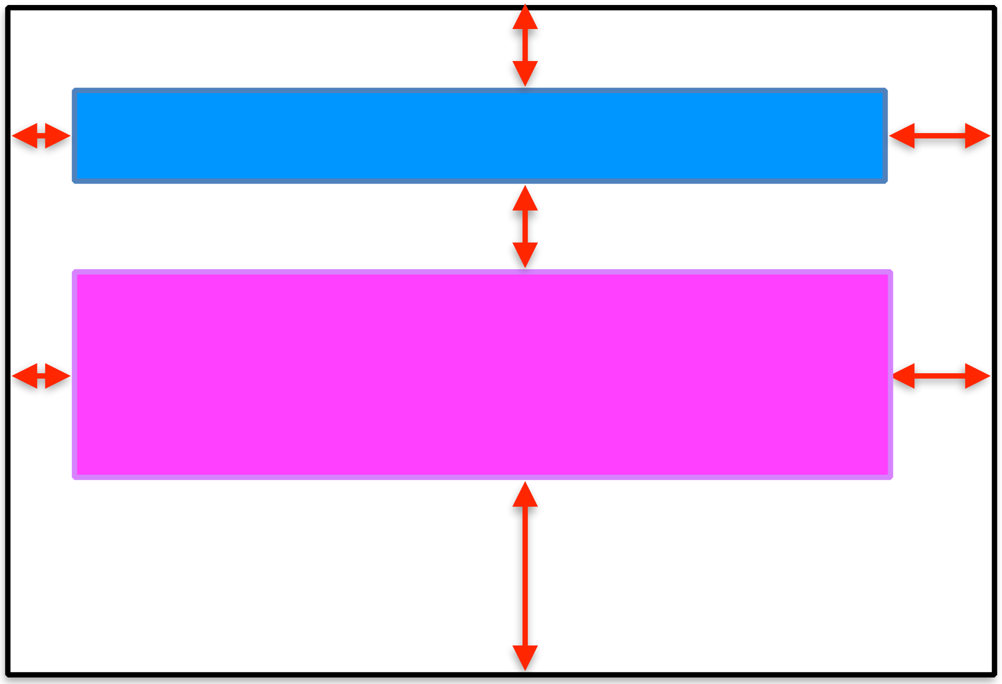
But this is an odd case…



Examples - Ambiguous Case

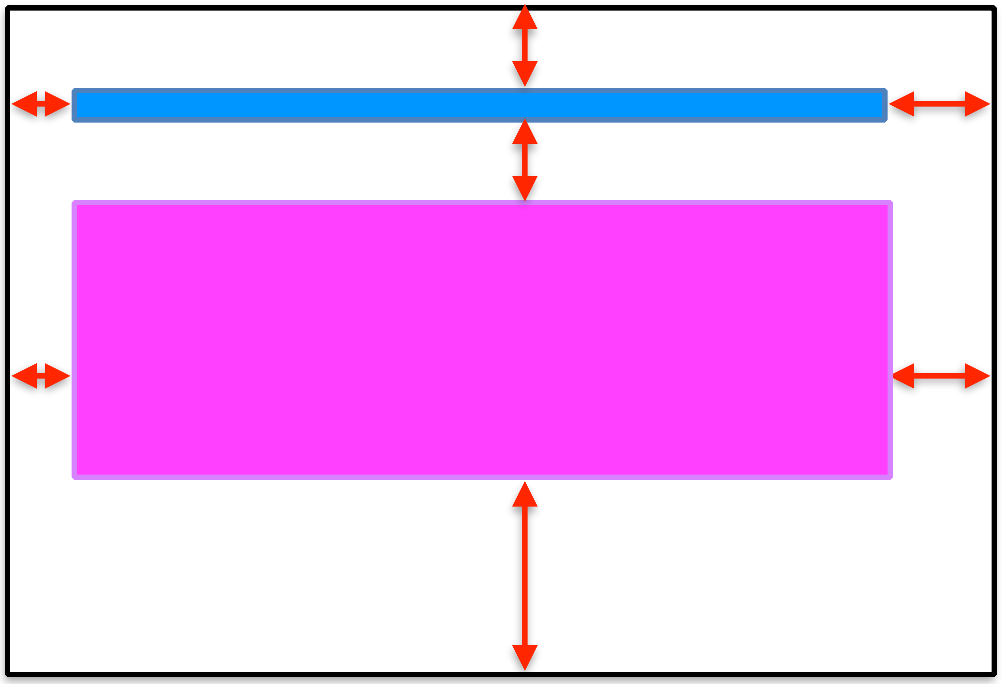
Do you see

why?



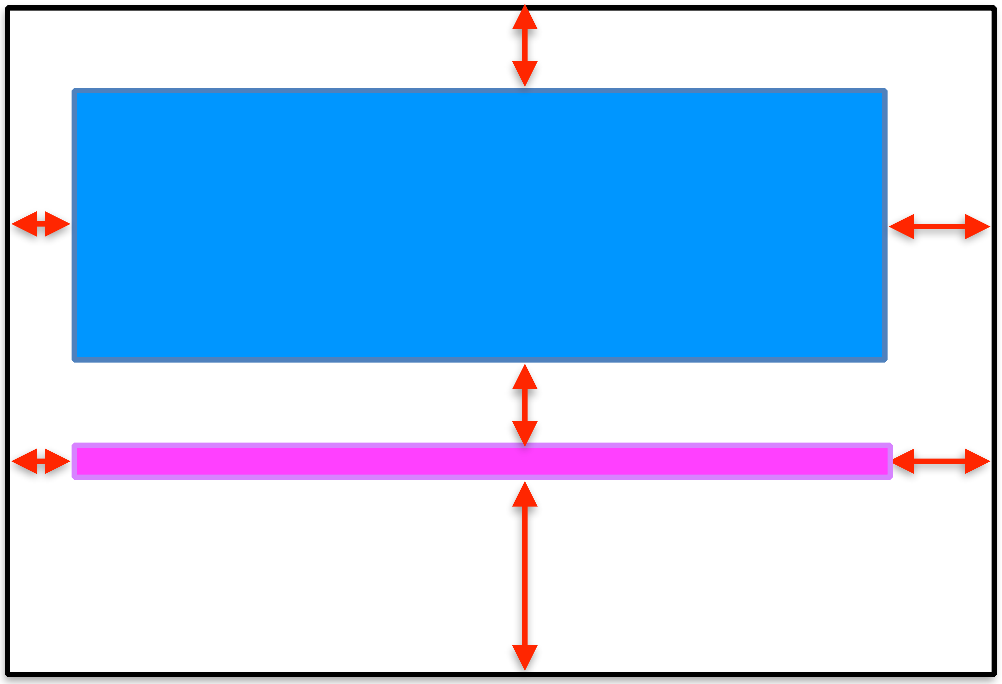
Examples - Ambiguous Case

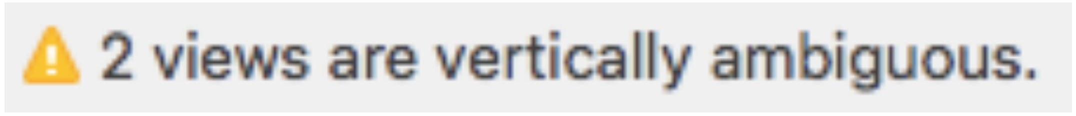
This layout **meets** the constraints



Examples - Ambiguous Case

This layout also **meets** the constraints

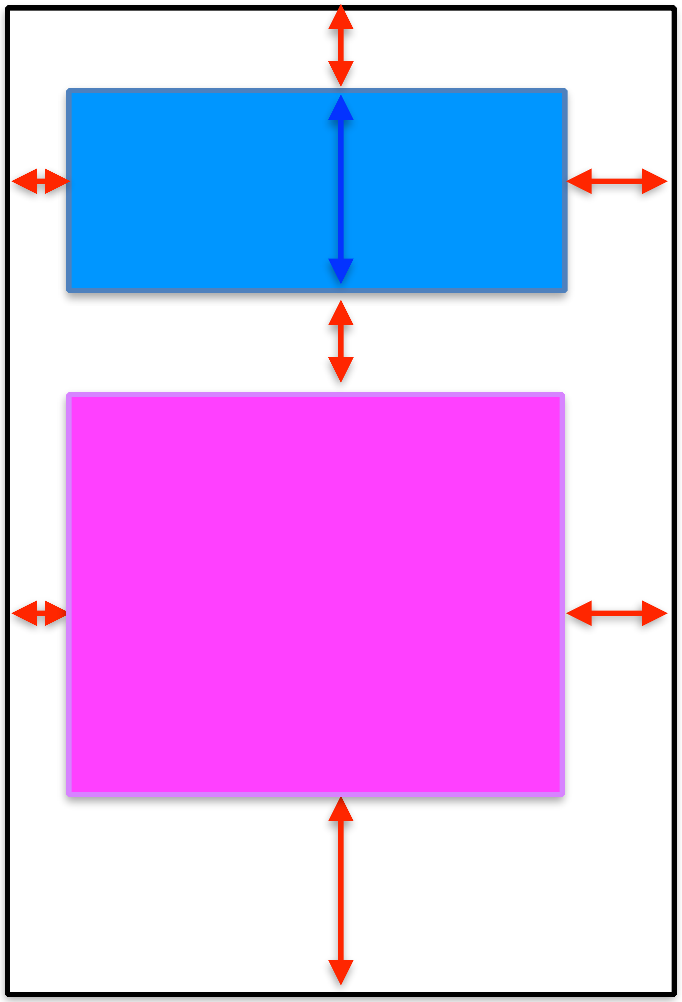




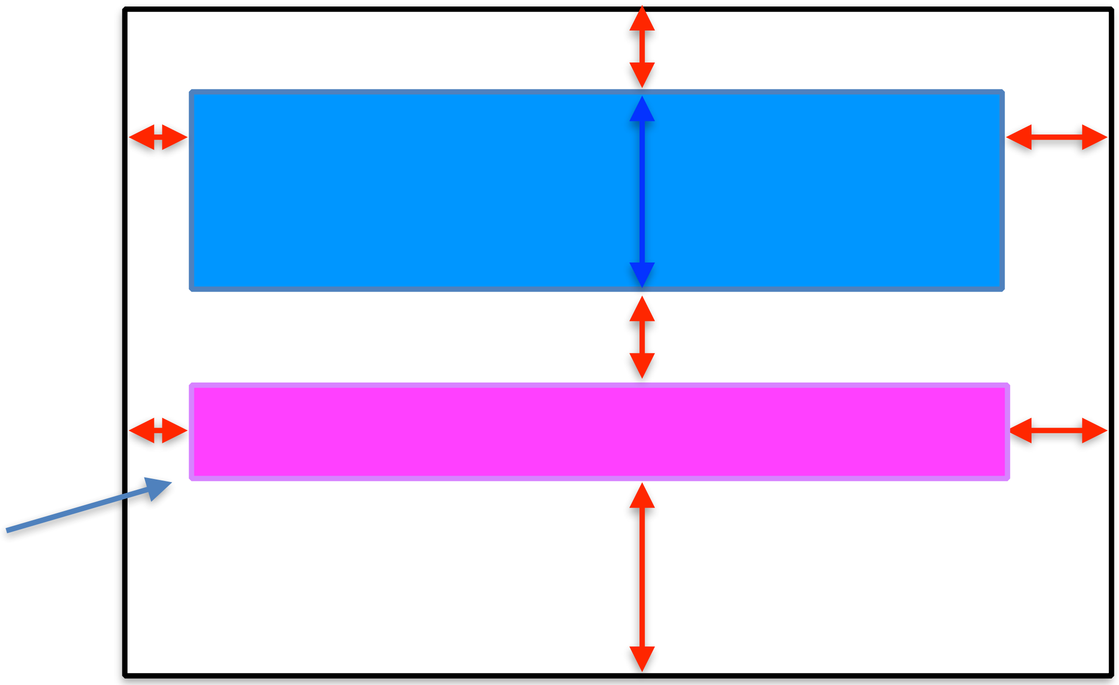
⚠ 2 views are vertically ambiguous.

Examples - Resolving the Ambiguous Case

Add one more constraint to resolve ambiguity



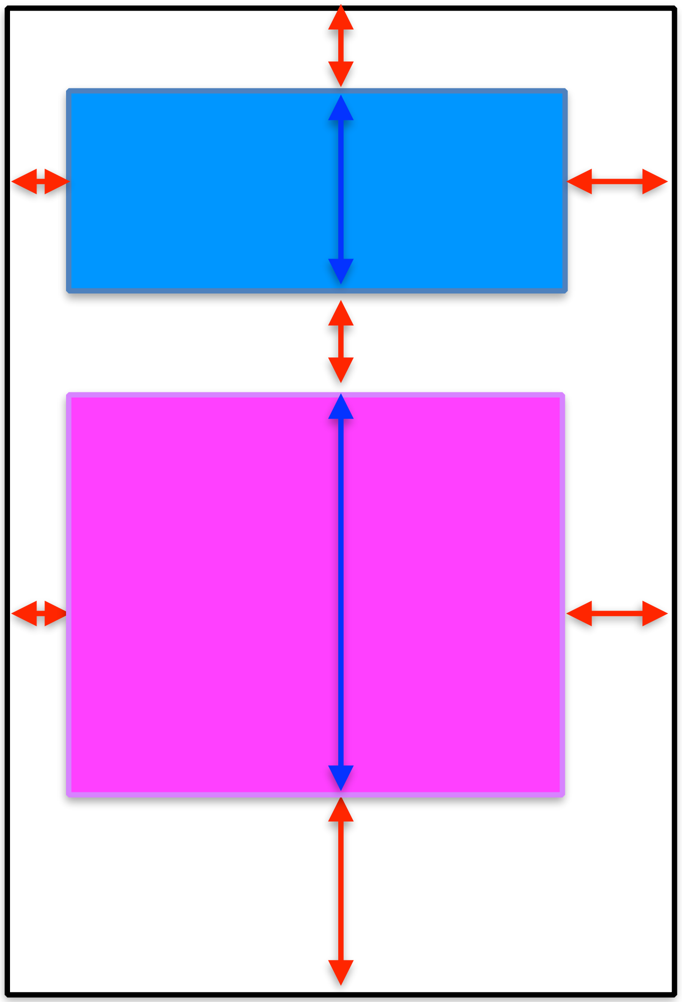
Examples - Resolving the Ambiguous Case



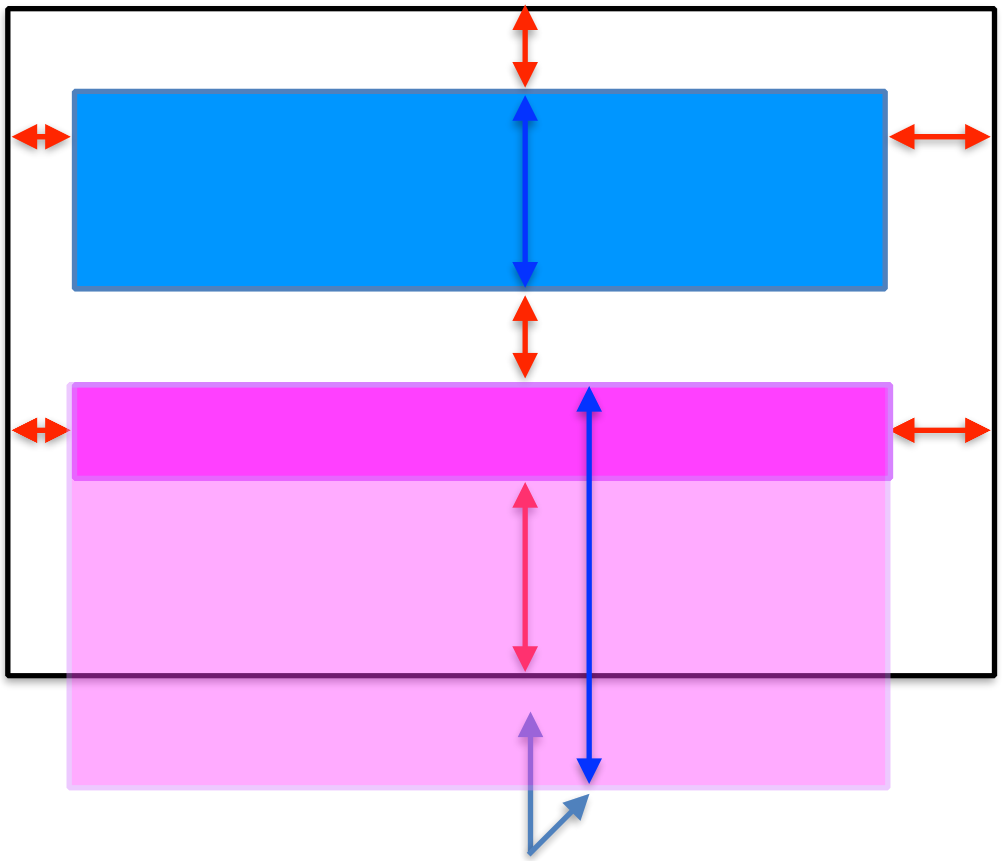
iOS just needs
to vary this
View's height

Examples - Conflicts

Now, there's too many constraints!

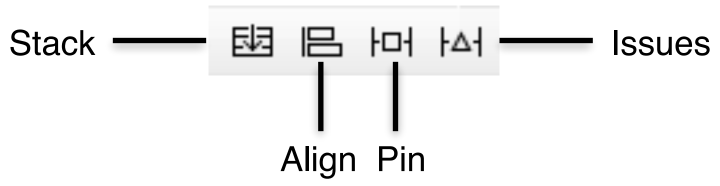


Examples - Conflicts



iOS will probably choose one of the constraints

Adding Constraints with Align and Pin Menus

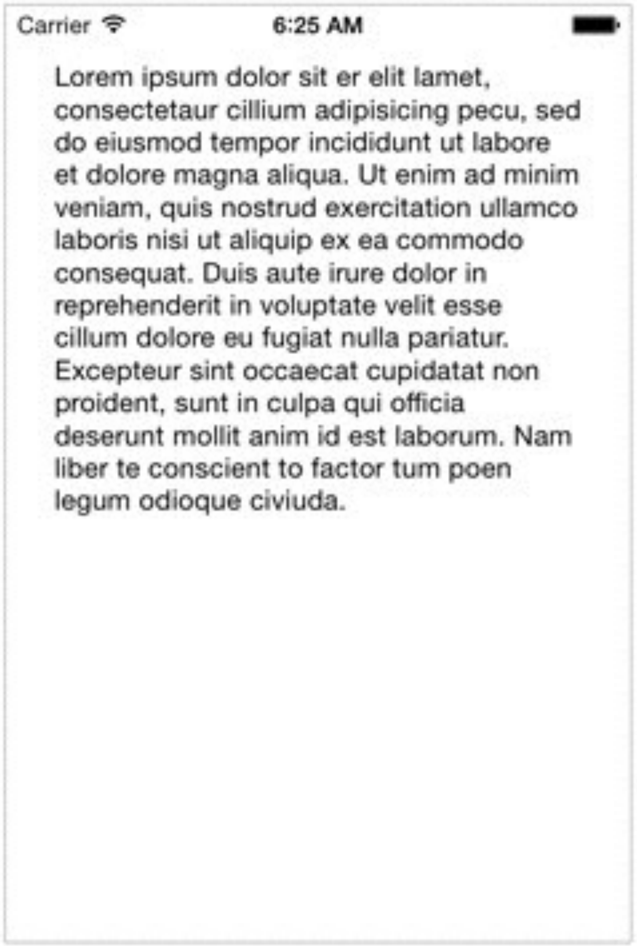


Menu	Description
Stack	Insert the selected view into a UIStackView
Align	Create alignment constaints, such as centering a view in its container, or aligning the left edges of two views
Pin	Create spacing constraints, such as defining the height of a view, or specifying its horizontal distance from another view.
Issues	Resolve the layout issues by adding or resetting constraints based on suggestion

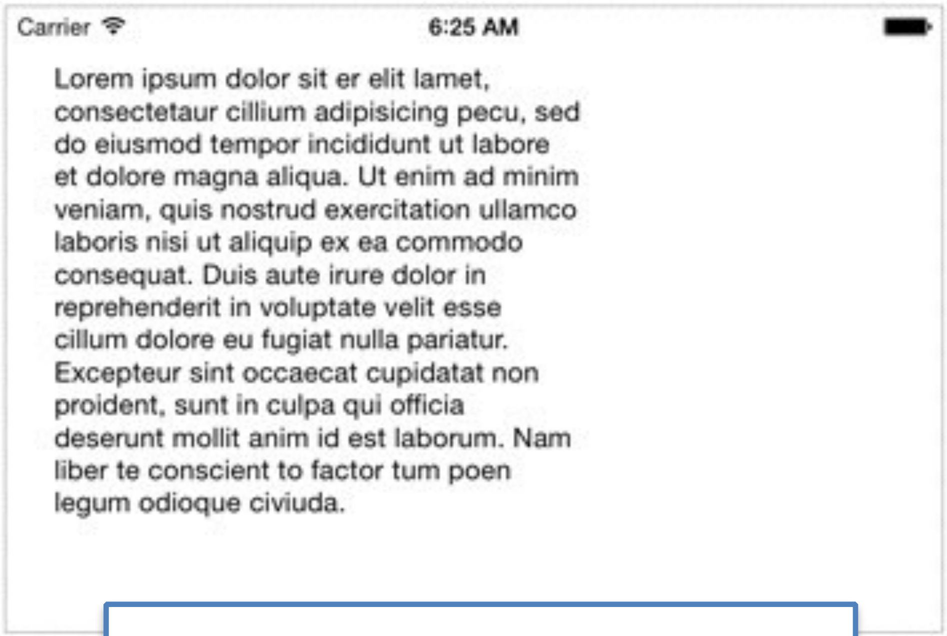
Adding Constraints

- Three common ways in Storyboard:
    - Align Button
    - Pin Button
    - Ctrl-drag from one View to another

Adding Constraints


Carrier 6:25 AM

Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

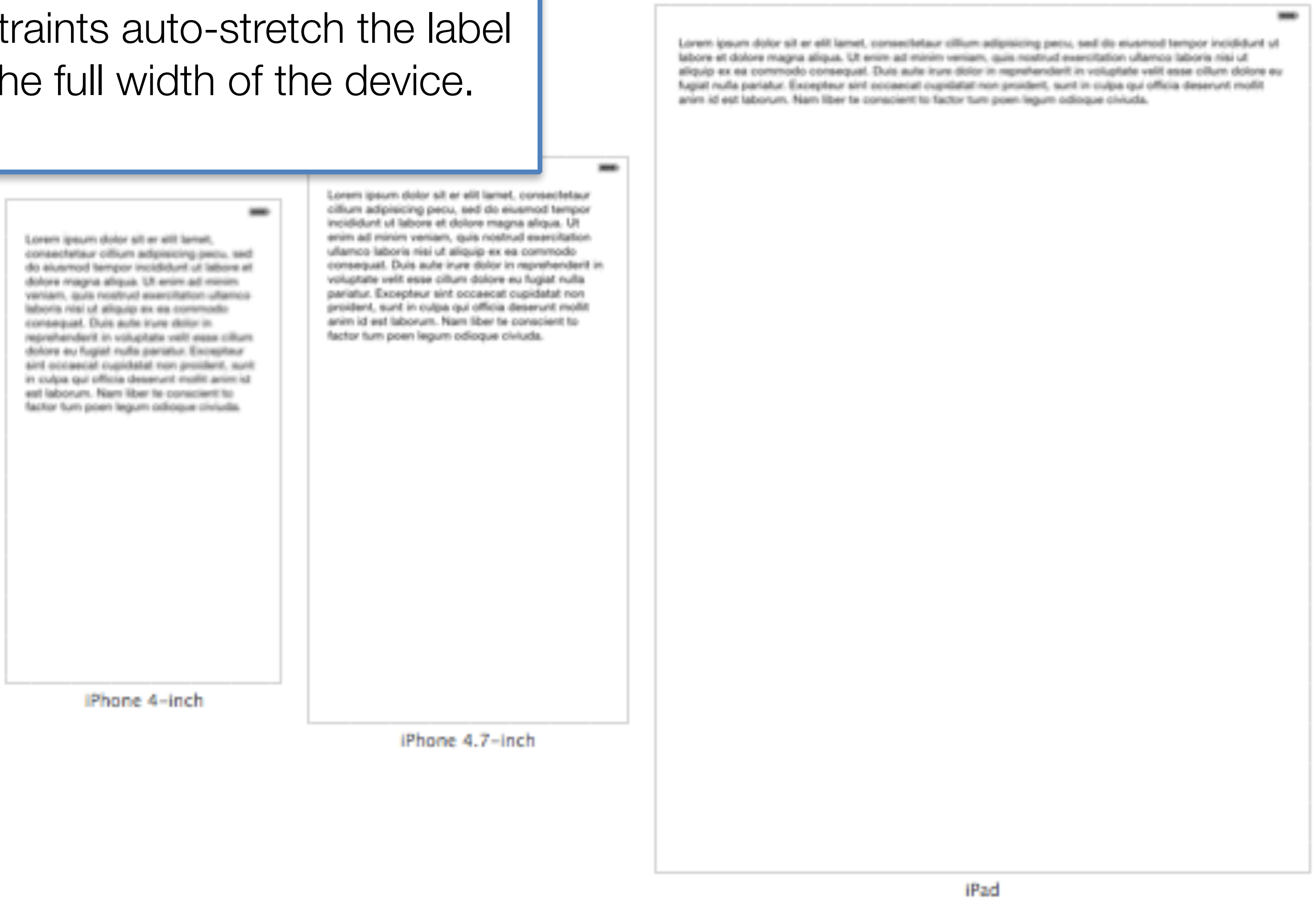

Carrier 6:25 AM

Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

No constraints. Label occupies fixed width / height regardless of resolution / orientation.

Adding Constraints

Constraints auto-stretch the label to the full width of the device.



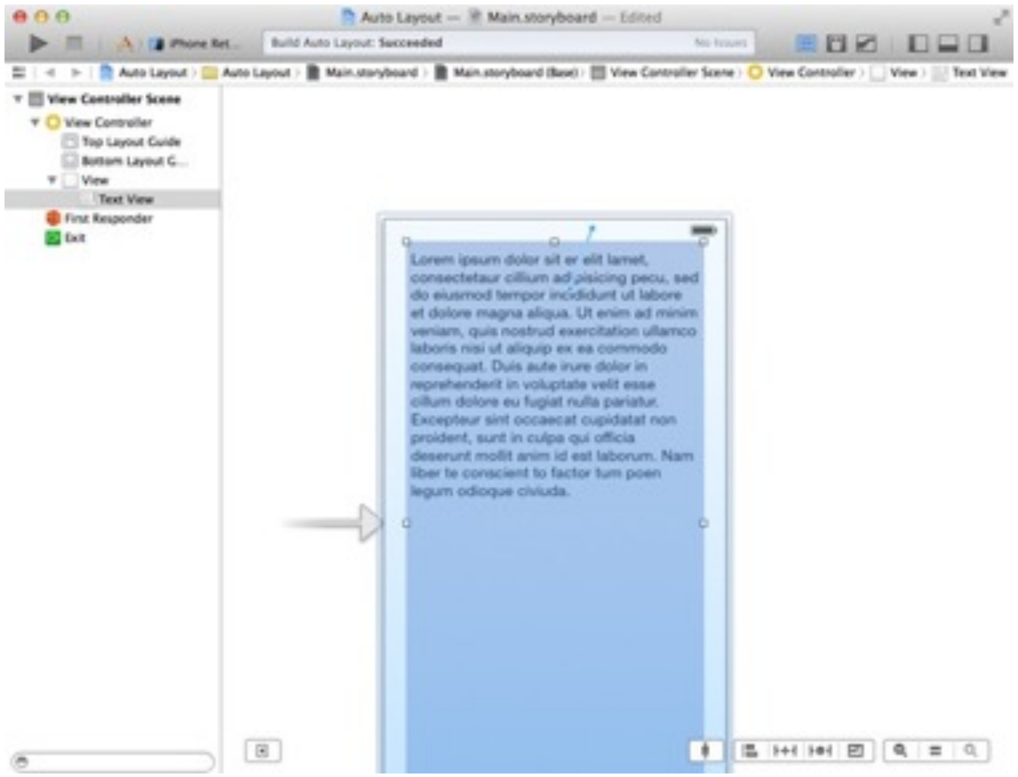
Lorem ipsum dolor sit er elit lamet, consectetaur cillum adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

Lorem ipsum dolor sit er elit lamet, consectetaur cillum adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

Lorem ipsum dolor sit er elit lamet, consectetaur cillum adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

Adding Constraints

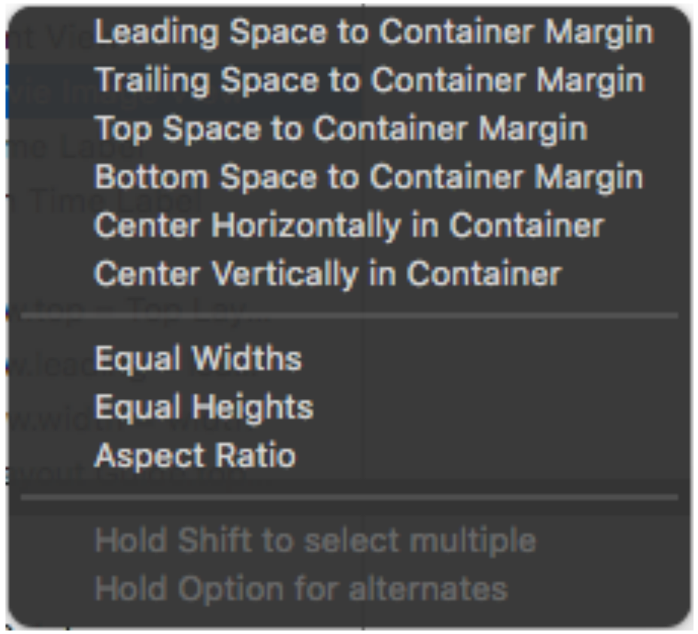
- To center the text view horizontally in the view controller's view, select Center Horizontally In Container from the menu. An orange line appears as a result, signifying the layout constraint you just added.



Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

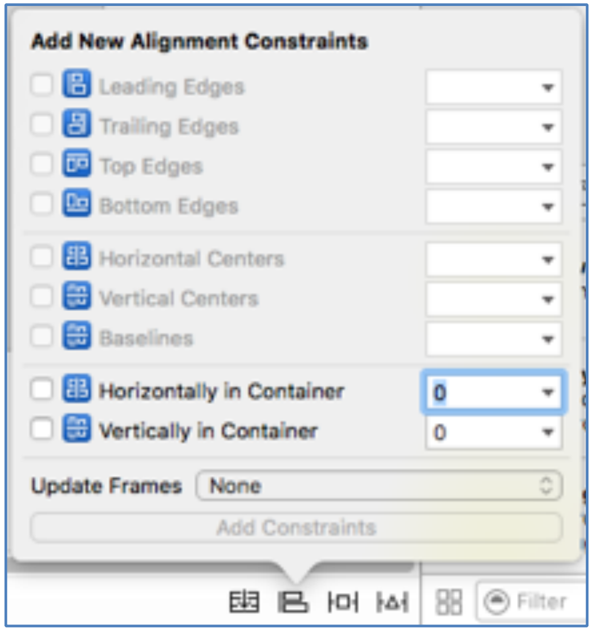
Adding Constraints

- To add constraints:
  - Ctrl-Drag from a view to the parent view
  - Select one of the following:


Leading Space to Container Margin
Trailing Space to Container Margin
Top Space to Container Margin
Bottom Space to Container Margin
Center Horizontally in Container
Center Vertically in Container
Equal Widths
Equal Heights
Aspect Ratio
Hold Shift to select multiple
Hold Option for alternates

Adding Constraints

- Or,
  - Select a view to add constraints:
  - Click on the 昌 ⊢□⊣ icons.

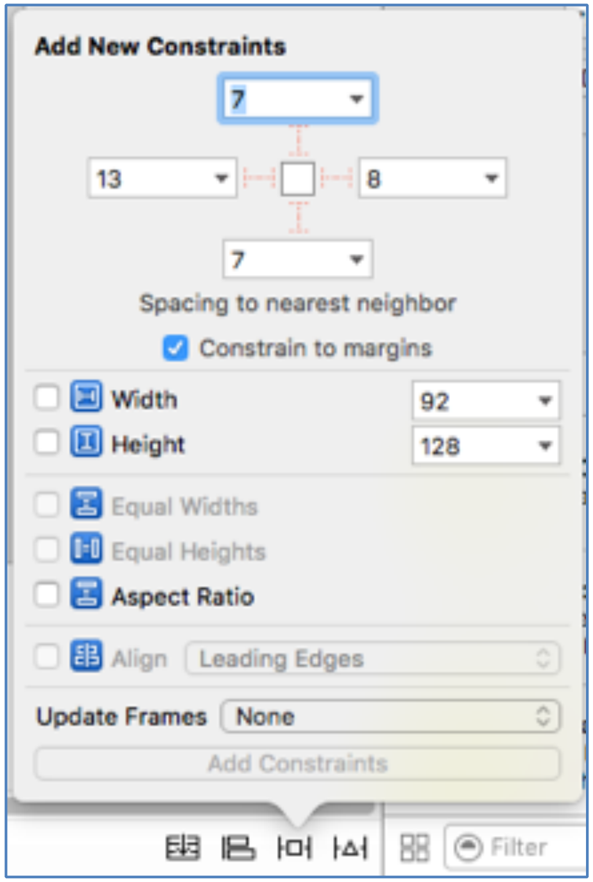


Add New Alignment Constraints

- [ ] Leading Edges
- [ ] Trailing Edges
- [ ] Top Edges
- [ ] Bottom Edges
- [ ] Horizontal Centers
- [ ] Vertical Centers
- [ ] Baselines
- [ ] Horizontally in Container: 0
- [ ] Vertically in Container: 0

Update Frames: None

Add Constraints



Add New Constraints

Top: 7; Left: 13; Right: 8; Bottom: 7

Spacing to nearest neighbor

- [✓] Constrain to margins
- [ ] Width: 92
- [ ] Height: 128
- [ ] Equal Widths
- [ ] Equal Heights
- [ ] Aspect Ratio
- [ ] Align: Leading Edges

Update Frames: None

Add Constraints

Auto Layout

- Demonstrate the Auto Layout in the Practical
- References:
  - https://developer.apple.com/library/mac/documentation/UserExperience/Conceptual/AutolayoutPG/index.html

Adaptive UIs

Stack Views

Stack Views

- **Only supported in iOS 9 and above**

- Allows you to arrange your UI elements in horizontal / vertical stacks, while being responsive to the resolution / orientation

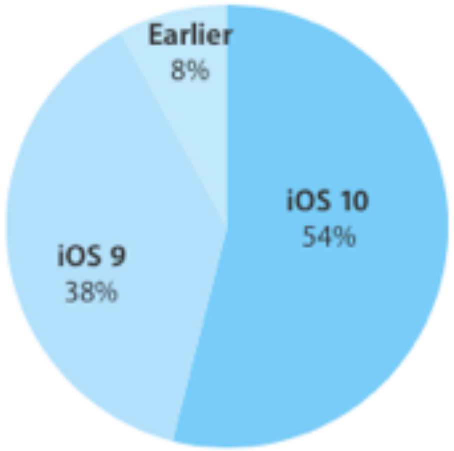
  This is like Android’s LinearLayout

- Internally uses Auto-Layout, but easier to understand and design

Stack Views

- Fortunately:

54% of devices are using iOS 10.



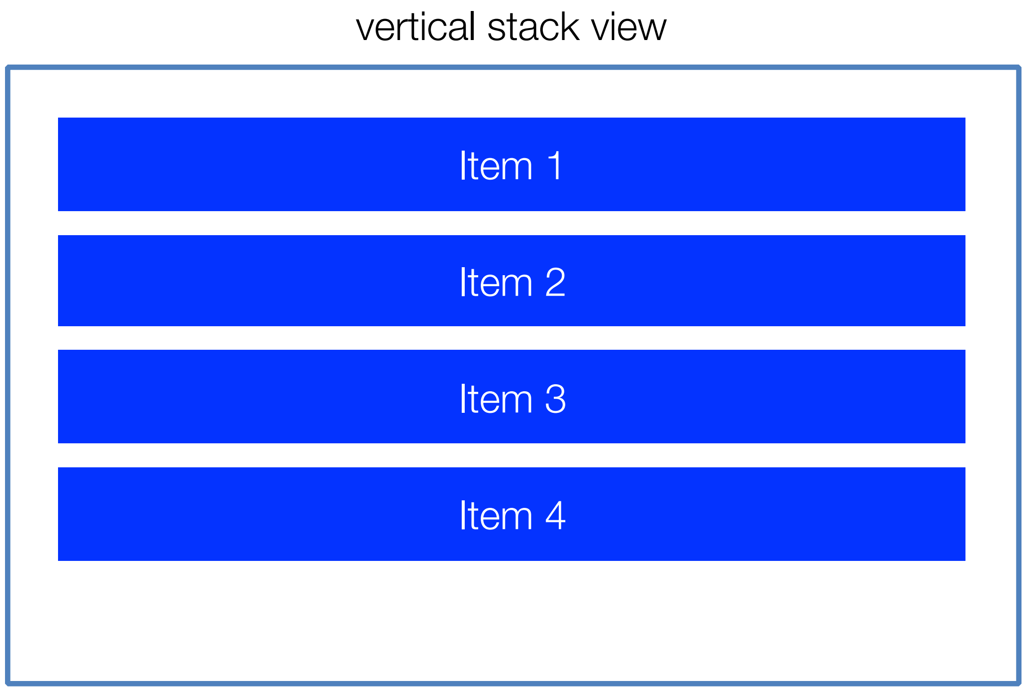
As measured by the App Store on October 7, 2016.

If you **must** target iOS 8 and earlier, consider other open source solutions.

We will not cover those in this module.

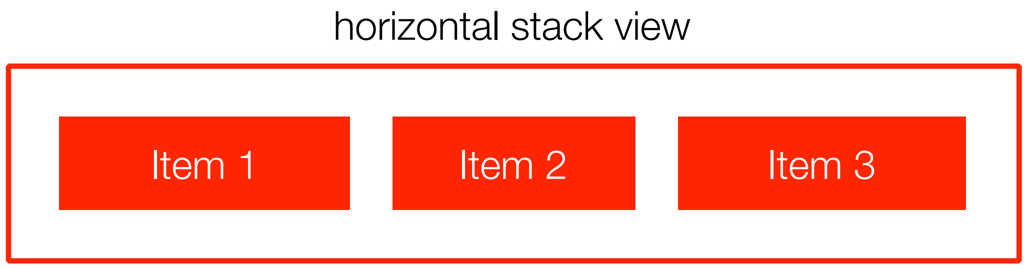
Stack Views

- Example: we can use a **vertical** stack view for the following list of items:



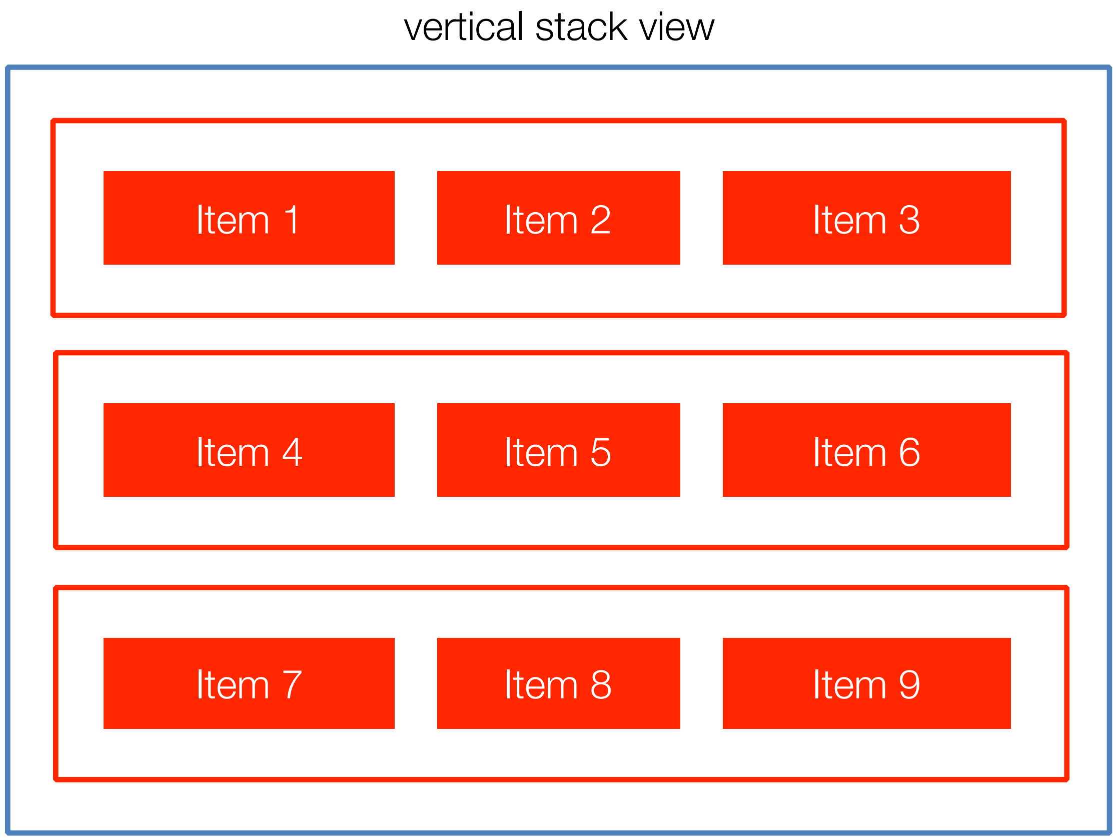
Stack Views

- Example: Or a **horizontal** stack view for the following list of items:



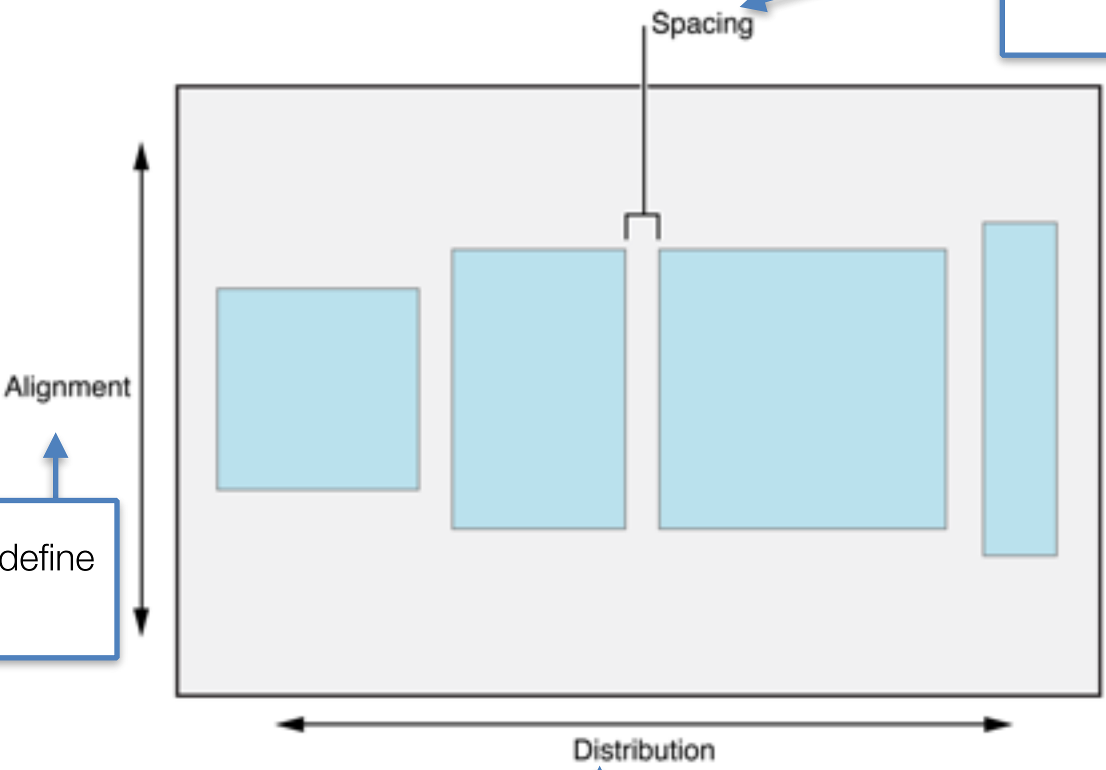
Stack Views

- Example: A mix of both **vertical** and **horizontal** stack views:



Stack Views

- Properties of a stack view



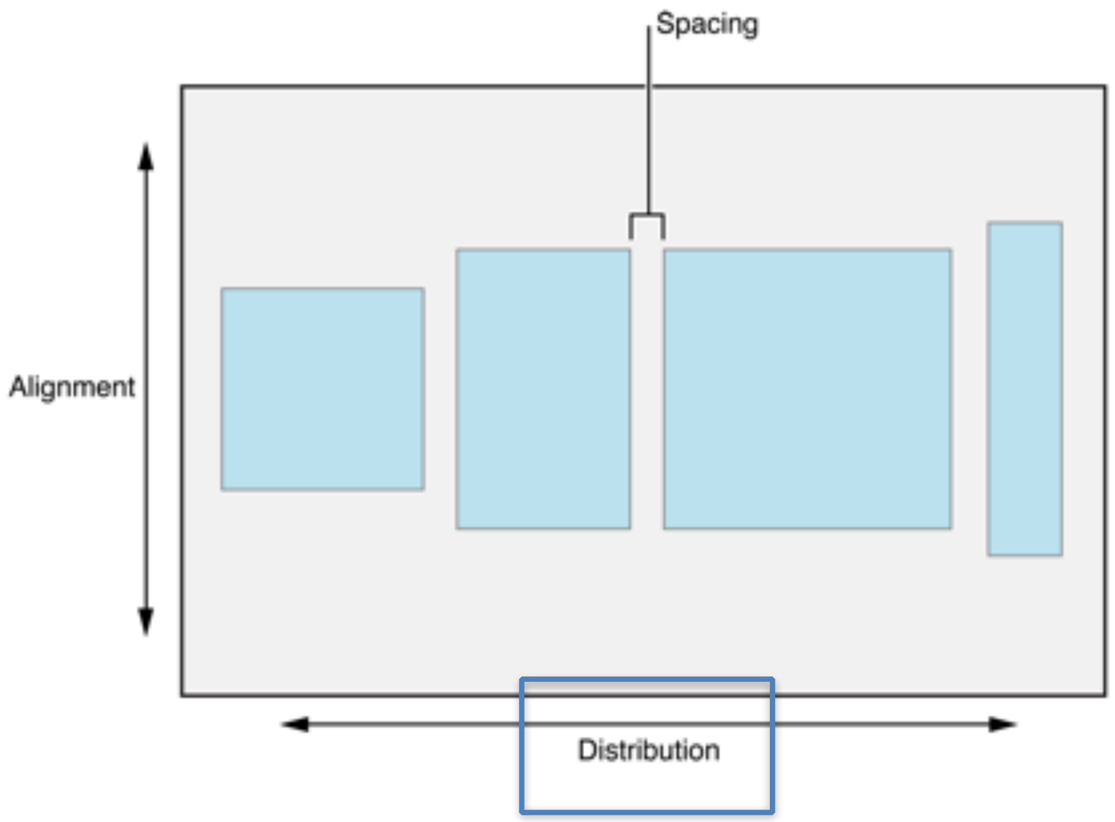
Numeric value you can define in Xcode

Properties you can define in Xcode

Properties you can define in Xcode

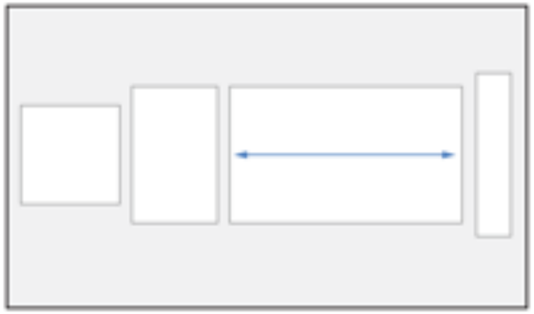
Stack Views

- **Distribution**

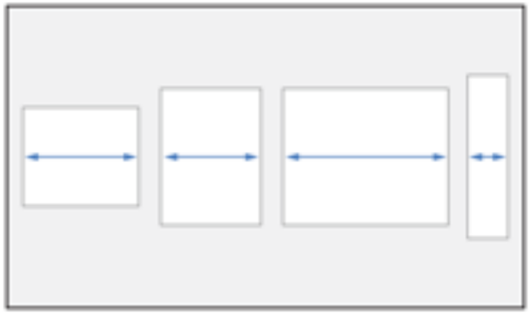


Stack Views Distributions

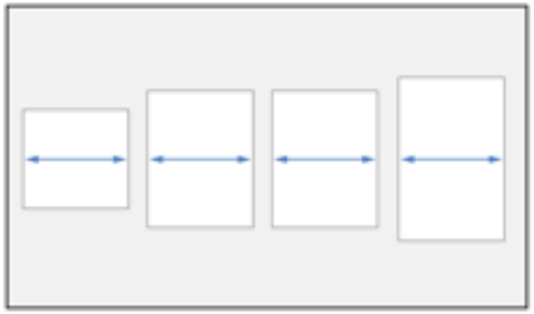
- Commonly-Used Distributions



Fill
Shrink all children, then stretch where possible



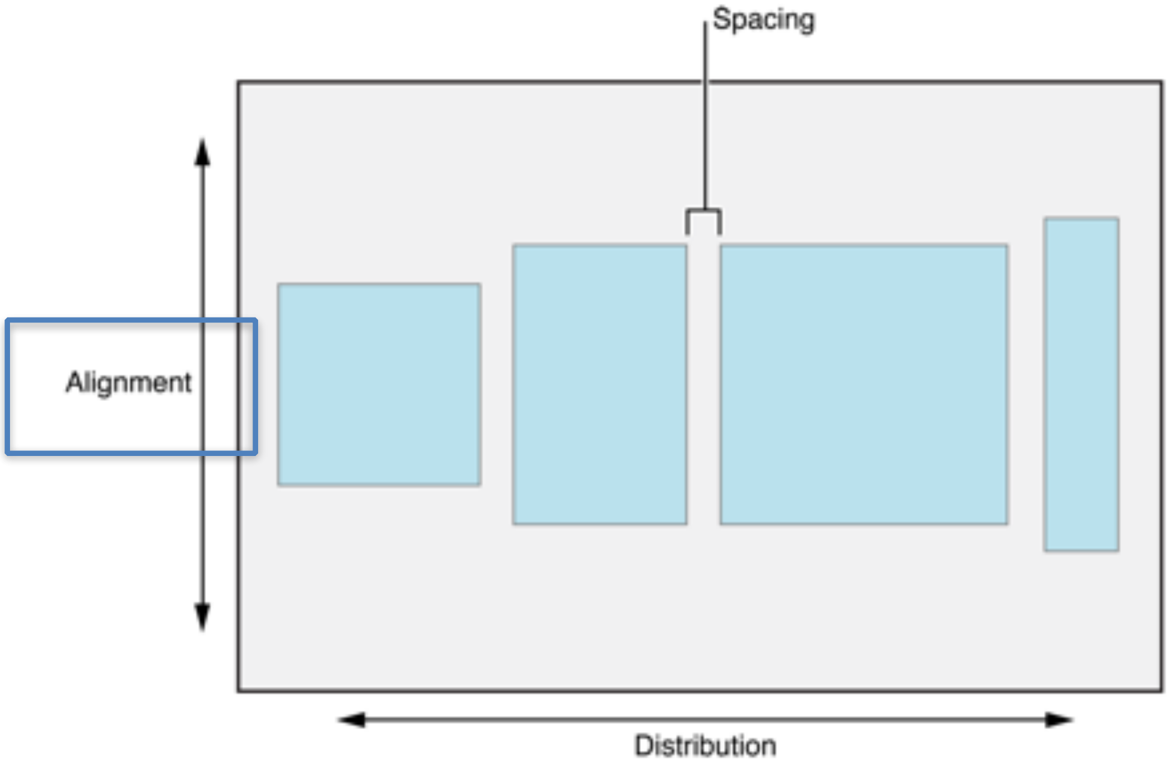
Fill Proportionally
Stretch all based on size of each child. (ratios are maintained)



Fill Equally
All children same widths / heights

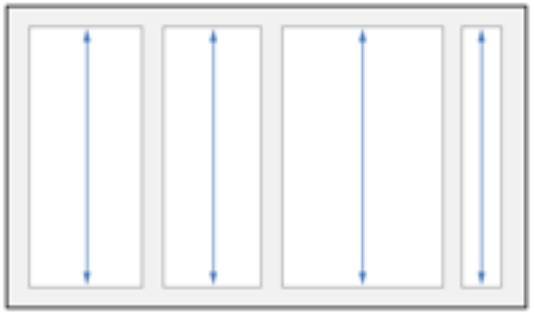
Stack Views

- **Alignment**

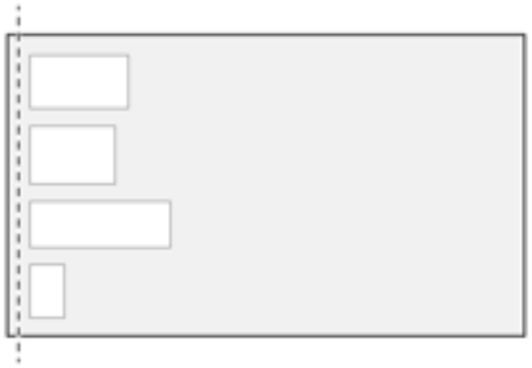


Stack Views Alignments

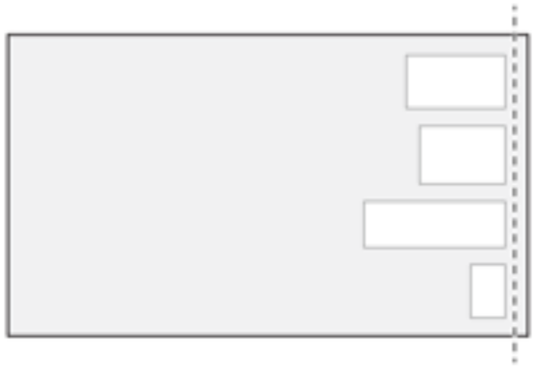
- Commonly-Used Alignments



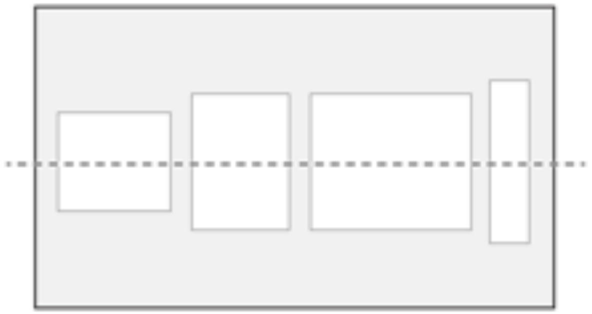
Fill
Stretch to height / width
of stack view



Top / Leading
Align to
top for horizontal stack,
left for vertical stack



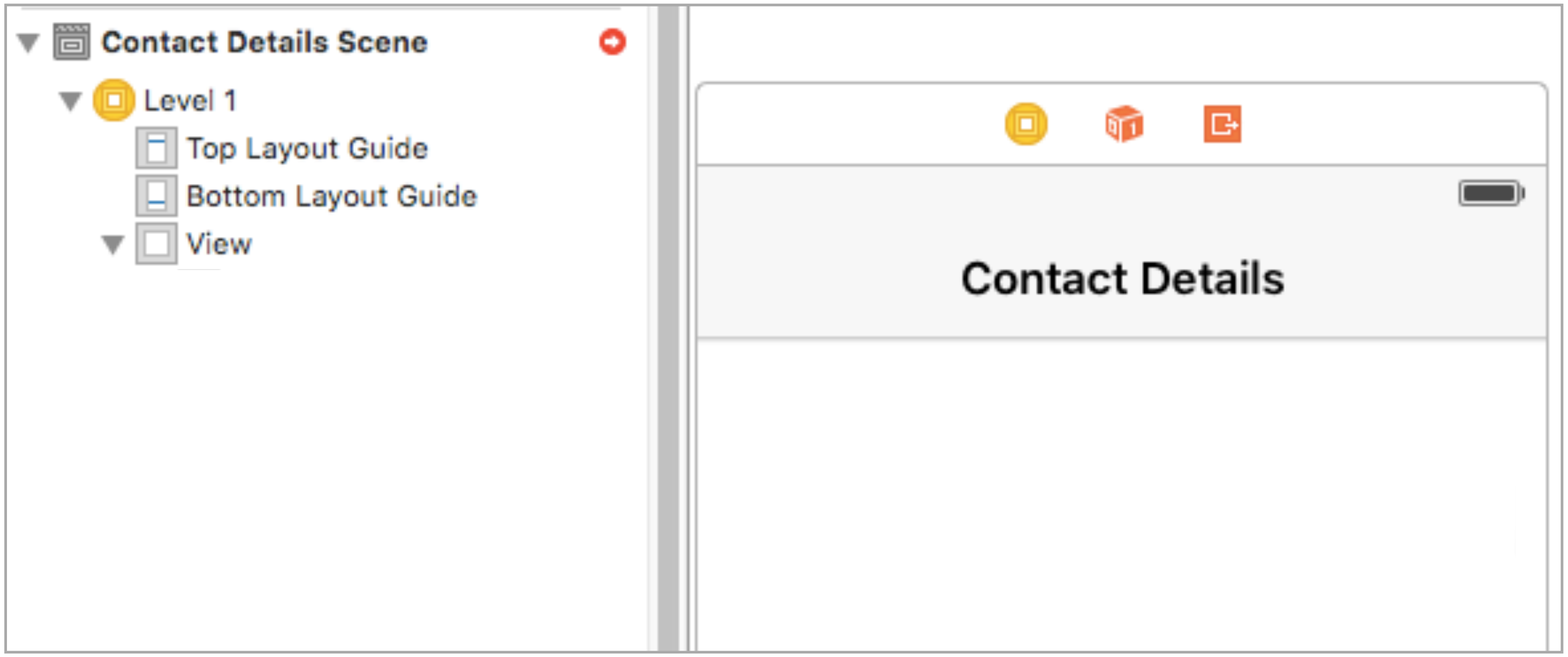
Bottom / Trailing
Align to
bottom for horizontal stack,
right for vertical stack



Centre
Align to centre of stack

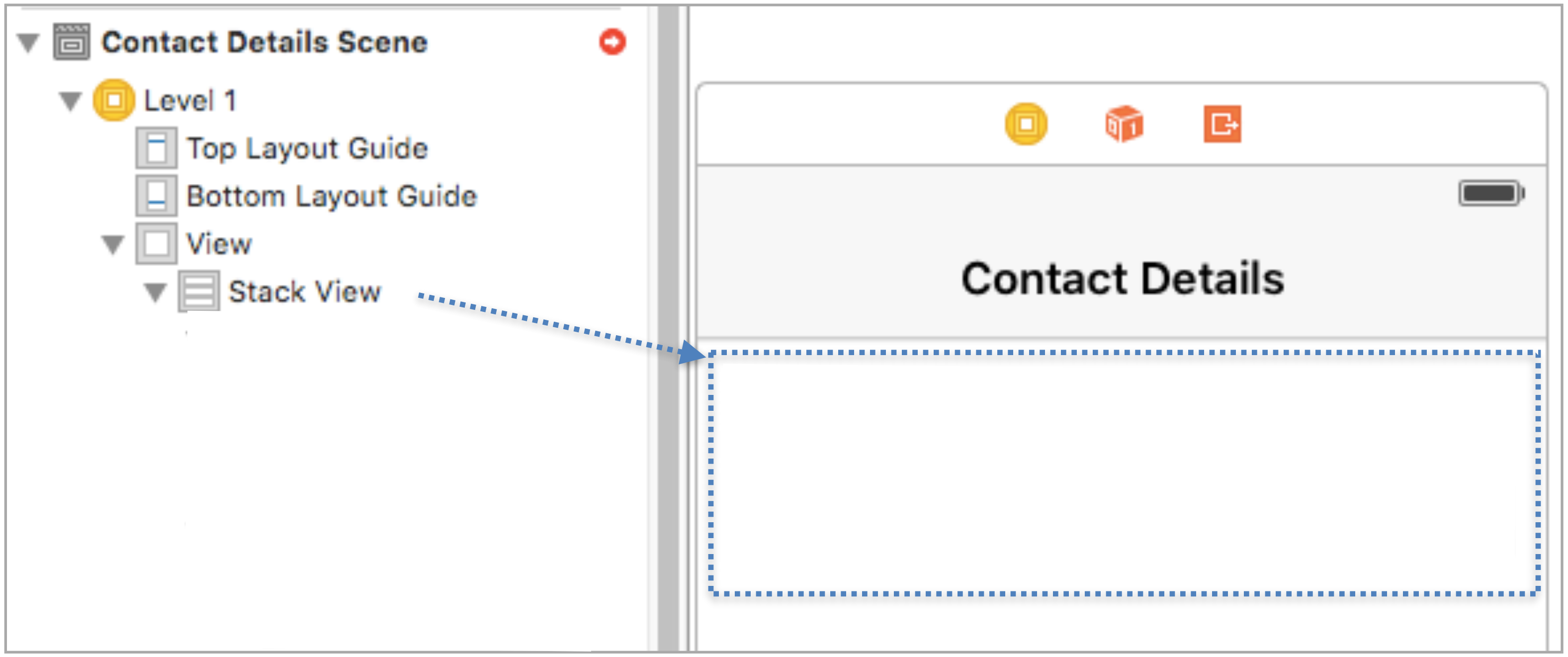
Stack Views

- Example: we can combine nested **horizontal** / **vertical** stack views to achieve something like that:



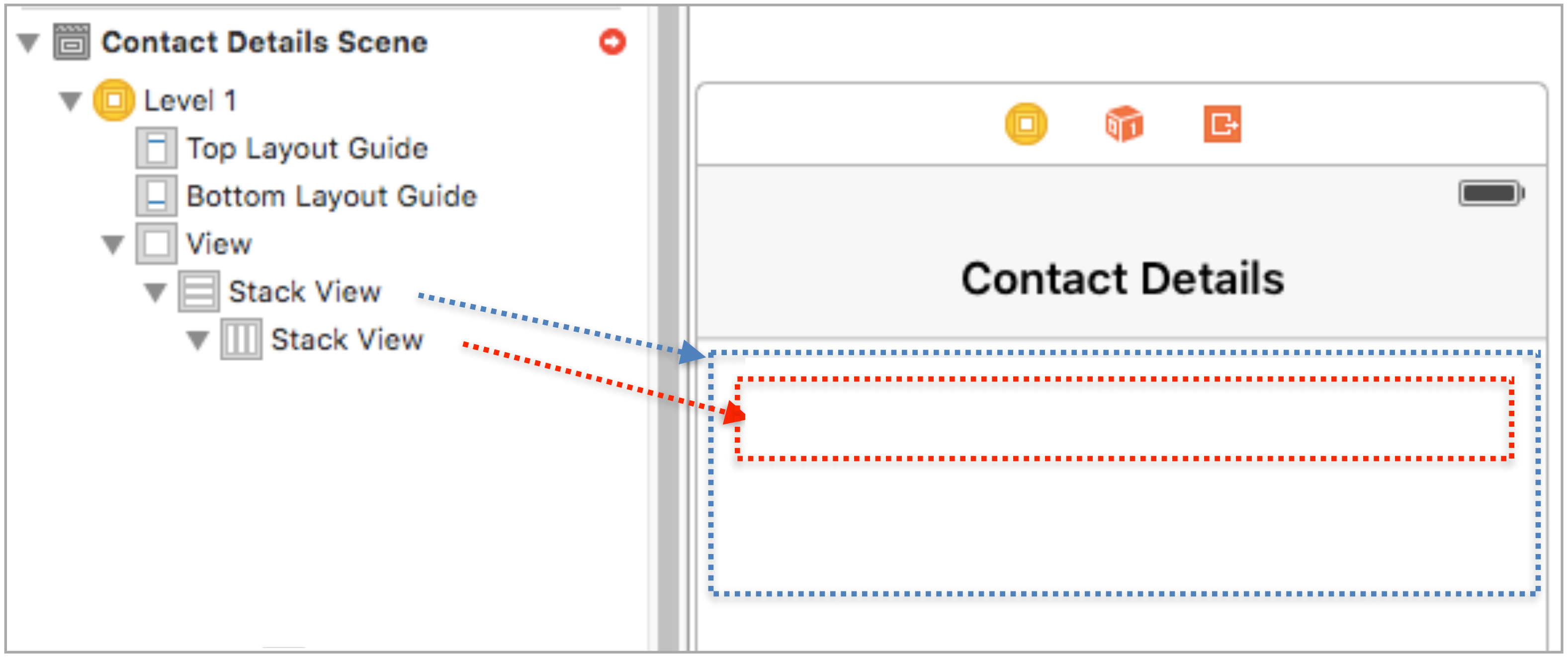
Stack Views

- Example: we can combine nested **horizontal** / **vertical** stack views to achieve something like that:



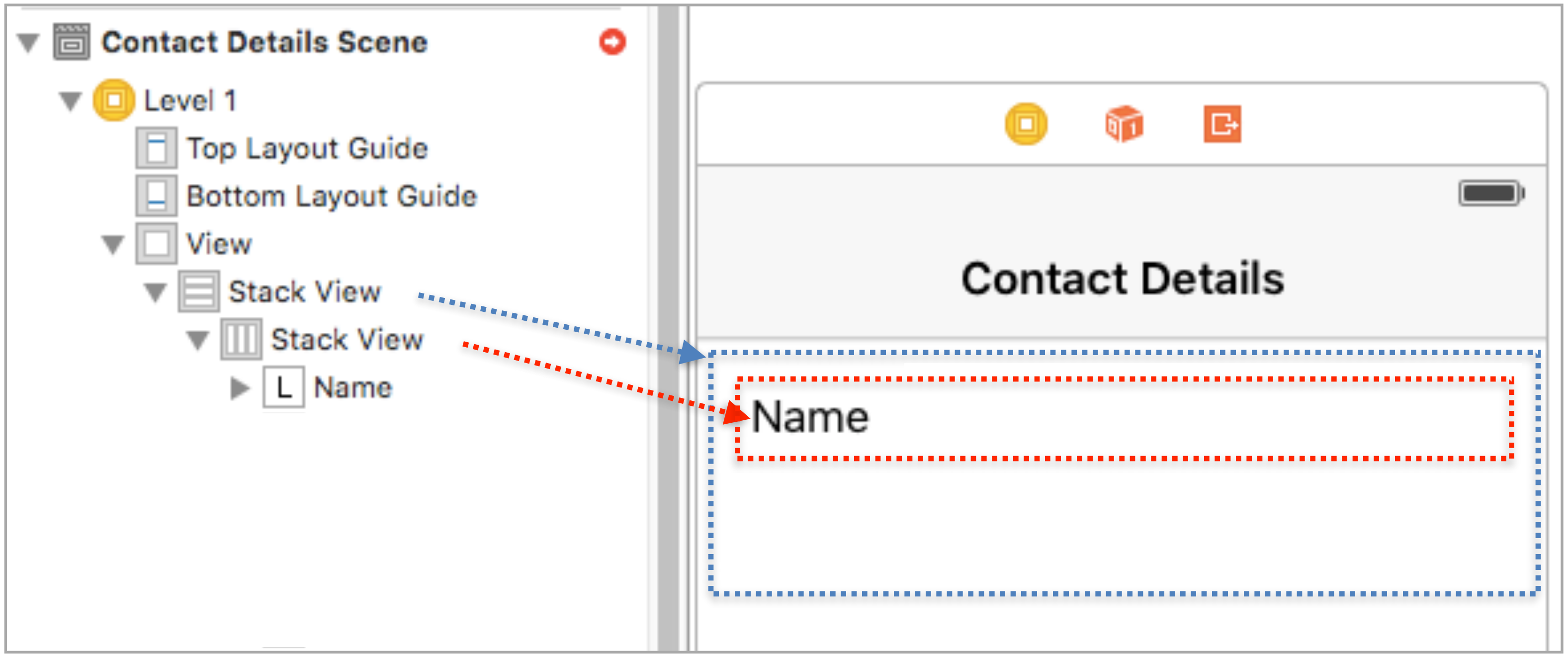
Stack Views

- Example: we can combine nested **horizontal** / **vertical** stack views to achieve something like that:



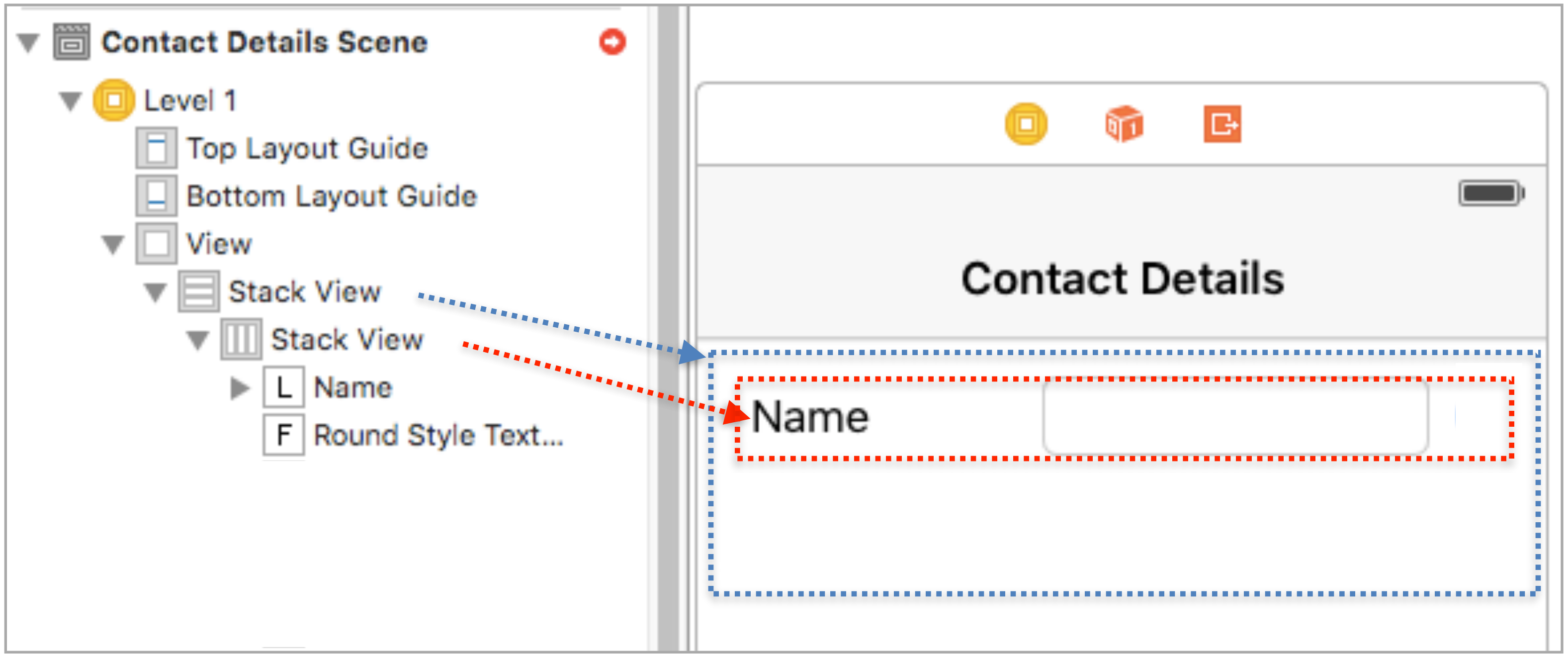
Stack Views

- Example: we can combine nested **horizontal** / **vertical** stack views to achieve something like that:



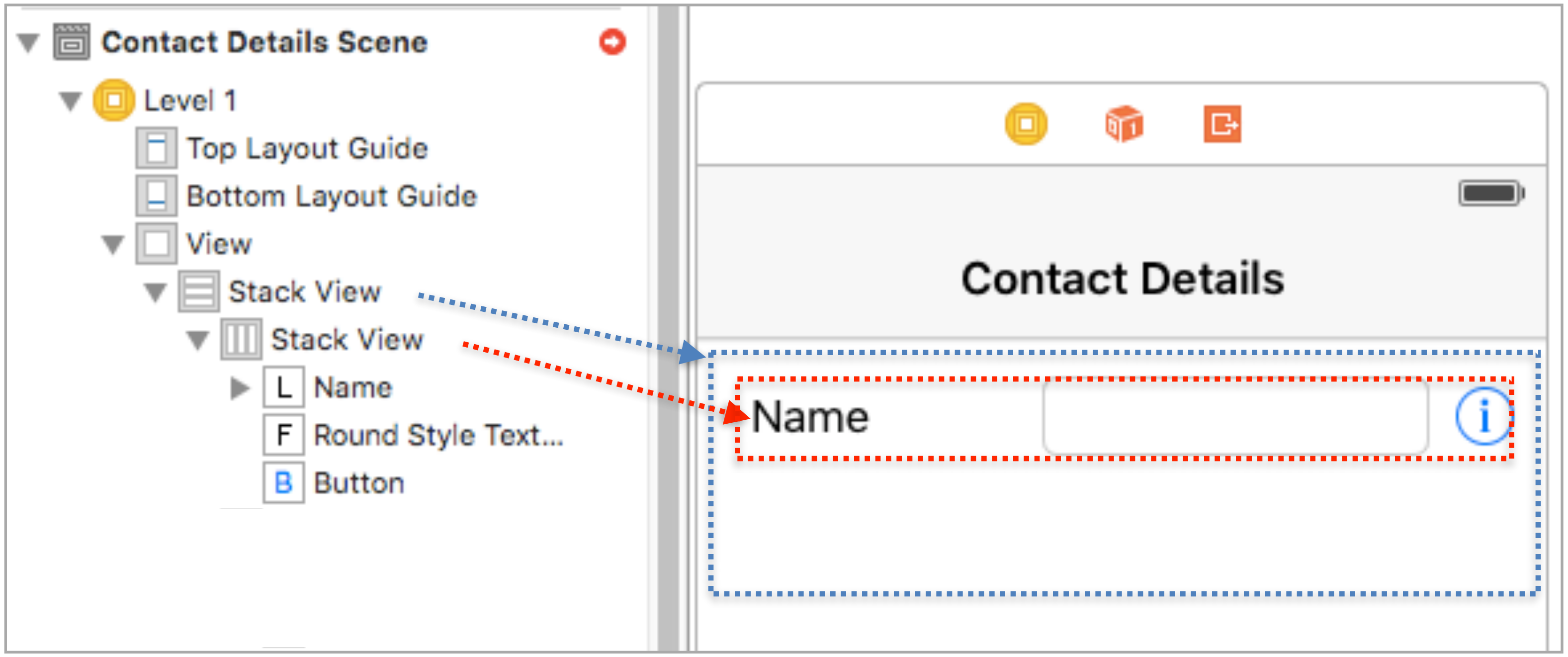
Stack Views

- Example: we can combine nested **horizontal** / **vertical** stack views to achieve something like that:



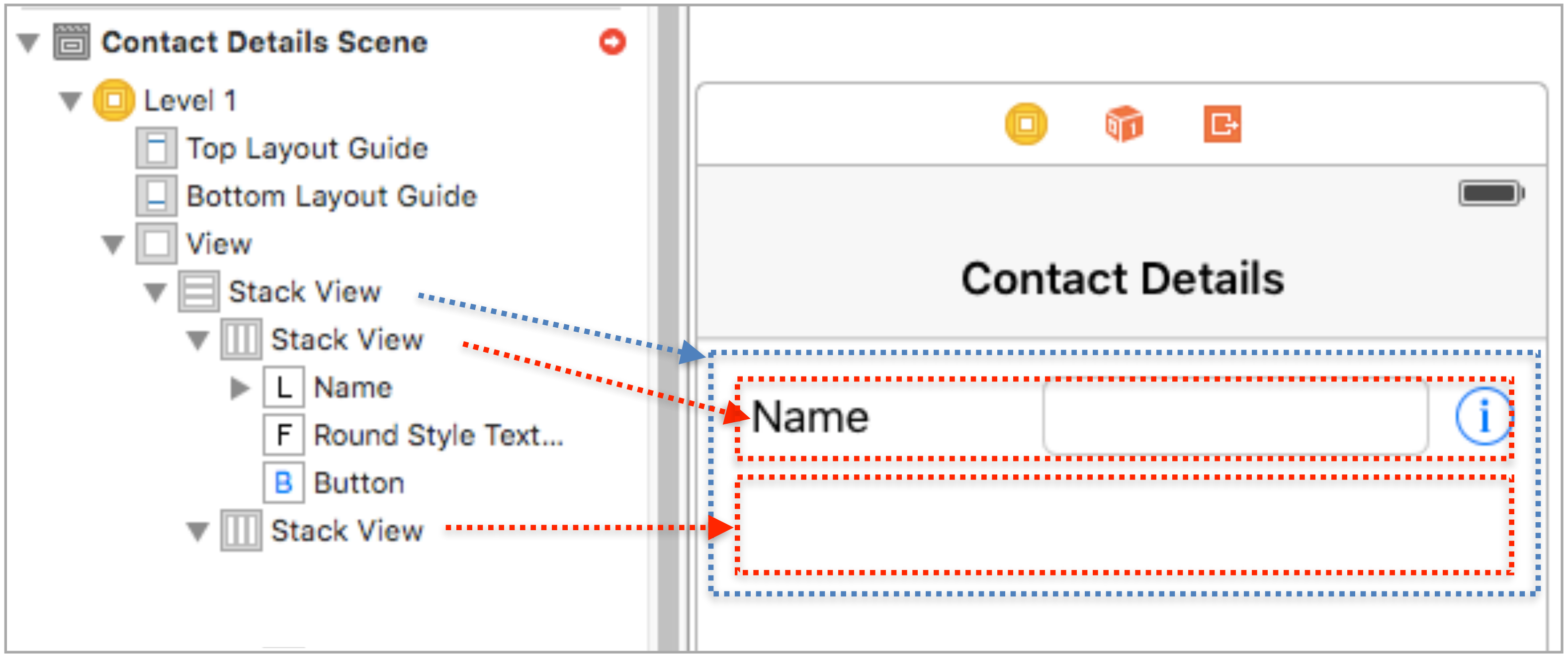
Stack Views

- Example: we can combine nested **horizontal** / **vertical** stack views to achieve something like that:



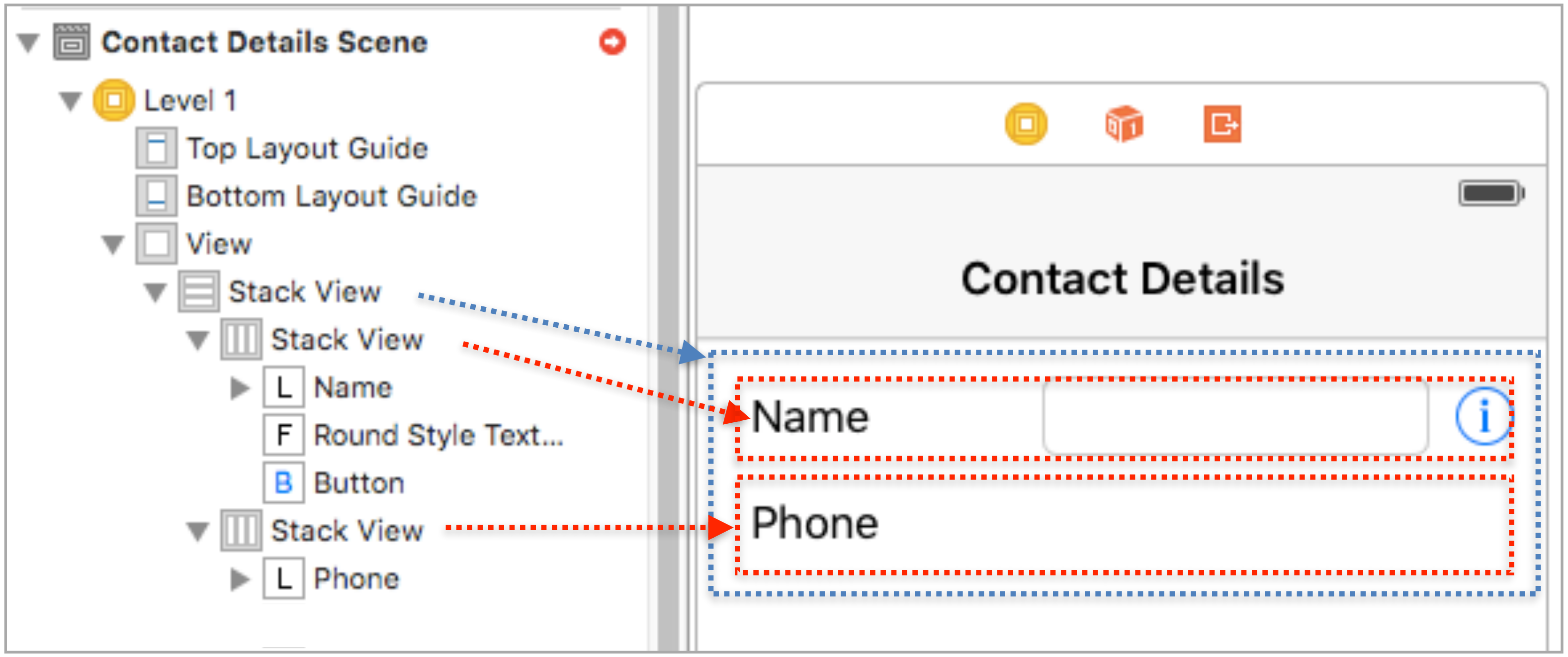
Stack Views

- Example: we can combine nested **horizontal** / **vertical** stack views to achieve something like that:



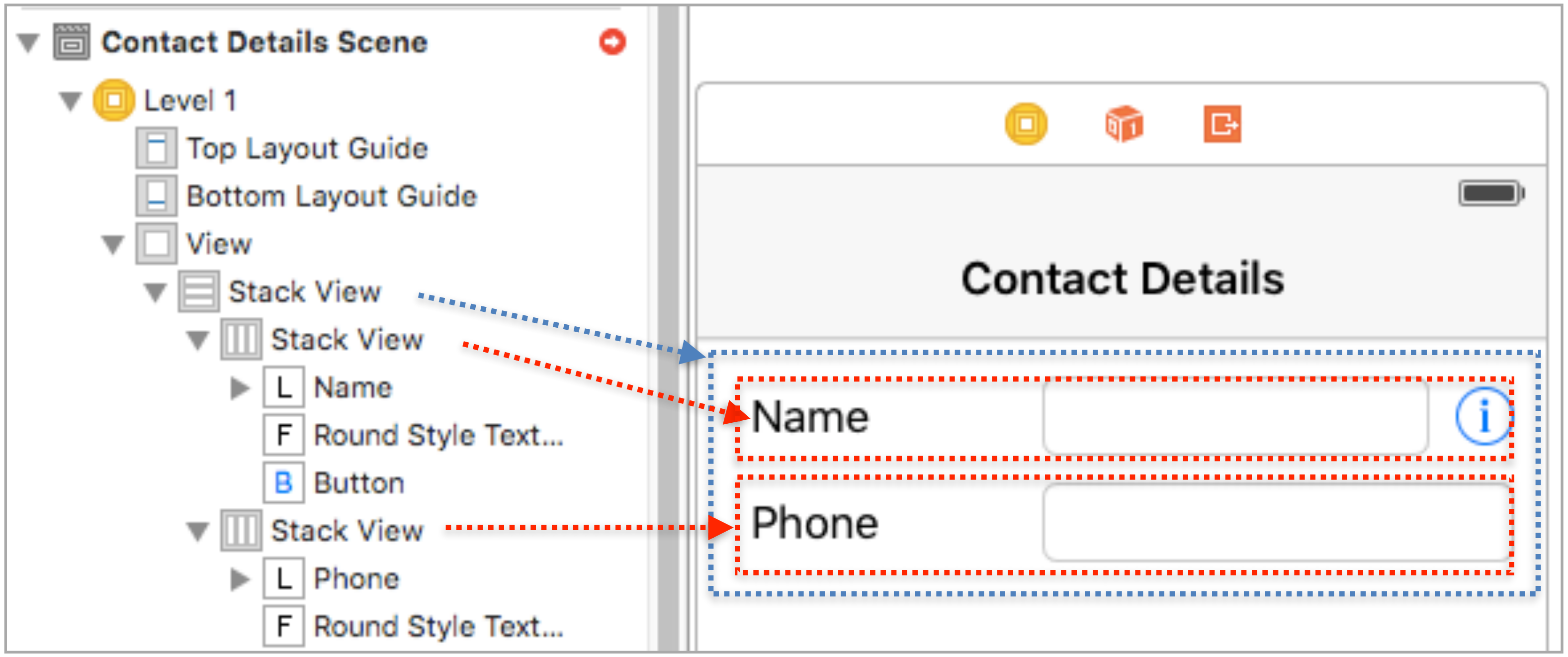
Stack Views

- Example: we can combine nested **horizontal** / **vertical** stack views to achieve something like that:



Stack Views

- Example: we can combine nested **horizontal** / **vertical** stack views to achieve something like that:



Stack Views

- More alignments, distributions and information at:

  https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIStackView_Class_Reference/

Adaptive UIs

Trait Variations

Trait Variations

- Enables designing a single universal storyboard with customised layouts for both iPad and iPhone.
- You can define views and constraints once in the common layout.
- Then, add variations for each supported form factor.

Trait Variations

- Standard Traits and Devices

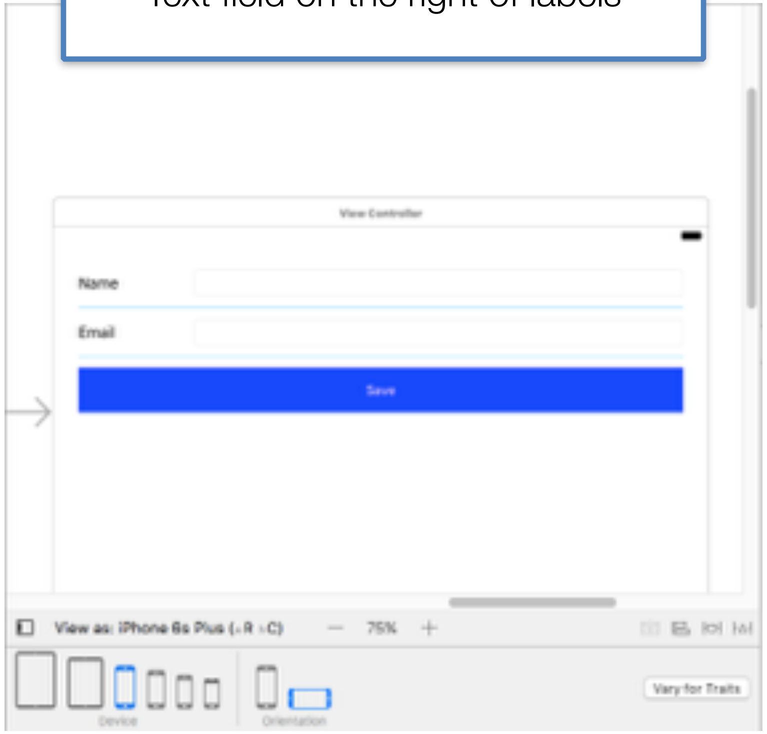
Device	Portrait	Landscape
iPad Pro 12.9"	w: Regular h: Regular	w: Regular h: Regular
iPad Pro 9.7" iPad Original / Air / Mini	w: Regular h: Regular	w: Regular h: Regular
iPhone 6 Plus	w: Compact h: Regular	w: Regular h: Compact
iPhone 6 iPhone 5S/5C/5 **iPhone SE**	w: Compact h: Regular	w: Compact h: Compact
iPhone 4S	w: Compact h: Regular	w: Compact h: Compact

*For iPad devices, there are traits for apps in split-screen mode.

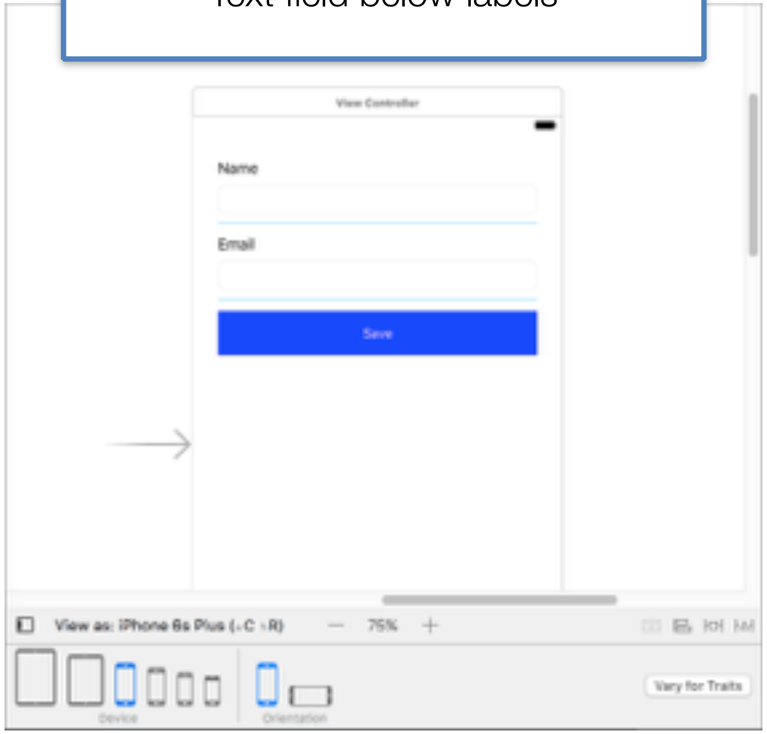
Trait Variations - Example

- Change layouts based on the device / orientation

Text field on the right of labels



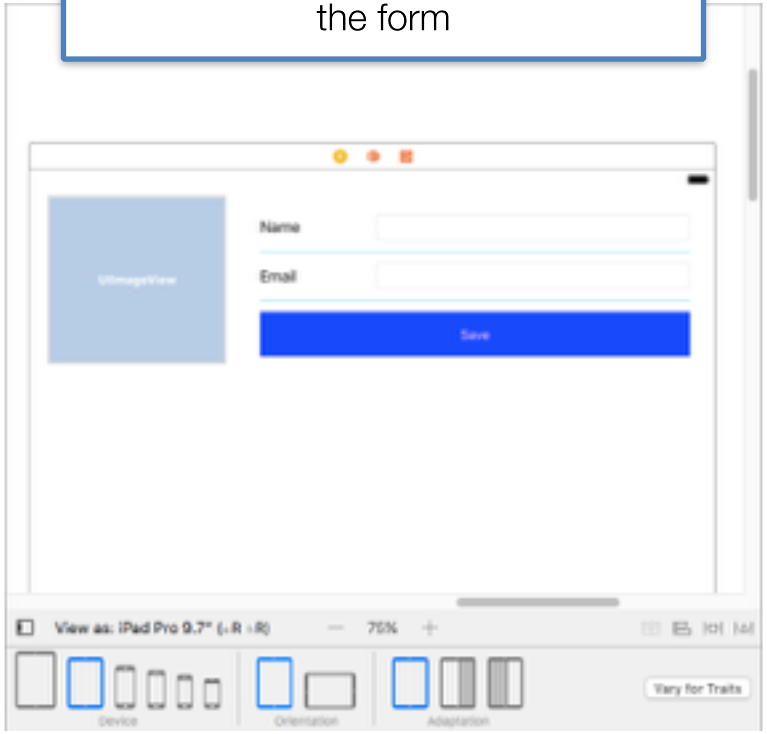
Text field below labels



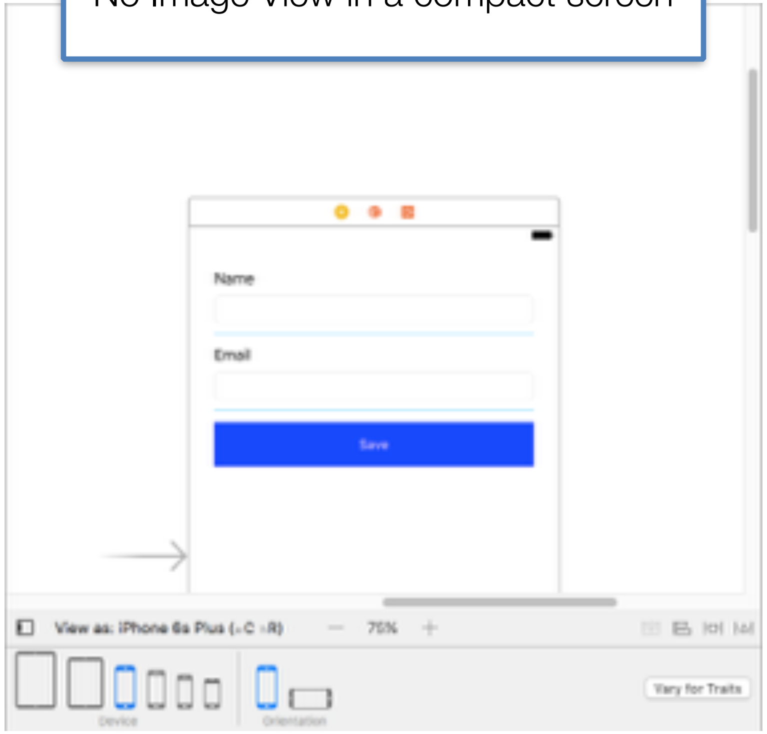
Trait Variations - Example

- Show/hide Views based on the device / orientation

Image View appearing to the right of the form

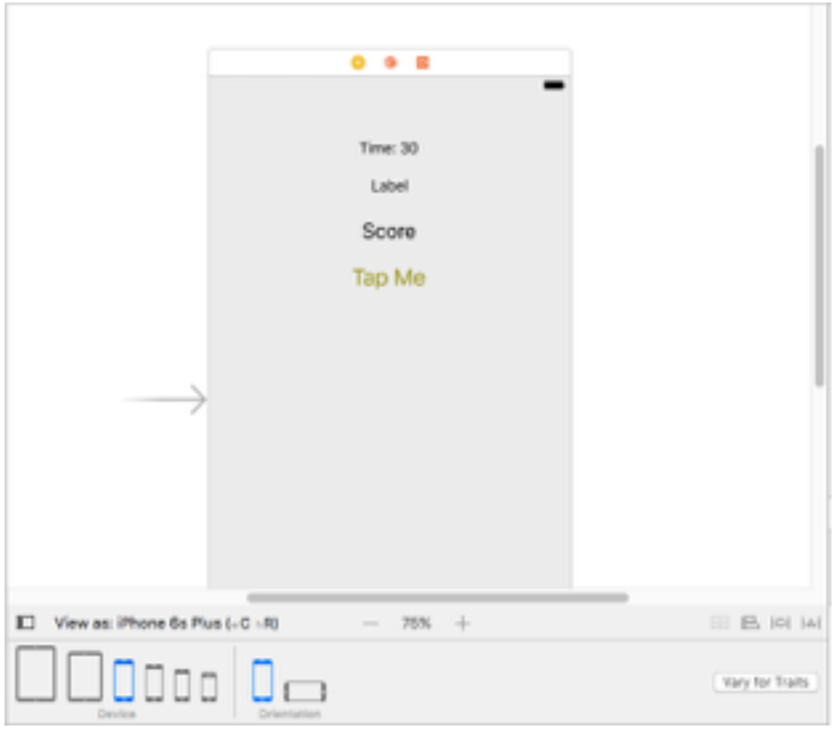


No Image View in a compact screen



Trait Variations - Default

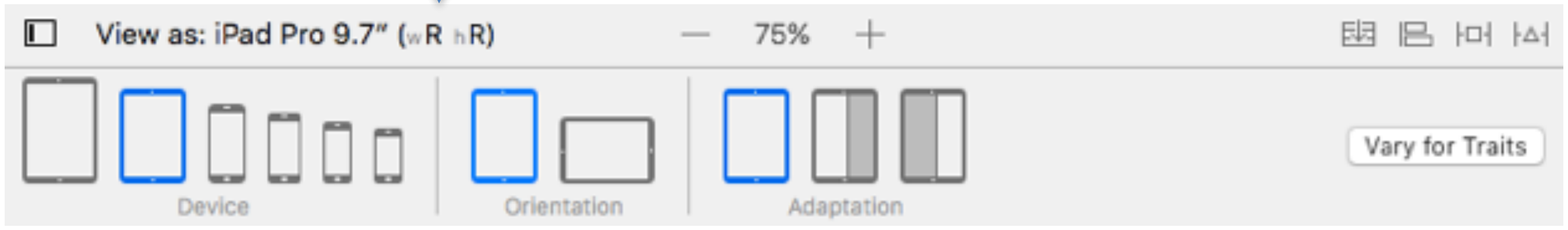
- By default, when you design:
  - The views, view properties and constraints apply to **all** devices



Trait Variations - In XCode

- Switch between different devices, orientations, adaption to see how your user interface looks like in Storyboard

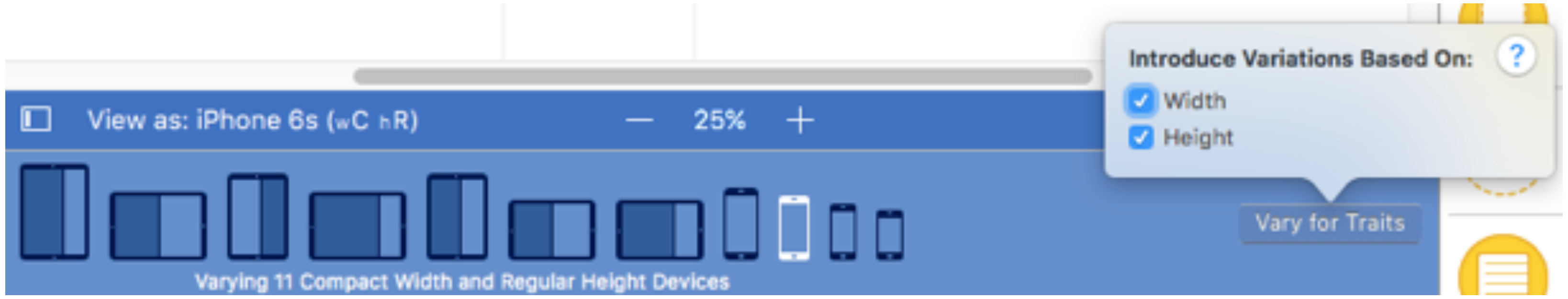
The size class this device belongs to.
R = regular; C = compact
In this example, weight: Regular, height: Regular



For iPads that support split screen multi-tasking

Trait Variations - In XCode

- Click 'Vary for Traits' to add views and constraints specific to devices of specific width / height.


View as: iPhone 6s (wC hR)
25%
Varying 11 Compact Width and Regular Height Devices
Vary for Traits

Introduce Variations Based On:
✓ Width
✓ Height

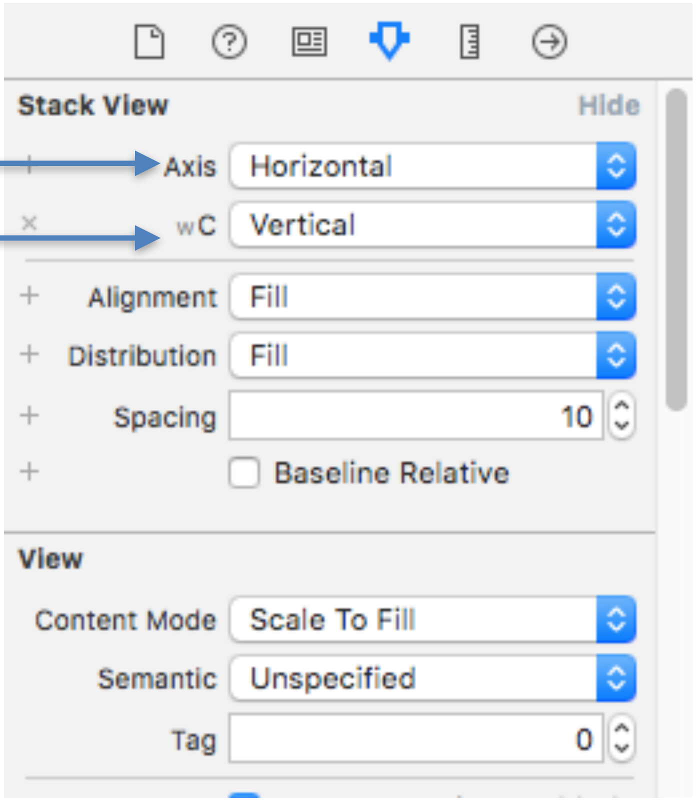
Now you are in the mode designing for width:Compact, height:Regular devices

Trait Variations - In XCode

- Control individual View properties with respect to traits:

For all other devices/orientations

For compact width devices/orientations



Stack View Hide

+	Axis	Horizontal
×	wC	Vertical
+	Alignment	Fill
+	Distribution	Fill
+	Spacing	10
+		Baseline Relative

View

Content Mode	Scale To Fill
Semantic	Unspecified
Tag	0

Summary

- Understand Auto Layout in iOS 8 and above
- Understand Stack Views in iOS 9 and above
- Understand Trait Variations