

Solving Problems in Auto Layout

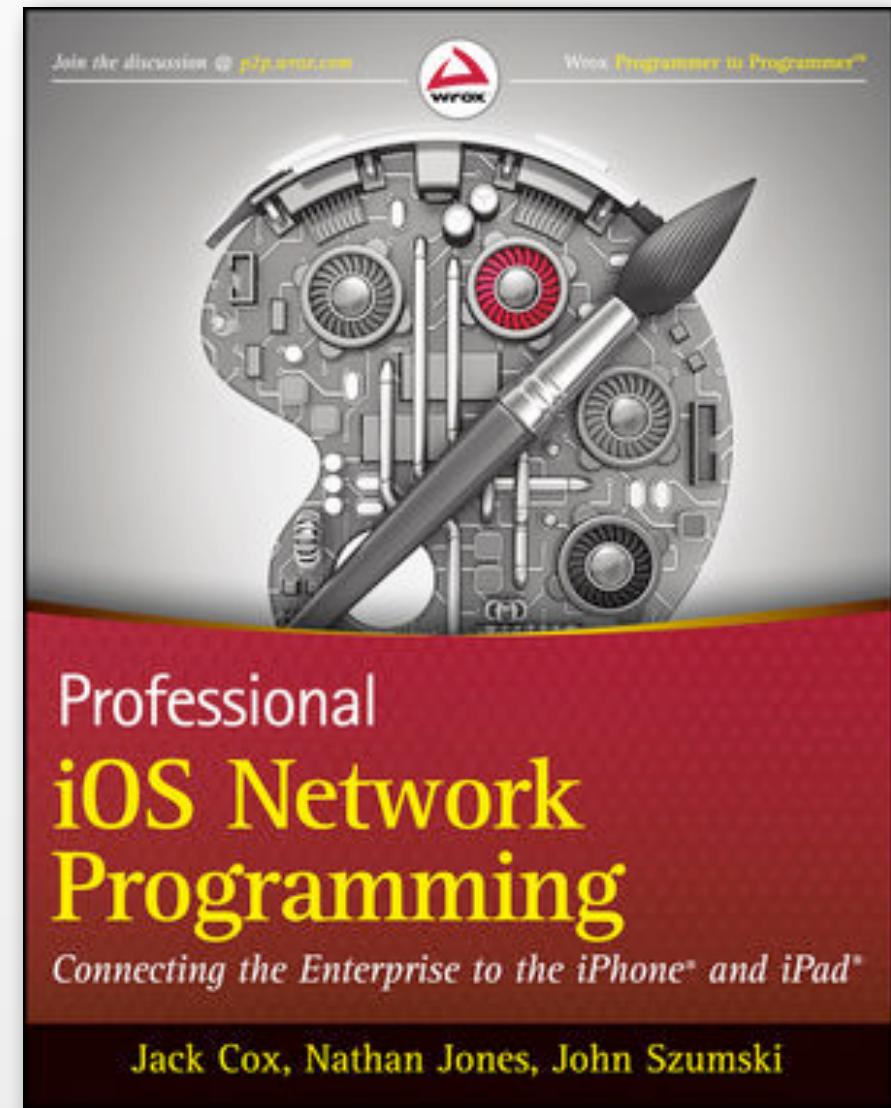
Please, just make the pain stop.

Agenda

- Introduction
- Concepts
- Problem Areas
- Debugging

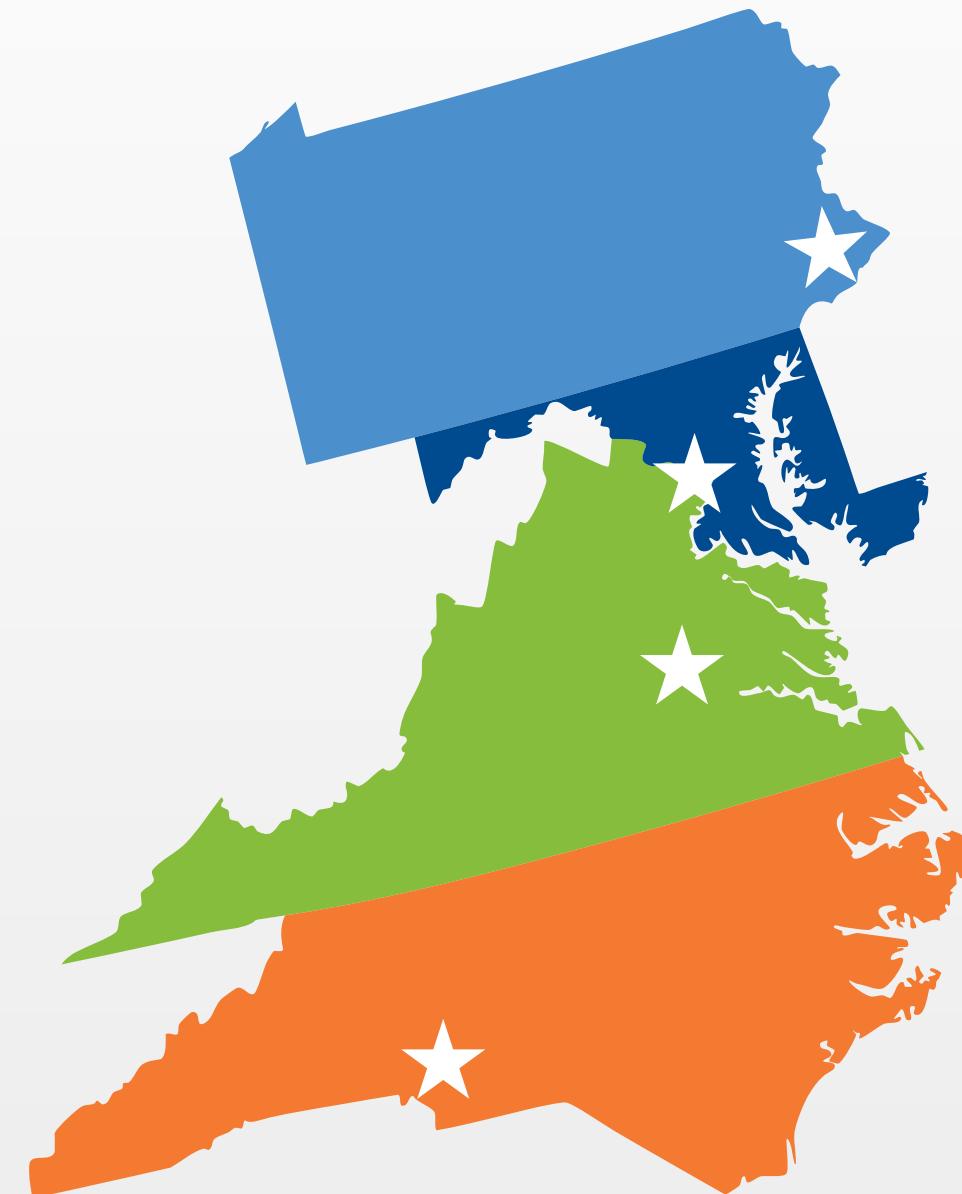
Introduction

- self = [
 - Jack Cox
 - Managing Director, Mobile Technologies at CapTech Consulting
 - Author: Professional iOS Network Programming
 - Husband, Father, Christian];
- Contact Info
 - jcox@captechconsulting.com
 - @jcox_mobile

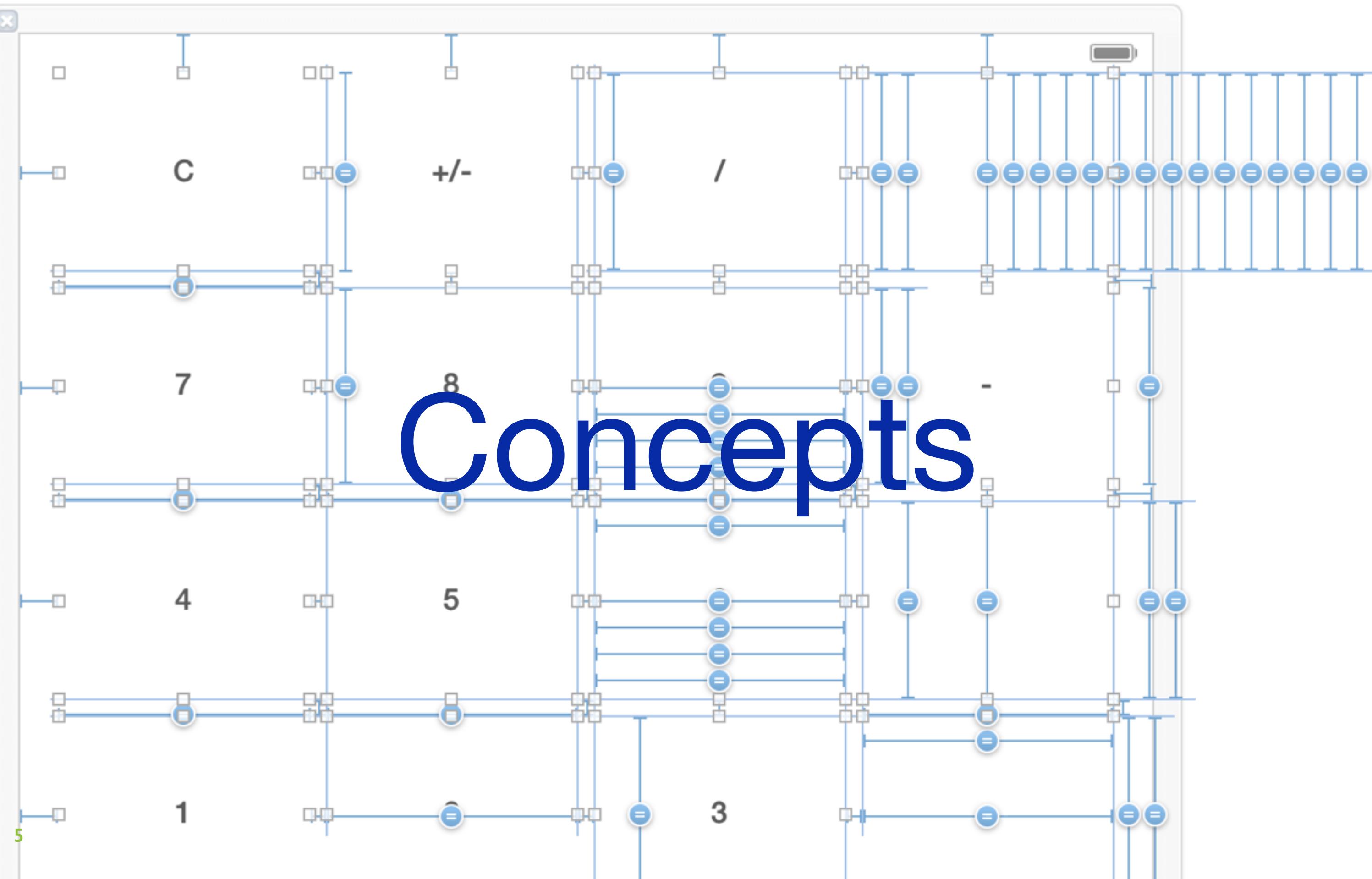


Who is CapTech

- Mid sized consulting firm
- Based in the Mid-Atlantic region
- ~500 Consultants
- ~90 involved in mobile projects



Concepts



Concepts

- View Hierarchy
 - Any 2 views can affect one another



Concepts

- Constraint

Equal Widths Constraint

First Item `ScrollableContent.Width` ▾

Relation `Equal` ▾

Second Item `Superview.Width` ▾

+ Constant `0` ▾ ▾

Priority `1000` ▾ ▾

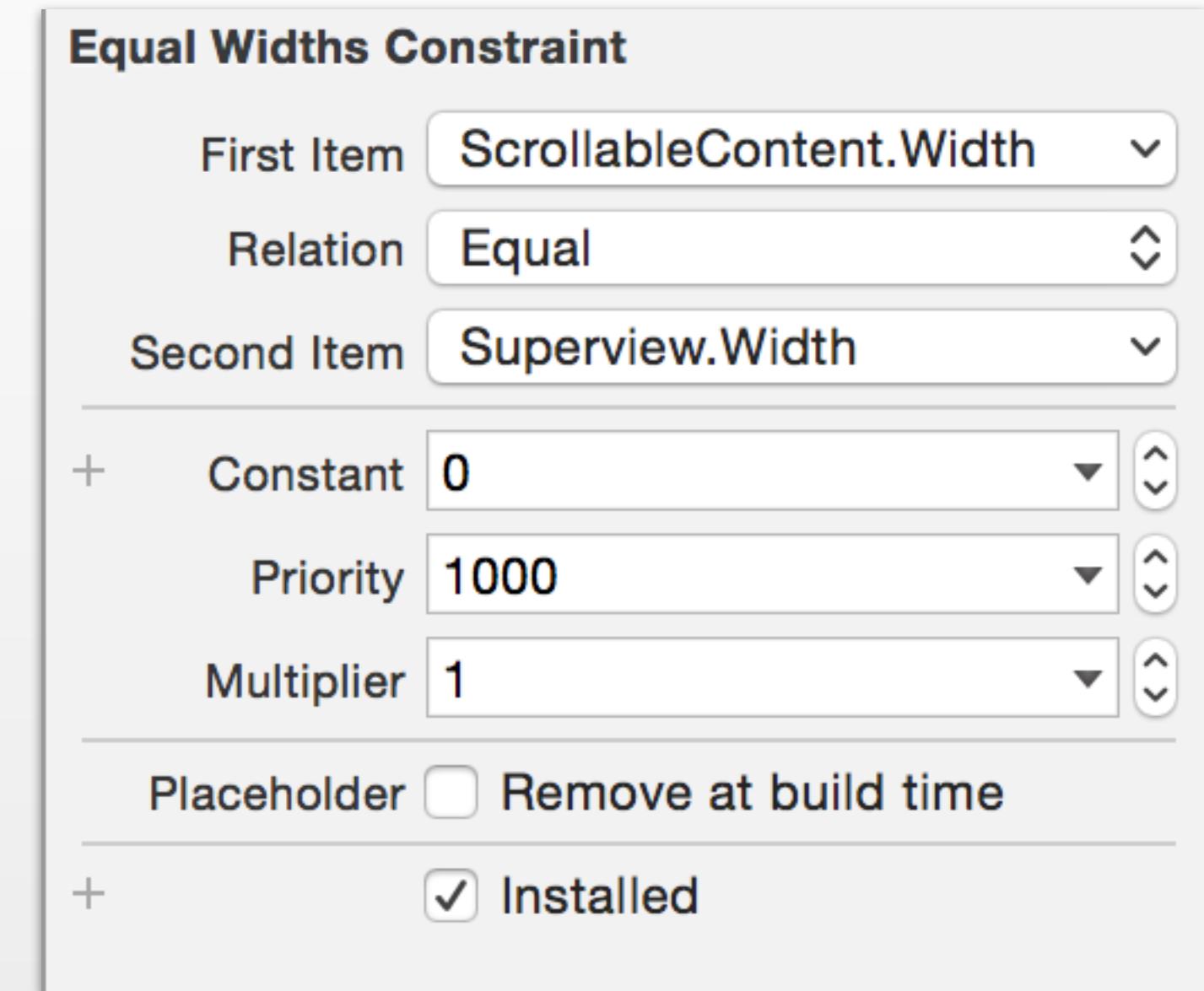
Multiplier `1` ▾ ▾

Placeholder Remove at build time

+ Installed

Mutable Properties

- Constant
- Priority
- Hugging priority
- Resistance priority

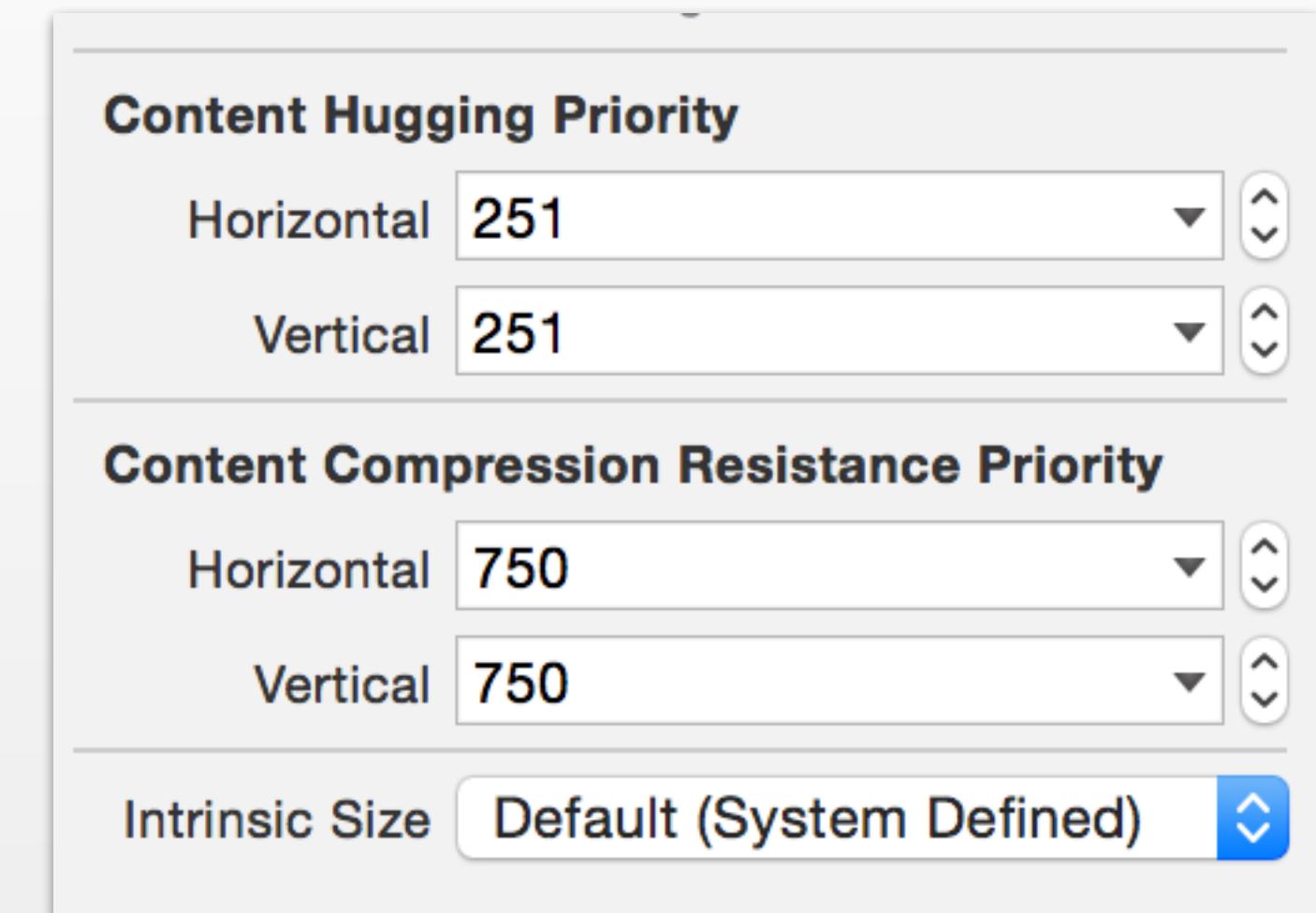


Intrinsic Size

- **How big the view wants to be**
- Based on contents of view
- Child views imbue the intrinsic size on parents
- Most standard controls have intrinsic size
- Views can define their own intrinsic size

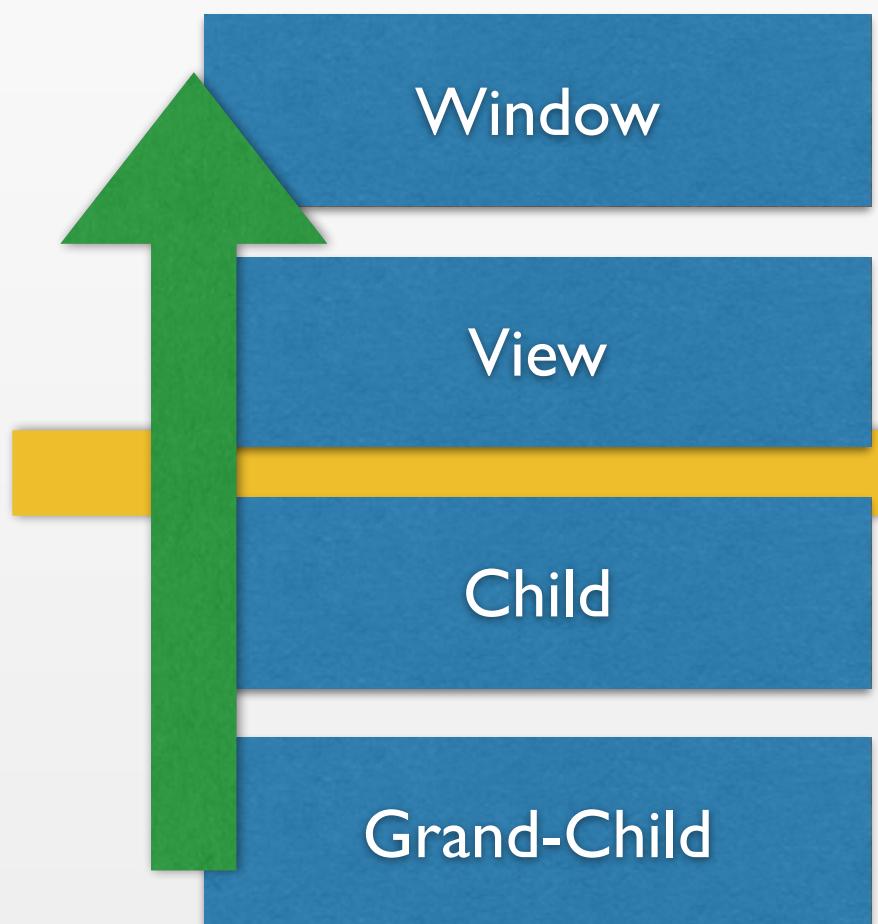
Hugging and Resistance

- Hugging
 - Lower priority = more likely to be enlarged
- Compression resistance
 - Lower priority = more likely to be shrunk

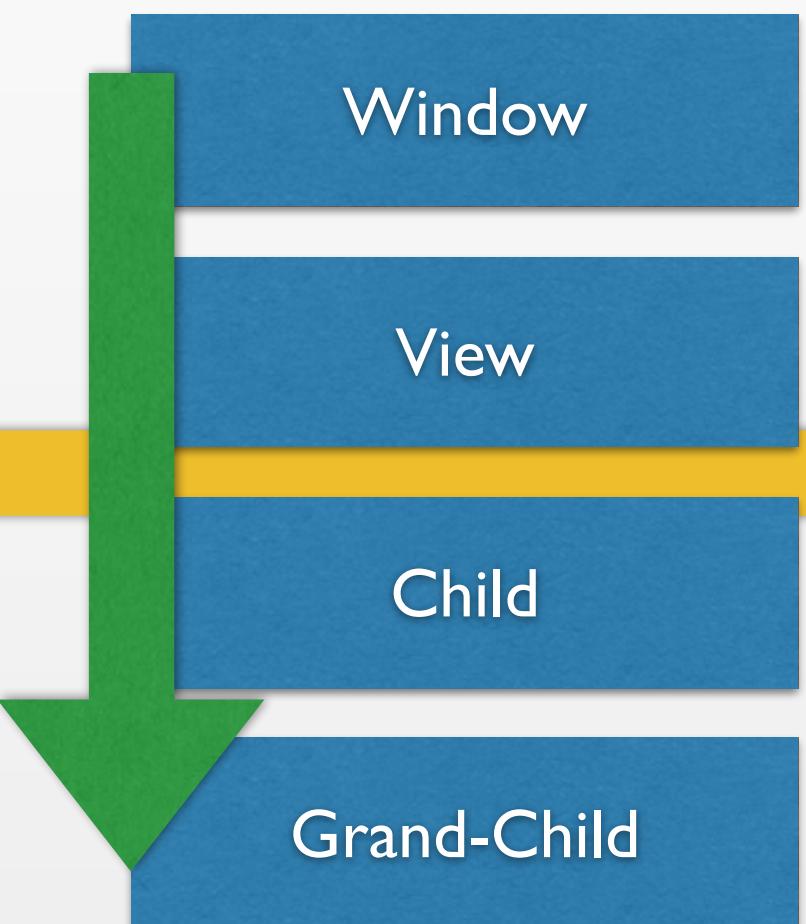


Rendering Cycle

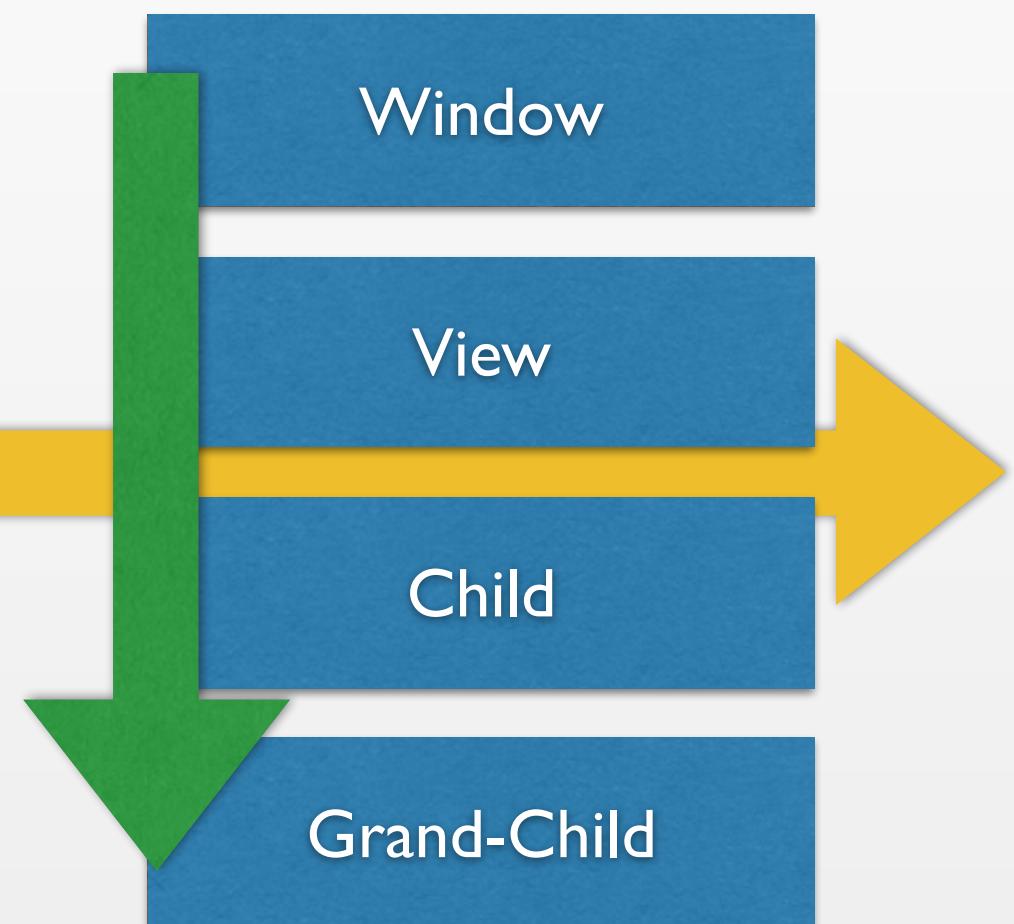
Update Constraints



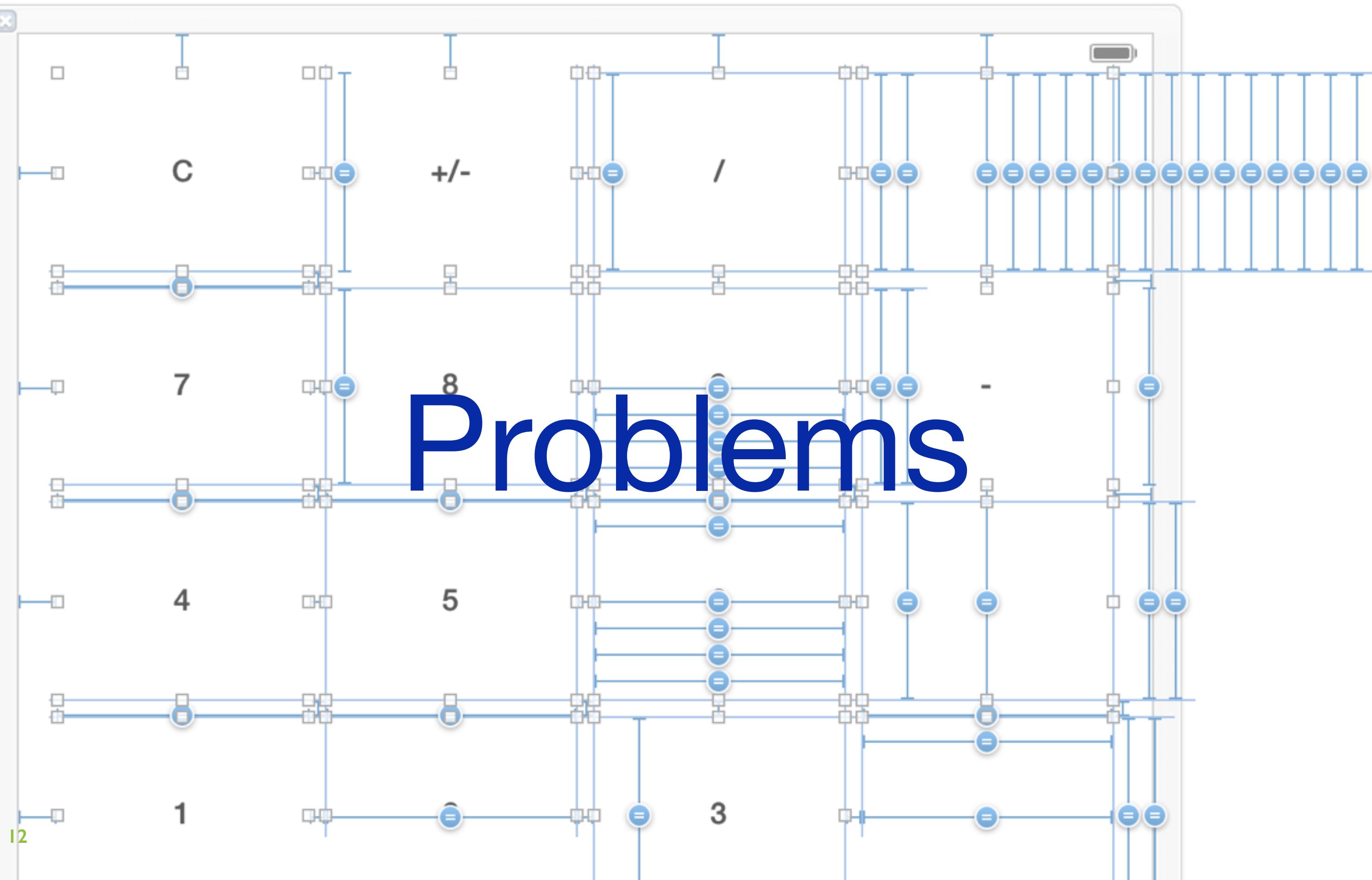
Layout



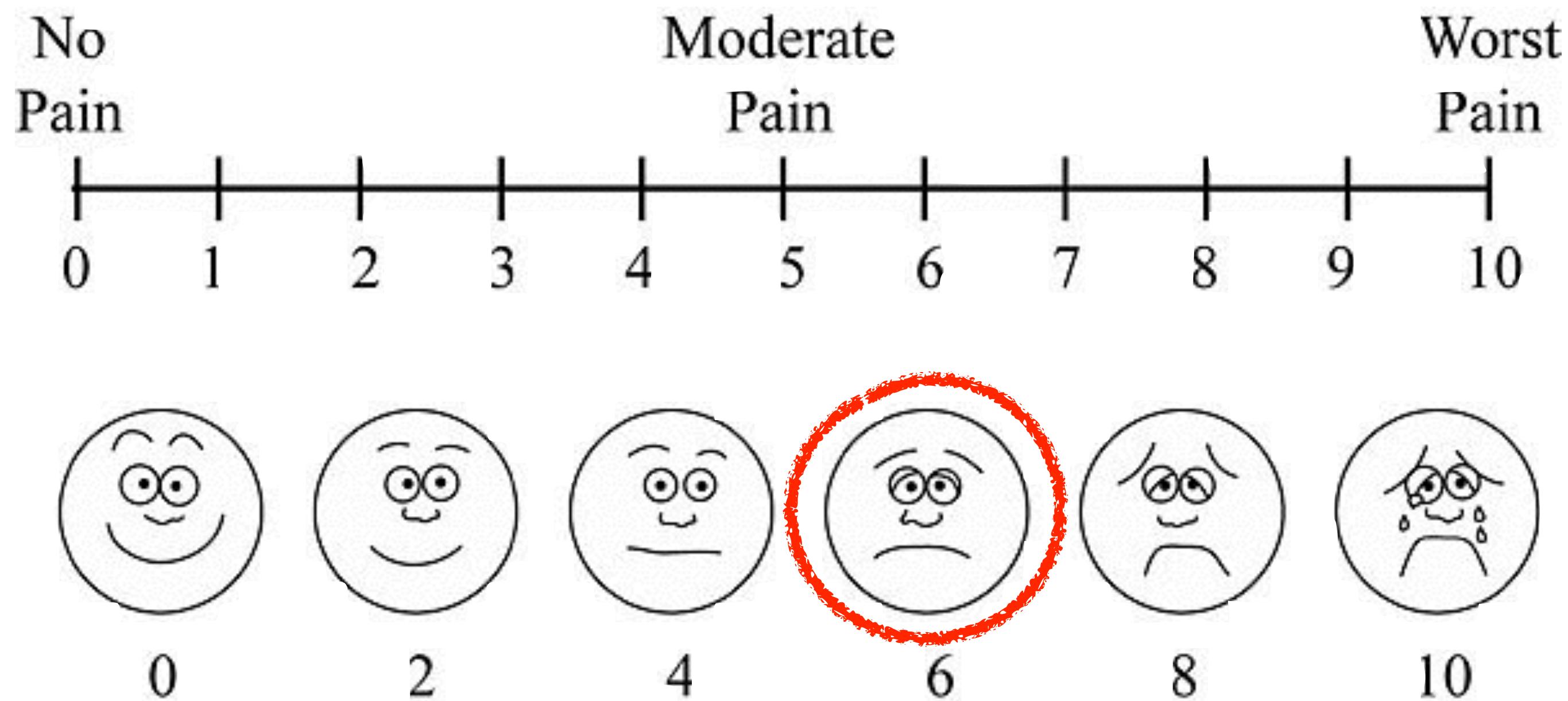
Render



Problems



ScrollViews



Auto Layout in ScrollViews

- Goals
 - Components resize
 - No manual calculation of contentSize
 - No conflicts or ambiguities

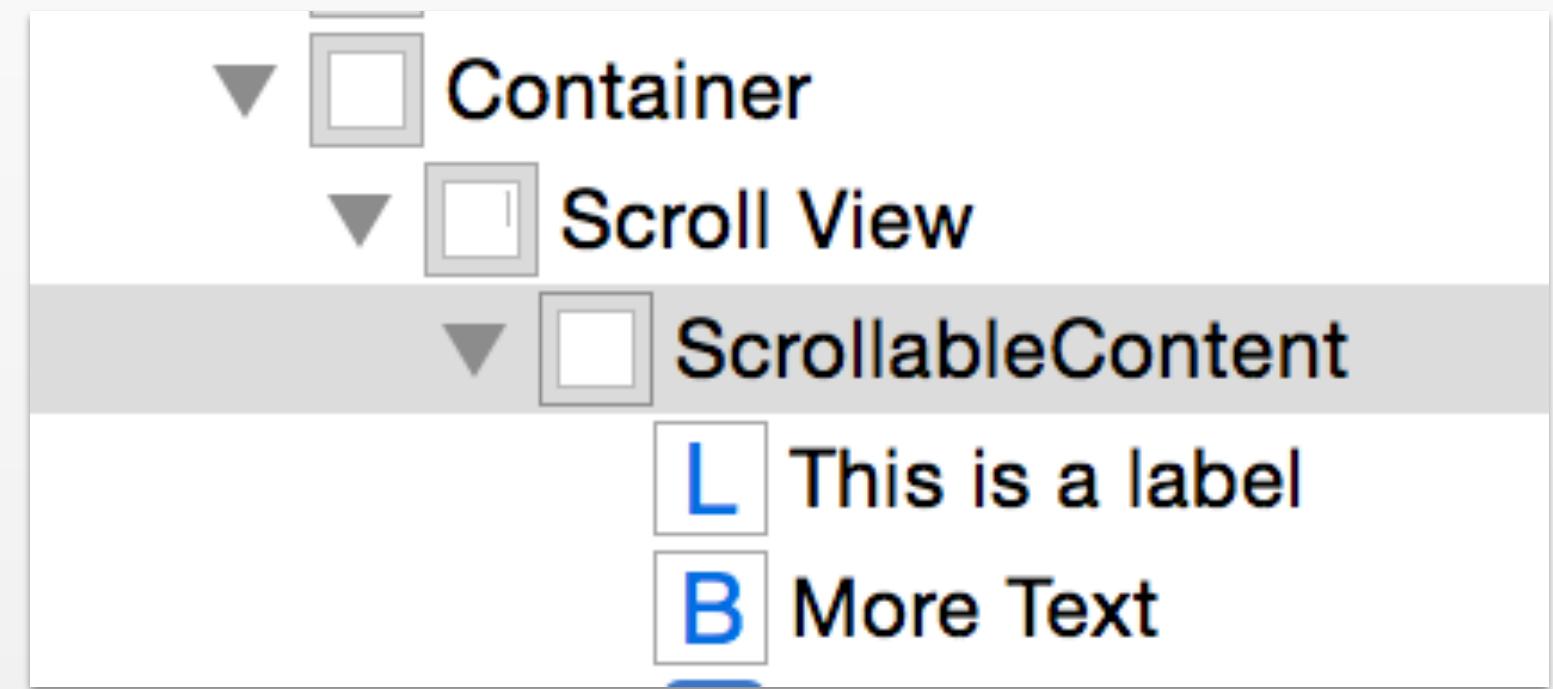
Fly in the ointment

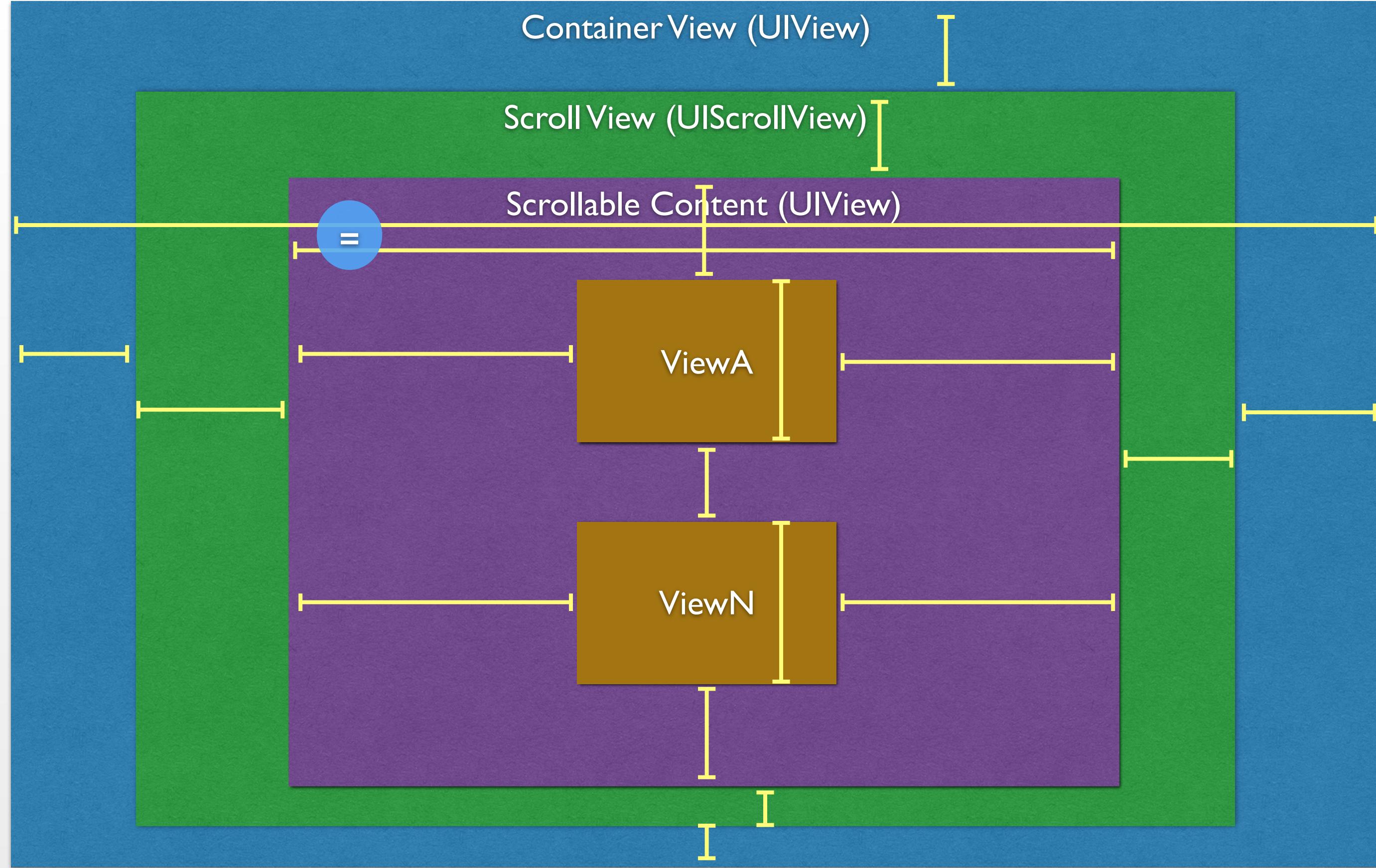
- ScrollViews don't have:
 - Height
 - Width
 - No intrinsic size



Approach

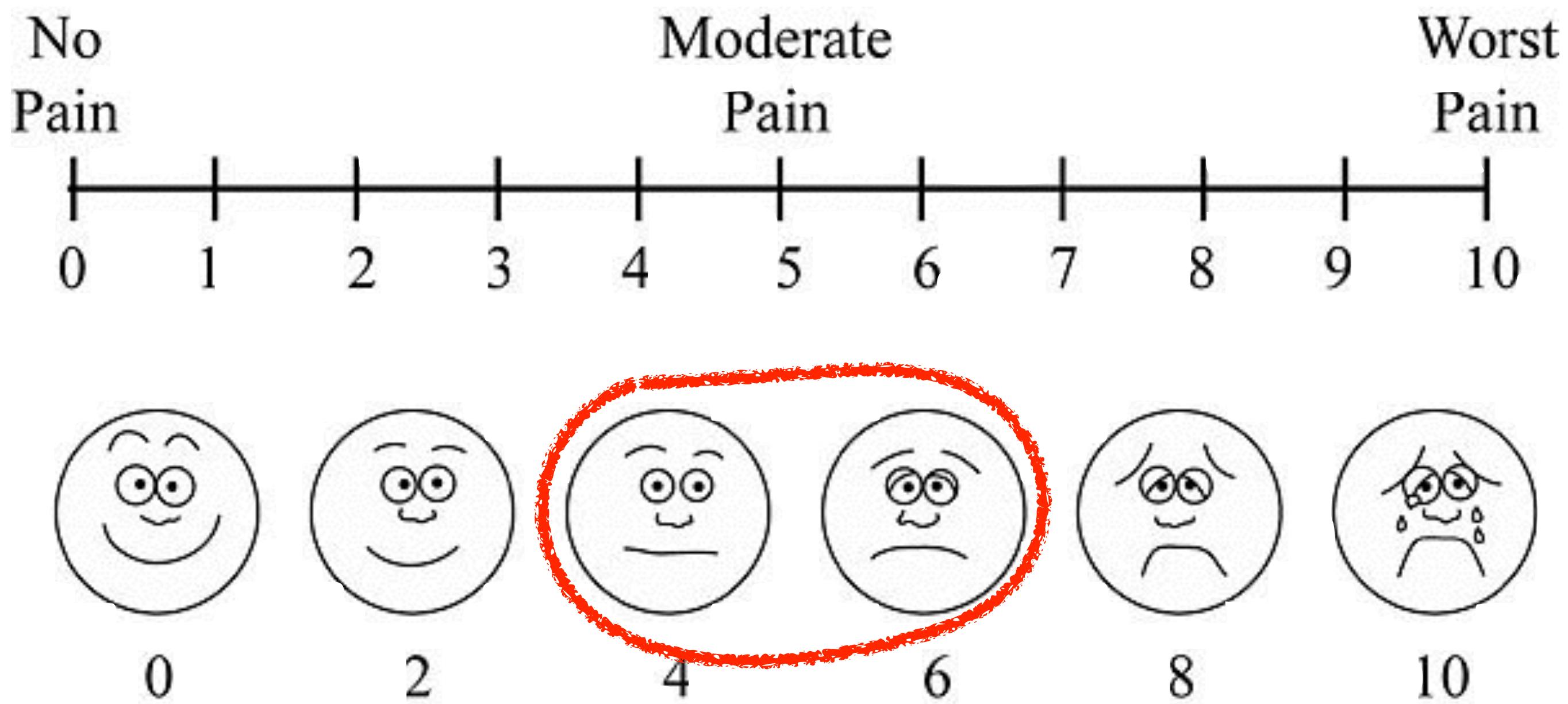
- Pure approach
- Technote: TN2154
- Leverages
 - Bi-directional
 - Content Compression Resistance





Code!

TableViews

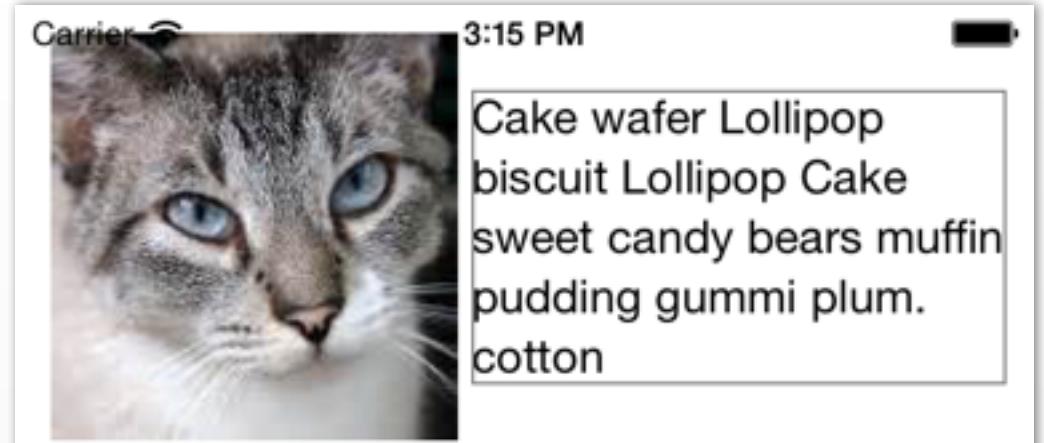


Autolayout In UITableViews

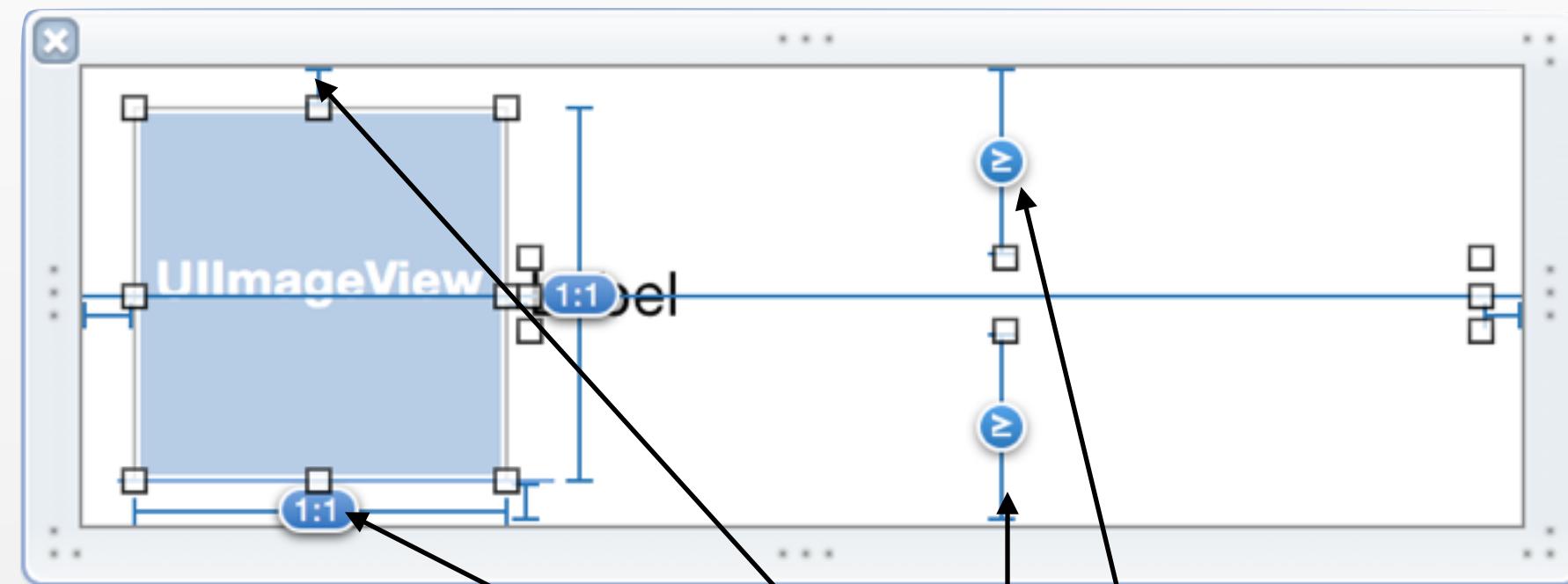
- Goals
- Rows that change size based on the content
- Avoid pre-render cell contents to calculate the size

Auto Layout in Tableviews

- Build a cell that is bi-directional
- Specify Estimated Size



Self-Sizing Cell



Constraints to set height

Code Changes

- In viewDidLoad()

```
tableView.estimatedRowHeight = 100.0  
tableView.rowHeight = UITableViewAutomaticDimension
```

Code!



Bugs!



UILabel Layout Widths

- Layout impacts wrapping
- Wrapping impacts intrinsic size
- Intrinsic size impacts layout



preferredMaxLayoutWidth

- Tells layout where to wrap
- Often known before layout
- Sometimes not
 - What to do then

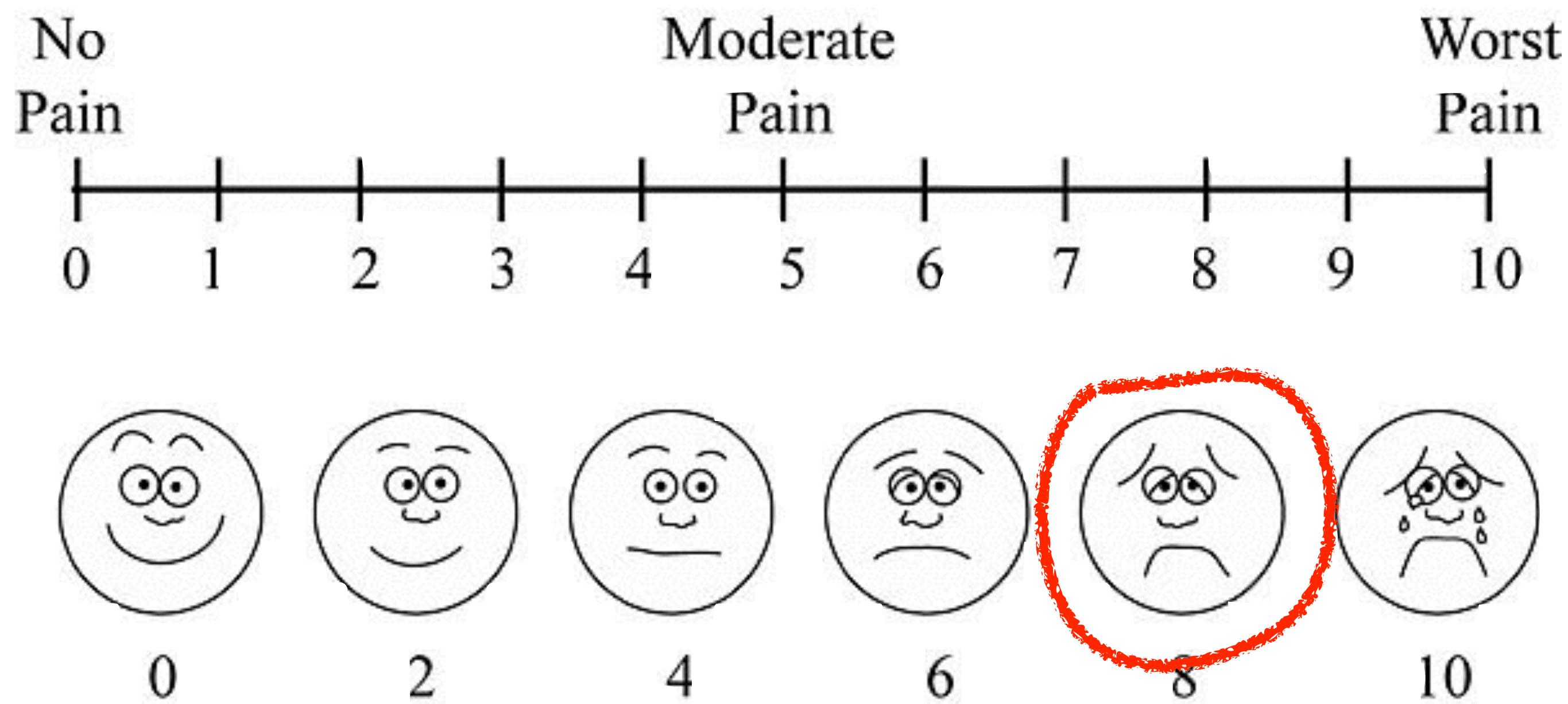


Using preferredMaxLayoutWidth

```
override func layoutSubviews() {  
    super.layoutSubviews()  
  
    title.preferredMaxLayoutWidth = title.frame.size.width  
  
    super.layoutSubviews()  
}
```

Code!

Collection Views



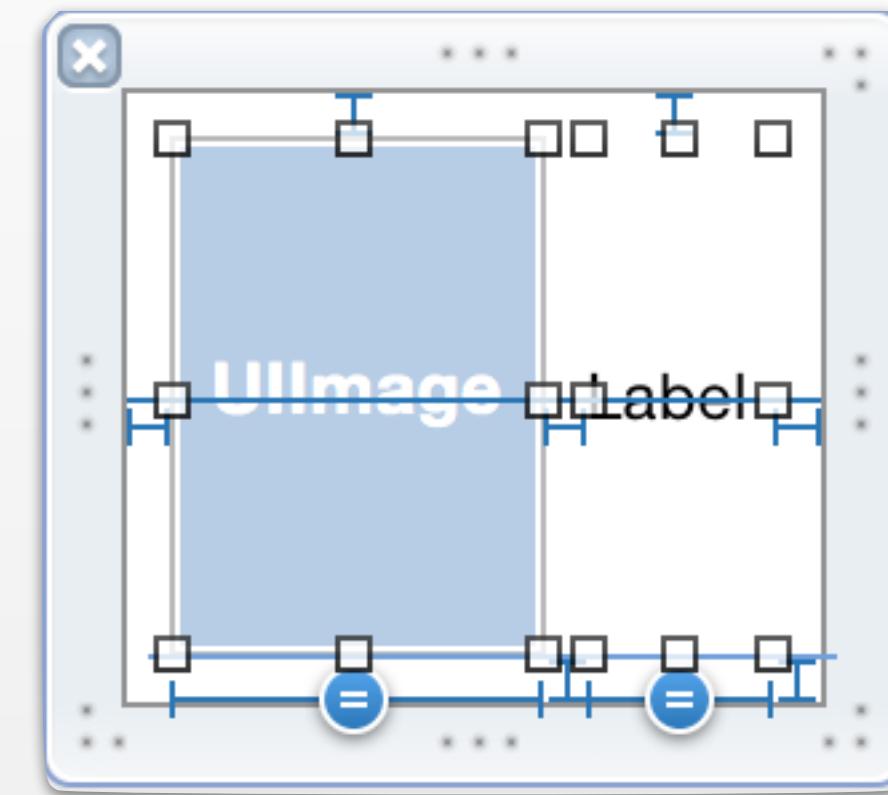
Auto layout in CollectionViews

- Create bi-directional cells.
- Provide an estimated size
- Do other unexpected things



Bi-Directional Self-sizing cell

- Self-size in 2 directions
- Text fields can be very fluid
 - Constrain them a bit

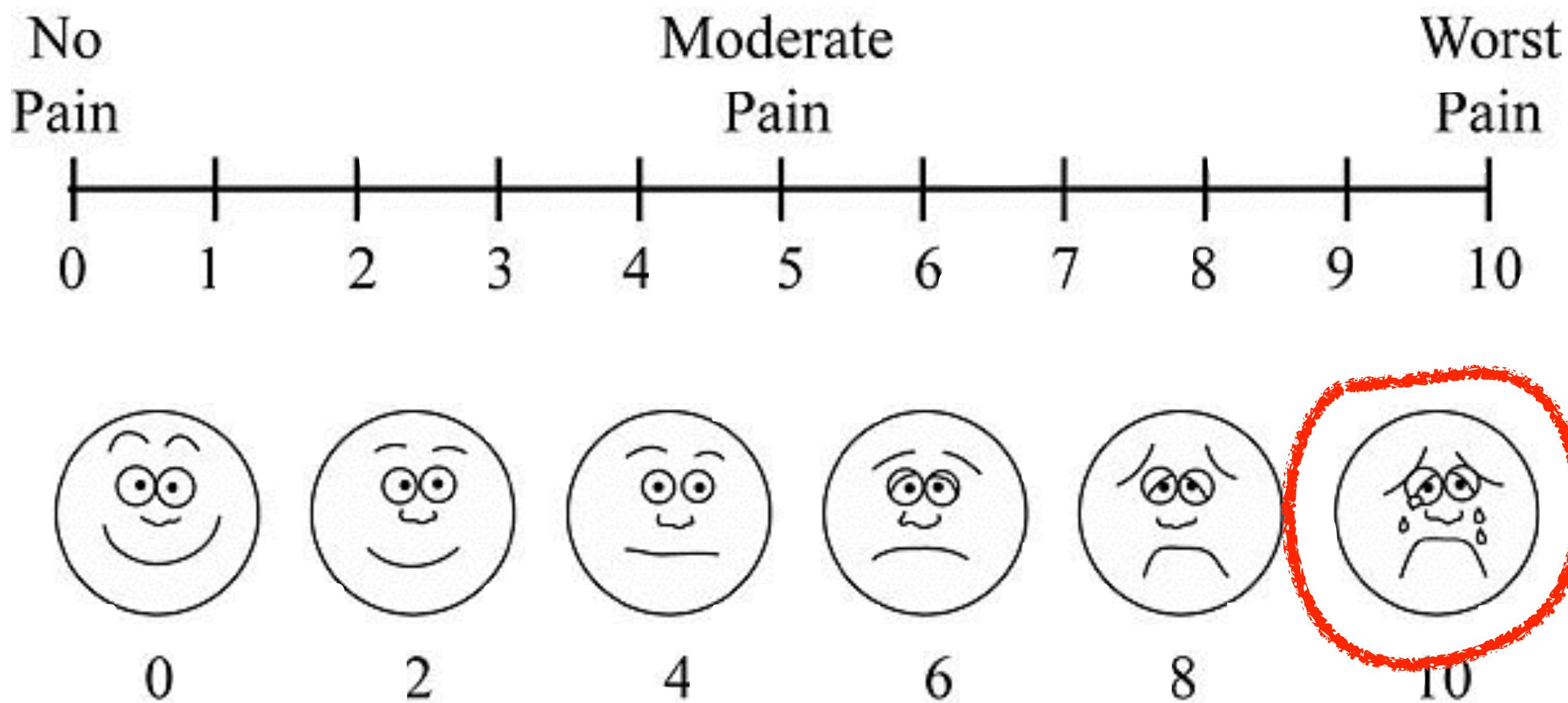


Pitfalls

- Estimated Size
 - Be close
- Need to limit sizes
- May need to reload on rotation

Code!

Debugging Auto Layout

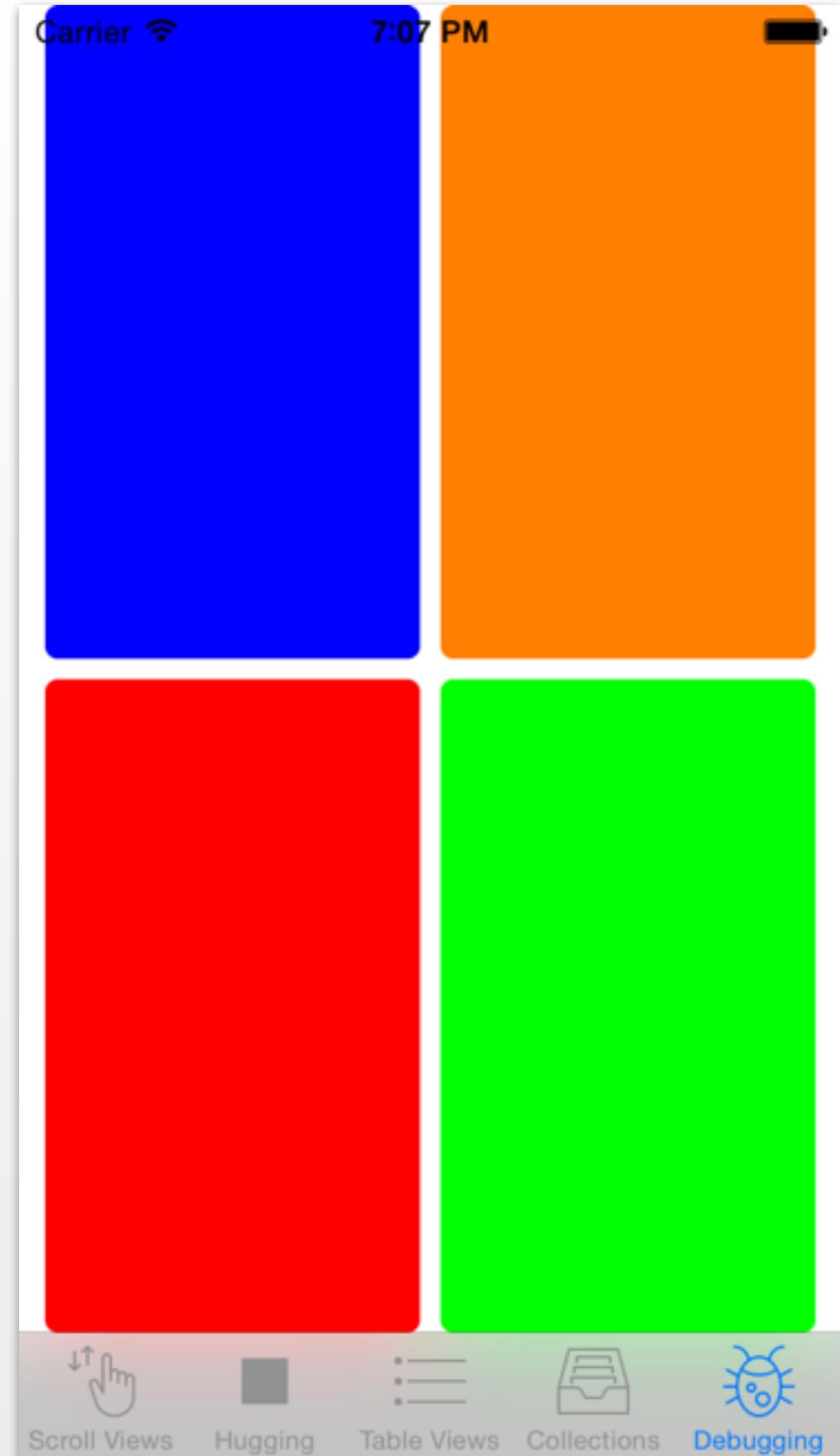


Conflicts & Ambiguity

- Conflicts
 - Over specification of constraints
- Ambiguity
 - Under specification of constraints

Debugging Example

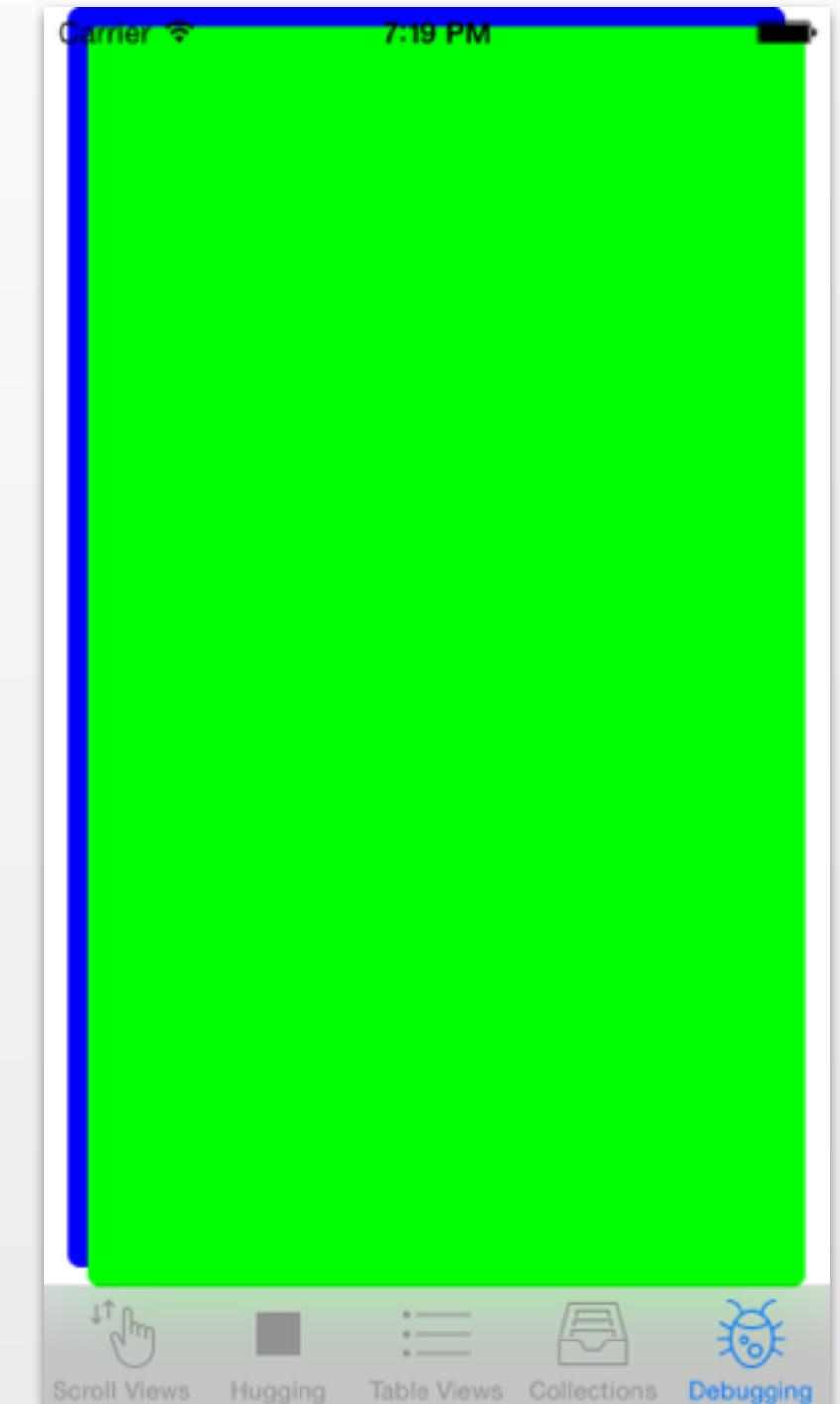
- Walkthrough
- Start with broken view
- Ambiguity First
- Conflicts Second



Steps to Debugging Ambiguity

- Detecting It
- 4 required constraints

✓ X
✓ Y
✓ Height
✓ Width



Detecting Ambiguity

- Autolayout Trace

```
(lldb) po [[UIWindow keyWindow] _autolayoutTrace]
```

Code

Detecting Ambiguity

- Exercising Ambiguity

```
@implementation UIView (AmbiguityHelpers)

- (void)exerciseAmbiguityInLayoutRepeatedly
{
#ifndef DEBUG
    if (self.hasAmbiguousLayout) {
        [NSTimer scheduledTimerWithTimeInterval:.5
                                         target:self
                                         selector:@selector(exerciseAmbiguityInLayout)
                                         userInfo:nil
                                         repeats:YES];
    }
    for (UIView *Subview in self.subviews) {
        [Subview exerciseAmbiguityInLayoutRepeatedly];
    }
#endif
}
@end
```

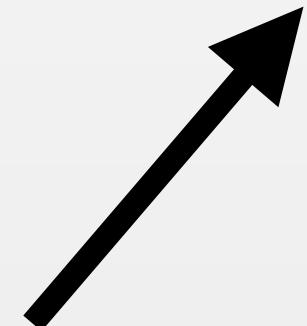
(lldb) expr (void)[(UIView *)<<view>> exerciseAmbiguityInLayoutRepeatedly]

Code

Identify Views

- Use Accessibility Identifiers

```
view.accessibilityIdentifier = "Some Great Name"
```



This will now show up in the debug views

Code

Steps to Resolving Ambiguity

- Identify Errant Views
- Identify Missing Constraints
- Start from the highest ancestor

Code

Steps to Debugging Conflicts

- Eliminate the easy things
- Identify Errant Views
- Work through the relations

Conflicts - Easy Things

- Make sure
`setTranslatesAutoResizingMasksToConstraints` is
false
- Views you instantiate
- Views you load from a XIB and insert into a
hierarchy

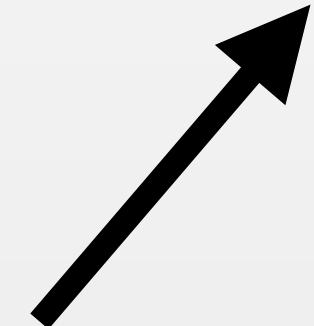
Identifying Offending Views

- Identify Constraints
- Enhancing the error messages
- Recursive Description
- Blinking the view

Identify Constraints

- Use Constraint Identifiers

```
constraint.identifier = "Some Great Name"
```



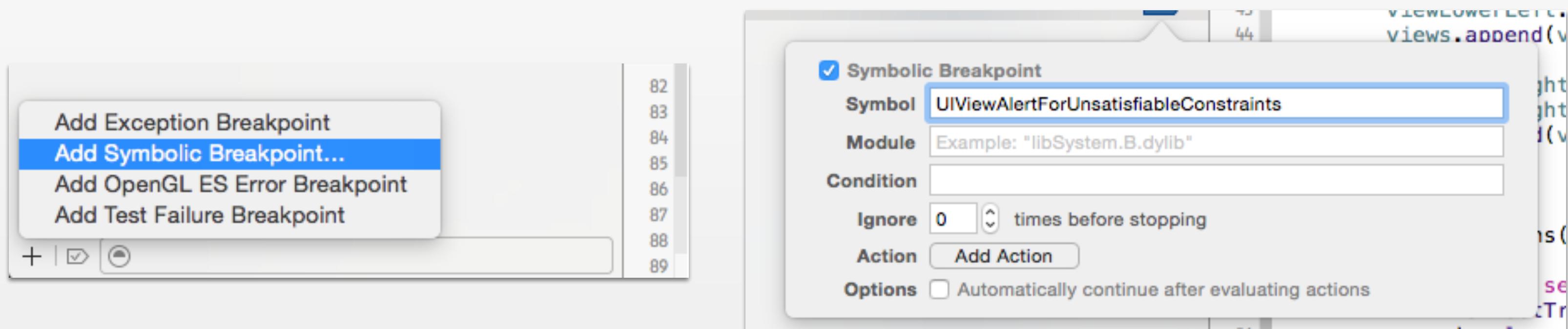
This will now show up in the debug views

Enhancing Error Messages

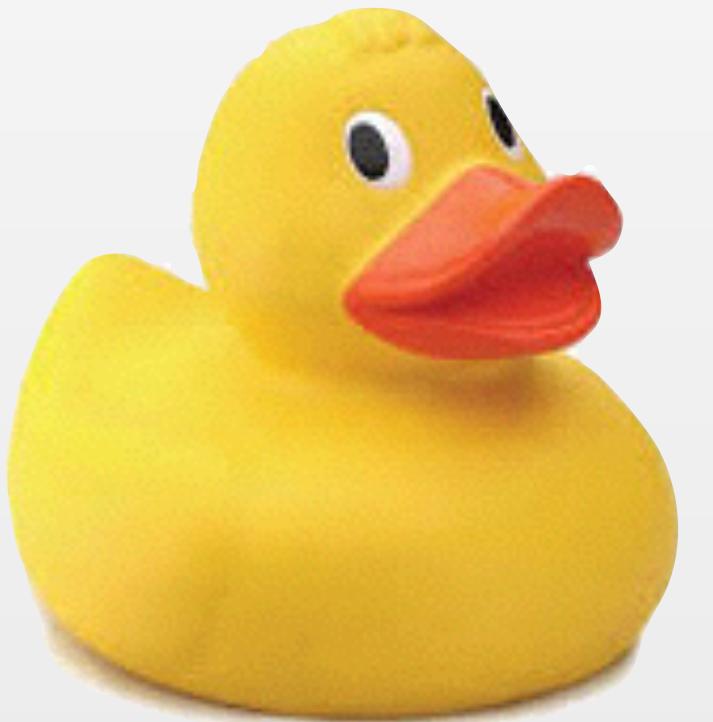
- Use accessibility IDs
- extend NSLayoutConstraint for more information
- <http://www.objc.io/issue-3/advanced-auto-layout-toolbox.html>

Breakpoints

- UIAlertViewForUnsatisfiableConstraints



Examining Relations



Examine the Message

```
(  
    "<NSLayoutConstraint:0x7fcf7c8b5890 H:|-(10)-[view-1]   (Names: view-1:0x7fcf7ac30370, container:0x7fcf7ac20d60, '|':container:0x7fcf7ac20d60 )>",  
    "<NSLayoutConstraint:0x7fcf7c8b5d60 H:[view-1]-(0)-[view-2]   (Names: view-2:0x7fcf7c814aa0, view-1:0x7fcf7ac30370 )>",  
    "<NSLayoutConstraint:0x7fcf7c8b5db0 view-2.width == view-1.width   (Names: view-2:0x7fcf7c814aa0, view-1:0x7fcf7ac30370 )>",  
    "<NSLayoutConstraint:0x7fcf7c8b6030 H:[view-2]-(10)-|   (Names: container:0x7fcf7ac20d60, view-2:0x7fcf7c814aa0, '|':container:0x7fcf7ac20d60 )>",  
    "<NSLayoutConstraint:0x7fcf7c8b76f0 H:|-(10)-[view-3]   (Names: view-3:0x7fcf7c8ae7f0, container:0x7fcf7ac20d60, '|':container:0x7fcf7ac20d60 )>",  
    "<NSLayoutConstraint:0x7fcf7c8b7af0 view-3.width == view-1.width   (Names: view-3:0x7fcf7c8ae7f0, view-1:0x7fcf7ac30370 )>",  
    "<NSLayoutConstraint:0x7fcf7c8b7b70 H:[view-3]-(NSSpace(8))-[view-4]   (Names: view-4:0x7fcf7c8aeac0, view-3:0x7fcf7c8ae7f0 )>",  
    "<NSLayoutConstraint:0x7fcf7c8b7bc0 view-4.width == view-1.width   (Names: view-4:0x7fcf7c8aeac0, view-1:0x7fcf7ac30370 )>",  
    "<NSLayoutConstraint:0x7fcf7c8b7c10 H:[view-4]-(10)-|   (Names: container:0x7fcf7ac20d60, view-4:0x7fcf7c8aeac0, '|':container:0x7fcf7ac20d60 )>"  
)  
  
Will attempt to recover by breaking constraint  
<NSLayoutConstraint:0x7fcf7c8b7b70 H:[view-3]-(NSSpace(8))-[view-4]   (Names: view-4:0x7fcf7c8aeac0, view-3:0x7fcf7c8ae7f0 )>  
  
Axis Indicator  
Container Edges  
View Names  
View Details  
Sizing Constraints  
Spacing
```

$$4x - 2 + 2x = 6x - 12 + 10$$

$$6x - 2 = 6x - 2$$

All values of x

$$13. -2x + 6 \leq 4$$

$$-2x \leq -2$$



$$16. x \geq -4 \quad x < 2$$



$$4x - 18 + 12x = 4x - 8 - 4 - 10x \Rightarrow x = 6$$

$$\begin{array}{r} 16x - 18 = -6x - 12 \\ + 6x + 18 \quad + 6x + 18 \\ \hline 22x = 6 \end{array}$$

$$x = \frac{6}{22} \text{ or } \frac{3}{11}$$

$$11. |2x - 3| = -7$$

No SOLUTION

$$12. |2x - 3| = 7$$

$$2x - 3 = 7 \quad 2x - 3 = -7$$

$$2x = 10 \quad 2x = -4$$

$$x = 5 \quad x = -2$$



Algebra

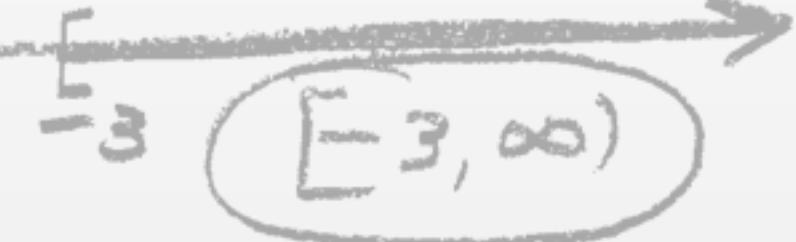
$$17. (5x - 4y)^2$$

$$a) 25x^2 - 40xy + 16y^2$$

$$b) [5x - 4y - 6][5x - 4y + 4]$$

$$(5x - 4y)^2 - (2x - 4y) = 28$$

a) or



Making the Formulas

```
H: |-(10)-[view-1]
H: [view-1]-(0)-[view-2]
view-2.width == view-1.width
H: [view-2]-(10)-|
H: |-(10)-[view-3]
view-3.width == view-1.width
H: [view-3]-(NSSpace(8))- [view-4]
view-4.width == view-1.width
H: [view-4]-(10)-|
```

$$10 + \text{view-1}_w + \text{view-2}_w + 10$$

$$10 + \text{view-3}_w + 8 + \text{view-4}_w + 10$$

$$\begin{aligned}\text{view-1}_w &== \text{view-2}_w \\ \text{view-1}_w &== \text{view-3}_w \\ \text{view-1}_w &== \text{view-4}_w\end{aligned}$$

Demo

System Generated Constraints

UIView-Encapsulated-Layout-Width

UIView-Encapsulated-Layout-Height

Demo



Wrap up

- Constraints Are Bi-Directional and have intrinsic size
- Estimated Size for Table Views
- Preferred Layout Width for Text
- Identify your views and constraints
- Debug Methodically
- Break down the problem

Conclusion

Questions?

Source:

<https://github.com/jack-cox-captech/ALProblems>

Contact Info

jcox@captechconsulting.com

@jcox_mobile