Solving Problems in Auto Layout

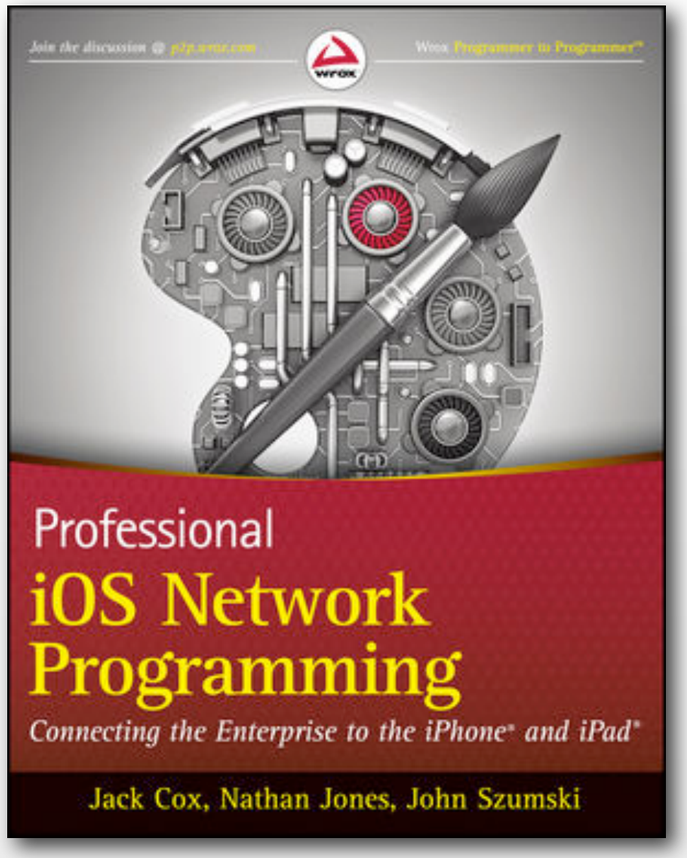
Please, just make the pain stop.

Introduction

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

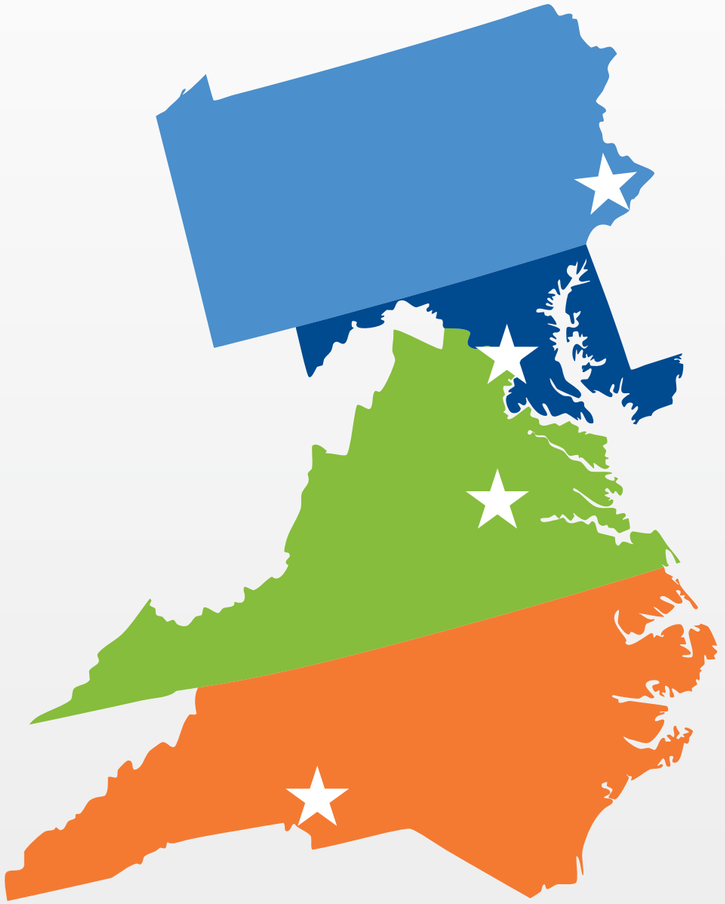
  jcox@captechconsulting.com

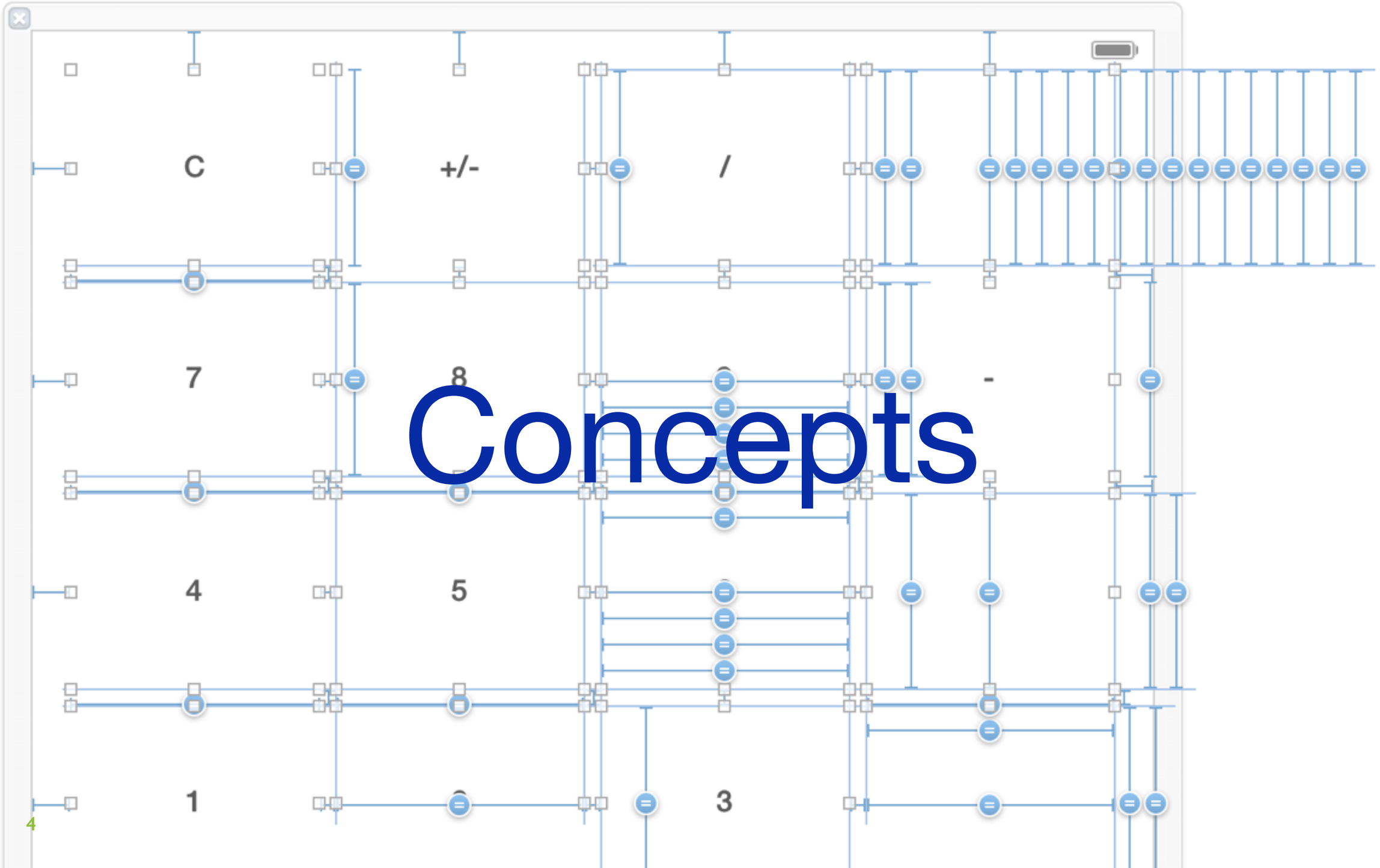
  @jcox_mobile



Who is CapTech

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

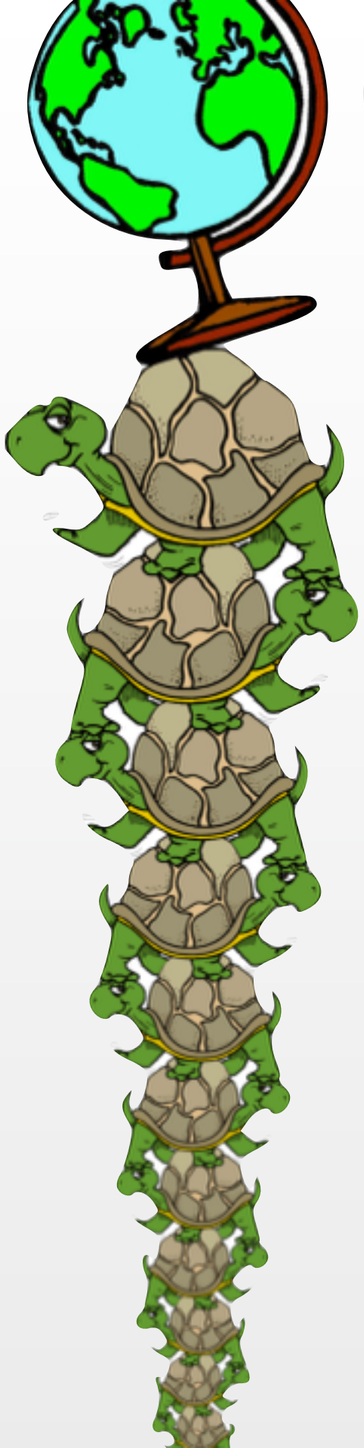



C
+/-
/
7
8
-
4
5
1
3
Concepts

We Hate Auto Layout

- We don’t feel in control
- It never seems to cooperate
- Feel like it hates us

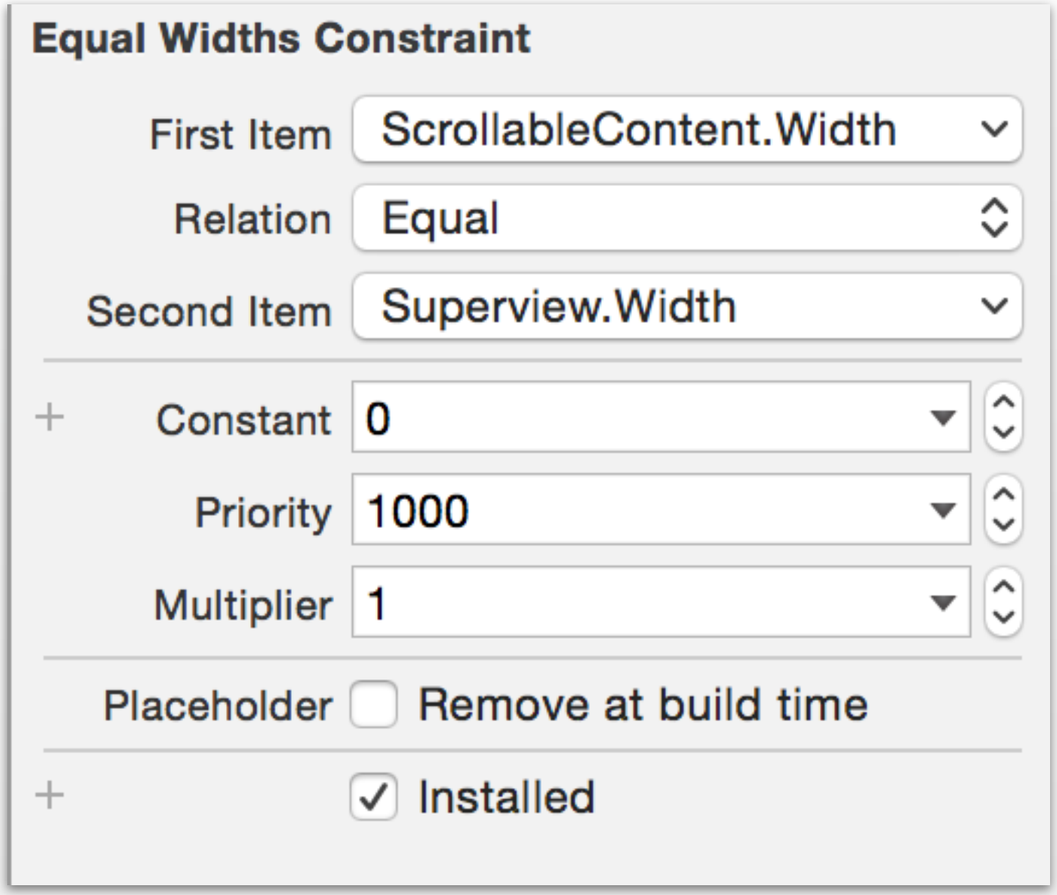
Concepts



- View Hierarchy
  - Any 2 views can affect one another

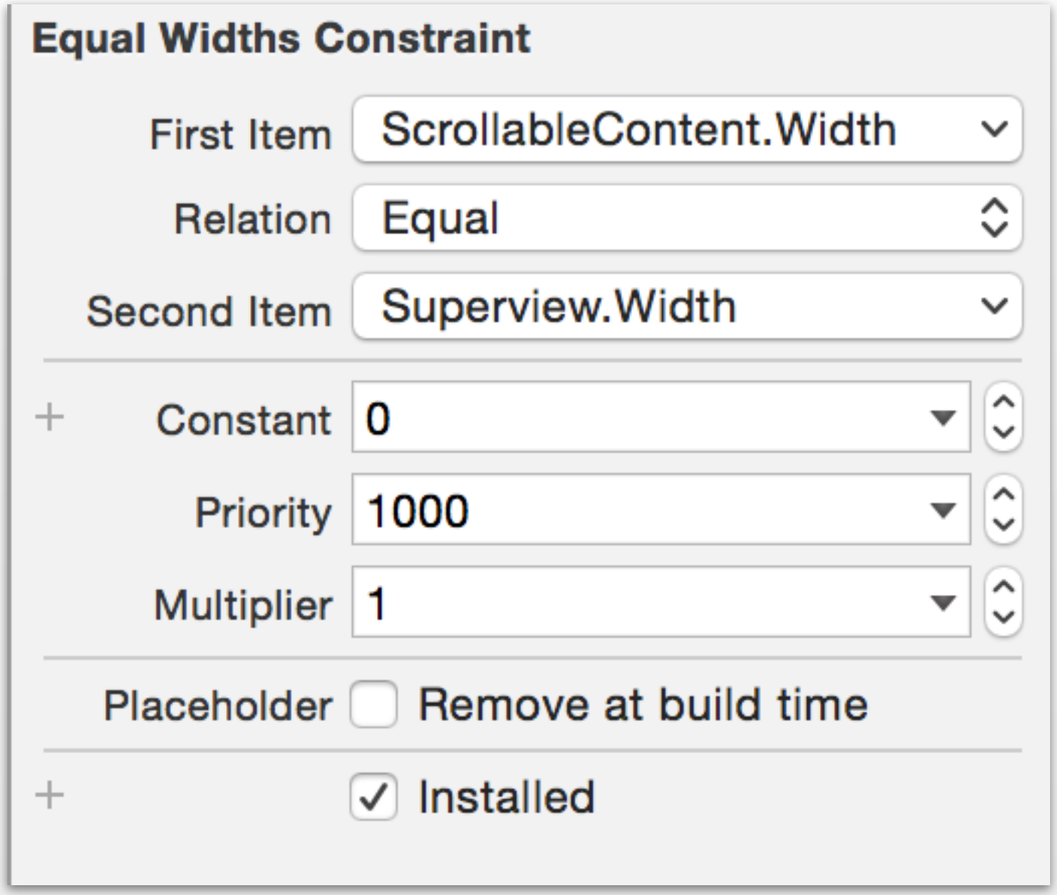
Concepts

- Constraint



Mutable Properties

- Constant
- Priority
- Hugging priority
- Resistance priority

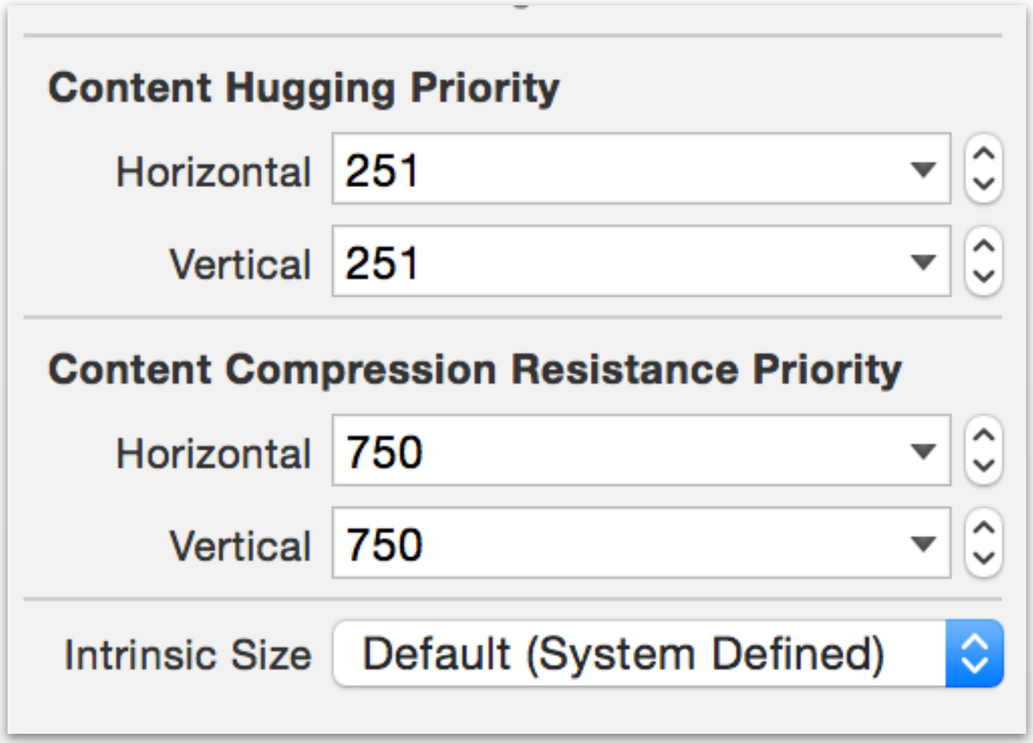


Intrinsic Size

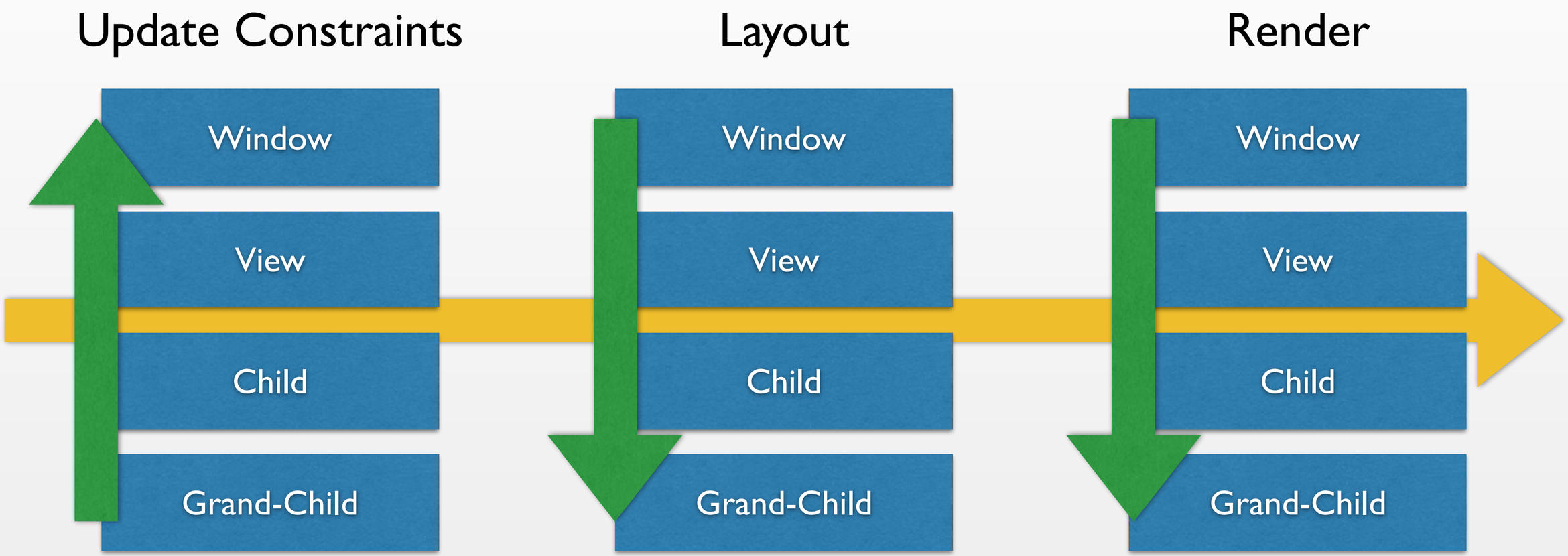
- **How big the view wants to be**
- Based on contents of view
- Child views can imbue their intrinsic size on parents
- Most standard controls have intrinsic size
- Views can define their own intrinsic size

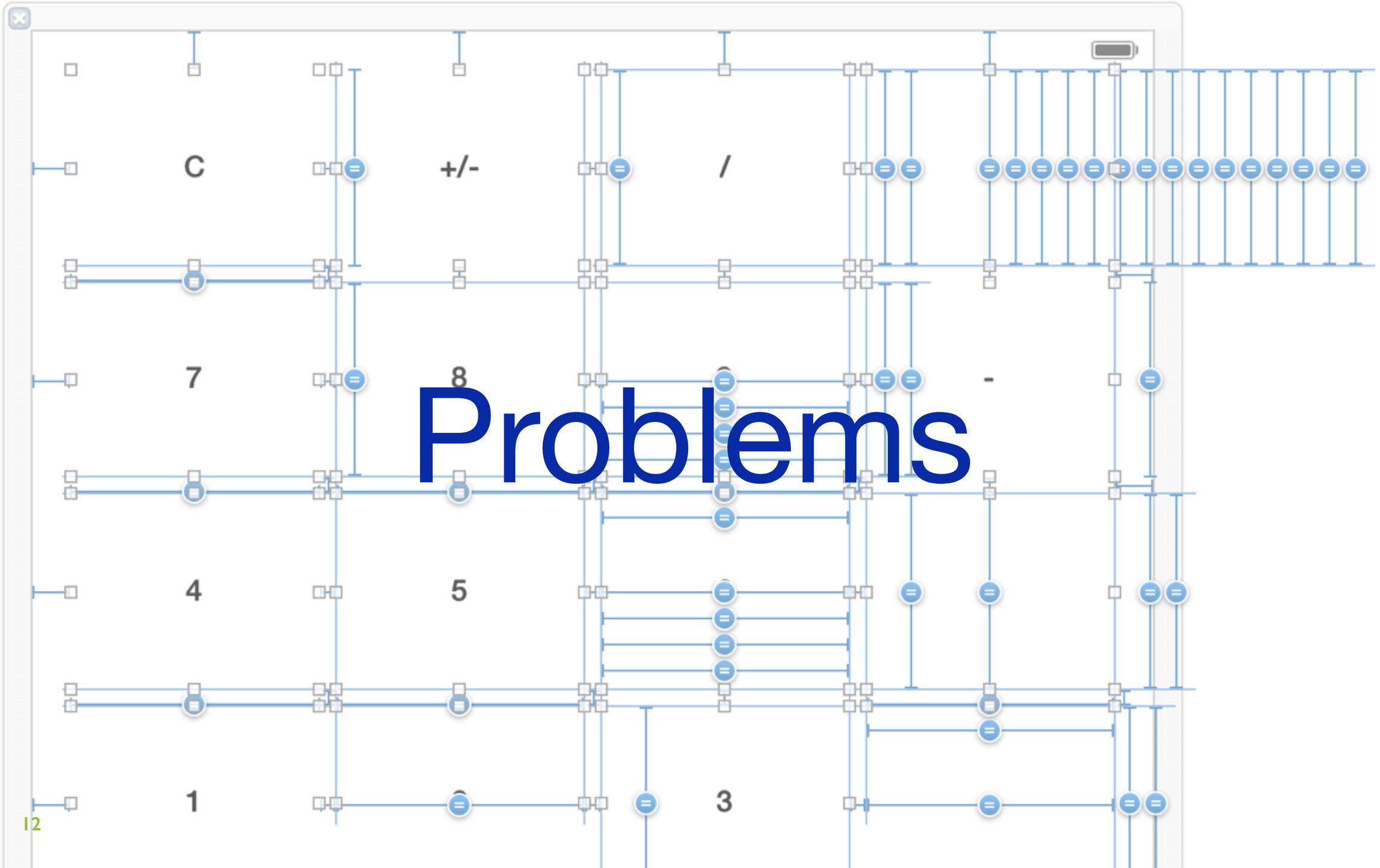
Hugging and Resistance

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



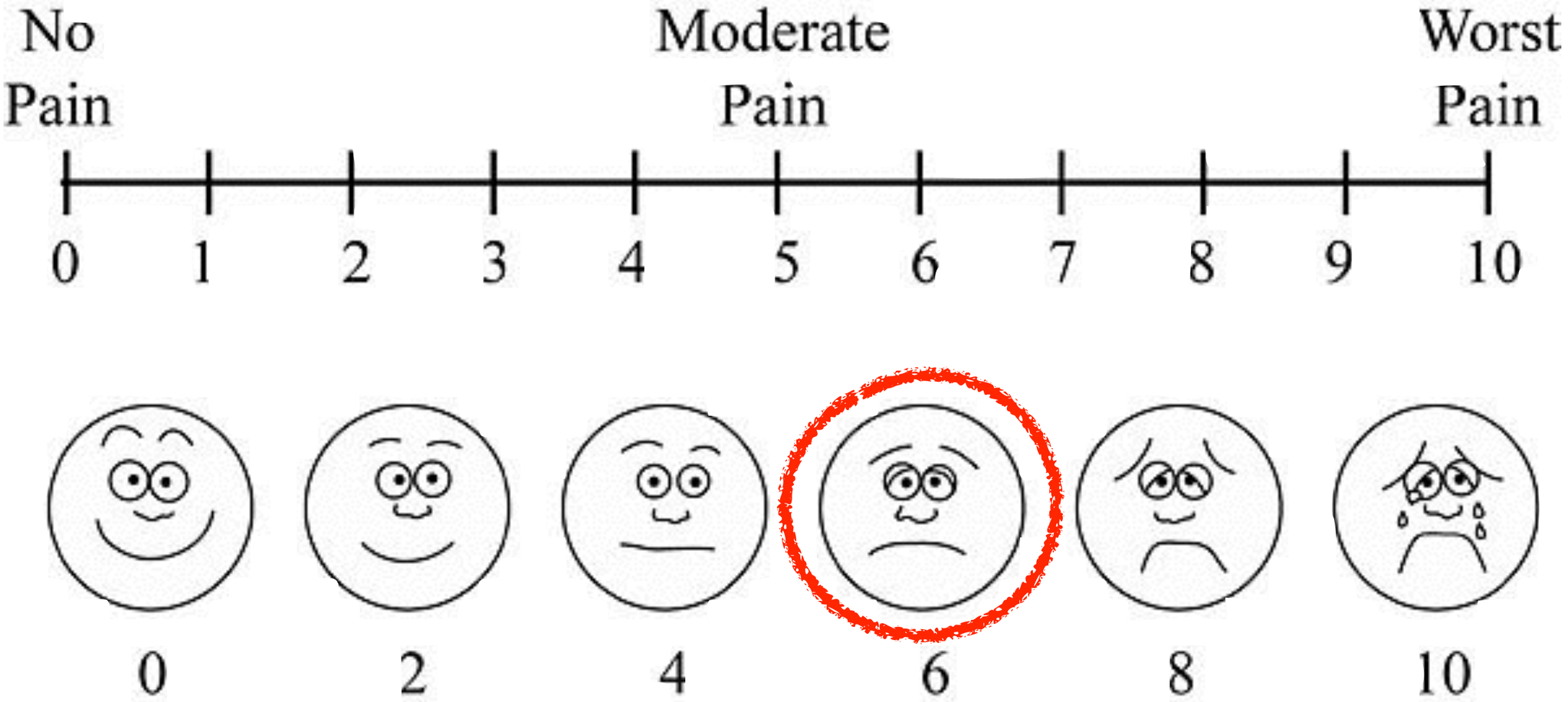
Rendering Cycle





Problems

ScrollViews



Auto Layout in ScrollViews

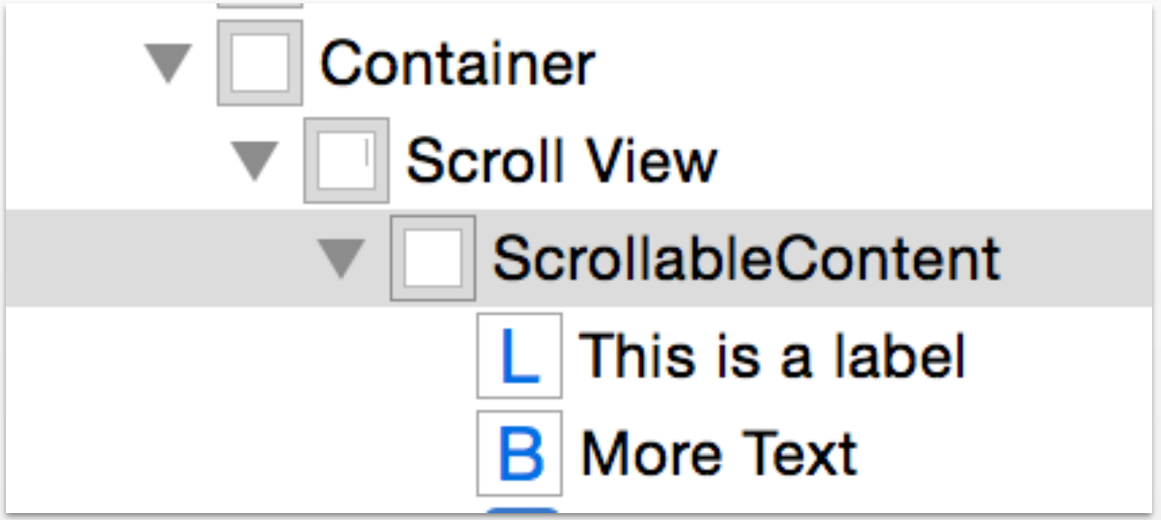
- Goals
  - Components resize
  - No manual calculation of contentSize
  - No conflicts or ambiguities

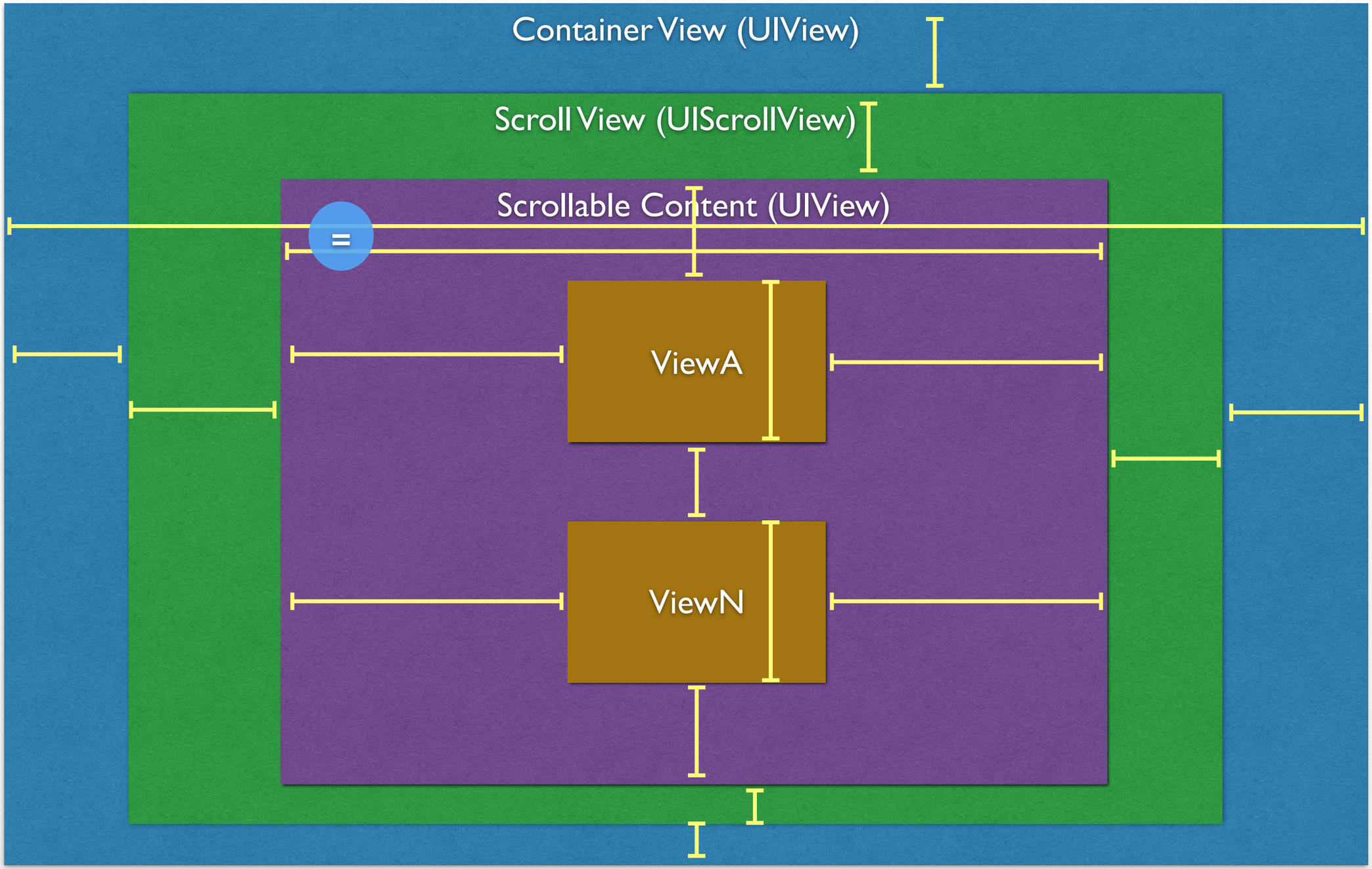
Fly in the ointment

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

Approach

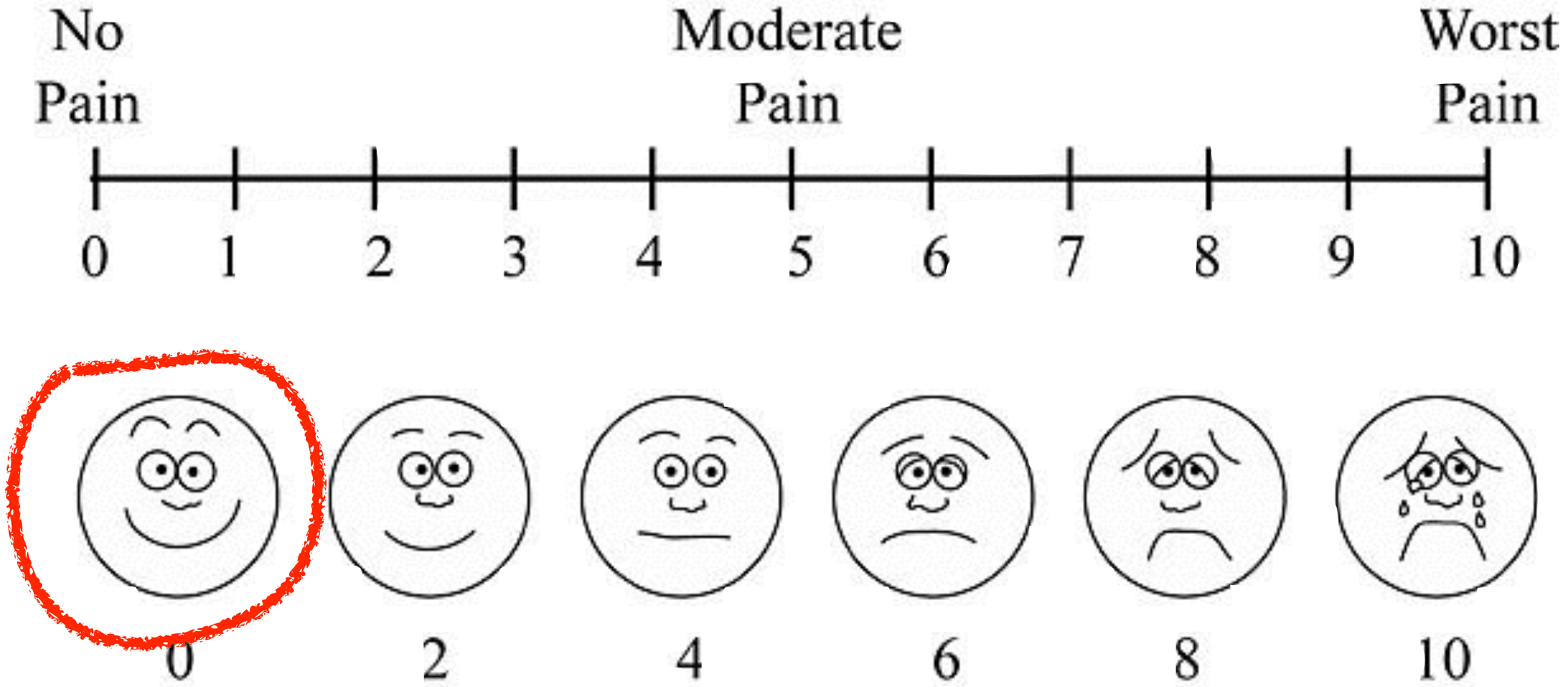
- Pure approach
  - Technote: TN2154
- Leverages
  - Bi-directional
  - Content Compression Resistance




Container View (UIView)
Scroll View (UIScrollView)
Scrollable Content (UIView)
=
ViewA
ViewN

Code!

iOS 9 StackViews

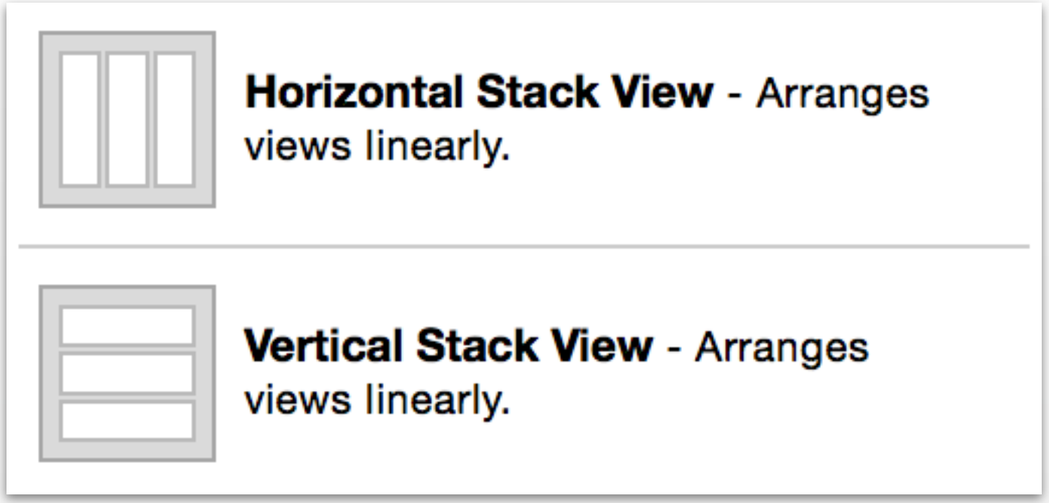


StackViews

- Goals
  - Make Auto layout simple
  - Support interesting and complex layouts

StackView Internals

- No New Magic
- Horizontal or Vertical
- Arranged Views vs Sub Views



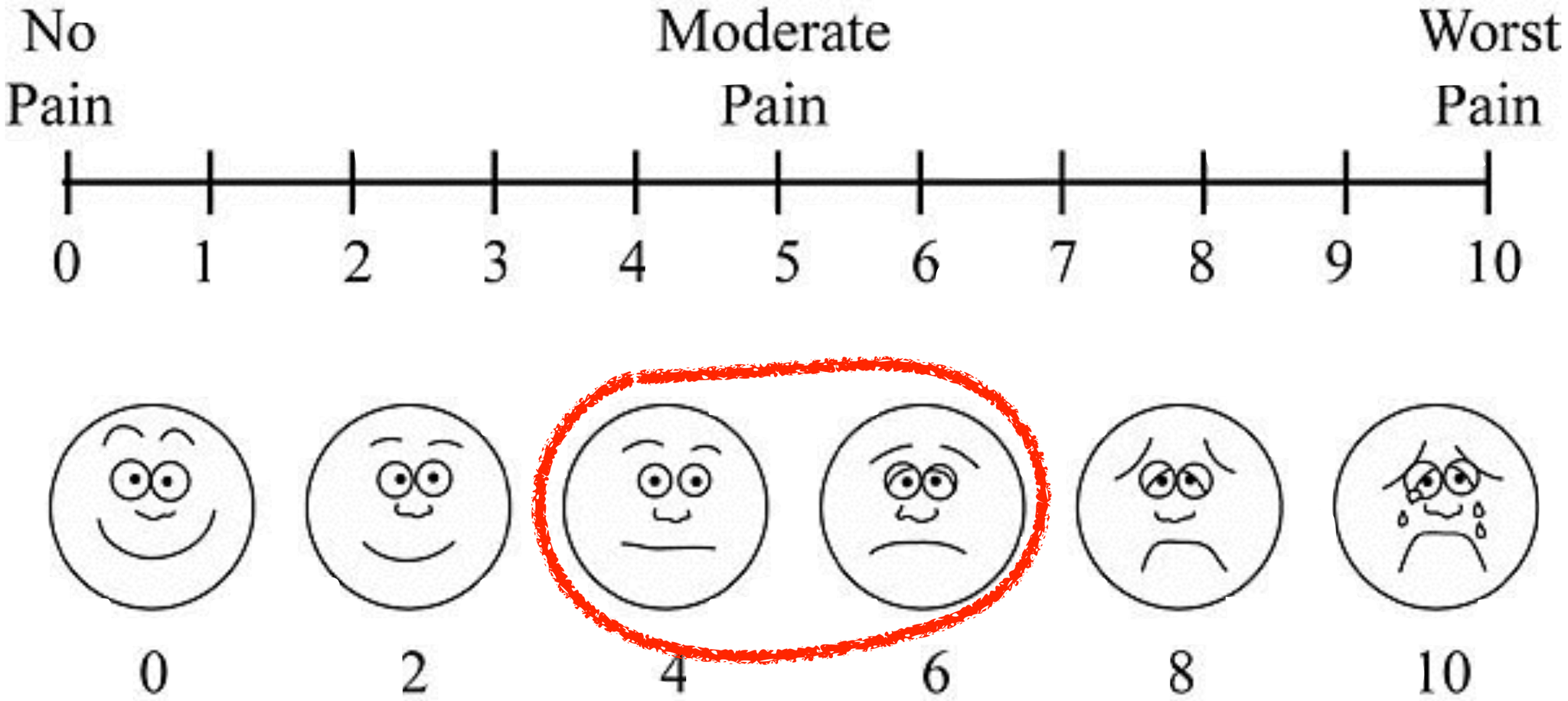
StackView Problems

StackView Warnings

- No Intrinsic Size
- Doesn’t Replace TableViews
- Beware of adding constraints
- Removing arranged views

Code!

TableViews

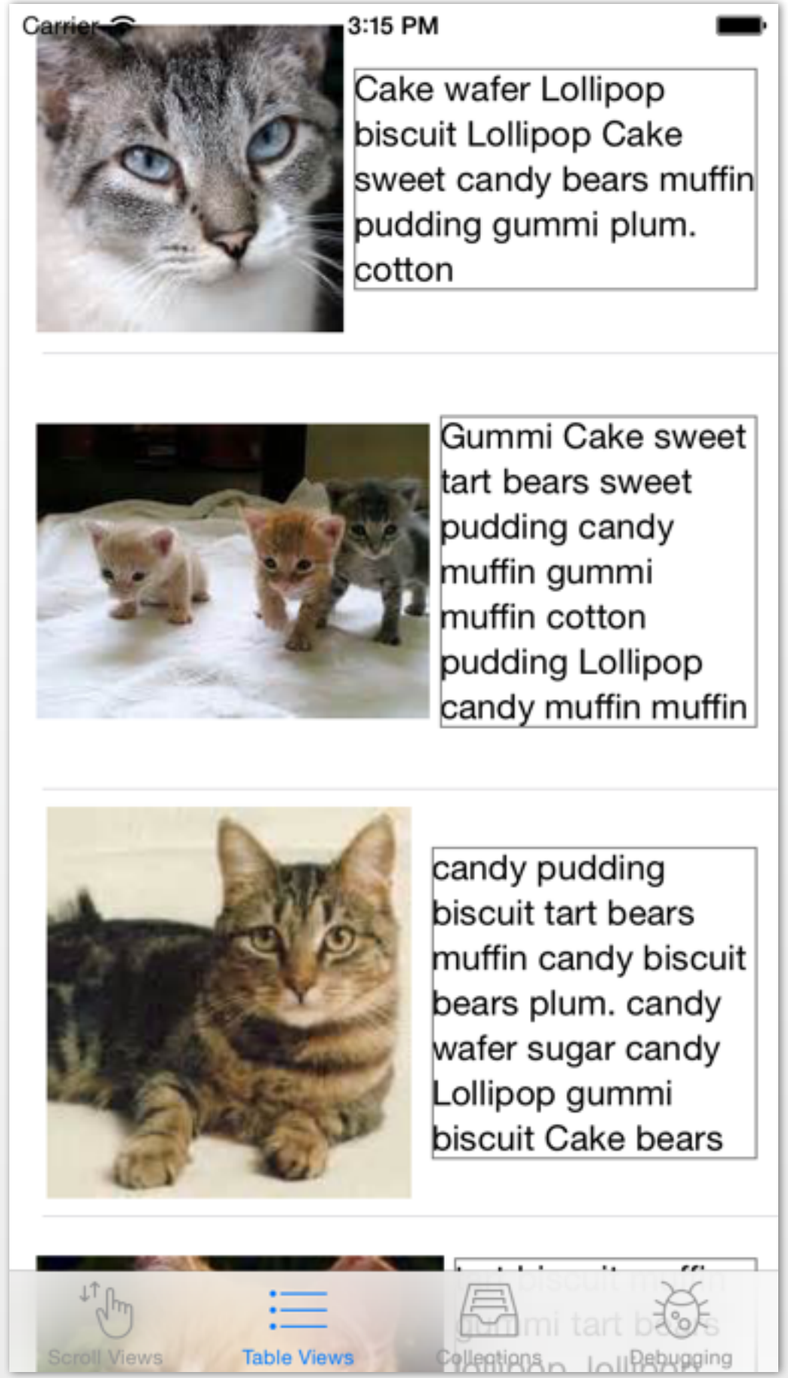


Auto Layout In TableViews

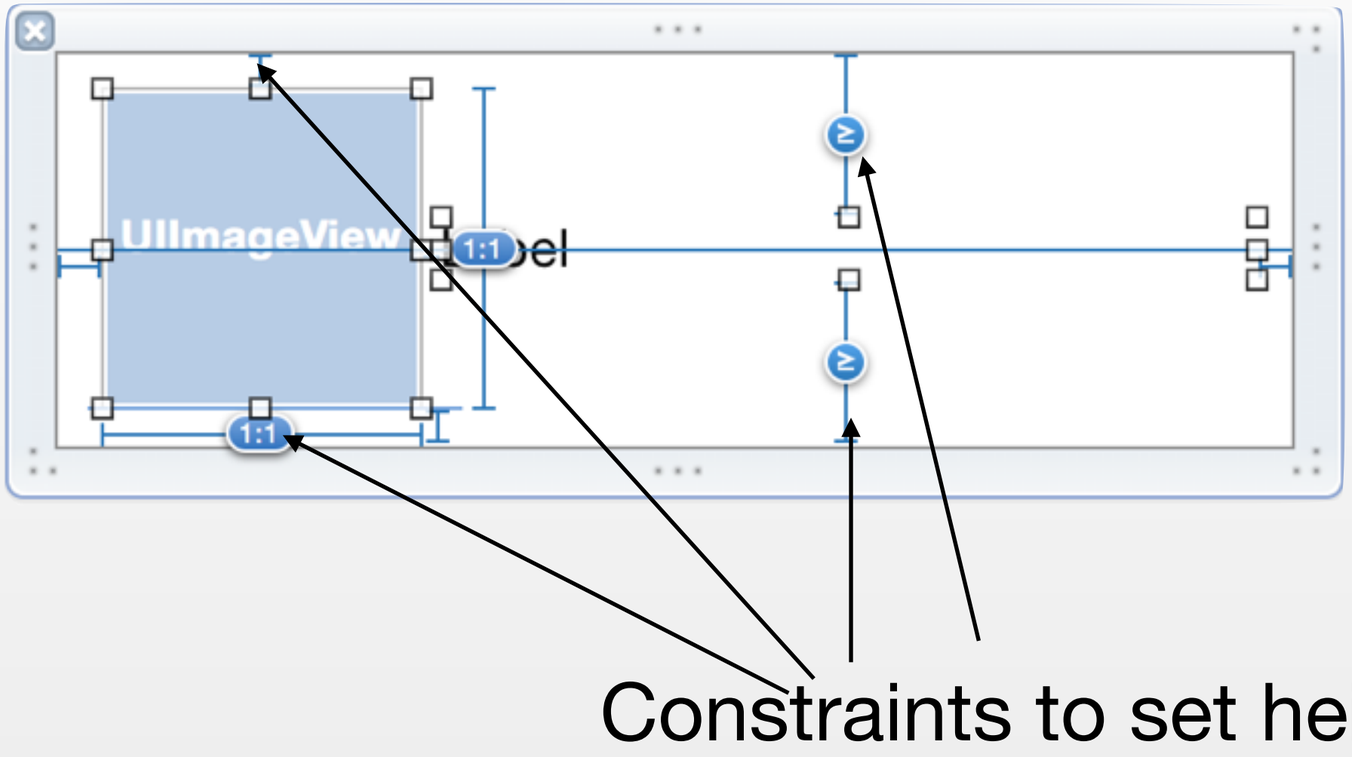
- Goals
  - Rows that change size based on the content
  - Avoid pre-render cell contents to calculate the size

Auto Layout in Tableviews

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



Self-Sizing Cell



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

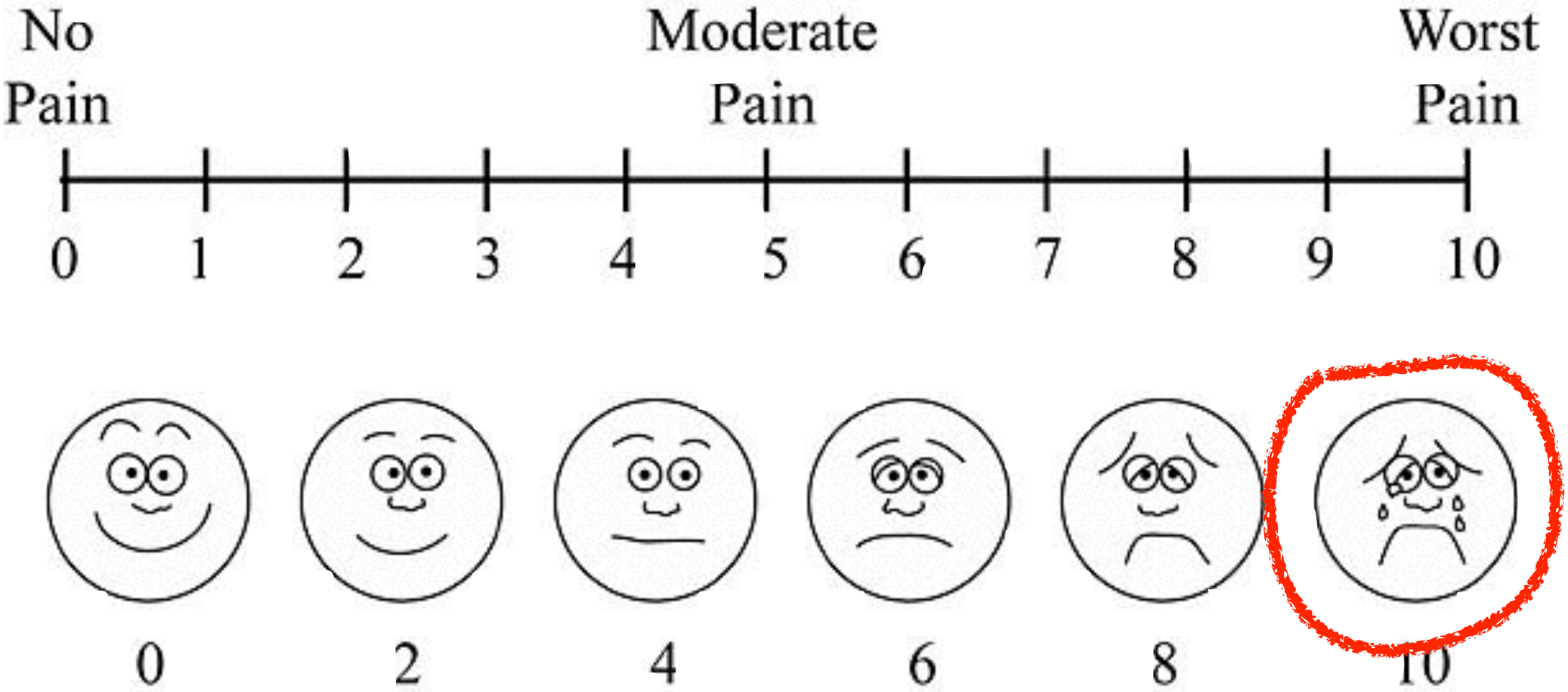
- Tells layout where to wrap
- Often known before layout
- Sometimes not
  - What to do then

Using preferredMaxLayoutWidth

```
override func layoutSubviews() {
    super.layoutSubviews()

    title.preferredMaxLayoutWidth = title.frame.size.width

    super.layoutSubviews()
}
```

Code!

Debugging Auto Layout

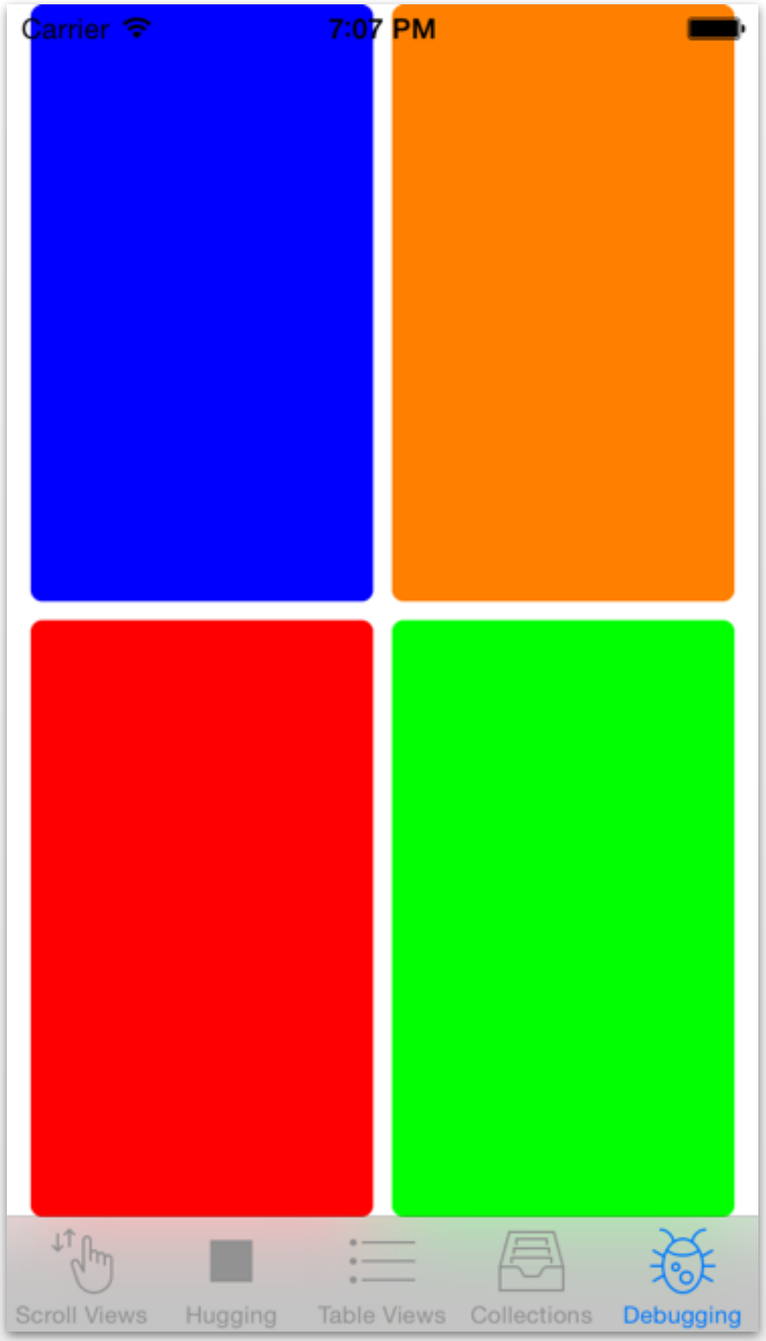


Conflicts & Ambiguity

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

Debugging Example

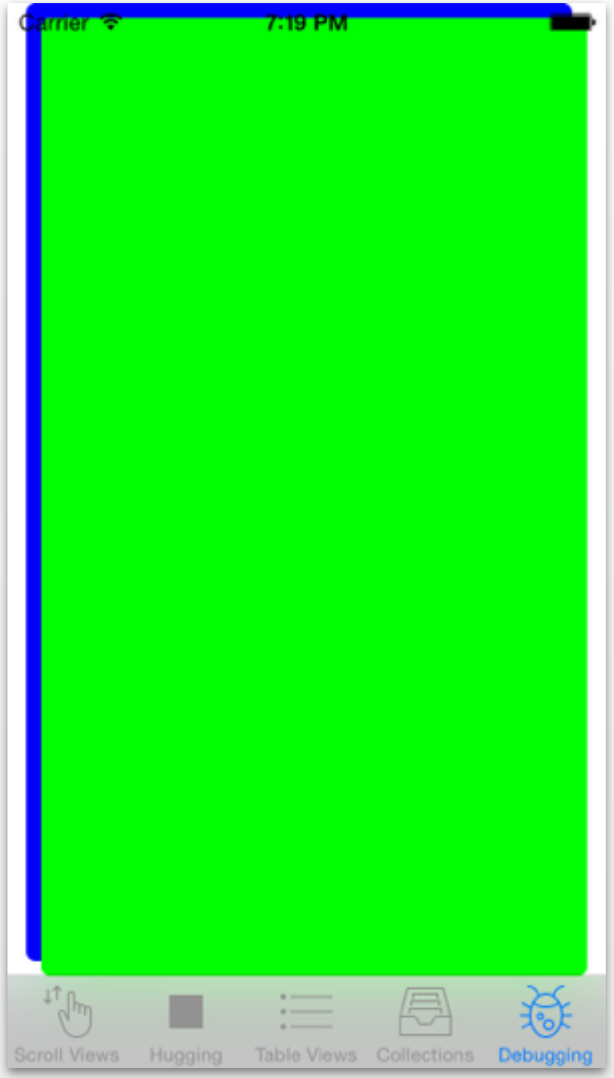
- Walkthrough
- Start with broken view
- Ambiguity First
- Conflicts Second



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

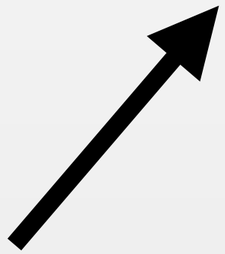
```
@implementation UIView (AmbiguityHelpers)

- (void)exerciseAmbiguityInLayoutRepeatedly
{
#ifdef 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]
```

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

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

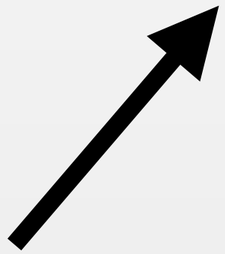
Enhancing Error Messages

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

Identify Constraints

- Use Constraint Identifiers

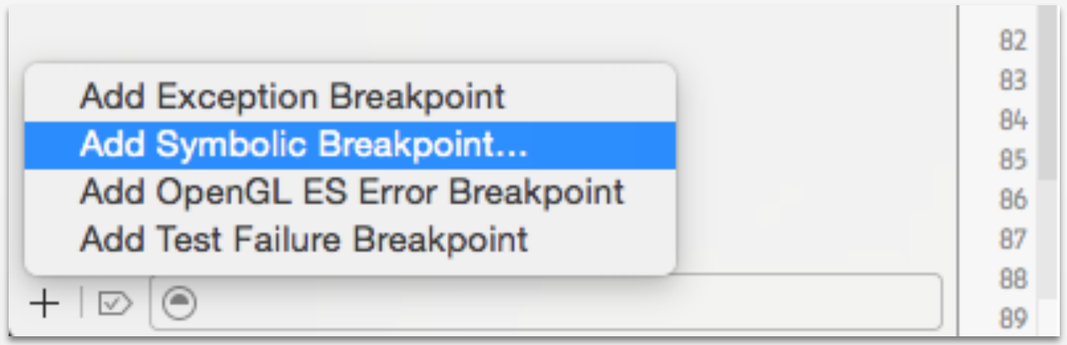
```
constraint.identifier = "Some Great Name"
```

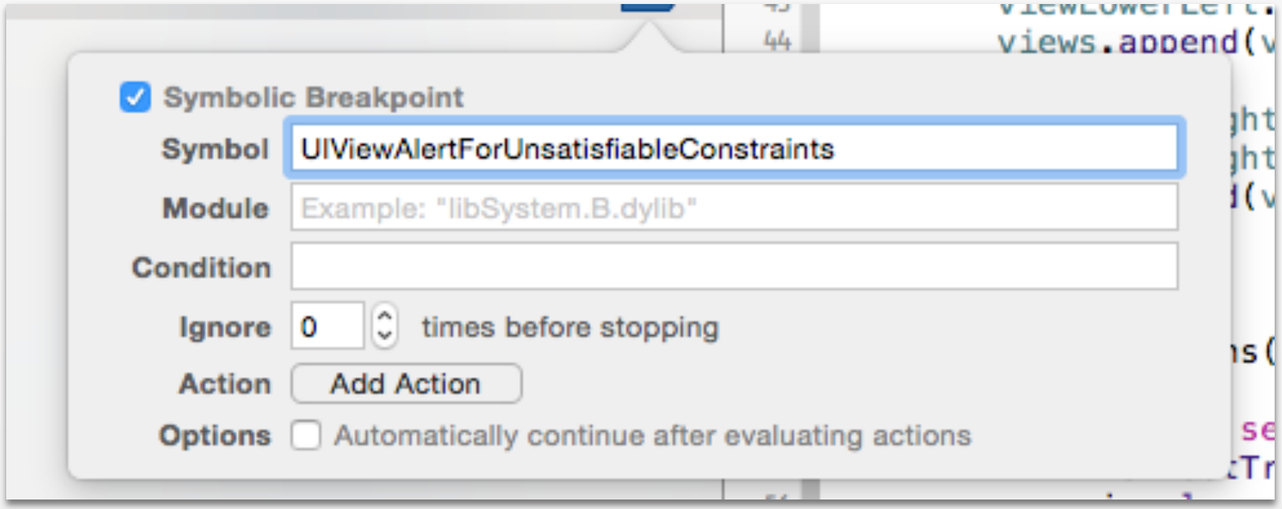


This will now show up in the debug views

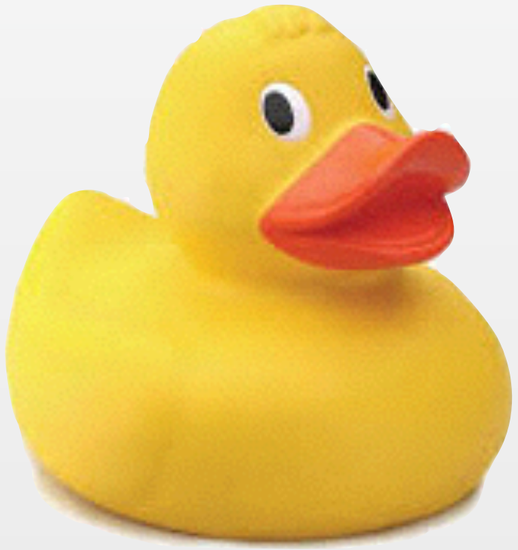
Breakpoints

- UIViewAlertForUnsatisfiableConstraints

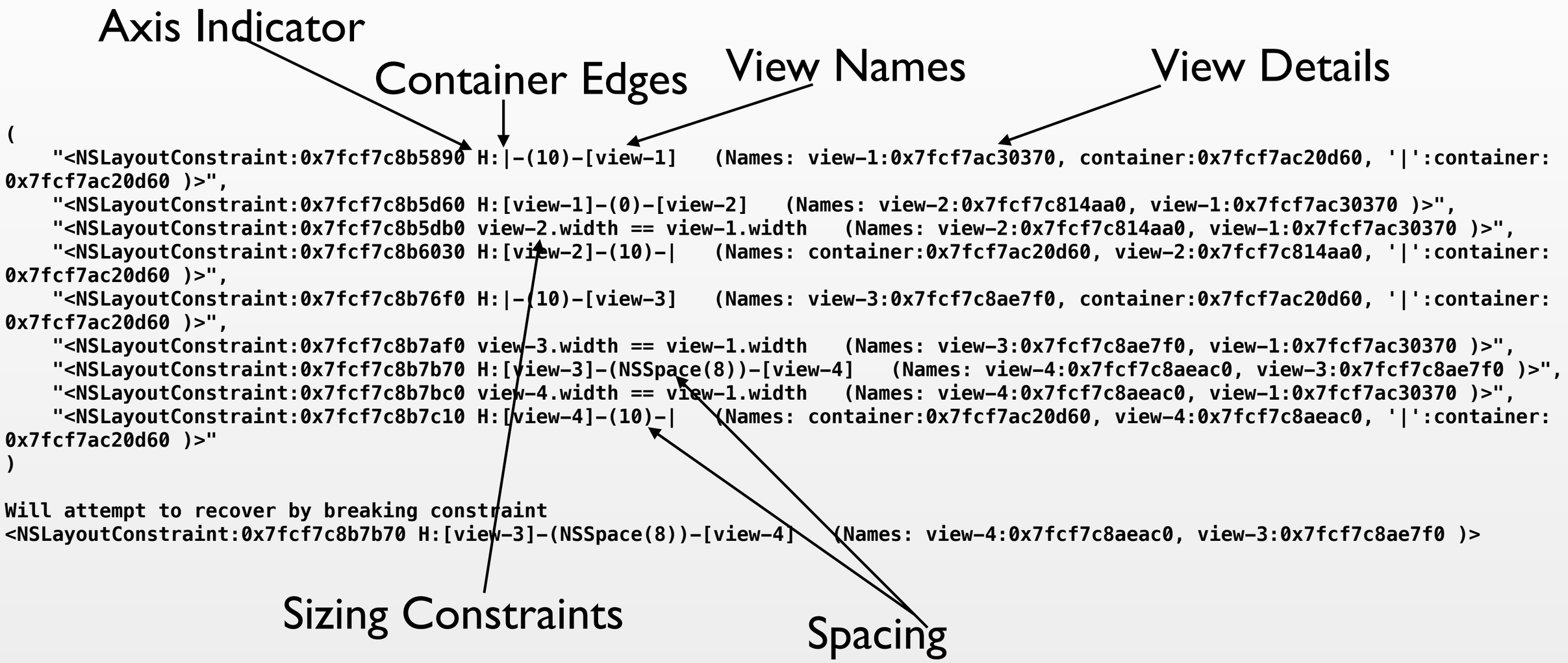




Examining Relations



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 )>
```

Algebra

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)-|
```

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

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

$view\text{-}1_w == view\text{-}2_w$
$view\text{-}1_w == view\text{-}3_w$
$view\text{-}1_w == view\text{-}4_w$

Code

System Generated Constraints

UIView-Encapsulated-Layout-Width

UIView-Encapsulated-Layout-Height

Code

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