GENERAL ASSEMBLY

# MOBILE DEVELOPMENT
## DRAWING IN CODE: PART 2

Rudd Taylor
Founder, SALT

# LEARNING OBJECTIVES

- Devise layouts relative to their superviews with springs and struts
- Design layouts with autolayout both programmatically and through interface builder
- Use NSLayoutConstraints to set our constraints in code
- Debug autolayout errors and warnings
- Differentiate between autolayout and Springs & Struts

# REVIEWING SPRINGS & STRUTS

# WHY DRAW IN CODE?

- Views have frames associated with them, always
  - Their position within their superview
- In springs and struts, we set the frame directly on the view being added (the ***strut***)
- We also define how the view moves when its superview changes (the ***spring***)
  - aka ***autoresizing masks***

# SPRINGS & STRUTS CODE-ALONG

# YOUR ASSIGNMENT

- Create a ‘face’ with springs and struts
- The face must contain
  - Two eyes
  - A nose
  - A mouth
- They must stay in position when the device rotates

# AUTOLAYOUT

# WHAT'S WRONG WITH S&S?

# WHAT'S WRONG WITH S&S?

- Springs & struts allows us to define how a view changes when its superview changes
  - e.g. "If my superview gets wider, I'll get wider too"
- Springs & struts does not allow us to define relations between subviews
  - e.g. "If the image view next to me moves to the left, I want to move to the left too"
- This makes some common tasks ***painful***
  - "I want a label to always be below an image" when the image moves or changes size
  - "I always want my image to be below some large block of variable text"
  - and more

# ENTER AUTOLAYOUT

- Another way to lay out views
  - As opposed to Springs & Struts
- The ‘new’ way to do things
  - Recommended, but not required
- ***Wildly*** more complex than Springs & Struts

# CONSTRAINTS

- We work with ***constraints*** in autolayout
- Constraints have:
  - ‘From’ and ‘To’ views
    - Each with an attribute
  - A relation
  - A multiplier
  - A constant

# CONSTRAINTS

- ‘From’ view: someImage
  - Attribute: Right
- ‘To’ view: someLabel
  - Attribute: Left
- Relation: Equals
- Multiplier: 1.0
- Constant: 0

- Translation: someImage’s right edge should equal someLabel’s left edge

# CONSTRAINTS

- ‘From’ view: someImage
  - Attribute: Width
- ‘To’ view: someLabel
  - Attribute: Width
- Relation: Equals
- Multiplier: 0.5
- Constant: 0

- Translation: someImage’s width should equal half someLabel’s width

# CONSTRAINTS

- ‘From’ view: someImage
  - Attribute: Top
- ‘To’ view: someLabel
  - Attribute: Top
- Relation: Equals
- Multiplier: 0.5
- Constant: 10

- Translation: someImage’s top should equal 10 pixels below someLabel’s top

# ADDING A CONSTRAINT

```
var subview = UIView()
subview.setTranslatesAutoresizingMaskIntoConstraints(false)
superview.addSubview(subview) // happens before constraints
superview.addConstraint(NSLayoutConstraint(item: subview,
                                attribute: .CenterX,
                                relatedBy: .Equal,
                                toItem: superview,
                                attribute: .CenterX,
                                multiplier: 1,
                                constant: 0)
```

# CONSTRAINTS

- Views likely have multiple constraints
- From those constraints we must be able to figure out origin and size

# AUTOLAYOUT CODEALONG

# GROUP ASSIGNMENT

- Recreate our ‘face’ using autolayout
- Bonus: When the user taps the screen, animate the members of the face changing color
- Bonus: When the user taps the screen, animate the mouth moving in some way

# ANIMATING AUTOLAYOUT

# ANIMATIONS

```
someConstraint.constant = 100 // someConstraint is a
constraint within self.view

UIView.animateWithDuration(5, animations: {

   self.view.layoutIfNeeded() // This animates the above
   change

})
```

# SCROLL VIEWS

# SCROLL VIEWS

- Set constraints on the scrollview first using constraints outside the scrollView
- Add subviews to the scroll view
- Tie something to all four sides of the scrollView
    - But don’t rely on it for its size