GENERAL ASSEMBLY

MOBILE DEVELOPMENT
DRAWING IN CODE: PART 1

Rudd Taylor
Founder, SALT

LEARNING OBJECTIVES

- Create views programmatically through springs/struts
- Recognize view hierarchy and how views are constructed in our applications
- Distinguish very clearly the differences between frame and bounds
- Implement scroll views and understand the properties needed to construct them
- Use CGGeometry framework to place and size our views

springs/struts, versus interface view controller, versus straight through code. Try to stick with 1 for remainder of class.

REVIEWING VIEWS

WHY DRAW IN CODE?

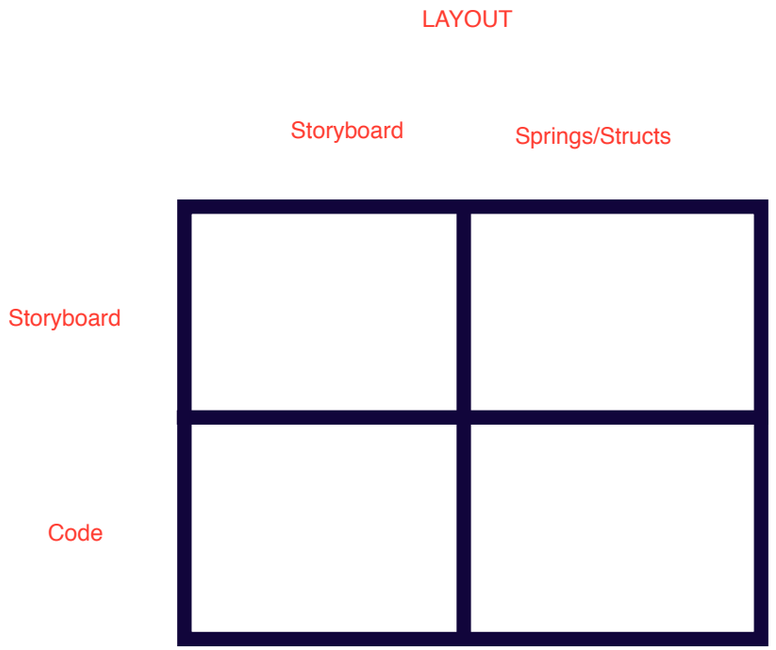
- Adding many subviews in interface builder can be unwieldy
- Managing storyboards and nibs when collaborating with others is hard
- Sometimes we want to create or destroy views when an action happens
- Some people prefer laying out in code to keep all view logic in one place

Example, we want certain view to appear upon Login (Arrival Layer) for only 5 seconds, then disappear.

DRAWING IN CODE

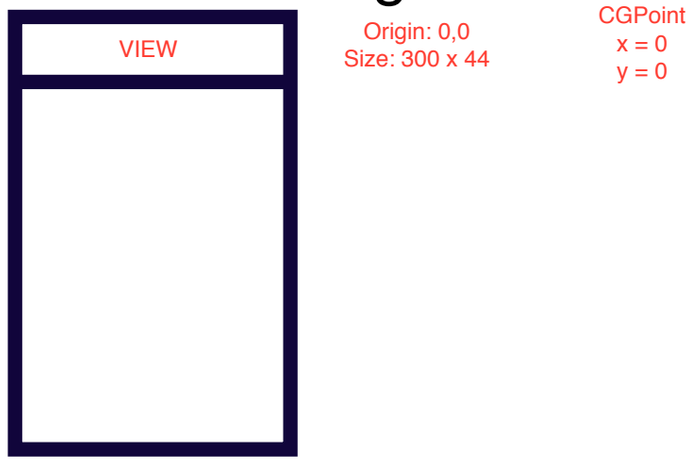
LOTS OF OPTIONS

- Two ways to manage view hierarchies
  - Storyboard
  - Code
- Two ways to lay out views
  - Springs & struts (the older way, today)
  - Autolayout (the newer way, next class)
- These can be mixed and matched, to a point



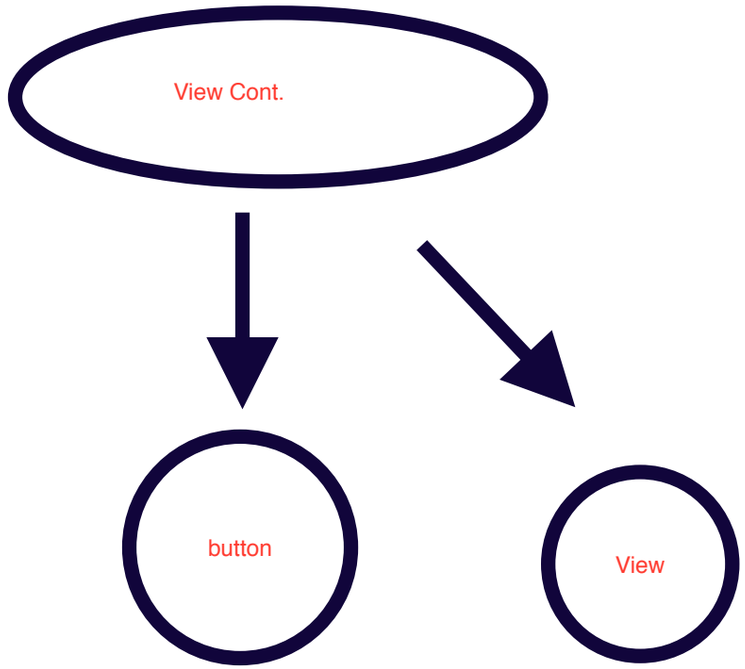
SOME NEW TYPES

- CGFloat: Just a float (a possibly non-whole number)
- CGPoint: A struct that contains an x and y coordinate (both CGFloats)
- CGSize: A struct that contains width and height (both CGFloats)
- CGRect: A rectangle that contains size and an origin
  - A suite of CGRectGet… functions help us do math on these rectangles



THE PROPERTIES OF A VIEW

- superview: A UIView that contains it. Only one of these
- subviews: The UIViews it contains. Many of these are possible
- frame: The position and size of the view within its superview (the *external* coordinate system)
  - Has origin x, y coordinate
  - Has size (width/height)
  - Usually what we deal with
- bounds: The view's *internal* coordinates system
  - Usually, just the frame but with (0, 0) as the origin
- center: The center point



THE PROPERTIES OF A VIEW (CON'T)

- alpha: A float representing transparency (0 is hidden, 1 is visible)
- backgroundColor: The background color of your view
- Things like corner radius and shadow are hidden in a view's ***layer***
  - e.g. view.layer.cornerRadius = 5
- Fun fact: All of these things can be animated using *UIView.animateWithDuration*

POINTS, NOT PIXELS

- Everything we do in iOS is using ***points, not pixels***
  - Virtual pixel which may actually be rendered by multiple physical pixels
  - http://www.paintcodeapp.com/news/ultimate-guide-to-iphone-resolutions

SPRINGS AND STRUTS

SPRINGS AND STRUTS

- The old way of laying things out on screen
  - Still around
  - Much simpler, conceptually
- Things are displayed using a combination of ***frame*** and ***autoresizing masks***

origin, width, height

what happens when superview size changes

  - Frame: Where the view is in its superview
  - Autoresizing masks: What the view does when its superview changes size
    - Default: Nothing
    - Can fix top, left, bottom, right margins
    - Can adjust width/height according to superview

SPRINGS AND STRUTS

```
let frame = CGFrame(x: 0, y: 0, width: 10, height: 10)

var view = UIView(frame: frame) // Create the view

someOtherView.addSubview(view) // Adds to another view

view.removeFromSuperview() // Removes the view from its
superview
```

SPRINGS AND STRUTS

CODEALONG

GROUP ASSIGNMENT

- Create a new project ***without storyboards or nibs***
- Create a login window programmatically
- Programmatically create elements needed for login (username, password, login button, and label for the title at the top.
- When login button is pressed, print a message with println.
- Bonus 1: Create a view that will be the container of the above elements
- Bonus 2: Make the container the size of its superview
- Bonus 3: When user taps ‘log in’, add another UIView with a success message and a dismiss button, which removes the login container

SCROLL VIEWS

SCROLL VIEWS

- Phones are small
- What if we have more content than can appear on the screen?
- UIScrollView is what helps us in this situation

SCROLL VIEWS

- UIScrollViews are just UIViews that contain other
- They have content which extends beyond their frame
  - contentSize CGSize - size of the content itself, that you're scrolling through. Maps, Websites, pieces of large content.
  - This ***must*** be set

SCROLL VIEW CODE-ALONG

VIEW CONTROLLERS IN CODE

```
let vc = MyCoolViewController()

// Configure your VC here

self.navigationController.pushViewController(vc, animated:
true) // Push

self.presentViewController(vc, animated: true) // Modal
```

GROUP ASSIGNMENT

- Continue from your previous project
- Once the user is logged in, present a confirmation modal view controller
- This view controller should display three images, of your choosing, that are stacked vertically
- The user should be able to scroll through the images