More iOS UI Considerations

Lecture 06: Attributed Strings and Auto Layout

Jonathan R. Engelsma, Ph.D.

TOPICS

- Attributed Strings
- Autorotation
- Auto Layout

ATTRIBUTED STRINGS

- Attributed String: text with multiple style runs with different font, size, color and other text features in different parts of the text.
- NSAttributedString / NSMutableAttributedStrings were promoted in iOS6 and are now much more easier to work with!
- e.g. integrated with UI controls that deal with text: UILabel, UITextView.

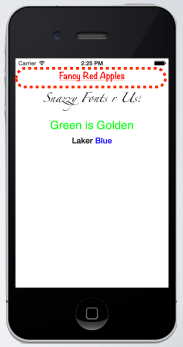
https://developerapple.com/library/ios/documentation/Cocoa/Conceptual/AttributedStrings/AttributedStrings.html#//apple_ref/doc/uid/10000036-BBCCGDBG

ATTRIBUTED STRING FEATURES

- Attributes can be associated with single characters, a range of characters or the entire string.
- Preservation of attribute-to-character mappings after changes.
- Support for RTF, including file attachments and graphics.
- Linguistic unit (word) and line calculations

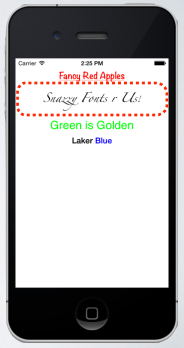
ATTRIBUTED STRINGS

```
if let font = UIFont(name: "Noteworthy-Bold", size: 18) {
    let attributes = [NSFontAttributeName : font,
        NSUnderlineStyleAttributeName : 1,
        NSForegroundColorAttributeName : UIColor.redColor()]

    let lab1 : NSAttributedString =
     NSAttributedString(string: "Fancy Red Apples",
         attributes: attributes)

    label1.attributedText = lab1
}
```



ATTRIBUTED STRINGS

```
if var font = UIFont(name:"Zapfino", size: 18) {
    let attributes = [NSFontAttributeName : font]
    let lab2 = NSAttributedString(
        string: "Snazzy Fonts r Us!",
        attributes: attributes)

    label2.attributedText = lab2
}
```



ATTRIBUTED STRINGS

```
if var font = UIFont(name:"Helvetica", size: 24) {
  let attributes = [NSFontAttributeName : font,
      NSForegroundColorAttributeName : UIColor.greenColor()]
  let lab3 = NSAttributedString(
      string: "Green is Golden",
      attributes: attributes)

  label3.attributedText = lab3
}
```

ATTRIBUTED STRINGS

```
var lakerString : String = "Laker Blue"
var font =
UIFont.preferredFontForTextStyle(UIFontTextStyleHeadline)
let attributes = [NSFontAttributeName : font]
let lab4 = NSMutableAttributedString(
    string: lakerString,
    attributes: attributes)
label4.attributedText = lab4

// change attrs on a substring.
let nsText = lakerString as NSString
let r = nsText.rangeOfString("Blue")
lab4.addAttribute(
    NSForegroundColorAttributeName, value:
UIColor.blueColor(),
    range: r)
label4.attributedText = lab4
```

AVAILABLE ATTRIBUTES

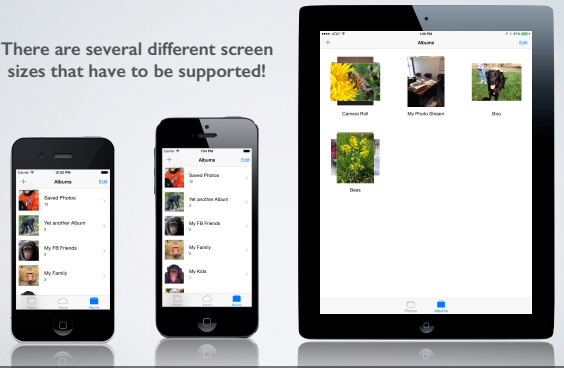
Font	Paragraph Style	Foreground Color
Background Color	Ligature	Kern
Strikethrough Style	Underline Style	Stroke Color
Stroke Width	Shadow	Text Effect
Attachment	Link	Baseline Offset
Underline Color	Strikethrough Color	Obliqueness
Expansion	Writing Direction	Vertical Glyph Form

https://developerapple.com/library/ios/documentation/UIKit/Reference/NSAttributedString_UIKit_Additions/Reference/Reference.html

ATTRIBUTED STRING DEMO

OTHER IOS UI CHALLENGES

There are several different screen sizes that have to be supported!

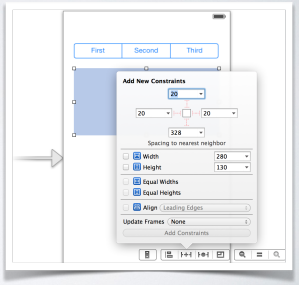


APPROACHES USED

- Handling different iPhone / iPod Touch screen sizes:
  - Normally done via Auto Layout and legacy springs-and-struts based autosizing approach.
- Supporting iPad:
  - Usually an entirely different layout in a separate storyboard.
  - Still can do significant amount of code reuse though!

AUTO LAYOUT

- Introduced in iOS 6.
- Replaces the legacy springs-and-struts based autosizing.
- builds relationships between views, specifying how views and their superviews related to each other.



AUTO LAYOUT

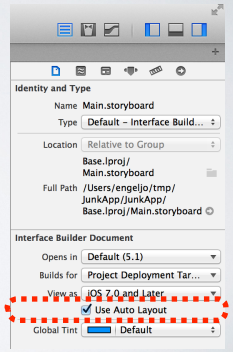
- Auto Layout is based on the Cassowary constraint-solving toolkit developed at the University of Washington.
- Solves systems of linear equalities / inequalities based on constraints.
- Constraints are rules that describe how one view's layout is limited with respect to another.
- Constraints can either be requirements or preferences.

AUTO LAYOUT

- Has the reputation for being hard to use. (An entire book has been written on the subject!).
- In reality, it is not so bad, and gives you significant flexibility in designing layouts over the legacy mechanisms.
- Can adopt incrementally. (e.g. not every view in our app has to utilize Auto Layout - can introduce controller at a time!)

TURNING ON AUTO LAYOUT

- Select any view controller in Interface Builder.
- Open the file inspector (⌘ - 1).
- Under the Interface Builder Document section, make sure "Use AutoLayout" is checked.



USING AUTO LAYOUT

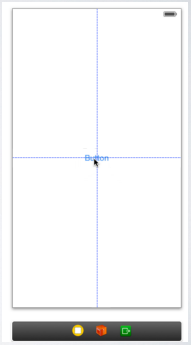
- Constraints can be expressed programmatically in Objective-C / Swift code.
- ✓ Constraints can be established "visually" using Interface Builder.
  - Detects/suggests missing constraints.
  - Warns and flags errors.

CONSTRAINT EXAMPLES

- Match one view's size to another view's size so they are always the same width.
- Center a view (or group of views) in a superview, not matter how the superview resizes.
- Align the bottoms of several views.
- Tie the bottom of one view to the top of another, so if one moves the other will.
- Prevent an image view from shrinking beyond a certain size.

SOME RULES OF THUMB

- In IB, always use the "snapping" guides when doing layouts.
- Use "default" or standard spacings - beware of rules with "magic" numbers in them.
- Always fix layout warnings/ errors in IB before attempting to run.
- Keep an eye on the debug log!



AUTO LAYOUT DEMO

AUTOROTATION

- If the user changes the device orientation (e.g. portrait to landscape or vice versa) the app might want to adjust the screen layout:

AUTOROTATION

- When the device is rotated, the top level view will have its bounds reoriented if:
  - view controller returns YES from method shouldAutoRotate
  - app allows rotation to that orientation (see general tab on app target in xCode)
  - view controller returns the new orientation in the method supportedInterfaceOrientations.

USE OF ROTATION

- **Compensatory** - adjust the display to accommodate how the user is holding the device
- **Forced Rotation** - force the orientation because the interface has been designed specifically for a given orientation.

AUTOROTATION

- Try to support different device orientations, when they make sense!
  - Supporting landscape is particularly useful when the soft keypad is displayed and user is entering text!
- Autolayout constraints may help you support the 90 degree different orientations, but not always!

READING ASSIGNMENT

- Chapter 1: Programming iOS 8 (by Neuburg) (See Auto Layout material in this chapter)

AUTOROTATION DEMO