GENERAL ASSEMBLY

MOBILE DEVELOPMENT

ARRAYS, TABLE VIEWS, DELEGATION

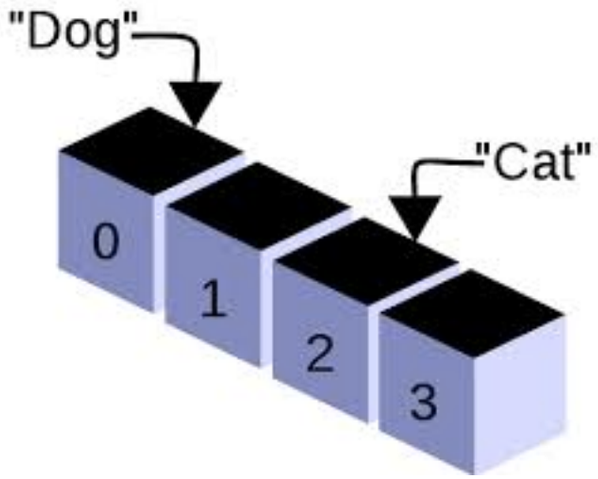
Tedi Konda
Executive Director, Technology, Unison

Learning Objectives

- Identify arrays in Swift
- Explore table views and add data programmatically
- Identify iOS design patterns and how they are used in our apps
- Define delegation and implement delegates in our apps

ARRAYS

ARRAYS



- Arrays have a few interesting properties
  - They contain things (we'll call them **elements**)
    - Arrays can also be empty
  - Each element has an index
    - Indexes start at 0
  - The array has a **count** of elements
  - Arrays have order, and can be iterated over in order
  - Looking up an element by index is **fast**

ARRAYS SYNTAX

- Creating an array
  - var array = [1, 2, 3] // Type is inferred if the array is populated
  - var array: [Int] = [] // Must declare type if array is empty
  - let array = [1, 2, 3] // Array constants cannot be modified
- Accessing an array
  - for i in [1, 2, 3] { /* This loops three times. i is first 1, then 2, then 3. */ }
  - for (index, element) in enumerate([“hi”, “there”, “class!”]) /* Loops three times, index is 0, 1 then 2.

    Element is “hi”, “there” then “class!” */
  - let firstElement = array[1] // We can access elements by index using this syntax

ARRAYS DEMO

PAIR EXERCISE: TO-DO LIST

- Create a new empty project
- In the provided view controller class, create a new array at the top (named toDoList) with 5+ to-do items.
- On viewDidLoad do the following:
  - Append a new item to the array
  - Iterate through each of the array items and println each array item
- Bonus: add a text field and button that appends the entered text to the array upon button being pressed

TABLE VIEWS

TABLE VIEWS

- Table views are a one dimensional list
  - Vocabulary:
    - Section: All table views contain multiple sections
    - Row: Every section has a number of rows, which are entries in that section
    - Index path: The combination of a section and row that is a unique entry in a table view
    - Cell: The view that is displayed for an index path (the class UITableViewCell is a subclass of UIView)
- Table views must have a number of sections, a number of cells in each section, and (optionally), the cells themselves
- Table views have a data source and a delegate
  - Data source: Provides cells, number of cells and sections
  - Delegate: Gets called when things happen to the table view, provides some views (e.g. header and footer)

TABLE VIEWS DEMO

PAIR EXERCISE: TO-DO LIST

- Display the data we added to the array in the previous exercise in the table view.
- Bonus: add the ability to delete a table cell on swipe.

WHAT IS A DESIGN PATTERN

- A design pattern is a reusable pattern to solve common issues that come up in software development
  - **NOT** new syntax
- An attempt to look at common issues that pop up
- A pretty generic definition (because ‘design pattern’ is a pretty generic term)
- iOS has several such patterns

REMEMBER PROTOCOLS?

- Like a superclass, but doesn't specify behavior
  - Only methods signatures (just the 'func' line) and variables
  - **NO** implementation of any methods
- If a class **meets** a protocol, it has all of the methods and variables the protocol specifies
- Used when a class needs to know what methods something has
- Protocols can be used as types, just like classes and structs
- When we have a variable that has a protocol type, we can use all the variables / methods that the protocol specifies (just like a class or struct)
- Classes can meet as many protocols as they like

THE DELEGATE

- The delegate is a relationship between two classes instances. One instance has a delegate variable which refers to an instance that has certain methods (**meets a protocol**). The is the original class's **trusted friend**
  - E.g. UITableView has var delegate: UITableViewDelegate?
- Instances tell their delegates information about when things happen to them
  - Or they get critical information from them
  - Many of Apple's classes do, e.g. UITableView, UITextField, UINavigationController
- A class has a delegate when it wants to delegate some behavior to another class
  - E.g. UITextField's delegate gets called when a text field text changes, the user presses return, etc
- Classes may have **one** delegate

DELEGATE CODE-ALONG

PAIR EXERCISE: TO-DO LIST

- Embed the table view controller in a navigation controller.
- Create another scene that will be used to add a to-do item to our table view.
- The new scene will include a text field and a button.
- When button is pressed, the user is taken to our table view, and the newly added item is added to the table view.
- Bonus: add the item to the top of the table view.