GENERAL ASSEMBLY

MOBILE DEVELOPMENT
ARRAYS AND DICTIONARIES

Rudd Taylor
Founder, SALT

LEARNING OBJECTIVES

- Recognize the need for arrays and create/update arrays
- Identify array methods and properties
- Discover dictionaries
- Identify dictionary methods and properties
- Compare and contrast arrays and dictionaries
- Define closures, use them with maps and filters

ASSIGNMENT AND CODE REVIEW

WHAT IS AN ARRAY?

- An array is a type (specifically, a struct) in Swift
  - Since arrays are structs, they are passed by ***value***
- It contains a one-dimensional list of things (instances of classes or structs)
- It can be added to and removed from (if not a constant)
- Just like everything else in Swift, it is typed
  - e.g. var strings: [String] = []

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***

ARRAY 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

CLOSURES

- We've been passing variables and constants into our functions, and getting those things back out of those functions
- Those variables and constants were generally values of some type
- But what if they were functions themselves?
  - These are called closures, which are a kind of inline, possibly unnamed function definition
- Just like everything else in Swift, they have a type
  - What does this function return, and what params does it take?
- Syntax: { (*params)* -> *ReturnType* in /* code */ }
- Swift is pretty smart about figuring out syntax here, and in many cases you can leave out large chunks of closure param/return information. We won't cover this in class.

USING CLOSURES WITH ARRAYS

- Arrays have a few interesting functions that can be used to work with them
  - Map: Return a new array, of the same length, where each new element was created from the corresponding old element
    - e.g. You have an array of People, each of which have an Int age. Use map to get an array of ints representing people's age.
  - Filter: Return a new array with a subset of the original contents, based on some criteria
    - e.g. You have an array of integers, use filter to get the even ones
  - Reduce: Return one value that was created using each of the array's values
    - e.g. You have an array of integers, use reduce to get the sum of them

ARRAY DEMO & EXERCISE

ARRAY EXERCISE

- Create a new app with a navigation controller and a table view controller
- The table view contains a list of strings, which is initially set to a few test values
- The cells in the table view display the list of strings in the array, in order
- Then, add an “add” button as a navigation item. When clicked, it should display a modal dialog
- That modal dialog should ask the user for a new string. When the user clicks a confirmation button, add that string to the array and display it in the table view
- Bonus: Add a button called ‘sort’, which should sort the array alphabetically
- Bonus: Add another button to the navigation controller on the main page, when clicked, it should make the table view only display strings that start with capital letters. Use ***filter***
- Bonus: Implement deleting an item from the table view

DICTIONARIES

WHAT IS A DICTIONARY?

- A dictionary has a unique set of ***keys.*** Each of those keys is unique in the dictionary
- Each key has a ***value***, which can be quickly referenced if you have the ***key***
  - Values do not have to be unique in the dictionary
- Storage: ages[“rudd”] = 29
- Retrieval: if let ruddAge = ages[“rudd”] {/* if ages[“rudd”] exists, this is run */}
- Also referred to as ***maps***

WHAT IS A DICTIONARY?

- We use dictionaries when there is an association between one thing and another
- You ***really really should*** query a dictionary for a value when you already have the key
- Looking up values for keys in dictionaries is ***fast***

DICTIONARY SYNTAX

- Creating a dictionary with values: var ages = [“rudd”: 29] // Type is [String: Int]
- Creating an empty dictionary: var ages: [String: Int] = [:]
- Creating a constant: let ages = [“rudd”: 29]
- Accessing: let ruddAge = ages[“rudd”] // ruddAge is an Int? with value 29
  - Hint: This is a great chance to use ‘if let’!
- Setting: ages[“kevin”] = 21

DICTIONARY DEMO & EXERCISE

DICTIONARY EXERCISE

- Change your app to store an array of Dictionary<String, String>
  - E.g. [[“name”: “rudd", “dueDate”: “Thurs”, “status”: “done”], [“name”: “travis”]]
- Change your ‘add’ screen to add two more text boxes, one for due date and one for status. Both accept strings
- When a new item is added, a new dictionary should be added to the array
- Change your table view to display name, due date and status
- Bonus: Add a ‘sort’ button which sorts the table view by name
- Bonus: Add a ‘names’ button which prints, to the console, an array of the names in the table view. Use map()