

MOBILE DEVELOPMENT

SWIFT DATA STRUCTURES

William Martin
Head of Product, Floored

Angel X. Moreno
EIR, Developer

ARRAYS AND DICTIONARIES

LEARNING OBJECTIVES

- › Create, modify, and iterate over arrays.
- › Identify and use array methods and properties.
- › Create, modify, and iterate over dictionaries.
- › Identify and use dictionary methods and properties.
- › Compare and contrast arrays and dictionaries.
- › Deploy arrays and dictionaries in UITableViews to hold app data.

ARRAYS AND DICTIONARIES

REVIEW OF IBOUTLETS AND IBACTIONS

EXERCISE: DISMISS A MODAL

ARRAYS AND DICTIONARIES

DATA STRUCTURES

ARRAYS AND DICTIONARIES

WHAT IS A DATA STRUCTURE?

- A data structure is a type that manages potentially large amounts of data.
- What are data?

ARRAYS AND DICTIONARIES

WHAT ARE DATA?

- Data are pieces of information devoid of meaning.
- e.g. What do these numbers mean?
 - 83, 80, 82, 87, 86, 88, 78, 87, 88, 89, 85, 85, 88, 81, 83, 87, 90, 88, 95, 88, 96, 94, 93, 92, 89, 92, 86, 88, 97, 92, 89

ARRAYS AND DICTIONARIES

WHAT ARE DATA?

- Data are pieces of information devoid of meaning.
- e.g. What do these numbers mean?
 - 83, 80, 82, 87, 86, 88, 78, 87, 88, 89, 85, 85, 88, 81, 83, 87, 90, 88, 95, 88, 96, 94, 93, 92, 89, 92, 86, 88, 97, 92, 89
- Hint: We don't know.

ARRAYS AND DICTIONARIES

WHAT ARE DATA?

- Data are pieces of information devoid of meaning.
- e.g. What do these numbers mean?
 - 83, 80, 82, 87, 86, 88, 78, 87, 88, 89, 85, 85, 88, 81, 83, 87, 90, 88, 95, 88, 96, 94, 93, 92, 89, 92, 86, 88, 97, 92, 89
- Hint: We don't know.
- But when I tell you that they're the record high temperatures for NYC in May collected in Central Park, then they *mean* something. They're promoted to *information*.

ARRAYS AND DICTIONARIES

ARRAYS

ARRAYS AND DICTIONARIES

WHAT IS AN ARRAY?

- › An array is a type (implemented as 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” (i.e. has a specific type).
 - › e.g. `var strings: [String] = []`

ARRAYS AND DICTIONARIES

ARRAYS

- › Arrays have a few interesting properties.
 - › They contain objects. (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.
- › Arrays can change their order by reversing or sorting their elements.
- › Arrays elements can change by adding, removing, or filtering them.

ARRAYS AND DICTIONARIES

SYNTAX: ARRAYS

Creating an array:

// Type is inferred (implicit) if the array is populated with elements of the same type.

```
var arr = [1, 2, 3]
```

// Must declare type explicitly if array is empty, ...

```
var arr: [Int] = []
```

// ... or instantiate the type itself:

```
var arr = [Int]()
```

ARRAYS AND DICTIONARIES

SYNTAX: MANIPULATING ARRAYS

Adding elements to an array (adds an element to the end of the array):

```
arr.append(4)
```

Insert an element wherever you want, as long as its in the arrays “bounds”:

```
arr.insert(7, atIndex:0)
```

Concatenating two arrays (produces a new array with the elements from both in order):

```
arr + [4, 5, 6]
```

Array constants cannot be modified.

```
let arr = [1, 2, 3]
```

ARRAYS AND DICTIONARIES

SYNTAX: ARRAY ELEMENTS

Accessing elements in an array:

```
// We can access elements by index using this syntax
```

```
let firstElement = arr[0]
```

```
for el in [2, 4, 6] {
```

```
    // This loops three times. el is first 2, then 4, then 6.
```

```
}
```

```
for (index, element) in enumerate(["hi", "there", "class!"]) {
```

```
    // Loops three times...
```

```
    // index is 0, 1 then 2. element is "hi", "there" then "class!"
```

```
}
```

ARRAYS AND DICTIONARIES

SYNTAX: REMOVING AND COUNTING ELEMENTS

// Remove an element from an array like this:

```
var words = ["hello", "doctor", "name", "continue", "yesterday", "tomorrow"]
```

```
words.removeAtIndex(3)
```

```
words
```

// Count elements in an array by using the function “count”:

```
count(words)
```

ARRAYS AND DICTIONARIES

SORTING ARRAYS

Sorting elements in an array.

```
// Sort an array of basic types like this. Note the &.
sort(&arr)
```

```
// Sort arrays of more complex types by providing a “comparator”:
func sortByName(a:Dog, b:Dog) -> Bool {
    return a.name < b.name
}
dogs.sort(sortByName)
```

ARRAYS AND DICTIONARIES

FILTERING ARRAYS

Filter elements in an array.

```
// Filter arrays by providing a filter function:  
func isMale(dog:Dog) -> Bool {  
    return dog.gender == "m"  
}  
let maleDogs = dogs.filter(sortByName)  
maleDogs
```

ARRAYS AND DICTIONARIES

ARRAY DEMO & EXERCISE

ARRAYS AND DICTIONARIES

ARRAY EXERCISE

- › Create a class called Dog.
- › Create a class called DogCollection.
- › Give DogCollection the ability to do the following:
 - › Hold a collection of dogs (perhaps, an array property called ... “dogs”?)
 - › Add a single dog to the collection.
 - › Sort the dogs by name.
 - › Return the current number of dogs.
 - › Return a dog given its name. (Hint: What happens if no dog of that name lives in the collection?)
- › Create an instance myFamilyDogs and add some dogs to it.

ARRAYS AND DICTIONARIES

DICTIONARIES

ARRAYS AND DICTIONARIES

WHAT IS A DICTIONARY?

- A dictionary is a “mapping” from one set of unique things (like phone numbers) to another set of things (like people).
- It’s like an array, but instead of using indices (0, 1, 2, 3, ... count - 1), we are given the ability to define our own indices, called “keys.”

ARRAYS AND DICTIONARIES

WHAT IS A DICTIONARY?

- A dictionary has a set of unique **keys**. *i.e.* Each of those keys is only found once in the dictionary.
- Each key points to a **value**, which can be easily 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**

ARRAYS AND DICTIONARIES

SYNTAX: DICTIONARIES

Creating a Dictionary:

```
// Type is [String: Double]. String = type for the keys, Dog for the values.
```

```
// [:] means an “empty” Dictionary.
```

```
var constants : [String: Double] = [:]
```

```
// Can implicitly set a type for a Dictionary by giving it the first pair:
```

```
var constants = ["e": 2.71828]
```

```
// or by instantiating the empty type:
```

```
var constants = [String: Double]()
```

ARRAYS AND DICTIONARIES

SYNTAX: MANIPULATING DICTIONARIES

```
var constants = ["e": 2.71828]
```

```
// Add another pair:
```

```
constants["pi"] = 3.14159
```

```
// Look up a value in the dictionary.
```

```
constants["e"]
```

```
// But if that key doesn't exist, you get nil!
```

```
constants["c"] // OOPS!
```

ARRAYS AND DICTIONARIES

SYNTAX: DEALING WITH ABSENT KEYS

```
// Declare a dictionary with multiple key-value pairs from the start:
var constants = ["e": 2.71828, "pi": 3.14159]

if let c = constants["c"] {
    // Does this look familiar?
} else {

}
```

ARRAYS AND DICTIONARIES

MORE ABOUT DICTIONARIES

- We use dictionaries when there is an association between a set of unique identifying things and another set of things (not necessarily unique) to be identified.
 - SSN → person
 - phone number → person
 - registration number → dog
- You *really should* query a dictionary for a value first when you have a key. Don't assume it has a value or the key!

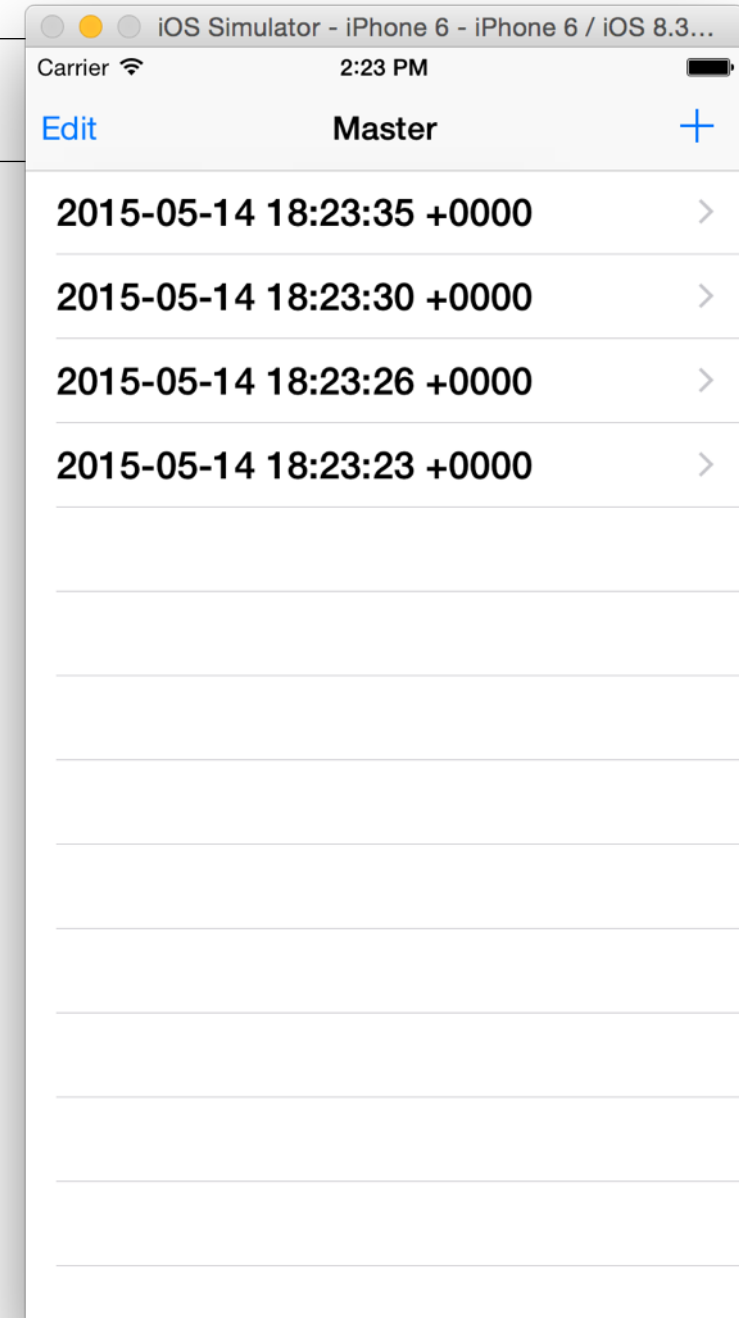
ARRAYS AND DICTIONARIES

TABLE VIEWS

ARRAYS AND DICTIONARIES

TABLE VIEWS

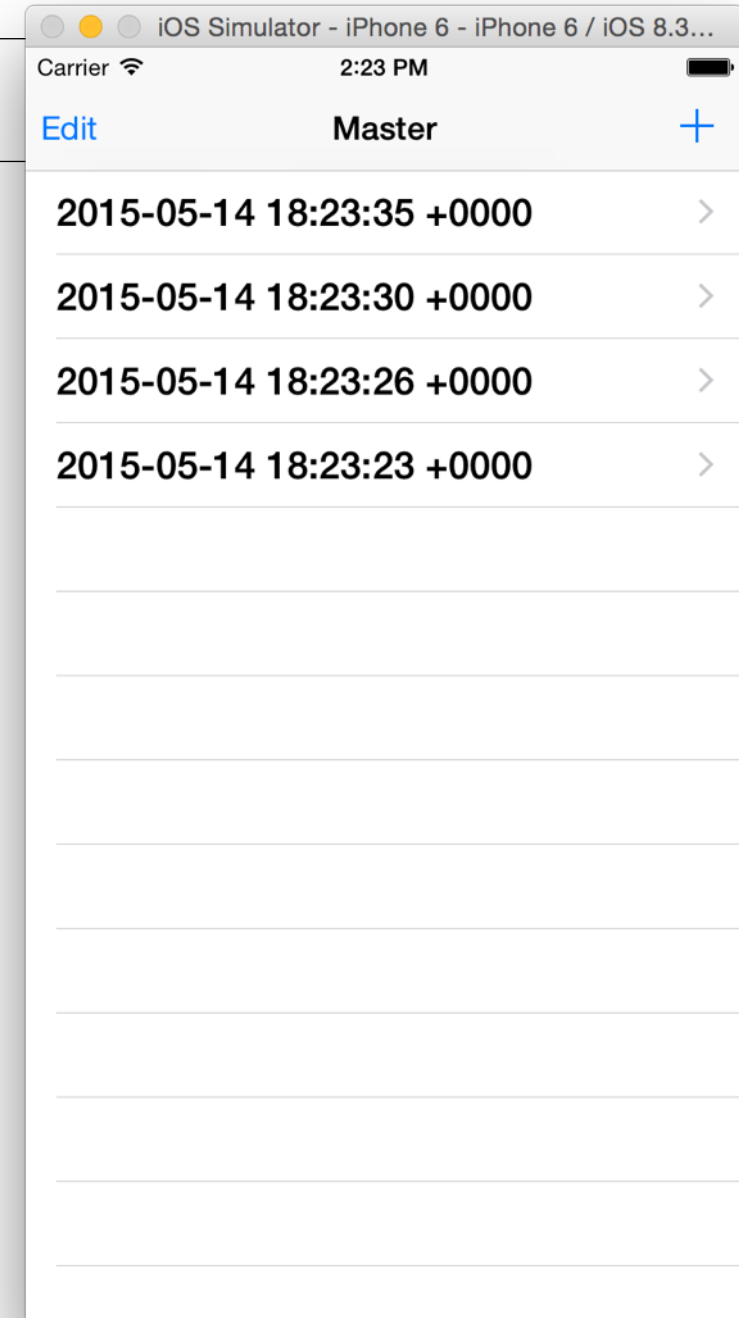
- Table views display a one-dimensional list of cells.
 - Section: All table views contain multiple sections.
 - Row: Every section has a number of rows, which are entries in that section.
 - NSIndexPath: The combination of a section and row that is a unique entry in a table view.
 - Cell: The view that is displayed for an NSIndexPath (class UITableViewCell, which is a subclass of UIView).



ARRAYS AND DICTIONARIES

TABLE VIEWS

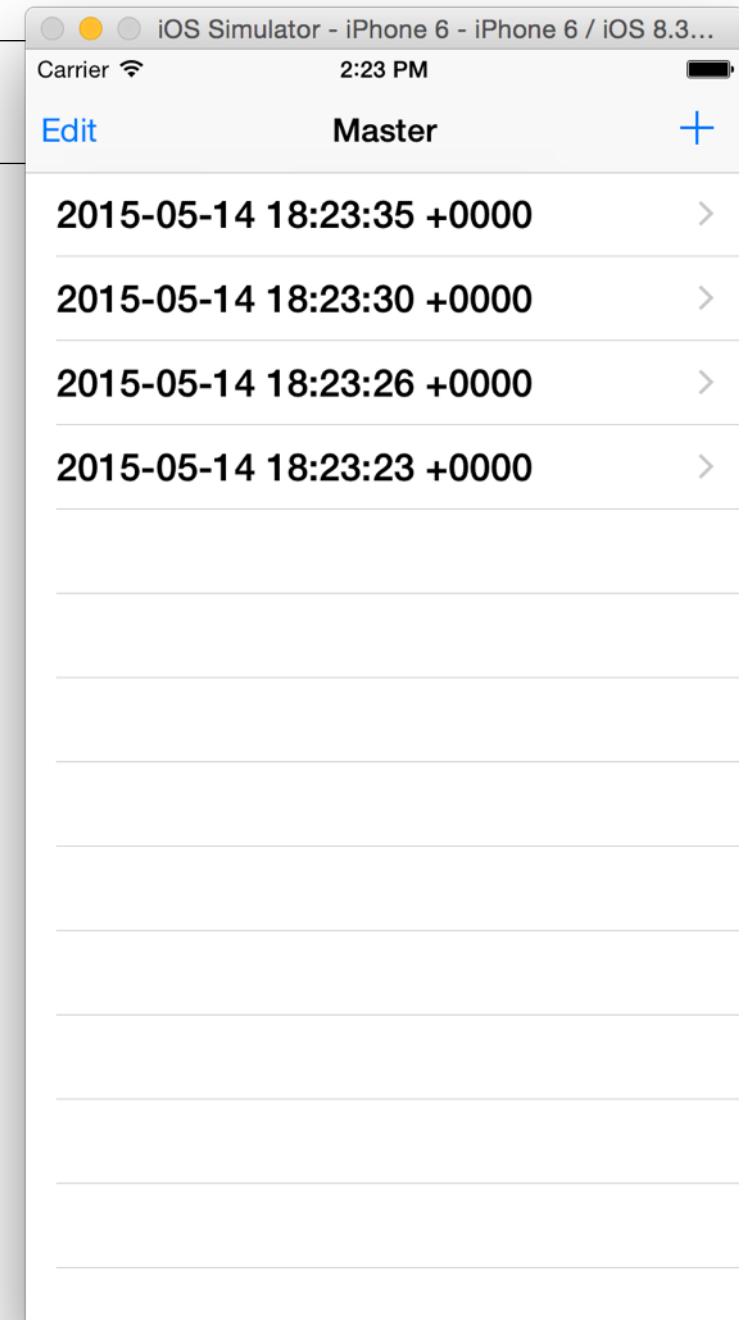
- Table View Controllers provide:
 - the number of sections (think the mini letter headings in the Contacts app),
 - the number of cells in each section, and
 - (optionally) the cells themselves.
- Table views also can have a data source and a delegate.
 - Data source: Provides cells, number of cells, and sections.
 - Delegate: Gets called when things happen to the table view, provides some views (e.g. header and footer).



ARRAYS AND DICTIONARIES

TABLE VIEWS

- › How do we achieve this?
 - › Extend a UIViewController.
 - › Override methods that return these values (e.g. number of sections, etc.)
- › Recommend using the Master-Detail template to start learning with a new app.



ARRAYS AND DICTIONARIES

PUTTING IT ALL TOGETHER

ARRAYS AND DICTIONARIES

DATA STRUCTURES EXERCISE

- › Create a new app from the Master-Detail Application template. *Wait to get a tour.*
- › The Table View should display the list of dogs' names.
- › Use the Dog and DogCollection classes you've already created. Put them in a separate Swift file.
- › Then, place an "Add" button as a "Bar Button Item" in the upper right. When tapped, it should display a "Add another dog" form via a modal Segue.
- › That modal dialog should ask the user for a dog's name (Bonus: And other info, too!). When the user taps an "Add" button, create a new Dog, add it to the collection, then dismiss the modal. The new dog should be in the table.
- › Bonus: Add a button called 'sort', which should sort the dogs alphabetically by name.
- › Bonus: Add two buttons to the navigation controller on the main page, when tapped, it should make the table view only display dogs that are male or female.
- › Bonus: Implement deleting an item from the table view.