iOS Dev Accelerator
Week 1 Day 1

- Introductions & Info
- Intro to Swift
- MVC
- TDD

Instructors

- John Clem - john@codefellows.com
- Brad Johnson - brad@codefellows.com

Class Format

- From 9 to 12 everyday we will be here in the east room for lectures.
- From 1 to 4 we go upstairs and work on the homework/projects.
- Fridays are dedicated to job hunt related activities/workshops and guest speakers.
- Every week we will build a separate app that uses everything we learn in the lectures
- Week 4 is dedicated to building a solo app of your own, week 8 is for a group app.

Lectures

- 3-4 major concepts / topics each day.
- Start with a short lecture, then go into live coding examples.
- Slides will be posted immediately after each lecture
- While we are live coding, **please do not just mindlessly copy everything we are typing**.
- Instead, pay attention, ask questions, and take notes. All code will be posted to a class repository on Github for your convenience.

Afternoons

- Afternoons are for getting your homework done.
- The Instructors will be up there with you answering questions and helping you along.
- You will have a repository setup for each week's app, and then submit the link to your repository as your homework submission.
- Occasionally we will have quiz-like questions in the homework, submit those answers with your link.

Why we like Swift

- concise
- safe
- fast
- modern

Obj-C & Swift Similarities

- Same runtime
- “Objective C without the C”
- LLVM compiler & debugger
- Still Cocoa

Variables & Constants
or
var and let

- var for variable variables (value can be set to a different value)
- let for constant variables (value cannot be set to different value)
- let isn't just for constants
- similar to mutable and immutable BUT

Primitive Types

Int

- Int8,Int6,Int32,Int64 bit
- unsigned can’t be negative
- You usually don’t choose, just use Int!

Float & Double

- Use Float for 32-bit, Double for 64-bit
- Double is the default
- Must alway have number on both sides of the decimal point

Number Conversions

- Fundamentally different from ObjC's conversion system
- Must cast with Type() for any form of mixing different number types
- Safer and faster

String

- var declares mutable String, let declares immutable
- No longer a reference type, copied when passed
- String interpolation

```
var name = “johnny”
name += “clem”
var githubName = “@“ + name
var gmailName = name + “@gmail.com”
```

String

- Strings are just unicode character arrays
- Iterating over a String

```
var name = “johnny”
for character in name {
  // do stuff
}
```

Collection Types

Arrays

- Always clear about what type of values it will store
- isEmpty property to check if count is 0
- append() or += to add items to end of array

Dictionaries

- Always clear about type of values AND keys
- Dictionary<KeyType, ValueType>
- .keys and .values properties (for loop)
- can switch on keys now as well

Functions

- self-contained chunks of code
- functions have names that are used to call the function
- parameters are separated with commas, and are written as name : type
- return type denoted with ->

Functions

- Parameters and return values are not required
- Tuples allow functions to return multiple values

Methods

- Methods are functions that are associated with a type (class,type,enums), so still use the func keyword
- Methods are called just like functions with one difference: parameter names in methods are also used when calling the method (except for the first one!)

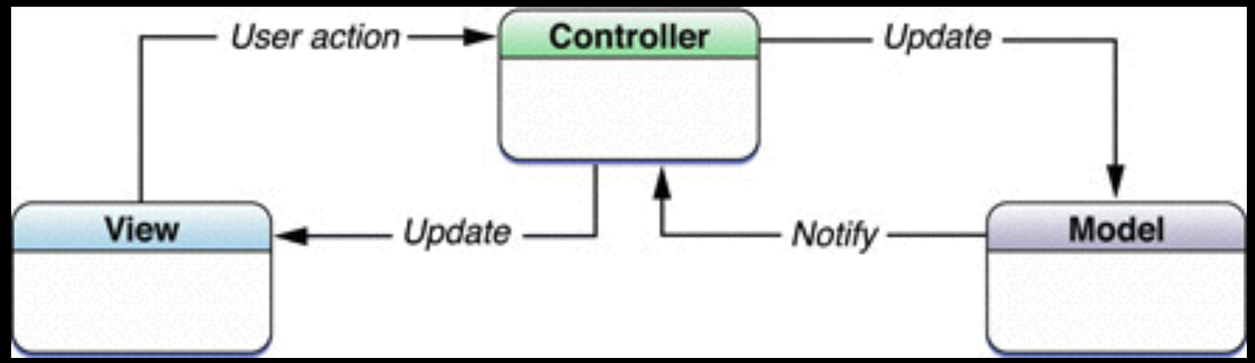
Optionals

- Use optionals in situations where a value may be absent.
- Swift does not allow you to leave properties in an undetermined state. They must either be given a default value, a value set in the initializer, or marked as optional.

MVC (Model–View–Controller)

- A design pattern commonly used in the development of Cocoa Apps and also championed by Apple.
- Assigns objects in an application one of three MVC roles: Model, View, Controller.
- The separate layers are separated by abstract boundaries.

MVC or MCV LOL?



Some people joke its more like MCV, because the controller is the middle man so the C should go in the middle

Classic programming joke

Model Layer

- Model objects encapsulate data and logic that are going to be used by your application
- The Twitter App has a Tweet model class, a User model class, a Favorite model class, etc (probably)

View Layer

- A View object is an object the user can see and possibly interact with.
- Enables displaying and editing of the app's data.
- Communication between the View and Model layers is made possible by…..

Controller Layer

- Act as the intermediary between the model layer and view layer.
- The most common form of a controller in iOS is a view controller.
- Another common controller is a network controller.
- At first your view controllers will have a lot of code. Eventually you should strive to make them lighter so its easier to understand what they are doing.

TDD – Test Driven Development

- Write tests during every phase of your app's development.
- Test small chunks and micro features of your app, not large scale or abstract features or problems.
- Write a failing test first, then write the code to get that test to pass.
- Testing is a trade off: you trade the cost of the time it takes to write the tests for a better understanding of your code and automatic bug finding.

Unit Tests

- Unit tests are small pieces of code that test the behavior of other pieces of code.
- They setup the necessary preconditions,run the code under test, and then test an assertion about the final state after the code has run.
- A passing test will turn into a failing test if the underlying code being tested is changed and the test no longer passes. This is referred to as a detected regression and is an awesome way of finding bugs and pretty much the whole point of this!