GENERAL ASSEMBLY

MOBILE DEVELOPMENT

Tedi Konda
VP, Engineering and Technology, RepEquity

MEET YOUR INSTRUCTOR

Instructor

Tedi Konda

YOU

1. WHAT'S YOUR PREVIOUS EXPERIENCE WITH MOBILE AND OTHER PROGRAMMING IN GENERAL?
2. WHY ARE YOU TAKING THIS COURSE?
3. WHAT'S YOUR FAVORITE APP?

GETTING STARTED

ONE RULE:
INTERRUPT ME!

Seriously.

Don’t ever be ashamed or afraid of asking us questions.

There are NO stupid questions.

LEARNING OBJECTIVES

LEARNING OBJECTIVES: LESSON

- Course Expectations
- Nomenclature (iOS, Swift, etc.)
- Overview of Developer Tools
- Overview of Supplemental Learning Resources
- Pre-Work Debrief and Github
- Jumping into Xcode
- Jump into Interface Builder

BY THE END OF THIS COURSE…

you will have created your own iOS app from scratch and submitted it to the App Store!

(Let that sink in for a moment)

HOW WILL I DO THAT?

- 72 hours of class (3hrs/class * 24 classes)
  - Lectures, Code-Alongs, Code-Reviews, Pair Programming
- 72 hours of homework (6 hrs/week * 12 weeks)
  - R&D, Inline Code Feedback
- 48 hours of office hours (4hrs/week * 12 weeks)
  - Each instructor will be available 2 hours a week.
  - 1-on-1 assistance outside of class

FOOD FOUR THOUGHT

There are 2016 hours in a 12 week time period.

At the very least, you will spend 192 hours,
or 9.5% of your time working with Swift and iOS.

Think about what else your could accomplish in your life if you devoted only 10% of your time.

COURSE EXPECTATIONS

EXPECTATIONS & SYLLABUS

- Four Units (3 weeks each)

1. Translate Wireframes into Functional App Interfaces
2. Experiment with Object Oriented Swift and add Logic to iOS Apps
3. Build Apps with persistent Data and Remote APIs
4. Submit to the App Store

EXPECTATIONS: UNIT 1

- Learn how to create bare-bones template projects
  - …and extend them!
- Learn how to create multiple views (e.g., screens) using Interface Builder
- Learn basics of Swift using Playgrounds
- Learn how to save and store your code using Git
  - …and Github

EXPECTATIONS: UNIT 2

- Learn your ideas through Pseudo-Code
- Turn your Pseudo-Code into Swift code
- Learn Object Oriented Programming
- Learn Object Oriented Paradigms
  - Design Patterns
  - Data Structures
- Learn how to add interactivity
  - …using gestures and animations
- Learn how to programmatically create views

EXPECTATIONS: UNIT 3

- Learn how to store data
  - Temporary local storage
  - Permanent local storage
  - Permanent remote storage
- Learn how to interact with internet platforms via APIs (e.g., Networking)
- Learn how to use Open Source repositories
  - …and how to avoid reinventing the wheel!
- Learn how to mix Objective-C code into your Swift projects
  - …without having to know an iota of Objective-C!

EXPECTATIONS: UNIT 4

- App Optimization
- Distributing your App
  - App IDs, Device IDs, Certificates, Provisioning Profiles
  - TestFlight
  - iTunes Connect
- App Store Best Practices

EXPECTATIONS: EXTRA

- We don’t cover everything in this class, however, we have a couple of classes set aside to teach you a topic you’re interested in.
- Think about what you may want to learn that’s not covered in the class.
- We’ll ask you in 6 weeks.

ASSESSMENTS

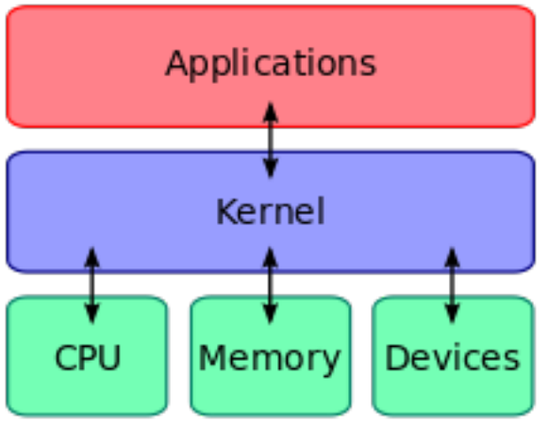
- Each week, you will be assessed
  - Homework assignment
  - Rubric
  - There will also be a Midterm and a Final
- We will grade your homework assignment on a 0-2 scale using the Rubric for that week
  - 0 = Doesn’t Pass (e.g., Needs improvement)
  - 1 = Pass (e.g., Code works, but may have a few bugs/issues)
  - 2 = Exceed (e.g., Everything works as expected)
- All assessments will be made available to you tonight
  - You’ll know what you’re graded on from day one.

NOMENCLATURE

NOMENCLATURE (PT. 1)

- OS (Operating System)
  - Software infrastructure (Kernel) that communicates between hardware (CPU, Memory, etc.) and software (Applications)

- iOS (“internet” or “i/me/my personal” Operating System)
  - Operating system that works exclusively on Apple’s hardware



NOMENCLATURE (PT. 2)

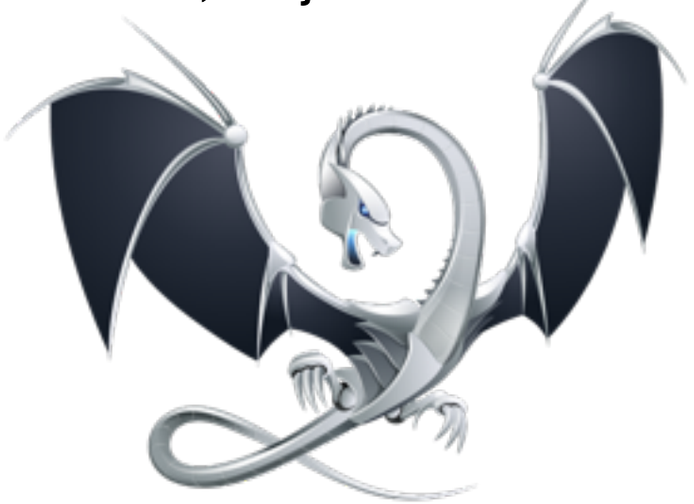
- Programming language
  - A way to communicate (semi-)human-readable instructions to a computer to perform a certain set of actions.

- Swift
  - A programming language built by Apple to write software that works with Apple's operating systems (iOS, OS X)

NOMENCLATURE (PT. 3)

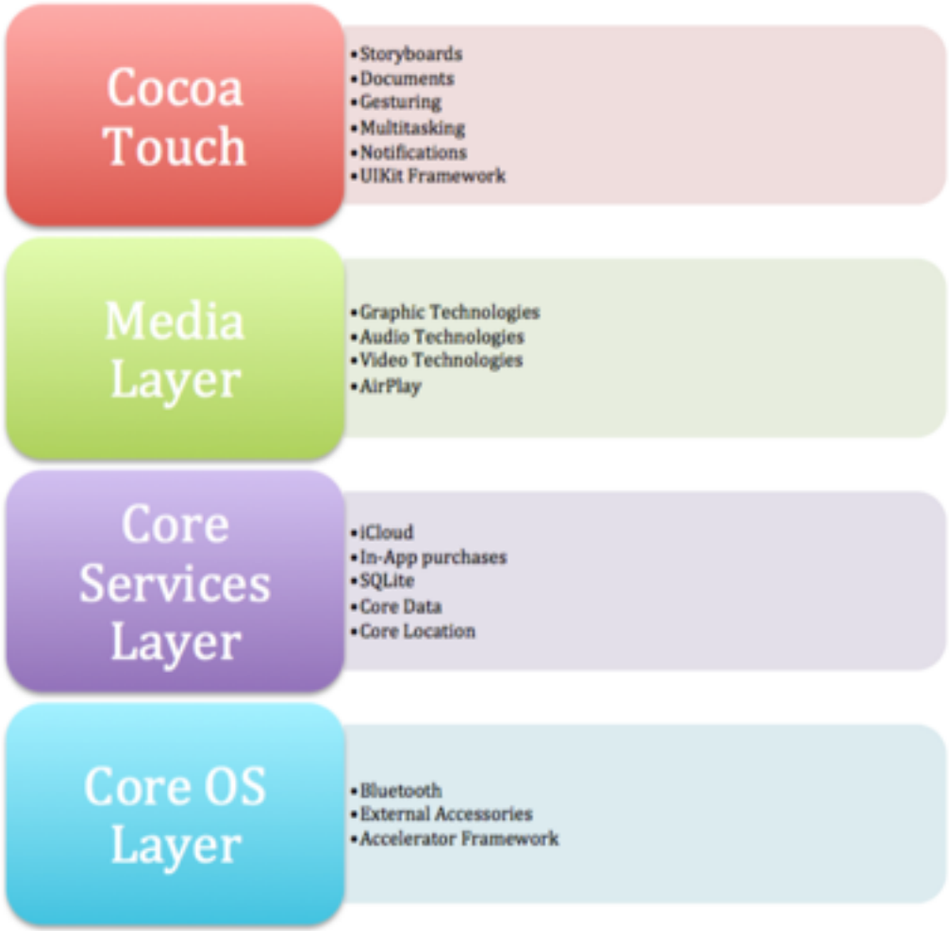
- Compiler
  - Converts code from (semi-)human-readable instructions to machine code

- LLVM (Low Level Virtual Machine)
  - Converts Swift, C, C++, Objective-C, Objective-C++ code to machine code



NOMENCLATURE (PT. 4)

- SDK: Software Development Kit
  - A library, or collection of software tools that are built to perform multiple operations to achieve complex functionality without needing to know the inner-workings of the code, also known as the implementation.
- Cocoa Touch
  - A collection of software that allows you to build apps for iOS
    - Foundation
    - UIKit
    - Dozens of other libraries



NOMENCLATURE (PT. 5)

- API: Application Programming Interface
  - The method in which one piece of software (e.g., Twitter) exposes its functionality to another piece of software (e.g., Your App).
  - This exposed functionality can then be used to transfer data between two pieces of software via a secure channel (e.g, Lincoln Tunnel).

NOMENCLATURE: SUMMARY

- iOS is an operating system that is used to communicate with hardware built by/for Apple.

- To make iOS apps, one must program in Swift (or Objective-C) and make use of the APIs in the Cocoa Touch framework.

- The LLVM compiler converts Swift code to machine code (e.g., something that iOS can understand), and executes your application.

DEVELOPER TOOLS

GETTING STARTED

DEVELOPER TOOLS

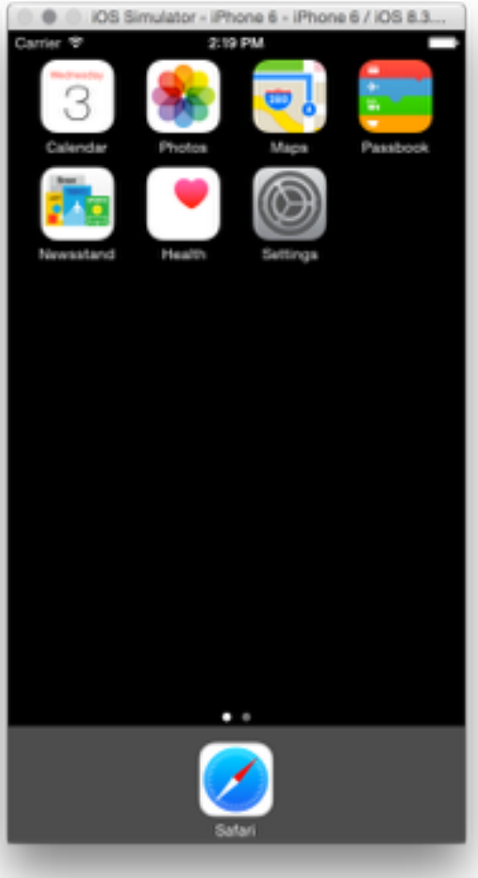
- Xcode
- iOS Simulator
- Github
- Slack

DEVELOPER TOOLS: XCODE

- Integrated Development Environment (IDE)
  - Write source code
  - Create views by dragging and dropping elements
  - Clean, Build, Compiler, Run your App
  - Debug your App

DEVELOPER TOOLS: IOS

- Simulate your app on your computer as you build it
  - General functionality
  - Multiple device resolutions
  - Location
  - Gestures
  - etc



DEVELOPER TOOLS: GITHUB

- Github is many things:
  - Version Control System
  - Collaboration Tool
  - Social Network

…you should already know all of this, as it was the part of the pre-work assignment :)

DEVELOPER TOOLS: SLACK

- Group communication tool
  - Includes:
    - Public Group Chats
    - Private Group Chats
    - Private Direct Messaging
    - File Upload

Think of it as Facebook Messenger on steroids!

SUPPLEMENTAL LEARNING RESOURCES

LEARNING RESOURCES

- Books
  - Official General Assembly MOB GitBook
    - http://mobbook.generalassemb.ly/
  - Official “Swift Programming Language”
    - iBook
    - Website

- Websites
  - Stack Overflow (http://www.stackoverflow.com)

GETTING STARTED

PRE-WORK DEBRIEF: GIT & GITHUB

GETTING STARTED

GIT & GITHUB

- http://www.github.com
- Create a free Github account
- Download Github Mac app
- Clone your first repository
- Pull your first set of commits from the cloned repository
- Push a new repository
- Commit your first change

GETTING STARTED

DEV WORKFLOW: XCODE & INTERFACE BUILDER

DEV WORKFLOW

- Learn about the different app templates
- Create a blank iOS App
- Learn how to add views in Interface Builder

CODE ALONG

GETTING STARTED

KEY OBJECTIVE(S)

Learn the flow of building a new project. Add UI elements to project and modify their properties.

TIMING

15 min 1. Work with a partner

5 min 2. Debrief

DELIVERABLE

Create a new project. The view controller should display text that contains a short bio. The project should have a button with the text “Goals”.

HOMEWORK

HOMEWORK

- In case you didn't do this already, you definitely should:
  - Read 'Understanding Mobile Devices'
  - Read Chapter 1 in the Official MOB Gitbook
- The real homework assignment:
  - Read Chapter 2 in the Official MOB Gitbook
  - Hands-On Git tutorial by Code School (https://try.github.io/)
    - This uses the Terminal app, which isn't required for this class, but will give you a better understanding of Git and how it works.
  - It's more powerful than using the Github Mac App.

Q&A