Tien-Che Tsai, 蔡典哲

Tien-Che Tsai, 蔡典哲

- Received a Master degree after National Taiwan University
  資訊網路與多媒體研究所 行動裝置與人機介面實驗室 指導教授為陳彥仰助理教授
- An iOS app Developer and Lecturer
  Worked for Tickle Labs, Inc and 台大資工系 資訊系統訓練班
- Also a backend developer with **Python/Django** and **Node.js**.
- Deploy web apps using cloud infrastructures like **Amazon Web Services**, **Heroku**, and **Google App Engine**.

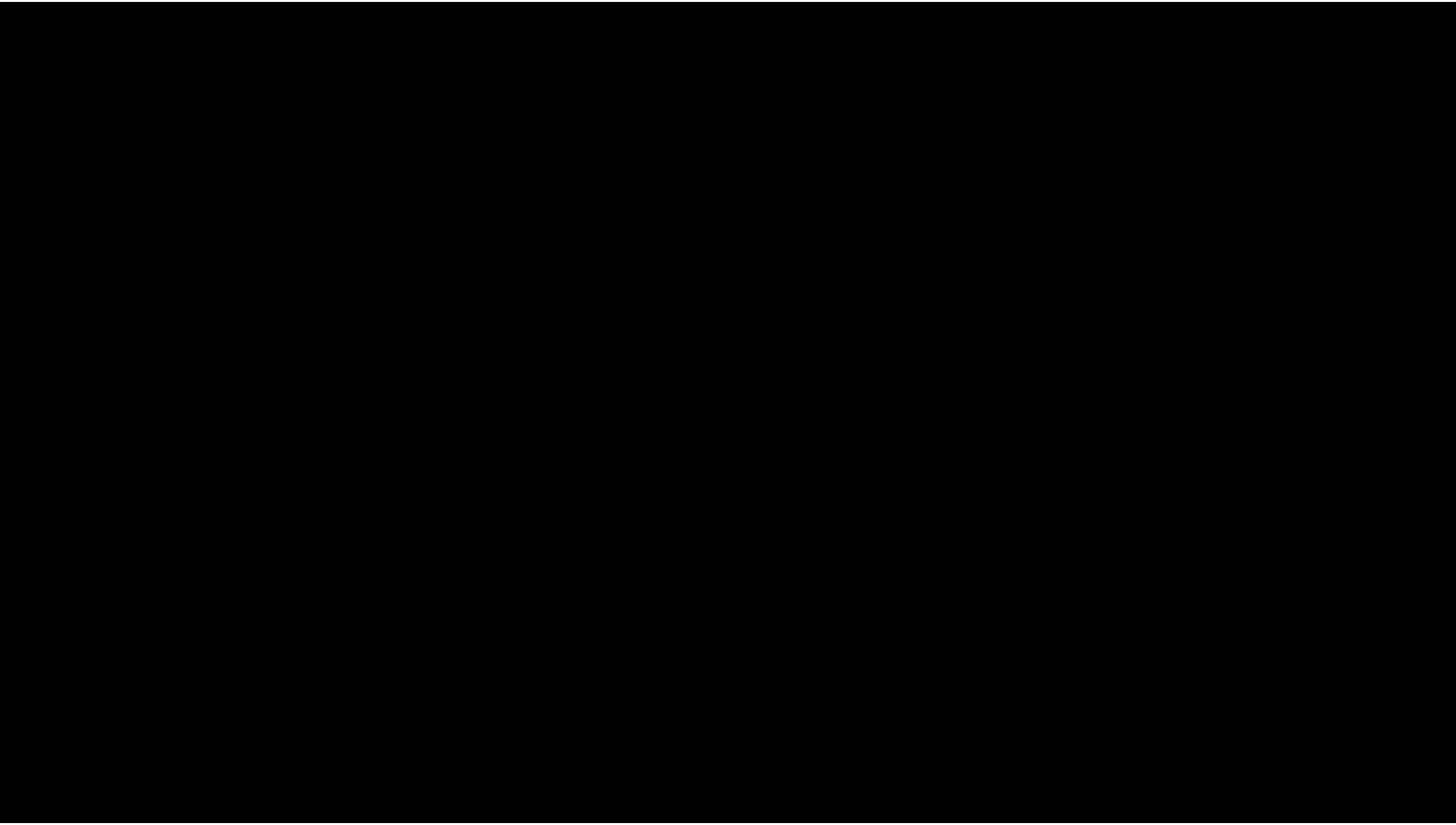
Tien-Che Tsai, 蔡典哲

- mail: sodas@icloud.com / tctsai@nccu.edu.tw
- twitter: @sodastsai
- LinkedIn: linkedin.com/in/sodastsai

- Slides and materials would be available at Moodle (moodle.nccu.edu.tw)

Mobile Application Development

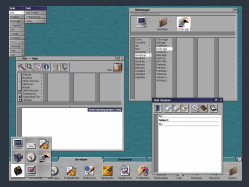
Spring 2016 @ NCCU CS




Jan. ‘07
Apple reinvents the phone

Apple I & II
1976

Macintosh
1984



NeXT
1985

iMac
1998

iPod & iTunes
2001

iPhone
2007

iPad
2010

WATCH
2015

tv
2015

How to build an app?

Instead of final exam, this class asks you to build an app for grading

找到問題或需求

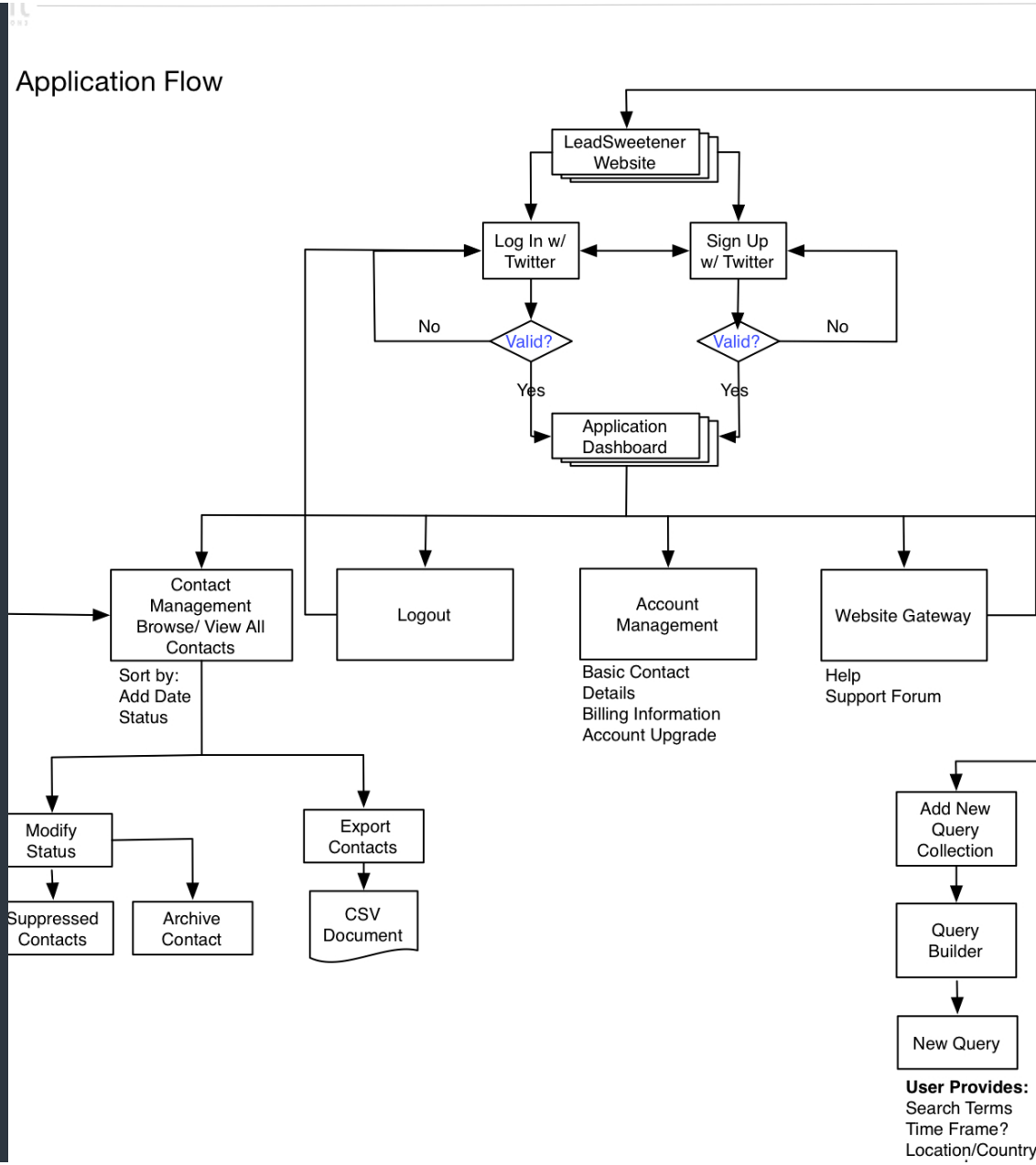
然後為了解決而設計一個服務

Solution

- 尋找現有的方案或服務，找出差異性或不足的地方
- 找出 Target Audience
  給老人用？給小孩用？給新手用？
- 現有技術可提供新的解決方式嗎
  哪些技術符合 Target Audience 及我們的需求，包含提供更好的擴充性

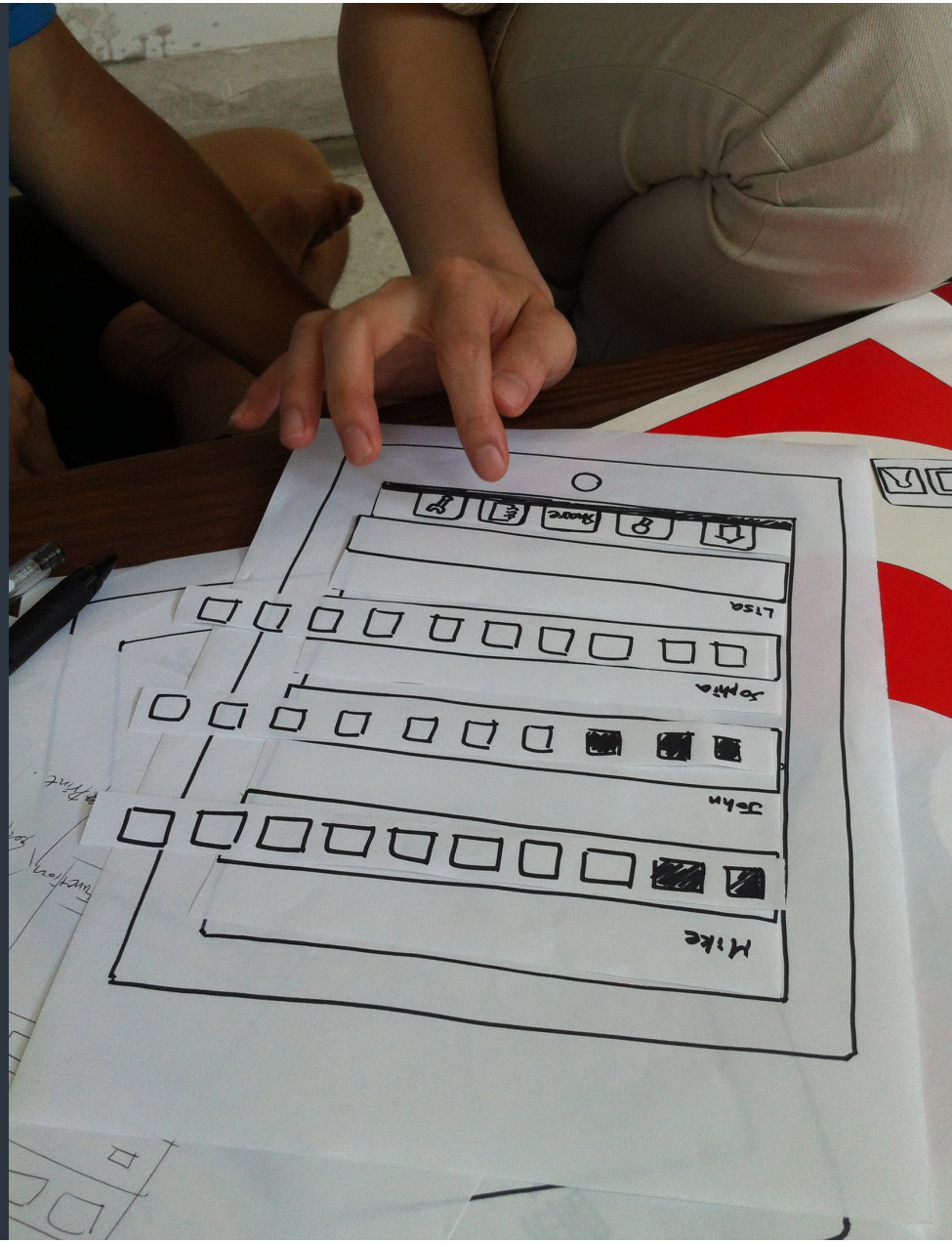
Flow Chart

- 服務/解決方案的運作流程
- App的操作及使用流程



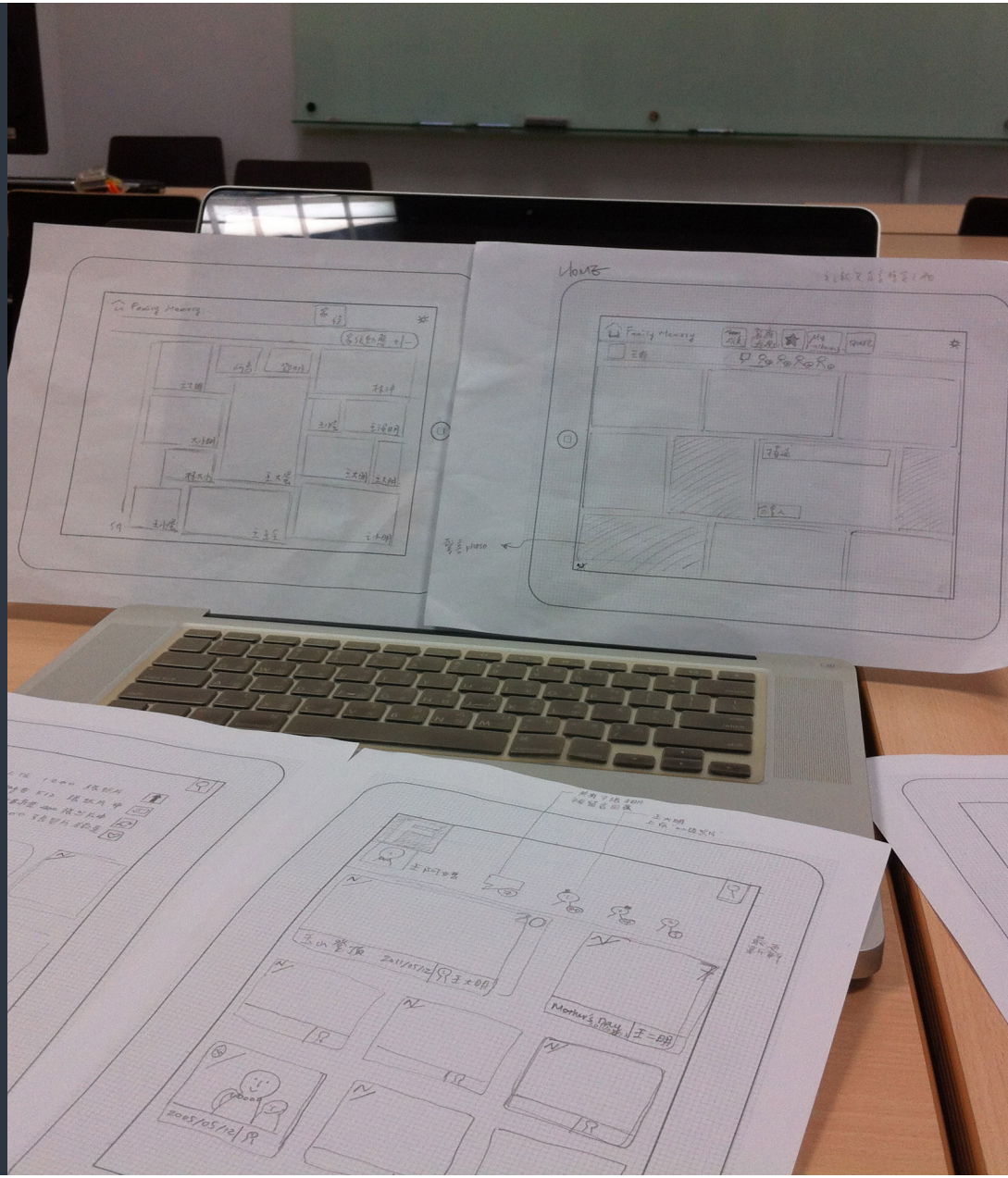
Wireframe & Paper Prototyping

- 參考其他服務設計流程與介面
  因為人類會借用過往經驗
- 利用簡單的工具快速製作服務的「原型」
  包含使用介面、操作流程與互動的設計
- 「手繪」方便修改與調整
  「紙」可以快速模擬互動模式
  「觀察」並一再修正
- 找 Target Audience 開始測試吧！



Mockup

- 「原型」歷經多次測試與修正後，可以製成 Mockup，準備實做
- 定義細部畫面元素位置、大小等，以及按鈕或控制項的行為
- 介面設計師 (UI)、互動設計師 (UX) 與程式設計師 (Programmer) 開始討論細節，分工進行實作



While writing code …

- Focus on your product, use Open-Source packages and libraries for common utilities or elements.
- Following design patterns and conventions saves you from common issues which occurred to most developers.
- Use automation tools and test driven development to speed up.

After writing code …

- Test with target audience and fix the solution of how you solve “problems” iteratively.
- Promote your work with social media and communities.
- Analyze / Measure what you have done and decide what to do next.
- Find a business model.

Preface

Goals

- Understand the newest and most popular programming language: Swift
- Be able to use Xcode to build apps for iOS devices *and even watchOS, tvOS and OS X*
- Real-life OOP experience
- Use git and Work with open source projects/libraries

Prerequisites

- Programming Experiences
  This is not a class for programming newbies. You should have done at least a final project of other programming class before.

- Bring your own Mac (required) and iOS device (recommended).
  We would target on iOS 9, so you should have your Mac at least OS X El Capitan 10.11 and Xcode 7.0 or newer version installed.
  You could install Xcode from the Mac App Store.

Prerequisites

- Object Oriented Programming, like C++ or Java.

- Classes and Instances
- Members and Methods
- Inheritance and Override
- Polymorphism & Overloading
- Encapsulation
- Interfaces or Virtual classes

Grading

- 4~5 Personal homework assignments (**40%**)
  May include a small and simple quiz or report
- a Final team project and presentations (**45%**)
  Including a proposal presentation, a progress review presentation, and a final demo representation. A team would be composed by 4 ppl.
- Participation of in-class interaction and online group discussion (**15%**)
  You are encouraged to ask problems, discuss issues of your homework or final projects, post some related news, and even attend meet-up events held by developer communities. No roll call in the class.

Auditors

- Leave the seat and resources to the students who enrolled in this class first.
- You're welcome to participate the discussion with enrolled students.
  Register via moodle.nccu.edu.tw
- If possible, you should also finish all the homework and assigned tasks, even the final project if you could group a team for this.

Feedback

- An anonymous Google Form for feedback of classes in this course.
  Don't be shy, you could response any questions, fixes, or suggestions after the class as soon as possible.
- Sample projects would be hosted on GitHub, submit a PR or an issue if something goes wrong.
  The Keynote slides maybe hosted there too.

Syllabus

Syllabus

Introduction to Xcode and Swift & Design your app

3/3	Xcode, Storyboard, and the Swift Language
3/10	Use MVC Pattern to create a simple app
3/17	Structure of an App & Foundation framework
3/24	Common UI Elements & App Design
3/31	Human Interface Guideline & Midterm Team Project Proposal

Syllabus

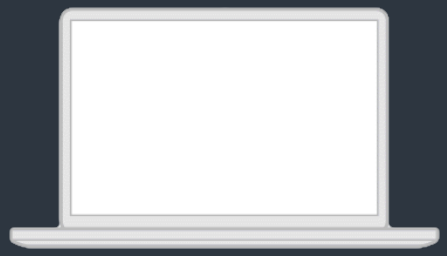
Implement your app & Interact with the world

4/7	Open Source libraries, Networking, and Web/Social services
4/14	File Storage, Database, CoreData, and iCloud
4/21	Notification & Accessories, Location, and Motion Sensors
4/28	Dive into Swift deeply, Brief introduction to Objective-C, and Gaming SDK
5/5	Midterm Project Progress Presentation and Review

Syllabus

Analyze and Improve your app

5/12	Advanced Xcode (Debugging, Automation, and Testing) & Analytics
5/19	CoreGraphics and CoreAnimation & Advanced Concurrency
5/26	[TBA]
6/16	Final Team Project Presentation
6/23	[TBA] *Final Team Project Presentation*

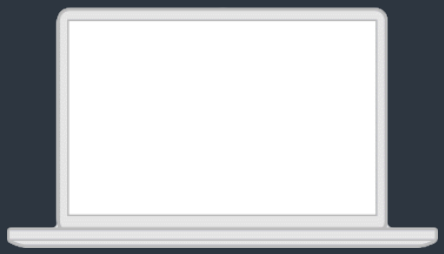


OS X

iOS

watchOS

tvOS



OS X

iOS

watchOS

tvOS

AppKit *(Cocoa, OS X)*

UIKit *(Cocoa Touch, iOS, watchOS, and tvOS)*

MapKit / WatchKit / …

Cocoa & CocoaTouch

CoreGraphics / CoreImage / Metal / AVKit / SpriteKit / …

Media

CFNetwork / CoreData / CoreMotion / CoreLocation / HomeKit / …

Core Services

BSD System *(Unix)* / GCD / CoreBluetooth / Accelerate / Security / …

Core OS

Xcode

LLVM Compiler

The Swift Language

The Swift Language

- A modern language for iOS, OS X, watchOS, and tvOS apps.
- It's built on the best of C and Objective-C,
  without the constraints of C compatibility.
  Apple characterized Swift as **the Objective-C without the C**
- A new language that is fully utilized the LLVM compiler infrastructure.
- Works with current C and Objective-C codebase including the
  Cocoa and CocoaTouch frameworks.

Swift v.s. Objective-C

- Objective-C is a very old language. (1983) It's also known as a superset of the C language.
  This makes all C libraries available, but also constrains the possibility of using modern programming paradigms like closure.
- Objective-C has a very clear and expressive method naming (called "named parameters")

Swift v.s. Objective-C

- Swift is derived from Objective-C which also keeps the “named parameters”, late binding, and dynamic dispatch.
  These makes the language also expressive and easier to be compatible with Objective-C. Also makes the language extensible and flexible to use.

- These features have also performance and safety trade-offs.
  Swift addresses these issues by adding new annotate syntax, inferring types, and patching code to fix common issues during the compile time.

- The performance is greater than Objective-C.
  The LLVM team invested considerable effort in aggressive optimization for Swift.

Swift v.s. Objective-C

- Swift is also influenced by modern programming languages like **Ruby**, **Python**, **Haskell**, and **JavaScript**.
- Modern paradigm/concepts like “Type inference”, “Generics”, and “Functional programming” are also supported.
- Swift is a “Protocol-oriented programming” language which provides great extensibility to developers.

Xcode

https://goo.gl/7aJXSm

Simulator

git

https://try.github.io/
https://www.codecademy.com/learn/learn-git

moodle.nccu.edu.tw