Lecture 1

Beginning iOS

czeluff@westminstercollege.edu
@chadzeluff

Homework

- Xcode installed
- iOS 7.x simulator installed
- Create a GitHub account
- Clone this repo - https://github.com/czeluff/Westminster-iOS

Header Files (.h)

- “Definition” files
- Declare public interfaces here
- Includes public variables (properties), functions, protocols, etc.

Implementation Files (.m)

- Logic goes here
- Declares private interface
- Includes private variables, functions, & protocols
- Always imports its corresponding header file

Precompiled Header (.pch)

- Compiled before any other file
- Global - import a file here if you want it accessible throughout every file
- Imports - <> for library / external files, “” for your own files

Preference File (.plist)

- Included plist file specifies default preferences for app:
- Name below icon on Home Screen
- Version Number
- LOTS that can be added/removed here
- XML-based

- Custom plist files to handle other preferences

Foundation & UIKit

- Foundation includes classes beginning with “NS”
- Foundation deals primarily with data
- UIKit includes classes beginning with “UI”
- UIKit focuses on iOS-specific User Interfaces

main.m

- First file executed by Objective-C (and base C) applications
- Needs a “main” function
- public static void main (String[] args)
- Specifies the Application’s Delegate - which file will handle entering / exiting the app
- Don’t touch it

Documentation Overview: UIApplicationMain

AppDelegate

- Inherits from UIResponder
- Implements the UIApplicationDelegate protocol
- Has a property called window

Properties

- In Objective-C, we access all variables through getter/setter
- Using the property does just that!
- nonatomic - 99% of the time, it works every time!
- strong, weak, retain, assign, readonly, readwrite, copy, and more!
- Use strong - it means this class “owns” that property
- @property (nonatomic, strong) ClassType *propertyName;

UIApplicationDelegate Methods

Now with more NSLogs!

Objective-C Syntax

- Access properties via dot-notation or bracket notation: self.window
- Access functions through bracket notation [self doSomething]
- To be more concise, I recommend using dot-notation whenever possible
  [self.window functionOnTheWindow]
  [[self window] functionOnTheWindow]

Objective-C Syntax

- Why does bracket notation work on properties?
- Because technically it’s also a function!
- Getter on right side of = sign
  UIWindow *window = self.window;
  UIWindow *window = [self window];
  (Don’t use the words ‘get’/‘set’ to name your properties)
- Setter on left side of = sign
  self.window = [[UIWindow alloc] init];
  [self setWindow: [[UIWindow alloc] init]];

Objective-C Syntax

- Function names interweave variables.
  Self-describing
  + (instancetype)dateWithTimeInterval:
  (NSTimeInterval)seconds sinceDate:(NSDate *)date

- Minus (-) sign indicates instance function/method

- Plus (+) sign indicates class function/method
  Also called static method

- Return type indicated within parentheses

More Xcode Walkthroughs