Reading Assignment I:

Intro to Swift

Objective

The goal of our first week of reading assignments is to start to get a handle on this new language you must learn called Swift. We'll cover basic stuff like variables and control flow, but also a bit of a trickier topic: closures (functions as types).

Most of you have not had experience with Objective-C, but don't worry about that. Nothing in the Swift documentation really assumes that. However, if you have never programmed in C (or C++ or any other variant), then Swift might be extremely new to you (but hopefully still not too steep a hill to climb to learn).

Read all of the material referenced here by the start of Lecture 3.

Materials

• The reading in this assignment comes from two on-line documents: the [Swift Programming Language](#) and the [Swift API Guidelines](#).

Swift Programming Language

Read the sections described below in the <u>Swift Programming Language</u>. To better utilize your valuable time and to emphasize important concepts, the sections in the reading have been broken down into four categories:

Red sections are VERY IMPORTANT and might be more difficult to understand. Read these carefully.

Yellow sections are important, but won't be as difficult to understand.

Green sections are important, but cover very basic, simple stuff (much of it just like C).

Grayed-out sections are not required reading (this week). They may be in future weeks.

Don't gloss over reading any NOTE text (inside gray boxes)—many of those things are quite important. However, if a NOTE refers to Objective-C or bridging, you can ignore it.

If there is a link to another section in the text, you don't have to follow that link unless what it links to is also part of this week's reading assignment.

In the **Language Guide** area, read the following sections in the following chapters:

The Basics

Constants and Variables
Comments
Semicolons
Integers
Floating-Point Numbers
Type Safety and Type Inference
Numeric Literals
Numeric Type Conversion
Type Aliases
Booleans
Tuples
Optionals
Error Handling
Assertions

Unicode variable and constant names (e.g., 🔥) can be fun, but you will be held accountable for the quality of your naming (of all kinds) and readability in your code. Do not put semicolons at the ends of lines (only use them to (rarely) separate two statements on a single line).

Basic Operators

Most of this section is almost identical to C (that's why most sections are green).

Terminology
Assignment Operator (don't worry about references to tuples)
Arithmetic Operators
Compound Assignment Operators
Comparison Operators
Ternary Conditional Operator
Nil Coalescing Operator
Range Operators
Logical Operators

Strings and Characters

String Literals
Initializing an Empty String
String Mutability
Strings Are Value Types
Working with Characters
Concatenating Strings and Characters
String Interpolation
Unicode
Counting Characters
Accessing and Modifying a String
Comparing Strings
Unicode Representations of Strings

Collection Types

Mutability of Collections
Arrays
Sets (ignore first NOTE box in this section)
Performing Set Operations
Dictionaries

If you try to access an index in an `Array` higher than its `count`, your program will crash.

There is also a function called `last` in `Array` that returns an `Optional` (i.e. it will return `nil` if, and only if, the `Array isEmpty`).

Note that the += array operator takes *another array* as the right-hand-side (not an element to add to the array).

Control Flow

For Loops (for-in might be new to some of you as well as ranges, e.g. `1..5`)
While Loops
Conditional Statements (especially switch, ignore Tuples, Value Bindings & Where)
Control Transfer Statements (but ignore Labeled Statements)
Early Exit
Checking API Availability

The `switch` statement is much more important in Swift than you are used to in C or other languages.

Functions

Defining and Calling Functions
Function Parameters and Return Values (ignore Functions with Multiple Return Values and Optional Tuple Return Types)
Function Argument Labels and Parameter Names (ignore Variadic and In-Out Parameters)
Function Types (this will probably be the biggest challenge section for many of you)
Nested Functions

If you started this assignment after the first lecture, you might want to pause here on this reading assignment until the end of the second lecture (which will cover closures, enums and the difference between classes and structs).

Closures

Understanding that a description of a function (i.e. it's arguments and return value) can be a first-class "type" (just like an `Array` or an `Int`) in Swift is important. Many iOS APIs have closures as arguments.

Closure Expressions (be sure to understand Function Types above first)
Trailing Closures
Capturing Values
Closures are Reference Types
Nonescaping Closures
Autoclosures

Enumerations

Enumeration Syntax
Matching Enumeration Values with a Switch Statement
Associated Values
Raw Values
Recursive Enumerations

Classes and Structures

Comparing Classes and Structures
Structures and Enumerations are Value Types
Classes are Reference Types
Choosing Between Classes and Structures
Assignment and Copy Behavior for Strings, Arrays, and Dictionaries

Note that this week we only know two ways to *create* a `class` or `struct`: by putting the name of the `class` or `struct` followed by empty parentheses, e.g. `let x = VideoMode()`, or, for `structs` only, using the name of the `struct` with all of its variables getting a value, e.g. `let hd = Resolution(width: 1920, height: 1080)`. We'll learn all about much more powerful ways to initialize `classes` and `structs` next week.

Properties

Stored Properties (ignore Lazy Stored Properties and Stored Properties and Instance Variables)
Computed Properties
Property Observers
Global and Local Variables
Type Properties

Swift API Guidelines

Read the Swift API Guidelines document in its entirety.

Given that you are completely new to Swift, some of what is in this document will be a bit hard to fully absorb at first. But familiarizing yourself with what is in this document is crucial to writing good Swift code. So, for this assignment, the goal is to know what's there rather than completely and fully master the guidelines right off the bat. As the quarter progresses, you should eventually become an expert namer of properties, methods and other Swift constructs. This will require you to refer back to this document often.

Be sure to click everywhere that it says "MORE DETAIL".

Pay special attention to the "Write a documentation comment" section.

Pay special attention to the "Follow case conventions" section.

Pay special attention to the entire "Argument Labels" section.

You can ignore (for now), points that reference Protocols. When we learn about Protocols next week, be sure to check back with this document after that.

Since you haven't seen initializers yet in Swift, the section "Initializer and factory method calls" will not make a whole lot of sense to you, but file the information away for when you learn about initializers next week.

You can also ignore the final section (Special Instructions) for now.