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).

Materials

- All of the reading comes from this Swift Programming Language document.

Sections to Read

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.

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 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
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

Don't worry about the NOTE box in the chapter introduction. We'll talk about Objective-C classes like `NSString` next week.

String Literals
Initializing an Empty String (don't worry about "initializer syntax" yet)
String Mutability
Strings Are Value Types (don't worry about NOTE)
Working with Characters
Concatenating Strings and Characters
String Interpolation
Unicode
Counting Characters
Comparing Strings
Unicode Representations of Strings

A `String` can also be converted to an `Array<Character>` like this:
`let myArrayOfCharacters = Array(myString).characters`. This is not mentioned in this chapter because this chapter comes before the chapter on `Array`.

Collection Types

Mutability of Collections
Arrays (don't worry about the talk of initializers toward the end of this section)
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)

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 Parameter Names
Function Types (this will probably be the biggest challenge section for many of you)
Nested Functions

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

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

Ignore references to enumerations (`enum`) throughout. Enumerations are important, but we will be covering that in detail next week.

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