Reading Assignment I: Intro to Swift

Objective

The goal of our first reading assignment is to start to get a handle on this new language you must learn called Swift. This one covers mostly basic stuff like variables and control flow, methods and properties, `Array`s and `String`s, but note that Swift does some of this basic stuff in powerful ways (e.g. labelled arguments to functions, pattern-matched switch statements, functions as first-class types, mixing object-oriented and functional programming technique in the same programming language, etc.).

Below you will find a list of the sections in the Swift Programming Language document that you must read. It is color-coded to help you be efficient in consuming all this information. For example, Gray topics are ones which you do not have to read this time (i.e. you'll read about them next time or the time after). The other color-coding (described below) may mean different things to different people depending on your experience with various programming languages.

Learning a new language should be something that becomes "old hat" for you over time. Most veteran computer scientists probably know at least a dozen programming languages (some know dozens). If you are such a veteran (i.e. you know a dozen or more programming languages), the color-coding will mean very little. You'll recognize the various elements of the language easily and will just have to scan to figure out Swift's specific syntax.

Strangely, if Swift is only the second programming language you've ever had to learn, then the color-coding may well not mean much to you either since you'll probably have to carefully read **all** of the sections to understand this new language.

But if you lie somewhere in the middle, then the color-coding might be a help. Unhighlighted topics are normal features found in almost all programming languages with nothing particularly tricky about Swift's implementation thereof. You might find yellow topics to be a bit unusual (depending on your experience) and so you want to pay some attention when reading about them. Red topics are very important and/or are likely to be completely new to you unless you're an above-mentioned veteran. These sections require close and careful reading.

There are reading assignments in the first few weeks of this course, but not after that. This reading is a bit larger than the other two assignments both because the programming assignments at the start of the quarter are relatively easy (and so won't be as time-consuming as usual) and because, while there's a lot to read this week, a whole lot of it is pretty basic stuff (although there is some more complex stuff mixed in too, so beware). Plus you get a few extra days to do this one.

Due

Read all of the material referenced below before the start of Lecture 4.

Set aside sufficient time because there's quite a bit of reading here. This is the highest volume reading assignment week, but also has the most stuff that you're likely to be familiar with from other languages, so hopefully some of this week's content will be stuff you'll be able to move through relatively quickly. Again though, slow down a bit when a topic is highlighted in yellow or red.

Materials

We recommend scrolling around through this document to give yourself an idea of how much time you want to budget for this before you dive in. For quick reference, in addition to A Swift Tour and the Swift API Guidelines you'll be reading most of the content in 11 ½ of the first 14 chapters listed along the left-hand side of Swift Programming Language (The Basics through the first half of Initialization, excluding Subscripts or Inheritance).

Playgrounds

The documentation has a number of gray-boxed areas titled "Experiment". If you are adventurous, download the Swift Playgrounds application on your Mac or iPad and copy/paste the code referred to in the Experiment box and then try to do what the Experiment suggests.

Just fire up Swift Playgrounds, click the plus sign above Playground in the lower left More Playgrounds area, then double click on the My Playground it creates. Inside there, just start typing or copy/pasting any code you want and click "Run My Code" in the lower right. The values of variables and results of function calls will show up as little icons along the right-hand side.

Swift Programming Language

Don't gloss over reading any `NOTE` text (inside gray boxes) since 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.

Always read the overview at the top of each major section (e.g., in The Basics, be sure to read the part that starts "Swift is a new programming language for iOS, macOS, watchOS, and tvOS app development…").

Start with A Swift Tour. It is a good place to "find your bearings" with the Swift language. We'll see at least a little bit of all the topics in the sections listed here in the first couple of weeks of lecture and this is a great place to read more about them. This covers most of the core of Swift. The rest of the reading (both this week and in coming weeks) is just going to dive deeper in all of these topics.

> Simple Values
> Control Flow
> Functions and Closures
> Objects and Classes
> Enumerations and Structures
> Concurrency
> Protocols and Extensions
> Error Handling
> Generics

The Protocols and Extensions section (note that it is in gray) is *super* important, so much so that we'll be covering it in slides in lecture in week 3 (which is why we're not requiring you to read that section at this time). Having said that, you're always welcome to read ahead on anything in this document (we're just trying to keep your overall amount of reading to a manageable level). If you want to try to read that section and link it in your mind to the demo so far, know that `View`, `Identifiable`, and `Equatable` are all protocols. And when we talk about "constrains and gains" or "behaves like" in lecture, we're talking about protocols.

In the Language Guide area, read the following sections in the following chapters. It is important to understand the information in these sections for you to be able to continue to follow along in lecture as the quarter progresses, so don't blow off this reading.

By the way, 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, so name things wisely. Also, we do not put semicolons at the ends of lines in Swift (we only use them to (very rarely, if ever) separate two statements on a single line).

The Basics

Note that the type between a full heading (like Constants and Variables) and a subheading (like Declaring Constants and Variables or Type Annotations which are subheadings of Constants and Variables) is not very dramatic in this document. When a full heading is listed here, we mean "the entire section, including all sub-sections." So when you see "Constants and Variables" below, it means all of its sub-sections (Declaring Constants and Variables, Type Annotations, Naming Constants and Variables, and Printing Constants and Variables). "Comments" is the next full heading.

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 and Preconditions

Basic Operators

While ternary conditional operators are very simple, they also are vary often used in SwiftUI (as the arguments to View modifiers). That's why they are highlighted here.

Terminology
Assignment Operator
Arithmetic Operators
Compound Assignment Operators
Comparison Operators
Ternary Conditional Operator
Nil-Coalescing Operator
Range Operators
Logical Operators

Strings and Characters

`String`s are very powerful in Swift since they can represent text in almost any language on earth (which adds some understandable complexity). Because of that, this is one of the sections that might be fairly time-consuming to read through. On the other hand, you're also probably not going to be caught flat-footed if you aren't an expert in Swift `String`s, so if you need to postpone this to next week because you're time-constrained, you may.

You'll also see a reference to Strings as being a "value type". This is an important concept and is a part of the reading this week (see further down in this document).

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
Substrings
Comparing Strings
Unicode Representations of Strings

Collection Types

`Array` (and to a lesser extent `Dictionary`) is one of the most powerful and often-used `structs` in all of Swift. This is a type that you will want to master early, not only by reading the sections below carefully, but also by fully acquainting yourself with the **documentation page** for `Array`. `Array` has a vast array (pun intended) of built-in capabilities that will save you lots of coding time down the road if you know they're there.

Note that this section starts off talking about mutability. This is simply the difference between something being stored in a "let" variable (immutable, i.e. "a constant") versus a "var" variable (mutable, i.e. "a variable"). Swift handles mutability very powerfully and elegantly.

Mutability of Collections
Arrays
Sets
Performing Set Operations
Dictionaries

Control Flow

There are a lot of words in this section (and in the next one about `functions`), but most of it will be very familiar (save for maybe the yellow sections), so likely this will not be as time-consuming to read as it may appear at first.

- For-In Loops
- While Loops
- Conditional Statements
  - If
  - Switch
    - No Implicit Fallthrough
    - Interval Matching
    - Tuples
    - Value Bindings
    - Where
    - Compound Cases
- Control Transfer Statements
  - Continue
  - Break
  - Fallthrough
  - Labeled Statements
- Early Exit
- Checking API Availability

Functions

The fact that functions are a normal, first-class type in Swift is crucial to its design.

- Defining and Calling Functions
- Function Parameters and Return Values
- Function Argument Labels and Parameter Names
  - Specifying Argument Labels
  - Omitting Argument Labels
  - Default Parameter Values
  - Variadic Parameters
  - In-Out Parameters
- Function Types
- Nested Functions

Closures

We referred to closures as "in line `functions`" in the demo and showed off most of what is covered in the two subsections here that you are required to read this week. But closures are more powerful than just "in line `functions`" and we'll read about that in the grayed-out sections in the next reading assignment.

- Closure Expressions
- Trailing Closures
- Capturing Values
- Closures are Reference Types
- Escaping Closures
- Autoclosures

Enumerations

Enumerations are surprisingly important in Swift, so be sure to pay close attention to this chapter.

- Enumeration Syntax
- Matching Enumeration Values in a Switch Statement
- Iterating over Enumeration Cases
- Associated Values
- Raw Values
- Recursive Enumerations

Structures and Classes

Understanding that `structs` are value types and get copied around all over the place and are usually immutable is fundamental to adjusting to the functional programming methodology in Swift (which is why that section is highlighted in red here). The languages you are likely to be coming from will probably be based on (object-oriented) reference types.

- Comparing Structures and Classes
- Structures and Enumerations Are Value Types
- Classes Are Reference Types

Properties

We saw computed properties a couple of different times in the demo (e.g. `var body`). Note that the section on Computed Properties here goes into how to create computed properties that you can also *set* (i.e. that aren't read-only) and we can wait until next week to see that (which is why it is in gray here).

For some reason Type Properties and Type Methods confuse many students (i.e. they get them confused with normal (instance) functions and properties), but you can just think of them as "global" properties and `func`tions that are "name-spaced" to be associated with a type.

- Stored Properties
- Computed Properties
- Property Observers
- Property Wrappers
- Global and Local Variables
- Type Properties

Methods

- Instance Methods
  - The self Property
  - Modifying Value Types from Within Instance Methods
  - Assigning to self Within a Mutating Method
- Type Methods

Subscripts

Inheritance

We're kind of going to gloss over object-oriented programming in Swift because (outside of your ViewModel), you're not going to use it much in SwiftUI.

Initialization

Setting Initial Values for Stored Properties
Customizing Initialization
Default Initializers
Initializer Delegation for Value Types
Class Inheritance and Initialization
Failable Initializers
Required Initializers
Setting a Default Property Value with a Closure or Function

Swift API Guidelines

Read this 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 week, 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.

You can also ignore the final subsection of the final section "Special Instructions -> Take extra care with unconstrained polymorphism". We won't be doing anything with the `Any` and `AnyObject` types.