Reading Assignment IV: Less Important Stuff

Objective

We are going to emphasize one thing mightily this week: the Swift API Guidelines document. You should have read this in its entirely at least twice (since it was assigned to you every week). Now you should read it and fully understand all of it (except the `Any/AnyObject` stuff). **You will be held accountable going forward for choosing good names for types, variables, functions and their arguments in your assignments and final project.**

As for the rest of the programming reference guide, most of it is esoterica as far as we're concerned as SwiftUI developers (e.g. OOP topics). Definitely these topics will be the least important thing you learn in this course so it's fine to just skim through them. Focus this Reading Assignment intensively on really understanding the Guidelines mentioned above and going back and catching up on any past reading that you might have had to rush through.

Due

You should have all of this reading done by lecture 14.

Materials

You'll continue reading from the same document(s) (e.g. Swift Programming Language) as last week.

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 provides many fundamental data types …").

Everything in A Swift Tour should now make sense to you. Review it this week to be sure.

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, so don't blow off this reading.

Strings and Characters

Hopefully you did not continue to procrastinate this section! If you did, the time of reckoning has arrived. Read it!

- Unicode Representations of Strings

Control Flow

Generally control transfer out of loops and such is to be avoided.

- Control Transfer Statements
  - Continue
  - Labeled Statements

Enumerations

We probably want to avoid recursive enumerations.

- Recursive Enumerations

Inheritance

Object-oriented programming is not important in SwiftUI. Read this section to get an idea of Swift's support for OOP in case maybe one day you want to do an OOP project and want to code in Swift. Swift's support for OOP is quite strong because the previous way to develop for iOS (called UIKit) was entirely object-oriented.

Initialization

Ditto the above about OOP.

- Class Inheritance and Initialization
- Required Initializers

Deinitialization

Ditto the above about OOP.

Macros

Macros were recently added to Swift and have provided a relatively clean mechanism for Apple to let you focus more on the details of your own code without having to worry about underlying mechanisms (like "how does a SwiftUI `View` know it has to update itself when a var in an `@Observable` changes?"). You'll be reading this week about how these things actually work. I doubt you'll be creating your own macros any time soon, but at least you'll know it's possible.

Type Casting

Type casting has become more and more unnecessary as SwiftUI completely takes over from UIKit as the way people write iOS apps. Generally a well-formed Swift application would not use very much type casting, but occasionally we see it for backwards compatibility with UIKit-era APIs and we also might find ourselves doing it if we are using `any` with a protocol (to unbox it and get at the actual type). If you find yourself doing type casting in any other context, there's probably a better way to do it (via protocols and generics usually).

Protocols

These remaining protocol topics are all related to the backwards compatibility with UIKit.

Delegation
Class-Only Protocols
Optional Protocol Requirements

Automatic Reference Counting

This is how reference types get their memory cleaned up. Since we use value types the vast majority of the time, this is mostly just for your general understanding of how Swift allocates memory in the heap.

Memory Safety

If you for some reason have to pick through raw data in memory (like something mapped from a data file or raw data from a device or image source or something), then this section is something you should understand.

Access Control

Just this one last OOP-related access control topic.

Subclassing

Swift API Guidelines

Read this Swift API Guidelines document in its entirety.

Read this over **yet again** this week. You should now be fully a master of this information. 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 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.