

# Reading Assignment III: The Rest of Swift

---

## Objective

Time to read about the rest of Swift!

One of the more voluminous items here is object-oriented programming and related topics (including inheritance, initializer inheritance, deinitialization and the memory management topics). This is also one of the least important topics for SwiftUI (which is mostly functional programming). If you ever want to do any programming in UIKit, though (i.e. pre-SwiftUI iOS development), you will definitely need to know object-oriented programming in Swift very well. So at the very least familiarize yourself with how Swift does object-oriented programming even if you won't be able to call yourself an object-oriented Swift programmer after reading this.

Important topics are marked in **yellow** or **red** below. Subscript-related topics are “medium importance.”

---

## Due

Read the rest of [Swift Programming Language](#) (topics listed below for reference) as soon as your schedule allows (there are no more reading assignments after this one). Reading Assignment 2 is not due until the start of Lecture 7, but there's no reason you can't get started on this assignment once you have finished Reading Assignment 2, even if that's before the start of Lecture 7.

---

## Materials

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

As threatened in lecture, you should read [Apple's Human Interface Guidelines](#).

---

## 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 you read something and don't understand it, that's what Piazza is for! Don't be shy.

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 ...").

You'll now have read everything in [A Swift Tour](#).

### Error Handling

---

## Human Interface Guidelines

You should read [Apple's Human Interface Guidelines](#) as well.

In the [Language Guide](#) area, read the rest of the sections (recounted below for easy reference).

## [The Basics](#)

Thrown errors are not all that common, but you need to about them or you won't be able to call functions marked throws. There's another whole section on it below.

- Error Handling
- Assertions and Preconditions

## [Strings and Characters](#)

If you've been postponing reading this section for time reasons, definitely finish off this entire section this week.

- 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

## [Control Flow](#)

This topic is needed if you want to write apps that work on older versions of iOS.

- Checking API Availability

## [Closures](#)

Autoclosures are kind of a cool feature, but we don't actually use them that much.

- Autoclosures

## Properties

Probably the **most important** and useful sections this week.

Property Observers  
Property Wrappers

## Subscripts

Read this entire section. Subscripts can actually be used to create some pretty cool API in specific circumstances. But it's really nothing more than syntactic sugar in the end.

## Inheritance

We've been postponing reading about object-oriented programming in this course because, frankly, you don't need to do very much of it to program in SwiftUI. However, most of you probably come from object-oriented programming backgrounds, so now it's time to see how Swift implements this.

## Initialization

The first and last sections are pretty dense reading and probably won't sink in very well since you're not doing any object-oriented programming in this course. But at least understand that there are restrictions when it comes to inheritance of initializers in object-oriented programming.

The other section (failable initializers) is actually quite useful.

Class Inheritance and Initialization  
Failable Initializers  
Required Initializers

## Deinitialization

Another object-oriented programming section.

## Error Handling

Again, it's important to be able to know how to call functions marked with `throws`.

## [Type Casting](#)

Type casting was a lot more common and necessary in the less-strongly-typed language of Objective-C. Most of our type-casting nowadays has to do with interacting with the old Objective-C APIs (like `UserDefaults` or `NSItemProvider` for example). The type `Any/AnyObject` are pretty much never used in SwiftUI.

## [Extensions](#)

All that remains here is handling subscripts in extensions.

### Subscripts

## [Protocols](#)

These two sections are mostly tied to integration with Objective-C. See the comments about Type Casting above.

### Checking for Protocol Conformance Optional Protocol Requirements

## [Generics](#)

The final piece of the Generics puzzle is how protocols do “don’t cares.” Also, now that we’ve read about subscripts we can see how those fit into the world of Generics.

### Associated Types Associated Types with a Generic Where Clause Generic Subscripts

## [Automatic Reference Counting](#)

Only applicable to reference types (i.e. classes mostly). Thus not really a big issue for SwiftUI. You don’t need to really understand this section to program in SwiftUI.

## [Memory Safety](#)

This also is a bit of arcana. Good knowledge to have, but not necessary to know.

## Access Control

You've basically got most of what you need to know about Access Control already (`private` and `private(set)` and `fileprivate`). This section is covering some of the corner cases of Access Control that can arise.

Custom Types

Subclassing

Constants, Variables, Properties, and Subscripts

Initializers

Protocols

Extensions

Generics

Type Aliases

---

## Swift API Guidelines

Read this [Swift API Guidelines](#) document in its entirety.

Read this over **again** this week. It should be starting to sink in. 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 should definitely fully understand this document with respect to protocols (in previous weeks this was mentioned here as “optional to know” but not anymore).

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.