Kenneth Allen
10-31-22
Reference vs. Value

What is a reference type versus a value type and how do they differ in terms of swift coding? Swift uses two “type” categories, one being a reference type, which shares the same type of data between the original and the new instance. Whereas with a value type, the original is copied and can hold its own unique information per copy.

In a value type, when you make changes to one copy, or the original type, it doesn’t change the new copied type. One main reason people choose to use value types vs reference types is that you can make alterations to one part of your copy without changing your entire code. You will always get a unique copied instance.

In a reference type all changes made to the original type will reflect into the called reference type as well. This is not exactly a “copy”. With reference types you are just referencing the original instance over and over throughout the code. The only way to make mutations is to change one reference type and ALL other instances will then also reflect these changes. **“**When one variable is assigned to another, the object is. not copied; both variables refer to the same object. — Modifying the value of one variable will affect others.”

In objective C, compound data types are represented using “classes”, which are handled using reference semantics. Meaning when an instance is assigned or passed as a parameter to a function, the memory it is assigned to will be represented and can be changed or mutated by operations done to any other references in the instance.

In swift, “structs” , “enums” , and “tuples” are all value types while “classes” and “closures” are reference types(just like in Objective C). Other fundamental types, such as ints and floats are considered value types. Advanced language types such as string, array, and other collection types are “structs”, not classes.