"Value Types" and "Reference Types" are the two possible object Types available in Swift. It is important to know the behavior of the data in our programs and how to choose the correct Type depending on our needs. When using Value types, each object of the Type keeps its unique data whenever a new variable is assigned while Reference Types' instances share a single copy of the data. When the original is modified, so is the new object referring to it. Structures, enumerations, and tuples are common value types in Swift, and on the other hand, classes, arrays, and dictionaries are common reference types. Copying a value type therefore results in an *independent instance* with its own copy of the data. If you find yourself needing to compare instance data using the == (equal-to) operator, you want copies to have independent states, or the data will be processed by multiple threads, you should be using a value type. Reference types, therefore, are used when you need two separate variables to affect a single shared instance because when you copy a reference type, the two variables now refer to the same instance of data. You should be using a reference type in Swift when you need to use the === (instance identity) operator, which checks if two references point to the same object instance. Once last reason to consider using value types over reference types is because they allow you to reason through your code more easily when trying to debug. Value types always give you a unique instance of data to work with and will not be affecting any part of your app "under the hood". It is important to note that in

Objective-C there is only value types. Reference types can be imitated by creating a pointer variable, with the nuance that its true value will not actually be the same value of the original instance of data, but that of a pointer. Everything in Objective-C is value type.

