Compare Dictionaries with a Value of Any type (Swift)

Comparing a Type of Any? No problem!

Steven Curtis Apr 15, 2020 · 3 min read ★

Photo by Pisit Heng on Unsplash

Difficulty: Beginner | Easy | **Normal** | Challenging

• • •

Whilst going through Apple's guide "Storing Keys in the Keychain" I noticed one thing.

The queries are of type **[String: Any]** , so I want to test the creation of the query.

I'll need to test the keys and values of type String and Any are created correctly, respectively.

I'll need to compare Any value types. :(

. . .

Prerequisites:

- Be able to produce a "Hello, World!" iOS application (guide HERE)
- Knowledge of Swift's Equatable protocol (guide HERE)
- Knowledge of Swift's Hashable protocol (guide HERE)

Some knowledge of:

- Value and reference types (guide HERE)

. . .

Terminology

Any: An instance of any type, including function types

Dictionary: The association between keys and values (where all the keys are of the same type, and all values are of the same type)

Equatable: A protocol that can be applied to a type to allow value equality

Key-value pairs: A set of two linked data items, a unique identifier (key) and an item of data (value)

. . .

Comparing Dictionaries

Creating a test to test the elements of a `Dictionary` is ordinarily quite easy. We can simply use the equality operator (==) as long as the type is `Equatable` (that is, the value conforms to the equatable protocol)

A Simple Comparison

If you wish to compare a `dictionary` with **[String: String]** type (or any value types).

```
let actual: [String: String] = ["id": "12345", "name": "James"]
```

```
let actual: [String: String] = ["id": "12345", "name": "James"]
var expected: [String: String] = ["id": "12345", "name": "James"]
actual == expected // true
```

SimpleDictionaryComparison hosted with ♥ by GitHub view raw

If the actual and expected `dictionaries` are different (for either the key or the value), the equality (==) will return false. If they are the same, the equality will return true.

The issue

Within guide there is a section to “Create a Query Dictionary” **addquery** is setup as follows:

```
let key = "AKey"
let tag = "com.example.keys.mykey".data(using: .utf8)!
let addquery: [String: Any] = [kSecClass as String: kSecClassKey,
                               kSecAttrApplicationTag as String: tag,
                               kSecValueRef as String: key]
```

so I wish to calculate if this is equal to another `dictionary`. However the type is **[String: Any]** and `Any` does *not* conform to `Equatable`.

That means even comparing the `Dictionary` to itelf:

```
addquery == addquery
```

will generate an error like the following

```
addquery == addquery    2 ⓘ '<Self where Self : Equatable> (Self.Type) -> (Self, Self) -> Bool' requires that 'Any' conform to 'Equatable'
```

Oh Dear!

The Solution

We can extend the `equality` operator and cast the Swift type `Dictionary` to the Objective-C type NSDictionary where the values are Hashable.

```swift
public func ==(lhs: [String: Any], rhs: [String: Any] ) -> Bool {
    return NSDictionary(dictionary: lhs).isEqual(to: rhs)
}
```

comparisonequality.swift hosted with ❤ by GitHub view raw

Now by using this function we can compare the `dictionaries`! Take a look at the full Playground as attached here

```swift
let key = "AKey"
let keyTwo = "AKeys"
let tag = "com.example.keys.mykey".data(using: .utf8)!
let addquery: [String: Any] = [kSecClass as String: kSecClassKey,
                               kSecAttrApplicationTag as String: tag,
                               kSecValueRef as String: key]


let comparisonDictionary: [String: Any] = [kSecClass as String: kSecClass
                                           kSecAttrApplicationTag as Stri
                                           kSecValueRef as String: keyTwo


public func ==(lhs: [String: Any], rhs: [String: Any] ) -> Bool {
    return NSDictionary(dictionary: lhs).isEqual(to: rhs)
}
```

Conclusion

The `Any` type does not conform to the `equality` protocol. However, we shouldn't let this stop us, and neither we will let it.

Making comparisons of `Dictionary` types is extremely important, and can even be used when we create some tests while coding in Swift.

. . .

Extend your knowledge

- Here is Apple's guide for implementing the Keychain HERE
- Swift have documentation about Dictionaries HERE

. . .

The Twitter contact:

Any questions? You can get in touch with me here