Innotech
MohammadHosein
March 17, 2018

Some Additional

in the last document there was some questions that I couldn't answer.

Why do we use throw instead of return?

We can return error and we can throw it but to handle error with try and catch; if you use return instead of throw you can't handle it.

* function which returns error, won't stop the program.

(returnError function won't stop the program)

```
enum err : Error{
    case errType
}

func returnError() -> Error {
    return err.errType
}

func throwError() throws {
    throw err.errType
}

do{
    try throwError()
}catch{
    print("error occured")
}

returnError()
/* do{
    try returnError()
}catch{
    print("error occured")
}*/
```

What is the difference between remove and removeValue in Dictionaries

In the remove function: Removes and returns the key-value pair at the specified index

In removeValue function: Removes by the key and returns the value that was removed; if there is no value nil will returned.

When to use guard and when if

Like an if statement, guard executes statements based on a Boolean value of an expression. Unlike an if statement, guard statements only run if the conditions are not met. You can think of guard more like an assert, but rather than crashing, you can gracefully exit.

The 'else' of guard must exit the current scope by calling return or throw.

so an example of using guard could be like verifying an input.

Access Control

https://medium.com/@abhimuralidharan/swift-3-0-1-access-control-9e71d641a56c

open and public

Enable an entity to be used out of a module like frameworks. Difference of open and public is in public mode subclassing and overriding function is not allowed.(in Extensions too.)

```
class A{
    public var publicName = "public name"
    open var openName = "open name"
}
extension A {
    func printNname() {
        print(openName)
     //   print(publicname)
    }
}
```

internal(default access level)

Internal classes and members can be accessed anywhere within the same module they are defined. Even if we import the module we cannot access internals in other modules.

fileprivate

Restricts the use of an entity in other files.

Example.swift:

```
fileprivate func printingError(){
        print("error …")
}
```

viewcontroller.swift:

```
let obj = Example()
obj.printingError()  //error
```

private

You typically use private access to hide the implementation details of a specific piece of functionality when those details are used only within a single declaration. If you declare a entity in class you can not use it out of that class.

```
class A{
    private var privateName = "private name"
}

var aa = A()
//aa.privateName
```