MÓDULO IOS

SWIFT 4

CLOSURES

Closures bloques autocontenidos de funcionalidades que pueden ser pasados y usados in nuestro código. Son similares a los bloques en objective C o lambdas en otro lenguajes de programación.

Los closures capturan y almacenan las constantes y variables del contexto en el cual fueron definidos.

Sintaxis:

```
{ (parameters) -> return type in
    statements
}
```

Closures y funciones son tipos por referencia. Si se asigna un closure a una constante o variable esta tiene la referencia de memoria donde se encuentra el closure.

@escaping (Escaping closures). Un closure escapa de una función cuando el closure es pasado como un argumento a la función, pero es llamado después.

Un closure se dice que escapa de una función cuando el closure se pasa como un argumento a la función, pero se llama después de que la función devuelve. Cuando se declara una función que toma un closure como uno de sus parámetros, puede escribir @escaping antes del tipo del parámetro para indicar que el closure se le permite escapar. Una forma que puede escapar un closure es almacenarse en una variable que se define fuera de la función. Como ejemplo, muchas funciones que inician una operación asincrónica toman un argumento de closure como un controlador de finalización. La función vuelve después de que se inicia la operación, pero el closure no se llama hasta que la operación se completa-el closure tiene que escapar, para ser llamado más tarde.

EJEMPLO NONESCAPING - ESCAPING

```
class ClassA {
  // takes a closure (non-escaping by default)
  func someMethod(closure: () -> Void) {
    // secret stuff
  }
}

class ClassB {
  let classA = ClassA()
  var someProperty = "Hello"

  func testClosure() {
    classA.someMethod {
      // self is captured!
      someProperty = "Inside the closure!"
    }
  }
}
```

```
func someMethod(closure: @escaping () -> Void) {
  // secret stuff
}
```

@non-escaping. El closure esta aun capturando a self, pero como el no vivirá cuando se termine la función entonces no hay riesgo de retain circle.

@escaping. Tenemos que especificar dentro del método a self si nos queremos referir a una propiedad de la claseB.