iOS Avanzado

KEEPCODING
Tech School

Temario

- Closures
- Genéricos
- Grand Central Dispatch

Closures

Son funciones anónimas que se pueden guardar en una variable.

Además, pueden pasarse como parámetro de otra función, utilizarse como tipo de retorno o añadirlos a una colección.

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

¿Los Closure son funciones?

En realidad, una función en Swift es un closure pero con azúcar sintáctico (o syntatic lugar, en inglés). Es decir, con una sintaxis más amigable y bonita para el desarrollador.

```
func add42(_ number: Int) -> Int {
    return number + 42
}

let add42Closure = { (number: Int) -> Int in
    return number + 42
}

add42(8) // 50
add42Closure(8) // 50
```

Tipos de Closures

- Una función que acepta un parámetro de tipo entero, y devuelve un entero.
- Una función que no acepta parámetros y devuelve una cadena.
- Una función que acepta un primer parámetro de tipo cadena y un segundo de tipo entero, y no retorna nada.

Closures como parámetro de entrada de una función

```
func add42(_ number: Int) -> Int {
    return number + 42
}

func add10(_ number: Int) -> Int {
    return number + 10
}

func apply(
    _ function: (Int) -> Int, // <-- Closure como parámetro
    with number: Int
) -> Int {

    return function(number)
}

// 1
apply(add42, with: 8) // 50

// 2
apply(add10, with: 8) // 18
```

Sintaxis abreviadas de los closures

```
let functionList = [
    add42, // En este momento, Swift ya sabe que es un array [(Int) -> Int]
    { (number: Int) -> Int in return number +  42 }, // 1
    { (number: Int) in return number * 2 }, // 2
    { number in return number + 8 } // 3
    { number in number + 42 } // 4
    { $0 + 100 } // 5
]
```

¿Sabes cómo usar el $0, $1 dentro de un Closure?

@escaping vs @nonescaping

- Los closures @nonescaping son aquellos que se ejecutan inmediatamente dentro del ámbito (o scope, en inglés) de la función.
- Los closures @escaping son aquellos que pueden ser ejecutados o llamados más tarde, después de que la función que los contiene haya terminado de ejecutarse.

@escaping vs @nonescaping

```
func apply(_ function: (Int) -> Void, to number: Int) {
    function(number)
}

print("Begin")
apply(add42, to:8)
print("End")
```

```
func applyInBackground(_ function: @escaping (Int) -> Void, with number: Int) {
    DispatchQueue.main.asyncAfter(deadline: .now() + 5.0) {
        function(number)
    }
}

print("Begin")
applyInBackground(add42, with: 8)
print("End")
```

KEEPCODING

Tech School

Madrid | Barcelona | Bogotá

Datos de contacto