Turn To Tech

Learn. Develop. Deliver

iOS development using Swift

Closures, Swift Standard Library, Sorting

1. Closures

1. Related terms: Functional programming, Lambda Calculus

2. JavaScript supports closures. Most other languages also support closures.

1.1 Functions as closures

```
func functionWithACounter() -> (() -> ()) {
    var counter = 1

    func implementSomeAlternatingBehavior() {

        var s:Bool
        s = false
        if counter % 2 == 0 {
            println("This function was called an EVEN number of times")
        } else {
            println("This function was called an ODD number of times")
        }
        counter++
    }
    return implementSomeAlternatingBehavior
}
```

1.1. Let's run the code

1.2. Closure terminology

1. Context
2. 'Closing over' of variables
3. Nested functions
4. 'Self-contained' blocks of code
5. Blocks – in the Objective-C world

Closing over of variables

- The var counter is available inside the nested function

Return type

Nested function as a closure

```
func functionWithACounter() -> (() -> ()) {
    var counter = 1

    func implementSomeAlternatingBehavior() {

        var s:Bool
        s = false
        if counter % 2 == 0 {
            println("This function was called an EVEN number of times")
        } else {
            println("This function was called an ODD number of times")
        }
        counter++
    }
    return implementSomeAlternatingBehavior
}
```

Context

Return a reference to the closure

1.3 Regular closures

Without using functions

```
//----- a function to describe the sort -----


func sortPrefFunction(s1:String, s2:String) -> Bool {
    if s1 > s2 {
        return true;
    } else {
        return false;
    }
}

//------- a closure to describe the sort --------

var sortPrefClosure = {
    (s1:String, s2:String) -> Bool in
    if s1 > s2 {
        return true;
    } else {
        return false;
    }
}
```

1.4 Closures in practice

1. Closures are passed as args to functions
2. Closures can be declared on the fly
3. Conventions exist when closures are the last arg
4. Special notation for closures – to reduce code clutter

1.4.1 Closures can be passed as args to functions

```
var sortPrefClosureInline = {
    (s1:String, s2:String) -> Bool in
    if s1 > s2 {
        return true;
    } else {
        return false;
    }
}
var sortedArray = sort(unsortedArray, sortPrefClosureInline)

println(sortedArray)
```

1.4.2 Closures can be declared on the fly

```
var sortedArray = sort(unsortedArray,{
    (s1:String, s2:String) -> Bool in
    if s1 > s2 {
        return true;
    } else {
        return false;
    }
    })

    println(sortedArray)

}
```

1.4.3 When a closure is the last argument

```
var sortedArray = sort(unsortedArray){
    (s1:String, s2:String) -> Bool in
    if s1 > s2 {
        return true;
    } else {
        return false;
    }
    }

println(sortedArray)
```

1.4.4 Special closure notation

```
var sortedArray = sort(unsortedArray){ $0 > $1 }

println(sortedArray)
```

2. Swift Standard Lib

1. String
2. Array
3. Dictionary
4. Numeric types
5. Protocols - Equatable and Comparable
6. Algorithms – Sort, Reverse, Map, Reduce

3. Practical examples

- Map
- Sorting – simple array
- Sorting – object array
- Asynchronous network calls

3.1 Map algorithm

3.2 Sorting a simple array

3.3 Sorting an object array

3.4 Asynch network calls

4. Assignment