Swift's Higher Order Functions

Practical use of { map, filter and reduce }

Created by Vina Rianti (@vinamelody)

- It's weekend at GeekcampID, relax !
- Lots of Swift code for , but fun!
- No such thing as a stupid question, ask !

Swift Language

Swift is a general-purpose programming language built using a modern approach to safety, performance, and software design patterns. (https://swift.org/about/)

One example of software design patterns is the use of closures.

Functions

- Function name
- Parameter variable & type
- Return type
- Code block / implementation

```
func greet(name: String?) -> String {
    let greeting = "Hello, " + (name == nil ? "there" : na
    return greeting
}
```

To call the function

```
greet(name: "John Appleseed")
```

Swift Closures

- A closure is a self-contained block of code that can be passed around.
- Declare it like a variable

```
var sayHelloClosure: (_ name: String?) -> String = { (nam
    return greet(name: name)
}
```

To call the closure

```
sayHelloClosure("Vina")
sayHelloClosure(nil)
```

Map

Given an array of customers.

```
let customers = ["Budi", "Joni", "Dian"]
```

How to create a new array of greetings?

The usual way is ...

```
func greet(name: String?) -> String {
    let greeting = "Hello, " + (name == nil ? "there" : na
    return greeting
}
var greetings = [String]()
for customer in customers {
    greetings.append(greet(name: customer))
}
```

Map

What's a better way to do this using closure?

```
let smartGreetings = customers.map({ (name) in
    return greet(name: name)
})
```

Even shorter...

```
let smarterGreetings = customers.map({ return greet(name:
```

Filter

Given an array of customers with status data

```
let customersStatusData: [[String: String]] = [
    ["name": "Karmen", "status": "Regular"],
    ["name": "Joni", "status": "Premium"],
    ["name": "Dian", "status" : "Regular"],
    ["name": "Rangga", "status": "Regular"]]
```

How to create a new set array of customers who are Regular?

```
func filterRegular(customers: [[String:String]]) -> [[Str
    var regularCustomers: [[String:String]] = [[String:St
    for customer in customers {
        if (customer["status"]?.contains("Regular"))! {
            regularCustomers.append(customer)
        }
    }
    return regularCustomers
}
let regularFilter = filterRegular(customers: customersSta
```

Filter

What's a better way to do this using closure?

```
let smartRegularFilter = customersStatusData.filter({
    customer in (customer["status"]?.contains("Premium"))
})
```

Even shorter...

```
let smarterRegularFilter = customersStatusData.filter({
    return ($0["status"]?.contains("Regular"))!
})
```

Reduce

One last thing.

```
let customersData: [[String: Any]] = [
    ["name": "Karmen", "status": "Regular", "totalSpend":
    ["name": "Joni", "status": "Premium", "totalSpend": 2
    ["name": "Dian", "status" : "Regular", "totalSpend":
    ["name": "Rangga", "status": "Regular", "totalSpend":
```

How to get total spending of all customers?

```
func sumTotalSpend(customers: [[String:Any]]) -> Double {
    var total: Double = 0
    for customer in customers {
        if let spending = customer["totalSpend"] as? Doub
            total += spending
        }
    }
    return total
}
let totalSpend = sumTotalSpend(customers: customersData)
```

Reduce

What's a better way to do this using closure?

```
let smartReduce = customersData.map({ return $0["totalSpe
```

Challenge

Up for challenge? We'll use method chaining to find out how much is the total spending for regular customers.

Summary

- You can achieve what you want with or without closure
- However, closure makes you code cleaner and more readable
- There are other benefits of closure that we can explore beyond this talk

Thank you!