QUICK TIPS : iOS

WHAT ARE
HIGHER ORDER FUNCTIONS IN SWIFT?

Rohit Saini
iOS Developer

QUICK TIPS : iOS

Higher Order Functions:

- Higher order functions are simply functions that can either accept functions or closures as arguments, or return a function/closure.
- Higher order functions in Swift are extremely powerful tool to have in your developer toolkit.

Useful Higher Order Functions:

- Map
- Filter
- Reduce

QUICK TIPS : iOS

- Sorted
- CompactMap

Map:

- Map is used when you want to apply the same operation to each element of a collection.
- It takes a single argument in the form of a mapping closure and returns an array with the transformed elements of the input sequence.

```
// Example: Convert Meters to Feet
let meters = [10.0, 22.0, 55.0, 74.0]

let feet = meters.map { $0 * 3.281}
print("Meters converted to feet: \(feet)")
//-> Meters converted to feet: [32.81,72.182,180.455,242.794]
```

QUICK TIPS : iOS

Filter:

Filter is used when you want to have a result with only elements that match a condition.

```
// Example : Filter only the planets that start with the letter "M"
let filteredPlanetNames = planetNames.filter {$0.prefix(1).uppercased() == “M"}

print("Count of filtered planet names: \(filteredPlanetNames.count)”)
//-> Count of filtered planet names: 2
```

Reduce:

Reduce is used when you want to combine all elements in a collection into one value.

```
// Example : Sum of numbers
let numbers = [5, 3, 2, 6, 10, 23, 01, 43, 5, 7, 8, 9]

let sumOfNumbers = numbers.reduce(0, {$0 + $1})print("Sum of numbers: \
(sumOfNumbers)”)
//-> Sum of numbers : 122
```

QUICK TIPS : iOS

Sorted:

- When calling sorted() on an array, it will return a new array that has the items sorted in ascending order.

```
// Example : Sorting numbers ascending
let sortedNumbersAscending = numbers.sorted()
print("Sorted numbers ascending: \(sortedNumbersAscending)")

//—> Sorted numbers ascending: [1, 2, 3, 5, 5, 6, 7, 8, 9, 10, 23, 43]
```

CompactMap:

- CompactMap Filter out nil-values from a sequence.
- CompactMap is the same as the Map function but with optional handling capability.

```
//Example
let numbers = ["1", "2", "Fish"]
let integers = numbers.compactMap { Int($0) }
print(integers)
```

QUICK TIPS : iOS

```
//Fish cannot be converted Int value so compactMap unwrap the optional value and
remove nil value from collection.
//[1,2]
```

Like, Share, Enjoy!

Rohit Saini
iOS Developer