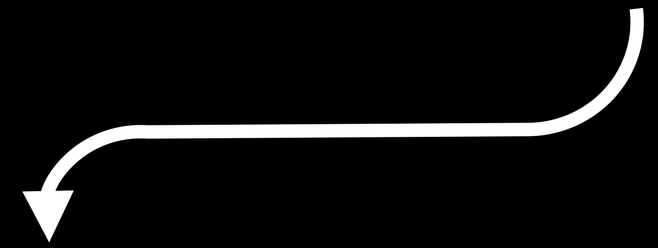
Unit 5—Lesson 1: Closures

Closures

```
(firstTrack: Track, secondTrack: Track) -> Bool in
 return firstTrack.trackNumber < secondTrack.trackNumber
```



```
let sortedTracks = tracks.sorted (   )
```

Syntax

```
func sum(numbers: [Int]) -> Int {
  // Code that adds together the numbers array
  return total
}
```

```
 let sumClosure = { (numbers: [Int]) -> Int in
   // Code that adds together the numbers array
   return total
 }
```

```swift
// A closure with no parameters and no return value
let printClosure = { () -> Void in
  print("This closure does not take any parameters and does not return a value.")
}

// A closure with parameters and no return value
let printClosure = { (string: String) -> Void in
  print(string)
}

// A closure with no parameters and a return value
let randomNumberClosure = { () -> Int in
  // Code that returns a random number
}

// A closure with parameters and a return value
let randomNumberClosure = { (minValue: Int, maxValue: Int) -> Int in
  // Code that returns a random number between `minValue` and `maxValue`
}
```

Passing closures as arguments

```
let sortedTracks = tracks.sorted { (firstTrack: Track, secondTrack: Track) -> Bool in
  return firstTrack.trackNumber < secondTrack.trackNumber
}
```

```
 let sortedTracks = tracks.sorted { (firstTrack: Track, secondTrack: Track) -> Bool in
   return firstTrack.starRating < secondTrack.starRating
 }
```

Syntactic sugar

```
let sortedTracks = tracks.sorted { (firstTrack: Track, secondTrack: Track) -> Bool in
  return firstTrack.starRating < secondTrack.starRating
}
```

Syntactic sugar

```
let sortedTracks = tracks.sorted { (firstTrack, secondTrack) -> Bool in
  return firstTrack.starRating < secondTrack.starRating
}
```

Syntactic sugar

```
let sortedTracks = tracks.sorted { (firstTrack, secondTrack) in
  return firstTrack.starRating < secondTrack.starRating
}
```

Syntactic sugar

```
let sortedTracks = tracks.sorted { return $0.starRating < $1.starRating }
```

Syntactic sugar

```
let sortedTracks = tracks.sorted { $0.starRating < $1.starRating }
```

Collection functions using closures

Map

Filter

Reduce

Collection functions using closures

```
// Initial array
let firstNames = ["Johnny", "Nellie", "Aaron", "Rachel"]

// Creates an empty array that will be used
// to store the full names
var fullNames: [String] = []

for name in firstNames {
    let fullName = name + " Smith"
    fullNames.append(fullName)
}
```

fullNames

0	"Johnny Smith"
1	"Nellie Smith"
2	"Aaron Smith"
3	"Rachel Smith"

Collection functions using closures

```
// Initial array
let firstNames = ["Johnny", "Nellie", "Aaron", "Rachel"]

// Creates a new array of full names by adding "Smith"
// to each first name
let fullNames = firstNames.map { (name) -> String in
    return name + " Smith"
}
```

fullNames

0	"Johnny Smith"
1	"Nellie Smith"
2	"Aaron Smith"
3	"Rachel Smith"

Collection functions using closures

```
// Initial array
let firstNames = ["Johnny", "Nellie", "Aaron", "Rachel"]

// Creates a new array of full names by adding "Smith"
// to each first name
let fullNames = firstNames.map{ $0 + " Smith" }
```

`fullNames`

0	"Johnny Smith"
1	"Nellie Smith"
2	"Aaron Smith"
3	"Rachel Smith"

Collection functions using closures

```
let numbers = [4, 8, 15, 16, 23, 42]

var numbersLessThan20: [Int] = []

for number in numbers {
    if number < 20 {
        numbersLessThan20.append(number)
    }
}
```

numbersLessThan20

0	4
1	8
2	15
3	16

Collection functions using closures

```
let numbers = [4, 8, 15, 16, 23, 42]

let numbersLessThan20 = numbers.filter { (number) -> Bool in

    return number < 20
}
```

numbersLessThan20

0	4
1	8
2	15
3	16

Collection functions using closures

```
let numbers = [4, 8, 15, 16, 23, 42]

let numbersLessThan20 = numbers.filter{ $0 < 20 }
```

numbersLessThan20

0	4
1	8
2	15
3	16

Collection functions using closures

```
let numbers = [8, 6, 7, 5, 3, 0, 9]

var total = 0

for number in numbers {
    total = total + number
}
```

Collection functions using closures

```
let numbers = [8, 6, 7, 5, 3, 0, 9]

let total = numbers.reduce(0) { (currentTotal, newValue) -> Int in
    return currentTotal + newValue
}
```

Collection functions using closures

```
let numbers = [8, 6, 7, 5, 3, 0, 9]

let total = numbers.reduce(0,{ $0 + $1})
```

Closures capture their environment

```
animate {
  self.view.backgroundColor = .red
}
```

Unit 5—Lesson 1
Lab: Closures

Open and complete the exercises in Lab - Closures.playground