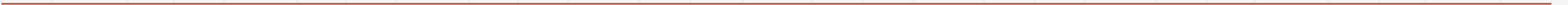


Agenda

- Collections
- Control Flow and Functions



Collections

- What an array is in Swift and how to use it
 - What a dictionary is in Swift and how we can use it
 - What a set is in Swift and how we can use it
 - What a tuple is in Swift and how we can use it
-

Mutability

- Define with **let** for immutable collection (faster)
 - Define with **var** for mutable collection
-

Creating Arrays

```
let arrayOne = [1,2,3]
```

```
var arrayTwo = [4,5,6]
```

```
var arrayThree = [Int]()
```

Creating Arrays

```
var arrayOne = [String]()
```

```
var arrayTwo = [Double]()
```

```
var arrayThree = [MyObject]()
```

Creating Arrays

```
var myArray: [Any] = [1, "Two"]
```



Creating Arrays

```
var arrayFour = [Int](repeating: 3, count: 7)
```



Creating Arrays

```
var multiArrayOne = [[1,2],[3,4],[5,6]]
```

```
var multiArrayTwo = [[Int]]()
```

Accessing Array Elements

```
let arrayOne = [1,2,3,4,5,6]
```

```
print(arrayOne[0]) //Displays '1'
```

```
print(arrayOne[3]) //Displays '4'
```

Accessing Array Elements

```
var multiArray = [[1,2],[3,4],[5,6]]
```

```
var arr = multiArray[0] //arr contains the array [1,2]
```

```
var value = multiArray[0][1] //value contains 2
```

Accessing Array Elements

```
let arrayOne = [1,2,3,4,5,6]
var first = arrayOne.first      //first contains 1
var last = arrayOne.last       //last contains 6
```

```
let multiArray = [[1,2],[3,4],[5,6]]
var arrFirst1 = multiArray[0].first //arrFirst1 contains 1
var arrFirst2 = multiArray.first    //arrFirst2 contains [1,2]
var arrLast1 = multiArray[0].last   //arrLast1 contains 2
var arrLast2 = multiArray.last      //arrLast2 contains [5,6]
```

Counting Array Elements

```
let arrayOne = [1,2,3]  
let multiArrayOne = [[3,4],[5,6],[7,8]]
```

```
print(arrayOne.count)           //Displays 3  
print(multiArrayOne.count)      //Displays 3 for the three arrays  
print(multiArrayOne[0].count)   //Displays 2 for the two elements
```

Counting Array Elements

```
let arrayOne = [0,1]  
print(arrayOne[0]) //Displays 0  
print(arrayOne[1]) //Displays 1  
print(arrayOne.count) //Displays 2
```

Counting Array Elements

```
//This example will throw an array index out of range error var  
arrayTwo = [1,2,3,4]  
print(arrayTwo[6])
```

```
//This example will not throw an array index out of range error  
var arrayOne = [1,2,3,4]  
if (arrayOne.count > 6) {  
    print(arrayOne[6])  
}
```

Empty Array

```
var arrayOne = [1,2]  
var arrayTwo = [Int]()
```

```
arrayOne.isEmpty //Returns false because the array is not empty  
arrayTwo.isEmpty //Returns true because the array is empty
```

Appending to an Array

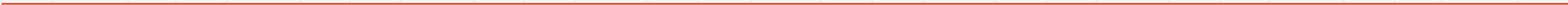
```
var arrayOne = [1,2]
```

```
arrayOne.append(3) //arrayOne will now contain 1, 2, and 3
```

Appending to an Array

```
var arrayOne = [1,2]
```

```
arrayOne += [3,4] //arrayOne will now contain 1, 2, 3, and 4
```



Inserting value into an Array

```
var arrayOne = [1,2,3,4,5]
```

//arrayOne now contains 1, 2, 3, 10, 4, and 5

```
arrayOne.insert(10, at: 3)
```

Replacing elements in an Array

```
var arrayOne = [1,2,3]
```

```
arrayOne[1] = 10           //arrayOne now contains 1, 10, 3
```

Removing elements from an Array

```
var arrayOne = [1,2,3,4,5]
```

```
arrayOne.removeLast() //arrayOne now contains 1, 2, 3, and 4
```

```
arrayOne.remove(at:2) //arrayOne now contains 1, 2, and 4
```

```
arrayOne.removeAll() //arrayOne is now empty
```

Adding two Arrays

```
let arrayOne = [1,2]
```

```
let arrayTwo = [3,4]
```

```
//combine contains 1, 2, 3, and 4
```

```
var combine = arrayOne + arrayTwo
```

Reversing an array

```
var arrayOne = [1,2,3]
```

```
//reverse contains 3, 2, and 1  
var reverse = arrayOne.reversed()
```

Retrieving a subarray from an array

```
let arrayOne = [1,2,3,4,5]
```

```
var subArray = arrayOne[2...4] //subArray contains 3, 4, and 5
```

Retrieving a subarray from an array

```
let arrayOne = [1,2,3,4,5]
```

```
var subArray = arrayOne[2..<4] //subArray contains 3 and 4
```

Making bulk changes to an array

```
var arrayOne = [1,2,3,4,5]
```

//arrayOne contains 1, 12, 13, 4, and 5

```
arrayOne[1...2] = [12,13]
```

Making bulk changes to an array

```
var arrayOne = [1,2,3,4,5]
```

//arrayOne now contains 1, 12, 13 and 5 (four elements)

```
arrayOne[1...3] = [12,13]
```

Making bulk changes to an array

```
var arrayOne = [1,2,3,4,5]
```

```
//arrayOne now contains 1, 12, 13, 14, 15 and 5 (six elements)
```

```
arrayOne[1...3] = [12,13,14,15]
```

SORT an array

```
var arrayOne = [9,3,6,2,8,5]
```

```
//arrayOne contains 2, 3, 5, 6, 8, and 9
```

```
arrayOne.sort(){ $0 < $1 }
```

SORT an array in reverse order

```
var arrayOne = [9,3,6,2,8,5]
```

```
//arrayOne contains 9,8,6,5,3 and 2
```

```
arrayOne.sort(){ $1 < $0 }
```

SORTED

```
var arrayOne = [9,3,6,2,8,5]
```

```
//sorted contains 2,3,5,6,8 and 9
```

```
//arrayOne contains 9,3,6,2,8 and 5
```

```
let sorted = arrayOne.sorted(){ $0 < $1 }
```

FILTER

```
var arrayOne = [1,2,3,4,5,6,7,8,9]
```

```
//filtered contains 4, 5, and 6
```

```
let filtered = arrayOne.filter{$0 > 3 && $0 < 7}
```

FILTER

```
var city = ["Boston", "London", "Chicago", "Atlanta"]
```

```
//filtered contains "Boston", "London" and "Chicago"  
let filtered = city.filter{$0.range(of:"o") != nil}
```

MAP

```
var arrayOne = [10, 20, 30, 40]
```

```
//applied contains 1,2,3 and 4  
let applied = arrayOne.map{ $0 / 10}
```

MAP

```
var arrayOne = [1, 2, 3, 4]
```

```
//applied contains "num:1", "num:2", "num:3" and "num:4"  
let applied = arrayOne.map{ "num:\($0)"}
```

FOREACH

```
var arrayOne = [10, 20, 30, 40]  
arrayOne.forEach{ print($0) }
```

10
20
30
40

Iterating over an array

```
var arr = ["one", "two", "three"]  
for item in arr {  
    print(item)  
}
```

one
two
three

Iterating over an array

```
var arr = ["one", "two", "three"]  
for (index,value) in arr.enumerated() {  
    print("\(index) \(value)")  
}
```

```
0 one  
1 two  
2 three
```

DICTIONARIES

Key	Value
US	United States
IN	India
UK	United Kingdom

Creating and initializing dictionaries

```
let countries = ["US":"UnitedStates","IN":"India","UK":"United Kingdom"]
```

```
var countries = ["US":"UnitedStates","IN":"India","UK":"United Kingdom"]
```

Creating empty dictionaries

```
var dic1 = [String:String]()
```

```
var dic2 = [Int:String]()
```

```
var dic3 = [String:MyObject]()
```

Accessing dictionary values

```
let countries = ["US":"United States", "IN":"India", "UK":"United Kingdom"]  
var name = countries["US"]
```

Counting dictionary values

```
let countries = ["US":"United States", "IN":"India","UK":"United Kingdom"];  
var cnt = countries.count //cnt contains 3
```



Is the dictionary empty?

```
let countries = ["US":"United States", "IN":"India", "UK":"United Kingdom"]  
var empty = countries.isEmpty
```

Updating key value

```
var countries = ["US":"United States", "IN":"India","UK":"United Kingdom"]
```

```
//The value of UK is now set to "Great Britain"
```

```
countries["UK"] = "Great Britain"
```

```
//The value of UK is now set to "Britain" and orig now contains "Great Britain"
```

```
var orig = countries.updateValue("Britain", forKey: "UK")
```

Adding key-value pair

```
var countries = ["US":"United States", "IN":"India","UK":"United Kingdom"]
```

```
//The value of "FR" is set to "France"  
countries["FR"] = "France"
```

```
//The value of "DE" is set to "Germany" and orig is nil  
var orig = countries.updateValue("Germany", forKey: "DE")
```

Removing key-value pair

```
var countries = ["US":"United States", "IN":"India","UK":"United Kingdom"];
```

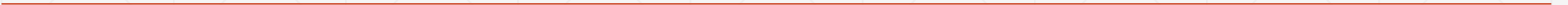
```
//The "IN" key/value pair is removed  
countries["IN"] = nil
```

```
//The "UK" key value pair is removed and orig contains "United Kingdom"  
var orig = countries.removeValue(forKey:"UK")
```

```
//Removes all key/value pairs from the countries dictionary  
countries.removeAll()
```

SET

```
var mySet = Set(["one", "two", "three"])
```



Initializing a set

```
//Initializes an empty Set of the String type  
var mySet = Set<String>()
```

```
//Initializes a mutable set of the String type with initial values  
var mySet = Set(["one", "two", "three"])
```

```
//Creates an immutable set of the String type.  
let mySet = Set(["one", "two", "three"])
```

Inserting items into a set

```
var mySet = Set<String>()
```

```
mySet.insert("One")  
mySet.insert("Two")  
mySet.insert("Three")
```

The number of items in a set

```
var mySet = Set<String>()  
mySet.insert("One")  
mySet.insert("Two")  
mySet.insert("Three")  
print("\n(mySet.count) items")
```

Checking whether a set contains an item

```
var mySet = Set<String>()
```

```
mySet.insert("One")
```

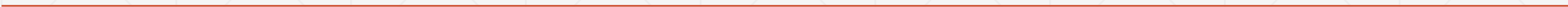
```
mySet.insert("Two")
```

```
mySet.insert("Three")
```

```
var contain = mySet.contains("Two")
```

Iterating over a set

```
for item in mySet {  
    print(item)  
}
```



Removing items in a set

//The remove method will return and remove an item from a set
`var item = mySet.remove("Two")`

//The removeAll method will remove all items from a set
`mySet.removeAll()`

Set Operations

- `union` and `fromUnion`: These create a set with all the unique values from both sets
 - `subtracting` and `subtract`: These create a set with values from the first set that are not in the second set
 - `intersection` and `fromIntersection`: These create a set with values that are common to both sets
 - `symmetricDifference` and `fromSymmetricDifference`: These create a new set with values that are in either set but not in both sets
 -
-

Union

```
var mySet1 = Set(["One", "Two", "Three", "abc"])  
var mySet2 = Set(["abc","def","ghi", "One"])  
var newSetUnion = mySet1.union(mySet2)
```

fromUnion

```
var mySet1 = Set(["One", "Two", "Three", "abc"])  
var mySet2 = Set(["abc", "def", "ghi", "One"])  
mySet1.fromUnion(mySet2)
```

subtract

```
var mySet1 = Set(["One", "Two", "Three", "abc"])
```

```
var mySet2 = Set(["abc","def","ghi", "One"])
```

```
var newSetSubtract = mySet1.subtracting(mySet2)
```

```
mySet1.subtract(mySet2)
```

intersection

```
var mySet1 = Set(["One", "Two", "Three", "abc"])
var mySet2 = Set(["abc","def","ghi", "One"])
var newSetIntersect = mySet1.intersection(mySet2)
mySet1.fromIntersection(mySet2)
```

symmetricDifference

```
var mySet1 = Set(["One", "Two", "Three", "abc"])
```

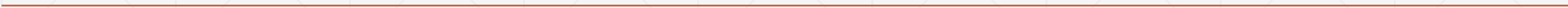
```
var mySet2 = Set(["abc","def","ghi", "One"])
```

```
var newSetExclusiveOr = mySet1.symmetricDifference(mySet2)
```

```
mySet1.fromSymmetricDifference(mySet2)
```

TUPLES

```
var team = ("Boston", "Red Sox", 97, 65, 59.9)
```



TUPLES

```
var team = ("Boston", "Red Sox", 97, 65, 59.9)
```

```
var (city, name, wins, loses, percent) = team
```

Retrieving values from Tuples

```
var team = ("Boston", "Red Sox", 97, 65, 59.9)
```

```
var city = team.0
```

```
var name = team.1
```

```
var wins = team.2
```

```
var loses = team.3
```

```
var percent = team.4
```

Named Tuples

```
var team = (city:"Boston", name:"Red Sox", wins:97, loses:65,  
percent:59.9)
```

