Swift 数组 Swift 函数

Swift 字典

Swift 字典用来存储无序的相同类型数据的集合，Swift 字典会强制检测元素的类型，如果类型不同则会报错。

Swift 字典每个值（value）都关联唯一的键（key），键作为字典中的这个值数据的标识符。

和数组中的数据项不同，字典中的数据项并没有具体顺序。我们在需要通过标识符（键）访问数据的时候使用字典，这种方法很大程度上和我们在现实世界中使用字典查字义的方法一样。

Swift 字典的key没有类型限制可以是整型或字符串，但必须是唯一的。

如果创建一个字典，并赋值给一个变量，则创建的字典就是可以修改的。这意味着在创建字典后，可以通过添加、删除、修改的方式改变字典里的项目。如果将一个字典赋值给常量，字典就不可修改，并且字典的大小和内容都不可以修改。

创建字典

我们可以使用以下语法来创建一个特定类型的空字典：

```
var someDict =  [KeyType: ValueType]()
```

以下是创建一个空字典，键的类型为 Int，值的类型为 String 的简单语法：

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

以下为创建一个字典的实例：

```
var someDict:[Int:String] = [1:"One", 2:"Two", 3:"Three"]
```

访问字典

我们可以根据字典的索引来访问数组的元素，语法如下：

```
var someVar = someDict[key]
```

我们可以通过以下实例来学习如何创建，初始化，访问字典:

```
import Cocoa

var someDict:[Int:String] = [1:"One", 2:"Two", 3:"Three"]

var someVar = someDict[1]

print( "key = 1 的值为 \(someVar)" )
```

```
print( "key = 2 的值为 \(someDict[2])" )
print( "key = 3 的值为 \(someDict[3])" )
```

以上程序执行输出结果为：

```
key = 1 的值为 Optional("One")
key = 2 的值为 Optional("Two")
key = 3 的值为 Optional("Three")
```

修改字典

我们可以使用 **updateValue(forKey:)** 增加或更新字典的内容。如果 key 不存在，则添加值，如果存在则修改 key 对应的值。updateValue(_:forKey:)方法返回Optional值。实例如下：

```
import Cocoa

var someDict:[Int:String] = [1:"One", 2:"Two", 3:"Three"]

var oldVal = someDict.updateValue("One 新的值", forKey: 1)

var someVar = someDict[1]

print( "key = 1 旧的值 \(oldVal)" )
print( "key = 1 的值为 \(someVar)" )
print( "key = 2 的值为 \(someDict[2])" )
print( "key = 3 的值为 \(someDict[3])" )
```

以上程序执行输出结果为：

```
key = 1 旧的值 Optional("One")
key = 1 的值为 Optional("One 新的值")
key = 2 的值为 Optional("Two")
key = 3 的值为 Optional("Three")
```

你也可以通过指定的 key 来修改字典的值，如下所示：

```
import Cocoa

var someDict:[Int:String] = [1:"One", 2:"Two", 3:"Three"]

var oldVal = someDict[1]
someDict[1] = "One 新的值"
var someVar = someDict[1]
```

```
print( "key = 1 旧的值 \(oldVal)" )
print( "key = 1 的值为 \(someVar)" )
print( "key = 2 的值为 \(someDict[2])" )
print( "key = 3 的值为 \(someDict[3])" )
```

以上程序执行输出结果为：

```
key = 1 旧的值 Optional("One")
key = 1 的值为 Optional("One 新的值")
key = 2 的值为 Optional("Two")
key = 3 的值为 Optional("Three")
```

移除 Key-Value 对

我们可以使用 **removeValueForKey()** 方法来移除字典 key-value 对。如果 key 存在该方法返回移除的值，如果不存在返回 nil 。实例如下：

```
import Cocoa

var someDict:[Int:String] = [1:"One", 2:"Two", 3:"Three"]

var removedValue = someDict.removeValue(forKey: 2)

print( "key = 1 的值为 \(someDict[1])" )
print( "key = 2 的值为 \(someDict[2])" )
print( "key = 3 的值为 \(someDict[3])" )
```

以上程序执行输出结果为：

```
key = 1 的值为 Optional("One")
key = 2 的值为 nil
key = 3 的值为 Optional("Three")
```

你也可以通过指定键的值为 nil 来移除 key-value（键-值）对。实例如下:

```
import Cocoa

var someDict:[Int:String] = [1:"One", 2:"Two", 3:"Three"]

someDict[2] = nil

print( "key = 1 的值为 \(someDict[1])" )
print( "key = 2 的值为 \(someDict[2])" )
print( "key = 3 的值为 \(someDict[3])" )
```

以上程序执行输出结果为：

```
key = 1 的值为 Optional("One")
key = 2 的值为 nil
key = 3 的值为 Optional("Three")
```

遍历字典

我们可以使用 **for-in** 循环来遍历某个字典中的键值对。实例如下:

```
import Cocoa

var someDict:[Int:String] = [1:"One", 2:"Two", 3:"Three"]

for (key, value) in someDict {
   print("字典 key \(key) -  字典 value \(value)")
}
```

以上程序执行输出结果为：

```
字典 key 2 -  字典 value Two
字典 key 3 -  字典 value Three
字典 key 1 -  字典 value One
```

我们也可以使用enumerate()方法来进行字典遍历，返回的是字典的索引及 (key, value) 对，实例如下:

```
import Cocoa

var someDict:[Int:String] = [1:"One", 2:"Two", 3:"Three"]

for (key, value) in someDict.enumerated() {
    print("字典 key \(key) -  字典 (key, value) 对 \(value)")
}
```

以上程序执行输出结果为：

```
字典 key 0 -  字典 (key, value) 对 (2, "Two")
字典 key 1 -  字典 (key, value) 对 (3, "Three")
字典 key 2 -  字典 (key, value) 对 (1, "One")
```

字典转换为数组

你可以提取字典的键值(key-value)对，并转换为独立的数组。实例如下：

```
import Cocoa

var someDict:[Int:String] = [1:"One", 2:"Two", 3:"Three"]

let dictKeys = [Int](someDict.keys)
let dictValues = [String](someDict.values)

print("输出字典的键(key)")

for (key) in dictKeys {
    print("\(key)")
}

print("输出字典的值(value)")

for (value) in dictValues {
    print("\(value)")
}
```

以上程序执行输出结果为：

```
输出字典的键(key)
2
3
1
输出字典的值(value)
Two
Three
One
```

count 属性

我们可以使用只读的 **count** 属性来计算字典有多少个键值对：

```
import Cocoa

var someDict1:[Int:String] = [1:"One", 2:"Two", 3:"Three"]
var someDict2:[Int:String] = [4:"Four", 5:"Five"]

print("someDict1 含有 \(someDict1.count) 个键值对")
print("someDict2 含有 \(someDict2.count) 个键值对")
```

以上程序执行输出结果为：

```
someDict1 含有 3 个键值对
someDict2 含有 2 个键值对
```

isEmpty 属性

Y我们可以通过只读属性 **isEmpty** 来判断字典是否为空，返回布尔值:

```
import Cocoa

var someDict1:[Int:String] = [1:"One", 2:"Two", 3:"Three"]
var someDict2:[Int:String] = [4:"Four", 5:"Five"]
var someDict3:[Int:String] = [Int:String]()

print("someDict1 = \(someDict1.isEmpty)")
print("someDict2 = \(someDict2.isEmpty)")
print("someDict3 = \(someDict3.isEmpty)")
```

以上程序执行输出结果为：

```
someDict1 = false
someDict2 = false
someDict3 = true
```

Swift 数组 Swift 函数