字面量

@M了个J

https://github.com/CoderMJLee

http://cnblogs.com/mjios

码拉松

字面量（Literal）

```
var age = 10
var isRed = false
var name = "Jack"
```

- 上面代码中的10、false、"Jack"就是字面量

- 常见字面量的默认类型
  - public typealias IntegerLiteralType = Int
  - public typealias FloatLiteralType = Double
  - public typealias BooleanLiteralType = Bool
  - public typealias StringLiteralType = String

```
// 可以通过typealias修改字面量的默认类型
typealias FloatLiteralType = Float
typealias IntegerLiteralType = UInt8
var age = 10 // UInt8
var height = 1.68 // Float
```

- Swift自带的绝大部分类型，都支持直接通过字面量进行初始化
  - Bool、Int、Float、Double、String、Array、Dictionary、Set、Optional等

字面量协议

- Swift自带类型之所以能够通过字面量初始化，是因为它们遵守了对应的协议
- Bool : ExpressibleByBooleanLiteral
- Int : ExpressibleByIntegerLiteral
- Float、Double : ExpressibleByIntegerLiteral、ExpressibleByFloatLiteral
- Dictionary : ExpressibleByDictionaryLiteral
- String : ExpressibleByStringLiteral
- Array、Set : ExpressibleByArrayLiteral
- Optional : ExpressibleByNilLiteral

```
var b: Bool = false // ExpressibleByBooleanLiteral
var i: Int = 10 // ExpressibleByIntegerLiteral
var f0: Float = 10 // ExpressibleByIntegerLiteral
var f1: Float = 10.0 // ExpressibleByFloatLiteral
var d0: Double = 10 // ExpressibleByIntegerLiteral
var d1: Double = 10.0 // ExpressibleByFloatLiteral
var s: String = "jack" // ExpressibleByStringLiteral
var arr: Array = [1, 2, 3] // ExpressibleByArrayLiteral
var set: Set = [1, 2, 3] // ExpressibleByArrayLiteral
var dict: Dictionary = ["jack" : 60] // ExpressibleByDictionaryLiteral
var o: Optional<Int> = nil // ExpressibleByNilLiteral
```

字面量协议应用

```
extension Int : ExpressibleByBooleanLiteral {
    public init(booleanLiteral value: Bool) { self = value ? 1 : 0 }
}
var num: Int = true
print(num) // 1
```

- 有点类似于C++中的转换构造函数

```
class Student : ExpressibleByIntegerLiteral, ExpressibleByFloatLiteral, ExpressibleByStringLiteral,
CustomStringConvertible {
    var name: String = ""
    var score: Double = 0
    required init(floatLiteral value: Double) { self.score = value }
    required init(integerLiteral value: Int) { self.score = Double(value) }
    required init(stringLiteral value: String) { self.name = value }
    required init(unicodeScalarLiteral value: String) { self.name = value }
    required init(extendedGraphemeClusterLiteral value: String) { self.name = value }
    var description: String { "name=\(name),score=\(score)" }
}
var stu: Student = 90
print(stu) // name=,score=90.0
stu = 98.5
print(stu) // name=,score=98.5
stu = "Jack"
print(stu) // name=Jack,score=0.0
```

小码哥教育 SEEMYGO

字面量协议应用

```
struct Point {
    var x = 0.0, y = 0.0
}
extension Point : ExpressibleByArrayLiteral, ExpressibleByDictionaryLiteral {
    init(arrayLiteral elements: Double...) {
        guard elements.count > 0 else { return }
        self.x = elements[0]
        guard elements.count > 1 else { return }
        self.y = elements[1]
    }
    init(dictionaryLiteral elements: (String, Double)...) {
        for (k, v) in elements {
            if k == "x" { self.x = v }
            else if k == "y" { self.y = v }
        }
    }
}
var p: Point = [10.5, 20.5]
print(p) // Point(x: 10.5, y: 20.5)
p = ["x" : 11, "y" : 22]
print(p) // Point(x: 11.0, y: 22.0)
```