# 方法

@M了个J

https://github.com/CoderMJLee

http://cnblogs.com/mjios

码拉松

# 方法（Method）

- 枚举、结构体、类都可以定义实例方法、类型方法
  - 实例方法（Instance Method）：通过实例对象调用
  - 类型方法（Type Method）：通过类型调用，用static或者class关键字定义

```
class Car {
    static var cout = 0
    init() {
        Car.cout += 1
    }
    static func getCount() -> Int { cout }
}
let c0 = Car()
let c1 = Car()
let c2 = Car()
print(Car.getCount()) // 3
```

- self
  - 在实例方法中代表实例对象
  - 在类型方法中代表类型

- 在类型方法static func getCount中
  - cout等价于self.cout、Car.self.cout、Car.cout

# mutating

- 结构体和枚举是值类型，默认情况下，值类型的属性不能被自身的实例方法修改
- 在func关键字前加mutating可以允许这种修改行为

```
struct Point {
    var x = 0.0, y = 0.0
    mutating func moveBy(deltaX: Double, deltaY: Double) {
        x += deltaX
        y += deltaY
        // self = Point(x: x + deltaX, y: y + deltaY)
    }
}
```

```
enum StateSwitch {
    case low, middle, high
    mutating func next() {
        switch self {
        case .low:
            self = .middle
        case .middle:
            self = .high
        case .high:
            self = .low
        }
    }
}
```

# @discardableResult

- 在func前面加个@discardableResult，可以消除：函数调用后返回值未被使用的警告⚠️

```
struct Point {
    var x = 0.0, y = 0.0
    @discardableResult mutating
    func moveX(deltaX: Double) -> Double {
        x += deltaX
        return x
    }
}
var p = Point()
p.moveX(deltaX: 10)
```

```
@discardableResult
func get() -> Int {
    return 10
}
get()
```