流程控制

@M了个J

https://github.com/CoderMJLee

http://cnblogs.com/mjios

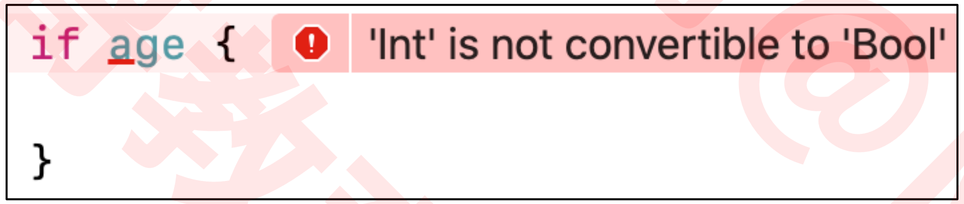
码拉松

if-else

```
let age = 4
if age >= 22 {
    print("Get married")
} else if age >= 18 {
    print("Being a adult")
} else if age >= 7 {
    print("Go to school")
} else {
    print("Just a child")
}
```

- if后面的条件可以省略小括号
- 条件后面的大括号不可以省略

- if后面的条件只能是Bool类型



```
if age {    'Int' is not convertible to 'Bool'

}
```

while

```swift
var num = 5
while num > 0 {
    print("num is \(num)")
    num -= 1
} // 打印了5次
```

```swift
var num = -1
repeat {
    print("num is \(num)")
} while num > 0 // 打印了1次
```

- repeat-while相当于C语言中的do-while
- 这里不用num--，是因为
  - 从Swift3开始，去除了自增（++）、自减（--）运算符

小码哥教育 SEEMYGO

for

- 闭区间运算符：a...b， a <= 取值 <= b

```
let names = ["Anna", "Alex", "Brian", "Jack"]
for i in 0...3 {
    print(names[i])
} // Anna Alex Brian Jack
```

```
let range = 1...3
for i in range {
    print(names[i])
} // Alex Brian Jack
```

```
let a = 1
var b = 2
for i in a...b {
    print(names[i])
} // Alex Brian


for i in a...3 {
    print(names[i])
} // Alex Brian Jack
```

```
// i默认是let，有需要时可以声明为var
for var i in 1...3 {
    i += 5
    print(i)
} // 6 7 8
```

```
for _ in 1...3 {
    print("for")
} // 打印了3次
```

- 半开区间运算符：a..<b， a <= 取值 < b

```
for i in 1..<5 {
    print(i)
} // 1 2 3 4
```

for – 区间运算符用在数组上

```
let names = ["Anna", "Alex", "Brian", "Jack"]
for name in names[0...3] {
    print(name)
} // Anna Alex Brian Jack
```

■ 单侧区间：让区间朝一个方向尽可能的远

```
for name in names[2...] {
    print(name)
} // Brian Jack

for name in names[...2] {
    print(name)
} // Anna Alex Brian

for name in names[..<2] {
    print(name)
} // Anna Alex
```

```
let range = ...5
range.contains(7) // false
range.contains(4) // true
range.contains(-3) // true
```

区间类型

```
let range1: ClosedRange<Int> = 1...3
let range2: Range<Int> = 1..<3
let range3: PartialRangeThrough<Int> = ...5
```

■ 字符、字符串也能使用区间运算符，但默认不能用在for-in中

```
let stringRange1 = "cc"..."ff" // ClosedRange<String>
stringRange1.contains("cb") // false
stringRange1.contains("dz") // true
stringRange1.contains("fg") // false

let stringRange2 = "a"..."f"
stringRange2.contains("d") // true
stringRange2.contains("h") // false
```

```
// \0到~囊括了所有可能要用到的ASCII字符
let characterRange: ClosedRange<Character> = "\0"..."~"
characterRange.contains("G") // true
```

带间隔的区间值

```swift
let hours = 11
let hourInterval = 2
// tickMark的取值：从4开始，累加2，不超过11
for tickMark in stride(from: 4, through: hours, by: hourInterval) {
    print(tickMark)
} // 4 6 8 10
```

switch

```
var number = 1
switch number {
case 1:
    print("number is 1")
    break
case 2:
    print("number is 2")
    break
default:
    print("number is other")
    break
} // number is 1
```

- case、default后面不能写大括号{}

```
var number = 1
switch number {
case 1:
    print("number is 1")
case 2:
    print("number is 2")
default:
    print("number is other")
} // number is 1
```

- 默认可以不写break，并不会贯穿到后面的条件

fallthrough

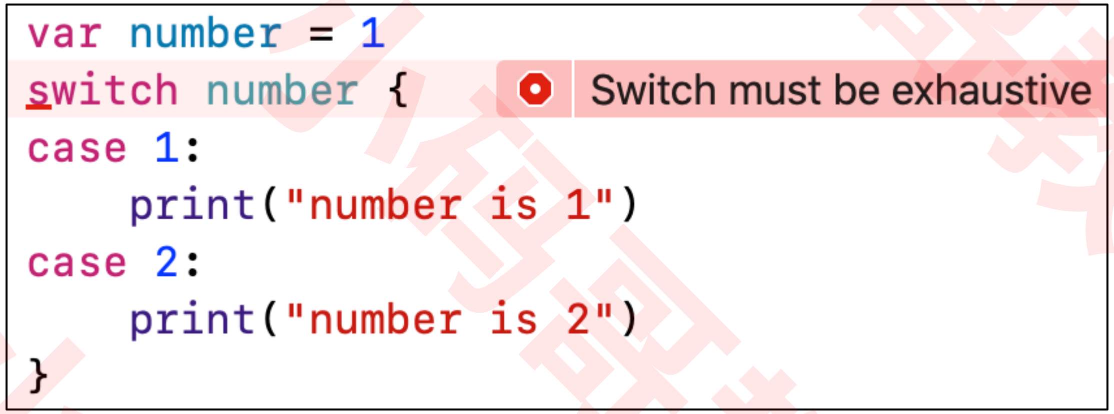
■ 使用fallthrough可以实现贯穿效果

```swift
var number = 1
switch number {
case 1:
    print("number is 1")
    fallthrough
case 2:
    print("number is 2")
default:
    print("number is other")
}
// number is 1
// number is 2
```

小码哥教育 SEEMYGO

switch注意点

- switch必须要保证能处理所有情况



```
var number = 1
switch number {        Switch must be exhaustive
case 1:
    print("number is 1")
case 2:
    print("number is 2")
}
```

- case、default后面至少要有一条语句
- 如果不想做任何事，加个break即可

```
var number = 1
switch number {
case 1:
    print("number is 1")
case 2:
    print("number is 2")
default:
    break
}
```

switch注意点

■ 如果能保证已处理所有情况，也可以不必使用default

```
enum Answer { case right, wrong }
let answer = Answer.right
switch answer {
case Answer.right:
    print("right")
case Answer.wrong:
    print("wrong")
}
```

```
// 由于已确定answer是Ansewer类型，因此可以省略Answer
switch answer {
case .right:
    print("right")
case .wrong:
    print("wrong")
}
```

小码哥教育 SEEMYGO

复合条件

- switch也支持Character、String类型

```
let string = "Jack"
switch string {
case "Jack":
    fallthrough
case "Rose":
    print("Right person")
default:
    break
} // Right person
```

```
switch string {
case "Jack", "Rose":
    print("Right person")
default:
    break
} // Right person
```

```
let character: Character = "a"
switch character {
case "a", "A":
    print("The letter A")
default:
    print("Not the letter A")
} // The letter A
```

区间匹配、元组匹配

```swift
let count = 62
switch count {
case 0:
    print("none")
case 1..<5:
    print("a few")
case 5..<12:
    print("several")
case 12..<100:
    print("dozens of")
case 100..<1000:
    print("hundreds of")
default:
    print("many")
} // dozens of
```

```swift
let point = (1, 1)
switch point {
case (0, 0):
    print("the origin")
case (_, 0):
    print("on the x-axis")
case (0, _):
    print("on the y-axis")
case (-2...2, -2...2):
    print("inside the box")
default:
    print("outside of the box")
} // inside the box
```

- 可以使用下划线 _ 忽略某个值
- 关于case匹配问题，属于模式匹配（Pattern Matching）的范畴，以后会再次详细展开讲解

值绑定

```
let point = (2, 0)
switch point {
case (let x, 0):
    print("on the x-axis with an x value of \(x)")
case (0, let y):
    print("on the y-axis with a y value of \(y)")
case let (x, y):
    print("somewhere else at (\(x), \(y))")
} // on the x-axis with an x value of 2
```

- 必要时let也可以改为var

where

```
let point = (1, -1)
switch point {
case let (x, y) where x == y:
    print("on the line x == y")
case let (x, y) where x == -y:
    print("on the line x == -y")
case let (x, y):
    print("(\(x), \(y)) is just some arbitrary point")
} // on the line x == -y
```

```
// 将所有正数加起来
var numbers = [10, 20, -10, -20, 30, -30]
var sum = 0
for num in numbers where num > 0 { // 使用where来过滤num
    sum += num
}
print(sum) // 60
```

小码哥教育 SEEMYGO

标签语句

```
outer: for i in 1...4 {
    for k in 1...4 {
        if k == 3 {
            continue outer
        }
        if i == 3 {
            break outer
        }
        print("i == \(i), k == \(k)")
    }
}
```