Зимин Александр

azimin@me.com

1. Перегрузка операторов
2. Debug View Hierarchy и @IBDesignable/@IBInspectable
3. Apps groups & today extension

Перегрузка операторов

```
<Модификатор> operator <операция> {
    <Свойства>
}
```

Модификаторы операций

infix

Модификатор инфикс операции

Выполняется для двух объектов

Пример: a + b

prefix

Модификатор префиксов операции

Выполняется для одного объекта

Пример: !flag

postfix

Модификатор постфиксной операции

Выполняется для одного объекта

Пример: i++

Свойства операций (для infix)

Precedence (UInt8)

- Приоритет операции
- Чем больше, тем приоритетнее
- Числа от 0 до 255
- Базовое значение 100

Associativity

- Ассоциативность операции
- Может быть left, right, none

Assignment

- Присвоение
- Используется для операций формата =
- Только для **infix** операций

Приоритет

```
infix operator >+ {
    precedence 40
}
func >+ (left: Double, right: Double) -> Double {
    return left + right
}

infix operator >* {
    precedence 30
}
func >* (left: Double, right: Double) -> Double {
    return left * right
}

let value = 10 >+ 5 >* 2 // 30
let newValue = 10 >+ 5 >+ 2 error: non-associative operator
is adjacent to operator of same precedence
```

Ассоциативность

```
infix operator >- {
    associativity left
}
func >- (left: Double, right: Double) -> Double {
    return left - right
}
```

```
infix operator >/ {
    associativity left
}
func >/ (left: Double, right: Double) -> Double {
    return left / right
}

let nValue = 10 >- 4 >/ 2 // 3
let anotherValue = 10 >/ 2 >- 4 // 1
```

```
infix operator **= {
    precedence 120
    assignment
}
func **= (inout left: Double, right: Double) -> Double {
    left = left * right
    return left / right / 2
}

infix operator ** {
    associativity left
}
func ** (left: Double, right: Double) -> Double {
    return left * right
}

infix operator +++ {
    associativity left
}
func +++ (left: Double, right: Double) -> Double {
    return left + right
}

var x = 3 +++ 4 ** 5
let y = x **= 2 ** 4
y
x
```

```
infix operator **= {
    precedence 120
    assignment
}
func **= (inout left: Double, right: Double) -> Double {
    left = left * right
    return left / right / 2
}

infix operator ** {
    associativity left
}
func ** (left: Double, right: Double) -> Double {
    return left * right
}

infix operator +++ {
    associativity left
}
func +++ (left: Double, right: Double) -> Double {
    return left + right
}

var x = 3 +++ 4 ** 5 // 35.0
let y = x **= 2 ** 4
y // 70.0
x // 70.0
```

Перегрузка операций для классов

```
struct Vector2D {
    var x = 0.0, y = 0.0
}
```

```
func + (left: Vector2D, right: Vector2D) -> Vector2D {
    return Vector2D(x: left.x + right.x, y: left.y + right.y)
}
```

```
let vectorSum1 = Vector2D(x: 1.0, y: 2.0)
let vectorSum2 = Vector2D(x: 0.5, y: -1.0)
let vectorSum3 = vectorSum1 + vectorSum2 // {x 1.5, y 1.0}
```

```swift
prefix func -(vector: Vector2D) -> Vector2D {
    return Vector2D(x: -vector.x, y: -vector.y)
}

prefix func +(vector: Vector2D) -> Double {
    return vector.x + vector.y
}

let minusVector = -vectorSum3 // {x -1.5, y -1.0}
let result = +vectorSum3 // 2.5
```

```swift
func += (inout left: Vector2D, right: Vector2D) {
    left = left + right
}

var newVector = Vector2D(x: 1.0, y: 1.0) // {x 1.0, y 1.0}
newVector += Vector2D(x: 1.0, y: 2.0) // {x 2.0, y 3.0}
```

```swift
postfix func ++ (inout vector: Vector2D) -> Vector2D {
    vector += Vector2D(x: 1.0, y: 1.0)
    return vector
}

newVector++ // {x 3.0, y 4.0}
let updatedNewVector = newVector++ // {x 4.0, y 5.0}
```

```
infix operator ** {
    precedence 160
    associativity left
}

func ** (left: Double, right: Double) -> Double {
    return pow(left, right)
}

5 ** 2 * 2 // 50.0
3 ** 5 // 243.0
```

Pipe-Forward Operator

```swift
func fPow(a: Double)(b: Double) -> Double {
    return pow(a, b)
}
fPow(2)(b: 3) // 8.0
let powOfFive = fPow(5) // (Function)
powOfFive(b: 3) // 125.0
```

```
infix operator |>  {
  precedence 50
  associativity left
}

public func |> <T,U>(lhs: T, rhs: T -> U) -> U {
    return rhs(lhs)
}
```

```
let seqResultMultiLine =
", ".join(
    map(
        sorted(
        filter(numbers) { $0 % 2 == 0 }
        ) { $1 < $0 }
    ) { $0.description })
```

Debug View Hierarchy
@IBDesignable
@IBInspectable

Практическое применение

Debug (Debug View Hierarchy)

Легче найти потерянные элементы

Можно проверить верность сетки

Custom controls (@IBDesignable/@IBInspectable)

Вашему дизайнеру или пользователю библиотеки будет проще подправить UI

Не надо каждый раз перезапускать проект