ADVANCED SWIFT 3

Part 3: Custom Operators

Custom Operators

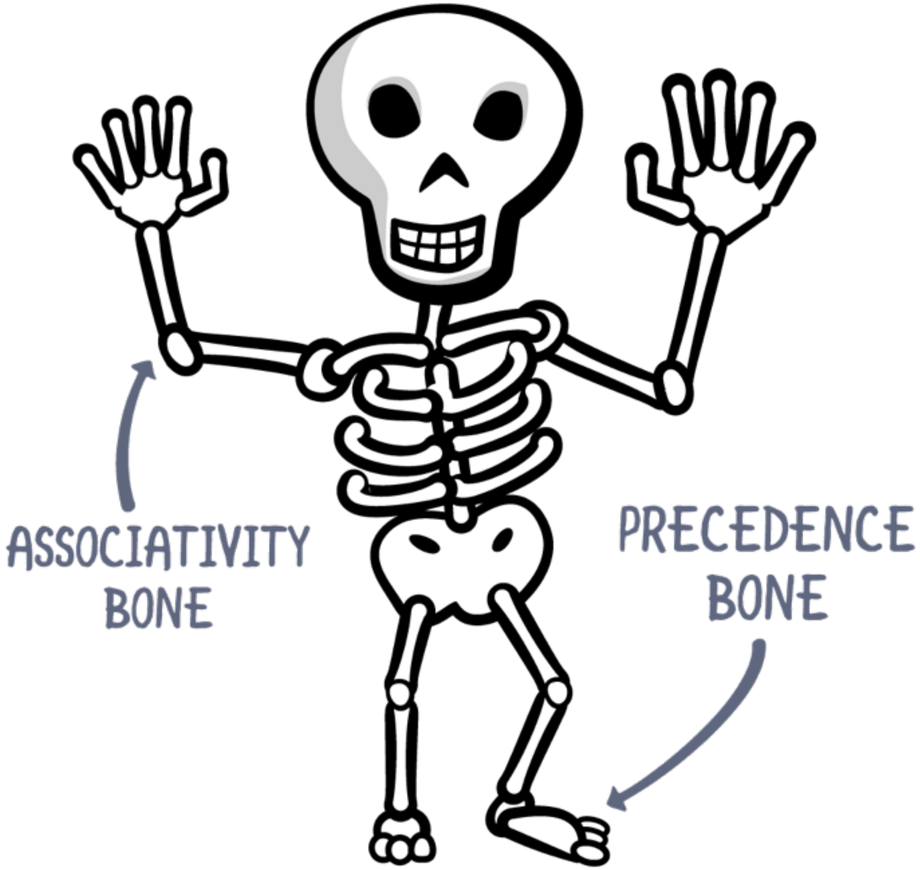
```
func printMarsha
{
                                                      "")}
                                                     ("O")};
              {p("A")}                              ~> {p("P")};
          {p("R")} ~> {p("E")};               {p("S")} ~> {p("R")};
       {p("H")}            ~> {p("A")}; {p("A")}  ~> {p("T")};
                                  {p("L")}      ~> {p("O")};
                                                 {p(" ")}
                                             ~> {p("R")};

} // http://ijoshsmith.com/2014/07/05/custom-threading-operator-in-swift/
```

```
let a = CGPoint(x: 1, y: -5)
let b = CGPoint(x: 10, y: 10)
let c = a + b
```

Anatomy of an Operator

- Name
- Arity and Position
- Precedence
- Associativity

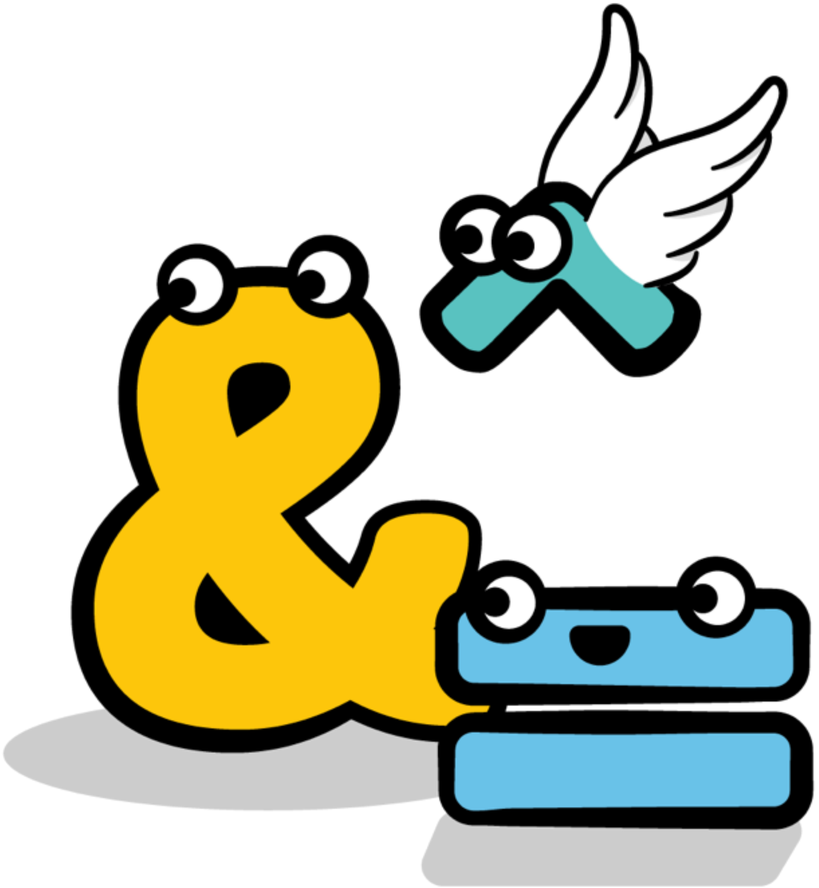


What Names Include

- Mathematical Operators
- Miscellaneous Symbols
- Dingbats

/, =, -, +, !, *, %, <, >, &, |, ^, ?, or ~

Certain operators, prefix and postfix operators are off-limits



Example: Using Existing Operators

```
extension CGPoint {
  static func +(lhs: CGPoint, rhs: CGPoint) -> CGPoint {
    return CGPoint(x: lhs.x+rhs.x, y: lhs.y+rhs.y)
  }

  static prefix func -(input: CGPoint) -> CGPoint {
    return CGPoint(x: -input.x, y: -input.y)
  }

  static func -(lhs: CGPoint, rhs: CGPoint) -> CGPoint {
    return lhs+(-rhs)
  }
}
```

Zero-Cost Abstraction

```
public func testSubtraction(a: CGPoint, b: CGPoint) ->  CGPoint {
  return a - b
}
```

```
% swiftc -emit-assembly -O CGPointOperators.swift > CGPointOperators.asm
```

```
  .globl
__TF16CGPointOperators15testSubtractionFT1aVSC7CGPoint1bS0__S0_
  .p2align 4, 0x90
__TF16CGPointOperators15testSubtractionFT1aVSC7CGPoint1bS0__S0_:
  pushq  %rbp
  movq %rsp, %rbp
  subsd  %xmm2, %xmm0
  subsd  %xmm3, %xmm1
  popq %rbp
  retq
```

Example: New Operator

```
infix operator !!: NilCoalescingPrecedence
```

Swift Standard Library Precedence Groups (Highest to Lowest)

Group Name	Associativity	Examples
BitwiseShiftPrecedence	none	<< >>
MultiplicationPrecedence	left	* /
AdditionPrecedence	left	+ -
RangeFormationPrecedence	none	... ..<
CastingPrecedence	none	is as as? as!
NilCoalescingPrecedence	right	??
ComparisonPrecedence	none	!= < > >= ==
LogicalConjunctionPrecedence	left	&&
LogicalDisjunctionPrecedence	left	||
DefaultPrecedence	none	
TernaryPrecedence	right	? :
AssignmentPrecedence	right	+= = *= -= /=

Example: New Precedence Group

```
precedencegroup ExponentiationPrecedence  {
  higherThan: MultiplicationPrecedence
  associativity: none  // you must use parenthesis
}
```

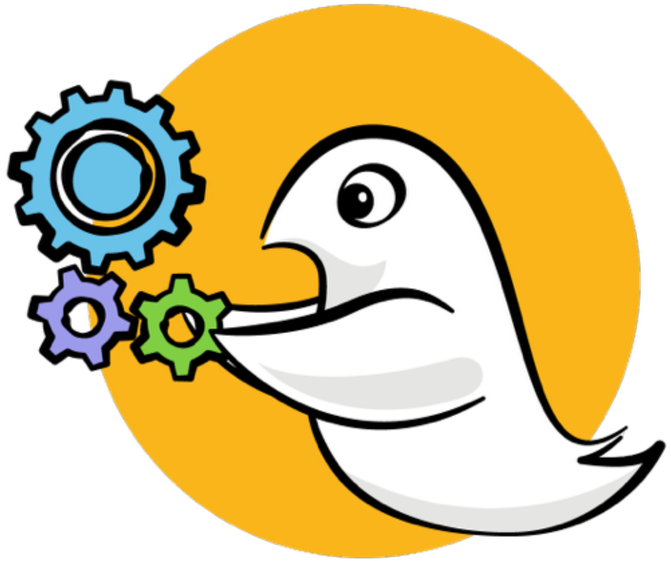
Short Circuiting and Deferred Execution

```
if expensiveFirst() && expensiveSecond() {
  // ...
}
```

```
extension Bool {
  static func &&(lhs: Bool, rhs: @autoclosure () -> Bool) -> Bool
}
```

Challenge

- Define a new operator +/- that creates a floating point Range around a center point. The center point is the first argument to the operator and the value spread is the second.



Example: 3.0 +/- 1.0
produces a range from
2.0 to 4.0