吐槽一些奇怪的关键字

很多程序设计语言都会有自己的关键字，有一些关键字是多语言通用的，如大家熟悉的 while, var, let, for 等等。这些关键字一般来说，要么是英文单词，要么是单词的缩写，比如 var 就是 variable 的缩写。但有些语言，会使用一些不太符合我的审美的关键字，使得这门语言看起来多多少少有些奇怪。

"val"-Kotlin

我首先要说的是 Kotlin 中的 val 关键字。val 不是 Kotlin 首创的，但此处谈论的对象仅仅是 Kotlin 中的 "val"。这个由3个字母组成的关键字多多少少有点奇怪：在语义上，它表示声明运行时常量，其它语言具有类似功能的关键字可能是 "const", "let" 等等（Kotlin 也有自己的 const，但语义不同），"val" 的问题在于它和 Kotlin 另一个同样用来声明变量的关键字 "var" 只差一个字母！在没有代码高亮的情况下，给出数行变量和常量声明，你能一眼看出来哪些是变量哪些是常量吗?

```
val a = 5
var b = 6
val c = 7
val d = 8
var e = 9
var f = 10
var g = 11
val h = 12
val i = 13
val j = 14
val k = 15
val l = 16
var m = 17
val n = 18
val o = 19
```

对于我这种佛性程序员，Kotlin 这种设计简直要了我小命了。如果你觉得上面这段代码很清晰，很容易分辨出哪些是 val 哪些是 var，那么你看一看下面这段同样语义的 Swift 风格的代码，再进行一下比较（你可以通过统计 var 的个数来评定哪一种更清晰）：

```
let a = 5
var b = 6
let c = 7
let d = 8
var e = 9
```

```
var f = 10
var g = 11
let h = 12
let i = 13
let j = 14
let k = 15
let l = 16
var m = 17
let n = 18
let o = 19
```

我不想再多讨论这个关键字了，真让眼睛难受！

"fun"和"fn"

声明函数的相关关键字有诸如 "function"(JavaScript), "func"(Swift, Golang), "fun"(ML家族)等，这其中 "function" 是最直接——一眼就能看出语义的关键字，它在 JavaScript 中的作用不仅是声明函数，还可以是匿名函数字面表达式的标志（就像某些语言的 lambda）；"func" 简短了一些，倒也能让人看出所以然；"fun" 被用得很多，前面刚提过的 Kotlin 就使用 "fun" 来做函数声明的关键字，但这个词其实是有小问题的。

首先，关键字的长度一般都在 3~5 个字符为宜，太长写起来麻烦，太短看不懂（除了像 if 这样单词本身就只有 2 个字符的关键字）。有些语言中用 double 声明双精度浮点数，倒也无可厚非，毕竟在像 C 这样的语言中，这样的关键字不多，而且 double 一词也很难进行缩写（continue 有些太长了，但出现频率低）。相比起来 fun 就有点过分，因为 "fun" 本身就是一个英文单词！当我使用 Kotlin 写的程序出了 bug，我对着源码进行修改时，看到满屏的 "fun"，仿佛受到了 Kotlin 的嘲讽！

那时候我的眼睛看到的东西：

```
fun XXXXXXXXXX(xxxxxxxxxxx) {
  XXXXXXXXXXXXXXXXX
  xxxxxxxxxxxxxxxxx
}

fun XXXXXX(XXX) {
  XXXXXXX
}

fun xxxxxxxxxx(xxxxxxxx) {
  xxxxxxxxx
  xxxx
  xxxxxxxxxxxxxxxx
```

```
  xxxxxxxxx
}
```

就像这样，只有 fun 是最抢眼的。这些代码有时候甚至就像这样：

```
^_^ XXXXXXXXXX(xxxxxxxxxxx) {
  XXXXXXXXXXXXXXXXX
  xxxxxxxxxxxxxxxxx
}

^_^ XXXXXX(XXX) {
  XXXXXXX
}

^_^ xxxxxxxxxx(xxxxxxxx) {
  xxxxxxxxx
  xxxx
  xxxxxxxxxxxxxxxx
  xxxxxxxxx
}
```

或者，你可以把颜文字脑补成#滑稽。

如果你和我一样觉得 fun 这个关键字缩写得有些过分，那么你也应当接受不了 Rust 中的 "fn" 关键字。Rust 中的关键字大都比较短小，fn 就是一个典型。说起 fn，想必你会想起很多笔记本键盘左下角的那个按键。Rust 和那个按键通过同样的两个字母让我有了同样不怎么美好的回忆。

Go的类型声明

这个问题可能和关键字无关，但它们别扭的程度让人抓狂。

很多语言，尤其是动态类型或者自带类型推导的语言，在声明关键字的时候会使用后置类型标注语法。也就是说，你可以不为一个变量声明类型，交给编译器/解释器自己推导或者根本不关心其类型；也可以手动进行限定，当你需要在声明期就确定其类型时，或者为了防止类型错误发生时。

你可能会写出诸如这样的代码：

```
let a = 5
var b = "Kotlin"
let c: Double = 3.0
```

乍看起来也没有让人感到不适之处，可是 Go 为什么要这样设计呢?

```
var a int = 5

func add(a, b int) {
   ...
}
```

第一条语句，到底 a 是 5 还是 int 是 5？程序员当然知道这是变量 a 的声明，但这看起来不奇怪吗？你看看第二个函数声明语句，这个 int 到底是修饰 b 的类型还是同时修饰 a 和 b 的类型？从语法上来说，Go 的做法肯定可以让这两条语句得到单一、确定的语义，但从人的直觉上来说，它们就是别扭！好在大部分人写 Go 代码都会这样写代码：

```
a := 5
```

吐槽了这么多，其实我并不是真的不喜欢 Kotlin。Kotlin 有它自己的设计上的问题，但不在于此。不过 Go 语言各方面问题都很大。不少人说 Go 的诞生是历史的倒退，我赞同这种看法。改天我可能会聊聊 Kotlin 和 Go 的其它问题。

（本文纯属吐槽+扯淡，如果它给你带来了欢乐，请出门转账。本文定价1元）