Yoda 表示法错在哪里

在上一篇博文里，我提到了 Yoda 表示法。

Yoda Notation（Yoda 表示法）

它的含义是，在 C/C++ 里面使用这样的表达式顺序：

```
if ("blue" == theSky) ...
```

这是为了避免意外的写成：

```
if (theSky = "blue") ...
```

“Yoda 表示法”的名字来源于《星球大战》的 Yoda 大师。他说话的单词顺序相当奇特，比如：“Backwards it is, yes!”

一般认为

> 使用这个表示法是为了“变通”（workaround） C/C++ 的一个设计抉择：使用 = 来表示赋值，而使用 == 来表示比较。这个设计充分的展现了“先辈的罪”（Sins of our Forefathers）这一词汇的精髓。

我认为

> 使用 = 来表示赋值其实并不是真正的错误所在。真正的错误在于 C/C++ 的赋值语句不应该返回一个值。

也就是说，`theSky = "blue"` 的所有功能应该只是“赋值”这种“副作用”，副作用不应该具有“值”。即使你牵强附会说它有一个值，它的“值”也应该是 `void`（随之这个 `void` 会被类型检查所拒绝，因为它不是 `if` 所期望的 `bool`）。所以，一个良好的语言不应该允许你把 `theSky = "blue"` 放进 `if (...)` 的“条件”里面。如果你真的要赋值又要判断，它会迫使你把这拆开成两行：

```
theSky = "blue";
if (theSky) ...
```

更近一步。`if (theSky)` 这个写法其实也是一个先辈的罪。`theSky` 的类型是 `string`，它不应该可以直接被作为 `bool` 使用。`if (...)` 的条件应该必须是一个 `bool`。 所以这里其实应该写成：

```
theSky = "blue";
if (theSky != NULL) ...
```

因为赋值语句永远不可能出现在条件的位置，所以之前的那种错误，即使我们使用 = 作为赋值操作符，也完全不可能出现。这样我们也就完全没必要用 Yoda 表示法了。

相反，如果我们只是把 = 换成像 Pascal 的 := 这样的赋值操作符，而保留其它的“特性”（赋值操作会返回值）的话，我们其实还是会遇到同样的问题：

```
if (theSky := "blue") ...
```

这里假设你想打 =，却不小心打成了 :=。机会虽然小，但是仍然有可能。而我推荐的解决方案，会让你故意想犯错误都不可能，编译器会拒绝接受你的程序。

所以你看到了，问题的根源其实不在于赋值操作的名字，而是有更深的原因。