Programming Experiences

Yuhao Zhang

October 30, 2018

Contents

1 Say Goodbye to Error Prone

2 Increase the Code Readability

2.1 Argument Labels and Keyword

In a language like Swift, we can add argument labels to arguments in our function. The argument label in Swift works similarly to the keyword facility in Racket.

To see what argument labels and keyword could achieve for us, see the code below:

```
#lang racket
(require 2htdp/image)

;; (star-polygon side-length
;;               side-count
;;               step-count
;;               outline-mode
;;               pen-or-color) → image?
;;   side-length : (and/c real? (not/c negative?))
;;   side-count : side-count?
;;   step-count : step-count?
```

```
;;   outline-mode : (or/c 'outline "outline")
;;   pen-or-color : (or/c pen? image-color?)
(star-polygon 40 7 3 "outline" "darkred")
```

Can you understand what does the call of function do without looking at the function signature of it? I guess you would tend to feel very hard to guess what does these function do. In this case, use argument labels or keywords would increate the readability in a very fancy way:

```
(star-ploygon #:side-length 40
      #:side-count 7
      #:step-count 3
      "outline" "darked")
```

Notice here we use line break to expose the keyword arguments; this is a technique worth noticing.

Though constructs like label seems doesn't do any bad, it can cause over-engineering if it was overly used. On the otherside we always have doc comments. Since IDE is so popular in recent days, it could be more important to write a good documentation instead of focus your mind thinking a verb for the use of an argument label. Every programmer who wants to be a 'function caller' needs to take their time reading docs and comments, especially when you encountered a function like `star-polygon`.

```
struct Posn {
    var x: Int
    var y: Int
}

enum Direction {
    case north, south, east, west
}

let position = Posn(x: 0, y: 1)

/// This function moves a position Posn
func move(to direction: Direction, distance: Int) {
    position.x = position.x + distance
}
```

KeywordLabelKeyword

```
(define greet
  (lambda (given #:last surname)
    (string-append "Hello, " given " " surname)))

(greet "John" #:last "Smith")
; "Hello, John Smith"
(greet #:last "Doe" "John")
; "Hello, John Doe"
```

keywordkeywordkeyword