ShiftLess

低 编 码 疲 劳 语 法 之 野 望

—— 喻恒春

GDG DevFest ZhengZhou 2016

万能的 Ada 神啊, 请赐予我们低编码疲劳的语言吧

愤怒的小指

- Choices-Statement　if, switch

可以统一语义么?

- Type-Assertion　instanceof, .(type)

IF

JavaScript

```
if (x instanceof String) {
} else {
}
```

Golang

```
switch (val) {
    case v1:break;
    case v2, v2:break;
    default:
}
```

Golang

```
func TypeAssertion(x interface{}) {
    switch instance := x.(type) {
    case string:
    case int:
    default:
    }
}
```

IF-OF

```
if x of 1
    ...
of 2,3
    ...
else
    ...
```

```
if instanc, x of string
    ...
of int
    ...
else
    ...
```

- Loop-Statement　for, while

可以统一语义么?

- Range　first..end, range x

FOR

JavaScript

```
while (c)
do ... while (!c)
```

JavaScript,Golang,Python

```
for true {
    for x-- {
        for i := 0; i < count; i++ {
            for x in first..end {
                for x of first...last {
                }
            }
        }
    }
}
```

Python

```
for iterating_var in sequence:
    statements(s)
```

OF-ELSE

```coffee
if instance, x of string
    echo 'instance is string'
of int
    echo 'instance is int'
else

for x of 0
    echo 'x is 0'
```

```coffee
for x of 1...5,step
    echo x
else
    echo 'other'

for x of string, int
    echo 'x is string or int'
```

- true, 1, ‘string’, -1

非黑即白么?

- false, 0,‘’, nil

```javascript
function compare(left, right) {
    return left < right && -1 ||
        left > right && 1 || 0
}
```

JavaScript

```golang
func compare(left, right int) (c int) {
    if left < right {
        return -1
    } else if left > right {
        return 1
    }
    return
}
```

Golang

NULLABLE

```
fun compare(int left, right; out bool)
    if left == right
        out null
    out left < right

    out left == right and null or
        left < right
```

来排个版

```
++++++++++[>+++++++>++++++++++>+++>+<<<<-]
>++.>+.+++++++..+++.>++.<<+++++++++++++++.
>.+++.------.--------.>+.>.
```

JavaScript

```
x = obj
      .c()
      .d()
y = [
      1,
      2,
]
if (
      x == obj
      .c()
      .d()
) {}
z = obj.
c().
d()
```

Golang

```
x := obj.
      c().
      d()
y := []int{
      1,
      2,
}
if x == obj
      c().
      d() {
}
z := T{
      {1, 2}, {3, {4,
            5},
            6},
}
```

INDENT

AST

YAML

```yaml
ast:
  SelfChain:
    - Ident:
        NamePos: 1:1
        Name: "obj"
    - CallExpr:
        NamePos: 1:5
        Name: "c"
    - CallExpr:
        NamePos: 1:9
        Name: "d"
```

https://astexplorer.net

AST Explorer | Save | Fork | JavaScript | babylon6

```
1 obj.c().d()
2
```

Tree | JSON

Autofocus | Hide methods | Hide empty keys | Hide location

Golang

```
*ast.CallExpr {
Fun: *ast.SelectorExpr {
.  X: *ast.CallExpr {
.  .  Fun: *ast.SelectorExpr {
.  .  .  X: *ast.Ident {
.  .  .  .  NamePos: 4:1
.  .  .  .  Name: "obj"
.  .  .  }
.  .  .  Sel: *ast.Ident {
.  .  .  .  NamePos: 4:5
.  .  .  .  Name: "c"
.  .  .  }
.  .  }
.  .  Lparen: 4:6
.  .  Ellipsis: -
.  .  Rparen: 4:7
.  }
.  Sel: *ast.Ident {
.  .  NamePos: 4:9
.  .  Name: "d"
.  }
}
```

JavaScript

```
- expression: CallExpression {
    type: "CallExpression"
  - callee: MemberExpression {
      type: "MemberExpression"
    - object: CallExpression {
        type: "CallExpression"
      - callee: MemberExpression {
          type: "MemberExpression"
        - object: Identifier = $node {
            type: "Identifier"
            name: "obj"
          }
        - property: Identifier {
            type: "Identifier"
            name: "c"
          }
        computed: false
      }
      arguments: [ ]
    }
- property: Identifier {
    type: "Identifier"
    name: "d"
  }
```

存在即合理

- undefined
- null
- nil

凡是真实的东西都是合乎理性的 – 黑格尔

```coffee
# Assignment:
number   = 42
opposite = true
```

CoffeeScript

let

VAR

```
    _, .KB, .MB, .GB, .TB = 1 << 10*iota
    .human-size = [string]['KB', 'MB', 'GB', 'TB']

var
    .customize-size = [string]['KB', 'MB', 'GB', 'TB']

fun local-variables()
    KB = 9
    customize-size[0] = 'kilo-bytes'
    call(MB)
     illegal
    human[0] = 'kilo-bytes'
```

Golang

```
// The appe// result of append, ofds elements
// it has s//       slice = append(stination i
// new elem//       slice = append(w underlyin
// Append r// As a special case, iIt is there
// result o                       iable holdi
//  slice = append(slice, elem1, elem2)
//  slice = append(slice, anotherSlice...)
// As a special case, it is legal to append a st
//  slice = append([]byte("hello "), "world"...)
func append(slice []Type, elems ...Type) []Type
```

COMMENTS

```
fun append([type] slice, type ...elems; out [type])
    #The append built-in function appends elements to
    #it has sufficient capacity, the destination is r
     new elements. If it does not, a new underlying a
     Append returns the updated slice. It is therefor
     result of append, often in the variable holding
        slice = append(slice, elem1, elem2)
        slice = append(slice, anotherSlice...)
     As a special case, it is legal to append a strir
    #   slice = append([]byte("hello "), "world"...)
```

ERROR HANDLING

```
fun open(string name; out io.file)
    out io.open(name)

fun example()
    err, file = open('somefile')
    if err
        out handing(err)
    processing(file)

    file = open('somefile')
    processing(file)
    catch err
        out handing(err)
```

JavaScript

```
try {

}catch(e){

}
```

Ruby

```
begin
    file = open("/unexistant_file")
    if file
       puts "File opened successfully"
    end
rescue
       file = STDIN
end
```

Golang

```
defer func() {
    if r := recover(); r != nil {
        //...
    }
}()
```

TEMPLATE

```
let emptytitle =
    `<section class="empty">这家伙很懒</section>`

fun title({any} root; out string)
    out `\
    <section class="container">\
      <h1>${.title}</h1><h2>${root['subtitle']}</h2>\
      <div class="content">${htmlescape(.content)}</div>\
    </section>`

fun tag(string tag, content;out string)
    out `<${tag}>${content}</${tag}>`

fun links([string] url; out string)
    out `${for s of url out tag('a',s)}`

fun template({any} root; out string)
    out `${if .title out title(root); else out emptytitle;}\
    ${if un .links out;}\
    ${links(.links)}`
```

LINKS

- https://gobyexample.com/
- http://coffeescript.org/
- https://astexplorer.net/
- http://rigaux.org/language-study/syntax-across-languages.html
- http://hyperpolyglot.org/scripting

谢谢&QA