Dao

(programming language)
Language

- childhood
- express human will
- ideology/culture
- reliably transfer this inf. to a machine
- phrases, idioms
- (partial) formality
IT goals

- strong sides (clarity, expressiveness, control)
- clean and convenient interfaces
- modern concurrency
- accent on embedding
- satisfaction from coding
- full ecosystem (stdlib, utilities, apps, games, commercial systems)
Step aside

✓ borrows heavily from various languages
✓ every idea rethought (with substantially different results from the origin)
✓ e.g.
  • enums (Ruby)
  • decorators (Python)
  • defer (Go)
  • future concurrency (Scala)
Language “market“

✓ functional prog. concepts, advanced typing system, immutability etc.
  • expressiveness, readability and safety
  • **required** control (run-time → compile)

✓ broadening adoption of scripting languages (increase in code's size)
  • readability and controllability more important than ease of use
Design

- VM (with GC), interoperability
- fast exec. in mind and prepared for JIT
- UTF-8
- sophisticated type system (inference, compile-time, templates, any, type holders)
- modularity (OOP, modules)
- control flow (including BNF macros)
Design details

✓ modules (Dao, C, C++)
✓ higher-order functions, code sections (with X, Y), decorators, code deferring
✓ classes, interfaces (abstract, concrete), mixins, wrapped types
✓ invar, casting, ...
✓ exception handling, concurrency means
Example 1

0$ dao -e 'io.writeln("Hello world!")'
Hello world!
= none
0$ dao -e ' "Hello world!" ' 
= Hello world!
0$
Example 2

0$ cat ex1.dao
#!/usr/bin/dao
routine main(... as args) {
  io.writeln(args)
  return 2
}
0$ ./ex1.dao
( )
2$ ./ex1.dao abc 5 7
( "abc", "5", "7" )
2$
Example 3a

```plaintext
load mt
routine abc(c: mt.Channel<int>, i: int) {
  defer { c.cap(0) }
  c.send(i)
}
chans = {=>}
for (i = 0 : 3) {
  c = mt.Channel<int>()
  chans[c] = 0
  abc(c, i)!!
}
```
while (true) {
    x = mt.select(chans)
    if (x.status == $finished) break
    io.writeln('received int:', x.value)
}
received int: 0
received int: 1
received int: 3
received int: 2
received int: none
received int: none
received int: none
received int: none
received int: none
Example 4a

load time
interface DateTimeX {
    routine add(invar self: time.DateTime,
                ...: tuple<enum<years, months, days>, int> as
         vars) => time.DateTime
}

interface DateTimeX for DateTime {
    routine add(invar self: time.DateTime,
                ...: tuple<enum<years, months, days>, int> as
         vars) => time.DateTime {
        self.add(vargs, ...)
        return self
    }
}

Example 4b

load web.html import html

io.writeln( document {
    head { meta(http_equiv='$content_type', content='text/html; charset=utf-8') }
    body { h1(style='font-size: 50pt') {
        'My page at ' + (string)((DateTimeX)time.now()).add(years=3)
    } }
})
Example 4c

```html
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
</head>
<body>
    <h1 style="font-size: 50pt">
        My page at 2018-11-07 12:58:17.254210 +1:00
    </h1>
</body>
</html>
```
Project

✔ http://daovm.net/ (powered by Dao)
✔ https://github.com/daokoder/dao
✔ https://github.com/daokoder/dao-modules
✔ subprojects (JIT, Graphics - 3D, SQL, SDL, CXX, Studio, etc.)
✔ 6 main contributors, 7 other contributors (data from Nov 2015)
✔ reference VM (9896 B, 926192 B)