

Io

the programming language

inspirations

Smalltalk

Self

NewtonScript

Lua

Lisp

semantics

prototype-based

all values are objects

all operations are dynamic message sends

code is "data" and runtime modifiable

arguments passed by expression

no globals

syntax

simple, consistent

no keywords

no statements (only expressions)

expressions are composed only of messages

message syntax

foo

foo(a)

foo(a, b)

sample expressions

a := people select(person, person age < 30)

a := people select(age < 30)

names := people map(i, person, person name)

names := people map(name)

assignment

expression: $a := 2$

compiles to: `setSlot("a", 2)`

expression: $a = 2$

compiles to: `updateSlot("a", 2)`

conditions are expressions too

```
a := if(b == 1, c, d)
```

```
if(a == b) then(  
    ...  
) elseif(...) then(  
    ...  
)
```

loops

`while(x < 10, ...)`

`for(i, 1, 10, ...)`

`loop(...)`

`10 repeatTimes(...)`

enumeration

```
a := list("x", 2.3, "foo")
```

```
a foreach(i, v, writeln(i, " : ", v))
```

blocks and methods

`foo := method(a, a + b) // object scoped`

`foo := block(a, a + b) // lexically scoped`

objects

```
Account := Object clone do(  
  balance := 0  
  deposit := method(amount,  
    balance = balance + amount  
  )  
)
```

example

```
account := Account clone
```

```
account deposit(10.00)
```

```
writeln("balance: ", account balance)
```

everything is an object

Number double := method(self * 2)

100 double

==> 200

introspection

Number double := method(self * 2)

Number getSlot("double") code

==> "method(self *(2))"

concurrency

coroutines

async sockets and async dns

all objects can be actors

futures

concurrency

```
url := URL with("http://www.yahoo.com")
```

```
url fetch // sync message
```

```
f := url @fetch // future message
```

```
url @@fetch // async message
```

bindings and frameworks

Regex, XML, Databases, ...

OpenGL, PortAudio, FreeType, Flux, ...

platforms

osx, unix, windows (mingw)

more info

<http://www.iolanguage.com/>