The Linguist Programmer

David Muñoz
seedtag

$> t3chfest

seedtag

9 y 10 de febrero
#T3chFest2017

$$y = mx + n$$

Call cost calculator

Has this ever happened to you?

```
[user 192619247]
base_plan = two_cents_per_minute
holiday_plan = one_cent_per_minute
weekend_plan = one_cent_per_minutes
apply_discount_by_less_that_1_year = true
discount_by_less_than_1_year = 10
discount_by_more_than_1_year = 5
apply_family_discount = true
family_discount_amount = 5
apply_family_discount_before_discount_by_less_than_1_year = no
apply_credit_before_family_discount = yes
credit_enabled = true_for_holidays_only
credit_amount = 30\9 # couldn't decide if dot or comma
credit_enabled_on_holidays = false
credit_enabled_on_weekends = nope
credit_enabled_in_roaming = nah
discount_percent = whatever
discount_enable_weekends = probably_not|doubtful
roaming_affects_discounts = europe:1epermin;asia:5epersec
what_will_happen_to_us_if_we_keep_using_this_strategy = 💀
```

```
[ plan basic($cpm1, $cpm2) ]
cost =
  if #is_workday
    then (#call_length * $cpm1)
    else (#call_length * $cpm2)

[ plan premium($a, $b) ]
discount =
  if #user_age > 1 year
    then $a
    else $b

[ user 19283 ]
plan = basic(5, 2) + premium(5%, 10%)
```

Make your product hackable by design

- Define the business components (C++, Java)
- Orchestrate them with an embedded language

The easy -and probably more convenient- solution: use something that already exists

C/C++: Javascript (V8), Lua, Guile, …

JVM: Javascript (Rhino), Groovy, Clojure, …

...or create a DSL

https://github.com/voiser/t3chfest-2017

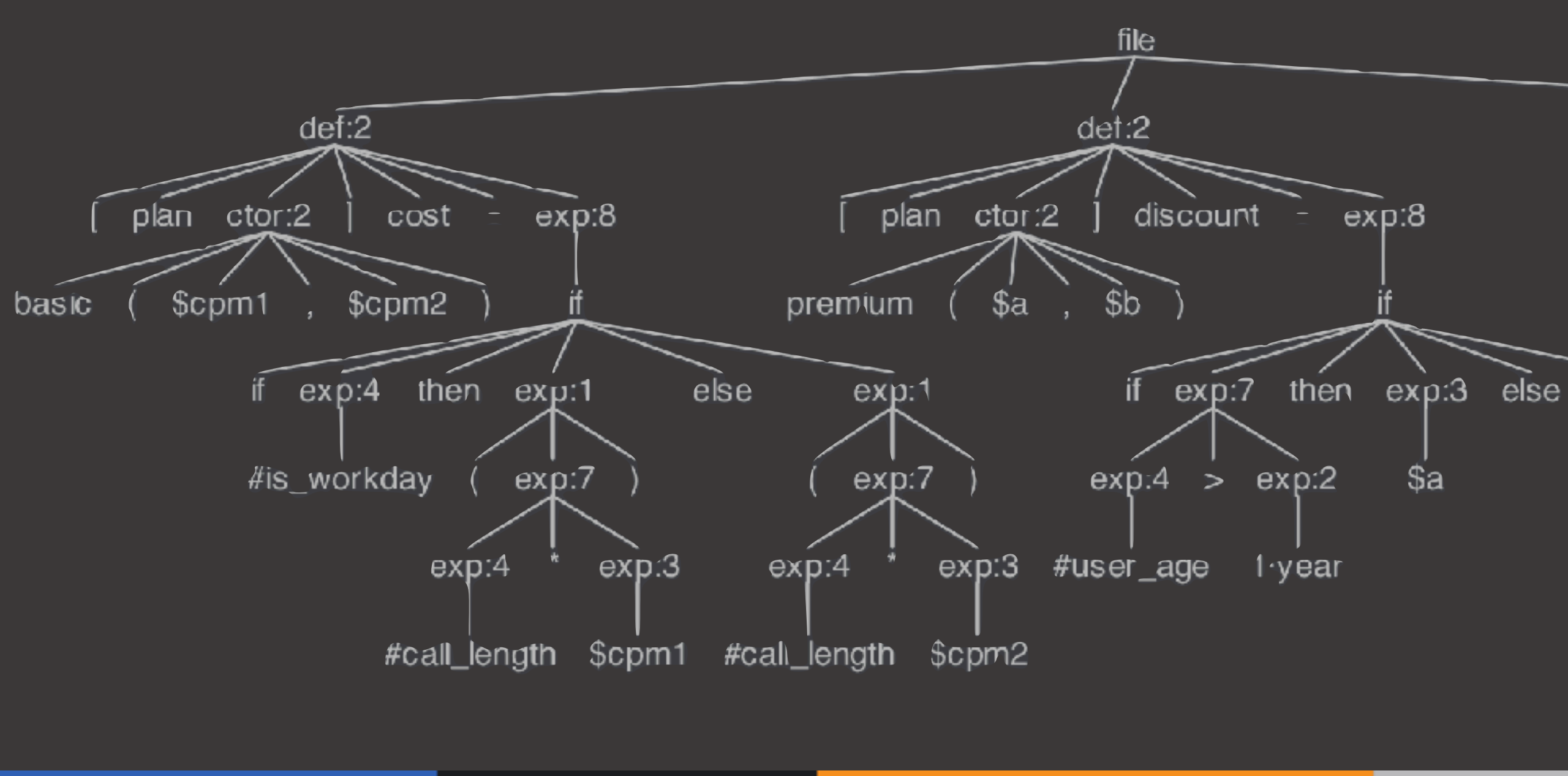
```
grammar CallCost;

file        : def+ ;
def         : '[' type=ID name=ID ']' (ID '=' exp)+
            | '[' type=ID ctor']' (ID '=' exp)+ ;
ctor        : ID '(' ')' | ID '(' ID (',' ID)* ')';
exp         : '(' exp ')' | VALUE | PARAM | ID | inst | exp BINOP exp | if;
inst        : ID '(' exp (',' exp)* ')' ;
if          : 'if' exp 'then' exp 'else' exp ;
WS          : [ \t\r\n]+ -> skip ;
PARAM       : '$'[a-zA-Z0-9_]+;
ID          : [a-zA-Z0-9_]+ ;
VALUE       : [0-9]+ ' '* ('%' | 'cent' | 'year');
BINOP       : '+' | '-' | '*' | '/' | '<' | '<=' | '>' | '>=' | '==' ;
```

```
[ plan basic($cpm1, $cpm2) ]
cost =
  if #is_workday
    then (#call_length * $cpm1)
    else (#call_length * $cpm2)

[ plan premium($a, $b) ]
discount =
  if #user_age > 1 year
    then $a
    else $b

[ user 19283 ]
plan = basic(5, 2) + premium(5%, 10%)
```


file
def:2
[plan ctor:2] cost - exp:8
basic ($cpm1 , $cpm2)
if
if exp:4 then exp:1 else
#is_workday (exp:7)
exp:4 * exp:3
#call_length $cpm1
def:2
[plan ctor:2] discount - exp:8
premium ($a , $b)
if
exp:1
(exp:7)
exp:4 * exp:3
#call_length $cpm2
if exp:7 then exp:3 else
exp:4 > exp:2
$a
#user_age 1·year

AST

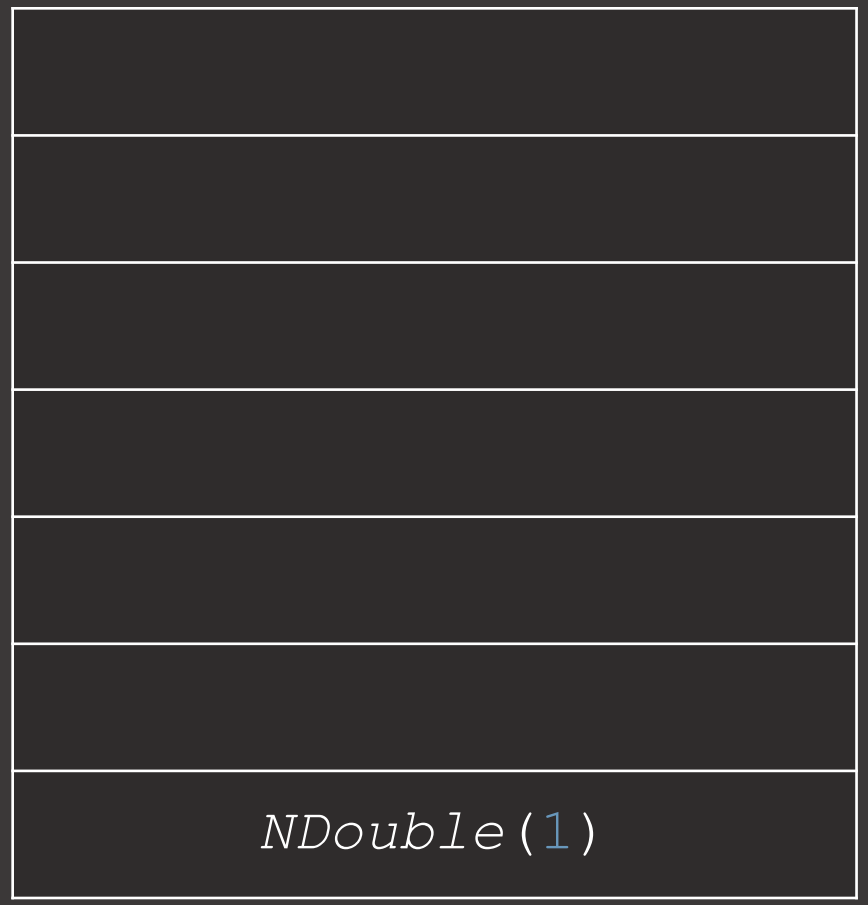
```
[ plan basic($c1, $c2) ]
cost = #call_length *
  if #is_workday
    then $c1
    else if #user_age < 1 year
      then $c2 * 90%
      else $c2
```

```
file
  defplan(“base”, [“c1”, “c2”])
    setattr(“cost”)
      NBinop("*",
        NInspect("call_length"),
        NIf(
          NInspect("is_workday"),
          NAttr("c1"),
          NIf(
            NBinop("<", NInspect("user_age"), NDouble(365)),
            NBinop("*", NDouble(0.9), NAttr("c2")),
            NAttr("c2")
          )
        )
      )
```

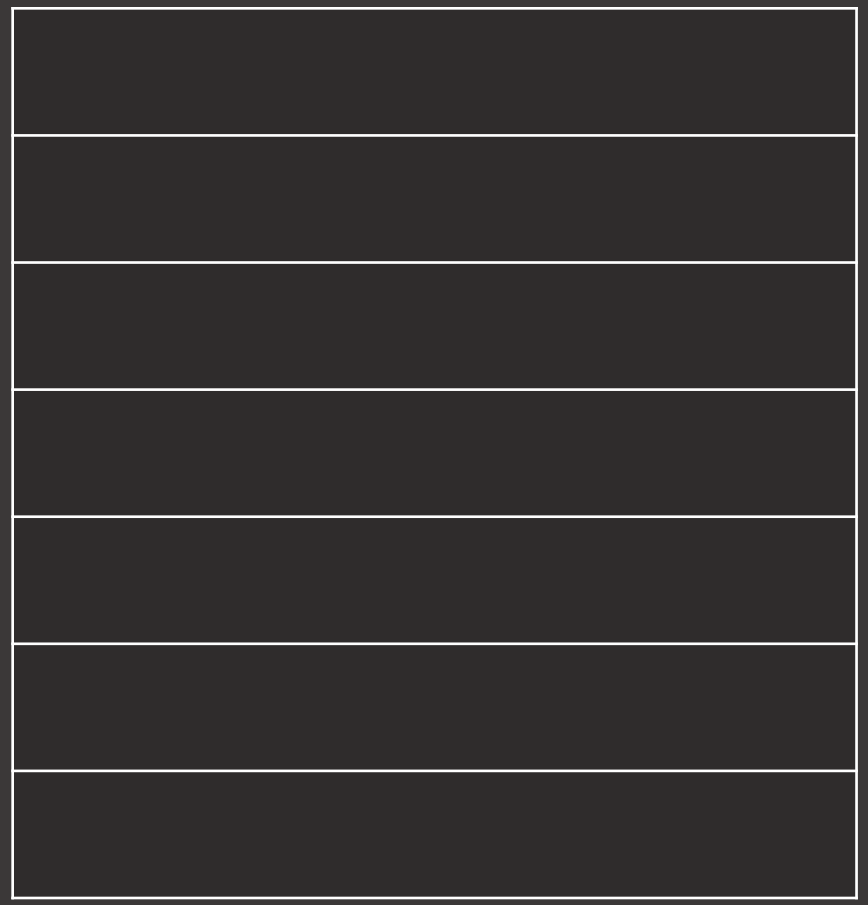
Code execution:

AST traversal + stack

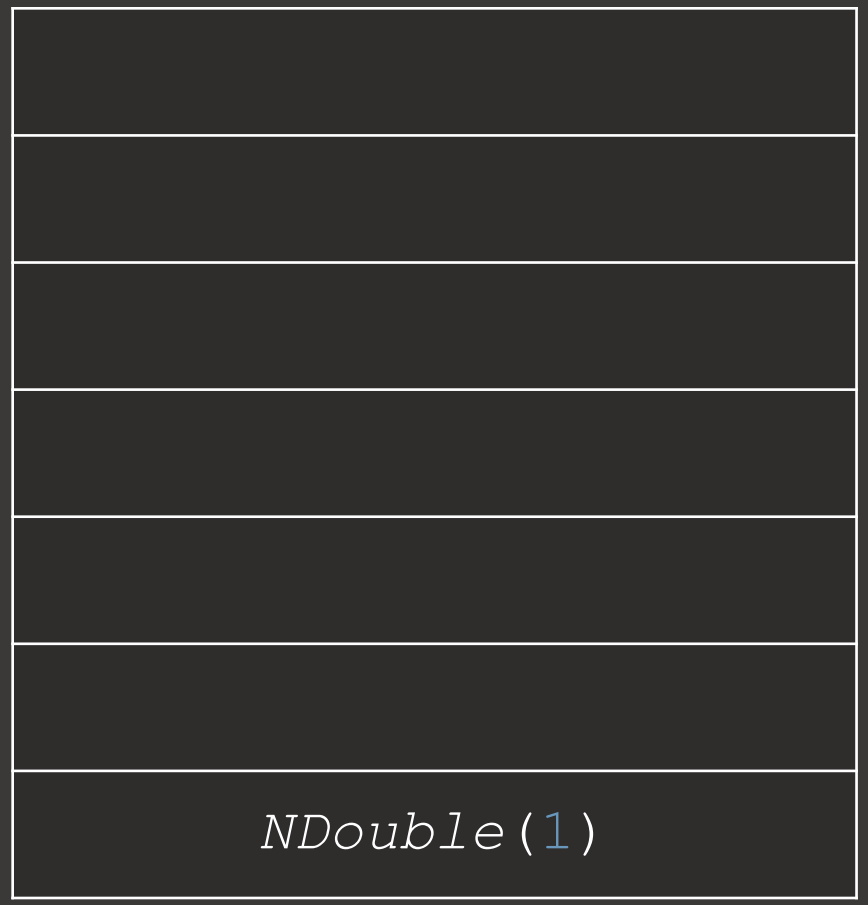
NDouble(1)



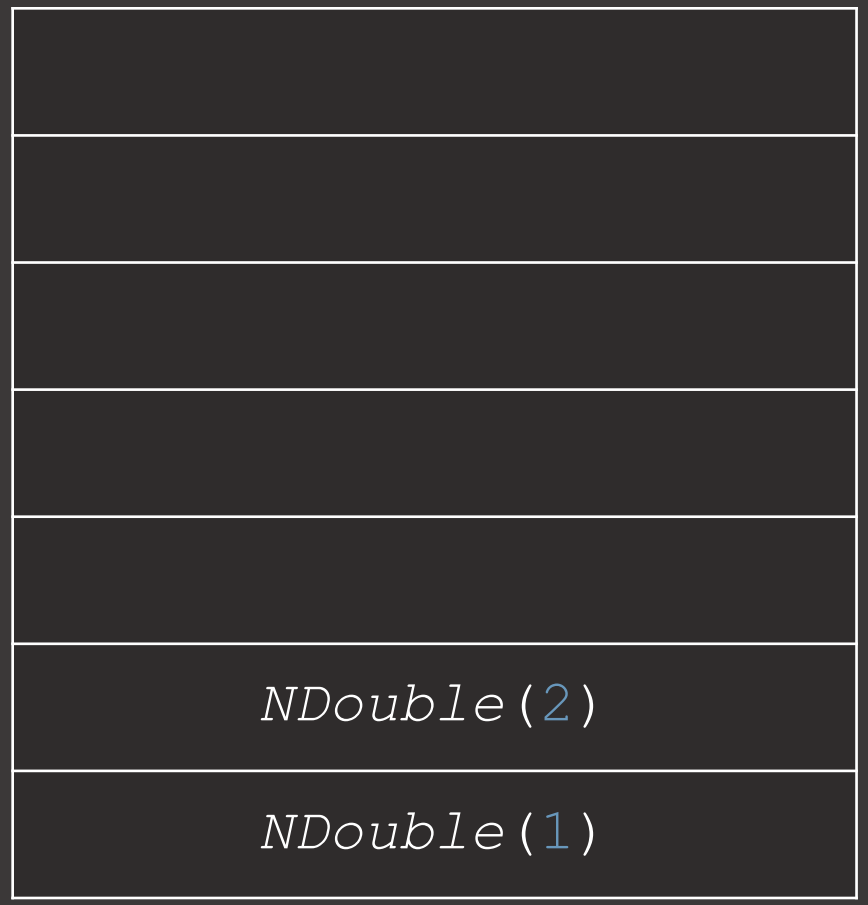
```
NBinop("*",
  NDouble(1),
  NDouble(2)
)
```



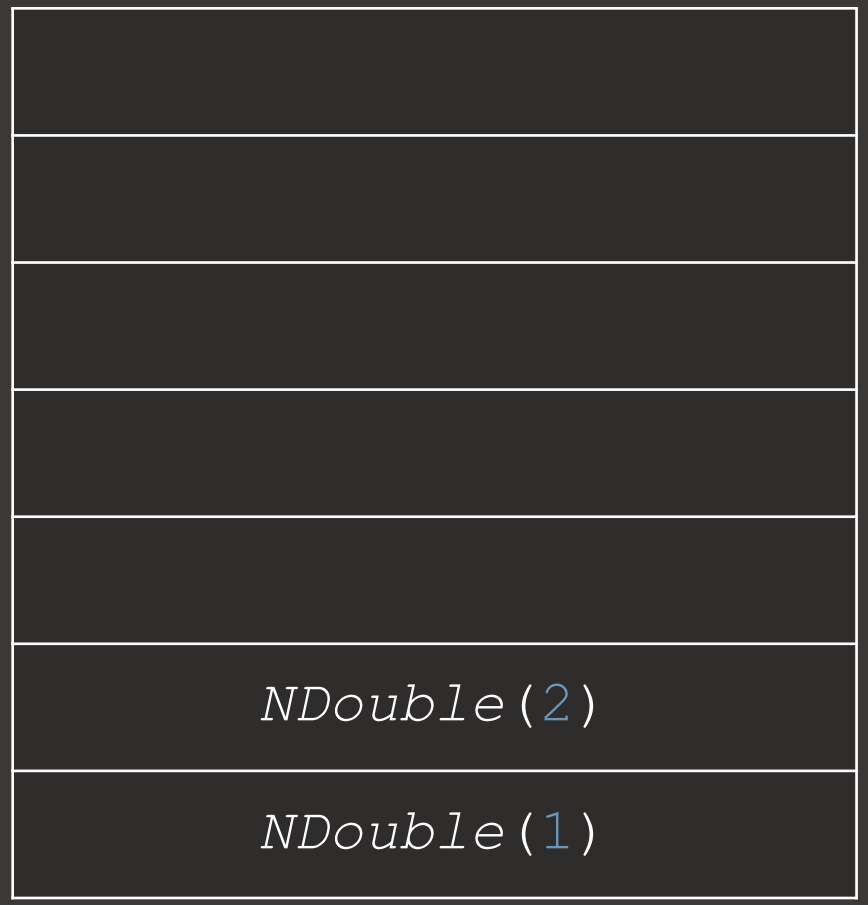
```
NBinop("*",
  NDouble(1),
  NDouble(2)
)
```



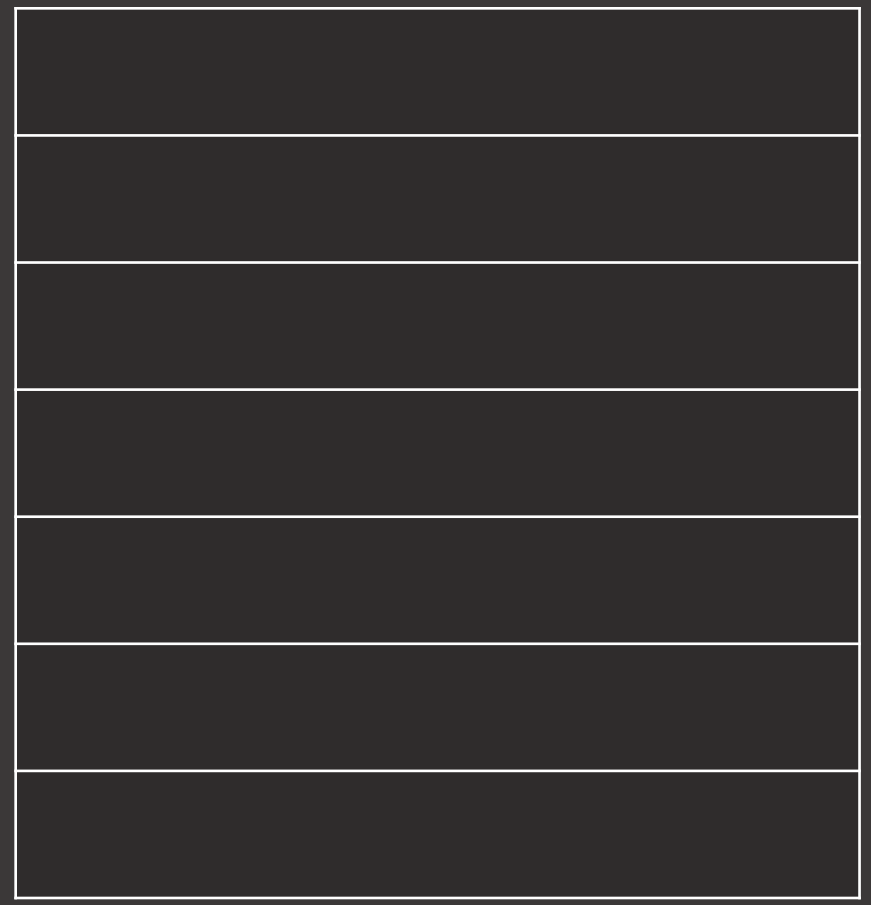
```
NBinop("*",
  NDouble(1),
  NDouble(2)
)
```



```
NBinop("*",
  NDouble(1),
  NDouble(2)
)
```

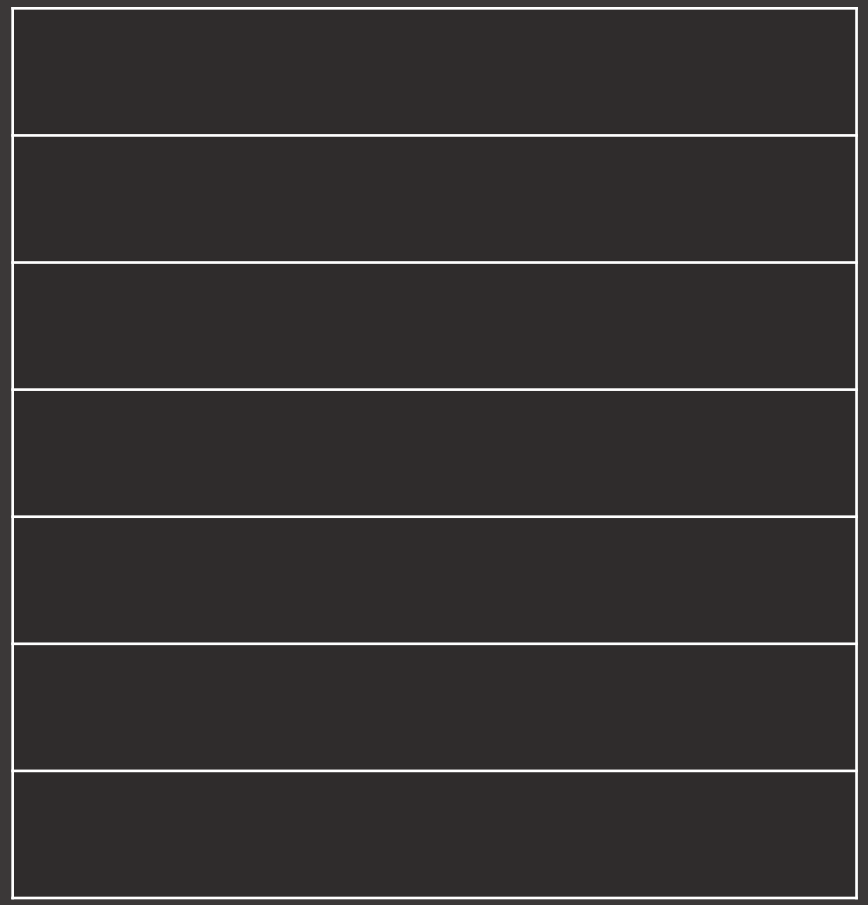


```
NBinop("*",
  NDouble(1),
  NDouble(2)
)
```

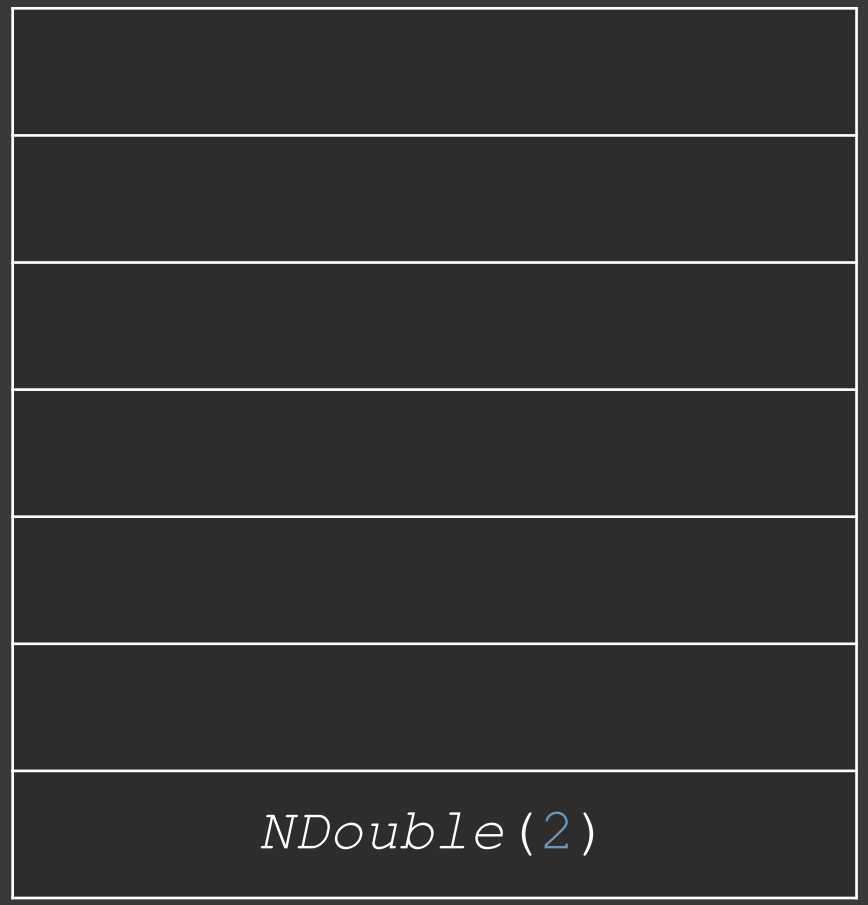


```
NBinop("*",
  NDouble(1),
  NDouble(2)
)
```

NDouble(2)



```
NBinop("*",
  NDouble(1),
  NDouble(2)
)
```



```
Traversing NBinop(*,NInspect(call_length),NIf(NInspect(is_workday),…
Stack before: List()
  Traversing NInspect(call_length)
  Stack before: List()
  Stack after:  List(NDouble(60.0))
  Traversing NIf(NInspect(is_workday),NAttr(c1),NIf(NBinop(<,NInspect(user_age),…
  Stack before: List(NDouble(60.0))
    Traversing NInspect(is_workday)
    Stack before: List(NDouble(60.0))
    Stack after:  List(NBoolean(false), NDouble(60.0))
    Traversing NIf(NBinop(<,NInspect(user_age),NDouble(365.0)),NBinop(*…
    Stack before: List(NDouble(60.0))
      Traversing NBinop(<,NInspect(user_age),NDouble(365.0))
      Stack before: List(NDouble(60.0))
        Traversing NInspect(user_age)
        Stack before: List(NDouble(60.0))
        Stack after:  List(NDouble(30.0), NDouble(60.0))
        Traversing NDouble(365.0)
        Stack before: List(NDouble(30.0), NDouble(60.0))
        Stack after:  List(NDouble(365.0), NDouble(30.0), NDouble(60.0))
      Stack after:  List(NBoolean(false), NDouble(60.0))
      Traversing NAttr(c2)
      Stack before: List(NDouble(60.0))
      Stack after:  List(NDouble(3.0), NDouble(60.0))
    Stack after:  List(NDouble(3.0), NDouble(60.0))
  Stack after:  List(NDouble(3.0), NDouble(60.0))
Stack after:  List(NDouble(180.0))
```

A bot as a distributed language: Zoe

Sending a daily report to the sysadmin

```
{
   intent: send-email
   to: {
      intent: get-email
      user: sysadmin
   }
   message: {
      intent: get-daily-report
   }
}
```

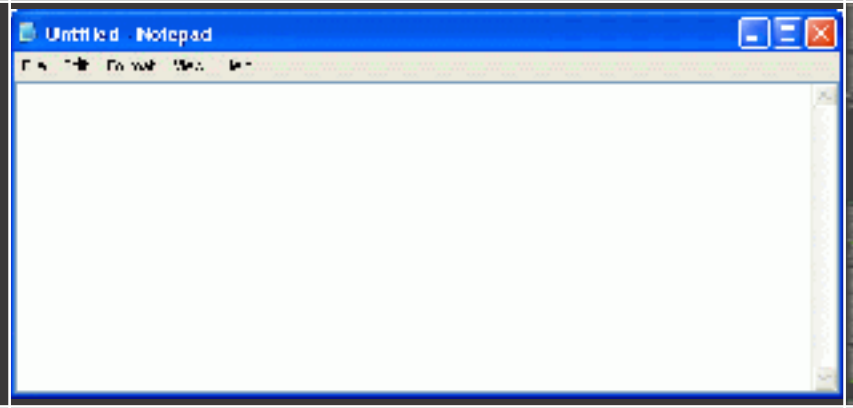
```
{
   intent: send-email
   to: {
      intent: get-email
      user: sysadmin
   }
   message: {
      intent: get-daily-report
   }
}
```

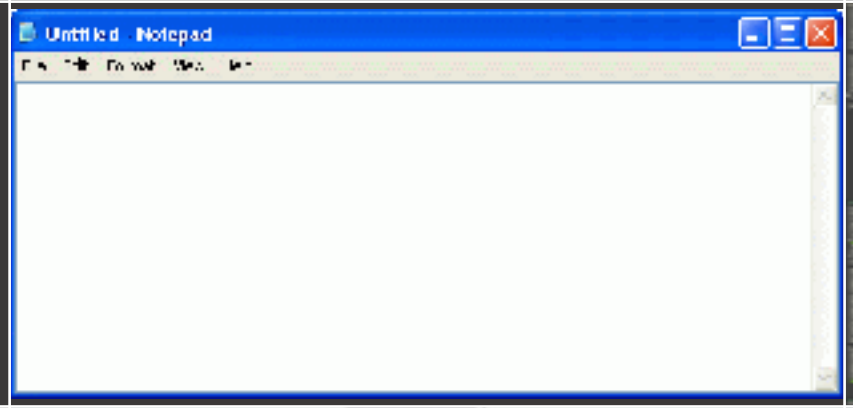
```
{
   intent: send-email
   to: sysadmin@corp.com
   message: {
      intent: get-daily-report
   }
}
```

```
{
   intent: send-email
   to: sysadmin@corp.com
   message: {
      intent: get-daily-report
   }
}
```

```
{
  intent: send-email
  to: sysadmin@corp.com
  message: “Every little thing is OK”
}
```

```
{
  intent: send-email
  to: sysadmin@corp.com
  message: “Every little thing is OK”
}
```

	Before	After the Moment of Illumination and acceptance of the Hacker Way
Product		
You		

	Before	After the Moment of Illumination and acceptance of the Hacker Way
Product		
You		

Thanks!

david@seedtag.com 
@voiser