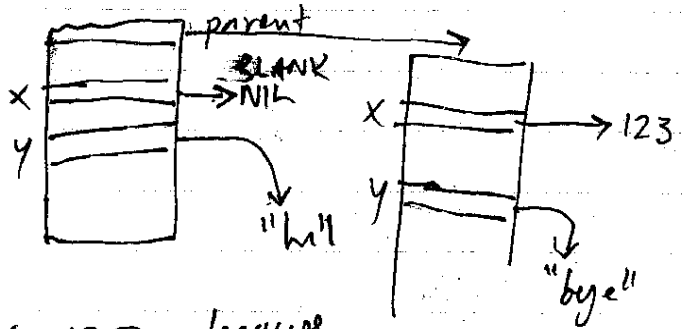
Slate is complicated by its multiple-dispatch semantics.

Slate objects have
- header (gc)
- "traits"
- "map"
:
slots

I suspect "javascript + more" are embedding-style prototype languages. The method (slot) lookup seems different in Slate (which is a delegating-style language).

JS.



x = 123 — because slots blank.
y = "hi"

x = 123
y = "bye"

Slots are an implementation property, not an interface property.

⇒ slots to be visible only to methods.

Interesting that slots have "protected" visibility in ST/Squeak, cf. the expected "private" visibility.

What is the algorithm for slot/method lookup in delegating languages??

abcdef

Given some process eg $\mu B.\ x?nc.(c!ht \mid B)$.
How to lift it?

6:

nu	$\nu X.P$
par	$P \mid P$
rec	$\mu A.P$
ref	A
out	$x!y$
in	$x?y.P$

$\ulcorner \nu x.P \urcorner_k \to \nu q.\ k\,q.\ \mu A.\ q?abcdef.\ a!$

k needs to be a function (at metalevel) for proper CPS.

nu	$\ulcorner \nu x.P \urcorner k$	=	$\nu q.\ k(q) \mid \mu A.\ q?abcdef.(\ulcorner P \urcorner \lambda p.\ a!xp \mid A)$
par	$\ulcorner P_1 \mid P_2 \urcorner k$	=	$\nu q.\ k(q) \mid \mu A.\ q?abcdef.(\ulcorner P_1 \urcorner \lambda p_1.\ \ulcorner P_2 \urcorner \lambda p_2.\ b!p_1p_2 \mid A)$
rec	$\mu A.P$	=	" " " " $.(\ulcorner P \urcorner \lambda p.\ c!A'p \mid A)$
ref	A'	=	" " " " $.(d!A' \mid A)$
out	$x!yz$	=	" " $.(e!x(\text{cons } y \ (\text{cons } z \ \text{NIL})) \mid A)$
in	$x?yz.P$	=	" " $(\ulcorner P \urcorner \lambda p.\ f!x(\text{cons } y \ (\text{cons } z \ \text{NIL}))p \mid A)$

$\ulcorner \urcorner k$ $\quad k = \lambda p. \cdots$ $\quad$ name → process

name ⇄ process

$\ulcorner P \urcorner \triangleq \ulcorner P \urcorner \lambda p.p$ $\quad \ulcorner \urcorner$: process → name

Assume a traditional ST80-style dispatcher.
with classes rather than prototypes

Then lifted processes have a visitor pattern.

Class methods?
Vars a namespace!

$\ulcorner \nu x.P \urcorner k = \ulcorner P \urcorner \lambda p.\ \nu q[x,p].\ k(q) \mid \mu A.\ q?^{\text{cont. msg. args}}.\ NuP!^{\text{cont msg args}}q$

$\nu q[x,p].\ k(q) \mid \text{ST80Object}(NuP, q)$ X

$\nu q[NuP, x, p].\ k(q) \mid \text{ST80Object}(q)$

~~ST80Object = $\lambda o.\ \mu h.o?\ k, m, args.\ A \mid$ MEMOIZE~~

ST80Object = $\lambda o.\ \mu A.\ o ::= \Lambda(A, m, args).\ meth \leftarrow \text{MEMOIZE}(\text{ST80Lookup}(o[0], m))$
$meth(o, args)$

$\mu A.$ ST80Lookup $::= \lambda m.\ \mu A.\ \Lambda(A, c, m).\ dict \leftarrow c[1].\ \cdots\ meth.$

~~$\mu A.$ SendToSuper = $\lambda m\ A.\ \Lambda(A,$~~

SendToSuper = $\lambda m A.\ \Lambda(A, m, args).\ meth \leftarrow \text{MEMOIZE}(\text{ST80Lookup}(c, m)).meth(o, args)$