When Two Languages Are Simpler Than One

Lessons for SES from Cajita, Original-Caja, and Valija

Mark S. Miller

Simultaneous Problems

D = Defensive code problem
O = Offensive code problem
T = Legacy tools problem
C = Legacy code problem

Simultaneous Solution?

D = Defensive code problem
O = Offensive code problem
T = Legacy tools problem
C = Legacy code problem

Original-Caja	dOTc	Secure Linux/Windows
Cajita	DOT	Secure microkernel OS

Don’t try this at home (or at all)

D = Defensive code problem
O = Offensive code problem
T = Legacy tools problem
C = Legacy code problem

~~Original Caja~~	~~dOTc~~	~~Secure Linux/Windows~~
Cajita	DOT	Secure microkernel OS

Separate Solutions

D = Defensive code problem
O = Offensive code problem
T = Legacy tools problem
C = Legacy code problem

Cajita	DOT	Secure microkernel OS
Valija	OTC	Virtual Machine

Layered Solutions

D = Defensive code problem
O = Offensive code problem
T = Legacy tools problem
C = Legacy code problem
V = Virtualizability problem

	Cajita*	DOT V	Secure microkernel OS
	Valija	OTC	Virtual Machine
+	Valija on Cajita	DOTCV	VMM + policy glue logic

Lessons for SES

D = Defensive code problem
O = Offensive code problem
T = Legacy tools problem
C = Legacy code problem
V = Virtualizability problem

	SES	DOT V	Secure microkernel OS
	~Harmony-strict	OTC	Virtual Machine
+	Safer scripting	DOTCV	VMM + policy glue logic

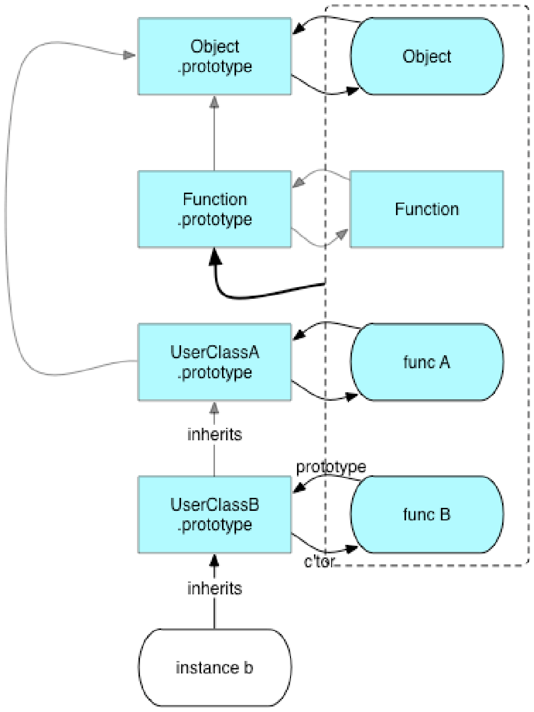
Proposed SES Goals

- SES is smallest secure subset of ~Harmony-strict without loss of functionality.
- SES is a good target for a multiply instantiable embedding of ~Harmony-strict.

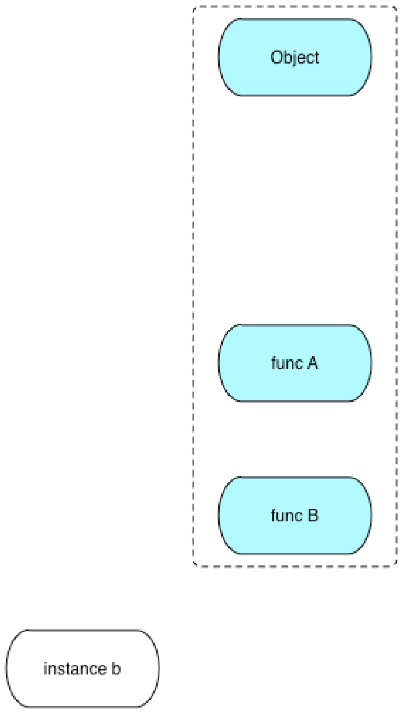
	SES	DOT V	Secure microkernel OS
	~Harmony-strict	OTC	Virtual Machine
+	Safer scripting	DOTCV	VMM + policy glue logic

Questions?

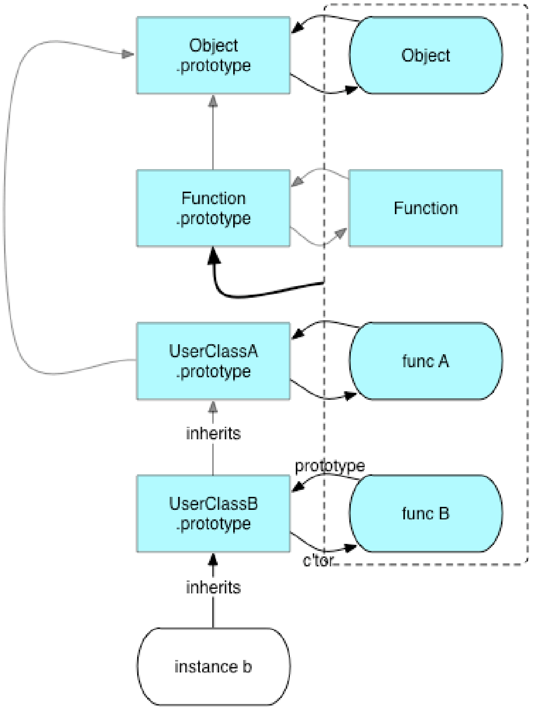
Freeze Primordials



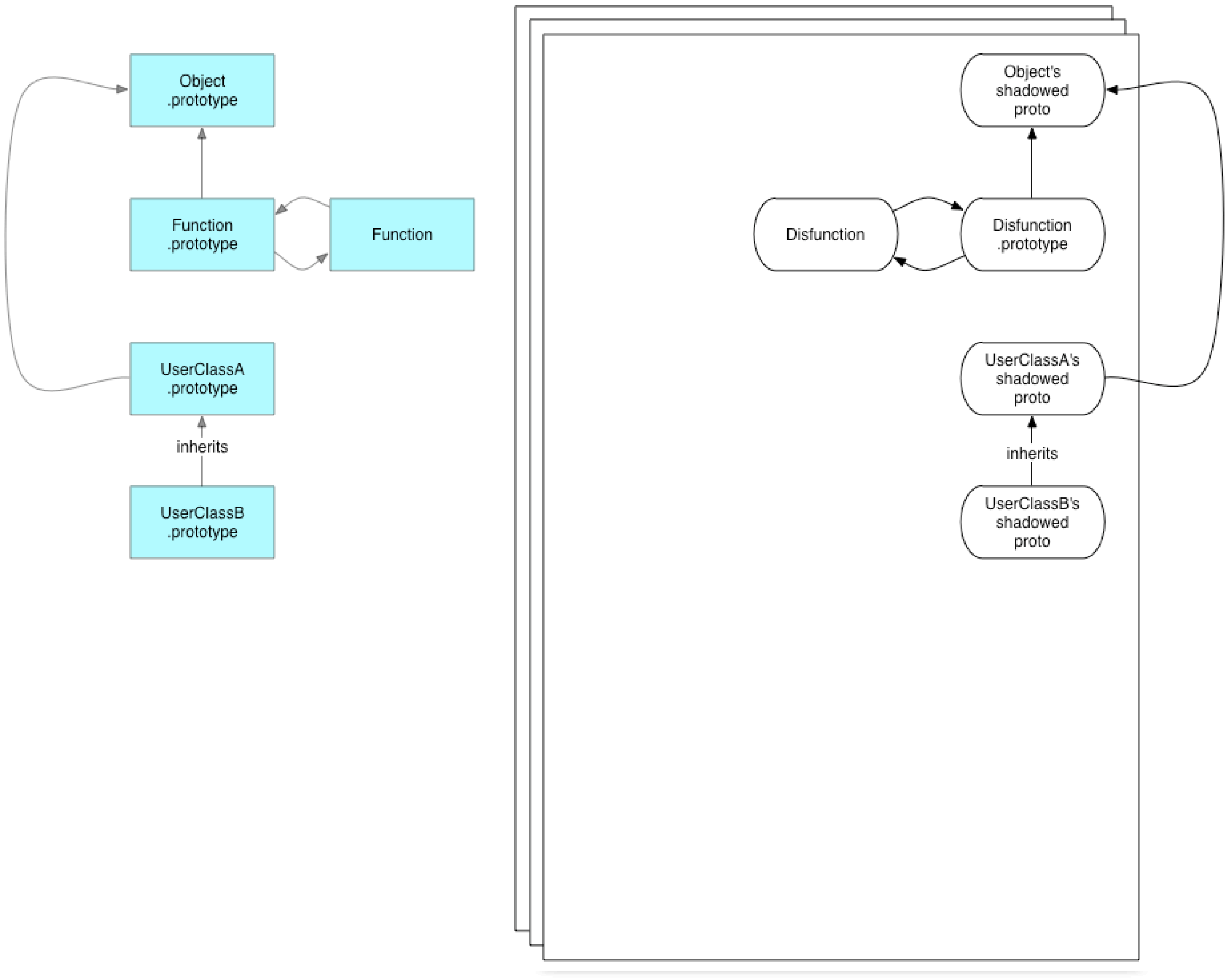
Hide Sharp Objects = Cajita



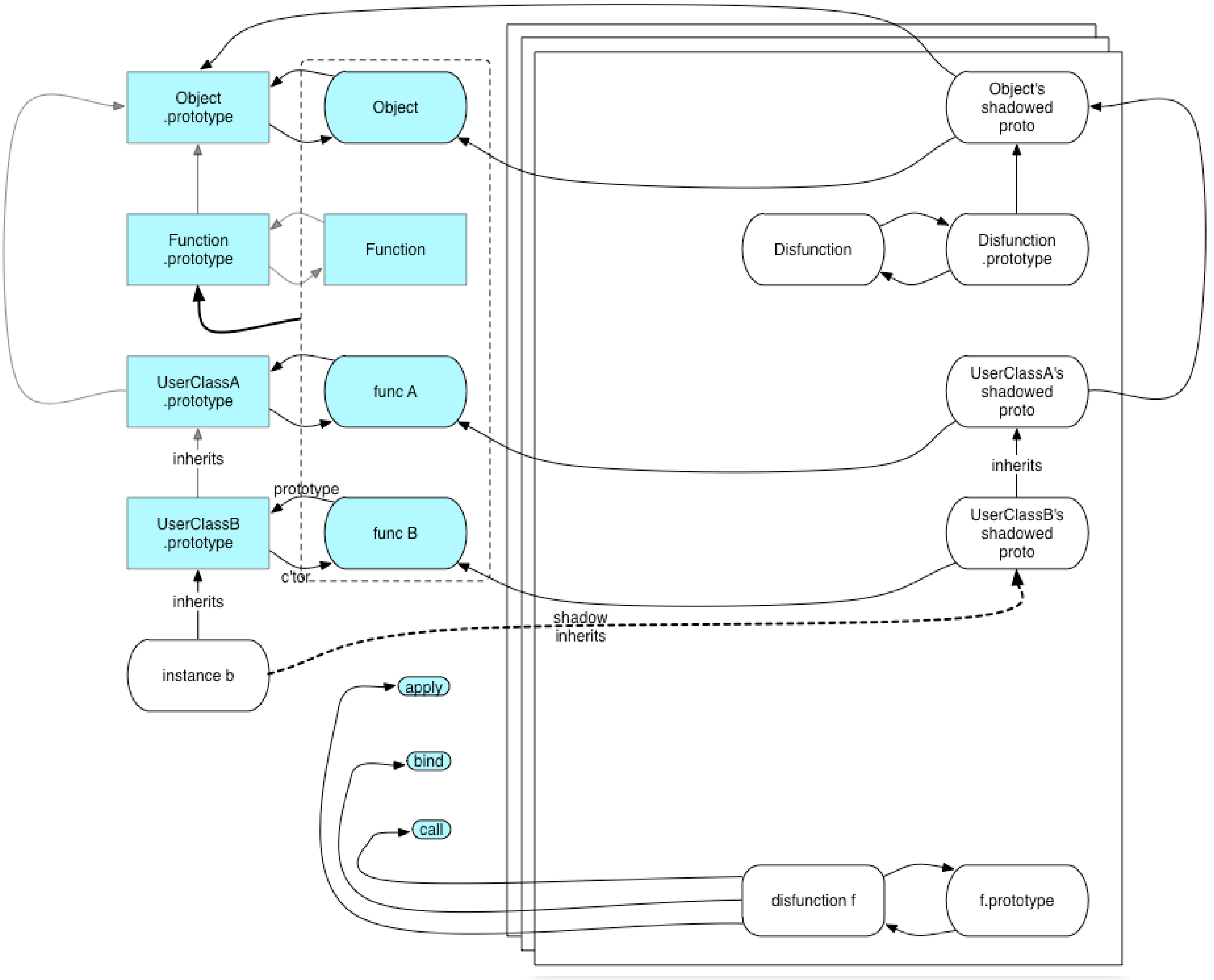
Cajita + Implementation



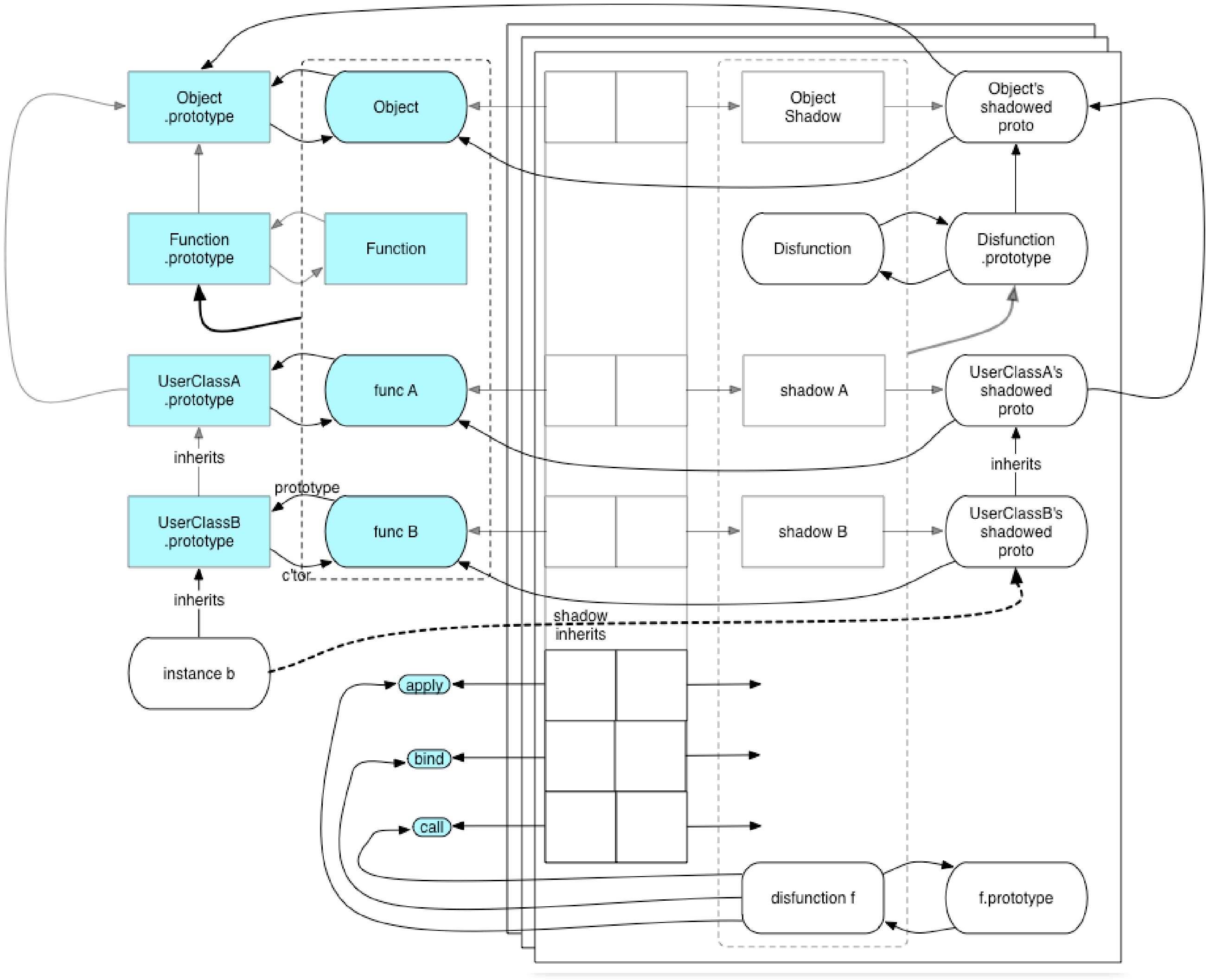
Replace with per-gadget toy knives



Valija on Cajita Impl



Valija Impl on Cajita Impl


Object .prototype
Object
Object Shadow
Object's shadowed proto
Function .prototype
Function
Disfunction
Disfunction .prototype
UserClassA .prototype
func A
shadow A
UserClassA's shadowed proto
inherits
prototype
UserClassB .prototype
func B
shadow B
UserClassB's shadowed proto
inherits
c'tor
inherits
shadow inherits
instance b
apply
bind
call
disfunction f
f.prototype