In Praise of Metacircular Virtual Machine Layering

Erick Lavoie
Université de Montréal
ericklavoie.com

Instrumenting JavaScript at run-time

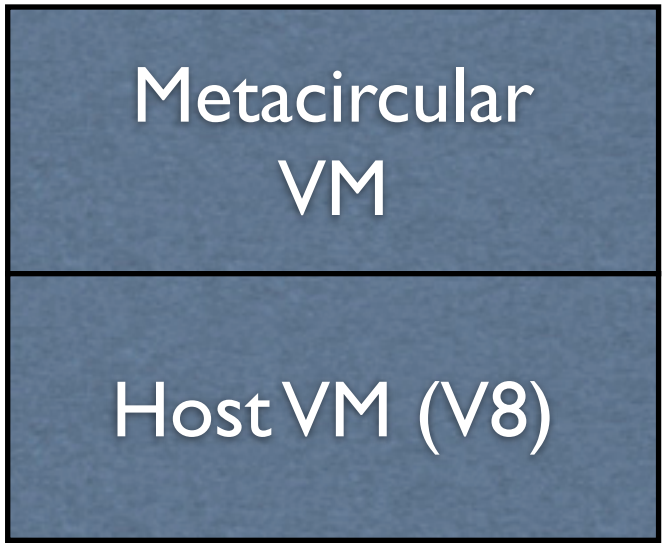
- Goal:
  - Obtaining information about the behavior of programs while they are executing
- Relevant for:
  - Performance profiling
  - Benchmarking
  - Security invariants monitoring

Direct Instrumentation of Browser VMs

- Compliant
- Integrated with the browser
- Straightforward on simple interpreters
- Need maintainance for long-term projects
- Harder on optimized implementations
- Restricted to a single browser

Metacircular VM Layering

- "Differential" implementation
- Independent from Host VM implementation
- Performance cost

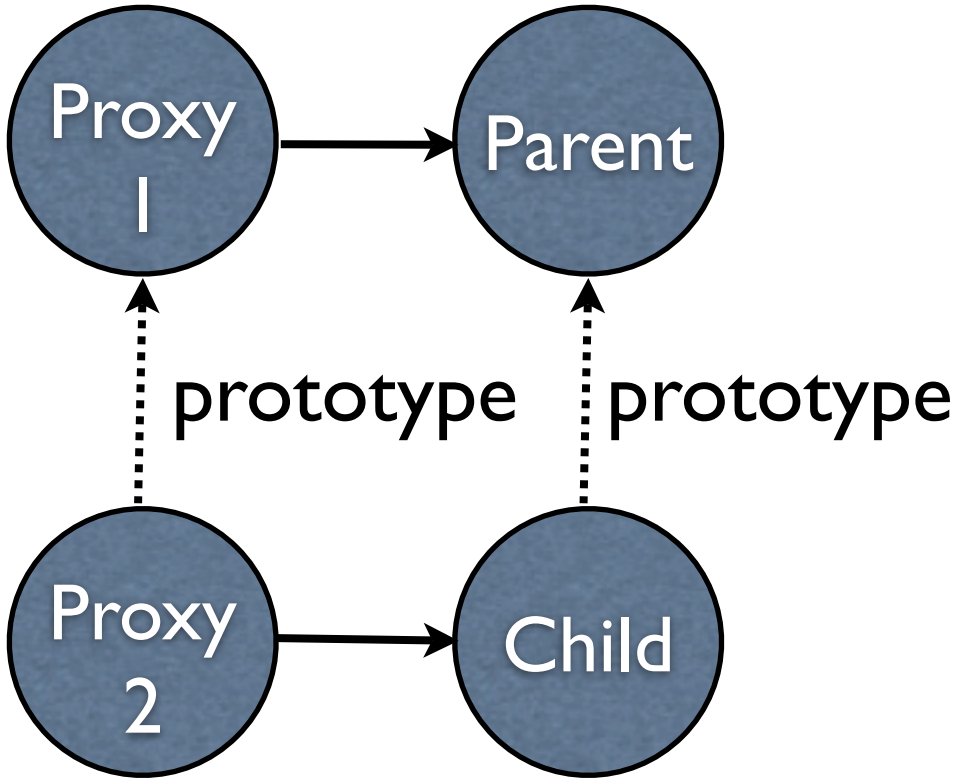


Reifications for Run-Time Instrumentation

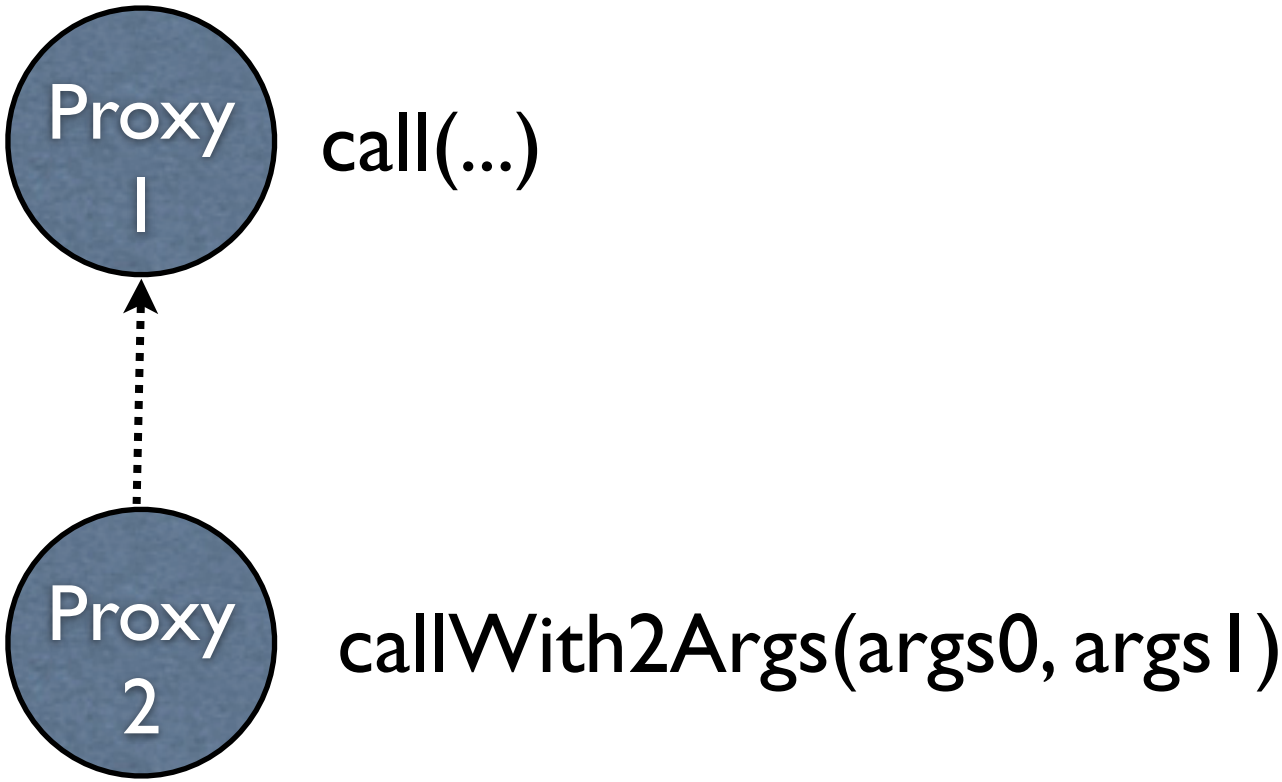
- Object Model operations:
  - property accesses, updates and deletions
  - object creations
- Function calls:
  - global, local and method calls
  - causally connected to *Function.prototype.call*

Exploiting Host VM Optimizations

Proxy native objects *while mirroring the prototype chain of native objects in proxies*



Optimization Example



Empirical Results

- Baseline performance on V8 benchmarks:
  - 2 times slower than SpiderMonkey(no JIT)
  - 52 times slower than V8
- Baseline memory usage:
  - Most benchmarks less than 3 times
  - Worst cases of 6.4 times
- Instrumentation of property operations can have a negligible additional overhead

Conclusion

- Metacircular VM Layering in JavaScript can:
  - instrument object model operations and functions calls
  - substitute direct instrumentation of browser VMs for interpreter-level performance
  - be simple (runtime library is 1700 lines of JavaScript, excluding JS-to-JS compiler)

Further Readings

- Article submitted for RESoLVE'13 workshop (under review)
  - ericklavoie.com/pubs/RESoLVE13.pdf
- Master Dissertation (under review)
  - ericklavoie.com/master/dissertation.pdf
- Prototype implementation
  - To Be Released