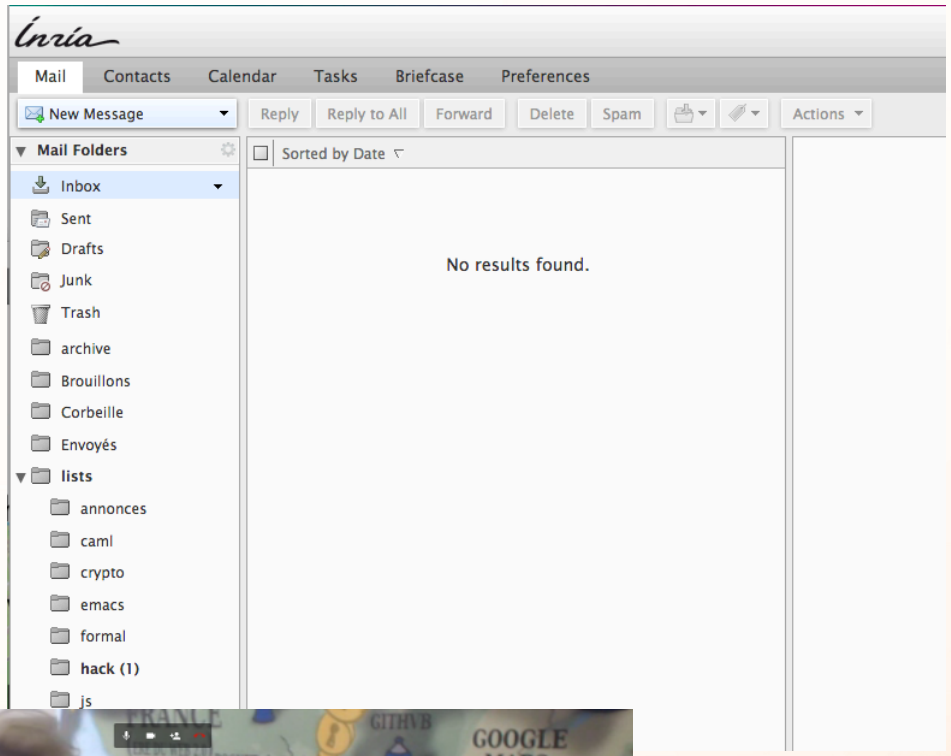
ComInLabs

SecCloud

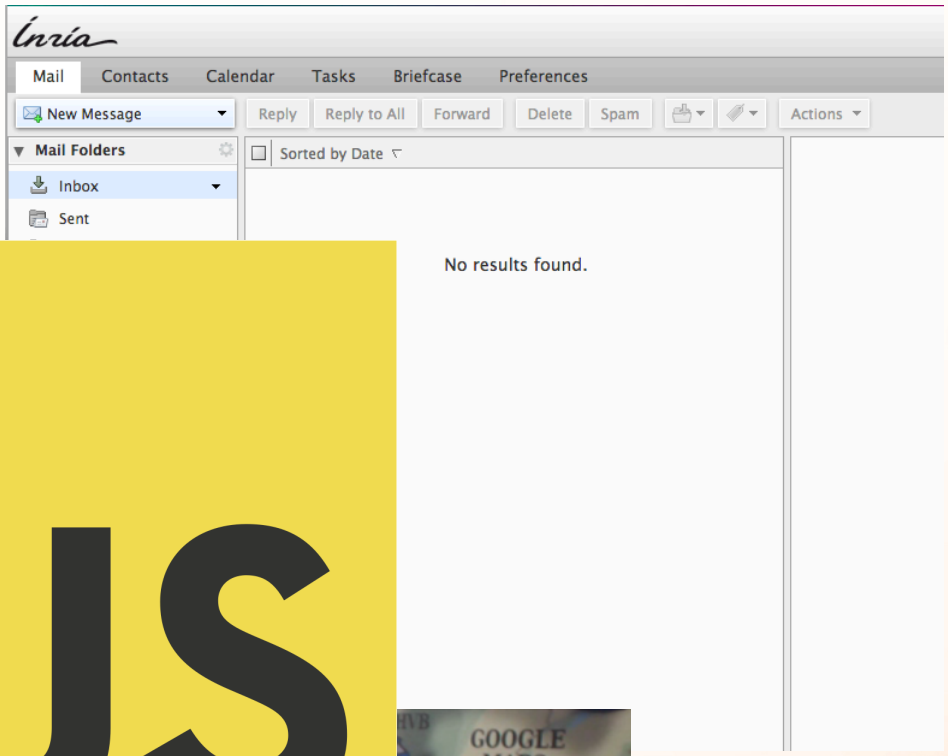
Alan Schmitt

May 28, 2018

Web Applications are like Magic



Web Applications are like Magic



Data Control and Web Applications

Our Data

The Browser

Programs from the Web

Non Interference

direct flow

```
var public = secret
```

indirect flow

```
if (secret) {
  public = true
} else {
  public = false
}
```

Non interference is a hyperproperty

```
/* Source */
var x = true
var y = true
if (secret) {
  x = false
}
if (x) {
  y = false
}
public = y
```

```
/* assume secret is true */
var x = true
var y = true
if (secret) {
  x = false
}

public = y
```

```
/* assume secret is false */
var x = true
var y = true


if (x) {
  y = false
}
public = y
```

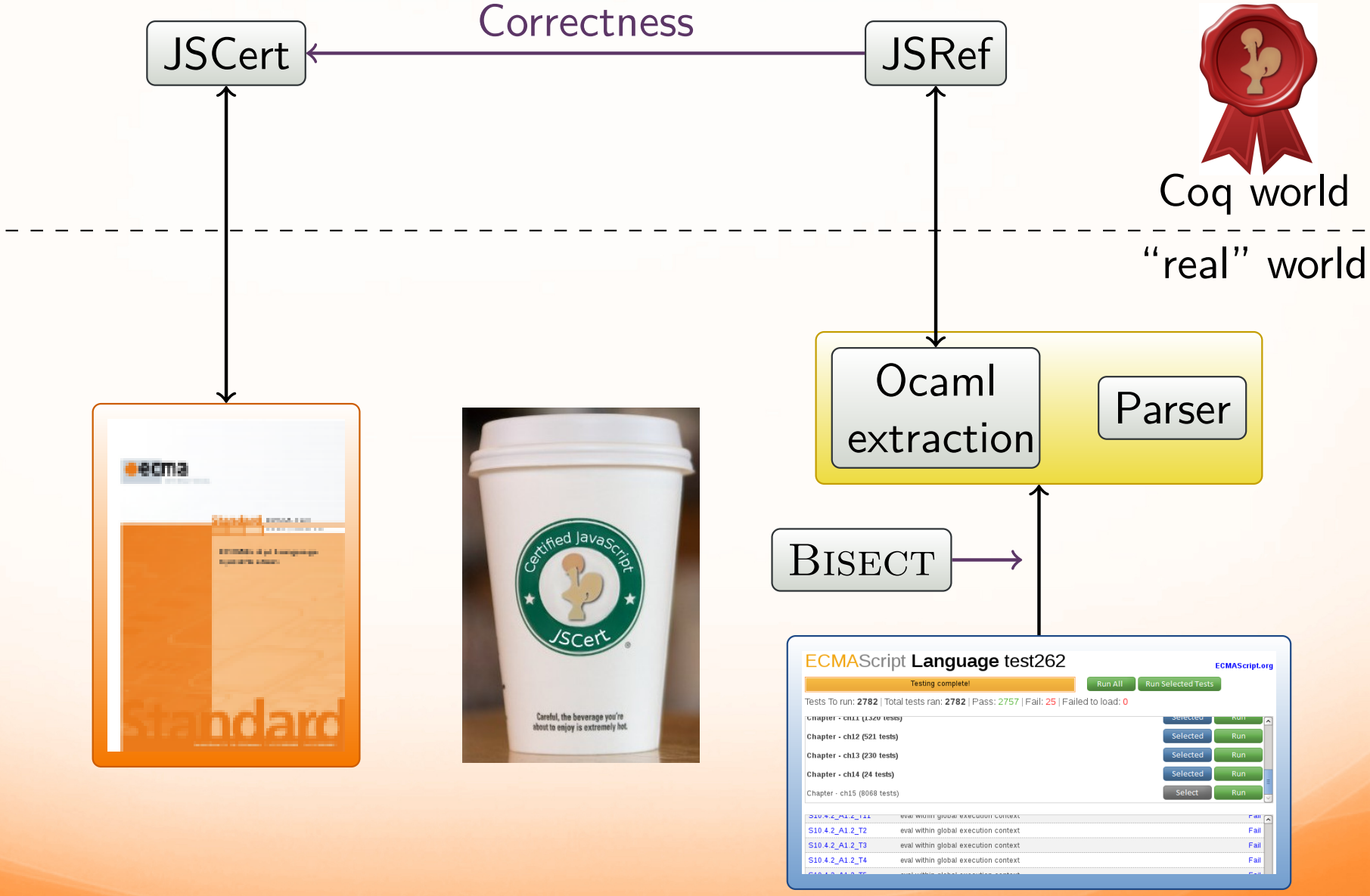
The SecCloud Project

CominLabs

A comprehensive language-based approach to the definition, analysis, and implementation of secure applications developed using JavaScript.

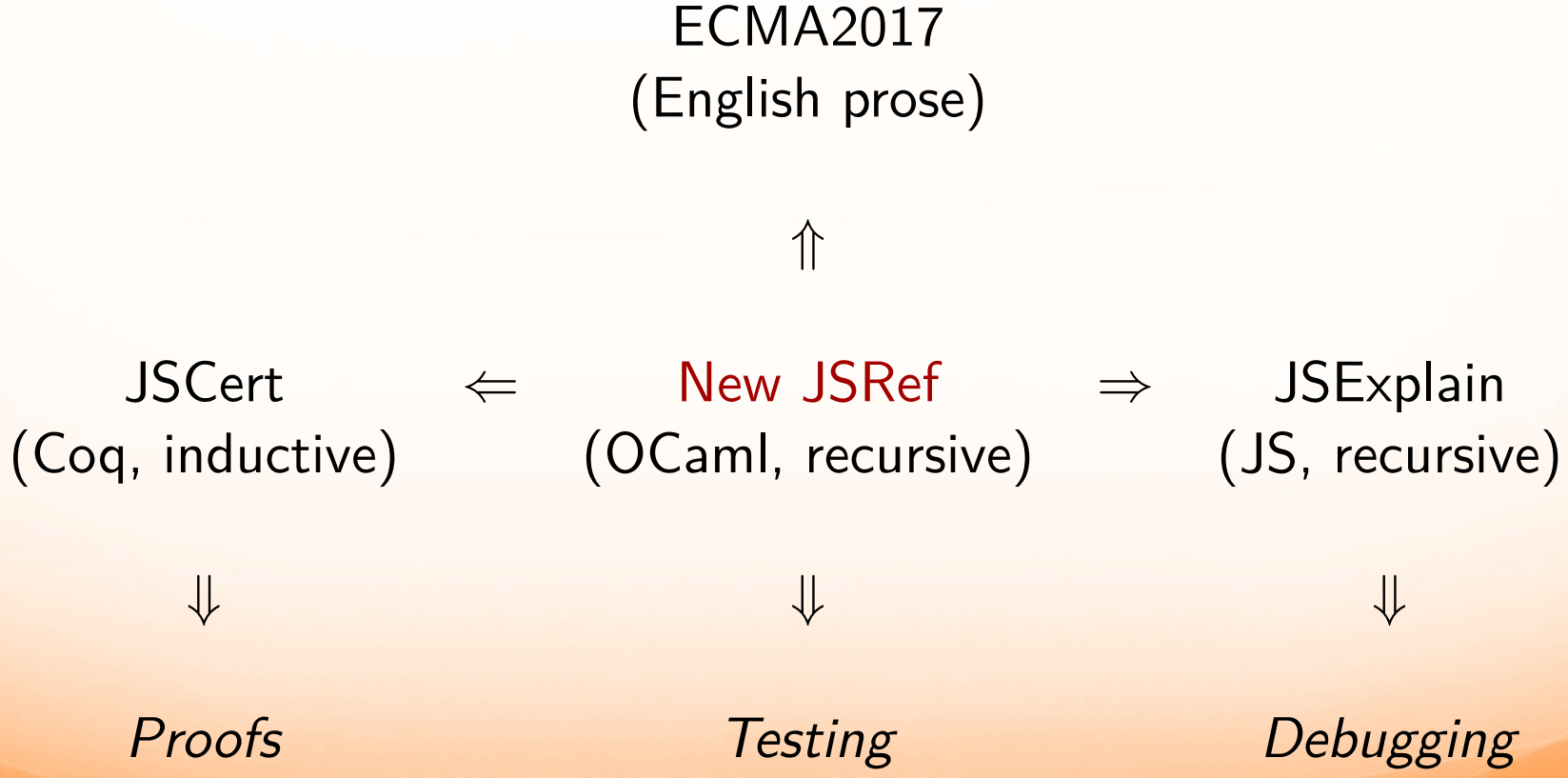
1. formal semantics of JavaScript
2. static and dynamic analyses
3. preventive information flow control

JSCert

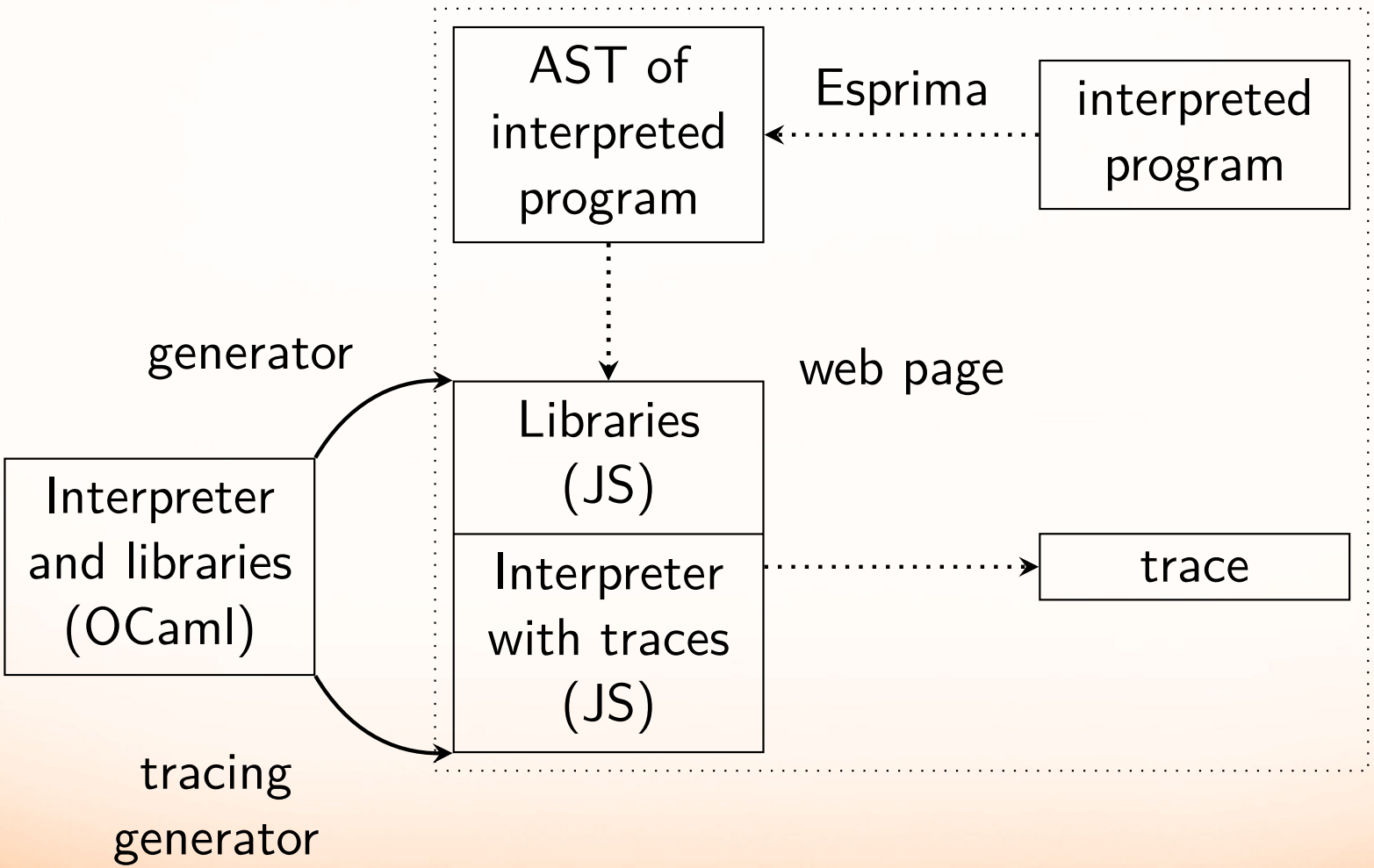

JSCert
Correctness
JSRef
Coq world
"real" world
Ocaml extraction
Parser
BISECT
ecma
Standard
Careful, the beverage you're about to enjoy is extremely hot.
ECMAScript Language test262
ECMAScript.org
Testing complete!
Run All
Run Selected Tests
Tests To run: 2782 | Total tests ran: 2782 | Pass: 2757 | Fail: 25 | Failed to load: 0
Chapter - ch12 (521 tests)
Chapter - ch13 (230 tests)
Chapter - ch14 (24 tests)
Chapter - ch15 (8068 tests)
Selected
Run
Select
S10.4.2_A1.2_T2
S10.4.2_A1.2_T3
S10.4.2_A1.2_T4
eval within global execution context
Fail

Lessons from JSCert

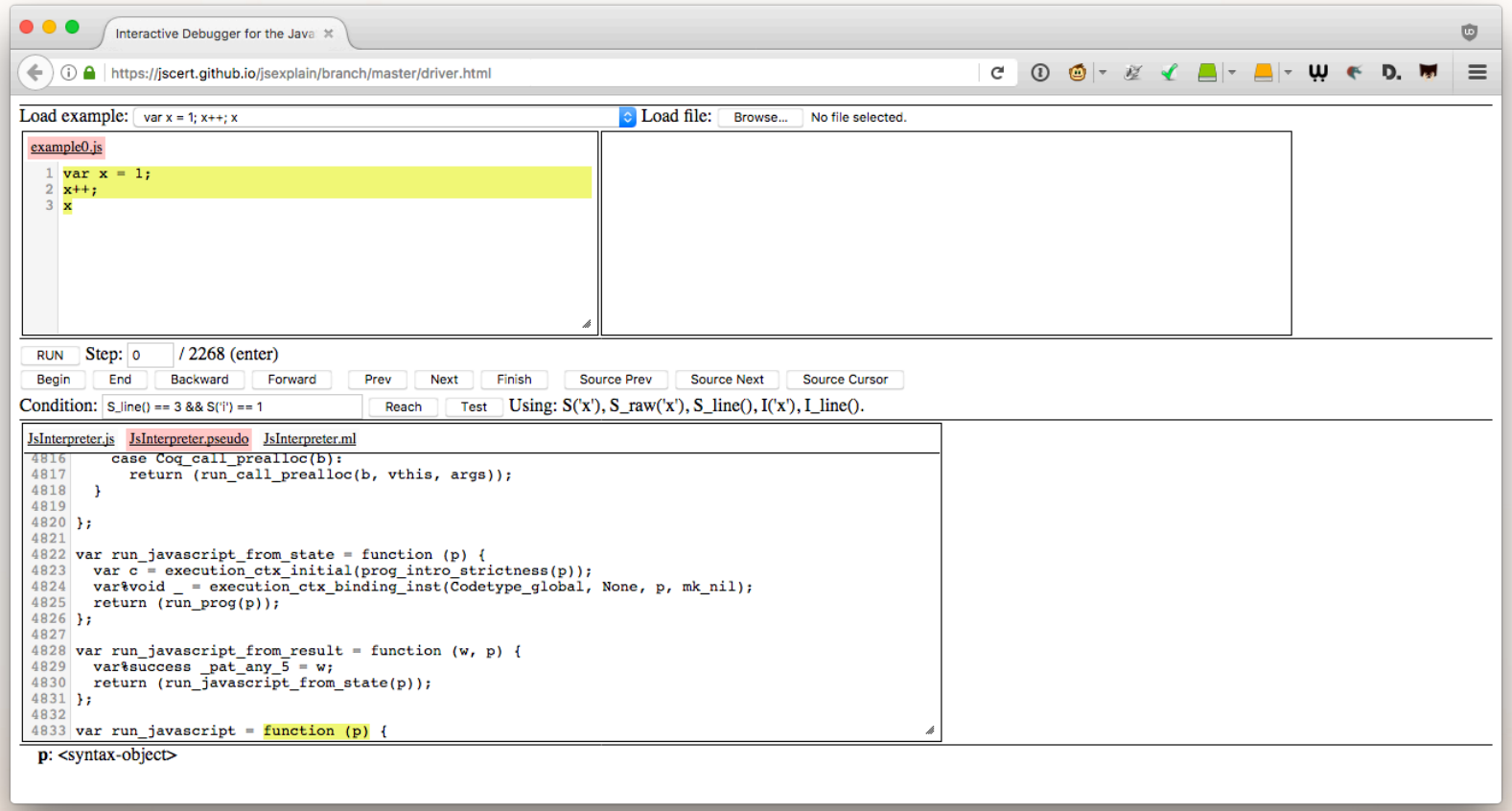
Hard to keep pace with the standardisation
JSCert inductive definition is too big
A implementation close to the spec is very useful



JSExplain


AST of interpreted program
Esprima
interpreted program
generator
web page
Libraries (JS)
Interpreter and libraries (OCaml)
Interpreter with traces (JS)
trace
tracing generator

JSExplain



https://jscert.github.io/jsexplain/branch/master/driver.html

Static and Dynamic Analyses

Static analyses: before running the program

- pro considers the whole program
- con may be less precise

Dynamic analyses: as the program runs

- pro sees only code that runs, access to exact values
- con does not capture every information flow

Hybrid analyses: combine both

Hybrid Monitoring

Hybrid Monitoring

Hybrid Information Flow Monitoring Against Web Tracking

Frédéric Besson, Nataliia Bielova, and Thomas Jensen
Inria
Rennes, France

Browser Randomisation against Fingerprinting: a Quantitative Information Flow Approach

Frédéric Besson, Nataliia Bielova, and Thomas Jensen*

Inria, France

Hybrid Monitoring of Attacker Knowledge

Frédéric Besson, Nataliia Bielova and Thomas Jensen
Inria, France

Multi-semantics

Given a huge formal semantics, how to prove non-interference?
Solution transform a hyperproperty (of the semantics) into a simple property (of the multi-semantics)

Theorem

If a program is interferent, then there exists a derivation in the annotated multi-semantics that witnesses it.

Preventive Information Leaks

facets: values with several values (e.g. `private` and `public`)
faceted evaluation:

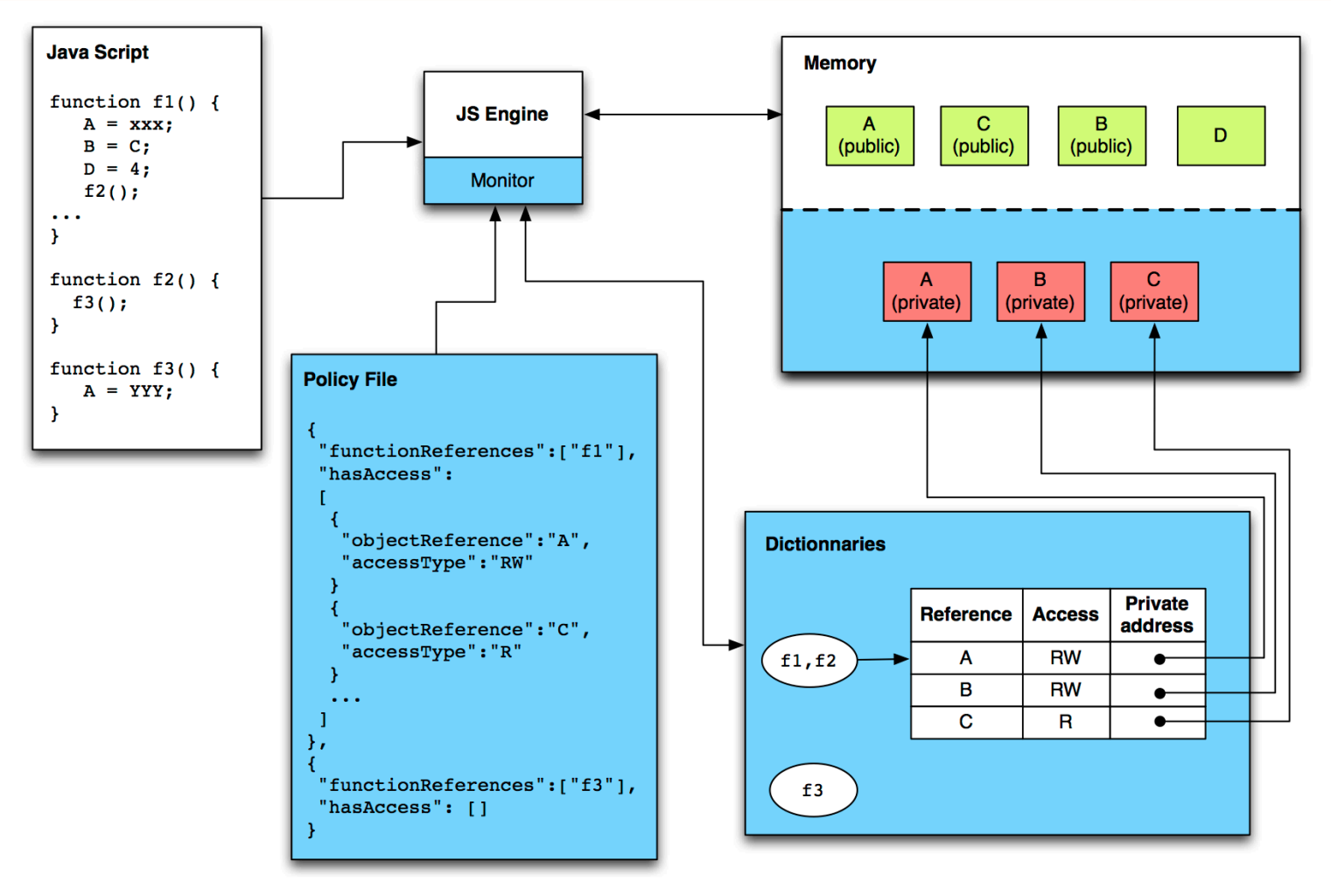
```
function(x)        x: ⟨a, true, false⟩
  y = true         y: true
  z = true         z: true
  if (x)              ⟨PC, a⟩
    y = false      y: ⟨a, false, true⟩
  if (y)              ⟨PC, ā⟩
    z = false      z: ⟨a, true, false⟩
  return z
```

Credits: Florent Marchand de Kerchove

experimentation by extending Narcissus

Split addresses

Change the address of references depending on the execution stack



Implementation in the Chromium V8 engine

Conclusion and Future Work

ComInLabs

Highlights

- formalization of the full JavaScript language
- analyses proven in Coq
- practical tools

Future

- transfer to TC39
- usable formalization of JavaScript
- extension to other languages (Hop.js)