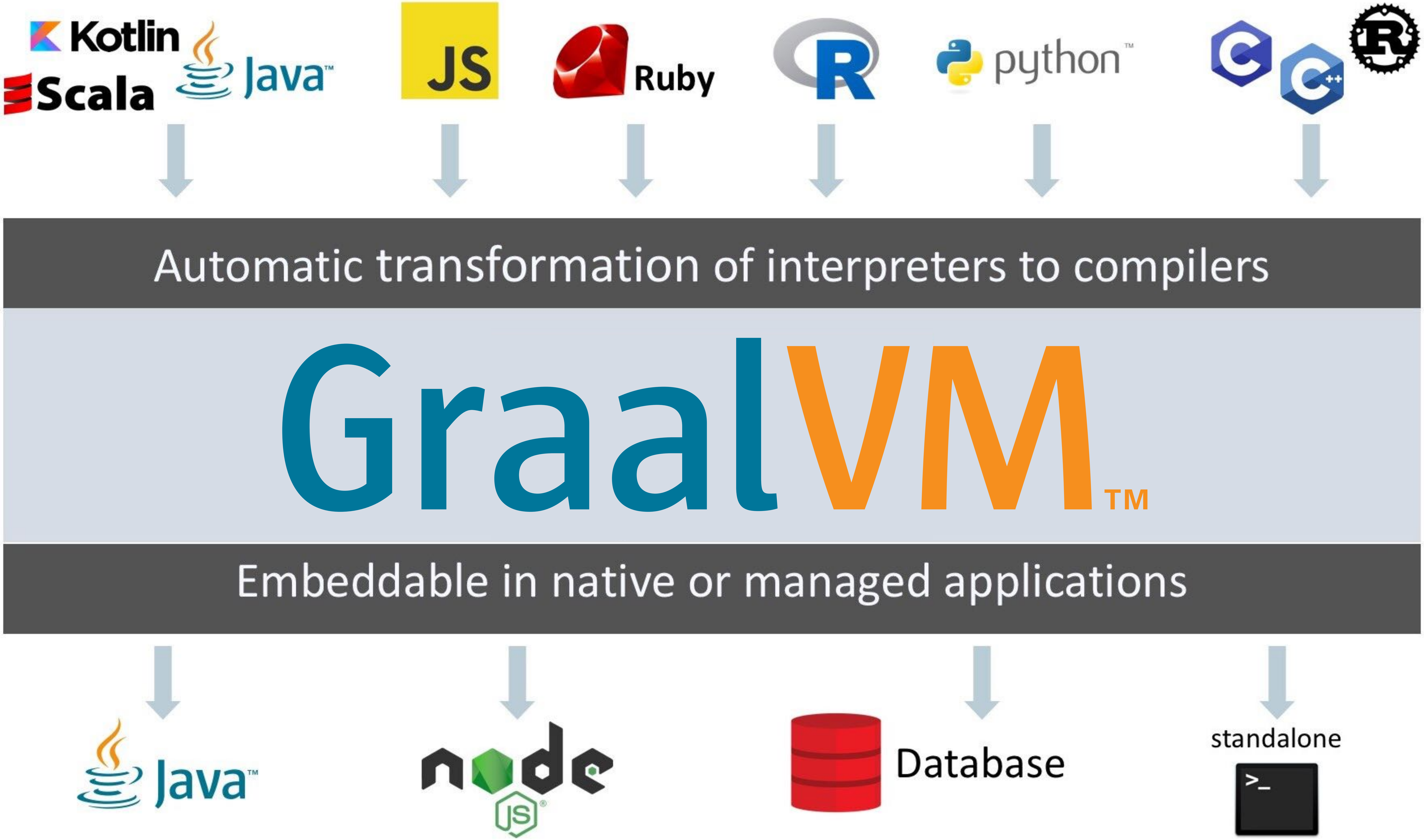
Node.js
Just as fast, higher, stronger with GraalVM™

Oleg Šelajev
Oracle Labs
@shelajev

ORACLE

Safe Harbor Statement

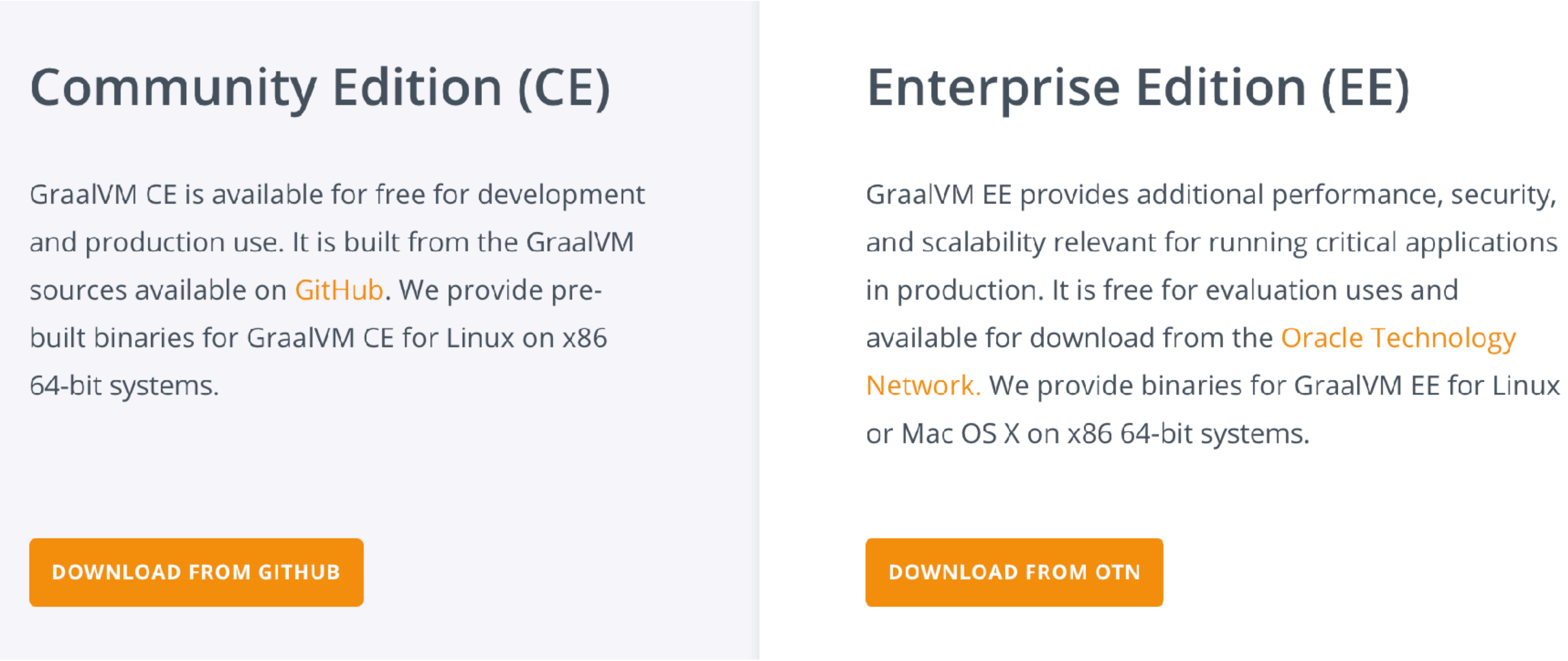
The following is intended to provide some insight into a line of research in Oracle Labs. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.



Automatic transformation of interpreters to compilers

GraalVM™

Embeddable in native or managed applications



Community Edition (CE)

GraalVM CE is available for free for development and production use. It is built from the GraalVM sources available on GitHub. We provide pre-built binaries for GraalVM CE for Linux on x86 64-bit systems.

Enterprise Edition (EE)

GraalVM EE provides additional performance, security, and scalability relevant for running critical applications in production. It is free for evaluation uses and available for download from the Oracle Technology Network. We provide binaries for GraalVM EE for Linux or Mac OS X on x86 64-bit systems.

Launched earlier this month: **GraalVM Enterprise 19.0**

ORACLE
GraalVM

- More performance
- Smaller footprint
- Managed runtime for better isolation when running native code
- Oracle Enterprise Support 7x24x365

Why GraalVM?

Fast Java, Scala, Kotlin, Groovy, Clojure...

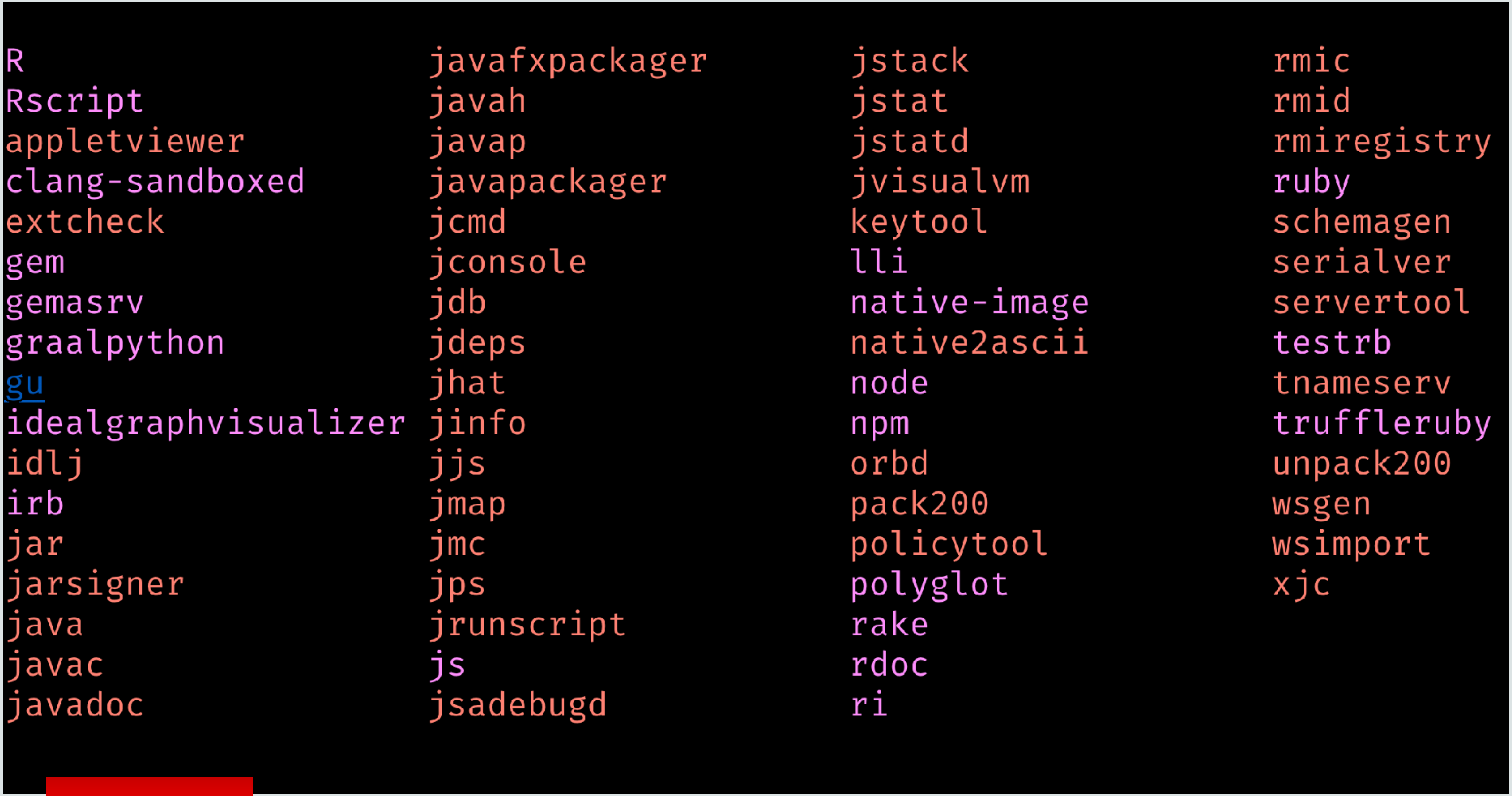
Instant startup, low footprint

Polyglot & embeddable VM

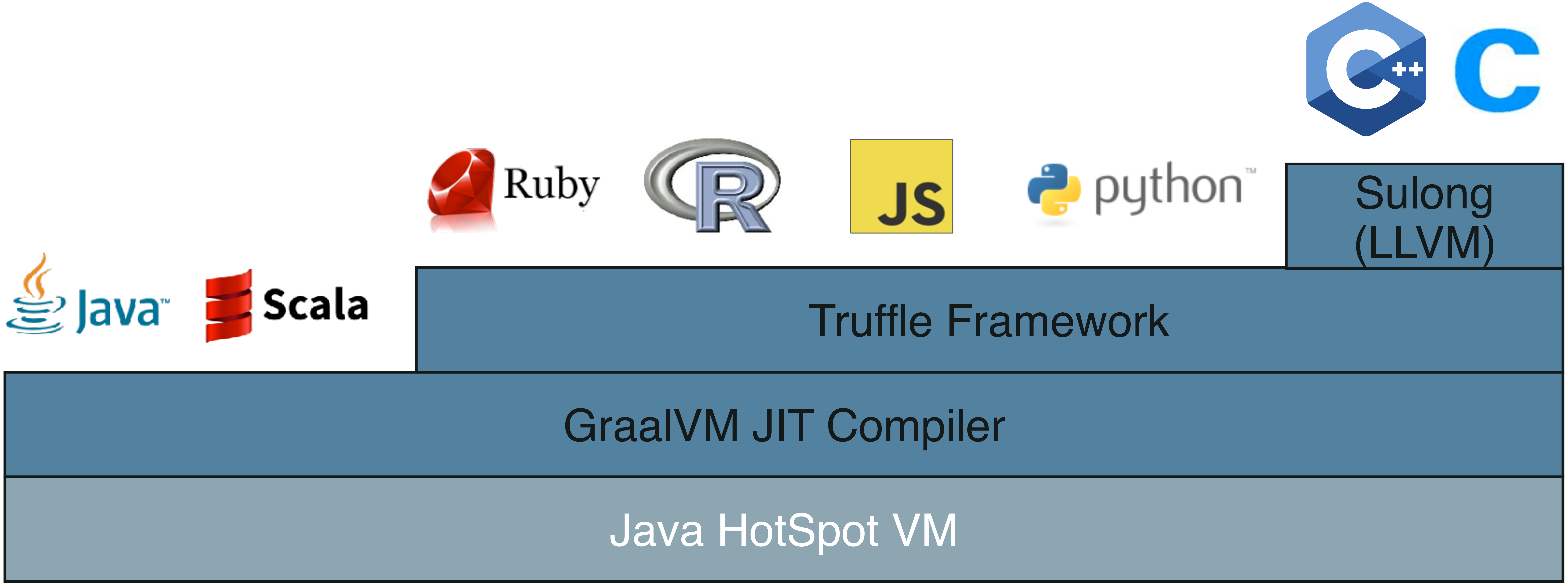
Interoperability between languages: LLVM, Python, Ruby, R

Why GraalVM? node js version

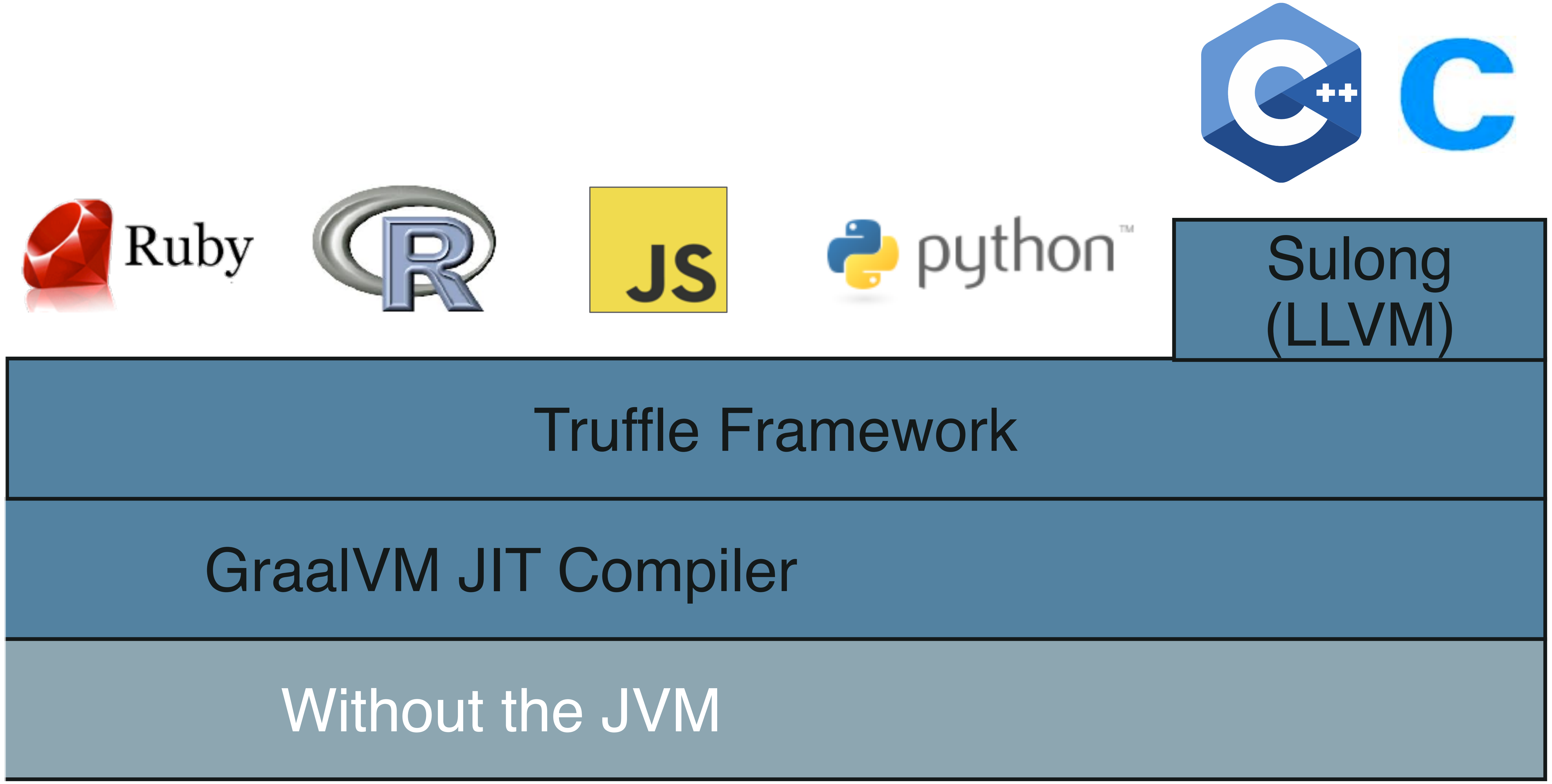
Let's find out



```
R                    javafxpackager       jstack               rmic
Rscript              javah                jstat                rmid
appletviewer         javap                jstatd               rmiregistry
clang-sandboxed      javapackager         jvisualvm            ruby
extcheck             jcmd                 keytool              schemagen
gem                  jconsole             lli                  serialver
gemasrv              jdb                  native-image         servertool
graalpython          jdeps                native2ascii         testrb
gu                   jhat                 node                 tnameserv
idealgraphvisualizer jinfo                npm                  truffleruby
idlj                 jjs                  orbd                 unpack200
irb                  jmap                 pack200              wsgen
jar                  jmc                  policytool           wsimport
jarsigner            jps                  polyglot             xjc
java                 jrunscript           rake
javac                js                   rdoc
javadoc              jsadebugd            ri
```



C++
C
Ruby
R
JS
python™
Sulong (LLVM)
Java™
Scala
Truffle Framework
GraalVM JIT Compiler
Java HotSpot VM

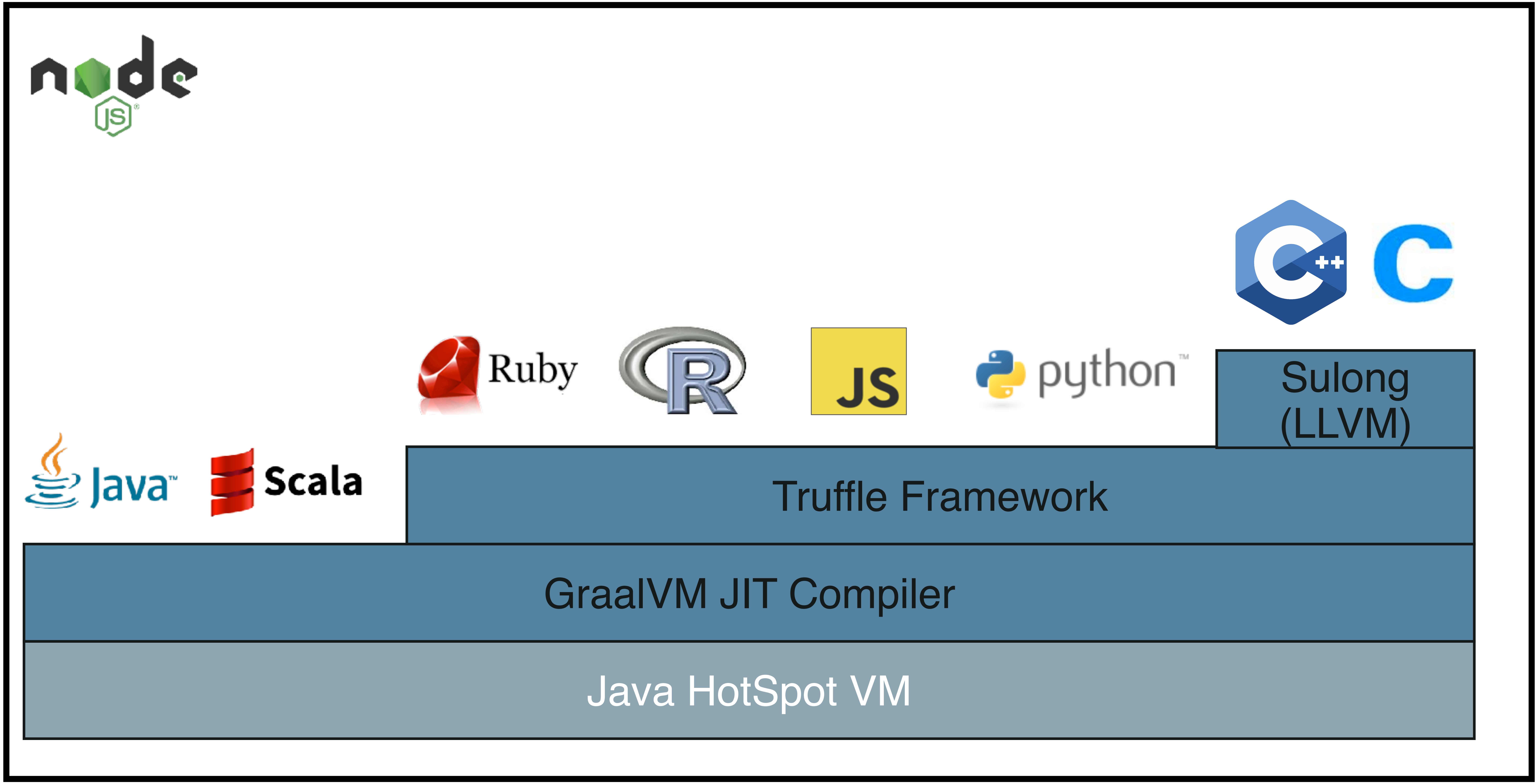


Sulong (LLVM)

Truffle Framework

GraalVM JIT Compiler

Without the JVM

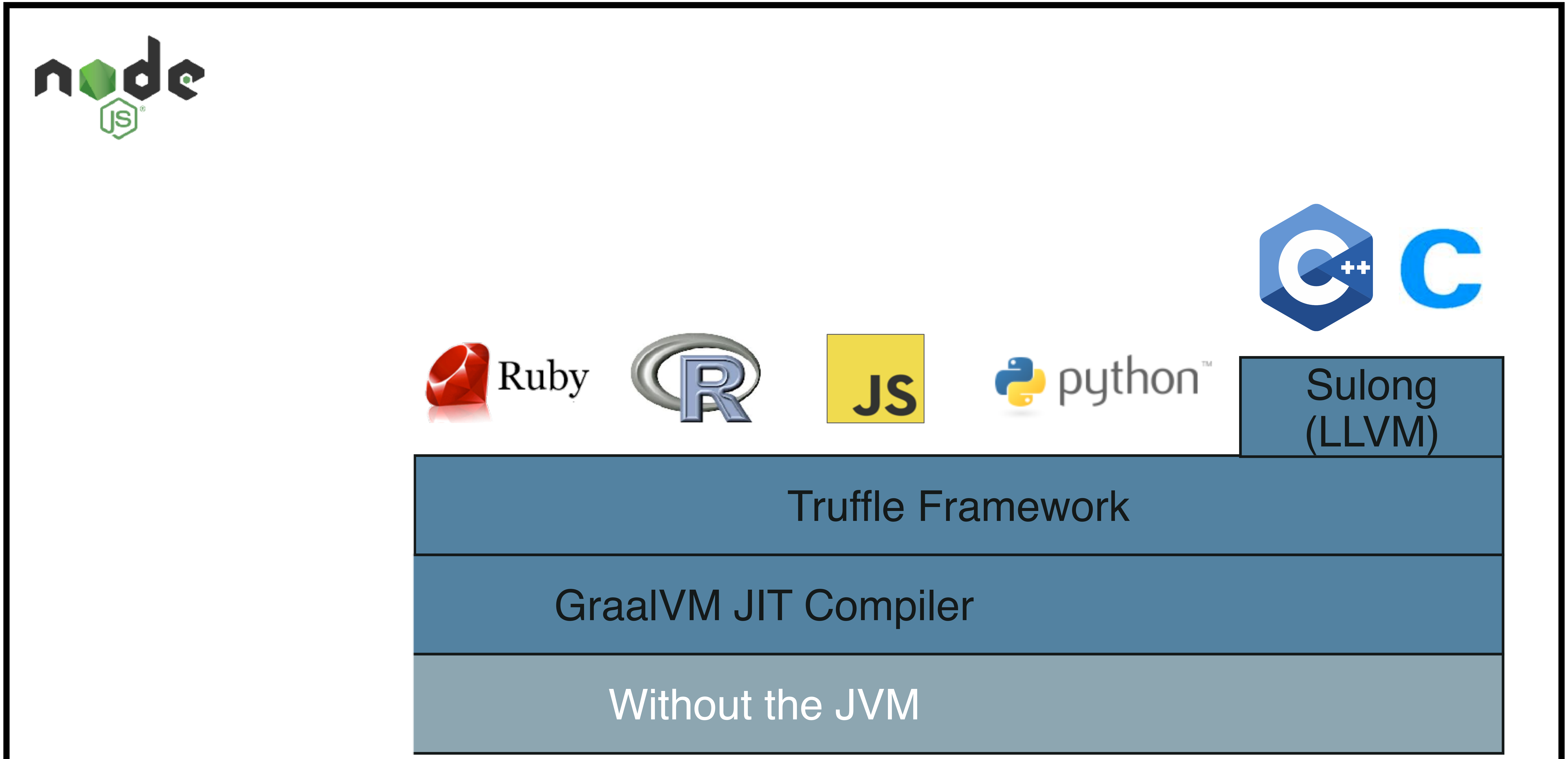


Sulong (LLVM)

Truffle Framework

GraalVM JIT Compiler

Java HotSpot VM



Sulong (LLVM)

Truffle Framework

GraalVM JIT Compiler

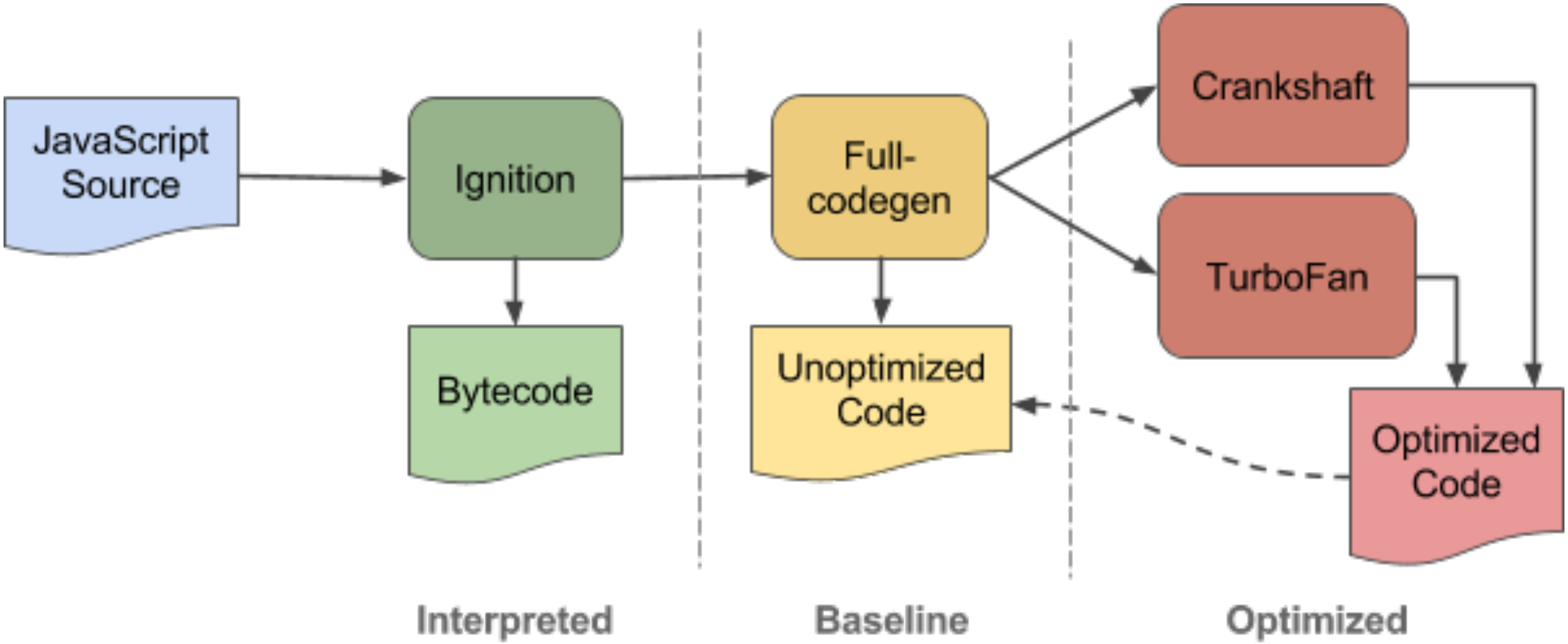
Without the JVM

Chakra Core

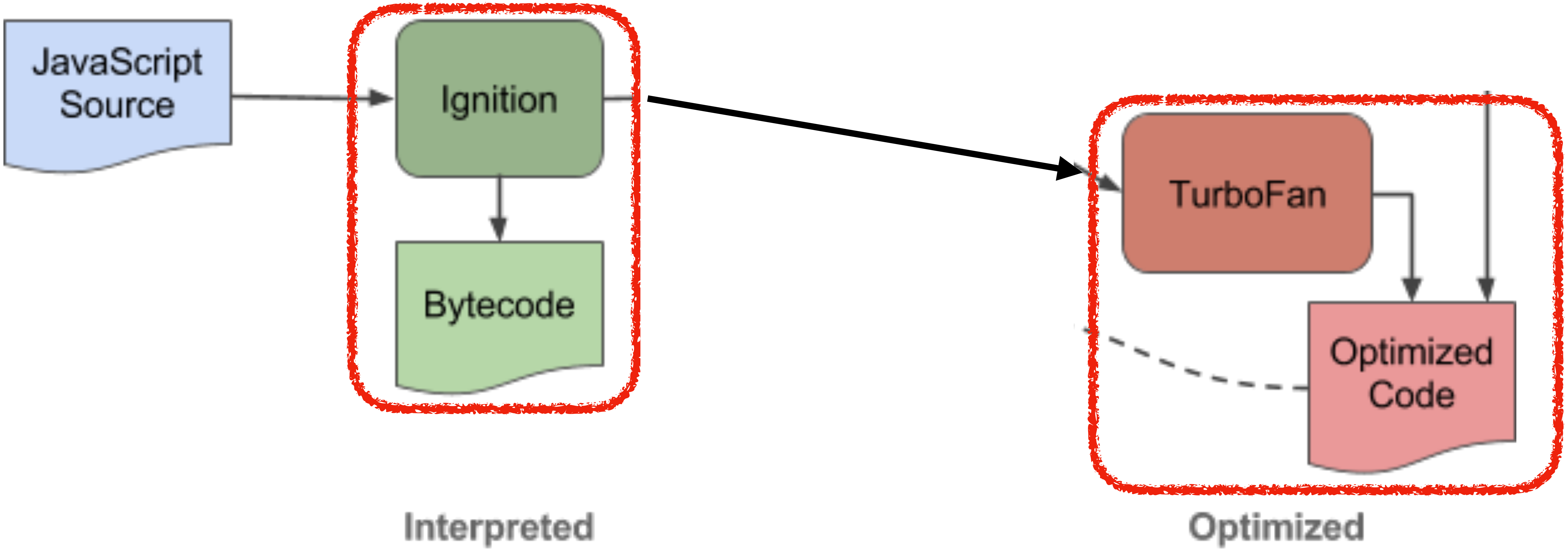
JavaScriptCore

moz://a

DEMO



V8 Dev Blog — https://v8.dev/blog/ignition-interpreter

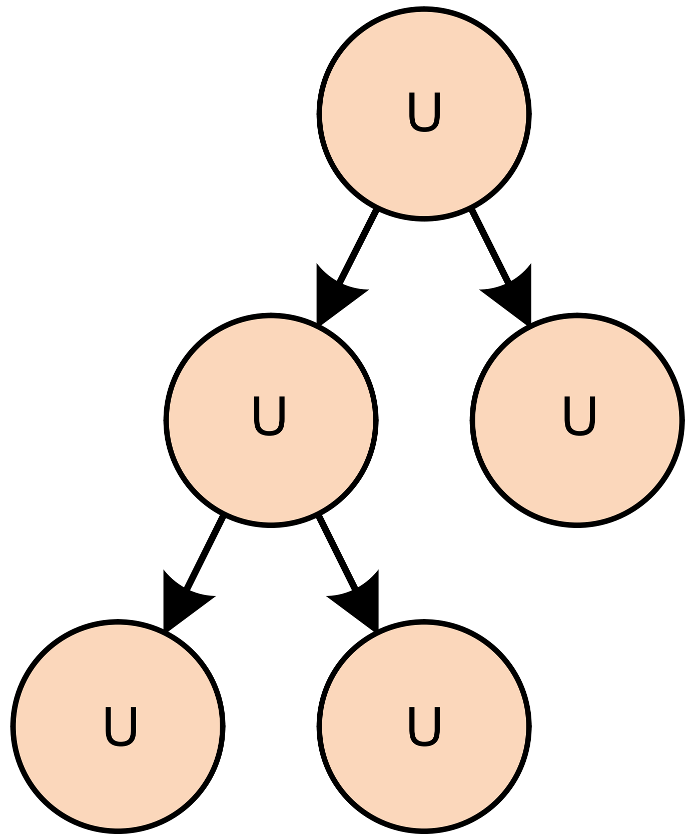


V8 Dev Blog — https://v8.dev/blog/ignition-interpreter

```
function TreeNode(execute, children) {
    this.execute = execute;
    this.children = children;
}
```

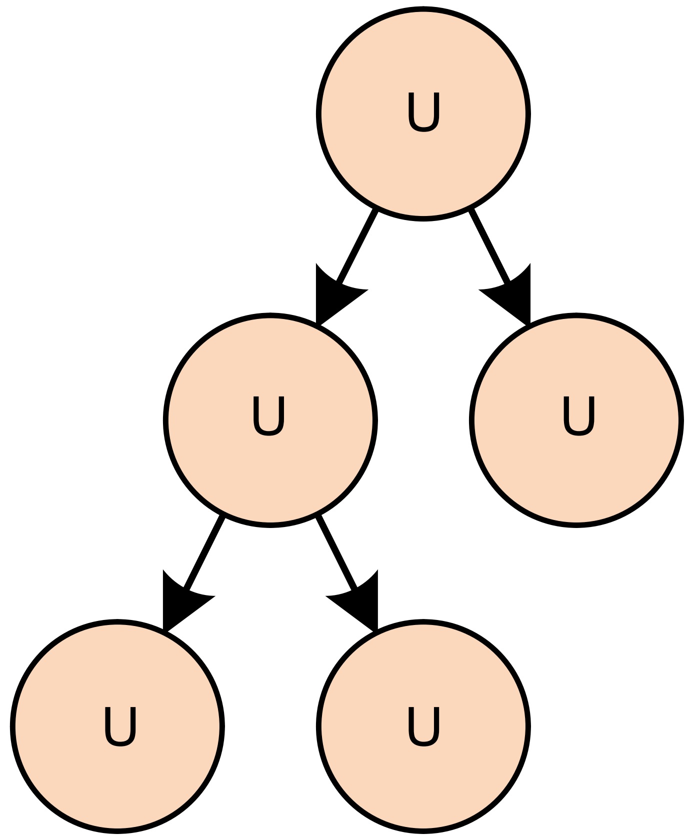
Interpreter

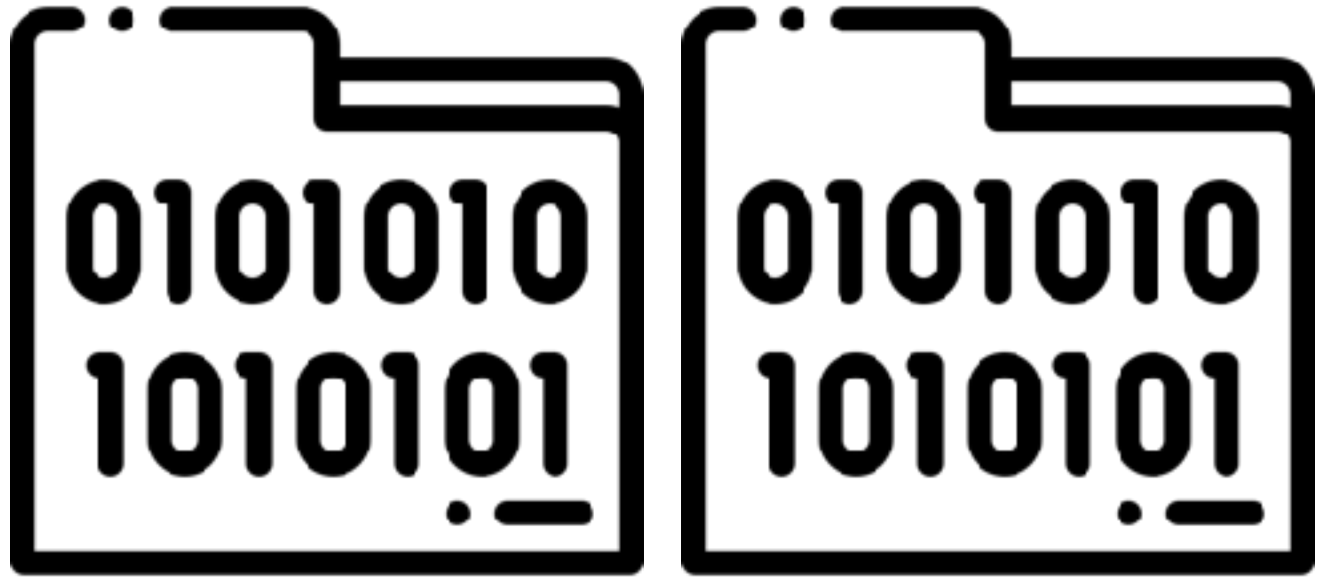
```
worklist.push(rootMethod.start)
do {
  node = worklist.peek();

  if (node.hasNotEvaluatedSuccessors()) {
    worklist.pushAll(node.successors)
  } else {
    worklist.pop();
    node.evaluate();
  }
} while (worklist.notEmpty)
```

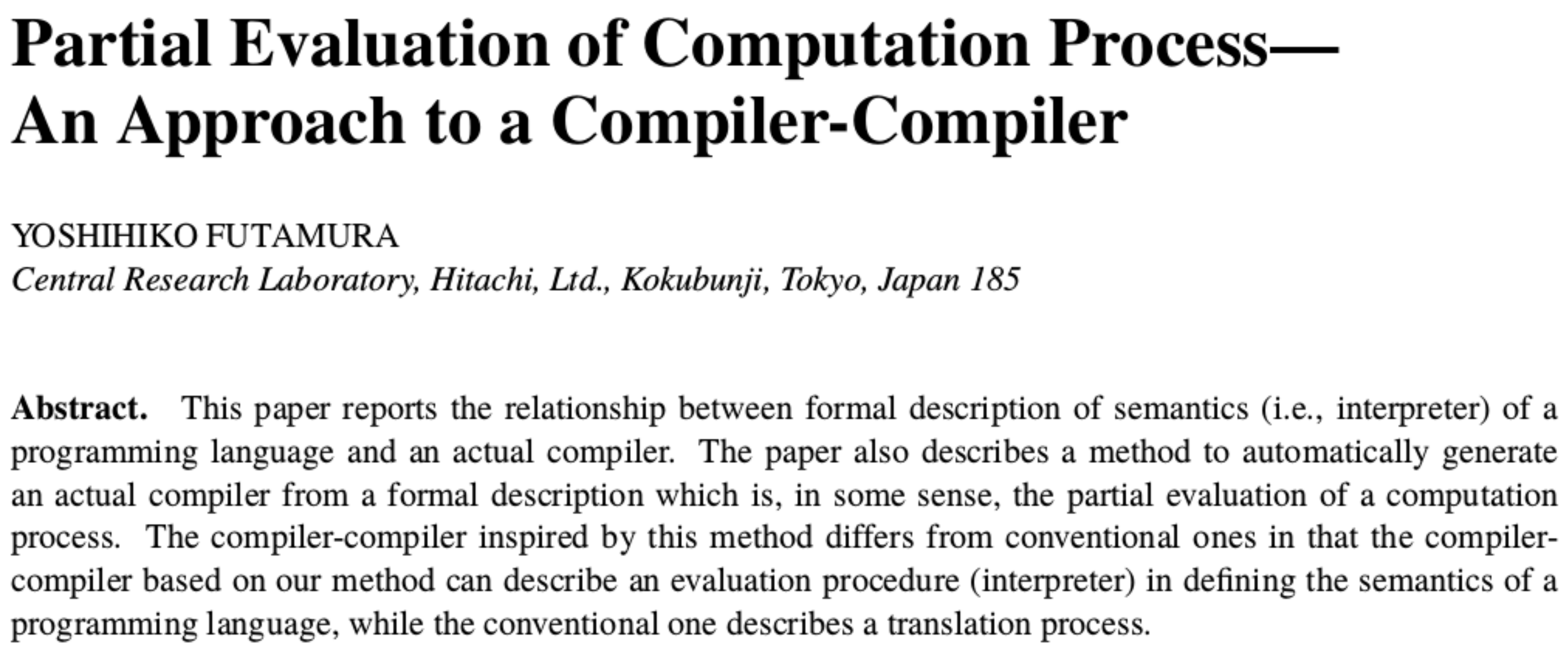


Reality: more lines of code

Execution







Partial Evaluation of Computation Process— An Approach to a Compiler-Compiler

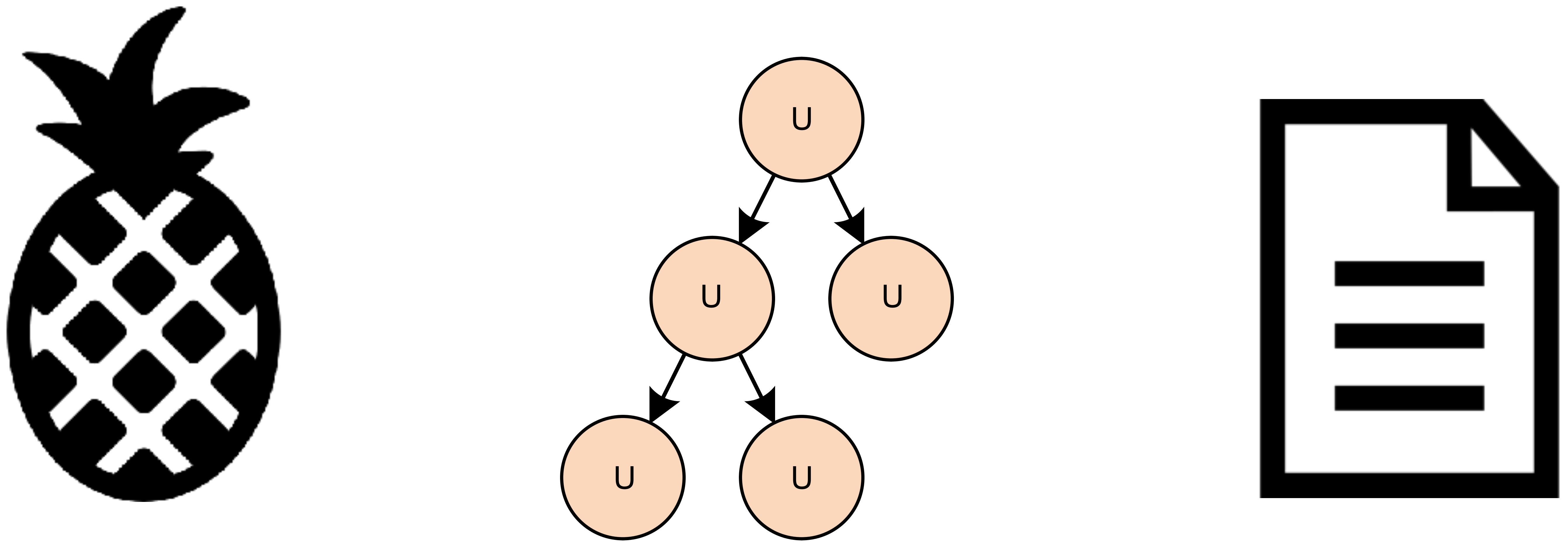
YOSHIHIKO FUTAMURA
Central Research Laboratory, Hitachi, Ltd., Kokubunji, Tokyo, Japan 185

Abstract. This paper reports the relationship between formal description of semantics (i.e., interpreter) of a programming language and an actual compiler. The paper also describes a method to automatically generate an actual compiler from a formal description which is, in some sense, the partial evaluation of a computation process. The compiler-compiler inspired by this method differs from conventional ones in that the compiler-compiler based on our method can describe an evaluation procedure (interpreter) in defining the semantics of a programming language, while the conventional one describes a translation process.

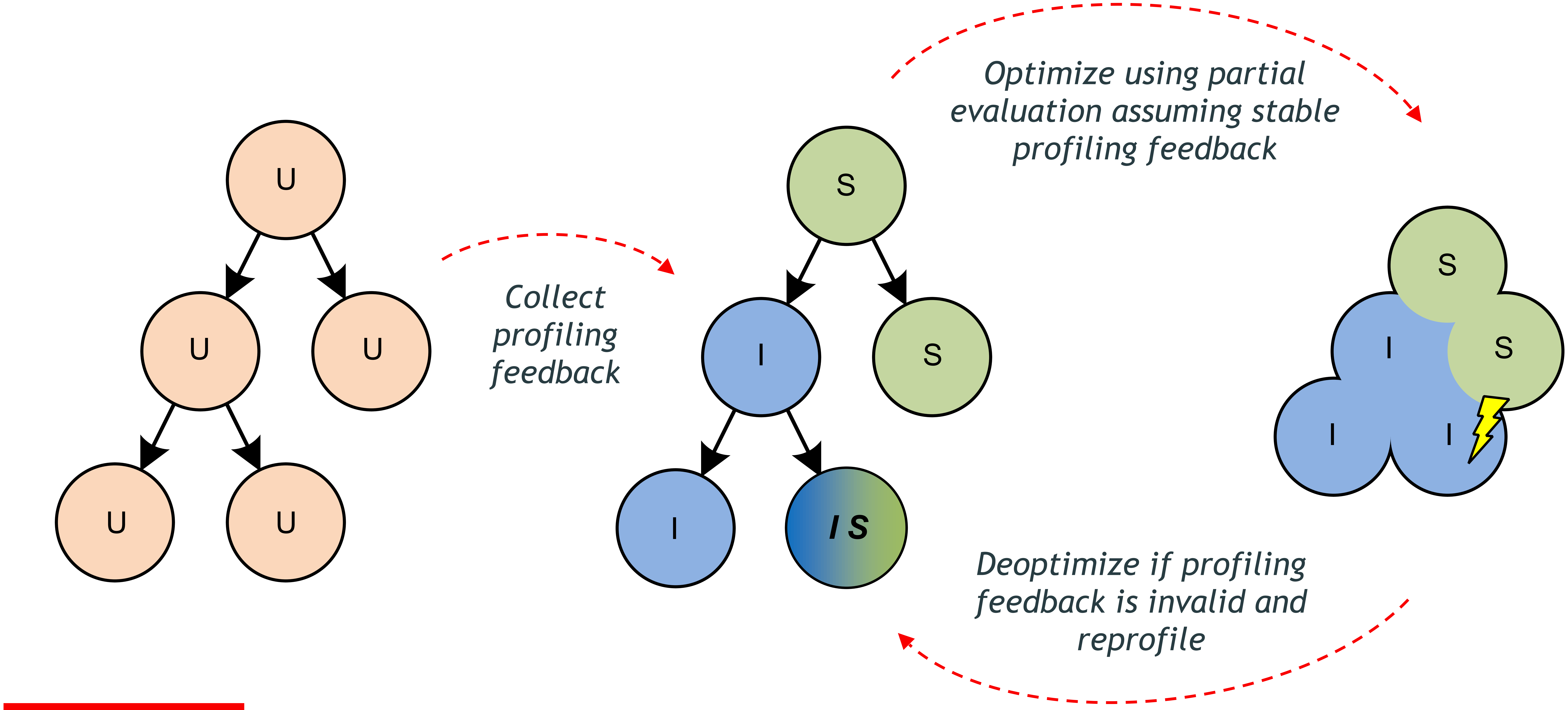
Systems.Computers.Controls, Volume 2, Number 5, 1971

```
function pow (x, y) {
    return x ** y;
}


function pow (x, 2) {
    return x * x;
}
```

Executable



Speculation and Deoptimization

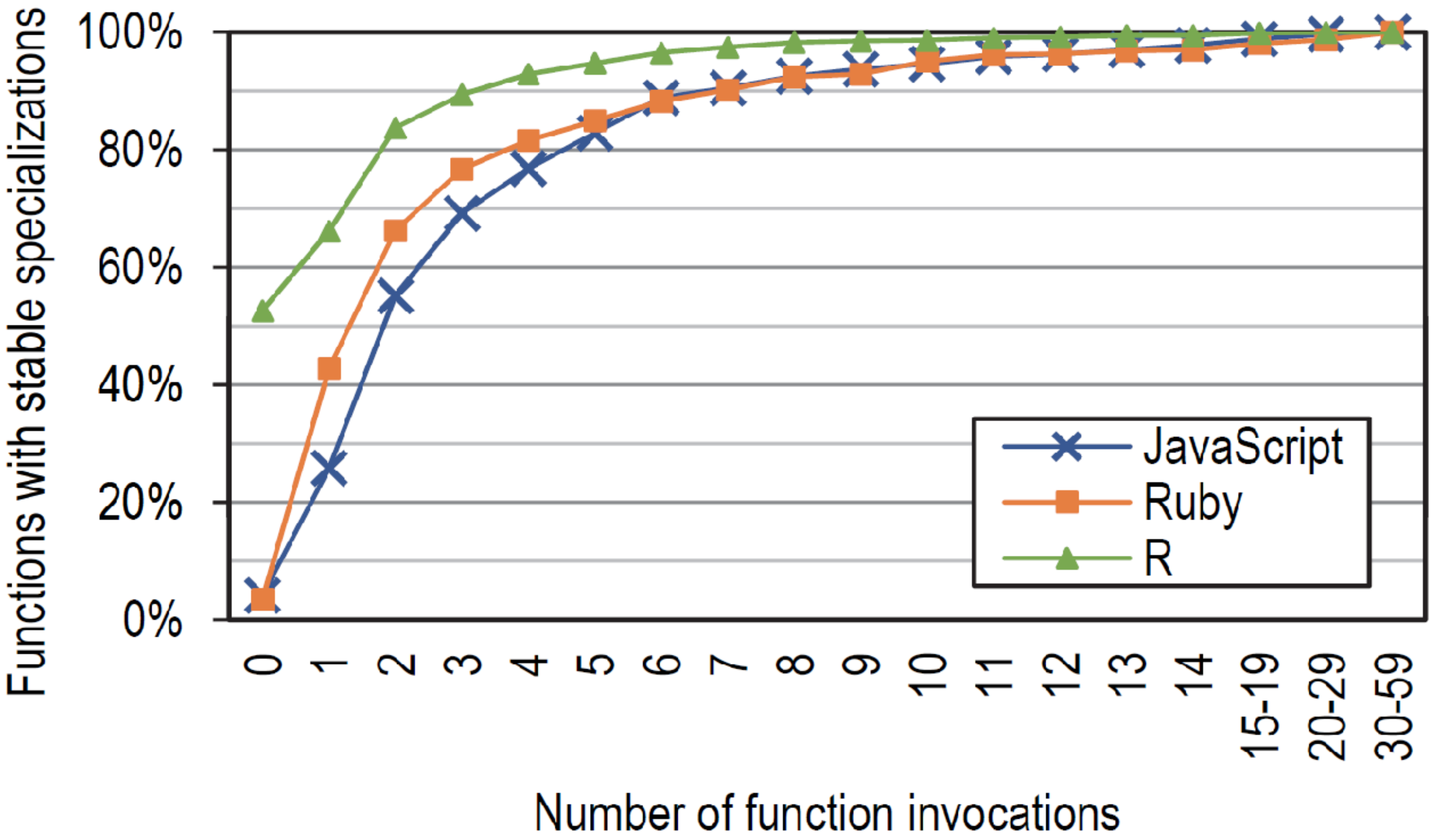


Collect profiling feedback

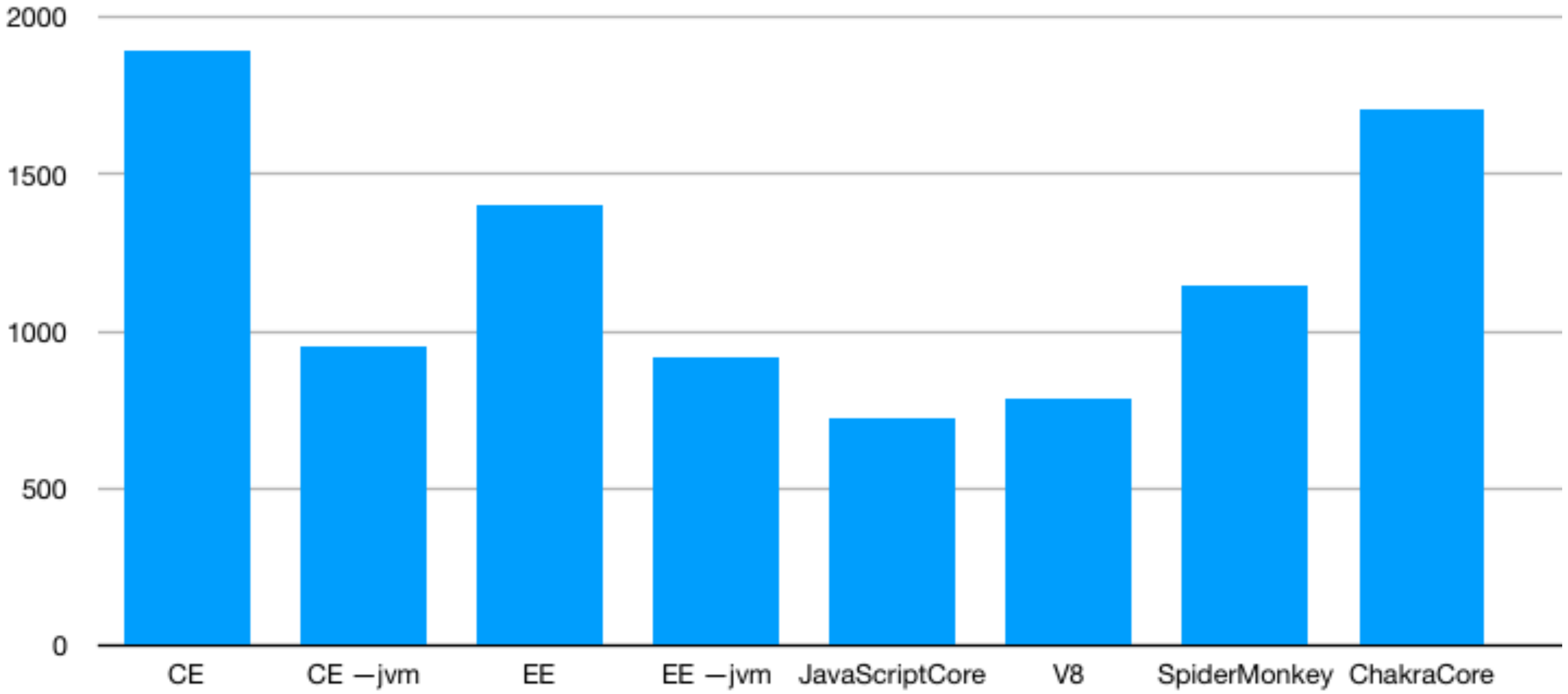
Optimize using partial evaluation assuming stable profiling feedback

Deoptimize if profiling feedback is invalid and reprofile

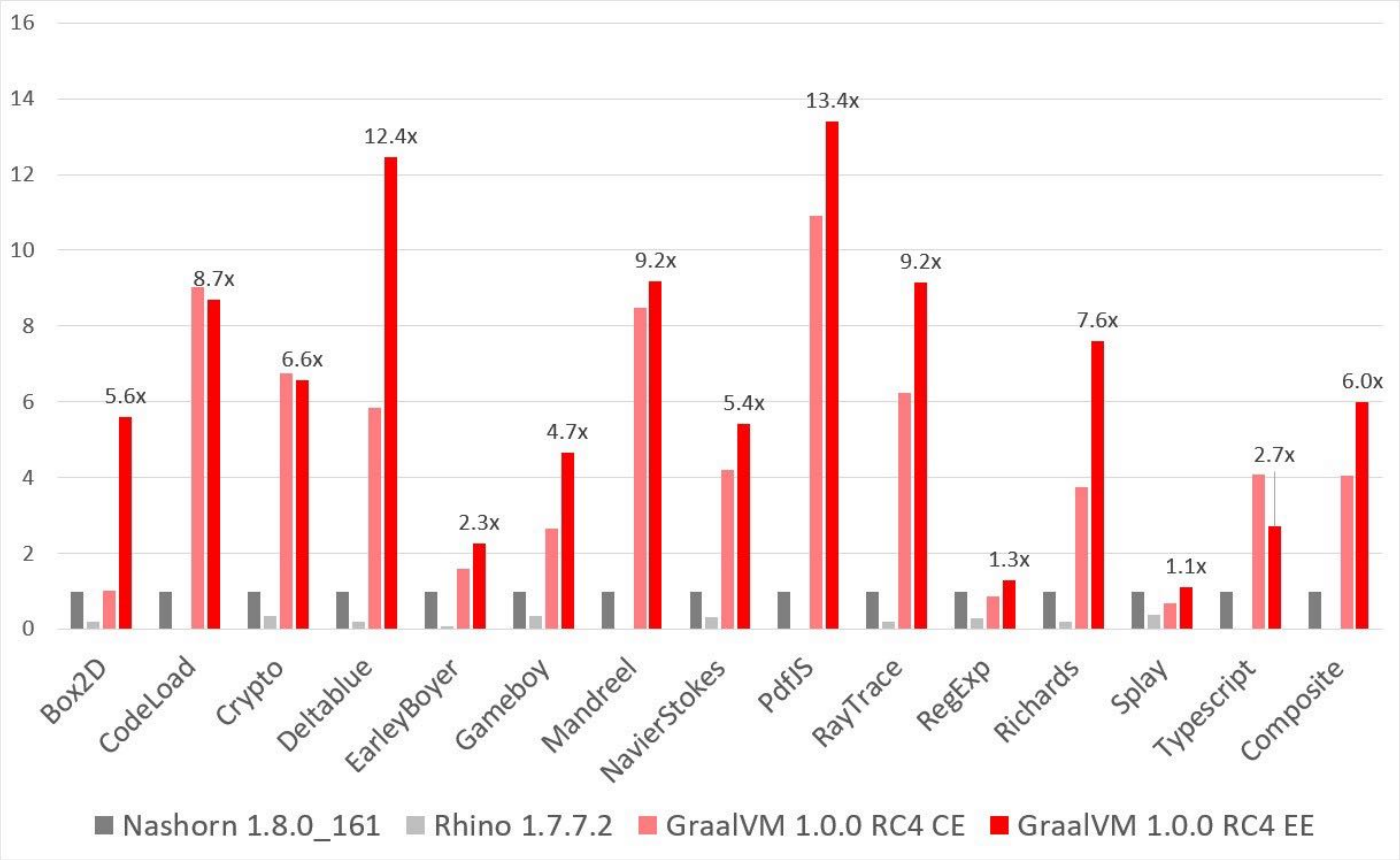
Stability


Functions with stable specializations
100%
80%
60%
40%
20%
0%
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15-19
20-29
30-59
Number of function invocations
JavaScript
Ruby
R

Warmed-up ClojureScript unit tests

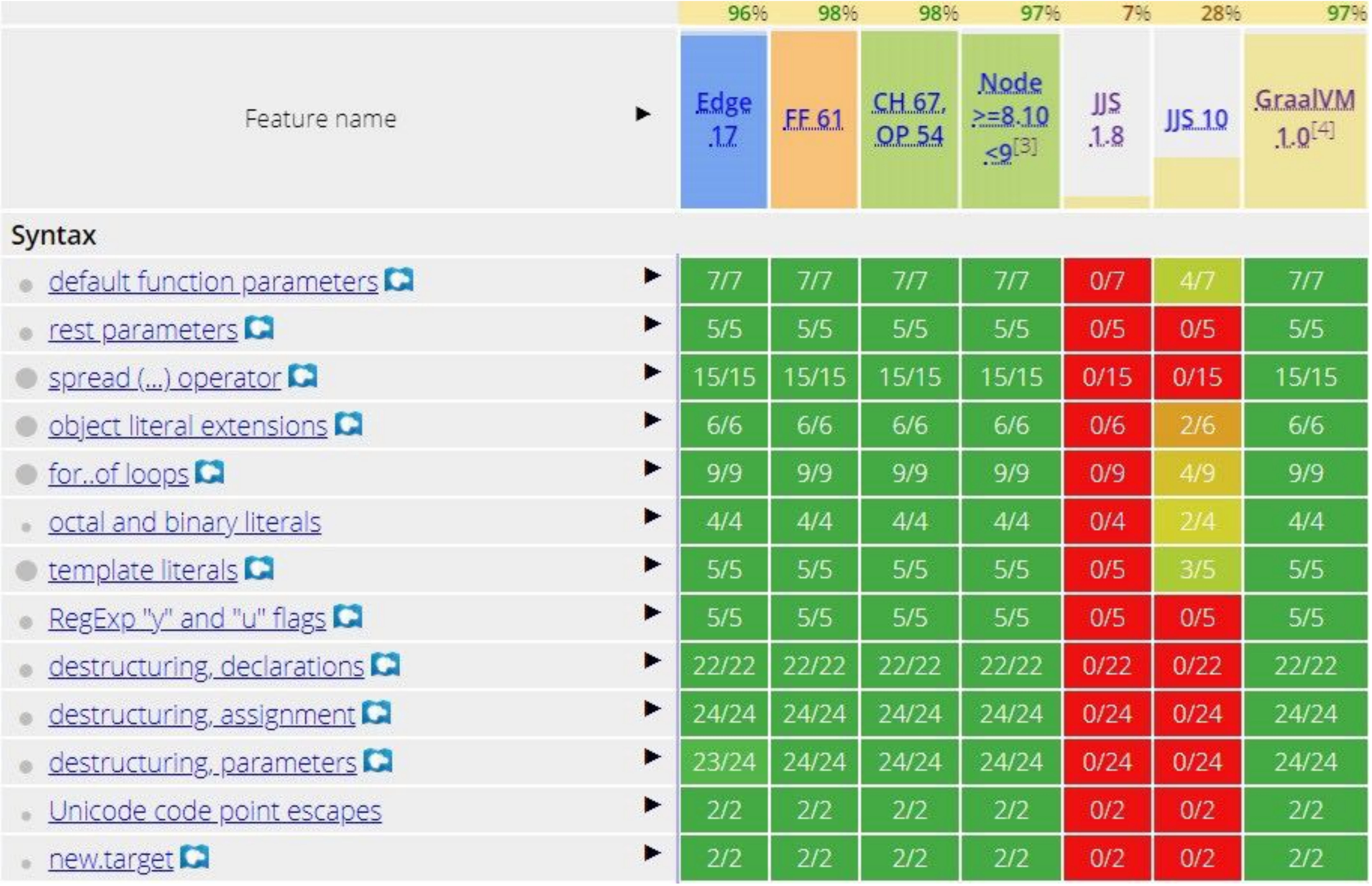


https://github.com/graalvm/graaljs/issues/29


16
14
12
10
8
6
4
2
0
5.6x
8.7x
6.6x
12.4x
2.3x
4.7x
9.2x
5.4x
13.4x
9.2x
1.3x
7.6x
1.1x
2.7x
6.0x
Box2D
CodeLoad
Crypto
Deltablue
EarleyBoyer
Gameboy
Mandreel
NavierStokes
PdfJS
RayTrace
RegExp
Richards
Splay
Typescript
Composite
Nashorn 1.8.0_161
Rhino 1.7.7.2
GraalVM 1.0.0 RC4 CE
GraalVM 1.0.0 RC4 EE

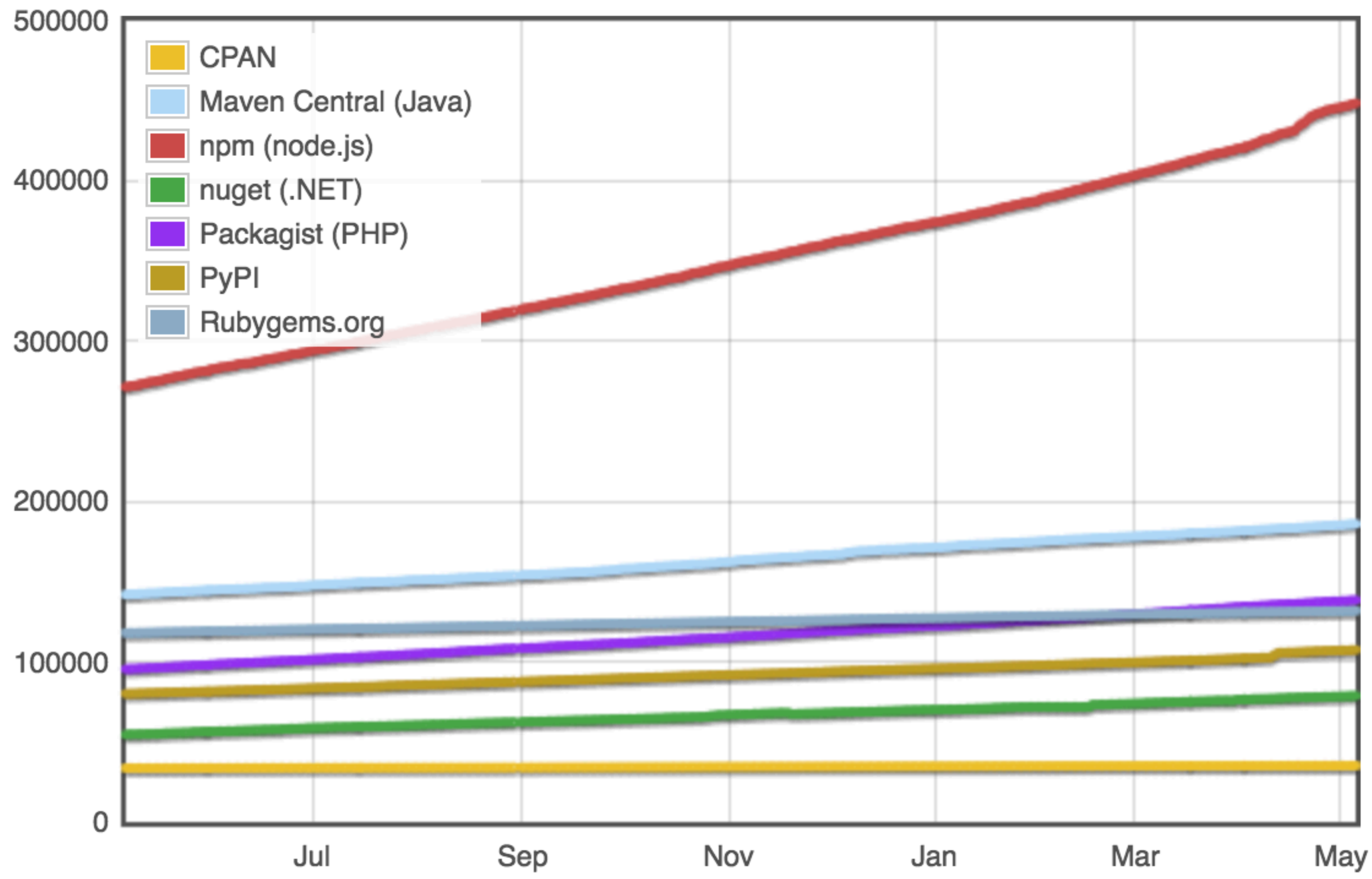
ORACLE

ECMAScript compatibility



	96%	98%	98%	97%	7%	28%	97%
Feature name	Edge 17	FF 61	CH 67, OP 54	Node >=8.10 <9[3]	JJS 1.8	JJS 10	GraalVM 1.0[4]
Syntax							
default function parameters	7/7	7/7	7/7	7/7	0/7	4/7	7/7
rest parameters	5/5	5/5	5/5	5/5	0/5	0/5	5/5
spread (...) operator	15/15	15/15	15/15	15/15	0/15	0/15	15/15
object literal extensions	6/6	6/6	6/6	6/6	0/6	2/6	6/6
for..of loops	9/9	9/9	9/9	9/9	0/9	4/9	9/9
octal and binary literals	4/4	4/4	4/4	4/4	0/4	2/4	4/4
template literals	5/5	5/5	5/5	5/5	0/5	3/5	5/5
RegExp "y" and "u" flags	5/5	5/5	5/5	5/5	0/5	0/5	5/5
destructuring, declarations	22/22	22/22	22/22	22/22	0/22	0/22	22/22
destructuring, assignment	24/24	24/24	24/24	24/24	0/24	0/24	24/24
destructuring, parameters	23/24	24/24	24/24	24/24	0/24	0/24	24/24
Unicode code point escapes	2/2	2/2	2/2	2/2	0/2	0/2	2/2
new.target	2/2	2/2	2/2	2/2	0/2	0/2	2/2

https://kangax.github.io/compat-table/es2016plus/

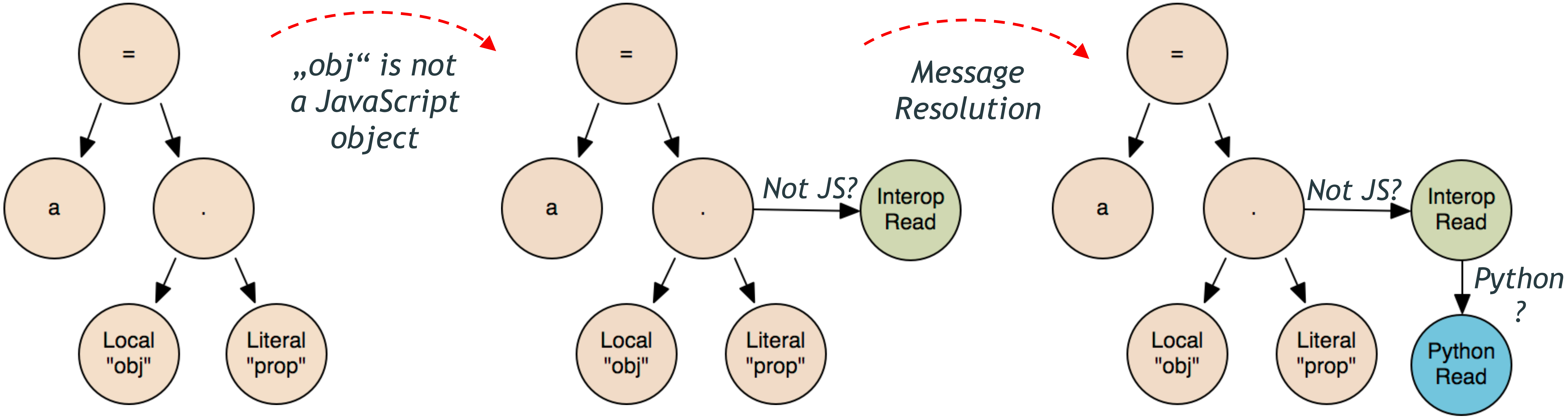

500000
400000
300000
200000
100000
0
Jul
Sep
Nov
Jan
Mar
May
CPAN
Maven Central (Java)
npm (node.js)
nuget (.NET)
Packagist (PHP)
PyPI
Rubygems.org

DEMO

DEMO

Interoperability Example

a = obj.prop



Uses an inline cache entry for every language.

Interoperability Protocol

```java
boolean isNull(Object receiver);
boolean isBoolean(Object receiver);
boolean asBoolean(Object receiver);
boolean isExecutable(Object receiver);
Object execute(Object receiver, Object... arguments);
boolean isInstantiable(Object receiver);
Object instantiate(Object receiver, Object... arguments);
boolean isString(Object receiver);
String asString(Object receiver);
boolean isNumber(Object receiver);
boolean fitsInByte(Object receiver);
boolean fitsInShort(Object receiver);
boolean fitsInInt(Object receiver);
boolean fitsInLong(Object receiver);
boolean fitsInFloat(Object receiver);
boolean fitsInDouble(Object receiver);
byte asByte(Object receiver);
short asShort(Object receiver);
int asInt(Object receiver);
long asLong(Object receiver);
float asFloat(Object receiver);
```

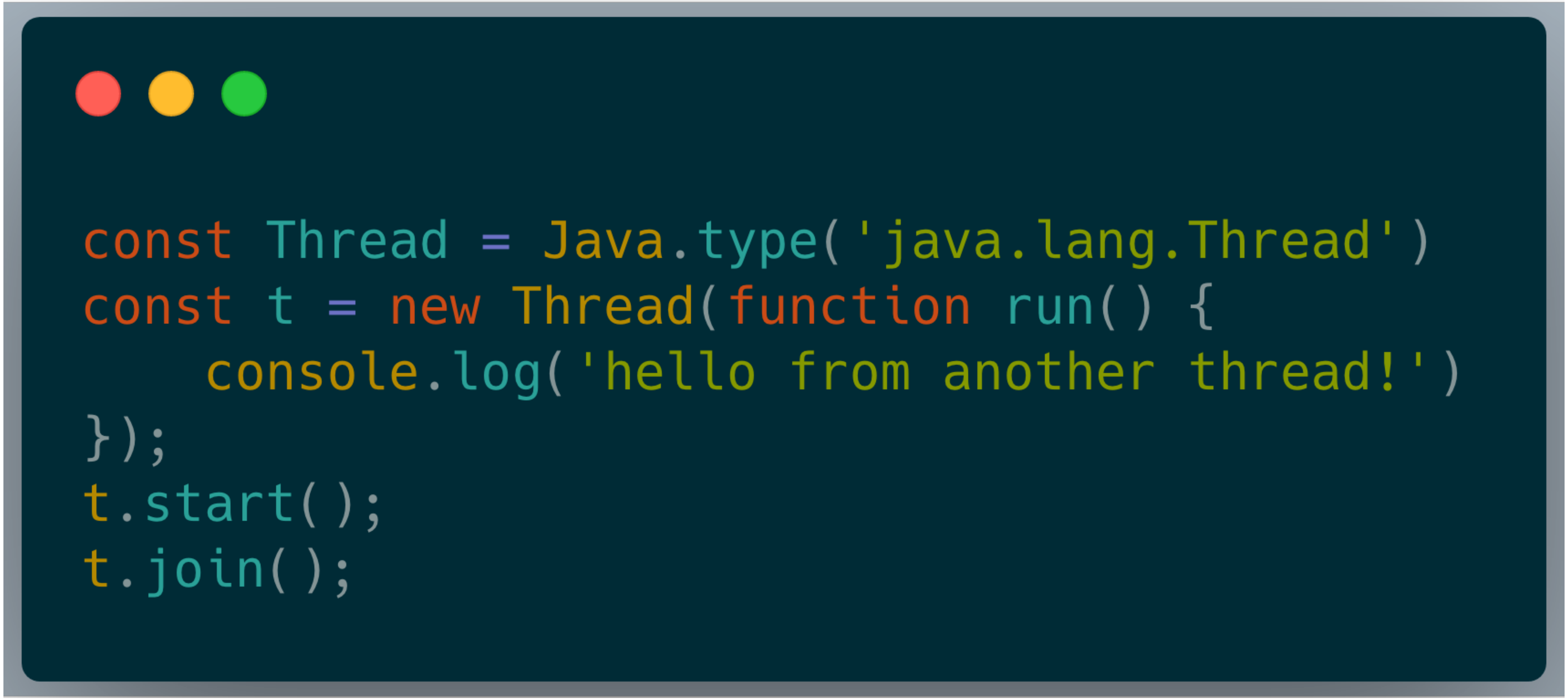
```java
boolean isObject(Object receiver);
Object getMembers(Object receiver, boolean includeInternal);
Object getMembers(Object receiver);
boolean isMemberReadable(Object receiver, String member);
Object readMember(Object receiver, String member);
boolean isMemberModifiable(Object receiver, String member);
boolean isMemberInsertable(Object receiver, String member);
void writeMember(Object receiver, String member, Object value);
boolean isMemberRemovable(Object receiver, String member);
void removeMember(Object receiver, String member);
boolean isMemberInvokable(Object receiver, String member);
Object invokeMember(Object receiver, String member, Object... arguments);
boolean isMemberInternal(Object receiver, String member);
boolean isMemberWritable(Object receiver, String member);
boolean isMemberExisting(Object receiver, String member);
boolean isArray(Object receiver);
Object readElement(Object receiver, long index);
long getArraySize(Object receiver);
boolean isElementReadable(Object receiver, long index);
void writeElement(Object receiver, long index, Object value);
void removeElement(Object receiver, long index);
boolean isElementModifiable(Object receiver, long index);
boolean isElementInsertable(Object receiver, long index);
boolean isElementRemovable(Object receiver, long index);
boolean isElementWritable(Object receiver, long index);
boolean isElementExisting(Object receiver, long index);
boolean isPointer(Object receiver);
long asPointer(Object receiver);
Object toNative(Object receiver);
```

node JS®

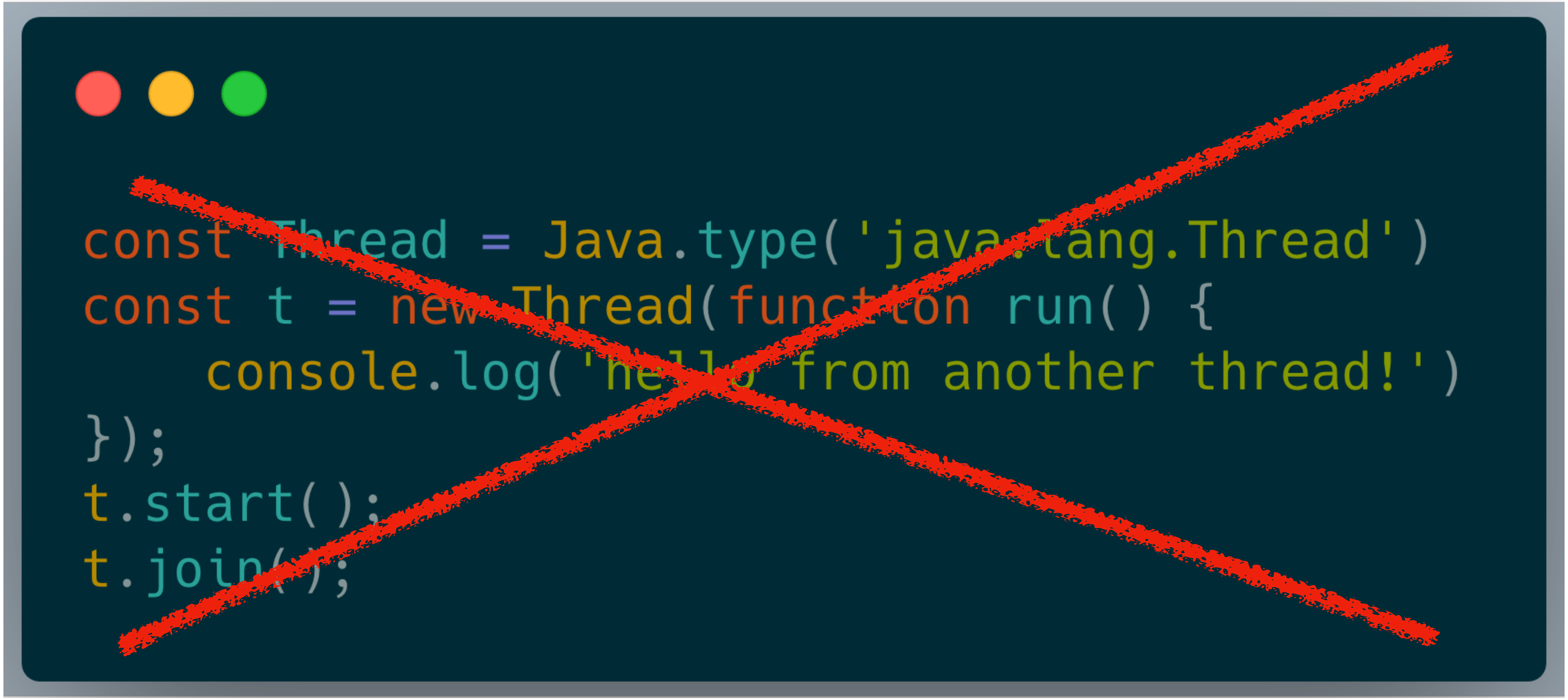
Java™

↻

⇅



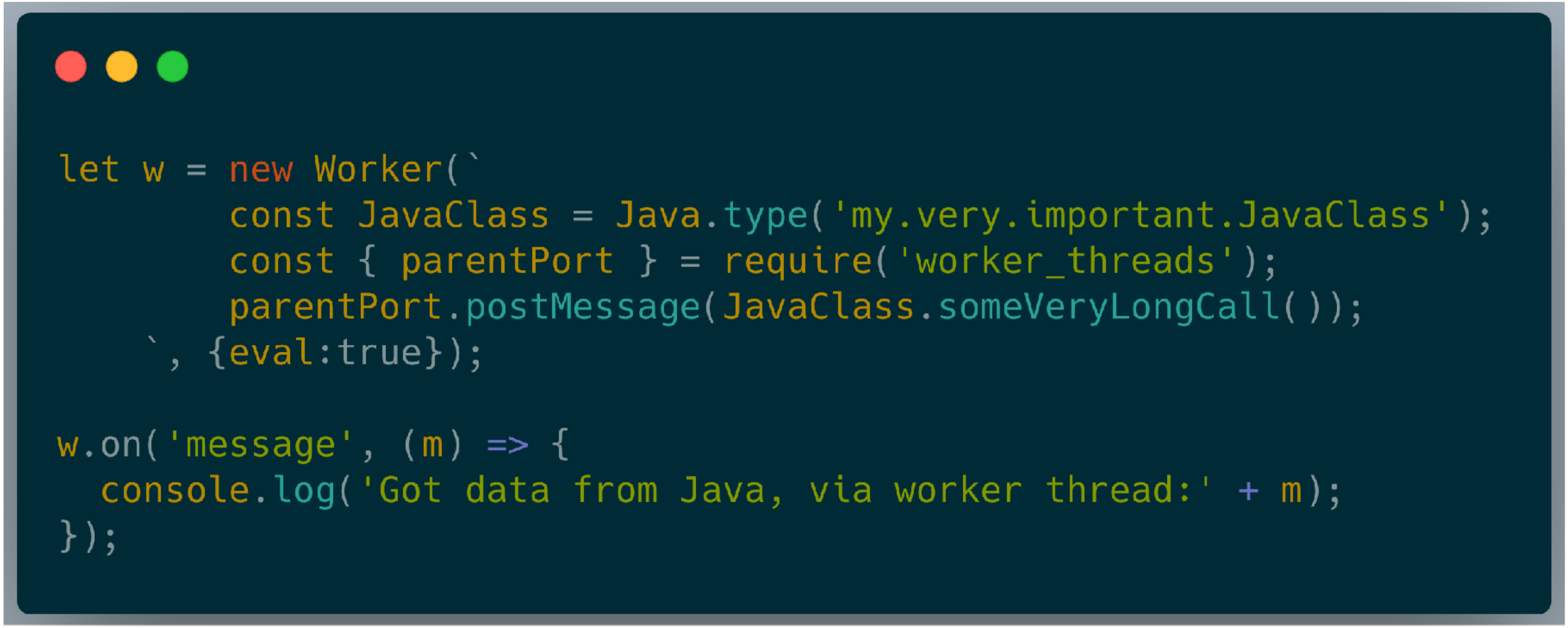
```
const Thread = Java.type('java.lang.Thread')
const t = new Thread(function run() {
    console.log('hello from another thread!')
});
t.start();
t.join();
```



```
const Thread = Java.type('java.lang.Thread')
const t = new Thread(function run() {
    console.log('hello from another thread!')
});
t.start();
t.join();
```

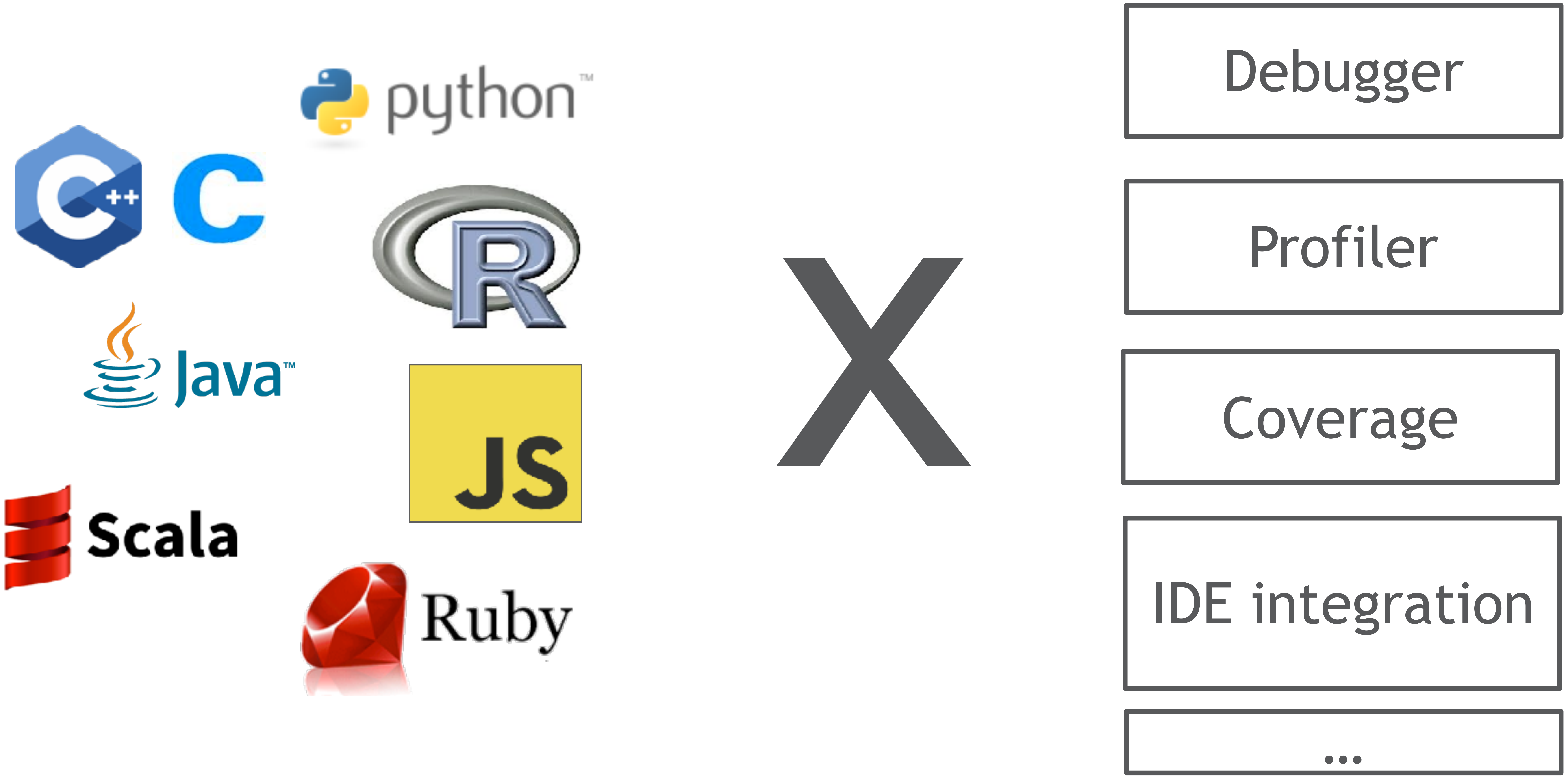
- An **arbitrary number** of JS runtimes can be **used by one thread at a time**.
- Concurrent access to Java objects is **allowed**
- Concurrent access to JavaScript objects is ***not*** **allowed**

https://medium.com/graalvm/multi-threaded-java-javascript-language-interoperability-in-graalvm-2f19c1f9c37b



```
let w = new Worker(`
        const JavaClass = Java.type('my.very.important.JavaClass');
        const { parentPort } = require('worker_threads');
        parentPort.postMessage(JavaClass.someVeryLongCall());
    `, {eval:true});

w.on('message', (m) => {
  console.log('Got data from Java, via worker thread:' + m);
});
```

Composing Tools



X

Debugger

Profiler

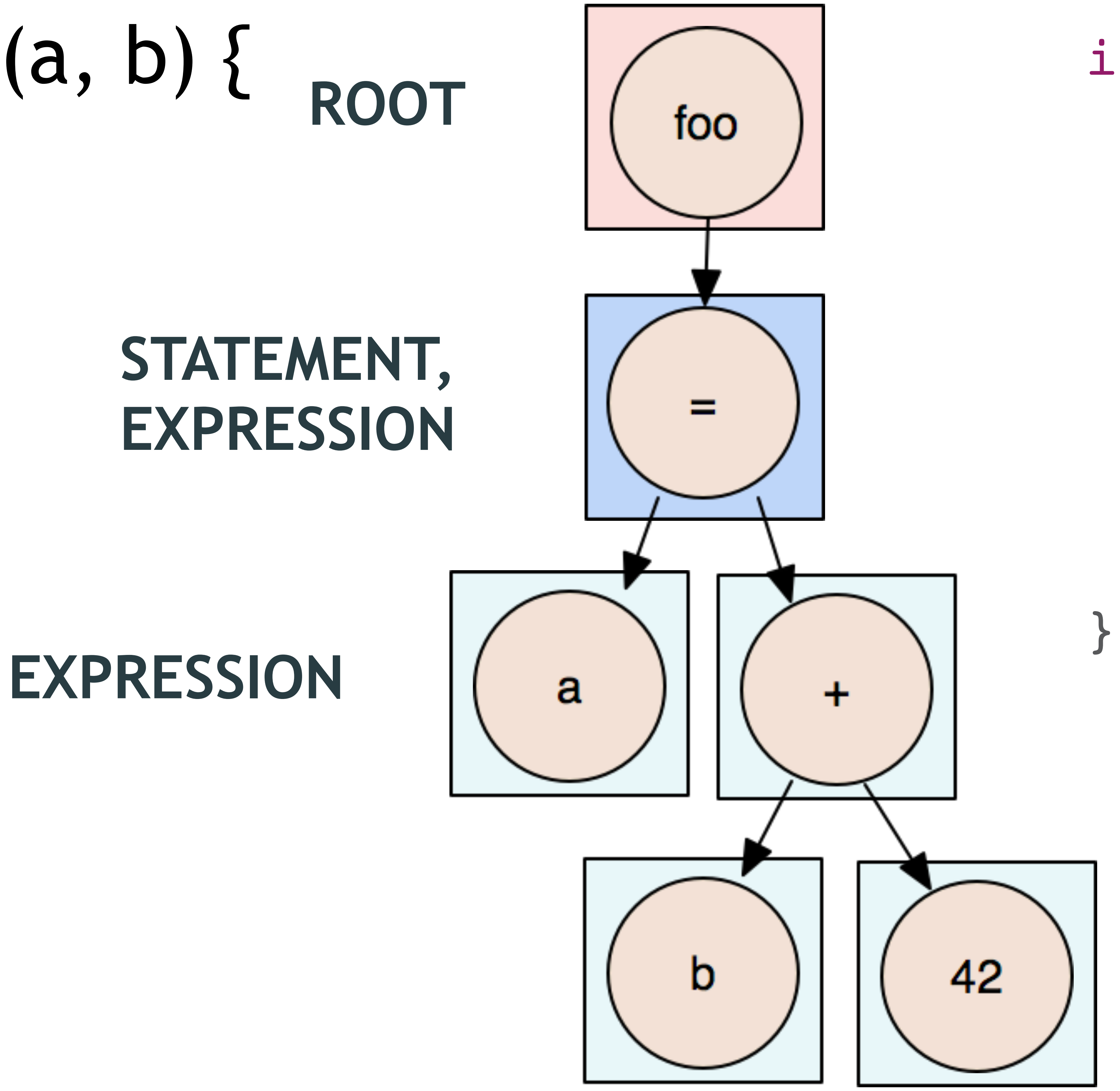
Coverage

IDE integration

…

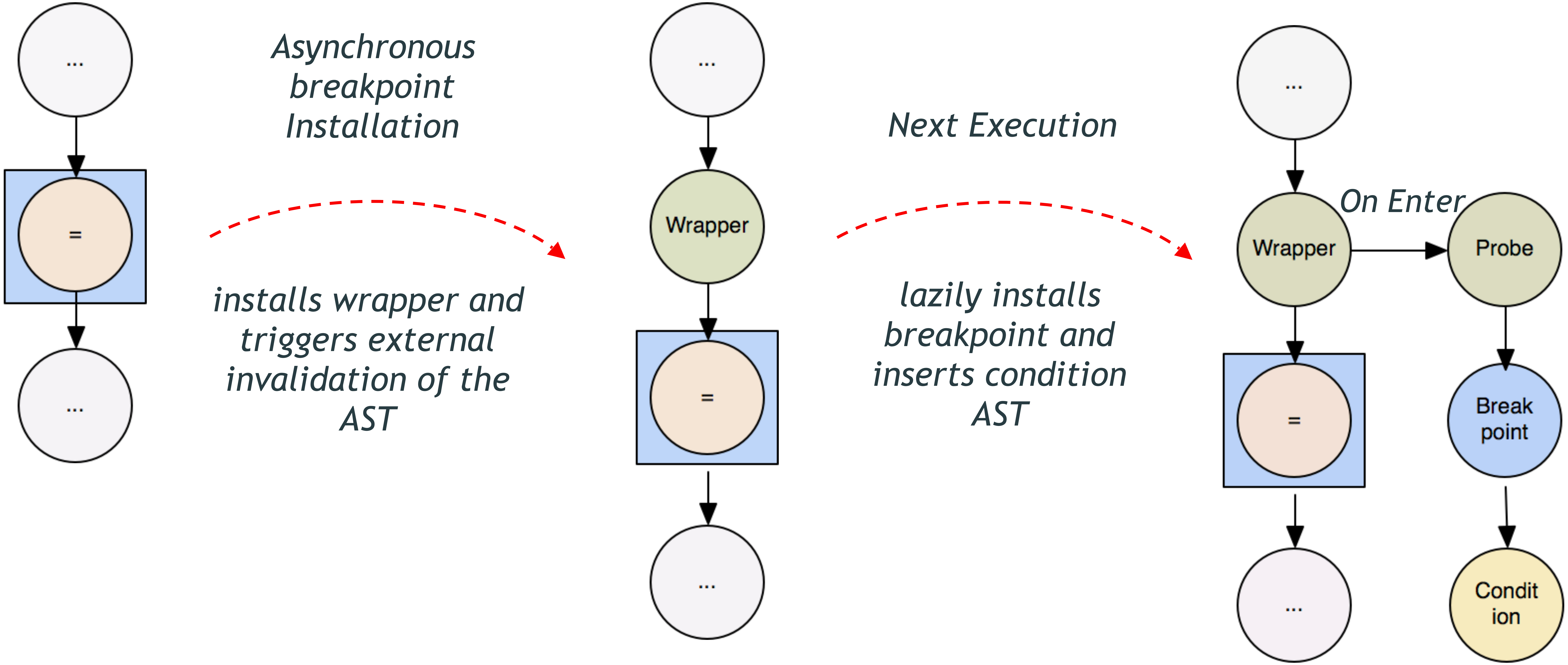
AST Node (vertex) Tagging Example

```
function foo(a, b) {
    a = b + 42
}
```



```
interface InstrumentableNode {

    boolean isInstrumentable();

    boolean hasTag(Class<Tag> tag);

    WrapperNode createWrapper(ProbeNode probe

    ...
}
```

Conditional Breakpoint Example



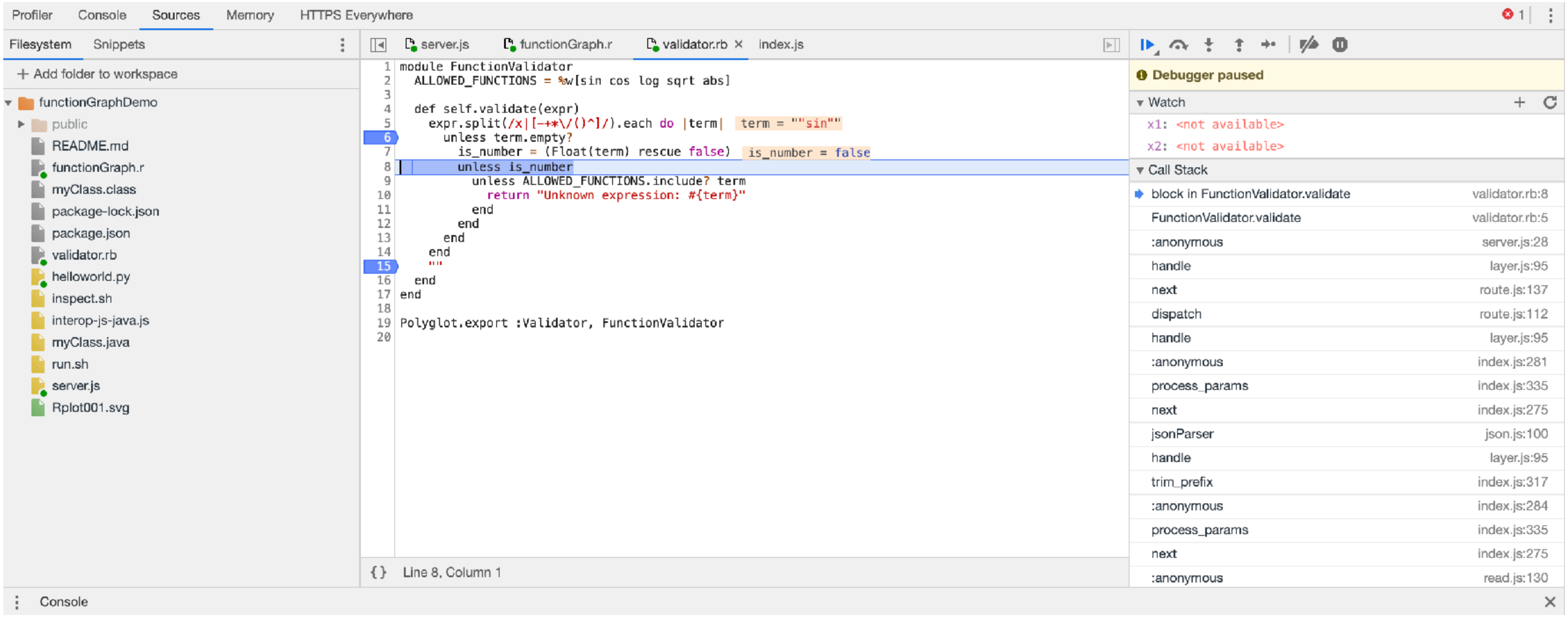
Asynchronous breakpoint Installation

installs wrapper and triggers external invalidation of the AST

Next Execution

lazily installs breakpoint and inserts condition AST

DEMO



```
module FunctionValidator
  ALLOWED_FUNCTIONS = %w[sin cos log sqrt abs]

  def self.validate(expr)
    expr.split(/x|[-+*\/()^]/).each do |term|  term = ""sin""
      unless term.empty?
        is_number = (Float(term) rescue false)  is_number = false
        unless is_number
          unless ALLOWED_FUNCTIONS.include? term
            return "Unknown expression: #{term}"
          end
        end
      end
    end
    ""
  end
end

Polyglot.export :Validator, FunctionValidator
```

Line 8, Column 1

Debugger paused

```
x1: <not available>
x2: <not available>
```

block in FunctionValidator.validate	validator.rb:8
FunctionValidator.validate	validator.rb:5
:anonymous	server.js:28
handle	layer.js:95
next	route.js:137
dispatch	route.js:112
handle	layer.js:95
:anonymous	index.js:281
process_params	index.js:335
next	index.js:275
jsonParser	json.js:100
handle	layer.js:95
trim_prefix	index.js:317
:anonymous	index.js:284
process_params	index.js:335
next	index.js:275
:anonymous	read.js:130

Why GraalVM? node version

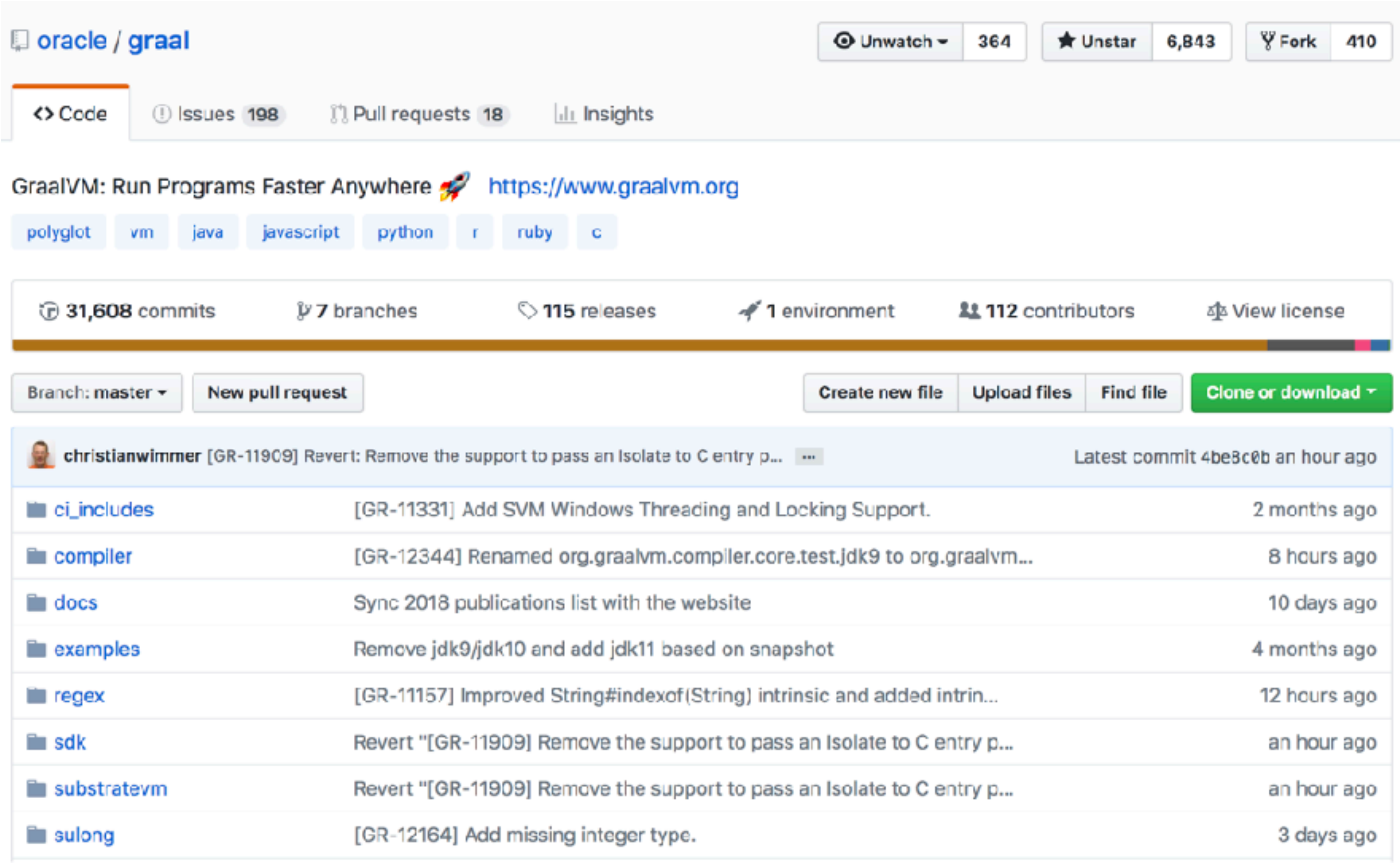
Polyglot applications: Java, Python, Ruby, R, LLVM

JVM infrastructure: threads, large heaps

Incremental rewrite of Java applications to Node.js

Building a Universal VM is a Community Effort

- Test your applications with GraalVM
  - Documentation and downloads at http://www.graalvm.org
- Connect your technology with GraalVM
  - Integrate GraalVM into your application
  - Run your own programming language or DSL
  - Build language-agnostic tools
- Join the conversation
  - Report issues or pull requests on GitHub
  - graalvm-users@oss.oracle.com
  - Follow @graalvm



GraalVM: Run Programs Faster Anywhere 🚀 https://www.graalvm.org

christianwimmer [GR-11909] Revert: Remove the support to pass an Isolate to C entry p... … Latest commit 4be3c0b an hour ago

ci_includes	[GR-11331] Add SVM Windows Threading and Locking Support.	2 months ago
compiler	[GR-12344] Renamed org.graalvm.compiler.core.test.jdk9 to org.graalvm...	8 hours ago
docs	Sync 2018 publications list with the website	10 days ago
examples	Remove jdk9/jdk10 and add jdk11 based on snapshot	4 months ago
regex	[GR-11157] Improved String#indexof(String) intrinsic and added intrin...	12 hours ago
sdk	Revert "[GR-11909] Remove the support to pass an Isolate to C entry p...	an hour ago
substratevm	Revert "[GR-11909] Remove the support to pass an Isolate to C entry p...	an hour ago
sulong	[GR-12164] Add missing integer type.	3 days ago