ORACLE

Truffle Framework

Introduction + New and Noteworthy

GraalVM Community Workshop - Christian Humer

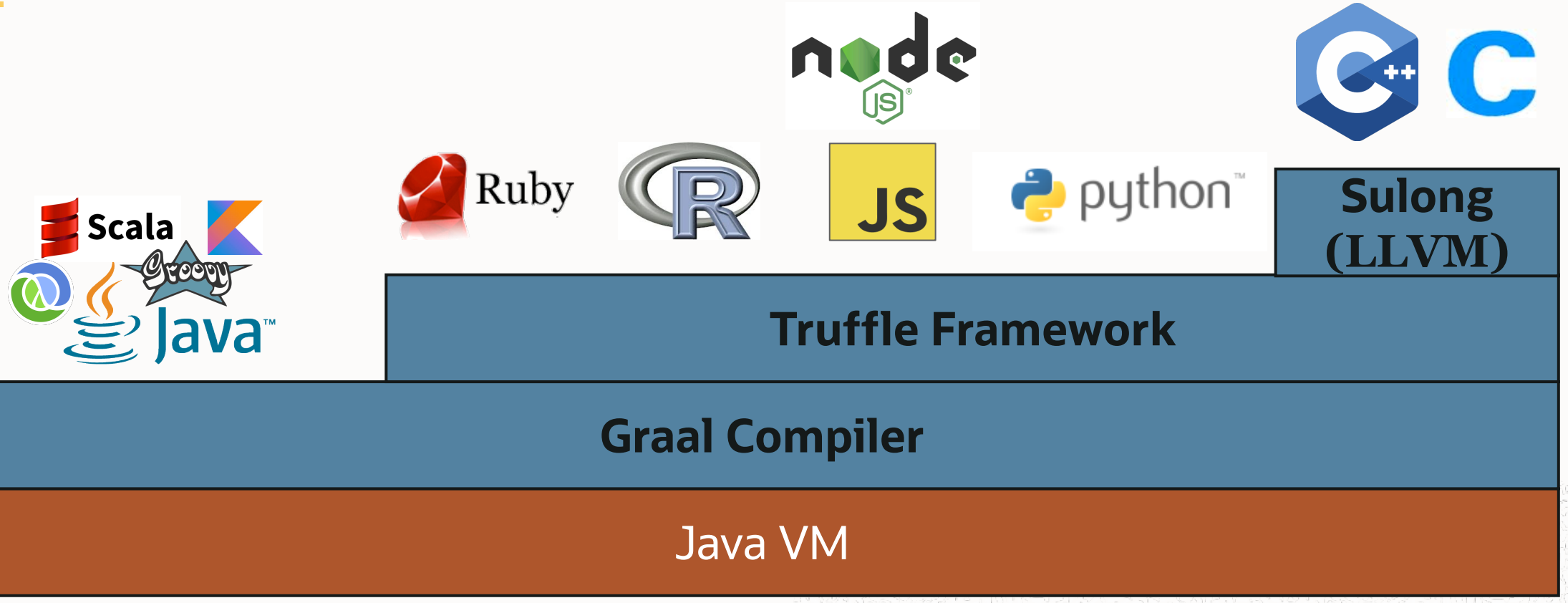
November 25, 2019

Safe harbor statement

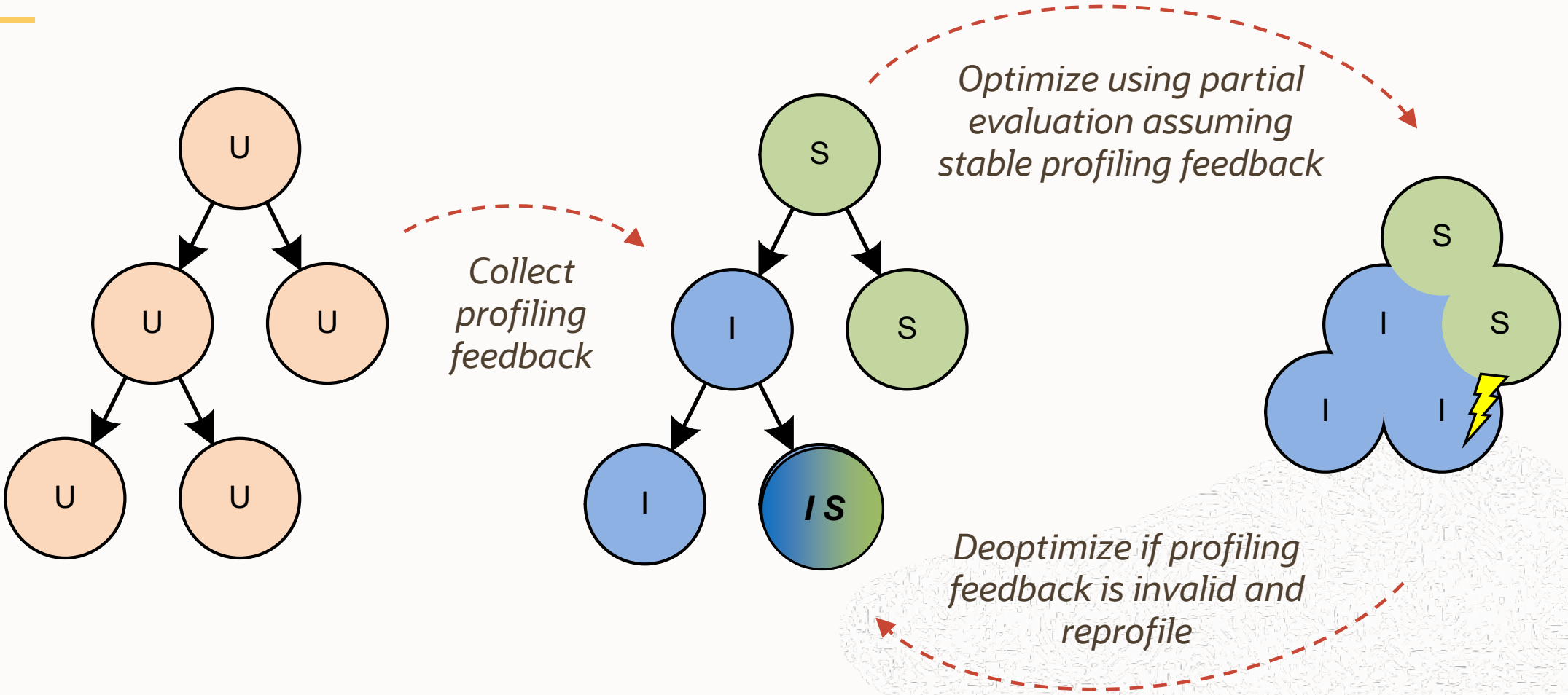
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

GraalVM Native Image technology (including Substrate VM) is Early Adopter technology. It is available only under an early adopter license and remains subject to potentially significant further changes, compatibility testing and certification.

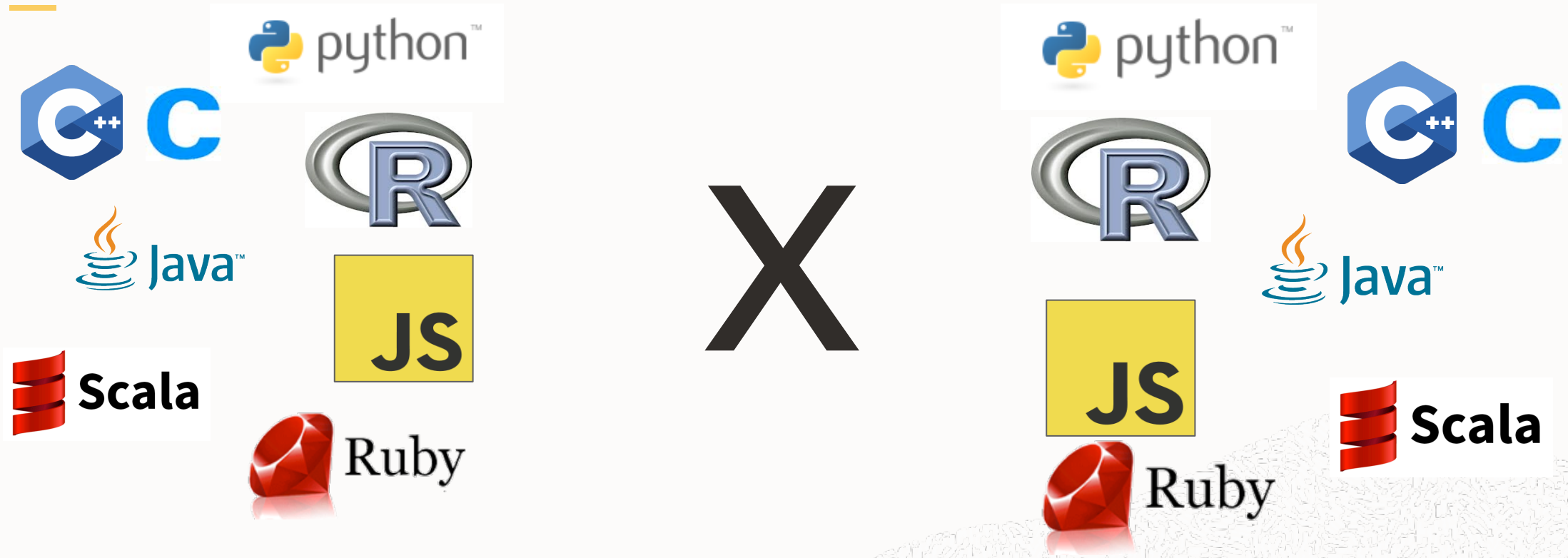
What is Truffle?



Automatic Transformation of Interpreters to Compilers



Truffle: Composing Languages



Truffle: Composing Languages

X

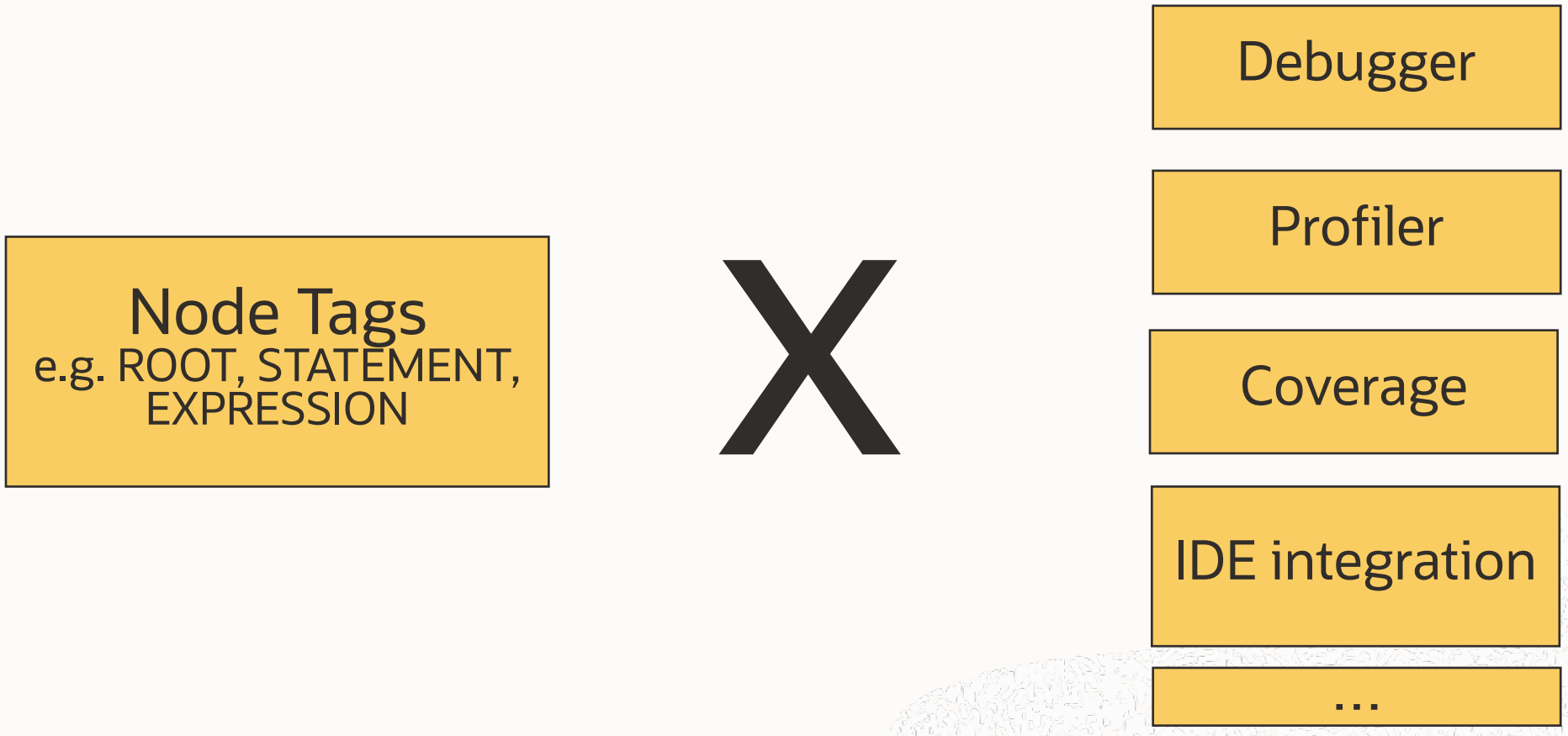
Interoperability Message Protocol

Truffle: Composing Tools

X

- Debugger
- Profiler
- Coverage
- IDE integration
- ...

Truffle: Composing Tools



Truffle: Polyglot Embeddability

```
import org.graalvm.polyglot.*;


try (Context context = Context.create()) {
    context.eval("js",     "print('Hello JavaScript!');");
    context.eval("R",      "print('Hello R!');");
    context.eval("ruby",   "puts 'Hello Ruby!'");
    context.eval("python", "print('Hello Python!')");
}
```

Embed once run all the languages.

Truffle: Focus Areas 19.0 -19.3

- Hardening the security boundary for polyglot contexts
  - Whitelist accessibility by default
- Environment Variable, Process and File System Abstractions
- Truffle Libraries: a new general framework for active libraries
  - Allows for better modularity and encapsulation in and between language implementations
- Migration of Polyglot Interoperability APIs to Truffle Libraries
  - Improved footprint and better usability

Truffle: New features in 19.3

- New embedding APIs for context specific time and statement count resource limits

```
ResourceLimits.newBuilder()
    .cpuTimeLimit(Duration.ofMillis(100),
                  Duration.ofMillis(10))
    .statementLimit(1000, null)
.build();
```

```
try (Context context = Context.newBuilder()
                    .resourceLimits(limits).build()) {
    context.eval("js", "while(true);");
    assert false;
} catch (PolyglotException e) {
    assert e.isCancelled();
}
```

- Support for temporary files and directories
- Partial Compilation with BlockNode
- New experimental language agnostic inlining heuristic.

New Inlining Example

```
function foo(a) {
    return a + a + a + a + a + a + a + a +
           ...
           a + a + a + a + a + a + a + a;
}
function bar() { return foo(22) + foo(20); }
```

```
[truffle] inline start   bar        |ASTSize    10
[truffle] inline failed    foo      |ASTSize  2814 |reason totalNodeCount > 2250
[truffle] inline failed    foo      |ASTSize  2814 |reason totalNodeCount > 2250
[truffle] inline done    bar        |ASTSize    10
```

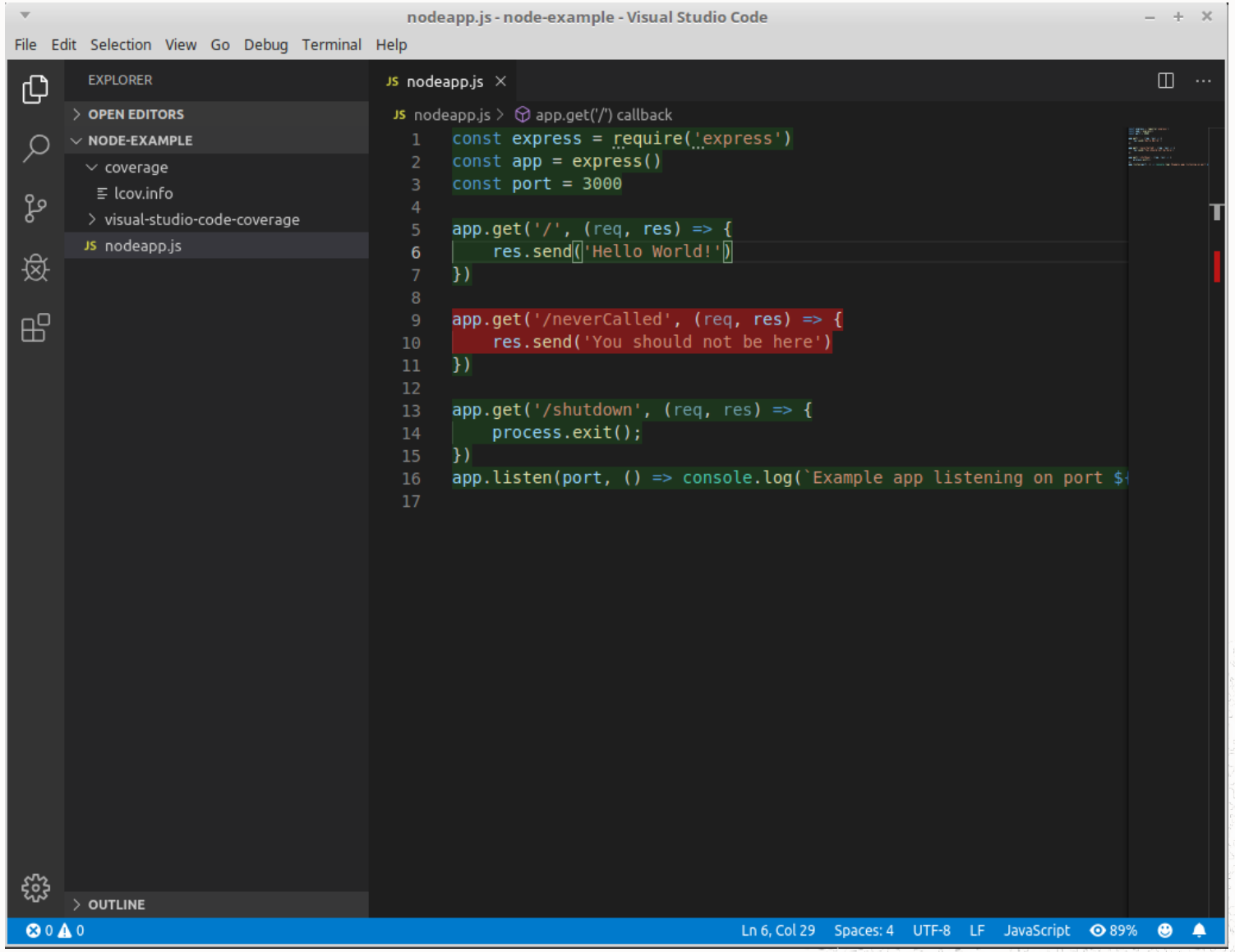
Enable New Inlining: `-Dgraal.TruffleLanguageAgnosticInlining=true`

```
[truffle] inline start   bar        |IR Nodes   162
[truffle] inline success   foo      |IR Nodes    16
[truffle] inline success   foo      |IR Nodes    16
[truffle] inline done    bar        |IR Nodes   162
```

Truffle: New features in 19.3

- New Version API for Embedders in Graal SDK
  - `Version.getCurrent().compareTo(19, 3) <= 0`
- Class Loader Isolation For JDK 11
  - No dependency conflicts with host applications.
- Code Coverage Support (`--coverage`)
  - No changes in the Truffle Languages required (uses statement and root tags)
  - `--coverage.Output = histogram | detailed | json | lcov`

Truffle Code Coverage in VS Code



Focus areas for Truffle 20.0 and Beyond

- Tooling Support for Language Server Protocol Support
- Better Interoperability (Dictionaries, Meta-Access)
- Improvements for Bytecode Interpreters
  - Instrumentation + Frames + OSR + Partial Compilation
- Memory Resource Limits
- Sharing Code/Warmup across multiple Processes
  - A new solution for polyglot FaaS platforms?