Mozilla Rhino

Attila Szegedi
@asz

History

- One of the oldest non-Java language runtimes on the JVM
- Originally part of Mozilla's all-Java browser project from 1997.
- Has its own bytecode generator engine from day one.

What is it?

- It's just JavaScript:
  - you can run any JavaScript program on top of it.
- It's more than JavaScript:
  - it integrates with the JVM. You can use Java objects and libraries.

Compatibility

- Most of ECMAScript 5
  - Everything except strict mode
- Full JS1.7 (and earlier) support
- Partial JS 1.8 support
  - expression closures, destructuring assignment
  - no generator expressions
- ECMAScript for XML (E4X)
- JSON parser/stringifier

What it isn't?

- It isn't a browser JavaScript runtime:
  - Doesn't come with HTML DOM out of the box.
  - HtmlUnit project can be used if that's needed.

Optimization settings

- Three optimization levels:
  - -1: pure interpreter
  - 0: bytecode compiler; no optimizations
  - 9: bytecode compiler with optimizations

Optimization settings

- Let's compare them using fib(30)

```
function time(f) {
  t1 = java.lang.System.currentTimeMillis()
  f()
  t2 = java.lang.System.currentTimeMillis()
  print(t2 - t1)
}

function fib(n) {
  return n < 2 ? n : fib(n-1) + fib(n-2)
}

time(function() { fib(30) })
```

Optimization settings

- Let's compare them using fib(30)
- Interpreted took 14 seconds
- Opt 0 took 0.9 seconds
- Opt 9 took 0.3 seconds

When to use which?

- Opt 9 in most cases
- Opt 0 - don't really have a modern use case for it
- Opt -1: constrained devices, environments that don't allow you to load generated bytecode at runtime, or when you need continuations*.

CommonJS

- Since Rhino 1.7R3, we have CommonJS Modules support.
- Code loading mechanism, originally for non-browser JS engines.

Integrating with Java

- Packages “com”, “org”, “net”, “java”, “javax” exposed as top-level objects.
- Just do “new javax.swing.JFrame” or such.
- Other packages available as “Packages.*”

Bundled Rhino

- Rhino is also shipping as standard part of the JRE since Java 6.

Continuations?

- Think of them as an object representing the stack at the point of execution.
- Can store them away and resume them any number of times.
- In a web app, can cure the lame “don’t click the back button” syndrome.

Future direction

- Invokedynamic (preferrably via Dynalink)
- Performance improvements
- Recent change in string concatenation sped up SunSpider performance 3x
-

Where to get it

- Stable releases are at http://mozilla.org/rhino
- Source code repository used to be in Mozilla CVS
- Now migrated to https://github.com/mozilla/rhino
  - Fork it!

Other contenders

- Nashorn, Oracle's new invokedynamic based JS runtime
- dyn.js, another invokedynamic based JS runtime at http:/github.org/dynjs/dynjs

Questions?

Mozilla Rhino

Attila Szegedi
@asz