Node4J

Running **Node.js** in a Java World

Dr. R. Ian Bull
EclipseSource

@irbull
Java and JavaScript

- Java a successful server side language
- JavaScript is a client side language
- SWT brought performant Java UIs to the desktop
- Node.js brought JavaScript to the server
- Java and JavaScript are two of the most popular programming languages
Polyglot Systems

- Single language systems are rarely an option
  - Legacy code
- New frameworks and technologies
- Evolving enterprises
- JEE will be here for another 20, 30, 50 (?) years
Bridging Java and JavaScript

- Three common Java technologies enable JS embedding
  - **Rhino**
    - Available since JDK 6
  - **Nashorn**
    - Replacing Rhino since JDK 8
    - More performant
  - V8 as a *separate process*, String based messages
Performance

- 30 Runs of the **Esprima parser and tokenizer**
- Nashorn compiles to **bytecode**
- V8 compiles to **native assembly**
- Best choice for raw JavaScript execution
A set of **bindings** that bring V8 to Java

- Inspired by SWT
  - Create a thin **JNI layer**
  - Expose (some) **V8 API** in Java
- Complicated logic lives in Java
J2V8 Goals

- Efficient JavaScript on **Android**
- Make **JavaScript** shine in an enterprise **Java** World
- **Standard** Java APIs
- **Efficient** Java / JavaScript bindings
J2V8 — History

- 1.0 Released in November 2014
- 2.0 Released in February 2015
- First presented at EclipseCon 2015
- 3.0 Released at EnterJS — Summer 2015
J2V8 Design

- Each V8 Object can be referenced using a Handle
- Each Object is stored in a V8 Persistent Object Store
- Objects must be explicitly freed
- Primitives where possible (no wrappers)
- Single Thread per isolate
Two-way binding

- **JS functions** and **scripts** can be invoked from Java
- **Java methods** can be called from JavaScript
- Data can be passed back and forth using **V8Objects**
J2V8 In Action — Tabris.js

- Mobile framework
- Apps written in JavaScript
- Native iOS and Android Apps
- Bindings to native UI components
Shameless Plug
```java
public String someJavaMethod(final String firstName, final String lastName) {
    return firstName + ", " + lastName;
}

public void start() {
    V8 v8 = V8.createV8Runtime();
    v8.registerJavaMethod(this,
        "someJavaMethod",
        "someJavaMethod",
        new Class[] { String.class, String.class });
    v8.executeScript("var result = someJavaMethod('Ian', 'Bull');");
    String result = v8.getString("result");
    System.out.println(result);
    }
```
J2V8 — What’s New

- Typed Arrays
- Threads & Workers
- ES 6
- ChromeDev Tools
- NodeJS Support
Typed Arrays

- Native support for JS **Typed Arrays**
- **Access** the values **efficiently** from Java

```javascript
V8Array result = (V8Array) v8.executeScript(""
+ "var buf = new ArrayBuffer(100);"
+ "var ints = new Int32Array(buf); "
+ "for(var i = 0; i < 25; i++) {
+  ints[i] = i;
+  }"); "
+ "ints");

int[] ints = result.getIntegers(0, 25);
```
Threads

- Every thread can have its own *Isolate* (Isolated V8 Instance)
- V8Thread is a Java Thread with an associated Isolate
- Provide an easy way to execute JavaScript

```java
Thread t = new V8Thread(new V8Runnable() {
    public void run(V8 v8) {
        int result = v8.executeIntegerScript("1+2");
    }
});
t.start();
```
Executors

- Long running V8Thread with a message queue and event loop
- Threads can communicate via message passing
- Useful for implementing Web Workers / Service Workers
ES 6

- Snapshot builds of J2V8 support **V8 4.10 & ES 6**
  - Arrows
  - Classes
  - Let / Const
  - Interators + For..Of
  - Generators
  - ...

Debug Support

- V8 (and now J2V8) no longer supports the Debug Agent
- JavaScript based Debug API is available instead
- J2V8 exposes this API in Java
- Integrated with the Stetho tool & Chrome Dev Tools
Debug Support Demo

Hello, World!

Native Widgets

Totally Rock x2!
Node.js® is a JavaScript runtime built on Chrome’s V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.

- JavaScript Virtual Machine (V8)
- Modules
  - Native
  - JavaScript
- Event Loop
Bridging to Node.js

- Out of process Node & REST Services
- Vert.x
- Node engine on Nashorn / Rhino?
Node4J

- Dynamically link Node.js to the JVM
- Access Node.js context via JNI
- Execute Node.js modules (require)
- Callbacks to Java
- Process Node.js message queue
public static void main(final String[] args) throws Exception {
    final V8 v8 = V8.createV8Runtime("global");
    v8.registerJavaMethod(...);

    NodeJS node = V8.createNodeJS(v8);
    V8Object exports = node.requireScript(nodeCode, "http");
    exports.release();

    boolean running = true;
    while (running) {
        running = node.pumpMessageLoop();
    }
}
Performance Considerations

- **Minimize** callbacks from JavaScript to Java
  - ~4000 Per Second on my MBP
- Use bulk array copy to move **primitives** from JS to Java
- **60fps** in our animation demo
Resources

- Getting started with J2V8
- Registering Java Callbacks with J2V8
- Implementing WebWorkers with J2V8
- Multithreaded JavaScript with J2V8
- Using J2V8 with Heroku

All linked from our GitHub Page
Future Work

- Advanced *exception handling* between Java and JS
- Improved debug support
- **Typed array** access in Java
- *You tell me?*
Using J2V8

- J2V8 is available in **Maven Central**

- **Currently 5 variants are available:**
  
  com.eclipsesource.j2v8.j2v8_win32_x86:3.1.6
  com.eclipsesource.j2v8.j2v8_macosx_x86_64:3.1.6
  com.eclipsesource.j2v8.j2v8:3.1.6 (aar)
  com.eclipsesource.j2v8.j2v8_android_armv7l:3.1.6
  com.eclipsesource.j2v8.j2v8_android_x86:3.1.6

- j2v8:3.1.6 (aar) contains both **x86** and **armv7l**
Thank-you

- Open Source Java bindings for V8
- Node4J extensions bring Node.js to Java
- Licensed under the EPL
- For J2V8 news, follow me on Twitter @irbull

https://github.com/eclipsesource/j2v8
Did you enjoy this session?

👍 👎

Best talk at EclipseCon (so far)

SUBMIT