Nashorn: a JavaScript Engine on the JVM

A JavaScript Engine for the JVM

José Paumard
blog.paumard.org
@JosePaumard

Module Outline

- **REPL: Java in JavaScript**

Module Outline

- REPL: Java in JavaScript
- ScriptEngine: Java in JavaScript

Module Outline

- REPL: Java in JavaScript
- ScriptEngine: Java in JavaScript
- JavaScript and JavaFX

A JavaScript REPL

What Is a REPL?

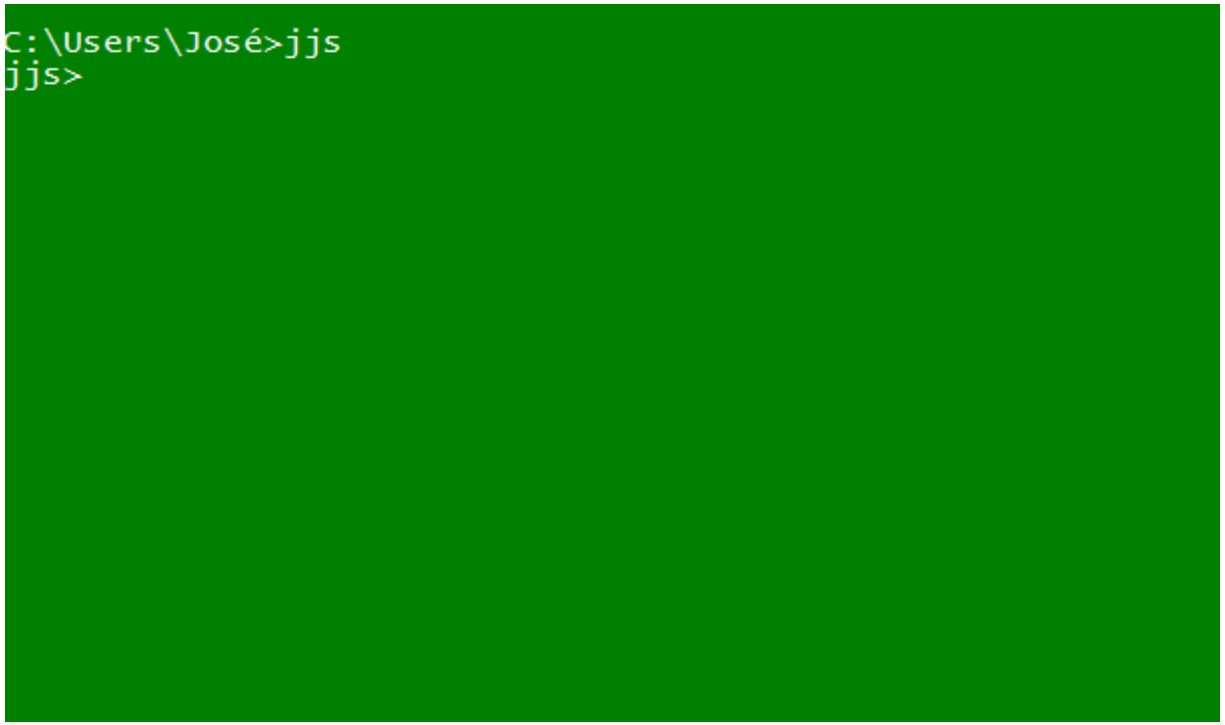
- REPL = Read, Eval, Print Loop
- It looks like a shell, ie with a prompt
- And enables one to type in JavaScript interactively

What Is a REPL?

- REPL = Read, Eval, Print Loop
- It looks like a shell, ie with a prompt
- And enables one to type in JavaScript interactively
- jjs is the REPL executable
- It is located in $JAVA_HOME/bin, in the same place as javac or java

What Is a REPL?

- REPL = Read, Eval, Print Loop
- It looks like a shell, ie with a prompt
- And enables one to type in JavaScript interactively



What Is a REPL?

- REPL = Read, Eval, Print Loop
- It looks like a shell, ie with a prompt
- And enables one to type in JavaScript interactively

```
jjs> 'Hello world!'.length()
12
jjs>
```

What Is a REPL?

- REPL = Read, Eval, Print Loop
- It looks like a shell, ie with a prompt
- And enables one to type in JavaScript interactively

```
jjs> function fibo(n) { return n <= 1 ? n : n + fibo(n - 1) }
function fibo(n) { return n <= 1 ? n : n + fibo(n - 1) }
jjs>fibo(100)
5050
jjs>
```

What Is a REPL?

- REPL = Read, Eval, Print Loop
- It looks like a shell, ie with a prompt
- And enables one to type in JavaScript interactively

```
jjs> function fibo(n) { return n <= 1 ? n : n + fibo(n - 1) }
function fibo(n) { return n <= 1 ? n : n + fibo(n - 1) }
jjs>fibo(100)
5050
jjs>function fact(n) { return n <= 1 ? n : n*fact(n - 1) }
function fact(n) { return n <= 1 ? n : n*fact(n - 1) }
jjs>fact(5)
120
jjs>
```

What Is a REPL?

- One can create Java objects and interact with them

```
jjs>var s = new java.lang.String("Hello")
jjs>s
Hello
jjs>
```

What Is a REPL?

- One can create Java objects and interact with them

```
jjs>var s = new java.lang.String("Hello")
jjs>s
Hello
jjs>s.toUpperCase()
HELLO
jjs>
```

The REPL

- The Nashorn REPL allows to interactively type in an execute Java and JavaScript

Running JavaScript in Java code

Running JavaScript in a Java Application

- Java has been supporting script engines since Java 6 (2006)
- Many languages are available: Groovy and JRuby

Running JavaScript in a Java Application

- Java has been supporting script engines since Java 6 (2006)
- Many languages are available: Groovy and JRuby
- One needs to get a script engine by its name

```
ScriptEngineManager manager = new ScriptEngineManager();

ScriptEngine engine = manager.getEngineByName("nashorn");
```

Running JavaScript in a Java Application

- Java has been supporting script engines since Java 6 (2006)
- Many languages are available: Groovy and JRuby
- One needs to get a script engine by its name

```
ScriptEngineManager manager = new ScriptEngineManager();

ScriptEngine engine = manager.getEngineByName("nashorn");
```

- This object is used to interact with the JavaScript interpreter

```
Object result = engine.eval("/* JavaScript code here */");
```

Running JavaScript in a Java Application

- Java has been supporting script engines since Java 6 (2006)
- Many languages are available: Groovy and JRuby
- One needs to get a script engine by its name

```
ScriptEngineManager manager = new ScriptEngineManager();

ScriptEngine engine = manager.getEngineByName("nashorn");
```

- This object is used to interact with the JavaScript interpreter

```
Object result = engine.eval("/* JavaScript code here */");
```

- One can also pass JavaScript code through a file

```
Object result = engine.eval(Files.newBufferedReader(path));
```

How to Pass Objects to JavaScript

- Two ways of passing Java objects to the JavaScript engine
- Suppose we want to pass the Stage object (from JavaFX)
- 1st solution:

```
public void start(Stage stage) {
   engine.put("stage", stage);
   engine.eval(script); // script is my JavaScript code
}
```

- In this case the stage variable is available in the JavaScript « global scope »

How to Pass Objects to JavaScript

- Two ways of passing Java objects to the JavaScript engine
- Suppose we want to pass the Stage object (from JavaFX)
- 2nd solution: we want to scope our variable

```
Bindings scope = engine.createBindings();
scope.put("stage", stage);
engine.eval(script, scope);
```

- In this case the stage variable is only available in the scope defined by the scope object

Invoking Getters and Setters

- The JavaFX stage object has a property named title
- In Java a property = a getter and a setter

Invoking Getters and Setters

- The JavaFX stage object has a property named title
- In Java a property = a getter and a setter
- In JavaScript one can write this:

```
stage.setTitle('This is JavaScript!')
```

Invoking Getters and Setters

- The JavaFX stage object has a property named title
- In Java a property = a getter and a setter
- In JavaScript one can write this:

```
stage.setTitle('This is JavaScript!')
```

- But also this:

```
stage.title = 'This is JavaScript!'
```

Invoking Getters and Setters

- The JavaFX stage object has a property named title
- In Java a property = a getter and a setter
- In JavaScript one can write this:

```
stage.setTitle('This is JavaScript!')
```

- But also this:

```
stage.title = 'This is JavaScript!'
```

- And also this:

```
stage['title'] = 'This is JavaScript!'
```

Nashorn and JavaFX

Lauching a JavaFX Application Through Nashorn

- One can use jjs to launch a JavaFX application

```
$ jjs –fx myJavaFXApp.js
```

Lauching a JavaFX Application Through Nashorn

- One can use jjs to launch a JavaFX application

```
$ jjs –fx myJavaFXApp.js
```

- Nashorn will make the stage object available through $STAGE

```
var message =
   new javafx.scene.control.Label("This is JavaScript!");
message.font =
   new javafx.scene.text.Font(100);
$STAGE.scene = new javafx.scene.Scene(message);
$STAGE.title = "Hello World!";
```

Summary

- **Quick overview of Java / JavaScript integration using Nashorn**
- **How to type in JavaScript code through the REPL jjs**
  - How to use Java objects and classes in the JavaScript code
- **How to evaluate JavaScript code in a Java application**
  - How to pass Java objects in the JavaScript code
- **How to create a JavaFX application using JavaScript**

Course Summary

- **Lamba expressions**
  - Anonymous class, functional interfaces, method references, collection API

Course Summary

- **Lamba expressions**
  - Anonymous class, functional interfaces, method references, collection API
- **Streams & Collectors**
  - Map / filter / reduce, patterns to build a stream, operations on a Stream

Course Summary

- **Lamba expressions**
  - Anonymous class, functional interfaces, method references, collection API
- **Streams & Collectors**
  - Map / filter / reduce, patterns to build a stream, operations on a Stream
- **Java Date & Time API**
  - Instance / Duration, LocalDate / Period, LocalTime, Zoned Time

Course Summary

- **Lamba expressions**
  - Anonymous class, functional interfaces, method references, collection API
- **Streams & Collectors**
  - Map / filter / reduce, patterns to build a stream, operations on a Stream
- **Java Date & Time API**
  - Instance / Duration, LocalDate / Period, LocalTime, Zoned Time
- **Strings, I/O, and other Bits & Pieces**
  - Strings, I/O, Collection, Comparators, Numbers, Maps, Annotations

Course Summary

- **Lamba expressions**
  - Anonymous class, functional interfaces, method references, collection API
- **Streams & Collectors**
  - Map / filter / reduce, patterns to build a stream, operations on a Stream
- **Java Date & Time API**
  - Instance / Duration, LocalDate / Period, LocalTime, Zoned Time
- **Strings, I/O, and other Bits & Pieces**
  - Strings, I/O, Collection, Comparators, Numbers, Maps, Annotations
- **Java FX**

Course Summary

- **Lamba expressions**
  - Anonymous class, functional interfaces, method references, collection API
- **Streams & Collectors**
  - Map / filter / reduce, patterns to build a stream, operations on a Stream
- **Java Date & Time API**
  - Instance / Duration, LocalDate / Period, LocalTime, Zoned Time
- **Strings, I/O, and other Bits & Pieces**
  - Strings, I/O, Collection, Comparators, Numbers, Maps, Annotations
- **Java FX**
- **Nashorn and JavaScript**