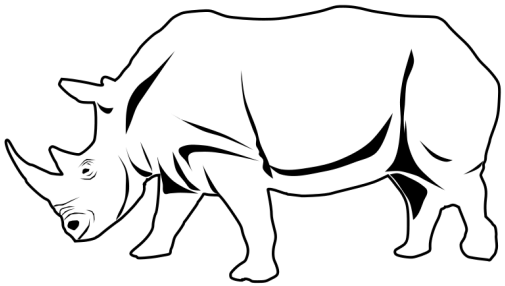
Nashorn

Java(Script)



Mitia Alexandrov

DAYS
CLOUD Days
mobile day
3 CONFERENCES
2 DAYS
1 PLACE
17-19 NOVEMBER 2014
INTER EXPO CENTER (IEC), SOFIA

About me

Unlike the other speakers…

During the day:

During the night:

But I do:

Was ist JavaScript?

- Designed in 15 days by Brendan Eich in 1995
- Dialect of ECMAScript (yes, they are not synonyms)
- Multi-paradigm: prototype-based, functional, imperative, scripting
- Now in every browser!

Was ist Nashorn?

- Oracles runtime for ECMAScript 5.1
- GPL licensed
- Part of OpenJDK
- Released this march!
- Just type jjs in the shell… and you are in!

Why?

Atwoods law: any application that *can* be written in **JavaScript**, *will* eventually be written in **JavaScript**

Why?

- Oracle proving ground for support of dynamic languages.
- Shipped as part of JRE.
- Currently supports ECMAScript 5.1 (but 100%). No backwards compatibility.

Why not Rhino

- All code compiled to bytecode. No interpretation. .. yes, even in console!
- The biggest consumer of “InvokeDynamic” in the world!
- JSR-223 javax.script.* is the only public API.

Part eins:

Java -> JavaScript

Simple:

```
import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager;
import javax.script.ScriptException;

public class EvalScript {
   public static void main(String[] args) throws Exception {
      // create a script engine manager
      ScriptEngineManager factory = new ScriptEngineManager();
      // create a Nashorn script engine
      ScriptEngine engine = factory.getEngineByName("nashorn");
      // evaluate JavaScript statement
      try {
         engine.eval("print('Hello, World!');");
      } catch (final ScriptException se) { se.printStackTrace(); }
   }
}
```

Multiple invoke

```
import javax.script.Invocable;
import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager;

public class Eval_Invocable {
   public static void main(String[] args) throws Exception {
      ScriptEngineManager factory = new ScriptEngineManager();
      ScriptEngine engine = factory.getEngineByName("nashorn");

      engine.eval("function f(x){return x < 2 ? 1 : f(x-1)*x} ");
      engine.eval("function sum(x,y){return x+y}");

      Invocable i = (Invocable) engine;

      System.out.println(i.invokeFunction("f",5));
      System.out.println(i.invokeFunction("f",10));
      System.out.println(i.invokeFunction("sum",5,10));
      System.out.println(i.invokeFunction("sum",10,15));
   }
}
```

Implement interface

```
public interface Adder {
   int sum(int a, int b);
}

public class Eval_InvocableInterface {
   public static void main(String... args) throws Exception{
      ScriptEngine engine = new
      ScriptEngineManager().getEngineByName("nashorn");
      engine.eval("function sum(x,y){return x+y}");

      Invocable i = (Invocable) engine;

      Adder adder= i.getInterface(Adder.class);
      System.out.println(adder.sum(1,3));
   }
}
```

Passing variables

```
public class Eval_Bind {
   public static void main(String... args) throws Exception {

     ScriptEngine engine = new
     ScriptEngineManager().getEngineByName("nashorn");

      Bindings b = engine.createBindings();
      b.put("file", new File("/"));
      engine.eval("list = file.listFiles()", b);

      System.out.println(Arrays.toString((File[]) b.get("list")));
   }
}
```

Part deux:

JavaScript -> Java

Simple

```
var timer = new java.util.Timer();

timer.schedule(
    new java.util.TimerTask({
        run: function(){
            print("Tick")
        }
    })
    ,0,1000)
java.lang.Thread.sleep(5000)
timer.cancel();
```

Even more simple

```
var timer2= new java.util.Timer();

timer2.schedule(function(){print("Tack")},0,1000)

java.lang.Thread.sleep(5000)
timer2.cancel();
```

Construct Java(Script) object

- var linkedList = new java.util.LinkedList()

- var LinkedList = java.util.LinkedList
  var list = new LinkedList()

- var LinkedList = Java.type("java.util.LinkedList")
  var list = new LinkedList()

Types

```
var ints = new (Java.type("int[]"))(6)
ints[0]=1
ints[1]=1.6
ints[2]=null
ints[3]="45"
ints[4]="str"
Ints[5]=undefined
print(ints)
print(java.util.Arrays.toString(ints))
```

Output will be:

[I@43re2sd

[1, 1, 0, 45, 0, 0]

Types 2

```
Var dbls = new (Java.type("double[]"))(6)
dbls [0]=1
dbls [1]=1.6
dbls [2]=null
dbls [3]="45"
dbls [4]="str"
dbls [5]=undefined
print(dbls )
print(java.util.Arrays.toString(dbls ))
```

Output will be:

[D@43re2sd

[1.0, 1.6, 0.0, 45.0, NaN, NaN]

Type conversion

- Passing JavaScript values to Java methods will use all allowed Java method invocation conversions… + all allowed JavaScript conversions
- All native JS objects implement java.util.Map
- And they do not implement java.util.List!

Arrays conversions

- Not converted automatically!

- Explicit APIs provided:
  - var javaArray = Java.toJavaArray(jsArray,type)
  - var jsArray = Java.toJavaScriptArray(javaArray)

Type conversion 2

Consider:

```
static void about(Object object) {
    System.out.println(object.getClass());
}
```

```
MyJavaClass.about(123);
// class java.lang.Integer

MyJavaClass.about(49.99);
// class java.lang.Double

MyJavaClass.about(true);
// class java.lang.Boolean

MyJavaClass.about("hi there")
// class java.lang.String

MyJavaClass.about(new Number(23));
// class jdk.nashorn.internal.objects.NativeNumber

MyJavaClass.about(new Date());
// class jdk.nashorn.internal.objects.NativeDate

MyJavaClass.about(new RegExp());
// class jdk.nashorn.internal.objects.NativeRegExp

MyJavaClass.about({foo: 'bar'});
// class jdk.nashorn.internal.scripts.JO4
```

Static accesses

```
var ps = java.io.File.pathSeparator
print(ps)
var File = Java.type("java.io.File")
var p = File.separator
print(p)
var MapEntry = java.util.AbstractMap.SimpleEntry
print(MapEntry)
var m = new MapEntry(1,"b");
print(m)
```

Output:

:

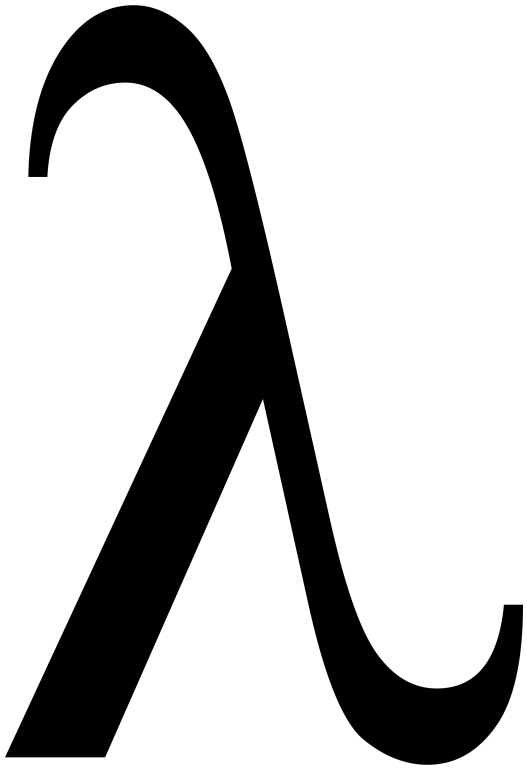
/

[JavaClass java.util.AbstractMap$SimpleEntry]

1=b

By the way.. Its Java 8

By the way.. Its Java 8



By the way.. Its Java 8

lambdas lambdas lambdas lambdas lambdas
lambdas lambdas lambdas lambdas lambdas
lambdas lambdas lambdas lambdas lambdas
lambdas lambdas lambdas lambdas lambdas
lambdas lambdas lambdas lambdas lambdas
lambdas lambdas lambdas lambdas lambdas
lambdas lambdas lambdas lambdas lambdas
lambdas lambdas lambdas lambdas lambdas
lambdas lambdas … and default methods

DAYS
CLOUD Days
mobile day
3 CONFERENCES
2 DAYS
1 PLACE
17-19 NOVEMBER 2014
INTER EXPO CENTER (IEC), SOFIA

..or

By the way.. Its Java 8

JavaScript:

```
var stack = new java.util.LinkedList();
[1, 2, 3, 4,54,87,42,32,65,4,5,8,43].forEach(function(item) {
   stack.push(item);
});

print(stack.getClass());

var sorted = stack
   .stream()
   .filter(function(i){return i%2==0})
   .sorted()
   .toArray();
print(java.util.Arrays.toString(sorted));
```

Output:

```
[Ljava.lang.Object;@6591f517
[2, 4, 4, 8, 32, 42, 54]
```

Java 8…default methods

Java:

```
public interface DefTest {
default void sayHello(String name){
        System.out.println("Hello "+name);
    }
}

public class DefTestImpl implements DefTest {
    //nothing here
}
```

JavaScript:

```
>jjs -cp .
jjs> var DT = Java.type("DefTestImpl")
jjs> DT
[JavaClass DefTestImpl]
jjs> var dt = new DT()
jjs> dt.sayHello("World")
Hello World
```

Part one before the last:

JavaScript(ing)

Scripting extensions

- Additional classpath elements can be specified for the Java Virtual Machine (JVM).
- JavaScript strict mode can be activated.
- An *intriguing scripting mode* can be enabled.

Shell invocations

JavaScript:

```
var lines =
   'ls -lsa'.split("\n");
for each (var line in lines) {
   print("|> " + line);
}
```

Output:

```
buzz:Nashorn mitia$ jjs -scripting scripting.js
|> total 112
|> 0 drwxr-xr-x  16 mitia  staff   544 21 апр 21:39 .
|> 0 drwxr-xr-x   3 mitia  staff   102 19 апр 14:26 ..
|> 8 -rw-r--r--   1 mitia  staff   113 21 апр 21:32 Adder.class
|> 8 -rw-r--r--   1 mitia  staff   603 21 апр 21:32 DefTest.class
|> 8 -rw-r--r--   1 mitia  staff   273 21 апр 21:39 DefTestImpl.class
|> 8 -rw-r--r--   1 mitia  staff  1001 21 апр 21:32 EvalScript.class
|> 8 -rw-r--r--   1 mitia  staff  1379 21 апр 21:32 Eval_Bind.class
|> 8 -rw-r--r--   1 mitia  staff  1402 21 апр 21:32 Eval_Invocable.class
```

Heredocs

JavaScript:

```
var data = {
        foo: “bar”,
        time: new Date()
};

print(<So...      foo = ${data.foo} and the current time is      ${data.time} EOF);
```

Output:

```
$  jjs -scripting heredocs.js
So...
     foo = bar
and the current time is
     Thu Aug 01 2013 16:21:16 GMT+0200 (CEST)
```

Shell invocations

JavaScript:

```
#!/usr/bin/env jjs -scripting
print(
   "Arguments (${$ARG.length})");
for each (arg in $ARG) {
   print("- ${arg}")
}
```

Output:

```
$ chmod +x executable.js
$ ./executable.js
Arguments (0)
$ ./executable.js -- hello world !
Arguments (3)
- hello
- world
- !
```

Part last:

All together for the enterprise

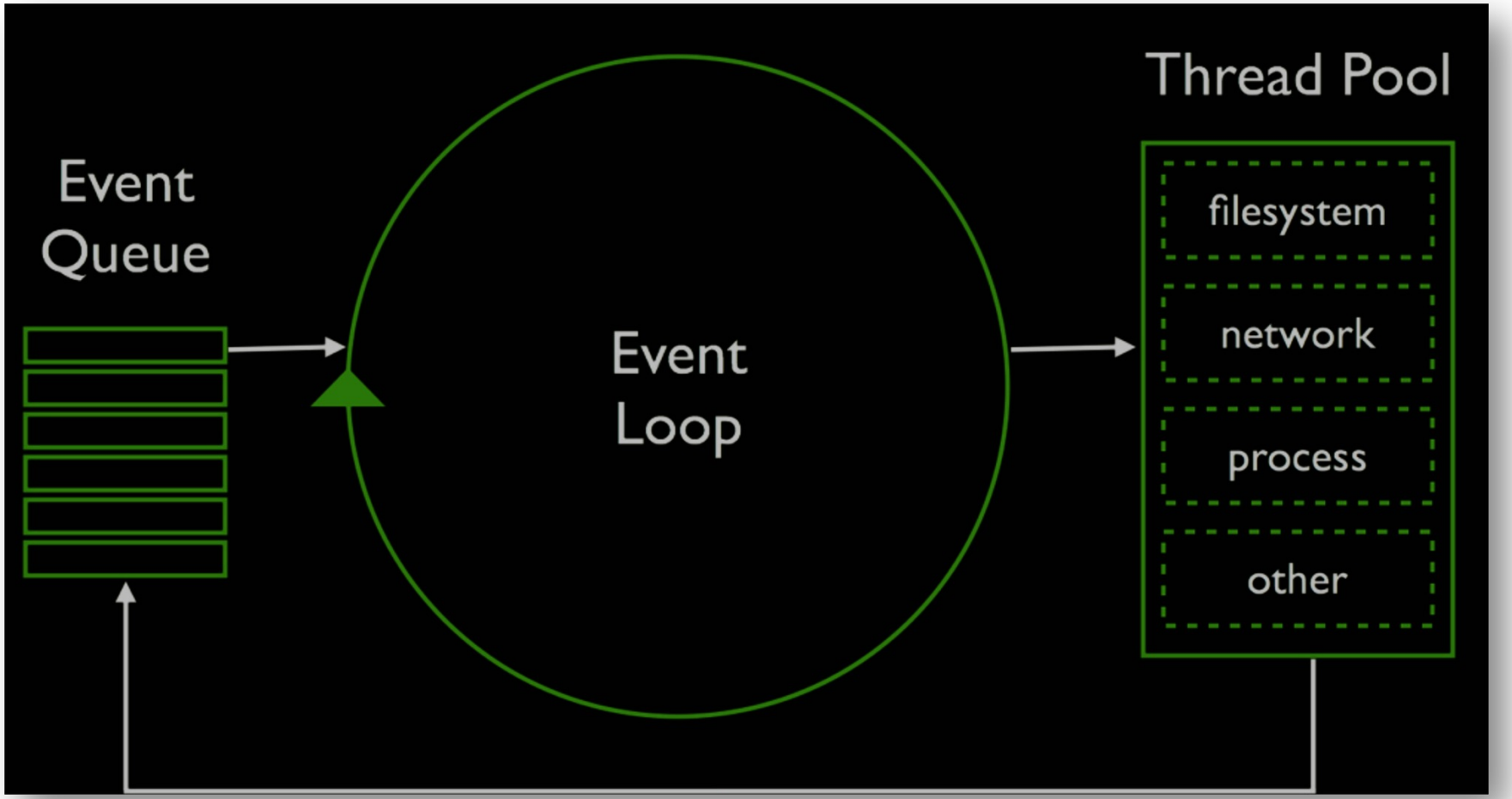
Modern Landscape

- With HTML5, CSS3 and fast JavaScript engines, modern browsers have become a powerful platform
- “Thin Server Architecture” (TSA) & “Single Page Architecture” rise
- …but they all spoke different languages until node.js

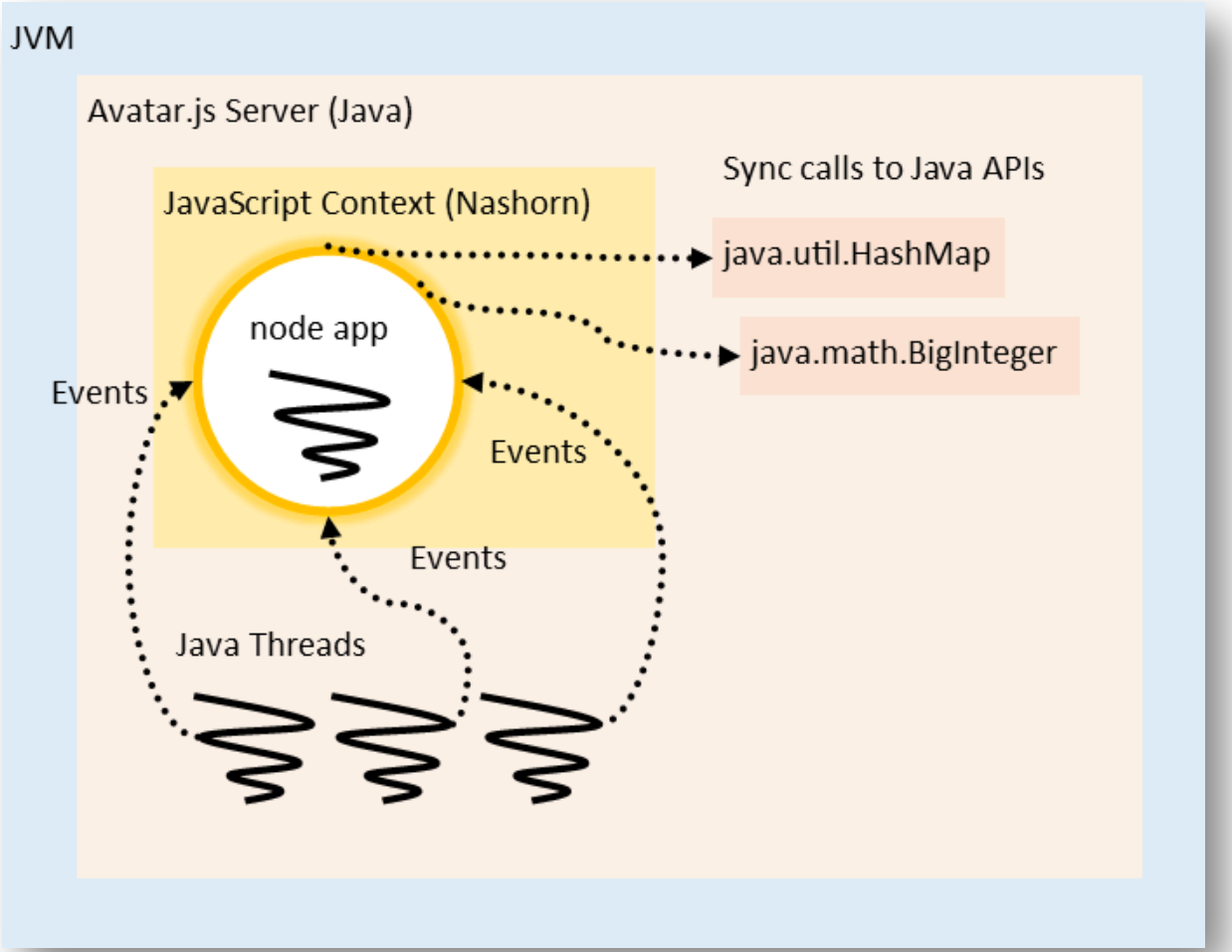
Project Avatar

- JavaScript services layer zeroed in on supporting REST, WebSocket and Server-Sent Events and a rich client side framework
- Not a replacement for Java based implementations;
- instead, they are an *additional* choice, additional abstraction
- Pure Java services also work well with Avatar clients.

Node.js event loop



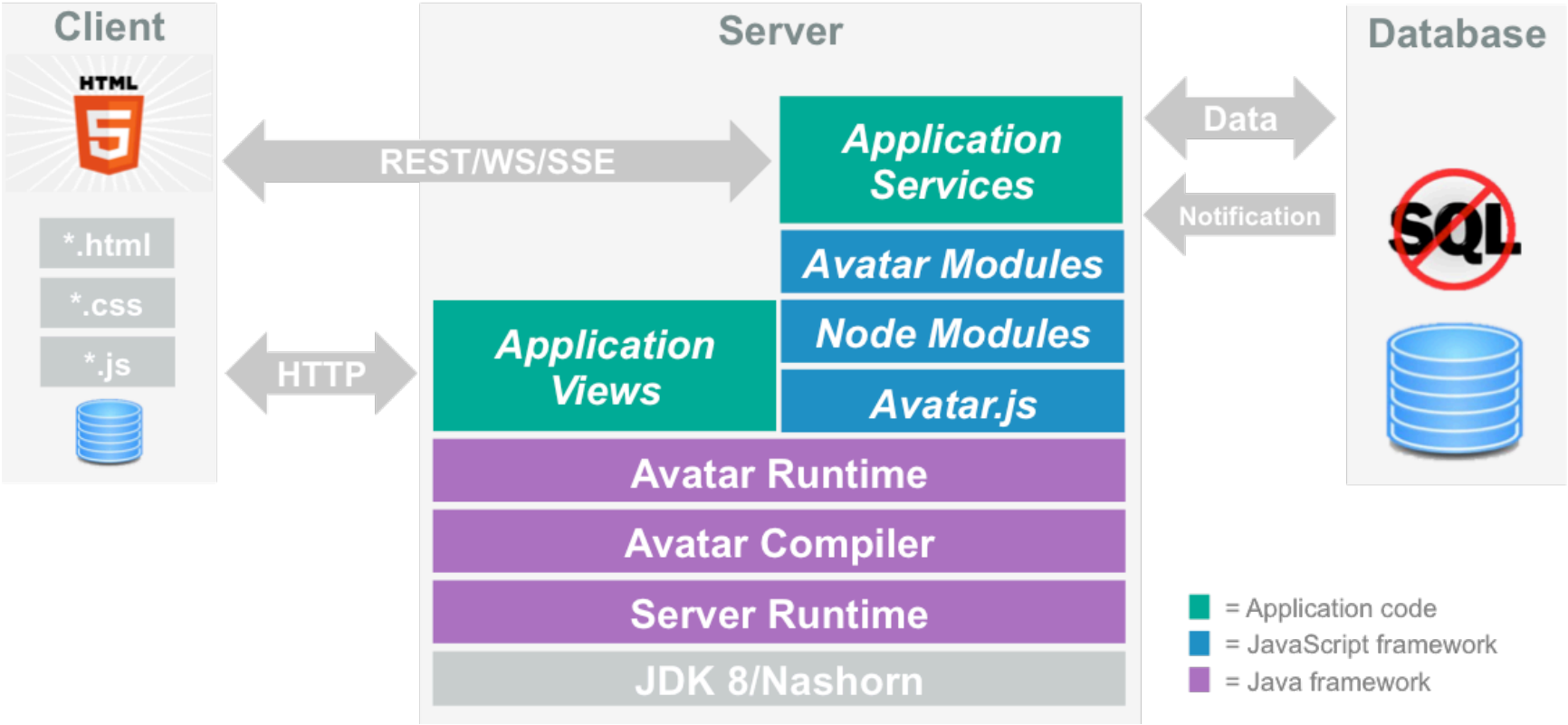
… and how it is in Avatar.js



DAYS
CLOUD Days
mobile day
3 CONFERENCES
2 DAYS
1 PLACE
17-19 NOVEMBER 2014
INTER EXPO CENTER (IEC), SOFIA

Demo

Architecture



Avatar application

- …or an archive
- Simple directory structure containing an avatar.properties and WEB-INF directory
- Has Views directory or a Services directory, or both
- Avatar contains a built-in “system” archive

Avatar "Services"

- An Avatar service simply extends either a REST, WebSocket or Push (SSE) abstraction.
- Instantiated and invoked on demand. Similar in spirit to the Servlet model.
- Service implementations can leverage built-in Node modules as well as most third-party modules.
- Since Nashorn supports direct invocation of Java code, the rich set of Java libraries is also available.

Avatar Client

- View
  - Declarative UI
  - Bundle widgets (jQuery-ui/mobile)

- Model
  - WS, REST, SSE, local
  - Autowire with services

Avatar Client

Server

```
var avatar = require("org/glassfish/avatar");

var getTime = function() {
    var current = new Date();
    return {
        msg: 'The server time is ',
        h: current.getHours(),
        m: current.getMinutes(),
        s: current.getSeconds() };
};

avatar.registerRestService({ url: "data/time" },
    function() {
        this.onGet = function(request, response) {
            return response.send(getTime());
        };
    }
);
```

Client

```
<html>
    <head>...</head>
    <body>
        <script data-model="rest">
            var Message = function() {
                this.msg = this.h = this.m = this.s = '';
            };
        </script>
        <script data-type="Message" data-instance="message" data-url="data/time"></script>
        <output class="time">#{message.msg}#{message.h}:#{message.m}:#{message.s}</output><br><br>
        <button onclick="#{message.get()}">Update</button>
    </body>
</html>
```

Avatar Model-Store

- Javacript API:
  - ORM
  - Support SQL and Non-SQL

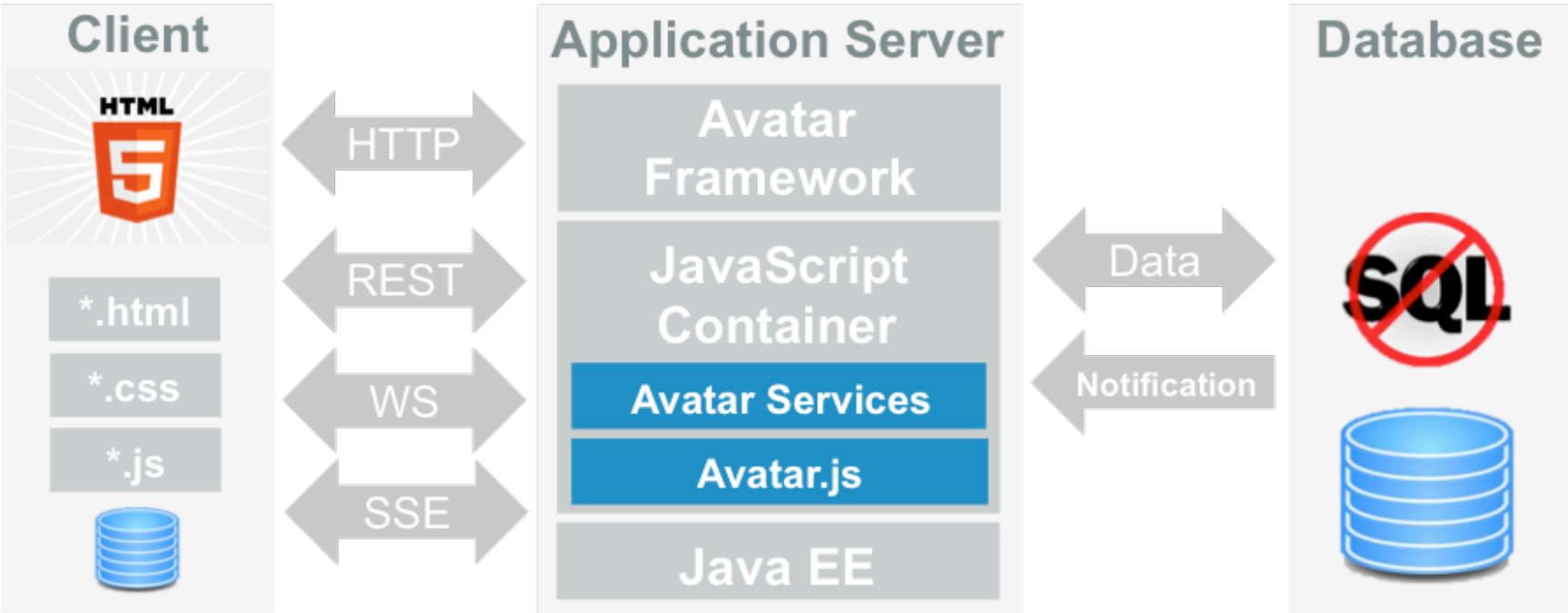
```
var Todo = avatarModel.newModel('todo', {
        "what": {
            type: "string",
            primary: false
        }
});
```

```
var store = avatar.newStore('mysql', {
    host: 'localhost',
    port: 3306,
    database: 'test',
    username: 'root',
    createDb: true,
    dropTables: true
});
```

Concurrency

- Actor-like concurrency model
- Avatar.js applications are both asynchronous and single-threaded (exactly as in Node.js)
- Built on Nashorn, JavaScript code in services can directly invoke Java code. This means that the 'no shared mutable state' model is not quite accurate

Avatar EE


Client
HTML
*.html
*.css
*.js
HTTP
REST
WS
SSE
Application Server
Avatar Framework
JavaScript Container
Avatar Services
Avatar.js
Java EE
Data
Notification
Database
SQL

Avatar EE

- Data providers
  - Files provider
  - JPA provider
- JavaScript API to interface with the Java Messaging Service (JMS)
- For future: “native” APIs to interface with other EE technologies:
  - Context and Dependency Injection (CDI)
  - Enterprise Java Beans (EJB)
  - ..etc

Modules known to work with Avatar

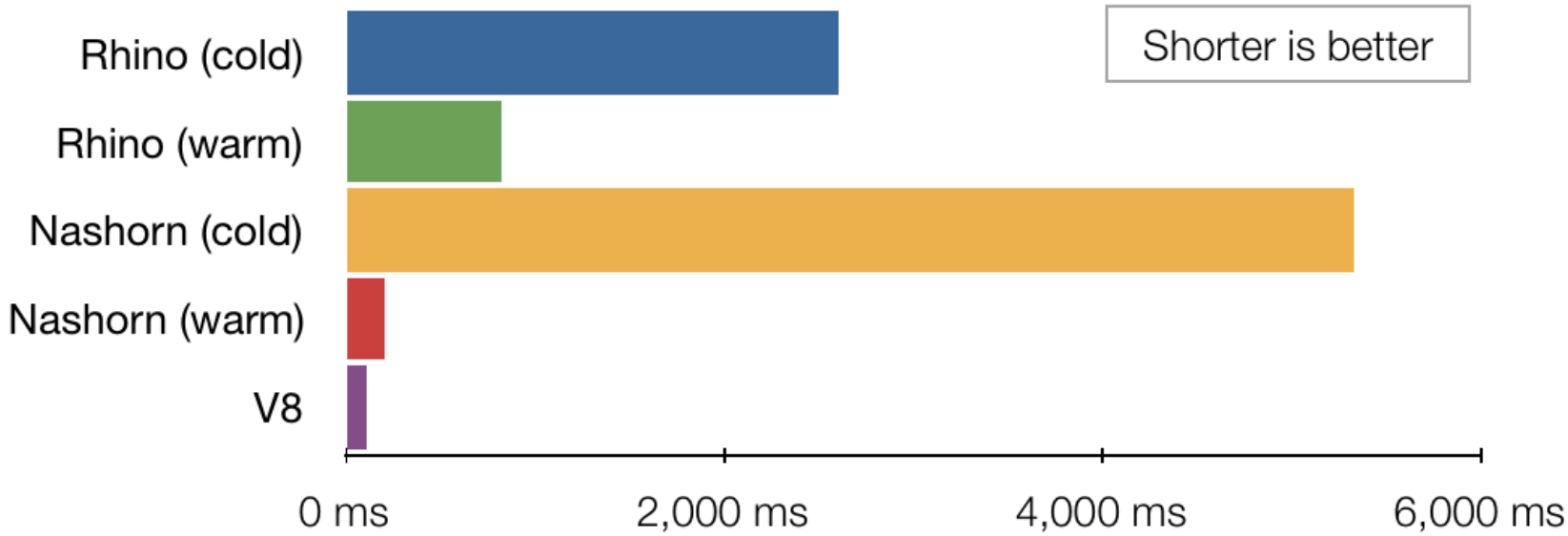
- abbrev
- ansi
- async
- block-stream
- chmodr
- chownr
- coffee-script
- colors
- commander
- connect
- debug
- engine.io
- express
- ftsream[1]
- glob[1]
- graceful-fs
- inherits
- ini
- init-package-json
- grunt
- grunt-bower-task
- jade
- lodash
- mime
- mkdirp
- mocha
- moment[2]
- mongodb[3]
- mongoose[3], [4]
- mustache
- node-unit
- node-uuid
- once
- opener
- optimist
- osenv
- passport
- q
- read
- redis
- request
- retry
- rimraf
- ronn
- semver
- slide
- socket.io
- tar
- uglify-js
- uid-number
- underscore
- which
- winston

[1]: with minor issues related to symbolic link handling
[2]: some issues with Ancient Greek lang
[3]: Replace all const occurrences with var
[4]: Buffer Schema data type is not supported

DAYS
CLOUD Days
mobile day
3 CONFERENCES
2 DAYS
1 PLACE
17-19 NOVEMBER 2014
INTER EXPO CENTER (IEC), SOFIA

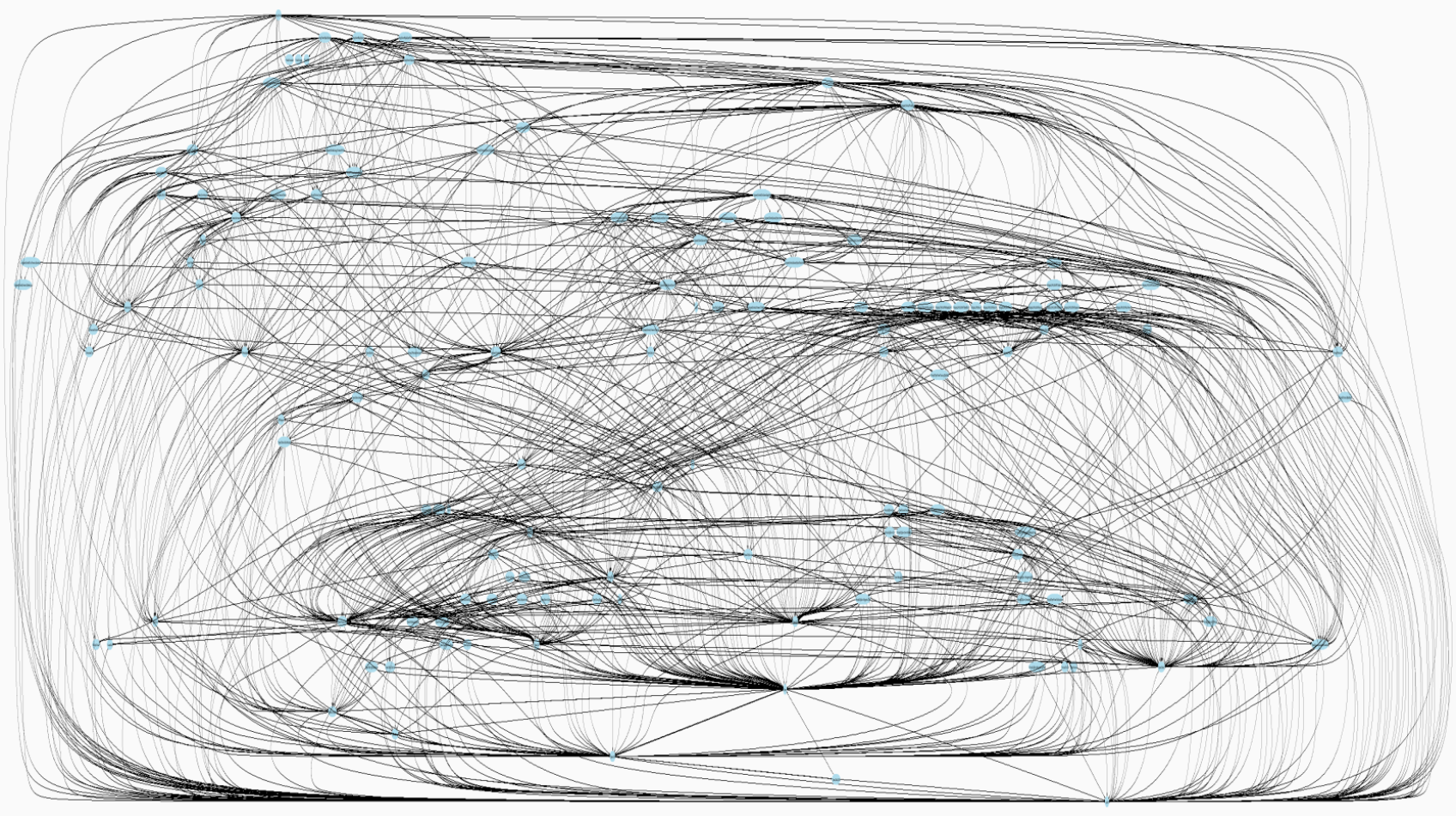
Demo

Benchmark


Shorter is better
Rhino (cold)
Rhino (warm)
Nashorn (cold)
Nashorn (warm)
V8
0 ms
2,000 ms
4,000 ms
6,000 ms

Still V8 is … faster..

But Java has the biggest ecosystem in the world!!



TBD

- Optimize Optimize Optimize Optimize …
- Native JS modules for Avatar
- Not-ECMA JS features not implemented
  - Generators, destructing assignment, etc

Homework:

- Try to write a multithreaded javascript application!

Yes, that sounds like nonsense, but...

DAYS
CLOUD Days
mobile day
3 CONFERENCES
2 DAYS
1 PLACE
17-19 NOVEMBER 2014
INTER EXPO CENTER (IEC), SOFIA

BG JUG

I NEED
YOU
for
JDK
DEVELOPMENT

THANK YOU FOR

your

ATTENTION! ANY QUESTIONS?

Additional info

- http://winterbe.com/posts/2014/04/05/java8-nashorn-tutorial/
- http://parleys.com/play/51681471e4b05b6ad71fa70f/chapter0/about
- http://www.oracle.com/technetwork/articles/java/jf14-nashorn-2126515.html