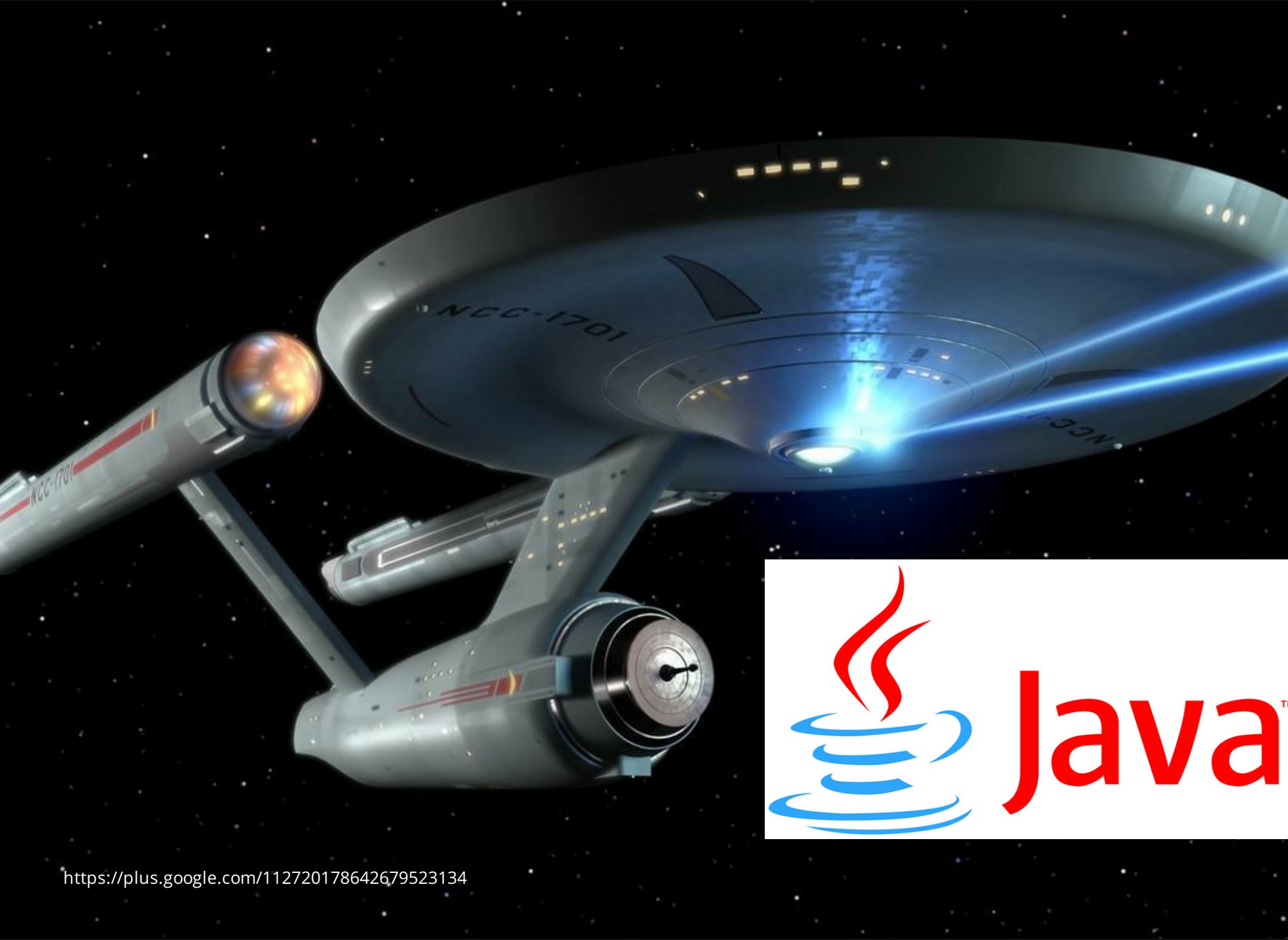
PROJECT AVATAR

MORE THAN JUST NODE.JS ON THE JAVA VIRTUAL MACHINE

Niko Köbler (@dasniko)
Heiko Spindler (@brainbrix)

Qualitects Group

http://de.james-camerons-avatar.wikia.com/wiki/Datei:Avatar-neytiri-wallpapers_16285_1920x1200.jpg



https://plus.google.com/112720178642679523134

WRITTEN IN

http://commons.wikimedia.org/wiki/File:JavaScript-logo.png

SPEAKING JAVASCRIPT

Like it or not, JavaScript is everywhere these days - from browser to server to mobile - and now you, too, need to learn the language or dive deeper than you have.

Dr. Axel Rauschmayer

http://speakingjs.com

GARTNER

Gartner predicts that through 2014, improved JavaScript performance will begin to push HTML5 and the browser as a mainstream enterprise application development environment.

(October 8, 2013)

http://www.gartner.com/newsroom/id/2603623

THOUGHTWORKS

I think JavaScript has been seen as a serious language for the last two or three years; I think now increasingly we're seeing JavaScript as a platform.

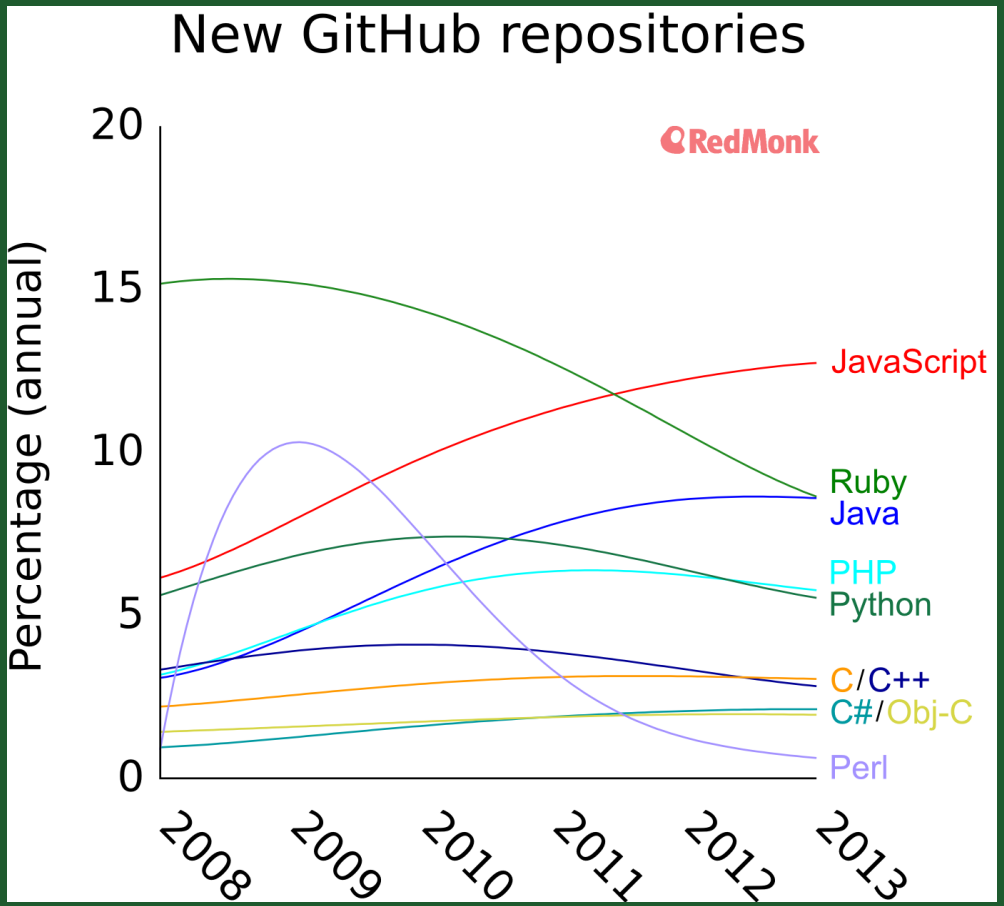
(Sam Newman, ThoughtWorks' Global Innovation Lead)

JavaScript has emerged both as a platform for server-side code but also a platform to host other languages.

(January, 2014)

http://www.techworld.com.au/article/536950/rise_rise_javascript

GITHUB



http://redmonk.com/dberkholz/2014/05/02/github-language-trends-and-the-fragmenting-landscape/

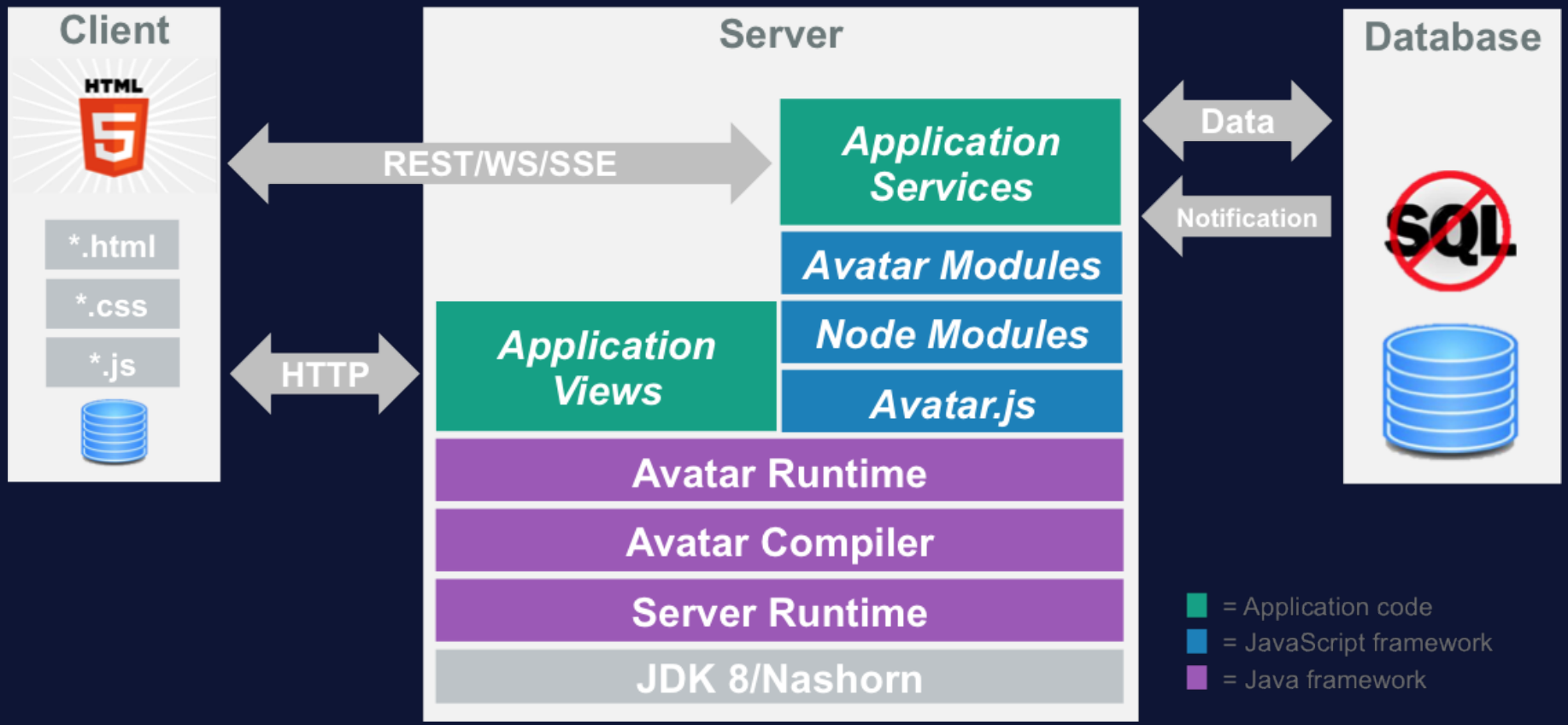
PROJECT AVATAR

- JavaScript service layer for Java EE
- REST, WebSockets & Server-Sent Events
- based on Nashorn, Avatar.js, Jersey, Grizzly, etc.

- Rich HTML5 client side framework
- Assumes very minor JavaScript knowledge
- Thin Server Architecture (TSA)

- https://avatar.java.net

AVATAR ARCHITECTURE



https://avatar.java.net

EXCURSION: NASHORN

Java-Nashorn – Wikipedia

de.wikipedia.org/wiki/Java-Nashorn

Benutzerkonto erstellen Anmelden

WIKIPEDIA
Die freie Enzyklopädie

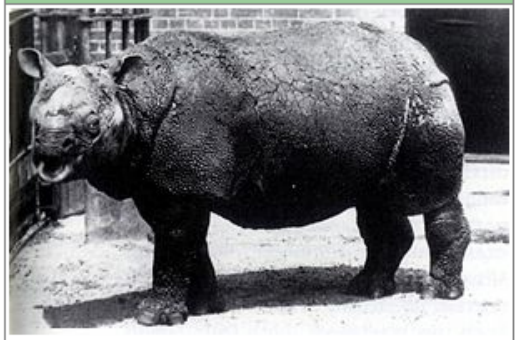
Artikel Diskussion Lesen Bearbeiten Versionsgeschichte Suchen

Java-Nashorn

Das **Java-Nashorn** (*Rhinoceros sondaicus*) ist eine in Asien beheimatete Nashornart mit nur einem Horn. Es ist nahe mit dem Panzernashorn (*Rhinoceros unicornis*) verwandt und der seltenste Vertreter der Nashörner und somit eines der seltensten Großsäugetiere. Die Art ist heute nur noch im Westen der Insel Java, im Ujung-Kulon-Nationalpark, mit etwa 40 Individuen anzutreffen. Ursprünglich war sie jedoch in weiten Teilen Südostasiens verbreitet und lebte teils sympatrisch mit den anderen asiatischen Nashornarten. Als Bewohner des tropischen Regenwaldes bevorzugt das Java-Nashorn weiche Pflanzennahrung, weiterhin lebt es überwiegend einzelgängerisch. Ein engagiertes Schutzprogramm soll helfen, das Java-Nashorn vor dem Aussterben zu bewahren.

Inhaltsverzeichnis [Verbergen]

1 Merkmale
2 Verbreitung
3 Lebensweise
  3.1 Territorialverhalten
  3.2 Ernährungsweise
  3.3 Fortpflanzung
  3.4 Interaktion mit anderen Tierarten
  3.5 Parasiten
4 Systematik
5 Stammesgeschichte
6 Forschungsgeschichte

Java-Nashorn	
	
Java-Nashorn im Londoner Zoo (Haltung 1874 bis 1885)	
Systematik	
Unterklasse:	Höhere Säugetiere (Eutheria)
Überordnung:	Laurasiatheria
Ordnung:	Unpaarhufer (Perissodactyla)
Familie:	Nashörner (Rhinocerotidae)
Gattung:	*Rhinoceros*
Art:	Java-Nashorn
Wissenschaftlicher Name	

Hauptseite
Themenportale
Von A bis Z
Zufälliger Artikel

Mitmachen
Artikel verbessern
Neuen Artikel anlegen
Autorenportal
Hilfe
Letzte Änderungen
Kontakt
Spenden

Drucken/exportieren

Werkzeuge

In anderen Sprachen
Aragonés
العربية
Azərbaycanca
Български
Brezhoneg

EXCURSION: NASHORN

- JavaScript Enginge on the JVM (native)
- competes with Google V8
- ECMAScript 5.1 compatible (ECMAScript 6 in future)
- Seamless integration of Java and JavaScript
- Language and API Extensions

  closures, collections & for each, multi-line string literals, string interpolation, __noSuchProperty__, __noSuchMethod__, typed arrays, binding properties, error extensions, conditional catch clause, String functions, and many, many more...

- https://blogs.oracle.com/nashorn/

NASHORN

COMMAND LINE CLIENT

```
$ $JAVA_HOME/bin/jjs
jjs> print('Hello Nashorn!');
```

INVOKING JAVASCRIPT FROM JAVA

```
ScriptEngine engine = new ScriptEngineManager().getEngineByName("nashorn");
engine.eval("print('Hello Nashorn!');");
```

```
engine.eval(new FileReader("scriptfile.js"));
```

```
Invocable invocable = (Invocable) engine;
Object result = invocable.invokeFunction("jsSayHello", "Nashorn");
```

NASHORN

INVOKING JAVA FROM JAVASCRIPT

```
static String sayHello(String name) {
    return String.format("Hello %s from Java!", name);
}
```

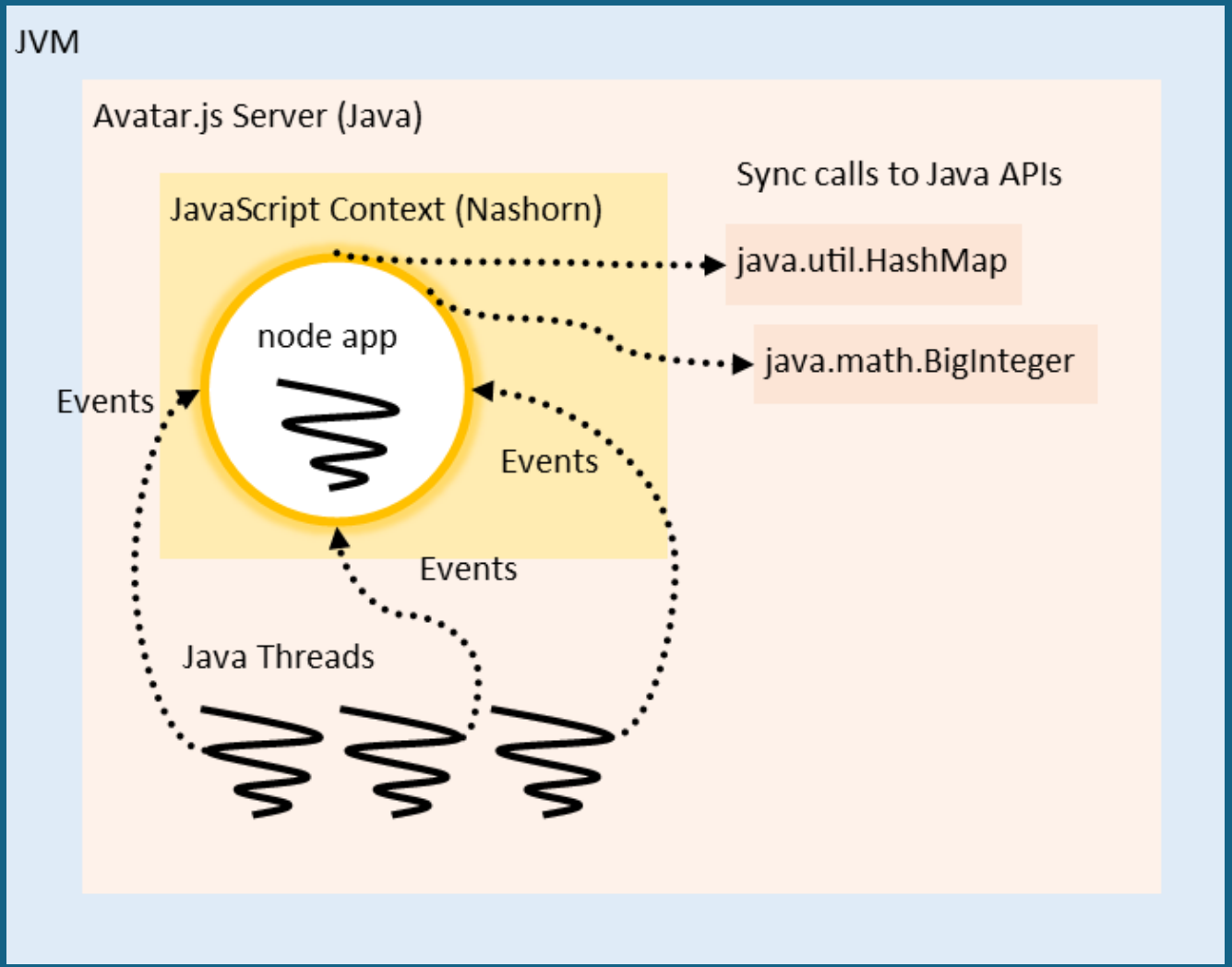
```
var MyJavaClass = Java.type('my.package.MyJavaClass');
var result = MyJavaClass.sayHello('Nashorn');
print(result); // Hello Nashorn from Java!
```

EXCURSION: AVATAR-JS

Node.js on the JVM

~95% Node.js API compatibility
no Chrome V8 native APIs
many of the node-modules work

(e.g.: abbrev, ansi, async, block-stream, chmodr, chownr, coffee-script, colors, commander, connect, debug, engine.io, express, ftsream, glob, graceful-fs, inherits, ini, init-package-json, grunt, grunt-bower-task, jade, lodash, mime, mkdirp, mocha, moment, mongodb, mongoose, 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)



https://avatar-js.java.net

UI

MODEL

```
<script data-model="local" data-instance="add">
 var AddMathModel = function() {
  this.left = 0;
  this.right = 0;
  this.reset = function() {
   this.left = this.right = 0;
  };
 };
</script>
```

2-WAY-BINDING

```
<input id="lfa" type="text" data-value="#{add.left}"/>
<input id="rta" type="text" data-value="#{add.right}"/>
<span id="output1">
 The result is #{add.left} + #{add.right} = #{add.left + add.right}
</span>
<button onclick="#{add.reset()}" id="reset">Reset</button>
```

WIDGETS & THEMES

default based on jQuery UI
other/custom themes & jQuery plugins possible
optional: Dojo dijit widgetLib extension

WHAT ABOUT ANGULAR.JS?

YES! FOR SURE!
WHY NOT?

Thanks to TSA! :-)

BACKEND

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

DATA PROVIDER

```
var itemsFileProvider = new avatar.FileDataProvider({
  filename: 'rest-sample.txt'
  key: 'key'
});

var itemsJpaProvider = new avatar.JPADataProvider({
  persistenceUnit: 'rest',
  createTables: true,
  entityType: 'Item'
});
```

```
myDataProvider.create(item, callback);
myDataProvider.del(item, callback);
myDataProvider.get(key, callback);
myDataProvider.getCollection(parameters, count, offset, callback);
myDataProvider.put(key, item, callback);
```

```
myDataProvider.create(item).then(...);
```

JMS

```
var myJMS = new avatar.JMS({
   connectionFactoryName: 'jms/myConnFactory',
   destinationName: 'jms/myQueue'
});
```

```
myJMS.addListener(function(message) {
   avatar.log('Got message: ' + message);
}).then(function() {
   avatar.log('Sending message...');
   return myJMS.send('Test message');
}).then(function() {
   avatar.log('Message sent.');
});
```

REST SERVICE

```
avatar.registerRestService({url: 'data/items/{item}', methods: ['GET']},
   function() {
      this.onGet = function(request, response) {
         myDataProvider.get(this.item)
           .then(response.send(itemValue));
      };
   }
);
```

```
avatar.registerRestService({
      url: 'data/items/{itemid}',
      dataProvider: myDataProvider,
      authorization: {...} },
   function() {}
);
```

PUSH SERVICE

```
avatar.registerPushService({url: 'push/stocks'},
   function() {
      this.onOpen = function(context){
         context.setTimeout(5000);
      };

      this.onTimeout = function(context) {
         context.sendMessage('HelloWorld!');
      };
   }
);
```

```
avatar.registerPushService({url: '/push/chat',
 jms: {
   connectionFactoryName: 'jms/MyConnectionFactory',
   destinationName: 'jms/ChatQueue'}
 }, function(){});
```

SOCKET SERVICE

```
avatar.registerSocketService({url: '/websocket/chat'},
   function() {
      this.onMessage = function(peer, message){
         peer.getContext().sendAll(message);
      };
   }
);
```

```
avatar.registerSocketService({
    url: '/websocket/jmschat/{chatroom}',
    jms: {
       connectionFactoryName: 'jms/MyConnectionFactory',
       destinationName: 'jms/ChatTopic',
       messageSelector: "chatroom='#{this.chatroom}'",
       messageProperties: {chatroom: '#{this.chatroom}'}}},
 function() {});
```

MESSAGE BUS

```
var bus = avatar.application.bus;
```

```
bus.publish('echo', { x : 'x', y : 'y' });
```

```
bus.on('echo', function(body, msg) {
   avatar.log('Got message: ' + JSON.stringify(body));
});
```

```
var app = avatar.application;
var bus = app.bus;
var echoOnce = function(body, msg) {
   body.name = app.name;
   body.threadIndex = app.threadIndex;
   body.threadCount = app.threadCount;
   body.startTime = app.startTime;
   body.uptime = app.uptime;
   body.topic = msg.topic;
   body.src = msg.sourceAddress;
   bus.reply(msg, body);
   bus.unsubscribe(msg.topic, echoOnce);
}
bus.on('echo', echoOnce);
```

DEMO TIME

https://github.com/dasniko/avatar-twitterwall

TIMELINE

Period	Milestone
JavaOne 2013	Project Avatar Launch GlassFish Runtime
June 2014	WebLogic Runtime (WLS 12.1.3)
Mid 2015	WebLogic 12.1.4 Avatar Commercial Support
???	Part of Java EE Spec?

(WITHOUT OBLIGATION)

PROJECT AVATAR CONCLUSION

- Lightweight integration in JavaEE and Enterprise context
  - JMS, JPA, REST, Java-APIs, ...
  - WebLogic Runtime
- Perfect interaction between client and server
- Use of client-component is not a MUST
  - But neither of the server-component
- Commercial support from Oracle (in future)

THANK YOU!

QUESTIONS?