
ORACLE®

JavaOne™

ORACLE®

JavaScript Across Tiers with Nashorn and Avatar.js

Kuassi Mensah
Director Product Management
Oracle Server Technologies, Java Group
September, 2014

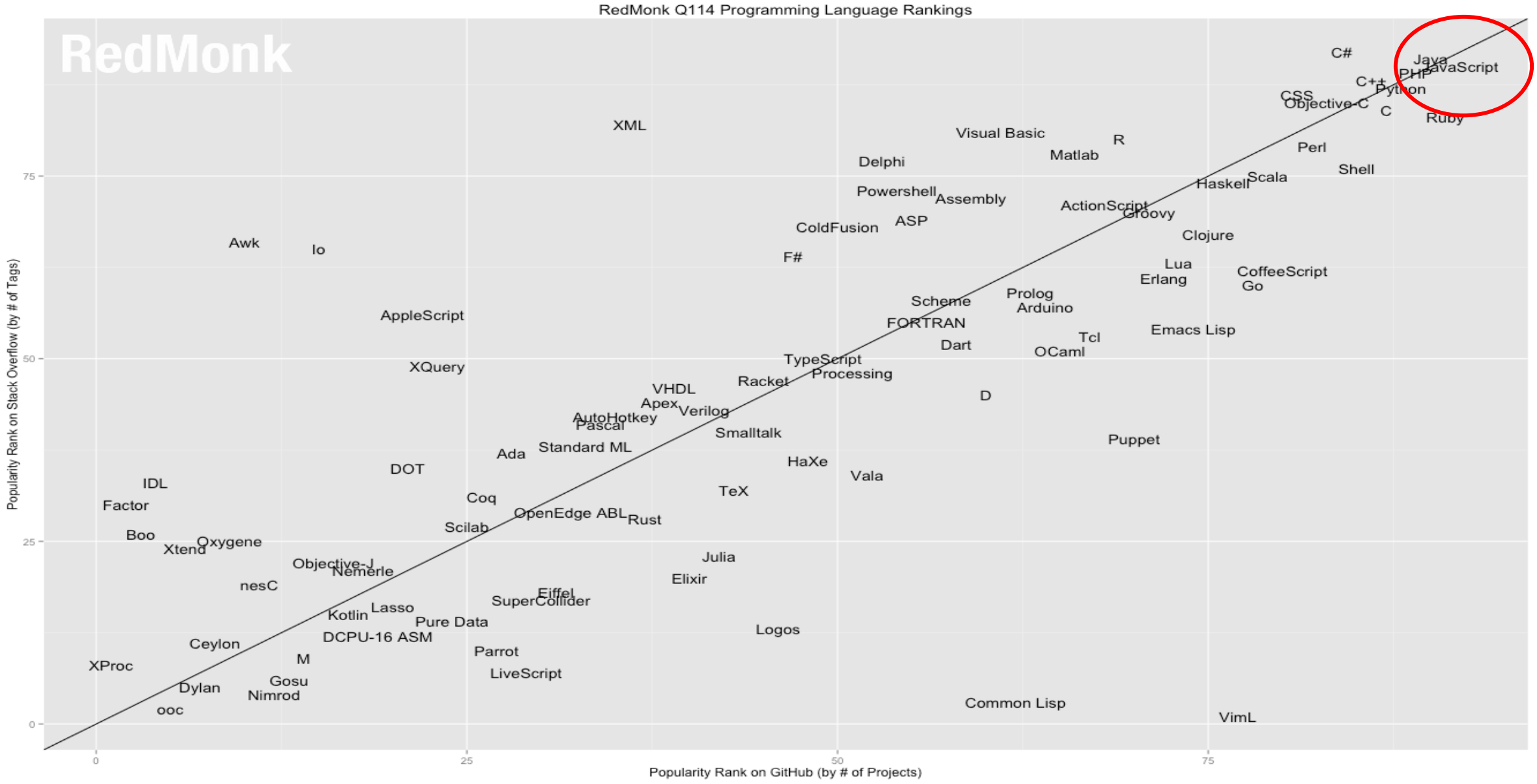
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

1. JavaScript and Web Applications Architecture
2. Nashorn: JavaScript engine on the JVM
3. JavaScript Stored Procedures with Nashorn
4. Java and Node.js: project Avatar & Avatar.js
5. Database Access with Avatar.js using JDBC and UCP

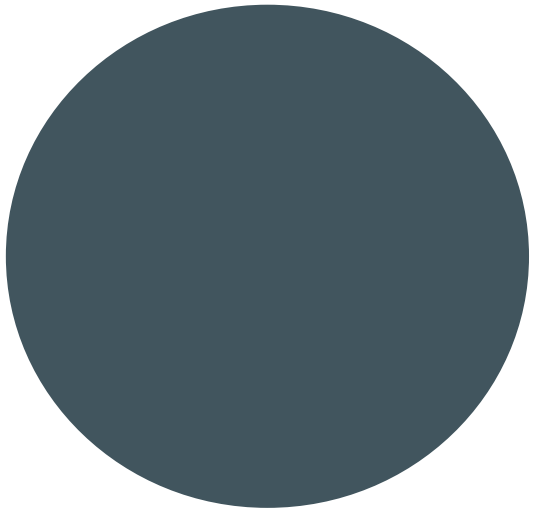
Top Two Popular Languages: JavaScript & Java



Developer’s Nirvana: Same Language Across Tiers

Java across tiers is already a reality!!
Can JavaScript accomplish the same thing?

Browser

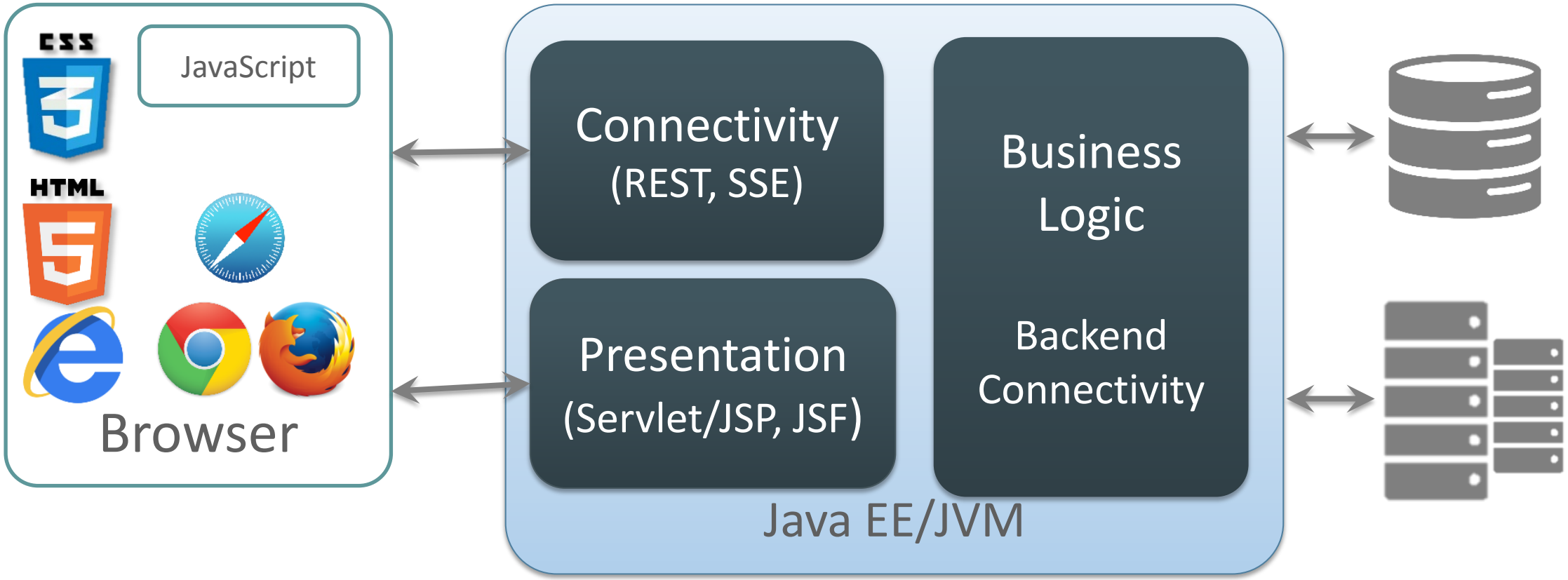


Middle-tier

Database

The Evolution of Web Applications Architecture

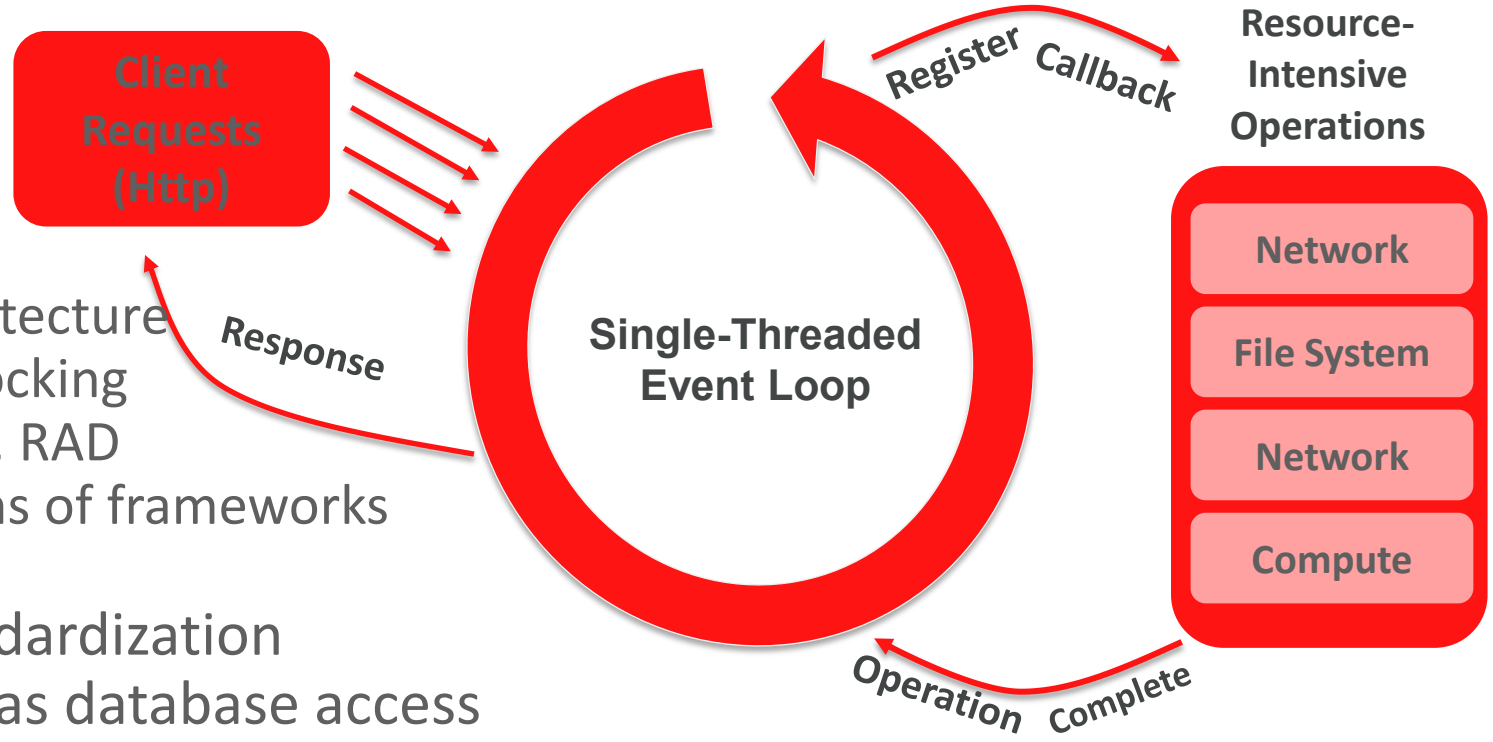
JavaScript in Browser – Java in Middle-tier (Java EE)



Node Programming Model

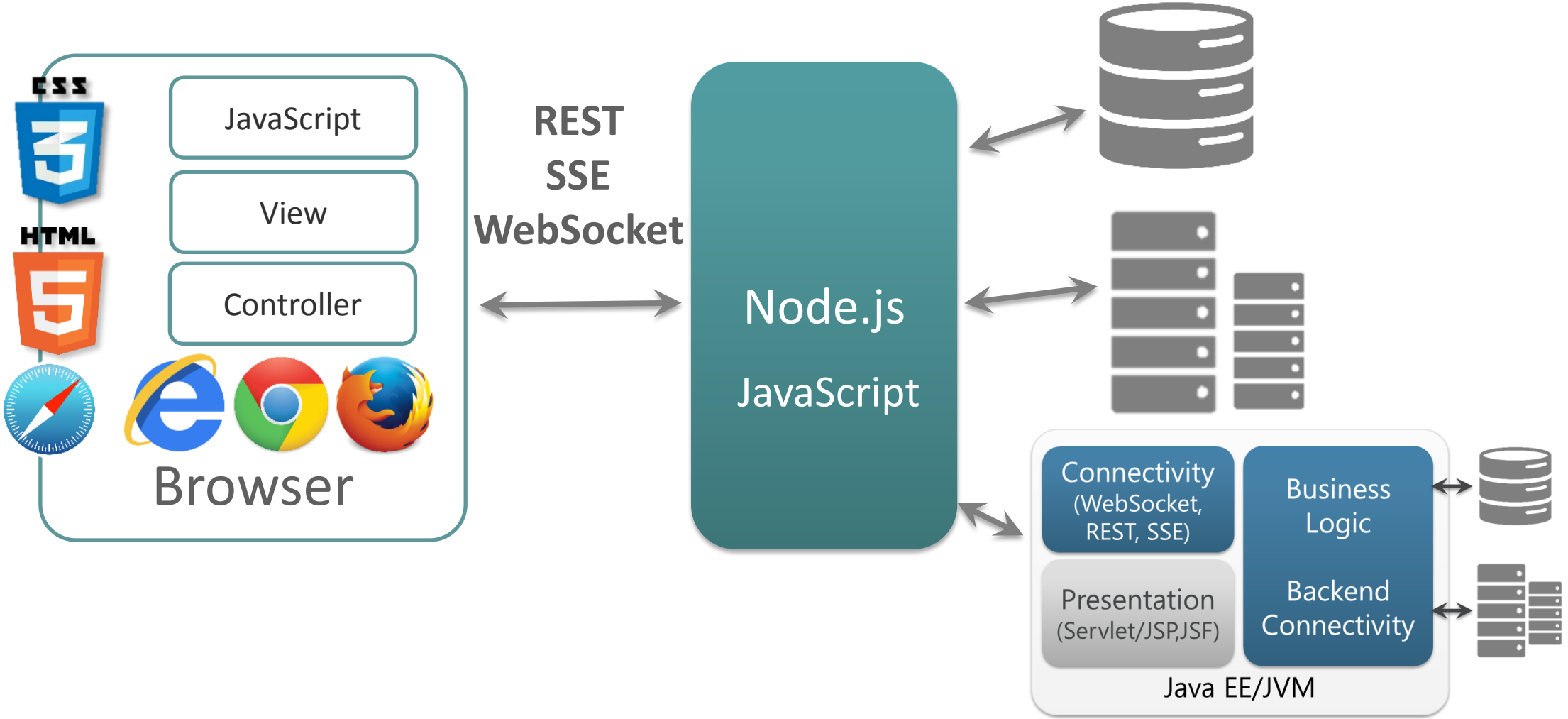
- Single-threaded architecture
- Event-driven, non-blocking
- Ease of development, RAD
- Large community, tons of frameworks

- But Node lacks standardization in many areas such as database access



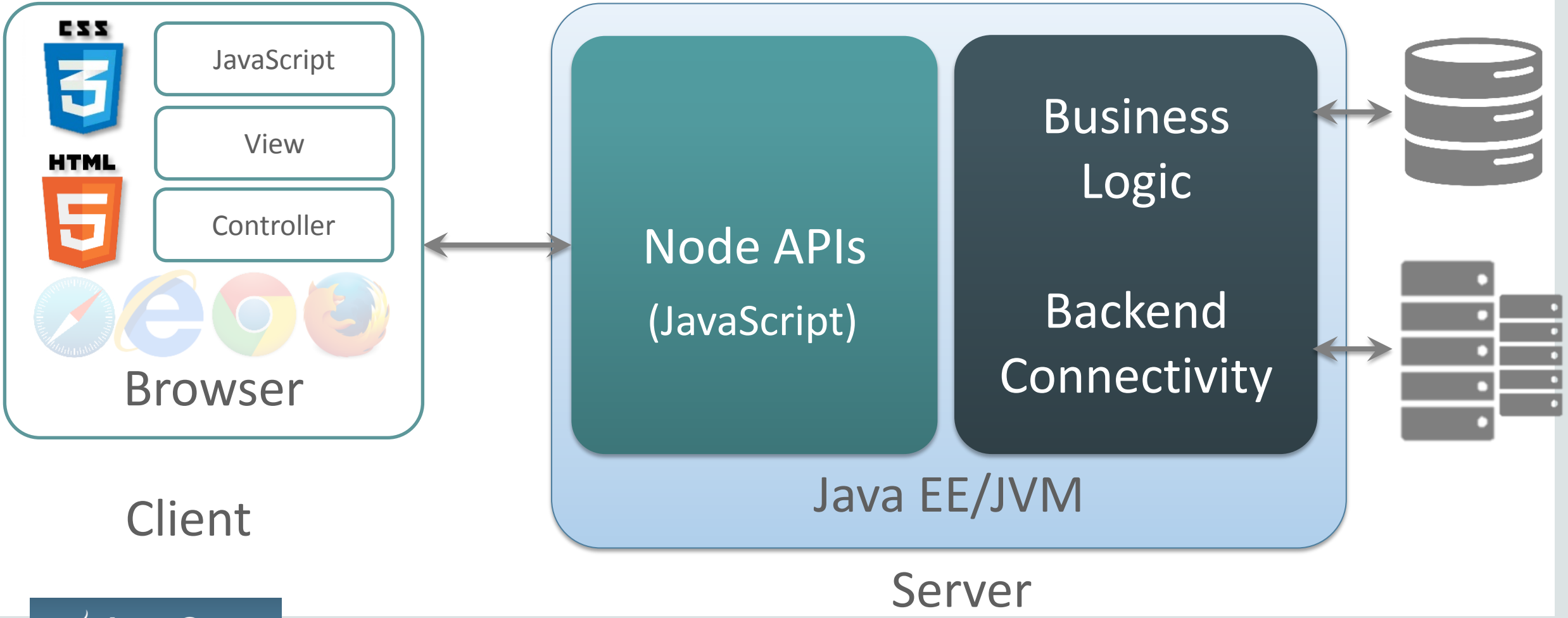
The Evolution of Web Applications Architecture

Mobile-enabling Java Services with Node.js



The Evolution of Web Applications Architecture

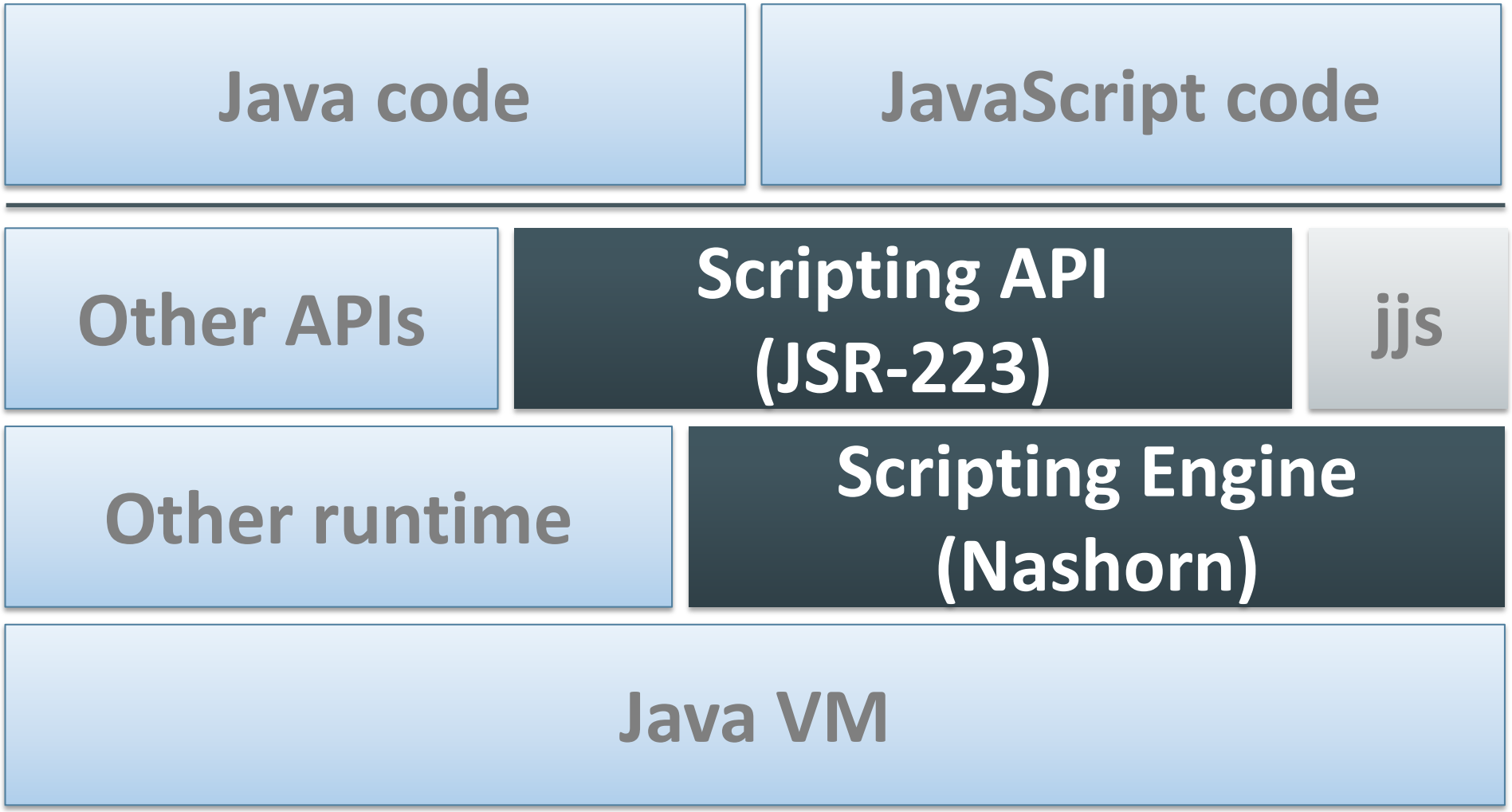
Co-locate Node and Java?



Program Agenda

1. JavaScript and Web Application Architecture
2. JavaScript engine on the JVM: project Nashorn
3. JavaScript Stored Procedures with Nashorn
4. Java and Node.js: project Avatar & Avatar.js
5. Database Access with Avatar.js using JDBC and UCP

Project Nashorn - JEP 174



The Nashorn JavaScript Engine in Java

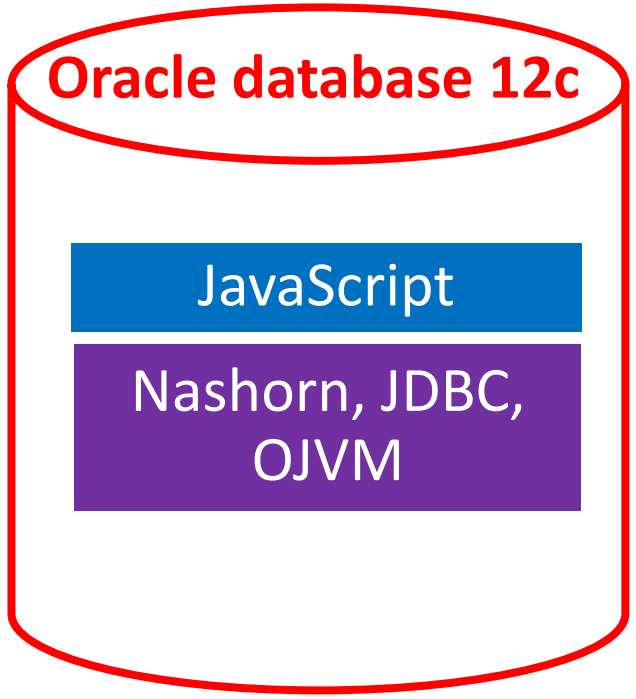
- Introduced in Java 7
- Replacing Rhino
  Security, performance, and so on
- Support for javax.script (JSR 223) API
- Seamless Interaction between Java and JavaScript
  Invoke Nashorn from Java
  Invoke Java from Nashorn

Program Agenda

1. Rationales for JavaScript in Middle and Database tiers?
2. Nashorn: JavaScript engine on the JVM
3. JavaScript Stored Procedures with Nashorn
4. Java and Node.js: project Avatar & Avatar.js
5. Database Access with Avatar.js using JDBC and UCP

JavaScript Stored Procedures in Oracle Database 12c

- Really? Why? How?



Enabling Nashorn in the Database

1. Requires an Oracle database 12c with Java 7 (embedded JVM)

2. a) Build Nashorn.jar
   b) Modify Nashorn Shell code to interpret script (given as parameter)
   c) invoke getResourceAsStream() on the current thread's context class loader
   d) Rebuild Nashorn.jar with the modified Shell

3. Load the modified Nashorn.jar into a database schema
   ```
   loadjava -v -r -u hr/<password> nashorn.jar
   ```

Enabling Nashorn in the Database (Cont'd)

4. Create a new dbms_javascript package

```
create or replace package body dbms_javascript as
  procedure run(script varchar2) as
  language java name
'com.oracle.nashorn.tools.Shell.main(java.lang.String[])';
end;
/
```

5. Create a new role and add some permissions (incomplete code)

```
SQL> create role nashorn;
SQL> call dbms_java.grant_permission('NASHORN', ...
'createClassLoader',  '' );
```

6. Grant NASHORN role to HR

```
SQL> grant NASHORN to HR;
```

JavaScript with JDBC code: Database.js

```
var Driver = Packages.oracle.jdbc.OracleDriver;
var oracleDriver = new Driver();
var url = "jdbc:default:connection:";   // server-side JDBC driver
var query ="SELECT first_name, last_name from employees";
// Establish a JDBC connection
var connection = oracleDriver.defaultConnection();
// Prepare statement
var preparedStatement = connection.prepareStatement(query);
// execute Query
var resultSet = preparedStatement.executeQuery();
// display results
    while(resultSet.next()) {
    print(resultSet.getString(1) + "== " + resultSet.getString(2) + " " );
    }
// cleanup
resultSet.close();
preparedStatement.close();
connection.close();
```

Running JavaScript Stored Procedures

Load database.js in the database

```
loadjava -v -r -u hr/<password> database.js
```

Execute JavaScript directly in the database

```
sqlplus HR/HR
SQL> set serveroutput on
SQL>call dbms_java.set_output(80000)
SQL>call dbms_javascript.run('database.js');
```

➔the result set will be displayed on the console

Enhancements:

(i) Call JavaScript stored procedures as CallableStatement returning refCursor => process JSON documents

(ii) Java 8 => performance

Demo

Running JavaScript in the Database using Nashorn

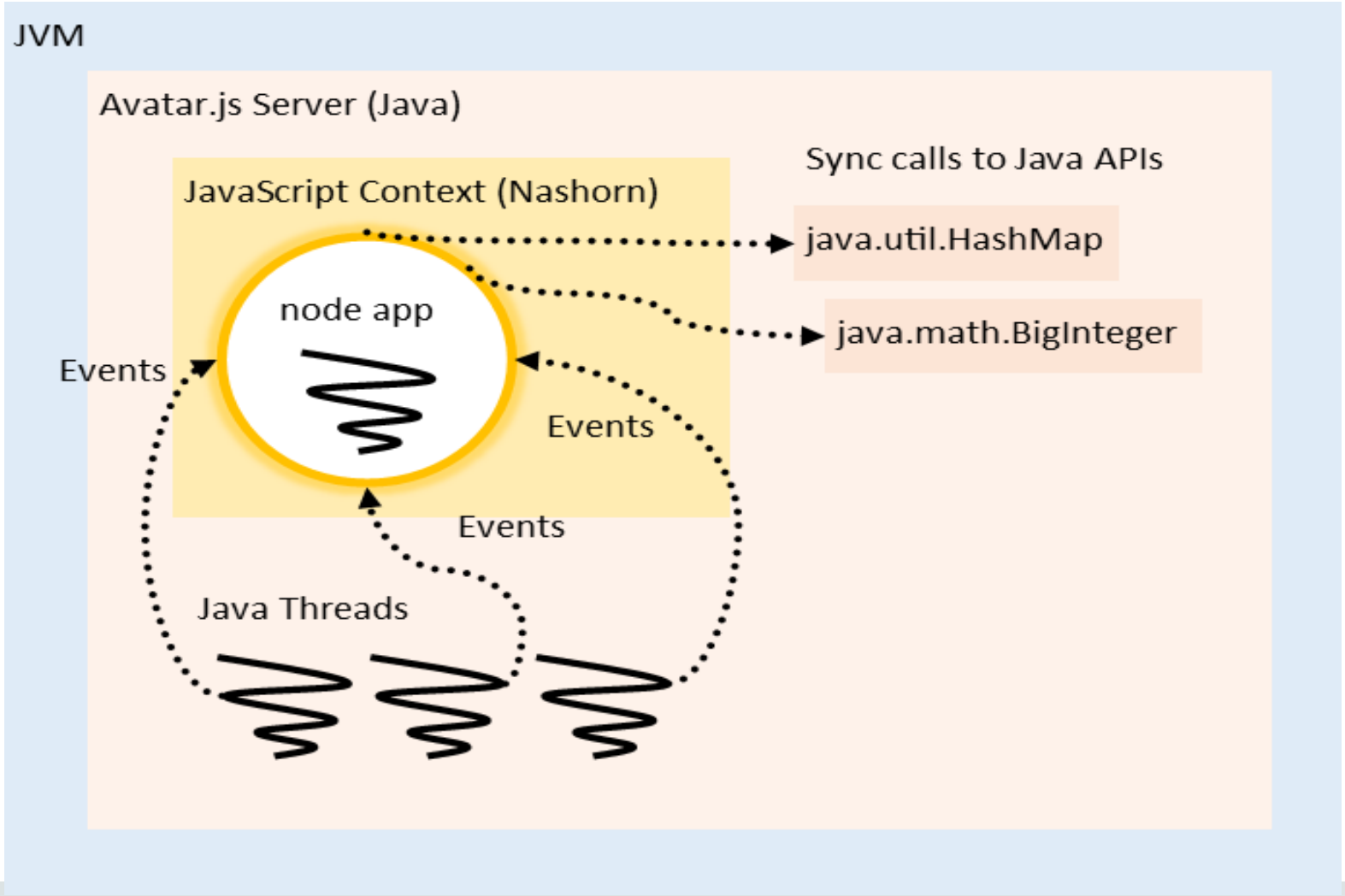
Program Agenda

1. Rationales for JavaScript in Middle and Database tiers
2. JavaScript engine on the JVM: Nashorn
3. JavaScript Stored Procedures with Nashorn
4. Java and Node.js: project Avatar & Avatar.js
5. Database Access with Avatar.js using JDBC and UCP

Co-Locating Node.js and Java: Avatar.js

https://avatar-js.java.net/

- JavaScript to leverage Java code, via Nashorn, including threads

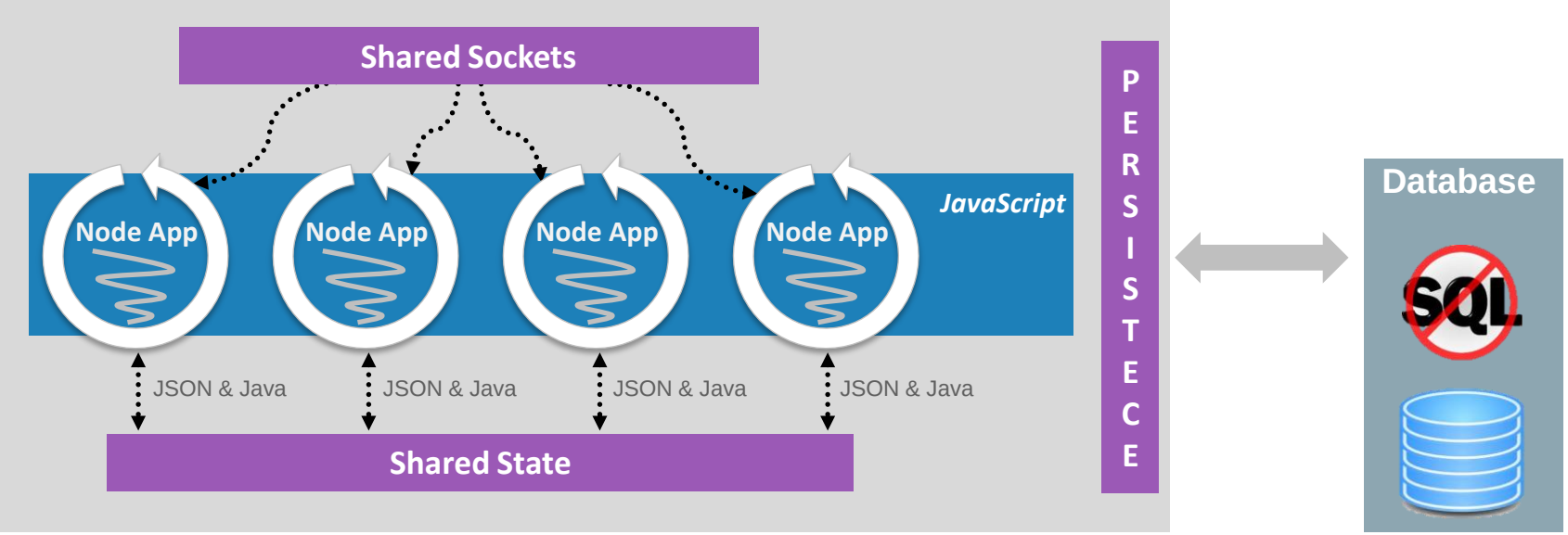


Project Avatar: JavaScript and Java EE
Enterprise Node.js on the JVM

NEW

Multiple event loop threads, the application is *cloned* for each

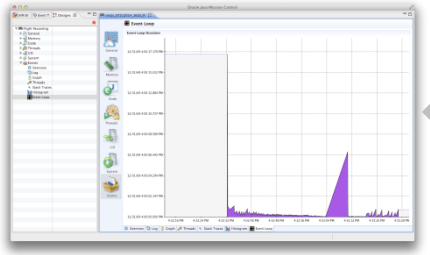
- Shared sockets enable server applications to open the same port on multiple threads
- Coordination via JavaScript state sharing APIs (messaging and map)
- Persistence via JavaScript model APIs (SQL and NoSQL)
- Oracle product integrations



Project Avatar: Architecture

NEW

REST/WS/SSE



Mission Control

Avatar Process

- *Application Modules & Jars*
- *Bundled Third-Party Modules*
- *Avatar API Modules*
- *Core Node Modules*
- Avatar
- Coherence
- MBeans | WLS T3 thin client
- Flight Recorder Probes | EclipseLink, NoSQL
- Nashorn & JRE 8
- libuv + JNI bindings | JVM

Optional Processes

- CAM Plugin → Admin Agent
- State → Coherence
- EJB/JMS → WebLogic
- Data → Database

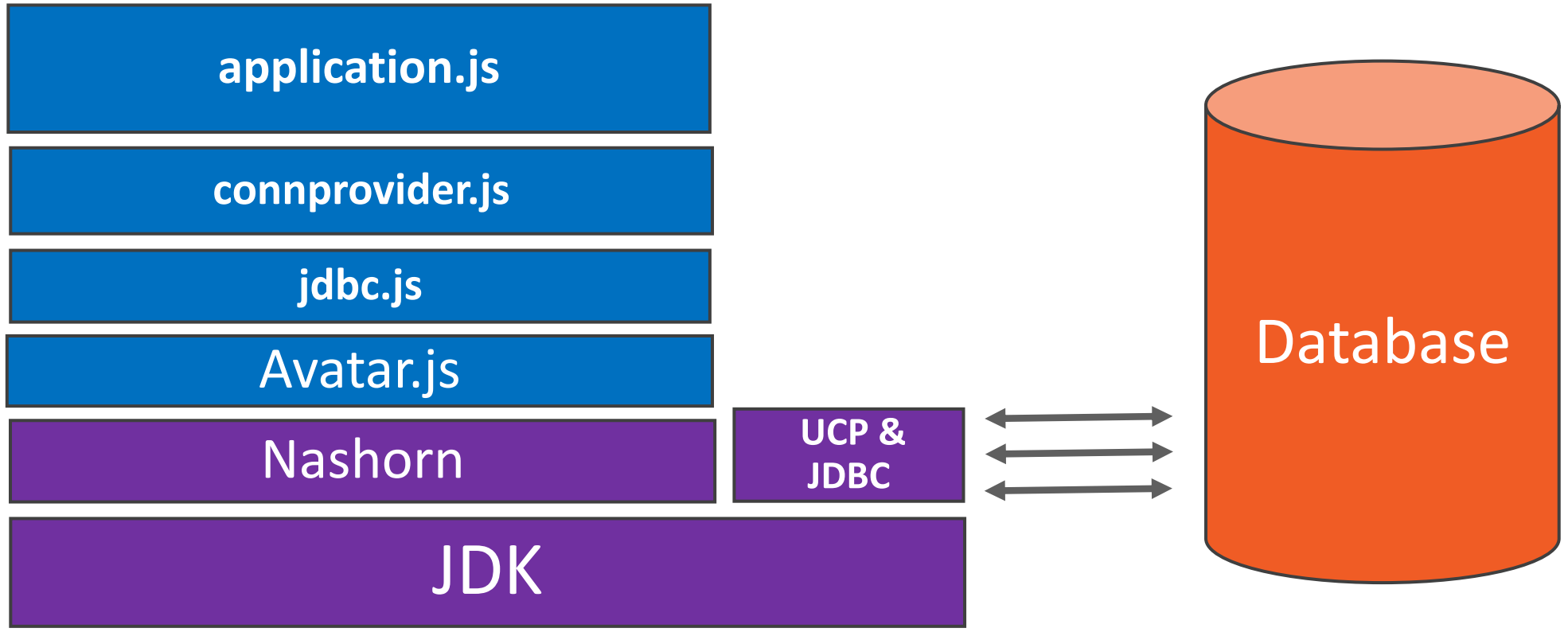
- Application JavaScript & Java
- JavaScript code
- Java code
- Platform specific native code

JavaOne ORACLE

Program Agenda

1. Rationales for JavaScript in Middle and Database tiers
2. JavaScript engine on the JVM: Nashorn
3. JavaScript Stored Procedures with Nashorn
4. Java and Node.js: project Avatar & Avatar.js
5. Database Access with Avatar.js using JDBC and UCP

JavaScript Database Access with Avatar.js
Building blocks



Non-Blocking JDBC Calls
jdbc.js

Turning synchronous JDBC apis into non-blocking function calls

```
var makeExecutecallback = function(userCallback) {
  return function(name, args){
        ...
        userCallback(undefined, args[1]);
  }
}

function submit(task, callback, msg) {
    var handle = evtloop.acquire();
    try {    var ret = task();
             evtloop.post(new EventType(msg, callback, null, ret)); {catch{}

  evtloop.submit(r);
}
```

Non-Blocking JDBC Calls – Cont'd
jdbc.js

The result is received via a callback

```
exports.connect = function(userCallback) {..} // JDBC and UCP settings

Statement.prototype.executeQuery = function(query, userCallback) {
        var statement = this._statement;
         var task = function() {
         return statement.executeQuery(query);
       }
     submit(task, makeExecutecallback(userCallback), "jdbc.executeQuery");
  }
```

Similarly

```
Connection.prototype.getConnection = function() {…}
Connection.prototype.createStatement = function() {..}
Connection.prototype.prepareCall = function(storedprocedure) {..}
Statement.prototype.executeUpdate = function(query, userCallback) {..}
```

JavaScript Database Access with Atavar.js

application.js

Returning ResultSet in callback

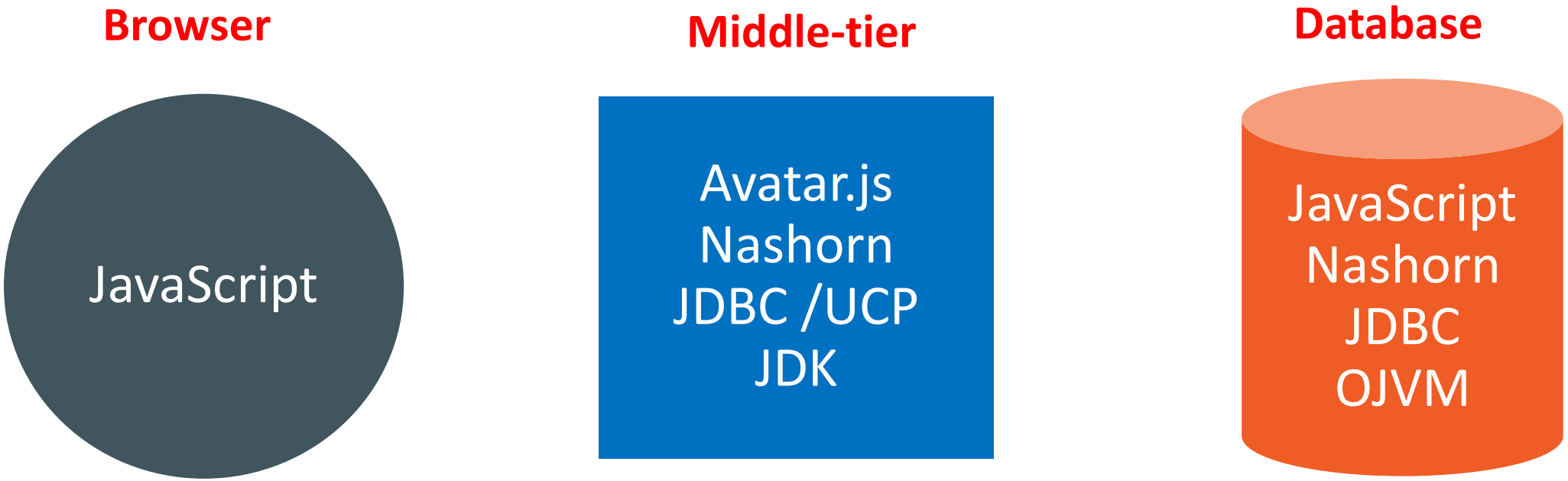
```
var ConnProvider = require('./connprovider').ConnProvider;
var connProvider = new ConnProvider(function(err, connection){.. });

var server = http.createServer(function(request, response) {
  connProvider.getConn(function(name,data){..});
  connProvider.prepStat(function(resultset) {
            while (resultset.next()) {
              response.write(resultset.getString(1) + " --" + resultset.getString(2));
              response.write('<br>');
    }
    response.write('</body></html>');
    response.end();
}
server.listen(4000, '127.0.0.1');
```

Demo

Database Access with Avatar.js using JDBC and UCP

JavaScript Across Tiers: QED



Nashorn

http://openjdk.java.net/projects/nashorn/

- OpenJDK wiki – Nashorn
  https://wiki.openjdk.java.net/display/Nashorn/Main
- Mailing List
  nashorn-dev@openjdk.java.net
- Blogs
  - Nashorn - JavaScript for the JVM
    http://blogs.oracle.com/nashorn/

Avatar.js

- Project Page
  https://avatar-js.java.net/
- Mailing List
  users@avatar-js.java.net

Avatar

- Project Page
  https://avatar.java.net/
- Mailing List
  users@avatar.java.net

Hardware and Software
Engineered to Work Together

JavaOne™

ORACLE®


ORACLE®