Dimitri Gielis

OMG! JavaScript Straight from the Oracle Database

 www.apexRnD.be
 dgielis.blogspot.com
 @dgielis
 dgielis@apexRnD.be

Dimitri Gielis

- Founder & CEO of APEX R&D
- 19+ years of Oracle Experience (OCP & APEX Certified)
- Oracle ACE Director
- “APEX Developer of the year 2009” by Oracle Magazine
- “Oracle Developer Choice award (ORDS)” in 2015
- Author Expert Oracle APEX
- Presenter at Conferences





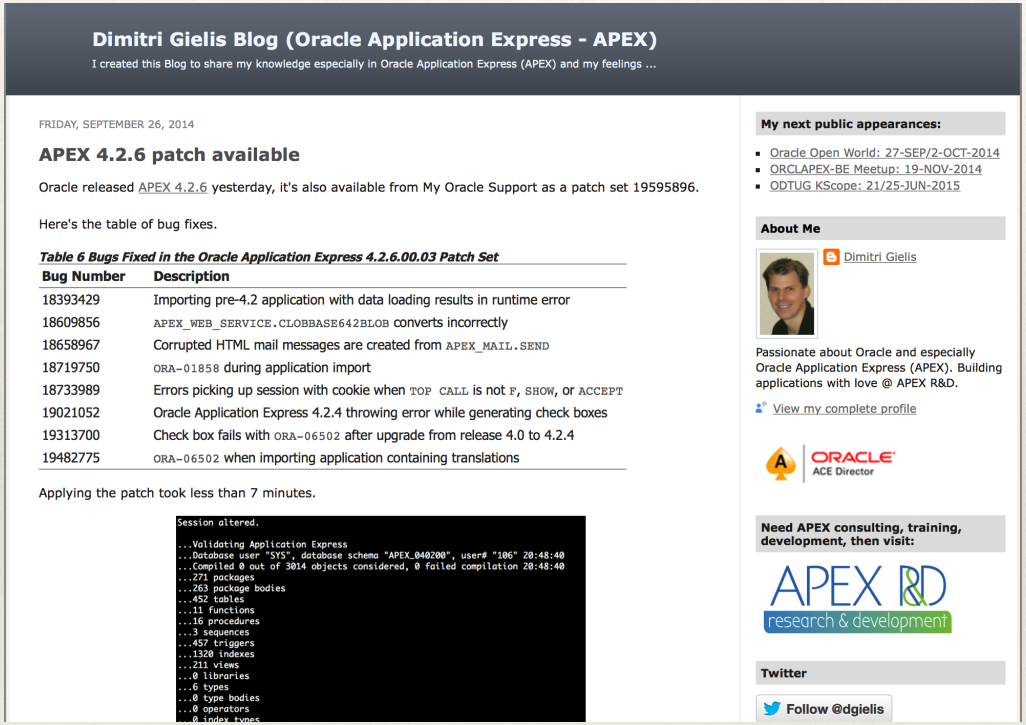


www.apexRnD.be

www.apexofficeprint.com

http://dgielis.blogspot.com

@dgielis



Dimitri Gielis Blog (Oracle Application Express - APEX)

I created this Blog to share my knowledge especially in Oracle Application Express (APEX) and my feelings ...

FRIDAY, SEPTEMBER 26, 2014

APEX 4.2.6 patch available

Oracle released APEX 4.2.6 yesterday, it's also available from My Oracle Support as a patch set 19595896.

Here's the table of bug fixes.

Table 6 Bugs Fixed in the Oracle Application Express 4.2.6.00.03 Patch Set

Bug Number	Description
18393429	Importing pre-4.2 application with data loading results in runtime error
18609856	APEX_WEB_SERVICE.CLOBBASE642BLOB converts incorrectly
18658967	Corrupted HTML mail messages are created from APEX_MAIL.SEND
18719750	ORA-01858 during application import
18733989	Errors picking up session with cookie when TOP CALL is not F, SHOW, or ACCEPT
19021052	Oracle Application Express 4.2.4 throwing error while generating check boxes
19313700	Check box fails with ORA-06502 after upgrade from release 4.0 to 4.2.4
19482775	ORA-06502 when importing application containing translations

Applying the patch took less than 7 minutes.

```
Session altered.

...Validating Application Express
...Database user "SYS", database schema "APEX_040200", user# "106" 20:48:40
...Compiled 0 out of 3014 objects considered, 0 failed compilation 20:48:40
...271 packages
...263 package bodies
...452 tables
...11 functions
...16 procedures
...3 sequences
...457 triggers
...1320 indexes
...211 views
...0 libraries
...6 types
...0 type bodies
...0 operators
...0 index types
```

My next public appearances:

- Oracle Open World: 27-SEP/2-OCT-2014
- ORCLAPEX-BE Meetup: 19-NOV-2014
- ODTUG KScope: 21/25-JUN-2015

About Me

Dimitri Gielis

Passionate about Oracle and especially Oracle Application Express (APEX). Building applications with love @ APEX R&D.

View my complete profile

ORACLE ACE Director

Need APEX consulting, training, development, then visit:

APEX R&D research & development

Twitter

Follow @dgielis




Why JavaScript in the DB?


HELL ON WHEELS


EH...SURE. WHY THE HELL NOT
memegenerator.net


Occupational Therapy
Demo

THURSDAY, AUGUST 04, 2016

SQLcl to the rescue when the Database and APEX fail (with JSON limitations)

In the last two years I've been using JSON in almost every project I was involved in.
For example with APEX Office Print our plugin is sending JSON to the backend. This JSON you can actually see yourself in the Remote Debug screen in your AOP Dashboard.
Another example is the wearables project (IoT) I showed at KScope 16; the wearable is sending data to a smartphone or tablet, which in his turn is doing a call to our backend (in ORDS) and sending JSON across.

At the end of the day we want the data in the Oracle Database, so our APEX apps can work with that data.

Since **Oracle DB 12c**, JSON is supported straight from the database. I wrote a number of blog posts how to read JSON from SQL within the database. Here's a quick demo of JSON in the database:

```
SQL> create table tbl_with_json (
  2    json_clob  clob,
  3    constraint json_clob_chk check (json_clob is json)
  4  );

Table TBL_WITH_JSON created.
```

My next public appearances:

- APEXConnect 2017: 9/11-MAY-2017
- ODTUG KScope: 25/29-JUN-2017

About Me



Dimitri Gielis

Passionate about Oracle and especially Oracle Application Express (APEX). Founder and CEO of APEX R&D, a company that provides Development, Training and Consulting in APEX.

View my complete profile

ORACLE ACE Director

...

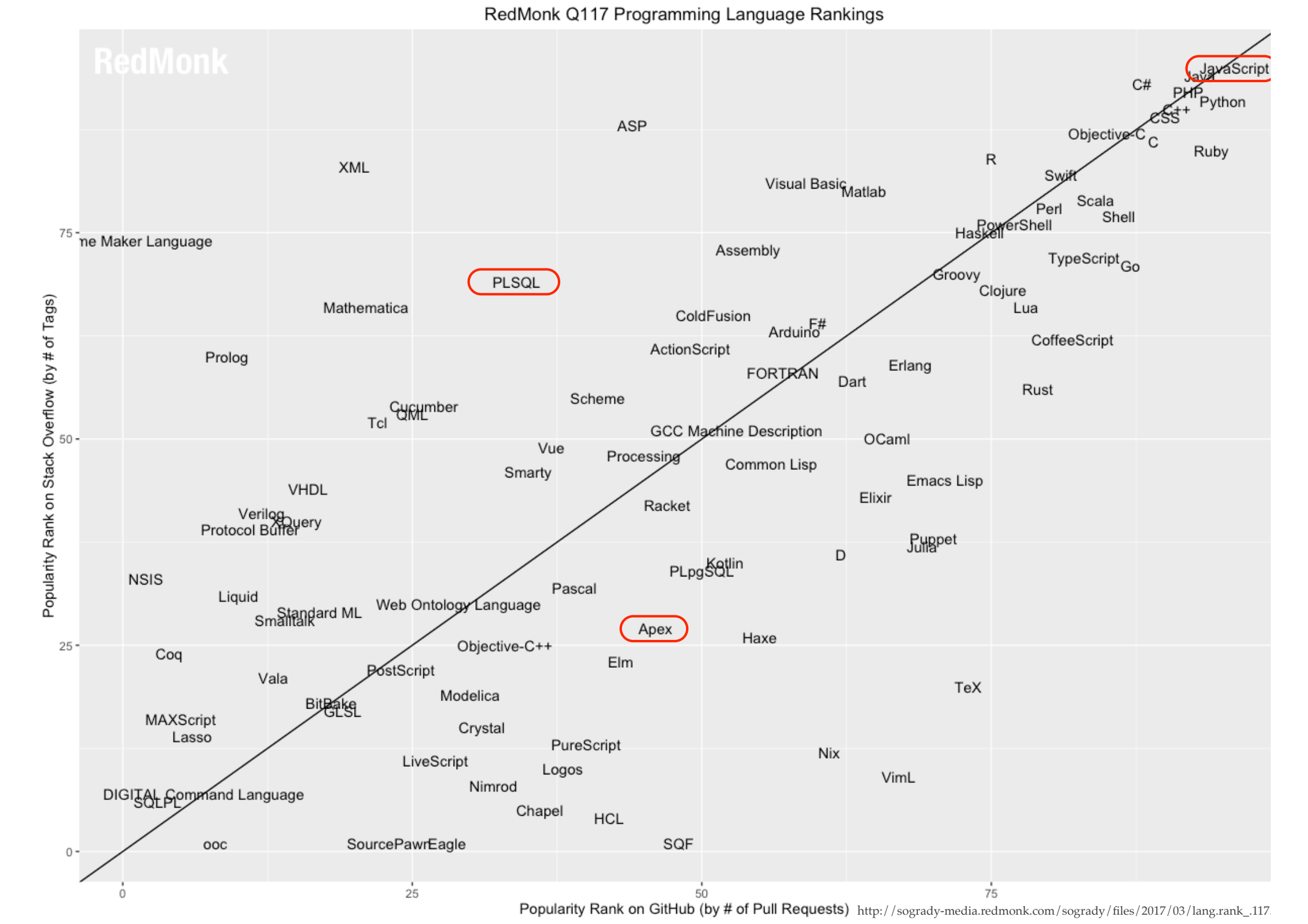
I was blown away by this... and I see a lot of potential be able to run JavaScript against the database.

There's actually a way to load Nashorn in your database too, so you can do JavaScript, Node.JS etc. straight from your database. Nashorn came with Java 8, but it should run in Java 7 too, now the default version of Java in the Oracle Database is 6, so there're some extra steps to do to get it to work. Running JavaScript from the database is something I've on my list to do R&D in and I actually submitted an abstract to KScope17 where I will present my results on this topic (if it gets accepted!) :)

...

Benefits Running JavaScript in Oracle DB

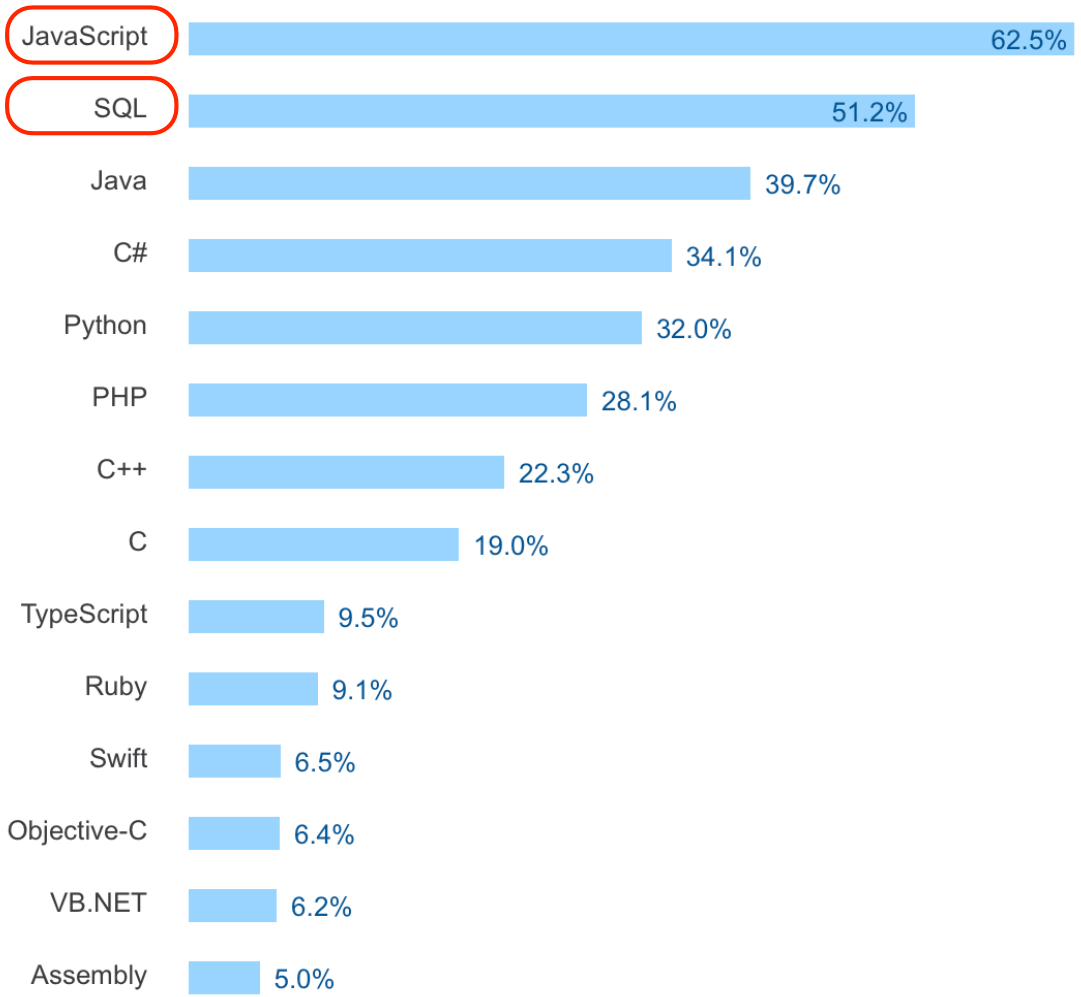
- Achieving new database capabilities
- In-place and faster processing of JSON documents
- Reusing existing skills and code


RedMonk Q117 Programming Language Rankings
RedMonk
JavaScript
Java
C#
PHP
Python
C++
CSS
Objective-C
C
Ruby
R
Swift
Visual Basic
Matlab
Scala
Perl
Shell
PowerShell
Haskell
ASP
XML
ne Maker Language
Assembly
TypeScript
Go
Groovy
Clojure
Lua
PLSQL
Mathematica
ColdFusion
F#
Arduino
CoffeeScript
ActionScript
Prolog
Erlang
FORTRAN
Dart
Rust
Scheme
Cucumber
QML
Tcl
GCC Machine Description
OCaml
Vue
Processing
Common Lisp
Smarty
Emacs Lisp
VHDL
Elixir
Racket
Verilog
XQuery
Protocol Buffer
Puppet
Julia
D
Kotlin
PLpgSQL
NSIS
Pascal
Liquid
Web Ontology Language
Standard ML
Smalltalk
Apex
Haxe
Objective-C++
Coq
Elm
PostScript
Vala
TeX
Modelica
BitBake
GLSL
MAXScript
Lasso
Crystal
PureScript
Nix
LiveScript
Logos
VimL
Nimrod
DIGITAL Command Language
SQLPL
Chapel
HCL
ooc
SourcePawrEagle
SQF
Popularity Rank on Stack Overflow (by # of Tags)
Popularity Rank on GitHub (by # of Pull Requests)
0
25
50
75
http://sogrady-media.redmonk.com/sogrady/files/2017/03/lang.rank_.117

Most Popular Technologies

Programming Languages

% of This Category | % of All Respondents | % of Professional Developers



https://insights.stackoverflow.com/survey/2017#technology


HOW?


Nashorn JavaScript Engine

Nashorn is a JavaScript engine developed in the Java programming language by Oracle. It is based on the Da Vinci Machine (JSR 292) and has been released with Java 8. The project was announced first at the JVM language summit in July 2011, and then confirmed at JavaOne in October 2011.

Nashorn (JavaScript engine) - Wikipedia
https://en.wikipedia.org/wiki/Nashorn_(JavaScript_engine)

STARTING



with Nashorn

A simple way to get started with Oracle Nashorn is to run JavaScript programs from the command line.

http://www.oracle.com/technetwork/articles/java/jf14-nashorn-2126515.html

Java SE
Java EE
Java ME
Java SE Support
Java SE Advanced & Suite
Java Embedded
Java DB
Web Tier
Java Card
Java TV
New to Java
Community
Java Magazine

Overview | Downloads | Documentation | Community | Technologies | Training

Java SE Downloads



Java Platform (JDK) 8u131



NetBeans with JDK 8

Java Platform, Standard Edition

Java SE 8u131

Java SE 8u131 includes important security fixes and bug fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release.
Learn more

Important planned change for MD5-signed JARs
Starting with the April Critical Patch Update releases, planned for April 18 2017, all JRE versions will treat JARs signed with MD5 as unsigned. Learn more and view testing instructions.
For more information on cryptographic algorithm support, please check the JRE and JDK Crypto Roadmap.

- Installation Instructions
- Release Notes
- Oracle License
- Java SE Products
- Third Party Licenses
- Certified System Configurations
- Readme Files
  - JDK ReadMe
  - JRE ReadMe

JDK DOWNLOAD

Server JRE DOWNLOAD

JRE DOWNLOAD

Java SDKs and Tools
- Java SE
- Java EE and Glassfish
- Java ME
- Java Card
- NetBeans IDE
- Java Mission Control

Java Resources
- Java APIs
- Technical Articles
- Demos and Videos
- Forums
- Java Magazine
- Java.net
- Developer Training
- Tutorials
- Java.com

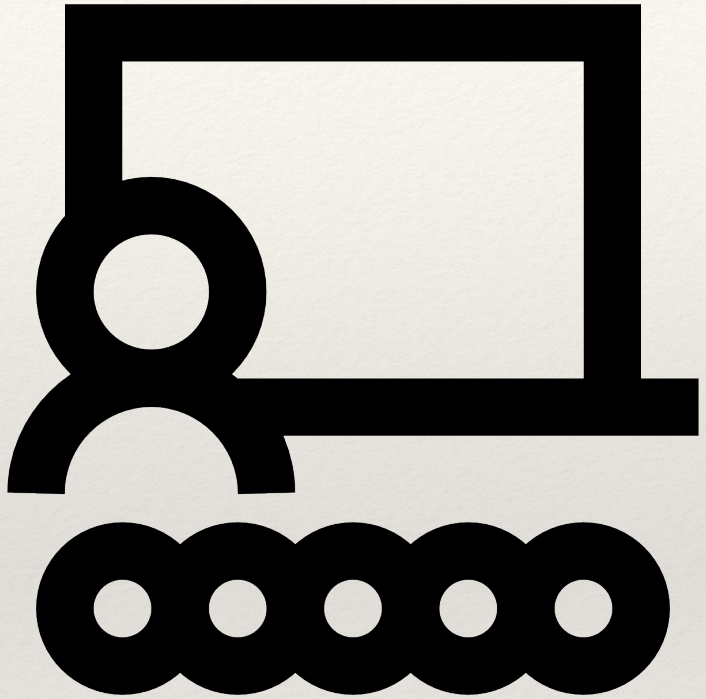
```
var hello = function() {
  print("Hello Nashorn!");
};

hello();
```



```
jjs hello.js
```





Demo



Nashorn in the Oracle Database

Step 1

Get the necessary privilege

grant DBJAVASCRIPT to hr;



Step 2

Create your JavaScript function

```
function hello(){
  var h = "Hello Nashorn!";
  return h;
}

var output = hello();
print(output);
```



Step 3

Upload your JS file to the Database

loadjava -v -u hr hello.js



Step 4

Check JavaScript is in Database

```
select object_name, object_type
  from user_objects
 where object_type = 'JAVA RESOURCE'
```

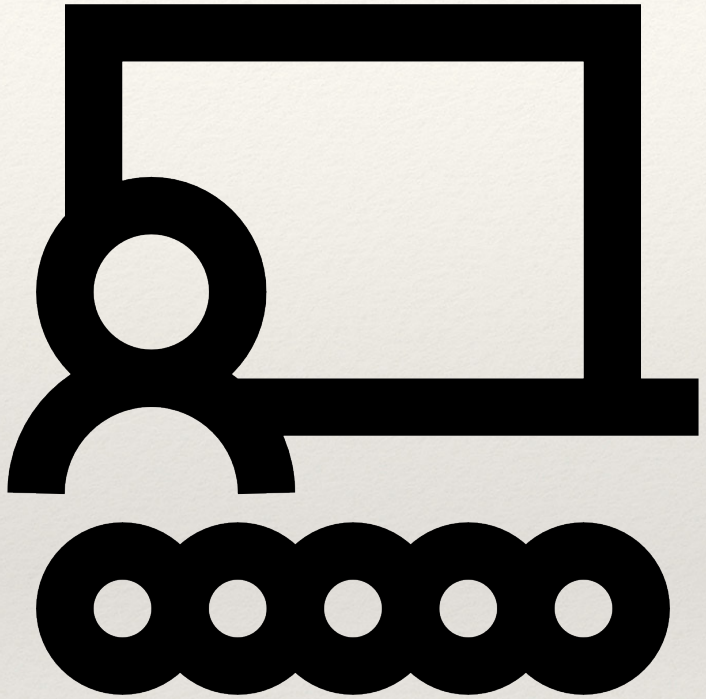


Step 5

Execute!

```
begin
  dbms_javascript.run('hello.js');
end;
/
```

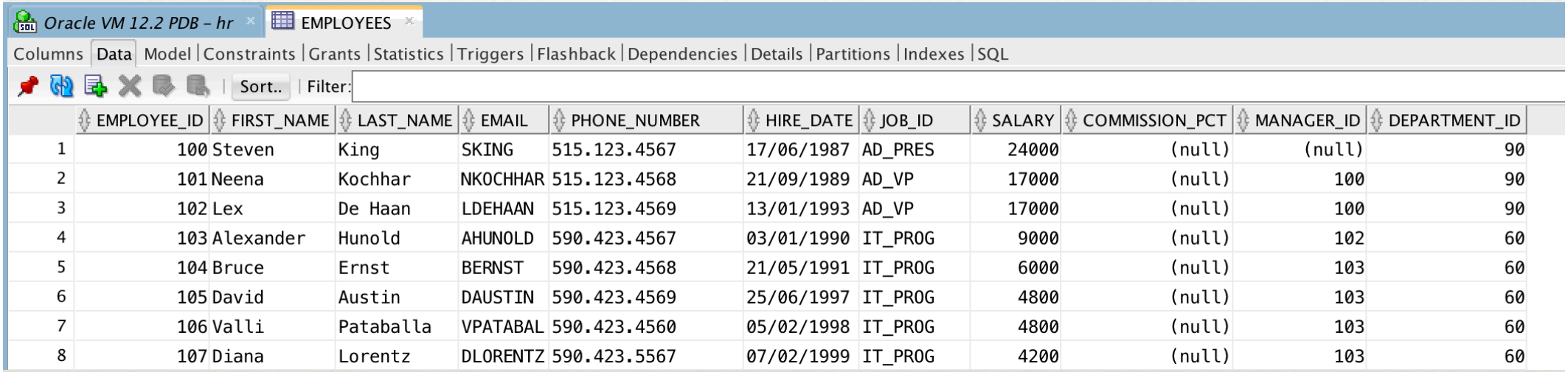




Demo


Table Data

Table



	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	100	Steven	King	SKING	515.123.4567	17/06/1987	AD_PRES	24000	(null)	(null)	90
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21/09/1989	AD_VP	17000	(null)	100	90
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13/01/1993	AD_VP	17000	(null)	100	90
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03/01/1990	IT_PROG	9000	(null)	102	60
5	104	Bruce	Ernst	BERNST	590.423.4568	21/05/1991	IT_PROG	6000	(null)	103	60
6	105	David	Austin	DAUSTIN	590.423.4569	25/06/1997	IT_PROG	4800	(null)	103	60
7	106	Valli	Pataballa	VPATABAL	590.423.4560	05/02/1998	IT_PROG	4800	(null)	103	60
8	107	Diana	Lorentz	DLORENTZ	590.423.5567	07/02/1999	IT_PROG	4200	(null)	103	60

JavaScript in DB way

Step 1: Privilege (done)

Step 2: Create JavaScript

Step 3: Load into DB

Step 4: Execute

```
var selectQuery = function(pId) {
  var Driver = Packages.oracle.jdbc.OracleDriver;
  var oracleDriver = new Driver();
  var url = "jdbc:default:connection:";
  var connection = oracleDriver.defaultConnection();

  // Prepare statement
  var query = "SELECT e.employee_id, e.first_name, e.last_name
                 FROM employees e
                WHERE e.employee_id = ?";
  var preparedStatement = connection.prepareStatement(query);
  preparedStatement.setString(1, pId);

  // execute Query
  var resultSet = preparedStatement.executeQuery();

  // display results
  while (resultSet.next()) {
    var output = resultSet.getString("FIRST_NAME");
  }

  // cleanup
  resultSet.close();
  preparedStatement.close();
  connection.close();
  return output;
```



```
var selectQuery = function(pId) {
  var Driver = Packages.oracle.jdbc.OracleDriver;
  var oracleDriver = new Driver();
  var url = "jdbc:default:connection:";
  var connection = oracleDriver.defaultConnection();

  // Prepare statement
  var query = "SELECT e.employee_id, e.first_name, e.last_name
                 FROM employees e
                WHERE e.employee_id = ?";
  var preparedStatement = connection.prepareStatement(query);
  preparedStatement.setString(1, pId);

  // execute Query
  var resultSet = preparedStatement.executeQuery();

  // display results
  while (resultSet.next()) {
    var output = resultSet.getString("FIRST_NAME");
  }

  // cleanup
  resultSet.close();
  preparedStatement.close();
  connection.close();
  return output;
```

Setup DB connection



```
var selectQuery = function(pId) {
  var Driver = Packages.oracle.jdbc.OracleDriver;
  var oracleDriver = new Driver();
  var url = "jdbc:default:connection:";
  var connection = oracleDriver.defaultConnection();

  // Prepare statement
  var query = "SELECT e.employee_id, e.first_name, e.last_name
                 FROM employees e
                WHERE e.employee_id = ?";
  var preparedStatement = connection.prepareStatement(query);
  preparedStatement.setString(1, pId);

  // execute Query
  var resultSet = preparedStatement.executeQuery();

  // display results
  while (resultSet.next()) {
    var output = resultSet.getString("FIRST_NAME");
  }

  // cleanup
  resultSet.close();
  preparedStatement.close();
  connection.close();
  return output;
```

SQL Statement



```
var selectQuery = function(pId) {
  var Driver = Packages.oracle.jdbc.OracleDriver;
  var oracleDriver = new Driver();
  var url = "jdbc:default:connection:";
  var connection = oracleDriver.defaultConnection();

  // Prepare statement
  var query = "SELECT e.employee_id, e.first_name, e.last_name
                 FROM employees e
                WHERE e.employee_id = ?";
  var preparedStatement = connection.prepareStatement(query);
  preparedStatement.setString(1, pId);

  // execute Query
  var resultSet = preparedStatement.executeQuery();

  // display results
  while (resultSet.next()) {
   var output = resultSet.getString("FIRST_NAME");
  }

  // cleanup
  resultSet.close();
  preparedStatement.close();
  connection.close();
  return output;
```

Execute



```
var selectQuery = function(pId) {
  var Driver = Packages.oracle.jdbc.OracleDriver;
  var oracleDriver = new Driver();
  var url = "jdbc:default:connection:";
  var connection = oracleDriver.defaultConnection();

  // Prepare statement
  var query = "SELECT e.employee_id, e.first_name, e.last_name
                 FROM employees e
                WHERE e.employee_id = ?";
  var preparedStatement = connection.prepareStatement(query);
  preparedStatement.setString(1, pId);

  // execute Query
  var resultSet = preparedStatement.executeQuery();

  // display results
  while (resultSet.next()) {
    var output = resultSet.getString("FIRST_NAME");
  }

  // cleanup
  resultSet.close();
  preparedStatement.close();
  connection.close();
  return output;
```

Display Results



```
var selectQuery = function(pId) {
  var Driver = Packages.oracle.jdbc.OracleDriver;
  var oracleDriver = new Driver();
  var url = "jdbc:default:connection:";
  var connection = oracleDriver.defaultConnection();

  // Prepare statement
  var query = "SELECT e.employee_id, e.first_name, e.last_name
                 FROM employees e
                WHERE e.employee_id = ?";
  var preparedStatement = connection.prepareStatement(query);
  preparedStatement.setString(1, pId);

  // execute Query
  var resultSet = preparedStatement.executeQuery();

  // display results
  while (resultSet.next()) {
   var output = resultSet.getString("FIRST_NAME");
  }

  // cleanup
  resultSet.close();
  preparedStatement.close();
  connection.close();
  return output;
```

Cleanup and Return



loadjava -v -u hr query.js



```
create or replace and compile java source named "InvokeScript" as
import javax.script.*;
import java.net.*;
import java.io.*;

public class InvokeScript {
   public static String eval(String inputId) throws Exception {
      String output = new String();
   try {
      // create a script engine manager
      ScriptEngineManager factory = new ScriptEngineManager();

      // create a JavaScript engine
      ScriptEngine engine = factory.getEngineByName("javascript");

      //read the script as a java resource
      engine.eval(new InputStreamReader(InvokeScript.class.getResourceAsStream("query.js")));

      Invocable invocable = (Invocable) engine;
      Object selectResult = invocable.invokeFunction("selectQuery", inputId);
      output = selectResult.toString();
   } catch(Exception e) {
            output =e.getMessage();
      }
      return output;
   }
}
/
```



```
create or replace and compile java source named "InvokeScript" as
import javax.script.*;
import java.net.*;
import java.io.*;

public class InvokeScript {
   public static String eval(String inputId) throws Exception {
      String output = new String();
   try {
      // create a script engine manager
      ScriptEngineManager factory = new ScriptEngineManager();

      // create a JavaScript engine
      ScriptEngine engine = factory.getEngineByName("javascript");

      //read the script as a java resource
      engine.eval(new InputStreamReader(InvokeScript.class.getResourceAsStream("query.js")));

      Invocable invocable = (Invocable) engine;
      Object selectResult = invocable.invokeFunction("selectQuery", inputId);
      output = selectResult.toString();
   } catch(Exception e) {
            output =e.getMessage();
      }
      return output;
   }
}
```

Enable JavaScript engine



```
create or replace and compile java source named "InvokeScript" as
import javax.script.*;
import java.net.*;
import java.io.*;

public class InvokeScript {
   public static String eval(String inputId) throws Exception {
      String output = new String();
   try {
      // create a script engine manager
      ScriptEngineManager factory = new ScriptEngineManager();

      // create a JavaScript engine
      ScriptEngine engine = factory.getEngineByName("javascript");

      //read the script as a java resource
      engine.eval(new InputStreamReader(InvokeScript.class.getResourceAsStream("query.js")));

      Invocable invocable = (Invocable) engine;
      Object selectResult = invocable.invokeFunction("selectQuery", inputId);
      output = selectResult.toString();
   } catch(Exception e) {
            output =e.getMessage();
      }
      return output;
   }
}
```

Tell which file



```
create or replace and compile java source named "InvokeScript" as
import javax.script.*;
import java.net.*;
import java.io.*;

public class InvokeScript {
   public static String eval(String inputId) throws Exception {
      String output = new String();
   try {
      // create a script engine manager
      ScriptEngineManager factory = new ScriptEngineManager();

      // create a JavaScript engine
      ScriptEngine engine = factory.getEngineByName("javascript");

      //read the script as a java resource
      engine.eval(new InputStreamReader(InvokeScript.class.getResourceAsStream("query.js")));

      Invocable invocable = (Invocable) engine;
      Object selectResult = invocable.invokeFunction("selectQuery", inputId);
      output = selectResult.toString();
   } catch(Exception e) {
            output =e.getMessage();
      }
      return output;
   }
}
```

Call function with param

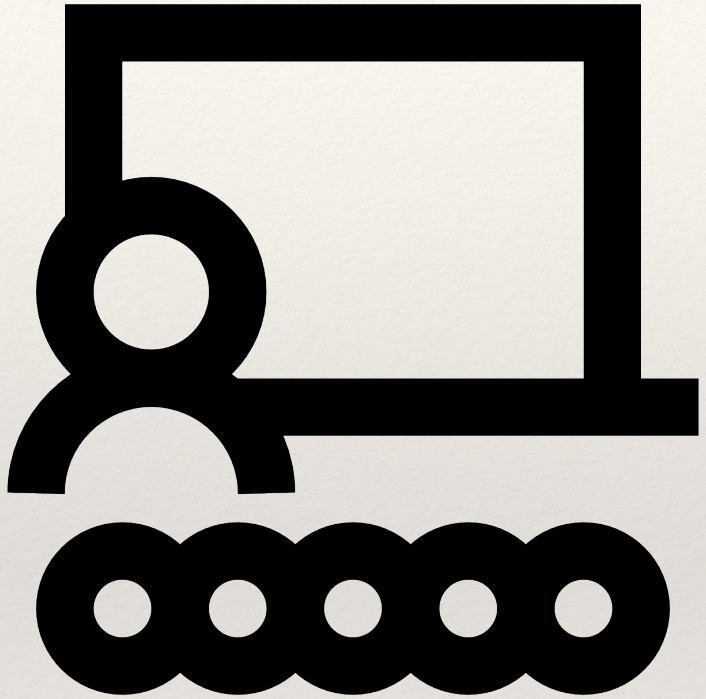


```
CREATE OR REPLACE FUNCTION invokeScriptEval(inputId varchar2)
return varchar2 as language java
name 'InvokeScript.eval(java.lang.String)
return java.lang.String';
/
```



```
SELECT invokeScriptEval(100)
   FROM dual;
```





Demo



J'SON

Query JSON data

Table

```
CREATE TABLE js_rest (
    id          NUMBER NOT NULL,
    json_clob   CLOB,
    CONSTRAINT js_rest_pk PRIMARY KEY (id),
    CONSTRAINT js_rest_json_chk CHECK (json_clob IS JSON)
);
```

Data

```
INSERT INTO js_rest (id, json_clob)
VALUES (1,
        '{
          "id"      : "1",
          "firstname" : "Dimitri",
          "lastname"  : "Gielils",
          "company"   : "APEX RnD",
          "email"     : "dgielis@apexrnd.be",
          "picture:   : null,
          "address"   : {
                          "City" : "Leuven",
                          "Country" : "Belgium"
                        }
        }');
```

Normal Database way

```
SELECT a.json_clob.id
  FROM js_rest a;
```

Works? (>32k)

```
SELECT a.json_clob.picture
  FROM js_rest a;
```



JavaScript in DB way

Step 1: Privilege (done)

Step 2: Create JavaScript

Step 3: Load into DB

Step 4: Execute

```
var selectQuery = function(pId) {
  var Driver = Packages.oracle.jdbc.OracleDriver;
  var oracleDriver = new Driver();
  var url = "jdbc:default:connection:";
  var connection = oracleDriver.defaultConnection();

  // Prepare statement
  var query = "SELECT a.json_clob FROM js_rest a WHERE a.id = ?";
  var preparedStatement = connection.prepareStatement(query);
  preparedStatement.setString(1, pId);

  // execute Query
  var resultSet = preparedStatement.executeQuery();

  // display results
  while (resultSet.next()) {
    var output = resultSet.getString(1);
  }

  // cleanup
  resultSet.close();
  preparedStatement.close();
  connection.close();
  return output;
}
```



```
var selectQuery = function(pId) {
  var Driver = Packages.oracle.jdbc.OracleDriver;
  var oracleDriver = new Driver();
  var url = "jdbc:default:connection:";
  var connection = oracleDriver.defaultConnection();

  // Prepare statement
  var query = "SELECT a.json_clob FROM js_rest a WHERE a.json_clob.id = ?";
  var preparedStatement = connection.prepareStatement(query);
  preparedStatement.setString(1, pId);

  // execute Query
  var resultSet = preparedStatement.executeQuery();

  // display results
  while (resultSet.next()) {
    var output = resultSet.getString(1);
  }

  // cleanup
  resultSet.close();
  preparedStatement.close();
  connection.close();
  return output;
}
```



```
var selectQuery = function(pId) {
  // setup connection
  …

  // Prepare statement
  var query = "SELECT a.json_clob FROM js_rest a WHERE a.id = ?";
  var preparedStatement = connection.prepareStatement(query);
  preparedStatement.setString(1, pId);

  // execute Query
  var resultSet = preparedStatement.executeQuery();
  while (resultSet.next()) {
    var jsonClob = resultSet.getString(1);
  }

  /* parse JSON */
  var obj = JSON.parse(jsonClob);
  output = obj.picture;

  // cleanup
  …
  return output;
}
```



loadjava -v -u hr query.js



```
create or replace and resolve java source named "InvokeScript" as
import javax.script.*;
import oracle.sql.*;
import oracle.jdbc.*;
import java.sql.*;
import java.net.*;
import java.io.*;

public class InvokeScript {
    public static oracle.sql.CLOB eval(String inputId) throws Exception {
        String output = new String();
        Connection con =
DriverManager.getConnection("jdbc:default:connection:");
        CLOB cl = CLOB.createTemporary(con, true, CLOB.DURATION_CALL);

    try {
        ScriptEngineManager factory = new ScriptEngineManager();
        ScriptEngine engine = factory.getEngineByName("javascript");
        engine.eval(new
InputStreamReader(InvokeScript.class.getResourceAsStream("query.js")));
        Invocable invocable = (Invocable) engine;
        Object selectResult = invocable.invokeFunction("selectQuery",
inputId);
        output = selectResult.toString();
    } catch(Exception e) {
                output = e.getMessage();
        }
        cl.putString(1, output);
        return cl;
    }
}
```

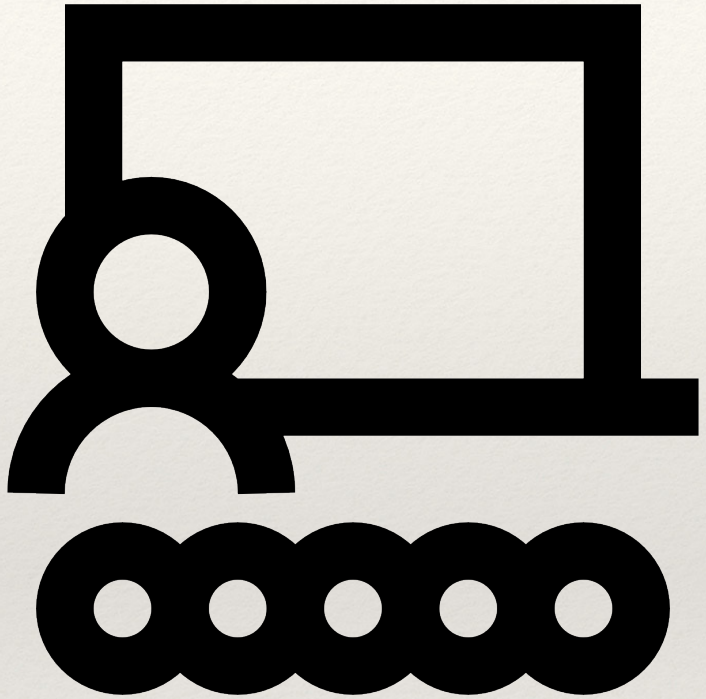


```
CREATE OR REPLACE FUNCTION invokeScriptEval(inputId varchar2)
return clob as language java
name 'InvokeScript.eval(java.lang.String)
return oracle.sql.CLOB';
/
```



```
SELECT invokeScriptEval(1)
  FROM dual;
```





Demo


SODA
Simple Oracle Document Access

http://www.oracle.com/technetwork/database/application-development/oracle-document-store/index.html

Database 12c
Database In-Memory
Multitenant
More Key Features
Application Development
Big Data Appliance
Cloud Database Services
Private Database Cloud
Data Warehousing & Big Data
Database Appliance
Exadata Database Machine
High Availability
Manageability
Migrations
Security
Unstructured Data
Upgrades
Windows
Database Technology Index

Oracle as a Document Store

Oracle Database 12c fully supports schemaless application development using the JSON data model. This allows for a hybrid development approach: all of the schema flexibility and speedy application development of NoSQL document stores, combined with all of the enterprise-ready features in Oracle Database 12c.



White Paper: Schemaless Application



Video: JSON in Oracle Database 12c

Critical Capabilities for Operational Database Management Systems

Read Gartner's Report ›

Oracle Live SQL

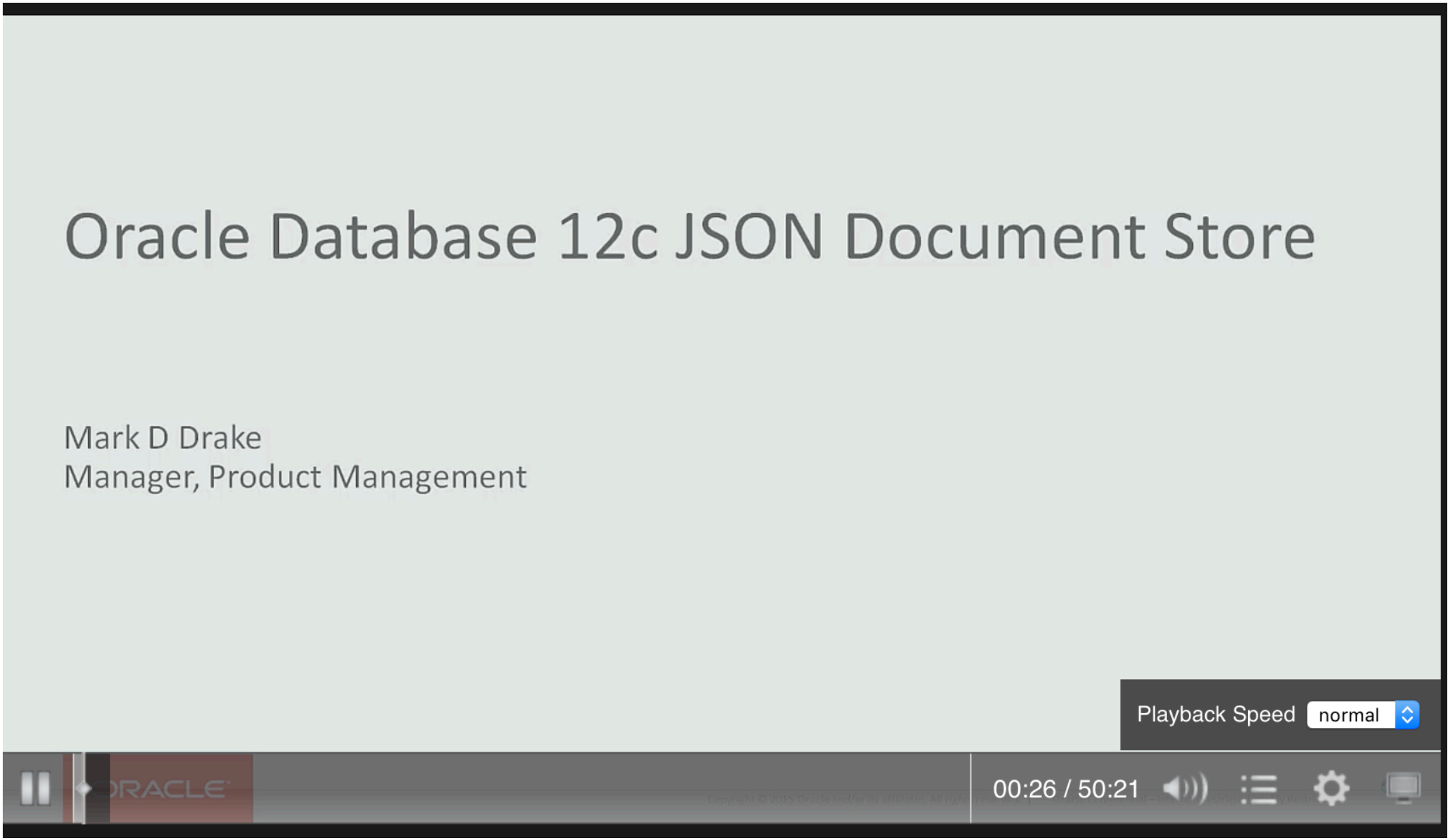
Learn and Share SQL

Try Now ›

Get the Latest Oracle Database 12c Tutorials

Oracle Database 12c JSON Document Store

Mark D Drake
Manager, Product Management



Program Agenda

1. **Next Generation Application Development**
2. Oracle 12c as JSON Document Store
3. Simple Oracle Document Access (SODA)
4. SODA for Java
5. SODA for REST
6. Query, Reporting and Analysis on JSON documents
7. Summary

ORACLE

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |



Kuassim Add files via upload 4dc4ba9 21 days ago

1 contributor

62 lines (34 sloc) 2.41 KB Raw Blame History

SODA for Java in the Database (OJVM)

SODA for Java is Oracle's fluent Java API for accessing JSON collections and documents without any knowledge of SQL. See Getting started with SODA for Java for more details on running testSODA.java with a client JVM (JDK or JRE). The goal of this write up is to furnish the steps for running testSODA.java in OJVM and manipulate the JSON collection directly in the database session without moving data around.

Requirements to run SODA with Java in the database

(i) Download the latest orajsoda.jar currently orajsoda-1.0.4.jar

(ii) Upload orasoda.jar in your database schema

```
loadjava -r -v -u hr/hr orajsoda-1.0.4.jar
```

(iii) Load the latest javax.json-1.0.4.jar



Kuassim Add files via upload 6d6067b 21 days ago

1 contributor

47 lines (24 sloc) | 1.99 KB | Raw | Blame | History

SODA for JavaScript Stored Procedures

SODA for Java is Oracle's fluent Java API for accessing JSON collections and documents without any knowledge of SQL.

See SODA for Java in the Database for running SODA with OJVM.

The goal of this write up is to furnish the steps for performing fluent JavaScript programming (i.e., No SQL and dot notation) in OJVM and manipulate the JSON collection directly in the database session without moving data around.

Requirements for SODA for JavaScript Stored Procedures

JavaScript in Oracle database is enabled using Nashorn in the embedded JVM (a.k.a. OJVM). Leveraging the interoperability of Java and JavaScript with Nashorn, we will reuse the SODA for Java APIs in JavaScript code.

(i) Follow steps (i), (ii) and (iii) in the section titled Requirements to run SODA with Java in the database

testSODA.js

(i) We've rewritten testSODA.java in JavaScript (i.e., testSODA.js), using SODA for Java API. Grap testSODA.js from Oracle


Loading External Libraries



Logic-less templates.

Mustache

```
<h1>{{header}}</h1>
{{#bug}}
{{/bug}}

{{#items}}
  {{#first}}
    <li><strong>{{name}}</strong></li>
  {{/first}}
  {{#link}}
    <li><a href="{{url}}">{{name}}</a></li>
  {{/link}}
{{/items}}

{{#empty}}
```

loadjava -v -u hr mustache.js



```
create or replace and compile java source named "InvokeMustache" as
import javax.script.*;
import java.net.*;
import java.io.*;

public class InvokeMustache {
    public static String eval(String template, String data) throws Exception {
        String output = new String();
    try {
        // create a script engine manager
        ScriptEngineManager factory = new ScriptEngineManager();

        // create a JavaScript engine
        ScriptEngine engine = factory.getEngineByName("javascript");

        //read the script as a java resource
        engine.eval(new
InputStreamReader(InvokeMustache.class.getResourceAsStream("mustache.js")));

        Invocable invocable = (Invocable) engine;
        Object selectResult = invocable.invokeFunction("callMustache", template, data);
        output = selectResult.toString();
    } catch(Exception e) {
            output =e.getMessage();
        }
        return output;
    }
}
```



```
CREATE OR REPLACE FUNCTION doMustache(template varchar2,
data varchar2)
return varchar2 as language java
name 'InvokeMustache.eval(java.lang.String)
return java.lang.String';
/
```



```
SELECT doMustache(t, d)
  FROM mustache_template
 WHERE id = 1;
```





http://momentjs.com

Moment.js 2.18.1

Parse, validate, manipulate, and display dates and times in JavaScript.

Download

moment.js

moment.min.js 16.6k gz

moment-with-locales.js

moment-with-locales.min.js 64.2k gz

Install

```
npm install moment --save    # npm
yarn add moment              # Yarn
Install-Package Moment.js    # NuGet
spm install moment --save    # spm
meteor add momentjs:moment   # meteor
bower install moment --save # bower (deprec
```

Format Dates

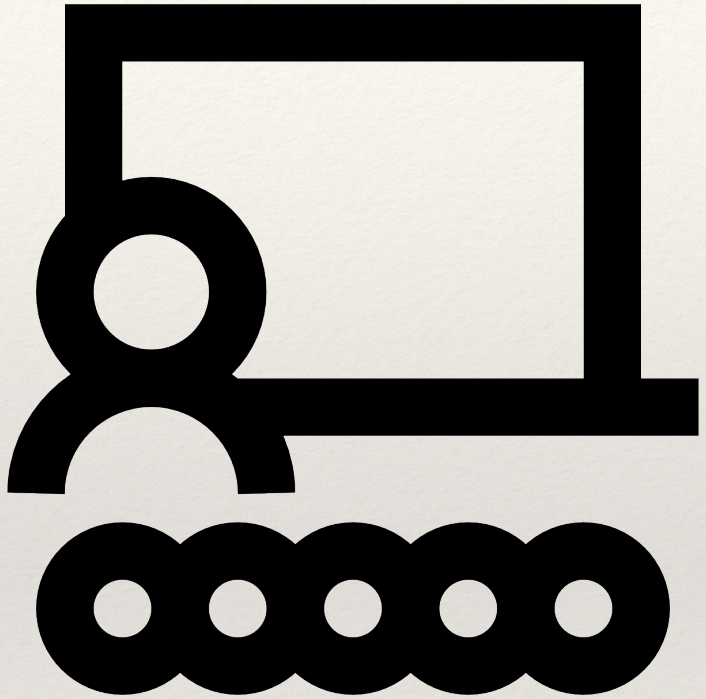
```
moment().format('MMMM Do YYYY, h:mm:ss a'); // June 21st 2017, 12:53:17 pm
moment().format('dddd');                    // Wednesday
```

UNDERSCORE.JS

Underscore is a JavaScript library that provides a whole mess of useful functional programming helpers without extending any built-in objects. It's the answer to the question: "If I sit down in front of a blank HTML page, and want to start being productive immediately, what do I need?" … and the tie to go along with jQuery's tux and Backbone's suspenders.

Underscore provides over 100 functions that support both your favorite workaday functional helpers: **map**, **filter**, **invoke** — as well as more specialized goodies: function binding, javascript templating, creating quick indexes, deep equality testing, and so on.

http://underscorejs.org

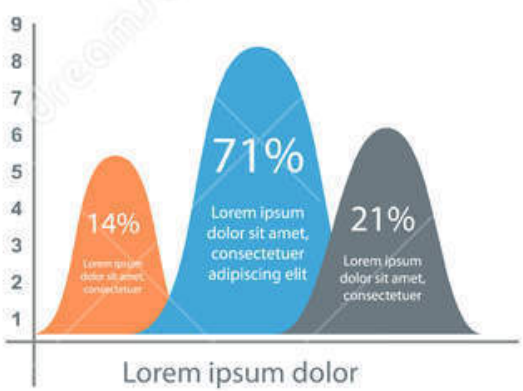


Demo

More things to explore…


node
JS
®

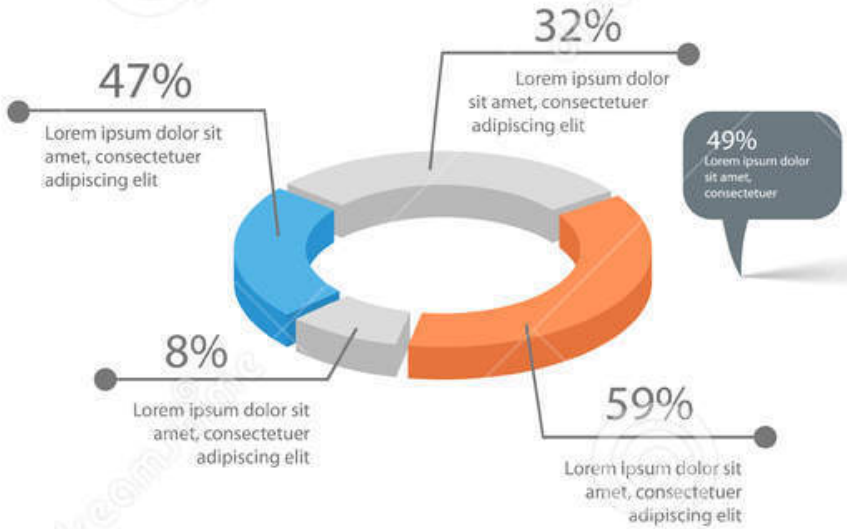


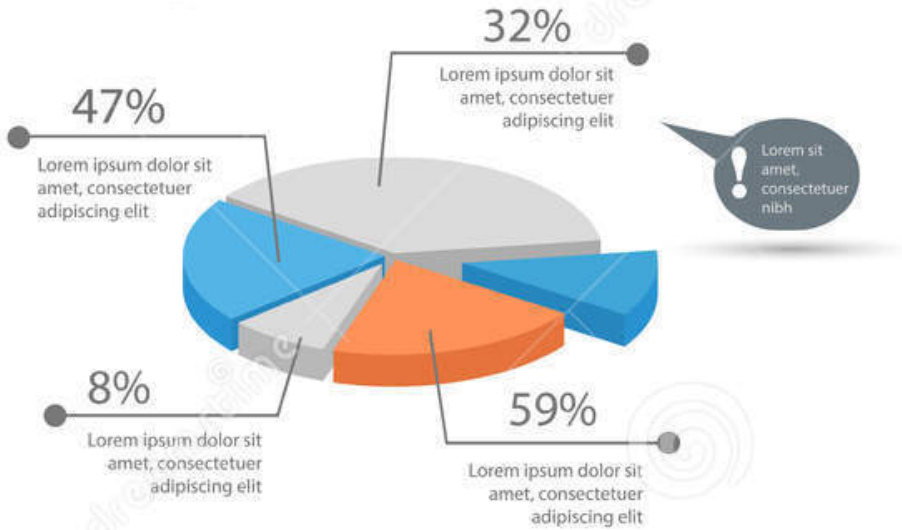

14%
71%
Lorem ipsum dolor sit amet, consectetuer adipiscing elit
21%
Lorem ipsum dolor sit amet, consectetuer
Lorem ipsum dolor

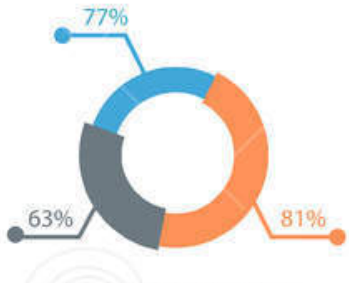

Lorem ipsum dolor sit amet, consectetuer
Lorem ipsum dolor sit amet, consectetuer
Lorem ipsum dolor sit amet, consectetuer


Lorem ipsum dolor

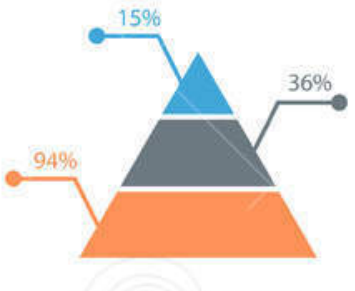

Lorem ipsum dolor sit amet, consectetuer
Lorem ipsum dolor sit amet, consectetuer
Lorem ipsum dolor sit amet, consectetuer


47%
Lorem ipsum dolor sit amet, consectetuer adipiscing elit
32%
Lorem ipsum dolor sit amet, consectetuer adipiscing elit
49%
Lorem ipsum dolor sit amet, consectetuer
8%
Lorem ipsum dolor sit amet, consectetuer adipiscing elit
59%
Lorem ipsum dolor sit amet, consectetuer adipiscing elit


47%
Lorem ipsum dolor sit amet, consectetuer adipiscing elit
32%
Lorem ipsum dolor sit amet, consectetuer adipiscing elit
Lorem sit amet, consectetuer nibh
8%
Lorem ipsum dolor sit amet, consectetuer adipiscing elit
59%
Lorem ipsum dolor sit amet, consectetuer adipiscing elit


77%
63%
81%
Lorem ipsum dolor


Lorem ipsum dolor

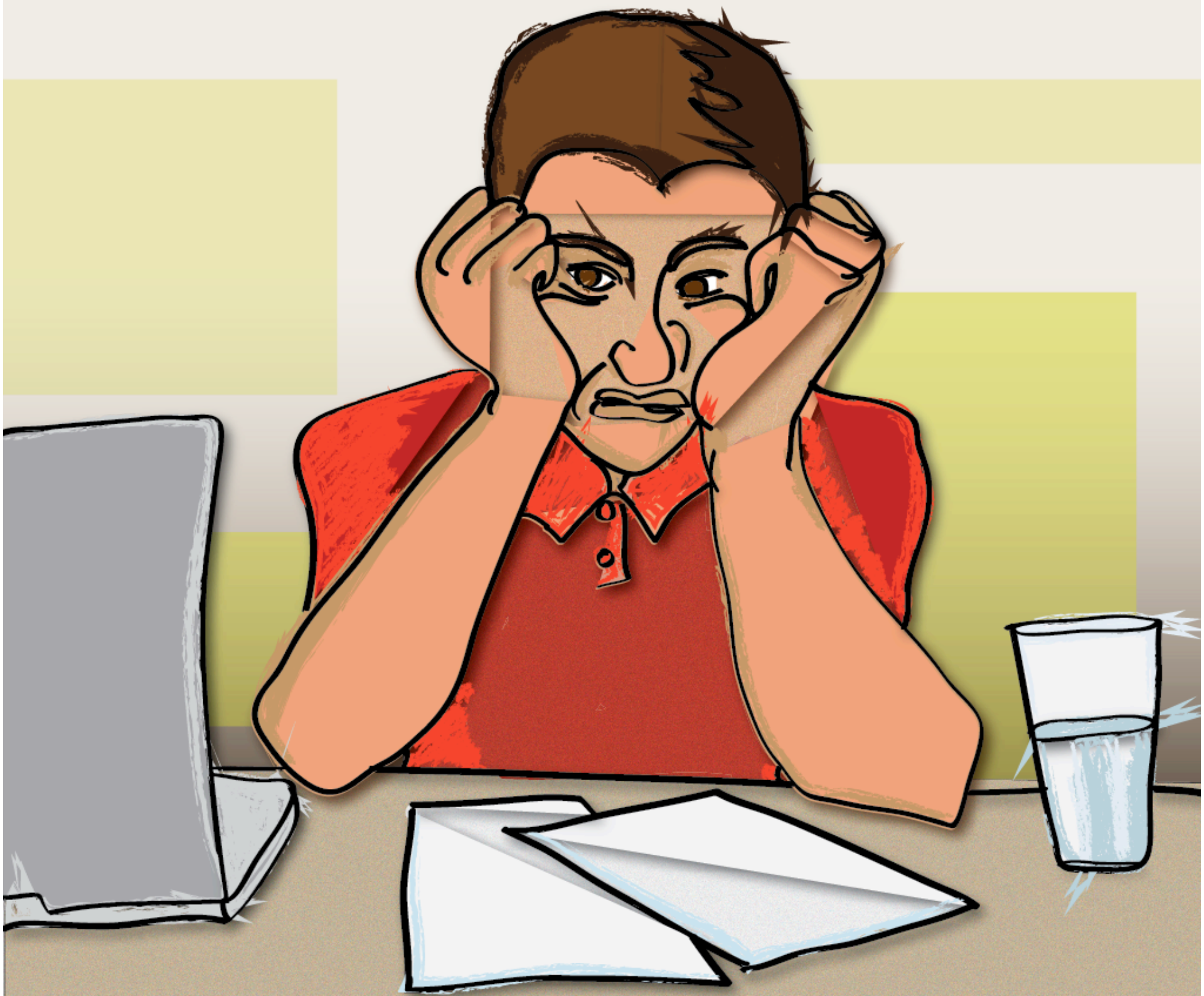

15%
36%
94%
Lorem ipsum dolor


Lorem ipsum dolor


How the hell do you use this thing?!
©TBoretti 2016 6/30

“While Oracle Nashorn runs ECMA-compliant JavaScript, it is important to note that objects normally **accessible in a web browser are not available**, for example, `console`, `window`, and so on.”

http://www.oracle.com/technetwork/articles/java/jf14-nashorn-2126515.html



☆ **kuassi mensah**

9 June 2017 at 00:10

KM

Re: presentation/demo JavaScript from the DB using Nashorn

To: Dimitri Gielis

I anticipate lot of hurdles with what you have in mind for example multi-threading, async operations, etc. Although the JVM in the database allows threaded Java code, it does not (and cannot) do preemptive scheduling; it's one thread at a time until it relinquishes control.

The JSON team has a Servlet running in the database for Native Web Services and added ORDS on top of it.

Hope this helps, Kuassi

See More from Dimitri Gielis

non-preemptive scheduling is not specific to OJVM but anything running in database session

More investigation necessary

- CommonJS Modules
- Polyfills for Nashorn (nashorn-polyfill.js)
- Compile JavaScript to one file
- WebJars

CommonJS Modules Support for Nashorn

This library adds support for CommonJS modules (aka `require`) inside a Nashorn script engine. It is based on the specification for NodeJS modules and it supports loading modules from the `node_modules` folder just as Node does. Of course, it doesn't provide an implementation for Node's APIs, so any module that depends on those won't work.

It is somehow similar to jvm-npm which I used before, but it is 100% implemented in Java and supports loading files through other means than the filesystem; you only need to implement a simple interface and you should be good to do. Also, having the implementation in Java allows using it with a JS interpreter on which access to Java packages has been disabled for sandboxing purposes.

Out-of-the-box, the library supports loading modules from the filesystem and from Java resources.

https://github.com/coveo/nashorn-commonjs-modules

jvm-npm

Support for NPM module loading in Javascript runtimes on the JVM. Implementation is based on http://nodejs.org/api/modules.html and should be fully compatible. This, of course, does not include the full node.js API, so don't expect all of the standard NPM modules that depend on it to work. If you want to use existing NPM modules that depend on the Node.js API, consider the not-yet-fully-baked nodyn project. If you are writing your own NPM modules in DynJS, Rhino or Nashorn, this should work just fine.

This module is known to work with DynJS, and has been briefly tested on Nashorn, and should work with Rhino as well.

https://github.com/nodyn/jvm-npm

Conclusion: JS in DB

- Interesting to follow and use when needed
- Need a specific skill-set
- Not that many “advanced” examples

Resources

- Oracle Database 12.2 VM: http://www.oracle.com/technetwork/database/enterprise-edition/databaseappdev-vm-161299.html
- Doc: http://docs.oracle.com/database/122/JJDEV/GUID-9D7B5AF2-5F63-4484-968D-01EFB6A2D50F.htm#JJDEV-GUID-9D7B5AF2-5F63-4484-968D-01EFB6A2D50F
- API: https://docs.oracle.com/javase/8/docs/technotes/guides/scripting/nashorn/api.html
- Article: http://www.oracle.com/technetwork/articles/java/jf14-nashorn-2126515.html
- Blog Post: http://www.n-k.de/riding-the-nashorn/#_loading_scripts
- Special thanks to: Carsten Czarski & Kuassi Mensah

Q&A

APEX R&D
research & development

www.apexRnD.be
dgielis.blogspot.com
@dgielis
dgielis@apexRnD.be

Consulting, Development, Training

- Looking for consulting, training and development in Oracle Application Express (APEX)?
- Contact : www.apexRnD.be
- Mail : info@apexRnD.be