ITOUG Tech Day 2025

Scripting in SQLcl

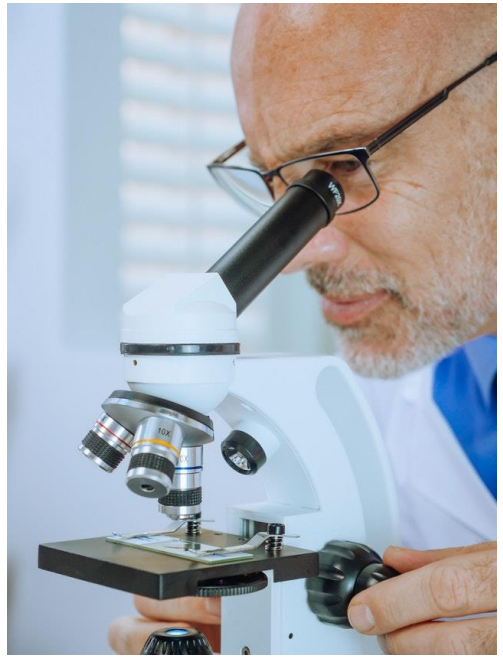
You can never have enough of a good thing

Erik van Roon

EvROCS
COMPLETING THE PUZZLE

Who Am I?

Erik van Roon



1995

ORACLE DATABASE
>=Oracle5

2009

EvROCS
COMPLETING THE PUZZLE

SYM42
Member of Symposium 42
https://sym42.org/

MASH Program

Oracle ACE Pro

: erik.van.roon@evrocs.nl
: www.evrocs.nl
: @evrocs_nl
: @evrocs.bsky.social

MASH PROGRAM

Mentor and Speaker Hub

BECOME A SPEAKER - MENTORING - EVOLVING - IMPROVING

.SYMPOSIUM 42

Created by the community, to support the community

Sharing of reliable knowledge
Supporting the various user groups and individuals

@sym_42

@sym42.bsky.social

https://sym42.org/

Oracle ACE

The Oracle ACE Program

400+ technical experts helping peers globally

- The Oracle ACE Program recognizes and rewards community members for their technical and community contributions to the Oracle community
- 3 membership levels: Director, Pro, and Associate
- Nominate yourself or a colleague at ace.oracle.com/nominate
- Learn more at ace.oracle.com

aceprogram_ww@oracle.com
Facebook.com/OracleACEs
@oracleace
Oracle ACE Program Group


What's that ?
SQLcl ???
Not a clue!?!

What is SQLcl?

Command line interface for the database

Like SQL*Plus, but with soooooo much more

This .....

```
SQLcl: Release 21.1 Production on Sun Apr 25 13:21:12 2021

Copyright (c) 1982, 2021, Oracle.  All rights reserved.

Last Successful login time: Sun Apr 25 2021 13:21:12 +02:00

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0


SQL>
```

Command history

Command completion

Aliases

Apex export

Liquibase integration

CD
PWD

In line editing

Codescan
(SQL Injection & performance warnings)

Syntax Highlighting

Code formatting

Load & Unload commands

Startup.sql

Datapump command

Query output formatting
(csv, json, insert statements, smart column widths, color highlighting)

DDL Command

Status Bar

Named Connections

Spool to zip file

Bridge command

Info / Info+
(Desc but a lot more information)

Project

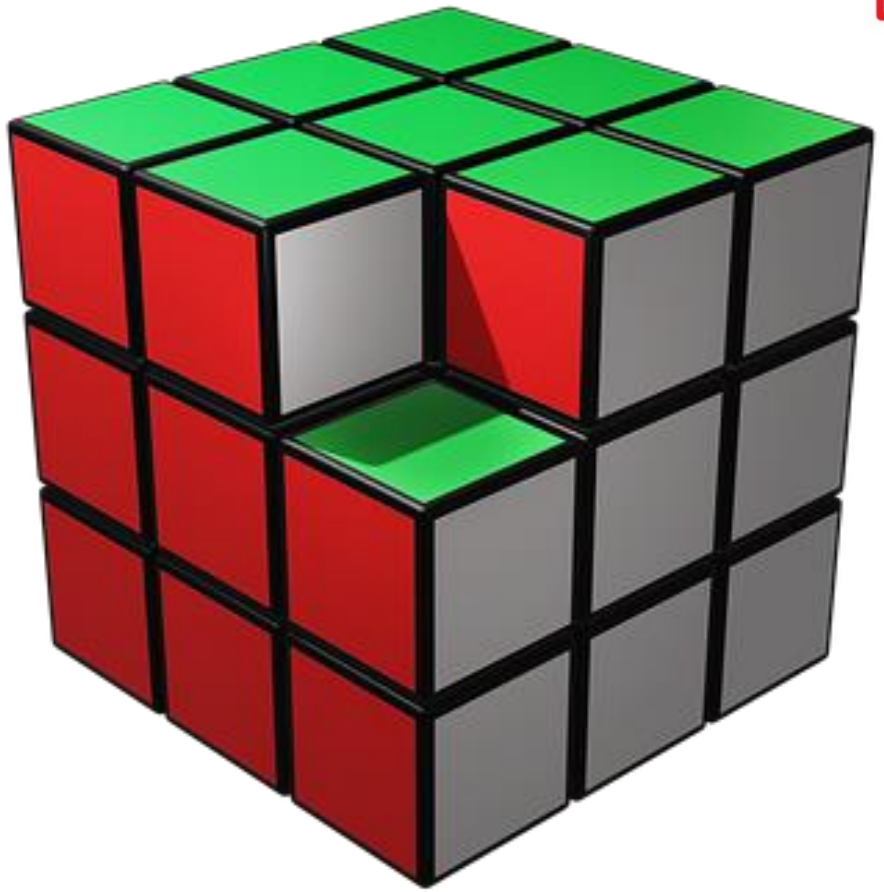
Dataguard command

... And much more...

Is it perfect?

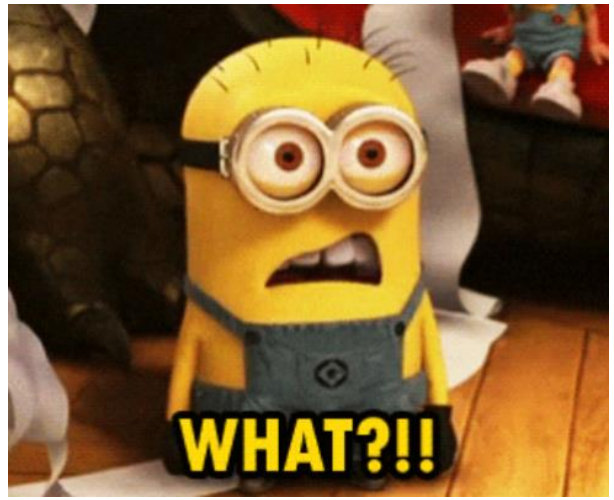
No!

No tool ever is……



Note/Disclaimer

- I wil show an example of builing functionality that does not exist in SQLcl
- I have been wating for this functionality for 30 years
- About 2 months after I started doing this talk……
- ….. this functionality got implemented in SQLcl



- Still, it is a good, not too complex example, so I still use it
- And I like my solution more than the one from Oracle 🤪

SQLcl implementation:
"argument" command
See "help argument"

```
ERO@EVROCS>help argument
Define SQL*Plus parameters to SQL scripts.
The argument command will add a DEFINE for the parameter if one does not exist.

Use the SET PARAMETERPOLICY command to control parameter retention.
The parameter retention applies to all parameters whether defined using this command or
not.

When SET PARAMETERPOLICY is SHARE (default), defined variables retain their values until
you:
  * Enter a new DEFINE command referencing the variable
  * Enter an UNDEFINE command referencing the variable
```

Disclaimer 2

- You will see JavaScript
- I'm not a JavaScript developer
- In fact: started looking at JavaScript to build SQLcl commands
- If you are comfortable with JavaScript you may see code that hurts
- Please, if you have tips, contact me afterwards and teach me!!

Example:

A SQL-script can have parameters.

Script SearchColumns.sql

```
select atc.table_name     "Table"
,      atc.column_name    "Column"
,      atc.data_type      "Data Type"
from   user_tab_columns     atc
where  atc.column_name like upper('&1')
  and  atc.data_type   like upper('&2')
order by 1, 2, 3
;

undef 1
undef 2
```

```
ERO@EVROCS>@searchcolumns %from %char%

                 Table            Column     Data Type
______________________ _________________ ____________
ERR$_BULK_ERRORS_PERF     CUST_EFF_FROM     VARCHAR2
EXT_TAB_TEST              DATE_FROM         VARCHAR2
EXT_TAB_TEST              VALID_FROM        VARCHAR2

3 rows selected.

ERO@EVROCS>_
```

But the parameters can not be optional

I want it to use "%" if I don't provide a second parameter value

But obviously…..

```
ERO@EVROCS>@searchcolumns %from
Enter value for 2: _
```

Solution 1: Use "Accept"

```
accept col_name prompt "Column name to look for: "
accept datatype prompt "Datatype of column     : "

select atc.table_name    "Table"
,      atc.column_name   "Column"
,      atc.data_type     "Data Type"
from   user_tab_columns    atc
where  atc.column_name like coalesce
                            (upper('&col_name'),'%')
  and  atc.data_type   like coalesce
                            (upper('&datatype'),'%')
order by 1, 2, 3
;

undef col_name
undef datatype
```

Awesome, but…….

The script is now interactive

```
ERO@EVROCS>@searchcolumns
Column name to look for: %from
Datatype of column     :

                    Table                          Column     Data Type
_________________________ ______________________________ ____________
BULK_ERRORS_PERF          CUST_EFF_FROM                  DATE
ERO_TEST_PLSQL_TESTS      CONFIDENCE95_INTERVAL_FROM     NUMBER
ERR$_BULK_ERRORS_PERF     CUST_EFF_FROM                  VARCHAR2
EXT_TAB_TEST              DATE_FROM                      VARCHAR2
EXT_TAB_TEST              VALID_FROM                     VARCHAR2
TECH_EXPERIENCE_17_PERF   CUST_EFF_FROM                  DATE

6 rows selected.

ERO@EVROCS>_
```

Solution 2: Use "column new_value"

```
column 1 new_value 1
column 2 new_value 2

set termout off
select null  "1"
,      null  "2"
from   dual
where  rownum = 0
;
set termout on

select atc.table_name     "Table"
,      atc.column_name    "Column"
,      atc.data_type      "Data Type"
from   user_tab_columns     atc
where  atc.column_name like coalesce(upper('&1'),'%')
  and  atc.data_type   like coalesce(upper('&2'),'%')
order by 1, 2, 3
;

undef 1
undef 2
```

```
ERO@EVROCS>@searchcolumns %from %char%

                    Table           Column    Data Type
_____________________ _______________ ___________
ERR$_BULK_ERRORS_PERF    CUST_EFF_FROM    VARCHAR2
EXT_TAB_TEST             DATE_FROM        VARCHAR2
EXT_TAB_TEST             VALID_FROM       VARCHAR2

3 rows selected.

ERO@EVROCS>@searchcolumns %from

                     Table                          Column    Data Type
_________________________ ______________________________ ___________
BULK_ERRORS_PERF          CUST_EFF_FROM                  DATE
ERO_TEST_PLSQL_TESTS      CONFIDENCE95_INTERVAL_FROM     NUMBER
ERR$_BULK_ERRORS_PERF     CUST_EFF_FROM                  VARCHAR2
EXT_TAB_TEST              DATE_FROM                      VARCHAR2
EXT_TAB_TEST              VALID_FROM                     VARCHAR2
TECH_EXPERIENCE_17_PERF   CUST_EFF_FROM                  DATE

6 rows selected.

ERO@EVROCS>_
```

Works as intended, but…..

- Is a bit of a hack
- Too many lines to get 1 thing done
- Needs termout off (in SQLcl) because "noprint" is not supported

What I would want…..

Something simple like this

```
VariableDefault 1 %
VariableDefault 2 %

select atc.table_name     "Table"
,      atc.column_name    "Column"
,      atc.data_type      "Data Type"
from   user_tab_columns     atc
where  atc.column_name like upper('&1')
  and  atc.data_type   like upper('&2')
order by 1, 2, 3
;

undef 1
undef 2
```

If substitution variable (1/2/…) exists:

Leave it alone

If it doesn't exist:

Create it with the default value ('%')

So, a command like this

```
VariableDefault <VariableName> <DefaultValue>
```

Example:

```
VariableDefault 2 %
```

Checks if variable 2 (second parameter) exists
If so : Leaves value of "2" as it is
If not : Assigns default value "%" to variable "2"

While we're at it, I want a third, optional, parameter

```
VariableDefault <VariableName> <DefaultValue> [<TargetVariable>]
```

Example:

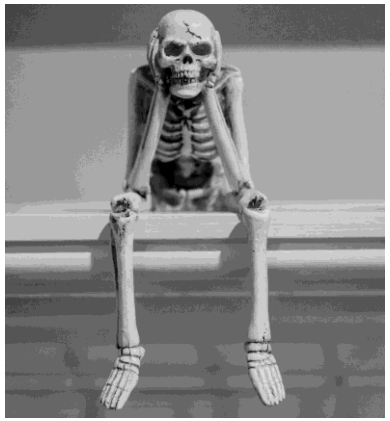
```
VariableDefault 2 % my_own_var
```

Checks if variable 2 (second parameter) exists
If so : Assigns value of "2" to variable "my_own_var"
If not : Assigns default value "%" to variable "my_own_var"

But that doesn't exist

What can we do?

1. We can wait until Oracle spontaneously implements it



2. We can keep asking Oracle for it

3. We can ……
   ……. implement it ourselves

We can run JavaScript scripts from SQLcl
(and/or java)

Current Version - 25.1.1

Requires JDK 11, 17 or 21

But Nashorn is removed in Java 15

So for JavaScript scripting:

JDK = 11 is required!

Requirements

- Supported Java Version
  - Oracle SQLcl requires Java 11 or 17 or 21. The supported Java Runtime Environments are:
    - Oracle Java 11
    - Oracle Java 17
    - Oracle Java 21
    - Oracle GraalVM Enterprise Edition for Java 11
    - Oracle GraalVM Enterprise Edition for Java 17
    - Oracle GraalVM Enterprise Edition for Java 21

Or: use GraalVM for Java 11, 17 or 21

I recommend this because:

- Fully supported by Oracle!
- JavaScript engine remains included
- Some JavaScript things not supported by Nashorn in JDK that *are* supported in JavaScript engine in GraalVM (Example: const)

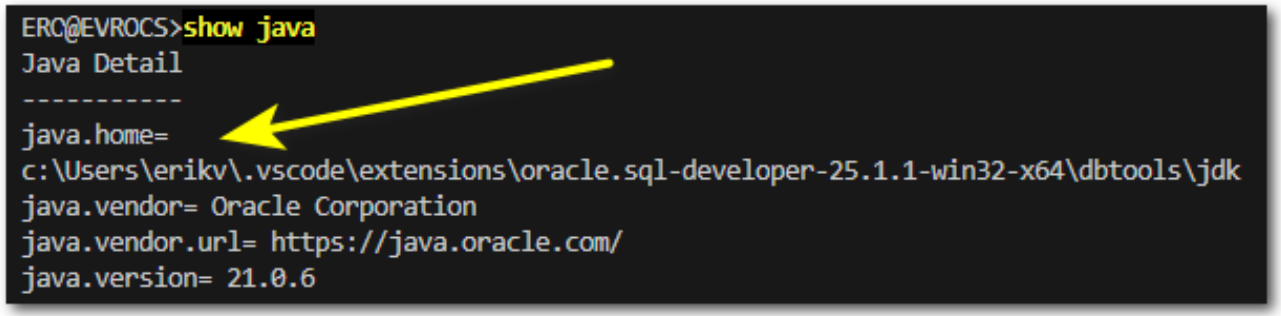
How about SQLcl in SQL Developer Extension for VS Code??

Standalone SQLcl

Uses Java in location defined by java_home environment variable
Can be set to anything

SQLcl in the Extension

Uses supplied Java (JDK) of the extension
Can currently not be changed in a supported way



So JavaScript won't work for now, Java will
Changing this currently has no priority
So if you want this too: make some noise on the forums

So we can execute JavaScript, you say?

From the command line

```
ERO@EVROCS>script
  2  ctx.write("Hello friends \n");
  3* /
Hello friends
ERO@EVROCS>_
```

(type "script" followed by JavaScript and terminated with slash)

From a script file

```
ERO@EVROCS>script hello.js
Hello friends
ERO@EVROCS>_
```

(type "script" followed by filename, possibly including path)

What would our script need to do?

Remember?

```
VariableDefault <VariableName> <DefaultValue> [<TargetVariable>]
```

It has to

- Accept 2 or 3 parameters
  1. <VariableName> - Name of the variable to check
  2. <DefaultValue> - Default value to use if variable doesn't exist
  3. <TargetVariable> - (Optional) Name of variable to set

- Check if a variable exists with name <VariableName>

- Set the value for a variable
  - If a variable name is given in <TargetVariable>, then <TargetVariable>
  - If not, then <VariableName>

Simple.....

Accepting 2 or 3 parameters….

The array "args" is available

- First element (index=0) is the name of the JS script being executed
- All following elements (index 1-n) are the parameter values

Execute this:

```
script MyJavascript.js param_1 param_2 param_3
```

And you will have:

```
args.length     = 4
args[0]         = MyJavascript.js
args[1]         = param_1
args[2]         = param_2
args[3]         = param_3
```

Checking if a substitution variable exists….

A "map" is available

ctx.getMap()

It's 'linked' to the substitution variables

Unfortunately the method "has" of maps does not seem to be available

So we'll have to loop through the entries to find one

```
for each (var substitutionVariableName in ctx.getMap().keySet()) {
  if (substitutionVariableName  === "WHAT_I_LOOK_FOR") {
    do_what_I_want_to_do_with_it();
  }
}
```

(Note that substitution variables are declared with uppercase names)

Setting the value of a substitution variable….

Use the ctx.getMap() "map" also to write new variables and new values

Just "put" the entry in the map.

```
ctx.getMap().put("MYSUBSTVAR", "The Value");
```

Soooooo……..

The Script…..

```
"use strict";

var paramCount       = args.length - 1;
var substVarSource   = null;
var substVarTarget   = null;
var substVarValue    = null;
var substVarFound    = false;

// Param 1: name of the substVar that needs to be defaulted if it does not exist
substVarSource     = args[1].toUpperCase();

// Check if the source variable exists
for each (var definedVarName in ctx.getMap().keySet()) {
  if (definedVarName === substVarSource) {
    substVarFound = true;
  }
}

// Param 2: value to be used as default for the substVar if it does not exist
if (substVarFound) {
  substVarValue = ctx.getMap().get(substVarSource);
} else {
  substVarValue = args[2];
}

// Param 3 (optional): name of a substVar to which the value should be written
if (paramCount === 3) {
  substVarTarget   = args[3].toUpperCase();
} else {
  substVarTarget   = substVarSource;
}

// Set the target variable to be the value we determined.
ctx.getMap().put(substVarTarget, substVarValue);
```

The working script……

All it takes: Just 34 lines.

Only 23 if we don't count empty lines and comments.

Nope!
It could be!!
But there's more……..

First: I made it a bit more robust/fancy

- Introduction of functions:
  - Modularization and smaller code blocks to test

- Parameter checks are built in:
  - Error message when less than 2 or more than 3 parameters supplied
  - Display syntax help when no parameters or parameter 'help' is supplied

So the script becomes

(Don't worry script files will be joined with the slides)

```
"use strict";

// Function writes text and an end-of-line to screen
function writeLine (line) {
  ctx.write (line + "\n");
}

// function displays an error message on screen
function errorMsg (message) {
  writeLine ("");
  writeLine ("#####################################################");
  writeLine ("== ERROR == ");
  writeLine (message);
  writeLine ("");
  writeLine ("Call this script with parameter 'help' for usage");
  writeLine ("#####################################################");
  writeLine ("");
}

// Prints the Help Text on screen
function displayHelp (scriptName) {
  writeLine ('                                                                              );
  writeLine ('==============================================================================');
  writeLine ('Usage of script ' + scriptName                                                );
  writeLine ('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~');
  writeLine (scriptName + ' help'                                                           );
  writeLine ('  => Display this help text'                                                  );
  writeLine ('                                                                              );
  writeLine ('>>>>> Version with 2 parameters <<<<<'                                        );
  writeLine (scriptName + ' SubstVar DefaultValue'                                          );
  writeLine ('                                                                              );
  writeLine ('  => If the substitution variable named in the first parameter exists it is');
  writeLine ('     left unchanged.'                                                         );
  writeLine ('     If the substitution variable named in the first parameter does NOT'      );
  writeLine ('     exist it is created and given the value of the second parameter'         );
  writeLine ('                                                                              );
```

Function to display error messages

Function to display usage notes

```
  writeLine ('>>>>> Version with 3 parameters <<<<<'                                  );
  writeLine (scriptName + ' SubstVar1 DefaultValue SubstVar2'                        );
  writeLine (''                                                                      );
  writeLine ('  => The substitution variable named in the third parameter is created if' );
  writeLine ('     it does not yet exist, and given either the value of the substitution' );
  writeLine ('     variable named in the first parameter if it exists, or the value of' );
  writeLine ('     the second parameter if the substitution variable named in the first' );
  writeLine ('     parameter does not exist.'                                      );
  writeLine (''                                                                      );
  writeLine ('First and Third parameter (variable names) are case-INsensistive'      );
  writeLine (''                                                                      );
  writeLine ('Examples:'                                                             );
  writeLine (scriptName + ' my_var my_default'                                       );
  writeLine ('   if variable &MY_VAR exists, nothing will happen'                    );
  writeLine ('   if variable &MY_VAR does not exist it will be created and given'    );
  writeLine ('   the vale "my_default"'                                              );
  writeLine (''                                                                      );
  writeLine (scriptName + ' my_var my_default target_var'                            );
  writeLine ('   if variable &MY_VAR exists, its value will be assigned to variable' );
  writeLine ('   &TARGET_VAR, which will be created if it does not yet exist'       );
  writeLine ('   if variable &MY_VAR does not exist "my_default" will be assigned to' );
  writeLine ('   variable &TARGET_VAR, which will be created if it does not yet exist' );
  writeLine ('=========================================================================');
  writeLine (''                                                                      );
}

// Function returns true or false indicating if a substitution variable with the indicated name exists
function substVarExists (substVarName) {
  var substVarFound    = false;

  for each (var definedVarName in ctx.getMap().keySet()) {
    if (definedVarName === substVarName) {
      substVarFound = true;
    }
  }

  return substVarFound;
}
```

Function to check existence of the substitution variable

The main code

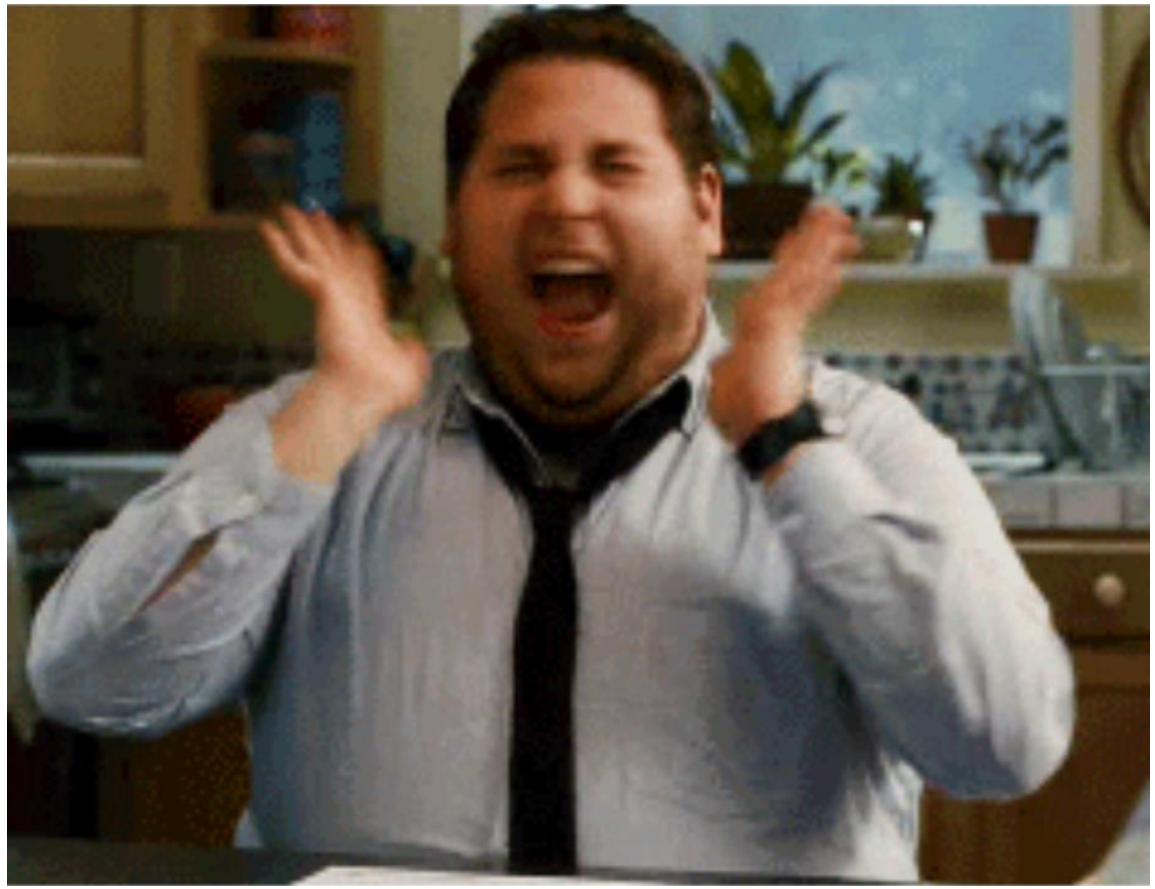
```
// Main code

var paramCount       = args.length - 1;
var substVarSource   = null;
var substVarTarget   = null;
var substVarValue    = null;

// Check and handle parameters
if (paramCount > 3) {

  // More than 3 parameters supplied: syntax error
  errorMsg (args[0] + " - Too many parameters (" + paramCount + ")");

} else if ((paramCount === 0) ||
           ((paramCount === 1) && (args[1].toLowerCase() === "help"))
          ) {

  // No parameters supplied, or 1 parameter which is "help": show help text
  displayHelp (args[0]);

} else if (paramCount < 2) {

  // Fewer than 2 parameters supplied: syntax error
  errorMsg (args[0] + " - Not enough parameters (" + paramCount + ")");

} else {

  // Either 2 or 3 parameters supplied
  // Param 1 containts the name of the substVar that needs to be defaulted if it does not exist
  substVarSource     =  args[1].toUpperCase();

  // Param 2 contains the value to be used as default for the substVar if it does not exist
  // If the source variable exists, use its value, else use the default value from param 2
  if (substVarExists (substVarSource)) {
    // The source variable exists, so instead of the provided default value, use its value for the target variable
    substVarValue = ctx.getMap().get(substVarSource);

  } else {

    substVarValue =  args[2];
```

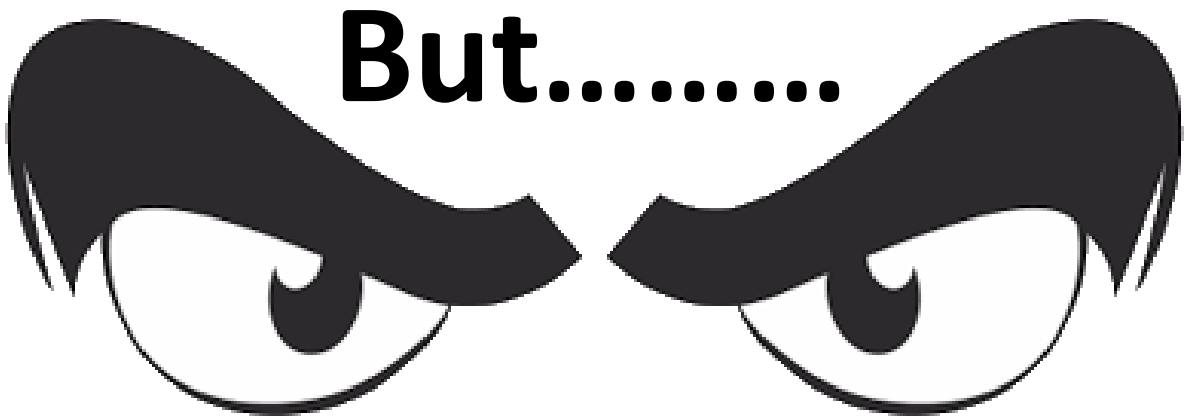
```
  }

  // Param 3 (optional) contains the name of a substVar in which the value should be placed
  //         if not supplied, it should be placed in the variable that is named in the first parameter
  if (paramCount === 3) {

    // 3 parameters supplied, so the target variable is named in the third parameter
    substVarTarget   =  args[3].toUpperCase();

  } else {

    // 2 parameters supplied, so the target variable is named in the first parameter
    substVarTarget   = substVarSource;

  }

  // Set the target variable to be the value we determined.
  ctx.getMap().put(substVarTarget, substVarValue);

}
```

Still only 91 lines
(excluding empty - and comment lines)

Of which about half to put nice help on screen

Awesome!!!!!





Does it work???

Run with 0 parameters:

script VariableDefault

```
ERO@EVROCS>script VariableDefault

================================================================================
Usage of script VariableDefault
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
VariableDefault help
  => Display this help text

>>>>> Version with 2 parameters <<<<<
VariableDefault SubstVar DefaultValue

  => If the substitution variable named in the first parameter exists it is
     left unchanged.
     If the substitution variable named in the first parameter does NOT
     exist it is created and given the value of the second parameter

>>>>> Version with 3 parameters <<<<<
VariableDefault SubstVar1 DefaultValue SubstVar2

  => The substitution variable named in the third parameter is created if
     it does not yet exist, and given either the value of the substitution
     variable named in the first parameter if it exists, or the value of
     the second parameter if the substitution variable named in the first
     parameter does not exist.

First and Third parameter (variable names) are case-INsensistive

Examples:
VariableDefault my_var my_default
   if variable &MY_VAR exists, nothing will happen
   if variable &MY_VAR does not exist it will be created and given
   the vale "my_default"

VariableDefault my_var my_default target_var
   if variable &MY_VAR exists, its value will be assigned to variable
   &TARGET_VAR, which will be created if it does not yet exist
   if variable &MY_VAR does not exist "my_default" will be assigned to
   variable &TARGET_VAR, which will be created if it does not yet exist
================================================================================
```

Run with 2 parameters:

script VariableDefault MyVar MyDefault

(If MyVar does not exist, create it with value 'MyDefault', otherwise leave it unchanged)

```
ERO@EVROCS>undefine MyVar
ERO@EVROCS>
ERO@EVROCS>define MyVar
SP2-0135: symbol myvar is UNDEFINED
ERO@EVROCS>
ERO@EVROCS>script VariableDefault MyVar MyDefault
ERO@EVROCS>
ERO@EVROCS>define MyVar
DEFINE MYVAR            = "MyDefault" (CHAR)
ERO@EVROCS>
ERO@EVROCS>define MyVar = Something Else
ERO@EVROCS>
ERO@EVROCS>define MyVar
DEFINE MYVAR            = "Something Else" (CHAR)
ERO@EVROCS>
ERO@EVROCS>script VariableDefault MyVar MyDefault
ERO@EVROCS>
ERO@EVROCS>define MyVar
DEFINE MYVAR            = "Something Else" (CHAR)
ERO@EVROCS>_
```

Run with 3 parameters:

script VariableDefault MyVar MyDefault MyNewVar

(Create variable MyNewVar and assign it the value
of MyVar if it exists,
or the value 'MyDefault' if MyVar doesn't exist)

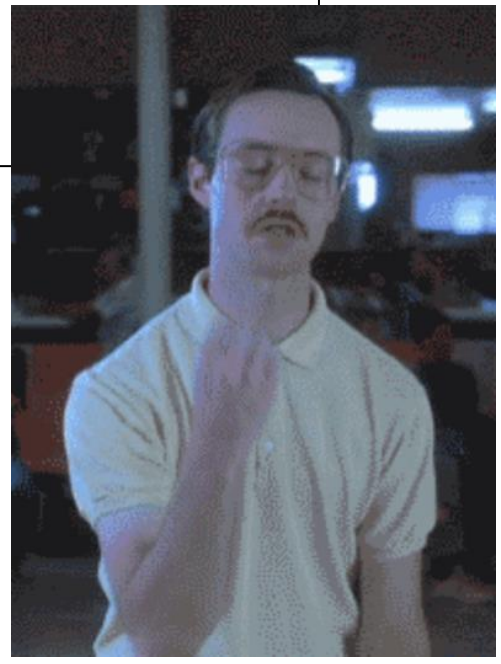
```
ERO@EVROCS>undefine MyVar
ERO@EVROCS>undefine MyNewVar
ERO@EVROCS>
ERO@EVROCS>define MyVar
SP2-0135: symbol myvar is UNDEFINED
ERO@EVROCS>define MyNewVar
SP2-0135: symbol mynewvar is UNDEFINED
ERO@EVROCS>
ERO@EVROCS>script VariableDefault MyVar MyDefault MyNewVar
ERO@EVROCS>
ERO@EVROCS>define MyVar
SP2-0135: symbol myvar is UNDEFINED
ERO@EVROCS>define MyNewVar
DEFINE MYNEWVAR        = "MyDefault" (CHAR)
ERO@EVROCS>
ERO@EVROCS>define MyVar = Something Else
ERO@EVROCS>
ERO@EVROCS>script VariableDefault MyVar MyDefault MyNewVar
ERO@EVROCS>
ERO@EVROCS>define MyVar
DEFINE MYVAR           = "Something Else" (CHAR)
ERO@EVROCS>define MyNewVar
DEFINE MYNEWVAR        = "Something Else" (CHAR)
ERO@EVROCS>_
```

So, are my sql-script **parameters** optional ?????

Change the sql script into:

```
script VariableDefault 1 %
script VariableDefault 2 %

select atc.table_name     "Table"
,      atc.column_name    "Column"
,      atc.data_type      "Data Type"
from   user_tab_columns     atc
where  atc.column_name like upper('&1')
  and  atc.data_type   like upper('&2')
order by 1, 2, 3
;

undef 1
undef 2
```

```
ERO@EVROCS>@SearchColumns.sql %from %char%

                 Table            Column     Data Type
______________________ _________________ ____________
ERR$_BULK_ERRORS_PERF  CUST_EFF_FROM     VARCHAR2
EXT_TAB_TEST           DATE_FROM         VARCHAR2
EXT_TAB_TEST           VALID_FROM        VARCHAR2

3 rows selected.
```



After 30 years of waiting for this……

```
ERO@EVROCS>@SearchColumns.sql %from

                 Table                          Column     Data Type
______________________ ______________________________ ____________
BULK_ERRORS_PERF       CUST_EFF_FROM                  DATE
ERO_TEST_PLSQL_TESTS   CONFIDENCE95_INTERVAL_FROM      NUMBER
ERR$_BULK_ERRORS_PERF  CUST_EFF_FROM                  VARCHAR2
EXT_TAB_TEST           DATE_FROM                      VARCHAR2
EXT_TAB_TEST           VALID_FROM                     VARCHAR2
TECH_EXPERIENCE_17_PERF CUST_EFF_FROM                 DATE

6 rows selected.
```

Happy?

Well, not completely ….

I wanted a command

```
VariableDefault 1 %
```

Not a sentence

```
script VariableDefault 1 %
```

Alias ???

For example

```
alias time=select to_char(sysdate,'hh24:mi:ss') now from dual;
```

```
ERO@EVROCS>alias time=select to_char(sysdate,'hh24:mi:ss') now from dual;
ERO@EVROCS>time

        NOW
___________
16:49:41
```

Also with binds as parameters

```
alias my_env=select sys_context('userenv',:B1) "Value" from dual;
```

```
ERO@EVROCS>alias my_env=select sys_context('userenv',:B1) "Value" from dual;
ERO@EVROCS>my_env sessionid

     Value
__________
7870119
```

So could we do….

```
alias variabledefault=script d:\Scripts\VariableDefault.js :1 :2 :3;
```

Unfortunately Alias variables are **not optional** ☹

```
ERO@EVROCS>my_env
Error: Alias with binds: not enough binds supplied at run time.
```

Solution: Custom Commands!!!

We can register our JavaScript as a new command in SQLcl

Custom Commands

Let's touch on the hard way first….

Custom Commands…

The code we have to add for this:

- Create a function expression which
  - Parses the command line for command and parameters
  - Determines if the command is one this code has to handle
  - Executes the code if it is
  - Signals other listeners whether the command has been handled
- Create a SQLcl Command Listener with that function expression

  Java Class:

  oracle.dbtools.raptor.newscriptrunner.CommandRegistry
- Register this listener with the SQLcl Command Registry

  Java Class:

  oracle.dbtools.raptor.newscriptrunner.CommandListener

Splitting the command line into arguments

```
function splitCommandLine (commandLine) {
  var posStart;
  var posEnd;
  var clArgs = [];

  commandLine = commandLine.trim();
  while (commandLine.length > 0) {
    if (commandLine.charAt(0) == '"') {
      // Starts with double quote, so argument runs from this double quote until the next
      // Do not include the double quotes
      posStart     = 1;
      posEnd       = commandLine.indexOf ('"', 1);

      // If no other double quote to match the first one is found, raise an error
      if (posEnd == -1) {
        errorMsg ("UNMATCHED DOUBLE QUOTE");
        return;
      }

      // No trim, because a parametervalue between double quotes has to be taken litterally
      clArgs[clArgs.length] = commandLine.substring(posStart, posEnd);

    } else {
      // Does not start with double quote, so argument runs from first character until the first space
      // If there is no space, then until the end of the string
      posStart     = 0;
      posEnd       = commandLine.indexOf (" ");

      if (posEnd == -1) {
        posEnd     = commandLine.length;
      }

      clArgs[clArgs.length] = commandLine.substring(posStart, posEnd).trim();
    }

    commandLine           = commandLine.substring(posEnd + 1).trim();
  }

  return clArgs
}
```

Function

- split command line in individual command and parameters
- Places each element in an array
- Returns the array
- Double quoted parameters (e.g. including spaces) are supported
- Leading and trailing spaces are discarded unless enclosed in double quotes

Custom command Registration

```
// *** ^^^ The code of our script is above this ^^^ ***
// mainCode();  moved to custom_handle

var CommandRegistry = Java.type("oracle.dbtools.raptor.newscriptrunner.CommandRegistry");
var CommandListener = Java.type("oracle.dbtools.raptor.newscriptrunner.CommandListener");

var customCmnd    = "variabledefault";
var custom_begin  = function (conn,ctx,cmd) {}
var custom_end    = function (conn,ctx,cmd) {}
var custom_handle = function (conn,ctx,cmd) {
                      if (cmd.getSql().toLowerCase() === customCmnd ||
                          cmd.getSql().toLowerCase().startsWith(customCmnd + " ")) {
                        var stmntArgs = splitCommandLine (cmd.getSql());
                        mainCode(stmntArgs);
                        return true;
                      }
                      return false;
                    }

var myCustomCommand = Java.extend(CommandListener, {
                                    handleEvent: custom_handle ,
                                    beginEvent:  custom_begin  ,
                                    endEvent:    custom_end
                                  }
                                 );

CommandRegistry.addForAllStmtsListener(myCustomCommand.class);
```

So this, plus the splitCommandLine function, is what we have to add to register the script as a custom command "variabledefault".

Put all this in a file "**VariableDefault_Cmd.js**"

And put

```
script d:\Scripts\VariableDefault_Cmd.js
```

In your login.sql script

And you will always have command **VariableDefault** at your disposal

Change the sql script into:

```
VariableDefault 1 %
VariableDefault 2 %

select atc.table_name     "Table"
,      atc.column_name    "Column"
,      atc.data_type      "Data Type"
from   user_tab_columns     atc
where  atc.column_name like upper('&1')
  and  atc.data_type   like upper('&2')
order by 1, 2, 3
;

undef 1
undef 2
```

```
ERO@EVROCS>@SearchColumns %from %char%

                 Table            Column    Data Type
______________________ _________________ ____________
ERR$_BULK_ERRORS_PERF    CUST_EFF_FROM     VARCHAR2
EXT_TAB_TEST             DATE_FROM         VARCHAR2
EXT_TAB_TEST             VALID_FROM        VARCHAR2

3 rows selected.

ERO@EVROCS>@SearchColumns %from

                   Table                          Column    Data Type
________________________ ______________________________ ____________
BULK_ERRORS_PERF         CUST_EFF_FROM                  DATE
ERO_TEST_PLSQL_TESTS     CONFIDENCE95_INTERVAL_FROM     NUMBER
ERR$_BULK_ERRORS_PERF    CUST_EFF_FROM                  VARCHAR2
EXT_TAB_TEST             DATE_FROM                      VARCHAR2
EXT_TAB_TEST             VALID_FROM                     VARCHAR2
TECH_EXPERIENCE_17_PERF  CUST_EFF_FROM                  DATE

6 rows selected.
```

Unregistering a custom command

Currently the command is simply added

So what if we run the registering script again?

- On new login in same SQLcl session (login.sql)
- Or we run the script with changed code

An extra listener gets added again

And again

And again

And again

You can't run changed code (without restarting SQLcl)
Because the old version will run and
report the command handled

So we need to make SQLcl 'forget' the old version

Please welcome……

Philipp Salvisberg

Came up with a neat trick to unregister a command

See:
https://github.com/Trivadis/plsql-formatter-settings/blob/main/sqlcl/format.js

Steps for unregistering

1. Make sure we can identify a listener by the name of the command
   On registration override **toString** method of listener: return the command

2. Create an unregister function that will
   1. Retrieve a list of current listeners
   2. Run a command that removes all listeners
   3. Retrieve a list of listeners that could not be removed (if any)
   4. Re-register all the listeners, except
      - The one we want to unregister
      - The ones that could not be removed

3. Execute the unregister function just before registering a custom command

The unregister function

```
function unregisterCommand () {
  var Collectors      = Java.type("java.util.stream.Collectors");
  var SQLCommand      = Java.type("oracle.dbtools.raptor.newscriptrunner.SQLCommand");
  var CommandRegistry = Java.type("oracle.dbtools.raptor.newscriptrunner.CommandRegistry");

  var listeners = CommandRegistry.getListeners (ctx.getBaseConnection(), ctx)
                  .get(SQLCommand.StmtSubType.G_S_FORALLSTMTS_STMTSUBTYPE);

  CommandRegistry.removeListener(SQLCommand.StmtSubType.G_S_FORALLSTMTS_STMTSUBTYPE);
  CommandRegistry.clearCaches(null, ctx);
  CommandRegistry.clearCaches(ctx.getBaseConnection(), ctx);

  var remainingListeners = CommandRegistry.getListeners(ctx.getBaseConnection(), ctx)
                           .get(SQLCommand.StmtSubType.G_S_FORALLSTMTS_STMTSUBTYPE)
                           .stream().map(function(l) l.getClass())
                           .collect(Collectors.toSet());

  for (var i in listeners) {
      if (!listeners.get(i).toString().equals("VariableDefault") &&
          !remainingListeners.contains(listeners.get(i).getClass())) {
          CommandRegistry.addForAllStmtsListener(listeners.get(i).getClass());
      }
  }
}
```

Thank you,
Philipp Salvisberg

But hold on….?!?

To properly register this simple one-liner as a custom command

```
ctx.write("First parameter is: " + args[1] + "\n");
```

I'd have to write **80 lines of code**

Excluding empty lines
Excluding comments


```
function splitCommandLine (commandLine) {
  var posStart;
  var posEnd;
  var clArgs = [];

  commandLine = commandLine.trim();
  while (commandLine.length > 0) {
    if (commandLine.charAt(0) == '"') {
      // Starts with double quote, so argument runs from this double quote until the next
      // Do not include the double quotes
      posStart    = 1;
      posEnd      = commandLine.indexOf ('"', 1);

      // If no other double quote to match the first one is found, raise an error
      if (posEnd == -1) {
        ctx.write ("== ERROR == " + "\n");
        ctx.write ("UNMATCHED DOUBLE QUOTE" + "\n");
        return;
      }

      // No trim, because a parametervalue between double quotes has to be taken litterally
      clArgs[clArgs.length] = commandLine.substring(posStart, posEnd);

    } else {
      // Does not start with double quote, so argument runs foreom first character until the first space
      // If there is no space, then until the end of the string
      posStart    = 0;
      posEnd      = commandLine.indexOf (" ");

      if (posEnd == -1) {
        posEnd    = commandLine.length;
      }

      clArgs[clArgs.length] = commandLine.substring(posStart, posEnd).trim();
    }

    commandLine           = commandLine.substring(posEnd + 1).trim();
  }

  return clArgs
}

function unregisterCommand (commandToUnregister) {
  var Collectors      = Java.type("java.util.stream.Collectors");
  var SQLCommand      = Java.type("oracle.dbtools.raptor.newscriptrunner.SQLCommand");
  var CommandRegistry = Java.type("oracle.dbtools.raptor.newscriptrunner.CommandRegistry");

  // Get the current list of listeners
  var listeners = CommandRegistry.getListeners (ctx.getBaseConnection(), ctx)
                                 .get(SQLCommand.StmtSubType.G_S_FORALLSTMTS_STMTSUBTYPE);

  // remove all commands registered with CommandRegistry.addForAllStmtsListener
  CommandRegistry.removeListener(SQLCommand.StmtSubType.G_S_FORALLSTMTS_STMTSUBTYPE);

  // clear the general (no-connection) cache and the connection specific cache
  CommandRegistry.clearCaches(null, ctx);
  CommandRegistry.clearCaches(ctx.getBaseConnection(), ctx);

  // get the list of listeners that have not been removed, if any
  var remainingListeners = CommandRegistry.getListeners(ctx.getBaseConnection(), ctx)
                           .get(SQLCommand.StmtSubType.G_S_FORALLSTMTS_STMTSUBTYPE)
                           .stream().map(function(i) i.getClass())
                           .collect(Collectors.toSet());

  // re-register all commands except the command-to-be-unregistered and anything that hasn not been removed
  for (var i in listeners) {
      if (!listeners.get(i).toString().equals(commandToUnregister) &&
          !remainingListeners.contains(listeners.get(i).getClass())) {
          CommandRegistry.addForAllStmtsListener(listeners.get(i).getClass());
      }
  }
}

function mainCode (scriptArguments) {
  ctx.write("First parameter is: " + scriptArguments[1] + "\n");
}

function registerCommand () {
  var commandName = "simpleOneLiner";

  // a variable referencing the Command Registry of SQLCL
  var CommandRegistry = Java.type("oracle.dbtools.raptor.newscriptrunner.CommandRegistry");

  // a variable referencing the CommandListener
  var CommandListener =  Java.type("oracle.dbtools.raptor.newscriptrunner.CommandListener");

  // a Function expression for the code to handle the command

  var customCmnd    = "variabledefault";
  var custom_handle = function (conn,ctx,cmd) {
                        // if first argument is the command we need to handle here, then call main code
                       if (cmd.getSql().toLowerCase() === customCmnd || cmd.getSql().toLowerCase().startsWith(customCmnd + " ")) {
                         var stmntArgs = splitCommandLine (cmd.getSql());  // split command line in array of arguments

                         mainCode (stmntArgs);

                         return true;  //exit the function and indicate the command has been handled
                       }

                      return false;  //exit the function and indicate the command has not been handled
                    }
  // a Function expression for the code to run before the command is executed
  var custom_begin = function (conn,ctx,cmd) {
                     }

  // a Function expression for the code to run after the command is executed
  var custom_end = function (conn,ctx,cmd) {
                   }

  // a Function expression that will override the toString method of the Listener
  // The function returns the command name
  var toString = function() {
                   return commandName;
                 }

  // Extend the CommandListener
  var myCustomCommand = Java.extend(CommandListener, {
                                    handleEvent: custom_handle ,
                                    beginEvent:  custom_begin  ,
                                    endEvent:    custom_end    ,
                                    toString:    toString
                                  }
                                 );

  // Register the command
  unregisterCommand (commandName);
  CommandRegistry.addForAllStmtsListener(myCustomCommand.class);
}

//Register the command
registerCommand();
```

Library to the rescue

See: https://github.com/erikvanroon/SQLcl
https://www.evrocs.nl/your-library-is-your-paradise/

```
"use strict";

// Load Library
var libraryPath = java.lang.System.getenv("SQLCL_JS_LIB")
          .replace(/\\/g, "/").replace(/\/?$/, "/");

load (libraryPath + "ELib_registration.js");

function scriptCode () {
  [... ALL THE CODE OF THE ORIGINAL SCRIPT ...]
}

// Execute or register the main code
registration.runCommand (scriptCode);
```

(optional) Get the location of your libraries from an environment variable

Load the library with hardcoded path or use an environment variable

Enclose all original code in a function.
Except "use strict" and any library loading

Pass the function expression (so without brackets) to the runCommand function

All it needs to take is 3 or 4 extra lines of code

Result - 1

You can still use the script as a regular script, without registering

```
ERO@EVROCS>prompt Contents of variable MyTest = &MyTest
Enter value for MyTest:
Contents of variable MyTest =
ERO@EVROCS>
ERO@EVROCS>script VariableDefault.js MyTest MyDefault
ERO@EVROCS>
ERO@EVROCS>prompt Contents of variable MyTest = &MyTest
Contents of variable MyTest = MyDefault
ERO@EVROCS>
ERO@EVROCS>script VariableDefault.js MyTest SecondDefault
ERO@EVROCS>
ERO@EVROCS>prompt Contents of variable MyTest = &MyTest
Contents of variable MyTest = MyDefault
ERO@EVROCS>_
```

Result - 2

You can register it as a custom command using special parameters

```
ERO@EVROCS>script VariableDefault.js -cmdReg varCmd -minimal
Command varCmd has been registered
ERO@EVROCS>_
```

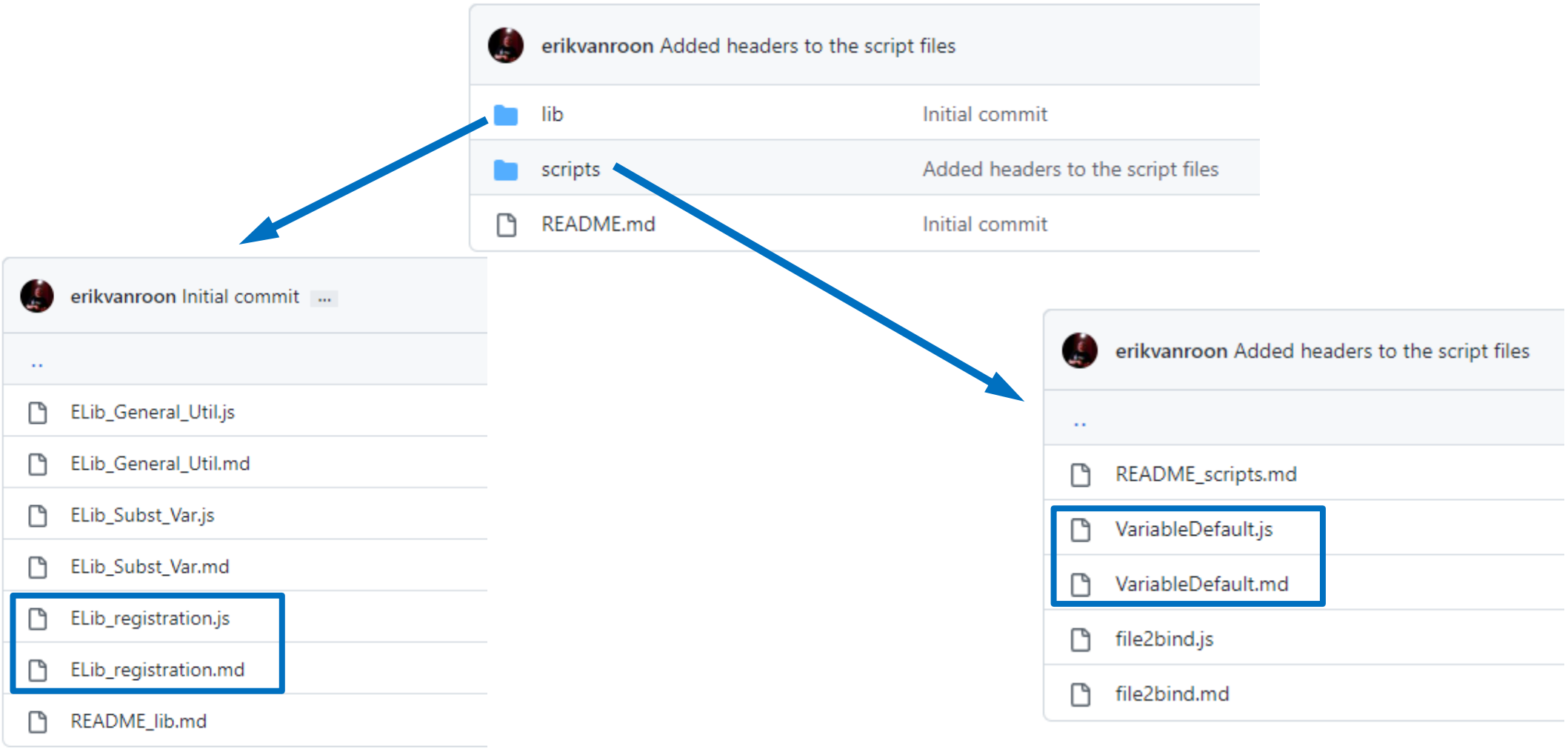
And then the command can be used

```
ERO@EVROCS>prompt Contents of variable MyTest = &MyTest
Enter value for MyTest:
Contents of variable MyTest =
ERO@EVROCS>
ERO@EVROCS>varCmd MyTest MyDefault
ERO@EVROCS>
ERO@EVROCS>prompt Contents of variable MyTest = &MyTest
Contents of variable MyTest = MyDefault
ERO@EVROCS>
ERO@EVROCS>varCmd  MyTest SecondDefault
ERO@EVROCS>
ERO@EVROCS>prompt Contents of variable MyTest = &MyTest
Contents of variable MyTest = MyDefault
ERO@EVROCS>_
```

Downloads

Latest versions can be found on GitHub

https://github.com/erikvanroon/SQLcl



"Stupid questions do exist.
But it takes a lot more time and energy to correct a stupid mistake than it takes to answer a stupid question, so please ask your stupid questions."

a wise teacher who taught me more than just physics

Thanks !!