REXX on NodeJS

Disclaimer

This SOFTWARE PRODUCT is provided by El Condor "as is" and "with all faults." El Condor makes no representations or warranties of any kind concerning the safety, suitability, lack of viruses, inaccuracies, typographical errors, or other harmful components of this SOFTWARE PRODUCT. There are inherent dangers in the use of any software, and you are solely responsible for determining whether this SOFTWARE PRODUCT is compatible with your equipment and other software installed on your equipment. You are also solely responsible for the protection of your equipment and backup of your data, and El Condor will not be liable for any damages you may suffer in connection with using, modifying, or distributing this SOFTWARE PRODUCT.

You can use this **SOFTWARE PRODUCT freely, if you would you can credit me in program comment:**

El Condor – CONDOR INFORMATIQUE – Turin

Comments, suggestions and criticisms are welcomed: mail to rossati@libero.it

Conventions

Commands syntax, instructions in programming language and examples are with font `COURIER NEW`. The optional parties of syntactic explanation are contained between `[square parentheses]`, alternatives are separated by `|` and the variable parties are in *`italics`*.

Contents table

Introduction

In a CodeProject Article[1] I described how to use PHP scripts in a NodeJS Web server, in this article I deal with the use of REXX scripts, but I show only the few JavaScript instructions needed to manage them while the main purpose is the analysis of REXX and problems that may arise from its use.

A short overview of REXX (and ooREXX)

REXX (Restructured Extended Executor) is an interpreted programming language developed at IBM by Mike Cowlishaw in 1979. It is a structured, high-level programming language designed to ease of learning and reading. REXX interpreters and compilers exist for a wide range of computing platforms, proprietary and open source; REXX is used as a scripting and macro language, mostly for processing data and text and for generating reports; it has evolved in the years and in 1996 ooREXX, a version with the support of the objects, has been released. REXX can works in Common Gateway Interface (CGI) programming and also embedded on HTML pages (like for example PHP).

Some points about REXX

All what is not recognized as statement is sent to the Operating System; this is due to the purpose for which REXX was created, namely the management of the various jobs carried out on mainframes; this also explains why the concatenation of strings occurs by juxtaposition with an inserted space (the operator || avoid this). Although in the object version of REXX there are complex data structures such as arrays, lists, queues, etc. these, in the original REXX, it can be built using a particular type of variable named `stem`, i.e. a variable identified by a period behind the name. After the period, one can add one or many variables (separated by period) obtaining arrays or complex structure. If the variable is numeric stem is an array otherwise it is a hash table[2].
Below an example to verify if the RANDOM function passes the chi square test.

```
dice. = 0   -- set array defaults value
loop i = 1 while i < 61
      a = RANDOM(1,6)
      dice.a = dice.a + 1
end
loop i = 1 while i < 7
      dice.7 = dice.7 + (dice.i - 10) ** 2
end
say dice.7 / 10   -- if value < 11.07 Ok (test square chi with 5 degrees of freedom)
```

Object can act by calling object's actions, namely by sending messages where the "message send" symbol is ~ called twiddle. Objects respond to these messages by performing an action and possibly returning data or an object.

1 See NodeJS WEB Server running PHP.
2 JavaScript arrays (1995) are very similar to REXX stems (1979).

Changes to the web server

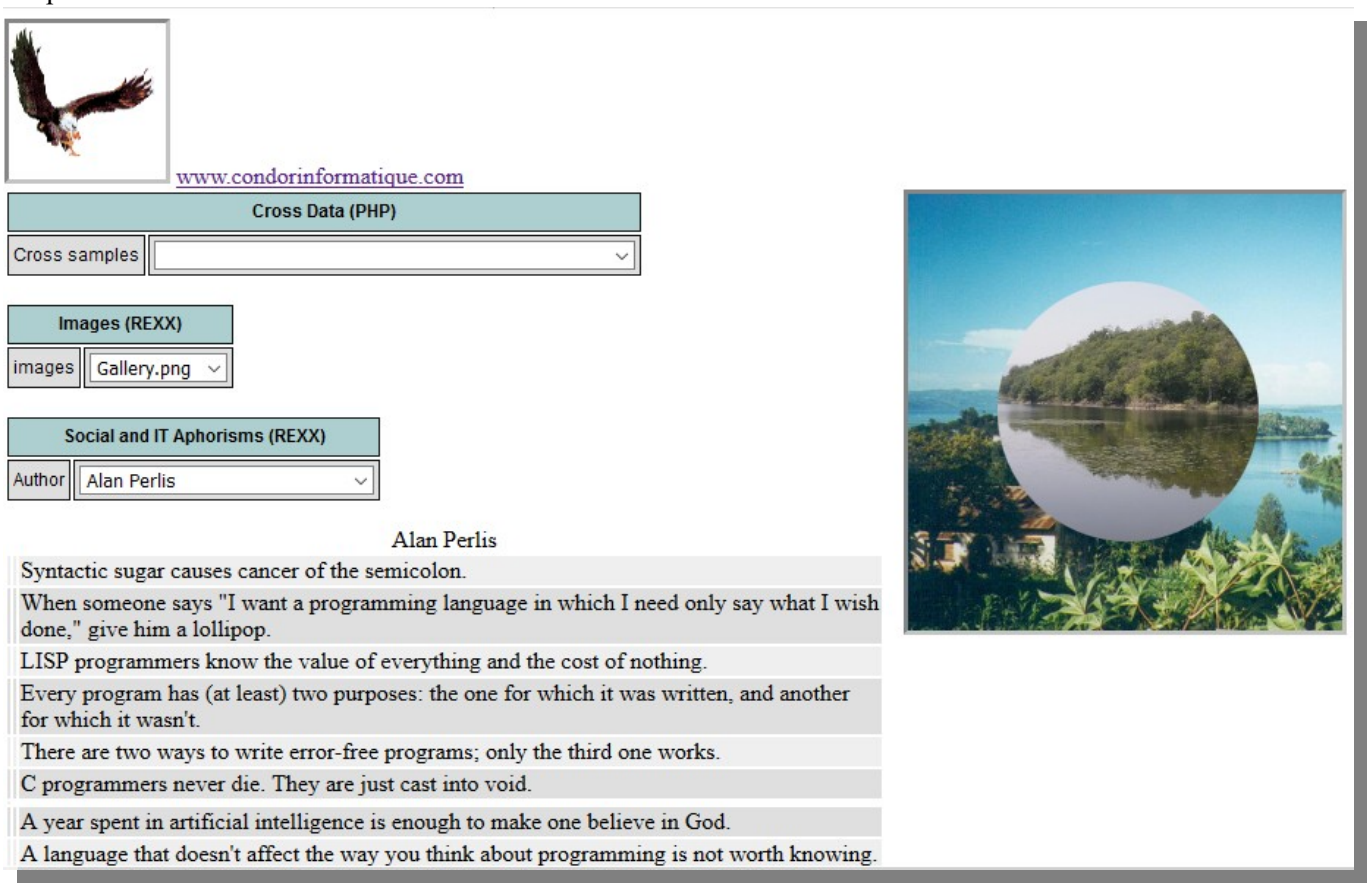
In the WEB Server script `tryNodeJS.js` described in the aforementioned Article the REXX script management has been added:

```
} else if (type == "rex") {
      var r = [__basedir + pathName];
      // prepare fields for PHP
      for(q in parms) {r[r.length] = q; r[r.length] = parms[q].replace(/ /g,"+");}
            console.log(__basedir + pathName + r)
            var spawnRex = spawn('rexx',r);
            spawnRex.stdout.on('data',function (data) {
                  response.writeHead(200, {'Content-Type': cType});
                  response.end(data);
            });
            spawnRex.stderr.on('data',function (data) {
                  console.log("Orror: " + data);
            });
} else {
```

In the above fragment the parameters for the REXX interpreter are a set of couples key value taken from the form, for example a request is analogous to: rexx getdata.rex Type Cite Author Adam+Smith.

The NodeJS demo

The demo is simply a page where one can choose to show an aphorism or an image (served by a REXX script questioning a SQLite Data Base) and a choice of some crossing data in SQLite Data Base managed by a PHP script[3].



3 See Cross of fields in a table of Data Base accessible via PDO.

The REXX script

The REXX interpreter is started with the syntax `rexx REXXscript parm1 ...`. The script deals with requests on list of authors of aphorisms, aphorisms of an author and a list of images.
Below are analyzed the main points of the script whose complete source is attached.

Recovery of the requests parameters

```
parms = arg(1)    -- arg(1) contains all parameters separated by space
do while parms \= ""
      parse var parms key value parms
      value = changestr("+",value," ")    -- restore spaces
      dizParms.key = value
end
```

`dizParms.` is a REXX `stem` used for storing key-value object; `parse var` is a statement that split `parms`, a string of parameters separated by space, into `key` and `value` (the possible remaining of the string is recovered in `parms`).

Analysis of requests

This is accomplished by the SELECT instruction that has the following structure:

```
SELECT
 When Condition Then
  Statement
 When Condition Then Do
  Statements
 End  Otherwise
 ...
End
```

In the `index.html` the combo boxes are built by a form generator[4] that calls, via Ajax, a list of the combo box items:

```
When dizParms.key == "Images" Then do  f = .File~new(curdir || "\images")~list~sortwith(.caselesscomparator~new)  myRE = .RegularExpression~new("?*(.gif|.png|.jpg)")  images = .array~of()  loop k over f   if myRE~match(k~lower) == 1 Then images~append(k)  end k  say images~toString(,",")  exit
end
```

This is enough cumbersome, but it is necessary for taking only the desired files; without this constraint it would be simply (so to speak): `say .File~new(curdir || "\images")~list~sortwith(.caselesscomparator~new)~toString(,",").`

For a request of citation the SQLite engine is started with a Data Base name and a SQL SELECT statement asking the fields *dove* (where), *Data* (Date) and *Citazione* (Quote); the answer is similar to:
The Wealth of Nations Book V Chapter I|1776|Wherever there is great property, there is great inequality. .

```
When dizParms.key == "Cite" Then do  table = "Citazioni"  authorGroup = "Author_Group"  Author = "Author"  if dizParms.authorGroup == "IT Aphorisms" Then table = "InforCitazioni"  sql = "SELECT Dove,Data,Citazione FROM " || table || " WHERE Autore = '" || dizParms.Author || "'"
 'sqlite3 condor.sqlite "' sql '"> ' || flname  response = "<table><caption>" || dizParms.Author || "</caption>" -- response string  do while lines(flname) > 0   line = linein(flname)   response = response || "<tr><td>" || Changestr("|",line,"</td><td>") || "</td></tr>"  end  say response||"</table>" end
```

4 A JavaScript Form Generator

Management of temporary file

```
signal on notready -- handle file errors curdir = directory() flname = SysTempFileName(curdir||"\tempfile.???")
...
CALL STREAM flname, 'C', 'CLOSE'
"del" flname
...
```

Conclusions

REXX can be easily used in NodeJS taking into account some (small) limitations among which:

- Regular Expression mast be activated including a special class: `::requires "rxregexp.cls"`; the functionalities are a rather limited subset and not entirely compatible with those present in JavaScript or PHP.
- The possibly space(s) in parameters must be "masked": `r[r.length] = parms[q].replace(/ /g,"+")` . Another obstacle is the difficulty, which I have not solved, to pass parameters with special characters (like Author Dom Hélder Câmara).
- Databases are supported by REXX/SQL a set of external functions by deal with Oracle, DB2, MySQL, SQLite and ODBC data-sources.