Tutorial

L'entête de script Pour fonctionner chaque script Rebol doit avoir une entête. Le mininum est :

```
REBOL [ ]
```

Cette entête peut servir aussi à fournir certaines informations comme l'auteur du script, la date de création, le numéro de version, le rôle du script, le nom du client, des commentaires ... en résumé, toutes les informations permettant de mieux cerner le script. Voici un exemple :

```
REBOL [
version: 1.1.2
auteur: "rebolzone"
date: 02-11-2002
role: "Rapport de fréquentation de la base ecoDB"
commentaire: { "Prend en compte l'indice 1. Remplace stateco1.r
et active le prochain."}
]
```

Remarques : on peut récupérer les infos de l'entête précédente :

```
print system/script/header/auteur ;Affiche l'auteur
print system/script/header/version ;Affiche le numéro de version
```

On peut inclure également du texte libre (préface) avant l'entête Rebol, dans ce cas, celui-ci est simplement ignoré par l'evaluateur Rebol.
Cela peut servir à ajouter des commentaires.

```
Voici un texte ignoré
rebol[
Version: 1.1.2
auteur: "rebolzone"
date: 02-11-2002
role: "Rapport de fréquentation de la base ecoDB"
commentaire: { "Prend en compte l'indice 1. Remplace stateco1.r
et active le prochain."}
]
;début de script
```

Les commentaires On peut ajouter des commentaires dans le script avec le caractère **;** (point-virgule). Des commentaires multilignes peuvent également être ajoutés avec **comment { commentaires}**

```
rebol[
Version: 1.1.2
auteur: "rebolzone"
date: 02-11-2002
role: "Rapport de fréquentation de la base ecoDB"
commentaire: { "Prend en compte l'indice 1. Remplace stateco1.r
et active le prochain."}
]
;début de script : ceci est un commentaire
var: 12345
comment { ceci est
un commentaire
multiligne}
print var
input
```

Les variables La déclaration d'une variable en Rebol se fait très simplement :

```
Total: 3258.22
print Total
3258.22
print type? Total ;Affichage du type de la variable decimal
```

Rebol comprend 45 types différents (date, integer, decimal, email, url, file, image, logic, time, function, object, string, money, binary...). La commande help datatype! (sous la console) permet d'afficher la liste des types.
On peut également déclarer une variable avec un constructeur

```
prochainRdv: make date! 25-04-2003
maxConnexion: make integer! 5400
fileLog: make file! %/c/infos/log/app.log
```

Attention, les variables ne sont pas typées strictement, si la variable Total est de type Decimal, on peut lui affecter la valeur "Hello", son type sera alors String.
Les noms de variables ne sont pas sensibles à la casse, la variable Total peut s'écrire TOTAL ou total ou TOTal ...
On peut utiliser les fonctions de conversion de types :

```
dateTraite: to-string prochainRdv ; la variable dateTraite est de type String
```

Les structures de contrôle Rebol comporte plusieurs instructions :

- either (condition) [instructions si condition vraie] [instructions si condition fausse]

```
either (total > 1000) [print "j'en ai plus de 1000"] [print "j'en ai au ma
```

- if (condition) [instructions si condition vraie]

```
if (total > 1000) [print "j'en ai plus de 1000"]
```

- if/else (condition) [instructions si condition vraie] [instructions si condition fausse]

```
if/else (total > 1000) [print "j'en ai plus de 1000"] [print "j'en ai au ma
```

- switch/default (valeur) [valeur1 [instructions] valeur2 [instructions] valeurN [instructions]] [instructions si defaut]

```
switch/default nombreErr [
          1 [print "1 erreur"]
          2 [print "2 erreurs"]
          3 [print "3 erreurs, attention" ]
          ]
     [ print "trop d'erreur !!! "]
```

Les structures itératives

- for variable valeurDebut valeurFin increment [Instructions]

```
for i 1 100 2 [ print i ] ; Affiche tous les nombres impairs compris entr
```

- loop nombre [Instructions]

```
loop 27 [print "hello"] ; Affiche 27 fois hello
```

- repeat valeurMaxi [Instructions]

```
repeat ind 15 [ print ind] ; Affiche les nombres de 1 à 15
```

- while [condition] [Instructions]

```
while [true] [
i: i + 1
print i
if (i > 10) [ break ] ; si i est > à 10 on sort de la boucle infinie
]
```

- until [Instructions]

```
until [
        i: i + 1
        print i
        greater? i 10
        ]
```

Les chaines

- La suppression des espaces (en début et en fin) d'une chaine se fait par **trim**.
  trim/head supprime les espaces en début de chaine, alors que trim/tail supprime en fin de chaine. Attention trim modifie la chaine originale.

```
chaine: " bonjour, Rebol "
print trim chaine ; renvoi bonjour, Rebol
print chaine ; chaine vaut "bonjour, Rebol"
```

- La convertion majuscules/minuscules se fait par **lowercase** et **uppercase**.
  L'option /part permet de préciser la longueur à convertir.

```
personne: "mr dupont"
print uppercase /part personne 4 ; renvoi MR Dupont : 4 caractères de cor
print personne ; renvoi MR Dupont , la chaine à été modifiée.
```

- Remplacement d'une sous-chaine dans une chaine de caractères : **replace**

```
chaine: "cette maison est une belle maison"
replace chaine "maison" "villa" ; chaine vaut maintenant "cette villa est
```

  L'option /all permet de remplacer plusieurs occurences.
- Longueur d'une chaine de caractères : **length?**

```
var: "bonjour"
print length? var ;renvoi 7
```

- La concaténation : **join**

```
var1: "bonjour"
var2: "Rebol"
var3: join var1 var2 ; var3 vaut "bonjour Rebol"
```

- La concaténation : en utilisant un block

```
var1: "Il y a "
var2: "facture(s) en erreur"
nbfacterr: 53
resultat: [ var1 nbfacterr var2 ] ;resultat est de type Block
print resultat ; affiche Il y a 53 facture(s) en erreur
```

- Tester si une variable est de type string : **string?**

```
var1: 123.58
var2: "bonjour"
print string? var1 ; renvoi false
print string? var2 ; renvoi true
```

- Compression / décompression : **compress** et **decompress**

```
    var1: {Les envahisseurs. Ces êtres étranges
venus d'une autre planète. Leur destination: la Terre. Leur but:
en faire leur univers. David Vincent les a vus. Pour lui, tout a commencé
par une nuit sombre, le long d'une route solitaire de campagne, alors qu'il
cherchait un raccourci qu'il ne trouva jamais. Cela a commencé par
une auberge abandonnée, et par un homme que le manque de sommeil
avait rendu trop las pour continuer sa route. Cela a commencé par
l'atterrissage d'un vaisseau venu d'une autre galaxie. Maintenant, David
Vincent sait que les Envahisseurs sont là, qu'ils ont pris forme
humaine, et qu'il lui faut convaincre un monde incrédule que le cauchemar
à déjà commencé.}

    varcomp: compress var1 ; compression
    print length? var1 ; 677
    print length? varcomp ; renvoi 379
    vardecomp: decompress varcomp ; décompression
    print length? vardecomp ; 677
```

- Conversion caractère code Ascii :

```
car: to-char 65 ; car vaut A et est de type Char
codeAsc: to-integer #"A" ; codeAsc vaut 65
```

- Extraction d'une sous-chaine : **copy/part at chaine début longeur**

```
var: "Bond, my name is Bond"
var2: copy/part at var 7 10
print var2 ;renvoi my name is
```

Les opérateurs de comparaison

Les opérateurs de comparaison disponibles en Rebol :

Opérateur	Signification	Commentaires
=	Egal à	print "HEllo" = "hello" ; renvoi True print 2 = 2.00 ; renvoi True
>	Strictement supérieur	
>=	Supérieur ou égal	
<	Strictement inférieur	
<=	Inférieur ou égal	
<>	Différent	
==	Strictement égal	print "HEllo" == "hello" ; renvoi False print 2 == 2.00 ; renvoi False, car les types différents
equal?	Egal à	
greater?	Strictement supérieur	
greater-or-equal?	Supérieur ou égal	
lesser?	Strictement inférieur	
lesser-or-equal?	Inférieur ou égal	
not-equal?	Différent	
strict-equal?	Strictement égal	

Les opérateurs

Opérateurs	Signification	Commentaires
+	Addition	
-	Soustraction	

*	Multiplication	
/	Division	
add	Addition	print add 2 3 ; renvoi 5 !!!!
substract	Soustraction	
multiply	Multiplication	
divide	Division	
absolute ou abs	Valeur absolue	print absolute -2 ; renvoi 2
negate	Change le signe	print negate 7 ; renvoi -7
//	Reste de la division entière	print // 20 7 ; renvoi 6
remainder	Reste de la division entière	print remainder 10 3 ; renvoi 1
**	Elvation à la puissance	print 10 ** 3 ; renvoi 1000
power	Elévation à la puissance	print power 2 4 ; renvoi 16
square-root	Racine carrée	print square-root 9 ; renvoi 3
odd?	Renvoi vrai si valeur impaire	print odd? 11 ; renvoi Vrai
even?	Renvoi vrai si valeur paire	print even? 11 ; renvoi Faux
positive?	Renvoi vrai si valeur positive	
negative?	Renvoi vrai si valeur negative	
min ou minimum	Renvoi la plus petite valeur	print min 45 6 ; renvoi 6
max ou maximum	Renvoi la plus grande valeur	print max 74 128 ; renvoi 128 print max "AZ" "AB" ; renvoi "AZ"
zero?	Renvoi vrai si valeur égale à Zéro	