The REBOL Documentation Project

-- FR - Documentation REBOL - Divers --

Divers

Des fonctions "fragiles"

Philippe Le Goff

Première publication : 8 juin 2006, et mis en ligne le jeudi 8 juin 2006

Résumé :

Traduction de l'article 25 du blog REBOl 3.0 sur les modules et les risques liés au fonctionnement des fonctions.

Des fonctions "fragiles"

NdT. Il aurait fallu traduire : des fonctions "percées" ou "trouées". Je n'aimais pas trop. J'ai préféré utiliser le terme "fragiles" qui rend mieux l'esprit du post.

Carl Sassenrath, CTO REBOL Technologies 8-May-2006 23:46 GMT Article #0025

Par défaut, toutes les fonctions définies par un utilisateur renvoient la dernière valeur évaluée. Par exemple, vous pouvez écrire :

```
add3 : func [a b c] [a + b + c]
```

C'est un raccourci pour :

```
add3 : func [a b c] [return a + b + c]
```

(Et ne pas utiliser return, c'est aussi plus rapide)

Cependant, REBOL 3.0 va ajouter des modules, et les modules auront la possibilité de masquer leurs mots et leurs données internes. Ceci est fait pour des raisons de sécurité, mais aussi pour permettre d'améliorer la capacité d'abstraction lors de la programmation. Par exemple, vous pourrez utiliser un module qui traitent les mots de passe (que vous ne voulez pas montrer ou voir atteindre).

Le problème est que si un programmeur ne fait pas attention, le fonctionnement standard de return ci-dessus pourra créer des fonctions "fragiles", qui peuvent accidentellement renvoyer des données internes importantes.

Voici une simple illustration dans le cas d'un module :

```
passwords : [] ; une liste privée de passwords

add-password : func [new-pass] [ append passwords new-pass ]
```

La fonction *add-passwords* est exportée, et la liste *passwords* ne l'est pas. Par malheur, le programmeur n'a pas fait attention, et la fonction *add-passwords* se termine en retournant la liste entière de mots de passe comme résultat ! Pas super.

Une solution simple consisterait en :

```
add-password : func [new-pass] [ append passwords new-pass none ; ou, vous pouvez utiliser "exit" aussi ]
```

mais, certains programmeurs devront se souvenir (ou savoir) qu'il faut faire ainsi.

Donc, nous nous demandons : quelle est la bonne solution à ce probléme ? Devons-nous renforcer le modèle pour un fonctionnement spécifique (ex. les blocs des fonctions devraient toujours utiliser return ou exit) ou devons-nous développer une sorte d'attribut optionnel (peut-être à l'intérieur de la spécification du module) pour essayer d'éviter cette situation ?

Bien sûr, nous ne pouvons changer le fonctionnement par défaut de return (comme montré dans le premier exemple), car cela impacterait trop de code. Mais, je pense que nous avons d'autres options, et je suis certain que certaines n'ont pas été envisagées pour l'instant. Faites-moi entendre vos idées là-dessus.

(Traduction : Philippe Le Goff)