ActiveX

Propriétés

className	String	"ActiveX"
members	Object	Une table hash qui décrit les méthodes et propriétés de l'objet COM. Utilisez Use `for (var x in obj.members) {writeln(x,'=',obj.members[x]);}` pour en voir la liste ainsi que la documentation COM délivrée.

Méthodes

ActiveX(id String, forceNew Any)

id CLSID ou désignation de l'objet à partir de la base de registres

forceNew CLSID ou désignation de l'objet à partir de la base de registres

Le constructeur essaie de se connecter à l'objet actif de la classe COM. S'il n'y a pas d'objet actif, il essaie d'en créer un avec CoCreateInstance(). Le constructeur lève une exception si la classe COM n'implémente pas l'interface IDispatch. Si vous utilisez un second paramètre (de n'importe quelle valeur), un nouvel objet COM est toujours créé.

close( )

Ferme l'objet COM et libère les bibliothèques ActiveX inusitées.

get(name String) returns Object

Si l'ID d'une propriété est inférieure à 255, elle apparaîtra automatiquement comme une propriété de l'objet JS correspondant. A défaut, utilisez get() et set() pour lire et modifier la valeur de cette propriété. Les types VARIANT reconnus sont BOOLEAN, INT4, DOUBLE, NULL, BSTR, IUnknown, and IDispatch.

set(name String, value Any) returns Object

Si l'ID d'une propriété est inférieure à 255, elle apparaîtra automatiquement comme une propriété de l'objet JS correspondant. A défaut, utilisez get() et set() pour lire et modifier la valeur de cette propriété. Les types VARIANT reconnus sont BOOLEAN, INT4, DOUBLE, NULL ou BSTR.

toString( ) returns String

Invoque la méthode toString(…) de COM. Si cette fonction n'existe pas, renvoie la propriété d'index nul (= 0).

```
 idl file:
   [propget, id(0), helpstring("property toString")]
HRESULT toString([out,retval] VARIANT *rv);
 implementation:
   STDMETHODIMP CFoo::get_whoami(VARIANT *rv)
   {
     rv->vt = VT_BSTR;
     rv->bstrVal = SysAllocString(L"[Foo object]");
```

```
  return S_OK;
}
```

Archive

Propriétés

count Number Nombre de fichiers inclus dans l'archive.

Méthodes

Archive(file String|Stream)

Le constructeur requiert un flux (Stream) où l'on peut se positionner (comme un fichier ou un bloc mémoire) ou un nom de fichier. L'instance Stream utilisée pour ouvrir l'archive reste ouverte (même si supprimée) jusqu'à ce que l'archive soit libérée.

close( )

Ferme l'archive et libère le flux (Stream) associé.

extract(index Number|String, stream Stream) => Number

Si le second paramètre est un flux (Stream), les données sont décompressées vers celui-ci. A défaut, un flux (en mémoire) contenant les données du fichier est retourné.

find(name String) => Number

Retourne l'index du fichier, ou -1 s'il n'existe pas.

has(name String|Number) => Number

Returne true si le fichier est dans l'archive.

name(index Number) => String

Retourne le nom du fichier.

size(index Number) => Number

Retourne la taille du fichier décompressé.

Form

Propriétés

count Number Nombre de questions

file_id Number Index de l'étiquette FORM (usuellement zéro)

name String Nom du fichier EZF

options Record Options du formulaire

Méthodes

Form(name StringIStream, index Number) => Form

name Nom du fichier Formulaire, texte XML ou flux (Stream).

index Un fichier HTML peut contenir plusieurs étiquettes (de formulaire). Si vous ne voulez pas lire les premières d'entre elle, spécifiez l'index du formulaire à l'intérieur du fichier.

Lit un formualire EZF ou HTML en mémoire. C'est une fonctionnaité particulère de l'interpréteur XML. Les propriétés Question sont accédées en utilisant les méthodes getXXX() et set XXX(). Les propriétés Question que vous pouvez utiliser sont :

- Name (fieldname): String
- Text :String
- Description:String
- Length: Number
- Help: String
- Extra (unparsed XML within the question): String
- Type: String (see below)
- Responses: Record (names = codes, values = descriptions)
- Options: Record
- Skips: Record

Pour chacune de ces propriétés, existe une méthode a getXXX() et setXXX() dont la syntaxe est simialire aux méthodes getType() et setType() décrite ci-après.

addQuestion(Type String, Fieldname String, Text String, Description String, Responses String, Length Integer, Options String, Before Integer) => Number

Responses Liste de codes et valeurs délimités par le caractère " | " (pipe)

Options Liste d'options délimités par le caractère " | " (pire)

Before Insérer avant cette question (-1 => à la fin)

find(fieldname String) => Number

fieldname Nom du champ

Retourne l'indice de la question (à partir de 0) qui correspond au nom de champ indiqué.

getType(index Number) => String

index Indice de la Question (à partir de 0)

Retrouve le type de question. Les types sont :

```
Text
Password
Date
Time
Number
```

```
Radio
Weighted
Check
Single
Multiple
Combo
Rank
Hidden
Section
Page
RichText
Image
PlainText
ToolButton
```

hasData(index Number) => Boolean

index Indice de la Question (à partir de 0)

Retourne true si le champ peut contenir des données ou false s'il s'agit d'un objet de mise en forme.

isPageStart(index Number) => Boolean

index Indice de la Question

Détermine si la question débute une nouvelle page. C'est la cas si :

- La question précédente a l'attribut fin de page ("end-of-page", EZSurvey)
- La question est une PAGE type (créée dans InterForm)
- La question the question a l'attribut nouvelle page ("new-page", EZSurvey)
- La question est une Section suivie par une question d'un type différent
- La question est la première du formulaire

move(start Number, finish Number) => Boolean

start Index d'origine de la question à déplacer

finish Index destination

Déplace une question de la position de départ (**start**) à la position d'arrivée (**finish**) dans le formulaire. Les autres questions sont réarrangées. Montrez vous prudent lors de l'usage de cette méthode lors d'un parcours de l'ensemble des questions.

remove(index Number) => Boolean

index Indice de la Question

Supprime une question. La question après l'indice (**index)** récupère l'indice de la question supprimée. Montrez vous prudent lors de l'usage de cette méthodes lors d'un parcours de l'ensemble des questions.

save(filename String) => String

filename Nouveau nom de fichier (optionel)

Enregistrement les modifications.

setType(index Number, type String) => String

index Indice de la Question (à partir de 0)

type Nouvelle valeur

Change le type d'une question. Voir getType() pou la liste des types.

Image

Propriétés

colors Number Nombre de couleurs (en plus de la couleur de fond)

height Number Hauteur (en pixels)

size Number Nombre d'octets (en fichier GIF)

width Number Largeur (en pixels)

Méthodes

Image(height Number, width Number) => Image

height Hauteur

width Largeur

Crée une nouvelle image GIF. La couleur de fond est blanc (couleur n° 0) et la couleur transparente a le numéro 127. 126 couleurs sont définissables par l'usager.

arc(X1 Number, Y1 Number, R Number, start Number, stop Number, color Number) => Boolean

X1 Abscisse du centre

Y1 Ordonnée du centre

R Rayon

start Angle de départ (en radians)

stop Angle d'arrivée (en radians)

color Numéro de la couleur

Dessine un arc.

color(R Number, G Number, B Number) => Integer

R 0..255

G 0..255

B 0..255

Ajoute une nouvelle couleur (RGB) et renvoie son numéro.

fill(X1 Number, Y1 Number, color Number) => Boolean

X1 Abscisse du point de départ

Y1 Ordonnée du point de départ

color Numéro de la couleur

Remplie tous les points reliés à celui choisi et de la même couleur pixels avec une nouvelle couleur.

getp(X1 Number, Y1 Number) => Number

X1 Abscisse

Y1 Ordonnée

Retourne la valeur du pixel (couleur) au point indiqué.

line(X1 Number, Y1 Number, X2 Number, Y2 Number, color Number) => Boolean

X1 Abscisse du point de départ

Y1 Ordonnée du point de départ

X2 Abscisse du point d'arrivée

Y2 Ordonnée du point d'arrivée

color Numéro de la couleur

Trace une ligne du point de départ au point d'arrivée d'un pixel de large (avec la couleur choisie)

print(X Number, Y Number, color Number, text String, background Number) => Boolean

X Abscisse du point de départ

Y Ordonnée du point de départ

color Numéro de la couleur

text Texte à afficher

background Numéro de la couleur de fond (0 par défaut)

Imprime le texte indiqué. A présent, la seule police disponible est Sans-Serif en 9 pixels.

setBGColor(R Number, G Number, B Number) => Boolean

R 0..255

G 0..255

B 0..255

Définie la couleur de fond (numéro 0).

setp(X1 Number, Y1 Number, C Number) => Boolean

X1 Abscisse du point

Y1 Ordonnée du point

C Numéro de la couleur

Définie la valeur d'un pixel (couleur du point).

slice(X1 Number, Y1 Number, R Number, start Number, stop Number, color Number) => Boolean

X1 Abscisse du centre

Y1 Ordonnée du centre

R Rayon

start Angle de départ (en radians)

stop Angle d'arrivée (en radians)

color Numéro de la couleur

Dessine une part de camembert.

write(out Stream) => Boolean

out Fichier cible ou connexion Web

Ecrit un fichier GIF (à partir de l'image générée)

Index

Méthodes

add(value String)

value Valeur Clef

Ajoute une clef à l'index

find(value String) => Number

value Valeur Clef

Retourne l'indice d'une entrée correspondant à la valeur clef ou -1 si aucune ne correspond.

rebuild( )

Reconstruit un index créé par Table.index().

Mail

Propriétés

name String Chaîne de connexion

Méthodes

Mail(type String, login String, password String, server String, smtpserver String, language String, return address String) => Mail

type	Système de messagerie : CCMAIL5, CCMAIL8, CCMAIL, BEYONDMAIL, VIM, NOTES, MAPI, MSMAIL, EXCHANGE, GROUPWISE, CMC, INTERNET, POP3 ou SMTP
login	Nom du compte
password	Mot de passe
server	Adresse du serveur ou profil windows mail
smtpserver	Adresse du serveur SMTP si différente de celle du serveur POP3
language	Chaîne spécifiant le langage utilisé pour l'encodage MIME, exemple : USASCII, ISO-8859-1

return address Adresse de réponse (usuellement celle de l'émetteur)

Se connecte à un serveur de messagerie. Dans le cas d'une messagerie Internet (INTERNET, POP3, SMTP sont équivalents), le mot de passe n'est pas indispensable. L'adresse de retour est login@server.

get( ) => Table

Renvoie la liste des messages sous la forme d'un objet Table. La méthode getMessage permet de récupérer le corps du message. La fermeture de la connexion est fermée, désactive la table.

send(recipient String, subject String, textNote String, htmlNote String) => Boolean

recipient Liste des adresses des destinataires séparées par une virgule

subject Sujet

textNote Note

htmlNote Note au format HTML (Internet seulement)

Envoie un message.

sendFiles(recipient String, subject String, attachments String, textNote String, htmlNote String) => Boolean

recipient Liste des adresses des destinataires séparées par une virgule

subject Sujet

attachments Liste de fichiers séparés par une virgule

textNote Note

htmlNote Note au format (Internet seulement)

Envoie un message avec des pièces jointes.

Matrix

```
include('matrix.js')
```

Propriétés

cols Number Nombre de colonnes

data Array Nombre de colonnes

rows Number Nombre de lignes

Méthodes

Matrix(data String)

Construit une nouvelle matrice où les lignes sont séparées par des "," et les colonnes par des espaces : `new Matrix("1 0, 0 1")` est la matrice de type unité 2x2.

Matrix(rows Number, cols Number, data ArrayINumbers)

rows Nombre de lignes

cols Nombre de colonnes

data Données initiales

Construit une nouvelle matrice en utilisant les éléments de **data** (ligne après ligne). Vous pouvez utiliser un objet Numbers si vous envisagez de travailler avec beaucoup de données.

Matrix.fit(O Integer, data Matrix) => Array

Fait correspondre les données Nx2 à polynôme d'ordre *O*. Renvoie un tableau C tel que le coefficient x^p coefficient du moindre carré soit C[p].

Matrix.identity(N Integer)

Renvoie une nouvelle matrice de type unité NxN

Matrix.solve(S Matrix, Y Matrix)

A partir d'une matrice carrée S et du vecteur Y, détermine X tel que S · X = Y

at(row Integer, column Integer) => Number

Voir get()

copy( ) => Matrix

Renvoie une copie de la matrice.

get(row Integer, column Integer) => Number

Renvoie l'entrée située à la ligne *row*, et à la colonne *column*. Aucun contrôle de la validité des valeurs indiquées.

put(row Integer, column Integer, value Number) => Number

Voir set()

set(row Integer, column Integer, value Number) => Number

Met à jour l'entrée située à la ligne *row,* et à la colonne *column*. Aucun contrôle de la validité des valeurs indiquées.

times(A Matrix, B Matrix) => Matrix

Renvoie A · B.

toSource( ) returns String

toString( ) returns String

transpose( ) returns Matrix

Retourne la matrice transposée.

zero( )

Mets à zéro toutes les données de la matrice

Numbers

Liste ordonnée rapide et efficace de nombre à virgule flottante en 64 bits. Numbers fonctionne comme un tableau et supporte `for .. in` et les opérateurs`[]`, mais les méthodes get() et set() sont bien plus efficaces.

Propriétés

length Integer Nombre d'éléments dans la liste.

Méthodes

Numbers(values String)

Crée un nouveau tableau, initialisé par la conversion des valeurs (*values*) en *numbers*. L'espace, la virgule, la tabulation sont acceptés comme séparateurs ; d'autres caractères sont utilisables (avec une perte de performances toutefois).

Numbers(length Integer)

Crée un nouveau tableau dont les valeurs sont initialisées à 0.

Numbers(source Numbers)

Crée un nouveau tableau réplique exacte de l'objet Numbers indiqué.

add(value)

Incrémente les valeurs du tableau avec celle indiquée (v*alue*)

at(index Integer) => Number

Voir get()

close( )

Libère immédiatement la mémoire allouée à cet objet.

exec( )

Méthode spéciale permettant de faire rapidement des opérations sur vecteurs avec un évaluateur RPN. Chaque paramètre passé à cette méthode doit être un objet Numbers (tous de la même taille !) ou une chaîne désignant une opération mathématique. Exemple :

```
a = new Numbers(5)
b = new Numbers("1 2 3 4 5")
c = new Numbers("10 20 30 40 50")
a.exec(b,c,'+')
writeln(a)
```

Le résultat est :

```
11,22,33,44,55
```

La pile de la calculatrice débute par (**b**, **c**, '+') et le résultat est stocké dans **a** (l'objet dont la méthode exec() est invoquée). Les arguments sont évalués de la gauche vers la droite de telle sorte que **a** est toujours le résultat de chaque opération et le nombre en fin de pile pour la prochaine opération. Vous pouvez ainsi faire :

```
a = new Numbers("1 2 3 4 5")
b = new Numbers("10 20 30 40 50")
```

```
a.exec(b,'+')
writeln(a)
```

avec le même résultat.

Lorsqu'il manque une opérande, l'objet dont la méthode est appelée est ajouté en fin de pile. Cette méthode peut avoir autant d'arguments que souhaité (dès lors que chaque tableau soit de la même taiille). Les opérateurs supportés et le nombre d'arguments requis sont spécifiés ci-après :

- + 2
- * 2
- - 2 (or 1 for négation)
- / 2 (or 1 for inversion)
- cos 1
- sin 1
- tan 1
- atan2 2
- atan 1
- pow 2
- exp 1
- log 1

Remarque : la négation et la division (inversion) unaires sont autorisées comme raccourcis

get(index Integer) => Number

Renvoie le nombre à la position *index*. Lève une exception si l'*index* est hors limites.

log( )

Remplace chaque élément du tableau par son logarithme (de base e)

max(start Integer, length Integer)

Renvoie l' **index** de la valeur maximale dans l'intervalle du tableau délimité par une position de départ (*start*) et un nombre d'éléments (*length*).

min(start Integer, length Integer)

Renvoie l' **index** de la valeur minimale dans l'intervalle du tableau délimité par une position de départ (*start*) et un nombre d'éléments (*length*).

pow(p)

Eléve chaque élément du tableau à la puissance *p*

put(index Integer, value Number) => Number

Voir set()

resize(length Integer)

Modifie la taille diu tableau. Lors d'une augmentation de taille, la mémoire requise est allouée et les valeurs ajoutées prennent 0 pour valeur. Lors d'une diminution de la taille, la mémoire résiduelle n'est pas libérée (elle sera réemployée en cas d'augmentation).

scale(value)

Multiplie chaque élément du tableau par la valeur indiquée (*value*)

set(index Integer, value Number) => Number

Actualise la valeur à la position (*index*) avec la valeur indiquée. Lève une exception si l'*index* est hors limites.

sum(start Integer, length Integer)

Renvoie la somme de tous les nombres de l'intervalle du tableau défini par la position de départ *start* et comprenant *length* éléments.

sum2(start Integer, length Integer)

Renvoie la somme des carrés de tous les nombres de l'intervalle du tableau défini par la position de départ *start* et comprenant *length* éléments.. Pourquoi voudriez-vous faire cela ? Pour calculer l'écart type rapidement.

```
var n = new Numbers("1 495 52 2 34 5 6 1 12 35 6 2 34")
writeln("Average: ",n.sum()/n.length)
writeln("Average of the square: ", n.sum2()/n.length)
```

toString(delimiter string, start Integer, length Integer) returns String

La classe Numbers fonctionne comme la classe JavaScript Array. Vous pouvez modifier le formatage des données renvoyées en choisissant votre séparateur (la virgule par défaut). Spécifiez la position de départ (*start*) et le nombre d'éléments (*length*) pour limiter l'opération à un intervalle choisi.

ODBC

Propriétés

className	String	"ODBC"
error	String	The last error message. It is initially undefined, and only changes when a query() or exec() function fails.
name	String	Driver name

Méthodes

ODBC(connection string String) => ODBC

connection string: "DSN=nom de la source;UID=identifiant;PWD=mot de passe;"

Se connecte à un pilote ODBC. Vous pouvez ainsi utiliser

```
var myDB = new ODBC("DRIVER={MySQL ODBC 3.51
Driver};SERVER=localhost;DATABASE=dbname;USER=root;PASSWORD=password;
OPTION=3;");
var myReader = myDB.query('select * from people');
```

ou

```
var myT = new Table('odbc://DRIVER={MySQL ODBC 3.51
Driver};SERVER=localhost;DATABASE=dbname;USER=root;PASSWORD=pa
ssword;OPTION=3;/people')
```

La différence est que la méthode query() renvoie une Table en lecture seule alors que le constructeur Table permet d'effectuer des modifications à l'aide des méthodes set() et setRow().
Vous pouvez apercevoir une boîte de dialogue d'erreur de Microsoft. Cela signifie seulement qu'il y a une erreur dans votre chaîne de connexion. JSDB est une interface en ligne de commande SQL médiocre . Préférez Raosoft's SQL.exe pour cet usage. [ndlr : cette dernière remarque n'engage que Shanti Rao ; pour ma part je trouve JSDB très efficace]

close( ) => Boolean

Ferme la connexion à la base de données et toutes les tables ouvertes avec la méthode query(). Les objets tables sont invalides (tout appel à l'une de leur méthode échoue) par contre ils ne sont pas supprimés.

columns(table String, longFormat Boolean) => Array

table A table name

Renvoie un tableau comprenant tous les noms de colonne de la table spécifiée. Si **longFormat** est vrai (*true*), la valeur renvoyée ressemble à `["id INTEGER","name VARCHAR(32)","value DECIMAL(12,5)"]`.

exec(command String) => Boolean

command "ALTER TABLE ..."

Cette méthode appelle SQLExecDirect() sur la base de données. Renvoie *true* si l'opération réussie (*false* sinon). Vous pouvez consulter la propriété *error* pour déterminer la nature de l'anomalie en cas d'erreur.

keys(table String) => Array

table A table name

Renvoie la liste de tous les index de la table spécifiée.

query(query String) => Table

query "SELECT * FROM table WHERE ..."

Exécute une commande SQL et renvoie le résultat dans un objet Table en lecture seule. L'objet Table fournit un cache et un accès indexé aux enregistrements, même si le pilote ODBC ne le permet pas.Si vous souhaitez seulement lire une table SQL avec une simple requête ceci est équivalent à **new Table('odbc://userid:password@service/tablename?field1=xxx*&field2=yyy')**. Les deux méthodes d'accès effectuent du chargement à la demande à partir de la source de données SQL.

table(name String) => Table

name Table name

Ouvre un objet Table dans la base de données (image de la table SQL de même nom).

tables( ) => Array

Renvoie un tableau de toutes les tables de la base de données.

Process

Propriétés

active Boolean Indique que le programme est encore en cours de fonctionnement.

className String "Process"

Méthodes

Process(cmd String)

cmd Line de commande : fichier .exe ou nom d'un document.

Le constructeur essaie de démarrer le programme. Seuiement les programmes en .exe sont lancés. Si *cmd* est un document ou un URL, le shell Windows lance le programme approprié à la requête pour ouvrir le document. Les noms des fichiers des documents doivent être pleinement qualifiés (inclure le chemin d'accès au fichier ad-hoc).

close( )

Envoie un message d'arrêt (quit) au processus fils et ferme les handles associés

Record

Propriétés

caseSensitive Boolean Les champs sont-ils sensibles à la casse?

Count Number Nombre de paires nom-valeur stockées dans l'enregistrement

Length Number Nombre de paires nom-valeur stockées dans l'enregistrement

Méthodes

Record(caseSensitive Boolean) => Record

caseSensitive true pour des champs sensibles à la casse

Crée un enregistrement. Utilisez Record(true) pour permettre au recherche sur nom d'être sensible à la casse.

Record(constructor Record) => Record

constructor Crée un enregistrement en copiant les champs et valeurs de l'instance passée en paramètre. Les valeurs sont converties par la méthode object's toString() .

Crée un enregistrement à partir d'un autre.

Record(initial String, delimiter String) => Record

Initial Valeurs initiales sous la forme de paires nom=valeur, séparées par un caractère (*delimiter*), exemple : `a=b,c=d,e=f,...`

delimiter Séparateur, par défaut la virgule

Crée un enregistrement avec les valeurs initiales. Voir read().

append(data Record) => Number

data Additional data

Ajoute les paires nom-valeur d'un autre enregistrement. Les paires dotées du même nom sont remplacées.

clear( ) => Boolean

Supprime toutes les paires nom-valeur de l'enregistrement.

get(name String) => String

Name Nom du champ

Renvoie la valeur associée au champ ou une chaîne vide si le champ n'existe pas.

has(name String) => Boolean

Name Nom du champ

Renvoie true si le champ existe dans l'enregistrement.

name(index Number) => String

Index Index de 0 à **length** - 1

Les paires nom-valeur sont indexées séquentiellement. Cette méthode renvoie le nom de la paire à la position indiquée.

read(data String, delimiter String) => Number

data Chaîne délimitée à lire (`a=b&c=d&e=f&...`)

delimiter Séparateur utilisé dans la chaîne à lire, par défaut & (HTTP GET)

Charge un enregistrement avec le contenu d'une chaîne de paires délimitées.

readINI(file String, file String) => Number

file Nom du fichier INI (dans le répertoire Windows par défaut)

file Section du fichier INI

Lit une section d'un fichier INI Windows. Renvoie le nombre d'entrées.

set(name String|Integer, value String) => Boolean

name Nom du champ ou index

value Nouvelle valeur

Affecte une valeur (value) à un champ ou crée une nouvelle paire nom-valeur (si elle n'existe pas).

toString( ) => String

Renvoie une liste délimitée par '\n' de l'enregistrement.

unSet(name String) => Boolean

name Nom du champ

Retire la paire nom-valeur de l'enregistrement si elle existe.

value(index Number) => String

index Index de 0 à **length** - 1

Les paires nom-valeur sont indexées séquentiellement. Cette méthode renvoie la valeur de la paire à la position indiquée.

write(delimiter String) => String

delimiter Séparateur utilisé dans la chaîne à lire, par défaut & (HTTP GET)

Renvoie l'enregistrement sous la fome d'une chaîne délimitée (par le séparateur choisi) contenant l'ensemble des paires nom-valeur.

writeINI(file String, file String) => Number

file Nom du fichier INI (dans le répertoire Windows par défaut)

file Section du fichier INI

Ecrit une section d'un fichier INI Windows. Renvoie le nombre d'entrées.

SQLite

Propriétés

className	String	"SQLite"
error	String	Dernière erreur reportée par SQLite.
name	String	Nom du fichier de la base de données, ":memory:" s'il s'agit d'une base en mémoire.

Méthodes

SQLite(database String)

database Nom du fichier

Ouvre ou crée un fichier de base de données SQLite3. Crée une base de données en mémoire si le nom de fichier est omis.

close( )

Ferme la base de données et libére les ressources (mémoire) associées.

exec(command String, callback Function, opaque) => Boolean

command Requête SQL

callback Fonction Callback

opaque Paramétres passés à la fonction callback

Exécute une requête SQL. Si des lignes sont retounées, appelle la fonction `callback(record, opaque)` pour chaque ligne. L'enregistrement résultat (record) est réutilisé.
[ndlr : le contexte (this) n'est pas mémorisé d'où l'intérêt du paramètre *opaque*]

Server

Propriétés

anyoneWaiting	Boolean	Est-ce qu'un client attend une réponse ?
hostName	String	
Port	Number	

Méthodes

Server(port Number) => Server

Port Port (par défaut 80).

accept( ) => Stream

Accepte la prochaine connexion en attente et renvoie un (pointeur vers un) objet Stream pour communiquer avec.

close( ) => Boolean

Ferme le serveur et toutes ses connexions.

toString( ) => String

Renvoie la configuration du serveur et son port.

Stream

Propriétés

canRead	Boolean	Vrai (true) si des caratères sont disponibles dans le buffer d'entrée ou si la fin du flux (stream) mémoire n'est pas atteinte.
canWrite	Boolean	True si le flux permet l'écriture.
eof	Boolean	Est-ce que la fin du flux est atteinte ? Pour les flux COM, précise si le flux a été fermé. Les flux réseau (htpp://, net://) peuvent seulement détecter s'ils ont été fermés après qu'un appel à read() ou write() retourne 0. Eof peut détecter des conditions d'erreur sur le socket mais cela arrive très rarement. Malheureusement, il n'y a pas de méthode fiable pour déterminer si un socket est fermé ou non sans une opération de lecture ou d'écriture.
hasChildren	Boolean	Voir readTag()
hostAddress	String	Adresse de l'hôte (flux réseau seulement)
hostName	String	Nom de l'hôte (flux réseau seulement)
name	String	Nom du fichier ou URL
pos	Number	Voir **position**
position	Number	Position courante en octet depuis le début. Cette propriété est inexploitable pour les flux http et COM.
size	Number	Nombre total d'octets du fichier ou du flux mémoire.
status	StringINumber	Message de réponse du serveur Web tel que 'HTTP/1.1 200 OK'.
stderr	Stream	Capture le flux stderr de exec://
tagText	String	Voir readTag()

Méthodes

Stream(name String, type String) => Stream

name Fichier, URL, Nombre d'octet pour un flux mémoire. Les URL adoptent la forme :

```
http://www.jsdb.org/
file://c:/test.txt
net://server.com:port
com1://56000 (or com2, com3, ...)
text://number_of_bytes
temp://number_of_bytes
exec://command line
```

type Précise le mode d'ouverture pour les fichiers (rwactb+). Un + indique un flux non bufferisé et `b` signale le mode binaire. Le mdoe par défaut est 'rt' (lecture d'un fichier texte).

Mode	Description	Utilisable avec
`a`	Ajout à un fichier en écriture seule. Positionnement dans le flux et lecture impossible.	`+tb`
`rw`	Création d'un nouveau fichier ou ouverture d'un fichier existant en lecture et écriture.	`+tb`
`w`	Création ou remplacement d'un fichier avec accès en lecture et écriture	`+tbr`
`r`	Ouverture d'un fichier existant en lecture	`+tbw`
`t`	Mode texte : `\r` est omis en lecture et `\n` est converti en `\r\n` lors de l'écriture sur disque. `seek` et `pos` sont opérationnels.	`arw+`
`b`	Mode binaire : aucune gestion de CRLF.	`arw+`
`+`	Pas de buffer : n'utilise pas la lecture anticipée, ni le cache d'écriture.	`arwtb`
`d`	Pipes seulement : crée un process détaché avec sa propre fenêtre console.	`(aucune)`

Si ce paramètre est omis, tout URL http:// utilise la configuration proxy d'Internet Explorer et décode les transmissions "Transfer-encoding: Chunked" (souvent créé par les CGI Apache). S'il est mis à 0, JSDB n'utilisera pas le proxy, ni ne décodera les flux "chunk". La propriété "status" est affectée de l'intégralité du statut HTTP (HTTP/1.1 200 OK) et pas seulement du code de statut (200). Dans tous les cas , l'usage de Record.readMIME(stream) pour analyser les en-têtes HTTP avant la lecture du contenu est vivement recommandé.

Par défaut, les flux réseau net:// fixe l'option TCP_NODELAY. Mettre ce paramètre à 0 pour désactiver l'option TCP_NODELAY. L'algorithme Nagle regroupe alors les appels write() successifs afin de limiter le nombre de paquets générés.

Si *name* est une chaîne, jsdb va d'abord essayer de l'ouvrir comme un URL. Pour récupérer le contenu d'une page Web en utilisant GET et les en-têtes standards, utilisez **Stream('http://server.com/')**. La propriété de statut est initialisée avec la première ligne de réponse du serveur. Pour simplement ouvrir un socket et envoyer vos propres en-têtes, utilisez **a = new Stream('net://server.com:80/'); a.writeln('POST /file.cgi/ HTTP/1.1\r\n')**. Le numéro de port par défaut (si omis) est 80. Un URL de type **file://** est toujours ouvert en mode binaire et en lecture seule. Si **name** ne s'apparente pas à un URL (absence de //), jsdb essaie d'ouvrir un fichier. Les caractères slash (/) sont convertis en backslash (\ ) sous Windows. Si **name** prend pour valeur **fifo://** alors un buffer FIFO est créé en mémoire. Si **name** est omis ou s'il s'agit d'un nombre, un flux mémoire est créé ayant pour la taille initiale celle indiquée (par **name**).

Si **name** débute avec "exec://", le flux renvoyé est un pipe vers le process fils. Essayez ceci:

```
p = new Stream("exec://jsdb.exe","d")
p.writeln("writeln('hello world');")
writeln(p.readln())
p.writeln("quit");
```

append(data Stream, length Number) => String

data Flux à ajouter

length Nombre maximum d'octets à lire

Ajoute des octets à partir de **data** tant que sa fin n'est pas atteinte. Pensez à reculer la position courante dans le flux **data** avant l'appel à append().

appendText(data Stream, length Number) => String

data Flux à ajouter

length Nombre maximum d'octets à lire

Comme append() avec convertion de \n' en \r\n'. Autorise la préparation d'un texte pour un contrôle edit de Windows.

clear( ) => Boolean

Réinitialise un flux mémoire. La taille est mise à zéro et la position courante remise au début mais la mémoire n'est pas libérée.

eatChars(skip String) => String

skip Caractères à passer

Lit le flux octet par octet. Si l'octet lu n'est pas dans la chaîne de ceux à éviter, renvoie cet octet lu sous la forme d'une chaîne. Un caractère null interrompt l'analyse.
Exemple : `var lastchar = file.eatChars(' \t\r\n')`

flush( )

Pour les flux fichiers, vide le buffer de 64k (en écrivant les données sur disque). Pour créé un flux non bufferisé, ouvrez le avec l'option "+".

format(source Stream|String, values Record, start_delimiter String, end_delimiter String)

Lit le flux source en rempaçant toutes les expressions de type {*expression*} par la valeur associée à *expression* dans l'enregistrement indiqué (**record**). Les séparateurs standards (accolades) peuvent être remplacés par les chaînes indiquées (**start_delimiter**, **end_delimiter**). Si **end_delimiter** est null et **start_delimiter** non null alors **end_delimiter** = **start_delimiter**.
Exemple :

```
var Str=new Stream();
var Dta=new Record('name=Shanti,home=California');
Str.format('{name} lives in {home}\n',Dta);
Str.rewind();
writeln(Str.readln());
Str.close();
```

get( ) => String

Lit un octet à partir du flux et avance la position courante (d'un octet).

goBack(offset Number) => String

Recule la position courante du nombre d'octets indiqué (**offset**).

goForward(offset Number) => String

Avance la position courante du nombre d'octets indiqué (**offset**).

peek( ) => String

Renvoie le prochain octet dans du flux mémoire ou fichier sans déplacer la position courante. Les flux HTTP et COM n'étant pas bufferisés, vous devez éviter d'utiliser cette méthode sans vous assurez préalablement de l'origine du flux.

put( ) => String

Ecrit un octet dans le flux et avance la position courante.

putBack(offset Number) => String

Recule la position courante du nombre d'octets indiqué (**offset**).

read(count Number, dest Stream) => String

count Nombre d'octets

dest Flux destination (optionnel)

Lit le nombre d'octets indiqués (au plus) à partir d'un fichier et les renvoie sous forme d'une chaîne. Si **dest** est précisé la méthode **readFile()** est appelée (à la place de **read()**).

IMPORTANT

La méthode read adopte un comportement particulier afin de faciliter le téléchargement de donnés à partir de flux HTTP. Le code pour lire le contenu d'une page Web est :

```
var s = new Stream('http://www.jsdb.org/')
```

Les entêtes de réponse, le code de statut et le texte associé sont

respectivement stockés dans les propriétés **header**, **status** et **statusText** de l'objet Stream. La réponse du serveur est automatiquement décodée et la propriété **header** modifiée en conséquence. Si vous préférez gérer les entêtes vous-même (comme initialement proposé en version 17.2.3 et antérieures), utilisez la syntaxe ci-après :

```
var s = new Stream('http://www.jsdb.org/',1)
```

Dans les rares occasions où vous devrez envoyer des entêtes spécifiques au serveur, utilisez :

```
var s = new Stream('http://www.jsdb.org/',
{header-field: "header-value", ...})
```

Si vous souhaitez envoyer des entêtes spécifiques et analyser intégralement la réponse HTTP (code de statut inclus), utilisez :

```
var s = new Stream('http://www.jsdb.org/',0,
{header-field: "header-value", ...})
```

readByte( ) => Number

Voir readUInt8()

readFile(size Number, dest Stream) => StringINumber

size Nombre maximal d'octets

dest Emplacement de stockage des données

Lit le flux en mode binaire et délivre la chaîne résultante. **size** et **dest** peuvent être renseignés dans n'importe quel ordre. Si **dest** est précisé, la valeur renvoyée est le nombre d'octets copiés à partir du flux vers le flux de destination (**dest**). Le flux est lu en mode binaire, la chaîne peut contenir des caractères null.

readInt(network Boolean) => Number

network BigEndian

Lit un entier binaire de 32 bits signé. (JavaScript utilise des entiers de 31 bits en interne). Mettre **BigEndian** à vrai (**true**) pour utiliser l'ordre natif des octets pour Sun et Mac.
Exemple :

```
var Str=new Stream();
Str.writeByte(1);
Str.writeByte(0);
Str.writeByte(0);
Str.writeByte(0);
Str.rewind();
writeln(Str.readInt()); // false est facultatif
writeln(Str.readInt(true));
Str.close();

Affiche 1 (LittleEndian) puis 8614456 (BigEndian)
```

readInt16(network Boolean) => Number

network BigEndian

Lit un entier binaire de 16 bits signé.

readInt32(network Boolean) => Number

network BigEndian

Voir **readInt()**.

readInt8( ) => Number

Lit un entier binaire 8 bits signé.

readLine( ) => String

Voir readln.

readList(data Record, delimiter String, equals String) => Number

Data Données à lire

delimiter Séparateur de paires ("\n" par défaut - sans garantie -)

equals Séparateur Nom/valeur ("=" par défaut)

Lit une liste de paires. Les séparateurs (de paires et de Nom/valeur) doivent être des caractères. Remarque : les séquences "\r\n" sont converties en "\n" lors de la mise en mémoire. La valeur renvoyée correspond au nombre de paires lues.
Exemple :

```
var Dta=new Record();
var Str=new Stream();
var Fld;
Str.write('a=b\nc=d');
Str.rewind();
Str.readList(Dta);
Str.close();
writeln(Dta.get('a'),Dta.get('c'));
```

Affiche : bd

readListB(data Record) => String

Lit des données à partir d'un enregistrement au format binaire.

readMIME(data Record) => Number

Data Données à lire

Lit une liste de paire nom-valeur au format MIME.
Reads a name-value list in MIME format. Les espaces après les deux-points sont significatifs et présument que toute ligne non vide commençant par un espace continue la ligne précédente. Renvoie le nombre de paires lues.

readTag(data Record, allowed String) => String

Data Champs étiquette

allowed Liste des champs autorisés (séparés par une virgule, sensibles à la casse)

Méthode Utilisée pour l'analyse des fichiers XML. Cette méthode analyse le fichier à la recherche de la prochaine balise. Si la liste des champs autorisés (**allowed**) est précisée, seulement les balises (d'ouverture de fermeture)

acceptées stoppent la recherche. En son absence, tout caractère < interrompt l'analyse. Le texte parcouru est stocké dans le paramètre **tagText** du stream. Lorsqu'une balise est atteinte ses paramètres sont stockés dans l'enregistrement **data**. Les séquences d'échappement XML < > & et " sont automatiquement décodées dans les paramètres mais pas dans les sections CDATA. Si la balise a un marqueur de fin (ex: `<tag field=value />`), le paramètre de flux **hasChildren** est mis à faux (**false**) à vrai (**true**) dans les autres cas. Le nom de la balise est renvoyé. Si la fin du fichier est atteinte, la valeur renvoyée est une chaîne vide. Si la balise est "![CDATA[", readTag() recherche "]]>" et ajoute le texte intermédiaire à **tagText**. Le module "xml.js" illustre l'usage de cette méthode.

```
var params = new Record; var name = stream.readTag(params,allowed);
```

readText( ) => String

Lit entièrement le flux, convertit tous les "\n" en "\r\n" et renvoie la chaîne résultante.

readUInt16(network Boolean) => Number

network BigEndian

Lit un entier binaire de 16 bits non signé.

readUInt8( ) => Number

Lit un entier binaire de 16 bits non signé.

readUntil(search String, skipped Stream) => Boolean

search Chaîne recherchée (qui interrompt la lecture)

skipped Flux cible des données lues jusqu'à la rencontre de la chaîne recherchée

Renvoie **true** si la chaîne recherchée a été trouvée. readUntil() est sensible à la casse. Pour une recherche binaire, utilisez readUntilBytes(). Si ***search*** est une chaîne unicode, elle sera convertie en UTF-8.
Exemple :

```
var memory = new Stream();
var file = new Stream('test.txt');
file.readUntil('</file>',memory) ;
writeln(memory);
```

readUntilBytes(search String, skipped Stream) => Boolean

search Motif recherché (suite d'octets qui stoppe la lecture)

skipped Flux cible des données lues jusqu'à la rencontre de la chaîne recherchée

Version binaire de readUntil(), readUntilBytes() peut rechercher des motif comprenant des caractères null.

readln(delimiter String) => String

delimiter Caractère de fin de ligne ("\n" par défaut)

Renvoie la prochaine ligne de texte dans le fichier (une chaîne vide lorsque la fin du fichier est atteinte).

resize(size Number) => String

size Spécifie la taille du nouveau buffer

Redimensionne un flux mémoire. Si la taille du flux diminue, les données sont tronquées à partir de la fin. Si la position courante se retrouve en dehors du flux, elle est déplacée à la fin du flux résultant. Si la taille est augmentée, de la mémoire est allouée par contre aucune donnée n'est ajoutée.

rewind( ) => String

Pour un fichier ou un flux mémoire, déplace la position courante au début du flux.
Exemple :

```
temp = new Stream();
temp.write('hello, world');
temp.rewind() ;
file = new Stream('test.txt','at') ;
file.append(temp)
```

seek(location Number) => String

Déplace la position courante à celle indiquée (calculée à partir du début du flux)

setEndOfFile(length Number) => Boolean

length Nouvelle taille maximale du fichier

Change la taille d'un fichier ou d'un flux mémoire. Les flux mémoire peuvent être tronqués ? Les flux fichiers peuvent être tronqués ou étendus.

toString( ) => String

Pour un fichier, un flux COM ou http, renvoie le nom du fichier. Pour un flux mémoire renvoie le buffer en intégralité.

write(data... String) => Number

data... Objets à écrire dans le flux

Exemple :
stream.write('hello', '4', 5).
Si les arguments de cette méthode ne sont pas des chaînes, la méthode toString() de chaque entité sera appellée. Renvoie le nombre d'octets écrits.

writeByte(value Number) => Number

Voir writeUInt8

writeInt(value Number, network Boolean) => Number

network BigEndian

Ecrit un entier binaire sur 32 bits au format Intel (LittleEnfian ou BigEndian si à **true**).

writeInt16(value Number, network Boolean) => Number

network BigEndian

Ecrit un entier binaire sur 16 bits au format Intel (LittleEndian ou BigEndian si à **true**).

writeInt32(value Number, network Boolean) => Number

network BigEndian

Voir **writeInt()**.

writeInt8(value Number) returns Number

Ecrit un entier binaire sur 8 bits au format Intel (LittleEndian ou BigEndian si à **true**).

writeLine( ) returns String

Voir writeln.

writeList(data Record, delimiter String, equals String) => Number

Data Données à écrire

delimiter Séparateur de paires à utiliser ("/" par défaut - sans garantie -)

equals Séparateur Nom/valeur ("=" par défaut)

Ecrit une liste de paires nom-valeur délimitée (par le séparateur de paires). Vous pouvez utiliser n'importe quelle chaîne comme séparateur (de paires ou de nom-valeur). La sortie par défaut est `name=value/name1=value1/name2=value2/...` Renvoie le nombre de paires écrites.

writeListB(data Record) => String

Ecrit des données à partie d'un enregistrement au format binaire. L'équivalent C++ est :

```
for (int i=0; i< data.count; i++) {
 fwrite(data[i].name,1,strlen(data[i].name) + 1, file);
 fwrite(data[i].value,1,strlen(data[i].value) + 1, file);
}
fwrite("",1,1,file);
```

writeMIME(data Record) => Number

Data Données à écrire

Ecrit une liste de paires nom-valeur au format MIME. C'est l'équivalent de `writeList(data,': ','\n')`. Une ligne vide termine la liste. Renvoie le nombre de paires écrites.

writeUInt16(value Number, network Boolean) => Number

network BigEndian

Ecrit un entier binaire non signé sur 16 bits au format Intel (LittleEndian ou BigEndian si à **true**).

writeUInt8(value Number) => Number

Ecrit un entier binaire non signé sur 8 bits au format Intel (LittleEndian ou BigEndian si à **true**).

writeln( ) => String

Comme write mais ajoute un retour chariot (\n pour les flux mémoire, \r\n pour les flux fichier). Le système d'entrée sortie s'assure que pour les fichiers en mode texte, les retours chariot \n en memoire soient correctement convertis en \r\n sur disque. Vous pouvez utilisez **pos** et **seek()** pour déplacer la position courante au sein des fichiers texte, mais ne vous basez pas dessus pour compter les octets dans les opérations de lecture écriture effectuées avec read() et write().

Table

Propriétés

className	String	"Table"
colCount	Number	Nombre de colonnes La numérotation débute à 1 (0 indique une colonne invalide).
count	Number	Voir rowCount
error	String	Dernier message d'erreur
length	Number	Voir rowCount
name	String	Nom du fichier associé à l'objet Table
rowCount	Number	Nombre de lignes La numérotation débute à 1 (0 indique une ligne invalide).

Méthodes

Table(name String) => Table

name Nom de la source de données, de la forme :

```
drive:/directory/filename.asc
drive:/directory/filename.dbf
odbc://login:password@data_source/table
notes://login:password@server/table (non disponible
actuellement)
```

Ouvre une table. La source de données peut ne pas être rensignée pour créer une feuille de tableur en mémoire.

Table.create(name String, fields String) => Boolean

name Nom de la nouvelle table à créer

fields "nom_du_champ [type] taille, ..."

(nouveau en 1.7) Crée une nouvelle table xBase. **name** est le nom du fichier(.dbf). **fields** comporte la liste des définitions des colonnes, par exemple `name C 33, address C 33, age N 3.`
Le nom des champs peut avoir jusqu'à 10 caractères. Le type et la taille des champs sont optionnels, leur valeur par défaut est C 255. La taille maximale d'un champ est 65535. La compatibilité xBase requiert que la taille totale de l'enregistrement soit inférieure à 65535. Les types acceptés sont Character, Number, Date, or Time (seule la première lettre est requise).

add(data RecordIObject) => Boolean

data Valeurs initiales

Ajoute une nouvelle ligne à la table, lui affecte les valeurs de **data** et renvoie l'indice de la nouvelle ligne.
Base de données SQL :

- JSDB ne gère pas les champs autoincrement dans les requêtes INSERT. La base de données doit s'en charger.
- Pour les champs clef qui ne sont pas de type autoincrement, JSDB génére une valeur unique, de 24 caractères, basée sur l 'heure système

et un compteur stocké dans la base de registre. Cette valeur ne se répéte pas sur la durée de vie du système solaire [ndrl : on peut effectivement considérer qu'elle est unique…].

- Si l'ajout s'effectue à une table Oracle et qu'une valeur de clef primaire n'est pas fournie, JSDB utilise *seq_**tablename**.nextval*.
- Si l'argument *data* est un Object, les valeurs chaîne (string) sont converties en UTF-8.

addColumn(name String) => Boolean

name Nom de la colonne

Ajoute une nouvelle colonne (qui doit posséder un nom unique) et renvoie le (nouveau) nombre de colonnes.

append(source String, callback Function, opaque Function) => Boolean

source Table source

callback Fonction callback

opaque Paramétres passés à la fonction callback

Copie les données d'une autre table. La fonction callback est de la forme `function cb(record, opaque, ...)`, où `record` est l'indice de la ligne en cours de transfert, et `opaque` (et ...) sont les paramètres complémentaires passés à `append()`.

column(column String) => Number

column Nom du champ

Renvoie l'indice de la colonne qui correspond au nom de champ. Si le nom n'existe pas dans la base de données, renvoie 0.

data(row Number, column Number|String) => String

row Numéro de la ligne (débute à 1)

column Numéro de colonne ou nom

Renvoie les données à la position indiquée dans la table. Vous pouvez déterminer si une ligne est marquée supprimée à l'aide du premier caractère de `table.data(row,0)` (prend alors 'D' pour valeur).

del(data Number) => Boolean

data Numéro de ligne

Supprime la ligne de la table.

deleteColumn(column Number) => Boolean

column Numéro de colonne

Retire la colonne d'une table tableur.

error( ) => String

Stocke le dernier message d'erreur renvoyé par la table. Très pratique pour la mise au point sur des tables SQL.

find(query Record, start Number, direction Number) => Number

query Valeur à rechercher, par exemple :

```
query = new Record('name=Alice');
```

```
row = table.find(query)
```

start Indice de la ligne à partir de laquelle la recherche débute (par défaut la première)

direction +1 recherche avant (de la position indiquée vers la dernière ligne), or -1 recherche arrière (de la position indiquée vers la première ligne)

Le résultat de la dernière recherche est stocké dans la propriété `lastFind` de la table. Les lignes supprimées ne sont pas prise en compte lors de la recherche.

findNext(query Record) => Number

query Requête de find()

Continue la dernière recherche (cherche l'élément suivant).

get(row Number, column NumberIString) => String

row Indice de la ligne (débute à 1)

column Indice de colonne ou nom

Voir data()

getMessage(row Number) => String

Row Indice de la ligne (débute à 1)

Pour les tables de messages mail (voir Mail), renvoie le texte du message suivant.

getRow(row Number, data Record) => BooleanIRecord

row Indice de la ligne

data Valeurs

Renvoie un enregistrement (record) contenant les valeurs de la ligne demandée. Si l'argument data est précisé, remplit l'enregistrement. La fonction renvoie alors `true` or `false`.

getWhere(row Number) => String

row Indice de la ligne (débute à 1)

Pour la mise au point avec des tables ODBC, renvoie la clause WHERE de la requête utilisée pour identifier une ligne particulière.

index(column StringINumber) => Index

column Nom de la colonne

Cette fonction renvoie un index à la base de données. Si vous appelez `index()`avec un nom ou un numéro de colonne, l'objet renvoyé dispose de la méthode `find(key)`, où `key` peut être la valeur du champ ou un enregistrement (record) comprenant la valeur appropriée.
Si vous appelez avec une liste `index()` de noms de champ, l'index se contruit à partir des chaînes où les données sont séparées par des retours chariot.
Vous pouvez ainsi appeler `index().find('value1\nvalue2\nvalue3')` ou `index().find(new Record('field1=value1,field2=value2,field3=value3')`.
La méthode `index().find()`renvoie le nombre de lignes compatibles ou -1 si aucune n'est trouvée.

La construction des index prenant du temps, vous pouvez procéder de la façon suivante :

```
var index = table.index('field')
writeln('adding record ',table.addRow(data))
index.add(data)
writeln('found record ',index.find(data));
```

save(file String|Stream, delimiter String, titles Boolean) => Boolean

file Nom du fichier

delimiter Par défaut : séparateur utilisé en lecture ou tab

titles True par défaut

Enregistre les modifications opérées. Si la base de données n'a pas de nom la fonction échoue. Pour les bases de données ODBC, effectue un COMMIT. Pour les bases de données DBF, ne fait rien.Pour les bases de données ASCII, vous pouvez enregistrer le fichier comme un flux. Dans ce cas, spécifiez un séparateur (**delimiter**) (ou null pour utiliser celui par défaut) et choisissez d'inclure (ou non) la ligne d'en-tête (**titles**).

set(row Number, column Number|String, value String) => Boolean

row Indice de la ligne (débute à 1)

column Indice de colonne ou nom

value Nouvelle valeur

Définie la valeur à la position indiquée de la table. Vous pouvez supprimer une ligne à l'aide de `table.set(row,0,'Delete')`. Pour les tables DBF et SQL, vous pouvez retirer la marque de suppression avec `table.set(row,0,' ')`. Les tables DBF ne retirent pas les enregistrements supprimés. Les tables SQL les retirent lors du rafrachissement (qui n'est pas nécessairement immédiat). Les tables tableur les retirent immédiatement.

setN(row Number, column Number|String, value Number) => Boolean

row Indice de la ligne (débute à 1)

column Numéro ou nom de colonne

value Nouvelle valeur

Définie la donnée numérique (number) à la posiiton indiquée dans la table. Les nombres sont usuellement convertis en texte pour le stockage dans la base de données.

setRow(row Number, data Record) => Boolean

row Indice de ligne

data Valeurs

Définie les valeurs dans la table. Renvoie **true** en cas de succès.

setTitle(column Number, title String) => Boolean

column Indice de colonne

title Nouveau titre

Change le titre de la colonne (son nom). Ne fonctionne que pour les tables de type tableur (ASCII et en mémoire).

title(column Number) => Number

column Indice de colonne

Renvoie le titre d'une colonne (son nom). Le numéro de colonnes débute à 1. Vous pouvez trouver le nombre de colonnes à l'aide de la propriété **colCount**.

toString( ) => String

Renvoie le nom du fichier

type(column String|Number) => String|Array

column Nom de colonne

Pour un argument, renvoie le type de données correspondant dans la ligne. Pour plusieurs arguments, retourne un tableau de type. Les types possibles sont C, N, D, T pour caractères, nombre, date et heure (time).

width(column Number) => Number

column Indice de colonne

Renvoie la largeur en caractères de la donnée qui peut être stockée dans cette colonne. Les tables de type tableur n'ont pas de limite.

XML

```
include('xml.js')
```

Le language JavScript 1.6 comporte un analyseur XML compatible avec le standard E4X. Cette classe XML date d'avant E4X et peut être utilisée pour se substituer aux objets internes. Utilisation standard : XML.read(stream,allowed_tags)

Propriétés

cdata String

children Array

name String

params Record

Méthodes

XML.read(source Stream|String, dtd String, ignored String, start XML) => Boolean

dtd Liste des balises séparées par des virgules

ignored Liste des balises, séparées par des virgules, à ignorer (par défaut : `BR,P,B,I,C,TT,U,IMG,A`)

start Un objet XML existant cible destiné à recevoir les nouvelles balises XML

This is a static function. Usage: `XML.read(stream, 'html,head,body,p')`

find(type String, parameter String, value String, parameter2 String, value2 String) => Array

type	Classe Enfant
parameter	Nom du champ
value	Valeur du champ
parameter2	Nom du champ
value2	Valeur du champ

Renvoie un tableau de tous les objets enfants dont le nom (sensible à la casse) est équivalent au **type**

findChildren(type String, parameter String, value String) => Array

type	Classe Enfant. Si omis, recherche tous les enfants
parameter	Nom du champ. Si omis, n'effectue pas la comparaison de champ
value	Valeur du champ

(Déprécié) Renvoie un tableau de tous les objets enfant dont le nom (sensible à la casse) est équivalent au **type** et qui dispose de la valeur recherchée. Exemple : fn.find('parameter','optional','1') renvoie la liste de tous les paramètres optionnels pour le fichier XML qui génére ce document. Utilisez XML.find() pour récupérer la liste de tous les enfants.

get(name String) => String

name	Nom du paramètre

Renvoie la valeur du champs **name**. Si le champ est émis, renvoie une chaîne vide. Pour tester l'existance d'un champ, utilisez `params.has('name')`.

getChildren(type String) => Array

type	Classe Enfant

(Déprécié) Renvoie un tableau de tous les objets enfant dont le nom (sensible à la casse) est équivalent au **type.**

select(type String, test Function, opaque) => Array

type	Classe Enfant
test	Fonction de test
opaque	Paramètre passé à la fonction de test

Renvoie un tableau de tous les objets enfant pour lesquels la fonction **test()** retourne **true**. Exemple :

```
var x = XML.read('');
y = x.select(null,function (a,b) {return a.get('id') ==
b},55);
writeln(y);
```

sort(field String, descending Boolean)

field	Champs à trier
descending	Ordre inverse ? (false par défaut)

Trie le tableau des enfants, selon les valeurs du champ indiqué.

toStream(output Stream) => String

output Ajoute l'objet XML à un flux (stream).

toString( ) => String

Ajoute l'objet XML à un flux mémoire puis retourne une chaîne.

global

Propriétés

jsArguments Array Egalement disponible en system.arguments. La liste des arguments de la ligne de commande (comme chaînes)

Méthodes

attributes(filename) => String

Renvoie un objet décrivant le fichier spécifié. `{ attributes: String, size: Number, date: Date, creation: Date }` Sous Windows, **attributes** peut inclure "archive, compressed, directory, hidden, offline, readonly, system, temporary". Sous Unix, **attributes** peut inclure "directory, symlink, regular, chardev, blockdev, fifo, socket".

copyFile(source String, destination String) => Boolean

(Déprécié) Utilisez system.copy() à la place. Copie un fichier. Echoue si la destination existe.

crc32(text , seed) => Number

Calcule le CRC32 d'une chaîne texte. La "graine" (valeur) intiale peut être spécifiée. Les arguments peuvent être passés dans n'importe quel ordre.

decodeANSI(text String) returns String

Convertit une chaîne Windows-1252 (ANSI) en une chaîne JS (UCS-2).

decodeB64(in Stream|String, out Stream) returns Number

Décode les données (base-64, MIME) et envoie le résulat vers **out**. Renvoie la taille des données décodées.

decodeB64(text String) => String

Décode les données (base-64, MIME) et renvoie les données décodées.

decodeHTML(text String) => String

Décode les séquences HTML < > " & and &#XXX; et renvoie le résultat.

decodeURL(text String) => String

Décode une chaîne URL-encodé.

decodeUTF8(text String) => String

Convertit une chaîne UTF-8 en une chaîne JS (UCS-2)

encodeB64(text String) => String

Encode la chaîne en base-64 (MIME) et renvoie le résultat.

encodeB64(in Stream|String, out Stream) returns Boolean

Encode la chaîne en base-64 (MIME), envoie le résultat vers **out** et renvoie la taille des données encodées.

encodeHTML(text String) => String

Encode < > " & et les caractères Unicode (après 127) au format HTML (requis pour XML et HTML).

encodeURL(text String) => String

Encode une chaîne au format URL (%) : les caractères alphanumériques et de ponctuation ne sont pas affectés. Renvoie la chaîne résultante.

encodeUTF8(text String) => String

Convertit une chaîne JS (UCS-2) en une chaîne UTF-8

fileExists(file)

(Déprécié) Utilisez system.exists() à la place. Renvoie **true** si le fichier existe et est accessible en lecture.

jsBuildDate( ) => String

(Déprécié). Utilisez system.buildDate() à la place. Renvoie une chaîne contenant la date de compilation de l'interpréteur JavaScript.

jsDebug(address String) => Boolean

address Adresse du debogueur

(Déprécié) Utilisez system.debug() à la place. Connexion à un débogueur distant `jsDebug('127.0.0.1:1002')`

jsGC( ) => Null

(Déprécié) Utilise system.gc() à laplace. Lance le garbage collecteur (libére toutes les ressources qui peuvent l'être)

jsIsSafe( ) => Boolean

(Déprécié) Utilisez system.isSafe() à la place. Indique si l'interpréteur est en mode sécurisé.

jsOptions(options String) => String

options 'strict' ou 'werror' (traite les avertissements (warnings) comme des erreurs)

(Déprécié) Utilisez system.options() à la place. Bascule les options de l'interpréteur et renvoie une chaîne précisant les options (séparées par une virgule). Par défaut, **strict** est actif au démarrage.

jsRestart( )

(Déprécié) Utilisez system.restart() à la place. Fixe le drapeau "shouldStop" à true et informe l'interpréteur pour répéter l'exécution. Cette option est utilisé seulement dans le serveur Web XYKE.

jsSafeMode(code Number) => Number

code Mot de passe

(Déprécié) Utilisez system.safeMode() à la place. Fixe un drapeau dans l'interpreteur pour indiquer que l'accès aux fichiers doit être interdit lors de l'exécution d'une pièce de code douteuse. Le premier appel à jsSafeMode() renvoie un nombre pseudo-aléatoire et verrouille les accès aux fichiers. Le

second appel, pour déverrouiller l'accès aux fichiers , doit spécifier ce même nombre. En mode sécurisé, les emails sont également bloqués, seuls les flux mémoire et http:// peuvent être créés. Les fichiers et ports série ouverts avant le verrouillage restent accessibles.

jsShellExec(command String, dir String)

command Ligne de commande destinée à l'invite du système (shell)

dir Répertoire de travail

(Déprécié) Utilisez system.execute() à la place. Appel ShellExec() sous Windows ou system() sous UNIX.

jsShouldStop( )

(Dépécié) Utilisez system.shouldStop() à la place. Renvoie **true** si le script doit se terminer.

jsVersion(new version Integer) => String

new version Version de l'interperteur

(Déprécié) Utilisez system.version() à la place. Renvoie une chaîne contenant le numéro de version de l'interpréteur JavaScript utilisé. Le numéro de version par défaut est actuellement 1.7. Les codes de version sont définis ci-après :

```
JSVERSION_1_0     = 100,
JSVERSION_1_1     = 110,
JSVERSION_1_2     = 120,
JSVERSION_1_3     = 130,
JSVERSION_1_4     = 140,
JSVERSION_ECMA_3  = 148,
JSVERSION_1_5     = 150,
JSVERSION_1_6     = 160,
JSVERSION_1_7     = 170,
JSVERSION_DEFAULT = 0,
JSVERSION_UNKNOWN = -1
```

kbhit( ) => Boolean

Renvoie **true** si une entrée console est disponible (touche frappée).

listFiles(filespec)

filespec "*.*" par défaut

(Déprécié) Utilisez system.files() à la place. Renvoie un tableau de noms de fichiers. Les répertoires sont exclus. Les fichiers système (System), cachés (hidden) et dont le nom commence par un point (.) sont également exclus.

listFolders(filespec)

filespec "*.*" par défaut

(Déprécié). Utilisez system.folders() à la place. Renvoie un tableau de noms de répertoires. Les fichiers sont exclus. Les réperoires système (System), cachés (hidden) et dont le nom commence par un point (.) sont également exclus.

load(filename) => Boolean

Exécute un fichier JavaScript. JSDB recherche le fichier indiqué selon l'ordre indiqué ci-après :

1. Dans une archive ZIP ajoutée à la fin du fichier exécutable. (Ne fonctionne sous UNIX que si le fichier ZIP est dans le répertoire courant.)
2. Dans le répertoire courant
3. Dans le chemin des bibliothèques (library path). (A définir en ligne de commande avec l'option -path, répertoire de l'exécutable pa défaut)
4. JSDB ignore le path système et n'exécute pas de fichiers à partir de répertoire inhabituel.

Cet ordre est utilsé afin de vous permettre de surcharger les bibliothèques standard de JSDB (mais amène quelques risques de sécurité). En l'état, quelqu'un peut écrite un "xml.js" malicieux dans le répertoire courant en espérant que cette bibliothèque soit chargée par un script exécuté par un script ultérieurement. Vous devez également être prudent avec l'usage de la méthode system.cwd(). Comme pour run(), load() échoue si JSDB est en mode sécurisé.
Contrairement à run(), load() n'exécute le fichier source indiqué qu'une seule fois. Les fichiers seront rechargés s'ils ont été modifiés.

loadResource(name) => Stream

(Déprécié) Utilisez system.resource() à la place. Renvoie un flux (stream) lisible contenant les données de la ressource. Le fichier ressource peut être un ZIP lié à l'exécutable ou placé dans le même répertoire que l'interpréteur.

moveFile(source String, destination String)

(Déprécié) Utilisez system.move() à la place. Déplace un fichier. Echoue si le fichier destination existe.

openBrowser(file)

file URL

(Déprécié) Utilisez system.browse() à la place.

print, write( )

Envoie les arguements vers stdout (en appelant toString() si nécessaire).

printReport(format Stream, values Record) => String

(Déprécié) Lit le format **format** et remplace {nom} avec la valeur correspondante de la paire nom-valeur de l'enregistrement (Record).

println, writeln( )

Comme print() et write() avec ajout d'un CR/LF à la fin.

quit,exit( )

(Déprécié) Utilisez system.quit() ou system.exit() à la place. Fixe le drapeau "shouldStop" à **true**. L'éxécution ne s'interrompte pas immédiatement.

readln(end String) => String

end Marqueur de fin de ligne par défaut "\n"

Lit une ligne à partir de stdin (en mode console seulement).

regGetKey(section , name)

section "hkey_local_machine\SOFTWARE\..."

name Nom de la clef

(Déprécié) Utilisez system.getKey() à la place. Renvoie une chaîne contenant les données de la base de registre associées à la clef.

regSetKey(section , name , value)

section "hkey_local_machine\SOFTWARE\..."

name Nom de la clef

value Valeur de la clef

(Déprécié) Utilisez system.setKey() à la place. Définit le contenu de l'entrée de la base de registre associée à la clef spécifiée. Renvoie **true** en cas de succès.

run(filename)

Exécute un fichier JavaScript. Run utilise les mêmes régles de recherche que load().

run(text Stream, filename String, line Number)

Exécute un script JavaScript stocké dans un flux (stream). Les méthodes run() et load() sont équivalentes.

sleep(time Number)

time millisecondes

(Déprécié) Utilisez system.sleep() à la place. Suspend le process et transfert le contrôle d'exécution aux autres programmes.

splitURL(URL String) => Array

Analyse un URL de la forme `service://user:password@host/file?query` et renvoie le tableau de chaînes : `[service, user, password, host, file, query]`

stripWhitespace(text String) => String

Renvoie la chaîne spécifiée sans les espaces initiaux et finaux.

testCompile(code) returns String

code Code JS à tester

(Déprécié) Utilisez system.compile() à la place. Renvoie le message d'erreurs en cas de problème de compilation.

system

Propriétés

arguments	Array	Liste des arguments de la ligne de commande (tableau de chaînes)
buildDate	String	Date de compilation de l'interpréteur
release	Number	Version de mise à jour (release) de JSDB, exemple "1.3".

stdin	Stream	Utilisez `system.stdin` pour lire directement à partir de la console. Le flux n'est pas bufferisé.
stdout	Stream	Utilisez `system.stdout` pour écrire directement sur la console. Le flux n'est pas bufferisé. L'interpréteur envoie toujours ses messages d'erreur vers stdout (jamais vers stderr).
version	Number	A diviser par 100 pour obtenir la version du langage. JavaScript 1.5 est représenté par 150.

Méthodes

browse()(file)

file URL

Ouvre le navigateur Web sur la console et affiche l'URL spécifié.

buildDate( ) returns String

Renvoie une chaîne contenant la date de compilation de l'interpréteur JavaScript.

compile(code) => String

code Code JS à tester

Renvoie un message d'erreur s'il y a un problème de compilation.

copy(source String, destination String) => Boolean

Copie un fichier. Echoue si le fichier de destination existe.

debug(address String) => Boolean

address Adresse du débogueur

Se connecte au débogueur distant `system.debug('127.0.0.1:1002')`

execute(command String, dir String)

command Ligne de commande (shell)

dir Répertoire de travail

Appelle ShellExec() sous Windows ou system() sous UNIX.

exists(file)

Renvoie **true** si le fichier existe et s'avère lisible.

exit( )

Fixe le drapeau "shouldStop" à **true**. L'exécution ne s'interrompt pas immédiatemment.

files(filespec)

filespec "*.*" par défaut

Renvoie un tableau des noms de fichiers. Les répertoires sont exclus. Les fichiers système (System), cachés (hidden) et dont le nom commence par un point (.) sont exclus.

folders(filespec)

filespec "*.*" par défaut

Renvoie un tabeau des noms de répertoires. Les fichiers sont exclus. Les

répertoires système (System), cachés (hidden) et dont le nom commence par un point (.) sont exclus.

gc( ) returns Null

Active la garbage collector (libère toutes les ressources qui peuvent l'être).

getKey(section , name)

section "hkey_local_machine\SOFTWARE\..."

name Nom de la clef

Renvoie une chaîne comprenant les données de la base de registre associées à la clef (Windows seulement).

getenv(name String) => String

name Nom de la variable

Renvoie la valeur de la variable d'environnement indiquée.

help( ) => String

Renvoie les informations relatives à la version de l'interpréteur, exemple "JSDB 1.1 ...".

isSafe( ) => Boolean

Indique si l'interpréteur est en mode sécurisé.

mkdir(directory)

Crée un répertoire (de façon récursive : construit également son chemin d'accès si nécessaire). Renvoie toujours **true**.

move(source String, destination String, replace Boolean)

replace **true** pour écraser les fichiers existants

Déplace un fichier.

options(options String) => String

options 'strict', 'werror' (traite les avertissements (warnings) comme des erreurs), 'atline', 'xml' (active E4X au sein de load() et run())

Bascule les options de l'interpréteur et renvoie une chaîne décrivant les options en cours (séparées par des virgules). Par défaut, **strict** est actif au démarrage.

print, write( )

Envoie les arguements vers stdout (en appelant toString() si nécessaire).

println, writeln( )

Comme print() et write() avec ajout d'un CR/LF à la fin.

quit( )

Fixe le drapeau "shouldStop" à **true**. L'éxécution ne s'interrompt pas immédiatement.

readln(end String) => String

end Marqueur de fin de ligne par défaut "\n"

Lit une ligne à partir de stdin (en mode console seulement).

resource(name) returns Stream

Renvoie un flux (stream) lisible contenant les données de la ressource. Le fichier ressource peut être un ZIP lié à l'exécutable ou placé dans le même répertoire que l'interpréteur.

restart( )

Fixe le drapeau "shouldStop" à true et informe l'interpréteur pour répéter l'exécution.

safeMode(code Number) returns Number

code password

Fixe un drapeau dans l'interpreteur pour indiquer que l'accès aux fichiers doit être interdit lors de l'exécution d'une pièce de code douteuse. Le premier appel à jsSafeMode() renvoie un nombre pseudo-aléatoire et verrouille les accès aux fichiers. Le second appel, pour déverrouiller l'accès aux fichiers , doit spécifier ce même nombre. En mode sécurisé, les emails sont également bloqués, seule les flux mémoire et http:// peuvent être créés. Les fichiers et ports série ouverts avant le verrouillage reste accessible.

setKey(section , name , value)

section "hkey_local_machine\SOFTWARE\..."

name Nom de la clef

value Valeur de la clef

Définit le contenu de l'entrée de la base de registre associée à la clef spécifiée. Renvoie **true** en cas de succès (Windows seuleument).

setcwd(directory) returns String

Définit le répertoire de travail courant (si **directory** n'est pas vide) et renvoie le répertoire de travail précédent. [ndlr : de fait setcwd() permet de connaître le répertoire de travail actuel.]

shouldStop( )

Renvoie **true** si le script doit se terminer.

sleep(time Number)

time millisecondes

Suspend le process et transfert le contrôle d'exécution aux autres programmes.

stripWhitespace(text String) returns String

Renvoie la chaîne spécifiée sans les espaces initiaux et finaux