JScript/VBScript + Win32 API: DynamicWrapperX

Par Yuri Popov - Traduction française: omen999

Date de publication : 21 janvier 2009

Accéder aux API Win32 en VBSCript ou JScript grâce à
DynamicWrapperX v1.0
License: Freeware

I - Introduction....3
II - Enregistrement du composant....3
III - Liste des méthodes exposées....3
IV - [1].Register(DllName, nomFonc [, i = ParamTypes] [, r = RetValType])....4
V - Types de paramètres d'entrée et de valeurs de retour....5
VI - Type de paramètres de sortie....5
VII - A propos des chaines de caractères....6
VIII - [2].RegisterCallback (FuncRef [, i = ParamTypes] [, r = RetValType])....7
IX - Autres méthodes (gestion des variables)....8
[3]. NumGet (Address [, Offset] [, Type])....8
[4]. NumPut (Var, Address [, Offset] [, Type])....9
[5]. StrPtr (Var [, Type])....9
[6]. StrGet (Address [, Type])....9
[7]. Space (Count [, Char])....9
X - Téléchargement....10

Copyright ® 2009 Yuri Popov. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

I - Introduction

DynamicWrapperX est un composant ActiveX (serveur COM) écrit dans le but d'étendre et d'améliorer les possibilités du composant initial Dynawrap. Il permet aux langages JScript et VBScript d'appeler les fonctions exportées par les fichiers bibliothèques DLL, en particulier les fonctions de l'API Windows. Ce composant n'est pas une adaptation du code original de Dynawrap et a été écrit en partant de zéro en assembleur GoAsm. Testé sous Windows XP SP1 et Windows 98 SE.

Les améliorations et modifications sont les suivantes:

- Possibilité d'inscrire le composant dans la base de registres uniquement pour l'utilisateur courant. Cela peut être utile si l'utilisateur n'a pas les droits d'administrateur.
- Ajout de nouveaux types de paramètres d'entrée et de valeurs de retour.
- Ajout des paramètres de sortie.
- Gestion identique des chaînes de caractères dans JScript et VBScript.
- Support des fonctions de rappel (callback) utilisées notamment par les fonctions d'énumération de l'API Windows comme EnumWindows. (voir les examples ci-dessous)
- Suppression d'un bug sous Win98. Il n'est plus nécessaire d'instancier un objet pour chaque fonction enregistrée.
- Ajout de méthodes utiles (AMHA) de gestion de variables.

P.S. Bien que le composant DynamicWrapperX ait été testé soigneusement, l'auteur ne peut garantir un fonctionnement sans défaut. L'usage se fait aux risques et périls de son utilisateur.

II - Enregistrement du composant

Pour inscrire le composant, deux options sont possibles :

pour tous les utilisateurs.
```
regsvr32.exe <chemin-du-composant>\ dynwrapx.dll
```

pour l'utilisateur courant.
```
regsvr32.exe /i <chemin-du-composant>\ dynwrapx.dll
```

Si dynwrapx.dll réside dans le répertoire System32, System, Windows, dans le répertoire courant, ou dans un de ceux qui sont énumérés dans la variable d'environnement PATH, le chemin peut être omis. Si le composant est enregistré pour l'utilisateur courant, il ne sera disponible que pour cet utilisateur.

Pour désinscrire le composant, procéder comme suit :

pour tous les utilisateurs.
```
regsvr32.exe /u <chemin-du-composant>\ dynwrapx.dll
```

pour l'utilisateur courant.
```
regsvr32.exe /u /i <chemin-du-composant>\ dynwrapx.dll
```

Sous Windows 98, vous devez spécifier le chemin complet vers regsvr32.exe (Dossier System). En outre, l'inscription pour l'utilisateur courant sera inopérant.

III - Liste des méthodes exposées

- [1] Register(DllName, nomFonc [, i = ParamTypes] [, r = RetValType])

Copyright ® 2009 Yuri Popov. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

- Référence une fonction de la DLL comme une méthode de l'objet.
- [2] RegisterCallback(FuncRef [, i = ParamTypes] [, r = RetValType])
  - Référence une fonction du script comme fonction de rappel (callback)
- [3] NumGet(Address [, Offset] [, Type])
  - Lit une valeur numérique en mémoire à l'adresse donnée <Adresse>+<Offset>
- [4] NumPut(Var, Address [, Offset] [, Type])
  - Ecrit en mémoire le contenu d'une variable numérique <Var>à l'adresse donnée <Adresse>+<Offset>
- [5] StrPtr(Var [, Type])
  - Renvoie le pointeur vers la chaîne <Var>
- [6] StrGet(Address [, Type])
  - Lit une chaine de caractères à partir d'une adresse donnée (pointeur) <Adresse>.
- [7] Space(Count [, Char])
  - Renvoie un pointeur vers une chaine de <Count> caractères <Char>.

IV - [1].Register(DllName, nomFonc [, i = ParamTypes] [, r = RetValType])

Cette méthode référence les fonctions exportées par la bibliothèque <DllName> comme une méthode de l'objet. <nomFonc> doit correspondre à une fonction existante de la dll. Sa syntaxe est semblable à celle du composant original Dynawrap. Toutefois le paramètre f est supprimé et sera tout simplement ignoré.
i = valeurs d'entrée attendues par la fonction, r = valeur de retour.
(voir ci-dessous pour les différents types supportés)

Code JScript

```
DX = new ActiveXObject ("DynamicWrapperX");                    // instancie l'objet
DX.Register ("user32.dll", "MessageBoxW", "i=hwwu", "r=l"); // référence la fonction de la dll
res = DX.MessageBoxW (0, "Bonjour, monde!", "Test", 4);     // appel de la fonction
```

Code VBScript

```
set DX = CreateObject ("DynamicWrapperX")                  ' instancie l'objet
DX.Register "user32.dll", "MessageBoxW", "i=hwwu", "r=l"   ' référence la fonction de la dll
res = DX.MessageBoxW(0, "Bonjour, monde!", "Test", 4)      ' appel de la fonction
```

Le nom de la bibliothèque ***DllName*** sans indication de répertoire signifie une recherche par nom, en premier lieu dans la mémoire du process parmi les bibliothèques déjà chargées (par ordre de leur chargement), puis sous Win XP l'ordre de recherche sur le disque sera le suivant:

1. Répertoire de l'application (dans ce cas, soit wscript.exe ou cscript.exe)
2. Le répertoire courant
3. System32
4. System
5. Windows
6. Répertoires de la variable d'environnement PATH

Si la bibliothèque est un fichier comportant l'extension .dll, cette extension peut être omise, ainsi dans l'exemple ci-dessus on pourrait simplement écrire "user32". Si la bibliothèque est un fichier sans extension, il faudra ajouter un point à la fin du nom comme par exemple: "mylib."

Le nom de la fonction ***nomFunc*** peut présenter des variantes selon que vous souhaitez la version Unicode ou son équivalent ANSI. Cela ne concerne normalement que les fonctions utilisant des paramètres de type chaine. Par exemple, la fonction décrite dans la documentation telle que MessageBox, est effectivement présente dans user32.dll sous deux formes: MessageBoxW (pour les chaînes de caractères Unicode) et MessageBoxA (pour les chaines de caractères classiques ANSI). Conformément à l'usage, le mode de recherche classique des fonctions a été maintenu : par exemple, si vous définissez le nom MessageBox et que cette fonction n'est pas trouvée dans user32.dll, la recherche se fera également pour la fonction MessageBoxA. Les noms de fonctions Unicode imposent en revanche de préciser la lettre W.

Copyright ® 2009 Yuri Popov. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Le paramètre i=***ParamTypes*** peut être omis quand les fonctions ne réclament pas de paramètres d'entrée sans qu'il soit nécessaire de conserver la virgule.

La valeur de retour r=***RetValType*** peut également être omise, quelle que soit la valeur renvoyée.

Code JScript
```
DX = new ActiveXObject("DynamicWrapperX");
DX.Register ("kernel32", "GetCommandLine", "r=s"); // cette fonction n'a pas de paramètres.
Cmdline = DX.GetCommandLine ();                    // ligne de commande qui a démarré le process
WScript.Echo (cmdline);
```

Code VBScript
```
Set DX = CreateObject("DynamicWrapperX")
DX.Register "kernel32", "Beep", "i=uu"     ' la valeur retournée par la fonction Beep n'est pas
 utile
DX.Beep 800, 1000                          ' son à partir du haut-parleur du PC
```

V - Types de paramètres d'entrée et de valeurs de retour

- **l** - entier signé 32-bits - LONG, INT, LPARAM, LRESULT etc
  compris entre : -2147483648 ... 2147483647;
- **u** - entier non signé 32-bits - ULONG, UINT, DWORD, WPARAM, etc
  compris entre : 0 ... 4294967295;
- **h** - valeur handle - HANDLE, HWND, HMODULE, HINSTANCE, HICON, etc
  compris entre : -2147483648 ... 4294967295;
- **p** - valeur identique à 'u' pour les nombres, mais peut également pointer vers une interface IDispatch * ou une chaine;
- **n** - entier signé 16-bits - SHORT
  compris entre : -32768 ... 32767;
- **t** - entier non signé 16-bits - USHORT, WORD, WCHAR, OLECHAR,
  compris entre : 0 ... 65535;
- **c** - entier signé 8 bits - CHAR
  compris entre: -128 ... 127;
- **b** - entier non signé 8 bits - UCHAR, BYTE,
  compris entre: 0 ... 255;
- **f** - réel virgule flottante simple précision (32 bits) - FLOAT;
- **d** - réel virgule flottante double précision (64 bits) - DOUBLE;
- **w** - chaine Unicode - BSTR, LPWSTR, LPOLESTR, OLECHAR *, WCHAR *, etc...;
- **s** - chaîne codage ANSI / Windows (code page par défaut) - LPSTR, LPCSTR, char *, etc...;
- **z** - chaîne codage OEM / DOS (code page par défaut) - LPSTR, LPCSTR, char *, etc ... ;

VI - Type de paramètres de sortie

Copyright ® 2009 Yuri Popov. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Type de paramètres de sortie

- **L** - pointeur vers son adresse en mémoire - LONG *, LPLONG, etc;
- **H** - idem - HANDLE*, PHANDLE, LPHANDLE, ...;
- **U** - idem - ULONG*, LPDWORD, ...;
- **P** - idem;
- **N** - idem - SHORT *;
- **T** - idem - USHORT *, LPWORD, WCHAR *, OLECHAR *, ...;
- **C** - idem - CHAR *, ...;
- **B** - idem - UCHAR *, LPBYTE, ...;
- **F** - idem - FLOAT *, PFLOAT;
- **D** - idem - DOUBLE *, PDOUBLE;
- **W** - pointeur vers chaine Unicode;
- **S** - pointeur vers chaine ANSI;
- **Z** - pointeur vers chaine OEM/DOS.

Les paramètres de sortie sont ceux dont la valeur doit être modifiée par la fonction appelée. Dans certains cas, vous devez passer à la fonction API le pointeur d'un nombre plutôt que le nombre lui-même. Vous n'avez pas besoin de rechercher le pointeur avant de le transmettre, la conversion sera faite automatiquement pour les paramètres définis en lettres capitales. Il suffit de passer une variable numérique ordinaire et la fonction API recevra en réalité le pointeur vers la valeur conservée par la variable (équivaut à l'argument ByRef). Pour les sorties chaines, voir ci-dessous. Particularité de JScript: Pour utiliser un nombre comme paramètre de sortie, il devra être préalablement déclaré avec l'opérateur new. Exemple : a = new Number(0); Pour les chaînes de caractères (en entrée comme en sortie) cette déclaration est inutile et même contre-indiquée, car l'objet String ne contient pas un pointeur vers une chaine normale mais vers un «flux versionné», notion obscure que je ne maitrise pas encore…

VII - A propos des chaines de caractères

JScript et VBScript utilisent des chaines au format BSTR. Il s'agit d'une chaîne de caractères Unicode, chaque caractère étant codé sur 2 octets. La chaine est terminée par un caractère spécial constitué de 2 octets nuls. En outre, le premier caractère est précédé par une valeur numérique codée sur quatre octets contenant la longueur de la chaîne en octets (à l'exclusion du caractère terminal). Le pointeur des variables chaines est l'adresse du premier caractère effectif de la chaine (c'est-à-dire sans prendre en compte l'en-tête de quatre octets).

Copyright ® 2009 Yuri Popov. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

La gestion d'un paramètre de type chaîne peut se faire de trois manières différentes:

1. Paramètres d'entrée chaine: **w**, **s**, **z**. Pour les types s et z la chaine est copiée et convertie en fonction de l'encodage. La fonction API reçoit alors un pointeur vers cette copie. Après l'exécution de la fonction, cette copie est supprimée. Dans le cas du type w c'est le pointeur vers la chaine originale qui est transmis.
2. Paramètres de sortie chaine: **W**, **S**, **Z**. Dans ce cas et pour les trois types, un pointeur vers la chaine originale est passé, mais S et Z sont d'abord convertis (sur place) selon la page code correspondante. Au retour de la fonction, le contenu des chaines définies par S et Z sont reconverties en Unicode avec réajustement de leur taille, celle de la variable W est simplement recalculée. La longueur (en octets) de la chaine est enregistrée dans l'en-tête. Cette dernière opération est nécessaire pour éviter les bogues, par exemple lors de la concaténation avec d'autres chaines. Si la chaine est destinée à recevoir des données de la fonction appelée, veillez à lui donner une taille suffisante.
3. Index: **p**. C'est la façon la plus simple. Le pointeur de la chaine originale est transmis sans conversion. En conséquence, si la fonction écrit dans cette chaine, les données seront conservées sous leur forme d'origine sans conversion ou ajustement de la longueur. La chaine devient une simple mémoire tampon pour stocker les données. En apparence, cette option ressemble à l'option w mais ce n'est pas exact. La différence tient dans le fait que l'option p accepte non seulement les variables chaînes, mais aussi les variables numériques.

Retourner une chaine avec **p** revient à récupérer une variable numérique contenant un pointeur vers la chaine. Retourner une chaine avec **w**, **s** ou **z** revient à récupérer une variable chaine contenant un pointeur vers une copie de la chaine retournée par la fonction API. Les chaines retournées avec les paramètres **s** et **z** sont copiées avec conversion préalable. Les chaines originales ne sont pas libérées car il n'est pas certain que cette libération soit sûre. Si les éventuelles fuites mémoire vous préoccupent, privilégiez l'utilisation du paramètre **p** autant que possible pour le retour des chaines.

Les fonctions API qui réclament des paramètres chaines existent généralement en deux versions - par exemple, MessageBoxA et MessageBoxW. Il semble plus logique d'utiliser dans les scripts les versions Unicode (celles dont le nom se termine par W) puisqu'on évite alors tout travail de conversion à partir et vers Unicode.

En ce qui concerne Windows 98, les deux moteurs de script utilisent également des chaînes Unicode, mais toutes les fonctions API dont les noms se terminent en W ne sont pas fonctionnelles. Beaucoup d'entre elles sont tout simplement fictives et retournent 0 sans rien faire. Par exemple, MessageBoxW fonctionne mais pas lstrcmpiW alors même qu'elle est présente dans la bibliothèque kernel32.

VIII - [2].RegisterCallback (FuncRef [, i = ParamTypes] [, r = RetValType])

Cette méthode permet de référencer un pointeur vers une fonction du script en vue de son exploitation par certaines fonctions de l'API. Par exemple, la fonction API EnumWindows utilise une fonction de rappel (callback). Elle énumère les fenêtres existantes, chaque fenêtre étant traitée par une fonction dont l'adresse est transmise en paramètre. Après chaque appel, la fonction principale attend la valeur renvoyée par la fonction callback. Si 1, l'énumération se poursuit, si 0 elle s'arrête.

En JScript et VBScript, une fonction de script n'est pas directement disponible sous forme de pointeur car ces fonctions sont des objets et leurs valeurs pointent vers une interface de type IDispatch. Aussi, la référence à une fonction de script est initialement transmise à la méthode RegisterCallback. La fonction API qui utilise une fonction callback obtiendra alors un pointeur vers une procédure interne de dynwrapx.dll qui renverra l'appel à la fonction du script et lira la valeur renvoyée. Il n'existe que 16 procédures « proxy » disponibles, de sorte que le nombre maximum de fonctions callback dans le script est également limité à 16.

Avec JScript, le nom de la fonction (sans parenthèses) servira directement de référence alors qu'avec VbScript, vous devrez utiliser la fonction GetRef. Outre la référence à la fonction, vous aurez à préciser les éventuels paramètres d'entrée ainsi que la valeur de retour exactement comme pour la méthode Register (les paramètres en minuscules sont seuls utilisables)

Code JScript

```
DX = new ActiveXObject("DynamicWrapperX");
DX.Register("user32", "EnumWindows", "i=pl");
```


Copyright ® 2009 Yuri Popov. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Code JScript

```
DX.Register("user32", "GetWindowTextW", "i=hWl");          //  option Unicode
// DX.Register("user32", "GetWindowText", "i=hSl");        // option ANSI
pCbkFunc = DX.RegisterCallback(CbkEnumWin, "i=hl", "r=l"); // enregistrement CbkEnumWin
n=0, m=0, WinList = "";
Title = DX.Space (256);                                     // le buffer de réception du texte.
DX.EnumWindows(pCbkFunc, 0);
WScript.Echo("Toutes les fenêtres:" + m + "\ n Titres:" + n + "\ n \ n" + WinList);

// la fonction callback

fonction CbkEnumWin (hwnd, lparam)
{
   DX.GetWindowTextW (hwnd, Title, 256);
   // DX.GetWindowText (hwnd, Title, 256); // option ANSI
   if (Title.length> 0) {                  // si le titre existe, on l'ajoute à la liste
 WinList += hwnd + "\ t" + Title + "\ n";
     n++;
   }
   ++m;
   return 1                              // l'énumération continue
}
```

Code VBScript

```
Set DX = CreateObject("DynamicWrapperX")
DX.Register "user32", "EnumWindows", "i = pl"
DX.Register "user32", "GetWindowTextW", "i=hWl"     ' option Unicode.
'DX.Register "user32"," GetWindowText ","i=hSl"     ' option ANSI.
Set Ref = getRef("CbkEnumWin")                       ' référence de la fonction.
pCbkFunc = DX.RegisterCallback(Ref, "i=hl", "r=l") ' enregistrement CbkEnumWin
n = 0: m = 0: WinList = ""
Titre = Space (256)                                  ' buffer de réception du texte
DX.EnumWindows pCbkFunc, 0
WScript.Echo "Toutes les fenêtres:" & m & vbCrLf & "Titres:" & n & _
                    vbCrLf & vbCrLf & WinList

' la fonction callback

Function CbkEnumWin (hwnd, lparam)
   DX.GetWindowTextW hwnd, titre, 256
   'DX.GetWindowText hwnd, titre, 256" Option ANSI.
   If Len (Titre)> 0 Then    ' si le titre existe, on l'ajoute à la liste
      WinList = WinList & hwnd  & vbTab & Titre & vbCrLf
      n = n +1
   End if
   m = m +1
   CbkEnumWin = 1 ' l'énumération continue
End Function
```

IX - Autres méthodes (gestion des variables)

Note: Les paramètres entre crochets sont facultatifs mais vous ne pouvez pas les omettre si le suivant est présent.

[3]. NumGet (Address [, Offset] [, Type])

Lit le contenu de la mémoire et l'affecte à une variable numérique
Address - adresse de base.
Offset - décalage (en octets), positif ou négatif: utile dans les cycles de lecture/écriture de nombres consécutifs, par défaut 0.
Type - le type de données à lire: même syntaxe que pour la méthode Register, mais uniquement les lettres minuscules, par défaut - l.
Seules les lettres minuscules peuvent être utilisées. Le nombre est retourné dans la valeur renvoyée par la méthode.

Copyright ® 2009 Yuri Popov. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

[4]. NumPut (Var, Address [, Offset] [, Type])

Ecrit une variable numérique dans la mémoire.
Var - soit une valeur littérale numérique ou la variable contenant ladite valeur Pour le reste syntaxe identique à NumGet.
Renvoie l'adresse disponible immédiatement après la valeur écrite.

Dans les deux dernières méthodes, Address peut être soit une valeur numérique soit une chaine. Dans ce dernier cas, le pointeur de chaine sert d'adresse de base. Cela vous permet d'utiliser la chaine comme un tampon pour les données à placer à cet endroit comme des tableaux, ou des données structurées, etc

[5]. StrPtr (Var [, Type])

Retourne un pointeur vers une chaîne de caractères.
Var - nom de la variable chaine.
Type - facultatif. Les valeurs possibles: w (par défaut), s, z. En cas de valeur s ou z , la chaine est préalablement convertie au même emplacement.

[6]. StrGet (Address [, Type])

Retourne une chaine lue à partir d'une adresse et renvoie une copie de celle-ci.
Address peut être soit sous la forme numérique ou une variable chaîne.
Type - facultatif. Les valeurs possibles: w (par défaut), s, z. En cas de valeur s ou z, une conversion au format Unicode est opérée.

[7]. Space (Count [, Char])

Retourne une variable chaine contenant un pointeur vers une chaine BSTR de longueur donnée.
Count - le nombre de caractères
Char - facultatif: le caractère qui remplira la chaine. Par défaut il s'agit du caractère espace comme dans la fonction VBScript Space(). Pour remplir la ligne de zéros binaires, définissez Char comme une chaine vide ("").

Code JScript

```
DX = new ActiveXObject("DynamicWrapperX");
str = "Bonjour le monde!";
codes = "";
for(i=0; i < str.length; ++i)
  codes += DX.NumGet(str, i*2, "t") + " ";
  WScript.Echo("codes de caractères:\n" + codes);
len = str.length;
buf = DX.Space(len);

for(i=0, j=len-1; i < len; ++i, --j) {
  code = DX.NumGet(str, i*2, "t");
  DX.NumPut(code, buf, j*2, "t");
}
WScript.Echo("s:\n" + buf);
ptr = DX.StrPtr(str);
WScript.Echo("s: " + ptr);
ptr = DX.StrPtr(str, "z");
str1 = DX.StrGet(ptr, "z");
WScript.Echo("s:\n" + str1);
```

Code VBScript

```
Set DX = CreateObject("DynamicWrapperX")
str = "Bonjour le monde!"
' Lecture de la mémoire. Lire des codes dans une chaîne.
strlen = Len(str)
```


Copyright ® 2009 Yuri Popov. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Code VBScript

```
codes = ""
For i = 0 To strlen-1
   codes = codes & DX.NumGet(str, i*2, "t") & " " ' i multiplié par 2 car la chaine est Unicode
 et
Next                            ' l'adresse en octets

WScript.Echo "codes des caractères:" & vbCrLf & codes

' Lecture/écriture. La chaine est enregistrée dans l'ordre inverse de ses caractères.

buf = Space (strlen)                                ' tampon pour l'écriture.
j = strlen-1    ' strlen-1 est le dernier symbole utile (moins le caractère terminal)
For i = 0 To strlen-1
   code = DX.NumGet (str, i * 2, "t")               ' lecture de gauche à droite
   DX.NumPut code, buf, j * 2, "t"                  ' écriture de droite à gauche
   j = j-1
next

WScript.Echo "Chaine inversée :" & vbCrLf & buf

' Divers
ptr = DX.StrPtr (str)                               ' renvoie un pointeur vers la chaîne qui reste
 en Unicode.
WScript.Echo "Adresse de la chaine :" & ptr
ptr = DX.StrPtr (str, "z")                          ' renvoie un pointeur vers la chaine
 préalablement convertie en OEM / DOS
str1 = DX.StrGet (ptr, "z")                         ' renvoie une chaine à partir d'un pointeur
 réencodée en Unicode
WScript.Echo "Chaine réencodée en Unicode :" & vbCrLf & str1
```

X - Téléchargement

Le composant peut être téléchargé ici: **dynwrapx1_00_eng.zip** (version 1.0.0.0 du 17.10.2008, taille archive 14 024 octets).

Vous pouvez également poser vos questions et laisser vos commentaires et suggestions sur le forum russe: **http://forum.script-coding.com/viewtopic.php?id=7328**
où en français ici : **Commentez**

Copyright ® 2009 Yuri Popov. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.