

Caractéristiques

[de VBScript](#)
[de VBA non incluses dans VBScript](#)
[de VBScript non incluses dans VBA](#)
[de l'exécutable Microsoft Scripting](#)

Liste des mots clés

[Abs, fonction](#)
[Addition \(+\), opérateur](#)
[And, opérateur](#)
[Array, fonction](#)
[Asc, fonction](#)
[Affectation \(=\), opérateur](#)
[Atn, fonction](#)
[Call, instruction](#)
[CBool, fonction](#)
[CByte, fonction](#)
[CCur, fonction](#)
[CDate, fonction](#)
[CDBl, fonction](#)
[Chr, fonction](#)
[CInt, fonction](#)
[Class, objet](#)
[Class, instruction](#)
[Clear, méthode](#)
[CLng, fonction](#)
[Couleur, constantes](#)
[Comparaison, constantes](#)
[Concatenation \(&\), opérateur](#)
[Const, instruction](#)
[Cos, fonction](#)
[CreateObject, fonction](#)
[CSng, fonction](#)
[CStr, fonction](#)
[Date/Heure, constantes](#)
[Format de date, constantes](#)
[Date, fonction](#)
[DateAdd, fonction](#)

[DateDiff, fonction](#)
[DatePart, fonction](#)
[DateSerial, fonction](#)
[DateValue, fonction](#)
[Day, fonction](#)
[Description, propriété](#)
[Dictionary, objet](#)
[Dim, instruction](#)
[Division \(/\), opérateur](#)
[Do...Loop, instruction](#)
[Empty](#)
[Eqv, opérateur](#)
[Erase, instruction](#)
[Err, objet](#)
[Eval, fonction](#)
[Execute, méthode](#)
[Execute, instruction](#)
[Exit, instruction](#)
[Exp, fonction](#)
[Exponentiation \(^\), opérateur](#)
[False](#)
[FileSystemObject, objet](#)
[Filter, fonction](#)
[FirstIndex, propriété](#)
[Fix, fonction](#)
[For...Next, instruction](#)
[For Each...Next, instruction](#)
[FormatCurrency, fonction](#)
[FormatDateTime, fonction](#)
[FormatNumber, fonction](#)
[FormatPercent, fonction](#)
[Function, instruction](#)
[GetObject, fonction](#)
[GetRef, fonction](#)
[Global, propriété](#)
[Hex, fonction](#)
[HelpContext, propriété](#)

[HelpFile, propriété](#)
[Hour, fonction](#)
[If...Then...Else, instruction](#)
[IgnoreCase, propriété](#)
[Imp, opérateur](#)
[Initialize, événement](#)
[InputBox, fonction](#)
[InStr, fonction](#)
[InStrRev, fonction](#)
[Int, fonction](#)
[Division entière \(\\), opérateur](#)
[Is, opérateur](#)
[IsArray, fonction](#)
[IsDate, fonction](#)
[IsEmpty, fonction](#)
[IsNull, fonction](#)
[IsNumeric, fonction](#)
[IsObject, fonction](#)
[Join, fonction](#)
[LBound, fonction](#)
[LCase, fonction](#)
[Left, fonction](#)
[Len, fonction](#)
[Length, propriété](#)
[LoadPicture, fonction](#)
[Log, fonction](#)
[LTrim, fonction](#)
[Match, objet](#)
[Matches, collection](#)
[Mid, fonction](#)
[Minute, fonction](#)
[Divers, constantes](#)
[Mod, opérateur](#)
[Month, fonction](#)
[MonthName, fonction](#)
[MsgBox, constantes](#)
[MsgBox, fonction](#)

[Muliplication, opérateur \(*\)](#)
[Négation, opérateur \(-\)](#)
[Not, opérateur](#)
[Now, fonction](#)
[Nothing](#)
[Null](#)
[Number, propriété](#)
[Oct, fonction](#)
[On Error, instruction](#)
[Opérateurs, priorité](#)
[Option Explicit, instruction](#)
[Or, opérateur](#)
[Pattern, propriété](#)
[Private, instruction](#)
[PropertyGet, instruction](#)
[PropertyLet, instruction](#)
[PropertySet, instruction](#)
[Public, instruction](#)
[Raise, méthode](#)
[Randomize, instruction](#)
[ReDim, instruction](#)
[RegExp, objet](#)
[Rem, instruction](#)
[Replace, fonction](#)
[Replace, méthode](#)
[RGB, fonction](#)
[Right, fonction](#)
[Rnd, fonction](#)
[Round, fonction](#)
[RTrim, fonction](#)
[ScriptEngine, fonction](#)
[ScriptEngineBuildVersion, fonction](#)
[ScriptEngineMajorVersion, fonction](#)
[ScriptEngineMinorVersion, fonction](#)
[Second, fonction](#)
[Select Case, instruction](#)
[Set, instruction](#)

[Sgn, fonction](#)
[Sin, fonction](#)
[Source, propriété](#)
[Space, fonction](#)
[Split, fonction](#)
[Sqr, fonction](#)
[StrComp, fonction](#)
[Chaîne, constantes](#)
[String, fonction](#)
[StrReverse, fonction](#)
[Sub, instruction](#)
[Soustraction, opérateur \(-\)](#)
[Tan, fonction](#)
[Terminate, événement](#)
[Test, méthode](#)
[Time, fonction](#)
[Timer, fonction](#)
[TimeSerial, fonction](#)
[TimeValue, fonction](#)
[Trim, fonction](#)
[3-états, constantes](#)
[True](#)
[TypeName, fonction](#)
[UBound, fonction](#)
[UCase, fonction](#)
[Value, propriété](#)
[VarType, constantes](#)
[VarType, fonction](#)
[VBScript, constantes](#)
[Weekday, fonction](#)
[WeekdayName, fonction](#)
[While...Wend, instruction](#)
[With, instruction](#)
[Xor, opérateur](#)
[Year, fonction](#)

Constantes

[Couleur, constantes](#)

[Comparaison, constantes](#)

[Date/Heure, constantes](#)

[Format de date, constantes](#)

[Divers, constantes](#)

[MsgBox, constantes](#)

[Chaîne, constantes](#)

[3-états, constantes](#)

[VarType, constantes](#)

[VBScript, constantes](#)

Microsoft® Visual Basic® Scripting Edition

Erreurs de VBScript

[Référence du langage VBScript](#)
[Informations sur la version](#)

[Erreurs d'exécution](#)

[Erreurs de syntaxe](#)

Événements

[Initialize, événement](#)

[Terminate, événement](#)

Edition **Fonctions**

[Abs, fonction](#)
[Array, fonction](#)
[Asc, fonction](#)
[Atn, fonction](#)
[CBool, fonction](#)
[CByte, fonction](#)
[CCur, fonction](#)
[CDate, fonction](#)
[CDBl, fonction](#)
[Chr, fonction](#)
[CInt, fonction](#)
[CLng, fonction](#)
[Cos, fonction](#)
[CreateObject, fonction](#)
[CSng, fonction](#)
[CStr, fonction](#)
[Date, fonction](#)
[DateAdd, fonction](#)
[DateDiff, fonction](#)
[DatePart, fonction](#)
[DateSerial, fonction](#)
[DateValue, fonction](#)
[Day, fonction](#)
[Eval, fonction](#)
[Exp, fonction](#)
[Filter, fonction](#)
[Fix, fonction](#)
[FormatCurrency, fonction](#)
[FormatDateTime, fonction](#)
[FormatNumber, fonction](#)
[FormatPercent, fonction](#)
[GetObject, fonction](#)

[GetRef, fonction](#)
[Hex, fonction](#)
[Hour, fonction](#)
[InputBox, fonction](#)
[InStr, fonction](#)
[InStrRev, fonction](#)
[Int, fonction](#)
[IsArray, fonction](#)
[IsDate, fonction](#)
[IsEmpty, fonction](#)
[IsNull, fonction](#)
[IsNumeric, fonction](#)
[IsObject, fonction](#)
[Join, fonction](#)
[LBound, fonction](#)
[LCase, fonction](#)
[Left, fonction](#)
[Len, fonction](#)
[LoadPicture, fonction](#)
[Log, fonction](#)
[LTrim, fonction](#)
[Mid, fonction](#)
[Minute, fonction](#)
[Month, fonction](#)
[MonthName, fonction](#)
[MsgBox, fonction](#)
[Now, fonction](#)
[Oct, fonction](#)
[Replace, fonction](#)
[RGB, fonction](#)
[Right, fonction](#)
[Rnd, fonction](#)
[Round, fonction](#)
[RTrim, fonction](#)
[ScriptEngine, fonction](#)
[ScriptEngineBuildVersion, fonction](#)
[ScriptEngineMajorVersion, fonction](#)

[ScriptEngineMinorVersion, fonction](#)

[Second, fonction](#)

[Sgn, fonction](#)

[Sin, fonction](#)

[Space, fonction](#)

[Split, fonction](#)

[Sqr, fonction](#)

[StrComp, fonction](#)

[String, fonction](#)

[StrReverse, fonction](#)

[Tan, fonction](#)

[Time, fonction](#)

[Timer, fonction](#)

[TimeSerial, fonction](#)

[TimeValue, fonction](#)

[Trim, fonction](#)

[TypeName, fonction](#)

[UBound, fonction](#)

[UCase, fonction](#)

[VarType, fonction](#)

[Weekday, fonction](#)

[WeekdayName, fonction](#)

[Year, fonction](#)

Méthodes

[Clear, méthode](#)

[Execute, méthode](#)

[Raise, méthode](#)

[Replace, méthode](#)

[Test, méthode](#)

Objets

[Class, objet](#)

[Dictionary, objet](#)

[Err, objet](#)

[FileSystemObject, objet](#)

[Match, objet](#)

[Matches, collection](#)

[RegExp, objet](#)

Opérateurs

[Addition \(+\), opérateur](#)
[And, opérateur](#)
[Assignment \(+\), opérateur](#)
[Concaténation \(&\), opérateur](#)
[Division \(/\), opérateur](#)
[Eqv, opérateur](#)
[Exponentiation \(^\), opérateur](#)
[Imp, opérateur](#)
[Division entière\(\\), opérateur](#)
[Is, opérateur](#)
[Mod, opérateur](#)
[Multiplication \(*\), opérateur](#)
[Négation \(-\), opérateur](#)
[Not, opérateur](#)
[Opérateurs, priorité](#)
[Or, opérateur](#)
[Soustraction \(-\), opérateur](#)
[Xor, opérateur](#)

Propriétés

[Description, propriété](#)
[FirstIndex, propriété](#)
[Global, propriété](#)
[HelpContext, propriété](#)
[HelpFile, propriété](#)
[IgnoreCase, propriété](#)
[Length, propriété](#)
[Number, propriété](#)
[Pattern, propriété](#)
[Source, propriété](#)
[Value, propriété](#)

Edition **Instructions**

[Call, instruction](#)
[Class, instruction](#)
[Const, instruction](#)
[Dim, instruction](#)
[Do...Loop, instruction](#)
[Erase, instruction](#)
[Execute, instruction](#)
[Exit, instruction](#)
[For...Next, instruction](#)
[For Each...Next, instruction](#)
[Function, instruction](#)
[If...Then...Else, instruction](#)
[On Error, instruction](#)
[Option Explicit, instruction](#)
[Private, instruction](#)
[Property Get, instruction](#)
[Property Let, instruction](#)
[Property Set, instruction](#)
[Public, instruction](#)
[Randomize, instruction](#)
[ReDim, instruction](#)
[Rem, instruction](#)
[Select Case, instruction](#)
[Set, instruction](#)
[Sub, instruction](#)
[While...Wend, instruction](#)
[With, instruction](#)

Didacticiel VBScript

[Qu'est-ce que VBScript ?](#)

[Ajout de code VBScript dans une page HTML](#)

Principes de base de VBScript

[Types de données de VBScript](#)

[Variables de VBScript](#)

[Constantes de VBScript](#)

[Opérateurs de VBScript](#)

[Utilisation des instructions conditionnelles](#)

[Boucles de répétition du code](#)

[Procédures de VBScript](#)

[Conventions de codage de VBScript](#)

Utilisation de VBScript dans Internet Explorer

[Une page VBScript simple](#)

[VBScript et les feuilles](#)

[Utilisation de VBScript avec les objets](#)

Qu'est-ce que VBScript ?

Microsoft Visual Basic Scripting Edition, le dernier membre de la famille de langages de programmation Visual Basic, permet d'utiliser des scripts actifs dans de nombreux environnements, comme les scripts clients dans Microsoft Internet Explorer et les scripts de serveur Web dans Microsoft Internet Information Server.

Facile à utiliser et à apprendre

Si vous connaissez déjà Visual Basic ou Visual Basic pour Applications, l'utilisation de VBScript sera un jeu d'enfant. Si vous ne connaissez pas Visual Basic, l'apprentissage des concepts de VBScript constitue un premier pas vers la programmation avec l'ensemble des langages de la famille Visual Basic. Bien que ces quelques pages Web puissent vous apprendre VBScript, elles ne vous enseigneront pas la programmation. Pour aborder la programmation, reportez-vous à *Step by Step*, publié par Microsoft Press.

Les scripts ActiveX actifs

VBScript parle aux applications hôtes qui utilisent ActiveX™ Scripting. Avec ActiveX Scripting, les navigateurs et les autres applications hôtes ne nécessitent pas de code d'intégration spécial pour chaque composant de script. ActiveX Scripting permet à un hôte de compiler les scripts, d'obtenir et d'appeler des points d'entrée et de gérer l'espace de nom disponible au développeur. Avec ActiveX Scripting, les fournisseurs de langages peuvent créer des exécutables de langage standard pour les scripts. Microsoft assurera le support des exécutables pour VBScript. Microsoft collabore avec différents acteurs d'Internet pour définir le standard ActiveX Scripting afin que les moteurs de script soient interchangeables. ActiveX Scripting est utilisé dans Microsoft® Internet Explorer et dans Microsoft® Internet Information Server.

VBScript dans d'autres applications et navigateurs

En tant que développeur, vous pouvez bénéficier d'une licence gratuite d'implémentation de la source VBScript dans vos produits. Microsoft fournit les implémentations binaires de VBScript pour l'API Windows® 32 bits, l'API Windows 16 bits et Macintosh®. VBScript est intégré aux navigateurs Web. VBScript et ActiveX Scripting sont également utilisables comme langage de script dans d'autres applications.

Types de données de VBScript

Quel sont les types de données de VBScript ?

VBScript comprend un seul type de données nommé **Variant**. Un **Variant** est un type de données spécial qui peut contenir différents types d'information, en fonction de son utilisation. Parce que le **Variant** est le seul type de données de VBScript, c'est aussi le type de données renvoyé par toutes les fonctions de VBScript.

À la base, un **Variant** peut contenir soit des informations numériques soit des informations de type chaîne de caractères. Un **Variant** se comporte comme un nombre lorsqu'il est utilisé dans un contexte numérique et comme une chaîne lorsqu'il est utilisé dans un contexte de chaîne. Si vous travaillez avec des données qui ressemblent à des nombres, VBScript suppose qu'il s'agit de nombres et agit de façon appropriée. De même, si vous travaillez avec des données qui ne peuvent être que des chaînes, VBScript les traite comme des chaînes. Bien sûr, vous pouvez toujours traiter les nombres comme des chaînes en les encadrant avec des guillemets (" ").

Sous-types Variant

Au-delà de la simple distinction nombre/chaîne, un **Variant** peut distinguer différents types d'information numérique. Par exemple, certaines informations numériques représentent une date ou une heure. Lorsque ces informations sont utilisées avec d'autres données de date ou d'heure, le résultat est toujours exprimé sous la forme d'une date ou d'une heure. Bien sûr, vous disposez aussi d'autres types d'information numérique, des valeurs booléennes jusqu'aux grands nombres à virgule flottante. Ces différentes catégories d'information qui peuvent être contenues dans un **Variant** sont des sous-types. Dans la plupart des cas, vous placez simplement vos données dans un **Variant** et celui-ci se comporte de la façon la plus appropriée en fonction de ces données.

Le tableau suivant présente les différents sous-types susceptibles d'être contenus dans un **Variant**.

Sous-type	Description
Empty	Le Variant n'est pas initialisé. Sa valeur est égale à zéro pour les variables numériques et à une chaîne de longueur nulle ("") pour les variables chaîne.
Null	Le Variant contient intentionnellement des données incorrectes.
Boolean	Contient True (vrai) ou False (faux).
Byte	Contient un entier de 0 à 255.
Integer	Contient un entier de -32 768 à 32 767.
Currency	-922 337 203 685 477,5808 à 922 337 203 685 477,5807.
Long	Contient un entier de -2 147 483 648 à 2 147 483 647.
Single	Contient un nombre à virgule flottante en précision simple de -3,402823E38 à -1,401298E-45 pour les valeurs négatives ; de 1,401298E-45 à 3,402823E38 pour les valeurs positives.
Double	Contient un nombre à virgule flottante en précision double de -1,79769313486232E308 à -4,94065645841247E-324 pour les valeurs négatives ; de 4,94065645841247E-324 à 1,79769313486232E308 pour les valeurs positives.
Date (Time)	Contient un nombre qui représente une date entre le 1er Janvier 100 et le 31 Décembre 9999.
String	Contient une chaîne de longueur variable limitée à environ 2 milliards de caractères.
Object	Contient un objet.
Error	Contient un numéro d'erreur.

Vous pouvez utiliser des [fonctions de conversion](#) pour convertir les données d'un sous-type vers un autre. En outre, la fonction [VarType](#) renvoie des informations sur la façon dont vos données sont stockées dans un **Variant**.

Variables de VBScript

Qu'est-ce qu'une variable ?

Une variable est un symbole pratique qui fait référence à un emplacement en mémoire où vous stockez des informations du programme pouvant varier au cours de l'exécution du script. Par exemple, vous pouvez créer une variable que vous nommez `NbClics` pour stocker le nombre de fois qu'un utilisateur clique sur un objet d'une page Web particulière. L'endroit où se trouve effectivement la variable dans la mémoire de l'ordinateur est sans importance. Ce qui compte, c'est le nom de la variable grâce auquel vous pouvez lire ou modifier sa valeur. Dans VBScript, les variables appartiennent toujours à un type de données fondamental : [Variant](#).

Déclaration des variables

Vous déclarez des variables explicitement dans votre script par l'intermédiaire des instructions [Dim](#), [Public](#) et [Private](#). Par exemple :

```
Dim DegrésFahrenheit
```

Vous pouvez déclarer plusieurs variables en séparant leur nom par une virgule. Par exemple :

```
Dim Haut, Bas, Gauche, Droite
```

Vous pouvez aussi déclarer une variable implicitement en utilisant simplement son nom dans votre script. Toutefois, cette technique n'est pas vraiment recommandée car une seule faute de syntaxe dans le nom de la variable peut donner des résultats incorrects à l'exécution. Pour cette raison, utilisez l'instruction [Option Explicit](#) pour rendre obligatoire la déclaration explicite des variables. L'instruction **Option Explicit** doit être la première

de votre script.

Restrictions de notation

Les noms de variable suivent les règles de dénomination standard de VBScript. Un nom de variable :

- doit commencer par un caractère alphabétique.
- ne peut pas contenir de point.
- ne doit pas excéder 255 caractères.
- doit être unique à l'intérieur de la même portée.

Portée et durée de vie des variables

La portée d'une variable dépend de l'endroit où elle est déclarée. Lorsque vous déclarez une variable dans une procédure, seul le code de cette procédure peut lire ou modifier la valeur de cette variable. Elle a une [portée](#) locale et elle est désignée comme variable de [niveau procédure](#). Si vous déclarez une variable en dehors d'une procédure, elle peut être reconnue par toutes les procédures de votre script. C'est alors une variable de [niveau script](#), et sa portée est l'ensemble du script.

La période d'existence d'une variable est sa durée de vie. La durée de vie d'une variable de niveau script s'étend de la déclaration de la variable à la fin de l'exécution du script. Au niveau procédure, une variable existe tant que la procédure est en cours. À la fin de la procédure, la variable est détruite. Les variables locales sont idéales pour les stockages temporaires de la procédure. Les mêmes noms de variables locales sont utilisables dans des procédures différentes parce que chacun n'est visible que dans sa procédure de déclaration.

Affectation de valeurs aux variables

Vous affectez une valeur à une variable en créant une expression de la forme suivante : la variable figure du côté gauche de l'expression et la valeur à affecter se trouve du côté droit. Par exemple :

$$B = 200$$

Variables scalaires et variables tableau

La plupart du temps, vous voulez simplement affecter une valeur à une variable que vous avez déclarée. Une variable contenant une valeur unique est une variable scalaire. Dans d'autres cas, il est pratique d'affecter plusieurs valeurs liées à une variable unique. Vous créez alors une variable contenant une série de valeurs. Dans ce cas, il s'agit d'une variable tableau. Les variables tableau et les variables scalaires sont déclarées de la même façon, sauf que la déclaration d'une variable tableau fait suivre le nom de la variable de parenthèses (). Dans l'exemple ci-dessous, on déclare un tableau à une dimension contenant 11 éléments :

Dim A(10)

Bien que le nombre entre parenthèses indique 10, ce tableau contient en fait 11 éléments car, dans VBScript, tous les [tableaux](#) commencent à zéro. Dans un tableau à base zéro, le nombre d'éléments correspond toujours au nombre entre parenthèses plus un. La taille de ce tableau est fixe.

Vous affectez les données à chaque élément en utilisant un index dans le tableau. En commençant à zéro et en finissant à 10, l'affectation des données aux éléments se présente comme suit :

A(0) = 256

A(1) = 324

A(2) = 100

...

A(10) = 55

De la même façon, vous lisez les données d'un élément particulier du tableau grâce à son index. Par exemple :

...

UneVariable = A(8)

...

Les tableaux ne sont pas limités à une seule dimension. Vous pouvez avoir jusqu'à 60 dimensions bien que la plupart des utilisateurs ne puissent

comprendre un tableau de plus de trois ou quatre dimensions. Les différentes dimensions sont déclarées en séparant à l'intérieur des parenthèses des nombres représentant leur taille. Dans l'exemple ci-dessous, la variable MaTable est un tableau à deux dimensions constitué de 6 lignes et 11 colonnes:

```
Dim MaTable(5, 10)
```

Dans les tableaux à deux dimensions, le premier nombre est le nombre de lignes, le second est le nombre de colonnes.

Vous pouvez aussi déclarer un tableau dont la taille change au cours de l'exécution du script. Il s'agit alors d'un tableau dynamique. Ce tableau est initialement déclaré au sein d'une procédure en utilisant l'instruction **Dim** ou l'instruction **ReDim**. Toutefois, pour un tableau dynamique, la taille et le nombre de dimensions ne figurent pas entre parenthèses. Par exemple :

```
Dim MonTableau()  
ReDim UnAutreTableau()
```

Pour utiliser un tableau dynamique, vous devez employer **ReDim** pour déterminer le nombre de dimensions et la taille de chaque dimension. Dans l'exemple ci-dessous, **ReDim** définit la taille initiale du tableau dynamique à 25. Une instruction **ReDim** suivante redimensionne le tableau à 30 mais utilise le mot clé **Preserve** pour préserver le contenu du tableau pendant l'opération.

```
ReDim MonTableau(25)  
...  
ReDim Preserve MonTableau(30)
```

Le nombre de redimensionnements d'un tableau n'est pas limité mais lorsque vous réduisez sa taille, vous perdez les données correspondant aux éléments supprimés.

Constantes de VBScript

Qu'est-ce qu'une constante ?

Une [constante](#) est un nom significatif qui symbolise un nombre ou une chaîne invariable. VBScript définit un certain nombre de [constantes intrinsèques](#). Pour d'autres informations sur ces constantes intrinsèques, reportez-vous à la [Référence du langage VBScript](#).

Création de constantes

Vous pouvez créer dans VBScript des constantes utilisateur par l'intermédiaire de l'instruction **Const**. L'instruction **Const** vous permet de créer des constantes de chaîne ou numériques avec des noms significatifs et de leur affecter une valeur. Par exemple :

```
Const MaChaîne = "Ceci est ma chaîne."
```

```
Const MonAge = 49
```

Remarquez que la chaîne de caractères est encadrée par des guillemets (" "). Les guillemets constituent le moyen le plus évident de différencier les valeurs chaîne des valeurs numériques. Les [littéraux de date](#) et les littéraux d'heure sont encadrés par des caractères dièse (#). Par exemple :

```
Const DateFinale = #6-1-97#
```

Il est judicieux d'adopter une convention de notation pour différencier les constantes des variables. Vous évitez ainsi de tenter d'affecter une valeur à une constante au cours de l'exécution de votre script. Par exemple, vous pouvez utiliser un préfixe "vb" ou "con" pour nommer vos constantes, ou les saisir complètement en majuscules. Différencier clairement les constantes des variables élimine les confusions lorsque les scripts

deviennent plus complexes.

Opérateurs de VBScript

VBScript comprend une gamme complète d'opérateurs : [opérateurs arithmétiques](#), [opérateurs de comparaison](#), [opérateurs de concaténation](#) et [opérateurs logiques](#).

Priorité des opérateurs

Lorsqu'une expression contient plusieurs opérations, chaque partie est évaluée et résolue dans un ordre prédéterminé appelé priorité des opérateurs. Vous pouvez utiliser des parenthèses pour modifier l'ordre de priorité et forcer l'évaluation de certaines parties d'une expression avant d'autres. Les opérations à l'intérieur des parenthèses sont toujours évaluées avant celles à l'extérieur. Toutefois, à l'intérieur des parenthèses, les priorités d'opérateurs reprennent leurs droits.

Lorsque des expressions contiennent des opérateurs de plusieurs catégories, les opérateurs arithmétiques sont évalués en premier, suivis des opérateurs de comparaison, puis des opérateurs logiques. Les opérateurs de comparaison ont tous la même priorité ; ils sont donc évalués de gauche à droite. Les opérateurs arithmétiques et logiques sont évalués dans l'ordre suivant :

Arithmétique		Comparaison		Logique	
Description	Symbole	Description	Symbole	Description	Symbole
Exponentiation	^	Égalité	=	Négation logique	Not
Négation unaire	-	Différence	<>	Conjonction logique	And
Multiplication	*	Inférieur à	<	Disjonction logique	Or
Division	/	Supérieur à	>	Exclusion logique	Xor
Division		Inférieur ou		Équivalence	

entière	\	égal à	<=	logique	Eqv
Modulo arithmétique	Mod	Supérieur ou égal à	>=	Implication logique	Imp
Addition	+	Équivalence d'objet	Is		
Soustraction	-				
Concaténation de chaînes	&				

Lorsqu'une expression contient des multiplications et des divisions, chaque opération est évaluée dans son ordre d'occurrence, de gauche à droite. De même, lorsqu'une expression contient additions et soustractions, chaque opération est évaluée dans son ordre d'occurrence, de gauche à droite.

L'opérateur de concaténation de chaîne (&) n'est pas un opérateur arithmétique mais sa priorité est inférieure à celle des opérateurs arithmétiques et supérieure à celle des options de comparaison.

L'opérateur **Is** est un opérateur de comparaison de référence d'objet. Il ne compare pas les objets ou leur valeur ; il détermine si deux références d'objet font référence au même objet.

Utilisation des instructions conditionnelles

[Didacticiel VBScript](#)
[Précédent](#)
[Suivant](#)

Contrôle de l'exécution du programme

Vous pouvez contrôler le déroulement de votre script à l'aide d'instructions conditionnelles et d'instructions de boucle. Les instructions conditionnelles vous permettent d'écrire du code VBScript qui prend des décisions et répète des actions. Les instructions conditionnelles suivantes sont disponibles dans VBScript :

- Instruction [If...Then...Else](#)
- Instruction [Select Case](#)

Prises de décision avec If...Then...Else

L'instruction **If...Then...Else** vous permet d'évaluer si une condition a pour valeur **True** (vraie) ou **False** (fausse) et, en fonction du résultat, de spécifier une ou plusieurs instructions à exécuter. En général, la condition est une expression qui utilise un opérateur de comparaison pour comparer une valeur ou une variable avec une autre. Pour des informations sur les opérateurs de comparaison, reportez-vous à [Opérateurs de comparaison](#). Il est possible d'imbriquer les instructions **If...Then...Else** sur autant de niveaux que nécessaire.

Exécution d'instructions si une condition a pour valeur True (vraie)

Pour exécuter une seule instruction lorsqu'une condition a pour valeur **True**, utilisez la syntaxe monoligne de l'instruction **If...Then...Else**.

L'exemple ci-dessous présente la syntaxe monoligne. Remarquez que cet exemple ne comporte pas le mot clé **Else**.

```
Sub MajDate()  
    Dim maDate  
    maDate = #2/13/95#  
    If maDate < Now Then maDate = Now  
End Sub
```

Pour exécuter plusieurs lignes de code, vous devez utiliser la syntaxe multiligne (ou syntaxe de bloc). Cette syntaxe comprend l'instruction **End If** comme le montre l'illustration ci-dessous :

```
Sub AlerterUtilisateur(valeur)  
    If valeur = 0 Then  
        AlertLabel.ForeColor = vbRed  
        AlertLabel.Font.Bold = True  
        AlertLabel.Font.Italic = True  
    End If  
End Sub
```

Exécution de certaines instructions si une condition a pour valeur True et exécution d'autres instructions si une condition a pour valeur False

Vous pouvez utiliser l'instruction **If...Then...Else** pour définir deux blocs d'instructions exécutables : un bloc à exécuter si la condition a pour valeur **True**, et un autre bloc à exécuter si la condition a pour valeur **False**.

```
Sub AlerterUtilisateur(valeur)  
    If valeur = 0 Then
```

```

AlertLabel.ForeColor = vbRed
AlertLabel.Font.Bold = True
AlertLabel.Font.Italic = True
Else
AlertLabel.ForeColor = vbBlack
AlertLabel.Font.Bold = False
AlertLabel.Font.Italic = False
End If
End Sub

```

Choisir entre plusieurs alternatives

Une variante de l'instruction **If...Then...Else** vous permet de choisir entre plusieurs alternatives. L'ajout de clauses **ElseIf** étend la fonctionnalité de l'instruction **If...Then...Else** pour vous permettre de contrôler le déroulement du programme en fonction de différentes possibilités. Par exemple :

```

Sub AfficherValeur(value)
If value = 0 Then
MsgBox valeur
ElseIf value = 1 Then
MsgBox valeur
ElseIf value = 2 Then
Msgbox valeur
Else
Msgbox "Valeur hors limites!"
End If

```

Vous pouvez ajouter autant de clauses **ElseIf** que nécessaire pour créer des alternatives. Mais une utilisation extensive des clauses **ElseIf** peut devenir complexe. L'instruction **Select Case** constitue le meilleur moyen de choisir entre plusieurs alternatives.

Prises de décision avec Select Case

La structure **Select Case** offre une alternative à **If...Then...ElseIf** pour exécuter de façon sélective un bloc d'instructions parmi plusieurs. Une instruction **Select Case** offre les mêmes possibilités que **If...Then...Else**, mais rend le code plus efficace et plus lisible.

Une structure **Select Case** fonctionne avec une seule expression de test, évaluée une fois, au début de la structure. Le résultat de l'expression est ensuite comparé avec les valeurs de chaque bloc **Case** de la structure. Lorsqu'il y a correspondance, le bloc d'instructions **Case** concerné s'exécute :

```
Select Case Document.Form1.CardType.Options(0)
  Case "MasterCard"
    AfficherLogoMC
    ValiderCompteMC
  Case "Visa"
    AfficherLogoVisa
    ValiderCompteVisa
  Case "American Express"
    AfficherLogoAMEXCO
    ValiderCompteAMEXCO
  Case Else
    AfficherInconnue
```

Redemander End Select

Remarquez que la structure **Select Case** évalue l'expression une fois au début de la structure. Par contre, la structure **If...Then...ElseIf** peut évaluer une expression différente pour chaque instruction **ElseIf**. Vous pouvez remplacer une structure **If...Then...ElseIf** par une structure **Select Case** uniquement si chaque instruction **ElseIf** évalue la même expression.

Boucles de répétition du code

Utilisation des boucles pour répéter du code

Les boucles vous permettent de répéter l'exécution d'un groupe d'instructions. Certaines boucles répètent les instructions jusqu'à ce qu'une condition ait pour valeur **False** (fausse) ; d'autres répètent les instructions jusqu'à ce qu'une condition ait pour valeur **True** (vraie). Il existe aussi des boucles qui répètent des instructions un nombre de fois spécifié.

Les instructions de boucle suivantes sont disponibles dans VBScript :

- [Do...Loop](#) : effectue une boucle tant qu'une condition est **True** ou jusqu'à ce qu'elle le devienne.
- [While...Wend](#) : effectue une boucle tant qu'une condition a pour valeur **True**.
- [For...Next](#) : utilise un compteur pour exécuter des instructions un nombre de fois spécifié.
- [For Each...Next](#) : répète un groupe d'instructions pour chaque élément d'une collection ou pour chaque élément d'un tableau.

Utilisation des boucles Do

Vous pouvez utiliser les instructions **Do...Loop** pour exécuter un bloc d'instructions un nombre de fois indéfini. Les instructions sont répétées tant qu'une condition a pour valeur **True** ou jusqu'à ce qu'elle ait pour valeur **True**.

Répétition d'instructions tant qu'une condition a pour valeur True

Utilisez le mot clé **While** pour contrôler une condition dans une instruction **Do...Loop**. Vous pouvez contrôler la condition avant d'entrer dans la boucle (comme dans l'exemple CtrlPremierWhile suivant) ou après que la boucle se soit exécutée au moins une fois (comme dans l'exemple CtrlDernierWhile suivant). Dans la procédure CtrlPremierWhile, si monNum a pour valeur 0 au lieu de 20, les instructions à l'intérieur de la boucle ne s'exécuteront jamais. Dans la procédure CtrlDernierWhile, les instructions à l'intérieur de la boucle s'exécutent une seule fois parce que la condition a déjà pour valeur **False**.

```
Sub CtrlPremierWhile()  
    Dim compteur, monNum  
    compteur = 0  
    monNum = 20  
    Do While monNum > 10  
        monNum = monNum - 1  
        compteur = compteur + 1  
    Loop  
    MsgBox "La boucle a effectué " & compteur & " répétitions."  
End Sub
```

```
Sub CtrlDernierWhile()  
    Dim compteur, monNum  
    compteur = 0  
    monNum = 9  
    Do  
        monNum = monNum - 1  
        compteur = compteur + 1  
    Loop While monNum > 10  
    MsgBox "La boucle a effectué " & compteur & " répétitions."  
End Sub
```

Répétition d'instructions jusqu'à ce qu'une condition prenne la valeur True

Vous pouvez utiliser le mot clé **Until** de deux façons pour contrôler une

condition dans une instruction **Do...Loop**. Vous pouvez contrôler la condition avant d'entrer dans la boucle (comme dans l'exemple CtrlPremierUntil suivant) ou après que la boucle se soit exécutée au moins une fois (comme dans l'exemple CtrlDernierUntil suivant). La boucle est effective tant que la condition a pour valeur **False**.

```
Sub CtrlPremierUntil()  
    Dim compteur, monNum  
    compteur = 0  
    monNum = 20  
    Do Until monNum = 10  
        monNum = monNum - 1  
        compteur = compteur + 1  
    Loop  
    MsgBox "La boucle a effectué " & compteur & " répétitions."  
End Sub
```

```
Sub CtrlDernierUntil()  
    Dim compteur, monNum  
    compteur = 0  
    monNum = 1  
    Do  
        monNum = monNum + 1  
        compteur = compteur + 1  
    Loop Until monNum = 10  
    MsgBox "La boucle a effectué " & compteur & " répétitions."  
End Sub
```

Sortie d'une instruction **Do...Loop** dans une boucle

Vous pouvez sortir d'une boucle **Do...Loop** en utilisant l'instruction **Exit Do**. En règle générale, vous ne sortez d'une boucle que dans des situations particulières (pour éviter une boucle infinie, par exemple). Dans ce cas, vous devez utiliser l'instruction **Exit Do** uniquement dans le bloc d'instructions **True** d'une instruction **If...Then...Else**. Si la condition a pour

valeur **False**, la boucle s'exécute normalement.

Dans l'exemple ci-dessous, monNum reçoit une valeur qui crée une boucle infinie. L'instruction **If...Then...Else** contrôle cette condition afin de prévenir une répétition infinie.

```
Sub ExempleDeSortie()  
    Dim compteur, monNum  
    compteur = 0  
    monNum = 9  
    Do Until monNum = 10  
        monNum = monNum - 1  
        compteur = compteur + 1  
        If monNum < 10 Then Exit Do  
    Loop  
    MsgBox "La boucle a effectué " & compteur & " répétitions."  
End Sub
```

Utilisation de While...Wend

L'instruction **While...Wend** figure dans VBScript à l'intention des utilisateurs avertis. Cependant, en raison du manque de souplesse de **While...Wend**, il est recommandé d'utiliser plutôt **Do...Loop**.

Utilisation de For...Next

Vous pouvez utiliser les instructions **For...Next** pour exécuter un bloc d'instructions un nombre de fois spécifié. Pour les boucles, utilisez une variable compteur dont la valeur est augmentée ou diminuée à chaque répétition de la boucle.

Dans l'exemple ci-dessous, la procédure MaProc s'exécute 50 fois. L'instruction **For** spécifie la variable compteur x et ses valeurs de début et de fin. L'instruction **Next** incrémente la variable compteur de 1.

```
Sub ExecMaProc50fois()  
    Dim x  
    For x = 1 To 50  
        MaProc
```

```
Next
End Sub
```

Grâce au mot clé **Step**, vous pouvez spécifier la valeur d'incrémentation ou de décrémentation de la variable. Dans l'exemple ci-dessous, la variable compteur `j` est incrémentée de 2 à chaque itération de la boucle. Lorsque la boucle se termine, le total est la somme de 2, 4, 6, 8 et 10.

```
Sub TotalDes2()
    Dim j, total
    For j = 2 To 10 Step 2
        total = total + j
    Next
    MsgBox "Le total est " & total
End Sub
```

Pour décrémenter la variable compteur, vous utilisez une valeur de **Step** négative. Vous spécifiez alors une valeur de fin inférieure à la valeur de départ. Dans l'exemple ci-dessous, la variable compteur `monNum` est décrémentée de 2 à chaque itération de la boucle. Lorsque la boucle se termine, le total est la somme de 16, 14, 12, 10, 8, 6, 4 et 2.

```
Sub NouveauTotal()
    Dim monNum, total
    For monNum = 16 To 2 Step -2
        total = total + monNum
    Next
    MsgBox "Le total est " & total
End Sub
```

Vous pouvez quitter une instruction **For...Next** avant que le compteur atteigne sa valeur de fin en utilisant l'instruction **Exit For**. En règle générale, vous ne sortez d'une boucle que dans des situations particulières. En cas d'erreur par exemple, vous devez utiliser l'instruction **Exit For** uniquement dans le bloc d'instructions **True** d'une instruction **If...Then...Else**. Si la condition a pour valeur **False**, la boucle s'exécute

normalement.

Utilisation de Each...Next

Une boucle **For Each...Next** ressemble à une boucle **For...Next**. Au lieu de répéter un groupe d'instructions un nombre de fois spécifié, une boucle **For Each...Next** le répète pour chaque élément d'une collection d'objets ou d'un tableau. Ceci s'avère particulièrement utile lorsque vous ne connaissez pas le nombre d'éléments d'une collection.

Dans l'exemple de code HTML suivant, le contenu d'un objet **Dictionary** est utilisé pour placer du texte dans plusieurs boîtes de texte :

```
<HTML>
<HEAD><TITLE>Feuilles et éléments</TITLE></HEAD>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub cmdChanger_OnClick
  Dim d          'Créer une variable
  Set d = CreateObject("Scripting.Dictionary")
  d.Add "0", "Athènes"  'Ajouter des clés et des éléments
  d.Add "1", "Belgrade"
  d.Add "2", "Le Caire"

  For Each I in d
    Document.frmForm.Elements(I).Value = D.Item(I)
  Next
End Sub
-->
</SCRIPT>
<BODY>
<CENTER>
<FORM NAME="frmForm"

<Input Type = "Text"><p>
```

```
<Input Type = "Text"><p>  
<Input Type = "Text"><p>  
<Input Type = "Text"><p>  
<Input Type = "Button" NAME="cmdChanger" VALUE="Clique  
</FORM>  
</CENTER>  
</BODY>  
</HTML>
```

Procédures de VBScript

Les types de procédures

Dans VBScript, il existe deux types de procédures : la procédure [Sub](#) et la procédure [Function](#).

Procédures Sub

Une procédure **Sub** est une série d'instructions VBScript, encadrée par les instructions **Sub** et **End Sub**, qui effectue des actions mais ne renvoie pas de valeur. Une procédure **Sub** peut accepter des arguments (constantes, variables ou expressions transmises par une procédure appelante). Si une procédure **Sub** n'a pas d'arguments, son instruction **Sub** doit présenter une paire de parenthèses vides.

La procédure **Sub** suivante utilise deux fonctions VBScript intrinsèques, ou intégrées, [MsgBox](#) et [InputBox](#), pour demander des informations à l'utilisateur. Elle affiche ensuite les résultats d'un calcul basé sur ces informations. Le calcul est effectué dans une procédure **Function** créée à l'aide de VBScript. La procédure **Function** est présentée dans le prochain paragraphe.

```
Sub ConvertTemp()  
    temp = InputBox("Veuillez entrer la température en degrés F.", "Température")  
    MsgBox "La température est de " & Celsius(temp) & " degrés C"  
End Sub
```

Procédures Function

Une procédure **Function** est une série d'instructions VBScript encadrée par les instructions **Function** et **End Function**. Une procédure **Function** est semblable à une procédure **Sub** mais peut renvoyer une valeur. Une procédure **Function** peut accepter des arguments (constantes, variables ou expressions transmises par une procédure appelante). Si une procédure

Function n'a pas d'arguments, son instruction **Function** doit présenter une paire de parenthèses vide. Une procédure **Function** renvoie une valeur en affectant une valeur à son nom dans une ou plusieurs instructions de la procédure. Le type du retour d'une **Function** est toujours **Variant**.

Dans l'exemple ci-dessous, la fonction Celsius calcule les degrés Celsius à partir des degrés Fahrenheit. Lorsque la fonction est appelée à partir de la procédure **Sub** ConvertTemp, une variable contenant la valeur argument lui est transmise. Le résultat du calcul est renvoyé à la procédure appelante et affiché dans une boîte de message.

```
Sub ConvertTemp()  
    temp = InputBox("Veuillez entrer la température en degrés F.",  
    MsgBox "La température est de " & Celsius(temp) & " degrés ("  
End Sub
```

```
Function Celsius(degrésF)  
    Celsius = (degrésF - 32) * 5 / 9  
End Function
```

Échange des données avec les procédures

Chaque élément de données est transmis à vos procédures par l'intermédiaire d'un [argument](#). Les arguments servent de symboles aux données que vous voulez transmettre à la procédure. Vous pouvez nommer vos arguments de la même manière que vous nommez des variables. Lorsque vous créez une procédure par une instruction **Sub** ou une instruction **Function**, les parenthèses doivent figurer après le nom de la procédure. Les arguments éventuels figurent entre ces parenthèses, séparés par des virgules. Par exemple, dans l'exemple suivant `degrésF` symbolise la valeur transmise à la fonction Celsius pour conversion :

```
Function Celsius(degrésF)  
    Celsius = (degrésF - 32) * 5 / 9  
End Function
```

Pour récupérer les données d'une procédure, vous devez utiliser une **Function**. Une procédure **Function** peut renvoyer une valeur, une procédure **Sub** ne le peut pas.

Utilisation des procédures Sub et Function dans le code

Dans votre code, une **Function** doit toujours apparaître à droite d'une affectation de variable ou dans une expression. Par exemple :

```
Temp = Celsius(degrésF)
```

ou

```
MsgBox "La température est de " & Celsius(degrésF) & " degrés
```

Pour appeler une procédure **Sub** à partir d'une autre procédure, faites simplement figurer le nom de la procédure suivi le cas échéant des valeurs arguments séparées par des virgules. L'instruction **Call** n'est pas obligatoire, mais si vous l'utilisez, vous devez encadrer les arguments éventuels par des parenthèses.

L'exemple ci-dessous présente deux appels à la procédure MaProc. L'un utilise l'instruction **Call**, l'autre non. Les deux font exactement la même chose.

```
Call MaProc(arg1, arg2)  
MaProc arg1, arg2
```

Remarquez que les parenthèses sont omises lorsque l'instruction **Call** n'est pas utilisée.

Conventions de codage de VBScript

[Didacticiel VBScript](#)
[Précédent](#)
[Suivant](#)

Qu'entend-on par conventions de codage ?

Ces conventions sont des suggestions pouvant simplifier l'écriture de code avec Microsoft Visual Basic Scripting Edition. Elles incluent notamment :

- des conventions d'affectation de noms à des objets, des variables et des procédures,
- des conventions de commentaire,
- des directives de mise en forme et de mise en retrait.

L'utilisation de conventions cohérentes vise principalement à normaliser la structure et le style de codage d'un script ou d'un jeu de scripts pour rendre le code plus lisible et mieux compréhensible. L'emploi de conventions de codage appropriées permet de créer un code source précis, lisible, intuitif, ne présentant aucune ambiguïté, et respectant les conventions des autres langages.

Conventions d'affectation de noms à des constantes

Les versions précédentes de VBScript n'avaient pas de mécanisme de création de constantes définies par l'utilisateur. Les constantes, si elles étaient utilisées, étaient mises en uvre comme des variables et distinguées des autres variables grâce à une syntaxe en caractères majuscules. Les différents mots étaient séparés par le caractère de soulignement (_). Par exemple :

```
MAX_LISTE_UTILISATEUR
```

SAUT_DE_LIGNE

Bien que ce mécanisme soit toujours acceptable pour identifier vos constantes, vous pouvez utiliser un nouveau schéma de notation car il est possible de créer de véritables constantes avec l'instruction [Const](#). Cette convention utilise un format mixte, dans lequel les noms de constante commencent par le préfixe "con". Par exemple :

conVotrePropreConstante

Conventions d'affectation de noms à des variables

Pour assurer lisibilité et cohérence, utilisez dans votre code VBScript les préfixes présentés dans le tableau suivant, avec des noms de variable descriptifs.

Sous-type	Préfixe	Exemple
Boolean	bln	blnTrouvé
Byte	byt	bytDonnéesRaster
Date (Time)	dtm	dtmDébut
Double	dbl	dblTolérance
Error	err	errNumOrdre
Integer	int	intQuantité
Long	lng	lngDistance
Object	obj	objCourant
Single	sng	sngMoyenne
String	str	strPrénom

Portée des variables

La portée des variables doit toujours être la plus petite possible. Les [variables VBScript](#) peuvent avoir la portée suivante.

Portée	Endroit de déclaration de la variable	Visibilité
--------	---------------------------------------	------------

Niveau procédure	Procédure Event, Fonction ou Sub	Visible dans la procédure où elle est déclarée
Niveau script	Section HEAD d'une page HTML, hors de toute procédure	Visible dans toutes les procédures du script

Préfixes de portée de variable

Plus la taille du script augmente, plus il devient indispensable de pouvoir différencier rapidement la portée des variables. Un préfixe d'une lettre placé devant le préfixe de type assure cette différenciation, sans trop augmenter la taille des noms de variable.

Portée	Préfixe	Exemple
Niveau procédure	Aucun	dblVélocité
Niveau script	s	sbInCalculEnCours

Noms descriptifs de variable et de procédure

Le corps d'un nom de variable ou de procédure doit être composé de lettres minuscules et majuscules et décrire sa fonction. En outre, les noms de procédure doivent commencer par un verbe, tel que InitialiserTableau ou FermerDialogue.

Dans le cas de termes longs ou fréquemment utilisés, il est préférable d'employer des abréviations normalisées pour maintenir une longueur de nom raisonnable. Pour assurer la lisibilité du code, il est généralement préférable d'utiliser des noms de variable de moins de 32 caractères. Lors de l'emploi d'abréviations, assurez-vous que celles-ci sont cohérentes dans tout le script. Par exemple, l'utilisation aléatoire de Ctr et Compteur dans un script ou un ensemble de scripts peut créer une confusion.

Conventions d'affectation de noms à des objets

Le tableau suivant présente les conventions recommandées pour les divers objets utilisables en programmation VBScript.

--	--	--

Type d'objet	Préfixe	Exemple
Panneau 3D	pnl	pnlGroupe
Bouton animé	ani	aniBoîteALettre
Case à cocher	chk	chkLectureSeule
Liste modifiable, zone de liste déroulante	cbo	cboAnglais
Bouton de commande	cmd	cmdQuitter
Boîte de dialogue commune	dlg	dlgFichierOuvrir
Cadre	fra	fraLangage
Barre de défilement horizontale	hsb	hsbVolume
Image	img	imgIcône
Étiquette	lbl	lblMessageAide
Ligne	lin	linVerticale
Zone de liste	lst	lstCodesEmplois
Compteur	spn	spnPages
Zone de texte	txt	txtNom
Barre de défilement verticale	vsb	vsbVitesse
Curseur	sld	sldÉchelle

Conventions pour les commentaires dans le code

Toutes les procédures doivent commencer par un bref commentaire décrivant leur action. Cette description doit exclure les détails de mise en uvre (techniques employées) parce que ceux-ci sont sujets à modification et pourraient imposer une gestion de commentaire inutile ou pire, devenir erronés. La mise en uvre est décrite par le code lui-même et d'éventuels commentaires sur une ligne.

Les arguments passés à une procédure doivent être décrits lorsque leur fonction n'est pas évidente et qu'ils doivent être compris dans une plage spécifique. Les valeurs de retour des fonctions et les variables modifiées par une procédure, notamment par l'intermédiaire d'arguments de référence, doivent également être décrites au début de chaque procédure.

Les commentaires des en-têtes de procédure doivent inclure les titres de section suivants. La section suivante, "Mise en forme du code", vous présente des exemples.

Titre de section	Contenu des commentaires
Objet	Ce que fait la procédure (et non comment elle le fait).
Commentaires	Liste de variables, contrôles, ou autres éléments externes dont l'état a une incidence sur cette procédure.
Effets	Présentation de l'effet de la procédure sur les variables, contrôles ou autres éléments externes.
Entrées	Explication de tous les arguments non évidents. Entrez une ligne de commentaire distincte pour chaque argument.
Valeurs renvoyées	Explication de la valeur renvoyée.

N'oubliez pas :

- Pour chaque déclaration de variable importante, prévoyez un commentaire sur une ligne décrivant l'utilisation.
- Les noms des variables, contrôles et procédures doivent être suffisamment clairs pour que les commentaires sur une ligne servent uniquement à fournir des détails de mise en uvre complexe.
- Entrez au début de votre script une présentation générale, décrivant le script, énumérant les objets, les procédures, les algorithmes, les boîtes de dialogue et les autres dépendances système. Il peut parfois être utile d'inclure un peu de pseudocode décrivant l'algorithme.

Mise en forme du code

Bien que la mise en forme doive refléter la structure logique et l'imbrication du code, il convient d'économiser au maximum l'espace écran. Voici quelques conseils à cet égard :

- Les blocs imbriqués standard doivent être mis en retrait de quatre espaces.

- Les commentaires généraux d'une procédure doivent être mis en retrait d'un espace.
- Les instructions du plus haut niveau venant après les commentaires généraux doivent être mises en retrait de quatre espaces, chaque bloc imbriqué étant lui-même mis en retrait de quatre espaces supplémentaires. Par exemple :

```

'*****:
' Objet:  Trouve la première occurrence d'un utilisateur
'         spécifié dans le tableau ListeUtilisateurs.
' Entrées: strListeUtilisateurs(): liste des utilisateurs
'         dans laquelle effectuer la recherche.
'         strUtilisateurCible: nom de l'utilisateur à
'         rechercher.
' Retours: L'index de la première occurrence de
'         strUtilisateurCible dans le tableau
'         ListeUtilisateurs.
'         Si l'utilisateur cible est introuvable,
'         renvoie -1.
'*****:

```

```

Function intChercherUtilisateur (strListeUtilisateurs(), strUtilisateurCible)
    Dim i          ' Compteur de boucle.
    Dim blnTrouvé ' Indicateur de cible trouvée
    intChercherUtilisateur = -1
    i = 0          ' Initialiser le compteur de boucle
    Do While i <= Ubound(strListeUtilisateurs) and Not blnTrouvé
        If strListeUtilisateurs(i) = strUtilisateurCible Then
            blnTrouvé = True ' Indicateur à True
            intChercherUtilisateur = i ' Définir valeur retour compteur
        End If
        i = i + 1          ' Incrément compteur de boucle
    Loop
End Function

```

Loop
End Function

VBScript et les feuilles

Validation simple

Vous pouvez utiliser Visual Basic Scripting Edition pour effectuer une grande partie des traitements de feuille qui nécessitent en général un serveur. Vous pouvez aussi réaliser des opérations impossibles sur un serveur.

Voici un exemple de validation simple côté client. Le code HTML correspond à un champ de texte et un bouton. Si vous utilisez Microsoft® Internet Explorer pour afficher la [page produite](#) par le code suivant, vous observerez un petit champ de texte à côté d'un bouton.

```
<HTML>
<HEAD><TITLE>Validation simple</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub Soumettre_OnClick
  Dim LaFeuille
  Set LaFeuille = Document.ValidForm
  If IsNumeric(LaFeuille.Text1.Value) Then
    If LaFeuille.Text1.Value < 1 Or LaFeuille.Text1.
      MsgBox "Veuillez entrer un nombre entre 1 et 10"
    Else
      MsgBox "Merci."
    End If
End If
```

```

Else
  MsgBox "Veuillez entrer une valeur numérique."
End If
End Sub
-->
</SCRIPT>
</HEAD>
<BODY>
<H3>Validation simple</H3><HR>
<FORM NAME="ValidForm">
Entrez une valeur entre 1 et 10:
<INPUT NAME="Text1" TYPE="TEXT" SIZE="
<INPUT NAME="Submit" TYPE="BUTTON" VA
</FORM>
</BODY>
</HTML>

```

La différence entre ce champ de texte et les exemples de [Une page VBScript simple](#) est que la propriété **Value** du champ de texte est utilisée pour vérifier la valeur entrée. Pour lire une propriété **Value**, le code doit qualifier la référence au nom du champ de texte.

Vous pouvez toujours écrire la référence complète Document.FeuilleValidation.Texte1. Toutefois, lorsque vous avez de multiples références aux contrôles de la feuille, vous emploierez la méthode suivante. Déclarez tout d'abord une variable. Utilisez ensuite l'instruction [Set](#) pour affecter la feuille à la variable LaFeuille. Une instruction d'affectation normale, comme [Dim](#), ne fonctionne pas dans ce cas ; vous devez utiliser [Set](#) pour préserver la référence à un objet.

Utilisation de valeurs numériques

Remarquez que l'exemple teste directement la valeur par rapport à un

nombre : il utilise la fonction [IsNumeric](#) pour vérifier que la chaîne dans le champ est un nombre. Bien que VBScript convertisse automatiquement les chaînes et les nombres, il est recommandé de tester le sous-type des valeurs saisies par l'utilisateur et d'utiliser les [fonctions de conversion](#) le cas échéant. Pour additionner des valeurs issues de champs de texte, convertissez les valeurs explicitement en nombres parce que le symbole d'opérateur \pm représente à la fois l'addition numérique et la concaténation de chaînes. Par exemple, si Texte1 contient "1" et Texte2 contient "2", on observe les résultats suivants :

```
A = Texte1.Value + Texte2.Value      ' A vaut "12"  
A = CDbI(Texte1.Value) + Texte2.Value  ' A vaut 3
```

Validation et renvoi de données vers le serveur

L'exemple de validation simple utilise un simple contrôle Button. Si un contrôle Submit était utilisé, l'exemple ne verrait pas les données pour les vérifier puisqu'elles seraient envoyées immédiatement au serveur. En évitant l'utilisation du contrôle Submit, vous pouvez vérifier les données mais celles-ci ne sont pas soumises au serveur. Ceci nécessite une ligne de code supplémentaire :

```
<SCRIPT LANGUAGE="VBScript">  
<!--  
Sub Soumettre_OnClick  
    Dim LaFeuille  
    Set LaFeuille = Document.ValidForm  
    If IsNumeric(LaFeuille.Texte1.Value) Then  
        If LaFeuille.Texte1.Value < 1 Or LaFeuille.Texte1.Value > 10 T  
            MsgBox "Veuillez entrer un nombre entre 1 et 10."  
        Else  
            MsgBox "Merci."  
            LaFeuille.Submit      ' Données correctes; envoyer au serveur.  
        End If  
    Else
```

```
    MsgBox "Veuillez entrer une valeur numérique."  
End If  
End Sub  
-->  
</SCRIPT>
```

Pour envoyer les données au serveur, le code appelle la méthode **Submit** sur l'objet feuille lorsque les données sont correctes. À partir de ce moment, le serveur gère les données comme il l'aurait fait directement, sauf que les données ont été vérifiées avant d'être envoyées. Vous trouverez des informations complètes sur la méthode **Submit** et sur d'autres méthodes dans la documentation Internet Explorer Scripting Object Model accessible à partir du site Web de Microsoft (<http://www.microsoft.com>) ou sur le site européen (<http://www.eu.microsoft.com>).

Pour l'instant, vous avez vu uniquement les objets <FORM> HTML standard. Internet Explorer vous permet aussi d'exploiter toute la puissance des contrôles ActiveX™ (anciennement contrôles OLE) et Java™.

, balises"> tags">

Microsoft® Visual Basic® Scripting Edition

Utilisation de VBScript avec les objets

[Didacticiel VBScript](#)
[Précédent](#)

Utilisation d'objets

Que vous utilisiez un contrôle ActiveX™ (anciennement nommé contrôle OLE) ou un objet Java™, Microsoft Visual Basic Scripting Edition et Microsoft® Internet Explorer le gèrent de la même façon. Si vous utilisez Internet Explorer et avez installé le contrôle **Label**, vous pouvez observer la [page produite](#) par le code ci-dessous.

Vous incorporez un objet avec des balises <OBJECT> et définissez ses valeurs de propriétés initiales avec des balises <PARAM>. Si vous êtes un programmeur Visual Basic, vous constaterez que l'utilisation des balises <PARAM> correspond à la définition des propriétés initiales d'un contrôle sur une feuille. Par exemple, l'ensemble de balises <OBJECT> et <PARAM> ajoute le contrôle Label ActiveX à une page :

```
<OBJECT
```

```
  classid="clsid:99B42120-6EC7-11CF-A6C7-0
```

```
  id=lblActiveLbl
```

```
  width=250
```

```
  height=250
```

```
  align=left
```

```
  hspace=20
```

```
  vspace=0
```

```
>
```

```

<PARAM NAME="Angle" VALUE="90">
<PARAM NAME="Alignment" VALUE="4">
<PARAM NAME="BackStyle" VALUE="0">
<PARAM NAME="Caption" VALUE="Une simpl
<PARAM NAME="FontName" VALUE="Verdana
<PARAM NAME="FontSize" VALUE="20">
<PARAM NAME="FontBold" VALUE="1">
<PARAM NAME="FrColor" VALUE="0">
</OBJECT>

```

Vous pouvez lire et définir des propriétés et appeler des méthodes comme pour tout contrôle de la feuille. Le code ci-dessous, par exemple, comprend des contrôles <FORM> que vous pouvez utiliser pour manipuler deux propriétés du contrôle Label :

```

<FORM NAME="LabelControls">
<INPUT TYPE="TEXT" NAME="txtNewText" S
<INPUT TYPE="BUTTON" NAME="cmdChange
<INPUT TYPE="BUTTON" NAME="cmdPivoter
</FORM>

```

Lors de la définition de la feuille, une procédure d'événement pour le bouton cmdChanger change le texte de l'étiquette :

```

<SCRIPT LANGUAGE="VBScript">
<!--
Sub cmdChanger_onClick
    Dim LaFeuille

```

```
Set LaFeuille = Document.LabelControls
lblActiveLbl.Caption = LaFeuille.txtNewText
End Sub
-->
</SCRIPT>
```

Le code qualifie les références aux contrôles et valeurs à l'intérieur des feuilles comme dans l'exemple [Validation Simple](#).

Plusieurs contrôles ActiveX™ sont disponibles pour utilisation avec Internet Explorer. Vous trouverez des informations complètes sur les propriétés, méthodes et événements, ainsi que sur les identificateurs de classe (CLSID) pour les contrôles sur le site Web de Microsoft (<http://www.microsoft.com>) ou sur le site européen (<http://www.eu.microsoft.com>). Vous trouverez d'autres informations sur la balise <OBJECT> dans la page *Internet Explorer 4.0 Author's Guide and HTML Reference*.

Remarque Les versions précédentes d'Internet Explorer exigeaient des accolades ({}) autour de l'attribut d'identificateur de classe et n'étaient pas conformes à la norme W3C. L'utilisation d'accolades avec la version actuelle génère le message "This page uses an outdated version of the <OBJECT> tag".

Référence de la bibliothèque d'exécution Scripting

[Référence du langage](#)
[Informations sur la version](#)

Bienvenue dans la référence de la bibliothèque d'exécution Scripting

Ces différentes sections présentent des informations qui vous aideront à découvrir les divers composants de la bibliothèque d'exécution Scripting.

Vous trouverez *tous* les composants de la bibliothèque d'exécution Scripting répertoriés par ordre alphabétique sous la liste des mots clés. Si vous souhaitez vous concentrer sur une seule catégorie, les objets par exemple, chaque catégorie du langage dispose de sa propre section plus concise.

Cliquez sur l'un des intitulés de gauche pour afficher la liste des éléments contenus dans cette catégorie. À partir de cette liste, sélectionnez la rubrique souhaitée. Une fois cette rubrique

- ◆ Caractéristiques
- ◆ Liste des mots clés
- ◆ Constantes
- ◆ Méthodes
- ◆ Objets
- ◆ Propriétés

affichée, vous pourrez facilement naviguer vers d'autres rubriques

Maintenant, à vous de jouer ! Étudiez des instructions, examinez les méthodes ou découvrez quelques caractéristiques. Vous constaterez la richesse de la bibliothèque d'exécution Scripting !

Microsoft® Visual Basic® Scripting Edition

Informations sur les caractéristiques

[Référence de la bibliothèque
d'exécution Scripting](#)
[Informations sur la version](#)

[Caractéristiques de la bibliothèque d'exécution Microsoft Scripting](#)

Liste des mots clés

[Add, méthode \(Dictionary\)](#)
[Add, méthode \(Folders\)](#)
[AtEndOfLine, propriété](#)
[AtEndOfStream, propriété](#)
[Attributes, propriété](#)
[Attributs de fichier, constantes](#)
[AvailableSpace, propriété](#)
[BuildPath, méthode](#)
[Close, méthode](#)
[Column, propriété](#)
[CompareMode, propriété](#)
[Copy, méthode](#)
[CopyFile, méthode](#)
[CopyFolder, méthode](#)
[Count, propriété](#)
[CreateFolder, méthode](#)
[CreateTextFile, méthode](#)
[DateCreated, propriété](#)
[DateLastAccessed, propriété](#)
[DateLastModified, propriété](#)
[Delete, méthode](#)
[DeleteFile, méthode](#)
[DeleteFolder, méthode](#)
[Dictionary, objet](#)
[Drive, objet](#)
[Drive, propriété](#)
[Drive Type, constantes](#)
[DriveExists, méthode](#)
[DriveLetter, propriété](#)
[Drives, collection](#)
[Drives, propriété](#)

[DriveType, propriété](#)
[Entrée/Sortie de fichier, constantes](#)
[Exists, méthode](#)
[FileExists, méthode](#)
[File, objet](#)
[Files, collection](#)
[Files, propriété](#)
[FileSystemObject, constantes](#)
[FileSystemObject, objet](#)
[FileSystem, propriété](#)
[Folder, objet](#)
[Folders, collection](#)
[FolderExists, méthode](#)
[FreeSpace, propriété](#)
[GetAbsolutePathName, méthode](#)
[GetBaseName, méthode](#)
[GetDrive, méthode](#)
[GetDriveName, méthode](#)
[GetExtensionName, méthode](#)
[GetFile, méthode](#)
[GetFileName, méthode](#)
[GetFileVersion, méthode](#)
[GetFolder, méthode](#)
[GetParentFolderName, méthode](#)
[GetSpecialFolder, méthode](#)
[GetTempName, méthode](#)
[IsReady, propriété](#)
[IsRootFolder, propriété](#)
[Item, propriété](#)
[Items, méthode](#)
[Key, propriété](#)
[Keys, méthode](#)
[Line, propriété](#)
[Move, méthode](#)
[MoveFile, méthode](#)
[MoveFolder, méthode](#)
[Name, propriété](#)

[OpenAsTextStream, méthode](#)
[OpenTextFile, méthode](#)
[ParentFolder, propriété](#)
[Path, propriété](#)
[Read, méthode](#)
[ReadAll, méthode](#)
[ReadLine, méthode](#)
[Remove, méthode](#)
[RemoveAll, méthode](#)
[RootFolder, propriété](#)
[SerialNumber, propriété](#)
[ShareName, propriété](#)
[ShortName, propriété](#)
[ShortPath, propriété](#)
[Size, propriété](#)
[Skip, méthode](#)
[SkipLine, méthode](#)
[SpecialFolder, constantes](#)
[Subfolders, propriété](#)
[TextStream, objet](#)
[TotalSize, propriété](#)
[Type, propriété](#)
[VolumeName, propriété](#)
[Write, méthode](#)
[WriteBlankLines, méthode](#)
[WriteLine, méthode](#)

Constantes

[Référence de la bibliothèque
d'exécution Scripting
Informations sur la version](#)

[DriveType, constantes](#)

[File Attribute, constantes](#)

[File Input/Output, constantes](#)

[FileSystemObject, constantes](#)

[SpecialFolder, constantes](#)

Méthodes

[Add, méthode \(Dictionary\)](#)
[Add, méthode \(Folders\)](#)
[BuildPath, méthode](#)
[Close, méthode](#)
[Copy, méthode](#)
[CopyFile, méthode](#)
[CopyFolder, méthode](#)
[CreateFolder, méthode](#)
[CreateTextFile, méthode](#)
[Delete, méthode](#)
[DeleteFile, méthode](#)
[DeleteFolder, méthode](#)
[DriveExists, méthode](#)
[Exists, méthode](#)
[FileExists, méthode](#)
[FolderExists, méthode](#)
[GetAbsolutePathName, méthode](#)
[GetBaseName, méthode](#)
[GetDrive, méthode](#)
[GetDriveName, méthode](#)
[GetExtensionName, méthode](#)
[GetFile, méthode](#)
[GetFileName, méthode](#)
[GetFileVersion, méthode](#)
[GetFolder, méthode](#)
[GetParentFolderName, méthode](#)
[GetSpecialFolder, méthode](#)
[GetTempName, méthode](#)
[Items, méthode](#)
[Keys, méthode](#)

[Move, méthode](#)
[MoveFile, méthode](#)
[MoveFolder, méthode](#)
[OpenAsTextStream, méthode](#)
[OpenTextFile, méthode](#)
[Read, méthode](#)
[ReadAll, méthode](#)
[ReadLine, méthode](#)
[Remove, méthode](#)
[RemoveAll, méthode](#)
[Skip, méthode](#)
[SkipLine, méthode](#)
[Write, méthode](#)
[WriteBlankLines, méthode](#)
[WriteLine, méthode](#)

Objets

[Dictionary, objet](#)

[Drive, objet](#)

[Drives, collection](#)

[File, objet](#)

[Files, collection](#)

[FileSystemObject, objet](#)

[Folder, objet](#)

[Folders, collection](#)

[TextStream, objet](#)

Propriétés

[AtEndOfLine, propriété](#)
[AtEndOfStream, propriété](#)
[Attributes, propriété](#)
[AvailableSpace, propriété](#)
[Column, propriété](#)
[CompareMode, propriété](#)
[Count, propriété](#)
[DateCreated, propriété](#)
[DateLastAccessed, propriété](#)
[DateLastModified, propriété](#)
[Drive, propriété](#)
[DriveLetter, propriété](#)
[Drives, propriété](#)
[DriveType, propriété](#)
[Files, propriété](#)
[FileSystem, propriété](#)
[FreeSpace, propriété](#)
[IsReady, propriété](#)
[IsRootFolder, propriété](#)
[Item, propriété](#)
[Key, propriété](#)
[Line, propriété](#)
[Name, propriété](#)
[ParentFolder, propriété](#)
[Path, propriété](#)
[RootFolder, propriété](#)
[SerialNumber, propriété](#)
[ShareName, propriété](#)
[ShortName, propriété](#)
[ShortPath, propriété](#)

[Size, propriété](#)

[SubFolders, propriété](#)

[TotalSize, propriété](#)

[Type, propriété](#)

[VolumeName, propriété](#)

Le modèle d'objet FileSystemObject

[Suivant](#)

Lorsque vous écrivez des scripts pour les pages ASP, pour Windows Scripting Host, ou pour d'autres applications utilisant les scripts, il est souvent important d'ajouter, de déplacer, de modifier, de créer ou de supprimer des dossiers (répertoires) et des fichiers sur le serveur Web. Il peut également se révéler nécessaire de lire des informations relatives aux lecteurs et de manipuler des lecteurs connectés au serveur Web.

Scripting vous permet de gérer les lecteurs, les dossiers et les fichiers grâce au modèle d'objet **FileSystemObject** (FSO) décrit dans les sections suivantes :

- [Présentation du FileSystemObject et référence de la bibliothèque d'exécution Scripting](#)
 - [Objets du FileSystemObject](#)
 - [Programmation du FileSystemObject](#)
 - [Utilisation des lecteurs et des dossiers](#)
 - [Utilisation des fichiers](#)
 - [Exemple de code du FileSystemObject](#)
-

Présentation du FileSystemObject et de la référence de la bibliothèque d'exécution Scripting

[Précédent](#)
[Suivant](#)

Le modèle d'objet **FileSystemObject** (FSO) vous permet d'utiliser la syntaxe familière *object.method* avec un ensemble riche de propriétés, de méthodes et d'événements pour gérer les dossiers et les fichiers.

Utilisez cet outil à base d'objets avec :

- HTML, pour créer des pages Web ;
- Windows Scripting Host, pour créer des fichiers de commande pour Microsoft Windows ;
- Script Control, pour fournir des fonctionnalités de script aux applications développées dans un autre langage.

Parce que l'utilisation du FSO du côté client soulève de questions importantes relatives à la sécurité en raison de l'accès au système de fichiers du client, cette documentation suppose que le modèle d'objet FSO est utilisé pour créer des scripts exécutés par des pages Web Internet côté serveur. Les opérations étant effectuées côté serveur, les paramètres de sécurité par défaut d'Internet Explorer n'autorisent pas l'utilisation de l'objet **FileSystemObject** côté client. La modification de ces paramètres induit pour l'ordinateur local un risque d'accès inopportun au système de fichiers qui peut se traduire par la destruction de l'intégrité du système de fichiers, la perte de données ou pire.

Le modèle d'objet FSO offre à vos applications côté serveur la possibilité de créer, modifier, déplacer et supprimer les dossiers, ou de détecter l'existence de dossiers particuliers et, le cas échéant, leur emplacement. Vous pouvez également lire des informations sur les dossiers, comme leur nom, leur date de création ou de dernière modification, etc.

Le modèle d'objet FSO facilite également le traitement des fichiers. Lors du traitement des fichiers, l'objectif principal consiste à stocker des données dans un format facile d'accès et efficace du point de vue des ressources comme de l'espace. Vous devez pouvoir créer des fichiers, insérer, modifier et sortir (lire) les données. Parce que le stockage dans une base de données, comme Access ou SQL Server, ajoute à votre application une charge significative, le stockage dans un fichier texte ou binaire peut être la solution la plus efficace. Vous pouvez souhaiter éviter cette charge ou vos besoins d'accès aux données n'ont peut-être pas besoin de toute la fonctionnalité d'une véritable base de données.

Le modèle d'objet FSO, qui est contenu dans la [bibliothèque de types](#) de Scripting (Scrrun.dll), gère la création et la manipulation des fichiers de texte par l'intermédiaire de l'objet **TextStream**. Bien que la création et la manipulation de fichiers binaires ne soient pas gérés pour l'instant, elles le seront bientôt.

Objets du FileSystemObject

[Précédent](#)
[Suivant](#)

Le modèle d'objet **FileSystemObject** (FSO) contient les [collections](#) et objets suivants.

Objet/Collection	Description
FileSystemObject	Objet principal. Il contient des méthodes et des propriétés qui vous permettent de créer, supprimer, interroger et plus généralement de manipuler des lecteurs, des dossiers et des fichiers. Un grand nombre des méthodes associées à cet objet dupliquent celles d'autres objets FSO ; elles sont donc fournies à titre pratique.
Drive	Objet. Il contient des méthodes et des propriétés qui vous permettent de réunir des informations relatives à un lecteur connecté au système, comme son nom de partage et sa quantité d'espace libre disponible. Notez qu'un "lecteur" n'est pas obligatoirement un disque dur ; il peut s'agir d'un lecteur de CD-ROM, d'un disque virtuel, etc. Un lecteur n'est pas obligatoirement physiquement connecté au système, il peut être connecté logiquement par l'intermédiaire du réseau.
Drives	Collection. Elle fournit la liste des lecteurs connectés au système, physiquement ou logiquement. La collection Drives comprend tous les lecteurs, quel que soit leur type. Les lecteurs à support amovible apparaissent dans cette collection même lorsqu'ils sont vides.

File	Objet. Il contient des méthodes et des propriétés qui vous permettent de créer, supprimer ou déplacer un fichier. Il vous permet également de rechercher dans le système un nom de fichier, un chemin et d'autres propriétés diverses.
Files	Collection. Elle fournit la liste des fichiers d'un dossier.
Folder	Objet. Il contient des méthodes et des propriétés qui vous permettent de créer, supprimer ou déplacer des dossiers. Il vous permet également de rechercher dans le système un nom de dossier, un chemin et d'autres propriétés diverses.
Folders	Collection. Elle fournit la liste des dossiers d'un Folder .
TextStream	Objet. Il vous permet de lire et d'écrire dans des fichiers textes.

Programmation du FileSystemObject

[Précédent](#)
[Suivant](#)

Pour programmer avec le modèle d'objet **FileSystemObject** (FSO) :

- Utilisez la méthode **CreateObject** pour créer un objet **FileSystemObject**.
- Utilisez la méthode appropriée sur l'objet nouvellement créé.
- Accédez aux propriétés de l'objet.

Le modèle d'objet FSO est contenu dans la [bibliothèque de types](#) de Scripting, qui se situe dans le fichier Sccrun.dll. Le fichier Sccrun.dll doit, par conséquent, figurer dans le répertoire système approprié de votre serveur Web pour pouvoir utiliser le modèle d'objet FSO.

Création d'un objet FileSystemObject

Commencez par créer un objet **FileSystemObject** en utilisant la méthode **CreateObject**. Dans VBScript, utilisez le code suivant pour créer une instance de **FileSystemObject** :

```
Dim fso  
Set fso = CreateObject("Scripting.FileSystemObject")
```

Cet [exemple de code](#) montre la création d'une instance du **FileSystemObject**.

Dans JScript, utilisez ce code pour effectuer la même opération :

```
var fso;  
fso = new ActiveXObject("Scripting.FileSystemObject");
```

Dans ces deux exemples, **Scripting** est le nom de la bibliothèque de types

et **FileSystemObject** est le nom de l'objet que nous voulons créer. Vous ne pouvez créer qu'une seule instance de l'objet **FileSystemObject**, quel que soit le nombre de tentatives de création.

Utilisation de la méthode appropriée

Ensuite, utilisez la méthode appropriée de l'objet **FileSystemObject**. Par exemple, pour créer un nouvel objet, utilisez **CreateTextFile** ou **CreateFolder** (le modèle d'objet FSO ne gère pas la création ou la suppression de lecteurs).

Pour supprimer des objets, utilisez les méthodes **DeleteFile** et **DeleteFolder** de l'objet **FileSystemObject** ou la méthode **Delete** des objets **File** et **Folder**. Vous pouvez aussi copier et déplacer des fichiers et des dossiers par l'intermédiaire des méthodes appropriées.

Remarque Certaines fonctionnalités du modèle d'objet **FileSystemObject** sont redondantes. Par exemple, vous pouvez copier un fichier par l'intermédiaire de la méthode **CopyFile** de l'objet **FileSystemObject** ou de la méthode **Copy** de l'objet **File**. Ces méthodes fonctionnent de la même façon. Leur présence offre une plus grande flexibilité de programmation.

Accès aux dossiers, fichiers et lecteurs existants

Pour accéder à un lecteur, à un fichier ou à un dossier existant, utilisez la méthode "get" appropriée ou l'objet **FileSystemObject** :

- **GetDrive**
- **GetFolder**
- **GetFile**

Pour accéder à un fichier existant dans VBScript :

```
Dim fso, f1  
Set fso = CreateObject("Scripting.FileSystemObject")
```

```
Set f1 = fso.GetFile("c:\test.txt")
```

Pour effectuer la même opération dans JScript, utilisez le code ci-dessous :

```
var fso, f1;  
fso = new ActiveXObject("Scripting.FileSystemObject");  
f1 = fso.GetFile("c:\\test.txt");
```

N'utilisez pas les méthodes "get" pour les objets nouvellement créés, puisque les fonctions "create" ont déjà renvoyé un descripteur vers cet objet. Par exemple, si vous créez un nouveau dossier en utilisant la méthode **CreateFolder**, n'utilisez pas la méthode **GetFolder** pour accéder à ses propriétés comme **Name**, **Path**, **Size**, etc. Affectez juste à une variable la fonction **CreateFolder** pour obtenir un descripteur vers le dossier nouvellement créé, puis accédez à ses propriétés, méthodes et événements. Pour effectuer la même opération dans VBScript, utilisez le code ci-dessous :

```
Sub CreateFolder  
  Dim fso, fldr  
  Set fso = CreateObject("Scripting.FileSystemObject")  
  Set fldr = fso.CreateFolder("C:\MyTest")  
  Response.Write "Dossier créé: " & fldr.Name  
End Sub
```

Pour affecter à une variable la fonction **CreateFolder** dans JScript, utilisez la syntaxe suivante :

```
function CreateFolder()  
{  
  var fso, fldr;  
  fso = new ActiveXObject("Scripting.FileSystemObject");  
  fldr = fso.CreateFolder("C:\\MyTest");  
  Response.Write("Created folder: " + fldr.Name);  
}
```

Accès aux propriétés de l'objet

À partir du descripteur d'un objet, vous avez accès à ses propriétés. Par exemple, pour obtenir le nom d'un dossier particulier, commencez par créer une instance de l'objet, puis obtenez un descripteur vers cet objet en utilisant la méthode appropriée (dans ce cas, la méthode **GetFolder** puisque le dossier existe déjà).

Utilisez ce code pour obtenir un descripteur vers la méthode **GetFolder** dans VBScript :

```
Set fldr = fso.GetFolder("c:\")
```

Pour effectuer la même opération dans JScript, utilisez le code ci-dessous :

```
var fldr = fso.GetFolder("c:\\");
```

Vous disposez désormais d'un descripteur vers un objet **Folder** ; vous pouvez lire sa propriété **Name**. Pour effectuer la même opération dans VBScript, utilisez le code ci-dessous :

```
Response.Write "Le nom du dossier est: " & fldr.Name
```

Pour lire la valeur d'une propriété **Name** dans JScript, utilisez la syntaxe suivante :

```
Response.Write("Le nom du dossier est: " + fldr.Name);
```

Pour connaître la date de dernière modification d'un fichier, utilisez la syntaxe VBScript suivante :

```
Dim fso, f1
Set fso = CreateObject("Scripting.FileSystemObject")
' Obtient un objet File à interroger.
Set f1 = fso.GetFile("c:\detlog.txt")
' Affiche les informations.
Response.Write "Dernière modification du fichier: " & f1.DateLastModified
```

Pour effectuer la même opération dans JScript, utilisez le code ci-dessous :

```
var fso, f1;
fso = new ActiveXObject("Scripting.FileSystemObject");
// Obtient un objet File à interroger.
f1 = fso.GetFile("c:\\detlog.txt");
// Affiche les informations.
Response.Write("Dernière modification du fichier: " + f1.DateLast
```

Utilisation des lecteurs et des dossiers

[Précédent](#)
[Suivant](#)

Le modèle d'objet **FileSystemObject** (FSO) vous permet d'utiliser dans vos programmes les lecteurs et les dossiers comme vous le faites déjà de façon interactive dans l'Explorateur de Windows. Vous pouvez copier et déplacer des dossiers, lire des informations relatives aux dossiers et aux lecteurs, etc.

Lecture des informations relatives aux lecteurs

L'objet **Drive** vous permet d'obtenir des informations relatives aux différents lecteurs connectés à un système physiquement ou à travers le réseau. Ses propriétés vous apportent les informations suivantes :

- La taille totale du lecteur en octets (**TotalSize**, propriété).
- L'espace disponible sur le lecteur en octets (**AvailableSpace** et **FreeSpace**, propriétés).
- La lettre affectée au lecteur (**DriveLetter**, propriété).
- Le type du lecteur, c'est-à-dire s'il est amovible, fixe, réseau, CD-ROM ou virtuel (**DriveType**, propriété).
- Le numéro de série du lecteur (**SerialNumber**, propriété).
- Le type de système de fichiers utilisé par le lecteur, comme FAT, FAT32, NTFS et autre (**FileSystem**, propriété).
- La disponibilité d'un lecteur (**IsReady**, propriété).
- Le nom du partage/ou du volume (**ShareName** et **VolumeName**, propriétés).

- Le chemin ou le dossier racine du lecteur (**Path** et **RootFolder**, propriétés).

Observez dans l'[exemple de code](#) la mise en œuvre de ces propriétés dans le **FileSystemObject**.

Exemple d'utilisation de l'objet Drive

Utilisez l'objet **Drive** pour obtenir des informations sur un lecteur. Le code ci-dessous ne présente pas de référence à un objet **Drive** ; utilisez plutôt la méthode **GetDrive** pour obtenir une référence à un objet **Drive** existant (dans le cas présent drv).

L'exemple ci-dessous illustre l'utilisation de l'objet **Drive** dans VBScript :

```
Sub ShowDriveInfo(drvPath)
    Dim fso, drv, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set drv = fso.GetDrive(fso.GetDriveName(drvPath))
    s = "Lecteur " & UCase(drvPath) & " - "
    s = s & drv.VolumeName & "<br>"
    s = s & "Espace total: " & FormatNumber(drv.TotalSize / 1024, 0)
    s = s & " Ko" & "<br>"
    s = s & "Espace disponible: " & FormatNumber(drv.FreeSpace /
    s = s & " Ko" & "<br>"
    Response.Write s
End Sub
```

Le code suivant illustre la même fonctionnalité dans JScript :

```
function ShowDriveInfo1(drvPath)
{
    var fso, drv, s = "";
    fso = new ActiveXObject("Scripting.FileSystemObject");
    drv = fso.GetDrive(fso.GetDriveName(drvPath));
    s += "Lecteur " + drvPath.toUpperCase() + " - ";
```

```

s += drv.VolumeName + "<br>";
s += "Espace total: " + drv.TotalSize / 1024;
s += " Ko" + "<br>";
s += "Espace disponible: " + drv.FreeSpace / 1024;
s += " Ko" + "<br>";
Response.Write(s);
}

```

Utilisation des dossiers

Le tableau ci-dessous présente des opérations sur les dossiers courants et les méthodes correspondantes.

Tâche	Méthode
Créer un dossier.	FileSystemObject.CreateFolder
Supprimer un dossier.	Folder.Delete ou FileSystemObject.DeleteFolder
Déplacer un dossier.	Folder.Move ou FileSystemObject.MoveFolder
Copier un dossier.	Folder.Copy ou FileSystemObject.CopyFolder
Extraire le nom d'un dossier.	Folder.Name
Déterminer l'existence d'un dossier sur un lecteur.	FileSystemObject.FolderExists
Obtenir une instance d'un objet Folder existant.	FileSystemObject.GetFolder
Déterminer le nom du dossier parent d'un dossier.	FileSystemObject.GetParentFolderName
Déterminer le	

chemin des dossiers | **FileSystemObject.GetSpecialFolder** |
systèmes.

Observez dans l'[exemple de code](#) la mise en œuvre de ces méthodes et propriétés dans le **FileSystemObject**.

L'exemple ci-dessous illustre l'utilisation des objets **Folder** et **FileSystemObject** pour manipuler les dossiers et obtenir des informations les concernant dans VBScript :

```
Sub ShowFolderInfo()  
  Dim fso, fldr, s  
  ' Obtient une instance du FileSystemObject.  
  Set fso = CreateObject("Scripting.FileSystemObject")  
  ' Obtient l'objet Drive.  
  Set fldr = fso.GetFolder("c:")  
  ' Affiche le nom du dossier parent.  
  Response.Write "Le nom du dossier parent est: " & fldr & "<br>"  
  ' Affiche le nom du lecteur.  
  Response.Write "Résidant sur le lecteur " & fldr.Drive & "<br>"  
  ' Affiche le nom de la racine.  
  If fldr.IsRootFolder = True Then  
    Response.Write "Ceci est le dossier racine." & ""<br><br>"  
  Else  
    Response.Write "Ce dossier n'est pas un dossier racine." & "<br>"  
  End If  
  ' Crée un nouveau dossier avec l'objet FileSystemObject.  
  fso.CreateFolder ("C:\Bogus")  
  Response.Write "Dossier C:\Bogus créé" & "<br>"  
  ' Affiche le nom de base du dossier.  
  Response.Write "Nom de base = " & fso.GetBaseName("c:\bogus")  
  ' Supprime le dossier nouvellement créé.  
  fso.DeleteFolder ("C:\Bogus")  
  Response.Write "Dossier C:\Bogus supprimé" & "<br>"  
End Sub
```

Cet exemple illustre l'utilisation des objets **Folder** et **FileSystemObject**

dans JScript :

```
function ShowFolderInfo()
{
    var fso, fldr, s = "";
    // Obtient une instance du FileSystemObject.
    fso = new ActiveXObject("Scripting.FileSystemObject");
    // Obtient l'objet Drive.
    fldr = fso.GetFolder("c:");
    // Affiche le nom du dossier parent.
    Response.Write("Le nom du dossier parent est: " + fldr + "<br>");
    // Affiche le nom du lecteur.
    Response.Write("Résidant sur le lecteur " + fldr.Drive + "<br>");
    // Affiche le nom de la racine.
    if (fldr.IsRootFolder)
        Response.Write("Ceci est le dossier racine.");
    else
        Response.Write("Ce dossier n'est pas un dossier racine.");
    Response.Write("<br><br>");
    // Crée un nouveau dossier avec l'objet FileSystemObject.
    fso.CreateFolder ("C:\\Bogus");
    Response.Write("Dossier C:\\Bogus créé" + "<br>");
    // Affiche le nom de base du dossier.
    Response.Write("Nom de base = " + fso.GetBaseName("c:\\bogu
    // Supprime le dossier nouvellement créé.
    fso.DeleteFolder ("C:\\Bogus");
    Response.Write("Dossier C:\\Bogus supprimé" + "<br>");
}
```

FileSystemObject

Exemple de code

[Précédent](#)

L'exemple de code décrit dans cette section offre un exemple réaliste des nombreuses fonctionnalités offertes par le modèle d'objet **FileSystemObject**. Ce code illustre l'emploi de ces fonctionnalités du modèle d'objet ainsi que leur mise en œuvre de façon efficace dans votre propre code.

Remarquez que ce code est relativement générique et que des adaptations mineures seront nécessaires pour qu'il puisse s'exécuter effectivement sur votre machine. Ces modifications sont requises en raison des différentes mises en œuvre des entrées et des sorties vers l'utilisateur entre les pages ASP et l'hôte Windows Scripting Host.

Pour exécuter ce code sur une page ASP, utilisez la procédure suivante :

1. Créez une page Web standard avec une extension .asp.
2. Copiez l'exemple de code suivant dans ce fichier, encadré de balises `<BODY;>...</BODY>`.
3. Encadrez tout le code entre les balises `<%...%>`.
4. Déplacez l'instruction **Option Explicit** de sa position actuelle dans le code vers le haut de votre page HTML, avant la balise d'ouverture `<HTML>`.
5. Placez des balises `<%...%>` autour de l'instruction **Option Explicit** pour qu'elle soit exécutée du côté serveur.
6. Ajoutez le code suivant à la fin de l'exemple de code :

```
Sub Print(x)
  Response.Write "<PRE><FONT; FACE=""Courier New"" SIZE=""1"">"
  Response.Write x
  Response.Write "</FONT></PRE>"
End Sub
Main
```

Le code ci-dessus ajoute une procédure d'impression qui s'exécute du côté serveur mais affiche ses résultats du côté client. Pour exécuter ce code sur Windows Scripting Host, ajoutez le code suivant à la fin de l'exemple de code :

```
Sub Print(x)
    WScript.Echo x
End Sub
Main
```

Le code est contenu dans la section suivante :

```
.....
'
' Exemple de code du FileSystemObject
'
' Copyright 1998 Microsoft Corporation. Tous droits réservés.
'
.....
```

Option Explicit

```
.....
'
' Remarques sur la qualité du code :
'
' 1) Ce code effectue de nombreuses manipulations de chaînes en
' concaténant de courtes chaînes à l'aide de l'opérateur "&".
' En raison du coût que représentent les opérations de concaténation, l'écriture de ce
' code selon cette méthode n'est pas très efficace.
' Cette méthode facilite néanmoins
' la maintenance et, dans le cas présent, les nombreuses
' opérations disques effectuées sont de toute façon beaucoup
' plus lentes que les opérations de concaténation de chaînes.
' N'oubliez pas qu'il s'agit de code de démonstration, et non
' pas de production.
'
' 2) L'option "Option Explicit" permet d'accélérer légèrement
' l'accès aux variables. Elle permet également d'éviter
' certaines erreurs, notamment les fautes de frappe telles
' que DriveTypeCDORM à la place de DriveTypeCDROM.
'
' 3) Ce code ne comprend pas de gestion d'erreur afin de simplifier
' sa lecture. Malgré les précautions prises afin qu'il
' fonctionne dans les cas courants, les systèmes de fichiers
' restent imprévisibles. En production, utilisez On Error
```

```
' Resume Next et l'objet Err afin d'intercepter les erreurs  
' éventuelles.  
'  
.....  
  
.....  
'  
' Variables globales pratiques  
'  
.....
```

```
Dim TabStop  
Dim NewLine
```

```
Const TestDrive = "C"  
Const TestFilePath = "C:\Test"
```

```
.....  
'
```

```
' Constantes renvoyées par Drive.DriveType  
'  
.....
```

```
Const DriveTypeRemovable = 1  
Const DriveTypeFixed = 2  
Const DriveTypeNetwork = 3  
Const DriveTypeCDROM = 4  
Const DriveTypeRAMDisk = 5
```

```
.....  
'
```

```
' Constantes renvoyées par File.Attributes  
'  
.....
```

```
Const FileAttrNormal = 0  
Const FileAttrReadOnly = 1  
Const FileAttrHidden = 2  
Const FileAttrSystem = 4  
Const FileAttrVolume = 8  
Const FileAttrDirectory = 16  
Const FileAttrArchive = 32  
Const FileAttrAlias = 64  
Const FileAttrCompressed = 128
```

```
.....  
'
```

```
' Constantes d'ouverture de fichiers  
'  
.....
```

```
Const OpenFileForReading = 1
Const OpenFileForWriting = 2
Const OpenFileForAppending = 8
```

```
.....
'
' ShowDriveType
'
' Objet :
'
' Génère une chaîne décrivant le type d'un objet Drive donné.
'
' Présente
'
' - Drive.DriveType
'
.....
```

```
Function ShowDriveType(Drive)
```

```
    Dim S

    Select Case Drive.DriveType
    Case DriveTypeRemovable
        S = "Amovible"
    Case DriveTypeFixed
        S = "Fixe"
    Case DriveTypeNetwork
        S = "Réseau"
    Case DriveTypeCDROM
        S = "CD-ROM"
    Case DriveTypeRAMDisk
        S = "Disque virtuel"
    Case Else
        S = "Inconnu"
    End Select

    ShowDriveType = S
```

```
End Function
```

```
.....
'
' ShowFileAttr
'
' Objet :
'
' Génère une chaîne décrivant les attributs d'un fichier ou d'un dossier.
```

```

'
' Présente
'
' - File.Attributes
' - Folder.Attributes
'
.....

Function ShowFileAttr(File) ' File représente un fichier ou un dossier

    Dim S
    Dim Attr

    Attr = File.Attributes

    If Attr = 0 Then
        ShowFileAttr = "Normal"
        Exit Function
    End If

    If Attr And FileAttrDirectory Then S = S & "Répertoire "
    If Attr And FileAttrReadOnly Then S = S & "En lecture seule "
    If Attr And FileAttrHidden Then S = S & "Caché "
    If Attr And FileAttrSystem Then S = S & "Système "
    If Attr And FileAttrVolume Then S = S & "Volume "
    If Attr And FileAttrArchive Then S = S & "Archive "
    If Attr And FileAttrAlias Then S = S & "Alias "
    If Attr And FileAttrCompressed Then S = S & "Compressé "

    ShowFileAttr = S

End Function

.....

'
' GenerateDriveInformation
'
' Objet :
'
' Génère une chaîne décrivant l'état actuel des lecteurs disponibles.
'
' Présente
'
' - FileSystemObject.Drives
' - L'itération dans la collection Drives
' - Drives.Count
' - Drive.AvailableSpace
' - Drive.DriveLetter
' - Drive.DriveType

```

```
' - Drive.FileSystem
' - Drive.FreeSpace
' - Drive.IsReady
' - Drive.Path
' - Drive.SerialNumber
' - Drive.ShareName
' - Drive.TotalSize
' - Drive.VolumeName
'
```

```
.....
```

Function GenerateDriveInformation(FSO)

```
Dim Drives
Dim Drive
Dim S
```

```
Set Drives = FSO.Drives
```

```
S = "Nombre de lecteurs:" & TabStop & Drives.Count & NewLine & NewLine
```

```
' Construit la première ligne du compte-rendu.
```

```
S = S & String(2, TabStop) & "Lecteur"
```

```
S = S & String(3, TabStop) & "Fichier"
```

```
S = S & TabStop & "Total"
```

```
S = S & TabStop & "Libre"
```

```
S = S & TabStop & "Disponible"
```

```
S = S & TabStop & "Série" & NewLine
```

```
' Construit la seconde ligne de l'état.
```

```
S = S & "Lettre"
```

```
S = S & TabStop & "Chemin"
```

```
S = S & TabStop & "Type"
```

```
S = S & TabStop & "Prêt?"
```

```
S = S & TabStop & "Nom"
```

```
S = S & TabStop & "Système"
```

```
S = S & TabStop & "Espace"
```

```
S = S & TabStop & "Espace"
```

```
S = S & TabStop & "Espace"
```

```
S = S & TabStop & "Numéro" & NewLine
```

```
' Ligne de séparation.
```

```
S = S & String(105, "-") & NewLine
```

```
For Each Drive In Drives
```

```
    S = S & Drive.DriveLetter
```

```
    S = S & TabStop & Drive.Path
```

```
    S = S & TabStop & ShowDriveType(Drive)
```

```
    S = S & TabStop & Drive.IsReady
```

```

    If Drive.IsReady Then
    If DriveTypeNetwork = Drive.DriveType Then
        S = S & TabStop & Drive.ShareName
    Else
        S = S & TabStop & Drive.VolumeName
    End If

    S = S & TabStop & Drive.FileSystem
    S = S & TabStop & Drive.TotalSize
    S = S & TabStop & Drive.FreeSpace
    S = S & TabStop & Drive.AvailableSpace
    S = S & TabStop & Hex(Drive.SerialNumber)

    End If

    S = S & NewLine

Next

GenerateDriveInformation = S

```

End Function

```

'-----
'
' GenerateFileInformation
'
' Objet :
'
' Génère une chaîne décrivant l'état actuel d'un fichier.
'
' Présente
'
' - File.Path
' - File.Name
' - File.Type
' - File.DateCreated
' - File.DateLastAccessed
' - File.DateLastModified
' - File.Size
'
'-----

```

Function GenerateFileInformation(File)

Dim S

```

S = NewLine & "Chemin:" & TabStop & File.Path
S = S & NewLine & "Nom:" & TabStop & File.Name

```

```
S = S & NewLine & "Type:" & TabStop & File.Type  
S = S & NewLine & "Attribs:" & TabStop & ShowFileAttr(File)  
S = S & NewLine & "Cr   le:" & TabStop & File.DateCreated  
S = S & NewLine & "Acc   le:" & TabStop & File.DateLastAccessed  
S = S & NewLine & "Modif:" & TabStop & File.DateLastModified  
S = S & NewLine & "Taille" & TabStop & File.Size & NewLine
```

```
GenerateFileInformation = S
```

End Function

```
.....  
,  
' GenerateFolderInformation  
,  
' Objet :  
,  
' G  n  re une cha  ne d  crivant l'  tat actuel d'un dossier.  
,  
' Pr  sente  
,  
' - Folder.Path  
' - Folder.Name  
' - Folder.DateCreated  
' - Folder.DateLastAccessed  
' - Folder.DateLastModified  
' - Folder.Size  
,  
.....
```

Function GenerateFolderInformation(Folder)

```
Dim S
```

```
S = "Chemin:" & TabStop & Folder.Path  
S = S & NewLine & "Nom:" & TabStop & Folder.Name  
S = S & NewLine & "Attribs:" & TabStop & ShowFileAttr(Folder)  
S = S & NewLine & "Cr   le:" & TabStop & Folder.DateCreated  
S = S & NewLine & "Acc   le:" & TabStop & Folder.DateLastAccessed  
S = S & NewLine & "Modif:" & TabStop & Folder.DateLastModified  
S = S & NewLine & "Taille:" & TabStop & Folder.Size & NewLine
```

```
GenerateFolderInformation = S
```

End Function

```
.....  
,  
' GenerateAllFolderInformation
```

```
'  
' Objet :  
'  
' Génère une chaîne décrivant l'état actuel d'un  
' dossier et de ses fichiers et sous-dossiers.  
'  
' Présente  
'  
' - Folder.Path  
' - Folder.SubFolders  
' - Folders.Count  
'
```

.....

Function GenerateAllFolderInformation(Folder)

```
Dim S  
Dim SubFolders  
Dim SubFolder  
Dim Files  
Dim File
```

```
S = "Dossier:" & TabStop & Folder.Path & NewLine & NewLine
```

```
Set Files = Folder.Files
```

```
If 1 = Files.Count Then
```

```
    S = S & "Il y a 1 fichier" & NewLine
```

```
Else
```

```
    S = S & "Il y a " & Files.Count & " fichiers" & NewLine
```

```
End If
```

```
If Files.Count <> 0 Then
```

```
    For Each File In Files
```

```
        S = S & GenerateFileInformation(File)
```

```
    Next
```

```
End If
```

```
Set SubFolders = Folder.SubFolders
```

```
If 1 = SubFolders.Count Then
```

```
    S = S & NewLine & "Il y a 1 sous-dossier" & NewLine & NewLine
```

```
Else
```

```
    S = S & NewLine & "Il y a " & SubFolders.Count & " sous-dossiers" & NewLine & NewLine
```

```
End If
```

```
If SubFolders.Count <> 0 Then
```

```

    For Each SubFolder In SubFolders
        S = S & GenerateFolderInformation(SubFolder)
    Next

    S = S & NewLine

    For Each SubFolder In SubFolders
        S = S & GenerateAllFolderInformation(SubFolder)
    Next

End If

GenerateAllFolderInformation = S

End Function

'-----
'
' GenerateTestInformation
'
' Objet :
'
' Génère une chaîne décrivant l'état actuel du dossier C:\Test
' et de ses fichiers et sous-dossiers.
'
' Présente
'
' - FileSystemObject.DriveExists
' - FileSystemObject.FolderExists
' - FileSystemObject.GetFolder
'
'-----

Function GenerateTestInformation(FSO)

    Dim TestFolder
    Dim S

    If Not FSO.DriveExists(TestDrive) Then Exit Function
    If Not FSO.FolderExists(TestFilePath) Then Exit Function

    Set TestFolder = FSO.GetFolder(TestFilePath)

    GenerateTestInformation = GenerateAllFolderInformation(TestFolder)

End Function

'-----
'
```

```
' DeleteTestDirectory
'
' Objet :
'
' Nettoie le répertoire de test.
'
' Présente
'
' - FileSystemObject.GetFolder
' - FileSystemObject.DeleteFile
' - FileSystemObject.DeleteFolder
' - Folder.Delete
' - File.Delete
'
```

.....

```
Sub DeleteTestDirectory(FSO)
```

```
    Dim TestFolder
    Dim SubFolder
    Dim File
```

```
    ' Deux façons de supprimer un fichier:
```

```
    FSO.DeleteFile(TestFilePath & "\Beatles\OctopusGarden.txt")
```

```
    Set File = FSO.GetFile(TestFilePath & "\Beatles\BathroomWindow.txt")
    File.Delete
```

```
    ' Deux façons de supprimer un dossier:
```

```
    FSO.DeleteFolder(TestFilePath & "\Beatles")
```

```
    FSO.DeleteFile(TestFilePath & "\ReadMe.txt")
```

```
    Set TestFolder = FSO.GetFolder(TestFilePath)
    TestFolder.Delete
```

```
End Sub
```

.....

```
' CreateLyrics
```

```
' Objet :
```

```
' Crée deux fichiers texte dans un dossier.
```

```

'
' Présente
'
' - FileSystemObject.CreateTextFile
' - TextStream.WriteLine
' - TextStream.Write
' - TextStream.WriteLine
' - TextStream.Close
'
'.....

Sub CreateLyrics(Folder)

    Dim TextStream

    Set TextStream = Folder.CreateTextFile("OctopusGarden.txt")

    TextStream.Write("Octopus' Garden ") ' Remarque que ceci n'ajoute pas de saut de ligne dans le fic
    TextStream.WriteLine("(par Ringo Starr)")
    TextStream.WriteLine(1)
    TextStream.WriteLine("I'd like to be under the sea in an octopus' garden in the shade,")
    TextStream.WriteLine("He'd let us in, knows where we've been -- in his octopus' garden in the shade")
    TextStream.WriteLine(2)

    TextStream.Close

    Set TextStream = Folder.CreateTextFile("BathroomWindow.txt")
    TextStream.WriteLine("She Came In Through The Bathroom Window (par Lennon/McCartney)")
    TextStream.WriteLine("")
    TextStream.WriteLine("She came in through the bathroom window protected by a silver spoon")
    TextStream.WriteLine("But now she sucks her thumb and wanders by the banks of her own lagoon")
    TextStream.WriteLine(2)
    TextStream.Close

End Sub

'.....

'
' GetLyrics
'
' Objet :
'
' Affiche le contenu des fichiers des paroles.
'
'
' Présente
'
' - FileSystemObject.OpenTextFile
' - FileSystemObject.GetFile
' - TextStream.ReadAll

```

```
' - TextStream.Close
' - File.OpenAsTextStream
' - TextStream.AtEndOfStream
' - TextStream.ReadLine
'
```

.....

Function GetLyrics(FSO)

```
Dim TextStream
Dim S
Dim File
```

```
' Il existe plusieurs moyens d'ouvrir un fichier texte et
' plusieurs moyens de lire les données d'un fichier.
' Voici deux façons d'effectuer chaque opération:
```

```
Set TextStream = FSO.OpenTextFile(TestFilePath & "\Beatles\OctopusGarden.txt", OpenFileForRe
```

```
S = TextStream.ReadAll & NewLine & NewLine
TextStream.Close
```

```
Set File = FSO.GetFile(TestFilePath & "\Beatles\BathroomWindow.txt")
Set TextStream = File.OpenAsTextStream(OpenFileForReading)
Do While Not TextStream.AtEndOfStream
    S = S & TextStream.ReadLine & NewLine
Loop
TextStream.Close
```

```
GetLyrics = S
```

End Function

.....

```
'
' BuildTestDirectory
'
' Objet :
'
' Crée une hiérarchie de répertoires pour exposer le FileSystemObject.
'
' Nous construirons une hiérarchie dans l'ordre suivant :
'
' C:\Test
' C:\Test\ReadMe.txt
' C:\Test\Beatles
' C:\Test\Beatles\OctopusGarden.txt
' C:\Test\Beatles\BathroomWindow.txt
'
```

```
'  
' Présente  
'  
' - FileSystemObject.DriveExists  
' - FileSystemObject.FolderExists  
' - FileSystemObject.CreateFolder  
' - FileSystemObject.CreateTextFile  
' - Folders.Add  
' - Folder.CreateTextFile  
' - TextStream.WriteLine  
' - TextStream.Close  
'  
.....
```

```
Function BuildTestDirectory(FSO)
```

```
    Dim TestFolder  
    Dim SubFolders  
    Dim SubFolder  
    Dim TextStream
```

```
    ' Quitte si (a) le lecteur n'existe pas ou, (b) si le répertoire construit existe déjà.
```

```
    If Not FSO.DriveExists(TestDrive) Then  
        BuildTestDirectory = False  
        Exit Function  
    End If
```

```
    If FSO.FolderExists(TestFilePath) Then  
        BuildTestDirectory = False  
        Exit Function  
    End If
```

```
    Set TestFolder = FSO.CreateFolder(TestFilePath)
```

```
    Set TextStream = FSO.CreateTextFile(TestFilePath & "\ReadMe.txt")  
    TextStream.WriteLine("Ma collection de paroles de chansons")  
    TextStream.Close
```

```
    Set SubFolders = TestFolder.SubFolders
```

```
    Set SubFolder = SubFolders.Add("Beatles")
```

```
    CreateLyrics SubFolder
```

```
    BuildTestDirectory = True
```

```
End Function
```

```
.....  
,  
' Routine principale  
,  
' Elle commence par créer un répertoire de test avec des sous-dossiers  
' et des fichiers.  
' Ensuite, elle affiche des informations relatives aux lecteurs de  
' disque disponibles et au répertoire de test, puis elle efface tout.  
,  
.....
```

Sub Main

Dim FSO

' Définit les données globales.

TabStop = Chr(9)

NewLine = Chr(10)

Set FSO = CreateObject("Scripting.FileSystemObject")

If Not BuildTestDirectory(FSO) Then

Print "Le répertoire de test existe déjà ou ne peut pas être créé. Impossible de continuer."

Exit Sub

End If

Print GenerateDriveInformation(FSO) & NewLine & NewLine

Print GenerateTestInformation(FSO) & NewLine & NewLine

Print GetLyrics(FSO) & NewLine & NewLine

DeleteTestDirectory(FSO)

End Sub

Caractéristiques de VBScript

[Référence du langage](#)

Catégorie	Mots clés
Gestions des tableaux	Array Dim , Private , Public , ReDim IsArray Erase LBound , UBound
Affectations	Set
Commentaires	Commentaires utilisant ' ou Rem
Constantes/Littéraux	Empty Nothing Null True , False
Structures de contrôle	Do...Loop For...Next For Each...Next If...Then...Else Select Case While...Wend With
Conversions	Abs Asc , AscB , AscW Chr , ChrB , ChrW CBool , CByte CCur , CDate CDBl , CInt CLng , CSng , CStr DateSerial , DateValue Hex , Oct Fix , Int Sgn TimeSerial , TimeValue
Dates/Heures	Date , Time DateAdd , DateDiff , DatePart DateSerial , DateValue Day , Month , MonthName Weekday , WeekdayName , Year Hour , Minute , Second Now

	TimeSerial , TimeValue
Déclarations	Class Const Dim , Private , Public , ReDim Function , Sub Property Get , Property Let , Property Set
Gestion des erreurs	On Error Err
Expressions	Eval Execute RegExp Replace Test
Mise en forme des chaînes	FormatCurrency FormatDateTime FormatNumber FormatPercent
Entrées/Sorties	InputBox LoadPicture MsgBox
Littéraux	Empty False Nothing Null True
Mathématiques	Atn , Cos , Sin , Tan Exp , Log , Sqr Randomize , Rnd
Divers	Eval , fonction Execute , instruction RGB , fonction
Objets	CreateObject Err Object GetObject RegExp
Opérateurs	Addition (+) , Soustraction (-) Élévation à la puissance (^) Modulo arithmétique (Mod) Multiplication (*) , Division (/) Division des entiers (\) Négation (-) Concaténation de chaînes (&) Égalité (=) , Différence (<>) Inférieur à (<) , Inférieur à ou égal à (<=) Supérieur à (>) Supérieur à ou égal à (>=) Is And , Or , Xor Eqv , Imp

Options	Option Explicit
Procédures	Call Function , Sub Property Get , Property Let , Property Set
Arrondis	Abs Int , Fix , Round Sgn
ID de moteur de script	ScriptEngine ScriptEngineBuildVersion ScriptEngineMajorVersion ScriptEngineMinorVersion
Chaînes	Asc , AscB , AscW Chr , ChrB , ChrW Filter , InStr , InStrB InStrRev Join Len , LenB LCase , UCase Left , LeftB Mid , MidB Right , RightB Replace Space Split StrComp String StrReverse LTrim , RTrim , Trim
Variants	IsArray IsDate IsEmpty IsNull IsNumeric IsObject TypeName VarType

Caractéristiques Visual Basic pour Applications non incluses dans VBScript

[Référence du langage](#)

Catégorie	Caractéristique/Mot clé absent
Gestion des tableaux	Option Base Déclaration de tableau avec limite inférieure <> 0
Collection	Add, Count, Item, Remove Accès aux collections en utilisant le caractère ! (exemple : MaCollection!Foo)
Compilation conditionnelle	#Const #If...Then...#Else
Structures de contrôle	DoEvents GoSub...Return, GoTo On Error GoTo On...GoSub, On...GoTo Numéro de ligne, étiquette de ligne
Conversion	CVar, CVDate Str, Val
Types de données	Tous les types de données intrinsèques sauf Variant de type Type...End
Date/Heure	Instruction Date, instruction Time
DDE	LinkExecute, LinkPoke, LinkRequest, LinkSend

Débogage	Debug.Print End, Stop
Déclaration	Declare (pour déclarer des DLL) Optional ParamArray Static
Gestion des erreurs	Erl Error Resume, Resume Next
Entrée/Sortie de fichier	Toutes les E/S de fichier traditionnelles Basic
Finances	Toutes les fonctions financières
Manipulation d'objet	TypeOf
Objets	Clipboard Collection
Opérateurs	Like
Options	Deftype Option Base Option Compare Option Private Module
Select Case	Expressions contenant le mot clé Is ou tout opérateur de comparaison. Expressions qui contiennent une plage de valeurs utilisant le mot clé To .
Chaînes	Chaînes de longueur fixe LSet, RSet Instruction Mid StrConv
Utilisation d'objets	Accès aux collections avec !

Caractéristiques de VBScript non incluses dans Visual Basic pour Applications

[Référence du langage](#)

Catégorie	Caractéristique/Mot clé
Déclarations	Class
Divers	Eval Execute
Objets	RegExp
Identification du moteur de script	ScriptEngine ScriptEngineBuildVersion ScriptEngineMajorVersion ScriptEngineMinorVersion

Caractéristiques de la référence de la bibliothèque d'exécution Microsoft Scripting

[Référence du langage](#)

Catégorie	Caractéristique/Mot clé
Collections	Drives Files Folders
Stockage des données	Dictionary
Dictionnaire	Add Exists Items , Keys Remove , RemoveAll Count Item , Key
Système de fichiers	Drive File FileSystemObject Folder TextStream
	BuildPath CopyFile , CopyFolder CreateFolder , CreateTextFile

FileSystemObject	DeleteFile , DeleteFolder DriveExists , FileExists , FolderExists GetAbsolutePathName , GetBaseName GetDrive , GetDriveName GetFile , GetExtensionName GetFileName GetFolder , GetParentFolderName GetSpecialFolder GetTempName MoveFile , MoveFolder OpenTextFile Drives
Lecteur, lecteurs	AvailableSpace Count DriveLetter DriveType FileSystem FreeSpace IsReady Item RootFolder SerialNumber ShareName TotalSize VolumeName
Fichier, fichiers Dossier, dossiers	Add Attributes Copy , Delete , Move Count OpenAsTextStream DateCreated , DateLastAccessed , DateLastModified Drive Item ParentFolder Name , Path ShortName , ShortPath Size

TextStream	Close Read , ReadAll , ReadLine Skip , SkipLine Write , WriteBlankLines , WriteLine AtEndOfLine , AtEndOfStream Column , Line
------------	--

Abs, fonction

[Voir aussi](#)

Description

Renvoie la valeur absolue d'un nombre.

Syntaxe

Abs(*number*)

L'argument *number* peut être toute [expression numérique](#) valide. Si *number* contient **Null**, **Null** est renvoyé ; s'il s'agit d'une variable non initialisée, zéro est renvoyé.

Notes

La valeur absolue d'un nombre correspond à la valeur de ce dernier privée du signe. Par exemple, **Abs(- 1)** et **Abs(1)** renvoient tous deux 1.

L'exemple ci-dessous utilise la fonction **Abs** pour calculer la valeur absolue d'un nombre :

```
Dim MyNumber
```

```
MyNumber = Abs(50.3) ' Renvoie 50,3.
```

```
MyNumber = Abs(-50.3) ' Renvoie 50,3.
```

opérateur

[Voir aussi](#)

Description

Effectue la somme de deux nombres.

Syntaxe

result = *expression1*+*expression2*

La syntaxe de l'opérateur + comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>expression1</i>	Toute expression .
<i>expression2</i>	Toute expression.

Notes

S'il vous est possible d'utiliser l'opérateur + pour concaténer deux chaînes de caractères, il est néanmoins préférable d'utiliser l'opérateur & pour la concaténation afin d'éliminer toute ambiguïté et de fournir un code compréhensible en lui-même.

Si vous utilisez l'opérateur +, il vous sera parfois difficile de déterminer si une addition ou une concaténation de chaîne se produira.

Le sous-type sous-jacent des expressions détermine le comportement de l'opérateur + de la manière suivante :

Si	alors
Les deux expressions sont numériques	Add.
Les deux expressions sont des chaînes	Concatenate.

Une expression est numérique et l'autre est une chaîne	Add.
--	------

Si l'une des expressions ou les deux sont [Null](#), *result* est **Null**. Si les deux expressions sont [Empty](#), *result* est un sous-type **Integer**. Toutefois, si une seule des expressions est **Empty**, l'autre expression est renvoyée inchangée comme *result*.

And, opérateur

[Voir aussi](#)

Description

Effectue la conjonction logique de deux expressions.

Syntaxe

result = *expression1* **And** *expression2*

La syntaxe de l'opérateur **And** comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>expression1</i>	Toute expression .
<i>expression2</i>	Toute expression.

Notes

Si, et seulement si, les deux expressions produisent la valeur **True**, *result* est **True**. Si l'une ou l'autre produit la valeur **False**, *result* est **False**. Le tableau suivant illustre la manière dont *result* est déterminé :

Si <i>expression1</i> est	et <i>expression2</i> est	<i>result</i> est
True	True	True
True	False	False
True	Null	Null
False	True	False
False	False	False
False	Null	False

Null	True	Null
Null	False	False
Null	Null	Null

L'opérateur **And** effectue aussi une [comparaison binaire](#) des bits de position identique dans deux [expressions numériques](#) et définit le bit correspondant dans *result* d'après la table de vérité suivante :

Si le bit dans <i>expression1</i> est	et le bit dans <i>expression2</i> est	<i>result</i> est
0	0	0
0	1	0
1	0	0
1	1	1

Array, fonction

[Voir aussi](#)

Description

Renvoie une variable de type **Variant** contenant un [tableau](#).

Syntaxe

Array(*arglist*)

L'argument *arglist* correspond à une liste de valeurs séparées par des virgules affectée aux éléments d'un tableau contenu dans la variable **Variant**. Si aucun argument n'est spécifié, un tableau de longueur nulle est créé.

Notes

La notation utilisée pour faire référence à un élément d'un tableau est composée du nom de variable suivi de parenthèses contenant un numéro d'index précisant l'élément concerné. Dans l'exemple suivant, la première instruction crée une variable A. La deuxième instruction affecte un tableau à la variable A. La dernière instruction affecte la valeur contenue dans le deuxième élément du tableau à une autre variable.

Dim A

A = **Array**(10,20,30)

B = A(2) ' B a désormais la valeur 30.

Remarque Une variable qui n'est pas déclarée comme tableau peut quand même contenir un tableau. Bien qu'une variable de type **Variant** contenant un tableau soit différente d'une variable de tableau contenant des éléments **Variant**, l'accès aux éléments du tableau s'effectue de la même manière.

fonction

[Voir aussi](#)

Description

Renvoie le code de caractère ANSI correspondant à la première lettre d'une chaîne.

Syntaxe

Asc(*string*)

L'argument *string* correspond à toute [expression de chaîne](#) valide. Si *string* ne contient aucun caractère, une [erreur d'exécution](#) se produit.

Notes

Dans l'exemple suivant, **Asc** renvoie le code de caractère ANSI de la première lettre de chaque chaîne :

```
Dim MyNumber
```

```
MyNumber = Asc("A") ' Renvoie 65.
```

```
MyNumber = Asc("a") ' Renvoie 97.
```

```
MyNumber = Asc("Apple") ' Renvoie 65.
```

Remarque Une autre fonction (**AscB**) peut être utilisée avec les données de type octet contenues dans une chaîne. Au lieu de

renvoyer le code de caractère du premier caractère, **AscB** renvoie le premier octet. **AscW** est disponible sur les plateformes 32 bits qui utilisent des caractères Unicode. Elle renvoie le code de caractère Unicode (large), ce qui évite une conversion de Unicode à ANSI.



opérateur

[Voir aussi](#)

Description

Affecte une valeur à une [variable](#) ou à une [propriété](#).

Syntaxe

variable = *value*

La syntaxe de l'opérateur = comprend les éléments suivants :

Élément	Description
<i>variable</i>	Toute variable ou propriété qui peut être définie.
<i>value</i>	Toute variable numérique ou chaîne, constante , ou expression .

Notes

Le nom à gauche du signe égal peut représenter une variable scalaire simple ou un élément d'un [tableau](#). Les propriétés à gauche du signe égal sont celles pour lesquelles l'écriture est autorisée au [moment de l'exécution](#).

fonction

[Voir aussi](#)

Description

Renvoie l'arc tangente d'un nombre.

Syntaxe

Atn(*number*)

L'argument *number* peut être toute [expression numérique](#) valide.

Notes

La fonction **Atn** prend le rapport des deux côtés d'un triangle rectangle (*number*) et renvoie l'angle correspondant exprimé en radians. Le rapport est la longueur du côté opposé à l'angle divisée par la longueur du côté adjacent à l'angle. Le résultat est compris entre $-\pi/2$ et $\pi/2$ radians.

Pour convertir des degrés en radians, multipliez les degrés par $\pi/180$. Pour convertir des radians en degrés, multipliez les radians par $180/\pi$.

L'exemple ci-dessous utilise la fonction **Atn** pour calculer la valeur de pi :

Dim pi

pi = 4 * **Atn**(1) ' Calcule la valeur de pi.

Remarque **Atn** est la fonction trigonométrique inverse de **Tan**,

qui prend un angle comme son argument et renvoie le rapport des deux côtés d'un triangle rectangle. Ne confondez pas **Atn** avec la cotangente, qui est l'inverse simple d'une tangente (1/tangente).

Call, instruction

Description

Transfère le contrôle à une procédure **Sub** ou **Function**.

Syntaxe

[**Call**] *name* [*argumentlist*]

La syntaxe de l'instruction **Call** comprend les éléments suivants :

Élément	Description
Call	Mot clé facultatif ; si spécifié, vous devez mettre <i>argumentlist</i> entre parenthèses. Par exemple : Call MyProc(0)

name Nom de la procédure d'appel. *argumentlist* Liste, délimitée par des virgules, de variables, de [tableaux](#) ou d'[expressions](#) transmises à la procédure.

Notes

Vous n'êtes pas obligé d'utiliser le mot clé **Call** quand vous appelez une procédure. Toutefois, si vous utilisez le mot clé **Call** pour appeler une procédure exigeant des arguments, *argumentlist* doit être placé entre parenthèses. Si vous omettez le mot clé **Call**, vous devez aussi omettre les parenthèses délimitant *argumentlist*. Si vous utilisez la syntaxe **Call** pour appeler toute fonction intrinsèque ou définie par l'utilisateur, la valeur

renvoyée de la fonction est ignorée.

Call

```
MyFunction("Bonjour")
```

```
Function MyFunction(text)
```

```
    MsgBox text
```

```
End Function
```

CBool, fonction

[Voir aussi](#)

Description

Renvoie une expression qui a été convertie en un **Variant** de sous-type **Boolean**.

Syntaxe

CBool(*expression*)

L'argument *expression* est toute expression valide.

Notes

Si *expression* correspond à zéro, la valeur **False** est renvoyée ; si tel n'est pas le cas, la valeur **True** est renvoyée. Si l'argument *expression* ne peut être interprété comme valeur numérique, une erreur [d'exécution](#) se produit.

L'exemple suivant utilise la fonction **CBool** pour convertir une expression en **Boolean**. Si l'expression représente une valeur non nulle, **CBool** renvoie **True** ; sinon elle renvoie **False**.

```
Dim A, B, Check
```

```
A = 5: B = 5      ' Initialise les variables.
```

```
Check = CBool(A = B) ' Check contient True.
```

```
A = 0           ' Définit la variable.
```

```
Check = CBool(A) ' Check contient False.
```

Cbyte, fonction

[Voir aussi](#)

Description

Renvoie une expression qui a été convertie en un **Variant** de sous-type **Byte**.

Syntaxe

CByte(*expression*)

L'argument *expression* représente toute expression valide.

Notes

En général, vous pouvez documenter votre code à l'aide des fonctions de conversion des sous-types pour indiquer que le résultat d'une certaine opération doit être exprimé sous forme d'un type de données particulier plutôt que sous la forme du type de données par défaut. Par exemple, utilisez **CByte** pour forcer l'arithmétique octet dans les cas où l'arithmétique monétaire, en simple précision, en double précision ou en nombre entier serait normalement utilisée.

Utilisez la fonction **CByte** pour effectuer des conversions reconnues au niveau international de tout autre type de données en sous-type **Byte**. Par exemple, différents séparateurs décimaux sont correctement reconnus selon [les paramètres régionaux](#) de votre système, comme le sont les différents séparateurs de milliers.

Si l'argument *expression* n'est pas compris dans la plage acceptable pour le sous-type **Byte**, une erreur se produit. L'exemple suivant utilise la fonction **CByte** pour convertir une expression en octet :

Dim MyDouble, MyByte

MyDouble = 125,5678

' MyDouble est un Do

MyByte = CByte(MyDouble) ' MyByte contient

CCur, fonction

[Voir aussi](#)

Description

Renvoie une expression qui a été convertie en un **Variant** de sous-type **Currency**.

Syntaxe

CCur(*expression*)

L'argument *expression* correspond à n'importe quelle expression valide.

Notes

En général, vous pouvez documenter votre code en utilisant les fonctions de conversion de sous-type pour indiquer que le résultat de certaines opérations doit être exprimé sous la forme d'un type de données particulier et non sous la forme du type de données par défaut. Par exemple, utilisez la fonction **CCur** pour forcer l'emploi d'une arithmétique monétaire lorsqu'une arithmétique entière devrait normalement être utilisée.

Il convient d'employer la fonction **CCur** pour convertir d'autres types de données en un sous-type **Currency** tout en respectant les conventions internationales. Par exemple différents séparateurs décimaux et séparateurs de milliers sont correctement reconnus en fonction des paramètres [régionaux](#) de votre système.

L'exemple suivant utilise la fonction **CCur** pour convertir une expression en Currency :

```
Dim MyDouble, MyCurr
MyDouble = 543,214588      ' MyDouble est un Double.
MyCurr = CCur(MyDouble * 2) ' Convertit le résultat de
MyDouble * 2 (1086,429176)
                          ' en Currency (1086,4292).
```



CDate, fonction

[Voir aussi](#)

Description

Renvoie une expression qui a été convertie en un **VARIANT** de sous-type **Date**.

Syntaxe

CDate(*date*)

L'argument *date* représente toute [expression date](#) valide.

Notes

Utilisez la fonction **IsDate** pour déterminer si l'argument *date* peut être converti en date ou en heure. **CDate** reconnaît les [littéraux de date](#) et heure ainsi que certains nombres compris dans la plage des dates acceptables. Lors de la conversion d'un nombre en date, la partie entière du nombre est convertie en date. Toute partie fractionnaire du nombre est convertie en heure du jour, commençant à minuit.

CDate reconnaît les formats de date en fonction des [paramètres régionaux](#) de votre système. L'ordre correct du jour, du mois et de l'année ne peut être déterminé s'il est fourni dans un format différent de celui reconnu par votre paramétrage de date. Par ailleurs, un format de date de type long n'est pas reconnu s'il contient aussi la chaîne jour de la semaine.

L'exemple ci-dessous utilise la fonction **CDate** pour convertir une chaîne en date. En général, l'utilisation de chaînes pour stocker des dates et des heures (comme dans cet exemple) n'est pas recommandé. Utilisez des littéraux date et heure (comme #19/10/1962#, #16:45:23#).

```
MyDate = "19 octobre 1962" ' Définit la date.
```

```
MyShortDate = CDate(MyDate) ' Convertit en type de données I
```

```
MyTime = "16:35:47" ' Définit l'heure.
```

```
MyShortTime = CDate(MyTime) ' Convertit en type de données I
```

CDBl, fonction

[Voir aussi](#)

Description

Renvoie une expression qui a été convertie en un **VARIANT** de sous-type **Double**.

Syntaxe

CDBl(*expression*)

L'argument *expression* représente toute expression valide.

Notes

En général, vous pouvez documenter votre code en utilisant les fonctions de conversion des sous-types pour indiquer que le résultat d'une opération doit être exprimé sous forme d'un type de données particulier plutôt que sous la forme du type de données par défaut. Par exemple, utilisez la fonction **CDBl** ou **CSng** pour forcer l'arithmétique en double ou en simple précision dans les cas où l'arithmétique monétaire ou entier serait normalement utilisée.

Utilisez la fonction **CDBl** pour fournir des conversions reconnues au niveau international de tout autre type de données en sous-type **Double**. Par exemple, différents séparateurs décimaux et séparateurs de milliers sont correctement reconnus en fonction des [paramètres régionaux](#) de votre système.

Cet exemple utilise la fonction **CDbl** pour convertir une expression en type **Double**.

```
Dim MyCurr, MyDouble  
MyCurr = CCur(234.456784)           ' MyCurr es  
MyDouble = CDbl(MyCurr * 8.2 * 0.01) ' Conve  
en Double (19,2254576).
```

Chr, fonction

[Voir aussi](#)

Description

Renvoie le caractère associé au code de caractère ANSI spécifié.

Syntaxe

Chr(*charcode*)

L'argument *charcode* représente un nombre qui identifie un [caractère](#).

Note

Les nombres de 0 à 31 sont identiques aux codes [ASCII](#) standard, non imprimables. Par exemple, **Chr**(10) renvoie un caractère de retour à la ligne.

L'exemple ci-dessous utilise la fonction **Chr** pour renvoyer le caractère associé au code spécifié :

```
Dim MyChar
```

```
MyChar = Chr(65) ' Renvoie A.
```

```
MyChar = Chr(97) ' Renvoie a.
```

```
MyChar = Chr(62) ' Renvoie >.
```

```
MyChar = Chr(37) ' Renvoie %.
```

Remarque Une autre fonction (**ChrB**) peut être utilisée avec les données d'octet contenues dans une chaîne. Au lieu de renvoyer un caractère, qui peut être un ou deux octets, la fonction **ChrB** renvoie toujours un seul octet. **ChrW** est disponible pour les plates-formes 32 bits utilisant des caractères Unicode. Son argument est un code de caractère Unicode (large), ce qui évite une conversion de ANSI à Unicode.

CInt, fonction

[Voir aussi](#)

Description

Renvoie expression qui a été convertie en un **Variant** de sous-type **Integer**.

Syntaxe

CInt(*expression*)

L'argument *expression* représente toute expression valide.

Notes

En général, vous pouvez documenter votre code en utilisant les fonctions de conversion des sous-types pour indiquer que le résultat d'une opération doit être exprimé sous forme d'un type de données particulier plutôt que sous la forme du type de données par défaut. Par exemple, utilisez la fonction **CInt** ou **CLng** pour forcer l'arithmétique entier dans les cas où l'arithmétique monétaire, en simple précision ou en double précision serait normalement utilisée.

Utilisez la fonction **CInt** pour fournir des conversions reconnues au niveau international de tout autre type de données en sous-type **Integer**. Par exemple, différents séparateurs décimaux et séparateurs des milliers sont reconnus en fonction des [paramètres régionaux](#) de votre système.

Si l'argument *expression* n'est pas compris dans la plage acceptable pour le sous-type [Integer](#), une erreur se produit.

L'exemple ci-dessous utilise la fonction **CInt** pour convertir une valeur en entier (Integer) :

```
Dim MyDouble, MyInt  
MyDouble = 2345,5678    ' MyDouble est un Double  
MyInt = CInt(MyDouble) ' MyInt contient 2346.
```

Remarque La fonction **CInt** est différente des fonctions **Fix** et **Int** qui tronquent au lieu d'arrondir la mantisse d'un nombre. Quand la mantisse est exactement 0,5, la fonction **CInt** l'arrondit toujours au nombre pair le plus proche. Par exemple, 0,5 est arrondi à 0 et 1,5 est arrondi à 2.

Class, objet

[Voir aussi](#)[Événements](#)

Description

Objet créé à l'aide de l'instruction **Class**. Procure un accès aux événements de la [classe](#).

Notes

Vous ne pouvez pas déclarer de façon explicite une [variable](#) de type Class. Dans le contexte VBScript, le terme "class object" fait référence à tout objet défini à l'aide de l'instruction de la **Class** VBScript.

Lorsque la définition d'une classe est créée à l'aide de l'instruction **Class**, vous êtes en mesure de créer une instance de la classe de la forme suivante :

```
Dim X  
Set X = New classname
```

VBScript étant un langage récemment dépendant, vous ne pouvez pas écrire les expressions suivantes :

```
Dim X as New classname
```

- ou -

```
Dim X  
X = New classname
```

- ou -

```
Set X = New Scripting.FileSystemObject
```

Class, instruction

[Voir aussi](#)

Description

Déclare le nom d'une [classe](#), ainsi que la définition des variables, des propriétés et des méthodes qui s'appliquent à la classe.

Syntaxe

Class *name*
statements

End Class

La syntaxe de l'instruction **Class** comprend les éléments suivants :

Élément	Description
<i>name</i>	Requis. Nom de la classe ; respecte les conventions standard d'affectation de noms de variable .
<i>statements</i>	Requis. Une ou plusieurs instructions définissant les variables, les propriétés et les méthodes de la classe .

Notes

Dans un bloc **Class**, les membres sont déclarés **Private** ou **Public** par l'intermédiaire des instructions de déclarations appropriées. Tout élément déclaré comme [Private](#) est visible seulement dans le bloc **Class**. En revanche, tout élément déclaré comme [Public](#) est visible dans le bloc **Class** et également par le

code à l'extérieur du bloc **Class**. Tout élément qui n'est pas déclaré de façon explicite, c'est-à-dire dont le statut est **Private** ou **Public**, prend le statut **Public** par défaut. Les [procédures](#) (**Sub** ou **Function**) déclarées **Public** dans le bloc de la classe deviennent des méthodes de la classe. Les variables **Public** sont des propriétés de la classe, tout comme les propriétés explicitement déclarées comme utilisant les propriétés **Property Get**, **Property Let** et **Property Set**. Les propriétés et les méthodes par défaut de la classe sont indiquées par le mot clé **Default**. Reportez-vous aux rubriques de chaque instruction de déclaration pour plus d'informations sur le fonctionnement du [mot clé](#).

Clear, méthode

[Voir aussi](#)

[Application](#)

Description

Réinitialise les propriétés de l'objet **Err**.

Syntaxe

object.**Clear**

L'objet représente toujours un objet **Err**.

Notes

Utilisez la méthode **Clear** pour réinitialiser explicitement l'objet **Err** après le traitement d'une erreur. Ceci est nécessaire, par exemple, lorsque vous utilisez le traitement différé d'une erreur avec **On Error Resume Next**. VBScript appelle automatiquement la méthode **Clear** chaque fois que l'une des instructions suivantes est exécutée :

- **On Error Resume Next**
- **Exit Sub**
- **Exit Function**

L'exemple ci-dessous illustre l'utilisation de la méthode **Clear** :

```
On Error Resume Next
```

```
Err.Raise 6 ' Génère une erreur de dépassement.
```

```
MsgBox ("Error # " & CStr(Err.Number) & " " & Err.Description)
```

```
Err.Clear ' Efface l'erreur.
```

CLng, fonction

[Voir aussi](#)

Description

Renvoie une expression qui a été convertie en un **VARIANT** de sous-type **Long**.

Syntaxe

CLng(*expression*)

L'argument *expression* représente toute expression valide.

Notes

En général, vous pouvez documenter votre code en utilisant les fonctions de conversion des sous-types pour indiquer que le résultat d'une opération doit être exprimé sous forme d'un type de données particulier plutôt que sous la forme du type de données par défaut. Par exemple, utilisez la fonction **CInt** ou **CLng** pour forcer l'arithmétique entier dans les cas où l'arithmétique monétaire, en simple précision ou en double précision serait normalement utilisée.

Utilisez la fonction **CLng** pour fournir des conversions reconnues au niveau international de tout autre type de données en sous-type **Long**. Par exemple, différents séparateurs décimaux et séparateurs des milliers sont correctement reconnus en fonction des [paramètres régionaux](#) de votre système.

Si l'argument *expression* n'est pas compris dans la plage acceptable pour le sous-type **Long**, une erreur se produit.

L'exemple ci-dessous utilise la fonction **CLng** pour convertir une valeur en entier de type Long :

```
Dim MyVal1, MyVal2, MyLong1, MyLong2
MyVal1 = 25427.45: MyVal2 = 25427.55 ' MyVal1, MyVal2
MyLong1 = CLng(MyVal1) ' MyLong1
MyLong2 = CLng(MyVal2) ' MyLong2
```

Remarque La fonction **CLng** est différente des fonctions **Fix** et **Int** qui tronquent au lieu d'arrondir la mantisse d'un nombre. Quand la mantisse est exactement 0,5, la fonction **CLng** l'arrondit toujours au nombre pair le plus proche. Par exemple, 0,5 est arrondi à 0 et 1,5 est arrondi à 2.

Comparaison, constantes

[Voir aussi](#)

Ces constantes étant intégrées dans VBScript, il n'est pas nécessaire de les définir pour les utiliser. Vous pouvez les insérer n'importe où dans le code pour représenter les valeurs qui leur sont associées.

Constante	Valeur	Description
vbBinaryCompare	0	Effectue une comparaison binaire.
vbTextCompare	1	Effectue une comparaison de texte.

opérateur

[Voir aussi](#)

Description

Force la concaténation de chaînes de deux expressions.

Syntaxe

result = *expression1* & *expression2*

La syntaxe de l'opérateur & comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable.
<i>expression1</i>	Toute expression .
<i>expression2</i>	Toute expression.

Notes

Chaque fois qu'une *expression* n'est pas une chaîne, elle est convertie en un sous-type **String**. Si les deux expressions sont **Null**, *result* est aussi **Null**. Toutefois, si une seule *expression* est **Null**, cette expression est traitée comme une chaîne de longueur nulle lorsqu'elle est concaténée avec l'autre expression. Toute expression **Empty** est également traitée comme une chaîne de longueur nulle.

Const, instruction

[Voir aussi](#)

Description

Déclare des [constantes](#) destinées à remplacer des valeurs littérales.

Syntaxe

[Public | Private] Const *constname* = *expression*

La syntaxe de l'instruction **Const** comprend les éléments suivants:

Élément	Description
Public	Facultatif. Mot clé utilisé au niveau du script pour déclarer des constantes accessibles dans toutes les procédures de tous les scripts. Interdit dans les procédures.
Private	Facultatif. Mot clé utilisé au niveau script pour déclarer des constantes accessibles uniquement dans le script où la déclaration est effectuée. Interdit dans les procédures.
<i>constname</i>	Nom de la constante ; respecte les conventions standard d'attribution de nom de variable .
<i>expression</i>	Littéral ou autre constante, ou toute combinaison incluant tous les opérateurs arithmétiques ou logiques, à l'exception de Is .

Notes

Les constantes sont publiques par défaut. à l'intérieur des procédures, elles sont toujours privées et leur visibilité ne peut pas être modifiée. Dans un

script, la visibilité par défaut d'une constante de niveau script peut être modifiée à l'aide du mot clé **Private**.

Pour combiner plusieurs déclarations de constante sur la même ligne, séparez chaque affectation de constante par une virgule. Lorsque des déclarations de constante sont combinées de cette manière, l'emploi éventuel d'un mot clé **Public** ou **Private** s'applique à toutes ces déclarations.

Vous ne pouvez pas utiliser des variables, des fonctions définies par l'utilisateur ou des fonctions VBScript intrinsèques (telles que **Chr**) dans des déclarations de constante. Par définition, elles ne peuvent pas être des constantes. Vous ne pouvez pas non plus créer de constantes à partir d'une expression impliquant un opérateur, c'est-à-dire que seules les constantes simples sont autorisées. Les constantes déclarées dans une procédure **Sub** ou **Function** sont locales à cette procédure. Une constante déclarée à l'extérieur d'une procédure est définie pour l'ensemble du script dans lequel elle est déclarée. Vous pouvez utiliser des constantes à tout endroit où vous pouvez employer une expression. Le code suivant illustre l'utilisation de l'instruction **Const** :

```
Const MyVar = 459 ' Les c  
Private Const MyString = "AIDE" ' Déclare les cc  
Const MyStr = "Bonjour", MyNumber = 3.4567 '
```

Remarque Les constantes peuvent documenter automatiquement vos scripts et en simplifier la modification. Contrairement aux variables, les constantes ne peuvent pas être modifiées accidentellement pendant l'exécution de votre script.

Cos, fonction

[Voir aussi](#)

Description

Renvoie le cosinus d'un angle.

Syntaxe

Cos(*number*)

L'argument *number* peut être toute [expression numérique](#) valide exprimant un angle en radians.

Notes

La fonction **Cos** prend un angle et renvoie le rapport des deux côtés d'un triangle rectangle. Le rapport correspond à la longueur du côté adjacent à l'angle divisée par la longueur de l'hypoténuse. Le résultat est compris entre -1 et 1.

Pour convertir des degrés en radians, multipliez les degrés par [pi](#)/180. Pour convertir des radians en degrés, multipliez les radians par 180/pi.

L'exemple ci-dessous utilise la fonction **Cos** pour renvoyer le cosinus d'un angle :

```
Dim MyAngle, MySecant
```

```
MyAngle = 1.3           ' Définir l'angle en radian.
```

```
MySecant = 1 / Cos(MyAngle) ' Calculer la sécant
```

CreateObject, fonction

[Voir aussi](#)

Description

Crée et renvoie une référence à un [objet Automation](#).

Syntaxe

CreateObject(*servername.typename* [, *location*])

La syntaxe de la fonction **CreateObject** comprend les éléments suivants :

Élément	Description
<i>servername</i>	Requis. Le nom de l'application fournissant l'objet.
<i>typename</i>	Requis. Le type ou la classe de l'objet à créer.
<i>location</i>	Facultatif. Le nom du serveur réseau sur lequel l'objet doit être créé. Cette fonction est disponible à partir de la version 5.1.

Notes

Les serveurs Automation fournissent au moins un type d'objet. Par exemple, une application de traitement de texte peut fournir un objet d'application, un objet de document et un objet de barre d'outils.

Pour créer un objet Automation, affectez l'objet renvoyé par **CreateObject** à une variable objet :

```
Dim ExcelSheet  
Set ExcelSheet = CreateObject("Excel.Sheet")
```

Ce code démarre l'application qui crée l'objet (dans ce cas, une feuille de

calcul Microsoft Excel). Une fois qu'un objet a été créé, faites référence à ce dernier dans le code en utilisant la variable objet définie. Comme l'indique l'exemple suivant, vous pouvez accéder aux propriétés et aux méthodes du nouvel objet à l'aide de la variable objet, **ExcelSheet**, et d'autres objets Excel, notamment l'objet Application et la collection `ActiveSheet.Cells`.

```
' Rend Excel visible par l'intermédiaire de l'objet A
ExcelSheet.Application.Visible = True
' Place du texte dans la première cellule de la feuille
ExcelSheet.ActiveSheet.Cells(1,1).Value = "Colon
' Enregistre la feuille.
ExcelSheet.SaveAs "C:.XLS"
' Ferme Excel avec la méthode Quit sur l'objet App
ExcelSheet.Application.Quit
' Libère la variable objet.
Set ExcelSheet = Nothing
```

Vous pouvez uniquement créer un objet sur un serveur distant lorsque la sécurité Internet est désactivée. Pour ce faire, il convient de transmettre le nom de la machine à l'argument *servername* de la fonction **CreateObject**. Dans un nom de partage, ce nom correspond à la partie réservée au nom de la machine. Par exemple, dans le nom de partage réseau "`\\myserver\public`", le nom du serveur (*servername*) correspond à "`myserver`". En outre, vous pouvez spécifier l'argument *servername* au format DNS ou sous la forme d'une adresse IP.

Le code suivant renvoie le numéro de version d'une instance d'Excel qui s'exécute sur un ordinateur réseau distant appelé "myserver" :

```
Function GetVersion
Dim XLApp
```

```
Set XLApp = CreateObject("Excel.Application",  
GetVersion = XLApp.Version  
End Function
```

Une erreur se produit si le serveur distant spécifié est inexistant ou introuvable.

[Voir aussi](#)

Description

Renvoie une expression qui a été convertie en un **VARIANT** de sous-type **Single**.

Syntaxe

CSng(*expression*)

L'argument *expression* représente toute expression valide.

Notes

En général, vous pouvez documenter votre code en utilisant les fonctions de conversion des types de données pour indiquer que le résultat d'une opération doit être exprimé sous forme d'un type de données particulier plutôt que sous la forme du type de données par défaut. Par exemple, utilisez la fonction **Cdbl** ou **CSng** pour forcer l'arithmétique en double ou en simple précision dans les cas où l'arithmétique monétaire ou entier serait normalement utilisée.

Utilisez la fonction **CSng** pour fournir des conversions reconnues au niveau international de tout autre type de données en sous-type **Single**. Par exemple, différents séparateurs décimaux sont correctement reconnus en fonction des [paramètres régionaux](#) de votre système, comme les différents séparateurs de milliers.

Si l'argument *expression* n'est pas compris dans la plage acceptable pour le sous-type **Single**, une erreur se produit.

L'exemple ci-dessous utilise la fonction **CSng** pour convertir une valeur en **Single** :

Dim MyDouble1, MyDouble2, MySingle1, MySingle2
sont des Doubles.

MyDouble1 = 75,3421115: MyDouble2 = 75,3421115

MySingle1 = **CSng(MyDouble1)** '1

MySingle2 = **CSng(MyDouble2)** '1

fonction

[Voir aussi](#)

Description

Renvoie une expression qui a été convertie en un **VARIANT** de sous-type **String**.

Syntaxe

CStr(*expression*)

L'argument *expression* représente toute expression valide.

Notes

En général, vous pouvez documenter votre code en utilisant des fonctions de conversion des types de données pour indiquer que le résultat d'une opération doit être exprimé sous forme d'un type de données particulier plutôt que sous la forme du type de données par défaut. Par exemple, utilisez la fonction **CStr** pour forcer le résultat à être exprimé sous forme d'un sous-type **String**.

Vous devez utiliser la fonction **CStr** à la place de **Str** pour fournir des conversions reconnues au niveau international de tout autre type de données en sous-type **String**. Par exemple, différents séparateurs décimaux sont correctement reconnus en fonction des [paramètres régionaux](#) de votre système.

Les données contenues dans l'argument *expression* déterminent ce qui est

renvoyé conformément au tableau suivant:

Si <i>expression</i> est	La fonction CStr renvoie
Boolean	Un String contenant True ou False .
Date	Un String contenant une date dans le format court de votre système.
Null	Une erreur d'exécution.
Empty	Un String de longueur nulle ("").
Error	Un String contenant le mot Erreur suivi d'un numéro d'erreur.
Autre élément numérique	Un String contenant le nombre.

L'exemple ci-dessous utilise la fonction **CStr** pour convertir une valeur numérique en **String** (chaîne) :

```
Dim MyDouble, MyString
```

```
MyDouble = 437,324      ' MyDouble est un Double
```

```
MyString = CStr(MyDouble) ' MyString contient '437,324'
```

Date/Heure, constantes

[Voir aussi](#)

Ces constantes étant intégrées dans VBScript, il n'est pas nécessaire de les définir pour les utiliser. Vous pouvez les insérer n'importe où dans le code pour représenter les valeurs qui leur sont associées.

Constante	Valeur	Description
vbSunday	1	Dimanche
vbMonday	2	Lundi
vbTuesday	3	Mardi
vbWednesday	4	Mercredi
vbThursday	5	Jeudi
vbFriday	6	Vendredi
vbSaturday	7	Samedi
vbUseSystem	0	Utilise le format de date contenu dans les paramètres régionaux de votre ordinateur.
vbUseSystemDayOfWeek	0	Utilise le jour de la semaine spécifié dans les paramètres régionaux de votre système pour le premier jour de la semaine.
vbFirstJan1	1	Utilise la semaine dans laquelle tombe le 1er janvier (par défaut).
vbFirstFourDays	2	Utilise la première semaine comportant au moins quatre

		jours dans la nouvelle année.
vbFirstFullWeek	3	Utilise la première semaine complète de l'année.

Format de date, constantes

[Voir aussi](#)

Ces constantes étant intégrées dans VBScript, il n'est pas nécessaire de les définir pour les utiliser. Vous pouvez les insérer n'importe où dans le code pour représenter les valeurs qui leur sont associées.

Constante	Valeur	Description
vbGeneralDate	0	Affiche une date et/ou une heure. Pour les nombres réels, affiche une date et une heure. En l'absence de parties décimales, affiche seulement une date. S'il n'y a pas de parties entières, affiche seulement l'heure. Le mode d'affichage de la date et de l'heure est fonction des paramètres de votre système.
vbLongDate	1	Affiche une date en utilisant le format de date complet spécifié dans les paramètres régionaux de votre ordinateur.
vbShortDate	2	Affiche une date en utilisant le format de date abrégé spécifié dans les paramètres régionaux de l'ordinateur.
vbLongTime	3	Affiche une heure en utilisant le format d'heure complet spécifié dans les paramètres régionaux de votre ordinateur.
vbShortTime	4	Affiche une heure en utilisant le format d'heure abrégé spécifié dans les paramètres régionaux de votre ordinateur.



Date, fonction

[Voir aussi](#)

Description

Renvoie la date système courante.

Syntaxe

Date

Remarques

L'exemple ci-dessous utilise la fonction **Date** pour renvoyer la date système actuelle :

```
Dim MyDate  
MyDate = Date ' MyDate contient la date système actuelle.
```

DateAdd, fonction

[Voir aussi](#)

Description

Renvoie une date à laquelle un intervalle spécifique a été ajouté.

Syntaxe

DateAdd(*interval*, *number*, *date*)

La syntaxe de la fonction **DateAdd** comprend les éléments suivants :

Élément	Description
<i>interval</i>	Requis. Expression de chaîne correspondant à l'intervalle à ajouter. Les valeurs sont indiquées dans la section Valeurs.
<i>number</i>	Requis. Expression numérique précisant le nombre d'intervalles à ajouter. L'expression numérique peut être positive, pour les dates dans le futur, ou négative pour les dates dans le passé.
<i>date</i>	Requis. Variant ou littéral représentant la date à laquelle l'argument <i>interval</i> est ajouté.

Valeurs

L'argument *interval* peut prendre les valeurs suivantes :

Valeur	Description
<i>yyyy</i>	Année
<i>q</i>	Trimestre
<i>m</i>	Mois
<i>y</i>	Jour de l'année

<i>d</i>	Jour
<i>w</i>	Jour de la semaine
<i>ww</i>	Semaine
<i>h</i>	Heure
<i>n</i>	Minute
<i>s</i>	Seconde

Notes

La fonction **DateAdd** permet d'ajouter ou de soustraire une date un intervalle spécifique, ou de le soustraire. Par exemple, **DateAdd** permet de calculer une date 30 jours d'aujourd'hui ou une heure 45 minutes de maintenant. Pour ajouter des jours l'argument *date*, vous pouvez utiliser Jour de l'année ("y"), Jour ("d") ou Jour de la semaine ("w").

La fonction **DateAdd** ne renvoie pas une date incorrecte. L'exemple suivant ajoute un mois au 31 janvier :

NouvDate = **DateAdd**("m", 1, "31-Jan-95")

Dans ce cas, la fonction **DateAdd** renvoie 28-Fév-95, et non 31-Fév-95. Si l'argument *date* a la valeur 31-Jan-96, la fonction renvoie 29-Fév-96, puisque 1996 est une année bissextile.

Si la date calculée précède de l'année 100, une erreur se produit.

Si un nombre ne correspond pas à une valeur de type **Long**, il est arrondi au nombre entier le plus proche avant d'être valué.

DateDiff, fonction

[Voir aussi](#)

Description

Renvoie le nombre d'intervalles entre deux dates.

Syntaxe

DateDiff(*interval*, *date1*, *date2* [,*firstdayofweek*[, *firstweekofyear*]])

La syntaxe de la fonction **DateDiff** comprend les éléments suivants :

élément	Description
<i>interval</i>	Expression de chaîne correspondant à l'intervalle à utiliser pour calculer la différence entre <i>date1</i> et <i>date2</i> . Reportez-vous à la section Valeurs.
<i>date1</i> , <i>date2</i>	Expressions de date. Deux dates à utiliser dans le calcul.
<i>firstdayofweek</i>	Facultatif. Constante qui spécifie le jour de la semaine. Si elle n'est pas spécifiée, dimanche est pris par défaut. Reportez-vous à la section Valeurs.
<i>firstweekofyear</i>	Facultatif. Constante spécifiant la première semaine de l'année. Si elle n'est pas spécifiée, la première semaine sera celle incluant le 1er janvier. Reportez-vous à la section Valeurs.

Valeurs

L'argument *interval* peut prendre les valeurs suivantes :

Valeur	Description
--------	-------------

yyyy	Année
q	Trimestre
m	Mois
y	Jour de l'année
d	Jour
w	Jour de la semaine
ww	Semaine
h	Heure
n	Minute
s	Seconde

L'argument *firstdayofweek* peut prendre les valeurs suivantes :

Constante	Valeur	Description
vbUseSystem	0	Utilise la valeur API NLS.
vbSunday	1	Dimanche (valeur par défaut)
vbMonday	2	Lundi
vbTuesday	3	Mardi
vbWednesday	4	Mercredi
vbThursday	5	Jeudi
vbFriday	6	Vendredi
vbSaturday	7	Samedi

L'argument *firstweekofyear* peut prendre les valeurs suivantes :

Constante	Valeur	Description
vbUseSystem	0	Utilise la valeur API NLS.
vbFirstJan1	1	Commence par la semaine incluant le 1er janvier (valeur par défaut).
vbFirstFourDays	2	Commence par la semaine comportant au moins quatre jours dans la nouvelle année.
vbFirstFullWeek	3	Commence par la première semaine

|| || complète de la nouvelle année. ||

Notes

La fonction **DateDiff** permet de déterminer combien d'intervalles spécifiés sont compris entre deux dates. Par exemple, elle peut calculer le nombre de jours entre deux dates, ou le nombre de semaines entre aujourd'hui et la fin de l'année.

Pour calculer le nombre de jours entre *date1* et *date2*, vous pouvez utiliser Jour de l'année ("y") ou Jour ("d"). Lorsque l'argument *interval* a la valeur Jour de la semaine ("w"), la fonction **DateDiff** renvoie le nombre de semaines entre les deux dates. Si *date1* tombe un lundi, la fonction **DateDiff** compte le nombre de lundis jusqu'à *date2*. Il compte *date2* mais pas *date1*. Cependant, si l'argument *interval* a la valeur Semaine ("ww"), la fonction **DateDiff** renvoie le nombre de semaines de calendrier entre les deux dates. Elle compte le nombre de dimanches entre *date1* et *date2*. La fonction **DateDiff** compte *date2* si elle tombe un dimanche, mais ne compte pas *date1*, même si elle tombe un dimanche.

Si *date1* se réfère à un moment ultérieur à *date2*, la fonction **DateDiff** renvoie un nombre négatif.

L'argument *firstdayofweek* a une incidence sur le calcul utilisant les symboles d'intervalle "w" et "ww".

Si *date1* ou *date2* correspond à un littéral de date, l'année spécifiée devient partie permanente de cette date. Cependant, si *date1* ou *date2* est placé entre guillemets (" "), et si vous omettez l'année, l'année courante est insérée dans votre code lors de chaque évaluation de l'expression *date1* ou *date2*. Il est ainsi possible d'écrire du code utilisable pendant plusieurs années.

Lors du calcul d'intervalle entre le 31 décembre de l'année précédente et le 1er janvier, la fonction **DateDiff** renvoie 1 pour Année ("yyyy"), même si un seul jour s'est écoulé.

L'exemple ci-dessous utilise la fonction **DateDiff** pour afficher le nombre de jours entre une date donnée et aujourd'hui :

```
Function DiffADate(theDate)
```

```
    DiffADate = "Jours à partir d'aujourd'hui: " & DateDiff("d",  
    Now, theDate)
```

```
End Function
```

DatePart, fonction

[Voir aussi](#)

Description

Renvoie la partie spécifiée d'une date donnée.

Syntaxe

DatePart(*interval*, *date*[, *firstdayofweek*[, *firstweekofyear*]])

La syntaxe de la fonction **DatePart** comprend les éléments suivants:

élément	Description
<i>interval</i>	Expression de chaîne représentant l'intervalle à renvoyer. Reportez-vous à la section Valeurs.
<i>date</i>	Expression de date à évaluer.
<i>firstdayofweek</i>	Facultatif. Constante qui spécifie le jour de la semaine. Si elle n'est pas spécifiée, dimanche est pris par défaut. Reportez-vous à la section Valeurs.
<i>firstweekofyear</i>	Facultatif. Constante spécifiant la première semaine de l'année. Si elle n'est pas spécifiée, la première semaine sera celle incluant le 1er janvier. Reportez-vous à la section Valeurs.

Valeurs

L'argument *interval* peut prendre les valeurs suivantes:

Valeur	Description
yyyy	Année
q	Trimestre

<i>m</i>	Mois
<i>y</i>	Jour de l'année
<i>d</i>	Jour
<i>w</i>	Jour de la semaine
<i>ww</i>	Semaine
<i>h</i>	Heure
<i>n</i>	Minute
<i>s</i>	Seconde

L'argument *firstdayofweek* peut prendre les valeurs suivantes:

Constante	Valeur	Description
vbUseSystem	0	Utilise la valeur API NLS.
vbSunday	1	Dimanche (valeur par défaut)
vbMonday	2	Lundi
vbTuesday	3	Mardi
vbWednesday	4	Mercredi
vbThursday	5	Jeudi
vbFriday	6	Vendredi
vbSaturday	7	Samedi

L'argument *firstweekofyear* peut prendre les valeurs suivantes:

Constante	Valeur	Description
vbUseSystem	0	Utilise la valeur API NLS.
vbFirstJan1	1	Commence par la semaine incluant le 1er janvier (valeur par défaut).
vbFirstFourDays	2	Commence par la semaine comportant au moins quatre jours dans la nouvelle année.
vbFirstFullWeek	3	Commence par la première semaine complète de la nouvelle année.

Notes

La fonction **DatePart** permet d'évaluer une date et de renvoyer un intervalle spécifique. Vous pouvez notamment l'utiliser pour calculer le jour de la semaine ou l'heure courante.

L'argument *firstdayofweek* a une incidence sur le calcul utilisant les symboles d'intervalle "w" et "ww".

Si *date* est un littéral de date, l'année spécifiée devient une partie permanente de cette date. Cependant, si *date* est entre guillemets (" "), et si vous omettez l'année, l'année courante est introduite dans votre code à chaque évaluation de l'expression *date*. Il est ainsi possible d'écrire du code utilisable pendant plusieurs années.

Cet exemple prend une date et, à l'aide de la fonction **DatePart**, affiche le trimestre concerné.

```
Function GetQuarter(TheDate)
    GetQuarter = DatePart("q", TheDate)
End Function
```

DateSerial, fonction

[Référence du langage](#)
[Version 1](#)

[Voir aussi](#)

Description

Renvoie un **VARIANT** de sous-type **DATE** pour une année, un mois et un jour spécifiés.

Syntaxe

DateSerial(*year*, *month*, *day*)

La syntaxe de la fonction **DateSerial** comprend les éléments suivants:

Élément	Description
<i>year</i>	Nombre compris entre 100 et 9999 inclus, ou une expression numérique .
<i>month</i>	Toute expression numérique.
<i>day</i>	Toute expression numérique.

Notes

Pour spécifier une date telle que le 31 décembre 1991, la plage des nombres pour chaque argument **DateSerial** doit être normalement comprise dans la plage acceptée pour l'unité; autrement dit, 1–31 pour les jours et 1–12 pour les mois. Toutefois, vous pouvez aussi spécifier des dates relatives pour chaque argument en utilisant toute expression numérique représentant un certain nombre de jours, de mois ou d'années antérieurs ou postérieurs à une date donnée.

L'exemple suivant utilise des expressions numériques à la place des nombres absolus de date. Ici, la fonction **DateSerial** renvoie une date correspondant au jour précédant le premier jour (1 - 1), deux mois précédant le mois d'août (8 - 2), 10 ans avant 1990 (1990 - 10); en d'autres termes, le 31 mai 1980.

```
Dim MyDate1, MyDate2
```

```
MyDate1 = DateSerial(1970, 1, 1) ' Renv
```

```
MyDate2 = DateSerial(1990 - 10, 8 - 2, 1 - 1) ' Renv
```

Pour l'argument *year*, les valeurs comprises entre 0 et 99 inclus sont interprétées comme les années 1900–1999. Pour tous les autres arguments *year*, utilisez une année complète à quatre chiffres (par exemple, 1800).

Quand tout argument excède la plage normalement acceptée pour cet argument, l'incrémentaire s'effectue sur l'unité supérieure suivante qui convient. Par exemple, si vous spécifiez 35 jours, ils sont transformés en un mois et un certain nombre de jours, selon le moment de l'année auxquels ils s'appliquent. Toutefois, si un seul argument se situe hors de la plage -32 768 à 32 767, ou si la date spécifiée par les trois arguments, soit directement ou par expression, n'est pas comprise dans la plage acceptable des dates, une erreur se produit.

DateValue, fonction

[Voir aussi](#)

Description

Renvoie un **Variant** de sous-type **Date**.

Syntaxe

DateValue(*date*)

L'argument *date* est normalement une [expression de chaîne](#) représentant une date comprise entre le 1er janvier 100 et le 31 décembre 9999. Toutefois, l'argument *date* peut être aussi toute expression pouvant représenter une date, une heure ou à la fois une date et une heure, dans cette plage.

Notes

Si l'argument *date* inclut des informations sur l'heure, la fonction **DateValue** ne le renvoie pas. Néanmoins, si l'argument *date* inclut des informations d'heure incorrectes (telles que "89:98"), une erreur se produit.

Si l'argument *date* est une chaîne n'incluant que des nombres séparés par des [séparateurs de date](#) valides, la fonction **DateValue** reconnaît l'ordre du mois, du jour et de l'année en fonction du format date courte que vous avez spécifié sur votre système. La fonction **DateValue** reconnaît également les dates non ambiguës contenant des noms de mois, sous forme longue ou abrégée. Par exemple, outre la reconnaissance de 30/12/1991 et 30/12/91, la fonction **DateValue** reconnaît aussi 30 décembre 1991 et 30 déc 1991.

Si la partie année de l'argument *date* est omise, la fonction **DateValue** utilise l'année en cours de la date système de votre ordinateur.

L'exemple ci-dessous utilise la fonction **DateValue** pour convertir une chaîne en date. Vous pouvez également utiliser des littéraux de dates pour affecter directement une date à une variable **Variant**, par exemple MyDate = #11/9/63#.

```
Dim MyDate
```

MyDate = DateValue("11 Septembre 1963") ' Re:

Day, fonction

[Voir aussi](#)

Description

Renvoie un nombre entier compris entre 1 et 31 inclus, représentant le jour du mois.

Syntaxe

Day(*date*)

L'argument *date* peut être toute expression pouvant représenter une date. Si l'argument *date* contient [Null](#), la valeur **Null** est renvoyée.

L'exemple ci-dessous utilise la fonction **Day** pour obtenir le jour du mois d'une date spécifiée :

```
Dim MyDay
```

```
MyDay = Day("19 octobre 1962") ' MyDay contie
```

Description, propriété

[Voir aussi](#)[Application](#)

Description

Renvoie ou définit une chaîne descriptive associée à une erreur.

Syntaxe

object.**Description** [= *stringexpression*]

La syntaxe de la propriété **Description** comprend les éléments suivants :

Élément	Description
<i>object</i>	Toujours l'objet Err .
<i>stringexpression</i>	Expression de chaîne contenant une description de l'erreur.

Notes

La propriété **Description** est une courte description de l'erreur. Utilisez cette propriété pour avertir l'utilisateur d'une erreur que vous ne pouvez pas ou ne voulez pas gérer. Lors de la génération d'une erreur définie par l'utilisateur, affectez une courte description de votre erreur à cette propriété. Si la propriété **Description** n'est pas remplie et que la valeur de **Number** correspond à une [erreur d'exécution](#) VBScript, la chaîne renvoyée par la fonction Error est placée dans la propriété Description lorsque l'erreur est générée.

On Error Resume Next

Err.Raise 6 ' Génère une erreur de dépassement.

MsgBox ("Erreur N° " & CStr(Err.Number) & " " & **Err.Description**)

Err.Clear ' Efface l'erreur.

Dim, instruction

[Voir aussi](#)

Description

Déclare des variables et alloue l'espace de stockage.

Syntaxe

Dim *varname*[(*subscripts*)][, *varname*[(*subscripts*)]] ...

La syntaxe de l'instruction **Dim** comprend les éléments suivants :

Élément	Description
<i>varname</i>	Nom de la variable ; respecte les conventions standard d'affectation de nom à des variables .
<i>subscripts</i>	Dimensions d'une variable tableau ; jusqu'à 60 dimensions multiples peuvent être déclarées. L'argument <i>subscripts</i> utilise la syntaxe suivante : <i>upperbound</i> [, <i>upperbound</i>] ... La limite inférieure d'un tableau est toujours zéro.

Notes

Les variables déclarées avec **Dim** au [niveau du script](#) sont disponibles pour toutes les procédures contenues dans le script. Les variables [de niveau procédure](#), ne sont disponibles que dans la procédure.

Vous pouvez aussi utiliser l'instruction **Dim** avec des parenthèses vides pour déclarer un tableau dynamique. Une fois cette déclaration effectuée, utilisez l'instruction **ReDim** à l'intérieur d'une procédure pour définir le nombre de dimensions et d'éléments contenus dans le tableau. Si vous essayez de déclarer de nouveau une dimension pour une variable tableau dont la taille a été spécifiée explicitement dans une instruction **Dim**, une erreur se produit.

Conseil Si vous utilisez l'instruction **Dim** à l'intérieur d'une procédure, il est couramment accepté, dans la pratique générale de programmation, de placer l'instruction **Dim** au commencement de la procédure.

L'exemple ci-dessous illustre l'utilisation de l'instruction **Dim** :

```
Dim Names(9) ' Déclare un tableau à 10 éléments.  
Dim Names() ' Déclare un tableau dynamique.  
Dim MyVar, MyNum ' Déclare deux variables.
```

opérateur

[Voir aussi](#)

Description

Effectue la division entre deux nombres et renvoie un résultat à virgule flottante.

Syntaxe

result = *number1*/*number2*

La syntaxe de l'opérateur / comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>number1</i>	Toute expression numérique .
<i>number2</i>	Toute expression numérique.

Notes

Si une ou les deux expressions sont des expressions [Null](#), *result* est **Null**.
Toute expression [Empty](#) est traitée comme 0.

Do...Loop, instruction

[Voir aussi](#)

Description

Répète un bloc d'instructions tant qu'une condition est **True** ou jusqu'à ce qu'une condition devienne **True**.

Syntaxe

```
Do [{While | Until} condition]  
    [statements]  
    [Exit Do]  
    [statements]
```

Loop

Vous pouvez aussi utiliser la syntaxe suivante :

```
Do  
    [statements]  
    [Exit Do]  
    [statements]  
Loop [{While | Until} condition]
```

La syntaxe de l'instruction **Do...Loop** comprend les éléments suivants :

Élément	Description
<i>condition</i>	Expression numérique ou expression de chaîne qui est True ou False . Si <i>condition</i> est Null , l'élément <i>condition</i> est traité comme False .
<i>statements</i>	Une ou plusieurs instructions qui sont répétées tant que l'élément <i>condition</i> est True , ou jusqu'à ce qu'il le devienne.

Notes

L'instruction **Exit Do** ne peut être utilisée que dans une structure de contrôle **Do...Loop** afin de proposer une solution alternative pour quitter une instruction **Do...Loop**. Vous pouvez placer autant d'instructions **Exit Do** que vous voulez n'importe où dans l'instruction **Do...Loop**. Souvent utilisée avec l'évaluation d'une condition (par exemple, l'instruction **If...Then**), l'instruction **Exit Do** transfère le contrôle à l'instruction qui suit immédiatement **Loop**.

Quand elle est utilisée à l'intérieur d'instructions **Do...Loop** imbriquées, l'instruction **Exit Do** transfère le contrôle à la boucle située au niveau d'imbrication supérieur à celui de la boucle dans laquelle elle se déroule.

L'exemple ci-dessous illustre l'utilisation de l'instruction **Do...Loop** :

```
Do Until DefResp = vbNo
    MyNum = Int (6 * Rnd + 1) ' Génère un entier aléatoire entre 1 e
    DefResp = MsgBox (MyNum & " Voulez-vous un autre nombre?
Loop

Dim Check, Counter
Check = True: Counter = 0          ' Initialise les variables.
Do                                ' Boucle externe.
    Do While Counter < 20        ' Boucle interne.
        Counter = Counter + 1    ' Incrémente le compteur.
        If Counter = 10 Then     ' Si la condition vaut True...
            Check = False        ' Affecte la valeur False à l'indicateur
        Exit Do                  ' Quitte la boucle interne.
    End If
Loop
Loop Until Check = False         ' Quitte immédiatement la boucle
```

[Voir aussi](#)

Description

Le mot clé **Empty** est utilisé pour indiquer la valeur d'une variable non initialisée. Cette valeur diffère de la valeur **Null**.

Eqv, opérateur

[Voir aussi](#)

Description

Effectue une équivalence logique de deux expressions.

Syntaxe

result = *expression1* **Eqv** *expression2*

La syntaxe de l'opérateur **Eqv** comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>expression1</i>	Toute expression .
<i>expression2</i>	Toute expression.

Notes

Si l'une ou l'autre expression est [Null](#), *result* est **Null** également. Si aucune des expressions n'est **Null**, *result* est déterminé en fonction du tableau suivant :

Si <i>expression1</i> est	et <i>expression2</i> est	<i>result</i> est
True	True	True
True	False	False
False	True	False
False	False	True

L'opérateur **Eqv** effectue une [comparaison binaire](#) des bits ayant une position identique dans deux [expressions numériques](#), et définit le bit correspondant dans *result* d'après la table de vérité suivante :

Si le bit dans <i>expression1</i> est	et si le bit dans <i>expression2</i> est	<i>result</i> est
0	0	1
0	1	0
1	0	0
1	1	1

Erase, instruction

[Voir aussi](#)

Description

Réinitialise les éléments des [tableaux](#) de taille fixe et libère l'espace de stockage des tableaux dynamiques.

Syntaxe

Erase *array*

L'argument *array* est le nom de la [variable](#) tableau à effacer.

Notes

Il est important de savoir s'il s'agit d'un tableau de taille fixe (ordinaire) ou dynamique, car le comportement de l'instruction **Erase** varie selon le type de tableau. L'instruction **Erase** ne récupère pas de mémoire pour les tableaux de taille fixe. L'instruction **Erase** définit les éléments d'un tableau de taille fixe de la façon suivante :

Type de tableau	Effet de l'instruction Erase sur les éléments d'un tableau de taille fixe
Tableau numérique fixe	Définit chaque élément avec la valeur zéro.
Tableau de chaînes fixe	Définit chaque élément avec une chaîne de longueur nulle ("").
Tableau d'objets	Définit chaque élément avec la valeur spéciale Nothing

L'instruction **Erase** libère la mémoire utilisée par les tableaux dynamiques. Avant que votre programme puisse à nouveau faire référence au tableau dynamique, vous devez déclarer de nouveau les dimensions des variables du tableau en utilisant une instruction **ReDim**.

L'exemple ci-dessous illustre l'utilisation de l'instruction **Erase** :

```
Dim NumArray(9)
```

```
Dim DynamicArray()
```

```
ReDim DynamicArray(9) ' Allouer l'espace de sto
```

```
Erase NumArray      ' Chaque élément est réinitia
```

```
Erase DynamicArray  ' Libérer la mémoire utilisé  
tableau
```

Err, objet

[Voir aussi](#)[Propriétés](#)[Méthodes](#)

Description

Contient l'information relative aux [erreurs d'exécution](#). Accepte les méthodes **Raise** et **Clear** pour générer et effacer des erreurs d'exécution.

Notes

L'objet **Err** est un objet intrinsèque de [portée](#) globale ; il n'est pas nécessaire de créer une instance de cet objet dans votre code. Les propriétés de l'objet **Err** sont définies par le générateur d'une erreur — Visual Basic, par un objet Automation ou par le programmeur VBScript.

La propriété par défaut de l'objet **Err** est **Number**. **Err.Number** contient un entier et peut être utilisé par un [objet Automation](#) pour renvoyer un [SCODE](#).

Lorsqu'une erreur d'exécution se produit, les propriétés de l'objet **Err** contiennent des informations qui identifient de façon unique l'erreur et d'autres qui permettent de la gérer. Pour générer une erreur d'exécution dans votre code, utilisez la méthode **Raise**.

Les propriétés de l'objet **Err** sont remises à zéro ou réinitialisées avec des chaînes de longueur nulle ("") après une instruction **On Error Resume** ou **Next**. La méthode **Clear** peut être utilisée pour réinitialiser explicitement **Err**.

L'exemple ci-dessous illustre l'utilisation de l'objet **Err** :

```
On Error Resume Next
```

```
Err.Raise 6 ' Génère une erreur de dépassement.
```

```
MsgBox ("Erreur N° " & CStr(Err.Number) & " " & Err.Descriptio
```

```
Err.Clear ' Efface l'erreur.
```

Eval, fonction

[Voir aussi](#)

Description

Évalue une [expression](#) et renvoie le résultat.

Syntaxe

[result =]**Eval**(*expression*)

La syntaxe de la fonction **Eval** comporte les éléments suivants :

Élément	Description
<i>result</i>	Facultatif. Variable à laquelle est affectée une valeur de retour. Si l'élément <i>result</i> n'est pas spécifié, pensez à utiliser l'instruction Execute à la place.
<i>expression</i>	Requis. Chaîne contenant toute expression VBScript légale.

Notes

Dans VBScript, l'expression $x = y$ peut être interprétée de deux manières différentes. La première est de considérer qu'il s'agit d'une instruction permettant d'affecter la valeur y à x . La seconde implique que l'expression recherche si x et y ont la même valeur. Si c'est le cas, l'élément *result* prend la valeur **True**. Dans le cas contraire, *result* prend la valeur **False**. La méthode **Eval** utilise toujours la deuxième interprétation, alors que l'instruction **Execute** utilise la première.

Remarque Dans Microsoft® JScript, il n'y a pas de confusion possible entre l'affectation et la comparaison car l'opérateur d'affectation (=) est différent de l'[opérateur de comparaison](#) (==).

Voici un exemple qui illustre l'utilisation de la fonction **Eval** :

```
Sub GuessANumber
  Dim Guess, RndNum
  RndNum = Int((100) * Rnd(1) + 1)
  Guess = CInt(InputBox("Tapez une proposition:",
Do
  If Eval("Guess = RndNum") Then
    MsgBox "Félicitations! Vous avez deviné!"
    Exit Sub
  Else
    Guess = CInt(InputBox("Désolé! Recommence
  End If
Loop Until Guess = 0
End Sub
```

Exécute, méthode

[Voir aussi](#)[Application](#)

Description

Exécute une recherche d'expression régulière dans une chaîne spécifiée.

Syntaxe

object.**Exécute**(*string*)

La syntaxe de la méthode **Exécute** comprend les éléments suivants :

Élément	Description
<i>object</i>	Requis. Il s'agit toujours du nom d'un objet RegExp .
<i>string</i>	Requis. Chaîne de caractères à laquelle l'expression régulière est appliquée.

Notes

Les critères réels de la recherche d'expression régulière sont définis à l'aide de la propriété **Pattern** de l'objet **RegExp**.

La méthode **Exécute** renvoie une collection **Matches** contenant un objet **Match** pour chaque correspondance trouvée dans l'élément *string*. **Exécute** renvoie une collection **Matches** vide si aucune correspondance n'est trouvée.

Le code suivant montre comment utiliser la méthode **Execute** :

```
Function RegExpTest(patrn, strng)
    Dim regEx, Match, Matches      ' Crée la variable.
    Set regEx = New RegExp          ' Crée une expression régu
    regEx.Pattern = patrn           ' Définit les critères.
    regEx.IgnoreCase = True         ' Ignore la casse.
    regEx.Global = True             ' Définit une application global
    Set Matches = regEx.Execute(strng) ' Lance la recherche.
    For Each Match in Matches       ' Itère la collection Matches.
        RetStr = RetStr & "Correspondance trouvée à la position "
        RetStr = RetStr & Match.FirstIndex & ". La valeur de la corresj
        RetStr = RetStr & Match.Value & "." & vbCrLf
    Next
    RegExpTest = RetStr
End Function
```

```
MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Exécute, instruction

[Voir aussi](#)

Description

Exécute une ou plusieurs instructions spécifiées.

Syntaxe

Exécute *statement*

L'argument *statement* représente toute [expression de chaîne](#) contenant une ou plusieurs instructions à exécuter. Pour séparer des instructions multiples dans l'argument *statement*, utilisez le symbole deux-points (:) ou insérez des sauts de lignes.

Notes

Dans VBScript, l'expression $x = y$ peut être interprétée de deux manières différentes. La première est de considérer qu'il s'agit d'une instruction permettant d'affecter la valeur y à x . La deuxième implique que l'[expression](#) recherche si x et y ont la même valeur. Si c'est le cas, *result* prend la valeur **True**. Dans le cas contraire, *result* prend la valeur **False**. L'instruction **Exécute** utilise toujours la première interprétation, alors que l'instruction **Eval** utilise toujours la deuxième.

Remarque Dans Microsoft® JScript , il n'y a pas de confusion possible entre l'affectation et la comparaison car l'opérateur d'affectation (=) est différent de l'[opérateur de comparaison \(==\)](#).

Le contexte dans lequel l'instruction **Exécute** est appelée détermine les objets et les [variables](#) disponibles pour le code en cours d'exécution. Les objets et les variables dans la portée sont accessibles par le code en cours dans une instruction **Exécute**. Cependant, rappelez-vous que si vous exécutez un code qui crée une [procédure](#), cette procédure n'hérite pas la [portée](#) de la procédure dans laquelle elle a été produite.

Comme toute procédure, la portée de la nouvelle procédure est globale ; elle hérite ainsi de tous les éléments de portée globale. En revanche, son contexte ne fait pas partie de la portée globale, elle peut donc être exécutée seulement dans le contexte de la procédure où l'instruction **Exécute** est apparue.

Cependant, si cette instruction **Execute** est appelée à l'extérieur d'une procédure (c'est-à-dire, dans le cadre de la portée générale), la procédure hérite de tout ce qui se trouve dans la portée globale, mais elle peut également être appelée de toute part puisque le contexte est global. L'exemple suivant illustre ce comportement :

```
Dim X          ' Déclarer X globalement.
X = "Global"   ' Affecte une valeur à la valeur X globale.
Sub Proc1      ' Déclare la procédure.
  Dim X        ' Déclare X localement.
  X = "Local"  ' Affecte une valeur à la valeur X locale.
               ' L'instruction Execute créé ici une
               ' procédure qui, lorsqu'elle est appelée, imprime X.
               ' Elle imprime la valeur X globale car Proc2
               ' hérite des éléments de portée globale.
  Execute "Sub Proc2: Print X: End Sub"
  Print Eval("X") ' Imprime la valeur X locale.
  Proc2          ' Appelle Proc2 dans la portée de Proc1.
End Sub
Proc2           ' Cette ligne provoque une erreur car
               ' Proc2 n'est pas disponible hors de Proc1.
Proc1           ' Appelle Proc1.
  Execute "Sub Proc2: Print X: End Sub"
Proc2           ' Cet appel réussit car Proc2
               ' est maintenant disponible globalement.
```

L'exemple suivant montre comment l'instruction **Execute** peut être réécrite de sorte qu'il n'est pas nécessaire de placer la procédure entière entre guillemets :

```
S = "Sub Proc2" & vbCrLf
S = S & " Print X" & vbCrLf
S = S & "End Sub"
Execute S
```

Exit, instruction

[Voir aussi](#)

Description

Quitte un bloc du code **Do...Loop**, **For...Next**, **Function** ou **Sub**.

Syntaxe

Exit Do

Exit For

Exit Function

Exit Property

Exit Sub

La syntaxe de l'instruction **Exit** prend les formes suivantes :

Instruction	Description
Exit Do	Fournit un moyen de quitter une instruction Do...Loop . Elle ne peut être utilisée qu'à l'intérieur d'une instruction Do...Loop . Exit Do transfère le contrôle à l'instruction suivant l'instruction Loop . Quand elle est utilisée dans des instructions Do...Loop imbriquées, l'instruction Exit Do transfère le contrôle à la boucle située au niveau d'imbrication supérieur à celui de la boucle dans laquelle elle se produit.
Exit For	Fournit un moyen de quitter une boucle For . Elle ne peut être utilisée que dans une boucle For...Next ou For Each...Next . Exit For transfère le contrôle à l'instruction suivant l'instruction Next . Quand elle

	est utilisée dans des boucles For imbriquées, l'instruction Exit For transfère le contrôle à la boucle située au niveau d'imbrication supérieur à celui de la boucle dans laquelle elle se produit.
Exit Function	Quitte immédiatement la procédure Function dans laquelle elle apparaît. L'exécution se poursuit avec l'instruction suivant l'instruction ayant appelé la procédure Function .
Exit Property	Quitte immédiatement la procédure Property dans laquelle elle apparaît. L'exécution se poursuit avec l'instruction suivant l'instruction ayant appelé la procédure Property .
Exit Sub	Quitte immédiatement la procédure Sub dans laquelle elle apparaît. L'exécution se poursuit avec l'instruction suivant l'instruction ayant appelé la procédure Sub .

L'exemple ci-dessous illustre l'utilisation de l'instruction **Exit** :

Sub RandomLoop

Dim I, MyNum

Do ' Boucle infinie.

For I = 1 To 1000 ' Boucler 1000 fois.

MyNum = Int(Rnd * 100) ' Générer des non

Select Case MyNum ' Évaluer le nombre

Case 17: MsgBox "Case 17"

Exit For ' Si 17, sortir de

For...Next.

Case 29: MsgBox "Case 29"

Exit Do ' Si 29, sortir de

Do...Loop.

Case 54: MsgBox "Case 54"

Exit Sub ' Si 54, sortir de la
procédure Sub.

End Select

Next

Loop

End Sub

Exp, fonction

[Voir aussi](#)

Description

Renvoie e (la base des logarithmes népériens) élevé à une puissance.

Syntaxe

Exp(*number*)

L'argument *number* représente toute [expression numérique](#) valide.

Note

Si la valeur de l'argument *number* excède 709,782712893, une erreur se produit. La constante e est approximativement 2,718282.

Remarque La fonction **Exp** complète l'action de la fonction **Log**. Elle est parfois appelée antilogarithme.

L'exemple ci-dessous utilise la fonction **Exp** pour renvoyer e élevé à une puissance :

```
Dim MyAngle, MyHSin
' Définir l'angle en radians.
MyAngle = 1.3
' Calculer le sinus hyperbolique.
MyHSin = (Exp(MyAngle) - Exp(-1 *
MyAngle)) / 2
```

opérateur

[Voir aussi](#)

Description

Élève un nombre à la puissance de l'exposant indiqué.

Syntaxe

result = *number*^*exponent*

La syntaxe de l'opérateur ^ comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>number</i>	Toute expression numérique .
<i>exponent</i>	Toute expression numérique.

Notes

number ne peut être négatif que si *exponent* est une valeur en nombre entier. Quand plusieurs élévations à une puissance sont effectuées dans une même expression, l'opérateur ^ est évalué à mesure qu'il est rencontré, de gauche à droite.

Si *number* ou *exponent* sont des expressions [Null](#), *result* est aussi **Null**.

[Voir aussi](#)

Description

Le mot clé **False** à une valeur égale à 0.

Filter, fonction

[Voir aussi](#)

Description

Renvoie un [tableau](#) commençant par zéro contenant un sous-ensemble d'un tableau de chaîne basé sur des critères de filtre spécifiés.

Syntaxe

Filter(*InputStrings*, *Value*[, *Include*[, *Compare*]])

La syntaxe de la fonction **Filter** comprend les éléments suivants :

élément	Description
<i>InputStrings</i>	Tableau de chaîne à une dimension dans lequel la recherche doit être effectuée.
<i>Value</i>	Chaîne à rechercher.
<i>Include</i>	Facultatif. Valeur de type Boolean indiquant s'il faut renvoyer des sous-chaînes incluant ou excluant l'argument <i>Value</i> . Si l'argument <i>Include</i> a la valeur True , la fonction Filter renvoie le sous-ensemble du tableau contenant l'argument <i>Value</i> comme sous-chaîne. Si l'argument <i>Include</i> a la valeur False , la fonction Filter renvoie le sous-ensemble du tableau ne contenant pas l'argument <i>Value</i> comme sous-chaîne.
<i>Compare</i>	Facultatif. Valeur numérique indiquant le type de comparaison de chaîne à utiliser. Reportez-vous à la section Valeurs.

Valeurs

L'argument *Compare* peut prendre les valeurs suivantes :

Constante	Valeur	Description
vbBinaryCompare	0	Effectue une comparaison binaire.
vbTextCompare	1	Effectue une comparaison texte.

Notes

Si aucune correspondance de l'argument *Value* n'est trouvée dans l'argument *InputStrings*, la fonction **Filter** renvoie un tableau vide. Une erreur se produit si l'argument *InputStrings* a la valeur **Null** ou s'il ne correspond pas à un tableau à une dimension.

Le tableau renvoyé par la fonction **Filter** ne contient que le nombre d'éléments suffisants pour accueillir le nombre d'éléments en correspondance.

L'exemple ci-dessous utilise la fonction **Filter** pour renvoyer le tableau contenant le critère de recherche "Lun" :

```
Dim MyIndex  
Dim MyArray (3)  
MyArray(0) = "Dimanche"  
MyArray(1) = "Lundi"  
MyArray(2) = "Mardi"  
MyIndex = Filter(MyArray, "Lun")  
' MyIndex(0) contient "Lundi".
```

FirstIndex, propriété

[Voir aussi](#)[Application](#)

Description

Indique la position dans une chaîne de recherche où une correspondance a été trouvée.

Syntaxe

object.**FirstIndex**

L'argument *object* représente tout objet **Match**.

Notes

La propriété **FirstIndex** utilise un décalage qui commence à zéro à partir du début de la chaîne de recherche. En d'autres termes, la première lettre de la chaîne est identifiée par le caractère zéro (0). Le code suivant montre comment utiliser la propriété **FirstIndex** :

```
Function RegExpTest(patrn, strng)
    Dim regEx, Match, Matches      ' Crée la variable.
    Set regEx = New RegExp         ' Crée l'expression littérale
    regEx.Pattern = patrn         ' Définit les critères.
    regEx.IgnoreCase = True       ' Ignore la casse.
    regEx.Global = True          ' Définit le champ d'application
    Set Matches = regEx.Execute(strng) ' Lance la recherche.
    For Each Match in Matches      ' Itère la collection Matches.
        RetStr = RetStr & "Correspondance " & I & " trouvée à la posi
        RetStr = RetStr & Match.FirstIndex & ". La valeur de la corresp
```

```
    RetStr = RetStr & Match.Value & "." & vbCrLf
Next
RegExpTest = RetStr
End Function
```

```
MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Int, Fix, fonctions

[Voir aussi](#)

Description

Renvoient la partie entière d'un nombre.

Syntaxe

Int(*number*)

Fix(*number*)

L'argument *number* représente toute [expression numérique](#) valide. Si l'argument *number* contient [Null](#), la valeur **Null** est renvoyée.

Notes

Les deux fonctions **Int** et **Fix** suppriment la partie fractionnaire de l'argument *number* et renvoie la valeur entière résultante.

La différence entre les fonctions **Int** et **Fix** tient au fait que si l'argument *number* est négatif, la fonction **Int** renvoie le premier entier négatif inférieur ou égal à *number*, tandis que la fonction **Fix** renvoie le premier entier négatif supérieur ou égal à *number*. Par exemple, la fonction **Int** convertit -8,4 en -9, tandis que la fonction **Fix** convertit -8,4 en -8.

Fix(*number*) est équivalent à :

Sgn(*number*) *
Int(Abs(*number*))

Les exemples suivants illustrent de quelle façon les fonctions **Int** et **Fix** renvoient les parties entières de nombres :

MyNumber = **Int**(99.8) ' Renvoie 99.

MyNumber = **Fix**(99.2) ' Renvoie 99.
MyNumber = **Int**(-99.8) ' Renvoie -100.
MyNumber = **Fix**(-99.8) ' Renvoie -99.
MyNumber = **Int**(-99.2) ' Renvoie -100.
MyNumber = **Fix**(-99.2) ' Renvoie -99.

For...Next, instruction

[Voir aussi](#)

Description

Répète un groupe d'instructions un nombre spécifié de fois.

Syntaxe

```
For counter = start To start [Step increment]  
    [statements]  
Exit For  
    [statements]
```

Next

La syntaxe de l'instruction **For...Next** comprend les éléments suivants :

Élément	Description
<i>counter</i>	Variable numérique utilisée comme compteur de boucles. La variable ne peut être un élément de tableau ou un élément d'un type défini par l'utilisateur.
<i>start</i>	Valeur initiale de <i>counter</i> .
<i>start</i>	Valeur finale de <i>counter</i> .
<i>increment</i>	Quantité par laquelle <i>counter</i> change à chaque accomplissement de la boucle. Si cet élément n'est pas spécifié, la valeur par défaut de <i>increment</i> est 1.
<i>statements</i>	Une ou plusieurs instructions entre For et Next qui sont exécutées le nombre de fois spécifié.

Notes

L'argument *increment* peut être positif ou négatif. La valeur de l'argument *increment* détermine le traitement de la boucle de la manière suivante :

Valeur	La boucle s'exécute si
Positif ou 0	$counter \leq start$
Négatif	$counter \geq start$

Une fois que la boucle démarre et que toutes les instructions sont exécutées, l'argument *increment* est ajouté à *counter*. à ce point, les instructions contenues dans la boucle sont à nouveau exécutées (sur la base du même test ayant provoqué l'exécution initiale de la boucle) ou la boucle est quittée et l'exécution se poursuit avec l'instruction suivant l'instruction **Next**.

Conseil Changer la valeur de *counter* quand une boucle est en cours d'exécution rendra la lecture et le débogage de votre code plus difficile.

L'instruction **Exit For** ne peut être utilisée que dans une structure de contrôle **For Each...Next** ou **For...Next** pour fournir un autre moyen de quitter. Vous pouvez placer autant d'instructions **Exit For** que vous voulez n'importe où dans la boucle. L'instruction **Exit For** est souvent utilisée avec l'évaluation d'une condition (par exemple, **If...Then**) et transfère le contrôle à l'instruction succédant immédiatement à **Next**.

Vous pouvez imbriquer **For...Next** en plaçant une boucle **For...Next** dans une autre. Donnez à chaque boucle un nom de variable unique comme son élément *counter*. La construction suivante est correcte :

```
For I = 1 To 10
  For J = 1 To 10
    For K = 1 To 10
      ...
    Next
  Next
Next
```

For Each...Next, instruction

[Voir aussi](#)

Description

Répète un groupe d'instructions pour chaque élément d'un [tableau](#) ou d'une [collection](#).

Syntaxe

For Each *element* **In** *group*

[*statements*]

[Exit For]

[*statements*]

Next [*element*]

La syntaxe de l'instruction **For Each...Next** comprend les éléments suivants :

Élément	Description
<i>element</i>	Variable employée pour effectuer une itération sur les éléments de la collection ou du tableau. Pour les collections, l'argument <i>element</i> peut uniquement être une variable de type Variant , une variable de type Object générique, ou n'importe quelle variable d' objet Automation spécifique. Pour les tableaux, l'argument <i>element</i> peut être uniquement une variable de type Variant .
<i>group</i>	Nom d'une collection d'objets ou d'un tableau.
<i>statements</i>	Une ou plusieurs instructions pouvant être exécutées sur chaque élément d'un groupe indiqué par <i>group</i> .

Notes

Le bloc d'instruction **For Each** peut être tapé si l'entité *group* comporte au moins un élément. Une fois la boucle, toutes les instructions de celle-ci sont exécutées pour le premier élément de l'entité *group*. Puis, tant qu'il reste des éléments dans l'entité *group*, les instructions de la boucle continuent à s'exécuter pour chaque élément. Lorsqu'il n'y a plus d'éléments dans l'entité *group*, le programme sort de la boucle et exécute l'instruction se trouvant immédiatement après l'instruction **Next**.

L'instruction **Exit For** ne peut être utilisée qu'à l'intérieur d'une structure de contrôle **For Each...Next** ou **For...Next** pour fournir un autre mode de sortie. La boucle peut inclure un nombre illimité d'instructions **Exit For**. L'instruction **Exit For** est souvent employée avec l'évaluation d'une condition (par exemple, **If...Then**), et transfère le contrôle à l'instruction se trouvant immédiatement après **Next**.

Vous pouvez imbriquer des boucles **For Each...Next** en plaçant une boucle **For Each...Next** à l'intérieur d'une autre. Cependant, chaque *element* de boucle doit être unique.

Remarque Si vous omettez l'argument *element* dans une instruction **Next**, l'exécution se poursuit comme si vous l'aviez incluse. Si une instruction **Next** est rencontrée avant son instruction **For** correspondante, une erreur se produit.

L'exemple ci-dessous illustre l'utilisation de l'instruction **For Each...Next** :

```
Function ShowFolderList(folderspec)
  Dim fso, f, f1, fc, s
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set f = fso.GetFolder(folderspec)
  Set fc = f.Files
  For Each f1 in fc
    s = s & f1.name
    s = s & "<BR>"
  Next
  ShowFolderList = s
End Function
```



FormatCurrency, fonction

[Voir aussi](#)

Description

Renvoie une expression formatée sous forme de valeur de type Currency utilisant le symbole monétaire défini dans le Panneau de configuration du système.

Syntaxe

FormatCurrency(*Expression* [, *NumDigitsAfterDecimal* [, *IncludeLeadingDigit* [, *UseParensForNegativeNumbers* [, *GroupDigits*]]]])

La syntaxe de la fonction **FormatCurrency** comprend les éléments suivants:

élément	Description
<i>Expression</i>	Requis. Expression à formater.
<i>NumDigitsAfterDecimal</i>	Facultatif. Valeur numérique indiquant combien de positions à la droite de la décimale sont affichées. La valeur par défaut (-1) indique que les paramètres régionaux de l'ordinateur sont employés.
<i>IncludeLeadingDigit</i>	Facultatif. Constante 3-états indiquant si un zéro non significatif s'affiche pour les valeurs décimales. Reportez-vous à la section Valeurs.
	Facultatif. Constante 3-états indiquant s'il faut mettre les

<i>UseParensForNegativeNumbers</i>	valeurs négatives entre parenthèses. Reportez-vous à la section Valeurs.
<i>GroupDigits</i>	Facultatif. Constante 3-états indiquant s'il faut ou non grouper les nombres en utilisant le séparateur de groupe spécifié dans les paramètres régionaux de l'ordinateur. Reportez-vous à la section Valeurs.

Valeurs

Les arguments *IncludeLeadingDigit*, *UseParensForNegativeNumbers* et *GroupDigits* peuvent prendre les valeurs suivantes:

Constante	Value	Description
TristateTrue	-1	True
TristateFalse	0	False
TristateUseDefault	-2	Utilise les paramètres régionaux de l'ordinateur.

Notes

Lorsque un ou plusieurs arguments sont omis, les valeurs de ces arguments sont fournies par les paramètres régionaux de l'ordinateur. La position du symbole monétaire relatif à la valeur monétaire est déterminée par les paramètres régionaux du système.

Remarque Toutes les informations de paramètres sont définies dans l'onglet Symbole monétaire des Paramètres régionaux, à l'exception du zéro non significatif issu de l'onglet Nombre.

L'exemple ci-dessous utilise la fonction **FormatCurrency** pour mettre en forme l'expression en devise et l'affecter à MyCurrency :

Dim MyCurrency

MyCurrency = **FormatCurrency(1000)** ' MyCurren

FormatDateTime, fonction

[Voir aussi](#)

Description

Renvoie une expression formatée sous forme de date ou d'heure.

Syntaxe

FormatDateTime(*Date*[,*NamedFormat*])

La syntaxe de la fonction **FormatDateTime** comprend les éléments suivants:

élément	Description
<i>Date</i>	Expression de date à formater.
<i>NamedFormat</i>	Facultatif. Valeur numérique indiquant le format de date/heure utilisé. Si cette valeur est omise, vbGeneralDate est employé.

Valeurs

L'argument *NamedFormat* peut prendre les valeurs suivantes:

Constante	Value	Description
vbGeneralDate	0	Affiche une date et/ou une heure. En présence d'une partie de date, elle l'affiche sous forme de date abrégée. En présence d'une partie d'heure, elle l'affiche sous forme d'heure complète. Si les deux parties sont présentes, elles sont toutes deux affichées.
		Affiche une date en utilisant le format de

vbLongDate	1	date complet spécifié dans les paramètres régionaux de votre ordinateur.
vbShortDate	2	Affiche une date en utilisant le format de date abrégé spécifié dans les paramètres régionaux de l'ordinateur.
vbLongTime	3	Affiche une heure en utilisant le format d'heure spécifié dans les paramètres régionaux de l'ordinateur.
vbShortTime	4	Affiche une heure au format 24 heures (hh:mm).

Notes

L'exemple ci-dessous utilise la fonction **FormatDateTime** pour formater l'expression en date longue et l'affecte à MyDateTime :

Function GetCurrentDate

' FormatDateTime formate Date en date longue

GetCurrentDate = **FormatDateTime**(Date, 1)

End Function

FormatNumber, fonction

[Voir aussi](#)

Description

Renvoie une expression formatée sous forme de nombre.

Syntaxe

FormatNumber(*Expression* [, *NumDigitsAfterDecimal* [, *IncludeLeadingDigit* [, *UseParensForNegativeNumbers* [, *GroupDigits*]]]])

La syntaxe de la fonction **FormatNumber** comprend les éléments suivants:

élément	Description
<i>Expression</i>	Expression à formater.
<i>NumDigitsAfterDecimal</i>	Facultatif. Valeur numérique indiquant combien de positions à droite de la décimale sont affichées. La valeur par défaut (-1) indique que les paramètres régionaux de l'ordinateur sont employés.
<i>IncludeLeadingDigit</i>	Facultatif. Constante 3-états indiquant si un zéro non significatif est affiché pour les valeurs décimales. Reportez-vous à la section Valeurs.
<i>UseParensForNegativeNumbers</i>	Facultatif. Constante 3-états indiquant s'il faut placer ou non les valeurs négatives entre parenthèses. Reportez-vous à

	la section Valeurs.
<i>GroupDigits</i>	Facultatif. Constante 3-états indiquant si les nombres doivent être regroupés ou non à l'aide du symbole de groupement spécifié dans le Panneau de configuration. Reportez-vous à la section Valeurs.

Valeurs

Les arguments *IncludeLeadingDigit*, *UseParensForNegativeNumbers* et *GroupDigits* prennent les valeurs suivantes:

Constante	Value	Description
TristateTrue	-1	True
TristateFalse	0	False
TristateUseDefault	-2	Utilise la valeur des paramètres régionaux de l'ordinateur.

Notes

Lorsqu'un ou plusieurs des arguments facultatifs sont omis, les valeurs de ces arguments sont fournies par les paramètres régionaux de l'ordinateur.

Remarque Tous les paramètres sont issus de l'onglet Nombre des Paramètres régionaux.

L'exemple ci-dessous utilise la fonction **FormatNumber** pour formater un nombre avec quatre décimales :

Function FormatNumberDemo

Dim MyAngle, MySecant, MyNumber

MyAngle = 1.3 ' Définir l'angle en radia

MySecant = 1 / Cos(MyAngle) ' Calculer la séca

FormatNumberDemo = **FormatNumber**(MySecan

End Function

FormatPercent, fonction

[Voir aussi](#)

Description

Renvoie une expression formatée sous forme de pourcentage (multiplié par 100) avec un caractère de fin %.

Syntaxe

FormatPercent(*Expression*[,*NumDigitsAfterDecimal* [,*IncludeLeadingDigit* [,*UseParensForNegativeNumbers* [,*GroupDigits*]]]])

La syntaxe de la fonction **FormatPercent** comprend les éléments suivants:

élément	Description
<i>Expression</i>	Expression à formater.
<i>NumDigitsAfterDecimal</i>	Facultatif. Valeur numérique indiquant combien de positions à droite de la décimale s'affichent. La valeur par défaut (-1) indique que les paramètres régionaux de l'ordinateur sont employés.
<i>IncludeLeadingDigit</i>	Facultatif. Constante 3-états indiquant s'il faut ou non afficher un zéro non significatif pour les valeurs décimales. Reportez-vous à la section Valeurs.
	Facultatif. Constante 3-états indiquant s'il faut mettre les

<i>UseParensForNegativeNumbers</i>	valeurs négatives entre parenthèses. Reportez-vous à la section Valeurs.
<i>GroupDigits</i>	Facultatif. Constante 3-états indiquant si les nombres doivent ou non être regroupés avec le symbole de regroupement spécifié dans le Panneau de configuration. Reportez-vous à la section Valeurs.

Valeurs

Les arguments *IncludeLeadingDigit*, *UseParensForNegativeNumbers* et *GroupDigits* prennent les valeurs suivantes:

Constante	Value	Description
TristateTrue	-1	True
TristateFalse	0	False
TristateUseDefault	-2	Utilise la valeur des paramètres régionaux de l'ordinateur.

Notes

Lorsqu'un ou plusieurs paramètres facultatifs sont omis, les valeurs de ces paramètres sont fournies par les valeurs des paramètres régionaux de l'ordinateur.

Remarque Toutes les informations des paramètres sont issues de l'onglet Nombre des Paramètres régionaux.

L'exemple ci-dessous utilise la fonction **FormatPercent** pour formater une

expression en pourcentage :

Dim MyPercent

MyPercent = **FormatPercent(2/32)** ' MyPercent cor

Function, instruction

[Voir aussi](#)

Description

Déclare le nom, les arguments et le code qui forment le corps d'une procédure **Function**.

Syntaxe

```
[Public [Default] | Private] Function name [(arglist)]  
    [statements]  
    [name = expression]  
    [Exit Function]  
    [statements]  
    [name = expression]  
End Function
```

La syntaxe de l'instruction **Function** comprend les éléments suivants :

Élément	Description
Public	Indique que la procédure Function est accessible à toutes les autres procédures dans tous les scripts.
Default	Utilisé uniquement avec le mot clé Public dans un bloc Class pour indiquer que la procédure Function est la méthode par défaut de la classe . Une erreur se produit si plusieurs procédures Default sont spécifiées dans une classe.
Private	Indique que la procédure Function est accessible uniquement aux autres procédures du script dans lequel elle est déclarée ou que la fonction est un membre d'une classe et que la procédure Function est accessible uniquement aux autres procédures de cette classe.
	Nom de la procédure Function ; respecte les conventions standard d'affectation de

<i>name</i>	nom à des variables .
<i>arglist</i>	Liste de variables représentant les arguments qui sont transmis à la procédure Function lorsqu'elle est appelée. Plusieurs variables sont séparées par des virgules.
<i>statements</i>	Tout groupe d'instructions à exécuter dans le corps de la procédure Function .
<i>expression</i>	Valeur renvoyée de la procédure Function .

L'argument *arglist* comporte la syntaxe et les éléments suivants :

[ByVal | ByRef] varname[()]

Élément	Description
ByVal	Indique que l' argument est transmis par valeur .
ByRef	Indique que l'argument est transmis par référence .
<i>varname</i>	Nom de la variable représentant l'argument qui respecte les conventions standard d'affectation de nom à des variables.

Notes

En l'absence de spécification explicite par l'intermédiaire de **Public** ou **Private**, les procédures **Function** sont publiques par défaut, autrement dit, elles sont visibles pour toutes les autres procédures de votre script. La valeur des variables locales dans une **Function** n'est pas conservée entre les appels à la [procédure](#).

Vous ne pouvez pas définir une procédure **Function** à l'intérieur d'une autre procédure **Sub** ou **Property Get**.

L'instruction **Exit Function** provoque la sortie immédiate d'une procédure **Function**. L'exécution du programme se poursuit avec l'instruction succédant à l'instruction ayant appelé la procédure **Function**. Il n'existe pas de limite au nombre d'instructions **Exit Function** pouvant apparaître n'importe où dans une procédure **Function**.

À l'instar d'une procédure **Sub**, une procédure **Function** est une procédure distincte qui peut prendre des arguments, exécuter une

série d'instructions et changer la valeur de ses arguments. Toutefois, contrairement à une procédure **Sub**, vous pouvez utiliser une procédure **Function** sur le côté droit d'une [expression](#) de la même manière que vous utilisez une fonction intrinsèque comme **Sqr**, **Cos** ou **Chr**, quand vous voulez utiliser la valeur renvoyée par la fonction.

Vous appelez une procédure **Function** en utilisant le nom de fonction, suivi de la liste des arguments entre parenthèses, dans une expression. Pour toute information spécifique sur la manière d'appeler les procédures **Function**, consultez l'instruction **Call**.

Attention Les procédures **Function** peuvent être récursives : autrement dit, elles peuvent s'appeler elles-mêmes pour effectuer une tâche donnée. Toutefois la récursivité peut amener au dépassement de la capacité de la pile.

Pour renvoyer une valeur à partir d'une fonction, affectez cette valeur au nom de la fonction. Il n'existe pas de limite au nombre de ces affectations pouvant apparaître n'importe où dans la procédure. Si aucune valeur n'est affectée à *name*, la procédure renvoie une valeur par défaut : une fonction numérique renvoie 0 et une fonction chaîne renvoie une chaîne de longueur nulle (""). Une fonction qui renvoie une référence d'objet renvoie la valeur **Nothing** si aucune référence d'objet n'est affectée à *name* (en utilisant **Set**) dans la **Function**.

L'exemple suivant décrit la manière d'affecter une valeur renvoyée à une fonction nommée BinarySearch. Dans ce cas, la valeur **False** est affectée au nom pour indiquer qu'une certaine valeur n'a pas été trouvée.

Function BinarySearch(. . .)

. . .

```
' Valeur non trouvée, renvoie la valeur False.  
If lower > upper Then  
    BinarySearch = False  
    Exit Function  
End If  
...  
End Function
```

Les variables utilisées dans les procédures **Function** se divisent en deux catégories : celles explicitement déclarées dans la procédure et celles qui ne le sont pas. Les premières (déclarées en utilisant **Dim** ou l'équivalent) sont toujours locales pour la procédure. Les variables qui sont utilisées mais qui ne sont pas explicitement déclarées dans une procédure sont également locales à moins qu'elles n'aient été explicitement déclarées à un niveau supérieur hors de la procédure.

Attention Une procédure peut utiliser une variable qui n'est pas déclarée explicitement dans la procédure, mais un conflit peut se produire si tout élément que vous avez défini au [niveau du script](#) porte le même nom que la variable. Si votre procédure fait référence à une variable non déclarée qui porte le même nom qu'une autre procédure, [constante](#) ou variable, il est supposé que votre procédure fait référence au nom situé au niveau du script. Pour éviter ce genre de conflit, utilisez une instruction **Option Explicit** pour forcer la déclaration explicite des variables.

Attention VBScript peut réorganiser les expressions arithmétiques de manière à optimiser l'efficacité interne. évitez d'utiliser une procédure **Function** dans une expression arithmétique quand la fonction change la valeur des variables dans la même expression.

GetObject, fonction

[Voir aussi](#)

Description

Renvoie une référence à l'[objet Automation](#) d'un fichier.

Syntaxe

GetObject([*pathname*] [, *class*])

La syntaxe de la fonction **GetObject** comprend les éléments suivants :

élément	Description
<i>pathname</i>	Facultatif; chaîne. Chemin complet et nom du fichier contenant l'objet à extraire. Si l'argument <i>pathname</i> est omis, l'argument <i>class</i> est requis.
<i>class</i>	Facultatif; chaîne. Classe de l'objet.

L'[argument](#) *class* utilise la syntaxe *appname.objectype* et comprend les éléments suivants :

élément	Description
<i>appname</i>	Chaîne. Nom de l'application fournissant l'objet.
<i>objectype</i>	Chaîne. Type ou classe de l'objet à créer.

Notes

Utilisez la fonction **GetObject** pour accéder à un objet ActiveX à partir d'un fichier et affectez l'objet à une variable objet. Utilisez l'instruction **Set** pour affecter l'objet renvoyé par la fonction **GetObject** à la variable objet. Par exemple:

```
Dim CADObject
```

```
Set CADObject = GetObject("C:.CAD")
```

Lorsque ce code est exécuté, l'application associée au nom de chemin spécifié démarre, et l'objet dans le fichier spécifié est activé. Si l'argument *pathname* est une chaîne de longueur nulle (""), la fonction **GetObject** renvoie une nouvelle instance d'objet de type spécifié. Si l'argument *pathname* est omis, la fonction **GetObject** renvoie un objet actuellement actif du type spécifié. Si aucun objet du type spécifié n'existe, une erreur se produit.

Certaines applications permettent d'activer une partie d'un fichier. Ajoutez un point d'exclamation (!) à la fin du fichier et faites suivre ce dernier d'une chaîne identifiant la partie du fichier à activer. Pour plus d'informations sur la création de cette chaîne, consultez la documentation de l'application ayant créé l'objet.

Par exemple, dans une application de dessin, un fichier pourrait contenir plusieurs couches d'un dessin. Vous pouvez utiliser le code suivant pour activer une couche dans un dessin nommé **SCHEMA.CAD**:

```
Set LayerObject = GetObject("C:.CAD!Layer3")
```

Si vous ne précisez pas la classe de l'objet, l'Automation identifie l'application à démarrer et l'objet à activer, en fonction du nom de fichier que vous fournissez. Cependant, certains fichiers peuvent gérer plusieurs classes d'objets. Par exemple, un dessin peut supporter trois types d'objet: un objet d'application, un objet de dessin et un objet de barre d'outils, chacun faisant partie du même fichier. Pour spécifier l'objet d'un fichier à activer, utilisez l'argument *class* facultatif. Par exemple:

```
Dim MyObject
```

```
Set MyObject = GetObject("C:.DRW", "FIGMENT")
```

Dans l'exemple précédent, FIGMENT est le nom d'une application de

dessin et DRAWING est l'un des types d'objet qu'il gère. Une fois qu'un objet est activé, vous pouvez y faire référence dans du code en utilisant la variable objet que vous avez définie. Dans l'exemple précédent, vous accédez aux [propriétés](#) et aux méthodes du nouvel objet en utilisant la variable objet MyObject. Par exemple :

```
MyObject.Line 9, 90  
MyObject.InsertText 9, 100, "Bonjour."  
MyObject.SaveAs "C:..DRW"
```

Remarque Utilisez la fonction **GetObject** lorsqu'il existe une instance courante de l'objet, ou lorsque vous souhaitez créer l'objet avec un fichier déjà chargé. En l'absence d'instance courante, et si vous ne souhaitez pas que l'objet commence avec un fichier chargé, utilisez la fonction **CreateObject**.

Si un objet s'est enregistré comme un objet d'instance simple, une seule instance de l'objet est créée, quel que soit le nombre d'exécutions de la fonction **CreateObject**. Avec un objet à simple instance, la fonction **GetObject** renvoie toujours la même instance lorsqu'il est appelé avec la syntaxe de chaîne de longueur nulle (""), et il provoque une erreur si l'argument *pathname* est omis.

GetRef, fonction

[Voir aussi](#)

Description

Renvoie une référence à une [procédure](#) éventuellement liée à un événement.

Syntaxe

Set *object.eventname* = **GetRef**(*procname*)

La syntaxe de la fonction **GetRef** comprend les éléments suivants :

Élément	Description
<i>object</i>	Requis. Nom de l'objet avec lequel l'événement est associé.
<i>event</i>	Requis. Nom de l'événement duquel dépend la fonction.
<i>procname</i>	Requis. Chaîne contenant le nom de la procédure Sub ou Function qui va être associée à l'événement.

Notes

La fonction **GetRef** vous permet d'associer une procédure VBScript (**Function** ou **Sub**) à tout événement disponible dans vos pages DHTML (Dynamic HTML). Le modèle objet DHTML fournit des informations sur les événements disponibles pour les différents objets.

Dans d'autres langages de programmation et de script, la fonctionnalité fournie par **GetRef** se présente sous la forme d'un pointeur de fonction, c'est-à-dire, qui pointe vers l'adresse d'une procédure à exécuter lorsqu'un événement se produit.

Voici un exemple illustrant l'utilisation de la fonction **GetRef** :

```
<SCRIPT LANGUAGE="VBScript">
```

```
Function GetRefTest()
```

```
    Dim Splash
```

```
    Splash = "GetRefTest Version 1.0" & vbCrLf
```

```
    Splash = Splash & Chr(169) & " Votre Société 1999 "
```

```
    MsgBox Splash
```

```
End Function
```

```
Set Window.Onload = GetRef("GetRefTest")
```

```
</SCRIPT>
```

Global, propriété

[Voir aussi](#)[Application](#)

Description

Définit ou renvoie une valeur **booléenne** indiquant si toutes les occurrences d'une chaîne de recherche ou seulement la première chaîne doivent satisfaire aux critères.

Syntaxe

object.**Global** [= **True** | **False**]

L'argument *object* représente tout objet **RegExp**. La valeur de la propriété **Global** est **True** si la recherche s'applique à la chaîne entière et **False** dans le cas contraire. La valeur **False** est définie par défaut.

Notes

Le code suivant montre comment utiliser la propriété **Global** (modifiez la valeur affectée à la propriété **Global** pour étudier son action) :

```
Function RegExpTest(patrn, strng)
    Dim regEx, Match, Matches      ' Crée la variable.
    Set regEx = New RegExp         ' Crée une expression régulièr
    regEx.Pattern = patrn         ' Définit les critères.
    regEx.IgnoreCase = True       ' Ignore la casse.
```

```
regEx.Global = True           ' Définit une application global
Set Matches = regEx.Execute(strng) ' Lance la recherche.
For Each Match in Matches     ' Itère la collection Matches.
    RetStr = RetStr & "Correspondance trouvée à la position "
    RetStr = RetStr & Match.FirstIndex & ". La valeur de la corres|
    RetStr = RetStr & Match.Value & "." & vbCRLF
Next
RegExpTest = RetStr
End Function

MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Hex, fonction

[Voir aussi](#)

Description

Renvoie une chaîne représentant la valeur hexadécimale d'un nombre.

Syntaxe

Hex(*number*)

L'argument *number* représente toute expression valide.

Notes

Si l'argument *number* n'est pas déjà un nombre entier, il est arrondi au nombre entier le plus proche avant d'être évalué.

Si <i>number</i> est	Hex renvoie
Null	Null.
Empty	Zéro (0).
Tout autre nombre	Huit caractères hexadécimaux maximum.

Vous pouvez représenter des nombres hexadécimaux directement en les faisant précéder de &H dans la plage correcte. Par exemple, en numérotation hexadécimale, &H10 représente le nombre décimal 16.

L'exemple ci-dessous utilise la fonction **Hex** pour renvoyer la valeur hexadécimale d'un nombre:

Dim MyHex

MyHex = Hex(5) ' Renvoie 5.

MyHex = Hex(10) ' Renvoie A.

MyHex = Hex(459) ' Renvoie 1CB.



HelpContext, propriété

[Voir aussi](#)

[Application](#)

Description

Définit ou renvoie un identificateur de contexte pour une rubrique dans un fichier d'aide.

Syntaxe

object.**HelpContext** [= *contextID*]

La syntaxe de la propriété **HelpContext** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours à l'objet Err .
<i>contextID</i>	Facultatif. Identificateur valide pour une rubrique du fichier d'aide.

Notes

Si un fichier d'aide est spécifié dans la propriété **HelpFile**, la propriété **HelpContext** est employée pour afficher automatiquement la rubrique d'aide identifiée. Si les propriétés **HelpFile** et **HelpContext** sont vides, la valeur de la propriété **Number** est vérifiée et, si elle correspond à une valeur [d'erreur d'exécution](#) VBScript, l'identificateur de contexte d'aide VBScript pour l'erreur est employée. Si la valeur de la propriété **Number** ne correspond pas à une erreur VBScript, l'écran de sommaire du fichier d'aide de VBScript s'affiche.

L'exemple ci-dessous illustre l'utilisation de la propriété **HelpContext** :

On Error Resume Next

```
Dim Msg
Err.Clear
Err.Raise 6 ' Génère une erreur de dépassement.
Err.Helpfile = "yourHelp.hlp"
Err.HelpContext = yourContextID
If Err.Number <> 0 Then
    Msg = "Appuyez sur F1 ou Aide pour afficher la rubrique " & Err.Description
    " le contexte d'aide suivant: " & Err.HelpContext
    MsgBox Msg, "erreur: " & Err.Description, Err.Helpfile, Err.HelpContext
End If
```

HelpFile, propriété

[Voir aussi](#)[Application](#)

Description

Définit ou renvoie le chemin complet d'un fichier d'aide.

Syntaxe

object.**HelpFile** [= *contextID*]

La syntaxe de la propriété **HelpFile** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours à l'objet Err .
<i>contextID</i>	Facultatif. Chemin complet du fichier d'aide.

Notes

Si un fichier d'aide est spécifié dans la propriété **HelpFile**, il est automatiquement appelé lorsque l'utilisateur clique sur le bouton Aide (ou appuie sur la touche F1) dans la boîte de dialogue de message d'erreur. Si la propriété **HelpContext** contient un identificateur de contexte valide pour le fichier spécifié, cette rubrique est automatiquement affichée. Si aucune propriété **HelpFile** n'est spécifiée, le fichier d'aide VBScript s'affiche.

```
On Error Resume Next
```

```
Dim Msg
```

```
Err.Clear
```

```
Err.Raise 6 ' Génère une erreur de dépassement.
```

```
Err.Helpfile = "yourHelp.hlp"
```

```
Err.HelpContext = yourContextID
```

```
If Err.Number <> 0 Then
```

```
    Msg = "Appuyez sur F1 ou Aide pour afficher la rubrique " & E
```

```
" le contexte d'aide suivant: " & Err.HelpContext  
MsgBox Msg, "erreur: " & Err.Description, Err.Helpfile, Err.H  
End If
```

Hour, fonction

[Voir aussi](#)

Description

Renvoie un nombre entier compris entre 0 et 23 inclus, représentant l'heure du jour.

Syntaxe

Hour(*time*)

L'argument *time* représente toute expression pouvant représenter une heure. Si l'argument *time* contient **Null**, la valeur **Null** est renvoyée.

L'exemple ci-dessous utilise la fonction **Hour** pour extraire le chiffre de l'heure à partir de l'heure actuelle :

```
Dim MyTime, MyHour
```

```
MyTime = Now
```

```
MyHour = Hour(MyTime) ' MyHour contient le  
' le chiffre de l'heure actuelle.
```

If...Then...Else, instruction

Description

Exécute un groupe d'instructions soumises à une condition, en fonction de la valeur d'une expression.

Syntaxe

If *condition* **Then** *statements* [**Else** *elstatements*]

Ou, vous pouvez utiliser la syntaxe suivante, plus polyvalente :

If *condition* **Then**
 statements
[ElseIf *condition-n* **Then**
 elseifstatements] . . .
[Else
 elstatements]
End If

La syntaxe de l'instruction **If...Then...Else** comporte les éléments suivants :

Élément	Description
<i>condition</i>	<p>Un ou plusieurs types d'expressions suivants :</p> <p>Une expression numérique ou une expression de chaîne qui produit la valeur True ou False. Si <i>condition</i> est Null, l'élément <i>condition</i> est traité comme False.</p> <p>Une expression de la forme TypeOf <i>objectname</i> Is <i>objecttype</i>. L'élément <i>objectname</i> est toute référence d'objet et l'élément <i>objecttype</i> est tout type d'objet valide. L'expression est True si <i>objectname</i> est le type d'objet spécifié par <i>typeobjet</i> ; dans le cas contraire, sa valeur est False.</p>

statements Une ou plusieurs instructions séparées par deux-points ; exécutées si *condition* est **True**. *condition-n* Condition identique. *elseifstatements* Une ou plusieurs instructions exécutées si la *condition-n* est **True**. *elstatements* Une ou

plusieurs instructions exécutées si aucune expression de *condition* ou *condition-n* n'est **True**.

Notes

Vous pouvez utiliser la forme en ligne simple (première syntaxe) pour les tests courts et simples. Toutefois, la forme en bloc (seconde syntaxe) fournit une structure plus solide et une plus grande souplesse que la forme en ligne simple, et elle est souvent plus facile à lire, à mettre à jour et à déboguer.

Remarque Avec la syntaxe en ligne simple, il est possible de provoquer l'exécution de plusieurs instructions comme résultat d'une décision **If...Then**, mais elles doivent toutes être sur la même ligne et séparées par deux- points, comme dans l'instruction suivante:

```
If A > 10 Then A = A + 1 : B =  
B + A : C = C + B
```

Lors de l'exécution d'un bloc **If** (seconde syntaxe), *condition* est testé. Si *condition* est **True**, les instructions suivant **Then** sont exécutées. Si *condition* est **False**, chaque clause **Elseif** (s'il en existe) est évaluée à tour de rôle. Quand une condition **True** est trouvée, les instructions suivant l'élément **Then** sont exécutées. Si aucune des instructions **Elseif** n'est **True** (ou s'il n'existe pas de clause **Elseif**), les instructions suivant **Else** sont exécutées. Après l'exécution des instructions suivant **Then** ou **Else**, l'exécution se poursuit avec l'instruction suivant **End If**.

Les clauses **Else** et **Elseif** sont toutes deux facultatives. Vous pouvez intégrer autant d'instructions **Elseif** que vous voulez dans un bloc **If**, mais aucune ne peut apparaître après la clause **Else**. Les instructions de blocs **If** peuvent être imbriquées, autrement dit, se contenir l'une l'autre.

Ce qui suit le mot clé **Then** est examiné pour déterminer si une instruction est un bloc **If** ou non. Si tout élément autre qu'un commentaire apparaît après **Then** sur la même ligne, l'instruction est traitée comme une instruction **If** en ligne simple.

Une instruction contenant des blocs **If** doit être la première instruction sur une ligne. Le bloc **If** doit terminer une instruction **End If**.

IgnoreCase, propriété

[Voir aussi](#)

[Application](#)

Description

Définit ou renvoie une valeur **booléenne** indiquant si les critères de recherche distinguent les minuscules et les majuscules.

Syntaxe

object.IgnoreCase [= **True** | **False**]

L'argument *object* représente toujours un objet **RegExp**. La valeur de la propriété **IgnoreCase** est **False** si la recherche respecte la casse. Elle prend la valeur **True** dans le cas contraire. La valeur par défaut est **False**.

Notes

Le code suivant montre comment utiliser la propriété **IgnoreCase** (modifiez la valeur affectée à la propriété **IgnoreCase** pour étudier son action) :

```
Function RegExpTest(patrn, strng)
    Dim regEx, Match, Matches      ' Crée la variable.
    Set regEx = New RegExp         ' Crée une expression régu
    regEx.Pattern = patrn         ' Définit les critères.
```

```
regEx.IgnoreCase = True           ' Ignore la casse.
regEx.Global = True              ' Définit une application global
Set Matches = regEx.Execute(strng) ' Lance la recherche.
For Each Match in Matches        ' Itère la collection Matches.
    RetStr = RetStr & "Correspondance trouvée à la position "
    RetStr = RetStr & Match.FirstIndex & ". La valeur de la corresj
    RetStr = RetStr & Match.Value & "." & vbCrLf
Next
RegExpTest = RetStr
End Function

MsgBox(RegExpTest("is", "IS1 is2 IS3 is4"))
```

Imp, opérateur

[Voir aussi](#)

Description

Effectue une implication logique entre deux expressions.

Syntaxe

result = *expression1* **Imp** *expression2*

La syntaxe de l'opérateur **Imp** comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>expression1</i>	Toute expression .
<i>expression2</i>	Toute expression.

Notes

Le tableau suivant illustre la manière dont est déterminé l'élément *result* :

Si <i>expression1</i> est	et <i>expression2</i> est	Alors <i>result</i> vaut
True	True	True
True	False	False
True	Null	Null
False	True	True
False	False	True
False	Null	True
Null	True	True
Null	False	Null

Null **Null** **Null**

L'opérateur **Imp** effectue une [comparaison binaire](#) des bits de position identique dans deux [expressions numériques](#) et définit le bit correspondant dans *result* d'après la table suivante :

Si le bit dans <i>expression1</i> est	et le bit dans <i>expression2</i> est	Alors <i>result</i> vaut
0	0	1
0	1	1
1	0	0
1	1	1

Initialize, événement

[Voir aussi](#)

[Application](#)

Description

Se produit lorsqu'une instance de la [classe](#) associée est créée.

Syntaxe

```
Private Sub Class_Initialize()
```

```
    instructions
```

```
End Sub
```

La partie *instructions* consiste en zéro instruction de code ou plus à exécuter lors de l'initialisation de la classe.

Notes

L'exemple ci-dessous illustre l'utilisation de l'événement

Initialize :

```
Class TestClass
```

```
    Private Sub Class_Initialize ' Configuration de l'événement Initiali
```

```
        MsgBox("TestClass démarré")
```

```
    End Sub
```

```
    Private Sub Class_Terminate      ' Configuration de l'événement
```

```
        MsgBox("TestClass terminé")
```

```
    End Sub
```

End Class

Set X = New TestClass ' Créer une instance de TestClass.

Set X = Nothing ' Détruire l'instance.

InputBox, fonction

[Voir aussi](#)

Description

Affiche une invite dans une boîte de dialogue, attend que l'utilisateur entre du texte ou choisisse un bouton et renvoie le contenu de la zone de texte.

Syntaxe

InputBox(*prompt*[, *title*][, *default*][, *xpos*][, *ypos*][, *helpfile*, *context*])

La syntaxe de la fonction **InputBox** comprend les éléments suivants :

élément	Description
<i>prompt</i>	Expression de chaîne qui est affichée sous la forme d'un message dans la boîte de dialogue. La longueur maximum de l'argument <i>prompt</i> est environ 1024 caractères, selon la largeur des caractères utilisés. Si l'argument <i>prompt</i> se compose de plusieurs lignes, vous pouvez les séparer en utilisant un caractère de retour chariot (Chr(13)), un caractère de retour à la ligne (Chr(10)) ou une combinaison de ces deux caractères (Chr(13) & Chr(10)).
<i>title</i>	Expression de chaîne qui est affichée dans la barre de titre de la boîte de dialogue. Si vous omettez l'argument <i>title</i> , le nom de l'application s'affiche dans la barre de titre.
<i>default</i>	Expression de chaîne qui est affichée dans la zone de texte comme la réponse par défaut si aucune autre entrée n'est fournie. Si vous omettez l'argument <i>default</i> , la zone de texte s'affiche vide.
	Expression numérique qui spécifie, en twips, la distance horizontale entre le bord gauche de la boîte de

<i>xpos</i>	dialogue et le bord gauche de l'écran. Si l'argument <i>xpos</i> est omis, la boîte de dialogue est centrée horizontalement.
<i>ypos</i>	Expression numérique qui spécifie, en twips, la distance verticale entre le bord supérieur de la boîte de dialogue et le haut de l'écran. Si l'argument <i>ypos</i> est omis, la boîte de dialogue est positionnée verticalement, de manière approximative, à une distance d'un tiers de la taille de l'écran à partir du haut.
<i>helpfile</i>	Expression de chaîne qui identifie le fichier d'aide à utiliser pour fournir l'aide contextuelle de la boîte de dialogue. Si l'argument <i>helpfile</i> est fourni, l'argument <i>context</i> doit l'être aussi.
<i>context</i>	Expression numérique qui identifie le numéro de contexte de l'aide affecté par l'auteur de l'Aide à la rubrique d'aide correspondante. Si l'argument <i>context</i> est fourni, l'argument <i>helpfile</i> doit l'être aussi.

Notes

Lorsque les arguments *helpfile* et *context* sont tous deux fournis, un bouton d'aide est ajouté automatiquement à la boîte de dialogue.

Si l'utilisateur clique sur **OK** ou appuie sur **ENTRÉE**, la fonction **InputBox** renvoie ce qui se trouve dans la zone de texte. Si l'utilisateur clique sur **Annuler**, la fonction renvoie une chaîne de longueur nulle ("").

L'exemple ci-dessous utilise la fonction **InputBox** pour afficher une boîte de saisie et affecter la chaîne à la variable Input :

Dim Input

Input = **InputBox**("Entrez votre nom")

MsgBox ("Vous avez entré: " & Input)

InStr, fonction

[Voir aussi](#)

Description

Renvoie la position de la première occurrence d'une chaîne à l'intérieur d'une autre.

Syntaxe

InStr([*start*,]*string1*, *string2*[, *compare*])

La syntaxe de la fonction **InStr** comprend les éléments suivants :

Élément	Description
<i>start</i>	Facultatif. Expression numérique qui définit la position de départ de chaque recherche. Si cet argument est omis, la recherche commence à la position du premier caractère. Si l'argument <i>start</i> contient la valeur Null , une erreur se produit. L'argument <i>start</i> est requis si l'argument <i>compare</i> est spécifié.
<i>string1</i>	Expression de chaîne faisant l'objet de la recherche.
<i>string2</i>	Expression de chaîne recherchée.
<i>compare</i>	Facultatif. Valeur numérique qui indique le type de comparaison effectué lors de l'évaluation des sous-chaînes. Reportez-vous à la section Valeurs. Si l'argument <i>compare</i> est omis, une comparaison binaire est effectuée.

Valeurs

L'argument *compare* peut prendre les valeurs suivantes :

Constante	Valeur	Description
<code>vbBinaryCompare</code>	0	Effectue une comparaison binaire.
<code>vbTextCompare</code>	1	Effectue une comparaison texte.

Valeurs renvoyées

La fonction **InStr** renvoie les valeurs suivantes :

Si	la fonction InStr renvoie
<i>string1</i> est de longueur nulle	0
<i>string1</i> est Null	Null
<i>string2</i> est de longueur nulle	<i>start</i>
<i>string2</i> est Null	Null
<i>string2</i> n'est pas trouvé	0
<i>string2</i> est trouvé dans <i>string1</i>	la position de correspondance
<i>start</i> > Len (<i>string2</i>)	0

Notes

Les exemples suivants utilisent la fonction **InStr** pour rechercher une chaîne :

```
Dim SearchString, SearchChar, MyPos
SearchString = "XXpXXpXXPXXP"
SearchChar = "P" ' Recherche
MyPos = Instr(4, SearchString, SearchChar, 1) ' Position
```

```

MyPos = Instr(1, SearchString, SearchChar, 0) ' position 4
MyPos = Instr(SearchString, SearchChar) ' position 1
MyPos = Instr(SearchString, SearchChar, 9) ' (le dernier)
MyPos = Instr(1, SearchString, "W") ' Renvoie 9
MyPos = Instr(SearchString, "W") ' Renvoie 6

```

Remarque Une autre fonction (**InStrB**) est disponible pour être utilisée avec les données de type octet contenues dans une chaîne. Au lieu de renvoyer la position du caractère de la première occurrence d'une chaîne à l'intérieur d'une autre, la fonction **InStrB** renvoie la position de l'octet.

InStrRev, fonction

[Voir aussi](#)

Description

Renvoie la position d'une occurrence d'une chaîne dans une autre, à partir de la fin de la chaîne.

Syntaxe

InStrRev(*string1*, *string2*[, *start*[, *compare*]])

La syntaxe de la fonction **InStrRev** comprend les éléments suivants :

élément	Description
<i>string1</i>	Expression de chaîne dans laquelle la recherche est effectuée.
<i>string2</i>	Expression de chaîne recherchée.
<i>start</i>	Facultatif. Expression numérique qui définit la position de départ de chaque recherche. Si elle est omise, -1 est employé, ce qui signifie que la recherche commence à la dernière position de caractère. Si l'argument <i>start</i> contient la valeur Null , une erreur se produit.
<i>compare</i>	Facultatif. Valeur numérique indiquant le type de comparaison à utiliser lors de l'évaluation des sous-chaînes. Si elle est omise, une comparaison binaire est effectuée. Reportez-vous à la section Valeurs.

Valeurs

L'argument *compare* peut prendre les valeurs suivantes :

Constante	Valeur	Description
vbBinaryCompare	0	Effectue une comparaison binaire.

vbTextCompare 1 Effectue une comparaison texte.

Valeurs renvoyées

La fonction **InStrRev** renvoie les valeurs suivantes :

Si	la fonction InStrRev renvoie
<i>string1</i> a une longueur nulle	0
<i>string1</i> a la valeur Null	Null
<i>string2</i> a une longueur nulle	<i>start</i>
<i>string2</i> a la valeur Null	Null
<i>string2</i> est introuvable	0
<i>string2</i> se trouve à l'intérieur de <i>string1</i>	la position à laquelle une correspondance est trouvée.
<i>start</i> > Len(string2)	0

Notes

Les exemples suivants utilisent la fonction **InStrRev** pour rechercher une chaîne :

```
Dim SearchString, SearchChar, MyPos
SearchString = "XXpXXpXXPXXP" ' Chaîne de recherche
SearchChar = "P" ' Rechercher "P".
MyPos = InstrRev(SearchString, SearchChar, 10, 0) ' Comparaison à la
' position 10. Renvoie 9.
MyPos = InstrRev(SearchString, SearchChar, -1, 1) ' Comparaison à la
' dernière position. Renvoie 1.
MyPos = InstrRev(SearchString, SearchChar, 8) ' La comparaison à la
' position 8 (le premier argument est omis). Renvoie 8.
```

Remarque La syntaxe de la fonction **InStrRev** est différente de celle de la fonction **InStr**.



opérateur

[Voir aussi](#)

Description

Effectue la division de deux nombres et renvoie le résultat sous forme de nombre entier.

Syntaxe

result = *number1**number2*

La syntaxe de l'opérateur \ comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>number1</i>	Toute expression numérique .
<i>number2</i>	Toute expression numérique.

Notes

Avant d'effectuer la division, les expressions numériques sont arrondies en expression de sous-type **Byte**, **Integer** ou **Long**.

Si une expression est **Null**, *result* est également **Null**. Toute expression qui est **Empty** est traitée comme 0.

opérateur

[Voir aussi](#)

Description

Compare deux variables de référence à un objet.

Syntaxe

result = *object1* **Is** *object2*

La syntaxe de l'opérateur **Is** comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>object1</i>	Tout nom d'objet.
<i>object2</i>	Tout nom d'objet.

Notes

Si *object1* et *object2* font tous deux référence au même objet, *result* est **True** ; si tel n'est pas le cas, *result* est **False**. Il existe plusieurs manières de faire en sorte que deux variables fassent référence au même objet.

Dans l'exemple suivant, A a été défini pour faire référence au même objet que B :

Set A = B

Dans l'exemple suivant, A et B font référence au même objet que C :

Set A = C

Set B = C

IsArray, fonction

[Voir aussi](#)

Description

Renvoie une valeur booléenne indiquant si la variable est un [tableau](#).

Syntaxe

IsArray(*varname*)

L'argument *varname* représente toute [variable](#).

Notes

La fonction **IsArray** renvoie la valeur **True** si la variable est un tableau si tel n'est pas le cas, elle renvoie la valeur **False**. La fonction **IsArray** est particulièrement utile avec des variants contenant des tableaux.

L'exemple ci-dessous utilise la fonction **IsArray** pour tester si MyVariable est un tableau :

```
Dim MyVariable  
Dim MyArray(3)  
MyArray(0) = "Dimanche"  
MyArray(1) = "Lundi"  
MyArray(2) = "Mardi"
```

MyVariable = **IsArray**(MyArray) ' MyVariable con

IsDate, fonction

[Voir aussi](#)

Description

Renvoie une valeur booléenne indiquant si une expression peut être convertie en date.

Syntaxe

IsDate(*expression*)

L'argument *expression* représente toute [date](#) ou [expression de chaîne](#) reconnaissable sous forme de date ou d'heure.

Notes

La fonction **IsDate** renvoie la valeur **True** si l'expression est une date ou si elle peut être convertie en date valide; si tel n'est pas le cas, elle renvoie la valeur **False**. Dans Microsoft Windows, la plage des dates valides est comprise entre le 1er janvier 100 (après J.C.) et le 31 décembre 9999 (après J.C.); cette plage varie selon les systèmes d'exploitation.

L'exemple ci-dessous utilise la fonction **IsDate** pour déterminer si une expression peut être convertie en date :

```
Dim MyDate, YourDate, NoDate, MyCheck
MyDate = "19 octobre 1962": YourDate = #19/10/62
MyCheck = IsDate(MyDate)           ' Renvoie True
MyCheck = IsDate(YourDate)         ' Renvoie True
MyCheck = IsDate(NoDate)           ' Renvoie False
```

IsEmpty, fonction

[Voir aussi](#)

Description

Renvoie une valeur booléenne indiquant si une variable a été initialisée.

Syntaxe

IsEmpty(*expression*)

L'argument *expression* représente toute [expression](#). Cependant, dans la mesure où la fonction **IsEmpty** est utilisée pour déterminer si des variables individuelles sont initialisées, l'argument *expression* est très souvent un nom de [variable](#) simple.

Notes

La fonction **IsEmpty** renvoie la valeur **True** si la variable n'est pas initialisée ou explicitement définie sur [Empty](#); si tel n'est pas le cas, elle renvoie la valeur **False**. La valeur **False** est toujours renvoyée si l'argument *expression* contient plusieurs variables.

L'exemple ci-dessous utilise la fonction **IsEmpty** pour déterminer si une variable a été initialisée :

```
Dim MyVar, MyCheck
MyCheck = IsEmpty(MyVar)    ' Renvoie True.
MyVar = Null                ' Affecte la valeur Null.
MyCheck = IsEmpty(MyVar)    ' Renvoie False.
```

```
MyVar = Empty           ' Affecte la valeur Empt  
MyCheck = IsEmpty(MyVar) ' Renvoie True.
```

IsNull, fonction

[Voir aussi](#)

Description

Renvoie une valeur booléenne indiquant si une expression contient des données valides ou non ([Null](#)).

Syntaxe

IsNull(*expression*)

L'argument *expression* représente toute [expression](#).

Notes

La fonction **IsNull** renvoie la valeur **True** si l'argument *expression* est **Null**, autrement dit s'il ne contient aucune donnée valide ; dans le cas contraire, la fonction **IsNull** renvoie la valeur **False**. Si l'argument *expression* se compose de plusieurs variables, la présence de la valeur **Null** dans toute variable constituante provoque le renvoi de la valeur **True** pour l'expression entière.

La valeur **Null** indique que la variable ne contient aucune donnée valide. La valeur **Null** est différente de la valeur [Empty](#) qui indique qu'une variable n'a pas encore été initialisée. Elle diffère également d'une chaîne de longueur nulle que l'on appelle parfois chaîne vide.

Important Utilisez la fonction **IsNull** pour déterminer si une expression contient une valeur **Null**. Les expressions qui donneront vraisemblablement comme résultat **True** dans certaines circonstances, telles que `If Var = Null` et `If Var <> Null`, sont toujours **False**, car toute expression contenant une valeur **Null** est elle-même **Null** et donc **False**.

L'exemple ci-dessous utilise la fonction **IsNull** pour déterminer si une variable contient une valeur **Null** :

```
Dim MyVar, MyCheck
MyCheck = IsNull(MyVar)    ' Renvoie False.
MyVar = Null                ' Affecte la valeur Null.
MyCheck = IsNull(MyVar)    ' Renvoie True.
MyVar = Empty               ' Affecte la valeur Empty
MyCheck = IsNull(MyVar)    ' Renvoie False.
```

IsNumeric, fonction

[Voir aussi](#)

Description

Renvoie une valeur booléenne indiquant si une expression peut être évaluée sous la forme d'un nombre.

Syntaxe

IsNumeric(*expression*)

L'argument *expression* représente toute [expression](#).

Notes

IsNumeric renvoie **True** si la totalité de l'expression *expression* est reconnue comme un nombre ; sinon elle renvoie **False**. **IsNumeric** renvoie **False** si *expression* est une [expression de date](#).

L'exemple ci-dessous utilise la fonction **IsNumeric** pour déterminer si une variable peut être évaluée en nombre :

```
Dim MyVar, MyCheck
MyVar = 53           ' Affecte une valeur.
MyCheck = IsNumeric(MyVar) ' Renvoie True.
MyVar = "459.95"    ' Affecte une valeur.
MyCheck = IsNumeric(MyVar) ' Renvoie True.
MyVar = "45 Help"   ' Affecte une valeur.
MyCheck = IsNumeric(MyVar) ' Renvoie False.
```

IsObject, fonction

[Voir aussi](#)

Description

Renvoie une valeur booléenne indiquant si une expression référence un [objet Automation](#) valide.

Syntaxe

IsObject(*expression*)

L'argument *expression* représente toute [expression](#).

Note

La fonction **IsObject** renvoie la valeur **True** si l'argument *expression* est une variable de sous-type **Object** ou un objet défini par l'utilisateur ; si ce n'est pas le cas, elle renvoie la valeur **False**.

L'exemple ci-dessous utilise la fonction **IsObject** pour déterminer si un identificateur représente une variable objet :

```
Dim MyInt, MyCheck, MyObject
Set MyObject = Me
MyCheck = IsObject(MyObject) ' Renvoie True.
MyCheck = IsObject(MyInt)    ' Renvoie False.
```

Join, fonction

[Voir aussi](#)

Description

Renvoie une chaîne créée par la jonction de plusieurs sous-chaînes contenues dans un [tableau](#).

Syntaxe

Join(*list*[, *delimiter*])

La syntaxe de la fonction **Join** comprend les éléments suivants :

élément	Description
<i>list</i>	Un tableau à une dimension contenant les sous-chaînes à joindre.
<i>delimiter</i>	Facultatif. Caractère de chaîne utilisé pour séparer les sous-chaînes dans la chaîne renvoyée. S'il est omis, le caractère (" ") est employé. Si l'argument <i>delimiter</i> est une chaîne de longueur nulle, tous les éléments de la liste sont concaténés sans séparateurs.

Notes

L'exemple ci-dessous utilise la fonction **Join** pour joindre les sous-chaînes de MyArray :

```
Dim MyString  
Dim MyArray(4)  
MyArray(0) = "M."
```

MyArray(1) = "Alfred "

MyArray(2) = "Gautier "

MyArray(3) = "junior"

MyString = **Join**(MyArray) ' MyString contient "
junior".

LBound, fonction

[Voir aussi](#)

Description

Renvoie le plus petit indice disponible pour la dimension indiquée d'un [tableau](#).

Syntaxe

LBound(*arrayname*[, *dimension*])

La syntaxe de la fonction **LBound** comprend les éléments suivants :

Élément	Description
<i>arrayname</i>	Nom de la variable du tableau ; respecte les conventions standard d'affectation de noms à des variables .
<i>dimension</i>	Nombre entier indiquant quelle limite inférieure de la dimension est renvoyée. Utilisez 1 pour la première dimension, 2 pour la deuxième, etc. Si l'élément <i>dimension</i> est omis, 1 est supposé.

Notes

La fonction **LBound** est utilisée avec la fonction **UBound** pour déterminer la taille d'un tableau. Utilisez la fonction **UBound** pour trouver la limite supérieure d'une dimension de tableau.

La limite inférieure de toute dimension est toujours 0.

LCASE,

fonction

[Voir aussi](#)

Description

Renvoie une chaîne qui a été convertie en minuscules.

Syntaxe

LCASE(*string*)

L'argument *string* représente toute [expression de chaîne](#) valide. Si l'argument *string* contient **Null**, la valeur **Null** est renvoyée.

Note

Seules les majuscules sont converties en minuscules ; toutes les lettres en minuscules et les caractères autres que les lettres demeurent inchangés.

L'exemple ci-dessous utilise la fonction **LCASE** pour convertir les lettres majuscules en lettres minuscules :

```
Dim MyString
Dim LCaseString
MyString = "VBSCript"
LCaseString = LCASE(MyString) ' LCaseString contient "vbscript"
```

Left, fonction

[Voir aussi](#)

Description

Renvoie un nombre spécifié de caractères à partir de la gauche d'une chaîne.

Syntaxe

Left(*string*, *length*)

La syntaxe de la fonction **Left** comprend les éléments suivants :

élément	Description
<i>string</i>	Expression de chaîne à partir de laquelle les caractères situés à l'extrême gauche sont renvoyés. Si <i>string</i> contient Null , la valeur Null est renvoyée.
<i>length</i>	Expression numérique indiquant le nombre de caractères à renvoyer. Si 0, une chaîne de longueur nulle est renvoyée. Si supérieure ou égale aux nombres de caractères contenus dans <i>string</i> , la chaîne entière est renvoyée.

Notes

Pour déterminer le nombre de caractères contenus dans *string*, utilisez la fonction **Len**.

L'exemple ci-dessous utilise la fonction **Left** pour renvoyer les trois premiers caractères de MyString :

```
Dim MyString, LeftString
```

```
MyString = "VBScript"
```

```
LeftString = Left(MyString, 3) ' LeftString contien
```

Remarque Une autre fonction (**LeftB**) est disponible pour être utilisée avec les données de type octet contenues dans une chaîne. Au lieu de spécifier le nombre de caractères à renvoyer, *length* spécifie le nombre d'octets.

fonction

[Voir aussi](#)

Description

Renvoie le nombre de caractères contenus dans une chaîne, ou le nombre d'octets requis pour mémoriser une variable.

Syntaxe

Len(*string* | *varname*)

La fonction **Len** comprend les éléments suivants :

Élément	Description
<i>string</i>	Toute expression de chaîne valide. Si <i>string</i> contient Null , la valeur Null est renvoyée.
<i>varname</i>	Tout nom de variable valide. Si <i>varname</i> contient Null , la valeur Null est renvoyée.

Notes

L'exemple ci-dessous utilise la fonction **Len** pour renvoyer le nombre de caractères d'une chaîne :

```
Dim MyString
```

```
MyString = Len("VBSCRIPT") ' MyString contient 8.
```

Remarque Une autre fonction (**LenB**) est disponible pour être utilisée avec les données de type octet contenues dans une chaîne. Au lieu de renvoyer le nombre de caractères d'une chaîne, la fonction **LenB** renvoie le nombre d'octets utilisés pour représenter cette chaîne.



Length, propriété

[Voir aussi](#)[Application](#)

Description

Renvoie la longueur d'une correspondance trouvée dans une chaîne de recherche.

Syntaxe

object.**Length**

L'argument *object* représente toujours un objet **Match**.

Notes

Le code suivant montre comment utiliser la propriété **Length** :

```
Function RegExpTest(patrn, strng)
    Dim regEx, Match, Matches      ' Crée la variable.
    Set regEx = New RegExp         ' Crée l'expression régulièr
    regEx.Pattern = patrn         ' Définit les critères.
    regEx.IgnoreCase = True       ' Ignore la casse.
    regEx.Global = True          ' Définit le champ d'application
    Set Matches = regEx.Execute(strng) ' Lance la recherche.
    For Each Match in Matches      ' Itère la collection Matches.
        RetStr = RetStr & "Correspondance " & I & " trouvée à la posit
        RetStr = RetStr & Match.FirstIndex & ". La longueur de corres
        RetStr = RetStr & Match.Length
```

```
    RetStr = RetStr & " caractères." & vbCrLf
Next
RegExpTest = RetStr
End Function

MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

LoadPicture, fonction

[Référence du langage](#)
[Version 2](#)

Description

Renvoie un objet image. Disponible seulement sur les plates-formes 32 bits.

Syntaxe

LoadPicture(*picturename*)

L'argument *picturename* est une [expression de chaîne](#) représentant le nom du fichier d'image à charger.

Note

Les formats graphiques reconnus par **LoadPicture** sont les fichiers bitmap (.bmp), icônes (.ico), codés RLE (.rle), métafichiers (.wmf), métafichiers étendus (.emf), GIF (.gif) et JPEG (.jpg).

Log, fonction

[Voir aussi](#)

Description

Renvoie le logarithme népérien d'un nombre.

Syntaxe

Log(*number*)

L'argument *number* représente toute [expression numérique](#) valide supérieure à 0.

Notes

Le logarithme népérien est le logarithme de base *e*. La constante *e* est approximativement égale à 2,718282.

Vous pouvez calculer les logarithmes de base *n* d'un nombre *x* en divisant le logarithme népérien de *x* par le logarithme népérien de *n* de la façon suivante :

$$\text{Log}_n(x) = \text{Log}(x) / \text{Log}(n)$$

L'exemple suivant illustre une **Fonction** personnalisée qui calcule les logarithmes de base 10 :

```
Function Log10(X)
    Log10 = Log(X) / Log(10)
End Function
```



LTrim, RTrim et Trim, fonctions

[Référence du langage](#)
[Version 1](#)

[Voir aussi](#)

Description

Renvoient une copie d'une chaîne sans espaces à gauche (**LTrim**), sans espaces à droite (**RTrim**), ou sans espaces ni à gauche ni à droite (**Trim**).

Syntaxe

LTrim(*string*)

RTrim(*string*)

Trim(*string*)

L'argument *string* représente toute [expression de chaîne](#) valide. Si l'argument *string* contient [Null](#), la valeur **Null** est renvoyée.

Notes

L'exemple ci-dessous utilise les fonctions **LTrim**, **RTrim** et **Trim** pour supprimer les espaces à gauche, à droite et des deux côtés, respectivement :

```
Dim MyVar
```

```
MyVar = LTrim(" vbscript ") ' MyVar contient "vb
```

```
MyVar = RTrim(" vbscript ") ' MyVar contient " v
```

```
MyVar = Trim(" vbscript ") ' MyVar contient "vb
```

Match, objet

[Voir aussi](#)[Propriétés](#)

Description

Procure un accès aux propriétés d'une correspondance d'expression régulière.

Notes

Un objet **Match** peut être créé seulement par l'intermédiaire de la méthode **Execute** de l'objet **RegExp**. Ce dernier renvoie une [collection](#) d'objets **Match**. Toutes les propriétés des objets **Match** sont en lecture seule.

Lorsqu'une expression régulière est exécutée, il en résulte aucun, un ou plusieurs objets **Match**. Chaque objet **Match** procure un accès à la chaîne trouvée par l'expression régulière, fournit la longueur de la chaîne et un index indiquant où a été trouvée la correspondance.

Le code suivant montre comment utiliser un objet **Match** :

```
Function RegExpTest(patrn, strng)
    Dim regEx, Match, Matches      ' Crée la variable.
    Set regEx = New RegExp         ' Crée l'expression réguliè
    regEx.Pattern = patrn         ' Définit les critères.
    regEx.IgnoreCase = True       ' Ignore la casse.
    regEx.Global = True          ' Définit le champ d'applicatio
```

```
Set Matches = regEx.Execute(strng) ' Lance la recherche.
For Each Match in Matches ' Itère la collection Matches.
    RetStr = RetStr & "Correspondance " & I & " trouvée à la posit
    RetStr = RetStr & Match.FirstIndex & ". La valeur de la corresj
    RetStr = RetStr & Match.Value & "." & vbCRLF
Next
RegExpTest = RetStr
End Function

MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Matches, collection

[Voir aussi](#)[Propriétés](#)

Description

[Collection](#) d'objets **Match** sous forme d'expressions régulières.

Notes

Une collection **Matches** contient des objets individuels **Match** et peut être créée seulement à l'aide de la méthode **Execute** de l'objet **RegExp**. La collection **Matches** possède uniquement la propriété d'être en lecture seule, tout comme les objets individuels **Match**.

Lorsqu'une expression régulière est exécutée, il en résulte aucun, un ou plusieurs objets **Match**. Chaque objet **Match** procure un accès à la chaîne trouvée par l'expression régulière, fournit la longueur de la chaîne et un index indiquant où a été trouvée la correspondance.

Le code suivant montre la façon d'obtenir une collection **Matches** en utilisant une recherche d'expression régulière et la façon de parcourir la [collection](#) :

```
Function RegExpTest(patrn, strng)
    Dim regEx, Match, Matches      ' Crée la variable.
    Set regEx = New RegExp          ' Crée l'expression réguliè
    regEx.Pattern = patrn           ' Définit les critères.
    regEx.IgnoreCase = True         ' Définit le respect de la ca
```

```
regEx.Global = True           ' Définit le champ d'application
Set Matches = regEx.Execute(strng) ' Lance la recherche.
For Each Match in Matches     ' Itère la collection Matches.
    RetStr = RetStr & " Correspondance trouvée à la position "
    RetStr = RetStr & Match.FirstIndex & ". La valeur de la corres|
    RetStr = RetStr & Match.Value & "." & vbCrLf
Next
RegExpTest = RetStr
End Function

MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Mid, fonction

[Voir aussi](#)

Description

Renvoie un nombre spécifié de caractères d'une chaîne.

Syntaxe

Mid(*string*, *start*[, *length*])

La syntaxe de la fonction **Mid** comprend les éléments suivants :

élément	Description
<i>string</i>	Expression de chaîne à partir de laquelle les caractères sont renvoyés. Si l'argument <i>string</i> contient Null , la valeur Null est renvoyée.
<i>start</i>	Position du caractère dans l'argument <i>string</i> à partir duquel commence la partie à extraire. Si l'argument <i>start</i> est supérieur au nombre de caractères contenus dans l'argument <i>string</i> , la fonction Mid renvoie une chaîne de longueur nulle.
<i>length</i>	Nombre de caractères à renvoyer. Si cet argument est omis ou si le nombre de caractères dans le texte (y compris le caractère à l'argument <i>start</i>) est inférieur à ceux compris dans l'argument <i>length</i> , tous les caractères entre la position de l'argument <i>start</i> et la fin de la chaîne sont renvoyés.

Notes

Pour déterminer le nombre de caractères contenus dans l'argument *string*,

utilisez la fonction **Len**.

L'exemple ci-dessous utilise la fonction **Mid** pour renvoyer six caractères à partir du quatrième, dans une chaîne :

```
Dim MyVar
```

```
MyVar = Mid("VB Script est super!", 4, 6) ' MyVar
```

Remarque Une autre fonction (**MidB**) est disponible pour être utilisée avec les données d'octet contenues dans une chaîne. Au lieu de spécifier le nombre de caractères, les arguments spécifient le nombre d'octets.

Minute, fonction

[Voir aussi](#)

Description

Renvoie un nombre entier compris entre 0 et 59 inclus, représentant la minute de l'heure.

Syntaxe

Minute(*time*)

L'argument *time* est toute expression représentant une heure. Si l'argument *time* contient [Null](#), la valeur **Null** est renvoyée.

Notes

L'exemple ci-dessous utilise la fonction **Minute** pour renvoyer le chiffre des minutes de l'heure actuelle :

```
Dim MyVar  
MyVar = Minute(Now)
```

Divers, constantes

[Voir aussi](#)

Cette constante étant intégrée dans VBScript, il n'est pas nécessaire de la définir pour l'utiliser. Vous pouvez l'insérer n'importe où dans le code pour représenter les valeurs qui lui sont associées.

Constante	Valeur	Description
vbObjectError	-2147221504	Les numéros d'erreur définis par l'utilisateur do supérieurs à cette valeur, par exemple Err.Raise Number = vbObjectE

Mod, opérateur

[Voir aussi](#)

Description

Effectue la division de deux nombres et renvoie seulement le reste.

Syntaxe

result = *number1* **Mod** *number2*

La syntaxe de l'opérateur **Mod** comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>number1</i>	Toute expression numérique .
<i>number2</i>	Toute expression numérique.

Notes

L'opérateur modulo, ou reste, divise *number1* par *number2* (en arrondissant en entiers les nombres en virgules flottantes) et ne renvoie que le reste comme *result*. Par exemple, dans l'expression suivante, A (qui est l'élément *result*) est égal à 5.

A = 19 **Mod** 6.7

Si toute expression est [Null](#), *result* est aussi **Null**. Toute expression qui est [Empty](#) est traitée comme 0.



Month, fonction

[Voir aussi](#)

Description

Renvoie un nombre entier compris entre 1 et 12 inclus, représentant le mois de l'année.

Syntaxe

Month(*date*)

L'argument *date* est toute expression pouvant représenter une date. Si l'argument *date* contient [Null](#), la valeur **Null** est renvoyée.

Notes

L'exemple ci-dessous utilise la fonction **Month** pour renvoyer le mois en cours :

```
Dim MyVar
```

```
MyVar = Month(Now) ' MyVar contient le numéro correspondant  
' au mois en cours.
```

MonthName, fonction

[Voir aussi](#)

Description

Renvoie une chaîne indiquant le mois spécifié.

Syntaxe

MonthName(*month*[, *abbreviate*])

La syntaxe de la fonction **MonthName** comprend les éléments suivants :

élément	Description
<i>month</i>	Désignation numérique du mois. Par exemple, 1 pour janvier, 2 pour février, et ainsi de suite.
<i>abbreviate</i>	Facultatif. Valeur de type Boolean indiquant si le nom de mois est abrégé. Si cette valeur est omise, la valeur par défaut est False , ce qui signifie que le nom du mois n'est pas abrégé.

Notes

L'exemple ci-dessous utilise la fonction **MonthName** pour renvoyer le nom du mois abrégé pour une expression de date :

```
Dim MyVar  
MyVar = MonthName(10, True) ' MyVar  
contient "Oct".
```

MsgBox, constantes

[Voir aussi](#)

Les constantes ci-dessous sont utilisées avec la fonction **MsgBox** pour identifier les boutons et les icônes qui apparaissent sur une boîte de message, ainsi que le bouton par défaut. En outre, la modalité de **MsgBox** peut être spécifiée. Ces constantes étant intégrées dans VBScript, il n'est pas nécessaire des les définir pour les utiliser. Vous pouvez les insérer n'importe où dans le code pour représenter les valeurs qui leur sont associées.

Constante	Valeur	Description
vbOKOnly	0	Affiche uniquement le bouton OK .
vbOKCancel	1	Affiche les boutons OK et Annuler .
vbAbortRetryIgnore	2	Affiche les boutons Abandon , Réessayer et Ignorer .
vbYesNoCancel	3	Affiche les boutons Oui , Non et Annuler .
vbYesNo	4	Affiche les boutons Oui et Non .
vbRetryCancel	5	Affiche les boutons Réessayer et Annuler .
vbCritical	16	Affiche l'icône Message critique .
vbQuestion	32	Affiche l'icône Demande d'avertissement .
vbExclamation	48	Affiche l'icône Message d'avertissement .
vbInformation	64	Affiche l'icône Message d'information .

vbDefaultButton1	0	Le premier bouton est le bouton par défaut.
vbDefaultButton2	256	Le deuxième bouton est le bouton par défaut.
vbDefaultButton3	512	Le troisième bouton est le bouton par défaut.
vbDefaultButton4	768	Le quatrième bouton est le bouton par défaut.
vbApplicationModal	0	Boîte modale pour l'application. L'utilisateur doit répondre à la boîte de message avant de poursuivre le travail dans l'application courante.
vbSystemModal	4096	Boîte modale pour le système. Sur les systèmes Win16, toutes les applications sont suspendues jusqu'à ce que l'utilisateur réponde à la boîte de message. Sur les systèmes Win32, cette constante affiche une boîte de message modale pour l'application, laquelle reste toujours affichée quel que soit le programme que vous utilisez.

Les constantes suivantes sont utilisées avec la fonction **MsgBox** pour identifier le bouton sur lequel l'utilisateur a cliqué. Ces constantes sont disponibles uniquement lorsque votre projet contient une référence explicite à la bibliothèque de types contenant leurs définitions. Pour VBScript, vous devez déclarer ces constantes explicitement dans votre code.

Constante	Valeur	Description
vbOK	1	L'utilisateur a cliqué sur OK .
vbCancel	2	L'utilisateur a cliqué sur Annuler .
vbAbort	3	L'utilisateur a cliqué sur Abandon .

vbRetry	4	L'utilisateur a cliqué sur Réessayer.
vbIgnore	5	L'utilisateur a cliqué sur Ignorer.
vbYes	6	L'utilisateur a cliqué sur Oui.
vbNo	7	L'utilisateur a cliqué sur Non.

MsgBox, fonction

[Voir aussi](#)

Description

Affiche un message dans une boîte de dialogue, attend que l'utilisateur clique sur un bouton et renvoie une valeur indiquant le bouton choisi par l'utilisateur.

Syntaxe

MsgBox(*prompt*[, *buttons*][, *title*][, *helpfile*, *context*])

La syntaxe de la fonction **MsgBox** comprend les arguments suivants:

élément	Description
<i>prompt</i>	Expression de chaîne qui est affichée sous la forme d'un message dans la boîte de dialogue. La longueur maximum de l'argument <i>prompt</i> est environ 1024 caractères, selon la largeur des caractères utilisés. Si l'argument <i>prompt</i> se compose de plusieurs lignes, vous pouvez les séparer en utilisant un caractère de retour chariot (Chr(13)), un caractère de retour à la ligne (Chr(10)) ou une combinaison de ces deux caractères (Chr(13) & Chr(10)).
<i>buttons</i>	Expression numérique correspondant à la somme des valeurs spécifiant le nombre et le type de boutons à afficher, le style d'icône à utiliser, l'identité du bouton par défaut et la modalité du message. Pour les valeurs, reportez-vous à la section ci-après. Si elle est omise, la valeur par défaut de l'argument <i>buttons</i> est 0.
<i>title</i>	Expression de chaîne affichée dans la barre de titre de la boîte de dialogue. Si vous omettez l'argument <i>title</i> , le nom de l'application s'affiche dans la barre de titre.

<i>helpfile</i>	Expression de chaîne qui identifie le fichier d'aide à utiliser pour fournir l'aide contextuelle de la boîte de dialogue. Si l'argument <i>helpfile</i> est fourni, l'argument <i>context</i> doit aussi l'être. Non disponible sur les plateformes 16 bits
<i>context</i>	Expression numérique correspondant au numéro de contexte d'aide affecté par l'auteur de l'Aide à la rubrique d'aide appropriée. Si l'argument <i>context</i> est fourni, l'argument <i>helpfile</i> doit aussi l'être.

Valeurs

L'argument *buttons* peut prendre les valeurs suivantes:

Constante	Valeur	Description
vbOKOnly	0	Affiche uniquement le bouton OK .
vbOKCancel	1	Affiche les boutons OK et Annuler .
vbAbortRetryIgnore	2	Affiche les boutons Abandon , Réessayer et Ignorer .
vbYesNoCancel	3	Affiche les boutons Oui , Non et Annuler .
vbYesNo	4	Affiche les boutons Oui et Non .
vbRetryCancel	5	Affiche les boutons Réessayer et Annuler .
vbCritical	16	Affiche l'icône Message critique .
vbQuestion	32	Affiche l'icône Demande d'avertissement .
vbExclamation	48	Affiche l'icône Message d'avertissement .
vbInformation	64	Affiche l'icône Message d'information .
		Le premier bouton est le bouton

vbDefaultButton1	0	par défaut.
vbDefaultButton2	256	Le deuxième bouton est le bouton par défaut.
vbDefaultButton3	512	Le troisième bouton est le bouton par défaut.
vbDefaultButton4	768	Le quatrième bouton est le bouton par défaut.
vbApplicationModal	0	Application modale; l'utilisateur doit répondre au message avant de continuer à travailler dans l'application courante.
vbSystemModal	4096	Système modal; toutes les applications sont suspendues jusqu'à ce que l'utilisateur réponde au message.

Le premier groupe de valeurs (0 à 5) décrit le nombre et le type de boutons affichés dans la boîte de dialogue; le deuxième groupe (16, 32, 48, 64) décrit le style d'icône; le troisième groupe (0, 256, 512, 768) détermine le bouton par défaut; et le quatrième groupe (0, 4096) détermine la modalité du message. Au moment de l'ajout de nombres en vue de créer une valeur finale pour l'argument *buttons*, n'utilisez qu'un seul nombre de chaque groupe.

Valeurs renvoyées

La fonction **MsgBox** renvoie les valeurs suivantes:

Constante	Valeur	Bouton choisi
vbOK	1	OK
vbCancel	2	Annuler
vbAbort	3	Abandon
vbRetry	4	Réessayer
vbIgnore	5	Ignorer
vbYes	6	Oui
vbNo	7	Non

Notes

Quand les arguments *helpfile* et *context* sont tous deux fournis, l'utilisateur peut appuyer sur **F1** pour afficher la rubrique d'aide correspondant au contexte.

Si la boîte de dialogue affiche un bouton **Annuler**, le fait d'appuyer sur la touche **ÉCHAP** a le même effet que de cliquer sur **Annuler**. Si la boîte de dialogue contient un bouton **Aide**, l'aide contextuelle est disponible pour la boîte de dialogue. Toutefois, aucune valeur n'est renvoyée avant qu'un des autres boutons ne soit sélectionné.

Lorsque la fonction **MsgBox** est utilisée avec Microsoft Internet Explorer, le titre de toute boîte de dialogue présentée contient toujours l'indication "VBScript:" pour la différencier des boîtes de dialogue système standard.

L'exemple ci-dessous utilise la fonction **MsgBox** pour afficher une boîte de message et renvoyer une valeur indiquant sur quel bouton l'utilisateur a cliqué :

Dim MyVar

MyVar = **MsgBox** ("Bonjour!", 65,

"Exemple MsgBox") ' MyVar contient 1 ou 2,

' en fonction du bouton sur lequel

' l'utilisateur a cliqué.

opérateur

[Voir aussi](#)

Description

Multiplie deux nombres.

Syntaxe

result = *number1***number2*

La syntaxe de l'opérateur * comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>number1</i>	Toute expression numérique .
<i>number2</i>	Toute expression numérique.

Notes

Si une ou les deux expressions sont des expressions [Null](#), *result* est **Null**. Si une expression est [Empty](#), elle est traitée comme s'il s'agissait de 0.

opérateur

[Voir aussi](#)

Description

Effectue la différence entre deux nombres ou indique la valeur négative d'une expression numérique.

Syntaxe 1

result = *number1* - *nombre2*

Syntaxe 2

-*number*

La syntaxe de l'opérateur - comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>number</i>	Toute expression numérique .
<i>number1</i>	Toute expression numérique.
<i>number2</i>	Toute expression numérique.

Notes

Dans la syntaxe 1, l'opérateur - est l'opérateur de soustraction arithmétique utilisé pour trouver la différence entre deux nombres. Dans la syntaxe 2, l'opérateur - est utilisé comme opérateur de négation unaire pour indiquer la valeur négative d'une expression.

Si une ou les deux expressions sont des expressions [Null](#), *result* est **Null**. Si une expression est [Empty](#), elle est traitée comme s'il s'agissait de 0.

Not, opérateur

[Voir aussi](#)

Description

Effectue la négation logique d'une expression.

Syntaxe

result = **Not** *expression*

La syntaxe de l'opérateur **Not** comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>expression</i>	Toute expression .

Notes

Le tableau suivant illustre la manière dont l'élément *result* est déterminé :

Si <i>expression</i> est	alors <i>result</i> vaut
True	False
False	True
Null	Null

De plus, l'opérateur **Not** inverse les valeurs de bit de toute variable et définit le bit correspondant dans *result* d'après la table suivante :

Bit dans <i>expression</i>	Bit dans <i>result</i>
0	1
1	0

Now, fonction

[Voir aussi](#)

Description

Renvoie la date et l'heure en cours en fonction de leur paramétrage dans le système de votre ordinateur.

Syntaxe

Now

Notes

L'exemple ci-dessous utilise la fonction **Now** pour renvoyer la date et l'heure actuelles :

```
Dim MyVar  
MyVar = Now ' MyVar contient la date et l'heure actuelles.
```

[Voir aussi](#)

Description

Le mot clé **Nothing** dans VBScript est utilisé pour dissocier une variable objet de l'objet réel. Utilisez l'instruction **Set** pour affecter **Nothing** à une variable objet. Par exemple :

```
Set MyObject = Nothing
```

Plusieurs variables objets peuvent faire référence au même objet réel. Quand **Nothing** est affecté à une variable objet, celle-ci ne fait plus référence à un objet réel. Quand plusieurs variables objets font référence au même objet, les ressources mémoire et système associées à l'objet référencé par les variables ne sont libérées qu'une fois que toutes les variables ont été définies sur **Nothing** explicitement en utilisant **Set**, ou implicitement après que la dernière variable objet définie sur **Nothing** soit hors de [portée](#).

[Voir aussi](#)

Description

Le mot clé **Null** est utilisé pour indiquer qu'une variable ne contient aucune donnée valide. Cette valeur est différente de la valeur **Empty**.

Number, propriété

[Voir aussi](#)[Application](#)

Description

Renvoie ou définit une valeur numérique spécifiant une erreur. **Number** est la propriété par défaut de l'objet **Err**.

Syntaxe

object.**Number** [= *errornumber*]

La syntaxe de la propriété **Number** comprend les éléments suivants :

Élément	Description
<i>object</i>	Toujours l'objet Err .
<i>errornumber</i>	Entier représentant un numéro d'erreur VBScript ou une valeur d'erreur SCODE .

Notes

Lors du renvoi d'une erreur définie par l'utilisateur à partir d'un [objet Automation](#), définissez **Err.Number** en ajoutant le numéro que vous avez choisi comme code d'erreur à la constante **vbObjectError**.

Le code suivant illustre l'emploi de la propriété **Number** :

```
On Error Resume Next
Err.Raise vbObjectError + 1, "UnObject" ' Génère objet erreur ]
MsgBox ("Erreur N°" & CStr(Err.Number) & " " & Err.Descriptio
Err.Clear ' Efface l'erreur.
```

Oct, fonction

[Voir aussi](#)

Description

Renvoie une chaîne représentant la valeur octale d'un nombre.

Syntaxe

Oct(*number*)

L'argument *number* représente toute expression valide.

Notes

Si l'argument *number* n'est pas déjà un entier, il est arrondi au nombre entier le plus proche avant d'être évalué.

Si <i>number</i> est	Oct renvoie la valeur
Null	Null.
Empty	Zéro (0).
Tout autre nombre	11 caractères octaux maximum.

Vous pouvez représenter les nombres octaux directement en faisant précéder directement de &O les nombres compris dans la plage correcte. Par exemple, &O10 est la notation octale de la décimale 8.

L'exemple ci-dessous utilise la fonction **Oct** pour renvoyer la valeur octale d'un nombre :

Dim MyOct

MyOct = **Oct**(4) ' Renvoie 4.

MyOct = **Oct**(8) ' Renvoie 10.

MyOct = **Oct**(459) ' Renvoie 713.



Error, instruction

[Voir aussi](#)

Description

Active ou désactive la gestion des erreurs.

Syntaxe

On Error Resume Next
On Error GoTo 0

Notes

Si vous n'utilisez pas d'instruction **On Error Resume Next** à un autre endroit de votre code, toute [erreur d'exécution](#) se produisant pourra générer l'affichage d'un message d'erreur et l'arrêt de l'exécution du code. Toutefois, le comportement exact est déterminé par l'hôte exécutant le code. Parfois, il peut choisir de gérer certaines erreurs de manière différente. Dans certains cas, le débogueur de scripts pourra être appelé au niveau de l'erreur. Dans d'autres circonstances, aucune indication apparente d'erreur ne sera mentionnée car l'hôte ne notifie pas l'utilisateur. Là encore, tout est fonction de la manière dont l'hôte gère les erreurs qui se produisent.

Au sein d'une procédure particulière, une erreur n'est pas nécessairement fatale dès l'instant où la gestion des erreurs est activée le long de la pile d'appels. Si la gestion locale des erreurs n'est pas activée dans une procédure et qu'une erreur se produit, le contrôle est repassé à la pile d'appels jusqu'à ce qu'une procédure avec gestion des erreurs soit trouvée ; l'erreur est alors traitée à ce niveau. Si la gestion des erreurs n'est activée pour aucune des procédures de la pile d'appels, un message d'erreur est affiché à ce niveau et l'exécution prend fin ou l'hôte traite l'erreur comme il se doit.

L'instruction **On Error Resume Next** provoque la poursuite de l'exécution avec l'instruction succédant immédiatement à l'instruction ayant provoqué l'erreur d'exécution, ou avec l'instruction suivant immédiatement l'appel le plus récent de la procédure contenant l'instruction **On Error Resume Next**. Ceci permet de poursuivre l'exécution malgré une erreur d'exécution. Vous pouvez alors créer la routine de gestion d'erreur en ligne à l'intérieur de la procédure.

L'instruction **On Error Resume Next** devient inactive quand une autre procédure est appelée, de sorte que vous devez exécuter une instruction **On Error Resume Next** dans chaque routine appelée si vous souhaitez intégrer la gestion d'erreurs en ligne dans cette routine. Lorsque vous quittez une procédure, la fonction de gestion des erreurs retrouve l'état qu'elle présentait avant que vous n'accédiez à cette procédure.

Utilisez **On Error GoTo 0** pour désactiver la gestion des erreurs si vous l'avez activée précédemment avec **On Error Resume Next**.

L'exemple ci-dessous illustre l'utilisation de l'instruction **On Error Resume Next** :

On Error Resume Next

Err.Raise 6 ' Génère une erreur de dépassement.

MsgBox "Error # " & CStr(Err.Number) & " " & Err.Description

Err.Clear ' Efface l'erreur.

Priorité des opérateurs

[Voir aussi](#)

Description

Quand plusieurs opérations ont lieu dans une expression, chaque partie est évaluée et résolue dans un ordre prédéterminé. Cet ordre est connu sous le nom de priorité des opérateurs. Des parenthèses peuvent être utilisées pour annuler l'ordre de priorité et forcer l'évaluation de certaines parties d'une expression avant d'autres. Les opérations entre parenthèses sont toujours effectuées avant celles qui ne le sont pas. A l'intérieur des parenthèses, cependant, la priorité normale des opérateurs est conservée.

Quand des expressions contiennent des opérateurs de plusieurs catégories, les opérateurs arithmétiques sont évalués d'abord, puis les opérateurs de comparaison et enfin les opérateurs logiques. Les opérateurs de comparaison ont tous la même priorité : ils sont évalués de gauche à droite dans l'ordre de leur apparition. Les opérateurs arithmétiques et logiques sont évalués dans l'ordre de priorité suivant :

Arithmétique	Comparaison	Logique
Élévation à une puissance (^)	Égalité (=)	Not
Négation (-)	Inégalité (<>)	And
Multiplication et division (*, /)	Inférieur à (<)	Or
Division entière (\)	Supérieur à (>)	Xor
Modulo arithmétique (Mod)	Inférieur à ou égal à (<=)	Eqv
Addition et soustraction (+, -)	Supérieur à ou égal à (>=)	Imp
Concaténation de chaînes (&)	Is	&

Lorsqu'une multiplication et une division apparaissent dans la même expression, chaque opération est effectuée en fonction de son ordre d'apparition, en partant de la gauche de l'expression. Il en de même, lorsqu'une addition et une soustraction apparaissent dans une expression.

L'opérateur de concaténation de chaîne (&) n'est pas un opérateur arithmétique. Il n'est pas prioritaire par rapport aux opérateurs arithmétiques mais est traité avant tous les opérateurs de comparaison.

L'opérateur **Is** est un opérateur de comparaison entre des références à des objets. Il ne compare pas des objets ni leurs valeurs ; il contrôle uniquement si deux références se rapportent au même objet.

Option Explicit, instruction

Description

Force la déclaration explicite de toutes les variables dans un script.

Syntaxe

Option Explicit

Notes

Si elle est utilisée, l'instruction **Option Explicit** doit apparaître dans un script avant toute autre instruction.

Quand vous utilisez l'instruction **Option Explicit**, vous devez déclarer explicitement toutes les variables en utilisant les instructions **Dim**, **Private**, **Public** ou **ReDim**. Si vous essayez d'utiliser le nom d'une variable non déclarée, une erreur se produit.

Conseil Utilisez **Option Explicit** pour éviter de taper incorrectement le nom d'une variable existante ou de créer une confusion dans le code si la portée de la variable n'est pas claire.

L'exemple ci-dessous illustre l'utilisation de l'instruction **Option Explicit** :

Option Explicit ' Forcer la déclaration explicite d
Dim MyVar ' Déclarer une variable.
MyInt = 10 ' La variable non déclarée génère
MyVar = 10 ' La variable déclarée ne génère p

Or, opérateur

[Voir aussi](#)

Description

Effectue l'opération logique de disjonction sur deux expressions.

Syntaxe

result = *expression1* **Or** *expression2*

La syntaxe de l'opérateur **Or** comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>expression1</i>	Toute expression .
<i>expression2</i>	Toute expression.

Notes

Si une ou les deux expressions produisent la valeur **True**, *result* vaut **True**. Le tableau suivant illustre la manière dont *result* est déterminé :

Si <i>expression1</i> est	et <i>expression2</i> est	alors <i>result</i> vaut
True	True	True
True	False	True
True	Null	True
False	True	True
False	False	False
False	Null	Null
Null	True	True

Null	False	Null
Null	Null	Null

L'opérateur **Or** effectue aussi une [comparaison binaire](#) des bits de position identique dans deux [expressions numériques](#) et définit le bit correspondant dans *result* d'après la table suivante :

Si le bit <i>expression1</i> est	et le bit dans <i>expression2</i> est	alors <i>result</i> vaut
0	0	0
0	1	1
1	0	1
1	1	1

Pattern, propriété

[Voir aussi](#)[Application](#)

Description

Définit ou renvoie les critères de recherche de l'expression régulière.

Syntaxe

object.**Pattern** [= "*searchstring*"]

La syntaxe de la propriété **Pattern** comprend les éléments suivants :

Élément	Description
<i>object</i>	Requis. Il s'agit toujours d'une variable objet RegExp .
<i>searchstring</i>	Facultatif. Expression de chaîne régulière à laquelle s'applique la recherche. Elle inclut tout caractère de la chaîne régulière défini dans le tableau figurant dans la section Valeurs .

Valeurs

Les conventions d'écriture utilisées pour les expressions régulières autorisent les caractères spéciaux et les séquences de caractères. Le tableau suivant décrit et donne un exemple des caractères et des séquences utilisés.

Caractère	Description
\	Marque le caractère suivant comme caractère spécial ou littéral. Par exemple, "n" correspond au caractère "n". "\n" correspond à un caractère de nouvelle ligne. La

	séquence "\\" correspond à "\", tandis que \"(\" correspond à \"(\".
^	Correspond au début de la saisie.
\$	Correspond à la fin de la saisie.
*	Correspond au caractère précédent zéro fois ou plusieurs fois. Ainsi, "zo*" correspond à "z" ou à "zoo".
+	Correspond au caractère précédent une ou plusieurs fois. Ainsi, "zo+" correspond à "zoo", mais pas à "z".
?	Correspond au caractère précédent zéro ou une fois. Par exemple, "a?ve?" correspond à "ve" dans "lever".
.	Correspond à tout caractère unique, sauf le caractère de nouvelle ligne.
(modèle)	Correspond au <i>modèle</i> et mémorise la correspondance. La sous-chaîne correspondante peut être extraite de la collection Matches obtenue, à l'aide d'Item [0]...[n] . Pour trouver des correspondances avec des caractères entre parenthèses (), utilisez \"(\" ou \"\").
x y	Correspond soit à x soit à y. Par exemple, "z foot" correspond à "z" ou à "foot". "(z f)oo" correspond à "zoo" ou à "foot".
{n}	n est un nombre entier non négatif. Correspond exactement à n fois le caractère. Par exemple, "o{2}" ne correspond pas à "o" dans "Bob", mais aux deux premiers "o" dans "fooooot".
{n,}	n est un entier non négatif. Correspond à au moins n fois le caractère. Par exemple, "o{2,}" ne correspond pas à "o" dans "Bob", mais à tous les "o" dans "fooooot". "o{1,}" équivaut à "o+" et "o{0,}" équivaut à "o*".
{n,m}	m et n sont des entiers non négatifs. Correspond à au moins n et à au plus m fois le caractère. Par exemple, "o{1,3}" correspond aux trois premiers "o" dans "fooooot" et "o{0,1}" équivaut à "o?".
[xyz]	Jeu de caractères. Correspond à l'un des caractères indiqués. Par exemple, "[abc]" correspond à "a" dans "plat".
[^xyz]	Jeu de caractères négatif. Correspond à tout caractère non indiqué. Par exemple, "[^abc]" correspond à "p" dans "plat".
[a-z]	Série de caractères. Correspond à tout caractère dans la série spécifiée. Par exemple, "[a-z]" correspond à tout caractère alphabétique minuscule compris entre "a" et "z".
[^m-z]	Série de caractères négative. Correspond à tout caractère ne se trouvant pas dans la série spécifiée. Par exemple, "[^m-z]" correspond à tout caractère ne se trouvant pas entre "m" et "z".
\b	Correspond à une limite représentant un mot, autrement dit, à la position entre un mot et un espace. Par exemple, "er\b" correspond à "er" dans "lever", mais pas à "er" dans "verbe".
\B	Correspond à une limite ne représentant pas un mot. "en*t\B" correspond à "ent" dans "bien entendu".
\d	Correspond à un caractère représentant un chiffre. Équivaut à [0-9].
\D	Correspond à un caractère ne représentant pas un chiffre. Équivaut à [^0-9].
\f	Correspond à un caractère de saut de page.
\n	Correspond à un caractère de nouvelle ligne.
\r	Correspond à un caractère de retour chariot.
\s	Correspond à tout espace blanc, y compris l'espace, la tabulation, le saut de page, etc. Équivaut à "[\f\n\r\t\v]".
\S	Correspond à tout caractère d'espace non blanc. Équivaut à "[^\f\n\r\t\v]".
\t	Correspond à un caractère de tabulation.
\v	Correspond à un caractère de tabulation verticale.

<code>\w</code>	Correspond à tout caractère représentant un mot et incluant un trait de soulignement. Équivaut à "[A-Za-z0-9_]".
<code>\W</code>	Correspond à tout caractère ne représentant pas un mot. Équivaut à "[^A-Za-z0-9_]".
<code>\num</code>	Correspond à <i>num</i> , où <i>num</i> est un entier positif. Fait référence aux correspondances mémorisées. Par exemple, "(.)\1" correspond à deux caractères identiques consécutifs.
<code>\n</code>	Correspond à <i>n</i> , où <i>n</i> est une valeur d'échappement octale. Les valeurs d'échappement octales doivent comprendre 1, 2 ou 3 chiffres. Par exemple, "\11" et "\011" correspondent tous les deux à un caractère de tabulation. "\0011" équivaut à "\001" & "1". Les valeurs d'échappement octales ne doivent pas excéder 256. Si c'était le cas, seuls les deux premiers chiffres seraient pris en compte dans l'expression. Permet d'utiliser les codes ASCII dans des expressions régulières.
<code>\xn</code>	Correspond à <i>n</i> , où <i>n</i> est une valeur d'échappement hexadécimale. Les valeurs d'échappement hexadécimales doivent comprendre deux chiffres obligatoirement. Par exemple, "\x41" correspond à "A". "\x041" équivaut à "\x04" & "1". Permet d'utiliser les codes ASCII dans des expressions régulières.

Notes

Le code suivant montre comment utiliser la propriété **Pattern** :

Function RegExpTest(patrn, strng)

```

Dim regEx, Match, Matches      ' Crée la variable.
Set regEx = New RegExp          ' Crée une expression régulièr
regEx.Pattern = patrn         ' Définit les critères.
regEx.IgnoreCase = True        ' Ignore la casse.
regEx.Global = True            ' Définit une application global
Set Matches = regEx.Execute(strng) ' Lance la recherche.
For Each Match in Matches      ' Itère la collection Matches.
    RetStr = RetStr & "Correspondance trouvée à la position "
    RetStr = RetStr & Match.FirstIndex & ". La valeur de la corres
    RetStr = RetStr & Match.Value & "." & vbCrLf
Next
RegExpTest = RetStr
End Function

```

MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))

Private, instruction

[Voir aussi](#)

Description

Déclare les variables privées et alloue l'espace de stockage.
Déclare, dans un bloc **Class**, une variable privée.

Syntaxe

Private *varname*[(*subscripts*)][, *varname*[(*subscripts*)]] . . .

La syntaxe de l'instruction **Private** comprend les éléments suivants :

Élément	Description
<i>varname</i>	Nom de la variable ; respecte les conventions standard d'affectation de noms de variable.
<i>subscripts</i>	Dimensions d'une variable de tableau ; jusqu'à 60 dimensions peuvent être déclarées. L'argument <i>subscripts</i> utilise la syntaxe suivante : <i>upper</i> [, <i>upper</i>] . . . La limite inférieure d'un tableau a toujours la valeur zéro.

Notes

Les variables **Private** sont accessibles uniquement dans le script où elles ont été déclarées.

Une variable se référant à un objet doit être affectée à un objet existant à l'aide de l'instruction **Set** avant de pouvoir être utilisée. Jusqu'à ce qu'elle ait été affectée à un objet, la variable objet déclarée a la valeur **Empty**.

Vous pouvez également employer l'instruction **Private** avec des parenthèses vides pour déclarer un tableau dynamique. Après la déclaration d'un tableau dynamique, utilisez l'instruction **ReDim** dans une [procédure](#) pour définir le nombre de dimensions et d'éléments du tableau. Si vous tentez de redéclarer une dimension pour une variable de tableau dont la taille a été explicitement spécifiée dans une instruction **Private**, **Public** ou **Dim**, une erreur se produit.

Conseil Il convient de placer l'instruction **Private** au début de la procédure lorsque vous l'utilisez.

L'exemple ci-dessous illustre l'utilisation de l'instruction **Private** :

```
Private MyNumber      ' Variable Variant Private.
```

```
Private MyArray(9) ' Variable tableau Private.
```

```
      ' Déclaration Private multiples de variables Variant
```

```
Private MyNumber, MyVar, YourNumber
```

Property Get, instruction

[Voir aussi](#)

Description

Déclare, dans un bloc **Class**, le nom, les [arguments](#) et le code formant une procédure **Property** qui obtient (renvoie) la valeur d'une [propriété](#).

Syntaxe

```
[Public [Default] | Private] Property Get name [(arglist)]
    [statements]
    [[Set] name = expression]
Exit Property
    [statements]
    [[Set] name = expression]
End Property
```

La syntaxe de l'instruction **Property Get** comporte les éléments suivants :

Élément	Description
Public	Indique que la procédure Property Get est accessible par toutes les procédures contenues dans les scripts.
Default	Utilisé seulement avec le mot clé Public pour indiquer que la propriété définie dans la procédure Property Get est la propriété par défaut pour la classe .
Private	Indique que la procédure Property Get est accessible par toutes les procédures inscrites dans le bloc Class où elle est déclarée.
<i>name</i>	Nom de la procédure Property Get ; respecte les conventions standard d'affectation de noms de variable , excepté le fait que ce nom peut être le même que celui de la

	procédure Property Let ou Property Set dans un même bloc Class .
<i>arglist</i>	Liste des variables représentant les arguments qui sont transférés à la procédure Property Get lorsqu'elle est appelée. Les arguments sont séparés par des virgules. Le nom de chaque argument d'une procédure Property Get doit être le même que celui de l'argument correspondant dans une procédure Property Let (si elle existe).
<i>statements</i>	Tout groupe d'instructions à exécuter à l'intérieur d'une procédure Property Get .
Set	Mot clé utilisé lorsqu'un objet est affecté de la valeur renvoyée par une procédure Property Get .
<i>expression</i>	Valeur renvoyée par la procédure Property Get .

Notes

Si elles ne sont pas explicitement spécifiées comme **Public** ou **Private**, les procédures **Property Get** sont publiques par défaut. Elles sont accessibles par toutes les procédures contenues dans votre script. La valeur des variables locales dans une procédure **Property Get** n'est pas préservée entre les appels de la procédure.

Vous ne pouvez pas définir une procédure **Property Get** à l'intérieur d'une autre procédure (en d'autres termes, **Function** ou **Property Let**).

L'instruction **Exit Property** quitte immédiatement la procédure **Property Get** en cours. Le programme continue en passant à l'instruction qui suit celle à l'origine de l'appel de la procédure **Property Get**. Plusieurs instructions **Exit Property** peuvent apparaître dans une procédure **Property Get**.

Comme les procédures **Sub** et **Property Let**, une procédure **Property Get** est séparée et peut prendre en charge des arguments, exécuter une série d'instructions et modifier la valeur de ses arguments. En revanche, contrairement aux procédures **Sub** et **Property Let**, il est possible d'utiliser une procédure **Property Get** à droite d'une [expression](#) de la même manière que vous le faites avec le nom d'une instruction **Function** ou d'une

propriété lorsque vous souhaitez renvoyer la valeur d'une propriété.

Property Let, instruction

[Voir aussi](#)

Description

Déclare, dans un bloc **Class**, le nom, les [arguments](#) et le code formant une procédure **Property** qui affecte (définit) la valeur d'une [propriété](#).

Syntaxe

```
[Public | Private] Property Let name ([arglist,] value)  
    [statements]  
    [Exit Property]  
    [statements]  
End Property
```

La syntaxe de l'instruction **Property Let** comprend les éléments suivants :

Élément	Description
Public	Indique que la procédure Property Let est accessible par toutes les procédures contenues dans les scripts.
Private	Indique que la procédure Property Let est accessible seulement par les autres procédures du bloc Class où elle est déclarée.
<i>name</i>	Nom de la procédure Property Let ; respecte les conventions standard d'affectation de noms de variable , excepté le fait que ce nom peut être le même que celui de la procédure Property Get ou Property Set dans un même bloc Class .
<i>arglist</i>	Liste des variables représentant les arguments qui sont transférés à la procédure Property Let lorsqu'elle est appelée. Les arguments sont séparés par des virgules. Le nom de chaque argument d'une procédure Property Let doit être le même que celui de l'argument correspondant dans une procédure Property Get . De plus, la procédure Property Let devra toujours posséder un argument supplémentaire par rapport à la

	procédure Property Get correspondante. Cet argument représente la valeur qui est couramment affectée à la propriété.
<i>value</i>	Variable qui contient la valeur affectée à la propriété. Lorsque la procédure est appelée, cet argument apparaît à la droite de l'expression à l'origine de l'appel.
<i>statements</i>	Tout groupe d'instructions à exécuter à l'intérieur d'une procédure Property Let .

Remarque Chaque instruction **Property Let** doit définir au moins un argument pour la procédure qu'il traite. Cet argument, ou le dernier argument s'il en existe plusieurs, contient la valeur réelle qui sera affectée à la propriété lorsque la procédure définie par l'instruction **Property Let** sera appelée. Cet argument est appelé *value* dans la syntaxe précédente.

Notes

Si elles ne sont pas explicitement spécifiées comme **Public** ou **Private**, les procédures **Property Let** sont publiques par défaut. Elles sont accessibles par toutes les procédures contenues dans votre script. La valeur des variables locales dans une procédure **Property Let** n'est pas préservée entre les appels de la procédure.

Vous ne pouvez pas définir une procédure **Property Let** à l'intérieur d'une autre procédure (en d'autres termes, **Function** ou **Property Get**).

L'instruction **Exit Property** quitte immédiatement la procédure **Property Let** en cours. Le programme continue en passant à l'instruction qui suit celle à l'origine de l'appel de la procédure **Property Let**. Plusieurs instructions **Exit Property** peuvent apparaître dans une procédure **Property Let**.

Comme les procédures **Function** et **Property Get**, une procédure **Property Let** est séparée et peut prendre en charge des arguments, exécuter une série d'instructions et modifier la valeur de ses arguments. En revanche, contrairement aux procédures **Function** et **Property Get**, ces deux procédures renvoyant une

valeur, utilisez une procédure **Property Let** à gauche d'une [expression](#) d'affectation de propriété.

Property Set, instruction

[Voir aussi](#)

Description

Déclare, dans un bloc **Class**, le nom, les arguments et le code formant une procédure **Property** qui définit la référence à un objet.

Syntaxe

```
[Public | Private] Property Set name([arglist,] reference)  
    [statements]  
    [Exit Property]  
    [statements]  
End Property
```

La syntaxe de l'instruction **Property Set** comporte les éléments suivants :

Élément	Description
Public	Indique que la procédure Property Set est accessible par toutes les procédures contenues dans les scripts.
Private	Indique que la procédure Property Set est accessible par toutes les procédures inscrites dans le bloc Class où elle est déclarée.
<i>name</i>	Nom de la procédure Property Set ; respecte les conventions standard d'affectation de noms de variable , excepté le fait que ce nom peut être le même que celui de la procédure Property Get ou Property Let dans un même bloc Class .
<i>arglist</i>	Liste des variables représentant les arguments qui sont transférés à la procédure Property Set lorsqu'elle est appelée. Les arguments sont séparés par des virgules. De plus, la procédure Property Set devra toujours posséder un argument supplémentaire par rapport à la procédure Property Get correspondante. Cet argument représente l'objet qui est couramment affecté à la propriété .

<i>reference</i>	Variable contenant la référence à l'objet apparaissant à la droite de l'affectation de référence.
<i>statements</i>	Tout groupe d'instructions à exécuter à l'intérieur d'une procédure Property Set .

Remarque Chaque instruction **Property Set** doit définir au moins un argument pour la [procédure](#) qu'il traite. Cet argument, ou le dernier argument s'il en existe plusieurs, contient la référence de la propriété lorsque la procédure définie par l'instruction **Property Set** est appelée. Cet argument est appelé *référence* dans la syntaxe précédente.

Notes

Si elles ne sont pas explicitement spécifiées comme **Public** ou **Private**, les procédures **Property Set** sont publiques par défaut. Elles sont accessibles par toutes les procédures contenues dans votre script. La valeur des variables locales dans une procédure **Property Set** n'est pas préservée entre les appels de la procédure.

Vous ne pouvez pas définir une procédure **Property Set** à l'intérieur d'une autre procédure (en d'autres termes, **Function** ou **Property Let**).

L'instruction **Exit Property** quitte immédiatement la procédure **Property Set** en cours. Le programme continue en passant à l'instruction qui suit celle à l'origine de l'appel de la procédure **Property Set**. Plusieurs instructions **Exit Property** peuvent apparaître dans une procédure **Property Set**.

Comme les procédures **Function** et **Property Get**, une procédure **Property Set** est séparée et peut prendre en charge des arguments, exécuter une série d'instructions et modifier la valeur de ses arguments. En revanche, contrairement aux procédures **Function** et **Property Get**, les deux renvoyant une valeur, il est possible d'utiliser une procédure **Property Set** seulement à gauche de l'affectation de référence à l'objet (instruction **Set**).



Public, instruction

[Voir aussi](#)

Description

Déclare des variables publiques et affecte l'espace de stockage.
Déclare, dans un bloc **Class**, une variable privée.

Syntaxe

Public *varname*[(*subscripts*)][, *varname*[(*subscripts*)]] . . .

La syntaxe de l'instruction **Public** comprend les éléments suivants :

Élément	Description
<i>varname</i>	Nom de la variable ; respecte les conventions standard d'affectation de noms de variable.
<i>subscripts</i>	Dimensions d'une variable de tableau ; jusqu'à 60 dimensions peuvent être déclarées. Syntaxe de l'argument <i>subscripts</i> : <i>upper</i> [, <i>upper</i>] . . . La limite inférieure d'un tableau a toujours la valeur zéro.

Notes

Les variables avec l'instruction **Public** sont accessibles dans toutes les procédures de tous les scripts.

Une variable se référant à un objet doit être affectée à un objet existant à l'aide de l'instruction **Set** avant de pouvoir être utilisée. Jusqu'à ce qu'elle ait été affectée à un objet, la variable objet a la valeur **Empty**.

Vous pouvez également utiliser l'instruction **Public** avec des parenthèses vides pour déclarer un tableau dynamique. Après la déclaration d'un tableau dynamique, utilisez l'instruction **ReDim** dans une [procédure](#) pour définir le nombre de dimensions et d'éléments du tableau. Si vous tentez de déclarer à nouveau une dimension pour une variable de tableau dont la taille a été explicitement spécifiée dans une instruction **Private**, **Public** ou **Dim**, une erreur se produit.

L'exemple ci-dessous illustre l'utilisation de l'instruction **Public** :

```
Public MyNumber      ' Variable Variant Public.  
Public MyArray(9) ' Variable tableau Public.  
                ' Déclaration Public multiple de variables Variant.  
Public MyNumber, MyVar, YourNumber
```

Raise, méthode

[Voir aussi](#)[Application](#)

Description

Génère une [erreur d'exécution](#).

Syntaxe

object.**Raise**(*number*, *source*, *description*, *helpfile*, *helpcontext*)

La méthode **Raise** comprend les éléments suivants :

Élément	Description
<i>object</i>	Toujours l'objet Err .
<i>number</i>	Un sous-type entier Long qui identifie la nature de l'erreur. Les erreurs VBScript (tant définies par VBScript que par l'utilisateur) sont comprises dans la plage 0–65535.
<i>source</i>	Une expression de chaîne nommant l'objet ou l'application ayant initialement généré l'erreur. Lorsque vous définissez cette propriété pour un objet Automation, utilisez la forme <i>project.class</i> . Si rien n'est spécifié, l'identificateur de ressource du projet VBScript en cours est utilisé.
<i>description</i>	Une expression de chaîne décrivant l'erreur. Si elle n'est pas spécifiée, la valeur contenue dans l'argument <i>number</i> est examinée. Si elle peut être associée à un code d'erreur d'exécution VBScript, une chaîne fournie par VBScript est utilisée comme <i>description</i> . S'il n'existe aucune erreur VBScript correspondant à la valeur de l'argument <i>number</i> , un

	message d'erreur générique est utilisé.
<i>helpfile</i>	Le chemin d'accès complet du fichier d'aide contenant l'aide relative à cette erreur. Si cet argument n'est pas spécifié, VBScript utilise le lecteur, le chemin d'accès et le nom du fichier d'aide VBScript.
<i>helpcontext</i>	L'identificateur de contexte identifiant une rubrique contenue dans l'argument <i>helpfile</i> qui fournit l'aide correspondant à l'erreur. S'il est omis, l'identificateur de contexte du fichier d'aide VBScript pour l'erreur correspondant à la propriété <i>number</i> est utilisé, s'il existe.

Notes

Tous les arguments sont facultatifs à l'exception de *number*. Cependant, si vous utilisez la méthode **Raise** sans spécifier certains arguments et que les définitions des propriétés de l'objet **Err** contiennent des valeurs qui n'ont pas encore été supprimées, ces valeurs deviennent les valeurs de votre erreur.

Lorsque vous définissez la propriété *number* pour votre propre code d'erreur dans un [objet Automation](#), vous ajoutez le numéro de votre code d'erreur à la constante **vbObjectError**. Par exemple, pour générer le numéro d'erreur 1050, affectez **vbObjectError** + 1050 à la propriété *number*.

L'exemple ci-dessous illustre l'utilisation de la méthode **Raise** :

On Error Resume Next

Err.Raise 6 ' Génère une erreur de dépassement.

MsgBox ("Erreur N° " & CStr(**Err.Number**) & " " & **Err.Description**)

Err.Clear ' Efface l'erreur.

Randomize, instruction

[Voir aussi](#)

Description

Initialise le générateur de nombres aléatoires.

Syntaxe

Randomize [*number*]

L'argument *number* représente toute [expression numérique](#) valide.

Notes

L'instruction **Randomize** utilise l'argument *number* pour initialiser le générateur de nombres aléatoires de la fonction **Rnd**, en lui donnant une nouvelle [valeur initiale](#). Si vous omettez l'argument *number*, la valeur renvoyée par l'horloge système est utilisée comme nouvelle valeur initiale.

Si l'instruction **Randomize** n'est pas utilisée, la fonction **Rnd** (sans argument) utilise le même nombre comme valeur initiale la première fois qu'elle est appelée, et utilise ensuite le dernier nombre généré comme valeur initiale.

Remarque Pour répéter des séquences de nombres aléatoires, appelez l'instruction **Rnd** avec un argument négatif immédiatement avant d'utiliser **Randomize** avec un argument numérique. Utiliser **Randomize** avec la même valeur pour *number* ne répète pas la séquence précédente.

L'exemple ci-dessous illustre l'utilisation de l'instruction **Randomize** :

```
Dim MyValue, Response  
Randomize ' Initialise le générateur de
```

```
Do Until Response = vbNo  
    MyValue = Int((6 * Rnd) + 1) ' Génère une val  
    MsgBox MyValue  
    Response = MsgBox ("Recommencer? ", vbYesN  
Loop
```

ReDim, instruction

[Voir aussi](#)

Description

Déclare les variables de tableau dynamique et attribue ou réattribue l'espace de stockage au niveau de la procédure.

Syntaxe

ReDim [**Preserve**] *varname*(*subscripts*) [, *varname*(*subscripts*)] . . .

La syntaxe de l'argument **ReDim** comprend les éléments suivants :

Élément	Description
Preserve	Conserve les données d'un tableau existant quand vous changez la taille de la dernière dimension.
<i>varname</i>	Nom de la variable ; respecte les conventions standard d'affectation de nom à des variables .
<i>subscripts</i>	Dimensions d'une variable d'un tableau ; jusqu'à 60 dimensions multiples peuvent être déclarées. L'argument <i>subscripts</i> utilise la syntaxe suivante : <i>upper</i> [, <i>upper</i>] . . . La valeur inférieure d'un tableau est toujours zéro.

Notes

L'instruction **ReDim** est utilisée pour dimensionner ou redimensionner un tableau dynamique qui a déjà été déclaré formellement en utilisant une instruction **Private**, **Public** ou **Dim** avec des parenthèses vides (sans indice de dimension). Vous pouvez utiliser l'instruction **ReDim** de façon itérative

pour changer le nombre d'éléments et les dimensions d'un tableau.

Si vous utilisez le mot clé **Preserve**, vous ne pouvez modifier que la dernière dimension du tableau et, en aucun cas, le nombre de dimensions. Par exemple, si votre tableau ne comporte qu'une seule dimension, vous pouvez la modifier car c'est la dernière et seule dimension. Toutefois, si votre tableau comporte deux ou plusieurs dimensions, vous ne pouvez modifier que la dernière dimension, tout en conservant le contenu du tableau.

L'exemple suivant montre comment vous pouvez augmenter la taille de la dernière dimension d'un tableau dynamique, sans pour autant effacer les données contenues dans ce dernier.

ReDim X(10, 10, 10)

...

ReDim Preserve X(10, 10, 15)

Attention Si vous réduisez la taille originale d'un tableau, les données contenues dans les éléments éliminés sont perdues.

Quand les variables sont initialisées, une variable numérique est initialisée à 0 et une variable de chaîne est initialisée avec une chaîne de longueur nulle (""). Une variable faisant référence à un objet doit être affectée à un objet existant à l'aide de l'instruction **Set** avant de pouvoir être utilisée. Jusqu'à ce qu'elle soit affectée à un objet, la variable objet déclarée possède la valeur spéciale **Nothing**.

RegExp, objet

[Voir aussi](#)[Méthodes](#)[Propriétés](#)

Description

Permet uniquement la gestion des expressions régulières.

Notes

Le code suivant montre comment utiliser l'objet **RegExp** :

```
Function RegExpTest(patrn, strng)
    Dim regEx, Match, Matches      ' Crée la variable.
    Set regEx = New RegExp        ' Crée une expression régulièr
    regEx.Pattern = patrn          ' Définit les critères.
    regEx.IgnoreCase = True        ' Ignore la casse.
    regEx.Global = True           ' Définit le champ d'application
    Set Matches = regEx.Execute(strng) ' Lance la recherche.
    For Each Match in Matches      ' Itère la collection Matches.
        RetStr = RetStr & "Correspondance trouvée à la position "
        RetStr = RetStr & Match.FirstIndex & ". La valeur de la correspon
        RetStr = RetStr & Match.Value & "." & vbCRLF
    Next
    RegExpTest = RetStr
End Function

MsgBox(RegExpTest("is.", "IS1 is2 IS3 is4"))
```

Description

Inclut des remarques explicatives dans un programme.

Syntaxe

Rem *comment*

ou

' *comment*

L'argument *comment* représente le texte de tout commentaire que vous voulez inclure. Vous devez insérer un espace entre le mot clé **Rem** et l'argument *comment*.

Notes

Comme indiqué dans la section Syntaxe, vous pouvez remplacer le mot clé **Rem** par une apostrophe ('). Si le mot clé **Rem** suit les autres instructions sur une ligne, insérez le caractère deux-points pour les séparer. Toutefois, quand vous utilisez l'apostrophe, les deux-points ne sont pas requis après d'autres instructions.

L'exemple ci-dessous illustre l'utilisation de l'instruction **Rem** :

```
Dim MyStr1, MyStr2
```

```
MyStr1 = "Bonjour" : Rem Commentaire après une instruction, séparé par deux-points.
```

```
MyStr2 = "Aurevoir" ' Ceci est aussi un commentaire ; deux-points ne sont pas nécessaires.
```

Replace, fonction

[Voir aussi](#)

Description

Renvoie une chaîne dans laquelle une sous-chaîne donnée a été remplacée par une autre sous-chaîne le nombre de fois spécifié.

Syntaxe

Replace(*expression*, *find*, *replacewith*[, *start*[, *count*[, *compare*]]])

La syntaxe de la fonction **Replace** comprend les éléments suivants :

Élément	Description
<i>expression</i>	Expression de chaîne contenant une sous-chaîne à remplacer.
<i>find</i>	Sous-chaîne recherchée.
<i>replacewith</i>	Sous-chaîne de remplacement.
<i>start</i>	Facultatif. Position dans l'argument <i>expression</i> où la recherche de sous-chaîne doit commencer. Si elle est omise, la position 1 est prise par défaut. Elle doit être utilisée en conjonction avec <i>count</i> .
<i>count</i>	Facultatif. Nombre de remplacements de sous-chaîne à effectuer. Si cette valeur est omise, la valeur par défaut -1, qui signifie tous les remplacements possibles, est employée. Elle doit être utilisée en conjonction avec <i>start</i> .
<i>compare</i>	Facultatif. Valeur numérique indiquant le type de comparaison à utiliser lors de l'évaluation des sous-chaînes. Reportez-vous à la section Valeurs. Si elle est omise, la valeur par défaut est 0, comparaison binaire.

Valeurs

L'argument *compare* peut prendre les valeurs suivantes :

Constante	Valeur	Description
<code>vbBinaryCompare</code>	0	Effectue une comparaison binaire.
<code>vbTextCompare</code>	1	Effectue une comparaison texte.

Valeurs renvoyées

La valeur **Replace** renvoie les valeurs suivantes :

Si	La fonction Replace renvoie
<i>expression</i> a une longueur nulle	Une chaîne de longueur nulle ("").
<i>expression</i> a la valeur Null	Une erreur.
<i>find</i> a une longueur nulle	Une copie d' <i>expression</i> .
<i>replacewith</i> a une longueur nulle	Une copie d' <i>expression</i> , toutes les occurrences de <i>find</i> étant retirées.
<i>start</i> > Len (<i>expression</i>)	Une chaîne de longueur nulle.
<i>count</i> a une valeur de 0	Une copie d' <i>expression</i> .

Notes

La valeur renvoyée par la fonction **Replace** est une chaîne, une fois les substitutions effectuées, qui commence à la position spécifiée par l'argument *start* et se termine à la fin de la chaîne *expression*. Elle n'est pas une copie de la chaîne d'origine du début à la fin.

L'exemple ci-dessous utilise la fonction **Replace** pour renvoyer une chaîne :

Dim MyString

MyString = **Replace**("XXpXXPXXp", "p", "Y") ' ' au début de la (

MyString = **Replace**("XXpXXPXXp", "p", "Y", ' ' Renvoie "YXX

Replace, méthode

[Voir aussi](#)[Application](#)

Description

Remplace le texte trouvé dans une recherche d'expression régulière.

Syntaxe

object.**Replace**(*string1*, *string2*)

La syntaxe de la méthode **Replace** comprend les éléments suivants :

Élément	Description
<i>object</i>	Requis. Il s'agit toujours du nom d'un objet RegExp .
<i>string1</i>	Requis. <i>String1</i> est la chaîne de caractères dans laquelle le remplacement a été effectué.
<i>string2</i>	Requis. <i>String2</i> est la chaîne de caractères de remplacement.

Notes

Les critères réels du remplacement du texte en cours sont définis par la propriété **Pattern** de l'objet **RegExp**.

La méthode **Replace** renvoie une copie de la chaîne *string1* avec le texte de l'élément **RegExp.Pattern** remplacé par la chaîne *string2*. Si aucune correspondance n'est trouvée, une copie de la chaîne *string1* est renvoyée sans qu'elle soit modifiée.

Le code suivant montre comment utiliser la méthode **Replace** :

```
Function ReplaceTest(patrn, replStr)
    Dim regEx, str1                ' Crée des variables.
    str1 = "Le renard s'est jeté sur le chien."
    Set regEx = New RegExp        ' Crée l'expression ré
    regEx.Pattern = patrn        ' Définit les critères.
    regEx.IgnoreCase = True      ' Ignore la casse.
    ReplaceTest = regEx.Replace(str1, replStr) ' Effectue le remplac
End Function

MsgBox(ReplaceTest("renard", "chat"))    ' Remplace 'renard'
```

De plus, la méthode **Replace** est en mesure de remplacer des sous-expressions selon les critères définis. L'appel suivant de la fonction présentée dans l'exemple ci-dessus permute chaque paire de mots de la chaîne d'origine :

```
MsgBox(ReplaceText("(\\S+)(\\s+)(\\S+)", "$3$2$1"))    ' Permute l
```

RGB, fonction

Description

Renvoie un nombre entier représentant une valeur de couleur RVB.

Syntaxe

RGB(*red*, *green*, *blue*)

La fonction **RGB** comprend les éléments suivants :

Élément	Description
<i>red</i>	Nombre de 0 à 255 représentant la composante rouge de la couleur.
<i>green</i>	Nombre de 0 à 255 représentant la composante verte de la couleur.
<i>blue</i>	Nombre de 0 à 255 représentant la composante bleue de la couleur.

Notes

Les propriétés et les méthodes de l'application qui acceptent une spécification de couleur s'attendent à la recevoir sous forme de valeur de couleur RVB. Une valeur de couleur RVB spécifie l'intensité des composantes rouge, verte et bleue qui composent la couleur affichée.

L'octet de poids faible contient la valeur du rouge, l'octet central contient la valeur du vert et l'octet de poids fort contient la valeur du bleu.

Pour les applications qui utilisent un ordre d'octets inversé, la fonction ci-dessous fournit la même information avec les octets inversés :

Function RevRGB(rouge, vert, bleu)

 RevRGB= CLng(bleu + (vert * 256) + (rouge * 256))

End, fonction

Toute valeur d'argument supérieure à 255 est ramenée à 255.

Right, fonction

[Voir aussi](#)

Description

Renvoie un nombre spécifié de caractères à partir de la droite d'une chaîne.

Syntaxe

Right(*string*, *length*)

La syntaxe de la fonction **Right** comporte les arguments suivants :

Élément	Description
<i>string</i>	Expression de chaîne à partir de laquelle les caractères à l'extrême droite sont renvoyés. Si l'argument <i>string</i> contient la valeur Null , la valeur Null est renvoyée.
<i>length</i>	Expression numérique indiquant le nombre de caractères à renvoyer. Pour la valeur 0, une chaîne de longueur nulle est renvoyée. Pour une valeur supérieure ou égale au nombre de caractères contenus dans l'argument <i>string</i> , la chaîne entière est renvoyée.

Notes

Pour déterminer le nombre de caractères contenus dans l'argument *string*, utilisez la fonction **Len**.

L'exemple ci-dessous utilise la fonction **Right** pour renvoyer un nombre spécifié de caractères à partir de la droite d'une chaîne :

```
Dim AnyString, MyStr  
AnyString = "Bonjour"      ' Définit la chaîne.
```

MyStr = **Right**(AnyString, 1) ' Renvoie "r".
MyStr = **Right**(AnyString, 4) ' Renvoie "jour".
MyStr = **Right**(AnyString, 20) ' Renvoie "Bonjour".

Remarque Une autre fonction **RightB** est disponible pour les données de type octet contenues dans une chaîne. Au lieu de spécifier le nombre de caractères à renvoyer, l'argument *length* spécifie le nombre d'octets.

Rnd, fonction

[Voir aussi](#)

Description

Revoie un nombre aléatoire.

Syntaxe

Rnd[(*number*)]

L'argument *number* peut être toute [expression numérique](#) valide.

Notes

La fonction **Rnd** renvoie une valeur inférieure à 1 mais supérieure ou égale à 0. La valeur de *number* détermine de quelle façon **Rnd** génère un nombre aléatoire :

Si <i>number</i> est	Rnd génère
Inférieur à zéro	Le même nombre chaque fois, en utilisant l'argument <i>number</i> comme valeur initiale .
Supérieur à zéro	Le prochain nombre aléatoire de la séquence.
Égal à zéro	Le nombre le plus récemment généré.
Non fourni	Le prochain nombre aléatoire de la séquence.

Pour toute valeur initiale donnée, la même séquence de nombres est générée car chaque appel successif de la fonction **Rnd** utilise le nombre précédent comme valeur initiale pour le nombre

suivant de la séquence.

Avant d'appeler la fonction **Rnd**, utilisez l'instruction **Randomize** sans argument pour initialiser le générateur de nombres aléatoires avec une valeur initiale basée sur l'horloge système.

Pour produire des entiers aléatoires dans une plage donnée, utilisez la formule suivante :

$$\text{Int}(\textit{upperbound} - \textit{lowerbound} + 1) * \text{Rnd} + \textit{lowerbound}$$

Dans cette formule, *upperbound* est le plus grand nombre de la plage et *lowerbound* le plus petit nombre de la plage.

Remarque Pour répéter des séquences de nombres aléatoires, appelez la fonction **Rnd** avec un argument négatif immédiatement avant d'utiliser **Randomize** avec un argument numérique. L'utilisation de **Randomize** avec la même valeur pour *number* ne répète pas la séquence précédente.

Round, fonction

[Voir aussi](#)

Description

Renvoie un nombre arrondi à un nombre spécifié de positions décimales.

Syntaxe

Round(*expression*[, *numdecimalplaces*])

La syntaxe de la fonction **Round** comprend les éléments suivants :

Élément	Description
<i>expression</i>	Expression numérique arrondie.
<i>numdecimalplaces</i>	Facultatif. Nombre indiquant combien de positions à la droite de la virgule sont incluses dans le nombre arrondi. Si cette valeur est omise, les entiers sont arrondis par la fonction Round .

Notes

L'exemple ci-dessous utilise la fonction **Round** pour arrondir un nombre à deux décimales :

```
Dim MyVar, pi  
pi = 3.14159  
MyVar = Round(pi, 2) ' MyVar contient 3.14.
```

ScriptEngine, fonction

[Voir aussi](#)

Description

Renvoie une chaîne représentant le langage de script utilisé.

Syntaxe

ScriptEngine

Valeurs renvoyées

La fonction **ScriptEngine** renvoie l'une des chaînes suivantes :

Chaîne	Description
<i>VBScript</i>	Indique que Microsoft® Visual Basic® Scripting Edition est le moteur de script en cours.
<i>JScript</i>	Indique que Microsoft JScript® est le moteur de script en cours.
<i>VBA</i>	Indique que Microsoft Visual Basic pour Applications est le moteur de script en cours.

Notes

L'exemple ci-dessous utilise la fonction **ScriptEngine** pour renvoyer une chaîne décrivant le langage de script utilisé :

Function GetScriptEngineInfo

```
Dim s
```

```
s = "" ' Construit une chaîne contenant les informations nécess
```

```
s = ScriptEngine & " Version "  
s = s & ScriptEngineMajorVersion & "."  
s = s & ScriptEngineMinorVersion & "."  
s = s & ScriptEngineBuildVersion  
GetScriptEngineInfo = s ' Renvoie le résultat.  
End, fonction
```

ScriptEngineBuildVersion, [Référence du lan](#) [Vers](#) fonction

[Voir aussi](#)

Description

Renvoie le numéro de version du moteur de script employé.

Syntaxe

ScriptEngineBuildVersion

Notes

La valeur renvoyée correspond directement aux informations de version contenues dans la DLL du langage de script employé.

L'exemple ci-dessous utilise la fonction **ScriptEngineBuildVersion** pour renvoyer le numéro de génération du moteur de script :

Function GetScriptEngineInfo

```
Dim s
```

```
s = "" ' Construit une chaîne contenant les infor
```

```
s = ScriptEngine & " Version "
```

```
s = s & ScriptEngineMajorVersion & "."
```

```
s = s & ScriptEngineMinorVersion & "."
```

```
s = s & ScriptEngineBuildVersion
```

```
GetScriptEngineInfo = s ' Renvoie le résultat.
```

```
End, fonction
```

ScriptEngineMajorVersion, [Référence du l](#) [Ve](#) fonction

[Voir aussi](#)

Description

Renvoie le numéro de version principal du moteur de script employé.

Syntaxe

ScriptEngineMajorVersion

Notes

La valeur renvoyée correspond directement aux informations de version contenues dans la DLL du langage de script employé.

L'exemple ci-dessous utilise la fonction **ScriptEngineMajorVersion** pour renvoyer le numéro de version du moteur de script :

Function GetScriptEngineInfo

```
Dim s
s = "" ' Construit une chaîne contenant les infor
s = ScriptEngine & " Version "
s = s & ScriptEngineMajorVersion & "."
s = s & ScriptEngineMinorVersion & "."
s = s & ScriptEngineBuildVersion
GetScriptEngineInfo = s ' Renvoie le résultat.
End, fonction
```

ScriptEngineMinorVersion, [Référence du l](#) [Ve](#) fonction

[Voir aussi](#)

Description

Renvoie le numéro de version secondaire du moteur de script employé.

Syntaxe

ScriptEngineMinorVersion

Notes

La valeur renvoyée correspond directement aux informations de version contenues dans la DLL du langage de script employé.

L'exemple ci-dessous utilise la fonction **ScriptEngineMinorVersion** pour renvoyer le numéro de version secondaire du moteur de script :

Function GetScriptEngineInfo

```
Dim s
s = "" ' Construit une chaîne contenant les infor
s = ScriptEngine & " Version "
s = s & ScriptEngineMajorVersion & "."
s = s & ScriptEngineMinorVersion & "."
s = s & ScriptEngineBuildVersion
GetScriptEngineInfo = s ' Renvoie le résultat.
End, fonction
```

Second, fonction

[Voir aussi](#)

Description

Renvoie un nombre entier compris entre 0 et 59 inclus, représentant la seconde de la minute.

Syntaxe

Second(*time*)

L'argument *time* peut être toute expression pouvant représenter une heure. Si l'argument *time* contient [Null](#), la valeur **Null** est renvoyée.

Notes

L'exemple ci-dessous utilise la fonction **Second** pour renvoyer le chiffre des secondes en cours :

```
Dim MySec  
MySec = Second(Now) ' MySec contient le chiffre  
' des secondes.
```

Select Case, instruction

[Voir aussi](#)

Description

Exécute un groupe d'instructions parmi plusieurs, en fonction de la valeur d'une expression.

Syntaxe

```
Select Case testexpression  
    [Case expressionlist-n  
        [statement-n]] . . .  
    [Case Else expressionlist-n  
        [elstatements-n]]
```

End Select

La syntaxe de l'instruction **Select Case** comprend les éléments suivants :

Élément	Description
<i>testexpression</i>	Toute expression numérique ou de chaîne .
<i>expressionlist-n</i>	Requis si Case apparaît. Liste délimitée d'une ou de plusieurs expressions.
<i>statements-n</i>	Une ou plusieurs instructions exécutées si <i>testexpression</i> correspond à un élément de <i>expressionlist-n</i> .
<i>elstatements-n</i>	Une ou plusieurs instructions exécutées si <i>testexpression</i> ne correspond à aucune des clauses Case .

Notes

Si *testexpression* correspond à une expression **Case** *expressionlist*, les instructions suivant cette clause **Case** sont exécutées jusqu'à la clause **Case** suivante ou, pour la dernière clause, jusqu'à **End Select**. L'instruction suivant **End Select** prend ensuite le contrôle. Si *testexpression* correspond à une expression *expressionlist* dans plusieurs clauses **Case**, seules les instructions suivant la première correspondance sont exécutées.

La clause **Case Else** est utilisée pour indiquer les *elstatements* à exécuter si aucune correspondance n'était trouvée entre l'expression *testexpression* et une *expressionlist* dans toutes les autres sélections **Case**. Bien que ce ne soit pas obligatoire, il est judicieux d'insérer une instruction **Case Else** dans votre bloc **Select Case** pour gérer les valeurs *testexpression* imprévues. Si aucune *Case expressionlist* ne correspond à *testexpression* et s'il n'y a pas d'instruction **Case Else**, l'exécution continue à partir de l'instruction suivant **End Select**.

Les instructions **Select Case** peuvent être imbriquées. Chaque instruction **Select Case** imbriquée doit avoir une instruction **End Select** correspondante.

L'exemple ci-dessous illustre l'utilisation de l'instruction **Select Case** :

Dim Color, MyVar

Sub ChangeBackground (Color)

MyVar = lcase (Color)

Select Case MyVar

Case "rouge" document.bgColor = "rouge"

Case "vert" document.bgColor = "vert"

Case "bleu" document.bgColor = "bleu"

Case Else MsgBox "choisissez une autre

End Select

End Sub

Set, instruction

[Voir aussi](#)

Description

Affecte une référence d'objet à une [variable](#) ou à une [propriété](#) ou associe une référence de procédure à un événement.

Syntaxe 1

Set *objectvar* = { *objectexpression* | **New** *classname* | **Nothing** }

Syntaxe 2

Set *object.eventname* = **GetRef**(*procname*)

La syntaxe de l'instruction **Set** comprend les éléments suivants :

Élément	Description
<i>objectvar</i>	Requis. Nom de la variable ou propriété ; respecte les conventions standard d'affectation de noms à des variables.
<i>objectexpression</i>	Facultatif. Expression composée du nom d'un objet, d'une autre variable déclarée du même type d'objet ou d'une fonction ou méthode qui renvoie un objet appartenant au même type d'objet.
New	Mot clé utilisé pour créer une nouvelle instance de classe. Si <i>objectvar</i> contient une référence à un objet, celle-ci est ignorée lorsqu'une nouvelle référence est affectée. Le mot clé New est uniquement utilisé pour créer l'instance d'une classe .
<i>classname</i>	Facultatif. Nom de la classe en cours de création. Une classe et ses membres sont définis par l'instruction Class .
Nothing	Facultatif. Met fin à l'association de l'élément <i>objectvar</i> à un objet ou une classe spécifique. L'affectation à l'élément <i>objectvar</i> de la valeur Nothing libère toutes les ressources système et mémoire associées à l'objet précédemment référencé quand aucun autre élément n'y fait référence.

<i>object</i>	Requis. Nom de l'objet avec lequel un <i>événement</i> est associé.
<i>event</i>	Requis. Nom de l'événement auquel une fonction va être liée.
<i>procname</i>	Requis. Chaîne contenant le nom de la procédure Sub ou Function en cours d'association avec l'événement.

Notes

Pour être valide, *objectvar* doit être un type d'objet en cohérence avec l'objet qui lui est affecté.

Les instructions **Dim**, **Private**, **Public** ou **ReDim** ne déclarent qu'une variable faisant référence à un objet. Il n'est fait référence à aucun objet réel avant que vous n'utilisiez l'instruction **Set** pour affecter un objet spécifique.

En général, quand vous utilisez **Set** pour affecter une référence d'objet à une variable, aucune copie de l'objet n'est créée pour cette variable. à la place, une référence à l'objet est créée. Plusieurs variables objets peuvent faire référence au même objet. Dans la mesure où ces variables sont des références à l'objet (plutôt que des copies), tout changement apporté à l'objet est répercuté dans toutes les variables y faisant référence.

```
Function ShowFreeSpace(drvPath)
```

```
    Dim fso, d, s
```

```
    Set fso = CreateObject("Scripting.FileSystemObject")
```

```
    Set d = fso.GetDrive(fso.GetDriveName(drvPath))
```

```
    s = "Lecteur " & UCase(drvPath) & " - "
```

```
    s = s & d.VolumeName & "<BR>"
```

```
    s = s & "Espace disponible: " & FormatNumber(d.FreeSpace/10:
```

```
    s = s & " KOctets"
```

```
    ShowFreeSpace = s
```

```
End Function
```

L'utilisation du mot clé **New** vous permet de créer une instance de classe et de lui affecter une variable de référence à un objet. La variable à laquelle l'instance de la classe est affectée doit être déclarée au préalable avec l'instruction **Dim** ou une instruction équivalente.

Reportez-vous à la documentation relative à la fonction **GetRef** pour associer une procédure à un événement au moyen de l'instruction **Set**.

Sgn, fonction

[Voir aussi](#)

Description

Renvoie un entier indiquant le signe d'un nombre.

Syntaxe

Sgn(*number*)

L'argument *number* représente toute [expression numérique](#) valide.

Valeur renvoyées

La fonction **Sgn** comprend les valeurs renvoyées suivantes :

Si <i>number</i> est	Sgn renvoie
Supérieur à zéro	1
Égal à zéro	0
Inférieur à zéro	-1

Notes

Le signe de l'argument *number* détermine la valeur renvoyée de la fonction **Sgn**.

L'exemple ci-dessous utilise la fonction **Sgn** pour déterminer le signe d'un nombre :

```
Dim MyVar1, MyVar2, MyVar3, MySign  
MyVar1 = 12: MyVar2 = -2.4: MyVar3 = 0  
MySign = Sgn(MyVar1) ' Renvoie 1.
```

MySign = Sgn(MyVar2) ' Renvoie -1.

MySign = Sgn(MyVar3) ' Renvoie 0.

Sin, fonction

[Voir aussi](#)

Description

Renvoie le sinus d'un angle.

Syntaxe

Sin(*number*)

L'argument *number* représente toute [expression numérique](#) valide qui exprime un angle en radians.

Notes

La fonction **Sin** prend un angle et renvoie le rapport des deux côtés d'un triangle rectangle. Le rapport correspond à la longueur du côté opposé à l'angle, divisée par la longueur de l'hypoténuse. Le résultat est compris dans la plage -1 à 1.

Pour convertir les degrés en radians, multipliez les degrés par [pi](#)/180. Pour convertir les radians en degrés, multipliez les radians par 180/pi.

L'exemple ci-dessous utilise la fonction **Sin** pour renvoyer le sinus d'un angle :

```
Dim MyAngle, MyCosecant
```

```
MyAngle = 1.3           ' Définir l'angle en radians
```

```
MyCosecant = 1 / Sin(MyAngle) ' Calculer la cosécante
```

Source, propriété

[Voir aussi](#)[Application](#)

Description

Renvoie ou définit le nom de l'objet ou de l'application qui est à l'origine de l'erreur.

Syntaxe

object.**Source** [= *stringexpression*]

La syntaxe de la propriété **Source** comprend les éléments suivants :

Élément	Description
<i>object</i>	Toujours l'objet Err .
<i>stringexpression</i>	Expression de chaîne représentant l'application qui a généré l'erreur.

Notes

La propriété **Source** spécifie une expression de chaîne qui correspond, en général, au nom de [classe](#) ou à l'identificateur de ressource de l'objet qui a provoqué l'erreur. Utilisez la propriété **Source** pour fournir à vos utilisateurs les informations nécessaires lorsque votre code est incapable de gérer une erreur générée dans un objet en cours d'accès. Par exemple, si vous accédez à Microsoft Excel et qu'il génère une erreur *Division par zéro*, il affecte à **Err.Number** le code de cette erreur et à la propriété **Source** la chaîne "Excel.Application". Notez que si l'erreur est générée dans un autre objet appelé par Microsoft Excel, Excel intercepte l'erreur et affecte à **Err.Number** son propre code correspondant à *Division par zéro*. Il conserve, toutefois, l'autre objet **Err** (y compris la description de la propriété **Source**) tel que défini par l'objet ayant généré l'erreur.

La propriété **Source** contient toujours le nom de l'objet qui est à l'origine de l'erreur — votre code peut essayer de gérer l'erreur d'après la documentation d'erreur de l'objet auquel vous avez accédé. En cas d'échec de votre gestionnaire d'erreurs, vous pouvez utiliser les informations de l'objet **Err** pour décrire l'erreur à votre utilisateur, en utilisant la propriété **Source** et l'autre objet **Err** pour indiquer à l'utilisateur l'objet à l'origine de l'erreur, sa description de l'erreur, etc.

En cas de génération d'une erreur à partir du code, la propriété **Source** est l'identificateur de ressource de votre application.

Le code suivant illustre l'utilisation de la propriété **Source** :

```
On Error Resume Next
```

```
Err.Raise 6 ' Génère une erreur de dépassement.
```

```
MsgBox ("Erreur N° " & CStr(Err.Number) & " " & Err.Description)
```

```
Err.Clear ' Efface l'erreur.
```

Space, fonction

[Voir aussi](#)

Description

Renvoie une chaîne composée d'un nombre spécifié d'espaces.

Syntaxe

Space(*number*)

L'argument *number* représente le nombre d'espaces que vous voulez dans la chaîne.

Notes

L'exemple ci-dessous utilise la fonction **Space** pour renvoyer une chaîne consistant en un nombre spécifié d'espaces :

```
Dim MyString  
MyString = Space(10)           ' Renvoie une chaîne de 10 esp  
MyString = "Bonjour" & Space(10) & "Bonsoir" ' Insère 10 espa
```

Split, fonction

[Voir aussi](#)

Description

Renvoie un [tableau](#) à une dimension commençant par zéro contenant le nombre spécifié de sous-chaînes.

Syntaxe

Split(*expression*[, *delimiter*[, *count*[, *compare*]])

La syntaxe de la fonction **Split** comprend les éléments suivants :

Élément	Description
<i>expression</i>	Expression de chaîne contenant des sous-chaînes et des séparateurs. Si l'argument <i>expression</i> est une chaîne de longueur nulle, la fonction Split renvoie un tableau vide, c'est-à-dire un tableau ne comportant ni éléments, ni données.
<i>delimiter</i>	Facultatif. Caractère de chaîne utilisé pour identifier les limites de sous-chaîne. S'il est omis, le caractère espace (" ") est utilisé comme séparateur par défaut. Si l'argument <i>delimiter</i> est une chaîne de longueur nulle, un tableau à un élément contenant toute la chaîne <i>expression</i> est renvoyée.
<i>count</i>	Facultatif. Nombre de sous-chaînes à renvoyer ; -1 indique que toutes les sous-chaînes sont renvoyées.
<i>compare</i>	Facultatif. Valeur numérique indiquant le type de comparaison à utiliser lors de l'évaluation des sous-chaînes. Reportez-vous à la section Valeurs.

Valeurs

L'argument *compare* peut prendre les valeurs suivantes :

Constante	Valeur	Description
<code>vbBinaryCompare</code>	0	Effectue une comparaison binaire.
<code>vbTextCompare</code>	1	Effectue une comparaison texte.

Notes

L'exemple ci-dessous utilise la fonction **Split** pour renvoyer un tableau à partir d'une chaîne. La fonction effectue une comparaison textuelle du délimiteur et renvoie toutes les sous-chaînes.

```
Dim MyString, MyArray, Msg
MyString = "VBScriptXestSuper!"
MyArray = Split(MyString, "x", -1, 1)
' MyArray(0) contains "VBScript".
' MyArray(1) contient "est".
' MyArray(2) contient "super !".
Msg = MyArray(0) & " " & MyArray(1)
Msg = Msg & " " & MyArray(2)
MsgBox Msg
```

Description

Renvoie la racine carrée d'un nombre.

Syntaxe

Sqr(*number*)

L'argument *number* représente toute [expression numérique](#) supérieure ou égale à 0.

Notes

L'exemple ci-dessous utilise la fonction **Sqr** pour calculer la racine carrée d'un nombre :

```
Dim MySqr
MySqr = Sqr(4) ' Renvoie 2.
MySqr = Sqr(23) ' Renvoie 4.79583152331272.
MySqr = Sqr(0) ' Renvoie 0.
MySqr = Sqr(-4) ' Génère une erreur d'exécution.
```

StrComp, fonction

Description

Renvoie une valeur indiquant le résultat d'une [comparaison de chaîne](#).

Syntaxe

StrComp(*string1*, *string2*[, *compare*])

La syntaxe de la fonction **StrComp** comprend les éléments suivants :

Élément	Description
<i>string1</i>	Toute expression de chaîne valide.
<i>string2</i>	Toute expression de chaîne valide.
<i>compare</i>	Facultatif. Valeur numérique qui indique le type de comparaison à effectuer pour l'évaluation des chaînes. Si l'argument <i>compare</i> est omis, une comparaison binaire est effectuée. Les valeurs sont indiquées dans la section Valeurs.

Valeurs

L'argument *compare* peut prendre les valeurs suivantes :

Constante	Valeur	Description
vbBinaryCompare	0	Effectue une comparaison binaire.
vbTextCompare	1	Effectue une comparaison texte.

Valeurs renvoyées

La fonction **StrComp** renvoie les valeurs suivantes :

Si	La fonction StrComp renvoie
<i>string1</i> est inférieur à <i>string2</i>	-1
<i>string1</i> est égal à <i>string2</i>	0
<i>string1</i> est supérieur à <i>string2</i>	1
<i>string1</i> ou <i>string2</i> est Null	Null

Notes

L'exemple ci-dessous utilise la fonction **StrComp** pour renvoyer le résultat d'une comparaison de chaînes. Si le troisième argument vaut 1, la comparaison est textuelle. S'il vaut 0 ou s'il est absent, la comparaison est binaire.

```
Dim MyStr1, MyStr2, MyComp
```

```
MyStr1 = "ABCD": MyStr2 = "abcd"      ' Définir
```

```
MyComp = StrComp(MyStr1, MyStr2, 1)  ' Renv
```

```
MyComp = StrComp(MyStr1, MyStr2, 0)  ' Renv
```

```
MyComp = StrComp(MyStr2, MyStr1)     ' Renvo
```

Chaîne, constantes

[Voir aussi](#)

Ces constantes étant intégrées dans VBScript, il n'est pas nécessaire de les définir pour les utiliser. Vous pouvez les insérer n'importe où dans le code pour représenter les valeurs qui leur sont associées.

Constante	Valeur	Description
vbCr	Chr(13)	Retour chariot.
vbCrLf	Chr(13) et Chr(10)	Combinaison de retour chariot et de saut de ligne.
vbFormFeed	Chr(12)	Saut de page ; pas pratique dans Microsoft Windows.
vbLf	Chr(10)	Saut de ligne.
vbNewLine	Chr(13) et Chr(10) ou Chr(10)	Caractère de nouvelle ligne spécifique à la plate-forme ; adapté à celle-ci.
vbNullChar	Chr(0)	Caractère ayant la valeur 0.
vbNullString	Chaîne ayant la valeur 0.	Différent d'une chaîne de longueur nulle ("") ; utilisé pour l'appel de procédures externes.
vbTab	Chr(9)	Tabulation horizontale.
vbVerticalTab	Chr(11)	Tabulation verticale ; non utilisée dans Microsoft Windows.

String, fonction

[Voir aussi](#)

Description

Renvoie une chaîne constituée d'un caractère répété sur la longueur spécifiée

Syntaxe

String(*number*, *character*)

La syntaxe de la fonction **String** comprend les éléments suivants :

Élément	Description
<i>number</i>	Longueur de la chaîne renvoyée. Si l'argument <i>number</i> contient Null , la valeur Null est renvoyée.
<i>character</i>	Code de caractère spécifiant le caractère ou l' expression de chaîne dont le premier caractère est utilisé pour construire la chaîne renvoyée. Si l'argument <i>character</i> contient Null , la valeur Null est renvoyée.

Notes

Si vous spécifiez pour l'argument *character* un nombre supérieur à 255, la fonction **String** convertit le nombre en un code de caractère valide à l'aide de la formule :

character Mod 256

L'exemple ci-dessous utilise la fonction **String** pour renvoyer des chaînes

de caractères répétés de longueur spécifiée :

Dim MyString

MyString = **String**(5, "*") ' Renvoie "*****".

MyString = **String**(5, 42) ' Renvoie "*****".

MyString = **String**(10, "ABC") ' Renvoie "AAA/

StrReverse, fonction

Description

Renvoie une chaîne contenant des caractères dont l'ordre a été inversé par rapport à une chaîne donnée.

Syntaxe

StrReverse(*string1*)

L'argument *string1* est la chaîne pour laquelle l'inversion des caractères a été demandée. Si l'argument *string1* est une chaîne de longueur nulle (""), alors la fonction renvoie une chaîne de longueur nulle. Si l'argument *string1* est **Null**, une erreur se produit.

Notes

L'exemple ci-dessous utilise la fonction **StrReverse** pour renvoyer une chaîne inversée :

```
Dim MyStr  
MyStr = StrReverse("VBScript") ' MyStr contient "tpircSBV".
```

Sub, instruction

[Voir aussi](#)

Description

Déclare le nom, les arguments et le code qui forment le corps d'une procédure **Sub**.

Syntaxe

[**Public** [**Default**] **Private**] **Sub** *name* [(*arglist*)]

[*statements*]

[**Exit Sub**]

[*statements*]

End Sub

La syntaxe de l'instruction **Sub** comprend les éléments suivants :

Élément	Description
Public	Indique que la procédure Sub est accessible à toutes les autres procédures dans tous les scripts.
Default	Utilisé seulement avec le mot clé Public dans un bloc Class pour indiquer que la procédure Sub est la méthode par défaut de la classe . Une erreur se produit si plusieurs procédures Default sont spécifiées dans une classe.
Private	Indique que la procédure Sub est accessible uniquement aux autres procédures du script dans lequel elle est déclarée.
<i>name</i>	Nom de la procédure Sub ; respecte les conventions standard d'affectation de nom à des variables .
<i>arglist</i>	Liste de variables représentant les arguments passés à la procédure Sub quand elle est appelée. Les variables multiples sont séparées par des virgules.
<i>statements</i>	Tout groupe d'instructions à exécuter dans le corps de la procédure Sub .

L'argument *arglist* comprend la syntaxe et les éléments suivants :

[**ByVal** | **ByRef**] *varname*[()]

Élément	Description
ByVal	Indique que l' argument est transmis par valeur .
ByRef	Indique que l'argument est transmis par référence .
<i>varname</i>	Nom de la variable représentant l'argument ; respecte les conventions standard d'affectation de nom à des variables.

Notes

En l'absence des mots clés **Public** ou **Private**, les procédures **Sub** sont publiques par défaut. En d'autres termes, elles sont visibles pour toutes les autres procédures de votre script. La valeur des variables locales dans une procédure **Sub** n'est pas conservée entre les appels à la [procédure](#).

Vous ne pouvez définir une procédure **Sub** à l'intérieur d'une autre procédure **Function** ou **Property Get**.

L'instruction **Exit Sub** provoque la sortie immédiate d'une procédure **Sub**. L'exécution du programme se poursuit avec l'instruction suivant l'instruction ayant appelé la procédure **Sub**. Une procédure **Sub** peut comporter un nombre indéterminé d'instructions **Exit Sub** apparaissant en n'importe quel point.

À l'instar d'une procédure **Function**, une procédure **Sub** est une procédure distincte qui peut prendre des arguments, exécuter une série d'instructions et changer la valeur de ses arguments. Toutefois, contrairement à la procédure **Function** procédure qui renvoie une valeur, une procédure **Sub** ne peut pas être utilisée dans une [expression](#).

Vous appelez une procédure **Sub** en utilisant le nom de la procédure, suivi de la liste des arguments. Pour plus

d'informations sur la manière d'appeler les procédures **Sub**, consultez l'instruction **Call**.

Attention Les procédures **Sub** peuvent être récursives, autrement dit, elles peuvent s'appeler elles-mêmes pour effectuer une tâche donnée. Toutefois, la récursivité peut amener au dépassement de la capacité de la pile.

Les variables utilisées dans les procédures **Sub** se divisent en deux catégories : celles qui sont déclarées explicitement dans la procédure et celles qui ne le sont pas. Les premières (déclarées en utilisant **Dim** ou l'équivalent) sont toujours locales pour la procédure. Les variables utilisées sans avoir été déclarées explicitement dans une procédure sont toujours locales, à moins qu'elles n'aient été explicitement déclarées à un niveau supérieur hors de la procédure.

Attention Une procédure peut utiliser une variable qui n'est pas déclarée explicitement dans la procédure, mais un conflit peut se produire si vous avez défini un élément au [niveau du script](#) qui porte le même nom que celui de la variable. Si votre procédure fait référence à une variable non déclarée portant le même nom qu'une autre procédure, [constante](#) ou variable, il est supposé que votre procédure fait référence à ce nom au niveau du script. Pour éviter ce genre de conflit, utilisez une instruction **Option Explicit** pour forcer la déclaration explicite des variables.

Tan, fonction

[Voir aussi](#)

Description

Renvoie la tangente d'un angle.

Syntaxe

Tan(*number*)

L'argument *number* représente toute [expression numérique](#) valide qui exprime un angle en radians.

Notes

La fonction **Tan** prend un angle et renvoie le rapport des deux côtés d'un triangle rectangle. Le rapport correspond à la longueur du côté opposé à l'angle, divisée par la longueur du côté adjacent à l'angle.

Pour convertir des degrés en radians, multipliez les degrés par [pi](#)/180. Pour convertir des radians en degrés, multipliez les radians par 180/[pi](#).

L'exemple ci-dessous utilise la fonction **Tan** pour renvoyer la tangente d'un angle :

```
Dim MyAngle, MyCotangent
MyAngle = 1.3           ' Définir l'angle en rad
MyCotangent = 1 / Tan(MyAngle) ' Calculer la c
```

Terminate, événement

[Voir aussi](#)

[Application](#)

Description

Se produit lorsqu'une instance de la [classe](#) associée prend fin.

Syntaxe

```
Private Sub Class_Terminate()
```

```
    instructions
```

```
End Sub
```

La partie *instructions* consiste en zéro instruction de code ou plus à exécuter lors de l'initialisation de la classe.

Notes

L'exemple ci-dessous illustre l'utilisation de l'événement

Terminate :

```
Class TestClass
```

```
    Private Sub Class_Initialize ' Configuration de l'événement Initiali
```

```
        MsgBox("TestClass démarré")
```

```
    End Sub
```

```
    Private Sub Class_Terminate ' Configuration de l'événement
```

```
        MsgBox("TestClass terminé")
```

```
    End Sub
```

End Class

Set X = New TestClass ' Crée une instance de TestClass.

Set X = Nothing ' Détruire l'instance.

Test, méthode

[Voir aussi](#)[Application](#)

Description

Lance une recherche d'expression régulière dans une chaîne spécifiée et renvoie une valeur **booléenne** qui indique si une correspondance selon les critères spécifiés a été trouvée.

Syntaxe

object.**Test**(*string*)

La syntaxe de la méthode **Execute** comprend les éléments suivants :

Élément	Description
<i>object</i>	Requis. Il s'agit toujours du nom d'un objet RegExp .
<i>string</i>	Requis. Chaîne de caractères à laquelle l'expression régulière est appliquée.

Notes

Les critères réels d'une recherche d'expression régulière sont définis par la propriété **Pattern** de l'objet **RegExp**. La propriété **RegExp.Global** n'a aucun effet sur la méthode **Test**.

La méthode **Test** renvoie la valeur **True** si une correspondance selon les critères spécifiés a été trouvée. Dans le cas contraire, la valeur **False** est renvoyée.

Le code suivant montre comment utiliser la méthode **Test** :

```
Function RegExpTest(patrn, strng)
    Dim regEx, retVal          ' Crée la variable.
    Set regEx = New RegExp     ' Crée l'expression réguliè
    regEx.Pattern = patrn     ' Définit les critères.
    regEx.IgnoreCase = False  ' Définit le respect de la casse.
    retVal = regEx.Test(strng) ' Lance le test de recherche.
    If retVal Then
        RegExpTest = "Une ou plusieurs correspondances ont été trouv
    Else
        RegExpTest = "Aucune correspondance n'a été trouvée."
    End If
End Function

MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Time, fonction

[Voir aussi](#)

Description

Renvoie un **VARIANT** de sous-type **DATE** indiquant l'heure système en cours.

Syntaxe

Time

Notes

L'exemple ci-dessous utilise la fonction **Time** pour renvoyer l'heure système actuelle :

```
Dim MyTime
```

```
MyTime = Time ' Renvoie l'heure système actuelle.
```

Timer, fonction

[Voir aussi](#)

Description

Renvoie le nombre de secondes qui se sont écoulées depuis 00:00 (minuit).

Syntaxe

Timer

Notes

L'exemple suivant utilise la fonction **Timer** pour déterminer le temps nécessaire pour l'itération d'une boucle **For...Next** un nombre *N* spécifié :

```
Function TimeIt(N)
  Dim StartTime, EndTime
  StartTime = Timer
  For I = 1 To N
  Next
  EndTime = Timer
  TimeIt = EndTime - StartTime
End Function
```

TimeSerial, fonction

[Voir aussi](#)

Description

Renvoie un **VARIANT** de sous-type **DATE** contenant l'heure correspondant à des éléments spécifiques d'heure, de minute et de seconde.

Syntaxe

TimeSerial(*hour, minute, second*)

La syntaxe de la fonction **TimeSerial** comprend les éléments suivants :

Élément	Description
<i>hour</i>	Nombre entre 0 (12:00) et 23 (11:00) inclus ou expression numérique .
<i>minute</i>	Toute expression numérique.
<i>second</i>	Toute expression numérique.

Notes

Pour spécifier une heure telle que 11:59:59, la plage des nombres pour chaque argument **TimeSerial** doit se situer dans la plage normalement acceptée pour l'unité ; autrement dit, 0–23 pour les heures et 0–59 pour les minutes et les secondes. Toutefois, vous pouvez aussi spécifier des heures relatives pour chaque argument en utilisant toute expression numérique qui représente un certain nombre d'heures, de minutes ou de secondes avant ou après une heure donnée.

L'exemple suivant utilise des expressions à la place de nombres absolus d'heure. La fonction **TimeSerial** renvoie une heure correspondant à 15 avant (-15) six heures avant midi (12 - 6) ou 5:45:00.

Dim MyTime1

MyTime1 = **TimeSerial(12 - 6, -15, 0)** ' Renvoie 5:

Lorsqu'un argument dépasse la plage normalement acceptée, il s'incrémente sur l'unité supérieure suivante. Si, par exemple, vous spécifiez 75 minutes, elles sont évaluées en 1 heure et 15 minutes. Toutefois, si un seul argument n'est pas compris dans la plage -32,768 à 32,767, ou si l'heure spécifiée par les trois arguments, soit directement soit par expression, génère une date non comprise dans la plage des dates acceptables, une erreur se produit.

TimeValue, fonction

[Voir aussi](#)

Description

Renvoie un **Variant** de sous-type **Date** contenant l'heure.

Syntaxe

TimeValue(*time*)

L'argument *time* est habituellement une [expression de chaîne](#) représentant un heure de 0:00:00 (12:00:00) à 23:59:59 (11:59:59) inclus. Toutefois, l'argument *time* peut aussi être toute expression représentant une heure comprise dans cette plage. Si l'argument *time* contient **Null**, la valeur **Null** est renvoyée.

Notes

Vous pouvez entrer des heures valides en utilisant une horloge au format 12 ou 24 heures. Par exemple, "2:24" et "14:24" sont tous deux des valeurs de l'argument *time*. si l'argument *time* contient des informations de date, la fonction **TimeValue** ne les renvoie pas. Toutefois, si l'argument *time* inclut des informations de date incorrectes, une erreur se produit.

L'exemple ci-dessous utilise la fonction **TimeValue** pour convertir une chaîne en heure. Vous pouvez également utiliser [littéraux de dates](#) pour affecter directement une heure à une variable **Variant**, par exemple MyTime = #16:35:17#.

Dim MyTime

MyTime = **TimeValue**("16:35:17") ' MyTime cor

[Voir aussi](#)

Description

Le mot clé **True** a une valeur égale à -1.

TypeName, fonction

[Voir aussi](#)

Description

Renvoie une chaîne qui fournit des informations de sous-type **Variant** sur une variable.

Syntaxe

TypeName(*varname*)

L'argument *varname* représente toute variable.

Valeurs renvoyées

La fonction **TypeName** renvoient les valeurs suivantes :

Valeur	Description
Byte	Valeur de type octet
Integer	Valeur de type entier
Long	Valeur de type entier long
Single	Valeur en virgule flottante à simple précision
Double	Valeur en virgule flottante à double précision
Currency	Valeur de type monétaire
Decimal	Valeur de type décimal
Date	Valeur de date ou d'heure
String	Valeur de chaîne de caractères
Boolean	Valeur de type booléen ; True ou False
Empty	Non initialisée

Null	Aucune donnée valide
<object; type>	Nom de type réel d'un objet
Object	Objet générique
Unknown	Type d'objet inconnu
Nothing	Variable d'objet ne se référant encore à aucune instance d'objet
Error	Erreur

Notes

L'exemple ci-dessous utilise la fonction **TypeName** pour renvoyer des informations sur une variable :

```
Dim ArrayVar(4), MyType
NullVar = Null           ' Affecter la valeur Null.

MyType = TypeName("VBScript") ' Renvoie "String".
MyType = TypeName(4)         ' Renvoie "Integer".
MyType = TypeName(37.50)     ' Renvoie "Double".
MyType = TypeName(NullVar)   ' Renvoie "Null".
MyType = TypeName(ArrayVar)  ' Renvoie "Variant()".
```

UBound, fonction

[Voir aussi](#)

Description

Renvoie le plus grand indice disponible pour la dimension indiquée d'un [tableau](#).

Syntaxe

UBound(*arrayname*[, *dimension*])

La syntaxe de la fonction **UBound** comprend les éléments suivants :

Élément	Description
<i>arrayname</i>	Nom de la variable tableau ; respectez les conventions standard d'affectation de noms à des variables .
<i>dimension</i>	Nombre entier indiquant pour quelle dimension la limite supérieure est renvoyée. Utilisez 1 pour la première dimension, 2 pour la deuxième, etc. Si l'élément <i>dimension</i> est omis, 1 est supposé.

Notes

La fonction **UBound** est utilisée avec la fonction **LBound** pour déterminer la taille d'un tableau. Utilisez la fonction **LBound** pour trouver la limite inférieure d'une dimension de tableau.

La limite inférieure de toute dimension est toujours 0. En conséquence, la fonction **UBound** renvoie les valeurs répertoriées dans la table ci-dessous pour un tableau avec les dimensions suivantes :

Dim A(100,3,4)

Instruction	Valeur renvoyée
-------------	-----------------

UBound(A, 1)	100
UBound(A, 2)	3
UBound(A, 3)	4

UCase, fonction

[Voir aussi](#)

Description

Renvoie une chaîne qui a été convertie en majuscules.

Syntaxe

UCase(*string*)

L'argument *string* représente toute [expression de chaîne](#) valide. Si l'argument *string* contient [Null](#), la valeur **Null** est renvoyée.

Notes

Seules les lettres minuscules sont converties en majuscules ; toutes les majuscules et les caractères autres que des lettres demeurent inchangés.

L'exemple ci-dessous utilise la fonction **UCase** pour renvoyer une version en majuscules d'une chaîne :

```
Dim MyWord
```

```
MyWord = UCase("Bonjour") ' Renvoie "Bonjou"
```

Value, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie la valeur ou le texte d'une correspondance trouvée dans une chaîne de recherche.

Syntaxe

object.**Value**

L'argument *object* représente toujours un objet **Match**.

Notes

Le code suivant montre comment utiliser la propriété **Value** :

```
Function RegExpTest(patrn, strng)
    Dim regEx, Match, Matches      ' Crée la variable.
    Set regEx = New RegExp          ' Crée l'expression réguliè
    regEx.Pattern = patrn          ' Définit les critères.
    regEx.IgnoreCase = True        ' Ignore la casse.
    regEx.Global = True            ' Définit le champ d'applicatio
    Set Matches = regEx.Execute(strng) ' Lance la recherche.
    For Each Match in Matches      ' Itère la collection Matches.
        RetStr = RetStr & "Correspondance " & I & " trouvée à la posit
        RetStr = RetStr & Match.FirstIndex & ". La valeur de la corres
```

```
RetStr = RetStr & Match.Value & "." & vbCRLF
Next
RegExpTest = RetStr
End Function
```

```
MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

VarType, constantes

[Voir aussi](#)

Ces constantes sont disponibles uniquement lorsque votre projet contient une référence explicite à la [bibliothèque de types](#) contenant leurs définitions. Pour VBScript, vous devez déclarer ces constantes explicitement dans votre code.

Constante	Valeur	Description
vbEmpty	0	Non initialisé (par défaut)
vbNull	1	Ne contient pas de données valides
vbInteger	2	Sous-type Integer
vbLong	3	Sous-type Long
vbSingle	4	Sous-type Single
vbDouble	5	Sous-type Double
vbCurrency	6	Sous-type Currency
vbDate	7	Sous-type Date
vbString	8	Sous-type String
vbObject	9	Objet
vbError	10	Sous-type Error
vbBoolean	11	Sous-type Boolean
vbVariant	12	Variant (utilisé uniquement pour les tableaux de données de type Variant)
vbDataObject	13	Objet d'accès aux données
vbDecimal	14	Sous-type Decimal
vbByte	17	Sous-type Byte
vbArray	8192	Tableau.

VarType, fonction

[Voir aussi](#)

Description

Renvoie une valeur indiquant le sous-type d'une variable.

Syntaxe

VarType(varname)

L'argument *varname* représente toute [variable](#).

Valeurs renvoyées

La fonction **VarType** renvoie les valeurs suivantes :

Constante	Valeur	Description
vbEmpty	0	Empty (non initialisée)
vbNull	1	Null (aucune donnée valide)
vbInteger	2	Entier
vbLong	3	Entier long
vbSingle	4	Nombre en virgule flottante en simple précision
vbDouble	5	Nombre en virgule flottante en double précision
vbCurrency	6	Monétaire
vbDate	7	Date
vbString	8	Chaîne
vbObject	9	Objet Automation
vbError	10	Erreur
vbBoolean	11	Booléen

vbVariant	12	Variant (utilisé seulement avec des tableaux de Variants)
vbDataObject	13	Objet non Automation
vbByte	17	Octet
vbArray	8192	Tableau

Remarque Ces [constantes](#) sont spécifiées par VBScript. En conséquence, les noms peuvent être utilisés n'importe où dans votre code à la place des valeurs réelles.

Notes

La fonction **VarType** ne renvoie jamais la valeur du sous-type Tableau par elle-même. Elle est toujours ajoutée à une autre valeur pour indiquer un tableau d'un type particulier. La valeur du sous-type Variant n'est renvoyée que si elle a été ajoutée à la valeur du sous-type Tableau pour indiquer que l'argument de la fonction **VarType** est un tableau. Par exemple, la valeur renvoyée pour un tableau d'entiers est calculée comme 2 + 8192 ou 8194. Si un objet possède une [propriété](#) par défaut, la fonction **VarType (object)** renvoie le type de cette propriété.

L'exemple ci-dessous utilise la fonction **VarType** pour déterminer le sous-type d'une variable.

```
Dim MyCheck
```

```
MyCheck = VarType(300)           ' Renvoie 2.
```

```
MyCheck = VarType(#10/19/62#)   ' Renvoie 7.
```

```
MyCheck = VarType("VBScript")  ' Renvoie 8.
```

VBScript, constantes

[Voir aussi](#)

VBScript présente de nombreuses constantes très utiles, que vous pouvez employer dans l'écriture de votre code. Les constantes sont un moyen pratique d'utiliser des valeurs spécifiques sans qu'il soit nécessaire de se souvenir de celles-ci. Elles rendent également votre code plus facile à entretenir, lorsque la valeur d'une constante est modifiée. Du fait que ces constantes sont déjà définies dans VBScript, il ne vous est pas nécessaire de les déclarer de façon explicite dans votre code. Il vous suffit de les utiliser à la place des valeurs qu'elles représentent.

Voici les différentes catégories de constantes disponibles dans VBScript ainsi qu'une brève description de chacune d'elle :

[Chaîne, constantes](#)

Définissent un éventail de caractères qui ne sont pas imprimés et qui servent à la manipulation des chaînes.

[Couleur, constantes](#)

Définissent les huit couleurs de base qui sont utilisées dans l'écriture d'un script.

[Date/Heure, constantes](#)

Définissent les constantes d'heure et de date utilisées par les fonctions correspondantes.

[Divers, constantes](#)

Définissent les constantes qui ne correspondent à aucune autre catégorie.

[Format de date, constantes](#)

Définissent les constantes utilisées pour présenter les dates et les heures.

[MsgBox, constantes](#)

Définissent les constantes utilisées dans la fonction **MsgBox** afin de décrire la visibilité du bouton, son étiquetage, son comportement et les valeurs renvoyées.

[3-états, constantes](#)

Définissent les constantes utilisées dans les fonctions qui mettent en forme les nombres.

[VarType, constantes](#)

Définissent les différents sous-types Variant.

Weekday, fonction

[Voir aussi](#)

Description

Renvoie un nombre entier représentant le jour de la semaine.

Syntaxe

Weekday(*date*, [*firstdayofweek*])

La syntaxe de la fonction **Weekday** comprend les éléments suivants :

Élément	Description
<i>date</i>	Toute expression représentant une date . Si l'argument <i>date</i> contient <i>date</i> la valeur Null , la valeur Null est renvoyée.
<i>firstdayofweek</i>	Une constante qui spécifie le premier jour de la semaine. Si cette valeur est omise, vbSunday est utilisé par défaut.

Valeurs

L'argument *firstdayofweek* peut prendre les valeurs suivantes :

Constante	Valeur	Description
vbUseSystem	0	Utilise la valeur de l'API NLS.
vbSunday	1	Dimanche
vbMonday	2	Lundi
vbTuesday	3	Mardi
vbWednesday	4	Mercredi
vbThursday	5	Jeudi
vbFriday	6	Vendredi

vbSaturday	7	Samedi
-------------------	---	--------

Valeurs renvoyées

La fonction **Weekday** renvoie l'une des valeurs suivantes :

Constante	Valeur	Description
vbSunday	1	Dimanche
vbMonday	2	Lundi
vbTuesday	3	Mardi
vbWednesday	4	Mercredi
vbThursday	5	Jeudi
vbFriday	6	Vendredi
vbSaturday	7	Samedi

Notes

L'exemple ci-dessous utilise la fonction **Weekday** pour obtenir le jour de la semaine correspondant à une date spécifiée :

```
Dim MyDate, MyWeekDay
```

```
MyDate = #October 19, 1962# ' Affecter une date.
```

```
MyWeekDay = Weekday(MyDate) ' MyWeekDay contient 6 par  
' MyDate représente un vendredi.
```

WeekdayName, fonction

[Voir aussi](#)

Description

Renvoie une chaîne indiquant le jour de la semaine spécifié.

Syntaxe

WeekdayName(*weekday*, *abbreviate*, *firstdayofweek*)

La syntaxe de la fonction **WeekdayName** comprend les éléments suivants :

Élément	Description
<i>weekday</i>	Désignation numérique du jour de la semaine. La valeur numérique du jour de la semaine dépend de la valeur du paramètre <i>firstdayofweek</i> .
<i>abbreviate</i>	Facultatif. Valeur booléenne indiquant si le nom du jour de semaine doit être abrégé. Si cette valeur est omise, la valeur par défaut est False , ce qui signifie que le nom du jour de semaine n'est pas abrégé.
<i>firstdayofweek</i>	Facultatif. Valeur numérique indiquant le premier jour de la semaine.

Valeurs

L'argument *firstdayofweek* prend les valeurs suivantes :

Constante	Valeur	Description
vbUseSystem	0	Utilise la valeur API NLS.
vbSunday	1	Dimanche (par défaut)

vbMonday	2	Lundi
vbTuesday	3	Mardi
vbWednesday	4	Mercredi
vbThursday	5	Jeudi
vbFriday	6	Vendredi
vbSaturday	7	Samedi

Notes

L'exemple ci-dessous utilise la fonction **WeekDayName** pour renvoyer le jour spécifié :

```
Dim MyDate
```

```
MyDate = WeekDayName(6, True) ' MyDate contient Ven.
```

While...Wend, instruction

[Voir aussi](#)

Description

Exécute une série d'instructions tant qu'une condition donnée est **True**.

Syntaxe

While *condition*

Version [*statements*]

Wend

La syntaxe de l'instruction **While...Wend** comprend les éléments suivants :

Élément	Description
<i>condition</i>	Expression numérique ou expression de chaîne qui produit la valeur True ou False . Si <i>condition</i> est Null , l'élément <i>condition</i> est traité comme étant False .
<i>statements</i>	Une ou plusieurs instructions exécutées alors que condition est True .

Notes

Si *condition* est **True**, toutes les instructions contenues dans *statements* sont exécutées jusqu'à ce que l'instruction **Wend** soit rencontrée. L'instruction **While** prend ensuite le contrôle et l'élément *condition* est à nouveau vérifié. Si *condition* est toujours **True**, le processus est répété. Si condition n'est pas **True**, l'exécution reprend en commençant par l'instruction succédant à l'instruction **Wend**.

Des boucles **While...Wend** peuvent être imbriquées à tous les niveaux. Chaque instruction **Wend**

correspond à l'instruction **While** la plus récente.

Conseil L'instruction **Do...Loop** fournit un moyen plus structuré et plus souple d'effectuer une itération en boucle.

L'exemple ci-dessous illustre l'utilisation de l'instruction **While...Wend** :

Dim Counter

Counter = 0 ' Initialiser la variable.

While Counter < 20 ' Teste la valeur du compt

Counter = Counter + 1 ' Incrémente le compteur

Alert Counter

Wend ' Fin de la boucle While lorsqu

Counter > 19.

With, instruction

[Voir aussi](#)

Description

Exécute une série d'instructions sur un objet unique.

Syntaxe

With *object*
statements

End With

La syntaxe de l'instruction **With** comprend les éléments suivants :

Élément	Description
<i>object</i>	Nom d'un objet ou d'une fonction qui renvoie un objet.
<i>statements</i>	Requis. Une ou plusieurs instructions à appliquer sur un <i>objet</i> .

Notes

L'instruction **With** vous permet d'effectuer une série d'instructions sur un objet spécifique sans qu'il ne soit nécessaire de qualifier à nouveau le nom de l'objet. Pour modifier le nombre des [propriétés](#) relatives à un objet unique, par exemple, placez les instructions d'affectation de propriété dans la structure de contrôle **With**. Celle-ci fait référence à l'objet une seule fois au lieu de faire référence à l'objet à chaque affectation. L'exemple suivant

montre comment l'instruction **With** est utilisée pour affecter des valeurs à plusieurs propriétés d'un même objet.

```
With MyLabel
  .Height = 2000
  .Width = 2000
  .Caption = "Ceci est MonÉtiquette"
End With
```

La manipulation des propriétés est un aspect important de la fonctionnalité **With** mais elle n'en représente pas la seule utilité. Tout code légal peut être utilisé dans le bloc **With**.

Remarque Lorsque le bloc **With** est saisi, l'*objet* ne peut pas être modifié. Par conséquent, vous ne pouvez pas utiliser une seule instruction **With** pour affecter différents objets.

Il est possible d'imbriquer des instructions **With** en plaçant un bloc **With** à l'intérieur d'un autre. Cependant, les membres des blocs **With** externes étant cachés au sein des blocs **With** internes, il est nécessaire de fournir une référence à l'objet complète dans un bloc **With** interne à tout membre d'un objet figurant dans un bloc **With** externe.

Important Ne passez pas dans ou hors des blocs **With**. Si les instructions d'un bloc **With** sont exécutées, sauf l'instruction **With** ou l'instruction **End With**, vous risquez de constater des erreurs ou un comportement différent.

Xor, opérateur

[Voir aussi](#)

Description

Effectue l'opération logique d'exclusion sur deux expressions.

Syntaxe

result = *expression1* **Xor** *expression2*

La syntaxe de l'opérateur **Xor** comprend les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>expression1</i>	Toute expression .
<i>expression2</i>	Toute expression.

Notes

Si une, et une seule, de ces expressions produit la valeur **True**, *result* vaut **True**. Toutefois, si l'une ou l'autre expression est [Null](#), *result* vaut également **Null**. Si aucune expression n'a la valeur **Null**, l'élément *result* est déterminé conformément au tableau suivant :

Si <i>expression1</i> est	et <i>expression2</i> est	alors <i>result</i> vaut
True	True	False
True	False	True
False	True	True
False	False	False

L'opérateur **Xor** effectue aussi une [comparaison binaire](#) des bits de position identique dans deux [expressions numériques](#) et définit le bit correspondant dans *result* d'après la table suivante :

Si le bit dans <i>expression1</i> est	et le bit dans <i>expression2</i> est	alors <i>result</i> vaut
0	0	0
0	1	1
1	0	1
1	1	0

Year, fonction

[Voir aussi](#)

Description

Renvoie un nombre entier représentant l'année.

Syntaxe

Year(*date*)

L'argument *date* peut être toute expression représentant une date. Si l'argument *date* contient [Null](#), la valeur **Null** est renvoyée.

Notes

L'exemple ci-dessous utilise la fonction **Year** pour obtenir l'année à partir d'une date spécifiée :

```
Dim MyDate, MyYear
MyDate = #October 19, 1962# ' Affecter une date.
MyYear = Year(MyDate)     ' MyYear contient 1962.
```

Couleur, constantes

[Voir aussi](#)

Ces constantes étant intégrées dans VBScript, il n'est pas nécessaire de les définir pour les utiliser. Vous pouvez les insérer n'importe où dans le code pour représenter les valeurs qui leur sont associées.

Constante	Valeur	Description
vbBlack	&h00	Noir
vbRed	&hFF	Rouge
vbGreen	&hFF00	Vert
vbYellow	&hFFFF	Jaune
vbBlue	&hFF0000	Bleu
vbMagenta	&hFF00FF	Magenta
vbCyan	&hFFFF00	Cyan
vbWhite	&hFFFFFF	Blanc

3-états, constantes

[Voir aussi](#)

Ces constantes étant intégrées dans VBScript, il n'est pas nécessaire de les définir pour les utiliser. Vous pouvez les insérer n'importe où dans le code pour représenter les valeurs qui leur sont associées.

Constante	Valeur	Description
vbUseDefault	-2	Utiliser la valeur par défaut des paramètres régionaux.
vbTrue	-1	True
vbFalse	0	False

Erreurs d'exécution de VBScript

[Référence du langage](#)
[Version 1](#)

[Erreurs de syntaxe de VBScript](#)

Numéro de l'erreur	Description
5	Argument ou appel de procédure incorrect
6	Dépassement de capacité
7	Mémoire insuffisante
9	Indice en dehors de la page
10	Ce tableau est fixe ou temporairement verrouillé
11	Division par zéro
13	Type incompatible
14	Espace de chaîne insuffisant
17	Impossible d'effectuer l'opération demandée
28	Espace pile insuffisant
35	Sub ou Function non définie
48	Erreur de chargement de la DLL
51	Erreur interne
52	Numéro ou nom de fichier incorrect
53	Fichier introuvable
54	Mode de fichier incorrect
55	Fichier déjà ouvert
57	Erreur d'entrée/sortie du dispositif
58	Le fichier existe déjà
61	Disque saturé
62	Entrée après la fin du fichier
67	Trop de fichiers
68	Dispositif introuvable
70	Permission refusée
71	Disque non prêt
74	Nouvelle dénomination avec un lecteur différent impossible
75	Erreur dans le chemin d'accès

76	Chemin d'accès introuvable
91	Variable de l'objet non définie
92	Boucle For non initialisée
94	Utilisation incorrecte de Null
322	Impossible de créer fichier temporaire
424	Objet requis
429	Un composant ActiveX ne peut pas créer l'objet
430	La classe ne gère pas Automation
432	Nom de fichier ou de classe introuvable au cours de l'opération Automation
438	Propriété ou méthode non gérée par cet objet
440	Erreur Automation
445	L'objet ne gère pas cette action
446	L'objet ne gère pas les arguments nommés
447	L'objet ne gère pas les paramètres régionaux en cours
448	Argument nommé introuvable
449	Argument obligatoire
450	Nombre d'arguments incorrect ou affectation de propriété incorrecte
451	L'objet n'est pas une collection
453	Fonction de DLL spécifiée introuvable
455	Erreur de verrouillage de la ressource de code
458	La variable utilise un type Automation non géré dans VBScript
462	La machine du serveur distant n'existe pas ou n'est pas disponible
481	Image incorrecte
500	Variable indéfinie
501	Affectation illégale
502	Objet non sécurisé pour le script
503	Objet non sécurisé pour l'initialisation
504	Objet non sécurisé pour la création
505	Référence incorrecte ou non qualifiée
506	Classe non définie
507	Une exception s'est produite
5016	Objet expression régulière attendu
5017	Erreur de syntaxe dans une expression régulière
5018	Quantifiant inattendu
5019	']' attendu dans l'expression régulière
5020	'\)' attendu dans l'expression régulière
5021	Plage incorrecte dans le jeu de caractères
32811	Élément introuvable



Erreurs de syntaxe de VBScript

[Référence du langage](#)
[Version 1](#)

[Erreurs d'exécution de VBScript](#)

Numéro de l'erreur	Description
1001	Mémoire insuffisante
1002	Erreur de syntaxe
1003	'.' attendu
1005	'(' attendu
1006)' attendu
1007	']' attendu
1010	Identificateur attendu
1011	'=' attendu
1012	'If' attendu
1013	'To' attendu
1014	'End' attendu
1015	'Function' attendu
1016	'Sub' attendu
1017	'Then' attendu
1018	'Wend' attendu
1019	'Loop' attendu
1020	'Next' attendu
1021	'Case' attendu
1022	'Select' attendu
1023	Expression attendue
1024	Instruction attendue
1025	Fin d'instruction attendue
1026	Constante (entier) attendue
1027	'While' ou 'Until' attendu
1028	'While', 'Until' ou fin d'instruction attendue
1029	'With' attendu
1030	Identificateur trop long

1031	Nombre incorrect
1032	Caractère incorrect
1033	Constante de chaîne indéterminée
1034	Commentaire indéterminé
1037	Utilisation incorrecte du mot clé 'Me'
1038	'loop' sans 'do'
1039	Instruction 'exit' incorrecte
1040	Variable de contrôle de boucle 'for' incorrecte
1041	Nom redéfini
1042	Ce devrait être la première instruction de la ligne
1043	Affectation impossible à un argument autre que ByVal
1044	Parenthèses interdites lors de l'appel d'une procédure Sub
1045	Constante littérale attendue
1046	'In' attendu
1047	'Class' attendu
1048	Doit être défini à l'intérieur d'une classe
1049	Let, Set ou Get attendu dans la déclaration de propriété
1050	'Property' attendu
1051	Le nombre d'arguments doit être le même dans la spécification des propriétés
1052	Impossible d'avoir plusieurs propriétés/méthodes par défaut dans une classe
1053	L'initialisation ou la fin de classe ne comporte pas d'arguments
1054	La propriété set ou let doit avoir au moins un argument
1055	'Next' inattendu
1056	'Default' ne peut être spécifié que dans 'Property', 'Function' ou 'Sub'
1057	La spécification 'Default' doit également spécifier 'Public')
1058	La spécification 'Default' ne peut se trouver que dans Property Get

Fonctions de conversion

[Asc, fonction](#)

[CBool, fonction](#)

[CByte, fonction](#)

[CCur, fonction](#)

[CDate, fonction](#)

[CDBl, fonction](#)

[Chr, fonction](#)

[CInt, fonction](#)

[CLng, fonction](#)

[CSng, fonction](#)

[CStr, fonction](#)

[Hex, fonction](#)

[Oct, fonction](#)

Glossaire VBScript

[Référence du langage](#)

argument

Constante, variable ou expression transmise à une procédure.

bibliothèque de types

Fichier ou composant d'un autre fichier contenant des descriptions standard d'objets, méthodes et propriétés exposés.

classe

Définition formelle d'un objet. La classe joue le rôle d'un modèle à partir duquel une instance d'un objet est créée à l'exécution. La classe définit les propriétés de l'objet et les méthodes utilisées pour contrôler le comportement de celui-ci.

code de caractère

Nombre représentant un caractère particulier d'un jeu, le jeu de caractères ASCII par exemple.

collection

Objet contenant un ensemble d'objets en relation. La position d'un objet dans la collection peut être modifiée chaque fois qu'un changement se produit dans la collection. La position dans la collection de tout objet spécifique peut donc varier.

commentaire

Texte ajouté au code par un développeur pour expliquer le fonctionnement du code. Dans Visual Basic Script, une ligne de commentaire commence habituellement par une apostrophe (''); vous pouvez aussi utiliser le mot clé **Rem** suivi d'un espace.

comparaison binaire

Comparaison bit par bit des bits dont la position est identique dans deux expressions numériques.

comparaison de chaîne

Combinaison de deux séquences de caractères. Sauf spécification contraire dans la fonction de comparaison, toutes les comparaisons de chaîne sont binaires. En Anglais, les comparaisons binaires respectent la casse, au contraire des comparaisons de texte.

constante

Élément nommé qui conserve une valeur constante pendant toute l'exécution d'un programme. Vous pouvez utiliser des constantes n'importe où dans votre code à la place de valeurs réelles. Une constante peut être une chaîne ou un littéral numérique, une autre constante ou toute combinaison incluant des opérateurs arithmétiques ou logiques à l'exception de **Is** et de l'élévation à une puissance. Par exemple :

by value

```
Const A = "MaChaîne"
```

constante intrinsèque

Constante fournie par une application. Etant donné qu'une constante intrinsèque ne peut être désactivée, vous ne pouvez pas créer de constante utilisateur portant le même nom.

contrôle ActiveX

Objet placé sur une feuille pour permettre ou améliorer l'interaction de l'utilisateur avec l'application. Les contrôles ActiveX comprennent des événements et peuvent être incorporés à d'autres contrôles. Leur nom de fichier comporte l'extension .ocx.

Empty

Valeur indiquant qu'aucune valeur n'a encore été affectée à une variable. Les variables **Empty** correspondent à 0 dans un contexte numérique ou de longueur nulle dans un contexte de chaîne.

erreur d'exécution

Erreur qui se produit au cours de l'exécution du code. Une erreur d'exécution se produit lorsqu'une instruction tente une opération incorrecte.

exécution

Phase pendant laquelle le code s'exécute. Pendant l'exécution, vous ne pouvez pas modifier le code.

expression

Combinaison de mots clés, d'opérateurs, de variables et de constantes qui produit une chaîne, un nombre ou un objet. Une expression peut effectuer un calcul, manipuler des caractères ou tester des données.

Is est également un opérateur de comparaison, mais il est exclusivement utilisé pour déterminer si la référence d'un objet est identique à une autre.

expression booléenne

Élément nommé qui conserve une valeur constante pendant toute l'exécution d'un programme. Vous pouvez utiliser des constantes n'importe où dans votre code à la place des valeurs réelles. Une constante peut être une chaîne ou un littéral numérique, une autre constante ou toute combinaison incluant des opérateurs arithmétiques ou logiques à l'exception de **Is** et de l'élévation à une puissance. Exemple :

Expression dont le résultat est **True** ou **False**.

expression de chaîne

Toute expression qui peut être évaluée en séquence de caractères contigus. Parmi les éléments d'une expression de chaîne peuvent figurer une fonction renvoyant une chaîne, une constante de chaîne, un littéral chaîne ou une variable chaîne.

Sous-type	Plage
Byte	0 à 255.
Boolean	True ou False .
Integer	-32,768 à 32,767.
Long	-2,147,483,648 à 2,147,483,647.
Single	-3.402823E38 à -1.401298E-45 pour les valeurs négatives ; 1.401298E-45 à 3.402823E38 pour les valeurs positives.
Double	-1.79769313486232E308 à -4.94065645841247E-324 pour les valeurs négatives ; 4.94065645841247E-324 à 1.79769313486232E308 pour les valeurs

	positives.
Currency	-922,337,203,685,477.5808 à 922,337,203,685,477.5807.
Date	Du 1er janvier 100 au 31 décembre 9999, inclus.
Object	Toute référence Object .
String	Chaînes de longueur variable comprise entre 0 et approximativement 2 milliards de caractères.

expression de date

Toute expression pouvant être interprétée comme une date. Ceci comprend toute combinaison de littéraux de date, de nombres ou de chaînes ressemblant à des dates, et de dates retournées par des fonctions. Une expression de date est limitée aux nombres ou chaînes, sous toute forme de combinaison, qui représentent une date comprise entre le 1er janvier 100 et le 31 décembre 9999.

Les dates sont enregistrées comme partie d'un nombre réel. Les valeurs à gauche de la virgule représentent la date, tandis que les valeurs à droite représentent le temps. Les valeurs négatives correspondent aux dates antérieures au 30 décembre 1899.

expression numérique

Toute expression pouvant être résolue sous forme de nombre. Les éléments de l'expression peuvent inclure toute combinaison de mots clés, de variables, de constantes et d'opérateurs dont le résultat est un nombre.

VBScript interprète toujours un littéral date comme Anglais-U.S. s'il en a la possibilité. Si un littéral date ne peut pas être interprété comme une date, une erreur survient.

module de classe

Module contenant la définition d'une classe (la définition de ses propriétés et de ses méthodes).

mot clé

Mot ou symbole reconnu comme faisant partie du langage VBScript ; par exemple une instruction, un nom de fonction ou un opérateur.

niveau procédure

Sub MaSub() ' Cette instruction déclare une procédure Sub. Dim A ' Cette instruction débute le corps de la procédure. A = "Ma variable" ' Code de niveau procédure. Debug.Print A ' Code de niveau procédure. End Sub ' Cette instruction finit la procédure Sub.

niveau script

Tout code figurant hors d'une procédure est désigné comme code de niveau script.

Nothing

Valeur spéciale indiquant qu'une variable objet n'est plus associée à un objet réel.

Null

Valeur indiquant qu'une variable ne contient aucune donnée valide. **Null** est le résultat :

- d'une affectation explicite de la valeur **Null** à une variable ;
 - de toute opération entre expressions contenant la valeur **Null**.
-

numéro de l'erreur

Nombre entier compris entre 0 et 65 535 inclus qui correspond à la propriété **Number** de l'objet **Err**. Lorsqu'il est combiné à la propriété **Name** de l'objet **Err**, ce numéro représente un message d'erreur particulier.

objet ActiveX

Objet exposé aux autres applications ou outils de programmation par l'intermédiaire des interfaces Automation.

- An explicit assignment of **Null** to a variable.
 - Any operation between expressions that contain **Null**.
-

objet Automation

Objet exposé aux autres applications ou outils de programmation par l'intermédiaire des interfaces Automation.

opérateur de comparaison

Is est aussi un opérateur de comparaison, mais il est exclusivement utilisé pour déterminer si une référence d'un objet est identique à une autre.

paramètres régionaux

Informations relatives à une langue et à un pays donné qui ont une incidence particulière sur le code et le système.

par référence

Moyen de transmettre l'adresse, au lieu de la valeur, d'un argument à une procédure. Ceci permet à la procédure d'accéder à la variable elle-même. En conséquence, la valeur de la variable peut être modifiée par la procédure à laquelle elle est transmise.

par valeur

Moyen de transmettre la valeur, au lieu de l'adresse, d'un argument à une procédure. Ceci permet à la procédure d'accéder à une copie de la variable. En conséquence, la valeur de la variable ne peut pas être modifiée par la procédure à laquelle elle est transmise.

pi

Décrit les instructions situées à l'intérieur d'une procédure **Function** ou **Sub**. Généralement, les déclarations figurent en premier, suivies des affectations et du code exécutable. Exemple :

```
Sub MySub() ' Cette instruction déclare un bloc de sous-procédure.  
  Dim A ' Cette instruction démarre le bloc de procédure.  
  A = "My variable" ' Code de niveau de procédure.  
  Debug.Print A ' Code de niveau de procédure.  
End Sub ' Cette instruction met fin au bloc de sous-procédure.
```

Pi est une constante mathématique approximativement égale à 3,1415926535897932.

plages de données

Chaque sous-type Variant possède une plage spécifique de valeurs autorisées :

portée

Définit la visibilité d'une variable, d'une procédure ou d'un objet. Par exemple, une variable déclarée comme **Public** est visible pour toutes les procédures dans tous les modules. Les variables déclarées dans les procédures ne sont visibles qu'à l'intérieur de la procédure et elles perdent leur valeur entre les appels.

Private

Variables visibles uniquement à l'intérieur du script dans lequel elles sont déclarées.

procédure

Séquence nommée d'instructions exécutée comme une unité. Par exemple, **Function** et **Sub** sont des types de procédures.

propriété

Attribut nommé d'objet. Les propriétés définissent les caractéristiques de l'objet telles que la taille, la couleur, l'emplacement sur l'écran ou l'état d'un objet (activé ou désactivé, par exemple).

Public

Variables déclarées avec l'instruction **Public** ; elles sont visibles par toutes les procédures de tous les modules de toutes les applications.

SCODE

Entier long utilisé pour transmettre des informations détaillées à l'appelant d'un membre d'interface ou d'une fonction API. Les codes d'état des interfaces OLE et des API sont définis dans FACILITY_ITF.

séparateurs de date

Caractères utilisés pour séparer le jour, le mois et l'année quand les valeurs de date sont mises en forme.

tableau

Ensemble d'éléments indexés de façon séquentielle et ayant le même type de données. Chaque élément d'un tableau est repéré par un numéro d'index unique. Les changements effectués sur un seul élément d'un tableau n'affectent pas les autres éléments.

type d'objet

Type d'objet exposé par une application, par exemple, Application, File, Range et Sheet. Pour obtenir une liste complète des objets disponibles, reportez-vous à la documentation de l'application (Microsoft Excel, Microsoft Project, Microsoft Word, etc.).

valeur initiale

Valeur initiale utilisée pour générer des nombres pseudo-aléatoires. Par exemple, l'instruction **Randomize** crée une valeur initiale utilisée par la fonction **Rnd** afin de générer des séquences de nombres pseudo-aléatoires uniques.

variable

Emplacement de stockage nommé qui peut contenir des données pouvant être modifiées pendant l'exécution du programme. Chaque variable a un nom qui l'identifie de façon unique à l'intérieur de son niveau de portée.

Les noms de variables :

- doivent commencer par un caractère alphabétique ;
 - ne peuvent contenir une virgule intégrée ou un caractère de déclaration de type ;
 - doivent être uniques à l'intérieur de la même portée ;
 - ne doivent pas dépasser 255 caractères.
-

Opérateurs arithmétiques

[Référence du langage](#)
[Version 1](#)

[^, opérateur](#)

[*, opérateur](#)

[/, opérateur](#)

[\, opérateur](#)

[Mod, opérateur](#)

[+, opérateur](#)

[-, opérateur](#)

[Opérateurs de concaténation](#)

< opérateur" > opérateur" > <= opérateur" > = opérateur" > opérateur" > < operator;
less than operator; > operator; greater than operator; <= operator; less than or
equal to operator; >= operator; greater than or equal to operator; = operator;
equal operator; operator; not equal operator; string comparison; Empty">

Microsoft® Visual Basic® Scripting Edition

Opérateurs de comparaison

[Référence du langage](#)
[Version 1](#)

[Voir aussi](#)

Description

Utilisés pour comparer des expressions.

Syntaxe

result = *expression1* *comparisonoperator* *expression2*

result = *object1* **Is** *object2*

Les opérateurs de comparaison comportent les éléments suivants :

Élément	Description
<i>result</i>	Toute variable numérique.
<i>expression</i>	Toute expression .
<i>comparisonoperator</i>	Tout opérateur de comparaison .
<i>object</i>	Tout nom d'objet.

Notes

L'opérateur **Is** comporte une fonctionnalité de comparaison spécifique qui diffère des autres opérateurs dans le tableau suivant. Ce tableau contient une liste des opérateurs de comparaison et des conditions qui déterminent si la valeur de *result* est **True**, **False** ou [Null](#) :

Opérateur	Description	True si	False si	Null si
-----------	-------------	---------	----------	---------

<	Inférieur à	<i>expression1</i> < <i>expression2</i>	<i>expression1</i> >= <i>expression2</i>	<i>expression1</i> ou <i>expression2</i> = Null
<=	Inférieur à ou égal à	<i>expression1</i> <= <i>expression2</i>	<i>expression1</i> > <i>expression2</i>	<i>expression1</i> ou <i>expression2</i> = Null
>	Supérieur à	<i>expression1</i> > <i>expression2</i>	<i>expression1</i> <= <i>expression2</i>	<i>expression1</i> ou <i>expression2</i> = Null
>=	Supérieur à ou égal à	<i>expression1</i> >= <i>expression2</i>	<i>expression1</i> < <i>expression2</i>	<i>expression1</i> ou <i>expression2</i> = Null
=	Égal à	<i>expression1</i> = <i>expression2</i>	<i>expression1</i> <> <i>expression2</i>	<i>expression1</i> ou <i>expression2</i> = Null
<>	Différent de	<i>expression1</i> <> <i>expression2</i>	<i>expression1</i> = <i>expression2</i>	<i>expression1</i> ou <i>expression2</i> = Null

Lorsque vous comparez deux expressions, il n'est pas toujours facile de déterminer si elles ont été comparées comme nombres ou comme chaînes.

Le tableau suivant décrit la manière dont les expressions sont comparées, ou ce qui résulte de la comparaison, en fonction du sous-type sous-jacent :

Si	alors
Les deux expressions sont numériques	Effectue une comparaison numérique.
Les deux expressions sont des chaînes	Effectue une comparaison de chaînes .

chaînes	
Une expression est numérique et l'autre est une chaîne	L'expression numérique est inférieure à l'expression de chaîne.
Une expression est Empty et l'autre est numérique	Effectue une comparaison numérique, en utilisant 0 comme expression Empty .
Une expression est Empty et l'autre est une chaîne	Effectue une comparaison de chaîne, en utilisant une chaîne de longueur nulle ("") comme l'expression Empty .
Les deux expressions sont Empty	Les expressions sont égales

Opérateurs de concaténation

[Référence du langage](#)
[Version 1](#)

[&, opérateur](#)

[+, opérateur](#)

Opérateurs logiques

[Référence du langage](#)
[Version 1](#)

[And, opérateur](#)

[Not, opérateur](#)

[Or, opérateur](#)

[Xor, opérateur](#)

Référence du langage VBScript

[Didacticiel VBScript](#)
[Informations sur la version](#)

- ◆ Caractéristiques
- ◆ Liste des mots clés
- ◆ Constantes
- ◆ Erreurs
- ◆ Événements
- ◆ Fonctions
- ◆ Méthodes
- ◆ Objets
- ◆ Opérateurs
- ◆ Propriétés

Bienvenue dans la référence du langage Scripting

Ces différentes sections présentent des informations qui vous aideront à découvrir les divers composants du langage Visual Basic Scripting.

Vous trouverez *tous* les composants du langage VBScripting répertoriés par ordre alphabétique sous la liste des mots clés. Si vous souhaitez vous concentrer sur une seule catégorie, les objets par exemple, chaque catégorie du langage dispose de sa propre section plus concise.

Cliquez sur l'un des intitulés de gauche pour afficher la liste des éléments contenus dans cette catégorie. À partir de cette liste, sélectionnez la rubrique souhaitée. Une fois cette rubrique affichée, vous pourrez facilement naviguer vers d'autres rubriques

Maintenant, à vous de jouer ! Étudiez des

◆ Instructions

instructions, examinez les méthodes ou découvrez quelques caractéristiques. Vous constaterez la richesse du langage VBScript !

Informations sur la version

[Référence du langage](#)

Le tableau suivant indique la version de Microsoft Visual Basic Scripting Edition utilisée par les applications hôtes.

Application hôte	Version VBScript				
	1.0	2.0	3.0	4.0	5.0
Microsoft Internet Explorer 3.0	x				
Microsoft Internet Information Server 3.0		x			
Microsoft Internet Explorer 4.0			x		
Microsoft Internet Information Server 4.0			x		
Microsoft Windows Scripting Host 1.0			x		
Microsoft Outlook 98			x		
Microsoft Visual Studio 6.0				x	
Microsoft Internet Explorer 5.0					x
Microsoft Internet Information Services 5.0					x

Le tableau suivant dresse la liste des caractéristiques du langage VBScript et signale la version lors de la première apparition.

Élément du langage	Version lors de la première apparition				
	1.0	2.0	3.0	4.0	5.0
Abs, fonction	x				
Opérateur d'addition (+)	x				
Opérateur And	x				
Array, fonction		x			

Asc, fonction	x				
Opérateur d'affectation (≡)	x				
Atn, fonction	x				
Call, instruction	x				
CBool, fonction	x				
CByte, fonction	x				
CCur, fonction	x				
CDate, fonction	x				
CDBl, fonction	x				
Chr, fonction	x				
CInt, fonction	x				
Class, objet					x
Class, instruction					x
Clear, méthode	x				
CLng, fonction	x				
Couleur, constantes		x			
Comparaison, constantes		x			
Opérateur de concaténation (&)	x				
Const, instruction		x			
Cos, fonction	x				
CreateObject, fonction		x			
CSng, fonction	x				
CStr, fonction	x				
Date/Heure, constantes		x			
Format de date, constantes		x			
Date, fonction	x				
DateAdd, fonction		x			
DateDiff, fonction		x			
DatePart, fonction		x			
DateSerial, fonction	x				

DateValue, fonction	X				
Day, fonction	X				
Description, propriété	X				
Dim, instruction	X				
Opérateur de division (/)	X				
Do...Loop, instruction	X				
Empty	X				
Opérateur Eqv	X				
Erase, instruction	X				
Err, objet	X				
Eval, fonction					X
Execute, méthode					X
Execute, instruction					X
ExecuteGlobal, instruction					X
Exit, instruction	X				
Exp, fonction	X				
Opérateur d'exponentiation (^)	X				
False	X				
Filter, fonction		X			
FirstIndex, propriété					X
Fix, fonction	X				
For...Next, instruction	X				
For Each...Next, instruction		X			
FormatCurrency, fonction		X			
FormatDateTime, fonction		X			
FormatNumber, fonction		X			
FormatPercent, fonction		X			
Function, instruction	X				
GetLocale, fonction					X

GetObject, fonction		x			
GetRef, fonction					x
Global, propriété					x
Hex, fonction	x				
HelpContext, propriété		x			
HelpFile, propriété		x			
Hour, fonction	x				
If...Then...Else, instruction	x				
IgnoreCase, propriété					x
Opérateur Imp	x				
Initialize, événement					x
InputBox, fonction	x				
InStr, fonction	x				
InStrRev, fonction		x			
Int, fonction	x				
Opérateur de division entière (\)	x				
Opérateur Is	x				
IsArray, fonction	x				
IsDate, fonction	x				
IsEmpty, fonction	x				
IsNull, fonction	x				
IsNumeric, fonction	x				
IsObject, fonction	x				
Join, fonction		x			
LBound, fonction	x				
LCase, fonction	x				
Left, fonction	x				
Len, fonction	x				
Length, propriété					x

LoadPicture, fonction		x			
Log, fonction	x				
LTrim, fonction	x				
Match, objet					x
Matches, collection					x
Mid, fonction	x				
Minute, fonction	x				
Divers, constantes		x			
Opérateur Mod	x				
Month, fonction	x				
MonthName, fonction		x			
MsgBox, constantes		x			
MsgBox, fonction	x				
Opérateur de multiplication (*)	x				
Opérateur de négation (-)	x				
Opérateur Not	x				
Now, fonction	x				
Nothing	x				
Null	x				
Number, propriété	x				
Oct, fonction	x				
On Error, instruction	x				
Option Explicit, instruction	x				
Opérateur Or	x				
Pattern, propriété					x
Private, instruction		x			
PropertyGet, instruction					x
PropertyLet, instruction					x
PropertySet, instruction					x

Public, instruction		X			
Raise, méthode	X				
Randomize, instruction	X				
ReDim, instruction	X				
RegExp, objet					X
Rem, instruction	X				
Replace, fonction		X			
Replace, méthode					X
RGB, fonction		X			
Right, fonction	X				
Rnd, fonction	X				
Round, fonction		X			
RTrim, fonction	X				
ScriptEngine, fonction		X			
ScriptEngineBuildVersion, fonction		X			
ScriptEngineMajorVersion, fonction		X			
ScriptEngineMinorVersion, fonction		X			
Second, fonction	X				
Select Case, instruction	X				
Set, instruction	X				
SetLocale, fonction					X
Sgn, fonction	X				
Sin, fonction	X				
Source, propriété	X				
Space, fonction	X				
Split, fonction		X			
Sqr, fonction	X				

StrComp, fonction	X				
Chaîne, constantes		X			
String, fonction	X				
StrReverse, fonction		X			
Sub, instruction	X				
Opérateur de soustraction (-)	X				
Tan, fonction	X				
Terminate, événement					X
Test, méthode					X
Time, fonction	X				
Timer, fonction					X
TimeSerial, fonction	X				
TimeValue, fonction	X				
Trim, fonction	X				
3-états, constantes		X			
True	X				
TypeName, fonction		X			
UBound, fonction	X				
UCase, fonction	X				
Value, propriété					X
VarType, constantes		X			
VarType, fonction	X				
VBScript, constantes		X			
Weekday, fonction	X				
WeekdayName, fonction		X			
While...Wend, instruction	X				
With, instruction					X
Opérateur Xor	X				
Year, fonction	X				

Add, méthode (Dictionary)

[Voir aussi](#)

[Application](#)

Description

Ajoute une paire clé/élément à un objet **Dictionary**.

Syntaxe

object.**Add** *key*, *item*

La méthode **Add** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet Dictionary .
<i>key</i>	Argument <i>key</i> associé à l'argument <i>item</i> ajouté.
<i>item</i>	Argument <i>item</i> associé à l'argument <i>key</i> ajouté.

Notes

Une erreur se produit si *key* existe déjà.

L'exemple ci-dessous illustre l'utilisation de la méthode **Add** :

```
Dim d           ' Crée une variable.  
Set d = CreateObject("Scripting.Dictionary")  
d.Add "a", "Athènes" ' Ajoute des clés et des élé  
d.Add "b", "Belgrade"  
d.Add "c", "Casablanca"
```



Add, méthode (Folders)

[Voir aussi](#)

[Application](#)

Description

Ajoute un nouveau élément **Folder** à une collection **Folders**.

Syntaxe

object.**Add**(*folderName*)

La méthode **Add** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'une collection Folders collection.
<i>folderName</i>	Le nom du nouveau Folder ajouté.

Notes

L'exemple ci-dessous illustre l'utilisation de la méthode **Add** pour ajouter un nouveau dossier :

```
Sub AddNewFolder(path, folderName)
    Dim fso, f, fc, nf
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFolder(path)
    Set fc = f.SubFolders
    If folderName <> "" Then
        Set nf = fc.Add(folderName)
    Else
        Set nf = fc.Add("Nouveau dossier")
    End If
End Sub
```

End If
End Sub

Une erreur se produit si *folderName* existe déjà.

AtEndOfLine, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie la valeur **True** si le pointeur de fichier est positionné immédiatement après le marqueur de fin de ligne d'un fichier **TextStream**. La valeur **False** est renvoyée dans le cas contraire. En lecture seule.

Syntaxe

object.**AtEndOfLine**

L'espace réservé *object* correspond toujours à un nom d'objet **TextStream**.

Notes

La propriété **AtEndOfLine** s'applique uniquement aux fichiers **TextStream** qui sont ouverts en lecture ; sinon, une erreur se produit.

Le code suivant illustre l'emploi de la propriété **AtEndOfLine** :

```
Function ReadEntireFile(filespec)
    Const ForReading = 1
    Dim fso, theFile, retstring
    Set fso = CreateObject("Scripting.FileSystemObj
    Set theFile = fso.OpenTextFile(filespec, ForReadi
    Do While theFile.AtEndOfLine <> True
        retstring = theFile.Read(1)
    Loop
```

```
theFile.Close  
ReadEntireFile = retstring  
End Function
```

AtEndOfStream, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie la valeur **True** si le pointeur de fichier se trouve à la fin d'un fichier **TextStream**. La valeur **False** est renvoyée dans le cas contraire. En lecture seule.

Syntaxe

object.**AtEndOfStream**

L'espace réservé *object* correspond toujours au nom d'un objet **TextStream**.

Notes

La propriété **AtEndOfStream** s'applique seulement aux fichiers **TextStream** qui sont ouverts en lecture, sinon une erreur se produit.

Le code suivant illustre l'emploi de la propriété **AtEndOfStream** :

```
Function ReadEntireFile(filespec)
    Const ForReading = 1
    Dim fso, theFile, retstring
    Set fso = CreateObject("Scripting.FileSystemObj
    Set theFile = fso.OpenTextFile(filespec, ForReadi
    Do While theFile.AtEndOfStream <> True
        retstring = theFile.ReadLine
    Loop
```

```
theFile.Close  
ReadEntireFile = retstring  
End Function
```

Attributes, propriété

[Voir aussi](#)

[Application](#)

Description

Définit ou renvoie les attributs de fichiers ou de dossiers. En lecture/écriture ou lecture seule, en fonction de l'attribut.

Syntaxe

object.**Attributes** [= *newattributes*]

La propriété **Attributes** comprend les éléments suivants :

Élément	Description
<i>object</i>	Requis. Correspond toujours au nom d'un objet File ou Folder .
<i>newattributes</i>	Facultatif. Si elle est fournie, la valeur <i>newattributes</i> est la nouvelle valeur des attributs de l' <i>object</i> spécifié.

Valeurs

L'argument *newattributes* peut prendre les valeurs suivantes ou toute combinaison de ces valeurs :

Constante	Valeur	Description
Normal	0	Fichier normal. Aucun attribut n'est défini.
ReadOnly	1	Fichier en lecture seule. L'attribut est lecture/écriture.
Hidden	2	Fichier caché. L'attribut est lecture/écriture.

System	4	Fichier système. L'attribut est lecture/écriture.
Directory	16	Dossier ou répertoire. L'attribut est lecture seule.
Archive	32	Le fichier a été modifié depuis la dernière sauvegarde. L'attribut est lecture/écriture.
Alias	1024	Lien ou raccourci. L'attribut est lecture seule.
Compressed	2048	Fichier compressé. L'attribut est lecture seule.

Notes

Toute tentative de modification de l'un des attributs en lecture seule (Alias, Compressed ou Directory) est ignorée.

Lors de la définition d'attributs, il est généralement recommandé de commencer par lire les attributs en cours, puis de changer chaque attribut individuellement, si besoin est, puis de réécrire les attributs.

Le code suivant illustre l'emploi de la propriété **Attributes** avec un fichier :

Function ToggleArchiveBit(filespec)

Dim fso, f

Set fso = CreateObject("Scripting.FileSystemObj

Set f = fso.GetFile(filespec)

If **f.attributes** and 32 Then

f.attributes = **f.attributes** - 32

ToggleArchiveBit = "Bit d'archive effacé."

Else

f.attributes = **f.attributes** + 32

ToggleArchiveBit = "Bit d'archive défini."

End If

End Function

AvailableSpace, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie la quantité d'espace libre disponible pour un utilisateur sur le lecteur ou le partage réseau.

Syntaxe

object.**AvailableSpace**

L'argument *object* représente toujours un objet **Drive**.

Notes

La valeur renvoyée par la propriété **AvailableSpace** est en général la même que celle renvoyée par la propriété **FreeSpace**. Elles peuvent présenter des différences dans le cas des systèmes d'ordinateur qui gèrent les quotas.

Le code suivant illustre l'utilisation de la propriété **AvailableSpace** :

```
Function ShowAvailableSpace(drvPath)
    Dim fso, d, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set d = fso.GetDrive(fso.GetDriveName(drvPath))
    s = "Lecteur " & UCase(drvPath) & " - "
    s = s & d.VolumeName & "<BR>"
    s = s & "Espace disponible: " & FormatNumber(d.AvailableSpa
    s = s & " KOctets"
    ShowAvailableSpace = s
End Function
```

BuildPath, méthode

[Voir aussi](#)

[Application](#)

Description

Ajoute un nom à un chemin existant.

Syntaxe

object.**BuildPath**(*path*, *name*)

La syntaxe de la méthode **BuildPath** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>path</i>	Chemin existant auquel <i>name</i> est ajouté. Le chemin peut être absolu ou relatif et n'est pas obligé de correspondre à un chemin existant.
<i>name</i>	Nom ajouté au <i>path</i> existant.

Notes

La méthode **BuildPath** insère si nécessaire un séparateur de chemin entre le chemin existant et le nouveau nom.

L'exemple ci-dessous illustre l'utilisation de la méthode **BuildPath** :

```
Function GetBuildPath(path)
```

```
    Dim fso, newpath
```

```
    Set fso = CreateObject("Scripting.FileSystemObj
```

```
newpath = fso.BuildPath(path, "Sub Folder")  
GetBuildPath = newpath  
End Function
```

Close, méthode

[Voir aussi](#)

[Application](#)

Description

Ferme un fichier **TextStream** ouvert.

Syntaxe

object.**Close**

L'espace réservé *object* correspond toujours au nom d'un objet **TextStream**.

Notes

L'exemple ci-dessous illustre l'utilisation de la méthode **Close** pour fermer un fichier **TextStream** :

```
Sub CreateAFile
  Dim fso, MyFile
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set MyFile = fso.CreateTextFile("c:\testfile.txt", True)
  MyFile.WriteLine("Ceci est un test.")
  MyFile.Close
End Sub
```

Column, propriété

[Voir aussi](#)[Application](#)

Description

Propriété en lecture seule. Renvoie le numéro de colonne de la position de caractère courante dans un fichier **TextStream**.

Syntaxe

object.**Column**

L'espace réservé *object* correspond toujours au nom d'un objet **TextStream**.

Notes

Après l'écriture d'un caractère de nouvelle ligne, mais avant l'écriture de tout autre caractère, la propriété **Column** a la valeur 1.

L'exemple ci-dessous illustre l'utilisation de la propriété **Column** :

Function GetColumn

```
Const ForReading = 1, ForWriting = 2
```

```
Dim fso, f, m
```

```
Set fso = CreateObject("Scripting.FileSystemObj
```

```
Set f = fso.OpenTextFile("c:\testfile.txt", ForWriti
```

```
f.Write "Bonjour!"
```

```
f.Close
```

```
Set f = fso.OpenTextFile("c:\testfile.txt", ForReac
```

```
m = f.ReadLine
```

```
GetColumn = f.Column  
End Function
```

CompareMode, propriété

[Voir aussi](#)

[Application](#)

Description

Définit et renvoie le mode de comparaison pour comparer les clés de chaîne dans un objet **Dictionary**.

Syntaxe

object.**CompareMode**[= *compare*]

La propriété **CompareMode** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet Dictionary .
<i>compare</i>	Facultatif. Si elle est fournie, la valeur <i>compare</i> représente le mode de comparaison employé par diverses fonctions telles que StrComp .

Valeurs

L'argument *compare* peut prendre l'une des valeurs suivantes :

Constante	Valeur	Description
vbBinaryCompare	0	Effectuer une comparaison binaire.
vbTextCompare	1	Effectuer une comparaison texte.

Notes

Les valeurs supérieures à 2 peuvent être utilisées pour faire référence à des comparaisons utilisant des ID de paramètres locaux (LCID). Une erreur se produit si vous tentez de changer le mode de comparaison d'un objet **Dictionary** contenant déjà des données.

La propriété **CompareMode** utilise les mêmes valeurs que l'argument *compare* de la fonction **StrComp**.

L'exemple ci-dessous illustre l'utilisation de la propriété **CompareMode** :

```
Dim d, vbTextCompare
vbTextCompare = 1
Set d = CreateObject("Scripting.Dictionary")
d.CompareMode = vbTextCompare
d.Add "a", "Athènes"    ' Ajoute des clés et des éléments
d.Add "b", "Belgrade"
d.Add "c", "Casablanca"
d.Add "B", "Baltimore"  ' La méthode Add échoue
                        ' lettre b existe déjà dans Dictionary
```

Copy, méthode

[Voir aussi](#)

[Application](#)

Description

Copie un fichier ou un dossier spécifié d'un emplacement vers un autre.

Syntaxe

object.**Copy** *destination*[, *overwrite*]

La syntaxe de la méthode **Copy** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet File ou Folder .
<i>destination</i>	Destination de la copie du fichier ou du dossier. Les caractères génériques ne sont pas autorisés.
<i>overwrite</i>	Facultatif. Valeur booléenne True (par défaut) si les fichiers ou dossiers existants sont remplacés ; False sinon.

Notes

Le résultat de la méthode **Copy** sur un objet **File** ou **Folder** est identique à celui de **FileSystemObject.CopyFile** ou **FileSystemObject.CopyFolder** où le fichier ou le dossier indiqués par *object* est passé en argument. Notez cependant que ces méthodes alternatives sont capables de copier plusieurs fichiers ou dossiers.

L'exemple ci-dessous illustre l'utilisation de la méthode **Copy** :

Dim fso, MyFile

```
Set fso = CreateObject("Scripting.FileSystemObject")
Set MyFile = fso.CreateTextFile("c:\testfile.txt", True)
MyFile.WriteLine("Ceci est un test.")
MyFile.Close
Set MyFile = fso.GetFile("c:\testfile.txt")
MyFile.Copy ("c:\windows\desktop\test2.txt")
```

CopyFile, méthode

[Voir aussi](#)[Application](#)

Description

Copie un ou plusieurs fichiers d'un emplacement vers un autre.

Syntaxe

object.**CopyFile** *source*, *destination*[, *overwrite*]

La syntaxe de la méthode **CopyFile** comporte les éléments suivants :

Élément	Description
<i>object</i>	L'argument <i>object</i> représente toujours un objet FileSystemObject .
<i>source</i>	Chaîne de spécification de fichiers pouvant contenir des caractères génériques afin de copier un ou plusieurs fichiers.
<i>destination</i>	Chaîne de spécification de la destination de la copie du ou des fichiers issus de <i>source</i> . Les caractères génériques ne sont pas autorisés.
<i>overwrite</i>	Facultatif. Valeur de type Boolean indiquant si les fichiers existants sont remplacés. Si l'argument vaut True , les fichiers sont remplacés ; s'il vaut False , ils ne le sont pas. La valeur par défaut est True . Notez que CopyFile échoue si <i>destination</i> est en lecture seule, quelle que soit la valeur de <i>overwrite</i> .

Notes

Les caractères génériques sont autorisés uniquement pour le dernier composant de l'argument *source* argument. Par exemple, vous pouvez

utiliser :

```
FileSystemObject.CopyFile "c:\Documents\lettres\*.doc", "c:\temp"
```

Vous ne pouvez pas utiliser :

```
FileSystemObject.CopyFile "c:\Documents\*\R1???97.xls", "c:\temp"
```

Si l'argument *source* contient des caractères génériques ou si *destination* se termine par un séparateur de chemin (\), *destination* est considéré comme un dossier existant vers lequel s'effectue la copie des dossiers et sous-dossiers désignés. Sinon, *destination* est considéré comme le nom du dossier à créer. Dans les deux cas, la copie d'un dossier individuel présente trois possibilités.

- Si *destination* n'existe pas, le dossier *source* et son contenu sont copiés. C'est le cas le plus courant.
- Si *destination* est un fichier existant, une erreur se produit si *overwrite* est **False**. Sinon, la copie tente de remplacer le fichier *source* existant.
- Si *destination* est un dossier existant, une erreur se produit.

Une erreur se produit aussi si une *source* contenant des caractères génériques ne correspond à aucun dossier. La méthode **CopyFile** s'arrête sur la première erreur rencontrée. Aucune tentative n'est effectuée pour annuler les modifications précédant l'erreur.

CopyFolder, méthode

[Voir aussi](#)[Application](#)

Description

Copie récursivement un dossier d'un emplacement vers un autre.

Syntaxe

object.**CopyFolder** *source*, *destination*[, *overwrite*]

La syntaxe de la méthode **CopyFolder** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>source</i>	Chaîne de spécification de dossiers pouvant contenir des caractères génériques afin de copier un ou plusieurs dossiers.
<i>destination</i>	Chaîne de spécification de la destination de la copie du ou des dossiers et sous-dossiers issus de <i>source</i> . Les caractères génériques ne sont pas autorisés.
<i>overwrite</i>	Facultatif. Valeur de type Boolean indiquant si les dossiers existants sont remplacés. Si l'argument a la valeur True , les fichiers sont remplacés ; s'il prend la valeur False , ils ne le sont pas. La valeur par défaut est True .

Notes

Les caractères génériques sont autorisés uniquement pour le dernier composant de l'argument *source*. Par exemple, vous pouvez utiliser :

`FileSystemObject.CopyFolder "c:\Documents\lettres*", "c:\tempf`

Vous ne pouvez pas utiliser :

`FileSystemObject.CopyFolder "c:\Documents**", "c:\tempfolder`

Si l'argument *source* contient des caractères génériques ou si *destination* se termine par un séparateur de chemin (\), *destination* est considéré comme un dossier existant vers lequel s'effectue la copie des dossiers et sous-dossiers désignés. Sinon, *destination* est considéré comme le nom du dossier à créer. Dans les deux cas, la copie d'un dossier individuel présente quatre possibilités.

- Si *destination* n'existe pas, le dossier *source* et son contenu sont copiés. C'est le cas le plus courant.
- Si *destination* est un fichier existant, une erreur se produit.
- Si *destination* est un dossier, l'opération tente de copier le dossier et son contenu. Si un fichier contenu dans *source* existe déjà dans *destination*, une erreur se produit si *overwrite* est **False**. Sinon, la copie tente de remplacer le fichier existant.
- Si *destination* est un dossier en lecture seule, une erreur se produit si l'opération tente de copier un fichier existant en lecture seule dans ce dossier alors que *overwrite* vaut **False**.

Une erreur se produit aussi si une *source* contenant des caractères génériques ne correspond à aucun dossier.

La méthode **CopyFolder** s'arrête sur la première erreur rencontrée. Aucune tentative n'est effectuée pour annuler les modifications précédant l'erreur.

Count, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie le nombre d'éléments d'une collection ou d'un objet **Dictionary**.
Lecture seule.

Syntaxe

object.**Count**

L'argument *object* représente toujours le nom d'un élément de la liste Application.

Notes

Le code suivant illustre l'emploi de la propriété **Count** :

```
Function ShowKeys
    Dim a, d, i, s          ' Crée des variables
    Set d = CreateObject("Scripting.Dictionary")
    d.Add "a", "Athènes"   ' Ajoute des clés et des éléments.
    d.Add "b", "Belgrade"
    d.Add "c", "Casablanca"
    a = d.Keys             ' Lit les clés
    For i = 0 To d.Count -1 ' Parcoure le tableau
        s = s & a(i) & "<BR>" ' Crée la chaîne de retour
    Next
    ShowKeys = s
End Function
```

CreateFolder, méthode

[Voir aussi](#)[Application](#)

Description

Crée un dossier.

Syntaxe

object.**CreateFolder**(*foldername*)

La syntaxe de la méthode **CreateFolder** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>foldername</i>	Expression de chaîne identifiant le dossier à créer.

Notes

Une erreur se produit si le dossier spécifié existe déjà.

L'exemple ci-dessous illustre l'utilisation de la méthode **CreateFolder** :

Function CreateFolderDemo

```
Dim fso, f
```

```
Set fso = CreateObject("Scripting.FileSystemObj
```

```
Set f = fso.CreateFolder("c:\dossier")
```

```
CreateFolderDemo = f.Path
```

```
End, fonction
```



CreateTextFile, méthode

[Voir aussi](#)

[Application](#)

Description

Crée un nom de fichier spécifié et renvoie un objet **TextStream** pouvant être utilisé pour lire ou y écrire.

Syntaxe

object.**CreateTextFile**(*filename*[, *overwrite*[, *unicode*]])

La syntaxe de la méthode **CreateTextFile** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject ou Folder .
<i>filename</i>	Expression de chaîne identifiant le fichier à créer.
<i>overwrite</i>	Facultatif. Valeur de type Boolean indiquant si un fichier existant peut être remplacé. La valeur est True si le fichier peut être écrasé, False dans le cas contraire. Si cette valeur est omise, les fichiers existants ne peuvent pas être écrasés.
<i>unicode</i>	Facultatif. Valeur de type Boolean qui indique si le fichier est créé sous forme de fichier Unicode ou ASCII. La valeur est True si le fichier est créé sous forme de fichier Unicode, False s'il est créé sous forme de fichier ASCII. Si cette valeur est omise, un fichier ASCII est pris par défaut.

Notes

Le code suivant illustre l'emploi de la méthode **CreateTextFile** pour créer et ouvrir un fichier texte :

```
Sub CreateAfile
  Dim fso, MyFile
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set MyFile = fso.CreateTextFile("c:\testfile.txt", True)
  MyFile.WriteLine("Ceci est un test.")
  MyFile.Close
End Sub
```

Si l'argument *overwrite* vaut **False**, ou est absent, pour un *filename* déjà existant, une erreur se produit.

DateCreated, propriété

[Référence de la bibliothèque
d'exécution Scripting
Version 3](#)

[Voir aussi](#)

[Application](#)

Description

Renvoie la date et l'heure de création du fichier ou du dossier spécifié.
Lecture seule.

Syntaxe

object.**DateCreated**

L'argument *object* représente toujours un objet **File** ou **Folder**.

Notes

Le code suivant illustre l'emploi de la propriété **DateCreated** avec un fichier :

```
Function ShowFileInfo(filespec)
    Dim fso, f
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFile(filespec)
    ShowFileInfo = "Créé le: " & f.DateCreated
End Function
```

DateLastAccessed, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie la date et l'heure de dernier accès au fichier ou au dossier spécifié.
Lecture seule.

Syntaxe

object.**DateLastAccessed**

L'argument *object* représente toujours un objet **File** ou **Folder**.

Notes

Le code suivant illustre l'emploi de la propriété **DateLastAccessed** avec un fichier :

```
Function ShowFileAccessInfo(filespec)
    Dim fso, f, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFile(filespec)
    s = UCase(filespec) & "<BR>"
    s = s & "Créé le: " & f.DateCreated & "<BR>"
    s = s & "Dernier accès le: " & f.DateLastAccessed &
    "<BR>"
    s = s & "Dernière modification le: " & f.DateLastModified
    ShowFileAccessInfo = s
End Function
```

Important Le fonctionnement de cette méthode dépend du système d'exploitation sous-jacent. Les informations de date ne sont pas renvoyées si le système d'exploitation n'en fournit pas.

DateLastModified, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie la date et l'heure à laquelle le fichier ou le dossier spécifié a été modifié pour la dernière fois. Lecture seule.

Syntaxe

object.**DateLastModified**

L'argument *object* représente toujours un objet **File** ou **Folder**.

Notes

Le code suivant illustre l'emploi de la propriété **DateLastModified** avec un fichier :

```
Function ShowFileAccessInfo(filespec)
    Dim fso, f, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFile(filespec)
    s = UCase(filespec) & "<BR>"
    s = s & "Créé le: " & f.DateCreated & "<BR>"
    s = s & "Dernier accès le: " & f.DateLastAccessed & "<BR>"
    s = s & "Dernière modification le: " & f.DateLastModified
    ShowFileAccessInfo = s
End Function
```

Delete, méthode

[Voir aussi](#)

[Application](#)

Description

Supprime un fichier ou un dossier spécifié.

Syntaxe

object.**Delete** *force*

La syntaxe de la méthode **Delete** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet File ou Folder .
<i>force</i>	Facultatif. Valeur booléenne True si les fichiers ou dossiers en lecture seule sont supprimés, False (par défaut) dans le cas contraire.

Notes

Une erreur se produit si le fichier ou dossier spécifié n'existe pas. La méthode **Delete** ne fait pas de distinction entre les dossiers vides et les autres. Le dossier spécifié est supprimé qu'il ait un contenu ou pas.

Le résultat de la méthode **Delete** sur un objet **File** ou **Folder** est identique à celui de **FileSystemObject.DeleteFile** ou **FileSystemObject.DeleteFolder**.

L'exemple ci-dessous illustre l'utilisation de la méthode **Delete** :

```
Dim fso, MyFile
```

```
Set fso = CreateObject("Scripting.FileSystemObject")
```

```
Set MyFile = fso.CreateTextFile("c:\testfile.txt", True)
```

```
MyFile.WriteLine("Ceci est un test.")  
MyFile.Close  
Set MyFile = fso.GetFile("c:\testfile.txt")  
MyFile.Delete
```

DeleteFile, méthode

[Voir aussi](#)

[Application](#)

Description

Supprime un fichier spécifié.

Syntaxe

object.**DeleteFile** *filespec*[, *force*]

La syntaxe de la méthode **DeleteFile** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>filespec</i>	Correspond toujours au nom du fichier à supprimer. L'argument <i>filespec</i> peut contenir des caractères génériques dans le dernier composant du chemin.
<i>force</i>	Facultatif. Valeur booléenne True si les fichiers en lecture seule sont supprimés, False (par défaut) dans le cas contraire.

Notes

Une erreur se produit si aucun fichier correspondant n'est trouvé. La méthode **DeleteFile** s'arrête sur la première erreur rencontrée. Aucune tentative n'est effectuée pour restaurer l'état précédent ou annuler les modifications précédant l'erreur.

L'exemple ci-dessous illustre l'utilisation de la méthode **DeleteFile** :

```
Sub DeleteAFile(filespec)
  Dim fso
  Set fso = CreateObject("Scripting.FileSystemObj
  fso.DeleteFile(filespec)
End Sub
```

DeleteFolder, méthode

[Voir aussi](#)

[Application](#)

Description

Supprime un dossier spécifié et son contenu.

Syntaxe

object.**DeleteFolder** *folderspec* [, *force*]

La syntaxe de la méthode **DeleteFolder** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>folderspec</i>	Correspond toujours au nom du dossier à supprimer. L'argument <i>folderspec</i> peut contenir des caractères génériques dans le dernier composant du chemin.
<i>force</i>	Facultatif. Valeur booléenne True si les dossiers en lecture seule sont supprimés, False (par défaut) dans le cas contraire.

Notes

La méthode **DeleteFolder** ne fait pas de distinction entre les dossiers vides et les autres. Le dossier spécifié est supprimé qu'il ait un contenu ou pas.

Une erreur se produit si aucun dossier correspondant n'est trouvé. La méthode **DeleteFolder** s'arrête sur la première erreur rencontrée. Aucune tentative n'est effectuée pour restaurer l'état précédent ou annuler les modifications précédant l'erreur.

L'exemple ci-dessous illustre l'utilisation de la méthode **DeleteFolder** :

```
Sub DeleteAFolder(filespec)
  Dim fso
  Set fso = CreateObject("Scripting.FileSystemObj
  fso.DeleteFolder(filespec)
End Sub
```

Dictionary, objet

[Voir aussi](#)[Propriétés](#)[Méthodes](#)

Description

Objet qui stocke des paires clé/élément de données.

Notes

Un objet **Dictionary** est l'équivalent d'un tableau associatif PERL. Les éléments, pouvant correspondre à n'importe quelle forme de données, sont stockés dans le tableau. Chaque élément est associé à une clé qui lui est propre. La clé est utilisée pour extraire un élément individuel et correspond généralement à un nombre entier ou à une chaîne, mais peut être n'importe quelle information à l'exception d'un tableau.

Le code suivant illustre la création d'un objet **Dictionary** :

```
Dim d           ' Crée une variable
Set d = CreateObject("Scripting.Dictionary")
d.Add "a", "Athènes"   ' Ajoute des clés et des élé
d.Add "b", "Belgrade"
d.Add "c", "Casablanca"
...
```

Drive, objet

[Voir aussi](#)

[Propriétés](#)

[Méthodes](#)

Description

Donne accès aux propriétés d'un lecteur de disque ou d'un partage réseau spécifié.

Notes

Le code suivant illustre l'emploi de l'objet **Drive** pour accéder aux propriétés de lecteur :

```
Function ShowFreeSpace(drvPath)
    Dim fso, d, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set d = fso.GetDrive(fso.GetDriveName(drvPath))
    s = "Lecteur " & UCase(drvPath) & " - "
    s = s & d.VolumeName & "<BR>"
    s = s & "Espace disponible: " & FormatNumber(d.FreeSpace/102
    s = s & " KOctets"
    ShowFreeSpace = s
End Function
```

Drive, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie la lettre du lecteur sur lequel réside le fichier ou le dossier spécifié.
Lecture seule.

Syntaxe

object.**Drive**

L'argument *object* représente toujours un objet **File** ou **Folder**.

Notes

Le code suivant illustre l'emploi de la propriété **Drive** :

```
<Function ShowFileAccessInfo(filespec)
  Dim fso, f, s
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set f = fso.GetFile(filespec)
  s = f.Name & " sur le lecteur " & UCase(f.Drive) & "<BR>"
  s = s & "Créé le: " & f.DateCreated & "<BR>"
  s = s & "Dernier accès le: " & f.DateLastAccessed & "<BR>"
  s = s & "Dernière modification le: " & f.DateLastModified
  ShowFileAccessInfo = s
End Function
```

DriveExists, méthode

[Voir aussi](#)

[Application](#)

Description

Renvoie la valeur **True** si le lecteur spécifié existe, **False** dans le cas contraire.

Syntaxe

object.**DriveExists**(*drivespec*)

La syntaxe de la méthode **DriveExists** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>drivespec</i>	Une lettre désignant un lecteur ou une spécification de chemin complète.

Notes

Pour les lecteurs de supports amovibles, la méthode **DriveExists** renvoie **True** même si le lecteur ne contient pas de support. Utilisez la méthode **IsReady** de l'objet **Drive** pour déterminer si le lecteur est prêt.

L'exemple ci-dessous illustre l'utilisation de la méthode **DriveExists** :

```
Function ReportDriveStatus(drv)
    Dim fso, msg
    Set fso = CreateObject("Scripting.FileSystemObject")
    If fso.DriveExists(drv) Then
```

```
    msg = ("Le lecteur " & UCase(drv) & " existe.")  
Else  
    msg = ("Le lecteur " & UCase(drv) & " n'existe pas.")  
End If  
ReportDriveStatus = msg  
End, fonction
```

DriveLetter, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie la lettre d'un lecteur local physique ou d'un partage réseau. Lecture seule.

Syntaxe

object.**DriveLetter**

L'argument *object* représente toujours un objet **Drive**.

Notes

La propriété **DriveLetter** renvoie une chaîne de longueur nulle ("") si le lecteur spécifié n'est pas associé avec une lettre de lecteur comme, par exemple, un partage réseau qui n'est pas mappé sur une lettre de lecteur.

Le code suivant illustre l'emploi de la propriété **DriveLetter** :

```
Function ShowDriveLetter(drvPath)
    Dim fso, d, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set d = fso.GetDrive(fso.GetDriveName(drvPath))
    s = "Lecteur " & d.DriveLetter & ": - "
    s = s & d.VolumeName & "<BR>"
    s = s & "Espace disponible: " & FormatNumber(d.FreeSpace/102
    s = s & " KOctets""
    ShowDriveLetter = s
End Function
```

Drives, collection

[Voir aussi](#)

[Propriétés](#)

[Méthodes](#)

Description

[Collection](#) en lecture seule de tous les lecteurs disponibles.

Notes

Les lecteurs de supports amovibles apparaissent dans la collection **Drives** même lorsqu'ils sont vides.

Le code suivant illustre la lecture de la collection **Drives** et son itération à l'aide de l'instruction **For Each...Next** :

```
Function ShowDriveList
    Dim fso, d, dc, s, n
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set dc = fso.Drives
    For Each d in dc
        n = ""
        s = s & d.DriveLetter & " - "
        If d.DriveType = Remote Then
            n = d.ShareName
        ElseIf d.IsReady Then
            n = d.VolumeName
        End If
        s = s & n & "<BR>"
    Next
    ShowDriveList = s
End Function
```

Drives, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie une collection **Drives** constituée de tous les objets **Drive** disponibles sur la machine locale.

Syntaxe

object.**Drives**

L'argument *object* représente toujours un objet **FileSystemObject**.

Notes

Les lecteurs de supports amovibles apparaissent dans la collection **Drives** même lorsqu'ils sont vides.

Vous pouvez parcourir les membres de la collection **Drives** en utilisant une structure **For Each...Next** comme illustré dans le code suivant :

```
Function ShowDriveList
    Dim fso, d, dc, s, n
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set dc = fso.Drives
    For Each d in dc
        n = ""
        s = s & d.DriveLetter & " - "
        If d.DriveType = 3 Then
            n = d.ShareName
        ElseIf d.IsReady Then
            n = d.VolumeName
        End If
    Next d
    Print s & n
End Function
```

```
End If
s = s & n & "<BR>"
Next
ShowDriveList = s
End Function
```

DriveType, constantes

[Voir aussi](#)

Ces constantes sont disponibles uniquement lorsque votre projet contient une référence explicite à la [bibliothèque de types](#) contenant leurs définitions. Pour VBScript, vous devez déclarer ces constantes explicitement dans votre code.

Constante	Valeur	Description
Unknown	0	Le type du lecteur ne peut pas être déterminé.
Removable	1	Le lecteur contient un support amovible. Ceci comprend tous les lecteurs de disquette et de nombreux autres périphériques de stockage.
Fixed	2	Le lecteur contient un support permanent (non amovible). Ceci comprend tous les lecteurs de disques durs, y compris ceux qui sont amovibles.
Remote	3	Lecteurs de réseau. Ceci comprend tous les lecteurs partagés partout dans le réseau.
CDROM	4	Le lecteur est un CD-ROM. Aucune distinction n'est effectuée entre les lecteurs de CD-ROM en lecture seule et en lecture/écriture.
RAMDisk	5	Le lecteur est en fait un bloc de mémoire RAM de l'ordinateur local qui se comporte comme un lecteur de disque.

DriveType, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie une valeur indiquant le type d'un lecteur spécifié.

Syntaxe

object.**DriveType**

L'argument *object* représente toujours un objet **Drive**.

Notes

Le code suivant illustre l'emploi de la propriété **DriveType** :

```
Function ShowDriveType(drvpath)
    Dim fso, d, t
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set d = fso.GetDrive(drvpath)
    Select Case d.DriveType
        Case 0: t = "Inconnu"
        Case 1: t = "Amovible"
        Case 2: t = "Fixe"
        Case 3: t = "Réseau"
        Case 4: t = "CD-ROM"
        Case 5: t = "RAM Disk"
    End Select
    ShowDriveType = "Lecteur " & d.DriveLetter & ": - " & t
End Function
```



Exists, méthode

[Voir aussi](#)

[Application](#)

Description

Renvoie la valeur **True** si une clé spécifiée existe dans l'objet **Dictionary**.
La valeur **False** est renvoyée dans le cas contraire.

Syntaxe

object.**Exists**(*key*)

La syntaxe de la méthode **Exists** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet Dictionary .
<i>key</i>	Requis. Valeur <i>Key</i> recherchée dans l'objet Dictionary .

Notes

L'exemple ci-dessous illustre l'utilisation de la méthode **Exists** :

Function KeyExistsDemo

```
Dim d, msg          ' Crée des variables.  
Set d = CreateObject("Scripting.Dictionary")  
d.Add "a", "Athènes" ' Ajoute des clés et des éléments.  
d.Add "b", "Belgrade"  
d.Add "c", "Casablanca"  
If d.Exists("c") Then  
    msg = "La clé spécifiée existe."  
Else
```

```
    msg = "La clé spécifiée n'existe pas."  
End If  
KeyExistsDemo = msg  
End, fonction
```

Attributs de fichier, constantes

[Voir aussi](#)

Ces constantes sont disponibles uniquement lorsque votre projet contient une référence explicite à la [bibliothèque de types](#) contenant leurs définitions. Pour VBScript, vous devez déclarer ces constantes explicitement dans votre code.

Constante	Valeur	Description
Normal	0	Fichier normal. Aucun attribut n'est défini.
ReadOnly	1	Fichier en lecture seule.
Hidden	2	Fichier caché.
System	4	Fichier système.
Directory	16	Dossier ou répertoire.
Archive	32	Le fichier a été modifié depuis la dernière sauvegarde.
Alias	1024	Lien ou raccourci.
Compressed	2048	Fichier compressé.

Entrée/Sortie de fichier, constantes

[Voir aussi](#)

Ces constantes sont disponibles uniquement lorsque votre projet contient une référence explicite à la [bibliothèque de types](#) contenant leurs définitions. Pour VBScript, vous devez déclarer ces constantes explicitement dans votre code.

Constante	Valeur	Description
ForReading	1	Ouvrir un fichier en lecture seule. Vous ne pouvez pas écrire dans ce fichier.
ForWriting	2	Ouvrir un fichier en écriture. Si un fichier existe sous le même nom, son contenu est écrasé.
ForAppending	8	Ouvrir un fichier et écrire à la fin du fichier.

FileExists, méthode

[Voir aussi](#)

[Application](#)

Description

Renvoie la valeur **True** si un fichier spécifié existe La valeur **False** est renvoyée dans le cas contraire.

Syntaxe

object.**FileExists**(*filespec*)

La syntaxe de la méthode **FileExists** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>filespec</i>	Le nom du fichier dont l'existence doit être déterminée. Une spécification de chemin complet (absolue ou relative) est nécessaire si le fichier ne se trouve pas dans le dossier en cours.

Notes

L'exemple ci-dessous illustre l'utilisation de la méthode **FileExists** :

```
Function ReportFileStatus(filespec)
    Dim fso, msg
    Set fso = CreateObject("Scripting.FileSystemObject")
    If (fso.FileExists(filespec)) Then
        msg = filespec & " existe."
    Else
```

```
    msg = filespec & " n'existe pas."  
End If  
ReportFileStatus = msg  
End, fonction
```

File, objet

[Voir aussi](#)

[Propriétés](#)

[Méthodes](#)

Description

Donne accès aux propriétés d'un fichier.

Notes

Le code suivant illustre l'obtention d'un objet **File** et l'affichage d'une de ses propriétés.

```
Function ShowDateCreated(filespec)
    Dim fso, f
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFile(filespec)
    ShowDateCreated = f.DateCreated
End Function
```

Files, collection

[Voir aussi](#)

[Propriétés](#)

[Méthodes](#)

Description

[Collection](#) de tous les objets **File** d'un dossier.

Notes

Le code suivant illustre la lecture de la collection **Files** et son itération à l'aide de l'instruction **For Each...Next** :

```
Function ShowFolderList(folderspec)
    Dim fso, f, f1, fc, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFolder(folderspec)
    Set fc = f.Files
    For Each f1 in fc
        s = s & f1.name
        s = s & "<BR>"
    Next
    ShowFolderList = s
End Function
```

Files, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie une collection **Files** constituée de tous les objets **File** contenus dans le dossier spécifié, y compris ceux qui sont cachés ou système.

Syntaxe

object.**Files**

L'argument *object* représente toujours un objet **Folder**.

Notes

Le code suivant illustre l'emploi de la propriété **Files** :

```
Function ShowFileList(folderspec)
    Dim fso, f, f1, fc, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFolder(folderspec)
    Set fc = f.Files
    For Each f1 in fc
        s = s & f1.name
        s = s & "<BR>"
    Next
    ShowFileList = s
End Function
```

FileSystemObject, objet

[Référence de la bibliothèque
d'exécution Scripting](#)
[Version 2](#)

[Voir aussi](#)

[Propriétés](#)

[Méthodes](#)

Description

Donne accès au système de fichiers d'un ordinateur.

Notes

Le code suivant illustre comment l'objet **FileSystemObject** est employé pour renvoyer un objet **TextStream** pouvant être lu, ou dans lequel il est possible d'écrire des informations :

```
Dim fso, MyFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set MyFile = fso.CreateTextFile("c:\testfile.txt", True)
MyFile.WriteLine("Ceci est un test.")
MyFile.Close
```

Dans le code ci-dessus, la fonction **CreateObject** renvoie l'objet **FileSystemObject** (fso). La méthode **CreateTextFile** crée ensuite le fichier sous forme d'objet **TextStream** (a) et la méthode **WriteLine** écrit une ligne de texte dans le fichier texte créé. La méthode **Close** vide le tampon et ferme le fichier.

FileSystemObject, constantes

[Référence de la bibliothèque
d'exécution Scripting
Version 2](#)

[Voir aussi](#)

FileSystemObject possède de nombreuses constantes très utiles que vous pouvez employer dans l'écriture de votre code. Les constantes sont un moyen pratique d'utiliser des valeurs spécifiques sans qu'il soit nécessaire de se souvenir de celles-ci. Elles rendent également votre code plus facile à entretenir, lorsque la valeur d'une constante est modifiée.

En fonction de votre hôte de script, les constantes peuvent éventuellement être déjà définies. Si tel est le cas, il suffit d'utiliser les constantes dans votre code à la place des valeurs qu'elles représentent. Lorsque votre hôte de script ne fait pas explicitement référence au fichier SCRRUN.DLL, vous devrez définir ces constantes dans votre code avant de pouvoir les utiliser. C'est le cas avec Microsoft Internet Explorer et Microsoft Internet; Information; Services; (IIS).

La liste suivante répertorie les différentes catégories de constantes fournies avec **FileSystemObject** et les accompagne d'une brève description :

[Drive Type, constantes](#)

Définissent les différents types de lecteurs disponibles sur l'ordinateur hôte, tels que les lecteurs fixes, amovibles, de CD-ROM, etc.

[Attributs de fichier, constantes](#)

Définissent les différents attributs de fichier (Caché, Lecture seule, etc).

[Entrée/Sortie de fichier, constantes](#)

Définissent des constantes utilisées avec des entrées et des sorties de fichier.

[SpecialFolder, constantes](#)

Définissent les dossiers spéciaux disponibles sur votre système d'exploitation.

FileSystem, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie le type de système de fichiers du lecteur spécifié.

Syntaxe

object.**FileSystem**

L'argument *object* représente toujours un objet **Drive**.

Notes

Les types de retour disponibles sont FAT, NTFS et CDFS.

Le code suivant illustre l'utilisation de la propriété **FileSystem** :

```
Function ShowFileSystemType(drvspec)
    Dim fso,d
    Set fso = CreateObject("Scripting.FileSystemObj
    Set d = fso.GetDrive(drvspec)
    ShowFileSystemType = d.FileSystem
End Function
```

Folder, objet

[Voir aussi](#)

[Propriétés](#)

[Méthodes](#)

Description

Donne accès aux propriétés d'un dossier.

Notes

Le code suivant illustre l'obtention d'un objet **Folder** et comment renvoyer une de ses propriétés :

```
Function ShowDateCreated(folderspec)
    Dim fso, f
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFolder(folderspec)
    ShowDateCreated = f.DateCreated
End Function
```

Folders, collection

[Voir aussi](#)

[Propriétés](#)

[Méthodes](#)

Description

[Collection](#) de tous les objets **Folder** d'un objet **Folder**.

Notes

Le code suivant illustre la lecture de la collection **Folders** et son itération à l'aide de l'instruction **For Each...Next** :

```
Function ShowFolderList(folderspec)
    Dim fso, f, f1, fc, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFolder(folderspec)
    Set fc = f.SubFolders
    For Each f1 in fc
        s = s & f1.name
        s = s & "<BR>"
    Next
    ShowFolderList = s
End Function
```

FolderExists, méthode

[Voir aussi](#)

[Application](#)

Description

Renvoie la valeur **True** si le dossier spécifié existe et la valeur **False** dans le cas contraire.

Syntaxe

object.**FolderExists**(*folderspec*)

La syntaxe de la méthode **FolderExists** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>folderspec</i>	Le nom du dossier dont l'existence doit être déterminée. Une spécification de chemin complet (absolue ou relative) est nécessaire si le dossier ne se trouve pas dans le dossier en cours.

Notes

L'exemple ci-dessous illustre l'utilisation de la méthode **FolderExists** :

```
Function ReportFolderStatus(fldr)
  Dim fso, msg
  Set fso = CreateObject("Scripting.FileSystemObject")
  If (fso.FolderExists(fldr)) Then
    msg = fldr & " existe."
  Else
```

```
    msg = fldr & " n'existe pas."  
End If  
ReportFolderStatus = msg  
End, fonction
```

FreeSpace, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie la quantité d'espace disponible pour un utilisateur sur le lecteur ou le partage réseau. Lecture seule.

Syntaxe

object.FreeSpace

L'argument *object* représente toujours un objet **Drive**.

Notes

La valeur renvoyée par la propriété **FreeSpace** est en général la même que celle renvoyée par la propriété **AvailableSpace**. Elles peuvent présenter des différences dans le cas des systèmes d'ordinateur qui gèrent les quotas.

Le code suivant illustre l'emploi de la propriété **FreeSpace** :

```
Function ShowFreeSpace(drvPath)
    Dim fso, d, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set d = fso.GetDrive(fso.GetDriveName(drvPath))
    s = "Lecteur " & UCase(drvPath) & " - "
    s = s & d.VolumeName & "<BR>"
    s = s & "Espace disponible: "" & FormatNumber(d.FreeSpace/10
0)
    s = s & " KOctets"
    ShowFreeSpace = s
```

End Function

GetAbsolutePathName, méthode

[Référence de la bibliothèque d'exécution Scrip](#)
[Versi](#)

[Voir aussi](#)

[Application](#)

Description

Renvoie un chemin complet et non ambigu à partir d'une spécification de chemin fournie.

Syntaxe

object.**GetAbsolutePathName**(*pathspec*)

La syntaxe de la méthode **GetAbsolutePathName** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>pathspec</i>	Spécification de chemin à changer en chemin complet et non ambigu.

Notes

Un chemin est complet et non ambigu s'il fournit une référence complète à partir de la racine du lecteur spécifié. Un chemin complet peut se terminer avec un caractère séparateur (\) uniquement s'il spécifie le chemin racine d'un lecteur mappé.

En supposant que le répertoire en cours est c:\documents\rapports, le tableau suivant illustre le fonctionnement de la méthode **GetAbsolutePathName**.

<i>pathspec</i>	Chemin renvoyé
"c:"	"c:\documents\rapports"
"c:.."	"c:\documents"

"c:\\\"	"c:\"
"c:*. *\mai97"	"c:\documents\rapports*. *\mai97"
"région1"	"c:\documents\rapports\région1"
"c:\..\..\documents"	"c:\documents"

GetBaseName, méthode

[Voir aussi](#)[Application](#)

Description

Renvoie une chaîne contenant le nom de base (sans extension) du fichier ou du dossier dans une spécification de chemin fournie.

Syntaxe

object.**GetBaseName**(*path*)

La syntaxe de la méthode **GetBaseName** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>path</i>	La spécification de chemin pour le fichier ou le dossier dont le nom de base doit être renvoyé.

Notes

La méthode **GetBaseName** renvoie une chaîne de longueur nulle ("") si aucun fichier ou dossier ne correspond à l'argument *path*.

L'exemple ci-dessous illustre l'utilisation de la méthode **GetBaseName** :

```
Function GetTheBase(filespec)
```

```
    Dim fso
```

```
    Set fso = CreateObject("Scripting.FileSystemObj
```

```
    GetTheBase = fso.GetBaseName(filespec)
```

End, fonction

Remarque La méthode **GetBaseName** travaille uniquement sur la chaîne *path* fournie. Elle ne tente pas de résoudre le chemin ni de vérifier son existence.

GetDrive, méthode

[Voir aussi](#)[Application](#)

Description

Renvoie un objet **Drive** correspondant au lecteur figurant dans un chemin spécifié.

Syntaxe

object.**GetDrive** *drivespec*

La syntaxe de la méthode **GetDrive** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>drivespec</i>	L'argument <i>drivespec</i> peut être une lettre de lecteur (c), une lettre de lecteur avec deux points (c:), une lettre de lecteur avec deux points et un séparateur (c:\) ou toute spécification de partage réseau (\\ordinateur2\partage1).

Notes

L'existence des partages réseau est contrôlée.

Une erreur se produit si l'argument *drivespec* n'est pas conforme à un des formats acceptés ou n'existe pas. Pour appeler la méthode **GetDrive** sur une chaîne de chemin normale, utilisez la séquence suivante pour obtenir une chaîne utilisable comme argument *drivespec*:

```
DriveSpec = GetDriveName(GetAbsolutePathName(Path))
```

L'exemple ci-dessous illustre l'utilisation de la méthode **GetDrive** :

```
Function ShowFreeSpace(drvPath)
  Dim fso, d, s
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set d = fso.GetDrive(fso.GetDriveName(drvPath))
  s = "Lecteur " & UCase(drvPath) & " - "
  s = s & d.VolumeName & "<BR>"
  s = s & "Espace disponible: " & FormatNumber(d.FreeSpace/102
  s = s & " KOctets"
  ShowFreeSpace = s
End Function
```

GetDriveName, méthode

[Voir aussi](#)[Application](#)

Description

Renvoie une chaîne contenant le nom du lecteur pour un chemin spécifié.

Syntaxe

object.**GetDriveName**(*path*)

La syntaxe de la méthode **GetDriveName** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>path</i>	La spécification de chemin pour le composant dont le nom de lecteur doit être renvoyé.

Notes

La méthode **GetDriveName** renvoie une chaîne de longueur nulle ("") si le lecteur ne peut pas être déterminé.

L'exemple ci-dessous illustre l'utilisation de la méthode **GetDriveName** :

```
Function GetAName(DriveSpec)
    Dim fso
    Set fso = CreateObject("Scripting.FileSystemObj
    GetAName = fso.GetDriveName(Drivespec)
End Function
```

Remarque La méthode **GetDriveName** travaille uniquement sur la chaîne *path* fournie. Elle ne tente pas de résoudre le chemin ni de vérifier son existence.

GetExtensionName, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)
[Version 3](#)

[Voir aussi](#)

[Application](#)

Description

Renvoie une chaîne contenant le nom de l'extension du dernier composant d'un chemin.

Syntaxe

object.**GetExtensionName**(*path*)

La syntaxe de la méthode **GetExtensionName** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>path</i>	La spécification de chemin pour le composant dont le nom d'extension doit être renvoyé.

Notes

Pour les lecteurs réseau, le répertoire racine (\) est considéré comme un composant.

La méthode **GetExtensionName** renvoie une chaîne de longueur nulle ("") si aucun composant ne correspond à l'argument *path*.

L'exemple ci-dessous illustre l'utilisation de la méthode **GetExtensionName** :

```
Function GetAnExtension(DriveSpec)  
    Dim fso
```

```
Set fso = CreateObject("Scripting.FileSystemObject")
GetAnExtension = fso.GetExtensionName(DriveLetter & "\")
End Function
```

GetFile, méthode

[Voir aussi](#)

[Application](#)

Description

Renvoie un objet **File** correspondant au fichier dans un chemin spécifié.

Syntaxe

object.**GetFile**(*filespec*)

La syntaxe de la méthode **GetFile** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>filespec</i>	L'argument <i>filespec</i> est le chemin (absolu ou relatif) d'un fichier spécifique.

Notes

Une erreur se produit si le fichier spécifié n'existe pas.

L'exemple ci-dessous illustre l'utilisation de la méthode **GetFile** :

```
Function ShowFileInfo(filespec)
```

```
    Dim fso, f, s
```

```
    Set fso = CreateObject("Scripting.FileSystemObject")
```

```
    Set f = fso.GetFile(filespec)
```

```
    s = f.Path & "<br>"
```

```
    s = s & "Créé le: " & f.DateCreated & "<br>"
```

```
    s = s & "Dernier accès le: " & f.DateLastAccessed & "<br>"
```

```
    s = s & "Dernière modification le: " & f.DateLastModified
```

```
ShowFileAccessInfo = s  
End Function
```

GetFileName, méthode

[Voir aussi](#)

[Application](#)

Description

Renvoie le dernier nom de fichier ou dossier d'un chemin spécifié qui ne fait pas partie de la spécification de lecteur.

Syntaxe

object.**GetFileName**(*pathspec*)

La syntaxe de la méthode **GetFileName** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>pathspec</i>	Le chemin (absolu ou relatif) vers un fichier spécifique.

Notes

La méthode **GetFileName** renvoie une chaîne de longueur nulle ("") si *pathspec* ne se termine pas par le fichier ou dossier nommé.

L'exemple ci-dessous illustre l'utilisation de la méthode **GetFileName** :

```
Function GetAName(DriveSpec)
```

```
    Dim fso
```

```
    Set fso = CreateObject("Scripting.FileSystemObj
```

```
    GetAName = fso.GetFileName(DriveSpec)
```

End Function

Remarque La méthode **GetFileName** travaille uniquement sur la chaîne de chemin fournie. Elle ne tente pas de résoudre le chemin ni de vérifier son existence.

GetFileVersion, méthode

[Voir aussi](#)

[Application](#)

Description

Renvoie le numéro de version du fichier spécifié.

Syntaxe

object.**GetFileVersion**(*pathspec*)

La syntaxe de la méthode **GetVersion** comprend les éléments suivants :

Élément	Description
<i>object</i>	Requis. Correspond toujours au nom d'un objet FileSystemObject .
<i>pathspec</i>	Requis. Le chemin (absolu ou relatif) vers un fichier spécifique.

Notes

La méthode **GetFileVersion** renvoie une chaîne de longueur nulle ("") si *pathspec* ne se termine pas par le fichier nommé ou que le fichier ne contient aucune information de version.

L'exemple ci-dessous illustre l'utilisation de la méthode **GetFileVersion** :

```
Function GetVersion(PathSpec)
```

```
    Dim fso, temp
```

```
    Set fso = CreateObject("Scripting.FileSystemObj
```

```
    temp = fso.GetFileVersion(PathSpec)
```

```
If Len(temp) Then
  GetVersion = temp
Else
  GetVersion = "Aucune information de version di
End If
End Function
```

Remarque La méthode **GetFileVersion** travaille uniquement sur la chaîne de chemin fournie. Elle ne tente pas de résoudre le chemin ni de vérifier son existence.

GetFolder, méthode

[Voir aussi](#)

[Application](#)

Description

Renvoie un objet **Folder** correspondant au dossier d'un chemin spécifié.

Syntaxe

object.**GetFolder**(*folderspec*)

La syntaxe de la méthode **GetFolder** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>folderspec</i>	L'argument <i>folderspec</i> est le chemin (absolu ou relatif) vers un dossier spécifique.

Notes

Une erreur se produit si le dossier spécifié n'existe pas.

L'exemple ci-dessous illustre l'utilisation de la méthode **GetFolder** pour renvoyer un objet Folder :

```
Sub AddNewFolder(path, folderName)
  Dim fso, f, fc, nf
  Set fso = CreateObject("Scripting.FileSystemObj
  Set f = fso.GetFolder(path)
  Set fc = f.SubFolders
```

```
If folderName <> "" Then
  Set nf = fc.Add(folderName)
Else
  Set nf = fc.Add("Dossier")
End If
End Sub
```

GetParentFolderName, méthode

[Référence de la bibliothèque
d'exécution Scrip
Versi](#)

[Voir aussi](#)

[Application](#)

Description

Renvoie une chaîne contenant le nom du dossier parent du dernier fichier ou dossier dans un chemin spécifié.

Syntaxe

object.**GetParentFolderName**(*path*)

La syntaxe de la méthode **GetParentFolderName** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>path</i>	La spécification de chemin pour le fichier ou dossier dont le nom de dossier parent doit être renvoyé.

Notes

La méthode **GetParentFolderName** renvoie une chaîne de longueur nulle ("") s'il n'y a pas de dossier parent pour le fichier ou dossier spécifié dans l'argument *path*.

L'exemple ci-dessous illustre l'utilisation de la méthode **GetParentFolderName** :

```
Function GetTheParent(DriveSpec)
```

```
    Dim fso
```

```
    Set fso = CreateObject("Scripting.FileSystemObj
```

```
GetTheParent = fso.GetParentFolderName(Drive:  
End Function
```

Remarque La méthode **GetParentFolderName** travaille uniquement sur la chaîne *path* fournie. Elle ne tente pas de résoudre le chemin ni de vérifier son existence.

GetSpecialFolder, méthode

[Voir aussi](#)

[Application](#)

Description

Renvoie le dossier spécial spécifié.

Syntaxe

object.**GetSpecialFolder**(*folderspec*)

La syntaxe de la méthode **GetSpecialFolder** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>folderspec</i>	Le nom du dossier spécial à renvoyer. Ce nom peut être toute constante de la section Valeurs.

Valeurs

L'argument *folderspec* peut prendre les valeurs suivantes :

Constante	Valeur	Description
WindowsFolder	0	Le dossier Windows contient des fichiers installés par le système d'exploitation Windows.
SystemFolder	1	Le dossier Système contient les bibliothèques, les polices et les pilotes de périphérique.
TemporaryFolder	2	Le dossier Temp stocke les fichiers temporaires. Son chemin figure dans

|||la variable d'environnement TMP. |||

Notes

L'exemple ci-dessous illustre l'utilisation de la méthode **GetSpecialFolder** :

```
Dim fso, tempfile
Set fso = CreateObject("Scripting.FileSystemObject")

Function CreateTempFile
    Dim tfolder, tname, tfile
    Const TemporaryFolder = 2
    Set tfolder = fso.GetSpecialFolder(TemporaryFolder)
    tname = fso.GetTempName
    Set tfile = tfolder.CreateTextFile(tname)
    Set CreateTempFile = tfile
End Function

Set tempfile = CreateTempFile
tempfile.WriteLine "Bonjour"
tempfile.Close
```

GetTempName, méthode

[Voir aussi](#)

[Application](#)

Description

Renvoie un nom de fichier ou de dossier temporaire généré aléatoirement utile pour les opérations qui nécessitent un fichier ou un dossier temporaire.

Syntaxe

object.**GetTempName**

L'argument *object* facultatif est toujours le nom d'un **FileSystemObject**.

Notes

La méthode **GetTempName** ne crée pas de fichier. Elle fournit un nom de fichier temporaire que vous pouvez utiliser avec **CreateTextFile** pour créer un fichier.

L'exemple ci-dessous illustre l'utilisation de la méthode **GetTempName** :

```
Dim fso, tempfile  
Set fso = CreateObject("Scripting.FileSystemObject")
```

```
Function CreateTempFile
```

```
    Dim tfolder, tname, tfile
```

```
    Const TemporaryFolder = 2
```

```
    Set tfolder = fso.GetSpecialFolder(TemporaryFolder)
```

```
    tname = fso.GetTempName
```

```
Set tfile = tfolder.CreateTextFile(tname)
Set CreateTempFile = tfile
End Function
```

```
Set tempfile = CreateTempFile
tempfile.WriteLine "Bonjour"
tempfile.Close
```

IsReady, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie la valeur **True** si le lecteur spécifié est prêt et la valeur **False** dans le cas contraire.

Syntaxe

object.**IsReady**

L'argument *object* représente toujours un objet **Drive**.

Notes

Pour les lecteurs de support amovible et les lecteurs de CD-ROM, **IsReady** renvoie **True** uniquement lorsque le support approprié est présent et prêt.

Le code suivant illustre l'emploi de la propriété **IsReady** :

```
Function ShowDriveInfo(drvpath)
    Dim fso, d, s, t
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set d = fso.GetDrive(drvpath)
    Select Case d.DriveType
        Case 0: t = "Inconnu"
        Case 1: t = "Amovible"
        Case 2: t = "Fixe"
        Case 3: t = "Réseau"
        Case 4: t = "CD-ROM"
        Case 5: t = "RAM Disk"
    End Select
```

```
s = "Lecteur "" & d.DriveLetter & ":- " & t
If d.IsReady Then
    s = s & "<BR>" & "Lecteur prêt."
Else
    s = s & "<BR>" & "Lecteur non prêt."
End If
ShowDriveInfo = s
End Function
```

IsRootFolder, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie la valeur **True** si le dossier spécifié est le dossier racine et la valeur **False** dans le cas contraire.

Syntaxe

object.**IsRootFolder**

L'argument *object* représente toujours un objet **Folder**.

Notes

Le code suivant illustre l'emploi de la propriété **IsRootFolder** :

```
Function DisplayLevelDepth(pathspec)
    Dim fso, f, n
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFolder(pathspec)
    If f.IsRootFolder Then
        DisplayLevelDepth = "Le dossier spécifié est le dossier racine."
    Else
        Do Until f.IsRootFolder
            Set f = f.ParentFolder
            n = n + 1
        Loop
        DisplayLevelDepth = "Le dossier spécifié se trouve " & n & " n
    End If
```

End Function

Item, propriété

[Voir aussi](#)

[Application](#)

Description

Définit ou renvoie un *élément* pour un argument *key* spécifié dans un objet **Dictionary**. Pour les collections, cette propriété renvoie un *item* basé sur la *key* spécifiée. Lecture/écriture.

Syntaxe

object.**Item**(*key*) [= *newitem*]

La propriété **Item** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'une collection ou d'un objet Dictionary .
<i>key</i>	Key associée avec l'élément <i>item</i> lu ou ajouté.
<i>newitem</i>	Facultatif. Utilisé pour l'objet Dictionary uniquement, sans application pour les collections. S'il est fourni, l'élément <i>newitem</i> est la nouvelle valeur associée à l'argument <i>key</i> spécifié.

Notes

Si *key* est introuvable lors de la modification d'un *item*, une nouvelle *key* est créée avec le *newitem* spécifié. Si *key* est introuvable lors de la tentative de retour d'un élément existant, une nouvelle *key* est créée et l'élément correspondant reste vide.

L'exemple ci-dessous illustre l'utilisation de la propriété **Item** :

Function ItemDemo

```
Dim d           ' Crée des variables
Set d = CreateObject("Scripting.Dictionary")
d.Add "a", "Athènes" ' Ajoute des clés et des éléments
d.Add "b", "Belgrade"
d.Add "c", "Casablanca"
ItemDemo = d.Item("c") ' Lit l'élément.
End Function
```

Items, méthode

[Voir aussi](#)

[Application](#)

Description

Renvoie un [tableau](#) contenant tous les éléments d'un objet **Dictionary**.

Syntaxe

object.**Items**

L'espace réservé *object* correspond toujours au nom d'un objet **Dictionary**.

Notes

Le code suivant illustre l'emploi de la méthode **Items** :

Function DicDemo

Dim a, d, i, s ' Crée des variables

Set d = CreateObject("Scripting.Dictionary")

d.Add "a", "Athènes" ' Ajoute des clés et des éléments

d.Add "b", "Belgrade"

d.Add "c", "Casablanca"

a = d.**Items** ' Obtient les éléments

For i = 0 To d.Count -1 ' Effectue une itération sur le tableau

s = s & a(i) & "
" ' Crée la chaîne renvoyée

Next

DicDemo = s

End Function

Microsoft® Visual Basic® Scripting Edition

Key, propriété

[Référence de la bibliothèque
d'exécution Scripting
Version 2](#)

[Voir aussi](#)

[Application](#)

Description

Définit un argument *key* dans un objet **Dictionary**.

Syntaxe

object.**Key**(*key*) = *newkey*

La propriété **Key** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet Dictionary .
<i>key</i>	La valeur de l'argument <i>Key</i> modifiée.
<i>newkey</i>	Nouvelle valeur remplaçant la valeur de l'argument <i>key</i> spécifiée.

Notes

Si l'argument *key* est introuvable lors de la modification d'une *clé*, une [erreur d'exécution](#) se produit.

L'exemple ci-dessous illustre l'utilisation de la propriété **Key** :

Function DicDemo

```
Dim d          ' Crée des variables.  
Set d = CreateObject("Scripting.Dictionary")  
d.Add "a", "Athènes" ' Ajoute des clés et des éléments.  
d.Add "b", "Belgrade"
```

```
d.Add "c", "Casablanca"  
d.Key("c") = "d"      ' Affecte à la clé de "c" la va  
DicDemo = d.Item("d") ' Renvoie l'élément asso  
End Function
```

Keys, méthode

[Voir aussi](#)

[Application](#)

Description

Renvoie un tableau contenant toutes les clés existantes dans un objet **Dictionary**.

Syntaxe

object.**Keys**

L'espace réservé *object* correspond toujours au nom d'un objet **Dictionary**.

Notes

Le code suivant illustre l'emploi de la méthode **Keys** :

Function DicDemo

Dim a, d, i ' Crée des variables

Set d = CreateObject("Scripting.Dictionary")

d.Add "a", "Athènes" ' Ajoute des clés et des éléments

d.Add "b", "Belgrade"

d.Add "c", "Casablanca"

a = d.**Keys** ' Obtient les clés

For i = 0 To d.Count -1 ' Effectue une itération sur le tableau

s = s & a(i) & "
" ' Résultat à renvoyer

Next

DicDemo = s

End Function

Line, propriété

[Voir aussi](#)

[Application](#)

Description

Propriété en lecture seule qui renvoie le numéro de ligne courant d'un fichier **TextStream**.

Syntaxe

object.**Line**

L'espace réservé *object* correspond toujours au nom d'un objet **TextStream**.

Notes

Après l'ouverture initiale d'un fichier et avant toute écriture, **Line** a la valeur 1.

L'exemple ci-dessous illustre l'utilisation de la propriété **Line** :

Function GetLine

```
Const ForReading = 1, ForWriting = 2
```

```
Dim fso, f, ra
```

```
Set fso = CreateObject("Scripting.FileSystemObject")
```

```
Set f = fso.OpenTextFile("c:\testfile.txt", ForWriting, True)
```

```
f.Write "Bonjour!" & vbCrLf & "Vive VB Script!" & vbCrLf
```

```
Set f = fso.OpenTextFile("c:\testfile.txt", ForReading)
```

```
ra = f.ReadAll
```

```
GetLine = f.Line
```

```
End Function
```

Move, méthode

[Voir aussi](#)

[Application](#)

Description

Change l'emplacement d'un fichier ou d'un dossier spécifié.

Syntaxe

object.**Move** *destination*

La syntaxe de la méthode **Move** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet File ou Folder .
<i>destination</i>	Destination du déplacement du fichier ou du dossier. Les caractères génériques ne sont pas autorisés.

Notes

Le résultat de la méthode **Move** sur un objet **File** ou **Folder** est identique à celui de **FileSystemObject.MoveFile** ou **FileSystemObject.MoveFolder**. Notez cependant que ces méthodes alternatives sont capables de déplacer plusieurs fichiers ou dossiers.

L'exemple ci-dessous illustre l'utilisation de la méthode **Move** :

```
Dim fso, MyFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set MyFile = fso.CreateTextFile("c:\testfile.txt", True)
MyFile.WriteLine("Ceci est un test.")
```

MyFile.Close

Set MyFile = fso.GetFile("c:\testfile.txt")

MyFile.Move "c:\windows\bureau\

MoveFile, méthode

[Voir aussi](#)[Application](#)

Description

Change l'emplacement d'un ou plusieurs fichiers.

Syntaxe

object.**MoveFile** *source*, *destination*

La syntaxe de la méthode **MoveFile** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>source</i>	Chemin du ou des fichiers à déplacer. La chaîne argument <i>source</i> peut contenir des caractères génériques dans le dernier composant du chemin uniquement.
<i>destination</i>	Chemin de destination du ou des fichiers à déplacer. L'argument <i>destination</i> ne peut pas contenir de caractères génériques.

Notes

Si l'argument *source* contient des caractères génériques ou si *destination* se termine par un séparateur de chemin (\), *destination* est considéré comme un dossier existant vers lequel s'effectue le déplacement des dossiers et sous-dossiers désignés. Sinon, *destination* est considéré comme le nom d'un fichier de destination à créer. Dans les deux cas, le déplacement d'un fichier individuel présente trois possibilités :

- Si *destination* n'existe pas, le fichier est déplacé. C'est le cas le plus courant.
- Si *destination* est un fichier existant, une erreur se produit.
- Si *destination* est un dossier, une erreur se produit.

Une erreur se produit aussi si une *source* contenant des caractères génériques ne correspond à aucun fichier. La méthode **MoveFile** s'arrête sur la première erreur rencontrée. Aucune tentative n'est effectuée pour annuler les modifications précédant l'erreur.

L'exemple ci-dessous illustre l'utilisation de la méthode **MoveFile** :

```
Sub MoveAFile(Drivespec)
  Dim fso
  Set fso = CreateObject("Scripting.FileSystemObject")
  fso.MoveFile Drivespec, "c:\windows\bureau\"
End Sub
```

Important Cette méthode permet le déplacement de fichiers entre des volumes distincts uniquement si le système d'exploitation gère cette opération.

MoveFolder, méthode

[Voir aussi](#)

[Application](#)

Description

Change l'emplacement d'un ou plusieurs dossiers.

Syntaxe

object.**MoveFolder** *source*, *destination*

La syntaxe de la méthode **MoveFolder** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>source</i>	Chemin du ou des dossiers à déplacer. La chaîne argument <i>source</i> peut contenir des caractères génériques dans le dernier composant du chemin uniquement.
<i>destination</i>	Chemin de destination du ou des dossiers à déplacer. L'argument <i>destination</i> ne peut pas contenir de caractères génériques.

Notes

Si l'argument *source* contient des caractères génériques ou si *destination* se termine par un séparateur de chemin (\), *destination* est considéré comme un dossier existant vers lequel s'effectue le déplacement des dossiers et sous-dossiers désignés. Sinon, *destination* est considéré comme le nom d'un dossier de destination à créer. Dans les deux cas, le déplacement d'un dossier individuel présente trois possibilités :

- Si *destination* n'existe pas, le dossier est déplacé. C'est le cas le plus courant.
- Si *destination* est un fichier existant, une erreur se produit.
- Si *destination* est un dossier, une erreur se produit.

Une erreur se produit aussi si une *source* contenant des caractères génériques ne correspond à aucun dossier. La méthode **MoveFolder** s'arrête sur la première erreur rencontrée. Aucune tentative n'est effectuée pour annuler les modifications précédant l'erreur.

L'exemple ci-dessous illustre l'utilisation de la méthode **MoveFolder** :

```
Sub MoveAFolder(Drivespec)
  Dim fso
  Set fso = CreateObject("Scripting.FileSystemObject")
  fso.MoveFolder Drivespec, "c:\windows\bureau\"
End Sub
```

Important Cette méthode permet le déplacement de dossiers entre des volumes distincts uniquement si le système d'exploitation gère cette opération.

Name, propriété

[Voir aussi](#)

[Application](#)

Description

Définit ou renvoie le nom d'un fichier ou dossier spécifié. Lecture/écriture.

Syntaxe

object.**Name** [= *newname*]

La syntaxe de la propriété **Name** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet File ou Folder .
<i>newname</i>	Facultatif. S'il est fourni, l'argument <i>newname</i> est le nouveau nom de l' <i>object</i> spécifié.

Notes

Le code suivant illustre l'emploi de la propriété **Name** :

```
Function ShowFileInfo(filespec)
    Dim fso, f, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFile(filespec)
    s = f.Name & " sur le lecteur " & UCase(f.Drive) & "<BR>"
    s = s & "Créé le: " & f.DateCreated & "<BR>"
    s = s & "Dernier accès le: " & f.DateLastAccessed & "<BR>"
    s = s & "Dernière modification le: " & f.DateLastModified
    ShowFileInfo = s
```

End Function

OpenAsTextStream, méthode

[Voir aussi](#)

[Application](#)

Description

Ouvre un fichier spécifié et renvoie un objet **TextStream** pouvant être utilisé pour lire le fichier, écrire ou insérer un élément dans le fichier.

Syntaxe

object.**OpenAsTextStream**([*iomode*, [*format*]])

La méthode **OpenAsTextStream** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet File .
<i>iomode</i>	Facultatif. Indique le mode entrée/sortie. Cet argument peut être une des trois constantes : ForReading , ForWriting ou ForAppending .
<i>format</i>	Facultatif. L'une des trois valeurs 3-états permettant d'indiquer le format du fichier ouvert. Si cette valeur est omise, le fichier est ouvert en mode ASCII.

Valeurs

L'argument *iomode* peut prendre les valeurs suivantes :

Constante	Valeur	Description
ForReading	1	Ouvre un fichier en lecture seule. Vous ne pouvez pas écrire dans ce fichier.
ForWriting	2	Ouvre un fichier en écriture. Si un fichier existe sous le même nom, son

		contenu est écrasé.
ForAppending	8	Ouvre un fichier et écrit à la fin du fichier.

L'argument *format* peut prendre l'une des valeurs suivantes :

Constante	Valeur	Description
TristateUseDefault	-2	Ouvre le fichier avec la valeur par défaut du système.
TristateTrue	-1	Ouvre le fichier comme de l'Unicode.
TristateFalse	0	Ouvre le fichier comme de l'ASCII.

Notes

La méthode **OpenAsTextStream** offre la même fonctionnalité que la méthode **OpenTextFile** du **FileSystemObject**. En outre, la méthode **OpenAsTextStream** permet d'écrire dans un fichier.

Le code suivant illustre l'emploi de la méthode **OpenAsTextStream** :

Function TextStreamTest

```

Const ForReading = 1, ForWriting = 2, ForAppending = 4
Const TristateUseDefault = -2, TristateTrue = -1, TristateFalse = 0
Dim fso, f, ts
Set fso = CreateObject("Scripting.FileSystemObject")
fso.CreateTextFile "test1.txt" ' Crée un fichier
Set f = fso.GetFile("test1.txt")
Set ts = f.OpenAsTextStream(ForWriting, TristateTrue)
ts.Write "Bonjour"
ts.Close
Set ts = f.OpenAsTextStream(ForReading, TristateFalse)

```

```
TextStreamTest = ts.ReadLine  
ts.Close  
End Function
```

OpenTextFile, méthode

[Voir aussi](#)[Application](#)

Description

Ouvre un fichier spécifié et renvoie un objet **TextStream** pouvant être utilisé pour lire le fichier, écrire ou insérer un élément dans le fichier.

Syntaxe

object.**OpenTextFile**(*filename*[, *iomode*[, *create*[, *format*]])

La méthode **OpenTextFile** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet FileSystemObject .
<i>filename</i>	Expression de chaîne identifiant le fichier à ouvrir.
<i>iomode</i>	Facultatif. Indique le mode entrée/sortie. Cet argument peut être une des trois constantes : ForReading , ForWriting ou ForAppending .
<i>create</i>	Facultatif. Valeur de type Boolean qui indique si un nouveau fichier peut être créé si le nom <i>filename</i> spécifié n'existe pas. La valeur est True si un nouveau fichier est créé, False dans le cas contraire. La valeur par défaut est False .
<i>format</i>	Facultatif. L'une des trois valeurs 3-états permettant d'indiquer le format du fichier ouvert. Si cette valeur est omise, le fichier est ouvert en mode ASCII.

Valeurs

L'argument *iomode* peut prendre les valeurs suivantes :

Constante	Valeur	Description
ForReading	1	Ouvre un fichier en lecture seule. Vous ne pouvez pas écrire dans ce fichier.
ForWriting	2	Ouvre un fichier en écriture seule. Vous ne pouvez pas lire à partir de ce fichier.
ForAppending	8	Ouvre un fichier et écrit à la fin du fichier.

L'argument *format* peut prendre l'une des valeurs suivantes :

Constante	Valeur	Description
TristateUseDefault	-2	Ouvre le fichier avec la valeur par défaut du système.
TristateTrue	-1	Ouvre le fichier comme de l'Unicode.
TristateFalse	0	Ouvre le fichier comme de l'ASCII.

Notes

Le code suivant illustre l'utilisation de la méthode **OpenTextFile** pour ouvrir un fichier et y écrire du texte :

```
Sub OpenTextFileTest
  Const ForReading = 1, ForWriting = 2, ForAppending = 8
  Dim fso, f
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set f = fso.OpenTextFile("c:\testfile.txt", ForWriting, True)
  f.Write "Bonjour!"
  f.Close
End Sub
```

ParentFolder, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie l'objet dossier pour le parent du fichier ou dossier spécifié. Lecture seule.

Syntaxe

object.**ParentFolder**

L'argument *object* représente toujours un objet **File** ou **Folder**.

Notes

Le code suivant illustre l'emploi de la propriété **ParentFolder** avec un fichier :

```
Function ShowFileAccessInfo(filespec)
    Dim fso, f, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFile(filespec)
    s = UCase(f.Name) & " dans " & UCase(f.ParentFolder) & "<BR>"
    s = s & "Créé le: " & f.DateCreated & "<BR>"
    s = s & "Dernier accès le: " & f.DateLastAccessed & "<BR>"
    s = s & "Dernière modification le: " & f.DateLastModified
    ShowFileAccessInfo = s
End Function
```

Path, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie le chemin d'un fichier, dossier ou lecteur spécifié.

Syntaxe

object.**Path**

L'argument *object* représente toujours un objet **File**, **Folder** ou **Drive**.

Notes

Pour les lettres de lecteur, la racine ne figure pas. Par exemple, le chemin du lecteur C est C:, et non pas C:\.

Le code suivant illustre l'emploi de la propriété **Path** avec un objet **File** :

```
Function ShowFileInfo(filespec)
    Dim fso, d, f, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFile(filespec)
    s = UCase(f.Path) & "<BR>"
    s = s & "Créé le: " & f.DateCreated & "<BR>"
    s = s & "Dernier accès le: " & f.DateLastAccessed & "<BR>"
    s = s & "Dernière modification le: " & f.DateLastModified
    ShowFileInfo = s
End Function
```

Read, méthode

[Voir aussi](#)

[Application](#)

Description

Lit un nombre spécifié de caractères dans un fichier **TextStream** et renvoie la chaîne résultante.

Syntaxe

object.**Read**(*characters*)

La syntaxe de la méthode **Read** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet TextStream .
<i>characters</i>	Nombre de caractères à lire dans le fichier.

Notes

L'exemple ci-dessous illustre l'utilisation de la méthode **Read** pour lire cinq caractères dans un fichier et renvoyer la chaîne résultante :

Function ReadTextFileTest

```
Const ForReading = 1, ForWriting = 2, ForAppend = 4
```

```
Dim fso, f, Msg
```

```
Set fso = CreateObject("Scripting.FileSystemObject")
```

```
Set f = fso.OpenTextFile("c:\testfile.txt", ForWriting)
```

```
f.Write "Bonjour!"
```

```
Set f = fso.OpenTextFile("c:\testfile.txt", ForReading)
ReadTextFileTest = f.Read(5)
End Function
```

ReadAll, méthode

[Voir aussi](#)[Application](#)

Description

Lit la totalité d'un fichier **TextStream** et renvoie la chaîne résultante.

Syntaxe

object.**ReadAll**

object correspond toujours au nom d'un objet **TextStream**.

Notes

Pour les fichiers volumineux, la méthode **ReadAll** n'utilise pas de façon rationnelle les ressources mémoire. D'autres techniques d'entrée doivent être employées, telles que la lecture d'un fichier ligne par ligne.

Function ReadAllTextFile

```
Const ForReading = 1, ForWriting = 2
```

```
Dim fso, f
```

```
Set fso = CreateObject("Scripting.FileSystemObject")
```

```
Set f = fso.OpenTextFile("c:\testfile.txt", ForWriting, True)
```

```
f.Write "Bonjour!"
```

```
Set f = fso.OpenTextFile("c:\testfile.txt", ForReading)
```

```
ReadAllTextFile = f.ReadAll
```

```
End Function
```

ReadLine, méthode

[Voir aussi](#)

[Application](#)

Description

Lit toute une ligne (caractère de nouvelle ligne exclus) d'un fichier **TextStream** et renvoie la chaîne résultante.

Syntaxe

object.**ReadLine**

L'argument *object* correspond toujours au nom d'un objet **TextStream**.

Notes

L'exemple ci-dessous utilise la méthode **ReadLine** pour lire une ligne dans un fichier **TextStream** et renvoyer une chaîne :

```
Function ReadLineTextFile
    Const ForReading = 1, ForWriting = 2
    Dim fso, MyFile
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set MyFile = fso.OpenTextFile("c:\testfile.txt", ForWriting, True)
    MyFile.WriteLine "Bonjour!"
    MyFile.WriteLine "Le petit renard roux et vif"
    MyFile.Close
    Set MyFile = fso.OpenTextFile("c:\testfile.txt", ForReading)
    ReadLineTextFile = MyFile.ReadLine ' Renvoie "Bonjour!"
End Function
```

Remove, méthode

[Voir aussi](#)[Application](#)

Description

Retire une paire clé/élément d'un objet **Dictionary**.

Syntaxe

object.**Remove**(*key*)

La syntaxe de la méthode **Remove** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet Dictionary .
<i>key</i>	Argument associé à la paire <i>clé/élément</i> à retirer de l'objet Dictionary .

Notes

Une erreur se produit si la paire clé/élément n'existe pas.

Le code suivant illustre l'emploi de la méthode **Remove**:

```
Dim a, d           ' Crée des variables
Set d = CreateObject("Scripting.Dictionary")
d.Add "a", "Athènes"   ' Ajoute des clés et des éléments
d.Add "b", "Belgrade"
d.Add "c", "Casablanca"
...
a = d.Remove("b")     ' Retire la deuxième paire
```

RemoveAll, méthode

[Voir aussi](#)

[Application](#)

Description

La méthode **RemoveAll** retire toutes les paires clé/élément d'un objet **Dictionary**.

Syntaxe

object.**RemoveAll**

L'espace réservé *object* correspond toujours au nom d'un objet **Dictionary**.

Notes

Le code suivant illustre l'emploi de la méthode **RemoveAll**:

```
Dim a, d, i          ' Crée des variables
Set d = CreateObject("Scripting.Dictionary")
d.Add "a", "Athènes" ' Ajoute des clés et des éléments
d.Add "b", "Belgrade"
d.Add "c", "Casablanca"
...
a = d.RemoveAll    ' Purge le dictionnaire
```

RootFolder, propriété

[Référence de la bibliothèque
d'exécution Scripting
Version 3](#)

[Voir aussi](#)

[Application](#)

Description

Renvoie un objet **Folder** correspondant au dossier racine d'un lecteur spécifié. Lecture seule.

Syntaxe

object.**RootFolder**

L'argument *object* représente toujours un objet **Drive**.

Notes

Tous les fichiers et dossiers du lecteur sont accessibles à l'aide de l'objet **Folder** renvoyé.

L'exemple ci-dessous illustre l'utilisation de la propriété **RootFolder** :

```
Function ShowRootFolder(drvspec)
    Dim fso, f
    Set fso = CreateObject("Scripting.FileSystemObj
    Set f = fso.GetDrive(drvspec)
    ShowRootFolder = f.RootFolder
End Function
```

SerialNumber, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie le numéro de série décimal utilisé pour identifier de manière unique un volume de disque.

Syntaxe

object.**SerialNumber**

L'argument *object* représente toujours un objet **Drive**.

Notes

Vous pouvez utiliser la propriété **SerialNumber** pour vous assurer que le disque approprié est présent dans un lecteur à support amovible.

Le code suivant illustre l'emploi de la propriété **SerialNumber** :

```
Function ShowDriveInfo(drvpath)
    Dim fso, d, s, t
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set d = fso.GetDrive(fso.GetDriveName(fso.GetAbsolutePathNa
    Select Case d.DriveType
        Case 0: t = "Inconnu"
        Case 1: t = "Amovible"
        Case 2: t = "Fixe"
        Case 3: t = "Réseau"
        Case 4: t = "CD-ROM"
        Case 5: t = "RAM Disk"
```

```
End Select
s = "Lecteur " & d.DriveLetter & ": - " & t
s = s & "<BR>" & "SN: " & d.SerialNumber
ShowDriveInfo = s
End Function
```

ShareName, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie le nom du partage réseau pour un lecteur spécifié.

Syntaxe

object.**ShareName**

L'argument *object* représente toujours un objet **Drive**.

Notes

Si l'argument *object* ne représente pas un lecteur réseau, la propriété **ShareName** renvoie une chaîne de longueur nulle ("").

Le code suivant illustre l'emploi de la propriété **ShareName** :

```
Function ShowDriveInfo(drvpath)
    Dim fso, d
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set d = fso.GetDrive(fso.GetDriveName(fso.GetAbsolutePathName(drvpath)))
    ShowDriveInfo = "Lecteur " & d.DriveLetter & ": - " & d.ShareName
End Function
```

ShortName, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie le nom abrégé utilisé par les programmes qui emploient la convention de dénomination 8.3.

Syntaxe

object.**ShortName**

L'argument *object* représente toujours un objet **File** ou **Folder**.

Notes

Le code suivant illustre l'emploi de la propriété **ShareName** avec un objet **File** :

```
Function ShowShortName(filespec)
    Dim fso, f, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFile(filespec)
    s = "Le nom abrégé de " & UCase(f.Name) & "<BR>"
    s = s & "est: " & f.ShortName
    ShowShortName = s
End Function
```

ShortPath, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie le chemin abrégé utilisé par les programmes qui emploient la convention de dénomination 8.3.

Syntaxe

object.**ShortPath**

L'argument *object* représente toujours un objet **File** ou **Folder**.

Notes

Le code suivant illustre l'emploi de la propriété **ShareName** avec un objet **File** :

```
Function ShowShortPath(filespec)
    Dim fso, f, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFile(filespec)
    s = "Le chemin abrégé de " & UCase(f.Name) & "<BR>"
    s = s & "est: " & f.ShortPath
    ShowShortPath = s
End Function
```

Size, propriété

[Voir aussi](#)

[Application](#)

Description

Pour les fichiers, cette propriété renvoie la taille en octets du fichier spécifié. Pour les dossiers, cette propriété renvoie la taille en octets de tous les fichiers et sous-dossiers du dossier.

Syntaxe

object.**Size**

L'argument *object* représente toujours un objet **File** ou **Folder**.

Notes

Le code suivant illustre l'emploi de la propriété **Size** avec un objet **File** :

```
Function ShowFolderSize(filespec)
    Dim fso, f, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFolder(filespec)
    s = UCase(f.Name) & " utilise " & f.size & " octets."
    ShowFolderSize = s
End Function
```

Skip, méthode

[Voir aussi](#)

[Application](#)

Description

Omet un nombre spécifié de caractères lors de la lecture d'un fichier **TextStream**.

Syntaxe

object.**Skip**(*characters*)

La syntaxe de la méthode **Skip** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet TextStream .
<i>characters</i>	Nombre de caractères à omettre lors de la lecture d'un fichier.

Notes

Les caractères omis sont ignorés.

L'exemple ci-dessous utilise la méthode **Skip** pour passer les huit premiers caractères avant de lire dans un fichier texte :

Function SkipTextFile

```
Const ForReading = 1, ForWriting = 2
```

```
Dim fso, f
```

```
Set fso = CreateObject("Scripting.FileSystemObj
```

```
Set f = fso.OpenTextFile("c:\testfile.txt", ForWrit
```

```
f.Write "Bonjour!"  
Set f = fso.OpenTextFile("c:\testfile.txt", ForReading)  
f.Skip(6)  
SkipTextFile = f.ReadLine  
End Function
```

SkipLine, méthode

[Voir aussi](#)[Application](#)

Description

Omet la ligne suivante lors de la lecture d'un fichier **TextStream**.

Syntaxe

object.**SkipLine**

object correspond toujours au nom d'un objet **TextStream**.

Notes

L'omission d'une ligne implique la lecture et la suppression de tous les caractères d'une ligne, y compris du caractère de saut de ligne. Une erreur se produit si le fichier n'est pas ouvert en lecture.

L'exemple ci-dessous illustre l'utilisation de la méthode **SkipLine** :

Function SkipLineInFile

```
Const ForReading = 1, ForWriting = 2
Dim fso, f
Set fso = CreateObject("Scripting.FileSystemObject")
Set f = fso.OpenTextFile("c:\testfile.txt", ForWriting, True)
f.Write "Bonjour!" & vbCrLf & "Vive VB Script!"
Set f = fso.OpenTextFile("c:\testfile.txt", ForReading)
f.SkipLine
SkipLineInFile = f.ReadLine
End Function
```

SpecialFolder, constantes

[Voir aussi](#)

Ces constantes sont disponibles uniquement lorsque votre projet contient une référence explicite à la [bibliothèque de types](#) contenant leurs définitions. Pour VBScript, vous devez déclarer ces constantes explicitement dans votre code.

Constante	Valeur	Description
WindowsFolder	0	Le dossier Windows contient des fichiers installés par le système d'exploitation Windows.
SystemFolder	1	Le dossier Système contient les bibliothèques, les polices et les pilotes de périphérique.
TemporaryFolder	2	Le dossier Temp stocke les fichiers temporaires. Son chemin figure dans la variable d'environnement TMP.

SubFolders, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie une collection **Folders** comprenant tous les dossiers contenus dans le dossier spécifié, y compris ceux qui sont cachés ou système.

Syntaxe

object.**SubFolders**

L'argument *object* représente toujours un objet **Folder**.

Notes

Le code suivant illustre l'emploi de la propriété **SubFolders** :

```
Function ShowFolderList(folderspec)
    Dim fso, f, f1, s, sf
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFolder(folderspec)
    Set sf = f.SubFolders
    For Each f1 in sf
        s = s & f1.name
        s = s & "<BR>"
    Next
    ShowFolderList = s
End Function
```

TextStream, objet

[Voir aussi](#)

[Propriétés](#)

[Méthodes](#)

Description

Facilite l'accès séquentiel à un fichier.

Notes

Dans le code suivant, **a** correspond à l'objet **TextStream** renvoyé par la méthode **CreateTextFile** sur l'objet **FileSystemObject** :

```
Dim fso, MyFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set MyFile= fso.CreateTextFile("c:\testfile.txt", True)
MyFile.WriteLine("Ceci est un test.")
MyFile.Close
```

WriteLine et **Close** sont deux méthodes de l'objet **TextStream**.

TotalSize, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie l'espace total, en octets, d'un lecteur ou d'un partage réseau.

Syntaxe

object.**TotalSize**

L'argument *object* représente toujours un objet **Drive**.

Notes

Le code suivant illustre l'emploi de la propriété **TotalSize** :

```
Function ShowSpaceInfo(drvpath)
    Dim fso, d, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set d = fso.GetDrive(fso.GetDriveName(fso.GetAbsolutePathName(drvpath)))
    s = "Lecteur " & d.DriveLetter & ":"
    s = s & vbCrLf
    s = s & "Taille totale: " & FormatNumber(d.TotalSize/1024, 0) & " Ko"
    s = s & vbCrLf
    s = s & "Disponible: " & FormatNumber(d.AvailableSpace/1024, 0) & " Ko"
    ShowSpaceInfo = s
End Function
```

Type, propriété

[Voir aussi](#)

[Application](#)

Description

Renvoie des informations sur le type d'un fichier ou dossier. Par exemple, pour les fichiers terminés par .TXT, la propriété renvoie "Document texte".

Syntaxe

object.**Type**

L'argument *object* représente toujours un objet **File** ou **Folder**.

Notes

Le code suivant illustre l'emploi de la propriété **Type** pour renvoyer un type de dossier. Dans cet exemple, essayez de fournir à la procédure le chemin de la Corbeille ou d'un autre dossier unique.

```
Function ShowFolderType(filespec)
    Dim fso, f, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFolder(filespec)
    s = UCase(f.Name) & " est un " & f.Type
    ShowFolderType = s
End Function
```

VolumeName, propriété

[Voir aussi](#)

[Application](#)

Description

Définit ou renvoie le nom de volume du lecteur spécifié. Lecture/écriture.

Syntaxe

object.**VolumeName** [= *newname*]

La propriété **VolumeName** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet Drive .
<i>newname</i>	Facultatif. S'il est fourni, l'argument <i>newname</i> est le nouveau nom de l' <i>object</i> spécifié.

Note

Le code suivant illustre l'emploi de la propriété **VolumeName** :

```
Function ShowVolumeInfo(drspath)
    Dim fso, d, s
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set d = fso.GetDrive(fso.GetDriveName(fso.GetAbsolutePathName(drspath)))
    s = "Lecteur " & d.DriveLetter & ": - " & d.VolumeName
    ShowVolumeInfo = s
End Function
```

Write, méthode

[Voir aussi](#)

[Application](#)

Description

Écrit une chaîne spécifiée dans un fichier **TextStream**.

Syntaxe

object.**Write**(*string*)

La syntaxe de la méthode **Write** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet TextStream .
<i>string</i>	Le texte à écrire dans le fichier.

Notes

Les chaînes spécifiées sont écrites dans le fichier sans espaces ni caractères entre chaque chaîne. Utilisez la méthode **WriteLine** pour écrire un caractère de saut de ligne ou une chaîne se terminant par un caractère de saut de ligne.

L'exemple ci-dessous illustre l'utilisation de la méthode **Write** :

Function WriteToFile

```
Const ForReading = 1, ForWriting = 2
```

```
Dim fso, f
```

```
Set fso = CreateObject("Scripting.FileSystemObj
```

```
Set f = fso.OpenTextFile("c:\testfile.txt", ForWrit
```

```
f.Write "Bonjour!"  
Set f = fso.OpenTextFile("c:\testfile.txt", ForReading)  
WriteToFile = f.ReadLine  
End Function
```

WriteBlankLines, méthode

[Voir aussi](#)

[Application](#)

Description

Écrit un nombre spécifié de caractères de nouvelle ligne dans un fichier **TextStream**.

Syntaxe

object.**WriteBlankLines**(*lines*)

La syntaxe de la méthode **WriteBlankLines** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet TextStream .
<i>lines</i>	Nombre de caractères de saut de ligne à écrire dans le fichier.

Notes

L'exemple ci-dessous illustre l'utilisation de la méthode **WriteBlankLines** :

```
Function WriteBlankLinesToFile
  Const ForReading = 1, ForWriting = 2
  Dim fso, f
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set f = fso.OpenTextFile("c:\testfile.txt", ForWriting, True)
  f.WriteBlankLines 2
  f.WriteLine "Bonjour!"
  Set f = fso.OpenTextFile("c:\testfile.txt", ForReading)
```

```
WriteBlankLinesToFile = f.ReadAll  
End Function
```

WriteLine, méthode

[Voir aussi](#)

[Application](#)

Description

Écrit une chaîne spécifiée et un caractère de nouvelle ligne dans un fichier **TextStream**.

Syntaxe

object.**WriteLine**([*string*])

La syntaxe de la méthode **WriteLine** comprend les éléments suivants :

Élément	Description
<i>object</i>	Correspond toujours au nom d'un objet TextStream .
<i>string</i>	Facultatif. Le texte à écrire dans le fichier. S'il est omis, un caractère de saut de ligne est écrit dans le fichier.

Notes

L'exemple ci-dessous illustre l'utilisation de la méthode **WriteLine** :

```
Function WriteLineToFile
  Const ForReading = 1, ForWriting = 2
  Dim fso, f
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set f = fso.OpenTextFile("c:\testfile.txt", ForWriting, True)
  f.WriteLine "Bonjour!"
  f.WriteLine "Vive VBScript!"
End Function
```

```
Set f = fso.OpenTextFile("c:\testfile.txt", ForReading)
WriteLineToFile = f.ReadAll
End Function
```

Copyright

Microsoft® Visual Basic® Scripting Edition

Les informations contenues dans ce document pourront faire l'objet de modifications sans préavis. Sauf mention contraire, les sociétés, les noms et les données utilisés dans les exemples sont fictifs. L'utilisateur est tenu d'observer la réglementation relative aux droits d'auteur applicable dans son pays. Aucune partie de ce manuel ne peut être reproduite ou transmise à quelque fin ou par quelque moyen que ce soit, électronique ou mécanique, sans la permission expresse et écrite de Microsoft Corporation.

Microsoft peut détenir des brevets, avoir déposé des demandes d'enregistrement de brevets ou être titulaire de marques, droits d'auteur ou autres droits de propriété intellectuelle portant sur tout ou partie des éléments qui font l'objet du présent document. Sauf stipulation expresse contraire d'un contrat de licence écrit de Microsoft, la fourniture de ce document n'a pas pour effet de vous concéder une licence sur ces brevets, marques, droits d'auteur ou autres droits de propriété intellectuelle.

© 1991-2000 Microsoft Corporation. Tous droits réservés.

Microsoft, MS, MS-DOS, ActiveX, JScript, Microsoft Press, Visual Basic, Windows, Windows NT, Win32 et Win32s sont soit des marques de Microsoft Corporation, soit des marques déposées de Microsoft Corporation, aux États-Unis d'Amérique et/ou dans d'autres pays.

Les autres noms de produits et sociétés mentionnés sont soit des marques, soit des marques déposées de leurs propriétaires respectifs.

Utilisation des fichiers

[Précédent](#)
[Suivant](#)

Les manipulations de fichiers se divisent en deux catégories principales :

- La création, l'ajout ou la suppression de données et la lecture de fichiers.
- Le déplacement, la copie et la suppression de fichiers.

La création de fichiers

Trois moyens différents permettent de créer un fichier de texte vide (parfois désigné comme "flux de texte").

Le premier moyen est l'utilisation de la méthode **CreateTextFile**. L'exemple ci-dessous illustre la création d'un fichier de texte à l'aide de cette méthode dans VBScript :

```
Dim fso, f1
Set fso = CreateObject("Scripting.FileSystemObject")
Set f1 = fso.CreateTextFile("c:\testfile.txt", True)
```

Pour utiliser cette méthode dans JScript, utilisez le code suivant :

```
var fso, f1;
fso = new ActiveXObject("Scripting.FileSystemObject");
f1 = fso.CreateTextFile("c:\\testfile.txt", true);
```

Cet [exemple de code](#) illustre l'emploi de la méthode **CreateTextFile** dans le **FileSystemObject**.

Le second moyen de créer un fichier de texte consiste à utiliser la méthode **OpenTextFile** de l'objet **FileSystemObject** avec l'indicateur **ForWriting**. Dans VBScript, le code ressemble à ceci :

```
Dim fso, ts
Const ForWriting = 2
```

```
Set fso = CreateObject("Scripting.FileSystemObject")
Set ts = fso.OpenTextFile("c:\test.txt", ForWriting, True)
```

Pour créer un fichier de texte en utilisant cette méthode dans JScript, utilisez le code suivant :

```
var fso, ts;
var ForWriting= 2;
fso = new ActiveXObject("Scripting.FileSystemObject");
ts = fso.OpenTextFile("c:\\test.txt", ForWriting, true);
```

Le troisième moyen de créer un fichier de texte consiste à utiliser la méthode **OpenAsTextStream** avec l'indicateur **ForWriting**. Pour cette méthode, utilisez le code ci-dessous dans VBScript :

```
Dim fso, f1, ts
Const ForWriting = 2
Set fso = CreateObject("Scripting.FileSystemObject")
fso.CreateTextFile ("c:\test1.txt")
Set f1 = fso.GetFile("c:\test1.txt")
Set ts = f1.OpenAsTextStream(ForWriting, True)
```

Dans JScript, utilisez le code de l'exemple ci-dessous :

```
var fso, f1, ts;
var ForWriting = 2;
fso = new ActiveXObject("Scripting.FileSystemObject");
fso.CreateTextFile ("c:\\test1.txt");
f1 = fso.GetFile("c:\\test1.txt");
ts = f1.OpenAsTextStream(ForWriting, true);
```

Ajout de données dans le fichier

Une fois le fichier créé, procédez ainsi pour ajouter des données au fichier :

1. Ouvrez le fichier.

2. Écrivez les données.
3. Fermez le fichier.

Pour ouvrir un fichier existant, utilisez soit la méthode **OpenTextFile** de l'objet **FileSystemObject** soit la méthode **OpenAsTextStream** de l'objet **File**.

Pour écrire des données dans le fichier de texte, utilisez les méthodes **Write**, **WriteLine** ou **WriteBlankLines** de l'objet **TextStream**, en fonction des tâches répertoriées dans le tableau ci-dessous.

Tâche	Méthode
Écrire des données dans un fichier de texte ouvert sans caractère de nouvelle ligne final.	Write
Écrire des données dans un fichier de texte ouvert avec un caractère de nouvelle ligne final.	WriteLine
Écrire une ou plusieurs lignes blanches dans un fichier de texte ouvert.	WriteBlankLines

Cet [exemple de code](#) illustre l'emploi des méthodes **Write**, **WriteLine** et **WriteBlankLines** dans le **FileSystemObject**.

Pour fermer un fichier ouvert, utilisez la méthode **Close** de l'objet **TextStream**.

Cet [exemple de code](#) illustre l'emploi de la méthode **Close** dans le **FileSystemObject**.

Remarque Le caractère de saut de ligne contient un ou plusieurs caractères (en fonction du système d'exploitation) qui avancent le curseur au début de la ligne d'après (retour chariot/saut de ligne). Tenez compte du fait que certaines chaînes se terminent déjà par ce type de caractères non imprimables.

L'exemple VBScript suivant montre comment ouvrir un fichier, utiliser les trois méthodes d'écritures pour ajouter des données au fichier, puis fermer le fichier:

```
Sub CreateFile()  
Dim fso, tf
```

```

Set fso = CreateObject("Scripting.FileSystemObject")
Set tf = fso.CreateTextFile("c:\testfile.txt", True)
' Écrire une ligne terminée par un caractère de nouvelle ligne.
tf.WriteLine("Testing 1, 2, 3.")
' Écrire trois caractères de nouvelle ligne dans le fichier
tf.WriteLine(3)
' Écrire une ligne.
tf.Write ("Ceci est un essai.")
tf.Close
End Sub

```

Cet exemple illustre l'utilisation de ces trois méthodes dans JScript :

```

function CreateFile()
{
    var fso, tf;
    fso = new ActiveXObject("Scripting.FileSystemObject");
    tf = fso.CreateTextFile("c:\\testfile.txt", true);
    // Écrit une ligne terminée par un caractère de nouvelle ligne.
    tf.WriteLine("Testing 1, 2, 3.") ;
    // Écrit trois caractères de nouvelle ligne dans le fichier.
    tf.WriteLine(3) ;
    // Écrit une ligne.
    tf.Write ("Ceci est un essai.");
    tf.Close();
}

```

Lecture de fichiers

Pour lire des données d'un fichier de texte, utilisez les méthodes **Read**, **ReadLine** ou **ReadAll** de l'objet **TextStream**. Le tableau suivant présente les méthodes à utiliser en fonction des tâches à accomplir.

Tâche	Méthode
Lire un nombre spécifié de caractères d'un fichier.	Read

Lire une ligne entière (jusqu'au caractère de nouvelle ligne non compris).	ReadLine
Lire le contenu entier d'un fichier de texte.	ReadAll

Cet [exemple de code](#) illustre l'emploi des méthodes **ReadAll** et **ReadLine** dans le **FileSystemObject**.

Si vous utilisez la méthode **Read** ou **ReadLine** et voulez accéder directement à un emplacement de données particulier, utilisez la méthode **Skip** ou **SkipLine**. Le texte issu des méthodes de lecture est stocké dans une chaîne qui peut être affichée dans un contrôle, analysé par des fonctions de chaîne (comme **Left**, **Right** et **Mid**), concaténé, etc.

L'exemple VBScript suivant montre comment ouvrir un fichier, écrire puis lire :

```

Sub ReadFiles
  Dim fso, f1, ts, s
  Const ForReading = 1
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set f1 = fso.CreateTextFile("c:\testfile.txt", True)
  ' Écrire une ligne.
  Response.Write "Écriture du fichier <br>"
  f1.WriteLine "Bonjour"
  f1.WriteBlankLines(1)
  f1.Close
  ' Lire le contenu du fichier.
  Response.Write "Lecture du fichier <br>"
  Set ts = fso.OpenTextFile("c:\testfile.txt", ForReading)
  s = ts.ReadLine
  Response.Write "Contenu du fichier = " & s & ""
  ts.Close
End Sub

```

Ce code effectue les mêmes opérations dans JScript :

```

function ReadFiles()
{
  var fso, f1, ts, s;

```

```

var ForReading = 1;
fso = new ActiveXObject("Scripting.FileSystemObject");
f1 = fso.CreateTextFile("c:\\testfile.txt", true);
// Écrire une ligne.
Response.Write("Écriture du fichier <br>");
f1.WriteLine("Bonjour");
f1.WriteBlankLines(1);
f1.Close();
// Lire le contenu du fichier.
Response.Write("Lecture du fichier <br>");
ts = fso.OpenTextFile("c:\\testfile.txt", ForReading);
s = ts.ReadLine();
Response.Write("Contenu du fichier = '" + s + "'");
ts.Close();
}

```

Déplacement, copie et suppression de fichiers

Le modèle d'objet FSO offre deux méthodes pour chacune des opérations de déplacement, copie et suppression de fichier, comme le présente le tableau suivant.

Tâche	Méthode
Déplacer un fichier	File.Move ou FileSystemObject.MoveFile
Copier un fichier	File.Copy ou FileSystemObject.CopyFile
Supprimer un fichier	File.Delete ou FileSystemObject.DeleteFile

Cet [exemple de code](#) illustre les deux façons de supprimer un fichier dans le **FileSystemObject**.

L'exemple VBScript suivant crée un fichier de texte dans le répertoire racine du lecteur C, y écrit des informations, le déplace dans un répertoire nommé \tmp, le copie dans un répertoire nommé \temp puis supprime les deux copies des deux répertoires.

Pour exécuter cet exemple, créez des répertoires nommés \tmp et \temp sous la racine du lecteur C:

Sub ManipFiles

```

Dim fso, f1, f2, s
Set fso = CreateObject("Scripting.FileSystemObject")
Set f1 = fso.CreateTextFile("c:\testfile.txt", True)
Response.Write "Écriture du fichier <br>"
' Écrire une ligne.
f1.Write ("Ceci est un essai.")
' Fermeture du fichier en écriture.
f1.Close
Response.Write "Déplacement du fichier vers c:\tmp <br>"
' Obtenir un descripteur vers le fichier sous la racine de C:\.
Set f2 = fso.GetFile("c:\testfile.txt")
' Déplacer le fichier vers le répertoire /tmp.
f2.Move ("c:\tmp\testfile.txt")
Response.Write "Copie du fichier vers c:\temp <br>"
' Copier le fichier vers \temp.
f2.Copy ("c:\temp\testfile.txt")
Response.Write "Suppression des fichiers <br>"
' Obtenir des descripteurs vers l'emplacement actuel des fichiers.
Set f2 = fso.GetFile("c:\tmp\testfile.txt")
Set f3 = fso.GetFile("c:\temp\testfile.txt")
' Supprimer les fichiers.
f2.Delete
f3.Delete
Response.Write "Terminé!"
End Sub

```

Le code suivant vous présente la même opération dans JScript :

```

function ManipFiles()
{
    var fso, f1, f2, s;
    fso = new ActiveXObject("Scripting.FileSystemObject");
    f1 = fso.CreateTextFile("c:\\testfile.txt", true);

```

```
Response.Write("Écriture du fichier <br>");
// Écrire une ligne.
f1.Write("Ceci est un essai.");
// Fermeture du fichier en écriture.
f1.Close();
Response.Write("Déplacement du fichier vers c:\\tmp <br>");
// Obtenir un descripteur vers le fichier sous la racine de C:\.
f2 = fso.GetFile("c:\\testfile.txt");
// Déplacer le fichier vers le répertoire /tmp.
f2.Move ("c:\\tmp\\testfile.txt");
Response.Write("Copie du fichier vers c:\\temp <br>");
// Copier le fichier vers \temp.
f2.Copy ("c:\\temp\\testfile.txt");
Response.Write("Suppression des fichiers <br>");
// Obtenir des descripteurs vers l'emplacement actuel des fichiers.
f2 = fso.GetFile("c:\\tmp\\testfile.txt");
f3 = fso.GetFile("c:\\temp\\testfile.txt");
// Supprimer les fichiers.
f2.Delete();
f3.Delete();
Response.Write("Terminé!");
}
```

Microsoft® Visual Basic® Scripting Edition

Abs, fonction

[Référence du langage](#)

Voir aussi

[Sgn, fonction](#)



opérateur

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[&, opérateur](#)

[-, opérateur](#)

[Opérateurs arithmétiques](#)

[Opérateurs de concaténation](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

And, opérateur

[Référence du langage](#)

Voir aussi

[Opérateurs logiques](#)

[Not, opérateur](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

[Or, opérateur](#)

[Xor, opérateur](#)

Microsoft® Visual Basic® Scripting Edition

Array, fonction

[Référence du langage](#)

Voir aussi

[Dim, instruction](#)

Microsoft® Visual Basic® Scripting Edition

Asc , fonction

[Référence du langage](#)

Voir aussi

[Chr, fonction](#)

Opérateur =

[Référence du langage](#)

Voir aussi

[Opérateurs de comparaison](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

[Set, instruction](#)

Atn, fonction

[Référence du langage](#)

Voir aussi

[Cos, fonction](#)

[Fonctions mathématiques dérivées](#)

[Sin, fonction](#)

[Tan, fonction](#)

CBool, fonction

[Référence du langage](#)

Voir aussi

[CByte, fonction](#)

[CCur, fonction](#)

[CDate, fonction](#)

[CDBl, fonction](#)

[CInt, fonction](#)

[CLng, fonction](#)

[CSng, fonction](#)

[CStr, fonction](#)

Cparte, fonction

[Référence du langage](#)

Voir aussi

[CBool, fonction](#)

[CCur, fonction](#)

[CDate, fonction](#)

[CDBl, fonction](#)

[CInt, fonction](#)

[CLng, fonction](#)

[CSng, fonction](#)

[CStr, fonction](#)

fonction

[Référence du langage](#)

Voir aussi

[CBool, fonction](#)

[CByte, fonction](#)

[CDate, fonction](#)

[CDBl, fonction](#)

[CInt, fonction](#)

[CLng, fonction](#)

[CSng, fonction](#)

[CStr, fonction](#)

Microsoft® Visual Basic® Scripting Edition

CDate, fonction

[Référence du langage](#)

Voir aussi

[IsDate, fonction](#)

fonction

[Référence du langage](#)

Voir aussi

[CBool, fonction](#)

[CByte Function](#)

[CCur, fonction](#)

[CDate, fonction](#)

[CInt, fonction](#)

[CLng, fonction](#)

[CSng, fonction](#)

[CStr, fonction](#)

Microsoft® Visual Basic® Scripting Edition

Chr, fonction

[Référence du langage](#)

Voir aussi

[Asc, fonction](#)

CInt, fonction

[Référence du langage](#)

Voir aussi

[CBool, fonction](#)

[CByte Function](#)

[CCur, fonction](#)

[CDate, fonction](#)

[Cdbl, fonction](#)

[CLng, fonction](#)

[CSng, fonction](#)

[CStr, fonction](#)

[Int, Fix, fonctions](#)

Class, objet

[Référence du langage](#)

Voir aussi

[Class, instruction](#)

[Dim, instruction](#)

[Set, instruction](#)

Microsoft® Visual Basic® Scripting Edition

Class,

objet

Événements

[Référence du langage](#)

[Initialize, événement](#)

[Terminate, événement](#)

Class, instruction

[Référence du langage](#)

Voir aussi

[Dim, instruction](#)

[Function, instruction](#)

[Private, instruction](#)

[Property Get, instruction](#)

[Property Let, instruction](#)

[Property Set, instruction](#)

[Public, instruction](#)

[Set, instruction](#)

[Sub, instruction](#)

Clear, méthode

[Référence du langage](#)

Voir aussi

[Description, propriété](#)

[Err Object](#)

[Number, propriété](#)

[On Error, instruction](#)

[Raise, méthode](#)

[Source, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Clear, méthode Application

[Référence du langage](#)

[Err. objet](#)

fonction

[Référence du langage](#)

Voir aussi

[CBool, fonction](#)

[CByte Function](#)

[CCur, fonction](#)

[CDate, fonction](#)

[CDBl, fonction](#)

[CInt, fonction](#)

[CSng, fonction](#)

[CStr, fonction](#)

[Int, Fix, fonctions](#)

Comparaison, constantes

[Référence du langage](#)

Voir aussi

[Couleur, constantes](#)

[Date/Heure, constantes](#)

[Format de date, constantes](#)

[Divers, constantes](#)

[MsgBox, constantes](#)

[Chaîne, constantes](#)

[3-états, constantes](#)

[VarType, constantes](#)



opérateur

Voir aussi

[Référence du langage](#)

[Opérateurs de concaténation](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

Const, instruction

[Référence du langage](#)

Voir aussi

[Dim, instruction](#)

[Function, instruction](#)

[Private, instruction](#)

[Public, instruction](#)

[Sub, instruction](#)

Cos, fonction

[Référence du langage](#)

Voir aussi

[Atn, fonction](#)

[Fonctions mathématiques dérivées](#)

[Sin, fonction](#)

[Tan, fonction](#)

CreateObject, fonction

[Référence du langage](#)

Voir aussi

[GetObject, fonction](#)

fonction

[Référence du langage](#)

Voir aussi

[CBool, fonction](#)

[CByte Function](#)

[CCur, fonction](#)

[CDate, fonction](#)

[CDbl, fonction](#)

[CInt, fonction](#)

[CLng, fonction](#)

[CStr, fonction](#)

fonction

[Référence du langage](#)

Voir aussi

[CBool, fonction](#)

[CByte Function](#)

[CCur, fonction](#)

[CDate, fonction](#)

[CDBl, fonction](#)

[CInt, fonction](#)

[CLng, fonction](#)

[CSng, fonction](#)

Date/Heure, constantes

[Référence du langage](#)

Voir aussi

[Couleur, constantes](#)

[Comparaison, constantes](#)

[Format de date, constantes](#)

[Divers, constantes](#)

[MsgBox, constantes](#)

[Chaîne, constantes](#)

[3-états, constantes](#)

[VarType, constantes](#)

Format de date, constantes

[Référence du langage](#)

Voir aussi

[Couleur, constantes](#)

[Comparaison, constantes](#)

[Date/Heure, constantes](#)

[Divers, constantes](#)

[MsgBox, constantes](#)

[Chaîne, constantes](#)

[3-états, constantes](#)

[VarType, constantes](#)

Microsoft® Visual Basic® Scripting Edition

Date, fonction

[Référence du langage](#)

Voir aussi

[CDate, fonction](#)

[Now, fonction](#)

[Time, fonction](#)

DateAdd, fonction

[Référence du langage](#)

Voir aussi

[DateDiff, fonction](#)

[DatePart, fonction](#)

DateDiff, fonction

[Référence du langage](#)

Voir aussi

[DateAdd, fonction](#)

[DatePart, fonction](#)

DatePart, fonction

[Référence du langage](#)

Voir aussi

[DateAdd, fonction](#)

[DateDiff, fonction](#)

DateSerial, fonction

[Référence du langage](#)

Voir aussi

[Date, fonction](#)

[DateValue, fonction](#)

[Day, fonction](#)

[Month, fonction](#)

[Now, fonction](#)

[TimeSerial, fonction](#)

[TimeValue, fonction](#)

[Weekday, fonction](#)

[Year, fonction](#)

DateValue, fonction

[Référence du langage](#)

Voir aussi

[CDate, fonction](#)

[DateSerial, fonction](#)

[Day, fonction](#)

[Month, fonction](#)

[Now, fonction](#)

[TimeSerial, fonction](#)

[TimeValue, fonction](#)

[Weekday, fonction](#)

[Year, fonction](#)

Day, fonction

[Référence du langage](#)

Voir aussi

[Date, fonction](#)

[Hour, fonction](#)

[Minute, fonction](#)

[Month, fonction](#)

[Now, fonction](#)

[Second, fonction](#)

[Weekday, fonction](#)

[Year, fonction](#)

Description, propriété

Voir aussi

[Référence du langage](#)

[Err, objet](#)

[HelpContext, propriété](#)

[HelpFile, propriété](#)

[Number, propriété](#)

[Source, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Description, propriété Application

[Référence du langage](#)

[Err. objet](#)

Microsoft® Visual Basic® Scripting Edition

Dim, instruction

[Référence du langage](#)

Voir aussi

[Private, instruction](#)

[Public, instruction](#)

[ReDim, instruction](#)

[Set, instruction](#)

opérateur

[Référence du langage](#)

Voir aussi

[*, opérateur](#)

[\, opérateur](#)

[Opérateurs arithmétiques](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

Do...Loop, instruction

[Référence du langage](#)

Voir aussi

[Exit, instruction](#)

[For...Next, instruction](#)

[While...Wend, instruction](#)

Eqv, opérateur

[Référence du langage](#)

Voir aussi

[Imp. opérateur](#)

[Opérateurs logiques](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

Microsoft® Visual Basic® Scripting Edition

Erase, instruction

[Référence du langage](#)

Voir aussi

[Dim, instruction](#)

[Nothing](#)

[ReDim, instruction](#)

Microsoft® Visual Basic® Scripting Edition

Err, objet

Voir aussi

[Référence du langage](#)

[Messages d'erreur](#)

[On Error, instruction](#)

objet (Propriétés)

[Description, propriété](#)

[HelpContext, propriété](#)

[HelpFile, propriété](#)

[Number, propriété](#)

[Source, propriété](#)

objet

Méthodes

[Clear, méthode](#)

[Raise, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Eval, fonction

Voir aussi

[Référence du langage](#)

[Execute, instruction](#)

Exécute, méthode

[Référence du langage](#)

Voir aussi

[Remplace, méthode](#)

[Test, méthode](#)

Execute, méthode Application

[Référence du langage](#)

[RegExp, object](#)

Execute, instruction

[Référence du langage](#)

Voir aussi

[Eval, fonction](#)

[ExecuteGlobal, instruction](#)

Exit, instruction

[Référence du langage](#)

Voir aussi

[Do...Loop, instruction](#)

[For Each...Next, instruction](#)

[For...Next, instruction](#)

[Function, instruction](#)

[Sub, instruction](#)

Exp, fonction

[Référence du langage](#)

Voir aussi

[Fonctions mathématiques dérivées](#)

[Log, fonction](#)



opérateur

Voir aussi

[Référence du langage](#)

[Opérateurs arithmétiques](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

Microsoft® Visual Basic® Scripting Edition

Filter, fonction

[Référence du langage](#)

Voir aussi

[Replace, fonction](#)

FirstIndex, propriété

Voir aussi

[Référence du langage](#)

[Length, propriété](#)

[Value, propriété](#)

Microsoft® Visual Basic® Scripting Edition

FirstIndex, propriété Application

[Référence du langage](#)

[Match, objet](#)

Microsoft® Visual Basic® Scripting Edition

Int, Fix, fonctions

[Référence du langage](#)

Voir aussi

[CInt, fonction](#)

[Round, fonction](#)

For...Next, instruction

[Référence du langage](#)

Voir aussi

[Do...Loop, instruction](#)

[Exit, instruction](#)

[For Each...Next, instruction](#)

[While...Wend, instruction](#)

Microsoft® Visual Basic® Scripting Edition **For**

Each...Next, instruction

[Référence du langage](#)

Voir aussi

[Do...Loop, instruction](#)

[Exit, instruction](#)

[For...Next, instruction](#)

[While...Wend, instruction](#)

FormatCurrency, fonction

[Référence du langage](#)

Voir aussi

[FormatDateTime, fonction](#)

[FormatNumber, fonction](#)

[FormatPercent, fonction](#)

FormatDateTime, fonction

[Référence du langage](#)

Voir aussi

[FormatCurrency, fonction](#)

[FormatNumber, fonction](#)

[FormatPercent, fonction](#)

FormatNumber, fonction

[Référence du langage](#)

Voir aussi

[FormatCurrency, fonction](#)

[FormatDateTime, fonction](#)

[FormatPercent, fonction](#)

FormatPercent, fonction

[Référence du langage](#)

Voir aussi

[FormatCurrency, fonction](#)

[FormatDateTime, fonction](#)

[FormatNumber, fonction](#)

Function, instruction

[Référence du langage](#)

Voir aussi

[Call, instruction](#)

[Dim, instruction](#)

[Exit, instruction](#)

[Nothing](#)

[Set, instruction](#)

[Sub, instruction](#)

Microsoft® Visual Basic® Scripting Edition

GetObject, fonction

[Référence du langage](#)

Voir aussi

[CreateObject, fonction](#)

GetRef, fonction

[Référence du langage](#)

Voir aussi

[Function, instruction](#)

[Set, instruction](#)

[Sub, instruction](#)

Global, propriété

Voir aussi

[Référence du langage](#)

[IgnoreCase, propriété](#)

[Pattern, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Global, propriété Application

[Référence du langage](#)

[RegExp, objet](#)

Microsoft® Visual Basic® Scripting Edition

Hex, fonction

[Référence du langage](#)

Voir aussi

[Oct, fonction](#)

HelpContext, propriété

Voir aussi

[Référence du langage](#)

[Description, propriété](#)

[HelpFile, propriété](#)

[Number, propriété](#)

[Source, propriété](#)

Microsoft® Visual Basic® Scripting Edition

HelpContext, propriété Application

[Référence du langage](#)

[Err. objet](#)

HelpFile, propriété

[Référence du langage](#)

Voir aussi

[Description, propriété](#)

[HelpContext, propriété](#)

[Number, propriété](#)

[Source, propriété](#)

Microsoft® Visual Basic® Scripting Edition

HelpFile, propriété Application

[Référence du langage](#)

[Err. objet](#)

Hour, fonction

[Référence du langage](#)

Voir aussi

[Day, fonction](#)

[Minute, fonction](#)

[Now, fonction](#)

[Second, fonction](#)

[Time, fonction](#)

IgnoreCase, propriété

Voir aussi

[Référence du langage](#)

[Global, propriété](#)

[Pattern, propriété](#)

IgnoreCase, propriété Application

[Référence du langage](#)

[RegExp, objet](#)

Imp, opérateur

[Référence du langage](#)

Voir aussi

[Eqv, opérateur](#)

[Opérateurs logiques](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

Initialize, événement

[Référence du langage](#)

Voir aussi

[Class, objet](#)

[Class, instruction](#)

[Terminate, événement](#)

Microsoft® Visual Basic® Scripting Edition

Initialize, événement Application

[Référence du langage](#)

[Class, objet](#)

InputBox, fonction

[Référence du langage](#)

Voir aussi

[MsgBox, fonction](#)

Microsoft® Visual Basic® Scripting Edition

Instr, fonction

[Référence du langage](#)

Voir aussi

[InstrRev, fonction](#)

InstrRev, fonction

[Référence du langage](#)

Voir aussi

[Instr, fonction](#)



opérateur

[Référence du langage](#)

Voir aussi

[*, opérateur](#)

[/, opérateur](#)

[Opérateurs arithmétiques](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

Microsoft® Visual Basic® Scripting Edition

Is, opérateur

Voir aussi

[Référence du langage](#)

[Opérateurs de comparaison](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

IsArray, fonction

[Référence du langage](#)

Voir aussi

[IsDate, fonction](#)

[IsEmpty, fonction](#)

[IsNull, fonction](#)

[IsNumeric, fonction](#)

[IsObject, fonction](#)

[VarType, fonction](#)

IsDate, fonction

[Référence du langage](#)

Voir aussi

[CDate, fonction](#)

[IsArray, fonction](#)

[IsEmpty, fonction](#)

[IsNull, fonction](#)

[IsNumeric, fonction](#)

[IsObject, fonction](#)

[VarType, fonction](#)

IsEmpty, fonction

[Référence du langage](#)

Voir aussi

[IsArray, fonction](#)

[IsDate, fonction](#)

[IsNull, fonction](#)

[IsNumeric, fonction](#)

[IsObject, fonction](#)

[VarType, fonction](#)

IsNull, fonction

[Référence du langage](#)

Voir aussi

[IsArray, fonction](#)

[IsDate, fonction](#)

[IsEmpty, fonction](#)

[IsNumeric, fonction](#)

[IsObject, fonction](#)

[VarType, fonction](#)

IsNumeric, fonction

[Référence du langage](#)

Voir aussi

[IsArray, fonction](#)

[IsDate, fonction](#)

[IsEmpty, fonction](#)

[IsNull, fonction](#)

[IsObject, fonction](#)

[VarType, fonction](#)

IsObject, fonction

[Référence du langage](#)

Voir aussi

[IsArray, fonction](#)

[IsDate, fonction](#)

[IsEmpty, fonction](#)

[IsNull, fonction](#)

[IsNumeric, fonction](#)

[Set, instruction](#)

[VarType, fonction](#)

Microsoft® Visual Basic® Scripting Edition

Join, fonction

[Référence du langage](#)

Voir aussi

[Split, fonction](#)

LBound, fonction

[Référence du langage](#)

Voir aussi

[Dim, instruction](#)

[ReDim, instruction](#)

[UBound, fonction](#)

Microsoft® Visual Basic® Scripting Edition

LCase, fonction

[Référence du langage](#)

Voir aussi

[UCase, fonction](#)

Microsoft® Visual Basic® Scripting Edition

Left, fonction

[Référence du langage](#)

Voir aussi

[Len, fonction](#)

[Mid, fonction](#)

[Right, fonction](#)

Microsoft® Visual Basic® Scripting Edition

Len, fonction

[Référence du langage](#)

Voir aussi

[InStr, fonction](#)

Length, propriété

Voir aussi

[Référence du langage](#)

[FirstIndex, propriété](#)

[Value, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Length, propriété Application

[Référence du langage](#)

[Match, objet](#)

Microsoft® Visual Basic® Scripting Edition

Log, fonction

[Référence du langage](#)

Voir aussi

[Fonctions mathématiques dérivées](#)

[Exp, fonction](#)

Microsoft® Visual Basic® Scripting Edition

LTrim, RTrim et Trim, fonctions

[Référence du langage](#)

Voir aussi

[Left, fonction](#)

[Right, fonction](#)

Microsoft® Visual Basic® Scripting Edition

Match, objet

Voir aussi

[Référence du langage](#)

[Matches, collection](#)

[RegExp, objet](#)

objet

Propriétés

[FirstIndex, propriété](#)

[Length, propriété](#)

[Value, propriété](#)

Matches, collection

[Référence du langage](#)

Voir aussi

[For Each...Next, instruction](#)

[Match, objet](#)

[RegExp, objet](#)

Matches, collection

Propriétés

[Référence du langage](#)

[Count, propriété](#)

[Item, propriété](#)

Mid, fonction

[Référence du langage](#)

Voir aussi

[Left, fonction](#)

[Len, fonction](#)

[LTrim, RTrim et Trim, fonctions](#)

[Right, fonction](#)

Minute, fonction

[Référence du langage](#)

Voir aussi

[Day, fonction](#)

[Hour, fonction](#)

[Now, fonction](#)

[Second, fonction](#)

[Time, fonction](#)

Divers, constantes

[Référence du langage](#)

Voir aussi

[Couleur, constantes](#)

[Comparaison, constantes](#)

[Date/Heure, constantes](#)

[Format de date, constantes](#)

[MsgBox, constantes](#)

[Chaîne, constantes](#)

[3-états, constantes](#)

[VarType, constantes](#)

Microsoft® Visual Basic® Scripting Edition

Mod, opérateur

[Référence du langage](#)

Voir aussi

[Opérateurs arithmétiques](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

Month, fonction

[Référence du langage](#)

Voir aussi

[Date, fonction](#)

[Day, fonction](#)

[Now, fonction](#)

[Weekday, fonction](#)

[Year, fonction](#)

Microsoft® Visual Basic® Scripting Edition

MonthName, fonction

[Référence du langage](#)

Voir aussi

[WeekDayName, fonction](#)

MsgBox, constantes

[Référence du langage](#)

Voir aussi

[Couleur, constantes](#)

[Comparaison, constantes](#)

[Date/Heure, constantes](#)

[Format de date, constantes](#)

[Divers, constantes](#)

[Chaîne, constantes](#)

[3-états, constantes](#)

[VarType, constantes](#)

MsgBox, fonction

[Référence du langage](#)

Voir aussi

[InputBox, fonction](#)



opérateur

Voir aussi

[Référence du langage](#)

[\, opérateur](#)

[Opérateurs arithmétiques](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

opérateur

Voir aussi

[+, opérateur](#)

[Opérateurs arithmétiques](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

Not, opérateur

[Référence du langage](#)

Voir aussi

[And, opérateur](#)

[Opérateurs logiques](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

[Or, opérateur](#)

[Xor, opérateur](#)

Now, fonction

[Référence du langage](#)

Voir aussi

[Date, fonction](#)

[Day, fonction](#)

[Hour, fonction](#)

[Minute, fonction](#)

[Month, fonction](#)

[Second, fonction](#)

[Time, fonction](#)

[Weekday, fonction](#)

[Year, fonction](#)

Nothing

[Référence du langage](#)

Voir aussi

[Dim, instruction](#)

[Set, instruction](#)

Number, propriété

[Référence du langage](#)

Voir aussi

[Description, propriété](#)

[HelpContext, propriété](#)

[HelpFile, propriété](#)

[Err, objet](#)

[Source, propriété](#)

[Messages d'erreur](#)

Number, propriété

Application

[Référence du langage](#)

[Err. objet](#)

Microsoft® Visual Basic® Scripting Edition

Oct, fonction

[Référence du langage](#)

Voir aussi

[Hex, fonction](#)

Microsoft® Visual Basic® Scripting Edition

On Error, instruction

[Référence du langage](#)

Voir aussi

[Err, objet](#)

[Exit, instruction](#)

Microsoft® Visual Basic® Scripting Edition

Priorité des opérateurs

[Référence du langage](#)

Voir aussi

[Is, opérateur](#)

[Résumé des opérateurs](#)

Or, opérateur

[Référence du langage](#)

Voir aussi

[And, opérateur](#)

[Opérateurs logiques](#)

[Not, opérateur](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

[Xor, opérateur](#)

Pattern, propriété

Voir aussi

[Référence du langage](#)

[Global, propriété](#)

[IgnoreCase, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Pattern, propriété Application

[Référence du langage](#)

[RegExp, objet](#)

Private, instruction

[Référence du langage](#)

Voir aussi

[Dim, instruction](#)

[Public, instruction](#)

[ReDim, instruction](#)

[Set, instruction](#)

Property Get, instruction

[Référence du langage](#)

Voir aussi

[Class, instruction](#)

[Dim, instruction](#)

[Exit, instruction](#)

[Function, instruction](#)

[Private, instruction](#)

[Property Let, instruction](#)

[Property Set, instruction](#)

[Public, instruction](#)

[Set, instruction](#)

[Sub, instruction](#)

Property Let, instruction

[Référence du langage](#)

Voir aussi

[Class, instruction](#)

[Dim, instruction](#)

[Exit, instruction](#)

[Function, instruction](#)

[Private, instruction](#)

[Property Get, instruction](#)

[Property Set, instruction](#)

[Public, instruction](#)

[Set, instruction](#)

[Sub, instruction](#)

Property Set, instruction

[Référence du langage](#)

Voir aussi

[Class, instruction](#)

[Dim, instruction](#)

[Exit, instruction](#)

[Function, instruction](#)

[Private, instruction](#)

[Property Get, instruction](#)

[Property Let, instruction](#)

[Public, instruction](#)

[Set, instruction](#)

[Sub, instruction](#)

Microsoft® Visual Basic® Scripting Edition

Public, instruction

[Référence du langage](#)

Voir aussi

[Dim, instruction](#)

[Private, instruction](#)

[ReDim, instruction](#)

[Set, instruction](#)

Raise, méthode

[Référence du langage](#)

Voir aussi

[Clear, méthode](#)

[Description, propriété](#)

[Err, objet](#)

[Number, propriété](#)

[Source, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Raise, méthode Application

[Référence du langage](#)

[Err. objet](#)

Randomize, instruction

[Référence du langage](#)

Voir aussi

[Rnd, fonction](#)

[Timer, fonction](#)

Microsoft® Visual Basic® Scripting Edition

ReDim, instruction

[Référence du langage](#)

Voir aussi

[Dim, instruction](#)

[Set, instruction](#)

Object

Voir aussi

[Match, objet](#)

[Matches, collection](#)

RegExp, objet

Méthodes

[Execute, méthode](#)

[Replace, méthode](#)

[Test, méthode](#)

RegExp, objet

Propriétés

[Global, propriété](#)

[IgnoreCase, propriété](#)

[Pattern, propriété](#)

Replace, fonction

[Référence du langage](#)

Voir aussi

[Filter, fonction](#)

Replace, méthode

Voir aussi

[Référence du langage](#)

[Execute, méthode](#)

[Test, méthode](#)

Replace, méthode Application

[Référence du langage](#)

[RegExp, objet](#)

Microsoft® Visual Basic® Scripting Edition

Right, fonction

[Référence du langage](#)

Voir aussi

[Left, fonction](#)

[Len, fonction](#)

[Mid, fonction](#)

Microsoft® Visual Basic® Scripting Edition

Rnd, fonction

[Référence du langage](#)

Voir aussi

[Randomize, instruction](#)

Microsoft® Visual Basic® Scripting Edition

Round, fonction

[Référence du langage](#)

Voir aussi

[Int, Fix, fonctions](#)

Microsoft® Visual Basic® Scripting Edition

ScriptEngine, fonction

[Référence du langage](#)

Voir aussi

[ScriptEngineBuildVersion, fonction](#)

[ScriptEngineMajorVersion, fonction](#)

[ScriptEngineMinorVersion, fonction](#)

ScriptEngineBuildVersion, fonction

[Référence du lan](#)

Voir aussi

[ScriptEngine, fonction](#)

[ScriptEngineMajorVersion, fonction](#)

[ScriptEngineMinorVersion, fonction](#)

ScriptEngineMajorVersion, fonction

[Référence du l](#)

Voir aussi

[ScriptEngine, fonction](#)

[ScriptEngineBuildVersion, fonction](#)

[ScriptEngineMinorVersion, fonction](#)

ScriptEngineMinorVersion, fonction

[Référence du 1](#)

Voir aussi

[ScriptEngine, fonction](#)

[ScriptEngineBuildVersion, fonction](#)

[ScriptEngineMajorVersion, fonction](#)

Second, fonction

[Référence du langage](#)

Voir aussi

[Day, fonction](#)

[Hour, fonction](#)

[Minute, fonction](#)

[Now, fonction](#)

[Time, fonction](#)

Microsoft® Visual Basic® Scripting Edition

Select Case, instruction

[Référence du langage](#)

Voir aussi

[If ... Then ... Else, instruction](#)

Set, instruction

[Référence du langage](#)

Voir aussi

[=, opérateur](#)

[Dim, instruction](#)

[GetRef, fonction](#)

[ReDim, instruction](#)

Microsoft® Visual Basic® Scripting Edition

Sgn, fonction

[Référence du langage](#)

Voir aussi

[Abs, fonction](#)

Sin, fonction

[Référence du langage](#)

Voir aussi

[Atn, fonction](#)

[Cos, fonction](#)

[Derived Math, fonctions](#)

[Tan, fonction](#)

Source, propriété

[Référence du langage](#)

Voir aussi

[Description, propriété](#)

[Err, objet](#)

[HelpContext, propriété](#)

[HelpFile, propriété](#)

[Number, propriété](#)

[On Error, instruction](#)

Microsoft® Visual Basic® Scripting Edition

Source, propriété Application

[Référence du langage](#)

[Err. objet](#)

Microsoft® Visual Basic® Scripting Edition

Space, fonction

[Référence du langage](#)

Voir aussi

[String, fonction](#)

Microsoft® Visual Basic® Scripting Edition

Split, fonction

[Référence du langage](#)

Voir aussi

[Join, fonction](#)

Chaîne, constantes

[Référence du langage](#)

Voir aussi

[Couleur, constantes](#)

[Comparaison, constantes](#)

[Date/Heure, constantes](#)

[Format de date, constantes](#)

[Divers, constantes](#)

[MsgBox, constantes](#)

[3-états, constantes](#)

[VarType, constantes](#)

Microsoft® Visual Basic® Scripting Edition

String, fonction

[Référence du langage](#)

Voir aussi

[Space, fonction](#)

Sub, instruction

[Référence du langage](#)

Voir aussi

[Call, instruction](#)

[Dim, instruction](#)

[Exit, instruction](#)

[Function, instruction](#)

Tan, fonction

[Référence du langage](#)

Voir aussi

[Atn, fonction](#)

[Cos, fonction](#)

[Derived Math, fonctions](#)

[Sin, fonction](#)

Terminate, événement

[Référence du langage](#)

Voir aussi

[Class, objet](#)

[Class, instruction](#)

[Initialize, événement](#)

Terminate, événement Application

[Référence du langage](#)

[Class, objet](#)

Test, méthode

Voir aussi

[Référence du langage](#)

[Execute, méthode](#)

[Replace, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Test, méthode Application

[Référence du langage](#)

[RegExp, objet](#)

Microsoft® Visual Basic® Scripting Edition

Time, fonction

[Référence du langage](#)

Voir aussi

[Date, fonction](#)

Timer, fonction

[Référence du langage](#)

Voir aussi

[Randomize, instruction](#)

TimeSerial, fonction

[Référence du langage](#)

Voir aussi

[DateSerial, fonction](#)

[DateValue, fonction](#)

[Hour, fonction](#)

[Minute, fonction](#)

[Now, fonction](#)

[Second, fonction](#)

[TimeValue, fonction](#)

TimeValue, fonction

[Référence du langage](#)

Voir aussi

[DateSerial, fonction](#)

[DateValue, fonction](#)

[Hour, fonction](#)

[Minute, fonction](#)

[Now, fonction](#)

[Second, fonction](#)

[TimeSerial, fonction](#)

TypeName, fonction

[Référence du langage](#)

Voir aussi

[IsArray, fonction](#)

[IsDate, fonction](#)

[IsEmpty, fonction](#)

[IsNull, fonction](#)

[IsNumeric, fonction](#)

[IsObject, fonction](#)

[VarType, fonction](#)

UBound, fonction

[Référence du langage](#)

Voir aussi

[Dim, instruction](#)

[LBound, fonction](#)

[ReDim, instruction](#)

Microsoft® Visual Basic® Scripting Edition

UCase, fonction

[Référence du langage](#)

Voir aussi

[LCase, fonction](#)

Value, propriété

Voir aussi

[Référence du langage](#)

[FirstIndex, propriété](#)

[Length, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Value, propriété Application

[Référence du langage](#)

[Match, objet](#)

VarType, constantes

[Référence du langage](#)

Voir aussi

[Couleur, constantes](#)

[Comparaison, constantes](#)

[Date/Heure, constantes](#)

[Format de date, constantes](#)

[Divers, constantes](#)

[MsgBox, constantes](#)

[Chaîne, constantes](#)

[3-états, constantes](#)

VarType, fonction

[Référence du langage](#)

Voir aussi

[IsArray, fonction](#)

[IsDate, fonction](#)

[IsEmpty, fonction](#)

[IsNull, fonction](#)

[IsNumeric, fonction](#)

[IsObject, fonction](#)

[TypeName, fonction](#)

Microsoft® Visual Basic® Scripting Edition

VBScript, constantes

[Référence du langage](#)

Voir aussi

[FileSystemObject, constantes](#)

Weekday, fonction

[Référence du langage](#)

Voir aussi

[Date, fonction](#)

[Day, fonction](#)

[Month, fonction](#)

[Now, fonction](#)

[Year, fonction](#)

WeekdayName, fonction

[Référence du langage](#)

Voir aussi

[MonthName, fonction](#)

While...Wend, instruction

[Référence du langage](#)

Voir aussi

[Do...Loop, instruction](#)

Microsoft® Visual Basic® Scripting Edition

With, instruction

[Référence du langage](#)

Voir aussi

[Set, instruction](#)

Xor, opérateur

[Référence du langage](#)

Voir aussi

[And, opérateur](#)

[Opérateurs logiques](#)

[Not, opérateur](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

[Or, opérateur](#)

Year, fonction

[Référence du langage](#)

Voir aussi

[Date, fonction](#)

[Day, fonction](#)

[Month, fonction](#)

[Now, fonction](#)

[Weekday, fonction](#)

Couleur, constantes

[Référence du langage](#)

Voir aussi

[Comparaison, constantes](#)

[Date/Heure, constantes](#)

[Format de date, constantes](#)

[Divers, constantes](#)

[MsgBox, constantes](#)

[Chaîne, constantes](#)

[3-états, constantes](#)

[VarType, constantes](#)

3-états, constantes

[Référence du langage](#)

Voir aussi

[Couleur, constantes](#)

[Comparaison, constantes](#)

[Date/Heure, constantes](#)

[Format de date, constantes](#)

[Divers, constantes](#)

[MsgBox, constantes](#)

[Chaîne, constantes](#)

[VarType, constantes](#)

Opérateurs de comparaison

[Référence du langage](#)

Voir aussi

[=, opérateur](#)

[Is, opérateur](#)

[Priorité des opérateurs](#)

[Résumé des opérateurs](#)

ExecuteGlobal, instruction

[Voir aussi](#)

Description

Exécute une ou plusieurs instructions spécifiées dans l'espace de nom global d'un script.

Syntaxe

ExecuteGlobal *instruction*

L'argument *instruction* requis est une [expression de chaîne](#) contenant une ou plusieurs instructions pour l'exécution. Incluez plusieurs instructions dans l'argument *instructions* en les séparant par deux points ou des sauts de ligne intégrés.

Notes

Dans VBScript, $x = y$ peut être interprété de deux manières. La première correspond à une instruction d'affectation, où la valeur de y est affectée à x . La seconde interprétation est une [expression](#) qui teste si x et y ont la même valeur. Si c'est le cas, *result* a la valeur **True** ; dans le cas contraire, *result* a la valeur **False**. L'instruction **ExecuteGlobal** utilise toujours la première interprétation, tandis que la méthode **Eval** adopte la seconde.

Remarque Dans Microsoft® JScript , il n'y a pas de risque de confondre l'affectation et la comparaison, car l'opérateur d'affectation (=) est différent de l'[opérateur de comparaison \(==\)](#).

Toutes les instructions utilisées avec **ExecuteGlobal** sont exécutées dans le nom d'espace global du script. Ceci permet au code d'être ajouté au programme afin que toutes les [procédures](#) puissent y accéder. Par exemple, une instruction **Class** de VBScript peut être définie au moment de l'exécution et les fonctions peuvent alors créer de nouvelles instances de la classe.

L'ajout de procédures et de classes au moment de l'exécution peut s'avérer très utile, mais crée le risque d'écraser des [variables](#) globales et fonctions existantes. Ceci pouvant causer d'importants problèmes de programmation, le plus grand soin est nécessaire lors de l'utilisation de l'instruction **ExecuteGlobal**. Si vous n'avez pas à accéder à une variable ou à une fonction en dehors d'une

procédure, utilisez l'instruction **Execute** qui affecte uniquement le nom d'espace de la fonction appelante.

L'exemple ci-dessous illustre l'utilisation de l'instruction **ExecuteGlobal** :

```
Dim X          ' Déclare X globalement.
X = "Global"   ' Affecte une valeur à la valeur X globale.
Sub Proc1      ' Déclare la procédure.
  Dim X        ' Déclare X localement.
  X = "Local"  ' Affecte une valeur à la valeur X locale.
               ' L'instruction Execute crée ici une procédure
               ' qui, lorsqu'elle est appelée, affiche X.
               ' Elle affiche la valeur X globale car Proc2
               ' hérite des éléments de portée globale.
  ExecuteGlobal "Sub Proc2: Print X: End Sub"
  Print Eval("X") ' Affiche la valeur X locale.
Proc2          ' Appelle Proc2 dans la portée globale,
               ' "Global" étant affiché.
End Sub
Proc2          ' Cette ligne provoque une erreur car
               ' Proc2 n'est pas disponible hors de Proc1.
Proc1          ' Appelle Proc1.
  Execute "Sub Proc2: Print X: End Sub"
Proc2          ' Cet appel réussit car Proc2
               ' est maintenant disponible globalement.
```

L'exemple ci-dessous montre comment l'instruction **ExecuteGlobal** peut être réécrite de sorte qu'il n'est pas nécessaire de placer la procédure entière entre guillemets :

```
S = "Sub Proc2" & vbCrLf
S = S & " Print X" & vbCrLf
S = S & "End Sub"
ExecuteGlobal S
```

GetLocale, fonction

[Voir aussi](#)

Description

Renvoie la valeur d'ID des paramètres régionaux en cours.

Syntaxe

GetLocale()

Notes

Les *paramètres régionaux* sont un ensemble d'informations indiquant les préférences d'un utilisateur quant à sa langue, son pays et ses conventions culturelles. Ces *paramètres* déterminent notamment la disposition des touches sur le clavier, l'ordre utilisé pour le tri alphabétique, ainsi que les formats à respecter pour les dates, les heures, les nombres et les devises.

La valeur de retour peut correspondre à toute valeur 32 bits acceptable indiquée dans le [tableau des ID de langue](#) :

L'exemple ci-dessous illustre l'utilisation de la fonction **GetLocale**. Pour utiliser ce code, collez l'exemple dans son intégralité entre les balises <BODY>; d'une page HTML standard.

```
Entrer la date au format Anglais-GB : <input type="text" id="UKI
Voici l'équivalent pour le format Anglais-US : <input type="text" i
<input type="button" value="Convert" id="button1"><p>
Entrer un prix au format allemand : &nbsp; <input type="text" id=
<p>
Voici l'équivalent au format Anglais-GB : <input type="text" id="1
<input type="button" value="Convert" id="button2"><p>
```

```
<script language="vbscript">
Dim currentLocale
' Get the current locale
currentLocale = GetLocale

Sub Button1_onclick
  Dim original
  original = SetLocale("en-gb")
  mydate = CDate(UKDate.value)
  ' Internet Explorer définit toujours les paramètres régionaux sur /
  ' la variable currentLocale pour les définir sur Anglais-US
  original = SetLocale(currentLocale)
  USDate.value = FormatDateTime(mydate,vbShortDate)
End Sub

Sub button2_onclick
  Dim original
  original = SetLocale("de")
  myvalue = CCur(GermanNumber.value)
  original = SetLocale("en-gb")
  USNumber.value = FormatCurrency(myvalue)
End Sub

</script>
```

Tableau des ID de langue (LCID)

[Référence du langage](#)
[Version 5](#)

[Voir aussi](#)

Description de la langue	Chaîne abrégée	Valeur hex.	Valeur décimale	Description de la langue	Chaîne abrégée	Valeur hex.	Valeur décimale
Africaans	af	0x0436	1078	Hébreu	he	0x040D	1037
Albanais	sq	0x041C	1052	Hindi	hi	0x0439	1081
Arabe - Émirats Arabes Unis	ar-ae	0x3801	14337	Hongrois	hu	0x040E	1038
Arabe - Bahreïn	ar-bh	0x3C01	15361	Islandais	is	0x040F	1039

Arabe - Algérie	ar-dz	0x1401	5121	Indonésien	in	0x0421	1057
Arabe - Égypte	ar-eg	0x0C01	3073	Italien - Italie	it	0x0410	1040
Arabe - Iraq	ar-iq	0x0801	2049	Italien - Suisse	it-ch	0x0810	2064
Arabe - Jordanie	ar-jo	0x2C01	11265	Japonais	ja	0x0411	1041
Arabe - Koweït	ar-kw	0x3401	13313	Coréen	ko	0x0412	1042
Arabe - Liban	ar-lb	0x3001	12289	Letton	lv	0x0426	1062
Arabe - Libye	ar-ly	0x1001	4097	Lituanien	lt	0x0427	1063
Arabe - Maroc	ar-ma	0x1801	6145	Macédonien (Ex Rép. Yougoslave de Macédoine)	mk	0x042F	1071
Arabe - Oman	ar-om	0x2001	8193	Malais - Malaisie	ms	0x043E	1086
Arabe - Qatar	ar-qa	0x4001	16385	Maltais	mt	0x043A	1082
Arabe - Arabie Saoudite	ar-sa	0x0401	1025	Norvégien - Bokmaal	no	0x0414	1044
Arabe - Syrie	ar-sy	0x2801	10241	Polonais	pl	0x0415	1045

Arabe - Tunisie	ar-tn	0x1C01	7169	Portugais - Portugal	pt	0x0816	2070
Arabe - Yémen	ar-ye	0x2401	9217	Portugais - Brésil	pt-br	0x0416	1046
Basque	eu	0x042D	1069	RhétO-roman	rm	0x0417	1047
Biélorusse	be	0x0423	1059	Roumain	ro	0x0418	1048
Bulgare	bg	0x0402	1026	Roumain - Moldavie	ro-mo	0x0818	2072
Catalan	ca	0x0403	1027	Russe	ru	0x0419	1049
Chinois	zh	0x0004	4	Russe - Moldavie	ru-mo	0x0819	2073
Chinois - Rép. Populaire de Chine	zh-cn	0x0804	2052	Serbe - Cyrillique	sr	0x0C1A	3098
Chinois - Hong Kong RAS	zh-hk	0x0C04	3076	Setswana	tn	0x0432	1074
Chinois - Singapour	zh-sg	0x1004	4100	Slovène	sl	0x0424	1060

Chinois - Taïwan	zh-tw	0x0404	1028	Slovaque	sk	0x041B	1051
Croate	hr	0x041A	1050	Sorbe	sb	0x042E	1070
Tchèque	cs	0x0405	1029	Espagnol - Espagne	es	0x040A	1034
Danois	da	0x0406	1030	Espagnol - Argentine	es-ar	0x2C0A	11274
Néerlandais	nl	0x0413	1043	Espagnol - Bolivie	es-bo	0x400A	16394
Néerlandais - Belgique	nl-be	0x0813	2067	Espagnol - Chili	es-cl	0x340A	13322
Anglais	en	0x0009	9	Espagnol - Colombie	es-co	0x240A	9226
Anglais - Australie	en-au	0x0C09	3081	Espagnol - Costa Rica	es-cr	0x140A	5130
Anglais - Belize	en-bz	0x2809	10249	Espagnol - République dominicaine	es-do	0x1C0A	7178
Anglais - Canada	en-			Espagnol -	es-		

	ca	0x1009	4105	Équateur	ec	0x300A	12298
Anglais - Irlande	en-ie	0x1809	6153	Espagnol - Guatemala	es-gt	0x100A	4106
Anglais - Jamaïque	en-jm	0x2009	8201	Espagnol - Honduras	es-hn	0x480A	18442
Anglais - Nouvelle-Zélande	en-nz	0x1409	5129	Espagnol - Mexique	es-mx	0x080A	2058
Anglais - Afrique du Sud	en-za	0x1C09	7177	Espagnol - Nicaragua	es-ni	0x4C0A	19466
Anglais - Trinidad	en-tt	0x2C09	11273	Espagnol - Panama	es-pa	0x180A	6154
Anglais - Royaume-Uni	en-gb	0x0809	2057	Espagnol - Pérou	es-pe	0x280A	10250
Anglais - États-Unis	en-us	0x0409	1033	Espagnol - Puerto Rico	es-pr	0x500A	20490
Estonien	et	0x0425	1061	Espagnol - Paraguay	es-py	0x3C0A	15370
Farsi (Persan)	fa	0x0429	1065	Espagnol - El Salvador	es-sv	0x440A	17418

Finnois	fi	0x040B	1035	Espagnol - Uruguay	es-uy	0x380A	14346
Féroen	fo	0x0438	1080	Espagnol - Venezuela	es-ve	0x200A	8202
Français - France	fr	0x040C	1036	Sutu	sx	0x0430	1072
Français - Belgique	fr-be	0x080C	2060	Suédois	sv	0x041D	1053
Français - Canada	fr-ca	0x0C0C	3084	Suédois - Finlande	sv-fi	0x081D	2077
Français - Luxembourg	fr-lu	0x140C	5132	Thaï	th	0x041E	1054
Français - Suisse	fr-ch	0x100C	4108	Turc	tr	0x041F	1055
Gaélique - Écosse	gd	0x043C	1084	Tsonga	ts	0x0431	1073
Allemand - Allemagne	de	0x0407	1031	Ukrainien	uk	0x0422	1058
Allemand - Autriche	de-at	0x0C07	3079	Ourdou - Pakistan	ur	0x0420	1056

Allemand - Liechtenstein	de-li	0x1407	5127	Vietnamien	vi	0x042A	1066
Allemand - Luxembourg	de- lu	0x1007	4103	Xhosa	xh	0x0434	1076
Allemand - Suisse	de- ch	0x0807	2055	Yiddish	ji	0x043D	1085
Grec	el	0x0408	1032	Zoulou	zu	0x0435	1077

SetLocale, fonction

[Voir aussi](#)

Description

Définit les paramètres régionaux globaux et renvoie les paramètres régionaux précédents.

Syntaxe

SetLocale(*lcid*)

L'argument *lcid* peut correspondre à toute valeur 32 bits acceptable ou à une abréviation identifiant de manière unique une langue parlée sur un territoire géographique bien précis. Les valeurs reconnues figurent dans le tableau des [ID de langue](#).

Notes

Si *lcid* est nulle, les paramètres régionaux sont définis de manière à correspondre à ceux en vigueur pour le système.

Les paramètres régionaux sont un ensemble d'informations indiquant les préférences d'un utilisateur quant à sa langue, son pays et ses conventions culturelles. Ils déterminent notamment la disposition des touches sur le clavier, l'ordre utilisé pour le tri alphabétique, ainsi que les formats à respecter pour les dates, les heures, les nombres et les devises.

L'exemple ci-dessous illustre l'utilisation de la fonction **SetLocale**. Pour utiliser ce code, collez l'exemple dans son intégralité entre les balises <BODY> d'une page HTML standard.

Entrer la date en format Anglais-GB : `<input type=`
Voici l'équivalent pour le format Anglais-US : `<inp`
`<input type="button" value="Convert" id="button1`
Entrer un prix au format allemand : ` ` `<input`
`<p>`

Voici l'équivalent au format Anglais-GB : `<input ty`
`<input type="button" value="Convert" id="button2`

```
<script language="vbscript">
```

```
Dim currentLocale
```

```
' Get the current locale
```

```
currentLocale = GetLocale
```

```
Sub Button1_onclick
```

```
Dim original
```

```
original = SetLocale("en-gb")
```

```
mydate = CDate(UKDate.value)
```

```
' Internet Explorer définit toujours les paramètres
```

```
' la variable currentLocale pour les définir sur An;
```

```
original = SetLocale(currentLocale)
```

```
USDate.value = FormatDateTime(mydate,vbShor
```

```
End Sub
```

```
Sub button2_onclick
```

```
Dim original
original = SetLocale("de")
myvalue = CCur(GermanNumber.value)
original = SetLocale("en-gb")
USNumber.value = FormatCurrency(myvalue)
End Sub
```

```
</script>
```

Add, méthode (Dictionary)

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Add, méthode \(Folders\)](#)

[Exists, méthode](#)

[Items, méthode](#)

[Keys, méthode](#)

[Remove, méthode](#)

[RemoveAll, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Add, méthode (Dictionary) Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Dictionary, objet](#)

Microsoft® Visual Basic® Scripting Edition

Add Method (Folders)

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Add, méthode \(Dictionary\)](#)

Microsoft® Visual Basic® Scripting Edition

Add, méthode (Folders) Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Folders, collection](#)

AtEndOfLine, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[AtEndOfStream, propriété](#)

[Column, propriété](#)

[Line, propriété](#)

Microsoft® Visual Basic® Scripting Edition

AtEndOfLine, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[TextStream, objet](#)

AtEndOfStream, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[AtEndOfLine, propriété](#)

[Column, propriété](#)

[Line, propriété](#)

Microsoft® Visual Basic® Scripting Edition

AtEndOfStream, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)
Version

[TextStream, objet](#)

Attributes, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[DateCreated, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Drive, propriété](#)

[Files, propriété](#)

[IsRootFolder, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[Path, propriété](#)

[ShortName, propriété](#)

[ShortPath, propriété](#)

[Size, propriété](#)

[SubFolders, propriété](#)

[Type, propriété](#)

Attributes, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)

[Folder, objet](#)

AvailableSpace, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[DriveLetter, propriété](#)

[DriveType, propriété](#)

[FileSystem, propriété](#)

[FreeSpace, propriété](#)

[IsReady, propriété](#)

[Path, propriété](#)

[RootFolder, propriété](#)

[SerialNumber, propriété](#)

[ShareName, propriété](#)

[TotalSize, propriété](#)

[VolumeName, propriété](#)

Microsoft® Visual Basic® Scripting Edition

AvailableSpace, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drive, objet](#)

BuildPath, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[GetAbsolutePathName, méthode](#)

[GetBaseName, méthode](#)

[GetDriveName, méthode](#)

[GetExtensionName, méthode](#)

[GetFileName, méthode](#)

[GetParentFolderName, méthode](#)

[GetTempName, méthode](#)

Microsoft® Visual Basic® Scripting Edition

BuildPath, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

Microsoft® Visual Basic® Scripting Edition

Close, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Read, méthode](#)

[ReadAll, méthode](#)

[ReadLine, méthode](#)

[Skip, méthode](#)

[SkipLine, méthode](#)

[Write, méthode](#)

[WriteLine, méthode](#)

[WriteBlankLines, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Close, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[TextStream, objet](#)

Column, propriété

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[AtEndOfLine, propriété](#)

[AtEndOfStream, propriété](#)

[Line, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Column, propriété

Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[TextStream, objet](#)

CompareMode, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Count, propriété](#)

[Item, propriété](#)

[Key, propriété](#)

Microsoft® Visual Basic® Scripting Edition

CompareMode, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Dictionary, objet](#)

Microsoft® Visual Basic® Scripting Edition

Copy, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[CopyFile, méthode](#)

[CopyFolder, méthode](#)

[Delete, méthode](#)

[Move, méthode](#)

[OpenAsTextStream, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Copy, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)
[Folder, objet](#)

CopyFile, méthode

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Copy, méthode](#)

[CopyFolder, méthode](#)

[CreateTextFile, méthode](#)

[DeleteFile, méthode](#)

[MoveFile, méthode](#)

Microsoft® Visual Basic® Scripting Edition

CopyFile, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

CopyFolder, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[CopyFile, méthode](#)

[Copy, méthode](#)

[CreateFolder, méthode](#)

[DeleteFolder, méthode](#)

[MoveFolder, méthode](#)

Microsoft® Visual Basic® Scripting Edition

CopyFolder, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

Microsoft® Visual Basic® Scripting Edition

Count, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[CompareMode, propriété](#)

[Item, propriété](#)

[Key, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Count, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Dictionary, objet](#)

[Drives, collection](#)

[Files, collection](#)

[Folders, collection](#)

[Matches, collection](#)

CreateFolder, méthode

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[CopyFolder, méthode](#)
[DeleteFolder, méthode](#)
[MoveFolder, méthode](#)

Microsoft® Visual Basic® Scripting Edition

CreateFolder, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

CreateTextFile, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[CreateFolder, méthode](#)

[OpenAsTextStream, méthode](#)

[OpenTextFile, méthode](#)

Microsoft® Visual Basic® Scripting Edition

CreateTextFile, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

[Folder, objet](#)

DateCreated, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Drive, propriété](#)

[Files, propriété](#)

[IsRootFolder, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[Path, propriété](#)

[ShortName, propriété](#)

[ShortPath, propriété](#)

[Size, propriété](#)

[SubFolders, propriété](#)

[Type, propriété](#)

Microsoft® Visual Basic® Scripting Edition

DateCreated, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)

[Folder, objet](#)

DateLastAccessed, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[DateCreated, propriété](#)

[DateLastModified, propriété](#)

[Drive, propriété](#)

[Files, propriété](#)

[IsRootFolder, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[Path, propriété](#)

[ShortName, propriété](#)

[ShortPath, propriété](#)

[Size, propriété](#)

[SubFolders, propriété](#)

[Type, propriété](#)

DateLastAccessed, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)

[Folder, objet](#)

DateLastModified, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)
[DateCreated, propriété](#)
[DateLastAccessed, propriété](#)
[Drive, propriété](#)
[Files, propriété](#)
[IsRootFolder, propriété](#)
[Name, propriété](#)
[ParentFolder, propriété](#)
[Path, propriété](#)
[ShortName, propriété](#)
[ShortPath, propriété](#)
[Size, propriété](#)
[SubFolders, propriété](#)
[Type, propriété](#)

DateLastModified, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)

[Folder, objet](#)

Microsoft® Visual Basic® Scripting Edition

Delete, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Copy, méthode](#)

[DeleteFile, méthode](#)

[DeleteFolder, méthode](#)

[Move, méthode](#)

[OpenAsTextStream, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Delete, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)
[Folder, objet](#)

DeleteFile, méthode

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[CopyFile, méthode](#)
[CreateTextFile, méthode](#)
[Delete, méthode](#)
[DeleteFolder, méthode](#)
[MoveFile, méthode](#)

Microsoft® Visual Basic® Scripting Edition

DeleteFile, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

DeleteFolder, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[CopyFolder, méthode](#)
[CreateFolder, méthode](#)
[Delete, méthode](#)
[DeleteFile, méthode](#)
[MoveFolder, méthode](#)

Microsoft® Visual Basic® Scripting Edition

DeleteFolder, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

Dictionary, objet

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[FileSystemObject, objet](#)

[TextStream, objet](#)

Dictionary, objet

Propriétés

[CompareMode, propriété](#)

[Count, propriété](#)

[Item, propriété](#)

[Key, propriété](#)

Dictionary, objet

Méthodes

[Référence de la bibliothèque
d'exécution Scripting](#)

[Add, méthode \(Dictionary\)](#)

[Exists, méthode](#)

[Items, méthode](#)

[Keys, méthode](#)

[Remove, méthode](#)

[RemoveAll, méthode](#)

Drive, objet

Voir aussi

[Drives, collection](#)

[File, objet](#)

[Files, collection](#)

[Folder, objet](#)

[Folders, collection](#)

[GetDrive, méthode](#)

[Référence de la bibliothèque
d'exécution Scripting](#)

Drive,

objet

Propriétés

[Référence de la bibliothèque
d'exécution Scripting](#)

[AvailableSpace, propriété](#)

[DriveLetter, propriété](#)

[DriveType, propriété](#)

[FileSystem, propriété](#)

[FreeSpace, propriété](#)

[IsReady, propriété](#)

[Path, propriété](#)

[RootFolder, propriété](#)

[SerialNumber, propriété](#)

[ShareName, propriété](#)

[TotalSize, propriété](#)

[VolumeName, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Drive,

objet

Méthodes

[Référence de la bibliothèque
d'exécution Scripting](#)

L'objet **Drive** n'a pas de méthode.

Drive, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[DateCreated, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Files, propriété](#)

[IsRootFolder, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[Path, propriété](#)

[ShortName, propriété](#)

[ShortPath, propriété](#)

[Size, propriété](#)

[SubFolders, propriété](#)

[Type, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Drive, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)

[Folder, objet](#)

DriveExists, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Drive Object](#)

[Drives Collection](#)

[FileExists, méthode](#)

[FolderExists, méthode](#)

[GetDrive, méthode](#)

[GetDriveName, méthode](#)

[IsReady Property](#)

Microsoft® Visual Basic® Scripting Edition

DriveExists, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

DriveLetter, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[AvailableSpace, propriété](#)

[DriveType, propriété](#)

[FileSystem, propriété](#)

[FreeSpace, propriété](#)

[IsReady, propriété](#)

[Path, propriété](#)

[RootFolder, propriété](#)

[SerialNumber, propriété](#)

[ShareName, propriété](#)

[TotalSize, propriété](#)

[VolumeName, propriété](#)

Microsoft® Visual Basic® Scripting Edition

DriveLetter, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drive, objet](#)

Microsoft® Visual Basic® Scripting Edition

Drives, collection

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Drive, objet](#)

[Drives, propriété](#)

[File, objet](#)

[Files, collection](#)

[Folder, objet](#)

[Folders, collection](#)

Microsoft® Visual Basic® Scripting Edition

Drives, collection Propriétés

[Référence de la bibliothèque
d'exécution Scripting](#)

[Count, propriété](#)

[Item, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Drives, collection Méthodes

[Référence de la bibliothèque
d'exécution Scripting](#)

La collection **Drives** n'a pas de méthode.

Microsoft® Visual Basic® Scripting Edition

Drives, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drives, collection](#)

[Files, propriété](#)

[SubFolders, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Drives, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

DriveType, constantes

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Comparaison, constantes](#)

[Attributs de fichier, constantes](#)

[Entrée/Sortie de fichier, constantes](#)

[SpecialFolder, constantes](#)

[3-états, constantes](#)

DriveType, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[AvailableSpace, propriété](#)

[DriveLetter, propriété](#)

[FileSystem, propriété](#)

[FreeSpace, propriété](#)

[IsReady, propriété](#)

[Path, propriété](#)

[RootFolder, propriété](#)

[SerialNumber, propriété](#)

[ShareName, propriété](#)

[TotalSize, propriété](#)

[VolumeName, propriété](#)

Microsoft® Visual Basic® Scripting Edition

DriveType, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drive, objet](#)

Exists, méthode

[Référence du langage](#)

Voir aussi

[Add, méthode \(Dictionary\)](#)

[Items, méthode](#)

[Keys, méthode](#)

[Remove, méthode](#)

[RemoveAll, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Exists,

méthode

Application

[Référence du langage](#)

[Dictionary, objet](#)

FileAttribute, constantes

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Comparaison, constantes](#)

[DriveType, constantes](#)

[Entrée/Sortie de fichier, constantes](#)

[SpecialFolder, constantes](#)

[3-états, constantes](#)

Entrée/Sortie de fichier, constantes

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Comparaison, constantes](#)

[DriveType, constantes](#)

[Attributs de fichier, constantes](#)

[SpecialFolder, constantes](#)

[3-états, constantes](#)

FileExists, méthode

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[DriveExists, méthode](#)

[FolderExists, méthode](#)

[GetFile, méthode](#)

[GetFileName, méthode](#)

Microsoft® Visual Basic® Scripting Edition

FileExists, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

File, objet

Voir aussi

[Drive, objet](#)

[Drives, collection](#)

[Files, collection](#)

[Folder, objet](#)

[Folders, collection](#)

[Référence de la bibliothèque
d'exécution Scripting](#)

File,

objet

Propriétés

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[DateCreated, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Drive, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[Path, propriété](#)

[ShortName, propriété](#)

[ShortPath, propriété](#)

[Size, propriété](#)

[Type, propriété](#)

File,

objet

Méthodes

[Référence de la bibliothèque
d'exécution Scripting](#)

[Copy, méthode](#)

[Delete, méthode](#)

[Move, méthode](#)

[OpenAsTextStream, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Files, collection

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Drive, objet](#)

[Drives, collection](#)

[File, objet](#)

[Folder, objet](#)

[Folders, collection](#)

Microsoft® Visual Basic® Scripting Edition

Files, collection Propriétés

[Référence de la bibliothèque
d'exécution Scripting](#)

[Count, propriété](#)

[Item, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Files, collection Méthodes

[Référence de la bibliothèque
d'exécution Scripting](#)

La collection **Files** n'a pas de méthode.

Files, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[DateCreated, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Drives, propriété](#)

[IsRootFolder, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[Path, propriété](#)

[ShortName, propriété](#)

[ShortPath, propriété](#)

[Size, propriété](#)

[SubFolders, propriété](#)

[Type, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Files, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Folder, objet](#)

FileSystemObject, objet

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[CreateObject, fonction](#)

[Dictionary, objet](#)

[Drive, objet](#)

[Drives, collection](#)

[File, objet](#)

[FileSystem, propriété](#)

[Files, collection](#)

[Folder, objet](#)

[Folders, collection](#)

[TextStream, objet](#)

Microsoft® Visual Basic® Scripting Edition

FileSystemObject, objet

Propriétés

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drives, propriété](#)

FileSystemObject, objet

Méthodes

[Référence de la bibliothèque
d'exécution Scripting](#)

[BuildPath, méthode](#)
[CopyFile, méthode](#)
[CopyFolder, méthode](#)
[CreateFolder, méthode](#)
[CreateTextFile, méthode](#)
[DeleteFile, méthode](#)
[DeleteFolder, méthode](#)
[DriveExists, méthode](#)
[FileExists, méthode](#)
[FolderExists, méthode](#)
[GetAbsolutePath, méthode](#)
[GetBaseName, méthode](#)
[GetDrive, méthode](#)
[GetDriveName, méthode](#)
[GetExtensionName, méthode](#)
[GetFile, méthode](#)
[GetFileName, méthode](#)
[GetFolder, méthode](#)
[GetParentFolderName, méthode](#)
[GetSpecialFolder, méthode](#)
[GetTempName, méthode](#)
[MoveFile, méthode](#)
[MoveFolder, méthode](#)
[OpenTextFile, méthode](#)

Microsoft® Visual Basic® Scripting Edition

FileSystemObject, constantes

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[VBScript, constantes](#)

FileSystem, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[AvailableSpace, propriété](#)

[DriveLetter, propriété](#)

[DriveType, propriété](#)

[FileSystemObject, objet](#)

[FreeSpace, propriété](#)

[IsReady, propriété](#)

[Path, propriété](#)

[RootFolder, propriété](#)

[SerialNumber, propriété](#)

[ShareName, propriété](#)

[TotalSize, propriété](#)

[VolumeName, propriété](#)

Microsoft® Visual Basic® Scripting Edition

FileSystem, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drive, objet](#)

Folder, objet

Voir aussi

[Drive, objet](#)

[Drives, collection](#)

[File, objet](#)

[Files, collection](#)

[Folders, collection](#)

[Référence de la bibliothèque
d'exécution Scripting](#)

Folder,

objet

Propriétés

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[DateCreated, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Drive, propriété](#)

[Files, propriété](#)

[IsRootFolder, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[Path, propriété](#)

[ShortName, propriété](#)

[ShortPath, propriété](#)

[Size, propriété](#)

[SubFolders, propriété](#)

[Type, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Folder,

objet

Méthodes

[Référence de la bibliothèque
d'exécution Scripting](#)

[Copy, méthode](#)

[Delete, méthode](#)

[Move, méthode](#)

[CreateTextFile, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Folders, collection

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Drive, objet](#)

[Drives, collection](#)

[File, objet](#)

[Files, collection](#)

[Folder, objet](#)

[SubFolders, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Folders, collection Propriétés

[Référence de la bibliothèque
d'exécution Scripting](#)

[Count, propriété](#)

[Item, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Folders, collection Méthodes

[Référence de la bibliothèque
d'exécution Scripting](#)

[Add, méthode](#)

FolderExists, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[DriveExists, méthode](#)

[FileExists, méthode](#)

[GetFolder, méthode](#)

[GetParentFolderName, méthode](#)

Microsoft® Visual Basic® Scripting Edition

FolderExists, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

FreeSpace, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[AvailableSpace, propriété](#)

[DriveLetter, propriété](#)

[DriveType, propriété](#)

[FileSystem, propriété](#)

[IsReady, propriété](#)

[Path, propriété](#)

[RootFolder, propriété](#)

[SerialNumber, propriété](#)

[ShareName, propriété](#)

[TotalSize, propriété](#)

[VolumeName, propriété](#)

Microsoft® Visual Basic® Scripting Edition

FreeSpace, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drive, objet](#)

GetAbsolutePathName, [Référence de la bibliothèque d'exécution Scrip](#) méthode

Voir aussi

[GetBaseName, méthode](#)

[GetDrive, méthode](#)

[GetDriveName, méthode](#)

[GetExtensionName, méthode](#)

[GetFile, méthode](#)

[GetFileName, méthode](#)

[GetFileVersion, méthode](#)

[GetFolder, méthode](#)

[GetParentFolderName, méthode](#)

[GetSpecialFolder, méthode](#)

[GetTempName, méthode](#)

GetAbsolutePathName, [Référence de la bibliothèque d'exécution Scrip](#)

méthode

Application

[FileSystemObject, objet](#)

GetBaseName, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[GetAbsolutePathName, méthode](#)

[GetDrive, méthode](#)

[GetDriveName, méthode](#)

[GetExtensionName, méthode](#)

[GetFile, méthode](#)

[GetFileName, méthode](#)

[GetFileVersion, méthode](#)

[GetFolder, méthode](#)

[GetParentFolderName, méthode](#)

[GetSpecialFolder, méthode](#)

[GetTempName, méthode](#)

GetBaseName, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

GetDrive, méthode

Voir aussi

[GetAbsolutePathName, méthode](#)

[GetBaseName, méthode](#)

[GetDriveName, méthode](#)

[GetExtensionName, méthode](#)

[GetFile, méthode](#)

[GetFileName, méthode](#)

[GetFileVersion, méthode](#)

[GetFolder, méthode](#)

[GetParentFolderName, méthode](#)

[GetSpecialFolder, méthode](#)

[GetTempName, méthode](#)

Microsoft® Visual Basic® Scripting Edition

GetDrive, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

GetDriveName, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[GetAbsolutePathName, méthode](#)

[GetBaseName, méthode](#)

[GetDrive, méthode](#)

[GetExtensionName, méthode](#)

[GetFile, méthode](#)

[GetFileName, méthode](#)

[GetFileVersion, méthode](#)

[GetFolder, méthode](#)

[GetParentFolderName, méthode](#)

[GetSpecialFolder, méthode](#)

[GetTempName, méthode](#)

Microsoft® Visual Basic® Scripting Edition

GetDriveName, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

GetExtensionName, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[GetAbsolutePathName, méthode](#)

[GetBaseName, méthode](#)

[GetDrive, méthode](#)

[GetDriveName, méthode](#)

[GetFile, méthode](#)

[GetFileName, méthode](#)

[GetFileVersion, méthode](#)

[GetFolder, méthode](#)

[GetParentFolderName, méthode](#)

[GetSpecialFolder, méthode](#)

[GetTempName, méthode](#)

GetExtensionName, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

GetFile, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[GetAbsolutePathName, méthode](#)

[GetBaseName, méthode](#)

[GetDrive, méthode](#)

[GetDriveName, méthode](#)

[GetExtensionName, méthode](#)

[GetFileName, méthode](#)

[GetFileVersion, méthode](#)

[GetFolder, méthode](#)

[GetParentFolderName, méthode](#)

[GetSpecialFolder, méthode](#)

[GetTempName, méthode](#)

Microsoft® Visual Basic® Scripting Edition

GetFile, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

GetFileName, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[GetAbsolutePathName, méthode](#)

[GetBaseName, méthode](#)

[GetDrive, méthode](#)

[GetDriveName, méthode](#)

[GetExtensionName, méthode](#)

[GetFile, méthode](#)

[GetFileVersion, méthode](#)

[GetFolder, méthode](#)

[GetParentFolderName, méthode](#)

[GetSpecialFolder, méthode](#)

[GetTempName, méthode](#)

Microsoft® Visual Basic® Scripting Edition

GetFileName, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

GetFileVersion, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[GetAbsolutePathName, méthode](#)

[GetBaseName, méthode](#)

[GetDrive, méthode](#)

[GetDriveName, méthode](#)

[GetExtensionName, méthode](#)

[GetFile, méthode](#)

[GetFileName, méthode](#)

[GetFolder, méthode](#)

[GetParentFolderName, méthode](#)

[GetSpecialFolder, méthode](#)

[GetTempName, méthode](#)

Microsoft® Visual Basic® Scripting Edition

GetFileVersion, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

GetFolder, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[GetAbsolutePathName, méthode](#)

[GetBaseName, méthode](#)

[GetDrive, méthode](#)

[GetDriveName, méthode](#)

[GetExtensionName, méthode](#)

[GetFile, méthode](#)

[GetFileName, méthode](#)

[GetFileVersion, méthode](#)

[GetParentFolderName, méthode](#)

[GetSpecialFolder, méthode](#)

[GetTempName, méthode](#)

Microsoft® Visual Basic® Scripting Edition

GetFolder, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

GetParentFolderName, [Référence de la bibliothèque d'exécution Scrip](#) méthode

Voir aussi

[GetAbsolutePathName, méthode](#)

[GetBaseName, méthode](#)

[GetDrive, méthode](#)

[GetDriveName, méthode](#)

[GetExtensionName, méthode](#)

[GetFile, méthode](#)

[GetFileName, méthode](#)

[GetFileVersion, méthode](#)

[GetFolder, méthode](#)

[GetSpecialFolder, méthode](#)

[GetTempName, méthode](#)

GetParentFolderName, [Référence de la bibliothèque d'exécution Scrip](#)

méthode

Application

[FileSystemObject](#), objet

GetSpecialFolder, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[GetAbsolutePathName, méthode](#)

[GetBaseName, méthode](#)

[GetDrive, méthode](#)

[GetDriveName, méthode](#)

[GetExtensionName, méthode](#)

[GetFile, méthode](#)

[GetFileName, méthode](#)

[GetFileVersion, méthode](#)

[GetFolder, méthode](#)

[GetParentFolderName, méthode](#)

[GetTempName, méthode](#)

Microsoft® Visual Basic® Scripting Edition

GetSpecialFolder, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

GetTempName, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[GetAbsolutePathName, méthode](#)

[GetBaseName, méthode](#)

[GetDrive, méthode](#)

[GetDriveName, méthode](#)

[GetExtensionName, méthode](#)

[GetFile, méthode](#)

[GetFileName, méthode](#)

[GetFileVersion, méthode](#)

[GetFolder, méthode](#)

[GetParentFolderName, méthode](#)

[GetSpecialFolder, méthode](#)

Microsoft® Visual Basic® Scripting Edition

GetTempName, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

IsReady, propriété

Voir aussi

[AvailableSpace, propriété](#)

[DriveLetter, propriété](#)

[DriveType, propriété](#)

[FileSystem, propriété](#)

[FreeSpace, propriété](#)

[Path, propriété](#)

[RootFolder, propriété](#)

[SerialNumber, propriété](#)

[ShareName, propriété](#)

[TotalSize, propriété](#)

[VolumeName, propriété](#)

Microsoft® Visual Basic® Scripting Edition

IsReady, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drive, objet](#)

IsRootFolder, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[DateCreated, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Drive, propriété](#)

[Files, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[Path, propriété](#)

[ShortName, propriété](#)

[ShortPath, propriété](#)

[Size, propriété](#)

[SubFolders, propriété](#)

[Type, propriété](#)

Microsoft® Visual Basic® Scripting Edition

IsRootFolder, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Folder, objet](#)

Microsoft® Visual Basic® Scripting Edition

Item, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[CompareMode, propriété](#)

[Count, propriété](#)

[Key, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Item, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Dictionary, objet](#)

[Drives, collection](#)

[Files, collection](#)

[Folders, collection](#)

[Matches, collection](#)

Items, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Add, méthode \(Dictionary\)](#)

[Exists, méthode](#)

[Keys, méthode](#)

[Remove, méthode](#)

[RemoveAll, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Items, méthode

Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Dictionary, objet](#)

Microsoft® Visual Basic® Scripting Edition

Key, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[CompareMode, propriété](#)

[Count, propriété](#)

[Item, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Key, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Dictionary, objet](#)

Glossaire

FileSystemObject

[Référence de la bibliothèque d'exécution Scripting](#)

bibliothèque de types

Fichier ou composant d'un autre fichier contenant des descriptions standard d'objets, propriétés et méthodes exposés.

collection

Objet qui contient un ensemble d'objets associés. La position de l'objet dans la collection peut changer lorsqu'une modification se produit dans la collection. Par conséquent, la position d'un objet spécifique dans la collection varie.

erreur d'exécution

Erreur qui se produit au cours de l'exécution du code. Une erreur d'exécution se produit quand une instruction tente une opération incorrecte.

expression de chaîne

Toute expression produisant une séquence de caractères contigus. Une expression de chaîne peut contenir une fonction qui renvoie une chaîne, un caractère de chaîne, une constante de chaîne ou une variable de chaîne.

tableau

Fichier ou composant d'un autre fichier contenant des descriptions standard d'objets, propriétés et méthodes exposés.

Microsoft® Visual Basic® Scripting Edition

Keys, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Add, méthode \(Dictionary\)](#)

[Exists, méthode](#)

[Items, méthode](#)

[Remove, méthode](#)

[RemoveAll, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Keys, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Dictionary, objet](#)

Microsoft® Visual Basic® Scripting Edition

Line, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[AtEndOfStream, propriété](#)

[AtEndOfLine, propriété](#)

[Column, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Line, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)
Version

[TextStream, objet](#)

Microsoft® Visual Basic® Scripting Edition

Move, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Copy, méthode](#)

[Delete, méthode](#)

[MoveFile, méthode](#)

[MoveFolder, méthode](#)

[OpenAsTextStream, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Move, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)

[Folder, objet](#)

MoveFile, méthode

Voir aussi

[CopyFile, méthode](#)

[DeleteFile, méthode](#)

[GetFile, méthode](#)

[GetFileName, méthode](#)

[Move, méthode](#)

[MoveFolder, méthode](#)

[OpenTextFile, méthode](#)

Microsoft® Visual Basic® Scripting Edition

MoveFile, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

MoveFolder, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[CopyFolder, méthode](#)

[CreateFolder, méthode](#)

[DeleteFolder, méthode](#)

[GetFolder, méthode](#)

[GetParentFolderName, méthode](#)

[Move, méthode](#)

[MoveFile, méthode](#)

Microsoft® Visual Basic® Scripting Edition

MoveFolder, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

Name, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)
[DateCreated, propriété](#)
[DateLastAccessed, propriété](#)
[DateLastModified, propriété](#)
[Drive, propriété](#)
[Files, propriété](#)
[IsRootFolder, propriété](#)
[ParentFolder, propriété](#)
[Path, propriété](#)
[ShortName, propriété](#)
[ShortPath, propriété](#)
[Size, propriété](#)
[SubFolders, propriété](#)
[Type, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Name, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)
[Folder, objet](#)

OpenAsTextStream, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Copy, méthode](#)

[CreateTextFile, méthode](#)

[Delete, méthode](#)

[Move, méthode](#)

[OpenTextFile, méthode](#)

Microsoft® Visual Basic® Scripting Edition

OpenAsTextStream, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)

OpenTextFile, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[OpenAsTextStream, méthode](#)

[CreateTextFile, méthode](#)

Microsoft® Visual Basic® Scripting Edition

OpenTextFile, méthode

Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[FileSystemObject, objet](#)

ParentFolder, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[DateCreated, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Drive, propriété](#)

[Files, propriété](#)

[IsRootFolder, propriété](#)

[Name, propriété](#)

[Path, propriété](#)

[ShortName, propriété](#)

[ShortPath, propriété](#)

[Size, propriété](#)

[SubFolders, propriété](#)

[Type, propriété](#)

ParentFolder, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)

[Folder, objet](#)

Path, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[AvailableSpace, propriété](#)

[DateCreated, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Drive, propriété](#)

[DriveLetter, propriété](#)

[DriveType, propriété](#)

[Files, propriété](#)

[FileSystem, propriété](#)

[FreeSpace, propriété](#)

[IsReady, propriété](#)

[IsRootFolder, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[RootFolder, propriété](#)

[SerialNumber, propriété](#)

[ShareName, propriété](#)

[ShortName, propriété](#)

[ShortPath, propriété](#)

[Size, propriété](#)

[SubFolders, propriété](#)

[TotalSize, propriété](#)

[Type, propriété](#)

[VolumeName, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Path, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drive, objet](#)

[File, objet](#)

[Folder, objet](#)

Microsoft® Visual Basic® Scripting Edition

Read, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Close, méthode](#)

[ReadAll, méthode](#)

[ReadLine, méthode](#)

[Skip, méthode](#)

[SkipLine, méthode](#)

[Write, méthode](#)

[WriteLine, méthode](#)

[WriteBlankLines, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Read, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[TextStream, objet](#)

ReadAll, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Close, méthode](#)

[Read, méthode](#)

[ReadLine, méthode](#)

[Skip, méthode](#)

[SkipLine, méthode](#)

[Write, méthode](#)

[WriteLine, méthode](#)

[WriteBlankLines, méthode](#)

Microsoft® Visual Basic® Scripting Edition

ReadAll, méthode

Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[TextStream, objet](#)

ReadLine, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Close, méthode](#)

[Read, méthode](#)

[ReadAll, méthode](#)

[Skip, méthode](#)

[SkipLine, méthode](#)

[Write, méthode](#)

[WriteLine, méthode](#)

[WriteBlankLines, méthode](#)

Microsoft® Visual Basic® Scripting Edition

ReadLine, méthode

Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[TextStream, objet](#)

Remove, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Add, méthode \(Dictionary\)](#)

[Exists, méthode](#)

[Items, méthode](#)

[Keys, méthode](#)

[RemoveAll, méthode](#)

Microsoft® Visual Basic® Scripting Edition

RemoveAll, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Dictionary, objet](#)

RemoveAll, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Add, méthode \(Dictionary\)](#)

[Exists, méthode](#)

[Items, méthode](#)

[Keys, méthode](#)

[Remove, méthode](#)

Microsoft® Visual Basic® Scripting Edition

RemoveAll, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Dictionary, objet](#)

RootFolder, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[AvailableSpace, propriété](#)

[DriveLetter, propriété](#)

[DriveType, propriété](#)

[FileSystem, propriété](#)

[FreeSpace, propriété](#)

[IsReady, propriété](#)

[Path, propriété](#)

[SerialNumber, propriété](#)

[ShareName, propriété](#)

[TotalSize, propriété](#)

[VolumeName, propriété](#)

Microsoft® Visual Basic® Scripting Edition

RootFolder, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drive, objet](#)

SerialNumber, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[AvailableSpace, propriété](#)

[DriveLetter, propriété](#)

[DriveType, propriété](#)

[FileSystem, propriété](#)

[FreeSpace, propriété](#)

[IsReady, propriété](#)

[Path, propriété](#)

[RootFolder, propriété](#)

[ShareName, propriété](#)

[TotalSize, propriété](#)

[VolumeName, propriété](#)

Microsoft® Visual Basic® Scripting Edition

SerialNumber, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drive, objet](#)

ShareName, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[AvailableSpace, propriété](#)

[DriveLetter, propriété](#)

[DriveType, propriété](#)

[FileSystem, propriété](#)

[FreeSpace, propriété](#)

[IsReady, propriété](#)

[Path, propriété](#)

[RootFolder, propriété](#)

[SerialNumber, propriété](#)

[TotalSize, propriété](#)

[VolumeName, propriété](#)

ShareName, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drive, objet](#)

ShortName, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[DateCreated, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Drive, propriété](#)

[Files, propriété](#)

[IsRootFolder, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[Path, propriété](#)

[ShortPath, propriété](#)

[Size, propriété](#)

[SubFolders, propriété](#)

[Type, propriété](#)

ShortName, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)

[Folder, objet](#)

ShortPath, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[DateCreated, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Drive, propriété](#)

[Files, propriété](#)

[IsRootFolder, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[Path, propriété](#)

[ShortName, propriété](#)

[Size, propriété](#)

[SubFolders, propriété](#)

[Type, propriété](#)

ShortPath, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)

[Folder, objet](#)

Size, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[DateCreated, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Drive, propriété](#)

[Files, propriété](#)

[IsRootFolder, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[Path, propriété](#)

[ShortName, propriété](#)

[ShortPath, propriété](#)

[SubFolders, propriété](#)

[Type, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Size, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)
[Folder, objet](#)

Skip, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Close, méthode](#)

[Read, méthode](#)

[ReadAll, méthode](#)

[ReadLine, méthode](#)

[SkipLine, méthode](#)

[Write, méthode](#)

[WriteLine, méthode](#)

[WriteBlankLines, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Skip, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[TextStream, objet](#)

SkipLine, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Close, méthode](#)

[Read, méthode](#)

[ReadAll, méthode](#)

[ReadLine, méthode](#)

[Skip, méthode](#)

[Write, méthode](#)

[WriteLine, méthode](#)

[WriteBlankLines, méthode](#)

Microsoft® Visual Basic® Scripting Edition

SkipLine, méthode

Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[TextStream, objet](#)

SpecialFolder, constantes

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Comparaison, constantes](#)

[DriveType, constantes](#)

[FileAttribute, constantes](#)

[Entrée/Sortie de fichier, constantes](#)

[3-états, constantes](#)

SubFolders, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[DateCreated, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Drive, propriété](#)

[Files, propriété](#)

[IsRootFolder, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[Path, propriété](#)

[ShortName, propriété](#)

[ShortPath, propriété](#)

[Size, propriété](#)

[Type, propriété](#)

Microsoft® Visual Basic® Scripting Edition

SubFolders, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Folder, objet](#)

TextStream, objet

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Dictionary, objet](#)

[FileSystemObject, objet](#)

TextStream, objet

Propriétés

[AtEndOfLine, propriété](#)

[AtEndOfStream, propriété](#)

[Column, propriété](#)

[Line, propriété](#)

TextStream, objet

Méthodes

[Référence de la bibliothèque
d'exécution Scripting](#)

[Close, méthode](#)

[Read, méthode](#)

[ReadAll, méthode](#)

[ReadLine, méthode](#)

[Skip, méthode](#)

[SkipLine, méthode](#)

[Write, méthode](#)

[WriteLine, méthode](#)

[WriteBlankLines, méthode](#)

TotalSize, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[AvailableSpace, propriété](#)

[DriveLetter, propriété](#)

[DriveType, propriété](#)

[FileSystem, propriété](#)

[FreeSpace, propriété](#)

[IsReady, propriété](#)

[Path, propriété](#)

[RootFolder, propriété](#)

[SerialNumber, propriété](#)

[ShareName, propriété](#)

[VolumeName, propriété](#)

TotalSize, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drive, objet](#)

Type, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[Attributes, propriété](#)

[DateCreated, propriété](#)

[DateLastAccessed, propriété](#)

[DateLastModified, propriété](#)

[Drive, propriété](#)

[Files, propriété](#)

[IsRootFolder, propriété](#)

[Name, propriété](#)

[ParentFolder, propriété](#)

[Path, propriété](#)

[ShortName, propriété](#)

[ShortPath, propriété](#)

[Size, propriété](#)

[SubFolders, propriété](#)

Microsoft® Visual Basic® Scripting Edition

Type, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[File, objet](#)

[Folder, objet](#)

VolumeName, propriété

Voir aussi

[Référence de la bibliothèque
d'exécution Scripting](#)

[AvailableSpace, propriété](#)

[DriveLetter, propriété](#)

[DriveType, propriété](#)

[FileSystem, propriété](#)

[FreeSpace, propriété](#)

[IsReady, propriété](#)

[Path, propriété](#)

[RootFolder, propriété](#)

[SerialNumber, propriété](#)

[ShareName, propriété](#)

[TotalSize, propriété](#)

VolumeName, propriété Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[Drive, objet](#)

Write, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Close, méthode](#)

[Read, méthode](#)

[ReadAll, méthode](#)

[ReadLine, méthode](#)

[Skip, méthode](#)

[SkipLine, méthode](#)

[WriteLine, méthode](#)

[WriteBlankLines, méthode](#)

Microsoft® Visual Basic® Scripting Edition

Write, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[TextStream, objet](#)

WriteBlankLines, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Close, méthode](#)

[Read, méthode](#)

[ReadAll, méthode](#)

[ReadLine, méthode](#)

[Skip, méthode](#)

[SkipLine, méthode](#)

[Write, méthode](#)

[WriteLine, méthode](#)

Microsoft® Visual Basic® Scripting Edition

WriteBlankLines, méthode

Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[TextStream, objet](#)

WriteLine, méthode

[Référence de la bibliothèque
d'exécution Scripting](#)

Voir aussi

[Close, méthode](#)

[Read, méthode](#)

[ReadAll, méthode](#)

[ReadLine, méthode](#)

[Skip, méthode](#)

[SkipLine, méthode](#)

[Write, méthode](#)

[WriteBlankLines, méthode](#)

Microsoft® Visual Basic® Scripting Edition

WriteLine, méthode Application

[Référence de la bibliothèque
d'exécution Scripting](#)

[TextStream, objet](#)

Résumé des opérateurs

[Voir aussi](#)

[Opérateurs arithmétiques](#)

Opérateurs utilisés pour effectuer des calculs mathématiques.

[Opérateur d'affectation](#)

Opérateur utilisé pour affecter une valeur à une propriété ou à une variable.

[Opérateurs de comparaison](#)

Opérateurs utilisés pour effectuer des comparaisons.

[Opérateurs de concaténation](#)

Opérateurs utilisés pour combiner des chaînes.

[Opérateurs logiques](#)

Opérateurs utilisés pour effectuer des opérations logiques.

Fonctions mathématiques dérivées

[Référence du langage](#)
[Version 1](#)

[Voir aussi](#)

Description

La liste suivante récapitule les fonctions mathématiques non intrinsèques qui peuvent être dérivées des fonctions mathématiques intrinsèques :

Fonction	Équivalents dérivés
Sécante	$\text{Sec}(X) = 1 / \text{Cos}(X)$
Cosécante	$\text{Cosec}(X) = 1 / \text{Sin}(X)$
Cotangente	$\text{Cotan}(X) = 1 / \text{Tan}(X)$
Inverse sinus	$\text{Arcsin}(X) = \text{Atn}(X / \text{Sqr}(-X * X + 1))$
Inverse cosinus	$\text{Arccos}(X) = \text{Atn}(-X / \text{Sqr}(-X * X + 1)) + 2 * \text{Atn}(1)$
Inverse sécante	$\text{Arcsec}(X) = \text{Atn}(X / \text{Sqr}(X * X - 1)) + \text{Sgn}((X) - 1) * (2 * \text{Atn}(1))$
Inverse cosécante	$\text{Arccosec}(X) = \text{Atn}(X / \text{Sqr}(X * X - 1)) + (\text{Sgn}(X) - 1) * (2 * \text{Atn}(1))$
Inverse cotangente	$\text{Arccotan}(X) = \text{Atn}(X) + 2 * \text{Atn}(1)$
Sinus hyperbolique	$\text{HSin}(X) = (\text{Exp}(X) - \text{Exp}(-X)) / 2$
Cosinus hyperbolique	$\text{HCos}(X) = (\text{Exp}(X) + \text{Exp}(-X)) / 2$
Tangente hyperbolique	$\text{HTan}(X) = (\text{Exp}(X) - \text{Exp}(-X)) / (\text{Exp}(X) + \text{Exp}(-X))$
Sécante	$\text{HSec}(X) = 2 / (\text{Exp}(X) + \text{Exp}(-X))$

hyperbolique	
Cosécante hyperbolique	$H\text{Cosec}(X) = 2 / (\text{Exp}(X) - \text{Exp}(-X))$
Cotangente hyperbolique	$H\text{Cotan}(X) = (\text{Exp}(X) + \text{Exp}(-X)) / (\text{Exp}(X) - \text{Exp}(-X))$
Inverse sinus hyperbolique	$H\text{Arcsin}(X) = \text{Log}(X + \text{Sqr}(X * X + 1))$
Inverse cosinus hyperbolique	$H\text{Arccos}(X) = \text{Log}(X + \text{Sqr}(X * X - 1))$
Inverse tangente hyperbolique	$H\text{Arctan}(X) = \text{Log}((1 + X) / (1 - X)) / 2$
Inverse sécante hyperbolique	$H\text{Arcsec}(X) = \text{Log}((\text{Sqr}(-X * X + 1) + 1) / X)$
Inverse cosécante hyperbolique	$H\text{Arccosec}(X) = \text{Log}((\text{Sgn}(X) * \text{Sqr}(X * X + 1) + 1) / X)$
Inverse cotangente hyperbolique	$H\text{Arccotan}(X) = \text{Log}((X + 1) / (X - 1)) / 2$
Logarithme de base N	$\text{LogN}(X) = \text{Log}(X) / \text{Log}(N)$

Messages d'erreur VBScript

[Référence du langage](#)

Code d'erreur	Message
5	Appel ou argument de procédure incorrect
6	Dépassement de capacité
7	Mémoire insuffisante
9	Indice en dehors de la plage
10	Ce tableau est fixe ou temporairement verrouillé
11	Division par zéro
13	Type incompatible
14	Espace de chaîne insuffisant
28	Espace pile insuffisant
35	Sub ou Function non définie
48	Erreur de chargement de la DLL
51	Erreur interne
53	Fichier introuvable
57	Erreur d'entrée/sortie de périphérique
58	Ce fichier existe déjà
61	Disque plein
67	Trop de fichiers
70	Permission refusée
75	Erreur dans le chemin d'accès
76	Chemin d'accès introuvable
91	Variable objet ou variable de bloc With non définie

92	Boucle For non initialisée
94	Utilisation incorrecte de Null
322	Impossible de créer le fichier temporaire nécessaire
424	Objet requis
429	Un composant ActiveX ne peut pas créer un objet
430	Cette classe ne gère pas Automation
432	Nom du fichier ou de la classe introuvable lors de l'opération Automation
438	Propriété ou méthode non gérée par cet objet
440	Erreur Automation
445	Cet objet ne gère pas cette action
446	Cet objet ne gère pas les arguments nommés
447	Cet objet ne gère pas les paramètres régionaux en cours
448	Argument nommé introuvable
449	Argument non facultatif
450	Nombre d'arguments incorrect ou affectation de propriété incorrecte
451	Cet objet n'est pas une collection
453	Fonction de DLL spécifiée introuvable
455	Erreur de verrouillage de la ressource de code
457	Cette clé est déjà associée à un élément de cette collection
458	Cette variable utilise un type Automation non géré par Visual Basic
500	Variable non définie
501	Impossible d'affecter à la variable
502	Objet non sécurisé pour le script
503	Objet non sécurisé pour l'initialisation
1001	Mémoire insuffisante
1002	Erreur de syntaxe

1003	Attendu ':'
1004	Attendu ';'
1005	Attendu '('
1006	Attendu ')'
1007	Attendu ']'
1008	Attendu '{'
1009	Attendu '}'
1010	Identificateur attendu
1011	Attendu '='
1012	'If' attendu
1013	'To' attendu
1014	'End' attendu
1015	'Function' attendu
1016	'Sub' attendu
1017	'Then' attendu
1018	'Wend' attendu
1019	'Loop' attendu
1020	'Next' attendu
1021	'Case' attendu
1022	'Select' attendu
1023	Expression attendue
1024	Instruction attendue
1025	Fin d'instruction attendue
1026	Constante (entier) attendue
1027	'While' ou 'Until' attendu
1028	'While', 'Until' ou fin d'instruction attendus
1029	Trop de variables locales ou d'arguments
1030	Identificateur trop long
1031	Nombre incorrect

1032	Caractère incorrect
1033	Constante de chaîne non terminée
1034	Commentaire non terminé
1035	Commentaire imbriqué
1037	Utilisation incorrecte du mot clé 'Me'
1038	'Loop' sans 'Do'
1039	Instruction 'Exit' incorrecte
1040	Variable de contrôle de boucle 'For' incorrecte
1041	Nom redéfini
1042	Ce devrait être la première instruction de la ligne
1043	Impossible d'affecter un argument autre que ByVal
1044	Impossible d'utiliser les parents pendant l'appel d'une procédure Sub
1045	Constantes (littéraux) attendues
1046	'In' attendu
32766	True
32767	False
32811	Élément introuvable

Microsoft® Visual Basic® Scripting Edition

ExecuteGlobal, instruction

[Référence du langage](#)

Voir aussi

[Eval, fonction](#)

[Execute, instruction](#)

GetLocale, fonction

[Référence du langage](#)

Voir aussi

[SetLocale, fonction](#)

[Tableau des ID de langue](#)

Tableau des ID de langue (LCID)

[Référence du langage](#)

Voir aussi

[GetLocale, fonction](#)

[SetLocale, fonction](#)

SetLocale, fonction

[Référence du langage](#)

Voir aussi

[GetLocale, fonction](#)

[Tableau des ID de langue \(LCID\)](#)

Microsoft® Visual Basic® Scripting Edition

Résumé des opérateurs

[Référence du langage](#)

Voir aussi

[Priorité des opérateurs](#)

Opérateur d'affectation

[Référence du langage](#)
[Version 1](#)

[Opérateur =](#)

Fonctions mathématiques dérivées

[Référence du langage](#)

Voir aussi

[Atn, fonction](#)

[Cos, fonction](#)

[Exp, fonction](#)

[Log, fonction](#)

[Sin, fonction](#)

[Sqr, fonction](#)

[Tan, fonction](#)
