Documentation sur Visual Basic Script

Qu'est-ce que VBScript ?

Microsoft ® Visual Basic Scripting Edition permet d'utiliser des scripts actifs dans de nombreux environnements..

Facile à utiliser et à apprendre

Si vous connaissez déjà Visual Basic ou Visual Basic pour Applications (VBA), 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 lignes puissent vous apprendre VBScript, elles ne vous enseigneront pas la programmation. Pour apprendre la programmation, reportez-vous à *Step by Step*, publié par Microsoft Press.

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 traitent comme des chaînes. 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. 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 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 Cxxx 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

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. 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 une 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 dépend de 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. Vous pouvez déclarer plusieurs dimensions 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

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

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. Vous représentez les littéraux de date et les littéraux d'heure en les encadrant 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

Description	Symbole
Élévation à une puissance	^
Négation unaire	-
Multiplication	*
Division	/
Division entière	\
Modulo arithmétique	Mod
Addition	+
Soustraction	-
Concaténation de chaînes	&

Comparaison

Description	Symbole
Égalité	=
Différence	<>
Inférieur à	<
Supérieur à	>
Inférieur ou égal à	<=
Supérieur ou égal à	>=
Équivalence d'objet	Is

Logique

Description	Symbole
Négation logique	Not
Conjonction logique	And
Disjonction logique	Or
Exclusion logique	Xor
Équivalence logique	Eqv
Implication logique	Imp

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

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(value)
   If value = 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 value = 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, comme l'illustre l'exemple suivant.

```
Select Case Document.Form1.CardType.Options(SelectedIndex).Text
   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

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

Il existe deux façons d'utiliser le mot clé **Until** 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 augmente ou diminue 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, 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 For 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.

Procédures de VBScript

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 vide.

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.", 1)
   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.", 1)
   MsgBox "La température est de " & Celsius(temp) & " degrés C."
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 C."
```

Pour appeler une procédure **Sub** à partir d'une autre procédure, tapez 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

Ces conventions sont des suggestions conçues pour 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 clair, précis, lisible, intuitif 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 œuvre 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 améliorer la 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, Function ou Sub.	Visible dans la procédure où elle est déclarée.
Niveau script	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	sblnCalculEnCours

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 œuvre (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 œuvre 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 clairs pour que les commentaires sur une ligne servent uniquement à fournir des détails de mise en œuvre 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 :

- Mettez les blocs imbriqués standard en retrait de quatre espaces.
- Mettez les commentaires généraux d'une procédure en retrait d'un espace.
- Mettez les instructions du plus haut niveau venant après les commentaires généraux en retrait de quatre espaces, chaque bloc imbriqué étant lui-même mis en retrait de quatre espaces supplémentaires.

Le code suivant est conforme aux conventions de codage VBScript.

```
'*******************************************************
' 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 de
boucle
      End If
      i = i + 1   ' Incrément compteur de boucle
    Loop
End Function
```

MsgBox, constantes

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**.

Chaîne, constantes

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.

VarType, constantes

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

Liste des Mots clés de VBScript

Empty False Nothing Null
True

Empty

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

Voir aussi
Null

False

Le mot clé False à une valeur égale à 0.

Voir aussi
True

Nothing

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
Dim, instruction | Set, instruction

Null

Le mot clé Null est utilisé pour indiquer qu'une variable ne contient aucune donnée valide. Cette valeur est différente de la valeur Empty.

Voir aussi
Empty

True

Le mot clé True a une valeur égale à -1.

Voir aussi
False

Liste des Oprateurs de VBScript

Priorité des opérateurs

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
Négation (-) Égalité (=) Not
Élévation à une puissance (^) 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.

Liste des opérateurs

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.

+ (addition), opérateur

Effectue la somme de deux nombres.

result = expression1 + expression2
Arguments
result
Toute variable numérique.
expression1
Toute expression.

expression2
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.

Voir aussi
&, opérateur | -, opérateur | Opérateurs arithmétiques | Opérateurs de concaténation | Priorité des opérateurs | Liste des opérateurs

And, opérateur

Effectue la conjonction logique de deux expressions.

result = expression1 And expression2
Arguments
result
Toute variable numérique.
expression1
Toute expression.
expression2
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 expression1 est et expression2 est result 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 expression1 est et le bit dans expression2 est result est
0 0 0
0 1 0
1 0 0
1 1 1

Voir aussi

Opérateurs logiques | Not, opérateur | Priorité des opérateurs | Liste des opérateurs | Or, opérateur | Xor, opérateur

= (affectation), opérateur

Affecte une valeur à une variable ou à une propriété.

variable = value
Arguments
variable
Toute variable ou propriété qui peut être définie.
value
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.

Voir aussi
Opérateurs de comparaison | Priorité des opérateurs | Liste des opérateurs | Set, instruction

& (concaténation), opérateur

Force la concaténation de chaînes de deux expressions.

result = expression1 & expression2
Arguments
result
Toute variable.
expression1
Toute expression.
expression2
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.

Voir aussi
Opérateurs de concaténation | Priorité des opérateurs | Liste des opérateurs

/ (division), opérateur

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

result = number1/number2
Arguments
result
Toute variable numérique.
number1
Toute expression numérique.
number2
Toute expression numérique.
Notes
Si une ou les deux expressions sont des expressions Null, result est Null. Toute expression Empty est traitée comme 0.

Voir aussi
*, opérateur | \, opérateur | Opérateurs arithmétiques | Priorité des opérateurs | Liste des opérateurs

Eqv, opérateur

Effectue une équivalence logique de deux expressions.

result = expression1 Eqv expression2

Arguments
result
Toute variable numérique.
expression1
Toute expression.
expression2
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 expression1 est	et expression2 est	result 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 expression1 est	et si le bit dans expression2 est	result est
0	0	1
0	1	0
1	0	0
1	1	1

Voir aussi
Imp, opérateur | Opérateurs logiques | Priorité des opérateurs | Liste des opérateurs

^ (élévation à la puissance), opérateur

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

result = number^exponent
Arguments
result
Toute variable numérique.
number
Toute expression numérique.
exponent
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
Opérateurs arithmétiques | Priorité des opérateurs | Liste des opérateurs

Imp, opérateur

Effectue une implication logique entre deux expressions.

result = expression1 Imp expression2
Arguments
result
Toute variable numérique.
expression1
Toute expression.
expression2
Toute expression.

Notes
Le tableau suivant illustre la manière dont est déterminé l'élément result :

Si expression1 est	et expression2 est	Alors result 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 expression1 est	et le bit dans expression2 est	Alors result vaut
0	0	1
0	1	1
1	0	0
1	1	1

Voir aussi
Eqv, opérateur | Opérateurs logiques | Priorité des opérateurs | Liste des opérateurs

\ (division entière), opérateur

Effectue la division de deux nombres et renvoie le résultat sous forme de nombre entier.

result = number1\number2
Arguments
result
Toute variable numérique.
number1
Toute expression numérique.
number2
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.

Voir aussi
*, opérateur | /, opérateur | Opérateurs arithmétiques | Priorité des opérateurs | Liste des opérateurs

Is, opérateur

Compare deux variables de référence à un objet.

result = object1 Is object2
Arguments
result
Toute variable numérique.
object1
Tout nom d'objet.
object2
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

Voir aussi
Opérateurs de comparaison | Priorité des opérateurs | Liste des opérateurs

Mod, opérateur

Effectue la division de deux nombres et renvoie seulement le reste.

result = number1 Mod number2
Arguments
result
Toute variable numérique.
number1
Toute expression numérique.
number2
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.

Voir aussi
Opérateurs arithmétiques | Priorité des opérateurs | Liste des opérateurs

* (multiplication), opérateur

Multiplie deux nombres.

result = number1*number2
Arguments
result
Toute variable numérique.
number1
Toute expression numérique.
number2
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.

Voir aussi
\, opérateur | Opérateurs arithmétiques | Priorité des opérateurs | Liste des opérateurs

Not, opérateur

Effectue la négation logique d'une expression.

result = Not expression
Arguments
result
Toute variable numérique.
expression
Toute expression.
Notes

Le tableau suivant illustre la manière dont l'élément result est déterminé :

Si expression est alors result 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 expression Bit dans result
0 1
1 0

Voir aussi
And, opérateur | Opérateurs logiques | Priorité des opérateurs | Liste des opérateurs | Or, opérateur | Xor, opérateur

Or, opérateur

Effectue l'opération logique de disjonction sur deux expressions.

result = expression1 Or expression2
Arguments
result
Toute variable numérique.
expression1
Toute expression.
expression2
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 expression1 est et expression2 est alors result 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 expression1 est et le bit dans expression2 est alors result vaut
0 0 0
0 1 1
1 0 1
1 1 1

Voir aussi
And, opérateur | Opérateurs logiques | Not, opérateur | Priorité des opérateurs | Liste des opérateurs | Xor, opérateur

- (soustraction), opérateur

Effectue la différence entre deux nombres ou indique la valeur négative d'une expression numérique.

Syntaxe 1
result = number1-number2

Syntaxe 2
-number
Arguments
result
Toute variable numérique.
number
Toute expression numérique.
number1
Toute expression numérique.
number2
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.

Voir aussi
+, opérateur | Opérateurs arithmétiques | Priorité des opérateurs | Liste des opérateurs

Xor, opérateur

Effectue l'opération logique d'exclusion sur deux expressions.

result = expression1 Xor expression2
Arguments
result
Toute variable numérique.
expression1
Toute expression.
expression2
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 expression1 est et expression2 est alors result 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 expression1 est et le bit dans expression2 est alors result vaut
0 0 0
0 1 1
1 0 1
1 1 0

Voir aussi
And, opérateur | Opérateurs logiques | Not, opérateur | Priorité des opérateurs | Liste des opérateurs | Or, opérateur

Autres opérateurs

Opérateurs arithmétiques
^, opérateur

*, opérateur

/, opérateur

\, opérateur

Mod, opérateur

+, opérateur

-, opérateur

Opérateurs de comparaison
Utilisés pour comparer des expressions.

result = expression1 comparisonoperator expression2
result = object1 Is object2
Arguments
result
Toute variable numérique.
expression
Toute expression.
comparisonoperator
Tout opérateur de comparaison.
object
Tout nom d'objet.
Notes
L'opérateur Is comporte une fonctionnalité de comparaison spécifique qui diffère des autres opérateurs énumérés 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 expression1 < expression2 expression1 >= expression2 expression1 ou expression2 = Null
<= Inférieur ou égal à expression1 <= expression2 expression1 > expression2 expression1 ou expression2 = Null
> Supérieur à expression1 > expression2 expression1 <= expression2 expression1 ou expression2 = Null
>= Supérieur ou égal à expression1 >= expression2 expression1 < expression2 expression1 ou expression2 = Null
= Égal à expression1 = expression2 expression1 <> expression2 expression1 ou expression2 = Null
<> Différent de expression1 <> expression2 expression1 = expression2 expression1 ou expression2 = 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
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 expression Empty.
Les deux expressions sont Empty Les expressions sont égales.

Voir aussi
=, opérateur | Is, opérateur | Priorité des opérateurs | Liste des opérateurs

Opérateurs de concaténation
&, opérateur

+, opérateur

Opérateurs logiques
And, opérateur

Not, opérateur

Or, opérateur

Xor, opérateur

Liste des Instructions de VBScript

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

Call, instruction

Transfère le contrôle à une procédure Sub ou Function.

[Call] name [argumentlist]
Arguments
Call
Mot clé facultatif ; si spécifié, vous devez mettre argumentlist 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
```

Class, instruction

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

```
Class name
   statements
End Class
```

Arguments
name
Requis. Nom de la classe ; respecte les conventions standard d'affectation de noms de variable.
statements
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.

Voir aussi

Dim, instruction | Function, instruction | Private, instruction | Property Get, instruction | Property Let, instruction | Property Set, instruction | Public, instruction | Set, instruction | Sub, instruction

Const, instruction

Déclare des constantes destinées à remplacer des valeurs littérales.

[Public | Private] Const constname = expression
Arguments
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.
constname
Nom de la constante ; respecte les conventions standard d'attribution de nom de variable.
expression
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 constantes sont publiques par défaut.
Private Const MyString = "AIDE"   ' Déclare les constantes privées.
Const MyStr = "Bonjour", MyNumber = 3.4567   ' Déclare plusieurs constantes sur la même ligne.
```

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.

Voir aussi
Dim, instruction | Function, instruction | Private, instruction | Public, instruction | Sub, instruction

Dim, instruction

Déclare des variables et alloue l'espace de stockage.

Dim varname[([subscripts])][, varname[([subscripts])]] . . .
Arguments
varname
Nom de la variable ; respecte les conventions standard d'affectation de nom à des variables.
subscripts
Dimensions d'une variable tableau ; jusqu'à 60 dimensions multiples peuvent être déclarées. L'argument subscripts utilise la syntaxe suivante :
upperbound [,upperbound] . . .

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.

Remarque 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.
```

Voir aussi
Private, instruction | Public, instruction | ReDim, instruction | Set, instruction

Do...Loop, instruction

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

```
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]
```

Arguments
condition
Expression numérique ou expression de chaîne qui est True ou False. Si condition est Null, l'élément condition est traité comme False.
statements
Une ou plusieurs instructions qui sont répétées tant que l'élément condition 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 et 6.
  DefResp = MsgBox (MyNum & " Voulez-vous un autre nombre?", vbYesNo)
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 externe.
```

Voir aussi
Exit, instruction | For...Next, instruction | While...Wend, instruction

Erase, instruction

Réinitialise les éléments des tableaux de taille fixe et libère l'espace de stockage des tableaux dynamiques.

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 stockage.
Erase NumArray   ' Chaque élément est réinitialisé.
Erase DynamicArray   ' Libérer la mémoire utilisée par le tableau
```

Voir aussi
Dim, instruction | Nothing | ReDim, instruction

Execute, instruction

Exécute une ou plusieurs instructions spécifiées.

Execute 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 Execute utilise toujours la première interprétation, alors que l'instruction Eval utilise toujours la deuxième.

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 Execute 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 comme 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
```

Voir aussi
Eval, fonction | ExecuteGlobal, instruction

ExecuteGlobal, instruction

Exécute une ou plusieurs instructions spécifiées dans l'espace de nom global d'un script.

ExecuteGlobal statement
L'argument statement est une expression de chaîne contenant une ou plusieurs instructions à exécuter. Insérez plusieurs instructions dans l'argument statement, en les séparant par deux points (:) ou des sauts de lignes 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 prend la valeur True ; dans le cas contraire, result prend la valeur False. L'instruction ExecuteGlobal utilise toujours la première interprétation, tandis que la méthode Eval adopte la deuxième.

L'ajout de procédures et de classes au moment de l'exécution peut s'avère très utile, mais comporte 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 suivant 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 pour qu'il ne soit pas nécessaire de placer la procédure entière entre guillemets :

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

Exit, instruction

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

```
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 nombres aléatoires.
      Select Case MyNum   ' Évaluer le nombre aléatoire.
        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
```

Voir aussi
Do...Loop, instruction | For Each...Next, instruction | For...Next, instruction | Function, instruction | Sub, instruction

For Each...Next, instruction

Répète un groupe d'instructions pour chaque élément d'un tableau ou d'une collection.

```
For Each element In group
  [statements]
  [Exit For]
  [statements]
Next [element]
```

Arguments
element
Variable employée pour effectuer une itération sur les éléments de la collection ou du tableau. Pour les collections, l'argument element 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 element peut être uniquement une variable de type Variant.
group
Nom d'une collection d'objets ou d'un tableau.
statements
Une ou plusieurs instructions pouvant être exécutées sur chaque élément d'un groupe indiqué par group.
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
```

Voir aussi
Do...Loop, instruction | Exit, instruction | For...Next, instruction | While...Wend, instruction

For...Next, instruction

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

For counter = start To end [Step step]
[statements]
[Exit For]
[statements]
Next
Arguments
counter
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.
start
Valeur initiale de counter.
end
Valeur finale de counter.
step
Quantité par laquelle counter change à chaque accomplissement de la boucle. Si cet élément n'est pas spécifié, la valeur par défaut de increment est 1.
statements
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 <= start
Négatif counter >= 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.

Remarque 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
```

Voir aussi
Do...Loop, instruction | Exit, instruction | For Each...Next, instruction | While...Wend, instruction

Function, instruction

Déclare le nom, les arguments et le code qui forment le corps d'une procédure Function.

[Public [Default] | Private] Function name [(arglist)]
[statements]

[name = expression]
[Exit Function]
[statements]
[name = expression]
End Function
Arguments
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.
name
Nom de la procédure Function ; respecte les conventions standard d'affectation de nom à des variables.
arglist
Liste de variables représentant les arguments qui sont transmis à la procédure Function lorsqu'elle est appelée. Des virgules séparent plusieurs variables.
statements
Tout groupe d'instructions à exécuter dans le corps de la procédure Function.
expression
Valeur renvoyée de la procédure Function.
L'argument arglist comporte la syntaxe et les éléments suivants :

[ByVal | ByRef] varname[( )]

Arguments
ByVal
Indique que l'argument est transmis par valeur.
ByRef
Indique que l'argument est transmis par référence.
varname
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 de 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.

Voir aussi
Call, instruction | Dim, instruction | Exit, instruction | Nothing | Set, instruction | Sub, instruction

If...Then...Else, instruction

Exécute un groupe d'instructions soumises à une condition, en fonction de la valeur d'une expression.

If condition Then statements [Else elsestatements]
Ou, vous pouvez utiliser la syntaxe suivante, plus polyvalente :

```
If condition Then
  [statements]
[ElseIf condition-n Then
  [elseifstatements]] . . .
[Else
  [elsestatements]]
End If
```

Arguments
condition
Un ou plusieurs types d'expressions suivants :
Une expression numérique ou une expression de chaîne qui produit la valeur True ou False. Si condition est Null, l'élément condition est traité comme False.

Une expression de la forme TypeOf objectname Is objecttype. L'élément objectname est toute référence d'objet et l'élément objecttype est tout type d'objet valide. L'expression est True si objectname est le type d'objet spécifié par typeobjet ; dans le cas contraire, sa valeur est False.

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.
elsestatements
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.

On Error, instruction

Active ou désactive la gestion des erreurs.

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, l'hôte exécutant le code détermine le comportement exact. 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.
```

Voir aussi
Err, objet | Exit, instruction

Option Explicit, instruction

Force la déclaration explicite de toutes les variables dans un script.

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 des variables.
Dim MyVar   ' Déclarer une variable.
MyInt = 10   ' La variable non déclarée génère une erreur.
MyVar = 10   ' La variable déclarée ne génère pas d'erreur.
```

Private, instruction

Déclare les variables privées et alloue l'espace de stockage. Déclare, dans un bloc Class, une variable privée.

Private varname[([subscripts])][, varname[([subscripts])]] . . .
Arguments
varname
Nom de la variable ; respecte les conventions standard d'affectation de noms de variable.
subscripts
Dimensions d'une variable de tableau ; jusqu'à 60 dimensions peuvent être déclarées. L'argument subscripts utilise la syntaxe suivante :
upper [, upper] . . .

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.

Remarque 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
```

Voir aussi
Dim, instruction | Public, instruction | ReDim, instruction | Set, instruction

Property Get, instruction

Déclare, dans un bloc Class, le nom, les arguments et le code formant une procédure Property qui obtient (renvoie) la valeur d'une propriété.

```
[Public [Default] | Private] Property Get name [(arglist)]
  [statements]
  [[Set] name = expression]
  [Exit Property]
  [statements]
  [[Set] name = expression]
End Property
```

Arguments
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.
name
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.
arglist
Liste des variables représentant les arguments qui sont transférés à la procédure Property Get lorsqu'elle est appelée. Des virgules séparent plusieurs arguments. 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).
statements
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.
expression
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.

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

Déclare, dans un bloc Class, le nom, les arguments et le code formant une procédure Property qui affecte (définit) la valeur d'une propriété.

[Public | Private] Property Let name ([arglist,] value)
  [statements]
  [Exit Property]
  [statements]
End Property
Arguments
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.
name
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.
arglist
Liste des variables représentant les arguments qui sont transférés à la procédure Property Let lorsqu'elle est appelée. Des virgules séparent les arguments. 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.
value
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.
statements
Tout groupe d'instructions à exécuter à l'intérieur d'une procédure Property Let.
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.

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.
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.

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

Déclare, dans un bloc Class, le nom, les arguments et le code formant une procédure Property qui définit la référence à un objet.

[Public | Private] Property Set name([arglist,] reference)

[statements]
[Exit Property]
[statements]
End Property
Arguments
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.
name
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.
arglist
Liste des variables représentant les arguments qui sont transférés à la procédure Property Set lorsqu'elle est appelée. Des virgules séparent les arguments. 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.
reference
Variable contenant la référence à l'objet apparaissant à la droite de l'affectation de référence.
statements
Tout groupe d'instructions à exécuter à l'intérieur d'une procédure Property Set.
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.

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. 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'affection de référence à l'objet (instruction Set).

Voir aussi
Class, instruction | Dim, instruction | Exit, instruction | Function, instruction | Private, instruction | Property Get, instruction | Property Let, instruction | Public, instruction | Set, instruction | Sub, instruction

Public, instruction

Déclare des variables publiques et affecte l'espace de stockage. Déclare, dans un bloc Class, une variable privée.

Public varname[([subscripts])][, varname[([subscripts])]] . . .
Arguments
varname
Nom de la variable ; respecte les conventions standard d'affectation de noms de variable.
subscripts
Dimensions d'une variable de tableau ; jusqu'à 60 dimensions peuvent être déclarées. Syntaxe de l'argument subscripts :
upper [, upper] . . .

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
```

Voir aussi
Dim, instruction | Private, instruction | ReDim, instruction | Set, instruction

Randomize, instruction

Initialise le générateur de nombres aléatoires.

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 nombres aléatoires.
Do Until Response = vbNo
  MyValue = Int((6 * Rnd) + 1)   ' Génère une valeur aléatoire entre 1 et 6.
  MsgBox MyValue
  Response = MsgBox ("Recommencer? ", vbYesNo)
Loop
```

Voir aussi
Rnd, fonction | Timer, fonction

ReDim, instruction

Déclare les variables de tableau dynamique et attribue ou réattribue l'espace de stockage au niveau de la procédure.

ReDim [Preserve] varname(subscripts) [, varname(subscripts)] . . .
Arguments
Preserve
Conserve les données d'un tableau existant quand vous changez la taille de la dernière dimension.
varname
Nom de la variable ; respecte les conventions standard d'affectation de nom à des variables.
subscripts

Dimensions d'une variable d'un tableau ; jusqu'à 60 dimensions multiples peuvent être déclarées. L'argument subscripts utilise la syntaxe suivante :
upper [,upper] . . .

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.

Voir aussi
Dim, instruction | Set, instruction

Rem, instruction

Inclut des remarques explicatives dans un programme.

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, séparer par deux-points.
MyStr2 = "Au revoir" ' Ceci est aussi un commentaire ; deux-points ne sont pas nécessaires.
Rem Commentaire sur une ligne sans code ; deux-points ne sont pas nécessaires.
```

Select Case, instruction

Exécute un groupe d'instructions parmi plusieurs, en fonction de la valeur d'une expression.

```
Select Case testexpression
  [Case expressionlist-n
    [statements-n]] . . .
```

```
  [Case Else expressionlist-n
    [elsestatements-n]]
End Select
```

Arguments

testexpression

Toute expression numérique ou de chaîne.

expressionlist-n

Requis si Case apparaît. Liste délimitée d'une ou de plusieurs expressions.

statements-n

Une ou plusieurs instructions exécutées si testexpression correspond à un élément de expressionlist-n.

elsestatements-n

Une ou plusieurs instructions exécutées si testexpression 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 elsestatements 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 couleur"
  End Select
End Sub
```

Voir aussi

If ... Then ... Else, instruction

Set, instruction

Affecte une référence d'objet à une variable ou à une propriété, ou associe une référence de procédure à un événement.

```
Set objectvar = {objectexpression | New classname | Nothing}
```

-ou-

```
Set object.eventname = GetRef(procname)
```

Arguments

objectvar

Nom de la variable ou de la propriété ; respecte les conventions standard d'affectation de noms à des variables.

objectexpression

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 objectvar 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.

classname
Facultatif. Nom de la classe en cours de création. Une classe et ses membres sont définies par l'instruction Class.
Nothing
Facultatif. Met fin à l'association de l'élément objectvar à un objet ou une classe spécifique. L'affectation à objectvar de la valeur Nothing libère toutes les ressources système et mémoire associées à l'objet précédemment référencé quand aucune autre variable n'y fait référence.
object
Nom de l'objet avec lequel event est associé.
event
Nom de l'événement auquel une fonction va être liée.
procname
Chaîne contenant le nom de la procédure Sub ou Function en cours d'association avec event.
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, lorsque 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 d'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/1024, 0)
  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.

Voir aussi
=, opérateur | Dim, instruction | GetRef, fonction | ReDim, instruction

Sub, instruction

Déclare le nom, les arguments et le code qui forment le corps d'une procédure Sub.

```
[Public [Default] | Private] Sub name [(arglist)]
  [statements]
  [Exit Sub]
  [statements]
End Sub
```

Arguments
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.
name
Nom de la procédure Sub ; respecte les conventions standard d'affectation de nom à des variables.

arglist
Liste de variables représentant les arguments passés à la procédure Sub quand elle est appelée. Des virgules séparent les variables multiples.
statements
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[( )]

Arguments
ByVal
Indique que l'argument est transmis par valeur.
ByRef
Indique que l'argument est transmis par référence.
varname
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 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.

Voir aussi
Call, instruction | Dim, instruction | Exit, instruction | Function, instruction

While...Wend, instruction

Exécute une série d'instructions tant qu'une condition donnée est True.

While condition
  Version [statements]
Wend
Arguments
condition
Expression numérique ou expression de chaîne qui produit la valeur True ou False. Si condition est Null, l'élément condition est traité comme étant False.

statements
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.

Remarque 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 compteur.
  Counter = Counter + 1   ' Incrémente le compteur.
  Alert Counter
Wend   ' Fin de la boucle While lorsque Counter > 19.
```

Voir aussi
Do...Loop, instruction

With, instruction

Exécute une série d'instructions sur un objet unique.

```
With object
  statements
End With
```

Arguments
object
Nom d'un objet ou d'une fonction qui renvoie un objet.
statements
Requis. Une ou plusieurs instructions à appliquer sur un objet.
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'affection 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.

Voir aussi

Set, instruction

Liste des Fonctions de VBScript

Abs	Array	Asc	Atn
Cbool	CByte	CCur	CDate
CDbl	Chr	CInt	CLng
Conversions	Cos	CreateObject	CSng
Date	DateAdd	DateDiff	DatePart
DateSerial	DateValue	Day	Derived Maths
Eval	Exp	Filter	FormatCurrency
FormatDateTime	FormatNumber	FormatPercent	GetLocale
GetObject	GetRef	Hex	Hour
InputBox	InStr	InStrRev	Int, Fixs
IsArray	IsDate	IsEmpty	IsNull
IsNumeric	IsObject	Join	LBound
Lcase	Left	Len	Log
LTrim; Rtrim; and Trims	Maths	Mid	Minute
Month	MonthName	MsgBox	Now
Oct	Replace	RGB	Right
Rnd	Round	ScriptEngine	Second
SetLocale	Sgn	Sin	Space
Split	Sqr	StrComp	String
Tan	Time	Timer	TimeSerial
TimeValue	TypeName	UBound	UCase
VarType	Weekday	WeekdayName	Year

Abs, fonction

Renvoie la valeur absolue d'un nombre.

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.
```

Voir aussi
Sgn, fonction

Array, fonction

Renvoie une variable de type Variant contenant un tableau.

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.

Voir aussi
Dim, instruction

Asc, fonction

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

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 plates-formes 32 bits qui utilisent des caractères Unicode. Elle renvoie le code de caractère Unicode (large), ce qui évite une conversion de Unicode à ANSI.

Voir aussi
Chr, fonction

Atn, fonction

Renvoie l'arc tangente d'un nombre.

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).

Voir aussi
Cos, fonction | Fonctions mathématiques dérivées | Sin, fonction | Tan, fonction

CBool, fonction

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

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.
```

Voir aussi
CByte, fonction | CCur, fonction | CDate, fonction | CDbl, fonction | CInt, fonction | CLng, fonction | CSng, fonction | CStr, fonction

CByte, fonction

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

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 Double.
MyByte = CByte(MyDouble)   ' MyByte contient 126.
```

Voir aussi
CBool, fonction | CCur, fonction | CDate, fonction | CDbl, fonction | CInt, fonction | CLng, fonction | CSng, fonction | CStr, fonction

CCur, fonction

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

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).
```

Voir aussi
CBool, fonction | CByte, fonction | CDate, fonction | CDbl, fonction | CInt, fonction | CLng, fonction | CSng, fonction | CStr, fonction

CDate, fonction

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

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 recommandée. 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 Date.
MyTime = "16:35:47"          ' Définit l'heure.
MyShortTime = CDate(MyTime)   ' Convertit en type de données Date.
```

Voir aussi
IsDate, fonction

CDbl, fonction

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

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 est de type Currency (234,4567).
```

MyDouble = CDbl(MyCurr * 8.2 * 0.01) ' Convertit le résultat en Double (19,2254576).

Voir aussi
CBool, fonction | Cbyte, fonction | CCur, fonction | CDate, fonction | CInt, fonction | CLng, fonction | CSng, fonction | CStr, fonction

Chr, fonction

Renvoie le caractère associé au code de caractère ANSI spécifié.

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.

Voir aussi
Asc, fonction

CInt, fonction

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

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.

Voir aussi
CBool, fonction | Cbyte, fonction | CCur, fonction | CDate, fonction | CDbl, fonction | CLng, fonction | CSng, fonction | CStr, fonction | Int, Fix, fonctions

CLng, fonction

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

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 sont des Doubles.
MyLong1 = CLng(MyVal1)   ' MyLong1 contient 25427.
MyLong2 = CLng(MyVal2)   ' MyLong2 contient 25428.
```

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.

Voir aussi
CBool, fonction | Cbyte, fonction | CCur, fonction | CDate, fonction | CDbl, fonction | CInt, fonction | CSng, fonction | CStr, fonction | Int, Fix, fonctions

Cos, fonction

Renvoie le cosinus d'un angle.

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 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 radians.
MySecant = 1 / Cos(MyAngle)   ' Calculer la sécante.
```

Voir aussi
Atn, fonction | Fonctions mathématiques dérivées | Sin, fonction | Tan, fonction

CreateObject, fonction

Crée et renvoie une référence à un objet Automation.

CreateObject(servername.typename [, location])

Arguments
servername
Requis. Le nom de l'application fournissant l'objet.
typename
Requis. Le type ou la classe de l'objet à créer.
location
Facultatif. Le nom du serveur réseau sur lequel l'objet doit être créé.
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 Application.
ExcelSheet.Application.Visible = True
' Place du texte dans la première cellule de la feuille.
ExcelSheet.ActiveSheet.Cells(1,1).Value = "Colonne A, ligne 1"
' Enregistre la feuille.
ExcelSheet.SaveAs "C:.XLS"
' Ferme Excel avec la méthode Quit sur l'objet Application.
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", "MyServer")
  GetVersion = XLApp.Version
End Function
```

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

Voir aussi
GetObject, fonction

CSng, fonction

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

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   ' MyDouble1, MyDouble2 sont des Doubles.
MyDouble1 = 75,3421115: MyDouble2 = 75,3421555
MySingle1 = CSng(MyDouble1)   ' MySingle1 contient 75,34211.
MySingle2 = CSng(MyDouble2)   ' MySingle2 contient 75,34216.
```

Voir aussi
CBool, fonction | Cbyte, fonction | CCur, fonction | CDate, fonction | CDbl, fonction | CInt, fonction | CLng, fonction | CStr, fonction

CStr, fonction

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

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 expression 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".
```

Voir aussi
CBool, fonction | Cbyte, fonction | CCur, fonction | CDate, fonction | CDbl, fonction | CInt, fonction | CLng, fonction | CSng, fonction

Date, fonction

Renvoie la date système courante.

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

```
Dim MyDate
```

MyDate = Date   ' MyDate contient la date système actuelle.

Voir aussi
CDate, fonction | Now, fonction | Time, fonction

DateAdd, fonction

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

DateAdd(interval, number, date)
Arguments
interval
Requis. Expression de chaîne correspondant à l'intervalle à ajouter. Les valeurs sont indiquées dans la section Paramètres.
number
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.
date
Requis. Variant ou littéral représentant la date à laquelle l'argument interval est ajouté.
Paramètres
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

Notes
La fonction DateAdd permet d'additionner à une date un intervalle spécifique, ou de le soustraire. Par exemple, DateAdd permet de calculer une date à 30 jours d'aujourd'hui ou une heure à 45 minutes de maintenant. Pour ajouter des jours à 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 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 évalué.

Voir aussi
DateDiff, fonction | DatePart, fonction

DateDiff, fonction

Renvoie le nombre d'intervalles entre deux dates.

DateDiff(interval, date1, date2 [,firstdayofweek[, firstweekofyear]])
La syntaxe de la fonction DateDiff comprend les éléments suivants :

Arguments

interval
Expression de chaîne correspondant à l'intervalle à utiliser pour calculer la différence entre date1 et date2. Reportez-vous à la section Paramètres.
date1, date2
Expressions de date. Deux dates à utiliser dans le calcul.
firstdayofweek
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 Paramètres.
firstweekofyear
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 Paramètres.
Paramètres
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
VbUseSystemDayOfWeek 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
```

Voir aussi
DateAdd, fonction | DatePart, fonction

DatePart, fonction

Renvoie la partie spécifiée d'une date donnée.

DatePart(interval, date[, firstdayofweek[, firstweekofyear]])
Arguments
interval
Expression de chaîne représentant l'intervalle à renvoyer. Reportez-vous à la section Paramètres.
date
Expression de date à évaluer.
firstdayofweek
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 Paramètres.
firstweekofyear
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 Paramètres.
Paramètres
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
VbUseSystemDayOfWeek 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
```

Voir aussi
DateAdd, fonction | DateDiff, fonction

DateSerial, fonction

Renvoie un Variant de sous-type Date pour une année, un mois et un jour spécifiés.

DateSerial(year, month, day)
Arguments
year
Nombre compris entre 100 et 9999 inclus, ou une expression numérique.
month
Toute expression numérique.
day
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)   ' Renvoie 1 janvier 1970.
MyDate2 = DateSerial(1990 - 10, 8 - 2, 1 - 1)   ' Renvoie 31 mai 1980.
```

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'incrémentation 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.

Voir aussi
Date, fonction | DateValue, fonction | Day, fonction | Month, fonction | Now, fonction | TimeSerial, fonction | TimeValue, fonction | Weekday, fonction | Year, fonction

DateValue, fonction

Renvoie un Variant de sous-type Date.

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")   ' Renvoie une date.
```

Voir aussi
CDate, fonction | DateSerial, fonction | Day, fonction | Month, fonction | Now, fonction | TimeSerial, fonction | TimeValue, fonction | Weekday, fonction | Year, fonction

Day, fonction

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

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 contient 19.
```

Voir aussi
Date, fonction | Hour, fonction | Minute, fonction | Month, fonction | Now, fonction | Second, fonction | Weekday, fonction | Year, fonction

Fonctions mathématiques dérivées

La liste suivante récapitule les fonctions mathématiques non intrinsèques qui peuvent être dérivées des fonctions mathématiques intrinsèques :

Fonction	Équivalents dérivés
Sécante	Sec(X) = 1 / Cos(X)
Cosécante	Cosec(X) = 1 / Sin(X)

Cotangente Cotan(X) = 1 / Tan(X)
Inverse sinus Arcsin(X) = Atn(X / Sqr(-X * X + 1))
Inverse cosinus Arccos(X) = Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
Inverse sécante Arcsec(X) = Atn(X / Sqr(X * X - 1)) + Sgn((X) -1) * (2 * Atn(1))
Inverse cosécante Arccosec(X) = Atn(X / Sqr(X * X - 1)) + (Sgn(X) - 1) * (2 * Atn(1))
Inverse cotangente Arccotan(X) = Atn(X) + 2 * Atn(1)
Sinus hyperbolique HSin(X) = (Exp(X) - Exp(-X)) / 2
Cosinus hyperbolique HCos(X) = (Exp(X) + Exp(-X)) / 2
Tangente hyperbolique HTan(X) = (Exp(X) - Exp(-X)) / (Exp(X) + Exp(-X))
Sécante hyperbolique HSec(X) = 2 / (Exp(X) + Exp(-X))
Cosécante hyperbolique HCosec(X) = 2 / (Exp(X) - Exp(-X))
Cotangente hyperbolique HCotan(X) = (Exp(X) + Exp(-X)) / (Exp(X) - Exp(-X))
Inverse sinus hyperbolique HArcsin(X) = Log(X + Sqr(X * X + 1))
Inverse cosinus hyperbolique HArccos(X) = Log(X + Sqr(X * X - 1))
Inverse tangente hyperbolique HArctan(X) = Log((1 + X) / (1 - X)) / 2
Inverse sécante hyperbolique HArcsec(X) = Log((Sqr(-X * X + 1) + 1) / X)
Inverse cosécante hyperbolique HArccosec(X) = Log((Sgn(X) * Sqr(X * X + 1) +1) / X)
Inverse cotangente hyperbolique HArccotan(X) = Log((X + 1) / (X - 1)) / 2
Logarithme de base N LogN(X) = Log(X) / Log(N)

Voir aussi
Atn, fonction | Cos, fonction | Exp, fonction | Log, fonction | Sin, fonction | Sqr, fonction | Tan, fonction

Eval

Évalue une expression et renvoie le résultat.

[result =]Eval(expression)
Arguments
result
Facultatif. Variable à laquelle est affectée une valeur de retour. Si l'élément result n'est pas spécifié, pensez à utiliser l'instruction Execute à la place.
expression
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 comparaisons (==).
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:",,0))
  Do
    If Eval("Guess = RndNum") Then
      MsgBox "Félicitations! Vous avez deviné!"
      Exit Sub
    Else
      Guess = CInt(InputBox("Désolé! Recommencez.",,0))
    End If
  Loop Until Guess = 0
End Sub
```

Voir aussi
Execute, instruction

Exp, fonction

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

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
```

Voir aussi
Fonctions mathématiques dérivées | Log, fonction

Filter, fonction

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

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

Voir aussi
Replace, fonction

FormatCurrency, fonction

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

FormatCurrency(Expression[,NumDigitsAfterDecimal [,IncludeLeadingDigit [,UseParensForNegativeNumbers [,GroupDigits]]]])
Arguments
Expression
Requis. Expression à formater.
NumDigitsAfterDecimal
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.
IncludeLeadingDigit
Facultatif. Constante 3-états indiquant si un zéro non significatif s'affiche pour les valeurs décimales. Reportez-vous à la section Paramètres.
UseParensForNegativeNumbers
Facultatif. Constante 3-états indiquant s'il faut mettre les valeurs négatives entre parenthèses. Reportez-vous à la section Paramètres.
GroupDigits
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 Paramètres.
Paramètres
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
Lorsqu'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) ' MyCurrency contient 1000,00 F

Voir aussi
FormatDateTime, fonction | FormatNumber, fonction | FormatPercent, fonction

FormatDateTime, fonction

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

FormatDateTime(Date[,NamedFormat])
Arguments
Date
Expression de date à formater.
NamedFormat

Facultatif. Valeur numérique indiquant le format de date/heure utilisé. Si cette valeur est omise, vbGeneralDate est employé.
Paramètres
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.
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 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
```

Voir aussi
FormatCurrency, fonction | FormatNumber, fonction | FormatPercent, fonction

FormatNumber, fonction

Renvoie une expression formatée sous forme de nombre.

FormatNumber(Expression [,NumDigitsAfterDecimal [,IncludeLeadingDigit [,UseParensForNegativeNumbers [,GroupDigits]]]])
Arguments
Expression
Expression à formater.
NumDigitsAfterDecimal
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.
IncludeLeadingDigit
Facultatif. Constante 3-états indiquant si un zéro non significatif est affiché pour les valeurs décimales. Reportez-vous à la section Paramètres.
UseParensForNegativeNumbers
Facultatif. Constante 3-états indiquant s'il faut placer ou non les valeurs négatives entre parenthèses. Reportez-vous à la section Paramètres.
GroupDigits
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 Paramètres.
Paramètres
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 radians.
  MySecant = 1 / Cos(MyAngle)   ' Calculer la sécante.
  FormatNumberDemo = FormatNumber(MySecant,4) ' Formater Mycant avec quatre décimales
End Function
```

Voir aussi
FormatCurrency, fonction | FormatDateTime, fonction | FormatPercent, fonction

FormatPercent, fonction

Renvoie une expression formatée sous forme de pourcentage (multiplié par 100) avec un caractère de fin %.

FormatPercent(Expression[,NumDigitsAfterDecimal [,IncludeLeadingDigit [,UseParensForNegativeNumbers [,GroupDigits]]]])
La syntaxe de la fonction FormatPercent comprend les éléments suivants :

Arguments
Expression
Expression à formater.
NumDigitsAfterDecimal
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.
IncludeLeadingDigit
Facultatif. Constante 3-états indiquant s'il faut ou non afficher un zéro non significatif pour les valeurs décimales. Reportez-vous à la section Paramètres.
UseParensForNegativeNumbers
Facultatif. Constante 3-états indiquant s'il faut mettre les valeurs négatives entre parenthèses. Reportez-vous à la section Paramètres.
GroupDigits
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 Paramètres.
Paramètres
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 contient 6,25%.
```

Voir aussi
FormatCurrency, fonction | FormatDateTime, fonction | FormatNumber, fonction

GetLocale, fonction

Renvoie la valeur d'ID des paramètres régionaux en cours.

GetLocale()

Notes
Les paramètres régionaux sont un ensemble d'informations indiquant les préférences d'un utilisateur quant à sa langue, son pays, sa région 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 :

GetObject, fonction

Renvoie une référence à l'objet Automation d'un fichier.

GetObject([pathname] [, class])
Arguments
pathname
Facultatif ; chaîne. Chemin complet et nom du fichier contenant l'objet à extraire. Si l'argument pathname est omis, l'argument class est requis.
class
Facultatif ; chaîne. Classe de l'objet.
L'argument class utilise la syntaxe appname.objectype et comprend les éléments suivants :

Arguments
appname
Chaîne. Nom de l'application fournissant l'objet.
objectype
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\SCHEMA.CAD.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\SCHEMA.CAD.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:\DRAWINGS\SAMPLE.DRW", "FIGMENT.DRAWING")
```

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:\DRAWINGS\SAMPLE.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.

Voir aussi
CreateObject, fonction

GetRef, fonction

Renvoie une référence à une procédure éventuellement liée à un événement.

Set object.eventname = GetRef(procname)
Arguments
object
Requis. Nom de l'objet avec lequel l'événement est associé.
event
Requis. Nom de l'événement duquel dépend la fonction.
procname
Requis. Chaîne contenant le nom de la procédure Sub ou Function qui va être associée à l'événement.
Notes

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.

Voir aussi
Function, instruction | Set, instruction | Sub, instruction

Hex, fonction

Renvoie une chaîne représentant la valeur hexadécimale d'un nombre.

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 number 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.
```

Voir aussi
Oct, fonction

Hour, fonction

Renvoie un nombre entier compris entre 0 et 23 inclus, représentant l'heure du jour.

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 nombre représentant
                      ' le chiffre de l'heure actuelle.
```

Voir aussi
Day, fonction | Minute, fonction | Now, fonction | Second, fonction | Time, fonction

InputBox, fonction

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.

InputBox(prompt[,title][,default][,xpos][,ypos][,helpfile,context])
Arguments
prompt
Expression de chaîne qui est affichée sous la forme d'un message dans la boîte de dialogue. La longueur maximum de l'argument prompt est environ 1024 caractères, selon la largeur des caractères utilisés. Si l'argument prompt 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)).
title
Expression de chaîne qui est affichée dans la barre de titre de la boîte de dialogue. Si vous omettez l'argument title, le nom de l'application s'affiche dans la barre de titre.
default
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 default, la zone de texte s'affiche vide.
xpos
Expression numérique qui spécifie, en twips, la distance horizontale entre le bord gauche de la boîte de dialogue et le bord gauche de l'écran. Si l'argument xpos est omis, la boîte de dialogue est centrée horizontalement.
ypos
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 ypos 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.
helpfile
Expression de chaîne qui identifie le fichier d'aide à utiliser pour fournir l'aide contextuelle de la boîte de dialogue. Si l'argument helpfile est fourni, l'argument context doit l'être aussi.
context
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 context est fourni, l'argument helpfile 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)
```

Voir aussi
MsgBox, fonction

InStr, fonction

Renvoie la position de la première occurrence d'une chaîne à l'intérieur d'une autre.

InStr([start,]string1, string2[, compare])
Arguments
start
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 start contient la valeur Null, une erreur se produit. L'argument start est requis si l'argument compare est spécifié.
string1
Expression de chaîne faisant l'objet de la recherche.
string2
Expression de chaîne recherchée.
compare
Facultatif. Valeur numérique qui indique le type de comparaison effectué lors de l'évaluation des sous- chaînes. Reportez-vous à la section Paramètres. Si l'argument compare est omis, une comparaison binaire est effectuée.
Paramètres
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 InStr renvoie les valeurs suivantes :

Si la fonction InStr renvoie
string1 est de longueur nulle 0
string1 est Null Null
string2 est de longueur nulle start
string2 est Null Null
string2 n'est pas trouvé 0
string2 est trouvé dans string1 la position de correspondance
start > Len(string2) 0

Notes
Les exemples suivants utilisent la fonction InStr pour rechercher une chaîne :

```
Dim SearchString, SearchChar, MyPos
SearchString ="XXpXXpXXPXXP"   ' Chaîne dans laquelle rechercher.
SearchChar = "P"   ' Recherche "P".
MyPos = Instr(4, SearchString, SearchChar, 1)   ' Comparaison textuelle commençant à la position 4. Renvoie 6.
MyPos = Instr(1, SearchString, SearchChar, 0)   ' Comparaison binaire commençant à la position 1. Renvoie 9.
MyPos = Instr(SearchString, SearchChar)   ' La comparaison est binaire par défaut (le dernier argument est omis).
Renvoie 9.
MyPos = Instr(1, SearchString, "W")   ' Comparaison binaire commençant à la position 1. Renvoie 0 ("W" est
introuvable).
```

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.

Voir aussi
InstrRev, fonction

InStrRev, fonction

Renvoie la position d'une occurrence d'une chaîne dans une autre, à partir de la fin de la chaîne.

InStrRev(string1, string2[, start[, compare]])
Arguments
string1
Expression de chaîne dans laquelle la recherche est effectuée.
string2
Expression de chaîne recherchée.
start
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 start contient la valeur Null, une erreur se produit.
compare
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 Paramètres.
Paramètres
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
string1 a une longueur nulle 0
string1 a la valeur Null Null
string2 a une longueur nulle start
string2 a la valeur Null Null
string2 est introuvable 0
string2 se trouve à l'intérieur de string1 la position à laquelle une correspondance est trouvée.
start > Len(string2) 0

Notes
Les exemples suivants utilisent la fonction InStrRev pour rechercher une chaîne :

```
Dim SearchString, SearchChar, MyPos
SearchString ="XXpXXpXXPXXP"   ' Chaîne dans laquelle rechercher.
SearchChar = "P"   ' Rechercher "P".
MyPos = InstrRev(SearchString, SearchChar, 10, 0)   ' Comparaison binaire commençant à la position 10. Renvoie 9.
MyPos = InstrRev(SearchString, SearchChar, -1, 1)   ' Comparaison textuelle commençant à la dernière position. Renvoie 12.
MyPos = InstrRev(SearchString, SearchChar, 8)   ' La comparaison est binaire par défaut (le dernier argument est omis). Renvoie 0.
```

Remarque La syntaxe de la fonction InStrRev est différente de celle de la fonction InStr.

Voir aussi
Instr, fonction

Int, Fix, fonctions

Renvoient la partie entière d'un nombre.

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.
```

Voir aussi
CInt, fonction | Round, fonction

IsArray, fonction

Renvoie une valeur booléenne indiquant si la variable est un tableau.

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 contient "True".
```

Voir aussi
IsDate, fonction | IsEmpty, fonction | IsNull, fonction | IsNumeric, fonction | IsObject, fonction | VarType, fonction

IsDate, fonction

Renvoie une valeur booléenne indiquant si une expression peut être convertie en date.

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#: NoDate = "Bonjour"
```

```
MyCheck = IsDate(MyDate)   ' Renvoie True.
MyCheck = IsDate(YourDate)   ' Renvoie True.
MyCheck = IsDate(NoDate)   ' Renvoie False.
```

Voir aussi
CDate, fonction | IsArray, fonction | IsEmpty, fonction | IsNull, fonction | IsNumeric, fonction | IsObject, fonction | VarType, fonction

IsEmpty, fonction

Renvoie une valeur booléenne indiquant si une variable a été initialisée.

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 Empty.
MyCheck = IsEmpty(MyVar)   ' Renvoie True.
```

Voir aussi
IsArray, fonction | IsDate, fonction | IsNull, fonction | IsNumeric, fonction | IsObject, fonction | VarType, fonction

IsNull, fonction

Renvoie une valeur booléenne indiquant si une expression contient des données valides ou non (Null).

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 renvoie 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.

Attention 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.
```

Voir aussi
IsArray, fonction | IsDate, fonction | IsEmpty, fonction | IsNumeric, fonction | IsObject, fonction | VarType, fonction

IsNumeric, fonction

Renvoie une valeur booléenne indiquant si une expression peut être évaluée sous la forme d'un nombre.

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.
```

Voir aussi
IsArray, fonction | IsDate, fonction | IsEmpty, fonction | IsNull, fonction | IsObject, fonction | VarType, fonction

IsObject, fonction

Renvoie une valeur booléenne indiquant si une expression référence un objet Automation valide.

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.
```

Voir aussi
IsArray, fonction | IsDate, fonction | IsEmpty, fonction | IsNull, fonction | IsNumeric, fonction | Set, instruction | VarType, fonction

Join, fonction

Renvoie une chaîne créée par la jonction de plusieurs sous-chaînes contenues dans un tableau.

Join(list[, delimiter])
Arguments
list
Un tableau à une dimension contenant les sous-chaînes à joindre.
delimiter

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 delimiter 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(3)
MyArray(0) = "M."
MyArray(1) = "Alfred "
MyArray(2) = "Gautier "
MyArray(3) = "junior"
MyString = Join(MyArray) ' MyString contient "M. Alfred Gautier junior".
```

Voir aussi
Split, fonction

LBound, fonction

Renvoie le plus petit indice disponible pour la dimension indiquée d'un tableau.

LBound(arrayname[, dimension])
Arguments
arrayname
Nom de la variable du tableau ; respecte les conventions standard d'affectation de noms à des variables.
dimension
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 dimension 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.

Voir aussi
Dim, instruction | ReDim, instruction | UBound, fonction

LCase, fonction

Renvoie une chaîne qui a été convertie en minuscules.

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".
```

Voir aussi
UCase, fonction

Left, fonction

Renvoie un nombre spécifié de caractères à partir de la gauche d'une chaîne.

Left(string, length)
Arguments
string
Expression de chaîne à partir de laquelle les caractères situés à l'extrême gauche sont renvoyés. Si string contient Null, la valeur Null est renvoyée.
length
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 string, 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 contient "VBS".
```

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.

Voir aussi
Len, fonction | Mid, fonction | Right, fonction

Len, fonction

Renvoie le nombre de caractères contenus dans une chaîne, ou le nombre d'octets requis pour mémoriser une variable.

Len(string | varname)
Arguments
string
Toute expression de chaîne valide. Si string contient Null, la valeur Null est renvoyée.
varname
Tout nom de variable valide. Si varname 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.

Voir aussi
InStr, fonction

LoadPicture, fonction

Renvoie un objet image. Disponible seulement sur les plates-formes 32 bits.

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

Renvoie le logarithme népérien d'un nombre.

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 :

Logn(x) = Log(x) / 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
```

Voir aussi
Fonctions mathématiques dérivées | Exp, fonction

LTrim, RTrim et Trim, fonctions

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).

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 "vbscript ".
MyVar = RTrim("   vbscript ")   ' MyVar contient "   vbscript".
MyVar = Trim("   vbscript ")   ' MyVar contient "vbscript".
```

Voir aussi
Left, fonction | Right, fonction

Mid, fonction

Renvoie un nombre spécifié de caractères d'une chaîne.

Mid(string, start[, length])
Arguments
string
Expression de chaîne à partir de laquelle les caractères sont renvoyés. Si l'argument string contient Null, la valeur Null est renvoyée.
start
Position du caractère dans l'argument string à partir duquel commence la partie à extraire. Si l'argument start est supérieur au nombre de caractères contenus dans l'argument string, la fonction Mid renvoie une chaîne de longueur nulle.
length
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 start) est inférieur à ceux compris dans l'argument length, tous les caractères entre la position de l'argument start 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 contient "Script".
```

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.

Voir aussi
Left, fonction | Len, fonction | LTrim, RTrim et Trim, fonctions | Right, fonction

Minute

Renvoie un nombre entier compris entre 0 et 59 inclus, représentant la minute de l'heure.

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)
```

Voir aussi
Day, fonction | Hour, fonction | Now, fonction | Second, fonction | Time, fonction

Month, fonction

Renvoie un nombre entier compris entre 1 et 12 inclus, représentant le mois de l'année.

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.
```

Voir aussi
Date, fonction | Day, fonction | Now, fonction | Weekday, fonction | Year, fonction

MonthName, fonction

Renvoie une chaîne indiquant le mois spécifié.

MonthName(month[, abbreviate])
Arguments
month
Désignation numérique du mois. Par exemple, 1 pour janvier, 2 pour février, et ainsi de suite.
abbreviate
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".
```

Voir aussi
WeekDayName, fonction

MsgBox, fonction

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.

MsgBox(prompt[, buttons][, title][, helpfile, context])
Arguments
prompt
Expression de chaîne qui est affichée sous la forme d'un message dans la boîte de dialogue. La longueur maximum de l'argument prompt est environ 1024 caractères, selon la largeur des caractères utilisés. Si l'argument prompt 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)).
buttons
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 buttons est 0.
title
Expression de chaîne affichée dans la barre de titre de la boîte de dialogue. Si vous omettez l'argument title, le nom de l'application s'affiche dans la barre de titre.
helpfile
Expression de chaîne qui identifie le fichier d'aide à utiliser pour fournir l'aide contextuelle de la boîte de dialogue. Si l'argument helpfile est fourni, l'argument context doit aussi l'être. Non disponible sur les plates-formes 16 bits.
context
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 context est fourni, l'argument helpfile doit aussi l'être.
Paramètres
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.
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 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.
```

Voir aussi
InputBox, fonction

Now

Renvoie la date et l'heure en cours en fonction de leur paramétrage dans le système de votre ordinateur.

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
Date, fonction | Day, fonction | Hour, fonction | Minute, fonction | Month, fonction | Second, fonction | Time, fonction | Weekday, fonction | Year, fonction

Oct

Renvoie une chaîne représentant la valeur octale d'un nombre.

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 number 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.
```

Voir aussi
Hex, fonction

Replace, fonction

Renvoie une chaîne dans laquelle une sous-chaîne donnée a été remplacée par une autre sous-chaîne le nombre de fois spécifié.

Replace(expression, find, replacewith[, start[, count[, compare]]])
Arguments
expression
Expression de chaîne contenant une sous-chaîne à remplacer.
find
Sous-chaîne recherchée.
replacewith
Sous-chaîne de remplacement.
start
Facultatif. Position dans l'argument expression 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 count.
count
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 start.
compare
Facultatif. Valeur numérique indiquant le type de comparaison à utiliser lors de l'évaluation des sous-chaînes. Reportez-vous à la section Paramètres. Si elle est omise, la valeur par défaut est 0, comparaison binaire.
Paramètres
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 valeur Replace renvoie les valeurs suivantes :

Si	La fonction Replace renvoie
expression a une longueur nulle	Une chaîne de longueur nulle ("").
expression a la valeur Null	Une erreur.
find a une longueur nulle	Une copie d'expression.
replacewith a une longueur nulle	Une copie d'expression, toutes les occurrences de find étant retirées.
start > Len(expression)	Une chaîne de longueur nulle.
count a une valeur de 0	Une copie d'expression.

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")   ' Comparaison binaire commençant au début de la chaîne. Renvoie "XXYXXPXXY".
MyString = Replace("XXpXXPXXp", "p", "Y",   ' Comparaison textuelle commençant à la position 3. Renvoie "YXXYXXY". 3, -1, 1)
```

Voir aussi
Filter, fonction

RGB, fonction

Renvoie un nombre entier représentant une valeur de couleur RVB.

RGB(red, green, blue)
Arguments
red
Nombre de 0 à 255 représentant la composante rouge de la couleur.
green
Nombre de 0 à 255 représentant la composante verte de la couleur.
blue
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 * 65536))
End fonction
```

Toute valeur d'argument supérieure à 255 est ramenée à 255.

Right, fonction

Renvoie un nombre spécifié de caractères à partir de la droite d'une chaîne.

Right(string, length)
Arguments
string
Expression de chaîne à partir de laquelle les caractères à l'extrême droite sont renvoyés. Si l'argument string contient la valeur Null, la valeur Null est renvoyée.
length
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 string, 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, 6)    ' 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.

Voir aussi
Left, fonction | Len, fonction | Mid, fonction

Rnd, fonction

Renvoie un nombre aléatoire.

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 number est Rnd génère
Inférieur à zéro Le même nombre chaque fois, en utilisant l'argument number 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 :

Int((upperbound - lowerbound + 1) * Rnd + 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.

Voir aussi
Randomize, instruction

Round, fonction

Renvoie un nombre arrondi à un nombre spécifié de positions décimales.

Round(expression[, numdecimalplaces])
Arguments
expression
Expression numérique arrondie.
numdecimalplaces
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.
```

Voir aussi
Int, Fix, fonctions

ScriptEngine, fonction

Renvoie une chaîne représentant le langage de script utilisé.

ScriptEngine
Valeurs renvoyées
La fonction ScriptEngine renvoient l'une des chaînes suivantes :

Chaîne Description

VBScript Indique que Microsoft® Visual Basic® Scripting Edition est le moteur de script en cours.
JScript Indique que Microsoft JScript® est le moteur de script en cours.
VBA 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 nécessaires.
  s = ScriptEngine & " Version "
  s = s & ScriptEngineMajorVersion & "."
  s = s & ScriptEngineMinorVersion & "."
  s = s & ScriptEngineBuildVersion
  GetScriptEngineInfo =  s   ' Renvoie le résultat.
End, fonction
```

Voir aussi
ScriptEngineBuildVersion, fonction | ScriptEngineMajorVersion, fonction | ScriptEngineMinorVersion, fonction

ScriptEngineBuildVersion, fonction

Renvoie le numéro de version du moteur de script employé.

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 informations nécessaires.
  s = ScriptEngine & " Version "
  s = s & ScriptEngineMajorVersion & "."
  s = s & ScriptEngineMinorVersion & "."
  s = s & ScriptEngineBuildVersion
  GetScriptEngineInfo = s   ' Renvoie le résultat.
End, fonction
```

Voir aussi
ScriptEngine, fonction | ScriptEngineMajorVersion, fonction | ScriptEngineMinorVersion, fonction

ScriptEngineMajorVersion, fonction

Renvoie le numéro de version principal du moteur de script employé.

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 informations nécessaires.
  s = ScriptEngine & " Version "
  s = s & ScriptEngineMajorVersion & "."
```

```
  s = s & ScriptEngineMinorVersion & "."
  s = s & ScriptEngineBuildVersion
  GetScriptEngineInfo = s   ' Renvoie le résultat.
End, fonction
```

Voir aussi
ScriptEngine, fonction | ScriptEngineBuildVersion, fonction | ScriptEngineMinorVersion, fonction

ScriptEngineMinorVersion, fonction

Renvoie le numéro de version secondaire du moteur de script employé.

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 informations nécessaires.
  s = ScriptEngine & " Version "
  s = s & ScriptEngineMajorVersion & "."
  s = s & ScriptEngineMinorVersion & "."
  s = s & ScriptEngineBuildVersion
  GetScriptEngineInfo = s   ' Renvoie le résultat.
End, fonction
```

Voir aussi
ScriptEngine, fonction | ScriptEngineBuildVersion, fonction | ScriptEngineMajorVersion, fonction

Second, fonction

Renvoie un nombre entier compris entre 0 et 59 inclus, représentant la seconde de la minute.

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.
```

Voir aussi
Day, fonction | Hour, fonction | Minute, fonction | Now, fonction | Time, fonction

SetLocale, fonction

Définit les paramètres régionaux globaux et renvoie les paramètres régionaux précédents.

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, sa région 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.

Sgn, fonction

Renvoie un entier indiquant le signe d'un nombre.

Sgn(number)
L'argument number représente toute expression numérique valide.

Valeurs renvoyées
La fonction Sgn comprend les valeurs renvoyées suivantes :

Si number 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.
```

Voir aussi
Abs, fonction

Sin, fonction

Renvoie le sinus d'un angle.

Sin(number)
L'argument number représente toute expression numériques 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.
```

Voir aussi
Atn, fonction | Cos, fonction | Fonctions mathématiques dérivées | Tan, fonction

Space, fonction

Renvoie une chaîne composée d'un nombre spécifié d'espaces.

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 espaces.
MyString = "Bonjour" & Space(10) & "Bonsoir" ' Insère 10 espaces entre les deux chaînes.
```

Voir aussi
String, fonction

Split, fonction

Renvoie un tableau à une dimension commençant par zéro contenant le nombre spécifié de sous-chaînes.

Split(expression[, delimiter[, count[, compare]]])
Arguments
expression
Expression de chaîne contenant des sous-chaînes et des séparateurs. Si l'argument expression 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.
delimiter
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 delimiter est une chaîne de longueur nulle, un tableau à un élément contenant toute la chaîne expression est renvoyée.
count
Facultatif. Nombre de sous-chaînes à renvoyer ; -1 indique que toutes les sous-chaînes sont renvoyées.
compare
Facultatif. Valeur numérique indiquant le type de comparaison à utiliser lors de l'évaluation des sous-chaînes. Reportez-vous à la section Paramètres.
Paramètres
L'argument compare peut prendre les valeurs suivantes :

Constante	Valeur	Description
vbBinaryCompare	0	Effectue une comparaison binaire.
vbTextCompare	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) contient "VBScript".
' MyArray(1) contient "est".
' MyArray(2) contient "super !".
Msg = MyArray(0) & " " & MyArray(1)
Msg = Msg   & " " & MyArray(2)
MsgBox Msg
```

Voir aussi
Join, fonction

Sqr, fonction

Renvoie la racine carrée d'un nombre.

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

Renvoie une valeur indiquant le résultat d'une comparaison de chaîne.

StrComp(string1, string2[, compare])
Arguments
string1
Toute expression de chaîne valide.
string2
Toute expression de chaîne valide.
compare
Facultatif. Valeur numérique qui indique le type de comparaison à effectuer pour l'évaluation des chaînes. Si l'argument compare est omis, une comparaison binaire est effectuée. Les valeurs sont indiquées dans la section Paramètres.
Paramètres
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
string1 est inférieur à string2	-1
string1 est égal à string2	0
string1 est supérieur à string2	1
string1 ou string2 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 les variables.
MyComp = StrComp(MyStr1, MyStr2, 1)   ' Renvoie 0.
MyComp = StrComp(MyStr1, MyStr2, 0)   ' Renvoie -1.
MyComp = StrComp(MyStr2, MyStr1)   ' Renvoie 1.
```

String, fonction

Renvoie une chaîne constituée d'un caractère répété sur la longueur spécifiée.

String(number, character)
Arguments
number
Longueur de la chaîne renvoyée. Si l'argument number contient Null, la valeur Null est renvoyée.
character

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 character 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 "AAAAAAAAAA".
```

Voir aussi
Space, fonction

StrReverse, fonction

Renvoie une chaîne contenant des caractères dont l'ordre a été inversé par rapport à une chaîne donnée.

StrReverse(string1)
L'argument string1 est la chaîne pour laquelle l'inversion des caractères a été demandée. Si l'argument string1est 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".
```

Tan, fonction

Renvoie la tangente d'un angle.

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 radians.
MyCotangent = 1 / Tan(MyAngle)   ' Calculer la cotangente.
```

Voir aussi
Atn, fonction | Cos, fonction | Fonctions mathématiques dérivées | Sin, fonction

Time, fonction

Renvoie un Variant de sous-type Date indiquant l'heure système en cours.

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.
```

Voir aussi
Date, fonction

Timer, fonction

Renvoie le nombre de secondes qui se sont écoulées depuis 00:00 (minuit).

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
```

Voir aussi
Randomize, instruction

TimeSerial, fonction

Renvoie un Variant de sous-type Date contenant l'heure correspondant à des éléments spécifiques d'heure, de minute et de seconde.

TimeSerial(hour, minute, second)
Arguments
hour
Nombre entre 0 (12:00) et 23 (11:00) inclus ou expression numérique.
minute
Toute expression numérique.
second
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 5:45:00.
```

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.

Voir aussi
DateSerial, fonction | DateValue, fonction | Hour, fonction | Minute, fonction | Now, fonction | Second, fonction | TimeValue, fonction

TimeValue

Renvoie un Variant de sous-type Date contenant l'heure.

TimeValue(time)
L'argument time est habituellement une expression de chaîne représentant une 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 contient 16:35:17.
```

Voir aussi
DateSerial, fonction | DateValue, fonction | Hour, fonction | Minute, fonction | Now, fonction | Second, fonction | TimeSerial, fonction

TypeName, fonction

Renvoie une chaîne qui fournit des informations de sous-type Variant sur une variable.

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()".
```

Voir aussi
IsArray, fonction | IsDate, fonction | IsEmpty, fonction | IsNull, fonction | IsNumeric, fonction | IsObject, fonction | VarType, fonction

UBound, fonction

Renvoie le plus grand indice disponible pour la dimension indiquée d'un tableau.

UBound(arrayname[, dimension])
Arguments
arrayname
Nom de la variable tableau ; respectez les conventions standard d'affectation de noms à des variables.
dimension
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 dimension 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

Voir aussi
Dim, instruction | LBound, fonction | ReDim, instruction

UCase, fonction

Renvoie une chaîne qui a été convertie en majuscules.

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 "Bonjour".
```

Voir aussi
LCase, fonction

VarType, fonction

Renvoie une valeur indiquant le sous-type d'une variable.

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.
```

Voir aussi
IsArray, fonction | IsDate, fonction | IsEmpty, fonction | IsNull, fonction | IsNumeric, fonction | IsObject, fonction | TypeName, fonction

Weekday, fonction

Renvoie un nombre entier représentant le jour de la semaine.

Weekday(date, [firstdayofweek])
Arguments
date
Toute expression représentant une date. Si l'argument date contient date la valeur Null, la valeur Null est renvoyée.
firstdayofweek
Une constante qui spécifie le premier jour de la semaine. Si cette valeur est omise, vbSunday est utilisé par défaut.
Paramètres
L'argument firstdayofweek peut prendre les valeurs suivantes :

Constante Valeur Description
VbUseSystemDayOfWeek 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 parce que MyDate représente un vendredi.
```

Voir aussi
Date, fonction | Day, fonction | Month, fonction | Now, fonction | Year, fonction

WeekdayName, fonction

Renvoie une chaîne indiquant le jour de la semaine spécifié.

WeekdayName(weekday, abbreviate, firstdayofweek)
Arguments
weekday
Désignation numérique du jour de la semaine. La valeur numérique du jour de la semaine dépend de la valeur du paramètre firstdayofweek.
abbreviate
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.
firstdayofweek
Facultatif. Valeur numérique indiquant le premier jour de la semaine.
Paramètres
L'argument firstdayofweek prend les valeurs suivantes :

Constante Valeur Description
VbUseSystemDayOfWeek 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.
```

Voir aussi
MonthName, fonction

Year, fonction

Renvoie un nombre entier représentant l'année.

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.
```

Voir aussi
Date, fonction | Day, fonction | Month, fonction | Now, fonction | Weekday, fonction

Liste des Méthodes de VBScript

Clear Execute Raise Replace
Test

Clear, méthode

Réinitialise les propriétés de l'objet Err.

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.
```

Voir aussi
Description, propriété | Err, objet | Number, propriété | On Error, instruction | Raise, méthode | Source, propriété

Application : Err, objet

Execute, méthode

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

object.Execute(string)
Arguments
object
Requis. Il s'agit toujours du nom d'un objet RegExp.
string
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 Execute renvoie une collection Matches contenant un objet Match pour chaque correspondance trouvée dans l'élément string. Execute 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 régulière.
  regEx.Pattern = patrn          ' Définit les critères.
  regEx.IgnoreCase = True        ' Ignore la casse.
  regEx.Global = True          ' Définit une application globale.
  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 correspondance est '"
    RetStr = RetStr & Match.Value & "'." & vbCRLF
  Next
  RegExpTest = RetStr
End Function
MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Voir aussi
Replace, méthode | Test, méthode

Application : RegExp, objet

Raise, méthode

Génère une erreur d'exécution.

object.Raise(number, source, description, helpfile, helpcontext)
Arguments
object
Toujours l'objet Err.
number
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.
source
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 project.class. Si rien n'est spécifié, l'identificateur de ressource du projet VBScript en cours est utilisé.
description
Une expression de chaîne décrivant l'erreur. Si elle n'est pas spécifiée, la valeur contenue dans l'argument number est examinée. Si elle peut être associée à un code d'erreur d'exécution VBScript, une chaîne fournie par VBScript est utilisée comme description. S'il n'existe aucune erreur VBScript correspondant à la valeur de l'argument number, un message d'erreur générique est utilisé.
helpfile
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.
helpcontext
L'identificateur de contexte identifiant une rubrique contenue dans l'argument helpfile 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 number 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.
```

Voir aussi

Clear, méthode | Description, propriété | Err, objet | Number, propriété | Source, propriété

Application : Err, objet

Replace, méthode

Remplace le texte trouvé dans une recherche d'expression régulière.

object.Replace(string1, string2)
Arguments
object
Requis. Il s'agit toujours du nom d'un objet RegExp.
string1
Requis. String1 est la chaîne de caractères dans laquelle le remplacement a été effectué.
string2
Requis. String2 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 régulière.
  regEx.Pattern = patrn           ' Définit les critères.
  regEx.IgnoreCase = True         ' Ignore la casse.
  ReplaceTest = regEx.Replace(str1, replStr)   ' Effectue le remplacement
End Function

MsgBox(ReplaceTest("renard", "chat"))      ' Remplace 'renard' par 'chat'.
```

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 les paires de mots.
```

Voir aussi
Execute, méthode | Test, méthode

Application : RegExp, objet

Test, méthode

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.

object.Test(string)
Arguments
object
Requis. Il s'agit toujours du nom d'un objet RegExp.
string
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 spécifiées 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 régulière.
  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 trouvées."
  Else
    RegExpTest = "Aucune correspondance n'a été trouvée."
  End If
End Function
MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Voir aussi
Execute, méthode | Replace, méthode

Application : RegExp, objet

Divers

Tableau des ID de langue (LCID)
Le tableau suivant répertorie les ID de langue (LCID).

Description de la langue Chaîne abrégée Valeur hex. Valeur décimale Description de la langue Chaîne abrégée Valeur hex Valeur décimale
Africaans af 0x0436 1078 Islandais is 0x040F 1039
Albanais sq 0x041C 1052 Indonésien id 0x0421 1057
Arabe - Émirats Arabes Unis. ar-ae 0x3801 14337 Italien - Italie it-it 0x0410 1040
Arabe - Bahreïn ar-bh 0x3C01 15361 Italien - Suisse it-ch 0x0810 2064
Arabe - Algérie ar-dz 0x1401 5121 Japonais ja 0x0411 1041
Arabe - Égypte ar-eg 0x0C01 3073 Coréen ko 0x0412 1042
Arabe - Iraq ar-iq 0x0801 2049 Letton lv 0x0426 1062
Arabe - Jordanie ar-jo 0x2C01 11265 Lituanien lt 0x0427 1063
Arabe - Koweït ar-kw 0x3401 13313 FYRO Macédonien mk 0x042F 1071
Arabe - Liban ar-lb 0x3001 12289 Malais - Malaisie ms-my 0x043E 1086
Arabe - Libye ar-ly 0x1001 4097 Malais – Brunei ms-bn 0x083E 2110
Arabe - Maroc ar-ma 0x1801 6145 Maltais mt 0x043A 1082
Arabe - Oman ar-om 0x2001 8193 Marathi mr 0x044E 1102
Arabe - Qatar ar-qa 0x4001 16385 Norvégien - Bokmaal no 0x0414 1044
Arabe - Arabie Saoudite ar-sa 0x0401 1025 Norvégien - Nynorsk No-no 0x0814 2068
Arabe - Syrie ar-sy 0x2801 10241 Polonais pl 0x0415 1045
Arabe - Tunisie ar-tn 0x1C01 7169 Portugais - Portugal pt-pt 0x0816 2070
Arabe - Yémen ar-ye 0x2401 9217 Portugais - Brésil pt-br 0x0416 1046
Arménien hy 0x042B 1067 Rhéto – Roman rm 0x0417 1047
Azéri - Latin az-az 0x042C 1068 Roumain - Roumanie ro 0x0418 1048
Azéri - Cyrillique az-az 0x082C 2092 Roumain - Moldavie ro-mo 0x0818 2072
Basque eu 0x042D 1069 Russe ru 0x0419 1049
Biélorusse be 0x423 1059 Russe - Moldavie ru-mo 0x0819 2073
Bulgare bg 0x402 1026 Sanskrit sa 0x044F 1103
Catalan ca 0x403 1027 Serbe - Cyrillique sr-sp 0x0C1A 3098
Chinois – Rép. Populaire de Chine zh-cn 0x804 2052 Serbe - Latin sr-sp 0x081A 2074
Chinois – Hong-Kong zh-hk 0x0C04 3076 Setswana tn 0x0432 1074
Chinois - Macao zh-mo 0x1404 5124 Slovène sl 0x0424 1060
Chinois - Singapour Zh-sg 0x1004 4100 Slovaque sk 0x041B 1051
Chinois - Taïwan zh-tw 0x0404 1028 Sorbian sb 0x042E 1070
Croate hr 0x041A 1050 Espagnol - Espagne es-es 0x0C0A 1034
Tchèque cs 0x0405 1029 Espagnol - Argentine es-ar 0x2C0A 11274
Danois de 0x406 1030 Espagnol - Bolivie es-bo 0x400A 16394
Néerlandais - Pays-Bas nl-nl 0x0413 1043 Espagnol - Chili es-cl 0x340A 13322
Néerlandais - Belgique nl-be 0x813 2067 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-ca 0x1009 4105 Espagnol - Équateur es-ec 0x300A 12298
Anglais - Caraïbes en-cb 0x2409 9225 Espagnol - Guatemala es-gt 0x100A 4106
Anglais - Irlande en-ie 0x1809 6153 Espagnol - Honduras es-hn 0x480A 18442
Anglais - Jamaïque en-jm 0x2009 8201 Espagnol - Mexique es-mx 0x4C0A 19466
Anglais - Philippines en-ph 0x3409 13321 Espagnol - Panama es-pa 0x180A 6154
Anglais – Afrique du Sud en-za 0x1C09 7177 Espagnol - Pérou es-pe 0x280A 10250
Anglais - Trinidad en-tt 0x2C09 11273 Espagnol – Puerto Rico es-pr 0x500A 20490
Anglais – Royaume-Uni en-gb 0x0809 2057 Espagnol - Paraguay es-py 0x3C0A 15370
Anglais – États-Unis en-us 0x0409 1033 Espagnol – El Salavador es-sv 0x440A 17418
Estonien et 0x0425 1061 Espagnol - Uruguay es-uy 0x380A 14346
Farsi fa 0x0429 1065 Espagnol - Venezuela es-ve 0x200A 8202
Finlandais fi 0x040B 1035 Sutu sx 0x0430 1072
Féroïen fo 0x0438 1080 Swahili sw 0x0441 1089
Français - France fr-fr 0x040C 1036 Suédois - Suède sv-se 0x041D 1053
Français - Belgique fr-be 0x080C 2060 Suédois - Finlande sv-fi 0x081D 2077
Français - Canada fr-ca 0x0C0C 3084 Tamoul ta 0x0449 1097

Français - Luxembourg fr-lu 0x140C 5132 Tatar tt 0x0444 1092
Français - Suisse fr-ch 0x100C 4108 Thaï th 0x041E 1054
Gaélique - Irlande gd-ie 0x083C 2108 Turque tr 0x041F 1055
Gaélique - Écosse gd 0x043C 1084 Tsonga ts 0x0431 1073
Allemand - Allemagne de-de 0x0407 1031 Ukrainien uk 0x0422 1058
Allemand - Autriche de-at 0x0C07 3079 Ourdou ur 0x0420 1056
Allemand - Liechtenstein de-li 0x1407 5127 Ouzbek - Cyrillique uz-uz 0x0843 2115
Allemand - Luxembourg de-lu 0x1007 4103 Ouzbek - Latin uz-uz 0x0443 1091
Allemand - Suisse de-ch 0x0807 2055 Vietnamien vi 0x042A 1066
Grec el 0x0408 1032 Xhosa xh 0x0434 1076
Hébreu he 0x040D 1037 Yiddish yi 0x043D 1085
Hindi hi 0x0439 1081 Zoulou zu 0x0435 1077
Hongrois hu 0x040E 1038

Liste des Objets et Collections de VBScript

Class Err Matches RegExp
SubMatches

Class, objet

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
Événements
Classe, objet Événements

Voir aussi
Class, instruction | Dim, instruction | Set, instruction

Class, objet : événements

L'objet Class, objet permet d'accéder aux événements de la classe.

Événements
Initialize, événement

Terminate, événement

Matches, collection

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 régulière.
  regEx.Pattern = patrn   ' Définit les critères.
  regEx.IgnoreCase = True   ' Définit le respect de 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 correspondance est "
    RetStr = RetStr & Match.Value & "." & vbCRLF
  Next
  RegExpTest = RetStr
End Function
MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Voir aussi
For Each...Next, instruction | Match, objet | RegExp, objet | SubMatches, collection

Err Objet

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

Propriétés et méthodes

Voir aussi
On Error, instruction

Match, objet

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 régulière.
  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 position "
    RetStr = RetStr & Match.FirstIndex & ". La valeur de la correspondance est '"
    RetStr = RetStr & Match.Value & "'." & vbCRLF
  Next
  RegExpTest = RetStr
End Function
MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Propriétés
Match, objet Propriétés

Voir aussi
Matches, collection | RegExp, objet | SubMatches, collection

Match, objet : propriétés

L'objet Match, objet permet d'accéder aux propriétés en lecture seule de la correspondance d'une expression régulière.

Propriétés
FirstIndex, propriété

Length, propriété

Value, propriété

Regular Expression (RegExp), objet

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 régulière.
  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 correspondance est '"
    RetStr = RetStr & Match.Value & "'." & vbCRLF
  Next
  RegExpTest = RetStr
End Function
MsgBox(RegExpTest("is.", "IS1 is2 IS3 is4"))
```

Propriétés et méthodes
Expression régulière Objet Propriétés et méthodes

Voir aussi
Match, objet | Matches, collection

SubMatches, Collection

Collection de chaînes de sous-correspondance d'une expression régulière.

Notes
Une collection SubMatches contient des chaînes de sous-correspondance individuelles, et peut être créée seulement à l'aide de la méthode Execute de l'objet RegExp. La collection SubMatches possède uniquement la propriété d'être en lecture seule.

Lorsqu'une expression régulière est exécutée, il en résulte aucune, une ou plusieurs sous-correspondances lorsque les sous-expressions sont comprises entre des parenthèses de capture. Chaque élément de la collection SubMatches est la chaîne trouvée et capturée par l'expression régulière.

Le code suivant montre la façon d'obtenir une collection SubMatches en utilisant une recherche d'expression régulière et la façon d'accéder à ses membres individuels :

```
Function SubMatchTest(inpStr)
  Dim oRe, oMatch, oMatches
  Set oRe = New RegExp
  ' Rechercher une adresse électronique(pas une RegExp parfaite)
  oRe.Pattern = "(\w+)@(\w+)\.(\w+)"
  ' Obtenir la collection Matches
  Set oMatches = oRe.Execute(inpStr)
  ' Obtenir le premier élément de la collection Matches
  Set oMatch = oMatches(0)
  ' Créer la chaîne de résultats.
  ' L'objet Match est la correspondance exacte - dragon@xyzzy.com
  retStr = "L'adresse électronique est : " & oMatch & vbNewline
  ' Obtenir les sous-correspondances de l'adresse.
  retStr = retStr & "L'alias de l'adresse est : " & oMatch.SubMatches(0)  ' dragon
  retStr = retStr & vbNewline
  retStr = retStr & "La société est : " & oMatch. SubMatches(1)' xyzzy
  SubMatchTest = retStr
End Function

MsgBox(SubMatchTest("Veuillez envoyer un courrier électronique à dragon@xyzzy.com. Merci!"))
```

Voir aussi
For Each...Next, instruction | Match, Objet | Matches, Collection |RegExp, Objet

Liste des Propriétés de VBScript

Description	FirstIndex	Global	HelpContext
HelpFile	IgnoreCase	Length	Number
Pattern	Source	Value	

Description, propriété

Renvoie ou définit une chaîne descriptive associée à une erreur.

object.Description [= stringexpression]
Arguments
object
Toujours l'objet Err.
stringexpression
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.
```

Voir aussi
Err, objet | HelpContext, propriété | HelpFile, propriété | Number, propriété | Source, propriété

Application : Err, objet

FirstIndex, propriété

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

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 position "
    RetStr = RetStr & Match.FirstIndex & ". La valeur de la correspondance est "'
    RetStr = RetStr & Match.Value & "'." & vbCRLF
  Next
  RegExpTest = RetStr
End Function
```

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

Voir aussi
Length, propriété | Value, propriété

Application : Match, objet

Global, propriété

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.

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 régulière.
  regEx.Pattern = patrn   ' Définit les critères.
  regEx.IgnoreCase = True   ' Ignore la casse.
  regEx.Global = True   ' Définit une application globale.
  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 correspondance est '"
    RetStr = RetStr & Match.Value & "'." & vbCRLF
  Next
  RegExpTest = RetStr
End Function
MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Voir aussi
IgnoreCase, propriété | Pattern, propriété

Application : RegExp, objet

HelpContext, propriété

Définit ou renvoie un identificateur de contexte pour une rubrique dans un fichier d'aide.

object.HelpContext [= contextID]
Arguments
object
Correspond toujours à l'objet Err.
contextID
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 employé. 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.Helpfile & " pour" & _
  " le contexte d'aide suivant: " & Err.HelpContext
  MsgBox Msg, , "erreur: " & Err.Description, Err.Helpfile, Err.HelpContext
End If
```

Voir aussi
Description, propriété | HelpFile, propriété | Number, propriété | Source, propriété

Application : Err, objet

HelpFile, propriété

Définit ou renvoie le chemin complet d'un fichier d'aide.

object.HelpFile [= contextID]
Arguments
object
Correspond toujours à l'objet Err.
contextID
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 " & Err.Helpfile & " pour" & _
  " le contexte d'aide suivant: " & Err.HelpContext
  MsgBox Msg, , "erreur: " & Err.Description, Err.Helpfile, Err.HelpContext
End If
```

Voir aussi
Description, propriété | HelpContext, propriété | Number, propriété | Source, propriété

Application : Err, objet

IgnoreCase, propriété

Définit ou renvoie une valeur booléenne indiquant si les critères de recherche distinguent les minuscules et les majuscules.

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 régulière.
  regEx.Pattern = patrn   ' Définit les critères.
  regEx.IgnoreCase = True   ' Ignore la casse.
  regEx.Global = True   ' Définit une application globale.
  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 correspondance est '"
    RetStr = RetStr & Match.Value & "'." & vbCRLF
  Next
  RegExpTest = RetStr
End Function
MsgBox(RegExpTest("is", "IS1 is2 IS3 is4"))
```

Voir aussi
Global, propriété | Pattern, propriété

Application : RegExp, objet

Length, propriété

Renvoie la longueur d'une correspondance trouvée dans une chaîne de recherche.

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 régulière.
  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 position "
    RetStr = RetStr & Match.FirstIndex & ". La longueur de correspondance est "
    RetStr = RetStr & Match.Length
    RetStr = RetStr & " caractères." & vbCRLF
  Next
  RegExpTest = RetStr
End Function
MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Voir aussi
FirstIndex, propriété | Value, propriété

Application : Match, objet

Number, propriété

Renvoie ou définit une valeur numérique spécifiant une erreur. Number est la propriété par défaut de l'objet Err.

object.Number [= errornumber]
Arguments
object
Toujours l'objet Err.
errornumber
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 N°1.
MsgBox ("Erreur N°" & CStr(Err.Number) & " " & Err.Description)
Err.Clear   ' Efface l'erreur.
```

Voir aussi
Description, propriété | HelpContext, propriété | HelpFile, propriété | Err, objet | Source, propriété

Application : Err, objet

Pattern, propriété

Définit ou renvoie les critères de recherche de l'expression régulière.

object.Pattern [= "searchstring"]
Arguments
object
Requis. Il s'agit toujours d'une variable objet RegExp.
searchstring
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 utilisées.

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) Recherche le modèle 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 "foo".
{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] Plage 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] Plage 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.
\w Correspond à tout caractère représentant un mot et incluant un trait de soulignement. Équivaut à "[A-Za-z0-9_]".
\W Correspond à tout caractère ne représentant pas un mot. Équivaut à "[^A-Za-z0-9_]".
\num Correspond à num, où num est un entier positif. Fait référence aux correspondances mémorisées. Par exemple, "(.)\1" correspond à deux caractères identiques consécutifs.
\n Correspond à n, où n 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.
\xn Correspond à n, où n 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 régulière.
  regEx.Pattern = patrn   ' Définit les critères.
  regEx.IgnoreCase = True   ' Ignore la casse.
  regEx.Global = True   ' Définit une application globale.
  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 correspondance est '"
    RetStr = RetStr & Match.Value & "'." & vbCRLF
  Next
  RegExpTest = RetStr
End Function
MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Voir aussi
Global, propriété | IgnoreCase, propriété

Application : RegExp, objet

Source, propriété

Renvoie ou définit le nom de l'objet ou de l'application qui est à l'origine de l'erreur.

object.Source [= stringexpression]
Arguments

object
Toujours l'objet Err.
stringexpression
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.Source)
Err.Clear   ' Efface l'erreur.
```

Voir aussi
Description, propriété | Err, objet | HelpContext, propriété | HelpFile, propriété | Number, propriété | On Error, instruction

Application : Err, objet

Value, propriété

Renvoie la valeur ou le texte d'une correspondance trouvée dans une chaîne de recherche.

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 régulière.
  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 position "
    RetStr = RetStr & Match.FirstIndex & ". La valeur de la correspondance est "'
    RetStr = RetStr & Match.Value & "'." & vbCRLF
  Next
  RegExpTest = RetStr
End Function
MsgBox(RegExpTest("est.", "IS1 is2 IS3 is4"))
```

Voir aussi
FirstIndex, propriété | Length, propriété

Application : Match, objet