ADOBE® ILLUSTRATOR® CS5

GUIDE DES SCRIPTS POUR ADOBE ILLUSTRATOR CS5

Adobe

Guide des scripts pour Adobe Illustrator CS5

Sommaire

1 Introduction

Ce guide décrit l'interface de script pour Adobe® Illustrator® CS5.

Si vous êtes novice en matière de scripts ou que vous souhaitez obtenir des informations sur la manière d'utiliser les différents langages de script, reportez-vous au document *Introduction aux scripts Adobe*.

Qu'est-ce qu'un script ?

Un script est une série de commandes demandant à Illustrator d'exécuter une ou plusieurs opérations. Ces opérations peuvent être simples et n'affecter qu'un objet du document actuel, ou complexes et affecter des objets dans tous vos documents Illustrator. Ces opérations peuvent aussi impliquer d'autres applications, telles que des programmes de traitement de texte, des tableurs ou des logiciels de gestion de base de données.

La plupart des blocs de construction du script correspondent aux outils Illustrator, aux menus, aux panneaux et aux boîtes de dialogue pour lesquels vous possédez une expertise. Si vous savez ce que vous souhaitez exécuter par le biais d'Illustrator, vous pouvez écrire un script à cet effet.

Pourquoi utiliser les scripts ?

La conception graphique est un domaine où la créativité est reine, mais certains aspects de la conception ne sont pas du tout créatifs. Vous aurez sans doute remarqué que le positionnement et le repositionnement d'images, la rectification d'erreurs dans le texte et la préparation des fichiers pour impression chez un fournisseur de services de réglage d'image empiètent largement sur le temps dont vous disposez pour le travail créatif à proprement parler.

Si vous investissez un peu de temps et d'efforts, vous pouvez apprendre à écrire des scripts courts et simples qui effectuent les opérations répétitives à votre place. Au fur et à mesure du développement de vos compétences, vous pourrez écrire des scripts de plus en plus complexes.

Les scripts peuvent également améliorer votre créativité en effectuant rapidement des opérations que vous pourriez ne pas avoir le temps de tester. Par exemple, vous pouvez écrire un script pour créer systématiquement une série d'objets, en modifiant les propriétés de position, contour et fond de nouveaux objets à un moment donné. Vous pouvez également écrire un script qui accède aux fonctions de matrice de transformation intégrée afin d'étirer, de mettre à l'échelle et de distordre une série d'objets. Sans script, vous perdriez certainement une partie du potentiel créatif de telles techniques qui requièrent une masse importante de travail.

Qu'est-ce qu'une action ?

Les actions et les scripts sont deux manières d'automatiser des opérations répétitives, mais elles fonctionnent de deux façons différentes :

- Les actions utilisent une interface utilisateur de programme. Lors de l'exécution d'une action, les choix de menus sont exécutés, les objets sont sélectionnés et les tracés enregistrés sont créés. Les scripts n'utilisent pas d'interface utilisateur de programmes pour effectuer les opérations et peuvent s'exécuter plus rapidement que les actions.
- Les actions possèdent des fonctionnalités très limitées pour obtenir des informations et y répondre. Vous ne pouvez pas ajouter de logique conditionnelle à une action. Par conséquent, les actions ne peuvent pas prendre de décisions reposant sur la situation actuelle, telles que le changement de type de contour des rectangles mais pas des ellipses. Les scripts sont capables d'obtenir des informations, de prendre des décisions et d'effectuer des calculs reposant sur les informations qu'ils reçoivent d'Illustrator.
- Un script peut exécuter une action, mais les actions ne peuvent pas exécuter de scripts.

Prise en charge du langage de script dans Adobe Illustrator CS5

Les scripts d'Illustrator prennent en charge les scripts VBScript et JavaScript pour Windows, et les scripts AppleScript et JavaScript pour Mac OS.

Extensions de scripts de fichiers

Pour qu'un fichier soit reconnu par Adobe Illustrator CS5 en tant que fichier de script valide, il doit posséder un nom d'extension de fichier correct :

Type de script	Type de fichier (extension)	Plates-formes
AppleScript	fichier OSAS (`.scpt`) de script compilé (sans extension)	Mac OS
JavaScript ou ExtendScript	texte (`.js` ou `.jsx`)	Windows Mac OS
VBScript	texte (`.vbs`)	Windows

Options de développement JavaScript

Vous pouvez utiliser l'utilitaire d'écriture de scripts ExtendScript Toolkit pour créer des scripts JavaScript spécialement conçus pour Illustrator. Vous pouvez aussi utiliser Creative Suite Extension Builder (CS Extension Builder) pour développer des *extensions CS* en ActionScript. Les extensions CS sont basées sur le format Flash (SWF) et sont potentiellement compatibles avec toute une gamme d'applications Creative Suite.

Développement d'une extension CS5 en ActionScript

Dans Creative Suite 5, les applications disposent d'une infrastructure d'extensibilité qui permet aux développeurs d'étendre les capacités des applications. Cette infrastructure repose sur la technologie Flash/Flex et chaque extension CS5 se présente sous la forme d'un fichier Flash (SWF) compilé. Creative Suite 5 inclut le gestionnaire d'extensions afin d'activer l'installation d'extensions CS5.

Adobe Kuler est un exemple d'extension CS5 fourni avec les produits ciblés. Kuler dispose d'une interface utilisateur cohérente à travers l'ensemble des applications de la suite, mais chaque application hôte utilise sa propre logique, spécifiquement adaptée.

L'interface utilisateur d'une extension est écrite en ActionScript, à l'aide de l'infrastructure Flex. En général, pour accéder à une extension CS5, vous pouvez passer par son propre élément dans le menu Extensions de l'application. CS Extension Builder permet de concevoir l'interface utilisateur de façon interactive à l'aide de la vue de conception de FlashBuilder. CS Extension Builder permet également de développer toute la logique de l'application pour l'extension CS5 en ActionScript. Vous pouvez développer et déboguer l'extension dans l'environnement familier de FlashBuilder.

Pour développer la logique de l'application, il est conseillé d'utiliser la bibliothèque Creative Suite ActionScript Wrapper Library (`CSAWLib`), qui expose le modèle d'objets de document (DOM) du script pour chaque application hôte sous la forme d'une bibliothèque ActionScript. Ceci est étroitement intégré à l'environnement CS Extension Builder, qui inclut des assistants afin de vous aider à créer la structure de base de l'extension, puis à exécuter et à déboguer le code des applications de la suite, telles qu'Adobe InDesign, Photoshop et Illustrator.

Les méthodes, les propriétés et le comportement du modèle d'objets de document du script sont tels que le décrit la *référence de scripts JavaScript* pour l'application hôte. Pour plus d'informations sur l'utilisation de CS Extension Builder et des bibliothèques de wrappers, consultez la documentation du kit de développement logiciel Creative Suite, accessible depuis le système d'aide de Flash Builder ou d'Eclipse lorsque vous avez installé CS Extension Builder.

Fonctions ExtendScript

Si vous rédigez des scripts spécifiques à Illustrator qui utilisent directement le modèle d'objets de document JavaScript Illustrator, vous créez des fichiers ExtendScript qui se distinguent par l'extension `.jsx`. En attribuant une extension `.jsx` aux fichiers JavaScript (au lieu de l'extension standard `.js`), vous pouvez tirer parti des fonctions et des outils ExtendScript.

ExtendScript propose toutes les fonctions JavaScript standard, ainsi qu'un environnement de développement et de débogage, l'utilitaire d'écriture de scripts ExtendScript Toolkit (ESTK). ESTK est installé avec toutes les applications Adobe capables de prendre en charge les scripts ; il s'agit de l'éditeur par défaut pour les fichiers JSX. ESTK comprend un afficheur de modèle d'objet qui contient une documentation complète concernant les méthodes et propriétés des objets JavaScript. Pour plus de détails sur l'accès à ESTK et à l'afficheur de modèle d'objet, reportez-vous à la section « Affichage du modèle d'objet JavaScript », page 9.

ExtendScript offre également divers outils et utilitaires tels que :

- un utilitaire de localisation ;
- des outils qui vous permettent de combiner les scripts et de les diriger vers des applications spécifiques ;
- une représentation de fichiers et de dossiers indépendante des plates-formes ;
- des outils pour construire des interfaces utilisateur dans vos scripts ;
- un cadre de messagerie électronique qui vous permet d'envoyer et de recevoir des scripts et des données dans les applications Adobe adaptées aux scripts.

Toutes ces fonctions sont disponibles, que vous utilisiez le modèle d'objets de document directement avec un fichier JSX ou indirectement via la bibliothèque de wrappers ActionScript et CS Extension Builder. Pour plus de détails sur ces fonctions et sur d'autres fonctions, reportez-vous au *Guide des outils JavaScript*.

Affichage des exemples de scripts

Adobe fournit des exemples de scripts pour de nombreux objets, propriétés et méthodes dans le modèle d'objets de document Illustrator CS5. Vous pouvez consulter des exemples de scripts à deux emplacements :

- Dans le dossier `/Scripting/Sample Scripts` du répertoire d'installation d'Illustrator CS5
- Dans le document de référence de scripts d'Adobe Illustrator CS5 pour votre langage de script, que vous pouvez télécharger depuis le site http://www.adobe.com/devnet/illustrator/scripting/

Affichage du modèle d'objet

Les langages de script pris en charge fournissent chacun une fonction permettant d'afficher les objets de script définis par Illustrator avec des détails de référence.

Affichage du modèle d'objet JavaScript

Pour afficher le modèle d'objet JavaScript dans Illustrator, procédez comme suit :

1. Démarrez l'utilitaire d'écriture de scripts ESTK.

   Dans une installation Adobe par défaut, ESTK se trouve à l'emplacement suivant :

   - Windows :
     system drive\Program Files\Adobe\Adobe Utilities CS5\ExtendScript Toolkit CS5
   - Mac OS :
     system drive:Applications:Utilities:Adobe Utilities CS5:ExtendScript Toolkit CS5

2. Dans ESTK, choisissez la commande Aide > Outil de visualisation d'Object Model.

3. Dans la fenêtre Outil de visualisation d'Object Model, sélectionnez Adobe Illustrator CS5 Type Library dans la liste déroulante Navigateur.

Vous trouverez plusieurs exemples de scripts étendus dans le dossier /Scripting/Sample Scripts du répertoire d'installation d'Illustrator CS5.

Vous pouvez également consulter des exemples de scripts et des informations concernant les classes, les objets, les propriétés, les méthodes et les paramètres individuels dans le guide *Adobe Illustrator CS5 Scripting Reference: JavaScript* (Référence de script Adobe Illustrator CS5 : JavaScript) que vous pouvez télécharger depuis le site http://www.adobe.com/devnet/illustrator/scripting/.

Affichage du modèle d'objet AppleScript

Apple fournit l'application Editeur de scripts avec tous les systèmes Mac OS. Vous pouvez utiliser l'application Editeur de scripts pour consulter le dictionnaire AppleScript qui décrit les divers objets et commandes d'Illustrator.

Pour plus de détails sur l'utilisation de l'Editeur de scripts, reportez-vous à l'aide de l'Editeur de scripts.

1. Lancez l'application Editeur de scripts.

   REMARQUE : dans une installation Mac OS par défaut, l'application Editeur de scripts se trouve dans le dossier Applications:AppleScript:Script Editor. Si vous ne trouvez pas l'application Editeur de scripts, vous devez la réinstaller à partir du CD du système Mac OS.

2. Choisissez la commande Fichier > Ouvrir un dictionnaire. Une boîte de dialogue Ouvrir un dictionnaire s'affiche dans l'application Editeur de scripts.

3. Dans cette boîte de dialogue, recherchez l'application Adobe Illustrator CS5, puis sélectionnez-la et cliquez sur le bouton Ouvrir.

   La liste des objets et commandes Illustrator, qui inclut les propriétés et éléments associés à chacun des objets et les paramètres pour chaque commande, s'affiche dans l'application Editeur de scripts.

Vous trouverez plusieurs exemples de scripts étendus dans le dossier :Scripting:Sample Scripts du répertoire d'installation d'Illustrator CS5.

Vous pouvez également consulter des exemples de scripts et des informations concernant les classes, les objets, les propriétés, les méthodes et les paramètres individuels dans le guide *Adobe Illustrator CS5 Scripting Reference: AppleScript* (Référence de script Adobe Illustrator CS5 : JavaScript) que vous pouvez télécharger depuis le site http://www.adobe.com/devnet/illustrator/scripting/.

Affichage du modèle d'objet VBScript

VBScript fournit une bibliothèque type que vous pouvez utiliser pour consulter les propriétés et méthodes des objets Illustrator. Cette procédure explique comment consulter la bibliothèque type à partir de n'importe quel programme Microsoft Office. Votre éditeur VBScript possède certainement un accès à la bibliothèque. Pour plus d'informations, reportez-vous à l'aide de votre éditeur.

1. A partir de toute application Microsoft Office, choisissez la commande Outils > Macro > Visual Basic Editor.
2. Dans Visual Basic Editor, choisissez la commande Outils > Références.
3. Dans la boîte de dialogue qui s'affiche, cochez la case correspondant à la bibliothèque Adobe Illustrator CS5 Type Library, puis cliquez sur le bouton OK.
4. Choisissez la commande Affichage > Explorateur d'objets pour afficher la fenêtre Explorateur d'objets.
5. Choisissez l'option Illustrator dans la liste de bibliothèques ouvertes du menu déroulant situé en haut à gauche de la fenêtre Explorateur d'objets.

Vous trouverez plusieurs exemples de scripts étendus dans le dossier `/Scripting/Sample Scripts` du répertoire d'installation d'Illustrator CS5.

Vous pouvez également consulter des exemples de scripts et des informations concernant les classes, les objets, les propriétés, les méthodes et les paramètres individuels dans le guide *Adobe Illustrator CS5 Scripting Reference: VBScript* (Référence de script Adobe Illustrator CS5 : VBScript) que vous pouvez télécharger depuis le site http://www.adobe.com/devnet/illustrator/scripting/.

Exécution de scripts

L'interface Illustrator comprend un menu Scripts (Fichier > Scripts) qui fournit un accès facile et rapide à vos scripts.

Les scripts peuvent être directement répertoriés en tant qu'éléments de menu qui s'exécutent lorsque vous les sélectionnez. (Reportez-vous à la section « Installation de scripts dans le menu Scripts », page 10).

Vous pouvez également naviguer vers n'importe quel script de votre système de fichiers à partir du menu, puis exécuter le script. (Reportez-vous à la section « Exécution de scripts à partir de l'élément de menu Autre script », page 11).

Vous pouvez également choisir des scripts JavaScript avec une extension `.jsx` qui démarrent automatiquement lorsque vous lancez l'application. Pour plus de détails, reportez-vous à la section « Scripts de démarrage (scripts .jsx uniquement) », page 11.

Installation de scripts dans le menu Scripts

Pour inclure un script dans le menu Scripts (Fichier > Scripts), enregistrez-le dans le dossier Scripts qui se situe dans le dossier `/Illustrator CS5/Presets` de votre répertoire d'installation d'Illustrator CS5. Le nom de fichier du script, sans l'extension du fichier, apparaît dans le menu Scripts.

Les scripts que vous ajoutez au dossier Scripts lorsqu'Illustrator est en cours d'exécution apparaîtront dans le menu Scripts au prochain lancement d'Illustrator.

Vous pouvez installer un nombre illimité de scripts dans le menu Scripts. Si vous possédez un grand nombre de scripts, utilisez des sous-dossiers dans le dossier Scripts pour mieux organiser les scripts du menu Scripts. Chaque sous-dossier apparaît en tant que sous-menu séparé comportant les scripts de ce sous-dossier.

Exécution de scripts à partir de l'élément de menu Autre script

L'élément Autre script figurant au bas du menu Scripts (Fichier > Scripts > Autre script) vous permet d'exécuter des scripts qui ne sont pas installés dans le dossier Scripts.

Lorsque vous sélectionnez l'option Autre script, la boîte de dialogue Ouvrir s'affiche : utilisez-la pour accéder à un fichier de script. Lorsque vous sélectionnez le fichier, le script s'exécute.

Seuls les fichiers dont les types sont pris en charge apparaissent dans cette boîte de dialogue. Pour plus de détails, reportez-vous à la section « Prise en charge du langage de script dans Adobe Illustrator CS5 », page 7.

Scripts de démarrage (scripts .jsx uniquement)

Vous pouvez installer les scripts JavaScript possédant une extension de fichier `.jsx` dans un ou deux dossiers afin qu'ils s'exécutent automatiquement lorsque vous lancez Illustrator ou à chaque exécution de script. Ces dossiers sont les suivants :

- un dossier de scripts de démarrage spécifique à l'application, qui contient des scripts pour Illustrator CS5 ;
- un dossier de scripts de démarrage général, qui contient des scripts s'exécutant automatiquement lorsque vous lancez une application Creative Suite 5.

Dossier de scripts de démarrage spécifique à l'application

Vous devez placer les scripts de démarrage spécifiques à l'application dans un dossier appelé `Startup Scripts`, que vous créez dans le répertoire d'installation d'Illustrator.

Par exemple, lorsqu'Illustrator CS5 est installé dans son emplacement par défaut, vous créez le dossier `Startup Scripts` aux emplacements suivants :

- Windows : `C:\Program Files\Adobe\Adobe Illustrator CS5\Startup Scripts\`
- Mac OS : `/Applications/Adobe Illustrator CS5/Startup Scripts/`

Les scripts JavaScript qui possèdent une extension `.jsx` et qui sont placés dans le dossier `Startup Scripts` s'exécutent automatiquement lorsque :

- l'application est lancée ;
- tout fichier JavaScript est sélectionné à partir du menu Scripts (Fichier > Scripts).

Dossier de scripts de démarrage général

Le dossier de scripts de démarrage général comporte des scripts qui s'exécutent automatiquement lorsque vous lancez une application Creative Suite 5. Vous créez le dossier aux emplacements suivants :

- Windows : `Program Files/Common Files/Adobe/Startup Scripts CS5/Illustrator`
- Mac OS : `:Library:Application Support:Adobe:Startup Scripts CS5:Illustrator`

Si un script du dossier de démarrage général doit être exécuté uniquement par Illustrator, ce script doit inclure la directive ExtendScript `#target` (`#target illustrator`) ou le code suivant :

```
if( BridgeTalk.appName == "illustrator" ) {
    //continue executing script
}
```

Pour plus de détails, reportez-vous au *Guide des outils JavaScript*.

Changements survenus dans la version CS5

Cette section énumère les changements apportés au modèle d'objet de script pour prendre en charge les fonctions d'Illustrator CS5.

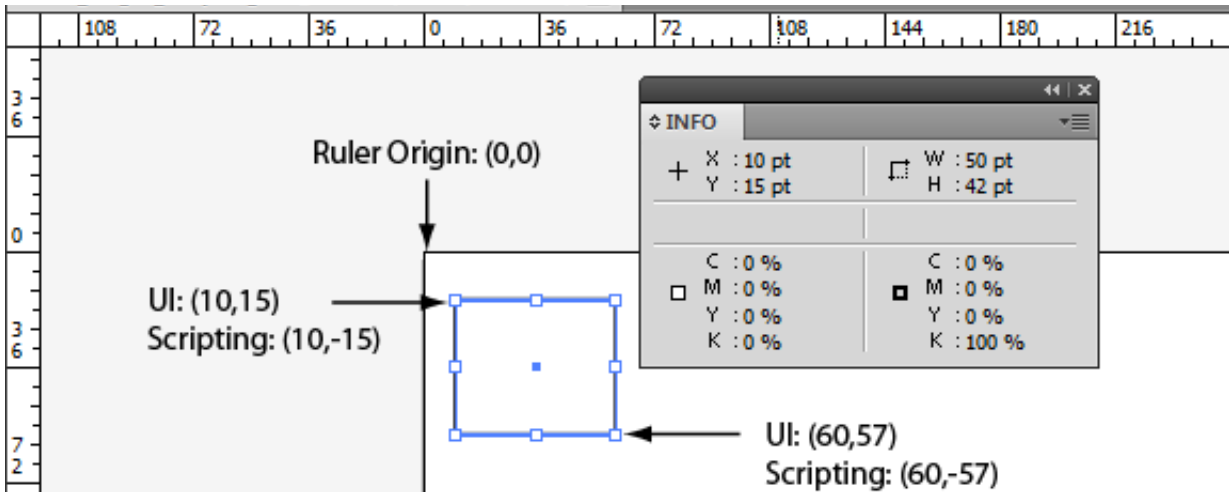
Origine de la règle

Dans les versions précédentes, l'*origine* du plan de travail, la position avec coordonnées (0,0), se trouvait dans l'angle inférieur gauche. Les incréments positifs X allaient de gauche à droite, tandis que les incréments positifs Y allaient de bas en haut.

Dans Illustrator CS5, l'origine du plan de travail est définie de façon à être dans l'angle supérieur gauche. Les incréments positifs X vont toujours de gauche à droite, mais les incréments positifs Y vont désormais de haut en bas.

Pour afficher les règles, choisissez **Affichage > Règles > Afficher les règles**. Utilisez le panneau Infos (**Fenêtres > Infos**) pour vérifier les coordonnées X/Y d'une position.

Pour éviter d'endommager des scripts existants, le script Illustrator utilise encore le système de coordonnées hérité. Autrement dit, tous les documents créés à l'aide de scripts utilisent le système de coordonnées hérité. Les documents que vous créez directement dans l'interface utilisateur utilisent le nouveau système de coordonnées. Si le script fonctionne sur un document de ce type, pour que son comportement soit le même que dans CS4, il doit convertir la valeur de la coordonnée Y d'une position en -Y, et inversement lorsqu'il interprète les valeurs des coordonnées du document.



Enumération

- Nouvelle énumération pour le système de coordonnées utilisé par Illustrator :
  - AppleScript : `artboard coordinate system/document coordinate system`
  - JavaScript : `CoordinateSystem`
  - VBScript : `AiCoordinateSystem`
- Nouvelle énumération pour les points d'enregistrement des symboles :
  - AppleScript — N/A
  - JavaScript : `SymbolRegistrationPoint`
  - VBScript : `AiSymbolRegistrationPoint`

- Nouvelle énumération pour le type de plan de la grille de perspective :
  - AppleScript : `floorplane/leftplane/noplane/rightplane`
  - JavaScript : `PerspectiveGridPlaneType`
  - VBScript : `AiPerspectiveGridPlaneType`
- Nouvelle énumération pour le type de lissage de texte :
  - AppleScript : `crisp/none/sharp/strong`
  - JavaScript : `TextAntiAlias`
  - VBScript : `AiTextAntialias`
- Nouvelle énumération pour l'orientation de l'impression des illustrations :
  - AppleScript : `auto rotate`
  - JavaScript : `PrintOrientation.AUTOROTATE`
  - VBScript : `AiAutoRotate`
- Nouvelle valeur d'énumération pour la version du format de fichier FXG :
  - AppleScript : `version 2.0`
  - JavaScript : `FXGVersion.VERSION2PT0`
  - VBScript : `AiVersion2Pt0`
- Nouvelle valeur d'énumération pour la règle de préservation du texte par le format de fichier FXG :
  - AppleScript : `autoConvert text`
  - JavaScript : `TextPreservePolicy.AUTOMATICALLYCONVERTTEXT`
  - VBScript : `AiAutomaticallyConvertText`

Classe

- Nouvelle propriété pour le système de coordonnées dans la classe `Application` :
  - AppleScript : `coordinate system`
  - JavaScript : `coordinateSystem`
  - VBScript : `CoordinateSystem`
- Nouvelles propriétés dans la classe `Artboard` pour définir l'origine de la règle, par rapport à (0,0) dans l'angle supérieur gauche, et pour le nom du plan de travail :
  - AppleScript

```
ruler origin of artboard 1 of document 1
name of artboard 1 of document 1
```

  - JavaScript

```
activeDocument.artboards[0].rulerOrigin
activeDocument.artboards[0].name
```

  - VBScript

```
ActiveDocument.Artboards[1].RulerOrigin
ActiveDocument.Artboards[1].Name
```

- Nouvelle méthode/commande dans la classe `Document` pour travailler avec différents systèmes de coordonnées :
  - AppleScript : `Convert coordinate`
  - JavaScript : `app.activeDocument.convertCoordinate()`
  - VBScript : `App.ActiveDocument.ConvertCoordinate()`
- Nouvelles méthodes/commandes dans la classe `Document` pour travailler avec plusieurs plans de travail :
  - AppleScript

```
select objects on active artboard
fit artboard to selected art
rearrange artboards
```

  - JavaScript

```
activeDocument.selectObjectsOnActiveArtboard()
activeDocument.fitArtboardToSelectedArt()
activeDocument.rearrangeArtboards()
```

  - VBScript

```
ActiveDocument.SelectObjectsOnActiveArtboard()
ActiveDocument.FitArtboardToSelectedArt()
ActiveDocument.RearrangeArtboards()
```

- Nouvelles méthodes/commandes dans la classe `Document` pour travailler avec des grilles de perspective :
  - AppleScript

```
import perspective grid preset
export perspective grid preset
select perspective preset
show perspective grid
hide perspective grid
get perspective active plane
set perspective active plane
```

  - JavaScript

```
app.activeDocument.importPerspectiveGridPreset()
app.activeDocument.exportPerspectiveGridPreset()
app.activeDocument.selectPerspectiveGrid()
app.activeDocument.showPerspectiveGrid()
app.activeDocument.hidePerspectiveGrid()
app.activeDocument.getPerspectiveActivePlane()
app.activeDocument.setPerspectiveActivePlane()
```

- VBScript

```
App.ActiveDocument.ImportPerspectiveGridPreset()
App.ActiveDocument.ExportPerspectiveGridPreset()
App.ActiveDocument.SelectPerspectiveGrid()
App.ActiveDocument.ShowPerspectiveGrid()
App.ActiveDocument.HidePerspectiveGrid()
App.ActiveDocument.GetPerspectiveActivePlane()
App.ActiveDocument.SetPerspectiveActivePlane()
```

- Nouvelle méthode dans la classe `PageItem` pour travailler avec des grilles de perspective :
  - AppleScript : `bring in perspective`
  - JavaScript : `bringInPerspective()`
  - VBScript : `BringInPerspective()`
- Nouvelle propriété dans la classe `PageItem` pour l'alignement des pixels :
  - AppleScript : `pixel aligned`
  - JavaScript : `pixelAligned`
  - VBScript : `PixelAligned`
- Nouvel argument de création facultatif pour la méthode `Add` de la collection `Symbols` qui permet de définir le point d'enregistrement des nouveaux symboles : `RegistrationPoint`.
- Nouvelle propriété dans la classe `TextFrameItem` pour le type de lissage de texte :
  - AppleScript : `antialias`
  - JavaScript : `antialias`
  - VBScript : `Antialias`

Classe de données

- La classe `PrintFLOptions` s'appelle désormais `PrintFlattenerOptions` en JavaScript et VBScript. En AppleScript, `flattening options` reste inchangé.
- Changements apportés aux propriétés de la classe `FXGSaveOptions` :
  - AppleScript

    Ajoutée : `blends policy`
    Supprimée : `clip content`
    Paramètres par défaut modifiés :
    `version` : `version 1.0` remplacée par `version 2.0`
    `text policy` : `keep text editable` remplacée par `autoConvert text`
    `gradients policy` : `keep gradients editable` remplacée par `autoConvert gradients`
  - JavaScript

    Ajoutée : `blendsPolicy`
    Supprimée : `clipContent`
    Paramètres par défaut modifiés :

`version` : `FXGVersion.VERSION1PT0` remplacée par `FXGVersion.VERSION2PT0`
`textPolicy` : `textPolicy.KEEPTEXTEDITABLE` remplacée par `AUTOMATICALLYCONVERTTEXT`
`gradientsPolicy` : `gradientsPolicy.KEEPGRADIENTSEDITABLE` remplacée par `AUTOMATICALLYCONVERTGRADIENTS`

▷ VBScript

Ajoutée : `BlendsPolicy`
Supprimée : `ClipContent`
Paramètres par défaut modifiés :
`Version` : `FXGVersion.VERSION1PT0` remplacée par `FXGVersion.VERSION2PT0`
`TextPolicy` : `aiKeepTextEditable` remplacée par `aiAutomaticallyConvertText`
`GradientsPolicy` : `aiKeepGradientsEditable` remplacée par `aiAutomaticallyConvertGradients`

▶ Nouvelles propriétés dans la classe de données `OpenOptions` pour travailler avec les plans de travail :

▷ AppleScript

```
convert crop area to artboard
convert tiles to artboard
create artboard with artwork bounding box
preserve legacy artboard
```

▷ JavaScript

```
convertCropAreaToArtboard
convertTilesToArtboard
createArtboardWithArtworkBoundingBox
preserveLegacyArtboard
```

▷ VBScript

```
ConvertCropAreaToArtboard
ConvertTilesToArtboard
CreateArtboardWithArtworkBoundingBox
PreserveLegacyArtboard
```

Problèmes connus

▶ Les scripts qui créent, enregistrent et ferment un grand nombre de fichiers Illustrator devraient régulièrement quitter Illustrator, puis le relancer. Le nombre maximal recommandé de fichiers à traiter avant de quitter Illustrator, puis de le relancer est de :

▷ Sous Windows 500 fichiers

▷ Sous Mac OS 1 000 fichiers

Pour plus d'informations sur la manière dont vous pouvez quitter et relancer Illustrator, reportez-vous aux sections « Lancement et activation d'Illustrator », page 26 et « Fermeture d'Illustrator », page 27.

▶ Le message d'avertissement « An Illustrator error occurred: 1346458189 ("PARM") » peut apparaître quand des scripts mal rédigés sont exécutés plusieurs fois dans Illustrator depuis ESTK.

Les auteurs de scripts doivent faire très attention à l'initialisation des variables ainsi qu'aux conflits engendrés par les espaces de noms lorsqu'ils lancent l'exécution répétée d'un lot de scripts Illustrator dans Illustrator par le biais d'ESTK, et ce dans le cadre d'une session Illustrator unique. Chaque script lancé est exécuté au sein du même moteur ExtendScript persistant dans Illustrator.

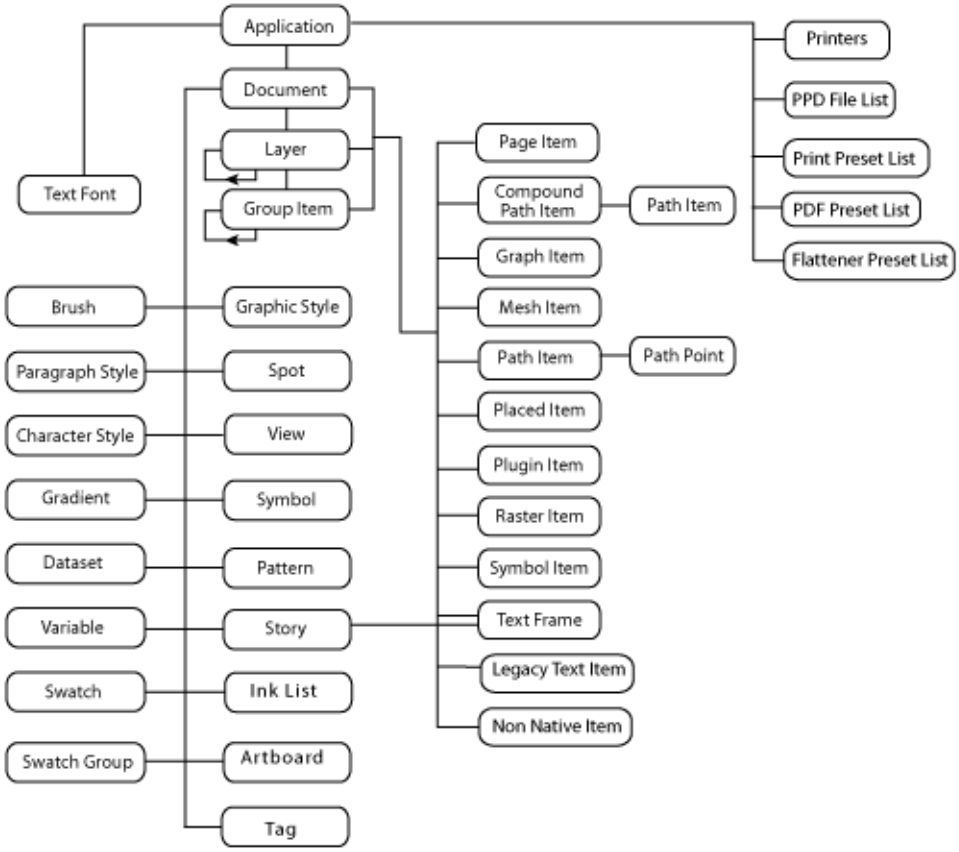
Le débogueur ESTK utilise BridgeTalk pour communiquer avec Illustrator. Un moteur ExtendScript persistant, global et unique au sein d'Illustrator gère l'ensemble des communications BridgeTalk. Grâce à Internet, le moteur ExtendScript s'ajoute à tous les scripts exécutés précédemment. Les problèmes de code de script pouvant être à l'origine de cet incident sont les suivants :

- ▷ lecture de variables non initialisées ;
- ▷ conflits d'espaces de noms globaux, par exemple lorsque deux ensembles de scripts différents s'opposent.

▶ Si vous créez plusieurs objets d'illustration dans AppleScript et affectez chacun d'eux à une variable, toutes les variables sont définies sur le dernier élément. Cela signifie que les éléments créés précédemment ne sont pas accessibles.

2 Modèle d'objet de script Illustrator

Bien comprendre le modèle d'objet Illustrator vous permettra d'améliorer vos capacités d'écriture de script. Le schéma suivant présente la hiérarchie de contenance du modèle d'objet, qui commence par l'objet `application`. Les classes `layer` et `group item` peuvent comporter des objets imbriqués susceptibles, eux aussi, de renfermer des objets imbriqués supplémentaires.



En plus de ces modèles d'objets spécifiques aux applications, JavaScript fournit certains objets utilitaires, tels que les objets `File` et `Folder`, qui vous permettent un accès au système de fichiers indépendant du système d'exploitation. Pour plus de détails, reportez-vous au *Guide des outils JavaScript*.

Conventions d'attribution de nom pour les objets

Il existe un modèle d'objet pour l'interface de script Illustrator, mais les noms d'objets réels varient légèrement selon les langages de script :

- Sous AppleScript, les noms figurent tous en minuscules et les mots individuels sont séparés par un espace, par exemple :

  ```
  graphic style
  ```

- Sous VBScript, les noms commencent par une majuscule et les autres mots contenus dans le nom se distinguent par une première lettre en majuscule, par exemple :

  ```
  GraphicStyle
  ```

- Sous JavaScript, le nom commence par une minuscule et les autres mots contenus dans le nom se distinguent par une première lettre en majuscule, par exemple :

  ```
  graphicStyle
  ```

Ce chapitre utilise des noms d'objets et de propriétés génériques, mais vous pouvez appliquer aisément ces conventions pour déterminer les noms correspondant à un langage spécifique.

Les noms de propriétés, noms de méthodes et objets cités dans ce document sont en police `monospaced`.

Objets de niveau supérieur (objets conteneurs)

Utilisez ces objets pour accéder à des informations globales sur l'application Illustrator ou à un document individuel.

Application

Les propriétés de l'objet `application` permettent à votre script d'accéder à des valeurs globales, telles que :

- la propriété `preferences` qu'un utilisateur définit de manière interactive dans l'application Illustrator à l'aide de la boîte de dialogue Préférences (Edition > Préférences) ;
- les informations système telles que les polices installées (propriété `text fonts`) et les imprimantes (propriété `printer list`).

Il existe aussi des propriétés qui fournissent des informations spécifiques aux applications et des informations de niveau supérieur relatives à tout document ouvert :

- les informations relatives aux applications, telles que le chemin d'installation, la version et le caractère visible d'Illustrator (propriétés `path`, `version` et `visible` respectivement) ;
- le document sélectionné (`current active`), c'est-à-dire la toile affichée et l'acceptation de la saisie de texte de l'utilisateur ;
- tous les documents ouverts (`documents`).

Les commandes ou méthodes de l'objet `application` permettent à votre script d'effectuer des actions au niveau des applications, par exemple :

- `Open` : ouvrir des fichiers
- `Undo` et `Redo` : annuler et rétablir des opérations
- `Quit` : fermer Illustrator

Document

L'objet `document` que vos scripts permettent de créer ou auquel ils peuvent accéder à partir de l'objet `application` représente une toile ou un fichier Illustrator chargé. Les propriétés de l'objet `document` vous permettent d'accéder au contenu du document, par exemple :

- la sélection en cours (`selection`) ou les objets graphiques sélectionnés par l'utilisateur dans le document ;
- tous les objets graphiques insérés, appelés `page items`, qui organisent la hiérarchie des illustrations ;
- les objets graphiques particuliers, tels que les symboles (`symbols`) et les blocs de texte (`text frames`) ;
- tous les calques (`layers`) et le calque actif (`active layer`).

Les propriétés du document vous renseignent également sur l'état du document en lui-même, par exemple :

- paramètres utilisateur associés au document, tels que `ruler units` ;
- enregistrement (`saved`) éventuel du document depuis la dernière modification du contenu ;
- chemin (`path`) du fichier associé.

Les méthodes de l'objet `document` permettent à vos scripts d'agir sur le document, par exemple :

- Enregister (`save`) dans un fichier Illustrator ou enregistrer sous (`save as`) l'un des formats de fichier pris en charge
- Activer (`activate`) ou fermer (`close`) un document
- Imprimer (`print`) le document. Vos scripts peuvent sélectionner une imprimante en référençant un objet `print options` ou bien référencer les imprimantes disponibles via la propriété `printer list` de l'objet d'application.

Calque

L'objet `layer` permet d'accéder au contenu, ou hiérarchie des illustrations, d'un calque spécifique. Vous pouvez accéder à l'objet `layer` via l'objet `document`. Les propriétés de l'objet `layer` permettent d'accéder à un calque ou d'obtenir des informations sur celui-ci, notamment :

- le caractère visible (`visible`) ou verrouillé (`locked`) du calque ;
- l'opacité (`opacity`) du calque (transparence globale) et sa position dans la pile (`z order position`) ;
- les préférences associées au calque en matière de création artistique, telles que `artwork knockout` et `blending mode`.

Hiérarchie des illustrations

Le contenu d'un document Illustrator est appelé *hiérarchie des illustrations*. L'illustration est représentée par les objets suivants :

- `compound path item`
- `graph item`
- `group item`
- `legacy text item`

- `mesh item`
- `non native item`
- `path item`
- `placed item`
- `plugin item`
- `raster item`
- `symbol item` (voir « Objets dynamiques », page 25).
- `text frame`

Vos scripts peuvent accéder à des objets graphiques et les manipuler via des collections à partir des objets `document` et `layer`. Il existe deux types de collections d'objets graphiques :

- Les objets de collection qui correspondent à chaque type d'objet d'illustration individuel, tel que l'objet `graph items` ou l'objet `mesh items`.
- L'objet `page items`, qui inclut tous les types d'objets graphiques.

Vous pouvez également utiliser l'objet `group item` pour référencer un ensemble groupé d'éléments d'illustration.

Vous pouvez créer des objets graphiques à l'aide de la commande `make` sous AppleScript ou de la méthode `add` de l'objet de collection d'un élément d'illustration. Par exemple, pour créer un objet `path item` :

AppleScript set myPathItem to make new path item in current document

JavaScript var myPathItem = activeDocument.pathItems.add();

VBScript Set myPathItem = appRef.ActiveDocument.PathItems.Add()

Les collections d'illustrations ci-dessous ne permettent pas la création d'objets par le biais de la commande `make` ou de la méthode `add` :

- Objet `graph items`
- Objet `mesh items`
- Objet `plugin items`
- Objet `legacy text items`

Pour plus d'informations sur la création d'objets de ce type, reportez-vous au document *Adobe Illustrator CS5 Scripting Reference* (Référence de script Adobe Illustrator CS5) correspondant au langage utilisé.

Styles d'illustration

Votre script peut appliquer un style graphique à une illustration à l'aide de l'objet `graphic style`. Pour appliquer un style graphique, utilisez la propriété `graphic styles` de l'objet `document` afin d'accéder à la méthode `apply to` de l'objet `graphic style`.

De la même manière, l'objet `brush` vous permet d'indiquer la forme à appliquer à l'illustration. Vous pouvez accéder à n'importe quelle forme à partir de l'objet de collection `brushes`, qui est une propriété de l'objet `document`.

Coloration d'objets

Votre script peut appliquer une couleur, un motif ou un dégradé à un objet `path item`, à l'aide des propriétés `fill color` ou `stroke color`.

- Les scripts peuvent définir de nouvelles nuances de couleur grâce à la commande `make` ou à la méthode `add` de l'objet `swatches`. Votre script permet également de créer des tons directs à l'aide de la commande `make` ou de la propriété `add` de l'objet `spots`.
- Vous pouvez définir les attributs d'un objet `ink` à l'aide de l'objet `ink info`, qui est une propriété de l'objet `ink`. Pour accéder aux objets `ink`, utilisez la propriété `ink list` de l'objet `document`.

Les objets ci-après vous permettent de créer des couleurs dans des espaces colorimétriques définis :

- l'objet `RGB color`, que vous pouvez définir à l'aide d'une plage de valeurs allant de 0 à 255 pour chacune des trois valeurs chromatiques ;
- l'objet `CMYK color`, que vous pouvez définir à l'aide de valeurs de pourcentage allant de 0 à 100 pour chacune des quatre valeurs chromatiques ;
- les objets `grayscale color` ou `LAB color`, que vous pouvez définir grâce à la même plage et au même nombre de valeurs que celles utilisées dans l'application Illustrator.

Objets texte

Lorsque vous saisissez du contenu dans un document Illustrator, le type devient automatiquement un objet `text frame` et, en même temps, un objet `story`.

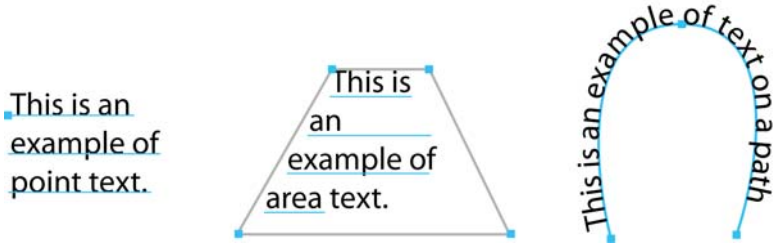
Pour observer cela, ouvrez un nouveau document dans Illustrator. Utilisez ensuite l'outil de texte horizontal pour saisir du texte, puis utilisez l'outil de texte vertical pour saisir du texte supplémentaire. Créez enfin un rectangle avant d'y saisir du texte. Exécutez maintenant le script JavaScript suivant :

```
var myDoc = app.activeDocument
alert("There are " + myDoc.textFrames.length + " text frames.")
alert("There are " + myDoc.stories.length + " stories.")
```

Blocs de texte

Il existe trois types de bloc de texte :

- point (texte de point)
- area (texte captif)
- path (texte curviligne)



Pour créer un type de bloc de texte spécifique, utilisez la propriété `kind` de l'objet `text frames` sous AppleScript. Les objets `text frames` sous JavaScript et VBScript contiennent des méthodes spécifiques de création de blocs de texte captifs et curvilignes.

Tout comme dans l'application Illustrator, vous pouvez lier des blocs de texte captifs ou curvilignes.

Pour lier des blocs de texte existants, utilisez la propriété `next frame` ou `previous frame` de l'objet `text frame`. Les blocs liés créent un objet `story` unique.

Pour plus de détails sur la création ou la liaison de blocs de texte, reportez-vous au chapitre de ce guide correspondant à votre langage de script.

Géométrie du texte

Même si les trois types de blocs de texte ont des caractéristiques communes, comme la propriété `orientation`, chacun possède des qualités qui lui sont propres, reflétées dans les propriétés de l'objet `text frame`. Par exemple :

- Un bloc de texte captif peut comporter des lignes et des colonnes auxquelles vous accédez par le biais des propriétés `row count` et `column count`.
- Le texte curviligne possède des propriétés `start T value` et `end T value` qui indiquent où le texte commence et se termine sur le tracé.
- Les blocs de texte captifs et curvilignes sont associés à un objet `text path`, spécifié à l'aide de la propriété `text path` de l'objet `text frame`. Le tracé de texte définit la position du bloc de texte et son orientation, horizontale ou verticale, sur le plan de travail (tandis que la propriété `orientation` de l'objet `text frame` définit l'orientation du texte à l'intérieur du bloc de texte).

  La propriété `text path` n'est pas valide pour le texte de point, car la position et l'orientation du texte de point sont entièrement définies par les propriétés du bloc de texte lui-même.

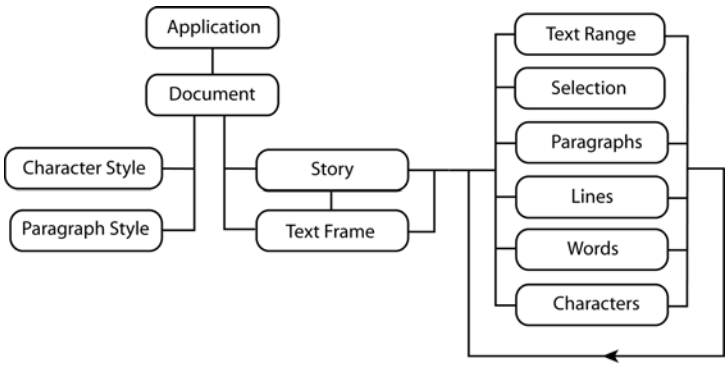
REMARQUE : un tracé de texte n'est pas le même qu'un élément d'illustration de tracé. Les tracés de texte sont associés aux éléments d'illustration de tracé ; vous pouvez y accéder et les manipuler pour modifier l'apparence du bloc de texte associé.

Objets représentant le contenu de texte

Il est possible d'accéder au contenu du texte réel d'un bloc de texte ou d'un article, de même qu'aux objets suivants :

- `characters`
- `words`
- `paragraphs`
- `lines`

Un objet `line` correspond à tous les caractères d'une ligne entière contenue dans un objet `text frame` ou `story`. Tous les éléments d'illustration de texte comportent au moins une ligne de texte définie en tant qu'objet `line`. Les illustrations de texte peuvent comporter plusieurs lignes de texte si le texte contient des retours forcés à la ligne ou si ces caractères passent à la ligne suivante, car ils ne s'adaptent pas à la largeur de l'illustration de texte. Les collections accèdent aux objets de texte et les identifient dans les objets `text frame` et `story` ; par exemple, `textFrame("My Text Frame").paragraphs` ou `story("My Story").paragraphs`.



Les objets `text frame` et `story` comportent les propriétés `insertion point` et `text selection`. Les propriétés de l'objet `text frame` incluent également les caractéristiques qui définissent le bloc de texte, telles que :

- les valeurs de largeur (`width`), hauteur (`height`) et position (`position`) du bloc ;
- le caractère masqué (`hidden`) ou verrouillé (`locked`) du cadre ;
- le caractère modifiable (`editable`) du texte.

REMARQUE : Il n'est pas possible de créer un objet `line` dans un script. Un script permet de créer des objets `character`, `paragraph` et `word`.

Plages de texte

Les différents objets de texte d'un bloc de texte ou d'un article sont également représentés collectivement par l'objet `text range`. Par exemple, un caractère correspond à une plage de texte d'une longueur de 1 tandis qu'un mot correspond à une plage de texte qui comporte un espace à l'avant.

Vous pouvez définir le contenu d'un objet `text range` en transmettant une chaîne à l'aide de la propriété `contents`.

Styles de texte

Les éléments de style de texte, tels que `font`, `capitalization` et `justification`, sont représentés par les objets `paragraph attribute` et `character attribute`. Ces objets d'attribut sont des propriétés des objets `paragraph style` et `character style`. Les objets `paragraph style` et `character style` sont associés à des méthodes `apply to` et `remove` qui permettent à votre script d'attribuer ou d'effacer les attributs d'une plage donnée de paragraphes, de texte ou de caractères.

Vous pouvez modifier l'affichage des propriétés d'une plage de texte en appliquant un style approprié ou en fournissant des priorités locales des attributs au niveau du texte ou du paragraphe :

- Les objets `character style` s'appliquent à des ensembles d'un ou de plusieurs caractères. Ils contrôlent des caractéristiques relatives aux caractères, telles que `font`, `alignment`, `leading`, `language` et `capitalization`, qui sont des propriétés de l'objet `character attribute`.
- Les objets `paragraph style` s'appliquent aux paragraphes. Ils contrôlent des caractéristiques relatives aux paragraphes, telles que `first line indent`, `left indent` et `right indent`, qui sont des propriétés de l'objet `paragraph attribute`.

Objets dynamiques

Vous pouvez réaliser des graphiques adaptés aux données en créant des objets dynamiques. Dans l'application Illustrator, utilisez le panneau Variables pour créer ou éditer des variables telles que les données d'un graphe, un fichier lié, une chaîne de caractères et la visibilité ou des variables dont le type n'est pas spécifié. Lors de l'écriture d'un script, l'objet `variable` vous permet de représenter ce type de variable. La propriété `kind` de l'objet `variable` indique le type de données dynamiques figurant dans un objet `variable`. Les objets `variable` sont des objets de niveau document : vous pouvez les créer dans un objet `document`.

REMARQUE : ne confondez pas objets `variable` et variables de script. Pour plus d'informations sur les variables Illustrator, les objets dynamiques et les graphiques adaptés aux données, reportez-vous à l'aide d'Illustrator.

Les ensembles de données, qui rassemblent des variables et leurs données dynamiques associées dans un même objet, sont représentés par l'objet `dataset` dans un script. L'objet `dataset` fournit des méthodes permettant de mettre à jour et de supprimer les objets `dataset` dans vos scripts.

Symboles

Dans Illustrator, les symboles sont des éléments d'illustration stockés dans le panneau Symboles. Vos scripts peuvent créer, supprimer et dupliquer les objets `symbol`. Lorsque vous créez des objets `symbol` dans votre script, Illustrator les ajoute au panneau Symboles relatif au document cible.

Un objet `symbol item` correspond à une instance d'un objet `symbol` dans un document. Chaque objet `symbol item` est lié à sa définition de symbole, de telle sorte que si cette définition est modifiée, toutes les instances du symbole sont mises à jour.

Votre script peut créer, supprimer et dupliquer les éléments de symbole. Les éléments de symbole sont des éléments d'illustration d'Illustrator et peuvent ainsi être traités de la même manière que les autres éléments d'illustration ou de page. Vous pouvez faire pivoter, redimensionner, sélectionner, verrouiller, masquer et effectuer d'autres opérations sur les éléments de symbole.

Transformations

L'objet `matrix` permet d'accéder à la puissance des matrices de transformation géométrique. Les matrices de transformation d'Illustrator stockent les paramètres d'une opération qui met à l'échelle, fait pivoter ou déplace (translate) un objet sur une page. L'utilisation des matrices présente certains avantages :

- En stockant des valeurs de transformation dans un objet `matrix`, vous pouvez réutiliser ces valeurs sur d'autres objets du script.
- Grâce à la concaténation de la rotation, la translation et/ou la mise à l'échelle des matrices et à l'application de la matrice obtenue, vous pouvez effectuer de nombreuses transformations géométriques à l'aide d'une instruction de script unique.
- Vous pouvez inverser les valeurs de la matrice.
- Vous pouvez comparer les valeurs de deux matrices.

L'objet `application` propose des commandes ou méthodes qui permettent de créer, d'obtenir, d'inverser, de comparer ou de concaténer des matrices.

La commande ou méthode utilisée pour appliquer une matrice s'intitule `transform` ; elle appartient à tout type d'objet sur lequel des transformations peuvent être effectuées.

3 Scripts Illustrator

Ce chapitre fournit une présentation sur l'utilisation des objets de script pour programmer Illustrator CS5. Des exemples spécifiques pour les langages de scripts pris en charge sont fournis dans les chapitres qui suivent.

Lancement et fermeture d'Illustrator à partir d'un script

Vos scripts peuvent vous permettre de lancer et de quitter Illustrator.

Lancement et activation d'Illustrator

AppleScript

Dans AppleScript, vous pouvez utiliser une instruction `tell` pour cibler Illustrator. La commande `activate` active Illustrator s'il n'est pas encore actif.

```
tell application "Adobe Illustrator"
    activate
end tell
```

JavaScript

Vous pouvez généralement exécuter des scripts JavaScript à partir du menu Scripts de l'application (Fichier > Scripts) ou du dossier de démarrage ; il n'est pas nécessaire de lancer Illustrator à partir de votre script.

Les informations sur le lancement d'Illustrator dans un script JavaScript ne figurent pas dans ce guide. Pour plus d'informations, faites une recherche sur la « messagerie interapplications » ou sur le « cadre de la messagerie JavaScript » dans le *Guide des outils JavaScript*.

VBScript

Il existe plusieurs façons de créer une instance d'Illustrator :

- `CreateObject` lance Illustrator en tant qu'application invisible si ce dernier n'est pas encore lancé. Si Illustrator est lancé en tant qu'application invisible, vous devez activer manuellement l'application pour la rendre visible :

  ```
  Set appRef = CreateObject("Illustrator.Application")
  ```

  Si plusieurs versions d'Illustrator sont installées sur la même machine et que vous utilisez la méthode `CreateObject` pour obtenir une référence d'application, l'utilisation de `« Illustrator.Application »` crée une référence à la dernière version d'Illustrator. Pour cibler spécifiquement une version précédente, indiquez l'identifiant de version à la fin de la chaîne :

  Pour Illustrator 10, utilisez `« Illustrator.Application.1 »`
  Pour Illustrator CS, utilisez `« Illustrator.Application.2 »`
  Pour Illustrator CS2, utilisez `« Illustrator.Application.3 »`
  Pour Illustrator CS3, utilisez `« Illustrator.Application.4 »`
  Pour Illustrator CS4, utilisez `« Illustrator.Application.CS4 »`
  Pour Illustrator CS5, utilisez `« Illustrator.Application.CS5 »`

- Utilisez l'opérateur `New` si vous avez ajouté une référence à la bibliothèque type Illustrator du projet. Par exemple, la ligne suivante crée une référence à l'objet `Application` :

```
Set appRef = New Illustrator.Application
```

Fermeture d'Illustrator

AppleScript

Utilisez la commande `quit` suivante :

```
tell application "Adobe Illustrator"
    quit
end tell
```

JavaScript

Utilisez la méthode `app.quit()` suivante :

```
app.quit()
```

VBScript

Utilisez la méthode `Quit` d'objet `Application` suivante :

```
Set appRef = CreateObject("Illustrator.Application")
appRef.Quit
```

Utilisation d'objets

Obtention du document ou du calque situé au premier plan

Pour faire référence au document sélectionné, utilisez la propriété `current document` de l'objet `application` dans AppleScript ou la propriété `active document` dans JavaScript ou VBScript. De la même manière, vous pouvez utiliser la propriété `current layer` ou `active layer` de l'objet `document` pour faire référence au calque sélectionné.

Il existe d'autres types de propriétés d'objet « actives » ou « sélectionnées », par exemple `active dataset` ou `active view`. Pour obtenir des détails, reportez-vous au document *Adobe Illustrator CS5 Scripting Reference* (Référence de script Adobe Illustrator CS5) correspondant à votre langage de script.

Création d'objets

Certains objets (outre l'objet `application` lui-même) ne peuvent être obtenus à partir de conteneurs ou d'objets parents. Votre script doit créer ces objets directement.

Les objets suivants doivent être créés spécifiquement :

```
CMYK color                  ink info                          print coordinate options
document preset             lab color                         printer
EPS save options            matrix                            printer info
export options AutoCAD      MXG save options                  print flattener options
export options Flash        no color                          print font options
export options GIF          open options                      print job options
export options JPEG         open options AutoCAD              print options
export options Photoshop    open options FreeHand             print page marks options
export options PNG8         open options PDF                  print paper options
export options PNG24        open options Photoshop            print postscript options
export options SVG          paper info                        raster effect options
file                        Pattern color                     rasterize options
folder                      PDF save options                  screen
gradient color              PPD file                          screen spot function
gray color                  PPD file info                     RGB color
Illustrator save options    print color management options     spot color
ink                         print color separation options    tracing options
```

Les objets `file` et `folder` sont des périphériques Adobe ExtendScript conçus pour fournir un accès indépendant des plates-formes au système de fichier sous-jacent. Pour plus de détails sur l'utilisation de ces objets, reportez-vous au *Guide des outils JavaScript*.

Pour plus d'informations sur la création explicite d'un objet, reportez-vous au chapitre correspondant à votre langage de script.

Objets de collection

Vous pouvez obtenir la plupart des objets de collection à partir d'un conteneur. Par exemple, un objet de collection `path items` peut figurer dans un objet `document` ou un objet `layer`. Pour obtenir un objet d'une collection `path items`, vous devez faire référence à un autre objet contenant. Pour obtenir un exemple, reportez-vous aux sections sur les langages ci-dessous.

AppleScript

Pour faire référence à un objet `path items` dans un document :

```
path item 1 in document 1
```

Pour faire référence à un objet `path items` dans un calque :

```
path item 1 in layer 1 in document 1
```

JavaScript

Pour faire référence à un objet `path items` dans un document :

```
documents[0].pathItems[1]
```

Pour faire référence à un objet `path items` dans un calque :

```
documents[0].layers[0].pathItems[0]
```

VBScript

Pour faire référence à un objet `path items` dans un document :

```
Documents(1).PathItems(1)
```

Pour faire référence à un objet `path items` dans un calque :

```
Documents(1).Layers(1).PathItems(1)
```

Pour obtenir plus d'exemples sur les conteneurs d'éléments de collection, reportez-vous au tableau Eléments de l'objet `document` dans le document *Adobe Illustrator CS5 Scripting Reference: AppleScript* (Référence de script Adobe Illustrator CS5 : AppleScript), ou le tableau Properties du document *Adobe Illustrator CS5 Scripting Reference: JavaScript* (Référence de script Adobe Illustrator CS5 : JavaScript) ou *Adobe Illustrator CS5 Scripting Reference: VBScript* (Référence de script Adobe Illustrator CS5 : VBScript). Un diagramme du modèle d'objet Illustrator CS5 est disponible à la section « Modèle d'objet de script Illustrator », page 18.

Objets sélectionnés

Vous souhaitez parfois écrire des scripts qui agissent sur les objets actuellement sélectionnés ou non. Par exemple, vous souhaitez appliquer une mise en forme donnée au texte sélectionné ou modifier la forme d'un tracé sélectionné.

Sélection de texte

Pour sélectionner du texte, utilisez la commande `select` ou la méthode de l'objet `text range`.

Sélection d'éléments d'illustration

Vous pouvez sélectionner un objet graphique (tel qu'un élément graphique, un élément de filet, un élément pixellisé, un élément de symbole, etc.) en définissant sa propriété `selected` sur `true`. (Dans AppleScript, `selected` est une propriété de l'objet `page items`.)

Référence à des éléments d'illustration sélectionnés

Pour faire référence à tous les objets actuellement sélectionnés d'un document, utilisez la propriété `selection` de l'objet `document`. Pour les objets du tableau de données de sélection, vous devez déterminer leur type, de manière à savoir quelles propriétés et méthodes ou commandes vous pouvez utiliser. Dans JavaScript et VBScript, chaque type d'objet d'illustration possède une propriété `typename` en lecture seule que vous pouvez utiliser pour déterminer le type d'objet. Dans AppleScript, utilisez la propriété `class`.

Remarques sur le changement de nom d'objets stockés dans les panneaux de l'application

Plusieurs objets peuvent être renommés, c'est-à-dire que vous pouvez modifier leur propriété `name`. Les types d'objets ci-après peuvent être classés par ordre alphabétique dans le panneau Illustrator correspondant. Si un script modifie le nom d'un tel objet, les références d'index à cet objet peuvent ne plus être valides.

```
Brush
Gradient
Graphic Style
Pattern
Swatch
Symbol
Variable
```

Unités de mesure

Illustrator utilise les points en tant qu'unité de mesure pour presque toutes les distances. 1 pouce correspond à 72 points. Il existe une exception pour les valeurs des propriétés telles que `kerning`, `tracking` et `aki` (utilisée pour les compositions de texte japonaises), pour lesquelles l'unité est l'« em ». (voir la section « Unités em », page 30).

Illustrator utilise des points lors de la communication avec vos scripts, *quelle que soit l'unité de graduation de la réglette*. Si votre script repose sur l'addition, la soustraction, la multiplication ou la division de valeurs de mesure spécifiques pour des unités autres que le point, il doit effectuer toutes les conversions d'unités nécessaires afin que vos mesures soient en points. Par exemple, si vous souhaitez utiliser des pouces pour vos coordonnées ou vos unités de mesure, vous devez multiplier toutes les valeurs en pouces par 72 lorsque vous saisissez les valeurs dans votre script.

Le tableau suivant présente les formules de conversion pour les différentes unités de mesure :

Unité	Formule de conversion
centimètres	28,346 points = 1 centimètre
pouces	72 points= 1 pouce
millimètres	2,834645 points = 1 millimètre
picas	12 points = 1 pica
Q	0,709 point = 1 Q (1 Q équivaut à 0,23 millimètre)

JavaScript fournit le type d'objet `UnitValue`, qui propose des utilitaires de conversion d'unités. Pour plus de détails, reportez-vous au *Guide des outils JavaScript*.

Unités em

Les valeurs qui utilisent l'unité em au lieu de points sont mesurées en millièmes d'unité em.

Une unité em est proportionnelle à la taille du corps de police actuel. Par exemple, avec une police de 6 points, 1 em équivaut à 6 points ; avec une police de 10 points, 1 em équivaut à 10 points. Avec une police de 10 points, une valeur de crénage de 20 unités em correspond à :

```
(20 units x 10 points) / 1000 units/em = 0.2 points
```

Positionnement et dimensions d'un élément de page

Illustrator utilise une géométrie simple à deux dimensions sous forme de points pour enregistrer la `position` des objets `page item` dans un document. Chaque objet `page item` d'un document a une propriété `position` qui définit un point fixe correspondant aux coordonnées de la page au format [x, y]. Le point fixe se trouve dans le coin supérieur gauche du cadre de sélection de l'objet.

Pour plus d'informations sur les types d'objets qui constituent la collection `page items`, reportez-vous à la section « Hiérarchie des illustrations », page 20.

Un point est désigné par des coordonnées :

- la position horizontale, *x* ;
- la position verticale, *y*.

Vous pouvez consulter ces coordonnées depuis le panneau Informations lorsque vous sélectionnez ou que vous créez un objet dans Illustrator.

Pour le plan de travail, l'origine des coordonnées par défaut, (0,0), figure dans l'angle supérieur gauche, qui se reflète dans la propriété `ruler origin` de l'objet `artboard`. Les valeurs X des coordonnées augmentent lorsque l'on progresse de la gauche vers la droite, tandis que les valeurs Y augmentent du haut vers le bas. Cette modification a été apportée dans la version CS5. Pour conserver la compatibilité entre les scripts, un document créé par un script utilise toujours l'ancien système, avec l'origine située en bas à gauche du plan de travail et la valeur Y augmentant à mesure que l'on progresse du bas vers le haut. La propriété `page origin` d'un objet `document` définit l'angle inférieur gauche de la zone imprimable du document en tant que point fixe.

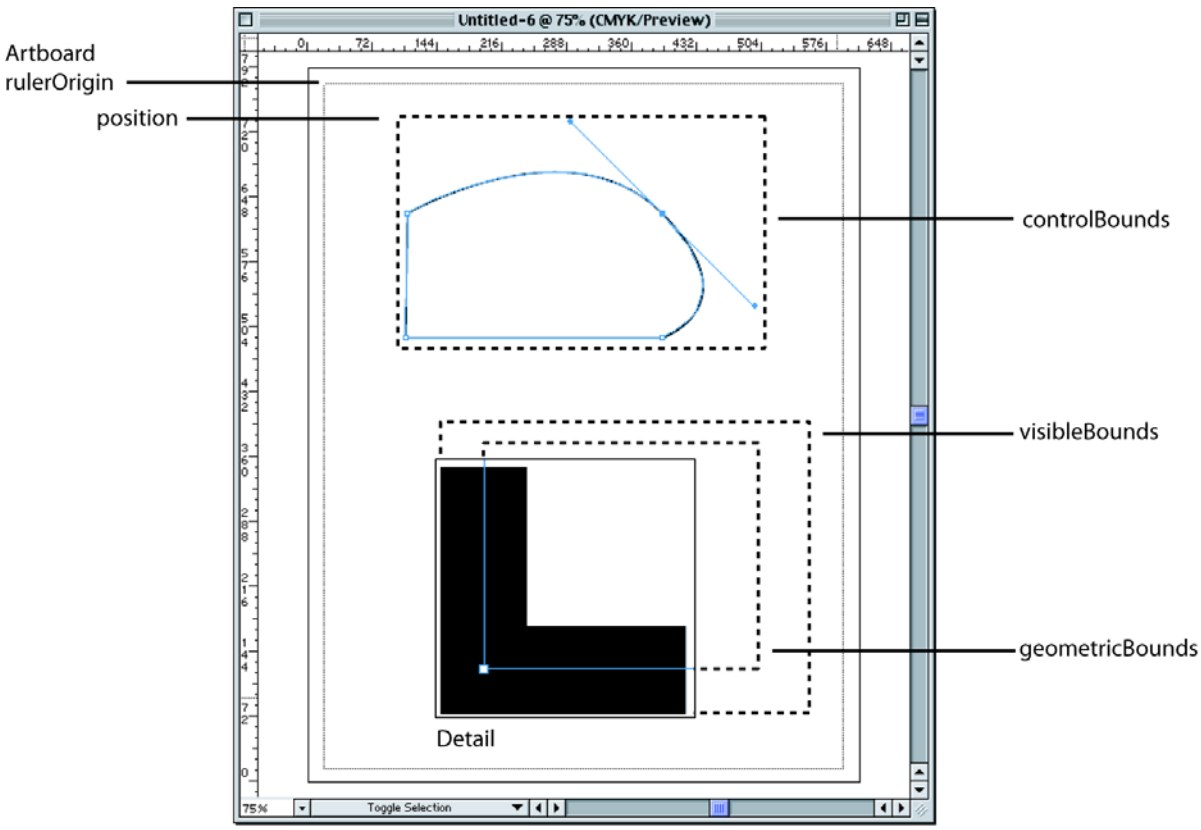
Chaque objet `page item` possède également des propriétés `width` et `height`. La valeur maximale autorisée pour la largeur ou la hauteur de l'élément de la page est de 16 348 points.

Limites des éléments d'illustration

Chaque objet `page item` possède trois propriétés qui utilisent les rectangles fixes pour décrire l'étendue globale de l'objet :

- La propriété `geometric bounds` d'un élément de page correspond aux dimensions rectangulaires du cadre de sélection de l'objet, à l'exclusion de l'épaisseur de contour.
- La propriété `visible bounds` d'un élément de page correspond aux dimensions de l'objet, y compris les épaisseurs de contour.
- La propriété `control bounds` correspond aux dimensions rectangulaires de l'objet, y compris les points de contrôle internes et externes.

L'image suivante illustre ces propriétés, à l'aide des conventions d'attribution de nom JavaScript.

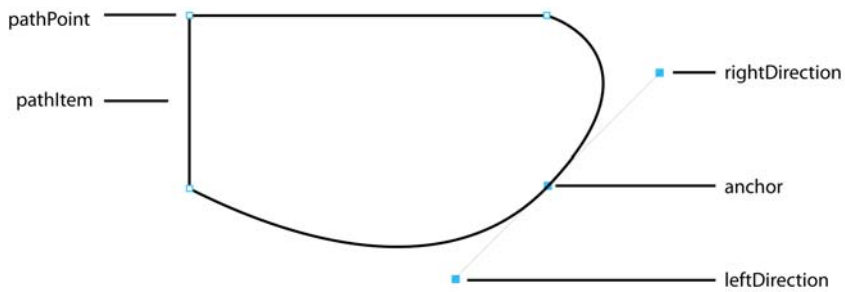


Tracés et formes

Les tracés sont représentés dans le modèle d'objets de document (DOM) Illustrator par l'objet `path item`. Les éléments de tracé comprennent toutes les illustrations qui contiennent des tracés, telles que les rectangles, les ellipses et les polygones, de même que les tracés libres.

Un tracé libre consiste en une série de points de tracé. Un point de tracé peut être défini de deux manières :

- en tant que série de coordonnées de page x et y ;
- en tant qu'objet `path point`, qui définit un point d'ancrage et deux points de direction ou poignées qui indiquent la courbe de segment du tracé.



Pour plus de détails, d'exemples et d'informations sur la création de formes, reportez-vous au chapitre correspondant à votre langage de script.

Niveaux d'interaction de l'utilisateur

Lorsque l'avis de l'utilisateur est requis, une boîte de dialogue apparaît généralement. C'est ce que l'on appelle l'*interaction de l'utilisateur*. Cette fonction très utile survient souvent en cas d'interaction directe avec une application. Cependant, lorsqu'un script interagit avec une application, la boîte de dialogue interrompt l'exécution du script jusqu'à sa fermeture. Cela peut constituer un problème sérieux dans le cas d'un environnement automatisé où personne ne se charge des boîtes de dialogue.

L'objet `application` comporte une propriété `user interaction level` qui vous permet de contrôler le niveau d'interaction autorisé pendant l'exécution du script. Vous pouvez supprimer l'interaction dans un environnement automatisé ou permettre quelques interactions lorsque les scripts sont utilisés d'une manière plus interactive.

AppleScript

Il est possible d'envoyer des commandes d'une machine à une autre à l'aide d'AppleScript afin d'autoriser d'autres interactions. Dans AppleScript, il existe quatre autres valeurs possibles pour la propriété `user interaction level` :

Valeur de la propriété	Résultat
`never interact`	Aucune interaction autorisée
`interact with self`	Interaction uniquement avec des scripts exécutés à partir du menu Scripts (Fichier > Scripts)
`interact with local`	Interaction uniquement avec des scripts exécutés sur la machine locale (y compris la machine elle-même)
`interact with all`	Interaction avec tous les scripts

Ces quatre valeurs vous permettent de contrôler l'interaction en fonction de la source des commandes de scripts. Par exemple, si l'application agit en tant que serveur pour les utilisateurs à distance, il serait difficile pour un utilisateur à distance de fermer une boîte de dialogue. En revanche, cela ne présente pas de problème pour quelqu'un qui se trouve devant la machine. Dans ce cas, un niveau d'interaction de type *interact with local* empêcherait les boîtes de dialogue d'interrompre les scripts à distance, mais permettrait de présenter les boîtes de dialogue associées aux scripts locaux.

JavaScript

Dans JavaScript, il existe deux valeurs possibles pour la propriété `app.userInteractionLevel` :

Valeur de la propriété	Résultat
`DISPLAYALERTS`	Interaction autorisée
`DONTDISPLAYALERTS`	Aucune interaction autorisée

VBScript

Dans VBScript, il existe deux valeurs possibles pour la propriété `UserInteractionLevel` de l'objet `Application` :

Valeur de la propriété	Résultat
`aiDisplayAlerts`	Interaction autorisée
`aiDontDisplayAlerts`	Aucune interaction autorisée

Impression de documents Illustrator

En utilisant la fonction de script `print options`, vous pouvez capturer et automatiser des parties de votre flux d'impression. Le script expose l'ensemble des possibilités d'impression d'Illustrator, dont certaines peuvent ne pas être accessibles via l'interface utilisateur de l'application.

Illustrator prend en charge une session d'impression au maximum à la fois, en raison de limitations de l'architecture d'impression actuelle.

La commande ou la méthode `print` de l'objet `document` prend en compte un seul paramètre en option, qui vous permet d'indiquer l'objet `print options`.

L'objet `print options` vous permet de définir les paramètres d'impression tels que PPD, les options PostScript, les options papier, les options de gestion de couleur, etc. L'objet `print options` comporte également une propriété `print preset` qui vous permet d'indiquer un paramètre prédéfini pour spécifier votre travail d'impression.

Lorsque vous définissez les propriétés d'un objet `print options`, vous pouvez prendre connaissance des imprimantes, PPD, paramètres prédéfinis d'impression et autres éléments disponibles grâce aux propriétés de la « liste » en lecture seule de l'objet `application`, comme les propriétés `printer list`, `PPD file list` et `print presets list`.

4 Création de scripts à l'aide d'AppleScript

Ce chapitre utilise des exemples de script, accompagnés d'explications pour vous aider à vous familiariser avec la création de scripts Illustrator à l'aide d'AppleScript.

Pour plus d'informations

Vous trouverez plusieurs exemples de scripts étendus dans le dossier `:Scripting:Sample Scripts` du répertoire d'installation d'Illustrator CS5.

Pour plus de détails sur les classes, objets, propriétés, commandes et paramètres individuels, ainsi que pour obtenir des exemples de scripts illustrant leur application, reportez-vous au guide *Adobe Illustrator CS5 Scripting Reference: AppleScript* (Référence de script Adobe Illustrator CS5 : AppleScript), disponible dans le dossier `:Scripting:Documentation` du répertoire d'installation d'Illustrator CS5. Vous pouvez également consulter le dictionnaire d'Illustrator CS5 depuis l'application Editeur de scripts (voir « Affichage du modèle d'objet AppleScript », page 9).

Si vous ne comprenez pas les concepts et les termes utilisés dans ce chapitre, reportez-vous au document *Introduction aux scripts Adobe*.

Votre premier script Illustrator

Le premier projet classique abordé lors de l'apprentissage d'un nouveau langage de programmation consiste à afficher le message « Hello World! ». Dans cet exemple, vous allez créer un document Illustrator, puis ajouter un bloc de texte contenant ce message. Procédez comme suit :

1. Ouvrez l'application Editeur de scripts.

   Dans une installation Mac OS par défaut, l'application Editeur de scripts se trouve dans le dossier `Applications:AppleScript:Script Editor`. Si vous ne trouvez pas l'application Editeur de scripts, vous devez la réinstaller à partir du CD du système Mac OS.

2. Entrez le script ci-dessous.

```
--Send the following commands to Illustrator
tell application "Adobe Illustrator"
--Create a new document
set docRef to make new document
--Create a new text frame with the string "Hello World"
set textRef to make new text frame in docRef ?
       with properties {contents: "Hello World!", position:{200, 200}}
end tell
```

3. Dans la barre d'outils de l'application Editeur de scripts, cliquez sur la commande Exécuter.

CONSEIL : pour ajouter le script au menu des scripts Illustrator (Fichier > Scripts), vous devez l'enregistrer dans le dossier Scripts. Le script s'affichera dans le menu au prochain démarrage d'Illustrator. Pour plus de détails, reportez-vous à la section « Installation de scripts dans le menu Scripts », page 10.

Ajout de caractéristiques à « Hello World »

Nous allons maintenant créer un script qui modifiera le document Illustrator créé avec votre premier script. Notre deuxième script indique comment :

- obtenir le document actif ;
- obtenir la largeur du document actif ;
- redimensionner le bloc de texte pour l'adapter à la largeur du document.

Si vous avez fermé le document Illustrator, exécutez votre premier script une nouvelle fois pour créer un document.

Procédez comme suit :

1. Dans l'application Editeur de scripts, choisissez la commande Fichier > Nouveau pour créer un script.
2. Entrez le code suivant :

```
tell application "Adobe Illustrator"
-- current document is always the active document
    set docRef to the current document
    set docWidth to the width of docRef
-- resize the text frame to match the page width
    set width of text frame 1 of docRef to docWidth
-- alternatively, one can reference the item directly, as follows:
    set width of text frame 1 of current document to docWidth
end tell
```

3. Exécutez le script.

Référencement d'objets

Dans AppleScript, Illustrator renvoie des références à l'objet par index ou par nom. Voici, par exemple, une référence au premier tracé sur le calque 2 :

```
path item 1 of layer 2 of document 1
```

La position d'index de l'objet peut changer à mesure que vous créez ou supprimez d'autres objets. Par exemple, lorsque vous créez un élément de tracé sur le calque `layer 2`, ce nouvel élément de tracé devient `path item 1 of layer 2 of document 1`. Le nouvel objet déplace l'élément de tracé initial en le forçant à la position 2. Ainsi, toute référence à `path item 1 of layer 2 of document 1` renvoie vers le nouvel objet. Cette méthode d'application de numéros d'index assure que le numéro le plus bas fait référence au dernier objet sur lequel vous avez travaillé.

Observez le script d'exemple suivant :

```
-- Make 2 new objects and try to select both
tell application "Adobe Illustrator"
    set newDocument to make new document
    set rectPath to make new rectangle in newDocument
    set starPath to make new star in newDocument
    set selection of newDocument to {rectPath, starPath}
end tell
```

Ce script ne sélectionne pas le rectangle et l'étoile, contrairement à ce qui était prévu. En fait, il ne sélectionne que l'étoile. Essayez d'exécuter le script avec la fenêtre Historique des événements ouverte, et observez les références renvoyées par Illustrator pour chacune des commandes `make` consécutives. (Choisissez l'option Hist. des événements au bas de la fenêtre de l'application Editeur de scripts.) Les deux commandes renvoient la même référence d'objet, à savoir : `path item 1 of layer 1 of document 1`. Par conséquent, la dernière ligne est résolue de la façon suivante :

```
set selection of document 1 to {path item 1 of layer 1 of document 1, ?
    path item 1 of layer 1 of document 1}
```

Une meilleure approche consiste à référencer les objets par leur nom :

```
tell application "Adobe Illustrator"
    set newDocument to make new document
    make new rectangle in newDocument with properties {name:"rectangle"}
    make new star in newDocument with properties {name:"star"}
    set selection of newDocument to ?
        {path item "rectangle" of newDocument, ?
         path item "star" of newDocument}
end tell
```

Cet exemple illustre la nécessité d'identifier de manière unique les objets dans les scripts AppleScript. Nous vous recommandons d'affecter des noms ou des variables aux objets auxquels vous devrez accéder ultérieurement, car rien ne garantit que vous accédez aux bons objets lorsque vous utilisez la méthode d'accès par index.

Obtention d'objets de documents et calques

Ce script fait référence à un objet en tant que partie d'un document :

```
-- Get reference for first page item of document 1
tell application "Adobe Illustrator"
    set pageItemRef to page item 1 of document 1
end tell
```

Dans le script suivant, la variable `pageItemRef` ne fait pas nécessairement référence au même objet que dans le script précédent, car ce script inclut une référence à un calque :

```
-- Get reference for first page item of layer 1 of document 1
tell application "Adobe Illustrator"
    set pageItemRef to page item 1 of layer 1 of document 1
end tell
```

Création d'objets

Pour créer un objet dans AppleScript, utilisez la commande `make`.

Utilisation de sélections

Lorsque l'utilisateur effectue une sélection dans un document, les objets sélectionnés sont stockés dans la propriété `selection` du document en question. Pour accéder à tous les objets sélectionnés dans le document actif :

```
tell application "Adobe Illustrator"
    set myDoc to current document
    set selectedObjects to selection of myDoc
end tell
```

En fonction des éléments sélectionnés, la valeur de la propriété `selection` peut être un tableau de tout type d'objets d'illustration. Pour obtenir ou manipuler les propriétés des éléments d'illustration sélectionnés, vous devez récupérer les éléments individuels dans le tableau de données. Pour connaître le type d'un objet, utilisez la propriété `class`.

L'exemple suivant permet d'obtenir le premier objet dans le tableau de données, puis d'afficher le type d'objet :

```
tell application "Adobe Illustrator"
    set myDoc to current document
    set selectedObjects to selection of myDoc
    set topObject to item 1 of selectedObjects
    display dialog (class of topObject)
end tell
```

Le premier objet d'un tableau de données de sélection correspond à l'objet sélectionné *ajouté* en dernier à la page, et non au dernier objet sélectionné.

Sélection d'objets d'illustration

Pour sélectionner un objet d'illustration, utilisez la propriété `selected` de l'objet.

Utilisation de blocs de texte

Pour créer un bloc de texte d'un type donné dans AppleScript, utilisez la propriété `kind` de l'objet `text frame`.

```
set myRect to make new rectangle in current document with properties ?
{position:{100, 700}, height:100, width:100}
set myAreaText to make new text frame in current document with properties ?
{kind:point text,contents:"Text Frame 1"}
```

Blocs liés

Tout comme dans l'application Illustrator, vous pouvez lier des blocs de texte captifs ou curvilignes.

Pour lier des blocs de texte existants, utilisez la propriété `next frame` ou `previous frame` de l'objet `text frame`.

Lorsque vous copiez le script suivant dans votre éditeur de scripts, placez la valeur de la propriété `contents` sur une seule ligne. Le caractère de ligne longue (¬) n'est pas valide dans une valeur de chaîne.

```
tell application "Adobe Illustrator"
    make new document
    make new rectangle in current document with properties ?
        {position:{100, 500}, height:100, width:100}
    make new text frame in current document with properties ?
        {kind:area text, text path:the result, name:"tf1", ?
        contents:"This is two text frames linked together as one story, with?
        text flowing from the first to the last. First frame content. "}
    make new rectangle in current document with properties ?
        {position:{300, 700}, height:100, width:100}
    make new text frame in current document with properties ?
    {kind:area text, text path:the result, name:"tf2", ?
        contents:"Second frame content." }
    --use the next frame property to thread the frames
        set next frame of text frame "tf1" of current document to ?
        text frame "tf2" of current document
    redraw
end tell
```

Création d'un objet d'article unique à l'aide de blocs liés

Les blocs liés créent un objet `story` unique. Pour observer cette opération, exécutez le script AppleScript suivant après l'exécution du script à la section « Blocs liés », page 37.

```
display dialog ("There are " & (count(text frames of current document)) & " text frames.")
display dialog("There are " & (count(stories of current document)) & " stories.")
```

Création de tracés et de formes

Cette section explique comment créer des éléments comportant des tracés.

Tracés

Pour créer un tracé linéaire ou libre, spécifiez une série de points de tracé sous la forme d'une série de coordonnées x-y ou d'objets `path point`.

Les coordonnées x-y limitent le tracé à des segments droits. Pour créer un tracé incurvé, vous devez créer des objets `path point`. Un tracé peut consister en une combinaison de coordonnées de page et d'objets `path point`.

Spécification d'une série de coordonnées x-y

Pour spécifier un tracé à l'aide de paires de coordonnées de page, utilisez la propriété `entire path` de l'objet `path items`. Le script suivant spécifie trois paires de coordonnées x, y permettant de créer un tracé comportant trois points :

```
tell application "Adobe Illustrator"
set docRef to make new document
-- set stroked to true so we can see the path
set lineRef to make new path item in docRef with properties {stroked:true}
set entire path of lineRef to {{220, 475},{200, 300},{375, 300}}
end tell
```

Utilisation d'objets PathPoint

Pour créer un objet `path point`, vous devez définir trois valeurs relatives au point :

- un point `anchor` fixe, représentant le point sur le tracé ;
- deux points directeurs, `left direction` et `right direction`, qui vous permettent de contrôler la courbure du segment de tracé.

Vous définissez chaque propriété en tant que tableau de données de coordonnées de page au format [x, y] :

- Si les trois propriétés d'un objet `path point` ont les mêmes coordonnées et que les propriétés de l'objet `path point` suivant sur la ligne sont égales les unes aux autres, vous obtenez un segment de ligne droite.
- Si plusieurs propriétés d'un objet `path point` présentent des valeurs différentes, le segment relié au point est incurvé.

Pour créer un tracé ou ajouter des points à un tracé existant à l'aide d'objets `path point`, créez un objet `path item`, puis ajoutez les points de tracé en tant qu'objets enfants dans l'objet `path item` :

```
tell application "Adobe Illustrator"
set docRef to make new document
-- set stroked to true so we can see the path
set lineRef to make new path item in docRef with properties {stroked:true}
    --giving the direction points the same value as the
    --anchor point creates a straight line segment
set newPoint to make new path point of lineRef with properties ?
    {anchor:{220, 475},left direction:{220, 475},right direction:{220, 475},
    point type:corner}

set newPoint2 to make new path point of lineRef with properties ?
    {anchor:{375, 300},left direction:{375, 300},right direction:{375, 300},
    point type:corner}

    --giving the direction points the different values
    --creates a curve
set newPoint3 to make new path point of lineRef with properties ?
    {anchor:{220, 300},left direction:{180, 260},right direction:{240, 320},
    point type:corner}

end tell
```

Combinaison de types de point de tracé

L'exemple de script suivant permet de créer un tracé avec trois points, en combinant la propriété du tracé entier à un objet `path point` :

```
tell application "Adobe Illustrator"
set docRef to make new document
-- set stroked to true so we can see the path
set lineRef to make new path item in docRef with properties {stroked:true}
set entire path of lineRef to {{220, 475},{375, 300}}
set newPoint to make new path point of lineRef with properties ?
    {anchor:{220, 300},left direction:{180, 260},right direction:{240, 320},
    point type:corner}
end tell
```

Formes

Pour créer une forme, vous devez utiliser l'objet correspondant au nom de la forme (par exemple, `ellipse`, `rectangle` ou `polygon`), puis utiliser les propriétés de l'objet pour indiquer la position de la forme, sa taille et d'autres informations, telles que le nombre de côtés dans le cas d'un polygone.

Rappel :

- Toutes les mesures, ainsi que les coordonnées de page sont traitées par le moteur de création de scripts. Pour plus de détails, reportez-vous à la section « Unités de mesure », page 30.
- Les coordonnés x et y sont mesurées à partir de l'angle inférieur gauche du document, comme indiqué dans le panneau Informations de l'application Illustrator. Pour plus de détails, reportez-vous à la section « Positionnement et dimensions d'un élément de page », page 30.

Accès non réinscriptible

Les propriétés des formes d'éléments de tracés utilisent l'état d'accès non réinscriptible, ce qui indique que vous ne pouvez écrire la propriété qu'au moment de la création de l'objet. Pour les objets d'élément de tracé, les propriétés, non modifiables, sont en lecture seule.

Création d'un rectangle

Observez l'exemple suivant :

```
tell application "Adobe Illustrator"
set docRef to make new document
set rectRef to make new rectangle in docRef with properties ?
    {bounds:{288, 360, 72, 144}}
end tell
```

L'exemple permet de créer un rectangle avec les propriétés suivantes :

- L'angle supérieur droit du rectangle est placé à 4 pouces (288 points) du bas de la page et à 5 pouces (360 points) du bord gauche de la page.
- L'angle inférieur gauche du rectangle est placé à 1 pouce (72 points) du bord gauche de la page et à 2 pouces (144 points) du bas de la page.

Création d'un polygone

Observez l'exemple suivant :

```
tell application "Adobe Illustrator"
set docRef to make new document
set pathRef to make new polygon in docRef with properties ?
{center point:{144, 288},sides:7,radius:72.0}
end tell
```

L'exemple permet de créer un polygone avec les propriétés suivantes :

- Le point central de l'objet est placé à 2 pouces (144 points) sur l'axe horizontal et à 4 pouces (288 points) sur l'axe vertical.
- Le polygone est composé de 7 côtés.
- La longueur du rayon du centre à chaque coin s'élève à 1 pouce (72 points).

Utilisation de la grille de perspective

La grille de perspective est une nouvelle fonctionnalité proposée par Illustrator CS5, qui vous permet de créer des illustrations et de les manipuler dans un environnement spatial à partir de lois de perspective établies. Pour activer la grille de perspective, ouvrez le menu Affichage et cliquez sur Grille de perspective, ou utilisez les outils de perspective disponibles dans la barre d'outils.

Le SDK fournit une interface de programmation qui permet d'utiliser la grille de perspective avec des programmes et qui est partiellement compatible avec vos scripts. Un script permet d'effectuer les opérations suivantes :

- définir les paramètres par défaut de la grille à partir de valeurs prédéfinies ;

- afficher ou masquer la grille ;
- définir le plan actif ;
- dessiner un objet en perspective sur le plan actif ;
- mettre un objet en perspective.

Utilisation des paramètres de perspective prédéfinis

Illustrator fournit des paramètres prédéfinis de grille par défaut, correspondant à des perspectives à un, deux ou trois points. Ces paramètres prédéfinis s'intitulent « [1P-NormalView] », « [2P-NormalView] » et « [3P-NormalView] ».

Le script suivant montre comment sélectionner le paramètre prédéfini de perspective à deux points par le biais d'un programme :

```
tell application "Adobe Illustrator"
    --Create a new document
    set docRef to make new document
    tell docRef
        --Select the default two-point perspective preset
        select perspective preset perspective preset "[2P-Normal View]"
    end tell
end tell
```

Vous pouvez créer de nouveaux paramètres prédéfinis de perspective, exporter ces paramètres prédéfinis vers un fichier ou les importer depuis un fichier. Les scripts suivants montrent comment exporter et importer des paramètres prédéfinis :

```
tell application "Adobe Illustrator"
    set docRef to make new document
    set filePath to "Macintosh HD:scripting:PGPresetsExported"
    export perspective grid preset of docRef to file filePath
end tell

tell application "Adobe Illustrator"
    set docRef to make new document
    set filePath to "Macintosh HD:scripting:PGPresets"
    import perspective grid preset of docRef from file filePath
end tell
```

Affichage ou masquage de la grille

Le script suivant permet d'afficher ou de masquer la grille de perspective par le biais d'un programme :

```
tell application "Adobe Illustrator"
    --Create a new document
    set docRef to make new document
    tell docRef
        --Display the perspective grid defined in the document
        show perspective grid
        --Hide the perspective grid defined in the document
        hide perspective grid
    end tell
end tell
```

Définition du plan actif

Les types de plan de la grille de perspective sont :

Plan de gauche	`perspective grid plane leftplane`
Plan de droite	`perspective grid plane rightplane`
Plan au sol	`perspective grid plane floorplane`
Plan incorrect	`perspective grid plane noplane`

Pour une grille de perspective à un point, seuls les plans de gauche et au sol sont valides.

Le script suivant permet de définir le plan de gauche comme plan de perspective actif :

```
tell application "Adobe Illustrator"
    --Create a new document
    set docRef to make new document
    tell docRef
        --Set the active plane to the left plane
        set perspective active plane perspective grid plane leftplane
    end tell
end tell
```

Réalisation d'un dessin sur une grille de perspective

Lorsque la grille de perspective est activée, les différents modes de dessin vous permettent de dessiner des objets ou de les manipuler en perspective. Le script suivant permet de créer un document, d'afficher une grille de perspective à deux points et de dessiner des objets graphiques sur le plan de gauche :

```
tell application "Adobe Illustrator"
    --Create a new document
    set docRef to make new document
    tell docRef
        --Select the default two-point perspective preset
        select perspective preset perspective preset "[2P-Normal View]"

        --Display the perspective grid defined in the document
        show perspective grid

        --Check if active plane is set to left, otherwise set it to left
        if (get perspective active plane) is not leftplane then
            set perspective active plane perspective grid plane leftplane
        end if

        --Draw rectangle in perspective, then resize to 200% and move
        set rectRef to make new rectangle with properties {bounds:{0, 0, 30, 30},
reversed:false}
        scale rectRef horizontal scale 200 vertical scale 200 about top left with
transforming objects
        translate rectRef delta x -420 delta y 480

        --Draw ellipse in perspective
        set ellipseRef to make new ellipse with properties {bounds:{60, -60, 90, -30},
reversed:false, inscribed:true}
```

```
        --Draw rounded rectangle in perspective
        set rrectRef to make new rounded rectangle with properties {bounds:{90, -90, 30,
30}, horizontal radius:10, vertical radius:10, reversed:false}

        --Draw polygon in perspective
        set polyRef to make new polygon with properties {center point:{105, 105},
radius:15, sides:7, reversed:false}

        --Draw star in perspective
        set starRef to make new star with properties {center point:{135, 135},
radius:15, inner radius:10, point count:6, reversed:false}

        --Draw path in perspective
        set newPath to make new path item with properties {entire path:{{anchor:{0, 0}},
{anchor:{60, 0}}, {anchor:{30, 45}}, {anchor:{90, 110}}}}
    end tell
end tell
```

Mise en perspective d'objets

Si un objet graphique n'est pas en perspective, utilisez la méthode `bringInPerspective()` pour le mettre en perspective et le placer sur un plan.

Le script suivant permet de créer un document, de dessiner un objet graphique et de le placer dans une grille de perspective à trois points :

```
tell application "Adobe Illustrator"
    --Create a new document
    set docRef to make new document
    tell docRef
        --Draw star
        set starRef to make new star with properties {center point:{135, 135},
radius:15, inner radius:10, point count:6, reversed:false}

        --Select the default three-point perspective preset
        select perspective preset perspective preset "[3P-Normal View]"

        --Display the perspective grid defined in the document
        show perspective grid

        --Check if active plane is set to left, otherwise set it to left
        if (get perspective active plane) is not leftplane then
            set perspective active plane perspective grid plane leftplane
        end if

        --Bring star to floor plane
        bring in perspective starRef position x 100 position y 100 perspective grid plane
floorplane
    end tell
end tell
```

5 Création de scripts à l'aide de JavaScript

Ce chapitre utilise des exemples de script, accompagnés d'explications pour vous aider à vous familiariser avec la création de scripts Illustrator à l'aide de JavaScript.

Pour plus d'informations

Vous trouverez plusieurs exemples de scripts étendus dans le dossier `/Scripting/Sample Scripts` du répertoire d'installation d'Illustrator CS5.

Pour plus de détails sur les classes, objets, propriétés, méthodes et paramètres individuels, ainsi que pour obtenir des exemples de scripts illustrant leur application, reportez-vous au guide *Adobe Illustrator CS5 Scripting Reference: JavaScript* (Référence de script Adobe Illustrator CS5 : JavaScript), disponible dans le dossier `/Scripting/Documentation` du répertoire d'installation d'Illustrator CS5. Vous pouvez également consulter le dictionnaire d'Illustrator depuis l'afficheur de modèle d'objet d'ESTK. Pour plus de détails sur l'afficheur de modèle d'objet d'ExtendScript Toolkit, reportez-vous à la section « Affichage du modèle d'objet JavaScript », page 9 ou consultez le *Guide des outils JavaScript*.

Si vous ne comprenez pas les concepts et les termes utilisés dans ce chapitre, reportez-vous au document *Introduction aux scripts Adobe*.

Votre premier script Illustrator

Le premier projet classique abordé lors de l'apprentissage d'un nouveau langage de programmation consiste à afficher le message « Hello World! ». Dans cet exemple, vous allez créer un document Illustrator, puis ajouter un bloc de texte contenant ce message. Procédez comme suit :

1. Utilisez n'importe quelle application de traitement de texte (comme Adobe InDesign® ou ESTK) pour saisir le texte ci-dessous :

```
//Hello World!
var myDocument = app.documents.add();
//Create a new text frame and assign it to the variable "myTextFrame"
var myTextFrame = myDocument.textFrames.add();
// Set the contents and position of the text frame
myTextFrame.position = [200,200];
myTextFrame.contents = "Hello World!"
```

   Pour plus de détails sur l'emplacement d'ExtendScript Toolkit, reportez-vous à la section « Affichage du modèle d'objet JavaScript », page 9.

2. Pour tester le script, utilisez l'une des méthodes suivantes :

- Si vous utilisez ESTK, vous devez sélectionner Adobe Illustrator CS5 dans la liste déroulante située dans l'angle supérieur gauche, cliquer sur le bouton Oui pour lancer Illustrator, puis choisir la commande Déboguer > Exécuter dans ESTK afin d'exécuter le script.
- Si vous utilisez un autre éditeur de texte, enregistrez le fichier en texte seul dans un dossier de votre choix avec l'extension `.jsx`, puis lancez Illustrator. Dans Illustrator, choisissez la commande Fichier > Scripts > Autre script, puis recherchez et exécutez votre fichier de script.

CONSEIL : pour ajouter le script au menu des scripts Illustrator (Fichier > Scripts), vous devez l'enregistrer dans le dossier Scripts. Le script s'affichera dans le menu au prochain démarrage d'Illustrator. Pour plus de détails, reportez-vous à la section « Installation de scripts dans le menu Scripts », page 10.

Ajout de caractéristiques à « Hello World »

Nous allons maintenant créer un script qui modifiera le document Illustrator créé avec votre premier script. Notre deuxième script indique comment :

- obtenir le document actif ;
- obtenir la largeur du document actif ;
- redimensionner le bloc de texte pour l'adapter à la largeur du document.

Si vous avez fermé le document Illustrator, réexécutez votre premier script pour créer un document avant de continuer cet exercice.

Procédez comme suit :

1. Choisissez la commande Fichier > Nouveau dans l'éditeur de texte pour créer un script.
2. Entrez le code suivant :

```
var docRef = app.activeDocument;

var docWidth = docRef.width

var frameRef = docRef.textFrames[0]

frameRef.width = docWidth
```

3. Exécutez le script.

Utilisation de méthodes dans JavaScript

Lorsque les méthodes que vous utilisez présentent plusieurs paramètres, vous pouvez ignorer les paramètres facultatifs en fin de liste, mais pas ceux qui apparaissent au milieu. Si vous ne souhaitez pas spécifier un paramètre apparaissant en milieu de liste, vous devez insérer la valeur `undefined` pour appliquer la valeur par défaut du paramètre. Par exemple, la définition suivante décrit la méthode `rotate()` pour un objet d'illustration.

```
rotate
  (angle
   [,changePositions]
   [,changeFillPatterns]
   [,changeFillGradients]
   [,changeStrokePattern]
   [,rotateAbout])
```

Dans la définition, extraite du document *Adobe Illustrator CS5 Scripting Reference: JavaScript* (Référence de script Adobe Illustrator CS5 : JavaScript), les paramètres facultatifs figurent entre crochets (`[]`).

Pour appliquer une rotation de 30 degrés à l'objet et modifier le paramètre `fillGradients`, vous devez utiliser l'instruction suivante :

```
myObject.rotate(30, undefined, undefined, true);
```

Vous devez spécifier la valeur `undefined` pour les paramètres `changePositions` et `changeFillPatterns`. Vous n'avez rien à spécifier pour les deux paramètres facultatifs qui suivent le paramètre `changeFillGradients`, puisqu'ils sont à la fin de la liste des paramètres.

Accès et référencement d'objets

Lors de l'écriture d'un script, vous devez d'abord indiquer le fichier, ou `document`, sur lequel le script doit agir. A l'aide de l'objet `application`, le script peut créer un document, ouvrir un document existant ou agir sur un document déjà ouvert.

Le script peut créer des objets dans le document, effectuer des opérations sur des objets sélectionnés par l'utilisateur ou sur les objets d'une collection. Les sections suivantes illustrent plusieurs techniques d'accès, de référencement et de manipulation d'objets Illustrator.

Référencement de l'objet application

Pour obtenir une référence à un objet spécifique, vous devez explorer la hiérarchie du contenu. Tous les scripts JavaScript s'exécutent de l'intérieur de l'application Illustrator ; toutefois, aucune référence spécifique à l'objet `application` n'est nécessaire. Par exemple, pour affecter le document actif dans Illustrator à la variable `frontMostDocument`, vous pouvez référencer la propriété `activeDocument` de l'objet `application`, comme suit :

```
var frontMostDocument = activeDocument;
```

Il est possible d'utiliser l'objet `application` dans une référence. Pour référencer cet objet `application`, utilisez la variable globale `app`. Les deux instructions qui suivent semblent identiques au moteur JavaScript :

```
var frontMostDocument = activeDocument;

var frontMostDocument = app.activeDocument;
```

Accès aux objets de collection

Tous les documents ouverts, ainsi que les objets qu'ils contiennent, sont rassemblés en objets de collection pour le type d'objet. Un objet de collection contient un tableau de données des objets auxquels vous pouvez accéder par index ou par nom. L'objet de collection prend la forme plurielle du nom de l'objet. Par exemple, l'objet de collection de l'objet `document` devient `documents`.

Le script d'exemple suivant permet d'obtenir tous les objets `graphic style` de la collection `graphic styles`, c'est-à-dire tous les styles graphiques disponibles pour le document actif :

```
var myStyles = app.activeDocument.graphicStyles;
```

Toutes les références numériques dans une collection commencent par zéro en JavaScript, c'est-à-dire que le premier objet dans la collection prend l'index [0].

En JavaScript, les numéros d'index ne changent pas lors de l'ajout d'un objet à une collection. Il existe toutefois une exception : l'objet `documents[0]` correspond toujours au document actif ou de premier plan.

Pour accéder au premier style d'une collection `graphic styles`, vous pouvez utiliser la variable déclarée dans le script d'exemple précédent ou bien utiliser la hiérarchie du contenu pour faire référence à la collection :

- Utilisation de la variable `myStyles` :

```
var firstStyle = myStyles[0];
```

- Utilisation de la hiérarchie du contenu :

```
var firstStyle = app.activeDocument.graphicStyles[0];
```

Les instructions suivantes affectent à une variable le nom du premier style graphique de la collection. Ces instructions sont interchangeables.

```
var styleName = myStyles[0].name

var styleName = firstStyle.name

var styleName = app.activeDocument.graphicStyles[0].name
```

Pour obtenir le nombre total d'objets d'une collection, utilisez la propriété `length` :

```
alert ( myStyles.length );
```

L'index du dernier style graphique de la collection est `myStyles.length-1` (-1, car le comptage de l'index commence à 0, et la propriété `length` à 1) :

```
var lastStyle = myStyles[ myStyles.length - 1 ];
```

Une expression représentant la valeur de l'index figure entre crochets (`[]`) et entre guillemets.

Si vous connaissez le nom d'un objet, vous pouvez y accéder dans les collections en utilisant le nom encadré de crochets, par exemple :

```
var getStyle = myStyles[?Ice Type?];
```

Chaque élément de la collection est un objet du type recherché et vous pouvez accéder à ses propriétés dans la collection. Par exemple, pour rechercher un nom d'objet, utilisez la propriété `name` :

```
var styleName = app.activeDocument.graphicStyles[0].name;
```

Pour appliquer le style `lastStyle` au premier élément `pageItem` du document, utilisez sa méthode `applyTo()` :

```
lastStyle.applyTo( app.activeDocument.pageItems[0] );
```

Création d'objets

Pour créer des objets, vous pouvez utiliser un script. Pour créer des objets disponibles à partir d'objets de collection ou de *conteneurs*, vous devez utiliser la méthode `add()` de l'objet de conteneur :

```
var myDoc = app.documents.add()
var myLayer = myDoc.layers.add()
```

Certains types d'objet ne sont pas disponibles dans les conteneurs. Vous pouvez créer un objet de ce type en définissant une variable, puis en appliquant l'opérateur `new` avec un constructeur d'objet pour affecter un objet en tant que valeur. Par exemple, pour créer un objet `CMYKColor` à l'aide du nom de variable `myColor` :

```
var myColor = new CMYKColor()
```

Utilisation de sélections

Lorsqu'un utilisateur effectue une sélection dans un document, les objets sélectionnés sont stockés dans la propriété `selection` du document. Pour accéder à tous les objets sélectionnés dans le document actif :

```
var selectedObjects = app.activeDocument.selection;
```

En fonction des types d'éléments sélectionnés, la valeur de la propriété `selection` peut être un tableau de données de tout type d'objets d'illustration. Pour obtenir ou manipuler les propriétés des éléments d'illustration sélectionnés, vous devez récupérer les éléments individuels dans le tableau de données. Pour rechercher un type d'objet, utilisez la propriété `typename`.

L'exemple suivant permet d'obtenir le premier objet dans le tableau de données, puis d'afficher le type d'objet :

```
var topObject = app.activeDocument.selection[0];
alert(topObject.typename)
```

Le premier objet d'un tableau de données de sélection correspond à l'objet sélectionné *ajouté* en dernier à la page, et non au dernier objet sélectionné.

Sélection d'objets d'illustration

Pour sélectionner un objet d'illustration, utilisez la propriété `selected` de l'objet.

Utilisation de blocs de texte

Pour créer un bloc de texte d'un type donné en JavaScript, vous devez utiliser la méthode `textFrames` dont le nom correspond au type du bloc de texte, par exemple :

```
var rectRef = docRef.pathItems.rectangle(700, 50, 100, 100);
//use the areaText method to create the text frame
var areaTextRef = docRef.textFrames.areaText(rectRef);
```

Blocs liés

Tout comme dans l'application Illustrator, vous pouvez lier des blocs de texte captifs ou curvilignes.

Pour lier des blocs de texte existants, utilisez la propriété `nextFrame` ou `previousFrame` de l'objet `text frame`.

Lors de la copie du script suivant dans ESTK, placez la valeur de la propriété `contents` sur une seule ligne.

```
var myDoc = documents.add();
var myPathItem1 = myDoc.pathItems.rectangle(244, 64, 82, 76);
var myTextFrame1 = myDoc.textFrames.areaText(myPathItem1);
var myPathItem2 = myDoc.pathItems.rectangle(144, 144, 42, 116);
var myTextFrame2 = myDoc.textFrames.areaText(myPathItem2);

// use the nextFrame property to thread the text frames
myTextFrame1.nextFrame = myTextFrame2;
var sText = "This is two text frames linked together as one story, with text
flowing from the first to the last. This is two text frames linked together as one
story, with text flowing from the first to the last. This is two text frames linked
together as one story. ";
myTextFrame1.contents = sText;
redraw();
```

Création d'objet story unique à l'aide de blocs liés

Les blocs liés créent un objet `story` unique. Pour observer cette opération, exécutez le script JavaScript suivant après l'exécution du script à la section « Blocs liés », page 48.

```
var myDoc = app.activeDocument
alert("There are " + myDoc.textFrames.length + " text frames.")
alert("There are " + myDoc.stories.length + " stories.")
```

Création de tracés et de formes

Cette section explique comment créer des éléments comportant des tracés.

Tracés

Pour créer un tracé libre, vous devez spécifier une série de points de tracé sous la forme d'une série de coordonnées x, y ou sous la forme d'objets `pathPoint`.

Les coordonnées x, y limitent le tracé à des segments droits. Pour créer un tracé incurvé, vous devez créer des objets `pathPoint`. Votre tracé consiste en une combinaison de coordonnées de page et d'objets `pathPoint`.

Spécification d'une série de coordonnées x, y

Pour spécifier un tracé à l'aide de paires de coordonnées de page, vous devez utiliser la méthode `setEntirePath()` de l'objet `pathItems`. Le script suivant spécifie trois paires de coordonnées x, y permettant de créer un tracé comportant trois points :

```
var myDoc = app.activeDocument;
var myLine = myDoc.pathItems.add();
//set stroked to true so we can see the path
myLine.stroked = true;
myLine.setEntirePath([[220, 475], [375, 300], [200, 300]]);
```

Utilisation d'objets pathPoint

Pour créer un objet `pathPoint`, vous devez définir trois valeurs relatives au point :

- un point `anchor` fixe, représentant le point sur le tracé ;
- deux points directeurs, `left direction` et `right direction`, qui vous permettent de contrôler la courbure du segment de tracé.

Vous définissez chaque propriété en tant que tableau de données de coordonnées de page au format [x, y].

- Si les trois propriétés d'un objet `pathPoint` présentent des coordonnées identiques et que les propriétés de l'objet `pathPoint` suivant sur la ligne sont égales entre elles, vous obtenez un segment de ligne droite.
- Si plusieurs propriétés d'un objet `pathPoint` présentent des valeurs différentes, le segment relié au point est incurvé.

Pour créer un tracé ou ajouter des points à un tracé existant à l'aide d'objets `pathPoint`, créez un objet `pathItem`, puis ajoutez les points de tracé en tant qu'objets enfants dans l'objet `pathItem` :

```
var myDoc = app.activeDocument;
var myLine = myDoc.pathItems.add();
    //set stroked to true so we can see the path
    myLine.stroked = true;
```

```
var newPoint = myLine.pathPoints.add();
    newPoint.anchor = [220, 475];
    //giving the direction points the same value as the
    //anchor point creates a straight line segment
    newPoint.leftDirection = newPoint.anchor;
    newPoint.rightDirection = newPoint.anchor;
    newPoint.pointType = PointType.CORNER;

var newPoint1 = myLine.pathPoints.add();
    newPoint1.anchor = [375, 300];
    newPoint1.leftDirection = newPoint1.anchor;
    newPoint1.rightDirection = newPoint1.anchor;
    newPoint1.pointType = PointType.CORNER;

var newPoint2 = myLine.pathPoints.add();
    newPoint2.anchor = [220, 300];
    //giving the direction points different values
    //than the anchor point creates a curve
    newPoint2.leftDirection =[180, 260];
    newPoint2.rightDirection = [240, 320];
    newPoint2.pointType = PointType.CORNER;
```

Combinaison de types de point de tracé

L'exemple suivant permet de créer un tracé avec trois points :

```
var myDoc = app.activeDocument;
var myLine = myDoc.pathItems.add();
    myLine.stroked = true;
    myLine.setEntirePath( [[220, 475], [375, 300]]);

// Append another point to the line
var newPoint = myDoc.myLine.pathPoints.add();
    newPoint.anchor = [220, 300];
    newPoint.leftDirection = newPoint.anchor;
    newPoint.rightDirection = newPoint.anchor;
    newPoint.pointType = PointType.CORNER;
```

Formes

Pour créer une forme, vous devez utiliser la méthode `pathItems` correspondant au nom de la forme (par exemple, `ellipse`, `rectangle` ou `polygon`), puis utiliser les paramètres pour indiquer la position de la forme, la taille et d'autres informations, telles que le nombre de côtés dans un polygone.

Rappel :

- Toutes les mesures, ainsi que les coordonnées de page sont traitées comme des points par le moteur de création de scripts. Pour plus de détails, reportez-vous à la section « Unités de mesure », page 30.
- Les coordonnés x et y sont mesurées à partir de l'angle inférieur gauche du document, comme indiqué dans le panneau Informations de l'application Illustrator. Pour plus de détails, reportez-vous à la section « Positionnement et dimensions d'un élément de page », page 30.

Création d'un rectangle

Observez l'exemple suivant :

```
var myDocument = app.documents.add()
var artLayer = myDocument.layers.add()
var rect = artLayer.pathItems.rectangle( 144, 144, 72, 216 );
```

L'exemple utilise la méthode `rectangle()` de l'objet `pathItems` pour créer un rectangle avec les propriétés suivantes :

- L'angle supérieur du rectangle est à 2 pouces (144 points) du bord inférieur de la page.
- Le bord gauche est à 2 pouces (144 points) du bord gauche de la page.
- Le rectangle mesure 1 pouce (72 points) de large sur 3 pouces (216 points) de long.

Création d'un polygone

Observez l'exemple suivant :

```
var myDocument = app.documents.add()
var artLayer = myDocument.layers.add()
var poly = artLayer.pathItems.polygon( 144, 288, 72.0, 7 );
```

L'exemple utilise la méthode `polygon()` pour créer un polygone avec les propriétés suivantes :

- Le point central de l'objet est placé à 2 pouces (144 points) sur l'axe horizontal et à 4 pouces (288 points) sur l'axe vertical.
- La longueur du rayon du centre à chaque coin s'élève à 1 pouce (72 points).
- Le polygone est composé de 7 côtés.

Utilisation de la grille de perspective

La grille de perspective est une nouvelle fonctionnalité proposée par Illustrator CS5, qui vous permet de créer des illustrations et de les manipuler dans un environnement spatial à partir de lois de perspective établies. Pour activer la grille de perspective, ouvrez le menu Affichage et cliquez sur Grille de perspective, ou utilisez les outils de perspective disponibles dans la barre d'outils.

Le SDK fournit une interface de programmation qui permet d'utiliser la grille de perspective avec des programmes et qui est partiellement compatible avec vos scripts. Un script permet d'effectuer les opérations suivantes :

- définir les paramètres par défaut de la grille à partir de valeurs prédéfinies ;
- afficher ou masquer la grille ;
- définir le plan actif ;
- dessiner un objet en perspective sur le plan actif ;
- mettre un objet en perspective.

Utilisation des paramètres de perspective prédéfinis

Illustrator fournit des paramètres prédéfinis de grille par défaut, correspondant à des perspectives à un, deux ou trois points. Ces paramètres prédéfinis s'intitulent « [1P-NormalView] », « [2P-NormalView] » et « [3P-NormalView] ».

Le script suivant montre comment sélectionner un paramètre prédéfini par le biais d'un programme :

```
//Set the default one-point perspective preset
app.activeDocument.selectPerspectivePreset("[1P-Normal View]");

//Set the default two-point perspective preset
app.activeDocument.selectPerspectivePreset("[2P-Normal View]");

//Set the default three-point perspective preset
app.activeDocument.selectPerspectivePreset("[3P-Normal View]");
```

Vous pouvez créer de nouveaux paramètres prédéfinis de perspective, exporter ces paramètres prédéfinis vers un fichier ou les importer depuis un fichier. Les scripts suivants montrent comment exporter et importer des paramètres prédéfinis :

```
//Create a new document
var mydoc = app.documents.add();
//Export perspective presets to a file
var exportPresetFile = new File("C:/scripting/PGPresetsExported")
mydoc.exportPerspectiveGridPreset(exportPresetFile);

//Create a new document
var mydoc = app.documents.add();
//Import perspective presets from a file
var importPresetFile = new File("C:/scripting/PGPresets")
mydoc.importPerspectiveGridPreset(importPresetFile);
```

Affichage ou masquage de la grille

Le script suivant permet d'afficher ou de masquer la grille de perspective par le biais d'un programme :

```
//Show the Perspective Grid defined in the document
app.activeDocument.showPerspectiveGrid();

//Hide the Perspective Grid defined in the document
mydoc.hidePerspectiveGrid();
```

Définition du plan actif

Les types de plan de la grille de perspective sont :

Plan de gauche	`PerspectiveGridPlaneType.LEFTPLANE`
Plan de droite	`PerspectiveGridPlaneType.RIGHTPLANE`
Plan au sol	`PerspectiveGridPlaneType.FLOORPLANE`
Plan incorrect	`PerspectiveGridPlaneType.NOPLANE`

Pour une grille de perspective à un point, seuls les plans de gauche et au sol sont valides.

Le script suivant permet de définir le plan de perspective actif :

```
//Set left plane as the active plane
app.activeDocument.setPerspectiveActivePlane(PerspectiveGridPlaneType.LEFTPLANE);

//Set right plane as the active plane
app.activeDocument.setPerspectiveActivePlane(PerspectiveGridPlaneType.RIGHTPLANE);

//Set floor plane as the active plane
app.activeDocument.setPerspectiveActivePlane(PerspectiveGridPlaneType.FLOORPLANE);
```

Réalisation d'un dessin sur une grille de perspective

Lorsque la grille de perspective est activée, les différents modes de dessin vous permettent de dessiner des objets ou de les manipuler en perspective. Le script suivant permet de créer un document, d'afficher une grille de perspective à deux points et de dessiner des objets graphiques sur le plan de gauche :

```
//Create a new document
var mydoc = app.documents.add();

//Select the default two-point perspective preset
mydoc.selectPerspectivePreset("[2P-Normal View]");

//Display the perspective grid defined in the document
mydoc.showPerspectiveGrid();

//Check if active plane is set to left; if not, set it to left
if (mydoc.getPerspectiveActivePlane() != PerspectiveGridPlaneType.LEFTPLANE)
{
    mydoc.setPerspectiveActivePlane(PerspectiveGridPlaneType.LEFTPLANE);
}

//Draw rectangle in perspective, then resize to 200% and move
var myrect = mydoc.pathItems.rectangle(30, -30, 30, 30, false);
myrect.resize(200, 200, true, false, false, false, 100, Transformation.TOPLEFT);
myrect.translate (-420, 480);

//Draw ellipse in perspective
var myellipse = mydoc.pathItems.ellipse(60, -60, 30, 30, false, true);

//Draw rounded rectangle in perspective
var myrrect = mydoc.pathItems.roundedRectangle(90, -90, 30, 30, 10, 10, false);

//Draw polygon in perspective
var mypoly = mydoc.pathItems.polygon(-105, 105, 15, 7, false);

//Draw star in perspective
var mystar = mydoc.pathItems.star(-135, 135, 15, 10, 6, false);

//Draw path in perspective
var newPath = mydoc.pathItems.add();
var lineList = new Array(4);
lineList[0] = new Array(0,0);
lineList[1] = new Array(60,0);
lineList[2] = new Array(30,45);
lineList[3] = new Array(90,110);
newPath.setEntirePath(lineList);
```

Mise en perspective d'objets

Si un objet graphique n'est pas en perspective, utilisez la méthode `bringInPerspective()` pour le mettre en perspective et le placer sur un plan.

Le script suivant permet de créer un document, de dessiner des objets graphiques et de les placer dans une grille de perspective à trois points :

```
ï»¿//Create a new document
var mydoc = app.documents.add();

//Draw ellipse
var myellipse = mydoc.pathItems.ellipse(60, -60, 30, 30, false, true);

//Draw polygon
var mypoly = mydoc.pathItems.polygon(-105, 105, 15, 7, false);

//Draw star
var mystar = mydoc.pathItems.star(-135, 135, 15, 10, 6, false);

//Select the default three-point perspective preset
mydoc.selectPerspectivePreset("[3P-Normal View]");

//Display the perspective grid defined in the document
mydoc.showPerspectiveGrid();

//Check if active plane is set to left; if not, set it to left
if (mydoc.getPerspectiveActivePlane() != PerspectiveGridPlaneType.LEFTPLANE)
{
    mydoc.setPerspectiveActivePlane(PerspectiveGridPlaneType.LEFTPLANE);
}

//Bring the ellipse to the active plane (left plane)
myellipse.bringInPerspective(-100,-100, PerspectiveGridPlaneType.LEFTPLANE);

//Bring the polygon to the right plane
mypoly.bringInPerspective(100,-100,PerspectiveGridPlaneType.RIGHTPLANE);

//Bring the star to the floor plane
mystar.bringInPerspective(100,100,PerspectiveGridPlaneType.FLOORPLANE);
```

6 Création de scripts à l'aide de VBScript

Ce chapitre utilise des exemples de script, accompagnés d'explications pour vous aider à vous familiariser avec la création de scripts Illustrator à l'aide de VBScript.

Pour plus d'informations

Vous trouverez plusieurs exemples de scripts étendus dans le dossier `/Scripting/Sample Scripts` du répertoire d'installation d'Illustrator CS5.

Pour plus de détails sur les classes, objets, propriétés, méthodes et paramètres individuels, ainsi que pour obtenir des exemples de scripts illustrant leur application, reportez-vous au guide *Adobe Illustrator CS Scripting Reference: VBScript* (Référence de script Adobe Illustrator CS : VBScript), disponible dans le dossier `/Scripting/Documentation` du répertoire d'installation d'Illustrator CS5. Vous pouvez également consulter la bibliothèque de types d'Illustrator CS5 depuis la plupart des éditeurs VBScript ou de n'importe quelle application Microsoft Office ; voir « Affichage du modèle d'objet VBScript », page 10.

Si vous ne comprenez pas les concepts et les termes utilisés dans ce chapitre, reportez-vous au document *Introduction aux scripts Adobe*.

Votre premier script Illustrator

Le premier projet classique abordé lors de l'apprentissage d'un nouveau langage de programmation consiste à afficher le message « Hello World! ». Procédez comme suit :

1. Lancez une application de traitement de texte (par exemple, Bloc-notes).
2. Entrez le code suivant :

```
Rem Hello World
Set appRef = CreateObject("Illustrator.Application")
Rem Create a new document and assign it to a variable
Set documentRef = appRef.Documents.Add
Rem Create a new text frame item and assign it to a variable
Set sampleText = documentRef.TextFrames.Add
Rem Set the contents and position of the TextFrame
sampleText.Position = Array(200, 200)
sampleText.Contents = "Hello World!"
```

3. Enregistrez le fichier en texte seul dans le dossier de votre choix, avec l'extension `.vbs`.
4. Pour tester le script, utilisez l'une des méthodes suivantes :

- Cliquez deux fois sur le fichier.
- Lancez l'application Illustrator, choisissez la commande Fichier > Scripts > Autre script, puis recherchez et exécutez votre fichier de script.

CONSEIL : pour ajouter le script au menu des scripts Illustrator (Fichier > Scripts), vous devez l'enregistrer dans le dossier Scripts. Le script s'affichera dans le menu au prochain démarrage d'Illustrator. Pour plus de détails, reportez-vous à la section « Installation de scripts dans le menu Scripts », page 10. En général, lorsque vous lancez un script VBScript à partir du menu Scripts, les boîtes de dialogue `msgBox` ne s'affichent pas correctement.

Ajout de caractéristiques à « Hello World »

Nous allons maintenant créer un script qui modifiera le document Illustrator créé avec votre premier script. Pour ce deuxième script, vous allez apprendre à :

- obtenir le document actif ;
- obtenir la largeur du document actif ;
- redimensionner le bloc de texte pour l'adapter à la largeur du document.

Si vous n'avez pas enregistré le document Illustrator avant de le fermer, exécutez votre premier script une nouvelle fois pour créer un document.

Procédez comme suit :

1. Copiez le script suivant dans votre éditeur de texte, puis enregistrez le fichier.

```
Set appRef = CreateObject("Illustrator.Application")
'Get the active document
Set documentRef = appRef.ActiveDocument
Set sampleText = documentRef.TextFrames(1)
' Resize the TextFrame item to match the document width
sampleText.Width = documentRef.Width
sampleText.Left = 0
```

2. Exécutez le script.

Accès et référencement d'objets

Lors de l'écriture d'un script, vous devez d'abord indiquer le fichier, ou `Document`, sur lequel le script doit agir. A l'aide de l'objet `Application`, le script peut créer un document, ouvrir un document existant ou agir sur un document déjà ouvert.

Le script peut créer des objets dans le document, effectuer des opérations sur des objets sélectionnés par l'utilisateur ou sur les objets d'une collection. Les sections suivantes illustrent des techniques d'accès, de référencement et de manipulation d'objets Illustrator.

Obtention d'objets de collection

En général, pour obtenir une référence à un objet spécifique, vous pouvez explorer la hiérarchie du contenu. Par exemple, pour utiliser la variable `myPath` afin de stocker une référence vers le premier élément `PathItem` dans le second calque du document actif :

```
Set myPath = appRef.ActiveDocument.Layers(2).PathItems(1)
```

Les scripts suivants montrent comment référencer un objet en tant que partie d'un document :

```
Set documentRef = appRef.ActiveDocument

Set pageItemRef = documentRef.PageItems(1)
```

Dans le script suivant, la variable `pageItemRef` ne fait pas nécessairement référence au même objet que le script précédent, car ce script inclut une référence à un calque :

```
Set documentRef = appRef.ActiveDocument
Set pageItemRef = documentRef.Layers(1).PageItems(1)
```

Les index VBScript commencent par 1 dans les collections d'objets. Toutefois, VBScript vous permet de spécifier si les index de tableaux de données commencent par 1 ou par 0. Pour plus de détails sur la spécification du numéro de départ des index de tableaux de données, reportez-vous aux documents ou didacticiels de VBScript.

Création d'objets

Pour créer des objets, vous pouvez utiliser un script. Pour créer des objets disponibles à partir d'objets de collection, utilisez la méthode `Add` de l'objet de collection :

```
Set myDoc = appRef.Documents.Add()

Set myLayer = myDoc.Layers.Add()
```

Certains objets de collection ne disposent pas d'une méthode `Add`. Pour créer un objet de ce type, définissez une variable et appliquez la méthode `CreateObject`. Par exemple, le code suivant crée un objet `CMYKColor` avec le nom de variable `newColor` :

```
Set newColor = CreateObject ("Illustrator.CMYKColor")
```

Utilisation de sélections

Lorsque l'utilisateur effectue une sélection dans un document, les objets sélectionnés sont stockés dans la propriété `selection` du document en question. Pour accéder à tous les objets sélectionnés dans le document actif :

```
Set appRef = CreateObject ("Illustrator.Application")
Set documentRef = appRef.ActiveDocument
selectedObjects = documentRef.Selection
```

En fonction des éléments sélectionnés, la valeur de la propriété `selection` peut être un tableau de tout type d'objets d'illustration. Pour obtenir ou manipuler les propriétés des éléments d'illustration sélectionnés, vous devez récupérer les éléments individuels dans le tableau de données. Pour connaître le type d'un objet, utilisez la propriété `typename`.

L'exemple suivant permet d'obtenir le premier objet dans le tableau de données, puis d'afficher le type d'objet :

```
Set appRef = CreateObject ("Illustrator.Application")
Set documentRef = appRef.ActiveDocument
selectedObjects = documentRef.Selection
Set topObject = selectedObjects(0)
MsgBox(topObject.Typename)
```

La méthode `MsgBox` n'affiche pas de boîte de dialogue lorsque le script est exécuté à partir du menu Scripts d'Illustrator (Fichier > Scripts).

Le premier objet d'un tableau de données de sélection correspond à l'objet sélectionné *ajouté* en dernier à la page, et non au dernier objet sélectionné.

Sélection d'objets d'illustration

Pour sélectionner un objet d'illustration, utilisez la propriété `Selected` de l'objet en question.

Utilisation de blocs de texte

Pour créer un bloc de texte d'un type donné sous VBScript, utilisez la méthode `TextFrames` correspondant au type de bloc souhaité :

```
Set rectRef = docRef.PathItems.Rectangle(700, 50, 100, 100)

' Use the AreaText method to create the text frame
Set areaTextRef = docRef.TextFrames.AreaText(rectRef)
```

Blocs liés

Tout comme dans l'application Illustrator, vous pouvez lier des blocs de texte captifs ou curvilignes.

Pour lier des blocs de texte existants, utilisez la propriété `NextFrame` ou `PreviousFrame` de l'objet `TextFrames`.

Lorsque vous copiez le script suivant dans un éditeur de texte ou de scripts, placez la valeur de la propriété `Contents` sur une seule ligne. Le caractère de ligne longue (_) n'est pas valide dans le cadre d'une chaîne.

```
Set appRef = CreateObject("Illustrator.Application")
Set myDoc = appRef.Documents.Add
Set myPathItem1 = myDoc.PathItems.Rectangle(244, 64, 82, 76)
Set myTextFrame1 = myDoc.TextFrames.AreaText(myPathItem1)
   myTextFrame1.Contents = "This is two text frames linked together as one story, with
text flowing from the first to the last."
Set myPathItem2 = myDoc.PathItems.Rectangle(144, 144, 42, 116)
Set myTextFrame2 = myDoc.TextFrames.AreaText(myPathItem2)

'Use the NextFrame property to thread the frames
myTextFrame1.NextFrame = myTextFrame2

appRef.Redraw()
```

Création d'objet d'article unique à l'aide de blocs liés

Les blocs liés créent un objet `story` unique. Pour observer cette opération, exécutez le script VBScript suivant après l'exécution du script à la section « Blocs liés », page 58.

```
Set myDoc = appRef.ActiveDocument
myMsg = "alert(""There are " & CStr(myDoc.TextFrames.Count) & " text frames. "")"
appRef.DoJavaScript myMsg
myMsg = "alert(""There are " & CStr(myDoc.Stories.Count) & " storiess. "")"
appRef.DoJavaScript myMsg
```

Création de tracés et de formes

Cette section explique comment créer des éléments comportant des tracés.

Tracés

Pour créer un tracé libre, spécifiez une série de points de tracé sous la forme d'une série de coordonnées x-y ou d'objets `PathPoint`.

Les coordonnées x-y limitent le tracé à des segments droits. Pour créer un tracé incurvé, vous devez créer des objets `PathPoint`. Le tracé peut consister en une combinaison de coordonnées de page et d'objets `PathPoint`.

Spécification d'une série de coordonnées x-y

Pour spécifier un tracé à l'aide de paires de coordonnées de page, utilisez la méthode `SetEntirePath()` de l'objet `PathItems`. Le script suivant spécifie trois paires de coordonnées x-y permettant de créer un tracé comportant trois points :

```
Set appRef = CreateObject ("Illustrator.Application")

Set firstPath = appRef.ActiveDocument.PathItems.Add
    firstPath.Stroked = True
firstPath.SetEntirePath(Array(Array(220, 475),Array(375, 300),Array(200, 300)))
```

Utilisation d'objets PathPoint

Pour créer un objet `PathPoint`, vous devez définir trois valeurs relatives au point :

- un point `anchor` fixe, représentant le point sur le tracé ;
- deux points directeurs, `left direction` et `right direction`, qui vous permettent de contrôler la courbure du segment de tracé.

Vous définissez chaque propriété en tant que tableau de coordonnées de page au format `(Array (x,y))`.

- Si les trois propriétés d'un objet `PathPoint` ont les mêmes coordonnées et que les propriétés de l'objet `PathPoint` suivant sur la ligne sont égales les unes aux autres, vous obtenez un segment de ligne droite.
- Si plusieurs propriétés d'un objet `PathPoint` présentent des valeurs différentes, le segment relié au point est incurvé.

Pour créer un tracé ou ajouter des points à un tracé existant à l'aide d'objets `PathPoint`, créez un objet `PathItem`, puis ajoutez les points de tracé en tant qu'objets enfants dans l'objet `PathItem` :

```
Set appRef = CreateObject ("Illustrator.Application")

Set firstPath = appRef.ActiveDocument.PathItems.Add
    firstPath.Stroked = true
Set newPoint = firstPath.PathPoints.Add
'Using identical coordinates creates a straight segment
newPoint.Anchor = Array(75, 300)
newPoint.LeftDirection = Array(75, 300)
newPoint.RightDirection = Array(75, 300)

Set newPoint2 = firstPath.PathPoints.Add
newPoint2.Anchor = Array(175, 250)
newPoint2.LeftDirection = Array(175, 250)
newPoint2.RightDirection = Array(175, 250)
```

```
Set newPoint3 = firstPath.PathPoints.Add
'Using different coordinates creates a curve
newPoint3.Anchor = Array(275, 290)
newPoint3.LeftDirection = Array(135, 150)
newPoint3.RightDirection = Array(155, 150)
```

Combinaison de types de point de tracé

L'exemple suivant permet de créer un tracé avec trois points :

```
Set appRef = CreateObject("Illustrator.Application")
Set myDoc = appRef.ActiveDocument
Set myLine = myDoc.PathItems.Add
    myLine.Stroked = True
    myLine.SetEntirePath( Array( Array(320, 475), Array(375, 300)))

' Append another point to the line
Set newPoint = myLine.PathPoints.Add
    'Using identical coordinates creates a straight segment
    newPoint.Anchor = Array(220, 300)
    newPoint.LeftDirection = Array(220, 300)
    newPoint.RightDirection = Array(220, 300)
```

Formes

Pour créer une forme, utilisez la méthode `PathItems` correspondant au nom de la forme (par exemple, `ellipse`, `rectangle` ou `polygon`), puis utilisez les paramètres pour indiquer la position de la forme, sa taille et d'autres caractéristiques, telles que le nombre de côtés dans le cas d'un polygone.

Rappel :

- Toutes les mesures, ainsi que les coordonnées de page sont traitées par le moteur de création de scripts. Pour plus de détails, reportez-vous à la section « Unités de mesure », page 30.
- Les coordonnés x et y sont mesurées à partir de l'angle inférieur gauche du document, comme indiqué dans le panneau Informations de l'application Illustrator. Pour plus de détails, reportez-vous à la section « Positionnement et dimensions d'un élément de page », page 30.

Création d'un rectangle

Observez l'exemple suivant :

```
Set appRef = CreateObject("Illustrator.Application")
Set frontDocument = appRef.ActiveDocument
' Create a new rectangle with
' top = 144, left side = 144, width = 72, height = 144
Set newRectangle = frontDocument.PathItems.Rectangle(144,144,72,144)
```

L'exemple permet de créer un rectangle avec les propriétés suivantes :

- L'angle supérieur du rectangle est à 2 pouces (144 points) du bord inférieur de la page.
- Le bord gauche est à 2 pouces (144 points) du bord gauche de la page.
- Le rectangle mesure 1 pouce (72 points) de large sur 2 pouces (144 points) de long.

Création d'un polygone

Observez l'exemple suivant :

```
Set appRef = CreateObject("Illustrator.Application")
Set frontDocument = appRef.ActiveDocument
' Create a new polygon with
' top = 144, left side = 288, width = 72, height = 144
Set newPolygon = frontDocument.PathItems.Polygon(144,288,72,7)
```

L'exemple permet de créer un polygone avec les propriétés suivantes :

- Le point central de l'objet est placé à 2 pouces (144 points) sur l'axe horizontal et à 4 pouces (288 points) sur l'axe vertical.
- La longueur du rayon du centre à chaque coin s'élève à 1 pouce (72 points).
- Le polygone est composé de 7 côtés.

Utilisation de valeurs d'énumération

Les propriétés utilisant des valeurs d'énumération en VBScript font appel à une valeur sous forme numérique à la place d'une valeur sous forme de texte. Par exemple, la propriété `Orientation` de l'objet `TextFrame` détermine si le contenu textuel d'un bloc de texte est horizontal ou vertical. La propriété utilise l'énumération `aiTextOrientation`, qui a deux valeurs possibles, à savoir `aiHorizontal` et `aiVertical`.

Pour rechercher les valeurs numériques des énumérations, utilisez l'une des méthodes suivantes :

- l'explorateur d'objets dans l'environnement de votre éditeur de scripts (voir la section « Affichage du modèle d'objet VBScript », page 10) ;
- le guide *Adobe Illustrator CS5 Scripting Reference: VBScript* (Référence de script Adobe Illustrator CS5 : VBScript), qui répertorie les valeurs numériques immédiatement à la suite des valeurs constantes dans le chapitre traitant des énumérations à la fin du manuel. L'exemple suivant est extrait de ce document :

Type d'énumeration	Valeurs	Signification
AiTextOrientation	aiHorizontal = 0 aiVertical = 1	Orientation du texte dans un bloc de texte

L'exemple suivant spécifie une orientation verticale du texte :

```
Set appRef = CreateObject ("Illustrator.Application")
Set docRef = appRef.Documents.Add
Set textRef = docRef.TextFrames.Add
textRef.Contents = "This is some text content."
textRef.Left = 50
textRef.Top = 700
textRef.Orientation = 1
```

En général, une bonne méthode de travail consiste à placer la valeur de texte dans un commentaire à la suite de la valeur numérique, comme dans l'instruction d'exemple suivante :

```
textRef.Orientation = 1 ' aiVertical
```

Utilisation de la grille de perspective

La grille de perspective est une nouvelle fonctionnalité proposée par Illustrator CS5, qui vous permet de créer des illustrations et de les manipuler dans un environnement spatial à partir de lois de perspective établies. Pour activer la grille de perspective, ouvrez le menu Affichage et cliquez sur Grille de perspective, ou utilisez les outils de perspective disponibles dans la barre d'outils.

Le SDK fournit une interface de programmation qui permet d'utiliser la grille de perspective avec des programmes et qui est partiellement compatible avec vos scripts. Un script permet d'effectuer les opérations suivantes :

- définir les paramètres par défaut de la grille à partir de valeurs prédéfinies ;
- afficher ou masquer la grille ;
- définir le plan actif ;
- dessiner un objet en perspective sur le plan actif ;
- mettre un objet en perspective.

Utilisation des paramètres de perspective prédéfinis

Illustrator fournit des paramètres prédéfinis de grille par défaut, correspondant à des perspectives à un, deux ou trois points. Ces paramètres prédéfinis s'intitulent « `[1P-NormalView]` », « `[2P-NormalView]` » et « `[3P-NormalView]` ».

Le script suivant montre comment sélectionner le paramètre prédéfini de perspective à deux points par le biais d'un programme :

```
Set appRef = CreateObject ("Illustrator.Application")

Rem Create a new document
Set docRef = appRef.Documents.Add()

Rem Select the default two-point perspective preset
docRef.SelectPerspectivePreset("[2P-Normal View]")
```

Vous pouvez créer de nouveaux paramètres prédéfinis de perspective, exporter ces paramètres prédéfinis vers un fichier ou les importer depuis un fichier. Les scripts suivants montrent comment exporter et importer des paramètres prédéfinis :

```
Set appRef = CreateObject ("Illustrator.Application")
Rem Create a new document
Set docRef = appRef.Documents.Add()
Rem Export perspective presets to a file
docRef.ExportPerspectiveGridPreset("C:/scripting/PGPresetsExported")


Set appRef = CreateObject ("Illustrator.Application")
Rem Create a new document
Set docRef = appRef.Documents.Add()
Rem Import perspective presets from a file
docRef.ImportPerspectiveGridPreset("C:/scripting/PGPresets")
```

Affichage ou masquage de la grille

Le script suivant permet d'afficher ou de masquer la grille de perspective par le biais d'un programme :

```
Set appRef = CreateObject ("Illustrator.Application")
```

```
Rem Create a new document
Set docRef = appRef.Documents.Add()

Rem Show the Perspective Grid defined in the document
docRef.ShowPerspectiveGrid();

Rem Hide the Perspective Grid defined in the document
docRef.HidePerspectiveGrid();
```

Définition du plan actif

Les types de plan de la grille de perspective sont :

Plan de gauche	`aiLEFTPLANE (1)`
Plan de droite	`aiRIGHTPLANE (2)`
Plan au sol	`aiFLOORPLANE (3)`
Plan incorrect	`aiNOPLANE (4)`

Pour une grille de perspective à un point, seuls les plans de gauche et au sol sont valides.

Le script suivant permet de définir le plan de gauche comme plan de perspective actif :

```
Set appRef = CreateObject ("Illustrator.Application")

Rem Create a new document
Set docRef = appRef.Documents.Add()

Rem Set left plane as the active plane
docRef.SetPerspectiveActivePlane(1) 'aiLEFTPLANE
```

Réalisation d'un dessin sur une grille de perspective

Lorsque la grille de perspective est activée, les différents modes de dessin vous permettent de dessiner des objets ou de les manipuler en perspective. Le script suivant permet de créer un document, d'afficher une grille de perspective à deux points et de dessiner des objets graphiques sur le plan de gauche :

```
Set appRef = CreateObject ("Illustrator.Application")

Rem Create a new document
Set docRef = appRef.Documents.Add()

Rem Select the default two point perspective preset
docRef.SelectPerspectivePreset("[2P-Normal View]")

Rem Display the perspective grid defined in the document
docRef.ShowPerspectiveGrid()

Rem Check if active plane is set to left, otherwise set it to left
If docRef.GetPerspectiveActivePlane() <> 1 Then
   docRef.SetPerspectiveActivePlane(1) 'aiLEFTPLANE
End If

Rem Draw rectangle in perspective, then resize to 200% and move
Set pathItemRect = docRef.PathItems.Rectangle(30, -30, 30, 30, False)
```

```
call pathItemRect.Resize(200, 200, True, False, False, False, 100, 2)
call pathItemRect.Translate(-420, 480)

Rem Draw ellipse in perspective
Set pathItemEllipse = docRef.PathItems.Ellipse(60, -60, 30, 30, False, True)

Rem Draw rounded rectangle in perspective
Set pathItemRRect = docRef.PathItems.RoundedRectangle(90, -90, 30, 30, 10, 10, False)

Rem Draw polygon in perspective
Set pathItemPoly = docRef.PathItems.Polygon(-105, 105, 15, 7, False)

Rem Draw star in perspective
Set pathItemStar = docRef.PathItems.Star(-135, 135, 15, 10, 6, False)

Rem Draw path in perspective
Set newPath = docRef.PathItems.Add()
newPath.SetEntirePath(Array(Array(0,0),Array(60,0),Array(30,45),Array(90,110)))
```

Mise en perspective d'objets

Si un objet graphique n'est pas perspective, utilisez la méthode `bringInPerspective()` pour le mettre en perspective et le placer sur un plan.

Le script suivant permet de créer un document, de dessiner des objets graphiques et de les placer dans une grille de perspective à trois points :

```
Set appRef = CreateObject ("Illustrator.Application")

Rem Create a new document
Set docRef = appRef.Documents.Add()

Rem Draw ellipse
Set pathItemEllipse = docRef.PathItems.Ellipse(60, -60, 30, 30, False, True)

Rem Draw polygon
Set pathItemPoly = docRef.PathItems.Polygon(-105, 105, 15, 7, False)

Rem Draw star
Set pathItemStar = docRef.PathItems.Star(-135, 135, 15, 10, 6, False)

Rem Select the default three-point perspective preset
docRef.SelectPerspectivePreset("[3P-Normal View]")

Rem Display the perspective grid defined in the document
docRef.ShowPerspectiveGrid()

Rem Check if active plane is set to left, otherwise set it to left
If docRef.GetPerspectiveActivePlane() <> 1 Then
   docRef.SetPerspectiveActivePlane(1) 'aiLEFTPLANE
End If

Rem Bring the ellipse to the active plane (left plane)
Call pathItemEllipse.BringInPerspective(100,100, 1) 'aiLEFTPLANE

Rem Bring the polygon to the right plane
Call pathItemPoly.BringInPerspective(100,-100,2) 'aiRIGHTPLANE
```

```
Rem Bring the star to the floor plane
Call pathItemStar.BringInPerspective(100,100,3) 'aiFLOORPLANE
```

Index

L

M

N

O

P

Q

R

S

T

U

V