ADOBE PHOTOSHOP CS4

GUIDE DES SCRIPTS

Guide des scripts Photoshop® Adobe® Creative Suite® 4

Adobe Systems Inc., 345 Park Avenue, San Jose, Californie 95110, Etats-Unis.

Table des matières

1 Introduction 6

A propos de ce manuel 6

Conventions utilisées dans ce guide 6

2 Concepts de base pour les scripts Photoshop ....

Définition du calque actif 28
Définition des couches actives 29

Ouverture d'un document 30
Ouverture d'un fichier avec le format de fichier par défaut 30
Définition des formats de fichier à ouvrir 30

Enregistrement d'un document 32

Définition des préférences de l'application 34

Autorisation ou refus d'ouverture de boîtes de dialogue 34

Utilisation du modèle d'objet de Photoshop 35
Utilisation de l'objet Application 35
Utilisation de l'objet Document 36
Manipulation d'un objet document 36
Utilisation des objets calque 38
Création d'un objet ArtLayer 38
Création d'un objet Layer Set 39
Référencement des objets ArtLayer 40
Utilisation des objets Layer Set 41
Liaison d'objets calque 41
Application de styles aux calques 42
Utilisation de l'objet Text Item 43
Définition d'un type de calque 43
Ajout et manipulation de texte dans un objet Text Item 44
Utilisation des objets Selection 45
Création et définition d'une sélection 45
Application d'un contour au cadre de sélection 46
Inversion des sélections 47
Dilatation, contraction et contour progressif des sélections 47
Remplissage d'une sélection 47
Chargement et stockage des sélections 48
Utilisation des objets Channel 49
Changement des types de couches 49
Utilisation de l'objet Document Info 49
Utilisation des objets history state 50
Utilisation des objets Notifier 51
Utilisation de l'objet PathItem 52

Utilisation des objets de couleur 54
Classes de couleur unie 54
Utilisation des valeurs hexadécimales pour les couleurs RVB 55
Obtention et conversion de couleurs 55
Comparaison des couleurs 56
Obtention d'une couleur Web sécurisée 56

Utilisation de filtres 56
Filtres divers 57

Présentation de l'interaction avec le Presse-papiers 57
Utilisation des commandes copy et paste 57
Utilisation de la commande/méthode copy merged 58

Utilisation des unités 59
Valeurs d'unité 59
Types de valeurs d'unité spéciaux 59

Précisions sur l'unité en AppleScript 59
Utilisation des valeurs d'unité dans des calculs 60
Utilisation de la valeur d'unité 61
Définition des unités de règle et de texte dans un script 62

Exemples de scripts JavaScript pour l'automatisation d'un flux de production 63

Programmation avancée de scripts 64
Utilisation des préférences de document 64
Application de couleur à un élément de texte 67
Application d'un filtre Onde 71
Définition de la zone d'un objet de sélection 71
Application d'un filtre Flou directionnel 75

4 Gestionnaire de scripts 77

Module externe Ecouteur de scripts 77
Installation de l'écouteur de scripts 77

Objets du gestionnaire de scripts 78

Enregistrement d'un script à l'aide de l'écouteur de scripts 78

Utilisation du gestionnaire de scripts à partir d'un script JavaScript 79

Utilisation du gestionnaire de scripts à partir d'un script VBS 81

Exécution de code du gestionnaire de scripts JavaScript à partir d'un script VBScript 83

Exécution de code du gestionnaire de scripts JavaScript à partir d'un script AppleScript 84

Utilisation de l'écouteur de scripts pour rechercher des ID d'événements et de classes 85

Index 88

1 Introduction

A propos de ce manuel

Ce manuel propose une introduction aux scripts pour Adobe® Photoshop® CS4 sous Mac OS® et Windows®.

Le chapitre 1 traite des conventions de base utilisées dans ce manuel.

Le chapitre 2 présente brièvement les scripts, notamment comment les exécuter, ainsi que le modèle d'objet de Photoshop.

Le chapitre 3 expose les éléments et objets spécifiques à Photoshop et décrit des techniques de script avancées pour l'application Photoshop. Des exemples de code sont fournis dans trois langages :

- AppleScript
- VBScript
- JavaScript™

REMARQUE : des informations de référence distinctes pour les scripts Photoshop sont fournies pour chacun des trois langages dans les Guide de référence pour les scripts livrés avec cette installation, ou via les explorateurs d'objets disponibles pour chaque langage (voir les sections « Affichage du dictionnaire AppleScript de Photoshop », page 22 et « Affichage de la bibliothèque de types de Photoshop (VBS) », page 23). Pour plus de détails sur l'utilisation de l'afficheur modèles d'objets Extend Script, consultez le *JavaScript Tools Guide (Guide des outils JavaScript)*.

Le chapitre 4 traite du gestionnaire de scripts qui vous permet d'écrire des scripts ciblant la fonctionnalité Photoshop sinon inaccessible dans l'interface de scripts.

REMARQUE : consultez le fichier `LISEZ-MOI` livré avec Photoshop pour obtenir des informations de dernière minute, des exemples de scripts et des informations sur les problèmes non résolus.

Conventions utilisées dans ce guide

Les exemples de code et de langage particulier apparaissent dans une police Courier à chasse fixe :

```
app.documents.add
```

Plusieurs conventions sont utilisées lorsqu'il est fait référence à AppleScript, VBScript et JavaScript. Gardez à l'esprit les abréviations suivantes :

- AS pour AppleScript
- VBS pour VBScript
- JS pour JavaScript

Le terme « commandes » est utilisé pour faire référence à la fois aux commandes en AppleScript et aux méthodes en VBScript et JavaScript.

Lorsqu'il est fait référence à des propriétés et à des commandes spécifiques, ce manuel suit la convention de dénomination AppleScript pour cette propriété, et les noms VBScript et JavaScript apparaissent entre parenthèses. Par exemple :

« La propriété `display dialogs` (`DisplayDialogs/displayDialogs`) fait partie de l'objet Application. »

Dans ce cas, `display dialogs` fait référence à la propriété AppleScript, `DisplayDialogs` fait référence à la propriété VBScript, et `displayDialogs` fait référence à la propriété JavaScript.

Pour de plus grands blocs de code, les exemples de scripts sont répertoriés sur des lignes distinctes.

AS

```
layer 1 of layer set 1 of current document
```

VBS

```
appRef.ActiveDocument.LayerSets(1).Layers(1)
```

JS

```
app.activeDocument.layerSets[0].layers[0]
```

Enfin, des tableaux sont parfois utilisés pour organiser des listes de valeurs spécifiques à chaque langage.

2 Concepts de base pour les scripts Photoshop

Ce chapitre offre un aperçu des scripts pour Photoshop, décrit la prise en charge des scripts pour les langages AppleScript, VBScript et JavaScript, la manière d'exécuter les scripts et traite du modèle d'objet de Photoshop. Il fournit un exemple simple pour vous aider à écrire votre premier script Photoshop.

Si vous êtes déjà familier des scripts ou des langages de programmation, vous pouvez vous permettre de passer une grande partie de ce chapitre. Utilisez la liste suivante pour trouver les informations dont vous avez besoin.

- Reportez-vous à la section « Modèle d'objet de Photoshop », page 11 pour plus de détails sur le modèle d'objet de Photoshop.
- Pour plus de détails sur la sélection d'un langage de script, consultez le guide *Introduction to Scripting (Introduction aux scripts)*.
- Pour consulter des exemples de scripts créés spécialement pour être utilisés avec Photoshop, reportez-vous au chapitre 3, « Scripts Photoshop », page 22.
- Pour plus de détails sur les objets et commandes Photoshop, utilisez les informations de référence contenues dans les trois manuels fournis avec cette installation : *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4*, *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* et *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4*.

REMARQUE : vous pouvez également consulter des informations sur les objets et commandes Photoshop en utilisant les explorateurs d'objets pour chacun des trois langages de script (voir la section « Affichage des objets, commandes et méthodes Photoshop », page 22).

Présentation des scripts

Un script est une série de commandes ordonnant à Photoshop d'effectuer un ensemble d'actions spécifiques, comme appliquer différents filtres à des sélections dans un document ouvert. Ces actions peuvent être simples et ne concerner qu'un seul objet. Elles peuvent aussi être complexes et concerner plusieurs objets dans un document Photoshop. Les actions peuvent invoquer Photoshop uniquement ou d'autres applications.

Les scripts permettent d'automatiser des tâches répétitives et sont souvent utilisés comme outils d'aide à la création pour rationaliser des tâches qui seraient trop longues à réaliser manuellement. Par exemple, vous pouvez écrire un script pour générer un nombre de versions localisées d'une image particulière ou pour collecter des informations sur les différents profils colorimétriques utilisés par une collection d'images.

Si vous débutez dans l'écriture de scripts, il est préférable de prendre connaissance des informations de base sur les scripts fournies dans le manuel *Introduction to Scripting (Introduction aux scripts)*.

Pourquoi utiliser des scripts plutôt que des actions ?

Si vous avez utilisé les actions Photoshop, vous connaissez leur utilité en matière d'automatisation des tâches répétitives. Vous pouvez exploiter cet avantage grâce aux scripts par l'ajout de fonctions qui ne sont pas disponibles pour les actions Photoshop. Par exemple, les fonctions suivantes sont possibles avec des scripts, pas avec des actions :

- Vous pouvez ajouter une *logique conditionnelle*, de sorte que le script prenne automatiquement ses « décisions » en fonction de la situation en cours. Par exemple, vous pouvez écrire un script qui décide quelle couleur de bordure ajouter en fonction de la taille de la zone sélectionnée dans une image : « Si la zone sélectionnée est inférieure à 2 x 4 pouces, ajoutez une bordure verte ; sinon ajoutez une bordure rouge. »
- Un même script peut effectuer des actions concernant plusieurs applications. Ainsi, en fonction du langage de script que vous utilisez, vous pouvez cibler à la fois Photoshop et une autre application Adobe Creative Suite 4, comme Adobe Illustrator® CS4, dans le même script.
- Vous pouvez ouvrir, enregistrer et renommer des fichiers à l'aide de scripts.
- Vous pouvez copier des scripts d'un ordinateur sur un autre. En revanche, si vous utilisez des actions et que vous changez d'ordinateur, vous devez les recréer.
- Les scripts offrent une plus grande souplesse pour l'ouverture automatique de fichiers. Lorsque vous ouvrez un fichier dans une action, vous devez entrer l'emplacement du fichier dans le code. Les scripts permettent d'utiliser des variables pour les chemins d'accès aux fichiers.

REMARQUE : consultez l'Aide de Photoshop pour plus de détails sur les actions Photoshop.

Prise en charge des scripts dans Photoshop

Photoshop prend en charge les scripts dans trois langages : AppleScript, VBScript et JavaScript. AppleScript et JavaScript s'exécutent sous Mac OS, tandis que JavaScript et VBScript s'exécutent sous Windows. Pour plus de détails sur le choix du langage de script et sur l'utilisation de ces langages avec des applications Adobe, reportez-vous au manuel *Introduction to Scripting (Introduction aux scripts)*.

Reportez-vous aux sections « Création et exécution d'un script AppleScript », page 19, « Création et exécution d'un script VBScript », page 20 et « Création et exécution d'un script JavaScript », page 20.

Vous pouvez appeler des scripts JavaScript à partir de scripts AppleScript et VBScript (voir la section « Exécution de scripts JavaScript à partir d'AS ou de VBS », page 11).

Pour qu'un fichier soit reconnu en tant que script valide par Photoshop, il doit avoir l'extension qui convient.

Type de script	Type de fichier	Extension	Plate-forme
AppleScript	script compilé fichier OSAS	`.scpt` (none)	Mac OS®
JavaScript ExtendScript	texte	`.js` `.jsx`	Mac OS et Windows
VBScript	texte	`.vbs`	Windows
Visual Basic	exécutable	`.exe`	Windows

Prise en charge de JavaScript

Pour qu'un fichier JavaScript soit reconnu en tant que script valide par Photoshop, il doit avoir l'extension `.js` ou `.jsx`. Sous Mac OS, les scripts avec ces deux extensions fonctionnent de manière identique. Sous Windows, si les fichiers de script sont ouverts depuis Photoshop, les extensions `.js` et `.jsx` peuvent être utilisées indifféremment. Cependant, lorsque vous lancez un script avec l'extension `.js` en cliquant deux fois dessus, il est interprété par le moteur Microsoft® JScript et ne peut pas lancer Photoshop. Il est donc préférable de choisir l'extension `.jsx` sous Windows, car le script est interprété avec le moteur ExtendScript.

Les scripts écrits en JavaScript sont accessibles à partir du menu Scripts d'Adobe Photoshop (**Fichier > Scripts**), qui donne un accès simple et rapide à vos scripts JavaScripts. En plaçant un fichier JavaScript à l'emplacement adéquat sur le disque, il est accessible directement à partir du menu Photoshop.

Pour installer un script JavaScript dans le menu Scripts, placez-le dans le dossier Scripts (**Photoshop CS4 /Paramètres prédéfinis /Scripts**). Tous les scripts se trouvant dans ce dossier sont affichés, sans leur extension, dans le menu Scripts. Le nombre de scripts dans le menu Scripts n'est pas limité.

Les scripts ajoutés au dossier Scripts pendant le fonctionnement de Photoshop apparaîtront dans le menu Scripts uniquement après le prochain lancement de l'application.

Tous les scripts se trouvant dans le dossier Scripts et ses sous-dossiers apparaissent en haut du menu **Fichier > Scripts**. Aucune organisation hiérarchique n'est effectuée dans le menu Scripts lorsque vous ajoutez des sous-dossiers.

Exécution d'autres scripts

La commande **Parcourir** au bas du menu **Scripts** (**Fichier > Scripts > Parcourir**) vous permet d'exécuter des scripts qui ne sont pas installés dans le dossier Scripts. Vous pouvez également utiliser la commande Parcourir pour sélectionner les scripts placés dans le dossier Scripts depuis la dernière exécution de l'application.

Lorsque vous choisissez la commande **Parcourir**, la boîte de dialogue d'exploration qui s'affiche vous permet de sélectionner un script à exécuter. Seuls les fichiers `.js` ou `.jsx` s'affichent dans la boîte de dialogue d'exploration. Lorsque vous en sélectionnez un, il est exécuté de la même manière qu'un script installé.

Scripts de démarrage

Pendant son lancement, Photoshop exécute tous les fichiers `.jsx` qui se trouvent dans les dossiers de démarrage.

- Sous Windows, le dossier de démarrage des scripts définis par l'utilisateur est le suivant :

  ```
  C:\Program Files\Common Files\Adobe\Startup Scripts CS4\Adobe Photoshop
  ```

- Sous Mac OS, le dossier de démarrage des scripts définis par l'utilisateur est le suivant :

  ```
  ~/Library/Application Support/Adobe/Startup Scripts CS4/Adobe Photoshop
  ```

Si votre script se trouve dans le dossier de démarrage principal, il est également exécuté par toutes les autres applications de la suite Adobe Creative Suite 3 au démarrage. Si un script de ce type doit être exécuté uniquement par Photoshop, il doit comporter les lignes de code suivantes :

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

Pour plus de détails, consultez le *JavaScript Tools Guide (Guide des outils JavaScript)*.

Exécution de scripts JavaScript à partir d'AS ou de VBS

Comme les scripts JavaScript sont indépendants de la plate-forme, vous pouvez les exécuter à partir d'AppleScript ou de VBScript. Vous pouvez exécuter soit une seule instruction JavaScript, soit un fichier JavaScript complet. Pour plus de détails, consultez le guide *Introduction to Scripting (Introduction aux scripts)*.

Modèle d'objet de Photoshop

Un modèle d'objet de document (DOM) est une interface de programmation qui vous permet d'accéder par programmation à diverses composantes d'un *document* (tel que défini pour cette application) au moyen d'un langage de script. Pour plus de détails sur les modèles d'objets d'Adobe et sur les langages de script qui les prennent en charge, reportez-vous au guide *Introduction to Scripting (Introduction aux scripts)*.

Le DOM Photoshop se compose d'une représentation hiérarchique de l'application Photoshop, des documents utilisés dans celle-ci et des éléments de ces documents. Il vous permet d'accéder aux documents et à leurs composantes et de les manipuler par programmation. Par exemple, à partir du DOM, vous pouvez créer un document, ajouter un calque à un document existant ou modifier la couleur d'arrière-plan d'un calque. La plupart des fonctions disponibles via l'interface utilisateur de Photoshop sont disponibles via le DOM.

Une bonne compréhension du DOM Photoshop et de la manière dont chaque aspect du DOM est associé à l'application Photoshop et à ses documents facilitera l'écriture des scripts.

Hiérarchie d'imbrication

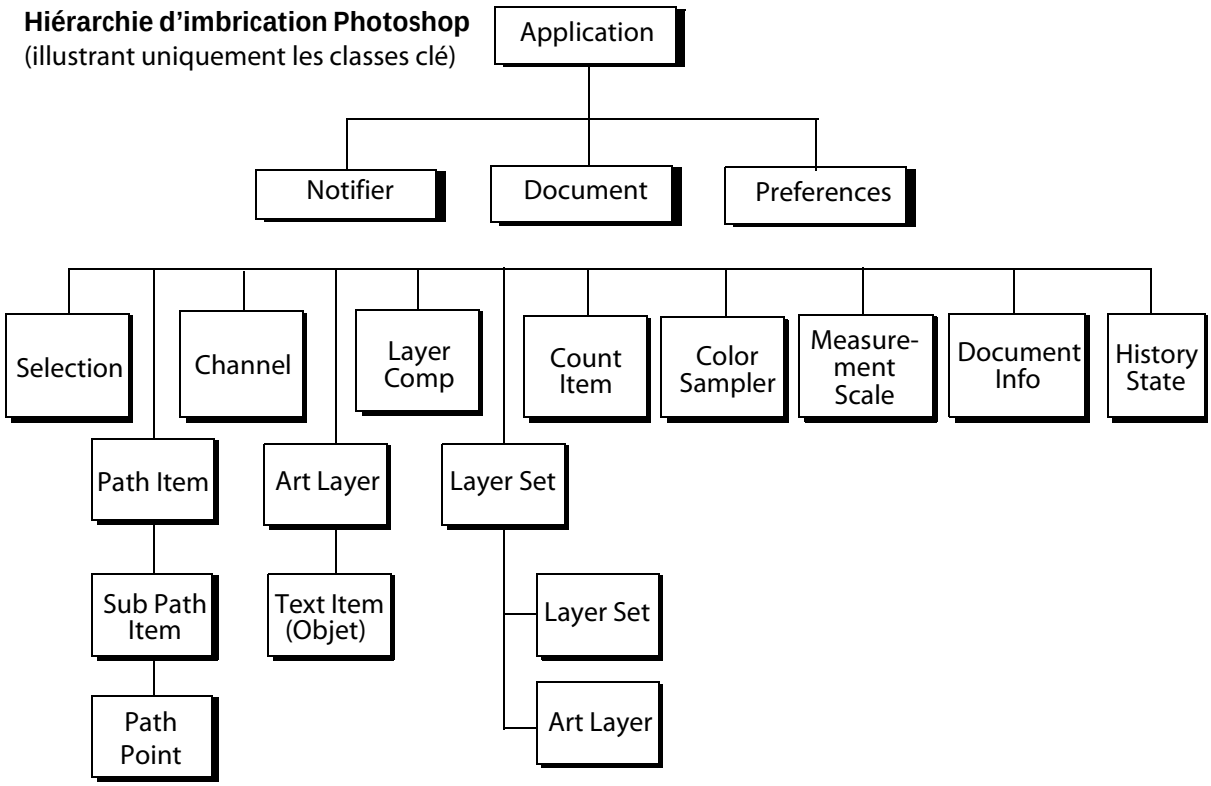
Le modèle d'objet de Photoshop est une *hiérarchie d'imbrication*, ce qui signifie que les objets contenus dans le modèle sont identifiés partiellement par les objets qui les contiennent. Dans Photoshop, l'objet `Application` se trouve en haut de la hiérarchie. Les applications contiennent une collection Documents. La collection Documents contient des objets Document. Un objet Document contient une collection ArtLayers, une collection HistoryStates, une collection Layers, une collection Layersets et une collection Channels. Les commandes ou méthodes du DOM vous permettent d'ordonner à des documents Photoshop d'ajouter et de supprimer des objets, ou de définir ou changer des propriétés individuelles d'objet comme la couleur, la taille et la forme. Dans le diagramme ci-dessous, chaque nœud de la hiérarchie représente une classe dans le DOM Photoshop.

Le modèle d'objet de Photoshop utilise des éléments (AppleScript) ou des collections (VBScript, JavaScript) pour regrouper facilement les classes. L'illustration du modèle d'objet ci-dessous ne représente pas d'éléments ou de collections d'objets. Les classes ne sont pas toutes associées à une collection. Cependant, certaines classes clé sont regroupées par éléments ou collection. Les éléments/collections suivants existent dans Photoshop : `Art Layers`, `Channels`, `Color Samplers`, `Count Items`, `Documents`, `Layers`, `Layer Comps`, `Layer Sets`, `History States`, `Notifiers`, `Path Items`, `Path Points Sub Path`

`Items` et `Text Fonts`. Reportez-vous au guide *Introduction to Scripting (Introduction aux scripts)* pour plus de détails sur les éléments et les collections.

REMARQUE : dans Photoshop, les collections VBScript sont indexées à partir de 1 et non de 0 (à la différence d'autres groupes VBScript, qui commencent l'indexation à partir de 0).

Hiérarchie d'imbrication Photoshop
(illustrant uniquement les classes clé)



Classes Application et Document

La classe `Application` est la racine de la hiérarchie du modèle d'objet de Photoshop. Les scripts doivent cibler l'application appropriée pour être exécutés correctement (voir la section « Ciblage et référencement de l'objet Application », page 24).

La classe `Document` permet d'apporter des modifications à l'image d'un document. Grâce à l'objet `Document`, vous pouvez recadrer, faire pivoter ou appliquer une symétrie à la zone de travail, redimensionner l'image ou la zone de travail et rogner l'image. Vous pouvez également utiliser l'objet `Document` pour obtenir le calque actif, puis enregistrer le document actif puis copier et coller du contenu dans le document actif ou entre plusieurs documents. Pour plus de détails sur l'utilisation des objets Document, reportez-vous aux sections « Création d'objets dans un script », page 24 et « Utilisation de l'objet Document », page 36.

Classes Layer

Photoshop possède deux types de calques : un calque `Art Layer` qui comporte un contenu d'image et un groupe `Layer Set` qui ne comporte aucun ou plusieurs calques graphiques.

Un groupe `Art Layer` est une classe layer à l'intérieur d'un document qui vous permet de travailler sur un élément d'une image sans perturber les autres. Les images se composent généralement de plusieurs calques, définis par un groupe `Layer Set`. Vous pouvez modifier la composition d'une image en modifiant l'ordre et les attributs des calques qui la composent.

Un élément `Text Item` est un type particulier de calque graphique qui vous permet d'ajouter du texte à une image. Dans Photoshop, un élément `Text Item` est mis en œuvre comme une propriété du calque graphique. Pour plus de détails sur les éléments de texte, reportez-vous à la section « Utilisation de l'objet Text Item », page 43.

Un groupe `Layer Set` est une classe qui comporte plusieurs calques. Elle s'apparente à un dossier placé sur votre bureau. A l'image d'un dossier qui peut contenir d'autres dossiers, un groupe de calques est récursif. Ainsi, un groupe de calques peut appeler un autre groupe de calques dans la hiérarchie du modèle d'objet.

Pour plus de détails sur les calques, reportez-vous à la section « Utilisation des objets calque », page 38.

Classe Layer Comp

La classe `Layer Comp` vous permet de créer, gérer et afficher plusieurs versions d'une mise en page dans un seul document.

Classe Channel

La classe `Channel` est utilisée pour stocker les informations de pixel sur la couleur d'une image. La couleur de l'image détermine le nombre de couches disponibles. Par exemple, une image RVB possède quatre couches par défaut : une couche pour chaque couleur primaire et une couche utilisée pour l'édition de l'image entière. Vous pouvez activer la couche rouge pour manipuler les seuls pixels rouges de l'image ou bien choisir de manipuler toutes les couches à la fois.

Ces types de couches sont associés au mode du document et appelés des *couches de composante.* Outre les couches de composante, Photoshop vous permet de créer des couches supplémentaires : vous pouvez créer une *couche de ton direct*, une *couche de zone masquée* et une *couche de zone sélectionnée*.

A l'aide des commandes ou des méthodes d'un objet `Channel`, vous pouvez créer, supprimer et dupliquer des couches. Vous pouvez également récupérer l'histogramme d'une couche, modifier son type ou modifier les couches actuellement sélectionnées.

Pour plus de détails sur les couches, reportez-vous à la section « Utilisation des objets Channel », page 49.

Classe Selection

La classe `Selection` est utilisée pour spécifier une zone de pixels dans le document actif (ou dans un calque sélectionné du document actif) que vous souhaitez manipuler. Pour plus de détails sur les sélections, reportez-vous à la section « Utilisation des objets Selection », page 45.

Classe History State

La classe `History State` est un objet panneau qui conserve une trace des modifications apportées à un document. Chaque fois que vous apportez une modification à une image, le nouvel état de cette image est ajouté au panneau. Ces états sont accessibles à partir de l'objet Document et peuvent être utilisés pour rétablir une ancienne version du document. Un état d'historique peut également être utilisé pour remplir une sélection. Pour plus de détails sur les objets History, reportez-vous à la section « Utilisation des objets history state », page 50.

REMARQUE : dans AppleScript, si vous créez un document et essayez d'afficher immédiatement l'état d'historique, Photoshop renvoie une erreur. Avant de pouvoir accéder aux états d'historique, vous devez donc activer Photoshop en faisant passer l'application au premier plan.

Classe Document Info

La classe `Document Info` stocke des métadonnées à propos d'un document. Les métadonnées sont des données qui permettent de décrire le contenu ou les caractéristiques d'un fichier. Pour plus de détails sur la classe Document Info, reportez-vous à la section « Utilisation de l'objet Document Info », page 49.

Classes Path Item, Sub Path Item et Path Point

La classe `Path Item` représente des informations sur un objet de dessin, comme le contour d'une forme ou une ligne courbe. La classe `Sub Path Item` est contenue dans la classe `Path Item` et donne la véritable géométrie de la forme. La classe `Path Point` contient des informations sur chaque point d'un sous-tracé (voir la section « Utilisation de l'objet PathItem », page 52).

Classe Preferences

La classe `Preferences` vous permet d'accéder aux paramètres des préférences utilisateur et de les définir (voir la section « Utilisation des préférences de document », page 64).

Classe Notifier

L'objet `Notifier` lie un événement à un script. Si, par exemple, vous souhaitez que Photoshop crée automatiquement un document lorsque vous ouvrez l'application, vous pouvez lier un script qui crée un objet `Document` à un événement `Open Application`. Pour plus de détails sur les notifications, reportez-vous à la section « Utilisation des objets Notifier », page 51.

Classe Count Item

L'objet `Count Item` fournit la prise en charge des scripts pour l'outil Comptage.

Classe Color Sampler

L'objet `Color Sampler` fournit la prise en charge des scripts pour l'outil Echantillonnage de couleur.

Classe Measurement Scale

L'objet `Measurement Scale` fournit la prise en charge des scripts pour la nouvelle fonction Echelle de mesure qui vous permet de définir une échelle pour votre document.

Hiérarchie d'imbrication et interface utilisateur de Photoshop

Le tableau ci-dessous décrit la relation de chaque objet avec l'interface utilisateur de Photoshop.

Nom de l'objet	Description	Pour créer cet objet sans utiliser de script
Application	Application Photoshop.	Lancez l'application Photoshop.
Document	Objet utilisé, dans lequel vous créez des calques, des couches, des actions et autres. Dans un script, vous nommez, ouvrez ou enregistrez un document en procédant de la même façon que pour un fichier dans l'application.	Dans Photoshop, choisissez **Fichier > Nouveau** ou **Fichier > Ouvrir**.
Selection	Zone sélectionnée dans un calque ou un document.	Choisissez l'outil Rectangle de sélection ou Lasso et faites glisser la souris.
Path Item	Objet de dessin, tel que le contour d'une forme, une ligne droite ou une courbe.	Choisissez l'outil Sélection de tracé ou Plume et dessinez un tracé à l'aide de la souris.
Channel	Informations de pixel sur la couleur d'une image.	Choisissez **Fenêtre > Couches**.
Art Layer	Classe de calques d'un document vous permettant de travailler sur un élément d'une image sans toucher aux autres.	Choisissez **Calque > Nouveau > Calque** ou **Fenêtre > Calques**.
Layer Set	Collection d'objets `Art Layer`. Group est le nom actuel utilisé dans l'interface Photoshop. Layer Set était le nom utilisé dans une version antérieure de Photoshop. Le nom Object demeure identique à des fins de rétrocompatibilité.	Choisissez **Calque > Nouveau > Groupe.**
Layer Comp	Instantané de l'état des calques d'un document.	Choisissez **Fenêtre > Compositions de calques**. Sélectionnez ensuite l'icône de composition de calques.

Nom de l'objet	Description	Pour créer cet objet sans utiliser de script
Document Info	Métadonnées sur un objet `Document`. **REMARQUE :** les métadonnées sont des données qui permettent de décrire le contenu ou les caractéristiques d'un fichier, comme le nom de fichier, la date et l'heure de création, le nom de l'auteur, le nom de l'image enregistrée dans le fichier, etc.	Choisissez **Fichier > Informations**.
Notifier	Notifie à un script qu'un événement se produit, cet événement déclenchant ensuite l'exécution du script. Par exemple, lorsqu'un utilisateur clique sur le bouton OK, l'objet notifier indique au script ce qu'il doit faire.	Choisissez la commande **Fichier > Scripts > Gestionnaire d'événements de script**.
Preferences	Paramètres de préférences de l'application.	Choisissez **Edition > Préférences** sous Windows, ou **Photoshop > Préférences** sous Mac OS.
History State	Stocke une version du document dans l'état où il se trouvait à chaque fois que vous l'avez enregistré. **REMARQUE :** vous pouvez utiliser un objet `History State` pour remplir un objet `Selection` ou pour rétablir un état antérieur du document.	Choisissez **Fenêtre > Historique**, puis sélectionnez un état d'historique dans le panneau Historique.
Color Sampler	Représente un échantillonnage de couleur dans votre document.	Choisissez l'outil Echantillonnage de couleur, puis cliquez dans le document.
Count Item	Représente un élément compté dans le document.	Choisissez l'outil Comptage, puis cliquez dans le document.
Measurement Scale	Représente l'échelle de mesure de votre document.	L'objet Measurement Scale ne peut pas être créé, mais vous pouvez modifier ses propriétés en choisissant une **échelle de mesure personnalisée pour l'analyse**.

Objets supplémentaires

Le modèle d'objet de Photoshop inclut d'autres objets que ceux décrits dans la hiérarchie d'imbrication ci-dessus. La plupart des classes sont utilisées en tant que types pour les propriétés ou afin d'apporter des informations (en tant qu'arguments) aux commandes et aux méthodes. Par exemple :

- La classe `color value (SolidColor/SolidColor)` fournit le type pour les propriétés `background color (backgroundColor/backgroundColor)` et `foreground color (ForegroundColor/foregroundColor)` de l'objet `Application` (voir la section « Utilisation des objets de couleur », page 54).

- Les options d'ouverture et d'enregistrement des documents sont définies comme des classes transmises aux commandes qui ouvrent et enregistrent les documents ; par exemple, la classe `BMP save options` (`BMPSaveOptions/BMPSaveOptions`) peut être transmise comme argument à la commande ou méthode `save` (`saveAs`/`saveAs`) (voir les sections « Ouverture d'un document », page 30 et « Enregistrement d'un document », page 32).

Constantes

Les *constantes* constituent un élément supplémentaire important du modèle d'objet de Photoshop pour les langages JavaScript et VBScript. Les constantes sont un type de valeur définissant une propriété. Par exemple, la propriété `kind` d'un objet `Art Layer` vous permet de définir uniquement des types spécifiques que Photoshop autorise. Pour des informations d'ordre général sur les constantes, reportez-vous au guide *Introduction to Scripting (Introduction aux scripts)*.

REMARQUE : dans ce document, les valeurs réelles des énumérations pour VBScript sont données au format suivant :

```
newLayerRef.Kind = 2 '2 indicates psLayerKind --> 2 (psTextLayer)
```

Le signe ' avant l'explication crée un *commentaire* et empêche le texte situé à droite de ce signe d'être lu par le moteur de script. Pour plus de détails sur l'utilisation des commentaires, reportez-vous au guide *Introduction to Scripting (Introduction aux scripts)*.

Par exemple, consultez l'objet `ArtLayer` soit dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4*, soit dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4*. Une des propriétés de cet objet est `Kind( kind)`. Le type de valeur de cette propriété comprend un lien vers la constante qui définit les valeurs autorisées pour la propriété. Pour VBScript et JavaScript, les constantes sont respectivement `PSLayerKind` et `LayerKind`. Cliquez dessus afin d'afficher les valeurs utilisables pour définir la propriété `kind`.

REMARQUE : des objets différents peuvent utiliser le même nom de propriété avec des valeurs constantes différentes. Les valeurs constantes pour la propriété `kind` de l'objet `Channel` sont différentes des valeurs constantes pour la propriété `kind` de l'objet `Art Layer`.

Création d'un exemple de script Hello World

Cette section présente un script très simple dans chacun des trois langages de script pour Photoshop. Le premier projet classique abordé lors de l'apprentissage d'un nouvel environnement de programmation consiste à afficher le message « Hello World ».

Les scripts Hello World vont effectuer les tâches suivantes :

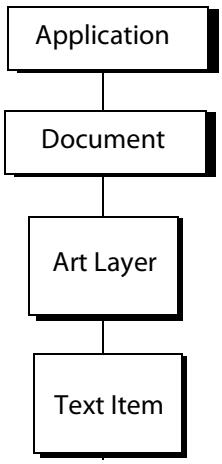
1. Ouvrir l'application Photoshop.
2. Créer un nouvel objet `Document`.

   A la création du document, nous allons également créer une variable nommée `docRef`, puis affecter une référence au document comme valeur de `docRef`. Le document va mesurer 4 pouces de largeur et de 2 pouces de hauteur
3. Créer un objet `Art Layer`.

   Dans notre script, nous créons une variable nommée `artLayerRef` puis attribuons une référence à l'objet `Art Layer` comme valeur de `artLayerRef`.
4. Définir `artLayerRef` en tant qu'objet texte.
5. Définir le contenu de l'objet texte, qui sera « Hello World ».

REMARQUE : nous allons également inclure des commentaires tout au long des scripts, et comme il s'agit de notre premier script, nous allons même en abuser.

Ces étapes reflètent un chemin spécifique dans la structure hiérarchique, comme illustré ci-dessous.



Création et exécution d'un script AppleScript

Vous devez ouvrir l'application Apple® Script Editor afin d'accomplir cette procédure.

REMARQUE : l'emplacement par défaut de ce dernier est **Applications > AppleScript > Editeur de scripts**.

Pour créer et exécuter votre premier script AppleScript Photoshop :

1. Entrez le script ci-dessous dans l'éditeur de scripts :

   REMARQUE : les lignes commençant par « -- » sont des commentaires. Les commentaires sont facultatifs.

```
-- Sample script to create a new text item and
-- change its contents.
--target Photoshop CS4
tell application "Adobe Photoshop CS4"

    -- Create a new document and art layer.
    set docRef to make new document with properties ¬
        {width:4 as inches, height:2 as inches}
    set artLayerRef to make new art layer in docRef

    -- Change the art layer to be a text layer.
    set kind of artLayerRef to text layer

    -- Get a reference to the text object and set its contents.
    set contents of text object of artLayerRef to "Hello, World"
end tell
```

2. Cliquez sur le bouton **Exécuter** pour exécuter le script. Photoshop crée un document, ajoute un nouveau calque, change le calque en calque de texte et définit le texte sur « Hello, World ».

   REMARQUE : si vous rencontrez des erreurs, reportez-vous au guide *Introduction to Scripting (Introduction aux scripts)* dont une section porte sur le débogage d'AppleScript.

Création et exécution d'un script VBScript

Suivez ces étapes pour créer un script VBScript qui affiche le texte *Hello World!* dans un document Photoshop.

Pour créer et exécuter votre premier script VBScript Photoshop :

1. Entrez le script ci-dessous dans un éditeur de scripts ou de texte.

   REMARQUE : les commentaires sont facultatifs.

```
Dim appRef
Set appRef = CreateObject( "Photoshop.Application" )

' Remember current unit settings and then set units to
' the value expected by this script
Dim originalRulerUnits
originalRulerUnits = appRef.Preferences.RulerUnits
appRef.Preferences.RulerUnits = 2

' Create a new 2x4 inch document and assign it to a variable.
Dim docRef
Dim artLayerRef
Dim textItemRef
Set docRef = appRef.Documents.Add(2, 4)

' Create a new art layer containing text
Set artLayerRef = docRef.ArtLayers.Add
artLayerRef.Kind = 2

' Set the contents of the text layer.
Set textItemRef = artLayerRef.TextItem
textItemRef.Contents = "Hello, World!"

' Restore unit setting
appRef.Preferences.RulerUnits = originalRulerUnits
```

2. Enregistrez le fichier en tant que fichier texte, avec l'extension `.vbs`.
3. Cliquez deux fois sur ce fichier dans l'Explorateur de Windows pour exécuter le script.

   Le script ouvre Photoshop.

Création et exécution d'un script JavaScript

Suivez ces étapes pour créer un script JavaScript qui affiche le texte *Hello World!* dans un document Photoshop.

Puisque vous utiliserez Photoshop pour exécuter vos scripts JavaScript, il n'est pas nécessaire d'inclure un code qui ouvre Photoshop au début du script.

REMARQUE : Adobe a créé le langage de script Extend Script pour augmenter le nombre de scripts JavaScript utilisables avec Photoshop. Vous pouvez utiliser la commande Extend Script `#target` pour cibler l'application Photoshop et créer la possibilité d'ouvrir des scripts JavaScript qui manipulent Photoshop à partir d'une zone quelconque de votre système de fichiers. Consultez le chapitre « Script UI » du *JavaScript Tools Guide (Guide des outils JavaScript)* pour plus de détails.

Pour créer et exécuter votre premier script JavaScript Photoshop :

1. Entrez le script ci-dessous.

   REMARQUE : les commentaires sont facultatifs.

```
// Hello Word Script
// Remember current unit settings and then set units to
// the value expected by this script
var originalUnit = preferences.rulerUnits
preferences.rulerUnits = Units.INCHES

// Create a new 2x4 inch document and assign it to a variable
var docRef = app.documents.add( 2, 4 )

// Create a new art layer containing text
var artLayerRef = docRef.artLayers.add()
artLayerRef.kind = LayerKind.TEXT

// Set the contents of the text layer.
var textItemRef = artLayerRef.textItem
textItemRef.contents = "Hello, World"

// Release references
docRef = null
artLayerRef = null
textItemRef = null

// Restore original ruler unit setting
app.preferences.rulerUnits = originalUnit
```

2. Enregistrez un fichier sous forme de fichier texte avec une extension de nom de fichier `.jsx` dans le dossier Paramètres prédéfinis/Scripts de votre répertoire Adobe Photoshop CS4.

   REMARQUE : vous devez placer vos scripts JavaScript dans le dossier Paramètres prédéfinis/Scripts afin de les rendre accessibles à partir du menu **Fichier > Scripts** dans Photoshop. Les scripts apparaîtront dans le menu **Fichier > Scripts** à la session suivante.

   REMARQUE : Photoshop prend également en charge les fichiers JavaScript qui portent une extension `.js`.

3. Utilisez l'une des méthodes suivantes :

   - Si Photoshop est déjà ouvert, choisissez **Fichier > Scripts > Parcourir,** naviguez ensuite vers le dossier Paramètres prédéfinis > Scripts, puis sélectionnez votre script.
   - Démarrez ou redémarrez Photoshop, choisissez ensuite **Fichier > Scripts**, puis sélectionnez votre script à partir du menu **Scripts**.

3 Scripts Photoshop

Ce chapitre présente plusieurs techniques pour utiliser le modèle d'objet de document de Photoshop (DOM) pour créer des scripts à utiliser spécifiquement avec Photoshop.

Vous apprendrez aussi à vous servir des manuels de référence et des explorateurs de modèles d'objets pour trouver des informations sur les objets, les classes, les propriétés, les commandes et même sur certaines valeurs (appelées *constantes* ou *énumérations*) que vous pouvez utiliser pour créer des scripts AppleScripts, VBScript et JavaScripts pour Photoshop.

CONSEIL : tout au long de ce chapitre, les explications sur la création de scripts sont suivies d'instructions permettant de localiser les informations relatives aux éléments spécifiques utilisés dans chaque script. L'utilisation de ces instructions vous aidera à comprendre rapidement comment créer des scripts Photoshop.

Affichage des objets, commandes et méthodes Photoshop

Les contenus de référence Photoshop pour chacun des trois langages de script figurent dans les manuels de référence fournis dans cette installation :

- *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4*
- *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4*
- *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4*

Vous pouvez également accéder à ces références en utilisant un explorateur de modèles d'objets associé à chaque langage :

- Pour AppleScript, utilisez l'éditeur de scripts d'AppleScript pour afficher le dictionnaire AppleScript de Photoshop.
- Pour VBScript, utilisez l'éditeur VBA de Microsoft Word, l'explorateur d'objets Visual Basic de Visual Basic ou Visual Studio.
- Pour JavaScript, utilisez l'afficheur de modèles d'objets ExtendScript. Consultez le *JavaScript Tools Guide (Guide des outils JavaScript)* pour plus de détails.

Affichage du dictionnaire AppleScript de Photoshop

Vous pouvez utiliser l'éditeur de scripts d'Apple pour afficher le dictionnaire.

REMARQUE : par défaut, l'éditeur de scripts est stocké dans le dossier **Applications > AppleScript > Editeur de scripts**.

Pour afficher le dictionnaire AppleScript :

1. Dans l'éditeur de scripts, choisissez la commande **Fichier > Ouvrir un dictionnaire**.

   L'éditeur de scripts affiche la boîte de dialogue correspondante.

2. Sélectionnez Adobe Photoshop CS4, puis cliquez sur **Ouvrir**.

   L'éditeur de scripts ouvre Photoshop puis affiche le dictionnaire Photoshop qui répertorie les objets ainsi que les commandes, les propriétés et les éléments associés à chaque objet. Les paramètres de chaque commande sont également répertoriés.

REMARQUE : le dictionnaire AppleScript de Photoshop n'affiche pas la liste complète de formats d'ouverture et d'enregistrement.

Affichage de la bibliothèque de types de Photoshop (VBS)

Vous pouvez utiliser l'éditeur VBA de Microsoft Word pour afficher les objets et commandes disponibles pour VBScript dans Photoshop.

Pour afficher la bibliothèque d'objets VBS dans Microsoft Word :

1. Démarrez Word, puis choisissez la commande **Outils > Macro > Visual Basic Editor**.
2. Choisissez **Outils > Références**, cochez ensuite la case Adobe Photoshop Type Library, puis cliquez sur **OK**.
3. Choisissez la commande **Affichage > Explorateur d'objets**.
4. Sélectionnez Photoshop CS4 **type library** dans la liste des bibliothèques ouvertes affichées dans le menu déroulant en haut à gauche.
5. Choisissez une classe d'objet pour afficher des informations supplémentaires sur la classe.

Vous pouvez également utiliser l'explorateur d'objets dans l'environnement de développement Visual Basic pour afficher les objets et commandes disponibles pour VBScript dans Photoshop.

Pour afficher la bibliothèque d'objets VBS dans l'environnement de développement de Visual Basic :

1. Démarrez Visual Studio 2005 ou Visual Basic.
2. Choisissez **Affichage > Explorateur d'objets.**
3. Dans la liste déroulante Parcourir, sélectionnez l'option de **modification du jeu de composants personnalisés**.
4. Sous l'onglet COM, recherchez « Adobe Photoshop CS4 Object Library », et sélectionnez-la.
5. Cliquez sur le bouton **Ajouter**. La bibliothèque sélectionnée s'affiche dans la section des projets et composants sélectionnés de la fenêtre.
6. Cliquez sur le bouton **OK**.
7. A présent, la bibliothèque Photoshop est chargée dans l'explorateur d'objets. Cliquez sur le signe plus en regard de l'icône de la bibliothèque Photoshop.
8. Cliquez sur le signe plus en regard de l'icône d'objets Photoshop.
9. Les objets définis dans la bibliothèque Photoshop sont répertoriés. Vous pouvez les sélectionner individuellement pour afficher plus de détails sur la classe.

Ciblage et référencement de l'objet Application

Puisque vous exécutez vos scripts AppleScript et VBScript en dehors de l'application Photoshop, votre script doit indiquer en premier lieu que les commandes sont exécutées dans Photoshop.

REMARQUE : en JavaScript, il n'est pas utile de cibler l'objet `Application` puisque vous ouvrez les scripts à partir de l'application Photoshop elle-même (voir la section « Création et exécution d'un script JavaScript », page 20).

AS Pour cibler Photoshop en AppleScript, vous devez encadrer votre script avec les instructions suivantes :

```
tell application "Adobe Photoshop CS4"
...
end tell
```

REMARQUE : toutes les commandes étant incluses dans le bloc `tell`, vous n'avez pas besoin de référencer l'objet `Application` dans l'ensemble du script.

VBS En VBScript, procédez comme suit pour cibler l'application :

```
Dim appRef
Set appRef = CreateObject("Photoshop.Application")
```

JS En JavaScript, l'objet `Application` n'ayant pas besoin d'être référencé, toutes les propriétés et méthodes de l'application sont accessibles sans qualification. Vous pouvez référencer l'application en tant qu'élément de la structure hiérarchique ou l'ignorer, en fonction de la clarté que procure l'une ou l'autre des solutions à vos scripts.

Pour référencer l'objet `Application`, utilisez l'objet global prédéfini `app`, à la place du nom de classe.

Les instructions suivantes sont équivalentes :

```
var docRef = app.documents[1]
```

et

```
var docRef=documents[1]
```

REMARQUE : les exemples JavaScript de ce guide ne référencent pas l'objet `Application`.

Création d'objets dans un script

Pour créer un document dans l'application Photoshop, choisissez **Fichier > Nouveau**. Pour créer d'autres types d'objets dans un document, tels qu'un calque, une couche ou un tracé, utilisez le menu Fenêtre ou choisissez l'icône *Nouveau* dans le panneau approprié. Cette section explique comment réaliser ces tâches dans un script.

Pour créer un objet dans un script, vous devez nommer le type d'objet que vous souhaitez créer, puis utiliser la commande suivante :

- AS : `make`
- VBS : `Add`
- JS : `add()`

Comme vous pouvez le constater à la section « Modèle d'objet de Photoshop », page 11, l'objet `Document` contient tous les autres objets à l'exception des objets `Application`, `Notifier` et `Preferences`

REMARQUE : dans cet exemple, l'objet `Application` est référencé via une variable nommée `appRef` (voir la section « Ciblage et référencement de l'objet Application », page 24 pour plus de détails).

Pour ajouter un objet `ArtLayer`, vous devez référencer les objets `Application` et `Document` qui contiendront le calque graphique. L'exemple suivant référence l'objet `Application` à l'aide de la variable `appRef` et l'objet `Document` à l'aide de l'index du document plutôt que du nom du document.

```
appRef.Documents(1).ArtLayers.Add()
```

REMARQUE : dans Photoshop, les collections VBScript sont indexées à partir de 1 et non de 0, ce qui signifie que le premier document créé prend l'indice 1 et non l'indice 0.

Si vous recherchez l'objet `Document` dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic, vous verrez qu'il n'y a pas de méthode `Add()` pour l'objet. La méthode `Add()` est cependant disponible pour l'objet `Documents`. De même, l'objet `ArtLayer` n'a pas de méthode `Add()` ; l'objet `ArtLayers`, si.

REMARQUE : l'objet `Layers` est une exception, car il s'agit d'un objet de collection, mais sans méthode `Add()`. La collection `Layers` inclut à la fois les objets `ArtLayer` et `LayerSet`, créés à l'aide de la méthode `Add` sur la collection `ArtLayers` ou `LayerSets`. Pour plus de détails, recherchez l'objet `Layers` dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4*.

JS En JavaScript, vous pouvez utiliser la méthode `add()` uniquement avec le nom de collection. La méthode `add()` n'est autorisée qu'avec les objets de collection.

L'instruction JavaScript pour la création d'un document est semblable à l'instruction VBScript :

```
documents.add()
```

et *non* :

```
document.add()
```

REMARQUE : vous pouvez si vous le souhaitez inclure une référence à l'objet `Application`. L'instruction suivante est équivalente à l'exemple précédent :

```
app.documents.add()
```

Pour ajouter un objet `ArtLayer`, vous devez référencer l'objet `Document` qui contient le calque et utiliser la méthode `add()` pour la collection `ArtLayers`, avec la propriété `artLayers` de l'objet `Document`.

```
documents[0].artLayers.add()
```

Comme avec VBScript, la méthode `add()` est associée à l'objet JavaScript `Documents` mais pas à l'objet `Document`. De même, l'objet `ArtLayer` n'a pas de méthode `add()` ; l'objet `ArtLayers`, si.

REMARQUE : la collection `Layers` n'inclut pas de méthode `add()`. Pour plus de détails, recherchez l'objet `Layers` dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4*.

Définition de l'objet actif

Pour travailler sur un objet dans l'application Photoshop, vous devez en faire l'objet de premier plan, ou l'objet *actif*. Par exemple, pour travailler sur un calque, vous devez commencer par l'amener au premier plan.

La même règle s'applique aux scripts. Si votre script crée deux documents ou plus, les commandes et méthodes du script sont exécutées sur le document actif. Il convient donc, pour s'assurer que les commandes portent sur le bon document, de désigner lors de la programmation le document actif avant d'exécuter des commandes ou méthodes dans le script.

Pour définir un objet actif, procédez comme suit :

- En AppleScript, utilisez la propriété `current` de l'objet parent.
- En VBScript, utilisez la propriété `Active`*`Object`* de l'objet parent (comme `ActiveDocument` ou `ActiveLayer`).
- En JavaScript, utilisez la propriété `active`*`Object`* de l'objet parent (comme `activeDocument` ou `activeLayer`).

REMARQUE : l'objet parent est celui qui contient l'objet spécifié. Par exemple, l'application est le parent du document ; un document est le parent d'un calque, d'une sélection ou d'une couche.

Par exemple, si vous recherchez l'objet `Application` dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4*, ou dans l'afficheur de modèles d'objets ExtendScript, vous trouvez que l'une de ses propriétés est `activeDocument` ; si vous recherchez l'objet `Document`, vous trouverez `activeLayer` et `activeHistoryState` comme propriétés. De la même manière, si vous recherchez `application` dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4*, ou dans le dictionnaire AppleScript de Photoshop, vous le trouvez comme propriété de `current`, etc.

Pour des exemples de scripts définissant des objets actifs, reportez-vous aux sections suivantes :

- « Définition du document actif », page 27
- « Définition du calque actif », page 28
- « Définition des couches actives », page 29

Définition du document actif

Les exemples suivants expliquent comment définir le document actif.

AS

```
--create 2 documents
    set docRef to make new document with properties ¬
        {width:4 as inches, height:4 as inches}
    set otherDocRef to make new document with properties ¬
        {width:4 as inches, height:6 as inches}

    --make docRef the active document
    set current document to docRef
    --here you would include command statements
    --that perform actions on the active document. Then, you could
    --make a different document the active document

    --use the current document property of the application class to
    --bring otherDocRef front-most as the new active document
    set current document to otherDocRef
```

VBS

```
'Create 2 documents
Set docRef = app.Documents.Add ( 4, 4)
Set otherDocRef = app.Documents.Add (4,6)

'make docRef the active document
Set app.ActiveDocument = docRef
'here you would include command statements
'that perform actions on the active document. Then, you could
'make a different document the active document

'use the ActiveDocument property of the Application object to
'bring otherDocRef front-most as the new active document
Set app.ActiveDocument = otherDocRef
```

JS

```
// Create 2 documents
var docRef = app.documents.add( 4, 4)
var otherDocRef = app.documents.add (4,6)

//make docRef the active document
app.activeDocument = docRef
//here you would include command statements
//that perform actions on the active document. Then, you could
//make a different document the active document

//use the activeDocument property of the Application object to
//bring otherDocRef front-most as the new active document
app.activeDocument = otherDocRef
```

Définition du calque actif

Les exemples suivants expliquent comment utiliser la propriété `current layer (ActiveLayer/activeLayer)` de l'objet `Document` pour définir le calque actif. Pour définir le calque actif d'un document, le document doit être actif.

AS

```
set current layer of current document to layer "Layer 1" of current document
```

REMARQUE : par défaut, Photoshop attribue les noms Calque 1, Calque 2, etc., aux calques.

VBS

```
' This example assumes appRef and docRef have been previously defined and assigned
' to the application object and a document object that contains at least one layer.
appRef.ActiveDocument = docRef
docRef.ActiveLayer = docRef.Layers(1)
```

Recherchez la propriété `ActiveLayer` sur l'objet `Document` dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic.

REMARQUE : vous pouvez également indiquer le calque à utiliser par son nom. Par défaut, Photoshop attribue les noms Calque 1, Calque 2, etc., aux calques (voir la section « Référencement des objets ArtLayer », page 40).

JS

```
// This example assumes docRef has been previously defined and assigned to a
// document object that contains at least one layer.
activeDocument = docRef
docRef.activeLayer = docRef.layers[0]
```

Recherchez la propriété `activeLayer` sur l'objet `Document` dans le *Guide de référence pour les scripts Javascript Adobe Photoshop CS4* ou dans l'afficheur de modèles d'objets ExtendScript.

REMARQUE : vous pouvez également indiquer le calque à utiliser par son nom. Par défaut, Photoshop attribue les noms Calque 1, Calque 2, etc., aux calques (voir la section « Référencement des objets ArtLayer », page 40).

Définition des couches actives

Plusieurs couches peuvent être actives simultanément, si bien que la propriété `channels` (`ActiveChannels/activeChannels`) actuelle de l'objet `Document` prend comme valeur un tableau de couches. Pour définir les couches actives d'un document, le document doit être actif.

AS Définissez comme couches actives les première et troisième couches à l'aide d'un tableau de couches :

```
set current channels of current document to ¬
    { channel 1 of current document, channel 3 of current document }
```

Vous pouvez sinon sélectionner toutes les couches de composante en utilisant la propriété `component channels` de l'objet `Document`.

```
set current channels of current document to component channels ¬
    of current document
```

VBS Définissez comme couches actives du document actif les première et troisième couches à l'aide d'un tableau de couches :

```
' This example assumes docRef is already the ActiveDocument
Dim theChannels
theChannels = Array(docRef.Channels(1), docRef.Channels(3))
docRef.ActiveChannels = theChannels
```

Vous pouvez également sélectionner toutes les couches de composante à l'aide de la propriété `ComponentChannels` de l'objet `Document`.

```
appRef.ActiveDocument.ActiveChannels= _
    appRef.ActiveDocument.ComponentChannels
```

JS Définissez comme couches actives les première et troisième couches à l'aide d'un tableau de couches :

```
theChannels = new Array(docRef.channels[0], docRef.channels[2])
docRef.activeChannels = theChannels
```

Vous pouvez également sélectionner toutes les couches de composante à l'aide de la propriété `componentChannels` de l'objet `Document`.

```
app.activeDocument.activeChannels =
    activeDocument.componentChannels
```

Ouverture d'un document

Pour ouvrir un document existant, vous devez utiliser la commande `open`/`Open`/`open()` de l'objet `Application`. Vous devez spécifier le nom du document (le chemin d'accès au fichier qui contient le document) à l'aide de la commande.

Ouverture d'un fichier avec le format de fichier par défaut

Du fait que Photoshop prend en charge plusieurs formats de fichiers différents, la commande `open`/`Open`/`open()` vous permet de spécifier le format du document que vous ouvrez. Si vous ne spécifiez pas le format, Photoshop choisit à votre place le type de fichier, qui est appelé le format de fichier par défaut. Les exemples suivants ouvrent un document en choisissant le format par défaut le mieux adapté :

AS

```
set theFile to alias "Applications:Documents:MyFile"
open theFile
```

VBS

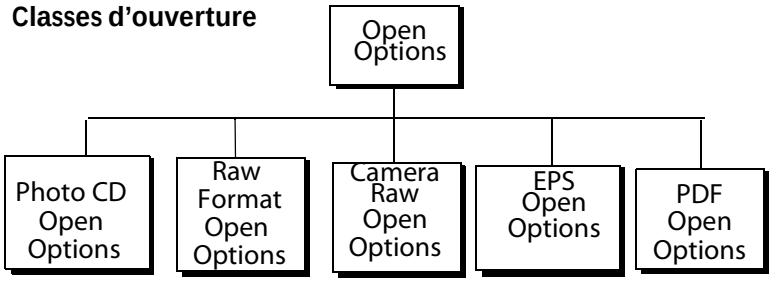
```
fileName = "C:\MyFile"
Set docRef = appRef.Open(fileName)
```

JS

```
var fileRef = File(app.path + "/Samples/Fish.psd")
var docRef = app.open(fileRef)
```

En JavaScript, vous devez créer un objet `File`, puis transmettre à la commande `open()` une référence à l'objet.

Définition des formats de fichier à ouvrir



Vous pouvez définir des options pour les types de documents de la liste suivante, afin de spécifier le mode d'ouverture du document (hauteur et largeur de la fenêtre d'ouverture du document, page à afficher à l'ouverture d'un document multipage, etc.).

- PhotoCD
- CameraRaw
- RawFormat
- Adobe PDF
- EPS

Pour prendre connaissance des options disponibles pour chaque format de fichier, recherchez les propriétés des objets *OpenOptions* commençant par le nom du format de fichier requis. Par exemple :

- Dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4*, recherchez la classe `Photo CD open options` ou la classe `EPS open objects`.
- Dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4*, recherchez les objets `PhotoCDOpenOptions` ou `EPSOpenOptions`.

Les exemples suivants montrent comment ouvrir un document PDF générique (multiplage/multi-image) avec les spécifications suivantes :

- Le document s'ouvre en mode RVB avec une résolution de 72 ppp.
- Un lissage sera appliqué afin de minimiser l'aspect crénelé du contour des images du document.
- Le document s'ouvrira à la page 3.
- La forme initiale du document sera modifiée conformément aux propriétés de hauteur et de largeur si sa largeur n'atteint pas le double de sa hauteur.

AS

```
tell application "Adobe Photoshop CS4"
    set myFilePath to alias "OS X 10.4.8 US:Users:psauto:Desktop:opal_screen.pdf"
    with timeout of 300 seconds
        open myFilePath as PDF with options ¬
            {class:PDF open options, ¬
                mode:RGB, resolution:72, use antialias:true, page:3}
    end timeout
end tell
```

VBS

```
Dim appRef
Set appRef = CreateObject("Photoshop.Application")

'Remember unit settings and set to values expected by this script
Dim originalRulerUnits
originalRulerUnits = appRef.Preferences.RulerUnits
appRef.Preferences.RulerUnits = 1 'value of 1 = psPixels

'Create a PDF option object
Dim pdfOpenOptionsRef
Set pdfOpenOptionsRef = CreateObject("Photoshop.PDFOpenOptions")
pdfOpenOptionsRef.AntiAlias = True
pdfOpenOptionsRef.Mode = 2 ' psOpenRGB
pdfOpenOptionsRef.Resolution = 72
pdfOpenOptionsRef.Page = 3

' open the file
Dim docRef
Set docRef = appRef.Open("C:\\PDFFiles\MyFile.pdf", pdfOpenOptionsRef)

'Restore unit setting
appRef.Preferences.RulerUnits = originalRulerUnits
```

JS

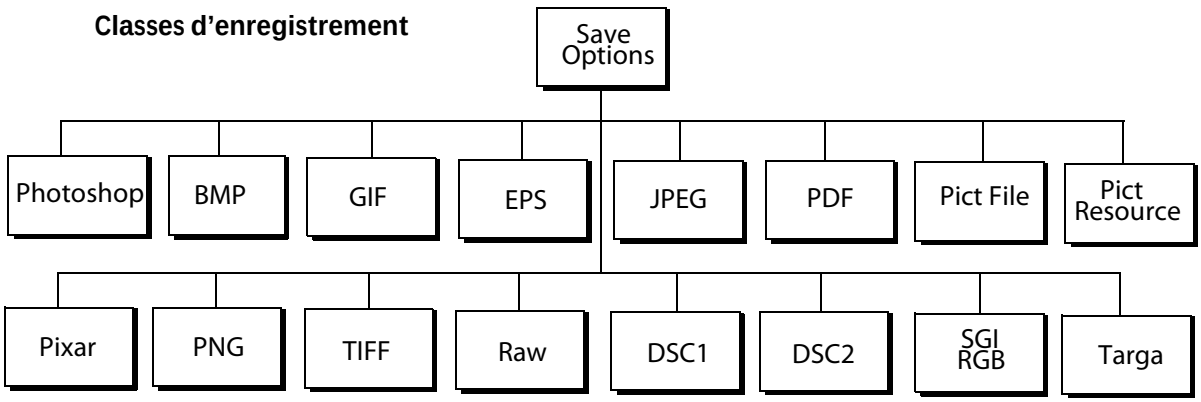
REMARQUE : l'objet `File` d'ExtendScript attend une notation URI (universal resource identifier). Reportez-vous au *JavaScript Tools Guide (Guide des outils JavaScript)* pour plus de détails.

```
// Set the ruler units to pixels
var originalRulerUnits = app.preferences.rulerUnits
app.preferences.rulerUnits = Units.PIXELS
// Get a reference to the file that we want to open
var fileRef = new File("/c/pdffiles/myfile.pdf")

// Create a PDF option object
var pdfOpenOptions = new PDFOpenOptions
pdfOpenOptions.antiAlias = true
pdfOpenOptions.mode = OpenDocumentMode.RGB
pdfOpenOptions.resolution = 72
pdfOpenOptions.page = 3
// open the file
app.open( fileRef, pdfOpenOptions )

// restore unit settings
app.preferences.rulerUnits = originalRulerUnits
```

Enregistrement d'un document

Les options d'enregistrement des documents dans Photoshop sont illustrées ci-dessous. Pour connaître les propriétés que vous pouvez spécifier pour une option d'enregistrement relative à un format de fichier particulier, recherchez l'objet qui commence par le nom du format de fichier requis. Par exemple, pour connaître les propriétés d'enregistrement d'un fichier `.eps`, procédez comme suit :

- Dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4*, recherchez la classe `EPS save options`.
- Dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4*, recherchez `EPSSaveOptions`.



REMARQUE : il est important de noter que les formats d'ouverture (`Open`) et d'enregistrement (`Save`) ne sont pas identiques (voir la section « Définition des formats de fichier à ouvrir », page 30 à titre de comparaison).

REMARQUE : les formats facultatifs suivants sont disponibles uniquement lorsqu'ils sont installés explicitement :

- Alias PIX
- Electric Image
- SGI RGB

- Wavefront RLA
- SoftImage

Les scripts suivants enregistrent les documents au format `.jpeg`.

AS

```
tell application "Adobe Photoshop CS4"
    make new document
    set myFile to "OS X 10.4.8 US:Users:psauto:Desktop:Rat.jpg"
    set myOptions to ¬
        {class:JPEG save options, embed color profile:false, ¬
            format options:standard, matte:background color matte}
    save current document in file myFile as JPEG with options ¬
        myOptions appending no extension without copying
end tell
```

VBS

```
Dim appRef,docRef
Set appRef = CreateObject("Photoshop.Application")
Set docRef = appRef.Documents.Add()

Set jpgSaveOptions = CreateObject("Photoshop.JPEGSaveOptions")
jpgSaveOptions.EmbedColorProfile = True
jpgSaveOptions.FormatOptions = 1 'for psStandardBaseline
jpgSaveOptions.Matte = 1 'for psNoMatte
jpgSaveOptions.Quality = 1
appRef.ActiveDocument.SaveAs "c:\temp\myFile2", _
    jpgSaveOptions,     True, 2 'for psLowercase
```

JS

```
app.documents.add( 4, 4 )
jpgFile = new File( "/Temp001.jpeg" )
jpgSaveOptions = new JPEGSaveOptions()
jpgSaveOptions.embedColorProfile = true
jpgSaveOptions.formatOptions = FormatOptions.STANDARDBASELINE
jpgSaveOptions.matte = MatteType.NONE
jpgSaveOptions.quality = 1
app.activeDocument.saveAs(jpgFile, jpgSaveOptions, true,
    Extension.LOWERCASE)
```

Définition des préférences de l'application

Votre script peut définir des préférences liées à l'application (sélecteur de couleurs, options d'enregistrement de fichier, paramètres des repères-grilles-tranches, etc.).

REMARQUE : les propriétés de la classe `settings` / l'objet `Preferences` correspondent aux options de la boîte de dialogue Préférences de Photoshop CS4 que vous pouvez afficher en choisissant **Photoshop > Préférences** avec la version Mac OS de Photoshop ou **Edition > Préférences** avec la version Windows. Pour obtenir une description de chaque préférence, consultez l'Aide de Photoshop.

AS Les propriétés de la classe `settings` permettent de définir les préférences de l'application dans AppleScript. Le script suivant définit l'unité utilisée pour les règles et le texte :

```
set ruler units of settings to inch units
set type units of settings to pixel units
```

Dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop, recherchez la classe `settings-object` pour afficher toutes les propriétés de paramètres que vous pouvez utiliser.

VBS L'objet `Preferences` est une propriété de l'objet `Application`. Lorsque vous utilisez l'objet `Preferences` dans un script VBScript, vous devez indiquer son appartenance à l'objet `Application`.

```
appRef.Preferences.RulerUnits = 2 'for PsUnits --> 2 (psInches)
appRef.Preferences.TypeUnits = 1 'for PsTypeUnits --> 1 (psPixels)
```

Dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic, recherchez l'objet `Preferences` pour afficher toutes les propriétés de paramètres que vous pouvez utiliser. Recherchez également la propriété `Preferences` sur l'objet `Application`.

JS L'objet `Preferences` est une propriété de l'objet `Application`.

```
preferences.rulerUnits = Units.INCHES
preferences.typeUnits = TypeUnits.PIXELS
```

Dans le *Guide de référence pour les scripts Javascript Adobe Photoshop CS4* ou dans l'afficheur de modèles d'objets ExtendScript, recherchez l'objet `Preferences` pour afficher toutes les propriétés de paramètres que vous pouvez utiliser. Recherchez également la propriété `preferences` sur l'objet `Application`.

Autorisation ou refus d'ouverture de boîtes de dialogue

Il est important de pouvoir contrôler les boîtes de dialogue correctement depuis un script. Lorsqu'une boîte de dialogue s'affiche, le script est interrompu jusqu'à ce qu'un utilisateur ferme la fenêtre. Cette solution est généralement adaptée à un script interactif pour lequel un utilisateur est assis devant l'ordinateur. S'il s'agit d'un script exécuté automatiquement (par lots), il convient d'éviter l'apparition de boîtes de dialogue afin de ne pas interrompre l'exécution du script.

Vous utilisez la propriété `display dialogs` (`DisplayDialogs/displayDialogs`) de l'objet `Application` pour contrôler l'affichage ou non des boîtes de dialogue.

REMARQUE : l'utilisation de boîtes de dialogue dans votre script équivaut à peu près à l'utilisation de points d'arrêt dans un script Photoshop.

AS Le script suivant empêche l'affichage de boîtes de dialogue :

```
set display dialogs to never
```

Dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop, recherchez la `Class application` pour trouver les valeurs que vous pouvez utiliser pour la propriété `display dialogs`.

VBS Pour définir les préférences relatives aux boîtes de dialogue, utilisez la propriété `DisplayDialogs` de l'objet `Application`.

```
appRef.DisplayDialogs = 3
'for PsDialogModes --> 3 (psDisplayNoDialogs)
```

`DisplayDialogs` étant une propriété de l'objet `Application`, vous devez référencer l'objet `Application` dans le script pour obtenir la propriété.

Dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic, recherchez la propriété `DisplayDialogs` de l'objet `Application`. Vous verrez que le type de valeur de cette propriété est la constante `PsDialogModes`. Vous pouvez également rechercher les options pour `PsDialogModes`.

JS Pour définir les préférences relatives aux boîtes de dialogue, utilisez la propriété `displayDialogs` de l'objet `Application`.

```
displayDialogs = DialogModes.NO
```

Dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4* ou dans l'afficheur de modèles d'objets ExtendScript, recherchez la propriété `displayDialogs` de l'objet `Application`, puis recherchez la constante `DialogModes`.

Utilisation du modèle d'objet de Photoshop

Cette section comporte des informations sur l'utilisation des objets du modèle d'objet de Photoshop. Pour plus de détails sur les modèles d'objet, reportez-vous au guide *Introduction to Scripting (Introduction aux scripts)* et à la section « Modèle d'objet de Photoshop », page 11.

Utilisation de l'objet Application

Cette section explique comment et quand utiliser l'objet `Application` dans un script. Elle explique également comment utiliser certaines propriétés de l'objet `Application`.

Vous utilisez les propriétés et commandes de l'objet `Application` pour travailler avec la fonctionnalité Photoshop et les objets tels que les suivants :

- **Paramètres ou préférences globaux de Photoshop, tels que les valeurs d'unité ou les paramètres de couleur** (voir la section « Définition des préférences de l'application », page 34).
- **Documents** : vous pouvez ajouter ou ouvrir des documents et définir le document actif(voir les sections « Ouverture d'un document », page 30 et « Définition de l'objet actif », page 26).
- **Actions** : vous pouvez exécuter des actions créées soit par l'écriture de scripts, soit en utilisant le panneau Scripts de l'application Photoshop (voir la section « Gestionnaire de scripts », page 77).

Vous pouvez utiliser les propriétés de l'objet `Application` pour obtenir des informations, notamment :

- Liste des polices installées sur le système.
  - AS : définissez `theFonts` sur `fonts`

    REMARQUE : en AppleScript, fonts est une collection séparée, qui ne requiert pas de référence à l'objet application pour être récupérée.
  - VBS : définissez `fontsInstalled = AppRef.Fonts`
  - JS : `var fontsInstalled = app.fonts`
- Espace de mémoire inutilisé, disponible pour Adobe Photoshop, à l'aide de la propriété `free memory (FreeMemory/freeMemory)` de l'objet `Application`.
- Emplacement du dossier Préférences, à l'aide de la propriété `preferences folder (PreferencesFolder/preferencesFolder)` de l'objet `Application`.

Pour plus de détails, recherchez les propriétés de l'objet `Application` dans le guide de référence ou dans l'explorateur d'objets correspondant au langage utilisé.

Utilisation de l'objet Document

L'objet `Document` peut représenter tout document ouvert dans Photoshop. Un objet `Document` s'apparente à un fichier, ou encore à une zone de travail. L'objet `Document` permet de réaliser les opérations suivantes :

- Accès aux objets contenus dans l'objet `Document`, par exemple `ArtLayer` ou `Channel` (voir la section « Modèle d'objet de Photoshop », page 11).
- Manipulation d'un objet `Document` spécifique à l'aide de commandes ou de méthodes. Par exemple, vous pouvez recadrer, faire pivoter ou appliquer une symétrie à la zone de travail, redimensionner l'image ou la zone de travail et rogner l'image (voir la section « Manipulation d'un objet document », page 36 pour obtenir une démonstration).
- Récupération du calque actif (voir la section « Définition du calque actif », page 28).
- Enregistrement du document actif (voir la section « Enregistrement d'un document », page 32).
- Opérations de copier-coller dans le document actif ou entre plusieurs documents (voir la section « Présentation de l'interaction avec le Presse-papiers », page 57).

Manipulation d'un objet document

Les exemples suivants illustrent les opérations suivantes :

- Enregistrement des préférences d'unité de la règle existantes et définition de l'unité de la règle en pouces
- Ouverture d'un fichier existant en tant que document (à l'aide du fichier Ducky.tif)
- Modification de la taille de l'image à 4 pouces de large sur 4 pouces de haut (10 cm x 10 cm)
- Modification de la taille de la fenêtre de document (ou zone de travail) à 4 pouces de large sur 4 pouces de haut (10 cm x 10 cm)

- Rognage du haut et du bas de l'image
- Recadrage de l'image
- Application d'une symétrie à l'ensemble de la fenêtre
- Restauration des unités de règle d'origine

REMARQUE : reportez-vous à la section « Définition des préférences de l'application », page 34 pour plus de détails sur les unités de mesure des règles.

AS

```
tell application "Adobe Photoshop CS4"
    set saveUnit to ruler units of settings
    set ruler units of settings to inch units
    set duckFile to alias ¬
        "OS X 10.4.8 US:Applications:Adobe Photoshop CS4:Samples:Ducky.tif"
    open duckFile
    set docRef to current document
    resize image docRef width 4 height 4
    resize canvas docRef width 4 height 4
    trim docRef basing trim on top left pixel with top trim ¬
        and bottom trim without left trim and right trim
    set ruler units of settings to pixel units
    crop current document bounds {100, 200, 400, 500} angle 45 width 20 height 20
    flip canvas docRef direction horizontal
    set ruler units of settings to saveUnit
end tell
```

VBS

```
Dim appRef, docRef
Set appRef = CreateObject("Photoshop.Application")

'save original ruler units, then set ruler units to inches
startRulerUnits = appRef.Preferences.RulerUnits
appRef.Preferences.RulerUnits = 2 'for PsUnits --> 2 (psInches)

Set docRef = appRef.Open(appRef.Path & "\Samples\Ducky.tif")
docRef.ResizeImage 4,4
docRef.ResizeCanvas 4,4

'Trim the document with
'  type = 1 (psTopLeftPixel)
'  top=true, left=false, bottom=true, right=false
docRef.Trim 1,True,False,True,False

'the crop command uses unit values
'so change the ruler units to pixels
appRef.Preferences.RulerUnits = 1 ' (psPixels)

'Crop the document with
'  angle=45, width=20,height=20
docRef.Crop Array(100,200,400,500),45,20,20
docRef.FlipCanvas  1 ' psHorizontal

'restore ruler units
appRef.Preferences.RulerUnits = startRulerUnits
```

JS

```
//save original ruler units, then assign it to inches
startRulerUnits = app.preferences.rulerUnits
app.preferences.rulerUnits = Units.INCHES

//get a reference to the file, and open it
var fileRef = new File(app.path + "/samples/ducky.tif")
var docRef = app.open(fileRef)

//this sample script assumes the ruler units have been set to inches
docRef.resizeImage( 4,4 )
docRef.resizeCanvas( 4,4 )
docRef.trim(TrimType.TOPLEFT, true, false, true, false)

//the crop command uses unit values
//so change the ruler units to pixels
app.preferences.rulerUnits =Units.PIXELS
docRef.crop (new Array(100,200,400,500), 45, 20, 20)
docRef.flipCanvas(Direction.HORIZONTAL)

//restore original preferences
app.preferences.rulerUnits = startRulerUnits
```

Utilisation des objets calque

Le modèle d'objet de Photoshop comporte deux types d'objets calque :

- Les objets `ArtLayer`, qui peuvent comporter des contenus d'image et qui sont pour ainsi dire équivalents aux calques dans l'application Photoshop.

  REMARQUE : un objet `ArtLayer` peut également contenir du texte si vous utilisez la propriété `kind` pour définir le type de l'objet `ArtLayer` sur calque de texte.

- Les objets `Layer Set`, qui peuvent contenir zéro ou plusieurs objets `ArtLayer`.

Lorsque vous créez un calque, vous devez spécifier si vous créez un calque graphique (`ArtLayer`) ou un groupe de calques (`Layer Set`).

REMARQUE : les objets `ArtLayer` et `LayerSet` ont tous deux des collections correspondantes, à savoir `ArtLayers` et `LayerSets`, elles-mêmes dotées d'une commande `add/Add/add()`. Vous pouvez référencer (mais pas ajouter) des objets `ArtLayer` et `LayerSet` à l'aide de la collection `Layers`, car contrairement aux autres collections, celle-ci n'est pas dotée de commande `add/Add/add()`.

Création d'un objet ArtLayer

Les exemples ci-dessous expliquent comment créer un objet `ArtLayer` rempli avec du rouge au début du document actif.

AS

```
tell application "Adobe Photoshop CS4"
    make new document
    make new art layer at beginning of current document ¬
        with properties {name:"MyBlendLayer", blend mode:normal}
    select all current document
    fill selection of current document with contents ¬
        {class:RGB color, red:255, green:0, blue:0}
end tell
```

VBS

```
Dim appRef
Set appRef = CreateObject("Photoshop.Application")

' Create a new art layer at the beginning of the current document
Dim docRef
Dim layerObj
Set docRef = appRef.Documents.Add()
Set layerObj = appRef.ActiveDocument.ArtLayers.Add
layerObj.Name = "MyBlendLayer"
layerObj.BlendMode = 2 'psNormalBlend

' Select all so we can apply a fill to the selection
appRef.ActiveDocument.Selection.SelectAll

' Create a color to be used with the fill command
Dim colorObj
Set colorObj = CreateObject("Photoshop.SolidColor")
colorObj.RGB.Red = 255
colorObj.RGB.Green = 0
colorObj.RGB.Blue = 0

' Now apply fill to the current selection
appRef.ActiveDocument.Selection.Fill colorObj
```

JS

```
//make a new document
app.documents.add()

// Create a new art layer at the beginning of the current document
var layerRef = app.activeDocument.artLayers.add()
layerRef.name = "MyBlendLayer"
layerRef.blendMode = BlendMode.NORMAL

// Select all so we can apply a fill to the selection
app.activeDocument.selection.selectAll

// Create a color to be used with the fill command
var colorRef = new solidColor
colorRef.rgb.red = 255
colorRef.rgb.green = 100
colorRef.rgb.blue = 0

// Now apply fill to the current selection
app.activeDocument.selection.fill(colorRef)
```

Création d'un objet Layer Set

Les exemples suivants montrent comment créer un objet `Layer Set` après la création d'un premier objet `ArtLayer` dans le document actif :

AS

```
tell application "Adobe Photoshop CS4"
    make new document with properties {name:"My Document"}
    make new art layer at beginning of current document
    make new layer set after layer 1 of current document
end tell
```

VBS

```
Dim appRef
Set appRef = CreateObject("Photoshop.Application")

'Make a new document and a first layer in the document
appRef.Documents.Add()
appRef.ActiveDocument.ArtLayers.Add()

' Get a reference to the first layer in the document
Dim layerRef
Set layerRef = appRef.ActiveDocument.Layers(1)

' Create a new LayerSet (it will be created at the beginning of the document)
Dim newLayerSetRef
Set newLayerSetRef = appRef.ActiveDocument.LayerSets.Add

' Move the new layer to after the first layer
newLayerSetRef.Move layerRef, 4 'psPlaceAfter
```

JS

```
// make a new document and a layer in the document
app.documents.add()
app.activeDocument.artLayers.add()

// Get a reference to the first layer in the document
var layerRef = app.activeDocument.layers[0]

// Create a new LayerSet (it will be created at the beginning of the // document)
var newLayerSetRef = app.activeDocument.layerSets.add()

// Move the new layer to after the first layer
newLayerSetRef.move(layerRef, ElementPlacement.PLACEAFTER)
```

Référencement des objets ArtLayer

Lorsque vous créez un calque dans l'application Photoshop (plutôt qu'un script), le calque est ajouté au panneau Calques, et un numéro lui est attribué. Ce nu

Utilisation des objets Layer Set

Les calques existants peuvent être organisés en groupes de calques. Les exemples suivants montrent comment créer un objet `Layer Set`, dupliquer un objet `ArtLayer` existant et déplacer ce dernier dans le groupe de calques.

AS

```
set current document to document "My Document"
set layerSetRef to make new layer set at end of current document
set newLayer to duplicate layer "Layer 1" of current document ¬
    to end of current document
move newLayer to end of layerSetRef
```

En AppleScript, vous pouvez également dupliquer un calque directement dans son groupe de destination.

```
set current document to document "My Document"
set layerSetRef to make new layer set at end of current document
duplicate layer "Layer 1" of current document to end of layerSetRef
```

VBS En VBScript, vous pouvez également dupliquer le calque en appliquant la même méthode.

```
Dim appRef, docRef
Set appRef = CreateObject("Photoshop.Application")

'Make a new document and a first layer in the document
Set docRef = appRef.Documents.Add()
appRef.ActiveDocument.ArtLayers.Add()

Set layerSetRef = docRef.LayerSets.Add()
Set layerRef = docRef.ArtLayers(1).Duplicate(layerSetRef, 2)
```

JS En JavaScript, vous pouvez placer le calque lors de la duplication.

```
// create a document and an initial layer
var docRef = app.documents.add()
docRef.artLayers.add()

var layerSetRef = docRef.layerSets.add()
var layerRef = docRef.artLayers[0].duplicate(layerSetRef,
    ElementPlacement.PLACEATEND)
```

Liaison d'objets calque

Les scripts permettent également de lier des calques entre eux et de rompre ces liens. La liaison de calques permet de déplacer ou transformer les calques en une seule instruction.

AS

```
make new art layer in current document with properties {name:"L1"}
make new art layer in current document with properties {name:"L2"}
link art layer "L1" of current document with art layer "L2" of ¬
    current document
```

Recherchez la commande `link` dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop.

VBS

```
Set layer1Ref = docRef.ArtLayers.Add()
Set layer2Ref = docRef.ArtLayers.Add()
layer1Ref.Link layer2Ref
```

Recherchez `Link` comme méthode de l'objet `ArtLayer` dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic. Recherchez également `Add` comme méthode de l'objet `ArtLayers`.

JS

```
var layerRef1 = docRef.artLayers.add()
var layerRef2 = docRef.artLayers.add()
layerRef1.link(layerRef2)
```

Recherchez `link()` comme méthode de l'objet `ArtLayer` dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4* ou dans l'afficheur de modèles d'objets ExtendScript. Recherchez également `Add()` comme méthode de l'objet `ArtLayers`.

Application de styles aux calques

REMARQUE : cette procédure correspond à faire glisser un style du panneau Styles de Photoshop vers un calque.

Votre script peut appliquer des styles à un objet `ArtLayer`. Pour appliquer un style dans un script, utilisez la commande `apply layer style/ApplyStyle/applyStyle()` (le nom du style doit être entré comme argument entre guillemets doubles droits).

REMARQUE : les noms des styles de calque tiennent compte de la distinction entre majuscules et minuscules.

Consultez l'Aide de Photoshop pour obtenir une liste de styles et pour plus de détails sur les styles et le panneau Styles.

Les exemples suivants définissent le style de calque Puzzle pour le calque nommé L1.

AS

```
apply layer style art layer "L1" of current document using "Puzzle (Image)"
```

Recherchez la commande `apply layer style` dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop.

VBS

```
docRef.ArtLayers("L1").ApplyStyle "Puzzle (Image)"
```

Recherchez `ApplyStyle` comme méthode de l'objet `ArtLayer` dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic.

JS

```
docRef.artLayers["L1"].applyStyle("Puzzle (Image)")
```

Recherchez `applyStyle()` comme méthode de l'objet `ArtLayer` dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4* ou dans l'afficheur de modèles d'objets ExtendScript.

Utilisation de l'objet Text Item

Vous pouvez changer un objet `ArtLayer` existant en un calque de texte, c'est-à-dire un objet `Text Item`, si le calque est vide. A l'inverse, vous pouvez changer un objet `Text Item` en un objet `ArtLayer`. Cette procédure inverse pixellise le texte du calque.

L'objet `Text Item` est une propriété de l'objet `ArtLayer`. Cependant, pour créer un calque de texte, vous devez créer un objet `ArtLayer` puis définir la propriété `kind/Kind/kind` du calque graphique sur `text layer (2 (psTextLayer) / LayerKind.TEXT)`.

Pour définir ou manipuler du texte dans un calque de texte, vous utilisez l'objet `text-object (TextItem/TextItem)`, contenu dans la propriété `text object/TextItem/textItem` de l'objet `ArtLayer`.

Création d'un objet Text Item

Les exemples suivants créent un objet `ArtLayer` , puis utilisent la propriété `kind` pour le convertir en calque de texte.

AS
```
make new art layer in current document with properties { kind: text layer }
```

VBS
```
set newLayerRef = docRef.ArtLayers.Add()
newLayerRef.Kind = 2
'2 indicates psTextLayer
```

JS
```
var newLayerRef = docRef.artLayers.add()
newLayerRef.kind = LayerKind.TEXT
```

Reportez-vous à la section « Modèle d'objet de Photoshop », page 11 pour plus de détails sur la relation entre les objets `ArtLayer` et `TextItem`.

Recherchez également les éléments suivants :

- Les propriétés `Kind/kind` et `TextItem/textItem` de l'objet `ArtLayer` dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4*, dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic et l'afficheur de modèles d'objets ExtendScript.
- Les propriétés `kind` et `text object` de la classe `art layer` dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop.

Définition d'un type de calque

Les exemples suivants utilisent une instruction `if` pour vérifier si un calque existant est un calque de texte.

AS
```
if (kind of layerRef is text layer) then
...
endif
```

VBS
```
If layerRef.Kind = 2 Then '2 indicates psTextLayer
...
End If
```

JS
```
if (newLayerRef.kind == LayerKind.TEXT)
{...}
```

Ajout et manipulation de texte dans un objet Text Item

Les exemples suivants ajoutent du texte et le justifient à droite dans un calque de texte.

AS

```
set layerRef to make new art layer in current document with properties¬
    {kind:text layer}
set contents of text object of layerRef to "Hello, World!"
set justification of text object of layerRef to right
```

VBS

```
Set textLayerRef = docRef.ArtLayers.Add()
textLayerRef.Kind = 2
textLayerRef.Name = "my text"

Set textItemRef = docRef.ArtLayers("my text").TextItem
textItemRef.Contents = "Hello, World!"
textItemRef.Justification = 3
'3 = psRight (for the constant value psJustification)
```

JS

```
var textLayerRef = docRef.artLayers.add()
textLayerRef.name = "my text"
textLayerRef.kind = LayerKind.TEXT

var textItemRef = docRef.artLayers["my text"].textItem
textItemRef.contents = "Hello, World!"
textItemRef.justification = Justification.RIGHT
```

REMARQUE : l'objet `text-object` (`TextItem/TextItem`) possède une propriété `kind` `Kind/kind)` qui peut être définie soit sur `point` `text (psPointText/TextType.POINTTEXT`), soit sur `paragraph` `text (psParagraphText/TextType.PARAGRAPHTEXT.`). Lorsqu'un nouvel objet `text-object` est créé, sa propriété `kind` est automatiquement définie sur `point` `text`.

Les propriétés `height`, `width` et `leading` de l'objet `text-object` sont autorisées uniquement si la propriété `kind` du texte est définie sur `paragraph text`.

Pour vous familiariser avec les objets, propriétés et commandes présentés dans les guides de référence pour les scripts, utilisez l'une des méthodes suivantes :

- Dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop, recherchez les propriétés et les méthodes `text-object`.
- Dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic, recherchez la propriété `TextItem` de l'objet `DisplayDialogs`. Pour connaître les propriétés et méthodes disponibles avec un calque de texte, recherchez l'objet `TextItem`.

Dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4* ou dans l'afficheur de modèles d'objets ExtendScript, recherchez la propriété `textItem` de l'objet `ArtLayer`. Pour connaître les propriétés et méthodes disponibles avec un calque de texte, recherchez l'objet `TextItem`.

Utilisation des objets Selection

Vous utilisez un objet `Selection` pour restreindre l'action de vos scripts à une seule section spécifique et sélectionnée de votre document ou à un calque dans un document. Par exemple, vous pouvez appliquer des effets à une sélection ou copier la sélection active dans le Presse-papiers.

L'objet `Selection` est une propriété de l'objet `Document`. Recherchez les éléments suivants pour plus de détails :

- Dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop, recherchez la commande `select`. Recherchez également la propriété `selection` de l'objet `Document` et l'objet `selection-object`.
- Dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic, recherchez `Selection` comme propriété de l'objet `Document`. Recherchez également `Select` comme méthode de l'objet `Selection`.
- Dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4* ou dans l'afficheur de modèles d'objets ExtendScript, recherchez `selection` comme propriété de l'objet `Document`. Recherchez également `select` comme méthode de l'objet `Selection`.

REMARQUE : vous ne pouvez pas créer un objet `Selection`. La propriété `selection (Selection/selection)` sur l'objet `Document` contient un objet selection préexistant pour le document. Utilisez la commande `select (Select/select)` pour spécifier la zone de sélection.

Création et définition d'une sélection

Pour créer une sélection, utilisez la commande `select/Select/select()` de l'objet `Selection`.

Pour définir une sélection, vous devez spécifier les coordonnées à l'écran qui correspondent aux angles de la sélection. Le document étant un objet en deux dimensions, les coordonnées sont spécifiées à l'aide des axes x et y, comme suit :

- L'axe x sert à spécifier la position horizontale dans la zone de travail.
- L'axe y sert à spécifier la position verticale dans la zone de travail.

L'origine dans Photoshop, c'est-à-dire la position où l'axe x = 0 et l'axe y = 0, est le coin supérieur gauche de l'écran. Le coin opposé, soit le coin inférieur droit, correspond au point extrême de la zone de travail. Par exemple, si la zone de travail mesure 1000 x 1000 pixels, les coordonnées du coin inférieur droit sont axe x = 1000 et axe y = 1000.

Vous spécifiez les points de coordonnées qui décrivent la forme à sélectionner dans un tableau, qui devient à son tour la valeur d'argument ou de paramètre de la commande `select/Select/select()`.

Les exemples suivants supposent que l'unité de mesure des règles est le pixel et que vous créez une sélection ainsi :

1. Création d'une variable pour contenir un nouveau document de 500 x 500 pixels.
2. Création d'une variable pour contenir les coordonnées de la zone sélectionnée (c.-à-d. l'objet `Selection`).
3. Ajout d'un tableau comme valeur de la variable de la sélection.

4. Utilisation de la propriété `selection` de l'objet `Document` et de la commande `select` de l'objet `Selection` pour sélectionner une zone. Les coordonnées de la zone correspondent aux valeurs de la variable de la sélection.

AS

```
set docRef to make new document with properties {height:500, width:500}
set shapeRef to {{0, 0}, {0, 100}, {100, 100}, {100, 0}}
select current document region shapeRef
```

VBS

```
DocRef = appRef.Documents.Add
ShapeRef = Array(Array(0, 0), Array(0, 100), Array(100,100), Array(100,0))
docRef.Selection.Select ShapeRef
```

JS

```
var docRef = app.documents.add(500, 500)
var shapeRef = [
    [0,0],
    [0,100],
    [100,100],
    [100,0]
]
docRef.selection.select(shapeRef)
```

Application d'un contour au cadre de sélection

Les exemples ci-dessous utilisent la commande `stroke (Stroke/stroke())` de l'objet `Selection` pour tracer les bordures autour de la sélection active et définissent la couleur ainsi que la largeur du contour.

REMARQUE : le paramètre de transparence ne peut pas être utilisé pour les calques d'arrière-plan.

AS

```
stroke selection of current document using color ¬
    {class:CMYK color, cyan:20, magenta:50, yellow:30, black:0} ¬
        width 5 location inside blend mode vivid light opacity 75 ¬
    without preserving transparency
```

VBS

```
Set strokeColor = CreateObject ("Photoshop.SolidColor")
strokeColor.CMYK.Cyan = 20
strokeColor.CMYK.Magenta = 50
strokeColor.CMYK.Yellow = 30
strokeColor.CMYK.Black = 0
appRef.ActiveDocument.Selection.Stroke strokeColor, 5, 1, 15, 75, False
```

JS

```
strokeColor = new solidColor
strokeColor.cmyk.cyan = 20
strokeColor.cmyk.magenta = 50
strokeColor.cmyk.yellow = 30
strokeColor.cmyk.black = 0

app.activeDocument.selection.stroke (strokeColor, 2,
    StrokeLocation.OUTSIDE, ColorBlendMode.VIVIDLIGHT, 75,
        false)
```

Inversion des sélections

Vous pouvez appliquer la commande `invert (Invert/invert())` de l'objet `Selection` à une sélection, afin de pouvoir travailler sur le reste du document, calque ou couche tout en protégeant la sélection.

- AS : `invert selection of current document`
- VBS : `selRef.Invert`
- JS : `selRef.Invert()`

Dilatation, contraction et contour progressif des sélections

Vous pouvez modifier la taille d'une zone sélectionnée à l'aide des commandes Dilater, Contracter et Contour progressif.

Les valeurs sont transmises dans l'unité de mesure des règles stockée dans les préférences Photoshop et peuvent être modifiées par vos scripts. Si l'unité de mesure des règles est le pixel, la valeur de la dilatation, de la contraction et du contour progressif des exemples suivants sera de cinq pixels. Reportez-vous à la section « Définition des préférences de l'application », page 34 pour savoir comment modifier l'unité de mesure des règles.

AS

```
expand selection of current document by pixels 5
contract selection of current document by pixels 5
feather selection of current document by pixels 5
```

VBS

```
Dim selRef
Set selRef = appRef.ActiveDocument.Selection

selRef.Expand 5
selRef.Contract 5
selRef.Feather 5
```

JS

```
var selRef = app.activeDocument.selection
selRef.expand( 5 )
selRef.contract( 5 )
selRef.feather( 5 )
```

Remplissage d'une sélection

Vous pouvez remplir une sélection avec une couleur ou avec un état d'historique.

Pour remplir avec une couleur :

AS

```
fill selection of current document with contents ¬
    {class:RGB color, red:255, green:0, blue:0} blend mode ¬
    vivid light opacity 25 without preserving transparency
```

VBS

```
Set fillColor = CreateObject("Photoshop.SolidColor")
fillColor.RGB.Red = 255
fillColor.RGB.Green = 0
fillColor.RGB.Blue = 0
selRef.Fill fillColor, 15, 25, False
```

JS
```
var fillColor = new SolidColor()
fillColor.rgb.red = 255
fillColor.rgb.green = 0
fillColor.rgb.blue = 0
app.activeDocument.selection.fill( fillColor, ColorBlendMode.VIVIDLIGHT,
    25, false)
```

Pour remplir la sélection courante avec le dixième élément de l'état d'historique :

REMARQUE : reportez-vous à la section « Utilisation des objets history state », page 50 pour plus de détails sur les objets `History State`.

AS
```
fill selection of current document with contents history state 10¬
    of current document
```

VBS
```
selRef.Fill docRef.HistoryStates(10)
```

JS
```
selRef.fill(app.activeDocument.historyStates[9])
```

Chargement et stockage des sélections

Vous pouvez stocker des objets `Selection` dans les objets `Channel`, ou les charger depuis ces objets. Pour stocker une sélection dans une couche, sa propriété `kind` (`Kind`/`kind`) doit être définie sur un type qui indique que la couche contient une zone sélectionnée : `selected area channel` (`psSelectedAreaAlphaChannel`)/ `ChannelType.SELECTEDAREA`). Les exemples ci-dessous utilisent la commande `store (Store/store())` de l'objet `Selection` pour stocker la sélection active dans une couche nommée `My Channel` et dilater la sélection avec une sélection quelconque qui se trouve actuellement dans cette couche.

AS
```
set myChannel to make new channel of current document with properties ¬
    {name:"My Channel", kind::selected area channel}
store selection of current document into channel ¬
       "My Channel" of current document combination type extended
```

VBS
```
Set chanRef = docRef.Channels.Add
chanRef.Name = "My Channel"
chanRef.Kind = 3 'psSelectedAreaAlphaChannel

docRef.Selection.Store docRef.Channels("My Channel"), 2
'PsSelectionType is 2 (psExtendSelection)
```

JS
```
var chanRef = docRef.channels.add()
chanRef.name = "My Channel"
chanRef.kind = ChannelType.SELECTEDAREA

docRef.selection.store(docRef.channels["My Channel"], SelectionType.EXTEND)
```

Pour restaurer une sélection enregistrée dans un objet `Channel`, utilisez la méthode `load` (`Load/load`).

AS
```
set myChannel to make new channel of current document with properties ¬
    {name:"My Channel"}
load selection of current document from channel "My Channel" of ¬
    current document combination type extended
```

VBS
```
selRef.Load docRef.Channels("My Channel"), 2
'PsSelectionType is 2 (psExtendSelection)
```

JS
```
selRef.load (docRef.channels["My Channel"], SelectionType.EXTEND)
```

Reportez-vous à la section « Présentation de l'interaction avec le Presse-papiers », page 57 pour savoir comment copier, couper et coller des sélections.

Utilisation des objets Channel

L'objet `Channel` vous permet d'accéder à la plupart des fonctions disponibles dans les couches Photoshop. Vous pouvez créer, supprimer et dupliquer des couches ou récupérer l'histogramme d'une couche et modifier son type. Reportez-vous à la section « Création d'objets dans un script », page 24 pour plus de détails sur la création d'un objet `Channel` dans les scripts.

Utilisez la propriété `kind` pour définir ou connaître le type d'un objet `Channel`. Reportez-vous à la section « Chargement et stockage des sélections », page 48 pour obtenir des exemples de script illustrant la création d'une couche de zone masquée.

Changement des types de couches

Vous pouvez modifier le `type` de toutes les couches, à l'exception des couches de composante. Les exemples suivants montrent comment modifier une couche de zone masquée en couche de zone sélectionnée :

REMARQUE : les couches de composante sont liées au mode de document. Consultez l'Aide de Photoshop pour plus de détails sur les couches, les types de couches et les modes de document.

AS
```
set kind of myChannel to selected area channel
```

VBS
```
channelRef.ind = 3 'for psSelectedAreaAlphaChannel
'from the constant value PsChannelType
```

JS
```
channelRef.kind = ChannelType.SELECTEDAREA
```

Utilisation de l'objet Document Info

Dans Photoshop, vous pouvez associer des informations à un document en choisissant **Fichier > Informations**.

Pour effectuer cette tâche dans un script, vous devez utiliser l'objet `info-object` (`DocumentInfo/DocumentInfo`), stocké dans la propriété `info` (`Info/info`) de l'objet `Document`. Les exemples suivants montrent comment utiliser l'objet `DocumentInfo` pour définir l'état du copyright et l'URL propriétaire d'un document.

AS
```
set docInfoRef to info of current document
get EXIF of docInfoRef
set copyrighted of docInfoRef to copyrighted work
set owner url of docInfoRef to "http://www.adobe.com"
get EXIF of docInfoRef
```

VBS
```
Set docInfoRef = docRef.Info
docInfoRef.Copyrighted = 1 'for psCopyrightedWork
docInfoRef.OwnerUrl = "http://www.adobe.com"
```

JS
```
docInfoRef = docRef.info
docInfoRef.copyrighted = CopyrightedType.COPYRIGHTEDWORK
docInfoRef.ownerUrl = "http://www.adobe.com"
```

Recherchez les éléments suivants pour plus de détails sur les autres types d'informations (propriétés) que vous pouvez associer à un document :

- Dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop, recherchez les propriétés et méthodes pour la classe `info-object`.
- Dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4*, le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4*, l'explorateur d'objets Visual Basic ou l'afficheur de modèles d'objets ExtendScript, recherchez les propriétés pour l'objet `DocumentInfo`.

Utilisation des objets history state

Photoshop conserve un historique des actions qui concernent les documents. Chaque fois que vous apportez un changement à une image dans l'application Photoshop, vous créez un *état d'historique* ; vous pouvez accéder aux états d'historique d'un document à partir du panneau Historique en choisissant **Fenêtre > Historique**. Pour plus de détails sur l'état d'historique, consultez l'Aide de Photoshop.

Dans un script, vous pouvez accéder à l'état d'historique d'un objet `Document` à l'aide de l'objet `HistoryStates`, qui est une propriété de l'objet `Document`. Vous pouvez utiliser un objet `HistoryStates` pour rétablir l'état précédent d'un document ou pour remplir un objet `Selection`.

Les exemples suivants rétablissent la forme et les propriétés initiales (au premier enregistrement) du document contenu dans la variable `docRef`. Cette utilisation de l'état d'historique permet d'annuler des modifications apportées à un document.

AS
```
set current history state of current document to history state 1 ¬
   of current document
```

VBS
```
docRef.ActiveHistoryState = docRef.HistoryStates(1)
```

JS
```
docRef.activeHistoryState = docRef.historyStates[0]
```

REMARQUE : le rétablissement d'un état d'historique ne supprime pas les derniers états de la collection de l'historique. Utilisez la commande `Purge` pour supprimer les derniers états de la collection `History States`, comme indiqué ci-dessous :

- AS : `purge history caches`
- VBS : `appRef.Purge(2) 'for psPurgeTarget --> 2 (psHistoryCaches)`
- JS : `app.purge(PurgeTarget.HISTORYCACHES)`

L'exemple ci-dessous enregistre l'état actuel, applique un filtre et rétablit ensuite l'état d'historique enregistré.

AS
```
set savedState to current history state of current document
filter current layer of current document using motion blur with options ¬
   {class:motion blur, angle:20, radius:20}
set current history state of current document to savedState
```

VBS
```
Set savedState = docRef.ActiveHistoryState
docRef.ArtLayers(1).ApplyMotionBlur 20, 20
docRef.ActiveHistoryState = savedState
```

JS
```
savedState = docRef.activeHistoryState
docRef.artLayers[0].applyMotionBlur( 20, 20 )
docRef.activeHistoryState = savedState
```

Utilisation des objets Notifier

L'objet `Notifier` sert à lier un événement à un script. Par exemple, si vous souhaitez que Photoshop crée automatiquement un document lorsque vous ouvrez l'application, vous pouvez lier un script qui crée un objet `Document` à un événement `Open Application`.

REMARQUE : ce type de script équivaut à sélectionner *Lancer l'application* dans le Gestionnaire d'événements de script (**Fichier > Scripts > Gestionnaire d'événements de script**) de l'application Photoshop. Pour plus de détails sur l'utilisation du Gestionnaire d'événements de script, consultez l'Aide de Photoshop.

La commande `make` (`Add/add`) requiert la spécification d'un ID d'événement afin d'identifier l'événement pour lequel définir la notification. Plusieurs ID d'événements sont répertoriés dans une annexe du *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4*, du *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* et du *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4*. Certains événements agissent également sur plusieurs types d'objets, de même que la commande `make` (`Add/add`) requiert un argument supplémentaire pour un ID de classe, qui permet d'identifier l'objet. Par exemple, la commande New (Nouveau) s'applique aux objets Document, Art Layer et Channel.

REMARQUE : vous pouvez déterminer les ID d'événement et de classe de tout événement enregistrable à l'aide de l'écouteur de scripts (voir la section « Utilisation de l'écouteur de scripts pour rechercher des ID d'événements et de classes », page 85).

Les exemples suivants montrent comment définir une notification d'événement pour un événement d'ouverture de document. Le script vérifie d'abord que l'option de notification d'événement est activée, puis définit l'événement sur le déclenchement de l'exécution du fichier `Bienvenue.jsx`). A la fin du script, chaque fois que vous ouvrez un document en dehors d'un script, la notification se déclenche, et exécute le fichier `.jsx`. Ce fichier `.jsx` affiche un message d'alerte.

REMARQUE : en règle générale, la notification ne s'applique pas aux événements à l'intérieur d'un script, car ces événements sont incorporés dans un événement `AdobeScriptAutomation Scripts`.

AS

```
tell application "Adobe Photoshop CS4"
    try
        delete notifiers
    end try
    make new notifier with properties {event:"Opn ", ¬
          event file:alias "OS X 10.4.8 US:Users:psauto:Desktop:Welcome.jsx"}
end tell
```

VBS

```
Dim appRef,eventFile
Set appRef = CreateObject("Photoshop.Application")

appRef.NotifiersEnabled = True
eventFile = appRef.Path & "Presets\Scripts\Event Scripts Only\Welcome.jsx"
appRef.Notifiers.Add "Opn ", eventFile
```

JS

```
app.notifiersEnabled = true
var eventFile = new File(app.path +
                         "/Presets/Scripts/Event Scripts Only/Welcome.jsx")
app.notifiers.add("Opn ", eventFile)
```

Utilisation de l'objet PathItem

Pour créer un objet `PathItem`, vous devez ajouter un objet `PathItem` à l'élément ou à la collection `PathItems` d'un document. Pour cela, vous devez créer un ensemble d'objets `PathPointInfo`, qui spécifient les coordonnées des coins ou des points d'ancrage du tracé. Puis, vous créez un ensemble d'objets `SubPathInfo` destinés à contenir les tableaux `PathPoint`. Une fois que les points et la portion de tracé ont été créés, vous pouvez ajouter un nouvel objet `PathItem`.

Le script suivant crée un objet `PathItem` en forme de ligne droite.

AS

```
--line #1--it's a straight line so the coordinates for anchor, left, and
--right for each point have the same coordinates
tell application "Adobe Photoshop CS4"
    set ruler units of settings to pixel units
    set type units of settings to pixel units
    set docRef to make new document with properties {height:700, width:500, ¬
        name:"Snow Cone"}

    set pathPointInfo1 to {class:path point info, kind:corner point, ¬
        anchor:{100, 100}, left direction:{100, 100}, right direction:{100, 100}}
    set pathPointInfo2 to {class:path point info, kind:corner point, ¬
        anchor:{150, 200}, left direction:{150, 200}, right direction:{150, 200}}
    set subPathInfo1 to ¬
        {class:sub path info, ¬
        entire sub path:{pathPointInfo1, pathPointInfo2}, ¬
        operation:shape xor, closed:false}

    set newPathItem to make new path item in docRef with properties ¬
        {entire path:{subPathInfo1}, name:"Line", kind:normal}
end tell
```

VBS

```
Dim appRef, docRef
Dim lineArray(1), lineArray2(1), lineSubPathArray(0), myPathItem
Set appRef = CreateObject("Photoshop.Application")

' create a document to work with
Set docRef = appRef.Documents.Add(5000, 7000, 72, "Simple Line")


'line #1--it's a straight line so the coordinates for anchor, left, and
'right for each point have the same coordinates
'First create the array of PathPointInfo objects. The line has two points,
'so there are two PathPointInfo objects.
Set lineArray(0) = CreateObject("Photoshop.PathPointInfo")
lineArray(0).Kind = 2 ' for PsPointKind --> 2 (psCornerPoint)
lineArray(0).Anchor = Array(100, 100)
lineArray(0).LeftDirection = lineArray(0).Anchor
lineArray(0).RightDirection = lineArray(0).Anchor
Set lineArray(1) = CreateObject("Photoshop.PathPointInfo")
lineArray(1).Kind = 2
lineArray(1).Anchor = Array(150, 200)
lineArray(1).LeftDirection = lineArray(1).Anchor
lineArray(1).RightDirection = lineArray(1).Anchor
```

```
'Next create a SubPathInfo object, which will hold the line array
'in its EntireSubPath property.
Set lineSubPathArray(0) = CreateObject("Photoshop.SubPathInfo")
lineSubPathArray(0).Operation = 2 'for PsShapeOperation --> 2 (psShapeXOR)
lineSubPathArray(0).Closed = false
lineSubPathArray(0).EntireSubPath = lineArray

'create the PathItem object using Add. This method takes the SubPathInfo object
'and returns a PathItem object, which is added to the pathItems collection
'for the document.
Set myPathItem = docRef.PathItems.Add("A Line", lineSubPathArray)

' stroke it so we can see something
myPathItem.StrokePath(2) 'for PsToolType --> 2 (psBrush)
```

JS

```
// create a document to work with
var docRef = app.documents.add(5000, 7000, 72, "Simple Line")

//line #1--it's a straight line so the coordinates for anchor, left, and //right
//for each point have the same coordinates
// First create the array of PathPointInfo objects. The line has two points,
// so there are two PathPointInfo objects.
var lineArray = new Array()
    lineArray[0] = new PathPointInfo
    lineArray[0].kind = PointKind.CORNERPOINT
    lineArray[0].anchor = Array(100, 100)
    lineArray[0].leftDirection = lineArray[0].anchor
    lineArray[0].rightDirection = lineArray[0].anchor
    lineArray[1] = new PathPointInfo
    lineArray[1].kind = PointKind.CORNERPOINT
    lineArray[1].anchor = Array(150, 200)
    lineArray[1].leftDirection = lineArray[1].anchor
    lineArray[1].rightDirection = lineArray[1].anchor

// Next create a SubPathInfo object, which holds the line array
// in its entireSubPath property.
var lineSubPathArray = new Array()
    lineSubPathArray[0] = new SubPathInfo()
    lineSubPathArray[0].operation = ShapeOperation.SHAPEXOR
    lineSubPathArray[0].closed = false
    lineSubPathArray[0].entireSubPath = lineArray

//create the path item, using add. This method takes the SubPathInfo object
//and returns a PathItem object, which is added to the pathItems collection
// for the document.
var myPathItem = docRef.pathItems.add("A Line", lineSubPathArray);

// stroke it so we can see something
myPathItem.strokePath(ToolType.BRUSH)
```

Utilisation des objets de couleur

Vos scripts peuvent utiliser la même gamme de couleurs que celle disponible via l'interface utilisateur de Photoshop. Chaque modèle colorimétrique possède son propre ensemble de propriétés. Par exemple, la classe `RGB color (RGBColor/RGBColor)` contient trois propriétés : red (rouge), blue (bleu) et green (vert). Pour définir une couleur dans cette classe, vous devez indiquer les valeurs de chacune des trois propriétés.

En VBScript et JavaScript, la classe `SolidColor` contient une propriété pour chaque modèle colorimétrique. Pour utiliser cet objet, vous devez dans un premier temps créer une instance d'un objet `SolidColor`, puis définir les propriétés de modèle colorimétrique appropriées pour l'objet. Une fois qu'un modèle colorimétrique a été attribué à un objet `SolidColor`, ce dernier ne peut plus être attribué à un autre modèle colorimétrique.

Les exemples suivants montrent comment définir une couleur à l'aide de la classe `CMYK color`.

AS

```
set foreground color to {class:CMYK color, cyan:20.0,¬
    magenta:90.0, yellow:50.0, black:50.0}
```

VBS

```
'create a solidColor array
Dim solidColorRef
Set solidColorRef = CreateObject("Photoshop.SolidColor")
solidColorRef.CMYK.Cyan = 20
solidColorRef.CMYK.Magenta = 90
solidColorRef.CMYK.Yellow = 50
solidColorRef.CMYK.Black = 50
appRef.ForegroundColor = solidColorRef
```

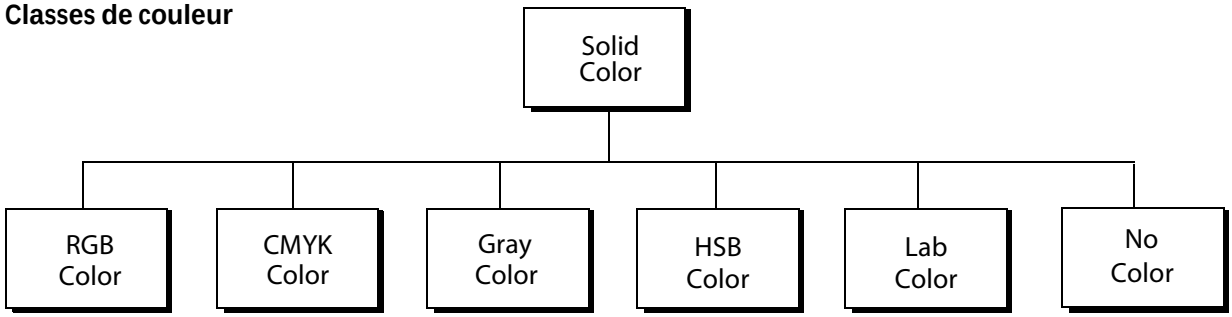
JS

```
//create a solid color array
var solidColorRef = new solidColor()
solidColorRef.cmyk.cyan = 20
solidColorRef.cmyk.magenta = 90
solidColorRef.cmyk.yellow = 50
solidColorRef.cmyk.black = 50
foregroundColor = solidColorRef
```

Classes de couleur unie

Les classes de couleur unie disponibles dans Photoshop sont exposées ci-dessous.

Classes de couleur



Utilisation des valeurs hexadécimales pour les couleurs RVB

Vous pouvez exprimer des valeurs RVB comme valeurs *hexadécimales*. Une valeur hexadécimale contient trois paires de chiffres qui représentent le rouge, le bleu et le vert (dans cet ordre).

En AppleScript, la valeur hexadécimale est représentée par la propriété de chaîne `hex value` dans la classe `RGB hex color`, et vous pouvez récupérer cette valeur hexadécimale à l'aide de la commande `convert color` décrite ci-dessous.

En VBScript et JavaScript, la propriété de l'objet `RGBColor` se nomme `HexValue/hexValue`.

Obtention et conversion de couleurs

Les exemples suivants convertissent une couleur RVB en son équivalent CMJN.

AS Le script suivant, qui suppose un modèle colorimétrique RVB, récupère la couleur de premier plan, puis utilise la commande `convert` de la classe `color` pour convertir la couleur en son équivalent CMJN.

```
get foreground color
convert color foreground color to CMYK
```

Recherchez les éléments suivants dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop.

- Propriété `foreground color` de la classe `application` (section Objets)
- Propriété `convert` (section Commandes)

VBS Le script suivant utilise une instruction `If Then` et la propriété `model` de l'objet `SolidColor` pour déterminer le modèle de couleur utilisé. L'instruction `If Then` renvoie un objet `SolidColor` ; si elle renvoie un objet `RGB`, la propriété `cmyk` de l'objet `SolidColor` vous permet alors d'accéder à la couleur avec son équivalent CMJN.

```
Dim someColor
If (someColor.model = 2) Then
    someColor.cmyk
    'someColor.model = 2 indicates psColorModel --> 2 (psRGBModel)
End If
```

Recherchez les éléments suivants dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic :

- `model` et `cmyk` comme propriétés de l'objet `SolidColor`

JS Cet exemple utilise la propriété `foregroundColor` de l'objet `Application` pour obtenir la couleur d'origine à convertir. La propriété `cmyk` de l'objet `SolidColor` à laquelle la propriété `foregroundColor` fait référence fournit une méthode d'accès à l'équivalent CMJN de la couleur RVB.

```
var someColor = foregroundColor.cmyk
```

Recherchez les éléments suivants dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4* ou dans l'afficheur de modèles d'objets ExtendScript.

- `cmyk` comme propriété de l'objet `SolidColor`
- `foregroundColor` comme propriété de l'objet `Application`

Comparaison des couleurs

L'utilisation de la commande `equal colors (IsEqual/isEqual)` vous permet de comparer des couleurs. Les instructions suivantes renvoient la valeur `true` si la couleur de premier plan est visuellement identique à la couleur d'arrière-plan.

- AS : `if equal colors foreground color with background color then`
- VBS : `If (appRef.ForegroundColor.IsEqual(appRef.BackgroundColor)) Then`
- JS : `if (app.foregroundColor.isEqual(backgroundColor))`

Obtention d'une couleur Web sécurisée

Pour convertir une couleur en une couleur Web sécurisée, utilisez la commande `web safe color` d'AppleScript et la propriété `NearestWebColor/nearestWebColor` de l'objet `SolidColor` pour VBScript et JavaScript.

AS
```
set myWebSafeColor to web safe color for foreground color
```

VBS
```
Dim myWebSafeColor
Set myWebSafeColor = appRef.ForegroundColor.NearestWebColor
```

JS
```
var webSafeColor = new RGBColor()
webSafeColor = app.foregroundColor.nearestWebColor
```

Utilisation de filtres

Pour appliquer un filtre en AppleScript, vous utilisez la commande `filter` avec une option de la classe `filter options`. En VBScript et JavaScript, vous devez utiliser une méthode de filtrage spécifique. Par exemple, pour appliquer un filtre Flou gaussien, vous devez utiliser la méthode `ApplyGaussianBlur/applyGaussianBlur()`. Toutes les méthodes de filtrage appartiennent à l'objet `ArtLayer`.

REMARQUE : pour plus de détails sur les effets produits par des types de filtre individuels, consultez l'Aide de Photoshop.

Les exemples suivants appliquent le filtre Flou gaussien au calque actif :

AS Utilisez la commande `filter`, puis spécifiez le calque et le nom du filtre (et les options, le cas échéant).

```
filter current layer of current document using gaussian blur ¬
   with options {radius:5}
```

REMARQUE : dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop, recherchez la commande `filter` ainsi que la classe `filter options`.

VBS
```
appRef.docRef.ActiveLayer.ApplyGaussianBlur 5
```

REMARQUE : dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic, recherchez la méthode `ApplyGaussianBlur` et d'autres méthodes de l'objet `ArtLayer` dont le nom commence par « *Apply* ».

JS
```
docRef.activeLayer.applyGaussianBlur(5)
```

REMARQUE : dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4* ou dans l'afficheur de modèles d'objets ExtendScript, recherchez la méthode `applyGaussianBlur()` et d'autres méthodes de l'objet `artLayer` dont le nom commence par « *apply* ».

Filtres divers

Si le type de filtre à utiliser sur le calque ne fait pas partie de l'interface de scripts, vous pouvez utiliser le gestionnaire de scripts à partir d'un script JavaScript pour exécuter un filtre. Si vous utilisez AppleScript ou VBScript, vous pouvez exécuter le script JavaScript depuis votre script. Pour plus de détails sur l'utilisation du gestionnaire de scripts, reportez-vous à la section « Gestionnaire de scripts », page 77, ainsi qu'à la section « Exécution de scripts JavaScript à partir d'AS ou de VBS », page 11.

Présentation de l'interaction avec le Presse-papiers

Les commandes du Presse-papiers dans Photoshop agissent sur les objets `ArtLayer`, `Selection` et `Document`. Les commandes peuvent servir à manipuler les objets d'un seul document ou à déplacer des informations d'un document dans un autre.

Les commandes de Presse-papiers des objets `layer (ArtLayer/ArtLayer)` et `selection (Selection/Selection)` sont les suivantes :

- `copy (Copy/copy)`
- `copy merged (Copy Merge parameter value/copy (merge parameter value))`
- `cut (Cut/cut)`

Les commandes de Presse-papiers de l'objet `document/Document/Document` sont les suivantes :

- `paste (Paste/paste)`
- `paste into (Paste IntoSelection parameter value/paste (intoSelection parameter value))`

REMARQUE : pour plus de détails sur les fonctions Copier, Copier avec fusion, Coller, Coller dedans et Couper, consultez l'Aide de Photoshop.

Utilisation des commandes copy et paste

Dans les exemples suivants, le contenu et le calque d'arrière-plan sont copiés dans le Presse-papiers, un document est créé, puis le contenu du Presse-papiers est collé dans ce nouveau document. Les scripts considèrent qu'un document est déjà ouvert dans Photoshop et que le document possède un calque d'arrière-plan.

REMARQUE : si le script crée un document dans lequel vous collez le contenu du Presse-papiers, assurez-vous que le document utilise la même unité de mesure que celle du document d'origine (voir la section « Définition des préférences de l'application », page 34).

AS **REMARQUE :** sous Mac OS, Photoshop doit être l'application de premier plan lors de l'exécution de ces commandes. Vous devez utiliser la commande `activate` pour activer l'application avant d'exécuter les commandes du Presse-papiers.

```
tell application "Adobe Photoshop CS4"
    activate
    select all of current document
    copy
    set current layer of current document to layer "Background" ¬
        of current document
    set newDocRef to make new document
    paste newDocRef
end tell
```

VBS
```
'make firstDocument the active document
Set docRef = appRef.ActiveDocument
docRef.ArtLayers("Background").Copy

Set newDocRef = appRef.Documents.Add(8, 6, 72, "New Doc")
newDocRef.Paste
```

JS
```
//make firstDocument the active document
var docRef = app.activeDocument
docRef.artLayers["Background"].copy()

var newDocRef = app.documents.add(8, 6, 72, "New Doc")
newDocRef.paste()
```

Utilisation de la commande/méthode copy merged

Vous pouvez également réaliser une copie avec fusion pour copier l'ensemble des calques visibles dans la zone sélectionnée. En AppleScript, vous utilisez la commande `copy merged`. Pour VBScript et JavaScript, vous utilisez la commande `Copy/copy`, en transmettant une valeur de `True/true` pour le paramètre facultatif `merge`.

AS **REMARQUE :** sous Mac OS, Photoshop doit être l'application de premier plan lors de l'exécution de ces commandes. Vous devez utiliser la commande `activate` pour activer l'application avant d'exécuter les commandes du Presse-papiers.

```
set docRef to make new document
make new art layer of docRef
select all of docRef
copy merged selection of docRef
```

VBS
```
docRef.Selection.Copy True
```

Recherchez la méthode `Copy` pour les objets `ArtLayer` et `Selection` dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic.

JS
```
docRef.selection.copy(true)
```

Recherchez la méthode `Copy()` pour les objets `ArtLayer` et `Selection` dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4* ou dans l'afficheur de modèles d'objets ExtendScript.

Utilisation des unités

Photoshop fournit deux règles pour les documents. Vous pouvez définir les unités de mesure pour les règles dans votre script à l'aide de propriétés sur l'objet `settings-object` (`Preferences/Preferences`). Les règles sont les suivantes :

- Une règle graphique, utilisée pour la plupart des mesures ou opérations liées aux présentations graphiques dans un document dont la hauteur, la largeur et la position sont spécifiées.

  Vous définissez des types d'unité de mesure pour la règle graphique à l'aide de la propriété `ruler units (RulerUnits/rulerUnits)`.

- Une règle texte, qui devient active lors de l'utilisation de l'outil Texte.

  Vous définissez des types d'unité de mesure pour la règle de texte à l'aide de la propriété `type units (TypeUnits/typeUnits)`.

REMARQUE : ces paramètres correspondent à ceux figurant dans la boîte de dialogue Préférences de Photoshop sous **Photoshop > Préférences > Unités et règles** sous Mac OS ou **Edition > Préférences > Unités et règles** sous Windows.

Valeurs d'unité

Tous les langages prennent en charge les chiffres entiers pour les valeurs d'unité. Ces valeurs sont traitées comme étant du type actuellement spécifié pour la règle appropriée.

Par exemple, si l'unité de la règle est définie en pouces, l'instruction VBScript suivante définit la taille d'un document sur 3 x 3 pouces :

```
docRef.ResizeImage 3,3
```

Si l'unité de la règle est définie sur pixels, le document sera de 3 x 3 pixels. Assurez-vous de définir l'unité de la règle en fonction de votre script, afin d'obtenir les résultats escomptés lors de l'exécution du script. Une fois le script exécuté, vous devez rétablir les paramètres d'origine de la règle s'ils ont été modifiés dans le script. Reportez-vous à la section « Définition des unités de règle et de texte dans un script », page 62 pour obtenir des explications sur la définition des unités.

Pour plus de détails sur les types de valeurs d'unité disponibles, consultez l'Aide de Photoshop.

Types de valeurs d'unité spéciaux

Les valeurs d'unité utilisées par Photoshop sont des unités de longueur, représentant des valeurs de mesure linéaire. Les pixels et les pourcentages sont également pris en charge. Même si ces deux types de valeurs d'unité ne sont pas, à proprement parler, des valeurs de longueur, ils sont inclus car ils sont énormément utilisés par Photoshop pour de multiples opérations et valeurs.

Précisions sur l'unité en AppleScript

AppleScript fournit une méthode supplémentaire pour l'utilisation des valeurs d'unité. Vous pouvez fournir des valeurs avec un type d'unité explicite lorsque des valeurs d'unité sont utilisées. Lorsqu'une valeur avec type est fournie, son type remplace les paramètres actuels de la règle.

Par exemple, pour créer un document de 4 pouces de large sur 5 pouces de haut, vous pouvez écrire :

```
make new document with properties {width:inches 4, ¬
    height:inches 5}
```

La valeur renvoyée pour une propriété Photoshop qui utilise des unités est renvoyée comme valeur du type de règle sélectionné. Obtention de la hauteur du document créé ci-dessus :

```
set docHeight to height of current document
```

renvoie une valeur de 5, ce qui représente 5 pouces d'après les paramètres actuels de la règle.

En AppleScript, vous pouvez également demander le renvoi d'une valeur de propriété exprimée dans un type d'unité particulier.

```
set docHeight to height of current document as points
```

Cette instruction renvoie une valeur de 360 (5 pouces x 72 ppp).

Les types d'unités `points` et `picas` sont des points PostScript, de 72 ppp. Les types d'unité `traditional points` et `traditional picas` reposent sur des valeurs classiques de définition de texte, de 72,27 points par pouce.

Vous pouvez convertir, ou faire passer, une valeur d'un type à un autre. Par exemple, le script suivant convertit une valeur en points en une valeur en pouces.

```
set pointValue to 72 as points
set inchValue to pointValue as inches
```

Lorsque ce script est exécuté, la variable `inchValue` contient 1 pouce, soit 72 points convertis en pouces. Cette capacité de conversion est intégrée au langage AppleScript.

REMARQUE : les valeurs `cm units` et `mm units` ne peuvent pas être utilisées de cette manière avec une référence correspondante en `cm` ou `mm`. Ces types d'unités ne sont pas reconnus par la terminologie AppleScript.

Utilisation des valeurs d'unité dans des calculs

Pour utiliser une valeur d'unité dans un calcul en AppleScript, vous devez d'abord convertir la valeur en nombre (une valeur d'unité ne peut pas être utilisée telle quelle dans les calculs). Pour multiplier une valeur en pouces, écrivez :

```
set newValue to (inchValue as number) * 5
```

REMARQUE : en AppleScript, vous pouvez récupérer et définir des valeurs en pixels ou en pourcentage, tout comme vous le faites avec les autres types d'unités. Cependant, vous ne pouvez pas convertir une valeur en pixels ou en pourcentage dans une autre unité de longueur comme vous le feriez avec tout autre type de valeur de longueur. Ainsi, l'exécution du script suivant entraînerait une erreur :

```
set pixelValue to 72 as pixels
-- Next line will result in a coercion error when run
set inchValue to pixelValue as inches
```

REMARQUE : du fait que Photoshop est une application orientée pixel, il se peut que vous ne puissiez pas retrouver toujours la même valeur que celle que vous transmettez lors de la définition d'une valeur. Par exemple, si `ruler units` est défini sur les unités mm et que vous créez un document de 30 x 30, la valeur retournée pour la hauteur ou la largeur sera 29.99 si la résolution de votre document est définie sur 72 ppp. L'interface de scripts suppose que les paramètres sont mesurés en ppp.

Utilisation de la valeur d'unité

Les tableaux suivants répertorient les propriétés des classes et objets définis pour utiliser des valeurs d'unité. Sauf indication contraire dans le tableau, les valeurs d'unité de ces propriétés sont basées sur la définition de la règle graphique.

Pour utiliser ce tableau, utilisez l'une des méthodes suivantes :

- Recherchez les propriétés de la classe dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop.
- Recherchez la propriété de l'objet dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4*, le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4*, l'explorateur d'objets Visual Basic ou l'afficheur de modèles d'objets ExtendScript.

Classe/Objet	Propriétés d'AppleScript	Propriétés de VBScript	Propriétés de JavaScript
Document	height width	Height Width	height width
EPS open options	height width	Height Width	height width
PDF open options	height width	Height Width	height width
lens flare open options	height width	Height Width	height width
offset filter	horizontal offset vertical offset	HorizontalOffset VerticalOffset	horizontalOffset verticalOffset
Text Item	baseline shift* first line indent* height hyphenation zone* leading* left indent* position right indent* space before* space after* width	BaselineShift* FirstLineIndent* Height HyphenationZone* Leading* LeftIndent* Position RightIndent* SpaceBefore* SpaceAfter* Width	baselineShift* firstLineIndent* height hyphenationZone* leading* leftIndent* position rightIndent* spaceBefore* spaceAfter* width

* Valeurs d'unité basées sur la définition de la règle texte.

Le tableau suivant répertorie les commandes qui utilisent des valeurs d'unité comme paramètres ou arguments. Dans certains cas, les paramètres sont obligatoires. Les méthodes VBScript et JavaScript sont précédées de l'objet auquel elles appartiennent.

Pour utiliser ce tableau :

- Pour les commandes AppleScript, recherchez la commande dans le chapitre « Commandes » du *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4*, ou utilisez le dictionnaire AppleScript de Photoshop.
- Pour les méthodes VBScript, recherchez la méthode dans le tableau Méthodes de l'objet dans le chapitre « Interface » du *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4*, ou utilisez l'explorateur d'objets Visual Basic.
- Pour les méthodes JavaScript, recherchez la méthode dans le tableau Méthodes de l'objet dans le chapitre « Référence des objets » du *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4*, ou utilisez l'afficheur de modèles d'objets ExtendScript.

AppleScript	VBScript	JavaScript
`crop (bounds, height, width)`	`Document.Crop (Bounds, Height, Width)`	`document.crop (bounds, height, width)`
`resize canvas (height, width)`	`Document.ResizeCanvas (Height, Width)`	`document.resizeCanvas (height, width)`
`resize image (height, width)`	`Document.ResizeImage (Height, Width)`	`document.resizeImage (height, width)`
`contract (by)`	`Selection.Contract (By)`	`selection.contract (by)`
`expand (by)`	`Selection.Expand (By)`	`selection.expand (by)`
`feather (by)`	`Selection.Feather (By)`	`selection.feather (by)`
`select border (width)`	`Selection.SelectBorder (Width)`	`selection.selectBorder (width)`
`translate (delta x, delta y)`	`Selection.Translate (DeltaX, DeltaY)`	`selection.translate (deltaX, deltaY)`
`translate boundary (delta x, delta y)`	`Selection.TranslateBoundary (DeltaX, DeltaY)`	`selection.translateBoundary (deltaX, deltaY)`

Définition des unités de règle et de texte dans un script

Les paramètres de type d'unité des deux règles Photoshop définissent combien de numéros sont interprétés en présence de propriétés et de paramètres qui prennent en charge des valeurs d'unité. Veillez à définir les unités de règle au début du script et à enregistrer et rétablir les paramètres d'origine une fois le script exécuté.

AS En AppleScript, `ruler units` et `type units` sont des propriétés de `settings-object`, accessibles via la propriété `settings` de l'objet Application, comme illustré ci-dessous.

```
set ruler units of settings to inch units
set type units of settings to pixel units
set point size of settings to postscript size
```

VBS En VBScript, `RulerUnits` et `TypeUnits` sont des propriétés de l'objet `Preferences`, accessibles via la propriété `Preferences` de l'objet `Application`, comme illustré ci-dessous.

```
    appRef.Preferences.RulerUnits = 2 'for PsUnits --> 1 (psInches)
    appRef.Preferences.TypeUnits = 1 'for PsTypeUnits --> 1 (psPixels)
appRef.Preferences.PointSize = 2
'2 indicates psPointType --> 2 (PsPostScriptPoints)
```

JS En JavaScript, `rulerUnits` et `typeUnits` sont des propriétés de l'objet `Preferences`, accessibles via la propriété `Preferences` de l'objet `Application`, comme illustré ci-dessous.

```
app.preferences.rulerUnits = Units.INCHES
app.preferences.typeUnits = TypeUnits.PIXELS
app.preferences.pointSize = PointType.POSTSCRIPT
```

REMARQUE : pensez à rétablir les paramètres d'unité initiaux lorsque le script a été exécuté. Reportez-vous à la section « Utilisation des préférences de document », page 64 pour obtenir un exemple de réalisation.

Exemples de scripts JavaScript pour l'automatisation d'un flux de production

L'exemple ci-dessous de scripts JavaScript pour l'automatisation d'un flux de production est fourni avec Photoshop et explique divers types d'utilisation des scripts. Ces scripts sont situés dans le dossier `Paramètres prédéfinis/Scripts` du répertoire de l'application. Reportez-vous à la section « Création et exécution d'un script JavaScript », page 20 pour plus de détails sur le dossier `Paramètres prédéfinis/Scripts`.

Nom du script	Description
`Layer Comps to Files.jsx`	Enregistre les compositions de calque sous forme de fichiers.
`Layer Comps to PDF.jsx`	Enregistre les compositions de calque sous forme de présentation PDF.
`Layer Comps to WPG.jsx`	Enregistre les compositions de calque sous forme de galerie de photos Web.
`Export Layers to Files.jsx`	Exporte chaque calque du document dans un fichier séparé.
`Script Events Manager.jsx`	Active et désactive les objets de notification.
`Image Processor.jsx`	Traite les images Camera Raw dans divers formats de fichiers.
`Load Files into Stack.jsx`	Charge les fichiers séparés dans une pile d'images dans un seul document.
`Merge to HDR.jsx`	Combine plusieurs images d'une même scène ou image, en capturant la plage dynamique d'une scène en une image HDR (High Dynamic Range).

Programmation avancée de scripts

Cette section présente comment utiliser les informations contenues dans les sections précédentes de ce chapitre pour créer des scripts réalisant les tâches suivantes :

- Configuration des préférences d'un document.
- Application de couleur au texte. Dans cette section, vous apprendrez également à effectuer les opérations suivantes :
  - Création d'une référence à un document existant.
  - Création d'un objet calque et conversion en calque de texte.
- Pixellisez le texte de sorte que les options *Onde* et *Flou* soient appliquées aux mots. Dans ces sections, vous apprendrez aussi à :
  - Sélection et utilisation d'une zone spécifique d'un calque via la création d'un objet sélection.
  - Application de filtres Onde et Flou directionnel à un texte sélectionné.

REMARQUE : enregistrez le script que vous avez créé à la fin de chaque leçon des sections suivantes ; chaque leçon se sert du script créé dans la leçon précédente.

Utilisation des préférences de document

Les exemples de scripts de cette section activent un objet Photoshop `Application` puis enregistrent les paramètres de configuration par défaut dans des variables de manière à ce qu'ils puissent être restaurés ultérieurement lorsque le script est achevé. Il s'agit des configurations par défaut que vous avez certainement définies dans la boîte de dialogue Préférences lorsque vous avez installé et configuré Photoshop pour la première fois.

REMARQUE : pour afficher ou définir les préférences, choisissez la commande **Photoshop > Préférences > Unités et règles** (Mac OS) ou **Edition > Préférences > Unités et règles** (Windows).

Ensuite, les scripts définissent les préférences suivantes avec les valeurs ci-après :

Préférence	Valeur	Description
rulers	inches	Utilise les pouces comme unité de mesure pour les graphiques.
units	pixels	Utilise les pixels comme unité de mesure pour le texte.
dialog modes	never	Supprime l'utilisation des boîtes de dialogue afin que les scripts soient exécutés sans intervention de l'utilisateur (comme le clic sur le bouton OK) lors des différentes étapes de l'exécution. **REMARQUE :** les modes de boîte de dialogue ne sont pas une option dans l'application Photoshop.

Ensuite, des variables sont déclarées pour stocker les dimensions du document en pouces et la résolution du document en pixels. Puis, une résolution d'affichage est déclarée et le texte « Hello, World! » est attribué à une variable de type chaîne.

Ensuite, une instruction `if` vérifie si un objet `Document` a été créé, puis crée un objet `Document` s'il n'en existe aucun.

Enfin, les préférences d'origine sont rétablies.

AS Pour utiliser les préférences de document :

1. Créez et exécutez le script suivant (voir la section « Création et exécution d'un script AppleScript », page 19 pour plus de détails).

```
tell application "Adobe Photoshop CS4"
    --make Photoshop CS4 the active (front-most) application
    activate

    --create variables for the default settings
    set theStartRulerUnits to ruler units of settings
    set theStartTypeUnits to type units of settings
    set theStartDisplayDialogs to display dialogs

    --change the settings
    set ruler units of settings to inch units
    set type units of settings to pixel units
    set display dialogs to never

    --create variables for default document settings
    set theDocWidthInInches to 4
    set theDocHeightInInches to 2
    set theDocResolution to 72
    set theDocString to "Hello, World!"

    --check to see whether any documents are open
    --if none are found, create a document
    --use the default document settings as its properties
    if (count of documents) is 0 then
        make new document with properties ¬
            {width:theDocWidthInInches, height:theDocHeightInInches,¬
            resolution:theDocResolution, name:theDocString}
    end if

    --change the settings back to the original units stored in the variables
    set ruler units of settings to theStartRulerUnits
    set type units of settings to theStartTypeUnits
    set display dialogs to theStartDisplayDialogs
end tell
```

2. Dans Photoshop, choisissez **Photoshop > Préférences > Unités et règles** pour vérifier que les paramètres d'origine de vos préférences ont été rétablis.

3. Après l'affichage du document dans Photoshop, fermez le document sans l'enregistrer.

4. Enregistrez le script sous le nom `HelloWorldDoc`.

VBS Pour utiliser les préférences de document :

1. Créez le script suivant (voir la section « Création et exécution d'un script VBScript », page 20 pour plus de détails).

```
'create variables for default preferences, new preferences
Dim startRulerUnits
    Dim startTypeUnits
    Dim docWidthInInches
    Dim docHeightInInches
    Dim resolution
    Dim helloWorldStr
    Dim appRef

    Set appRef = CreateObject("Photoshop.Application")

'assign default preferences to save values in variables
    startRulerUnits = appRef.Preferences.RulerUnits
    startTypeUnits = appRef.Preferences.TypeUnits
    startDisplayDialogs = appRef.DisplayDialogs

'set new preferences and document defaults
    appRef.Preferences.RulerUnits = 2 'for PsUnits --> 2 (psInches)
    appRef.Preferences.TypeUnits = 1 'for PsTypeUnits --> 1 (psPixels)
    appRef.DisplayDialogs = 3 'for PsDialogModes --> 3 (psDisplayNoDialogs)
docWidthInInches = 4
    docHeightInInches = 2
    resolution = 72
    helloWorldStr = "Hello, World!"

'see if any documents are open
'if none, create one using document defaults
If appRef.Documents.Count = 0 Then
appRef.Documents.Add docWidthInInches, docHeightInInches, resolution,
helloWorldStr
    End If

'restore beginning preferences
appRef.Preferences.RulerUnits = startRulerUnits
appRef.Preferences.TypeUnits = startTypeUnits
appRef.DisplayDialogs = startDisplayDialogs
Double click the file name in Windows Explorer to run the script.
```

2. Dans Photoshop, choisissez **Edition > Préférences > Unités et règles** pour vérifier que les paramètres d'origine de vos préférences ont été rétablis.

3. Après l'affichage du document dans Photoshop, fermez le document sans l'enregistrer.

4. Nommez le script `HelloWorldDoc` et enregistrez-le.

JS Pour utiliser les préférences de document :

1. Créez le script suivant.

   REMARQUE : reportez-vous à la section « Création et exécution d'un script JavaScript », page 20 pour plus de détails sur la création d'un script JavaScript.

```
//create and assign variables for default preferences
startRulerUnits = app.preferences.rulerUnits
startTypeUnits = app.preferences.typeUnits
startDisplayDialogs = app.displayDialogs

//change settings
app.preferences.rulerUnits = Units.INCHES
app.preferences.typeUnits = TypeUnits.PIXELS
app.displayDialogs = DialogModes.NO

//create and assign variables for document settings
docWidthInInches = 4
docHeightInInches = 2
resolution = 72
docName = "Hello World"

//use the length property of the documents object to
//find out if any documents are open
//if none are found, add a document
if (app.documents.length == 0)
    app.documents.add(docWidthInInches, docHeightInInches, resolution, docName)

//restore beginning preferences
app.preferences.rulerunits = startRulerUnits
app.preferences.typeunits = startTypeUnits
app.displayDialogs = startDisplayDialogs
```

2. Nommez le script `HelloWorldDoc.jsx` et enregistrez-le dans le dossier Paramètres prédéfinis/Scripts.
3. Ouvrez Photoshop, puis choisissez **Fichier > Scripts > HelloWorldDoc** pour exécuter le script.
4. Choisissez la commande **Edition > Préférences > Unités et règles** pour vérifier que les paramètres d'origine des préférences ont bien été rétablis.
5. Après l'affichage du document dans Photoshop, fermez le document sans l'enregistrer.
6. Enregistrez le script.

Application de couleur à un élément de texte

Dans cette section, nous ajoutons un calque au script `HelloWorldDoc`, puis nous changeons le calque en objet texte affichant le texte *Hello*, *World!* en rouge.

Avant de commencer, procédez comme suit :

- Assurez-vous que Photoshop est fermé.
- Ouvrez le fichier `HelloWorldDoc` dans votre éditeur de scripts.

AS Pour créer et spécifier des détails dans un texte :

1. Entrez le code suivant dans le script `HelloWorldDoc` immédiatement avant les instructions (à la fin du fichier) qui rétablissent les préférences d'origine.

```
--create a variable named theDocRef
--assign the current (active) document to it
set theDocRef to the current document

--create a variable that contains a color object of the RGB color class
--whose color is red
set theTextColor to {class:RGB color, red:255, green:0, blue:0}

--create a variable for the text layer, create the layer as an art layer object
--and use the kind property of the art layer object to make it a text layer
set theTextLayer to make new art layer in theDocRef with ¬
   properties {kind:text layer}

--Set the contents, size, position and color of the text layer
set contents of text object of theTextLayer to "Hello, World!"
set size of text object of theTextLayer to 36
set position of text object of theTextLayer to {0.75 as inches, 1 as inches}
set stroke color of text object of theTextLayer to theTextColor
```

2. Exécutez le script complet. Patientez pendant que Photoshop exécute vos commandes une par une.

3. Après l'affichage du document dans Photoshop, fermez le document sans l'enregistrer.

REMARQUE : recherchez les classes suivantes dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop pour vous assurer que vous comprenez comment vous les avez utilisées dans ce script :

- Classe `RGB color`
- Classe `art layer`

VBS Pour créer et spécifier des détails dans un texte :

1. Entrez le code suivant dans le script `HelloWorldDoc` immédiatement avant les instructions (à la fin du fichier) qui rétablissent les préférences d'origine.

```
'create a reference to the active (current) document
Set docRef = appRef.ActiveDocument

' create a variable named textColor
'create a SolidColor object whose color is red
'assign the object to textColor
Set textColor = CreateObject ("Photoshop.SolidColor")
textColor.RGB.Red = 255
textColor.RGB.Green = 0
textColor.RGB.Blue = 0

'create an art layer object using the
'Add method of the ArtLayers class
'assign the layer to the variable newTextLayer
Set newTextLayer = docRef.ArtLayers.Add()

'use the Kind property of the Art Layers class to
'make the layer a text layer
newTextLayer.Kind = 2
newTextLayer.TextItem.Contents = helloWorldStr
newTextLayer.TextItem.Position = Array(0.75, 1)
newTextLayer.TextItem.Size = 36
newTextLayer.TextItem.Color = textColor
```

2. Exécutez le script complet. Patientez pendant que Photoshop exécute vos commandes une par une.

3. Après l'affichage du document dans Photoshop, fermez le document sans l'enregistrer.

REMARQUE : recherchez les classes suivantes dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic pour vous assurer que vous comprenez comment vous les avez utilisées dans ce script :

- `SolidColor`
- `ArtLayer`

JS Pour créer et spécifier des détails dans un texte :

1. Entrez le code suivant dans le script `HelloWorldDoc` immédiatement avant les instructions (à la fin du fichier) qui rétablissent les préférences d'origine.

```
//create a reference to the active document
docRef = app.activeDocument

//create a variable named textColor
//create a SolidColor object whose color is red
//assign the object to textColor
textColor = new solidColor
textColor.rgb.red = 255
textColor.rgb.green = 0
textColor.rgb.blue = 0

helloWorldText = "Hello, World!"

//create a variable named newTextLayer
//use the add() method of the artLayers class to create a layer object
//assign the object to newTextLayer
newTextLayer = docRef.artLayers.add()

//use the kind property of the artLayer class to make the layer a text layer
newTextLayer.kind = LayerKind.TEXT

newTextLayer.textItem.contents = helloWorldText
newTextLayer.textItem.position = Array(0.75, 1)
newTextLayer.textItem.size = 36
newTextLayer.textItem.color = textColor
```

2. Enregistrez le script, ouvrez ensuite Photoshop, puis sélectionnez le script dans le menu Scripts (choisissez **Fichier > Script > HelloWorldDoc**). Patientez pendant que Photoshop exécute vos commandes une par une.

3. Après avoir visualisé le document dans Photoshop, fermez l'application sans enregistrer le document.

REMARQUE : recherchez les classes suivantes dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4* ou dans l'afficheur de modèles d'objets ExtendScript pour vous assurer que vous comprenez comment vous les avez utilisées dans ce script :

- `SolidColor`
- `ArtLayer`. Vous pouvez constater que la valeur `LayerKind.TEXT` de la propriété kind utilise la constante `LayerKind`. Les constantes sont toujours représentées en majuscules dans les scripts JavaScript Photoshop.

Application d'un filtre Onde

Dans cette section, nous allons appliquer un filtre Onde au texte *Hello* du document. Cette opération implique les étapes suivantes :

➤ Définissez la largeur et la hauteur du document en pixels, puis pixellisez l'objet texte dans le calque de texte.

REMARQUE : le texte étant un graphique vectoriel auquel il est impossible d'appliquer un filtre Onde, il convient dans un premier temps de convertir l'image en bitmap. La pixellisation convertit les images vectorielles définies mathématiquement en pixels. Pour plus de détails sur la pixellisation, consultez l'Aide de Photoshop.

➤ Sélectionnez la zone du calque à laquelle appliquer le filtre Onde.

REMARQUE : reportez-vous à la section « Définition de la zone d'un objet de sélection », page 71 pour vous familiariser avec le code du script effectuant cette tâche.

➤ Appliquez un filtre Onde à la sélection.

REMARQUE : l'onde est une courbe sinusoïdale tronquée.

Définition de la zone d'un objet de sélection

Pour définir la zone d'un objet de sélection, nous allons créer un tableau de coordonnées ou points spécifiés en pixels dans le document. Le tableau indique les coordonnées qui définissent les coins extérieurs d'une zone rectangulaire, commençant dans le coin supérieur gauche du document, et s'étendant jusqu'à sa moitié.

REMARQUE : vous pouvez définir autant de points que vous le souhaitez pour une zone sélectionnée. Le nombre de coordonnées détermine la forme de la sélection. La dernière coordonnée définie doit être la même que la première de sorte à obtenir une sélection fermée.

REMARQUE : reportez-vous à la section « Modèle d'objet de Photoshop », page 11 pour plus de détails sur les objets de sélection et sur d'autres objets Photoshop.

Les valeurs du tableau sont, dans l'ordre :

➤ Angle supérieur gauche de la sélection : `0,0`

  ➢ `0` indique la colonne la plus à gauche du document.

  ➢ `0` indique la rangée du haut du document.

➤ Angle supérieur droit de la sélection : `theDocWidthInPixels / 2, 0`

  ➢ `theDocWidthInPixels / 2` indique la colonne au milieu du document, c'est-à-dire la colonne dont la coordonnée est le nombre total de colonnes dans le document divisé par 2.

  REMARQUE : la valeur de `theDocWidthInPixels` équivaut au nombre total de pixels qui définissent la dimension horizontale du document. Les colonnes sont agencées horizontalement.

  ➢ `0` indique la rangée du haut du document.

- Angle inférieur droit : `theDocWidthInPixels / 2, theDocHeightInPixels`
  - `theDocWidthInPixels / 2` indique le milieu du document.
  - `theDocHeightInPixels` indique la rangée du bas du document, c'est-à-dire la rangée dont la coordonnée est le nombre total de rangées dans le document.

  REMARQUE : la valeur de `theDocHeightInPixels` équivaut au nombre total de pixels qui déterminent la dimension verticale du document. Les rangées sont agencées verticalement.
- Coin inférieur gauche : `0, theDocHeightInPixels`
  - `0` indique la colonne la plus à gauche dans le document.
  - `theDocHeightInPixels` indique la rangée du bas dans le document.
- Angle supérieur gauche de la sélection : `0,0`
  - Ferme le tracé de sélection sur son point de départ.

AS Pour sélectionner une zone et lui appliquer un filtre Onde :

1. Entrez le code suivant dans le fichier de script `HelloWorldDoc` immédiatement avant les instructions rétablissant les préférences d'origine :

```
--create new variables to contain the document object's width and height
--determine width and height values by multiplying the
--width and height in inches by the resolution
--(which equals the number of pixels per inch)
set theDocWidthInPixels to theDocWidthInInches * theDocResolution
set theDocHeightInPixels to theDocHeightInInches * theDocResolution

--use the rasterize command of the art layer object
rasterize theTextLayer affecting text contents

--create a variable named theSelRegion
--assign an array of coordinates as its value
set theSelRegion to {{0, 0}, ¬
   {theDocWidthInPixels / 2, 0}, ¬
   {theDocWidthInPixels / 2, theDocHeightInPixels}, ¬
   {0, theDocHeightInPixels}, ¬
   {0, 0}}

--replace the document object with the selection object
--so that the wave is applied only to the selected text
select theDocRef region theSelRegion combination type replaced

--apply the wave filter using the filter command of the
--wave filter class (inherited from the filter options super class)
filter current layer of theDocRef using wave filter ¬
   with options {class:wave filter, number of generators:1, minimum wavelength:1,¬
      maximum wavelength:100, minimum amplitude:5, maximum amplitude:10, ¬
      horizontal scale:100, vertical scale:100, wave type:sine,¬
      undefined areas:repeat edge pixels, random seed:0}
```

2. Choisissez le bouton **Exécuter** pour exécuter le script.
3. Après l'affichage du document dans Photoshop, fermez le document sans l'enregistrer.
4. Enregistrez le script dans l'éditeur de scripts.

REMARQUE : recherchez les classes suivantes dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop pour vous assurer que vous comprenez comment vous les avez utilisées dans ce script :

- Classe `wave filter`
- Classe `art layer` : commande `rasterize`, commande `filter`
- Classe `document` : commande `select`, paramètre `combination type`

VBS Pour sélectionner une zone et lui appliquer un filtre Onde :

1. Entrez le code suivant dans le fichier de script `HelloWorldDoc` immédiatement avant les instructions (à la fin du fichier) qui rétablissent les préférences d'origine :

```
'create new variables to contain doc width and height
'convert inches to pixels by multiplying the number of inches by
'the resolution (which equals number of pixels per inch)
docWidthInPixels = docWidthInInches * resolution
docHeightInPixels = docHeightInInches * resolution

'use the Rasterize() method of the ArtLayer class to
'convert the text in the ArtLayer object (contained in the newTextLayer variable)
'to postscript text type
newTextLayer.Rasterize (1)

'create an array to define the selection property
'of the Document object
'define the selected area as an array of points in the document
docRef.Selection.Select Array(Array(0, 0), _
Array(docWidthInPixels / 2, 0), _
Array(docWidthInPixels / 2, docHeightInPixels), _
Array(0, docHeightInPixels), Array(0, 0))

'use the ApplyWave() method of the ArtLayer class
'to apply the wave of the selected text
newTextLayer.ApplyWave 1, 1, 100, 5, 10, 100, 100, 1, 1, 0
```

2. Cliquez deux fois sur le nom du fichier dans l'Explorateur de Windows pour exécuter le script.
3. Après l'affichage du document dans Photoshop, fermez le document sans l'enregistrer.
4. Enregistrez le script.

REMARQUE : recherchez les classes suivantes dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic pour vous assurer que vous comprenez comment vous les avez utilisées dans ce script :

- Classe `ArtLayer` : méthode `ApplyWave`, méthode `Rasterize`
- Classe `Selection` : méthode `Select`

JS Pour sélectionner une zone et lui appliquer un filtre Onde :

1. Entrez le code suivant dans le fichier de script `HelloWorldDoc` immédiatement avant les instructions rétablissant les préférences d'origine :

```
//create new variables to contain doc width and height
//convert inches to pixels by multiplying the number of inches by
//the resolution (which equals number of pixels per inch)
docWidthInPixels = docWidthInInches * resolution
docHeightInPixels = docHeightInInches * resolution
//use the rasterize method of the artLayer class
newTextLayer.rasterize(RasterizeType.TEXTCONTENTS)

//create a variable to contain the coordinate values
//for the selection object
selRegion = Array(Array(0, 0),
Array(docWidthInPixels / 2, 0),
Array(docWidthInPixels / 2, docHeightInPixels),
Array(0, docHeightInPixels),
Array(0, 0))

//use the select method of the selection object
//to create an object and give it the selRegion values
//as coordinates
docRef.selection.select(selRegion)

newTextLayer.applyWave(1, 1, 100, 5, 10, 100, 100,
WaveType.SINE, UndefinedAreas.WRAPAROUND, 0)
```

2. Enregistrez le script, ouvrez ensuite Photoshop, puis sélectionnez le script dans le menu Scripts (choisissez **Fichier > Script > HelloWorldDoc**).

3. Après avoir visualisé le document dans Photoshop, fermez l'application sans enregistrer le document.

REMARQUE : recherchez les classes suivantes dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4* ou dans l'afficheur de modèles d'objets ExtendScript pour vous assurer que vous comprenez comment vous les avez utilisées dans ce script :

- `ArtLayer`
  - Méthode `rasterize()`. Vous remarquerez que l'argument `RasterizeType.TEXTCONTENTS` utilise la constante `RasterizeType`. Les constantes sont toujours représentées en majuscules dans les scripts JavaScript Photoshop.
  - Méthode `applyWave()`

Application d'un filtre Flou directionnel

Dans cette section, nous appliquons un filtre différent à la seconde moitié du document.

AS Pour appliquer un filtre Flou directionnel à HelloWorldDoc :

1. Entrez le code suivant dans le fichier de script `HelloWorldDoc` immédiatement avant les instructions rétablissant les préférences d'origine :

```
--change the value of the variable theSelRegion
--to contain the opposite half of the screen
set theSelRegion to {{theDocWidthInPixels / 2, 0},¬
   {theDocWidthInPixels, 0}, ¬
   {theDocWidthInPixels, theDocHeightInPixels}, ¬
   {theDocWidthInPixels / 2, theDocHeightInPixels}, ¬
   {theDocWidthInPixels / 2, 0}}

select theDocRef region theSelRegion combination type replaced
filter current layer of theDocRef using motion blur ¬
   with options {class:motion blur, angle:45, radius:5}
deselect theDocRef
```

2. Choisissez le bouton **Exécuter** pour exécuter le script.

REMARQUE : recherchez la classe `motion blur` dans le *Guide de référence pour les scripts AppleScript Adobe Photoshop CS4* ou dans le dictionnaire AppleScript de Photoshop pour vous assurer que vous comprenez comment vous l'avez utilisée dans ce script :

VBS Pour appliquer un filtre Flou directionnel à HelloWorldDoc :

1. Entrez le code suivant dans le fichier de script `HelloWorldDoc` immédiatement avant les instructions rétablissant les préférences d'origine :

```
docRef.Selection.Select Array(Array(docWidthInPixels / 2, 0), _
Array(docWidthInPixels, 0), _
Array(docWidthInPixels, docHeightInPixels), _
Array(docWidthInPixels / 2, docHeightInPixels), _
Array(docWidthInPixels / 2, 0))

newTextLayer.ApplyMotionBlur 45, 5

docRef.Selection.Deselect
```

2. Cliquez deux fois sur ce fichier dans l'Explorateur de Windows pour exécuter le script.

REMARQUE : recherchez la classe `ArtLayer` : méthode `ApplyMotionBlur` dans le *Guide de référence pour les scripts Visual Basic Adobe Photoshop CS4* ou dans l'explorateur d'objets Visual Basic pour vous assurer que vous comprenez comment vous l'avez utilisée dans ce script.

JS Pour appliquer un filtre Flou directionnel à HelloWorldDoc :

1. Entrez le code suivant dans le fichier de script `HelloWorldDoc` immédiatement avant les instructions rétablissant les préférences d'origine :

```
//change the value of selRegion to the other half of the document
selRegion = Array(Array(docWidthInPixels / 2, 0),
Array(docWidthInPixels, 0),
Array(docWidthInPixels, docHeightInPixels),
Array(docWidthInPixels / 2, docHeightInPixels),
Array(docWidthInPixels / 2, 0))

docRef.selection.select(selRegion)

newTextLayer.applyMotionBlur(45, 5)

docRef.selection.deselect()
```

2. Enregistrez le script, ouvrez ensuite Photoshop, puis sélectionnez le script dans le menu Scripts (choisissez **Fichier > Script > HelloWorldDoc**).

REMARQUE : recherchez la classe `ArtLayer` méthode `applyMotionBlur()` dans le *Guide de référence pour les scripts JavaScript Adobe Photoshop CS4* ou dans l'afficheur de modèles d'objets ExtendScript pour vous assurer que vous comprenez comment vous l'avez utilisée dans ce script.

4 Gestionnaire de scripts

Les scripts Photoshop vous permettent de gagner du temps en automatisant des tâches répétitives. Vous pouvez créer et exécuter des scripts dans l'interface de l'application à l'aide du panneau Scripts.

Il est également possible de gérer des actions dans des scripts à l'aide d'un utilitaire appelé *Gestionnaire de scripts*. Le gestionnaire de scripts vous permet d'écrire des scripts qui ciblent la fonctionnalité Photoshop sinon inaccessible dans l'interface de scripts, tels que des modules externes et filtres tiers. Pour pouvoir utiliser le gestionnaire de scripts, la tâche à laquelle vous souhaitez accéder via le gestionnaire de scripts doit être enregistrable.

Ce chapitre explique comment utiliser le gestionnaire de scripts et les objets d'interface de scripts qui lui sont associés.

Module externe Ecouteur de scripts

Avant d'utiliser le gestionnaire de scripts, vous devez installer le module externe Ecouteur de scripts. L'écouteur de scripts enregistre un fichier qui contient du code correspondant aux actions effectuées dans l'interface utilisateur.

CONSEIL : étant donné que l'écouteur de scripts enregistre la plupart de vos actions, installez-le uniquement lorsque vous créez le gestionnaire de scripts. Le fait de laisser ScriptListener installé en permanence engendre non seulement la création de fichiers volumineux qui occupent la mémoire de votre disque dur mais peut aussi ralentir les performances de Photoshop.

Lorsque vous effectuez des tâches ou des séries de tâches dans Photoshop, ScriptListener crée plusieurs fichiers qui contiennent un code représentant les actions effectuées dans Photoshop :

- `ScriptingListenerJS.log` contenant le code JavaScript
- `ScriptingListenerVB.log` contenant le code VBScript (Windows seulement)

L'écouteur de scripts crée ces fichiers sur le bureau.

REMARQUE : il n'existe pas d'interface AppleScript pour le gestionnaire de scripts. Cependant, vous pouvez accéder au gestionnaire de scripts à partir d'un script AppleScript en exécutant un script JavaScript depuis AppleScript (voir la section « Exécution de code du gestionnaire de scripts JavaScript à partir d'un script AppleScript », page 84).

Installation de l'écouteur de scripts

Le module externe ScriptListener est situé dans le dossier `..\Adobe Photoshop CS4\Scripting\Utilities`.

Installation de l'écouteur de scripts :

1. Sélectionnez le fichier `Ecouteur de scripts.8li`, puis choisissez la commande **Edition** > **Copier**.
2. Collez la copie du fichier à l'emplacement suivant :

   `..\Adobe Photoshop CS4\Plug-Ins\Automate`

3. Ouvrez Photoshop.

   REMARQUE : si Photoshop est déjà ouvert, fermez-le puis relancez-le. Cette opération permet à Photoshop de charger le module externe.

Désinstallation de l'écouteur de scripts :

1. Fermez Photoshop.
2. Vérifiez qu'une copie du fichier `ScriptListener.8li` existe déjà dans le dossier `..\Adobe Photoshop CS4\Scripting\Utilities`.
3. Supprimez le fichier `Ecouteur de scripts.8li` situé à l'emplacement suivant :

   `..\Adobe Photoshop CS\Plug-Ins\Automate`
4. Supprimez les fichiers d'historique `ScriptingListenerJS.log` et `ScriptingListenerVB.log` de votre bureau.

REMARQUE : sous Windows, l'écouteur de scripts peut continuer à enregistrer les actions même après avoir été retiré du dossier Automatisation. Pour empêcher le fichier `ScriptingListenerJS.log` de devenir trop volumineux, supprimez-le à chaque fois que vous finissez d'exécuter un script Photoshop.

Objets du gestionnaire de scripts

Les objets `Action Descriptor`, `Action List` et `Action Reference` font partie de la fonctionnalité Gestionnaire de scripts. Pour plus de détails sur ces objets, consultez le guide de référence correspondant ou utilisez l'explorateur d'objets du langage que vous utilisez.

REMARQUE : ces objets ne sont pas disponibles dans AppleScript.

Enregistrement d'un script à l'aide de l'écouteur de scripts

Cette section montre comment créer un fichier d'historique de script à l'aide de l'écouteur de scripts. Nous allons enregistrer les actions nécessaires à l'application du filtre Estampage à un document (Par défaut, le filtre Estampage est disponible uniquement via l'interface Photoshop.)

REMARQUE : vous devez avoir installé le module externe Ecouteur de scripts dans le dossier `Automatisation` avant de commencer la procédure suivante (voir la section « Installation de l'écouteur de scripts », page 77).

Création d'un script d'application du filtre Estampage

1. Ouvrez Photoshop, puis ouvrez un document.
2. Choisissez la commande **Fenêtre > Scripts**, puis **Nouveau script** dans le menu du panneau Scripts.
3. Attribuez un nom au script, puis cliquez sur le bouton **Enregistrer**.
4. Choisissez la commande **Filtre > Esthétiques > Estampage**.

5. Appliquez les paramètres suivants :
   - ➢ Angle : 135
   - ➢ Hauteur : 3
   - ➢ Facteur : 100
6. Cliquez sur le bouton **OK**.
7. Vérifiez les fichiers d'historique de script :
   - ➢ Sous Windows, les fichiers d'historique apparaissent sur votre bureau.
   - ➢ Sous Mac OS, les fichiers d'historique apparaissent sur le bureau.

Utilisation du gestionnaire de scripts à partir d'un script JavaScript

Cette section montre comment utiliser le contenu du fichier d'historique `ScriptingListenerJS.log` pour créer votre script. Avant de commencer, vous devez préalablement avoir enregistré une action. L'exemple utilisé dans cette section suppose que vous avez suivi les instructions de la section « Enregistrement d'un script à l'aide de l'écouteur de scripts », page 78.

Les procédures décrites dans cette section utilisent le gestionnaire de scripts pour rendre le filtre Estampage disponible dans l'interface de scripts.

Création d'un script JavaScript à partir du résultat de l'écouteur de scripts

1. Utilisez l'une des méthodes suivantes :
   - ➢ Ouvrez le fichier `ScriptingListenerJS.log` sur le bureau.

   Un code similaire à l'exemple ci-dessous apparaît à la fin du fichier (les nombres peuvent toutefois différer) :

```
var id19 = charIDToTypeID( "Embs" );
var desc4 = new ActionDescriptor();
var id20 = charIDToTypeID( "Angl" );
desc4.putInteger( id20, 135 );
var id21 = charIDToTypeID( "Hght" );
desc4.putInteger( id21, 3 );
var id22 = charIDToTypeID( "Amnt" );
desc4.putInteger( id22, 100 );
executeAction( id19, desc4 ,DialogModes.NO);
```

   REMARQUE : l'écouteur de scripts sépare les commandes consignées par des lignes horizontales composées de signes égal (=====...). Si l'action la plus récente n'est pas la première action enregistrée dans le fichier d'historique, elle se trouve à la suite de la dernière ligne de signes égal et est ainsi facilement identifiable.

2. Copiez le code JavaScript associé à l'action d'estampage du fichier `ScriptingListenerJS.log` dans un autre fichier, appelé `emboss.jsx`.

3. Dans le script `emboss.jsx`, identifiez les valeurs que vous avez utilisées avec le filtre (135, 3 et 100). Remplacez les valeurs de filtre spécifiées par des noms de variables.

   Dans l'exemple suivant, la valeur `135` a été remplacée par `angle`, `3` par `height` et `100` par `amount`.

```
var id19 = charIDToTypeID( "Embs" );
    var desc4 = new ActionDescriptor();
    var id20 = charIDToTypeID( "Angl" );
    desc4.putInteger( id20, angle );
    var id21 = charIDToTypeID( "Hght" );
    desc4.putInteger( id21, height );
    var id22 = charIDToTypeID( "Amnt" );
    desc7.putInteger( id22, amount );
    executeAction( id19, desc4,DialogModes.NO );
```

4. Incluez le code dans une fonction JavaScript. La fonction utilisée dans l'exemple suivant est appelée `emboss`.

```
function emboss( angle, height, amount )
{
    var id19 = charIDToTypeID( "Embs" );
    var desc4 = new ActionDescriptor();
    var id20 = charIDToTypeID( "Angl" );
    desc4.putInteger( id20, angle );
    var id21 = charIDToTypeID( "Hght" );
    desc4.putInteger( id21, height );
    var id22 = charIDToTypeID( "Amnt" );
    desc7.putInteger( id22, amount );
    executeAction( id19, desc4 ,DialogModes.NO);
}
```

5. Pour appliquer un filtre Estampage à un document à l'aide d'un script JavaScript, incluez la fonction emboss dans le script JavaScript et appelez cette fonction avec les paramètres souhaités. L'exemple suivant applique le filtre Estampage avec un angle de 75, une hauteur de 2 et un facteur de 89 (voir la section « Ouverture d'un document », page 30 pour obtenir de l'aide sur l'écriture de code permettant d'ouvrir un document dans le script).

```
// Open the document in the script
var fileRef = new File("/c/myfile")
var docRef = app.open(fileRef)

//Call emboss with desired parameters
emboss( 75, 2, 89 );
//finish the script

//include the function in the script file
function emboss(angle, height, amount )
{
    var id32 = charIDToTypeID( "Embs" );
    var desc7 = new ActionDescriptor();
    var id33 = charIDToTypeID( "Angl" );
    desc7.putInteger( id33, angle );
    var id34 = charIDToTypeID( "Hght" );
    desc7.putInteger( id34, height );
    var id35 = charIDToTypeID( "Amnt" );
    desc7.putInteger( id35, amount );
    executeAction( id32, desc7,DialogModes.NO );
}
```

6. Ouvrez Photoshop, pour appliquer le filtre Estampage en sélectionnant **Fichier > Scripts > Parcourir**, puis en sélectionnant l'emplacement de votre script emboss.jsx. Sélectionnez la commande **Charger** pour exécuter le script.

Utilisation du gestionnaire de scripts à partir d'un script VBS

Cette section montre comment utiliser le contenu du fichier d'historique `ScriptingListenerVB.log` pour créer votre script. Avant de commencer, vous devez avoir enregistré une action. L'exemple utilisé dans cette section suppose que vous avez suivi les instructions de la section « Enregistrement d'un script à l'aide de l'écouteur de scripts », page 78.

Les procédures décrites dans cette section utilisent le gestionnaire de scripts pour rendre le filtre Estampage disponible dans l'interface de scripts.

Création d'un script VBScript à partir du résultat de l'écouteur de scripts

1. Ouvrez le fichier `ScriptingListenerVB.log` à partir du bureau.

   Un code similaire à l'exemple ci-dessous apparaît à la fin du fichier (les nombres peuvent toutefois différer) :

```
DIM objApp
SET objApp = CreateObject("Photoshop.Application")
REM Use dialog mode 3 for show no dialogs
DIM dialogMode
dialogMode = 3
DIM id9
id9 = objApp.CharIDToTypeID( "Embs" )
DIM desc4
SET desc4 = CreateObject( "Photoshop.ActionDescriptor" )
DIM id10
id10 = objApp.CharIDToTypeID( "Angl" )
Call desc4.PutInteger( id10, 135 )
DIM id11
id11 = objApp.CharIDToTypeID( "Hght" )
Call desc4.PutInteger( id11, 3 )
DIM id12
id12 = objApp.CharIDToTypeID( "Amnt" )
Call desc4.PutInteger( id12, 100 )
Call objApp.ExecuteAction( id9, desc4, dialogMode )
```

   REMARQUE : l'écouteur de scripts sépare les commandes consignées par des lignes horizontales composées de signes égal (====...). Si l'action la plus récente n'est pas la première action enregistrée dans le fichier d'historique, elle se trouve à la suite de la dernière ligne de signes égal et est ainsi facilement identifiable.

2. Copiez le code VBScript associé à l'action d'estampage du fichier `ScriptingListenerVB.log` dans un autre fichier, appelé `emboss.vbs`.

3. Dans le script `emboss.vbs`, identifiez les valeurs que vous avez utilisées avec le filtre (135, 3 et 100). Remplacez les valeurs de filtre spécifiées par des noms de variables.

   Dans l'exemple suivant, `135` a été remplacé par `angle`, `3` par `height` et `100` par `amount`.

```
DIM objApp
SET objApp = CreateObject("Photoshop.Application")
REM Use dialog mode 3 for show no dialogs
DIM dialogMode
dialogMode = 3
DIM id9
id9 = objApp.CharIDToTypeID( "Embs" )
DIM desc4
SET desc4 = CreateObject( "Photoshop.ActionDescriptor" )
DIM id10
id10 = objApp.CharIDToTypeID( "Angl" )
Call desc4.PutInteger( id10, angle)
DIM id11
id11 = objApp.CharIDToTypeID( "Hght" )
Call desc4.PutInteger( id11, height )
DIM id12
id12 = objApp.CharIDToTypeID( "Amnt" )
Call desc4.PutInteger( id12, amount )
Call objApp.ExecuteAction( id9, desc4, dialogMode )
```

4. Incluez le code dans une fonction VBScript. La fonction utilisée dans l'exemple suivant est appelée `Emboss`. La création de l'objet d'application Photoshop doit être à l'extérieur de la fonction, comme nous l'expliquerons à la prochaine étape.

```
DIM objApp
SET objApp = CreateObject("Photoshop.Application")

Function Emboss( angle, height, amount )
    REM Use dialog mode 3 for show no dialogs
    DIM dialogMode
    dialogMode = 3
    DIM id9
    id9 = objApp.CharIDToTypeID( "Embs" )
     DIM desc4
    SET desc4 = CreateObject( "Photoshop.ActionDescriptor" )
    DIM id10
     id10 = objApp.CharIDToTypeID( "Angl" )
    Call desc4.PutInteger( id10, angle )
     DIM id11
    id11 = objApp.CharIDToTypeID( "Hght" )
    Call desc4.PutInteger( id11, height )
     DIM id12
    id12 = objApp.CharIDToTypeID( "Amnt" )
    Call desc4.PutInteger( id12, amount )
    Call objApp.ExecuteAction( id9, desc4, dialogMode )
End Function
```

5. Pour appliquer le filtre Estampage à un document à l'aide d'un script VBScript, incluez la fonction emboss dans le script et appelez cette fonction avec les paramètres souhaités. L'exemple suivant applique le filtre Estampage avec un angle de 75, une hauteur de 2 et un facteur de 89. Avant que le script n'appelle la fonction, il doit ouvrir un document (voir la section « Ouverture d'un document », page 30 pour obtenir de l'aide sur l'écriture de code permettant d'ouvrir un document dans le script). Pour cela, le script doit accéder aux modèles d'objets de document (DOM) Photoshop lorsqu'il appelle la méthode `Application.Open`. Il doit donc préalablement créer l'objet `Photoshop.Application`.

```
DIM objApp
SET objApp = CreateObject("Photoshop.Application")

'Open the document in the script
filename = "C:\MyFile"
DIM docRef
SET docRef = objApp.Open(filename)

'Call emboss with desired parameters
Call Emboss( 75, 2, 89 )

Function Emboss( angle, height, amount )
    REM Use dialog mode 3 for show no dialogs
    DIM dialogMode
    dialogMode = 3
    DIM id9
    id9 = objApp.CharIDToTypeID( "Embs" )
    DIM desc4
    SET desc4 = CreateObject( "Photoshop.ActionDescriptor" )
    DIM id10
    id10 = objApp.CharIDToTypeID( "Angl" )
    Call desc4.PutInteger( id10, angle )
    DIM id11
    id11 = objApp.CharIDToTypeID( "Hght" )
    Call desc4.PutInteger( id11, height )
    DIM id12
    id12 = objApp.CharIDToTypeID( "Amnt" )
    Call desc4.PutInteger( id12, amount )
    Call objApp.ExecuteAction( id9, desc4, dialogMode )
End Function
```

6. Appliquez le script du filtre Estampage en cliquant deux fois sur le fichier `emboss.vbs`. Cette opération lance Photoshop, ouvre le fichier et applique le filtre Estampage au fichier.

Exécution de code du gestionnaire de scripts JavaScript à partir d'un script VBScript

La méthode `DoJavaScriptFile` permet également d'accéder à du code du gestionnaire de scripts JavaScript à partir d'un script VBScript. Pour plus de détails sur la méthode `Application.DoJavaScriptFile`, utilisez l'explorateur d'objets VBscript.

Exécution de code du gestionnaire de scripts JavaScript à partir d'un script VBScript

1. Suivez les étapes 1 à 4 de la section « Utilisation du gestionnaire de scripts à partir d'un script JavaScript », page 79. Vous obtiendrez un fichier (`emboss.jsx`) contenant le code JavaScript suivant :

```
function emboss( angle, height, amount )
{
    var id32 = charIDToTypeID( "Embs" );
    var desc7 = new ActionDescriptor();
    var id33 = charIDToTypeID( "Angl" );
    desc7.putInteger( id33, angle );
    var id34 = charIDToTypeID( "Hght" );
    desc7.putInteger( id34, height );
    var id35 = charIDToTypeID( "Amnt" );
    desc7.putInteger( id35, amount );
    executeAction( id32, desc7 );
}
```

2. A la fin du fichier `emboss.jsx`, ajoutez la ligne de code JavaScript suivante pour exécuter la fonction emboss en lui transmettant des arguments à partir d'un appel externe. Reportez-vous au guide *Introduction to Scripting (Introduction aux scripts)* pour plus de détails sur la transmission d'arguments d'un script VBScript à un script JavaScript.

```
// Call emboss with values provided in the "arguments" collection
emboss( arguments[0], arguments[1], arguments[2] );
```

3. A partir d'un script VBScript, vous pouvez exécuter le filtre Estampage comme suit (cet exemple suppose que le fichier `emboss.jsx` se trouve sur le lecteur C:\) :

```
Set objApp = CreateObject("Photoshop.Application")

'Open the document in the script
filename = "C:\MyFile"
DIM docRef
SET docRef = objApp.Open(filename)

objApp.DoJavaScriptFile "C:\emboss.jsx", Array(75, 2, 89)
```

Exécution de code du gestionnaire de scripts JavaScript à partir d'un script AppleScript

AppleScript ne contient aucune fonctionnalité de gestionnaire de scripts. Cependant, vous pouvez exécuter un code et des fichiers JavaScript à partir de scripts AppleScript à l'aide de la commande `do javascript`. Pour plus de détails, reportez-vous au guide *Introduction to Scripting (Introduction aux scripts)*.

1. Suivez les étapes 1 à 4 de la section « Utilisation du gestionnaire de scripts à partir d'un script JavaScript », page 79. Vous obtiendrez un fichier (`emboss.jsx`) contenant le code JavaScript suivant :

```
function emboss( angle, height, amount )
{
    var id32 = charIDToTypeID( "Embs" );
    var desc7 = new ActionDescriptor();
    var id33 = charIDToTypeID( "Angl" );
    desc7.putInteger( id33, angle );
    var id34 = charIDToTypeID( "Hght" );
    desc7.putInteger( id34, height );
    var id35 = charIDToTypeID( "Amnt" );
    desc7.putInteger( id35, amount );
    executeAction( id32, desc7 );
}
```

2. A la fin du fichier `emboss.jsx`, ajoutez la ligne de code JavaScript suivante pour exécuter la fonction emboss en lui transmettant des arguments à partir d'un appel externe. Reportez-vous au guide *Introduction to Scripting (Introduction aux scripts)* pour plus de détails sur la transmission d'arguments d'un script AppleScript à un script JavaScript.

```
// Call emboss with values provided in the "arguments" collection
emboss( arguments[0], arguments[1], arguments[2] );
```

3. L'exemple de code AppleScript suivant ouvre un document et exécute le filtre Estampage :

```
tell application "Adobe Photoshop CS4"
    set theFile to alias "Application:Documents:MyFile"
    open theFile
    do javascript (file <path to Emboss.jsx>) ¬
        with arguments { 75,2,89 }
end tell
```

Utilisation de l'écouteur de scripts pour rechercher des ID d'événements et de classes

La section explique comment utiliser ScriptListener pour déterminer des ID d'événements et des ID de classe pour des actions effectuées par Photoshop. Ces ID sont utilisés pour configurer la notification à l'aide de la classe Classe Notifier.

Vous pouvez déterminer l'ID d'événement de toutes les actions enregistrables en utilisant l'écouteur de scripts. Il vous suffit pour cela d'installer le module externe Ecouteur de scripts comme indiqué à la section « Installation de l'écouteur de scripts », page 77. Exécutez ensuite l'action pour laquelle vous souhaitez rechercher l'ID d'événement. L'événement est alors consigné dans le fichier d'historique de l'écouteur de scripts (voir la section « Module externe Ecouteur de scripts », page 77). Si l'événement s'applique à plusieurs classes d'objets, l'ID de classe est également consigné dans le fichier d'historique.

Les exemples suivants montrent comment rechercher l'ID de l'événement Open Document (Ouvrir le document) et les ID d'événements et de classes de l'événement New (Nouveau), qui s'applique à plusieurs classes.

Recherche de l'ID de l'événement Open Document

1. Assurez-vous que le module externe Ecouteur de scripts est installé.

2. Ouvrez Photoshop, puis ouvrez un document.

3. Recherchez le fichier d'historique de l'écouteur de scripts et ouvrez-le. Vous pouvez utiliser le fichier d'historique VBScript ou JavaScript. Dans la version JavaScript du fichier, un code similaire à l'exemple ci-dessous apparaît à la fin du fichier (tous les éléments situés sous la ligne composée de signes égal correspondent à la dernière action effectuée) :

```
// =======================================================
var id14 = charIDToTypeID( "Opn " );
var desc5 = new ActionDescriptor();
var id15 = charIDToTypeID( "null" );
desc5.putPath( id15, new File( "C:\\Program Files\\Adobe\\Adobe Photoshop CS4\\
  Samples\\Fish.psd" ) );
executeAction( id14, desc5, DialogModes.NO );
```

4. La méthode `executeAction` exécute l'action à partir d'un script et requiert l'ID d'événement pour identifier l'action à effectuer. Le premier argument, dans ce cas `id14`, fournit l'ID d'événement à la méthode. Vous pouvez constater que la variable `id14` est définie quelques lignes plus haut, et que l'ID d'événement de l'action Open Document est `"Opn "`.

5. Vous pouvez désormais utiliser cet ID d'événement pour configurer une notification d'événement pour l'événement Open Document à partir de vos scripts. Par exemple, dans JavaScript :

```
var eventFile = new File(app.path +
                         "/Presets/Scripts/Event Scripts Only/Welcome.jsx")
app.notifiers.add( "Opn ", eventFile)
```

Recherche de l'ID d'événement et de l'ID de classe de l'événement New

1. Assurez-vous que le module externe Ecouteur de scripts est installé.

2. Ouvrez Photoshop, puis créez un document à l'aide de la commande Fichier > Nouveau.

3. Créez ensuite une nouvelle couche en cliquant sur l'icône Créer une couche dans le panneau Couches.

4. Recherchez le fichier d'historique de l'écouteur de scripts et ouvrez-le. Vous pouvez utiliser le fichier d'historique VBScript ou JavaScript. Comme nous avons enregistré deux actions, nous recherchons les deux dernières sections du fichier, qui sont délimitées par les lignes de signes égal. Dans le fichier historique JavaScript, un code similaire à l'exemple ci-dessous apparaît en fin de fichier :

```
// =======================================================
var id17 = charIDToTypeID( "Mk  " );
var desc6 = new ActionDescriptor();
var id18 = charIDToTypeID( "Nw  " );
var desc7 = new ActionDescriptor();
var id19 = charIDToTypeID( "Md  " );
var id20 = charIDToTypeID( "RGBM" );
desc7.putClass( id19, id20 );
var id21 = charIDToTypeID( "Wdth" );
var id22 = charIDToTypeID( "#Rlt" );
desc7.putUnitDouble( id21, id22, 800.000000 );
var id23 = charIDToTypeID( "Hght" );
var id24 = charIDToTypeID( "#Rlt" );
desc7.putUnitDouble( id23, id24, 800.000000 );
var id25 = charIDToTypeID( "Rslt" );
var id26 = charIDToTypeID( "#Rsl" );
desc7.putUnitDouble( id25, id26, 72.000000 );
var id27 = stringIDToTypeID( "pixelScaleFactor" );
desc7.putDouble( id27, 1.000000 );
var id28 = charIDToTypeID( "Fl  " );
var id29 = charIDToTypeID( "Fl  " );
var id30 = charIDToTypeID( "Wht " );
desc7.putEnumerated( id28, id29, id30 );
var id31 = charIDToTypeID( "Dpth" );
desc7.putInteger( id31, 8 );
var id32 = stringIDToTypeID( "profile" );
desc7.putString( id32, "sRGB IEC61966-2.1" );
var id33 = charIDToTypeID( "Dcmn" );
desc6.putObject( id18, id33, desc7 );
executeAction( id17, desc6, DialogModes.NO );
```

```
// =======================================================
var id34 = charIDToTypeID( "Mk  " );
var desc8 = new ActionDescriptor();
var id35 = charIDToTypeID( "Nw  " );
var desc9 = new ActionDescriptor();
var id36 = charIDToTypeID( "ClrI" );
var id37 = charIDToTypeID( "MskI" );
var id38 = charIDToTypeID( "MskA" );
desc9.putEnumerated( id36, id37, id38 );
var id39 = charIDToTypeID( "Clr " );
var desc10 = new ActionDescriptor();
var id40 = charIDToTypeID( "Rd  " );
desc10.putDouble( id40, 255.000000 );
var id41 = charIDToTypeID( "Grn " );
desc10.putDouble( id41, 0.000000 );
var id42 = charIDToTypeID( "Bl  " );
desc10.putDouble( id42, 0.000000 );
var id43 = charIDToTypeID( "RGBC" );
desc9.putObject( id39, id43, desc10 );
var id44 = charIDToTypeID( "Opct" );
desc9.putInteger( id44, 50 );
var id45 = charIDToTypeID( "Chnl" );
desc8.putObject( id35, id45, desc9 );
executeAction( id34, desc8, DialogModes.NO );
```

5. La première partie représente le code permettant d'exécuter l'événement New Document (Nouveau document). La deuxième partie représente le code pour l'événement New Channel (Nouvelle couche).

6. La méthode `executeAction` pour ces deux actions utilise un argument dont la valeur est définie comme `« Mk  »` (voir `id17` et `id34`). Il s'agit de l'ID d'événement de l'action New. Cette action a également besoin de connaître la classe à utiliser, l'ID de classe de l'événement.

7. La méthode `putObject` identifie la classe à laquelle s'applique l'action. Le deuxième argument de la méthode `putObject` nous fournit l'ID de classe dont nous avons besoin. Dans la première action, `id33` est défini comme `« Dcmn »`, tandis que dans la seconde action, `id45` est défini comme `« Chnl »`. soit respectivement nos ID de classes pour Document et pour Channel.

8. Vous pouvez désormais utiliser ces ID d'événements et de classes afin de configurer une notification d'événement pour les événements New Document et New Channel à partir de vos scripts. Par exemple, dans JavaScript :

```
var eventFile = new File(app.path +
                         "/Presets/Scripts/Event Scripts Only/Welcome.jsx")
app.notifiers.add("Mk  ", eventFile, "Dcmn")
app.notifiers.add("Mk  ", eventFile, "Chnl")
```

Index

A

actions
    vs. scripts, 9
affichage des boîtes de dialogue, 34
AppleScript
    conventions, 6
    création, 19
    exécution, 19
    exécution de scripts JavaScript à partir de, 11
    précisions sur la valeur d'unité, 59
Applescript
    affichage du dictionnaire, 22
Application, objet
    affichage des boîtes de dialogue, 34
    ciblage, 24
    défini, 12
    lien avec l'interface utilisateur, 15
    référencement, 24
    utilisation, 35
Art Layer, objet
    ajout dans un script JavaScript, 26
    ajout dans un script VBScript, 26
    application de styles, 42
    création, 38
    création d'un calque de texte, 43
    défini, 13
    filtres, 56
    lien avec l'interface utilisateur, 15
    référencement, 40
    utilisation, 38
automatisation d'un flux de production, JavaScript, 63

B

bibliothèque de types, VBScript, 23
boîtes de dialogue, contrôle, 34

C

calques de texte, 43
Channel, objet
    activation, 29
    défini, 13
    définition de la couche active, 29
    lien avec l'interface utilisateur, 15
    modification du type, 49
    types de, 13
    utilisation, 49
classes, recherche, 22
collections, indexation VBScript, 12
Color Sampler, objet
    défini, 14
    lien avec l'interface utilisateur, 16
Color, objet
    dans le DOM, 16
Color, objets
    application au texte, 67
    comparaison, 56
    définis, 54
    définition des valeurs hexadécimales, 55
    obtention et conversion, 55
    solid color, classes, 54
    utilisation, 54
    Web sécurisés, 56
commandes
    affichage, 22
    conventions, 7
constantes
    définies, 17
    recherche, 17, 22
contour
    sélections, 46
    texte, 68
conventions, 6
conventions typographiques, 6
copies avec fusion, 58
copy et paste, commandes, 57
copy merged, commande, 58
couches de composante, 13
couches de ton direct, 13
couches de zone masquée, 13
couches de zone sélectionnée, 13
couleur Web sécurisée, 56
Count Item, objet
    défini, 14
    lien avec l'interface utilisateur, 16

D

Document Info, objet
    défini, 14
    lien avec l'interface utilisateur, 16

utilisation, 49
Document, objet
activation, 27
ajout, 25
défini, 12
enregistrement, 32
informations sur le document, 49
lien avec l'interface utilisateur, 15
manipulation, 36
ouverture, 30
utilisation, 36
valeurs d'unité, 61

E

écouteur de scripts
désinstallation, 78
enregistrement de scripts, 78
fichiers journaux, 77
installation, 77
recherche d'ID d'événements, 85
recherche d'ID de classe, 85
Editeur de scripts
utilisation, 19
enregistrement des documents, 32
EPS open options, objet
valeurs d'unité, 61
états d'historique
définis, 50
extensions de fichier
fichiers de script, 10

F

fichier
définition du format, 30
fichiers
déduction du format, 30
enregistrement, 32
ouverture, 30
ouverture en utilisant des paramètres spécifiques, 30
filtres
application du filtre Flou directionnel, 75
application du filtre Onde, 71–74
rendre exécutable sous forme de script, 78
supplémentaires, 57
utilisation, 56
flou directionnel, définition, 75

G

Gestionnaire de scripts
défini, 77
exécution de code JavaScript à partir de scripts AppleScript, 84
exécution de code JavaScript à partir de scripts VBScript, 83
objets de scripts, 78
utilisation à partir de scripts JavaScript, 79
utilisation à partir de scripts VBScript, 81

H

hiérarchie, 11
hiérarchie d'imbrication, 11
History State, objet
défini, 14
lien avec l'interface utilisateur, 16
purge, 50
rétablissement, 50
utilisation, 50

I

ID d'événements, recherche avec l'écouteur de scripts, 85
ID de classe, recherche, 85
images, modification de la composition, 13

J

JavaScript
conventions, 6
création, 20
exécution, 10, 20
exécution à partir de scripts Applescript, 11
exécution à partir de scripts VBScript, 11
exemple d'automatisation d'un flux de production, 63
prise en charge, 10
utilisation du Gestionnaire de scripts, 79

L

langages de script
exemple de scripts, 18
pris en charge, 9
Layer Comp, objet
défini, 13
lien avec l'interface utilisateur, 15
Layer Set, objet
création, 39
défini, 13
lien avec l'interface utilisateur, 15
utilisation, 38, 41

layer, classes, 13
Layer, objets
    activation, 28
    ajout, 26
    application de styles, 42
    création, 38
    définis, 13
    définition du type, 43
    liaison, 41
    référencement, 40
    test pour calques de texte, 43
    utilisation, 38
lens flare open options, objet
    valeurs d'unité, 61
Les, 57
logique conditionnelle, 9

M

Measurement Scale, objet
    défini, 15
    lien avec l'interface utilisateur, 16
mesure, unités
    utilisation, 59
métadonnées, définies, 16
méthodes
    affichage, 22
    conventions, 7
modèle d'objet
    concepts, 11
    utilisation, 35
modèle d'objet d'Adobe Photoshop, 12, 35
modèle d'objet de document (DOM), *Voir* modèle d'objet
modèle d'objet de Photoshop, 12

N

notification d'événement, paramétrage, 51
Notifier, objet
    défini, 14
    lien avec l'interface utilisateur, 16
    recherche d'ID d'événements, 85
    recherche d'ID de classe, 85
    utilisation, 51

O

objet
    activation, 26
objets
    Voir aussi le nom de chaque objet
    affichage, 22
    création dans un script, 24–26
    hiérarchie, 11
    modèle d'objet d'Adobe Photoshop, 12
objets actifs, définition, 26
offset filter, objet
    valeurs d'unité, 61
Onde, application du filtre, 71–74
Open options, classes, 17
opérations, valeurs d'unité, 60

P

parent, objets définis, 27
paste, commandes, 57
Path Item, objet
    création d'une ligne droite, 52
    défini, 14
    lien avec l'interface utilisateur, 15
Path Point, objet
    défini, 14
PDF open options, objet
    valeurs d'unité, 61
préférences
    définition, 34
    utilisation, 64
Preferences, objet
    défini, 14
    lien avec l'interface utilisateur, 16
Presse-papiers, commandes, 57
propriétés
    conventions, 7
    recherche, 22

R

règle, unités
    définies, 59
    définition, 62
    utilisation de la valeur, 61
    valeurs, 59

S

Save options, classes, 17
script Hello World, 18–21
scripts
    avancés, 64
    création, 64
    création d'objets, 24–26
    définis, 8
    démarrage, 10
    emplacements du fichier, 10
    enregistrement, 78

exécution, 10
extensions de fichier valides, 10
fonctionnalité, 9
fonctions, 9
utilisation, 77
vs. actions, 9
scripts de démarrage, 10
Scripts, panneau, 77
Selection, objet
chargement, 48
contour, 46
contour progressif, 47
création, 45
défini, 13
définition de la zone, 71
inversion, 47
lien avec l'interface utilisateur, 15
redimensionnement, 47
remplissage, 47
restauration, 48
stockage, 48
utilisation, 45
Solid Color, classes, 54
styles, application aux calques, 42
Sub Path Item, objet
défini, 14

T

Text Item, objet
création, 43
défini, 13
mise en forme de texte, 44
utilisation, 43
valeurs d'unité, 61
texte
application de couleur, 67
calques, 43
contour, 68
mise en forme, 44
texte, unités
définies, 59
définition, 62
tracés, création, 52
types de valeurs
constantes, 17

U

unités
comme paramètres, 62
comme propriétés, 61
dans les arguments, 62
dans les opérations, 60
définition, 62
précisions pour AppleScript, 59
types spéciaux, 59
utilisation, 59
utilisation de la valeur, 61
valeurs, 59
unités de mesure
préférences document, 64

V

valeurs chromatiques hexadécimales, définition, 55
valeurs énumérées
recherche, 22
VBScript
bibliothèque de types, 23
conventions, 6
création, 20
exécution, 20
exécution de scripts JavaScript à partir de, 11
utilisation du Gestionnaire de scripts, 81