



ADOBE PHOTOSHOP CS6

スクリプティングガイド

© 2010 Adobe Systems Incorporated. All Rights Reserved.

Adobe® Creative Suite® 6 Photoshop® スクリプティングガイド

Adobe、Adobe ロゴ、Illustrator および Photoshop は、アドビ システムズ社の米国ならびに他の国における商標または登録商標です。Apple and Mac OS are trademarks of Apple Inc., registered in the United States and other countries. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries. JavaScript and all Java-related marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other trademarks are the property of their respective owners.

本マニュアルに記載される内容は、あくまでも参照用としてのみ使用されること、また、なんら予告なしに変更されることを条件として、提供されるものであり、従って、当該情報が、アドビ システムズ社による確約として解釈されてはなりません。アドビ システムズ社は、本マニュアルにおけるいかなる誤りまたは不正確な記述に対しても、いかなる義務や責任を負うものではありません。本マニュアルに記載されているソフトウェアは、エンドユーザ使用許諾契約にもとづいて提供されるものであり、当該エンドユーザ使用許諾契約の契約条件に従ってのみ使用または複製することが可能となるものです。

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

目次

1	はじめに	6
	このマニュアルについて	6
	このマニュアルの表記規則	6
2	Photoshop スクリプティングの基礎	8
	スクリプティングの概要	8
	アクションではなくスクリプトを使用する理由	8
	Photoshop のスクリプティングサポート	9
	JavaScript のサポート	10
	他のスクリプトの実行	10
	スタートアップスクリプト	10
	AS や VBS からの JavaScript の実行	11
	Photoshop オブジェクトモデル	11
	包含階層	11
	Application クラスと Document クラス	12
	レイヤークラス	12
	Layer Comp クラス	13
	Channel クラス	13
	Selection クラス	13
	History State クラス	13
	Document Info クラス	13
	Path Item、Sub Path Item、Path Point クラス	14
	Preferences クラス	14
	Notifier クラス	14
	Count Item クラス	14
	Color Sampler クラス	14
	Measurement Scale クラス	14
	包含階層と Photoshop のユーザインターフェイス	14
	その他のオブジェクト	16
	定数	16
	サンプル Hello World スクリプトの作成	17
	AppleScript の作成と実行	18
	VBScript の作成と実行	19
	JavaScript の作成と実行	19
3	Photoshop のスクリプティング	21
	Photoshop のオブジェクト、コマンド、メソッドの表示	21
	Photoshop の AppleScript 用語説明の表示	21
	Photoshop のタイプライブラリの表示 (VBS)	22
	Application オブジェクトの指定と参照	22
	スクリプトでの新規オブジェクトの作成	23
	アクティブなオブジェクトの設定	25
	アクティブなドキュメントの設定	26

アクティブなレイヤーの設定	27
アクティブなチャンネルの設定	28
ドキュメントのオープン	28
デフォルトファイル形式のファイルのオープン	28
ファイルを開く形式の指定	29
ドキュメントの保存	31
アプリケーションの環境設定	32
ダイアログボックスの有効化と無効化	33
Photoshop のオブジェクトモデルの操作	33
Application オブジェクトの使用	34
Document オブジェクトの使用	34
Document オブジェクトの操作	35
レイヤーオブジェクトの操作	36
ArtLayer オブジェクトの作成	37
Layer Set オブジェクトの作成	38
ArtLayer オブジェクトの参照	38
Layer Set オブジェクトの操作	39
レイヤーオブジェクトのリンク	40
レイヤーへのスタイルの適用	40
Text Item オブジェクトの使用	41
レイヤーのタイプの特定	41
Text Item オブジェクトへのテキストの追加と操作	42
Selection オブジェクトの操作	42
選択範囲の作成と定義	43
選択範囲の境界線の作成	44
選択範囲の反転	44
選択範囲の拡張／縮小と境界のぼかし	44
選択範囲の塗りつぶし	45
選択範囲の読み込みと保存	45
Channel オブジェクトの操作	46
チャンネルのタイプ変更	46
DocumentInfo オブジェクトの使用	47
HistoryState オブジェクトの使用	47
Notifier オブジェクトの使用	48
PathItem オブジェクトの使用	49
カラーオブジェクトの操作	51
Solid Color クラス	52
RGB カラーでの 16 進数値の使用	52
カラーの取得と変換	52
カラーの比較	53
Web セーフカラーの取得	53
フィルタの操作	53
その他のフィルタ	54
クリップボード操作の理解	54
コピーおよびペーストコマンドの使用	54
マージコピーコマンド	55
単位の操作	56
単位値	56
特別な単位	56

AppleScript での単位に関する考慮事項	56
計算における単位値の使用	57
単位値の使用	57
スクリプトによる定規単位や文字単位の設定	59
ワークフローを自動化するためのサンプル JavaScript	60
高度なスクリプティング	60
ドキュメントの環境設定の操作	61
テキストアイテムへのカラーの適用	64
波形フィルタの適用	66
選択オブジェクトの領域定義	67
ぼかし（移動）フィルタの適用	71
4 アクションマネージャ	73
ScriptListener プラグイン	73
ScriptListener のインストール	73
アクションマネージャのスクリプティングオブジェクト	74
ScriptListener によるスクリプトの記録	74
JavaScript からのアクションマネージャの使用	75
VBScript からのアクションマネージャの使用	76
VBScript での JavaScript ベースのアクションマネージャコードの実行	79
AppleScript での JavaScript ベースのアクションマネージャコードの実行	80
ScriptListener によるイベント ID やクラス ID の確認	81
索引	84

1 はじめに

このマニュアルについて

このマニュアルでは、Macintosh® や Windows® で Adobe® Photoshop® CS6 のスクリプトを作成するための基本事項について説明します。

第 1 章では、このマニュアルの基本的な表記規則について説明します。

第 2 章では、スクリプティングの概要について説明するとともに、スクリプトの実行方法や、Photoshop オブジェクトモデルについて説明します。

第 3 章では、Photoshop 特有のオブジェクトやコンポーネントについて説明します。また、Photoshop アプリケーションのスクリプトを作成するための高度なテクニックについて説明します。コード例は、次の 3 つの言語で示します。

- AppleScript
- VBScript
- JavaScript™

注: Photoshop のスクリプトを作成するために必要なリファレンス情報については、このインストールに含まれている言語別の『Scripting Reference』というマニュアルを参照してください。または、各言語の開発環境に用意されているオブジェクトブラウザを参照することもできます。[21 ページの「Photoshop の AppleScript 用語説明の表示」](#) および [22 ページの「Photoshop のタイプライブラリの表示 \(VBS\)」](#) を参照してください。ExtendScript のオブジェクトモデルビューアの使用方法について詳しくは、『JavaScript Tools Guide CS6』を参照してください。

第 4 章では、アクションマネージャについて説明します。アクションマネージャを使用すると、通常のスクリプティングインターフェイスからはアクセスできない Photoshop の機能にアクセスしてスクリプトを作成することができます。

注: Photoshop に付属している README ファイルに目を通して、最新のニュース、サンプルスクリプト、未解決問題などの情報を確認してください。

このマニュアルの表記規則

コードや特定の言語のサンプルは、次のように、固定幅の Courier フォントで示します。

```
app.documents.add
```

AppleScript、VBScript、JavaScript の名称については、次のように省略して表記する場合があります。

- AS は AppleScript を表します。
- VBS は VBScript を表します。
- JS は JavaScript を表します。

「コマンド」という用語は、AppleScript のコマンドを表す場合だけでなく、VBScript や JavaScript のメソッドを表す場合にも使用します。

このマニュアルで特定のプロパティやコマンドを示す場合は、AppleScript での名前を使用し、VBScript や JavaScript での名前はかっこに入れて示します。例えば、次のようになります。

「Application オブジェクトには、display dialogs (DisplayDialogs / displayDialogs) というプロパティがあります。」

この場合は、display dialogs が AppleScript のプロパティ、DisplayDialogs が VBScript のプロパティ、displayDialogs が JavaScript のプロパティになります。

コードが長い場合は、行をわけてスクリプト例を示します。

AS

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

VBS

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

JS

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

また、表を使用して、それぞれのスクリプト言語に固有な値を一覧表示することもあります。

2 Photoshop スクリプティングの基礎

この章では、Photoshop スクリプティングの概要について説明します。AppleScript、VBScript、JavaScript の各スクリプト言語のサポートについて説明するとともに、スクリプトの実行方法や、Photoshop オブジェクトモデルについても説明します。また、簡単な処理を実行する最初の Photoshop スクリプトも作成します。

スクリプト言語やプログラミング言語に習熟している方は、この章をすべて読む必要はありません。次のリストを参照して、必要な情報を見つけてください。

- ▶ Photoshop のオブジェクトモデルについて詳しくは、[11 ページの「Photoshop オブジェクトモデル」](#)を参照してください。
- ▶ スクリプト言語の選択について詳しくは、『Adobe Intro to Scripting』ガイドを参照してください。
- ▶ Photoshop 用のスクリプト例については、第 3 章の [21 ページの「Photoshop のスクリプティング」](#)を参照してください。
- ▶ Photoshop のオブジェクトやコマンドについて詳しくは、このインストールに含まれている 3 つのリファレンスマニュアル（『Adobe Photoshop CS6 AppleScript Scripting Reference』、『Adobe Photoshop CS6 Visual Basic Scripting Reference』、『Adobe Photoshop CS6 JavaScript Scripting Reference』）を参照してください。

注：または、各スクリプト言語の開発環境に用意されているオブジェクトブラウザでも、Photoshop のオブジェクトやコマンドが参照できます。[21 ページの「Photoshop のオブジェクト、コマンド、メソッドの表示」](#)を参照してください。

スクリプティングの概要

スクリプティングとは、スクリプトを記述することです。スクリプトとは、指定した一連の動作を行うように Photoshop に指示する一連のコマンドです。例えば、開いているドキュメント内の複数の選択個所に、それぞれ別のフィルタを適用したりします。一口に動作と言っても様々なものがあり、Photoshop ドキュメント内の 1 つのオブジェクトだけを処理する単純な動作もあれば、多数のオブジェクトを処理する複雑な動作もあります。また、Photoshop だけを呼び出す動作もあれば、他のアプリケーションを呼び出す動作もあります。

スクリプトを使用すれば、反復作業を自動化することができます。また、手動で行うには時間がかかりすぎる作業を短時間で処理できるクリエイティブツールとしても、よくスクリプトが利用されます。例えば、特定の画像の各国語版を生成するスクリプトを作成したり、複数の画像から様々なカラープロファイル情報を収集するスクリプトを作成したりすることができます。

スクリプトを初めて作成する方は、『Adobe Intro to Scripting』というマニュアルを参照して、スクリプティングの基本的な事柄について理解しておくことをお勧めします。

アクションではなくスクリプトを使用する理由

Photoshop のアクションを使用されている方なら、反復作業を自動化することがいかに便利なことか、既によくお分かりだろうと思います。スクリプティングを使用すれば、Photoshop のアクションでは行えなかった処理までできるようになるので、さらに便利な環境を手にすることができます。例えば、次のことはスクリプトでは行えますが、アクションでは行えません。

- スクリプトでは、**条件判断のロジック**を追加して、現在の状況に基づいた「判断」を自動的に行うことができます。例えば、「選択領域が 2×4 インチよりも小さければ緑色の境界を追加し、そうでない場合は赤色の境界を追加する」というように、画像の選択領域のサイズに基づいて追加する境界の色を決めるスクリプティングが可能です。
- 複数のアプリケーションに対するアクションを、1 つのスクリプトで実行することができます。例えば、使用するスクリプト言語によっては、同じスクリプトの中で、Photoshop だけでなく別の Adobe Creative Suite 6 アプリケーション（Adobe Illustrator® CS6 など）もターゲットにすることができます。
- スクリプトでは、ファイルを開いたり、保存したり、名前の変更を行ったりすることが可能です。
- スクリプトは、1 つのコンピュータから別のコンピュータにコピーすることができます。それに対してアクションは、使用するコンピュータを変えるたびに、今まで使用していたアクションをもう一度作り直す必要があります。
- スクリプトでは、自動的にファイルを開く際に、柔軟性の高い方法を使用できます。アクションでファイルを開く場合には、ファイルの場所をハードコードする必要があります。スクリプトでは、ファイルパスに変数を使用できます。

注: Photoshop のアクションについて詳しくは、Photoshop ヘルプを参照してください。

Photoshop のスクリプティングサポート

Photoshop では、AppleScript、VBScript、JavaScript の 3 つのスクリプト言語によるスクリプティングがサポートされています。Macintosh では AppleScript と JavaScript が実行できます。Windows では JavaScript と VBScript が実行できます。使用するスクリプト言語の選択方法や、アドビアプリケーションでこれらの言語を使用する方法について詳しくは、『Adobe Intro to Scripting』を参照してください。

[18 ページの「AppleScript の作成と実行」](#)、[19 ページの「VBScript の作成と実行」](#)、[19 ページの「JavaScript の作成と実行」](#)を参照してください。

AppleScript や VBScript のスクリプトで JavaScript のスクリプトを呼び出すことができます。[11 ページの「AS や VBS からの JavaScript の実行」](#)を参照してください。

有効なスクリプトファイルとして Photoshop で認識されるためには、ファイル名に適切な拡張子を使用する必要があります。

スクリプトの種類	ファイルの種類	拡張子	プラットフォーム
AppleScript	コンパイル済みスクリプト OSAS ファイル	.scpt (なし)	Macintosh®
JavaScript ExtendScript	テキスト	.js .jsx	Macintosh と Windows
VBScript	テキスト	.vbs	Windows
Visual Basic	実行可能ファイル	.exe	Windows

JavaScript のサポート

JavaScript の有効なスクリプトファイルとして Photoshop で認識されるためには、ファイルの拡張子に `.js` または `.jsx` を使用する必要があります。Macintosh では、2つの拡張子のどちらを使用してもスクリプトの動作は同じです。Windows では、Photoshop の内部からスクリプトファイルを開く場合は、`.js` と `.jsx` のどちらの拡張子であっても違いはありません。ただし、スクリプトをダブルクリックして起動する場合は、拡張子が `.js` のスクリプトは Microsoft® JScript エンジンで実行され、Photoshop は起動されません。拡張子が `.jsx` のスクリプトは ExtendScript エンジンで実行されるので、Windows では `.jsx` 拡張子を使用することをお勧めします。

JavaScript で作成したスクリプトは、Adobe Photoshop のスクリプトメニュー（ファイル／スクリプト）を使用して、素早く簡単に実行することができます。JavaScript ファイルをディスク上の適切な場所に保存すれば、Photoshop メニューから直接実行できるようになります。

スクリプトメニューに JavaScript を追加するには、Scripts フォルダ（**Adobe Photoshop CS6/Presets/Scripts**）にファイルを保存します。Scripts フォルダに保存されているスクリプトの名前（ファイル名の拡張子は除く）がスクリプトメニューに表示されます。スクリプトメニューに追加できるスクリプトの数に制限はありません。

Photoshop の実行中に Scripts フォルダに追加したスクリプトは、アプリケーションを次回起動するまでスクリプトメニューには表示されません。

Scripts フォルダとそのサブフォルダに保存されているスクリプトは、すべて**ファイル／スクリプト**メニューのトップレベルに表示されます。サブフォルダを追加しても、スクリプトメニューに階層構造は追加されません。

他のスクリプトの実行

スクリプトメニューの一番下にある「参照」という項目（ファイル／スクリプト／参照）を使用すれば、Scripts フォルダに保存されていないスクリプトを実行することができます。また、アプリケーションの実行中に Scripts フォルダに追加したスクリプトも、この方法を使用して実行できます。

「参照」を選択すると、ファイルブラウザダイアログボックスが表示され、実行するスクリプトファイルが選択できます。この参照ダイアログボックスには、`.js` ファイルまたは `.jsx` ファイルのみが表示されます。スクリプトファイルを選択すると、メニューに追加されているスクリプトを選択した場合と同様に実行されます。

スタートアップスクリプト

Photoshop を起動すると、スタートアップフォルダに保存されているすべての `.jsx` ファイルが実行されます。

- Windows では、ユーザ定義スクリプトを保存するスタートアップフォルダは、次のとおりです。

```
C:\Program Files\Common Files\Adobe\Startup Scripts CS6\Adobe Photoshop
```

- Macintosh では、ユーザ定義スクリプトを保存するスタートアップフォルダは、次のとおりです。

```
~/Library/Application Support/Adobe/Startup Scripts CS6/Adobe Photoshop
```

このメインのスタートアップフォルダに保存されているスクリプトは、Photoshop の起動時だけでなく、他のすべての Adobe Creative Suite 6 アプリケーションの起動時にも実行されます。Photoshop でのみ実行したいスクリプトには、次のようなコードを含める必要があります。

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

詳しくは、『JavaScript Tools Guide CS6』を参照してください。

AS や VBS から JavaScript の実行

AppleScript や VBScript から JavaScript を実行して、プラットフォームに依存しない JavaScript の利点を活用することができます。1 つの JavaScript 文を実行することもできますし、JavaScript ファイルを実行することもできます。詳しくは、『Adobe Intro to Scripting』を参照してください。

Photoshop オブジェクトモデル

ドキュメントオブジェクトモデル (DOM) は、アプリケーションプログラミングインターフェイス (API) の一種です。スクリプト言語でプログラムを作成するときは、この API を使用して、アプリケーションで定義されているドキュメントの様々なコンポーネントにアクセスします。Adobe オブジェクトモデル、およびそのモデルがサポートされているスクリプト言語について詳しくは、『Adobe Intro to Scripting』を参照してください。

Photoshop の DOM では、Photoshop アプリケーション、アプリケーションで使用されるドキュメント、ドキュメントの各コンポーネントが階層構造で表現されています。この DOM を使用すれば、ドキュメントやそのコンポーネントにプログラムでアクセスして操作を行うことができます。例えば、DOM を使用して新規ドキュメントを作成したり、既存のドキュメントにレイヤーを追加したり、レイヤーの背景色を変更したりすることができます。Photoshop のユーザインターフェイスで行える操作のほとんどは、DOM を使用して実行できます。

Photoshop の DOM を理解して、DOM と Photoshop アプリケーションの対応関係を把握すれば、スクリプトをスムーズに作成できるようになります。

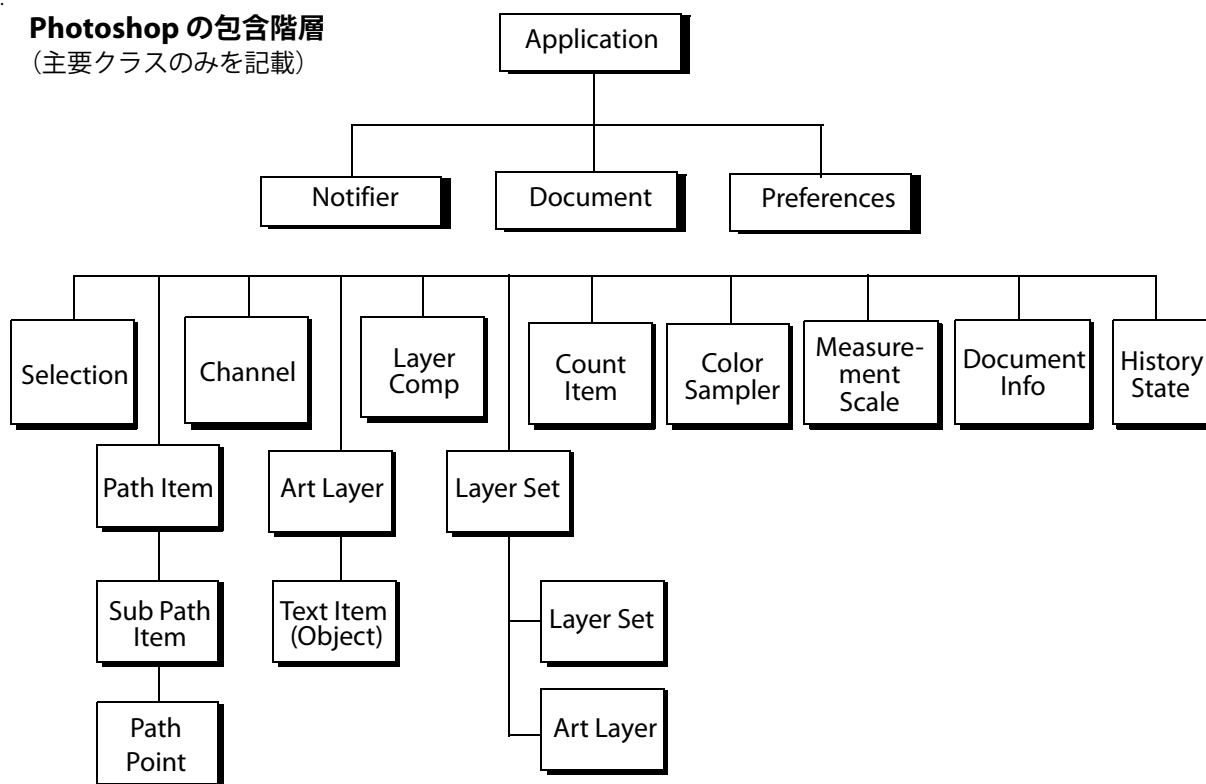
包含階層

Photoshop のオブジェクトモデルは、**包含階層**を形成しています。つまり、モデル内の各オブジェクトは、それを包含しているオブジェクトに基づいて識別されます。Photoshop では、階層の一番上に Application というオブジェクトがあります。Application には、Documents というコレクションが含まれています。Documents コレクションには、いくつかの Document オブジェクトが含まれています。Document オブジェクトには、ArtLayers コレクション、HistoryStates コレクション、Layers コレクション、Layersets コレクション、Channels コレクションなどが含まれています。DOM で定義されているコマンドやメソッドを使用して Photoshop ドキュメントに指示を出すことによって、オブジェクトの追加や削除を行ったり、カラー、サイズ、形状などの個々のオブジェクトプロパティを設定したり変更したりすることができます。次の図では、階層内の各ノードが Photoshop の DOM の各クラスを表しています。

Photoshop のオブジェクトモデルでは、エレメント (AppleScript) やコレクション (VBScript、JavaScript) を使用して、複数のクラスをグループ化しています。次のオブジェクトモデル図には、オブジェクトエレメントやコレクションは示されていません。すべてのクラスにコレクションが用意されているわけではありませんが、いくつかの重要なクラスはエレメントやコレクションによってグループ化されています。Photoshop で使用されるエレメントやコレクションには、Art Layers、Channels、Color Samplers、Count Items、Documents、Layers、Layer Comps、Layer Sets、History States、Notifiers、Path Items、Path Points Sub Path Items、Text Fonts があります。エレメントやコレクションについて詳しくは、『Adobe Intro to Scripting』を参照してください。

注: VBScript の場合、通常の配列のインデックスは 0 から数えますが、Photoshop のコレクションのインデックスは 0 ではなく 1 から数えます。

Photoshop の包含階層 (主要クラスのみを記載)



Application クラスと Document クラス

Photoshop のオブジェクトモデル階層のルートにあるのが `Application` クラスです。スクリプトを正しく実行するためには、適切なアプリケーションをターゲットに設定する必要があります。[22 ページの「Application オブジェクトの指定と参照」](#)を参照してください。

`Document` クラスは、ドキュメント画像に変更を加えるときに使用されます。`Document` オブジェクトを使用すれば、キャンバスの切り抜き、回転、反転が行えます。また、キャンバスや画像のサイズ変更、画像のトリミングなども行えます。さらに、アクティブなレイヤーを取得したり、現在のドキュメントを保存したり、アクティブなドキュメント内や異なるドキュメント間でコピー＆ペーストを行うこともできます。`Document` オブジェクトの使用方法について詳しくは、[23 ページの「スクリプトでの新規オブジェクトの作成」](#)および [34 ページの「Document オブジェクトの使用」](#)を参照してください。

レイヤークラス

Photoshop には、2 種類のレイヤーが用意されています。1 つは、画像コンテンツを格納できる `Art Layer` です。もう 1 つは、0 個以上のアートレイヤーを格納できる `Layer Set` です。

`Art Layer` は、画像内の 1 つの要素を、他の要素に影響を与えることなく操作できるようにするための、ドキュメント内のレイヤークラスです。画像は通常、`Layer Set` によって定義される複数のレイヤーで構成されます。画像を構成するレイヤーの順番や属性を変更することで、画像のコンポジションを変更することができ

ます。
`Text Item` は、画像に文字を追加できる特殊なアートレイヤーです。Photoshop では、`Text Item` アイテムは、アートレイヤーのプロパティとして実装されています。`Text Item` について詳しくは、[41 ページの「Text Item オブジェクトの使用」](#)を参照してください。

Layer Set は、複数のレイヤーを包含するクラスです。これは、デスクトップ上のフォルダに似ています。フォルダに別のフォルダを含めることができるのと同様に、レイヤーセットも再帰的に構成することができます。つまり、オブジェクトモデルの階層では、あるレイヤーセットから別のレイヤーセットを呼び出すことができます。

レイヤーについて詳しくは、[36 ページの「レイヤーオブジェクトの操作」](#)を参照してください。

Layer Comp クラス

Layer Comp クラスを使用すると、単一のドキュメントで、複数のバージョンのレイアウトを作成、管理、表示することができます。

Channel クラス

Channel クラスは、画像のカラーに関するピクセル情報を保存するために使用されます。利用できるチャンネルの数は、画像のカラーによって決まります。例えば、RGB 画像には、4 つのデフォルトチャンネルがあります。各基本カラー用のチャンネルと、画像全体を編集するためのチャンネルです。これらのチャンネルを使い分けることで、例えば、レッドチャンネルをアクティブにして画像内のレッドピクセルのみを操作したり、一度にすべてのチャンネルを操作したりすることができます。

これらのチャンネルは、ドキュメントのモードと関係しており、**コンポーネントチャンネル**と呼ばれます。コンポーネントチャンネルに加えて、Photoshop では追加のチャンネルを作成することができます。**スポットカラーチャンネル**、**マスク範囲チャンネル**、**選択領域チャンネル**を作成することができます。

Channel オブジェクトのコマンドやメソッドを使用して、チャンネルを作成、削除、複製することができます。また、チャンネルのヒストグラムを取得したり、チャンネルの種類を変更したり、現在のチャンネルの選択を変更したりすることもできます。

チャンネルについて詳しくは、[46 ページの「Channel オブジェクトの操作」](#)を参照してください。

Selection クラス

Selection クラスは、アクティブなドキュメント（または、アクティブなドキュメントの選択されたレイヤー）で、操作したいピクセル領域を指定するために使用します。選択について詳しくは、[42 ページの「Selection オブジェクトの操作」](#)を参照してください。

History State クラス

History State クラスは、ドキュメントに加えられた変更を記録するパレットオブジェクトです。画像に変更が加えられるたびに、その画像の新しい状態がパレットに追加されます。各状態にはドキュメントオブジェクトからアクセスでき、これを使用してドキュメントを前の状態に戻すことができます。また、ヒストリーを使用して選択範囲を塗りつぶすこともできます。ヒストリーオブジェクトについて詳しくは、[47 ページの「HistoryState オブジェクトの使用」](#)を参照してください。

注: AppleScript では、ドキュメントを作成した直後にヒストリーを取得しようとすると、エラーが発生します。ヒストリーにアクセスする前に、Photoshop をアクティブにする（最前面のアプリケーションにする）必要があります。

Document Info クラス

Document Info クラスには、ドキュメントに関するメタデータが保存されます。メタデータとは、ファイルの内容や特徴について記述した任意のデータのことです。Document Info について詳しくは、[47 ページの「DocumentInfo オブジェクトの使用」](#)を参照してください。

Path Item、Sub Path Item、Path Point クラス

Path Item クラスは、シェイプのアウトラインや曲線などの描画オブジェクトに関する情報を表します。Sub Path Item クラスは、Path Item クラスに含まれ、シェイプの実際の形状を表します。Path Point クラスは、Sub Path Item の各ポイントの情報を表します。[49 ページの「PathItem オブジェクトの使用」](#)を参照してください。

Preferences クラス

Preferences クラスを使用すると、ユーザ環境設定にアクセスして設定することができます。[61 ページの「ドキュメントの環境設定の操作」](#)を参照してください。

Notifier クラス

Notifier オブジェクトを使用すれば、スクリプトとイベントを関連付けることができます。例えば、アプリケーションを開いたときに Photoshop によって自動的に新規ドキュメントが作成されるようにするには、Document オブジェクトを作成するスクリプトを Open Application イベントに関連付けます。Notifier について詳しくは、[48 ページの「Notifier オブジェクトの使用」](#)を参照してください。

Count Item クラス

Count Item オブジェクトは、カウントツールをスクリプトで操作するためのオブジェクトです。

Color Sampler クラス

Color Sampler オブジェクトは、カラーサンプラーツールをスクリプトで操作するためのオブジェクトです。

Measurement Scale クラス

Measurement Scale オブジェクトは、新しく導入された計測スケール機能をスクリプトで操作するためのオブジェクトです。この機能を使用すると、ドキュメントのスケールを設定することができます。

包含階層と Photoshop のユーザインターフェイス

次の表に、各オブジェクトと Photoshop のユーザインターフェイスとの関係を示します。

オブジェクト名	説明	通常の操作で作成する方法
Application	Photoshop アプリケーション。	Photoshop アプリケーションを起動します。
Document	レイヤー、チャンネル、アクションなどを作成する作業オブジェクト。スクリプトでは、アプリケーションで行う場合と同様に、ドキュメントに名前を付けたり、ドキュメントを開いたり、保存したりできます。	Photoshop で、 ファイル／新規 または ファイル／開く を選択します。
Selection	レイヤーやドキュメントの選択された領域。	選択ツールやなげなわツールを選択して、マウスをドラッグします。

オブジェクト名	説明	通常の操作で作成する方法
Path Item	シェイプのアウトライン、直線、曲線などの描画オブジェクト。	パス選択ツールやペンツールを選択して、マウスでパスを描画します。
Channel	画像のカラーに関するピクセル情報。	ウィンドウ／チャンネルを選択します。
Art Layer	画像内の1つの要素を、その画像内の他の要素に影響を与えることなく操作できるようにするための、ドキュメント内のレイヤークラス。	レイヤー／新規／レイヤーまたはウィンドウ／レイヤーを選択します。
Layer Set	Art Layer オブジェクトのコレクション。Photoshop UI での現在の名前は「グループ」です。「レイヤーセット」は、以前のバージョンの Photoshop UI で使用されていた名前です。オブジェクト名は、下位互換性を保つために変更されていません。	レイヤー／新規／グループを選択します。
Layer Comp	ドキュメント内のレイヤーの状態のスナップショット。	ウィンドウ／レイヤーカンパを選択し、新規レイヤーカンパアイコンを選択します。
Document Info	Document オブジェクトに関するメタデータ。 注: メタデータとは、ファイルの内容や特徴について記述した任意のデータのことです。ファイル名、作成日時、作成者名、ファイルに格納されている画像の名前などがあります。	ファイル／ファイル情報を選択します。
Notifier	イベントの発生時にスクリプトに対して通知を行います。その後、イベントはスクリプトの実行をトリガします。例えば、ユーザが「OK」をクリックしたら、次に行う処理を Notifier オブジェクトでスクリプトに指示できます。	ファイル／スクリプト／スクリプトイベントマネージャを選択します。
Preferences	アプリケーションの環境設定。	編集／環境設定 (Windows の場合) または Photoshop ／環境設定 (Macintosh の場合) を選択します。
History State	保存するたびに、そのときまでのドキュメントの状態を記憶します。 注: History State オブジェクトを使用して、Selection オブジェクトを塗りつぶしたり、ドキュメントを以前の状態に戻したりすることができます。	ウィンドウ／ヒストリーを選択し、次にヒストリーパレットからヒストリー状態を選択します。
Color Sampler	ドキュメントのカラーサンプラーを表します。	カラーサンプラーツールを選択し、ドキュメント内でクリックします。
Count Item	ドキュメント内のカウントされるアイテムを表します。	カウントツールを選択し、ドキュメント内でクリックします。
Measurement Scale	ドキュメントの計測スケールを表します。	Measurement Scale オブジェクトを作成することはできませんが、 解析／計測スケール／カスタム を使用してそのプロパティを変更できます。

その他のオブジェクト

Photoshop のオブジェクトモデルには、包含階層を構成するオブジェクト以外のオブジェクトも含まれています。それらのクラスの多くは、プロパティの型として使用されたり、コマンドやメソッドに情報を提供するための引数として使用されます。例えば、次のようになります。

- ▶ `color value(SolidColor / SolidColor)` クラスは、Application オブジェクトの `background color (backgroundColor / backgroundColor)` プロパティや `foreground color (ForegroundColor / foregroundColor)` プロパティの型として使用されます。[51 ページの「カラーオブジェクトの操作」](#)を参照してください。
- ▶ ドキュメントのオープンオプションや保存オプションはクラスとして定義されており、ドキュメントを開くコマンドや保存するコマンドに渡されます。例えば、BMP save options (`BMPSaveOptions / BMPSaveOptions`) クラスは、`save (saveAs / saveAs)` コマンドまたはメソッドに引数として渡されます。[28 ページの「ドキュメントのオープン」](#)および [31 ページの「ドキュメントの保存」](#)を参照してください。

定数

JavaScript や VBScript の Photoshop オブジェクトモデルには、**定数**という重要な構成要素が含まれています。定数とは、プロパティを定義する値の型のことです。例えば、Art Layer オブジェクトの `kind` プロパティでは、Photoshop で許可されている特定の種類のみを定義できます。定数に関する一般的な情報については、『Adobe Intro to Scripting』を参照してください。

注: このマニュアルでは、VBScript の列挙型の実際の値を、次の形式で示しています。

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

説明の先頭にある `'` は**コメント**の開始を意味するもので、`'` の右にあるテキストはスクリプティングエンジンから無視されます。コメントの使用方法について詳しくは、『Adobe Intro to Scripting』を参照してください。

例えば、『Adobe Photoshop CS6 JavaScript Scripting Reference』や『Adobe Photoshop CS6 Visual Basic Scripting Reference』で、ArtLayer オブジェクトについて解説しているページを開きます。このオブジェクトには `Kind (kind)` というプロパティがあります。そのプロパティの値の型はリンクになっています。このリンクをクリックすると、そのプロパティで指定可能な値を定義する定数が確認できます。この定数は、VBScript では `PSLayerKind`、JavaScript では `LayerKind` です。リンクをクリックして、`kind` プロパティの定義に使用できる値を確認してみてください。

注: 複数のオブジェクトで、異なる定数値を取る同名のプロパティが使用されていることがあります。Channel オブジェクトの `kind` プロパティの定数値は、Art Layer オブジェクトの `kind` プロパティの定数値とは異なります。

サンプル Hello World スクリプトの作成

では、Photoshop で使用できる非常に単純なスクリプトを 3 つのスクリプト言語で作成してみましょう。ここでの課題は、「Hello World」というメッセージを表示することです。これは、あらゆるプログラミング環境で最初に出されるおなじみの課題です。

Hello World スクリプトは、次のことを行います。

1. Photoshop アプリケーションを開きます。
2. 新しい Document オブジェクトを作成します。

ドキュメントを作成するときに、docRef という名前の変数も作成し、docRef の値としてドキュメントへの参照を割り当てます。ドキュメントは幅 4 インチ、高さ 2 インチになります。

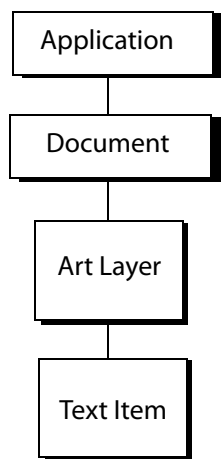
3. Art Layer オブジェクトを作成します。

このスクリプトでは、artLayerRef という名前の変数を作成し、artLayerRef の値として Art Layer オブジェクトへの参照を割り当てます。

4. artLayerRef をテキストアイテムとして定義します。
5. テキストアイテムの内容を「Hello World」に設定します。

注: また、スクリプト全体にコメントを追加します。これは最初に作成するスクリプトなので、コメントを多めに使用します。

これらの手順は、次に示す包含階層の個々のパスに対応します。



AppleScript の作成と実行

この手順を完了するには、Apple® 社のスクリプトエディタアプリケーションを開く必要があります。

注: スクリプトエディタのデフォルトの場所は、**アプリケーション／AppleScript／スクリプトエディタ**です。

初めての Photoshop AppleScript を作成して実行するには:

1. 次のスクリプトをスクリプトエディタに入力します。

注: 最初に「--」がある行はコメントです。コメントの入力はオプションです。

```
-- Sample script to create a new text item and
-- change its contents.
--target Photoshop CS6
tell application "Adobe Photoshop CS6"

    -- 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. 「**実行**」をクリックして、スクリプトを実行します。新規ドキュメントが作成され、新規レイヤーが追加され、そのレイヤーの種類がテキストに変更された後、テキストが「Hello, World」に設定されます。

注: エラーが発生した場合は、『Adobe Intro to Scripting』の AppleScript のデバッグに関する節を参照してください。

VBScript の作成と実行

Photoshop ドキュメント内に **Hello World!** というテキストを表示する VBScript を作成して実行するには、次の手順に従います。

初めての Photoshop VBScript を作成して実行するには：

1. 次のスクリプトをスクリプトエディタまたはテキストエディタに入力します。

注：コメントの入力はオプションです。

```
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. ファイルを .vbs という拡張子のテキストファイルとして保存します。
3. Windows Explorer でファイルをダブルクリックして、スクリプトを実行します。

スクリプトによって Photoshop が開かれます。

JavaScript の作成と実行

Photoshop ドキュメント内に **Hello World!** というテキストを表示する JavaScript を作成して実行するには、次の手順に従います。

JavaScript の実行には Photoshop を使用するの、スクリプトの先頭に Photoshop を開くコードを含める必要はありません。

注：Photoshop では、JavaScript に基づいてアドビ システムズ社が開発した ExtendScript というスクリプト言語が使用されています。ExtendScript の #target というコマンドを使用して Photoshop アプリケーションをターゲットにすれば、ファイルシステムのどの場所からその JavaScript を開いても Photoshop を操作することができます。詳しくは、『JavaScript Tools Guide CS6』の Script UI の章を参照してください。

初めての Photoshop JavaScript を作成して実行するには：

1. 次のスクリプトを入力します。

注：コメントの入力はオプションです。

```
// 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. ファイルを、.jsx という拡張子のテキストファイルとして、Adobe Photoshop CS6 ディレクトリの Presets¥Scripts フォルダに保存します。

注：スクリプトを Photoshop の **ファイル／スクリプトメニュー** からアクセスできるようにするためには、JavaScript を Presets¥Scripts フォルダ内に配置する必要があります。アプリケーションを再起動するまで、スクリプトは **ファイル／スクリプトメニュー** に表示されません。

注：Photoshop では、.js という拡張子の JavaScript ファイルもサポートされています。

3. 次のいずれかを行います。
 - Photoshop が既に開いている場合は、**ファイル／スクリプト／参照**を選択し、Presets フォルダ内の Scripts フォルダに移動してスクリプトを選択します。
 - Photoshop を起動または再起動し、**ファイル／スクリプト**を選択した後、**スクリプトメニュー**からスクリプトを選択します。

3 Photoshop のスクリプティング

この章では、Photoshop 用のスクリプトを作成するときに役立つ、Photoshop ドキュメントオブジェクトモデル (DOM) の様々な活用テクニックを紹介します。

また、Photoshop 用の AppleScript、VBScript、JavaScript を作成するときに使用するオブジェクト、クラス、プロパティ、コマンド、値 (**定数**または**列挙型**) に関する情報を、リファレンスマニュアルやオブジェクトモデルブラウザで調べる方法も紹介します。

ヒント: この章の各節では、スクリプトの作成方法に加えて、そのスクリプトで使用されている特定のエレメントについて調べる方法も説明しています。この説明に従って学習を進めていけば、Photoshop スクリプトの作成方法をスムーズに習得することができます。

Photoshop のオブジェクト、コマンド、メソッドの表示

インストールされるリファレンスマニュアルには、3 種類の各スクリプト言語用に、Photoshop のリファレンス情報が含まれています。

- 『Adobe Photoshop CS6 AppleScript Scripting Reference』
- 『Adobe Photoshop CS6 Visual Basic Scripting Reference』
- 『Adobe Photoshop CS6 JavaScript Scripting Reference』

また、各言語の開発環境に用意されているオブジェクトモデルブラウザを使用して、リファレンス情報を調べることもできます。

- AppleScript の場合は、AppleScript スクリプトエディタを使用して Photoshop AppleScript 用語説明を表示します。
- VBScript の場合は、Microsoft Word の VBA エディタ、Visual Basic のオブジェクトブラウザ、または Visual Studio を使用します。
- JavaScript の場合は、ExtendScript のオブジェクトモデルビューアを使用します。詳しくは、『JavaScript Tools Guide CS6』を参照してください。

Photoshop の AppleScript 用語説明の表示

用語説明は、Apple 社のスクリプトエディタアプリケーションを使用して表示します。

注: スクリプトエディタのデフォルトの場所は、**アプリケーション／AppleScript／スクリプトエディタ**です。

AppleScript 用語説明を表示するには:

1. スクリプトエディタで、**ファイル／用語説明を開く**を選択します。
用語説明を開くダイアログボックスが表示されます。
2. Adobe Photoshop CS6 を選択し、「**開く**」をクリックします。

スクリプトエディタによって Photoshop が開かれ、Photoshop の用語説明が表示されます。オブジェクトとそのオブジェクトのコマンド、プロパティ、エレメントが表示されます。各コマンドのパラメータも表示されます。

注: Photoshop AppleScript 用語説明には、open コマンドや save コマンドで扱えるすべてのファイル形式は表示されません。

Photoshop のタイプライブラリの表示 (VBS)

Microsoft Word の VBA エディタを使用して、VBScript で使用可能な Photoshop のオブジェクトやコマンドを表示できます。

Microsoft Word で VBS オブジェクトライブラリを表示するには：

1. Word を起動し、ツール／マクロ／**Visual Basic Editor** を選択します。
2. **ツール／参照設定**を選択し、「Adobe Photoshop Type Library」チェックボックスを選択して「**OK**」をクリックします。
3. **表示／オブジェクトブラウザ**を選択します。
4. 左上のプルダウンメニューに表示される、現在開いているライブラリのリストから、「**Photoshop CS6 type library**」を選択します。
5. オブジェクトクラスを選択すると、クラスに関する詳細が表示されます。

また、Visual Basic 開発環境のオブジェクトブラウザを使用して、VBScript で使用可能な Photoshop のオブジェクトやコマンドを表示することもできます。

Visual Basic 開発環境で VBS オブジェクトライブラリを表示するには：

1. Visual Studio 2005 または Visual Basic を起動します。
2. **表示／オブジェクトブラウザ**を選択します。
3. 参照ドロップダウンボックスで、「**カスタムコンポーネントセットの編集**」を選択します。
4. 「COM」タブで、「Adobe Photoshop CS6 Object Library」を見つけて選択します。
5. 「**追加**」をクリックします。ウィンドウの「選択されたプロジェクトとコンポーネント」に、選択したライブラリが表示されます。
6. 「**OK**」をクリックします。
7. これで、Photoshop ライブラリがオブジェクトブラウザに読み込まれました。Photoshop ライブラリのアイコンの横にあるプラス記号をクリックします。
8. Photoshop オブジェクトのアイコンの横にあるプラス記号をクリックします。
9. Photoshop ライブラリで定義されているオブジェクトが一覧表示されます。いずれかを選択すると、そのクラスの詳細が表示されます。

Application オブジェクトの指定と参照

AppleScript や VBScript のスクリプトは Photoshop アプリケーションの外部で実行するので、使用しているコマンドが Photoshop のものであることを、スクリプトの最初の部分で明示する必要があります。

注: JavaScript のスクリプトは Photoshop アプリケーションの内部から実行するので、Application オブジェクトをターゲットにする必要はありません（[19 ページの「JavaScript の作成と実行」](#)を参照してください）。

AS AppleScript で Photoshop をターゲットにするには、次の文でスクリプトを囲む必要があります。

```
tell application "Adobe Photoshop CS6"  
...  
end tell
```

注: すべてのコマンドを **tell** ブロックの中に記述すれば、各コマンドで Application オブジェクトを参照する手間が省けます。

VBS VBScript でアプリケーションをターゲットにするには、次のようにします。

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

JS JavaScript では、Application オブジェクトを参照する必要がないので、アプリケーションのすべてのプロパティやメソッドに修飾なしでアクセスできます。包含階層の一部としてアプリケーションへの参照を含めるかどうかは自由です。スクリプトが読みやすくなる方を選択してください。

Application オブジェクトを参照するには、クラス名ではなく、app という定義済みのグローバルオブジェクトを使用します。

次の文は同意です。

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

および

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

注: このマニュアルの多くの JavaScript サンプルでは、Application オブジェクトの参照は行っていません。

スクリプトでの新規オブジェクトの作成

Photoshop アプリケーションで新規ドキュメントを作成するには、**ファイル** / **新規**を選択します。レイヤー、チャンネル、パスなどのその他のオブジェクトをドキュメント内に作成するには、ウィンドウメニューを使用するか、適切なパレットを表示し、**新規**アイコンを選択します。ここでは、これらの操作をスクリプトで実行する方法について説明します。

スクリプトでオブジェクトを作成するには、作成するオブジェクトのタイプを指定して、次のコマンドを使用します。

➤ AS: make

➤ VBS: Add

➤ JS: add()

[11 ページの「Photoshop オブジェクトモデル」](#)に示されているように、Application、Notifier、Preferences オブジェクト以外のオブジェクトは、すべて Document オブジェクトに格納されています。したがって、Document オブジェクトと Notifier オブジェクト以外のオブジェクトをスクリプトに追加する場合は、Document オブジェクトを参照する必要があります (Preferences オブジェクトは新たに追加できません)。

注: VBScript や JavaScript でオブジェクトのタイプを指定するときは、オブジェクトのコレクション名を使用します。例えば、ドキュメントを追加するときは Documents コレクションを使用し、アートレイヤーを追加するときは art layers コレクションを使用します。エレメントやコレクションについて詳しくは、『Adobe Intro to Scripting』を参照してください。

AS 次の AppleScript 文を実行すると、Document オブジェクトが作成されます。

```
make new document
```

set コマンドを使用して、新規ドキュメントへの参照を保持する変数を作成することもできます。次の例の docRef という名前の変数には、新規ドキュメントへの参照が保持されます。

```
set docRef to make new document
```

ドキュメント以外のオブジェクトを作成するときは、そのオブジェクトを格納する Document オブジェクトを参照する必要があります。次のサンプルでは、docRef という名前の変数に保持されているドキュメントに、アートレイヤーが作成されます。

```
make new art layer in docRef
```

注: VBScript や JavaScript のオブジェクトがコレクションに追加されるのと同じように、AppleScript でも、作成したオブジェクトはエレメントに追加されます。ただし、AppleScript の make 文や set 文では、エレメント名は暗黙的に仮定されています。例えば、次の文は、

```
make new document
```

実際には次を意味します。

```
make new document in the documents element
```

AppleScript でのオブジェクトの作成について詳しくは、次の箇所を調べてください。

- 『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で、make コマンドや set コマンドを調べます。21 ページの「[Photoshop の AppleScript 用語説明の表示](#)」を参照してください。
- 特定のオブジェクトで利用できるコマンドを知りたい場合は、『Adobe Photoshop CS6 AppleScript Scripting Reference』でそのオブジェクトを調べます。オブジェクトに有効なコマンドがある場合は、オブジェクトの説明の最後に、有効なコマンドのリストが示されています。

VBS VBScript では、Add メソッドを使用できるのはコレクション名のみです。Add メソッドは、コレクションオブジェクト以外のオブジェクトでは無効です。また、VBScript では、オブジェクトを作成したり参照したりするときに、Application オブジェクトを参照する必要があります。

例えば、VBScript でドキュメントを作成する場合、次のサンプルのようにオブジェクト名を使用しても、**Document** オブジェクトは作成できません。

```
appRef.Document.Add()
```

次のように、コレクション名を使用する必要があります。コレクション名はオブジェクト名の複数形です。

```
appRef.Documents.Add()
```

注: このサンプル文では、appRef という名前の変数で Application オブジェクトを参照しています。詳しくは、22 ページの「[Application オブジェクトの指定と参照](#)」を参照してください。

ArtLayer オブジェクトを追加するには、Application オブジェクトと、アートレイヤーを格納する Document オブジェクトの両方を参照する必要があります。次のサンプルでは、appRef という変数を使用して Application オブジェクトを参照しています。また、ドキュメント名を使用せずに、ドキュメントのインデックスを使用して Document オブジェクトを参照しています。

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

注: Photoshop の VBScript コレクションのインデックスは、0 ではなく 1 から数えます。つまり、作成される最初のドキュメントは、インデックス 0 ではなく、インデックス 1 になります。

『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザを調べると、Document オブジェクトには Add() メソッドがなく、Documents オブジェクトには Add() メソッドがあることがわかります。同様に、ArtLayer オブジェクトには Add() メソッドがなく、ArtLayers オブジェクトにはあることがわかります。

注: Layers オブジェクトは例外です。このオブジェクトもコレクションですが、Add() メソッドはありません。Layers コレクションには ArtLayer オブジェクトと LayerSet オブジェクトの両方が含まれています。これらのオブジェクトを作成するには、ArtLayers コレクションや LayerSets コレクションの Add メソッドを使用します。詳しくは、『Adobe Photoshop CS6 Visual Basic Scripting Reference』で Layers オブジェクトを調べてください。

JS JavaScript では、add() メソッドを使用できるのはコレクション名のみです。add() メソッドは、コレクションオブジェクト以外のオブジェクトでは無効です。

VBScript と同じように、次の JavaScript 文でドキュメントを作成できます。

```
documents.add()
```

次の文は誤りです。

```
document.add()
```

注: Application オブジェクトへの参照を含めるかどうかは自由です。次の文は、前のサンプルと同意です。

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

ArtLayer オブジェクトを追加するには、レイヤーを格納する Document オブジェクトを参照し、Document の artLayers プロパティを使用して、ArtLayers コレクションの add() メソッドを呼び出す必要があります。

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

VBScript と同様に、JavaScript の Document オブジェクトに add() メソッドはありませんが、Documents オブジェクトにはあります。同様に、ArtLayer オブジェクトに add() メソッドはありませんが、ArtLayers オブジェクトにはあります。

注: Layers コレクションオブジェクトに add() メソッドはありません。詳しくは、『Adobe Photoshop CS6 JavaScript Scripting Reference』で Layers オブジェクトを調べてください。

アクティブなオブジェクトの設定

Photoshop アプリケーションでオブジェクトを操作するには、そのオブジェクトを最前面にするか、アクティブなオブジェクトにする必要があります。例えば、レイヤーを操作するには、そのレイヤーを最前面にする必要があります。

この規則はスクリプトにも当てはまります。複数のドキュメントが開いている場合、スクリプトのコマンドやメソッドはアクティブなドキュメントに対して実行されます。したがって、目的のドキュメントに対してコマンドが実行されるように、アクティブなドキュメントを指定してからコマンドまたはメソッドを実行するのが最も確実です。

アクティブなオブジェクトを設定する方法は、次のとおりです。

- AppleScript では、親オブジェクトの current プロパティを使用します。
- VBScript では、親オブジェクトの ActiveObject プロパティ (ActiveDocument や ActiveLayer など) を使用します。
- JavaScript では、親オブジェクトの activeObject プロパティ (activeDocument や activeLayer など) を使用します。

注: 親オブジェクトとは、目的のオブジェクトが格納されているオブジェクトのことです。例えば、ドキュメントの親はアプリケーションです。レイヤー、選択範囲、チャンネルの親はドキュメントです。

例えば、『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで Application オブジェクトを調べると、activeDocument というプロパティがあることがわかります。また、Document オブジェクトを調べると、activeLayer や activeHistoryState というプロパティがあることがわかります。同様に、『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で application を調べると、current というプロパティがあることがわかります。

アクティブなオブジェクトを設定するサンプルスクリプトについては、次の各節を参照してください。

- [26 ページの「アクティブなドキュメントの設定」](#)
- [27 ページの「アクティブなレイヤーの設定」](#)
- [28 ページの「アクティブなチャンネルの設定」](#)

アクティブなドキュメントの設定

アクティブなドキュメントの設定方法を、次の例に示します。

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
```

アクティブなレイヤーの設定

Document オブジェクトの current layer (ActiveLayer / activeLayer) プロパティを使用してアクティブなレイヤーを設定する方法を、次の例に示します。ドキュメントのアクティブなレイヤーを設定するには、そのドキュメントが現在のドキュメントであることが必要です。

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

注: Photoshop では、デフォルトで「レイヤー 1」、「レイヤー 2」のようにレイヤー名が付けられます。

```
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)
```

『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザで、Document オブジェクトの ActiveLayer プロパティを調べてください。

注: レイヤー名でレイヤーを指定することもできます。Photoshop では、デフォルトで「レイヤー 1」、「レイヤー 2」のようにレイヤー名が付けられます。[38 ページの「ArtLayer オブジェクトの参照」](#)を参照してください。

```
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]
```

『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで、Document オブジェクトの activeLayer プロパティを調べてください。

注: レイヤー名でレイヤーを指定することもできます。Photoshop では、デフォルトで「レイヤー 1」、「レイヤー 2」のようにレイヤー名が付けられます。[38 ページの「ArtLayer オブジェクトの参照」](#)を参照してください。

アクティブなチャンネルの設定

Document オブジェクトの `current channels` (`ActiveChannels` / `activeChannels`) プロパティには、チャンネルの配列を値として渡すことができます。これは、複数のチャンネルを同時にアクティブにできることを表しています。ドキュメントのアクティブなチャンネルを設定するには、そのドキュメントがアクティブなドキュメントであることが必要です。

AS チャンネルの配列を使用して、1 番目と 3 番目のチャンネルをアクティブにします。

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

Document オブジェクトの `component channels` プロパティを使用して、すべてのコンポーネントチャンネルを選択することもできます。

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

VBS チャンネルの配列を使用して、1 番目と 3 番目のチャンネルを、アクティブなドキュメントのアクティブなチャンネルに設定します。

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

Document オブジェクトの `ComponentChannels` プロパティを使用して、すべてのコンポーネントチャンネルを選択することもできます。

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

JS チャンネルの配列を使用して、1 番目と 3 番目のチャンネルをアクティブにします。

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

Document オブジェクトの `componentChannels` プロパティを使用して、すべてのコンポーネントチャンネルを選択することもできます。

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

ドキュメントのオープン

Application オブジェクトの `open` / `Open` / `open()` コマンドを使用して、既存のドキュメントを開くことができます。このコマンドの引数には、ドキュメント名（ドキュメントファイルへのパス）を渡す必要があります。

デフォルトファイル形式のファイルのオープン

`open` / `Open` / `open()` コマンドでは、Photoshop でサポートされている様々なファイル形式に対応できるように、開くドキュメントの形式を指定することができます。形式を指定しなかった場合は、Photoshop によってファイルのタイプ（デフォルト形式）が推測されます。次の例では、最も適切な形式が推測されてドキュメントが開かれます。

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

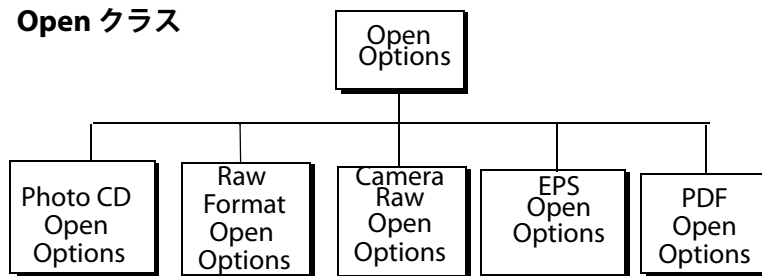
VBS `fileName = "C:¥MyFile"`
 `Set docRef = appRef.Open(fileName)`

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

JavaScript では、File オブジェクトを作成して、そのオブジェクトへの参照を `open()` コマンドに渡す必要があります。

ファイルを開く形式の指定

Open クラス



次のドキュメントタイプでは、ドキュメントを開く方法をオプションで指定できます。例えば、ドキュメントを開くウィンドウの高さや幅、複数のページが含まれているファイルのどのページを開くかななどを指定できます。

- PhotoCD
- CameraRaw
- RawFormat
- Adobe PDF
- EPS

それぞれのファイルタイプで設定できるオプションを確認するには、ファイル形式の名前を先頭に持つ **OpenOptions** オブジェクトのプロパティを調べます。例えば、次のようになります。

- 『Adobe Photoshop CS6 AppleScript Scripting Reference』で、Photo CD open options クラスや EPS open objects クラスを調べます。
- 『Adobe Photoshop CS6 Visual Basic Scripting Reference』または『Adobe Photoshop CS6 JavaScript Scripting Reference』で、PhotoCDOpenOptions オブジェクトや EPSOpenOptions オブジェクトを調べます。

次の例では、複数のページと複数の画像が含まれている一般的な PDF ドキュメントを、以下の設定で開きます。

- RGB モードで、72 ピクセル/インチの解像度でドキュメントを開きます。
- アンチエイリアスを使用して、ドキュメント内の画像のエッジをより滑らかに表示します。

- ドキュメントの3ページ目を開きます。

```
AS tell application "Adobe Photoshop CS6"
    set myFilePath to alias "OS X 10.5.8 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 **注:** ExtendScript の File オブジェクトの引数は、Universal Resource Identifier (URI) の表記法で指定します。詳しくは、『JavaScript Tools Guide CS6』を参照してください。

```
// 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/pdf/files/myfile.pdf")

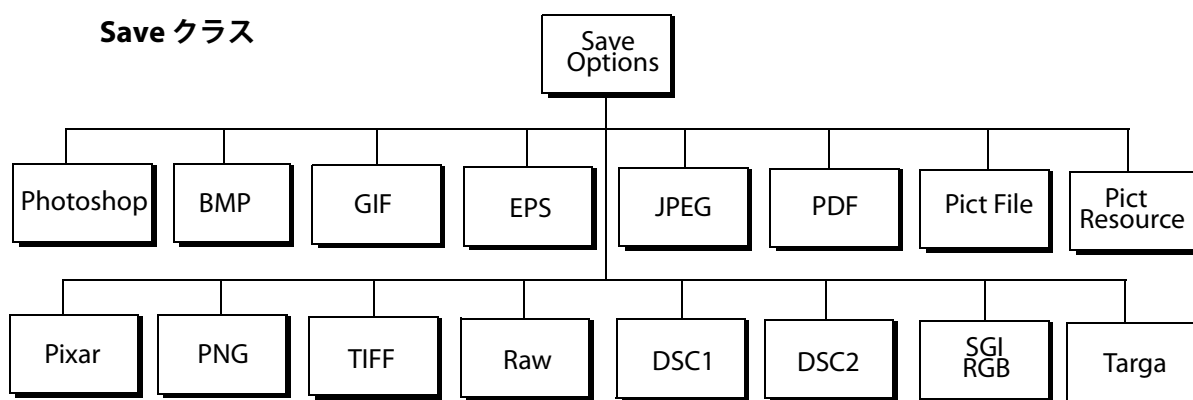
// Create a PDF option object
var pdfOpenOptions = new PDFOpenOptions
pdfOpenOptions.antiAlias = true
pdfOpenOptions.mode = OpenDocumentMode.RGB
pdfOpenOptions.resolution = 72
pdfOpenOptions.page = 3
// open the file
app.open( fileRef, pdfOpenOptions )

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

ドキュメントの保存

Photoshop でドキュメントを保存するときに使用できるオプションを、次の図に示します。それぞれのファイル形式の保存オプションで指定できるプロパティを確認するには、ファイル形式の名前を先頭に持つオブジェクトを調べます。例えば、.eps ファイルの保存に使用できるプロパティを確認するには、次のようにします。

- 『Adobe Photoshop CS6 AppleScript Scripting Reference』で、EPS save options クラスを調べます。
- 『Adobe Photoshop CS6 Visual Basic Scripting Reference』または『Adobe Photoshop CS6 JavaScript Scripting Reference』で、EPSSaveOptions を調べます。



注: Open で扱える形式と Save で扱える形式が異なることに注意してください。違いを比較するには、[29 ページの「ファイルを開く形式の指定」](#)を参照してください。

注: 次のオプションの形式は、明示的にインストールされている場合にのみ使用可能です。

- Alias PIX
- Electric Image
- SGI RGB
- Wavefront RLA
- SoftImage

次のスクリプトでは、ドキュメントを .jpeg ファイルとして保存します。

AS

```

tell application "Adobe Photoshop CS6"
    make new document
    set myFile to "OS X 10.5.8 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)
```

アプリケーションの環境設定

アプリケーションの環境設定（カラーピッカー、ファイルの保存オプション、ガイド・グリッド・スライスの設定など）を、スクリプトで指定することができます。

注: settings クラス／Preferences オブジェクトに用意されている各プロパティは、Photoshop CS6 の環境設定ダイアログボックスに表示される各オプションに対応しています。このダイアログボックスは、Photoshop の編集／環境設定（Windows）、または **Photoshop**／環境設定（Macintosh）を選択すると表示されます。環境設定の個々の設定について詳しくは、Photoshop ヘルプを参照してください。

AS

AppleScript では、settings クラスのプロパティを使用して、アプリケーションの環境設定を指定します。次のスクリプトでは、定規単位と文字単位を設定します。

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

使用可能なすべての設定プロパティを確認するには、『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で、settings-object クラスを調べてください。

VBS

Preferences オブジェクトは Application オブジェクトのプロパティです。したがって、VBScript で Preferences オブジェクトを使用するには、Application オブジェクトを参照する必要があります。

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

使用可能なすべての設定プロパティを確認するには、『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザで、Preferences オブジェクトを調べてください。また、Application オブジェクトの Preferences プロパティも調べてください。

JS Preferences オブジェクトは Application オブジェクトのプロパティです。

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

使用可能なすべての設定プロパティを確認するには、『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで、Preferences オブジェクトを調べてください。また、Application オブジェクトの preferences プロパティも調べてください。

ダイアログボックスの有効化と無効化

スクリプトでダイアログボックスを適切に制御することは、非常に重要です。ダイアログボックスが表示されると、ユーザがダイアログボックスを閉じるまでスクリプトが中断されます。コンピュータの前にユーザがいることを前提にしている対話型のスクリプトでは、ダイアログボックスが表示されても問題はありませんが、監視されていない状況で（バッチモードで）実行するスクリプトの場合は、ダイアログボックスが表示されてスクリプトが中断されるのを防ぐ必要があります。

Application オブジェクトの displayDialogs (DisplayDialogs / displayDialogs) プロパティを使用すれば、ダイアログボックスを表示するかどうかを制御できます。

注: スクリプトでダイアログボックスを使用することは、Photoshop のアクションで中止を挿入することに似ています。

AS 次のスクリプトを実行すると、ダイアログボックスが表示されなくなります。

```
set displayDialogs to never
```

displayDialogs プロパティで利用できる値を確認するには、『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で、application クラスを調べてください。

VBS Application オブジェクトの DisplayDialogs プロパティを使用して、ダイアログボックスを制御します。

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

DisplayDialogs が Application オブジェクトのプロパティであることに注意してください。このプロパティを設定するには Application オブジェクトを参照する必要があります。

『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザで、Application オブジェクトの DisplayDialogs プロパティを調べると、このプロパティの値の型は PsDialogModes という定数であることがわかります。PsDialogModes のオプションも調べてください。

JS Application オブジェクトの displayDialogs プロパティを使用して、ダイアログボックスを制御します。

```
displayDialogs = DialogModes.NO
```

『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで、Application オブジェクトの displayDialogs プロパティと、DialogModes という定数を調べてください。

Photoshop のオブジェクトモデルの操作

ここでは、Photoshop のオブジェクトモデルを構成する各オブジェクトの操作について説明します。オブジェクトモデルについて詳しくは、『Adobe Intro to Scripting』および [11 ページの「Photoshop オブジェクトモデル」](#)を参照してください。

Application オブジェクトの使用

ここでは、スクリプトで Application オブジェクトをいつ、どのように使用するかについて説明します。Application オブジェクトのいくつかのプロパティの使用方法についても説明します。

Application オブジェクトのプロパティやコマンドを使用すると、Photoshop の次のような機能やオブジェクトを操作できます。

- **Photoshop のグローバル設定や環境設定** - 単位やカラーなどの設定。 [32 ページの「アプリケーションの環境設定」](#) を参照してください。
- **ドキュメント** - ドキュメントの追加、ドキュメントのオープン、アクティブなドキュメントの設定。 [28 ページの「ドキュメントのオープン」](#) および [25 ページの「アクティブなオブジェクトの設定」](#) を参照してください。
- **アクション** - スクリプトやアクションパレットで作成された、Photoshop アプリケーションのアクションを実行できます。 [73 ページの「アクションマネージャ」](#) を参照してください。

Application オブジェクトのプロパティを使用して、次のような情報を取得できます。

- システムにインストールされているフォントのリスト
 - AS: theFonts を fonts に設定します。

注: AppleScript では、fonts は独立したコレクションとして扱われます。したがって、fonts を取得するために application オブジェクトを参照する必要はありません。
 - VBS: 次のように設定します。 fontsInstalled = AppRef.Fonts
 - JS: var fontsInstalled = app.fonts
- Adobe Photoshop で使用できる未使用メモリの容量。 Application オブジェクトの free memory (FreeMemory / freeMemory) プロパティを使用します。
- Preferences フォルダの場所。 Application オブジェクトの preferences folder (PreferencesFolder / preferencesFolder) プロパティを使用します。

詳しくは、各言語のリファレンスマニュアルやオブジェクトブラウザで、Application オブジェクトのプロパティを調べてください。

Document オブジェクトの使用

Document オブジェクトは、Photoshop で開いている任意のドキュメントを表します。Document は 1 つのファイルまたは 1 つのカンバスと考えることができます。Document オブジェクトを使用して、次の操作を実行できます。

- Document オブジェクトに格納されているスクリプトオブジェクト (ArtLayer オブジェクトや Channel オブジェクトなど) へのアクセス。詳しくは、 [11 ページの「Photoshop オブジェクトモデル」](#) を参照してください。
- コマンドやメソッドを使用した、特定の Document オブジェクトの操作。 Document オブジェクトを使用すれば、カンバスの切り抜き、回転、反転が行えます。また、カンバスや画像のサイズ変更、画像のトリミングなども行えます。詳しくは、 [35 ページの「Document オブジェクトの操作」](#) の例を参照してください。
- アクティブなレイヤーの取得。 [27 ページの「アクティブなレイヤーの設定」](#) を参照してください。
- 現在のドキュメントの保存。 [31 ページの「ドキュメントの保存」](#) を参照してください。
- アクティブなドキュメント内や異なるドキュメント間でのコピー＆ペースト。 [54 ページの「クリップボード操作の理解」](#) を参照してください。

Document オブジェクトの操作

以下の操作を実行する方法を、次の例に示します。

- 定規単位の既存の設定を保存して、定規単位をインチに設定する。
- 既存のファイルをドキュメントとして開く (Ducky.tif というファイルを使用)。
- 画像のサイズを幅 4 インチ、高さ 4 インチに変更する。
- ドキュメントウィンドウ (カンバス) のサイズを、高さ 4 インチ、幅 4 インチに変更する。
- 画像の上端と下端をトリミングする。
- 画像を切り抜く。
- ウィンドウ全体を反転する。
- 定規単位を元に戻す。

注: 定規単位について詳しくは、[32 ページの「アプリケーションの環境設定」](#)を参照してください。

AS

```
tell application "Adobe Photoshop CS6"
    set saveUnit to ruler units of settings
    set ruler units of settings to inch units
    set duckFile to alias ~
        "OS X 10.5.8 US:Applications:Adobe Photoshop CS6: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

```

レイヤーオブジェクトの操作

Photoshop のオブジェクトモデルには、2 種類のレイヤーオブジェクトが用意されています。

- ▶ ArtLayer オブジェクトは、画像コンテンツを格納することができます。基本的には、Photoshop アプリケーションのレイヤーと同じです。

注: kind プロパティを使用して ArtLayer オブジェクトのタイプをテキストレイヤーに設定すると、ArtLayer オブジェクトにテキストを格納できるようになります。

- ▶ Layer Set オブジェクトは、0 個以上の ArtLayer オブジェクトを格納できます。

レイヤーを作成するときには、ArtLayer を作成するのか Layer Set を作成するのかを指定する必要があります。

注: ArtLayer オブジェクトと LayerSet オブジェクトには、ArtLayers および LayerSets という対応するコレクションオブジェクトがあります。これらのコレクションオブジェクトには、add/Add/add() コマンドが用意されています。Layers コレクションオブジェクトでは、ArtLayer オブジェクトや LayerSet オブジェクトの参照は行えますが、追加は行えません。他のコレクションオブジェクトと異なり、Layers コレクションオブジェクトには add/Add/add() コマンドは用意されていません。

ArtLayer オブジェクトの作成

次の例では、赤色で塗りつぶされた ArtLayer オブジェクトを、現在のドキュメントの先頭に作成します。

AS

```
tell application "Adobe Photoshop CS6"
    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)
```

Layer Set オブジェクトの作成

現在のドキュメントに最初の ArtLayer オブジェクトを作成し、その後に Layer Set オブジェクトを作成する方法を、次の例に示します。

```
AS tell application "Adobe Photoshop CS6"
    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)
```

ArtLayer オブジェクトの参照

Photoshop アプリケーションで (スクリプトを使用せずに) レイヤーを作成すると、レイヤーパレットにレイヤーが追加されて、番号が付けられます。この番号は、レイヤーの名前として使用されているものであり、スクリプトで作成した ArtLayer オブジェクトのインデックス番号とは異なります。

VBScript や JavaScript のスクリプトでは、インデックスの最初のレイヤーが、レイヤーパレットの最上位にあるレイヤーになります。例えば、ドキュメントに 4 つのレイヤーがあり、Photoshop アプリケーションによって「背景レイヤー」、「レイヤー 1」、「レイヤー 2」、「レイヤー 3」という名前が付けられているとします。通常は、最後に作成された「レイヤー 3」が、レイヤーパレットの最上位に配置されています。

アプリケーションによって付けられた名前を使用してレイヤーを参照するには、次の構文を使用します。

AS layer 1 of layer set 1 of current document

注: JavaScript や VBScript のオブジェクト参照と異なり、AppleScript のオブジェクト参照名は常に同じオブジェクトを参照しているとは限りません。AppleScript の言語ガイドやテキストブックで、as alias や to a reference to file を使用したファイルの参照について調べてください。

VBS Layers("Layer 3").name

JS layers["Layer 3"].name //using the collection name and square brackets for the collection

Layer Set オブジェクトの操作

既存のレイヤーを、レイヤーセットの中に移動することができます。Layer Set オブジェクトを作成して既存の ArtLayer オブジェクトを複製し、複製したオブジェクトをレイヤーセットの中に移動する方法を、次の例に示します。

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
```

AppleScript では、目的のレイヤーセットにレイヤーを直接複製することもできます。

```
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 VBScript では、同様の方法でレイヤーを複製し配置することができます。

```
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 JavaScript では、複製メソッドの中でレイヤーを配置することができます。

```
// 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)
```

レイヤーオブジェクトのリンク

スクリプトで、レイヤーのリンクやリンク解除を行うことができます。レイヤーをリンクすると、1つの文で複数のレイヤーを移動したり変形したりできます。

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
```

『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で、link コマンドを調べてください。

VBS

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

『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザで、ArtLayer オブジェクトの Link メソッドを調べてください。また、ArtLayers オブジェクトの Add メソッドを調べてください。

JS

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

『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで、ArtLayer オブジェクトの link() メソッドを調べてください。また、ArtLayers オブジェクトの add() メソッドを調べてください。

レイヤーへのスタイルの適用

注: この処理は、Photoshop のスタイルパレットからスタイルをレイヤーにドラッグする操作に相当します。

スクリプトで、ArtLayer オブジェクトにスタイルを適用することができます。スクリプトでスタイルを適用するには、apply layer style / ApplyStyle / applyStyle() コマンドを使用します。この引数には、使用するスタイル名を二重引用符で囲んで渡します。

注: レイヤースタイルの名前は、大文字と小文字が区別されます。

スタイル、スタイルのリスト、スタイルパレットについて詳しくは、Photoshop ヘルプを参照してください。

次の例では、「L1」という名前のレイヤーに、パズルのレイヤースタイルが適用されます。

AS

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

『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で、apply layer style コマンドを調べてください。

VBS

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

『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザで、ArtLayer オブジェクトの ApplyStyle メソッドを調べてください。

JS

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

『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで、ArtLayer オブジェクトの applyStyle() メソッドを調べてください。

Text Item オブジェクトの使用

既存のレイヤーが空の場合は、その ArtLayer オブジェクトをテキストレイヤー (Text Item オブジェクト) に変更できます。逆に、Text Item オブジェクトを ArtLayer オブジェクトに変換することもできますが、この場合、レイヤーオブジェクト内のテキストはラスタライズされます。

Text Item オブジェクトは ArtLayer オブジェクトのプロパティです。新しいテキストレイヤーを作成するには、新規の ArtLayer オブジェクトを作成し、そのレイヤーの kind / Kind / kind プロパティを text layer (2 (psTextLayer) / LayerKind.TEXT) に設定する必要があります。

テキストレイヤーでテキストの作成や操作を行うには、ArtLayer オブジェクトの text object / TextItem / textItem プロパティに保持されている text-object (TextItem / TextItem) オブジェクトを使用します。

Text Item オブジェクトの作成

次の例では、ArtLayer オブジェクトを作成し、作成したレイヤーを kind プロパティを使用してテキストレイヤーに変換します。

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

ArtLayer オブジェクトと TextItem オブジェクトの関係について詳しくは、[11 ページの「Photoshop オブジェクトモデル」](#)を参照してください。

また、次の箇所を調べてください。

- 『Adobe Photoshop CS6 Visual Basic Scripting Reference』や『Adobe Photoshop CS6 JavaScript Scripting Reference』か、Visual Basic のオブジェクトブラウザや ExtendScript のオブジェクトモデルビューアで、ArtLayer オブジェクトの Kind / kind プロパティや TextItem / textItem プロパティを調べてください。
- 『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で、art layer クラスの kind プロパティや text object プロパティを調べてください。

レイヤーのタイプの特定

次の例では、if 文を使用して、既存のレイヤーがテキストレイヤーかどうかを調べます。

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)
{...}

Text Item オブジェクトへのテキストの追加と操作

次の例では、テキストレイヤーにテキストを追加し、テキストを右揃えにします。

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
```

注: text-object (TextItem / TextItem) オブジェクトには、kind (Kind / kind) プロパティがあります。このプロパティは、point text (psPointText / TextType.POINTTEXT) か paragraph text (psParagraphText / TextType.PARAGRAPHTEXT) のいずれかに設定できます。text-object を新しく作成した場合、その kind プロパティは自動的に point text に設定されます。

text-object の height、width および leading プロパティは、テキストアイテムの kind プロパティが paragraph text に設定されている場合にのみ有効です。

スクリプティングリファレンスの次の箇所で、これらのオブジェクト、プロパティ、コマンドを調べてください。

- ▶ 『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で、text-object のプロパティやメソッドを調べます。
- ▶ 『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザで、ArtLayer オブジェクトの TextItem プロパティを調べます。テキストレイヤーで利用できるプロパティやメソッドを確認するには、TextItem オブジェクトを調べます。

『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで、ArtLayer オブジェクトの textItem プロパティを調べます。テキストレイヤーで利用できるプロパティやメソッドを確認するには、TextItem オブジェクトを調べます。

Selection オブジェクトの操作

Selection オブジェクトを使用すると、ドキュメント内やレイヤー内の特定の選択範囲にスクリプトの処理対象を限定することができます。例えば、選択範囲に効果を適用したり、現在の選択範囲をクリップボードにコピーしたりすることができます。

Selection オブジェクトは Document オブジェクトのプロパティです。詳しくは、次の箇所を調べてください。

- 『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で、select コマンドを調べます。また、Document オブジェクトの selection プロパティと、selection-object を調べます。
- 『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザで、Document オブジェクトの Selection プロパティを調べます。また、Selection オブジェクトの Select メソッドを調べます。
- 『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで、Document オブジェクトの selection プロパティを調べます。また、Selection オブジェクトの select メソッドを調べます。

注: Selection オブジェクトを新たに作成することはできません。Document オブジェクトの selection (Selection / selection) プロパティには、ドキュメントの以前の選択オブジェクトが保持されています。選択範囲を指定するには、select (Select / select) コマンドを使用します。

選択範囲の作成と定義

選択範囲を作成するには、Selection オブジェクトの select / Select / select() コマンドを使用します。

選択範囲を定義するには、選択範囲のコーナーを表す画面上の座標を指定します。ドキュメントは 2 次元のオブジェクトなので、次のように x 軸と y 軸を使用して座標を指定します。

- x 軸を使用して、キャンバス上の水平方向の位置を指定します。
- y 軸を使用して、キャンバス上の垂直方向の位置を指定します。

Photoshop の原点である x 軸 = 0、y 軸 = 0 は、画面の左上隅になります。この対角は右下隅で、キャンバスの最も端の点になります。例えば、1000 x 1000 ピクセルのキャンバスでは、右下隅の座標は x 軸 = 1000、y 軸 = 1000 になります。

選択範囲を指定するには、その座標を配列に格納して、この配列を select / Select / select() コマンドの引数またはパラメータ値として指定します。

次の例では、定規単位がピクセルに設定されていることを前提として、選択範囲を作成します。

1. サイズが 500 x 500 ピクセルの新規ドキュメント作成し、それを保持する変数を作成します。
2. 選択範囲 (Selection オブジェクト) を表す座標を保持するための変数を作成します。
3. 選択範囲の変数の値として、配列を追加します。
4. Document オブジェクトの selection プロパティと、Selection オブジェクトの select コマンドを使用して、範囲を選択します。選択範囲の座標は、選択範囲の変数の値になります。

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)

```

選択範囲の境界線の作成

次の例では、Selection オブジェクトの `stroke` (`Stroke` / `stroke()`) コマンドを使用して、現在の選択範囲の周りに境界線を作成し、境界線のカラーと幅を設定します。

注: `transparency` パラメータは、背景レイヤーでは使用できません。

```

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)

```

選択範囲の反転

Selection オブジェクトの `invert` (`Invert` / `invert()`) コマンドを使用すると、現在の選択範囲を保護して、ドキュメント、レイヤー、チャンネルの残りの範囲を操作できます。

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

選択範囲の拡張／縮小と境界のぼかし

境界の拡張、縮小、ぼかしのコマンドを使用して、選択範囲のサイズを変更できます。

引数の値は、Photoshop の環境設定で設定されている定規単位で指定します。この単位はスクリプトで変更できます。次の例では、5 ピクセルだけ境界を拡大し、縮小し、ぼかします（定規単位がピクセルに設定されている場合）。定規単位を変更する方法の例は、[32 ページの「アプリケーションの環境設定」](#)を参照してください。

```
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 )
```

選択範囲の塗りつぶし

選択範囲を、特定のカラーやヒストリーで塗りつぶすことができます。

カラーで塗りつぶすには、次のようにします。

```
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)
```

現在の選択範囲を、ヒストリーの 10 番目のアイテムで塗りつぶすには、次のようにします。

注:History State オブジェクトについて詳しくは、[47 ページの「HistoryState オブジェクトの使用」](#)を参照してください。

```
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])
```

選択範囲の読み込みと保存

Selection オブジェクトを Channel オブジェクトに保存したり、Channel オブジェクトから読み込んだりすることができます。チャンネルに選択領域を保存するには、チャンネルの kind (Kind / kind) プロパティが、選択領域を保持するためのタイプである selected area channel (psSelectedAreaAlphaChannel / ChannelType.SELECTEDAREA) に設定されている必要があります。次の例では、Selection オブジェクトの

store (Store / store()) コマンドを使用して、現在の選択範囲を My Channel というチャンネルに保存します。保存した選択範囲が既に保存されている選択範囲に追加されて、その範囲が拡張されます。

```
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)
```

Channel オブジェクトに保存されている選択範囲を復元するには、load (Load / load) メソッドを使用します。

```
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)
```

選択範囲をコピー、カット、ペーストする方法の例は、[54 ページの「クリップボード操作の理解」](#)を参照してください。

Channel オブジェクトの操作

Channel オブジェクトを使用すると、Photoshop のチャンネルで使用できる様々な機能にアクセスできます。チャンネルの作成、削除、複製を行ったり、チャンネルのヒストグラムを取得したりすることができます。また、チャンネルのタイプを変更することもできます。スクリプトで Channel オブジェクトを作成する方法について詳しくは、[23 ページの「スクリプトでの新規オブジェクトの作成」](#)を参照してください。

Channel オブジェクトのタイプは、kind プロパティを使用して、設定または取得する（確認する）ことができます。選択範囲チャンネルの作成方法を示すサンプルスクリプトについては、[45 ページの「選択範囲の読み込みと保存」](#)を参照してください。

チャンネルのタイプ変更

コンポーネントチャンネル以外のチャンネルは、kind (タイプ) を変更できます。次の例では、マスク範囲チャンネルを選択範囲チャンネルに変更します。

注: コンポーネントチャンネルは、ドキュメントのモードに関連しています。チャンネル、チャンネルのタイプ、ドキュメントのモードについて詳しくは、Photoshop ヘルプを参照してください。

```
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
```

DocumentInfo オブジェクトの使用

Photoshop では、ドキュメントに関連付けられている情報を、**ファイル／ファイル情報**で表示できます。

この機能のスクリプトで使用するには、info-object (DocumentInfo／DocumentInfo) オブジェクトを使用します。このオブジェクトは、Document オブジェクトの info (Info／info) プロパティに保持されています。次の例では、DocumentInfo オブジェクトを使用して、ドキュメントの著作権ステータスと所有者の URL を設定します。

```
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"
```

ドキュメントに関連付けることができる情報（プロパティ）の種類について詳しくは、次の箇所を調べてください。

- 『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で、info-object クラスのプロパティを調べます。
- 『Adobe Photoshop CS6 Visual Basic Scripting Reference』や『Adobe Photoshop CS6 JavaScript Scripting Reference』か、Visual Basic のオブジェクトブラウザや ExtendScript のオブジェクトモデルビューアで、DocumentInfo オブジェクトのプロパティを調べます。

HistoryState オブジェクトの使用

Photoshop では、ドキュメントに影響を与えた操作の履歴（ヒストリー）が保持されます。Photoshop アプリケーションで画像に変更を加えるたびに、**ヒストリー**が作成されます。ドキュメントのヒストリーには、ヒストリーパレット（**ウィンドウ／ヒストリー**）からアクセスできます。ヒストリーについて詳しくは、Photoshop ヘルプを参照してください。

スクリプトで Document オブジェクトのヒストリーにアクセスするには、HistoryStates オブジェクトを使用します。これは Document オブジェクトのプロパティです。HistoryStates オブジェクトを使用して、ドキュメントを以前の状態に戻したり、Selection オブジェクトをヒストリーで塗りつぶしたりできます。

次の例では、docRef という変数に保持されているドキュメントを、最初に開かれたとき（または新規作成されたとき）の状態とプロパティに戻します。このようにヒストリーを使用することで、ドキュメントの状態を元に戻すことができます。

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]`

注: 以前のヒストリーに戻しても、それ以降のヒストリーが削除されることはありません。HistoryStates のコレクションから以降のヒストリーを削除するには、次のように Purge コマンドを使用します。

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

次の例では、現在のヒストリーを保存し、フィルタを適用して、保存したヒストリーに戻します。

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`

Notifier オブジェクトの使用

Notifier オブジェクトを使用すれば、スクリプトとイベントを関連付けることができます。例えば、アプリケーションを開いたときに Photoshop によって自動的に新規ドキュメントが作成されるようにするには、Document オブジェクトを作成するスクリプトを Open Application イベントに関連付けます。

注: これは、Photoshop アプリケーションのスクリプトイベントマネージャ（ファイル／スクリプト／スクリプトイベントマネージャ）で、「アプリケーションを起動」を選択してスクリプトを指定するのと同じです。スクリプトイベントマネージャの使用方法について詳しくは、Photoshop ヘルプを参照してください。

make (Add/add) コマンドでは、通知を行うイベントを識別するためのイベント ID を指定する必要があります。イベント ID の一覧については、『Adobe Photoshop CS6 JavaScript Scripting Reference』、『Adobe Photoshop CS6 Visual Basic Scripting Reference』、『Adobe Photoshop CS6 AppleScript Scripting Reference』の付録を参照してください。複数のタイプのオブジェクトで使用されているイベントの場合は、オブジェクトを識別するためのクラス ID を make (Add/add) コマンドの追加の引数に指定する必要があります。例えば、「New」コマンドは、Document、Art Layer、Channel の各オブジェクトで使用されています。

注: ScriptListener を使用すると、記録可能なイベントのイベント ID やクラス ID を確認できます。[81 ページの「ScriptListener によるイベント ID やクラス ID の確認」](#)を参照してください。

「Open Document」イベントの通知を設定する方法を、次の例に示します。このスクリプトでは、イベント通知を有効にした後で、イベントによって Welcome.jsx ファイルが実行されるように設定しています。このスクリプトを実行した後は、ドキュメントを開くたびに通知が行われて .jsx ファイルが実行されるようになります。この .jsx ファイルは警告ボックスを表示します。

注: 通常、スクリプトの内部で発生したイベントに対する通知は行われません。これらのイベントは「AdobeScriptAutomation Scripts」イベントに含まれます。


```

AS
tell application "Adobe Photoshop CS6"
    try
        delete notifiers
    end try
    make new notifier with properties {event:"Opn ", ~
        event file:alias "OS X 10.5.8 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)

```

PathItem オブジェクトの使用

PathItem オブジェクトを作成するには、ドキュメントの PathItems エlement またはコレクションに PathItem を追加する必要があります。そのためには、まず PathPointInfo オブジェクトの配列を作成する必要があります。この配列で、作成するパスのアンカーポイントや各コーナーの座標を指定します。次に、SubPathInfo オブジェクトの配列を作成して、PathPoint 配列を格納します。ポイントとサブパスが準備できたら、新しい PathItem を追加することができます。

次のスクリプトでは、直線を表す PathItem オブジェクトを作成します。

```

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 CS6"
    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)
```

カラーオブジェクトの操作

Photoshop ユーザーインターフェイスで利用できるカラーと同じ種類のカラーが、スクリプトでも使用できます。それぞれのカラーモデルには固有のプロパティが用意されています。例えば、RGB color (RGBColor / RGBColor) クラスには、レッド、ブルー、グリーンの3つのプロパティが含まれています。このクラスのカラーを設定するには、3つの各プロパティに値を指定します。

VBScript と JavaScript では、SolidColor クラスに各カラーモデルのプロパティが含まれています。このオブジェクトを使用するには、まず SolidColor オブジェクトのインスタンスを作成し、次にそのカラーモデルのプロパティを適切に設定します。一旦 SolidColor オブジェクトにカラーモデルを割り当てると、その SolidColor オブジェクトに別のカラーモデルを割り当て直すことはできません。

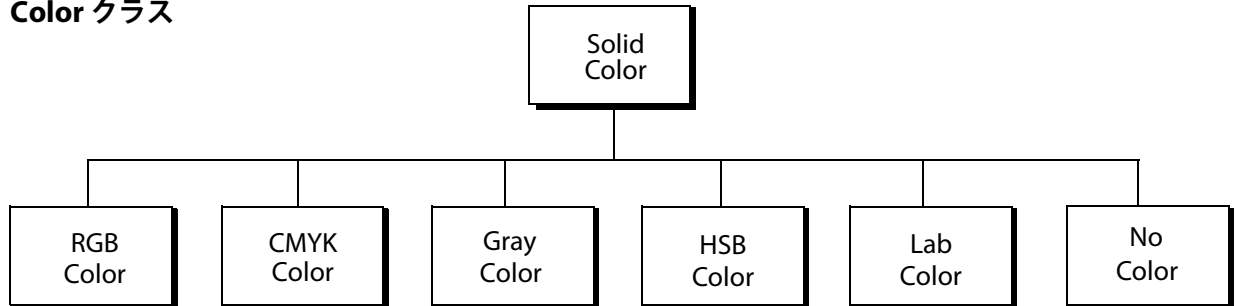
次の例では、CMYK color クラスを使用してカラーを設定します。

AS	<pre>set foreground color to {class:CMYK color, cyan:20.0, magenta:90.0, yellow:50.0, black:50.0}</pre>
VBS	<pre>'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</pre>
JS	<pre>//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</pre>

Solid Color クラス

次の図に、Photoshop で使用できる Solid Color クラスを示します。

Color クラス



RGB カラーでの 16 進数値の使用

RGB カラーは **16 進数**の値で表すことができます。この 16 進数の値は、2 つの桁を 1 つのペアとして、3 つのペアから構成されます。それぞれのペアは、左から順に、レッド、グリーン、ブルーを表します。

AppleScript の場合、16 進数の値は RGB hex color クラスの hex value 文字列プロパティで表されます。また、後述のように convert color コマンドを使用して 16 進数の値を取得することもできます。

VBScript と JavaScript の場合は、RGBColor オブジェクトに HexValue / hexValue という文字列プロパティがあります。

カラーの取得と変換

次の例では、RGB カラーを同等の CMYK カラーに変換します。

AS

次のスクリプトでは、RGB カラーモデルが使用されていることを前提としています。前景色を取得し、color クラスの convert コマンドを使用して、カラーを同等の CMYK カラーに変換します。

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

『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で、次の箇所を調べてください。

- 「Objects」の節で、application クラスの foreground color プロパティを調べます。
- 「Commands」の節で、convert を調べます。

VBS

次のスクリプトでは、If Then 文と SolidColor オブジェクトの model プロパティを使用して、使用されているカラーモデルを調べています。If Then 文で返される SolidColor オブジェクトが RGB オブジェクトである場合は、SolidColor オブジェクトの cmyk プロパティにアクセスして、同等の CMYK カラーを取得しています。

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

『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザで、次の箇所を調べてください。

- SolidColor オブジェクトの model プロパティと cmyk プロパティを調べます。

JS

この例では、Application オブジェクトの foregroundColor プロパティを使用して、変換元のカラーを取得しています。foregroundColor が参照している SolidColor オブジェクトの cmyk プロパティによって、RGB カラーと同等の CMYK カラーを取得しています。

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

『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで、次の箇所を調べてください。

- SolidColor オブジェクトの cmyk プロパティを調べます。
- Application オブジェクトの foregroundColor プロパティを調べます。

カラーの比較

equal colors (IsEqual/isEqual) コマンドを使用すると、カラーを比較することができます。次の文は、前景色と背景色が視覚的に同じ場合に true を返します。

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

Web セーフカラーの取得

特定のカラーを Web セーフカラーに変換するには、AppleScript では web safe color コマンドを使用し、VBScript と JavaScript では SolidColor オブジェクトの NearestWebColor/nearestWebColor プロパティを使用します。

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
```

フィルタの操作

AppleScript でフィルタを適用するには、filter コマンドに filter options クラスのオプションを指定します。VBScript と JavaScript では、個別のフィルタメソッドを使用します。例えば、ぼかし（ガウス）フィルタを適用するには、ApplyGaussianBlur/applyGaussianBlur() メソッドを使用します。すべてのフィルタメソッドは、ArtLayer オブジェクトに用意されています。

注:それぞれのフィルタタイプの効果については、Photoshop ヘルプを参照してください。

次の例では、アクティブなレイヤーにぼかし（ガウス）フィルタを適用します。

AS filter コマンドを使用して、レイヤーとフィルタの名前を両方指定し、オプションを指定します。

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

注:『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で、filter コマンドや filter options クラスを調べてください。

VBS appRef.docRef.ActiveLayer.ApplyGaussianBlur 5

注:『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザで、ArtLayer オブジェクトの ApplyGaussianBlur メソッドや、名前が「**Apply**」で始まるその他のメソッドを調べてください。

JS docRef.activeLayer.applyGaussianBlur(5)

注:『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで、artLayer オブジェクトの applyGaussianBlur() メソッドや、名前が「**apply**」で始まるその他のメソッドを調べてください。

その他のフィルタ

レイヤーに適用したいフィルタタイプがスクリプティングインターフェイスに用意されていない場合は、JavaScript からアクションマネージャを使用して、フィルタを実行することができます。AppleScript や VBScript を使用している場合でも、スクリプトから JavaScript を実行することができます。アクションマネージャの使用について詳しくは、[73 ページの「アクションマネージャ」](#)を参照してください。また、[11 ページの「AS や VBS からの JavaScript の実行」](#)も参照してください。

クリップボード操作の理解

Photoshop のクリップボードコマンドは、ArtLayer、Selection、Document オブジェクトに作用します。このコマンドを使用すれば、1 つのドキュメント内でオブジェクトを操作したり、ドキュメント間で情報を移動したりすることができます。

art layer (ArtLayer / ArtLayer) オブジェクトと selection (Selection / Selection) オブジェクトには、次のクリップボードコマンドがあります。

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

document / Document / Document オブジェクトには、次のクリップボードコマンドがあります。

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

注: copy、copy merged、paste、paste into、cut について詳しくは、Photoshop ヘルプを参照してください。

コピーおよびペーストコマンドの使用

次の例では、背景レイヤーの内容をクリップボードにコピーします。その後、新規ドキュメントを作成し、クリップボードの内容を新規ドキュメントにペーストします。このスクリプトでは、Photoshop でドキュメントが既に開かれていて、ドキュメントに背景レイヤーがあることを前提としています。

注: クリップボードのペースト先となるドキュメントをスクリプトで新規作成する場合は、コピー元のドキュメントと定規単位が同じになるようにしてください。詳しくは、[32 ページの「アプリケーションの環境設定」](#)を参照してください。

AS **注:** Macintosh でこれらのコマンドを実行する際には、Photoshop が最前面のアプリケーションになっている必要があります。クリップボードコマンドを実行する前に、activate コマンドを使用して、アプリケーションをアクティブにしてください。

```
tell application "Adobe Photoshop CS6"
    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()
```

マージコピーコマンド

マージコピーコマンドを使用すると、選択範囲に含まれているすべての表示レイヤーをコピーすることができます。AppleScript では、copy merged コマンドを使用します。VBScript や JavaScript では、Copy / copy コマンドを使用し、オプションの merge パラメータに True / true という値を設定します。

AS **注:** Macintosh でこれらのコマンドを実行する際には、Photoshop が最前面のアプリケーションになっている必要があります。クリップボードコマンドを実行する前に、activate コマンドを使用して、アプリケーションをアクティブにしてください。

```
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
```

『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザで、ArtLayer オブジェクトや Selection オブジェクトの Copy メソッドを調べてください。

JS

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

『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで、ArtLayer オブジェクトや Selection オブジェクトの copy() メソッドを調べてください。

単位の操作

Photoshop のドキュメントでは、2 つの定規が使用できます。settings-object (Preferences / Preferences) オブジェクトのプロパティを使用すれば、これらの定規の単位をスクリプトで設定することができます。使用できる定規は次のとおりです。

- ▶ グラフィック定規は、ほとんどのグラフィックレイアウト作業で使用される定規で、高さ、幅、位置が示されます。

グラフィック定規の単位の種類は、ruler units (RulerUnits / rulerUnits) プロパティを使用して設定します。

- ▶ 文字定規は、文字ツールを使用するときにアクティブになります。

文字定規の単位の種類は、type units (TypeUnits / typeUnits) プロパティを使用して設定します。

注: これらの設定は、Photoshop の環境設定ダイアログボックスに表示される設定に対応します。このダイアログボックスは、**編集 / 環境設定 / 単位・定規** (Windows) または **Photoshop / 環境設定 / 単位・定規** (Macintosh) で表示されます。

単位値

どの言語でも、単位が付いていない数値 (単位値) を使用することができます。単位が付いていない値は、対応する定規で現在設定されている単位を使用して処理されます。

例えば、定規単位がインチに設定されている場合、次の VBScript 文を実行すると、ドキュメントのサイズが 3 x 3 インチになります。

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

定規単位がピクセルに設定されている場合は、3 x 3 ピクセルになります。意図したとおりの結果を得るためには、スクリプトで定規単位を確認して、適切な単位を設定する必要があります。また、定規の設定を変更した場合は、処理を終えた後で、元の値に戻すようにしてください。単位値の設定については、[59 ページの「スクリプトによる定規単位や文字単位の設定」](#)を参照してください。

使用可能な単位の種類について詳しくは、Photoshop ヘルプを参照してください。

特別な単位

Photoshop で使用されている単位は、直線距離を表す長さの単位ですが、それとは別にピクセルやパーセントの単位もサポートされています。この 2 つの単位は、厳密には長さの単位ではありませんが、Photoshop の様々な操作や値で頻繁に活用されるので、これらの単位も用意されています。

AppleScript での単位に関する考慮事項

AppleScript は、単位値の処理に関して柔軟性を持っており、単位値を使用するところで、明示的に単位の種類を指定することができます。単位付きの値を指定した場合には、現在の定規の設定がオーバーライドされます。

例えば、幅 4 インチ、高さ 5 インチのドキュメントを作成するには、次のようにします。

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


Photoshop のプロパティのうち、単位を使用するプロパティの値は、現在の定規の単位で表されます。前の例で作成したドキュメントの高さを、次のようにして取得すると、

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

5.0 という値が返されます。これは、現在の定規の設定に基づいており、5 インチを表します。

AppleScript では、特定の単位でプロパティ値を返すように要求することができます。

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

このようにすると、360 という値 (5 インチ x 72 ポイント/インチ) が返されます。

points と picas は、PostScript でのポイントを表します (72 ポイント/インチ)。traditional points と traditional picas は、従来の組版で使用されていた値 (72.27 ポイント/インチ) に基づいています。

単位値を、ある単位から別の単位に換算することができます。例えば、次のスクリプトでは、値をポイントからインチに換算します。

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

このスクリプトを実行すると、inchValue という変数に inches 1 が格納されます (72 ポイントを換算すると 1 インチになります)。AppleScript 言語には、このような換算機能が用意されています。

注: cm units や mm units という単位を使用して、同様の操作で cm や mm に変換することはできません。そのような表記は、AppleScript ではサポートされていません。

計算における単位値の使用

AppleScript で単位値を計算に使用する場合は、まず単位値を数値に変換する必要があります (単位値を計算に直接使用することはできません)。インチ値を使用して乗算を行うには、次のようにします。

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

注: AppleScript では、他の単位と同じようにして、ピクセル値やパーセント値を取得および設定することができます。しかし、ピクセル値やパーセント値を他の長さの単位に換算することはできません。次のスクリプトを実行しようとすると、エラーが発生します。

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

注: Photoshop はピクセル指向のアプリケーションであるため、設定時に渡した値と異なる値が返される場合があります。例えば、定規単位が mm に設定されている場合、30 x 30 のドキュメントを作成すると、返される高さおよび幅の値は 29.99 になります (解像度が 72ppi の場合)。スクリプティングインターフェイスでは、設定は ppi で測定されているものと想定しています。

単位値の使用

プロパティに単位値が使用できるクラスやオブジェクトを、次の表に示します。これらのプロパティの単位値は、表内でマークが付けられているものを除き、グラフィック定規の設定に従います。

この表を使用する場合は、次のいずれかを行ってください。

- 『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で、クラスのプロパティを調べてください。
- 『Adobe Photoshop CS6 Visual Basic Scripting Reference』や『Adobe Photoshop CS6 JavaScript Scripting Reference』か、Visual Basic のオブジェクトブラウザや ExtendScript のオブジェクトモデルビューアで、オブジェクトのプロパティを調べてください。

クラス／オブジェクト	AppleScript の プロパティ	VBScript の プロパティ	JavaScript の プロパティ
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

* 文字定規の設定に従う単位値

パラメータ（引数）に単位値が使用できるコマンドを、次の表に示します。これらのパラメータのいくつかは、必須のパラメータです。VBScript や JavaScript のメソッドの前には、そのメソッドを持つオブジェクトが記載されています。

この表を使用する場合は、次のいずれかを行ってください。

- AppleScript のコマンドについては、『Adobe Photoshop CS6 AppleScript Scripting Reference』のコマンドの章か、Photoshop AppleScript 用語説明で、コマンドを調べてください。
- VBScript のメソッドについては、『Adobe Photoshop CS6 Visual Basic Scripting Reference』のインターフェイスの章のオブジェクトのメソッドテーブルか、Visual Basic のオブジェクトブラウザで、メソッドを調べてください。
- JavaScript のメソッドについては、『Adobe Photoshop CS6 JavaScript Scripting Reference』のオブジェクトリファレンスの章のオブジェクトのメソッドテーブルか、ExtendScript のオブジェクトモデルビューアで、メソッドを調べてください。

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)

スクリプトによる定規単位や文字単位の設定

単位値がサポートされているプロパティやパラメータで数値がどのように解釈されるかは、Photoshop の 2 つの定規の単位設定によって決まります。スクリプトの最初の部分で元の設定を保存し、必要に応じて定規単位を設定して、スクリプトの完了時に復元してください。

AS AppleScript では、ruler units と type units は settings-object のプロパティであり、次のように Application オブジェクトの settings プロパティからアクセスできます。

```
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 VBScript では、RulerUnits と TypeUnits は Preferences オブジェクトのプロパティであり、次のように Application オブジェクトの Preferences プロパティからアクセスできます。

```
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 JavaScript では、rulerUnits と typeUnits は Preferences オブジェクトのプロパティであり、次のように Application オブジェクトの preferences プロパティからアクセスできます。

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

注: スクリプトの最後で、忘れずに単位設定を元の値に戻してください。定規単位を変更する方法の例は、[61 ページの「ドキュメントの環境設定の操作」](#)を参照してください。

ワークフローを自動化するためのサンプル JavaScript

Photoshop には、スクリプトの活用例として、ワークフローを自動化するための次のようなサンプル JavaScript が用意されています。これらのスクリプトは、アプリケーションディレクトリの Presets フォルダ内の Scripts フォルダにあります。Presets フォルダ内の Scripts フォルダについて詳しくは、[19 ページの「JavaScript の作成と実行」](#)を参照してください。

スクリプト名	説明
Layer Comps to Files.jsx	レイヤーカンパをファイルとして保存します。
Layer Comps to PDF.jsx	レイヤーカンパを PDF スライドショーとして保存します。
Layer Comps to WPG.jsx	レイヤーカンパを Web フォトギャラリーとして保存します。
Export Layers to Files.jsx	ドキュメント内の各レイヤーを別々のファイルに書き出します。
Script Events Manager.jsx	Notifier オブジェクトを有効化または無効化します。
Image Processor.jsx	Camera Raw 形式の画像を様々なファイル形式に変換します。
Load Files into Stack.jsx	別々のファイルを単一のドキュメントの Image Stack に読み込みます。
Merge to HDR.jsx	同じシーンまたは画像を表す複数の画像を統合して、そのシーンのダイナミックレンジを反映した単一のハイダイナミックレンジ (HDR) 画像を作成します。

高度なスクリプティング

ここでは、この章の各節で学んだテクニックを利用して、次の操作を実行するスクリプトを作成します。

- ドキュメントの環境設定を変更する。
- テキストアイテムにカラーを適用する。この節では、次の方法についても学びます。
 - 既存のドキュメントへの参照を作成する。
 - レイヤーオブジェクトを作成し、レイヤーをテキストレイヤーに変更する。
- テキストをラスターライズして、波形処理やぼかし処理を文字に適用できるようにする。この節では、次の方法についても学びます。
 - 選択オブジェクトを作成して、レイヤーの特定の領域を選択して処理する。
 - 選択したテキストに波形フィルタとぼかし（移動）フィルタを適用する。

注: 各節の演習が終了するたびに、演習で作成したスクリプトを保存してください。各節で作成したスクリプトは、次の節でも使用します。

ドキュメントの環境設定の操作

この節のサンプルスクリプトでは、まず Photoshop の Application オブジェクトをアクティブにし、スクリプトの終了時に設定を元に戻せるようにデフォルト設定を変数に保存します。デフォルト設定とは、Photoshop をインストールした後に環境設定ダイアログボックスで指定した設定のことです。

注: 環境設定を表示または設定するには、**編集／環境設定／単位・定規** (Windows) または **Photoshop／環境設定／単位・定規** (Macintosh) を選択します。

その後、各環境設定を次の値に設定します。

環境設定	設定値	説明
定規	inches	グラフィックの単位としてインチを使用します。
単位	pixels	テキスト (文字) の単位としてピクセルを使用します。
ダイアログボックスのモード	never	ダイアログボックスを抑制して、スクリプトの実行過程でユーザ入力 (「OK」のクリックなど) が求められるのを防ぎます。 注: Photoshop のユーザインターフェイスには、ダイアログボックスのモードを設定するためのオプションはありません。

次に、ドキュメントのサイズ (インチ単位) を格納する変数と、ドキュメントの解像度 (ピクセル単位) を格納する変数を宣言し、表示解像度を宣言して、「Hello, World!」というテキストを文字列変数に割り当てます。

その後、Document オブジェクトが既に作成されているかどうかを if 文で確認し、作成されていなければ新規の Document オブジェクトを作成します。

最後に、元の環境設定に戻します。

AS

ドキュメントの環境設定を操作するには：

1. 次のスクリプトを作成し、実行します。詳しくは、[18 ページの「AppleScript の作成と実行」](#)を参照してください。

```
tell application "Adobe Photoshop CS6"
    --make Photoshop CS6 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. Photoshop で、**Photoshop／環境設定／単位・定規**を選択し、環境設定が元の設定に戻っていることを確認します。
3. Photoshop でドキュメントを確認し、保存せずにドキュメントを閉じます。
4. スクリプトを HelloWorldDoc という名前で保存します。

VBS

ドキュメントの環境設定を操作するには：

1. 次のスクリプトを作成します。詳しくは、[19 ページの「VBScript の作成と実行」](#)を参照してください。

```

'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. Photoshop で、**編集／環境設定／単位・定規**を選択し、環境設定が元の設定に戻っていることを確認します。
3. Photoshop でドキュメントを確認し、保存せずにドキュメントを閉じます。
4. スクリプトを HelloWorldDoc という名前で保存します。

JS

ドキュメントの環境設定を操作するには：

1. 次のスクリプトを作成します。

注：JavaScript の作成方法について詳しくは、[19 ページの「JavaScript の作成と実行」](#)を参照してください。

```
//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. Presets フォルダ内の Scripts フォルダに、HelloWorldDoc.jsx という名前でスクリプトを保存します。
3. Photoshop を開き、**ファイル／スクリプト／HelloWorldDoc**を選択してスクリプトを実行します。
4. **編集／環境設定／単位・定規**を選択し、環境設定が元の設定に戻っていることを確認します。
5. Photoshop でドキュメントを確認し、保存せずにドキュメントを閉じます。
6. スクリプトを保存します。

テキストアイテムへのカラーの適用

ここでは、HelloWorldDoc スクリプトを使用してレイヤーを追加し、レイヤーをテキストオブジェクトに変更します。テキストオブジェクトでは、**Hello, World!** が赤色で表示されるようにします。

最初に、次のことを行ってください。

- Photoshop が閉じていることを確認します。
- スクリプトエディタアプリケーションで、HelloWorldDoc スクリプトを開きます。

AS

テキストアイテムを作成し、テキストの内容を指定するには：

1. HelloWorldDoc スクリプトファイルの最後にある、元の環境設定を復元する文の直前に、次のコードを追加します。

```
--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. 完成したスクリプトを実行します。Photoshop によってコマンドがすべて処理されるまで待ちます。
3. Photoshop でドキュメントを確認し、保存せずにドキュメントを閉じます。

注：『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で次のクラスを調べて、このスクリプトでの使用方法を理解しているかどうかを確認してください。

- RGB color クラス
- art layer クラス

VBS

テキストアイテムを作成し、テキストの内容を指定するには：

1. HelloWorldDoc スクリプトファイルの最後にある、元の環境設定を復元する文の直前に、次のコードを追加します。

```
'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. 完成したスクリプトを実行します。Photoshop によってコマンドがすべて処理されるまで待ちます。
3. Photoshop でドキュメントを確認し、保存せずにドキュメントを閉じます。

注：『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザで次のクラスを調べて、このスクリプトでの使用方法を理解しているかどうかを確認してください。

- SolidColor
- ArtLayer

JS テキストアイテムを作成し、テキストの内容を指定するには:

1. HelloWorldDoc スクリプトファイルの最後にある、元の環境設定を復元する文の直前に、次のコードを追加します。

```
//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. スクリプトを保存して Photoshop を開き、スクリプトメニューからスクリプトを選択します (**ファイル／スクリプト／HelloWorldDoc** を選択)。Photoshop によってコマンドがすべて処理されるまで待ちます。
3. Photoshop でドキュメントを確認し、ドキュメントを保存せずに Photoshop を閉じます。

注:『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで次のクラスを調べて、このスクリプトでの使用方法を理解しているかどうかを確認してください。

- SolidColor
- ArtLayer。kind プロパティの LayerKind.TEXT 値は、LayerKind の定数を使用しています。Photoshop 用の JavaScript では、定数は常に大文字で表されます。

波形フィルタの適用

ここでは、ドキュメント内の **Hello** という文字に波形フィルタを適用します。次の操作を行います。

- ドキュメントの幅と高さをピクセル単位で設定し、テキストレイヤー内のテキストオブジェクトをラスターライズします。

注:ベクトルグラフィックであるテキストに波形フィルタは適用できないので、最初に画像をビットマップに変換する必要があります。ラスターライズにより、数学的に定義されているベクトルアートワークがピクセルに変換されます。ラスターライズについて詳しくは、Photoshop ヘルプを参照してください。

- 波形フィルタを適用するレイヤーの領域を選択します。

注:領域を選択するスクリプトコードについて理解するには、[67 ページの「選択オブジェクトの領域定義」](#)を参照してください。

- 選択範囲に波形フィルタを適用します。

注: 波形は、正弦曲線の一部です。

選択オブジェクトの領域定義

選択オブジェクトの領域を定義するには、座標（ピクセルで指定されたドキュメント内の点）の配列を作成します。配列で表されるのは、長方形の領域の外側の角を定義する座標です。ドキュメントの左上隅を起点とし、ここではドキュメントの半分の位置までを終点とします。

注: 選択範囲の領域を定義する点は、いくつでも使用することができます。座標の数によって、選択範囲の形が決定します。最後に定義する座標は最初の座標と同じにし、選択領域が閉じられた選択パスになるようにする必要があります。

注: 選択オブジェクトやその他の Photoshop のオブジェクトについて詳しくは、[11 ページの「Photoshop オブジェクトモデル」](#)を参照してください。

配列の値は次の順に指定します。

- 選択範囲の左上隅: 0, 0
 - 0 は、ドキュメント内の一番左の列を表します。
 - 0 は、ドキュメント内の一番上の行を表します。
- 選択範囲の右上隅: theDocWidthInPixels / 2, 0
 - theDocWidthInPixels / 2 は、ドキュメントの中央の位置にある列を表します。この座標は、ドキュメント内の列の総数を 2 で除算した数です。

注: theDocWidthInPixels の値は、ドキュメントの水平方向のサイズを示すピクセルの総数です。列は水平方向に整列しています。

 - 0 は、ドキュメント内の一番上の行を表します。
- 右下隅: theDocWidthInPixels / 2, theDocHeightInPixels
 - theDocWidthInPixels / 2 は、ドキュメントの中央の位置を表します。
 - theDocHeightInPixels は、ドキュメント内の一番下の行を表します。この行の座標は、ドキュメント内の行数と同じです。

注: theDocHeightInPixels の値は、ドキュメントの垂直方向のサイズを示すピクセルの総数です。行は垂直方向に積み重なっています。
- 左下隅: 0, theDocHeightInPixels
 - 0 は、ドキュメント内の一番左の列を表します。
 - theDocHeightInPixels は、ドキュメント内の一番下の行を表します。
- 選択範囲の左上隅: 0, 0
 - 最初の点に戻って、選択パスを閉じます。

AS 領域を選択し、波形フィルタを適用するには：

1. HelloWorldDoc スクリプトファイルの、元の環境設定を復元する文の直前に、次のコードを追加します。

```
--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. 「実行」を選択して、スクリプトを実行します。
3. Photoshop でドキュメントを確認し、保存せずにドキュメントを閉じます。
4. スクリプトをスクリプトエディタで保存します。

注:『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で次のクラスを調べて、このスクリプトでの使用方法を理解しているかどうかを確認してください。

- wave filter クラス
- art layer クラス：rasterize コマンド、filter コマンド
- document クラス：select コマンド、combination type パラメータ

VBS

領域を選択し、波形フィルタを適用するには：

1. HelloWorldDoc スクリプトファイルの最後にある、元の環境設定を復元する文の直前に、次のコードを追加します。

```
'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. Windows のエクスプローラでファイル名をダブルクリックして、スクリプトを実行します。
3. Photoshop でドキュメントを確認し、保存せずにドキュメントを閉じます。
4. スクリプトを保存します。

注：『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザで次のクラスを調べて、このスクリプトでの使用方法を理解しているかどうかを確認してください。

- ArtLayer クラス：ApplyWave メソッド、Rasterize メソッド
- Selection クラス：Select メソッド

JS 領域を選択し、波形フィルタを適用するには：

1. HelloWorldDoc スクリプトファイルの、元の環境設定を復元する文の直前に、次のコードを追加します。

```
//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. スクリプトを保存して Photoshop を開き、スクリプトメニューからスクリプトを選択します（**ファイル／スクリプト／HelloWorldDoc** を選択）。
3. Photoshop でドキュメントを確認し、ドキュメントを保存せずに Photoshop を閉じます。

注：『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで次のクラスを調べて、このスクリプトでの使用方法を理解しているかどうかを確認してください。

➤ ArtLayer

- rasterize() メソッド。RasterizeType.TEXTCONTENTS 引数は、RasterizeType の定数を使用しています。Photoshop 用の JavaScript では、定数は常に大文字で表されます。
- applyWave() メソッド

ぼかし（移動）フィルタの適用

ここでは、さらに別のフィルタをドキュメントの残り半分に適用します。

AS

ぼかし（移動）フィルタを HelloWorldDoc に適用するには：

1. HelloWorldDoc スクリプトファイルの、元の環境設定を復元する文の直前に、次のコードを追加します。

```
--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. 「実行」を選択して、スクリプトを実行します。

注：『Adobe Photoshop CS6 AppleScript Scripting Reference』や Photoshop AppleScript 用語説明で motion blur クラスを調べて、このスクリプトでの使用方法を理解しているかどうかを確認してください。

VBS

ぼかし（移動）フィルタを HelloWorldDoc に適用するには：

1. HelloWorldDoc スクリプトファイルの、元の環境設定を復元する文の直前に、次のコードを追加します。

```
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. Windows のエクスプローラでファイルをダブルクリックして、スクリプトを実行します。

注：『Adobe Photoshop CS6 Visual Basic Scripting Reference』や Visual Basic のオブジェクトブラウザで、ArtLayer クラスの ApplyMotionBlur メソッドを調べて、このスクリプトでの使用方法を理解しているかどうかを確認してください。

JS ぼかし（移動）フィルタを HelloWorldDoc に適用するには：

1. HelloWorldDoc スクリプトファイルの、元の環境設定を復元する文の直前に、次のコードを追加します。

```
//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. スクリプトを保存して Photoshop を開き、スクリプトメニューからスクリプトを選択します（**ファイル／スクリプト／HelloWorldDoc** を選択）。

注：『Adobe Photoshop CS6 JavaScript Scripting Reference』や ExtendScript のオブジェクトモデルビューアで、ArtLayer クラスの applyMotionBlur() メソッドを調べて、このスクリプトでの使用方法を理解しているかどうかを確認してください。

4 アクションマネージャ

Photoshop のアクションを使用すれば、反復作業を自動化して時間を節約することができます。アクションの作成や実行は、アプリケーションのユーザインターフェイスにあるアクションパレットを使用して行えます。

また、**アクションマネージャ**と呼ばれるユーティリティを使用して、スクリプトでアクションを管理することもできます。アクションマネージャを使用すると、通常のスクリプティングインターフェイスからはアクセスできない Photoshop の機能（サードパーティ製のプラグインやフィルタなど）にアクセスするスクリプトを作成することができます。記録可能なタスクであれば、どのようなタスクでもアクションマネージャからアクセスすることができます。

この章では、アクションマネージャとそのスクリプティングインターフェイスオブジェクトの使用方法について説明します。

ScriptListener プラグイン

アクションマネージャを使用する前に、ScriptListener プラグインをインストールする必要があります。ScriptListener は、ユーザが UI で行ったアクションに対応するスクリプティングコードをファイルに記録します。

ヒント：ScriptListener はユーザのアクションをほぼすべて記録するので、ScriptListener をインストールするのはアクションマネージャスクリプトの作成時のみにしてください。ScriptListener をインストールしたままにしておくと、大きなファイルが作成されてハードドライブが消費されるだけでなく、Photoshop のパフォーマンスが低下する可能性があります。

Photoshop で何らかの操作を行うと、ScriptListener によって複数のファイルが作成されて、Photoshop で行われたアクションを表すコードが記録されます。

- ScriptingListenerJS.log。JavaScript コードが記録されます。
- ScriptingListenerVB.log。VBScript コードが記録されます（Windows のみ）。

これらのファイルはデスクトップに作成されます。

注：アクションマネージャには、AppleScript のインターフェイスはありません。ただし、AppleScript から JavaScript を実行することによって、AppleScript からアクションマネージャにアクセスすることが可能です。[80 ページの「AppleScript での JavaScript ベースのアクションマネージャコードの実行」](#)を参照してください。

ScriptListener のインストール

ScriptListener プラグインは、`..¥Adobe Photoshop CS6¥Scripting¥Utilities` フォルダに用意されています。

ScriptListener をインストールするには：

1. ScriptListener.8li ファイルを選択し、**編集／コピー**を選択します。
2. ファイルのコピーを次の場所に貼り付けます。
`..¥Adobe Photoshop CS6¥Plug-Ins¥Automate`
3. Photoshop を開きます。

注：Photoshop が既に開いている場合は、一旦閉じてから再度起動してください。これによって、Photoshop にプラグインが読み込まれます。

ScriptListener をアンインストールするには：

1. Photoshop を閉じます。
2. ScriptListener.8li ファイルのコピーが ..¥Adobe Photoshop CS6¥Scripting¥Utilities フォルダにあることを確認します。
3. 次の場所から ScriptListener.8li ファイルを削除します。
..¥Adobe Photoshop CS¥Plug-Ins¥Automate
4. デスクトップから ScriptingListenerJS.log および ScriptingListenerVB.log という名前のログファイルを削除します。

注：Windows では、Automate フォルダから ScriptListener を削除しても、アクションが記録され続けることがあります。ScriptingListenerJS.log ファイルが肥大化しないように、Photoshop アクションの再生が終わるたびにこのファイルを削除してください。

アクションマネージャのスク립ティングオブジェクト

ActionDescriptor、ActionList、ActionReference オブジェクトは、アクションマネージャを構成する機能の一部です。これらのオブジェクトについて詳しくは、各言語のリファレンスマニュアルやオブジェクトブラウザを参照してください。

注：これらのオブジェクトは、AppleScript では利用できません。

ScriptListener によるスクリプトの記録

ここでは、ScriptListener を使用してスクリプトログファイルを作成する方法について説明します。次の例では、ドキュメントにエンボスフィルタを適用するために必要なアクションを記録します（デフォルトでは、エンボスフィルタは Photoshop のユーザインターフェイスでしか実行できません）。

注：次の手順を行う前に、ScriptListener を Automate フォルダにインストールする必要があります。[73 ページの「ScriptListener のインストール」](#)を参照してください。

エンボスフィルタをスクリプト可能にするには：

1. Photoshop を開き、次にドキュメントを開きます。
2. **ウィンドウ／アクション**を選択し、次にアクションパレットのメニューから「**新規アクション**」を選択します。
3. アクションに名前を付け、「**記録**」をクリックします。
4. **フィルタ／表現手法／エンボス**を選択します。
5. 次の設定を使用します。
 - 角度：135
 - 高さ：3
 - 量：100
6. 「**OK**」をクリックします。

7. スクリプトログファイルを確認します。
 - Windows では、ログファイルはデスクトップに保存されます。
 - Macintosh では、ログファイルはデスクトップに保存されます。

JavaScript からのアクションマネージャの使用

ここでは、ScriptingListenerJS.log ログの内容を使用してスクリプトを作成する方法について説明します。この節の手順を行う前に、既にアクションを記録している必要があります。この節の例では、[74 ページの「ScriptListener によるスクリプトの記録」](#)の手順を完了していることを前提としています。

ここでは、アクションマネージャを使用して、スクリプティングインターフェイスからエンボスフィルタを利用できるようにします。

ScriptListener の出力から JavaScript を作成するには：

1. 次のように行います。
 - デスクトップにある ScriptingListenerJS.log を開きます。

ファイルの最後に、次のようなコードが記録されています（数字は異なる可能性があります）。

```
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 );
```

注：ScriptListener は、コマンドを等号の水平線(====...)で区切って記録します。このアクションを記録する前に、既にいくつかアクションを記録している場合は、最後の等号線を見つけてください。その後ろにあるのが最新のアクションです。

2. エンボスアクションに関連する JavaScript コードを、ScriptListenerJS.log から別のファイルにコピーします（ファイル名は emboss.jsx にします）。
3. emboss.jsx スクリプトで、フィルタに使用した値（135、3、100）を見つけます。これらの設定値を、変数名に置き換えます。

次の例では、135 を angle に、3 を height に、100 を amount に置き換えています。

```
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. このコードを JavaScript 関数でラップします。次の例では、関数名を `emboss` にしています。

```
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. このようにエンボス関数を定義し、必要なパラメータを指定して呼び出せば、JavaScript でドキュメントにエンボスフィルタを適用することができます。例えば、次の例では、角度 75、高さ 2、量 89 のエンボスフィルタを適用しています（スクリプトでドキュメントを開く方法については、[28 ページの「ドキュメントのオープン」](#)を参照してください）。

```
// 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. エンボスフィルタを適用してみます。Photoshop を開き、**ファイル／スクリプト／参照**を選択し、作成した `emboss.jsx` スクリプトの場所に移動します。「開く」を選択して、スクリプトを実行します。

VBScript からのアクションマネージャの使用

ここでは、`ScriptingListenerVB.log` ログの内容を使用してスクリプトを作成する方法について説明します。この節の手順を行う前に、既にアクションを記録している必要があります。この節の例では、[74 ページの「ScriptListener によるスクリプトの記録」](#)の手順を完了していることを前提としています。

ここでは、アクションマネージャを使用して、スクリプティングインターフェイスからエンボスフィルタを利用できるようにします。

ScriptListener の出力から VBScript を作成するには：

1. デスクトップにある ScriptingListenerVB.log を開きます。

ファイルの最後に、次のようなコードが記録されています（数字は異なる可能性があります）。

```
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 )
```

注：ScriptListener は、コマンドを等号の水平線 (====...) で区切って記録します。このアクションを記録する前に、既にいくつかアクションを記録している場合は、最後の等号線を見つけてください。その後ろにあるのが最新のアクションです。

2. エンボスアクションに関連する VBScript コードを、ScriptListenerVB.log から別のファイルにコピーします（ファイル名は emboss.vbs にします）。
3. emboss.vbs スクリプトで、フィルタに使用した値（135、3、100）を見つけます。これらの設定値を、変数名に置き換えます。

次の例では、135 を angle に、3 を height に、100 を amount に置き換えています。

```
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. このコードを VBScript 関数でラップします。次の例では、関数名を `Emboss` にしています。次の手順で説明しているように、Photoshop アプリケーションオブジェクトはこの関数の外で作成する必要があります。

```
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. このようにエンボス関数を定義し、必要なパラメータを指定して呼び出せば、VBScript でドキュメントにエンボスフィルタを適用することができます。例えば、次の例では、角度 75、高さ 2、量 89 のエンボスフィルタを適用しています。このスクリプトでは、この関数を呼び出す前に、ドキュメントを開く必要があります（スクリプトでドキュメントを開く方法については、[28 ページの「ドキュメントのオープン」](#)を参照してください）。ドキュメントを開くには `Application.Open` メソッドを呼び出しますが、このメソッドを呼び出すためには Photoshop の DOM にアクセスする必要があります。Photoshop.Application object before it opens a new document.

```
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. emboss.vbs ファイルをダブルクリックして、エンボスフィルタスクリプトを適用します。Photoshop が起動し、ファイルが開き、そのファイルにエンボスフィルタが適用されます。

VBScript での JavaScript ベースのアクションマネージャコードの実行

DoJavaScriptFile メソッドを使用すれば、VBScript で JavaScript ベースのアクションマネージャコードにアクセスすることができます。Application.DoJavaScriptFile メソッドについて詳しくは、VBScript のオブジェクトブラウザを参照してください。

VBScript で JavaScript ベースのアクションマネージャコードを実行するには：

1. [75 ページの「JavaScript からのアクションマネージャの使用」](#)の手順 1 から 4 を行って、次の JavaScript コードを含むファイル (emboss.jsx) を作成します。

```

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. emboss.jsx ファイルの末尾に、次の JavaScript コードを追加します。これによって、外部から引数を渡してエンボス関数を実行できるようになります。VBScript から JavaScript に引数を渡す方法について詳しくは、『Adobe Intro to Scripting』を参照してください。

```

// Call emboss with values provided in the "arguments" collection
emboss( arguments[0], arguments[1], arguments[2] );

```

3. 次のようにすれば、VBScript からエンボスフィルタを実行できます（この例では、emboss.jsx は C:¥ にあるものとしています）。

```
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)
```

AppleScript での JavaScript ベースのアクションマネージャコードの実行

AppleScript 用のアクションマネージャ機能は用意されていませんが、do javascript コマンドを使用すれば、AppleScript で JavaScript コードやファイルを実行することができます。詳しくは、『Adobe Intro to Scripting』を参照してください。

1. [75 ページの「JavaScript からのアクションマネージャの使用」](#)の手順 1 から 4 を行って、次の JavaScript コードを含むファイル（emboss.jsx）を作成します。

```
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. emboss.jsx ファイルの末尾に、次の JavaScript コードを追加します。これによって、外部から引数を渡してエンボス関数を実行できるようになります。AppleScript から JavaScript に引数を渡す方法について詳しくは、『Adobe Intro to Scripting』を参照してください。

```
// Call emboss with values provided in the "arguments" collection
emboss( arguments[0], arguments[1], arguments[2] );
```

3. 次の AppleScript コードサンプルでは、ドキュメントを開いて、エンボスフィルタを実行しています。

```
tell application "Adobe Photoshop CS6"
    set theFile to alias "Application:Documents:MyFile"
    open theFile
    do javascript (file <path to Emboss.jsx>) ~
        with arguments { 75,2,89 }
end tell
```


ScriptListener によるイベント ID やクラス ID の確認

ここでは、Photoshop で行われたアクションのイベント ID やクラス ID を、ScriptListener を使用して確認する方法について説明します。イベント ID やクラス ID は、[Notifier クラス](#) で通知を設定するときに必要なになります。

ScriptListener を使用すると、記録可能なアクションのイベント ID を確認できます。まず、[73 ページの「ScriptListener のインストール」](#)の説明に従って、ScriptListener プラグインをインストールします。次に、イベント ID を確認したいアクションを実行します。イベントが Script Listener のログファイルに記録されます（[73 ページの「ScriptListener プラグイン」](#)を参照）。そのイベントが複数のクラスのオブジェクトで使用されている場合は、クラス ID もログファイルに記録されます。

「Open Document」イベントのイベント ID を確認する方法と、「New」イベントのイベント ID とクラス ID を確認する方法を、次の例に示します。「New」イベントは、複数のクラスで使用されているイベントです。

「Open Document」イベントのイベント ID の確認

1. ScriptListener プラグインがインストールされていることを確認します。
2. Photoshop を開き、次にドキュメントを開きます。
3. ScriptListener ログファイルを見つけて開きます。VBScript ログファイルか JavaScript ログファイルのいずれかを使用します。JavaScript のファイルの最後には、次のようなコードが記録されています。等号線の下にあるすべての行が、最後に実行されたアクションのログです。

```
// =====
var id14 = charIDToTypeID( "Opn " );
var desc5 = new ActionDescriptor();
var id15 = charIDToTypeID( "null" );
desc5.putPath( id15, new File( "C:¥¥Program Files¥¥Adobe¥¥Adobe Photoshop CS6¥¥
  Samples¥¥Fish.psd" ) );
executeAction( id14, desc5, DialogModes.NO );
```

4. executeAction メソッドは、スクリプトでアクションを実行するためのメソッドです。このメソッドには、実行するアクションを識別するためのイベント ID を指定する必要があります。このメソッドにイベント ID を提供しているのは、最初の引数（この場合は id14）です。この id14 という変数は、数行前のコードで定義されています。それを見ると、「Open Document」アクションのイベント ID が「Opn 」であることがわかります。
5. このイベント ID を使用すれば、「Open Document」アクションに対するイベント通知をスクリプトで設定することができます。例えば、JavaScript では次のようにします。

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

「New」イベントのイベント ID とクラス ID の確認

1. ScriptListener プラグインがインストールされていることを確認します。
2. Photoshop を開き、ファイル／新規を選択して新規ドキュメントを作成します。
3. 次に、チャンネルパレットの新規チャンネルを作成アイコンを使用して、新しいチャンネルを作成します。

4. ScriptListener ログファイルを見つけて開きます。VBScript ログファイルか JavaScript ログファイルのいずれかを使用します。2つのアクションを記録したので、等号行で区切られた最後の2つのセクションを参照します。JavaScript のログファイルでは、次のようなコードが記録されています。

```
// =====
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. 最初のセクションは、「New Document」イベントを実行するスクリプティングコードを表します。2 番目のセクションは、「New Channel」イベントのスクリプティングコードを表します。
6. どちらのアクションの `executeAction` メソッドにも、"Mk " という値の引数が渡されています (`id17` と `id34` を参照してください)。これは、「New」アクションのイベント ID です。このアクションを実行するためには、使用するクラス、つまりイベントのクラス ID も指定する必要があります。
7. `putObject` メソッドは、アクションの実行対象となるクラスを識別します。必要なクラス ID は、`putObject` の 2 番目の引数に指定されています。最初のアクションでは、`id33` は "Dcmn" と定義されています。2 番目のアクションでは、`id45` は "Chnl" と定義されています。これらが、Document と Channel のクラス ID です。
8. このイベント ID とクラス ID を使用すれば、「New Document」イベントや「New Channel」イベントに対するイベント通知をスクリプトで設定することができます。例えば、JavaScript では次のようにします。

```
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")
```

索引

数字

16 進数値、設定, 52

A

Adobe Photoshop のオブジェクトモデル, 11, 33

AppleScript

AppleScript からの JavaScript の実行, 11

作成, 18

実行, 18

単位値に関する考慮事項, 56

表記規則, 6

Applescript

用語説明の表示, 21

Application オブジェクト

display dialogs, 33

参照する, 22

使用, 34

ターゲットにする, 22

定義, 12

ユーザインターフェイスとの関係, 14

Art Layer オブジェクト

JavaScript での追加, 25

VBScript での追加, 24

作成, 37

参照, 38

スタイルの適用, 40

操作, 36

定義, 12

テキストレイヤーに変更する, 41

フィルタ, 53

ユーザインターフェイスとの関係, 15

C

Channel オブジェクト

アクティブなチャンネルの設定, 28

アクティブにする, 28

操作, 46

タイプの変更, 46

定義, 13

ユーザインターフェイスとの関係, 15

Channel オブジェクト、種類, 13

Color Sampler オブジェクト

定義, 14

ユーザインターフェイスとの関係, 15

Color オブジェクト

16 進数値の設定, 52

DOM に含まれる, 16

Solid Color クラス, 52

Web セーフ, 53

取得と変換, 52

操作, 51

定義, 51

テキストへの適用, 64

比較, 53

Count Item オブジェクト

定義, 14

ユーザインターフェイスとの関係, 15

D

display dialogs, 33

Document Info オブジェクト

使用, 47

定義, 13

ユーザインターフェイスとの関係, 15

Document オブジェクト

アクティブにする, 26

使用, 34

操作, 35

単位値, 58

追加, 24

定義, 12

ドキュメント情報, 47

開く, 28

保存, 31

ユーザインターフェイスとの関係, 14

E

EPS open options オブジェクト、単位値, 58

H

Hello World スクリプト, 17-20

History State オブジェクト

使用, 47

定義, 13

復帰, 48

メモリのクリア, 48

ユーザインターフェイスとの関係, 15

J

JavaScript

AppleScript から実行, 11

VBScript から実行, 11

アクションマネージャの使用, 75

作成, 19

サポート, 10

実行, 10, 19
表記規則, 6
ワークフローを自動化するためのサンプル, 60

L

Layer Comp オブジェクト
 定義, 13
 ユーザインターフェイスとの関係, 15
Layer Set オブジェクト
 作成, 38
 操作, 36, 39
 定義, 12
 ユーザインターフェイスとの関係, 15
Layer オブジェクト
 アクティブにする, 27
 作成, 37
 参照, 38
 スタイルの適用, 40
 操作, 36
 タイプの特定, 41
 追加, 25
 定義, 12
 テキストレイヤーかどうか調べる, 41
 リンク, 40
lens flare open options オブジェクト、単位値, 58

M

Measurement Scale オブジェクト
 定義, 14
 ユーザインターフェイスとの関係, 15

N

Notifier オブジェクト
 イベント ID の確認, 81
 クラス ID の確認, 81
 使用, 48
 定義, 14
 ユーザインターフェイスとの関係, 15

O

offset filter オブジェクト、単位値, 58

P

Path Item オブジェクト
 直線の作成, 49
 定義, 14
 ユーザインターフェイスとの関係, 15
Path Point オブジェクトの定義, 14
PDF open options オブジェクト、単位値, 58
Photoshop, 54
Photoshop のオブジェクトモデル, 11
Preferences オブジェクト

定義, 14
ユーザインターフェイスとの関係, 15

S

ScriptListener
 アンインストール, 74
 イベント ID の確認, 81
 インストール, 73
 クラス ID の確認, 81
 スクリプトの記録, 74
 ログファイル, 73
Selection オブジェクト
 境界のぼかし, 44
 サイズ変更, 44
 作成, 43, 44
 操作, 42
 定義, 13
 塗りつぶし, 45
 反転, 44
 復元, 46
 保存, 45
 ユーザインターフェイスとの関係, 14
 読み込み, 45
 領域の定義, 67
Solid Color クラス, 52
Sub Path Item オブジェクトの定義, 14

T

Text Item オブジェクト
 作成, 41
 操作, 41
 単位値, 58
 定義, 12
 テキストの書式設定, 42

V

VBScript
 VBScript からの JavaScript の実行, 11
 アクションマネージャの使用, 76
 作成, 19
 実行, 19
 タイプライブラリ, 22
 表記規則, 6

W

Web セーフカラー, 53

あ

アクション
 スクリプトとの比較, 8
 操作, 73
アクションパレット, 73

アクションマネージャ

AppleScript での JavaScript コードの実行, 80

JavaScript からの使用, 75

VBScript からの使用, 76

VBScript での JavaScript コードの実行, 79

スクリプティングオブジェクト, 74

定義, 73

アクティブなオブジェクト、設定, 25

値の型

定数, 16

い

イベント ID、ScriptListener による確認, 81

イベント通知、設定, 48

お

オープンオプションクラス, 16

オブジェクト

個々のオブジェクトも参照

Adobe Photoshop のオブジェクトモデル, 11

アクティブにする, 25

階層, 11

スクリプトでの作成, 23–25

表示, 21

オブジェクトモデル

概念, 11

操作, 33

親オブジェクトの定義, 26

か

階層, 11

画像、コンポジションの変更, 12

環境設定

設定, 32

操作, 61

く

クラス ID、ScriptListener による確認, 81

クラス、調べる, 21

クリップボードコマンド, 54

け

計算、単位値, 57

こ

コピーおよびペーストコマンド, 54

コマンド

表記規則, 7

表示, 21

コレクション、VBScript のインデックス, 11

コンポーネントチャンネル, 13

さ

作成

選択範囲, 44

テキスト, 64

し

定規単位

値, 56

値の使用, 57

設定, 59

定義, 56

条件判断のロジック, 9

す

スクリプト

アクションとの比較, 8

オブジェクトの作成, 23–25

機能, 8, 9

記録, 74

高度, 60

作成, 60

実行, 10

スタートアップ, 10

定義, 8

ファイルの場所, 10

有効なファイル拡張子, 10

スクリプトエディター

使用, 18

スクリプト言語

サポート, 9

スクリプトの例, 17

スタートアップスクリプト, 10

スタイル、レイヤーへの適用, 40

スポットカラーチャンネル, 13

せ

選択領域チャンネル, 13

そ

測定単位

操作, 56

ドキュメントの環境設定, 61

た

ダイアログボックス、制御, 33

タイプライブラリ、VBScript, 22

単位

AppleScript での考慮事項, 56

値, 56

値の使用, 57

計算, 57

- 設定, 59
- 操作, 56
- 特別なタイプ, 56
- パラメータとして, 58
- 引数, 58
- プロパティとして, 57

て

- 定数
 - 調べる, 16, 21
 - 定義, 16
- テキスト
 - カラーの適用, 64
 - 作成, 64
 - 書式設定, 42
 - レイヤー, 41
- テキストレイヤー, 41

と

- ドキュメントオブジェクトモデル (DOM)
 - オブジェクトモデルを参照
- ドキュメントの保存, 31

は

- 波形フィルタ、適用, 66–70
- パス、作成, 49

ひ

- ヒストリー
 - 定義, 47
- 表記規則, 6

ふ

- ファイル
 - 形式の指定, 29
 - 形式の推測, 28
 - 設定を指定して開く, 29
 - 開く, 28
 - 保存, 31
- ファイル拡張子
 - スクリプトファイル, 10
- フィルタ
 - スクリプト可能にする, 74
 - 操作, 53
 - その他, 54
 - 波形の適用, 66–70
 - ぼかし（移動）の適用, 71
- プロパティ
 - 調べる, 21
 - 表記規則, 7

へ

- ペーストコマンド, 54

ほ

- 包含階層, 11
- ぼかし（移動）フィルタ、定義, 71
- 保存オプションクラス, 16

ま

- マージコピー, 55
- マージコピーコマンド, 55
- マスク範囲チャンネル, 13

め

- メソッド
 - 表記規則, 7
 - 表示, 21
- メタデータの定義, 15

も

- 文字単位
 - 設定, 59
 - 定義, 56

れ

- レイヤークラス, 12
- 列挙型の値
 - 調べる, 21

ろ

- ロジック、条件判断, 9

わ

- ワークフローの自動化、JavaScript, 60