ADOBE® INDESIGN® CS5

ADOBE INDESIGN CS5
スクリプティングチュートリアル

Adobe

Adobe® InDesign® CS5 スクリプティングチュートリアル

目次

Adobe InDesign CS5 スクリプティングチュートリアル

概要

Adobe® InDesign® CS5 には、強力なスクリプティング機能が搭載されています。この機能を活用すれば、時間や手間やコストを大幅に削減することができます。

このマニュアルでは、すべての InDesign ユーザーを対象に説明を行います。まず、スクリプトを初めて作成する方のために、スクリプトの基本事項について説明します。次に、他のアプリケーション向けにスクリプトを作成した経験がある方のために、InDesign スクリプティングの特徴について説明します。

さらに、スクリプトを InDesign にインストールして実行する方法や、InDesign スクリプティングで実現できることと実現できないこと、スクリプトの作成に必要なソフトウェアについても説明します。

このチュートリアルを読み終えて InDesign スクリプティングの基本を理解したら、『Adobe InDesign CS5 スクリプティングガイド』に進んで、より高度なスクリプティング技術を習得することができます。この『Adobe InDesign CS5 スクリプティングガイド』では、テキストのフォーマット、テキストの検索や変更、メニュー項目へのスクリプトの関連付け、ページ上のオブジェクトの描画、ドキュメントの書き出しなどの、様々な処理を実行するチュートリアルスクリプトを紹介しています。

はじめに

ユーザーインターフェイスで行える操作のほとんどは、スクリプトを使用して実行することができます。フレームの描画、テキストの入力やフォーマット、グラフィックの配置、ドキュメントページの印刷や書き出しなど、ドキュメントやその内容に変更を加えるアクションは、すべてスクリプトで実行できます。また、ユーザーインターフェイスでは行えない高度な処理も実行できます。

スクリプトを使用すれば、メニューの作成、メニュー項目の追加、ダイアログボックスやパネルの作成と表示、ユーザーインターフェイスで行われた選択への応答などが行えます。また、テキストファイルの読み取りや書き込み、XML データの解析、他のアプリケーションとの通信なども行えます。

スクリプトは、カーソル位置でのタブストップの設定のような単純な処理から、非常に複雑な処理まで、様々な用途に活用できます（例えば、InDesign の XHTML 書き出し機能は、スクリプトを使用して実現されています）。最初は、1 つの処理を実行する簡単なスクリプトから始めましょう。慣れてくると、パブリッシングワークフロー全体を自動化するスクリプトを作成できるようになります。

ワークスペースの設定やキーボードショートカットの定義など、スクリプトで実行できない操作がいくつかありますが、その多くはユーザーインターフェイスで行うべき操作です。また、InDesign ドキュメントに新しいタイプのオブジェクトを追加したり、InDesign の基本機能を新たに追加する（例えば、新しいテキスト組版エンジンを追加する）こともできません。このような拡張を行いたい場合は、InDesign ソフトウェア開発キット（SDK）を使用して、C++ でコンパイル済みのプラグインを作成する必要があります。

スクリプティングは、時間のかかる反復作業を行う場合（例えば電話帳のレイアウトなどを行う場合）に有効であると考えていますが、次のような場合にも活用できます。

- こまごまとした日常の面倒な作業を自動化する。
- 自分の好みや作業スタイルに合わせて InDesign をカスタマイズする。
- 通常の方法では困難または不可能であるクリエイティブな効果を狙う。

スクリプトのインストール

InDesign のスクリプトは簡単にインストールできます。InDesign アプリケーションフォルダー内の Scripts フォルダーの下にある Scripts Panel フォルダーに、スクリプトファイルを保存します （Scripts フォルダーが存在しない場合は、作成してください）。

または、環境設定フォルダーの Scripts Panel フォルダーにスクリプトを保存します。環境設定フォルダーの場所は、次のとおりです。

Windows® XP： `C:¥Documents and Settings¥<ユーザー名>¥Application Data¥Adobe¥InDesign¥Version 7.0-J¥<ロケール>¥Scripts¥Scripts Panel`

Windows® Vista： `C:¥ユーザー¥<ユーザー名>¥AppData¥Roaming¥Adobe¥InDesign¥Version 7.0-J¥<ロケール>¥Scripts¥Scripts Panel`

Mac OS®： `/ユーザ/<ユーザー名>/ライブラリ/Preferences/Adobe InDesign/Version 7.0-J/<ロケール>/Scripts/Scripts Panel`

<ユーザー名> はユーザー名を表し、<ロケール> は場所と言語（例 ja_JP）を表します。

このフォルダーに保存したスクリプトは、InDesign のスクリプトパネルに表示されます。このパネルを表示するには、ウィンドウ／ユーティリティ／スクリプトを選択します。

Scripts Panel フォルダーには、スクリプト（またはスクリプトが保存されているフォルダー）のエイリアスやショートカットを保存することもできます。これらのエイリアスやショートカットも、スクリプトパネルに表示されます。

InDesign の起動時に特定のスクリプトを自動実行させるには、Scripts フォルダーの下にある「Startup Scripts」という名前のフォルダーにスクリプトファイルを保存します（このフォルダーが存在しない場合は、作成してください）

スクリプトの実行

スクリプトを実行するには、スクリプトパネルを表示し（ウィンドウ／ユーティリティ／スクリプトを選択し）、スクリプトパネルに表示されているスクリプト名をダブルクリックします。多くの場合、何らかのユーザーインターフェイス（ダイアログボックスやパネルなど）が表示され、必要に応じて警告が表示されます。

スクリプトパネルの使用

InDesign のスクリプトパネルを使用すれば、InDesign のほとんどのスクリプトを簡単かつ最適な方法で実行できます。このパネルが表示されていない場合は、ウィンドウ／ユーティリティ／スクリプトを選択して表示できます。

スクリプトは、Finder（Mac OS®）やエクスプローラー（Windows®）から実行するよりも、スクリプトパネルから実行するほうが実行速度が速くなります。

スクリプトの実行速度を速くするには、スクリプトの実行中に画面の再描画を行わないようにするという方法もあります。これを行うには、スクリプトパネルのメニューで「再描画を有効にする」をオフにします。スクリプトの実行中にその動作を表示させたい場合は、このオプションをオンにします。

スクリプトパネルでは、コンパイル済みまたは未コンパイルの AppleScript（拡張子が .spt、.as、.applescript のファイル）、JavaScript（拡張子が .js または .jsx のファイル）、VBScript（拡張子が .vbs のファイル）、実行可能ファイルを実行できます。

スクリプトパネルに表示されているスクリプトを編集するには、スクリプトを選択し、スクリプトパネルのメニューから「スクリプトを編集」を選択します。Option キー（Mac OS）または Alt キー（Windows）を押しながらスクリプト名をダブルクリックする方法もあります。スクリプトのファイルタイプに対応するエディターで、そのスクリプトファイルが開かれます。

スクリプトパネルに表示されているスクリプトの保存フォルダーを開くには、スクリプトを選択し、「Finder で表示」（Mac OS）または「エクスプローラーで表示」（Windows）を選択します。Command キー（Mac OS）または Ctrl キーと Shift キー（Windows）を押しながらスクリプト名をダブルクリックする方法もあります。そのスクリプトが保存されているフォルダーが、Finder（Mac OS）またはエクスプローラー（Windows）で表示されます。

スクリプトは通常、一連のアクションとして実行されます。つまり、編集メニューから「取り消し」を選択して、スクリプトで行った各変更を取り消すことができます。スクリプトのトラブルシューティングを行う場合は、これによって問題のアクションが見つけやすくなります。スクリプトでは取り消しモードを変更し、すべてのスクリプトアクションを単一の取り消し手順に含めることができます。これにより、スクリプトの実行速度が大幅に速くなります。この操作の実行について詳しくは、該当するスクリプト言語の『Adobe InDesign CS5 スクリプティングガイド』を参照してください。

スクリプトにキーボードショートカットを追加するには、編集／キーボードショートカットを選択し、セットメニューから編集可能なショートカットセットを選択し、機能エリアメニューから「スクリプト」を選択します。スクリプトパネル内のスクリプトが一覧表示されます。スクリプトを選択し、InDesign の通常の機能と同じようにキーボードショートカットを割り当てます。

スクリプト言語について

どの言語でスクリプトを作成するかは、使用しているプラットフォームで採用されているスクリプティングシステムによって決まります。Mac OS の場合は AppleScript が、Windows の場合は VBScript が使用できます。また、JavaScript はどちらのプラットフォームでも使用できます。このように、スクリプト言語にはいくつかの種類がありますが、InDesign に対する動作はどれもよく似ています。

このマニュアルでサンプルスクリプトを掲載する場合は、同様の処理を行うスクリプトを 3 つの言語で示します。ある言語のスクリプトを別の言語のスクリプトに翻訳するのは、非常に簡単です。

JavaScript

InDesign では、クロスプラットフォーム（Mac OS と Windows）でのスクリプティングを考慮して、JavaScript がサポートされています。InDesign では、ExtendScript と呼ばれるアドビ システムズ社独自の JavaScript 実装に基づいて JavaScript がサポートされています。ExtendScript インタープリターは、JavaScript の現行の標準である ECMA 262 に準拠しており、JavaScript 1.5 の言語機能がすべてサポートされています。Adobe Illustrator® や Adobe Photoshop® などの Adobe Creative Suite® 製品でも、ExtendScript JavaScript インタープリターが使用されています。

Microsoft® JScript（Windows）や、Late Night Software 社の OSA JavaScript（Mac OS）などの他のバージョンの JavaScript でもスクリプトを作成できますが、これらの言語と ExtendScript では使用されている用語が異なります。ExtendScript のサンプルは、他のバージョンの JavaScript では動作しません。

注意 : ExtendScript のツールや機能が複数のアドビ製品に搭載されたのに伴って、ExtendScript のマニュアルはすべて統合されました。ScriptUI ユーザーインターフェイスモジュールなどの JavaScript ユーティリティや、ExtendScript Toolkit（オブジェクトモデルを参照できる JavaScript 開発環境）について詳しくは、『Creative Suite 5 JavaScript Tools Guide』を参照してください。

Windows

Windows で InDesign スクリプティングを行う場合は、JavaScript または Microsoft Visual Basic の特定のバージョン（VBScript など）が使用できます。

Visual Basic のチュートリアルスクリプトは、VBScript で書かれています。VBScript を採用したのは、実行や編集に追加のソフトウェアが必要ないからです。VBScript は、任意のテキストエディター（メモ帳など）で編集でき、InDesign のスクリプトパネルから実行できます。

Visual Basic のその他のバージョンには、Visual Basic 5 Control Creation Edition（CCE）、Visual Basic 6、Visual Basic .NET、Visual Basic 2005 Express Edition があります。InDesign スクリプティングには、Visual Basic .NET より前のバージョンの Visual Basic が適しています。Visual Basic .NET 以降のバージョンでは、InDesign スクリプティングでよく使用される `Variant` データ型が利用できません。

Microsoft Word、Microsoft Excel、Microsoft Visio、AutoCAD などの多くのアプリケーションには、Visual Basic for Applications（VBA）が搭載されています。VBA を使用して InDesign のスクリプトを作成することはできますが、InDesign に VBA は搭載されていません。

Windows で VBScript や Visual Basic を使用して InDesign スクリプティングを行う場合は、管理者権限を持つユーザーアカウントで InDesign をインストールする必要があります。インストール後は、すべてのユーザーが InDesign スクリプトを実行できます。Power User 権限または Administrator 権限を持つユーザーは、InDesign のスクリプトパネルにスクリプトを追加できます。

Mac OS

Mac OS で InDesign スクリプティングを行う場合は、JavaScript または AppleScript が使用できます。AppleScript を作成するには、AppleScript バージョン 1.6 またはそれ以降がインストールされているシステムで、AppleScript のスクリプトエディタを使用する必要があります。AppleScript は、すべての Mac OS® に付属しています。また、Apple 社の Web サイトから無償でダウンロードすることもできます。Mac OS には Apple 社のスクリプトエディタが付属しています。スクリプトエディタには、次のメニューからアクセスできます。

Mac OSX 10.5　　アプリケーション／`AppleScript`／スクリプトエディタ

Mac OSX 10.6　　アプリケーション／ユーティリティ／`AppleScript` エディタ

また、Script Debugger（Late Night Software 社の製品。http://www.latenightsw.com）などのサードパーティーのスクリプトエディタを使用することもできます。

使用するスクリプト言語の選択

スクリプトを作成した経験がある方は、任意の言語を使用してください。スクリプトを初めて作成する方や、Mac OS 版と Windows 版の両方の InDesign でスクリプトを実行したい場合は、JavaScript を使用します。システム上のアドビ以外のアプリケーションと通信する必要がある場合は、使用しているプラットフォームの標準言語（Mac OS の場合は AppleScript、Windows の場合は VBScript）を使用します。

AppleScript、JavaScript、VBScript の言語機能については、このガイドですべてを解説することはできませんので、必要に応じてそれらのスクリプト言語に関する資料を参照してください。

注意 : このほかにも、プラットフォームの標準オートメーションシステムを操作できるプログラミング言語（Python や C# など）はほぼすべて使用できますが、その解説はこのマニュアルの範疇を超えています。

このマニュアルに掲載されているスクリプトの使用方法

このマニュアルに記載されているスクリプトを確認または編集するには、該当するスクリプトの編集アプリケーションで、対象のチュートリアルスクリプトファイルを開きます（ファイル名は各スクリプトの前に記載されています）。

スクリプトを実行するには、Scripts Panel フォルダーにスクリプトをインストールし（6 ページの「スクリプトのインストール」を参照）、次の操作を行います。

1. ウィンドウ／ユーティリティ／スクリプトを選択して、スクリプトパネルを表示します。
2. スクリプトパネルでスクリプト名をダブルクリックして、スクリプトを実行します。

編集したスクリプトを保存するには、Scripts Panel フォルダーにプレーンテキストファイルとしてスクリプトを保存します（6 ページの「スクリプトのインストール」を参照）。ファイル拡張子は次のものを使用します。

AppleScript： `.applescript`

JavaScript： `.jsx`

VBScript： `.vbs`

注意 : このマニュアルに掲載されている JavaScript の例を直接入力する場合は、掲載されているとおりに大文字と小文字を使い分けることが重要です。JavaScript では大文字と小文字が区別されるので、使い分けを誤ると正しく実行されません。AppleScript や VBScript の例では、大文字と小文字は区別されません。このチュートリアルに付属しているスクリプトファイルを使用することをお勧めします。

注意 : このマニュアルからスクリプトをコピーして貼り付ける場合は、レイアウトの都合で途中で改行されている文があるとエラーになる点に注意してください。このようなエラーを発見するのは難しいことが多いので、このチュートリアルに付属しているスクリプトを使用することをお勧めします。

最初の InDesign スクリプトの作成

では、InDesign スクリプトを実際に作成してみましょう。ここでは、新規ドキュメントを作成し、テキストフレームを追加し、そのテキストフレームにテキストを入力するスクリプトを作成します。このスクリプトでは、次の処理を行います。

- InDesign との通信を確立する。
- 新規ドキュメントを作成する。
- 特定のページにテキストフレームを作成する。
- テキストフレームにテキストを追加する。

AppleScript

スクリプトエディタアプリケーションを起動します（AppleScript フォルダーの中の Applications フォルダーにあります）。次のスクリプトを入力します（または、`HelloWorld.applescript` というチュートリアルスクリプトを開きます）。

```
tell application "Adobe InDesign CS5"
set myDocument to make document
tell page 1 of myDocument
set myTextFrame to make text frame
set geometric bounds of myTextFrame to {"6p", "6p", "24p", "24p"}
set contents of myTextFrame to "Hello World!"
end tell
end tell
```

Scripts Panel フォルダーに、テキストファイルとしてスクリプトを保存します（6 ページの「スクリプトのインストール」を参照）。ファイル拡張子は `.applescript` を使用します。スクリプトを実行するには、スクリプトパネルでスクリプト名をダブルクリックします。または、スクリプトエディタウィンドウで「実行」をクリックします。

JavaScript

ExtendScript Toolkit（またはテキストエディター）を起動します。次のスクリプトを入力します（または、`HelloWorld.jsx` というチュートリアルスクリプトを開きます）。

```
var myDocument = app.documents.add();
var myTextFrame = myDocument.pages.item(0).textFrames.add();
myTextFrame.geometricBounds = ["6p", "6p", "24p", "24p"];
myTextFrame.contents = "Hello World!";
```

Scripts Panel フォルダーに、プレーンテキストファイルとしてスクリプトを保存します（6 ページの「スクリプトのインストール」を参照）。ファイル拡張子は .jsx を使用します。スクリプトを実行するには、スクリプトパネルでスクリプト名をダブルクリックします。または、ExtendScript Toolkit で、ターゲットアプリケーションのポップアップメニューから InDesign を選択し、「実行」ボタンをクリックします。

VBScript

テキストエディター（メモ帳など）を起動し、次のスクリプトを入力します（または、HelloWorld.vbs というチュートリアルスクリプトを開きます）。

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDocument = myInDesign.Documents.Add
Set myTextFrame = myDocument.Pages.Item(1).TextFrames.Add
myTextFrame.GeometricBounds = Array("6p", "6p", "24p", "24p")
myTextFrame.Contents = "Hello World!"
```

Scripts Panel フォルダーに、テキストファイルとしてスクリプトを保存します（6 ページの「スクリプトのインストール」を参照）。ファイル拡張子は .vbs を使用します。スクリプトを実行するには、スクリプトパネルでスクリプト名をダブルクリックします。

スクリプトの動作の説明

ここでは、Hello World スクリプトの動作について、順を追って説明します（該当するスクリプト言語の行を参照してください）。

1. InDesign アプリケーションオブジェクトとの通信を確立します。

   AppleScript：
   ```
   tell application "Adobe InDesign CS5"
   ```

   JavaScript：　アプリケーションは app で参照できます。

   VBScript：
   ```
   Set myInDesign = CreateObject("InDesign.Application")
   ```

2. 新規ドキュメントを作成して、その参照を取得します。

   AppleScript：
   ```
   Set myDocument to make document
   ```

   JavaScript：
   ```
   Var myDocument = app.documents.add();
   ```

   VBScript：
   ```
   Set myDocument = myInDesign.Documents.Add
   ```

3. 最初のページに新規テキストフレームを作成し、そのテキストフレームへの参照を取得します。

   AppleScript：
   ```
   tell page 1 of myDocument
   set myTextFrame to make text frame
   ```

   JavaScript：
   ```
   var myTextFrame = myDocument.pages.item(0).textFrames.add();
   ```

   VBScript：
   ```
   Set myTextFrame = myDocument.Pages.Item(1).TextFrames.Add
   ```

4. テキストフレームの境界線（上端、左端、下端、右端）の位置を設定します。ここでは、デフォルトの測定単位がどのように設定されていても、テキストフレームが正しいサイズに設定されるように、測定単位をオーバーライドしています（「p」はパイカを表します）。位置は値のリスト（配列）として指定します。配列の作成方法はスクリプト言語によって多少異なります。配列変数について詳しくは、14 ページの「配列変数」を参照してください。

   AppleScript：
   ```
   set geometric bounds of myTextFrame to {"6p", "6p", "24p", "24p"}
   ```

   JavaScript：
   ```
   myTextFrame.geometricBounds = ["6p", "6p", "24p", "24p"];
   ```

VBScript：
```
myTextFrame.GeometricBounds = Array("6p", "6p", "24p", "24p")
```

5. コンテンツのプロパティに文字列を設定して、テキストフレームにテキストを追加します。

AppleScript：
```
set contents of myTextFrame to "Hello World!"
```

JavaScript：
```
myTextFrame.contents = "Hello World!";
```

VBScript：
```
myTextFrame.Contents = "Hello World!"
```

スクリプティングと InDesign オブジェクトモデル

ここでは、スクリプト言語に関する一般的な用語と、InDesign スクリプティングに固有の用語について説明します。

スクリプティングの用語

ここでは、スクリプティングに関する一般的な用語や概念について説明します。

コメント

コメントを使用して、スクリプトに説明用のテキストを追加することができます。コメントはスクリプトの実行時には無視され、エラーが発生しないようになっています。スクリプトの処理について、自分や他人のために注釈を残しておきたい場合は、コメントを使用すると便利です。このマニュアルのチュートリアルスクリプトでも、コメントを使用しています。

コメントを入力するには：

- AppleScript ではコメントの左に「`--`」と入力します。または、「`(*`」と「`*)`」でコメントを囲みます。例えば、次のようになります。

```
--this is a comment
(* and so is this *)
```

- JavaScript ではコメントの左に「`//`」と入力します。または、「`/*`」と「`*/`」でコメントを囲みます。例えば、次のようになります。

```
// this is a comment
/* and so is this */
```

- VBScript ではコメントの左に「`Rem`」（「remark」の意味です）または「`'`」と入力します。1 行全体をコメントにする場合は、行の先頭にコメントのマーカーを入力します。例えば、次のようになります。

```
Rem this is a comment
' and so is this
```

値

InDesign スクリプティングで使用される値の例としては、テキストの文字のポイントサイズ、ページ上でのテキストフレームの位置、長方形の線のカラーなどがあります。値とは、スクリプトで処理を行う際に使用するデータのことです。

値にどのような種類のデータが含まれるかは、値の型によって定義されます。例えば、単語が格納されている値の型は文字列で、段落の行送りの値の型は数値です。通常、スクリプトで使用される値は数値かテキストです。InDesign スクリプティングで最もよく使用される値の型を、次の表に示します。

値の型	説明	例
Boolean	真（True）か偽（False）かを表す論理値。	`True`
Integer	整数（小数点が付かない数）。正の整数と負の整数があります。VBScript では、整数（Integer）の代わりに long データ型を使用できます。AppleScript では、整数（Integer）および実数（real）の代わりに、fixed または long データ型を使用できます。	`14`
Double（VBScript）、fixed または real（AppleScript）、number（JavaScript）	精度の高い数値。小数点が含まれることがあります。	`13.9972`
String	一連のテキスト文字。文字列は引用符で囲みます。	`"I am a string"`

Array（VBScript、JavaScript）またはリスト（AppleScript）	値のリスト（この値はどの型であってもかまいません）。	AppleScript:
		`{"0p0", "0p0", "16p4", "20p6"}`
		VBScript:
		`Array("0p0","0p0","16p4","20p6")`
		JavaScript:
		`["0p0","0p0","16p4","20p6"]`

値の型の変換

InDesign でサポートされているすべてのスクリプト言語には、変数の値の型を変換するための方法が用意されています。最もよく行われるのは、数値から文字列への変換（テキストの中に挿入したり、ダイアログボックスに表示したりする場合に行います）と、文字列から数値への変換（ポイントサイズやページ位置を設定する場合に行います）の 2 つです。次の例を参照してください。

AppleScript

```
--To convert from a number to a string:
set myNumber to 2
set myString to (myNumber as string)
--To convert from a string to a number:
set myString to "2"
set myNumber to (myString as integer)
--if your string contains a decimal value, use "as real" rather than "as integer"
```

JavaScript

```
//To convert from a number to a string:
myNumber = 2;
myString = myNumber + "";
//To convert from a string to an integer:
myString = "2";
myNumber = parseInt(myString);
//If your string contains a decimal value, use "parseFloat" rather than "parseInt":
myNumber = parseFloat(myString);
//You can also convert strings to numbers using the following:
myNumber = +myString;
```

VBScript

```
Rem To convert from a number to a string:
myNumber = 2
myString = cstr(myNumber)
Rem To convert from a string to an integer:
myString = "2"
myNumber = cInt(myString)
Rem If your string contains a decimal value, use "cDbl" rather than "cInt":
myNumber = cDbl(myString)
```

変数

変数は、値を格納する入れ物のようなものです。「変数」という名前が表しているように、この入れ物に格納する値は変更することができます。変数には数値が格納されることもありますし、テキスト文字列や、InDesign オブジェクトへの参照が格納されることもあります。それぞれの変数は名前を持っており、変数を参照するときにはこの名前を使用します。変数に値を格納するには、その変数にデータを割り当てます。

最初に作成したサンプルスクリプトでも、`myDocument` や `myTextFrame` という変数を使用しています。これらの変数を使用することで、そのオブジェクトの完全な名前（「`text frame 1 of page 1 of document 1`」や「`app.documents.item(0).pages.item(0).textFrames.item(0)`」など）を指定しなくて済むようにしています。

InDesign のすべてのサンプルスクリプトやチュートリアルスクリプトでは、変数名はすべて `my` で始まっています。このようにすることで、スクリプトで独自に作成した変数と、スクリプト言語の予約語を簡単に区別できるようにしています。

変数への値の割り当て

変数に値や文字列を割り当てるのは簡単です。その方法を次の表に示します。

言語	変数に値を割り当てる例

AppleScript	`set myNumber to 10`
	`set myString to "Hello, World!"`
	`set myTextFrame to make text frame at page 1 of myDocument`

JavaScript	`var myNumber = 10;`
	`var myString = "Hello, World!";`
	`var myTextFrame = myDocument.pages.item(0).textFrames.add();`

VBScript	`myNumber = 10`
	`myString = "Hello, World!"`
	`Set myTextFrame = myDocument.Pages.Item(1).TextFrames.Add`

注意 : JavaScript では、var が前に付いていない変数はすべて、デフォルトでグローバルと見なされます（つまり、特定の関数に関連付けられることはありません）。var は必須ではありませんが、複数の関数を使用しているスクリプトでは var を使用することをお勧めします。AppleScript と VBScript では、グローバル変数と明示的に定義しない限り、変数はローカルとみなされます。つまり、変数が作成された関数の外部でその変数を使用することはできません。

変数には、意味のある名前を付けるようにしてください。firstPage や corporateLogo などは良い例です。x や c などは悪い例です。意味のある名前を使えば、スクリプトが読みやすくなります。長い名前を付けても、スクリプトの実行速度に悪影響を与えることはありません。

変数は、1 つの単語である必要があります。ただし、読みやすくするために途中で大文字を使用したり（myFirstPage）、アンダースコアを使用したり（my_first_page）することができます。変数名の最初の文字に数字を使用することはできません。また、変数名に句読点や引用符を含めることもできません。

配列変数

AppleScript、JavaScript、VBScript では、配列がサポートされています。配列は変数の一種で、値のリストのことです。AppleScript では、配列のことをリストと呼びます。配列を定義する例を次に示します。

言語	配列を定義する例
AppleScript	`set myArray to {1, 2, 3, 4}`
JavaScript	`myArray = [1, 2, 3, 4];`
VBScript	`myArray = Array(1, 2, 3, 4)`
Visual Basic.NET	`myArray = New Double (1, 2, 3, 4)`

配列の中の要素を参照するには、その要素のインデックスを使用します。配列の先頭の要素は、VBScript と JavaScript では要素 0、AppleScript では要素 1 です。配列の要素を参照する例を次の表に示します。

言語	配列の要素を参照する例
AppleScript	`set myFirstArrayItem to item 1 of myArray`
JavaScript	`var myFirstArrayItem = myArray[0];`
VBScript	`myFirstArrayItem = myArray(0)`

注意 : Visual Basic では、OptionBase 文を使用して、配列の最初の要素を要素 1 に設定することができます。このマニュアルの例では、配列の最初の要素は要素 1 ではなく、デフォルトのとおり要素 0 としています。OptionBase を 1 に設定している場合は、掲載されているサンプルスクリプトの配列参照の部分を、すべて適切に変更する必要があります。

次の表のように、配列の中にさらに別の配列を含めることができます。

言語	例
AppleScript	`set myArray to {{0, 0}, {72, 72}}`
JavaScript	`var myArray = [[0,0], [72,72]];`
VBScript	`myArray = Array(Array(0,0), Array(72, 72))`
Visual Basic.NET	`myArray = New Array(New Double(0,0), NewDouble (0,0))`

変数の値の型を調べる

オブジェクトの値の型に基づいて判断を行いたい場合があります。例えば、選択されたテキストに対して処理を実行するスクリプトでは、選択されているものがページアイテムである場合には処理を停止する必要があります。すべてのスクリプト言語には、変数の型を判断するための方法が用意されています。

AppleScript

```
-- Given a variable of unknown type, "myMysteryVariable"...
set myType to class of myMysteryVariable
--myType will be an AppleScript type (e.g., rectangle)
```

JavaScript

```
//Given a variable of unknown type, "myMysteryVariable"...
myType = myMysteryVariable.constructor.name;
//myType will be a string corresponding to the JavaScript type (e.g., "Rectangle")
```

VBScript

```
Rem Given a variable of unknown type, "myMysteryVariable"...
myType = TypeName(myMysteryVariable)
Rem myType will be a string corresponding to the variable type (e.g., "Rectangle")
```

演算子

演算子は、変数や値を使用して計算（加算、減算、乗算、除算）を実行し、値を返します。例えば、

```
MyWidth/2
```

この例では、変数 `myWidth` に格納されている値の半分の値が返されます。

また、演算子を使用して比較（等しい（`=`）、等しくない（`<>`）、より大きい（`>`）、より小さい（`<`））を行うこともできます。例えば、

```
MyWidth > myHeight
```

この例では、`myWidth` が `myHeight` よりも大きい場合には `true`（または 1）が、そうでない場合には `false`（0）が返されます。

どのスクリプト言語にも、さらに便利な演算子が用意されています。AppleScript と VBScript では、アンパサンド（`&`）を使用して 2 つの文字列を連結（結合）させることができます。

```
"Pride " & "and Prejudice"
```

この例では、次の文字列が返されます。

```
"Pride and Prejudice"
```

JavaScript では、プラス記号（`+`）を使用して 2 つの文字列を結合させます。

```
"Pride " + "and Prejudice"
//returns the string: "Pride and Prejudice"
```

条件文

「もし選択されているオブジェクトが長方形なら、その線幅を 12 ポイントに設定しなさい」のような命令のことを、条件文と呼びます。条件文は、何らかの判断を行うときに使用します。ある条件（選択されているオブジェクトのカラー、パブリケーションのページ数、日付など）を評価して、その結果に従って処理を行うことができます。条件文は、ほとんどの場合 `if` で始まります。

注意 : 多くの条件文では、論理比較を行います。AppleScript と VBScript では、オブジェクトの比較に等号（`=`）を使用します。JavaScript では、等号は変数に値を割り当てるときに使用し、オブジェクトを比較するときには二重等号（`==`）を使用します。

制御構造

InDesign に人間の言葉を理解する能力があったら、「今から言う手順を 20 回繰り返しなさい」というような命令をしていたことでしょう。スクリプティングの用語では、このような命令のことを制御構造と呼びます。制御構造を使用すると、反復作業、つまりループ処理を行うことができます。ループの考えは、それぞれループ内で変化があったかなかったかに関わらず、特定の条件を満たすまで、ある動作を何度も何度も繰り返すことです。制御構造は通常、`repeat`（AppleScript の場合）または `for`（JavaScript と VBScript の場合）で始まります。

関数とハンドラー

関数（VBScript または JavaScript）またはハンドラー（AppleScript）とは、スクリプトの中から参照することができるスクリプティングモジュールのことです。関数やハンドラーに値（または一連の値）を渡して、返される値を取得する、というのが一般的な使い方です。よく使用するコードを関数やハンドラーにしておけば、スクリプトの中で同じコードを何度も入力せずに済みます。

AppleScript では、ハンドラーは `on` で始まります。JavaScript と VBScript では、関数は `function` で始まります。

InDesign オブジェクトモデルの理解

InDesign と InDesign ドキュメントの関係を考えるとすると、きっとプログラムと、そのコンポーネントの関係を考えるでしょう。同様に、段落はテキストフレームの中にあり、テキストフレームはページの中にあり、ページはスプレッドの中にあり、いくつかのスプレッドが集まって 1 つのドキュメントを構成していることもご存知だと思います。ドキュメントにはカラー、スタイル、レイヤー、マススプレッドなども含まれています。レイアウトを作成するユーザーは、レイアウトを構成するコンポーネントの間に一定の序列があることを直感的に理解しています。

InDesign も、ドキュメントのコンテンツを同じような考え方で「認識」しています。ドキュメントの中にはページが含まれており、ページの中にはページアイテム（テキストフレーム、長方形、楕円形など）が含まれています。テキストフレームの中には文字、単語、段落、アンカーフレームなどが含まれています。グラフィックフレームの中には画像、EPS ファイル、PDF ファイルなどが含まれています。グループの中には他のページアイテムが含まれています。これらはすべて、InDesign パブリケーションを構成するオブジェクトです。InDesign スクリプトでは、このオブジェクトに対して何らかの操作を行います。

パブリケーションの中にあるオブジェクトは、一定の序列に従って編成されています。例えば、フレームはページの中にあり、ページはドキュメントの中にあり、ドキュメントは InDesign アプリケーションオブジェクトの中にあります。オブジェクトモデルや階層という言葉は、このような構造のことを指しています。目的のオブジェクトにアクセスするためには、このオブジェクトモデルを理解している必要があります。InDesign の構造を理解することが、InDesign スクリプティングをマスターする秘訣であると言えます。

オブジェクトにはプロパティ（属性）があります。例えば、テキストオブジェクトには、テキストの書式設定に使用するフォント、ポイントサイズ、テキストに適用する行送りなどのプロパティがあります。

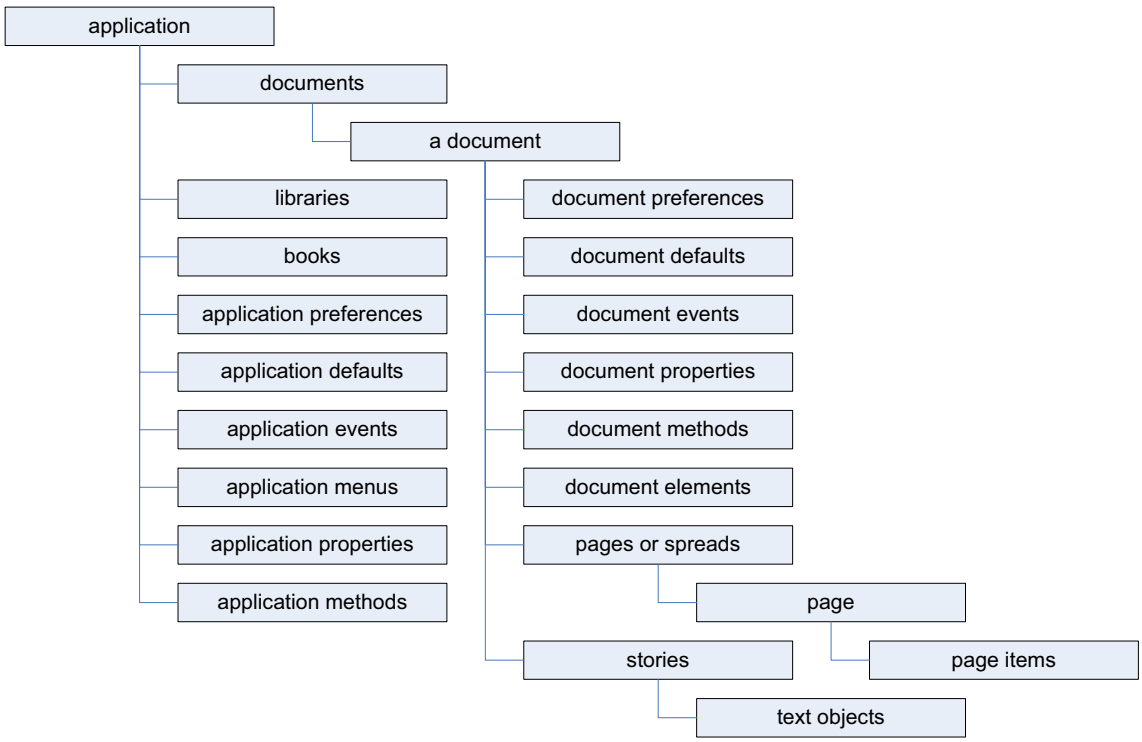
プロパティには値が設定されます。例えば、テキストのポイントサイズには、ポイント数を表す数値や、自動行送りを表す「Auto」などの文字列が設定されます。また、テキストの塗りカラーには、1 つのカラー、グラデーション、混合インキ、スウォッチなどが設定されます。

プロパティには、読み書き可能なものと読み取り専用のものがあります。読み書き可能なプロパティは、別の値に変更することができますが、読み取り専用のプロパティは変更できません。

また、オブジェクトにはメソッドが用意されています。これは、そのオブジェクトが実行できるアクションを表しており、スクリプトの中で「動詞」として使用されます。例えば、ドキュメントオブジェクトには、印刷、書き出し、保存などを行うメソッドが用意されています。

メソッドはパラメーターを取ります。パラメーターは、メソッドの動作を定義するための値です。例えば、ドキュメントの配置メソッドは、どのファイルを配置するかを指示するためのパラメーターを取ります。パラメーターには、必須のものとオプションのものがあり、メソッドによって異なります。

InDesign オブジェクトモデルの概要を次の図に示します。この図は、各種のオブジェクトの関係を理解するための概念的なフレームワークなので、InDesign スクリプティングで利用できるオブジェクトを完全に網羅したものではありません。



この図に示されている各オブジェクトについて、次の表で説明します。

用語	表しているもの
Application	InDesign。
Application defaults	アプリケーションのデフォルト設定。カラー、段落スタイル、オブジェクトスタイルなど。アプリケーションのデフォルト設定はすべての新規ドキュメントに影響を与えます。
Application events	ユーザーやスクリプトがアプリケーションを操作したときに発生します。例えば、ドキュメントのオープン、クローズ、保存を行ったときや、メニュー項目を選択したときなどにイベントが生成されます。イベントに反応してスクリプトを実行させることができます。

用語	表しているもの
Application menus	InDesign のユーザーインターフェイスに表示されるメニュー、サブメニュー、コンテキストメニュー。メニューの選択にスクリプトを関連付けたり、スクリプトからメニューアクションを実行したりできます。
Application methods	アプリケーションが実行できるアクション。テキストの検索と変更、選択項目のコピー、新規ドキュメントの作成、ライブラリのオープンなど。
Application preferences	テキストの環境設定、PDF 書き出しの環境設定、ドキュメントの環境設定など。環境設定オブジェクトの多くはドキュメントレベルにも存在しています。ユーザーインターフェイスの場合と同じように、アプリケーションの環境設定は新規ドキュメントに適用されます。ドキュメントの環境設定は特定のドキュメントの設定を変更します。
Application properties	アプリケーションのプロパティ。アプリケーションのフルパス、アプリケーションのロケール、ユーザー名など。
Books	開かれているブックのコレクション。
Document	InDesign ドキュメント。
Document defaults	ドキュメントのデフォルト設定。デフォルトのカラー、段落スタイル、デフォルトのテキスト書式など。
Document elements	ドキュメントのストーリー、読み込んだグラフィック、ページなど。ページやストーリーもドキュメント要素ですが、これらは他のオブジェクトを格納するコンテナとして特に重要な要素なので、この表の前にある図では独立して示されています。ドキュメント要素には、このほかにも、長方形、楕円形、グループ、XML 要素などの、読み込んだり作成したりできるあらゆる種類のオブジェクトが含まれます。
Document events	ドキュメントレベルで発生するイベント。テキストの読み込みなど。この表の「Application events」を参照してください。
Document methods	ドキュメントが実行できるアクション。ドキュメントのクローズ、印刷、書き出しなど。
Document preferences	ドキュメントの環境設定。ガイド、表示、ドキュメントの環境設定など。
Document properties	ドキュメントのファイル名、ページ数、原点の位置など。
Documents	開かれているドキュメントのコレクション。
Libraries	開かれているライブラリのコレクション。
Page	InDesign ドキュメント内の 1 つのページ。
Page items	ページ上に作成または配置できる任意のオブジェクト。ページアイテムには、テキストフレーム、長方形、グラフィックの線、グループなど多くの種類があります。
Pages or spreads	InDesign ドキュメント内のページまたはスプレッド。
Stories	InDesign ドキュメント内のテキスト。
Text objects	InDesign ストーリーには、文字、単語、線、段落、テキスト列などのテキストオブジェクトが含まれます。

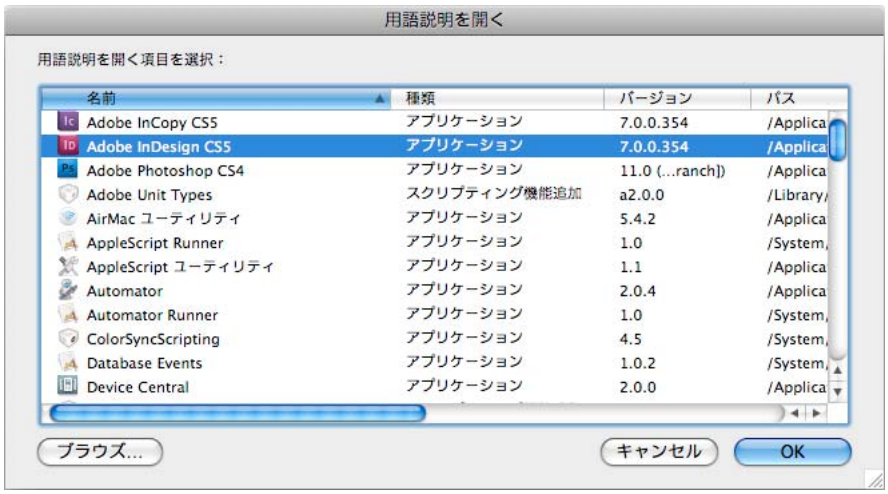
InDesign オブジェクトモデルの参照

InDesign オブジェクトモデルは、スクリプト編集アプリケーションの中で参照することができます。オブジェクトとそのプロパティやメソッドに関する参照情報は、すべてモデルに格納されており、スクリプト編集アプリケーションで参照することができます。

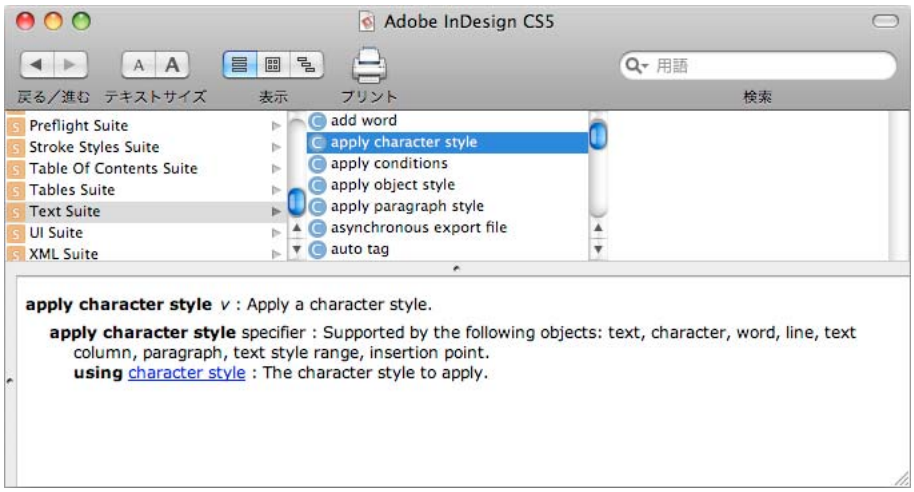
AppleScript

InDesign の AppleScript 用語説明を参照するには：

1. InDesign を起動します。
2. スクリプトエディタを起動します。
3. 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。スクリプト可能なアプリケーションのリストが表示されます。



4. InDesign アプリケーションがこの中にある場合はそれを選び「OK」をクリックします。この中にない場合は、ここでブラウズボタンをクリックすると、標準のオープンダイアログボックスが表示されるので、InDesign アプリケーションを選択して、「選択」をクリックします。InDesign のスイート（関連するオブジェクトのコレクション）のリストが表示されます。

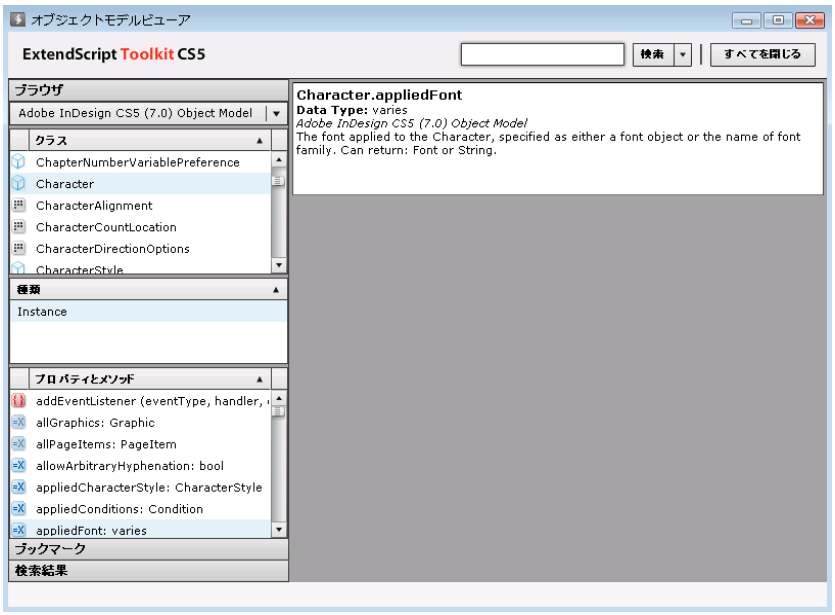


5. スイートを選択すると、そのスイートに含まれるオブジェクトとメソッド（コマンド）が表示されます。オブジェクトを選択すると、そのオブジェクトのプロパティが表示されます。

JavaScript

ExtendScript Toolkit で InDesign オブジェクトモデルを参照するには：

1. ExtendScript Toolkit を起動します。
2. ヘルプ／オブジェクトモデルビューアを選択します。
3. ブラウザーペインで「Adobe InDesign CS5 Object Model」を選択します。オブジェクトモデルのヘルプファイルが読み込まれ、InDesign スクリプトオブジェクトのリストがクラスペインに表示されます。
4. 表示するオブジェクトをクラスペインのオブジェクトのリストから選択し、詳細を表示するプロパティまたはメソッドを「プロパティとメソッド」リストでクリックします。選択したプロパティまたはメソッドの詳細が表示されます。



ExtendScript Toolkit 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、『Creative Suite 5 JavaScript Tools Guide』を参照してください。

注意 : オブジェクトモデルビューアに表示されるクラス（オブジェクト）名は、スクリプトに記述されている同一のオブジェクトのインスタンスと大文字小文字の表記が異なっています。JavaScript では大文字と小文字が区別されるので、スクリプトで用語を入力する際は、大文字と小文字を正しく使い分ける必要があります。例えば、クラスのリストには「Documents」という用語が表示されますが、スクリプトでこのクラスを参照するには「app.documents」を使用します。親オブジェクトがクラスペインで選択されると、大文字と小文字の正しい表記がオブジェクトモデルビューアのプロパティとメソッドペインに常に表示されます。

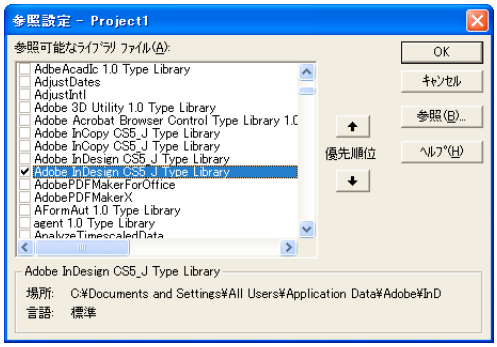
VBScript

InDesign オブジェクトモデルを表示するには、VBScript のエディターやデバッガー、Visual Basic の一部のバージョン、Visual Basic for Applications が搭載されているアプリケーションのいずれかが必要です。

Visual Basic 6

Visual Basic 6 を使用してオブジェクトモデルを参照するには：

1. Visual Basic のプロジェクトを新規作成し、プロジェクト／参照設定を選択します。参照設定ダイアログボックスが表示されます。

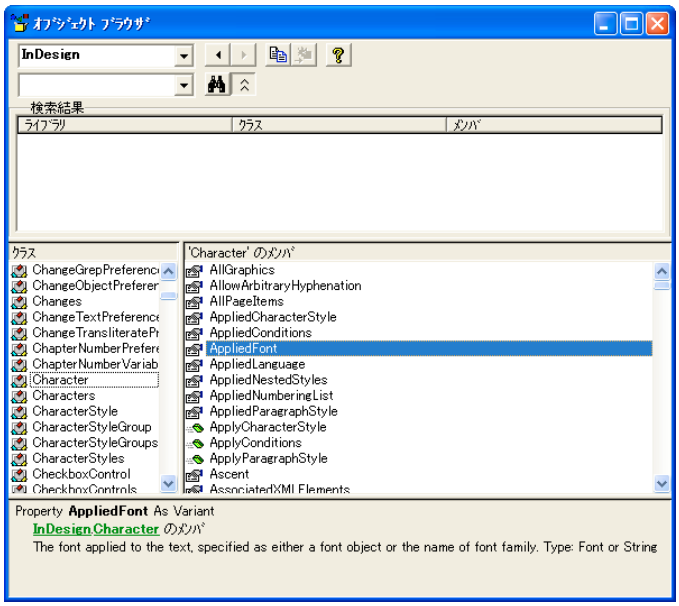


2. 利用可能な参照のリストから「Adobe InDesign CS5_J Type Library」を選択して、「OK」をクリックします。このライブラリがリストにない場合は、「参照」をクリックし、「Resources for Visual Basic.tlb」ファイルを探して選択します（このファイルは、通常は `C:¥Documents and Settings¥<ユーザー名>¥Application Data¥Adobe¥InDesign¥Version 7.0-J¥ja_JP¥Scripting Support¥7.0¥` の中にあります）。必要に応じてファイルを検索してください。ファイルが見つかったら「開く」をクリックして、プロジェクトに参照を追加します。

3. 表示／オブジェクトブラウザーを選択します。オブジェクトブラウザーダイアログボックスが表示されます。

4. プロジェクト／ライブラリメニューに表示されている、開いているライブラリのリストの中から、「InDesign」を選択します。InDesign オブジェクトモデルを構成するオブジェクトが表示されます。

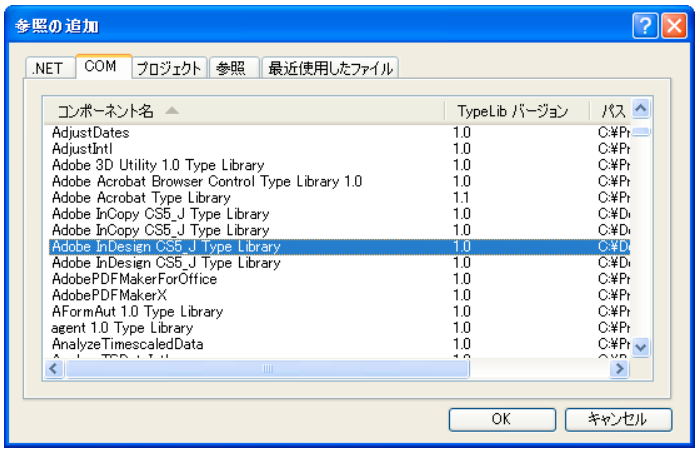
5. オブジェクトクラスをクリックします。そのオブジェクトのプロパティとメソッドが表示されます。プロパティまたはメソッドの詳細を参照するには、その項目を選択します。オブジェクトブラウザーウィンドウの一番下に、その項目の定義が表示されます。



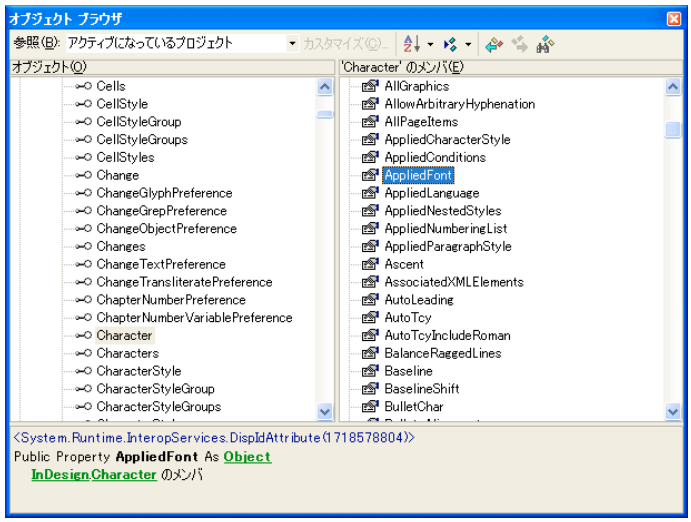
Visual Basic.NET

Visual Basic.NET を使用してオブジェクトモデルを参照するには：

1. Visual Basic のプロジェクトを新規作成し、プロジェクト／参照の追加を選択します。参照の追加ダイアログボックスが表示されます。
2. 「COM」タブを選択します。
3. 利用可能な参照のリストから「Adobe InDesign CS5_J Type Library」を選択して、「選択」をクリックします。「選択されたコンポーネント」リストに参照が追加されます。このライブラリがリストにない場合は、「参照」をクリックし、「`Resources for Visual Basic.tlb`」ファイルを探して選択します（このファイルは、通常は `C:¥Documents and Settings¥<ユーザー名>¥Application Data¥Adobe¥InDesign¥Version 7.0-J¥ja_JP¥Scripting Support¥7.0¥` の中にあります）。ファイルが見つかったら「開く」をクリックして、プロジェクトに参照を追加します。



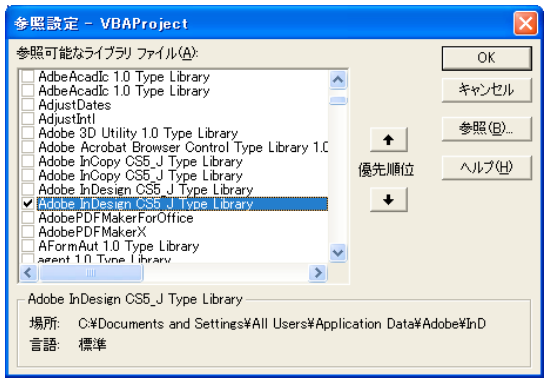
4. 「OK」をクリックします。
5. 表示／オブジェクトブラウザーを選択します。「オブジェクトブラウザー」タブが表示されます。
6. オブジェクトウィンドウで、開かれているライブラリのリストから「interop.indesign」を選択します。InDesign オブジェクトモデルを構成するオブジェクトが表示されます。
7. オブジェクトクラスをクリックします。そのオブジェクトのプロパティとメソッドが表示されます。プロパティまたはメソッドの詳細を参照するには、その項目を選択します。オブジェクトブラウザーウィンドウの一番下に、その項目の定義が表示されます。



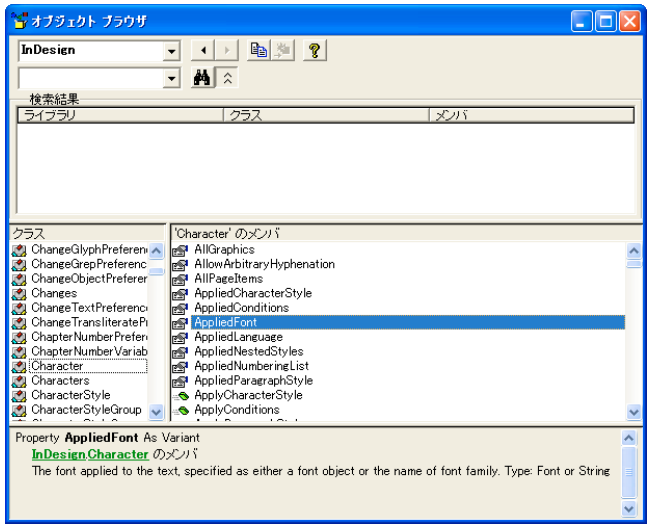
Visual Basic for Applications

Microsoft Excel から Visual Basic for Applications を使用してオブジェクトモデルを参照するには：

1. Excel を起動します。
2. ツール／マクロ／ Visual Basic Editor を選択します。Visual Basic Editor のウィンドウが表示されます。
3. ツール／参照設定を選択します。参照設定ダイアログボックスが表示されます。



4. 利用可能な参照のリストから「Adobe InDesign CS5_J Type Library」オプションを選択して、「OK」をクリックします。選択済みコンポーネントのリストに参照が追加されます。このライブラリがリストにない場合は、「参照」をクリックし、「`Resources for Visual Basic.tlb`」ファイルを探して選択します（このファイルは、通常は `C:¥Documents and Settings¥<` ユーザー名 `>¥Application Data¥Adobe¥InDesign¥Version 7.0-J¥ja_JP¥Scripting Support¥7.0¥` の中にあります）。ファイルが見つかったら「開く」をクリックして、プロジェクトに参照を追加します。
5. 表示／オブジェクトブラウザーを選択します。オブジェクトブラウザーウィンドウが表示されます。
6. ライブラリのポップアップメニューから「InDesign」を選択します。InDesign オブジェクトライブラリのオブジェクトのリストが表示されます。
7. オブジェクト名をクリックします。そのオブジェクトのプロパティとメソッドが表示されます。プロパティまたはメソッドの詳細を参照するには、その項目を選択します。オブジェクトブラウザーウィンドウの一番下に、その項目の定義が表示されます。



測定値と位置

InDesign では、どのアイテムやオブジェクトも、設定されている測定値に従ってページ上に配置されます。InDesign の座標系の仕組みと、使用される測定単位について理解しておくと、スクリプトを書く際にその知識が役に立ちます。

座標

他のページレイアウトプログラムやドロープログラムと同じように、InDesign でも、単純な 2 次元座標を使用してページやスプレッド上のオブジェクトの位置を設定します。座標は 2 つの値の組で表しますが、そのうちの水平成分を *x* と呼び、垂直成分を *y* と呼びます。選択ツールでオブジェクトを選択すると、変形パネルやコントロールパネルにこの座標が表示されます。座標は、InDesign のユーザーインターフェイスと同様に、定規の現在の原点からの相対位置で測定されます。

InDesign が採用している座標と、幾何学の教科書で使われる座標には、1 つだけ違うところがあります。それは、InDesign の座標では、垂直軸（y 軸）の原点より下が正の数であり、原点より上が負の数である、ということです。

注意 : パスポイントの位置を InDesign に問い合わせると、*x, y* の順序で座標が返されます。パスポイントの位置を設定するときも、これと同じ順序で座標を指定する必要があります。ただし、座標によってはこれとは異なる順番で値が返されることがあり、指定するときもその順番で指定する必要があります。オブジェクトの境界線や表示可能な境界線は、いずれも 4 つの座標値からなる配列です。これらはオブジェクトのバウンディングボックスの端を表し、上端、左端、下端、右端（`y1`、`x1`、`y2`、`x2`）の順番になります。

測定単位の操作

InDesign に測定値を渡すときには、数値（14.65 など）と測定文字列（「1p7.1」など）のどちらの形でも渡すことができます。数値を渡した場合は、パブリケーションの現在の測定単位が使用されます。測定文字列（次の表を参照）を渡した場合は、その文字列で指定されている測定単位が使用されます。

InDesign から返される測定値には、パブリケーションの現在の測定単位が使用されます。この InDesign から返される測定値は、変形パネルに表示される測定値と表記の仕方が違うことがあります。例えば、現在の測定単位系がパイカである場合、端数のある値はパイカとポイントの形（変形パネルでの表記法）ではなく、小数の形で返されます。例えば、「1p6」は「1.5」と返されます。このようにしている理由は、スクリプティングシステムで測定文字列を使用して算術演算を行おうとすると問題が生じるからです。例えば、「13p4」に「0p3.5」を足そうとするとスクリプトエラーが発生します。13.333 に .2916（どちらもパイカに換算した値）を足す場合にはエラーは発生しません。

スクリプトで特定の測定値の加減乗除を行う場合には、スクリプトの始めの部分で測定単位を設定します。そして、スクリプトの終わりの部分で、スクリプトを実行する前の測定単位に戻します。または、多くのサンプルスクリプトで行っているように、測定単位をオーバーライドすることができます。これを行うには、特別な文字が含まれた文字列を使用します。次に例を示します。

オーバーライド	意味	例
c	シセロ（必要に応じて c の後にディド値を追加できます）	1.4 c
cm	センチメートル	.635 cm
i（または in）	インチ	.25 i
mm	ミリメートル	6.35 mm

オーバーライド	意味	例
p	パイカ（必要に応じて p の後にポイント値を追加できます）	1p6
pt	ポイント	18 pt

「Hello World」への機能の追加

次に、最初のスクリプトで作成した「Hello World」パブリケーションに変更を加える新しいスクリプトを作成します。この 2 番目のスクリプトでは、次の処理を行います。

- アクティブなドキュメントを取得する。
- 関数（AppleScript の場合はハンドラー）を使用する。
- アクティブなドキュメントのページの大きさとページマージンを取得する。
- テキストフレームのサイズを変更する。
- テキストフレーム内のテキストのフォーマットを変更する。

AppleScript

「Hello World」ドキュメントが開いていることを確認します。このスクリプトでは、最初のスクリプトで作成したオブジェクトを使用します。ドキュメントを保存せずに閉じてしまった場合は、`HelloWorld.applescript` スクリプトをもう一度実行すればドキュメントが新たに作成されます。

`ImprovedHelloWorld.applescript` チュートリアルスクリプトを開くか、または次の手順に従ってスクリプトを作成します。

1. スクリプトエディタでファイル／新規スクリプトを選択して、新しいスクリプトを作成します。
2. 次のコードを入力します。

```
--Improved "Hello World"
tell application "Adobe InDesign CS5"
    --Get a reference to a font.
    try
        --Enter the name of a font on your system, if necessary.
        set myFont to font "Helvetica"
    end try
    --Get the active document and assign
    --the result to the variable "myDocument."
    set myDocument to document 1
    tell myDocument
        --Use the handler "myGetBounds" to get the bounds of the
        --"live area" inside the margins of page 1.
        set myBounds to my myGetBounds(myDocument, page 1)
```

```
            tell text frame 1 of page 1
                --Resize the text frame to match the page margins.
                set geometric bounds to myBounds
                tell paragraph 1
                    --Change the font, size, and paragraph alignment.
                    try
                        set applied font to myFont
                    end try
                    set point size to 72
                    set justification to center align
                end tell
            end tell
        end tell
    end tell
    --myGetBounds is a handler that returns the bounds
    --of the "live area" of a page.
    on myGetBounds(myDocument, myPage)
    tell application "Adobe InDesign CS5"
        tell document preferences of myDocument
            myPageWidth to page width
            set myPageHeight to page height
        end tell
        tell margin preferences of myPage
            if side of myPage is left hand then
                set myX2 to left
                set myX1 to right
            else
                set myX1 to left
                set myX2 to right
            end if
            set myY1 to top
            set myY2 to bottom
            end tell
            set myX2 to myPageWidth - myX2
            set myY2 to myPageHeight - myY2
            return {myY1, myX1, myY2, myX2}
        end tell
    end myGetBounds
```

3. Scripts Panel フォルダーに、プレーンテキストファイルとしてスクリプトを保存します（6 ページの「スクリプトのインストール」を参照）。ファイル拡張子は `.applescript` を使用します。

スクリプトファイルを開くか作成した後は、スクリプトエディタまたは InDesign のスクリプトパネルからスクリプトを実行できます。

JavaScript

「Hello World」ドキュメントが開いていることを確認します。このスクリプトでは、最初のスクリプトで作成したオブジェクトを使用します。ドキュメントを保存せずに閉じてしまった場合は、`HelloWorld.jsx` スクリプトをもう一度実行すればドキュメントが新たに作成されます。

`ImprovedHelloWorld.jsx` チュートリアルスクリプトを開くか、または次の手順に従ってスクリプトを作成します。

1. 新しいテキストファイルに、次の JavaScript を入力します。

```
//Improved Hello World!
//Enter the name of a font on your system, if necessary.
try{
    myFont = app.fonts.item("Arial");
}
catch (myError){};
var myDocument = app.documents.item(0);
with(myDocument){
    var myPage = pages.item(0);
    var myBounds = myGetBounds(myPage,myDocument);
    with(myDocument.pages.item(0)){
        //Get a reference to the text frame.
        var myTextFrame = textFrames.item(0);
        //Change the size of the text frame.
        myTextFrame.geometricBounds = myBounds;
        var myParagraph = myTextFrame.paragraphs.item(0);
        myParagraph.appliedFont = myFont;
        myParagraph.justification = Justification.centerAlign;
        myParagraph.pointSize = 48;
    }
}
//myGetBounds is a function that returns the bounds
//of the "live area" of a page.
function myGetBounds(myDocument, myPage){
    var myPageWidth = myDocument.documentPreferences.pageWidth;
    var myPageHeight = myDocument.documentPreferences.pageHeight
    if(myPage.side == PageSideOptions.leftHand){
        var myX2 = myPage.marginPreferences.left;
        var myX1 = myPage.marginPreferences.right;
    }
    else{
        var myX1 = myPage.marginPreferences.left;
        var myX2 = myPage.marginPreferences.right;
    }
    var myY1 = myPage.marginPreferences.top;
    var myX2 = myPageWidth - myX2;
    var myY2 = myPageHeight - myPage.marginPreferences.bottom;
    return [myY1, myX1, myY2, myX2];
}
```

2. Scripts Panel フォルダーに、プレーンテキストファイルとしてスクリプトを保存します（6 ページの「スクリプトのインストール」を参照）。ファイル拡張子は `.jsx` を使用します。

スクリプトファイルを開くか作成した後は、ExtendScript Toolkit または InDesign のスクリプトパネルからスクリプトを実行できます。

VBScript

「Hello World」ドキュメントが開いていることを確認します。このスクリプトでは、最初のスクリプトで作成したオブジェクトを使用します。ドキュメントを保存せずに閉じてしまった場合は、`HelloWorld.vbs` スクリプトをもう一度実行すればドキュメントが新たに作成されます。

ImprovedHelloWorld.vbs チュートリアルスクリプトを開くか、または次の手順に従ってスクリプトを作成します。

1. 任意のテキストエディター（メモ帳など）を起動します。
2. 次のコードを入力します。

```
Set myInDesign = CreateObject("InDesign.Application")
Rem Enter the name of a font on your system, if necessary.
Set myFont = myInDesign.Fonts.Item("Arial")
Set myDocument = myInDesign.ActiveDocument
Set myPage = myDocument.Pages.Item(1)
Rem Get page width and page height using the function "myGetBounds".
myBounds = myGetBounds(myDocument, myPage)
Set myTextFrame = myPage.TextFrames.Item(1)
Rem Resize the text frame to match the publication margins.
myTextFrame.GeometricBounds = myBounds
Set myParagraph = myTextFrame.Paragraphs.Item(1)
Rem Change the font, size, and alignment.
If TypeName(myFont) <> "Nothing" Then
    myParagraph.AppliedFont = myFont
End If
myParagraph.PointSize = 48
myParagraph.Justification = idJustification.idCenterAlign
Rem myGetBounds is a function that returns the bounds
Rem of the "live area" of a page.
Function myGetBounds(myDocument, myPage)
    myPageWidth = myDocument.DocumentPreferences.PageWidth
    myPageHeight = myDocument.DocumentPreferences.PageHeight
    If myPage.Side = idPageSideOptions.idLeftHand Then
        myX2 = myPage.MarginPreferences.Left
        myX1 = myPage.MarginPreferences.Right
    Else
        myX1 = myPage.MarginPreferences.Left
        myX2 = myPage.MarginPreferences.Right
    End If
    myY1 = myPage.marginPreferences.Top
    myX2 = myPageWidth - myX2
    myY2 = myPageHeight - myPage.MarginPreferences.Bottom
    myGetBounds =  Array(myY1, myX1, myY2, myX2)
End Function
```

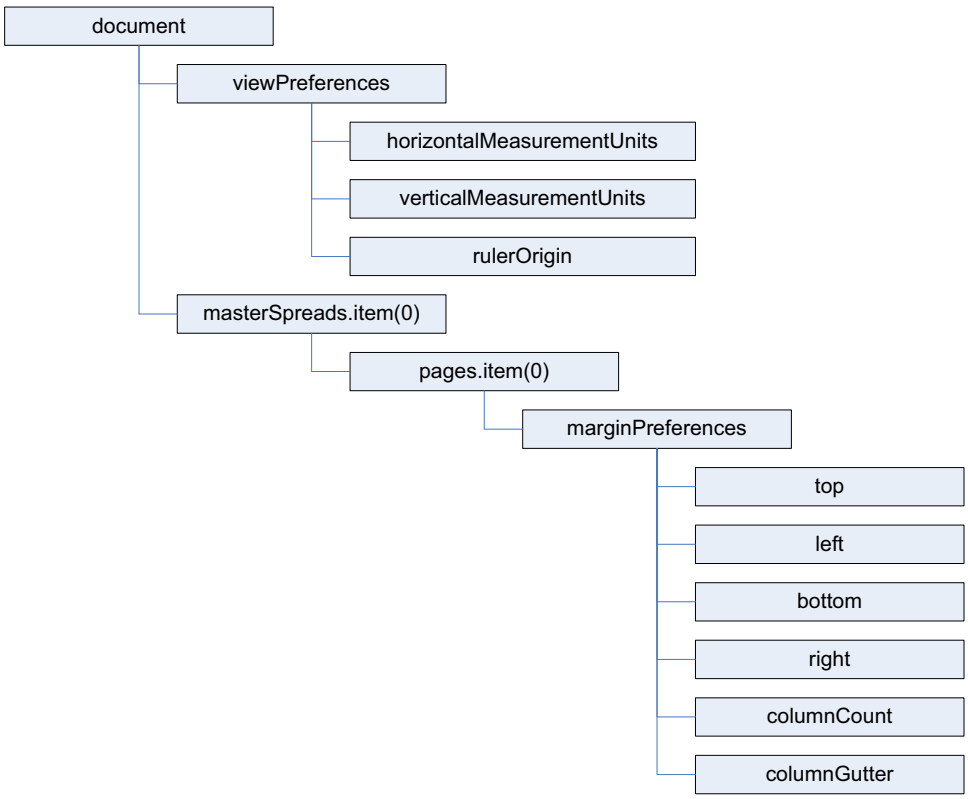
3. スクリプトパネル用のフォルダーに、プレーンテキストファイルとしてテキストを保存します（6 ページの「スクリプトのインストール」を参照）。ファイル拡張子は .vbs を使用します。

スクリプトファイルを開くか作成した後は、InDesign のスクリプトパネルからスクリプトを実行できます。

ドキュメントの構築

「Hello World!」スクリプトは、日々の作業に非常に役立つというものではありませんが、InDesign スクリプティングの基本を示しています。次の節では、もう少し複雑なスクリプトを紹介し、ユーザーが作成するスクリプトで使用されることが多いスクリプティングテクニックを説明します。

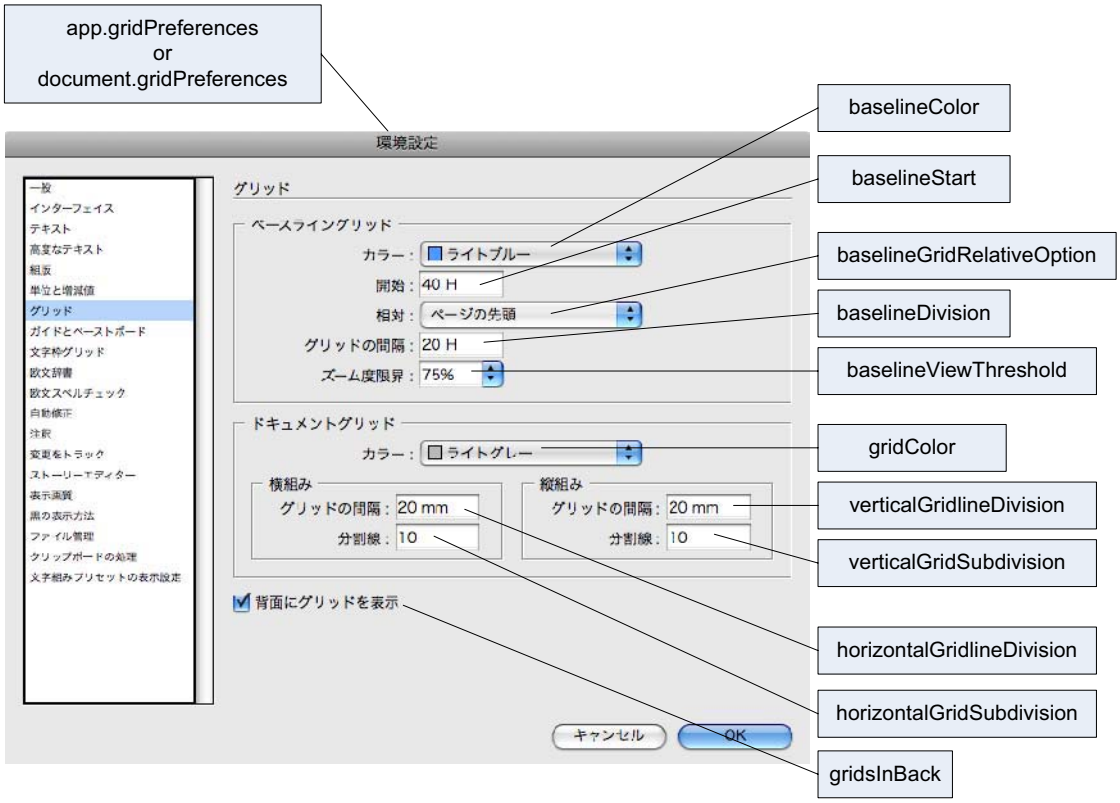
InDesign スクリプトは作業プロセスのどの段階でも活用できますが、ここではまず、作業プロセスの最初の段階を処理するスクリプトを作成してみます。つまり、新規ドキュメントを作成し、ページマージンを設定し、マスターページを定義して適用します。次の図は、これから使用するオブジェクトを表したブロック図です。

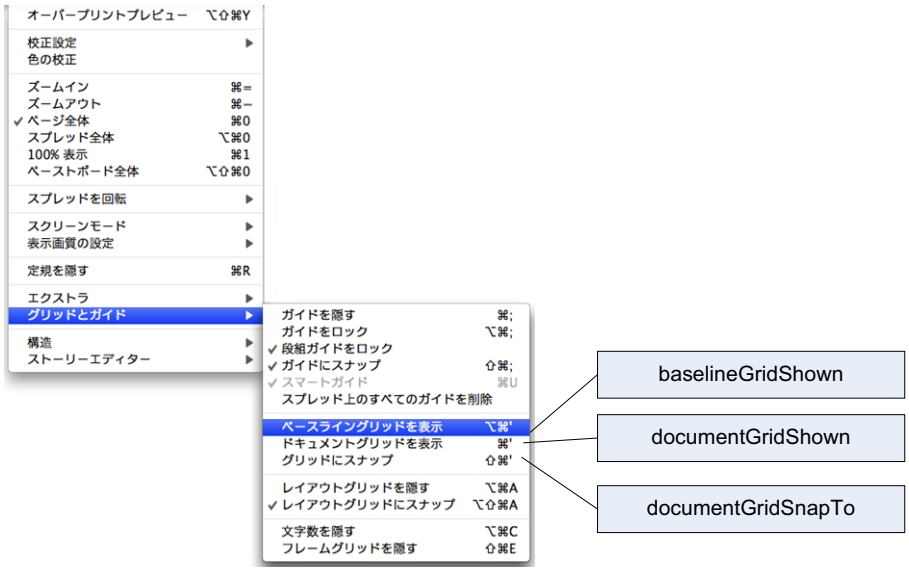


ここでは、DocumentTemplate チュートリアルスクリプトを見ていきます。スクリプトは、一連のブロックに分けられています。それぞれのブロックは、InDesign スクリプティングの特定の分野やタスクを示しています。

注意 : 前述の図では、JavaScript の表記方法が使用されています。AppleScript の場合は、単語間にスペースを追加します（例えば、「`viewPreferences`」は「`view preferences`」になります）。VBScript の場合、アイテムのインデックスは 0 ではなく 1 から始まります（例えば、`masterSpreads.item(0)` は `masterSpreads.item(1)` になります）。このようなスペースや大文字小文字表記に関するわずかな違いや、それぞれの言語におけるいくつかの予約語を除けば、各言語で使用される用語は共通しています。

このオブジェクトモデルのほとんどのオブジェクトが、ユーザーインターフェイスのコントロールの名前に対応しています（次の図を参照。この図でも、JavaScript の表記方法が使用されています）。





次の節では、`DocumentConstruction` スクリプトに記述されている各言語の機能的な部分について説明します。理解を進めたい場合は、該当する言語のスクリプトエディタでこのスクリプトを開いてください。

測定単位とマススプレッドマージンの設定

次のスクリプトでは、新規ドキュメントを作成して、最初のマススプレッドのマージンを設定します。

AppleScript

```
tell application "Adobe InDesign CS5"
    --Create a new document.
    set myDocument to make document
    --Set the measurement units and ruler origin.
    set horizontal measurement units of view preferences to points
    set vertical measurement units of view preferences to points
    set ruler origin of view preferences to page origin
    --Get a reference to the first master spread.
    set myMasterSpread to master spread 1 of myDocument
    --Get a reference to the margin preferences of the first page in the master spread.
    tell margin preferences of page 1 of myMasterSpread
        --Now set up the page margins and columns.
        set left to 84
        set top to 70
        set right to 70
        set bottom to 78
        set column count to 3
        set column gutter to 14
    end tell
    --Page margins and columns for the right-hand page.
    tell margin preferences of page 2 of myMasterSpread
        set left to 84
        set top to 70
        set right to 70
        set bottom to 78
        set column count to 3
        set column gutter to 14
    end tell
```

JavaScript

```
//Create a new document.
var myDocument = app.documents.add();
//Set the measurement units and ruler origin.
myDocument.viewPreferences.horizontalMeasurementUnits = MeasurementUnits.points;
myDocument.viewPreferences.verticalMeasurementUnits = MeasurementUnits.points;
myDocument.viewPreferences.rulerOrigin = RulerOrigin.pageOrigin;
//Get a reference to the first master spread.
var myMasterSpread = myDocument.masterSpreads.item(0);
//Get a reference to the margin preferences of the first page in the master spread.
var myMarginPreferences = myMasterSpread.pages.item(0).marginPreferences;
//Now set up the page margins and columns.
myMarginPreferences.left = 84;
myMarginPreferences.top = 70;
myMarginPreferences.right = 70;
myMarginPreferences.bottom = 78;
myMarginPreferences.columnCount = 3;
myMarginPreferences.columnGutter = 14;
```

```
//Page margins and columns for the right-hand page.
var myMarginPreferences = myMasterSpread.pages.item(1).marginPreferences;
myMarginPreferences.left = 84;
myMarginPreferences.top = 70;
myMarginPreferences.right = 70;
myMarginPreferences.bottom = 78;
myMarginPreferences.columnCount = 3;
myMarginPreferences.columnGutter = 14;
```

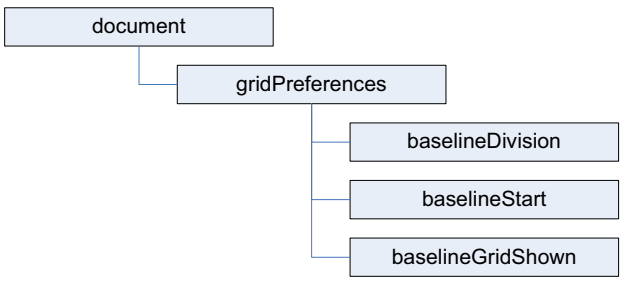
VBScript

スクリプトエディタやテキストエディターで次のコードを入力するか、`DocumentConstruction.vbs` チュートリアルスクリプトを開きます。

```
Set myInDesign = CreateObject("InDesign.Application")
Rem Create a new document.
Set myDocument = myInDesign.Documents.Add()
Rem Set the measurement units and ruler origin.
myDocument.ViewPreferences.HorizontalMeasurementUnits = idMeasurementUnits.idPoints
myDocument.ViewPreferences.VerticalMeasurementUnits = idMeasurementUnits.idPoints
myDocument.ViewPreferences.RulerOrigin = idRulerOrigin.idPageOrigin
Rem Get a reference to the first master spread.
Set myMasterSpread = myDocument.MasterSpreads.Item(1)
Rem Get a reference to the margin preferences of the first page in the master spread.
Set myMarginPreferences = myMasterSpread.Pages.Item(1).MarginPreferences
Rem Now set up the page margins and columns.
myMarginPreferences.Left = 84
myMarginPreferences.Top = 70
myMarginPreferences.Right = 70
myMarginPreferences.Bottom = 78
myMarginPreferences.ColumnCount = 3
myMarginPreferences.ColumnGutter = 14
Rem Page margins and columns for the right-hand page.
Set myMarginPreferences = myMasterSpread.Pages.Item(2).MarginPreferences
myMarginPreferences.Left = 84
myMarginPreferences.Top = 70
myMarginPreferences.Right = 70
myMarginPreferences.Bottom = 78
myMarginPreferences.ColumnCount = 3
myMarginPreferences.ColumnGutter = 14
```

ベースライングリッドの追加

マススプレッドを設定できたので、次にベースライングリッドを追加します。次の図は、これから使用するオブジェクト間の関係を示すブロック図です（表記方法は JavaScript のものです）。



AppleScript

```
set myGridPreferences to grid preferences
set baseline division of myGridPreferences to 14
set baseline start of myGridPreferences to 70
set baseline grid shown of myGridPreferences to true
```

JavaScript

```
var myGridPreferences = myDocument.gridPreferences;
myGridPreferences.baselineDivision = 14;
myGridPreferences.baselineStart = 70;
myGridPreferences.baselineGridShown = true;
```

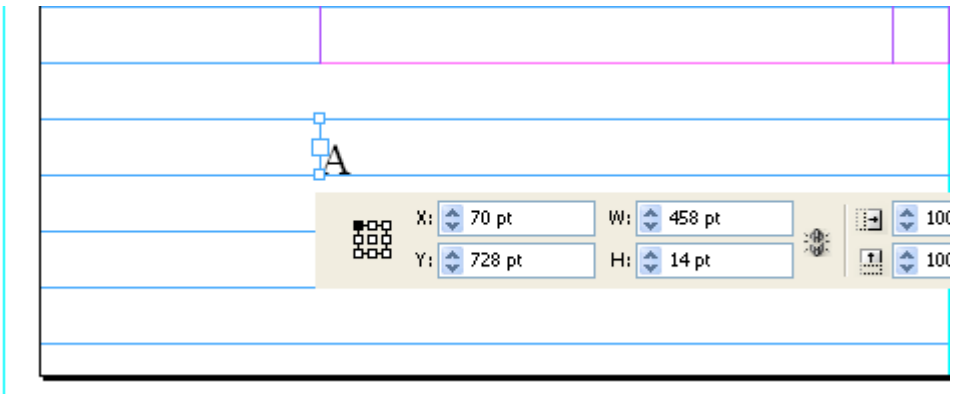
VBScript

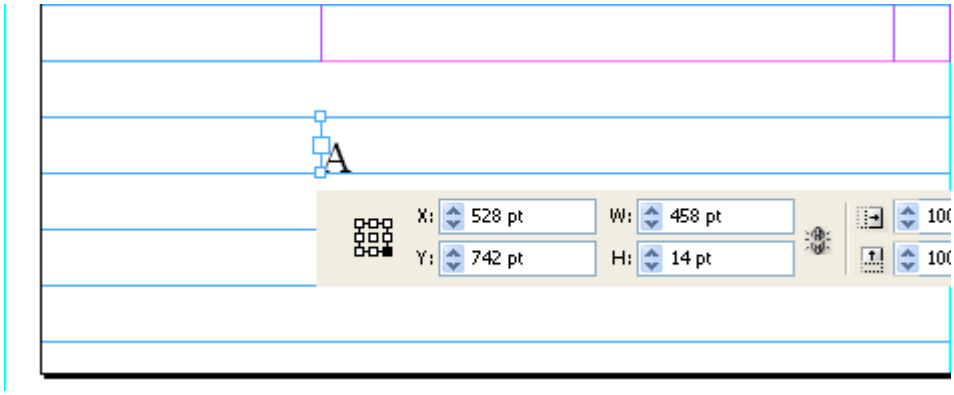
```
Set myGridPreferences = myDocument.GridPreferences
myGridPreferences.BaselineDivision = 14
myGridPreferences.BaselineStart = 70
myGridPreferences.BaselineGridShown = True
```

マスターページアイテムの追加

次に、マスターページに 2 つのテキストフレームを追加します。これらのフレームは、自動ページ番号を表す特殊文字を挿入して、ページの末尾に配置します。

「Hello World」の例では、1 つのテキストフレームを作成し、その境界線のプロパティ（フレームの上端、左端、下端、右端の座標を表す配列）を使用してフレームの位置とサイズを指定しました。これらの座標は、コントロールパネルに表示されるフレームの四隅の座標に対応しています。この例では、次の 2 つの図に示すように、上端 = 728、左端 = 70、下端 = 742、右端 = 528 の境界線を使用します。





AppleScript

```
set myLeftPage to page 1 of myMasterSpread
set myRightPage to page 2 of myMasterSpread
tell myLeftPage
    set myLeftFooter to make text frame
    set geometric bounds of myLeftFooter to {728, 70, 742, 528}
    set first baseline offset of text frame preferences of myLeftFooter to leading offset
    set contents of myLeftFooter to auto page number
    set point size of character 1 of parent story of myLeftFooter to 11
    set leading of character 1 of myLeftFooter to 14
end tell
tell myRightPage
    set myRightFooter to make text frame
    set geometric bounds of myRightFooter to {728, 84, 742, 542}
    set first baseline offset of text frame preferences of myRightFooter to leading offset
    set contents of myRightFooter to auto page number
    set point size of character 1 of parent story of myRightFooter to 11
    set leading of character 1 of myRightFooter to 14
    set justification of character 1 of myRightFooter to right align
end tell
```

JavaScript

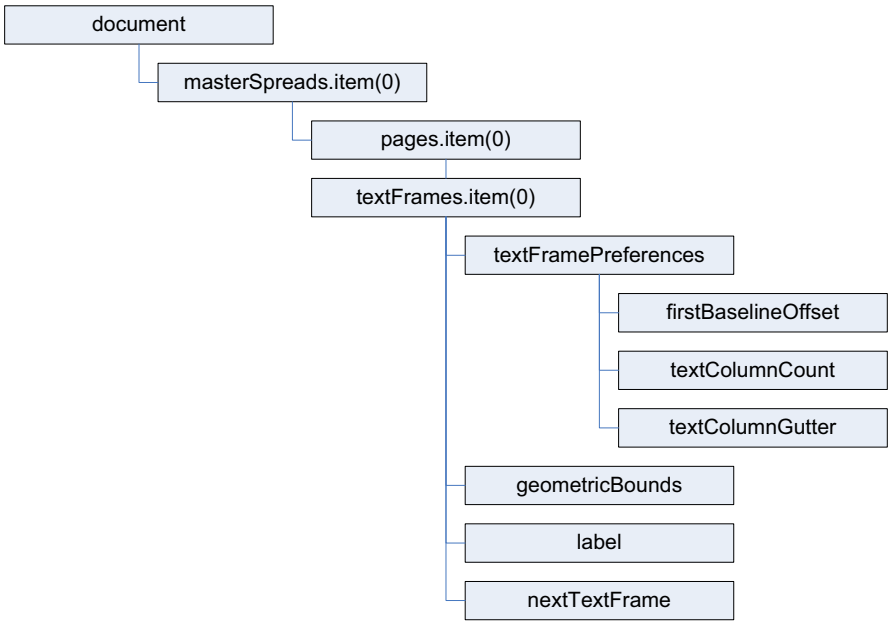
```
var myMasterSpread = myDocument.masterSpreads.item(0);
var myLeftPage = myMasterSpread.pages.item(0);
var myRightPage = myMasterSpread.pages.item(1);
var myLeftFooter = myLeftPage.textFrames.add();
myLeftFooter.geometricBounds = [728, 70, 742, 528];
myLeftFooter.textFramePreferences.firstBaselineOffset = FirstBaseline.leadingOffset;
myLeftFooter.contents = SpecialCharacters.autoPageNumber;
myLeftFooter.parentStory.characters.item(0).pointSize = 11;
myLeftFooter.parentStory.characters.item(0).leading = 14;
var myRightFooter = myRightPage.textFrames.add();
myRightFooter.geometricBounds = [728, 84, 742, 542];
myRightFooter.textFramePreferences.firstBaselineOffset = FirstBaseline.leadingOffset;
myRightFooter.contents = SpecialCharacters.autoPageNumber;
myRightFooter.parentStory.characters.item(0).pointSize = 11;
myRightFooter.parentStory.characters.item(0).leading = 14;
myRightFooter.parentStory.characters.item(0).justification = Justification.rightAlign;
```

VBScript

```
Set myMasterSpread = myDocument.MasterSpreads.Item(1)
Set myLeftPage = myMasterSpread.Pages.Item(1)
Set myRightPage = myMasterSpread.Pages.Item(2)
Set myLeftFooter = myLeftPage.TextFrames.Add
myLeftFooter.GeometricBounds = Array(728, 70, 742, 528)
myLeftFooter.TextFramePreferences.FirstBaselineOffset = idFirstBaseline.idLeadingOffset
myLeftFooter.Contents = idSpecialCharacters.idAutoPageNumber
myLeftFooter.ParentStory.Characters.Item(1).PointSize = 11
myLeftFooter.ParentStory.Characters.Item(1).Leading = 14
Set myRightFooter = myRightPage.TextFrames.Add()
myRightFooter.GeometricBounds = Array(728, 84, 742, 542)
myRightFooter.TextFramePreferences.FirstBaselineOffset = idFirstBaseline.idLeadingOffset
myRightFooter.Contents = idSpecialCharacters.idAutoPageNumber
myRightFooter.ParentStory.Characters.Item(1).PointSize = 11
myRightFooter.ParentStory.Characters.Item(1).Leading = 14
myRightFooter.ParentStory.Characters.Item(1).Justification = idJustification.idRightAlign
```

マスターテキストフレームの追加

次に、マスターテキストフレームを追加します。次のブロック図に、これから使用するオブジェクトとプロパティを示します（表記方法は JavaScript のものです）。



AppleScript

```
tell myLeftPage
    set myLeftTextFrame to make text frame
    set geometric bounds of myLeftTextFrame to {70, 70, 714, 528}
    set first baseline offset of text frame preferences of myLeftTextFrame to leading offset
    set text column count of text frame preferences of myLeftTextFrame to 3
    set text column gutter of text frame preferences of myLeftTextFrame to 14
    --Add a label to make the frame easier to find later on.
    set label of myLeftTextFrame to "BodyTextFrame"
end tell
tell myRightPage
    set myRightTextFrame to make text frame
    set geometric bounds of myRightTextFrame to {70, 84, 714, 542}
    set first baseline offset of text frame preferences of myRightTextFrame to leading offset
    set text column count of text frame preferences of myRightTextFrame to 3
    set text column gutter of text frame preferences of myRightTextFrame to 14
    --Add a label to make the frame easier to find later on.
    set label of myRightTextFrame to "BodyTextFrame"
end tell
--Link the two frames using the next text frame property.
set next text frame of myLeftTextFrame to myRightTextFrame
```

JavaScript

```
var myLeftPage = myMasterSpread.pages.item(0);
var myRightPage = myMasterSpread.pages.item(1);
var myLeftTextFrame = myLeftPage.textFrames.add();
myLeftTextFrame.geometricBounds = [70, 70, 714, 528];
myLeftTextFrame.textFramePreferences.firstBaselineOffset = FirstBaseline.leadingOffset;
myLeftTextFrame.textFramePreferences.textColumnCount = 3;
myLeftTextFrame.textFramePreferences.textColumnGutter = 14;
//Add a label to make the frame easier to find later on.
myLeftTextFrame.label = "BodyTextFrame";
var myRightTextFrame = myRightPage.textFrames.add();
myRightTextFrame.geometricBounds = [70, 84, 714, 542];
myRightTextFrame.textFramePreferences.firstBaselineOffset = FirstBaseline.leadingOffset;
myRightTextFrame.textFramePreferences.textColumnCount = 3;
myRightTextFrame.textFramePreferences.textColumnGutter = 14;
//Add a label to make the frame easier to find later on.
myRightTextFrame.label = "BodyTextFrame";
//Link the two frames using the nextTextFrame property.
myLeftTextFrame.nextTextFrame = myRightTextFrame;
```

VBScript

```
Set myLeftTextFrame = myLeftPage.TextFrames.Add
myLeftTextFrame.GeometricBounds = Array(70, 70, 714, 528)
myLeftTextFrame.TextFramePreferences.FirstBaselineOffset = idFirstBaseline.idLeadingOffset
myLeftTextFrame.TextFramePreferences.TextColumnCount = 3
myLeftTextFrame.TextFramePreferences.TextColumnGutter = 14
```

```
Rem Add a label to make the frame easier to find later on.
myLeftTextFrame.Label = "BodyTextFrame"
Set myRightTextFrame = myRightPage.TextFrames.Add
myRightTextFrame.GeometricBounds = Array(70, 84, 714, 542)
myRightTextFrame.TextFramePreferences.FirstBaselineOffset = idFirstBaseline.idLeadingOffset
myRightTextFrame.TextFramePreferences.TextColumnCount = 3
myRightTextFrame.TextFramePreferences.TextColumnGutter = 14
Rem Add a label to make the frame easier to find later on.
myRightTextFrame.Label = "BodyTextFrame"
Rem Link the two frames using the nextTextFrame property.
myLeftTextFrame.NextTextFrame = myRightTextFrame
```

マスターページアイテムのオーバーライドとテキストの追加

次に、作成したマスターテキストフレームのうちの 1 つをオーバーライドし、そのフレームにテキストを追加します。

AppleScript

```
tell text frame 1 of page 2 of master spread 1 of myDocument
    set myTextFrame to override destination page page 1 of myDocument
end tell
--Add text by setting the contents of an insertion point to a string.
--In AppleScript, "return" is a return character.
set contents of insertion point 1 of myTextFrame to "Headline!" & return
```

JavaScript

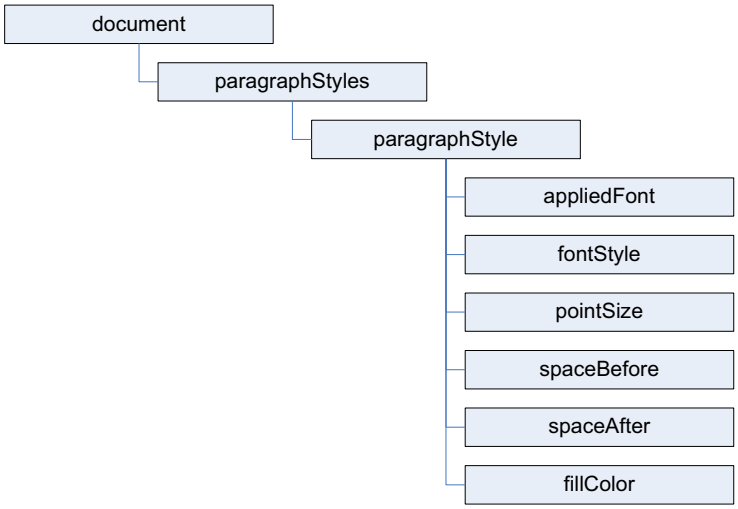
```
var myTextFrame =
myDocument.masterSpreads.item(0).pages.item(1).textFrames.item(0).override(myDocument.pages.item
(0));
//Add text by setting the contents of an insertion point to a string.
//In JavaScript, "¥r" is a return character.
myTextFrame.insertionPoints.item(0).contents = "Headline!¥r";
```

VBScript

```
Set myTextFrame =
myDocument.MasterSpreads.Item(1).Pages.Item(2).TextFrames.Item(1).Override(myDocument.Pages.Item
(1))
Rem Add text by setting the contents of an insertion point to a string.
Rem In VBScript, vbCr is a return character.
myTextFrame.InsertionPoints.Item(1).Contents = "Headline!" & vbCr
```

段落スタイルの追加と適用

見出しが目立つように、段落スタイルを適用します。そのためには、段落スタイルを作成する必要があります。次の図に、使用するオブジェクトとプロパティを示します（これまでと同様に、表記方法は JavaScript のものです）。



AppleScript

```
--First, check to see if the paragraph style already exists.
try
    set myParagraphStyle to paragraph style "Heading 1" of myDocument
on error
    --The paragraph style did not exist, so create it.
    tell myDocument
        set myParagraphStyle to make paragraph style with properties {name:"Heading 1"}
    end tell
end try
--We'll need to create a color. Check to see if the color already exists.
try
    set myColor to color "Red" of myDocument
on error
    --The color did not exist, so create it.
    tell myDocument
        set myColor to make color with properties {name:"Red", model:process,
        color value:{0, 100, 100, 0}}
    end tell
```

```
end try
--Now set the formatting of the paragraph style.
try
    set applied font of myParagraphStyle to "Arial"
    set font style of myParagraphStyle to "Bold"
end try
set point size of myParagraphStyle to 24
set space after of myParagraphStyle to 24
set space before of myParagraphStyle to 24
set fill color of myParagraphStyle to color "Red" of myDocument
--Apply the style to the paragraph.
tell paragraph 1 of myTextFrame to apply paragraph style using myParagraphStyle with clearing
overrides
--You could also use:
--set applied paragraph style of paragraph 1 of myTextFrame to myParagraphStyle
```

JavaScript

```
var myParagraphStyle = myDocument.paragraphStyles.item("Heading 1");
try {
    var myName = myParagraphStyle.name;
}
catch (myError){
    //The paragraph style did not exist, so create it.
    myParagraphStyle = myDocument.paragraphStyles.add({name:"Heading 1"});
}
//We'll need to create a color. Check to see if the color already exists.
var myColor = myDocument.colors.item("Red");
try {
    myName = myColor.name;
}
catch (myError){
    //The color did not exist, so create it.
    myColor = myDocument.colors.add({name:"Red", model:ColorModel.process,
    colorValue:[0,100,100,0]});
}

//Now set the formatting of the paragraph style.
myParagraphStyle.appliedFont = "Arial";
myParagraphStyle.fontStyle = "Bold";
myParagraphStyle.pointSize = 24;
myParagraphStyle.spaceAfter = 24;
myParagraphStyle.spaceBefore = 24;
myParagraphStyle.fillColor = myDocument.colors.item("Red");
//Apply the style to the paragraph.
myDocument.pages.item(0).textFrames.item(0).paragraphs.item(0).applyParagraphStyle(
myParagraphStyle, true);
//You could also use:
//myDocument.pages.item(0).textFrames.item(0).paragraphs.item(0).appliedParagraphStyle =
myParagraphStyle;
```

VBScript

```
Rem First, check to see if the paragraph style already exists.
Rem to do this, we disable error checking:
On Error Resume Next
Set myParagraphStyle = myDocument.ParagraphStyles.Item("Heading 1")
Rem if an error occurred on the previous line, then the paragraph
Rem style did not exist.
If Error.Number <> 0 Then
    Set myParagraphStyle = myDocument.ParagraphStyles.Add
    myParagraphStyle.Name = "Heading 1"
    Error.Clear
End If
Rem We'll need to create a color. Check to see if the color already exists.
Set myColor = myDocument.Colors.Item("Red")

If Error.Number <> 0 Then
    Set myColor = myDocument.Colors.Add
    myColor.Name = "Red"
    myColor.Model = idColorModel.idProcess
    myColor.colorValue = Array(0, 100, 100, 0)
    Error.Clear
End If
Rem Resume normal error handling.
On Error GoTo 0
Rem Now set the formatting of the paragraph style.
myParagraphStyle.AppliedFont = "Arial"
myParagraphStyle.FontStyle = "Bold"
myParagraphStyle.PointSize = 24
myParagraphStyle.SpaceAfter = 24
myParagraphStyle.SpaceBefore = 24
myParagraphStyle.FillColor = myDocument.Colors.Item("Red")
Rem Apply the style to the paragraph.
myDocument.Pages.Item(1).TextFrames.Item(1).Paragraphs.Item(1).ApplyParagraphStyle
myParagraphStyle, True
Rem You could also use:
Rem myDocument.pages.item(1).textFrames.item(1).paragraphs.item(1).appliedParagraphStyle =
myParagraphStyle
```

テキストファイルの配置

次に、テキストファイルを読み込みます。そのテキストは、1 ページ目の最初のテキストフレームの見出しの後に追加します。このスクリプトでは、読み込むテキストファイルを選択するためのダイアログボックスを表示します。

AppleScript

```
--Display a standard open file dialog box to select a text file.
set myTextFile to choose file ("Choose a text file")
--If a text file was selected, and if you didn't press Cancel,
--place the text file at the first insertion point after the headline.
if myTextFile is not "" then
    tell insertion point -1 of myTextFrame to place myTextFile
end if
```

JavaScript

```
//Display a standard open file dialog box to select a text file.
var myTextFile = File.openDialog("Choose a text file");
//If a text file was selected, and if you didn't press Cancel,
//place the text file at the first insertion point after the headline.
if((myTextFile != "")&&(myTextFile != null)){
    myTextFrame.insertionPoints.item(-1).place(myTextFile);
}
```

VBScript

```
Rem Display a standard open file dialog box to select a text file.
Rem VBScript does not have the ability to do this, so we'll use
Rem a JavaScript to get a file name. We'll run the JavaScript using
Rem InDesign's DoScript feature.
Rem Disable normal error handling.
On Error Resume Next
Rem Create a JavaScript as a string.
myJavaScriptString = "var myTextFile = File.openDialog(""Choose a text
file"");myTextFile.fsName;"
Rem Run the JavaScript using DoScript.
myFileName = myInDesign.DoScript(myJavaScriptString, idScriptLanguage.idJavascript)
If Error.Number = 0 Then
    Rem Place the text file at the end of the text frame.
    myTextFrame.InsertionPoints.Item(-1).Place myFileName
    Error.Clear
End If
Rem Restore normal error handling.
On Error GoTo 0
```

グラフィックの配置

グラフィックの配置は、テキストファイルの読み込みに似ています。このスクリプトでも、配置するグラフィックを選択するためのダイアログボックスを表示します。グラフィックを配置すると、グラフィックを配置したフレームではなく、グラフィック自体への参照が InDesign によって返されます。フレームへの参照を取得するには、グラフィックの `parent` プロパティを使用します。フレームへの参照を取得したら、オブジェクトのスタイルをフレームに適用できます。

AppleScript

```
--Display a standard open file dialog box to select a graphic file.
set myGraphicFile to choose file "Choose graphic file."
--If a graphic file was selected, and if you didn't press Cancel,
--place the graphic file on the page.
if myGraphicFile is not "" then
    set myGraphic to place myGraphicFile on page 1 of myDocument
    --Since you can place multiple graphics at once, the place method
    --returns an array. To get the graphic you placed, get the first
    --item in the array.
    set myGraphic to item 1 of myGraphic
    --Create an object style to apply to the graphic frame.
    try
        set myObjectStyle to object style "GraphicFrame" of myDocument on error
        --The object style did not exist, so create it.
        tell myDocument
            set myObjectStyle to make object style with properties{name:"GraphicFrame"}
        end tell
    end try
    set enable stroke of myObjectStyle to true
    set stroke weight of myObjectStyle to 3
    set stroke type of myObjectStyle to stroke style "Solid" of myDocument
    set stroke color of myObjectStyle to color "Red" of myDocument
    --The frame containing the graphic is the parent of the graphic.
    set myFrame to parent of myGraphic
    tell myFrame to apply object style using myObjectStyle

    --Resize the frame to a specific size.
    set geometric bounds of myFrame to {0, 0, 144, 144}
    --Fit the graphic to the frame proportionally.
    fit myFrame given proportionally
    --Next, fit frame to the resized graphic.
    fit myFrame given frame to content
    set myBounds to geometric bounds of myFrame
    set myGraphicWidth to (item 4 of myBounds) - (item 2 of myBounds)
    --Move the graphic frame.
    set myPageWidth to page width of document preferences of myDocument
    set myMarginPreferences to margin preferences of page 1 of myDocument
    set myTopMargin to top of myMarginPreferences
    move myFrame to {myPageWidth - myGraphicWidth, myTopMargin}
    --Apply a text wrap to the graphic frame.
    set text wrap mode of text wrap preferences of myFrame to bounding box text wrap
    set text wrap offset of text wrap preferences of myFrame to {24, 12, 24, 12}
    end if
end tell
```

JavaScript

```
//Display a standard open file dialog box to select a graphic file.
var myGraphicFile = File.openDialog("Choose a graphic file");
//If a graphic file was selected, and if you didn't press Cancel,
//place the graphic file on the page.
if((myGraphicFile != "")&&(myGraphicFile != null)){
    var myGraphic = myDocument.pages.item(0).place(myGraphicFile);
    //Since you can place multiple graphics at once, the place method
    //returns an array. To get the graphic you placed, get the first
    //item in the array (JavaScript arrays start with item 0).
    myGraphic = myGraphic[0];
    //Create an object style to apply to the graphic frame.
    var myObjectStyle = myDocument.objectStyles.item("GraphicFrame");
    try {
        var myName = myObjectStyle.name;
    }
    catch (myError){
        //The object style did not exist, so create it.
        myObjectStyle = myDocument.objectStyles.add({name:"GraphicFrame"});
    }
    myObjectStyle.enableStroke = true;
    myObjectStyle.strokeWeight = 3;
    myObjectStyle.strokeType = myDocument.strokeStyles.item("Solid");
    myObjectStyle.strokeColor = myDocument.colors.item("Red");
    //The frame containing the graphic is the parent of the graphic.
    var myFrame = myGraphic.parent;
    myFrame.applyObjectStyle(myObjectStyle, true);
    //Resize the frame to a specific size.
    myFrame.geometricBounds = [0,0,144,144];
    //Fit the graphic to the frame proportionally.
    myFrame.fit(FitOptions.proportionally);
    //Next, fit frame to the resized graphic.
    myFrame.fit(FitOptions.frameToContent);
    var myBounds = myFrame.geometricBounds;
    var myGraphicWidth = myBounds[3]-myBounds[1];

    //Move the graphic frame.
    var myPageWidth = myDocument.documentPreferences.pageWidth;
    var myTopMargin = myDocument.pages.item(0).marginPreferences.top;
    myFrame.move([myPageWidth-myGraphicWidth, myTopMargin]);
    //Apply a text wrap to the graphic frame.
    myFrame.textWrapPreferences.textWrapMode = TextWrapModes.BOUNDING_BOX_TEXT_WRAP;
    myFrame.textWrapPreferences.textWrapOffset = [24, 12, 24, 12];
}
```

VBScript

```
Rem create an object style
On Error Resume Next
Set myObjectStyle = myDocument.ObjectStyles.Item("GraphicFrame")
If Error.Number <> 0 Then
    Set myObjectStyle = myDocument.ObjectStyles.Add
    myObjectStyle.Name = "GraphicFrame"
    Error.Clear
End If
On Error GoTo 0
myObjectStyle.EnableStroke = True
myObjectStyle.StrokeWeight = 3
myObjectStyle.StrokeType = myDocument.StrokeStyles.Item("Solid")
myObjectStyle.StrokeColor = myDocument.Colors.Item("Red")
Rem Again, we'll use a JavaScript to get a file name.
Rem Disable normal error handling.
On Error Resume Next
Rem Create a JavaScript as a string.
myJavaScriptString = "var myTextFile = File.openDialog(""Choose a graphic
file"");myTextFile.fsName;"
Rem Run the JavaScript using DoScript.
myGraphicFileName = myInDesign.DoScript(myJavaScriptString, idScriptLanguage.idJavascript)
If Error.Number = 0 Then
    On Error GoTo 0
    Set myGraphic = myDocument.Pages.Item(1).Place(myGraphicFileName)
    Rem Since you can place multiple graphics at once, the place method
    Rem returns an object collection. To get the graphic you placed, get the first
    Rem item in the collection.
    Set myGraphic = myGraphic.Item(1)
    Rem Create an object style to apply to the graphic frame.
    Rem The frame containing the graphic is the parent of the graphic.
    Set myFrame = myGraphic.Parent
    myFrame.ApplyObjectStyle myObjectStyle, True
    Rem Resize the frame to a specific size.
    myFrame.GeometricBounds = Array(0, 0, 144, 144)
    Rem Fit the graphic to the frame proportionally.
    myFrame.Fit idFitOptions.idProportionally
    Rem Next, fit frame to the resized graphic.
    myFrame.Fit idFitOptions.idFrameToContent
    myBounds = myFrame.GeometricBounds
    myGraphicWidth = myBounds(3) - myBounds(1)
    Rem Move the graphic frame.
    myPageWidth = myDocument.DocumentPreferences.PageWidth
    myTopMargin = myDocument.Pages.Item(1).MarginPreferences.Top
    myFrame.Move Array(myPageWidth - myGraphicWidth, myTopMargin)
    Rem Apply a text wrap to the graphic frame.
    myFrame.TextWrapPreferences.TextWrapMode =
    idTextWrapModes.idBoundingBoxTextWrap
    myFrame.TextWrapPreferences.TextWrapOffset = Array(24, 12, 24, 12)
End If
```

基礎から応用へ

これまでの説明で、ドキュメントの作成、マスターページアイテムの設定、テキストの入力や読み込み、段落スタイルやオブジェクトスタイルの作成と適用、グラフィックの読み込み、フレームに合わせたグラフィックのサイズ調整、テキストの回り込みなどを行う方法を学びました。この例で作成したドキュメントは、デザインコンテストで入賞できるようなものではありませんが、InDesign スクリプティングの基礎を習得するには十分です。それぞれの例では、オブジェクトを作成し、オブジェクトのプロパティを設定し、オブジェクトのメソッドを使用しました。

InDesign スクリプティングについてさらに深く学習したい方は、『Adobe InDesign CS5 スクリプティングガイド』を参照してください。このマニュアルには、ドキュメントの作成、テキストのフォーマット、テキストの検索と変更、ユーザーインターフェイスの構築、メニュー項目の追加、XML や XML ルールの使用などに関する高度なチュートリアルが用意されています。

また、InDesign スクリプティングに関するユーザーフォーラム（http://www.adobeforums.com）でも、InDesign スクリプティングに関する情報が入手できます。このフォーラムでは、質問をしたり、質問に答えたり、新たに作成したスクリプトを別のユーザーと共有したりできます。このフォーラムでは、多数のサンプルスクリプトが提供されています。

InDesign スクリプティングのホームページ（http://www.adobe.com/jp/products/indesign/scripting/index.html）でも、InDesign スクリプティングに関する詳しい情報が入手できます。