ACTIONSCRIPT™ 2.0 の学習

Adobe®

© 2007 Adobe Systems Incorporated. All rights reserved.

ActionScript 2.0 の学習

本マニュアルがエンドユーザー使用許諾契約を含むソフトウェアと共に提供される場合、本マニュアルおよびその中に記載されているソフトウェアは、エンドユーザー使用許諾契約にもとづいて提供されるものであり、当該エンドユーザー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。当該エンドユーザー使用許諾契約により許可されている場合を除き、本マニュアルのいかなる部分といえども、Adobe Systems Incorporated (アドビシステムズ社) の書面による事前の許可なしに、電子的、機械的、録音、その他のいかなる形式・手段であれ、複製、検索システムへの保存、または伝送を行うことはできません。本マニュアルの内容は、エンドユーザー使用許諾契約を含むソフトウェアと共に提供されていない場合であっても、著作権法により保護されていることにご留意ください。

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。

新しいアートワークを創作するためにテンプレートとして取り込もうとする既存のアートワークまたは画像は、著作憲法により保護されている可能性のあるものであることをご留意ください。保護されているアートワークまたは画像を新しいアートワークに許可なく取り込んだ場合、著作権所有者の権利を侵害することがあります。従って、著作権者から必要なすべての許可を必ず取得してください。

例として使用されている会社名は、実在の会社・組織を示すものではありません。

Adobe®、Flash®、FlashHelp®、Flash® Player、JRun™、Macromedia®、および Shockwave® 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。

Macintosh® is a trademark of Apple Computer, Inc., registered in the United States and other countries. Windows® is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries. All other trademarks are the property of their respective owners.

Portions of this product contain code licensed from Nellymoser. (www.nellymoser.com).

Sorenson Spark™ video compression and decompression technology licensed from Sorenson Media, Inc.

Flash CS3 video is powered by On2 TrueMotion video technology. © 1992-2005 On2 Technologies, Inc. All Rights Reserved. http://www.on2.com.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA. Notice to U.S. Government End Users. The Software and Documentation are “Commercial Items,” as that term is defined at 48 C.F.R. §2.101, consisting of “Commercial Computer Software” and “Commercial Computer Software Documentation,” as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

目 次

はじめに 9
対象となる読者 9
システム要件 10
Flash XML ファイルの更新 10
本マニュアルについて 10
他の情報源 14

第 1 章 : Flash で ActionScript に追加された新機能 17
ActionScript 2.0 と Flash Player 9.x の新機能 17
ActionScript 2.0 と Flash Player 8 の新機能 18
ローカルでインストールされた SWF ファイルのセキュリティモデルの変更 26

第 2 章 : ActionScript について 29
ActionScript とは 30
ActionScript 1.0 または ActionScript 2.0 の選択について 30
ActionScript と Flash Player について 31

第 3 章 : データおよびデータ型 33
データについて 33
データ型について 34
変数について 47
オブジェクト内のデータの整理 69
キャストについて 71

第 4 章 : シンタックスと言語の基礎 75
シンタックス、ステートメント、および式について 76
ドットシンタックスとターゲットパスについて 79
言語区切子について 87
定数およびキーワードについて 98
ステートメントについて 102
配列について 123
演算子について 135

第 5 章 : 関数とメソッド 157
関数とメソッドについて 157
メソッドについて 176

第 6 章 : クラス 179
オブジェクト指向プログラミングと Flash について 180
カスタムクラスファイルの記述 188
アプリケーションでのカスタムクラスの使用について 191
例 : カスタムクラスの記述 214
例 : Flash でのカスタムクラスファイルの使用 227
Flash でのシンボルへのクラスの割り当て 230
クラスのコンパイルと書き出し 231
クラスとスコープについて 234
トップレベルクラスとビルトインクラスについて 237
ビルトインクラスの使用について 247

第 7 章 : 継承 253
継承について 253
Flash でのサブクラスの作成について 255
アプリケーションでのポリモーフィズムの使用 260

第 8 章 : インターフェイス 265
インターフェイスについて 265
データ型としてのインターフェイスの作成 270
継承とインターフェイスについて 272
例 : インターフェイスの使用 273
例 : 複雑なインターフェイスの作成 275

第 9 章 : イベントの処理 279
ActionScript とイベントについて 280
イベントハンドラメソッドの使用 281
イベントリスナーの使用 284
コンポーネントでのイベントリスナーの使用 287
ボタンおよびムービークリップのイベントハンドラの使用 289
コンポーネントインスタンスからのイベントのブロードキャスト 293
ボタン状態を持つムービークリップの作成 293
イベントハンドラのスコープ 295
this キーワードのスコープ 298
Delegate クラスの使用 298

第 10 章 : ムービークリップの操作 301
ActionScript によるムービークリップの制御について 302
1 つのムービークリップでの複数メソッドの呼び出し 304
SWF ファイルのロードおよびアンロード 304
ムービークリップの位置と画質の変更. 307
ムービークリップのドラッグ. 308
実行時のムービークリップの作成. 309
動的に作成したムービークリップへのパラメータの追加 313
ムービークリップの深度の管理. 314
ActionScript を使用したムービークリップのキャッシュとスクロール
について 318
マスクとしてのムービークリップの使用. 325
ムービークリップイベントの処理. 327
ムービークリップシンボルへのクラスの割り当て. 327
クラスプロパティの初期化. 328

第 11 章 : テキストとストリングの操作 331
テキストフィールドについて. 333
テキストと変数のテキストフィールドへのロードについて 341
フォントの使用 346
フォントのレンダリングとアンチエイリアステキストについて 354
テキストのレイアウトとフォーマットについて. 362
CSS スタイルによるテキストのフォーマット. 369
HTML 形式のテキストの使用 382
例 : スクロールテキストの作成 395
ストリングおよび String クラスについて 397

第 12 章 : アニメーション、フィルタおよび描画 415
ActionScript 2.0 のスクリプトによるアニメーション処理 416
ビットマップのキャッシュ、スクロール、およびパフォーマンスについて . . 426
Tween クラスと TransitionManager クラスについて 427
フィルタ効果の使用 442
ActionScript によるフィルタの操作 449
コードによるフィルタ効果の操作. 471
BitmapData クラスによるビットマップの作成. 475
ブレンドモードについて 478
操作順序について 481
ActionScript を使用した描画. 481
拡大・縮小とスライスガイドの理解. 497

第 13 章 : ActionScript によるインタラクティブ機能の作成 503
イベントとインタラクティブ機能について 504
SWF ファイル再生の制御 504
インタラクティブ機能と視覚効果の作成 508
ActionScript を使用したランタイムデータバインディングの作成 520
サンプルスクリプトの解説 528

第 14 章 : イメージ、サウンドおよびビデオの操作 531
外部メディアのロードおよび操作について 532
外部 SWF ファイルおよびイメージファイルのロード 533
外部 MP3 ファイルのロードと使用について 537
ライブラリ内のアセットへのリンケージ割り当て 542
FLV ビデオの使用について 543
メディアファイル用プログレスアニメーションの作成について 562

第 15 章 : 外部データの操作 569
変数の送信とロード 570
HTTP を使用したサーバーサイドスクリプトへの接続 574
ファイルのアップロードとダウンロードについて 579
XML について 586
Flash Player とのメッセージの送受信 595
External API について 599

第 16 章 : セキュリティについて 609
Flash Player の旧セキュリティモデルとの互換性について 610
ローカルファイルセキュリティ と Flash Player について 611
ネットワーク API の制限 627
ドメイン、ドメイン間のセキュリティ、および SWF ファイルについて . . . 629
サーバーサイドポリシーファイルによるデータへのアクセス許可 636
SWF ファイル間の HTTP プロトコルから HTTPS プロトコルへの
アクセス 641

**第 17 章 : ActionScript 2.0 のベストプラクティスとコーディング
規則 645**
命名規則 646
コード内のコメントの使用 656
ActionScript のコーディング規則 659
ActionScript と Flash Player の最適化 673
ActionScript シンタックスのフォーマット 675

付録 A: エラーメッセージ 685

付録 B: 使用を避ける Flash 4 演算子 691

付録 C: キーボードのキーとキーコードの値 693

付録 D: 以前のバージョンの Flash Player 用のスクリプト作成 699

以前のバージョンの Flash Player を対象にすることについて 699
Flash Player 4 用のコンテンツ作成での Flash の使用 700

付録 E: ActionScript 1.0 でのオブジェクト指向プログラミング 703

ActionScript 1.0 について 704
ActionScript 1.0 でのカスタムオブジェクトの作成 706
ActionScript 1.0 でのカスタムオブジェクトへのメソッドの割り当て 706
ActionScript 1.0 でのイベントハンドラメソッドの定義 708
ActionScript 1.0 での継承の作成 710
ActionScript 1.0 でのオブジェクトへの getter/setter プロパティの追加 711
ActionScript 1.0 での Function オブジェクトプロパティの使用. 712

索引 715

はじめに

Adobe Flash CS3 Professional は、インパクトの強い Web 体験を生み出すための本格的な標準オーサリングツールです。ActionScript は、アニメーション化されたシンプルな SWF ファイルから、より複雑なリッチインターネットアプリケーションまで、Flash アプリケーションにインタラクティブ機能を追加するための言語です。ActionScript がなくても、Flash を使用することはできます。しかし、基本的な、またはより高度なインタラクティブ機能を実装する場合や、Flash のビルトインオブジェクト (ボタンやムービークリップなど) 以外のオブジェクトを操作する場合、また、SWF ファイルをより堅固なユーザー体験に変貌させたい場合には、ActionScript が役立つと考えられます。

詳細については、次に示す各トピックを参照してください。

対象となる読者 9
Flash XML ファイルの更新 10
システム要件 10
本マニュアルについて 10
他の情報源 14

対象となる読者

本マニュアルでは、読者が既に Flash をインストールし、ユーザーインターフェイスの使用方法を理解していることを前提にしています。オブジェクトをステージ上に配置する方法や、Flash オーサリング環境でオブジェクトを操作する方法について知っていることが前提です。スクリプト言語を使用した経験のある方には、ActionScript はわかりやすいでしょう。プログラミング経験がなくても、ActionScript の基本を習得するのは難しいことではありません。ActionScript の習得にあたっては、非常にシンプルなコマンドから始めて、理解が進むにつれてより複雑なものを構築していくことができます。わずかなコードを学習 (または記述) するだけで、豊富なインタラクティブ機能をファイルに追加することができます。

システム要件

ActionScript 2.0 には、Flash 以外のシステム要件はありません。

Flash CS3 Professional には ActionScript 3.0 が導入されています。Flash Player 9 および ActionScript 3.0 は、Flash デフォルトのパブリッシュ設定です。本マニュアルでは、Flash で ActionScript 2.0 を使用する方法について説明します。Flash ファイルのパブリッシュ設定は、Flash Player 9 と ActionScript 2.0 に変更する必要があります。デフォルト設定を変更しないと、マニュアルの説明やコードサンプルが正常に機能しない場合があります。以前のバージョンの Flash Player 用にアプリケーションを開発する場合は、699 ページ、付録 D の「以前のバージョンの Flash Player 用のスクリプト作成」を参照してください。

Flash XML ファイルの更新

常に最新の Flash XML ファイルをインストールしておいてください。Flash Player のドットリリース (マイナーリリース) でも新機能が導入されることがあります。このようなリリースが提供された際は、Flash のバージョンを更新して最新の XML ファイルを取得してください。更新せずに新しいプロパティやメソッドを使用すると、Flash インストール時の Flash Player バージョンが対応しておらず、Flash コンパイラでエラーが発生する可能性があります。

たとえば、Flash Player 7 (7.0.19.0) では、System オブジェクトに新しいメソッド `System.security.loadPolicyFile` が追加されています。このメソッドにアクセスするには、Player Updater インストーラを使用して、Flash と共にインストールされているすべての Flash Player を更新する必要があります。更新しないと、Flash コンパイラでエラーが発生します。

Player Updater は、ご使用の Flash のバージョンよりメジャーバージョンが 1 バージョン以上新しいものであってもインストールできます。そうすることで必要な XML ファイルを入手でき、Flash Player の旧バージョン用にパブリッシュする際にもコンパイルエラーは発生しません。場合によっては、旧バージョンでも新しいメソッドやプロパティが利用可能になることがあります。最新の XML ファイルを使用することで、古いメソッドやプロパティにアクセスしようとした場合のコンパイルエラーを最小限にすることができます。

本マニュアルについて

本マニュアルでは、ActionScript シンタックスの概要、および型の異なるオブジェクト処理時の ActionScript の使用方法について説明します。各言語エレメントのシンタックスおよび使用法の詳細については、『ActionScript 2.0 リファレンスガイド』を参照してください。

詳細については、次のトピックを参照してください。

- 11 ページの「『ActionScript 2.0 の学習』マニュアルの概要」
- 14 ページの「サンプルファイルについて」
- 13 ページの「本マニュアルで使用する用語」
- 13 ページの「コードのコピーとペースト」

『ActionScript 2.0 の学習』マニュアルの概要

本マニュアルは次の内容で構成されています。

- 第 1 章の「Flash で ActionScript に追加された新機能」。ActionScript の新機能、コンパイラやデバッガに対する変更点、ActionScript 2.0 言語の新しいプログラミングモデルについて説明します。
- 第 2 章の「ActionScript について」。ActionScript 言語について定義し、どのバージョンの ActionScript を選択して使用すべきかを詳しく説明します。
- 第 3 章の「データおよびデータ型」。データ、データ型、および変数に関する用語と基本概念について説明します。これらの概念は、本マニュアル全体で使用されます。
- 第 4 章の「シンタックスと言語の基礎」。ActionScript 言語の用語と基本概念について説明します。これらの概念は、本マニュアル全体で使用されます。
- 第 5 章の「関数とメソッド」。さまざまな種類の関数とメソッドを作成し、アプリケーションで使用する方法について説明します。
- 第 6 章の「クラス」。ActionScript でカスタムクラスとオブジェクトを作成する方法について説明します。この章には、ActionScript のビルトインクラスの一覧も記載されています。また、これらのクラスを使用して ActionScript の強力な機能にアクセスする方法について簡単に説明します。
- 第 7 章の「継承」。ActionScript 言語における継承について説明し、ビルトインクラスまたはカスタムクラスを拡張する方法を示します。
- 第 8 章の「インターフェイス」。ActionScript のインターフェイスを作成および操作する方法について説明します。
- 第 9 章の「イベントの処理」。イベント処理の種類について説明します。具体的には、イベントハンドラメソッド、イベントリスナー、およびボタンやムービークリップのイベントハンドラです。
- 第 10 章の「ムービークリップの操作」。ムービークリップと、それを制御するための ActionScript について説明します。
- 第 11 章の「テキストとストリングの操作」。Flash におけるテキストおよびストリングのさまざまな制御の他、テキストの書式設定と高度なアンチエイリアスに関する情報が記載されています。
- 第 12 章の「アニメーション、フィルタおよび描画」。コードベースのアニメーションとイメージを作成する方法、オブジェクトにフィルタを追加する方法、および ActionScript を使用して描画する方法について説明します。
- 第 13 章の「ActionScript によるインタラクティブ機能の作成」。SWF ファイルを再生するタイミングの制御、カスタムポインタの作成、サウンド制御など、対話性に富んだアプリケーションを作成することのできる簡単な方法を説明します。
- 第 14 章の「イメージ、サウンドおよびビデオの操作」。Flash アプリケーションに外部メディア (ビットマップイメージ、MP3 ファイル、Flash Video (FLV) ファイル、他の SWF ファイルなど) を読み込む方法について説明します。この章では、アプリケーションでビデオを操作する方法、およびプログレスバーでロード時間のアニメーションを作成する方法についても説明します。

- 第 15 章の「外部データの操作」。アプリケーションでサーバーサイドまたはクライアントサイドのスクリプトを使用し、外部ソースからのデータを処理する方法を説明します。この章では、データをアプリケーションに統合する方法について説明します。
- 第 16 章の「セキュリティについて」。ハードディスク上の SWF ファイルをローカルで操作する場合の Flash Player におけるセキュリティについて説明します。この章では、ドメイン間のセキュリティの問題に加え、サーバーからのデータ読み込みまたはドメイン間のデータ読み込みの方法についても説明します。
- 第 17 章の「ActionScript 2.0 のベストプラクティスとコーディング規則」。Flash の使用と ActionScript の記述に関するベストプラクティス (最良の方法) の概要を説明します。この章では、変数の命名規則などの標準的なコーディング規則の一覧も示します。
- 付録 A の「エラーメッセージ」。Flash コンパイラで表示される可能性があるエラーメッセージの一覧が記載されています。
- 付録 B の「使用を避ける Flash 4 演算子」。使用されなくなった Flash 4 のすべての演算子とそれらの結合性の一覧が記載されています。
- 付録 C の「キーボードのキーとキーコードの値」。標準キーボード上のすべてのキーと、ActionScript でそれらのキーを識別するために使用する ASCII キーコード値の一覧が記載されています。
- 付録 D の「以前のバージョンの Flash Player 用のスクリプト作成」。対象とする Flash Player のバージョンに合った、正しいシンタックスのスクリプトを作成するためのガイドラインを提供します。
- 付録 E の「ActionScript 1.0 でのオブジェクト指向プログラミング」。ActionScript 1.0 オブジェクトモデルを使ってスクリプトを作成するための情報を提供します。

本マニュアルでは、ActionScript 言語の使用方法について説明しています。言語エレメントそのものについては、『ActionScript 2.0 リファレンスガイド』を参照してください。

表記規則

本マニュアルでは、次の表記規則を使用しています。

- コードフォントは ActionScript コードを示します。
- **ボールドコードフォント**は、一般的にプロシージャ内で使用し、FLA ファイルに既に追加したコードの変更または追加が必要なことを示します。場合によっては、参照するコードをハイライトするために使用します。
- **ボールドテキスト**は、ファイル名やインスタンス名など、ユーザーインターフェイスに入力する必要があるデータを示します。
- イタリックテキストは新しい用語を示し、その後に説明のテキストがあります。ファイルパス内では、置き換える必要がある値 (使用しているハードディスク上のディレクトリ名など) を示すことがあります。

本マニュアルで使用する用語

本マニュアルでは、次の用語を使用しています。

- " 自分 " とは、スクリプトやアプリケーションを作成する開発者のことです。
- " ユーザー " とは、自分が作成したスクリプトやアプリケーションを実行する人のことです。
- " コンパイル時 " とは、ドキュメントのパブリッシュ、書き出し、ムービープレビュー、またはデバッグを行うときのことです。
- " ランタイム " (実行時) とは、Flash Player で自分のスクリプトを実行するときのことです。

コードのコピーとペースト

[ヘルプ] パネルから FLA ファイルまたは ActionScript ファイルに ActionScript をペーストする場合は、特殊文字に注意する必要があります。特殊文字には、曲線引用符またはスマート引用符とも呼ばれる特殊な引用符が含まれます。特殊文字は ActionScript エディタによって解釈されないので、Flash でこれらの文字をコンパイルしようとすると、コードによってエラーがスローされます。

引用符文字が正しく色付けされない場合、これらの引用符は特殊文字であると判断できます。つまり、コードエディタで色が変わらないストリングがある場合は、特殊文字を標準の直線引用符文字で置換する必要があります。一重引用符文字または二重引用符文字を ActionScript エディタに直接入力する場合は、直線引用符文字を必ず入力してください。特殊引用符や曲線引用符などの無効な種類の文字がコードに含まれている場合に、SWF ファイルをテストまたはパブリッシュしようとすると、コンパイラによってエラーがスローされます。

メモ	特殊な引用符は、ActionScript を Web ページや Microsoft Word ドキュメントなどの他の場所からペーストした場合にも生じる場合があります。

コードをコピーしてペーストする場合は、改行が正しく適用されているかどうかに注意する必要があります。コードを別の場所からペーストすると、コード行が間違った場所で改行される場合があります。改行に問題があると考えられる場合は、ActionScript エディタでシンタックスが正しく色付けされているかどうかを確認します。[アクション] パネルと [ヘルプ] パネルのコードを比較して、コードが一致しているかどうかを確認することもできます。不要な改行がコードに挿入されないようにするには、ActionScript エディタで折り返し機能を有効にしてください。これを行うには、[スクリプト] ウィンドウで [表示]-[折り返し] メニューコマンドを選択するか、[アクション] パネルのポップアップメニューから [折り返し] を選択します。

他の情報源

ActionScript に関する本マニュアルに加え、コンポーネントや Adobe Flash Lite など、Flash 関連のその他のトピックについて扱ったマニュアルもあります。[ヘルプ]-[Flash ヘルプ] メニューコマンドを選択して、デフォルトの目次を表示すると、[ヘルプ] パネルの各マニュアルにアクセスできます。アクセス可能な各マニュアルを表示するには、[クリア] ボタンをクリックします。詳細については、16 ページの「他のトピックに関するマニュアルの場所」を参照してください。

その他の使用可能な情報源の詳細については、次のトピックを参照してください。

- 14 ページの「サンプルファイルについて」
- 14 ページの「PDF ファイルまたは印刷されたマニュアルの場所」
- 15 ページの「LiveDocs について」
- 16 ページの「その他のオンラインリソース」
- 16 ページの「他のトピックに関するマニュアルの場所」

サンプルファイルについて

Flash には、ActionScript ベースのサンプルファイルが多数付属しています。これらのサンプルファイルは、FLA ファイルにおけるコードの仕組みを示したもので、しばしば学習ツールとして使用できます。本マニュアルでは FLA ファイルがよく参照されますが、ハードディスクのサンプルファイルフォルダもチェックするようにしてください。

サンプルファイルには、Flash の一般的な機能を使用するアプリケーションの FLA ファイルが含まれています。これらのサンプルは Flash の初級開発者が Flash アプリケーションの機能を実際に体験できるように設計されています。また、上級開発者も Flash の機能を実際の現場でどのように応用すればよいかを確認できます。

ActionScript ベースのサンプルソースファイルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0" フォルダに移動してサンプルにアクセスします。コンポーネントベースのサンプルファイルについては、"ComponentsAS2" フォルダに移動します。

PDF ファイルまたは印刷されたマニュアルの場所

印刷形式のマニュアルをご希望の場合は、各ヘルプマニュアルの PDF 版をダウンロードすることができます。www.adobe.com/support/documentation/ にアクセスして、希望の製品を選択してください。ここでは、PDF 版のマニュアルを表示またはダウンロードしたり、LiveDocs 版のマニュアルにアクセスしたりできます。

印刷形式のマニュアルは購入できる場合もあります。最新の情報については、マニュアルのサポートサイトにアクセスしてください。

LiveDocs について

マニュアルには、[ヘルプ] パネルだけでなく LiveDocs の Web サイトからもアクセスすることができます。LiveDocs の Web サイトには、Flash 関連の多くのヘルプページが含まれており、マニュアルの内容を明確化、更新、または訂正したコメントが記載されている場合もあります。[ヘルプ] パネルのページの下部にある [LiveDocs のコメントを見る] をクリックすると、LiveDocs Web サイト上の該当するページが表示されます。LiveDocs 形式で利用できるすべてのマニュアルを一覧表示するには、http://livedocs.macromedia.com にアクセスしてください。

LiveDocs の Web サイトは、テクニカルライターによって管理されています。LiveDocs には、マニュアルの内容を明確化するコメントや、ソフトウェアがリリースされた後で見つかったエラーや問題を訂正するコメントを参照できるという利点があります。LiveDocs は、サポートを要求したり (正しく機能しないコードについの質問など)、ソフトウェアやインストールに関連した問題についてコメントしたり、Flash を使用して何かを作成する方法を質問したりする場所ではなく、マニュアルについてのフィードバック (説明が必要な文章や段落についての報告など) を提供する場所です。

LiveDocs にコメントを追加するためのボタンをクリックする前に、どのような種類のコメントを LiveDocs に追加できるかについて、いくつかの点に注意を払う必要があります。コメントが Web サイトから削除されないように、次に示すガイドラインによく目を通してください。

Flash に関する質問は、アドビ システムズ社の Web フォーラム (www.adobe.com/support/forums/) に投稿してください。Web フォーラムは質問を行うのに最適な場です。多くのアドビ システムズ社の社員、チーム Adobe のボランティア、Adobe ユーザーグループのマネージャとメンバー、さらにはテクニカルライターがこのフォーラムを見ています。

エンジニアは LiveDocs システムには目を通しませんが、Flash ウィッシュリストは管理しています。ソフトウェアのバグを発見したと考えられる場合や、Flash の機能強化を提案したい場合は、www.adobe.com/go/wish_jp のウィッシュフォームに入力してください。LiveDocs でバグを報告するか機能強化を提案しても、バグデータベースに正式に追加されるわけではありません。バグまたは提案をエンジニアに確認してもらいたい場合は、LiveDocs ではなくウィッシュフォームを使用する必要があります。

LiveDocs などの Web サイトからペーストする場合は、特殊文字や改行に注意することを忘れないでください。アドビ システムズ社では、サンプルコードからすべての特殊文字を削除するように努力していますが、コードのペーストで問題が発生する場合は、13 ページの「コードのコピーとペースト」を参照してください。

その他のオンラインリソース

Flash の学習に役立つ豊富な手順、ヘルプ、およびガイドを提供するいくつかのオンラインリソースがあります。次の Web サイトに頻繁にアクセスし、最新情報をチェックしてください。

Flash の最新情報、アドバンスドユーザーからのアドバイス、高度なテクニック、サンプル、ヒント、複数パートのチュートリアルを含むチュートリアル、その他の更新情報については、**Adobe デベロッパーセンター Web サイト** (www.adobe.com/devnet) を参照してください。このサイトは定期的に更新されます。Flash の最新ニュースと Flash を最大限に活用する方法を提供するこの Web サイトを頻繁にアクセスしてみてください。

Adobe Flash サポートセンター (www.adobe.com/support/flash) には、テクニカルノート、マニュアルの更新情報、および Flash コミュニティの追加情報へのリンクが用意されています。

Adobe Weblogs Web サイト (http://weblogs.macromedia.com) には、アドビ システムズ社の社員とコミュニティの両方の Web ログ (" ブログ " とも呼ばれます) の一覧が記載されています。

アドビ システムズ社の Web フォーラム (http://www.adobe.com/support/forums/) には、Flash、アプリケーション、または ActionScript 言語に関する具体的な質問を投稿できる数多くのフォーラムがあります。このフォーラムは、チーム Adobe のボランティアによって管理され、アドビ システムズ社の社員も頻繁にアクセスしています。参照する場所または問題の解決方法がわからない場合は、初めに Flash フォーラムを参照することをお勧めします。

Adobe Community Web サイト (www.adobe.com/community) では、MacroChat が定期的に開催され、Adobe 社員やコミュニティのメンバーがさまざまなトピックに関する一連のライブプレゼンテーションを行っています。Web サイトに頻繁にアクセスし、最新情報をチェックしてください。

他のトピックに関するマニュアルの場所

次のマニュアルには、ActionScript 2.0 関連の内容に関する追加情報が記載されています。

- ActionScript 言語を構成するエレメントについては、『ActionScript 2.0 リファレンスガイド』を参照してください。
- Flash オーサリング環境での操作については、『Flash ユーザーガイド』を参照してください。
- コンポーネントの操作については、『ActionScript 2.0 コンポーネントガイド』を参照してください。

第 1 章

Flash で ActionScript に追加された新機能

1

Adobe Flash CS3 Professional には、ActionScript 言語で強固なスクリプトを作成しやすくするための新機能がいくつか備わっています。新機能の内容としては、新しい言語エレメント (20 ページの「Flash Player 8 の ActionScript 言語に対する追加機能」を参照)、さらに機能が向上した編集ツール (25 ページの「ActionScript エディタの変更」を参照)、セキュリティモデルに対する変更、オーサリングツールに加えられた ActionScript 関連のその他の機能向上があります。この章では、これらの新機能について説明します。

詳細については、次のトピックを参照してください。

ActionScript 2.0 と Flash Player 9.x の新機能 17
ActionScript 2.0 と Flash Player 8 の新機能. 18
ローカルでインストールされた SWF ファイルのセキュリティモデルの変更 26

ActionScript 2.0 と Flash Player 9.x の新機能

Flash Player 9.x では、ユーザーがフルスクリーンモードの使用と不使用を切り替えることができます。この新機能は次のものでサポートされています。

- `Stage.displayState` プロパティ
- `Stage.onFullScreen` イベントハンドラ
- `object` および `embed` の各 HTML タグの `allowFullScreen` パラメータ

詳細については、『ActionScript 2.0 リファレンスガイド』を参照してください。

Flash Player 9.x には、新しい FileReference イベント `onUploadCompleteData` も導入されました。このイベントは、サーバーへのアップロードが正常に終了した後に呼び出すことができます。この新しいイベントの詳細については、『ActionScript 2.0 リファレンスガイド』を参照してください。

ActionScript 2.0 と Flash Player 8 の新機能

ActionScript 言語は、数年前に登場して以来、拡張と発展を続けています。Flash では、新しくリリースされるたびに、キーワード、オブジェクト、メソッド、その他の言語エレメントが ActionScript に追加されてきました。Flash 8 オーサリング環境でも、ActionScript 関連の改良点があります。Flash Basic 8 および Flash Professional 8 には、いくつかの新しい言語エレメントが導入されています。具体的には、フィルタおよびブレンドモードなどの表現力機能、および JavaScript の統合 (ExternalInterface) やファイル入出力 (FileReference および FileReferenceList) などのアプリケーション開発の言語エレメントが導入されました。

このセクションでは、Flash 8 で追加または変更された ActionScript 言語エレメントおよびクラスに加え、オーサリングツールに加えられた ActionScript 関連の機能向上について概説します。ActionScript 2.0 に特有の追加機能の一覧は、20 ページの「Flash Player 8 の ActionScript 言語に対する追加機能」を参照してください。新しい言語エレメントをスクリプトで使用するには、ドキュメントをパブリッシュするときに Flash Player 8 を指定する必要があります。

次の機能は、特に説明がない限り、Flash Basic 8 と Flash Professional 8 の両方に追加されています

- ActionScript エディタの拡張機能を使用すると、スクリプトで隠し文字を表示できます。詳細については、『Flash ユーザーガイド』の「隠し文字の表示」を参照してください。
- デバッグオプションは、[スクリプト] ウィンドウと [アクション] パネルの両方で ActionScript ファイルに対して使用できるようになりました。
- XML ファイルとクラスファイルを格納する "Configuration" ディレクトリが再編成されました。詳細については、『Flash ユーザーガイド』の「Flash とともにインストールされる設定フォルダ」を参照してください。
- アプリケーションの使用時に変更済みのスクリプトファイルをリロードするように環境設定で指定すると、古いバージョンのスクリプトファイルが使用されたり、新しいスクリプトファイルが上書きされたりするのを防ぐことができます。詳細については、『Flash ユーザーガイド』の「ActionScript の環境設定」を参照してください。
- Flash ではスクリプトウィンドウの機能を使用できます。つまり、どちらのプログラムでも ActionScript ファイルを作成できるということです。
- スクリプトアシストは、以前のエディションの Flash におけるノーマルモードに似ています。スクリプトアシストを使用すると、シンタックスを知らなくてもコードを作成することができます。スクリプトアシストの詳細については、『Flash ユーザーガイド』の「スクリプトアシストモードについて」を参照してください。
- プログレッシブ JPEG イメージ、アニメーション以外の GIF ファイルと PNG ファイルなど、新しい種類のイメージファイルをランタイムにロードできるようになりました。アニメーションファイルをロードすると、アニメーションの最初のフレームが表示されます。

- ライブラリに格納されたビットマップファイルやサウンドファイルには、リンケージ識別子を割り当てることができます。これにより、イメージをステージに割り当てたり、これらのアセットを共有ライブラリで使用したりできるようになりました。
- ビットマップキャッシュを使用すると、インスタンスのビットマップ表現をキャッシュしてアプリケーションのパフォーマンスを向上できます。このプロパティには、ActionScript のコードを使用してアクセスできます。詳細については、426 ページの「ビットマップのキャッシュ、スクロール、およびパフォーマンスについて」を参照してください。
- 9 つのスライスの拡大・縮小を使用すると、ムービークリップの外枠を広げずにムービークリップのインスタンスを拡大できます。Flash のこの機能には、ActionScript のコードを使用してアクセスできます。詳細については、499 ページの「ActionScript による 9 スライスの拡大 / 縮小の操作」を参照してください。オーサリングツールで 9 つのスライスの拡大・縮小機能にアクセスする方法については、『Flash ユーザーガイド』を参照してください。
- [ドキュメントプロパティ] ダイアログボックスでは、メタデータ情報を FLA ファイルに追加できるようになりました。このダイアログボックスを使用して FLA ファイルに名前と説明を追加すると、オンライン検索での可視性を高めることができます。
- [ストリング] パネルが機能向上され、[ストリング] フィールドと言語 XML ファイルで複数行がサポートされるようになりました。詳細については、398ページの「パネル」を参照してください。
- 新しいガベージコレクターが Flash Player に組み込まれました。このコレクターでは増分的なコレクションが行われるので、パフォーマンスの向上に役立ちます。
- アクセシビリティアプリケーションを作成するためのワークフローが改善されました。開発者は、Flash Player 8 でタブインデックスにすべてのオブジェクトを追加しなくても、スクリーンリーダーでコンテンツを正しく読み取ることができます。タブインデックスの詳細については、『ActionScript 2.0 リファレンスガイド』の「`tabIndex (Button.tabIndex プロパティ )`」、「`tabIndex (MovieClip.tabIndex プロパティ )`」、および「`tabIndex (TextField.tabIndex プロパティ )`」を参照してください。
- Flash Player では、ローカルファイルのセキュリティが強化されたので、ハードディスク上の SWF ファイルをさらに安全に実行できるようになりました。ローカルファイルのセキュリティについては、611 ページの「ローカルファイルセキュリティ と Flash Player について」を参照してください。
- ActionScript コードと Drawing API を使用すると、描画した線のスタイルを調整することができます。新しい線のスタイルについては、489 ページの「線スタイルの使用」を参照してください。
- ActionScript コードと Drawing API を使用すると、シェイプの塗りつぶしに使用するためのより複雑なグラデーションを作成できます。グラデーションの塗りについては、488 ページの「複雑なグラデーションの塗りの使用」を参照してください。
- ActionScript コードを使用すると、ステージ上のオブジェクトに複数のフィルタを適用できます (ムービークリップのインスタンスなど)。フィルタと ActionScript については、449 ページの「ActionScript によるフィルタの操作」を参照してください。

- FileReference と FileReferenceList の各 API を使用すると、ファイルをサーバーにアップロードできます。詳細については、579 ページの「ファイルのアップロードとダウンロードについて」を参照してください。
- ActionScript コードを使用すると、カラーを適用および操作するための新しい方法や高度な方法にアクセスできます。詳細については、『ActionScript 2.0 リファレンスガイド』の 513 ページの「カラー値の設定」および「ColorTransform (flash.geom.ColorTransform)」を参照してください。
- テキスト処理機能には、TextField クラスと TextFormat クラスの新しいオプション、プロパティ、パラメータなどの多数の機能向上が加えられています。詳細については、『ActionScript 2.0 リファレンスガイド』の「TextField」および「TextFormat」を参照してください。
- 高度なアンチエイリアス機能には、ActionScript のコードを使用してアクセスできます。詳細については、354 ページの「フォントのレンダリングとアンチエイリアステキストについて」を参照してください。
- アプリケーションのテスト時に ASO ファイルを削除することができます。オーサリングツールで [制御]-[ASO ファイルの削除] または [制御]-[ASO ファイルを削除してムービーをプレビューする] の各メニューコマンドを選択します。233 ページの「ASO ファイルの使用」を参照してください。

Flash 8 で ActionScript 2.0 に追加されたクラス、言語エレメント、メソッド、およびプロパティの一覧は、『Flash ユーザーガイド』の「ActionScript の環境設定」を参照してください。

Flash Player 8 の ActionScript 言語に対する追加機能

次のクラスと言語要素は、Flash Player 8 で新しく追加またはサポートされるようになったものです。

Flash 8 の ActionScript 2.0 には次のクラスが追加されました。

- flash.filters パッケージに含まれている BevelFilter クラスを使用すると、オブジェクトにベベル効果を追加できます。
- flash.display パッケージに含まれている BitmapData クラスを使用すると、任意のサイズの透明ビットマップイメージまたは不透明ビットマップイメージを作成および操作できます。
- flash.display パッケージに含まれている BitmapFilter クラスは、フィルタ効果の基本となるクラスです。
- BlurFilter クラスを使用すると、Flash のオブジェクトにぼかしを適用できます。
- flash.filters パッケージに含まれている ColorMatrixFilter クラスを使用すると、ARGB カラーとアルファ値に変形を適用できます。
- flash.geom パッケージに含まれている ColorTransform クラスを使用すると、ムービークリップのカラー値を調整できます。Color クラスは使用せず、代わりに ColorTransform クラスを使用してください。

- flash.filters パッケージに含まれている ConvolutionFilter クラスを使用すると、マトリックス畳み込みフィルタ効果を適用できます。
- flash.filters パッケージに含まれている DisplacementMapFilter クラスを使用すると、BitmapData オブジェクトのピクセル値を使用してオブジェクトで置き換えを実行できます。
- flash.filters パッケージに含まれている DropShadowFilter クラスを使用すると、オブジェクトにドロップシャドウを追加できます。
- flash.external パッケージに含まれている ExternalInterface クラスを使用すると、ActionScript によって Flash Player コンテナ (JavaScript 付きブラウザなどの Flash アプリケーションやデスクトップアプリケーションを搭載したシステム) と通信することができます。
- flash.net パッケージに含まれている FileReference クラスを使用すると、ユーザーのコンピュータとサーバーとの間でファイルをアップロードおよびダウンロードできます。
- flash.net パッケージに含まれている FileReferenceList クラスを使用すると、アップロードするファイル (複数可) を選択できます。
- flash.filters パッケージに含まれている GlowFilter クラスを使用すると、オブジェクトにグロー効果を追加できます。
- flash.filters パッケージに含まれている GradientBevelFilter クラスを使用すると、オブジェクトにグラデーションベベルを追加できます。
- flash.filters パッケージに含まれている GradientGlowFilter クラスを使用すると、オブジェクトにグラデーショングロー効果を追加できます。
- System クラスの IME クラスを使用すると、オペレーティングシステムの IME (Input Method Editor : 入力方式エディタ) を Flash Player 内で操作できます。
- mx.lang パッケージに含まれている Locale クラスを使用すると、SWF ファイルにおける多言語テキストの表示方法を制御できます。
- flash.geom パッケージに含まれている Matrix クラスは、座標空間の間でポイントをマッピングする方法を決定する変換マトリックス (変換行列) を表します。
- flash.geom パッケージに含まれている Point クラスは、x が水平軸、y が垂直軸を表す、2 次元の座標系内の位置を表します。
- flash.geom パッケージに含まれている Rectangle クラスを使用すると、Rectangle オブジェクトを作成および変更できます。
- flash.text パッケージに含まれている TextRenderer クラスには、埋め込みフォントの高度なアンチエイリアス機能が用意されています。
- flash.geom パッケージに含まれている Transform クラスは、MovieClip オブジェクトに適用されるカラー変換と座標操作に関するデータを収集します。

メモ	Flash 8 では、AsBroadcaster クラスが正式にサポートされるようになりました。

ActionScript の既存のクラスに追加された言語エレメント、メソッド、および関数は次のとおりです。

- showRedrawRegions グローバル関数。再描画されている画面領域 (つまり更新中のダーティ領域) をデバッガプレーヤーがアウトライン表示できるようにします。この関数を実行すると、再描画された内容がプレイヤーに表示されますが、ユーザーが再描画領域を制御することはできません。
- Button クラスの blendMode プロパティ。ボタンインスタンスのブレンドモードを設定します。
- Button クラスの cacheAsBitmap プロパティ。オブジェクトをインスタンスの内部ビットマップ表現としてキャッシュできます。
- Button クラスの filters プロパティ。ボタンに関連付けられた各フィルタオブジェクトが含まれているインデックス配列です。
- Button クラスの scale9Grid プロパティ。インスタンスの 9 つの拡大・縮小領域を定義する矩形領域です。
- System.capabilities クラスの hasIME プロパティ。システムに IME がインストールされているかどうかを示します。
- Date クラスの getUTCYear プロパティ。この日付の世界時に基づく年を返します。
- Key クラスの isAccessible() メソッド。他の SWF ファイルがセキュリティ制限に基づいて最後に押されたキーにアクセスできるかどうかを示すブール値を返します。
- LoadVars クラスの onHTTPStatus イベントハンドラ。サーバーから返されるステータスコード (ページが見つからなかった場合の値 404 など) を返します。詳細については、『ActionScript 2.0 リファレンスガイド』の「HTTPStatus (LoadVars.onHTTPStatus ハンドラ)」を参照してください。
- MovieClip クラスの attachBitmap() メソッド。ビットマップイメージをムービークリップに割り当てます。詳細については、『ActionScript 2.0 リファレンスガイド』の「BitmapData (flash.display.BitmapData)」を参照してください。
- MovieClip クラスの beginBitmapFill() メソッド。ムービークリップをビットマップイメージで塗りつぶします。
- MovieClip クラスの beginGradientFill() メソッドで使用される spreadMethod、interpolationMethod、および focalPointRatio の各パラメータ。このメソッドを使用すると、描画領域がビットマップイメージで塗りつぶされ、ビットマップを繰り返すか、タイリング表示して、領域を塗りつぶすことができます。
- MovieClip クラスの blendMode プロパティ。インスタンスのブレンドモードを設定できます。
- MovieClip クラスの cacheAsBitmap プロパティ。オブジェクトをインスタンスの内部ビットマップ表現としてキャッシュできます。
- MovieClip クラスの filters プロパティ。インスタンスに現在関連付けられた各フィルタオブジェクトが含まれているインデックス配列です。
- MovieClip クラスの getRect() メソッド。指定されたインスタンスの最小および最大の座標値を示すプロパティを返します。

- MovieClip クラスの lineGradientStyle() メソッド。Flash でパスを描画するときに使用されるグラデーション線のスタイルを指定します。
- MovieClip クラスの lineStyle() メソッドで使用される pixelHinting、noScale、capsStyle、jointStyle、および miterLimit の各パラメータ。これらのパラメータは、線を描画するときに使用できる線のスタイルの種類を指定します。
- MovieClip クラスの opaqueBackground プロパティ。ムービークリップの不透明な背景色を 16 進数の RGB 値が指定するカラーに設定します。
- MovieClip クラスの scale9Grid プロパティ。インスタンスの 9 つの拡大・縮小領域を定義する矩形領域です。
- MovieClip クラスの scrollRect プロパティ。ムービークリップコンテンツをすばやくスクロールし、ウィンドウよりも大きなコンテンツをウィンドウに表示できます。
- MovieClip クラスの transform プロパティ。ムービークリップのマトリックス、カラー変換、およびピクセル境界に関連した設定を指定できます。詳細については、『ActionScript 2.0 リファレンスガイド』の「Transform (flash.geom.Transform)」を参照してください。
- MovieClipLoader.onLoadComplete イベントハンドラの status パラメータ。サーバーから返されるステータスコード (ページが見つからなかった場合の値 404 など) を返します。詳細については、『ActionScript 2.0 リファレンスガイド』の「onLoadComplete (MovieClipLoader.onComplete イベントリスナー)」を参照してください。
- MovieClipLoader クラスの onLoadError イベントハンドラ。MovieClipLoader.loadClip() によるファイルのロードが失敗すると呼び出されます。
- SharedObject.getLocal() メソッドの secure パラメータ。この共有オブジェクトへのアクセスを HTTPS 接続を介して配布される SWF ファイルに制限するかどうかを決定します。詳細については、『ActionScript 2.0 リファレンスガイド』の「getLocal (SharedObject.getlocal メソッド)」を参照してください。
- System.security クラスの sandboxType プロパティ。呼び出し元の SWF ファイルに適用されている Sandbox セキュリティ機能の種類を示します。詳細については、『ActionScript 2.0 リファレンスガイド』の「sandboxType (security.sandboxType プロパティ)」を参照してください。
- TextField クラスの antiAliasType プロパティ。TextField インスタンスに対して使用する高度なアンチエイリアス処理の種類を設定します。
- TextField クラスの filters プロパティ。TextField インスタンスに現在関連付けられた各フィルタオブジェクトが含まれているインデックス配列です。
- TextField クラスの gridFitType プロパティ。インスタンスに対して使用するグリッドフィッティングの種類を設定します。グリッドフィッティングと TextField.gridFitType については、『ActionScript 2.0 リファレンスガイド』の「gridFitType (TextField.gridFitType プロパティ)」を参照してください。

- TextField クラスの sharpness プロパティ。TextField インスタンスの文字エッジのシャープネスを設定します。このプロパティを使用する場合は、antiAliasType() メソッドを advanced に設定する必要があります。
- TextField クラスの thickness プロパティ。TextField インスタンスの文字エッジの太さを設定します。このプロパティを使用する場合は、antiAliasType() メソッドを advanced に設定する必要があります。
- TextFormat クラスの align プロパティに対する justify 値。指定された段落を両端揃えにすることができます。
- TextFormat クラスの indent プロパティ。負の値を使用できるようになります。
- TextFormat クラスの kerning プロパティ。TextFormat オブジェクトのカーニングのオンとオフを切り替えることができます。
- TextFormat クラスの leading プロパティ。行間の行送りがテキストの高さより小さくなる、負の行送りを指定できます。これにより、アプリケーションでテキスト行を近くにまとめて配置できるようになります。
- TextFormat クラスの letterSpacing プロパティ。文字間に均等に配分されるスペースの量を指定できます。
- Video クラスの _alpha プロパティ。ビデオオブジェクトに対して指定された透明度の量を示します。
- Video クラスの _height プロパティ。ビデオインスタンスの高さを示します。
- Video クラスの _name プロパティ。ビデオのインスタンス名を示します。
- Video クラスの _parent プロパティ。ビデオインスタンスが含まれているムービークリップインスタンスまたはオブジェクトを示します。
- Video クラスの _rotation プロパティ。ビデオインスタンスの回転の量を度単位で設定できます。
- Video クラスの _visible プロパティ。ビデオインスタンスの可視性を設定できます。
- Video クラスの _width プロパティ。ビデオインスタンスの幅を設定できます。
- Video クラスの _x プロパティ。ビデオインスタンスの x 座標を設定できます。
- Video クラスの _xmouse プロパティ。マウスポインタの位置の x 座標を設定できます。
- Video クラスの _xscale プロパティ。ビデオインスタンスの水平スケールをパーセントで設定できます。
- Video クラスの _y プロパティ。ビデオインスタンスの y 座標を設定できます。
- Video クラスの _ymouse プロパティ。マウスポインタの位置の y 座標を設定できます。
- Video クラスの _yscale プロパティ。ビデオインスタンスの垂直スケールをパーセントで設定できます。

- XML クラスの onHTTPStatus イベントハンドラ。サーバーから返されるステータスコード (ページが見つからなかった場合の値 404 など) を返します。詳細については、『ActionScript 2.0 リファレンスガイド』の「onHTTPStatus（XML.onHTTPStatus ハンドラ）」を参照してください。
- XMLNode クラスの localName プロパティ。XML ノードオブジェクトの接頭辞とローカル名を両方含む完全な名前を返します。
- XMLNode クラスの namespaceURI プロパティ。XML ノードの接頭辞が解決される名前空間の URI を読み取ります。詳細については、『ActionScript 2.0 リファレンスガイド』の「namespaceURI（XMLNode.namespaceURI プロパティ）」を参照してください。
- XMLNode クラスの prefix プロパティ。ノード名の接頭辞を読み取ります。
- XMLNode クラスの getNamespaceForPrefix() メソッド。ノードに指定された接頭辞に関連付けられている名前空間 URI を返します。
- XMLNode クラスの getPrefixForNamespace メソッド。ノードに指定された名前空間に関連付けられている接頭辞を返します。

使用されなくなった言語エレメントについて

Flash Player 8 では、一部の言語エレメントが使用されなくなっています。これらの言語エレメントと Flash Player 8 で代わりに使用できる言語エレメントの一覧は、『ActionScript 2.0 リファレンスガイド』の次のセクションに記載されています。

- 使用されなくなったクラスの一覧
- 使用されなくなった関数の一覧
- 使用されなくなったプロパティの一覧
- 使用されなくなった演算子の一覧

ActionScript エディタの変更

[アクション] パネルと [スクリプト] ウィンドウの ActionScript エディタには、以前のバージョンより堅牢かつ使いやすくするためのいくつかの変更が加えられています。このセクションではそうした変更点について説明します。

隠し文字の表示　[アクション] パネルまたは [スクリプト] ウィンドウでスクリプトファイルを作成するときに、スクリプトペイン、[デバッガ] パネル、および [出力] パネルで、[オプション] ポップアップメニューを使って隠し文字の表示と非表示を切り替えることができるようになりました。この機能の詳細については、『Flash ユーザーガイド』を参照してください。

[アクション] パネルに追加されたスクリプトアシスト　以前のバージョンの Flash では、[アクション] パネルは " ノーマルモード " と " エキスパートモード " に分かれていました。ノーマルモードではオプションとパラメータを埋めていく形で、エキスパートモードではスクリプトペインにコマンドを直接追加する形で、それぞれコードを作成していました。これらのオプションは、Flash MX 2004 または Flash MX Professional 2004 では使用できませんでした。ただし、Flash Basic 8 および Flash Professional 8 では、" スクリプトアシスト " モードで作業できます。これは、ノーマルモードに似ていますがより安定しています。スクリプトアシストの詳細およびチュートリアルについては、『Flash ユーザーガイド』を参照してください。

変更したファイルのリロード　アプリケーションの使用時に変更済みのスクリプトファイルをリロードできるようになりました。この機能により、警告メッセージが表示され、使用中のアプリケーションに関連付けられた変更済みのスクリプトファイルをリロードするよう求められます。この機能は、古いバージョンのスクリプトが使用されたり、新しいバージョンのスクリプトが上書きされたりするのを防ぐので、アプリケーションで同時に作業するチームにとっては特に効果的です。スクリプトファイルが移動または削除された場合は、必要に応じてファイルを保存するよう求める警告メッセージが表示されます。詳細については、『Flash ユーザーガイド』の「ActionScript の環境設定」を参照してください。

ローカルでインストールされた SWF ファイルのセキュリティモデルの変更

Flash Player 9.x では、新しい HTML タグ `allowNetworking` のサポートが導入されました。詳細については、『ActionScript 3.0 のプログラミング』の「Flash Player セキュリティ」の章を参照してください。

Flash Player 8 には、機能向上された新しいセキュリティモデルが備わっています。このモデルの下では、ローカルコンピュータにインストールされた Flash アプリケーションと SWF ファイルをリモートの Web サーバーから実行するのではなく、インターネットやローカルのファイルシステムと通信させることができます。Flash アプリケーションを開発する場合は、SWF ファイルがネットワークまたはローカルファイルシステムのどちらと通信できるようにするかを指定する必要があります。

メモ	この説明中の " ローカル SWF ファイル " とは、Web サイトから提供された SWF ファイルではなく、ユーザーのコンピュータにローカルでインストールされた SWF ファイルであり、プロジェクタ (EXE) ファイルを含まないものとします。

以前のバージョンの Flash Player では、セキュリティ設定を構成しなくても、ローカル SWF ファイルが他の SWF ファイルと対話して、リモートまたはローカルの任意のコンピュータからデータをロードすることが可能でした。Flash Player 8 以降の SWF ファイルは、セキュリティ設定を構成せずに同じアプリケーション内のローカルファイルシステムやネットワーク (インターネットなど) と接続することはできません。これは、安全上の理由によるもので、SWF ファイルがハードディスク上のファイルを読み取って、ファイルのコンテンツをインターネットに送信してしまうのを防ぐ意味があります。

このセキュリティ制限は、FLA ファイルが Flash 8 以降で作成されたか、それより前のバージョンで作成されたかに関係なく、ローカルで展開されたすべてのコンテンツに影響します。Flash MX 2004 以前のオーサリングツールを使用すると、ローカルで実行され、インターネットにもアクセスする Flash アプリケーションをテストできました。Flash Player 8 以降では、このアプリケーションは、インターネットと通信するための許可をユーザーに要求するプロンプトを表示します。

ハードディスク上のファイルをテストする場合は、ファイルが信頼性のある (安全な) ローカルドキュメントであるか、信頼できない可能性がある (危険な) ドキュメントであるかを判定するために、いくつかの手順を実行する必要があります。[制御]-[ムービープレビュー] メニューコマンドを選択するなどして Flash オーサリング環境で作成したファイルは、テスト環境にあるので信頼できます。

Flash Player 7 以前では、ローカル SWF ファイルは、ローカルファイルシステムとネットワークの両方へのアクセスが許可されていました。Flash Player 8 以降では、ローカル SWF ファイルには、次の 3 つのレベルの許可を指定できます。

- ローカルファイルシステムにのみアクセスする (デフォルトのレベル)。ローカル SWF ファイルは、ローカルファイルシステムと UNC (Universal Naming Convention) ネットワークパスから読み取ることができ、インターネットと通信することはできません。
- ネットワークにのみアクセスする。ローカル SWF ファイルは、インターネットなどのネットワークにのみアクセス可能で、SWF ファイルがインストールされたローカルファイルシステムにはアクセスできません。
- ローカルファイルシステムとネットワークの両方にアクセスする。ローカル SWF ファイルは、ファイルがインストールされたローカルファイルシステムから読み取り可能であり、アクセス許可を与えられているすべてのサーバーで読み取りおよび書き込み可能で、さらにアクセス許可が与えられたネットワークまたはローカルファイルシステム上の他の SWF ファイルとクロススクリプト可能です。

各アクセス許可レベルの詳細については、611 ページの「ローカルファイルセキュリティ と Flash Player について」を参照してください。

また、System.security.allowDomain には若干の変更が加わり、System.security.allowInsecureDomain には機能向上が加えられています。ローカルファイルセキュリティの詳細については、第 16 章の「セキュリティについて」を参照してください。

第 2 章

ActionScript について

2

ActionScript 2.0 のオブジェクト指向プログラミング (OOP) 機能は、ECMA TC39-TG1 で策定中の ECMAScript 4 規格の仕様草案に基づいています (www.mozilla.org/js/language/es4/index.html を参照)。この ECMA-4 仕様案はまだ規格として確定したものではなく、現在も変更が加えられているため、ActionScript 2.0 はこの仕様に完全に準拠しているわけではありません。

ActionScript 2.0 は、ActionScript 言語の標準的なエレメントをすべてサポートしつつ、他のオブジェクト指向言語 (Java など) で使われている標準的な手法をより忠実に取り入れたスクリプトの作成を可能にしたものです。ActionScript 2.0 は、主として、クラスやサブクラスの実装が必要なアプリケーションを構築している中級から上級の Flash 開発者にとって有用です。また、ActionScript 2.0 では、変数を作成するときにそのオブジェクト型を宣言することができます (詳細については 42 ページの「データ型の割り当てと厳密な型指定について」を参照)。さらには、コンパイラのエラーも大幅に向上しています (詳細については 685 ページ、付録 A の「エラーメッセージ」を参照)。

次に、ActionScript 2.0 でキーポイントとなる点をいくつか述べます。

- クラスやインターフェイスを定義する ActionScript 2.0 のスクリプトは外部スクリプトファイルとして保存する必要があり、各スクリプトにそれぞれ 1 つのクラスを定義するという形で作成します。つまり、クラスやインターフェイスは [アクション] パネルでは定義できません。
- 各クラスファイルの読み込みは、暗黙に行うことも明示的に行うこともできます。暗黙に行う場合は、グローバルな検索パスまたはドキュメント固有の検索パスで指定された場所にクラスファイルを格納し、スクリプトで使用します。一方、明示的に行う場合は、`import` コマンドを使用します。また、ワイルドカードを使用すると、パッケージ (1 つのディレクトリ内のクラスファイルの集まり) を読み込むこともできます。
- ActionScript 2.0 で開発されたアプリケーションは Flash Player 6 以降でサポートされます。

注意	Flash 9 で作成する新規ファイルのデフォルトパブリッシュ設定は ActionScript 3.0 です。ActionScript 2.0 シンタックスを使用するように ActionScript 1.0 または ActionScript 2.0 の既存の FLA ファイルを変更する場合は、その FLA ファイルのパブリッシュ設定で必ず ActionScript 2.0 を指定してください。これを怠ると、コンパイルエラーは生成されませんが、ファイルが正しくコンパイルされない場合があります。

ActionScript 2.0 を使用して Flash でオブジェクト指向プログラムを作成する方法の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

アドビ システムズ社では ActionScript 3.0 の使用を推奨していますが、ActionScript 1.0 および ActionScript 2.0 のシンタックスも引き続き使用できます。

ActionScript とは

ここで、ActionScript 2.0 の特徴について説明します。

身近なオブジェクト指向プログラミング (OOP) モデル　ActionScript 2.0 の最も大きな特長は、オブジェクト指向プログラムを作成するための身近なモデルです。ActionScript 2.0 では、" クラス "、" インターフェイス "、" パッケージ " など、Java プログラミングの経験がある人にはなじみのあるオブジェクト指向の概念とキーワードが新たに導入されています。

ActionScript 2.0 の OOP モデルは、以前のバージョンの Flash でオブジェクトの作成や継承の確立のために使用されていたプロトタイプチェーン方式を体系化したものです。ActionScript 2.0 では、カスタムクラスを作成したり、Flash のビルトインクラスを拡張することができます。

厳密なデータ型指定　ActionScript 2.0 では、変数、関数パラメータ、関数の戻り値タイプを明示的に指定することもできます。たとえば、次のコードでは、`userName` という名前の String 型の変数を宣言しています。String は ActionScript のビルトインデータ型 (つまりクラス) です。

```
var userName:String = "";
```

コンパイラからの警告とエラー　前述の 2 つの機能 (OOP モデルおよび厳密なデータ型指定) により、オーサリングツールとコンパイラがコンパイル時に警告とエラーメッセージを提供できるようになりました。そのため、以前の Flash よりも、アプリケーション内のバグを迅速に見つけることができます。

ActionScript 2.0 を使用する場合は、FLA ファイルのパブリッシュ設定で ActionScript 2.0 を指定してください (Flash 9 のデフォルトは ActionScript 3.0 です)。また、ActionScript 1.0 を使用した古い FLA ファイルを開き、ActionScript 2.0 でコードを書き直す場合は、FLA ファイルのパブリッシュ設定を ActionScript 2.0 に変更する必要があります。この設定を行わないと、FLA ファイルが正しくコンパイルされず、エラーも生成されません。

ActionScript 1.0 または ActionScript 2.0 の選択について

Flash で新規ドキュメントやアプリケーションの作成を開始するとき、関連ファイルの構成方法を決定する必要があります。アプリケーションや複雑な FLA ファイルを作成する場合など、プロジェクトによってはクラスを使用しますが、すべてのドキュメントがクラスを使用する訳ではありません。たとえば、マニュアル内の短い多くの例ではクラスを使用していません。小規模なアプリケーションや単純な FLA ファイルの場合、クラスを使って機能を格納する方法は必ずしも簡単で最良のソリューションではありません。このような場合は、ドキュメント内に ActionScript を配置する方がしばしば効率的です。このようなケースでは、できるだけ少ない数のフレーム上のタイムラインにすべてのコードを配置して、FLA ファイル内のインスタンス (ボタン、ムービークリップなど) にはコードを配置しないようにします。

小規模なプロジェクトを作成する場合は、FLA ファイル内に ActionScript を追加する代わりにクラスや外部コードファイルを使用して ActionScript を構成すると、より多くの作業や手間が必要になる場合があります。FLA ファイルにすべての ActionScript コードを配置する方が、場合によってはクラスにして読み込むよりも簡単です。ただし、これは ActionScript 1.0 を使用する必要があるという意味ではありません。FLA ファイル内にコードを配置する場合でも、ActionScript 2.0 で厳格な型指定や新しいメソッド、プロパティを使用できます。また、ActionScript 2.0 のシンタックスは他の各種プログラミング言語における標準に則しているので、より学習しやすく、また学習する価値の高い言語となっています。たとえば、同様の構造やシンタックス標準に基づく他の言語を使用した経験があれば、ActionScript は身近な言語と感じられるでしょう。また、逆に ActionScript で学んだ知識を、後で別の言語に応用することもできます。ActionScript 2.0 では、追加された言語エレメントによって、オブジェクト指向アプローチを使用したアプリケーション開発が可能です。このこともアプリケーション開発に有利になる可能性があります。

場合によっては、ActionScript のバージョンを選択できる余地がないこともあります。旧バージョンの Flash Player をターゲットとする SWF ファイル (モバイルデバイス用アプリケーションなど) を作成する場合は、多くのデバイスの Flash Player と互換性のある ActionScript 1.0 を使用する必要があります。

ActionScript のバージョンにかかわらず、適切な規則に従うことが大切です。大文字と小文字の一貫した使い分け、コードコンプリートの使用、読みやすさの向上、インスタンス名にキーワードを使用しない、一貫した命名規則など、規則の多くは両方のバージョンに当てはまります。

アプリケーションを Flash の将来のバージョンに合わせて更新したり、その規模を大きくしたり複雑にする予定がある場合は、ActionScript 2.0 とクラスを使用することをお勧めします。これによってアプリケーションを簡単に更新および変更できます。

ActionScript と Flash Player について

ActionScript 2.0 を使用した SWF ファイルを、[パブリッシュ設定] を [Flash Player 6] と [ActionScript 1.0] に設定してコンパイルすると、コードは ActionScript 2.0 クラスを使用しない限りにおいては動作します。大文字と小文字はコードでは区別されず、Flash Player によってのみ区別されます。したがって、[パブリッシュ設定] に [Flash Player 7] 以降と [ActionScript 1.0] を設定して SWF ファイルをコンパイルすると、大文字と小文字が区別されます。

[パブリッシュ設定] に [ActionScript 2.0] を設定して Flash Player 7 以降を対象にコンパイルする場合は、コンパイル時に型注釈 (厳密な型指定) が適用されます。

アプリケーションをパブリッシュすると、ActionScript 2.0 は ActionScript 1.0 のバイトコードにコンパイルされます。そのため、ActionScript 2.0 を使用しながら、Flash Player 6 以降を対象にすることができます。

第 3 章

データおよびデータ型

3

この章と、後続のいくつかの章では、ActionScript の基本概念の概要および実例を示します。基本的なコーディング手法を練習しながら、複雑なアプリケーションを作成する方法を学習していきます。この章では、FLA ファイルでのデータの操作方法と、操作できるデータの種類についても説明します。次の章である第 4 章の「シンタックスと言語の基礎」では、ActionScript のシンタックスの使用方法とステートメントの構成方法について解説します。その後の第 5 章の「関数とメソッド」では、ActionScript 言語での関数およびメソッドの使用方法を示します。

データとデータ型の詳細については、以下のセクションを参照してください。

データについて 33
データ型について 34
変数について 47
オブジェクト内のデータの整理 69
キャストについて 71

データについて

" データ " とは、Flash で操作できる数値、ストリング、およびその他の情報を指します。通常、アプリケーションや Web サイトを作成する場合は、データの使用が不可欠です。高度なグラフィックを作成する場合や、スクリプトによってアニメーションを生成する場合も、データを使用します。その際、効果を生み出すために使用している値を操作することが必要な場合があります。

データは、Flash 内で " 変数 " として定義することも、XML、Web サービス、ActionScript のビルトインクラスなどを使用して外部のファイルやサイトからロードすることもできます。データベースにデータを保存しておき、その情報を SWF ファイルでさまざまに表現することができます。たとえば、情報をテキストフィールドまたはコンポーネントに表示することや、イメージをムービークリップインスタンスで表示することができます。

最も一般的なデータの種類は、ストリング (名前やテキスト節など、文字が連続したもの)、数値、オブジェクト (ムービークリップなど)、ブール値 (`true` および `false`) などです。この章では、Flash で使用されるデータ型とその使用方法についても説明します。

データ型の詳細については、34 ページの「データ型について」を参照してください。変数の詳細については、47 ページの「変数について」を参照してください。

データ型について

" データ型 " とは、個々のデータの種類と、そのデータに対してどのような操作を実行できるかを定義するものです。データは変数に保存します。変数、オブジェクトインスタンス、および関数定義を作成するときには、データ型を使用して、扱うデータの型を割り当てます。ActionScript を記述する際は、多数の異なるデータ型を使用します。

ActionScript 2.0 では、一般的に使用されるデータ型がいくつか定義されています。データ型は、変数や ActionScript エレメントに格納できる値の種類を表します。データ型が割り当てられた変数には、そのデータ型の値セット内の値だけを格納できます。変数の詳細については、47 ページの「変数について」を参照してください。

ActionScript には、アプリケーション作成に必要となる基本的なデータ型が多数実装されています。詳細については、35 ページの「プリミティブデータ型および複合データ型について」の表を参照してください。

また、ActionScript は、Array や Date といったコアクラスを備えています。これらのコアクラスは、複合データ型またはリファレンスデータ型のように扱うことができます。複合データ型およびリファレンスデータ型の詳細については、35 ページの「プリミティブデータ型および複合データ型について」を参照してください。さらに、すべてのデータ型およびクラスの詳細については、『ActionScript 2.0 リファレンスガイド』を参照してください。

アプリケーション用にカスタムクラスを作成することもできます。class 宣言を使用して定義したクラスも、データ型のように扱うことができます。コアクラスおよび他のビルトインクラスの詳細については、237 ページの「トップレベルクラスとビルトインクラスについて」を参照してください。カスタムクラスの作成の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

ActionScript 2.0 では、変数の宣言時に、データ型を割り当てることができます。割り当てるデータ型は、コアデータ型でも、自分で作成したカスタムクラスでもかまいません。詳細については、42 ページの「データ型の割り当てと厳密な型指定について」を参照してください。

スクリプトをデバッグするときには、その動作を理解するために、式や変数のデータ型を特定することが必要となる場合があります。その際には、`instanceof` 演算子および `typeof` 演算子を使用します。詳細については、46 ページの「データ型の判定について」を参照してください。

あるデータ型を実行時に別のデータ型に変換することも可能です。その際には変換関数として、`Array()`、`Boolean()`、`Number()`、`Object()`、`String()` のいずれかを使用します。

アプリケーションにおけるデータ型の使用法を示す "datatypes.fla" ファイルのサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/DataTypes" フォルダに移動してサンプルにアクセスします。

プリミティブデータ型および複合データ型について

さまざまなデータ型の値はすべて、" プリミティブ " または " 複合 " の 2 つの大きなカテゴリのいずれかに分類されます。

"プリミティブ値" (プリミティブデータ型) とは、ActionScript が最も低い抽象度で保存する値です。つまり、プリミティブデータ型に対する操作は、複合データ型に対して行う操作よりも一般的に高速で効率的です。Boolean (ブール)、null (ヌル)、Number (数値)、String (ストリング)、および undefined (未定義) の各データ型はすべて、プリミティブ値のセットを定義します。

" 複合値 " (複合データ型) とは、プリミティブ値ではなく、プリミティブ値を参照する値です。" リファレンス " データ型とも呼ばれます。複合値は、Object データ型、または Object データ型に基づく他のデータ型に属します。複合値のセットを定義するデータ型には、Array (配列)、Date (日付)、Error (エラー)、Function (関数)、XML などがあります。これらの複合データ型の詳細については、『ActionScript 2.0 リファレンスガイド』の各項目を参照してください。

状況によって、プリミティブ型のデータを含む変数と複合型のデータを含む変数で、動作が異なることがあります。詳細については、67 ページの「プロジェクトでの変数の使用」を参照してください。

ActionScript には、次のように、アプリケーションで使用できる基本的なデータ型が実装されています。

データ型	説明
Boolean	プリミティブ。Boolean データ型は、`true` と `false` の 2 つの値から成ります。この型の変数に有効な値は、この 2 つだけです。宣言されていて、初期化されていない Boolean 型変数のデフォルト値は `false` です。詳細については、36 ページの「ブール (Boolean) データ型」を参照してください。
MovieClip	複合。MovieClip データ型では、MovieClip クラスのメソッドを使用して、ムービークリップシンボルを制御できます。詳細については、37 ページの「MovieClip データ型」を参照してください。
null	プリミティブ。null データ型の値は `null` です。この値は、値なし、つまりデータがないことを表します。`null` 値の代入は、プロパティや変数に値が割り当てられていないことを示すために、さまざまな状況で使用されます。null データ型は、複合データ型を定義するすべてのクラスでのデフォルトのデータ型です。ただし、Object クラスは例外で、`undefined` がデフォルトです。詳細については、39 ページの「ヌル (null) データ型」を参照してください。
Number	プリミティブ。このデータ型は、整数、符号なし整数、および浮動小数点数を表すことができます。浮動小数点数を格納するには、数値に小数点を含める必要があります。小数点を含まない場合、数値は整数として保存されます。Number データ型は、`Number.MAX_VALUE` (非常に大きい) ～ `Number.MIN_VALUE` (非常に小さい) の範囲の値を格納できます。詳細については、『ActionScript 2.0 リファレンスガイド』および 39 ページの「数値 (Number) データ型」を参照してください。

データ型	説明
Object	複合。Object データ型は Object クラスによって定義されます。Object クラスは、ActionScript のすべてのクラス定義に対する基本クラスとして機能し、オブジェクトの内部に別のオブジェクト (ネストされたオブジェクト) を配置することができます。詳細については、40 ページの「オブジェクト (Object) データ型」を参照してください。
String	プリミティブ。String データ型は、16 ビット文字 (英字、数字、区切り記号など) が連続したものを表します。ストリングは、UTF-16 形式を使用して、Unicode 文字として格納されます。String 値に対する操作は、そのストリングの新しいインスタンスを返します。詳細については、41 ページの「ストリング (String) データ型」を参照してください。
undefined	プリミティブ。undefined データ型の値は `undefined` だけです。これは、Object クラスのインスタンスのデフォルト値です。Object クラスに属する変数には、`undefined` という値のみを割り当てることができます。詳細については、42 ページの「未定義 (undefined) データ型」を参照してください。
Void	複合。Void データ型の値は `void` だけです。このデータ型は、値を返さない関数を指定する場合に使用します。Void は、プリミティブの Void データ型を参照する複合データ型です。詳細については、42 ページの「Void データ型」を参照してください。

"datatypes.fla" ファイルのサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/DataTypes" フォルダに移動してサンプルにアクセスします。

ブール (Boolean) データ型

ブール値は `true` か `false` のいずれかの値です。また、状況に応じて、値 `true` は 1 に、`false` は 0 に変換されます。ブール値は、比較することによりスクリプトのフローを制御する論理演算子と共に使用されます。

次の例では、テキストファイルを SWF ファイルにロードし、テキストファイルが正しくロードされなかった場合は [出力] パネルにメッセージを表示し、正しくロードされた場合はパラメータを表示します。詳細については、コード例のコメントを参照してください。

```
var my_lv:LoadVars = new LoadVars();
//success はブール値
my_lv.onLoad = function(success:Boolean) {
  //success が true の場合は monthNames をトレース
  if (success) {
    trace(my_lv.monthNames);
  //success が false の場合はメッセージをトレース
  } else {
    trace("unable to load text file");
  }
};
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

次の例では、TextInput コンポーネントのインスタンスに入力された 2 つの値を検査しています。userNameEntered と isPasswordCorrect という 2 つのブール型変数を作成し、両方の変数が true に評価された場合は、String 変数 titleMessage に "Welcome" というメッセージを代入します。

```
// 2 つの TextInput コンポーネント、1 つの Label コンポーネント、および 1 つの Button コ
   ンポーネントをステージに追加
// この 3 つのコンポーネントインスタンスに厳密なデータ型を指定
var userName_ti:mx.controls.TextInput;
var password_ti:mx.controls.TextInput;
var submit_button:mx.controls.Button;
var welcome_lbl:mx.controls.Label;

// ラベルを非表示
welcome_lbl.visible = false;

//Button コンポーネントで使用するリスナーオブジェクトを作成
// ボタンがクリックされたら、ユーザー名とパスワードをチェック
var btnListener:Object = new Object();
btnListener.click = function(evt:Object) {
   // TextInput のインスタンスに 1 文字以上入力されていることを確認
   // ブール値 (true/false) を返す
   var userNameEntered:Boolean = (userName_ti.text.length > 0);
   var isPasswordCorrect:Boolean = (password_ti.text == "vertigo");
   if (userNameEntered && isPasswordCorrect) {
      var titleMessage:String = "Welcome " + userName_ti.text + "!";
      welcome_lbl.text = titleMessage;
      // ラベルを表示
      welcome_lbl.visible = true;
   }
};
submit_button.addEventListener("click", btnListener);
```

詳細については、169 ページの「Flash での関数の使用」および 151 ページの「論理演算子について」を参照してください。

MovieClip データ型

ムービークリップは Flash アプリケーションでアニメーションを再生するためのシンボルです。グラフィックエレメントを参照するデータ型はムービークリップだけです。MovieClip データ型では、MovieClip クラスのメソッドを使用して、ムービークリップシンボルを制御できます。

MovieClip クラスのメソッドを呼び出す際には、コンストラクタを使用しません。ステージ上にムービークリップのインスタンスを作成することも、動的にインスタンスを作成することもできます。その後、ドット (.) 演算子を使用して、MovieClip クラスのメソッドを呼び出すだけです。

ステージ上のムービークリップを操作　次の例は、ステージ上の異なるムービークリップインスタンスについて、startDrag() メソッドと getURL() メソッドを呼び出しています。

```
my_mc.startDrag(true);
parent_mc.getURL("http://www.adobe.com/support/" + product);
```

2 番目の例は、ステージ上の my_mc というムービークリップの幅を返します。ムービークリップをターゲットインスタンスとし、数値を戻り値として返します。

```
function getMCWidth(target_mc:MovieClip):Number {
  return target_mc._width;
}
trace(getMCWidth(my_mc));
```

動的にムービークリップを作成　ステージ上に手動でムービークリップを作成したり、ライブラリからムービークリップを取得して割り当てるのが面倒な場合は、ActionScript を使用して、動的にムービークリップを作成することができます。たとえば、多数のサムネールイメージを含むイメージギャラリーを作成し、ステージ上で管理することができます。MovieClip.createEmptyMovieClip() を使用すると、ActionScript だけを使用してアプリケーションを作成できます。

ムービークリップを動的に作成するには、次の例に示すように、MovieClip.createEmptyMovieClip() を使用します。

```
// コンテナを保持するムービークリップを作成する。
this.createEmptyMovieClip("image_mc", 9);
// イメージを image_mc にロードする。
image_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

2 番目の例では、Drawing API を使用して矩形を描画する square_mc というムービークリップを作成しています。矩形をドラッグできるようにするため、MovieClip クラスのイベントハンドラ、startDrag() メソッド、および stopDrag() メソッドが追加されています。

```
this.createEmptyMovieClip("square_mc", 1);
square_mc.lineStyle(1, 0x000000, 100);
square_mc.beginFill(0xFF0000, 100);
square_mc.moveTo(100, 100);
square_mc.lineTo(200, 100);
square_mc.lineTo(200, 200);
square_mc.lineTo(100, 200);
square_mc.lineTo(100, 100);
square_mc.endFill();
square_mc.onPress = function() {
  this.startDrag();
};
square_mc.onRelease = function() {
  this.stopDrag();
};
```

詳細については、301 ページ、第 10 章の「ムービークリップの操作」、および『ActionScript 2.0 リファレンスガイド』の「MovieClip」の項目を参照してください。

ヌル (null) データ型

ヌルデータ型の値は null だけです。この値は " 値なし "、つまりデータがないことを表します。null 値の代入は、プロパティや変数にまだ値が割り当てられていないことを示すために、さまざまな状況で使用されます。たとえば、次のような状況で null 値を代入します。

- 変数は存在するが、まだ値が設定されていないことを示す場合
- 変数は存在するが、保持する値がなくなっていることを示す場合
- 関数から返される値がなかったことを示す、関数の戻り値として
- パラメータが省略されていることを示す、関数に対するパラメータとして

いくつかのメソッドや関数は、値が設定されていない場合に null を返します。フォームフィールドに現在フォーカスが存在するかどうかを、null を使用してテストする例を次に示します。

```
if (Selection.getFocus() == null) {
  trace("no selection");
}
```

数値 (Number) データ型

数値データ型とは、倍精度浮動小数点数のことです。数値オブジェクトの最小値は約 5e-324、最大値は、約 1.79E+308 です。

数値は、加算 (+)、減算 (-)、乗算 (*)、除算 (/)、剰余 (%)、インクリメント (++)、およびデクリメント (--) の算術演算子を使用して計算することができます。詳細については、145 ページの「数値演算子の使用」を参照してください。

また、ビルトインクラスである Math クラスおよび Number クラスのメソッドを使用して数値を計算することも可能です。これらのクラスのメソッドおよびプロパティについては、『ActionScript 2.0 リファレンスガイド』の「Math」および「Number」の項目を参照してください。

次の例では、Math クラスの sqrt() (平方根) メソッドを使用して、数値 100 の平方根を返します。

```
Math.sqrt(100);
```

次の例では、10 以上 17 以下のランダムな整数をトレース出力します。

```
var bottles:Number = 0;
bottles = 10 + Math.floor(Math.random() * 7);
trace("There are " + bottles + " bottles");
```

次の例では、intro_mc ムービークリップが何パーセント読み込まれたかを調べ、それを整数で表します。

```
var percentLoaded:Number = Math.round((intro_mc.getBytesLoaded() /
  intro_mc.getBytesTotal()) * 100);
```

オブジェクト (Object) データ型

オブジェクトはプロパティの集まりです。" プロパティ " とは、オブジェクトを説明する属性です。たとえば、オブジェクト (ムービークリップなど) の透明度は、オブジェクトの外観を説明する属性です。したがって、_alpha (透明度) はプロパティです。プロパティにはそれぞれ名前と値があります。プロパティには、Flash で使用可能なあらゆるデータ型の値を指定できます。また、オブジェクトデータ型でも問題ありません。これにより、オブジェクトの内部に別のオブジェクトを配置する (オブジェクトを " ネスト " させる) ことができます。

オブジェクトとプロパティを指定するには、ドット (.) 演算子を使用します。たとえば、次のコードでは、hoursWorked は employee のプロパティである weeklyStats のプロパティです。

```
employee.weeklyStats.hoursWorked
```

ActionScript MovieClip オブジェクトには、ステージでムービークリップシンボルインスタンスを制御するためのメソッドが用意されています。次の例では、play() メソッドと nextFrame() メソッドが使用されています。

```
mcInstanceName.play();
mc2InstanceName.nextFrame();
```

また、カスタムオブジェクトを作成して、Flash アプリケーション内の情報をまとめることもできます。ActionScript を使用してアプリケーションにインタラクティブ機能を追加するには、さまざまな情報が必要になります。たとえば、ユーザー名、年齢、電話番号、ボールの速度、ショッピングカート内のアイテム名、ロードされるフレーム数、ユーザーが最後に押したキーなどの情報です。カスタムオブジェクトを作成することにより、これらの情報をグループにまとめ、スクリプトの作成を簡略化し、スクリプトを再利用することができます。

次の ActionScript コードは、カスタムオブジェクトを使用して情報を整理する例です。この例では、user という新しいオブジェクトを作成し、ストリングデータ型および数値データ型の 3 つのプロパティ、name、age、および phone を作成しています。

```
var user:Object = new Object();
user.name = "Irving";
user.age = 32;
user.phone = "555-1234";
```

詳細については、214 ページの「例 : カスタムクラスの記述」を参照してください。

ストリング (String) データ型

ストリングとは、アルファベット、数字、区切り記号などの文字が連続したものです。ActionScript のステートメントにストリングを入力する場合は、一重引用符 ('')、または二重引用符 ("") で囲みます。

一般に、ストリング型を使用する場合は、変数に文字列を割り当てます。たとえば、次のステートメントは、"L7" という文字列を変数 favoriteBand_str に割り当てています。

```
var favoriteBand_str:String = "L7";
```

加算演算子 (+) を使用して、2 つのストリングを " 連結 " (結合) できます。ストリングの先頭、または末尾に入力されたスペースは、ストリングのリテラル部分として扱われます。次の式には、カンマの後にスペースが含まれています。

```
var greeting_str:String = "Welcome, " + firstName;
```

ストリング内で引用符を使用する場合は、引用符の前に円記号 (\) を付けます。円記号は " エスケープ " 文字と呼ばれます。他にも、特別なエスケープシーケンスがなければ ActionScript で記述できない文字があります。ActionScript で使用されるエスケープ文字は、次の表のとおりです。

エスケープシーケンス	文字
\b	バックスペース文字 (ASCII 8)
\f	改ページ文字 (ASCII 12)
\n	改行文字 (ASCII 10)
\r	復帰文字 (ASCII 13)
\t	タブ文字 (ASCII 9)
\"	二重引用符
\'	一重引用符
\\	円記号
\000 - \377	8 進法で表した 1 バイト
\x00 - \xFF	16 進法で表した 1 バイト
\u0000 - \uFFFF	16 進法で表した 16 ビットの Unicode 文字

ActionScript のストリングは、Java の場合と同様に不変です。ストリングに変更を加える操作では、常に新しいストリングが返されます。

String クラスは、ActionScript のビルトインクラスです。String クラスのメソッドおよびプロパティについては、『ActionScript 2.0 リファレンスガイド』の「String」の項目を参照してください。

未定義 (undefined) データ型

未定義データ型の値は undefined の 1 つだけであり、コードやユーザー操作の結果、値が割り当てられなかった変数に対して自動的に代入されます。

開発者が変数やプロパティに対して undefined を割り当てるのではありません。undefined 値は null とは異なり、自動的に代入されます。変数に値が設定されているかどうか、または変数が定義されているかどうかを検査する場合に未定義データ型を使用します。このデータ型を使用すると、アプリケーションの動作中にだけ実行されるコードを作成できます。その例を次に示します。

```
if (init == undefined) {
  trace("initializing app");
  init = true;
}
```

このコードは、アプリケーションに複数のフレームがあっても最初の 1 回しか実行されません。1 回実行すると、init 変数が未定義 (undefined) でなくなるためです。

Void データ型

Void データ型の値は void の 1 つだけであり、関数定義内で、値を返さない関数であることを表す場合に使用します。その例を次に示します。

```
// 戻り値の型が Void の関数を作成
function displayFromURL(url:String):Void {}
```

データ型の割り当てと厳密な型指定について

Flash では、コード内で値を保持するために変数を使用します。変数を作成するときに、型を明示的に宣言することができます。これを " 厳密な型指定 " と呼びます。

数値やストリングなどのデータ型を定義していないアイテムについては、実行時に値を代入する際に、Flash Player によってデータ型の特定が試みられます。次の例のように変数に値を代入すると、実行時に Flash Player によって演算子の右側のエレメントが評価され、その値が数値データ型であると判断されます。

```
var x = 3;
```

x には、厳密な型指定での宣言がなされていないため、コンパイラはこの変数のデータ型を特定することができません。コンパイラにとっては、x はあらゆるデータ型をとりうる変数です。詳細については、43 ページの「データ型の割り当て」を参照してください。後で別の値が割り当てられたときに、x の型が変更されることもあります。たとえば、x = "hello" というステートメントで割り当てると、x の型はストリングに変わります。

ActionScript では、厳密な型指定がなされていない変数のデータ型の変換が式で必要になると、プリミティブデータ型 (Boolean、Number、String、null、undefined など) が自動的に変換されます。

厳密な型指定を行うことによって、コンパイル時にいくつかの利点が生まれます。データ型の宣言 (厳密な型指定) は、コンパイル時にコード内のエラーを回避または診断するために役立ちます。厳密な型指定を使用して変数を宣言するには、次の形式を使用します。

```
var variableName:datatype;
```

メモ	厳密な型指定は、変数の " 強力な型指定 " とも呼ばれます。

データ型の不一致があるとコンパイルエラーが発生するので、コンパイル時にコード内のバグを見つけ、既存の変数に誤った型のデータを代入するのを防ぐことができます。また、厳密な型指定を行うことによって、オーサリング作業時に、ActionScript エディタでコードヒントがアクティブになります (ビジュアルエレメントの場合は、インスタンス名の接尾辞が必要です)。

厳密なデータ型指定を使うと、変数に対して不適切な型の値をうっかり代入してしまうのを防ぐことができます。Flash では、コンパイル時に型の不一致エラーをチェックし、不適切な型の値が使用されている場合はエラーメッセージを表示します。したがって、厳密な型指定により、オブジェクトの型に該当しないプロパティやメソッドにアクセスしてしまうことを防ぐことができます。厳密な型指定を行うことで、オブジェクトに関するコードヒントが ActionScript エディタに自動的に表示されるようになります。

変数の作成の詳細については、47 ページの「変数について」を参照してください。変数の命名の詳細については、52 ページの「変数の命名について」を参照してください。データ型の割り当て、および割り当て可能な型の詳細については、43 ページの「データ型の割り当て」を参照してください。

アプリケーションにおけるデータ型の使用法を示す "datatypes.fla" ファイルのサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/DataTypes" フォルダに移動してサンプルにアクセスします。

データ型の割り当て

変数を定義する際は、必ずデータ型を割り当てます。`var` キーワードを使用した変数宣言、関数の引数の作成、関数の戻り値の設定、または `for` ループや `for..in` ループで使用する変数の定義のいずれを行う場合も、データ型の割り当てが必要です。データ型を割り当てるには、" ポストコロンシンタックス " を使用します。つまり、変数名の後にコロン (:) を付け、その後にデータ型を指定します。

```
var my_mc:MovieClip;
```

データ型にはさまざまな種類があります。数値、ストリング、ブールなどのネイティブのデータ型の他に、BitmapData や FileReference などの Flash Player 8 に実装されているビルトインクラス、さらには開発者が作成したカスタムクラスもあります。最も一般的に使用されるデータ型は、数値、ストリング、ブール、配列、オブジェクトなどのビルトインデータ型です。次のコード例では、これらのデータ型を示します。

アイテムに特定のデータ型を割り当てるには、次の例のように、var キーワードと、変数名の後にコロン (:) を付けるポストコロンシンタックスを使って型を指定します。

```
// 変数またはオブジェクトの厳密な型指定
var myNum:Number = 7;
var birthday:Date = new Date();

// パラメータの厳密な型指定
function welcome(firstName:String, age:Number) {
}

// パラメータと戻り値の厳密な型指定
function square(myNum:Number):Number {
  var squared:Number = myNum * myNum;
  return squared;
}
```

オブジェクトのデータ型は、ビルトインクラス (Button や Date など) や、自分で作成したクラスおよびインターフェイスを基にして宣言することもできます。たとえば、"Student.as" というファイルの中で Student という自作のクラスを定義している場合、次のように記述すれば、作成するオブジェクトの型を Student と指定できます。

```
var myStudent:Student = new Student();
```

たとえば、次のコードを入力したとします。

```
// "Student.as" クラスファイル内で次のように記述
class Student {
  public var status:Boolean; // Student オブジェクトのプロパティ
}

// FLA ファイル内で次のように記述
var studentMaryLago:Student = new Student();
studentMaryLago.status = "enrolled"; /* 代入ステートメントでタイプが一致しません。
   String が見つかりましたが、Boolean が必要です。*/
```

Flash でこのスクリプトをコンパイルすると、ブール値以外の値が代入されているため、" タイプが一致しません " というエラーが発生します。

戻り値のない関数を記述する場合は、その関数の戻り値の型として Void を指定できます。また、関数へのショートカットを作成する場合は、新しい変数に Function というデータ型を割り当てることができます。オブジェクトが Function 型または Void 型であることを指定する方法については、次の例を参照してください。

```
function sayHello(name_str:String):Void {
   trace("Hello, " + name_str);
}
sayHello("world"); // Hello, world
var greeting:Function = sayHello;
greeting("Augustus"); // Hello, Augustus
```

厳密な型指定のもう 1 つのメリットは、ビルトインオブジェクトを厳密に型指定した場合、自動的にコードヒントが表示されることです。詳細については、42 ページの「データ型の割り当てと厳密な型指定について」を参照してください。

ActionScript 1.0 を使用してパブリッシュされたファイルは、コンパイル時に厳密な型指定を無視します。つまり、厳密に型指定した変数に誤った型の値を代入しても、コンパイルエラーは発生しません。

```
var myNum:String = "abc";
myNum = 12;
/* ActionScript 1.0 ではエラーが発生しないが、ActionScript 2.0 では " タイプが一致しませ
   ん " エラーが発生 */
```

これは、ActionScript 1.0 のパブリッシュ時に Flash は `var myNum:String = "abc"` のようなステートメントを厳密な型指定ではなく、スラッシュシンタックスと解釈するためです。ActionScript 2.0 は、スラッシュシンタックスをサポートしていません。この結果、不適切な型の変数にオブジェクトが代入されることになって、不正なメソッド呼び出しや未定義のプロパティ参照がコンパイラでエラーとならずに通ってしまう可能性があります。

ActionScript 2.0 を使用してパブリッシュしたファイルでは、必要に応じてデータの型指定を使用できます。このため、コードに厳密な型指定を実装する場合は、パブリッシュ設定を ActionScript 2.0 に設定する必要があります。パブリッシュ設定を指定し、パブリッシュするファイルの ActionScript のバージョンを定義するには、メインメニューからパブリッシュ設定を変更するか ([ファイル]-[パブリッシュ設定])、インスタンスを何も選択していない状態でプロパティインスペクタの [設定] ボタンをクリックします。ActionScript の特定のバージョンまたは Flash Player を使用するには、[パブリッシュ設定] ダイアログボックスで [Flash] タブを選択し、[ActionScript のバージョン] ポップアップメニューでアイテムを選択します。

型チェックの詳細については、45 ページの「型チェックについて」を参照してください。

型チェックについて

" 型チェック " とは、変数の型と式の互換性を確認する作業のことです。Flash では、変数に対して指定した型と、その変数に代入した値が一致しているかどうかをチェックします。厳密な型指定とデータ型の割り当ての詳細については、42 ページの「データ型の割り当てと厳密な型指定について」および 43 ページの「データ型の割り当て」を参照してください。

型チェックは、コンパイル時または実行時のいずれかの時点で行われます。厳密な型指定を使用すると、コンパイル時に型チェックが行われます。ActionScript は型指定を動的に行う言語なので、実行時にも型チェックが行われる場合があります。

たとえば、次のコードでは、パラメータ `xParam` の型を指定していません。実行時には、このパラメータが数値型の値を保持するために使用され、さらに `String` 型の値を保持するために使用されます。その後、`dynamicTest()` 関数は、`typeof` 演算子を使用して、このパラメータがストリング型か数値型かをテストします。

```
function dynamicTest(xParam) {
  if (typeof(xParam) == "string") {
    var myStr:String = xParam;
    trace("String: " + myStr);
  } else if (typeof(xParam) == "number") {
    var myNum:Number = xParam;
    trace("Number: " + myNum);
  }
}
dynamicTest(100);
dynamicTest("one hundred");
```

データ型情報を ActionScript に明示的に追加する必要はありません。ActionScript コンパイラでは、プロパティを使用して、コンパイル時に存在しないメソッドを呼び出すことを許可しています。このため、プロパティを作成したり、実行時にメソッドを動的に割り当てたりすることが可能です。

動的な型チェックによって提供される柔軟性の一例として、コンパイル時には未知のプロパティやメソッドの使用が挙げられます。これにより、コードの制約が緩和されるので、コーディングの状況によっては利点が生まれます。たとえば、次のコードでは runtimeTest() という関数を記述していますが、この関数が呼び出すメソッドと返すプロパティは、いずれもコンパイラにとっては未知のものです。このコードではコンパイル時エラーは発生しませんが、実行時にこのプロパティまたはメソッドにアクセスできない場合は、実行時エラーが発生します。

```
function runtimeTest(myParam) {
  myParam.someMethod();
  return myParam.someProperty;
}
```

データ型の判定について

プログラムのテストやデバッグの際には、さまざまなアイテムのデータ型に関係するものと思われる問題が見つかることがあります。また、データ型が明示的に割り当てられていない変数を使用する場合に、変数のデータ型を調べる方法があると便利なことがあります。ActionScript では、アイテムのデータ型を判定することができます。typeof 演算子を使用すると、データに関する情報が返されます。

データ型を取得するには、typeof 演算子を使用します。ただし、typeof では、インスタンスがどのクラスに属するかという情報は取得できません。

次の例は、typeof 演算子を使用して、トレースするオブジェクトの種類を取得する方法を示しています。

```
// LoadVars クラスの新しいインスタンスを作成
var my_lv:LoadVars = new LoadVars();

/* typeof 演算子ではクラスを判定できないので、my_lv がオブジェクトであることしかわからない
  */
var typeResult:String = typeof(my_lv);
trace(typeResult); // オブジェクト
```

この例では、myName という新しいストリング型変数を作成し、それを数値データ型に変換しています。

```
var myName:String = new String("17");
trace(myName instanceof String); // true
var myNumber:Number = new Number(myName);
trace(myNumber instanceof Number); // true
```

これらの演算子の詳細については、『ActionScript 2.0 リファレンスガイド』の「typeof operator」および「instanceof operator」を参照してください。テストとデバッグの詳細については、『Flash ユーザーガイド』を参照してください。継承とインターフェイスの詳細については、253 ページ、第 7 章の「継承」を参照してください。クラスの詳細については、179 ページ、第 6 章の「クラス」を参照してください。

変数について

" 変数 " とは、情報を格納する容器のようなものです。次の ActionScript は、ActionScript 内で変数がどのように表されているかを示しています。

```
var myVariable:Number = 10;
```

この変数には数値が格納されます。上のコードの :Number は、変数に格納される値の型を指定しています。これを " データ型指定 " と呼びます。データ型指定の詳細については、42 ページの「データ型の割り当てと厳密な型指定について」および 43ページの「データ型の割り当て」を参照してください。

容器 (変数名で表される) そのものが ActionScript 内で変化することはありませんが、その内容 (値) は変化します。スクリプトでは、変数の値を何回でも変更できます。SWF ファイルの再生時に変数の値を変更すると、ユーザーが実行した操作に関する情報を記録して保存できます。また、SWF ファイルの再生中に変化した値を記録したり、条件が true なのか false なのかを評価することもできます。SWF ファイルの再生中に変数の更新を頻繁に行う必要がある場合があります。たとえば、Flash ゲームでプレーヤーの得点が変更された場合の変数の更新がそうです。SWF ファイルでユーザーの対話操作を作成したり処理したりする場合は、変数が不可欠です。

変数を最初に宣言するときに、変数に値を割り当てておくことをお勧めします。初期値を割り当てることを値の " 初期化 " と呼び、通常は、タイムラインのフレーム 1 で、または SWF ファイルの再生開始時にロードされるクラス内で行われます。変数にはさまざまな種類があり、スコープの影響を受けます。変数の種類とスコープの詳細については、57 ページの「変数とスコープについて」を参照してください。

ヒント	変数の初期化は、SWF ファイルの再生時に変数の値を記録して比較するために必要です。

メモ	初期化されていない変数の評価は、Flash Player 7 以降と Flash Player 6 以前とで異なります。Flash Player6 用に作成されたスクリプトを Flash Player 7 以降に移植することを検討している場合、この違いを理解しておかないと、予想外の動作が生じる場合があります。

変数には、さまざまな型のデータを格納できます。詳細については、34 ページの「データ型について」を参照してください。変数に格納されるデータの型によって、スクリプトで変数の値を割り当てたときに、その値がどのように変化するかが異なります。

変数には、一般的に、URL (String 型)、ユーザー名 (String 型)、数値演算の結果 (Number 型)、イベントの発生回数 (Number 型)、ユーザーが特定のボタンをクリックしたかどうか (Boolean 型) などの情報が格納されます。各 SWF ファイルとオブジェクトインスタンス (ムービークリップなど) には変数のセットがあり、各変数には、他の SWF ファイルやムービークリップの変数とは独立して値が割り当てられます。

変数の値を確認するには、`trace()` ステートメントを使用して、[出力] パネルに値を送信します。その後、テスト環境で SWF ファイルをテストすると、[出力] パネルに値が表示されます。たとえば、`trace(hoursWorked)` により、変数 `hoursWorked` の値が、テスト環境の [出力] パネルに送信されます。テスト環境では、デバッガで変数値のチェックと設定もできます。

変数の詳細については、次のトピックを参照してください。

- 48 ページの「変数の宣言について」
- 49 ページの「値の割り当てについて」
- 52 ページの「変数の命名について」
- 53 ページの「アプリケーションでの変数の使用」
- 57 ページの「変数とスコープについて」
- 49 ページの「デフォルト値について」
- 51 ページの「演算子と変数について」
- 61 ページの「変数のロードについて」
- 67 ページの「プロジェクトでの変数の使用」

変数の宣言について

変数は、タイムライン内の 1 つのフレームで宣言することも、オブジェクトで直接宣言することもできます。また、外部のクラスファイルで宣言することもできます。

`var` キーワードを使用し、変数命名規則に従って変数を定義します。次の例では、`firstName` という変数を宣言しています。

```
var firstName:String;
```

変数を宣言するときには、変数にデータ型を割り当てます。この例では、`firstName` 変数に String データ型を割り当てています。データ型の割り当ての詳細については、42 ページの「データ型の割り当てと厳密な型指定について」を参照してください。

デフォルト値について

" デフォルト値 " とは、値を設定する前に変数に格納されている値です。変数に初めて値を設定する場合は、変数を " 初期化 " します。変数を宣言して、値を設定しない場合、その変数は初期化されません。初期化されていない変数の値は、デフォルトで undefined になります。変数の作成および使用の詳細については、47 ページの「変数について」を参照してください。

値の割り当てについて

" 値 " とは、変数の現在の内容です。値には、ストリング、数値、配列、オブジェクト、XML、日時、または独自に作成したカスタムクラスを使用できます。Flash では、var キーワードを使用して変数を宣言します。変数を宣言する際には、データ型の割り当ても行います。また、値を変数に割り当てることもできます (変数のデータ型に適合する値である必要があります)。

次の例は、catName という変数の作成方法を示しています。

```
var catName:String;
```

変数を宣言した後、変数に値を割り当てることができます。たとえば、上記の ActionScript の行の後に、次に示す行を記述します。

```
catName = "Pirate Eye";
```

メモ	Pirate Eye はストリングなので、引用符 (") で囲む必要があります。

この例では、Pirate Eye という値を catName 変数に割り当てています。上記の例のように変数の宣言の後で変数に値を割り当てる代わりに、変数を宣言するときに変数に値を割り当てることもできます。たとえば、次の例のように、catName 変数を宣言するときに、この変数の値を設定できます。

```
var catName:String = "Pirate Eye";
```

テスト環境で catName 変数の値を表示するには、trace() ステートメントを使用します。このステートメントは、値を [出力] パネルに送信します。次の ActionScript を使用して catName 変数の値をトレースすると、実際の値には引用符が含まれていないことを確認できます。

```
var catName:String = "Pirate Eye";
trace(catName); // Pirate Eye
```

変数に割り当てる値は、変数のデータ型 (この場合は String) と一致している必要があります。後で catName = 10 のように catName 変数に数値を割り当てようとすると、SWF ファイルをテストしたときに [出力] パネルに次のようなエラーが表示されます。

```
Type mismatch in assignment statement: found Number where String is required.
```

このエラーは、指定した変数に不適切なデータ型を設定しようとしたことを示しています。

変数に数値を割り当てる場合は、次のコードに示すように、引用符は必要ありません。

```
var numWrinkles:Number = 55;
```

コードで後から numWrinkles の値を変更する場合は、次の ActionScript を使用して新しい値を割り当てます。

```
numWrinkles = 60;
```

既存の変数に値を再割り当てする場合は、var キーワードの使用も、変数のデータ型 (この場合は、:Number) の定義も必要ありません。

値が数値またはブール値 (true または false) の場合、引用符は使用しません。次のコードに数値およびブール値の例を示します。

```
var age:Number = 38;
var married:Boolean = true;
var hasChildren:Boolean = false;
```

上記の例では、変数 age には、10 進数でない整数値が格納されていますが、10 進数値または 38.4 のような浮動小数点値を使用することもできます。ブール型変数 (married や hasChildren など) の有効な値は、true または false の 2 つだけです。

配列を作成して、それに値を割り当てる場合は、次のコードに示すように、形式が少し異なります。

```
var childrenArr:Array = new Array("Pylon", "Smithers", "Gil");
```

配列アクセス演算子を使用して配列を作成するための代替 (簡易) シンタックスがあります。このシンタックスでは、角カッコ ([]) を使用します。上記の例を次のように書き換えることができます。

```
var childrenArr:Array = ["Pylon", "Smithers", "Gil"];
```

配列の作成および配列アクセス演算子の詳細については、123 ページの「配列について」および 80 ページの「ドットシンタックスを使用したインスタンスのターゲット化について」を参照してください。

同様に、myObj という新しいオブジェクトを作成することができます。新しいオブジェクトを作成するには、次のいずれかの方法を使用します。配列を記述する 1 つ目の方法 (長い方式) は次のとおりです。

```
var myObj:Object = new Object();
myObj.firstName = "Steve";
myObj.age = 50;
myObj.childrenArr = new Array("Mike", "Robbie", "Chip");
```

myObj 配列を記述する 2 つ目の方法 (簡易方式) は次のとおりです。

```
var myObj:Object = {firstName:"Steve", age:50, childrenArr:["Mike", "Robbie",
   "Chip"]};
```

この例を見るとわかるように、簡易方式を使用すると、入力作業と時間をかなり節約することができます。オブジェクトのインスタンスを定義する場合などは特にそうです。チームで作業する場合、またはインターネットや書籍などで見つけたサードパーティの ActionScript コードを使用する場合は、この代替シンタックスを目にすることがあるので、これに慣れておくことが大切です。

メモ

すべての変数を明示的に定義する必要はありません。一部の変数は Flash によって自動的に作成されます。たとえば、ステージのサイズを調べる場合は、`Stage.width` および `Stage.height` という 2 つの定義済みの変数の値を使用できます。

演算子と変数について

コード内の数学記号を不思議に思うかもしれません。ActionScript では、これらの記号を " 演算子 " と呼びます。演算子は 1 つまたは複数の値から新しい値を計算します。また、コード内で変数に値を割り当てるときにも演算子を使用します。変数に値を割り当てるには、代入演算子 (=) を使用します。

```
var username:String = "Gus";
```

もう 1 つの例として、加算演算子 (+) があります。加算演算子は複数の値を加算して新しい値を生成するために使用します。複数のストリング値に対して + 演算子を使用した場合は、ストリングが連結されます。演算子を使用した演算の対象となる値は、" オペランド " と呼ばれます。

値を割り当てる場合、演算子を使用して変数に値を定義します。たとえば、次のスクリプトでは、代入演算子を使用して変数 `numChildren` に 7 という値を割り当てています。

```
var numChildren:Number = 7;
```

`numChildren` 変数の値を変更する場合は、次のコードを使用します。

```
numChildren = 8;
```

メモ

この変数は既に定義されているので、`var` を使用する必要はありません。

ActionScript での演算子の使用の詳細については、135 ページの「演算子について」を参照してください。

変数の命名について

変数に名前を付ける際は注意が必要です。変数には、ほとんど自由に名前を付けることができますが、いくつかの規則があります。変数の名前は、次の規則に従って付ける必要があります。

- 変数は識別子である必要があります。

メモ	" 識別子 " とは、変数、プロパティ、オブジェクト、関数、またはメソッドの名前です。識別子の先頭の 1 文字は、英字、アンダースコア (_)、ドル記号 ($) のいずれかである必要があります。それ以降の文字としては、英字、数字、アンダースコア、ドル記号のいずれかを使用できます。

- 変数には、キーワードや ActionScript のリテラル (`true`、`false`、`null`、または `undefined`) は使用できません。リテラルの詳細については、93 ページの「リテラルについて」を参照してください。
- 変数は、スコープ内で一意である必要があります。詳細については、57 ページの「変数とスコープについて」を参照してください。
- 変数は、ActionScript 言語のどのエレメント (クラス名など) とも同じであってはなりません。

変数に名前を付ける際にこれらの規則に従わないと、シンタックスエラーや予期しない結果が生じる場合があります。次の例では、変数に `new` という名前を付けて、ドキュメントをテストすると、コンパイルエラーが発生します。

```
// このコードは意図したとおりに動作する
var helloStr:String = new String();
trace(helloStr.length); // 0
// ビルトインクラスと同じ名前を変数に付ける
var new:String = "hello"; // エラー : 識別子が必要です
var helloStr:String = new String();
trace(helloStr.length); // undefined
```

ActionScript エディタは、ビルトインクラスおよびそれらのクラスを基にした変数に対するコードヒント表示をサポートしています。変数に割り当てた特定のオブジェクト型に対応するコードヒントを表示させるには、変数の型を厳密に指定します。コードヒントを使用してツールヒントスタイルのシンタックスヒントやポップアップメニューを表示すると、コードを簡単に記述できます。

たとえば、次のコードを入力します。

```
var members:Array = new Array();
members.
```

[アクション] パネルにピリオド (`.`) を入力するとすぐに、配列オブジェクトで利用できるメソッドとプロパティのリストが表示されます。

変数の命名に関する推奨コーディング規則については、650 ページの「変数の命名」を参照してください。

アプリケーションでの変数の使用

このセクションでは、ActionScript の短いコードで変数を使用します。変数は、式で使用する前に、あらかじめ宣言して初期化しておく必要があります。式は、オペランドと演算子を組み合わせて 1 つの値を表します。たとえば、式 i+2 では、i と 2 がオペランドであり、+ が演算子です。

初期化していない変数を式で使用すると、変数が定義されていないために、予期しない結果が生じる場合があります。式の記述の詳細については、75 ページ、第 4 章の「シンタックスと言語の基礎」を参照してください。

Flash Player 7 以降では、次の例のように、定義していない変数を使用すると、変数値が NaN となり、予期しない結果が生じる場合があります。

```
var squared:Number = myNum * myNum;
trace(squared); // NaN
var myNum:Number = 6;
```

次の例では、変数 myNum を宣言および初期化するステートメントが前にあるので、squared を値で置き換えることができます。

```
var myNum:Number = 6;
var squared:Number = myNum * myNum;
trace(squared); // 36
```

次に示すように、未定義の変数をメソッドや関数に渡した場合にも、同様の動作になります。

関数に未定義の変数を渡した場合と定義済みの変数を渡した場合を比較するには :

1. Button コンポーネントを、[コンポーネント] パネルからステージへドラッグします。
2. プロパティインスペクタを開き、[インスタンス名] テキストボックスに「bad_button」と入力します。
3. タイムラインのフレーム 1 に次のコードを入力します。

```
// 正常に動作しない
function badClickListener(evt:Object):Void {
  getURL(targetUrl);
  var targetUrl:String = "http://www.adobe.com";
}
bad_button.addEventListener("click", badClickListener);
```

4. [制御]-[ムービープレビュー] を選択すると、ボタンが機能しない (Web ページが開かれない) ことがわかります。
5. 別の Button コンポーネントをステージにドラッグします。そのボタンを選択します。
6. プロパティインスペクタを開き、[インスタンス名] テキストボックスに「good_button」と入力します。

7. タイムラインのフレーム 1 の、この前に追加した ActionScript の後に、次の ActionScript を追加します。

```
// 正常に動作する
function goodClickListener(evt:Object):Void {
  var targetUrl:String = "http://www.adobe.com";
  getURL(targetUrl);
}
good_button.addEventListener("click", goodClickListener);
```

8. [制御]-[ムービープレビュー] を選択して、ステージに追加した 2 番目のボタンをクリックします。

   Web ページが正常に開かれます。

変数の値がいつ、どのように変更されるかは、変数に格納されるデータの型によって決まります。ストリングや数値などのプリミティブデータ型は、" 値渡し " になります。つまり、変数の参照ではなく、その変数の現在の値が使用されます。複合データ型の例には、配列データ型およびオブジェクトデータ型があります。

次の例では、myNum を 15 に設定し、その値を otherNum にコピーします。コードの 3 行目で myNum を 30 に変更しても、otherNum は myNum の値を参照しないので、otherNum の値は 15 のままです。otherNum 変数には、コードの 2 行目で受け取った myNum の値が格納されています。

ActionScript で変数を使用するには :

1. 新しい Flash ドキュメントを作成し、var_example.fla という名前で保存します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
var myNum:Number = 15;
var otherNum:Number = myNum;
myNum = 30;
trace(myNum); // 30
trace(otherNum); // 15
```

   コードの 3 行目で myNum を 30 に変更しても、otherNum は myNum の値を参照しないので、otherNum の値は 15 のままです。otherNum 変数には、コードの 2 行目で受け取った myNum の値が格納されています。

3. [制御]-[ムービープレビュー] を選択して、[出力] パネルに表示される値を確認します。
4. 手順 2 で追加したコードの後に、次の ActionScript を追加します。

```
function sqr(myNum:Number):Number {
  myNum *= myNum;
  return myNum;
}
var inValue:Number = 3;
var outValue:Number = sqr(inValue);
trace(inValue); // 3
trace(outValue); // 9
```

このコードでは、変数 inValue にはプリミティブ値 3 が格納されているので、この値が sqr() 関数に渡され、9 という値が返されます。関数の myNum の値が変わっても、変数 inValue の値は変わりません。

5. [制御]-[ムービープレビュー] を選択して、[出力] パネルに表示される値を確認します。

オブジェクトデータ型では、複雑な情報を大量に格納することができますが、実際の値が格納されるのではなく、値への参照が保持されています。参照は、変数の内容を指し示すエイリアスのようなものです。変数の値が必要になると、その値が参照され、変数に直接値が代入されることなく、参照結果が返されます。

参照渡しによる変数の受け渡しについては、55 ページの「参照渡しによる変数の受け渡し」を参照してください。

参照渡しによる変数の受け渡し

配列データ型およびオブジェクトデータ型では、実際の値が格納されるのではなく、値への参照が保持されているので、配列やオブジェクトを扱う場合は注意が必要です。

次の例は、参照によりオブジェクトを渡す方法を示しています。配列のコピーを作成する場合、実際には、配列の内容への参照 (エイリアス) のコピーだけが作成されます。2 番目の配列の内容を編集すると、最初の配列と 2 番目の配列は同じ値を指しているので、最初と 2 番目の両方の配列の内容が変更されます。

参照渡しでオブジェクトを渡すには :

1. [ファイル]-[新規]-[Flash ドキュメント] を選択して新しい FLA ファイルを作成し、copybyref.fla という名前で保存します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
var myArray:Array = new Array("tom", "josie");
var newArray:Array = myArray;
myArray[1] = "jack";
trace(myArray); // tom,jack
trace(newArray); // tom,jack
```

3. [制御]-[ムービープレビュー] を選択して ActionScript をテストします。

   この ActionScript では、2 つのエレメントを持つ配列オブジェクト myArray が作成されます。変数 newArray を作成し、myArray への参照を渡します。myArray の 2 番目のエレメント (配列の添え字は 0 から始まるので、myArray[1] は 2 番目を指します) を "jack" に変更すると、このエレメントを参照しているすべての変数が影響を受けます。trace() ステートメントは、tom,jack を [出力] パネルに送ります。

メモ	Flash では、ゼロベースのインデックスが使用されます。つまり、配列内の第 1 のエレメントは 0 であり、第 2 のエレメントは 1 です (以下同様)。

次の例では、myArray は配列オブジェクトを含んでいるため、関数 zeroArray( ) にはこの配列を参照渡しで渡します。関数 zeroArray( ) は配列オブジェクトをパラメータとして受け取り、この配列の全エレメントを 0 に設定します。配列は参照渡しで渡されているので、関数による配列の変更が可能です。

参照渡しで配列を渡すには :

1. [ファイル]-[新規]-[Flash ドキュメント] を選択して新しい FLA ファイルを作成し、arraybyref.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
function zeroArray (theArr:Array):Void {
  var i:Number;
  for (i = 0; i < theArr.length; i++) {
    theArr[i] = 0;
  }
}

var myArr:Array = new Array();
myArr[0] = 1;
myArr[1] = 2;
myArr[2] = 3;
trace(myArr); // 1,2,3
zeroArray(myArr);
trace(myArr); // 0,0,0
```

3. [制御]-[ムービープレビュー] を選択して ActionScript をテストします。

   この ActionScript 内の最初の trace( ) ステートメントは、myArray 配列の元の内容 (1,2,3) を表示します。zeroArray( ) 関数を呼び出して myArray 配列への参照を渡すと、配列の各値が上書きされてゼロに設定されます。その次の trace( ) ステートメントは、myArray 配列の新しい内容 (0,0,0) を表示します。値渡しではなく参照渡しで配列を渡したので、zeroArray( ) 関数から配列の更新された内容を戻す必要はありません。

配列の詳細については、123 ページの「配列について」を参照してください。

変数とスコープについて

変数のスコープとは、変数が有効 (定義済み) であり、その参照が可能な範囲のことです。変数が有効な範囲が、特定のタイムライン内または関数内である場合もあれば、変数がアプリケーション全体でグローバルに有効な場合もあります。スコープの詳細については、85 ページの「スコープとターゲット化について」を参照してください。

ActionScript を使用して Flash アプリケーションを開発する場合、変数のスコープを理解することが大切です。スコープは、いつ、どこで変数を参照できるかを示すだけではなく、特定の変数がアプリケーション内に存在する期間も示します。関数の本体で変数を定義する場合、その特定の関数が終了すると同時に、変数は消滅します。不適切なスコープ内でオブジェクトを参照しようとしたり、無効になった変数を参照しようとすると、Flash ドキュメントでエラーが発生して、予期しない動作に生じたり、機能が動作しなくなることがあります。

ActionScript の変数のスコープには、次の 3 種類があります。

- グローバル変数と関数は、ドキュメント内のどのタイムラインやスコープでも参照できます。したがって、グローバル変数はコードのすべての範囲で定義されています。
- タイムライン変数は、そのタイムラインにある任意のスクリプトで利用できます。
- ローカル変数は、その変数が宣言された関数の本体 (中カッコで囲まれた範囲内) だけで使用できます。したがって、ローカル変数は、コード内の該当する部分に対してのみ定義されます。

スコープおよび変数の使用に関するガイドラインについては、85 ページ、第 4 章の「スコープとターゲット化について」を参照してください。

メモ

自分で作成した ActionScript 2.0 クラスでは、パブリック、プライベート、および静的の各変数スコープを使用できます。詳細については、203 ページの「クラスメンバーについて」および 224 ページの「クラスでのメンバーアクセスの制御」を参照してください。

グローバル変数の厳密な型指定を行うことはできません。詳細と回避策については、58 ページの「グローバル変数」を参照してください。

グローバル変数

グローバル変数と関数は、ドキュメント内のどのタイムラインやスコープでも参照できます。グローバルスコープの変数を宣言 (作成) するには、変数名の前に _global という識別子を付け、var = シンタックスは使用しません。たとえば、次のコードでは、グローバル変数 myName が作成されます。

```
var _global.myName = "George"; // このグローバル変数のシンタックスは誤り
_global.myName = "George"; // グローバル変数の正しいシンタックス
```

ただし、次の例のように、グローバル変数と同じ名前でローカル変数を初期化した場合、そのローカル変数のスコープ内にとどまっている間はグローバル変数へのアクセスはできません。

```
_global.counter = 100; // グローバル変数を宣言
trace(counter); // グローバル変数にアクセスして "100" を表示
function count():Void {
  for (var counter:Number = 0; counter <= 2; counter++) { // ローカル変数
    trace(counter); // ローカル変数にアクセスして "0" 〜 "2" を表示
  }
}
count();
trace(counter); // グローバル変数にアクセスして "100" を表示
```

この例は、count( ) 関数のスコープでは、グローバル変数がアクセスされないことのみを示しています。しかし、_global というプレフィックスを付けると、グローバルスコープの変数にアクセスできます。たとえば、次のコードのように、counter に _global というプレフィックスを付けると、グローバルスコープの counter にアクセスできます。

```
trace(_global.counter);
```

データ型を割り当てるときは var キーワードを使用する必要があるので、_global スコープで作成した変数に厳密な型指定を割り当てることはできません。たとえば、次のように指定することはできません。

```
_global.foo:String = "foo";  // シンタックスエラー
var _global.foo:String = "foo"; // シンタックスエラー
```

Flash Player バージョン 7 以降では、別のセキュリティドメインから読み込んだ SWF ファイルでグローバル変数にアクセスする場合、Sandbox セキュリティ機能による制限が適用されます。詳細については、609 ページ、第 16 章の「セキュリティについて」を参照してください。

タイムライン変数

タイムライン変数は、その特定のタイムラインにある任意のスクリプトで利用できます。タイムライン変数を宣言するには、var ステートメントを使用し、タイムライン内の任意のフレームで初期化します。次の例のように、タイムライン変数はそのフレーム以降のすべてのフレームで利用可能です。

ドキュメント内でタイムライン変数を使用するには :

1. 新しい Flash ドキュメントを作成し、timelinevar.fla という名前を付けます。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var myNum:Number = 15; /* フレーム 1 で初期化されるので、すべてのフレームで利用可能 */
```

3. タイムラインのフレーム 20 を選択します。
4. [挿入]-[タイムライン]-[空白キーフレーム] を選択します。
5. 新規キーフレームを選択した状態で、[アクション] パネルに次の ActionScript を入力します。

```
trace(myNum);
```

6. [制御]-[ムービープレビュー] を選択して新しいドキュメントをテストします。

   約 1 秒後に、[出力] パネルに値 15 が表示されます。デフォルトでは Flash ドキュメントはループするので、再生ヘッドがタイムラインのフレーム 20 に達するたびに、値 15 が [出力] パネルに連続的にトレース出力されます。ループ処理を止めるには、trace() ステートメントの後に stop(); を追加します。

タイムライン変数は、スクリプト内でアクセスする前に宣言する必要があります。たとえば、フレーム 20 にコード var myNum:Number = 15; を置いた場合、フレーム 20 より前のフレームに割り当てられたスクリプトは myNum にアクセスでないため、値 15 は格納されずに未定義になります。

ローカル変数

関数ブロック内で var ステートメントを使用すると、" ローカル変数 " の宣言になります。関数ブロック (" 関数定義 " とも呼ばれます) 内でローカル変数を宣言すると、ローカル変数は関数ブロックのスコープ内で定義され、その関数ブロックの最後で無効になります。したがって、ローカル変数は関数の中だけで存在します。

たとえば、変数 myStr を関数 localScope の中で宣言すると、この変数は関数の外部では利用できません。

```
function localScope():Void {
  var myStr:String = "local";
}
localScope();
trace(myStr); // myStr はグローバルに定義されていないので未定義
```

ローカル変数に使用した変数名が既にタイムライン変数として宣言されている場合、ローカル変数がスコープ内にある間は、ローカル定義がタイムライン定義よりも優先されます。ただし、タイムライン変数は関数の外でも存在しています。たとえば、次のコードではタイムラインストリング変数 str1 を作成した後、同じ名前のローカル変数を scopeTest() 関数内で作成しています。関数内の trace ステートメントはこの変数のローカル定義を出力しますが、関数の外の trace ステートメントはこの変数のタイムライン定義を出力します。

```
var str1:String = "Timeline";
function scopeTest():Void {
  var str1:String = "Local";
  trace(str1); // ローカル
}
scopeTest();
trace(str1); // タイムライン
```

次の例では、特定の変数は特定の関数の生存期間のみ有効で、その変数を関数のスコープ外で参照しようとするとエラーが発生することを確認できます。

アプリケーションでローカル変数を使用するには :

1. 新しい Flash ドキュメントを作成します。
2. [アクション] パネルを開き ([ウィンドウ]-[アクション])、タイムラインのフレーム 1 に次の ActionScript を追加します。

   ```
   function sayHello(nameStr:String):Void {
     var greetingStr:String = "Hello, " + nameStr;
     trace(greetingStr);
   }
   sayHello("world"); // Hello, world
   trace(nameStr); // undefined
   trace(greetingStr); // undefined
   ```

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   [出力] パネルに “Hello, world” というストリングが表示されます。また、nameStr と greetingStr の各値については、これらの変数が現在のスコープでは利用可能でないため、undefined と表示されます。nameStr と greetingStr は、sayHello 関数の実行時にのみ参照できます。この関数が終了すると、これらの変数は消滅します。

変数 i および j はループカウンタとしてよく使用されます。次の例では、i をローカル変数として使用します。つまり、この変数は initArray() 関数内にのみ存在します。

```
var myArr:Array = new Array();
function initArray(arrayLength:Number):Void {
  var i:Number;
  for(i = 0; i < arrayLength; i++) {
    myArr[i] = i + 1;
  }
}
```

```
trace(myArr); // <空白>
initArray(3);
trace(myArr); // 1,2,3
trace(i); // undefined
```

メモ

`for (var i:Number = 0; i < arrayLength; i++) {...}` という for ループのシンタックスもよく使用されます。

この例では、変数 i がメインタイムラインで定義されていないので、Flash のテスト環境では undefined と表示されます。この変数は、initArray() 関数内にのみ存在します。

ローカル変数を使用すると、アプリケーションで予期しない結果が発生する原因となる名前のコンフリクトを防ぐことができます。たとえば、age をローカル変数として使用すると、あるコンテキストではこの変数にある人の年齢を格納し、別のコンテキストでは、その人の子供の年齢を格納することができます。変数を異なるスコープで使用しているので、この状況ではコンフリクトは起こりません。

関数の本体ではローカル変数を使用し、関数が独立したコードとして実行されるようにすることをお勧めします。ローカル変数は、その変数が使用されているコードのブロック内でのみ変更可能です。関数内の式でグローバル変数が使用されていると、関数の外部のコードやイベントでその変数の値を変更できてしまうので、結果として関数が異なった動作をする可能性があります。

宣言時に、ローカル変数にデータ型を割り当てることにより、誤ったデータ型の値が既存の変数に代入されるのを防ぐことができます。詳細については、42 ページの「データ型の割り当てと厳密な型指定について」を参照してください。

変数のロードについて

以後のセクションでは、サーバーから変数をさまざまな方法でロードします。HTML コード内の URL ストリングまたは FlashVars (FlashVars を使用すると Flash に変数を渡すことができます) からドキュメントに変数をロードします。これらの手法は、SWF ファイルの外部の変数をする方法が複数あることを示しています。

変数 (名前と値のペアなど) をロードする方法の詳細については、569 ページ、第 15 章の「外部データの操作」を参照してください。

SWF ファイルでは、変数の用途に応じて、さまざまな方法で変数を使用できます。詳細については、次のトピックを参照してください。

- 62 ページの「URL から取得した変数の使用」
- 64 ページの「アプリケーションでの FlashVars の使用」
- 66 ページの「サーバーからの変数のロード」

URL から取得した変数の使用

Flash でアプリケーションや簡単なサンプルなどを開発する場合、HTML ページから Flash ドキュメントに値を渡す必要が生じることがあります。渡される値は、" クエリーストリング " または "URL エンコードされた変数 " と呼ばれることがあります。たとえば、Flash でメニューを作成する場合に URL 変数を使用すると、正しいナビゲーションをデフォルトで表示するようにメニューを初期化することなどができて便利です。また、Flash でイメージビューアを構築し、Web サイトに表示するデフォルトのイメージを定義することもできます。

ドキュメント内で URL 変数を使用するには :

1. Flash ドキュメントを作成し、urlvariables.fla という名前を付けます。
2. [ファイル]-[名前を付けて保存] を選択し、ドキュメントをデスクトップに保存します。
3. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
this.createTextField("myTxt", 100, 0, 0, 100, 20);
myTxt.autoSize = "left";
myTxt.text = _level0.myURL;
```

4. [制御]-[ムービープレビュー] を選択して、Flash Player で SWF ファイルをテストします。

   テキストフィールドに undefined と表示されます。次に進む前に、変数が適切に定義されていることを確認するには、その変数が存在するかどうかを Flash でチェックする必要があります。変数が存在するかどうかは、変数が定義されているかどうかをチェックすることによって確認できます。

5. 変数が定義されているかどうかをチェックするには、手順 3 で [アクション] パネルに追加した ActionScript を、次のコードに合わせて変更します。**ボールド**で示した部分が追加するコードです。

```
this.createTextField("myTxt", 100, 0, 0, 100, 20);
myTxt.autoSize = "left";
if (_level0.myURL == undefined) {
  myTxt.text = "myURL is not defined";
} else {
  myTxt.text = _level0.myURL;
}
```

   Flash ドキュメントをパブリッシュすると、デフォルトで SWF ファイルと同じディレクトリに HTML ドキュメントが作成されます。HTML ファイルが作成されない場合は、[ファイル]-[パブリッシュ設定] を選択して、[形式] タブで HTML が選択されていることを確認します。その後、再びドキュメントをパブリッシュします。

   次のコードは、作成された HTML ドキュメント内で、HTML ページに Flash ドキュメントを埋め込む処理を行っている部分を示しています。次の手順で URL 変数にその他のコードを追加する場合の URL 変数の動作を理解するには、この HTML コードを参照する必要があります。

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" codebase="http:/
  /fpdownload.adobe.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400" id="urlvariables"
  align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="urlvariables.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="urlvariables.swf" quality="high" bgcolor="#ffffff" width="550"
  height="400" name="urlvariables" align="middle"
  allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
  pluginspage="http://www.adobe.com/go/getflashplayer_jp" />
</object>
```

6. 生成された HTML ドキュメントから Flash ドキュメントに変数を渡すには、パスとファイル名 ("urlvariables.swf") の後に変数を渡します。デスクトップに作成された HTML ファイルに、**ボールドのテキスト**を追加します。

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" codebase="http:/
  /fpdownload.adobe.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400" id="urlvariables"
  align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="urlvariables.swf?myURL=http://
  weblogs.macromedia.com" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="urlvariables.swf?myURL=http://weblogs.macromedia.com"
  quality="high" bgcolor="#ffffff" width="550" height="400"
  name="urlvariables" align="middle" allowScriptAccess="sameDomain"
  type="application/x-shockwave-flash" pluginspage="http://www.adobe.com/go/
  getflashplayer_jp" />
</object>
```

7. 複数の変数を Flash に渡す場合は、名前と値の各ペアをアンパサンド (&) で区切る必要があります。手順 6 のコードの中から次のコードを探します。

```
?myURL=http://weblogs.macromedia.com
```

それを次のテキストで置き換えます。

```
?myURL=http://weblogs.macromedia.com&myTitle=adobe+News+Aggregator
```

すべてのブラウザでの一貫性を保つには、object タグと embed タグの両方に同じ変更を加える必要があります。各単語は + 記号で区切られています。各単語がこのように区切られているのは、値が URL エンコードされているためであり、+ 記号は 1 つの空白スペースを表します。

メモ

URL エンコード特殊文字の一覧は、Flash テクニカルノート「URL エンコーディング : テキストファイルからの特殊文字の読み込み」を参照してください。

アンパサンド (&) は、名前と値の各ペアの区切り記号として機能するので、渡す値にアンパサンドが含まれていると、予期しない結果が生じることがあります。たとえば、次のような値を Flash に渡すと、名前と値のペアおよび解析の性質により問題が発生します。

```
my.swf?name=PB+&+J&flavor=strawberry+rhubarb
```

この結果としては、次の変数 (および値) がルートのスコープに作成されます。

```
'name': 'PB ' (note space at end of value)
' J': '' (note space at beginning of variable name and an empty value)
'flavor': 'strawberry rhubarb'
```

これを回避するには、名前と値のペアに含まれているアンパサンド (&) 文字を、等価な URL エンコード (%26) を使用して " エスケープ " する必要があります。

8. "urlvariables.html" ドキュメントを開き、次のコードを探します。

```
?myURL=http://weblogs.macromedia.com&myTitle=Adobe+News+Aggregator
```

該当する部分を次のコードで置き換えます。

```
?myURL=PB+%26+J&flavor=strawberry+rhubarb
```

9. 変更した HTML ファイルを保存し、Flash ドキュメントを再度テストします。

今度は、次のような名前と値のペアが作成されます。

```
'name': 'PB & J'
'flavor': 'strawberry rhubarb'
```

メモ	ストリングサイズを 64KB (65535 バイト) までの長さであれば、すべてのブラウザでサポートされます。すべてのブラウザで動作するようにするには、object タグと embed タグの両方で FlashVars を割り当てる必要があります。

アプリケーションでの FlashVars の使用

FlashVars を使用して Flash に変数を渡すのは、HTML コードで URL と共に変数を渡すのと同様です。ただし、FlashVars の場合は、ファイル名の後で変数を渡すのではなく、個別の param タグおよび embed タグで変数を渡します。

ドキュメント内で FlashVars を使用するには :

1. 新しい Flash ドキュメントを作成し、myflashvars.fla という名前を付けます。
2. [ファイル]-[パブリッシュ設定] を選択し、[HTML] が選択されていることを確認したら、[OK] をクリックしてダイアログボックスを閉じます。

3. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createTextField("myTxt", 100, 0, 0, 100, 20);
myTxt.autoSize = "left";
if (_level0.myURL == undefined) {
  myTxt.text = "myURL is not defined";
} else {
  myTxt.text = _level0.myURL;
}
```

メモ	デフォルトでは、HTML コードは "myflashvars.fla" と同じ場所にパブリッシュされます。

4. [ファイル]-[パブリッシュ] を選択して、SWF ファイルおよび HTML ファイルをパブリッシュします。

5. パブリッシュされたファイルがあるディレクトリ (ハードディスク上で "myflashvars.fla" を保存した場所) を開き、Dreamweaver またはメモ帳などの HTML エディタで HTML ドキュメント (デフォルトでは "myflashvars.html") を開きます。

6. 次のコードと同様の内容になるように、**ボールド**で示した部分を HTML ドキュメントに追加します。

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.adobe.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400" id="myflashvars"
  align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="myflashvars.swf" />
<param name="FlashVars" value="myURL=http://weblogs.adobe.com/">
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="myflashvars.swf" FlashVars="myURL=http://weblogs.adobe.com/"
  quality="high" bgcolor="#ffffff" width="550" height="400"
  name="myflashvars" align="middle" allowScriptAccess="sameDomain"
  type="application/x-shockwave-flash" pluginspage="http://www.adobe.com/go/
  getflashplayer_jp" />
</object>
```

このコードは、ストリング http://weblogs.macromedia.com を含む myURL という 1 つの変数を渡します。SWF ファイルがロードされると、_level0 スコープに myURL というプロパティが作成されます。FlashVars を使用するか、または URL と共に変数を渡すことの利点の 1 つは、SWF ファイルがロードされた時点で変数が直ちに使用できることです。つまり、変数のロードが完了したかどうかをチェックするために関数を記述する必要がありません。LoadVars や XML を使用して変数をロードする場合は、そのような関数が必要になります。

7. 変更を保存して HTML ドキュメントを閉じます。

8. "myflashvars.html" をダブルクリックしてアプリケーションをテストします。

   HTML ファイル内の変数のテキスト `http://weblogs.macromedia.com` が SWF ファイル内に現れます。

メモ	ストリングサイズを 64KB (65,535 バイト) までの長さであれば、すべてのブラウザでサポートされます。すべてのブラウザで動作するようにするには、`object` タグと `embed` タグの両方で FlashVars を割り当てる必要があります。

サーバーからの変数のロード

外部ソース (テキストファイルや XML ドキュメントなど) から Flash に変数をロードする方法はいくつかあります。名前と値のペアなどの変数のロードの詳細については、569 ページ、第 15 章の「外部データの操作」を参照してください。

次の例に示すように、Flash では LoadVars クラスを使用して変数を簡単にロードできます。

サーバーから変数をロードするには :

1. 新しい Flash ドキュメントを作成します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を追加します。

```
var my_lv:LoadVars = new LoadVars();
my_lv.onLoad = function(success:Boolean):Void {
   if (success) {
      trace(this.dayNames); // Sunday,Monday,Tuesday,...
   } else {
      trace("Error");
   }
}
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

   このコードは、リモートサーバーからテキストファイルをロードして、その名前と値のペアを解析します。

ヒント	変数のフォーマットを調べるには、テキストファイル (http://www.helpexamples.com/flash/params.txt) をダウンロードするかブラウザで表示します。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   ファイルが正常にロードされると、`complete` イベントが呼び出され、[出力] パネルに `dayNames` の値が表示されます。テキストファイルをダウンロードできない場合、`success` 引数が `false` に設定され、[出力] パネル (または [出力] タブ) にテキスト `Error` が表示されます。

プロジェクトでの変数の使用

Flash でアニメーションやアプリケーションを作成する場合は、ほとんど必ず、変数の使用が必要になります。たとえば、ログインシステムを構築する場合、通常は、ユーザー名とパスワードが有効かどうか、またはそもそもそれらが入力されたかどうかを判断する変数が必要です。

変数 (名前と値のペアなど) をロードする方法の詳細については、569 ページ、第 15 章の「外部データの操作」を参照してください。

次の例では、Loader クラスでロードするイメージのパスを格納する変数、Loader クラスのインスタンス用の変数、およびファイルが正常にロードされたかどうかに応じて呼び出される関数を使用します。

プロジェクトで変数を使用するには :

1. 新しい Flash ドキュメントを作成し、imgloader.fla という名前で保存します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を追加します。

```
/* FlashVars を使用して値が渡されなかった場合のためのデフォルトイメージを指定 */
var imgUrl:String = "http://www.helpexamples.com/flash/images/image1.jpg";
if (_level0.imgURL != undefined) {
   // イメージが指定された場合はデフォルト値を上書き
   imgUrl = _level0.imgURL;
}

this.createEmptyMovieClip("img_mc", 10);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
   target_mc._x = (Stage.width - target_mc._width) / 2;
   target_mc._y = (Stage.height - target_mc._height) / 2;
}
mclListener.onLoadError = function(target_mc:MovieClip):Void {
   target_mc.createTextField("error_txt", 1, 0, 0, 100, 20);
   target_mc.error_txt.autoSize = "left";
   target_mc.error_txt.text = "Error downloading specified image;\n\t" +
   target_mc._url;
}
var myMCL:MovieClipLoader = new MovieClipLoader();
myMCL.addListener(mclListener);
myMCL.loadClip(imgUrl, img_mc);
```

コードの最初の行では、Flash ドキュメントに動的にロードするイメージを指定しています。次に、FlashVars または URL エンコードされた変数を使用して imgURL に新しい値が指定されたかどうかをチェックします。新しい値が指定された場合、デフォルトのイメージの URL は新しい値で上書きされます。URL 変数の使用の詳細については、62 ページの「URL から取得した変数の使用」を参照してください。FlashVars の詳細については、64 ページの「アプリケーションでの FlashVars の使用」を参照してください。

次の数行のコードでは、MovieClip インスタンス、および将来の MovieClipLoader インスタンスのための Listener オブジェクトを定義しています。MovieClipLoader の Listener オブジェクトは、2 つのイベントハンドラ `onLoadInit` および `onLoadError` を定義します。これらのハンドラは、イメージが正常にロードされて初期化された場合、またはイメージのロードが失敗した場合に呼び出されます。次に、MovieClipLoader インスタンスを作成し、`addListener()` メソッドを使用して、前に定義した Listener オブジェクトをこの MovieClipLoader インスタンスに追加します。最後に、ロードするイメージファイルとそのイメージをロードするターゲットムービークリップを指定する `MovieClipLoader.loadClip()` メソッドを呼び出すと、イメージがダウンロードおよびトリガされます。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   ここでは、オーサリングツールで Flash ドキュメントのテストをしているので、`imgUrl` の値は FlashVars によっても URL によっても渡されません。このため、デフォルトイメージが表示されます。

4. Flash ドキュメントを保存し、[ファイル]-[パブリッシュ] を選択して、ファイルを SWF および HTML ドキュメントとしてパブリッシュします。

メモ	[パブリッシュ設定] ダイアログボックスで、Flash と HTML の両方を選択する必要があります。[ファイル]-[パブリッシュ設定] を選択し、[形式] タブをクリックします。次に、両方のオプションを選択します。

5. Flash ツール ([制御]-[ムービープレビュー]) またはローカルブラウザ ([ファイル]-[パブリッシュプレビュー]-[HTML]) でドキュメントをテストすると、イメージの表示は縦方向、横方向共に中央揃えになります。

6. 生成された HTML ドキュメントを Dreamweaver やメモ帳などのエディタで編集し、デフォルトの HTML を次のテキストに合わせて変更します。

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.adobe.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400" id="imgloader"
  align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="imgloader.swf" />
<param name="FlashVars" value="imgURL=http://www.helpexamples.com/flash/
  images/image2.jpg">
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="imgloader.swf" quality="high" FlashVars="imgURL=http://
  www.helpexamples.com/flash/images/image2.jpg" bgcolor="#ffffff"
  width="550" height="400" name="imgloader" align="middle"
  allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
  pluginspage="http://www.adobe.com/go/getflashplayer" />
</object>
```

7. HTML ドキュメントをテストして変更を確認します。HTML コードで指定したイメージが SWF ファイル内に現れます。

   この例を編集して独自のイメージを使用するには、FlashVars 値 (二重引用符で囲まれたストリング) を変更する必要があります。

オブジェクト内のデータの整理

ステージ上に配置できる " オブジェクト " としては、たとえば MovieClip オブジェクトなどがありますが、そうしたオブジェクトの内部には他のムービークリップを含めることができます。テキストフィールド、ムービークリップ、およびボタンは、ステージ上に配置された場合、オブジェクトと呼ばれることもよくあります。

一方、ActionScript におけるオブジェクトとは、プロパティやメソッドの集まりです。オブジェクトにはそれぞれ独自の名前があり、各オブジェクトは特定のクラスの 1 つのインスタンスです。ビルトインオブジェクトとは、ActionScript であらかじめ定義されているクラスのオブジェクトです。たとえば、ビルトインクラス Date は、ユーザーのコンピュータのシステムクロックの情報を提供します。ビルトインクラス LoadVars を使用すると、SWF ファイルに変数をロードすることができます。

また、オブジェクトやクラスは ActionScript を使用して作成することもできます。人の名前、住所、電話番号などのデータの集まりを保持するオブジェクトを作成することができます。イメージのカラー情報を保持するオブジェクトを作成することもできます。オブジェクト内のデータを整理することは、Flash ドキュメントをより整然とした状態に保つために役立ちます。メソッドとプロパティの集まりを保持するカスタムクラスの作成の一般的な情報については、188 ページの「カスタムクラスファイルの記述」を参照してください。ビルトインクラスとカスタムクラスの詳細については、179 ページ、第 6 章の「クラス」を参照してください。

ActionScript には、オブジェクトを作成する方法が複数用意されています。次の例では、2 つの異なる方法で単純なオブジェクトを作成し、各オブジェクトの内容をループ処理します。

Flash で単純なオブジェクトを作成するには :

1. 新しい Flash ドキュメントを作成し、simpleObjects.fla という名前で保存します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を入力します。

```
// 第 1 の方法
var firstObj:Object = new Object();
firstObj.firstVar = "hello world";
firstObj.secondVar = 28;
firstObj.thirdVar = new Date(1980, 0, 1); // January 1, 1980
```

   このコードは、単純なオブジェクトを作成する方法の 1 つを示しています。ここでは、新しいオブジェクトインスタンスを作成し、オブジェクト内のいくつかのプロパティを定義しています。

3. 手順 2 で入力したコードの後に次の ActionScript を入力します。

```
// 第 2 の方法
var secondObj:Object = {firstVar:"hello world", secondVar:28, thirdVar:new
  Date(1980, 0, 1)};
```

   これは、オブジェクトを作成するもう 1 つの方法です。両方とも、オブジェクトは同じです。このコードは、新しいオブジェクトを作成し、オブジェクトの簡易記法を使用していくつかのプロパティを初期化しています。

4. 作成した各オブジェクトをループ処理し、各オブジェクトの内容を表示するために、タイムラインのフレーム 1 のこれまでに入力したコードの後に、次の ActionScript を追加します。

```
var i:String;
for (i in secondObj) {
  trace(i + ": " + secondObj[i]);
}
```

5. [制御]-[ムービープレビュー] を選択すると、[出力] パネルに次のテキストが表示されます。

```
firstVar: hello world
secondVar: 28
thirdVar: Tue Jan 1 00:00:00 GMT-0800 1980
```

メモ

変数は必ずしもこの順序で [出力] パネルに表示されるとは限りません。for..in ループを使用する場合、順序についての保証はありません。オブジェクトの内容は、プレーヤーから予期しない順序で返されます。

配列を使用してオブジェクトを作成することもできます。firstname1、firstname2、および firstname3 などの一連の変数で変数の集まりを表す代わりに、オブジェクトの配列を作成して同じデータを表すことができます。次にこの方法を示します。

配列を使用してオブジェクトを作成するには :

1. 新しい Flash ドキュメントを作成し、arrayObject.fla という名前で保存します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を入力します。

```
var usersArr:Array = new Array();
usersArr.push({firstname:"George"});
usersArr.push({firstname:"John"});
usersArr.push({firstname:"Thomas"});
```

配列やオブジェクトに変数をまとめて整理する利点は、次の手順に示すように、変数をループ処理してその値を表示することが非常に簡単になることです。

3. 手順 2 で追加した ActionScript の後に、次のコードを入力します。

```
var i:Number;
for (i = 0; i < usersArr.length; i++) {
  trace(usersArr[i].firstname); // George, John, Thomas
}
```

4. [制御]-[ムービープレビュー] を選択すると、[出力] パネルに次のテキストが表示されます。

```
George
John
Thomas
```

次の例では、オブジェクトをループ処理するもう 1 つの方法を示します。この例では、オブジェクトを作成し、for..in ループを使用してループ処理し、各プロパティを [出力] パネルに表示します。

```
var myObj:Object = {var1:"One", var2:"Two", var3:18, var4:1987};
var i:String;
for (i in myObj) {
  trace(i + ": " + myObj[i]);
}
// 出力は次のとおり
/*
    var1: One
    var2: Two
    var3: 18
    var4: 1987
*/
```

for ループの作成の詳細については、117 ページ、第 4 章の「for ループの使用」を参照してください。for...in ループの詳細については、118 ページの「for..in ループの使用」を参照してください。オブジェクトの詳細については、179 ページ、第 6 章の「クラス」を参照してください。

キャストについて

ActionScript 2.0 では、あるデータ型を別のデータ型にキャストできます。オブジェクトを異なる型にキャストするとは、オブジェクトや変数が保持する値を異なる型に変換することです。

型のキャストの結果は、使用されるデータ型によって異なります。オブジェクトを異なる型にキャストするには、オブジェクト名をカッコ (( )) で囲み、その前に新しい型の名前を置きます。たとえば、次のコードではブール値を整数にキャストします。

```
var myBoolean:Boolean = true;
var myNumber:Number = Number(myBoolean);
```

キャストの詳細については、次に示す各トピックを参照してください。

- 71 ページの「オブジェクトのキャストについて」

オブジェクトのキャストについて

キャストのシンタックスは type(item) です。これによって、コンパイラは、アイテム (item) のデータ型が type であるかのように動作します。キャストは、実質的には関数呼び出しです。実行時にキャストエラーが発生した場合は、null が返されます (キャストエラーが発生するのは、Flash Player 7 以降用にパブリッシュされたファイルだけです。Flash Player 6 用にパブリッシュされたファイルでは、実行時のキャストエラーはサポートされません)。キャストが成功すると、元のオブジェクトが返されます。しかし、実行時にキャストエラーが発生する場合でも、コンパイラでのコンパイル時には、実行時にキャストエラーが発生するかどうかを判断できず、エラーは生成されません。

その例を次のコードに示します。

```
// Cat クラスと Dog クラスはいずれも Animal クラスのサブクラス
function bark(myAnimal:Animal) {
   var foo:Dog = Dog(myAnimal);
   foo.bark();
}
var curAnimal:Animal = new Dog();
bark(curAnimal); // 正常に動作する
curAnimal = new Cat();
bark(curAnimal); // 正常に動作しない
```

この例では、コンパイラに対して foo が Dog オブジェクトであると指定しているため、コンパイラは foo.bark(); が正しいステートメントであると見なします。しかし、コンパイラは、このキャストが実行時に失敗すること、つまり Cat オブジェクトを Animal タイプにキャストしようとしていることは認識していないため、コンパイル時エラーは発生しません。しかし、次の例に示すように、キャストが正常に行われたことを確認するチェックをスクリプトに含めると、実行時にキャストエラーを検出できます。

```
function bark(myAnimal:Animal) {
  var foo:Dog = Dog(myAnimal);
  if (foo) {
    foo.bark();
  }
}
```

式をインターフェイスにキャストできます。その式が、インターフェイスを実装するオブジェクトまたはインターフェイスを実装する基本クラスを持つオブジェクトである場合、キャストは正常に行われます。それ以外の場合、キャストは失敗します。

注	null または undefined にキャストした場合は、undefined が返されます。

対応するグローバル変換関数を持つプリミティブデータ型を、同名のキャスト演算子でオーバーライドすることはできません。グローバル変換関数には、キャスト演算子よりも高い優先度が割り当てられているためです。たとえば、Array() 変換関数はキャスト演算子よりも高い優先度を持つため、Array にキャストすることはできません。

この例では、2 つのストリング変数 (firstNum および secondNum) を定義し、これらを加算します。これらはストリングデータ型なので、最初の結果では、数字は加算されるのではなく連結されます。2 番目の trace ステートメントは、両方の数字を数値データ型に変換してから、加算を実行します。これにより適切な結果が得られます。次の例に示すように、XML または FlashVars を使用してロードされるデータを使用する場合、データ変換は重要です。

```
var firstNum:String = "17";
var secondNum:String = "29";
trace(firstNum + secondNum); // 1729
trace(Number(firstNum) + Number(secondNum)); // 46
```

データ変換関数の詳細については、『ActionScript 2.0 リファレンスガイド』の「Array 関数」、「Boolean 関数」、「Number 関数」、「Object 関数」、「String 関数」の各変換関数の項目を参照してください。

第 4 章

シンタックスと言語の基礎

4

ActionScript のシンタックスとステートメントを学習することは、単語を文にまとめ、文を段落にまとめる方法を習得するようなものです。ActionScript は、簡単に組み立てることもできます。たとえば、英語では文章はピリオド (.) で終わりますが、ActionScript では、ステートメントの最後にはセミコロン (;) を入力します。ActionScript 言語では、`stop()` アクションを入力すると、ムービークリップインスタンスまたは SWF ファイルの再生ヘッドのループを停止することができます。または、数千行ものコードを記述して、インタラクティブなバンキングアプリケーションを作成することも可能です。このように、ActionScript では、非常に単純な処理から非常に複雑な処理まで実行することができます。

第 3 章の「データおよびデータ型」では、ActionScript 言語でどのようにデータが使用されるかと、コードでデータをフォーマットする方法について学習しました。この章では、" シンタックス " を使って ActionScript でステートメントを作成する方法について説明します。この章には、言語の基礎的な概念を示すために、多くの短いコードといくつかの例が含まれています。後の章では、この章で学習した基礎を組み合わせて応用し、より長く複雑なサンプルコードを示します。

このセクションでは、すべての ActionScript に適用される一般的な規則について説明します。ActionScript で使用する多くの用語には、個別の要件があります。具体的な用語の規則については、『ActionScript 2.0 リファレンスガイド』の各項目を参照してください。

ActionScript の基本を応用して洗練されたプログラムを作成することは、ActionScript の初級ユーザーにとって難しい場合があります。このセクションで取り上げた規則をどのように応用すればよいかについては、645 ページ、第 17 章の「ActionScript 2.0 のベストプラクティスとコーディング規則」を参照してください。

メモ	この章では、タイムライン上のフレームに直接 ActionScript を追加します。後の章では、クラスを使用して FLA ファイルから ActionScript を分離します。

ActionScript のシンタックスと言語の基礎を操作する際の詳細については、次のトピックを参照してください。

シンタックス、ステートメント、および式について 76
ドットシンタックスとターゲットパスについて 79
言語区切子について 87
定数およびキーワードについて 98
ステートメントについて 102
配列について 123
演算子について 135

シンタックス、ステートメント、および式について

ActionScript 言語は、ActionScript 言語を構成するビルトインクラスから成ります。コードがコンパイルされて Flash で正しく実行されるようなステートメントを作成するには、ActionScript の正しい " シンタックス " を使用する必要があります。この場合のシンタックスとは、プログラミングに使用する言語の文法およびスペルのことです。コンパイラは不適切なシンタックスを理解することができないので、テスト環境でこのようなドキュメントをテストしようとすると、[出力] パネルにエラーまたは警告が表示されます。したがって、シンタックスは正しい ActionScript を作成するためのルールとガイドラインの集まりです。

" ステートメント " は、特定のアクションなど、何かを実行するために FLA ファイルに与える指示です。たとえば、条件ステートメントを使用すると、何かが true であるか存在しているかを判断できます。次に、条件が true であるかどうかに基づいて、関数や式などの指定するアクションを実行することができます。if ステートメントは条件ステートメントであり、条件を評価して、コードで次に発生する必要があるアクションを判断します。

```
//if ステートメント
if (condition) {
  // ステートメント ;
}
```

ステートメントの詳細については、102 ページの「ステートメントについて」を参照してください。

" 式 " とは、ステートメントとは異なり、値を表す ActionScript シンボルを規則に従って組み合わせたものです。式には " 値 " があり、値とプロパティには " 型 " があります。式は、演算子とオペランド、値、関数、およびプロシージャで構成されます。また、優先順位および関連付けに関する ActionScript のルールに従っています。通常、Flash Player は式を解釈した後に、アプリケーションで使用できる値を返します。

式の例を次に示します。

```
x + 2
```

この式で、x および 2 はオペランドであり、+ は演算子です。演算子およびオペランドの詳細については、135 ページの「演算子について」を参照してください。オブジェクトとプロパティの詳細については、40 ページの「オブジェクト (Object) データ型」を参照してください。

ActionScript をどのようにフォーマットするかによって、コードが保守しやすいかどうかが決まります。たとえば、インデントやコメントがなく、フォーマットや命名規則にも一貫性のない FLA ファイルでは、ロジックを読み取ることは困難です。ActionScript のブロック (ループや `if` ステートメントなど) をインデントすることで、コードが読みやすくなり、問題が発生したときのデバッグが簡単になります。ActionScript のフォーマットの詳細については、675 ページの「ActionScript シンタックスのフォーマット」を参照してください。これらのセクションでも、ActionScript の正しいフォーマットについて参照することができます。

シンタックスと言語の基礎の詳細については、次のトピックを参照してください。

- 「ActionScript と JavaScript の違い」
- 「大文字と小文字の区別について」

ActionScript と JavaScript の違い

ActionScript は、基本的な部分は JavaScript プログラミング言語と似ています。ActionScript の使用および習得に、必ずしも JavaScript の経験は必要ありません。ただし、JavaScript の知識があれば、ActionScript をより効率よく学ぶことができるでしょう。

本マニュアルでは、一般的なプログラミングについては触れません。一般的なプログラミングの概念や JavaScript 言語の詳細に関する資料は豊富にあります。

- JavaScript を起源とする ECMAScript (ECMA-262) Edition 3 の言語仕様は、JavaScript 言語の国際標準の役割を担っています。ActionScript はこの仕様に準拠しています。詳細については、www.ecma-international.org/publications/standards/Ecma-262.htm を参照してください。
- Java のテクノロジーについて解説したサイト (http://java.sun.com/docs/books/tutorial/java/index.html) には、オブジェクト指向プログラミングに関するチュートリアルが用意されています。Java 言語を対象としていますが、その概念を ActionScript に応用することができます。

ActionScript と JavaScript の違いとして次の点が挙げられます。

- ActionScript は、ドキュメント、ウィンドウ、アンカーなど、ブラウザ指定のオブジェクトをサポートしません。
- ActionScript がサポートする JavaScript ビルトインオブジェクトは限られています。
- ActionScript は、ステートメントラベルなど、JavaScript のシンタックス構造をサポートしません。
- ActionScript では、`eval()` 関数は変数参照しか行えません。
- ActionScript 2.0 は、クラスや厳密なタイプ指定など、ECMA-262 仕様に含まれていない機能をいくつかサポートしています。これらの機能の多くは、ECMAScript (ECMA-262) Edition 3 の言語仕様 (www.ecma-international.org/publications/standards/Ecma-262.htm を参照) に基づいて設計されました。
- ActionScript では、RegExp オブジェクトを使用した正規表現はサポートされません。しかし、Adobe Central で、RegExp オブジェクトがサポートされています。Adobe Central の詳細については、www.adobe.com/products/central/ を参照してください。

大文字と小文字の区別について

Flash Player 7 以降を対象にして ActionScript を記述する場合、コードでは大文字と小文字が区別されます。したがって、大文字と小文字がわずかに異なる変数であっても、異なる変数として見なされます。次の ActionScript コードに、この例を示します。

```
// 大文字と小文字が混在する
var firstName:String = "Jimmy";
// すべて小文字を使用する
trace(firstname); // undefined
```

または、次のコードを記述することができます。

```
// Flash Player 8 をターゲットとしたファイルで、
// ActionScript 1.0 または ActionScript 2.0 を使用している場合
//
// 2 つの異なるオブジェクトのプロパティを設定する
cat.hilite = true;
CAT.hilite = true;

// 3 つの異なる変数を作成する
var myVar:Number = 10;
var myvar:Number = 10;
var mYvAr:Number = 10;
```

メモ

大文字と小文字を変更することで、変数または識別子を区別することはお勧めできません。変数の命名の詳細については、645 ページ、第 17 章の「ActionScript 2.0 のベストプラクティスとコーディング規則」を参照してください。

以前のバージョンの Flash Player (Flash Player 6 以前) を対象にしてパブリッシュを行う場合は、ストリング "Jimmy" がトレースされ、[出力] パネルに表示されます。Flash Player 7 以降のバージョンでは大文字と小文字が区別されるので、firstName および firstname は 2 つの異なる変数となります (ActionScript 1.0 または ActionScript 2.0 を使用する場合)。これは重要な概念なので必ず理解してください。Flash Player 6 以前を対象にして FLA ファイルを作成し、変数で大文字と小文字が一致しない場合は、新しいバージョンの Flash Player 用のファイルまたはアプリケーションへの変換中に、機能やファイルが壊れる可能性があります。

そのため、本マニュアルのコード例のように、一定の規則に従って大文字と小文字を使い分けるようにすることをお勧めします。このようにすることで、変数名、クラス名、および関数名が識別しやすくなります。大文字と小文字を使い分けることで、2 つの識別子を区別しないでください。大文字と小文字ではなく、インスタンス名、変数名、またはクラス名を変更します。コーディング規則の詳細については、645 ページ、第 17 章の「ActionScript 2.0 のベストプラクティスとコーディング規則」を参照してください。

Web サービスを利用する際、大文字と小文字の区別は場合によって大きな影響があります。変数の命名方法や、サーバーから SWF ファイルに返す変数での大文字と小文字の使用方法について、Web サービスが独自の規則を採用している場合があるためです。たとえば、ColdFusion の Web サービスでは、構造体またはオブジェクト内のプロパティ名が、`FIRSTNAME` のようにすべて大文字になっている可能性があります。Flash 内で大文字と小文字を一致させないと、予期しない結果となる場合があります。

メモ	大文字と小文字の区別は、`LoadVars.load()` を使用してロードする変数など、SWF ファイルにロードする外部変数にも影響を与えます。

外部スクリプト (ActionScript 2.0 のクラスファイルや `#include` コマンドを使って読み込むスクリプト、FLA ファイルのスクリプト) についても、大文字と小文字は区別されます。複数のバージョンの Flash Player に対して書き出しを行う際にランタイムエラーが発生する場合は、外部のスクリプトファイルと FLA ファイルのスクリプトを両方チェックして、大文字と小文字を一定の規則に従って使い分けていることを確認してください。

大文字と小文字の区別は、SWF ファイルに依存する形で実装されています。大文字と小文字を区別する Flash Player 8 アプリケーションが、大文字と小文字を区別しない Flash Player 6 SWF ファイルを呼び出した場合、Player 6 の SWF ファイルで実行される ActionScript では大文字と小文字は区別されません。たとえば、`loadMovie()` を使用して Flash Player 6 の SWF ファイルを Flash Player 8 の SWF ファイルに読み込んだ場合、バージョン 8 の SWF ファイルでは大文字と小文字が区別されるにもかかわらず、バージョン 6 の SWF ファイルは大文字と小文字が区別されないままになります。

シンタックスの色付けが有効になっている場合、大文字小文字が正しく入力されている言語エレメントはデフォルトでは青で表示されます。詳細については、101 ページの「予約語について」を参照してください。

ドットシンタックスとターゲットパスについて

ActionScript では、ドット (`.`) 演算子 (" ドットシンタックス ") を使用して、ステージ上のオブジェクトまたはインスタンスに属するプロパティまたはメソッドにアクセスします。ドット演算子は、ムービークリップなどのインスタンス、変数、関数、またはオブジェクトへのターゲットパスを識別するためにも使用します。

ドットシンタックス式は、オブジェクトやムービークリップの名前、ドット、エレメントの順に記述します。以下の各セクションでは、ドットシンタックス式を記述する方法を説明します。

ムービークリップ、ロードした SWF ファイル、ボタンを制御するには、" ターゲットパス " を指定する必要があります。ターゲットパスとは、SWF ファイル内のムービークリップインスタンス名、変数、およびオブジェクトの階層アドレスのことです。ムービークリップまたはボタンにターゲットパスを指定するには、あらかじめムービークリップまたはボタンにインスタンス名を割り当てておく必要があります。ムービークリップインスタンスに名前を付けるには、インスタンスを選択し、プロパティインスペクタでインスタンス名を入力します。または、ActionScript を使用してインスタンスを作成する場合は、コードを使ってインスタンス名を指定することができます。ターゲットパスを使用すると、ムービークリップへのアクションの割り当てや、変数またはプロパティの値の取得または設定が可能です。

インスタンス名の割り当てと、ドットシンタックスを使用したインスタンスのターゲット化の詳細については、次のトピックを参照してください。

- 80 ページの「ドットシンタックスを使用したインスタンスのターゲット化について」
- 85 ページの「スコープとターゲット化について」
- 86 ページの「[ターゲットパスの挿入] ボタンの使用」
- 87 ページの「スラッシュシンタックスについて」

オブジェクトとプロパティの詳細については、40 ページの「オブジェクト (Object) データ型」を参照してください。

ドットシンタックスを使用したインスタンスのターゲット化について

ムービークリップなどのインスタンスを制御するか、ロードした SWF ファイルでアセットを操作する ActionScript を記述するには、コードでその名前とアドレスを指定する必要があります。これを " ターゲットパス " と呼びます。SWF ファイルのオブジェクトをターゲットにする (アドレス指定する) には、ドットシンタックス (" ドット表記 " とも呼ばれます) を使用します。たとえば、ムービークリップやボタンインスタンスにアクションを適用するには、最初にそれをターゲットにする必要があります。ドットシンタックスを使用すると、ターゲットにするインスタンスへのパスを作成できます。ターゲットにするインスタンスへのパスは、ターゲットパスとも呼ばれます。

FLA ファイルには特定の階層があります。ステージ上でインスタンスを作成するか、ActionScript を使用してインスタンスを作成できます。他のインスタンスの内側にあるインスタンスを作成することもできます。また、他のいくつかのインスタンス内にネストされたインスタンスを作成することも可能です。いずれのインスタンスも、名前を付けることで操作できます。

インスタンスに名前を付けるには " インスタンス名 " を使用しますが、インスタンス名は 2 つの異なる方法で指定できます。その方法は次のとおりです。

- インスタンスを選択し、(インスタンスがステージ上にあるときに) インスタンス名をプロパティインスペクタに入力する手動の方法。
- ActionScript を使用する動的な方法。ActionScript を使用してインスタンスを作成し、作成時にインスタンス名を割り当てます。

プロパティインスペクタでインスタンスにインスタンス名を割り当て、[インスタンス名] テキストボックスにインスタンス名を入力します。

ActionScript を使用して作成するオブジェクトに、インスタンス名を付けることもできます。これは、次のように簡単なコードとすることができます。

```
this.createEmptyMovieClip("pic_mc", this.getNextHighestDepth());
pic_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

このコードでは、新しいムービークリップを作成し、それにインスタンス名 pic_mc を割り当てます。これで、前のコードで示したようにイメージをロードするなど、コードを使用して pic_mc インスタンスを操作できるようになります。

スコープの操作の詳細については、85 ページの「スコープとターゲット化について」および 57 ページの「変数とスコープについて」を参照してください。

インスタンスのターゲット化

SWF ファイルで何かを行う場合は、目的のインスタンスをターゲットにして、アクションを割り当てるかプロパティを変更するなどの指示をします。通常は、ターゲットパスを作成してそのインスタンスの SWF ファイルでの場所 (たとえば、タイムライン上の場所や、ネストされているインスタンス) を定義する必要があります。これまでに、FLA ファイルの多くのインスタンスにインスタンス名を付け、それらのインスタンス名を使用するコードを FLA ファイルに追加しました。この操作を行うときに、そのインスタンスをターゲットにし、再生ヘッドを移動する、Web ページを開くなどの指示をします。オブジェクトとプロパティの詳細については、40 ページの「オブジェクト (Object) データ型」を参照してください。

インスタンスをターゲットにするには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択します。
2. [ファイル]-[名前を付けて保存] を選択し、target.fla という名前を付けてファイルを保存します。
3. 楕円ツールを使用して、ステージにシェイプを描画します。任意のサイズとカラーの楕円を描画します。
4. 選択ツールを使用して、ステージ上の楕円を選択します。

ヒント	必要に応じて線と塗りを選択します。

5. [修正]-[シンボルに変換] を選択し、[ムービークリップ] オプションを選択し、[OK] をクリックしてシンボルを作成します。
6. ステージ上のムービークリップを選択し、プロパティインスペクタで myClip というインスタンス名を付けます。
7. 新しいレイヤーを挿入し、レイヤーの名前を actions に変更します。
8. actions レイヤーのフレーム 1 に次の ActionScript を追加します。

```
myClip._xscale = 50;
```

このコード行は、ステージ上の `myClip` インスタンスをターゲットにしています。ActionScript により、インスタンスが元の幅の半分に縮小されます。ActionScript はムービークリップシンボルと同じタイムライン上にあるので、インスタンス名を使用するだけで、インスタンスをターゲットにすることができます。ただし、インスタンスが別のタイムライン上にあるか、他のインスタンス内にネストされている場合は、ターゲットパスを適切に修正する必要があります。

ネストされたインスタンスのターゲット化

他のインスタンスの内側でネストされたインスタンスをターゲットにすることもできます。81 ページの「インスタンスのターゲット化」の演習を基にして、myClip インスタンスの内側に 2 番目のムービークリップインスタンスを配置できます。ActionScript を使用して、ネストされたインスタンスをターゲットにすることもできます。以下の演習に進む前に、81 ページの「インスタンスのターゲット化」の演習を完了してください。次に、以下の手順に従ってネストされたインスタンスをターゲットにします。

ネストされたインスタンスをターゲットにするには :

1. インスタンスのターゲット化に関する手順に従って "target.fla" を開き、target2.fla という名前に変更します。
2. ステージ上の myClip インスタンスをダブルクリックします。
3. 楕円ツールを選択し、myClip インスタンスの内側に別の楕円を描画します。
4. 新しいシェイプを選択し、[修正]-[シンボルに変換] を選択します。
5. [ムービークリップ] オプションを選択し、[OK] をクリックします。
6. 新しいインスタンスを選択し、プロパティインスペクタの [インスタンス名] テキストボックスに「myOtherClip」と入力します。
7. 編集バーの [シーン 1] をクリックしてメインタイムラインに戻ります。

8. actions レイヤーのフレーム 1 に次の ActionScript を追加します。

```
myClip.myOtherClip._xscale = 50;
```

この ActionScript は、myOtherClip インスタンスのサイズを現在の幅の 50% に変更します。"target.fla" ファイルによって myClip インスタンスの _xscale プロパティが修正されたこと、および myOtherClip はネストされたシンボルであることにより、myOtherClip は元の幅の 25 パーセントになります。

固有のタイムラインを持っている、ネストされたムービークリップを操作する場合は、次のようなコードを使って、ネストされたインスタンスのタイムラインで再生ヘッドを操作できます。

```
myClip.nestedClip.gotoAndPlay(15);
myClip.someOtherClip.gotoAndStop("tweenIn");
```

操作するクリップ (nestedClip など) は、アクションの前に表示されます。この詳細については、後述のセクションで説明します。

前述の例で示したように、アクセスできるのは、ステージ上のインスタンスの定義済みのメソッドとプロパティに限られません。starClip ムービークリップに変数を設定する次のコードのように、ムービークリップ内に変数を設定することもできます。

```
starClip.speed = 1.1;
starClip.gravity = 0.8;
```

starClip ムービークリップインスタンスに以前に速度変数または重力変数のいずれかが存在した場合、以前の値は新しい値が設定されるとすぐに上書きされます。MovieClip クラスは dynamic キーワードを使用して定義されているので、新しいプロパティを starClip ムービー クリップに追加することができます。dynamic キーワードは、指定したクラス (この場合は MovieClip) に基づくオブジェクトが実行時にダイナミックプロパティを追加したり、ダイナミックプロパティにアクセスできるようにします。ダイナミックステートメントの詳細については、『ActionScript 2.0 リファレンスガイド』の「dynamic statement」を参照してください。

ダイナミックインスタンスとロードされたコンテンツのターゲット化

ActionScript を使用してオブジェクトを作成し、後でターゲットパスを使用してオブジェクトをターゲットにすることができます。たとえば、次の ActionScript を使用してムービークリップを作成できます。続いて、次の例に示すように、ActionScript を使用してムービークリップの回転を変更できます。

動的に作成したムービークリップインスタンスをターゲットにするには :

1. 新しい Flash ドキュメントを作成し、targetClip.fla という名前を付けて保存します。
2. 新しいレイヤーを挿入し、レイヤーの名前を actions に変更します。
3. actions レイヤーのフレーム 1 に次の ActionScript を追加します。

```
this.createEmptyMovieClip("rotateClip", this.getNextHighestDepth());
trace(rotateClip);
rotateClip._rotation = 50;
```

4. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   trace ステートメントによってムービークリップを作成したことがわかりますが、ステージ上には何も表示されません。ムービークリップインスタンスを作成するコードを追加しましたが、ムービークリップに何かを追加しないと、ステージ上には何も表示されません。たとえば、イメージをムービークリップにロードします。

5. オーサリング環境に戻り、[アクション] パネルを開きます。

6. 手順 3 で追加したコードの後に、次の ActionScript を入力します。

   ```
   rotateClip.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
   ```

   このコードは、コードで作成した rotateClip ムービークリップにイメージをロードします。ActionScript を使用して rotateClip インスタンスをターゲットにします。

7. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   これで、時計回りに 50 度回転するイメージがステージ上に表示されます。

また、ベース SWF ファイルにロードする SWF ファイルの一部をターゲットにするか識別できるようになります。

ロードされた SWF ファイルを識別するには :

- _level*X* を使用します。*X* は、SWF ファイルをロードした loadMovie() 関数で指定されたレベル番号です。

  たとえば、レベル 99 にロードされた SWF ファイルのターゲットパスは _level99 になります。次の例では、SWF ファイルをレベル 99 にロードし、可視性を false に設定します。

  ```
  //SWF をレベル 99 にロード
  loadMovieNum("contents.swf", 99);
  // レベル 99 の可視性を false に設定
  loaderClip.onEnterFrame = function(){
    _level99._visible = false;
  };
  ```

ヒント	通常、さまざまな深度のムービー クリップにコンテンツをロードできる場合は、レベルを使用しないことが賢明です。MovieClip.getNextHighestDepth() メソッドを使用すると、特定の深度に既にインスタンスがあるかどうかをチェックしなくても、新しいムービー クリップ インスタンスをステージ上で動的に作成できるようになります。

パスを使用した変数の設定

他のインスタンスの内側にネストするインスタンスに、変数を設定することができます。たとえば、他のフォームの内側にあるフォームに変数を設定する場合は、次のコードを使用できます。インスタンス submitBtn は、メインタイムラインの formClip の内側にあります。

```
this.formClip.submitBtn.mouseOver = true;
```

このパターンを使用して、特定のオブジェクト (ムービークリップまたはテキストフィールド) のメソッドまたはプロパティを表すことができます。たとえば、オブジェクトのプロパティは次のようになります。

```
myClip._alpha = 50;
```

スコープとターゲット化について

インスタンスをネストする場合に、2 番目のムービークリップをネストするムービークリップは、ネストしたインスタンスの " 親 " と呼ばれます。ネストしたインスタンスは、子インスタンスと呼ばれます。メインステージおよびメインタイムラインは、基本的にはムービークリップそのものであり、ムービークリップとしてターゲットにすることができます。スコープの詳細については、57 ページの「変数とスコープについて」を参照してください。

ActionScript を使用すると、親インスタンスと親タイムラインをターゲットにすることができます。現在のタイムラインをターゲットにする場合は、this キーワードを使用します。たとえば、現在のメインタイムラインにある、myClip というムービークリップをターゲットにする場合は、次のコードを使用します。

```
this.myClip.
```

オプションで、this キーワードをドロップし、次のコードを使用できます。

```
myClip
```

読みやすさと一貫性のために、this キーワードを追加する選択も可能です。推奨されるコーディング方法の詳細については、645 ページ、第 17 章の「ActionScript 2.0 のベストプラクティスとコーディング規則」を参照してください。

前述のいずれかのコードに関してムービークリップをトレースすると、[出力] パネルに "_level0.myClip" と表示されます。ただし、myClip ムービークリップの内側に ActionScript があるが、メインタイムラインをターゲットにしたい場合は、ムービークリップの親 (メインステージ) をターゲットにする必要があります。ムービークリップをダブルクリックし、次の ActionScript をムービークリップのタイムラインに配置します。

```
trace("me: " + this);
trace("my parent: " + this._parent);
```

SWF ファイルをテストすると、[出力] パネルに次のメッセージが表示されます。

```
me: _level0.myClip
my parent: _level0
```

これは、メインタイムラインをターゲットにしたことを示します。parent を使って、オブジェクトへの相対パスを作成できます。たとえば、ムービークリップ dogClip がアニメーションムービークリップ animalClip の内側にネストされている場合は、インスタンス dogClip の次のステートメントにより、animalClip にアニメーションの停止が指示されます。

```
this._parent.stop();
```

Flash および ActionScript に関する詳しい知識がある場合は、_root スコープを使用するユーザーがいることにお気づきでしょう。通常、_root スコープは現在の Flash ドキュメントのメインタイムラインを参照します。_root スコープの使用は、どうしても必要でない限り避けてください。_root の代わりに相対ターゲットパスを使用することができます。

コードで _root を使用すると、SWF ファイルを別の Flash ドキュメントにロードするときにエラーが発生する場合があります。SWF ファイルを別の SWF ファイルにロードする場合、ロードされたファイルの _root は、目的とする独自のルートを参照する代わりに、ロードされる SWF ファイルのルートスコープを示すことがあります。この場合は予期しない結果が生じるか、機能全体が損なわれる可能性があります。

[ターゲットパスの挿入] ボタンの使用

特定のターゲットパスや、コードに必要なターゲットパスを調べるのに時間がかかることがあります。ステージ上のインスタンスをターゲットにする場合は、[ターゲットパスの挿入] ボタンを使用すると、そのインスタンスへのパスを判断することができます。

[ターゲットパスの挿入] ボタンを使用するには :

1. [アクション] パネル ([ウィンドウ]-[アクション]) を開き、[ターゲットパスを挿入] ボタンをクリックします。現在のドキュメントのムービークリップがダイアログボックスに表示されます。
2. ダイアログボックスのリストからインスタンスを 1 つ選択します。
3. [OK] をクリックします。
4. 選択したインスタンスのターゲットパスがスクリプトペインに表示されます。

スラッシュシンタックスについて

Flash 3 と 4 では、ムービークリップや変数のターゲットパスを表すためにスラッシュシンタックスが使用されていました。このシンタックスは Flash Player 7 以前の ActionScript 1.0 でサポートされますが、Flash Player 7 または 8、および ActionScript 2.0 ではサポートされません。

スラッシュシンタックスの使用は、他のオプションがない限りお勧めしません。たとえば、Flash Player 4 または Flash Lite 1.1 以前を対象にしたコンテンツを作成し、スラッシュシンタックスを使用しなければならない場合などです。Flash Lite の詳細については、「Flash Lite 製品ページ」を参照してください。

言語区切子について

Flash には、いくつかの言語 " 区切子 " があります。最も一般的なタイプの区切子は、セミコロン (;)、コロン (:)、カッコ [( )]、および中カッコ ({ }) です。これらの各区切子には、Flash 言語における特別な意味があり、データ型の定義、ステートメントの終了、および ActionScript の構築に役立ちます。以下のセクションでは、コードで区切子を使用する方法について説明します。

言語区切子の詳細については、次のトピックを参照してください。

- 87 ページの「セミコロンおよびコロン」
- 89 ページの「中括弧」
- 92 ページの「括弧」
- 93 ページの「リテラルについて」
- 94 ページの「コメントについて」

ドット (.) 演算子および配列アクセス ([ ]) 演算子の詳細については、142 ページの「ドット演算子と配列アクセス演算子の使用」を参照してください。空白とコードのフォーマットについては、675 ページの「ActionScript シンタックスのフォーマット」を参照してください。

セミコロンおよびコロン

次の 2 行のコードに示すように、ActionScript ステートメントはセミコロン (;) で終わります。

```
var myNum:Number = 50;
myClip._alpha = myNum;
```

セミコロンを省略することは可能ですが、その場合、ActionScript コンパイラは各行が 1 つのステートメントであると見なします。ただし、コードを読みやすくするためにもセミコロンを付けることをお勧めします。[アクション] パネルまたはスクリプトウィンドウで [自動フォーマット] ボタンをクリックすると、デフォルトで末尾のセミコロンがステートメントの最後に追加されます。

メモ

セミコロンを使用してステートメントを終了すると、1 行に複数のステートメントを配置することができますが、このようにすると、一般的にコードが読みにくくなります。

セミコロンを使用する別の場所として、for ループがあります。次の例のように、セミコロンを使用すると、パラメータを区切ることができます。例では、0 から 9 までループし、[出力] パネルに各番号を表示します。

```
var i:Number;
for (i = 0; i < 10; i++) {
  trace(i); // 0,1,...,9
}
```

変数にデータ型を割り当てるには、コードでコロン (:) を使用します。アイテムに特定のデータ型を割り当てるには、次の例のように、var キーワードと名前の後にコロン (:) を付けるシンタックスを使って型を指定します。

```
// 変数またはオブジェクトの厳密な型指定
var myNum:Number = 7;
var myDate:Date = new Date();
// パラメータの厳密な型指定
function welcome(firstName:String, myAge:Number) {
}
// パラメータと戻り値の厳密な型指定
function square(num:Number):Number {
  var squared:Number = num * num;
  return squared;
}
```

オブジェクトのデータ型は、ビルトインクラス (Button、Date、MovieClip など) と、自分で作成したクラスおよびインターフェイスのどちらを基にしても宣言できます。次のコードでは、カスタムデータ型 Student の新しいオブジェクトを作成します。

```
var firstStudent:Student = new Student();
```

また、オブジェクトのデータ型には Function または Void と指定することもできます。データ型の割り当ての詳細については、33 ページ、第 3 章の「データおよびデータ型」を参照してください。

中括弧

ActionScript イベント、クラス定義、および関数は、中カッコ ({ }) を使ってブロックにまとめます。同じ行に、宣言として左カッコを配置します。

メモ	宣言の次の行にも左カッコを配置することができます。コーディング規則では、一貫性のために同じ行に左カッコを配置することが推奨されています。カッコとコーディング規則の詳細については、645 ページ、第 17 章の「ActionScript 2.0 のベストプラクティスとコーディング規則」を参照してください。

制御構造 (`if-else` または `for` など) の要素となるステートメントは、たとえ 1 つだけでも中カッコで囲むようにしてください。このような習慣を付けておくと、中カッコの付け忘れによる ActionScript のエラーを回避しやすくなります。次の例は、良くない書き方です。

```
var numUsers:Number;
if (numUsers == 0)
  trace("no users found.");
```

このコードは間違いではありませんが、ステートメントを中括弧で囲んでいないので、推奨される書き方とは言えません。

ヒント	[自動フォーマット] ボタンをクリックすると、このステートメントに中括弧が追加されます。

この trace ステートメントの後に別のステートメントを追加すると、2 番目のステートメントは、`numUsers` 変数が 0 かどうかに関係なく実行されますが、これによって予期しない結果となる可能性があります。そこで、コードに次のような中カッコを追加すれば混乱が避けられます。

```
var numUsers:Number;
if (numUsers == 0) {
  trace("no users found");
}
```

次の例では、イベントリスナーオブジェクトと MovieClipLoader インスタンスの両方を作成します。

```
var imgUrl:String = "http://www.helpexamples.com/flash/images/image1.jpg";
this.createEmptyMovieClip("img_mc", 100);
var mclListener:Object = new Object();
mclListener.onLoadStart = function() {
  trace("starting");
};
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  trace("success");
};
mclListener.onLoadError = function(target_mc:MovieClip):Void {
  trace("failure");
};
var myClipl:MovieClipLoader = new MovieClipLoader();
myClipl.addListener(mclListener);
myClipl.loadClip(imgUrl, img_mc);
```

次の例では、Student オブジェクトを作成するために使用できる、簡単なクラスファイルを表示します。クラスファイルの詳細については、179 ページ、第 6 章の「クラス」を参照してください。

ActionScript ファイルで中カッコを使用するには :

1. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択します。
2. [ファイル]-[名前を付けて保存] を選択し、新しいドキュメントに Student.as という名前を付けて保存します。
3. AS ファイルに次の ActionScript を追加します。

```
// Student.as
class Student {
  private var _id:String;
  private var _firstName:String;
  private var _middleName:String;
  private var _lastName:String;

  public function Student(id:String, firstName:String, middleName:String,
  lastName:String) {
    this._id = id;
    this._firstName = firstName;
    this._middleName = middleName;
    this._lastName = lastName;
  }
  public function get firstName():String {
    return this._firstName;
  }
  public function set firstName(value:String):Void {
    this._firstName = value;
  }
  // ...
}
```

4. クラスファイルを保存します。
5. [ファイル]-[新規] を選択し、[Flash ドキュメント] をクリックして新しい FLA ファイルを作成します。
6. 新しい FLA ファイルに student_test.fla という名前を付けて保存します。
7. メインタイムラインのフレーム 1 に次の ActionScript を入力します。

```
// student_test.fla
import Student;
var firstStudent:Student = new Student("cst94121", "John", "H.", "Doe");
trace(firstStudent.firstName); // John
firstStudent.firstName = "Craig";
trace(firstStudent.firstName); // Craig
```

8. [ファイル]-[保存] を選択して "student_test.fla" への変更内容を保存します。

9. [制御]-[ムービープレビュー] を選択して FLA ファイルおよび AS ファイルをテストします。

次の例では、関数を操作するときの中カッコの使用方法を示します。

関数と共に中カッコを使用するには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して新しい FLA ファイルを作成します。
2. [ファイル]-[名前を付けて保存] を選択し、新しいドキュメントに checkform.fla という名前を付けます。
3. Label コンポーネントのインスタンスを、[コンポーネント] パネル内からステージにドラッグします。
4. プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) を開き、Label コンポーネントインスタンスを選択した状態で、インスタンス名 status_lbl を [インスタンス名] テキストボックスに入力します。
5. [W (幅)] テキストボックスに「200」と入力して、コンポーネントの幅を 200 ピクセルに変更します。
6. TextInput コンポーネントのインスタンスをステージにドラッグし、firstName_ti というインスタンス名を付けます。
7. Button コンポーネントのインスタンスをステージにドラッグし、submit_button というインスタンス名を付けます。
8. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を追加します。

```
function checkForm():Boolean {
  status_lbl.text = "";
  if (firstName_ti.text.length == 0) {
    status_lbl.text = "Please enter a first name.";
    return false;
  }
  return true;
}
function clickListener(evt_obj:Object):Void {
  var success:Boolean = checkForm();
};
submit_button.addEventListener("click", clickListener);
```

9. [ファイル]-[保存] を選択して Flash ドキュメントを保存します。
10. [制御]-[ムービープレビュー] を選択して、オーサリング環境でコードをテストします。

    SWF ファイルで、`firstName_ti` TextInput コンポーネントにテキストがない場合に Button インスタンスをクリックすると、エラーメッセージが表示されます。このエラーは Label コンポーネントで表示され、名を入力する必要があることをユーザーに伝えます。

中カッコを使用した次の例では、オブジェクト内にプロパティを作成および定義する方法を示します。この例では、中カッコ ({}) 内に変数名を指定することで、オブジェクトにプロパティが定義されます。

```
var myObject:Object = {id:"cst94121", firstName:"John", middleName:"H.",
  lastName:"Doe"};
var i:String;
for (i in myObject) {
  trace(i + ": " + myObject[i]);
}
/*
  id: cst94121
  firstName: John
  middleName: H.
  lastName: Doe
*/
```

new Object() 関数では、シンタックスのショートカットとして空白の中カッコを使用することもできます。たとえば、次のコードでは、空白の Object インスタンスを作成しています。

```
var myObject:Object = {};
```

ヒント	左中カッコに対しては、対応する右カッコが必要です。

括弧

ActionScript で関数を定義する場合は、次のコードに示すように、カッコ [()] 内にパラメータを配置します。

```
function myFunction(myName:String, myAge:Number, happy:Boolean):Void {
  // ここにコードを記述
}
```

関数を呼び出すときに、パラメータを次のようにカッコで囲んで関数に渡します。

```
myFunction("Carl", 78, true);
```

また、ActionScript での優先順位とは異なる順序で演算が実行されるようにしたり、ステートメントを読みやすくするためにもカッコを使用できます。つまり、次のように特定の値を角カッコで囲むことで、値を計算する優先順位を変更することができます。

```
var computedValue:Number = (circleClip._x + 20) * 0.8;
```

括弧を使用しないか、2 つの異なるステートメントを使用しない場合は、優先順位により、乗算が最初に計算されます。つまり、最初の演算は 20 * 0.8 となります。次に、その結果である 16 が circleClip._x の現在の値に加算され、最終的に computedValue 変数に代入されます。

カッコを使用しないで式が評価されるようにするには、次のようにステートメントを追加する必要があります。

```
var tempValue:Number = circleClip._x + 20;
var computedValue:Number = tempValue * 0.8;
```

角カッコや中カッコと同じように、左カッコに対しては右カッコが必要です。

リテラルについて

"リテラル" は、コードに直接表示される値です。Flash ドキュメント内では、リテラルは定数 (変化しない) 値となります。リテラルの例には、true、false、0、1、52、ストリング “foo” などがあります。

次の例はすべてリテラルです。

```
17
"hello"
-3
9.4
null
undefined
true
false
```

リテラルはグループ化して複合リテラルとすることもできます。配列リテラルは、角カッコ ([]) で囲まれ、カンマ (,) を使用して配列エレメントが区切られます。配列リテラルを使用して配列を初期化することもできます。次の例では、配列リテラルを使用して初期化される 2 つの配列を示します。new ステートメントを使用し、複合リテラルをパラメータとして Array クラスコンストラクタに渡すことができますが、任意の ActionScript ビルトインクラスのインスタンスを初期化すると、リテラル値を直接代入することができます。

```
// 新しいステートメントを使用
var myStrings:Array = new Array("alpha", "beta", "gamma");
var myNums:Array = new Array(1, 2, 3, 5, 8);

// 直接リテラルを代入
var myStrings:Array = ["alpha", "beta", "gamma"];
var myNums:Array = [1, 2, 3, 5, 8];
```

汎用オブジェクトを初期化するためにリテラルを使用することもできます。汎用オブジェクトは、Object クラスのインスタンスです。オブジェクトリテラルは中カッコ ({}) で囲まれ、カンマ (,) を使ってオブジェクトプロパティが区切られます。各プロパティはコロン (:) で宣言されます。コロンにより、プロパティ名とプロパティ値が区切られます。

new ステートメントを使用して汎用オブジェクトを作成し、オブジェクトリテラルをパラメータとして Object クラスコンストラクタに渡すか、宣言するインスタンスにオブジェクトリテラルを直接渡すことができます。次の例では、新しい汎用オブジェクトを作成し、propA、propB、および propC の 3 つのプロパティを使ってオブジェクトを初期化します。それぞれの値は、1、2、および 3 に設定されます。

```
// 新しいステートメントを使用
var myObject:Object = new Object({propA:1, propB:2, propC:3});

// 直接リテラルを代入
var myObject:Object = {propA:1, propB:2, propC:3};
```

ストリングリテラルと String オブジェクトを混同しないように注意してください。次の例では、コードの 1 行目でストリングリテラル firstStr を作成し、2 行目で String オブジェクト secondStr を作成します。

```
var firstStr:String = "foo"
var secondStr:String = new String("foo")
```

String オブジェクトを特に使用する必要がない限り、ストリングリテラルを使用してください。ストリングの詳細については、397 ページの「ストリングおよび String クラスについて」を参照してください。

コメントについて

コメントは、コンパイラによって評価されない、わかりやすい説明を使用してコードに注釈を付ける方法です。コメントはコード内で使用して、コードの役割や、ドキュメントに返されるデータなどについて説明することができます。コメントを使用すると、コーディングでの重要な判断を覚えておくことができます。また、コードを読むユーザーにとっても役立ちます。コメントは、コードの目的を明確に説明する必要があり、コードを単に変換したようなものであってはなりません。コードを一見しただけでわからない内容があれば、そこにコメントを追加してください。

スクリプトにはコメントを付けておくことを強くお勧めします。コメントはコード内で行う判断を記録するものであり、方法と理由の両方を示します。また、ActionScript が理解しやすくなります。たとえば、回避策などをコメントで説明しておくことが考えられます。そのため、自分や他の開発者が、コードで更新または修正する部分を簡単に見つけることができます。さらに、将来のバージョンの Flash または Flash Player で問題が修正または改善された場合は、回避策を削除することで、ActionScript の機能を向上させることができます。

雑然としたコメントの使用は避けてください。雑然としたコメントの例として、ブロックの作成またはコメントとの区切りに等号 (=) やアスタリスク (*) を使用する場合があります。代わりに、空白を使ってコメントと ActionScript を区切ります。[アクション] パネルまたはスクリプトウィンドウの [自動フォーマット] ボタンを使って ActionScript をフォーマットすると、空白が削除されます。その場合は空白を追加し直すか、空白を維持するための単一のコメント行 (//) を使用します。このようなコメント行を使用すれば、空白がどこであったかを確認するよりも、コードをフォーマットした後で削除するのが簡単になります。

プロジェクトを展開する前に、余分なコメント (“x および y 変数を定義 ” など)、または他の開発者がすぐにわかるようなコメントはコードから削除します。ActionScript 内の余分なコメントが多くなった場合は、コードの一部を書き直す必要がないか検討してください。コードの動作について多くのコメントを含める必要がある場合、一般的には ActionScript が洗練されておらず、直感的でないことを意味します。

シンタックスの色付けが有効になっている場合、コメントはデフォルトでグレーで表示されます。コメントの長さにより、書き出される SWF ファイルのサイズが変わることはありません。また、コメントは AcrionScript のシンタックスやキーワードの規則に従って記述する必要もありません。ActionScript の実行時に、コメントの内容は無視されます。

メモ	読者に教えることが目的の ActionScript の中では、コメントの使用が最も重要です。Flash の教育目的でサンプルアプリケーションを作成する場合や、ActionScript の記事またはチュートリアルを作成する場合には、コードにコメントを追加します。

単一行コメント

単一行コメントを使用すると、コードの単一行にコメントを追加できます。単一行のコードをコメントアウトするか、コードの目的に関する短い説明を追加します。行全体または行の途中以降をコメントにするには、次の例のように、その部分の先頭にスラッシュ 2 つ (//) を入力します。

```
// 年齢のローカル変数を設定
var myAge:Number = 26;
```

単一行コメントは通常、コードの一部分を説明するために使用します。1 行に収まる短いコメントで使用してください。次の ActionScript は単一行コメントを含むコードの例です。

```
while (condition) {
   // ステートメントで条件を操作する
}
```

複数行コメント

コメントが複数行にわたる場合は、ブロックコメントと呼ばれる複数行のコメントを使用します。一般的に開発者は、ファイル、データ構造、メソッド、およびファイル記述の説明に複数行コメントを使用します。ブロックコメントは通常、ファイルの先頭やメソッドの前またはメソッドの内部に置かれます。

コメントブロックを作成するには、コメントブロックの先頭行に /* を配置し、末尾行に */ を配置します。コメントが長くなってしまう場合に、この方法を使用すると、各行の先頭に「//」を入力せずに済みます。連続する多くの行に対して // を使用すると、コメントを修正するときに問題が発生する可能性があります。

複数行コメントの形式は次のとおりです。

```
/*
  The following ActionScript initializes variables used in the main and sub-menu
  systems. Variables are used to track what options are clicked.
*/
```

ヒント	コメント文字 (/* と */) は、コメントの前後にそれぞれ独立した行として配置しておけば、行頭に 2 つのスラッシュを挿入する (///* と //*/) ことで容易にコメント化できます。こうすることで、コードをコメント化および非コメント化する作業が手早く簡単になります。

スクリプト全体の中の一部分だけをテストしたいときに、残りの部分をまとめてコメントアウトするのに便利です。たとえば、次のスクリプトを実行すると、コメントブロック内のいずれのコードも実行されません。

```
// 以下のコードは実行される
var x:Number = 15;
var y:Number = 20;

// 以下のコードはコメントアウトされ、実行されない
/*
// 新しい Date オブジェクトを作成する
var myDate:Date = new Date();
var currentMonth:Number = myDate.getMonth();
// 月の数字を月の名前に変換する
var monthName:String = calcMonth(currentMonth);
var year:Number = myDate.getFullYear();
var currentDate:Number = myDate.getDate();
*/

// 以下のコードは実行される
var namePrefix:String = "My name is";
var age:Number = 20;
```

ヒント	ブロックコメントの前には空白行を配置することをお勧めします。

後続コメント

コードと同じ行にコメントを追加するには、後続コメントを使用します。後続コメントは、ActionScript コードと同じ行に表示されます。開発者は、一般的に後続コメントを使用して、変数に含まれている内容を示したり、ActionScript の行から返される値について説明します。後続コメントは次のようにフォーマットします。

```
var myAge:Number = 26; // 自分の年齢を表す変数
trace(myAge); // 26
```

コードと区別しやすいように、コメントの左には余白を取ってください。次のコードに示すように、できればコメントの位置を揃えるようにすることをお勧めします。

```
var myAge:Number = 28;            // 年齢
var myCountry:String = "Canada";  // 国籍
var myCoffee:String = "Hortons";  // コーヒーの好み
```

自動フォーマット機能 ([アクション] パネルの [自動フォーマット] ボタンをクリック) を使用すると、後続コメントは次の行に移動します。このようなコメントはコードをフォーマットした後に追加するか、[自動フォーマット] ボタンを使用した後でコメントの配置を変更する必要があります。

クラス内部のコメント

クラスおよびインターフェイス内でコメントを使用すると、クラスの内容を開発者が理解する上で役立ちます。クラス名、バージョン番号、日付、著作権表記などのコメントをクラスファイルの先頭に記述するようにしてください。たとえば、次のようなコメントを書いておきます。

```
/**
  Pelican class
  version 1.2
  10/10/2005
  copyright Adobe Systems Incorporated
*/
```

ブロックコメントは、ファイル、データ構造、メソッド、およびファイル記述の説明に使用します。ブロックコメントは通常、ファイルの先頭やメソッドの前またはメソッドの内部に置かれます。

典型的なクラスファイルまたはインターフェイスファイルのコメントには、"ドキュメントコメント" と " 実装コメント " の 2 種類があります。ドキュメントコメントはコードの仕様を記述するために使われ、実装を記述することはありません。ドキュメントコメントを使用して、インターフェイス、クラス、メソッド、およびコンストラクタを説明します。実装コメントはコードをコメントアウトする、つまりコードの特定の部分の実装をコメント化するために使われます。

クラス、インターフェイス、またはメンバーごとに 1 つのドキュメントコメントを含めてください。ドキュメントコメントは、宣言の直前に配置します。ドキュメントコメントとしては適さない追加情報がある場合は、実装コメント (ブロックコメントまたは単一行コメント形式) を使用します。実装コメントは、宣言の直後に配置します。

この 2 種類のコメントでは、異なる区切り記号を使用しています。ドキュメントコメントは /** と */ で区切り、実装コメントは /* と */ で区切ります。

ヒント	当該クラスと直接関係のないコメントは含めないでください。たとえば、対応するパッケージを説明するようなコメントは書かないようにします。

単一行のコメント、ブロックコメント、および後続コメントをクラスファイルで使用することもできます。これらの種類のコメントの詳細については、次のセクションを参照してください。

- 95 ページの「単一行コメント」
- 96 ページの「複数行コメント」
- 97 ページの「後続コメント」

定数およびキーワードについて

定数およびキーワードは、ActionScript シンタックスの中心となります。定数は、変更できない固定値を持つプロパティであり、アプリケーションを通じて変更されない値です。

Flash には、アプリケーション開発を簡単にするために、定義済みの定数がいくつか用意されています。定数の例は、`Key.ENTER`、`Key.PGDN` などの多くのプロパティを含む、Key クラス内にあります。定数を利用すると、たとえば Enter キーおよび Page Down キーのキーコード値が 13 および 34 であることを覚える必要はありません。定数値を使用すると、開発とデバッグが簡単になるだけでなく、他の開発者にとってコードが読みやすくなります。

ActionScript のキーワードは、特定の種類のアクションを実行するために使用されます。キーワードは予約語でもあるので、キーワードを変数名、関数名、ラベル名などの識別子として使用することはできません。予約済みのキーワードには、`if`、`else`、`this`、`function`、`return` などがあります。

定数とキーワードの詳細については、次のトピックを参照してください。

- 99 ページの「定数の使用」
- 101 ページの「キーワードについて」
- 101 ページの「予約語について」

オブジェクトとプロパティの詳細については、40 ページの「オブジェクト (Object) データ型」を参照してください。言語の定数 (`false`、`NaN` など) のリストについては、『ActionScript 2.0 リファレンスガイド』で「ActionScript 言語エレメント」の「定数」のカテゴリを参照してください。

定数の使用

定数は、変更できない固定値を持つプロパティです。つまり、アプリケーションを通じて変更されない値です。ActionScript 言語には、定義済みの定数が数多く用意されています。たとえば、定数 BACKSPACE、ENTER、SPACE、および TAB は、Key クラスのプロパティであり、キーボードのキーを表します。定数 Key.TAB の意味は常に同じであり、キーボード上の Tab キーを示します。定数は、値の比較や、アプリケーションで変更しない値を使用する場合に便利です。

ユーザーが Enter キーを押しているかどうかを調べるには、次のステートメントを使用します。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.getCode() == Key.ENTER) {
    trace("Are you ready to play?");
  }
};
Key.addListener(keyListener);
```

この ActionScript が機能するためには、オーサリング環境でキーワードショートカットを無効にしなければならない場合があります。メインメニューから [制御]-[ムービープレビュー] を選択し、Player で SWF ファイルをプレビュー中に、SWF ファイルのプレビューウィンドウから [制御]-[キーボードショートカットを無効] を選択します。

Flash では、プライベートメンバー変数を使用して独自のカスタムクラスを作成する場合を除き、ユーザー独自の定数値を作成する方法はありません。また、Flash 内で "読み取り専用" の変数を作成することはできません。

変数名は小文字で指定するか、大文字と小文字を混在して指定するのに対し、定数 (変化しない変数) の名前は大文字で指定します。次の ActionScript に示すように、単語間はアンダースコアで区切ります。

```
var BASE_URL:String = "http://www.adobe.com"; // 定数
var MAX_WIDTH:Number = 10;                     // 定数
```

静的定数は大文字で記述し、単語間はアンダースコアで区切ります。for ループのカウンタ値として使用する 1、0、-1 のような場合を除き、コード内に数値定数を直接に記述しないでください。

値が決して変化しないプロパティを参照する必要がある状況では、定数を使用できます。定数を使用することで、リテラルを使用した場合には見つけられないタイプミスを見つけることができます。また、1 か所で値を変更することもできます。リテラルの詳細については、93 ページの「リテラルについて」を参照してください。

たとえば、次の例のクラス定義では、ActionScript 2.0 によって使用される命名規則に従う、3 つの定数を作成します。

アプリケーションで定数を使用するには :

1. [ファイル]-[新規]-[ActionScript ファイル] を選択し、AS ファイルを作成します。
2. 新しいファイルに ConstExample.as という名前を付けます。
3. 次のコードをスクリプトウィンドウに入力します。

```
class ConstExample {
  public static var EXAMPLE_STATIC:String = "Global access";
  public var EXAMPLE_PUBLIC:String = "Public access";
  private var EXAMPLE_PRIVATE:String = "Class access";
}
```

   EXAMPLE_STATIC プロパティは静的プロパティです。つまり、クラスの特定のインスタンスではなく、クラス全体に適用されます。クラスの静的プロパティにアクセスするには、インスタンス名ではなくクラス名を使用する必要があります。クラスインスタンスを通じて静的プロパティにアクセスすることはできません。

4. 新しい Flash ドキュメントを作成し、const.fla という名前を付けて保存します。
5. [アクション] パネルを開き、タイムラインのフレーム 1 に 次のコード入力します。

```
trace(ConstExample.EXAMPLE_STATIC); // 出力 : グローバルアクセス
```

   EXAMPLE_STATIC プロパティを静的プロパティとして宣言する場合は、次のコードを使用してプロパティの値にアクセスします。

6. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   [出力] パネルに " グローバルアクセス " と表示されます。

7. [アクション] パネルで、手順 5 で追加したコードの後に次のコードを入力します。

```
trace(ConstExample.EXAMPLE_PUBLIC); // エラー
trace(ConstExample.EXAMPLE_PRIVATE); // エラー
```

8. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   EXAMPLE_PUBLIC および EXAMPLE_PRIVATE プロパティは静的プロパティではありません。クラスを通じてそれらの値にアクセスしようとすると、次のエラーメッセージが表示されます。

```
The property being referenced does not have the static attribute.
```

   静的でないプロパティにアクセスするには、クラスのインスタンスを通じて値にアクセスする必要があります。EXAMPLE_PUBLIC プロパティはパブリックプロパティであるため、クラス定義の外側のコーディングに使用できます。

9. [アクション] パネルで、手順 5 および 7 で追加した trace ステートメントを削除します。
10. [アクション] パネルで、次のコードを入力します。

```
var myExample:ConstExample = new ConstExample();
trace(myExample.EXAMPLE_PUBLIC); // 出力 : パブリッククラス
```

   このコードは、myExample インスタンスをインスタンス化し、EXAMPLE_PUBLIC プロパティにアクセスします。

11. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

    [出力] パネルに " パブリックアクセス " と表示されます。

12. [アクション] パネルで、手順 10 で追加した trace ステートメントを削除します。

13. [アクション] パネルで、次のコードを入力します。

    ```
    trace(myExample.EXAMPLE_PRIVATE); // エラー
    ```

    EXAMPLE_PRIVATE プロパティはプライベートプロパティであり、クラス定義内のみで使用できます。

14. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

    [出力] パネルに " メンバーがプライベートで、アクセスできません。" と表示されます。

    ビルトインクラスの詳細と、カスタムクラスの作成の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

キーワードについて

キーワードは、1 つの特定の処理を実行する ActionScript の単語です。たとえば、var キーワードは変数を宣言するために使用されます。次のコード行に、var キーワードを示します。

```
var myAge:Number = 26;
```

キーワードは、特定の意味を持つ予約語です。たとえば、class キーワードを使用して新しい ActionScript クラスを定義し、var キーワードを使用してローカル変数を宣言します。予約済みのキーワードの他の例としては、if、else、this、function、return などがあります。

キーワードを識別子 (変数、関数、またはラベル名) として使用することはできません。また、他の目的 (たとえばインスタンス名) のために、FLA ファイルの別の部分で使用しないでください。特に33 ページ、第 3 章の「データおよびデータ型」を読んだ場合は、既に var キーワードを数多く使用しています。ActionScript では、言語における特定の使用目的のために単語が予約されています。そのため、キーワードを識別子 (変数、関数、またはラベル名) として使用することはできません。これらのキーワードの一覧は、101 ページの「予約語について」に記載されています。

予約語について

" 予約語 " は、コードで識別子として使用することができない単語です。これは、ActionScript によって予約されているためです。予約語には、" キーワード " が含まれます。これは ActionScript ステートメントであり、将来の使用のために予約されている単語です。したがって、変数、インスタンス、カスタムクラスなどに名前を付けるためにキーワードを使用することはできません。使用すると、技術的な問題が発生する可能性があります。

予約語ではないが、ActionScript コードで識別子 (変数またはインスタンス名) としては使用してはならない単語がいくつかあります。これらは言語コンストラクトと呼ばれ、ActionScript 言語を構成するビルトインクラスで使用される単語です。そのためコードでは、変数、クラス、またはインスタンスに名前を付けるときなどに、プロパティ名、メソッド名、クラス名、インターフェイス名、およびコンポーネントクラス名を識別子として使用しないでください。

ActionScript または ECMAScript (ECMA-262) Edition 4 の言語仕様草案で将来に使用するために予約されているキーワード (スクリプトのエラーとなりうる) および保護されているキーワードの詳細については、648 ページの「予約語および言語コンストラクトの回避」を参照してください。

言語コンストラクトの名前については、『ActionScript 2.0 リファレンスガイド』を参照してください。

ステートメントについて

"ステートメント" は、特定のアクションなど、何かを実行するために FLA ファイルに与える指示です。たとえば、条件ステートメントを使用すると、何かが true であるか存在しているかを判断できます。次に、条件が true であるかどうかに基づいて、関数や式などの指定するアクションを実行することができます。

たとえば、if ステートメントは条件ステートメントであり、条件を評価して、コードで発生する次のアクションを判断します。

```
//if ステートメント
if (condition) {
   // ステートメント ;
}
```

また、return ステートメントは、関数の実行結果を値として返します。

ActionScript のフォーマットまたは記述には、多くの異なる方法があります。シンタックスを作成する方法は、ActionScript を記述する開発者によって異なる可能性があります。たとえば、ステートメントに空白を挿入する方法や、コードで中カッコ ({ }) を配置する場所などです。コードを壊すことなくステートメントを作成するにはいくつか別の方法がありますが、適格な ActionScript を記述するためには、従う必要があるガイドラインがあります。

ActionScript を読みやすくするには、1 行に 1 つのステートメントのみを書くようにします。次の例では、ステートメントの推奨の使用方法と推奨されない使用方法を示します。

```
theNum++;        // 推奨されます
theOtherNum++;   // 推奨されます
aNum++; anOtherNum++;   // 推奨されません
```

変数は別個のステートメントとして代入します。次に例を示します。

```
var myNum:Number = (a = b + c) + d;
```

この ActionScript はコード内に代入ステートメントを埋め込んでいますが、読みにくくなります。これは、次のように各変数を別のステートメントで代入する方が読みやすくなります。

```
var a:Number = b + c;
var myNum:Number = a + d;
```

以下のセクションでは、ActionScript で特定のステートメントを作成する方法を示します。イベントの記述とフォーマットの詳細については、279 ページ、第 9 章の「イベントの処理」を参照してください。

各ステートメントの詳細については、次のトピックを参照してください。

- 103 ページの「複合ステートメントについて」
- 103 ページの「条件について」
- 113 ページの「ループを使用した繰り返しアクション」

複合ステートメントについて

複合ステートメントには、中カッコ ({ }) で囲まれた数多くのステートメントが含まれます。複合ステートメント内のステートメントは、任意の種類の ActionScript ステートメントとすることができます。一般的な複合ステートメントを次に示します。

中カッコ内のステートメントは、複合ステートメント自体よりもインデントを下げて記述します。次に例を示します。

```
var a:Number = 10;
var b:Number = 10;
if (a == b) {
  // このコードはインデントされる
  trace("a == b");
  trace(a);
  trace(b);
}
```

この複合ステートメントにはいくつかのステートメントが含まれますが、ActionScript コードでは 1 つのステートメントとして機能します。左中カッコを複合ステートメントの末尾に配置しています。また、右中カッコは行の先頭に、複合ステートメントの先頭と位置を揃えて配置しています。

中カッコの使用の詳細については、89 ページの「中括弧」を参照してください。

条件について

条件を使用すると、何かが true または存在するかどうかを判断し、条件が true かどうかに基づいて、オプションでループを使用してアクションを繰り返すか、指定するアクション (関数、式など) を実行することができます。たとえば、特定の変数が定義されているかどうかや、特定の値があるかどうかを判断し、その結果に基づいてコードブロックを実行できます。また、ユーザーのシステムクロックに設定されている時刻や、現在の場所の天気に基づいて、Flash ドキュメント内でグラフィックを変更することもできます。

条件が存在するかどうかに応じてアクションを実行する、またはアクションを繰り返す (ループステートメントを作成する) には、`if`、`else`、`else if`、`for`、`while`、`do while`、`for..in`、または `switch` の各ステートメントを使用できます。

使用できる条件の詳細や、条件を記述する方法については、以下のトピックを参照してください。

- 104 ページの「条件の記述について」
- 105 ページの「if ステートメントの使用」
- 106 ページの「if..else ステートメントの使用」
- 107 ページの「if..else if ステートメントの使用」
- 108 ページの「switch ステートメントの使用」
- 110 ページの「try..catch および try..catch..finally ステートメントの使用」
- 112 ページの「条件演算子と代替シンタックスについて」

条件の記述について

条件が true であるか false であるかをチェックするステートメントは、`if` で始まります。条件が `true` に評価された場合は、その次のステートメントが実行されます。条件が `false` に評価された場合は、コードブロックの外側にあるその次のステートメントまで処理がスキップされます。

ヒント	コードの処理速度を向上させるには、最も一般的な条件 (多くの場合に満たされる条件) が最初にチェックされるようにします。

次のステートメントでは 3 つの条件がテストされます。`else if` は、直前の条件が false である場合に比較される条件を表します。

```
if ((passwordTxt.text.length == 0) || (emailTxt.text.length == 0)) {
  gotoAndStop("invalidLogin");
}  else if (passwordTxt.text == userID){
  gotoAndPlay("startProgram");
}
```

このコードでは、[passwordTxt] または [emailTxt] テキストフィールドの長さが 0 である (ユーザーが値を入力していないなど) 場合に、Flash ドキュメントは `invalidLogin` フレームラベルにリダイレクトされます。[passwordTxt] および [emailTxt] テキストフィールドの両方に値が含まれていて、[passwordTxt] テキストフィールドが `userID` 変数に一致する場合、SWF ファイルは `startProgram` フレームラベルにリダイレクトされます。

いくつかの条件の 1 つをチェックする場合は、複数の `else if` ステートメントの代わりに、`switch` ステートメントを使用できます。`switch` ステートメントの詳細については、108 ページの「switch ステートメントの使用」を参照してください。

ActionScript アプリケーションで異なる種類の条件を記述する方法については、以下のセクションを参照してください。

if ステートメントの使用

特定の条件が true かどうかに基づいて一連のステートメントを実行する場合は、if ステートメントを使用します。

```
//if ステートメント
if (condition) {
  // ステートメント ;
}
```

Flash プロジェクトを操作する場合は、if ステートメントを使用することが何回かあります。たとえば、ユーザーが Web サイトの特定のセクションにアクセスできるようにする前にログインを要求する Flash サイトを作成する場合は、if ステートメントを使用して、ユーザーがユーザー名フィールドおよびパスワードフィールドに何らかのテキストを入力したことを検証できます。

外部データベースを使用してユーザー名とパスワードを検証する必要がある場合は、ユーザーが送信するユーザー名とパスワードの組み合わせが、データベースのレコードと一致するかどうかを確認する必要があります。また、サイトの指定した部分にアクセスする許可をユーザーが持っているかどうかもチェックする必要があります。

Flash でアニメーションのスクリプトを作成する場合は、if ステートメントを使用して、ステージ上のインスタンスがステージの境界内にあるかどうかをテストします。たとえば、y 軸に沿ってボールが下方向に移動する場合は、ボールがステージの下端に当たったら向きを変え、ボールが上方向に跳ねるように見せるために、ボールが下端に当たったことを検出する必要があります。

if ステートメントを使用するには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を入力します。

```
// AM と PM を格納するストリングを作成する
var amPm:String = "AM";
// Date にパラメータを渡さないので、現在の日時を返す
var current_date:Date = new Date();
// 現在の時間が 12 以上の場合は、amPm ストリングを "PM" に設定する
if (current_date.getHours() >= 12) {
   amPm = "PM";
}
trace(amPm);
```

3. [制御]-[ムービープレビュー] を選択して ActionScript をテストします。

   このコードでは、現在の時刻に基づいて、AM または PM を格納するストリングを作成します。現在の時間が 12 以上の場合、amPm ストリングは PM に設定されます。最後に、amPm ストリングをトレースすると、時間が 12 以上の場合は "PM" が表示されます。それ以外の場合は、"AM" が表示されます。

if..else ステートメントの使用

if..else 条件ステートメントを使用すると、条件をテストし、その条件が満たされている場合はコードブロックを実行し、条件が満たされていない場合は別のコードブロックを実行することができます。

たとえば、次のコードは、値 x が 20 を超えているかどうかをテストし、超えている場合は trace() ステートメントを生成し、超えていない場合は別の trace() ステートメントを生成します。

```
if (x > 20) {
  trace("x is > 20");
} else {
  trace("x is <= 20");
}
```

別のコードブロックを実行しない場合は、else ステートメントのない if ステートメントを使用できます。

Flash の if..else ステートメントは、if ステートメントに似ています。たとえば、if ステートメントを使って、ユーザーが入力したユーザー名とパスワードが、データベースに保存されている値と一致するかどうかを検証する場合は、ユーザー名とパスワードが正しいかどうかに基づいて、ユーザーをリダイレクトする必要があります。ログインが有効な場合は、if ブロックを使って、ユーザーを [Welcome] ページにリダイレクトできます。ただし、ログインが無効であった場合は、else ブロックを使ってユーザーをログインフォームにリダイレクトして、エラーメッセージを表示することができます。

ドキュメントで if..else ステートメントを使用するには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して新しい FLA ファイルを作成します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を入力します。

```
// 時刻に基づいて AM/PM を格納するストリングを作成する
var amPm:String;
// Date に渡すパラメータがないので、現在の日時を返す
var current_date:Date = new Date();
// 現在の時間が 12 以上の場合は、amPm ストリングを "PM" に設定する
if (current_date.getHours() >= 12) {
  amPm = "PM";
} else {
  amPm = "AM";
}
trace(amPm);
```

3. [制御]-[ムービープレビュー] を選択して ActionScript をテストします。

   このコードでは、現在の時刻に基づいて、AM または PM を格納するストリングを作成します。現在の時間が 12 と等しい場合、amPm ストリングは PM に設定されます。最後に、amPm ストリングをトレースすると、時間が 12 以上の場合は "PM" が表示されます。それ以外の場合、[出力] パネルに "AM" が表示されます。

if..else if ステートメントの使用

if..else if 条件ステートメントを使用すると、複数の条件をテストできます。if..else if ステートメントで次のシンタックスを使用します。

```
//else-if ステートメント
if (condition) {
  // ステートメント ;
} else if (condition) {
  // ステートメント ;
} else {
  // ステートメント ;
}
```

一連の条件をチェックする必要がある場合は、Flash プロジェクトで if..else if ブロックを使用します。たとえば、ユーザーのアクセス時刻に基づいて、画面に異なるイメージを表示する場合は、現在が早朝、午後、夕方、または夜であるかを判断する一連の if ステートメントを作成できます。次に、適切なグラフィックを表示できます。

次のコードは、x の値が 20 を超えるかどうかをテストすると共に、x が負の値かどうかをテストします。

```
if (x > 20) {
  trace("x is > 20");
} else if (x < 0) {
  trace("x is negative");
}
```

ドキュメントで if..else if ステートメントを使用するには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を入力します。

```
var now_date:Date = new Date();
var currentHour:Number = now_date.getHours();
// 現在の時間が 11AM. より早い場合は if...
if (currentHour < 11) {
  trace("Good morning");
  // 現在の時間が 3PM. より早い場合は else..if...
} else if (currentHour < 15) {
  trace("Good afternoon");
  // 現在の時間が 8PM. より早い場合は else..if...
} else if (currentHour < 20) {
  trace("Good evening");
  // 現在の時間が 8PM 〜 11:59PM の場合は else
} else {
  trace("Good night");
}
```

3. [制御]-[ムービープレビュー] を選択して ActionScript をテストします。

このコードでは、現在の時間の数値を格納する currentHour というストリングを作成します。たとえば、現在が 6:19 pm である場合、currentHour には値 18 が格納されます。現在の時間を取得するには、Data クラスの getHours() メソッドを使用します。次に、if..else if ステートメントを使用して、戻り値に基づいて情報をトレースし、[出力] パネルにその情報を表示できます。詳細については、前述のコードのコメントを参照してください。

switch ステートメントの使用

switch ステートメントは ActionScript ステートメントの分岐構造を作成します。if ステートメントと同様に、switch ステートメントは条件をテストし、条件が true を返すと、ステートメントを実行します。

switch ステートメントを使用する場合、break ステートメントは、case ブロック内の残りのステートメントをスキップし、囲んでいる switch ステートメントに続く最初のステートメントにジャンプするよう Flash に指示します。case ブロックに break ステートメントが含まれない場合は、“ フォールスルー ” と呼ばれる状態が発生します。この状況では、break ステートメントが見つかるか、switch ステートメントが終了するまで、次の case ステートメントも実行されます。次の例に、この動作を示します。最初の case ステートメントには break ステートメントは含まれていないので、最初の 2 つの case (A および B) の両方のコードブロックが実行されます。

switch ステートメントには必ずデフォルトの case を含める必要があります。デフォルトの case は、必ず switch ステートメントの最後の case となり、他で別の case が追加された場合にフォールスルーエラーが発生しないように break ステートメントを含める必要があります。たとえば、次の例で条件が A に評価された場合は、case A に break ステートメントがないため、case A と B の両方のステートメントが実行されます。フォールスルーが発生した場合は、break ステートメントがない代わりに、次の case A の後の例のように、break ステートメントの位置にコメントがあります。switch ステートメントを記述するときは、次のフォーマットを使用します。

```
switch (condition) {
case A :
  // ステートメント
  // フォールスルー
case B :
  // ステートメント
  break;
case Z :
  // ステートメント
  break;
default :
  // ステートメント
  break;
}
```

ドキュメントで switch ステートメントを使用するには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を入力します。

```
var listenerObj:Object = new Object();
listenerObj.onKeyDown = function() {
  // String.fromCharCode() メソッドを使用してストリングを返す
  switch (String.fromCharCode(Key.getAscii())) {
  case "A" :
    trace("you pressed A");
    break;
  case "a" :
    trace("you pressed a");
    break;
  case "E" :
  case "e" :
    /* E doesn't have a break statement, so this block executes if you press
  e or E. */
    trace("you pressed E or e");
    break;
  case "I" :
  case "i" :
    trace("you pressed I or i");
    break;
  default :
    /* If the key pressed isn't caught by any of the above cases, execute the
  default case here. */
    trace("you pressed some other key");
  }
};
Key.addListener(listenerObj);
```

3. [制御]-[ムービープレビュー] を選択して ActionScript をテストします。

   a、e、または i キーを含めて、キーボードを使って文字を入力します。これらの 3 つの文字を入力すると、上記の ActionScript に `trace` ステートメントが表示されます。コード行により、Key クラスのリスナーとして使用する新しいオブジェクトが作成されます。このオブジェクトを使って、ユーザーがキーを押したときに `onKeyDown()` イベントを通知します。`Key.getAscii()` メソッドは、ユーザーが最後に押すか離したキーの ASCII コードを返すので、`String.fromCharCode()` メソッドを使って、パラメータの ASCII 値で表される文字を含むストリングを返す必要があります。"E" には `break` ステートメントがないので、ユーザーが *e* または *E* キーを押したときに、ブロックが実行されます。最初の 3 つの case のいずれによってもキャッチされないキーをユーザーが押した場合は、デフォルトの case が実行されます。

try..catch および try..catch..finally ステートメントの使用

try..catch..finally ブロックを使用すると、Flash アプリケーションにエラー処理を追加できます。try..catch..finally キーワードを使用すると、エラーが発生するコードブロックを囲み、そのエラーに対応することができます。try コードブロック内のコードが throw ステートメントを使用してエラーをスローする場合、制御は catch ブロック (存在する場合) に渡されます。次に、制御は finally コードブロック (存在する場合) に渡されます。オプションの finally ブロックは、エラーがスローされたかどうかに関係なく、必ず実行されます。

try ブロック内のコードがエラーをスローしない場合 (try ブロックの実行が正常に終了した場合) でも、finally ブロックのコードは実行されます。

メモ	return ステートメントを使用して try ブロックを終了した場合でも、finally ブロックは実行されます。

try..catch および try..catch..finally ステートメントは、次のフォーマットを使用して記述します。

```
//try-catch
try {
  // ステートメント
} catch (myError) {
  // ステートメント
}

// try-catch-finally
try {
  // ステートメント
} catch (myError) {
  // ステートメント
} finally {
  // ステートメント
}
```

コードがエラーをスローするたびに、カスタムハンドラを書いて、エラーを正常に処理し、適切なアクションを実行することができます。Web サービスまたはテキストファイルから外部データをロードするか、エンドユーザーにエラーメッセージを表示する必要があります。catch ブロックを使って、特定のエラーが発生したことを管理者に通知する Web サービスに接続し、アプリケーションが正しく動作していることを管理者が確認するようにできます。

番号を分割する前に try..catch..finally ブロックをデータ検証に使用するには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を入力します。

```
var n1:Number = 7;
var n2:Number = 0;
try {
   if (n2 == 0) {
      throw new Error("Unable to divide by zero");
   }
   trace(n1/n2);
} catch (err:Error) {
   trace("ERROR! " + err.toString());
} finally {
   delete n1;
   delete n2;
}
```

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。
4. [出力] パネルには、`Unable to divide by zero` と表示されます。
5. オーサリング環境に戻り、次のコード行を変更します。

   `var n2:Number = 0;`

   このコード行を次のように変更します。

   `var n2:Number = 2;`
6. [制御]-[再生] を選択して再度ドキュメントをテストします。

   `n2` の値がゼロに等しい場合は、エラーがスローされ、`catch` ブロックによってキャッチされます。この結果、[出力] パネルにメッセージが表示されます。`y` の値がゼロに等しい場合、[出力] パネルには `n1` を `n2` で除算した結果が表示されます。`finally` ブロックは、エラーが発生するかどうかとは無関係に実行され、Flash ドキュメントから `n1` および `n2` 変数の値を削除します。

エラーが発生したときに、Error クラスの新しいインスタンスをスローできるだけではありません。次の例に示すように、Error クラスを拡張して独自のカスタムエラーを作成できます。

カスタムエラーを作成するには :

1. [ファイル]-[新規] を選択し、新規 ActionScript ファイルを作成します。
2. [ファイル]-[名前を付けて保存] を選択し、ファイルに DivideByZeroException.as という名前を付けます。
3. 次の ActionScript をスクリプトペインに入力します。

```
// DivideByZeroException.as の場合 :
class DivideByZeroException extends Error {
   var message:String = "Divide By Zero error";
}
```

4. ActionScript ファイルを保存します。
5. ActionScript ファイルと同じディレクトリに exception_test.fla という名前の新しい Flash ドキュメントを作成し、ファイルを保存します。

6. メインタイムラインのフレーム 1 で、[アクション] パネルに次の ActionScript を入力します。

```
var n1:Number = 7;
var n2:Number = 0;
try {
   if (n2 == 0) {
      throw new DivideByZeroException();
   } else if (n2 < 0) {
      throw new Error("n2 cannot be less than zero");
   } else {
      trace(n1/n2);
   }
} catch (err:DivideByZeroException) {
   trace(err.toString());
} catch (err:Error) {
   trace("An unknown error occurred; " + err.toString());
}
```

7. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択してテスト環境でファイルをテストします。

   n2 の値が 0 に等しいため、DivideByZeroException カスタムエラークラスがスローされ、[出力] パネルに `Divide By Zero error` が表示されます。2 行目の n2 の値を 0 から -1 に変更し、Flash ドキュメントを再テストした場合、[出力] パネルには "`An unknown error occurred; n2 cannot be less than zero`" と表示されます。n2 の値を 0 より大きい値に設定すると、除算の結果が [出力] パネルに表示されます。カスタムクラスの作成の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

条件演算子と代替シンタックスについて

ショートカットが必要な場合は、条件 (`?:`) 演算子を使用できます。これは、" 条件式 " とも呼ばれます。条件演算子を使用すると簡単な `if..else` ステートメントを 1 行のコードに書き換えることができます。この演算子を使用すると、同じ処理を実行するために書くコードの量を減らすことができますが、ActionScript が読みにくくなります。

次の条件は、通常の方法で記述されたものです。この条件は変数 `numTwo` がゼロより大きいかどうかをチェックし、`numOne/numTwo` の結果またはストリング `carrot` を返します。

```
var numOne:Number = 8;
var numTwo:Number = 5;
if (numTwo > 0) {
  trace(numOne / numTwo); // 1.6
} else {
  trace("carrot");
}
```

条件式を使用すると、次のフォーマットを使用して同じコードを記述することができます。

```
var numOne:Number = 8;
var numTwo:Number = 0;
trace((numTwo > 0) ? numOne/numTwo : "carrot");
```

このように、簡易シンタックスは読みにくくなるので、お勧めできません。条件演算子を使用する必要がある場合は、先頭の条件 (疑問符 [?] の前の条件) をカッコで囲むようにしてください。これで ActionScript の読みやすさが向上します。次のコードは、読みやすさを向上した ActionScript の例です。

```
var numOne:Number;
(numOne >= 5) ? numOne : -numOne;
```

ブール値を返す条件ステートメントは、次のような形式で書くことができます。

```
if (cartArr.length > 0) {
  return true;
} else {
  return false;
}
```

しかし、この例のような書き方よりも、次の形式が推奨されています。

```
return (cartArr.length > 0);
```

2 番目の例の方が簡潔で、評価する式も少なくて済みます。また、読みやすさが向上し、理解しやすくなります。

複雑な条件を書くときは、カッコ [( )] を使用して条件をグループ化することをお勧めします。カッコを使用しないと、自分やこの ActionScript を使用する他の人が演算子の優先順位を間違えることも考えられます。演算子の優先順位の詳細については、138 ページの「演算子の優先順位と結合性について」を参照してください。

たとえば、次のコードは条件を囲むカッコを使用していません。

```
if (fruit == "apple" && veggie == "leek") {}
```

次のように条件をカッコで囲む方が適切です。

```
if ((fruit == "apple") && (veggie == "leek")) {}
```

ループを使用した繰り返しアクション

ActionScript では、指定された回数、またはある条件が満たされている間、1 つのアクションを繰り返すことができます。ループを使用すると、特定の条件が true の場合に、一連のステートメントを繰り返すことができます。ActionScript には、for、for..in、while、および do..while の 4 種類のループがあります。それぞれの種類のループの動作はいくらか異なり、さまざまな目的に役立ちます。

ループの多くは、何らかのカウンタを使用してループの実行回数を制御しています。毎回のループの実行は " 反復 " と呼ばれます。変数を宣言し、ループが実行されるたびにその変数の値を増加、または減少させるステートメントを記述できます。for アクションでは、カウンタと、カウンタを増加させるステートメントを記述します。

ループ	説明
for ループ	ビルトインカウンタを使用してアクションを繰り返します。
for..in ループ	ムービークリップまたはオブジェクトの子の繰り返し処理を実行します。
while ループ	条件が満たされている間アクションを繰り返します。
do..while ループ	コードブロックの最後で式が評価されることを除き、while ループと同様であるので、ループは必ず最低 1 回は実行されます。

最も一般的な種類のループは for ループであり、あらかじめ定義された回数だけコードブロックがループします。たとえば、アイテムの配列があり、配列の各アイテムで一連のステートメントを実行する場合は、for ループを使用し、0 から配列のアイテム数までループします。別の種類のループには、for..in ループがあります。これは、オブジェクト内の名前と値のペアをループし、何らかの処理を実行する場合に非常に役立ちます。これは、Flash プロジェクトをデバッグしていて、Web サービスや外部テキスト/XML ファイルなどの外部ソースからロードする値を表示するときに非常に便利です。最後の 2 種類のループ (while および do..while) は、一連のステートメントをループしたいが、ループする必要がある回数がわからない場合に役立ちます。この場合は、特定の条件が true である限りループする、while ループを使用できます。

ActionScript では、指定された回数、またはある条件が満たされている間、1 つのアクションを繰り返すことができます。ループを定義するには、while、do..while、for、および for..in アクションを使用します。このセクションでは、これらのループに関する一般的な情報を記載します。各ループの詳細については、次の手順を参照してください。

条件が満たされている間アクションを繰り返すには :

- while ステートメントを使用します。

  while ループは式を評価し、式が true である場合に、ループの本体で定義されているアクションを実行します。ループ本体のステートメントの実行が完了すると、式がもう一度評価されます。次の例では、ループは 4 回実行されます。

```
var i:Number = 4;
while (i > 0) {
   myClip.duplicateMovieClip("newMC" + i, i, {_x:i*20, _y:i*20});
   i --;
}
```

  do..while ステートメントを使用して、while ループと同様のループを定義することができます。do..while ループでは、コードブロックの最後で式が評価されるので、ループは必ず最低 1 回は評価されます。

  このことは、次のようにして確認できます。

```
var i:Number = 4;
do {
   myClip.duplicateMovieClip("newMC" + i, i, {_x:i*20, _y:i*20});
   i --;
} while (i > 0);
```

  while ステートメントの詳細については、120 ページの「while ループの使用」を参照してください。

組み込みカウンタを使用してアクションを繰り返すには :

- for ステートメントを使用します。

  ループの多くは、何らかのカウンタを使用してループの実行回数を制御しています。毎回のループの実行は " 反復 " と呼ばれます。変数を宣言し、ループが実行されるたびにその変数の値を増加、または減少させるステートメントを記述できます。for アクションでは、カウンタと、カウンタを増加させるステートメントを記述します。

  次の例では、最初の式 (var i:Number = 4) が、反復の開始前に評価される初期化式になります。2 番目の式 (i > 0) は、ループの実行前に毎回チェックされる条件です。3 番目の式 (i--) は、ループ実行後に毎回評価される " 再初期化式 " と呼ばれるものです。

```
for (var i:Number = 4; i > 0; i--) {
  myClip.duplicateMovieClip("newMC" + i, i, {_x:i*20, _y:i*20});
}
```

  for ステートメントの詳細については、117 ページの「for ループの使用」を参照してください。

ムービークリップやオブジェクトの子に対してループするには :

- for..in ステートメントを使用します。

  子とは、他のムービークリップ、関数、オブジェクト、変数などのことです。次の例では、trace ステートメントを使用して、[出力] パネルに結果を出力しています。

```
var myObject:Object = {name:'Joe', age:25, city:'San Francisco'};
var propertyName:String;
for (propertyName in myObject) {
  trace("myObject has the property: " + propertyName + ", with the value: " +
  myObject[propertyName]);
}
```

  この例では、[出力] パネルに結果が次のように表示されます。

```
myObject has the property: name, with the value: Joe
myObject has the property: age, with the value: 25
myObject has the property: city, with the value: San Francisco
```

  このスクリプトを、子ムービークリップなど、特定のタイプの子に対して繰り返すことができます。そのためには、for..in を typeof 演算子と組み合わせて使います。次の例では、instance2 という名前の子ムービークリップインスタンスは、ステージ上のムービークリップインスタンス内にあります。タイムラインのフレーム 1 に次の ActionScript を追加します。

```
for (var myName in this) {
  if (typeof (this[myName]) == "movieclip") {
    trace("I have a movie clip child named " + myName);
  }
}
```

for..in ステートメントの詳細については、118 ページの「for..in ループの使用」を参照してください。

警告

Flash での反復は Flash Player で非常にすばやく実行されますが、ループはプロセッサによって大きく左右されます。ループ内の反復が多く、各ブロック内で実行されるステートメントが多いほど、消費されるプロセッサリソースも多くなります。また、ループが正しく記述されていないと、パフォーマンスや安定性に関する問題が発生しかねません。

各ステートメントの詳細については、120 ページの「while ループの使用」など、この章の後の個別のセクション、および『ActionScript 2.0 リファレンスガイド』の各項目を参照してください。

ループの作成と終了について

次の例では、月名の簡単な配列を示します。for ループは 0 から配列のアイテム数まで繰り返し、[出力] パネルに各アイテムを表示します。

```
var monthArr:Array = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
  "Aug", "Sep", "Oct", "Nov", "Dec");
var i:Number;
for (i = 0; i < monthArr.length; i++) {
  trace(monthArr[i]);
}
```

配列を操作する場合は、配列が単純であるか複雑であるかとは無関係に、" 無限ループ " と呼ばれる状態に注意する必要があります。無限ループは、名前が示すように、際限なく続くループ状態です。無限ループにより、Flash アプリケーションのクラッシュ、Web サーバーに対する Flash ドキュメントの応答の停止、Flash ドキュメントの非常に一貫性のない動作など、大きな問題が発生します。次のコードに、無限ループの例を示します。

```
// 不正なコード : 無限ループを発生させる
// 自己責任において使用すること
var i:Number;
for (i = 0; i < 10; i--) {
  trace(i);
}
```

値 i が 0 に初期化され、終了条件が満たされるのは、i が 10 以上で、反復後に値 i がデクリメントされる場合です。ここで、明確なエラーがあることはすぐにわかります。ループの反復後に値 i がデクリメントされる場合は、終了条件が満たされることはありません。結果はコードを実行するコンピュータによって異なり、コードが失敗する速度は、CPU の処理速度および他の要因によって異なります。たとえば、あるコンピュータでは、エラーメッセージが表示されるまでにループが約 142,620 回実行されます。

次のエラーメッセージがダイアログボックスに表示されます。

```
A script in this movie is causing Flash Player to run slowly. If it continues to
  run, your computer may become unresponsive. Do you want to abort the script?
```

ループを操作する場合 (特に while ループおよび do..while ループを使用する場合)、ループが正しく終了でき、無限ループにならないことを常に確認してください。

ループ制御の詳細については、108 ページの「switch ステートメントの使用」を参照してください。

for ループの使用

for ループを使用すると、特定の範囲の値に対する変数の繰り返し処理を実行することができます。for ループは、一連の ActionScript ステートメントを繰り返す必要がある回数を正確にわかっている場合に便利です。このループは、ステージ上でムービークリップを特定の回数だけ繰り返す場合や、配列でループを実行し、その配列の各アイテム上でタスクを実行する場合に役立ちます。for ループは、ビルトインカウンタを使用してアクションを繰り返します。for ステートメントでは、カウンタおよびカウンタをインクリメントするステートメントは、すべて for ステートメントの一部です。for ステートメントは次の基本的な形式を使って記述します。

```
for (init; condition; update) {
  // ステートメント ;
}
```

for ステートメントには、3 つの式を指定する必要があります。それらの式は、初期値に設定する変数、ループがいつ終了するかを指定する条件ステートメント、および各ループで変数の値を変更する式です。たとえば、次のコードは 5 回ループします。変数 i は 0 で始まり、4 で終わります。出力は 0 ～ 4 の数値となり、各数値は個別の行に出力されます。

```
var i:Number;
for (i = 0; i < 5; i++) {
  trace(i);
}
```

次の例では、最初の式 (i = 0) が、反復の開始前に評価される初期化式になります。2 番目の式 (i < 5) は、ループの実行前に毎回チェックする条件です。3 番目の式 (i++) は、ループ実行後に毎回評価される " 再初期化式 " と呼ばれるものです。

for ループを作成するには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択します。
2. ステージ上のムービークリップを作成します。
3. [ライブラリ] パネルで、ムービークリップシンボルを右クリックし、コンテキストメニューから [リンケージ] を選択します。
4. [ActionScript に書き出し] チェックボックスをオンにして、[クラス] 入力フィールドに「libraryLinkageClassName」と入力します。[OK] をクリックします。
5. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を入力します。

```
var i:Number;
for (i = 0; i < 5; i++) {
  this.attachMovie("libraryLinkageClassName", "clip" + i + "_mc", i, {_x:(i *
  100)});
}
```

6. [制御]-[ムービープレビュー] を選択して、Flash Player でコードをテストします。

   ステージの上部で 5 つのムービークリップが複製されます。この ActionScript は、ライブラリのムービークリップシンボルを複製し、ステージ上の x 座標の 0、100、200、300、および 400 ピクセルにクリップを再配置します。ループは 5 回実行され、変数 i に値 0 ～ 4 が代入されます。ループの反復で、値 i が 4 にインクリメントされ、2 番目の式 (i < 5) が true ではなくなるので、ループが終了します。

なお、for ステートメント内では各式の後ろに、スペースを 1 つずつ入れるようにしてください。詳細については、『ActionScript 2.0 リファレンスガイド』の「for ステートメント」を参照してください。

for..in ループの使用

ムービークリップの子、オブジェクトのプロパティ、または配列のエレメントをループする (繰り返す) には、for..in ステートメントを使用します。子とは、他のムービークリップ、関数、オブジェクト、変数などのことです。for..in ループの一般的な使用方法として、タイムライン上のインスタンスをループするか、オブジェクト内のキーと値のペアをループします。オブジェクトをループすると、Web サービスや、テキストまたは XML ファイルなどの外部ドキュメントからどのようなデータが返るかを確認できるので、アプリケーションをデバッグするための効果的な方法となります。

たとえば、for...in ループを使用して、汎用オブジェクトのプロパティの繰り返し処理を実行できます。オブジェクトのプロパティは特定の順序では維持されないので、プロパティは予期しない順序で表示されます。

```
var myObj:Object = {x:20, y:30};
for (var i:String in myObj) {
  trace(i + ": " + myObj[i]);
}
```

このコードによって、[出力] パネルに次の情報が出力されます。

```
x: 20
y: 30
```

配列のエレメントの繰り返し処理を実行することもできます。

```
var myArray:Array = ["one", "two", "three"];
for (var i:String in myArray) {
  trace(myArray[i]);
}
```

このコードによって、[出力] パネルに次の情報が出力されます。

```
three
two
one
```

オブジェクトとプロパティの詳細については、40 ページの「オブジェクト (Object) データ型」を参照してください。

メモ	クラスがダイナミッククラスでない限り、オブジェクトがカスタムクラスのインスタンスである場合、そのプロパティの繰り返し処理を実行することはできません。ダイナミッククラスのインスタンスの場合であっても、動的に追加されるプロパティによってしか繰り返し処理を実行できません。

メモ	`for..in` ステートメントによって実行するステートメントブロックを囲む中カッコ (`{}`) は、実行するステートメントが 1 つしかない場合は必要ありません。

次の例では、`for..in` を使用してオブジェクトのプロパティに対して繰り返し処理を実行します。

for ループを作成するには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を入力します。

```
var myObj:Object = {name:"Tara", age:27, city:"San Francisco"};
var i:String;
for (i in myObj) {
  trace("myObj." + i + " = " + myObj[i]);
}
```

3. [制御]-[ムービープレビュー] を選択して、Flash Player でコードをテストします。

   SWF ファイルをテストすると、[出力] パネルに次のテキストが表示されます。

```
myObj.name = Tara
myObj.age = 27
myObj.city = San Francisco
```

`for..in` ループをクラスファイル (外部 ActionScript ファイル) 内に記述した場合、インスタンスメンバーをループ内で使用することはできませんが、静的メンバーはループ内で使用できます。ただし、クラスのインスタンスの FLA ファイル内に `for..in` ループを記述した場合、インスタンスメンバーは使用できますが、静的メンバーは使用できません。クラスファイルの記述の詳細については、179 ページ、第 6 章の「クラス」を参照してください。詳細については、『ActionScript 2.0 リファレンスガイド』の「`for..in statement`」を参照してください。

while ループの使用

while ステートメントを使用して、条件が満たされている間はアクションを繰り返すことができます。これは、条件が true である場合に繰り返す if ステートメントと同様です。

while ループは式を評価し、式が true である場合に、ループの本体で定義されているアクションを実行します。条件が true に評価される場合はステートメントを実行します。その後、ループの先頭に戻り、再び条件を評価します。条件が false に評価された場合、ステートメントはスキップされ、ループが終了します。コードブロックを何回ループする必要があるのかわからない場合に、while ループが非常に役立ちます。

たとえば、次のコードは [出力] パネルに数値を出力します。

```
var i:Number = 0;
while (i < 5) {
   trace(i);
   i++;
}
```

[出力] パネルには次の数字が出力されます。

```
0
1
2
3
4
```

for ループの代わりに while ループを使用する 1 つの短所は、while ループを使用すると無限ループを記述する可能性が高いことです。カウンタ変数をインクリメントする式を省略した場合、for ループのサンプルコードはコンパイルされませんが、while ループのサンプルコードはコンパイルされます。i をインクリメントする式がないと、ループは無限ループになります。

FLA ファイルで while ループを作成して使用するには、次のコードを使用します。

while ループを作成するには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択します。
2. [コンポーネント] パネルを開き、DataSet コンポーネントをステージにドラッグします。
3. プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) を開き、インスタンス名 "**users_ds**" を入力します。

4. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を入力します。

```
var users_ds:mx.data.components.DataSet;
//
users_ds.addItem({name:"Irving", age:34});
users_ds.addItem({name:"Christopher", age:48});
users_ds.addItem({name:"Walter", age:23});
//
users_ds.first();
while (users_ds.hasNext()) {
  trace("name:" + users_ds.currentItem["name"] + ", age:" +
   users_ds.currentItem["age"]);
  users_ds.next();
}
```

5. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   次の情報が [出力] パネルに表示されます。

```
name:Irving, age:34
name:Christopher, age:48
name:Walter, age:23
```

詳細については、『ActionScript 2.0 リファレンスガイド』の「while ステートメント」を参照してください。

do..while ループについて

do..while ステートメントを使用して、while ループと同様のループを定義することができます。ただし、do..while ループではコードブロックの下端で式が評価される (コードブロックの実行後にチェックされる) ので、ループは常に少なくとも 1 回は実行されます。条件の評価が true である場合のみ、ステートメントが実行されます。

次のコードでは、条件が満たされない場合でも生成される、簡単な do...while ループの例を示します。

```
var i:Number = 5;
do {
  trace(i);
  i++;
} while (i < 5);
// 出力 : 5
```

ループを使用する場合は、無限ループとならないようにする必要があります。do..while ループの条件が継続して true に評価される場合、Flash Player で警告が表示されるか、Flash Player をクラッシュさせる無限ループを作成してしまいます。ループする回数がわかっている場合は、代わりに for ループを使用します。do..while statement の詳細と例については、『ActionScript 2.0 リファレンスガイド』を参照してください。

ActionScript でのネストしたループの使用

次の例では、オブジェクトの配列を作成し、ネスト構造でそれぞれの値を表示する方法を示します。この例では、for ループを使用して配列の各アイテムをループし、for..in ループを使用して、ネストされたオブジェクトのキーと値のペアの繰り返し処理を実行する方法を示します。

他のループ内にループをネストするには :

1. 新しい Flash ドキュメントを作成します。
2. [ファイル]-[名前を付けて保存] を選択し、ドキュメントに loops.fla という名前を付けて保存します。
3. タイムラインのフレーム 1 に次のコードを追加します。

```
var myArr:Array = new Array();
myArr[0] = {name:"One", value:1};
myArr[1] = {name:"Two", value:2};
//
var i:Number;
var item:String;
for (i = 0; i < myArr.length; i++) {
   trace(i);
   for (item in myArr[i]) {
      trace(item + ": " + myArr[i][item]);
   }
   trace("");
}
```

4. [制御]-[ムービープレビュー] を選択してコードをテストします。

   次の情報が [出力] パネルに表示されます。

```
0
name: One
value: 1

1
name: Two
value: 2
```

   配列内のアイテム数はわかっているので、簡単な for ループを使って各アイテムをループできます。配列の各オブジェクトは異なる名前と値のペアを持つことができるので、for..in ループを使ってそれぞれの値の繰り返し処理を実行し、結果を [出力] パネルに表示します。

配列について

" 配列 " とは、構造内での位置を表す番号でプロパティが識別されるオブジェクトです。配列は、基本的にはアイテムのリストです。配列の各エレメントは、同じデータ型である必要はありません。数値、日付、ストリング、およびオブジェクトを組み合わせたり、配列インデックスにネストした配列を追加することもできます。

次の例では、月名の簡単な配列を示します。

```
var myArr:Array = new Array();
myArr[0] = "January";
myArr[1] = "February";
myArr[2] = "March";
myArr[3] = "April";
```

この配列は、次のように再記述することもできます。

```
var myArr:Array = new Array("January", "February", "March", "April");
```

または、次のような簡易シンタックスを書くことも可能です。

```
var myArr:Array = ["January", "February", "March", "April"];
```

配列はデータの構造のようなものです。配列をオフィスビルにたとえると、各フロアには異なるデータが含まれます。たとえば、3 階には " 経理 "、5 階には " 技術 " というデータが含まれるなどです。このために、他の配列を含めて、異なる種類のデータを1つの配列に格納することができます。このビルの各フロアは、複数の種類のコンテンツを含むことができます。たとえば、3 階を " エグゼクティブ " と " 経理 " が共有できます。

配列には " エレメント " が含まれます。これはビルの各フロアに相当します。各エレメントには数値の位置 (" インデックス ") があり、これにより、配列において各エレメントの位置を参照することができます。これは、ビルの各フロアに階を示す数があるのと似ています。各エレメントはデータ (数値、ストリング、ブール値、またはオブジェクトの配列) を格納するか、空にすることができます。

また、配列そのものを制御および修正することもできます。たとえば、技術部をビルの地下に移動することができます。配列を使用すると、値を移動し、配列のサイズを変更できます。これはビルを改装し、階を増やしたり廃止するのと似ています。このように、エレメントの追加や削除、および異なるエレメントへの値の移動が可能です。

ビル (配列) にはフロア (エレメント) があり、フロアには階数を示す値 (インデックス) を設定できます。また、各フロアには部署 (値) があります。

配列の修正の詳細については、126 ページの「配列の変更について」を参照してください。配列の使用の詳細とインデックスの詳細については、124 ページの「配列の使用」を参照してください。エレメントの追加と削除の詳細については、127 ページの「エレメントの追加と削除について」を参照してください。配列アクセス演算子の詳細については、142 ページの「ドット演算子と配列アクセス演算子の使用」を参照してください。

ActionScript を使用した配列の操作を示すサンプルのソースファイル "array.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Arrays" フォルダに移動してサンプルにアクセスします。このサンプルのコードでは、配列を作成し、2 つの List コンポーネントの項目をソート、追加、および削除しています。

配列の使用

作業で配列を使用するには、いくつか異なる方法があります。配列を使用すると、返されるアイテムなどのオブジェクトのリストを格納できます。リモート Web サーバーからデータをロードする場合、ネストされたオブジェクトの配列としてデータを受け取ることもできます。配列には、同じような形式でデータが含まれます。たとえば、Flash でオーディオアプリケーションを作成する場合、ユーザーの再生リストを音楽情報の配列として、オブジェクトに格納していることがあります。各オブジェクトには、曲名、アーティスト名、曲の長さ、サウンドファイル (MP3 など) の場所、または特定のファイルに関連付ける必要がある、その他の情報が含まれます。

配列内のアイテムの場所は、"インデックス" と呼ばれます。すべての配列はゼロベースです。つまり、配列内の第 1 のエレメントは [0] であり、第 2 のエレメントは [1] です (以下同様)。

配列には異なる種類がありますが、それについては以下のセクションで説明します。最も一般的な配列では、数値インデックスを使用して、"インデックス付き配列" で特定のアイテムを参照します。2 番目の種類の配列は "結合配列" と呼ばれ、数値インデックスではなくテキストインデックスを使って情報を参照します。一般的な配列の詳細については、123 ページの「配列について」を参照してください。結合配列の修正の詳細については、132ページの「結合配列の作成」を参照してください。多次元配列の詳細については、129ページの「多次元配列の作成」を参照してください。配列アクセス演算子の詳細については、142ページの「ドット演算子と配列アクセス演算子の使用」を参照してください。

Array ビルトインクラスを使用すると、配列にアクセスして操作することができます。Array オブジェクトを作成するには、コンストラクタ `new Array()` または配列アクセス演算子 (`[]`) を使用します。配列の各エレメントにアクセスする場合も、配列アクセス演算子 (`[]`) を使用します。次の例ではインデックス付き配列を使用します。

コードで配列を使用するには :

1. 新しい Flash ドキュメントを作成し、basicArrays.fla という名前を付けて保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
// 新しい配列を定義する
var myArr:Array = new Array();
// 2 つのインデックスで値を定義する
myArr[1] = "value1";
myArr[0] = "value0";
// 配列のアイテムの繰り返し処理を実行する
var i:String;
for (i in myArr) {
   // キーと値のペアをトレースする
   trace("key: " + i + ", value: " + myArr[i]);
}
```

   ActionScript の最初の行では、新しい値を定義して値を保持します。次に、配列の 2 つのインデックスでデータ (`value0` および `value1`) を定義します。`for..in` ループを使用して、その配列の各アイテムの繰り返し処理を実行し、trace ステートメントを使用して [出力] パネルにキーと値のペアを表示します。

3. [制御]-[ムービープレビュー] を選択してコードをテストします。

   次のテキストが [出力] パネルに表示されます。

```
key: 0, value: value0
key: 1, value: value1
```

   `for..in` ループの詳細については、118 ページの「for..in ループの使用」を参照してください。

異なる種類の配列を作成する方法については、以下のセクションを参照してください。

- 128 ページの「インデックス付き配列の作成」
- 129 ページの「多次元配列の作成」
- 132 ページの「結合配列の作成」

ActionScript を使用した配列の操作を示すサンプルのソースファイル "array.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Arrays" フォルダに移動してサンプルにアクセスします。このサンプルのコードでは、配列を作成し、2 つの List コンポーネントの項目をソート、追加、および削除しています。

配列の変更について

ActionScript を使用すると、配列を制御および修正することができます。配列の周囲で値を移動したり、配列のサイズを変更できます。たとえば、配列の 2 つのインデックスでデータを交換する場合は、次のコードを使用できます。

```
var buildingArr:Array = new Array();
buildingArr[2] = "Accounting";
buildingArr[4] = "Engineering";
trace(buildingArr); // undefined,undefined,Accounting,undefined,Engineering

var temp_item:String = buildingArr[2];
buildingArr[2] = buildingArr[4];
buildingArr[4] = temp_item;
trace(buildingArr); // undefined,undefined,Engineering,undefined,Accounting
```

この例で、なぜ一時変数を作成しなければならないのか、疑問に思うかもしれません。配列インデックス 4 の内容を配列インデックス 2 にコピーするか、その逆の場合、配列インデックス 2 の元の内容は失われます。1 つの配列インデックスから値を一時変数にコピーすると、値を保存し、後で安全にコードに戻すことができます。たとえば、次のコードを代わりに使用すると、配列 2 (Accounting) の配列の値が失われたことがわかります。その結果、2 つの技術チームが存在しますが、経理担当者は存在しなくなります。

```
// 不適切な方法（一時変数なし）
buildingArr[2] = buildingArr[4];
buildingArr[4] = buildingArr[2];
trace(buildingArr); // undefined,undefined,Engineering,undefined,Engineering
```

ActionScript を使用した配列の操作を示すサンプルのソースファイル "array.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Arrays" フォルダに移動してサンプルにアクセスします。このサンプルのコードでは、配列を作成し、2 つの List コンポーネントの項目をソート、追加、および削除しています。

長さの参照と検索について

配列を操作する場合、配列に存在するアイテムの数を知りたいことがあります。この情報は、配列の各エレメントの繰り返し処理を実行し、一連のステートメントを実行する for ループを作成するときに非常に役立ちます。次のコードに例を示します。

```
var monthArr:Array = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
   "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
trace(monthArr); // Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec
trace(monthArr.length); // 12
var i:Number;
for (i = 0; i < monthArr.length; i++) {
   monthArr[i] = monthArr[i].toUpperCase();
}
trace(monthArr); // JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
```

この例では、配列を作成し、それに月名を埋め込みます。内容が表示され、配列の長さも表示されます。for ループは配列の各アイテムの繰り返し処理を実行し、値を大文字に変換します。次に、配列の内容が再び表示されます。

次の ActionScript では、配列のインデックス 5 でエレメントを作成すると、返される配列の長さは 6 となり (配列がゼロベースであるため)、予期されるようにアイテムの実際の番号とはなりません。

```
var myArr:Array = new Array();
myArr[5] = "five";
trace(myArr.length); // 6
trace(myArr); // undefined,undefined,undefined,undefined,undefined,five
```

for ループの詳細については、117 ページの「for ループの使用」を参照してください。配列アクセス演算子の詳細については、142 ページの「ドット演算子と配列アクセス演算子の使用」を参照してください。

ActionScript を使用した配列の操作を示すサンプルのソースファイル "array.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Arrays" フォルダに移動してサンプルにアクセスします。このサンプルのコードでは、配列を作成し、2 つの List コンポーネントの項目をソート、追加、および削除しています。

エレメントの追加と削除について

配列にはエレメントが含まれ、各エレメントには数値の位置 (" インデックス ") が含まれます。これにより、配列において各エレメントの位置を参照することができます。各エレメントはデータを格納することも、空にすることもできます。エレメントは、数値、ストリング、ブール値、またはオブジェクトの配列を保持することができます。

配列でエレメントを作成するときは、可能な限り順番にインデックスを作成する必要があります。こうすることで、アプリケーションのデバッグが容易になります。127 ページの「長さの参照と検索について」では、インデックス 5 の配列に単一の値を代入すると、返される配列の長さは 6 となります。この結果、5 つの未定義の値が配列に挿入されます。

次の例では、新しい配列の作成、特定のインデックスでのアイテムの削除、および配列のインデックスでのデータの追加と置換の方法を示します。

```
var monthArr:Array = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
   "Aug", "Sep", "Oct", "Nov", "Dec");
delete monthArr[5];
trace(monthArr); // Jan,Feb,Mar,Apr,May,undefined,Jul,Aug,Sep,Oct,Nov,Dec
trace(monthArr.length); // 12
monthArr[5] = "JUN";
trace(monthArr); // Jan,Feb,Mar,Apr,May,JUN,Jul,Aug,Sep,Oct,Nov,Dec
```

配列インデックス 5 でアイテムを削除しましたが、配列の長さは 12 のままであり、配列インデックス 5 のアイテムは、完全に消えるのではなく空白のストリングに変更されました。

ActionScript を使用した配列の操作を示すサンプルのソースファイル "array.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Arrays" フォルダに移動してサンプルにアクセスします。このサンプルのコードでは、配列を作成し、2 つの List コンポーネントの項目をソート、追加、および削除しています。

インデックス付き配列の作成

インデックス付き配列には、1 つまたは複数の値が格納されます。前のセクションで行ったように、アイテムは配列での位置によって参照できます。最初のインデックスは常に数値 0 になり、それ以降に 1 つのエレメントを配列に追加するたびに、インデックスは 1 つずつインクリメントされます。インデックス付き配列を作成するには、Array クラスコンストラクタを呼び出すか、配列リテラルを使って配列を初期化します。次の例では、Array コンストラクタおよび配列リテラルを使って配列を作成します。

インデックス付き配列を作成するには :

1. 新しい Flash ドキュメントを作成し、indexArray.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

   ```
   var myArray:Array = new Array();
   myArray.push("one");
   myArray.push("two");
   myArray.push("three");
   trace(myArray); // one,two,three
   ```

   ActionScript の最初の行では、新しい配列を定義して値を保持します。
3. [制御]-[ムービープレビュー] を選択してコードをテストします。

   次のテキストが [出力] パネルに表示されます。

   ```
   one,two,three
   ```
4. オーサリングツールに戻り、[アクション] パネルでコードを削除します。

5. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var myArray:Array = ["one", "two", "three"];
trace(myArray); // one,two,three
```

このコードでは、配列リテラルを使って、コードの新しい配列を定義します。このコードは、手順 2 で書いた ActionScript と同等です。コードをテストすると、[出力] パネルに同じ情報が表示されます。

多次元配列の作成

ActionScript では、配列を " ネストされた配列 " として実装できます。ネストされた配列とは、基本的には配列の配列です。ネストされた配列は、" 多次元配列 " とも呼ばれ、マトリックスまたはグリッドと見なすことができます。そのため、プログラミングの際は、多次元配列を使用してこのような種類の構造を作成することができます。たとえば、チェスボードは 8 個の列と行で構成されるグリッドです。チェスボードは、8 個のエレメントを含み、それぞれが、8 個のエレメントを含む配列のモデルとなります。

たとえば、ストリングのインデックス付き配列として保存されるタスクのリストを考えてみます。

```
var tasks:Array = ["wash dishes", "take out trash"];
```

曜日ごとに異なるタスクのリストを保存する場合は、曜日ごとに 1 つのエレメントを持つ多次元配列を作成できます。各エレメントには、タスクのリストを格納するインデックス付き配列が含まれます。

注意	配列アクセス演算子を使用した場合、ActionScript のコンパイラは、アクセスされるエレメントがそのオブジェクトの有効なプロパティであるかどうかをチェックできません。

基本的な多次元配列を作成し、配列からエレメントを取得するには :

1. 新しい Flash ドキュメントを作成し、multiArray1.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var twoDArray:Array = new Array(new Array("one","two"), new Array("three",
   "four"));
trace(twoDArray);
```

この twoDArray 配列は、2 つの配列エレメントで構成されます。これらのエレメントそのものが、2 つのエレメントで構成される配列です。この場合、twoDArray は 2 つのネストされた配列を含むメイン配列です。

3. [制御]-[ムービープレビュー] を選択してコードをテストします。[出力] パネルに次の結果が表示されます。

```
one,two,three,four
```

4. オーサリングツールに戻り、[アクション] パネルを開きます。次に示すように、trace ステートメントをコメントアウトします。

```
// trace(twoDArray);
```

5. タイムラインのフレーム 1 のコードの末尾に次の ActionScript を追加します。

```
trace(twoDArray[0][0]); // 1
trace(twoDArray[1][1]); // 4
```

多次元配列のエレメントを取得するには、最上位の配列の名前の後に、複数の配列アクセス演算子 ([ ]) を使用します。最初の [ ] は、最上位の配列のインデックスを参照します。以降の配列アクセス演算子は、ネストされた配列のエレメントを参照します。

6. [制御]-[ムービープレビュー] を選択してコードをテストします。[出力] パネルに次の結果が表示されます。

```
one
four
```

ネストされた for ループを使用して、多次元配列を作成できます。次の例にその方法を示します。

for ループを使用して多次元配列を作成するには :

1. 新しい Flash ドキュメントを作成し、multiArray2.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var gridSize:Number = 3;
var mainArr:Array = new Array(gridSize);
var i:Number;
var j:Number;
for (i = 0; i < gridSize; i++) {
  mainArr[i] = new Array(gridSize);
  for (j = 0; j < gridSize; j++) {
    mainArr[i][j] = "[" + i + "][" + j + "]";
  }
}
trace(mainArr);
```

この ActionScript は、3 x 3 配列を作成し、それぞれの配列ノードの値をそのインデックスに設定します。次に、配列 (mainArr) をトレースします。

3. [制御]-[ムービープレビュー] を選択してコードをテストします。

[出力] パネルに次の結果が表示されます。

```
[0][0],[0][1],[0][2],[1][0],[1][1],[1][2],[2][0],[2][1],[2][2]
```

次の例に示すように、ネストされた for ループを使って、多次元配列のエレメントの繰り返し処理を実行することができます。

for ループを使用して多次元配列の繰り返し処理を実行するには :

1. 新しい Flash ドキュメントを作成し、multiArray3.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
// 前の例より
var gridSize:Number = 3;
var mainArr:Array = new Array(gridSize);
var i:Number;
var j:Number;
for (i = 0; i < gridSize; i++) {
  mainArr[i] = new Array(gridSize);
  for (j = 0; j < gridSize; j++) {
    mainArr[i][j] = "[" + i + "][" + j + "]";
  }
}
```

このコードでは、前の例のように、外側のループが mainArray の各エレメントの繰り返し処理を実行します。内側のループはネストされた配列の繰り返し処理を実行し、各配列ノードを出力します。

3. 手順 2 で入力したコードに従って、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
// エレメントの繰り返し処理を実行
var outerArrayLength:Number = mainArr.length;
for (i = 0; i < outerArrayLength; i++) {
  var innerArrayLength:Number = mainArr[i].length;
  for (j = 0; j < innerArrayLength; j++) {
    trace(mainArr[i][j]);
  }
}
```

この ActionScript は、配列のエレメントの繰り返し処理を実行します。各配列の length プロパティを、ループ条件として使用します。

4. [制御]-[ムービープレビュー] を選択して、[出力] パネルに表示されるエレメントを確認します。[出力] パネルに次の結果が表示されます。

```
[0][0]
[0][1]
[0][2]
[1][0]
[1][1]
[1][2]
[2][0]
[2][1]
[2][2]
```

配列の使用の詳細については、124 ページの「配列の使用」を参照してください。配列エレメントの詳細については、127 ページの「エレメントの追加と削除について」を参照してください。配列アクセス演算子の詳細については、142 ページの「ドット演算子と配列アクセス演算子の使用」を参照してください。

結合配列の作成

結合配列はオブジェクトに似ており、順序のない " キー " と " 値 " で構成されます。結合配列は数値のインデックスの代わりにキーを使って、格納された値を整理します。各キーは一意のストリングであり、1 つの値に関連付けてその値にアクセスするために使用されます。その値は、数値、配列、オブジェクトなどのデータ型とすることができます。キーに関連付けられた値を検索するコードを作成するときは、インデックスを付けるか " 参照 " を実行します。これが、結合配列を最もよく使用する目的です。

一般的に、キーと値の関連は " バインディング " と呼ばれます。キーと値はお互いに " マッピング " されます。たとえば、アドレス帳は、名前がキーとなり、電子メールアドレスが値となる、結合配列と見なすことができます。

メモ	結合配列は、キーと値の順序なしコレクションです。コードでは、結合配列のキーが特定の順序であることは想定しないでください。

結合配列を使用する場合は、数値ではなくストリングを使用して配列エレメントを呼び出すことができます。通常はこれが覚えやすい方法です。短所としては、インデックス値として数値を使用しないために、ループではあまり役立たないことです。ただし、キー値を使って頻繁に参照を行う必要がある場合は役立ちます。たとえば、何度も参照する必要がある名前と年齢の配列がある場合は、結合配列を使うことができます。

次の例では、オブジェクトを作成し、結合配列で一連のプロパティを定義する方法を示します。

簡単な結合配列を作成するには :

1. 新しい Flash ドキュメントを作成します。
2. タイムラインのフレーム 1 に次の ActionScript を入力します。

```
// 結合配列として使用するようにオブジェクトを定義する
var someObj:Object = new Object();
// 一連のプロパティを定義する
someObj.myShape = "Rectangle";
someObj.myW = 480;
someObj.myH = 360;
someObj.myX = 100;
someObj.myY = 200;
someObj.myAlpha = 72;
someObj.myColor = 0xDFDFDF;
// ドット演算子および配列アクセスシンタックスを使用してプロパティを表示する
trace(someObj.myAlpha); // 72
trace(someObj["myAlpha"]); // 72
```

最初の行の ActionScript では、関連配列として使用する新しいオブジェクト (`someObj`) を定義します。これに続いて、一連のプロパティを `someObj` で定義します。最後に、ドット演算子および配列アクセスシンタックスの両方を使用して、選択するプロパティを使用します。

> メモ
> 2 つの異なる方法を使って、結合配列の変数にアクセスできます。それらの方法は、ドットシンタックス (`someObj.myColor`) および配列シンタックス (`someObj['myColor']`) です。

3. [制御]-[ムービープレビュー] を選択して ActionScript をテストします。

   [出力] パネルには、数値 72 が 2 回表示されますが、これはトレースした両方のアルファレベルを表します。

ActionScript 2.0 では、結合配列を作成するための 2 つの方法があります。

- Object コンストラクタを使用する
- Array コンストラクタを使用する

これらのいずれの方法についても、後の例で示します。

> メモ
> 前述の例では、Object コンストラクタを使って結合配列を作成しました。

Object コンストラクタを使って結合配列を作成する場合は、オブジェクトリテラルを使用して配列を初期化するという利点が得られます。Object クラスのインスタンスは汎用オブジェクトとも呼ばれますが、機能的には結合配列と同じです。Object インスタンスは基本的には結合配列です。辞書のような機能では、数値のインデックスよりもストリングキーの方が便利な場合に、結合配列を使用します。汎用オブジェクトのプロパティ名は、格納されている値にアクセスするためのキーとなります。リテラルの詳細については、93 ページの「リテラルについて」を参照してください。クラスの詳細については、179 ページ、第 6 章の「クラス」を参照してください。

Object コンストラクタを使用して結合配列を作成するには :

1. 新しい Flash ドキュメントを作成し、assocArray.fla という名前を付けて保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var monitorInfo:Object = {type:"Flat Panel", resolution:"1600 x 1200"};
trace(monitorInfo["type"] + ", " + monitorInfo["resolution"]);
```

   このコードは、`monitorInfo` という名前の結合配列を作成し、オブジェクトリテラルを使用して、2 つのキーと値のペアで配列を初期化します。

> メモ
> 宣言時に配列を初期化する必要がある場合は、Object コンストラクタを使って配列を作成できます。

```
var monitorInfo:Object = new Object();
```

3. [制御]-[ムービープレビュー] を選択します。

   [出力] パネルに次のテキストが表示されます。

```
Flat Panel, 1600 x 1200
```

4. 前に入力したコードに従って、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
monitorInfo["aspectRatio"] = "16:10";
monitorInfo.colors = "16.7 million";
trace(monitorInfo["aspectRatio"] + ", " + monitorInfo.colors);
```

オブジェクトリテラルまたは Object クラスコンストラクタのいずれかを使用して配列を作成した後は、このコードに示すように、角カッコ ([ ]) またはドット演算子 (.) を使って配列に新しい値を追加できます。入力したコードでは、2 つの新しい値が monitorInfo 配列に追加されます。

5. [制御]-[ムービープレビュー] を選択します。

[出力] パネルに次のテキストが表示されます。

```
16:10, 16.7 million
```

キーには空白を含めることができます。角カッコを含めることは可能ですが、ドット演算子を含めるとエラーが発生します。キー名にスペースを使用することはお勧めしません。角カッコおよびドット演算子の詳細については、135 ページの「演算子について」を参照してください。正しくフォーマットされたコードの詳細については、675 ページの「ActionScript シンタックスのフォーマット」を参照してください。

結合配列を作成する 2 番目の方法は、Array コンストラクタを使用し、角括弧 ([ ]) またはドット演算子 (.) を使ってキーと値のペアを配列に追加します。結合配列を Array 型として宣言する場合は、オブジェクトリテラルを使って配列を初期化することはできません。

メモ	Array コンストラクタを使って結合配列を作成する利点はありません。Array コンストラクタは、インデックス付き配列を作成するのに最適です。

次の例では、Array コンストラクタを使って結合配列を作成する方法を示します。

Array コンストラクタを使用して結合配列を作成するには :

1. 新しい Flash ドキュメントを作成し、"assocArray2.fla" という名前を付けて保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var monitorInfo:Array = new Array();
monitorInfo["type"] = "Flat Panel";
monitorInfo["resolution"] = "1600 x 1200";
trace(monitorInfo["type"] + ", " + monitorInfo["resolution"]);
```

このコードは、Array コンストラクタを使って monitorInfo という名前の結合配列を作成し、type という名前のキーおよび resolution という名前のキーを、値と共に追加します。

3. [制御]-[ムービープレビュー] を選択します。

[出力] パネルに次のテキストが表示されます。

```
Flat Panel, 1600 x 1200
```

メモ	Array コンストラクタを使って結合配列を作成する利点はありません。Array コンストラクタは、インデックス付き配列を作成するのに最適です。

結合配列は基本的には Object クラスのインスタンスであり、Array コンストラクタを使って結合配列を作成する利点はありません。結合配列の作成には `new Array()` コンストラクタが使用されますが、結合配列の使用時に Array クラスのメソッドとプロパティ (`sort()` や `length` など) を使用することはできません。数値インデックスではなくキーと値のペアを使用するには、結合配列の代わりに Object クラスを使用する必要があります。

演算子について

このセクションでは、よく使用される演算子の一般的な規則、演算子の優先順位、および結合性を説明します。

演算子は式の値の組み合わせ、比較、または修正の方法を指定する文字です。式とは Flash で評価できるステートメントで、値を返します。式を作成するには、演算子と値を組み合わせるか、関数を呼び出します。式の詳細については、76 ページの「シンタックス、ステートメント、および式について」を参照してください。

たとえば、数式では数値演算子を使って、使用する値を操作します。演算子の例には、+、<、*、および = があります。式は演算子と " オペランド " で構成され、値を表す ActionScript シンボルを規則に従って組み合わせたものです。オペランドは、演算子が演算を実行するコードの一部です。たとえば、式 `x + 2` では、x と 2 がオペランドであり、+ が演算子です。

コードでは、式と演算子を頻繁に使用します。演算子と値を組み合わせて式を作成するか、関数を呼び出すことができます。

メモ	このセクションでは、それぞれの種類の演算子の使用方法について説明します。ただし、紙面の都合で十分な説明はできません。各演算子の詳細については、次のカテゴリに含まれない特殊演算子も含め、『ActionScript 2.0 リファレンスガイド』を参照してください。

演算子が実行する演算のコード部分は " オペランド " と呼ばれます。たとえば、加算 (+) 演算子を使用すると、数値リテラルの値を追加できます。これは、`myNum` という名前の変数の値を追加するために行います。

```
myNum + 3;
```

この例では、`myNum` および 3 がオペランドです。

このセクションでは、よく使用される演算子の一般的な規則、演算子の優先順位、および結合性を説明します。

- 136 ページの「演算子を使用した値の操作」
- 138 ページの「演算子の優先順位と結合性について」
- 140 ページの「演算子とストリングの使用について」
- 142 ページの「ドット演算子と配列アクセス演算子の使用」
- 143 ページの「後置演算子について」
- 144 ページの「単項演算子について」

■ 145 ページの「乗算演算子について」
■ 145 ページの「加算演算子について」
■ 145 ページの「数値演算子の使用」
■ 147 ページの「関係演算子について」
■ 147 ページの「等価演算子について」
■ 147 ページの「関係演算子および等価演算子の使用」
■ 150 ページの「代入演算子について」
■ 151 ページの「代入演算子の使用」
■ 151 ページの「論理演算子について」
■ 152 ページの「論理演算子の使用」
■ 153 ページの「ビット単位シフト演算子について」
■ 153 ページの「ビット論理演算子について」
■ 154 ページの「ビット演算子の使用について」
■ 155 ページの「条件演算子について」
■ 156 ページの「ドキュメントでの演算子の使用」

これらのカテゴリに含まれない演算子の詳細については、『ActionScript 2.0 リファレンスガイド』を参照してください。このマニュアルには、使用できるすべての演算子に関する情報が記載されています。

以下のセクションでは、演算子の一般的な使用方法をいくつか示します。1 つのサンプルコードで数多くの演算子を使用する場合の詳細については、156 ページの「ドキュメントでの演算子の使用」を参照してください。

演算子を使用した値の操作

一般的に、Flash で値を操作するには演算子を使用します。たとえば、ステージ上のインスタンスとユーザーの操作に応じて、得点が変わるゲームを Flash で作成するとします。変数を使用して値を格納し、演算子を使用して変数の値を操作することができます。

たとえば、`myScore` という名前の変数の値を増やすとします。次のコードでは、+ (加算) 演算子および += (加入後代入) 演算子を使って、コードで値を加算およびインクリメントする方法を示します。

演算子を使用して値を操作するには :

1. 新しい Flash ドキュメントを作成します。
2. [アクション] パネルを開き ([ウィンドウ]-[アクション])、スクリプトペインに次のコードを入力します。

```
// 例 1
var myScore:Number = 0;
myScore = myScore + 1;
trace("Example one: " + myScore); // 1

// 例 2
var secondScore:Number = 1;
secondScore += 3;
trace("Example two: " + secondScore); // 4
```

3. [制御]-[ムービープレビュー] を選択します。

   [出力] パネルに次のテキストが表示されます。

```
Example one: 1
Example two: 4
```

   加算演算子は 2 つの値を加算するので、比較的単純です。最初のサンプルコードでは、myScore の現在の値および値 1 を追加し、結果を変数 myScore に格納します。

2 番目のサンプルコードは、加算後代入演算子を使って、1 つの手順で新しい値を追加および代入します。行 myScore = myScore + 1 (前の例より) は、myScore++ として再記述するか、myScore += 1 として再記述できます。インクリメント演算子 (++) は、インクリメントと代入を同時に処理するので、myScore = myScore + 1 と記述するための簡単な方法です。次の ActionScript に、インクリメント演算子の例を示します。

```
var myNum:Number = 0;
myNum++;
trace(myNum); // 1
myNum++;
trace(myNum); // 2
```

このコードには、代入演算子がありません。その代わり、インクリメント演算子に依存します。

条件が true である間に、演算子を使って変数の値を操作できます。たとえば、インクリメント演算子 (++) を使用すると、条件が true である間に変数 i をインクリメントできます。次のコードでは、i が 10 未満のときに、条件が true となります。true のときは、i++ を使用して i の数値を 1 つずつインクリメントできます。

```
var i:Number;
for (i = 1; i < 10; i++) {
   trace(i);
}
```

[出力] パネルには数値 1 ～ 9 が表示されますが、これは i がインクリメントしているもので、終了条件 (i が 10 に等しい) に達すると停止します。最後に表示される値は 9 です。そのため、SWF ファイルが再生を開始すると i の値は 1 になり、トレースが完了すると 9 になります。

条件とループの詳細については、102 ページの「ステートメントについて」を参照してください。

演算子の優先順位と結合性について

ステートメントで複数の演算子を使用する場合、他の演算子に優先される演算子があります。演算子の優先順位と結合性により、演算子を処理する順序が決まります。ActionScript には、どの演算子が他の演算子よりも先に実行されるかを指定する優先順位があります。このセクションの最後に、この優先順位を示した表が記載されています。

算術プログラミングまたは基本的なプログラミングに慣れている開発者にとって、コンパイラが乗算 (*) 演算子を加算 (+) 演算子より先に処理するのは自然なことですが、コンパイラは、どの演算子を最初に処理するかについて明示的な指示を必要とします。このような指示を、総称して " 演算子の優先順位 " と呼びます。

次の例では、算演算子および加算演算子を操作し、演算子の優先順位について確認できます。

```
var mySum:Number;
mySum = 2 + 4 * 3;
trace(mySum); // 14
```

このステートメントの出力は 14 ですが、これは乗算の優先順位が高いためです。そのため、4 * 3 が最初に評価され、その結果が 2 に追加されます。

式をカッコで囲むと、結果を制御できます。ActionScript では、カッコ (( )) を使用して変更できる、演算子のデフォルトの優先順位が定義されています。加算部分がカッコで囲まれていれば、加算が最初に実行されます。

```
var mySum:Number;
mySum = (2 + 4) * 3;
trace(mySum); // 18
```

このステートメントの出力は 18 になります。

演算子に同じ優先順位を設定することもできます。この場合、結合性により、演算子の実行順序が決まります。結合性は左から右、または右から左のいずれかです。

再び乗算演算子について考えてみます。乗算演算子には左から右の結合性があるので、次の 2 つのステートメントは同じになります。

```
var mySum:Number;
var myOtherSum:Number;
mySum = 2 * 4 * 3;
myOtherSum = (2 * 4) * 3;
trace(mySum); // 24
trace(myOtherSum); // 24
```

同じ優先順位の複数の演算子が同じ式にある状況があります。このような状況では、コンパイラは " 結合性 " のルールを使用して、最初に処理する演算子を決定します。代入演算子を除くすべてのバイナリ演算子は " 左結合 " となります。つまり、左にある演算子が右にある演算子よりも先に処理されます。代入演算子と条件 (`?:`) 演算子は " 右結合 " となります。つまり、右にある演算子が左にある演算子よりも先に処理されます。代入演算子の詳細については、151 ページの「代入演算子の使用」を参照してください。条件 (?) 演算子の詳細については、155 ページの「条件演算子について」を参照してください。

たとえば、" より小さい " (<) および " より大きい " (>) 演算子について考えてみます。これらの演算子の優先順位は同じです。両方の演算子を同じ式で使用すると、左の演算子が最初に処理されます。これは、両方の演算子が左結合であるためです。つまり、次の 2 つのステートメントでは同じ出力が得られます。

```
trace(3 > 2 < 1);   // false
trace((3 > 2) < 1); // false
```

" より大きい " (>) 演算子が最初に処理され、値は `true` となります。これは、オペランド 3 がオペランド 2 より大きいためです。次に、値 `true` がオペランド 1 と共に " より小さい " (<) 演算子に渡されます。" より小さい " (<) 演算子は、値 `true` を数値 1 に変換し、その数値を 2 番目のオペランド 1 と比較して、値 `false` を返します。これは、値 1 が 1 より小さくないためです。

ActionScript では、オペランドの優先順位を考慮してください。特に、複雑な条件を設定し、それらの条件の 1 つが何回 true になるかわかっている場合は、これが重要になります。たとえば、条件で `i` が 50 より大きくなる場合、最初に `i<50` を記述する必要があります。そのため、この条件が最初にチェックされます。記述する 2 番目の条件は、それほど頻繁にチェックされる必要はありません。

次の表に、ActionScript 2.0 の演算子を優先順位の高いものから順に示します。表の各行には、同じ優先順位の演算子が含まれています。表内の演算子は、上位の列に含まれるものほど優先順位が高くなります。演算子と括弧の使用に関する詳細とガイドラインについては、675 ページ、第 17 章の「ActionScript シンタックスのフォーマット」を参照してください。

グループ	演算子
プライマリ	`[] {x:y} () f(x) new x.y x[y]`
後置	`x++ x--`
単項	`++x --x + - ~ ! delete typeof void`
乗算	`* / %`
加算	`+ -`
ビット単位シフト	`<< >> >>>`
関係	`< > <= >= instanceof`
等しい	`== != === !==`

グループ	演算子
ビット単位の論理積 (AND)	`&`
ビット単位の排他的論理和 (XOR)	`^`
ビット単位の論理和 (OR)	`|`
論理積 (AND)	`&&`
論理和 (OR)	`||`
三項演算	`?:`
代入	`= *= /= %= += -= <<= >>= >>>= &= ^= |=`
カンマ	`,`

演算子とストリングの使用について

2 つのデータ型が異なる場合、比較演算子は 1 つのオペランドをもう一方のデータ型に変換します。1 つのオペランドがストリングでもう一方が数値の場合、ストリングオペランドが数値に変換され、それらの数値比較が実行されます。このルールの例外として厳密な等価 (===) 演算子があります。厳密な等価演算子の動作は、データ型が変換されない以外は等価 (==) 演算子と同じです。両方の式が、そのデータ型も含めて等しい場合、結果は `true` です。数値演算子の詳細については、145 ページの「数値演算子の使用」を参照してください。

等価演算子 (==) と比較演算子 (>、 >=、<、および <=) によるストリングへの影響は、独自の値を操作するときは異なります。

比較演算子は、ストリングを比較してアルファベット順に最初のストリングを判断します。大文字のストリングは、小文字のストリングに優先されます。つまり、"Egg" が "chiken" よりも先になります。

```
var c:String = "chicken";
var e:String = "Egg";
trace(c < e); // false
var riddleArr:Array = new Array(c, e);
trace(riddleArr); // chicken,Egg
trace(riddleArr.sort()); // Egg,chicken
```

この ActionScript では、Array クラスの `sort()` メソッドが、配列の内容の順序をアルファベット順に再変更します。値 “Egg” が値 “chicken” よりも先になるのは、大文字の E が小文字の c よりも先になるためです。大文字と小文字とは無関係にストリングを比較する場合は、ストリングを比較する前に大文字または小文字に変換する必要があります。比較演算子の詳細については、147 ページの「等価演算子について」および 147 ページの「関係演算子および等価演算子の使用」参照してください。

toLowerCase() または toUpperCase() メソッドを使用して、ストリングを比較する前に同じ大文字または小文字に変換することができます。次の例では、両方のストリングが小文字に変換されて比較されます。この結果、chiken が egg よりも先になります。

```
var c:String = "chicken";
var e:String = "Egg";
trace(c.toLowerCase() < e.toLowerCase()); // true
```

メモ

比較演算子では、2 つのストリングのみが比較されます。たとえば、1 つのオペランドが数値である場合、演算子は値を比較しません。オペランドのどちらか一方だけがストリングである場合、両方のオペランドが数値に変換され、数値として比較されます。

演算子を使用してストリングを操作できます。加算 (+) 演算子を使用して、2 つのストリングオペランドを連結することができます。trace ステートメントを書いたときに、加算演算子を使用してストリングを連結しました。たとえば、次のように入力するとします。

```
var myNum:Number = 10;
trace("The variable is " +  myNum + ".");
```

このコードは、[出力] パネルに次のような情報を出力します。

```
The variable is 10.
```

この例では、trace ステートメントは + 演算子を使用して加算ではなく連結を行います。Flash でストリングと数値を扱う場合、数値として加算するのではなく、連結される場合があります。

たとえば、異なる変数の 2 つのストリングを 1 つのテキストフィールドに連結するとします。次の ActionScript コードでは、変数 myNum がストリングと連結し、ステージ上の [myTxt] テキストフィールドにストリングが表示されます。

```
this.createTextField("myTxt", 11, 0, 0, 100, 20);
myTxt.autoSize = "left";
var myNum:Number = 10;
myTxt.text = "One carrot. " + myNum + " large eggplants.";
myTxt.text += " Lots of vegetable broth.";
```

このコードは、次の情報をインスタンス名 myTxt と共にテキストフィールドに出力します。

```
One carrot. 10 large eggplants. Lots of vegetable broth.
```

この例では、加算 (+) 演算子および加算代入 (+=) 演算子を使用してストリングを連結する方法を示します。3 行目のコードでは、加算演算子を使用して myNum 変数の値をテキストフィールドに連結し、4 行目のコードでは加算代入演算子を使用して、テキストフィールドの既存の値にストリングを連結します。

テキストストリングのオペランドの 1 つのみが実際にストリングである場合、Flash は他のオペランドをストリングに変換します。そのため、値 myNum は前述の例のストリングに変換されます。

メモ

ストリングの先頭、または末尾に入力されたスペースは、ストリングのリテラル部分として扱われます。

ドット演算子と配列アクセス演算子の使用

ドット演算子 (.) と配列アクセス演算子 ([ ]) を使用すると、ActionScript のビルトインプロパティや、カスタムプロパティにアクセスすることができます。ドット演算子は、オブジェクトの特定のインデックスをターゲットにするために使用します。たとえば、次の ActionScript に示すように、ユーザー情報が含まれたオブジェクトがある場合は、配列アクセス演算子で特定のキー名を指定して、ユーザー名を取得することができます。

```
var someUser:Object = {name:"Hal", id:2001};
trace("User's name is: " + someUser["name"]); // ユーザー名 : Hal
trace("User's id is: " + someUser["id"]); // ユーザー ID : 2001
```

たとえば、次の ActionScript は、ドット演算子を使用してオブジェクト内で特定のプロパティを設定します。

```
myTextField.border = true;
year.month.day = 9;
myTextField.text = "My text";
```

ドット演算子および配列アクセス演算子は非常に似ています。ドット演算子では、識別子がプロパティとして見なされますが、配列アクセス演算子では、その内容が名前として評価され、その名前が付いたプロパティの値にアクセスします。配列アクセス演算子を使用すると、インスタンス名および変数を動的に設定および取得できます。

配列アクセス演算子は、オブジェクトに含まれているキーが完全にわからない場合に役立ちます。この場合は、for..in ループを使ってオブジェクトまたはムービークリップの繰り返し処理を実行し、内容を表示できます。

ドット演算子と配列アクセス演算子を使用するには :

1. 新しい Flash ドキュメントで、メインタイムラインにムービークリップを作成します。
2. ムービークリップを選択して、プロパティインスペクタを開きます。
3. myClip のインスタンス名を入力します。
4. タイムラインのフレーム 1 に次の ActionScript を追加します。

   ```
   myClip.spam = 5;
   trace(myClip.spam); // 5
   ```

   現在のタイムラインで myClip インスタンスに値を設定する場合は、この ActionScript に示すように、ドット演算子または配列アクセス演算子を使用します。配列アクセス演算子内で式を記述する場合、その式が最初に評価され、結果が変数名として使用されます。
5. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   [出力] パネルに "5" と表示されます。
6. オーサリング環境に戻り、ActionScript の最初の行を次のコードで置き換えます。

   ```
   myClip["spam"] = 10;
   ```

7. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   [出力] パネルに "10" と表示されます。

8. オーサリング環境に戻り、myClip インスタンスをダブルクリックします。

9. 新しい 4 つのインスタンスを myClip インスタンスの内部に追加します。

10. プロパティインスペクタを使用して、4 つの新しい各インスタンスに、nestedClip1、nestedClip2、nestedClip3、および nestedClip4 のインスタンス名を追加します。

11. メインタイムラインのフレーム 1 に次のコードを追加します。

```
var i:Number;
for (i = 1; i <= 4; i++) {
  myClip["nestedClip" + i]._visible = false;
}
```

   この ActionScript は、ネストされた各ムービークリップの表示と非表示を切り替えます。

12. [制御]-[ムービープレビュー] を選択して、追加した ActionScript をテストします。

   ネストされた 4 つのインスタンスが非表示になります。配列アクセス演算子を使用して、myClip インスタンスでネストされた各ムービークリップの繰り返し処理を実行し、可視性のプロパティを動的に設定しています。各インスタンスを特にターゲットにする必要がないので、時間を節約することができます。

代入の左側で配列アクセス演算子を使用して、インスタンス、変数、およびオブジェクト名を動的に設定することもできます。

```
myNum[i] = 10;
```

ActionScript 2.0 では、動的に作成されるオブジェクトのクラス定義が dynamic 属性で指定されない場合、角カッコを使用して、そのオブジェクトのプロパティにアクセスします。また、この演算子を使用して多次元配列を作成することもできます。配列アクセス演算子を使用した多次元配列の作成の詳細については、129 ページの「多次元配列の作成」を参照してください。

後置演算子について

後置演算子は、1 つの演算子を取り、演算子の値をインクリメントまたはデクリメントします。これらの演算子は単項演算子ですが、優先順位が高いことと、その特別なビヘイビアにより、他の単項演算子からは区別して分類されます。単項演算子の詳細については、144 ページの「単項演算子について」を参照してください。

後置演算子をより大きい式の一部として使用する場合、後置演算子が処理される前に、式の値が返されます。たとえば、次のコードでは、式 xNum++ の値がインクリメントされる前に返されます。

```
var xNum:Number = 0;
trace(xNum++); // 0
trace(xNum); // 1
```

このコードをトレースすると、[出力] パネルに表示されるテキストは次のようになります。

```
0
1
```

この表の演算子の優先順位は同じです。

演算子	実行される演算
`++`	インクリメント (後置方式)
`--`	デクリメント (後置方式)

単項演算子について

単項演算子は 1 つのオペランドを取ります。このグループのインクリメント (++) 演算子とデクリメント (--) 演算子は " 前置 " 演算子であり、式ではオペランドの前に表示されます。オペランドの後に表示されることもありますが、その場合は " 後置 " 演算子と呼ばれます。後置演算子の詳細については、143 ページの「後置演算子について」を参照してください。

前置演算子が後置演算子と異なるのは、インクリメント処理またはデクリメント処理が、式全体が返される前に完了することです。たとえば、次のコードでは、式 `xNum++` の値がインクリメントされた後に返されます。

```
var xNum:Number = 0;
trace(++xNum); // 1
trace(xNum); // 1
```

この表の演算子の優先順位はすべて同じです。

演算子	実行される演算
`++`	インクリメント (前置方式)
`--`	デクリメント (前置方式)
`+`	単項プラス
`!`	単項マイナス
`typeof`	タイプ情報を返す
`void`	未定義の値を返す

乗算演算子について

乗法演算子は 2 つのオペランドを取り、乗算、除算、または剰余計算を実行します。他の数値演算子には加算演算子が含まれます。加算演算子の詳細については、145 ページの「加算演算子について」を参照してください。

この表の演算子の優先順位はすべて同じです。

演算子	実行される演算
*	乗算
/	除算
%	剰余

乗算演算子の使用の詳細については、145 ページの「数値演算子の使用」を参照してください。

加算演算子について

加算演算子は 2 つのオペランドを取り、加算または減算を実行します。他の数値演算子には乗算演算子が含まれます。乗算演算子の詳細については、145 ページの「乗算演算子について」を参照してください。

この表の演算子の優先順位は同じです。

演算子	実行される演算
+	加算
-	減算

加算演算子の使用の詳細については、145 ページの「数値演算子の使用」を参照してください。

数値演算子の使用

ActionScript で値を加算、減算、除算、または乗算するには、数値演算子を使用します。異なる種類の算術操作を実行できます。最も一般的な演算子はインクリメント演算子であり、一般的には i++ のように指定します。この演算子では、さらに実行できる操作があります。インクリメント演算子の詳細については、136 ページの「演算子を使用した値の操作」を参照してください。

インクリメント演算子は、オペランドの前 ("事前インクリメント") または後 ("事後インクリメント") に追加できます。

ActionScript の数値演算子を理解するには :

1. 新しい Flash ドキュメントを作成します。
2. 次の ActionScript をタイムラインのフレーム 1 に入力します。

```
// 例 1
var firstScore:Number = 29;
if (++firstScore >= 30) {
   // トレースが必要
   trace("Success! ++firstScore is >= 30");
}
// 例 2
var secondScore:Number = 29;
if (secondScore++ >= 30) {
   // トレースは不要
   trace("Success! secondScore++ is >= 30");
}
```

3. [制御]-[ムービープレビュー] を選択して ActionScript をテストします。

   “ 例 1” のコードブロックはトレースしますが、“ 例 2” のコードブロックはトレースしません。最初の例では、事前インクリメント (++firstScore) を使ってインクリメントし、値が 30 以上かどうかをテストする前に firstScore を計算します。そのため、firstScore は 30 にインクリメントしてから、30 以上かどうかがテストされます。

   しかし、例 2 では事後インクリメント (secondScore++) を使用し、テスト後に評価が実行されます。そのため、29 が 30 に対して比較され、評価後に 30 にインクリメントされます。

加算演算子を操作する場合、次の例のように、式で値を追加しようとすると、予期しない結果が生じることがあります。

```
trace("the sum of 5 + 2 is: " + 5 + 2); // 5 + 2 の合計 : 52
```

値 5 および 2 が加算されるのではなく、連結されます。この問題を回避するには、次のコードのように、式 5+2 をカッコで囲みます。

```
trace("the sum of 5 + 2 is: " + (5 + 2)); // 5 + 2 の合計 : 7
```

演算子の優先順位の詳細については、138 ページの「演算子の優先順位と結合性について」を参照してください。

外部ソース (XML ファイル、FlashVars、Web サービスなど) からデータをロードする場合は、数値演算子は非常に慎重に操作する必要があります。SWF ファイルは数値のデータ型を認識しないので、数値がストリングのように扱われることがあります。この場合は、3 および 7 を加算し、結果が 37 になります。これは、両方の数値が数値として加算されず、ストリングのように連結されるためです。この状況では、Number( ) 関数を使用して、ストリングから数値にデータを手動で変換する必要があります。

関係演算子について

関係演算子は 2 つのオペランドを取り、値を比較してブール値を返します。この表の演算子の優先順位はすべて同じです。

演算子	実行される演算
`<`	より小さい
`>`	より大きい
`<=`	より小さいか等しい
`>=`	より大きいか等しい
`instanceof`	プロトタイプチェーンをチェックする
`in`	オブジェクトプロパティをチェックする

関係演算子の使用の詳細については、147 ページの「関係演算子および等価演算子の使用」を参照してください。

等価演算子について

等価演算子は 2 つのオペランドを取り、値を比較してブール値を返します。この表の演算子の優先順位はすべて同じです。

演算子	実行される演算
`==`	等しい
`!=`	不等価
`===`	厳密な等価
`!==`	厳密な不等価

等価演算子の使用の詳細については、147 ページの「関係演算子および等価演算子の使用」を参照してください。

関係演算子および等価演算子の使用

関係演算子と等価演算子は " 比較演算子 " とも呼ばれ、式の値を比較し、`true` または `false` (ブール値) を返します。条件ステートメントおよびループでは比較演算子を頻繁に使用して、ループを停止する条件を指定します。

等価 (==) 演算子を使用して、2 つのオペランドの値や参照が等しいかどうかを調べることができます。また、この比較ではブール値が返されます。ストリング、数値、またはブール値のオペランド値は、値を使用して比較します。オブジェクトおよび配列のオペランドは、参照によって比較されます。

この例では、等価演算子を使用して配列の長さをテストし、配列にアイテムがない場合は [出力] パネルにメッセージを表示することができます。

```
var myArr:Array = new Array();
if (myArr.length == 0) {
  trace("the array is empty.");
}
```

[制御]-[ムービープレビュー] を選択すると、the array is empty というストリングが [出力] パネルに表示されます。

等価演算子を使用して値を比較することができますが、等価演算子を使用して値を設定することはできません。代入演算子 (=) を使用して値が等しいかどうかをチェックすることができます。

コードで関係演算子および等価演算子を使用するには :

1. 新しい Flash ドキュメントを作成します。
2. 次の ActionScript をタイムラインのフレーム 1 に入力します。

   ```
   var myNum:Number = 2;
   if (myNum == 2) {
     // 何らかの処理をする。
     trace("It equals 2");
   }
   ```

   この ActionScript では、等価演算子 (==) を使用して値が等しいかどうかをチェックします。変数 myNum が 2 に等しいかどうかをチェックします。
3. [制御]-[ムービープレビュー] を選択します。

   [出力] パネルにストリング It equals 2 が表示されます。
4. オーサリング環境に戻って次のコードを変更します。

   ```
   var myNum:Number = 2;
   ```

   このコード行を次のように変更します。

   ```
   var myNum:Number = 4;
   ```
5. [制御]-[ムービープレビュー] を再び選択します。

   [出力] パネルにはストリング It equals 2 が表示されません。
6. オーサリング環境に戻って次のコードを変更します。

   ```
   if (myNum == 2) {
   ```

   このコード行を次のように変更します。

   ```
   if (myNum = 2) {
   ```

7. [制御]-[ムービープレビュー] を再び選択します。

   [出力] パネルにストリング `It equals 2` が再び表示されます。

   手順 6 では、`myNum` を 2 と比較するのではなく、2 を `myNum` に代入しました。この場合、`myNum` の前の値とは無関係に `if` ステートメントが実行され、Flash ドキュメントをテストするときに予期しない結果が発生する可能性があります。

   代入演算子の正しい使用に関する詳細については、151 ページの「代入演算子の使用」を参照してください。

厳密な等号演算子 (===) は等価演算子と似ていますが、型変換が実行されないことが異なります。2 つのオペランドの型が異なる場合、等価演算子は `false` を返します。厳密な不等価演算子 (!==) は、厳密な等価演算子とは逆の結果を返します。

次の ActionScript に、等価演算子 (==) と厳密な等価演算子 (===) の主な違いを示します。

```
var num1:Number = 32;
var num2:String = new String("32");
trace(num1 == num2); // true
trace(num1 === num2); // false
```

最初に、数値変数 `num1` および `num2` を定義します。等価演算子を使用して変数を比較する場合、同じデータ型に値の変換が試みられてから、値を比較して等しいかどうかが確認されます。厳密な等号演算子 (===) を使用する場合は、値が比較される前にデータ型の変換は試みられません。その結果、変数は 2 つの異なる値として見なされます。

次の例では、" 以上 " (>=) 演算子を使用して値を比較し、ユーザーがテキストフィールドに入力する値に基づいてコードを実行します。

コードで " 以上 " 演算子を使用するには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して新しい FLA ファイルを作成します。
2. メインタイムラインのフレーム 1 に次のコードを追加します。

```
this.createTextField("myTxt", 20, 0, 0, 100, 20);
myTxt.type = "input";
myTxt.border = true;
myTxt.restrict = "0-9";

this.createEmptyMovieClip("submit_mc", 30);
submit_mc.beginFill(0xFF0000);
submit_mc.moveTo(0, 0);
submit_mc.lineTo(100, 0);
submit_mc.lineTo(100, 20);
submit_mc.lineTo(0, 20);
submit_mc.lineTo(0, 0);
submit_mc.endFill();
submit_mc._x = 110;

submit_mc.onRelease = function(evt_obj:Object):Void {
   var myNum:Number = Number(myTxt.text);
   if (isNaN(myNum)) {
      trace("Please enter a number");
      return;
```

```
    }
    if (myNum >= 10) {
      trace("Your number is greater than or equal to 10");
    } else {
      trace("Your number is less than 10");
    }
};
```

3. [制御]-[ムービープレビュー] を選択して ActionScript をテストします。

   特定の条件が true かどうかをチェックし、true でない場合は別のブロックを実行することもできます。

4. ActionScript で条件を次のように変更します。

```
if (myNum == 10) {
  trace("Your number is 10");
} else {
     trace("Your number is not 10");
}
```

5. [制御]-[ムービープレビュー] を選択して再度 ActionScript をテストします。

厳密な等価演算子 (===) を除き、比較演算子でストリングが比較されるのは、オペランドが両方ともストリングである場合だけです。オペランドのどちらか一方だけがストリングである場合、両方のオペランドが数値に変換され、数値として比較されます。ストリングおよび演算子の詳細については、140 ページの「演算子とストリングの使用について」を参照してください。順序および演算子の優先順位による ActionScript への影響の詳細については、138 ページの「演算子の優先順位と結合性について」を参照してください。

代入演算子について

代入演算子は 2 つのオペランドを取り、他のオペランドの値に基づいて 1 つのオペランドに値を代入します。この表の演算子の優先順位はすべて同じです。

演算子	実行される演算
=	代入
*=	乗算後代入
/=	除算後代入
%=	剰余を代入
+=	加算後代入
-=	減算後代入
<<=	ビット単位での左シフト後代入
>>=	ビット単位での右シフト後代入
>>>=	ビット単位での符号なし右シフト後代入

演算子	実行される演算
&=	ビット単位の論理積 (AND) を代入
^=	ビット単位の排他的論理和 (XOR) を代入
|=	ビット単位の排他的論理和 (OR) を代入

代入演算子の使用の詳細については、151 ページの「代入演算子の使用」を参照してください。

代入演算子の使用

代入 (=) 演算子を使用すると、指定した値を変数に代入できます。次のようにして、変数にストリングを代入できます。

```
var myText:String = "ScratchyCat";
```

また、同じ式で複数の変数の値を代入する場合も代入演算子を使用することができます。次のステートメントでは、値 10 を変数 numOne、numTwo、および numThree に代入します。

```
var numOne:Number;
var numTwo:Number;
var numThree:Number;
numOne = numTwo = numThree = 10;
```

複合代入演算子を使用して、演算を組み合わせて実行することも可能です。これらの演算子により、両方のオペランドについて演算が実行され、結果として求められた値が最初のオペランドに代入されます。たとえば、次のステートメントはいずれも同じ演算を実行します。

```
var myNum:Number = 0;
myNum += 15;
myNum = myNum + 15;
```

論理演算子について

論理演算子を使用してブール値 (true および false) を比較し、比較に基づいてブール値を返します。たとえば、オペランドが両方とも true と評価される 2 つのオペランドがある場合、論理積 (AND) 演算子 (&&) は true を返します。または、オペランドのどちらか一方、あるいは両方が true と評価される場合、論理和 (OR) 演算子 (||) は true を返します。

論理演算子は 2 つのオペランドを取り、ブール値の結果を返します。論理演算子は優先順位が異なりますが、優先順位の高いものから順に次の表に示します。

演算子	実行される演算
&&	論理積 (AND)
||	論理和 (OR)

論理演算子の使用の詳細については、152 ページの「論理演算子の使用」を参照してください。

論理演算子の使用

論理演算子は if ステートメントの条件を評価するために、比較演算子と共によく使用します。これについては、次の例に示します。

コードで論理演算子を使用するには :

1. [ファイル]-[新規] を選択し、新規 Flash ドキュメントを作成します。
2. Actions パネルを開き、タイムラインのフレーム 1 に 次の ActionScript を入力します。

```
this.createTextField("myTxt", 20, 0, 0, 100, 20);
myTxt.type = "input";
myTxt.border = true;
myTxt.restrict = "0-9";

this.createEmptyMovieClip("submit_mc", 30);
submit_mc.beginFill(0xFF0000);
submit_mc.moveTo(0, 0);
submit_mc.lineTo(100, 0);
submit_mc.lineTo(100, 20);
submit_mc.lineTo(0, 20);
submit_mc.lineTo(0, 0);
submit_mc.endFill();
submit_mc._x = 110;

submit_mc.onRelease = function():Void {
  var myNum:Number = Number(myTxt.text);
  if (isNaN(myNum)) {
    trace("Please enter a number");
    return;
  }
  if ((myNum > 10) && (myNum < 20)) {
    trace("Your number is between 10 and 20");
  } else {
    trace("Your number is NOT between 10 and 20");
  }
};
```

この ActionScript では、実行時にテキストフィールドを作成します。テキストフィールドに数値を入力し、ステージ上のボタンをクリックすると、論理演算子を使用して [出力] パネルにメッセージが表示されます。メッセージは、テキストフィールドに入力する数値によって異なります。

オペランドを使用するときは、順序に注意する必要があります。複雑な条件を使用するときは、これが特に重要となります。次のコードでは、論理積 (AND) 演算子を使用して、10 ～ 20 の番号をチェックします。その結果に基づいて、適切なメッセージを表示します。数値が 10 未満または 20 より大きい場合は、別のメッセージが [出力] パネルに表示されます。

```
submit_mc.onRelease = function():Void {
   var myNum:Number = Number(myTxt.text);
   if (isNaN(myNum)) {
      trace("Please enter a number");
      return;
   }
   if ((myNum > 10) && (myNum < 20)) {
      trace("Your number is between 10 and 20");
   } else {
      trace("Your number is NOT between 10 and 20");
   }
};
```

ビット単位シフト演算子について

ビット単位シフト演算子は 2 つのオペランドを取り、2 番目のオペランドで指定された範囲で最初のオペランドのビットをシフトします。この表の演算子の優先順位はすべて同じです。

演算子	実行される演算
<<	ビット単位の左シフト
>>	ビット単位の右シフト
>>>	(ビット単位の符号なし右シフト)

ビット演算子の使用の詳細については、154 ページの「ビット演算子の使用について」を参照してください。各ビット演算子の詳細については、『ActionScript 2.0 リファレンスガイド』の該当項目を参照してください。

ビット論理演算子について

ビット論理演算子は 2 つのオペランドを取り、ビットレベルの論理演算を実行します。ビット論理演算子は優先順位が異なりますが、優先順位の高いものから順に次の表に示します。

演算子	実行される演算
&	ビット単位の論理積 (AND)
^	ビット単位の排他的論理和 (XOR)
|	ビット単位の論理和 (OR)

ビット演算子の使用の詳細については、154 ページの「ビット演算子の使用について」を参照してください。各ビット演算子の詳細については、『ActionScript 2.0 リファレンスガイド』の該当項目を参照してください。

ビット演算子の使用について

ビット演算子は、浮動小数点値を内部処理して、32 ビット整数に変換します。実際に実行されるビット演算は演算子によって異なりますが、どのビット演算でも 32 ビット整数が 2 進数 (ビット) 単位で評価され、新しい値が計算されます。ビット単位シフト演算子の一覧については、153 ページの「ビット単位シフト演算子について」を参照してください。ビット論理演算子の一覧については、153 ページの「ビット論理演算子について」を参照してください。

Flash でのビット演算子の使用はあまり一般的ではありませんが、状況によっては役立つことがあります。たとえば、Flash プロジェクトの許可のマトリックスを作成する場合で、許可の種類ごとに別の変数を作成したくないときがあります。このような場合は、ビット演算子を使用できます。

次の例では、`Array.sort()` メソッドと共にビット単位の論理和 (OR) 演算子を使用する方法を示します。

ビット単位の論理和 (OR) 演算子を使用するには :

1. [ファイル]-[新規] を選択し、新規 Flash ドキュメントを作成します。
2. 次の ActionScript を [アクション] パネルに入力します。

```
var myArr:Array = new Array("Bob", "Dan", "doug", "bill", "Hank", "tom");
trace(myArr); // Bob,Dan,doug,bill,Hank,tom
myArr.sort(Array.CASEINSENSITIVE | Array.DESCENDING);
trace(myArr); // tom,Hank,doug,Dan,Bob,bill
```

   最初の行では、ランダムな名前の配列を定義し、それらの名前をトレースして [出力] パネルに表示します。次に、`Array.sort()` メソッドを呼び出し、定数値 `Array.CASEINSENSITIVE` および `Array.DESCENDING` を使って 2 つのソートオプションを指定します。ソートメソッドにより、配列のアイテムは逆の順番 (z から a) でソートされます。この検索では大文字と小文字が区別されません。`a` および `A` は同じように扱われますが、これは `Z` が `a` よりも先になる、大文字と小文字を区別する検索とは異なります。

3. [制御]-[ムービープレビュー] を選択して ActionScript をテストします。次のテキストが [出力] パネルに表示されます。

```
Bob,Dan,doug,bill,Hank,tom
tom,Hank,doug,Dan,Bob,bill
```

ソートメソッドでは、5 つのオプションを利用できます。

- 1 または Array.CASEINSENSITIVE (バイナリ = 1)
- 2 または Array.DESCENDING (バイナリ = 10)
- 4 または Array.UNIQUESORT (バイナリ = 100)
- 8 または Array.RETURNINDEXEDARRAY (バイナリ = 1000)
- 16 または Array.NUMERIC (バイナリ = 10000)

配列のソートオプションを定義するには、3 つの異なる方法があります。

```
my_array.sort(Array.CASEINSENSITIVE | Array.DESCENDING); // 定数
my_array.sort(1 | 2); // 数値
my_array.sort(3); // 数値を追加
```

ソートオプションの数値は、明白ではないものの実際にはビット単位の桁 (バイナリまたは基数 2) です。定数値 `Array.CASEINSENSITIVE` は数値 1 に等しく、これはバイナリ値 1 でもあります。定数値 `Array.DECENDING` は数値 2 またはバイナリ値 10 を持ちます。

バイナリ値の操作では、混乱が発生しがちです。バイナリには、1 または 0 の 2 つの値のみが可能であり、このために値 2 が 10 として表されます。数値 3 をバイナリで表示する場合は、11 (1+10) になります。バイナリでは、数値 4 は 100、数値 5 は 101、のようになります。

次の ActionScript では、ビット単位の論理積 (AND) 演算子を使用して `Array.DESCENDING` および `Array.NUMERIC` 定数を加算し、数値の配列を降順でソートする方法を示します。

```
var scores:Array = new Array(100,40,20,202,1,198);
trace(scores); // 100,40,20,202,1,198
trace(scores.sort()); // 1,100,198,20,202,40
var flags:Number = Array.NUMERIC|Array.DESCENDING;
trace(flags); // 18 ( 基数 10)
trace(flags.toString(2)); // 10010 ( バイナリ -- 基数 2)
trace(scores.sort(flags)); // 202,198,100,40,20,1
```

条件演算子について

条件演算子は三項演算子であり、3 つのオペランドを取ります。条件演算子は、`if...else` 条件ステートメントを適用する簡易的な方法です。

演算子	実行される演算
`?:`	三項演算

条件演算子と例の使用の詳細については、112 ページの「条件演算子と代替シンタックスについて」を参照してください。

ドキュメントでの演算子の使用

次の例では、Math.round() メソッドを使って、任意の小数点以下の桁数に計算を四捨五入することができます。このメソッドは score パラメータを整数に切り上げまたは切り捨て、その値を返します。ActionScript を少し修正すると、この代わりに特定の小数点以下の桁数に数値を四捨五入できます。

この例では、除算演算子および乗算演算子を使用して、指定した質問の合計数で正解数を割り、それに基づいてユーザーの得点を計算します。ユーザーの得点は、数値で乗算し、0% ～ 100% の得点を表示することができます。次に、加算演算子を使用して、ユーザーの得点をストリングと連結し、[出力] パネルに表示できます。

ActionScript で演算子を使用するには :

1. 新しい Flash ドキュメントを作成します。
2. メインタイムラインのフレーム 1 に次の ActionScript を入力します。

```
var correctAnswers:Number = 11;
var totalQuestions:Number = 13;
// 整数に四捨五入する
//var score:Number = Math.round(correctAnswers / totalQuestions * 100);
// 小数点以下 2 桁に四捨五入する
var score:Number = Math.round(correctAnswers / totalQuestions * 100 * 100) /
  100;
trace("You got " + correctAnswers + " out of " + totalQuestions + " answers
  correct, for a score of " + score + "%.");
```

3. [制御]-[ムービープレビュー] を選択します。

   [出力] パネルに次のテキストが表示されます。

   ```
   You got 11 out of 13 answers correct, for a score of 84.62%.
   ```

   この例で Math.round() を呼び出すと、得点は整数 (85) に四捨五入され、[出力] パネルに表示されます。Math.round() を呼び出す前に数値にさらに 100 を掛けて、100 で割ると、小数点以下 2 桁に数値を四捨五入することができます。この結果、より正確な得点が得られます。
4. correctAnswers 変数を 3 に変更し、[制御]-[ムービープレビュー] を選択して、SWF ファイルを再びテストしてみてください。

テストアプリケーションを作成する場合は、RadioButton および Label コンポーネントを使って、" はい / いいえ " 形式または複数選択形式の一連の質問をよく作成します。ユーザーが質問に答え、[送信] ボタンをクリックしたら、その答えを回答キーと比較して得点を計算します。

第 5 章

関数とメソッド

5

ActionScript を記述する際、クラスを作成する際、およびメソッドを使用する際は、関数について理解していることが重要です。使用する関数には、いくつかの異なる種類があります。この章では、ビルトインクラスを使用するときにアプリケーションで関数やメソッドを使用する方法、および関数やメソッドを記述する方法について学習します。第 6 章の「クラス」では、通常の方法で関数を記述してカスタムクラスを作成します。また、ActionScript クラスファイル内に関数を記述する方法についても学習します。

それらの関数をコードで使用して、インタラクティブ機能、アニメーション、その他のエフェクトをアプリケーションに追加できます。この章では、Flash アプリケーション内に記述できるさまざまな関数について説明します。関数とメソッドの概要、および Flash で関数とメソッドを記述して使用する実習については、次のトピックを参照してください。

関数とメソッドについて 157
メソッドについて 176

関数とメソッドについて

メソッドおよび関数とは、SWF ファイルのあらゆる場所で再利用できる ActionScript コードのブロックです。関数は FLA ファイルまたは外部の ActionScript ファイルに記述して、ドキュメント内の任意の場所から呼び出すことができます。メソッドは、ActionScript クラス定義内に属する関数です。渡される値に対して一連のステートメントを実行するような関数を定義できます。また、関数から値を返すこともできます。関数が定義された後は、あらゆるタイムライン (ロードされた SWF ファイルのタイムラインも含む) から、その関数を呼び出すことができます。

値が関数のパラメータとして渡されると、関数内では、その値を使用して計算を実行できます。個々の関数にはそれぞれ固有の特性があり、特定の型や個数の値を受け取る必要がある関数もあります。関数に必要とされる以上に多くのパラメータを渡した場合、余分な値は無視されます。必要なパラメータを渡さなかった場合、空のパラメータに `undefined` データ型が割り当てられ、場合によっては実行時にエラーが発生します。また、関数から値を返すこともできます (175 ページの「関数からの値の戻り」を参照)。

メモ

関数を呼び出すには、その関数の定義があるフレームに再生ヘッドが既に到達していることが必要です。

適切に記述されている関数は、" ブラックボックス " として扱うことができます。関数の入出力などの処理内容について、適切なコメントが記述してあれば、関数の使用者は、関数が内部でどのような処理を実行しているかを詳しく理解する必要はなくなります。

単純な名前付き関数の基本シンタックスは次のとおりです。

```
function traceMe() {
  trace("your message");
}
traceMe();
```

名前付き関数の記述方法については、162 ページの「名前付き関数の記述」を参照してください。

上記の例の単純な名前付き関数で、パラメータ yourMessage を渡す基本シンタックスは次のとおりです。

```
function traceMe(yourMessage:String) {
  trace(yourMessage);
}
traceMe("How you doing?");
```

または、複数のパラメータを渡すには、次のコードを使用できます。

```
var yourName:String = "Ester";
var yourAge:String = "65";
var favSoftware:String = "Flash";
function traceMe(favSoftware:String, yourName:String, yourAge:String) {
  trace("I'm " + yourName + ", I like " + favSoftware + ", and I'm " + yourAge +
  ".");
}
traceMe(favSoftware,yourName,yourAge);
```

パラメータの受け渡しの詳細については、173 ページの「関数へのパラメータの受け渡し」を参照してください。

記述できる関数にはさまざまな種類があります。関数を記述する方法の詳細と、特定の種類の関数を記述する方法を説明したセクションへのリンクについては、159 ページの「メソッドと関数の種類について」を参照してください。メソッドと関数の比較例については、176 ページの「メソッドについて」を参照してください。

メモ

スクリプトアシストを使用してコードを記述する方法については、『Flash ユーザーガイド』を参照してください。

関数とメソッドの詳細については、次のトピックを参照してください。

■ 159 ページの「メソッドと関数の種類について」

メソッドと関数の種類について

クラスに属する関数は、そのクラスのメソッドと呼ばれます。アプリケーションで使用できる関数には、ビルトイン関数、ユーザー定義の名前付き関数、匿名関数、コールバック関数、コンストラクタ関数、関数リテラルなどのさまざまな種類があります。以降のセクションでは、これらの関数を定義する方法について説明します。

ActionScript クラスファイル内に関数を記述することもできます。これらの関数は、スクリプトでメソッドとして使用します。次の例の Person クラスには、コンストラクタメソッド、クラスメソッド、インスタンスメソッド、およびアクセサメソッド (getter と setter) があります。このコードサンプル内のコメントは、コード内でこれらのメソッドが出現する場所を示しています。

メモ

次のようなクラスファイルを記述する方法については、179 ページ、第 6 章の「クラス」を参照してください。

```
class Person {
  public static var numPeople:Number = 0;

  // インスタンスメンバー
  private var _speed:Number;

  // コンストラクタ
  public function Person(speed:Number) {
    Person.numPeople++;
    this._speed = speed;
  }

  // 静的メソッド
  public static function getPeople():Number {
    return Person.numPeople;
  }

  // インスタンスメソッド
  public function walk(speed:Number):Void {
    this._speed = speed;
  }
  public function run():Void {
    this._speed *= 2;
  }
  public function rest():Void {
    this._speed = 0;
  }

  // getter/setter ( アクセサメソッド )
  public function get speed():Number {
    return this._speed;
  }
}
```

上記のコードサンプルのようなメソッドを記述する詳細手順については、179 ページ、第 6 章の「クラス」を参照してください。コードで使用するメソッドが、ActionScript 言語のビルトインクラスに属している場合もあります。たとえば、MovieClip および Math は、アプリケーションで使用できる最上位クラスです。このようなクラスのメソッドをコードで使用する場合、これらは (上記のコードサンプルと同様の) ビルトインクラスに記述された関数です。その他、自分で記述したカスタムクラスのメソッドを使用することもできます。

クラスに属さない関数は、" トップレベル関数 " (" 定義済み関数 "、" ビルトイン関数 ") と呼ばれ、コンストラクタなしで呼び出すことができます。ActionScript 言語の最上位に位置するビルトイン関数としては、`trace()` と `setInterval()` などがあります。

トップレベル関数の呼び出しをコードに追加するには、[アクション] パネルのスクリプトペインで、単一のコード行を追加します。たとえば、次のように入力します。

```
trace("my message");
```

この単一のコード行で SWF ファイルをテストすると、トップレベル関数 `trace()` が呼び出され、[出力] パネルにテキストが表示されます。

注意 : メソッドをプロパティに割り当てる場合は、関数への参照を渡すので、次のようにメソッド名の後の括弧を省略します。

```
my_mc.myMethod = aFunction;
```

ただし、コード内でメソッドを呼び出す際は、メソッド名の後のカッコが必要になります。

```
my_mc.myMethod();
```

メモ	トップレベル関数の詳細については、161 ページの「ビルトイン関数とトップレベル関数について」を参照してください。

関数の定義は、これ以外にもさまざまな方法で行うことができます。各種類の関数の詳細については、次のセクションを参照してください。

- 161 ページの「ビルトイン関数とトップレベル関数について」
- 162 ページの「名前付き関数の記述」
- 163 ページの「匿名関数とコールバック関数の記述」
- 165 ページの「関数リテラルについて」
- 167 ページの「ユーザー定義関数の指定と呼び出し」
- 166 ページの「コンストラクタ関数について」

関数とメソッドの記述方法と使用方法については、次の関連セクションを参照してください。関数の使用方法については、169 ページの「Flash での関数の使用」を参照してください。メソッドの使用方法については、176 ページの「メソッドについて」を参照してください。

メモ	スクリプトアシストを使用してコードを記述する方法については、『Flash ユーザーガイド』を参照してください。

ビルトイン関数とトップレベル関数について

157 ページの「関数とメソッドについて」に述べたとおり、関数とは、SWF ファイルのあらゆる場所で再利用できる ActionScript コードのブロックです。関数はパラメータとして受け渡された値を処理します。また、関数から値を返すこともできます。

ActionScript 言語に組み込まれている関数を使用できます。それらの関数には、159 ページの「メソッドと関数の種類について」で説明したようなトップレベル関数と、Math や MovieClip などのビルトインクラスに属する関数があります。後者は、アプリケーションでメソッドとして使用します。

ActionScript のビルトイン関数は、特定のタスクの実行や情報へのアクセスに使用します。たとえば、`getTimer()` を使用すると、SWF ファイルを再生しているミリ秒数を取得できます。また、`getVersion()` を使用すると、ファイルのホストである Flash Player のバージョン番号を取得できます。オブジェクトに割り当てられた関数は、" メソッド " と呼ばれます。オブジェクトに割り当てられていない関数は " トップレベル関数 " と呼ばれ、[アクション] パネルのグローバル関数カテゴリのサブカテゴリに表示されます。

特定の値を受け取る必要があるビルトイン関数もあります。関数が必要とするパラメータよりも多くのパラメータを渡した場合、余分な値は無視されます。必要なパラメータを渡さなかった場合、空のパラメータに `undefined` データ型が割り当てられ、場合によっては実行時にエラーが発生します。

メモ	関数を呼び出すには、その関数の定義があるフレームに再生ヘッドが既に到達していることが必要です。

トップレベル関数の使用方法は簡単です。関数を呼び出すには、単にその名前を指定すればよく、その関数に必要なパラメータがある場合はあわせて指定します。必要なパラメータについては、『ActionScript 2.0 リファレンスガイド』の関数の項目を参照してください。たとえば、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
trace("my message");
```

ファイルをテストすると、[出力] パネルに `my message` が表示されます。トップレベル関数としては、`setInterval()` と `getTimer()` もあります。次の例に、これら 2 つの関数を共に使用する方法を示します。タイムラインのフレーム 1 に次のコードを追加します。

```
function myTimer():Void {
  trace(getTimer());
}
var intervalID:Number = setInterval(myTimer, 100);
```

このコードでは、`getTimer()` を使用して単純なタイマーを作成し、トップレベル関数 `setInterval()` および `trace()` を使用して、Flash Player で SWF ファイルの再生が開始されてから経過したミリ秒数を表示します。

ビルトイン関数の呼び出しは、" ユーザー定義関数 " の呼び出しと似ています。詳細については、167 ページの「ユーザー定義関数の指定と呼び出し」を参照してください。各関数の詳細については、『ActionScript 2.0 リファレンスガイド』の該当項目を参照してください。

名前付き関数の記述

名前付き関数は、あらゆる種類のアクションを実行するために ActionScript コードでよく作成する関数です。SWF ファイルの作成時には、名前付き関数が最初にコンパイルされます。つまり、関数が現在のフレームまたは前のフレームで定義済みであれば、コードのどこからでも関数を参照できます。たとえば、関数がタイムラインのフレーム 2 で定義されている場合、タイムラインのフレーム 1 でその関数にアクセスすることはできません。

名前付き関数の標準形式は次のとおりです。

```
function functionName(parameters) {
  // 関数ブロック
}
```

このコード片は、次の部分から構成されています。

- `functionName` : 関数の一意な名前です。ドキュメント内のすべての関数名は一意である必要があります。
- `parameters` : 関数に渡されるパラメータを (必要に応じて複数) 配置します。パラメータは "引数" とも呼ばれます。パラメータの詳細については、173 ページの「関数へのパラメータの受け渡し」を参照してください。
- `// 関数ブロック` : 関数で実行するすべての ActionScript コードを配置します。これは " 目的の処理を実行する " ステートメントを配置するための部分であり、必要なコードをここに記述できます。"`// 関数ブロック`" というコメントは、関数ブロック用のコードを配置する場所を示すプレースホルダです。

名前付き関数を使用するには :

1. 新しいドキュメントを namedFunc.fla という名前で作成します。
2. [ファイル]-[読み込み]-[ライブラリに読み込み] を選択し、短いサウンドファイルを選択してライブラリに読み込みます。
3. サウンドファイルを右クリックし、[リンケージ] を選択します。
4. [識別子] テキストボックスに「mySoundID」と入力します。
5. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

   ```
   function myMessage() {
     trace("mySoundID completed");
   }
   var my_sound:Sound = new Sound();
   my_sound.attachSound("mySoundID");
   my_sound.onSoundComplete = myMessage;
   my_sound.start();
   ```

   このコードで作成した `myMessage` という名前付き関数は、`trace()` 関数を呼び出すために後でスクリプトで使用します。

6. [制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

ActionScript 内に独自の関数を作成するには、`function` ステートメントを使用します。パラメータは省略できますが、パラメータを指定しない場合でも、角カッコは指定する必要があります。中カッコ ({ }) で囲まれた部分は、" 関数ブロック " と呼ばれます。

関数は、メインタイムライン上またはクラスファイルを含む外部の ActionScript ファイル内に記述できます。

クラスファイル内にコンストラクタ関数を記述する場合も、この形式を使用します (ただし、関数名とクラス名を同じ名前にします)。コンストラクタ関数の詳細については、220 ページの「コンストラクタ関数の記述」を参照してください。クラス内での関数の記述方法と記述例については、179 ページ、第 6 章の「クラス」も参照してください。

匿名関数とコールバック関数の記述

名前付き関数は、その定義前または定義後にスクリプト内で参照できる関数です。一方、" 匿名関数 " は名前のない関数で、作成時にそれ自体を参照します。ActionScript コードを記述するときは、多数の匿名関数を作成することになります。

匿名関数は、イベントハンドラを作成する場合によく使用されます。匿名関数を記述するには、変数内に関数リテラルを格納します。このことにより、それ以降のコードで関数を参照できます。次の例に、匿名関数の記述方法を示します。

匿名関数を記述するには :

1. ステージ上にムービークリップを作成し、そのクリップを選択します。
2. プロパティインスペクタを開き、[インスタンス名] テキストボックスに「my_mc」と入力します。
3. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
var myWidth = function () {
  trace(my_mc._width);
};
// コードの後の部分で追加可能
myWidth();
```

4. [制御]-[ムービープレビュー] を選択します。

   ムービークリップの幅が [出力] パネルに表示されます。

関数は、XML インスタンスや LoadVars インスタンスなどのオブジェクト内に作成することもできます。匿名関数を特定のイベントに関連付けると、" コールバック関数 " を作成できます。コールバック関数は、ロードの完了 (`onLoad()`) 後またはアニメーションの完了 (`onMotionFinished()`) 後など、特定のイベントの発生後に関数から呼び出されます。

たとえば、サーバーから SWF ファイルにロードされるデータを処理する ActionScript を記述しなければならないことがあります。SWF ファイルへのデータのロードが終了すると、その場所からデータにアクセスできます。ActionScript を使用してデータが完全にロードされたかどうかをチェックすることが重要です。コールバック関数を使って、データがドキュメントにロードされたことを通知することもできます。

リモート XML ドキュメントをロードする次のコールバック関数では、匿名関数を onLoad() イベントに関連付けます。次の例に示すように、XML.load() とコールバック関数を使用します。タイムラインのフレーム 1 に次のコードを入力します。

```
var my_xml:XML = new XML();
my_xml.onLoad = function(success:Boolean):Void {
  trace(success);
};
my_xml.load("http://www.helpexamples.com/crossdomain.xml");
```

前のコードから、onLoad() イベントハンドラは匿名関数を使用して onLoad() イベントを処理していることがわかります。

コールバック関数の詳細については、279 ページ、第 9 章の「イベントの処理」を参照してください。

次のコードのように、匿名関数を setInterval() 関数と共に使用することもできます。このコードでは、setInterval() を使用して約 1000 ミリ秒 (1 秒) ごとに匿名関数を呼び出します。

```
setInterval(function() {trace("interval");}, 1000);
```

匿名関数の代わりに、名前付き関数を使用することもできます。コールバック関数などの一部の状況を除き、多くの場合、名前付き関数は匿名関数に比べて解読や理解が容易です。また、名前付き関数は前方参照することもできます。つまり、関数がタイムライン上に現れる前に参照できます。

名前付き関数とは異なり、匿名関数は、変数に割り当てない限り、コードのどこからでも参照できるわけではありません。たとえば、次のように、FLA ファイルのフレーム 5 に匿名関数があるとします。

```
// タイムラインにわたる my_mc というムービークリップの場合
stop();
var myWidth = function () {
  trace(my_mc._width);
};
```

次のコードをフレーム 1 に配置した場合は、関数を参照できません。

```
myWidth();
```

同様に、次のコードをどのフレームに配置しても、正常に動作しません。

```
myWidth();
var myWidth:Function = function () {
  trace(my_mc._width);
};
```

ただし、次のコードは正常に動作します。

```
var myWidth:Function = function () {
  trace(my_mc._width);
};
myWidth();
```

メモ	また、myWidth 関数があるフレームより後であれば、任意のフレームに myWidth() を配置できます。

名前付き関数を定義する場合、フレームスクリプトでの名前付き関数の呼び出しは正常に動作しますが、匿名関数を使用した同様のコードは正常に動作しません。

```
// 次のコードは、名前付き関数を呼び出しているので、正常に動作する
myWidth();
function myWidth() {
  trace("foo");
}

// 次のコードは、匿名関数を呼び出しているので、正常に動作しない
myWidth();
var myWidth:Function = function () {
  trace("foo");
};
```

詳細については、162 ページの「名前付き関数の記述」を参照してください。

メモ	スクリプトアシストを使用してコードを記述する方法については、『Flash ユーザーガイド』を参照してください。

関数リテラルについて

" 関数リテラル " とは、ステートメント内ではなく式内に宣言する、名前のない関数です。一時的に関数を使用する場合や、コード内で関数の代わりに式を使用する可能性がある場所で関数を使用する場合に便利です。関数リテラルのシンタックスは次のとおりです。

```
function (param1, param2, etc) {
  // ステートメント
};
```

たとえば、次のコードでは、式として関数リテラルが使用されています。

```
var yourName:String = "Ester";
setInterval(function() {trace(yourName);}, 200);
```

メモ	関数リテラルを再定義すると、古い定義が新しい定義で置き換えられます。

関数リテラルを変数に格納することにより、その関数に後のコードでアクセスできます。これを行うには、" 匿名関数 " を使用します。詳細については、163 ページの「匿名関数とコールバック関数の記述」を参照してください。

コンストラクタ関数について

クラスのコンストラクタは、new キーワード (var my_xml:XML = new XML(); など) を使用してクラスのインスタンスを作成するときに自動的に呼び出される特殊な関数です。コンストラクタ関数は、それを含んでいるクラスと同じ名前になります。たとえば、カスタムの Person クラスを作成する場合は、次のコンストラクタ関数が含まれることになります。

```
public function Person(speed:Number) {
  Person.numPeople++;
  this._speed = speed;
}
```

その後、次のように新しいインスタンスを作成できます。

```
var myPerson:Person = new Person();
```

メモ	クラスファイル内にコンストラクタ関数を明示的に宣言していない場合、つまり、クラスと同じ名前の関数を作成していない場合は、コンパイラによって空のコンストラクタ関数が自動的に作成されます。

1 つのクラスに含まれるコンストラクタ関数は 1 つのみです。ActionScript 2.0 ではコンストラクタ関数のオーバーロードは認められていません。また、コンストラクタ関数では戻り値の型を指定できません。クラスファイル内にコンストラクタ関数を記述する方法の詳細については、220 ページの「コンストラクタ関数の記述」を参照してください。

グローバル関数とタイムライン関数の定義

157 ページの「関数とメソッドについて」では、Flash で使用できる各種の関数について学習しました。変数と同様に、関数も、その関数が定義されるムービークリップのタイムラインに割り当てられます。関数を呼び出すには、必ずターゲットパスを使用します。また、変数と同様に、_global 識別子を使用して、ターゲットパスを指定せずにすべてのタイムラインおよびスコープで使用できるグローバル関数を定義できます。グローバル関数を定義するには、次の例のように関数名の前に識別子 _global を付け加えます。

```
_global.myFunction = function(myNum:Number):Number {
   return (myNum * 2) + 3;
};
trace(myFunction(5)) // 13
```

_global とスコープの詳細については、57 ページの「変数とスコープについて」を参照してください。

タイムライン関数を定義するには、function ステートメントに続けて、関数名、関数に渡されるパラメータ、関数の動作を表す ActionScript ステートメントを入力します。

次の例は、パラメータ radius が指定された areaOfCircle 関数の例です。

```
function areaOfCircle(radius:Number):Number {
  return (Math.PI * radius * radius);
}
trace (areaOfCircle(8));
```

関数の定義は、これ以外にもさまざまな方法で行うことができます。各種類の関数の詳細については、次のセクションを参照してください。

- 161 ページの「ビルトイン関数とトップレベル関数について」
- 162 ページの「名前付き関数の記述」
- 163 ページの「匿名関数とコールバック関数の記述」
- 165 ページの「関数リテラルについて」
- 166 ページの「コンストラクタ関数について」
- 167 ページの「ユーザー定義関数の指定と呼び出し」

関数の命名法については、168 ページの「関数の命名」を参照してください。外部クラスファイル内での関数の使用例の詳細については、169 ページの「Flash での関数の使用」および179 ページ、第 6 章の「クラス」を参照してください。

メモ	スクリプトアシストを使用してコードを記述する方法については、『Flash ユーザーガイド』を参照してください。

ユーザー定義関数の指定と呼び出し

"ユーザー定義関数" とは、アプリケーションで使用するために独自に作成する関数で、事前に定義された機能を実行するビルトインクラス内の関数とは異なります。ユーザー定義関数を作成するには、独自に名前を付け、関数ブロック内にステートメントを追加します。名前付き関数、匿名関数、コールバック関数などの関数の記述方法は、前のセクションで説明されています。関数の命名法については、168 ページの「関数の命名」を参照してください。関数の使用方法については、169 ページの「Flash での関数の使用」を参照してください。

ターゲットパスを使用すると、ロードされた SWF ファイルのタイムラインなど、あらゆるタイムラインの関数を、どのタイムラインからも呼び出すことができます。関数を呼び出すには、必要であれば関数名へのターゲットパスを指定した上で、必要なパラメータをカッコ内で渡します。ユーザー定義関数のシンタックスには、いくつかの形式があります。次のコードでは、パスを使用して `initialize()` 関数を呼び出します。この関数は現在のタイムライン上で定義されたもので、パラメータは必要ありません。

```
this.initialize();
```

次の例では、相対パスを使用して `list()` 関数を呼び出します。この関数は、ムービークリップ `functionsClip` 内で定義されたものです。

```
this._parent.functionsClip.list(6);
```

名前付き関数の記述方法については、162 ページの「名前付き関数の記述」を参照してください。パラメータの詳細については、173 ページの「関数へのパラメータの受け渡し」を参照してください。

また、独自の名前付き関数を定義することもできます。たとえば、次の名前付き関数 helloWorld() はユーザー定義関数です。

```
function helloWorld() {
  trace("Hello world!");
};
```

次の例では、FLA ファイル内のユーザー定義関数の使用方法を示します。

単純なユーザー定義関数を作成して呼び出すには :

1. 新しい Flash ドキュメントを作成し、udf.fla という名前で保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

   ```
   function traceHello(name:String):Void {
     trace("hello, " + name + "!");
   }
   traceHello("world"); // hello, world!
   ```

   上記のコードでは、traceHello() というユーザー定義関数が作成されます。この関数は、1 つの引数 name を取り、グリーティングメッセージをトレース出力します。このユーザー定義関数を呼び出すには、1 つのストリング値を指定し、関数定義と同じタイムラインから traceHello を呼び出します。
3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

名前付き関数の詳細については、162 ページの「名前付き関数の記述」を参照してください。クラスには多数のユーザー定義関数が含まれています。クラスファイル内での関数の記述方法については、169 ページの「Flash での関数の使用」を参照してください。また、第 6 章の「クラス」のセクション197 ページの「クラスファイルのメソッドとプロパティの使用」、199 ページの「パブリック、プライベート、および静的なメソッドとプロパティ (メンバー) について」、および 203 ページの「クラスメンバーについて」も参照してください。

関数の命名

関数名は、先頭を小文字で開始し、戻り値がある場合はその値を説明する名前にすることをお勧めします。たとえば、曲名を返す関数には getCurrentSong() のような名前を付けます。

ActionScript はオーバーロードをサポートしていないため、複数の類似した (機能的に関連性のある) 関数を作成する場合には、それらをグループ化できるように関数名の命名規則を定める必要があります。" オーバーロード " とは OOP (Object-Oriented Programming : オブジェクト指向プログラミング) の概念であり、渡される値のデータ型に応じて関数に異なる処理を実行させる機能のことです。

変数と同様に、特殊文字を使用することや、メソッド名を数字で始めることはできません。詳細については、646 ページの「命名規則」を参照してください。メソッドの命名法については、178 ページの「メソッドの命名」を参照してください。

Flash での関数の使用

このセクションでは、アプリケーションでの関数の使用方法を示します。次のコード例の中には、FLA ファイル内にある ActionScript を使用しているものや、比較のために関数をクラスファイル内に配置しているものがあります。クラスファイル内での関数の使用方法の詳細と使用例については、179 ページ、第 6 章の「クラス」を参照してください。クラスファイル用の関数の記述方法の詳細については、214 ページの「例 : カスタムクラスの記述」を参照してください。

コード作成の作業量や SWF ファイルのサイズを削減するために、コードはできるだけ再利用してください。コードを再利用する基本は、毎回異なるコードを作成するのではなく、同じ関数を何度も呼び出すようにすることです。汎用的なコードを関数にすれば、SWF ファイル内で、同じブロックのコードを複数の多少異なる目的に使用できます。コードを再利用することで、効率のよいアプリケーションを作成できると共に、ActionScript コードの記述量を減らすことができ、結果として開発期間の短縮につながります。

関数は、FLA ファイルやクラスファイルに作成する方法と、コードを中心とするコンポーネントの ActionScript として作成する方法があります。次の例では、タイムライン上およびクラスファイル内での関数の作成方法を示します。

ヒント	クラスファイル内またはコードベースのコンポーネント内にコードをパッケージ化すると、コードの共有、配分、または再利用が簡単になります。ユーザーは、コンポーネントをインストールし、ステージにドラッグし、ファイルに保存されたコードを使用できます。たとえば、Flash で利用可能なコードベースコンポーネントのワークフロー ([ウィンドウ]-[サンプルライブラリ]-[クラス]) を使用できます。

次の例では、FLA ファイル内で関数を作成し呼び出す方法を示します。

FLA ファイル内で関数を作成し呼び出すには :

1. 新しい Flash ドキュメントを作成し、「basicFunction.fla」という名前で保存します。
2. [ウィンドウ]-[アクション] を選択して、[アクション] パネルを開きます。
3. スクリプトペインに次の ActionScript コードを入力します。

```
function helloWorld() {
   // ここにステートメントを記述
   trace("Hello world!");
};
```

   この ActionScript では、`helloWorld()` という (ユーザー定義、名前付き) 関数を定義しています。この時点で SWF ファイルをテストしても、何も起こりません。たとえば、この `trace` ステートメントによって [出力] パネルへの表示が実行されることはありません。`trace` ステートメントを表示するには、`helloWorld()` 関数を呼び出す必要があります。

4. 関数の後に、次の ActionScript コードを入力します。

```
helloWorld();
```

   このコードは `helloWorld()` 関数を呼び出します。

5. [制御]-[ムービープレビュー] を選択して FLA ファイルをテストします。

次のテキストが [出力] パネルに表示されます。`Hello world!`

値 (パラメータ) を関数に渡す方法については、173 ページの「関数へのパラメータの受け渡し」を参照してください。

メインタイムライン上で関数を記述するには複数の方法があります。最も重要なのは、名前付き関数と匿名関数の両方を使用できることです。たとえば、関数を作成する際には次のシンタックスを使用できます。

```
function myCircle(radius:Number):Number {
  return (Math.PI * radius * radius);
}
trace(myCircle(5));
```

多くの場合、匿名関数の方が読みにくくなります。次のコードと前のコードを比較してください。

```
var myCircle:Function = function(radius:Number):Number {
  // ここに関数ブロックを記述
  return (Math.PI * radius * radius);
};
trace(myCircle(5));
```

ActionScript 2.0 を使用する場合、次の例に示すように、クラスファイル内に関数を配置することもできます。

```
class Circle {
  public function area(radius:Number):Number {
    return (Math.PI * Math.pow(radius, 2));
  }
  public function perimeter(radius:Number):Number {
    return (2 * Math.PI * radius);
  }
  public function diameter(radius:Number):Number {
    return (radius * 2);
  }
}
```

クラスファイル内に関数を記述する方法の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

上記のコードサンプルからわかるとおり、関数はタイムライン上に配置する必要はありません。次の例でも、関数をクラスファイル内に配置しています。ActionScript 2.0 を使用して大規模なアプリケーションを作成する場合は、複数のアプリケーションで簡単にコードを再利用できるこの方法を採用することをお勧めします。関数を他のアプリケーションで再利用する場合には、コードを最初から書き直す代わりに既存のクラスを読み込むか、新しいアプリケーション内に関数を複写することができます。

クラスファイル内に関数を作成するには :

1. 新しい ActionScript ドキュメントを作成し、「Utils.as」という名前で保存します。
2. スクリプトペインに次の ActionScript を入力します。

```
class Utils {
  public static function randomRange(min:Number, max:Number):Number {
    if (min > max) {
      var temp:Number = min;
      min = max;
      max = temp;
    }
    return (Math.floor(Math.random() * (max - min + 1)) + min);
  }
  public static function arrayMin(num_array:Array):Number {
    if (num_array.length == 0) {
      return Number.NaN;
    }
    num_array.sort(Array.NUMERIC | Array.DESCENDING);
    var min:Number = Number(num_array.pop());
    return min;
  }
  public static function arrayMax(num_array:Array):Number {
    if (num_array.length == 0) {
      return undefined;
    }
    num_array.sort(Array.NUMERIC);
    var max:Number = Number(num_array.pop());
    return max;
  }
}
```

3. [ファイル]-[保存] を選択し、ActionScript ファイルを保存します。
4. 新しい Flash ドキュメントを作成し、"Utils.as" と同じディレクトリに、「classFunctions.fla」として保存します。
5. [ウィンドウ]-[アクション] を選択して、[アクション] パネルを開きます。
6. スクリプトペインに次の ActionScript を入力します。

```
var randomMonth:Number = Utils.randomRange(0, 11);
var min:Number = Utils.arrayMin([3, 3, 5, 34, 2, 1, 1, -3]);
var max:Number = Utils.arrayMax([3, 3, 5, 34, 2, 1, 1, -3]);
trace("month: " + randomMonth);
trace("min: " + min); // -3
trace("max: " + max); // 34
```

7. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。次のテキストが [出力] パネルに表示されます。

```
month: 7
min: -3
max: 34
```

メモ

スクリプトアシストを使用してコードを記述する方法については、『Flash ユーザーガイド』を参照してください。

関数内での変数の使用

ローカル変数は、コードを体系化し、理解しやすくするための貴重なツールです。関数では、SWF ファイルで使用されている他のすべてのスクリプトからは参照できない、その関数内でのみ有効なローカル変数を使用することができます。ローカル変数は関数本体のスコープ内で作成され、関数が終了すると消滅します。関数に渡されたパラメータもすべてローカル変数として扱われます。

メモ

関数内ではタイムライン変数も使用できます。ただし、タイムライン変数を関数内で変更する場合は、その変更の内容をコメントとして記述しておくことをお勧めします。

関数内で変数を使用するには :

1. 新しい Flash ドキュメントを作成し、flashvariables.fla という名前で保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var myName:String = "Ester";
var myAge:String = "65";
var myFavSoftware:String = "Flash";
function traceMe(yourFavSoftware:String, yourName:String, yourAge:String) {
  trace("I'm " + yourName + ", I like " + yourFavSoftware + ", and I'm " +
  yourAge + ".");
}
traceMe(myFavSoftware, myName, myAge);
```

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

パラメータの受け渡しの詳細については、173 ページの「関数へのパラメータの受け渡し」を参照してください。変数とデータの詳細については、33 ページ、第 3 章の「データおよびデータ型」を参照してください。

関数へのパラメータの受け渡し

パラメータは " 引数 " とも呼ばれ、関数のコードの処理の対象となるエレメントです。本マニュアルでは、" パラメータ " と " 引数 " を同じ意味で使用します。関数にはパラメータ (値) を渡し、それらのパラメータを使用して関数の処理を実行させることができます。関数ブロック内 (関数内のステートメント) では、パラメータの値が使用されます。

パラメータは、必須の場合と省略できる場合があります。パラメータが複数ある関数については、一部のパラメータが必須で残りのパラメータが省略可能である場合もあります。関数に十分な数のパラメータを渡さないと、不足するパラメータ値は undefined に設定されます。これにより、SWF ファイルで予期しない結果が発生する可能性があります。

たとえば、次の myFunc( ) という関数は、パラメータ someText を取ります。

```
function myFunc(someText:String):Void {
  trace(someText);
}
```

パラメータを渡すことにより、関数の呼び出し時に値を渡すことができます。この値は、次のように [出力] パネルに出力されます。

```
myFunc("This is what traces");
```

関数で未定義の値をチェックして適切なデフォルト値を設定するようになっている場合を除き、関数の呼び出し時には、常に指定された個数のパラメータを渡す必要があります。渡した値は、関数の定義で指定されているパラメータに代入されます。パラメータが不足している場合、その値は undefined に設定されます。ActionScript コードの作成時には、パラメータを関数に渡すコードを多数記述することになります。

パラメータは 1 つの関数に複数個渡すこともできます。その方法は次のように簡単です。

```
var birthday:Date = new Date(1901, 2, 3);
trace(birthday);
```

各パラメータの間はカンマ記号で区切ります。ActionScript 言語の多くのビルトイン関数は、複数のパラメータを取ります。たとえば、MovieClip クラスの startDrag( ) メソッドは、lockCenter、left、top、right、および bottom の 5 つのパラメータを取ります。

```
startDrag(lockCenter:Boolean, left:Number, top:Number, right:Number,
  bottom:Number):Void
```

関数にパラメータを渡すには :

1. 新しい Flash ドキュメントを作成し、parameters.fla という名前で保存します。
2. タイムラインのフレーム 1 に次のコードを追加します。

   ```
   function traceMe(yourMessage:String):Void {
     trace(yourMessage);
   }
   traceMe("How are you doing?");
   ```

   コードの最初の数行は、traceMe( ) というユーザー定義関数を作成します。この関数は、1 つのパラメータ yourMessage を取ります。コードの最後の行は、traceMe( ) 関数を呼び出し、ストリング値 "How are you doing?" を渡します。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

次の例では、関数に複数のパラメータを渡す方法を示します。

関数に複数のパラメータを渡すには :

1. 新しい Flash ドキュメントを作成し、「functionTest.fla」という名前で保存します。
2. メインタイムラインのフレーム 1 に次のコードを追加します。

```
function getArea(width:Number, height:Number):Number {
     return width * height;
}
```

   getArea( ) 関数は、width と height の 2 つのパラメータを取ります。

3. 関数の後に次のコードを入力します。

```
var area:Number = getArea(10, 12);
trace(area); // 120
```

   getArea( ) 関数呼び出しで、width と height にそれぞれ 10 と 12 の値を割り当て、戻り値を area インスタンスに保存します。次に、area インスタンスに保存した値をトレース出力します。

4. [制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

   [出力] パネルに 120 と表示されます。

   getArea( ) 関数のパラメータはローカル変数と同様に扱われます。つまり、関数が呼び出されている間は存在しますが、関数が終了すると消滅します。

次の例では、addNumbers( ) 関数に十分な数のパラメータを渡さない場合、ActionScript は値 NaN (非数) を返します。

可変数のパラメータを関数に渡すには :

1. 新しい Flash ドキュメントを作成し、「functionTest2.fla」という名前で保存します。
2. メインタイムラインのフレーム 1 に次のコードを追加します。

```
function addNumbers(a:Number, b:Number, c:Number):Number {
   return (a + b + c);
}
trace(addNumbers(1, 4, 6)); // 11
trace(addNumbers(1, 4)); // NaN ( 非数 )、c は undefined
trace(addNumbers(1, 4, 6, 8)); // 11
```

   addNumbers 関数に十分な数のパラメータを渡さないと、不足するパラメータにはデフォルト値として undefined が割り当てられます。渡すパラメータが多すぎると、余分なパラメータは無視されます。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

   表示される値は、11、NaN、11 です。

関数からの値の戻り

関数から値を返すには、return ステートメントを使用します。return ステートメントは、関数から返される値を指定します。return ステートメントは、関数内で実行される式を評価し、その結果を関数の値として返します。return ステートメントにより、結果は即座に呼び出し元のコードに返されます。

詳細については、『ActionScript 2.0 リファレンスガイド』の「return ステートメント」を参照してください。

関数で return ステートメントを使用する場合、次のような決まりがあります。

- 関数に対して Void ではなく戻り値の型を指定する場合は、関数の return ステートメントに続けて、戻り値を記述する必要があります。
- 戻り値の型を Void にする場合、return ステートメントを記述する必要はなく、記述する場合は return の後に値を指定してはなりません。
- 戻り値の型に関係なく、return ステートメントを使用して関数の途中で処理を抜けることができます。
- 戻り値タイプを指定しない場合は、return ステートメントを記述するかどうかは任意で選択できます。

たとえば、次の関数ではパラメータ myNum の 2 乗を返し、戻り値のデータ型は必ず数値にするように指定しています。

```
function sqr(myNum:Number):Number {
  return myNum * myNum;
}
```

関数の中には、一連の処理を実行するだけで、値を返さないものもあります。次の例では、処理済みの値が関数から返されます。返された値は、変数内に格納することによりアプリケーション内で使用できます。

値を返し、その値を変数に格納するには :

1. 新しい Flash ドキュメントを作成し、「return.fla」という名前で保存します。
2. メインタイムラインのフレーム 1 に次のコードを追加します。

   ```
   function getArea(width:Number, height:Number):Number {
        return width * height;
   }
   ```

   getArea() 関数は、width と height の 2 つのパラメータを取ります。
3. 関数の後に次のコードを入力します。

   ```
   var area:Number = getArea(10, 12);
   trace(area); // 120
   ```

   getArea() 関数呼び出しで、width と height にそれぞれ 10 と 12 の値を割り当て、戻り値を area インスタンスに保存します。次に、area インスタンスに保存した値をトレース出力します。

4. [制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

   [出力] パネルに 120 と表示されます。

   getArea() 関数のパラメータはローカル変数と同様に扱われます。つまり、関数が呼び出されている間は存在しますが、関数が終了すると消滅します。

ネストされた関数について

関数内からは別の関数を呼び出すことができます。これによって関数をネストし、Flash で特定のタスクを実行させることができます。

たとえば、タイムライン上の関数をネストして、ストリングに関する特定のタスクを実行できます。タイムラインのフレーム 1 に次のコードを入力します。

```
var myStr:String = "My marshmallow chicken is yellow.";
trace("Original string: " + myStr);
function formatText():Void {
  changeString("Put chicken in microwave.");
  trace("Changed string: " + myStr);
}
function changeString(newtext:String):Void {
  myStr = newtext;
}
// 関数を呼び出す
formatText();
```

[制御]-[ムービープレビュー] を選択して、ネストされた関数をテストします。formatText() 関数を呼び出すと、ストリングに formatText() と changeString() の両方の関数が適用されます。

メソッドについて

メソッドとは、クラスに関連付けられた関数です。クラスは、カスタムクラスまたは ActionScript 言語の一部であるビルトインクラスのいずれかです。メソッドと関数の比較については、157 ページの「関数とメソッドについて」および 159ページの「メソッドと関数の種類について」を参照してください。

たとえば、sortOn() は、Array クラスに関連付けられたビルトインメソッドです。sortOn は、Flash に組み込まれている定義済みの Array クラスです。

FLA ファイルで sortOn() メソッドを使用するには :

1. 新しい Flash ドキュメントを作成し、methods.fla という名前で保存します。
2. タイムラインのフレーム 1 に次のコードを追加します。

```
var userArr:Array = new Array();
userArr.push({firstname:"George", age:39});
userArr.push({firstname:"Dan", age:43});
userArr.push({firstname:"Socks", age:2});
userArr.sortOn("firstname");
var userArrayLenth:Number = userArr.length;
var i:Number;
for (i = 0; i < userArrayLenth; i++) {
  trace(userArr[i].firstname);
}
```

   Array クラスの `sortOn()` メソッドを使用して、`userArr` という名前の新しい Array オブジェクトを作成します。名と年齢を含む 3 つのオブジェクトを配列に格納してから、その配列を、各オブジェクトの `firstname` プロパティの値に基づいてソートします。最後にループ処理で、配列内の各アイテムについて firstname を [出力] パネルに表示します。表示される名前は頭文字のアルファベット順にソートされています。

3. [制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

   このコードの場合、[出力] パネルには次の情報が表示されます。

```
Dan
George
Socks
```

162 ページの「名前付き関数の記述」で示したとおり、タイムラインのフレーム 1 に次のコードを記述すると、その ActionScript コードでは `eatCabbage()` という関数が定義されます。

```
function eatCabbage() {
    trace("tastes bad");
}
eatCabbage();
```

ただし、`eatCabbage()` 関数をクラスファイル内に記述し、`eatCabbage()` を FLA ファイルで呼び出す場合、`eatCabbage()` はメソッドと見なされます。

次の例では、クラス内にメソッドを作成する方法を示します。

メソッドと関数を比較するには :

1. 新しい ActionScript ファイルを作成し、[ファイル]-[名前を付けて保存] を選択して、ファイルを EatingHabits.as という名前で保存します。
2. スクリプトウィンドウに次の ActionScript コードを入力します。

```
class EatingHabits {
  public function eatCabbage():Void {
    trace("tastes bad");
  }
}
```

3. "EatingHabits.as" の変更を保存します。
4. 新しい Flash ドキュメントを作成し、[ファイル]-[名前を付けて保存] を選択します。このファイルに methodTest.fla という名前を付け、"EatingHabits.as" と同じディレクトリに保存します。
5. タイムラインのフレーム 1 に次の ActionScript コードを入力します。

```
var myHabits:EatingHabits = new EatingHabits();
myHabits.eatCabbage();
```

この ActionScript を使用すると、EatingHabits クラスの `eatCabbage()` メソッドを呼び出すことになります。

メモ	ビルトインクラスのメソッドを (前述の手順で記述したカスタムクラスの他に) 使用すると、タイムラインの " メソッド " が使用されます。

6. ActionScript の前の行の後に、次のコードを追加します。

```
function eatCarrots():Void {
    trace("tastes good");
}
eatCarrots();
```

このコードでは、`eatCarrots()` 関数を記述して呼び出します。

7. [制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

メソッドの命名

メソッド名は動詞で、先頭を小文字で開始し、複数の単語を連結した名前の場合は単語の区切りを大文字で示すようにすることをお勧めします。メソッド名の例を次に示します。

```
sing();
boogie();
singLoud();
danceFast();
```

メソッドはオブジェクトに対して実行する操作を表すため、ほとんどの場合、メソッド名には動詞を使用します。変数と同様に、特殊文字を使用することや、メソッド名を数字で始めることはできません。詳細については、646 ページの「命名規則」を参照してください。

第 6 章

クラス

6

ここでは、ActionScript 2.0 を使用したクラスの使用と記述について説明します。クラスは ActionScript 2.0 の根幹であり、現在では、Flash の以前のバージョンよりもクラスの重要性が高まっています。ここでは、Flash におけるクラスの重要性について学習します。

まず、基本的な用語とクラスおよびオブジェクト指向プログラミング (OOP) との関係について説明します。次に、サンプルクラスファイル、クラスファイルの各セクションの動作、およびクラスの構成について説明します。最後に、カスタムクラスを作成する方法と、そのカスタムクラスを Flash ドキュメント内で使用する方法について説明します。Flash のクラスパスについて学習し、他の人がコードを使用する際に、コードやクラスの使用目的を簡単に理解することができるようにするためのクラスのドキュメント化についても学習します。

このセクションには、ActionScript 2.0 を使用したクラス作成に役立つコード例が記載されています。この章を読み終える頃には、一般的なクラスファイルの記述、Flash のクラス、他人が記述したクラスファイルを理解することができるようになります。

ActionScript 2.0 によるスクリプト作成に慣れていない場合は、75 ページ、第 4 章の「シンタックスと言語の基礎」および645 ページ、第 17 章の「ActionScript 2.0 のベストプラクティスとコーディング規則」を参照してください。

カスタムクラスとビルトインクラスの操作については、次のトピックを参照してください。

オブジェクト指向プログラミングと Flash について 180
カスタムクラスファイルの記述 188
アプリケーションでのカスタムクラスの使用について 191
例 : カスタムクラスの記述 214
例 : Flash でのカスタムクラスファイルの使用 227
Flash でのシンボルへのクラスの割り当て 230
クラスのコンパイルと書き出し 231
クラスとスコープについて 234
トップレベルクラスとビルトインクラスについて 237
ビルトインクラスの使用について 247

オブジェクト指向プログラミングと Flash について

ActionScript 2.0 はオブジェクト指向言語です。ActionScript と同様に、OOP 言語は " クラス " と " インスタンス " の概念に基づいています。クラスは、一連のオブジェクトを特徴付けるすべてのプロパティを定義します。たとえば、User クラスは、アプリケーションを使用するユーザーを集合的に表現します。User クラスを " インスタンス化 " すると、User クラスの 1 メンバー、つまり 1 人のユーザーが作成されます。インスタンス化によって作成される " インスタンス " は、User クラスのすべてのプロパティを持ちます。

また、クラスは新しい種類のオブジェクトを定義するために作成する " データ型 " または " テンプレート " のようなものであると考えることもできます。たとえば、アプリケーションで Lettuce というデータ型が必要な場合には、Lettuce クラスを記述することで、Lettuce オブジェクトを定義し、Lettuce のメソッド (`wash()`) やプロパティ (`leafy` または `bugs`) を割り当てることができます。クラスは、外部のスクリプトファイル内で `class` キーワードを使用して定義します。Flash オーサリングツールで外部のスクリプトファイルを作成するには、[ファイル]-[新規] を選択し、[ActionScript ファイル] を選択します。

ActionScript 2.0 では、フィルタ効果、ファイルのアップロードとダウンロード、および External API などの機能を使用できるだけでなく、Java などの他のプログラミング言語でも使用できる OOP の概念およびキーワード (`class`、`interface`、`package` など) を使用できます。ActionScript 2.0 を使用すると、再利用と拡張が可能で、堅牢かつ保守の容易なプログラム構造を記述することができます。また、コーディング支援機能とデバッグ情報により、開発時間を短縮することもできます。
ActionScript 2.0 を使用してオブジェクトを作成し、継承を確立して、カスタムクラスを作成し、Flash のトップレベルクラスとビルトインクラスを拡張できます。ここでは、クラスの作成方法とカスタムクラスの使用方法について学習します。

Flash には、約 65 種類のトップレベルクラスとビルトインクラスがあります。それらによって、基本となるプリミティブデータ型 (Array、Boolean、Date など) や、カスタムエラーとカスタムイベント、外部コンテンツ (XML、イメージ、生バイナリデータなど) をロードする手段が提供されています。また、カスタムクラスを記述して Flash ドキュメントに統合することや、トップレベルクラスを継承して独自機能の追加や既存機能の変更を行うこともできます。たとえば、この章の 203 ページの「クラスメンバーについて」では、人物の名前と年齢に関するカスタムプロパティを含む Person クラスを作成する方法について説明します。次に、このカスタムクラスをドキュメントの新規データ型とし、`new` 演算子を使用してクラスの新規インスタンスを作成します。

OOP の操作の詳細については、次のトピックを参照してください。

- 181 ページの「クラスを使用する利点」
- 181 ページの「パッケージについて」
- 185 ページの「値とデータ型について」
- 185 ページの「オブジェクト指向プログラミングの基礎」

クラスを使用する利点

OOP では、オブジェクトの分類をクラスによって定義します。クラスにはオブジェクトの特性と動作が定義されています。これは、設計図に建物の特徴が定義されているのに似ています。オブジェクト指向では、特性をプロパティ (データ) と呼び、動作をメソッド (ビヘイビア) と呼びます。カスタムクラスは外部の ActionScript (AS) ファイルに記述し、FLA ファイルをコンパイルするときにアプリケーションに読み込みます。

大規模な Flash アプリケーションを作成する場合は、クラスが非常に役立ちます。外部のクラスファイルを使用することで、アプリケーションの複雑な管理が容易になります。多くのロジックをカスタムクラス内に配置すると、コードを再利用しやすくなるだけでなく、メソッドやプロパティが " 隠蔽 " されて ActionScript コードの他の部分から見えなくなります。そのため、外部から機密情報にアクセスされたりデータが改ざんされたりするのを防ぐことができます。

クラスを使用する際には、既存クラスを継承して、新機能を追加することや既存の機能を変更することができます。たとえば、よく似たクラスを 3 つ作成する場合は、基本となるクラスを 1 つ記述してから、その基本クラスを継承するクラスを 2 つ記述することができます。後の 2 つのクラスには追加のメソッドとプロパティを記述すればよく、同じコードやロジックを複製して 3 つのファイルに記述する必要はありません。

コードを使用するもう 1 つの利点は、コードの再利用性です。たとえば、Drawing API (Application Programming Interface : アプリケーションプログラミングインターフェイス) を使用したカスタムプログレスバーを作成するカスタムクラスを作成する場合、プログレスバークラスをクラスパスに保存しておき、すべての Flash ドキュメントでそのカスタムクラスを読み込むことで、同じコードを再利用できます。クラスパスの設定の詳細については、192 ページの「クラスファイルの読み込みについて」および 193 ページの「クラスパスの設定と修正について」を参照してください。

パッケージについて

クラスを作成するとき、ActionScript クラスファイルを " パッケージ " として整理することができます。パッケージとは、所定のクラスパスディレクトリ内にあって、クラスファイルを格納しているディレクトリのことです。詳細については、192 ページの「クラスファイルの読み込みについて」および 193 ページの「クラスパスの設定と修正について」を参照してください。パッケージには、独自のクラスファイルを格納している別のパッケージを含めることができます。それらを " サブパッケージ " と呼びます。

変数名と同様に、パッケージ名は識別子になっている必要があります。つまり、先頭文字は英字、アンダースコア (_)、ドル記号 ($) のいずれかとし、それ以降の文字は、英字、数字、アンダースコア、ドル記号のいずれかである必要があります。パッケージの命名方法には、推奨される規則があります。たとえば、アンダースコアやドル記号は使用しないことをお勧めします。パッケージの命名の詳細については、655 ページの「パッケージの命名」を参照してください。

通常、パッケージは関連するクラスを整理するために使用します。たとえば、Square、Circle、Triangle という 3 つの関連するクラスがあり、それぞれが "Square.as"、"Circle.as"、"Triangle.as" というファイル内で定義されているとします。また、次のように、これらの ActionScript ファイルが、クラスパス内で指定されたディレクトリに保存されているものとします。

```
// "Square.as" ファイル
class Square {}

// "Circle.as" ファイル
class Circle {}

// "Triangle.as" ファイル
class Triangle {}
```

この 3 つのファイルは関連しているので、これらを "Shapes" という名前のパッケージ (ディレクトリ) にまとめることにします。その場合、完全修飾クラス名には、単純なクラス名の他にパッケージパスが含まれるようになります。パッケージパスはドット (.) シンタックスで表され、それぞれのドットがサブディレクトリを示します。

たとえば、個々のシェイプを定義する各 ActionScript ファイルを "Shapes" ディレクトリ内に配置した場合、次のように、新しい場所を反映するように各クラスファイルの名前を変更する必要が出てきます。

```
// "Shapes/Square.as" ファイル
class Shapes.Square {}

// "Shapes/Circle.as" ファイル
class Shapes.Circle {}

// "Shapes/Triangle.as" ファイル
class Shapes.Triangle {}
```

パッケージディレクトリ内にあるクラスを参照するには、完全修飾クラス名を指定するか、`import` ステートメントを使用してそのパッケージを読み込みます。詳細については、183 ページの「パッケージの操作」を参照してください。

クラスとパッケージとの比較

OOP では、オブジェクトの分類をクラスによって定義します。クラスとは、新しい種類のオブジェクトをアプリケーションで定義する必要がある場合に作成するデータ型のようなものです。クラスにはオブジェクトの特性と動作が定義されています。これは、設計図に建物の特徴が定義されているのに似ています。オブジェクト指向では、特性を " プロパティ " (データ) と呼び、動作を " ビヘイビア " (メソッド) と呼びます。クラスのプロパティ (クラス内に定義されている変数) とメソッドをまとめて、クラスの " メンバー " と呼びます。クラスによって定義されたプロパティとメソッドを使用するには、通常、まずクラスのインスタンスを作成する必要があります。ただし、静的メンバーのみを持つクラス (トップレベルの Math クラスなど) は例外です (249 ページの「クラス (静的) メンバーについて」および 201 ページの「静的メソッドと静的プロパティ」を参照)。インスタンスとそのクラスの関係は、家と家の設計図の関係に似ています。

Flash のパッケージとは、クラスファイルを含むディレクトリで、所定のファイルパス内に存在します。関連する複数のカスタムクラスファイルは、必要に応じて単一のディレクトリ内に置くことができます。たとえば、SteelWidget、PlasticWidget、および WoodWidget という 3 つの関連クラスがあり、"SteelWidget.as"、"PlasticWidget.as"、および "WoodWidget.as" で定義されているとします。この場合、この 3 つのクラスを Widget パッケージにまとめて整理することができます。パッケージの詳細については、183 ページの「パッケージの操作」および 217 ページの「クラスファイルの作成とパッケージ化」を参照してください。

パッケージの操作

パッケージとは、クラスファイルを含むディレクトリで、所定のクラスパスディレクトリ内に存在します。たとえば、flash.filters package はハードディスク上のディレクトリで、Flash 8 の各フィルタタイプ (BevelFilter、BlurFilter、DropShadowFilter など) に対応するいくつかのクラスファイルが入っています。

メモ	`import` ステートメントを使用するには、FLA ファイルの [パブリッシュ設定] ダイアログボックスの [Flash] タブで、ActionScript 2.0、および Flash Player 6 以降を指定する必要があります。

`import` ステートメントを使用すると、完全修飾名を指定しなくてもクラスにアクセスできるようになります。たとえば、スクリプト内で BlurFilter クラスを使用する場合、このクラスを完全修飾名 (flash.filters.BlurFilter) で参照するか、import で読み込む必要があります。import で読み込んでおくと、クラス名 (BlurFilter) での参照が可能になります。次の ActionScript コードでは、`import` ステートメントを使用する場合と完全修飾クラス名を使用する場合の違いを示します。

import ステートメントを使用して BlurFilter クラスを読み込まない場合、フィルタを使用するには、完全修飾名 (パッケージ名にクラス名を続けたもの) を使用する必要があります。

```
// import ステートメントを使わない場合
var myBlur:flash.filters.BlurFilter = new flash.filters.BlurFilter(10, 10, 3);
```

同じコードで import ステートメントを使用すると、クラス名だけで BlurFilter にアクセスできるようになり、完全修飾名を使用する必要がなくなります。これにより、入力作業を節約でき、タイプミスも減ります。

```
// import ステートメントを使う場合
import flash.filters.BlurFilter;
var myBlur:BlurFilter = new BlurFilter(10, 10, 3);
```

パッケージ内の複数のクラス (BlurFilter、DropShadowFilter、GlowFilter など) を読み込むには、2 つの方法があります。複数のクラスを読み込む方法の 1 つは、次のコードのように、クラスごとに個別の import ステートメントで読み込むことです。

```
import flash.filters.BlurFilter;
import flash.filters.DropShadowFilter;
import flash.filters.GlowFilter;
```

パッケージ内のクラスごとに個別の import ステートメントを使用すると、時間がかかり、タイプミスも発生しやすくなります。パッケージ内の複数のクラスを読み込む 2 つ目の方法は、ワイルドカードを使用して、パッケージの特定のレベルにあるすべてのクラスを読み込むことです。次の ActionScript コードは、ワイルドカード文字を使用した import ステートメントの例を示します。

```
import flash.filters.*; // flash.filters パッケージ内の各クラスを読み込む
```

import ステートメントは、それを呼び出している現在のスクリプト (フレームまたはオブジェクト) にのみ適用されます。たとえば、macr.util パッケージのすべてのクラスを Flash ドキュメントのフレーム 1 に読み込んだと仮定します。そのフレームでは、そのパッケージ内のクラスを、完全修飾クラス名ではなくクラス名で参照できます。ただし、他のフレームのスクリプトでは、そのパッケージのクラスを完全修飾名で参照するか、そのパッケージのクラスを読み込む import ステートメントを追加する必要があります。

import ステートメントを使用するときは、指定したレベルのクラスだけが読み込まれることにも注意してください。たとえば、mx.transitions パッケージ内のすべてのクラスを読み込む場合は、/transitions/ ディレクトリにあるクラスだけが読み込まれます。mx.transitions.easing パッケージ内のクラスなど、サブディレクトリにあるクラスは読み込まれません。

ヒント	読み込んだクラスがスクリプト内で使用されなかった場合、そのクラスは SWF ファイルには出力されません。したがって、SWF ファイルのサイズを気にすることなく、大きなパッケージを読み込むことができます。クラスに関連付けられたバイトコードは、それが実際に使用された場合にのみ、SWF ファイルに含まれます。

値とデータ型について

クラスを記述したり使用したりできるためには、データ、値、および型についての理解が重要です。データと型については、33 ページ、第 3 章の「データおよびデータ型」で学習しました。クラスを扱う際には、変数または ActionScript エレメントに格納できる情報の種類がデータ型 (Boolean、Number、String など) によって表されることを常に意識する必要があります。詳細については、34 ページの「データ型について」を参照してください。

式には値があり、値やプロパティには " 型 " があります。クラスのプロパティに対して値を設定および取得するには、その値とプロパティの間に型の互換性が必要です。型の互換性とは、次に示す例のように、値の型と使用されている型とが適合することを意味します。

```
var myNum:Number = 10;
```

厳密な型指定の詳細については、42 ページの「データ型の割り当てと厳密な型指定について」を参照してください。

オブジェクト指向プログラミングの基礎

次のセクションからは、ActionScript コードを記述する前に知っておく必要がある、この章で使用する用語について説明します。また、この章およびマニュアル全体の中で示す例やセクションの理解を助けるために、オブジェクト指向のプログラム開発に関する基礎知識についても簡単に説明します。この章では、これ以降、オブジェクト指向の原理に関する詳細と、Flash におけるオブジェクト指向の実装について説明します。

次のセクションからは、猫の比喩を使用して OOP の概念を説明していきます。

オブジェクト

実世界に存在するオブジェクト (物体) の例として、ここでは " 猫 " を考えてみることにします。猫には、名前、年齢、色などの特性 (状態) があります。また、猫は、眠る、食べる、のどを鳴らすなどの動作を行います。OOP の世界でも同じように、オブジェクトには決まった特性 (プロパティ) と動作 (ビヘイビア) があります。オブジェクト指向のテクニックを使用すると、実世界のオブジェクト (猫など) や、より抽象的なオブジェクト (化学工程など) をモデル化することができます。

メモ	ここでは " 動作 (ビヘイビア)" という用語を一般的な意味で用いています。Flash オーサリング環境の [ビヘイビア] パネルを指すものではありません。

オブジェクトの詳細については、40 ページの「オブジェクト (Object) データ型」を参照してください。

インスタンスとクラスメンバー

引き続き、実世界にいる猫を例にして説明します。色、年齢、名前が異なり、食べ方や、のどの鳴らし方も違う猫が何匹かいるとします。それぞれが個性を持っていますが、すべての猫は同じカテゴリに属しています。これを OOP では、同じ " 猫 " というクラスに属するメンバーであると考えます。OOP の用語で言えば、" 個々の猫は Cat クラスのインスタンスである " ということになります。

また、OOP における 1 つのクラスは、1 つの種類のオブジェクトを表す設計図を定義するものです。クラスに属する特性と動作はすべて、そのクラスの " メンバー " と呼ばれます。特性 (猫の例でいえば、名前、年齢、色など) は、クラスの " プロパティ " と呼ばれ、変数で表現されます。動作 (遊ぶ、眠るなど) は、クラスの " メソッド " と呼ばれ、関数で表現されます。

インスタンスとクラスメンバーの詳細については、203 ページの「クラスメンバーについて」および206 ページの「クラスメンバーの使用」を参照してください。

継承

OOP の主な長所の 1 つは、クラスの " サブクラス " (または " 拡張 ") を作成できることです。サブクラスは、元のクラス (スーパークラス) のプロパティとメソッドをすべて継承します。サブクラスでは、通常、追加のメソッドやプロパティを定義します。また、スーパークラスで定義されたメソッドやプロパティをオーバーライドします。サブクラスは、スーパークラスで定義されたメソッドをオーバーライドする、つまり独自の定義を用意することもできます。

スーパークラス / サブクラス構造を使用する主な利点は、各クラスで同じようなコードを再利用しやすくなる点です。たとえば、すべての動物に共通な特性と動作を含む Animal というスーパークラスを作成します。次に、Animal スーパークラスから継承したサブクラスをいくつか作成して、各タイプの動物に固有の特性と動作を追加します。

Cat クラスを作成する場合には、別のクラスを継承することができます。たとえば、哺乳類を表す Mammal クラスを作成し、すべての哺乳類に共通するプロパティとビヘイビアを定義したとします。この Mammal クラスを拡張して、Cat サブクラスを作成することができます。さらに、Cat クラスを拡張 (継承) してまた別のサブクラス (Siamese クラスなど) を作成することもできます。

サブクラスを作成することによりコードの再利用が可能になります。両方のクラスに共通するコードをすべて作り直す代わりに、既存のクラスを拡張するだけで済みます。

ヒント	複雑なアプリケーションでは、クラス階層の構築方法を決定することが、設計プロセスの重要な部分を占めます。プログラミングを開始する前には、この階層構造を決定しておく必要があります。

継承とサブクラスの詳細については、253 ページ、第 7 章の「継承」を参照してください。

インターフェイス

OOP における " インターフェイス " は、クラス定義のテンプレートです。あるインターフェイスを実装するクラスでは、そのテンプレートにあるメソッドを実装する必要があります。猫に例えるなら、インターフェイスは猫の設計図のようなものです。設計図では、必要な部品は指示されますが、部品の組み立て方法や部品の動作については指示されません。

インターフェイスを使用すると、アプリケーションに構造を追加でき、アプリケーションの保守が簡単になります。ActionScript 2.0 の場合、単一のスーパークラスしか継承はできませんが、インターフェイスを制限付きの継承として使用することで一種の多重継承が可能となります。

インターフェイスは、関係のないクラスの間に関係を持たせるための " プログラミング契約 " と考えることもできます。たとえば、プログラマがチームを組んで作業し、各メンバーは同じアプリケーションのそれぞれ異なる部分 (クラス) を担当するとします。チームでは、アプリケーションの設計の際に、さまざまなクラスが通信するために使用する一連のメソッドについて合意しています。そこで、それらのメソッド、メソッドのパラメータ、および戻り値の型を宣言するインターフェイスを作成しておきます。この場合、このインターフェイスを実装するクラスは、これらのメソッドの定義を提供する必要があります。そうしないとコンパイルエラーが起こります。

継承の詳細については、253 ページ、第 7 章の「継承」を参照してください。インターフェイスの詳細については、265 ページ、第 8 章の「インターフェイス」を参照してください。

カプセル化

オブジェクト指向の流儀に従って要領よく設計されているオブジェクトは、機能を内蔵 (カプセル化) した " ブラックボックス " と見なすことができます。そのようなオブジェクトは、プログラマがプロパティ、メソッド、およびイベント (プログラミングインターフェイス) さえ知っていれば操作することができます。オブジェクトがどのように実装されているかを詳しく知っている必要はありません。このようなアプローチにより、プログラマは抽象的な視点で考え、複雑なシステムを構築する上で必要なフレームワークを作成することができます。

ActionScript 2.0 におけるカプセル化の意義の 1 つに、メンバーアクセスの制御があります。実装の詳細は公開されず、オブジェクト外のコードからは見えないようになっています。したがって、オブジェクトの外側のコードは、(プライベートメソッドやプライベートプロパティに隠されている) 実装の細部にアクセスするのではなく、強制的にオブジェクトのプログラミングインターフェイスとのやり取りを実行します。このアプローチには、いくつかの重要な利点があります。たとえば、オブジェクトの作成者は、プログラミングインターフェイスが変わらない限り、オブジェクトの外側のコードに手を加えることなく、オブジェクトの実装を変更できます。

カプセル化の詳細については、213 ページの「カプセル化の使用について」を参照してください。

ポリモーフィズム

OOP では、" ポリモーフィズム " と呼ばれる手法を使用して、個々のクラス間の違いを表現することができます。具体的には、クラスが、対応するスーパークラスのメソッドをオーバーライドして、これらのメソッドの特殊な実装を定義することになります。Flash では、スーパークラスから継承したメソッドの特殊な実装をサブクラス内に定義できます。ただし、他のプログラミング言語とは異なり、サブクラスからスーパークラスの実装にアクセスすることはできません。

たとえば、`play()` メソッドと `sleep()` メソッドを持つ Mammal というクラスを定義したとします。次に、Mammal クラスを継承する Cat、Monkey、および Dog というサブクラスを作成します。これらのサブクラスは、個々の動物に固有の習性を反映させるために、Mammal クラスの `play()` メソッドをオーバーライドできます。たとえば、Monkey は木にぶら下がる `play()` メソッドを、Cat は毛糸の玉を追いかける `play()` メソッドを、Dog はボールを取りに行く `play()` メソッドをそれぞれ実装するといったことが可能です。`sleep()` の動作はこれらの動物にほぼ共通しているため、スーパークラスの実装を利用することができます。

ポリモーフィズムの詳細については、253 ページ、第 7 章の「継承」および 260 ページの「アプリケーションでのポリモーフィズムの使用」を参照してください。

カスタムクラスファイルの記述

次の例では、クラスファイルの各部分を検証します。クラスを記述する方法と、Flash で使用するためにクラスを変更して拡張する方法について学習します。クラスの各部分と、その読み込み方法、および Flash でカスタムクラスファイルを使用する上での関連情報についても学習します。

まず、簡単なクラスから始めます。次の例では、UserClass という簡単なクラスの構成を示します。

クラスを定義するには、アクションパネルで記述しているスクリプト内ではなく、外部のスクリプトファイルで `class` キーワードを使用します。このクラス定義の構造は、インターフェイスファイルにも当てはまります。この構造を次にまず示し、続いて、クラスの作成について説明します。

- クラスファイルは、ドキュメントコメント (コードの概要と、作成者情報およびバージョン情報) で始まります。
- 必要に応じて、`import` ステートメントを追加します。
- 次に示すように、パッケージステートメント、クラス宣言、またはインターフェイス宣言を記述します。

  ```
  class UserClass {...}
  ```

- クラスまたはインターフェイスの実装に関する注意事項を説明するコメントを追加します。これらのコメントには、クラスまたはインターフェイス全体に関係する情報を記述します。
- すべての静的変数を追加します。最初にパブリッククラス変数を書き、その後にプライベートクラス変数を続けます。

- インスタンス変数を追加します。最初にパブリックメンバー変数を書き、その後にプライベートメンバー変数を続けます。
- 次の例に示すようなコンストラクタステートメントを追加します。

  ```
  public function UserClass(username:String, password:String) {...}
  ```

- メソッドを記述します。メソッドは、アクセシビリティやスコープではなく、機能ごとにグループ化してください。そのようにメソッドを整理するとコードが読みやすくなり、理解しやすくなります。
- getter/setter メソッドをクラスファイルに記述します。

次の例では、User という名前の簡単な ActionScript クラスの場合を示します。

クラスファイルを作成するには :

1. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択して、[OK] をクリックします。
2. [ファイル]-[名前を付けて保存] を選択し、ファイルを User.as という名前で保存します。
3. 次の ActionScript コードをスクリプトウィンドウに入力します。

```
/**
  User クラス
  作成者 : John Doe
  バージョン : 0.8
  変更日 : 08/21/2005
  著作権 : Adobe Systems Incorporated

  このコードでは、新規ユーザーの作成とユーザーログイン情報の指定ができるカスタム User クラスを定義する。
*/

class User {
  // プライベートインスタンス変数
  private var __username:String;
  private var __password:String;

  // コンストラクタステートメント
  public function User(p_username:String, p_password:String) {
    this.__username = p_username;
    this.__password = p_password;
  }

  public function get username():String {
    return this.__username;
  }
  public function set username(value:String):Void {
    this.__username = value;
  }

  public function get password():String {
    return this.__password;
  }
```

```
    public function set password(value:String):Void {
      this.__password = value;
    }
}
```

4. 変更内容をクラスファイルに保存します。

   前述のコードは、クラス名、作成者、バージョン、最終修正日、著作権情報、クラスの機能概要を示す標準的な " ドキュメントコメント " で始まります。

   User クラスのコンストラクタステートメントには p_username パラメータと p_password パラメータがあり、クラスのプライベートインスタンス変数である __username と __password にコピーされます。クラス内のその他のコードでは、プライベートインスタンス変数の getter プロパティと setter プロパティを定義します。読み取り専用プロパティを作成する場合は、getter 関数を定義し、setter 関数は定義しません。たとえば、ユーザー名を定義した後に変更できないようにするには、User クラスファイルの username setter 関数を削除します。

5. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択します。

6. [ファイル]-[名前を付けて保存] を選択し、ファイルを user_test.fla という名前で保存します。保存場所は、"User.as" と同じディレクトリにします。

7. 次の ActionScript をタイムラインのフレーム 1 に入力します。

```
import User;
var user1:User = new User("un1", "pw1");
trace("Before:");
trace("\t username = " + user1.username); // un1
trace("\t password = " + user1.password); // pw1
user1.username = "1nu";
user1.password = "1wp";
trace("After:");
trace("\t username = " + user1.username); // 1nu
trace("\t password = " + user1.password); // 1wp
```

   ここで作成した User クラスは非常に単純なので、Flash ドキュメントの ActionScript も単純明快なものになっています。コードの第 1 行では、カスタム User クラスを Flash ドキュメントに読み込みます。User クラスを読み込むと、そのクラスをカスタムデータ型として使用することができます。

   User クラスの単一インスタンスが定義され、user1 という名前の変数に割り当てられます。user1 User オブジェクトに値を代入し、un1 の username を定義し、pw1 の password を定義します。次の 2 つの trace ステートメントでは、User クラスの getter 関数を使用して、user1.username と user1.password の現在値をストリングで表示します。その次の 2 行では、User クラスの setter 関数を使用して、username 変数と password 変数に新しい値を設定します。最後に、username と password の値を [出力] パネルに表示します。この trace ステートメントでは、setter 関数を使用して設定した変更後の値が表示されます。

8. FLA ファイルを保存し、[制御]-[ムービープレビュー] を選択してファイルをテストします。

   trace ステートメントの結果が [出力] パネルに表示されます。次の例では、これらのファイルをアプリケーションで使用します。

XML データとカスタムクラスファイルを使用して動的にメニューを作成する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。このサンプルでは、ActionScript `XmlMenu()` コンストラクタを呼び出し、XML メニューファイルへのパスおよび現在のタイムラインへの参照をパラメータとして渡します。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/XML_Menu" フォルダに移動して次のサンプルにアクセスします。

- XmlMenu.as
- xmlmenu.fla

アプリケーションでのカスタムクラスの使用について

188 ページの「カスタムクラスファイルの記述」では、カスタムクラスファイルを作成しました。次のセクションでは、そのクラスファイルをアプリケーションで使用します。クラスの作成作業では、少なくとも次の手順を実行します。

1. 外部の ActionScript クラスファイル内にクラスを定義します。クラスファイルの定義と記述の詳細については、188 ページの「カスタムクラスファイルの記述」を参照してください。
2. このクラスファイルを所定のクラスパスディレクトリ (Flash がクラスを検索する場所) またはアプリケーションの FLA ファイルと同じディレクトリに保存します。クラスパスの設定の詳細については、193 ページの「クラスパスの設定と修正について」を参照してください。クラスファイルの読み込みの比較と詳細については、192 ページの「クラスファイルの読み込みについて」を参照してください。
3. 別のスクリプトである FLA ドキュメントまたは外部スクリプトファイルのいずれかに、クラスのインスタンスを作成するか、オリジナルのクラスを基にしてサブクラスを作成します。クラスのインスタンスの作成については、229 ページの「クラスのインスタンスの作成例」を参照してください。

次のセクションには、ActionScript 2.0 でクラスを作成するのに役立つコード例があります。ActionScript 2.0 に慣れていない場合は、33 ページ、第 3 章の「データおよびデータ型」および75 ページ、第 4 章の「シンタックスと言語の基礎」を参照してください。

カスタムクラスの操作の詳細については、次のトピックを参照してください。

- 192 ページの「クラスファイルの読み込みについて」
- 197 ページの「Flash でのクラスファイルの使用」
- 197 ページの「クラスファイルのメソッドとプロパティの使用」
- 203 ページの「クラスメンバーについて」
- 207 ページの「getter メソッドと setter メソッドについて」

■ 196 ページの「コンパイラがクラス参照を解決する方法」
■ 210 ページの「ダイナミッククラスについて」
■ 213 ページの「カプセル化の使用について」
■ 214 ページの「クラスでの this キーワードの使用について」

XML データとカスタムクラスファイルを使用して動的にメニューを作成する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。このサンプルでは、ActionScript `XmlMenu()` コンストラクタを呼び出し、XML メニューファイルへのパスおよび現在のタイムラインへの参照をパラメータとして渡します。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/XML_Menu" フォルダに移動して次のサンプルにアクセスします。

■ XmlMenu.as
■ xmlmenu.fla

クラスファイルの読み込みについて

自分で定義したクラスやインターフェイスを使用するには、そのクラスやインターフェイスの定義を含んでいる ActionScript 外部ファイルを Flash で特定して、ファイルを読み込めるようにする必要があります。Flash でクラス、インターフェイス、関数および変数の定義を検索するディレクトリのリストをクラスパスと呼びます。Flash には、グローバルなクラスパスとドキュメントレベルのクラスパスという 2 種類のクラスパス設定があります。

■ **グローバルなクラスパス**は、すべての Flash ドキュメントで共有されるクラスパスです。これは [環境設定] ダイアログボックスで設定します。このダイアログボックスを開くには、[編集]-[環境設定] (Windows) または [Flash]-[環境設定] (Macintosh) を選択し、[カテゴリ] リストで [ActionScript] をクリックしてから、[ActionScript 2.0 設定] をクリックします。
■ **ドキュメントレベルのクラスパス**は、単一の Flash ドキュメントに固有に定義するクラスパスです。これは [パブリッシュ設定] ダイアログボックスで設定します。このダイアログボックスを開くには、[ファイル]-[パブリッシュ設定] を選択し、[Flash] タブを選択してから、[設定] ボタンをクリックします。

クラスファイルを読み込むときに適用される規則は次のとおりです。

■ `import` ステートメントは次の場所に配置できます。
  ■ クラスファイル内の、クラス定義より前の任意の場所
  ■ フレームスクリプトまたはオブジェクトスクリプトの任意の場所
  ■ `#include` ステートメントを使用してアプリケーションにインクルードする ActionScript ファイルの任意の場所

- このシンタックスを使用して、パッケージされた個々の定義を読み込みます。

```
import flash.display.BitmapData;
```

- ワイルドカードシンタックスを使用して、パッケージ全体を読み込むことができます。

```
import flash.display.*;
```

include ステートメントを使用して、ActionScript コードを Flash ドキュメント (FLA) ファイルにインクルードすることもできます。次の規則は include ステートメントに適用されます。

- include ステートメントは、基本的には、追加された ActionScript ファイル内のコンテンツのコピー & ペーストです。
- ActionScript クラスファイル内の include ステートメントは、そのファイルを含むサブディレクトリを基準とします。
- FLA ファイルの include ステートメントでは、FLA ファイル内で有効なコードだけが使用できます。include ステートメントを使用できる他の場所にも同じことが言えます。たとえば、include ステートメントがクラス定義内にある場合、追加された ActionScript ファイルにはプロパティとメソッドの定義だけが使用できます。

```
// "Foo.as"
class Foo {
  #include "FooDef.as"
}

// "FooDef.as" :
var fooProp;
function fooMethod() {}
trace("Foo"); // このステートメントはクラス定義で許可されてない。
```

include ステートメントの詳細については、『ActionScript 2.0 リファレンスガイド』の「#initclip ディレクティブ」を参照してください。クラスパスの詳細については、193 ページの「クラスパスの設定と修正について」を参照してください。

クラスパスの設定と修正について

自分で定義したクラスやインターフェイスを使用するには、そのクラス定義やインターフェイス定義を含んでいる ActionScript 外部ファイルを Flash が探し出せるようにする必要があります。Flash でクラス定義やインターフェイス定義を検索するディレクトリの一覧のことを " クラスパス " と呼びます。

ActionScript クラスファイルを作成したら、そのファイルを、クラスパス内で指定されているディレクトリのいずれか (またはそのサブディレクトリ) に保存する必要があります希望のディレクトリパスをクラスパスに追加することもできます。指定以外のディレクトリに保存した場合、Flash はスクリプト内で指定されたクラスまたはインターフェイスを解決 (または検出) できません。クラスパスディレクトリ内に作成したサブディレクトリはパッケージと呼ばれ、これを利用してクラスを編成することができます。パッケージの詳細については、217 ページの「クラスファイルの作成とパッケージ化」を参照してください。

Flash には、グローバルなクラスパスとドキュメントレベルのクラスパスという 2 種類のクラスパス設定があります。グローバルなクラスパスは、すべての Flash ドキュメントで共有されるクラスパスです。ドキュメントレベルのクラスパスは、単一の Flash ドキュメントに固有に定義するクラスパスです。

グローバルなクラスパスは外部の ActionScript ファイル と FLA ファイルに適用され、[環境設定] ダイアログボックスで設定します。このダイアログボックスを開くには、[編集]-[環境設定] (Windows) または [Flash]-[環境設定] (Macintosh) を選択し、[カテゴリ] リストで [ActionScript] をクリックしてから、[ActionScript 2.0 設定] をクリックします。ドキュメントレベルのクラスパスは、Flash ドキュメントの [パブリッシュ設定] ダイアログボックスで設定します。このダイアログボックスを開くには、[ファイル]-[パブリッシュ設定] を選択し、[Flash] タブを選択してから、[設定] ボタンをクリックします。

メモ	ActionScript ファイルの編集中に、スクリプトペインの上にある [シンタックスチェック] ボタンをクリックすると、コンパイラはグローバルのクラスパスしか参照しません。編集モードでは、ActionScript ファイルは FLA ファイルと関連付けられておらず、専用のクラスパスもありません。

グローバルなクラスパスの使用

グローバルなクラスパスは、すべての Flash ドキュメントで共有されるクラスパスです。

グローバルなクラスパスを変更するには、[環境設定] ダイアログボックスを使用します。ドキュメントレベルのクラスパス設定を変更するには、FLA ファイルの [パブリッシュ設定] ダイアログボックスを使用します。どちらの場合も、絶対ディレクトリパス (`C:/my_classes` など) を追加することも、相対ディレクトリパス (`../my_classes` や `"."` など) を追加することもできます。このダイアログボックス内の順番に基づいてディレクトリが検索されます。

デフォルトでは、グローバルなクラスパスには絶対パスと相対パスが 1 つずつ含まれています。絶対パスは [環境設定] ダイアログボックスで、$(LocalData)/Classes のように指定されます。絶対パスの場所は次のとおりです。

- Windows : < ハードディスク >¥Documents and Settings¥< ユーザー >¥Local Settings¥Application Data¥Adobe¥Adobe Flash CS3¥< 言語 >¥Configuration¥Classes
- Macintosh : < ハードディスク >/Users/< ユーザー >/Library/Application Support/Adobe/Adobe Flash CS3/< 言語 >/Configuration/Classes

メモ	絶対パスで示されたグローバルなクラスパスを削除してはなりません。Flash では、このクラスパスを使用してビルトインクラスにアクセスします。このクラスパスをうっかり削除してしまった場合は、新しいクラスパスとして $(LocalData)/Classes を追加し、復元してください。

グローバルなクラスパスの相対パス部分 (単一のドット (.)) は、現在のドキュメントディレクトリを指しています。この相対クラスパスは、コンパイルまたはパブリッシュされるドキュメントの場所によっては別のディレクトリを指すこともあるという点に注意してください。

グローバルなクラスパスを追加する場合や、既存のクラスパスを編集する場合は、次の手順に従います。

グローバルなクラスパスを変更するには :

1. [編集]-[環境設定] (Windows) または [Flash]-[環境設定] (Macintosh) を選択して、[環境設定] ダイアログボックスを開きます。
2. 左列の [ActionScript] をクリックし、[ActionScript 2.0 設定] ボタンをクリックします。
3. [パスの参照] ボタンをクリックし、追加するディレクトリを参照します。
4. 追加するパスを参照し、[OK] をクリックします。

クラスパスからディレクトリを削除するには :

1. [クラスパス] リストでパスを選択します。
2. [選択したパスの削除] ボタンをクリックします。

メモ	絶対パスで示されたグローバルなクラスパスを削除してはなりません。Flash では、このクラスパスを使用してビルトインクラスにアクセスします。このクラスパスをうっかり削除してしまった場合は、新しいクラスパスとして $(LocalData)/Classes を追加して復元できます。

パッケージの読み込みの詳細については、183 ページの「パッケージの操作」を参照してください。

ドキュメントレベルのクラスパスの使用

ドキュメントレベルのクラスパスは FLA ファイルだけに適用されます。ドキュメントレベルのクラスパスは、特定の FLA ファイルの [パブリッシュ設定] ダイアログボックスで設定します。このダイアログボックスを開くには、[ファイル]-[パブリッシュ設定] を選択し、[Flash] タブを選択してから、[ActionScript 2.0 設定] をクリックします。ドキュメントレベルのクラスパスは、デフォルトでは空になっています。FLA ファイルを作成および保存したディレクトリが所定のクラスパスディレクトリになります。

たとえば次のような状況では、クラスを作成した後で必要に応じてそれを特定のディレクトリに配置し、配置場所をグローバルクラスパスディレクトリのリストに追加することができます。

- すべてのプロジェクトで使用されるユーティリティクラスのセットが存在する場合
- その外部 ActionScript ファイル内にあるコードのシンタックスを ([シンタックスチェック] ボタンをクリックして) チェックする場合

カスタムクラスをデフォルトのグローバルクラスパスのディレクトリに配置すると、Flash をアンインストールして再インストールした場合などに、ディレクトリが削除または上書きされてカスタムクラスが失われることがあります。別のディレクトリに配置すれば、そのような事態を防ぐことができます。

たとえば、カスタムクラス用に次のようなディレクトリを作成できます。

- Windows : ‹ ハードディスク ›¥Documents and Settings¥‹ ユーザー ›¥custom classes
- Macintosh : ‹ ハードディスク ›/Users/‹ ユーザー ›/custom classes

次に、上記のパスをグローバルクラスパスのリストに追加します (194 ページの「グローバルなクラスパスの使用」を参照)。

Flash は FLA スクリプト内のクラス参照を解決する際、最初に、その FLA ファイルに対して指定されたドキュメントレベルのクラスパスを検索します。指定されたクラスパスに当該のクラスが見つからなかった場合、またはクラスパスに何も指定されていなかった場合は、グローバルクラスパスが検索されます。グローバルクラスパスでもクラスが見つからなかった場合は、コンパイルエラーが発生します。

ドキュメントレベルのクラスパスを修正するには :

1. [ファイル]-[パブリッシュ設定] を選択して [パブリッシュ設定] ダイアログボックスを開きます。
2. [Flash] タブをクリックします。
3. [ActionScript のバージョン] ポップアップメニューの隣にある [設定] ボタンをクリックします。
4. ファイルパスを手動で入力するか、[パスの参照] ボタンをクリックして、クラスパスに追加するディレクトリを選択します。

メモ

既存のクラスパスディレクトリを編集するには、[クラスパス] リスト内で目的のパスを選択し、[パスの参照] ボタンをクリックし、追加するディレクトリを選択して、[OK] をクリックします。

メモ

クラスパスからディレクトリを削除するには、[クラスパス] リスト内のパスを選択し、[選択したパスの削除] (-) ボタンをクリックします。

パッケージの詳細については、181 ページの「パッケージについて」を参照してください。

コンパイラがクラス参照を解決する方法

Flash は FLA スクリプト内のクラス参照を解決する際、最初に、その FLA ファイルに対して指定されたドキュメントレベルのクラスパスを検索します。指定されたクラスパスに当該のクラスが見つからなかった場合、またはクラスパスに何も指定されていなかった場合は、グローバルクラスパスが検索されます。グローバルクラスパスでもクラスが見つからなかった場合は、コンパイルエラーが発生します。

ActionScript ファイルの編集中に [シンタックスチェック] ボタンをクリックすると、コンパイラはグローバルクラスパス内だけを検索します。編集モードでは ActionScript ファイルが FLA と関連付けられておらず、その ActionScript ファイルのクラスパスが存在しないためです。

Flash でのクラスファイルの使用

ActionScript クラスのインスタンスを作成するには、new 演算子を使用して、そのクラスのコンストラクタ関数を呼び出します。コンストラクタ関数の名前は、必ずクラスの名前と同じになります。コンストラクタ関数を実行すると、クラスのインスタンスが返されるので、通常はその戻り値を変数に代入します。たとえば、188 ページの「カスタムクラスファイルの記述」の User クラスを使用する場合、次のコードを記述して新規 User オブジェクトを作成します。

```
var firstUser:User = new User();
```

メモ

場合によっては、インスタンスを作成せずにクラスのプロパティとメソッドを使用できることがあります。クラスメンバー (静的メンバー) の詳細については、249 ページの「クラス (静的) メンバーについて」および 201 ページの「静的メソッドと静的プロパティ」を参照してください。

インスタンスのプロパティ値を参照するには、ドット (.) 演算子を使用します。ドットの左側にインスタンス名を指定し、右側にプロパティ名を指定します。次の例では、firstUser がインスタンスで、username がプロパティです。

```
firstUser.username
```

Flash ドキュメントの ActionScript 言語を構成するトップレベルクラスまたはビルトインクラスを使用することもできます。たとえば、次のコードでは、新しい Array オブジェクトを作成し、その length プロパティを表示します。

```
var myArray:Array = new Array("apples", "oranges", "bananas");
trace(myArray.length); // 3
```

Flash でカスタムクラスを使用する場合の詳細については、227 ページの「例 : Flash でのカスタムクラスファイルの使用」を参照してください。コンストラクタ関数の詳細については、220 ページの「コンストラクタ関数の記述」を参照してください。

クラスファイルのメソッドとプロパティの使用

OOP では、クラスのメンバー (プロパティまたはメソッド) はインスタンスメンバーまたはクラスメンバーになります。クラスのインスタンスごとにインスタンスメンバーが作成されます。これらは、クラス定義内で初期化されるときに、クラスのプロトタイプに定義されます。これに対し、クラスメンバーはクラスごとに 1 回だけ作成されます。クラスメンバーは静的メンバーと呼ばれることもあります。

プロパティとは、オブジェクトを定義する属性です。たとえば、length は配列のエレメント数を指定するプロパティで、すべての配列に備わっています。メソッドとは、クラスに関連付けられた関数です。関数とメソッドの詳細については、157 ページ、第 5 章の「関数とメソッド」を参照してください。

次の例では、クラスファイルでメソッドを作成する方法を示します。

```
class Sample {
  public function myMethod():Void {
    trace("myMethod");
  }
}
```

次に、ドキュメントでメソッドを実行します。インスタンスメソッドを呼び出したり、インスタンスプロパティにアクセスしたりするには、クラスのインスタンスを参照します。次の例では、カスタム Picture クラス (次の実習で作成) のインスタンスである picture01 で、showInfo() メソッドを呼び出しています。

```
var img1:Picture = new Picture("http://www.helpexamples.com/flash/images/
  image1.jpg");
// showInfo() メソッドを呼び出す。
img1.showInfo();
```

次の例では、カスタム Picture クラスを記述して写真の各情報を維持する方法について説明しています。

FLA ファイルで Picture クラスと PictureClass クラスを使用するには :

1. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択します。ドキュメントを Picture.as という名前で保存し、[OK] をクリックします。

   このドキュメントにカスタム Picture クラスを記述します。

2. 次の ActionScript コードをスクリプトウィンドウに入力します。

```
/**
  Picture クラス
  作成者 : John Doe
  バージョン : 0.53
  変更日 : 6/24/2005
  著作権 : Adobe Systems Incorporated

  Picture クラスはイメージとその URL のコンテナとして使用される。
*/

class Picture {
  private var __infoObj:Object;

  public function Picture(src:String) {
    this.__infoObj = new Object();
    this.__infoObj.src = src;
  }

  public function showInfo():Void {
    trace(this.toString());
  }
  private function toString():String {
    return "[Picture src=" + this.__infoObj.src + "]";
  }
```

```
    public function get src():String {
      return this.__infoObj.src;
    }
    public function set src(value:String):Void {
      this.__infoObj.src = value;
    }
}
```

3. ActionScript ファイルを保存します。
4. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して新しい FLA ファイルを作成します。Picture クラスファイルを保存したディレクトリと同じディレクトリに picture_test.fla という名前で保存します。
5. 次の ActionScript コードをタイムラインのフレーム 1 に入力します。

```
var picture1:Picture = new Picture("http://www.helpexamples.com/flash/images/
  image1.jpg");
picture1.showInfo();
this.createEmptyMovieClip("img_mc", 9);
img_mc.loadMovie(picture1.src);
```

6. Flash ドキュメントを保存します。
7. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   次のテキストが [出力] パネルに表示されます。

```
[Picture src=http://www.helpexamples.com/flash/images/image1.jpg]
```

XML データとカスタムクラスファイルを使用して動的にメニューを作成する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。このサンプルでは、ActionScript `XmlMenu()` コンストラクタを呼び出し、XML メニューファイルへのパスおよび現在のタイムラインへの参照をパラメータとして渡します。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/XML_Menu" フォルダに移動して次のサンプルにアクセスします。

- XmlMenu.as
- xmlmenu.fla

パブリック、プライベート、および静的なメソッドとプロパティ (メンバー) について

外部のスクリプトファイルに ActionScript クラスファイルを記述するとき、4 種類のメソッドとプロパティを作成できます。4 種類のメソッドとプロパティとは、パブリックメソッドとパブリックプロパティ、プライベートメソッドとプライベートプロパティ、パブリック静的メソッドとパブリック静的プロパティ、およびプライベート静的メソッドとプライベート静的プロパティです。このメソッドとプロパティを使用して、Flash で変数にアクセスする方法を定義し、特定のメソッドやプロパティにアクセスできるコードを指定することができます。

クラスベースのアプリケーションを作成するとき、アプリケーションの大きさに関係なく、メソッドまたはプロパティをプライベートにするかパブリックにするかを考慮することが特に重要です。考慮することで、コードのセキュリティを高めることができます。たとえば、User クラスを作成する場合に、このクラスを使用するユーザーに対して、ユーザー ID の変更を禁止するとします。クラスプロパティ (" インスタンスメンバー " とも呼ぶ) を private に設定すると、クラス内またはそのクラスのサブクラス内にあるコードに対して、このプロパティへのアクセスを制限することができ、ユーザーがこのプロパティを直接変更することができなくなります。

パブリックメソッドとパブリックプロパティ

public キーワードは、すべての呼び出し元で変数または関数にアクセスできることを示します。デフォルトでは変数および関数がパブリックとして扱われるため、this キーワードは主に形式上の使用され、変数が現在のスコープに存在することを示します。たとえば、this キーワードを使用して、プライベート変数または静的変数が含まれるコードブロックでパブリック変数を明確に区別する場合などです。this キーワードは、public キーワードまたは private キーワードと共に使用できます。

次の Sample クラスには、既に myMethod() という名前のパブリックメソッドがあります。

```
class Sample {
  private var ID:Number;
  public function myMethod():Void {
    this.ID = 15;
    trace(this.ID); // 15
    trace("myMethod");
  }
}
```

パブリックプロパティを追加する場合は、次のコード例に示すように、"private" ではなく "public" を使用します。

```
class Sample {
  private var ID:Number;
  public var email:String;
  public function myMethod():Void {
    trace("myMethod");
  }
}
```

この email プロパティはパブリックなので、Sample クラス内から変更できるだけでなく、FLA 内からも直接変更できます。

プライベートメソッドとプライベートプロパティ

private キーワードは、変数または関数を宣言または定義するクラス、またはそのクラスのサブクラスだけがこれらにアクセスできるように指定します。デフォルトでは、変数または関数がパブリックであり、すべての呼び出し元でアクセスできます。変数または関数へのアクセスを制限する場合は、次の例に示すように、this キーワードを使用します。

```
class Sample {
  private var ID:Number;
  public function myMethod():Void {
    this.ID = 15;
    trace(this.ID); // 15
    trace("myMethod");
  }
}
```

プライベートプロパティを前述のクラスに追加する場合は、var キーワードの前に private キーワードを指定します。

Sample クラスの外部からプライベート ID プロパティにアクセスすると、コンパイルエラーになり、[出力] パネルにメッセージが表示されます。このメッセージは、メンバーがプライベートで、アクセスできないことを示します。

静的メソッドと静的プロパティ

static キーワードは、変数または関数を、そのクラスを基に生成される各オブジェクトに対して生成するのではなく、クラスに対して 1 回だけ生成するよう指定します。静的クラスメンバーは、クラスのインスタンスを作成しなくてもアクセスできます。静的メソッドと静的プロパティのスコープは、パブリックでもプライベートでもかまいません。

静的メンバーは " クラスメンバー " とも呼ばれ、クラスのインスタンスではなくクラス自体に割り当てられます。クラスメソッドを呼び出したり、クラスプロパティにアクセスしたりするには、次のコードに示すように、クラスの特定のインスタンスではなくクラス名を参照します。

```
trace(Math.PI / 8); // 0.392699081698724
```

このコード 1 行を [アクション] パネルのスクリプトペインに入力すると、トレース結果が [出力] パネルに表示されます。

たとえば、前述の Sample クラスの例では、作成されたクラスのインスタンス数をトレースするための静的変数を作成することができます。そのコードを次に示します。

```
class Sample {
  public static var count:Number = 0;
  private var ID:Number;
  public var email:String;
  public function Sample() {
    Sample.count++;
    trace("count updated: " + Sample.count);
  }
```

```
  public function myMethod():Void {
    trace("myMethod");
  }
}
```

Sample クラスの新規インスタンスを作成するたびに、それまで定義された Sample クラスのインスタンスの総数がコンストラクタメソッドでトレース出力されます。

トップレベルの ActionScript クラスの中には、クラスメンバー (または静的メンバー) を持つものがあります。前のセクションで呼び出した `Math.PI` プロパティはこれに該当します。クラスメンバー (プロパティとメソッド) にアクセスするときは、クラスのインスタンスではなく、クラス名を呼び出します。つまり、静的プロパティと静的メソッドを使用するときには、クラスのインスタンスを作成する必要はありません。

たとえば、トップレベルの Math クラスは静的メソッドと静的プロパティのみで構成されています。このクラスのメソッドを呼び出すために、Math クラスのインスタンスを作成する必要はありません。その代わりに、単純に Math クラス自身のメソッドを呼び出します。次のコードでは、Math クラスの `sqrt()` メソッドを呼び出しています。

```
var squareRoot:Number = Math.sqrt(4);
trace(squareRoot); // 2
```

次のコードでは、Math クラスの `max()` メソッドを呼び出して、2 つの数値の大小を判定しています。

```
var largerNumber:Number = Math.max(10, 20);
trace(largerNumber); // 20
```

クラスメンバーの作成の詳細については、203 ページの「クラスメンバーについて」および 206 ページの「クラスメンバーの使用」を参照してください。

XML データとカスタムクラスファイルを使用して動的にメニューを作成する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。このサンプルでは、ActionScript `XmlMenu()` コンストラクタを呼び出し、XML メニューファイルへのパスおよび現在のタイムラインへの参照をパラメータとして渡します。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/XML_Menu" フォルダに移動して次のサンプルにアクセスします。

- XmlMenu.as
- xmlmenu.fla

クラスメンバーについて

この章でこれまで説明してきた大半のメンバー (メソッドおよびプロパティ) は、" インスタンスメンバー " です。各インスタンスメンバーには、クラスの個々のインスタンスごとに一意のコピーが存在します。たとえば、Sample クラスの場合、各人物の電子メールアドレスは異なるので、email メンバー変数はインスタンスメンバーです。

もう 1 種類のメンバーは " クラスメンバー " です。クラスメンバーには、クラス全体に対して使用されるコピーが 1 つだけ存在します。クラス内で (ただし関数の外側で) 宣言したすべての変数は、そのクラスのプロパティになります。次の例では、Person クラスには age と username という 2 つのプロパティがあり、それぞれのデータ型は Number と String です。

```
class Person {
  public var age:Number;
  public var username:String;
}
```

同様に、クラス内で宣言した関数は、そのクラスのメソッドになります。Person クラスの例では、getInfo() というメソッドを 1 つ作成します。

```
class Person {
  public var age:Number;
  public var username:String;
  public function getInfo():String {
    // getInfo() メソッドの定義
  }
}
```

前述の例では、Person クラスの getInfo() メソッド、および age プロパティと username プロパティは、すべてパブリックインスタンスメンバーです。age プロパティをクラスメンバーとするのは適切ではありません。年齢は人によって異なるからです。Person クラスに属するすべての人によって共有されるプロパティやメソッドだけが、クラスメンバーとなります。

たとえば、すべてのクラスで species 変数を持ち、そのクラスを表す種の適切な名前をラテン語で指定するとします。Person オブジェクトの種はすべて、"*Homo sapiens*" となるはずです。したがって、このクラスの各インスタンスごとに "Homo sapiens" というストリングの一意なコピーを保存することは無駄であり、このメンバーはクラスメンバーとするのが適切と考えられます。

クラスメンバーは、static キーワードで宣言します。たとえば、species クラスメンバーは、次のようなコードで宣言します。

```
class Person {
  public static var species:String = "Homo sapiens";
  // ...
}
```

次のコードのように、クラスのメソッドを静的なメソッドとして宣言することもできます。

```
public static function getSpecies():String {
  return Person.species;
}
```

静的メソッドは、静的プロパティだけにアクセスでき、インスタンスプロパティにはアクセスできません。たとえば、次のコードでは、クラスメソッド getAge() がインスタンス変数 age を参照しているので、コンパイルエラーになります。

```
class Person {
  public var age:Number = 15;
  // ...
  public static function getAge():Number {
    return age; /* ** エラー **: 静的関数の中ではインスタンス変数にアクセスできない */
  }
}
```

この問題を解決するには、メソッドをインスタンスメソッドにするか、変数をクラス変数にします。

クラスメンバー (または静的プロパティ) の詳細については、201 ページの「静的メソッドと静的プロパティ」を参照してください。

XML データとカスタムクラスファイルを使用して動的にメニューを作成する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。このサンプルでは、ActionScript XmlMenu() コンストラクタを呼び出し、XML メニューファイルへのパスおよび現在のタイムラインへの参照をパラメータとして渡します。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/XML_Menu" フォルダに移動して次のサンプルにアクセスします。

- XmlMenu.as
- xmlmenu.fla

シングルトンデザインパターンの使用

クラスメンバーの一般的な使用方法の 1 つに、シングルトンデザインパターンがあります。" デザインパターン " とは、コードの構造を決める際の形式的なアプローチを定義したものです。通常、デザインパターンは共通的なプログラミングの問題を解決するために構成されます。これまで多数のデザインパターンが確立されており、シングルトンもその 1 つです。シングルトンパターンでは、クラスがインスタンスを 1 つしか持てないようにすることで、アクセスをそのインスタンスに限定させることができます。シングルトンパターンの詳細については、www.adobe.com/devnet/coldfusion/articles/design_patterns.html を参照してください。

システム内に特定の種類のオブジェクトが複数存在しては都合が悪い状況があります。たとえば、チェスの試合で使われるチェス盤は 1 つだけです。また、首都は国ごとに 1 つしかありません。オブジェクトが 1 つだけであっても、その機能は 1 つのクラスとしてカプセル化することが必要です。ただし、そのオブジェクトについてはインスタンスが 1 つだけになるように管理およびアクセスする必要があります。解決策としてグローバル変数を使用することも考えられますが、多くのプロジェクトでは、グローバル変数を使用することはできるだけ避ける必要があります。より望ましいアプローチは、クラスメンバーを使用して、オブジェクトの単一のインスタンスをクラス自身に管理させることです。次の例はシングルトンパターンの一般的な使い方を示しており、Singleton インスタンスは 1 回だけ作成されます。

シングルトンデザインパターンを使用するには :

1. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択します。ドキュメントを Singleton.as という名前で保存します。
2. 次の ActionScript コードをスクリプトウィンドウに入力します。

```
/**
  Singleton クラス
  作成者 : John Doe
  バージョン : 0.53
  変更日 : 6/24/2008
  著作権 : Adobe Systems Incorporated
*/

class Singleton {
  private static var instance:Singleton = null;
  public function trackChanges():Void {
    trace("tracking changes.");
  }
  public static function getInstance():Singleton {
    if (Singleton.instance == null) {
      trace("creating new Singleton.");
      Singleton.instance = new Singleton();
    }
    return Singleton.instance;
  }
}
```

3. "Singleton.as" ドキュメントを保存します。
4. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して新しい FLA を作成し、Singleton クラスファイルを保存したディレクトリと同じディレクトリに singleton_test.fla という名前で保存します。
5. 次の ActionScript コードをタイムラインのフレーム 1 に入力します。

```
Singleton.getInstance().trackChanges(); // 変更の追跡

var s:Singleton = Singleton.getInstance(); // 変更の追跡
s.trackChanges();
```

6. Flash ドキュメントを保存します。
7. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

Singleton オブジェクトは実際に使用されるまで (つまり、他のコードが `getInstance()` メソッドを呼び出して Singleton オブジェクトを要求しない限り) 作成されません。この手法は一般的に " 遅延生成 " と呼ばれ、さまざまな状況でコードの効率化に役立ちます。

アプリケーションでは適切な数のクラスファイルを使用することが重要です。必要以上に減らしたり増やしたりすると、設計に問題のあるクラスファイルが多くなり、結局、アプリケーションのパフォーマンスやワークフローによい影響を与えないからです。常にクラスファイルを使用するように心がけ、タイムラインなどの他の場所にコードを配置しないようにしてください。ただし、わずかな機能だけを備えたクラスを多数作成したり、多くの機能を処理するクラスを少数作成したりすることは避けてください。どちらの場合も、設計に問題があることを示しています。

クラスメンバーの使用

クラスメンバー (静的メンバー) の用途の 1 つは、クラスとそのインスタンスについての状態情報を保持することです。たとえば、特定のクラスから作成されたインスタンスの数を追跡する場合などが考えられます。これを実現する簡単な方法は、新しいインスタンスが作成されるたびにインクリメントされるクラスプロパティを使用することです。

次の例では、`widgetCount` という名前の静的インスタンスカウンタを定義する Widget というクラスを作成します。このクラスの新規インスタンスが作成されるたびに、`widgetCount` の値が 1 ずつインクリメントされ、`widgetCount` の現在値が [出力] パネルに表示されます。

クラス変数を使ったインスタンスカウンタを作成するには :

1. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択して、[OK] をクリックします。
2. 次のコードをスクリプトウィンドウに入力します。

```
class Widget {
   // クラス変数を初期化する
   public static var widgetCount:Number = 0;
   public function Widget() {
      Widget.widgetCount++;
      trace("Creating widget #" + Widget.widgetCount);
   }
}
```

   `widgetCount` 変数は static として宣言されているので、一度だけ 0 に初期化されます。Widget クラスのコンストラクタステートメントが呼び出されるたびに、`widgetCount` に 1 が加算され、現在作成されているインスタンスの数が表示されます。

3. このファイルを Widget.as という名前で保存します。
4. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して新しい FLA を作成し、"Widget.as" と同じディレクトリに widget_test.fla という名前で保存します。
5. "widget_test.fla" で、タイムラインのフレーム 1 に次のコードを入力します。

```
// クラスのインスタンスを 1 つも作成していないときは、
// Widget.widgetCount はゼロ (0)。
trace("Widget count at start: " + Widget.widgetCount); // 0
var widget1:Widget = new Widget(); // 1
var widget2:Widget = new Widget(); // 2
var widget3:Widget = new Widget(); // 3
trace("Widget count at end: " + Widget.widgetCount); // 3
```

6. 変更内容を "widget_test.fla" に保存します。

7. [制御]-[ムービープレビュー] を選択してファイルをテストします。

   [出力] パネルに次の情報が表示されます。

```
Widget count at start: 0
Creating widget # 1
Creating widget # 2
Creating widget # 3
Widget count at end: 3
```

getter メソッドと setter メソッドについて

getter メソッドと setter メソッドはアクセサメソッドで、通常は、プライベートクラスメンバーを変更するためのパブリックインターフェイスです。プロパティは getter メソッドおよび setter メソッドを使用して定義されます。getter メソッドと setter メソッドは、クラス内にはメソッドとして定義しますが、クラスの外側からはプロパティとしてアクセスされます。クラス外に対するプロパティ名には、クラス内で使用するプロパティ名とは異なる名前を付けることができます。

getter メソッドと setter メソッドを使用することにはいくつかのメリットがあり、たとえば、プロパティとしてアクセスできる高機能なメンバーを作成することができます。また、読み取り専用プロパティや書き込み専用プロパティを作成することもできます。

getter メソッドと setter メソッドは便利ですが、多用しすぎるとコードの保守が困難になる場合もあるので注意が必要です。また、この 2 つのメソッドを使用すると、パブリックメンバーにアクセスするようにしてクラス実装にアクセスできます。OOP の慣例では、クラス内のプロパティに直接アクセスすることは推奨されません。

クラスを記述する際は、できる限り、インスタンス変数をプライベートで宣言してから、getter メソッドと setter メソッドを追加するようにします。これは、クラス内の特定の変数をユーザーが変更することのないようにするためです。たとえば、特定クラスで作成されたインスタンスの数を追跡するプライベート静的メソッドがある場合、ユーザーがコードを使用してこの数を変更できないようにすることが必要です。この変数をインクリメントするタイミングは、コンストラクタステートメントが呼び出されたときにだけにする必要があります。この場合、プライベートインスタンス変数を作成して、getter メソッドだけでしかアクセスできないようにします。ユーザーは getter メソッドを使用して現在値を取得するだけです。setter メソッドは使用できないので数は変更できません。クラス内の特定の変数を読み取り専用にする簡単な方法は、setter なしで getter を作成することです。

getter メソッドと setter メソッドの使用

getter メソッドと setter メソッドのシンタックスは次のとおりです。

- getter メソッドにはパラメータはなく、常に値を返します。
- setter メソッドでは必ずパラメータを指定しますが、値は返しません。

クラスでは、各プロパティについて、値を読み取るための getter メソッドと、値を書き込むための setter メソッドを定義するのが一般的です。たとえば、次のような `userName` プロパティを含むクラスがあるとします。

```
private var userName:String;
```

クラスのインスタンスが `user.userName = "Buster"` のようにしてこのプロパティに直接アクセスするのを許可する代わりに、`getUserName()` と `setUserName()` という 2 つのメソッドをクラスに用意します。これらのメソッドの実装方法を次の例で示します。

getter メソッドと setter メソッドを使用するには :

1. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択して、[OK] をクリックします。
2. 次のコードをスクリプトウィンドウに入力します。

```
class Login {
   private var __username:String;
   public function Login(username:String) {
      this.__username = username;
   }
   public function getUserName():String {
      return this.__username;
   }
   public function setUserName(value:String):Void {
      this.__username = value;
   }
}
```

3. ActionScript ドキュメントを Login.as という名前で保存します。

   このように、`getUserName()` メソッドは `userName` の現在値を返し、`setUserName()` メソッドは受け取ったストリングパラメータを `userName` に設定します。

4. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して新しい FLA を作成し、"Login.as" と同じディレクトリに login_test.fla という名前で保存します。
5. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var user:Login = new Login("RickyM");

// getUserName() メソッドの呼び出し
var userName:String = user.getUserName();
trace(userName); // RickyM

// setUserName() メソッドの呼び出し
user.setUserName("EnriqueI");
trace(user.getUserName()); // EnriqueI
```

6. [制御]-[ムービープレビュー] を選択してファイルをテストします。

   [出力] パネルに次の情報が表示されます。

```
RickyM
EnriqueI
```

さらに簡単なシンタックスを使用する場合は、暗黙的な getter メソッドと setter メソッドを使用します。暗黙的な getter メソッドと setter メソッドを使用すると、OOP の慣例を守りつつ、クラスのプロパティに直接アクセスすることができます。

これらのメソッドを定義するには、`get` と `set` というメソッド属性を使用します。プロパティの値を取得または設定するメソッドを作成し、メソッド名の前に `get` または `set` というキーワードを追加します。次に例を示します。

メモ	暗黙的な getter メソッドと setter メソッドは、ActionScript 1.0 における `Object.addProperty()` メソッドのシンタックスを簡略化したものと考えることができます。

暗黙的な getter メソッドと setter メソッドを使用するには :

1. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択して、[OK] をクリックします。
2. 次のコードをスクリプトウィンドウに入力します。

```
class Login2 {
  private var __username:String;
  public function Login2(username:String) {
    this.__username = username;
  }
  public function get userName():String {
    return this.__username;
  }
  public function set userName(value:String):Void {
    this.__username = value;
  }
}
```

3. ActionScript ドキュメントを Login2.as という名前で保存します。

   getter メソッドにはパラメータはありません。setter メソッドは必須パラメータを 1 つだけ取ります。setter メソッドは、同じスコープ内の getter メソッドと同じ名前にすることができます。getter メソッドおよび setter メソッドを他のプロパティと同じ名前にすることはできません。たとえば、前述のサンプルコードでは、`userName` という名前の getter メソッドおよび setter メソッドを定義していますが、その同じクラスに `userName` という名前のプロパティを持つことはできません。
4. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して新しい FLA を作成し、"Login2.as" と同じディレクトリに login2_test.fla という名前で保存します。

5. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var user:Login2 = new Login2("RickyM");

// "get" メソッドの呼び出し
var userNameStr:String = user.userName;
trace(userNameStr); // RickyM

// "set" メソッドの呼び出し
user.userName = "EnriqueI";
trace(user.userName); // EnriqueI
```

通常のメソッドとは異なり、getter メソッドと setter メソッドは、カッコやパラメータなしで呼び出します。getter メソッドと setter メソッドは、プロパティと同様に、同じ名前で呼び出します。

6. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択してファイルをテストします。

[出力] パネルに次の情報が表示されます。

```
RickyM
EnriqueI
```

メモ	getter メソッドと setter メソッドの属性をインターフェイスのメソッド宣言で使用することはできません。

ダイナミッククラスについて

クラス定義に dynamic キーワードを追加すると、指定したクラスに基づくオブジェクトが、実行時にダイナミックプロパティを追加したり、ダイナミックプロパティにアクセスできるようになります。この機能が必要な場合にだけ、ダイナミッククラスを作成します。

クラス定義内およびクラスインスタンス内部でアクセスされるメンバーは、クラススコープ内のメンバーと比較されないので、ダイナミッククラスに対するタイプチェックは、非ダイナミッククラスの場合よりも緩くになります。ただし、その場合でも、クラスメンバー関数の戻り値とパラメータのタイプに対しては、タイプチェックが実行されます。

ダイナミッククラスの作成方法については、211 ページの「ダイナミッククラスの作成」を参照してください。

ダイナミッククラスの作成

デフォルトでは、クラスのプロパティとメソッドは固定です。つまり、クラスのインスタンスが、そのクラス内で宣言または定義されているプロパティやメソッド以外のものを作成したり、アクセスしたりすることはできません。たとえば、`userName` と `age` の 2 つのプロパティを持つ Person クラスを考えます。

非ダイナミッククラスを作成するには :

1. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択して、[OK] をクリックします。
2. 次の ActionScript をスクリプトウィンドウに入力します。

```
class Person {
   public var userName:String;
   public var age:Number;
}
```

   別のスクリプト内でこの Person クラスのインスタンスを作成し、このクラスの存在しないプロパティにアクセスしようとすると、コンパイルエラーになります。
3. ファイルを Person.as という名前でハードディスクに保存します。
4. 新しい FLA ファイルを作成するために [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して、[OK] をクリックします。
5. [ファイル]-[名前を付けて保存] を選択し、ファイルに person_test.fla という名前を付けて、前に作成した Person クラスと同じディレクトリに保存します。
6. 次のコードを追加して、Person クラスの新しいインスタンス (`firstPerson`) を作成し、Person クラスに存在しない `hairColor` というプロパティに値を代入します。

```
var firstPerson:Person = new Person();
firstPerson.hairColor = "blue"; // エラー。"hairColor" という名前のプロパティはありません。
```

7. Flash ドキュメントを保存します。
8. [制御]-[ムービープレビュー] を選択してコードをテストします。

   Person クラスは `hairColor` という名前のプロパティを宣言していないので、このコードはコンパイルエラーになります。ほとんどの場合、これは期待どおりの動作であるはずです。コンパイルエラーというと悪いことのように思えますが、プログラマにとっては非常に有益な情報です。エラーメッセージによってコード作成時のミスが早めに指摘されるので、正確なコードを効率よく作成することができます。

場合によっては、オリジナルのクラス定義に含まれていないプロパティやメソッドを実行時に追加して、アクセスする必要が生じる場合があります。その場合に使用するのが dynamic クラス修飾子です。

ダイナミッククラスを作成するには :

1. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択して、[OK] をクリックします。
2. [ファイル]-[名前を付けて保存] を選択し、ファイルを Person2.as という名前で保存します。ファイルをハードディスクに保存します。
3. 次のコードをスクリプトウィンドウに入力します。

```
dynamic class Person2 {
  public var userName:String;
  public var age:Number;
}
```

   この ActionScript では、dynamic キーワードを前述の例の Person クラスに追加します。Person2 クラスのインスタンスでは、このクラスで定義されていないメソッドやプロパティを追加しアクセスすることができます。
4. 変更内容を ActionScript ファイルに保存します。
5. 新しい FLA ファイルを作成するために [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して、[OK] をクリックします。
6. [ファイル]-[名前を付けて保存] を選択し、ファイルを person2_test.fla という名前で保存します。保存場所は、"Person2.as" と同じディレクトリにします。
7. 次のコードを追加して、Person2 クラスの新しいインスタンス (firstPerson) を作成し、Person2 クラスに存在しない hairColor というプロパティに値を代入します。

```
var firstPerson:Person2 = new Person2();
firstPerson.hairColor = "blue";
trace(firstPerson.hairColor); // 青
```

8. 変更内容を "person2_test.fla" ファイルに保存します。
9. [制御]-[ムービープレビュー] を選択してコードをテストします。

   このカスタム Flash クラスはダイナミックなので、実行時にメソッドとプロパティをクラスに追加することができます (SWF ファイルの再生時)。コードをテストすると、テキスト blue が [出力] パネルに表示されます。

アプリケーションを開発するときは、必要でない限り、クラスはダイナミックにしないでください。ダイナミッククラスを使用しない理由の 1 つとして、クラス定義内およびクラスインスタンス内部でアクセスされるメンバーは、クラススコープ内のメンバーと比較されないので、ダイナミッククラスに対する型チェックは、非ダイナミッククラスの場合よりも寛容になる点があげられます。ただし、その場合でも、クラスメンバー関数の戻り値とパラメータの型に対しては、型チェックが実行されます。

ダイナミッククラスのサブクラスもダイナミックになりますが、例外が 1 つあります。MovieClip クラス自体はダイナミックですが、MovieClip クラスのサブクラスはデフォルトではダイナミックになりません。この実装により、MovieClip クラスのサブクラスをダイナミックにするかどうかを選択できるようになっているので、サブクラスをより柔軟に制御できます。

```
class A extends MovieClip {}         // A は非ダイナミック
dynamic class B extends A {}         // B はダイナミック
class C extends B {}                 // C はダイナミック
class D extends A {}                 // D は非ダイナミック
dynamic class E extends MovieClip{} // E はダイナミック
```

サブクラス化の詳細については、253 ページ、第 7 章の「継承」を参照してください。

カプセル化の使用について

オブジェクト指向の流儀に従って要領よく設計されているオブジェクトは、機能を内蔵 (カプセル化) した " ブラックボックス " と見なすことができます。そのようなオブジェクトは、プログラマがプロパティ、メソッド、およびイベント (プログラミングインターフェイス) さえ知っていれば操作することができます。オブジェクトがどのように実装されているかを詳しく知っている必要はありません。このようなアプローチにより、プログラマは抽象的な視点で考え、複雑なシステムを構築する上で必要なフレームワークを作成することができます。

ActionScript 2.0 におけるカプセル化の意義の 1 つに、メンバーアクセスの制御があります。実装の詳細は公開されず、オブジェクト外のコードからは見えないようになっています。したがって、オブジェクトの外側のコードは、実装の細部にアクセスするのではなく、強制的にオブジェクトのプログラミングインターフェイスとのやり取りを実行します。このアプローチには、いくつかの重要な利点があります。たとえば、オブジェクトの作成者は、プログラミングインターフェイスが変わらない限り、オブジェクトの外側のコードに手を加えることなく、オブジェクトの実装を変更できます。

Flash でのカプセル化の例として、たとえば、すべてのメンバー変数とクラス変数をプライベートに設定して、クラスを実装したユーザーがこれらの変数にアクセスするときに、getter メソッドと setter メソッドを使用するようにします。この方法でカプセル化すると、将来、変数の構造を変更する必要が生じた場合でも、クラスの変数へのアクセス方法を変更することなく、getter 関数と setter 関数の動作を変更するだけで済みます。

次のコードでは、前述の例の Person クラスを変更して、そのインスタンスメンバーをプライベートに設定し、プライベートインスタンスメンバーの getter メソッドと setter メソッドを定義する方法を示します。

```
class Person {
  private var __userName:String;
  private var __age:Number;
  public function get userName():String {
    return this.__userName;
  }
  public function set userName(value:String):Void {
    this.__userName = value;
  }
  public function get age():Number {
    return this.__age;
  }
  public function set age(value:Number):Void {
    this.__age = value;
  }
}
```

クラスでの this キーワードの使用について

クラス内で同じクラスのメソッドやメンバー変数を使用するときは、this キーワードを接頭辞として指定してください。必須ではありませんが、this キーワードを使用することで、プロパティやメソッドがそのクラスのメンバーであることがすぐわかります。接頭辞が付いていないと、プロパティやメソッドをスーパークラスのメンバーと区別できません。

静的な変数とメソッドに対してはクラス内でもクラス名接頭辞を使用できます。これで参照を修飾すれば、コードが理解しやすくなります。コーディング環境によっては、接頭辞を追加すると、コードヒントが表示される場合があります。

メモ	これらの接頭辞は必須でないため、不要と考える開発者もいます。アドビ システムズ社では、this キーワードを接頭辞として追加することをお勧めします。メソッドと変数のコンテキストが限定されてコードがわかりやすくなります。

例 : カスタムクラスの記述

これまで、クラスファイルの基礎知識とクラスのコンテンツについて学習しました。ここでは、クラスファイルを作成する上での一般的なガイドラインについて学習します。この章の 1 番目の例では、クラスを記述する方法とパッケージする方法を示します。2 番目の例では、クラスファイルと FLA ファイルを使用する方法を示します。

注意	外部ファイル内の ActionScript コードは、FLA ファイルのパブリッシュ、書き出し、ムービープレビュー、またはデバッグ時に、SWF ファイルにコンパイルされます。このため、外部ファイルに何らかの変更を行った場合には、ファイルを保存し、その外部ファイルを使用している FLA ファイルを再コンパイルする必要があります。

188 ページの「カスタムクラスファイルの記述」で説明したように、1つのクラスは " 宣言 " と " 本体 " の 2 つの部分から構成されます。最も簡単なクラス宣言は、class ステートメントと、それに続くクラス名の識別子、そして左右の中カッコ ({}) から成ります。次の例のように、中カッコの中にあるものはすべてクラスの本体です。

```
class className {
  // クラスの本体
}
```

注意 : クラスを定義できるのは、外部 ActionScript ファイル内のみです。たとえば、FLA ファイル内のフレームスクリプト上でクラスを定義することはできません。したがって、この例では新しいファイルを作成する必要があります。

最も基本的な形式のクラス宣言では、`class` キーワード、その後にクラス名 (この例では Person)、最後に左右の中カッコ ({}) が続きます。中カッコの間に記述する部分がクラスの " 本体 " と呼ばれ、クラスのプロパティとメソッドは、ここで定義されます。

この例では、クラスファイルの基本的な構成順序は次のとおりです。

- ドキュメントコメント
- クラスの宣言
- コンストラクタ関数
- クラスの本体

この章では、サブクラスは作成しません。継承とサブクラスの詳細については、253 ページ、第 7 章の「継承」を参照してください。

この例では、次のトピックを扱います。

- 216 ページの「クラス作成に関する一般的なガイドラインについて」
- 217 ページの「クラスファイルの作成とパッケージ化」
- 220 ページの「コンストラクタ関数の記述」
- 221 ページの「メソッドとプロパティの追加」
- 224 ページの「クラスでのメンバーアクセスの制御」
- 226 ページの「クラスのドキュメンテーション」

XML データとカスタムクラスファイルを使用して動的にメニューを作成する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。このサンプルでは、ActionScript `XmlMenu()` コンストラクタを呼び出し、XML メニューファイルへのパスおよび現在のタイムラインへの参照をパラメータとして渡します。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/XML_Menu" フォルダに移動して次のサンプルにアクセスします。

- XmlMenu.as
- xmlmenu.fla

クラス作成に関する一般的なガイドラインについて

カスタムクラスファイルを記述する上でのガイドラインを以下に示します。クラスを正確でわかりやすく記述するのに役立ちます。これらのガイドラインは、以降の例で実際に使用します。

- 原則として、宣言は 1 行に 1 つずつ書き、型が同じでも同じ行に複数の宣言を記述しないでください。宣言は次の例のような形式で記述します。

```
private var SKU:Number; // 製品 SKU ( 識別 ) 番号
private var quantity:Number; // 製品数量
```

- ローカル変数は、初期値が計算で決定される場合を除いて、宣言するときに初期化します。変数の初期化の詳細については、221 ページの「メソッドとプロパティの追加」を参照してください。
- 変数は、使用する前に宣言する必要があります (ループの変数も同様です)。たとえば、次のコードでは、ループイテレータ変数 (i) を for ループで使用する前に、事前に宣言しています。

```
var my_array:Array = new Array("one", "two", "three");
var i:Number;
for (i = 0 ; i < my_array.length; i++) {
   trace(i + " = " + my_array[i]);
}
```

- 上位の宣言が見えなくなるようなローカル宣言は行わないでください。たとえば、次のように同じ変数を 2 度宣言することは望ましくありません。

```
// よくない例
var counter:Number = 0;
function myMethod() {
   var counter:Number;
   for (counter = 0; counter <= 4; counter++) {
      // ステートメント ;
   }
}
```

このコードは、内側のブロックで同じ変数を宣言しています。

- わかりにくくなるので、次の ActionScript サンプルコードに示すように 1 つのステートメントで複数の変数に同じ値を代入することは避けてください。

```
// よくない例
xPos = yPos = 15;
```

または

```
// よくない例
class User {
   private var m_username:String, m_password:String;
}
```

- インスタンス変数、静的変数、クラス変数、またはメンバー変数をパブリックと宣言する場合は、相応の理由が必要です。そのような変数を作成する前に、明確にパブリックにする必要がある性質の変数かどうか確認してください。
- メンバー変数は、パブリックにする適切な理由がない限り、原則としてプライベートにします。設計の観点から見た場合、メンバー変数はプライベートにして、getter 関数と setter 関数の小さなグループを通じてのみアクセスできるようにするのが適切です。

クラスファイルの命名について

クラス名は識別子になっている必要があります。つまり、先頭文字は英字、アンダースコア (_)、ドル記号 ($) のいずれかとし、それ以降の文字は、英字、数字、アンダースコア、ドル記号のいずれかである必要があります。通常、クラス名には英字だけを使用することをお勧めします。

クラス名は、それを含む ActionScript ファイルと同じ名前にする必要があります。大文字と小文字も区別されます。たとえば、Rock というクラスを作成する場合には、そのクラス定義を含む ActionScript ファイルの名前を "Rock.as" にする必要があります。次に例を示します。

```
// "Rock.as" ファイル
class Rock {
  // Rock クラスの本体
}
```

クラス定義の命名と作成は、次のセクションで行います。クラスファイルの作成、命名、パッケージ化については、217 ページの「クラスファイルの作成とパッケージ化」を参照してください。クラスファイルの命名の詳細については、653 ページの「クラスとオブジェクトの命名」を参照してください。

クラスファイルの作成とパッケージ化

このセクションでは、この例で使用するクラスファイルの作成、命名、パッケージ化を行います (214 ページの「例 : カスタムクラスの記述」)。次のセクションでは、(単純なものですが) 完全なクラスファイルの記述方法を示します。パッケージの詳細については、181 ページの「パッケージについて」、183 ページの「クラスとパッケージとの比較」、および 183 ページの「パッケージの操作」を参照してください。

クラスファイルを作成する場合は、ファイルの保存先を決める必要があります。次の手順では、作業を単純化するため、クラスファイルと、そのクラスファイルを使用するアプリケーションの FLA ファイルを同じディレクトリに保存します。ただし、Flash でシンタックスをチェックする場合は、ファイルの場所を Flash に指定しておく必要があります。通常、アプリケーションを作成するとき、アプリケーションとクラスファイルの保存先ディレクトリを Flash のクラスパスに追加します。クラスパスの詳細については、193 ページの「クラスパスの設定と修正について」を参照してください。

クラスファイルは ActionScript (AS) ファイルとも呼ばれます。AS ファイルを作成するには、Flash オーサリングツールを使用するか、外部エディタを使用します。たとえば Macromedia Dreamweaver で AS ファイルを作成できます

メモ	クラスの名前 (ClassA) は、それを含む AS ファイル ("ClassA.as") と同じ名前にする必要があります。これは非常に重要です。この 2 つの名前が完全に一致しないと、クラスはコンパイルされません。

クラスファイルとクラス宣言を作成するには :

1. 新しい FLA ドキュメントを作成するために [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して、[OK] をクリックします。
2. [ファイル]-[名前を付けて保存] を選択して、ファイルを package_test.fla という名前にし、Flash ドキュメントを現在のディレクトリに保存します。

   後で、この Flash ドキュメントにコンテンツを追加します。
3. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択して、[OK] をクリックします。
4. [ファイル]-[名前を付けて保存] を選択し、com という名前のサブディレクトリを作成してから、次の操作を行います
   a. "com" サブディレクトリ内に adobe という名前の新しいサブディレクトリを作成します。
   b. "adobe" サブディレクトリ内に utils という名前の新しいサブディレクトリを作成します。
   c. 現在の ActionScript ドキュメントを ClassA.as という名前で "utils" ディレクトリに保存します。
5. 次のコードをスクリプトウィンドウに入力します。

```
class com.adobe.utils.ClassA {
}
```

   上記のコードでは、com.adobe.utils パッケージ内に ClassA という新しいクラスが作成されます。
6. ActionScript ドキュメント "ClassA.as" を保存します。
7. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択して、[OK] をクリックします。
8. [ファイル]-[名前を付けて保存] を選択し、ファイルに ClassB.as という名前を付けて、前の手順で作成した "ClassA.as" と同じディレクトリに保存します。
9. 次のコードをスクリプトウィンドウに入力します。

```
class com.adobe.utils.ClassB {
}
```

   上記のコードでは、com.adobe.utils パッケージ内に ClassB という新しいクラスが作成されます。
10. 変更内容を "ClassA.as" と "ClassB.as" の両方のクラスファイルに保存します。

FLA ファイルで使用するクラスファイルは、コンパイル時に SWF ファイルに読み込まれます。クラスファイルにコードを記述する際、一定の方法論と順序を意識する必要があります。以下のセクションでこれを説明します。

複数のカスタムクラスを作成する場合は、パッケージを使用してクラスファイルを整理します。パッケージとは、所定のクラスパスディレクトリ内にあって、クラスファイルを格納しているディレクトリのことです。クラス名は、宣言されたファイル内における完全修飾名である必要があります。つまり、クラスが格納されたディレクトリ (パッケージ) を反映した名前にしなければなりません。クラスパスの詳細については、193 ページの「クラスパスの設定と修正について」を参照してください。

たとえば、com.adobe.docs.YourClass という名前のクラスは com/adobe/docs に保存します。"YourClass.as" ファイルのクラス宣言は、次のようになります。

```
class com.adobe.docs.YourClass {
   // クラス
}
```

メモ

パッケージディレクトリを反映するクラス宣言は、次のセクション 214 ページの「例 : カスタムクラスの記述」で記述します。

そのため、クラスを作成する前に、どのようなパッケージ構造にするかあらかじめ決めておくことをお勧めします。そうしないと、クラスファイルを作成した後に移動する場合、新しい場所を反映させるためにクラス宣言ステートメントを修正しなければなりません。

クラスファイルをパッケージ化するには :

1. パッケージ名を決定します。

   パッケージ名は、他の開発者にもわかりやすい名前にしてください。また、固有のディレクトリ構造にパッケージ名を一致させる必要があります。たとえば、com.adobe.utils パッケージのクラスは、"com/adobe/utils" フォルダに置く必要があります。

2. パッケージ名が決定したら、必要なディレクトリ構造を作成します。

   たとえば、パッケージ名が com.adobe.utils である場合、"com/adobe/utils" というディレクトリ構造を作成し、"utils" フォルダにクラスを保存する必要があります。

3. このパッケージ内に作成するクラスには、すべて com.adobe.utils という接頭辞を使用します。

   たとえば、ClassA という名前のクラスは、完全なクラス名が com.adobe.utils.ClassA で、"com/adobe/utils/ClassA.as" クラスファイル内に配置されている必要があります。

4. 後でパッケージ構造を変更することになると、ディレクトリ構造だけではなく、各クラスファイル内のパッケージ名、すべての import ステートメント、パッケージ内のクラスへの参照も変更する必要があります。

これ以降のクラスファイル記述方法については、220 ページの「コンストラクタ関数の記述」を参照してください。

コンストラクタ関数の記述

クラス宣言の記述方法については、217 ページの「クラスファイルの作成とパッケージ化」で学習しました。ここでは、クラスファイルのコンストラクタ関数を記述します。

メモ	コメント、ステートメント、宣言の記述方法についても後で学習します。

コンストラクタとは、クラスのプロパティとメソッドを初期化 (定義) する関数です。コンストラクタとは、クラスと同じ名前を持つクラス定義内の関数です。たとえば、次のコードでは Person クラスを定義し、コンストラクタ関数を実装しています。OOP では、コンストラクタ関数でクラスの個々の新しいインスタンスを初期化します。

クラスのコンストラクタは、new 演算子を使用してクラスのインスタンスを作成するときに、自動的に呼び出される特殊な関数です。コンストラクタ関数の名前は、それを含んでいるクラスの名前と同じになります。たとえば、前述の Person クラスには、次のコンストラクタ関数が含まれています。

```
// Person クラスのコンストラクタ関数
public function Person (uname:String, age:Number) {
  this.__name = uname;
  this.__age = age;
}
```

コンストラクタ関数を記述する際には、次の点に注意してください。

- コンストラクタ関数が明示的に宣言されていない場合、つまり、クラスと同じ名前の関数を作成していない場合は、コンパイラによって空のコンストラクタ関数が自動的に作成されます。
- 1 つのクラスに含まれるコンストラクタ関数は 1 つのみです。ActionScript 2.0 ではコンストラクタ関数のオーバーロードは認められていません。
- コンストラクタには戻り値の型を指定できません。

" コンストラクタ " という用語は、通常、特定のクラスに基づいたオブジェクトを作成する (インスタンス化する) ときにも使用されます。次のステートメントでは、トップレベル Array クラスとカスタム Person クラスのコンストラクタ関数を呼び出しています。

```
var day_array:Array = new Array("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat");
var somePerson:Person = new Person("Tom", 30);
```

次に、コンストラクタ関数と呼ばれる特別な関数を追加します。

メモ	次の実習は、214 ページの「例 : カスタムクラスの記述」の一部です。この例を実施する必要がない場合は、www.helpexamples.com/flash/learnas/classes/ からクラスファイルをダウンロードしてもかまいません。

コンストラクタ関数をクラスファイルに追加するには :

1. Flash オーサリングツールで "ClassA.as" クラスファイルを開きます。
2. 次のコードに合致するように、既存のクラスファイルを変更します (太字の部分を変更します)。

```
class com.adobe.utils.ClassA {
   function ClassA() {
      trace("ClassA constructor");
   }
}
```

   上記のコードでは、ClassA クラスのコンストラクタメソッドを定義しています。このコンストラクタでは、クラスの新規インスタンスが作成されたことを通知する簡単なストリングを [出力] パネルに表示します。

3. Flash オーサリングツールで "ClassB.as" クラスファイルを開きます。
4. 次のコードに合致するように、クラスファイルを変更します (太字の部分を変更します)。

```
class com.adobe.utils.ClassB {
   function ClassB() {
      trace("ClassB constructor");
   }
}
```

5. 次に進む前に、両方の ActionScript ファイルを保存します。

これ以降のクラスファイル記述方法については、221 ページの「メソッドとプロパティの追加」を参照してください。

メソッドとプロパティの追加

ClassA クラスと ClassB クラスのプロパティを作成するには、`var` キーワードを使用して変数を定義します。

メモ	次に示す 3 つの実習は、214 ページの「例 : カスタムクラスの記述」の一部です。この例を実施する必要がない場合は、www.helpexamples.com/flash/learnas/classes/ からクラスファイルをダウンロードしてもかまいません。

ClassA クラスと ClassB クラスにプロパティを追加するには :

1. Flash オーサリングツールで "ClassA.as" および "ClassB.as" を開きます。
2. 次のコードに合致するように、"ClassA.as" ActionScript ファイルを変更します (太字の部分を変更します)。

```
class com.adobe.utils.ClassA {
   static var _className:String;
   function ClassA() {
      trace("ClassA constructor");
   }
}
```

   上記のコードブロックに、現在のクラスの名前を格納するための新規の静的変数 `_className` を 1 つ追加します。

3. 前のコードと合致するように、ClassB クラスを変更して、静的変数を追加します。
4. 次に進む前に、両方の ActionScript ファイルを保存します。

ヒント	慣例により、クラスのプロパティはクラス本体の先頭で定義します。必須ではありませんが、先頭で定義することにより、コードがわかりやすくなります。

変数の宣言では、後にコロン (:) を付けるシンタックスを使用します (`var username:String` や `var age:Number` など)。これは、厳密な型指定の例です。変数を `var variableName:variableType` という形式で宣言しておくと、その変数に代入される値が指定のデータ型に一致するかどうかが ActionScript 2.0 コンパイラによってチェックされます。このクラスを読み込む FLA ファイルで正しいデータ型が使用されていない場合は、コンパイルエラーが発生します。厳密な型指定の詳細については、42 ページの「データ型の割り当てと厳密な型指定について」を参照してください。

クラスのメンバーには、プロパティ (変数宣言) とメソッド (関数定義) の 2 種類があります。すべてのプロパティとメソッドは、クラス本体の中 (つまり中カッコ [{}] 内) で宣言および定義する必要があります。そうしないと、コンパイルエラーが発生します。メンバーの詳細については、199 ページの「パブリック、プライベート、および静的なメソッドとプロパティ (メンバー) について」を参照してください。

ClassA クラスと ClassB クラスにメソッドを追加するには :

1. Flash オーサリングツールで "ClassA.as" および "ClassB.as" を開きます。
2. 次のコードに合致するように、ClassA クラスファイルを変更します (太字の部分を変更します)。

```
class com.adobe.utils.ClassA {
  static var _className:String;

  function ClassA() {
    trace("ClassA constructor");
  }
  function doSomething():Void {
    trace("ClassA - doSomething()");
  }
}
```

上記の太字のコードブロックでは、ストリングを [出力] パネルに表示するための新規メソッドをクラスに作成します。

3. "ClassA.as" で、[ツール]-[シンタックスチェック] を選択して、ActionScript ファイルのシンタックスをチェックします。

[出力] パネルにエラーが出力される場合は、自分のスクリプトの ActionScript と、前述の手順で記述した完全なコードとを比較します。コードエラーを解決できない場合は、次に進む前に、完成版のコードをスクリプトウィンドウにコピーしてください。

4. "ClassA.as" と同様に、"ClassB.as" のシンタックスをチェックします。

   [出力] パネルにエラーが出力される場合は、次に進む前に、完成版のコードをスクリプトウィンドウにコピーしてください。

```
class com.adobe.utils.ClassB {
  static var _className:String;

  function ClassB() {
    trace("ClassB constructor");
  }
  function doSomething():Void {
    trace("ClassB - doSomething()");
  }
}
```

5. 次に進む前に、両方の ActionScript ファイルを保存します。

プロパティは、インラインで初期化できます。つまり、次のようにデフォルト値を指定して宣言できます。

```
class Person {
  var age:Number = 50;
  var username:String = "John Doe";
}
```

プロパティをインラインで初期化するときには、代入式の右辺に必ずコンパイル時定数を指定します。つまり、右辺の式では、設定や定義が実行時に行われる値を参照することはできません。コンパイル時定数には、ストリングリテラル、数値、ブール値、null、undefined に加えて、Array、Boolean、Number、Object、String というトップレベルクラスのコンストラクタ関数があります。

プロパティをインラインで初期化するには :

1. Flash オーサリングツールで "ClassA.as" および "ClassB.as" を開きます。
2. 次の ActionScript コードに合致するように、ClassA クラスファイルを変更します (太字の部分を変更します)。

```
class com.adobe.utils.ClassA {
  static var _className:String = "ClassA";

  function ClassA() {
    trace("ClassA constructor");
  }
  function doSomething():Void {
    trace("ClassA - doSomething()");
  }
}
```

   既存のクラスファイルと上記のコードブロックとの違いは、静的変数 `_className` に値 "ClassA" が定義されている点だけです。

3. ClassB クラスファイルを変更し、インラインプロパティを追加し、値を "ClassB" に変更します。
4. 次に進む前に、両方の ActionScript ファイルを保存します。

この規則はインスタンス変数 (クラスの各インスタンスにコピーされる変数) にのみ適用されるもので、クラス変数 (クラスに属する変数) には適用されません。

メモ	配列をインラインで初期化した場合、クラスのすべてのインスタンスに対して配列が 1 つだけ作成されます。

これ以降のクラスファイル記述方法については、224 ページの「クラスでのメンバーアクセスの制御」を参照してください。

クラスでのメンバーアクセスの制御

デフォルトでは、クラスのプロパティやメソッドは他のどのクラスからもアクセスできます。クラスのすべてのメンバーは、デフォルトではパブリックになります。しかし、場合によっては、クラスのデータやメソッドを他のクラスからアクセスできないようにしたいこともあります。その場合は、保護するメンバーをプライベートにして、それを宣言または定義しているクラスからのみ利用できるようにします。

メンバーをパブリックまたはプライベートとして定義するには、public または private のメンバー属性を使用します。たとえば、次のコードでは、プライベート変数 (プロパティ) とプライベートメソッド (関数) を宣言しています。次のクラス (LoginClass) は、`userName` という名前のプライベートプロパティと、`getUserName()` という名前のプライベートメソッドを定義しています。

```
class LoginClass {
  private var userName:String;
  private function getUserName():String {
    return this.userName;
  }
  // コンストラクタ :
  public function LoginClass(user:String) {
    this.userName = user;
  }
}
```

プライベートメンバー (プロパティとメソッド) は、それらのメンバーを定義しているクラスとそのオリジナルクラスのサブクラスのみに対してアクセス可能です。オリジナルクラスのインスタンス、またはそのクラスのサブクラスのインスタンスは、プライベートに宣言されたプロパティとメソッドにはアクセスできません。つまり、プライベートメンバーはクラス定義内のみでアクセス可能であり、インスタンスレベルではアクセス可能ではないということです。次の例では、クラスファイルのメンバーアクセスを変更します。

メモ	次の実習は、214 ページの「例 : カスタムクラスの記述」の一部です。この例を実施する必要がない場合は、www.helpexamples.com/flash/learnas/classes/ からクラスファイルをダウンロードしてもかまいません。

メンバーアクセスを制御するには :

1. Flash オーサリングツールで "ClassA.as" および "ClassB.as" を開きます。
2. 次の ActionScript コードに合致するように、"ClassA.as" ActionScript ファイルを変更します (太字の部分を変更します)。

```
class com.adobe.utils.ClassA {
  private static var _className:String = "ClassA";

  public function ClassA() {
    trace("ClassA constructor");
  }
  public function doSomething():Void {
    trace("ClassA - doSomething()");
  }
}
```

   上記のコードでは、両方のメソッド (ClassA コンストラクタと doSomething() メソッド) をパブリックに設定し、外部のスクリプトからアクセスできるようにします。静的変数 _className はプライベートに設定して、クラス内からだけアクセスできるようにし、外部のスクリプトからはアクセスできないようにします。

3. "ClassB.as" ActionScript ファイルを変更し、ClassA クラスと同じメソッドとプロパティのアクセスを追加します。
4. 次に進む前に、両方の ActionScript ファイルを保存します。

ClassA または ClassB のインスタンスはプライベートメンバーにアクセスできません。たとえば、次のコードは、FLA ファイルのタイムラインのフレーム 1 に追加したものですが、メソッドがプライベートであり、アクセスできないというコンパイルエラーになります。

```
import com.adobe.utils.ClassA;
var a:ClassA = new ClassA();
trace(a._className); // エラー。メンバーがプライベートで、アクセスできません。
```

メンバーアクセスの制御はコンパイル時のみの機能です。実行時には、Flash Player はプライベートメンバーとパブリックメンバーを区別しません。

これ以降のクラスファイル記述方法については、226 ページの「クラスのドキュメンテーション」を参照してください。

クラスのドキュメンテーション

クラスやインターフェイスにはドキュメンテーションの観点から、他のユーザーのためにコメントを付けておくことが大切です。たとえば、クラスファイルを Flash コミュニティに配布する場合や、一緒に作業している設計者チームや開発者チームで自分のクラスファイルを使用する場合やプロジェクトの一部として使用する場合などがあります。ドキュメンテーションによって、他のユーザーがクラスの目的などを理解しやすくなります。

典型的なクラスファイルまたはインターフェイスファイルのコメントには、" ドキュメントコメント " と " 実装コメント " の 2 種類があります。ドキュメントコメントはコードの仕様を説明するために使用し、実装内容についての説明は含めません。実装コメントは、コードをコメントアウトする目的や、コードの特定の部分の実装について説明を加えるために使用します。この 2 種類のコメントには、異なる区切り記号を使用します。ドキュメントコメントは /** と */ で区切り、実装コメントは /* と */ で区切ります。

メモ	ドキュメントコメントは、ActionScript 2.0 の言語コンストラクトではありません。ただし、AS ファイルで使用可能なクラスファイルにコメントを記述する方法としてよく使用されます。

ドキュメントコメントは、インターフェイス、クラス、メソッド、およびコンストラクタを説明するのに使用します。クラス、インターフェイス、またはメンバーごとに 1 つのドキュメントコメントを含めてください。ドキュメントコメントは、宣言の直前に配置します。

ドキュメントコメントに適さない追加情報がある場合は、実装コメント (94ページの「コメントについて」に示す形式のブロックコメントまたは単一行コメント) を使用します。実装コメントを記述する場合は、宣言の直後に記述します。

メモ	当該クラスと直接関係のないコメントは含めないでください。たとえば、対応するパッケージを説明するようなコメントは書かないようにします。

メモ	次の実習は、214 ページの「例 : カスタムクラスの記述」の一部です。この例を実施する必要がない場合は、www.helpexamples.com/flash/learnas/classes/ からクラスファイルをダウンロードしてもかまいません。

クラスファイルをドキュメント化するには :

1. Flash オーサリングツールで "ClassA.as" および "ClassB.as" を開きます。
2. ClassA クラスファイルを変更し、クラスファイルの先頭に新規コードを追加します (太字の部分を変更します)。

```
/**
  ClassA クラス
  version 1.1
  6/21/2005
  著作権 : Adobe Systems Incorporated
 */
class com.adobe.utils.ClassA {
```

```
    private static var _className:String = "ClassA";

    public function ClassA() {
      trace("ClassA constructor");
    }
    public function doSomething():Void {
      trace("ClassA - doSomething()");
    }
}
```

上記のコードでは、コメントをクラスファイルの先頭に追加しています。クラスの作成者、最終修正日、著作権情報、既知の問題点などを ActionScript ファイルと Flash ファイルにコメントとして記述しておくことは有益です。

3. "ClassB.as" ActionScript ファイルにも、クラス名などを変更して同様のコメントを記述します。
4. 次に進む前に、両方の ActionScript ファイルを保存します。

クラスのコードには、ブロックコメント、単一行コメント、後続コメントも記述することができます。コード内にコメントを記述する方法については、657 ページの「適切なコメントの書き方」を参照してください。コメントに関する一般的な情報については、95 ページの「単一行コメント」、96 ページの「複数行コメント」、および 97 ページの「後続コメント」を参照してください。

SWF ファイルでのカスタムクラスファイルの使用方法については、227 ページの「例 : Flash でのカスタムクラスファイルの使用」を参照してください。

例 : Flash でのカスタムクラスファイルの使用

この例では、214 ページの「例 : カスタムクラスの記述」で作成したクラスファイルを使用します。これらのクラスファイルは、www.helpexamples.com/flash/learnas/classes/ からダウンロードすることもできます。214 ページの「例 : カスタムクラスの記述」を完了した場合は、ハードディスク上で "ClassA.as" と "ClassB.as" を見つけてください。

ClassA クラスファイルのパッケージ名は `com.adobe.utils.ClassA` なので、クラスファイルはこれに応じた適切なディレクトリ構造に保存する必要があります。現在のディレクトリに "com" という名前のサブディレクトリを作成し、"com" フォルダ内に "adobe" という名前の新しいフォルダを作成します。最後に、"adobe" フォルダに "utils" という名前のサブフォルダを作成します。この "utils" フォルダに "ClassA.as" と "ClassB.as" の両方のクラスファイルを保存します。以上で、この例を進めるための準備が整います。

214 ページの「例 : カスタムクラスの記述」で記述したカスタムクラスは、FLA ファイルと共に使用することができます。この例では、カスタムクラスを使用して、Flash の小さなアプリケーションを作成します。作成したクラスはドキュメントをパブリッシュする際に SWF ファイルにコンパイルされ、すべてが一緒に動作します。次の実習では、クラスパスのしくみ、クラスファイルをアプリケーションで使用する方法、およびクラスとパッケージを読み込む方法について学習します。

この例を続ける場合は、228 ページの「クラスとパッケージの読み込み」に進んでください。

クラスとパッケージの読み込み

別のスクリプト内のクラスを参照するには、そのクラス名の前に、クラスのパッケージ名を接頭辞として付ける必要があります。クラス名とパッケージパスを組み合わせたものが、そのクラスの完全修飾クラス名になります。クラスがクラスパスディレクトリ内のサブディレクトリではなく、最上位のクラスパスディレクトリに置かれている場合は、完全修飾クラス名は単なるクラス名になります。

パッケージパスを指定するときは、ドット (.) 表記でディレクトリ名を区切ります。パッケージパスは階層になっており、各ドットはディレクトリのネストを表します。たとえば、ClassName という名前のクラスを作成し、それをクラスパス内の com/adobe/docs/learnAs2 パッケージに置いたとします。そのクラスのインスタンスを作成するとき、完全修飾クラス名を指定することができます。

次の例のように、変数の型を指定するときにも、この完全修飾クラス名を使用できます。

```
var myInstance:com.adobe.docs.learnAs2.ClassName = new
  com.adobe.docs.learnAs2.ClassName();
```

スクリプトにパッケージを読み込むには import ステートメントを使用できます。このとき、クラスの完全修飾名の代わりに簡略名を使用できます。また、ワイルドカード文字 (*) を使用して、パッケージ内のすべてのクラスを読み込むこともできます。ワイルドカード文字を使用すると、クラスを使用するたびに完全修飾クラス名を指定する必要がありません。

たとえば、次の例に示すように、スクリプト内で import ステートメントを使用して、このクラスを読み込んだとします。

```
import com.adobe.docs.learnAs2.util.UserClass;
```

次の例のように、このスクリプトの以降の部分では、UserClass クラスを次のような簡略名で参照することができます。

```
var myUser:UserClass = new UserClass();
```

ワイルドカード文字 (*) を使用すると、特定のパッケージ内のすべてのクラスを読み込むことができます。たとえば、"ClassA.as" と "ClassB.as" という 2 つの ActionScript クラスファイルを含む com.adobe.utils というパッケージがあるとします。別のスクリプト内で次のようなワイルドカード文字を使用すると、このパッケージ内の両方のクラスを読み込むことができます。

```
import com.adobe.utils.*;
```

次のように、このスクリプト内でいずれかのクラスを直接参照できるようになります。

```
var myA:ClassA = new ClassA();
var myB:ClassB = new ClassB();
```

import ステートメントは、それを呼び出している現在のスクリプト (フレームまたはオブジェクト) にのみ適用されます。読み込んだクラスをスクリプト内で使用しなかった場合、そのクラスは、最終的な SWF のバイトコードには含まれません。したがって、FLA ファイルの import ステートメントでロードする SWF ファイルから、そのクラスを利用することはできません。

メモ

次の実習は、「例 : カスタムクラスの記述」の例に続く、227 ページの「例 : Flash でのカスタムクラスファイルの使用」の一部です。ClassA と ClassB のクラスファイルが必要な場合は、www.helpexamples.com/flash/learnas/classes/ からダウンロードできます。

クラスまたはパッケージを読み込むには :

1. "package_test.fla" という名前のファイルを開きます。
2. 次のコードをスクリプトウィンドウに入力します。

```
import com.adobe.utils.*;
var a = new ClassA(); // ClassA のコンストラクタ
var b = new ClassB(); // ClassB のコンストラクタ
```

   上記のコードブロックでは、まず、ワイルドカード文字 (*) を使用して、com.adobe.utils パッケージ内にある各クラスを読み込みます。次に、ClassA クラスの新規インスタンスを作成します。このインスタンスは、コンストラクタメソッドでメッセージを [出力] パネルに表示します。デバッグメッセージを [出力] パネルに表示するための ClassB クラスのインスタンスも作成します。
3. 変更内容を Flash ドキュメントに保存してから、次に進みます。

これらのクラスファイルを Flash ファイルで使用する方法については、229 ページの「クラスのインスタンスの作成例」を参照してください。

クラスのインスタンスの作成例

インスタンスとは、特定のクラスのすべてのプロパティおよびメソッドを保持するオブジェクトです。たとえば、配列は Array クラスのインスタンスです。したがって、どの配列でも、Array クラスのメソッドやプロパティを使用することができます。または、UserSettings など、独自のクラスを作成して、UserSettings クラスのインスタンスを作成することができます。

227 ページの「例 : Flash でのカスタムクラスファイルの使用」で開始した例では、FLA ファイルを変更して、作成したクラスを読み込むことで、参照するたびに完全修飾クラス名を指定しなくてもよいようにしています。

この例 (227 ページの「例 : Flash でのカスタムクラスファイルの使用」) の次の手順では、"package_test.fla" Flash ドキュメントのフレームスクリプトなど、スクリプト内で ClassA クラスと ClassB クラスのインスタンスを作成し、変数に代入します。カスタムクラスのインスタンスを作成するには、トップレベルの ActionScript クラス (Date クラスや Array クラスなど) のインスタンスを作成するときと同じように、new 演算子を使用します。完全修飾クラス名を参照するか、228 ページの「クラスとパッケージの読み込み」で説明している方法でクラスを読み込みます。

メモ

次の実習は、「例 : カスタムクラスの記述」の例に続く、227 ページの「例 : Flash でのカスタムクラスファイルの使用」の一部です。

ClassA クラスと ClassB クラスの新規インスタンスを作成するには :

1. package_test.fla という名前のファイルを開きます。
2. 次の太字で示したコードをスクリプトウィンドウに入力します。

```
import com.adobe.utils.*;
var a:ClassA = new ClassA(); // ClassA のコンストラクタ
a.doSomething(); // ClassA の doSomething() メソッドを呼び出す
var b:ClassB = new ClassB(); // ClassB のコンストラクタ
b.doSomething(); // ClassB の doSomething() メソッドを呼び出す
```

   このサンプルコードのようにしてオブジェクトのデータ型を指定すると、カスタムクラスで定義されていないプロパティやメソッドへのアクセスが行われていないかコンパイラにチェックさせることができます。厳密な型指定の詳細については、42 ページの「データ型の割り当てと厳密な型指定について」を参照してください。ただし、dynamic キーワードでクラスをダイナミックにすると宣言している場合は、オブジェクトのデータ型指定の例外です。211 ページの「ダイナミッククラスの作成」を参照してください。

3. 変更内容を FLA ファイルに保存してから、次に進みます。

ここまでは、Flash ドキュメントでクラスを作成し使用するための基本的な方法について説明してきました。この他、トップレベルの ActionScript クラスまたはビルトインクラスのインスタンスを作成することもできます (247 ページの「ビルトインクラスの使用について」を参照)。

これらのクラスファイルを Flash ファイルで使用する方法については、230 ページの「Flash でのシンボルへのクラスの割り当て」を参照してください。

Flash でのシンボルへのクラスの割り当て

ステージ上のムービークリップオブジェクトなど、Flash ファイルで使用する可能性のあるシンボルにクラスを割り当てることもできます。

ムービークリップシンボルにクラスを割り当てるには :

1. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択して、[OK] をクリックします。
2. [ファイル]-[名前を付けて保存] を選択し、ファイルを Animal.as という名前でハードディスクに保存します。
3. 次のコードをスクリプトウィンドウに入力します。

```
class Animal {
  public function Animal() {
    trace("Animal::constructor");
  }
}
```

   この ActionScript では、ストリングを [出力] パネルに表示するコンストラクタメソッドを持つ Animal という名前の新規クラスを作成します。

4. 変更内容を ActionScript ファイルに保存します。

5. 新しい FLA ファイルを作成するために [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して、[OK] をクリックします。
6. [ファイル]-[名前を付けて保存] を選択し、ファイルに animal_test.fla という名前を付けて、手順 2 で作成した "Animal.as" ファイルと同じフォルダに保存します。
7. [挿入]-[新規シンボル] を選択して、[新規シンボルの作成] ダイアログボックスを表示します。
8. animal のシンボル名を入力し、[ムービークリップ] オプションを選択します。
9. 詳細オプションを設定するために、[新規シンボルの作成] ダイアログボックスの右下隅にある [詳細] ボタンをクリックします。

   [詳細] ボタンは、[新規シンボルの作成] ダイアログボックスが基本モードのときに使用できます。
10. [リンケージ] セクションの [ActionScript に書き出し] チェックボックスをオンにします。

    このオプションを有効にすると、実行時に、このシンボルのインスタンスを Flash ドキュメントに動的に割り当てることができます。
11. animal_id の識別子の値を入力し、(手順 3 で指定したクラス名と合致するように) [ActionScript 2.0 クラス] に Animal を設定します。
12. [最初のフレームに書き出し] チェックボックスをオンにし、[OK] をクリックして変更内容を適用し、ダイアログボックスを閉じます。
13. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択します。

    [出力] パネルに、Animal クラスのコンストラクタ関数のテキストが表示されます。

メモ	ムービークリップの Linkage プロパティを変更する必要がある場合は、ドキュメントのライブラリでシンボルを右クリックし、コンテキストメニューから [プロパティ] または [リンケージ] を選択します。

クラスのコンパイルと書き出し

デフォルトでは、SWF ファイルで使用されるクラスは、SWF の最初のフレームにパッケージ化され、書き出されます。また、指定したフレームについてクラスをパッケージ化して書き出すこともできます。これは、たとえば、SWF ファイルが長時間のダウンロードを必要とするような多くのクラスを使用するときに便利です (コンポーネントなど)。最初のフレームにクラスが書き出された場合、すべてのクラスコードがダウンロードされるまで待ってから最初のフレームが表示されます。タイムラインの後のフレームを指定すると、後のフレームにあるクラスコードがダウンロードされている間に最初の 2、3 のフレームで短いロード時間のアニメーションを表示することができます。

Flash ドキュメントのクラス用に書き出しフレームを指定するには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択します。新規ドキュメントを exportClasses.fla という名前で保存します。
2. デフォルトレイヤーの名前を content に変更し、ProgressBar コンポーネントを [コンポーネント] パネルからステージまでドラッグし、インスタンス名を my_pb にします。
3. 新しいレイヤーを作成し、content レイヤーの上にドラッグし、名前を actions に変更します。
4. メインタイムラインの actions レイヤーのフレーム 1 に次の ActionScript コードを追加します。

```
my_pb.indeterminate = true;
```

5. actions レイヤーのフレーム 2 に新しいキーフレームを作成し、次の ActionScript コードを追加します。

```
var classesFrame:Number = 10;
if (_framesloaded < classesFrame) {
  trace(this.getBytesLoaded() + " of " + this.getBytesTotal() + " bytes
  loaded");
  gotoAndPlay(1);
} else {
  gotoAndStop(classesFrame);
}
```

6. actions レイヤーのフレーム 10 に新しいキーフレームを作成し、次の ActionScript コードを追加します。

```
stop();
```

7. content レイヤーのフレーム 10 に新しいキーフレームを作成し、いくつかのコンポーネントをステージにドラッグします。
8. [ライブラリ] パネルで、ProgressBar 以外の各コンポーネントを右クリックし、コンテキストメニューから [リンケージ] を選択して [リンケージプロパティ] ダイアログボックスを開きます。
9. [リンケージプロパティ] ダイアログボックスで、[ActionScript に書き出し] チェックボックスをオン、[最初のフレームに書き出し] チェックボックスをオフにし、[OK] をクリックします。
10. [ファイル]-[パブリッシュ設定] を選択します。
11. [パブリッシュ設定] ダイアログボックスで、[Flash] タブを選択します。
12. [ActionScript のバージョン] ポップアップメニューの隣にある [設定] ボタンをクリックして [ActionScript 設定] ダイアログボックスを開きます。
13. [クラス用のフレームの書き出し] テキストボックスで、クラスコードを書き出すフレームの番号 (フレーム 10) を入力します。

    指定したフレームがタイムライン内に存在しない場合、SWF ファイルのパブリッシュ時にエラーメッセージが表示されます。
14. [OK] をクリックして [ActionScript 設定] ダイアログボックスを閉じ、次に [OK] をクリックして [パブリッシュ設定] ダイアログボックスを閉じます。

15. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。SWF ファイルからのコンポーネントのロードが速すぎる場合は、[表示]-[ダウンロードのシミュレート] を選択します。より低速で Flash ドキュメントのダウンロードがシミュレートされるので、クラスファイルのダウンロードにつれて、プログレスバーのコンポーネントのアニメーションを確認できます。

ASO ファイルの詳細については、233 ページの「ASO ファイルの使用」を参照してください。

ASO ファイルの使用

コンパイル時に、デフォルトのグローバルクラスパスディレクトリのサブディレクトリ /aso に、拡張子 .aso のファイルが作成される場合があります (193ページの「クラスパスの設定と修正について」を参照)。この拡張子 .aso は "ActionScript Object" (ASO) を意味しています。ActionScript 2.0 ファイルが暗黙的または明示的に読み込まれ、正常にコンパイルされると、それらの各ファイルに対する ASO ファイルが生成されます。ASO ファイルには、対応する ActionScript (AS) ファイルから作成されたバイトコードが含まれます。したがって、これらのファイルには、クラスファイルのコンパイル済み形式 (バイトコード) が格納されます。

Flash で ASO ファイルの再生成が必要になるのは、次の場合だけです。

- 対応する AS ファイルが変更された場合。
- 読み込まれた定義または対応する ActionScript ファイルで使用される定義のある ActionScript ファイルが変更された場合。
- 対応する ActionScript ファイルでインクルードされている ActionScript ファイルが変更された場合。

コンパイラはキャッシュを目的として ASO ファイルを作成します。実際にコンパイルをしてみると、初回コンパイルには、それ以降のコンパイルよりも時間がかかることがわかります。これは、変更された AS ファイルだけが再コンパイルされて、ASO ファイルに格納されるためです。変更されていない AS ファイルについては、AS ファイルを最初からコンパイルするのではなく、既にコンパイルされたバイトコードが ASO ファイルから直接読み込まれます。

ASO ファイル形式は、内部使用だけを目的として作成される中間形式です。ファイル形式を説明するドキュメントは提供されておらず、再配布を前提としたものでもありません。

編集したファイルについて、古いままの内容がコンパイルされていると考えられる問題が発生した場合は、該当する ASO ファイルを削除してから、もう一度コンパイルしてください。ASO ファイルを削除する場合は、Flash が他の作業 (シンタックスのチェックや SWF の書き出しなど) を実行していないときに削除するようにしてください。

ASO ファイルを削除するには :

FLA ファイルの編集中に ASO ファイルを削除する場合は、オーサリング環境で次のいずれかの操作を行います。

- [制御]-[ASO ファイルを削除] を選択して、ASO ファイルを削除し、編集を続けます。
- [制御]-[ASO ファイルを削除してムービーをプレビューする] を選択して、ASO ファイルを削除し、アプリケーションをテストします。

ActionScript ドキュメントをスクリプトウィンドウで編集している場合は、次の操作を行います。

- [制御]-[ASO ファイルを削除] を選択して、ASO ファイルを削除し、編集を続けます。
- [制御]-[ASO ファイルを削除してプロジェクトをテストする] を選択して、ASO ファイルを削除し、アプリケーションをテストします。

単一のクラスに格納できるコードサイズには制限があります。SWF ファイルに書き出すことのできるクラス定義のバイトコードは、最大 32,767 バイトになります。バイトコードがこの上限を超えた場合は、警告メッセージが表示されます。

特定のクラスが、バイトコードでどのぐらいのサイズになるかを事前に予測することはできませんが、一般に 1,500 行程度のクラスであれば、この制限を超えることはまずありません。

クラスのサイズが上限を超える場合は、コードの一部を別のクラスに移すようにしてください。通常、オブジェクト指向のプログラミングでは、クラスのサイズを小さくすることが推奨されます。

クラスとスコープについて

ActionScript コードをクラスに移植するとき、場合によっては `this` キーワードの使い方を変更する必要があります。たとえば、コールバック関数を使用するクラスメソッド (LoadVars クラスの `onLoad()` メソッドなど) では、`this` キーワードがクラスを指すのか LoadVars オブジェクトを指すのか判断しにくいことがあります。その場合は、次の例のようにして現在のクラスへのポインタを作成します。

スコープと外部クラスファイルを理解するには :

1. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択して、[OK] をクリックします。
2. 次のコードをスクリプトウィンドウに入力またはコピーします。

```
/**
   Product クラス
   "Product.as"
*/
class Product {
  private var productsXml:XML;
  // コンストラクタ
  // targetXmlStr - ストリング。XML ファイルへのパスを含む
  function Product(targetXmlStr:String) {
    /* 現在のクラスへのローカル参照を作成する。
```

```
        XML の onLoad イベントハンドラ内を実行している場合でも、
        XML パケットだけでなく、現在のクラスも参照できる。
    */
    var thisObj:Product = this;
    // ローカル変数を作成する。これは XML のロードに使用する。
    var prodXml:XML = new XML();
    prodXml.ignoreWhite = true;
    prodXml.onLoad = function(success:Boolean) {
      if (success) {
        /* XML を適切にロードおよび解析できた場合は、
           クラスの productsXml 変数に、解析した
           XML ドキュメントを設定し、init 関数を呼び出す。
        */
        thisObj.productsXml = this;
        thisObj.init();
      } else {
        /* XML ファイルのロードでエラーが発生した。*/
        trace("error loading XML");
      }
    };
    // XML ドキュメントのロードを開始する。
    prodXml.load(targetXmlStr);
  }
  public function init():Void {
    // XML パケットを表示する。
    trace(this.productsXml);
  }
}
```

this キーワードは、onLoad ハンドラ内のプライベートメンバー変数を参照することになるので、Product クラスではなく、実際には prodXml インスタンスを表します。このため、onLoad ハンドラからクラスを直接参照するには、このようにローカルクラスファイルへのポインタを作成する必要があります。以上で、このクラスを Flash ドキュメントで使用する準備が整います。

3. 上記の ActionScript コードを Product.as という名前で保存します。
4. 同じディレクトリに、testProduct.fla という名前の新規 Flash ドキュメントを作成します。
5. メインタイムラインのフレーム 1 を選択します。
6. 次の ActionScript を [アクション] パネルに入力します。

```
var myProduct:Product = new Product("http://www.helpexamples.com/
  crossdomain.xml");
```

7. [制御]-[ムービープレビュー] を選択して、このコードをテスト環境でテストします。

   指定した XML ドキュメントのコンテンツが [出力] パネルに表示されます。

これらのクラスを使用する際に関係する他の種類のスコープとしては、静的変数と静的関数があります。static キーワードは、変数または関数を、クラスの各インスタンスに対して作成するのではなく、クラスに対して 1 回だけ作成するよう指定します。クラスのインスタンスを作成せずに静的クラスメンバーにアクセスするには、シンタックス someClassName.username を使用します。静的変数と静的関数の詳細については、199 ページの「パブリック、プライベート、および静的なメソッドとプロパティ(メンバー) について」および206ページの「クラスメンバーの使用」を参照してください。

静的変数のもう 1 つの利点は、静的変数のスコープが終了しても、その値が失われないことです。次の例では、static キーワードを使用して、Flash で作成したクラスのインスタンス数を追跡するカウンタを作成する方法を示します。numInstances 変数は静的なので、単独の各インスタンスに対して作成されるのではなく、クラス全体に対して 1 回だけ作成されます。

static キーワードを使用するには :

1. [ファイル]-[新規] を選択し、[ActionScript ファイル] を選択して、[OK] をクリックします。
2. 次のコードをスクリプトウィンドウに入力します。

```
class User {
  private static var numInstances:Number = 0;
  public function User() {
    User.numInstances++;
  }
  public static function get instances():Number {
    return User.numInstances;
  }
}
```

   上記のコードでは、コンストラクタを呼び出した回数を追跡する User クラスを定義しています。プライベートな静的変数 (User.numInstances) は、コンストラクタメソッド内でインクリメントされます。

3. ドキュメントを User.as という名前で保存します。
4. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して新しい FLA を作成し、この FLA ファイルを "User.as" と同じディレクトリに保存します。
5. 次の ActionScript コードをタイムラインのフレーム 1 に入力します。

```
trace(User.instances); // 0
var user1:User = new User();
trace(User.instances); // 1
var user2:User = new User();
trace(User.instances); // 2
```

   コードの第 1 行では、静的 instances() getter メソッドを呼び出します。このメソッドはプライベート静的 numInstances 変数の値を返します。コードの残りの部分では、User クラスの新規インスタンスを作成し、instances() getter メソッドから返された現在値を表示します。

6. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

クラス内での this キーワードの使用方法については、214 ページの「クラスでの this キーワードの使用について」を参照してください。

トップレベルクラスとビルトインクラスについて

ActionScript には、33ページ、第3章の「データおよびデータ型」および75ページ、第4章の「シンタックスと言語の基礎」で説明した中核的な言語エレメントやコンストラクト (for ループ、while ループなど)、プリミティブデータ型 (数値型、ストリング型、およびブール型) の他に、いくつかのビルトインクラスも用意されています。単純なプリミティブデータ型に対し、ビルトインクラスは "複雑なデータ型" だと考えることができます。ビルトインクラスを使用すると、スクリプトの中でさまざまな機能や操作を実行できます。ActionScript 言語の一部であるトップレベルクラスと他のビルトインクラスを前の章で使用しましたが、この後の章でも同じものを使用します。SWF ファイルでインタラクティブ機能を作成するために使用する多数のクラスが Flash には付属しており、それらを使用して複雑なアプリケーションを作成できます。たとえば、Math クラスはアプリケーション内で式を計算するために使用できます。また、BitmapData クラスはピクセルやスクリプトアニメーションの作成に使用できます。

239 ページの「トップレベルクラス」の一覧に示すトップレベルクラスは、Flash Player に書き込まれます。[アクション] ツールボックスでは、これらのクラスは "ActionScript 2.0 クラス " ディレクトリにあります。トップレベルクラスの一部は ECMAScript (ECMA-262) Edition 3 の言語仕様に基づいたもので、それらは "ActionScript コアクラス " と呼ばれます。例としては、Array、Boolean、Date、Math の各クラスがあります。パッケージの詳細については、183 ページの「パッケージの操作」を参照してください。

ActionScript クラスはハードディスクにインストールされています。クラスフォルダは次の場所にあります。

- Windows : < ハードディスク >¥Documents and Settings¥< ユーザー >¥Local Settings¥Application Data¥Adobe¥Flash CS3¥< 言語 >¥Configuration¥Classes
- Macintosh : < ハードディスク >/Users/< ユーザー >/Library/Application Support/Adobe/Adobe Flash CS3/< 言語 >/Configuration/Classes

構造の詳細については、このディレクトリにある ReadMe ファイルを参照してください。

ActionScript コアクラスと Flash 固有のクラスの相違を理解するには、コア JavaScript とクライアントサイド JavaScript の相違を検討してみるとよいでしょう。クライアントサイド JavaScript は、クライアント環境 (Web ブラウザや Web ページのコンテンツ) を制御します。それと同様に、Flash 固有のクラスは、Flash アプリケーションの外観と動作を実行時に制御します。

コアクラス以外の ActionScript ビルトインクラスは、Flash および Flash Player オブジェクトモデルに固有のクラスです。例としては、Camera、MovieClip、LoadVars の各クラスがあります。他のクラスは、flash.display などのパッケージに整理されています。これらすべてのクラスは、" ビルトインクラス " (アプリケーションに機能を追加するのに使用する事前に定義されたクラス) とも呼ばれます。

次のセクションでは、ActionScript のビルトインクラスについて説明し、ビルトインクラスを使用して実行できる基本的な作業の例を示します。オブジェクト指向プログラミングでのクラスとオブジェクトの概要については、247 ページの「ビルトインクラスの使用について」を参照してください。これらのクラスを使用するコード例については、本マニュアルの全体を参照してください

言語エレメント (定数、演算子、ディレクティブなど) の詳細については、75 ページ、第 4 章の「シンタックスと言語の基礎」を参照してください。

トップレベルクラスとビルトインクラスの詳細については、次のトピックを参照してください。

- 239 ページの「トップレベルクラス」
- 243 ページの「flash.display パッケージ」
- 243 ページの「flash.external パッケージ」
- 244 ページの「flash.filters パッケージ」
- 245 ページの「flash.geom パッケージ」
- 245 ページの「flash.net パッケージ」
- 246 ページの「flash.text パッケージ」
- 246 ページの「mx.lang パッケージ」
- 246 ページの「System パッケージと TextField パッケージ」

その他の言語エレメント

クラスの他にも、ActionScript を構成する言語エレメントがあります。ディレクティブ、定数、グローバル関数、グローバルプロパティ、演算子、ステートメントなどです。これらの各言語エレメントの使用方法については、次のトピックを参照してください。

- 第 4 章の「シンタックスと言語の基礎」
- 第 5 章の「関数とメソッド」

これらの言語エレメントの一覧については、『ActionScript 2.0 リファレンスガイド』の次のセクションを参照してください。

- コンパイラディレクティブ
- 定数
- グローバル関数
- グローバルプロパティ
- 演算子
- ステートメント

トップレベルクラス

トップレベルには、ActionScript のクラスとグローバル関数が含まれ、その多くは、アプリケーションの中心的機能を実現するために使用されます。" コアクラス " は ECMAScript の仕様をそのまま採用したクラスで、Array、Boolean、Date、Error、Function、Math、Number、Object、String、および System があります。各クラスの詳細については、次の表を参照してください。

注	CustomActions クラスと XMLUI クラスは、Flash オーサリング環境内でのみ使用できます。

クラス	説明
Accessibility	Accessibility クラスは、SWF ファイルとスクリーンリーダー間の通信を管理します。このクラスのメソッドを、グローバルな `_accProps` プロパティとともに使用して、ムービークリップ、ボタン、およびテキストフィールドに対してアクセス可能なプロパティを実行時に制御できます。詳細については、「Accessibility」を参照してください。
Array	Array クラスは ActionScript 内における配列を表します。配列オブジェクトはすべて Array クラスのインスタンスです。Array クラスには、配列オブジェクトを操作するためのメソッドとプロパティが含まれています。詳細については、「Array」を参照してください。
AsBroadcaster	他のオブジェクトに追加できるイベント通知機能とリスナー管理機能を提供します。詳細については、「AsBroadcaster」を参照してください。
Boolean	Boolean クラスは、ブール値 (`true` または `false`) 用のラッパーです。詳細については、「Boolean」を参照してください。
Button	Button クラスには、ボタンを操作するためのメソッド、プロパティ、およびイベントハンドラが用意されています。詳細については、「Button」を参照してください。ビルトイン Button クラスは、バージョン 2 コンポーネントである Button に対応する Button コンポーネントクラスとは異なります。
Camera	Camera クラスは、カメラが装備されている場合、カメラへのアクセスを提供します。Flash Media Server と共に使用すると、SWF ファイルで、ユーザーのカメラからのキャプチャ、ブロードキャスト、およびイメージやビデオの録画ができます。詳細については、「Camera」を参照してください。
Color	Color クラスを使用すると、ムービークリップインスタンスの RGB カラー値とカラー変換を設定し、設定した値を取得することができます。Flash Player 8 以降では、Color クラスを使用しないでください。代わりに ColorTransform クラスを使用します。カラー変換の詳細については、「ColorTransform (flash.geom.ColorTransform)」を参照してください。

クラス	説明
ContextMenu	ContextMenu クラスを使用すると、実行時に Flash Player のコンテキストメニューの内容を制御できます。MovieClip、Button、または TextField クラスの `menu` プロパティによって、これらのクラスのオブジェクトに別個の ContextMenu オブジェクトを関連付けることができます。また、ContextMenuItem クラスを使用すると、ContextMenu オブジェクトにカスタムメニューアイテムを追加することができます。詳細については、「ContextMenu」を参照してください。
ContextMenuItem	ContextMenuItem クラスを使用すると、Flash Player コンテキストメニューに表示される新しいメニューアイテムを作成できます。このクラスで作成した新しいメニューアイテムを Flash Player コンテキストメニューに追加するには、ContextMenu クラスを使用します。詳細については、「ContextMenuItem」を参照してください。
CustomActions	CustomActions クラスでは、オーサリングツールに登録されたカスタムアクションを管理できます。詳細については、「CustomActions」を参照してください。
Date	Date クラスは ActionScript の日付と時刻の表示方法を表し、日付と時刻の操作を支援します。Date クラスは、現在の日付と時刻をオペレーティングシステムから取得する役割もになっています。詳細については、「Date」を参照してください。
Error	Error クラスには、スクリプトで発生したランタイムエラーに関する情報が含まれています。通常、エラー条件を生成するには、`throw` ステートメントを使用します。次に、`try..catch..finally` ステートメントでエラー条件を取り扱うことができます。詳細については、「Error」を参照してください。
Function	Function クラスは、ActionScript のすべての関数を表すクラスです。ActionScript にあらかじめ用意されている関数も、ユーザーが定義した関数も、このクラスで表されます。詳細については、「Function」を参照してください。
Key	Key クラスでは、キーボードとキーを押す操作に関する情報を取得するためのメソッドとプロパティが用意されています。詳細については、「Key」を参照してください。
LoadVars	LoadVars クラスを使用すると、SWF ファイルとサーバーとの間で、名前と値のペアである変数を転送することができます。詳細については、「LoadVars」を参照してください。
LocalConnection	LocalConnection クラスを使用すると、`fscommand()` メソッドや JavaScript を使用しなくても、相互に指示を送ることができる SWF ファイルを作成できます。詳細については、「LocalConnection」を参照してください。

クラス	説明
Math	Math クラスを使用すると、一般的な数学定数を簡単に利用できる他、いくつかの一般的な数学関数を使用できます。Math クラスのプロパティとメソッドはすべて静的なもので、`Math.メソッド（パラメータ）`または`Math.定数`というシンタックスで呼び出す必要があります。詳細については、「Math」を参照してください。
Microphone	Microphone クラスは、マイクが装備されている場合、マイクへのアクセスを提供します。Flash Media Server と共に使用すると、SWF ファイルで、ユーザーのマイクからブロードキャストやオーディオの録音ができます。詳細については、「Microphone」を参照してください。
Mouse	Mouse クラスは SWF ファイル内でマウスを制御する手段を提供します。たとえば、マウスポインタの表示と非表示を切り替えることができます。詳細については、「Mouse」を参照してください。
MovieClip	SWF ファイルムービーのムービークリップは、必ず MovieClip クラスのインスタンスになります。ムービークリップオブジェクトを制御するときには、このクラスのメソッドとプロパティを使用します。詳細については、「MovieClip」を参照してください。
MovieClipLoader	このクラスを使用すると、SWF、JPEG、GIF、PNG の各ファイルをムービークリップインスタンスにロードするときに、ステータス情報を提供するリスナーコールバックを実装できます。詳細については、「MovieClipLoader」を参照してください。
NetConnection	NetConnection クラスは、ローカルなストリーミング接続を確立して、HTTP アドレスまたはローカルファイルシステムから FLV (Flash Video) ファイルを再生します。詳細については、「NetConnection」を参照してください。
NetStream	NetStream クラスは、ローカルファイルシステムまたは HTTP アドレスからの FLV ファイルの再生を制御します。詳細については、「NetStream」を参照してください。
Number	Number クラスは、プリミティブな数値データ型を操作するためのラッパーです。詳細については、「Number」を参照してください。
Object	Object クラスは ActionScript クラス階層のルートに存在します。他のすべてのクラスは、Object クラスのメソッドとプロパティを継承します。詳細については、「Object」を参照してください。
PrintJob	PrintJob クラスを使用すると、動的にレンダリングされたコンテンツや複数ページのドキュメントを含む、SWF ファイルのコンテンツを印刷できます。詳細については、「PrintJob」を参照してください。
Selection	Selection クラスを使用すると、ムービー内でカーソルのあるテキストフィールド、つまりフォーカスが置かれているテキストフィールドを設定、制御することができます。詳細については、「Selection」を参照してください。

クラス	説明
SharedObject	SharedObject クラスを使用すると、クライアントコンピュータにローカルの永続データを保存できます。これは、クッキーに似ています。クライアントのコンピュータ上にあるオブジェクト間で、リアルタイムにデータを共有することができます。詳細については、「SharedObject」を参照してください。
Sound	Sound クラスは、SWF ファイルでのサウンドへの制御を提供します。詳細については、「Sound」を参照してください。
Stage	Stage クラスは、SWF ファイルのサイズ、整列、および伸縮モードに関する情報を提供します。ステージのサイズ変更イベントも報告します。詳細については、「Stage」を参照してください。
String	String クラスは、ストリングプリミティブデータ型のラッパーです。このクラスで、String オブジェクトのメソッドとプロパティを使用してプリミティブストリング値の型を操作できます。詳細については、「String」を参照してください。
System	System クラスでは、Flash Player の情報と Flash Player が実行されているシステムの情報 (画面解像度や現在のシステム言語など) が提供されます。また、Flash Player の [設定] パネルの表示と非表示や、SWF ファイルセキュリティ設定の変更ができます。詳細については、「System」を参照してください。
TextField	TextField クラスを使用すると、ダイナミックテキストフィールドおよびテキスト入力フィールドを制御できます。フォーマット情報の取得、イベントハンドラの呼び出し、およびアルファや背景色などのプロパティの変更が可能です。詳細については、「TextField」を参照してください。
TextFormat	TextFormat クラスを使用すると、TextField オブジェクトの文字や段落にフォーマットスタイルを適用できます。詳細については、「TextFormat」を参照してください。
TextSnapshot	TextSnapshot オブジェクトを使用すると、ムービークリップ内の静的テキストにアクセスしてレイアウトすることができます。詳細については、「TextSnapshot」を参照してください。
Video	Video クラスは、SWF ファイルでビデオオブジェクトを表示するのに使用します。このクラスを Flash Media Server で使用して、SWF ファイルのライブストリーミングビデオを表示することや、Flash 内で Flash Video (FLV) ファイルを表示することができます。詳細については、「Video」を参照してください。
XML	このクラスには、XML のオブジェクトを操作するためのメソッドとプロパティが含まれています。詳細については、「XML」を参照してください。
XMLNode	XMLNode クラスは、XML ドキュメントツリーの 1 つのノードを表します。これは XML クラスのスーパークラスです。詳細については、「XMLNode」を参照してください。

クラス	説明
XMLSocket	XMLSocket クラスは、サーバーコンピュータと Flash Player を実行しているクライアントとの間に永続的なソケット接続を確立します。クライアントソケットを使用すると、リアルタイムチャットアプリケーションで必要とされるような待ち時間の少ないデータ転送を行うことができます。詳細については、「XMLSocket」を参照してください。
XMLUI	XMLUI オブジェクトを使用すると、Flash オーサリングツールの拡張機能 (ビヘイビア、コマンド、エフェクト、ツールなど) のカスタムユーザーインターフェイスとして使用される SWF ファイルとやり取りすることができます。詳細については、「XMLUI」を参照してください。

flash.display パッケージ

flash.display パッケージには、ビジュアル表示の構築に使用する BitmapData クラスがあります。

クラス	説明
BitmapData	BitmapData クラスでは、任意のサイズの透明または不透明のビットマップイメージをドキュメント内に作成し、実行時にさまざまな方法で操作できます。詳細については、「BitmapData (flash.display.BitmapData)」を参照してください。

flash.external パッケージ

flash.external パッケージを使用すると、ActionScript コードを使用して Flash Player コンテナとやり取りすることができます。たとえば、SWF ファイルを HTML ページに埋め込んでいる場合、コンテナとはその HTML ページです。HTML ページとやり取りするには、ExternalInterface クラスと JavaScript を使用します。外部 API とも呼ばれます。

クラス	説明
ExternalInterface	ExternalInterface クラスは外部 API であり、Flash Player コンテナ (JavaScript を使用する HTML ページなど) に対する、または Flash Player を使用するデスクトップアプリケーションに対する ActionScript でのやり取りを可能にするサブシステムです。詳細については、「ExternalInterface (flash.external.ExternalInterface)」を参照してください。

flash.filters パッケージ

flash.filters パッケージには、Flash Player 8 で使用できるビットマップフィルタ効果のクラスがあります。フィルタを使用すると、ぼかし、ベベル、グロー、ドロップシャドウなど、豊富な視覚効果を Image インスタンスと MovieClip インスタンスに適用することができます。各クラスの詳細については、次の表で該当するクロスリファレンスを参照してください。

クラス	説明
BevelFilter	BevelFilter クラスは、ベベル効果をムービークリップインスタンスに追加します。詳細については、「BevelFilter (flash.filters.BevelFilter)」を参照してください。
BitmapFilter	BitmapFilter クラスは、すべてのフィルタ効果の基本となるクラスです。詳細については、「BitmapFilter (flash.filters.BitmapFilter)」を参照してください。
BlurFilter	BlurFilter クラスは、ぼかし効果をムービークリップインスタンスに適用します。詳細については、「BlurFilter (flash.filters.BlurFilter)」を参照してください。
ColorMatrixFilter	ColorMatrixFilter クラスは、入力イメージの各ピクセルの ARGB カラー値とアルファ値に 4 x 5 マトリックス変換を適用します。変換を適用した後には、新しい ARGB カラー値とアルファ値のセットを使用して結果を生成することができます。詳細については、「ColorMatrixFilter (flash.filters.ColorMatrixFilter)」を参照してください。
ConvolutionFilter	ConvolutionFilter クラスは、マトリックス畳み込みフィルタ効果を適用します。詳細については、「ConvolutionFilter (flash.filters.ConvolutionFilter)」を参照してください。
DisplacementMapFilter	DisplacementMapFilter クラスは、指定したイメージ (置き換えマップイメージ) のピクセル値を使用して、フィルタの適用先であるオリジナルのインスタンス (ムービークリップ) を空間的に置き換えます。詳細については、「DisplacementMapFilter (flash.filters.DisplacementMapFilter)」を参照してください。
DropShadowFilter	DropShadowFilter クラスは、ドロップシャドウをムービークリップに追加します。詳細については、「DropShadowFilter (flash.filters.DropShadowFilter)」を参照してください。
GlowFilter	GlowFilter クラスは、グロー効果をムービークリップに追加します。詳細については、「GlowFilter (flash.filters.GlowFilter)」を参照してください。
GradientBevelFilter	GradientBevelFilter クラスは、グラデーションベベル効果をムービークリップに適用します。詳細については、「GradientBevelFilter (flash.filters.GradientBevelFilter)」を参照してください。
GradientGlowFilter	GradientGlowFilter クラスは、グラデーショングロー効果をムービークリップに適用します。詳細については、「GradientGlowFilter (flash.filters.GradientGlowFilter)」を参照してください。

flash.geom パッケージ

flash.geom パッケージには、ポイント、矩形、変換マトリックスなどのジオメトリクラスがあります。これらのクラスでは、BitmapData クラスとビットマップキャッシング機能をサポートしています。各クラスの詳細については、次の表で該当するクロスリファレンスを参照してください。

クラス	説明
ColorTransform	ColorTransform クラスを使用すると、インスタンスの RGB カラー値とカラー変換を数学的に設定できます。設定されている値を取得することもできます。詳細については、「ColorTransform (flash.geom.ColorTransform)」を参照してください。
Matrix	Matrix クラスは、座標空間の間でポイントをマッピングする方法を決定する変換マトリックス (変換行列) を表します。詳細については、「Matrix (flash.geom.Matrix)」を参照してください。
Point	Point オブジェクトは 2 次元の座標系の位置を表します。x は水平方向の軸を表し、y は垂直方向の軸を表します。詳細については、「Point (flash.geom.Point)」を参照してください。
Rectangle	Rectangle クラスは、Rectangle オブジェクトの作成および変更に使用されます。詳細については、「Rectangle (flash.geom.Rectangle)」を参照してください。
Transform	オブジェクトインスタンスに適用されるカラー変換と座標操作に関するデータを収集します。詳細については、「Transform (flash.geom.Transform)」を参照してください。

flash.net パッケージ

flash.net パッケージには、ユーザーのコンピュータとサーバーとの間でファイルをアップロードまたはダウンロードするのに使用するクラスがあります。各クラスの詳細については、次の表で該当するクロスリファレンスを参照してください。

クラス	説明
FileReference	FileReference クラスは、ユーザーのコンピュータとサーバーとの間でファイルをアップロードまたはダウンロードするのに使用します。詳細については、「FileReference (flash.net.FileReference)」を参照してください。
FileReferenceList	FileReferenceList クラスは、ユーザーのコンピュータからサーバーにファイルをアップロードするのに使用します。詳細については、「FileReferenceList (flash.net.FileReferenceList)」を参照してください。

flash.text パッケージ

flash.text パッケージには、Flash Player 8 で使用できる高度なアンチエイリアスを操作するための TextRenderer クラスがあります。

クラス	説明
TextRenderer	このクラスは、Flash Player 8 の高度なアンチエイリアス機能を提供します。詳細については、「TextRenderer (flash.text.TextRenderer)」を参照してください。

mx.lang パッケージ

mx.lang パッケージには、多言語テキストを操作するための Locale クラスがあります。

クラス	説明
Locale	このクラスは、SWF ファイル内の多言語テキストの表示方法を制御します。詳細については、「Locale (mx.lang.Locale)」を参照してください。

System パッケージと TextField パッケージ

System パッケージには、Capabilities クラス、IME クラス、および Security クラスがあります。これらは、Flash Player 内のアプリケーションに影響する可能性があるクライアント設定に関するクラスです。各クラスの詳細については、次の表で該当するクロスリファレンスを参照してください。

クラス	説明
Capabilities	Capabilities クラスは、SWF ファイルをホスティングするシステムと Flash Player の機能を判定します。これを使用することで、複数の異なるフォーマットに応じてコンテンツをカスタマイズできます。詳細については、「capabilities (System.capabilities)」を参照してください。
IME	IME クラスを使用すると、クライアントコンピュータで動作する Flash Player アプリケーション上で、オペレーティングシステムの IME を直接操作できます。詳細については、「IME (System.IME)」を参照してください。
security	security クラスには、異なるドメインに属する複数の SWF ファイルが互いにどのようにやり取りするかを指定するためのメソッドがあります。詳細については、「security (System.security)」を参照してください。

TextField パッケージには、CSS スタイルをテキストに適用するのに使用する StyleSheet クラスがあります。

クラス	説明
StyleSheet	StyleSheet クラスを使用すると、テキストのフォーマット規則 (フォントサイズ、色、その他のフォーマットスタイル) を含むスタイルシートオブジェクトを作成できます。詳細については、「StyleSheet (TextField.StyleSheet)」を参照してください。

ビルトインクラスの使用について

オブジェクト指向プログラミング (OOP) では、オブジェクトの分類を " クラス " で定義します。クラスにはオブジェクトの特性と動作が定義されています。これは、設計図に建物の特徴が定義されているのに似ています。オブジェクト指向では、特性をプロパティ (データ) と呼び、動作をメソッド (ビヘイビア) と呼びます。オブジェクト指向プログラミングでのクラスおよびその他の概念については、次のセクションを参照してください。

- 185 ページの「オブジェクト指向プログラミングの基礎」
- 188 ページの「カスタムクラスファイルの記述」

Flash には、コードで使用できる多くのビルトインクラスがあります (237 ページの「トップレベルクラスとビルトインクラスについて」を参照)。これらを利用すると、アプリケーションにインタラクティブ機能を簡単に追加できます。ビルトインクラスによって定義されたプロパティとメソッドを使用するには、通常、まずクラスの " インスタンス " を作成する必要があります。ただし、静的メンバーを持つクラスは例外です。237 ページの「トップレベルクラスとビルトインクラスについて」で説明したように、インスタンスとそのクラスの関係は、家と家の設計図の関係に似ています。

Flash に組み込まれているクラスの使用方法の詳細については、次のトピックを参照してください。

- 248 ページの「ビルトインクラスの新しいインスタンスの作成について」
- 248 ページの「ビルトインオブジェクトプロパティの参照」
- 249 ページの「ビルトインオブジェクトメソッドの呼び出しについて」
- 249 ページの「クラス (静的) メンバーについて」
- 251 ページの「クラスファイルのプリロード」
- 250 ページの「クラスの除外」

ビルトインクラスの新しいインスタンスの作成について

ActionScript クラスのインスタンスを作成するには、new 演算子を使用して、そのクラスのコンストラクタ関数を呼び出します。コンストラクタ関数の名前は、必ずクラスの名前と同じになります。コンストラクタ関数を実行すると、クラスのインスタンスが返されるので、通常はその戻り値を変数に代入します。

たとえば、次のコードでは、新しい Sound オブジェクトを作成しています。

```
var song_sound:Sound = new Sound();
```

場合によっては、インスタンスを作成せずにクラスのプロパティとメソッドを使用できることがあります。詳細については、249 ページの「クラス (静的) メンバーについて」を参照してください。

ビルトインオブジェクトプロパティの参照

オブジェクトのプロパティ値を参照するには、ドット (.) 演算子を使用します。ドットの左側にはオブジェクト名、右側にはプロパティ名を指定します。たとえば、次のステートメントでは、my_obj がオブジェクトで、firstName がプロパティです。

```
my_obj.firstName
```

次のコードは、新しい Array オブジェクトを作成し、そのオブジェクトの length プロパティを表示します。

```
var my_array:Array = new Array("apples", "oranges", "bananas");
trace(my_array.length); // 3
```

また、デバッグ用に配列アクセス演算子を使用する場合などに、配列アクセス演算子 ([]) を使用して、オブジェクトのプロパティにアクセスすることも可能です。次の例では、オブジェクトをループさせて個々のプロパティを表示します。

オブジェクトの内容をループ処理するには :

1. 新しい Flash ドキュメントを作成し、"forin.fla" という名前で保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

   ```
   var results:Object = {firstName:"Tommy", lastName:"G", age:7, avg:0.336,
     b:"R", t:"L"};
   for (var i:String in results) {
       trace("the value of [" + i + "] is: " + results[i]);
   }
   ```

   上記のコードは、results という名前の新しいオブジェクトを定義し、firstName、lastName、age、avg、b、および t の値を定義します。for..in ループは、results オブジェクトの各プロパティの値を [出力] パネルにトレース出力します。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

ドット演算子や配列アクセス演算子などの演算子の詳細については、135 ページの「演算子について」を参照してください。メソッドとプロパティの詳細については、157 ページ、第 5 章の「関数とメソッド」を参照してください。ビルトイン MovieClip クラスのプロパティを操作する例については、301 ページ、第 10 章の「ムービークリップの操作」を参照してください。TextField、String、TextRenderer、および TextFormat の各クラスのプロパティを操作する例については、331 ページ、第 11 章の「テキストとストリングの操作」を参照してください。

ビルトインオブジェクトメソッドの呼び出しについて

オブジェクトのメソッドを呼び出すには、ドット (.) 演算子の後に目的のメソッドを指定します。次のコードでは、まず新しい Sound オブジェクトを作成し、次にそのオブジェクトの `setVolume()` メソッドを呼び出しています。

```
var my_sound:Sound = new Sound(this);
my_sound.setVolume(50);
```

ビルトイン MovieClip クラスのメソッドを操作する例については、301 ページ、第 10 章の「ムービークリップの操作」を参照してください。TextField、String、TextRenderer、および TextFormat の各ビルトインクラスのメソッドを操作する例については、331 ページ、第 11 章の「テキストとストリングの操作」を参照してください。

クラス (静的) メンバーについて

いくつかのビルトイン ActionScript クラスには、クラスメンバー (静的メンバー) と呼ばれるメンバーがあります。クラスメンバー (プロパティとメソッド) にアクセスするときは、クラスのインスタンスではなく、クラス名を呼び出します。つまり、静的なプロパティとメソッドを使用するときには、クラスのインスタンスを作成する必要がありません。

たとえば、Math クラスの場合、プロパティはすべて静的です。次のコードでは、Math クラスの `max()` メソッドを呼び出して、2 つの数値の大小を判定しています。

```
var largerNumber:Number = Math.max(10, 20);
trace(largerNumber); // 20
```

Math クラスの静的メソッドの詳細と使用例については、『ActionScript 2.0 リファレンスガイド』の「Math」を参照してください。

クラスの除外

場合によっては、SWF ファイルのサイズを減らすためにコンパイルからクラスを除外することが必要で、しかも、型チェックを目的としたアクセスおよび使用は除外したクラスに対しても行う必要があることがあります。たとえば、複数の SWF ファイルまたは共有ライブラリを使用する (特に、それらが多くの同じクラスにアクセスする) ようなアプリケーションを開発する場合などが該当します。クラスを除外すると、複数のファイル内に同じクラスが重複して含まれるのを防ぐことができます。

クラスの除外の詳細については、次のトピックを参照してください。

- 251 ページの「クラスファイルのプリロード」

コンパイル時にクラスを除外するには :

1. 新しい XML ファイルを作成します。
2. この XML ファイルに "‹FLA_ ファイル名›_exclude.xml" という名前を付けます (‹FLA_ ファイル名 › は、FLA ファイルの名前から拡張子を除いた部分)。

   たとえば、FLA ファイル名が "sellStocks.fla" の場合、XML ファイルの名前は "sellStocks_exclude.xml" になります。
3. FLA ファイルと同じディレクトリにファイルを保存します。
4. この XML ファイル内に次のタグを指定します。

```
<excludeAssets>
  <asset name="className1" />
  <asset name="className2" />
</excludeAssets>
```

   <asset> タグの name 属性には、SWF ファイルから除外するクラス名を指定します。アプリケーションで必要な数だけ追加します。たとえば、次の XML ファイルを使用すると、mx.core.UIObject クラスと mx.screens.Slide クラスが SWF ファイルから除外されます。

```
<excludeAssets>
  <asset name="mx.core.UIObject" />
  <asset name="mx.screens.Slide" />
</excludeAssets>
```

クラスのプリロードの詳細については、251 ページの「クラスファイルのプリロード」を参照してください。

クラスファイルのプリロード

このセクションでは、Flash でクラスをプリロードする方法と書き出す方法について説明します。これには、Component Architecture バージョン 2 のコンポーネントで使用されるクラスも含まれます。" プリロード " では、SWF ファイル用のデータの一部が、ユーザーが操作を開始する前にロードされます。外部クラスを使用する場合、SWF ファイルの最初のフレームにクラスが読み込まれ、そのデータが SWF ファイルにロードされる最初のエレメントとなります。これはコンポーネントのクラスの場合と似ています。コンポーネントのフレームワークも、SWF ファイルの最初のフレームでロードを実行します。大きなアプリケーションでは、データを読み込むときロードにかなり時間がかかるので、次に示す手順のように、このデータをうまく扱う必要があります。

クラスは最初にロードされるデータなので、プログレスバーよりも前にクラスがロードされ、思うようにプログレスバーの作成やアニメーションのロードができない可能性があります。プログレスバーには、すべてのデータ (クラスを含む) のロードの進行状況を表示したいことが普通と考えられるからです。そのためには、クラスをロードするタイミングを、SWF ファイルの他の部分をロードするよりは後、ただしコンポーネントを使用するよりは前にする必要があります。

次の手順は、クラスをロードする SWF ファイルのフレームを変更する方法を示しています。

SWF ファイルにロードされるクラスのために異なるフレームを選択するには :

1. [ファイル]-[パブリッシュ設定] を選択します。
2. [Flash] タブを選択し、[設定] ボタンをクリックします。
3. [クラス用のフレームの書き出し] テキストボックスに、クラスをロードするタイミングを決定する新しいフレームの番号を入力します。
4. [OK] をクリックします。

クラスのロードを指定したフレームに再生ヘッドが到達するまでは、どのクラスも使用できません。たとえば、バージョン 2 コンポーネントが機能するためにはクラスが必要なので、コンポーネントは ActionScript 2.0 クラス用の書き出しフレームより後でロードする必要があります。フレーム 3 に書き出した場合は、再生ヘッドがフレーム 3 に達してデータがロードされるまで、どのクラスのどのような機能も使用できません。

バージョン 2 コンポーネントのクラスなど、クラスを使用するファイルをプリロードする場合は、SWF ファイル内のコンポーネントをプリロードしなければなりません。これを実現するには、コンポーネントが SWF ファイル内の異なるフレームに書き出されるように設定する必要があります。デフォルトでは、UI コンポーネントは SWF ファイルのフレーム 1 に書き出されます。したがって、コンポーネントの [リンケージ] ダイアログボックスで、[最初のフレームに書き出し] を必ずオフにしてください。

メモ	ActionScript を使用してコンポーネントをステージに追加する場合は、追加するコンポーネントのインスタンスをペーストボード (ステージ周辺のエリア) にドラッグする必要があります。これにより、アプリケーション内でコンポーネントを使用すること、およびそのコンポーネントが未使用ライブラリアイテムではないことが Flash に伝わります。Flash では、未使用ライブラリアイテムは SWF ファイルに追加されない点に注意してください。

コンポーネントを最初のフレームにロードしなければ、SWF ファイルの最初のフレームのカスタムプログレスバーを作成することができます。[クラス用のフレームの書き出し] テキストボックスで指定したフレームのクラスをロードするまでは、ActionScript でコンポーネントを参照することも、ステージに何かコンポーネントを含めることもできません。

注意	コンポーネントは、そのコンポーネントで使用する ActionScript クラスよりも後に書き出す必要があります。

第 7 章

継承

7

第 6 章の「クラス」で、クラスファイルの作成方法、およびクラスを利用して外部ファイル内にコードを整理する方法について説明しました。また、クラスファイルを関連するパッケージ内に整理する方法についても説明しました。この章では、既存のクラスの機能を拡張する、より高度なクラスの作成方法について説明します。新しいメソッドやプロパティを追加できるように独自のカスタムクラスや既存のクラスを拡張する場合があるので、このトピックは役に立ちます。

継承の詳細については、253 ページの「継承について」を参照してください。メソッドとプロパティの詳細については、157 ページ、第 5 章の「関数とメソッド」を参照してください。

継承の詳細については、次のトピックを参照してください。

継承について 253
Flash でのサブクラスの作成について 255
アプリケーションでのポリモーフィズムの使用 260

継承について

第 6 章の「クラス」では、独自のカスタムデータ型を作成するためにクラスファイルを作成する方法について説明しました。カスタムクラスファイルの作成方法を学習することで、タイムラインから外部ファイルにコードを移動する方法を説明しました。外部ファイル内にコードを移動すると、コードを簡単に編集できるようになります。独自のカスタムクラスの基本的な作成方法を学習したので、次に、" サブクラス " または " 拡張クラス " と呼ばれる OOP (Object-Oriented Programming : オブジェクト指向プログラミング) 手法について学習します。この手法を使用して、既存のクラスを基に新しいクラスを作成することができます。

OOP の利点の 1 つは、クラスの " サブクラス " を作成できることです。サブクラスは、" スーパークラス " のすべてのプロパティとメソッドを継承します。たとえば、MovieClip クラスを拡張するかそのサブクラスを作成する場合、MovieClip クラスを拡張するカスタムクラスを作成します。サブクラスは、MovieClip クラスのすべてのプロパティとメソッドを継承します。または、カスタムスーパークラスを拡張する一連のクラスを作成することができます。たとえば、Lettuce クラスは、Vegetable スーパークラスから拡張できます。

サブクラスは、一般的に、アプリケーションで使用できる追加のメソッドとプロパティを定義するので、スーパークラスを " 拡張 " することになります。サブクラスは、スーパークラスから継承したメソッドをオーバーライドする、つまり継承したメソッドに対して独自の定義を用意することもできます。スーパークラスから継承したメソッドをサブクラスでオーバーライドすると、サブクラス内でスーパークラスの定義にアクセスすることはできなくなります。唯一の例外として、サブクラスのコンストラクタ関数内では、`super` ステートメントを使用してスーパークラスのコンストラクタを呼び出すことができます。オーバーライドの詳細については、258 ページの「メソッドとプロパティのオーバーライド」を参照してください。

たとえば、哺乳類を表す Mammal クラスを作成し、すべての哺乳類に共通するプロパティとビヘイビアを定義したとします。この Mammal クラスを拡張して、Cat サブクラスを作成することができます。サブクラスを使用することによりコードの再利用が可能になります。両方のクラスに共通するコードをすべて作り直す代わりに、既存のクラスを拡張するだけで済みます。Cat クラスを拡張して、さらに別のサブクラス (Siamese クラスなど) を作成することもできます。複雑なアプリケーションでは、クラス階層の構築方法を決定することが、設計プロセスの大半を占めます。

継承とサブクラスを使用すると、同じ機能を持つ一連の関連クラスを作成できるので、大きなアプリケーションで非常に役に立ちます。たとえば、社内の一般的な従業員のメソッドとプロパティを定義する Employee クラスを作成することができます。次に、Employee クラスを拡張しそのすべてのメソッドとプロパティを継承する Contractor というクラスを作成することができます。Contractor クラスは、独自のメソッドとプロパティを追加することも、Employee スーパークラスで定義されたメソッドとプロパティをオーバーライドすることもできます。さらに、Employee クラスを拡張し、`hire()`、`fire()`、`raise()`、`promote()` などの追加のメソッドやプロパティを定義する Manager という新しいクラスを作成することができます。Manager などのサブクラスを拡張して、Director という新しいクラスを作成することもできます。このクラスでも新しいメソッドの追加や既存のメソッドのオーバーライドを行うことができます。

既存のクラスを拡張するたびに、新しいクラスはサブクラスの現在のすべてのメソッドとプロパティを継承します。各クラスが関連付けられていない場合、それらのクラスの機能が同じである場合でも、各個別のクラスファイル内に各メソッドとプロパティを再作成する必要があります。類似のロジックを複数のファイルで変更する場合、コーディングだけでなくアプリケーションのデバッグとプロジェクトの維持のために長い時間がかかります。

ActionScript では、`extends` キーワードを使用することで、クラスとそのスーパークラスの間に継承関係を確立し、インターフェイスを拡張できます。`extends` キーワードの使用方法の詳細については、255 ページの「Flash でのサブクラスの作成について」および255 ページの「サブクラスの作成について」を参照してください。`extends` キーワードの詳細については、『ActionScript 2.0 リファレンスガイド』の「extends ステートメント」を参照してください。

Flash でのサブクラスの作成について

オブジェクト指向プログラミングでは、サブクラスは、スーパークラスと呼ばれる別のクラスのプロパティとメソッドを継承できます。コアクラスおよび Flash Player ActionScript クラスの多くと同じように、独自のカスタムクラスを拡張することができます。TextField クラスは拡張できません。

2 つのクラス間にこのような関係を作るには、class ステートメントの `extends` 節を使用します。スーパークラスを指定するには、次のシンタックスを使用します。

```
class SubClass extends SuperClass {}
```

SubClass に指定したクラスは、SuperClass によって定義されているプロパティとメソッドをすべて継承します。

たとえば、哺乳類を表す Mammal クラスを作成し、すべての哺乳類に共通するプロパティとメソッドを定義したとします。この Mammal クラスのバリエーションとして、たとえば Marsupial (有袋類) クラスを作成するには、Mammal クラスを拡張します。つまり、次のように Mammal クラスのサブクラスを作成します。

```
class Marsupial extends Mammal {}
```

サブクラスはスーパークラスのすべてのプロパティとメソッドを継承します。その中には、`private` キーワードを使用してプライベートであると宣言しているプロパティやメソッドも含まれます。

クラスの拡張の詳細については、次のトピックを参照してください。

- 255 ページの「サブクラスの作成について」
- 258 ページの「メソッドとプロパティのオーバーライド」

プライベートメンバーの詳細については、199 ページの「パブリック、プライベート、および静的なメソッドとプロパティ (メンバー) について」を参照してください。サブクラスの作成の例については、256 ページの「例 : Widget クラスの拡張」を参照してください。

サブクラスの作成について

次のコードでは、Sound クラスを拡張するカスタムクラス JukeBox を定義します。このクラスでは、`song_arr` という配列を定義する他に、歌を演奏し、Sound クラスから継承した `loadSound()` メソッドを呼び出す `playSong()` というメソッドを定義しています。

```
class JukeBox extends Sound {
  public var song_arr:Array = new Array("beethoven.mp3", "bach.mp3",
   "mozart.mp3");
  public function playSong(songID:Number):Void {
    super.loadSound(song_arr[songID], true);
  }
}
```

サブクラスのコンストラクタ関数で super( ) を呼び出さない場合は、関数の最初のステートメントとして、すぐ上のスーパークラスのコンストラクタへのパラメータなしの呼び出しが、コンパイラによって自動的に生成されます。スーパークラスにコンストラクタが存在しない場合、コンパイラは、サブクラスから呼び出すための空のコンストラクタ関数を作成します。ただし、スーパークラスのコンストラクタがパラメータを受け取るように定義されている場合は、サブクラス内にコンストラクタを作成し、必須パラメータを指定してスーパークラスのコンストラクタを呼び出す必要があります。

ActionScript 2.0 では、多重継承、つまり複数のクラスから継承することは認められていません。ただし次の例のように、extends ステートメントを個別に使用することにより、複数のクラスを効率よく継承できます。

```
// 使用不可
class C extends A, B {} // ** エラー : クラスは複数のクラスを拡張できない。

// 使用可
class B extends A {}
class C extends B {}
```

また、インターフェイスを使用すると、" 多重継承 " を制限付きですが実現できます。インターフェイスの詳細については、265 ページ、第 8 章の「インターフェイス」を参照してください。サブクラスの作成の例については、256 ページの「例 : Widget クラスの拡張」を参照してください。super の詳細については、『ActionScript 2.0 リファレンスガイド』の「super ステートメント」を参照してください。

例 : Widget クラスの拡張

クラスメンバーは、そのメンバーを定義しているスーパークラスのサブクラスに反映されます。次の例では、Widget クラスを作成する方法を示します。SubWidget というクラスを記述することによってこのクラスを拡張 (サブクラス作成) します。

Widget クラスと SubWidget サブクラスを作成するには :

1. 新しい ActionScript ファイルを作成し、Widget.as という名前で保存します。
2. 新しいドキュメントに次のコードを追加します。

```
class Widget {
   public static var widgetCount:Number = 0;
   public function Widget() {
      Widget.widgetCount++;
   }
}
```

3. ActionScript ファイルの変更を保存します。
4. 新しい ActionScript ファイルを作成し、SubWidget.as という名前で Widget クラスと同じディレクトリに保存します。

5. "SubWidget.as" で、スクリプトウィンドウに次のコードを入力します。

```
class SubWidget extends Widget {
  public function SubWidget() {
    trace("Creating subwidget #" + Widget.widgetCount);
  }
}
```

6. "SubWidget.as" の変更を保存します。

7. 新しい FLA ファイルを作成し、subWidgetTest.fla という名前を付けて、前の ActionScript クラスファイルと同じディレクトリに保存します。

8. "subWidgetTest.fla" ファイルで、メインタイムラインのフレーム 1 に次のコードを入力します。

```
var sw1:SubWidget = new SubWidget();
var sw2:SubWidget = new SubWidget();
trace("Widget.widgetCount = " + Widget.widgetCount);
trace("SubWidget.widgetCount = " + SubWidget.widgetCount);
```

上記のコードでは、SubWidget クラスから sw1 と sw2 という 2 つのインスタンスを作成しています。`SubWidget` コンストラクタの各呼び出しでは、静的 `Widget.widgetCount` プロパティの現在の値がトレース出力されます。SubWidget クラスは Widget クラスのサブクラスなので、SubWidget クラスを通して `widgetCount` プロパティにアクセスできます。また、コンパイラは、この参照を `Widget.widgetCount` として (ActionScript ファイルではなくバイトコード内で) 上書きします。Widget クラスまたは SubWidget クラスのインスタンス (sw1、sw2 など) から静的 `widgetCount` プロパティにアクセスしようとすると、コンパイルエラーが発生します。

9. ドキュメントの変更を保存します。

10. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

[出力] パネルに次の結果が表示されます。

```
Creating subwidget #1
Creating subwidget #2
Widget.widgetCount = 2
SubWidget.widgetCount = 2
```

Widget クラスのコンストラクタは明示的に呼び出されませんが、SubWidget クラスのコンストラクタにより呼び出されるので、この出力が表示されます。これにより、Widget クラスのコンストラクタが、Widget クラスの静的 `widgetCount` 変数をインクリメントします。

ActionScript 2.0 コンパイラは静的メンバーの参照をクラス定義内で解決できます。

Widget.widgetCount プロパティで、クラス名を指定する代わりに widgetCount とだけ指定した場合、ActionScript 2.0 コンパイラは、それが Widget.widgetCount を参照していることを解決し、該当するプロパティを正しく出力します。同様に、このプロパティを SubWidget.widgetCount のように指定した場合、SubWidget は Widget クラスのサブクラスであるため、コンパイラは、この参照を Widget.widgetCount として (AS ファイルではなくバイトコード内で) オーバーライドします。

注意	sw1 インスタンスや sw2 インスタンスを使用して Widget クラスの静的 widgetCount 変数にアクセスしようとすると、静的メンバーには直接クラスを通してのみアクセスできるというエラーメッセージが表示されます。

コードを読みやすくするため、上の例のようにコード内では常に静的メンバー変数への明示的な参照を使用することをお勧めします。明示的な参照を使用すると、静的メンバーの定義がどこに存在するのかを容易に確認できるようになります。

メソッドとプロパティのオーバーライド

サブクラスがスーパークラスを拡張する場合、サブクラスはスーパークラスのすべてのメソッドとプロパティを継承します。クラスを使用し、既存のクラスを拡張することの利点の 1 つとして、既存のクラスに新しい機能を提供できるだけでなく、既存の機能を変更することもできます。たとえば、256 ページの「例 : Widget クラスの拡張」で作成した Widget クラスについて考えます。スーパークラス (Widget) で新しいメソッドを作成し、そのメソッドをサブクラス (SubWidget) でオーバーライドすることも、Widget クラスから継承したメソッドをそのまま使用することもできます。次の例では、クラス内の既存のメソッドをオーバーライドする方法を示します。

サブクラスでメソッドをオーバーライドするには :

1. 新しい ActionScript ドキュメントを作成し、Widget.as という名前で保存します。
2. "Widget.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

   メモ : 前の例で Widget クラスを作成した場合は、次のように doSomething() メソッドを追加することによって既存のメソッドを変更します。

```
class Widget {
   public static var widgetCount:Number = 0;
   public function Widget() {
      Widget.widgetCount++;
   }
   public function doSomething():Void {
      trace("Widget::doSomething()");
   }
}
```

3. ActionScript ドキュメントの変更を保存します。

   Widget クラスは、コンストラクタと doSomething() というパブリックメソッドを定義するようになりました。

4. SubWidget.as という新しい ActionScript ファイルを作成し、"Widget.as" と同じディレクトリに保存します。

> メモ
> 256 ページの「例 : Widget クラスの拡張」で SubWidget クラスを作成した場合は、代わりにこのファイルを使用できます。

5. "SubWidget.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
class SubWidget extends Widget {
  public function SubWidget() {
    trace("Creating subwidget # " + Widget.widgetCount);
    doSomething();
  }
}
```

6. "SubWidget.as" の変更を保存します。

   SubWidget クラスのコンストラクタは、スーパークラスで定義した `doSomething()` メソッドを呼び出します。

7. 新しい Flash ドキュメントを作成し、subWidgetTest.fla という名前で ActionScript ドキュメントと同じディレクトリに保存します。

8. "subWidgetTest.fla" で、メインタイムラインのフレーム 1 に次の ActionScript を入力します。

```
var sw1:SubWidget = new SubWidget();
var sw2:SubWidget = new SubWidget();
```

9. Flash ドキュメントの変更を保存します。

10. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。[出力] パネルに次の結果が表示されます。

```
Creating subwidget # 1
Widget::doSomething()
Creating subwidget # 2
Widget::doSomething()
```

   この出力は、SubWidget クラスのコンストラクタがスーパークラス (Widget) のコンストラクタを呼び出し、これにより静的 `widgetCount` プロパティがインクリメントされたことを示しています。SubWidget のコンストラクタは、スーパークラスの静的プロパティをトレースし、スーパークラスから継承する `doSomething()` メソッドを呼び出します。

11. SubWidget クラスを開き、`doSomething()` という新しいメソッドを追加します。次のコードと一致するようにクラスを変更します (ボールド体のコードを追加します)。

```
class SubWidget extends Widget {
  public function SubWidget() {
    trace("Creating subwidget # " + Widget.widgetCount);
    doSomething();
  }
  public function doSomething():Void {
    trace("SubWidget::doSomething()");
  }
}
```

12. クラスファイルの変更を保存し、"subwidgetTest.fla" をもう一度開きます。

13. [制御]-[ムービープレビュー] を選択してファイルをテストします。[出力] パネルに次の結果が表示されます。

```
Creating subwidget # 1
SubWidget::doSomething()
Creating subwidget # 2
SubWidget::doSomething()
```

前の出力は、SubWidget クラスのコンストラクタ内の doSomething() メソッドが、スーパークラスではなく現在のクラス内の doSomething() メソッドを呼び出していることを示しています。

SubWidget クラスをもう一度開き、スーパークラスの doSomething() メソッドを呼び出すように SubWidget クラスのコンストラクタを変更します (ボールド体のコードを追加します)。

```
public function SubWidget() {
   trace("Creating subwidget # " + Widget.widgetCount);
   super.doSomething();
}
```

前に説明したように、super キーワードを追加して、現在のクラスの doSomething() メソッドの代わりにスーパークラスの doSomething() メソッドを呼び出すことができます。super の詳細については、『ActionScript 2.0 リファレンスガイド』の「super」の項目を参照してください。

14. 変更したコンストラクタと共に SubWidget クラスファイルを保存し、[制御]-[ムービープレビュー] を選択して、Flash ドキュメントを再パブリッシュします。

[出力] パネルに、Widget クラスの doSomething() メソッドの内容が表示されます。

アプリケーションでのポリモーフィズムの使用

オブジェクト指向プログラミングでは、" ポリモーフィズム " と呼ばれる手法を使用して、個々のクラス間の違いを表現することができます。具体的には、クラスが、対応するスーパークラスのメソッドをオーバーライドして、これらのメソッドの特殊な実装を定義することになります。

たとえば、play() メソッドと sleep() メソッドを持つ Mammal というクラスを定義したとします。次に、Mammal クラスを継承する Cat、Monkey、および Dog というサブクラスを作成します。これらのサブクラスは、個々の動物に固有の習性を反映させるために、Mammal クラスの play() メソッドをオーバーライドできます。たとえば、Monkey は木にぶら下がる play() メソッドを、Cat は毛糸の玉を追いかける play() メソッドを、Dog はボールを取りに行く play() メソッドをそれぞれ実装するといったことが可能です。sleep() の動作はこれらの動物にほぼ共通しているため、スーパークラスの実装を利用することができます。次の手順では、Flash でのこの例について説明します。

アプリケーションでポリモーフィズムを使用するには :

1. 新しい ActionScript ドキュメントを作成し、Mammal.as という名前で保存します。

   このドキュメントは、この後の手順で作成するいくつかの動物クラスの基本クラスです。

2. "Mammal.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
class Mammal {
  private var _gender:String;
  private var _name:String = "Mammal";

  // コンストラクタ
  public function Mammal(gender:String) {
    this._gender = gender;
  }

  public function toString():String {
    return "[object " + speciesName + "]";
  }
  public function play():String {
    return "Chase another of my kind.";
  }
  public function sleep():String {
    return "Close eyes.";
  }

  public function get gender():String {
    return this._gender;
  }
  public function get speciesName():String {
    return this._name;
  }
  public function set speciesName(value:String):Void {
    this._name = value;
  }
}
```

   前のクラスは、_gender と _name という 2 つのプライベート変数を定義します。これらは、動物の性別と哺乳類タイプを格納するために使用されます。次に、Mammal コンストラクタが定義されます。このコンストラクタは、1 つのパラメータ gender を取り、このパラメータを使用して、前に定義したプライベート変数 _gender を設定します。さらに、toString()、play()、sleep() という 3 つのパブリックメソッドも指定されます。これらはそれぞれストリングオブジェクトを返します。最後の 3 つのメソッドは、哺乳類の _gender プロパティと _name プロパティの getter メソッドと setter メソッドです。

3. ActionScript ドキュメントを保存します。

   このクラスは、この後に作成する Cat クラス、Dog クラス、Monkey クラスのスーパークラスとして機能します。Mammal クラスの toString() メソッドを使用して、Mammal インスタンス (または Mammal クラスを拡張する任意のインスタンス) のストリング表現を表示することができます。

4. 新しい ActionScript ファイルを作成し、Cat.as という名前で、手順 1 で作成した "Mammal.as" クラスファイルと同じディレクトリに保存します。

5. "Cat.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
class Cat extends Mammal {
  // コンストラクタ
  public function Cat(gender:String) {
    super(gender);
    speciesName = "Cat";
  }

  public function play():String {
    return "Pounce a ball of yarn.";
  }
}
```

Mammal スーパークラスの play() メソッドをオーバーライドします。Cat クラスは、コンストラクタと play() メソッドという 2 つのメソッドのみを定義します。Cat クラスは Mammal クラスを拡張するので、Mammal クラスのメソッドとプロパティは、Cat クラスによって継承されます。オーバーライドの詳細については、258 ページの「メソッドとプロパティのオーバーライド」を参照してください。

6. ActionScript ドキュメントの変更を保存します。

7. 新しい ActionScript ドキュメントを作成し、Dog.as という名前で 前の 2 つのクラスファイルと同じディレクトリに保存します。

8. "Dog.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
class Dog extends Mammal {
  // コンストラクタ
  public function Dog(gender:String) {
    super(gender);
    speciesName = "Dog";
  }

  public function play():String {
    return "Fetch a stick.";
  }
}
```

Dog クラスは、Cat クラスと構造が非常によく似ていますが、いくつかの値が変更されています。Dog クラスも Mammal クラスを拡張するので、すべてのメソッドとプロパティを継承します。Dog コンストラクタは 1 つのプロパティ gender を取り、Dog クラスの親クラスである Mammal に渡します。speciesName 変数もオーバーライドされ、Dog というストリングに設定されます。play() メソッドも親クラスからオーバーライドされます。

9. ActionScript ドキュメントの変更を保存します。

10. 他のファイルと同じディレクトリ内に別の ActionScript ドキュメントを作成し、Monkey.as という名前で保存します。

11. "Monkey.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
class Monkey extends Mammal {
  // コンストラクタ
  public function Monkey(gender:String) {
    super(gender);
    speciesName = "Monkey";
  }

  public function play():String {
    return "Swing from a tree.";
  }
}
```

前の 2 つのクラス (Cat と Dog) と同じように、Monkey クラスは Mammal クラスを拡張します。Monkey クラスのコンストラクタは、Mammal クラスのコンストラクタを呼び出し、Mammal クラスのコンストラクタに gender を渡し、`speciesName` を `Monkey` というストリングに設定します。Monkey クラスも、`play()` メソッドの動作をオーバーライドします。

12. ActionScript ドキュメントの変更を保存します。

13. これで Mammal クラスの 3 つのサブクラスを作成しました。次に mammalTest.fla という新しい Flash ドキュメントを作成します。

14. "mammalTest.fla" ドキュメントで、メインタイムラインのフレーム 1 に次の ActionScript コードを入力します。

```
var mammals_arr:Array = new Array();
this.createTextField("info_txt", 10, 10, 10, 450, 80);
info_txt.html = true;
info_txt.multiline = true;
info_txt.border = true;
info_txt.wordWrap = true;

createMammals()
createReport()

function createMammals():Void {
  mammals_arr.push(new Dog("Female"));
  mammals_arr.push(new Cat("Male"));
  mammals_arr.push(new Monkey("Female"));
  mammals_arr.push(new Mammal("Male"));
}
```

```
function createReport():Void {
   var i:Number;
   var len:Number = mammals_arr.length;
   // タブストップを使用して、Mammal の情報を 4 列の HTML テキストに表示する。
   info_txt.htmlText = "<textformat tabstops='[110, 200, 300]'>";
   info_txt.htmlText += "<b>Mammal\tGender\tSleep\tPlay</b>";
   for (i = 0; i < len; i++) {
      info_txt.htmlText += "<p>" + mammals_arr[i].speciesName
           + "\t" + mammals_arr[i].gender
           + "\t" + mammals_arr[i].sleep()
           + "\t" + mammals_arr[i].play() + "</p>";
      // trace ステートメントで Mammal.toString() メソッドを呼び出す。
      trace(mammals_arr[i]);
   }
   info_txt.htmlText += "</textformat>";
}
```

"mammalTest.fla" のコードは、前のクラスよりも少々複雑です。最初に、3 つの動物クラスを呼び出します。

15. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して、ドキュメントをテストします。

    Mammal の情報が、ステージ上のテキストフィールドに表示され、次のテキストが [出力] パネルに表示されます。

```
[object Dog]
[object Cat]
[object Monkey]
[object Mammal]
```

第 8 章

インターフェイス

8

オブジェクト指向プログラミング (OOP) では、インターフェイスはドキュメントであり、このドキュメントを使用して、クラス内に存在する必要があるメソッドを宣言することができます (定義はできません)。開発者チームの一員として作業する場合または Flash で大きなアプリケーションを作成する場合、開発時にはインターフェイスが非常に役に立ちます。インターフェイスを使用すると、開発者は、ActionScript クラス内の基本メソッドを簡単に識別することができます。これらのメソッドは、開発者が各インターフェイスを使用するときに実装する必要があります。

この章では、いくつかのサンプルのインターフェイスについて説明します。この章の終わりには、独自のインターフェイスファイルを構築できるようになります。クラスの構築方法について詳しくない場合は、この章のチュートリアルと例を実行する前に、第 6 章の「クラス」をお読みください。

インターフェイスの操作の詳細については、次のトピックを参照してください。

インターフェイスについて 265
データ型としてのインターフェイスの作成 270
継承とインターフェイスについて 272
例 : インターフェイスの使用 273
例 : 複雑なインターフェイスの作成 275

インターフェイスについて

オブジェクト指向プログラミングでは、インターフェイスは、メソッドが実装 (定義) されないクラスのようなものです。つまり、このままでは何も実行しません。そのため、インターフェイスは " 空の " メソッドから成ります。インターフェイスによって宣言されるメソッドは、別のクラス内で実装できます。ActionScript でインターフェイスとオブジェクトに違いが生じるのは、コンパイル時のエラーチェックと言語ルールの施行に関する部分だけです。

インターフェイスはクラスではありませんが、実行時の ActionScript ではそうとも言い切れません。これは、インターフェイスが抽象化されているためです。ActionScript のインターフェイスは型キャスト (既存のデータ型から別のデータ型への変更) を有効にするために実行時も存在します。ActionScript 2.0 のオブジェクトモデルは、多重継承をサポートしていません。したがって、クラスは常に 1 つの親クラスから継承できます。この親クラスとなるのは、コアクラスまたは Flash Player クラス、またはユーザー定義の (カスタム) クラスです。インターフェイスを使用して、多重継承を制限付きですが実現できます。これにより、1 つのクラスが他の複数のクラスを継承できます。

たとえば C++ では、Cat クラスが Mammal クラスを拡張しつつ、`chaseTail` (しっぽを追いかける) メソッドと `eatCatNip` (キャットミントを食べる) メソッドを備えた Playful クラスを拡張することができます。ActionScript 2.0 では、Java と同様に、1 つのクラスが複数のクラスを直接的に拡張することはできませんが、単一のクラスを拡張して、複数のインターフェイスを実装することはできます。したがって、`chaseTail()` メソッドと `eatCatNip()` メソッドを宣言する Playful インターフェイスを作成することができます。そこで、Cat クラスや他の任意のクラスがこのインターフェイスを実装し、これらのメソッドに定義を提供することができます。

インターフェイスは、関係のないクラスの間に関係を持たせるための " プログラミング契約 " と考えることもできます。たとえば、あなたがプログラマチームで働いていて、チームの各メンバーは同じアプリケーションのそれぞれ異なるクラスを担当しているとします。チームでは、アプリケーションの設計の際に、さまざまなクラスが通信するために使用する一連のメソッドについて合意しています。これらのメソッドとそのパラメータ、戻り値タイプを宣言するインターフェイスを作成しました。この場合、このインターフェイスを実装するクラスは、これらのメソッドの定義を提供する必要があります。そうしないとコンパイルエラーが起こります。インターフェイスは、すべてのクラスが従う必要がある通信プロトコルに似ています。

これを実現する 1 つの方法は、必要なメソッドをすべて定義したクラスを作成し、各クラスにこのスーパークラスを拡張 (継承) させることです。しかし、このアプリケーションは互いに関連性のないクラスから構成されているので、すべてのクラスを共通のクラス階層に入れることには無理があります。より良い方法は、クラス間の通信に使われるメソッド群を宣言するインターフェイスを作成し、各クラスにこれらのメソッドを実装させる (独自に定義させる) ことです。

通常、インターフェイスを使用しなくてもプログラムは問題なく記述できます。しかし、インターフェイスを適切に使用すれば、より洗練された、拡張性と保守性の高いアプリケーションを設計することができます。

ActionScript のインターフェイスは型キャストを有効にするために実行時も存在します。詳細については、71 ページ、第 3 章の「オブジェクトのキャストについて」を参照してください。インターフェイスはオブジェクトでもクラスでもありませんが、ワークフローはクラスを操作する場合と似ています。クラスのワークフローの詳細については、188 ページの「カスタムクラスファイルの記述」を参照してください。インターフェイスを使用したアプリケーション作成のチュートリアルについては、273 ページの「例 : インターフェイスの使用」を参照してください。

インターフェイスの使用の詳細については、次に示す各セクションを参照してください。

- 267 ページの「interface キーワードについて」
- 267 ページの「インターフェイスの命名について」
- 267 ページの「インターフェイスの定義と実装」

interface キーワードについて

interface キーワードは、インターフェイスを定義します。インターフェイスはクラスに似ていますが、次に示す重要な違いがあります。

- インターフェイスにはメソッドの宣言だけが含まれます。メソッド実装は含まれません。つまり、インターフェイスを実装するすべてのクラスは、インターフェイスで宣言されている各メソッドの実装を定義する必要があります。
- インターフェイス定義では、パブリックメンバーしか使用できず、静的なクラスメンバーは使用できません。
- get ステートメントおよび set ステートメントはインターフェイス定義では使用できません。
- interface キーワードを使用するには、FLA ファイルの [パブリッシュ設定] ダイアログボックスの [Flash] タブで、ActionScript 2.0、および Flash Player 6 以降を指定する必要があります。

interface キーワードは、[アクション] パネルに記述するスクリプトではありません。外部スクリプトファイルで使用する場合にのみサポートされます。

インターフェイスの命名について

インターフェイス名は、クラス名と同じように大文字で始まります。インターフェイス名は、通常、Printable のように形容詞を表します。次のインターフェイス名 IEmployeeRecords は、先頭が大文字で始まり、大文字と小文字を混在させて単語を連結した名前になっています。

```
interface IEmployeeRecords {}
```

メモ	クラス名と区別するために、インターフェイス名の先頭に大文字の "I" を付ける開発者もいます。インターフェイスと通常のクラスを簡単に区別できるので、この方法は効果的です。

命名規則の詳細については、645 ページ、第 17 章の「ActionScript 2.0 のベストプラクティスとコーディング規則」を参照してください。

インターフェイスの定義と実装

インターフェイスの作成プロセスは、クラスの作成プロセスと同じです。クラスと同様に、インターフェイスは外部 ActionScript ファイル内でのみ定義できます。インターフェイスの作成作業では、少なくとも次の手順を実行します。

- 外部 ActionScript ファイル内でインターフェイスを定義します。
- このインターフェイスファイルを所定のクラスパスディレクトリ (Flash がクラスを検索する場所) またはアプリケーションの FLA ファイルと同じディレクトリに保存します。
- 別のスクリプトである Flash (FLA) ドキュメントまたは外部スクリプトファイルのいずれかに、クラスのインスタンスを作成するか、オリジナルのインターフェイスを基にしてサブインターフェイスを作成します。
- インターフェイスを実装するクラスを外部スクリプトファイル内に作成します。

インターフェイスを宣言するには、interface キーワードを使用し、その後にインターフェイス名と左右の中カッコ ({ }) を続けます。この中カッコ内にインターフェイスの本体を定義します。次に例を示します。

```
interface IEmployeeRecords {
  // インターフェイスのメソッド宣言
}
```

インターフェイスには、メソッド (関数) 宣言だけを含めます (パラメータ、パラメータの型、関数の戻り値タイプも含む)。

クラスとインターフェイスの構成規則の詳細については、645 ページ、第 17 章の「ActionScript 2.0 のベストプラクティスとコーディング規則」を参照してください。インターフェイスを使用したアプリケーション作成のチュートリアルについては、273 ページの「例 : インターフェイスの使用」を参照してください。

たとえば、次のコードでは、method1() と method2() という 2 つのメソッドを含む IMyInterface という名前のインターフェイスを宣言しています。1 つ目のメソッド method1() はパラメータを取らず、戻り値タイプとして Void (戻り値なし) が指定されています。2 番目のメソッド method2() は String 型のパラメータを 1 つ取り、戻り値タイプとして Boolean を指定しています。

単純なインターフェイスを作成するには :

1. 新しい ActionScript ファイルを作成し、IMyInterface.as という名前で保存します。
2. スクリプトウィンドウに次の ActionScript コードを入力します。

   ```
   interface IMyInterface {
     public function method1():Void;
     public function method2(param:String):Boolean;
   }
   ```

3. ActionScript ファイルの変更を保存します。

   アプリケーション内のインターフェイスを使用するには、まず新しいインターフェイスを実装するクラスを作成する必要があります。

4. 新しい ActionScript ファイルを作成し、MyClass.as という名前で "IMyInterface.as" と同じディレクトリに保存します。
5. "MyClass.as" クラスファイルで、スクリプトウィンドウに次の ActionScript コードを入力します。

   ```
   class MyClass {
   }
   ```

   カスタムクラス (MyClass) に対してインターフェイス (IMyInterface) を使用するように指示するために、implements キーワードを使用する必要があります。これにより、実装するインターフェイスで宣言されているすべてのメソッドをクラスで定義する必要があることを指定します。

6. 次のコードに一致するように、"MyClass.as" の ActionScript コードを変更します (ボールド体のコードを追加します)。

```
class MyClass implements IMyInterface {
}
```

implements キーワードは、クラス名の後に置きます。

7. [シンタックスチェック] ボタンをクリックします。

[出力] パネルに、MyClass がメソッド ‹X› をインターフェイス IMyInterface から実装する必要があることを示すエラーが表示されます。このエラーメッセージが表示されるのは、インターフェイスを拡張するすべてのクラスが、インターフェイスのドキュメントに記載された各メソッドを定義する必要があるためです。

8. MyClass ドキュメントをもう一度変更し (ボールド体のコードを追加します)、次のコードのように、method1() メソッドと method2() メソッドの ActionScript コードを記述します。

```
class MyClass implements IMyInterface {
  public function method1():Void {
    // ...
  };
  public function method2(param:String):Boolean {
    // ...
    return true;
  }
}
```

9. "MyClass.as" ドキュメントを保存し、[シンタックスチェック] をクリックします。

2 つのメソッドを定義したので、[出力] パネルに、エラーメッセージも警告も表示されなくなります。

作成するクラスファイルは、インターフェイスファイルで定義するパブリックメソッドに制限されません。インターフェイスファイルは、実装する必要がある最小限のメソッド、およびそれらのメソッドのプロパティと戻り値タイプの概要を示すだけです。特定のインターフェイスを実装するクラスは、ほとんどの場合、追加のメソッド、変数、および getter メソッドと setter メソッドを含んでいます。

インターフェイスファイルに変数の宣言や代入を含めることはできません。また、インターフェイス内で宣言する関数には中カッコを記述できません。たとえば、次のインターフェイスはコンパイルできません。

```
interface IBadInterface {
  // コンパイルエラー。インターフェイス内では変数宣言はできない。
  public var illegalVar:String;

  // コンパイルエラー。インターフェイス内では関数の本体を記述できない。
  public function illegalMethod():Void {
  }

  // コンパイルエラー。プライベートメンバーはインターフェイスでは許可されない。
  private function illegalPrivateMethod():Void;
```

```
  // コンパイルエラー。getter/setter はインターフェイスでは許可されない。
  public function get illegalGetter():String;
}
```

複雑なインターフェイスの作成方法を示すチュートリアルについては、273 ページの「例 : インターフェイスの使用」を参照してください。

インターフェイスに名前を付け、パッケージに格納するための規則は、クラスの場合と同じです。217 ページの「クラスファイルの命名について」を参照してください。

データ型としてのインターフェイスの作成

クラスと同様に、インターフェイスは新しいデータ型を定義します。インターフェイスを実装するクラスが、インターフェイスによって定義される型になるようにすることができます。これは、特定のオブジェクトが特定のインターフェイスを実装しているかどうかを判断するときに役に立ちます。たとえば、次の例で作成するインターフェイス IMovable について考えます。

データ型としてインターフェイスを作成するには :

1. 新しい ActionScript ドキュメントを作成し、IMovable.as という名前でハードディスクに保存します。
2. "IMovable.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
interface IMovable {
  public function moveUp():Void;
  public function moveDown():Void;
}
```

3. ActionScript ファイルの変更を保存します。
4. 新しい ActionScript ドキュメントを作成し、Box.as という名前で "IMovable.as" と同じディレクトリに保存します。

   このドキュメント内で、前の手順で作成した IMovable インターフェイスを実装する Box クラスを作成します。
5. "Box.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
class Box implements IMovable {
  public var xPos:Number;
  public var yPos:Number;

  public function Box() {
  }

  public function moveUp():Void {
    trace("moving up");
    // メソッド定義
  }
  public function moveDown():Void {
```

```
        trace("moving down");
        // メソッド定義
    }
}
```

6. ActionScript ドキュメントの変更を保存します。
7. 新しい Flash ドキュメントを作成し、boxTest.fla という名前を付けて、前の 2 つの ActionScript ドキュメントと同じディレクトリに保存します。
8. タイムラインのフレーム 1 を選択し、ActionScript エディタを開いて、次の ActionScript コードを [アクション] パネル (またはスクリプトウィンドウ) に入力します。

   ```
   var newBox:Box = new Box();
   ```

   この ActionScript コードは、Box クラスのインスタンスを作成します。このインスタンスを Box 型の変数として宣言します。
9. Flash ドキュメントの変更を保存し、[制御]-[ムービープレビュー] を選択して、SWF ファイルをテストします。

   Flash Player 7 以降では、実行時に式をインターフェイス型または他のデータ型にキャストできます。Java のインターフェイスとは異なり、ActionScript のインターフェイスは実行時に存続するため、型のキャストが可能です。その式がインターフェイスを実装するオブジェクトである場合、またはインターフェイスを実装するスーパークラスを持つオブジェクトである場合は、そのオブジェクトが返されます。そうでない場合は、`null` が返されます。これは、特定のオブジェクトが特定のインターフェイスを実装することを確実にしたい場合に便利です。型キャストの詳細については、71 ページ、第 3 章の「オブジェクトのキャストについて」を参照してください。
10. 次のコードを、"boxTest.fla" の ActionScript コードの末尾に追加します。

    ```
    if (IMovable(newBox) != null) {
      newBox.moveUp();
    } else {
      trace("box instance is not movable");
    }
    ```

    この ActionScript コードは、オブジェクトに対して `moveUp()` メソッドを呼び出す前に、`newBox` インスタンスによって IMovable インターフェイスが実装されているかどうかを確認します。
11. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して、SWF ファイルをテストします。

    Box インスタンスは IMovable インターフェイスを実装するので、`Box.moveUp()` メソッドが呼び出され、"moving up" というテキストが [出力] パネルに表示されます。

キャストの詳細については、71 ページ、第 3 章の「オブジェクトのキャストについて」を参照してください。

継承とインターフェイスについて

また、extends キーワードを使用して、インターフェイスのサブクラスを作成することもできます。これは、既存のインターフェイスを拡張 (サブクラス作成) し、追加のメソッドを作成する必要がある大きなプロジェクトで非常に役に立ちます。これらのメソッドは、そのインターフェイスを実装するすべてのクラスで定義される必要があります。

インターフェイスを拡張する場合は、複数のインターフェイスファイルが同じ名前の関数を宣言していながらそれらのパラメータまたは戻り値タイプが異なっていると、エラーメッセージが表示されることに注意する必要があります。

次の例では、extends キーワードを使用してインターフェイスのサブクラスを作成する方法を示します。

インターフェイスを拡張するには :

1. 新しい ActionScript ファイルを作成し、Ia.as という名前で保存します。
2. "Ia.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
interface Ia {
   public function f1():Void;
   public function f2():Void;
}
```

3. ActionScript ファイルの変更を保存します。
4. 新しい ActionScript ファイルを作成し、Ib.as という名前を付けて、手順 1 で作成した "Ia.as" ファイルと同じフォルダに保存します。
5. "Ib.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
interface Ib extends Ia {
   public function f8():Void;
   public function f9():Void;
}
```

6. ActionScript ファイルの変更を保存します。
7. 新しい ActionScript ファイルを作成し、ClassA.as という名前を付けて、前の 2 つのファイルと同じディレクトリに保存します。
8. "ClassA.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
class ClassA implements Ib {
   // f1() および f2() はインターフェイス Ia で定義される。
   public function f1():Void {
   }
   public function f2():Void {
   }

   // f8() および f9() は、Ia を拡張するインターフェイス Ib で定義される。
   public function f8():Void {
   }
   public function f9():Void {
   }
}
```

9. クラスファイルを保存し、スクリプトウィンドウの上にある [シンタックスチェック] ボタンをクリックします。

   4 つのメソッドがすべて定義され、それぞれのインターフェイスファイルの定義と一致している限り、エラーメッセージは生成されません。

メモ	ActionScript 2.0 では、クラスが拡張できるのは 1 つのクラスのみですが、必要な数のインターフェイスを実装できます。

前の例で、ClassA クラスで複数のインターフェイスを実装したい場合は、単純にインターフェイスをカンマで区切ります。または、スーパークラスを拡張して、複数のインターフェイスを実装したクラスがある場合は、次のようなコードを使用します。

```
class ClassA extends ClassB implements Ib, Ic, Id {...}.
```

例 : インターフェイスの使用

この例では、多くの異なるクラスで再利用できる単純なインターフェイスを作成します。

インターフェイスを構築するには :

1. 新しい ActionScript ファイルを作成し、IDocumentation.as という名前で保存します。
2. "IDocumentation.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
interface IDocumentation {
  public function downloadUpdates():Void;
  public function checkForUpdates():Boolean;
  public function searchHelp(keyword:String):Array;
}
```

3. ActionScript インターフェイスファイルに加えた変更を保存します。
4. 新しい ActionScript ファイルを "IDocumentation.as" ファイルと同じディレクトリに作成し、このファイルを FlashPaper.as. という名前で保存します。
5. "FlashPaper.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
class FlashPaper implements IDocumentation {
}
```

6. ActionScript ファイルに加えた変更を保存します。
7. ActionScript クラスの [シンタックスチェック] ボタンをクリックします。

   次のようなエラーメッセージが表示されます。

```
**Error** path\FlashPaper.as: Line 1: The class must implement method
  'checkForUpdates' from interface 'IDocumentation'.

     class FlashPaper implements IDocumentation {

Total ActionScript Errors: 1  Reported Errors: 1
```

   このエラーが表示されるのは、現在の FlashPaper クラスが、IDocumentation インターフェイスで定義したどのパブリックメソッドも定義していないためです。

8. "FlashPaper.as" クラスファイルをもう一度開き、次のコードと一致するように既存の ActionScript コードを変更します。

```
class FlashPaper implements IDocumentation {
   private static var __version:String = "1,2,3,4";
   public function downloadUpdates():Void {
   };
   public function checkForUpdates():Boolean {
      return true;
   };
   public function searchHelp(keyword:String):Array {
      return []
   };
}
```

9. ActionScript ファイルの変更を保存し、[シンタックスチェック] ボタンをもう一度クリックします。

今回は、[出力] パネルにエラーは表示されません。

メモ	必要な数の追加の静的変数、パブリック変数、プライベート変数、静的メソッド、パブリックメソッド、プライベートメソッドを FlashPaper クラスファイルに追加することができます。インターフェイスファイルは、そのインターフェイスを実装するすべてのクラス内に必要な最小限のメソッドのセットのみを定義します。

10. IDocumentation インターフェイスドキュメントをもう一度開き、次のボールド体のコード行 (`searchHelp()` メソッドの下) を追加します。

```
interface IDocumentation {
   public function downloadUpdates():Void;
   public function checkForUpdates():Boolean;
   public function searchHelp(keyword:String):Array;
   public function addComment(username:String, comment:String):Void;
}
```

11. インターフェイスファイルの変更を保存し、"FlashPaper.as" ドキュメントをもう一度開きます。

12. [シンタックスチェック] ボタン をクリックすると、[出力] パネルに新しいエラーメッセージが表示されます。

```
**Error** path\FlashPaper.as: Line 1: The class must implement method
   'addComment' from interface 'IDocumentation'.

      class FlashPaper implements IDocumentation {

Total ActionScript Errors: 1  Reported Errors: 1
```

インターフェイスファイルで定義したすべてのクラスが、"FlashPaper.as" クラスファイルで定義されなくなったので、前のエラーが表示されます。このエラーを修正するには、`addComment()` メソッドを FlashPaper クラスに追加するか、メソッドの定義を IDocumentation インターフェイスファイルから削除します。

13. 次のメソッドを FlashPaper クラスに追加します。

```
public function addComment(username:String, comment:String):Void {
   /* サーバーサイドページにパラメータを送信し、データベースにコメントを挿入する。*/
}
```

14. "FlashPaper.as" の変更を保存し、[シンタックスチェック] ボタンをクリックすると、エラーは表示されなくなります。

前のセクションでは、IDocumentation インターフェイスファイルを基にクラスを作成しました。このセクションでは、同じように IDocumentation インターフェイスを実装しながら、メソッドとプロパティを追加する新しいクラスを作成します。

このチュートリアルでは、インターフェイスを使用することの利点を示します。つまり、IDocumentation インターフェイスを拡張する別のクラスを作成する場合、新しいクラス内で必要なメソッドを簡単に識別できます。

例 : 複雑なインターフェイスの作成

次の例は、インターフェイスを定義および実装するためのいくつかの方法を示しています。このチュートリアルでは、単純なインターフェイスの作成方法、複数のインターフェイスを実装するクラスの作成方法、およびインターフェイスで他のインターフェイスを拡張してより複雑なデータ構造を作成する方法について説明します。

複雑なインターフェイスを作成するには :

1. 新しい ActionScript ドキュメントを作成し、InterfaceA.as という名前で保存します。
2. complexInterface という名前の新しいフォルダを作成し、"InterfaceA.as" をこのフォルダに保存します。

   このチュートリアルで作成するすべてのファイルをこのディレクトリに保存します。
3. "Interface.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
// ファイル名 : InterfaceA.as
interface InterfaceA {
  public function k():Number;
  public function n(z:Number):Number;
}
```

4. ActionScript ドキュメントを保存し、ClassB.as という名前の新しい ActionScript ドキュメントを作成して、"complexInterface" ディレクトリに保存します。

   "ClassB.as" は、前に作成した InterfaceA インターフェイスを実装します。
5. "ClassB.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
// ファイル名 : ClassB.as
class ClassB implements InterfaceA {
  public function k():Number {
    return 25;
  }
  public function n(z:Number):Number {
    return (z + 5);
  }
}
```

6. "ClassB.as" ドキュメントの変更を保存し、新しい Flash ドキュメントを作成して、classbTest.fla という名前で "complexInterface" ディレクトリに保存します。

   このクラスファイルは、前に作成した ClassB クラスをテストします。

7. "classbTest.fla" で、タイムラインのフレーム 1 に次の ActionScript コードを入力します。

```
// ファイル名 : classbTest.fla
import ClassB;
var myB:ClassB = new ClassB();
trace(myB.k()); // 25
trace(myB.n(7)); // 12
```

8. Flash ドキュメントの変更を保存し、[制御]-[ムービープレビュー] を選択して、Flash ドキュメントをテストします。

   [出力] パネルに、25 と 12 という 2 つの数字が表示されます。これは、ClassB クラスの `k()` メソッドと `n()` メソッドの結果です。

9. 新しい ActionScript ファイルを作成し、ClassC.as という名前で "complexInterface" ディレクトリに保存します。

   このクラスファイルは、手順 1 で作成した InterfaceA インターフェイスを実装します。

10. "ClassC.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
// ファイル名 : ClassC.as
class ClassC implements InterfaceA {
  public function k():Number {
    return 25;
  }
  // ** エラー ** クラスは、インターフェイス 'InterfaceA' のメソッド 'n' も実装する必要
  がある。
}
```

   "ClassC.as" クラスファイルの [シンタックスチェック] ボタンをクリックすると、[出力] パネルに、InterfaceA インターフェイスで定義された `n()` メソッドを現在のクラスに実装する必要があることを示すエラーメッセージが表示されます。インターフェイスを実装するクラスを作成する場合は、インターフェイス内の各エントリのメソッドを定義することが重要です。

11. 新しい ActionScript ドキュメントを作成し、InterfaceB.as という名前で "complexInterface" ディレクトリに保存します。

12. "InterfaceB.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
// ファイル名 : InterfaceB.as
interface InterfaceB {
  public function o():Void;
}
```

13. "InterfaceB.as" ドキュメントの変更を保存し、新しい ActionScript ドキュメントを作成して、ClassD.as という名前で "complexInterface" ディレクトリに保存します。

   このクラスは、前の手順で作成した InterfaceA インターフェイスと InterfaceB インターフェイスの両方を実装します。ClassD クラスは、各インターフェイスファイルで定義された各メソッドの実装を含んでいる必要があります。

14. "ClassD.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
// ファイル名 : ClassD.as
class ClassD implements InterfaceA, InterfaceB {
  public function k():Number {
    return 15;
  }
  public function n(z:Number):Number {
    return (z * z);
  }
  public function o():Void {
    trace("o");
  }
}
```

15. "ClassD.as" ファイルの変更を保存し、新しい Flash ドキュメントを作成して、classdTest.fla という名前で保存します。

    この Flash ドキュメントは、前に作成した ClassD クラスをテストします。

16. "classdTest.fla" で、タイムラインのフレーム 1 に次の ActionScript コードを追加します。

```
// ファイル名 : classdTest.fla
import ClassD;
var myD:ClassD = new ClassD();
trace(myD.k()); // 15
trace(myD.n(7)); // 49
myD.o(); // o
```

17. "classdTest.fla" ファイルの変更を保存し、[制御]-[ムービープレビュー] を選択してファイルをテストします。

    値 15、49、および文字 o が、[出力] パネルに表示されます。これらの値は、それぞれ `ClassD.k()` メソッド、`ClassD.n()` メソッド、および `ClassD.o()` メソッドの結果です。

18. 新しい ActionScript ドキュメントを作成し、InterfaceC.as という名前で保存します。

    このインターフェイスは、前に作成した InterfaceA インターフェイスを拡張し、さらに新しいメソッドの定義を追加します。

19. "InterfaceC.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
// ファイル名 : InterfaceC.as
interface InterfaceC extends InterfaceA {
  public function p():Void;
}
```

20. ActionScript ファイルの変更を保存し、新しい ActionScript ファイルを作成して、ClassE.as という名前で "complexInterface" ディレクトリに保存します。

    このクラスは、InterfaceB と InterfaceC という 2 つのインターフェイスを実装します。

21. "ClassE.as" で、スクリプトウィンドウに次の ActionScript コードを入力します。

```
// ファイル名 : ClassE.as
class ClassE implements InterfaceB, InterfaceC {
  public function k():Number {
    return 15;
  }
  public function n(z:Number):Number {
    return (z + 5);
  }
  public function o():Void {
    trace("o");
  }
  public function p():Void {
    trace("p");
  }
}
```

22. ActionScript ドキュメントの変更を保存し、新しい Flash ドキュメントを作成して、classeTest.fla という名前で "complexInterface" ディレクトリに保存します。

23. "classeTest.fla" で、タイムラインのフレーム 1 に次の ActionScript コードを入力します。

```
// ファイル名 : classeTest.fla
import ClassE;
var myE:ClassE = new ClassE();
trace(myE.k()); // 15
trace(myE.n(7)); // 12
myE.o(); // o
myE.p(); // p
```

24. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して、SWF ファイルをテストします。

15、12、o、p という値が、[出力] パネルに表示されます。これらの値は、`ClassE.k()` メソッド、`ClassE.n()` メソッド、`ClassE.o()` メソッド、および `ClassE.p()` メソッドによって返される値です。ClassE クラスは、InterfaceB インターフェイスと InterfaceC インターフェイスの両方を実装したので、2 つのインターフェイスファイルの各メソッドが定義されている必要があります。InterfaceB インターフェイスと InterfaceC インターフェイスは、`o()` メソッドと `p()` メソッドのみを定義しますが、InterfaceC は InterfaceA を拡張します。つまり、このインターフェイスで定義されたすべてのメソッド (`k()` と `n()`) も実装されている必要があります。

第 9 章

イベントの処理

9

イベントとは、SWF ファイルの再生中に発生する動作です。マウスのクリックやキーを押す操作などのイベントは、ユーザーからの直接の操作によって発生するので、ユーザーイベントと呼ばれます。一方、ムービークリップがステージに最初に読み込まれたときなど、Flash Player によって自動的に生成されるイベントは、ユーザーからの直接の操作によって発生するものではないので、システムイベントと呼ばれます。

アプリケーションがイベントに反応するためには、イベントハンドラを使用する必要があります。イベントハンドラとは、特定のオブジェクトおよびイベントに関連付けられた ActionScript コードのことです。たとえば、ユーザーがステージ上のボタンをクリックしたときに再生ヘッドを次のフレームに進めたり、ネットワークから XML ファイルのロードが終わったときにその内容をテキストフィールドに表示するなどの処理を実装できます。

ActionScript のイベントは、次のようにいくつかの方法で処理できます。

- 280 ページの「ActionScript とイベントについて」
- 284 ページの「イベントリスナーの使用」
- 289 ページの「ボタンおよびムービークリップのイベントハンドラの使用」、とりわけ on ハンドラおよび onClipEvent ハンドラを使用する。
- 293 ページの「コンポーネントインスタンスからのイベントのブロードキャスト」

イベントハンドラと loadMovie (MovieClip.loadMovie method) を併用すると、予期しない結果が生じる場合があります。`on()` を使用してイベントハンドラをボタンに割り当てるか、onPress (MovieClip.onPress ハンドラ) などのイベントハンドラメソッドを使用してダイナミックハンドラを作成してから `loadMovie()` を呼び出す場合、新しいコンテンツがロードされた後にイベントハンドラが使用できなくなります。ただし、onClipEvent ハンドラまたは on ハンドラを使用してイベントハンドラをムービークリップに割り当ててから、そのムービークリップの `loadMovie()` を呼び出すと、新しいコンテンツがロードされた後でもイベントハンドラを利用できます。

イベント処理の詳細については、次に示す各セクションを参照してください。

ActionScript とイベントについて 280
イベントリスナーの使用 284
コンポーネントでのイベントリスナーの使用 287
ボタンおよびムービークリップのイベントハンドラの使用. 289
コンポーネントインスタンスからのイベントのブロードキャスト 293
ボタン状態を持つムービークリップの作成 293
イベントハンドラのスコープ 295
this キーワードのスコープ 298
Delegate クラスの使用 298

ActionScript とイベントについて

Flash では、イベントが発生したときに ActionScript コードが実行されます。たとえば、ムービークリップがロードされたときやタイムラインのキーフレームが入力されたとき、またはユーザーがボタンをクリックしたときなどです。イベントは、ユーザーまたはシステムによってトリガされます。ユーザーの場合は、マウスボタンをクリックしたときや、キーを押したときにイベントがトリガされます。システムの場合は、特定の条件が満たされたときや、何らかのプロセスが完了したとき (SWF ファイルがロードされたとき、タイムラインが特定のフレームに達したとき、グラフィックがダウンロードを完了したときなど) にイベントがトリガされます。

イベントが発生したときには、" イベントハンドラ " を記述することにより、何らかのアクションでイベントに応答します。いつ、どこでイベントが発生するのかを理解しておくと、どこでどのようにアクションで応答すればよいか、また、どのような場合にどの ActionScript ツールを使用するかを判断するために役立ちます。

イベントは多くのカテゴリに分類できます。これには、ユーザーがマウスやキーボードを使用して Flash アプリケーションを操作したときに発生するマウスイベントとキーボードイベント、ムービークリップ内で発生するクリップイベント、タイムラインのフレーム内で発生するフレームイベントなどが含まれます。

マウスおよびキーボードのイベント

ユーザーが SWF ファイルまたはアプリケーションを操作すると、マウスおよびキーボードのイベントがトリガされます。たとえば、ユーザーがポインタをボタン上に移動すると、`Button.onRollOver` イベントまたは `on(rollOver)` イベントが発生します。ボタンをクリックすると `Button.onRelease` イベントが、キーボード上のキーを押すと `on(keyPress)` イベントが発生します。フレームでコードを記述するか、スクリプトをインスタンスに割り当てることで、これらのイベントを処理し、必要なインタラクティブ機能をすべて追加することができます。

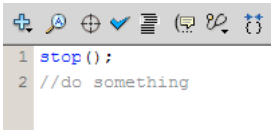
クリップのイベント

ユーザーがシーンを開始または終了したとき、またはマウスかキーボードでシーンを操作したときにトリガされる多くのクリップイベントに、ムービークリップ内で応答することができます。たとえば、シーンが開始されたときに外部 SWF ファイルまたは JPG イメージをムービークリップにロードしたり、マウスの動きによりシーン内のエレメントの位置を変更したりすることができます。

フレームのイベント

メインまたはムービークリップのタイムラインで、再生ヘッドがキーフレームに達するとシステムイベントが発生します。これが、" フレームイベント " です。フレームイベントは、(タイムラインに沿って進む) 時間の経過に基づいてアクションをトリガするとき、または現在ステージに表示されているエレメントを操作するときに使用できます。スクリプトをキーフレームに追加すると、再生中にキーフレームに達したときにスクリプトが実行されます。フレームに割り当てられたスクリプトは、" フレームスクリプト " と呼ばれます。

あるキーフレームに達したときに再生を停止するために使用するのが、フレームスクリプトの最も一般的な用途です。これは、`stop()` 関数で行います。キーフレームを選択し、`stop()` 関数をスクリプトエレメントとして [アクション] パネルに追加します。



あるキーフレームで SWF ファイルを停止している場合には、何らかのアクションを取る必要があります。たとえば、フレームスクリプトを使用して、テキストフィールドの値を動的に更新することや、ステージ上のエレメントの操作を管理することができます。

イベントハンドラメソッドの使用

イベントハンドラメソッドとは、クラスのインスタンスでイベントが発生したときに呼び出されるクラスメソッドのことです。たとえば、MovieClip クラスには `onPress` イベントハンドラが定義されています。これは、ムービークリップオブジェクト上でマウスがクリックされたときに必ず呼び出されるイベントハンドラです。ただし、イベントハンドラメソッドは、クラスの他のメソッドと違い、コードで直接呼び出すことはありません。該当するイベントが発生したときに、Flash Player から自動的に呼び出されます。

イベントハンドラを定義している ActionScript クラスの例としては、Button、ContextMenu、ContextMenuItem、Key、LoadVars、LocalConnection、Mouse、MovieClip、MovieClipLoader、Selection、SharedObject、Sound、Stage、TextField、XML、XMLSocket といったクラスを挙げることができます。クラスのイベントハンドラの詳細については、『ActionScript 2.0 リファレンスガイド』の各クラス項目を参照してください。各イベントハンドラのタイトルには " ハンドラ " という語が追加されています。

デフォルトでは、イベントハンドラメソッドは未定義です。つまり、特定のイベントが発生したときに、それに対応するイベントハンドラは呼び出されるものの、そこから先の処理はアプリケーションに実装されていません。イベントに反応する処理をアプリケーションに実行させるには、function ステートメントを使って関数を定義し、それを該当するイベントハンドラに割り当てる必要があります。すると、イベントが発生したときには、そのイベントハンドラに割り当てた関数が自動的に呼び出されるようになります。

イベントハンドラは、イベントを適用するオブジェクト、オブジェクトのイベントハンドラメソッドの種類、イベントハンドラに割り当てる関数の 3 つの部分で構成されます。次の例は、イベントハンドラの基本的な構造を表したものです。

```
object.eventMethod = function () {
  // イベントに反応するコードをここに記述する
}
```

たとえば、next_btn という名前のボタンがステージ上にあるとします。次のコードは、このボタンの onPress イベントハンドラに関数を割り当てます。この関数は現在のタイムライン内の次のフレームに再生ヘッドを進めます。

```
next_btn.onPress = function () {
  nextFrame();
}
```

関数参照の割り当て　前のコードでは、nextFrame() 関数が onPress イベントハンドラに直接割り当てられています。イベントハンドラメソッドに関数参照 (関数名) を割り当て、その関数自体は後で定義するという方法もあります。例を示します。

```
// ボタンの onPress イベントハンドラに関数参照を割り当てる
next_btn.onPress = goNextFrame;

// goNextFrame() 関数を定義する
function goNextFrame() {
  nextFrame();
}
```

次の例で注意が必要なのは、onPress イベントハンドラに割り当てているのは、関数の戻り値ではなく関数参照であるという点です。

```
// これは誤り
next_btn.onPress = goNextFrame();
// こちらが正解
next_btn.onPress = goNextFrame;
```

渡されたパラメータの受け取り　イベントハンドラの中には、発生したイベントに関する情報を通知するパラメータを受け取るものがあります。その一例が TextField.onSetFocus イベントハンドラです。これは、テキストフィールドのインスタンスにキーボードのフォーカスが移ったときに発生するイベントです。このイベントハンドラは、その前にキーボードフォーカスを持っていたテキストフィールドオブジェクトへの参照をパラメータで受け取ります。

たとえば、次のコードは、キーボードフォーカスがなくなったテキストフィールドにテキストを挿入します。

```
this.createTextField("my_txt", 99, 10, 10, 200, 20);
my_txt.border = true;
my_txt.type = "input";
this.createTextField("myOther_txt", 100, 10, 50, 200, 20);
myOther_txt.border = true;
myOther_txt.type = "input";
myOther_txt.onSetFocus = function(my_txt:TextField) {
  my_txt.text = "I just lost keyboard focus";
};
```

ランタイムオブジェクトのイベントハンドラ　実行時に自分で作成するオブジェクトのイベントハンドラに関数を割り当てることもできます。たとえば、次のコードは、新しいムービークリップインスタンス (newclip_mc) を作成して、その onPress イベントハンドラに関数を割り当てています。

```
this.attachMovie("symbolID", "newclip_mc", 10);
newclip_mc.onPress = function () {
  trace("You pressed me");
}
```

詳細については、309 ページの「実行時のムービークリップの作成」を参照してください。

イベントハンドラメソッドの上書き　ActionScript クラスを拡張するクラスを作成すると、イベントハンドラメソッドを記述した関数で上書きすることができます。新しいサブクラスにイベントハンドラを定義し、拡張したクラスのライブラリのシンボルを新しいサブクラスにリンクさせて各種オブジェクトを再利用することができます。次のコードでは、MovieClip クラスの onPress イベントハンドラを無効にして、ムービークリップの透明度を下げる関数にします。

```
// FadeAlpha クラス -- ムービークリップをクリックしたときに透明度を設定する。
class FadeAlpha extends MovieClip {
  function onPress() {
    this._alpha -= 10;
  }
}
```

ActionScript クラスの拡張とライブラリのシンボルへのリンク手順については、230 ページの「Flash でのシンボルへのクラスの割り当て」の例を参照してください。カスタムクラスの記述および使用の詳細については、第 6 章の「クラス」を参照してください。

イベントリスナーの使用

イベントリスナーとは、あるオブジェクトでブロードキャストしたイベントを別のオブジェクトで受け取れるようにするというものです。イベントが発生する側のオブジェクトをブロードキャスターオブジェクト、イベントを受け取る側のオブジェクトをリスナーオブジェクトと呼びます。ブロードキャスターオブジェクトは、ブロードキャスターが生成したイベントを受け取るリスナーオブジェクトを登録します。たとえば、ステージで発生した `onResize` の通知をムービークリップオブジェクトが受け取るように登録したり、テキストフィールドオブジェクトからの `onChanged` の通知をボタンインスタンスが受け取ることができます。また、1つのブロードキャスターで発生したイベントを複数のリスナーオブジェクトが受け取るように登録したり、複数のブロードキャスターで発生したイベントを1つのリスナーオブジェクトが受け取るように登録することも可能です。

イベントハンドラメソッドと違って、イベント用のリスナー / ブロードキャスターモデルでは、コンフリクトを発生させずに複数のコードで同一イベントをリッスンできます。`XML.onLoad()` など、リスナーモデルまたはブロードキャスターモデルを使用しないイベントモデルでは、複数のコードが同一イベントをリッスンすると問題が発生する可能性があります。異なるコードが1つの `XML.onLoad` コールバック関数参照を制御しようとしてコンフリクトが発生するためです。リスナーモデルまたはブロードキャスターモデルを使うと、コードのボトルネックを心配せず、1つのイベントに対して複数のリスナーを気軽に追加できます。

イベントをブロードキャストできる ActionScript クラスには、Key、Mouse、MovieClipLoader、Selection、Stage、TextField があります。クラスでどのリスナーが利用可能かについては、『ActionScript 2.0 リファレンスガイド』の各クラス項目を参照してください。

イベントリスナーの詳細については、次に示す各トピックを参照してください。

- 285 ページの「イベントリスナーモデル」
- 286 ページの「イベントリスナーの例」

ブラウザウィンドウがサイズ変更されたときに `Stage.scaleMode` プロパティが `Stage.width` および `Stage.height` の値にどのように影響するかを示す、サンプルソースファイル "stagesize.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/StageSize" フォルダに移動してサンプルにアクセスします。

イベントリスナーモデル

イベントリスナーのイベントモデルは、イベントハンドラ（詳細については 280 ページの「ActionScript とイベントについて」を参照）のモデルと似ていますが、次の 2 点が大きく違います。

- イベントハンドラは、イベントをブロードキャストするオブジェクトではなく、リスナーオブジェクトに割り当てる。
- ブロードキャスターオブジェクトに対して addListener() という特別なメソッドを呼び出して、ブロードキャスターのイベントを受け取るリスナーオブジェクトを登録する必要がある。

次のコードは、イベントリスナーモデルの概略を表したものです。

```
var listenerObject:Object = new Object();
listenerObject.eventName = function(eventObj:Object) {
  // ここにコードを記述
};
broadcasterObject.addListener(listenerObject);
```

このコードは、オブジェクト *listenerObject*、プロパティ *eventName* から始まります。リスナーオブジェクトは任意のオブジェクトにすることができます。既存のオブジェクト、ムービークリップ、ステージ上のボタンのインスタンス、あるいは ActionScript クラスのインスタンスのいずれでも可能です。たとえば、カスタムムービークリップがステージリスナーのリスナーメソッドを実装できるとします。複数の種類のリスナーをリッスンするオブジェクトも可能です。

eventName プロパティは *broadcasterObject* で発生するイベントです。このイベントが *listenerObject* に対してブロードキャストされます。1 つのイベントのブロードキャスターに対して複数のリスナーを登録することも可能です。

続いて、イベントリスナーに対して、イベントに反応する何らかの処理を実行する関数を割り当てます。

最後に、ブロードキャスターオブジェクトの addListener() メソッドを呼び出してリスナーオブジェクトを渡します。

イベントの受け取りをやめるようにリスナーオブジェクトの登録を解除するには、ブロードキャスターオブジェクトの removeListener() メソッドを呼び出し、削除するイベントの名前およびリスナーオブジェクトを渡します。

```
broadcasterObject.removeListener(listenerObject);
```

イベントリスナーの例

次の例は、Selection クラスで onSetFocus イベントのイベントリスナーを使って、複数のテキスト入力フィールドのフォーカスを管理する簡単なフォーカスマネージャを作成する方法を示したものです。ここでは、キーボードフォーカスがあたったテキストフィールドのボーダーを有効に (表示) して、フォーカスのないテキストフィールドのボーダーを無効にしています。

イベントリスナーを使って簡単なフォーカスマネージャを作成するには :

1. テキストツールを使用して、テキストフィールドをステージに作成します。
2. 作成したテキストフィールドを選択し、プロパティインスペクタで、[テキストの種類] ポップアップメニューから [テキスト入力] を選択し、[テキストの周囲にボーダーを表示] ボタンをクリックします。
3. テキスト入力フィールドの下に、もう 1 つ別のテキスト入力フィールドを作成します。

   こちらのテキストフィールドは [テキストの周囲にボーダーを表示] オプションが選択されていないことを確認してください。テキスト入力フィールドの作成を続行できます。
4. タイムラインでフレーム 1 を選択し、[アクション] パネルを開きます ([ウィンドウ]-[アクション])。
5. 次のコードを [アクション] パネルに入力します。これは、Selection クラスからのフォーカス通知を受け取るオブジェクトを作成するコードです。

```
// リスナーオブジェクト focusListener を作成する
var focusListener:Object = new Object();
// リスナーオブジェクトの関数を定義する
focusListener.onSetFocus = function(oldFocus_txt:TextField,
  newFocus_txt:TextField) {
  oldFocus_txt.border = false;
  newFocus_txt.border = true;
}
```

   このコードは、focusListener という名前のオブジェクトを作成します。このオブジェクトは、onSetFocus プロパティを定義し、このプロパティに関数を割り当てます。この関数は 2 つのパラメータを取ります。2 つのパラメータとは、フォーカスのないテキストフィールドへの参照と、フォーカスがあるテキストフィールドへの参照です。関数では、フォーカスのないテキストフィールドの border プロパティを false に、フォーカスがあるテキストフィールドの border プロパティを true に、それぞれ設定しています。
6. 次のコードを [アクション] パネルに追加します。これは、Selection オブジェクトからイベントを受け取るように focusListener オブジェクトを登録するためのコードです。

```
// ブロードキャスターを使用して focusListener を登録する
Selection.addListener(focusListener);
```

7. アプリケーションをテストします ([制御]-[ムービープレビュー])。最初のテキストフィールドをクリックし、Tab キーを押してフィールド間でフォーカスを移動します。

コンポーネントでのイベントリスナーの使用

コンポーネントを操作する場合、イベントリスナーのシンタックスが少し異なります。コンポーネントは、イベントを生成します。これらのイベントは、リスナーオブジェクトまたはカスタム関数を使用して特別に監視する必要があります。

次の例では、イベントリスナーを使用して、動的にロードされるイベントのダウンロードの進捗状況を監視する方法について説明します。

Loader コンポーネントのイベントを監視するには :

1. Loader コンポーネントのインスタンスを [コンポーネント] パネルからステージ上にドラッグします。
2. ローダーを選択し、プロパティインスペクタの [インスタンス名] テキストボックスに「my_ldr」と入力します。
3. メインタイムラインのフレーム 1 に次のコードを追加します。

```
System.security.allowDomain("http://www.helpexamples.com");

var loaderListener:Object = new Object();
loaderListener.progress = function(evt_obj:Object):Void {
  trace(evt_obj.type); // 進捗状況
  trace("\t" + evt_obj.target.bytesLoaded + " of " +
  evt_obj.target.bytesTotal + " bytes loaded");
}
loaderListener.complete = function(evt_obj:Object):Void {
  trace(evt_obj.type); // 完了
}

my_ldr.addEventListener("progress", loaderListener);
my_ldr.addEventListener("complete", loaderListener);
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

この ActionScript コードは、progress と complete という 2 つのイベントを監視する loaderListener という名前のリスナーオブジェクトを定義します。これらの各イベントが送り出されると、これらのコードが実行され、[出力] パネルにデバッグテキストが表示されます (オーサリングツールで SWF ファイルをテストする場合)。

次に、my_ldr インスタンスに対して 2 つの指定した各イベント (progress と complete) を監視するように指示し、イベントが送り出されたときに実行するリスナーオブジェクトまたは関数を指定します。最後に、Loader.load( ) メソッドが呼び出され、イメージのダウンロードの開始をトリガします。

4. [制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

   イメージは、ステージ上の Loader インスタンス内にダウンロードされ、[出力] パネルにいくつかのメッセージが表示されます。ダウンロードするイメージのサイズ、およびイメージがユーザーのローカルシステムにキャッシュされたかどうかに応じて、progress イベントは何度も送り出される可能性があります。これに対して、complete イベントは、イメージが完全にダウンロードされた後に 1 回だけ送り出されます。

   コンポーネントおよび dispatch イベントを操作する場合のシンタックスは、前の例のイベントリスナーの場合と少し異なります。特に、addListener() を呼び出す代わりに addEventListener() メソッドを呼び出す必要があります。さらに、イベントリスナーオブジェクトまたは関数に加えて、監視する特定のイベントを指定する必要があります。

リスナーオブジェクトを使う代わりに、287 ページの「コンポーネントでのイベントリスナーの使用」で説明されている最初の手順にしたがってカスタム関数を使用できます。前の例のコードは、次のように書き直すことができます。

```
System.security.allowDomain("http://www.helpexamples.com");

my_ldr.addEventListener("progress", progressListener);
my_ldr.addEventListener("complete", completeListener);
my_ldr.load("http://www.helpexamples.com/flash/images/image1.png");

function progressListener(evt_obj:Object):Void {
  trace(evt_obj.type); // 進捗状況
  trace("\t" + evt_obj.target.bytesLoaded + " of " +
  evt_obj.target.bytesTotal + " bytes loaded");
}
function completeListener(evt_obj:Object):Void {
  trace(evt_obj.type); // 完了
}
```

メモ

前の例では、常に、Loader.load() メソッドが呼び出される前に常にイベントリスナーが追加されていました。それで、イベントリスナーを指定する前に Loader.load() メソッドを呼び出すと、イベントリスナーが完全に定義される前にロードが完了する可能性があります。これは、コンテンツが表示されても、complete イベントがキャッチされない可能性があることを意味します。

ボタンおよびムービークリップのイベントハンドラの使用

ステージ上のボタンおよびムービークリップのインスタンスに対しては、onClipEvent() イベントハンドラおよび on() イベントハンドラを使って、イベントハンドラを直接割り当てることができます。onClipEvent() イベントハンドラはムービークリップイベントをブロードキャストし、on() イベントハンドラはボタンイベントをブロードキャストします。

ボタンまたはムービークリップのインスタンスにイベントハンドラを割り当てるには、ステージ上のボタンまたはムービークリップのインスタンスをクリックしてフォーカスをあててから [アクション] パネルにコードを入力します。コードがボタンまたはムービークリップに割り当てられると、[アクション] パネルのタイトルは、[アクション - ボタン] または [アクション - ムービークリップ] に変わります。ボタンとムービークリップのインスタンスに割り当てられたコード使用方法のガイドラインについては、660ページの「オブジェクトへのコードのアタッチ」を参照してください。

メモ	ボタンとムービークリップのイベントハンドラを SimpleButton.click、UIObject.hide および UIObject reveal などのコンポーネントイベントと混同しないように注意してください。コンポーネントイベントはコンポーネントインスタンスに割り当てる必要があります。詳細については、『ActionScript 2.0 コンポーネントガイド』を参照してください。

onClipEvent() および on() は、オーサリング時にステージ上に配置されたムービークリップインスタンスに対してのみ割り当てることができます。attachMovie() メソッドなどを使用して実行時に作成されるムービークリップインスタンスに対しては、onClipEvent() や on() を割り当てることはできません。実行時に作成されるオブジェクトにイベントハンドラを割り当てるには、イベントハンドラメソッドかイベントリスナーを使用する必要があります。詳細については、280 ページの「ActionScript とイベントについて」と 284 ページの「イベントリスナーの使用」を参照してください。

メモ	onClipEvent() および on() ハンドラを割り当てるという方法は推奨されていません。その代わり、本マニュアルで示されているように、コードをフレームスクリプトまたはクラスファイル内に配置するようにしてください。詳細については、280 ページの「ActionScript とイベントについて」および 660 ページの「オブジェクトへのコードのアタッチ」を参照してください。

ボタンとムービークリップのイベントハンドラの詳細については、次のトピックを参照してください。

- 290 ページの「on や onClipEvent とイベントハンドラメソッドの併用」
- 291 ページの「on メソッドまたは onClipEvent メソッドに対するイベントの指定」
- 292 ページの「1 つのオブジェクトに対する複数のハンドラの割り当て」

on や onClipEvent とイベントハンドラメソッドの併用

場合によっては、他の方法でコンフリクトを避けながらイベントを処理することもできます。on() メソッドおよび onClipEvent() メソッドによるイベント処理は、定義したイベントハンドラメソッドを使用するイベント処理とはコンフリクトしません。

たとえば、SWF ファイル内のボタンの例で考えてみます。このボタンには、SWF を再生させる on(press) ハンドラと、ステージ上のオブジェクトを回転させる処理を関数で定義する onPress() メソッドの両方を実装することが可能です。ボタンをクリックすると、SWF ファイルが再生され、オブジェクトが回転します。呼び出すイベントの種類や時期に応じて、on() メソッドと onClipEvent() メソッド、イベントハンドラメソッド、またはその両方を組み合わせてイベント処理方法として使用します。

ただし、on() および onClipEvent() ハンドラでの変数およびオブジェクトのスコープは、イベントハンドラやイベントリスナーのスコープとは異なります。295 ページの「イベントハンドラのスコープ」を参照してください。

また、on() をムービークリップで使用すると、ボタンイベントを受け取るムービークリップを作成できます。詳細については、293 ページの「ボタン状態を持つムービークリップの作成」を参照してください。on() および onClipEvent() に対応するイベントを指定する方法の詳細については、291 ページの「on メソッドまたは onClipEvent メソッドに対するイベントの指定」を参照してください。

on ハンドラおよび onPress イベントハンドラを使用するには :

1. 新しい Flash ドキュメントを作成し、handlers.fla という名前で保存します。
2. 矩形ツールを選択し、ステージに大きな四角形を描画します。
3. 選択ツールを選択し、ステージ上の四角形をダブルクリックします。それから F8 キーを押して [シンボルに変換] ダイアログボックスを表示します。
4. ボックスのシンボル名を入力し、型をムービークリップに設定して [OK] をクリックします。
5. ステージ上のムービークリップに box_mc というインスタンス名を付けます。
6. 次の ActionScript をステージ上のムービークリップシンボルに直接追加します。

```
on (press) {
   trace("on (press) {...}");
}
```

7. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
box_mc.onPress = function() {
   trace("box_mc.onPress = function() {...};");
};
```

8. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

ステージ上のムービークリップシンボルをクリックすると、次の出力が [出力] パネルに送られます。

```
on (press) {...}
box_mc.onPress = function() {...};
```

メモ

onClipEvent() および on() ハンドラを割り当てるという方法は推奨されていません。その代わり、本マニュアルで示されているように、コードをフレームスクリプトまたはクラスファイル内に配置するようにしてください。詳細については、280 ページの「ActionScript とイベントについて」および 660 ページの「オブジェクトへのコードのアタッチ」を参照してください。

on メソッドまたは onClipEvent メソッドに対するイベントの指定

on() ハンドラおよび onClipEvent() ハンドラを使用するには、ステージ上のボタンまたはムービークリップのインスタンスに対してハンドラを直接割り当て、そのインスタンスで処理するイベントを指定します。on() および onClipEvent() イベントハンドラでサポートされる全イベントの一覧については、『ActionScript 2.0 リファレンスガイド』の on ハンドラおよび onClipEvent ハンドラを参照してください。

たとえば、次の on() イベントハンドラは、そのハンドラを割り当てたボタンをユーザーがクリックするたびに実行されます。

```
on (press) {
  trace("Thanks for pressing me.");
}
```

1 つの on() ハンドラに対して複数のイベントを指定することも可能です。その場合はイベントをカンマで区切って指定します。ハンドラ内の ActionScript は、ハンドラで指定したイベントのいずれかが発生したときに実行されます。たとえば、次の on() ハンドラをボタンに対して割り当てると、マウスがボタン上を移動したときかボタン外に移動したときにこのハンドラが実行されます。

```
on (rollOver, rollOut) {
  trace("You rolled over, or rolled out");
}
```

on() ハンドラを使用してキー入力イベントを追加することもできます。たとえば次のコードでは、キーボードで数字の 3 が押されたときにストリングが表示されます。ボタンまたはムービークリップのインスタンスを選択し、次のコードを [アクション] パネルに追加してください。

```
on (keyPress "3") {
  trace("You pressed 3")
}
```

Enter キーがユーザーによって押されたときにストリングを表示したい場合は、次のコードフォーマットを使用できます。ボタンまたはムービークリップのインスタンスを選択し、次のコードを [アクション] パネルに追加してください。

```
on (keyPress "<Enter>") {
   trace("Enter Pressed");
}
```

[制御]-[ムービープレビュー] を選択し、Enter キーを押して、[出力] パネルにストリングが表示されるか確認します。何も表示されない場合、[制御]-[キーボードショートカットを無効] を選択してもう一度やり直してください。キーを押した場合のインタラクティブ機能をアプリケーションに追加する方法の詳細については、Key を参照してください。

メモ

onClipEvent() および on() ハンドラを割り当てるという方法は推奨されていません。その代わり、本マニュアルで示されているように、コードをフレームスクリプトまたはクラスファイル内に配置するようにしてください。詳細については、280 ページの「ActionScript とイベントについて」および 660 ページの「オブジェクトへのコードのアタッチ」を参照してください。

1 つのオブジェクトに対する複数のハンドラの割り当て

さまざまなイベントをトリガとして各種のスクリプトが実行されるようにするには、オブジェクトに複数のハンドラを割り当てます。たとえば、1 つのムービークリップインスタンスに対して、次の 2 つの onClipEvent() ハンドラを割り当てることができます。1 番目のハンドラは、ムービークリップが最初にロードされるとき (またはステージに表示されるとき) に実行されます。2 番目は、ムービークリップがステージからアンロードされるときに実行されます。

```
on (press) {
   this.unloadMovie()
}
onClipEvent (load) {
   trace("I've loaded");
}
onClipEvent (unload) {
   trace("I've unloaded");
}
```

メモ

onClipEvent() および on() ハンドラを割り当てるという方法は推奨されていません。その代わり、本マニュアルで示されているように、コードをフレームスクリプトまたはクラスファイル内に配置するようにしてください。詳細については、280 ページの「ActionScript とイベントについて」および 660 ページの「オブジェクトへのコードのアタッチ」を参照してください。

複数のハンドラを、タイムラインに配置されたコードを使用して 1 つのオブジェクトに割り当てる方法については、次の例を参照してください。このコードでは、onPress ハンドラおよび onRelease ハンドラをムービークリップインスタンスに割り当てます。

複数のハンドラをオブジェクトに割り当てるには :

1. 新しい Flash ドキュメントを作成し、assignMulti.fla という名前を付けます。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
this.createEmptyMovieClip("img_mc", 10);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
     target_mc.onPress = function() {
          target_mc.startDrag();
     };
     target_mc.onRelease = function() {
          target_mc.stopDrag();
     };
}
mclListener.onLoadError = function(target_mc:MovieClip) {
     trace("error downloading image");
}
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   イメージが img_mc インスタンスにロードされ、onPress() および onRelease() イベントハンドラにより、ステージにおいてイメージをドラッグできるようになります。

コンポーネントインスタンスからのイベントのブロードキャスト

任意のコンポーネントインスタンスに対して、イベント処理の方法を指定できます。コンポーネントイベントは、ネイティブ ActionScript オブジェクトからブロードキャストされるイベントとは異なった方法で処理されます。

詳細については、『ActionScript 2.0 コンポーネントガイド』を参照してください。

ボタン状態を持つムービークリップの作成

ムービークリップに on() ハンドラを割り当てるか、またはムービークリップインスタンスの MovieClip マウスイベントハンドラのいずれかに関数を割り当てると、ムービークリップはボタンと同じようにマウスイベントに反応することができます。また、ムービークリップのタイムラインに、_up、_over、および _down というフレームラベルを追加すると、ムービークリップに自動的にボタン状態 (アップ、オーバー、およびダウン) を作成することができます。

ユーザーがムービークリップ上でマウスの移動やクリックを行うと、再生ヘッドが該当するフレームラベルのフレームに送られます。ムービークリップが使用するヒット領域を指定するには、hitArea (MovieClip.hitArea property) プロパティを使用します。

ムービークリップにボタン状態を作成するには :

1. 新しい Flash ドキュメントを作成し、mcbutton.fla という名前で保存します。
2. 矩形ツールを使用して、小さな矩形 (だいたい幅 100 ピクセル、高さ 20 ピクセルくらい) をステージに描画します。
3. 選択ツールでシェイプをダブルクリックし、それから F8 キーを押して [シンボルに変換] ダイアログボックスを表示します。
4. シンボル名を「mcbutton」と入力し、シンボルタイプはムービークリップに設定して [OK] をクリックします。
5. ステージでムービークリップシンボルをダブルクリックし、シンボル編集モードに切り替えます。
6. ムービークリップのタイムラインで新しいレイヤーを作成し、新しいレイヤーの名前を labels に変更します。
7. プロパティインスペクタでフレームラベルを「`_up`」と入力します。
8. デフォルトレイヤーおよび labels レイヤーの上に新しいレイヤーを作成します。
9. 新しいレイヤーの名前を actions に変更し、ムービークリップのタイムラインのフレーム 1 に次の ActionScript を追加します。

   ```
   stop();
   ```

10. フレーム 10、および 3 つのレイヤーすべてを選択し、[挿入]-[タイムライン]-[キーフレーム] を選択します。
11. actions レイヤーのフレーム 10 に `stop()` アクションを追加し、labels レイヤーのフレーム 10 にフレームラベル _over を追加します。
12. フレーム 10 で矩形を選択し、プロパティインスペクタを使用して別の塗りのカラーを選択します。
13. 3 つのレイヤーそれぞれのフレーム 20 に新しいキーフレームを作成し、プロパティインスペクタでフレームラベル _down を追加します。
14. フレーム 20 の矩形の色を、3 つのボタン状態がそれぞれ別のカラーになるように変更します。
15. メインタイムラインに戻ります。
16. ムービークリップがマウスイベントに対応できるようにするには、次のいずれかの操作を実行します。
    - ムービークリップインスタンスに `on()` イベントハンドラを割り当てます。詳細については、289 ページの「ボタンおよびムービークリップのイベントハンドラの使用」を参照してください。
    - ムービークリップオブジェクトのマウスイベントハンドラ (`onPress` や `onRelease` など) のいずれかに関数を割り当てます。詳細については、280 ページの「ActionScript とイベントについて」を参照してください。
17. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

    マウスポインタをステージのムービークリップに合わせると、ムービークリップは自動的に _over 状態になります。ムービークリップインスタンスをクリックすると、再生ヘッドは自動的にムービークリップの _down 状態に進みます。

イベントハンドラのスコープ

イベントハンドラ内で宣言や実行を行う変数やコマンドのスコープ (コンテキスト) は、使用しているイベントハンドラの種類によって異なります。つまり、イベントハンドラメソッドやイベントリスナーを使用している場合と、on() および onClipEvent() ハンドラを使用している場合では異なります。新しい ActionScript クラスのイベントハンドラを定義すると、イベントハンドラの定義に応じてスコープも変わります。ここでは、ActionScript 1.0 と ActionScript 2.0 の両方の例について説明します。

ActionScript 1.0 の例　自分で作成した ActionScript のすべての関数と同様に、イベントハンドラメソッドやイベントリスナーに割り当てられる関数ではローカル変数スコープが定義されますが、on() ハンドラおよび onClipEvent() ハンドラでは定義されません。

たとえば、次の 2 つのイベントハンドラを検討してみます。1 番目は clip_mc という名前のムービークリップに関連付けられた onPress イベントハンドラであり、2 番目は同じムービークリップインスタンスに割り当てられた on() ハンドラです。

```
// clip_mc の親クリップタイムラインに割り当てられている
clip_mc.onPress = function () {
   var shoeColor; // ローカル関数の変数
   shoeColor = "blue";
}
// clip_mc に割り当てられた on() ハンドラ :
on (press) {
   var shoeColor; // ローカル変数スコープではない
   shoeColor = "blue";
}
```

どちらのイベントハンドラもコードは同じですが、結果は異なります。1 番目のハンドラの場合は、color 変数は、onPress 用に定義された関数にローカルな変数です。一方、2 番目のハンドラの場合、on() ハンドラがローカル変数スコープを定義しないため、この変数はムービークリップ clip_mc のタイムラインのスコープで定義されます。

ムービークリップにではなく、ボタンに割り当てられる on() イベントハンドラの場合、変数 (と関数呼び出しおよびメソッド呼び出し) がボタンインスタンスを含むタイムラインのスコープに呼び出されます。

たとえば、次の on() イベントハンドラをボタンオブジェクトに割り当てた場合とムービークリップオブジェクトに割り当てた場合とでは、結果は異なるものになります。前者の場合、play() 関数呼び出しは、ボタンを含むタイムラインの再生ヘッドを開始し、後者の場合、play() 関数呼び出しは、ハンドラを割り当てたムービークリップのタイムラインを開始します。

```
// ボタンに割り当てる。
on (press) {
   play(); // 親タイムラインを再生する。
}
// ムービークリップに割り当てる。
on (press) {
   play(); // ムービークリップのタイムラインを再生する。
}
```

ボタンオブジェクトに割り当てられると、play() 関数は、ボタンを含むタイムライン、つまり、ボタンの親タイムラインに適用されます。一方、on(press) ハンドラがムービークリップオブジェクトに割り当てられると、play() 関数呼び出しはそのハンドラを持つムービークリップに適用されます。次のコードをムービークリップに割り当てると、親タイムラインが再生されます。

```
// ムービークリップに割り当てる。
on (press) {
  _parent.play(); // 親タイムラインを再生する。
}
```

イベントハンドラ内またはイベントリスナー定義内では、同一の play() 関数が関数定義を含むタイムラインに適用されます。たとえば、my_mc ムービークリップインタンスを持つタイムラインで次の my_mc.onPress イベントハンドラメソッドを宣言するとします。

```
// タイムラインで定義された関数は
my_mc.onPress = function () {
  play(); // 定義されているタイムラインを再生する。
};
```

onPress イベントハンドラを定義するムービークリップを再生するには、次のように、this キーワードを使用してそのムービークリップを明示的に参照してください。

```
// ルートタイムラインに定義されている関数は
my_mc.onPress = function () {
  this.play(); // my_mc クリップのタイムラインを再生する。
};
```

ただし、同じコードがボタンインスタンスのルートタイムラインに配置されていると、代わりにルートタイムラインが再生されます。

```
my_btn.onPress = function () {
  this.play(); // ルートタイムラインを再生する
};
```

イベントハンドラでの this キーワードのスコープの詳細については、298 ページの「this キーワードのスコープ」を参照してください。

ActionScript 2.0 の例　次のコードでは、TextLoader クラスを使用してテキストファイルをロードし、ファイルのロード完了後にテキストを表示します。

```
// TextLoader.as
class TextLoader {
  private var params_lv:LoadVars;
  public function TextLoader() {
    params_lv = new LoadVars();
    params_lv.onLoad = onLoadVarsDone;
    params_lv.load("http://www.helpexamples.com/flash/params.txt");
  }
  private function onLoadVarsDone(success:Boolean):Void {
    _level0.createTextField("my_txt", 999, 0, 0, 100, 20);
    _level0.my_txt.autoSize = "left";
    _level0.my_txt.text = params_lv.monthNames; // undefined
  }
}
```

このコードは、イベントハンドラのスコープに関して問題があるため正しく動作しません。this が onLoad イベントハンドラを参照しているのか、このクラスを参照しているのかが混同されています。onLoadVarsDone() メソッドが TextLoader オブジェクトのスコープで呼び出されることが前提のコーディングとなっていますが、このメソッドは TextLoader オブジェクトから取り出して LoadVars オブジェクトに移植されているため、実際には LoadVars オブジェクトのスコープで呼び出されます。テキストファイルのロードが正常に終了すると、LoadVars オブジェクトから this.onLoad イベントハンドラが呼び出され、続いて onLoadVarsDone() 関数が呼び出されます。このとき、this は TextLoader ではなく LoadVars に設定されています。params_lv オブジェクトは、呼び出されたとき、onLoadVarsDone() 関数が参照によって params_lv オブジェクトに依存しているにも関わらず、this スコープ内にあります。したがって、onLoadVarsDone() 関数は params_lv.params_lv というインスタンスにアクセスしようとしていることになりますが、これは実際には存在しません。

TextLoader オブジェクトのスコープの onLoadVarsDone() メソッドを正しく呼び出すには、次のようにします。関数リテラルを使用して、該当する関数を呼び出す匿名関数を作成します。owner オブジェクトが匿名関数のスコープで参照可能なため、呼び出す TextLoader オブジェクトの検索に使用できます。

```
// TextLoader.as
class TextLoader {
  private var params_lv:LoadVars;
  public function TextLoader() {
    params_lv = new LoadVars();
    var owner:TextLoader = this;
    params_lv.onLoad = function (success:Boolean):Void {
      owner.onLoadVarsDone(success);
    }
    params_lv.load("http://www.helpexamples.com/flash/params.txt");
  }
  private function onLoadVarsDone(success:Boolean):Void {
    _level0.createTextField("my_txt", 999, 0, 0, 100, 20);
    _level0.my_txt.autoSize = "left";
    _level0.my_txt.text = params_lv.monthNames; // 1 月、2 月、3 月、...
  }
}
```

this キーワードのスコープ

this キーワードは、現在実行中のスコープ内のオブジェクトを参照します。使用しているイベントハンドラの技法に従って、this は異なるオブジェクトを参照します。

イベントハンドラまたはイベントリスナー関数内では、this は、イベントハンドラまたはイベントリスナーメソッドを定義するオブジェクトを参照します。たとえば、次のコードでは this は my_mc を参照します。

```
// メインタイムラインに割り当てられた onPress() イベントハンドラ :
my_mc.onPress = function () {
  trace(this); // _level0.my_mc
}
```

ムービークリップに割り当てられた on() ハンドラ内では、this は on( ) ハンドラが割り当てられているムービークリップを参照します。コードは次のようになります。

```
// メインタイムライン上の my_mc という名前のムービークリップに割り当てられる
on (press) {
  trace(this); // _level0.my_mc
}
```

ボタンに割り当てられた on() ハンドラ内では、this はそのボタンを含むタイムラインを参照します。コードは次のようになります。

```
// メインタイムライン上のボタンに割り当てられる
on (press) {
  trace(this); // _level0
}
```

Delegate クラスの使用

Delegate クラスを使用すると、特定のスコープ内で関数を実行できます。このクラスは、同じイベントを 2 つの異なる関数に送り出すことや (『ActionScript 2.0 コンポーネントガイド』を参照)、関数をそれが属しているクラスのスコープ内で呼び出すことを可能にするために用意されています。

関数をパラメータとして EventDispatcher.addEventListener( ) に渡すと、渡した関数は、それが宣言されているオブジェクトのスコープではなく、ブロードキャスターコンポーネントインスタンスのスコープで呼び出されます (『ActionScript 2.0 コンポーネントガイド』を参照)。関数を、それが宣言されているオブジェクトのスコープで呼び出すには、Delegate.create( ) を使用します。

次の例では、Button コンポーネントインスタンスのイベントをリッスンする 3 種類の方法を示します。それぞれの方法で Button コンポーネントインスタンスにイベントリスナーを追加すると、それぞれ別のスコープにイベントが送り出されます。

Delegate クラスを使ってイベントをリッスンするには :

1. 新しい Flash ドキュメントを作成し、delegate.fla という名前で保存します。
2. Button コンポーネントを [コンポーネント] パネルの "User Interface" フォルダからライブラリにドラッグします。

   ステージへのボタンインスタンスの追加および配置は ActionScript を使用してこの後の手順で行います。
3. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
import mx.controls.Button;
import mx.utils.Delegate;

function clickHandler(eventObj:Object):Void {
  trace("[" + eventObj.type + "] event on " + eventObj.target + "
  instance.");
  trace("\t this -> " + this);
}

var buttonListener:Object = new Object();
buttonListener.click = function(eventObj:Object):Void {
  trace("[" + eventObj.type + "] event on " + eventObj.target + "
  instance.");
  trace("\t this -> " + this);
};

this.createClassObject(Button, "one_button", 10, {label:"One"});
one_button.move(10, 10);
one_button.addEventListener("click", clickHandler);

this.createClassObject(Button, "two_button", 20, {label:"Two"});
two_button.move(120, 10);
two_button.addEventListener("click", buttonListener);

this.createClassObject(Button, "three_button", 30, {label:"Three"});
three_button.move(230, 10);
three_button.addEventListener("click", Delegate.create(this, clickHandler));
```

上記のコードは 6 つのセクションに分けられています (各セクションは空白行で区切られています)。最初のセクションでは、Delegate クラスとともに (Button コンポーネントの) Button クラスが読み込まれます。コードの 2 番目のセクションでは、ユーザーがボタンをクリックしたときに呼び出される関数が定義されています。コードの 3 番目のセクションでは、イベントリスナーとして使用するオブジェクトを作成しています。このオブジェクトはイベント `click` をリッスンします。

コードの残りの 3 つのセクションでは、ステージに新しい Button コンポーネントを作成し、インスタンスの位置を変更し、click イベントに対するイベントリスナーを追加しています。最初のボタンは click イベントに対するイベントリスナーを追加し、click ハンドラ関数に参照を直接渡します。2 番目のボタンは click イベントのイベントリスナーを追加し、リスナーオブジェクトへの参照を渡します。このリスナーオブジェクトには click イベントのハンドラが含まれています。最後に、3 つめの関数では click イベントのイベントリスナーを追加し、Delegate クラスを使って this スコープ (ここで this は _level0) の click イベントを送り出し、click ハンドラ関数に参照を渡します。

4. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。
5. ステージで各ボタンインスタンスをクリックし、どのスコープでイベントが処理されているかを確認してください。
   a. ステージで最初のボタンをクリックすると、[出力] パネルに次のテキストが表示されます。

      ```
      [click] event on _level0.one_button instance.
         this -> _level0.one_button
      ```

      one_button インスタンスをクリックすると、this スコープはボタンインスタンスそのものを参照します。
   b. ステージで 2 番目のボタンをクリックすると、[出力] パネルに次のテキストが表示されます。

      ```
      [click] event on _level0.two_button instance.
         this -> [object Object]
      ```

      two_button インスタンスをクリックすると、this スコープは buttonListener オブジェクトを参照します。
   c. ステージで 3 番目のボタンをクリックすると、[出力] パネルに次のテキストが表示されます。

      ```
      [click] event on _level0.three_button instance.
         this -> _level0
      ```

      three_button インスタンスをクリックすると、this スコープは Delegate.create( ) メソッド呼び出しで指定されたスコープを参照します。ここでは _level0 です。

第 10 章

ムービークリップの操作

10

ムービークリップとは、必要な機能をすべて備えたミニチュア版の SWF ファイルのようなものです。個々のムービークリップは互いに独立し、配置先のタイムラインとは異なる独自のタイムラインで再生されます。たとえば、メインタイムラインにフレームが 1 つだけあり、そのフレームの中に 10 フレームの長さのムービークリップが 1 つ含まれているとします。この場合、メイン SWF ファイルを再生すると、ムービークリップの各フレームが再生されます。逆に、ムービークリップの中に別のムービークリップを含めることもできます。これをムービークリップを「ネストする」といいます。このようにネストしたムービークリップには、階層的な関係が構築されます。つまり、「親クリップ」の中に 1 つ以上の「子クリップ」が含まれます。

ムービークリップのインスタンスに名前を付けて、ムービークリップを ActionScript で制御可能なオブジェクトとして一意に識別することができます。ムービークリップのインスタンスに付けた「インスタンス名」は、MovieClip クラスタイプのオブジェクトとしてインスタンスを識別します。MovieClip クラスのプロパティとメソッドを使用すると、実行時のムービークリップの外観と動作を制御できます。

ムービークリップは、イベントへの応答、他のムービークリップオブジェクトへのメッセージの送信、自身の状態の保持、および子クリップの管理ができるため、自律オブジェクトと見なすことができます。このようなムービークリップの性質が、Flash CS3 Professional で " コンポーネントベースアーキテクチャ " を実現する基礎となっています。[コンポーネント] パネル ([ウィンドウ]-[コンポーネント]) に用意されているコンポーネント群も、実際には、外観や動作が特定の決まりに基づいて設計およびプログラムされている高度なムービークリップに他なりません。

Drawing API (MovieClip クラスの描画メソッド)、フィルタ、ブレンド、スクリプトで作成されたアニメーションなどの使用方法の詳細については、第 12 章の「アニメーション、フィルタおよび描画」を参照してください。

ムービークリップの詳細については、次のトピックを参照してください。

ActionScript によるムービークリップの制御について 302
1 つのムービークリップでの複数メソッドの呼び出し 304
SWF ファイルのロードおよびアンロード 304
ムービークリップの位置と画質の変更 307
ムービークリップのドラッグ 308
実行時のムービークリップの作成. 309
動的に作成したムービークリップへのパラメータの追加. 313
ムービークリップの深度の管理. 314
ActionScript を使用したムービークリップのキャッシュとスクロールについて 318
マスクとしてのムービークリップの使用 325
ムービークリップイベントの処理. 327
ムービークリップシンボルへのクラスの割り当て 327
クラスプロパティの初期化 328

ActionScript によるムービークリップの制御について

ActionScript のグローバル関数や MovieClip クラスのメソッドを使用すると、ムービークリップに対するタスクを実行することができます。MovieClip クラスのメソッドには、同名の関数と同じタスクを実行するものもあれば、`hitTest()` や `swapDepths()` などのように対応する関数名がないものもあります。

次の例は、メソッドを使用する場合と関数を使用する場合の違いを示したものです。いずれのステートメントでも、インスタンス `my_mc` を複製し、新しいクリップに `new_mc` という名前を付けて深度 5 に配置します。

```
my_mc.duplicateMovieClip("new_mc", 5);
duplicateMovieClip(my_mc, "new_mc", 5);
```

関数とメソッドの動作が似ているときは、いずれか一方を使用してムービークリップを制御します。どちらを選択するかは、ユーザーの好みと、ActionScript を使ったスクリプト作成の習熟度によって決まります。関数とメソッドのどちらを使用する場合でも、関数またはメソッドを呼び出すときにターゲットタイムラインを Flash Player にロードする必要があります。

メソッドを使用するには、次のステートメントに示すように、インスタンス名のターゲットパスの後ろにドット (.)、その後ろにメソッド名とパラメータを付けてメソッドを呼び出します。

```
myMovieClip.play();
parentClip.childClip.gotoAndPlay(3);
```

最初のステートメントでは、play() メソッドが myMovieClip インスタンスの再生ヘッドを移動します。2 番目のステートメントでは、gotoAndPlay() が childClip (parentClip インスタンスの子) にある再生ヘッドをフレーム 3 に送り、再生ヘッドの移動を続行します。

タイムラインを制御するグローバル関数には、*target* パラメータがあります。このパラメータには、制御対象のインスタンスへのターゲットパスを指定できます。たとえば、次のスクリプトでは、startDrag() 関数はコードが配置されているインスタンスをターゲットとし、ドラッグ可能にします。

```
my_mc.onPress = function() {
  startDrag(this);
};
my_mc.onRelease = function() {
  stopDrag();
};
```

loadMovie()、unloadMovie()、loadVariables()、setProperty()、startDrag()、duplicateMovieClip()、removeMovieClip() の各関数は、ムービークリップをターゲットにします。これらの関数を使用するには、関数の *target* パラメータにターゲットパスを入力して、関数のターゲットを指定する必要があります。

MovieClip.attachMovie()、MovieClip.createEmptyMovieClip()、MovieClip.createTextField()、MovieClip.getBounds()、MovieClip.getBytesLoaded()、MovieClip.getBytesTotal()、MovieClip.getDepth()、MovieClip.getInstanceAtDepth()、MovieClip.getNextHighestDepth()、MovieClip.globalToLocal()、MovieClip.localToGlobal()、MovieClip.hitTest()、MovieClip.setMask()、MovieClip.swapDepths() の各 MovieClip メソッドは、ムービークリップやロードされるレベルを制御でき、これらと同等の関数はありません。

これらの関数およびメソッドの詳細については、『ActionScript 2.0 リファレンスガイド』の該当する項目を参照してください。

スクリプトで作成された Flash のアニメーションを示すサンプルのソースファイル "animation.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Animation" フォルダに移動してサンプルにアクセスします。

フォトギャラリアプリケーションのサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Galleries" フォルダに移動して次のサンプルにアクセスします。

- gallery_tree.fla
- gallery_tween.fla

これらのファイルには、スクリプトアニメーションが組み込まれている SWF ファイルにイメージファイルをダウンロードする間に、ActionScript を使用してムービークリップを動的に制御する方法の例が含まれています。

1 つのムービークリップでの複数メソッドの呼び出し

with ステートメントを使用すると、一度指定したムービークリップに対して、一連のメソッドをまとめて実行できます。with ステートメントは、ムービークリップだけでなく、すべての ActionScript オブジェクト (Array、Color、Sound など) に対して使用できます。

with ステートメントは、パラメータとしてムービークリップを取ります。指定したオブジェクトは、現在のターゲットパスの末尾に追加されます。with ステートメント内にネストされたすべてのアクションは、新しいターゲットパス、つまり、スコープ内で実行されます。たとえば次のスクリプトでは、donut.hole オブジェクトが with ステートメントに渡され、hole のプロパティが変更されます。

```
with (donut.hole) {
  _alpha = 20;
  _xscale = 150;
  _yscale = 150;
}
```

with ステートメント内の各ステートメントは、hole インスタンスのタイムラインから呼び出されたときと同じように動作します。上記のコードは、次のコードと同じです。

```
donut.hole._alpha = 20;
donut.hole._xscale = 150;
donut.hole._yscale = 150;
```

上記のコードは、次のコードとも同じです。

```
with (donut) {
  hole._alpha = 20;
  hole._xscale = 150;
  hole._yscale = 150;
}
```

SWF ファイルのロードおよびアンロード

Flash Player を閉じずに追加の SWF ファイルを再生する場合や、別の HTML ページをロードせずに SWF ファイルを切り替える場合は、次のオプションを使用できます。

- MovieClip クラスの loadMovie() グローバル関数または loadMovie() メソッド。
- MovieClipLoader クラスの loadClip() メソッド。MovieClipLoader クラスの詳細については、『ActionScript 2.0 リファレンスガイド』の「MovieClipLoader」を参照してください。

loadMovie() メソッドは、CGI スクリプトに変数を渡して、SWF ファイルを CGI 出力として生成する場合にも使用できます。たとえば、この手順は、ムービークリップ内の指定した変数に基づいた動的 SWF ファイルまたはイメージファイルのロードに使用できます。SWF ファイルをロードする場合は、SWF ファイルがロードされるレベルまたはムービークリップターゲットを指定できます。ターゲットに SWF ファイルをロードすると、ロードされた SWF ファイルはターゲットのムービークリップのプロパティを継承します。Flash ムービーがロードされたら、これらのプロパティを変更することができます。

unloadMovie() メソッドは、loadMovie() メソッドによって以前にロードされた SWF ファイルを削除します。unloadMovie() で明示的に SWF ファイルをアンロードすると、SWF ファイル間の移動がスムーズになり、Flash Player で必要なメモリも少なくて済みます。場合によっては、クリップをアンロードするよりもムービークリップの _visible プロパティを false に設定するほうが効率的なことがあります。後でクリップをまた使用する場合は、_visible プロパティを false に設定しておいて、必要なときに true に設定します。

loadMovie() では、次のことが実現可能です。

- SWF バナーファイルを連続してロードおよびアンロードするようにコンテナ SWF ファイルに loadMovie() 関数を配置して、複数の SWF ファイルとして作成されている各バナー広告を連続して再生する。
- 複数の SWF ファイルの中からユーザーが選択してサイトのコンテンツを表示できるようにするリンクを持つ、分岐インターフェイスを開発する。
- コンテンツを他のレベルにロードするナビゲーションコントロールをレベル 0 に配置した、ナビゲーションインターフェイスを作成する。コンテンツをレベルにロードすると、ブラウザに新しい HTML ページをロードするよりもコンテンツのページ間の移動がよりスムーズになります。

SWC ファイルのロードに関する詳細については、533 ページの「外部 SWF ファイルおよびイメージファイルのロード」を参照してください。

詳細については、次のトピックを参照してください。

- 305 ページの「ロードした SWF ファイルのルートタイムラインの指定」
- 307 ページの「ムービークリップへのイメージファイルのロード」

ロードした SWF ファイルのルートタイムラインの指定

_root という ActionScript プロパティは、SWF ファイルのルートタイムラインへの参照を指定するか、持っています。SWF ファイルに複数のレベルがある場合、ルートタイムラインは現在実行中のスクリプトを含むレベルにあります。たとえば、レベル 1 のスクリプトの評価が _root であれば、_level1 が返されます。ただし、ムービーが独立して動作している (つまり独自のレベルにある) 場合と、SWF ファイルが loadMovie() 呼び出しによってムービークリップインスタンス内にロードされた場合とでは、_root が意味するタイムラインが異なることがあります。

たとえば、"container.swf" という名前のファイルがあり、そのメインタイムラインに target_mc という名前のムービークリップインスタンスが配置されているとします。"container.swf" ファイルでは、userName という変数をメインタイムラインで宣言します。その後、同じスクリプト内で、"contents.swf" という別のファイルを target_mc ムービークリップインスタンスにロードします。

```
// "container.swf" での操作
_root.userName = "Tim";
target_mc.loadMovie("contents.swf");
my_btn.onRelease = function():Void {
  trace(_root.userName);
};
```

次の例では、ロードした SWF ファイル "contents.swf" でも、そのルートタイムラインで userName という変数を宣言します。

```
// "content.swf" での操作
_root.userName = "Mary";
```

"contents.swf" を "container.swf" のムービークリップにロードすると、ロード先の SWF ファイル ("container.swf") のルートタイムラインに割り当てられている変数 userName の値は、"Tim" ではなく "Mary" に設定されてしまいます。この場合、"contents.swf" のコードはもちろん、"container.swf" のコードも、正しく機能しません。

_root プロパティが、実際のルートタイムラインではなく、必ずロードした SWF ファイルのタイムラインを評価するようにするには、_lockroot プロパティを使用します。このプロパティは、ロードされる SWF ファイル内から設定することも、ロードを開始する SWF ファイルに設定することもできます。ムービークリップインスタンスで _lockroot を true に設定すると、そのムービークリップは、その中にロードされる SWF ファイルの _root として機能します。SWF 内で _lockroot を true に設定した場合、他の SWF ファイルがその SWF ファイルをロードしていたとしても、その SWF ファイルは必ずそれ自体のルートになります。_lockroot を true に設定できるムービークリップの種類または数に制限はありません。このプロパティのデフォルト値は、false です。

たとえば、"container.swf" の作成者がメインタイムラインのフレーム 1 に次のコードを配置したとします。

```
// "container.swf" のフレーム 1 に追加 :
target_mc._lockroot = true;
```

これによって、"contents.swf" の _root へのすべての参照、つまり target_mc にロードされたすべての SWF ファイルが、"container.swf" の実際のルートタイムラインではなく、その SWF ファイル自体のタイムラインを参照します。この時点でボタンをクリックすると、"Tim" が表示されます。

または、"contents.swf" の作成者が次のコードをメインタイムラインに追加した場合を考えてみます。

```
// "contents.swf" のフレーム 1 に追加 :
this._lockroot = true;
```

この場合、"contents.swf" をどこにロードしても、それが _root を参照すると、ロード先の SWF ファイルのメインタイムラインではなく、"contents.swf" 自体のメインタイムラインを参照することが保証されます。

詳細については、「_lockroot (MovieClip._lockroot property)」を参照してください。

ムービークリップへのイメージファイルのロード

loadMovie() 関数または同じ名前の MovieClip メソッドを使用すると、イメージファイルをムービークリップインスタンスにロードすることができます。また、loadMovieNum() 関数を使用して、イメージファイルをレベルにロードすることもできます。

イメージをムービークリップにロードすると、イメージの左上隅がムービークリップの基準点に配置されます。この基準点は通常はムービークリップの中心にあるので、ロードされたイメージは中央配置で表示されない場合があります。また、ルートタイムラインにイメージをロードすると、イメージの左上隅はステージの左上隅に配置されます。ロードされたイメージはムービークリップの回転と伸縮を継承しますが、ムービークリップの元のコンテンツは削除されます。

詳細については、『ActionScript 2.0 リファレンスガイド』の「loadMovie 関数」、「loadMovie (MovieClip.loadMovie メソッド)」および「loadMovieNum 関数」や、533 ページの「外部 SWF ファイルおよびイメージファイルのロード」を参照してください。

ムービークリップの位置と画質の変更

ムービークリップの再生時にそのプロパティを変更するには、プロパティに値を割り当てるステートメントを記述するか、setProperty() 関数を使用します。たとえば、次のコードはインスタンス mc の回転を 45 に設定します。

```
my_mc._rotation = 45;
```

これは、getProperty() 関数を使用する次のコードと同じです。

```
setProperty("my_mc", _rotation, 45);
```

一部のプロパティは、読み取り専用プロパティであり、その値を読み取ることはできても設定はできません。そのようなプロパティは、『ActionScript 2.0 リファレンスガイド』の項目で読み取り専用と記述されています。_currentframe、_droptarget、_framesloaded、_parent、_target、_totalframes、_url、_xmouse、および _ymouse の各プロパティは、読み取り専用です。

読み取り専用でないプロパティを設定するためのステートメントを記述することができます。次のステートメントは、car_mc インスタンスの子であるムービークリップインスタンス wheel_mc の _alpha プロパティを設定します。

```
car_mc.wheel_mc._alpha = 50;
```

また、ムービークリッププロパティの値を取得するステートメントを記述することもできます。たとえば、次のステートメントは、現在のレベルのタイムラインの _xmouse プロパティの値を取得し、my_mc インスタンスの _x プロパティをその値に設定します。

```
this.onEnterFrame = function() {
  my_mc._x = _root._xmouse;
};
```

これは、getProperty() 関数を使用する次のコードと同じです。

```
this.onEnterFrame = function() {
  my_mc._x = getProperty(_root, _xmouse);
};
```

_x、_y、_rotation、_xscale、_yscale、_height、_width、_alpha、および _visible の各プロパティには、ムービークリップの親に対して行われた変形処理が反映されるため、ムービークリップとそのクリップの子すべてが同様に変形されます。_focusrect、_highquality、_quality、および _soundbuftime の各プロパティはグローバルです。つまり、レベル 0 のメインタイムラインだけに属します。他のプロパティはすべて、各ムービークリップまたはロードされたレベルに属します。

ムービークリッププロパティの一覧については、『ActionScript 2.0 リファレンスガイド』の MovieClip クラスのプロパティ一覧を参照してください。

スクリプトで作成された Flash のアニメーションを示すサンプルのソースファイル "animation.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Animation" フォルダに移動してサンプルにアクセスします。

フォトギャラリアプリケーションのサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Galleries" フォルダに移動して次のサンプルにアクセスします。

- gallery_tree.fla
- gallery_tween.fla

ムービークリップのドラッグ

startDrag() グローバル関数または MovieClip.startDrag() メソッドを使用すると、ムービークリップをドラッグ可能にできます。たとえば、ゲーム、ドラッグ & ドロップ機能、カスタマイズ可能なインターフェイス、スクロールバー、およびスライダ用に、ドラッグ可能なムービークリップを使用します。

stopDrag() でドラッグを明示的に停止するか、startDrag() で他のムービークリップがターゲットにされるまで、ムービークリップはドラッグ可能な状態のままです。SWF ファイル内でドラッグできるムービークリップは 1 つのみです。

ドラッグしているムービークリップの _droptarget プロパティを評価すると、さらに複雑なドラッグ & ドロップの動作を作成できます。たとえば、_droptarget プロパティを調べ、ムービークリップが特定のムービークリップ上 (「ごみ箱」ムービークリップなど) にドラッグされたかどうかを確認した後で、別のアクションを実行することができます。

```
// ごみをドラッグする。
garbage_mc.onPress = function() {
  this.startDrag(false);
};
// ごみ箱にごみをドラッグすると、ごみを見えなくする。
garbage_mc.onRelease = function() {
  this.stopDrag();
  // eval を使用してスラッシュ表記をドット表記に変換する。
  if (eval(this._droptarget) == trashcan_mc) {
    garbage_mc._visible = false;
  }
};
```

詳細については、『ActionScript 2.0 リファレンスガイド』の「startDrag 関数」または「startDrag (MovieClip.startDrag メソッド)」を参照してください。

ActionScript を使用して、各ムービークリップをドラッグ可能にする SWF ファイルにイメージファイルをロードしながらムービークリップを動的に制御する方法を示す、サンプルのソースファイル "gallery_tween.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Galleries" フォルダに移動してサンプルにアクセスします。

実行時のムービークリップの作成

ムービークリップのインスタンスは、Flash オーサリング環境で作成するだけでなく、次のように実行時に作成することもできます。

- 310 ページの「空のムービークリップの作成」
- 311 ページの「ムービークリップの複製または削除」
- 311 ページの「ステージへのムービークリップシンボルの割り当て」

実行時に作成する各ムービークリップインスタンスには、インスタンス名と深度 (重ね順、つまり z 順序空間) を指定する必要があります。ここで指定する深度によって、新しく作成するムービークリップが、同じタイムラインの他のムービークリップとどのように重なり合うかが決まります。同じ深度に指定して、ムービークリップを上書きすることもできます。詳細については、314 ページの「ムービークリップの深度の管理」を参照してください。

ActionScript を使用して、SWF ファイルにイメージファイルをロードしながらムービークリップを動的に制御する方法を示す、サンプルのソースファイル "gallery_tween.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Galleries" フォルダに移動してサンプルにアクセスします。

実行時に多数のムービークリップを作成および削除する方法を示す、サンプルのソースファイル "animation.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Animation" フォルダに移動してサンプルにアクセスします。

詳細については、次のトピックを参照してください。

- 310 ページの「空のムービークリップの作成」
- 311 ページの「ムービークリップの複製または削除」
- 311 ページの「ステージへのムービークリップシンボルの割り当て」

空のムービークリップの作成

ステージに新しく空のムービークリップインスタンスを作成するには、MovieClip クラスの `createEmptyMovieClip()` メソッドを使用します。このメソッドは、実行されたムービークリップの子としてムービークリップを作成します。新しく作成された空のムービークリップの基準点は左上隅です。

たとえば、次のコードは、`parent_mc` というムービークリップの深度 10 に、`new_mc` という名前の新しい子ムービークリップを作成します。

```
parent_mc.createEmptyMovieClip("new_mc", 10);
```

次のコードは、まず、スクリプトが実行される SWF ファイルのルートタイムラインに `canvas_mc` という名前の新しいムービークリップを作成し、次に `loadMovie()` を呼び出して外部の JPEG ファイルをその新しいムービークリップにロードします。

```
this.createEmptyMovieClip("canvas_mc", 10);
canvas_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

次の例が示すように、"image2.jpg" イメージをムービークリップにロードし、`MovieClip.onPress()` メソッドを使用して、イメージをボタンのように動作させることができます。`loadMovie()` を使用してイメージをロードすると、そのイメージでムービークリップが置き換えられますが、ムービークリップメソッドにはアクセスできません。ムービークリップにアクセスするには、空の親ムービークリップと子のコンテナムービークリップを作成する必要があります。イメージをコンテナにロードし、親ムービークリップにイベントハンドラを配置します。

```
// コンテナを持つ親ムービークリップを作成する。
this.createEmptyMovieClip("my_mc", 0);

// "my_mc" 内に子ムービークリップを作成する。
// これがイメージを置き換えるムービークリップになる。
my_mc.createEmptyMovieClip("container_mc",99);

// MovieClipLoader を使用してイメージをロードする。
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image2.jpg",
   my_mc.container_mc);

// イベントハンドラを親ムービークリップ my_mc に配置する。
my_mc.onPress = function():Void {
   trace("It works");
};
```

詳細については、『ActionScript 2.0 リファレンスガイド』の「createEmptyMovieClip (`MovieClip.createEmptyMovieClip` メソッド)」を参照してください。

実行時に多数のムービークリップを作成および削除する方法を示す、サンプルのソースファイル "animation.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Animation" フォルダに移動してサンプルにアクセスします。

ムービークリップの複製または削除

ムービークリップのインスタンスを複製または削除するには、duplicateMovieClip() グローバル関数または removeMovieClip() グローバル関数、あるいは同名の MovieClip クラスのメソッドを使用します。duplicateMovieClip() メソッドは、既存のムービークリップインスタンスの新しいインスタンスを作成して、それに新しいインスタンス名と深度 (つまり z 順) を割り当てます。元のムービークリップが複製時に別のフレームにあった場合でも、複製されたムービークリップは必ずフレーム 1 からスタートし、タイムラインに配置されている以前に定義されたすべてのムービークリップよりも前にあります。

duplicateMovieClip() で作成したムービークリップを削除するには、removeMovieClip() を使用します。親のムービークリップが削除されると、複製されたムービークリップも削除されます。

詳細については、『ActionScript 2.0 リファレンスガイド』の「duplicateMovieClip 関数」および「removeMovieClip 関数」を参照してください。

実行時に多数のムービークリップを作成および削除する方法を示す、サンプルのソースファイル "animation.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Animation" フォルダに移動してサンプルにアクセスします。

ステージへのムービークリップシンボルの割り当て

実行時にムービークリップインスタンスを作成する最後の方法は、attachMovie() メソッドを使用することです。attachMovie() メソッドは、SWF ファイルのライブラリにあるムービークリップシンボルのインスタンスをステージに割り当てます。割り当てた新しいムービークリップは、割り当て先のムービークリップの子クリップになります。

ActionScript を使用してライブラリからムービークリップシンボルを割り当てるには、目的のシンボルを書き出し、一意のリンケージ識別子をそのシンボルに割り当てる必要があります。そのためには、[リンケージプロパティ] ダイアログボックスを使用します。

デフォルトでは、ActionScript で使用するために書き出されたすべてのムービークリップは、ムービークリップを含む SWF ファイルの最初のフレームよりも前にロードされます。これによって、最初のフレームの再生が遅れることがあります。エレメントにリンケージ識別子を割り当てると、そのエレメントを最初のフレームの前に追加するかどうかを指定することもできます。エレメントが最初のフレームに追加されない場合は、SWF ファイルの他のフレームにそのエレメントのインスタンスを含める必要があります。これを行わない場合、そのエレメントは SWF ファイルに書き込まれません。

ムービークリップにリンケージ識別子を割り当てるには :

1. [ウィンドウ]-[ライブラリ] を選択して、[ライブラリ] パネルを開きます。
2. [ライブラリ] パネルでムービークリップを選択します。
3. [ライブラリ] パネルのポップアップメニューから [リンケージ] を選択します。

   [リンケージプロパティ] ダイアログボックスが表示されます。

4. [リンケージ] で、[ActionScript に書き出し] を選択します。
5. [識別子] に、ムービークリップの ID を入力します。

   デフォルトでは、リンケージ識別子はシンボル名と同じになります。

   ここで、ムービークリップシンボルに ActionScript クラスを割り当てることもできます。この操作によって、ムービークリップが指定したクラスのメソッドとプロパティを継承できるようになります。詳細については、327 ページの「ムービークリップシンボルへのクラスの割り当て」を参照してください。
6. ムービークリップを最初のフレームの前にロードしたくない場合は、[最初のフレームに書き出し] オプションの選択を解除します。

   このオプションの選択を解除した場合は、ムービークリップのインスタンスを、利用可能にするタイムラインのフレームに配置します。たとえば、フレーム 10 に到達するまでムービークリップを参照しないというスクリプトを作成している場合、ムービークリップシンボルのインスタンスを、タイムラインのフレーム 10 かそれ以前のいずれかのフレームに配置します。
7. [OK] をクリックします。

ムービークリップにリンケージ識別子を割り当てた後、`attachMovie()` を使用して実行時にムービークリップシンボルのインスタンスをステージに割り当てることができます。

ムービークリップを別のムービークリップに割り当てるには :

1. 前述の手順に従って、リンケージ識別子をムービークリップライブラリシンボルに割り当てます。
2. [ウィンドウ]-[アクション] を選択して [アクション] パネルを開き、タイムラインでフレームを 1 つ選択します。
3. [アクション] パネルのスクリプトペインで、新しいムービークリップを割り当てたいムービークリップの名前またはレベルを入力します。

   たとえば、ムービークリップをルートタイムラインに割り当てるには、「this」と入力します。
4. [アクション] パネルの左側にある [アクション] ツールボックスで、[ActionScript 2.0 クラス]-[ムービー]-[MovieClip]-[メソッド] を選択して、`attachMovie()` を選択します。
5. ガイドとして表示されるコードヒントを使用して、次のパラメータに値を入力します。
   - [`idName`] には、[リンケージプロパティ] ダイアログボックスで入力した識別子を指定します。
   - [`newName`] には、割り当てられたクリップをターゲットにすることができるように、そのクリップのインスタンス名を入力します。
   - [`depth`] には、複製されたムービークリップがムービークリップに割り当てられるときの深度を入力します。割り当てられた各ムービークリップには独自の重ね順があり、レベル 0 が元のムービークリップのレベルです。割り当てられたムービークリップは、次の例のように、常に元のムービークリップの一番上にあります。

     ```
     this.attachMovie("calif_id", "california_mc", 10);
     ```

詳細については、『ActionScript 2.0 リファレンスガイド』の「attachMovie (`MovieClip.attachMovie` メソッド)」を参照してください。

動的に作成したムービークリップへのパラメータの追加

MovieClip.attachMovie() および MovieClip.duplicateMovie() を使用して動的にムービークリップを作成または複製した場合、別のオブジェクトからそのムービークリップにパラメータを埋め込むことができます。動的に作成したムービークリップでクリップパラメータを取り込むには、attachMovie() および duplicateMovie() の *initObject* パラメータを使用します。

詳細については、『ActionScript 2.0 リファレンスガイド』の「attachMovie (MovieClip.attachMovie メソッド)」および「duplicateMovieClip (MovieClip.duplicateMovieClip メソッド)」を参照してください。

動的に作成したムービークリップに、特定のオブジェクトに基づいてパラメータを設定するには :

次のいずれかの操作を行います。

- 次のシンタックスで attachMovie() を使用します。

  myMovieClip.attachMovie(*idName*, *newName*, *depth* [, *initObject*]);

- 次のシンタックスで duplicateMovie() を使用します。

  myMovieClip.duplicateMovie(*idName*, *newName*, *depth* [, *initObject*]);

initObject パラメータには、動的に作成したムービークリップに埋め込むためのパラメータを持つオブジェクトの名前を指定します。

ムービークリップに attachMovie() を使用してパラメータを設定するには :

1. 新しい Flash ドキュメントで、[挿入]-[新規シンボル] を選択してムービークリップシンボルを作成します。
2. [シンボル名] テキストボックスに「dynamic_mc」と入力し、ビヘイビアとして [ムービークリップ] を選択します。
3. このシンボル内で、ステージに動的テキストフィールドを作成します。インスタンス名は、name_txt とします。

   このテキストフィールドが下にあり、かつ基準点の右側にあることを確認します。
4. ムービークリップのタイムラインのフレーム 1 を選択します。[ウィンドウ]-[アクション] を選択して、[アクション] パネルを開きます。
5. name_str という名前で新しい変数を作成し、その変数の値を name_txt の text プロパティに代入します。例を次に示します。

   ```
   var name_str:String;
   name_txt.text = name_str;
   ```
6. [編集]-[ドキュメントの編集] を選択して、メインタイムラインに戻ります。

7. ムービークリップシンボルをライブラリから選択し、[ライブラリ] パネルのポップアップメニューから [リンケージ] を選択します。

   [リンケージプロパティ] ダイアログボックスが表示されます。

8. [ActionScript に書き出し] を選択し、最初のフレームで [書き出し] を選択します。

9. [識別子] テキストボックスに「dynamic_id」と入力し、[OK] をクリックします。

10. メインタイムラインの先頭フレームを選択し、次のコードを [アクション] パネルのスクリプトペインに追加します。

```
/* 新しいムービークリップを割り当て、x 座標 50、y 座標 50 に移動する */
this.attachMovie("dynamic_id", "newClip_mc", 99, {name_str:"Erick", _x:50,
  _y:50});
```

11. [制御]-[ムービープレビュー] を選択して、Flash ドキュメントをテストします。

    `attachMovie()` メソッドを呼び出したときに指定した名前が、新しいムービークリップのテキストフィールドに表示されます。

ActionScript を使用して、各ムービークリップをドラッグ可能にする SWF ファイルにイメージファイルをロードしながらムービークリップを動的に制御する方法を示す、サンプルのソースファイル "gallery_tween.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Galleries" フォルダに移動してサンプルにアクセスします。

実行時に多数のムービークリップを作成および削除する方法を示す、サンプルのソースファイル "animation.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Animation" フォルダに移動してサンプルにアクセスします。

ムービークリップの深度の管理

すべてのムービークリップは、それぞれ固有の z 順序空間を持ち、これによって、親 SWF またはムービークリップ内でオブジェクトがどのように重なり合うかが決まります。ムービークリップには深度が関連付けられています。この深度によって、同じムービークリップタイムラインに配置された他のムービークリップよりも前面にレンダリングされるのか、後ろにレンダリングされるのかが決まります。`attachMovie` (`MovieClip.attachMovie` メソッド) や `duplicateMovieClip` (`MovieClip.duplicateMovieClip` メソッド)、`createEmptyMovieClip` (`MovieClip.createEmptyMovieClip` メソッド) を使用して実行時にムービークリップを作成する場合、その新しいムービークリップの深度をパラメータとして指定する必要があります。たとえば、次のコードは、`container_mc` という名前のムービークリップのタイムラインに、新しいムービークリップを深度 10 で割り当てます。

```
container_mc.attachMovie("symbolID", "clip1_mc", 10);
```

これにより、container_mc の z 順序空間内に深度 10 のムービークリップが新しく作成されます。

たとえば、次のコードでは、container_mc に新しいムービークリップを 2 つ割り当てます。1 番目の clip1_mc は、2 番目の clip2_mc よりも深度が小さいので、後ろにレンダリングされます。

```
container_mc.attachMovie("symbolID", "clip1_mc", 10);
container_mc.attachMovie("symbolID", "clip2_mc", 15);
```

ムービークリップに指定できる深度の範囲は、-16384 から 1048575 までです。ムービークリップを既に持つ深度に新しいムービークリップを作成するか割り当てると、既存のコンテンツが新しいまたは割り当てられたムービークリップによって上書きされます。この問題が発生するのを防ぐには、MovieClip.getNextHighestDepth() メソッドを使用します。ただし、別の深度管理システムが使われているコンポーネントに対してはこのメソッドを使用しないでください。その場合は、DepthManager クラスをコンポーネントインスタンスと共に使用してください。詳細については、『ActionScript 2.0 リファレンスガイド』を参照してください。

MovieClip クラスには、ムービークリップの深度を管理するためのメソッドがいくつか用意されています。詳細については、『ActionScript 2.0 リファレンスガイド』の「getNextHighestDepth (MovieClip.getNextHighestDepth メソッド)」、「getInstanceAtDepth (MovieClip.getInstanceAtDepth メソッド)」、「getDepth (MovieClip.getDepth メソッド)」および「swapDepths (MovieClip.swapDepths メソッド)」を参照してください。

ムービークリップの深度の詳細については、次のトピックを参照してください。

- 315 ページの「利用可能な上から 2 番目の深度の確認」
- 316 ページの「特定の深度にあるインスタンスの確認」
- 316 ページの「インスタンスの深度の確認」
- 317 ページの「ムービークリップの深度の入れ替え」

利用可能な上から 2 番目の深度の確認

ムービークリップ内で利用可能な上から 2 番目の深度を調べるには、MovieClip.getNextHighestDepth() を使用します。このメソッドから返される整数値は、ムービークリップ内の他のすべてのオブジェクトよりも前にレンダリングされる、利用可能な 2 番目の深度を示します。

次のコードは、file_mc というルートタイムライン上で、新しいムービークリップに深度 10 を割り当てています。次に、その同じムービークリップで利用可能な上から 2 番目の深度を判断して、その深度で edit_mc という新しいムービークリップを作成します。

```
this.attachMovie("menuClip","file_mc", 10, {_x:0, _y:0});
trace(file_mc.getDepth()); // 10
var nextDepth:Number = this.getNextHighestDepth();
this.attachMovie("menuClip", "edit_mc", nextDepth, {_x:200, _y:0});
trace(edit_mc.getDepth()); // 11
```

この例の場合、ムービークリップ edit_mc の利用可能な上から 2 番目の深度は 11 なので、変数 nextDepth には 11 が入ります。

コンポーネントで MovieClip.getNextHighestDepth() を使用しないでください。代わりに深度マネージャを使用します。詳細については、『ActionScript 2.0 リファレンスガイド』の「DepthManager クラス」を参照してください。MovieClip.getNextHighestDepth() の詳細については、「getNextHighestDepth (MovieClip.getNextHighestDepth メソッド)」を参照してください。

現時点で一番上の深度を取得するには、次のセクションに示すように、getNextHighestDepth() メソッドの戻り値から 1 を減算します。

特定の深度にあるインスタンスの確認

特定の深度にあるインスタンスを調べるには、MovieClip.getInstanceAtDepth() を使用します。このメソッドからは、指定された深度にある MovieClip インスタンスへの参照が返されます。

次のコードでは、getNextHighestDepth() と getInstanceAtDepth() を使用して、ルートタイムラインで (現在) 一番上の深度のムービークリップを調べています。

```
var highestOccupiedDepth:Number = this.getNextHighestDepth() - 1;
var instanceAtHighestDepth:MovieClip =
  this.getInstanceAtDepth(highestOccupiedDepth);
```

詳細については、『ActionScript 2.0 リファレンスガイド』の「getInstanceAtDepth (MovieClip.getInstanceAtDepth メソッド)」を参照してください。

インスタンスの深度の確認

ムービークリップインスタンスの深度を調べるには、MovieClip.getDepth() を使用します。

次のコードでは、SWF ファイルのメインタイムラインにあるすべてのムービークリップを反復処理して、各ムービークリップのインスタンス名と深度を [出力] パネルに表示しています。

```
for (var item:String in _root) {
  var obj:Object = _root[item];
  if (obj instanceof MovieClip) {
    var objDepth:Number = obj.getDepth();
    trace(obj._name + ":" + objDepth)
  }
}
```

詳細については、『ActionScript 2.0 リファレンスガイド』の「getDepth (MovieClip.getDepth メソッド)」を参照してください。

ムービークリップの深度の入れ替え

同じタイムライン上の 2 つのムービークリップの深度を入れ替えるには、MovieClip.swapDepths() メソッドを使用します。次の例では、2 つのムービークリップインスタンスの深度を実行時にどのように入れ替えるかが示されています。

ムービークリップの深度を入れ替えるには :

1. 新しい Flash ドキュメントを「swap.fla」という名前で作成します。
2. ステージに青い円を描画します。
3. 描画した青い円を選択し、[修正]-[シンボルに変換] を選択します。
4. [ムービークリップ] オプションを選択し、[OK] をクリックします。
5. ステージ上でインスタンスを選択し、プロパティインスペクタの [インスタンス名] テキストボックスに「first_mc」と入力します。
6. ステージに赤い円を描画し、[修正]-[シンボルに変換] を選択します。
7. [ムービークリップ] オプションを選択し、[OK] をクリックします。
8. ステージ上でインスタンスを選択し、プロパティインスペクタの [インスタンス名] テキストボックスに「second_mc」と入力します。
9. 2 つのインスタンスを、ステージ上でわずかに重なるようにドラッグします。
10. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
first_mc.onRelease = function() {
     this.swapDepths(second_mc);
  };
  second_mc.onRelease = function() {
     this.swapDepths(first_mc);
  };
```

11. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

    ステージでインスタンスをクリックすると、深度が入れ替わります。どちらのクリップがもう一方のクリップの上にくるかで、2 つのインスタンスが変わったのがわかります。

詳細については、『ActionScript 2.0 リファレンスガイド』の「swapDepths (MovieClip.swapDepths メソッド)」を参照してください。

ActionScript を使用したムービークリップのキャッシュとスクロールについて

Flash 内のデザインのサイズが大きくなるにつれて、アプリケーションを作成している場合であれ、複雑なスクリプトで記述されたアニメーションを作成している場合であれ、パフォーマンスや最適化について考慮する必要が生じてきます。静的な状態を維持するコンテンツ (矩形のムービークリップなど) がある場合、Flash ではコンテンツが最適化されません。そのため、矩形のムービークリップの位置を変更した場合、Flash Player 7 以前では、矩形全体が再描画されます。

Flash 8 以降では、指定したムービークリップおよびボタンをキャッシュして、SWF ファイルのパフォーマンスを向上させることができます。ムービークリップやボタンは、" サーフェス " であり、基本的にはインスタンスのベクターデータのビットマップバージョンです。このデータは、SWF ファイルの処理中に大きく変更されることはありません。そのため、キャッシュがオンになったインスタンスは、SWF ファイルの再生中に頻繁に再描画されないので、SWF ファイルをすばやくレンダリングできます。

メモ	サーフェスが再作成されるときに、ベクターデータを更新できます。そのため、サーフェスにキャッシュされたベクターデータは、SWF ファイル全体を通して同じになっている必要はありません。

ActionScript を使用してキャッシュとスクロールを有効にし、背景を制御することができます。プロパティインスペクタを使用して、ムービークリップインスタンスのキャッシュを有効にすることができます。ActionScript を使用せずにムービークリップやボタンをキャッシュするには、プロパティインスペクタで [ランタイムビットマップキャッシュを使用する] オプションを選択します。

次の表では、ムービークリップインスタンスの新しいプロパティについて簡単に説明されています。

プロパティ	説明
cacheAsBitmap	ムービークリップインスタンスにおいて、ムービークリップ自体のビットマップ表現がキャッシュされます。Flash が、インスタンスのサーフェスオブジェクトを作成します。これは、ベクターデータではなくキャッシュされたビットマップです。ムービークリップの境界を変更した場合、サーフェスのサイズが変更される代わりにサーフェスが再作成されます。詳細および使用例については、322 ページの「ムービークリップのキャッシュ」を参照してください。
opaqueBackground	不透明なムービークリップインスタンスの背景色を指定できます。このプロパティに数値を設定した場合、ムービークリップインスタンスのサーフェスが不透明になります。不透明のビットマップには、アルファチャンネル (透明度) がないので、すばやくレンダリングされます。詳細および使用例については、324 ページの「ムービークリップの背景の設定」を参照してください。

プロパティ	説明
scrollRect	ムービークリップコンテンツをすばやくスクロールし、ウィンドウよりも大きなコンテンツをウィンドウに表示できます。ムービークリップのコンテンツがトリミングされ、指定した幅、高さ、およびスクロールオフセットを使用してインスタンスがスクロールします。これにより、ユーザーが、ムービークリップコンテンツをすばやくスクロールし、ステージ領域よりも大きなコンテンツをウィンドウに表示することができます。Flash では、ムービークリップ全体のベクターデータが再生成されないので、インスタンス内に表示するテキストフィールドや複雑なコンテンツを速くスクロールすることができます。詳細および使用例については、「scrollRect (MovieClip.scrollRect プロパティ)」を参照してください。

これらの 3 つのプロパティは互いに独立していますが、opaqueBackground プロパティと scrollRect プロパティは、オブジェクトがビットマップとしてキャッシュされているときに最もよく機能します。opaqueBackground プロパティおよび scrollRect プロパティの使用によるパフォーマンスの向上は、cacheAsBitmap を true に設定した場合にのみ現れます。

スクロール可能なサーフェスを作成するには、ムービークリップインスタンスの cacheAsBitmap プロパティおよび scrollRect プロパティを設定する必要があります。サーフェスは、他のサーフェス内にネストすることができます。サーフェスは、親のサーフェス上にビットマップをコピーします。

アルファチャンネルのマスクでは cacheAsBitmap プロパティを true に設定する必要がありますが、このアルファチャンネルのマスクの詳細については、326 ページの「アルファチャンネルマスクについて」を参照してください。

メモ	キャッシュをテキストフィールドに直接適用することはできません。この機能を利用するには、ムービークリップ内にテキストを配置する必要があります。サンプルファイルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/CacheBitmap" フォルダに移動してサンプルにアクセスします。

インスタンスとスクロールテキストに対してビットマップキャッシュを適用する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。次のサンプルを使用できます。

- cacheBitmap.fla。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/CacheBitmap" フォルダに移動します。
- aliasing.fla。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Advanced Anti-Aliasing" フォルダに移動します。

キャッシュを有効にする状況

ムービークリップのキャッシュを有効にすると、サーフェスが作成されます。これにより、複雑なベクターアニメーションのレンダリング速度の向上などのいくつかの利点が得られます。キャッシュを有効にする必要があるシナリオはいくつか考えられます。常にキャッシュを有効にして SWF ファイルのパフォーマンスを向上させたいと思われるかもしれません。しかし、キャッシュを有効にしてもパフォーマンスが向上しない、またはかえってパフォーマンスが低下する場合もあります。このセクションでは、キャッシュを使用する必要があるシナリオ、および通常のムービークリップを使用する必要があるシナリオについて説明します。

キャッシュされたデータの全体のパフォーマンスは、インスタンスのベクターデータの複雑さ、変更するデータの量、および `opaqueBackground` プロパティを設定するかどうかによって異なります。小さな領域を変更する場合、サーフェスを使用してもベクターデータを使用してもほとんど違いはありません。アプリケーションを展開する前に、両方のシナリオで作業内容をテストしてください。

アルファチャンネルのマスクでは `cacheAsBitmap` プロパティを `true` に設定する必要がありますが、このアルファチャンネルのマスクの詳細については、326 ページの「アルファチャンネルマスクについて」を参照してください。

ビットマップキャッシュを使用する状況

次に、ビットマップキャッシュを有効にした場合に大きな利点が得られる一般的なシナリオを示します。

複雑な背景イメージ　ベクターデータによる詳細で複雑な背景イメージ ([ビットマップのトレース] コマンドを適用したイメージや、Adobe Illustrator で作成したアートワークなど) を使用するアプリケーション。背景で文字をアニメーションさせる場合。この場合、背景で連続的にベクターデータを再生成する必要があるため、アニメーションが遅くなります。パフォーマンスを改善するには、コンテンツを選択してムービークリップ内に保存し、`opaqueBackground` プロパティを `true` に設定します。これにより、背景はビットマップとしてレンダリングされるのですばやく再描画でき、アニメーションの再生がより高速になります。

テキストフィールドのスクロール　スクロールテキストフィールドに大量のテキストを表示するアプリケーション。スクロール範囲を (`scrollRect` プロパティで) 指定してスクロール可能に設定されたムービークリップ内にテキストフィールドを配置できます。これにより、指定したインスタンスの最初のピクセルがスクロール可能となります。ユーザーがムービークリップインスタンスをスクロールすると、Flash はスクロールされたピクセルを上にシフトし、テキストフィールド全体を再生成するのではなく、新しく表示された部分を生成します。

ウィンドウシステム　ウィンドウの重ね合わせが複雑なアプリケーション。各ウィンドウは開いたり閉じたりできます (たとえば Web ブラウザウィンドウなど)。各ウィンドウをサーフェスとして指定する (`cacheAsBitmap` プロパティを `true` に設定する) と、各ウィンドウは分離されてキャッシュされます。ユーザーはウィンドウをドラッグできるため、互いに重ね合わせることが可能となり、各ウィンドウではベクターコンテンツの再生成は必要ありません。

このようなシナリオにおいては、ベクターグラフィックの最適化により、アプリケーションの応答性およびインタラクティビティが改善されます。

インスタンスとスクロールテキストに対してビットマップキャッシュを適用する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。次のサンプルを使用できます。

- cacheBitmap.fla。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Cachebitmap" フォルダに移動します。
- aliasing.fla。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Advanced Anti-Aliasing" フォルダに移動します。

ビットマップキャッシュを使用すべきでない状況

この機能を誤って使用すると、SWF ファイルに悪影響を与える可能性があります。サーフェスを使用する FLA ファイルを開発する場合は、次のガイドラインに注意してください。

- サーフェス (キャッシュを有効にしたムービークリップ) を使いすぎないようにしてください。各サーフェスは通常のムービークリップよりも多くメモリを使用するので、レンダリングのパフォーマンスを向上させる必要がある場合のみサーフェスを有効にしてください。

  キャッシュされたビットマップは、通常のムービークリップインスタンスよりはるかに多くのメモリを使用します。たとえば、ステージ上のムービークリップのサイズが 250 ピクセル x250 ピクセルの場合、通常の (キャッシュされていない) ムービークリップインスタンスで使用されるメモリは 1KB ですが、キャッシュされるとメモリ使用量が 250 KB に達することがあります。
- キャッシュされたサーフェスの拡大は避けてください。ビットマップキャッシュを多用しすぎると大量のメモリが消費され (前の項目を参照) ますが、コンテンツを拡大したときにそれは著しくなります。
- 大部分が静的な (アニメーションではない) ムービークリップインスタンスに対してサーフェスを使用します。インスタンスをドラッグまたは移動できますが、インスタンスのコンテンツは、アニメーション化したり大幅に変更しないようにする必要があります。たとえば、インスタンスを回転または変形させると、インスタンスはサーフェスからベクターデータへ、あるいはその逆へと変更されますが、これは難しい処理であり、SWF ファイルに悪影響を与えます。
- サーフェスとベクターデータを組み合わせると、Flash Player の処理量 (そして場合によっては CPU の処理量も) が増加します。たとえばウィンドウアプリケーションを作成するときなど、サーフェスをできる限りグループ化するようにしてください。

ムービークリップのキャッシュ

ムービークリップインスタンスをキャッシュするには、cacheAsBitmap プロパティを true に設定する必要があります。cacheAsBitmap プロパティを true に設定すると、ムービークリップインスタンスのピクセルが全体の座標に自動的に吸着します。SWF ファイルをテストすると、複雑なベクターアニメーションがより速くレンダリングされることがわかります。

cacheAsBitmap を true に設定しても、次のいずれかの条件に該当する場合は、サーフェス (キャッシュされたビットマップ) は作成されません。

- ビットマップの高さまたは幅が 2880 ピクセルより大きい。
- ビットマップが割り当てに失敗した場合 (メモリ不足エラー)。

ムービークリップをキャッシュするには :

1. 新しい Flash ドキュメントを作成し、「cachebitmap.fla」という名前を付けます。
2. プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) の [フレームレート] テキストボックスに「24」と入力します。
3. 複雑なベクターグラフィックを FLA ファイルに作成または読み込みます。

   複雑なベクターグラフィックである CacheBitmap のサンプルソースファイルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/CacheBitmap" フォルダに移動してサンプルにアクセスします。
4. ベクターグラフィックを選択し、[修正]-[シンボルに変換] を選択します。
5. [名前] テキストボックスに「star」と入力し、(ダイアログボックスが拡張されていなければ) [詳細] をクリックします。
6. [ActionScript に書き出し] を選択します (これにより、[最初のフレームに書き出し] も選択されます)。
7. [識別子] テキストボックスに「star_id」と入力します。
8. [OK] をクリックして、リンケージ識別子 Star が設定されたムービークリップシンボルを作成します。
9. タイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を追加します。

```
import mx.transitions.Tween;

var star_array:Array = new Array();
for (var i:Number = 0; i < 20; i++) {
  makeStar();
}
function makeStar():Void {
  var depth:Number = this.getNextHighestDepth();
  var star_mc:MovieClip = this.attachMovie("star_id", "star" + depth, depth);
  star_mc.onEnterFrame = function() {
```

```
        star_mc._rotation += 5;
      }
      star_mc._y = Math.round(Math.random() * Stage.height - star_mc._height /
      2);
      var star_tween:Tween = new Tween(star_mc, "_x", null, 0, Stage.width,
      (Math.random() * 5) + 5, true);
      star_tween.onMotionFinished = function():Void  {
        star_tween.yoyo();
      };
      star_array.push(star_mc);
    }
    var mouseListener:Object = new Object();
    mouseListener.onMouseDown = function():Void {
      var star_mc:MovieClip;
      for (var i:Number = 0; i < star_array.length; i++) {
        star_mc = star_array[i];
        star_mc.cacheAsBitmap = !star_mc.cacheAsBitmap;
      }
    }
    Mouse.addListener(mouseListener);
```

10. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。
11. ステージのどこかをクリックしてビットマップキャッシュを有効にします。

    1 秒間に 1 フレームくらいでアニメーション表示されていたのが、表示がスムーズになってインスタンスがステージ上のあちらこちらをアニメーションされるようになったことにお気づきでしょう。ステージをクリックすると、cacheAsBitmap の設定の true と false が切り替わります。

前の例で行ったようにキャッシュのオンとオフを切り替えると、キャッシュされたデータが解放されます。このコードは Button インスタンスに対しても適用できます。詳細については、『ActionScript 2.0 リファレンスガイド』の「cacheAsBitmap (Button.cacheAsBitmap プロパティ)」を参照してください。

ムービークリップのスクロールの例については、『ActionScript 2.0 リファレンスガイド』の「scrollRect (MovieClip.scrollRect プロパティ)」を参照してください。アルファチャンネルマスクでは cacheAsBitmap プロパティを true に設定する必要がありますが、このアルファチャンネルマスクの詳細については、326 ページの「アルファチャンネルマスクについて」を参照してください。

インスタンスとスクロールテキストに対してビットマップキャッシュを適用する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。次のサンプルを使用できます。

- cacheBitmap.fla。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/CacheBitmap" フォルダに移動します。
- aliasing.fla。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Advanced Anti-Aliasing" フォルダに移動します。

ムービークリップの背景の設定

ムービークリップに対しては不透明な背景を設定できます。たとえば、背景に複雑なベクターアートが使用されている場合、opaqueBackground プロパティを指定の色 (通常はステージと同じ色) に設定できます。これによって背景はビットマップとして扱われるため、パフォーマンスの最適化につながります。

cacheAsBitmap を true に設定し、opaqueBackground プロパティを指定の色に設定すると、opaqueBackground プロパティの設定によって内部ビットマップが不透明となり、レンダリングが高速になります。cacheAsBitmap を true に設定しなかった場合、opaqueBackground プロパティによって、ムービークリップインスタンスの背景に不透明な四角形のベクターシェイプが追加されます。ビットマップが自動的に作成されることはありません。

次の例は、ムービークリップの背景を設定してパフォーマンスを最適化する方法を示しています。

ムービークリップの背景を設定するには :

1. 新しい Flash ドキュメントを「background.fla」という名前で作成します。
2. ステージに青い円を描画します。
3. 描画した青い円を選択し、[修正]-[シンボルに変換] を選択します。
4. [ムービークリップ] オプションを選択し、[OK] をクリックします。
5. ステージ上でインスタンスを選択し、プロパティインスペクタの [インスタンス名] テキストボックスに「my_mc」と入力します。
6. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
/* cacheAsBitmap を設定すると、内部ビットマップが不透明になり、レンダリングが高速になる。
   */
my_mc.cacheAsBitmap = true;
my_mc.opaqueBackground = 0xFF0000;
```

7. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   ステージにムービークリップが、指定された背景色で表示されます。

このプロパティの詳細については、『ActionScript 2.0 リファレンスガイド』の「opaqueBackground (MovieClip.opaqueBackground プロパティ)」を参照してください。

マスクとしてのムービークリップの使用

ムービークリップをマスクとして使用して、他のムービークリップのコンテンツが見えるマスク形状を作成することができます。マスクムービークリップは、通常のムービークリップと同様に、タイムライン内のすべてのフレームを再生します。マスクムービークリップをドラッグ可能にしたり、モーションガイドに沿ってアニメーションしたり、1つのマスク内のシェイプを区分したり、マスクのサイズを動的に変更したりできます。また、ActionScript でマスクのオンとオフを切り替えることもできます。

マスクを使用して、他のマスクをマスクすることはできません。マスクムービークリップの `_alpha` プロパティを設定することはできません。ムービークリップで使用されている塗りだけがマスクとして使用されます。線は無視されます。

マスクを作成するには :

1. 矩形ツールを使用して、ステージ上に四角形を作成します。
2. 四角形を選択し、F8 キーを押してムービークリップに変換します。

   このインスタンスがマスクになります。
3. プロパティインスペクタで、[インスタンス名] テキストボックスに「mask_mc」と入力します。

   マスクされたムービークリップは、マスクとして機能するムービークリップのすべての不透明領域の下に表示されます。
4. タイムラインでフレーム 1 を選択します。
5. まだ開いていなければ、[ウィンドウ]-[アクション] を選択して [アクション] パネルを開きます。
6. [アクション] パネルに次のコードを入力します。

```
System.security.allowDomain("http://www.helpexamples.com");

this.createEmptyMovieClip("img_mc", 10);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
   target_mc.setMask(mask_mc);
}
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mclListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
   img_mc);
```

7. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   外部の JPEG イメージが実行時に SWF ファイルにロードされ、ステージ上に前に描画したシェイプによってマスクされます。

詳細については、『ActionScript 2.0 リファレンスガイド』の「setMask (`MovieClip.setMask` メソッド)」を参照してください。

デバイスフォントのマスクについて

ムービークリップを使用して、デバイスフォントに設定されているテキストをマスクすることができます。デバイスフォントに対してムービークリップマスクを正しく機能させるには、Flash Player 6 (6.0.40.0) 以降を使用する必要があります。

ムービークリップを使用して、デバイスフォントに設定されているテキストをマスクする場合、マスクの矩形の境界ボックスがマスクシェイプとして使用されます。つまり、Flash オーサリング環境でデバイスフォントテキストに対して長方形以外のムービークリップマスクを作成すると、SWF ファイルに表示されるマスクの形は、そのマスク自体の形ではなく、マスクの境界ボックスを表す長方形になります。

デバイスフォントをマスクできるのは、ムービークリップをマスクとして使用した場合だけです。ステージでマスクレイヤーを使用してデバイスフォントをマスクすることはできません。

アルファチャンネルマスクについて

アルファチャンネルマスクは、マスクとマスクされたムービークリップの両方でビットマップキャッシュが使用されている場合にサポートされます。このアルファチャンネルマスクのサポートにより、マスクされたムービークリップそのものに対して適用されるフィルタとは無関係に、マスクに対してフィルタを使用できます。

アルファチャンネルマスクの例を参照するには、www.adobe.com/go/learn_fl_samples_jp からアルファチャンネルマスクのサンプルファイルをダウンロードしてください。

このサンプルファイルでは、マスクはアルファ値が 50% の楕円 (`oval_mask`) で、ぼかしフィルタが適用されています。マスキー (`flower_maskee`) はアルファ値が 100% で、フィルタは適用されていません。どちらのムービークリップも、プロパティインスペクタでランタイムビットマップキャッシュが適用されています。

[アクション] パネルで、タイムラインのフレーム 1 に 次のコードが入力されています。

```
flower_maskee.setMask(oval_mask);
```

[制御]-[ムービープレビュー] でドキュメントをテストすると、マスクを使用してマスキーがアルファブレンドされます。

メモ	マスクレイヤーではアルファチャンネルマスクはサポートされていません。アルファチャンネルマスクを使用するには、マスクを適用する ActionScript コードを使用し、かつランタイムビットマップキャッシュを使用する必要があります。

ムービークリップイベントの処理

ムービークリップには、ユーザーイベントやシステムイベントに応答する機能があります。ユーザーイベントとはマウスクリックやキー入力などのことで、システムイベントとはステージへのムービークリップの初期ロードなどのことです。ActionScript には、ムービークリップイベントを処理する方法が 2 つ用意されています。イベントハンドラメソッドを使用する方法と、`onClipEvent()` および `on()` の各イベントハンドラを使用する方法です。ムービークリップイベントの処理の詳細については、第 9 章の「イベントの処理」を参照してください。

ムービークリップシンボルへのクラスの割り当て

ActionScript 2.0 では、MovieClip ビルトインクラスのビヘイビアを継承したクラスを作成し、[リンケージプロパティ] ダイアログボックスを使用してそのクラスをムービークリップライブラリシンボルに割り当てることができます。クラスが割り当てられるムービークリップのインスタンスを作成すると、そのインスタンスは常に割り当てられているクラスによってプロパティとビヘイビアが定義されることを前提とします。ActionScript 2.0 の詳細については、214 ページの「例 : カスタムクラスの記述」を参照してください。

MovieClip クラスのサブクラスには、MovieClip ビルトインメソッドの定義を用意することができます。また、`onEnterFrame` や `onRelease` などのイベントハンドラを用意することもできます。次の 1 番目の手順では、MovieClip クラスを拡張した MoveRight というクラスを作成します。MoveRight は、ユーザーがムービークリップをクリックするたびにそのムービークリップを 20 ピクセルだけ右方向に移動する `onPress` ハンドラを定義します。2 番目の手順では、新しい Flash (FLA) ドキュメントでムービークリップシンボルを作成し、前の手順で作成した MoveRight クラスをそのシンボルに割り当てます。

ムービークリップのサブクラスを作成するには :

1. BallTest という名前で新規ディレクトリを作成します。
2. [ファイル]-[新規] を選択し、ドキュメントタイプの一覧から ActionScript ファイルを選択し、新しい ActionScript ファイルを作成します。
3. 作成したスクリプトファイルに次のコードを入力します。

```
// MoveRight クラス -- ムービークリップの各フレームを右方向に移動する
class MoveRight extends MovieClip {
  public function onPress() {
    this._x += 20;
  }
}
```

4. このドキュメントを MoveRight.as という名前で "BallTest" ディレクトリに保存します。

ムービークリップシンボルにクラスを割り当てるには :

1. Flash で、[ファイル]-[新規作成] を選択します。ファイルタイプの一覧から [Flash ドキュメント] を選択し、[OK] をクリックします。
2. 楕円ツールを使用して、ステージに円を描画します。
3. その円を選択し、[修正]-[シンボルに変換] を選択します。
4. [シンボルに変換] ダイアログボックスで、シンボルのビヘイビアとして [ムービークリップ] を選択し、[名前] テキストボックスに「ball_mc」と入力します。
5. [リンケージ] のオプションが表示されていない場合は、[詳細] を選択して表示します。
6. [ActionScript に書き出し] オプションを選択し、[クラス] テキストボックスに「MoveRight」と入力します。[OK] をクリックします。
7. このファイルを "ball.fla" という名前で "BallTest" ディレクトリに保存します。"MoveRight.as" ファイルが収められているのと同じディレクトリです。
8. Flash ドキュメントをテストします ([制御]-[ムービープレビュー])。

   Ball ムービークリップを何度かクリックしてみてください。クリックするたびに、20 ピクセルだけ右方向に移動します。

クラスに対してコンポーネントプロパティを作成し、ムービークリップにそのコンポーネントプロパティを継承する場合は、さらに処理が必要です。[ライブラリ] パネルでムービークリップのシンボルを選択し、[ライブラリ] ポップアップメニューから [コンポーネント定義] を選択して [クラス] ボックスに新しいクラス名を入力します。

クラスプロパティの初期化

「ムービークリップシンボルへのクラスの割り当て」の 2 番目の手順で示されている例では、オーサリング時にステージに Ball シンボルのインスタンスを追加しました。313 ページの「動的に作成したムービークリップへのパラメータの追加」で説明したように、`attachMovie()` および `duplicateMovie()` の *`initObject`* パラメータを使用して、実行時に作成するムービークリップに対してパラメータを割り当てることができます。この機能を利用して、ムービークリップに割り当てるクラスのプロパティを初期化することができます。

次に、MoveRightDistance クラスの例を挙げます。このクラスは、MoveRight クラス (327 ページの「ムービークリップシンボルへのクラスの割り当て」を参照) のバリエーションです。相違点は、`distance` というプロパティを使用している点です。このプロパティの値によって、クリック時のムービークリップの移動ピクセル数が決まります。

カスタムクラスに引数を渡すには :

1. 新しい ActionScript ドキュメントを作成し、「MoveRightDistance.as」という名前で保存します。
2. 次の ActionScript をスクリプトウィンドウに入力します。

```
// MoveRightDistance クラス -- ムービークリップの各フレームを distance の値だけ右に移
   動する
class MoveRightDistance extends MovieClip {
   // distance プロパティの値によって
   // マウスクリックのたびにムービークリップを移動するピクセル数が決まる
   var distance:Number;
   function onPress() {
      this._x += this.distance;
   }
}
```

3. ここまでの作業内容を保存します。
4. 新しい Flash ドキュメントを作成し、MoveRightDistance.fla という名前でクラスファイルと同じディレクトリに保存します。
5. 楕円などのベクターシェイプが格納されたムービークリップシンボルを作成し、それからステージから他のコンテンツを削除します。

   この例では、ライブラリ内に必要なのはムービークリップシンボルだけです。
6. [ライブラリ] パネルで、シンボルを右クリック (Windows) または Control キーを押しながらクリック (Macintosh) して、コンテキストメニューから [リンケージ] を選択します。
7. シンボルにリンケージ識別子 Ball を割り当てます。
8. [AS 2.0 クラス] テキストボックスに「MoveRightDistance」と入力します。
9. タイムラインのフレーム 1 に次のコードを追加します。

```
this.attachMovie("Ball", "ball50_mc", 10, {distance:50});
this.attachMovie("Ball", "ball125_mc", 20, {distance:125});
```

   このコードにより、SWF ファイルのルートタイムラインにシンボルのインスタンスが新しく 2 つ作成されます。1 つは `ball50_mc` で、マウスクリックのたびに 50 ピクセルだけ移動します。もう 1 つは `ball125_mc` で、マウスクリックのたびに 125 ピクセルだけ移動します。
10. [制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

第 11 章

テキストとストリングの操作

11

Flash を使用して作成するアプリケーション、プレゼンテーション、グラフィックスには、多くの場合何らかのテキストが含まれます。使用できるテキストにはさまざまな種類があります。レイアウトの中では静止テキストを使用し、長いテキストにはダイナミックテキストを使用できます。また、ユーザー入力を取得するためにテキスト入力を使用する場合や、背景デザイン用のイメージ内でテキストを使用する場合もあります。Flash オーサリングツールや ActionScript を使用して、テキストフィールドを作成することもできます。

テキストを表示する 1 つの方法として、実行時にステージにロードされて表示される前に、コードでストリングの表示方法を操作することができます。アプリケーションでは、ストリングをサーバーに送信して応答を取得する、配列中のストリングを解析する、ユーザーがテキストフィールドに入力したストリングを検証するなど、いくつかの方法でストリングを操作できます。

この章では、コードを使用してテキストを操作する方法を中心に、アプリケーション内でテキストとストリングを使用するいくつかの方法を説明します。

次のリストでは、この章で使用されている用語について説明されています。

エイリアス　エイリアス処理されたテキストでは、アンチエイリアス処理されたテキスト (次の定義を参照) とは違い、カラー変化を使用してぎざぎざのエッジを滑らかにする処理は行われません。

アンチエイリアス　画面に表示される文字の端のぎざぎざが少なくなるよう、アンチエイリアス処理を行ってテキストの表示を滑らかにします。Flash のアンチエイリアスオプションにより、テキストのアウトラインをピクセルの境界に揃えることでテキストが読みやすくなります。とりわけ、小さいフォントサイズをきれいにレンダリングするのに有効です。

文字　文字とは、アルファベット、数字、区切り記号のことで、これらを組み合わせてストリングが作成されます。

デバイスフォント　デバイスフォントは Flash の特殊なフォントで、SWF ファイルには埋め込まれません。代わりに、Flash Player では、ローカルコンピュータにインストールされているフォントの中で、デバイスフォントに最も近いフォントを使用します。デバイスフォントを使用するとフォントアウトラインが埋め込まれないため、埋め込みフォントアウトラインを使用した場合に比べて SWF ファイルのサイズが小さくなります。しかし、デバイスフォントは SWF ファイルに埋め込まれないため、デバイスフォントで作成されたテキストは、デバイスフォントに対応するフォントがインストールされていないコンピュータシステムでは期待どおりに表示されない可能性があります。Flash には、_ゴシック (Osaka または MS UI Gothic に類似)、_明朝 (細明朝、MS P明朝に類似)、_等幅 (Osaka 等幅または MS ゴシックに類似)、_sans (Helvetica または Arial に類似)、_serif (Times Roman に類似)、_typewriter (Courier に類似) という 6 種類のデバイスフォントが用意されています。

フォント　類似したフォントフェイス、スタイル、サイズの文字のセット。

ストリング　文字のシーケンス。

テキスト　テキストフィールドに、あるいはユーザーインターフェイスコンポーネント内に表示可能な一連のストリング。

テキストフィールド　ユーザーに対してテキストを表示するための、ステージ上のビジュアルエレメント。HTML のテキスト入力フィールドコントロールやテキストエリアフォームコントロールと同じように、Flash ではテキストフィールドを編集可能 (読み取り専用) に設定したり、HTML 形式にしたり、複数行をサポートしたり、パスワードマスクをかけたり、HTML 形式のテキストに CSS スタイルシートを適用したりできます。

テキストフォーマット　テキストフィールドやテキストフィールド内の特定の文字にはフォーマットを適用できます。テキストに適用できるテキストフォーマットオプションとしては、行揃え、インデント、ボールド、カラー、フォントサイズ、マージン幅、イタリック、文字間隔などがあります。

テキストの詳細については、次のトピックを参照してください。

テキストフィールドについて 333
TextField クラスの使用. 334
テキストと変数のテキストフィールドへのロードについて. 341
フォントの使用 346
フォントのレンダリングとアンチエイリアステキストについて 354
テキストのレイアウトとフォーマットについて 362
CSS スタイルによるテキストのフォーマット 369
スタイルシートオブジェクトの作成. 371
HTML 形式のテキストの使用. 382
例 : スクロールテキストの作成 395

テキストフィールドについて

ダイナミックテキストフィールドまたはテキスト入力フィールドは、TextField オブジェクト (TextField クラスのインスタンス) です。オーサリング環境でテキストフィールドを作成すると、プロパティインスペクタでインスタンス名を付けられるようになります。インスタンス名を ActionScript ステートメント内で使用すると、TextField クラスと TextFormat クラスを利用して、テキストフィールドとその内容を設定、変更、およびフォーマットすることができます。

ユーザーインターフェイスを使用していろいろなテキストフィールドを作成できます。また、ActionScript を使用してテキストフィールドを作成することもできます。Flash では以下のテキストフィールドを作成できます。

静止テキスト　変更する必要のない文字や少量のテキスト、またはほとんどのコンピュータでは利用不可能な特殊なフォントを表示するために使用します。動的テキストフィールドに文字を埋め込んで一般的でないフォントを表示することもできます。

ダイナミックテキスト　実行時に更新または変更される文字を表示する必要があるときに使用します。ダイナミックテキストフィールドにテキストをロードすることもできます。

テキスト入力　ユーザー入力を取得する必要があるときに使用します。ユーザーはテキスト入力フィールドに入力できます。

テキストコンポーネント　TextArea または TextInput コンポーネントを使用すると、アプリケーション内でテキストを表示または取得できます。TextArea コンポーネントは、スクロールバーが組み込まれたダイナミックテキストフィールドに似ています。TextInput コンポーネントは、テキスト入力フィールドに似ています。どちらのコンポーネントも対応するテキストフィールドより機能が豊富ですが、アプリケーションのサイズは大きくなります。

メモ	どのテキストフィールドも Unicode に対応しています。Unicode の詳細については、397 ページの「ストリングおよび String クラスについて」を参照してください。

TextField クラスのメソッドを使用すると、オーサリング時または実行時に作成したダイナミックテキストフィールドやテキスト入力フィールドにテキストを設定、選択、および操作できます。詳細については、334 ページの「TextField クラスの使用」を参照してください。実行時にテキストフィールドをデバッグする方法の詳細については、『Flash ユーザーガイド』を参照してください。

ActionScript には、テキストを実行時にフォーマットする方法がいくつも用意されています。TextFormat クラスでは、TextField オブジェクトの文字フォーマットと段落フォーマットを設定できます。詳細については、367 ページの「TextFormat クラスの使用」を参照してください。また、Flash Player では、テキストのフォーマットに利用できる HTML タグのサブセットもサポートしています。詳細については、382 ページの「HTML 形式のテキストの使用」を参照してください。Flash Player 7 およびそれ以降では、HTML タグの `img` をサポートしており、これによって外部イメージの埋め込みだけでなく、外部 SWF ファイルとライブラリ内に存在するムービークリップを埋め込むことができます。詳細については、386 ページの「イメージタグ」を参照してください。

Flash Player 7 以降では、TextField.StyleSheet クラスを使用して、テキストフィールドに CSS (Cascading Style Sheet) スタイルを適用できます。CSS スタイルを使用すると、ビルトイン HTML タグのスタイルを設定、新しいフォーマットタグを定義、またはスタイルを適用できます。CSS の使用の詳細については、369 ページの「CSS スタイルによるテキストのフォーマット」を参照してください。

HTML 形式のテキスト (CSS スタイルを使用している場合も可) をテキストフィールドに直接割り当てることもできます。Flash Player 7 以降では、テキストフィールドに割り当てる HTML テキストに埋め込みメディア (ムービークリップ、SWF ファイル、および JPEG ファイル) を含めることができます。Flash Player 8 以降では、PNG、GIF、および " プログレッシブ " JPEG イメージも動的にロードできます (Flash Player 7 はプログレッシブ JPEG イメージをサポートしていません)。この場合テキストは、Web ブラウザ上で HTML ドキュメントの埋め込みメディアの周りをテキストが囲むのと同じように、埋め込みメディアの周りを囲みます。詳細については、386 ページの「イメージタグ」を参照してください。

テキスト、ストリングなどを比較する用語の詳細については、本章 331 ページの「テキストとストリングの操作」の導入部分を参照してください。

TextField クラスの使用

TextField クラスは、Flash のテキストツールを使用して作成するダイナミックテキストフィールドまたはテキスト入力 (編集可能) フィールドを表します。このクラスのメソッドとプロパティを使用すると、テキストフィールドを実行時に制御できます。TextField オブジェクトは MovieClip オブジェクトと同じプロパティをサポートしますが、`_currentframe`、`_droptarget`、`_framesloaded`、および `_totalframes` プロパティは除きます。テキストフィールドのプロパティをダイナミックに取得または設定したり、メソッドを呼び出したりできます。

ダイナミックテキストフィールドまたはテキスト入力フィールドを ActionScript で制御するには、プロパティインスペクタ内でテキストフィールドにインスタンス名を割り当てる必要があります。そうすれば、テキストフィールドをインスタンス名で参照し、TextField クラスのメソッドとプロパティを使用して、そのテキストフィールドの内容や基本的な外観を制御することができます。

また、`MovieClip.createTextField()` メソッドを使用すれば、TextField オブジェクトを実行時に作成し、それにインスタンス名を割り当てることができます。詳細については、337 ページの「実行時におけるテキストフィールドの作成」を参照してください。

TextField クラスの詳細については、次のトピックを参照してください。

- 335 ページの「実行時のテキストフィールドへのテキスト割り当て」
- 336 ページの「テキストフィールドのインスタンス名と変数名について」

ActionScript を使用してテキストフィールドを制御する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript 2.0/TextFields" フォルダに移動して次のサンプルにアクセスします。

- textfieldsA.fla
- textfieldsB.fla

実行時のテキストフィールドへのテキスト割り当て

Flash でアプリケーションを作成する際、テキストファイルや XML ファイル、さらにはリモート Web サービスなどの外部ソースからテキストをロードする必要が生じることがあります。Flash では、HTML フォーマットのテキストやプレーンテキスト、XML フォーマットのテキスト、外部スタイルシートをサポートするなど、さまざまな方法でステージ上にテキストを作成および表示できます。また、ActionScript を使用してスタイルシートを定義することもできます。

テキストフィールドにテキストを割り当てるには、text プロパティまたは htmlTEXT プロパティを使用するか、プロパティインスペクタの [変数] フィールドでテキストフィールドの変数名を作成して、その変数名にテキストを割り当てるか、テキストフィールドを他のコンポーネントのテキストフィールドにバインドして値を割り当てます。

次の実習では、実行時にテキストフィールドにテキストが割り当てられます。

実行時にテキストフィールドにテキストを割り当てるには :

1. テキストツールを使用して、テキストフィールドをステージに作成します。
2. テキストフィールドを選択した状態で、プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) 内で、[テキストの種類] ポップアップメニューから [テキスト入力] を選択し、[インスタンス名] テキストボックスに「headline_txt」と入力します。

   インスタンス名に使用できる文字は、英字、数字、アンダースコア (_)、ドル記号 ($) だけです。
3. タイムラインのフレーム 1 を選択し、[アクション] パネルを開きます ([ウィンドウ]-[アクション])。
4. [アクション] パネルで、次のコードを入力します。

```
headline_txt.text = "New articles available on Developer Center";
```

5. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

ActionScript でテキストフィールドを作成し、テキストを割り当てることもできます。タイムラインのフレーム 1 に次の ActionScript を入力します。

```
this.createTextField("headline_txt", this.getNextHighestDepth(), 100, 100, 300, 20);
headline_txt.text = "New articles available on Developer Center";
```

このコードは、インスタンス名 `headline_txt` という新しいテキストフィールドを作成します。このテキストフィールドは、次に最も高い深度に作成され、x 座標と y 座標は 100 と 100、テキストフィールドの幅は 200 ピクセル、高さは 20 ピクセルです。[制御]-[ムービープレビュー] を選択してこの SWF ファイルをテストすると、ステージに "New articles available on Developer Center" というテキストが表示されます。

HTML 形式のテキストフィールドを作成するには :

テキストフィールドで HTML 形式を有効にするには、次のいずれかの手順を使用します。

- テキストフィールドを選択してプロパティインスペクタで [テキストを HTML としてレンダリング] ボタンをクリックする。
- ActionScript を使用してテキストフィールドの `html` プロパティを `true` に設定する (次のコード例を参照)。

ActionScript を使用してテキストフィールドに HTML 形式を適用するには、タイムラインのフレーム 1 に次の ActionScript を入力します。

```
this.createTextField("headline_txt", this.getNextHighestDepth(), 100, 100, 300,
  20);
headline_txt.html = true;
headline_txt.htmlText = "New articles available on <i>Developer Center</i>.";
```

前のコードでは、新しいテキストフィールドが動的に作成され、HTML 形式が有効になり、ステージに "New articles available on Developer Center" というテキストが表示され、そして単語 "Developer Center" がイタリックで表示されます。

注意	HTML 形式のテキストをステージ上のテキストフィールド (コンポーネントではなく) で使用する場合は、そのテキストをテキストフィールドの text プロパティではなく `htmlText` プロパティに割り当てる必要があります。

ActionScript を使用してテキストフィールドを制御する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript 2.0/TextFields" フォルダに移動して次のサンプルにアクセスします。

- textfieldsA.fla
- textfieldsB.fla

テキストフィールドのインスタンス名と変数名について

テキストフィールドのメソッドの呼び出し、そのプロパティの取得または設定を行うには、プロパティインスペクタの [インスタンス名] テキストボックスで、そのテキストフィールドにインスタンス名を割り当てる必要があります。

プロパティインスペクタの [変数] テキストボックスで、変数名をダイナミックテキストフィールドやテキスト入力フィールドに割り当てることができます。これによって、変数に値を割り当てることができます。これは、旧バージョンの Flash Player (Flash Player 4 など) 用のアプリケーションを作成するときに使用する可能性のある非推奨の機能です。それよりも新しいプレーヤーを対象にするときは、インスタンス名と ActionScript を使用して、テキストフィールドのテキストをターゲットにする必要があります。

テキストフィールドのインスタンス名と変数名を混同しないでください。テキストフィールドの変数名は、テキストフィールドに含まれているテキストを参照する変数であり、オブジェクトへの参照ではありません。

たとえば、テキストフィールドに `myTextVar` という変数名を割り当てた場合は、次のコードを使用してテキストフィールドの内容を設定できます。

```
var myTextVar:String = "This is what will appear in the text field";
```

ただし、myTextVar 変数名を使用してテキストフィールドの text プロパティを設定することはできません。次のコードのように、インスタンス名を使用する必要があります。

```
// このコードは無効
myTextVar.text = "A text field variable is not an object reference";

// インスタンス名 "myField" を使用するテキスト入力フィールドの場合は、このコードが機能する
myField.text = "This sets the text property of the myField object";
```

TextField クラスをサポートしていないバージョンの Flash Player を対象にする場合以外は、TextField.text プロパティを使用してテキストフィールドの内容を制御するのが一般的です。こうすれば、変数名のコンフリクトが起きて実行時に予想外の結果になる可能性が少なくなります。

ActionScript を使用してテキストフィールドを制御する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript 2.0/TextFields" フォルダに移動して次のサンプルにアクセスします。

- textfieldsA.fla
- textfieldsB.fla

実行時におけるテキストフィールドの作成

MovieClip クラスの createTextField() メソッドを使用すると、実行時に空のテキストフィールドをステージ上に作成することができます。この新規テキストフィールドは、メソッドを呼び出すムービークリップのタイムラインに割り当てられます。

ActionScript を使用して動的にテキストフィールドを作成するには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択して新しい FLA ファイルを作成します。
2. タイムラインのフレーム 1 に次の ActionScript を入力します。

   ```
   this.createTextField("test_txt", 10, 0, 0, 300, 100);
   ```

   このコードは、test_txt という名前の 300 x 100 ピクセルのテキストフィールドを、位置 (0, 0)、深度 (z 順序) 10 に作成します。
3. 新規に作成されたテキストフィールドのメソッドとプロパティにアクセスするには、createTextField() メソッドの最初のパラメータに指定されたインスタンス名を使用します。

   たとえば、次のコードでは、test_txt という名前の新しいテキストフィールドを作成し、次に、そのプロパティを変更して、挿入するテキストに合わせて伸縮する複数行の折り返し可能なテキストフィールドにしています。最後に、このテキストフィールドの text プロパティを使用してテキストを割り当てています。

   ```
   test_txt.multiline = true;
   test_txt.wordWrap = true;
   test_txt.autoSize = "left";
   test_txt.text = "Create new text fields with the MovieClip.createTextField()
     method.";
   ```

4. [制御]-[ムービープレビュー] を選択してテキストフィールドを確認します。

   テキストは実行時に作成され、ステージ上に表示されます。

`TextField.removeTextField()` メソッドを使用すると、`createTextField()` で作成したテキストフィールドを削除できます。`removeTextField()` メソッドは、オーサリング中にタイムラインによって配置されたテキストフィールドに対しては機能しません。

詳細については、『ActionScript 2.0 リファレンスガイド』の「createTextField (MovieClip.createTextField メソッド)」および「removeTextField (TextField.removeTextField メソッド)」を参照してください。

メモ	`_rotation` など、TextField プロパティによっては、実行時にテキストフィールドを作成する際に使用できないものもあります。テキストフィールドは、埋め込みフォントを使用した場合にのみ回転できます。348 ページの「フォントシンボルを埋め込むには :」を参照してください。

ActionScript を使用してテキストフィールドを制御する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript 2.0/TextFields" フォルダに移動して次のサンプルにアクセスします。

- textfieldsA.fla
- textfieldsB.fla

テキストフィールドの操作について

FLA 内に作成したテキストフィールドを操作する方法はいくつかあります。プロパティインスペクタでインスタンス名を割り当てればテキストフィールドを操作できますし、コードを使用してフィールドを作成する場合は、インスタンス名をコードで割り当てることができます。次の簡単な例では、テキストフィールドを作成し、それにテキストを割り当て、フィールドの border プロパティを変更します。

```
this.createTextField("pigeon_txt", this.getNextHighestDepth(), 100, 100, 200, 20);
pigeon_txt.text = "I like seeds";
pigeon_txt.border = true;
```

TextField クラスのプロパティの一覧については、『ActionScript 2.0 リファレンスガイド』を参照してください。

テキストフィールドの操作方法の例については、以下のセクションを参照してください。

- 339 ページの「テキストフィールドの位置の変更」
- 339 ページの「実行時のテキストフィールドのサイズ変更」

ActionScript を使用してテキストフィールドを制御する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript 2.0/TextFields" フォルダに移動して次のサンプルにアクセスします。

- textfieldsA.fla
- textfieldsB.fla

テキストフィールドの位置の変更

ステージ上のテキストフィールドの位置を実行時に変更できます。次の例のように、テキストフィールドの _x プロパティと _y プロパティに新しい値を設定します。

ActionScript を使用してテキストフィールドの位置を変更するには :

1. 新しい FLA ファイルを作成し、positionText.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createTextField("my_txt", 10, 0, 0, 300, 200);
my_txt.border = true;
my_txt.text = "Hello world";
my_txt._x = (Stage.width - my_txt._width) / 2;
my_txt._y = (Stage.height - my_txt._height) / 2;
```

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択してテキストフィールドがステージの中央にあることを確認します。

ActionScript を使用してテキストフィールドを制御する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript 2.0/TextFields" フォルダに移動して次のサンプルにアクセスします。

- textfieldsA.fla
- textfieldsB.fla

実行時のテキストフィールドのサイズ変更

オーサリング環境ではなく実行時に、テキストフィールドのサイズを動的に取得または設定することが必要になる場合があります。次の例では、タイムラインにテキストフィールドを作成し、初期サイズを幅 100 ピクセル、高さ 21 ピクセルに設定しています。その後、テキストフィールドのサイズを幅 300 ピクセル、高さ 200 ピクセルに変更し、ステージの中央に位置を変更しています。

ActionScript を使用してテキストフィールドのサイズを変更するには :

1. 新しい Flash ドキュメントを作成し、resizeText.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createTextField("my_txt", 10, 0, 0, 100, 21);
my_txt.border = true;
my_txt.multiline = true;
my_txt.text = "Hello world";
my_txt.wordWrap = true;
my_txt._width = 300;
my_txt._height = 200;
my_txt._x = (Stage.width - my_txt._width) / 2;
my_txt._y = (Stage.height - my_txt._height) / 2;
```

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択してオーサリング環境の結果を確認します。

前の例では、実行時に動的に作成したテキストフィールドのサイズを 300 x 200 ピクセルに変更していますが、外部の Web サイトからコンテンツをロードするときに、どのくらいの量のコンテンツが戻されるかわからない場合には、この方法は適していない可能性があります。ただし、Flash の `TextField.autoSize` プロパティを使用すると、テキストフィールドのサイズをコンテンツに合わせて自動的に変更することができます。次の例は、テキストフィールドにテキストが追加された後に、`TextField.autoSize` プロパティを使用してテキストフィールドのサイズを変更する方法を示しています。

コンテンツに応じてテキストフィールドのサイズを自動的に変更するには :

1. 新しい Flash ドキュメントを作成し、resizeTextAuto.fla という名前で保存します。
2. メインタイムラインのフレーム 1 に次のコードを追加します。

```
this.createTextField("my_txt", 10, 10, 10, 160, 120);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.text = "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
   do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
   minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex
   ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate
   velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
   cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id
   est laborum.";
my_txt.wordWrap = true;
```

メモ	Flash Help のバージョンによっては、このコードをペーストして直接 [アクション] パネルにペーストすると、長いテキストストリングが改行されてしまうことがあります。こうなると、コードはコンパイルされません。このような状況が発生した場合は、[アクション] パネルのポップアップメニューの [隠し文字] を有効にし、そのテキストストリングから改行文字を削除してください。

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択してオーサリング環境で Flash ドキュメントを表示します。

   すべてのコンテンツが、テキストフィールドの境界によってトリミングされることなく表示されるように、テキストフィールドが垂直方向にサイズ変更されます。`my_txt.wordWrap` プロパティを `false` に設定すると、テキストに合わせてテキストフィールドが水平方向にサイズ変更されます。

   サイズが自動調整されるテキストフィールドの最大の高さを (ステージの境界を超えないように) 制限するには、次のコードを使用します。

```
if (my_txt._height > 160) {
   my_txt.autoSize = "none";
   my_txt._height = 160;
}
```

ユーザーがテキストの残りの部分を表示できるようにするには、スクロールバーなどのスクロール機能を追加する必要があります。別の方法としては、マウスポインタをテキストに合わせてホイールを回転させるという方法もあります。このコードをテストするだけであれば、通常はこの方法で十分でしょう。

ActionScript を使用してテキストフィールドを制御する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript 2.0/TextFields" フォルダに移動して次のサンプルにアクセスします。

- textfieldsA.fla
- textfieldsB.fla

テキストと変数のテキストフィールドへのロードについて

テキストを Flash ドキュメントにロードするには、FlashVars、LoadVars、XML、Web サービスなど、いくつかの方法を利用できます。その中で最も簡単なのは、FlashVars プロパティを使用する方法でしょう。この方法では、SWF ファイルを HTML ページに埋め込むために使用される HTML コード内の `object` タグと `embed` タグによって、テキストの短いストリングが Flash ドキュメントにロードされます。Flash ドキュメントにテキストや変数をロードするもう 1 つの簡単な方法は、LoadVars クラスを使う方法です。LoadVars クラスは、テキストファイルから大きなテキストブロックや URL エンコードされた一連の変数をロードするのに使用できます。

このセクションで既に紹介した例でもわかるように、比較的簡単にテキストを SWF ファイルにロードする方法があります。しかし、外部サイトからデータを受け取る場合には、ロードするデータのフォーマットを選択できない可能性があります。

データの SWF ファイルからのロードや SWF ファイルへの送信を行ういずれの方法にも長所と短所があります。XML や Web サービス、Flash Remoting は外部データをロードする最も用途の広い方式ですが、その分だけ習得するのも簡単ではありません。Flash Remoting の詳細については、www.adobe.com/jp/support/flashremoting を参照してください。

FlashVars と LoadVars の使用は、342 ページの「FlashVars を使用したテキストのロードと表示」および 343 ページの「LoadVars を使用したテキストのロードと表示」に示されているように、それよりもはるかに簡単ですが、ロード可能なデータのタイプとフォーマットが大幅に限定される可能性があります。また、データの送信およびロードを行う際にセキュリティ制限に従う必要もあります。セキュリティの詳細については、第 16 章の「セキュリティについて」を参照してください。外部データのロードの詳細については、第 15 章の「外部データの操作」を参照してください。

以下のセクションでは、テキストと変数をドキュメントにロードするさまざまな方法を紹介しています。

- 342 ページの「FlashVars を使用したテキストのロードと表示」
- 343 ページの「LoadVars を使用したテキストのロードと表示」
- 344 ページの「LoadVars を使用した変数のロード」
- 345 ページの「XML ドキュメントからのテキストのロードと表示」

ActionScript を使用してテキストフィールドを制御する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript 2.0/LoadText" フォルダに移動して次のサンプルにアクセスします。

- loadText.fla
- formattedText.fla

テキストをロードし、ビットマップキャッシュに加えてアンチエイリアスフォーマットを適用するサンプルのソースファイル "aliasing.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript 2.0/Advanced Anti-Aliasing" フォルダに移動してサンプルにアクセスします。

FlashVars を使用したテキストのロードと表示

FlashVars は簡単に使用できますが、HTML ドキュメントと共に SWF ファイルをパブリッシュする必要があります。そのためには、生成された HTML コードを修正し、`object` タグと `embed` タグの両方に FlashVars プロパティを含める必要があります。それから、修正後の HTML ドキュメントを Web ブラウザで表示して Flash ドキュメントをテストします。

FlashVars を使用して HTML から Flash ドキュメントに変数を渡すには :

1. 新しい Flash ドキュメントを作成し、flashvars.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createTextField("my_txt", 10, 10, 10, 100, 21);
my_txt.text = _level0.username;
```

3. Flash ドキュメントを保存し、[ファイル]-[パブリッシュ] を選択して HTML および SWF ファイルを生成します。

メモ	HTML ドキュメントは、デフォルトでは、FLA ファイルと同じディレクトリにパブリッシュされます。HTML ドキュメントがパブリッシュされない場合、[ファイル]-[パブリッシュ設定] を選択し、それから [形式] タブを選択します。[HTML] が選択されていることを確認してください。

4. テキストエディタまたは HTML エディタで flashvars.html ドキュメントを開きます。

5. HTML ドキュメントで、object タグ内のコードを次に合わせて変更します。

   追加する必要のあるコードは、**ボールド体**で示されています。

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
   codebase="http://fpdownload.adobe.com/pub/shockwave/cabs/flash/
   swflash.cab#version=8,0,0,0" width="550" height="400" id="flashvars"
   align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="flashvars.swf" />
<param name="FlashVars" value="username=Thomas" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="flashvars.swf" FlashVars="username=Thomas" quality="high"
   bgcolor="#ffffff" width="550" height="400" name="flashvars" align="middle"
   allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
   pluginspage="http://www.adobe.com/go/getflashplayer_jp" />
</object>
```

6. HTML ドキュメントの変更を保存します。
7. 変更された HTML を Web ブラウザで開きます。

   SWF ファイルにより、ステージ上の動的に作成されたテキストフィールドに "Thomas" という名前が表示されます。

セキュリティの詳細については、第 16 章の「セキュリティについて」を参照してください。

LoadVars を使用したテキストのロードと表示

LoadVars クラスを使用してコンテンツを SWF ファイルにロードすることもできます。この場合は、同じサーバーの外部ファイルからテキストまたは変数をロードするだけでなく、別のサーバーからコンテンツをロードできます。次の例は、テキストフィールドを動的に作成し、それにリモートテキストファイルのコンテンツをロードする方法を示しています。

LoadVars を使用してテキストフィールドに外部テキストをロードするには :

1. 新しい Flash ドキュメントを作成し、loadvarsText.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createTextField("my_txt", 10, 10, 10, 320, 100);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function (src:String):Void {
   if (src != undefined) {
      my_txt.text = src;
   } else {
      my_txt.text = "Unable to load external file.";
   }
}
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

前の部分の最初のコードブロックにより、ステージ上に新しいテキストフィールドが作成され、複数行と折り返しが有効になります。2 番目のコードブロックにより、新しい LoadVars オブジェクトが定義されます。このオブジェクトは、リモート Web サーバーからテキストファイル (lorem.txt) をロードし、事前に作成されている my_txt テキストフィールドにそのコンテンツを表示するために使用されます。

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

   その少し後に、リモートファイルのコンテンツがステージ上のテキストフィールドに表示されます。

セキュリティの詳細については、第 16 章の「セキュリティについて」を参照してください。

LoadVars を使用した変数のロード

LoadVars クラスを使用すると、Web ブラウザでクエリーストリングに変数を渡す場合と同じように、URL エンコード形式の変数もロードできます。次の例は、リモートテキストファイルを SWF ファイルにロードし、その変数 `monthNames` および `dayNames` を表示する方法を示しています。

LoadVars を使用してテキストファイルから変数をロードするには :

1. 新しい Flash ドキュメントを作成し、loadvarsVariables.fla という名前で保存します。
2. タイムラインのフレーム 1 に次のコードを追加します。

```
this.createTextField("my_txt", 10, 10, 10, 320, 100);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onLoad = function (success:Boolean):Void {
  if (success) {
    my_txt.text = "dayNames: " + lorem_lv.dayNames + "\n\n";
    my_txt.text += "monthNames: " + lorem_lv.monthNames;
  } else {
    my_txt.text = "Unable to load external file.";
  }
}
/* params.txt の内容 :
  &monthNames= 1 月、2 月、... &dayNames= 日曜日、月曜日、...
*/
lorem_lv.load("http://www.helpexamples.com/flash/params.txt");
```

3. Flash ドキュメントを保存し、メインメニューから [制御]-[ムービープレビュー] を選択します。

   使用しているメソッドが `LoadVars.onData()` ではなく `LoadVars.onLoad()` であるため、Flash は変数を解析し、LoadVars オブジェクトインスタンス内に変数を作成します。外部テキストファイルには 2 つの変数、`monthNames` および `dayNames` が入っており、どちらもストリングを含んでいます。

セキュリティの詳細については、第 16 章の「セキュリティについて」を参照してください。

XML ドキュメントからのテキストのロードと表示

XML データは、インターネットにコンテンツを配布する一般的な方法になっていますが、広く支持されているデータ編成および解析の標準であることが理由の 1 つです。そのため、XML は Flash とデータを送受信する方法として優れた選択肢になります。ただし XML を使う方法は、LoadVars および FlashVars を使ってデータをロードしてテキストを表示する方法と比べてやや難しくなります。

外部 XML ドキュメントから Flash にテキストをロードするには :

1. 新しい Flash ドキュメントを作成し、xmlReviews.fla という名前で保存します。
2. タイムラインのフレーム 1 に次のコードを追加します。

```
this.createTextField("my_txt", 10, 10, 10, 320, 100);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;

var reviews_xml:XML = new XML();
reviews_xml.ignoreWhite = true;
reviews_xml.onLoad = function (success:Boolean):Void {
  if (success) {
    var childItems:Array = reviews_xml.firstChild.childNodes;
    for (var i:Number = 0; i < childItems.length; i++) {
      my_txt.text += childItems[i].firstChild.firstChild.nodeValue + "\n";
    }
  } else {
    my_txt.text = "Unable to load external file.";
  }
}
reviews_xml.load("http://www.helpexamples.com/flash/xml/reviews.xml");
```

   前の部分の最初のコードブロックにより、ステージ上に新しいテキストフィールドが作成されます。このテキストフィールドが、後でロードされる XML ドキュメントのさまざまな部分を表示するために使用されます。2 番目のコードブロックは、XML コンテンツをロードするために使用される XML オブジェクトの作成を処理します。データが完全にロードされ、Flash によって解析されたら、`XML.onLoad()` イベントハンドラが起動し、XML パケットの内容がテキストフィールドに表示されます。

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

   Flash により、ステージ上のテキストフィールドに以下の出力が表示されます。

```
Item 1
Item 2
...
Item 8
```

セキュリティの詳細については、第 16 章の「セキュリティについて」を参照してください。

フォントの使用

フォントとは、類似したフォントフェイス、スタイル、サイズの文字のセットのことです。Flash を使用して作成するものには、通常は Flash アプリケーション内で少なくとも 1 つか 2 つのフォントを含むテキストを使用することになります。アニメーションを作成するときに、エンドユーザーが特定のフォントをシステムにインストールしているかどうかがわからない場合は、埋め込みフォントに関する基本を理解しておく必要があります。

以下のセクションでは、文字、フォント全体、共有フォントを埋め込む方法、および Flash でフォントを操作するその他のテクニックを説明します。

フォントの詳細については、次に示す各セクションを参照してください。

- 347 ページの「文字の埋め込み」
- 348 ページの「フォントの埋め込み」
- 350 ページの「カスタム文字セットの作成」
- 352 ページの「埋め込みフォントでの TextField メソッドの使用」
- 353 ページの「フォントの共有について」

次の例では、Flash ドキュメント内に埋め込み文字および文字セットを追加および削除する方法が示されています。

埋め込み文字および文字セットを追加および削除するには :

1. 新しい Flash ドキュメントを作成し、embedding.fla という名前を付けて保存します。
2. テキストツールを使ってステージ上にダイナミックテキストフィールドを作成します。
3. [埋め込み] をクリックして [文字の埋め込み] ダイアログボックスを表示します。
4. 埋め込みたい特定の文字セットをマウスポインタでクリックして選択します。

   複数の文字セットを選択するには、Shift キーまたは Ctrl キーを押しながらマウスポインタでアイテムを選択します。文字セットをまとめて選択するには、マウスポインタで 1 つの文字セットを選択し、Shift キーを押したまま新しい文字セットをクリックします。Shift キーを使うと、選択した 2 つの文字セット間にある文字セットがすべて選択されます。複数の文字セットを非連続的に選択するには、Ctrl キーを押したまま文字セットを選択します。また、マウスで文字セットを 1 つ選択し、マウスボタンを押したまま複数の文字セット上にマウスをドラッグすることで、すばやく複数の文字セットを選択することもできます。
5. 前に追加した文字セットを削除するには、Ctrl キーを押したまま削除したい文字セットをマウスポインタでクリックすることで選択解除できます。
6. 選択した文字セットおよび [追加する文字を指定する] テキスト入力フィールドで指定された文字をすべて削除するには、[埋め込まない] をクリックします。

   [埋め込まない] をクリックすると、以前に指定された個々の文字および文字列がすべてクリアされます。

注意	[文字の埋め込み] ダイアログボックスで [埋め込まない] をクリックすると、指定された、以前に選択された埋め込み文字および文字セットがすべて、確認なしに削除されます。

文字の埋め込み

埋め込みフォントを使って作業している場合で、どの文字が必要か正確にわかっている場合は、使用しない余分なフォントのアウトラインを含めずに必要な文字だけを埋め込むことでファイルサイズを小さくすることができます。文字セット全体を埋め込むのではなく特定の文字だけをテキストフィールドに埋め込むには、[文字の埋め込み] ダイアログボックスで埋め込みたい文字を指定します。

使用する特定の文字をテキストフィールドに埋め込むには :

1. 新しい Flash ドキュメントを作成し、charembed.fla という名前を付けて保存します。
2. テキストツールを使用し、ステージ上にテキストフィールドを作成します。それからテキストフィールドのテキストの種類をダイナミックまたはテキスト入力に設定します。
3. テキストフィールドをステージ上で選択したまま、プロパティインスペクタで [埋め込み] をクリックして [文字の埋め込み] ダイアログボックスを開きます。

   [文字の埋め込み] ダイアログボックスでは、Flash ドキュメントにどの文字セットを埋め込むのか (および文字セットあたりの文字数) の設定や埋め込む文字の指定ができます。また、このテキストフィールドに埋め込まれる文字の総数も表示されます。
4. [追加する文字を指定する] テキストボックスに「hello world」というストリングを入力します。

   ダイアログボックスには、このテキストフィールドに埋め込まれる合計文字数は 8 であることが表示されます。ストリング "hello world" の文字数は 11 ですが、Flash で埋め込まれるのは一意の文字だけなので、文字 l と o は複数回ではなく 1 度だけ埋め込まれます。
5. [OK] をクリックして変更を適用し、ドキュメントに戻ります。
6. テキストツールを使用して、新しいテキストフィールドをステージに作成します。
7. プロパティインスペクタで、テキストフィールドのテキストの種類をダイナミックに設定します。
8. ステージのテキストフィールドに「hello world」というストリングを入力します。
9. プロパティインスペクタで [埋め込み] をクリックし、もう一度 [文字の埋め込み] ダイアログボックスを開きます。
10. [自動読み込み] をクリックすると、入力されたストリングが [追加する文字を指定する] テキストボックスに自動的に流し込まれます。

    "helo wrd" というストリングが表示されます。このように、どの文字を含めるかをわざわざ指定しなくても、指定されたテキストフィールドに埋め込まれる一意の文字の判断が Flash 側ですべて行われます。

ヒント	埋め込まれる文字の判断を Flash 側で行えるのは、ステージ上でテキストをテキストフィールドに設定する場合のみです。テキストフィールドに ActionScript を使用して値を設定する場合は、そのテキストフィールドに埋め込む文字を指定する必要があります。

11. [OK] をクリックします。

フォントの埋め込み

フォントを埋め込むと、Flash によりすべてのフォント情報が SWF ファイルに格納され、そのフォントがユーザーのコンピュータにインストールされていなくても、正しく表示されるようになります。ユーザーのシステムにインストールされていない FLA ファイル内のフォントを使用し、そのフォントを SWF ファイルに埋め込んでいない場合は、代わりに使用する代替フォントを Flash Player が自動的に選択します。

メモ	ダイナミックテキストフィールドまたはテキスト入力フィールドを使用している場合は、フォントを埋め込むだけで済みます。静止テキストフィールドを使用している場合は、フォントを埋め込む必要はありません。

フォントシンボルを埋め込むには :

1. [ウィンドウ]-[ライブラリ] を選択して、現在の FLA ファイルのライブラリを開きます。

   フォントシンボルを追加するライブラリを開きます。
2. [ライブラリ] パネルの右上隅にあるポップアップメニューから [新しいフォント] を選択します。
3. [フォントシンボルプロパティ] ダイアログボックスの [名前] テキストボックスにフォントシンボルの名前を入力します。
4. [フォント] メニューからフォントを選択するか、または [フォント] テキストボックスにフォント名を入力します。
5. フォントにスタイルを適用する場合、[ボールド]、[イタリック] または [ビットマップテキスト] を選択します。
6. 埋め込むフォントサイズを入力し、[OK] をクリックして、変更を適用してドキュメントに戻ります。

   現在のドキュメントのライブラリにそのフォントが表示されます。

フォントをライブラリに埋め込んだら、ステージ上のテキストフィールドで使用できます。

Flash ドキュメントで埋め込みフォントシンボルを使用するには :

1. 348 ページの「フォントの埋め込み」の手順に従って、ライブラリにフォントを埋め込みます。
2. テキストツールを使用して、ステージ上にテキストフィールドを作成します。
3. テキストフィールドにテキストを入力します。
4. テキストフィールドを選択し、プロパティインスペクタを開きます。
   a. テキストフィールドを単一行に設定します。
   b. [フォント] ドロップダウンメニューを使用して、埋め込みフォントの名前を選択します。

   埋め込みフォントのフォント名の後にはアスタリスク (*) が付きます。
5. プロパティインスペクタで [埋め込み] をクリックして、[文字の埋め込み] ダイアログボックスを表示します。

[文字の埋め込み] ダイアログボックスでは、選択されたテキストフィールドに埋め込む個々の文字または文字セットを選択できます。どの文字を埋め込むかを指定するには、埋め込む文字をダイアログボックスに入力するか、[自動読み込み] をクリックして現在テキストフィールド内にある一意の文字をテキストフィールドに自動的に設定します。どの文字が必要になるのかがわからない場合は (外部ファイルや Web サービスからテキストをロードする場合など)、大文字 [A..Z]、小文字 [a..z]、数字 [0..9]、区切り記号 [!@#%...]、および異なる言語の文字セットなど、埋め込む文字セット全体を選択できます。

メモ	文字セットを選択するたびに、Flash は文字セットごとにすべてのフォント情報を格納する必要があるため、SWF ファイルの最終的なサイズが大きくなります。

6. 埋め込む個々の文字または文字セットを選択し、[OK] をクリックして変更を適用し、ドキュメントに戻ります。
7. [制御]-[ムービープレビュー] を選択して、オーサリング環境で Flash ドキュメントをテストします。

   埋め込まれたフォントがステージ上のテキストフィールドに表示されます。フォントが埋め込まれたことを厳密にテストするには、そのフォントがインストールされていない別のコンピュータでテストすることが必要になります。

   または、フォントを埋め込んだテキストフィールドに `TextField._alpha` または `TextField._rotation` プロパティを設定します。これらのプロパティは埋め込みフォントだけに作用するからです。次の手順を参照してください。
8. SWF ファイルを閉じてオーサリングツールに戻ります。
9. ステージ上のテキストフィールドを選択し、プロパティインスペクタを開きます。
   a. テキストフィールドの [テキストの種類] を [ダイナミックテキスト] に設定します。
   b. [インスタンス名] テキストボックスに「font_txt」と入力します。
10. タイムラインのフレーム 1 に次のコードを追加します。

```
font_txt._rotation = 45;
```

11. [制御]-[ムービープレビュー] を再び選択して、オーサリング環境で変更内容を表示します。

    埋め込まれたフォントは時計回りに 45 度回転しますが、SWF ファイルに埋め込まれているため、テキストは表示されたままになります。

注意	Flash ドキュメント内にフォントを埋め込まず、Flash Player が自動的にユーザーのコンピュータ上で代替フォントを選択する場合、TextField.font には、代替フォントの名前ではなく、FLA 内で使用されている本来のフォントを返します。

メモ	テキストフィールドで使用する埋め込みフォントにさまざまなスタイルを適用する場合は、使用するスタイルを埋め込む必要があります。たとえば、Times という埋め込みフォントを使用していて、単語をイタリックにしたい場合は、標準のフォントとイタリック体のフォントの両方のアウトラインを埋め込む必要があります。そうしないと、テキストはテキストフィールドに表示されません。

カスタム文字セットの作成

Flash のデフォルトの文字セットを使用するだけでなく、独自の文字セットを作成して [文字の埋め込み] ダイアログボックスに追加できます。たとえば、各種のアクセント記号付き文字をサポートするために、あるフィールドでは Extended Latin を使用したいという場合があるかもしれません。しかし、数字と区切り記号が必要ない場合や、大文字だけが必要な場合もあります。そのような場合に、文字セット全体を埋め込むのではなく、必要な文字だけが含まれるカスタム文字セットを作成できます。この方法だと、必要のない文字に関する余分なフォント情報を格納する必要が一切ないため、SWF ファイルのサイズをできる限り小さく保つことができます。

カスタム文字セットを作成するには、"C:¥Program Files¥Adobe¥Adobe Flash CS3¥‹ 言語 ›¥First Run¥FontEmbedding¥" ディレクトリにある "UnicodeTable.xml" ファイルを編集する必要があります。このファイルでは、文字セットおよび文字セットに含まれる文字の範囲および文字が定義されます。

カスタム文字セットを作成する前にまず、必要とされる XML 構造について理解しておく必要があります。次の XML ノードでは、文字セット Uppercase [A..Z] が定義されています。

```
<glyphRange name="Uppercase [A..Z] " id="1" >
  <range min="0x0020" max ="0x0020" />
  <range min="0x0041" max ="0x005A" />
</glyphRange>
```

glyphRange ノードには name、Uppercase [A..Z]、id という項目がある点に注目してください。glyphRange ノードには必要なだけ子ノード range を定義できます。range には、前のコードに出てくる 0x0020 (空白文字) のように文字を 1 つだけ指定することも、2 番目の range 子ノードのように文字の範囲を指定することもできます。1 文字だけを埋め込むには、min 値と max 値に同じ Unicode 値を設定します。

XML の glyphRange ノードの別の例として、次の Numerals [0..9] ノードがあります。

```
<glyphRange name="Numerals [0..9] " id="3" >
  <range min="0x0030" max ="0x0039" />
  <range min="0x002E" max ="0x002E" />
</glyphRange>
```

ここでは、文字の範囲として Unicode 値 0x0030 (ゼロ) から 0x0039 (9) までと、0x002E (.) が設定されています。

カスタム文字セットを作成する前に、文字とそれに対応する Unicode 値を知っておく必要があります。Unicode 値を知りたい場合、Unicode 規格の Web サイト (www.unicode.org) を参照してください。このサイトでは、多数の言語の文字についての Unicode 文字コード表が参照できます。

注意	カスタム文字セットを追加するには、Flash のインストールフォルダにある XML ファイルを編集する必要があります。このファイルを編集する前には、元の Unicode 表に戻したい場合に備えてバックアップコピーを作成しておいてください。

注意	アドビ システムズ社では、Flash のインストール時に一緒にインストールされた既存の文字セットを変更することはお勧めしていません。代わりに、必要な文字および区切り記号が記述された独自のカスタム文字セットを作成することをお勧めします。

カスタム文字セットを作成および使用するには :

1. "‹Flash のインストールディレクトリ ›¥‹ 言語 ›¥First Run¥FontEmbedding¥" ディレクトリにあるドキュメント "UnicodeTable.xml" を、Notepad や TextEdit などの XML エディタやテキストエディタで開きます。

メモ	ファイルを Flash インストール時のオリジナルに戻したい場合に備えて、このドキュメントのバックアップを保存するのを忘れないでください。

2. XML ドキュメントの最後までスクロールし、一番最後の `</fontEmbeddingTable>` ノードの直前に次の XML コードを直接追加してください。

```
<glyphRange name="Uppercase and Numerals [A..Z,0..9] " id="100" >
  <range min="0x0020" max ="0x0020" />
  <range min="0x002E" max ="0x002E" />
  <range min="0x0030" max ="0x0039" />
  <range min="0x0041" max ="0x005A" />
</glyphRange>
```

3. 変更内容を UnicodeTable.xml に保存します。

   Flash が開いている場合、作成した新しい文字セットを使用するには Flash を再起動する必要があります。

4. Flash を起動または再起動し、新しい Flash ドキュメントを作成します。
5. テキストツールを使ってステージ上に新しく TextField インスタンスを追加します。
6. プロパティインスペクタでこの TextField インスタンスのテキストタイプを [ダイナミックテキスト] に設定し、[埋め込み] をクリックして [文字の埋め込み] ダイアログボックスを開きます。
7. [文字の埋め込み] ダイアログボックスの一番下までスクロールし、作成した新しいカスタム文字セット [Uppercase and Numerals [A..Z,0..9] (38 glyphs)] を選択します。
8. 任意の他の文字セットを選択し、[OK] をクリックします。

   カスタム文字セット [Uppercase and Numerals [A..Z,0..9]] に加えてデフォルトの文字セット [大文字 [A..Z]] や [数字 [0..9]] を選択しても、埋め込まれる文字数は変わらない点に注目してください。これは、大文字の文字がすべてカスタム文字セットに含まれているためです。Flash では文字を二重に埋め込むことはしないため、ファイルサイズを極力抑えることができます。含まれる文字数が 38 のこのカスタム文字セットに加えて、文字数が 52 の文字セット [句読点記号] を選択しても、Flash では 90 文字ではなく 88 文字分の情報を保存します。これは、カスタム文字セットに空白とピリオドが既に含まれていて重複するためです。

ヒント	[文字の埋め込み] ダイアログボックスでの文字セットの位置は、XML ドキュメントで記述されている位置に基づいて決まります。XML ファイル内で ‹glyphRange› パケットを移動させることで、カスタム文字セットも含め、表示される文字セットの順序を変更できます。

埋め込みフォントでの TextField メソッドの使用

TextField クラスのメソッドにより、作成するアプリケーションにおいて有益な機能を利用することができます。たとえば、次の例のように、ActionScript を使用してテキストフィールドの深度を制御できます。

ActionScript を使用してテキストフィールドの深度を設定するには :

1. 新しい Flash ドキュメントを作成し、textfieldThickness.fla という名前で保存します。
2. [ライブラリ] パネルを開き、パネルの右上隅にあるポップアップメニューから [新しいフォント] を選択します。

   [フォントシンボルプロパティ] ダイアログボックスが表示されます。このダイアログボックスでは、SWF ファイルに埋め込むフォントを選択できます (フォントスタイルとフォントサイズも含みます)。ドキュメントのライブラリおよびプロパティインスペクタの [フォント] ドロップダウンメニューに表示されるフォント名を割り当てることもできます (ステージ上でテキストフィールドを選択している場合)。

   a. [フォント] ドロップダウンメニューから、Times New Roman フォントを選択します。
   b. [ボールド] および [イタリック] オプションをオフにしてください。
   c. サイズを 30 ピクセルに設定します。
   d. フォント名を「Times (embedded)」と入力します。
   e. [OK] をクリックします。
3. [ライブラリ] パネルで、フォントシンボルを右クリックし、コンテキストメニューから [リンケージ] を選択します。

   [リンケージプロパティ] ダイアログボックスが表示されます。
4. [ActionScript に書き出し] と [最初のフレームに書き出し] オプションを選択し、[OK] をクリックします。
5. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
// 1
this.createTextField("thickness_txt", 10, 0, 0, Stage.width, 22);
this.createTextField("lorem_txt", 20, 0, 20, Stage.width, 0);
lorem_txt.autoSize = "left";
lorem_txt.embedFonts = true;
lorem_txt.antiAliasType = "advanced";
lorem_txt.text = "Lorem ipsum dolor sit amet, consectetur adipisicing elit,
  sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
  ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip
  ex ea commodo consequat. Duis aute irure dolor in reprehenderit in
  voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
  occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit
  anim id est laborum.";
lorem_txt.wordWrap = true;
```

```
// 2
var style_fmt:TextFormat = new TextFormat();
style_fmt.font = "Times (embedded)";
style_fmt.size = 30;
lorem_txt.setTextFormat(style_fmt);

// 3
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
   // TextField.thickness の値の範囲は -200 から +200 まで
   lorem_txt.thickness = Math.round(_xmouse * (400 / Stage.width) - 200);
   thickness_txt.text = "TextField.thickness = " + lorem_txt.thickness;
};
Mouse.addListener(mouseListener);
```

最初のコードブロックにより、2 つのテキストフィールド、thickness_txt と lorem_txt が作成され、ステージ上に位置付けられます。lorem_txt テキストフィールドは embedFonts プロパティを true に設定し、テキストフィールドにテキストのブロックを格納します。

2 番目のコードブロックは、フォントフェイスを Times New Roman としてテキストフォーマットを定義し、フォントサイズを 30 ピクセルに設定し、このテキストフォーマットを lorem_txt テキストフィールドに適用します。

3 番目 (最後) のコードブロックは、マウスリスナーを定義し、onMouseMove イベントに割り当てます。マウスポインタがステージを水平方向に移動すると、TextField.thickness プロパティが -200 ～ +200 の範囲で、_xmouse の現在値に応じて変化します。

6. FLA ファイルの変更を保存します。
7. [制御]-[ムービープレビュー] を選択し、Flash ドキュメントをテストします。

   マウスポインタをステージのステージの左半分側に動かすと、フォントの太さが減少します。マウスポインタをステージのステージの右半分側に動かすと、フォントの太さが増加します。

フォントの共有について

フォントを共有ライブラリのアイテムとして使用するために、[ライブラリ] パネルでフォントシンボルを作成し、次の属性を割り当てることができます。

- 識別子ストリング
- フォントシンボルを含むドキュメントのアップロード先の URL

これによってフォントとのリンクが設定され、フォントを FLA ファイル内に格納することなく、Flash アプリケーションで使用できるようになります。

フォントのレンダリングとアンチエイリアステキストについて

Flash におけるフォントのレンダリングは、テキストを SWF ファイルに表示する方法、つまり実行時にどのようにレンダリング (つまり描画) されるのかを制御します。Flash Player 8 以降では、高度なアンチエイリアスと呼ばれる高度なフォントレンダリング技術が採用されています。高度なアンチエイリアスでは、テキストフィールドに高度なアンチエイリアス処理を適用したときなどに、小〜中サイズのフォントのテキストが読みやすくクリアに表示されるよう、高度なレンダリング技術が使用されています。この技術については、このセクションの後の方でより詳しく説明します。

高度なアンチエイリアスにより、テキストを滑らかにし、画面に表示される文字のエッジがぎざぎざにならないようにすることができます。これは、小さテキストサイズを使用してテキストを表示するときに、特に有効です。テキストにアンチエイリアスオプションを設定すると、テキストのアウトラインがピクセルの境界に揃えられ、文字が読みやすくなります。これは、サイズの小さいフォントをよりクリアにレンダリングする場合に特に有効です。高度なアンチエイリアスを適用できるのは、個々の文字ではなく、アプリケーション内の各テキストフィールドに対してです。

ユーザーが Flash Player 7 以降を使用している場合、静止テキスト、ダイナミックテキスト、およびテキスト入力で高度なアンチエイリアスがサポートされます。ユーザーがそれ以前のバージョンを使用している場合、このオプションは静止テキストのみでサポートされています。Flash Player 8 以降では高度なアンチエイリアスオプションを利用できます。

Flash には、アンチエイリアスフォントを操作できるように、高度なアンチエイリアスと呼ばれる、大幅に強化されたフォントラスタライズおよびレンダリング技術が搭載されています。Flash には 5 種類のフォントレンダリングメソッドが用意されていますが、これらは Flash Player 8 以降向けに SWF ファイルをパブリッシュする場合にのみ利用可能です。Flash Player 7 以前のバージョンで使用するようファイルをパブリッシュする場合は、テキストフィールドに対して使用できるのは [アンチエイリアス (アニメーション優先)] オプションのみです。

高度なアンチエイリアスは、Flash オーサリングツールと ActionScript のいずれを利用して有効にすることも可能な、高品質なフォントレンダリング技術です。高度なアンチエイリアス技術を使用すると、小さなサイズのフォントフェイスをより高品質にレンダリングでき、より細かく制御することが可能です。高度なアンチエイリアスは、静止、ダイナミック、テキスト入力の各テキストフィールドでの埋め込みフォントのレンダリングに適用できます。この機能強化により、埋め込みテキストはデバイステキストと同レベルの品質で表示され、フォントは別のプラットフォームでも同じように表示されるようになります。

Flash Player 8 以降で利用可能なフォントレンダリングメソッドは、[デバイスフォントの使用]、[ビットマップテキスト (アンチエイリアスなし)]、[アンチエイリアス (アニメーション優先)]、[アンチエイリアス (読みやすさ優先)]、および [カスタムアンチエイリアス] です。[カスタムアンチエイリアス] を指定すると、太さおよびシャープネスのカスタム値を設定可能です。これらのオプションの詳細については、356 ページの「Flash のフォントレンダリングオプション」を参照してください。

メモ

Flash 8 以降で従来の FLA ファイルを開いても、テキストが自動的に [アンチエイリアス (読みやすさ優先)] オプションに更新されることはありません。それで、高度なアンチエイリアス技術を利用するには、個々のテキストフィールドを選択して、手動でアンチエイリアスの設定を変更する必要があります。

これらの高度なアンチエイリアス機能およびカスタムアンチエイリアス機能は、次の機能をサポートします。

- テキストの拡大 / 縮小と回転
- 225 pt サイズまでのすべてのフォント (プレーン、ボールド、イタリック)
- ほとんどのフォーマット (JPEG や GIF ファイルなど) へのファイルの書き出し

これらの高度なアンチエイリアス機能およびカスタムアンチエイリアス機能は、次の機能をサポートしていません。

- Flash Player 7 以前
- 傾斜または反転テキスト
- 印刷
- PNG ファイルフォーマットへのファイルの書き出し

メモ

テキストがアニメーション表示されるときは、プレーヤーが高度のアンチエイリアスをオフにし、動いているテキストの表示を改善します。アニメーション表示が終わると、アンチエイリアスはオンに戻ります。

アプリケーションで、アンチエイリアス処理されたテキストを適用して操作する方法を示すサンプルのソースファイル "aliasing.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript 2.0/Advanced Anti-Aliasing" フォルダに移動してサンプルにアクセスします。高度なアンチエイリアス技術を使用して、小さくても非常に読みやすいテキストを作成します。また、このサンプルは、`cacheAsBitmap` プロパティを使用することでテキストフィールドをどれだけ高速かつ円滑にスクロールできるかを示しています。

Flash のフォントレンダリングオプション

Flash では 5 種類のフォントレンダリングオプションを使用できます。オプションを選択するには、テキストフィールドを選択してプロパティインスペクタを開きます。そして [フォントのレンダリング方法] ポップアップメニューからオプションを選択します。

デバイスフォントの使用　SWF ファイルのサイズが小さくなります。このオプションでは、現在エンドユーザーのコンピュータにインストールされているフォントを使用してレンダリングします。

ビットマップテキスト (アンチエイリアスなし)　高度なアンチエイリアスが適用されず、テキストのエッジがシャープになります。このオプションでは、フォントのアウトラインが SWF ファイルに含まれるため、SWF ファイルのサイズが大きくなります。

アンチエイリアス (アニメーション優先)　滑らかなアニメーション表示が可能なアンチエイリアステキストが生成されます。テキストのアニメーション表示の際に整列とアンチエイリアスが適用されないため、場合によってはテキストのアニメーション表示が高速にもなります。大きいフォントを多くの文字に使用する場合や拡大 / 縮小フォントを使用する場合は、パフォーマンスは改善されません。このオプションでは、フォントのアウトラインが SWF ファイルに含まれるため、SWF ファイルのサイズが大きくなります。

アンチエイリアス (読みやすさ優先)　このオプションには、高度のアンチエイリアスエンジンが使用されます。このオプションを使用すると、最も読みやすい最高品質のテキストが得られます。このオプションでは、フォントのアウトラインだけでなく、特別の高度なアンチエイリアス情報も SWF ファイルに含まれるため、SWF ファイルのサイズが最も大きくなります。

カスタムアンチエイリアス　このオプションは、[アンチエイリアス (読みやすさ優先)] と同じですが、高度なアンチエイリアスパラメータを目で確認しながら手動で操作し、特別な表示を実現できます。このオプションは、新しいフォントや一般的でないフォントの最適な表示を実現するために使用できます。

ActionScript でアンチエイリアスを使用する方法の例については、357 ページの「ActionScript でのアンチエイリアスの設定」を参照してください。

連続的な線変調について

高度なアンチエイリアスフォントレンダリング技術では、距離フィールドの固有のプロパティを利用して、たとえばテキストの線の太さとエッジのシャープネスの両方を連続的に変調させるといった、CSM (Continuos Stroke Modulation : 連続的な線変調) を実現しています。CSM では 2 つのレンダリングパラメータを使用して、ADF (Adaptively sampled Distance Field) 距離の文字密度値へのマッピングを制御します。これらのパラメータの最適な値はかなり主観的なもので、ユーザーの好みやライティングの条件、表示プロパティ、書体、前景色、背景色、ポイントサイズによって決まります。ADF 距離を密度値にマップする機能には、外部カットオフ (距離がその値以下だと 0 に設定される) および内部カットオフ (距離がその値以上だと 255 など最大密度値に設定される) があります。

ActionScript でのアンチエイリアスの設定

Flash では、通常のアンチエイリアスと高度なアンチエイリアスの 2 種類が用意されています。高度なアンチエイリアスは Flash Player 8 以降でのみ利用でき、ライブラリ内のフォントを埋め込んでテキストフィールドの `embedFonts` プロパティを `true` に設定した場合にのみ使用可能です。Flash Player 8 以降では、ActionScript を使用して作成したテキストフィールドのデフォルト設定は `normal` になっています。

`TextField.antiAliasType` プロパティの値を設定するには、次のストリング値を使用します。

normal　通常のテキストのアンチエイリアスを適用します。これは、Flash Player のバージョン 7 以前で使用されているアンチエイリアスのタイプと一致します。

advanced　テキストの読みやすさを向上させて高度なアンチエイリアス (Flash Player 8 以降で使用できる) を適用します。高度なアンチエイリアスに設定すると、小さいフォントを高品質でレンダリングできます。この定数は、小さいフォントのテキストが多いアプリケーションで使用する場合に最適です。

ヒント	アドビ システムズ社では、48 ポイントより大きいフォントに対して高度なアンチエイリアスを使用することはお勧めしていません。

ActionScript を使用してアンチエイリアステキストを設定する方法については、次の例を参照してください。

高度なアンチエイリアスを使用するには :

1. 新しい Flash ドキュメントを作成し、antialiastype.fla という名前で保存します。
2. 2 つのムービークリップをステージ上に作成し、それぞれ normal_mc および advanced_mc というインスタンス名を付けます。

   これら 2 つのムービークリップは、通常のアンチエイリアスと高度なアンチエイリアスを切り替えるのに使用します。
3. [ライブラリ] パネルを開き、パネルの右上隅にあるオプションメニューから [新しいフォント] を選択します。

   [フォントシンボルプロパティ] ダイアログボックスが表示されます。このダイアログボックスでは、SWF ファイルに埋め込むフォントを (フォントスタイルとフォントサイズも含む) 選択できます。ドキュメントのライブラリおよびプロパティインスペクタの [フォント] ドロップダウンメニューに表示されるフォント名を割り当てることもできます (ステージ上でテキストフィールドを選択している場合)。

   a. [フォント] ドロップダウンメニューから、Arial フォントを選択します。
   b. [ボールド] および [イタリック] オプションが選択されていないことを確認してください。
   c. サイズを 10 ピクセルに設定します。
   d. フォント名を「Arial-10 (embedded)」と入力します。
   e. [OK] をクリックします。

4. [ライブラリ] パネルで、フォントシンボルを右クリックし、コンテキストメニューから [リンケージ] を選択します。

   [リンケージプロパティ] ダイアログボックスが表示されます。

5. [ActionScript に書き出し] オプションおよび [最初のフレームに書き出し] オプションを選択し、リンケージ識別子を「Arial-10」と入力して [OK] をクリックします。

6. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var text_fmt:TextFormat = new TextFormat();
text_fmt.font = "Arial-10";
text_fmt.size = 10;

this.createTextField("my_txt", 10, 20, 20, 320, 240);
my_txt.autoSize = "left";
my_txt.embedFonts = true;
my_txt.selectable = false;
my_txt.setNewTextFormat(text_fmt);
my_txt.multiline = true;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String) {
  if (src != undefined) {
    my_txt.text = src;
  } else {
    my_txt.text = "unable to load text file.";
  }
};
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");

normal_mc.onRelease = function() {
  my_txt.antiAliasType = "normal";
};
advanced_mc.onRelease = function() {
  my_txt.antiAliasType = "advanced";
};
```

上記のコードは 4 つの基本的な領域に分けられます。コードの最初のブロックでは新しい TextFormat オブジェクトを作成します。このオブジェクトでは、このすぐ後に作成されるテキストフィールドで使用されるフォントおよびフォントサイズが指定されます。指定されているフォント Arial-10 は、前の手順で埋め込んだフォントシンボルのリンケージ識別子です。

コードの 2 番目のブロックでは、インスタンス名 `my_txt` という新しいテキストフィールドが作成されます。フォントが正しく埋め込まれるよう、このテキストフィールドインスタンスについて `embedFonts` を `true` に設定する必要があります。このコードではまた、新しいテキストフィールドのテキストフォーマットとして、前に作成した TextFormat オブジェクトを設定します。

コードの 3 番目のブロックでは、ステージ上のテキストフィールドに外部テキストファイルの内容を流し込む LoadVars インスタンスが定義されています。ドキュメントが完全にロードされたら (ただし未解析)、ファイルの内容全体が、ステージに表示できるよう `my_txt.text` プロパティにコピーされます。

コードの 4 番目 (最後) のブロックでは、`normal_mc` ムービークリップおよび `advanced_mc` ムービークリップの両方のために `onRelease` イベントハンドラが定義されています。ユーザーがいずれかのオプションをクリックしてマウスボタンを離すと、ステージ上のテキストフィールドのアンチエイリアス処理の種類が変わります。

7. FLA ファイルの変更を保存します。
8. [制御]-[ムービープレビュー] を選択し、Flash ドキュメントをテストします。
9. ステージ上のムービークリップ `advanced_mc` をクリックします。

   ムービークリップをクリックすると、アンチエイリアス処理の種類が通常のアンチエイリアス (デフォルト) から高度なアンチエイリアスに切り替わります。テキストフィールドに小さなフォントサイズでフォントを表示する場合は、高度なアンチエイリアスに設定するとテキストの読みやすさが大きく向上します。

ヒント	高度なアンチエイリアスに設定すると、小さいフォントを高品質でレンダリングできます。この機能は、小さいフォントのテキストが多いアプリケーションで使用する場合に最適です。アドビ システムズ社では、48 ポイントより大きいフォントに対して高度なアンチエイリアスを使用することはお勧めしていません。

アンチエイリアステキストの書式設定の詳細については、364 ページの「グリッドフィットタイプの使用」および 362 ページの「アンチエイリアステキストのフォーマットについて」を参照してください。

アプリケーションで、テキストのアンチエイリアスを適用して操作する方法を示すサンプルのソースファイル "aliasing.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Advanced Anti-Aliasing" フォルダに移動してサンプルにアクセスします。高度なアンチエイリアス技術を使用して、小さくても非常に読みやすいテキストを作成します。また、このサンプルは、`cacheAsBitmap` プロパティを使用することでテキストフィールドをどれだけ高速かつ円滑にスクロールできるかを示しています。

フォントのテーブルの設定

SWF ファイルで使用する、あるいは Flash 開発者向けに配布する目的でフォントを作成する場合、ステージにフォントがどのようにレンダリングされるかを制御できるよう、フォントのテーブルを設定する必要が生じる場合があります。

高度なアンチエイリアスは、ADF (Adaptively sampled Distance Fields) を使用して、字形 (文字) を決定するアウトラインを表します。Flash で使われる値は次の 2 つです。

- 外部カットオフ値。これより密度が低い場合は 0 に設定されます。
- 内部カットオフ値。これより密度が大きいと、255 など、最大密度値に設定されます。

これら 2 つのカットオフ値間でマッピング関数を実行すると、外部カットオフの 0 から内部カットオフの最大密度までの直線になります。

外部カットオフ値と内部カットオフ値を調整すると、線の太さとエッジのシャープネスに影響します。これら 2 つのパラメータ間の間隔は、標準のアンチエイリアスメソッドのフィルタ半径の 2 倍に相当します。間隔が狭ければ、エッジがシャープになり、間隔が広ければ、エッジがぼやけて、フィルタの適用度が高い状態になります。間隔が 0 の場合、結果の密度イメージは 2 層のビットマップとなります。間隔が非常に広ければ、結果の密度イメージのエッジは水彩画のようになります。

通常、ユーザーは、小さなポイントサイズではシャープでコントラストのはっきりしたエッジを、アニメーションテキストや大きなポイントサイズではぼやけたエッジを好みます。

通常、外部カットオフは負の値、内部カットオフは正の値で、中間点は 0 の近辺になります。これらのパラメータを調整して、中間点を負の無限大に向かってシフトすると、線の太さが増します。中間点を正の無限大に向かってシフトすると、線の太さは減ります。

メモ	外部カットオフは、常に内部カットオフ以下にする必要があります。

Flash Player では、10 個の基本的なフォントについて高度なアンチエイリアス設定があります。高度なアンチエイリアス設定は、サイズが 6 ～ 20 のフォントに対してのみ用意されています。これらのフォントでは、6 より小さいサイズはすべて 6 用の設定を使用し、20 より大きいサイズはすべて 20 用の設定を使用します。他のフォントは、提供されたフォントデータにマッピングされます。`setAdvancedAntialiasingTable()` メソッドを使用すると、他のフォントやサイズ設定されたフォントに対してカスタムのアンチエイリアスデータを設定するか、提供されたフォントのデフォルトの設定をオーバーライドできます。アンチエイリアステーブルの作成の詳細については、次の例を参照してください。

埋め込みフォントに対する高度なアンチエイリアステーブルを作成するには :

1. 新しい Flash ドキュメントを作成し、advancedaatable.fla という名前で保存します。
2. パネルの [ライブラリ] ポップアップメニューから [新しいフォント] を選択します。
3. [フォント] ポップアップメニューから [Arial] を選択し、フォントサイズを 32 ポイントに設定します。
4. [ボールド] および [イタリック] オプションを両方とも選択します。
5. [名前] テキストボックスにフォント名を「Arial (embedded)」と入力し、[OK] をクリックします。
6. ライブラリ内のフォントシンボルを右クリック (Windows) または Control キーを押しながらクリック (Macintosh) し、[リンケージ] を選択します。
7. [リンケージプロパティ] ダイアログボックスで次の作業を実行します。
   a. [識別子] テキストボックスに「Arial-embedded」と入力します。
   b. [ActionScript に書き出し] および [最初のフレームに書き出し] を選択します。
   c. [OK] をクリックします。

8. メインタイムラインのフレーム1を選択し、[アクション] パネルに次の ActionScript を追加します。

```
import flash.text.TextRenderer;
var arialTable:Array = new Array();
arialTable.push({fontSize:16.0, insideCutoff:0.516, outsideCutoff:0.416});
arialTable.push({fontSize:32.0, insideCutoff:2.8, outsideCutoff:-2.8});
TextRenderer.setAdvancedAntialiasingTable("Arial", "bolditalic", "dark",
  arialTable);

var my_fmt:TextFormat = new TextFormat();
my_fmt.align = "justify";
my_fmt.font = "Arial-embedded";
my_fmt.size = 32;

this.createTextField("my_txt", 999, 10, 10, Stage.width-20, Stage.height-20);
my_txt.antiAliasType = "advanced";
my_txt.embedFonts = true;
my_txt.multiline = true;
my_txt.setNewTextFormat(my_fmt);
my_txt.sharpness = 0;
my_txt.thickness = 0;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String):Void {
  if (src != undefined) {
    my_txt.text = src + "\n\n" + src;
  } else {
    trace("error downloading text file");
  }
};
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

上記のコードは 4 つのセクションに分けられます。コードの最初のセクションでは、TextRenderer クラスを読み込み、サイズの異なる 2 つの Arial フォント用の新しいアンチエイリアステーブルを定義しています。コードの 2 番目のセクションでは新しい TextFormat オブジェクトを定義しています。このオブジェクトは (コードの次のセクションで作成される) テキストフィールドにテキストフォーマットを適用するのに使用されます。コードの次のセクションでは、my_txt というインスタンス名のテキストフィールドを新しく作成し、高度なアンチエイリアスを有効にし、(前の部分で作成した) テキストフォーマットオブジェクトを適用し、そして複数行のテキストおよび折り返しを有効にしています。コードの最後のブロックでは、外部テキストファイルからテキストをロードしてステージ上のテキストフィールドに流し込むのに使用する LoadVars オブジェクトを定義しています。

9. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

テキストがリモートサーバーからロードされたら、Flash によってテキストフィールドにテキストがロードされ、テキストフィールドに高度のアンチエイリアステーブルのプロパティが適用されたことを確認できます。ステージ上の内蔵フォントは、現在の insideCutoff および outsideCutoff 値により、わずかにぼかし効果が適用されたかのように表示されます。

テキストのレイアウトとフォーマットについて

ActionScript を使用して、テキストレイアウトおよびフォーマットを制御できます。TextFormat クラスを使用すれば、スタイルシート (369 ページの「CSS スタイルによるテキストのフォーマット」を参照) や HTML テキスト (382 ページの「HTML 形式のテキストの使用」を参照) などの他のフォーマット形式と同様、実行時にテキストがどのように表示されるかを細かく制御できます。

ActionScript を使用することで、SWF ファイル内でアンチエイリアステキストを使用する際に文字をグリッドに合わせる方法を制御することもできます。この方法は、実行時に文字の表示を制御するのに有効です。アプリケーションでグリッドフィットタイプを使用する方法の例については、364 ページの「グリッドフィットタイプの使用」を参照してください。

テキストフィールドについての一般的な情報については、333 ページの「テキストフィールドについて」を参照してください。テキストのフォーマットの詳細については、362 ページの「アンチエイリアステキストのフォーマットについて」を参照してください。TextFormat クラスの詳細については、367 ページの「TextFormat クラスの使用」および『ActionScript 2.0 リファレンスガイド』の「`TextFormat`」を参照してください。

TextFormat クラスを使用したテキストレイアウトおよびテキストフォーマットの詳細については、次のセクションを参照してください。

- 362 ページの「アンチエイリアステキストのフォーマットについて」
- 364 ページの「グリッドフィットタイプの使用」
- 367 ページの「TextFormat クラスの使用」
- 368 ページの「新規テキストフィールドのデフォルトプロパティ」

アンチエイリアステキストのフォーマットについて

Flash では、高度なアンチエイリアスが有効になっているテキストフィールドをフォーマットする際に使用できる 2 つのプロパティ、`sharpness` および `thickness` が用意されています。sharpness は、テキストフィールドインスタンスに適用されるエイリアス処理の量を示します。sharpness の値が大きくなると、埋め込みフォントのエッジはぎざぎざかつシャープに表示されます。sharpness の値が低くなると、フォントのエッジはそれだけソフトになり、ぼかしがかかって表示されます。フォントの thickness 設定は、テキストフィールドにボールドフォーマットを適用するのと似ています。thickness の値が大きくなると、それだけフォントは太く表示されます。

次の例では、テキストファイルを動的にロードしてステージにテキストを表示します。マウスポインタを x 軸に沿って動かすと sharpness の値を -400 ～ 400 の間で設定できます。マウスポインタを y 軸に沿って動かすと thickness の値を -200 ～ 200 の間で設定できます。

テキストフィールドの sharpness と thickness の値を修正するには :

1. 新しい Flash ドキュメントを作成し、sharpness.fla という名前で保存します。
2. [ライブラリ] パネルの右上隅にあるポップアップメニューから [新しいフォント] を選択します。
3. [フォント] ドロップダウンメニューから [Arial] を選択し、フォントサイズを 24 ポイントに設定します。
4. [名前] テキストボックスにフォント名を「Arial-24 (embedded)」と入力し、[OK] をクリックします。
5. ライブラリでフォントシンボルを右クリックし、[リンケージ] をクリックして [リンケージプロパティ] ダイアログボックスを開きます。
6. リンケージ識別子を [Arial-24] に設定し、[ActionScript に書き出し] および [最初のフレームに書き出し] チェックボックスを選択して [OK] をクリックします。
7. メインタイムラインのフレーム 1 に次のコードを追加します。

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.size = 24;
my_fmt.font = "Arial-24";

this.createTextField("lorem_txt", 10, 0, 20, Stage.width, (Stage.height -
  20));
lorem_txt.setNewTextFormat(my_fmt);
lorem_txt.text = "loading...";
lorem_txt.wordWrap = true;
lorem_txt.autoSize = "left";
lorem_txt.embedFonts = true;
lorem_txt.antiAliasType = "advanced";

this.createTextField("debug_txt", 100, 0, 0, Stage.width, 20);
debug_txt.autoSize = "left";
debug_txt.background = 0xFFFFFF;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String) {
  lorem_txt.text = src;
}
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  lorem_txt.sharpness = (_xmouse * (800 / Stage.width)) - 400;
  lorem_txt.thickness = (_ymouse * (400 / Stage.height)) - 200;
  debug_txt.text = "sharpness=" + Math.round(lorem_txt.sharpness) +
    ", thickness=" + Math.round(lorem_txt.thickness);
};
Mouse.addListener(mouseListener);
```

この ActionScript コードは 5 つのセクションに大きく分けることができます。コードの最初のセクションでは、動的に作成されるテキストフィールドに適用される TextFormat インスタンスを新しく定義しています。その次の 2 つのセクションでは、ステージ上に 2 つの新しいテキストフィールドを作成しています。最初のテキストフィールド、`lorem_txt` に対しては、その前に作成したカスタムテキストフォーマットオブジェクトを適用し、埋め込みフォントが有効になるよう設定し、そして `antiAliasType` プロパティを `true` に設定します。2 番目のテキストフィールド、`debug_txt` には、`lorem_txt` テキストフィールドの現在の sharpness および thicknesstext 値が表示されます。コードの 4 番目のセクションでは LoadVars オブジェクトを作成します。これは外部テキストファイルをロードして `lorem_txt` テキストフィールドに流し込むためのものです。コードの 5 番目の、最後のセクションでは、マウスポインタがステージ上を移動するたびに呼び出されるマウスリスナーを定義しています。`sharpness` および `thickness` の現在の値は、ステージ上の現在のマウスポインタの位置に基づいて計算されます。`sharpness` および `thickness` プロパティは `lorem_txt` テキストフィールドに対して設定され、プロパティの現在の値が `debug_txt` テキストフィールドに表示されます。

8. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   マウスポインタを x 軸に沿って動かすとテキストフィールドの sharpness 値が変わります。マウスポインタを左から右に動かすと sharpness 値は大きくなり、ぎざぎざが強くなります。マウスポインタを y 軸に沿って動かすとテキストフィールドの thickness 値が変わります。

SWF ファイルでのアンチエイリアステキストの使用について詳しくは、357 ページの「ActionScript でのアンチエイリアスの設定」、356 ページの「Flash のフォントレンダリングオプション」、および 364 ページの「グリッドフィットタイプの使用」を参照してください。

アプリケーションで、テキストのアンチエイリアスを適用して操作する方法を示すサンプルのソースファイル "aliasing.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript 2.0/Advanced Anti-Aliasing" フォルダに移動してサンプルにアクセスします。高度なアンチエイリアス技術を使用して、小さくても非常に読みやすいテキストを作成します。また、このサンプルは、`cacheAsBitmap` プロパティを使用することでテキストフィールドをどれだけ高速かつ円滑にスクロールできるかを示しています。

グリッドフィットタイプの使用

テキストフィールドに対して高度なアンチエイリアスを使用すると、次の 3 種類のグリッドフィッティングが使用できます。

none　グリッドフィッティングを指定しません。文字の水平方向と垂直方向の線が、ピクセルグリッドに合わせられることはありません。この設定は多くの場合、アニメーションや、大きなフォントサイズに適した設定です。

pixel　太い水平線と垂直線がピクセルグリッドに合わされるよう指定します。この設定は左揃えのテキストフィールドに対してのみ機能します。左揃えのテキストは通常、この設定を使用すると、最も読みやすくなります。

subpixel　太い水平線と垂直線が、LCD モニタのサブピクセルグリッドに合わされるよう指定します。subpixel 設定は通常、右揃えや中央揃えのダイナミックテキストに適した設定で、アニメーション品質とテキスト品質のバランスを取るのに便利な場合もあります。

次に、ActionScript を使用してテキストフィールドにグリッドフィットタイプを設定する方法を示します。

テキストフィールドにグリッドフィットタイプを設定するには :

1. 新しい Flash ドキュメントを作成し、gridfittype.fla という名前で保存します。
2. [ライブラリ] パネルの右上隅にあるポップアップメニューから [新しいフォント] を選択します。
3. [フォント] ドロップダウンメニューから [Arial] フォントを選択し、フォントサイズを 10 ポイントに設定します。
4. [名前] テキストボックスにフォント名を「Arial-10 (embedded)」と入力し、[OK] をクリックします。
5. ライブラリでフォントシンボルを右クリックし、[リンケージ] をクリックして [リンケージプロパティ] ダイアログボックスを開きます。
6. リンケージ識別子を [Arial-10] に設定し、[ActionScript に書き出し] および [最初のフレームに書き出し] チェックボックスを選択します。
7. [OK] をクリックします。
8. メインタイムラインのフレーム 1 に次のコードを追加します。

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.size = 10;
my_fmt.font = "Arial-10";
var h:Number = Math.floor(Stage.height / 3);

this.createTextField("none_txt", 10, 0, 0, Stage.width, h);
none_txt.antiAliasType = "advanced";
none_txt.embedFonts = true;
none_txt.gridFitType = "none";
none_txt.multiline = true;
none_txt.setNewTextFormat(my_fmt);
none_txt.text = "loading...";
none_txt.wordWrap = true;

this.createTextField("pixel_txt", 20, 0, h, Stage.width, h);
pixel_txt.antiAliasType = "advanced";
pixel_txt.embedFonts = true;
pixel_txt.gridFitType = "pixel";
pixel_txt.multiline = true;
pixel_txt.selectable = false;
pixel_txt.setNewTextFormat(my_fmt);
pixel_txt.text = "loading...";
pixel_txt.wordWrap = true;
```

```
this.createTextField("subpixel_txt", 30, 0, h*2, Stage.width, h);
subpixel_txt.antiAliasType = "advanced";
subpixel_txt.embedFonts = true;
subpixel_txt.gridFitType = "subpixel";
subpixel_txt.multiline = true;
subpixel_txt.setNewTextFormat(my_fmt);
subpixel_txt.text = "loading...";
subpixel_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String):Void {
  if (src != undefined) {
    none_txt.text = "[antiAliasType=none]\n" + src;
    pixel_txt.text = "[antiAliasType=pixel]\n" + src;
    subpixel_txt.text = "[antiAliasType=subpixel]\n" + src;
  } else {
    trace("unable to load text file");
  }
};
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

上記の ActionScript コードは 5 つのセクションに分けることができます。最初のセクションでは、size および font の 2 つのプロパティを指定する新しいテキストフォーマットオブジェクトを定義しています。font プロパティは、現在ドキュメントライブラリ内にあるフォントシンボルのリンケージ識別子を参照します。コードの 2 番目、3 番目、4 番目のセクションではそれぞれ、ステージ上に新しいダイナミックテキストフィールドを作成し、antiAliasType (必ず advanced に設定)、embedFonts (true に設定)、multiline および wordWrap の各共通プロパティを設定します。各セクションではまた、前のセクションで作成したテキストフォーマットオブジェクトを適用し、グリッドフィットタイプを normal、pixel、subpixel のいずれかに設定します。5 番目、最後のセクションでは LoadVars インスタンスを作成します。これにより、外部テキストファイルのコンテンツが、コードで作成した各テキストフィールドにロードされます。

9. ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

各テキストフィールドは値 "loading..." で初期化されます。外部テキストファイルのロードに成功したら、フォーマット済みのサンプルテキストが、さまざまなグリッドフィットタイプを使用して各テキストフィールドに表示されます。

ヒント	高度なアンチエイリアス技術では、回転が 0 度の場合にのみグリッドフィッティングが使用されます。

TextFormat クラスの使用

TextFormat クラスを使用して、テキストフィールドのフォーマットプロパティを設定できます。TextFormat クラスには、文字と段落の書式情報が組み込まれています。文字の書式情報では、フォント名、ポイントサイズ、カラー、および関連 URL など、各文字の外観について記述しています。段落の書式情報では、左マージン、右マージン、先頭行のインデント、左右または中央の行揃えなど、段落の外観について記述しています。

TextFormat クラスを使用するには、まず TextFormat オブジェクトを作成し、その文字フォーマットと段落フォーマットのスタイルを設定します。その後、`TextField.setTextFormat()` メソッドまたは `TextField.setNewTextFormat()` メソッドを使用して、テキストフィールドに TextFormat オブジェクトを適用します。

`setTextFormat()` メソッドは、テキストフィールドの各文字、文字のグループ、またはテキスト全体に適用するテキストフォーマットを変更します。ただし、新たに挿入されたテキスト (ユーザーが入力したテキストや、ActionScript で挿入したテキストなど) には、`setTextFormat()` 呼び出しで指定したフォーマットは反映されません。新たに挿入されたテキストのデフォルトのフォーマットを指定するには、`TextField.setNewTextFormat()` を使用します。詳細については、『ActionScript 2.0 リファレンスガイド』の「setTextFormat (TextField.setTextFormat メソッド)」および「setNewTextFormat (TextField.setNewTextFormat メソッド)」を参照してください。

テキストフィールドを TextFormat クラスでフォーマットするには :

1. 新しい Flash ドキュメント内で、テキストツールを使用してステージ上に新しいテキストフィールドを作成します。

   ステージ上のテキストフィールドに、「Bold, italic, 24 point text」などのテキストを入力します。

2. プロパティインスペクタ内で、[インスタンス名] テキストボックスに「myText_txt」と入力し、[テキストの種類] ポップアップメニューから [ダイナミックテキスト] を選択し、[行タイプ] ポップアップメニューから [複数行] を選択します。

3. タイムラインのフレーム 1 を選択し、[アクション] パネルを開きます ([ウィンドウ]-[アクション])。

4. [アクション] パネルに次のコードを入力します。このコードでは、TextFormat オブジェクトを作成し、その `bold` および `italic` プロパティを `true` に設定し、`size` プロパティを 24 に設定します。

```
// TextFormat オブジェクトを作成
var txt_fmt:TextFormat = new TextFormat();
// 段落および文字のフォーマットを指定
txt_fmt.bold = true;
txt_fmt.italic = true;
txt_fmt.size = 24;
```

5. `TextField.setTextFormat()` を使用して、手順 1 で作成したテキストフィールドにこの TextFormat オブジェクトを適用します。

```
myText_txt.setTextFormat(txt_fmt);
```

このバージョンの `setTextFormat()` は、指定のフォーマットをテキストフィールド全体に適用します。このメソッドの他の 2 つのバージョンでは、個々の文字または文字のグループにフォーマットを適用できます。たとえば、次のコードでは、テキストフィールドに入力されている最初の 3 文字に、ボールド、イタリック、24 ポイントというフォーマットを適用します。

```
myText_txt.setTextFormat(0, 3, txt_fmt);
```

詳細については、『ActionScript 2.0 リファレンスガイド』の「setTextFormat (TextField.setTextFormat メソッド)」を参照してください。

6. [制御]-[ムービープレビュー] を選択してアプリケーションをテストします。

TextFormat クラスの使用方法の詳細については、次のトピックを参照してください。

- 368 ページの「新規テキストフィールドのデフォルトプロパティ」
- 369 ページの「CSS スタイルによるテキストのフォーマット」

新規テキストフィールドのデフォルトプロパティ

実行時に `createTextField()` で作成したテキストフィールドは、次のプロパティを備えたデフォルトの TextFormat オブジェクトを受け取ります。

```
align = "left"
blockIndent = 0
bold = false
bullet = false
color = 0x000000
font = "Times New Roman" (default font is Times on Mac OS X)
indent = 0
italic = false
kerning = false
leading = 0
leftMargin = 0
letterSpacing = 0
rightMargin = 0
size = 12
tabStops = [] (empty array)
target = ""
underline = false
url = ""
```

メモ

Mac OS X でのデフォルトフォントプロパティは Times です。

TextFormat のメソッドの一覧と各メソッドの詳細については、『ActionScript 2.0 リファレンスガイド』の「TextFormat」を参照してください。

CSS スタイルによるテキストのフォーマット

CSS (Cascading Style Sheet : カスケーディングスタイルシート) スタイルは、HTML ドキュメントや XML ドキュメントに適用できるテキストスタイルを扱う 1 つの方法です。スタイルシートは、HTML エレメントや XML エレメントのフォーマット方法を指定するフォーマット規則の集まりです。個々の規則は、スタイル名 (またはセレクタ) を、1 つ以上のスタイルプロパティとその値に関連付けます。たとえば、次のスタイルでは、bodyText という名前のセレクタを定義しています。

```
.bodyText {
  text-align: left
}
```

Flash Player で使用されるビルトイン HTML フォーマットタグ (<p> や <li> など) を再定義するスタイルを作成できます。また、<p> タグや タグの class 属性を使って特定の HTML 要素に適用できるスタイルの " クラス " の作成や新しいタグの定義が可能です。

テキストスタイルシートを操作するには、TextField.StyleSheet クラスを使用します。TextField クラスは Flash Player 6 で使用できますが、TextField.StyleSheet クラスの場合は SWF ファイルが Flash Player 7 以降を対象にしている必要があります。これにより、外部の CSS ファイルからのスタイルのロードや、ActionScript を使用して内部でのスタイルの作成ができます。HTML または XML 形式のテキストを含んでいるテキストフィールドにスタイルシートを適用するには、TextField.styleSheet プロパティを使用します。スタイルシート内で定義されているスタイルが、HTML または XML ドキュメント内に定義されているタグに自動的にマップされます。

スタイルシートを使用するには、次の 3 段階の手順が必要です。

- TextField.StyleSheet クラスからスタイルシートオブジェクトを作成します。詳細については、『ActionScript 2.0 リファレンスガイド』の「StyleSheet (TextField.StyleSheet)」を参照してください。
- 外部の CSS ファイルからスタイルをロードするか、ActionScript で新しいスタイルを作成して、スタイルシートオブジェクトにスタイルを追加します。
- このスタイルシートを、HTML または XML 形式のテキストを含んでいる TextField オブジェクトに割り当てます。

詳細については、次のトピックを参照してください。

- 370 ページの「サポートされている CSS プロパティ」
- 371 ページの「スタイルシートオブジェクトの作成」
- 372 ページの「外部 CSS ファイルのロード」
- 373 ページの「ActionScript による新規スタイルの作成」
- 374 ページの「TextField オブジェクトへのスタイルの適用」
- 374 ページの「TextArea コンポーネントへのスタイルシートの適用」
- 375 ページの「スタイルの組み合わせ」

■ 376 ページの「スタイルクラスの使用」
■ 376 ページの「ビルトイン HTML タグのスタイル設定」
■ 377 ページの「HTML でのスタイルの使用例」
■ 379 ページの「スタイルによる新規タグの定義」
■ 380 ページの「XML でのスタイルの使用例」

実行時に SWF ファイルにロードするテキストに CSS フォーマットを適用する方法を示すサンプルのソースファイル "formattedText.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/LoadText" フォルダに移動してサンプルにアクセスします。

サポートされている CSS プロパティ

Flash Player は、オリジナルの CSS1 仕様 (www.w3.org/TR/REC-CSS1) にあるプロパティのサブセットをサポートしています。次の表に、サポートされている CSS プロパティとその値、および対応する ActionScript プロパティの名前を示します。ActionScript プロパティ名は、どれも対応する CSS プロパティ名から派生しており、ハイフンを省略し、後ろの文字を大文字にしています。

CSS プロパティ	ActionScript プロパティ	用途とサポートされる値
text-align	textAlign	有効な値は、`left`、`center`、`right`、および `justify` です。
font-size	fontSize	値の数字の部分だけを使用します。単位 (px、pt) は解析されません。ピクセルとポイントは同じ意味になります。
text-decoration	textDecoration	有効な値は、`none` と `underline` です。
margin-left	marginLeft	値の数字の部分だけを使用します。単位 (px、pt) は解析されません。ピクセルとポイントは同じ意味になります。
margin-right	marginRight	値の数字の部分だけを使用します。単位 (px、pt) は解析されません。ピクセルとポイントは同じ意味になります。
font-weight	fontWeight	有効な値は、`normal` と `bold` です。
kerning	kerning	有効な値は、`true` と `false` です。
font-style	fontStyle	有効な値は、`normal` と `italic` です。
letter-spacing	letterSpacing	値の数字の部分だけを使用します。単位 (px、pt) は解析されません。ピクセルとポイントは同じ意味になります。

CSS プロパティ	ActionScript プロパティ	用途とサポートされる値
text-indent	textIndent	値の数字の部分だけを使用します。単位 (px、pt) は解析されません。ピクセルとポイントは同じ意味になります。
font-family	fontFamily	使用するフォントをカンマ区切りリストで指定します。優先度の高い順に並べます。任意のフォントファミリ名を使用できます。汎用フォント名を指定した場合、適切なデバイスフォントに置換されます。次のようなフォント変換が行われます。`mono` は `_typewriter` に、`sans-serif` は `_sans` に、`serif` は `_serif` に変換されます。
color	color	16 進数のカラー値のみがサポートされます。`blue` などの名前付きカラーはサポートされません。カラーは、`#FF000` のようなフォーマットで記述されます。
display	display	サポートされる値は `inline`、`block`、および `none` です。

XML エレメントでのスタイルの使用例については、380 ページの「XML でのスタイルの使用例」を参照してください。

スタイルシートオブジェクトの作成

ActionScript 内では、CSS を TextField.StyleSheet クラスで表します。このクラスは、Flash Player 7 以降を対象とする SWF ファイルでのみ使用できます。スタイルシートオブジェクトを作成するには、TextField.StyleSheet クラスのコンストラクタ関数を呼び出します。

```
var newStyle:TextField.StyleSheet = new TextField.StyleSheet();
```

スタイルシートオブジェクトにスタイルを追加するには、外部の CSS ファイルをオブジェクトにロードするか、ActionScript 内でスタイルを定義します。詳細については、372 ページの「外部 CSS ファイルのロード」および 373 ページの「ActionScript による新規スタイルの作成」を参照してください。

実行時に SWF ファイルにロードするテキストに CSS フォーマットを適用する方法を示すサンプルのソースファイル "formattedText.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/LoadText" フォルダに移動してサンプルにアクセスします。

外部 CSS ファイルのロード

外部の CSS ファイル内でスタイルを定義しておき、そのファイルをスタイルシートオブジェクトにロードすることができます。CSS ファイル内で定義したスタイルがスタイルシートオブジェクトに追加されます。外部 CSS ファイルをロードするには、TextField.StyleSheet クラスの `load()` メソッドを使用します。CSS ファイルのロードの完了を判断するには、スタイルシートオブジェクトの `onLoad` イベントハンドラを使用します。

次の例では、外部 CSS ファイルを作成してロードし、`TextField.StyleSheet.getStyleNames()` メソッドを使用して、ロードしたスタイルの名前を取得します。

外部スタイルシートをロードするには :

1. 任意のテキストエディタまたは CSS エディタを使用して、ファイルを作成します。
2. 作成したファイルに次のスタイル定義を追加します。

```
.bodyText {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 12px;
}

.headline {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 24px;
}
```

3. この CSS ファイルを "styles.css" として保存します。
4. Flash で新規 FLA ファイルを作成します。
5. タイムライン内で ([ウィンドウ]-[タイムライン])、レイヤー 1 を選択します。
6. [アクション] パネルを開きます ([ウィンドウ]-[アクション])。
7. [アクション] パネルに次のコードを追加します。

```
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.onLoad = function(success:Boolean):Void {
  if (success) {
    // スタイル名を表示
    trace(this.getStyleNames());
  } else {
    trace("Error loading CSS file.");
  }
};
styles.load("styles.css");
```

メモ	前のコードでは、this.getStyleNames() は ActionScript の 1 行目で作成した styles オブジェクトを参照します。

8. この FLA ファイルを "styles.css" と同じディレクトリに保存します。

9. Flash ドキュメントをテストします ([制御]-[ムービープレビュー])。

   [出力] パネルに次の 2 つのスタイル名が表示されるはずです。

   ```
   .bodyText,.headline
   ```

   [出力] パネルに "CSS ファイルのロード中にエラーが発生しました" と表示された場合は、FLA ファイルと CSS ファイルを同じディレクトリに保存しているか、CSS ファイルの名前を正しく指定しているかを確認してください。

ネットワーク経由でデータをロードするその他すべての ActionScript メソッドと同様に、この場合も、CSS ファイルとそれをロードする SWF ファイルを同じドメイン内に配置する必要があります。詳細については、627 ページの「ネットワーク API の制限」を参照してください。Flash で CSS を使用する方法の詳細については、『ActionScript 2.0 リファレンスガイド』の「StyleSheet (TextField.StyleSheet)」を参照してください。

実行時に SWF ファイルにロードするテキストに CSS フォーマットを適用する方法を示すサンプルのソースファイル "formattedText.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/LoadText" フォルダに移動してサンプルにアクセスします。

ActionScript による新規スタイルの作成

TextField.StyleSheet クラスの `setStyle()` メソッドを使用すると、新しいテキストスタイルを ActionScript で作成することができます。このメソッドには 2 つのパラメータがあります。1 つはスタイル名で、もう 1 つはスタイルのプロパティを定義するオブジェクトです。

たとえば、次のコードでは、`styles` という名前のスタイルシートオブジェクトを作成します。これに定義されている 2 つのスタイルは、既に読み込んだスタイルとまったく同じものです (372 ページの「外部 CSS ファイルのロード」を参照)。

```
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.setStyle("bodyText",
  {fontFamily: 'Arial,Helvetica,sans-serif',
  fontSize: '12px'}
);
styles.setStyle("headline",
  {fontFamily: 'Arial,Helvetica,sans-serif',
  fontSize: '24px'}
);
```

TextField オブジェクトへのスタイルの適用

TextField オブジェクトにスタイルシートオブジェクトを適用するには、テキストフィールドの styleSheet プロパティにスタイルシートオブジェクトを割り当てます。

```
textObj_txt.styleSheet = styles;
```

注	TextField.styleSheet プロパティと TextField.StyleSheet クラスを混同しないように注意してください。このプロパティとクラスは、大文字と小文字で違いを示しています。

スタイルシートオブジェクトを TextField オブジェクトに割り当てると、テキストフィールドの通常のビヘイビアに次の変化が起きます。

- テキストフィールドの text プロパティと htmlText プロパティ、およびそのテキストフィールドに関連付けられている変数は、常にそれまでとまったく同じ値とビヘイビアを維持します。
- テキストフィールドは読み取り専用になり、ユーザーが編集することはできません。
- TextField クラスの setTextFormat() メソッドと replaceSel() メソッドがそのテキストフィールドに対して機能しなくなります。テキストフィールドを変更するには、テキストフィールドの text プロパティまたは htmlText プロパティを変更するか、テキストフィールドに関連付けられている変数を変更するしかありません。
- テキストフィールドの text プロパティ、htmlText プロパティ、または関連付けられた変数に割り当てられているテキストは、そのままの状態で保存されます。これらのプロパティに書き込まれているテキストは、元の状態のまま取得できます。

TextArea コンポーネントへのスタイルシートの適用

TextArea コンポーネントにスタイルシートを適用するには、スタイルシートオブジェクトを作成し、TextField.StyleSheet クラスを使用して HTML スタイルを割り当てます。次に、TextArea コンポーネントの styleSheet プロパティにスタイルシートを割り当てます。

次の例では、スタイルシートオブジェクト styles を作成し、それを myTextArea コンポーネントインスタンスに割り当てます。

TextArea コンポーネントへのスタイルシートの適用

1. 新しい Flash ドキュメントを作成し、textareastyle.fla という名前を付けて保存します。
2. TextArea コンポーネントを [コンポーネント] パネルの "User Interface" フォルダからステージまでドラッグし、インスタンス名を myTextArea とします。
3. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
// 新しい StyleSheet オブジェクトを作成し、スタイルを設定する。
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.setStyle("html", {fontFamily:'Arial,Helvetica,sans-serif',
            fontSize:'12px',
            color:'#0000FF'});
```

```
styles.setStyle("body", {color:'#00CCFF',
             textDecoration:'underline'});
styles.setStyle("h1",{fontFamily:'Arial,Helvetica,sans-serif',
             fontSize:'24px',
             color:'#006600'});

/* myTextArea コンポーネントにスタイルシートオブジェクトを割り当てる。html プロパティを
   true に設定し、styleSheet プロパティをスタイルシートオブジェクトに設定する。*/
myTextArea.styleSheet = styles;
myTextArea.html = true;

var myVars:LoadVars = new LoadVars();
// onData ハンドラを定義し、表示するテキストをロードする。
myVars.onData = function(myStr:String):Void {
   if (myStr != undefined) {
      myTextArea.text = myStr;
   } else {
      trace("Unable to load text file.");
   }
};
myVars.load("http://www.helpexamples.com/flash/myText.htm");
```

上記のコードでは、HTML タグ html、body、および h1 の 3 つのスタイルを定義する新しい TextField.StyleSheet インスタンスを定義しています。次に、スタイルシートオブジェクトを TextArea コンポーネントに対して適用し、HTML フォーマットを有効にします。ActionScript 残りの部分では、外部 HTML ファイルをロードし、ロードされたテキストをテキスト領域に流し込む、LoadVars オブジェクトを定義しています。

4. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

スタイルの組み合わせ

Flash Player における CSS スタイルは加算的です。つまり、スタイルをネストさせると、ネストの各レベルでスタイル情報を追加していくことになり、それらをすべて蓄積したものが最終的なフォーマットになります。

次の例では、XML データがテキストフィールドに割り当てられています。

```
<sectionHeading>This is a section</sectionHeading>
<mainBody>This is some main body text, with one
<emphasized>emphatic</emphasized> word.</mainBody>
```

このテキスト内の "emphatic" という単語に対しては、mainBody スタイルの中に emphasized スタイルがネストされています。mainBody スタイルでは、色、フォントサイズ、文字修飾の規則が定義されています。emphasized スタイルでは、これらの規則にフォントの太さの規則が追加されています。その結果、"emphatic" という単語には、mainBody と emphasized で指定されている規則を組み合わせてフォーマットが適用されます。

スタイルクラスの使用

スタイルの " クラス " を作成し、それを <p> タグや タグに適用することができます。スタイルクラスを適用するには、各タグの class 属性を使用します。これは厳密には、ActionScript 2.0 のクラスではありません。<p> タグに適用した場合、そのスタイルは段落全体に作用します。また、 タグを使用すれば、スタイルクラスを使ってテキストのある範囲にスタイルを適用できます。

たとえば、mainBody と emphasis という 2 つのスタイルクラスを定義した次のようなスタイルシートがあるとします。

```
.mainBody {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 24px;
}
.emphasis {
  color: #666666;
  font-style: italic;
}
```

テキストフィールドに割り当てる HTML テキストの中で、これらのスタイルを次の例のようにして <p> タグと タグに適用することができます。

```
<p class='mainBody'>This is <span class='emphasis'>really exciting!</span></p>
```

ビルトイン HTML タグのスタイル設定

Flash Player は HTML タグのサブセットをサポートしています。詳細については、382 ページの「HTML 形式のテキストの使用」を参照してください。テキストフィールド内に現れるビルトイン HTML タグのすべてのインスタンスに CSS スタイルを適用することができます。たとえば、次のコードでは、<p> というビルトイン HTML タグのスタイルを定義しています。このタグのすべてのインスタンスに、スタイル規則で指定したようにスタイルが適用されます。

```
p {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 12px;
  display: inline;
}
```

次の表に、ビルトイン HTML タグに設定できるスタイルと、スタイルの適用方法を示します。

スタイル名	スタイルの適用方法
p	すべての <p> タグに影響します。
body	すべての <body> タグに影響します。p スタイル (指定されている場合) は、body スタイルよりも優先されます。
li	すべての <li> 記号付きリストタグに影響します。
a	すべての <a> アンカータグに影響します。

スタイル名	スタイルの適用方法
a:link	すべての <a> アンカータグに影響します。このスタイルは a スタイルの後に適用されます。
a:hover	マウスポインタがリンク上を移動しているときに、そのリンクの <a> アンカータグに適用されます。このスタイルは a および a:link スタイルの後に適用されます。 マウスポインタがリンクから離れると、そのリンクから a:hover スタイルが削除されます。
a:active	ユーザーがリンクをクリックするときに、そのリンクの <a> アンカータグに適用されます。このスタイルは a および a:link スタイルの後に適用されます。 マウスボタンが離されると、そのリンクから a:active スタイルが削除されます。

HTML でのスタイルの使用例

このセクションでは、HTML タグでスタイルを使用する例を示します。まずスタイルシートを作成し、その中で一部のビルトインタグにスタイルを設定して、独自のスタイルクラスをいくつか定義します。その後、HTML 形式のテキストを含んでいる TextField オブジェクトにこのスタイルシートを適用します。

HTML をスタイルシートでフォーマットするには :

1. 任意のテキストエディタまたは CSS エディタで新規ファイルを作成します。
2. このファイルに次のスタイルシート定義を追加します。

```
p {
  color: #000000;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 12px;
  display: inline;
}

a:link {
  color: #FF0000;
}

a:hover{
  text-decoration: underline;
}

.headline {
  color: #000000;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 18px;
  font-weight: bold;
  display: block;
}
```

```
.byline {
  color: #666600;
  font-style: italic;
  font-weight: bold;
  display: inline;
}
```

このスタイルシートでは、<p> と <a> という 2 つのビルトイン HTML タグに対して、これらのタグのすべてのインスタンスに適用されるスタイルを定義しています。さらに、特定の段落とテキスト範囲に適用される .headline と .byline という 2 つのスタイルクラスを定義しています。

3. このファイルを html_styles.css として保存します。
4. テキストエディタまたは HTML エディタで新規テキストファイルを作成し、myText.htm という名前で保存します。

   ファイルに次のテキストを追加します。

```
<p class='headline'>Flash adds advanced anti-aliasing rendering technology!
  </p><p><span class='byline'>San Francisco, CA</span>--Adobe Inc. announced
  today a new version of Flash that features a brand new font rendering
  technology called Advanced Anti-Aliasing, most excellent at rendering small
  text with incredible clarity and consistency across platforms. For more
  information, visit the <a href='http://www.adobe.com'>Adobe Flash web
  site.</a></p>
```

メモ	このテキストストリングをコピー & ペーストする場合、テキストストリングに改行が追加されたときには必ず削除してください。

5. Flash オーサリングツールで新規 Flash ドキュメントを作成します。
6. タイムライン内で ([ウィンドウ]-[タイムライン])、レイヤー 1 の最初のフレームを選択します。
7. [アクション] パネルを開き ([ウィンドウ]-[アクション])、[アクション] パネルに次のコードを入力します。

```
this.createTextField("news_txt", 99, 50, 50, 450, 300);
news_txt.border = true;
news_txt.html = true;
news_txt.multiline = true;
news_txt.wordWrap = true;

// 新しいスタイルシートと LoadVars オブジェクトを作成する
var myVars_lv:LoadVars = new LoadVars();
var styles:TextField.StyleSheet = new TextField.StyleSheet();

// ロードする CSS ファイルとテキストファイルの場所
var txt_url:String = "myText.htm";
var css_url:String = "html_styles.css";

// onLoad ハンドラを定義し、CSS ファイルをロードする
styles.onLoad = function(success:Boolean):Void {
```

```
    if (success) {
        /* スタイルシートのロードが正常に完了した場合は、
            それをテキストオブジェクトに割り当て、
            その HTML テキストをテキストフィールドに割り当てる。*/
        news_txt.styleSheet = styles;
} else {
        trace("Unable to load CSS file.");
    }
};
styles.load(css_url);

// onData ハンドラを定義し、表示するテキストをロードする
myVars_lv.onData = function(src:String):Void {
    if (src != undefined) {
        news_txt.htmlText = src;
    } else {
        trace("Unable to load HTML file");
    }
};
myVars_lv.load(txt_url);
```

メモ	この ActionScript では、外部ファイルからテキストをロードしています。外部データをロードする方法については、第 14 章の「イメージ、サウンドおよびビデオの操作」を参照してください。

8. このファイルを、手順 3 で作成した CSS ファイルと同じディレクトリ内に news_html.fla という名前で保存します。
9. [制御]-[ムービープレビュー] を選択し、HTML テキストにスタイルが自動的に適用されることを確認します。

スタイルによる新規タグの定義

スタイルシート内で新しいスタイルを定義した場合、そのスタイルを、ビルトイン HTML タグとまったく同じようにタグとして使用することができます。たとえば、スタイルシート内で sectionHeading という名前の CSS スタイルを定義した場合は、そのスタイルシートに関連付けられたテキストフィールドの中で、<sectionHeading> をエレメントとして使用できます。この機能を利用すると、任意の XML 形式のテキストがスタイルシート内の規則に基づいて自動的にフォーマットされるように、XML 形式のテキストをテキストフィールドに直接割り当てることができます。

たとえば、次のスタイルシートでは、sectionHeading、mainBody、emphasized という新しいスタイルを作成しています。

```
sectionHeading {
  font-family: Verdana, Arial, Helvetica, sans-serif;
  font-size: 18px;
  display: block
}
```

```
mainBody {
   color: #000099;
   text-decoration: underline;
   font-size: 12px;
   display: block
}
emphasized {
   font-weight: bold;
   display: inline
}
```

このスタイルシートに関連付けられたテキストフィールドを次の XML 形式のテキストに埋め込むことができます。

```
<sectionHeading>This is a section</sectionHeading>
<mainBody>This is some main body text,
with one <emphasized>emphatic</emphasized> word.
</mainBody>
```

XML でのスタイルの使用例

このセクションでは、XML 形式のテキストが使われる FLA ファイルを作成します。377 ページの「HTML でのスタイルの使用例」で示したように CSS ファイルからスタイルを読み込むのではなく、ActionScript を使用してスタイルシートを作成します。

XML をスタイルシートでフォーマットするには :

1. Flash 内で新規 FLA ファイルを作成します。
2. テキストツールを使用して、幅 400 ピクセル、高さ 300 ピクセルのテキストフィールドを作成します。
3. プロパティインスペクタを開き ([ウィンドウ]-[プロパティ]-[プロパティ])、テキストフィールドを選択します。
4. プロパティインスペクタで、[テキストの種類] メニューから [ダイナミックテキスト] を選択し、[行タイプ] メニューから [複数行] を選択し、[テキストを HTML としてレンダリング] オプションを選択して、[インスタンス名] テキストボックスに「news_txt」と入力します。
5. タイムライン ([ウィンドウ]-[タイムライン]) のレイヤー 1 上で、最初のフレームを選択します。
6. スタイルシートオブジェクトを作成するために、[アクション] パネルを開き ([ウィンドウ]-[アクション])、[アクション] パネルに次のコードを入力します。

```
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.setStyle("mainBody", {
   color:'#000000',
   fontFamily:'Arial,Helvetica,sans-serif',
   fontSize:'12',
   display:'block'
});
```

```
styles.setStyle("title", {
   color:'#000000',
   fontFamily:'Arial,Helvetica,sans-serif',
   fontSize:'18',
   display:'block',
   fontWeight:'bold'
});
styles.setStyle("byline", {
   color:'#666600',
   fontWeight:'bold',
   fontStyle:'italic',
   display:'inline'
});
styles.setStyle("a:link", {
   color:'#FF0000'
});
styles.setStyle("a:hover", {
   textDecoration:'underline'
});
```

このコードで、setStyle() メソッドを使用してスタイルを定義する styles という名前の新しいスタイルシートオブジェクトを作成します。このスタイルは、この章で先に作成した外部 CSS ファイルと正確に一致します。

7. テキストフィールドに割り当てる XML テキストを作成するには、テキストエディタを開いて新規ドキュメントに次のテキストを入力します。

```
<story><title>Flash now has advanced anti-aliasing</
   title><mainBody><byline>San Francisco, CA</byline>--Adobe Inc. announced
   today a new version of Flash that features the new advanced anti-aliasing
   rendering technology. For more information, visit the <a href="http://
   www.adobe.com">Adobe Flash website</a></mainBody></story>
```

メモ	このテキストストリングをコピー & ペーストする場合、テキストストリングに改行が追加されたときには必ず削除してください。[アクション] パネルのポップアップメニューで [隠し文字] を選択し、余分な改行を表示して削除します。

8. テキストファイルを story.xml として保存します。

9. Flash で、手順 6 のコードに続けて次のコードを [アクション] パネルに追加ます。

このコードでは、story.xml ドキュメントをロードし、テキストフィールドの styleSheet プロパティにスタイルシートオブジェクトを割り当て、XML テキストをテキストフィールドに割り当てます。

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean):Void {
   if (success) {
      news_txt.styleSheet = styles;
      news_txt.text = my_xml;
```

```
    } else {
      trace("Error loading XML.");
    }
};
my_xml.load("story.xml");
```

メモ	この ActionScript では、外部ファイルから XML データをロードしています。外部データをロードする方法については、第 14 章の「イメージ、サウンドおよびビデオの操作」を参照してください。

10. ファイルを "story.xml" ファイルと同じフォルダ内に news_xml.fla として保存します。
11. SWF ファイルを実行し ([制御]-[ムービープレビュー])、フィールド内のテキストにスタイルが自動的に適用されることを確認します。

HTML 形式のテキストの使用

Flash Player では、<p> や <li> など、標準 HTML タグの一部をサポートし、ダイナミックテキストフィールドおよびテキスト入力フィールドのスタイル指定に使用できます。Flash Player 7 以降のテキストフィールドでは、<img> タグもサポートしています。このタグでは、テキストフィールドにイメージファイル (JPEG、GIF、PNG)、SWF ファイル、およびムービークリップを埋め込むことができます。Flash Player では、Web ブラウザで HTML ページの埋め込みイメージの周りをテキストが囲むのとまったく同じように、テキストフィールドに埋め込まれたイメージの周りをテキストが自動的に囲みます。詳細については、391 ページの「イメージ、SWF ファイル、およびムービークリップのテキストフィールドへの埋め込みについて」を参照してください。

また、Flash Player では、<textformat> タグをサポートしています。このタグでは、TextFormat クラスの段落書式設定スタイルを HTML が有効なテキストフィールドに適用できます。詳細については、367 ページの「TextFormat クラスの使用」を参照してください。

HTML 形式テキストの詳細については、次のトピックを参照してください。

- 383 ページの「HTML 形式のテキストの使用に必要なプロパティとシンタックス」
- 384 ページの「サポートされている HTML タグについて」
- 390 ページの「サポートされている HTML エンティティについて」
- 391 ページの「イメージ、SWF ファイル、およびムービークリップのテキストフィールドへの埋め込みについて」

HTML 形式のテキストの使用に必要なプロパティとシンタックス

HTML をテキストフィールドで使用するには、いくつかのテキストフィールドのプロパティを、プロパティインスペクタか ActionScript を使用して設定する必要があります。

- プロパティインスペクタで [テキストを HTML としてレンダリング] オプションを選択するか、テキストフィールドの `html` プロパティを `true` に設定して、テキストフィールドの HTML 形式設定を有効にします。
- `<p>`、`<br>`、および `<img>` などの HTML タグを使用するには、プロパティインスペクタで [複数行] オプションを選択するか、テキストフィールドの `multiline` プロパティを `true` に設定して、テキストフィールドを複数行テキストフィールドにする必要があります。
- ActionScript では、`TextField.htmlText` の値を、表示する HTML フォーマットのテキストストリングに設定します。

たとえば、次のコードでは、`headline_txt` という名前のテキストフィールドに対して HTML 形式設定を有効にしてから、いくつかの HTML をそのテキストフィールドに代入しています。

```
this.createTextField("headline_txt", 1, 10, 10, 500, 300);
headline_txt.html = true;
headline_txt.wordWrap = true;
headline_txt.multiline = true;
headline_txt.htmlText = "<font face='Times New Roman' size='25'>This is how you
  assign HTML text to a text field.</font><br>It's very useful.</br>";
```

HTML を正しくレンダリングするには、正しいシンタックスを使用する必要があります。HTML タグの属性は、一重引用符 (') または二重引用符 (") で囲む必要があります。引用符で囲んでいない属性値は、テキストの不適切なレンダリングなど、予期しない結果を生み出す可能性があります。たとえば、次の HTML コードは、`align` 属性に割り当てられている値 (`left`) が引用符で囲まれていないので、Flash Player で正しくレンダリングできません。

```
this.createTextField("myField_txt", 10, 10, 10, 400, 200);
myField_txt.html = true;
myField_txt.htmlText = "<p align=left>This is left-aligned text</p>";
```

属性値を二重引用符で囲む場合、引用符を " エスケープ " する (`\"`) 必要があります。なお、次のいずれかの方法をとることもできます。

```
myField_txt.htmlText = "<p align='left'>This uses single quotes</p>";
myField_txt.htmlText = "<p align=\"left\">This uses escaped double quotes</p>";
myField_txt.htmlText = '<p align="left">This uses outer single quotes</p>';
myField_txt.htmlText = '<p align=\'left\'>This uses escaped single quotes</p>';
```

外部ファイルからテキストをロードしている場合には、二重引用符をエスケープする必要はありません。ActionScript でテキストストリングを代入している場合にのみ二重引用符のエスケープが必要です。

サポートされている HTML タグについて

このセクションでは、Flash Player でサポートされているビルトイン HTML タグを一覧で示します。また、CCS を使用して新しいスタイルとタグを作成することもできます。詳細については、369 ページの「CSS スタイルによるテキストのフォーマット」を参照してください。

サポートされている HTML タグの詳細については、次のトピックを参照してください。

- 384 ページの「アンカータグ」
- 385 ページの「ボールドタグ」
- 385 ページの「ブレークタグ」
- 385 ページの「フォントタグ」
- 386 ページの「イメージタグ」
- 387 ページの「イタリックタグ」
- 387 ページの「リスト項目タグ」
- 387 ページの「段落タグ」
- 388 ページの「範囲タグ」
- 388 ページの「テキストフォーマットタグ」
- 389 ページの「下線タグ」

アンカータグ

<a> タグは、ハイパーリンクを作成し、次の属性をサポートします。

- `href` ブラウザにロードするページの URL を指定する最大 128 文字のストリングです。URL は、ページをロードしている SWF ファイルの場所に対して絶対的でも相対的でもかまいません。URL への絶対的な参照の例としては `http://www.adobe.com` が、相対的な参照の例としては `/index.html` が挙げられます。
- `target` ページをロードする先のターゲットウィンドウの名前を指定します。オプションには、`_self`、`_blank`、`_parent` および `_top` などがあります。`_self` オプションは現在のウィンドウで現在のフレームを指定します。`_blank` は新しいウィンドウを指定します。`_parent` は現在のフレームの親を指定します。`_top` は現在のウィンドウでトップレベルフレームを指定します。

たとえば、次の HTML コードの一部は、リンク "Go home" を作成し、これは、新しいブラウザウィンドウで www.adobe.com を開きます。

```
urlText_txt.htmlText = "<a href='http://www.adobe.com' target='_blank'>Go home</a>";
```

特殊な `asfunction` プロトコルを使用すると、リンクで URL を開くのではなく、SWF ファイルの ActionScript 関数を実行することができます。`asfunction` プロトコルの詳細については、『ActionScript 2.0 リファレンスガイド』の「asfunction プロトコル」を参照してください。

また、スタイルシートを使用すると、アンカータグに `a:link`、`a:hover`、および `a:active` の各スタイルを定義できます。376 ページの「ビルトイン HTML タグのスタイル設定」を参照してください。

メモ	絶対 URL には先頭に `http://` を付けます。それがない場合は、相対 URL と見なされます。

ボールドタグ

<b> タグはテキストをボールドとしてレンダリングします。例を次に示します。

```
text3_txt.htmlText = "He was <b>ready</b> to leave!";
```

ボールド体が、テキストを表示するために使用するフォントで利用できる必要があります。

ブレークタグ

 タグは、テキストフィールドに改行を作成します。このタグを使用するには、テキストフィールドを複数行テキストフィールドに設定する必要があります。

次の例では、文の間で改行しています。

```
this.createTextField("text1_txt", 1, 10, 10, 200, 100);
text1_txt.html = true;
text1_txt.multiline = true;
text1_txt.htmlText = "The boy put on his coat.<br />His coat was <font
  color='#FF0033'>red</font> plaid.";
```

フォントタグ

<font> タグは、テキストを表示するフォントまたはフォントリストを指定します。

フォントタグは、次の属性をサポートします。

- `color` 16 進数のカラー (#FFFFFF) 値のみがサポートされます。たとえば、次の HTML コードは赤いテキストを作成します。

  ```
  myText_txt.htmlText = "<font color='#FF0000'>This is red text</font>";
  ```

- `face` 使用するフォントの名前を指定します。次の例が示すように、カンマで区切られたフォント名の一覧を指定することもできます。その場合、Flash Player は最初に利用可能なフォントを選択します。

  ```
  myText_txt.htmlText = "<font face='Times, Times New Roman'>Displays as either
    Times or Times New Roman...</font>";
  ```

  指定したフォントがユーザーのコンピュータシステムにインストールされていないか、または SWF ファイルに組み込まれていない場合、Flash Player は代替フォントを選択します。

  Flash アプリケーションに対するフォント埋め込みの詳細については、『ActionScript 2.0 リファレンスガイド』の「embedFonts (TextField.embedFonts プロパティ)」、および『Flash ユーザーガイド』の「ダイナミックテキストおよびテキスト入力オプションの設定」を参照してください。

- `size` フォントのサイズをピクセル単位で指定します。次に例を示します。

  ```
  myText_txt.htmlText = "<font size='24' color='#0000FF'>This is blue, 24-point
    text</font>";
  ```

  ピクセル単位のサイズの代わりに、+2 または -4 などの相対ポイントサイズを使用することもできます。

イメージタグ

<img> タグでは、テキストフィールドと TextArea コンポーネントインスタンスの中に外部イメージファイル (JPEG、GIF、PNG)、SWF ファイル、およびムービークリップを埋め込むことができます。テキストは、テキストフィールドまたはコンポーネントに埋め込んだイメージの周りに自動的に流し込まれます。このタグを使用するには、ダイナミックテキストフィールドまたはテキスト入力フィールドを複数行および折り返しテキストに設定する必要があります。

単語の折り返しで複数行のテキストフィールドを作成するには、次のいずれか 1 つを行います。

- Flash オーサリング環境で、テキストフィールドをステージで選択してから、プロパティインスペクタで [テキストの種類] メニューから [複数行] を選択します。
- createTextField (MovieClip.createTextField メソッド) で実行時に作成されたテキストフィールドに対しては、新しいテキストフィールドインスタンスの multiline (TextField.multiline プロパティ) プロパティと multiline (TextField.multiline プロパティ) プロパティを true に設定します。

<img> タグには、src という必須属性が 1 つあり、ここでイメージファイル、SWF ファイル、またはライブラリ内のムービークリップシンボルのリンケージ識別子へのパスを指定します。他の属性はすべて省略可能です。

<img> タグは、次の属性をサポートします。

- **src** イメージファイルまたは SWF ファイルへの URL、またはライブラリ内のムービークリップシンボルへのリンケージ識別子を指定します。この属性は必須です。他の属性はすべて省略可能です。外部ファイル (JPEG、GIF、PNG および SWF ファイル) は、完全にダウンロードされるまで表示されません。
- **id** 埋め込みイメージファイル、SWF ファイル、またはムービークリップを含むムービークリップインスタンスの名前 (Flash Player が作成したもの) を指定します。これは、ActionScript で埋め込みコンテンツをコントロールする場合に便利です。
- **width** イメージ、SWF ファイル、またはムービークリップのピクセル単位の幅です。
- **height** 挿入されるイメージ、SWF ファイル、またはムービークリップのピクセル単位の高さです。
- **align** テキストフィールド内の埋め込みイメージの水平方向の整列を指定します。有効な値は、left および right です。デフォルト値は left です。
- **hspace** イメージの周りの水平方向の空白の量 (テキストが表示されない領域) を指定します。デフォルト値は 8 です。
- **vspace** イメージの周りの垂直方向の空白の量 (テキストが表示されない領域) を指定します。デフォルト値は 8 です。

<img> タグの使用の詳細と例については、391 ページの「イメージ、SWF ファイル、およびムービークリップのテキストフィールドへの埋め込みについて」を参照してください。

イタリックタグ

<i> タグは、タグ付きのテキストをイタリックで表示します。次にコード例を示します。

```
That is very <i>interesting</i>.
```

このコード例は次のようにレンダリングされます。

That is very *interesting*.

イタリック体が、使用されているフォントで利用できる必要があります。

リスト項目タグ

<li> タグは、タグが囲んでいるテキストの最初に箇条書き記号を配置します。次にコード例を示します。

```
Grocery list:
<li>Apples</li>
<li>Oranges</li>
<li>Lemons</li>
```

このコード例は次のようにレンダリングされます。

Grocery list:

- Apples
- Oranges
- Lemons

メモ

番号付きリストと番号なしリスト (<ol> タグと <ul> タグ) は Flash Player で認識されないため、これらのタグによって一覧のレンダリング方法が変更されることはありません。すべて一覧項目に、箇条書きの中黒記号が使用されます。

段落タグ

<p> タグは新しい段落を作成します。このタグを使用するには、テキストフィールドを複数行テキストフィールドに設定する必要があります。

<p> タグは、次の属性をサポートします。

- `align` パラグラフ内のテキストの整列を指定します。有効な値は、`left`、`right`、`justify`、および `center` です。
- `class` TextField.StyleSheet オブジェクトで定義される CSS スタイルクラスを指定します。詳細については、376 ページの「スタイルクラスの使用」を参照してください。

  次の例は、`align` 属性を使用して、テキストフィールドの右側にテキストを整列させています。

```
this.createTextField("myText_txt", 1, 10, 10, 400, 100);
myText_txt.html = true;
myText_txt.multiline = true;
myText_txt.htmlText = "<p align='right'>This text is aligned on the right
  side of the text field</p>";
```

次の例は、class 属性を使用して、<p> タグにテキストスタイルクラスを割り当てています。

```
var myStyleSheet:TextField.StyleSheet = new TextField.StyleSheet();
myStyleSheet.setStyle(".blue", {color:'#99CCFF', fontSize:18});
this.createTextField("test_txt", 10, 0, 0, 300, 100);
test_txt.html = true;
test_txt.styleSheet = myStyleSheet;
test_txt.htmlText = "<p class='blue'>This is some body-styled text.</p>.";
```

範囲タグ

 タグは、CSS テキストスタイルでのみ使用できます。詳細については、369 ページの「CSS スタイルによるテキストのフォーマット」を参照してください。次の属性をサポートします。

- **class**　TextField.StyleSheet オブジェクトで定義される CSS スタイルクラスを指定します。テキストスタイルクラスの作成方法については、376 ページの「スタイルクラスの使用」を参照してください。

テキストフォーマットタグ

<textformat> タグでは、行間、インデント、マージン、タブストップなどを含む、HTML テキストフィールド内にある TextFormat クラスの段落フォーマットプロパティの一部を使用できます。<textformat> タグは、ビルトイン HTML タグと組み合わせることができます。

<textformat> タグには次の属性があります。

- **blockindent**　ブロックのインデントをポイント単位で指定します。TextFormat.blockIndent に相当します。『ActionScript 2.0 リファレンスガイド』の「blockIndent (TextFormat.blockIndent プロパティ)」を参照してください。
- **indent**　左マージンから段落の最初の文字までのインデントを指定します。TextFormat.indent に相当します。負の値を指定可能です。『ActionScript 2.0 リファレンスガイド』の「indent (TextFormat.indent プロパティ)」を参照してください。
- **leading**　行間の垂直の行送りの量を指定します。TextFormat.leading に相当します。負の値を指定可能です。『ActionScript 2.0 リファレンスガイド』の「leading (TextFormat.leading プロパティ)」を参照してください。
- **leftmargin**　段落の左マージンをポイント単位で指定します。TextFormat.leftMargin に相当します。『ActionScript 2.0 リファレンスガイド』の「leftMargin (TextFormat.leftMargin プロパティ)」を参照してください。
- **rightmargin**　段落の右マージンをポイント単位で指定します。TextFormat.rightMargin に相当します。『ActionScript 2.0 リファレンスガイド』の「rightMargin (TextFormat.rightMargin プロパティ)」を参照してください。
- **tabstops**　カスタムタブストップを正の整数の配列で指定します。TextFormat.tabStops に相当します。『ActionScript 2.0 リファレンスガイド』の「tabStops (TextFormat.tabStops プロパティ)」を参照してください。

次に示されている、行ヘッダーがボールド体のデータの表は、その次の手順で示されているコード例を実行した結果です。

Name	Age	Occupation
Rick	33	Detective
AJ	34	Detective

タブストップを使用する、フォーマットされたデータの表を作成するには :

1. 新しい Flash ドキュメントを作成し、tabstops.fla という名前を付けて保存します。
2. タイムラインでは、レイヤー 1 の最初のフレームを選択します。
3. [アクション] パネルを開き ([ウィンドウ]-[アクション])、[アクション] パネルに次のコードを入力します。

```
// 新しいテキストフィールドを作成する
this.createTextField("table_txt", 99, 50, 50, 450, 100);
table_txt.multiline = true;
table_txt.html = true;
// ボールドでフォーマットされ、タブで区切られた列ヘッダーを作成する
var rowHeaders:String = "<b>Name\tAge\tOccupation</b>";

// データを持つ行を作成する
var row_1:String = "Rick\t33\tDetective";
var row_2:String = "AJ\t34\tDetective";

// 2 つの tabstops を 50 ポイントと 100 ポイントに設定する
table_txt.htmlText = "<textformat tabstops='[50,100]'>";
table_txt.htmlText += rowHeaders;
table_txt.htmlText += row_1;
table_txt.htmlText += row_2 ;
table_txt.htmlText += "</textformat>";
```

   タブ文字用エスケープシーケンス (\t) を使用して、表の各列の間にタブを挿入しています。+= 演算子を使用してテキストを追加します。

4. [制御]-[ムービープレビュー] を選択してフォーマットされたテーブルを確認します。

下線タグ

<u> タグはタグ付きテキストに下線を引きます。コード例を次に示します。

```
This is <u>underlined</u> text.
```

このコードは次のようにレンダリングされます。

This is <u>underlined</u> text.

サポートされている HTML エンティティについて

HTML エンティティを使用すると、特定の文字を HTML として解釈せず、HTML フォーマットのテキストフィールドにそのまま表示することができます。たとえば、" より小さい " (<) 文字と " より大きい " (>) 文字は <img> や などの HTML タグを囲むのに使用されます。それで、Flash で HTML フォーマットのテキストフィールドに " より小さい " 文字や " より大きい " 文字を表示するには、これらの文字の HTML エンティティで代用する必要があります。次の ActionScript では、ステージ上に HTML フォーマットのテキストフィールドを作成し、HTML エンティティを使用して、テキストをボールドにせずにストリング "<b>" を表示します。

```
this.createTextField("my_txt", 10, 100, 100, 100, 19);
my_txt.autoSize = "left";
my_txt.html = true;
my_txt.htmlText = "The &lt;b&gt; tag makes text appear <b>bold</b>.";
```

Flash で上記のコード例を実行すると、ステージに次のようなテキストが表示されます。

The <b> tag makes text appear **bold**.

" より大きい " 文字および " より小さい " 文字に加えて、Flash では次の表に一覧で示されている HTML エンティティが認識されます。

エンティティ	説明
<	< (より小さい)
>	> (より大きい)
&	& (アンパサンド)
"	" (二重引用符)
'	' (アポストロフィ、一重引用符)

Flash ではまた、' (ASCII のアンパサンド) や & (Unicode のアンパサンド) など、文字コードの明示的な指定もサポートしています。

次の ActionScript は、ASCII 文字コードや Unicode 文字コードを使用してチルダ (~) 文字を埋め込む方法を示しています。

```
this.createTextField("my_txt", 10, 100, 100, 100, 19);
my_txt.autoSize = "left";
my_txt.html = true;
my_txt.htmlText = "&#126;"; // チルダ (ASCII)
my_txt.htmlText += "\t"
my_txt.htmlText += "&#x007E;"; // チルダ (Unicode)
```

イメージ、SWF ファイル、およびムービークリップのテキストフィールドへの埋め込みについて

Flash Player 7 以降では、<img> タグを使って、イメージファイル (JPEG、GIF、PNG)、SWF ファイル、およびムービークリップをダイナミックテキストフィールドおよびテキスト入力フィールド、および TextArea コンポーネントインスタンスの内部に埋め込むことができます。<img> タグで使用する全属性の一覧については、386 ページの「イメージタグ」を参照してください。

Flash はテキストフィールドに埋め込まれたメディアをフルサイズで表示します。埋め込むメディアのサイズを指定するには、<img> タグの height 属性と width 属性を使用します。詳細については、393 ページの「高さと幅の値の指定について」を参照してください。

一般に、テキストフィールドに埋め込まれたイメージは、<img> タグの次の行に表示されます。ただし、<img> タグがテキストフィールドの最初の文字であった場合、イメージはテキストの最初の行に表示されます。

SWF ファイルとイメージファイルの埋め込み

テキストフィールドにイメージファイルまたは SWF ファイルを埋め込むには、<img> タグの src 属性にイメージファイル (GIF、JPEG、PNG) または SWF ファイルへの絶対パスまたは相対パスを指定します。たとえば、次の例は、SWF ファイルと同じディレクトリにある GIF ファイルを挿入します (相対アドレス、オンラインまたはオフライン)。

テキストフィールドにイメージを埋め込むには :

1. 新しい Flash ドキュメントを作成し、embedding.fla という名前を付けて保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createTextField("image1_txt", 10, 50, 50, 450, 150);
image1_txt.html = true;
image1_txt.htmlText = "<p>Here's a picture from my vacation:<img
  src='beach.gif'>";
```

   上記のコードでは、ステージ上に新しいダイナミックテキストフィールドを作成し、HTML 形式設定を有効にし、そしてテキストフィールドにテキストとローカルイメージを追加します。

3. 前の手順で追加したコードの下に、次の ActionScript を追加します。

```
this.createTextField("image2_txt", 20, 50, 200, 400, 150);
image2_txt.html = true;
image2_txt.htmlText = "<p>Here's a picture from my garden:<img src='http://
  www.helpexamples.com/flash/images/image2.jpg'>";
```

   絶対アドレスを使用してイメージを挿入することもできます。上記のコードは、サーバー上のディレクトリにある JPEG ファイルを挿入します。このコードが入った SWF ファイルは、ハードディスク上に置くこともサーバーに置くこともできます。

4. ドキュメントを保存し、[制御]-[ムービープレビュー] を選択してテストします。

   上側のテキストフィールドにはテキストによる文が表示され、多くの場合、[出力] パネルには Flash で "beach.gif" というファイルが現在のディレクトリに見つからなかったというエラーメッセージが表示されます。下側のテキストフィールドには、テキストによる文と、リモートサーバーからロードされた花のイメージが表示されます。

   GIF イメージを同じディレクトリに FLA としてコピーして名前を beach.gif に変更し、[制御]-[ムービープレビュー] を選択して Flash ドキュメントを再度テストします。

メモ	絶対 URL を使用する場合、その URL の先頭は必ず `http://` にする必要があります。

ムービークリップシンボルの埋め込み

テキストフィールドにムービークリップシンボルを埋め込むには、`<img>` タグの `src` 属性にシンボルのリンケージ識別子を指定します。リンケージ識別子の定義の詳細については、311 ページの「ステージへのムービークリップシンボルの割り当て」を参照してください。

たとえば、次のコードは、インスタンス名 `textField_txt` を持つダイナミックテキストフィールドに、リンケージ識別子 `symbol_ID` でムービークリップシンボルを挿入しています。

ムービークリップをテキストフィールドに埋め込むには :

1. 新しい Flash ドキュメントを作成し、embeddedmc.fla という名前を付けて保存します。
2. ステージ上に新しいシェイプを描画するか、[ファイル]-[読み込み]-[ステージに読み込み] を選択してだいたい幅と高さが 100 ピクセルのイメージを選択します。
3. シェイプまたは前の手順で読み込まれたイメージをステージ上で選択し、それから F8 キーを押して [シンボルに変換] ダイアログボックスを開いて変換します。
4. ムービークリップにビヘイビアを設定し、わかりやすいシンボル名を付けます。基準点グリッドの左上の四角形を選択し、[詳細] をクリックして詳細モードに切り替えます (詳細モードになっていない場合)。
5. [ActionScript に書き出し] チェックボックスおよび [最初のフレームに書き出し] チェックボックスを選択します。
6. [識別子] テキストボックスにリンケージ識別子を「img_id」と入力し、[OK] をクリックします。
7. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

   ```
   this.createTextField("textField_txt", 10, 0, 0, 300, 200);
   textField_txt.html = true;
   textField_txt.htmlText = "<p>Here's a movie clip symbol:<img src='img_id'>";
   ```

   埋め込みムービークリップを適切かつ完全に表示するには、そのシンボルの基準点が点 (0,0) にある必要があります。
8. Flash ドキュメントの変更を保存します。
9. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

高さと幅の値の指定について

<image> タグに width と height の属性を指定した場合、そのイメージファイル、SWF ファイル、またはムービークリップ用に領域がテキストフィールド内に予約されます。イメージファイルまたは SWF ファイルは、完全にダウンロードされると、予約された領域に表示されます。Flash は、指定された height 値と width 値に基づいてメディアを拡大 / 縮小します。イメージのサイズを決めるには、height 属性と width 属性の両方の値を入力する必要があります。

height 値と width 値を指定しない場合、埋め込みメディア用に予約される領域はありません。イメージファイルまたは SWF ファイルが完全にダウンロードされたら、Flash がテキストフィールドにフルサイズでメディアを埋め込み、その周りのテキストを区切り直します。

メモ	テキストを含むテキストフィールドにイメージを動的にロードする場合は、イメージに予約した領域の周囲にテキストが適切に配置されるように、元のイメージの幅と高さを指定することをお勧めします。

ActionScript での埋め込みメディアの制御

Flash は各 <image> タグに対して新しいムービークリップを作成し、そのムービークリップを TextField オブジェクト内に埋め込みます。<image> タグの id 属性では、作成されるムービークリップにインスタンス名を割り当てることができます。これによって ActionScript でそのムービークリップを制御することができます。

Flash で作成されるムービークリップは、イメージを含むテキストフィールドの子ムービークリップとして追加されます。

たとえば、次の例ではテキストフィールドに SWF ファイルを埋め込みます。

テキストフィールドに SWF ファイルを埋め込むには :

1. 新しい Flash ドキュメントを作成します。
2. ドキュメントのステージサイズを 100 x 100 ピクセルにサイズ変更します。
3. 矩形ツールを使用して、ステージに赤い四角形を描画します。
4. プロパティインスペクタを使用して四角形を 80 x 80 ピクセルにサイズ変更し、そのシェイプをステージの中央に移動します。
5. タイムラインのフレーム 20 を選択して F7 キー (Windows または Macintosh) を押し、新しい空白キーフレームを挿入します。
6. 楕円ツールを使用して、フレーム 20 のステージに青い円を描画します。
7. プロパティインスペクタを使用して円を 80 x 80 ピクセルにサイズ変更し、それをステージの中央に移動します。
8. フレーム 1 と フレーム 20 の間の空白フレームをクリックし、プロパティインスペクタでトゥイーンタイプを [シェイプ] に設定します。

9. 現在のドキュメントを animation.fla という名前で保存します。

10. [制御]-[ムービープレビュー] を選択してアニメーションをプレビューします。

    SWF ファイルが FLA と同じディレクトリに作成されます。この実習が正しく機能するには、別の FLA ファイルにロードできるよう SWF ファイルを生成する必要があります。

11. 新しい FLA ファイルを作成し、"animationholder.fla" という名前で保存します。

    ファイルは、前に作成した "animation.fla" ファイルと同じフォルダに保存してください。

12. 次の ActionScript コードをメインタイムラインのフレーム 1 に追加します。

```
this.createTextField("textField_txt", 10, 0, 0, 300, 200);
textField_txt.html = true;
textField_txt.htmlText = "Here's an interesting animation: <img
  src='animation.swf' id='animation_mc'>";
```

    この場合、新しく作成されたムービークリップへの完全修飾パスは

    `textField_txt.animation_mc` です。

13. Flash ドキュメントの変更を保存し、[制御]-[ムービープレビュー] を選択して、テキストフィールド内でアニメーションをプレビューします。

テキストフィールド内で再生されている SWF ファイルを制御する方法については、次の演習で扱われます。

テキストフィールド内で再生される SWF ファイルを制御するには :

1. 393 ページの「ActionScript での埋め込みメディアの制御」の最初に説明されている手順を実行します。

2. ステージにボタンインスタンスを作成し、プロパティインスペクタで stop_btn というインスタンス名を付けます。

3. メインタイムラインのフレーム 1 に記述されている既存のコードに続けて、次の ActionScript コードを追加します。

```
stop_btn.onRelease = function() {
  textField_txt.animation_mc.stop();
};
```

4. [制御]-[ムービープレビュー] を選択してアプリケーションをテストします。

    `stop_btn` ボタンインスタンスをクリックするといつでも、テキストフィールド内にネストされているアニメーションのタイムラインが停止するようになりました。

埋め込みメディアをハイパーリンクに設定する方法の詳細については、395 ページの「埋め込みメディアからのハイパーテキストリンクの作成について」を参照してください。

埋め込みメディアからのハイパーテキストリンクの作成について

埋め込まれた イメージファイル、SWF ファイル、またはムービークリップからハイパーリンクを作成するには、<img> タグを <a> タグ内に入れます。

```
textField_txt.htmlText = "Click the image to return home<a href='home.htm'><img
  src='home.jpg'></a>";
```

<a> タグで囲まれたイメージ、SWF ファイルまたはムービークリップ上にマウスが移動すると、標準的なハイパーリンク上の場合と同様に、マウスポインタが " 指差し " のアイコンに変わります。マウスのクリックやキーを押す動作などのインタラクティブ性は、<a> タグで囲まれた SWF ファイルやムービークリップには登録されません。

埋め込みメディアの詳細については、395 ページの「埋め込みメディアからのハイパーテキストリンクの作成について」を参照してください。

例 : スクロールテキストの作成

Flash でスクロールテキストを作成する方法はいくつかあります。[テキスト] メニューまたはコンテキストメニューの [スクロール可能] オプションを選択するか、または Shift キーを押しながらテキストブロックハンドルをダブルクリックすると、ダイナミックテキストフィールドまたはテキスト入力フィールドをスクロール可能にすることができます。

テキストフィールド内の垂直方向のスクロールを制御するには TextField オブジェクトの `scroll` プロパティおよび `maxscroll` プロパティを使用し、水平方向のスクロールを制御するには `hscroll` プロパティおよび `maxhscroll` プロパティを使用します。`scroll` プロパティと `hscroll` プロパティは、それぞれ現在の縦方向と横方向のスクロール位置を指定します。これらのプロパティに対しては読み取りと書き込みを行うことができます。`maxscroll` プロパティと `maxhscroll` プロパティは、それぞれ縦方向と横方向の最大のスクロール位置を指定します。これらのプロパティに対しては読み取りのみができます。

TextArea コンポーネントを使用すると、最小限のスクリプト記述でテキストフィールドのスクロールを簡単に作成することができます。詳細については、『ActionScript 2.0 リファレンスガイド』の「TextArea コンポーネント」を参照してください。

スクロール可能なダイナミックテキストフィールドを作成するには :

次のいずれかの操作を行います。

- ダイナミックテキストフィールドのハンドルを Shift キーを押しながらダブルクリックします。
- 選択ツールでダイナミックテキストフィールドを選択し、[テキスト]-[スクロール可能] を選択します。
- 選択ツールでダイナミックテキストフィールドを選択します。ダイナミックテキストフィールドを右クリック (Windows) または Control キーを押しながらクリック (Macintosh) し、[テキスト]-[スクロール可能] を選択します。

スクロールプロパティを使用してスクロールテキストを作成するには :

1. 次のいずれかの操作を行います。
   - テキストツールを使用して、ステージ上にテキストフィールドをドラッグします。プロパティインスペクタで、テキストフィールドにインスタンス名 textField_txt を割り当てます。
   - ActionScript を使って、`MovieClip.createTextField()` メソッドでダイナミックにテキストフィールドを作成します。テキストフィールドには、メソッドのパラメータとしてインスタンス名 textField_txt を割り当てます。

メモ	テキストを動的に SWF ファイルにロードしない場合は、メインメニューから [テキスト]-[スクロール可能] を選択します。

2. 上下のボタンを作成するか、[ウィンドウ]-[サンプルライブラリ]-[ボタン] を選択し、ステージにボタンをドラッグします。

   これらのボタンを使用して、テキストを上下にスクロールできます。

3. ステージで [下] ボタンを選択し、[インスタンス名] テキストボックスに「down_btn」と入力します。
4. ステージで [上] ボタンを選択し、[インスタンス名] テキストボックスに「up_btn」と入力します。
5. テキストフィールドでテキストを下にスクロールするには、タイムラインのフレーム 1 を選択し、[アクション] パネル ([ウィンドウ]-[アクション]) で、次のコードを入力します。

```
down_btn.onPress = function() {
   textField_txt.scroll += 1;
};
```

6. テキストを上にスクロールするには、ActionScript の手順 5 に従って、次のコードを入力します。

```
up_btn.onPress = function() {
   textField_txt.scroll -= 1;
};
```

   `textField_txt` テキストフィールドにロードするすべてのテキストが、上下のボタンを使用してスクロールできるようになります。

ストリングおよび String クラスについて

プログラミングにおいては、ストリングは順序付けられた一連の文字です。アプリケーションでテキストフィールド内などのテキストを表示するために、Flash ドキュメントおよびクラスファイルではストリングを頻繁に使用します。また、値をストリングとして格納し、アプリケーションの中でさまざまな目的のために使用することもできます。データの文字を引用符で囲むことによって、ActionScript コードに直接ストリングを設定できます。ストリングの作成の詳細については、404 ページの「ストリングの作成」を参照してください。テキストフィールドの使用の詳細については、334 ページの「TextField クラスの使用」を参照してください。

各文字と指定した文字コードを関連付けることができ、それをオプションでテキストの表示に使用することも可能です。たとえば、文字 "A" は、Unicode 文字コードでは 0041、ASCII (American Standard Code for Information Interchange) では 65 で表されます。文字コードとコードチャートの詳細については、www.unicode.org/charts/ を参照してください。このように、Flash ドキュメントでストリングを表す方法は、選択する文字セット、および文字をエンコードする方法によって大きく左右されます。

" 文字エンコーディング " とは、言語中の文字のセットを、数値などの代理コードで表すためのコードまたはメソッドを指しています。" 文字コード " は、前の段落で説明したように、マップされる値の表です。たとえば、ASCII 表では A は 65 になります。そしてエンコーディングメソッドにより、コンピュータプログラムの中でそれが解読されます。

たとえば、英語の各文字には、それぞれ文字エンコーディングで使用される代理数値コードが割り当てられています。ASCII では、文字、数字、記号がそれぞれ 7 ビットバイナリの整数にエンコードされます。ASCII は 95 個の印刷可能文字と多数の制御文字で構成される文字セットであり、コンピュータにおいてテキストを表すのに使用されます。

ASCII と同様、*Unicode* もコードとアルファベットの各文字を関連付ける 1 つの方法です。ASCII は中国語のような大きな文字セットをサポートできないため、Unicode 規格は言語をエンコーディングする重要な標準になります。Unicode は、あらゆる言語セットを表現できる文字セットの規格であり、複数の言語での開発を支援するために作られた標準です。Unicode の文字コードはそれが何を表すのかを指定し、標準として、あらゆる言語を構成する文字をエンコードする汎用的な方法を提供することを目指しています。ストリングは使用されるあらゆるコンピュータシステム、プラットフォーム、またはソフトウェア上で表示できます。その場合、文字グリフ (その視覚的外観) の表示は、Flash や Web ブラウザなど、使用するプログラムに任せられます。

Unicode がサポートする文字の数は年々増加し、より多くの (より大きな) 言語がサポートされるようになっています。文字エンコーディングは Unicode Transformation Format (UTF) および Universal Character Set (UCS) と呼ばれており、UTF-8、UTF-16、UTF-32 があります。UTF エンコーディングの数は 1 単位のビット数を示しており、UCS エンコーディングの数はバイト数を示しています。

- UTF-8 は、オンラインメールシステムなどのテキスト交換用の標準エンコーディングです。UTF は 8 ビットシステムです。
- UTF-16 は一般に内部処理に使用されます。

アプリケーションでは、さまざまな長さのストリングが使用されます。ストリングの長さは任意に決められますが、使用している言語に左右される場合があります。また、ストリングの最後に終了文字が置かれることがあります。この null 文字には値はありません。この終了文字は実際の文字ではありませんが、ストリングの最後を判定するために使用できます。たとえば、ソケット接続を操作している場合、(チャットプログラムなどで) 終了文字を監視してストリングの最後を知ることができます。

選択されたストリングとサブストリングを比較および取得する簡単なワードプロセッサを作成する方法を示す、サンプルのソースファイル "strings.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動してサンプルにアクセスします。

ストリングおよび String クラスの詳細については、次のトピックを参照してください。

- 398 ページの「パネル」
- 399 ページの「Locale クラスの使用」
- 401 ページの「IME の使用」
- 403 ページの「String クラスについて」
- 404 ページの「ストリングの作成」
- 405 ページの「エスケープ文字について」
- 406 ページの「ストリング内の文字の分析と比較」
- 409 ページの「ストリングの変換と連結」
- 411 ページの「返されるサブストリング」

パネル

[ストリング] パネルを使用すると、複数言語コンテンツを作成および更新できます。複数言語を含むテキストフィールドのコンテンツを指定することや、Flash Player が実行されるコンピュータの言語に基づいて、特定の言語で表示されるコンテンツを Flash に自動的に判定させることができます。

[ストリング] パネルに関する一般的な情報、およびアプリケーションでのその使用法については、『Flash ユーザーガイド』の次のトピックを参照してください。

- 多言語テキストのオーサリング
- "[ストリング] パネルについて "
- "[ストリング] パネルまたは XML ファイルにおけるテキストの翻訳 "
- "[ストリング] パネルへの XML ファイルの読み込み "

Locale クラスを使用すると、多言語テキストの表示方法を制御できます。詳細については、『ActionScript 2.0 リファレンスガイド』の 399 ページの「Locale クラスの使用」および「`Locale (mx.lang.Locale)`」を参照してください。

Locale クラスの使用

Locale クラス (mx.lang.Locale) クラスを使用すると、Flash アプリケーションで実行時に表示される多言語テキストを制御できます。[ストリング] パネルを使用することで、ダイナミックテキストフィールドにストリングリテラルではなくストリング ID を使用することができ、これによって言語別の XML ファイルからロードされたテキストを表示する SWF ファイルを作成することが可能となります。XLIFF (XML Localization Interchange File Format) ファイルに含まれている言語別の文字列を表示するには、次のような方法があります。

実行時に自動で　Flash Player によってストリング ID が、`language` (`capabilities.language` プロパティ) から返されるデフォルトのシステム言語と一致する XML ファイル内のストリングに置き換えられます。

ステージ言語を使用して手動で　コンパイル時にストリング ID がストリングに置き換えられるので、Flash Player では変更できません。

実行時に ActionScript を使用して　ストリング ID の置き換えを ActionScript を使用して実行時に制御します。このオプションでは、ストリング ID を置き換えるタイミングと言語を両方とも制御できます。

アプリケーションが Flash Player で再生される際にアプリケーションを ActionScript で制御してストリング ID を置き換える場合は、Locale クラスのプロパティとメソッドを使用できます。Locale の使用法の例については、次の手順を参照してください。

Locale クラスを使用して多言語サイトを作成するには :

1. 新しい Flash ドキュメントを作成し、locale.fla という名前で保存します。
2. [ストリング] パネル ([ウィンドウ]-[その他のパネル]-[ストリング]) を開き、[設定] をクリックします。
3. [en (英語)] および [fr (フランス語)] の 2 つの言語を選択し、[追加] をクリックして [アクティブな言語] ペインにこの 2 つの言語を追加します。
4. [実行時に ActionScript を使用して] オプションを選択し、実行時のデフォルト言語をフランス語に設定して [OK] をクリックします。
5. ComboBox コンポーネントを [コンポーネント] パネル ([ウィンドウ]-[コンポーネント]) の "User Interface" フォルダからステージまでドラッグし、インスタンス名を lang_cb とします。
6. テキストツールを使用してステージ上にダイナミックテキストフィールドを作成し、そのテキストフィールドのインスタンス名を greeting_txt とします。
7. ステージでそのテキストフィールドを選択し、[ストリング] パネルの [ID] テキストボックスにストリング識別子を「greeting」と入力します。それから [適用] をクリックしてください。

   Flash により、入力した greeting というストリングが IDS_GREETING へと変換されます。
8. [ストリング] パネルのグリッドで、[en] 列に「hello」というストリングを入力します。
9. [fr] 列に「bonjour」というストリングを入力します。

   これらのストリングは、`lang_cb` コンボボックスを使ってステージ上の言語を変更するときに使用します。

10. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
import mx.lang.Locale;
Locale.setLoadCallback(localeListener);
lang_cb.dataProvider = Locale.languageCodeArray.sort();
lang_cb.addEventListener("change", langListener);
greeting_txt.autoSize = "left";
Locale.loadLanguageXML(lang_cb.value);

function langListener(eventObj:Object):Void {
  Locale.loadLanguageXML(eventObj.target.value);
}
function localeListener(success:Boolean):Void {
  if (success) {
    greeting_txt.text = Locale.loadString("IDS_GREETING");
  } else {
    greeting_txt.text = "unable to load language XML file.";
  }
}
```

上記の ActionScript は 2 つのセクションに分けることができます。コードの最初のセクションでは、Locale クラスを読み込み、言語別の XML ファイルの読み込みが完了したときに呼び出されるコールバックリスナーを指定しています。次に、使用可能な言語のソート済みの配列が `lang_cb` コンボボックスに設定されます。`lang_cb` の値が変更されると、Flash のイベントディスパッチャーによって、言語別の XML ファイルをロードする `langListener()` 関数がトリガされます。コードの 2 番目のセクションでは、`langListener()` および `localeListener()` という 2 つの関数が定義されています。最初の関数、`langListener()` は、`lang_cb` コンボボックスの値がユーザーによって変更されたときに呼び出されます。2 番目の関数、`localeListener()` は、言語別の XML ファイルの読み込みが完了したときに呼び出されます。この関数では読み込みに成功したかどうかがチェックされ、成功していれば、`greeting_txt` インスタンスの `text` プロパティを選択された言語の挨拶の文に設定します。

11. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

ヒント	使用する XML ファイルでは、XLIFF (XML Localization Interchange File Format) を使用する必要があります。

注意	Locale クラスは、『ActionScript 2.0 リファレンスガイド』で説明されている他のクラスとは異なり、Flash Player には含まれていません。このクラスは Flash Authoring クラスパスにインストールされているため、自動的に SWF ファイルにコンパイルされます。Locale クラスは SWF ファイルにコンパイルする必要があるため、このクラスを使うことによって SWF ファイルのサイズはわずかに大きくなります。

詳細については、『ActionScript 2.0 リファレンスガイド』の「`Locale (mx.lang.Locale)`」を参照してください。

IME の使用

IME (Input Method Editor : 入力方式エディタ) を使用すると、中国語、日本語、韓国語といったアジアの言語で ASCII 以外の文字を入力できるようになります。ActionScript の IME クラスを使用すると、クライアントコンピュータ上で実行されている Flash Player アプリケーションでオペレーティングシステムの IME を直接操作できます。

ActionScript を使用して、次の点の判別が可能です。

- ユーザーのコンピュータに IME がインストールされているかどうか
- ユーザーのコンピュータで IME が有効になっているかどうか
- 現在 IME で使用されている変換モード

IME クラスでは、現在の IME でどの変換モードが使われているかを判別できます。たとえば日本語 IME がアクティブになっている場合、`System.IME.getConversionMode()` メソッドを使用することで、変換モードがひらがな、カタカナ (または他の変換モード) のいずれであるかを判別できます。また、`System.IME.setConversionMode()` メソッドを使えば変換モードの設定が可能です。

メモ	今のところ、どの IME がアクティブになっているかの判別 (IME がインストールされている場合) や、IME の種類の変更 (たとえば英語から日本語へ、韓国語から中国語へなど) はできません。

ユーザーのオペレーティングシステムによっては、実行時にアプリケーションで IME の有効 / 無効を切り替えたり、他の機能を実行したりすることもできます。`System.capabilities.hasIME` プロパティを使用すれば、システムに IME があるかどうかをチェックできます。次の例では、IME がインストールされているか、そしてアクティブかを判別する方法を示します。

IME がインストールされているか、そしてアクティブかを判別するには :

1. 新しい Flash ドキュメントを作成し、ime.fla という名前で保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
if (System.capabilities.hasIME) {
  if (System.IME.getEnabled()) {
    trace("You have an IME installed and enabled.");
  } else {
    trace("You have an IME installed but not enabled.");
  }
} else {
  trace("Please install an IME and try again.");
}
```

   上記のコードではまず、現在のシステムに IME がインストールされているかどうかをチェックします。IME がインストールされていれば、現在有効になっているかどうかが Flash によってチェックされます。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   [出力] パネルに、IME がインストールされているかどうか、そして現在アクティブかどうかを示すメッセージが表示されます。

また、IME クラスを使用することで、実行時に Flash で IME の有効 / 無効を切り替えることもできます。次の例を試してみるには、システムに IME がインストールされている必要があります。特定のプラットフォームに IME をインストールする方法の詳細については、次のリンクを参照してください。

- www.microsoft.com/globaldev/default.mspx
- http://developer.apple.com/documentation/
- http://java.sun.com

次の例に示されているように、SWF ファイルの再生時に IME の有効 / 無効を切り替えることができます。

IME の有効 / 無効を実行時に切り替えるには :

1. 新しい Flash ドキュメントを作成し、ime2.fla という名前を付けて保存します。
2. ステージ上にボタンシンボルインスタンスを 2 つ作成し、インスタンス名をそれぞれ enable_btn および disable_btn とします。
3. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
checkIME();

var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "_sans";

this.createTextField("ime_txt", 10, 100, 10, 320, 240);
ime_txt.border = true;
ime_txt.multiline = true;
ime_txt.setNewTextFormat(my_fmt);
ime_txt.type = "input";
ime_txt.wordWrap = true;

enable_btn.onRelease = function() {
  System.IME.setEnabled(true);
};
disable_btn.onRelease = function() {
  System.IME.setEnabled(false);
};

function checkIME():Boolean {
  if (System.capabilities.hasIME) {
    if (System.IME.getEnabled()) {
      trace("You have an IME installed and enabled.");
      return true;
    } else {
      trace("You have an IME installed but not enabled.");
      return false;
    }
  } else {
    trace("Please install an IME and try again.");
    return false;
  }
}
```

このコードは、5 つのセクションに分かれています。最初のセクションでは checkIME() メソッドを呼び出しています。これにより、システムに IME がインストールされていて、アクティブであれば [出力] パネルにメッセージが表示されます。2 番目のセクションではカスタムテキストフォーマットオブジェクトを定義しており、フォントを _sans に設定しています。3 番目のセクションではテキスト入力フィールドを作成し、カスタムテキストフォーマットを適用しています。4 番目のセクションでは、前の手順で作成した enable_btn インスタンスおよび disable_btn インスタンスのイベントハンドラをいくつか作成しています。5 番目の、コードの最後のセクションではカスタム関数 checkIME() を定義しており、この関数では、現在のシステムに IME がインストールされているか、されているならアクティブかどうかをチェックします。

4. FLA ファイルを保存し、[制御]-[ムービープレビュー] を選択してテストします。

メモ	この例を実行するには、システムに IME がインストールされている必要があります。IME のインストールの詳細については、この例の前に示されているリンクを参照してください。

ステージ上のテキスト入力フィールドにテキストを入力します。IME を別の言語に切り替え、テキスト入力フィールドにもう一度テキストを入力します。Flash Player で、新しい IME を使用して文字が入力されます。ステージ上の disable_btn ボタンをクリックすると、Flash は前の言語を使用する設定に戻り、現在の IME 設定は無視されます。

System.capabilities.hasIME の詳細については、『ActionScript 2.0 リファレンスガイド』の「hasIME (capabilities.hasIME プロパティ)」を参照してください。

String クラスについて

ストリングは、コア ActionScript 言語の 1 つのクラスおよびデータ型でもあります。String データ型は、文字、数字、区切り記号などを含む 16 ビット文字の連続を表します。ストリングは、UTF-16 形式を使用して、Unicode 文字として格納されます。String 値を操作すると、ストリングの新しいインスタンスが返されます。String データ型で宣言される変数のデフォルト値は null です。

ストリング、データ、値の詳細については、第 3 章の「データおよびデータ型」を参照してください。

String クラスには、テキストストリングを操作するためのメソッドが含まれています。ストリングは多くのオブジェクトを操作する上で重要であり、この章で説明するメソッドは、TextField、XML、ContextMenu、FileReference インスタンスなど、多くのオブジェクトで使用されるストリングを操作するために役立ちます。

String クラスは、ストリングプリミティブデータ型のラッパーです。このメソッドとプロパティを使用して、プリミティブストリング値を操作できます。String() 関数を使用して任意のオブジェクトの値をストリングに変換できます。concat()、fromCharCode()、slice()、substr() を除く String クラスのすべてのメソッドは汎用メソッドです。つまり、メソッドが toString() 関数を呼び出した後で、メソッドの操作が実行されます。これらのメソッドは String オブジェクト以外のオブジェクトでも使用できます。

すべてのストリングインデックスはゼロから始まるので、任意のストリング `myStr` の最終文字のインデックスは `myStr.length - 1` になります。

選択されたストリングとサブストリングを比較および取得する簡単なワードプロセッサを作成する方法を示す、サンプルのソースファイル "strings.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動してサンプルにアクセスします。

ストリングの作成

String クラスのメソッドを呼び出すには、コンストラクタメソッド `new String()` を使用するか、ストリングリテラル値を使用します。ストリングリテラルを指定すると、ActionScript インタプリタはそれをテンポラリ String オブジェクトに自動変換し、そしてテンポラリ String オブジェクトを破棄します。ストリングリテラルで `String.length` プロパティを使用することもできます。

ストリングリテラルと String オブジェクトを混同しないように注意してください。ストリングリテラルと String オブジェクトの詳細については、93 ページ、第 4 章の「リテラルについて」を参照してください。

次の例のコード行は、ストリングリテラル `firstStr` を作成しています。ストリングリテラルを宣言するには、直線の一重引用符 (') または直線の二重引用符 (") 区切り記号を使用します。

ストリングを作成および使用するには :

1. 新しい Flash ドキュメントを作成し、strings.fla という名前を付けて保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var firstStr:String = "foo";
var secondStr:String = new String("foo");
trace(firstStr == secondStr); // true
var thirdStr:String;
trace(thirdStr); // undefined
```

   このコードでは 3 つの String オブジェクトを定義しています。1 つはストリングリテラルを使用するオブジェクト、1 つは `new` 演算子を使用するオブジェクト、もう 1 つは初期値がないオブジェクトです。ストリングは、コードの 3 行目に示されているように 等価 (==) 演算子を使って比較できます。変数を参照する場合、データ型の指定は変数を定義するときのみ行います。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

String オブジェクトを特に使用する必要がない限り、ストリングリテラルを使用するようにしてください。ストリングリテラルと String オブジェクトの詳細については、93 ページ、第 4 章の「リテラルについて」を参照してください。

1 つのストリングリテラル内で直線の一重引用符 (') および直線の二重引用符 (") 区切り記号を使用するときは、円記号 (\) を使用してその文字をエスケープしてください。次の 2 つのストリングは等価です。

```
var firstStr:String = "That's \"fine\"";
var secondStr:String = 'That\'s "fine"';
```

ストリング内の円記号の使用の詳細については、405 ページの「エスケープ文字について」を参照してください。

" 曲線引用符 " や " 特殊引用符 " を ActionScript コードで使用することはできません。これらは、コードで使用可能な直線の引用符 (' および ") とは異なります。Web や Word ドキュメントなど、別のソースから ActionScript にテキストをペーストするときは、必ず直線の引用符を使用してください。

選択されたストリングとサブストリングを比較および取得する簡単なワードプロセッサを作成する方法を示す、サンプルのソースファイル "strings.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動してサンプルにアクセスします。

エスケープ文字について

エスケープ文字である円記号 (\) を使用して、ストリングリテラル内で他の文字を定義できます。

エスケープシーケンス	説明
`\b`	円記号。
`\f`	改ページ文字。
`\n`	改行文字。
`\r`	復帰文字。
`\t`	タブ文字。
`\u`*nnnn*	16 進数 *nnnn* で指定される文字コードを持つ Unicode 文字。たとえば、`\u263a` はスマイル文字です。
`\x`*nn*	16 進数 *nn* で指定される文字コードを持つ ASCII 文字 。
`\'`	一重引用符。
`\"`	二重引用符。
`\\`	単一の円記号。

ストリングリテラルの詳細については、93 ページ、第 4 章の「リテラルについて」および 404 ページの「ストリングの作成」を参照してください。

ストリング内の文字の分析と比較

ストリング内のすべての文字が、ストリング内のインデックス位置を持っています (整数)。最初の文字のインデックス位置は 0 です。たとえば、ストリング yellow 内の文字 y の位置は 0、文字 w の位置は 5 になります。

すべてのストリングには length プロパティがあります。これは、ストリング内の文字数と同じ値になります。

```
var companyStr:String = "adobe";
trace(companyStr.length); // 10
```

空のストリングと null ストリングはどちらも length がゼロです。

```
var firstStr:String = new String();
trace(firstStr.length); // 0

var secondStr:String = "";
trace(secondStr.length); // 0
```

ストリングに値が含まれていない場合、length は undefined に設定されます。

```
var thirdStr:String;
trace(thirdStr.length); // undefined
```

参考	ストリングに null バイト文字 (\0) が含まれている場合、ストリングの値はそこで切り詰められます。

文字コードを使用してストリングを定義することもできます。文字コードおよび文字エンコーディングの詳細については、397 ページの「ストリングおよび String クラスについて」を参照してください。

次の例では、myStr という名前の変数を作成し、String.fromCharCode() メソッドに渡された ASCII 値に基づいてストリングの値を設定します。

```
var myStr:String =
  String.fromCharCode(104,101,108,108,111,32,119,111,114,108,100,33);
trace(myStr); // hello world!
```

上記のコードの fromCharCode() メソッドにリストされている各数字はそれぞれ 1 つの文字を表しています。たとえば、ASCII 値 104 は小文字の h を、ASCII 値 32 は空白文字を表します。

また、Unicode 値は 16 進数値から 10 進数値に変換する必要がありますが、次の ActionScript に示されているように、String.fromCharCode() メソッドを使用して変換することもできます。

```
// Unicode 0068 == "h"
var letter:Number = Number(new Number(0x0068).toString(10));
trace(String.fromCharCode(letter)); // h
```

次の例のように、ストリング内のさまざまな位置にある文字を調べることができます。

ストリングをループ処理するには :

1. 新しい Flash ドキュメントを作成します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var myStr:String = "hello world!";
for (var i:Number = 0; i < myStr.length; i++) {
  trace(myStr.charAt(i));
}
```

3. [制御]-[ムービープレビュー] を選択し、Flash ドキュメントをプレビューします。出力パネルに各文字が、別々の行に出力されます。
4. 既存のこの ActionScript スクリプトコードを、各文字の ASCII 値を出力するように修正します。

```
var myStr:String = "hello world!";
for (var i:Number = 0; i < myStr.length; i++) {
  trace(myStr.charAt(i) + " - ASCII=" + myStr.charCodeAt(i));
}
```

5. 現在の Flash ドキュメントを保存し、[制御]-[ムービープレビュー]を選択して SWF ファイルをプレビューします。

   このコードを実行すると、次のように [出力] パネルに表示されます。

```
h - ASCII=104
e - ASCII=101
l - ASCII=108
l - ASCII=108
o - ASCII=111
  - ASCII=32
w - ASCII=119
o - ASCII=111
r - ASCII=114
l - ASCII=108
d - ASCII=100
! - ASCII=33
```

ヒント

また、String.split() メソッドを使用し、区切り文字として空のストリング（`""`）を入力することで、ストリングを文字の配列に分けることもできます (例 :`var charArray:Array = myStr.split("");`)。

演算子を使用してストリングを比較できます。ストリングでの演算子の使用の詳細については、140 ページの「演算子とストリングの使用について」を参照してください。

条件ステートメントでは、`if` や `while` などの演算子を使用できます。次の例では、演算子とストリングを使用して比較を行います。

2 つのストリングを比較するには :

1. 新しい Flash ドキュメントを作成し、comparestr.fla という名前を付けて保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var str1:String = "Apple";
var str2:String = "apple";
if (str1 < str2) {
  trace("Uppercase letters sort first.");
}
```

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

次の例に示されているように、等価演算子 (==) と不等価演算子 (!=) を使用して、ストリングと他のタイプのオブジェクトを比較できます。

ストリングを他のデータ型と比較するには :

1. 新しい Flash ドキュメントを作成し、comparenum.fla という名前を付けて保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var myStr:String = "4";
var total:Number = 4;
if (myStr == total) {
  trace("Types are converted.");
}
```

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

   ストリングと数字など、異なる 2 つのデータ型を比較する場合、Flash では比較できるようデータ型の変換を試みます。

厳密な等価演算子 (===) と厳密な不等価演算子 (!==) を使用して、両方の比較オブジェクトが同じタイプかどうかを検査できます。次の例では、厳密な比較演算子を使用して、値の比較の際に Flash によってデータ型の変換が試みられていないことを確認します。

厳密な型指定による比較を行うには :

1. 新しい Flash ドキュメントを作成し、comparestrict.fla という名前を付けて保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var str1:String = "4";
var str2:String = "5";
var total:Number = 4;
if (str1 !== total) {
  trace("Types are not converted.");
}
if (str1 !== str2) {
  trace("Same type, but the strings don't match.");
}
```

3. Flash ドキュメントを保存して、[制御]-[ムービープレビュー] を選択します。

ストリングへの演算子の使用の詳細については、140 ページの「演算子とストリングの使用について」を参照してください。

選択されたストリングとサブストリングを比較および取得する簡単なワードプロセッサを作成する方法を示す、サンプルのソースファイル "strings.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動してサンプルにアクセスします。

ストリングの変換と連結

`toString()` メソッドを使用すると、多くのオブジェクトをストリングに変換できます。ほとんどのビルトインオブジェクトはこの変換を行う `toString()` メソッドを持っています。

```
var n:Number = 0.470;
trace(typeof(n.toString())); // string
```

加算演算子 (+) をストリングおよびストリング以外のインスタンスの組み合わせに使用するときは、`toString()` メソッドを使用する必要はありません。連結の詳細については、このセクションの 2 番目の手順を参照してください。

`toLowerCase()` メソッドおよび `toUpperCase()` メソッドは、ストリング内のアルファベット文字をそれぞれ小文字と大文字に変換します。次の例は、ストリングを小文字から大文字に変換する方法を示しています。

ストリングを小文字から大文字に変換するには :

1. 新しい Flash ドキュメントを作成し、convert.fla という名前を付けて保存します。
2. タイムラインのフレーム 1 に次のコードを入力します。

```
var myStr:String = "Dr. Bob Roberts, #9.";
trace(myStr.toLowerCase()); // dr. bob roberts, #9.
trace(myStr.toUpperCase()); // DR. BOB ROBERTS, #9.
trace(myStr); // Dr. Bob Roberts, #9.
```

3. Flash ドキュメントを保存して、[制御]-[ムービープレビュー] を選択します。

メモ

これらのメソッドを実行しても、ソースストリングは変更されません。ソースストリングを変換するには、次の方法を使用します。

```
myStr = myStr.toUpperCase();
```

ストリングを連結すると、2 つのストリングが順番に結合されて 1 つのストリングになります。たとえば、加算演算子 (+) を使用して 2 つのストリングを連結できます。次の例では、2 つのストリングを連結する方法を示します。

2 つのストリングを連結するには :

1. 新しい Flash ドキュメントを作成し、concat.fla という名前を付けて保存します。
2. タイムラインのフレーム 1 に次のコードを追加します。

```
var str1:String = "green";
var str2:String = str1 + "ish";
trace(str2); // greenish
//
var str3:String = "yellow";
str3 += "ish";
trace(str3); // yellowish
```

   上記のコードでは、文字列を連結する 2 種類の方法が示されています。最初の方法では、加算演算子 (+) を使用してストリング str1 とストリング "ish" を結合させています。2 番目の方法では、加算後代入 (+=) 演算子を使用して、ストリング "ish" を str3 の現在の値と連結しています。
3. Flash ドキュメントを保存して、[制御]-[ムービープレビュー] を選択します。

String クラスの concat() メソッドを使用してストリングを連結することもできます。次の例は、このメソッドを示しています。

2 つのストリングを concat() メソッドで連結するには :

1. 新しい Flash ドキュメントを作成し、concat2.fla という名前を付けて保存します。
2. タイムラインのフレーム 1 に次のコードを追加します。

```
var str1:String = "Bonjour";
var str2:String = "from";
var str3:String = "Paris";
var str4:String = str1.concat(" ", str2, " ", str3);
trace(str4); // Bonjour from Paris
```

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

加算演算子 (+) (または加算後代入演算子 (+=)) をストリングとストリング以外のオブジェクトに対して使用すると、ActionScript は、式を評価するためにストリング以外のオブジェクトを自動的にストリングに変換します。この変換は、次のコードの例に示されています。

```
var version:String = "Flash Player ";
var rel:Number = 9;
version = version + rel;
trace(version); // Flash Player 9
```

ただし、次の ActionScript コードに示すように、括弧を使用して加算 (+) 演算子に算術的な評価を強制することができます。

```
trace("Total: $" + 4.55 + 1.46); // 合計 : $4.551.46
trace("Total: $" + (4.55 + 1.46)); // 合計 : $6.01
```

split() メソッドを使用して、ストリングのサブストリングの配列を作成することができます。この配列は、区切り文字で分割されます。たとえば、カンマまたはタブで区切られたストリングを複数のストリングに分割できます。

たとえば、次のコードは、アンパサンド文字 (&) を区切り文字として使用し、配列をサブストリングに分割できることを示しています。

区切り文字で分割されたサブストリングの配列を作成するには :

1. 新しい Flash ドキュメントを作成し、strsplit.fla という名前を付けて保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var queryStr:String = "first=joe&last=cheng&title=manager&startDate=3/6/65";
var params:Array = queryStr.split("&", 2);
trace(params); // first=joe,last=cheng
     /* params は 2 つのエレメントを持つ配列に設定
        params[0] == "first=joe"
        params[1] == "last=cheng"
     */
```

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

ヒント

split() メソッドの 2 番目のパラメータは、配列の最大サイズを定義しています。split() メソッドによって作成される配列のサイズを制限しないようにするには、2 番目のパラメータを省略します。

ヒント

クエリーストリング (& および = 文字で区切られているストリング) を解析する最も簡単な方法は、LoadVars.decode() メソッドを使用することです。その例を次の ActionScript に示します。

```
var queryStr:String = "first=joe&last=cheng&title=manager&startDate=3/6/65";
var my_lv:LoadVars = new LoadVars();
my_lv.decode(queryStr);
trace(my_lv.first); // joe
```

ストリングへの演算子の使用の詳細については、140 ページの「演算子とストリングの使用について」を参照してください。

選択されたストリングとサブストリングを比較および取得する簡単なワードプロセッサを作成する方法を示す、サンプルのソースファイル "strings.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動してサンプルにアクセスします。

返されるサブストリング

String クラスの substr() メソッドと substring() メソッドは類似したメソッドです。どちらもストリングのサブストリングを返しますし、どちらもパラメータは 2 つです。どちらのメソッドでも、最初のパラメータは、与えられたストリングの開始文字の位置です。ただし、substr() メソッドの 2 番目のパラメータは返されるサブストリングの長さであり、substring() メソッドの 2 番目のパラメータはサブストリングの末尾の文字の位置です (これは返されるストリングには含まれません)。この例は 2 つのメソッドの相違を示しています。

文字の位置によってサブストリングを見つけるには :

1. 新しい Flash ドキュメントを作成し、substring.fla という名前を付けて保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var myStr:String = "Hello from Paris, Texas!!!";
trace(myStr.substr(11,15)); // Paris, Texas!!!
trace(myStr.substring(11,15)); // Pari
```

   最初のメソッド、substr() は、11 番目の文字から 15 文字を返します。2 番目のメソッド、substring() は、11 番目と 15 番目のインデックス間のすべての文字を取り出し、4 文字のストリングを返します。

3. 前の手順で作成したコードの下に、次の ActionScript を追加します。

```
trace(myStr.slice(11, 15)); // Pari
trace(myStr.slice(-3, -1)); // !!
trace(myStr.slice(-3, 26)); // !!!
trace(myStr.slice(-3, myStr.length)); // !!!
trace(myStr.slice(-8, -3)); // Texas
```

   slice() メソッドの機能は substring() メソッドと類似しています。2 つの負でない整数をパラメータとして与えた場合、両方のメソッドは完全に同じ働きをします。ただし、slice() メソッドの場合は負の整数をパラメータとして使用できます。

4. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

メモ	slice() メソッドのパラメータには、負でない整数と負の整数を組み合わせて指定できます。

indexOf() メソッドおよび lastIndexOf() メソッドを使用して、次の例のように、ストリング内で一致するサブストリングを見つけることができます。

一致するサブストリングの文字位置を検索するには :

1. 新しい Flash ドキュメントを作成し、indexof.fla という名前を付けて保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var myStr:String = "The moon, the stars, the sea, the land";
trace(myStr.indexOf("the")); // 10
trace(myStr.indexOf("the", 11)); // 21
```

   indexOf() メソッドでは大文字小文字を区別するため、単語 "the" の最初のインデックスは 10 番目の文字で始まります。そのため、"The" の最初のインスタンスは考慮されません。indexOf() メソッドに 2 番目のパラメータを指定して、検索を開始するストリング内のインデックス位置を指示することもできます。上記のコードでは、Flash は 11 番目の文字以降に出現する単語 "the" の最初のインデックスを検索します。

3. 前の手順で追加したコードに続けて次の ActionScript を追加します。

```
trace(myStr.lastIndexOf("the")); // 30
trace(myStr.lastIndexOf("the", 29)); // 21
```

`lastIndexOf()` メソッドは、ストリング内の最後のサブストリングを検索します。つまり、`lastIndexOf()` は文字またはサブストリングをストリングの最初から検索するのではなく、ストリングの最後から始めて逆方向へと検索します。`indexOf()` メソッドと同じように、`lastIndexOf()` メソッドに 2 番目のパラメータを指定すると、インデックス位置から検索が行われます。ただし、`lastIndexOf()` の場合は逆方向 (右から左) に検索が行われる点が異なります。

4. [制御]-[ムービープレビュー] を選択し、Flash ドキュメントをテストします。

ヒント	`indexOf()` および `lastIndexOf()` メソッドでは大文字小文字を区別します。

選択されたストリングとサブストリングを比較および取得する簡単なワードプロセッサを作成する方法を示す、サンプルのソースファイル "strings.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Strings" フォルダに移動してサンプルにアクセスします。

第 12 章

アニメーション、フィルタおよび描画

12

この章では、ActionScript を使用して、Flash CS3 Professional アプリケーションにアニメーションを追加する方法を説明します。モーションやシェイプトゥイーンを使用してタイムラインベースのアニメーションを追加する方法にはほとんど触れません。コードを使用してアニメーションや効果を作成すると、完成したアプリケーションのファイルサイズを縮小し、アニメーションそのもののパフォーマンスおよび整合性を向上させることができます。また、ActionScript を使用してアニメーションを作成する方が、作業効率が高い場合があります。たとえば、コードの方が記述しやすい場合があります。また、多くのインスタンスに一度に適用することや、他のアプリケーションで再利用することが簡単になります。この章では、他に、ActionScript の基本的要素、Tween クラスと TransitionManager クラス、Drawing API、フィルタクラス、ブレンドモードを使用したアニメーション処理について説明します。

Drawing API を使用して、アニメーションを追加し、描画することができます。Drawing API は、MovieClip クラスの描画メソッドで構成されます。これらのメソッドを使用することによって、オーサリングツールで描画ツールを使用するのではなく、コードを使用して、線、塗り、シェイプを作成できます。

フィルタやその他の表現効果も、効果をすばやく適用し、アニメーション化するために、多くの Flash アプリケーションにおいて重要です。コードを使用して、フィルタ効果、ブレンドモード、およびビットマップイメージを追加し、アニメーション化できます。

この章は、次のセクションで構成されます。ActionScript を使用してアニメーションを作成し、効果を追加する方法と Drawing API を使用して ActionScript で描画を作成する方法を説明します。

ActionScript 2.0 のスクリプトによるアニメーション処理 416
ビットマップのキャッシュ、スクロール、およびパフォーマンスについて 426
Tween クラスと TransitionManager クラスについて 427
フィルタ効果の使用 442
ActionScript によるフィルタの操作 449
コードによるフィルタ効果の操作. 471
BitmapData クラスによるビットマップの作成 475
ブレンドモードについて 478
操作順序について 481
ActionScript を使用した描画. 481
拡大・縮小とスライスガイドの理解 497

ActionScript 2.0 のスクリプトによるアニメーション処理

タイムラインでモーションやシェイプトゥイーンを使用するのではなく、ActionScript 2.0 を使用して、Flash アプリケーションにアニメーションを追加できます。次のセクションでは、コードを使用してインスタンスの透明度や外観を変更し、ステージでインスタンスを動かすなど、コードによるインスタンスのアニメーション処理について説明します。

Tween クラスと TransitionManager クラスを使用して、コードベースのアニメーション処理をさらに自動化する方法の詳細については、『ActionScript 2.0 リファレンスガイド』を参照してください。この 2 つのクラスは、高度なイージング式とトランジションアニメーションをアプリケーションのムービークリップインスタンスに追加する場合に役立ちます。これらの作成済みクラスを使用せずに、ActionScript で効果を再作成するのは容易ではありません。効果を実現するためには、複雑な数式をコードに記述しなければならないからです。

コードを使用して作成した描画をアニメーション化する方法の詳細については、481 ページの「ActionScript を使用した描画」を参照してください。

次のセクションでは、アニメーションをスクリプトで作成する方法を説明します。

- 417 ページの「アニメーションとフレームレートについて」
- 418 ページの「コードによるオブジェクトのフェード」
- 420 ページの「コードによるカラー効果と明度効果の追加」
- 423 ページの「コードによるオブジェクトの移動」
- 424 ページの「コードによるイメージのパン」

スクリプトで作成された Flash のアニメーションのサンプルソースファイル "animation.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Animation" フォルダに移動してサンプルにアクセスします。

フォトギャラリアプリケーションのサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Galleries" フォルダに移動して次のサンプルにアクセスします。

- gallery_tree.fla
- gallery_tween.fla

これらのファイルには、スクリプトアニメーションが組み込まれている SWF ファイルにイメージファイルをダウンロードする間に、ActionScript を使用してムービークリップを動的に制御する方法の例が含まれています。

アニメーションとフレームレートについて

アプリケーションにアニメーションを追加する場合、FLA ファイルに設定するフレームレートを検討します。フレームレートは、アニメーションを再生する SWF ファイルとコンピュータのパフォーマンスに影響を与えるので、アニメーションを使用する場合には、フレームレートについて考える必要があります。フレームレートを高く設定すると、特に多くのアセットを使用する場合や ActionScript を使用してアニメーションを作成する場合に、CPU の負荷が問題になります。

アニメーションをスムーズに再生するためには、フレームレートを適切に設定する必要があります。たとえば、プロパティインスペクタで 12 fps (frames per second : 1 秒あたりのフレーム数) に設定されたアニメーションは、1 秒間に 12 のフレームを再生します。このドキュメントのフレームレートを 24 fps に設定すると、12 fps に設定したときよりも、アニメーションはスムーズに再生されます。同時に、24 fps のアニメーションは、12 fps のアニメーションよりも再生速度が速いので、再生時間 (秒単位) が短くなります。つまり、高いフレームレートの 5 秒間のアニメーションを作成する必要があると、低いフレームレートのアニメーションよりも多くのフレームを作成する必要があり、その結果、アニメーションのファイルサイズが大きくなります。通常、24 fps の 5 秒のアニメーションは、12 fps の 5 秒のアニメーションよりも、ファイルサイズが大きくなります。

メモ	`onEnterFrame` イベントハンドラを使用してスクリプトアニメーションを作成する場合、作成したアニメーションは、ドキュメントのフレームレートで再生されます。タイムラインでモーショントゥイーンを作成する場合と同様です。`onEnterFrame` イベントハンドラを使用せずに、`setInterval` (『ActionScript 2.0 リファレンスガイド』の「setInterval 関数」を参照) を使用することもできます。フレームレートを使用するのではなく、指定した間隔で関数を呼び出します。`onEnterFrame` の場合と同様に、`setInterval` を頻繁に使用して関数を呼び出すと、アニメーションが CPU に負荷をかけます。

実行時にアニメーションがスムーズに再生される最小限のフレームレートを設定してください。そうすることによって、エンドユーザーのコンピュータの CPU 負荷を抑えることができます。また、30 ～ 40 fps を超えるフレームレートを使用しないでください。フレームレートを高くしても、CPU の負荷が増すだけで、実行時のアニメーションの再生状態はほとんど変わりません。

特に、タイムラインベースアニメーションを作成する場合は、開発のできるだけ早い段階で、アニメーションのフレームレートを設定します。SWF ファイルをテストし、アニメーションの再生時間と SWF ファイルのサイズを確認します。フレームレートは、アニメーションの再生速度に大きな影響を及ぼします。

コードによるオブジェクトのフェード

ステージ上のムービークリップに対して、_visible プロパティを切り替えるのではなく、フェードインまたはフェードアウトによって表示と非表示を切り替えたい場合があります。次の手順は、onEnterFrame イベントハンドラを使用して、ムービークリップをアニメーション化する方法を示します。

コードを使用してムービークリップをフェードするには :

1. 新しい Flash ドキュメントを fade1.fla という名前で作成します。
2. 描画ツールを使用してステージ上にいくつかのグラフィックを描画するか、ステージにイメージを読み込みます ([ファイル]-[読み込み]-[ステージに読み込み])。
3. ステージ上のコンテンツを選択し、[修正]-[シンボルに変換] を選択します。
4. [ムービークリップ] オプションを選択し、[OK] をクリックしてシンボルを作成します。
5. ステージ上でムービークリップインスタンスを選択し、プロパティインスペクタ内の [インスタンス名] テキストボックスに「img1_mc」と入力します。
6. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
img1_mc.onEnterFrame = function() {
  img1_mc._alpha -= 5;
  if (img1_mc._alpha <= 0) {
    img1_mc._visible = false;
    delete img1_mc.onEnterFrame;
  }
};
```

   このコードは、onEnterFrame イベントハンドラを使用しています。このイベントハンドラは、SWF ファイルに設定されたフレームレートで繰り返し呼び出されます。1 秒間にイベントハンドラが呼び出される回数は、Flash ドキュメントに設定されているフレームレートに依存します。フレームレートが 12 fps の場合、onEnterFrame イベントハンドラは 1 秒間に 12 回呼び出されます。同様に、Flash ドキュメントのフレームレートが 30 fps の場合、このイベントハンドラは 1 秒間に 30 回呼び出されます。

7. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   ステージに追加したムービークリップが徐々にフェードアウトします。

次の手順に示すように、onEnterFrame イベントハンドラの代わりに setInterval( ) 関数を使用して、_alpha プロパティを変更できます。

setInterval() 関数を使用してオブジェクトをフェードするには :

1. 新しい Flash ドキュメントを fade2.fla という名前で作成します。
2. ステージ上にいくつかのグラフィックを描画するか、ステージにイメージを読み込みます ([ファイル]-[読み込み]-[ステージに読み込み])。
3. ステージ上のコンテンツを選択し、[修正]-[シンボルに変換] を選択します。
4. [ムービークリップ] オプションを選択し、[OK] をクリックしてシンボルを作成します。
5. ステージ上でムービークリップインスタンスを選択し、プロパティインスペクタ内の [インスタンス名] テキストボックスに「img1_mc」と入力します。
6. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
var alpha_interval:Number = setInterval(fadeImage, 50, img1_mc);
function fadeImage(target_mc:MovieClip):Void {
  target_mc._alpha -= 5;
  if (target_mc._alpha <= 0) {
    target_mc._visible = false;
    clearInterval(alpha_interval);
  }
}
```

   `setInterval()` 関数の動作は、`onEnterFrame` イベントハンドラとは少し異なります。`setInterval()` 関数は、特定の関数が Flash から呼び出される頻度を指定します。このコードでは、ユーザー定義関数 `fadeImage()` が 50 ミリ秒ごとに (1 秒間に 20 回) 呼び出されます。`fadeImage()` 関数は、現在のムービークリップの `_alpha` プロパティの値をデクリメントします。`_alpha` の値が 0 以下になったら、呼び出し間隔をクリアし、`fadeImage()` 関数は実行を停止します。

7. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   ステージに追加したムービークリップが徐々にフェードアウトします。

ユーザー定義の関数の詳細については、166 ページの「グローバル関数とタイムライン関数の定義」を参照してください。`onEnterFrame` イベントハンドラの詳細については、『ActionScript 2.0 リファレンスガイド』の「`onEnterFrame (MovieClip.onEnterFrame` ハンドラ)」を参照してください。`setInterval()` 関数の詳細については、『ActionScript 2.0 リファレンスガイド』の「`setInterval` グローバル関数」を参照してください。

スクリプトで作成された Flash のアニメーションのサンプルソースファイル "animation.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Animation" フォルダに移動してサンプルにアクセスします。

コードによるカラー効果と明度効果の追加

ActionScript を使用してアルファフェードを設定し、アニメーション化するだけでなく (418 ページの「コードによるオブジェクトのフェード」を参照)、プロパティインスペクタの [フィルタ] パネルを使用せずに、コードを使用して、さまざまなカラー効果や明度効果をアニメーションに追加できます。

次の手順では、JPEG イメージをロードしてカラー変換フィルタを適用し、x 軸と y 軸に沿ってマウスポインタを移動すると、赤と緑のチャンネルが変化するようにします。

ActionScript を使用してオブジェクトのカラーチャンネルを変更するには :

1. 新しい Flash ドキュメントを colorTrans.fla という名前で作成します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
import flash.geom.Transform;
import flash.geom.ColorTransform;

var imageClip:MovieClip = this.createEmptyMovieClip("imageClip", 1);
var clipLoader:MovieClipLoader = new MovieClipLoader();
clipLoader.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
   imageClip);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  var transformer:Transform = new Transform(imageClip);
  var colorTransformer:ColorTransform = transformer.colorTransform;
  colorTransformer.redMultiplier = (_xmouse / Stage.width) * 1;
  colorTransformer.greenMultiplier = (_ymouse / Stage.height) * 1;
  transformer.colorTransform = colorTransformer;
}
Mouse.addListener(mouseListener);
```

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストし、ステージ上でマウスポインタを動かします。

   マウスの動きに応じて、ロードしたイメージファイルのカラーが変化します。

次の手順に示すように、ColorMatrixFilter クラスを使用して、カラーイメージをモノクロのイメージに変換することもできます。

ColorMatrixFilter クラスを使用して、イメージをグレースケールイメージに変更するには :

1. 新しい Flash ドキュメントを grayscale.fla という名前で作成します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
import flash.filters.ColorMatrixFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip):Void {
   var myElements_array:Array = [0.3, 0.59, 0.11, 0, 0,
        0.3, 0.59, 0.11, 0, 0,
        0.3, 0.59, 0.11, 0, 0,
        0, 0, 0, 1, 0];
   var myColorMatrix_filter:ColorMatrixFilter = new
   ColorMatrixFilter(myElements_array);
   target_mc.filters = [myColorMatrix_filter];
}
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
   img_mc);
```

   このコードは、まず、ColorMatrixFilter クラスを読み込み、後のコードで作成する MovieClipLoader インスタンスと一緒に使用されるリスナーオブジェクトを作成しています。次に、ムービークリップインスタンス img_mc とムービークリップのローダーインスタンス img_mcl を新規に作成します。最後に、ソースムービークリップが、ステージの img_mc ムービークリップにロードされます。イメージがロードが完了すると、onLoadInit イベントハンドラが呼び出され、ロードしたイメージに ColorMatrixFilter をアタッチします。
3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   ステージにロードしたイメージがグレースケールイメージに変わります。オンラインでイメージを表示して (http://www.helpexamples.com/flash/images/image1.jpg)、イメージの元のカラーを確認します。

次の手順のように、ActionScript コードを使用して、イメージの明度を設定できます。

イメージの明度を変更するには :

1. 新しい Flash ドキュメントを brightness.fla という名前で作成します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
import flash.filters.ColorMatrixFilter;
System.security.allowDomain("http://www.helpexamples.com/");
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip):Void {
   var myElements_array:Array = [1, 0, 0, 0, 100,
        0, 1, 0, 0, 100,
        0, 0, 1, 0, 100,
        0, 0, 0, 1, 0];
   var myColorMatrix_filter:ColorMatrixFilter = new
   ColorMatrixFilter(myElements_array);
   target_mc.filters = [myColorMatrix_filter];
}
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image2.jpg",
   img_mc);
```

   このコードブロックは、MovieClipLoader クラスを使用して、外部の JPEG をロードします。イメージが適切にロードされると、MovieClipLoader クラスの onLoadInit イベントハンドラが呼び出され、ColorMatrixFilter フィルタを使用して、イメージの明度を 100 に変更します。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   SWF ファイルをテストすると、SWF ファイルにロードしたイメージの明度が変わります。オンラインでイメージを表示して (http://www.helpexamples.com/flash/images/image2.jpg)、イメージの元の外観を確認します。

スクリプトで作成された Flash のアニメーションのサンプルソースファイル "animation.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Animation" フォルダに移動してサンプルにアクセスします。

フォトギャラリアプリケーションのサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Galleries" フォルダに移動して次のサンプルにアクセスします。

- gallery_tree.fla
- gallery_tween.fla

これらのファイルには、スクリプトアニメーションが組み込まれている SWF ファイルにイメージファイルをダウンロードする間に、ActionScript を使用してムービークリップを動的に制御する方法の例が含まれています。

コードによるオブジェクトの移動

ActionScript を使用したオブジェクトの移動は、オブジェクトの _alpha プロパティの変更と似ていますが、この場合は、オブジェクトの _x プロパティまたは _y プロパティを変更します。

次の手順では、動的にロードされた JPEG イメージをアニメーション処理し、ステージ上で水平方向にスライドします。

コードを使用してステージ上のインスタンスを移動するには :

1. 新しい Flash ドキュメントを moveClip.fla という名前で作成します。
2. プロパティインスペクタでドキュメントのフレームレートを 24 fps に変更します。

   フレームレートを 24 fps のような高い値に設定すると、アニメーションはスムーズに再生されます。
3. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
// ムービークリップインスタンスを作成
this.createEmptyMovieClip("img1_mc", 10);
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function (target_mc:MovieClip):Void {
   target_mc._x = Stage.width;
   target_mc.onEnterFrame = function() {
      target_mc._x -= 3; // 現在の _x 位置を 3 ピクセル減少
      if (target_mc._x <= 0) {
         target_mc._x = 0;
         delete target_mc.onEnterFrame;
      }
   };
};
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
// ムービークリップにイメージをロード
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
   img1_mc);
```

   このコード例は、リモート Web サーバーから外部イメージをロードし、イメージのロードが完了したら、ステージ上でイメージを水平方向に動かします。onEnterFrame イベントハンドラの代わりに setInterval() 関数を使用してイメージをアニメーション処理することもできます。
4. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   イメージがロードされ、ステージの右から左上へ向かって動きます。

onEnterFrame イベントハンドラまたは setInterval() 関数を使用してイメージをアニメーション処理する方法の詳細については、418 ページの「コードによるオブジェクトのフェード」を参照してください。

スクリプトで作成された Flash のアニメーションのサンプルソースファイル "animation.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Animation" フォルダに移動してサンプルにアクセスします。

フォトギャラリアプリケーションのサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Galleries" フォルダに移動して次のサンプルにアクセスします。

- gallery_tree.fla
- gallery_tween.fla

これらのファイルには、スクリプトアニメーションが組み込まれている SWF ファイルにイメージファイルをダウンロードする間に、ActionScript を使用してムービークリップを動的に制御する方法の例が含まれています。

コードによるイメージのパン

ActionScript コードを使用して、Flash ドキュメント内で大きなイメージを簡単にパンできます。イメージの大きさがステージの大きさに合わない場合や、ステージの端から端へムービークリップをパンするアニメーション効果を作成する場合に有効です。たとえば、ステージのサイズよりも大きなパノラマイメージがあり、そのイメージサイズを縮小したり、ステージサイズを拡大したりせずに表示するには、その大きなイメージのマスクとして機能するムービークリップを作成します。

次の手順は、ムービークリップを動的にマスクし、`onEnterFrame` イベントハンドラを使用して、マスクの背後でイメージをアニメーション処理する方法を示しています。

コードを使用してステージでインスタンスをパンするには :

1. 新しい Flash ドキュメントを pan.fla という名前で作成します。
2. プロパティインスペクタでドキュメントのフレームレートを 24 fps に変更します。

   フレームレートを 24 fps のような高い値に設定すると、アニメーションはスムーズに再生されます。
3. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
System.security.allowDomain("http://www.helpexamples.com/");
// 変数を初期化
var direction:Number = -1;
var speed:Number = 5;
// イメージをロードするクリップを作成
this.createEmptyMovieClip("img_mc", 10);
// マスクとして使用するクリップを作成
this.createEmptyMovieClip("mask_mc", 20);
// Drawing API を使用してマスクを描画または作成
with (mask_mc) {
   beginFill(0xFF0000, 0);
   moveTo(0, 0);
   lineTo(300, 0);
   lineTo(300, 100);
   lineTo(0, 100);
   lineTo(0, 0);
   endFill();
```

```
}

var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip) {
  // ターゲットムービークリップのマスクを mask_mc に設定
  target_mc.setMask(mask_mc);
  target_mc.onEnterFrame = function() {
    target_mc._x += speed * direction;
    // target_mc が端に到達したら、アニメーション方向を反転
    if ((target_mc._x <= -(target_mc._width-mask_mc._width)) ||
  (target_mc._x >= 0)) {
      direction *= -1;
    }
  };
};

var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mcl_obj);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

このコードの最初のセクションは、direction と speed の 2 つの変数を定義しています。direction 変数は、マスクされたイメージを左から右 (1) と右から左 (-1) のどちらの方向にスクロールするかを制御します。speed 変数は、onEnterFrame イベントハンドラが呼び出されるたびに移動するピクセル数を制御します。大きな数値を指定すると、アニメーションの移動は速くなりますが、表示がスムーズではなくなります。

コードの次のセクションは、img_mc と mask_mc の 2 つの空のムービークリップを作成します。Drawing API を使用して、mark_mc ムービークリップ内に 300 x 100 ピクセルの矩形を描画しています。次に、最後のコードブロックで作成する MovieClipLoader インスタンスでリスナーとして使用される新しいオブジェクト (mcl_obj) を作成します。このオブジェクトは、onLoadInit イベントのイベントリスナーを定義し、動的にロードされたイメージをマスクし、スクロールするアニメーションを設定します。イメージがマスクの左端または右端に到達すると、アニメーションを反転します。

最後のコードブロックは、MovieClipLoader インスタンスを定義し、前のコードで作成したリスナーオブジェクトを指定し、さらに、img_mc ムービークリップへの JPEG のロードを開始します。

4. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

イメージがロードされ、繰り返し左右にパンして動きます。イメージは実行時にマスクされます。元のイメージを確認するには、オンラインでイメージを表示します (http://www.helpexamples.com/flash/images/image1.jpg)。

ビットマップのキャッシュ、スクロール、およびパフォーマンスについて

Flash では、ビットマップのキャッシュを使用して、アプリケーションで静止的なムービークリップのパフォーマンスを向上できます。`MovieClip.cacheAsBitmap` または `Button.cacheAsBitmap` プロパティを `true` に設定すると、ムービークリップまたはボタンインスタンスの内部ビットマップ表現がキャッシュされます。これで、複雑なベクターコンテンツを格納しているムービークリップのパフォーマンスを向上させることができます。ビットマップがキャッシュされているムービークリップのすべてのベクターデータは、メインステージではなくビットマップに描画されます。

メモ	その後、ビットマップは、最も近いピクセル境界に吸着された非伸縮、非回転のピクセルとして、メインステージにコピーされます。ピクセルは、親オブジェクトと 1 対 1 でマップされます。ビットマップの境界が変更されると、ビットマップは伸縮されずに再作成されます。

ボタンまたはムービークリップインスタンスのキャッシュの詳細については、第 10 章の「ムービークリップの操作」の次のセクションを参照してください。

- 318 ページの「ActionScript を使用したムービークリップのキャッシュとスクロールについて」
- 322 ページの「ムービークリップのキャッシュ」
- 324 ページの「ムービークリップの背景の設定」

`cacheAsBitmap` プロパティは、ほとんどのコンテンツが静的で、頻繁に拡大・縮小、回転されないムービークリップに使用するのが適しています。そのようなムービークリップでは、`cacheAsBitmap` を使用することによって、ムービークリップの変換時 (x 位置と y 位置の変更時) のパフォーマンスを向上できます。この機能の使用方法の詳細については、320 ページの「キャッシュを有効にする状況」を参照してください。

インスタンスとスクロールテキストに対してビットマップキャッシュを適用する方法を示すサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。次のサンプルを使用できます。

- cacheBitmap.fla。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/CacheBitmap" フォルダに移動します。
- aliasing.fla。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Advanced Anti-Aliasing" フォルダに移動します。

Tween クラスと TransitionManager クラスについて

Flash CS3 Professional をインストールすると、Tween クラスと TransitionManager クラスもインストールされます。このセクションでは、この 2 つの強力なクラスをムービークリップと Adobe コンポーネントで使用して、アニメーションを SWF ファイルに簡単に追加する方法を説明します。

Flash でスライドプレゼンテーションまたはフォームアプリケーションを作成する場合 (ActionScript 2.0 のみ)、スライド間に各種のトランジションを追加するビヘイビアを選択できます。これは、PowerPoint プレゼンテーションを作成する場合に似ています。この機能をスクリーンアプリケーションに追加するには、Tween クラスと TransitionManager クラスを使用します。これらにより、選択したビヘイビアに基づいてスクリーンをアニメーション化する ActionScript が生成されます。

Flash では、Tween クラスと TransitionManager クラスをスクリーンベースのドキュメント以外でも使用できます。たとえば、これらのクラスを、Adobe Component Architecture のバージョン 2 のコンポーネントセットやムービークリップで使用できます。ComboBox コンポーネントのアニメーション方法を変更する場合、TransitionManager クラスを使用すると、メニューが開く際に何らかの " イージング " を追加できます。イージングとは、アニメーションの中で動きを徐々に加速または減速することを意味し、アニメーションをよりリアルに見せるために役立ちます。タイムライン上のモーショントゥイーンを作成するか、カスタムコードを記述する代わりに、Tween クラスと TransitionManager クラスを使用して、アニメーション化された独自のメニューシステムを作成することもできます。

メモ	Tween クラスと TransitionManager クラスを使用できるのは ActionScript 2.0 のみですが、これらのクラスは Flash で使用できます。

Tween クラスの各メソッドおよびプロパティについては、『ActionScript 2.0 リファレンスガイド』を参照してください。TransitionManager クラスの各メソッドおよびプロパティについては、『ActionScript 2.0 リファレンスガイド』を参照してください。パッケージの操作の詳細については、443 ページの「フィルタパッケージの操作」を参照してください。

Tween クラスと TransitionManager クラスを使用するサンプルのソースファイル "tweenProgress.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/Tween ProgressBar" フォルダに移動してサンプルにアクセスします。

Tween クラスと TransitionManager クラスの詳細については、次のトピックを参照してください。

- 428 ページの「ファイルに対するトゥイーンとトランジションの追加 (Flash Professional 8 のみ)」
- 430 ページの「TransitionManager クラスと Tween クラスを使用したアニメーション」
- 432 ページの「イージングクラスとメソッドについて」
- 434 ページの「Tween クラスについて」
- 435 ページの「Tween クラスの使用」
- 440 ページの「TransitionManager クラスと Tween クラスの組み合わせ」

ファイルに対するトゥイーンとトランジションの追加 (Flash Professional 8 のみ)

メモ

このセクションでは、Flash Professional スライドプレゼンテーションにトゥイーンとトランジションを追加し、それらが Flash Professional ユーザーにどのように表示されるかを説明します。コードを使用する場合は、Flash Basic 8 (または Flash Professional 8) でも、トゥイーンとトランジションを追加できます。次のセクションで、具体的な方法を例で示します。

Tween クラスと TransitionManager クラスは、簡単な ActionScript で SWF ファイルの構成部品にアニメーションを追加できるように設計されています。Flash オーサリング環境には、これらの作成済みクラスを使用してスクリーンベースのアプリケーションにトランジションを追加するためのビヘイビアが用意されています。スライドプレゼンテーションまたはフォームアプリケーションを作成するために、スライド間に各種のトランジションを追加するビヘイビアを選択できます。

Flash のムービークリップでこれらのトランジションを使用し始める前に、スクリーンベースのアプリケーション使用時のトランジションの動作を確認する必要があります。

スライドプレゼンテーション内にトランジションを作成する ActionScript を表示するには :

1. Flash Professional 8 で、[ファイル]-[新規] を選択し、新しいスライドプレゼンテーションを作成します。
2. [一般] タブで、[Flash スライドプレゼンテーション] を選択し、[OK] をクリックします。
3. [ウィンドウ]-[ビヘイビア] を選択して、[ビヘイビア] パネルを開きます。
4. [ビヘイビアの追加] (+) ボタンをクリックします。
5. ポップアップメニューから [スクリーン]-[トランジション] を選択して、[トランジション] ダイアログボックスを開きます。
6. [ズーム] トランジションを選択します。
7. [トータル] テキストボックスに「1」と入力します。
8. [イージング] ポップアップメニューから [バウンス] を選択します。
9. [OK] をクリックして設定を適用し、ダイアログボックスを閉じます。

   これで約 15 行の ActionScript がスライドに直接追加されます。次のコードは、関連するトランジションコードです。

```
mx.transitions.TransitionManager.start(eventObj.target,
  {type:mx.transitions.Zoom, direction:0, duration:1,
  easing:mx.transitions.easing.Bounce.easeOut, param1:empty, param2:empty});
```

   このコードは、TransitionManager クラスを呼び出し、指定したイージングメソッド `mx.transitions.easing.Bounce.easeOut` を使用した [ズーム] トランジションを適用します。この場合は、選択したスライドにトランジションが適用されます。この効果をムービークリップに適用する場合は、Flash アニメーション内で使用する ActionScript を変更します。最初のパラメータを `eventObj.target` から目的のムービークリップのインスタンス名に変更するだけで、ムービークリップシンボルを使用するようにコードを変更できます。

Flash には 10 種類のトランジションが用意されています。それらは、イージングメソッドといくつかの省略可能なパラメータを使用してカスタマイズできます。既に述べたように、イージングとは、アニメーションの中で動きを徐々に加速または減速することを意味し、アニメーションをよりリアルに見せるために役立ちます。たとえば、ボールは、アニメーションの始まり近くでは徐々にその速度を増しますが、その後、徐々に速度を低下してアニメーションの最後に完全停止します。このような加速と減速を表す式は多数あり、それに基づいてイージングアニメーションが変化します。

次の表は、Flash Basic 8 に含まれるトランジション (コード使用) および Flash Professional 8 に含まれるトランジション (コードまたはビヘイビアを使用) を示します。

トランシジョン	説明
アイリス	ズームインするシェイプのアニメーションマスクを使用して、スクリーンまたはムービークリップを表示します。
ワイプ	水平方向に移動するシェイプのアニメーションマスクを使用して、スクリーンまたはムービークリップを表示します。
分解	消える矩形または表示される矩形を使用して、スクリーンまたはムービークリップをマスクします。
ブラインド	消える矩形または表示される矩形を使用して、次のスクリーンまたはムービークリップを表示します。
フェード	スクリーンまたはムービークリップをフェードインまたはフェードアウトします。
フライ	特定の方向からスクリーンまたはムービークリップにスライドインします。
ズーム	スクリーンまたはムービークリップをズームインまたはズームアウトします。
スクィーズ	現在のスクリーンまたはムービークリップを水平方向または垂直方向に拡大・縮小します。
回転	現在のスクリーンまたはムービークリップを回転します。
写真	スクリーンまたはムービークリップを写真のフラッシュのように表示します。

アニメーションに適用できるカスタマイズ内容は各トランジションごとに若干異なります。[トランジション] ダイアログボックスでは、効果をスライドまたはフォームに適用する前に、サンプルアニメーションをプレビューできます。

ヒント

イージングクラスの異なるメソッドを使用して各トランジションをプレビューするには、"‹ブートドライブ›¥Program Files¥Adobe¥Adobe Flash CS3¥‹言語›¥First Run¥Behaviors¥" フォルダまたは "‹*Macintosh HD*›:Applications:Adobe Flash CS3:First Run:Behaviors:" にある "Transition.swf" をダブルクリックして、この SWF ファイルをスタンドアローンプレーヤーで開きます。

TransitionManager クラスと Tween クラスを使用したアニメーション

Flash では、TransitionManager クラスと Tween クラスを使用して、ActionScript により、アニメーションをムービークリップ、コンポーネント、およびフレームに追加できます。TransitionManager クラスまたは Tween クラスを使用しない場合は、カスタムコードを記述して、ムービークリップをアニメーション化するか、透明度のレベル (アルファ) と座標 (位置) を変更する必要があります。アニメーションにイージングを追加すると、ActionScript (および計算) が急激に複雑になる可能性があります。ただし、特定のアニメーションのイージングを変更する場合、これらの作成済みクラスを使用すれば、スムーズなアニメーションの作成に必要な数式を新しく自分で見つけ出すのではなく別のクラスを選択すれば済みます。

次の手順は、ムービークリップをアニメーション化し、TransitionManager クラスを使用してステージ上でズームインするようにします。

TransitionManager クラスを使用してムービークリップをアニメーション化するには :

1. [ファイル]-[新規] を選択し、[Flash ドキュメント] を選択します。
2. [OK] をクリックして新しい FLA ファイルを作成します。
3. FLA ファイルを zoom.fla として保存します。
4. [ファイル]-[読み込み]-[ステージに読み込み] を選択し、ハードディスク上のイメージを選択して FLA ファイルに読み込みます。

   イメージはビットマップイメージとしてファイルに読み込まれるため、イメージを手動でムービークリップシンボルに変換する必要があります。
5. [開く] をクリックしてイメージを読み込みます。
6. ステージ上で、読み込まれたイメージを選択し、[修正]-[シンボルに変換] を選択します。
7. シンボルに img1 という名前を付け、ビヘイビアを [ムービークリップ] に設定したことを確認します。

   デフォルトでは、シンボルの基準点はシンボルの左上隅となります。
8. [OK] をクリックして、ビットマップイメージをムービークリップに変換します。
9. イメージを選択したまま、プロパティインスペクタ ([ウィンドウ]-[プロパティ]) を開き、ムービークリップにインスタンス名 img1_mc を割り当てます。

10. メインタイムラインのフレーム 1 を選択し、[アクション] パネルに次の ActionScript を追加します。

```
mx.transitions.TransitionManager.start(img1_mc, {type:mx.transitions.Zoom,
  direction:0, duration:1, easing:mx.transitions.easing.Bounce.easeOut,
  param1:empty, param2:empty});
```

メモ

パッケージの操作の詳細については、443 ページの「フィルタパッケージの操作」を参照してください。

11. [制御]-[ムービープレビュー] を選択してムービーをテストします。

イメージが拡大し、元のサイズに戻る前に少しのバウンス効果が現れます。アニメーションの移動が速すぎる場合は、アニメーションの継続時間を 1 秒から 2 または 3 秒に増やします (たとえば、duration:3)。

イメージの左上隅が固定され、右下隅に向かって拡大することがわかります。これは [トランジション] ダイアログボックスに表示されるプレビューとは異なります。

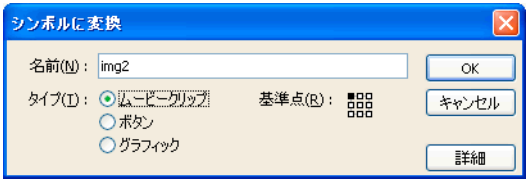
Tween クラスと TransitionManager クラスを使用すると、複雑なアニメーションの作成が簡単になり、タイムライン上にモーショントゥイーンやシェイプトゥイーンを作成する必要がなくなります。さらに、イージングメソッドを作成する際に複雑な数式を記述する必要がないという大きなメリットがあります。イメージを隅のアンカーポイントではなく中央からズームインする場合は、イメージをビットマップからシンボルに変換するときに、シンボルの基準点を変更します。

イメージの中央からイメージをズームインするには :

1. 前の手順を完了します。
2. "zoom.fla" ファイルを開き、[ファイル]-[名前を付けて保存] を選択して、ドキュメントの新しいコピーを保存します。

   ファイルを zoom2.fla として保存します。
3. ビットマップシンボルのコピーを、[ライブラリ] パネルから現在のムービークリップシンボルの横にあるステージ上にドラッグします。
4. ステージ上でビットマップイメージを選択したまま、F8 キーを押してシンボルをムービークリップに変換します。

   シンボルに img2 という名前を付けます。
5. [シンボルに変換] ダイアログボックスで、座標グリッドの中心をクリックして基準点をビットマップの中心に設定し、[OK] をクリックします。



6. ステージ上の新しいムービークリップを選択し、プロパティインスペクタを使用して、そのインスタンスに img2_mc という名前を付けます。

7. メインタイムラインのフレーム 1 を選択し、既存のコードに次の ActionScript を追加します。

```
mx.transitions.TransitionManager.start(img2_mc, {type:mx.transitions.Zoom,
  direction:mx.transitions.Transition.IN, duration:1,
  easing:mx.transitions.easing.Bounce.easeOut});
```

8. [制御]-[ムービープレビュー] を選択してアニメーションをテストします。

   これで 2 番目のムービークリップは、隅からでなくシンボルの中央から拡大するようになります。

メモ	トランジションによっては、基準点の配置場所を変えると表示が変化するものがあります。基準点を変更することが、SWF ファイルでのアニメーションの表示方法に大きく影響する可能性があります。たとえば、シンボルの基準点がデフォルトの左上隅にあるときに Zoom トランジションを使用すると、トランジションは左上隅から開始します。

Tween クラスと TransitionManager クラスの各メソッドおよびプロパティについては、『ActionScript 2.0 リファレンスガイド』を参照してください。

Tween クラスと TransitionManager クラスを使用してスクリプトアニメーションを追加するサンプルのソースファイル "tweenProgress.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Tween ProgressBar" フォルダに移動してサンプルにアクセスします。

イージングクラスとメソッドについて

「428 ページの「ファイルに対するトゥイーンとトランジションの追加 (Flash Professional 8 のみ)」」では、Bounce イージングクラスを使用して、ムービークリップにバウンス効果を追加する方法を説明しています。Bounce の他にも、Flash 8 には次の表に示す 5 つのイージングクラスが用意されています。

トランシジョン	説明
Back	アニメーションをトランジション範囲の片側または両側を一度超えて拡張し、オーバーフロー効果を発生させます。
Bounce	トランジション範囲を超えずに、この範囲の片側または両側でバウンス効果を発生させます。バウンドの回数は継続時間に比例します。継続時間が長くなると、バウンド回数が増えます。
Elastic	トランジション範囲の片側または両側で外部に伸縮効果を発生させます。伸縮度は、長さの影響を受けません。
Regular	片側または両側で動きを減速します。この機能を使用して、加速効果、減速効果、または両方を追加できます。

トランジション	説明
Strong	一端または両端で実行速度を減速します。これは Regular と似ていますが、より大きい効果があります。
None	効果、減速、または加速を設定せずに、最初から最後まで等速で実行されます。このようなトランジションは線状トランジションと呼ばれます。

上記の 6 つの各イージングクラスには、次の表に示す 3 つのイージングメソッドがあります。

メソッド	説明
`easeIn`	トランジションの最初にイージング効果を発生させます。
`easeOut`	トランジションの最後にイージング効果を発生させます。
`easeInOut`	トランジションの最初と最後にイージング効果を発生させます。

Flash または ActionScript エディタでこれらのクラスを開く場合、クラスファイルは、Windows の場合は "‹ ハードディスク ›¥Program Files¥Adobe¥Adobe Flash CS3¥‹ 言語 ›¥First Run¥Classes¥mx¥transitions¥easing¥" フォルダ (デフォルトのインストールの場合) に、Macintosh の場合は "‹*Macintosh HD*›:Applications:Adobe Flash CS3:First Run:Classes:mx:transitions:easing" フォルダにあります。

430 ページの「TransitionManager クラスと Tween クラスを使用したアニメーション」のイメージをズームする手順では、mx.transitions.easing.Bounce.easeOut イージングクラスおよびメソッドを使用しました。ハードディスクのフォルダでは、ActionScript が "Bounce.as" クラス内の `easeOut()` メソッドを参照しています。この ActionScript ファイルは、"easing" フォルダにあります。

Tween クラスと TransitionManager クラスの各メソッドおよびプロパティについては、『ActionScript 2.0 リファレンスガイド』を参照してください。

ヒント	イージングクラスの異なるメソッドを使用して各トランジションをプレビューするには、"‹ ブートドライブ ›¥Program Files¥Adobe¥Adobe Flash CS3¥‹ 言語 ›¥First Run¥Behaviors¥" または "‹*Macintosh HD*›:Applications:Adobe Flash CS3:First Run:Behaviors:" にある "Transition.swf" をダブルクリックして、この SWF ファイルをスタンドアローンプレーヤーで開きます。

Tween クラスと TransitionManager クラスを使用してスクリプトアニメーションを追加するサンプルのソースファイル "tweenProgress.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Tween ProgressBar" フォルダに移動してサンプルにアクセスします。

Tween クラスについて

Tween クラスを使用すると、ステージ上のムービークリップを簡単に移動、サイズ変更、およびフェードできます。mx.transitions.Tween クラスのコンストラクタに対して指定するパラメータの名前とタイプを次に示します。

```
function Tween(obj, prop, func, begin, finish, duration, useSeconds) {
    // code ...
}
```

`obj` Tween インスタンスのターゲットになるムービークリップオブジェクトです。

`prop` 値がトゥイーンされる obj のプロパティのストリング名です。

`func` トゥイーンされるオブジェクトのプロパティ値のイージング効果を計算するイージングメソッドです。

`begin` prop (トゥイーンされるターゲットオブジェクトのプロパティ) の開始値を示す数値です。

`finish` prop (トゥイーンされるターゲットオブジェクトのプロパティ) の終了値を示す数値です。

`duration` トゥイーンモーションの継続時間を示す数値です。省略した場合、デフォルトで継続時間が infinity に設定されます。

`useSeconds` duration パラメータで指定した値の単位に関して、秒 (true の場合) またはフレーム (false の場合) のどちらを使用するかを示すブール値です。

たとえば、ステージ上でムービークリップを移動させるとします。この場合、タイムラインにキーフレームを追加して、キーフレーム間にモーショントゥイーンやシェイプトゥイーンを挿入するか、何らかのコードを onEnterFrame イベントハンドラ内に記述できます。または、setInterval() 関数を使用して、一定の間隔で関数を呼び出すこともできます。Tween クラスを使用する場合は、ムービークリップの _x プロパティと _y プロパティを変更できる別のオプションもあります。前に説明したイージングメソッドを追加することもできます。Tween クラスを活用する場合、次の ActionScript を使用できます。

```
new mx.transitions.Tween(ball_mc, "_x", mx.transitions.easing.Elastic.easeOut,
  0, 300, 3, true);
```

この ActionScript コードでは、ball_mc ムービークリップをステージ上で x 軸に沿って (左から右へ) 動かす、Tween クラスの新しいインスタンスを作成します。このムービークリップでは 0 ピクセルから 300 ピクセルまでを 3 秒間でアニメーション化します。また、ActionScript では伸縮イージングメソッドを適用します。つまり、ボールは x 軸上の 300 ピクセルを越えて拡張してから、流れるようなモーション効果を使用して、アニメーションを元に戻って再生します。

Tween クラスと TransitionManager クラスを使用してスクリプトアニメーションを追加するサンプルのソースファイル "tweenProgress.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Tween ProgressBar" フォルダに移動してサンプルにアクセスします。

Tween クラスの使用

Flash ドキュメント内の複数の場所で Tween クラスを使用する場合、import ステートメントを使用することもできます。この場合、Tween クラスを使用するたびに完全修飾のクラス名を指定せずに、Tween クラスとイージングメソッドを読み込むことができます。次に手順を示します。

Tween クラスを読み込んで使用するには :

1. 新しいドキュメントを作成し、easeTween.fla という名前を付けます。
2. ステージ上にムービークリップを作成します。
3. ムービークリップインスタンスを選択し、プロパティインスペクタ内の [インスタンス名] テキストボックスに「ball_mc」と入力します。
4. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
new Tween(ball_mc, "_x", Elastic.easeOut, Stage.width, 0, 3, true);
```

このコードの例では、2 つの import ステートメントを使用します。最初の import ステートメントは、mx.transitions.Tween クラスのみを読み込み、2 番目のステートメントは、ワイルドカード (*) ショートカットを使用して、単一のコード行で 6 つの各イージングクラスを読み込みます。2 番目のステートメントは、クラスのパッケージ全体を読み込みます。

> メモ
> パッケージの操作の詳細については、443 ページの「フィルタパッケージの操作」を参照してください。

5. [制御]-[ムービープレビュー] を選択し、アニメーションを確認します。

Flash のマニュアルでは、" パッケージ " とはクラスファイルを格納するディレクトリを意味します。このディレクトリは所定のクラスパスディレクトリ内に存在します。この場合、パッケージの場所は、Windows では "C:¥Program Files¥Adobe¥Adobe Flash CS3¥‹ 言語 ›¥First Run¥Classes¥mx¥transitions¥easing¥" フォルダに、Macintosh では "‹Macintosh HD›:Applications:Adobe Flash CS3:First Run:Classes:mx:transitions:easing" フォルダにあります。1 つのパッケージ全体を読み込む方法は、6 つのクラスを別々に読み込むよりもはるかに便利です。完全修飾名で mx.transitions.Tween クラスを指定しなくても、ActionScript から Tween というクラス名を直接参照できるようになります。同様に、イージングクラスの完全修飾名、たとえば mx.transitions.easing.Elastic.easeOut を使用する代わりに、ActionScript コードに「Elastic.easeOut」と入力できます。詳細については、443 ページの「フィルタパッケージの操作」を参照してください。

同様のコードを使用して、次の手順に示すように、_x プロパティでなく、_alpha プロパティを設定してインスタンスをフェードインおよびフェードアウトします。

Tween クラスを使用してインスタンスをフェードするには :

1. 新しいドキュメントを作成し、fadeTween.fla という名前を付けます。
2. ステージ上にムービークリップを作成します。
3. ムービークリップインスタンスを選択し、プロパティインスペクタ内の [インスタンス名] テキストボックスに「ball_mc」と入力します。
4. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
new Tween(ball_mc, "_alpha", Strong.easeIn, 100, 0, 3, true);
```

   これで ball_mc は、ステージ上を移動する代わりに、100% 表示された状態から 3 秒間で完全に透明な状態にフェードアウトします。シンボルをより速くフェードアウトさせるには、duration パラメータを 3 から 1 または 2 に変更します。

5. [制御]-[ムービープレビュー] を選択し、アニメーションを確認します。

   ドキュメントのフレームレートを変更すると、アニメーションはよりスムーズに再生されます。アニメーションとフレームレートについては、417 ページの「アニメーションとフレームレートについて」を参照してください。

秒単位ではなく、シンボルをフレーム単位でフェードアウトさせることができます。Tween クラスでの継続時間の長さを秒単位でなくフレーム単位で設定するには、最後のパラメータ useSeconds を true から false に変更します。パラメータを true に設定すると、指定した長さは秒単位となります。パラメータを false に設定すると、長さはトゥイーンに使用するフレーム数になります。次の手順は、秒単位ではなく、フレーム単位でトゥイーンを設定する方法を示します。

秒数ではなくフレーム数で継続時間を設定するには :

1. 新しいドキュメントを作成し、framesTween.fla という名前を付けます。
2. ステージ上にムービークリップを作成します。
3. ムービークリップインスタンスを選択し、プロパティインスペクタ内の [インスタンス名] テキストボックスに「ball_mc」と入力します。
4. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
new Tween(ball_mc, "_alpha", Strong.easeIn, 100, 0, 24, false);
```

   このコードでは、Strong.easeIn イージングメソッドを使用して、ball_mc インスタンスをフェードアウトします。インスタンスを 3 秒間でフェードアウトする代わりに、24 フレームでフェードアウトします。

5. [制御]-[ムービープレビュー] を選択し、アニメーションを確認します。

   しばらくすると、インスタンスが 24 フレームでフェードアウトします。

6. オーサリング環境に戻り、プロパティインスペクタを開きます。
7. ドキュメントのフレームレートを 24 fps に変更します。

   FLA ファイルのフレームレートを大きくすると、インスタンスがフェードアウトするスピードが速くなります。アニメーションとフレームレートについては、417 ページの「アニメーションとフレームレートについて」を参照してください。

秒の代わりにフレームを使用すると、柔軟性は向上しますが、継続時間の長さは現在の Flash ドキュメントのフレームレートに関連することになります。Flash ドキュメントで 12 fps (1 秒あたりのフレーム数) のフレームレートを使用している場合、前のコードでは、インスタンスを 2 秒間 (24 フレーム /12 fps = 2 秒) でフェードアウトします。ただし、フレームレートが 24 fps の場合は、同じコードでインスタンスは 1 秒間 (24 フレーム /24 fps = 1 秒) でフェードアウトします。長さの指定にフレームを使用する場合、ドキュメントのフレームレートを変更すると、ActionScript を変更しなくてもアニメーションの速度が大幅に変化することがあります。

Tween クラスには、さらに役立つ機能がいくつかあります。たとえば、次の手順に示すように、アニメーションが終了したときに呼び出されるイベントハンドラを記述できます。

アニメーションが完了したときにコードをトリガするには :

1. 新しいドキュメントを作成し、triggerTween.fla という名前を付けます。
2. ステージ上にムービークリップを作成します。
3. ムービークリップインスタンスを選択し、プロパティインスペクタ内の [インスタンス名] テキストボックスに「ball_mc」と入力します。
4. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var tween_handler:Tween = new Tween(ball_mc, "_alpha", Strong.easeIn, 100, 0,
  3, true);
tween_handler.onMotionFinished = function() {
  trace("onMotionFinished triggered");
};
```

   この ActionScript を FLA ファイルでテストすると、ステージ上で `ball_mc` のフェードが完了したときに、[出力] パネルに "onMotionFinished triggered" というメッセージが表示されます。
5. [制御]-[ムービープレビュー] を選択し、アニメーションを確認します。

   しばらくすると、インスタンスがフェードアウトします。トゥイーンが終了すると、[出力] パネルにメッセージが表示されます。

この機能の詳細については、第 6 章の「クラス」を参照してください。

continueTo() メソッドを使用したアニメーションの継続について

435 ページの「Tween クラスの使用」では、アプリケーションで Tween クラスを使用する方法を説明しています。たとえば、最初のアニメーションが完了した後にボールを移動させたい場合、少なくとも 2 つの方法があります。1 つは、onMotionFinished イベントハンドラを使用してボールを再度アニメーション化する方法ですが、Tween クラスでは、continueTo() メソッドを使用する方が簡単に実現できます。continueTo() メソッドは、トゥイーンアニメーションに対して、次の ActionScript に示すように、現在の値から新しい値まで継続するように指示します。

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var ball_tween:Tween = new Tween(ball_mc, "_x", Regular.easeIn, 0, 300, 3, true);
ball_tween.onMotionFinished = function() {
    ball_tween.continueTo(0, 3);
};
```

最初のトゥイーンが完了すると、ball_mc ムービークリップは、0 ピクセルの元の位置に戻ってトゥイーンし直します。次のコードは、continueTo() メソッドの関数プロトタイプです (見やすいように編集済み)。

```
function continueTo(finish:Number, duration:Number):Void {
    /* 省略 */
}
```

Tween コンストラクタメソッドには 7 つのパラメータが渡されますが、continueTo() メソッドに渡されるパラメータは 2 つのみです。次にコードを示します。

```
function Tween (obj, prop, func, begin, finish, duration, useSeconds) {
    /* 省略 */
}
```

continueTo() メソッドで不要な 5 つのパラメータ (obj、prop、func、begin、および useSeconds) は、前に Tween クラスを呼び出す際に定義したパラメータを使用します。continueTo() メソッドへの呼び出しでは、obj、prop、func (イージングタイプ) および useSeconds の各パラメータが、前に Tween クラスを呼び出したときのパラメータと同じであることを前提とします。continueTo() メソッドでは、begin パラメータの値を指定する代わりに、Tween クラスを呼び出した際の finish 値を使用します。次に ActionScript の例を示します。

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var ball_tween:Tween = new Tween(ball_mc, "_x", Regular.easeIn, 0, 300, 3, true);
ball_tween.onMotionFinished = function() {
     ball_tween.continueTo(0, 3);
};
```

このコードは、ball_mc インスタンスを x 軸に沿って 0 ピクセルから 300 ピクセルまで 3 秒間で移動します。アニメーションが終了すると、onMotionFinished イベントハンドラが continueTo() メソッドを呼び出します。continueTo() メソッドは、ターゲットオブジェクト (ball_mc) に対して、現在の位置を起点として x 軸に沿って 0 ピクセルまで 3 秒間アニメーションを実行すること、および同じイージングメソッドを使用することを指示します。continueTo() メソッドで未定義のパラメータについては、Tween コンストラクタメソッドへの呼び出しで指定した値が使用されます。continueTo() メソッドで継続時間の長さを指定しなければ、Tween コンストラクタへの呼び出しで指定した長さが使用されます。

連続して実行されるアニメーションの作成

アニメーションを停止せずに x 軸に沿って繰り返し再生させることができます。Tween クラスには、この種のアニメーションを実現する手段として、yoyo() というわかりやすい名前のメソッドが用意されています。yoyo() メソッドは、onMotionFinished イベントハンドラの実行を待機してから、begin パラメータと finish パラメータを入れ替えます。これによりアニメーションが再度開始します。次に手順を示します。

際限なく続くアニメーションを作成するには :

1. 新しい Flash ドキュメントを yoyo.fla という名前で作成します。
2. [アクション] パネルを開き、タイムラインのフレーム 1 に次の ActionScript を入力します。

```
import mx.transitions.Tween;
import mx.transitions.easing.*;

this.createEmptyMovieClip("box_mc", this.getNextHighestDepth());
with (box_mc) {
  beginFill(0xFF0000, 60);
  moveTo(0, 0);
  lineTo(20, 0);
  lineTo(20, Stage.height);
  lineTo(0, Stage.height);
  lineTo(0, 0);
  endFill();
}
```

コードの最初のセクションは、まず Tween クラスとイージングパッケージ内の各クラスを読み込みます。次のコードセクションは、box_mc というインスタンス名を持つ新しいムービークリップを作成し、幅が 20 ピクセルで高さがステージと同じ矩形を描画しています。

3. 前の手順で作成したコードの後に、次の ActionScript を追加します。

```
var box_tween:Tween = new Tween(box_mc, "_x", Regular.easeInOut, 0,
  Stage.width, 3, true);
box_tween.onMotionFinished = function() {
     box_tween.yoyo();
};
```

このコードは、box_mc ムービークリップをステージ上で *x* 軸に沿って 3 秒間アニメーション化する新しいトゥイーンを作成します。

4. [制御]-[ムービープレビュー] を選択して、アニメーションをテストします。

ボックスが左から右に移動してから、元に戻ります。アニメーションがスムーズでない場合は、ドキュメントのフレームレートを 12 fps から 24 fps に変更できます。

ボックスがステージの右端に近づくと、ボックスのアニメーションはステージの境界線の外にはみ出します。これは大したことではないように思えますが、矩形がステージの端から消え、少し経ってから再び表示されて逆方向に移動することは望ましくない場合があります。

これを調整するには、矩形をアニメーション化する距離を、0 ピクセルの位置から、ステージの幅から box_mc ムービークリップの幅を差し引いた値の位置までに変更する必要があります。

5. 矩形が消えないようにするには、手順 3 のコードの対応する行を、次のコードのように修正します。

```
var box_tween:Tween = new Tween(box_mc, "_x", Regular.easeInOut, 0,
  (Stage.width - box_mc._width), 3, true);
```

6. [制御]-[ムービープレビュー] を選択して、アニメーションをテストします。

矩形がステージの端から消える前にイージングが停止します。

TransitionManager クラスと Tween クラスの組み合わせ

TransitionManager クラスと Tween クラスを組み合わせると、特殊効果を生成できます。TransitionManager クラスを使用すると、Tween クラスを使用してムービークリップの _alpha プロパティを調整しながら、同じクリップを *x* 軸に沿って移動できます。各クラスでは、さまざまなイージングメソッドを使用できます。つまり、SWF ファイル内のオブジェクトに対して、多様なアニメーションを作成できます。Tween クラスの continueTo() メソッドと yoyo() メソッド、または onMotionFinished イベントハンドラを活用すると、ユニークな効果を作成できます。

TransitionManager クラスと Tween クラスを組み合わせると、次の手順に示すように、動的にロードされるムービークリップのアニメーションを作成し、リモートサーバーからムービークリップが完全にロードされた後に、ステージ上でフェードインさせることができます。

TransitionManager クラスと Tween クラスを併用するには :

1. 新しい Flash ドキュメントを作成し、combination.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import mx.transitions.*;
import mx.transitions.easing.*;

var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip) {
    new Tween(target_mc, "_alpha", Strong.easeIn, 0, 100, 2, true);
    TransitionManager.start(target_mc, {type:Fly, direction:Transition.IN,
  duration:3, easing:Elastic.easeInOut, startPoint:6});
};

var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mcl_obj);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  this.createEmptyMovieClip("img_mc", this.getNextHighestDepth()));
```

このコードは、大きく次の 3 つのセクションに分かれています。

最初のセクションは、transitions パッケージと `transitions.easing` パッケージ内のクラスを読み込みます。この例では、transitions パッケージ全体を読み込んでいるので、Tween クラス、TransitionManager クラス、または選択したトランジション (この場合、"*Fly*") の完全修飾名を入力する必要はありません。これで、入力するコード量を削減し、入力ミスの可能性をある程度排除できます。

ActionScript の 2 番目のセクションでは、3 番目のコードセクションで作成する、MovieClipLoader クラスインスタンスのリスナーオブジェクトを作成します。ターゲットのムービークリップが MovieClipLoader インスタンスにロードされると、`onLoadInit` イベントで、Tween クラスと TransitionManager クラスの両方を呼び出すコードブロックが呼び出されて実行されます。このイベントハンドラでは、Tween クラスの `_alpha` プロパティを変更してターゲットのムービークリップをフェードインし、*x* 軸に沿ってターゲットのムービークリップを " 飛ばし " ます。

ActionScript の 3 番目のセクションでは、MovieClipLoader インスタンスを作成し、前に作成したリスナーオブジェクトを適用します。このため、ターゲットムービークリップの loader インスタンスは、`onLoadInit` イベントを受け取ることができます。次に、ターゲット JPEG イメージを、`createEmptyMovieClip()` メソッドを呼び出すことで動的に作成したムービークリップにロードします。

3. ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して、アニメーションをテスト環境で表示します。

サーバーからの外部 JPEG イメージのダウンロードが完了すると、イメージは徐々にフェードインし、ステージ上を左から右へアニメーション化されます。

Tween クラスの使用については、435 ページの「Tween クラスの使用」を参照してください。

Tween クラスと TransitionManager クラスの各メソッドおよびプロパティについては、『ActionScript 2.0 リファレンスガイド』を参照してください。

Tween クラスと TransitionManager クラスを使用してスクリプトアニメーションを追加するサンプルのソースファイル "tweenProgress.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Tween ProgressBar" フォルダに移動してサンプルにアクセスします。

フィルタ効果の使用

フィルタとは、Flash Player の実行時に表示されるムービークリップインスタンスなどのオブジェクトに適用される視覚効果です。フィルタには、ドロップシャドウ、ぼかし、グロー、ベベル、グラデーショングローおよびグラデーションベベルなどがあります。カラーフィルタを使用すると、ムービークリップの明度、コントラスト、彩度および色相を調整できます。フィルタは、Flash Basic 8 の Flash ユーザーインターフェイスを使用しても、Flash Basic 8 または Flash Professional 8 で ActionScript を使用しても、適用できます。

ムービークリップやボタン、テキストフィールドにフィルタ効果を適用するには、プロパティインスペクタの [フィルタ] タブを使用するか、ActionScript を使用します。ActionScript を使用してインスタンスにフィルタを適用する場合は、置き換えマップフィルタ (470 ページの「置き換えマップフィルタの使用」を参照) や、畳み込みフィルタ (468 ページの「畳み込みフィルタの使用」を参照) も使用できます。これらのフィルタはベクター定義に適用されるので、SWF ファイルにビットマップイメージを格納するというオーバヘッドが生じません。ActionScript を記述して、テキストフィールドやムービークリップ、ボタンに適用している既存のフィルタを変更することも可能です。

次の手順は、`onEnterFrame` イベントハンドラを使用して、ムービークリップにグローフィルタ効果を適用し、アニメーション化する方法を示します。

ムービークリップインスタンスに適用されたフィルタ効果をアニメーション化するには :

1. 新しい Flash ドキュメントを作成し、animFilter.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createEmptyMovieClip("box_mc", 10);
box_mc.lineStyle(20, 0x000000);
box_mc.beginFill(0x000000);
box_mc.moveTo(0, 0);
box_mc.lineTo(160, 0);
box_mc.lineTo(160, 120);
box_mc.lineTo(0, 120);
box_mc.lineTo(0, 0);
box_mc.endFill();
box_mc._x = 100;
box_mc._y = 100;

box_mc.filters = [new flash.filters.GlowFilter()];
var dir:Number = 1;
box_mc.blur = 10;
```

```
box_mc.onEnterFrame = function() {
   box_mc.blur += dir;
   if ((box_mc.blur >= 30) || (box_mc.blur <= 10)) {
      dir *= -1;
   }
   var filter_array:Array = box_mc.filters;
   filter_array[0].blurX = box_mc.blur;
   filter_array[0].blurY = box_mc.blur;
   box_mc.filters = filter_array;
};
```

このコードは 2 つは異なる機能を完了します。最初のセクションは、ムービークリップインスタンスを作成および配置し、ステージ上に黒の角丸矩形を描画します。2 番目のコードブロックは、ステージ上の矩形にグローフィルタを適用し、フィルタ効果をアニメーション化する onEnterFrame イベントハンドラを定義します。onEnterFrame イベントハンドラは、10 ピクセルと 30 ピクセルに設定されたぼかしの間で、グローフィルタをアニメーション化します。アニメーションが 30 より大きいか等しくなる、または、10 より小さいか等しくなると、反転します。

3. Flash ドキュメントの変更を保存し、[制御]-[ムービープレビュー] を選択して、SWF ファイルをテストします。

アプリケーションにおけるフィルタ操作の詳細については、次のトピックを参照してください。

- 443 ページの「フィルタパッケージの操作」
- 445 ページの「フィルタ、キャッシュおよび MovieClip クラスの操作」
- 447 ページの「ヒット検出およびフィルタの回転、傾斜、拡大・縮小について」
- 447ページの「オブジェクトインスタンスおよび BitmapData インスタンスへのフィルタの適用」
- 448 ページの「エラー処理、パフォーマンスおよびフィルタについて」

ActionScript を使用してフィルタを適用する例を示すサンプルのソースファイル "Filters.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Filters" フォルダに移動してサンプルにアクセスします。

フィルタパッケージの操作

パッケージとは、クラスファイルを含むディレクトリで、所定のクラスパスディレクトリ内に存在します。たとえば、flash.filters パッケージはハードディスク上のディレクトリで、Flash の各フィルタタイプ (BevelFilter、BlurFilter、DropShadowFilter など) に対応するいくつかのクラスファイルが入っています。クラスファイルがこのような形で整理されているので、クラスにアクセスするには、一定の手順をとる必要があります。クラスを " 読み込みむ " か、" 完全修飾クラス名 " を使用して参照します。

メモ

import ステートメントを使用するには、FLA ファイルの [パブリッシュ設定] ダイアログボックスの [Flash] タブで、ActionScript 2.0 および Flash Player 6 以降を指定する必要があります。

import ステートメントを使用すると、完全修飾名を指定しなくてもクラスにアクセスできるようになります。たとえば、スクリプト内で BlurFilter を使用する場合、このクラスを完全修飾名 (flash.filters.BlurFilter) で参照するか、import で読み込む必要があります。import で読み込むと、クラス名 (BlurFilter) での参照が可能になります。次の ActionScript コードは、import ステートメントを使用する場合と完全修飾名を使用する場合の違いを示します。

import ステートメントを使用して BlurFilter クラスを読み込まない場合、フィルタを使用するには、完全修飾名 (パッケージ名にクラス名を続けたもの) を使用する必要があります。

```
// import ステートメントを使わない場合
var myBlur:flash.filters.BlurFilter = new flash.filters.BlurFilter(10, 10, 3);
```

import ステートメントを使用した同じコードでは、完全修飾名を指定して、繰り替えし参照する必要がありません。クラス名の BlurFilter を指定することによって、アクセスできます。その結果、入力作業を減らしてタイプミスを防ぐことができます。

```
// import ステートメントを使う場合
import flash.filters.BlurFilter;
var myBlur:BlurFilter = new BlurFilter(10, 10, 3);
```

パッケージ内の複数のクラス (BlurFilter、DropShadowFilter、GlowFilter) を読み込むには、個々のクラスを読み込むための 2 つの方法のうちのいずれかを使用します。複数のクラスを読み込む 1 つ目の方法は、import ステートメントを個別に使用してそれぞれのクラスを読み込む方法です。次に例を示します。

```
import flash.filters.BlurFilter;
import flash.filters.DropShadowFilter;
import flash.filters.GlowFilter;
```

パッケージ内の各クラスを読み込む import ステートメントを個々に記述するのは、時間がかかるうえに、入力ミスが発生しやすくなります。import ステートメントでワイルドカード文字を使用すれば、クラスファイルを個別に読み込む必要がありません。あるレベルのパッケージ内にあるすべてのクラスを読み込むことができます。次の ActionScript コードは、ワイルドカード文字を使用した import ステートメントの例を示します。

```
import flash.filters.*; // flash.filters パッケージ内の各クラスを読み込む
```

import ステートメントは、それを呼び出している現在のスクリプト (フレームまたはオブジェクト) にのみ適用されます。たとえば、macr.util パッケージのすべてのクラスを Flash ドキュメントのフレーム 1 に読み込んだと仮定します。そのフレームでは、そのパッケージ内のクラスを、完全修飾名ではなく、クラス名で参照できます。ただし、他のフレームのスクリプトでクラス名を使用するには、そのパッケージのクラスを完全修飾名で参照するか、そのパッケージのクラスを読み込むための import ステートメントを追加する必要があります。

import ステートメントでは、指定したレベルのみのクラスが読み込まれます。たとえば、mx.transitions パッケージ内のすべてのクラスを import ステートメントで読み込む場合、"/transitions/" ディレクトリ内のクラスのみが読み込まれ、サブディレクトリ内のクラス (mx.transitions.easing パッケージ内のクラス) は読み込まれません。

ヒント	読み込んだクラスがスクリプト内で使用されなかった場合、そのクラスは SWF ファイルには出力されません。したがって、SWF ファイルのサイズを気にすることなく、大きなパッケージを読み込むことができます。クラスに関連付けられたバイトコードは、それが実際に使用された場合にのみ、SWF ファイルに含まれます。

ActionScript を使用してフィルタを適用する例を示すサンプルのソースファイル "Filters.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Filters" フォルダに移動してサンプルにアクセスします。

フィルタ、キャッシュおよび MovieClip クラスの操作

ムービークリップにフィルタが関連付けられている場合、SWF ファイルのロード時に、透明なビットマップとしてムービークリップ自身をキャッシュするようにマークされます。ムービークリップに少なくとも 1 つのフィルタが適用されていれば、cacheAsBitmap プロパティが true となり、実行時にムービークリップがビットマップとしてキャッシュされます。キャッシュされたビットマップは、フィルタ効果のソースイメージとして使用されます。各ムービークリップには通常、2 つのビットマップがあります。1 つはフィルタが適用されていない元のムービークリップで、2 つ目はフィルタ適用後の最終イメージです。実行時にムービークリップの外観が変更されていない限り、最終イメージを更新する必要がなく、その結果、パフォーマンスが向上します。

インスタンスに適用されたフィルタにアクセスするには、MovieClip.filters プロパティを呼び出します。このプロパティを呼び出すと、ムービークリップインスタンスに現在関連付けられている各フィルタオブジェクトが格納されている配列が返されます。1 つのフィルタには、次のような、そのフィルタ固有のプロパティのセットがあります。

```
trace(my_mc.filters[0].angle); // 45.0
trace(my_mc.filters[0].distance); // 4
```

通常の配列オブジェクトにアクセスして修正する場合と同じように、フィルタにアクセスして、修正できます。filters プロパティを使用して、設定や取得を行う場合、filters オブジェクトの複製が返されます。参照は返されません。

既存のフィルタを修正するには、次の手順のようなコードを使用します。

ムービークリップインスタンスに適用されたフィルタプロパティを変更するには :

1. 新しい Flash ドキュメントを作成し、modifyFilter.fla という名前を付けて保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createEmptyMovieClip("my_mc", 10);
// 四角形を描画
with (my_mc) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}
my_mc._x = 100;
my_mc._y = 100;

// デフォルトの DropShadowFilter 値を使用
my_mc.filters = [new flash.filters.DropShadowFilter()];
trace(my_mc.filters[0].distance); // 4
var filter_array:Array = my_mc.filters;
filter_array[0].distance = 10;
my_mc.filters = filter_array;
trace(my_mc.filters[0].distance); // 10
```

このコードの最初のセクションは、Drawing API を使用して赤色の四角形を作成し、ステージ上に配置します。2 番目のセクションは、四角形にドロップシャドウフィルタを適用します。次に、ステージ上の赤色の四角形に適用する現在のフィルタを格納する一時配列を作成します。最初のフィルタの `distance` プロパティを 10 ピクセルに設定し、修正されたフィルタが、`my_mc` ムービークリップインスタンスに再適用されます。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

メモ

現在、親の回転に応じて回転するフィルタや、一部の種類の回転をするフィルタはサポートしていません。ぼかしフィルタは、親オブジェクトツリーのアイテムの回転や傾斜に関係なく、常に完全な水平または垂直方向にぼかしを適用します。

ヒント

フィルタが適用されたコンテンツは、`cacheAsBitmap` プロパティが `true` に設定されたコンテンツと同じサイズ上の制約を受けます。SWF ファイル上でズームインして拡大し、いずれかの方向で、ビットマップ表現が 2880 ピクセルを超えると、フィルタは画面上で見えなくなります。フィルタを含む SWF ファイルをパブリッシュする場合、ズームオプションを無効に設定することをお勧めします。

ActionScript を使用してフィルタを適用する例を示すサンプルのソースファイル "Filters.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Filters" フォルダに移動してサンプルにアクセスします。

ヒット検出およびフィルタの回転、傾斜、拡大・縮小について

ムービークリップインスタンスの境界ボックスの外側で、フィルタが適用されない領域 (ドロップシャドウなど) は、ヒット検出の対象となるサーフェスの一部です。ヒット検出は、インスタンス同士の重なりや交差を検出します。ヒット検出はベクターベースなので、ビットマップ結果に対しては実行できません。たとえば、ボタンインスタンスにベベルフィルタを適用する場合、ベベルフィルタが適用されたインスタンスの領域では、ヒット検出を実行できません。

フィルタは、拡大・縮小、回転、傾斜には対応していません。インスタンス自体を拡大・縮小しても (`_xscale` と `_yscale` が 100% でない)、フィルタ効果はインスタンスと一緒に拡大・縮小しません。つまり、インスタンスの元のシェイプが回転、拡大・縮小、傾斜しても、フィルタはインスタンスと共に回転、拡大・縮小、傾斜しません。

フィルタを含むインスタンスをアニメーション化して、リアルな効果を作成できます。また、インスタンスをネストして BitmapData クラスを使用し、フィルタをアニメーション化して同様な効果を得ることができます。

オブジェクトインスタンスおよび BitmapData インスタンスへのフィルタの適用

フィルタの使い方は、フィルタの適用先のオブジェクトインスタンスによって異なります。オブジェクトまたは BitmapData インスタンスにフィルタを適用する場合は、次のガイドラインを参照してください。

- 実行時にムービークリップ、テキストフィールド、ボタンにフィルタを適用する場合は、`filters` プロパティを使用します。オブジェクトの `filters` プロパティを設定してもオブジェクトそのものは変更されません。また、`filters` プロパティをクリアすることで元に戻すことができます。
- フィルタを BitmapData インスタンスに適用する場合は、`BitmapData.applyFilter()` メソッドを使用します。BitmapData オブジェクトで `applyFilter()` を呼び出すことによって、ソース BitmapData オブジェクトとフィルタオブジェクトが取得され、フィルタを適用して得られるイメージが生成されます。

メモ	プロパティインスペクタの [フィルタ] タブを使用して、イメージやビデオにもフィルタ効果をオーサリング時に適用できます。

エラー処理、パフォーマンスおよびフィルタについて

1 つのアプリケーションで多くのフィルタを使用すると、メモリ使用量が増し、Flash Player のパフォーマンスが低下する問題が生じる場合があります。フィルタがアタッチされたムービークリップは、2 つの 32 ビットのビットマップを含むので、このようなビットマップを数多く使用すると、アプリケーションのメモリ使用量が非常に大きくなります。場合によっては、コンピュータのオペレーティングシステムから、メモリ不足を示すメッセージが表示されます。ステージ上に数千のビットマップを置くなど、アプリケーション内で多数のフィルタ効果を使用しない限り、最近のコンピュータで、メモリ不足のエラーメッセージが表示されることはほとんどありません。

しかし、メモリ不足のエラーメッセージが表示された場合は、次の事態が発生します。

- フィルタ配列が無視されます。
- ムービークリップが、標準ベクターレンダラを使用して描画されます。
- ムービークリップのビットマップがキャッシュされません。

メモリ不足のエラーメッセージが表示された後は、ムービークリップがフィルタ配列やビットマップキャッシュの使用を試みません。パフォーマンスに影響を与える他の要因として、適用する各フィルタの quality パラメータの値があります。quality パラメータの値を高く設定するほど、効果を表現する際の CPU とメモリの負荷が増し、この値を低くく設定すると、コンピュータリソースの使用量が少なくなります。したがって、多数のフィルタを必要以上に使用することは避け、また、可能な場合は、quality の値を低く設定してください。

注意	100 x 100 ピクセルのオブジェクトを 1 回ズームインすると、そのコンテンツのサイズが 200 x 200 ピクセルになるので、4 倍のメモリを使用するようになります。さらに 2 回ズームインすると、オブジェクトの描画サイズは 800 x 800 ピクセルになり、元の 100 x 100 ピクセルのオブジェクトの 64 倍のメモリを使用します。SWF ファイルでフィルタを使用する場合、SWF ファイルのコンテキストメニューのズームオプションを無効に設定することをお勧めします。

無効なパラメータタイプを使用している場合にも、エラーメッセージが表示されます。一部のフィルタパラメータには、値の有効範囲が設定されています。有効範囲外の値を指定すると、指定した値が、範囲内の有効な値に変わります。たとえば、quality の値は、標準の操作では、1～3 であり、0～15 の範囲外には設定できません。15 を超える値を指定すると、15 に設定されます。

また、コンストラクタの中には、入力パラメータとして必要な配列の長さに制限があるものがあります。畳み込みフィルタまたはカラーマトリックスフィルタを無効な配列 (正しいサイズではない) を使用して作成すると、コンストラクタが失敗し、フィルタは作成されません。このフィルタオブジェクトがムービークリップのフィルタ配列のエントリとして使用されると、無視されます。

ヒント	ぼかしフィルタを使用する場合、blurX と blurY の値に 2 の乗数 (2、4、8、16、32 など) を使用すると、計算処理が速くなり、パフォーマンスが 20 ～ 30% 向上します。

ActionScript によるフィルタの操作

flash.filters パッケージには、Flash Player 8 以降でビットマップフィルタ効果のあるクラスがあります。フィルタを使用することによって、ぼかし、ベベル、グロー、ドロップシャドウなど、豊富な視覚効果を、テキスト、ムービークリップおよびボタンインスタンスに適用できます。Flash のオーサリングツールを使用して、テキストやイメージ、ビデオなどのオブジェクトにもフィルタ効果を適用できます。Flash には 9 つのフィルタ効果がありますが、ユーザーインターフェイスから使用できるのは 7 つのフィルタのみです。ConvolutionFilter フィルタと DisplacementMapFilter フィルタを使用するには、ActionScript コードを使用しなければなりません。

メモ	ActionScript を使用すれば、すべてのフィルタが使用できます。

次の手順は、半透明の PNG イメージをロードし、そのイメージの不透明な部分に GlowFilter 効果を適用します。

半透明のイメージにフィルタを適用するには :

1. 新しい Flash ドキュメントを作成し、transparentImg.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.filters.GlowFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
   target_mc._x = (Stage.width - target_mc._width) / 2;
   target_mc._y = (Stage.height - target_mc._height) / 2;
   var glow:GlowFilter = new GlowFilter();
   target_mc.filters = [glow];
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/logo.png",
   img_mc);
```

   このコードは、ムービークリップのローダーインスタンスを使用して、半透明の PNG イメージをロードします。イメージのロードが完了すると、イメージがステージの中央に移動し、グローフィルタが適用されます。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   グローフィルタ効果は、PNG イメージの不透明領域のみに適用されます。

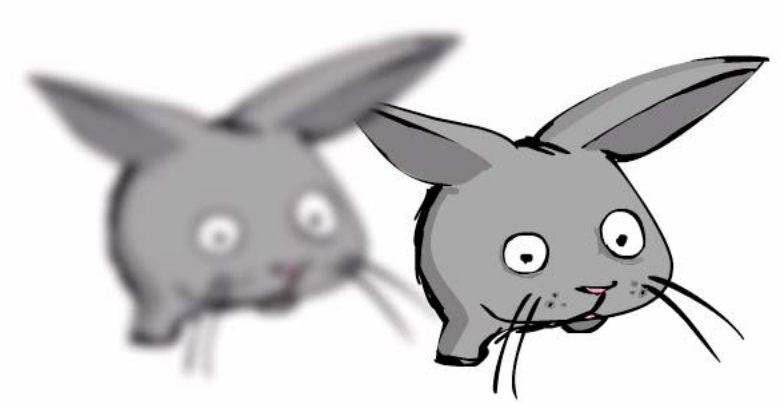
次のセクションでは、フィルタの使用方法を説明しています。

- 450 ページの「ぼかしフィルタの使用」
- 452 ページの「ドロップシャドウフィルタの使用」
- 456 ページの「グローフィルタの使用」
- 457 ページの「グラデーショングローの作成」
- 459 ページの「ベベルフィルタの使用」
- 465 ページの「グラデーションベベルフィルタの適用」
- 466 ページの「カラーマトリックスフィルタの使用」
- 468 ページの「畳み込みフィルタの使用」
- 470 ページの「置き換えマップフィルタの使用」

ActionScript を使用してフィルタを適用する例を示すサンプルのソースファイル "Filters.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Filters" フォルダに移動してサンプルにアクセスします。

ぼかしフィルタの使用

BlurFilter クラスを使用すると、Flash の各種オブジェクトにぼかし視覚効果を適用できます。ぼかし効果は、イメージの細部をぼかします。ソフトフォーカスがかかっているように見えるぼかしから、半透明ガラスを通してイメージを見るようにかすんで見えるガウスぼかしまで作成できます。ぼかしフィルタは、ボックスパスぼかしフィルタに基づいています。`quality` パラメータは、ぼかしの繰り返し回数を指定します。3 に設定すると、ガウスぼかしフィルタを近似したものになります。



メモ	ぼかしフィルタが拡大・縮小するのは、ステージをズームインする場合のみです。

このフィルタの詳細については、『ActionScript 2.0 リファレンスガイド』の「BlurFilter (flash.filters.BlurFilter)」を参照してください。

次の手順は、ステージ上のマウスポインタの位置に応じて、動的にロードされたイメージにぼかしを適用します。ポインタがステージの中心から遠ざかるほど、イメージのぼかしが強くなります。

マウスポインタの位置に応じてイメージにぼかしを適用するには :

1. 新しい Flash ドキュメントを作成し、dynamicblur.fla という名前を付けて保存します。
2. タイムラインのフレーム 1 に次のコードを追加します。

```
import flash.filters.BlurFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  // ステージの中央に target_mc ムービークリップを配置
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
var blur:BlurFilter = new BlurFilter(10, 10, 2);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  /* ポインタをステージの中心に動かし、blurX および blurY プロパティを 0% に設定 */
  blur.blurX = Math.abs(_xmouse - (Stage.width / 2)) / Stage.width * 2 * 255;
  blur.blurY = Math.abs(_ymouse - (Stage.height / 2)) / Stage.height * 2 *
  255;
  img_mc.filters = [blur];
};
Mouse.addListener(mouseListener);
```

コードの最初のセクションは、イメージをロードし、動的にロードしたイメージをステージ上に配置します。2 番目のセクションは、マウスが移動するたびに呼び出されるリスナーを定義します。ステージ上のマウスポインタの位置に応じて、水平および垂直方向のぼかし量を計算します。ステージの中心から遠ざかるようにポインタを動かすと、インスタンスに適用されるぼかしが徐々に強くなります。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

   x 軸に沿ってマウスポインタを動かして、水平方向のぼかし量を調節します。ポインタがステージの中心から離れるほど、インスタンスのぼかし量が多くなります。y 軸に沿ってポインタを動かすと、ステージの中心からの距離に応じて、垂直方向のぼかし量が変化します。

ヒント	ぼかしフィルタを使用する場合、blurX と blurY の値に 2 の乗数 (2、4、8、16、32 など) を使用すると、計算処理が速くなり、パフォーマンスが 20 ～ 30% 向上します。

注意	1.03125 より小さい値を設定すると、ぼかし効果は無効になります。

ドロップシャドウフィルタの使用

DropShadowFilter クラスを使用すると、Flash の各種オブジェクトにドロップシャドウ (陰影) を追加できます。シャドウアルゴリズムは、ぼかしフィルタで使用するのと同じボックスフィルタに基づいています (450 ページの「ぼかしフィルタの使用」を参照)。ドロップシャドウのスタイルには複数のオプションがあり、内側シャドウ、外側シャドウ、ノックアウトモードなどがあります。

ドロップシャドウフィルタの詳細については、『ActionScript 2.0 リファレンスガイド』の「DropShadowFilter (flash.filters.DropShadowFilter)」を参照してください。

次の手順は、Drawing API を使用してステージ上に四角形を描画します。ステージ上で水平方向にマウスポインタを動かすと、四角形とドロップシャドウの間の距離が変わり、垂直方向にマウスポインタを動かすと、ドロップシャドウのぼかし量が変わります。

ドロップシャドウフィルタを使用するには :

1. 新しい Flash ドキュメントを作成し、dropshadow.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

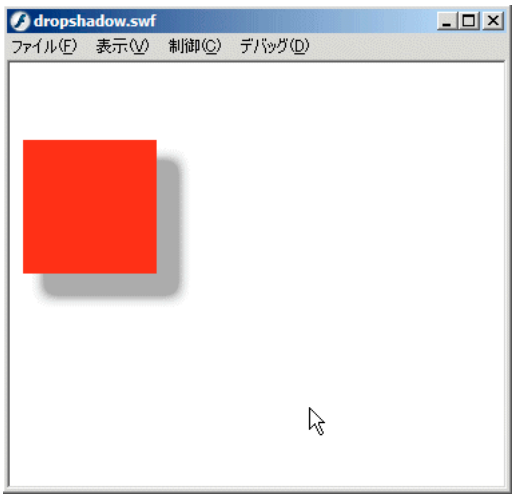
```
// フィルタクラスを読み込む
import flash.filters.DropShadowFilter;
// ムービークリップ shapeClip を作成
this.createEmptyMovieClip("shapeClip", 1);
// Drawing API を使用してシェイプを描画
with (shapeClip) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}
// シェイプを配置
shapeClip._x = 100;
shapeClip._y = 100;
```

```
// 四角形のクリックでシャドウの強さを増す
shapeClip.onPress = function():Void {
   dropShadow.strength++;
   shapeClip.filters = [dropShadow];
};
// フィルタを作成
var dropShadow:DropShadowFilter = new DropShadowFilter(4, 45, 0x000000, 0.4,
   10, 10, 2, 3);

var mouseListener:Object = new Object();
// マウスの動きに応じてフィルタを制御するリスナーを作成および適用
mouseListener.onMouseMove = function():Void {
   dropShadow.distance = (_xmouse / Stage.width) * 50 - 20;
   dropShadow.blurX = (_ymouse / Stage.height) * 10;
   dropShadow.blurY = dropShadow.blurX;
   shapeClip.filters = [dropShadow];
};
Mouse.addListener(mouseListener);
```

コードの最初のセクションは、ムービークリップを新規に作成し、Drawing API を使用して赤色の四角形を描画します。2 番目のセクションは、マウスが移動するたびに呼び出されるマウスリスナーを定義します。マウスリスナーは、マウスポインタの現在の x 位置と y 位置を基にドロップシャドウの距離とぼかしのレベルを計算し、ぼかしフィルタを再適用します。赤色の四角形をクリックすると、ドロップシャドウの強さが増します。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

マウスポインタを *x* 軸に沿って動かすと、ムービークリップインスタンスに適用されるドロップシャドウの距離が変わり、*y* 軸に沿って動かすと、ぼかし量が変わります。



動的にロードしたイメージにもドロップシャドウを作成して、適用できます。次の手順では、外部イメージをロードし、マウスポインタの動きを追うように現れるドロップシャドウを適用します。イメージの左上隅から遠ざかるようにポインタを動かすと、イメージに適用される水平方向と垂直方向のぼかしが徐々に大きくなります。

マウスポインタを追うドロップシャドウを作成するには :

1. 新しい Flash ドキュメントを作成し、dropshadowmouse.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.filters.DropShadowFilter;
System.security.allowDomain("http://www.helpexamples.com");
var dropShadow:DropShadowFilter = new DropShadowFilter(4, 45, 0x000000, 0.8,
  10, 10, 2, 2);
// ステージにイメージをロードして配置
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);

// マウスの動きに応じてドロップシャドウの位置を再計算
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  var p1:Number = img_mc._y - _ymouse;
  var p2:Number = img_mc._x - _xmouse;
  var degrees:Number = Math.atan2(p1, p2) / (Math.PI / 180);
  dropShadow.distance = Math.sqrt(Math.pow(p1, 2) + Math.pow(p2, 2)) * 0.5;
  dropShadow.blurX = dropShadow.distance;
  dropShadow.blurY = dropShadow.blurX;
  dropShadow.angle = degrees - 180;
  img_mc.filters = [dropShadow];
};
Mouse.addListener(mouseListener);
```

コードの最初のセクションは、ドロップシャドウインスタンスを定義し、外部イメージをロードし、ロードしたイメージをステージの中央に再配置します。2 番目のセクションは、マウスポインタがステージで移動するたびに呼び出されるマウスリスナーを定義します。マウスが移動するたびに、イベントハンドラが、マウスポインタの位置とイメージの左上隅の間の距離を再計算します。この計算結果に基づいて、ドロップシャドウフィルタがムービークリップに再適用されます。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

   SWF ファイルを実行すると、マウスポインタの動きを追うようにドロップシャドウが現れます。ステージ上で、イメージの左上隅にマウスポインタを近づけると、イメージに適用されるぼかし効果が小さくなります。イメージの左上隅からマウスポインタを遠ざけると、ドロップシャドウ効果がより鮮明になります。

動的にロードした半透明の PNG イメージにもドロップシャドウを適用できます。次の手順では、ドロップシャドウフィルタは、PNG イメージの不透明領域のみに適用されます。

ドロップシャドウを半透明のイメージに適用するには :

1. 新しい Flash ドキュメントを作成し、dropshadowTransparent.fla という名前を付けて保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.filters.DropShadowFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  var dropShadow:DropShadowFilter = new DropShadowFilter(4, 45, 0x000000,
  0.5, 10, 10, 2, 3);
  target_mc.filters = [dropShadow];
};
mclListener.onLoadError = function(target_mc:MovieClip):Void {
  trace("unable to load image.");
};
this.createEmptyMovieClip("logo_mc", 10);
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mclListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/logo.png",
  logo_mc);
```

   この ActionScript コードは、MovieClipLoader クラスを使用してイメージをロードし、リモートサーバーからのロードが完了したときに、イメージにドロップシャドウフィルタを適用します。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

   透明な背景の PNG イメージがロードされます。ドロップシャドウフィルタを適用すると、イメージの不透明領域のみにフィルタが適用されます。

グローフィルタの使用

GlowFilter クラスを使用すると、Flash の各種オブジェクトにグロー効果を追加できます。グローアルゴリズムは、ぼかしフィルタで使用するのと同じボックスフィルタに基づいています (450 ページの「ぼかしフィルタの使用」を参照)。グローには複数のスタイルがあり、内側グロー、外側グロー、ノックアウトモードなどがあります。グローフィルタは、ドロップシャドウの distance プロパティと angle プロパティを 0 に設定した効果と似ています。

グローフィルタの詳細については、『ActionScript 2.0 リファレンスガイド』の「GlowFilter (flash.filters.GlowFilter)」を参照してください。

次の手順は、ステージ上にある動的に作成されたムービークリップにグローフィルタを適用する方法を示しています。ステージでマウスポインタを動かすと、ムービークリップのぼかしが変化し、動的に作成されたシェイプをクリックすると、フィルタの強さが増します。

グローフィルタを使用するには :

1. 新しい Flash ドキュメントを作成し、glowfilter.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript コードを追加します。

```
import flash.filters.GlowFilter;

this.createEmptyMovieClip("shapeClip", 10);
with (shapeClip) {
   beginFill(0xFF0000, 100);
   moveTo(0, 0);
   lineTo(100, 0);
   lineTo(100, 100);
   lineTo(0, 100);
   lineTo(0, 0);
   endFill();
}
shapeClip._x = 100;
shapeClip._y = 100;
shapeClip.onPress = function():Void {
   glow.strength++;
   shapeClip.filters = [glow];
};
var glow:GlowFilter = new GlowFilter(0xCC0000, 0.5, 10, 10, 2, 3);
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
   glow.blurX = (_xmouse / Stage.width) * 255;
   glow.blurY = (_ymouse / Stage.width) * 255;
   shapeClip.filters = [glow];
};
Mouse.addListener(mouseListener);
```

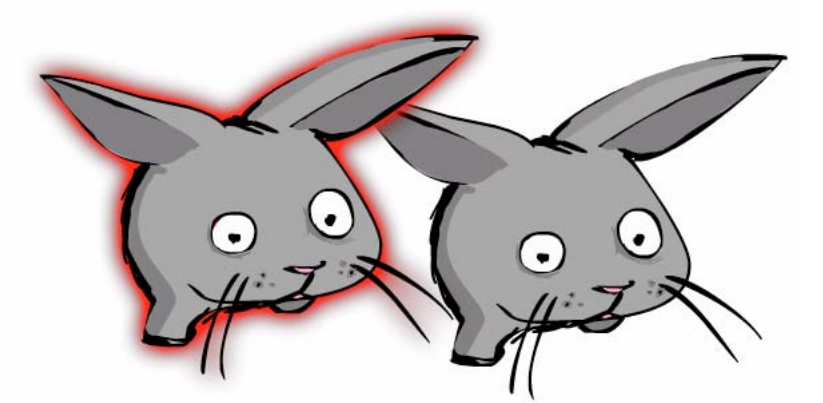
このコードは、Drawing API を使用してステージ上に四角形を描画し、描画したシェイプにグローフィルタを適用します。*x* 軸と *y* 軸に沿ってマウスポインタを動かすたびに、グローフィルタのぼかし量が計算され、シェイプに適用されます。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   水平および垂直方向のぼかし量は、マウスポインタの現在の `_xmouse` 位置および `_ymouse` 位置を基に計算されます。ステージの左上隅に向かってマウスポインタを動かすと、水平および垂直方向のぼかし量が減少します。逆に、ステージの右下隅に向かってマウスポインタを動かすと、水平および垂直方向のぼかし量が増加します。

グラデーショングローの作成

GradientGlowFilter クラスを使用すると、グラデーショングロー効果を Flash の各種オブジェクトに適用できます。グラデーショングローは本物のように見える輝きで、カラーグラデーションを指定できます。グラデーショングローは、オブジェクトの内側エッジや外側エッジの周囲、またはオブジェクトの上に適用できます。



このフィルタの詳細については、『ActionScript 2.0 リファレンスガイド』の「GradientBevelFilter (flash.filters.GradientBevelFilter)」を参照してください。

次の手順は、Drawing API を使用してステージ上に四角形を描画し、描画したシェイプにグラデーショングローフィルタを適用します。ステージ上の四角形をクリックすると、フィルタの強さが増し、水平または垂直方向にマウスポインタを動かすと、*x* 軸と *y* 軸に沿ってぼかし量が変わります。

グラデーショングローフィルタを適用するには :

1. 新しい Flash ドキュメントを作成し、gradientglow.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.filters.GradientGlowFilter;
// 新しい shapeClip インスタンスを作成
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 10);
// Drawing API を使用してシェイプを作成
with (shapeClip) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}

// シェイプを配置
shapeClip._x = 100;
shapeClip._y = 100;
// グラデーショングローを定義
var gradientGlow:GradientGlowFilter = new GradientGlowFilter(0, 45,
  [0x000000, 0xFF0000], [0, 1], [0, 255], 10, 10, 2, 3, "outer");

// マウスリスナーを定義、2 つのイベントをリッスン
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function():Void {
  gradientGlow.strength++;
  shapeClip.filters = [gradientGlow];
};
mouseListener.onMouseMove = function():Void {
  gradientGlow.blurX = (_xmouse / Stage.width) * 255;
  gradientGlow.blurY = (_ymouse / Stage.height) * 255;
  shapeClip.filters = [gradientGlow];
};
Mouse.addListener(mouseListener);
```

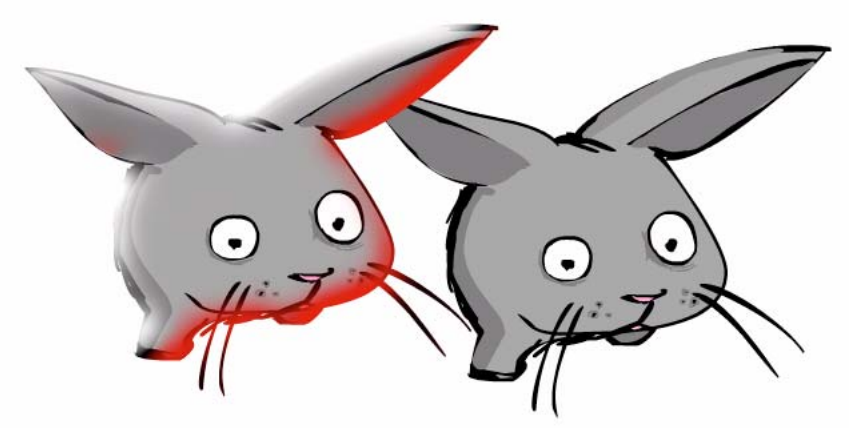
このコードは、3 つのセクションに分かれています。コードの最初のセクションは、Drawing API を使用して四角形を作成し、ステージ上に配置します。2 番目のセクションは、赤色から黒色へのグロー作成するように、グラデーショングローフィルタインスタンスを新規に定義します。最後のセクションは、2 つのマウスイベントハンドラをリッスンするマウスリスナーを定義します。イベントハンドラの 1 つは onMouseDown で、グラデーショングローの強さを増します。もう 1 つのイベントハンドラは onMouseMove で、SWF ファイル内でマウスポインタが移動するたびに呼び出されます。Flash ドキュメントの左上隅からマウスポインタを遠ざけると、グロー効果がより鮮明になります。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   ステージでマウスポインタを動かすと、グラデーショングローフィルタのぼかしの強さが増加または減少します。マウスの左ボタンをクリックすると、グローの強さが増します。

ベベルフィルタの使用

BevelFilter クラスを使用すると、Flash の各種オブジェクトにベベル効果を追加できます。オブジェクトにベベル効果を適用すると 3 次元的に表現されます。異なるハイライトカラー、シャドウカラー、ベベルのぼかし量、ベベルの角度、ベベルの配置、ノックアウト効果を使って、ベベルの外観をカスタマイズできます。



このフィルタの詳細については、『ActionScript 2.0 リファレンスガイド』の「BevelFilter (flash.filters.BevelFilter)」を参照してください。

次の手順は、Drawing API を使用して四角形を作成し、作成したシェイプにベベルを追加します。

ベベルフィルタを使用するには :

1. 新しい Flash ドキュメントを作成し、bevel.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.filters.BevelFilter;
// ベベルフィルタを定義
var bevel:BevelFilter = new BevelFilter(4, 45, 0xFFFFFF, 1, 0xCC0000, 1, 10,
   10, 2, 3);
// 新しい shapeClip インスタンスを作成
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 1);
// Drawing API を使用してシェイプを作成
with (shapeClip) {
   beginFill(0xFF0000, 100);
   moveTo(0, 0);
```

```
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}
// シェイプをステージに配置
shapeClip._x = 100;
shapeClip._y = 100;
// マウスクリックで強さを増す
shapeClip.onPress = function():Void {
  bevel.strength += 2;
  shapeClip.filters = [bevel];
};

// マウスの移動時にフィルタを変更するリスナーを定義
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  bevel.distance = (_xmouse / Stage.width) * 10;
  bevel.blurX = (_ymouse / Stage.height) * 10;
  bevel.blurY = bevel.blurX;
  shapeClip.filters = [bevel];
};
Mouse.addListener(mouseListener);
```

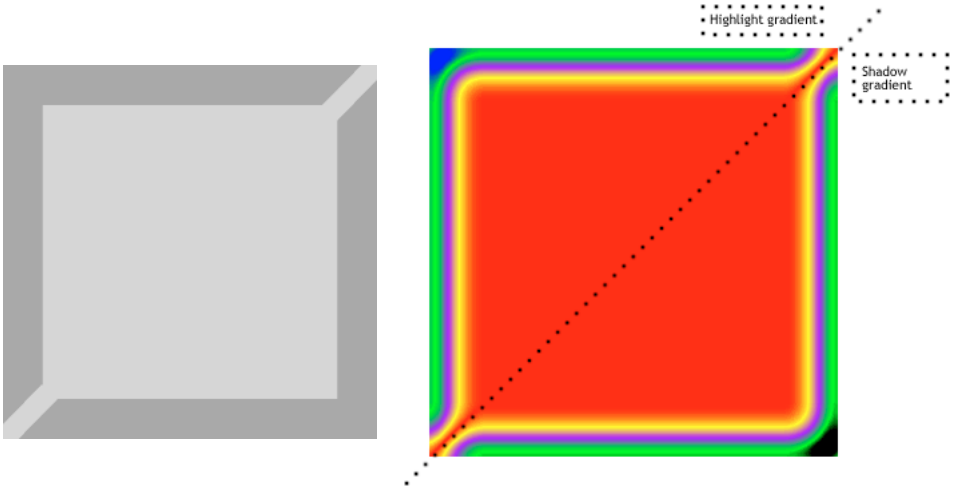
コードの最初のセクションは、BevelFilter インスタンスを定義し、Drawing API を使用してステージ上に四角形を描画します。ステージの四角形をクリックすると、ベベルの斜角面が長くなり、シャープなベベルになります。2 番目のセクションは、マウスポインタの現在位置に基づいてベベルの距離とぼかしを変更するように、マウスリスナーを定義します。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

   x 軸に沿ってマウスポインタを動かすと、ベベルのオフセット距離が増加または減少します。*y* 軸に沿ってマウスポインタを動かすと、ポインタの *y* 位置に基づいて水平および垂直方向のぼかし量が変わります。

グラデーションベベルフィルタについて

グラデーションベベルは、矩形などのオブジェクトに適用され、2 つの斜めのエッジ (" ハイライト " と " シャドウ ") と " ベースフィル " の 3 つの部分に分けて、グラデーションカラーを表現します。次の図の矩形には、内側のベベルが設定されています。左の矩形の濃いグレーの領域がベベルのエッジで、明るいグレーの領域がベースフィルです。右の矩形には、レインボウのグラデーションベベルが、各エッジに 4 色ずつのベベルで適用されています。



グラデーションベベルフィルタでは、複数のプロパティによって、フィルタを適用する方法を制御します。グラデーションベベルのカラーは、カラーの配列で設定します。矩形の各部分における実際の色分布は、比率の配列によって決まります。distance プロパティは、オフセット距離、つまり、オブジェクトから何ピクセル離れた位置にベベルエッジを適用するかを決定します。`blurX` と `blurY` のプロパティは、ベベルのカラーのシャープネスを制御します。値が高いほど、幅の広い、ソフトなベベルになり、値が低いほど、細いシャープなベベルになります。angle プロパティは、オブジェクトに光を当て、エッジにハイライトとシャドウの効果を生み出す理論上の光源を設定します。`strength` プロパティは、カラーの広がりを制御します。その値が低いと、左図に示すように、カラーが発色しません。その値が高いと、配列の外側のカラーが強く、配列の中央のカラーが弱くなります。`knockout` と `type` プロパティは、ベベルフィルタをオブジェクト全体に適用する方法と場所 (オブジェクトをノックアウトするかどうか、またどこで適用するか) を指定します。

グラデーションベベルフィルタを理解する際にわかりにくい概念の1つは、色分布です。グラデーションベベルでのカラーがどのように分布するかを理解するには、グラデーションベベルで表現したい色をまず第一に考えます。単純なベベルには、ハイライトカラーとシャドウカラーの2つの概念しかありません。グラデーションベベルについても、同じ考え方、つまり、ハイライトグラデーションとシャドウグラデーションの2つの概念で理解できます。ハイライトは左上隅に現れ、シャドウは右下隅に現れます。ハイライトに4色、シャドウに4色のカラーがあります。これらのカラーに加えて、ハイライトとシャドウのエッジが接する場所のカラーとなるベースフィルカラーがあります。つまり、図のように、カラーの配列には9つのカラーが並びます。

カラーの配列に含まれるカラーの数と、アルファ値の配列および比率の配列の要素の数は一致します。カラーの配列の最初の項目は、アルファ値の配列と比率の配列の最初の要素に対応し、2番目以降の項目も同様に、それぞれ順に対応します。9つのカラーがあるので、アルファ値の配列に9つ、比率の配列に9つの値があります。アルファ値はカラーの透明度を指定する値です。

比率の配列の有効な値の範囲は、0～255ピクセルです。範囲内のちょうど中間の値が128で、これがベースフィルの値です。ベベルで希望どおりの効果を得るには、ほとんどの場合、次のように比率の値を割り当てることをお勧めします。9つのカラーの例で説明します。

- 最初の4つのカラーは、0～127の範囲の値にし、値を段階的に増やして、次の値が前の値より大きいか等しくなるようにします。この部分が1つ目のベベルエッジ、つまり、ハイライトです。
- 5番目の色(中間色)はベース塗りで、128に設定します。ピクセル値の128によりベース塗りが設定されます。このベース塗りは、typeがouterに設定されている場合はシェイプの外側(およびベベルエッジの周囲)に現れ、typeがinnerに設定されている場合はシェイプの内側に現れてオブジェクト自体の塗りを効果的に隠します。
- 次の4つのカラーは、129～255の範囲の値にし、やはり値を段階的に増やして、次の値が前の値より大きいか等しくなるようにします。この部分が2つ目のベベルエッジ、つまり、シャドウです。

さまざまなカラーがブレンドするストライプ模様のグラデーションを作成する場合は、比率の各値が、そのカラーのピクセル数を設定し、グラデーションでのそのカラーのストライプの幅を決定します。両方のエッジでカラーが均等に分布するようにしたい場合は、次のようにします。

- 使用するカラーの数を奇数にし、中央のカラーをベースフィルにします。
- 値が0～127と129～255の間で均等に分布するように設定します。
- 値を調節して、グラデーションでカラーのストライプの幅を変更します。

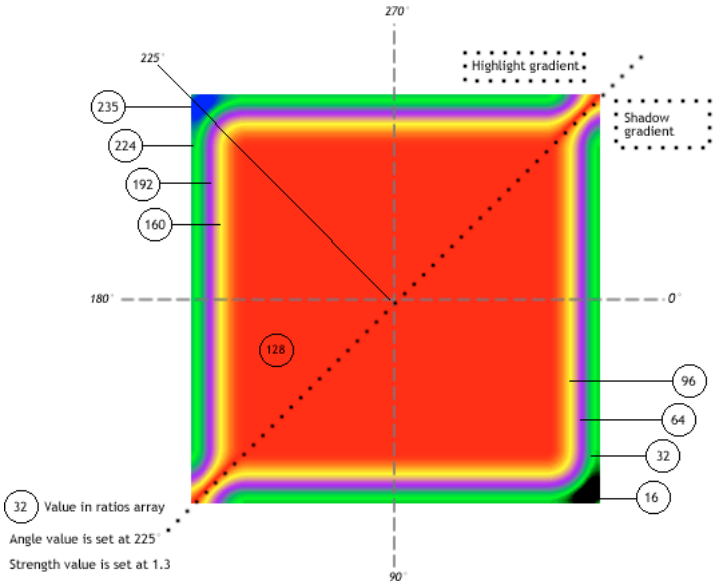
メモ	角度の値によって、どちらのエッジがハイライトまたはシャドウになるかを設定します。

角度の値は、グラデーションカラーがオブジェクトに適用される角度で、オブジェクト上でのハイライトとシャドウの位置も示します。カラーは、配列と同じ順序で適用されます。

次のコードでは、ピンク色の四角形 (Drawing API を使用して作成) にレインボウのグラデーションフィルタを適用します。カラーは、配列と同じ順序に並び、ハイライトの青、緑、紫および黄、ベースフィルの赤、シャドウの黄、紫、緑、黒の順となります。比率の値には、ハイライトのカラーを 0 ～ 127、シャドウのカラーを 129 ～ 255 の範囲内で割り当てます。最も外側のエッジのカラーは、青 (16) と黒 (235) に設定します。

```
var colors:Array = [0x0000FF, 0x00FF00, 0x9900FF, 0xFFFF00, 0xFF0000, 0xFFFF00,
  0x9900FF, 0x00FF00,0x000000];
var alphas:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var ratios:Array = [16, 32, 64, 96, 128, 160, 192, 224, 235];
var gradientBevel:GradientBevelFilter = new GradientBevelFilter(8, 225, colors,
  alphas, ratios, 16, 16, 1.3, 2, "inner", false);
```

下の図は、このコードで作成されたグラデーションベベルフィルタを示します。9 色のレインボウのベベルが赤色の矩形のムービークリップに適用されています。



破線は、決定された角度を示します。図では、225 度という角度が実際のフィルタでどのように適用されるか、および各カラーの比率がわかります。角度を 225 度に設定することによって、配列の最初のカラーが 225 度の位置 (左上のハイライトの位置) に現れます。破線は、ハイライトグラデーションとシャドウグラデーションがそれぞれ適用される領域の境界線を示します。

元のムービークリップのカラーはピンクですが、128 の値を赤に設定することによって、128 ピクセルの値がベースフィルとなり、元のムービークリップを塗りつぶします。しかし、適切にフィルタを設定している場合、フィルタによって元のオブジェクトは変更されません。filters プロパティの選択を解除すると、元のムービークリップのカラーが戻ります。

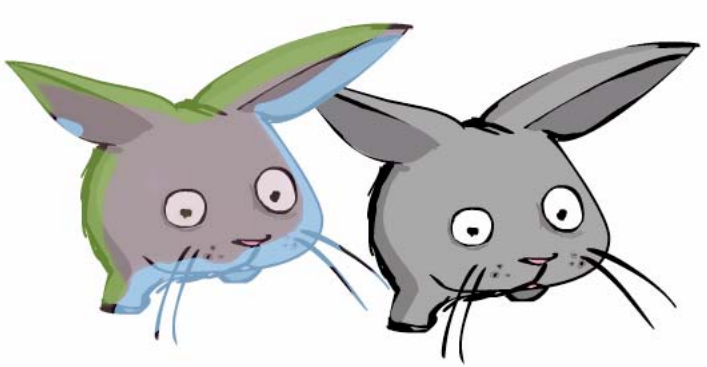
すべてのフィルタのプロパティは相互に影響し合います。1 つのプロパティを変更して、適用する効果を変えたら、他のプロパティも同様に変更しなければならない場合があります。

この図を作成する ActionScript コード全体は、次のとおりです。

```
import flash.filters.GradientBevelFilter;

// 塗りつぶされた四角形シェイプを描画
this.createEmptyMovieClip("square_mc", this.getNextHighestDepth());
     square_mc.beginFill(0xFF99CC);
     square_mc.moveTo(40, 40);
     square_mc.lineTo(200, 40);
     square_mc.lineTo(200, 200);
     square_mc.lineTo(40, 200);
     square_mc.lineTo(40, 40);
     square_mc.endFill();

/* GradientBevelFilter(distance:Number, angle:Number, colors:Array,
   alphas:Array, ratios:Array, blurX:Number, blurY:Number, strength:Number,
   quality:Number, type:String, knockout:Boolean) */

// カラー、アルファ値、比率の配列を作成
var colors:Array = [0x0000FF, 0x00FF00, 0x9900FF, 0xFFFF00, 0xFF0000, 0xFFFF00,
   0x9900FF, 0x00FF00,0x000000];// 青、緑、紫、黄、赤、黄、紫、緑、黒
var alphas:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var ratios:Array = [16, 32, 64, 96, 128, 160, 192, 224, 235];

// フィルタオブジェクトを作成
var gradientBevel:GradientBevelFilter = new GradientBevelFilter(8, 225, colors,
   alphas, ratios, 16, 16, 1.3, 2, "inner", false);

// 四角形のムービークリップにフィルタを適用
square_mc.filters = [gradientBevel];
```

グラデーションベベルフィルタの適用

GradientBevelFilter クラスを使用すると、Flash でオブジェクトにグラデーションベベル効果を適用できます。グラデーションベベルは、オブジェクトの外側、内側、または上側が斜めになったエッジであり、グラデーションカラーで強調されます。斜めのエッジによってオブジェクトが 3 次元的に見えるようになり、次の図のようなカラフルな結果が得られます。



このフィルタの詳細については、『ActionScript 2.0 リファレンスガイド』の「GradientBevelFilter (flash.filters.GradientBevelFilter)」を参照してください。

次の手順は、Drawing API を使用してステージ上に四角形を描画し、描画したシェイプにグラデーションベベルフィルタを適用します。

グラデーションベベルフィルタを使用するには :

1. 新しい Flash ドキュメントを作成し、gradientbevel.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.filters.GradientBevelFilter;
var shapeClip:MovieClip = this.createEmptyMovieClip("shape_mc", 1);
with (shapeClip) {
   beginFill(0xFF0000, 100);
   moveTo(0, 0);
   lineTo(200, 0);
   lineTo(200, 200);
   lineTo(0, 200);
   lineTo(0, 0);
   endFill();
}
```

```
shapeClip._x = (Stage.width - shapeClip._width) / 2;
shapeClip._y = (Stage.height - shapeClip._height) / 2;
var colors:Array = new Array(0xFFFFFF, 0xCCCCCC, 0x000000);
var alphas:Array = new Array(1, 0, 1);
var ratios:Array = new Array(0, 128, 255);
var gradientBevel:GradientBevelFilter = new GradientBevelFilter(10, 45,
  colors, alphas, ratios, 4, 4, 5, 3);
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  gradientBevel.strength++;
  shapeClip.filters = [gradientBevel];
};
mouseListener.onMouseMove = function() {
  gradientBevel.blurX = (_xmouse / Stage.width) * 255;
  gradientBevel.blurY = (_ymouse / Stage.height) * 255;
  shapeClip.filters = [gradientBevel];
};
Mouse.addListener(mouseListener);
```

このコードは、Drawing API を使用してステージ上に四角形を描画し、ステージの中央に配置します。ステージでマウスポインタを動かすと、*x* 軸と *y* 軸に沿ってぼかし量が増減します。ステージの左に向かってマウスポインタを動かすと、水平方向のぼかし量が減少します。ステージの右に向かってマウスポインタを動かすと、ぼかし量が増加します。同様に、ステージの上に向かってポインタを動かすと、*y* 軸に沿ってぼかし量が減少します。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストし、結果を確認します。

カラーマトリックスフィルタの使用

ColorMatrixFilter クラスを使用すると、入力イメージの各ピクセルの ARGB カラー値とアルファ値に 4 x 5 マトリックス変換を適用することで、新しい ARGB カラー値とアルファ値から成る結果を作成できます。このフィルタで、色相 (明暗の度合い) の回転、彩度 (特定の色相の鮮やかさ) の変更、輝度 (明度またはカラーの強さ) のアルファ変換などのさまざまな効果を実現できます。さらに、これらのフィルタをアニメーション化し、アプリケーションで効果を作成できます。

メモ	カラーマトリックスフィルタは、ビットマップとムービークリップインスタンスに適用できます。

カラーマトリックスフィルタの詳細については、『ActionScript 2.0 リファレンスガイド』の「ColorMatrixFilter (flash.filters.ColorMatrixFilter)」を参照してください。

カラーマトリックスフィルタを使用して、次のように、インスタンスの明度を変更できます。

ムービークリップの明度を高くするには :

1. 新しい Flash ドキュメントを作成し、brightness.fla という名前を付けて保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.filters.ColorMatrixFilter;
System.security.allowDomain("http://www.helpexamples.com/");
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip):Void {
   var myElements_array:Array = [1, 0, 0, 0, 100,
        0, 1, 0, 0, 100,
        0, 0, 1, 0, 100,
        0, 0, 0, 1, 0];
   var myColorMatrix_filter:ColorMatrixFilter = new
   ColorMatrixFilter(myElements_array);
   target_mc.filters = [myColorMatrix_filter];
}
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image2.jpg",
   img_mc);
```

   このコードは、MovieClipLoader インスタンスを使用して、JPEG イメージを動的にロードします。ロードが完了し、イメージをステージの中央に配置した後で、カラーマトリックスフィルタを使用して、インスタンスの明度を 100% に設定します。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

次の手順のように、Tween クラスと ColorMatrixFilter クラスを組み合わせて、アニメーション化された明度効果を作成することもできます。

Tween クラスを使用してインスタンスの明度レベルをアニメーション化するには :

1. 新しい Flash ドキュメントを作成し、brightnesstween.fla という名前を付けて保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.filters.ColorMatrixFilter;
import mx.transitions.Tween;
import mx.transitions.easing.*;
System.security.allowDomain("http://www.helpexamples.com");
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
   // ステージの中央にムービークリップを配置
   target_mc._x = (Stage.width - target_mc._width) / 2;
   target_mc._y = (Stage.height - target_mc._height) / 2;
   target_mc.watch("brightness", brightnessWatcher, target_mc);
   // target_mc ムービークリップの明度を -100 〜 +100 の間でアニメーション化
   var myTween:Tween = new Tween(target_mc, "brightness", Elastic.easeOut,
   100, -100, 3, true);
   myTween.onMotionFinished = function() {
```

```
      this.yoyo();
    };
  };
  this.createEmptyMovieClip("img_mc", 10);
  var img_mcl:MovieClipLoader = new MovieClipLoader();
  img_mcl.addListener(mclListener);
  img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
    img_mc);

  function brightnessWatcher(prop:String, oldVal:Number, newVal:Number,
    target_mc:MovieClip):Number {
    var brightness_array:Array = [1, 0, 0, 0, newVal,
         0, 1, 0, 0, newVal,
         0, 0, 1, 0, newVal,
         0, 0, 0, 1, 0];
    target_mc.filters = [new ColorMatrixFilter(brightness_array)];
    return newVal;
  };
```

このコードの最初のセクションは、MovieClipLoader クラスを使用して、ステージに JPEG イメージをロードします。ロードが完了したら、ステージの中央にイメージを再配置します。次に、Tween クラスを使用してイメージの明度レベルをアニメーション化します。明度をアニメーション化するには、`Object.watch()` メソッドを使用して、ActionScript オブジェクトの指定されたプロパティの変更に応じて呼び出されるイベントハンドラを登録します。ActionScript で、`target_mc` インスタンスのカスタム明度プロパティを設定する試みがあると、`brightnessWatcher` 関数を呼び出します。カスタムの `brightnessWatcher` 関数は新しい配列を作成し、その配列によって、ターゲットイメージの明度を指定されたレベルに設定するカラーマトリックスフィルタが使用されます。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   イメージのロードが完了し、ステージに配置すると、イメージの明度が、-100 ～ 100 の間でアニメーションのように変化します。明度トゥイーンが完了すると、`Tween.yoyo()` メソッドが呼び出されてアニメーションが逆に再生され、トゥイーンはアニメーションを再生し続けます。

畳み込みフィルタの使用

ConvolutionFilter クラスを使用すると、マトリックス畳み込みフィルタ効果を適用できます。畳み込みでは、選択したソースイメージ内のピクセルを隣接するピクセルと組み合わせて、イメージを作成します。畳み込みフィルタを使用すると、ぼかし、エッジ検出、シャープ、エンボス、ベベルなど、幅広いイメージ操作を実現できます。

メモ	このフィルタは、ビットマップとムービークリップインスタンスに適用できます。

マトリックスの畳み込みは、*n* x *m* マトリックスに基づいています。入力イメージ内の特定のピクセルを隣接するピクセルと組み合わせることによって、新しいピクセル値を生成します。結果として得られるピクセルは、対応するソースピクセルおよび連接するピクセルにマトリックスを適用することで算出されます。

このフィルタは ActionScript でのみ使用できます。このフィルタの詳細については、『ActionScript 2.0 リファレンスガイド』の「ConvolutionFilter (flash.filters.ConvolutionFilter)」を参照してください。

次の手順では、動的にロードする JPEG イメージに畳み込みフィルタを適用します。

畳み込みフィルタを使用してイメージのカラーを変更するには :

1. 新しい Flash ドキュメントを作成し、convolution.fla という名前を付けて保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

this.createEmptyMovieClip("shape_mc", 1);
shape_mc.createEmptyMovieClip("holder_mc", 1);
var imageLoader:MovieClipLoader = new MovieClipLoader();
imageLoader.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  shape_mc.holder_mc);
var matrixArr:Array = [1, 4, 6, 4, 1, 4, 16, 24, 16, 4, 16, 6, 24, 36, 24, 6,
  4, 16, 24, 16, 4, 1, 4, 6, 4, 1];
var convolution:ConvolutionFilter = new ConvolutionFilter(5, 5, matrixArr);
shape_mc.filters = [convolution];

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  convolution.divisor = (_xmouse / Stage.width) * 271;
  convolution.bias = (_ymouse / Stage.height) * 100;
  shape_mc.filters = [convolution];
};
Mouse.addListener(mouseListener);
```

このコードは、3 つのセクションに分かれています。最初のセクションは、ConvolutionFilter と BitmapData の 2 つのクラスを読み込みます。2 番目のセクションは、ネストしたムービークリップを作成し、ムービークリップローダーオブジェクトを使用して、作成したムービークリップにイメージをロードします。畳み込みフィルタオブジェクトが作成され、shape_mc ムービークリップに適用されます。最後のセクションは、マウスポインタの現在位置に基づいて、畳み込みフィルタの divisor プロパティと bias プロパティを変更するマウスリスナーオブジェクトを定義し、畳み込みフィルタを shape_mc ムービークリップに再適用します。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

マウスポインタをステージの *x* 軸に沿って動かすと、フィルタの除数が変わり、*y* 軸に沿って動かすと、フィルタのバイアスが変わります。

置き換えマップフィルタの使用

DisplacementMapFilter クラスは、指定された BitmapData オブジェクト (置き換えマップイメージ) のピクセル値を使用して、ムービークリップインスタンスやビットマップデータインスタンスなど、ステージ上のインスタンスの置き換えを実行します。このフィルタを使用すると、指定したインスタンスでワープ効果や斑点効果を実現できます。

このフィルタは ActionScript でのみ使用できます。このフィルタの詳細については、『ActionScript 2.0 リファレンスガイド』の「DisplacementMapFilter (flash.filters.DisplacementMapFilter)」を参照してください。

次の手順では、JPEG イメージをロードし、イメージが歪んでいるような効果を実現する置き換えマップフィルタを適用します。マウスが移動するたびに、置き換えマップが再生成されます。

置き換えマップフィルタを使用してイメージを歪曲するには :

1. 新しい Flash ドキュメントを作成し、displacement.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.filters.DisplacementMapFilter;
import flash.geom.Point;
import flash.display.BitmapData;

var perlinBmp:BitmapData;
var displacementMap:DisplacementMapFilter;
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void  {
     target_mc._x = (Stage.width - target_mc._width) / 2;
     target_mc._y = (Stage.height - target_mc._height) / 2;
     perlinBmp = new BitmapData(target_mc._width, target_mc._height);
     perlinBmp.perlinNoise(target_mc._width, target_mc._height, 10,
  Math.round(Math.random() * 100000), false, true, 1, false);
     displacementMap = new DisplacementMapFilter(perlinBmp, new Point(0, 0),
  1, 1, 100, 100, "color");
     shapeClip.filters = [displacementMap];
};
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 1);
shapeClip.createEmptyMovieClip("holderClip", 1);
var imageLoader:MovieClipLoader = new MovieClipLoader();
imageLoader.addListener(mclListener);
imageLoader.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  shapeClip.holderClip);
```

```
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
   perlinBmp.perlinNoise(shapeClip._width, shapeClip._height, 10,
   Math.round(Math.random() * 100000), false, true, 1, false);
   shapeClip.filters = [displacementMap];
};
Mouse.addListener(mouseListener);
```

このコードは、JPEG イメージをロードし、ステージ上に配置します。ロードの完了後に、BitmapData インスタンスを作成し、`perlinNoise()` メソッドを使用して、作成したインスタンスにランダムに配置されたピクセルを含めます。BitmapData インスタンスは置き換えマップフィルタに渡され、フィルタがイメージに適用されて、歪曲効果が実現します。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

ステージ上でマウスポインタを動かすと、`perlinNoise()` メソッドが呼び出されて、置き換えマップが再作成されます。その結果、JPEG イメージの外観が変わります。

コードによるフィルタ効果の操作

Flash では、ステージ上のムービークリップ、テキストフィールド、およびボタンにさまざまなフィルタを動的に追加できます。再生中にフィルタを追加し操作することによって、マウスの動きやユーザーイベントに反応するリアルなシャドウ、ぼかし、グローを追加することができます。

コードによるフィルタの操作方法の例については、次のトピックを参照してください。

- 472 ページの「フィルタプロパティの調整」
- 473 ページの「ActionScript によるフィルタのアニメーション化」
- 474 ページの「clone() メソッドの使用」

フィルタプロパティの調整

オブジェクトに適用されたフィルタの配列には、標準の ActionScript 呼び出しで `MovieClip.filters` プロパティを使用することによって、アクセスできます。このプロセスでは、MovieClip に現在関連付けられている各フィルタオブジェクトが格納されている配列が返されます。各フィルタには、そのフィルタ固有のプロパティのセットがあります。フィルタは、配列オブジェクトのようにアクセスして、変更できますが、`filters` プロパティを使用して設定や取得を行うと、filters オブジェクトの参照ではなく、複製が返されます。

`filters` プロパティを設定すると、渡されるフィルタ配列の複製を作成しますが、参照として保存しません。フィルタを正しく取得すると、配列のコピーを返します。このアプローチの問題点は、次のようなコードが正常に動作しないことです。

```
// 正常に動作しない
my_mc.filters[0].blurX = 20;
```

前のコードでフィルタ配列のコピーが返されているので、コードによって元の配列は変更されず、配列のコピーが変更されます。blurX プロパティを適切に変更するには、次の ActionScript コードを使用する必要があります。

```
// 正常に動作
var filterArray:Array = my_mc.filters;
filterArray[0].blurX = 20;
my_mc.filters = filterArray;
```

次の手順では、ステージ上のマウスポインタの位置に応じて、イメージにぼかしを適用します。水平または垂直方向にマウスポインタを動かすと、その動きに応じて、ぼかしフィルタの `blurX` および `blurY` プロパティが調整されます。

ムービークリップのフィルタプロパティを調整するには :

1. 新しい Flash ドキュメントを作成し、adjustfilter.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.filters.BlurFilter;

this.createEmptyMovieClip("holder_mc", 10);
holder_mc.createEmptyMovieClip("img_mc", 20);
holder_mc.img_mc.loadMovie("http://www.helpexamples.com/flash/images/
   image2.jpg");
holder_mc.filters = [new BlurFilter(10, 10, 2)];
holder_mc._x = 75;
holder_mc._y = 75;

holder_mc.onMouseMove = function() {
   var tempFilter:BlurFilter = holder_mc.filters[0];
   tempFilter.blurX = Math.floor((_xmouse / Stage.width) * 255);
   tempFilter.blurY = Math.floor((_ymouse / Stage.height) * 255);
   holder_mc.filters = [tempFilter];
};
```

このコードは、3 つのセクションに分かれています。最初のセクションは、flash.filters.BlurFilter クラスを読み込み、完全修飾名を使用しなくても BlurFilter クラスを参照できるようにします。2 番目のセクションは、2 つのムービークリップを作成し、ネストしたクリップの 1 つにイメージをロードします。3 番目のセクションは、ステージ上のマウスポインタの動きに応じて、ぼかしを調整します。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

   x 軸に沿ってマウスポインタを動かして、フィルタの `blurX` プロパティを調整します。*y* 軸に沿ってマウスポインタを動かして、フィルタの `blurY` プロパティを調整します。ステージの左上隅に向かってマウスポインタを近づけると、ムービークリップに適用されるぼかし量が減少します。

ActionScript によるフィルタのアニメーション化

Tween クラスなどの ActionScript コードを使用することによって、実行時にフィルタをアニメーション化し、Flash アプリケーションにさまざまなアニメーション効果を適用できます。

次の例では、BlurFilter クラスと Tween クラスを組み合わせて、実行時にぼかしフィルタを 0 と 10 の間で変更するぼかしのアニメーションを作成しています。

Tween クラスを使用してぼかしをアニメーション化するには :

1. 新しい Flash ドキュメントを作成し、animatedfilter.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.filters.BlurFilter;
import mx.transitions.Tween;
import mx.transitions.easing.*;

this.createEmptyMovieClip("holder_mc", 10);
holder_mc.createEmptyMovieClip("img_mc", 20);

var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  var myTween:Tween = new Tween(target_mc, "blur", Strong.easeInOut, 0, 20,
  3, true);
  myTween.onMotionChanged = function() {
    target_mc._parent.filters = [new BlurFilter(target_mc.blur,
  target_mc.blur, 1)];
  };
  myTween.onMotionFinished = function() {
    myTween.yoyo();
  }
};
```

```
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mclListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  holder_mc.img_mc);
```

このコードは、3 つのセクションに分かれています。最初のセクションは、必要なクラスとパッケージを読み込みます。2 番目のセクションは、イメージをロードするために使用されるネストしたムービークリップを作成し、この "holder" ムービークリップにフィルタを適用します。最後のセクションでは、ムービークリップローダーの新しい MovieClipLoader インスタンスとリスナーを作成します。リスナーオブジェクトは、単一の `onLoadInit` というイベントハンドラ関数を定義します。この関数は、イメージのロードが完了すると開始し、ステージ上で使用できます。まず、イメージが、ステージの中央に再配置され、次に、ムービークリップをアニメーション化して、0 と 10 の間のぼかしフィルタを適用する Tween オブジェクトが新規に作成されます。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

clone() メソッドの使用

各フィルタクラスに含まれる `clone()` クラスは、元のフィルタインスタンスと同じプロパティをすべて含む新しいフィルタインスタンスを返します。フィルタを操作する際に、フィルタのコピーを作成する場合は、`clone()` メソッドを使用してフィルタを複製する必要があります。clone メソッドを使わずにフィルタを複製すると、元のフィルタだけを指す参照が作成されます。元のフィルタを指す参照が作成されると、複製したフィルタへ加えられた変更が、元のフィルタオブジェクトにも適用されてしまいます。

次の手順は、DropShadowFilter インスタンス (`greenDropShadow`) を新規に作成し、`clone()` メソッドを呼び出して、緑色のドロップシャドウフィルタを複製し、複製したフィルタに `redDropShadow` という名前を付けて保存します複製されたフィルタは、新しいドロップシャドウカラーを設定し、両方のフィルタが、ステージ上の `flower_mc` ムービークリップインスタンスに適用されます。

clone メソッドを使用するには :

1. 新しい Flash ドキュメントを作成し、clone.fla という名前を付けます。
2. ステージ上にムービークリップを作成します。
3. ムービークリップインスタンスを選択し、プロパティインスペクタ内の [インスタンス名] テキストボックスに「flower_mc」と入力します。

4. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを追加します。

```
import flash.filters.DropShadowFilter;
var greenDropShadow:DropShadowFilter = new DropShadowFilter();
greenDropShadow.color = 0x00FF00; // 緑
var redDropShadow:DropShadowFilter = greenDropShadow.clone();
redDropShadow.color = 0xFF0000; // 赤
flower_mc.filters = [greenDropShadow, redDropShadow];
```

このコードは、ドロップシャドウフィルタの新しいインスタンスを作成し、greenDropShadow という名前を付けます。DropShadowFilter.clone() メソッドで、緑のドロップシャドウオブジェクトの複製が作成され、新しいフィルタオブジェクトとして、redDropShadow という名前が付けられます。緑と赤の両方のドロップシャドウフィルタが、ステージ上の flower_mc ムービークリップインスタンスに適用されます。clone() メソッドを呼び出して複製しない場合は、ドロップシャドウは両方とも赤になります。これは、赤のドロップシャドウには緑のドロップシャドウへの参照が含まれるため、redDropShadow.color プロパティを設定すると、赤と緑の両方のドロップシャドウオブジェクトが変更されてしまうからです。

5. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

clone メソッドでフィルタが複製され、両方のフィルタが flower_mc インスタンスに適用されます。

clone() メソッドの詳細については、『ActionScript 2.0 リファレンスガイド』の「clone (DropShadowFilter.clone メソッド)」を参照してください。関連情報については、フィルタクラスの clone() メソッドを参照してください。

BitmapData クラスによるビットマップの作成

BitmapData クラスでは、任意のサイズの透明または不透明のビットマップイメージを作成し、実行時にさまざまな方法で操作できます。ActionScript を使用して BitmapData インスタンスを直接操作することによって、Flash Player でベクターデータのコンテンツを連続的に再描画するオーバーヘッドを避けながら、複雑なイメージを作成できます。BitmapData クラスのメソッドは、Flash ワークスペースの [フィルタ] タブを通じては使用できない各種の効果をサポートします。

BitmapData オブジェクトには、ピクセルデータの配列が含まれています。このデータは、完全に不透明なビットマップも、アルファチャンネルデータを含む透明なビットマップも表現できます。いずれの種類の BitmapData オブジェクトも 32 ビット整数のバッファとして保存されます。各 32 ビット整数は、ビットマップ内の 1 つのピクセルのプロパティを決定します。各 32 ビット整数は、アルファ透明度とピクセルの赤緑青 (ARGB) の値を表す 4 つの 8 ビット「チャンネル」値 (0 ～ 255) の組み合わせです。

パッケージの操作の詳細については、443 ページの「フィルタパッケージの操作」を参照してください。

BitmapData クラスを使用してイメージを操作する例を示すサンプルのソースファイル "BitmapData.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/BitmapData" フォルダに移動してサンプルにアクセスします。

次の手順は、ステージに JPEG イメージを動的にロードし、BitmapData クラスを使用して、テレビのノイズ画面のようなノイズ効果を作成します。ノイズ効果は、100 ミリ秒 (1 秒の 1/10) ごとにランダムなパターンで再描画されます。*x* 軸および *y* 軸に沿ってマウスポインタを動かすと、一定の間隔でノイズが描画される様子が変化します。

BitmapData クラスでノイズ効果を作成するには :

1. 新しい Flash ドキュメントを作成し、noise.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.display.BitmapData;
this.createTextField("status_txt", 90, 0, 0, 100, 20);
status_txt.selectable = false;
status_txt.background = 0xFFFFFF;
status_txt.autoSize = "left";
function onMouseMove() {
  status_txt._x = _xmouse;
  status_txt._y = _ymouse-20;
  updateAfterEvent();
}
this.createEmptyMovieClip("img_mc", 10);
img_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
var noiseBmp:BitmapData = new BitmapData(Stage.width, Stage.height, true);
this.attachBitmap(noiseBmp, 20);
setInterval(updateNoise, 100);
var grayScale:Boolean = true;
function updateNoise():Void {
  var low:Number = 30 * _xmouse / Stage.width;
  var high:Number = 200 * _ymouse / Stage.height;
  status_txt.text = "low:" + Math.round(low) + ", high:" + Math.round(high);
  noiseBmp.noise(Math.round(Math.random() * 100000), low, high, 8, true);
}
```

このコードは、`status_txt` という名前のインスタンスを含むテキストフィールドを作成します。このインスタンスは、マウスポインタの位置を追跡し、`noise()` メソッドの `high` と `low` パラメータに、現在のマウスポインタの位置を示します。`setInterval()` 関数は、`updateNoise()` を呼び出し続けることによって、ノイズ効果を変化させます。ノイズ効果は、100 ミリ秒 (1 秒の 1/10) ごとに再描画されます。`noise()` メソッドの `high` パラメータと `low` パラメータは、ステージ上のマウスポインタの現在の位置を計算することによって、指定されます。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   マウスポインタをステージの *x* 軸に沿って動かすと low パラメータが変わり、*y* 軸に沿って動かすと high パラメータが変わります。

The BitmapData クラスは、perlinNoise() メソッド効果および置き換えマップフィルタと組み合わせて使用することによって、動的にロードされたイメージを歪曲する効果を与えます。次の手順で例を示します。

イメージに置き換えマップフィルタを適用するには :

1. 新しい Flash ドキュメントを作成し、displacement.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
// クラスを読み込み
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
// クリップ 1 つとネストされたクリップを 1 つ作成
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 1);
shapeClip.createEmptyMovieClip("holderClip", 1);
// JPEG をロード
var imageLoader:MovieClipLoader = new MovieClipLoader();
imageLoader.loadClip("http://www.helpexamples.com/flash/images/image4.jpg",
  shapeClip.holderClip);
// BitmapData インスタンスを作成
var perlinBmp:BitmapData = new BitmapData(Stage.width, Stage.height);
perlinBmp.perlinNoise(Stage.width, Stage.height, 10, Math.round(Math.random()
  * 100000), false, true, 1, false);
// 置き換えマップフィルタを作成および適用
var displacementMap:DisplacementMapFilter = new
  DisplacementMapFilter(perlinBmp, new Point(0, 0), 1, 1, 100, 100, "color",
  1);
shapeClip.filters = [displacementMap];
// リスナーを作成および適用
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  perlinBmp.perlinNoise(Stage.width, Stage.height, 10,
  Math.round(Math.random() * 100000), false, true, 1, false);
  shapeClip.filters = [displacementMap];
}
Mouse.addListener(mouseListener);
```

このコードは、5 つの論理セクションで構成されます。最初のセクションは、必要なクラスを読み込みます。2 番目のコードブロックは、ネストされたムービークリップを作成し、リモートサーバーから JPEG イメージをロードします。3 番目のコードブロックは、`perlinBmp` という名前の BitmapData インスタンスを新規に作成します。インスタンスの大きさは、ステージとほぼ同じです。`perlinBmp` インスタンスには、Perlin ノイズ効果の結果が含まれ、後で、置き換えマップフィルタのパラメータとして使用されます。4 番目のコードブロックは、置き換えマップフィルタを作成し、以前に作成され、動的にロードされたイメージに適用します。5 番目 (最後) のコードブロックは、マウスポインタがステージで移動するたびに、置き換えマップフィルタが使用する Perlin ノイズを再生成するマウスリスナーを作成します。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

ブレンドモードについて

ブレンドモードは、Flash ワークスペースまたは ActionScript を使用して、ムービークリップオブジェクトに適用できます。実行時に、複数のグラフィックが 1 つのシェイプとして結合されます。このため、グラフィックシンボルごとに異なるブレンドモードを適用することはできません。

ActionScript を使用してブレンドモードを適用する方法の詳細については、479 ページの「ブレンドモードの適用」を参照してください。

ブレンドモードには、あるイメージ (ベースイメージ) の色と別のイメージ (ブレンドイメージ) の色を組み合わせて第 3 のイメージを作成することも含まれます。イメージ内の各ピクセル値を、別のイメージの対応するピクセル値と共に処理することで、結果に含まれる同じ位置のピクセル値を作成します。

`MovieClip.blendMode` プロパティでは、次のブレンドモードをサポートします。

add　通常、2 つのイメージ間での色を薄くするディゾルブをアニメーション処理するのに使用されます。

alpha　通常、前景のアルファを背景に適用するのに使用されます。

darken　通常、重ね合わせ表示のタイプに使用されます。

difference　通常、明るくて刺激的な色の作成に使用されます。

erase　通常、前景のアルファを使用した背景の一部を " 切り抜く " (消す) ために使用されます。

hardlight　通常、陰影効果の作成に使用されます。

invert　背景を反転するために使用されます。

layer　特定のムービークリップを事前構成するための一時バッファーの作成を強制するために使用されます。

lighten　通常、重ね合わせ表示のタイプに使用されます。

multiply　通常、影と深度の効果の作成に使用されます。

normal　ブレンドイメージのピクセル値でベースイメージのピクセル値を上書きするように指定するために使用されます。

overlay　通常、陰影効果の作成に使用されます。

screen　通常、ハイライトとレンズのフレアに使用されます。

subtract　通常、2 つのイメージ間での色を濃くするディゾルブ効果をアニメーション処理するのに使用されます。

ブレンドモードの適用

次の手順は、動的なイメージをロードし、ステージ上のコンボボックスからブレンドモードを選択して、異なるブレンドモードをイメージに適用します。

異なるブレンドモードをイメージに適用するには :

1. 新しい Flash ドキュメントを作成し、blendmodes.fla という名前を付けて保存します。
2. ComboBox コンポーネントインスタンスをステージまでドラッグし、インスタンス名を blendMode_cb にします。
3. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var blendMode_dp:Array = new Array();
blendMode_dp.push({data:"add", label:"add"});
blendMode_dp.push({data:"alpha", label:"alpha"});
blendMode_dp.push({data:"darken", label:"darken"});
blendMode_dp.push({data:"difference", label:"difference"});
blendMode_dp.push({data:"erase", label:"erase"});
blendMode_dp.push({data:"hardlight", label:"hardlight"});
blendMode_dp.push({data:"invert", label:"invert"});
blendMode_dp.push({data:"layer", label:"layer"});
blendMode_dp.push({data:"lighten", label:"lighten"});
blendMode_dp.push({data:"multiply", label:"multiply"});
blendMode_dp.push({data:"normal", label:"normal"});
blendMode_dp.push({data:"overlay", label:"overlay"});
blendMode_dp.push({data:"screen", label:"screen"});
blendMode_dp.push({data:"subtract", label:"subtract"});
blendMode_cb.dataProvider = blendMode_dp;
```

```
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  var blendModeClip:MovieClip =
  target_mc.createEmptyMovieClip("blendModeType_mc", 20);
  with (blendModeClip) {
    beginFill(0x999999);
    moveTo(0, 0);
    lineTo(target_mc._width / 2, 0);
    lineTo(target_mc._width / 2, target_mc._height);
    lineTo(0, target_mc._height);
    lineTo(0, 0);
    endFill();
  }
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  blendModeClip.blendMode = blendMode_cb.value;
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);

function cbListener(eventObj:Object):Void {
  img_mc.blendModeType_mc.blendMode = eventObj.target.value;
}
blendMode_cb.addEventListener("change", cbListener);
```

この ActionScript コードは、ブレンドモードの各タイプを含むコンボボックスを作成し、ユーザーが、動的にロードされたイメージ上で各ブレンドモードの効果を確認できるようにします。MovieClipLoader インスタンスで使用されるリスナーオブジェクトが作成されます。リスナーオブジェクトは、単一の onLoadInit というイベントリスナーを定義します。このリスナーは、イメージのロードが完了し、Flash で初期化されると起動します。このイベントリスナーは、blendModeType_mc という新しいムービークリップを作成し、Drawing API を使用して矩形シェイプをイメージの半分に重なるように描画します。現在コンボボックスインスタンスで選択されているブレンドモードが、blendModeType_mc ムービークリップに適用されます。

続くコードは、MovieClipLoader インスタンスが、ステージ上のムービークリップに特定のイメージをロードするように設定します。最後のコードは、コンボボックスインスタンスで新しいアイテムを選択すると、そのブレンドモードが適用されるように、blendMode_cb コンボボックスインスタンスのリスナーを定義します。

4. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

操作順序について

次の一覧は、ムービークリップインスタンスに対して、フィルタ配列、ブレンドモード、カラー変換およびマスクレイヤーをアタッチまたは実行する場合の操作順序を示します。

1. ムービークリップのビットマップが、ベクターコンテンツから更新されます (`cacheAsBitmap` プロパティを `true` に設定)。
2. `setMask()` メソッドが使用され、マスクにビットマップキャッシュが含まれる場合、2 つのイメージのアルファブレンドが実行されます。
3. 次に、ぼかし、ドロップシャドウ、グローなどのフィルタが適用されます。
4. ColorTransform クラスを使用する場合は、カラー変換が実行され、ビットマップ結果としてキャッシュされます。
5. ブレンドモードを適用する場合、ベクターレンダラを使用してブレンドが実行されます。
6. 外部マスクレイヤーを適用する場合、ベクターレンダラを使用してマスキングが実行されます。

ActionScript を使用した描画

MovieClip クラスのメソッドを使用して、ステージで線と塗りを描画することができます。これによって、ユーザー用の描画ツールを作成することや、イベントに応じて SWF ファイル内にシェイプを描画することができます。MovieClip クラスの描画メソッドは次のとおりです。

- `beginFill()`
- `beginGradientFill()`
- `clear();`
- `curveTo()`
- `endFill();`
- `lineTo()`
- `lineStyle()`
- `moveTo()`

これらの描画メソッドは、どのようなムービークリップにも使用できます。ただし、オーサリングモードで作成されたムービークリップに描画メソッドを使用すると、描画メソッドはそのクリップが描画される前に実行されます。つまり、オーサリングモードで作成されたコンテンツは、描画メソッドによって描画されたコンテンツの上に描画されます。

描画メソッドを持つムービークリップをマスクとして使用できます。ただし、すべてのムービークリップマスクと同様に、線は無視されます。

ActionScript による描画の詳細については、次のトピックを参照してください。

- 482 ページの「描画メソッドによる線、曲線、シェイプの描画」
- 484 ページの「特定のシェイプの描画」
- 488 ページの「複雑なグラデーションの塗りの使用」
- 489 ページの「線スタイルの使用」
- 495 ページの「Drawing API メソッドの使用とスクリプトによるアニメーションの作成」

Flash アプリケーションでの Drawing API の使用法を示すサンプルのソースファイル "drawingapi.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/DrawingAPI" フォルダに移動してサンプルにアクセスします。

描画メソッドによる線、曲線、シェイプの描画

Flash の Drawing API を使用して、実行時にステージ上でシェイプを動的に作成できます。描画したシェイプを使用して、コンテンツの動的なマスク、フィルタの適用、ステージ上でのアニメーション化が可能です。また、Drawing API を使用して、ユーザーがマウスやキーボードで SWF ファイルにシェイプを描画できるような、さまざまなユーザー用描画ツールを作成できます。

線を描画するには :

1. 新しい Flash ドキュメントを作成し、line.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createEmptyMovieClip("line_mc", 10);
line_mc.lineStyle(1, 0x000000, 100);
line_mc.moveTo(0, 0);
line_mc.lineTo(200, 100);
line_mc._x = 100;
line_mc._y = 100;
```

   このコードは、ステージ上の 0,0 から 200,100 まで線を描画します。線の _x 座標と _y 座標が変更され、線の位置が 100,100 に変わります。

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

複雑なシェイプを描画するには、`MovieClip.lineTo()` を続けて呼び出して、矩形、四角形、楕円を描画します。

曲線を描画するには :

1. 新しい Flash ドキュメントを作成し、curve.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createEmptyMovieClip("circle_mc", 1);
with (circle_mc) {
  lineStyle(4, 0x000000, 100);
  beginFill(0xFF0000);
  moveTo(200, 300);
  curveTo(300, 300, 300, 200);
  curveTo(300, 100, 200, 100);
  curveTo(100, 100, 100, 200);
  curveTo(100, 300, 200, 300);
  endFill();
}
```

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

   このコードは、Drawing API を使用してステージ上に円を描画します。円シェイプは、`MovieClip.curveTo()` メソッドの呼び出しを 4 回だけ使用して描画されているので、やや歪んでいます。Drawing API を使用して円を作成する他の例としては、484 ページの「特定のシェイプの描画」の円の作成手順を参照してください。その手順では、`MovieClip.curveTo()` メソッドを 8 回呼び出しているので、よりリアルに円が描画されています。

三角形を描画するには :

1. 新しい Flash ドキュメントを作成し、triangle.fla という名前を付けて保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createEmptyMovieClip("triangle_mc", 1);
```

   このコードは、`MovieClip.createEmptyMovieClip()` を使用してステージに空のムービークリップを作成します。新しいムービークリップは、既存のムービークリップ (この場合はメインタイムライン) の子になります。

3. 前の手順で入力したコードに従って、タイムラインのフレーム 1 に次の ActionScript を追加します。

```
with (triangle_mc) {
  lineStyle(5, 0xFF00FF, 100);
  moveTo(200, 200);
  lineTo(300, 300);
  lineTo(100, 300);
  lineTo(200, 200);
}
```

   このコードでは、空のムービークリップ (`triangle_mc`) が描画メソッドを呼び出します。紫色の 5 ポイントの線で塗りのない三角形が描画されます。

4. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

これらのメソッドの詳細については、『ActionScript 2.0 リファレンスガイド』の「MovieClip」で該当する項目を参照してください。

Flash アプリケーションでの Drawing API の使用法を示すサンプルのソースファイル "drawingapi.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/DrawingAPI" フォルダに移動してサンプルにアクセスします。

特定のシェイプの描画

このセクションでは、角丸の矩形や円など、より高度なシェイプを柔軟に描画するための方法を説明します。

矩形を作成するには :

1. 新しい Flash ドキュメントを作成し、rect.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript コードを追加します。

```
this.createEmptyMovieClip("rectangle_mc", 10);
rectangle_mc._x = 100;
rectangle_mc._y = 100;
drawRectangle(rectangle_mc, 240, 180, 0x99FF00, 100);
function drawRectangle(target_mc:MovieClip, boxWidth:Number,
  boxHeight:Number, fillColor:Number, fillAlpha:Number):Void {
  with (target_mc) {
    beginFill(fillColor, fillAlpha);
    moveTo(0, 0);
    lineTo(boxWidth, 0);
    lineTo(boxWidth, boxHeight);
    lineTo(0, boxHeight);
    lineTo(0, 0);
    endFill();
  }
}
```

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

   ステージ上にシンプルな緑の矩形が描画され、100,100 の位置に配置されます。矩形のサイズ、塗りのカラー、透明度を変更するには、drawRectangle() メソッドの呼び出しの対応する値を変更します。MovieClip.beginFill() メソッドのコンテンツを変更する必要はありません。

次の手順では、Drawing API を使用して角の丸い矩形を作成します。

角丸矩形を作成するには :

1. 新しい Flash ドキュメントを作成し、roundrect.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript コードを追加します。

```
this.createEmptyMovieClip("rectangle_mc", 10);
rectangle_mc._x = 100;
rectangle_mc._y = 100;
drawRoundedRectangle(rectangle_mc, 240, 180, 20, 0x99FF00, 100);
function drawRoundedRectangle(target_mc:MovieClip, boxWidth:Number,
  boxHeight:Number, cornerRadius:Number, fillColor:Number,
  fillAlpha:Number):Void {
  with (target_mc) {
    beginFill(fillColor, fillAlpha);
    moveTo(cornerRadius, 0);
    lineTo(boxWidth - cornerRadius, 0);
    curveTo(boxWidth, 0, boxWidth, cornerRadius);
    lineTo(boxWidth, cornerRadius);
    lineTo(boxWidth, boxHeight - cornerRadius);
    curveTo(boxWidth, boxHeight, boxWidth - cornerRadius, boxHeight);
    lineTo(boxWidth - cornerRadius, boxHeight);
    lineTo(cornerRadius, boxHeight);
    curveTo(0, boxHeight, 0, boxHeight - cornerRadius);
    lineTo(0, boxHeight - cornerRadius);
    lineTo(0, cornerRadius);
    curveTo(0, 0, cornerRadius, 0);
    lineTo(cornerRadius, 0);
    endFill();
  }
}
```

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   ステージに、幅 240 ピクセル、高さ 180 ピクセルで、20 ピクセルの角の丸さを持つ緑色の矩形が現れます。`MovieClip.createEmptyMovieClip()` を使用して新しいムービークリップを作成し、カスタム関数の `drawRoundedRectangle()` を呼び出すことによって、角丸矩形の複数のインスタンスを作成できます。

次の手順では、Drawing API を使用して真円を作成します。

円を作成するには :

1. 新しい Flash ドキュメントを作成し、circle2.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript コードを追加します。

```
this.createEmptyMovieClip("circle_mc", 10);
circle_mc._x = 100;
circle_mc._y = 100;
drawCircle(circle_mc, 100, 0x99FF00, 100);

function drawCircle(target_mc:MovieClip, radius:Number, fillColor:Number,
  fillAlpha:Number):Void {
  var x:Number = radius;
  var y:Number = radius;
  with (target_mc) {
    beginFill(fillColor, fillAlpha);
    moveTo(x + radius, y);
    curveTo(radius + x, Math.tan(Math.PI / 8) * radius + y, Math.sin(Math.PI
  / 4) * radius + x, Math.sin(Math.PI / 4) * radius + y);
    curveTo(Math.tan(Math.PI / 8) * radius + x, radius + y, x, radius + y);
    curveTo(-Math.tan(Math.PI / 8) * radius + x, radius+ y, -
  Math.sin(Math.PI / 4) * radius + x, Math.sin(Math.PI / 4) * radius + y);
    curveTo(-radius + x, Math.tan(Math.PI / 8) * radius + y, -radius + x, y);
    curveTo(-radius + x, -Math.tan(Math.PI / 8) * radius + y, -
  Math.sin(Math.PI / 4) * radius + x, -Math.sin(Math.PI / 4) * radius + y);
    curveTo(-Math.tan(Math.PI / 8) * radius + x, -radius + y, x, -radius +
  y);
    curveTo(Math.tan(Math.PI / 8) * radius + x, -radius + y, Math.sin(Math.PI
  / 4) * radius + x, -Math.sin(Math.PI / 4) * radius + y);
    curveTo(radius + x, -Math.tan(Math.PI / 8) * radius + y, radius + x, y);
    endFill();
  }
}
```

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

このコードは、前の例よりも高度で正確な円を作成します。curveTo( ) メソッドの呼び出しを 4 回使用するのではなく、8 回使用することによって、真円に近い外観を作成できます。

次の手順では、Drawing API を使用して三角形を作成します。

高度な三角形を作成するには :

1. 新しい Flash ドキュメントを作成し、fancytriangle.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createEmptyMovieClip("triangle_mc", 10);
triangle_mc._x = 100;
triangle_mc._y = 100;
drawTriangle(triangle_mc, 100, 0x99FF00, 100);

function drawTriangle(target_mc:MovieClip, sideLength:Number,
  fillColor:Number, fillAlpha:Number):Void {
  var tHeight:Number = sideLength * Math.sqrt(3) / 2;
  with (target_mc) {
    beginFill(fillColor, fillAlpha);
    moveTo(sideLength / 2, 0);
    lineTo(sideLength, tHeight);
    lineTo(0, tHeight);
    lineTo(sideLength / 2, 0);
    endFill();
  }
}
```

   Drawing API によって、正三角形がステージ上に描画され、指定した塗りのカラーとアルファ値 (透明度) が適用されます。

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

Flash アプリケーションでの Drawing API の使用法を示すサンプルのソースファイル "drawingapi.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/DrawingAPI" フォルダに移動してサンプルにアクセスします。

複雑なグラデーションの塗りの使用

Flash の Drawing API では、単色の塗りだけでなく、グラデーション塗りもサポートします。次の手順は、ステージ上にムービークリップを新規作成し、Drawing API を使用して四角形を作成し、その四角形を赤と青の放射状グラデーションで塗りつぶします。

複雑なグラデーションを作成するには :

1. 新しい Flash ドキュメントを作成し、radialgradient.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createEmptyMovieClip("gradient_mc", 10);
var fillType:String = "radial";
var colors:Array = [0xFF0000, 0x0000FF];
var alphas:Array = [100, 100];
var ratios:Array = [0, 0xFF];
var matrix:Object = {a:200, b:0, c:0, d:0, e:200, f:0, g:200, h:200, i:1};
var spreadMethod:String = "reflect";
var interpolationMethod:String = "linearRGB";
var focalPointRatio:Number = 0.9;
with (gradient_mc) {
  beginGradientFill(fillType, colors, alphas, ratios, matrix, spreadMethod,
  interpolationMethod, focalPointRatio);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
  lineTo(100, 100);
  endFill();
}
```

   この ActionScript コードは、Drawing API を使用してステージ上に四角形を作成し、eginGradientFill() メソッドを呼び出して、その四角形を赤と青の放射状グラデーションで塗りつぶします。

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して Flash ファイルをテストします。

線スタイルの使用

Flash の Drawing API では、MovieClip.lineTo() および MovieClip.curveTo() メソッドへの今後の呼び出しに使用する線のスタイルを指定できます。指定したスタイルは、次のように、MovieClip.lineStyle() を異なるパラメータで呼び出すまで使われます。

```
lineStyle(thickness:Number, rgb:Number, alpha:Number, pixelHinting:Boolean,
  noScale:String, capsStyle:String, jointStyle:String, miterLimit:Number)
```

パスの描画中に MovieClip.lineStyle() メソッドを呼び出し、パス内の線のセグメントごとに異なるスタイルを指定できます。

ActionScript を使用して線スタイルを設定する方法の詳細については、次のセクションを参照してください。

- 489 ページの「線とキャップのスタイルの設定」
- 490 ページの「線スタイルのパラメータの設定」

線とキャップのスタイルの設定

Flash 8 では線の描画機能が向上しました。Flash Player 8 から新しい線パラメータとして、pixelHinting、noScale、capsStyle、jointStyle および miterLimit が追加されました。

次の手順は、Flash Player 8 で新しいキャップのスタイルとして追加された none、round および square の違いを示します。

ActionScript を使用してキャップのスタイルを設定するには :

1. 新しい Flash ドキュメントを作成し、capstyle.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
// グリッドムービークリップを設定
this.createEmptyMovieClip("grid_mc", 50);
grid_mc.lineStyle(0, 0x999999, 100);
grid_mc.moveTo(50, 0);
grid_mc.lineTo(50, Stage.height);
grid_mc.moveTo(250, 0);
grid_mc.lineTo(250, Stage.height);
// 線 1 (capsStyle:round)
this.createEmptyMovieClip("line1_mc", 10);
with (line1_mc) {
  createTextField("label_txt", 1, 5, 10, 100, 20);
  label_txt.text = "round";
  lineStyle(20, 0x99FF00, 100, true, "none", "round", "miter", 0.8);
  moveTo(0, 0);
  lineTo(200, 0);
  _x = 50;
  _y = 50;
}
```

```
// 線 2 (capsStyle:square)
this.createEmptyMovieClip("line2_mc", 20);
with (line2_mc) {
   createTextField("label_txt", 1, 5, 10, 100, 20);
   label_txt.text = "square";
   lineStyle(20, 0x99FF00, 100, true, "none", "square", "miter", 0.8);
   moveTo(0, 0);
   lineTo(200, 0);
   _x = 50;
   _y = 150;
}
// 線 3 (capsStyle:none)
this.createEmptyMovieClip("line3_mc", 30);
with (line3_mc) {
   createTextField("label_txt", 1, 5, 10, 100, 20);
   label_txt.text = "none";
   lineStyle(20, 0x99FF00, 100, true, "none", "none", "miter", 0.8);
   moveTo(0, 0);
   lineTo(200, 0);
   _x = 50;
   _y = 250;
}
```

このコードは、4 つのムービークリップを動的に作成し、Drawing API を使用して、ステージ上に一連の線を作成します。最初のムービークリップには、2 つの垂直な線が含まれます。2 つの垂直な線は、x 軸上の 50 ピクセルと 250 ピクセルの位置にあります。他の 3 つのムービークリップのそれぞれには、ステージ上で緑の線が描画され、capsStyle が round、square および none に設定されています。

3. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   実行時にステージ上で異なるキャップのスタイルが表示されます。

線スタイルのパラメータの設定

線スタイルのパラメータを設定して、線の外観を変更できます。パラメータを使用して、線スタイルの太さ、カラー、アルファ、拡大・縮小のタイプおよびその他の属性を変更できます。

線の太さの設定

MovieClip.lineStyle() の thickness パラメータを使用して、描画される線の太さをポイント単位で数値として指定できます。0 ～ 255 ポイントの太さの線を描画できます。0 の値を指定した場合は、いわゆる " 極細 " となり、SWF ファイルを拡大表示または表示サイズを変更しても、線の太さは常に 1 ピクセルです。

次の手順は、標準の 1 ピクセルの太さの線と極細線の違いを示します。

極細線を作成するには :

1. 新しい Flash ドキュメントを作成し、hairline.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createEmptyMovieClip("drawing_mc", 10);
// 赤色の極細線を作成
drawing_mc.lineStyle(0, 0xFF0000, 100);
drawing_mc.moveTo(0, 0);
drawing_mc.lineTo(200, 0);
drawing_mc.lineTo(200, 100);
// 青色の 1 ピクセルの太さの線を作成
drawing_mc.lineStyle(1, 0x0000FF, 100);
drawing_mc.lineTo(0, 100);
drawing_mc.lineTo(0, 0);
drawing_mc._x = 100;
drawing_mc._y = 100;
```

   このコードは、Drawing API を使用してステージ上に 2 つの線を描画します。最初の線は、太さが 0 に設定された赤色の極細線で、2 番目の線は、太さが 1 ピクセルに設定された青色の線です。
3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

   初めは、赤色と青色のどちらの線もまったく同じように見えます。SWF ファイルを右クリックして表示されるメニューの [ズームイン] を選択すると、赤色の線は常に 1 ピクセルの太さに表示されますが、青色の線は、[ズームイン] を選択するたびに拡大され、太く表示されます。

線のカラーの設定 (rgb)

lineStyle() メソッドの 2 番目のパラメータの rgb は、現在の線セグメントのカラーを数値として制御します。デフォルトでは黒色 (#000000) の線が描画されますが、0xRRGGBB のシンタックスで 16 進数のカラー値を指定して、他のカラーに変更できます。カラー値のシンタックスの RR は赤の値 (00 ～ FF)、GG は緑の値 (00 ～ FF)、BB は青の値 (00 ～ FF) です。

たとえば、赤色の線は 0xFF0000、緑色の線は 0x00FF00、青色の線は 0x0000FF、紫色の線は 0xFF00FF (赤と青)、白色の線は #FFFFFF、グレーの線は #999999 となります。

線のアルファの設定

lineStyle() メソッドの 3 番目のパラメータの alpha は、線の透明度 (アルファ値) を制御します。透明度は 0 ～ 100 の間の数値で指定し、0 は完全に透明な線、100 は完全に不透明な (表示される) 線です。

線のピクセルヒンティングの設定 (pixelHinting)

線のピクセルヒンティングを指定するパラメータの pixelHinting は、線や曲線のアンカーの位置がピクセルと完全に一致するように調整するかどうかを指定します。線の太さがピクセルと完全に一致するように調整すると、水平または垂直の線がぼやけて表示されるのを防ぐことができます。pixelHinting パラメータは、true または false のブール値で指定します。

線の拡大・縮小のタイプの設定 (noScale)

noScale パラメータをストリング値で指定することによって、線の拡大と縮小のタイプを設定します。水平または垂直方向の縮小・拡大モードで拡大・縮小しない線、常に拡大・縮小する線 (デフォルト)、常に拡大・縮小しない線、のいずれかのタイプを指定します。

ヒント	ユーザーがズームインして拡大表示するようなユーザーインターフェイス要素については、線を拡大・縮小するように設定すると便利ですが、ムービークリップが垂直方向または水平方向に拡大・縮小されるだけなら、その必要はありません。

次の 4 つのモードから、拡大・縮小のタイプを指定できます。次の値のいずれかを noScale プロパティに指定します。

normal　常に太さを拡大・縮小します (デフォルト)。

vertical　オブジェクトを垂直方向に拡大・縮小する場合、太さを拡大・縮小しません。

horizontal　オブジェクトを水平方向に拡大・縮小する場合、太さを拡大・縮小しません。

none　太さを拡大・縮小しません。

線のキャップ (capsStyle) と結合 (jointStyle) の設定

capsStyle パラメータでは 3 つのキャップスタイルを指定できます。

- round (デフォルト)
- square
- none

次の手順では、3 つのキャップスタイルの違いを示します。SWF ファイルのテスト時に、それぞれのキャップスタイルがステージ上でどのように表示されるか確認できます。

キャップのスタイルを設定するには :

1. 新しい Flash ドキュメントを作成し、capsstyle2.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var lineLength:Number = 100;
// round
this.createEmptyMovieClip("round_mc", 10);
round_mc.lineStyle(20, 0xFF0000, 100, true, "none", "round");
round_mc.moveTo(0, 0);
round_mc.lineTo(lineLength, 0);
round_mc.lineStyle(0, 0x000000);
round_mc.moveTo(0, 0);
round_mc.lineTo(lineLength, 0);
round_mc._x = 50;
round_mc._y = 50;
var lbl:TextField = round_mc.createTextField("label_txt", 10, 0, 10,
  lineLength, 20);
lbl.text = "round";
var lineLength:Number = 100;
// square
this.createEmptyMovieClip("square_mc", 20);
square_mc.lineStyle(20, 0xFF0000, 100, true, "none", "square");
square_mc.moveTo(0, 0);
square_mc.lineTo(lineLength, 0);
square_mc.lineStyle(0, 0x000000);
square_mc.moveTo(0, 0);
square_mc.lineTo(lineLength, 0);
square_mc._x = 200;
square_mc._y = 50;
var lbl:TextField = square_mc.createTextField("label_txt", 10, 0, 10,
  lineLength, 20);
lbl.text = "square";
// none
this.createEmptyMovieClip("none_mc", 30);
none_mc.lineStyle(20, 0xFF0000, 100, true, "none", "none");
none_mc.moveTo(0, 0);
none_mc.lineTo(lineLength, 0);
none_mc.lineStyle(0, 0x000000);
none_mc.moveTo(0, 0);
none_mc.lineTo(lineLength, 0);
none_mc._x = 350;
none_mc._y = 50;
var lbl:TextField = none_mc.createTextField("label_txt", 10, 0, 10,
  lineLength, 20);
lbl.text = "none";
```

このコードは、Drawing API を使用して 3 つの線を描画し、それぞれに異なる capsStyle を指定します。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

jointStyle パラメータでは、次の 3 つの結合スタイルを指定できます。

- round (デフォルト)
- miter
- bevel

次の手順では、3 つの結合スタイルの違いを示します。

結合スタイルを設定するには :

1. 新しい Flash ドキュメントを作成し、jointstyles.fla という名前で保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
var lineLength:Number = 100;
// miter
this.createEmptyMovieClip("miter_mc", 10);
miter_mc.lineStyle(25, 0xFF0000, 100, true, "none", "none", "miter", 25);
miter_mc.moveTo(0, lineLength);
miter_mc.lineTo(lineLength / 2, 0);
miter_mc.lineTo(lineLength, lineLength);
miter_mc.lineTo(0, lineLength);
miter_mc._x = 50;
miter_mc._y = 50;
var lbl:TextField = miter_mc.createTextField("label_txt", 10, 0, lineLength +
  20, lineLength, 20);
lbl.autoSize = "center";
lbl.text = "miter";
// round
this.createEmptyMovieClip("round_mc", 20);
round_mc.lineStyle(25, 0xFF0000, 100, true, "none", "none", "round");
round_mc.moveTo(0, lineLength);
round_mc.lineTo(lineLength / 2, 0);
round_mc.lineTo(lineLength, lineLength);
round_mc.lineTo(0, lineLength);
round_mc._x = 200;
round_mc._y = 50;
var lbl:TextField = round_mc.createTextField("label_txt", 10, 0, lineLength +
  20, lineLength, 20);
lbl.autoSize = "center";
lbl.text = "round";
// bevel
this.createEmptyMovieClip("bevel_mc", 30);
bevel_mc.lineStyle(25, 0xFF0000, 100, true, "none", "none", "bevel");
bevel_mc.moveTo(0, lineLength);
bevel_mc.lineTo(lineLength / 2, 0);
bevel_mc.lineTo(lineLength, lineLength);
bevel_mc.lineTo(0, lineLength);
bevel_mc._x = 350;
bevel_mc._y = 50;
```

```
var lbl:TextField = bevel_mc.createTextField("label_txt", 10, 0, lineLength +
  20, lineLength, 20);
lbl.autoSize = "center";
lbl.text = "bevel";
```

Drawing API を使用してステージ上に 3 つの三角形を描画します。それぞれの三角形に異なる結合スタイルを指定しています。

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して、ドキュメントをテストします。

線のマイター限度の設定 (miterLimit)

miterLimit プロパティは、マイター結合でマイターが切り取られる限度を数値で指定します (492ページの「線のキャップ (capsStyle) と結合 (jointStyle) の設定」を参照)。miterLimit の値は、一般的に線のサイズの乗数です。たとえば、miterLimit で 2.5 を指定すると、線のサイズの 2.5 倍の長さの位置で、マイターが切り取られます。有効値の範囲は 0 ～ 255 です (miterLimit の値が undefined の場合、デフォルト値は 3 です)。miterLimit プロパティは jointStyle が miter に設定されている場合にのみ使用できます。

Drawing API メソッドの使用とスクリプトによるアニメーションの作成

Drawing API と Tween クラスおよび TransitionManager クラスを組み合わせて使用することによって、ActionScript コードを少し記述するだけで、効果的なアニメーションを作成できます。

次の手順は、JPEG イメージをロードし、ロードしたイメージを動的にマスクして、イメージのマスクをトゥイーンすることにより、ロード後のイメージがゆっくり表示されるようにします。

動的なマスクをアニメーション化するには :

1. 新しい Flash ドキュメントを作成し、dynmask.fla という名前を付けて保存します。
2. タイムラインのフレーム 1 に次の ActionScript を追加します。

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._visible = false;
  // イメージをステージ上に作成
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  var maskClip:MovieClip = target_mc.createEmptyMovieClip("mask_mc", 20);
```

```
    with (maskClip) {
        // ロードしたイメージと同じサイズのマスクを描画
        beginFill(0xFF00FF, 100);
        moveTo(0, 0);
        lineTo(target_mc._width, 0);
        lineTo(target_mc._width, target_mc._height);
        lineTo(0, target_mc._height);
        lineTo(0, 0);
        endFill();
    }
    target_mc.setMask(maskClip);
    target_mc._visible = true;
    var mask_tween:Tween = new Tween(maskClip, "_yscale", Strong.easeOut, 0,
    100, 2, true);
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
    img_mc);
```

このコードは、Tween クラスと、イージングパッケージ内の各クラスを読み込みます。次に、後のコードブロックで作成する MovieClipLoader インスタンスでリスナーオブジェクトとして使用されるオブジェクトを作成します。リスナーオブジェクトは、単一の `onLoadInit` というイベントリスナーを定義します。このリスナーは、動的にロードした JPEG イメージをステージの中央に配置します。イメージが再配置されると、新しいムービークリップインスタンスを `target_mc` ムービークリップ内に作成します。このムービークリップは、動的にロードされた JPEG イメージを含みます。Drawing API コードが、新しいムービークリップ内の JPEG イメージと同じ大きさの矩形を描画します。`MovieClip.setMask()` メソッドを呼び出して、新しいムービークリップが JPEG イメージをマスクします。マスクの描画および設定が終わると、Tween クラスを使用してマスクがアニメーション化され、イメージがゆっくり表示されます。

3. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

メモ	この例で、`_yscale` ではなく `_alpha` をアニメーション化する場合は、ムービークリップをマスクするのではなく、`target_mc` を直接トゥイーンします。

Flash アプリケーションでの Drawing API の使用法を示すサンプルのソースファイル "drawingapi.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/DrawingAPI" フォルダに移動してサンプルにアクセスします。

拡大・縮小とスライスガイドの理解

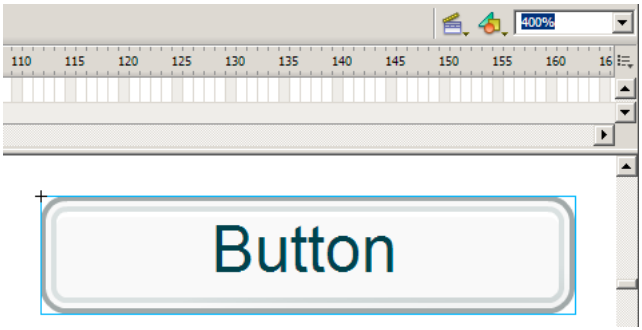
9 スライスの拡大/縮小 (Scale-9) を使って、ムービークリップのコンポーネントスタイルの拡大・縮小を指定することができます。これを使用してムービークリップシンボルを作成すると、グラフィックやデザインエレメントに通常適用されるような種類の拡大/縮小処理ではなく、ユーザーインターフェイスのコンポーネントに適した拡大/縮小処理による表示ができます。

9 スライスの拡大 / 縮小の理解

9 スライスの拡大 / 縮小機能を理解するには、Flash での実例を見るのが最も簡単です。

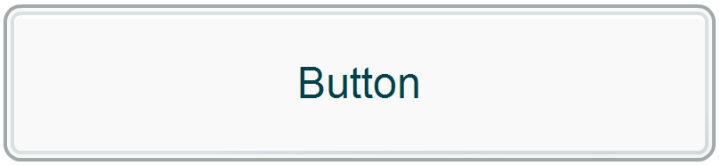
Flash での拡大 / 縮小を理解するには :

1. 新しい Flash ドキュメントを作成し、dynmask.fla という名前を付けて保存します。
2. Button コンポーネントを、[コンポーネント] パネルからステージにドラッグします ([ウィンドウ]-[コンポーネント])。
3. 虫めがねツールでステージのズーム率を 400% に設定します。



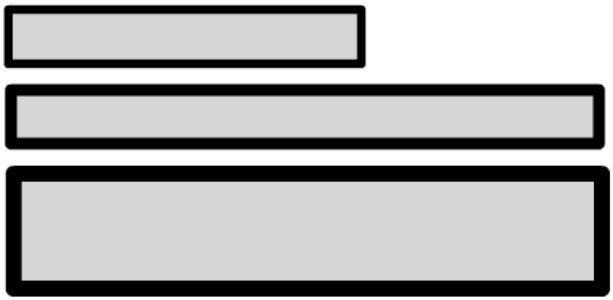
デフォルトでは、Button コンポーネントインスタンスのサイズは、幅 100 ピクセル、高さ 22 ピクセルです。

4. プロパティインスペクタを使用して Button コンポーネントインスタンスのサイズを変更し、幅 200 ピクセル、高さ 44 ピクセルにします。



コンポーネントのサイズを変更しても、ボタンの境界線やテキストラベルは歪んで表示されません。ボタンのラベルは変更前と同じように中央に位置し、フォントサイズも変更前と同じです。9 スライスの拡大 / 縮小を使用していない Adobe Component Architecture バージョン 2 のコンポーネントでも、アウトラインのサイズを変更しない拡大・縮小は可能です (次の図を参照)。

ボタンインスタンスが、3 x 3 のグリッドで 9 つに分割されているとします。電話やテンキーに 9 つの数字が並んでいる様子と同じです。ボタンインスタンスのサイズを横方向に変更すると、中央の縦の 3 つの部分 (テンキーの数字の 2、5、8 に相当する部分) だけが横に伸びます。したがって、コンテンツが歪んで表示されません。ボタンインスタンスのサイズを縦方向に変更すると、上下の真ん中の 3 つの部分 (テンキーの数字の 4、5、6 に相当する部分) だけのサイズが変更されます。グリッドの四隅のセグメントのサイズはまったく変わらないので、コンポーネントのサイズが大きくなっても、引き伸ばされているようには見えません。次の図を参照してください。



ヒント	線は、9 スライスの拡大/縮小の変換が行われた後に、エッジから作成されるので、歪んだり、粗くなったりしません。

Flash の [シンボルに変換] ダイアログボックスまたは [シンボルプロパティ] ダイアログボックスで、9 スライスを拡大/縮小するガイドを有効化できます。[9 スライスの拡大/縮小] チェックボックスは、Flash Player 8 以降用にパブリッシュし、ビヘイビアがムービークリップに設定されている場合にのみ、使用できます。9 スライスを拡大/縮小するガイドは、Flash Player 8 より前のバージョンでは使用できません。また、ボタンやグラフィックシンボルを作成する場合にも使用できません。ActionScript で 9 スライスを拡大/縮小するガイドを有効にするには、ムービークリップインスタンスの `scale9Grid` プロパティを設定します。

ユーザーインターフェイスと ActionScript のどちらを使用してスライスガイドを作成しても、ムービークリップの `scale9Grid` プロパティをトレースすることによって、*x* 座標、*y* 座標、幅および高さをトレースできます。

```
trace(my_mc.scale9Grid); // (x=20, y=20, w=120, h=120)
```

このコードは、`scale9Grid` プロパティが使用する矩形オブジェクトの値をトレースします。この矩形は、x 座標と y 座標が 20 ピクセル、幅 120 ピクセル、高さ 120 ピクセルです。

ActionScript による 9 スライスの拡大 / 縮小の操作

次の例では、描画ツールを使用して、300 x 300 ピクセルの四角形を描画し、9 スライスの拡大 / 縮小を使用して、描画した四角形のサイズを変更します。四角形は 9 つのセグメントに分割され、各セグメントの大きさは約 100 x 100 ピクセルです。四角形のサイズを変更すると、四角形全体が指定された幅と高さになるまで、四隅の 4 つのセグメント以外のセグメントの大きさが変わります。

ActionScript で 9 スライスの拡大 / 縮小を使用するには :

1. 新しい Flash ドキュメントを作成し、ninescale.fla という名前で保存します。
2. Button コンポーネントを現在のドキュメントのライブラリにドラッグします。
3. 矩形ツールを選択して、ステージ上に、300 x 300 ピクセルの赤色の四角形を描画し、線の太さを 15 ピクセル、線の色を黒に設定します。
4. 楕円ツールを選択して、ステージ上に、50 x 50 ピクセルの紫色の円を描画し、線の太さを 2 ピクセル、線の色を黒に設定します。
5. 紫の円を選択して、赤の四角形の右上隅までドラッグします。
6. 楕円ツールを選択して、約 200 x 200 ピクセルの円を新しく描画し、ステージの外へ配置します。
7. 後で作成した円を選択し、円の中心が四角形の左下隅に重なるまでドラッグします。
8. 円インスタンスの外側をクリックして、円の選択を解除します。
9. 円を再度ダブルクリックして選択し、Backspace キーを押してシェイプを削除し、四角形から円形の部分を取り除きます。
10. マウスを使用して、四角形とその内側にある紫の円の全体を選択します。
11. F8 キーを押して、シェイプをムービークリップシンボルに変換します。
12. ステージ上のムービークリップに my_mc というインスタンス名を付けます。
13. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
import mx.controls.Button;
import flash.geom.Rectangle;

var grid:Rectangle = new Rectangle(100, 100, 100, 100);

var small_button:Button = this.createClassObject(Button, "small_button", 10,
  {label:"Small"});
small_button.move(10, 10);
small_button.addEventListener("click", smallHandler);
function smallHandler(eventObj:Object):Void {
     my_mc._width = 100;
     my_mc._height = 100;
}
```

```
var large_button:Button = this.createClassObject(Button, "large_button", 20,
  {label:"Large"});
large_button.move(120, 10);
large_button.addEventListener("click", largeHandler);
function largeHandler(eventObj:Object):Void {
     my_mc._width = 450;
     my_mc._height = 300;
}

var toggle_button:Button = this.createClassObject(Button, "toggle_button",
  30, {label:"scale9Grid=OFF", toggle:true, selected:false});
toggle_button.move(420, 10);
toggle_button.setSize(120, 22);
toggle_button.addEventListener("click", toggleListener);
function toggleListener(eventObj:Object):Void {
  if (eventObj.target.selected) {
    eventObj.target.label = "scale9Grid=ON";
    my_mc.scale9Grid = grid;
  } else {
    eventObj.target.label = "scale9Grid=OFF";
    my_mc.scale9Grid = undefined;
  }
}
```

このコードは、5 つのセクションに分かれています。最初のセクションは、mx.controls.Button (Button コンポーネントクラス) と flash.geom.Rectangle の 2 つのクラスを読み込みます。2 番目のコードセクションは、Rectangle クラスインスタンスを新規に作成して、幅と高さをそれぞれ 100 ピクセルに指定し、*x* 座標と *y* 座標も 100 ピクセルに指定します。この四角形インスタンスは、後で作成されるムービークリップシェイプに 9 スライスの拡大/縮小グリッドを設定する際に使用されます。

次のセクションは、Button コンポーネントインスタンスを新規に作成し、インスタンス名を `small_button` にします。このボタンをクリックすると、作成したムービークリップのサイズが、幅 100 ピクセル、高さ 100 ピクセルになります。4 番目のセクションは、`large_button` という新規の Button インスタンスを動的に作成し、このボタンをクリックすると、ターゲットムービークリップのサイズが幅 450 ピクセル、高さ 300 ピクセルになります。最後のセクションは、ユーザーがオンとオフを切り替えられる Button インスタンスを新規に作成します。ボタンが「オン」の状態の場合、9 スライスグリッドが適用されます。ボタンが「オフ」の状態の場合、9 スライスグリッドが無効になります。

14. Flash ドキュメントを保存し、[制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

このコードは、3 つの Button コンポーネントインスタンスをステージ上に追加して配置し、各ボタンのイベントリスナーを作成します。9 スライスグリッドが無効の状態で大きいボタンをクリックすると、イメージが歪み、引き伸ばされたように見えます。切り替えボタンをクリックして 9 スライスグリッドを有効にし、大きいボタンをクリックします。9 スライスグリッドが有効になり、左上隅の円が歪んだように見えません。

第 13 章

ActionScript によるインタラクティブ機能の作成

13

シンプルなアニメーションでは、Flash Player が SWF ファイルのシーンとフレームを順番に再生します。一方、インタラクティブ SWF ファイルでは、SWF ファイルの他の部分へのジャンプ、オブジェクトの移動、フォームへの情報の入力などのさまざまなインタラクティブ操作を、キーボードとマウスを使用してユーザーが行うことができます。

ActionScript を使用して、イベント発生時に実行するアクションを作成します。スクリプトを実行できるイベントが発生するのは、再生ヘッドがフレームに達したとき、ムービークリップがロードまたはアンロードされたとき、ボタンをクリックしたとき、キーを押したときなどです。

スクリプトには、単一のコマンド (SWF ファイルの停止など) のみを記述することもあれば、一連のコマンドおよびステートメント (最初に条件を評価した後でアクションを実行するなど) を記述することもあります。ActionScript のコマンドの多くは単純なものであり、SWF ファイルの基本的な制御を行うことができます。一部のアクションでは、使用の際にある程度のプログラミング言語の知識が必要な場合があります。このようなアクションは、高度な開発に使用します。

ActionScript でのインタラクティブ機能作成の詳細については、次のトピックを参照してください。

イベントとインタラクティブ機能について 504
SWF ファイル再生の制御 504
インタラクティブ機能と視覚効果の作成 508
ActionScript を使用したランタイムデータバインディングの作成 520
サンプルスクリプトの解説 528

イベントとインタラクティブ機能について

ユーザーによるマウスクリックやキー押下のアクションがあると、そのたびにイベントが生成されます。この種のイベントは、ユーザーによる何らかの操作に呼応して生成されるので、一般に " ユーザーイベント " と呼ばれます。これらのイベントに対応し処理するための ActionScript コードを記述することができます。たとえば、ユーザーがボタンをクリックしたときに再生ヘッドを SWF ファイル内の他のフレームへ送ったり、ブラウザに新しい Web ページをロードするなどが考えられます。

SWF ファイルでは、ボタン、ムービークリップ、およびテキストフィールドは、すべて対応可能なイベントを生成します。ActionScript でイベントを処理する方法は 3 つあります。イベントハンドラメソッド、イベントリスナー、`on()` および `onClipEvent()` ハンドラです。イベントとイベント処理の詳細については、第 9 章の「イベントの処理」を参照してください。

SWF ファイル再生の制御

タイムラインの再生ヘッドの制御や、ブラウザウィンドウへの新規 Web ページのロードには、以下の ActionScript 関数を使用します。

- `gotoAndPlay()` 関数および `gotoAndStop()` 関数は、再生ヘッドを特定のフレームまたはシーンに送ります。これらはグローバル関数であり、どのスクリプトからでも呼び出すことができます。また、`MovieClip.gotoAndPlay()` メソッドおよび `MovieClip.gotoAndStop()` メソッドを使って、特定のムービークリップオブジェクトのタイムラインを移動することができます。505 ページの「フレームまたはシーンへのジャンプ」を参照してください。
- `play()` および `stop()` アクションで SWF ファイルの再生と停止が行われます。506 ページの「ムービークリップの再生と停止」を参照してください。
- `getURL()` アクションで別の URL にジャンプします。507 ページの「別の URL へのジャンプ」を参照してください。

詳細については、次のトピックを参照してください。

- 505 ページの「フレームまたはシーンへのジャンプ」
- 506 ページの「ムービークリップの再生と停止」
- 507 ページの「別の URL へのジャンプ」

フレームまたはシーンへのジャンプ

SWF ファイル内の特定のフレームやシーンにジャンプするには、グローバル関数である gotoAndPlay() および gotoAndStop() を使用するか、同等の機能を持つ MovieClip クラスの MovieClip.gotoAndPlay() メソッドおよび MovieClip.gotoAndStop() メソッドを使用します。どの関数またはメソッドでも、現在のシーン内にジャンプするフレームを指定できます。ドキュメントに複数のシーンがある場合には、ジャンプ先のシーンおよびフレームを指定できます。

次に示す例では、ボタンオブジェクトの onRelease イベントハンドラ内でグローバル関数 gotoAndPlay() を使用して、このボタンを含むタイムラインの再生ヘッドをフレーム 10 に送ります。

```
jump_btn.onRelease = function () {
  gotoAndPlay(10);
};
```

その次に示す例では、MovieClip.gotoAndStop() メソッドを使用して、categories_mc という名前のムービークリップインスタンスのタイムラインをフレーム 10 に送って停止します。MovieClip メソッドの gotoAndPlay() と gotoAndStop() を使用する場合には、メソッドの対象とするインスタンスを指定する必要があります。

```
jump_btn.onPress = function () {
  categories_mc.gotoAndStop(10);
};
```

最後に示す例では、グローバル関数 gotoAndStop() を使用して、再生ヘッドをシーン 2 のフレーム 1 に送ります。シーンを指定しないと、再生ヘッドは現在のシーン内の指定されたフレームに進みます。scene パラメータはルートタイムラインでのみ使用でき、ムービークリップやドキュメント内のその他のオブジェクトのタイムラインでは使用できません。

```
nextScene_mc.onRelease = function() {
  gotoAndStop("Scene 2", 1);
}
```

ムービークリップの再生と停止

特に指示しない限り、SWF ファイルを開始すると、タイムライン内の全フレームを通じて再生されます。SWF ファイルを開始および停止するには、グローバル関数 play( ) および stop( ) を使用するか、または同等の MovieClip メソッドを使用します。たとえば、次のシーンに進む前に、stop( ) を使用してシーンの最後で SWF ファイルを停止することができます。SWF ファイルを停止した後は、play( ) または gotoAndPlay( ) を呼び出して明示的に再開する必要があります。

play( ) 関数 および stop( ) 関数、または MovieClip の同名のメソッドを使用して、メインタイムライン、各種ムービークリップのタイムライン、またはロードした SWF ファイルを制御できます。制御するには、ムービークリップにインスタンス名があり、またそのムービークリップがタイムライン内に存在する必要があります。

次に示す on(press) はボタンに割り当てるハンドラの例で、このボタンオブジェクトを含む SWF ファイルまたはムービークリップの再生ヘッドの移動を開始します。

```
// ボタンインスタンスへの割り当て
on (press) {
   // このボタンを含むタイムラインの再生
   play();
}
```

この同じ on( ) イベントハンドラコードを、ボタンではなくムービークリップオブジェクトに割り当てると、異なる結果になります。ボタンオブジェクトに割り当てた場合、on( ) ハンドラ内のステートメントは、デフォルトでボタンを含むタイムラインに適用されます。一方、ムービークリップオブジェクトに割り当てた場合には、on( ) ハンドラ内のステートメントは、on( ) ハンドラが割り当てられたムービークリップに適用されます。

たとえば、次に示す onPress( ) ハンドラコードでは、ムービークリップを含んでいるタイムラインではなく、ハンドラが割り当てられているムービークリップのタイムラインが停止します。

```
// myMovie_mc ムービークリップインスタンスに割り当てる
myMovie_mc.onPress() {
   stop();
};
```

同じ条件は、ムービークリップオブジェクトに割り当てられた onClipEvent( ) ハンドラにも適用されます。たとえば、次のコードでは、onClipEvent( ) ハンドラを持つムービークリップが最初にロードされたときまたはステージ上に表示されたときに、そのムービークリップのタイムラインを停止させます。

```
onClipEvent(load) {
   stop();
}
```

別の URL へのジャンプ

ブラウザウィンドウで Web ページを開いたり、定義済みの URL の別のアプリケーションにデータを渡したりするには、getURL() グローバル関数または MovieClip.getURL() メソッドを使用します。たとえば、HTML フォームの場合と同様に、新たな Web サイトにリンクするボタンを用意したり、CGI スクリプトにタイムライン変数を送って処理することができます。また、HTML でアンカータグ (<a></a>) を使う場合と同様に、ターゲットウィンドウを指定することも可能です。

たとえば、次のコードでは、homepage_btn という名前のボタンインスタンスをユーザーがクリックすると、空のブラウザウィンドウに adobe.com のホームページが表示されます。

```
// フレームに割り当てる
homepage_btn.onRelease = function () {
  getURL("http://www.adobe.com", "_blank");
};
```

また、GET メソッドまたは POST メソッドのいずれかを使用し、URL と共に変数を送信することもできます。これは、ColdFusion Server (CFM) ページのように、アプリケーションサーバーからロードしているページに、フォーム変数を渡す必要がある場合に便利です。たとえば、"addUser.cfm" という名前の CFM ページをロードするときに、firstName および age という 2 つのフォーム変数を渡す必要があるとします。それには、variables_mc という名前のムービークリップを作成し、2 つの変数を次の例のように定義します。

```
variables_mc.firstName = "Francois";
variables_mc.age = 32;
```

その後、次のコードで "addUser.cfm" を新規のブラウザウィンドウにロードし、POST ヘッダーでこの CFM ページに variables_mc.name および variables_mc.age を渡します。

```
variables_mc.getURL("addUser.cfm", "_blank", "POST");
```

getURL() 機能は、使用するブラウザに依存します。すべてのブラウザを同じように機能させる上で最も信頼性が高いのは、ウィンドウを開くための JavaScript window.open() メソッドを使用する HTML コードで JavaScript 関数を呼び出す方法です。次の HTML と JavaScript を HTML テンプレートに追加します。

```
<script language="JavaScript">
<--
  function openNewWindow(myURL) {
    window.open(myURL, "targetWindow");
  }
// -->
</script>
```

次の ActionScript を使用して、SWF ファイルから openNewWindow を呼び出します。

```
var myURL:String = "http://foo.com";
getURL("javascript:openNewWindow('" + String(myURL) + "');");
```

詳細については、『ActionScript 2.0 リファレンスガイド』の「getURL function」を参照してください。

インタラクティブ機能と視覚効果の作成

インタラクティブ機能やその他の視覚効果を作成するには、以下の手法を理解しておく必要があります。

- 508 ページの「カスタムマウスポインタの作成」
- 509 ページの「ポインタ位置の取得」
- 510 ページの「キー押下動作の取得」
- 513 ページの「カラー値の設定」
- 514 ページの「サウンドの制御」
- 517 ページの「衝突の判定」
- 518 ページの「簡単な線画ツールの作成」

カスタムマウスポインタの作成

標準マウスポインタは、オペレーティングシステムによって画面に表示される、ユーザーのマウスの位置を表すものです。標準のポインタを Flash でデザインしたマウスポインタに置き換えることにより、ユーザーのマウス操作をより緊密に SWF ファイルに統合できます。このセクションのサンプルでは、大きな矢印のようなカスタムポインタを使用します。この機能の長所は、カスタムポインタをどのような形にも擬せられるという点にあります。たとえば、ゴールラインに運ばれるフットボールや、椅子の上にかけると色が変わる布地の見本などです。

カスタムポインタを作成するには、ステージでポインタとなるムービークリップをデザインします。次に、ActionScript で、標準ポインタを非表示にし、その動きを追跡します。標準ポインタを非表示にするには、ビルトイン Mouse クラスの `hide()` メソッドを使用します (『ActionScript 2.0 リファレンスガイド』の「hide (Mouse.hide method)」を参照)。

カスタムポインタを作成するには :

1. カスタムポインタとして使用するムービークリップを作成し、そのクリップのインスタンスをステージ上に配置します。
2. ステージ上のムービークリップインスタンスを選択します。
3. プロパティインスペクタで、[インスタンス名] テキストボックスに「cursor_mc」と入力します。
4. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
Mouse.hide();
cursor_mc.onMouseMove = function() {
  this._x = _xmouse;
  this._y = _ymouse;
  updateAfterEvent();
};
```

`Mouse.hide()` メソッドは、ムービークリップが最初にステージに表示されたときにポインタを非表示にします。`onMouseMove` 関数はポインタと同じ場所にカスタムポインタを配置し、マウスを移動するたびに `updateAfterEvent()` を呼び出します。

updateAfterEvent() 関数は、次のフレームが描画されるときに画面が更新されるというデフォルトの動作ではなく、指定したイベントの発生した直後に画面を更新します。詳細については、『ActionScript 2.0 リファレンスガイド』の「updateAfterEvent 関数」を参照してください。

5. [制御]-[ムービープレビュー] を選択し、カスタムポインタをテストします。

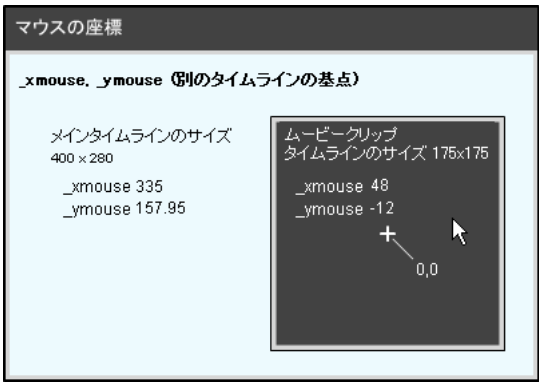
カスタムマウスポインタを使用するときもボタンは有効です。カスタムポインタはタイムラインの最上部レイヤーに配置することをお勧めします。そうすることで、SWF ファイルでマウスを動かしたときに、ボタンや他のレイヤー上のオブジェクトより手前にカスタムポインタが表示されます。また、カスタムポインタの先端は、カスタムポインタとして使用しているムービークリップの基準点です。そのため、ムービークリップの特定の部分をポインタの先端とする場合には、クリップの基準点の座標がその点になるように設定します。

Mouse クラスのメソッドの詳細については、『ActionScript 2.0 リファレンスガイド』の「Mouse」を参照してください。

ポインタ位置の取得

_xmouse プロパティと _ymouse プロパティを使用すると、SWF ファイル内でポインタの位置を調べることができます。たとえば、地図アプリケーションで _xmouse プロパティと _ymouse プロパティの値を取得して、特定の場所の経度と緯度を計算することなどが可能です。

各タイムラインには、座標系内のポインタの位置を返す _xmouse プロパティと _ymouse プロパティがあります。位置は、常に基準点に対して相対的です。メインタイムライン (_level0) については、基準点は左上隅です。ムービークリップの場合、基準点はクリップが作成されたとき、またはステージに配置されたときの設定に左右されます。



メインタイムライン内およびムービークリップタイムライン内の *_xmouse* プロパティと *_ymouse* プロパティ

次に、メインタイムラインまたはムービークリップ内のポインタ位置を取得する方法をいくつか示します。

現在のポインタ位置を取得するには :

1. 2 つのダイナミックテキストフィールドを作成し、box1_txt および box2_txt という名前を付けます。
2. これらのテキストボックスに、それぞれ "X position" および "Y position" というラベルを付けます。
3. [アクション] パネルが表示されていない場合は、[ウィンドウ]-[アクション] を選択して表示します。
4. スクリプトペインに次のコードを追加します。

```
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
   // マウスの X 位置と Y 位置を返す
   box1_txt.text = _xmouse;
   box2_txt.text = _ymouse;
};
Mouse.addListener(mouseListener);
```

5. [制御]-[ムービープレビュー] を選択して Flash ムービーをテストします。[box1_txt] フィールドと [box2_txt] フィールドには、ステージ上でポインタを移動しているときの位置が表示されます。

_xmouse プロパティと _ymouse プロパティの詳細については、『ActionScript 2.0 リファレンスガイド』の「_xmouse (MovieClip._xmouse プロパティ)」および「_ymouse (MovieClip._ymouse プロパティ)」を参照してください。

キー押下動作の取得

グローバルな on() ハンドラを使用すると、Flash Player でのキー押下操作に対するビルトインの動作を上書きできます。次に例を示します。

```
/* 左矢印キーまたは右矢印キーを押すと、ハンドラがアタッチされているムービークリップの透明度が変
   化します。*/
on (keyPress "<Left>") {
   this._alpha -= 10;
}
on (keyPress "<Right>") {
   this._alpha += 10;
}
```

必ず [制御]-[キーボードショートカットを無効] を選択しておいてください。そうでない場合、[制御]-[ムービープレビュー] を使用してアプリケーションをテストすると、ビルトインの動作が設定されている一部のキーについては動作が上書きされません。詳細については、『ActionScript 2.0 リファレンスガイド』で「on ハンドラ」の keyPress パラメータを参照してください。

ビルトインクラス Key クラスのメソッドを使用すると、ユーザーが最後に押したキーを検出できます。Key クラスは、コンストラクタ関数を必要としません。Key クラスのメソッドを使用するには、次の例のようにクラスのメソッドを呼び出すだけです。

```
Key.getCode();
```

押されたキーのバーチャルキーコードまたは ASCII (American Standard Code for Information Interchange) 値は、次のようにして取得できます。

- 最後に押したキーのバーチャルキーコードを取得するには、getCode() メソッドを使用します。
- 最後に押したキーの ASCII 値を取得するには、getAscii() メソッドを使用します。

バーチャルキーコードはキーボード上のすべての物理キーに割り当てられます。たとえば、左矢印キーにはバーチャルキーコード 37 が割り当てられています。バーチャルキーコードを使用すると、言語やプラットフォームの種類にかかわらず、すべてのキーボード上で同じように SWF ファイルを制御できます。

ASCII 値は、すべての文字セット内の最初の 127 個の文字に割り当てられます。ASCII 値は、画面上の文字に関する情報を示します。たとえば、英字 "A" と "a" の ASCII 値は異なります。

使用するキーを決め、バーチャルキーコードを判定するには、次のいずれかの方法を使用します。

- キーコードの一覧については、付録 C の「キーボードのキーとキーコードの値」を参照してください。
- Key クラスの定数を使用します。([アクション] ツールボックスで、[ActionScript 2.0 クラス]-[ムービー]-[キー]-[定数]を選択します)。
- 次の `onClipEvent()` ハンドラをムービークリップに割り当ててから、[制御]-[ムービープレビュー] を選択して、目的のキーを押します。

```
onClipEvent(keyDown) {
   trace(Key.getCode());
}
```

目的のキーのキーコードが [出力] パネルに表示されます。

Key クラスのメソッドは一般にイベントハンドラ内で使用します。次の例では、ユーザーは矢印キーを使って車を移動させます。`Key.isDown()` メソッドは、押されているのが右矢印キーなのか、左矢印キーなのか、上矢印キーなのか、下矢印キーなのかを示します。イベントリスナー `Key.onKeyDown` により、`if` ステートメントからの `Key.isDown (keyCode)` 値が決定されます。その値に応じて、ハンドラは Flash Player に車の位置を更新して方向を表示するよう指示します。

以下の例は、押されたキーを取得し、対応する矢印キー (上矢印、下矢印、左矢印、右矢印) に応じて、ステージ上でムービークリップを上下左右に移動する方法を示したものです。また、押されたキーの名前をテキストフィールドに表示しています。

キーボードで移動できるムービークリップを作成するには :

1. ステージで、キーボードの矢印キーの操作に応答して移動するムービークリップを作成します。

   次の例では、ムービークリップのインスタンス名は `car_mc` です。

2. タイムラインでフレーム 1 を選択します。次に、[アクション] パネルが表示されていない場合は、[ウィンドウ]-[アクション] を選択して表示します。

3. 1 回キーを押す操作での車の移動量を設定するために、`distance` という変数を定義して、その値を 10 とします。

```
var distance:Number = 10;
```

4. 次の ActionScript コードを [アクション] パネルの既存のコードの下に追加します。

```
this.createTextField("display_txt", 999, 0, 0, 100, 20);
```

5. 上下左右のどの矢印キーが押されているかを判別するイベントハンドラをムービークリップ car 用に作成するために、[アクション] パネルに次のコードを追加します。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
};
Key.addListener(keyListener);
```

6. 左矢印キーが押されているかどうかをチェックして、押されている場合には車のムービークリップを左に移動するよう、onEnterFrame イベントハンドラの本体にコードを追加します。

   コードは次の例のようになります (新しい部分を太字で示します)。

```
var distance:Number = 10;
this.createTextField("display_txt", 999, 0, 0, 100, 20);
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.LEFT)) {
    car_mc._x = Math.max(car_mc._x - distance, 0);
    display_txt.text = "Left";
  }
};
Key.addListener(keyListener);
```

   左矢印キーを押すと、車の _x プロパティが現在の _x 値から距離を差し引いた値または 0 のうち、いずれか大きい値に設定されます。したがって、_x プロパティの値が 0 より小さくなることはありません。また、SWF ファイルに [*Left*] という単語が表示されます。

7. 同様のコードを使用して、右矢印キー、上矢印キー、下矢印キーが押されたかどうかをチェックします。

   完成したコードは次の例のようになります (新しい部分を太字で示します)。

```
var distance:Number = 10;
this.createTextField("display_txt", 999, 0, 0, 100, 20);
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.LEFT)) {
    car_mc._x = Math.max(car_mc._x - distance, 0);
    display_txt.text = "Left";
  } else if (Key.isDown(Key.RIGHT)) {
    car_mc._x = Math.min(car_mc._x + distance, Stage.width -
  car_mc._width);
    display_txt.text = "Right";
  } else if (Key.isDown(Key.UP)) {
    car_mc._y = Math.max(car_mc._y - distance, 0);
    display_txt.text = "Up";
  } else if (Key.isDown(Key.DOWN)) {
    car_mc._y = Math.min(car_mc._y + distance, Stage.height -
  car_mc._height);
    display_txt.text = "Down";
  }
};
Key.addListener(keyListener);
```

8. [制御]-[ムービープレビュー] を選択してファイルをテストします。

Key クラスのメソッドの詳細については、『ActionScript 2.0 リファレンスガイド』の「Key」を参照してください。

カラー値の設定

ビルトインクラスである Color クラスのメソッドを使用すると、ムービークリップのカラーを調整できます。ColorTransform クラスの rgb プロパティを使って、16 進数の RGB (赤、緑、青) 値をムービークリップに割り当てます。次の例では、rgb を使用して、ユーザーによるクリックに基づいてオブジェクトのカラーを変更します。

ムービークリップのカラー値を設定するには :

1. 新しい Flash ドキュメントを作成し、「setrgb.fla」という名前で保存します。
2. 矩形ツールを選択し、ステージ上に大きな四角形を描画します。
3. その正方形をムービークリップシンボルに変換し、プロパティインスペクタで car_mc というインスタンス名を与えます。
4. colorChip という名前のボタンを作成します。そのボタンの 4 つのインスタンスをステージ上に配置し、それぞれ「red_btn」、「green_btn」、「blue_btn」、および「black_btn」という名前を付けます。
5. メインタイムラインのフレーム 1 を選択し、[ウィンドウ]-[アクション] を選択します。
6. メインタイムラインのフレーム 1 に次のコードを追加します。

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
var trans:Transform = new Transform(car_mc);
trans.colorTransform = colorTrans;
```

7. blue ボタンが押されたときに car_mc ムービークリップのカラーを青に変更するよう、次のコードを [アクション] パネルに追加します。

```
blue_btn.onRelease = function() {
  colorTrans.rgb = 0x333399; // 青
  trans.colorTransform = colorTrans;
};
```

上記のコードは、カラー変換オブジェクトの rgb プロパティを変更し、ボタンが押されるたびにカラー変換効果を mccar_ ムービークリップに再適用します。

8. 他の red_btn、green_btn、および black_btn に対しても手順 7 を繰り返し、ムービークリップを各ボタンに対応するカラーに変更します。

コードは次の例のようになります (新しい部分を太字で示します)。

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
var trans:Transform = new Transform(car_mc);
trans.colorTransform = colorTrans;
```

```
blue_btn.onRelease = function() {
  colorTrans.rgb = 0x333399; // 青
  trans.colorTransform = colorTrans;
};
red_btn.onRelease = function() {
  colorTrans.rgb = 0xFF0000; // 赤
  trans.colorTransform = colorTrans;
};
green_btn.onRelease = function() {
  colorTrans.rgb = 0x006600; // 緑
  trans.colorTransform = colorTrans;
};
black_btn.onRelease = function() {
  colorTrans.rgb = 0x000000; // 黒
  trans.colorTransform = colorTrans;
};
```

9. [制御]-[ムービープレビュー] を選択し、ムービークリップのカラーを変更します。

ColorTransform クラスのメソッドの詳細については、『ActionScript 2.0 リファレンスガイド』の「ColorTransform (flash.geom.ColorTransform)」を参照してください。

サウンドの制御

ビルトインクラスである Sound クラスを使用して、SWF ファイル内のサウンドを制御します。Sound クラスのメソッドを使用するには、まず Sound オブジェクトを作成する必要があります。次に、`attachSound()` メソッドを使用して、SWF ファイルの実行中にサウンドをライブラリから SWF ファイルに挿入します。

Sound クラスの `setVolume()` メソッドはボリュームの制御用、`setPan()` メソッドはサウンドの左右のバランスの調整用です。

以下はサウンド制御を作成するための手順です。

タイムラインにサウンドを割り当てるには :

1. [ファイル]-[読み込み]-[ライブラリに読み込み] を選択し、サウンドを読み込みます。
2. ライブラリでサウンドを選択し、右クリック (Windows) または Control キーを押しながらクリック (Macintosh) して、[リンケージ] を選択します。
3. [ActionScript に書き出し] と [最初のフレームに書き出し] を選択して、識別子 `a_thousand_ways` を割り当てます。
4. ステージにボタンを追加し、「`play_btn`」という名前を付けます。
5. ステージにボタンを追加し、「`stop_btn`」という名前を付けます。
6. メインタイムラインのフレーム 1 を選択し、[ウィンドウ]-[アクション] を選択します。

   [アクション] パネルに次のコードを追加します。

```
var song_sound:Sound = new Sound();
song_sound.attachSound("a_thousand_ways");
```

```
play_btn.onRelease = function() {
  song_sound.start();
};
stop_btn.onRelease = function() {
  song_sound.stop();
};
```

このコードでは、まず speaker ムービークリップが停止します。次に、新しい Sound オブジェクト (song_sound) を作成し、リンケージ識別子が a_thousand_ways であるサウンドを割り当てています。play_btn オブジェクトおよび stop_btn オブジェクトに関連付けられた onRelease イベントハンドラが、Sound.start() メソッドおよび Sound.stop() メソッドを使ってサウンドを開始および停止し、さらに、割り当てられたサウンドを再生および停止します。

7. [制御]-[ムービープレビュー] を選択し、サウンドを確認します。

スライドボリュームコントロールを作成するには :

1. 矩形ツールを使用して、ステージ上に高さ 30 ピクセル、幅 10 ピクセル程度の小さな矩形を描画します。
2. 選択ツールを選択し、ステージ上のシェイプをダブルクリックします。
3. F8 を押して [シンボルに変換] ダイアログボックスを開きます。
4. [ボタン] タイプを選択し、「volume」というシンボル名を入力し、[OK] をクリックします。
5. ステージでボタンシンボルを選択し、プロパティインスペクタで、インスタンス名として「handle_btn 」と入力します。
6. ボタンを選択し、[修正]-[シンボルに変換] を選択します。

   必ずムービークリップビヘイビアを選択してください。フレーム 1 にボタンがあるムービークリップが作成されます。
7. ムービークリップを選択し、プロパティインスペクタで、インスタンス名として「volume_mc 」と入力します。
8. メインタイムラインのフレーム 1 を選択し、[ウィンドウ]-[アクション] を選択します。
9. [アクション] パネルで、次のコードを入力します。

```
this.createTextField("volume_txt", 10, 30, 30, 200, 20);
volume_mc.top = volume_mc._y;
volume_mc.bottom = volume_mc._y;
volume_mc.left = volume_mc._x;
volume_mc.right = volume_mc._x + 100;
volume_mc._x += 100;

volume_mc.handle_btn.onPress = function() {
  startDrag(this._parent, false, this._parent.left, this._parent.top,
  this._parent.right, this._parent.bottom);
};
volume_mc.handle_btn.onRelease = function() {
  stopDrag();
  var level:Number = Math.ceil(this._parent._x - this._parent.left);
  this._parent._parent.song_sound.setVolume(level);
```

```
    this._parent._parent.volume_txt.text = level;
  };
  volume_mc.handle_btn.onReleaseOutside = slider_mc.handle_btn.onRelease;
```

startDrag() パラメータの left、top、right、および bottom は、ムービークリップアクションで設定される変数です。

10. [制御]-[ムービープレビュー] を選択し、ボリュームスライダをテストします。

バランススライドコントロールを作成するには :

1. 矩形ツールを使用して、ステージ上に高さ 30 ピクセル、幅 10 ピクセル程度の小さな矩形を描画します。
2. 選択ツールを選択し、ステージ上のシェイプをダブルクリックします。
3. F8 を押して [シンボルに変換] ダイアログボックスを開きます。
4. [ボタン] タイプを選択し、balance というシンボル名を入力し、[OK] をクリックします。
5. ステージでボタンシンボルを選択し、プロパティインスペクタで、インスタンス名として「handle_btn 」と入力します。
6. ボタンを選択し、[修正]-[シンボルに変換] を選択します。

   必ずムービークリップビヘイビアを選択してください。フレーム 1 にボタンがあるムービークリップが作成されます。
7. ムービークリップを選択し、プロパティインスペクタで、インスタンス名として「balance_mc 」と入力します。
8. [アクション] パネルで、次のコードを入力します。

```
balance_mc.top = balance_mc._y;
balance_mc.bottom = balance_mc._y;
balance_mc.left = balance_mc._x;
balance_mc.right = balance_mc._x + 100;
balance_mc._x += 50;
balance_mc.handle_btn.onPress = function() {
  startDrag(this._parent, false, this._parent.left, this._parent.top,
  this._parent.right, this._parent.bottom);
};
balance_mc.handle_btn.onRelease = function() {
  stopDrag();
  var level:Number = Math.ceil((this._parent._x - this._parent.left - 50) *
  2);
  this._parent._parent.song_sound.setPan(level);
};
balance_mc.handle_btn.onReleaseOutside = balance_mc.handle_btn.onRelease;
```

startDrag() パラメータの left、top、right、および bottom は、ムービークリップアクションで設定される変数です。

9. [制御]-[ムービープレビュー] を選択し、バランススライダをテストします。

Sound クラスのメソッドの詳細については、『ActionScript 2.0 リファレンスガイド』の「Sound」を参照してください。

衝突の判定

MovieClip クラスの hitTest() メソッドを使って、SWF ファイル内の衝突を判定することができます。このメソッドは、オブジェクトがムービークリップと衝突したかどうかを調べ、ブール値 (true または false) を返します。

衝突が発生したかどうかを知ることにより、ユーザーがステージ上のある静止領域に達したかどうかをテストしたり、一方のムービークリップが他方に到達した時点を調べたりすることができます。hitTest() により、これらの結果を判別することができます。

hitTest() のパラメータを使用すると、ステージ上のヒット領域の *x* 座標および *y* 座標を指定することや、別のムービークリップのターゲットパスをヒット領域として指定できます。*x* および *y* を指定すると、(*x*, *y*) で識別されたポイントが不透明ポイントの場合に、hitTest() は true を返します。ターゲットが hitTest() に渡されると、2 つのムービークリップの境界ボックスが比較されます。境界ボックスが交差する場合、hitTest() は true を返します。境界ボックスが交差しない場合、hitTest() は false を返します。

2 つのムービークリップ間の衝突は、hitTest() を使用してテストすることもできます。

次の例は、ステージ上のマウスとムービークリップの間の衝突を検出する方法を示しています。

ムービークリップとカーソルの間の衝突を検出するには :

1. タイムラインでレイヤー 1 の最初のフレームを選択します。
2. [アクション] パネルが表示されていない場合は、[ウィンドウ]-[アクション] を選択して表示します。
3. [アクション] パネルに次のコードを追加します。

```
this.createEmptyMovieClip("box_mc", 10);
with (box_mc) {
  beginFill(0xFF0000, 100);
  moveTo(100, 100);
  lineTo(200, 100);
  lineTo(200, 200);
  lineTo(100, 200);
  lineTo(100, 100);
  endFill();
}

this.createTextField("status_txt", 999, 0, 0, 100, 22);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  status_txt.text = _level0.hitTest(_xmouse, _ymouse, true);
}
Mouse.addListener(mouseListener);
```

4. [制御]-[ムービープレビュー] を選択し、ムービークリップ上をポインタを移動させ、衝突をテストします。

   ポインタが不透明なピクセル上にあるときは、常に true の値が表示されます。

2 つのムービークリップの衝突検出を実行するには :

1. 2 つのムービークリップをステージにドラッグし、それぞれのインスタンス名を car_mc および area_mc と付けます。
2. タイムラインでフレーム 1 を選択します。
3. [アクション] パネルが表示されていない場合は、[ウィンドウ]-[アクション] を選択して表示します。
4. [アクション] パネルで、次のコードを入力します。

```
this.createTextField("status_txt", 999, 10, 10, 100, 22);
area_mc.onEnterFrame = function() {
  status_txt.text = this.hitTest(car_mc);
};

car_mc.onPress = function() {
  this.startDrag(false);
  updateAfterEvent();
};
car_mc.onRelease = function() {
  this.stopDrag();
};
```

5. [制御]-[ムービープレビュー] を選択し、ムービークリップをドラッグして衝突検出をテストします。

   車の境界ボックスが領域の境界ボックスと交差する場合、ステータスは true となります。

詳細については、『ActionScript 2.0 リファレンスガイド』の「hitTest (MovieClip.hitTest メソッド)」を参照してください。

簡単な線画ツールの作成

MovieClip クラスのメソッドを使用して、SWF ファイルの再生時にステージで線と塗りを描画することができます。これによって、ユーザー用の描画ツールを作成することや、イベントに応じて SWF ファイル内にシェイプを描画することができます。描画メソッドとしては、beginFill()、beginGradientFill()、clear()、curveTo()、endFill()、lineTo()、lineStyle()、および moveTo() があります。

これらのメソッドは、任意のムービークリップインスタンス (例 : myClip.lineTo()) や、レベル (例 : _level0.curveTo()) に適用できます。

lineTo() メソッドでは直線を、curveTo() メソッドでは曲線を、それぞれ描画できます。直線や曲線の色、太さ、アルファ設定は、lineStyle() メソッドで指定できます。また、moveTo() メソッドを使うと、現在の描画位置を、指定した x と y のステージ座標に移動できます。

beginFill() メソッドは閉じたパスを単色で、beginGradientFill() メソッドはグラデーションで塗りつぶすことができます。また、endFill() メソッドは、beginFill() または beginGradientFill() の最後の呼び出しで指定した塗りつぶしを適用します。clear() メソッドは、指定したムービークリップオブジェクトに描画した内容を消去します。

簡単な線画ツールを作成するには :

1. 新規ドキュメントで、ステージ上にボタンを作成し、プロパティインスペクタでインスタンス名を `clear_btn` と指定します。
2. タイムラインでフレーム 1 を選択します。
3. [アクション] パネルが表示されていない場合は、[ウィンドウ]-[アクション] を選択して表示します。
4. [アクション] パネルに次のコードを入力します。

```
this.createEmptyMovieClip("canvas_mc", 999);
var isDrawing:Boolean = false;
//
clear_btn.onRelease = function() {
  canvas_mc.clear();
};
//
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  canvas_mc.lineStyle(5, 0xFF0000, 100);
  canvas_mc.moveTo(_xmouse, _ymouse);
  isDrawing = true;
};
mouseListener.onMouseMove = function() {
  if (isDrawing) {
    canvas_mc.lineTo(_xmouse, _ymouse);
    updateAfterEvent();
  }
};
mouseListener.onMouseUp = function() {
  isDrawing = false;
};
Mouse.addListener(mouseListener);
```

5. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。
6. ポインタをドラッグしてステージ上に線を描画します。
7. ボタンをクリックすると描画内容が消去されます。

ActionScript を使用したランタイムデータバインディングの作成

コンポーネントを使用してアプリケーションを作成する場合、データを操作したり、またはコンポーネント間のインタラクションを可能にしたりするために、コンポーネント間にバインディングを追加することがしばしば必要になります。コンポーネント間のインタラクションは、ユーザーが操作できる使いやすいフォームまたはインターフェイスを作成するために必要です。[コンポーネントインスペクタ] パネルの [バインディング] タブを使用して、ステージ上のコンポーネント間にバインディングを追加できます。

[バインディング] タブの使用の詳細については、『Flash ユーザーガイド』を参照してください。オンライン記事「Building a Tip of the day Application (Part 2)」、「Data Binding in Macromedia Flash MX Professional 2004」、および「Building a Google Search Application with Macromedia Flash MX Professional」からも追加情報を得ることができます。

[バインディング] タブの代わりに、ActionScript を使用してコンポーネント間のバインディングを作成することができます。コードを追加する方法は、多くの場合、オーサリング環境による方法よりも高速で効率的です。コードを使用してアプリケーションにコンポーネントを追加するときには、ActionScript を使用してバインディングを作成する必要があります。`createClassObject()` メソッドを使用すると、ステージにコンポーネントを動的に追加できますが、[バインディング] タブを使用してバインディングを作成することはできません。これは、そのコンポーネントが実行時まで存在しないためです。ActionScript を使用してデータバインディングを追加する方法は、" リアルタイムデータバインディング " と呼ばれることがよくあります。

詳細については、次のトピックを参照してください。

- ActionScript を使用した UI コンポーネント間のバインディングの作成
- 524 ページの「コンポーネント、バインディング、カスタムフォーマッタの使用」
- 527 ページの「ステージ上のコンポーネントの追加とバインディング」

ActionScript を使用した UI コンポーネント間のバインディングの作成

実行時に 2 つのコンポーネントの間でデータをバインドするのは難しいことではなく、DataBindingClasses コンポーネントには操作に必要なクラスが含まれているため、ドキュメントに含める必要があることに注意してください。

ActionScript を使用して 2 つの TextInput コンポーネント間のバインディングを作成するには :

1. 新しい Flash ドキュメントを「panel_as.fla」という名前で作成します。
2. TextInput コンポーネントの 2 つのコピーをステージにドラッグします。
3. それぞれのコンポーネントに in_ti および out_ti というインスタンス名を付けます。
4. [ウィンドウ]-[サンプルライブラリ]-[クラス] を選択し、Classes.fla という名前の新しいサンプルライブラリを開きます。
5. DataBindingClasses コンポーネントのコピーを [ライブラリ] パネルにドラッグするか、そのコンポーネントをステージにドラッグして、削除します。

   これを実行した後は、サンプルライブラリを閉じてかまいません。DataBindingClasses コンポーネントをステージから削除した後も、ライブラリにはコピーが残ります。

ヒント	DataBindingClasses コンポーネントをステージから削除するのを忘れると、実行時にコンポーネントのアイコンが表示されます。

メモ	前の例でコンポーネントインスペクタを使用してバインディングを作成したとき、DataBindingClasses コンポーネントが FLA ファイルに自動的に追加されています。ActionScript を使用してデータバインディングを作成するときは、次の手順により、そのクラスを手動でライブラリにコピーする必要があります。

6. 新規レイヤーを挿入し、actions という名前を付けます。
7. actions レイヤーのフレーム 1 に次の ActionScript を追加します。

```
var src:mx.data.binding.EndPoint = new mx.data.binding.EndPoint();
src.component = in_ti;
src.property = "text";
src.event = "focusOut";
var dest:mx.data.binding.EndPoint = new mx.data.binding.EndPoint();
dest.component = out_ti;
dest.property = "text";
new mx.data.binding.Binding(src, dest);
```

   コードが短いほうが望ましい場合は、上記に代えて、バインディングクラスを読み込んで次のコードを使用することもできます。

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = in_ti;
src.property = "text";
src.event = "focusOut";
var dest:EndPoint = new EndPoint();
dest.component = out_ti;
dest.property = "text";
new Binding(src, dest);
```

この ActionScript は、2 つのデータバインディングの終点を作成します。終点はそれぞれバインドしているコンポーネントに対応しています。最初の終点は、どのコンポーネントからバインドしているのか (in_ti)、どのプロパティを監視するのか (text)、およびどのイベントがバインディングをトリガするのか (focusOut) を定義します。2 番目の終点は、コンポーネントとプロパティ (それぞれ out_ti と text) だけを一覧で示します。最後に、Binding クラスのコンストラクタを呼び出すときに、2 つの終点間にバインディングを作成します (new Binding(src, dest))。

最初のコード例のように、ActionScript で完全修飾クラス名 (mx.data.binding.EndPoint など) を使用する必要はありません。コードの先頭で import ステートメントを使用すると、完全修飾名を使用しないようにすることができます。mx.data.binding パッケージのすべてのクラスをワイルドカード (*) を使用して読み込むと (このパッケージは EndPoint クラスと Binding クラスを含みます)、コードを短縮し、EndPoint クラスと Binding クラスを直接参照することができます。import ステートメントの詳細については、『ActionScript 2.0 リファレンスガイド』の「import」の項を参照してください。

8. [制御]-[ムービープレビュー] を選択してテスト環境でコードをテストします。in_ti テキスト入力フィールドに何かテキストを入力します。

   in_ti インスタンスが (ステージをクリックする操作、Tab キーを押す操作、または 2 番目のフィールドをクリックする操作により) フォーカスを失うと、in_ti に入力されたテキストが Flash により out_ti テキストフィールドにコピーされます。

9. [ファイル]-[保存] を選択して変更内容を保存します。

前の実習で使用した out_ti テキスト入力フィールドのテキストを変更すると、コードが非常に複雑なものになる可能性があります。[コンポーネントインスペクタ] パネルを使用してバインディングを設定する場合、デフォルトでは双方向接続が作成されます。つまり、ステージ上でいずれか一方のテキストフィールドを変更すると、もう一方のテキストフィールドも変更されます。ActionScript を使用してバインディングを作成するときは、アプリケーションはこれとは反対の動きをします。ランタイムデータバインディングは、次の例のように、特に指定しない限りデフォルトでは単方向になります。

ActionScript を使用して双方向バインディングを作成するには、前の例のコードを若干変更する必要があります。この例では、手順 7 で 2 番目に示した短い ActionScript コードを使用しています。

双方向バインディングを作成するには :

1. 前の例の "panel_as.fla" を開きます。
2. 次のコードに合わせて ActionScript (**太字**で示した部分) を若干変更します。

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = in_ti;
src.property = "text";
src.event = "focusOut";
var dest:EndPoint = new EndPoint();
dest.component = out_ti;
```

```
dest.property = "text";
dest.event = "focusOut";
new Binding(src, dest, null, true);
```

この ActionScript に対する 2 つの変更は、次の処理を実行するためのものです。

- ターゲットの EndPoint インスタンスのイベントプロパティを定義する。
- Binding コンストラクタの 2 つの追加パラメータを定義する。

高度なフォーマットオプションの最初のパラメータを使用します。この値は、`null` または `undefined` に設定できます。2 番目のパラメータは、バインディングが双方向 (`true`) か単方向 (`false`) かを定義します。

`focusOut` イベントがどこで発生するのかについては、該当する部分の ActionScript が複雑なため、わかりにくくなっています。TextInput クラスを調査して、一覧に示されたメソッド (`change()` や `enter()` など) をいくつか使用しても、`focusOut` イベントに関する処理は見つかりません。TextInput クラスは UIObject クラスおよび UIComponent クラスを継承しています。コンポーネントにフォーカスのサポートを追加するのは UIComponent クラスなので、このクラスを読んでみると、4 つのイベント、`focusIn`、`focusOut`、`keyDown`、および `keyUp` を見つけることができます。TextInput コンポーネントでは、これらのイベントを使用できます。

3. (オプション) 前の例で `out_ti` テキスト入力フィールドの値を更新したい場合は、イベントを `focusOut` から `change` に変更します。
4. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   Flash によって `in_ti` テキスト入力フィールドの 2 番目の値が変更され、`out_ti` の値が更新されます。これで双方向接続が正常に作成されました。

Binding クラスは、TextInput コンポーネントだけでなく、Adobe Component Architecture のほとんどのバージョン 2 ユーザーインターフェイスコンポーネントで使用できます。次の例は、ActionScript を使用して、実行時に CheckBox インスタンスと Label コンポーネントをバインドする方法を示しています。

CheckBox コンポーネントでバインディングクラスを使用するには :

1. 新しい Flash ドキュメントを作成します。
2. [ファイル]-[保存] を選択し、新規ファイルに「checkbox_as.fla」という名前を付けます。
3. [ウィンドウ]-[サンプルライブラリ]-[クラス] を選択します。
4. DataBindingClasses クラスのコピーをドキュメントのライブラリにドラッグします。
5. CheckBox コンポーネントのコピーをステージまでドラッグし、my_ch というインスタンス名を付けます。
6. Label コンポーネントのコピーをステージまでドラッグし、my_lbl というインスタンス名を付けます。
7. 新しいレイヤーを作成し、「actions」という名前を付けます。

8. actions レイヤーのフレーム 1 に次の ActionScript を追加します。

```
var srcEndPoint:Object = {component:my_ch, property:"selected",
  event:"click"};
var destEndPoint:Object = {component:my_lbl, property:"text"};
new mx.data.binding.Binding(srcEndPoint, destEndPoint);
```

この項の前の実習のように EndPoint クラスの新しいインスタンスを作成する代わりに、オブジェクトを使用して終点を定義します。この手順のコード例では、バインディングの終点の働きをする 2 つのオブジェクトを作成しています。Binding クラスのコンストラクタを呼び出すときにバインディングを作成します。コードの量をさらに減らすには (読みやすさも低下します)、次の例のように、オブジェクトをインラインで定義します。

```
new mx.data.binding.Binding({component:my_ch, property:"selected",
  event:"click"}, {component:my_lbl, property:"text"});
```

この ActionScript ではコードの読みやすさが低下しますが、ユーザーが入力するべき量も少なくなります。FLA (または ActionScript) ファイルを共有している場合、読みやすいことから ActionScript の最初の例を使用することをお勧めします。

コンポーネント、バインディング、カスタムフォーマッタの使用

カスタムフォーマッタは、特定の方法で複雑なデータをフォーマットするときに役立ちます。イメージ、HTML フォーマットのテキスト、およびその他のコンポーネントを DataGrid などのコンポーネント内で表示するときにもカスタムフォーマットを使用できます。カスタムフォーマッタが有用な場合の例を次に示します。

ドキュメント内でカスタムフォーマッタを使用するには :

1. 新規の FLA ファイルを作成し、DataBindingClasses クラスをライブラリに追加します ([ウィンドウ]-[サンプルライブラリ]-[クラス])。
2. DateChooser コンポーネントのコピーをステージまでドラッグし、my_dc というインスタンス名を付けます。
3. Label コンポーネントのコピーをステージまでドラッグし、my_lbl というインスタンス名を付けます。
4. 新規レイヤーを挿入し、actions という名前を付けます。
5. actions レイヤーのフレーム 1 に次の ActionScript コードを追加します。

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = my_dc;
src.property = "selectedDate";
src.event = "change";
var dest:EndPoint = new EndPoint();
```

```
dest.component = my_lbl;
dest.property = "text";
new Binding(src, dest);
```

このコードにより、DateChooser の selectedDate プロパティと、ステージ上の Label コンポーネントの text プロパティの間にバインディングが作成されます。カレンダーの新しい日付をクリックするたびに、選択された日付が Label コンポーネントに表示されます。

6. Flash ドキュメントをハードディスクの適当な場所に「customformat.fla」という名前で保存します。

   このファイルは次の実習で再利用します。

7. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   Calendar コンポーネントの日付を変更すると、現在選択されている日付が Label コンポーネントに表示されます。Label コンポーネントは幅が狭く、日付全体を表示できないため、テキストは切り捨てられます。

8. テスト SWF ファイルを閉じてオーサリング環境に戻ります。

   ステージ上の Label コンポーネントのサイズを変更するか、Label コンポーネントを選択し、プロパティインスペクタの [パラメータ] タブで autoSize プロパティを left に設定します。

9. [制御]-[ムービープレビュー] を選択してもう一度ドキュメントをテストします。

   今度はテキストフィールドに日付全体が表示されますが、読み取りにくく、フォーマットもされていません。タイムゾーンと選択された日付によって異なりますが、日付は "Thu Nov 4 00:00:00 GMT-0800 2004" のように表示されます。

   バインディングが正しく機能し、selectedDate プロパティが表示されてはいますが、これらの日付は決してユーザーフレンドリーなものではありません。ユーザーにとって必要でない情報 (タイムゾーンのオフセットや、場合によっては時、分、秒) が表示されるのは望ましくありません。そこで、読みやすく、あまり機械的な印象を与えないような形式に日付をフォーマットする方法が必要となります。カスタムフォーマッタは、テキストをフォーマットするときに特に有効です。

CustomFormatter クラスを使用したデータのフォーマット

CustomFormatter クラスでは format() と unformat() の 2 つのメソッドが定義されています。これらは、データ値を特定のデータ型から String 型に、またはその逆に変換するためのメソッドです。デフォルトでは、これらのメソッドは何もしません。mx.data.binding.CustomFormatter のサブクラスでこれらのメソッドを実装する必要があります。CustomFormatter クラスを使用すると、各種データ型からストリングへの変換、およびその逆の変換を実行できます。このケースでは、Label コンポーネントに値がコピーされたときに、DateChooser コンポーネントの selectedDate プロパティを適切にフォーマットされたストリングに変換します。

次の例は、日付を、デフォルトの日付ストリングではなく、NOV 4, 2004 として表示するカスタムフォーマッタを作成する方法を示しています。

メモ	この実習を開始する前に、524 ページの「コンポーネント、バインディング、カスタムフォーマッタの使用」を完了しておく必要があります。

CustomFormatter クラスを使用してデータをフォーマットするには :

1. [ファイル]-[新規] を選択し、次に [ActionScript ファイル] を選択して新規 AS ファイルを作成します。
2. [ファイル]-[保存] を選択し、新規ファイルを「DateFormat.as」という名前で保存します。
3. スクリプトウィンドウで、次のコードを入力します。

```
class DateFormat extends mx.data.binding.CustomFormatter {
   function format(rawValue:Date):String {
      var returnValue:String;
      var monthName_array:Array =
   ["JAN","FEB","MAR","APR","MAY","JUN","JUL","AUG","SEP","OCT","NOV","DEC"];
      returnValue = monthName_array[rawValue.getMonth()]+"
   "+rawValue.getDate()+", "+rawValue.getFullYear();
      return returnValue;
   }
}
```

コードの最初のセクションでは、DateFormat という名前の新規クラスが定義されており、これは、mx.data.binding パッケージの CustomFormatter クラスを拡張するものです。DataBindingClasses コンポーネントファイルのバインディングクラスは Flash によってコンパイルされるため、それらのクラスを直接表示することはできず、また、Flash のインストールディレクトリの "Classes" フォルダ内で見つけることもできません。

使用する唯一のメソッドは、日付インスタンスをカスタムストリングフォーマットに変換する format() メソッドです。次の手順では、最終結果をデフォルトの日付フォーマットから NOV 4, 2004 に近づけるために、月名の配列を作成します。Flash では配列はゼロから始まるため、rawValue.getMonth() が戻す値が 1 の場合は、1 月ではなく 2 月を表しています (1 月は月 0 になります)。コードの残りの部分では、値を連結し、returnValue ストリングを戻すことによって、カスタムフォーマットのストリングを作成します。

前の例に示されているように、コンパイルされたクリップ内でクラスを使用する際に問題が発生する場合があります。DataBindingClasses クラス内にあるクラスの 1 つを継承しているのに、それが Flash で直ちに利用可能にならないため、前のクラスのシンタックスをチェックすると次のエラーが発生します。

```
**Error** <path to DateFormat class>\DateFormat.as: Line 1: The class
   'mx.data.binding.CustomFormatter' could not be loaded.
      class DateFormat extends mx.data.binding.CustomFormatter {

Total ActionScript Errors: 1  Reported Errors: 1
```

このエラーが発生しても、作成したコードに問題があるとは限りません。これは Flash がクラスを見つけることができず、そのためにシンタックスのチェックがエラーになったことを示しています。

4. "DateFormat.as" ファイルを保存します。
5. 「コンポーネント、バインディング、カスタムフォーマッタの使用」の実習で使用した "customformat.fla" を開きます。"DateFormat.as" は、このファイルと同じディレクトリに保存またはコピーしてください。
6. "customformat.fla" では、actions レイヤーのフレーム 1 の ActionScript コードを次のコードのように変更します。

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = my_dc;
src.property = "selectedDate";
src.event = "change";
var dest:EndPoint = new EndPoint();
dest.component = my_lbl;
dest.property = "text";
new Binding(src, dest, {cls:mx.data.formatters.Custom,
   settings:{classname:"DateFormat", classname_class:DateFormat}});
```

今回定義する customFormatter オブジェクトは、ユーザーが新規に作成した DateFormat クラスを使用してバインディングの終点をフォーマットすることを Flash に伝えます。

7. 変更したドキュメントを保存して、[制御]-[ムービープレビュー] を選択し、コードをテストします。

ステージ上のコンポーネントの追加とバインディング

ActionScript でバインディングクラスを使用する最大のメリットは、Flash が実行時にステージに追加したコンポーネントの間にバインディングを作成できることです。実行時に適切なテキストフィールドをステージに追加し、必要なデータを検証し、必要なバインディングを追加するカスタムクラスを作成することを想像してみてください。コンポーネントがライブラリにあれば、それを動的に追加し、数行のコードを追加するだけでバインディングを作成できます。

ActionScript を使用してステージ上にコンポーネントを追加し、バインディングするには :

1. 新しい Flash ドキュメントを作成します。
2. ComboBox および Label コンポーネントをドキュメントのライブラリにドラッグします。
3. 新規レイヤーを挿入し、actions という名前を付けます。
4. actions レイヤーのフレーム 1 に次のコードを追加します。

```
import mx.data.binding.*;
this.createClassObject(mx.controls.ComboBox, "my_cb", 1, {_x:10, _y:10});
this.createClassObject(mx.controls.Label, "my_lbl", 2, {_x:10, _y:40});
my_cb.addItem("JAN", 0);
```

```
my_cb.addItem("FEB", 1);
my_cb.addItem("MAR", 2);
my_cb.addItem("APR", 3);
my_cb.addItem("MAY", 4);
my_cb.addItem("JUN", 5);
var src:EndPoint = new EndPoint();
src.component = my_cb;
src.property = "value";
src.event = "change";
var dest:EndPoint = new EndPoint();
dest.component = my_lbl;
dest.property = "text";
new Binding(src, dest);
```

ActionScript の最初の行で `mx.data.binding` パッケージからクラスを読み込んでいるため、コード内で完全修飾パスを使用する必要はありません。ActionScript の次の 2 行は、ドキュメントのライブラリからステージにコンポーネントをアタッチしています。次に、コンポーネントをステージ上に配置します。

最後に、ComboBox インスタンスにデータを追加し、ステージ上の `my_cb` ComboBox と `my_lbl` Label コンポーネントの間にバインディングを作成します。

サンプルスクリプトの解説

サンプルの "zapper.swf" ファイル ([Flash ヘルプ] で表示可能) では、ユーザーがテントウムシをコンセントにドラッグすると、テントウムシは落ち、コンセントは震えます。メインタイムラインにはフレームが 1 つだけあり、その中にはテントウムシ、コンセント、およびリセットボタンという 3 つのオブジェクトがあります。各オブジェクトはムービークリップインスタンスです。

次のスクリプトは、メインタイムラインのフレーム 1 にアタッチされます。

```
var initx:Number = bug_mc._x;
var inity:Number = bug_mc._y;
var zapped:Boolean = false;
```

```
reset_btn.onRelease = function() {
  zapped = false;
  bug_mc._x = initx;
  bug_mc._y = inity;
  bug_mc._alpha = 100;
  bug_mc._rotation = 0;
};

bug_mc.onPress = function() {
  this.startDrag();
};
bug_mc.onRelease = function() {
  this.stopDrag();
};
bug_mc.onEnterFrame = function() {
  if (this.hitTest(this._parent.zapper_mc)) {
    this.stopDrag();
    zapped = true;
    bug_mc._alpha = 75;
    bug_mc._rotation = 20;
    this._parent.zapper_mc.play();
  }
  if (zapped) {
    bug_mc._y += 25;
  }
};
```

テントウムシのインスタンス名は bug_mc であり、コンセントのインスタンス名は zapper_mc です。スクリプトで bug インスタンスが this と表されているのは、このスクリプトは bug インスタンスに割り当てられたものであり、予約語 this が bug インスタンスを含むオブジェクトという意味を表すからです。

イベントハンドラには、異なるイベントを扱う onRelease()、onPress()、および onEnterFrame() があります。これらのイベントハンドラは、SWF ファイルのロード後にフレーム 1 上で定義されます。再生ヘッドがフレームに入るたびに、onEnterFrame() イベントハンドラ内のアクションが実行されます。1 フレームの SWF ファイルであっても、再生ヘッドは繰り返しそのフレーム内に移動し、スクリプトは繰り返し実行されます。

2 つの変数 initx と inity は、bug_mc ムービークリップインスタンスの最初の *x* 座標および *y* 座標をそれぞれ格納するよう定義されています。また、関数が定義され、reset_btn インスタンスの onRelease イベントハンドラに割り当てられています。この関数は、reset_btn ボタン上でマウスボタンがクリックされて離されるたびに呼び出されます。この関数によって、テントウムシがステージ上の最初の位置に戻り、回転値とアルファ値がリセットされ、zapped 変数が false にリセットされます。

if 条件ステートメントでは、hitTest() メソッドを使用して、テントウムシのインスタンスがコンセントのインスタンス (this._parent.zapper_mc) に触れているかどうかをチェックします。評価の結果には、true と false があります。

- hitTest() メソッドによって true が返された場合、Flash により stopDrag() メソッドが呼び出され、zapped 変数が true に設定されます。また、alpha プロパティと rotation プロパティが変更されて、zapper_mc インスタンスに再生が指示されます。
- hitTest() メソッドによって false が返された場合は、if ステートメントの直後に記述された中カッコ ({}) 内のコードは実行されません。

bug_mc インスタンス上でマウスボタンを押すと、onPress() ステートメント内のアクションが実行されます。bug_mc インスタンス上でマウスボタンを離すと、onRelease() ステートメント内のアクションが実行されます。

startDrag() アクションで、テントウムシのドラッグを実現します。スクリプトが bug_mc インスタンスに割り当てられているので、キーワード this は、bug インスタンスがドラッグ可能であることを示しています。

```
bug_mc.onPress = function() {
  this.startDrag();
};
```

stopDrag() では、ドラッグの動作を停止します。

```
bug_mc.onRelease = function() {
  this.stopDrag();
};
```

第 14 章

イメージ、サウンドおよびビデオの操作

14

Flash CS3 Professional でのオーサリング時にイメージまたはサウンドをドキュメントに読み込んだ場合、パブリッシュ時にそのイメージおよびサウンドはパッケージ化され、SWF ファイルに格納されます。オーサリング時にメディアを読み込めるだけでなく、実行時に他の SWF ファイルなどの外部メディアをロードすることもできます。理由によっては、メディアを Flash ドキュメントの外部に置くことが必要な場合があります。

ファイルサイズの軽減　大きなメディアファイルを Flash ドキュメントの外部に置き、それを実行時にロードすることで、アプリケーションおよびプレゼンテーションの初期ダウンロード時間を短縮できます。これは、特に低速のインターネット接続を利用する際に有効です。

大きなプレゼンテーションのモジュール化　大きなプレゼンテーションやアプリケーションをいくつかの小さな SWF ファイルに分割し、それぞれのファイルを実行時に必要に応じてロードすることができます。このようにモジュール化することで初期ダウンロード時間を短縮することができ、プレゼンテーションの保守や更新が容易に行えるようになります。

コンテンツとプレゼンテーションの分離　これはアプリケーション開発、特にデータドリブンアプリケーションの開発における一般的なテーマです。たとえば、ショッピングカートアプリケーションで各製品のイメージを表示することがあります。各イメージを実行時にダウンロードするようにすれば、本体の FLA ファイルを修正しなくても、製品のイメージを簡単に更新できます。

実行時のみの機能の活用　Flash Video (FLV) および MP3 の動的ロードなど、実行時に ActionScript を通じてのみ利用可能な機能がいくつかあります。

この項では、Flash アプリケーションでのイメージファイル、サウンドファイル、および FLV ビデオの操作方法を説明します。詳細については、次のトピックを参照してください。

外部メディアのロードおよび操作について 532
外部 SWF ファイルおよびイメージファイルのロード 533
外部 MP3 ファイルのロードと使用について 537
ライブラリ内のアセットへのリンケージ割り当て 542
FLV ビデオの使用について. 543
メディアファイル用プログレスアニメーションの作成について 562

外部メディアのロードおよび操作について

実行時に、Flash アプリケーションにロードできるメディアファイルには、SWF、MP3、JPEG、GIF、PNG、FLV ファイルの 6 種類があります。しかし、Flash Player のすべてのバージョンがこれらの種類のメディアをサポートしているわけではありません。Flash Player 8 以降でサポートしているイメージファイルの種類については、533 ページの「外部 SWF ファイルおよびイメージファイルのロード」を参照してください。Flash Player での FLV ビデオサポートについては、543 ページの「FLV ビデオの使用について」を参照してください。

Flash Player では、外部メディアを任意の HTTP または FTP アドレスからロードすることや、相対パスを使用してローカルディスクからロードすることができます。また、`file://` プロトコルを使用する方法もあります。

外部の SWF ファイルとイメージファイルをロードするには、`loadMovie()` 関数または `loadMovieNum()` 関数を使用するか、`MovieClip.loadMovie()` メソッドまたは `MovieClipLoader.loadClip()` メソッドを使用します。クラスのメソッドは、一般的にグローバル関数よりも機能が豊富でより柔軟なため、複雑なアプリケーションに適しています。SWF ファイルまたはイメージファイルをロードするときには、そのメディアのターゲットとなるムービークリップまたは SWF ファイルレベルを指定します。SWF ファイルとイメージファイルのロード方法の詳細については、533 ページの「外部 SWF ファイルおよびイメージファイルのロード」を参照してください。

外部の MP3 ファイルを再生するには、Sound クラスの `loadSound()` メソッドを使用します。このメソッドでは、MP3 ファイルのプログレッシブダウンロードを行うか、ダウンロードが完了してから再生するかを指定できます。ID3 情報が利用可能な場合は、MP3 ファイル内に埋め込まれている ID3 情報を読み取ることもできます。詳細については、540 ページの「MP3 ファイル内の ID3 タグの読み取り」を参照してください。

Flash Video は、Flash Player で使用されるネイティブビデオ形式です。FLV ファイルは HTTP 経由で再生することも、ローカルファイルシステムから再生することもできます。Flash ドキュメント内にビデオを埋め込むよりも、外部の FLV ファイルを再生した方が、パフォーマンスやメモリ管理の効率改善、ビデオおよび Flash フレームレートの独立化などのメリットがあります。詳細については、545 ページの「外部 FLV ファイルのダイナミック再生」を参照してください。

また、MovieClipLoader クラスを使用して外部メディアをプリロードすることや、外部メディアのダウンロード進行状況を追跡することもできます。MovieClipLoader クラスは、SWF ファイルまたはイメージファイルのダウンロード進行状況の追跡に使用します。MP3 ファイルと FLV ファイルをプリロードするには、Sound クラスの `getBytesLoaded()` メソッドおよび NetStream クラスの `bytesLoaded` プロパティを使用します。詳細については、548 ページの「FLV ファイルのプリロード」を参照してください。

フォトギャラリアプリケーションのサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Galleries" フォルダに移動して次のサンプルにアクセスします。

- gallery_tree.fla
- gallery_tween.fla

これらのファイルは、イメージファイルの SWF ファイルへのダウンロード中に、ActionScript を使用してムービークリップを動的に制御する方法の例を示すものです。

外部 SWF ファイルおよびイメージファイルのロード

SWF ファイルまたはイメージファイルをロードするには、グローバル関数の `loadMovie()` または `loadMovieNum()` を使用するか、MovieClip クラスの `loadMovie()` メソッドまたは MovieClipLoader クラスの `loadClip()` メソッドを使用します。`loadClip()` メソッドの詳細については、『ActionScript 2.0 リファレンスガイド』の「`MovieClipLoader.loadClip()`」を参照してください。

Flash Player 8 以降でサポートされるイメージファイルは、JPEG イメージファイル (プログレッシブおよび非プログレッシブ)、GIF イメージ (透明および不透明、ただしアニメーション化された GIF は最初のフレームのみがロードされます)、および PNG ファイル (透明および不透明) です。

SWF ファイルまたはイメージファイルを Flash Player のあるレベルにロードする場合は、`loadMovieNum()` 関数を使用します。SWF ファイルまたはイメージファイルをムービークリップターゲットにロードする場合は、`loadMovie()` 関数またはメソッドを使用します。いずれの場合でも、ロードされたコンテンツが指定のレベルまたはターゲットムービークリップのコンテンツと置き替わります。

SWF ファイルまたはイメージファイルをムービークリップターゲットにロードした場合、その SWF ファイルまたはイメージの左上隅がムービークリップの基準点に配置されます。この基準点は通常はムービークリップの中心にあるので、ロードされたコンテンツが中央配置で表示されない場合があります。また、ルートタイムラインに SWF ファイルまたはイメージをロードすると、イメージの左上隅がステージの左上隅に配置されます。ロードされたコンテンツはムービークリップの回転と伸縮を継承しますが、ムービークリップの元のコンテンツは削除されます。

`loadMovie()` または `loadMovieNum()` を呼び出すときには、オプションで ActionScript 変数を渡すことができます。この方法は、これらのメソッド呼び出しに指定する URL が、Flash アプリケーションから渡されるデータに従って SWF ファイルまたはイメージファイルを返すサーバーサイドスクリプトである場合に便利です。

グローバル関数 loadMovie() または loadMovieNum() を使用するときは、ターゲットレベルまたはターゲットクリップをパラメータとして指定します。次の例では、"contents.swf" という Flash アプリケーションを image_mc という名前のムービークリップインスタンスにロードします。

```
loadMovie("contents.swf", image_mc);
```

次の MovieClip.loadMovie() メソッドを使用しても同じ結果を得られます。

```
image_mc.loadMovie("contents.swf");
```

次の例では、"image1.jpg" という JPEG イメージを image_mc というムービークリップインスタンスにロードします。

```
image_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

SWF ファイルおよびイメージファイルのロード方法の詳細については、537 ページの「SWF ファイルのロードとルートタイムラインについて」を参照してください。

SWF ファイルや JPEG ファイルをムービークリップインスタンスにプリロードするには、MovieClipLoader クラスを使用します。このクラスには、ムービークリップへのファイルのダウンロード状況を通知するためのイベントリスナーメカニズムが用意されています。MovieClipLoader オブジェクトを使用して SWF ファイルや JPEG ファイルをプリロードするには、次の手順を実行します。

新規 MovieClipLoader オブジェクトの作成　単一の MovieClipLoader オブジェクトを使用して、複数のファイルのダウンロード進行状況を追跡したり、個々のファイルの進行状況に応じて個別のオブジェクトを作成したりできます。新しいムービークリップを作成し、その中にコンテンツをロードしてから、次のコードに示されているように、MovieClipLoader オブジェクトを作成します。

```
this.createEmptyMovieClip("img_mc", 999);
var my_mcl:MovieClipLoader = new MovieClipLoader();
```

リスナーオブジェクトとイベントハンドラの作成　リスナーオブジェクトは、汎用 Object オブジェクト、ムービークリップ、カスタムコンポーネントなど、任意の ActionScript オブジェクトにすることができます。

次の例では、loadListener という名前の汎用リスナーオブジェクトを作成し、そのオブジェクトに対して onLoadError、onLoadStart、onLoadProgress、および onLoadComplete の各関数を定義します。

```
// リスナーオブジェクトを作成する :
var mclListener:Object = new Object();
mclListener.onLoadError = function(target_mc:MovieClip, errorCode:String,
   status:Number) {
   trace("Error loading image: " + errorCode + " [" + status + "]");
};
mclListener.onLoadStart = function(target_mc:MovieClip):Void {
   trace("onLoadStart: " + target_mc);
};
mclListener.onLoadProgress = function(target_mc:MovieClip,
   numBytesLoaded:Number, numBytesTotal:Number):Void {
   var numPercentLoaded:Number = numBytesLoaded / numBytesTotal * 100;
```

```
  trace("onLoadProgress: " + target_mc + " is " + numPercentLoaded + "%
  loaded");
};
mclListener.onLoadComplete = function(target_mc:MovieClip, status:Number):Void {
  trace("onLoadComplete: " + target_mc);
};
```

メモ

Flash Player 8 以降では、onLoadComplete および onLoadError イベントリスナーでの MovieClipLoader ダウンロードの HTTP ステータスをチェックできます。この機能を使用することで、ファイルのダウンロードが不可能であった理由、サーバーエラーかどうか、ファイルが見つからなかった理由などを確認できます。

MovieClipLoader オブジェクトへのリスナーオブジェクト登録　リスナーオブジェクトがロードイベントを受信するには、次のコードのように、MovieClipLoader オブジェクトに登録する必要があります。

```
my_mcl.addListener(mclListener);
```

ターゲットクリップへのファイル (JPEG または SWF) のロード開始　イメージファイルまたは SWF ファイルのダウンロードを開始するには、次のコードのように `MovieClipLoader.loadClip()` メソッドを使用します。

```
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg", img_mc);
```

メモ

MovieClipLoader メソッドは、`MovieClipLoader.loadClip()` メソッドでロードされたファイルのダウンロード進行状況を追跡する場合にだけ使用できます。`loadMovie()` 関数や `MovieClip.loadMovie()` メソッドは使用できません。

次の例では、SWF ファイルのダウンロード進行状況を表示するのに ProgressBar コンポーネントの `setProgress()` メソッドを使用しています詳細については、『ActionScript 2.0 リファレンスガイド』を参照してください。

ProgressBar コンポーネントを使用してダウンロード進行状況を表示するには :

1. 新しい Flash ドキュメントを作成し、「progress.fla」という名前を付けて保存します。
2. [コンポーネント] パネルを開きます ([ウィンドウ]-[コンポーネント])。
3. [コンポーネント] パネルからステージに ProgressBar コンポーネントをドラッグします。
4. プロパティインスペクタで ([ウィンドウ]-[プロパティ]-[プロパティ])、ProgressBar コンポーネントに `my_pb` という名前を付けます。
5. タイムラインでフレーム 1 を選択し、[アクション] パネルを開きます ([ウィンドウ]-[アクション])。
6. [アクション] パネルに次のコードを追加します。

```
var my_pb:mx.controls.ProgressBar;
my_pb.mode = "manual";

this.createEmptyMovieClip("img_mc", 999);
```

```
var my_mcl:MovieClipLoader = new MovieClipLoader();
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip):Void {
  my_pb.label = "loading: " + target_mc._name;
};
mclListener.onLoadProgress = function(target_mc:MovieClip,
  numBytesLoaded:Number, numBytesTotal:Number):Void {
  var pctLoaded:Number = Math.ceil(100 * (numBytesLoaded / numBytesTotal));
  my_pb.setProgress(numBytesLoaded, numBytesTotal);
};
my_mcl.addListener(mclListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

7. [制御]-[ムービープレビュー] を選択して、ドキュメントをテストします。

   ムービークリップ img_mc にイメージがロードされます。

8. [ファイル]-[パブリッシュ]-[形式] を選択し、SWF オプションおよび HTML オプションが選択されていることを確認します。

9. [パブリッシュ] をクリックして、ハードディスク内で HTML ファイルおよび SWF ファイルを検索します。

   これらのファイルは、手順 1 で保存した Flash ドキュメント progress.fla と同じフォルダにあります。

10. ブラウザ内で HTML ドキュメントをダブルクリックして開き、プログレスバーアニメーションを確認します。

メモ	テスト環境にファイルをロードするときに、プログレスバーに状況を表示したい場合は、ローカルファイルではなく、キャッシュされていないファイルをインターネットからロードしていることを確認してください。ローカルファイルは非常に速くロードされるので、進捗状況は表示されません。または、SWF ファイルをアップロードして、サーバー上でドキュメントをテストします。

詳細については、537 ページの「SWF ファイルのロードとルートタイムラインについて」を参照してください。MovieClipLoader クラスの詳細については、『ActionScript 2.0 リファレンスガイド』の「MovieClipLoader」を参照してください。プログレスバーアニメーション作成の詳細については、563 ページの「SWF ファイルおよびイメージファイルのロード用プログレスアニメーションの作成」を参照してください。

フォトギャラリアプリケーションのサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Galleries" フォルダに移動して次のサンプルにアクセスします。

- gallery_tree.fla
- gallery_tween.fla

これらのファイルは、イメージファイルの SWF ファイルへのダウンロード中に、ActionScript を使用してムービークリップを動的に制御する方法の例を示すものです。

SWF ファイルのロードとルートタイムラインについて

_root という ActionScript プロパティによって、SWF ファイルのルートタイムラインの参照が指定されるか、返されます。SWF ファイルを別の SWF ファイルのムービークリップにロードした場合、ロードされた SWF ファイル内の _root への参照は、そのロードされた SWF ファイルのルートタイムラインではなく、ホスト SWF ファイル内のルートタイムラインとして分割されます。これが実行時に予想外の動作を引き起こす場合があります。たとえば、ホスト SWF ファイルとロードされた SWF ファイルの両方が、変数を指定するときに _root を使用する場合などです。

Flash Player 7 以降では、_lockroot (MovieClip._lockroot property) プロパティを使用することで、ムービークリップによる _root への参照を、そのムービークリップを含んでいる SWF ファイルのタイムラインではなく、ムービークリップ自体のタイムラインへの参照に強制的に設定することができます。詳細については、305 ページの「ロードした SWF ファイルのルートタイムラインの指定」を参照してください。_root および _lockroot の使用の詳細については、645 ページ、第 17 章の「ActionScript 2.0 のベストプラクティスとコーディング規則」。を参照してください。

ある SWF ファイルは、インターネット上の任意の場所から別の SWF ファイルをロードできます。ただし、片方の SWF ファイルがもう一方の SWF ファイルに定義されているデータ (変数、メソッドなど) にアクセスするには、両方の SWF ファイルが同じドメインに存在する必要があります。Flash Player 7 以降では、ロードされる SWF ファイル側が System.security.allowDomain() を呼び出していない限り、ドメイン間にまたがるスクリプトを実行できません。

System.security.allowDomain については、『ActionScript 2.0 リファレンスガイド』の「allowDomain (security.allowDomain method)」および 627 ページの「ネットワーク API の制限」を参照してください。

外部 MP3 ファイルのロードと使用について

MP3 ファイルを実行時にロードするには、Sound クラスの loadSound() メソッドを使用します。次の例に示すように、まず Sound オブジェクトを作成します :

```
var song1_sound:Sound = new Sound();
```

この新しいオブジェクトを使用して loadSound() を呼び出し、イベントサウンドまたはストリーミングサウンドをロードします。イベントサウンドは、ロードが完了してから再生されます。ストリーミングサウンドは、ダウンロードと同時に再生されます。loadSound() の isStreaming パラメータを使用して、サウンドをイベントサウンドまたはストリーミングサウンドに指定できます。イベントサウンドをロードした後は、サウンドの再生を開始するために Sound クラスの start() メソッドを呼び出す必要があります。ストリーミングサウンドは、ある程度のデータが SWF ファイルにロードされた時点で再生を開始するので、start() を使用する必要はありません。

たとえば次のコードでは、my_sound という名前の Sound オブジェクトが作成され、"song1.mp3" という MP3 ファイルがロードされます。タイムラインのフレーム 1 に次の ActionScript を入力します。

```
var my_sound:Sound = new Sound();
my_sound.loadSound("http://www.helpexamples.com/flash/sound/song1.mp3", true);
```

ほとんどの場合、isStreaming パラメータを true に設定します。特に、大きなサウンドファイルをロードして、(MP3 の " ジュークボックス " アプリケーションを作成する場合など) できる限り早く再生を開始したい場合に、true に設定します。ただし、短いサウンドクリップをダウンロードしておき、(ユーザーがボタンをクリックしたときなど) 特定のタイミングで再生する必要がある場合には、isStreaming を false に設定します。

サウンド IS のダウンロードが完了したことを確認するには、Sound.onLoad イベントハンドラを使用します。このイベントハンドラは、ファイルのダウンロードが正常に完了したかどうかを示すブール値 (true または false) を自動的に受け取ります。

詳細については、次のトピックを参照してください。

- 538 ページの「MP3 ファイルのロード」
- 539 ページの「MP3 ファイルのプリロード」
- 540 ページの「MP3 ファイル内の ID3 タグの読み取り」

MP3 ファイルをロードするサンプルのソースファイル "jukebox.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ComponentsAS2/Jukebox" フォルダに移動してサンプルにアクセスします。このサンプルは、データ型、一般的なコーディングの原則、および複数のコンポーネントを使用してジュークボックスを作成する方法を示しています。

MP3 ファイルのロード

ユーザーの到達レベルに応じて異なるサウンドを使うオンラインゲームを作成している場合を想定してみます。次のコードでは、game_sound という Sound オブジェクトに MP3 ファイル ("song2.mp3") をロードし、IS のダウンロードが完了したときにそのサウンドを再生します。

MP3 ファイルをロードするには :

1. loadMP3.fla という新しい FLA ファイルを作成します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
var game_sound:Sound = new Sound();
game_sound.onLoad = function(success:Boolean):Void {
   if (success) {
      trace("Sound Loaded");
      game_sound.start();
   }
};
game_sound.loadSound("http://www.helpexamples.com/flash/sound/song2.mp3",
   false);
```

3. [制御]-[ムービープレビュー] を選択してサウンドをテストします。

実行時のサウンドファイルのロードについては、Flash Player では MP3 サウンドファイルタイプだけがサポートされています。

詳細については、『ActionScript 2.0 リファレンスガイド』の「Sound.loadSound()」、「Sound.start()」、および「Sound.onLoad」を参照してください。MP3 ファイルのロードについては、539 ページの「MP3 ファイルのプリロード」を参照してください。サウンドファイルロード時のプログレスバーアニメーション作成については、565 ページの「ActionScript による MP3 ファイルのロード用プログレスバーの作成」を参照してください。

MP3 ファイルをロードするサンプルのソースファイル "jukebox.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ComponentsAS2/Jukebox" フォルダに移動してサンプルにアクセスします。このサンプルは、データ型、一般的なコーディングの原則、および複数のコンポーネントを使用してジュークボックスを作成する方法を示しています。

MP3 ファイルのプリロード

MP3 ファイルをプリロードする場合、setInterval() 関数を使用して、Sound オブジェクトまたは NetStream オブジェクトにロードされたバイト数を定期的にチェックする " ポーリング " メカニズムを作成できます。MP3 ファイルのダウンロード進行状況を追跡するには、Sound.getBytesLoaded() および Sound.getBytesTotal() メソッドを使用します。

次の例では、setInterval() を使用して、既定の間隔で Sound オブジェクトにロードされるバイト数をチェックします。

MP3 ファイルをプリロードするには :

1. preloadMP3.fla という新しい FLA ファイルを作成します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
// サウンドを再生する Sound オブジェクトを新しく作成する
var songTrack:Sound = new Sound();
// ダウンロード進行状況を追跡するポーリング関数を作成する
// この関数がポーリングされるこの関数では、
// 参照として渡された Sound オブジェクトのダウンロード進行状況がチェックされる
function checkProgress (soundObj:Object):Void {
  var numBytesLoaded:Number = soundObj.getBytesLoaded();
  var numBytesTotal:Number = soundObj.getBytesTotal();
  var numPercentLoaded:Number = Math.floor(numBytesLoaded / numBytesTotal *
  100);
  if (!isNaN(numPercentLoaded)) {
    trace(numPercentLoaded + "% loaded.");
  }
};
```

```
// ファイルのロードが完了したら、定期ポーリングをクリアする
songTrack.onLoad = function ():Void {
   trace("load complete");
   clearInterval(poll);
};
// ストリーミング MP3 ファイルをロードし、checkProgress() 呼び出しを開始する
songTrack.loadSound("http://www.helpexamples.com/flash/sound/song1.mp3",
   true);
var poll:Number = setInterval(checkProgress, 100, songTrack);
```

3. [制御]-[ムービープレビュー] を選択してサウンドをテストします。

   [出力] パネルにはロード進行状況が表示されます。

このポーリングテクニックを使用して、外部 FLV ファイルをプリロードできます。FLV ファイルに対してロードされた合計バイト数と現在のバイト数を取得するには、`NetStream.bytesLoaded` プロパティと `NetStream.bytesTotal` プロパティを使用します。詳細については、「bytesLoaded (NetStream.bytesLoaded プロパティ)」および「bytesTotal (NetStream.bytesTotal プロパティ)」を参照してください。

詳細については、『ActionScript 2.0 リファレンスガイド』の「`MovieClip.getBytesLoaded()`」、「`MovieClip.getBytesTotal()`」、「`setInterval()`」、「`Sound.getBytesLoaded()`」、および「`Sound.getBytesTotal()`」を参照してください。

プログレスバーアニメーション作成の詳細については、565 ページの「ActionScript による MP3 ファイルのロード用プログレスバーの作成」を参照してください。

MP3 ファイルをロードするサンプルのソースファイル "jukebox.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ComponentsAS2/Jukebox" フォルダに移動してサンプルにアクセスします。このサンプルは、データ型、一般的なコーディングの原則、および複数のコンポーネントを使用してジュークボックスを作成する方法を示しています。

MP3 ファイル内の ID3 タグの読み取り

ID3 タグは MP3 ファイルに追加されるデータフィールドです。ID3 タグには、曲名、アルバム名、およびアーティスト名など、ファイルに関する情報が含まれています。

MP3 ファイル内の ID3 タグを読み取るには、`Sound.id3` プロパティを使用します。このオブジェクトのプロパティは、ロードする MP3 ファイル内の ID3 タグ名に対応しています。ダウンロード中の MP3 ファイルの ID3 タグが使用可能になったことを検出するには、`Sound.onID3` イベントハンドラを使用します。Flash Player 7 は、バージョン 1.0、1.1、2.3、および 2.4 のタグをサポートしています。バージョン 2.2 タグはサポートしていません。

次の例では、song1.mp3 という名前の MP3 ファイルを `song_sound` という Sound オブジェクトにロードします。このファイルの ID3 タグが使用可能な場合、`display_txt` というテキストフィールドにアーティスト名と曲名が表示されます。

MP3 ファイルから ID3 タグを読み取るには :

1. id3.fla という新しい FLA ファイルを作成します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
this.createTextField("display_txt", this.getNextHighestDepth(), 0, 0, 100,
  100);
display_txt.autoSize = "left";
display_txt.multiline = true;
var song_sound:Sound = new Sound();
song_sound.onLoad = function() {
  song_sound.start();
};
song_sound.onID3 = function():Void  {
  display_txt.text += "Artist:\t" + song_sound.id3.artist + "\n";
  display_txt.text += "Song:\t" + song_sound.id3.songname + "\n";
};
song_sound.loadSound("http://www.helpexamples.com/flash/sound/song1.mp3");
```

3. [制御]-[ムービープレビュー] を選択してサウンドをテストします。

   ID3 タグがステージに表示され、サウンドが再生されます。

ID3 2.0 のタグは MP3 ファイルの先頭 (サウンドデータの前) に置かれているので、ファイルのダウンロードが始まるとすぐに使用可能になります。しかし、ID3 1.0 のタグはファイルの末尾 (サウンドデータの後) に置かれているので、MP3 ファイル全体のダウンロードが完了するまで使用可能になりません。

`onID3` イベントハンドラは、新しい ID3 データが使用可能になるたびに呼び出されます。そのため、MP3 ファイルに ID3 2.0 タグと ID3 1.0 タグが含まれている場合は、タグがファイル内の別々の場所に置かれているので、`onID3` ハンドラが 2 回呼び出されることになります。

サポートされている ID3 タグの一覧については、『ActionScript 2.0 リファレンスガイド』の「`id3 (Sound.id3 プロパティ )`」を参照してください。

MP3 ファイルをロードするサンプルのソースファイル "jukebox.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ComponentsAS2/Jukebox" フォルダに移動してサンプルにアクセスします。このサンプルは、データ型、一般的なコーディングの原則、および複数のコンポーネントを使用してジュークボックスを作成する方法を示しています。

ライブラリ内のアセットへのリンケージ割り当て

ムービークリップやフォントシンボルなど、ライブラリ内のアセットにリンケージ識別子を割り当てることができます。Flash では、ライブラリ内のサウンドアセットおよびイメージアセットに対してリンケージ識別子を設定できます。そのため、共有ライブラリや新しい BitmapData クラスによるイメージファイルやサウンドファイルを使用することができます。

次の例では、リンケージが myImage に設定されたライブラリにビットマップイメージが追加されます。次に、そのイメージをステージに追加し、ドラッグ可能にします。

ビットマップファイルとのリンケージを使用するには :

1. linkBitmap.fla という新しい FLA ファイルを作成します。
2. ビットマップイメージをライブラリに読み込みます。
3. ライブラリでイメージを右クリックするか (Windows)、Control キーを押しながらクリックし (Macintosh)、コンテキストメニューから [リンケージ] を選択します。
4. [ActionScript に書き出し] と [最初のフレームに書き出し] を選択し、[識別子] テキストボックスに「myImage」と入力します。
5. [OK] をクリックして、リンケージ識別子を設定します。
6. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
import flash.display.BitmapData;
// imageBmp を作成し、ライブラリからビットマップを割り当てる
var imageBmp:BitmapData = BitmapData.loadBitmap("myImage");
// ムービークリップを作成し、imageBmp を割り当てる
this.createEmptyMovieClip("imageClip", 10);
imageClip.attachBitmap(imageBmp, 2);
// ムービークリップをドラッグ可能にする
imageClip.onPress = function() {
     this.startDrag();
};
imageClip.onRelease = function() {
     this.stopDrag();
}
```

7. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

   ライブラリ内のビットマップがステージに表示され、イメージがドラッグ可能になります。

FLV ビデオの使用について

FLV ファイル形式には、Flash Player を使って配信するエンコード済みのオーディオデータやビデオデータが含まれています。たとえば、QuickTime や Windows Media ビデオファイルの場合、エンコーダ (Flash 8 Video Encoder、Sorensen Squeeze など) を使用して、ファイルを FLV ファイルに変換できます。

Flash Player 7 では、Sorenson Spark ビデオコーデックでエンコードされた FLV ファイルをサポートしています。Flash Player 8 以降では、Sorenson Spark または Flash の On2 VP6 エンコーダでエンコードされた FLV ファイルをサポートしています。On2 VP6 ビデオコーデックはアルファチャンネルをサポートしています。Flash Player のバージョンが異なると、FLV のサポート方法も異なります。詳細については、次の表を参照してください。

コーデック	SWF ファイルバージョン (パブリッシュバージョン)	再生に必要な Flash Player バージョン
Sorenson Spark	6	6 以降
	7	7 以降
On2 VP6	6	8* 以降
	7	8 以降
	8 以降	8 以降

* SWF ファイルに FLV ファイルがロードされる場合、ユーザーが Flash Player 8 以降を使用して SWF ファイルを表示している間は On2 VP6 ビデオを使用できますが、Flash Player 8 以降用の SWF ファイルを再度パブリッシュする必要があります。Flash Player 8 以降では、On2 VP6 ビデオのパブリッシュと再生がサポートされます。

ストリーミング、プログレッシブダウンロード、サイズ、エンコード、読み込み、および帯域幅などビデオに関する基本情報については、『Flash ユーザーガイド』を参照してください。

この項では、コンポーネントのない FLV ビデオの使用について説明します。また、FLVPlayback コンポーネントを使用して FLV ファイルを再生することや、VideoPlayback クラスを使用して、FLV ファイルを動的にロードするカスタムビデオプレーヤーを作成することもできます。詳細については、www.adobe.com/devnet/flash/ または www.adobe.com/support/documentation/ を参照してください。FLVPlayback コンポーネントや Media コンポーネントによる FLV ビデオの使用については、『ActionScript 2.0 リファレンスガイド』の「FLVPlayback コンポーネントと Media コンポーネント」のセクションを参照してください。

Flash のオーサリング環境にビデオを直接読み込む代わりに、ActionScript を使用して、外部の FLV ファイルを Flash Player 内でダイナミックに再生できます。FLV ファイルは HTTP アドレスからでも、ローカルファイルシステムからでも再生できます。FLV ファイルを再生するには、NetConnection クラスおよび NetStream クラスと、Video クラスの `attachVideo()` メソッドを使用します。詳細については、『ActionScript 2.0 リファレンスガイド』の「`NetConnection`」、「`NetStream`」、および「`attachVideo (Video.attachVideo method)`」を参照してください。

FLV ファイルを作成するには、ビデオを Flash のオーサリングツールに読み込み、それを FLV ファイルとして書き出します Flash があれば、FLV 書き出しプラグインを使用して、サポート対象のビデオ編集アプリケーションから FLV ファイルを書き出すことができます。

外部 FLV ファイルを使うと、読み込んだビデオでは使用できない次のような機能を使用できます。

- 再生の速度を落とすことなく、Flash ドキュメント内でより長いビデオクリップを使用できます。外部 FLV ファイルは " キャッシュメモリ " を使用して再生されます。つまり、大きなファイルが小さなパーツとして格納され、ダイナミックにアクセスされるので、埋め込みビデオファイルよりもメモリ消費が少なくなります。
- 外部 FLV ファイルには、それを再生する Flash ドキュメントとは異なるフレームレートを持たせることができます。たとえば、Flash ドキュメントのフレームレートを毎秒 30 フレーム (fps) に設定する一方で、ビデオのフレームレートを 21 fps に設定できます。この設定により、埋め込みビデオに比べ、ビデオが制御しやすくなり、再生が滑らかになります。また、既存の Flash コンテンツを変更することなく、異なるフレームレートで FLV ファイルを再生することが可能になります。
- 外部 FLV ファイルを使用すると、ビデオファイルのロード中に Flash ドキュメントの再生を中断せずに済みます。読み込んだビデオファイルは、CD-ROM ドライブへのアクセスなど特定の機能を実行するためにドキュメントの再生を中断させることができます。外部 FLV ファイルは Flash ドキュメントからは独立して実行されるので、再生を遮ることはありません。
- 外部 FLV ファイルでは、ビデオのメタデータにアクセスするのにイベントハンドラを使用できるため、ビデオのコンテンツに容易にキャプションを付けることができます。

ヒント	Web サーバーから FLV ファイルをロードするには、ファイル拡張子と MIME タイプの Web サーバーへの登録が必要な場合があります。ご使用の Web サーバーのマニュアルを参照してください。FLV ファイルの MIME タイプは video/x-flv です。詳細については、561 ページの「FLV ファイルに合わせたサーバーの設定」を参照してください。

FLV ビデオの詳細については、次のトピックを参照してください。

- 545 ページの「ビデオオブジェクトの作成」
- 545 ページの「外部 FLV ファイルのダイナミック再生」
- 546 ページの「ビデオバナーの作成」
- 548 ページの「FLV ファイルのプリロード」
- 550 ページの「キューポイントの操作」
- 559 ページの「メタデータの操作」
- 561 ページの「FLV ファイルに合わせたサーバーの設定」
- 562 ページの「Macintosh 上のローカル FLV ファイルのターゲット設定について」

ビデオオブジェクトの作成

ActionScript を使用してビデオをロードし、操作するには、ビデオオブジェクトを作成して、それをステージまでドラッグし、インスタンス名を付ける必要があります。次の例では、アプリケーションへのビデオインスタンスの追加方法を説明します。

ビデオオブジェクトを作成するには :

1. Flash オーサリングツールでドキュメントを開き、[ライブラリ] パネル ([ウィンドウ]-[ライブラリ]) のポップアップメニューから [新規ビデオ] を選択します。
2. [ビデオプロパティ] ダイアログボックスで、ビデオシンボルに名前を付け、[ビデオ] (ActionScript により制御) を選択します。
3. [OK] をクリックして、ビデオオブジェクトを作成します。
4. ビデオオブジェクトを [ライブラリ] パネルからステージにドラッグして、ビデオオブジェクトインスタンスを作成します。
5. ステージ上でビデオオブジェクトを選択し、プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) の [インスタンス名] テキストボックスに「my_video」と入力します。

   これでステージ上にビデオインスタンスが作成され、これに ActionScript を追加してビデオをロードしたり、あるいはさまざまな方法でインスタンスを操作したりできます。

FLV ファイルの動的ロードについては、「外部 FLV ファイルのダイナミック再生」を参照してください。ビデオバナーの作成については、546 ページの「ビデオバナーの作成」を参照してください。

外部 FLV ファイルのダイナミック再生

実行時に FLV ファイルをロードし、SWF ファイルで再生することができます。FLV ファイルはビデオオブジェクトや、FLVPlayback コンポーネントなどのコンポーネントにロードできます。次の例は、"clouds.flv" というファイルをビデオオブジェクトで再生する方法を示したものです。

外部 FLV ファイルを Flash ドキュメント内で再生するには :

1. playFLV.fla という名前の新しい Flash ドキュメントを作成します。
2. [ライブラリ] パネル ([ウィンドウ]-[ライブラリ]) の [ライブラリ] ポップアップメニューから [新規ビデオ] を選択します。
3. [ビデオプロパティ] ダイアログボックスで、ビデオシンボルに名前を付け、[ビデオ] (ActionScript により制御) を選択します。
4. [OK] をクリックして、ビデオオブジェクトを作成します。
5. ビデオオブジェクトを [ライブラリ] パネルからステージにドラッグして、ビデオオブジェクトインスタンスを作成します。

6. ステージ上でビデオオブジェクトを選択し、プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) の [インスタンス名] テキストボックスに「my_video」と入力します。
7. タイムラインでフレーム 1 を選択し、[アクション] パネルを開きます ([ウィンドウ]-[アクション])。
8. [アクション] パネルで、次のコードを入力します。

```
this.createTextField("status_txt", 999, 0, 0, 100, 100);
status_txt.autoSize = "left";
status_txt.multiline = true;
// NetConnection オブジェクトを作成する
var my_nc:NetConnection = new NetConnection();
// ローカルストリーミング接続を作成する
my_nc.connect(null);
// NetStream オブジェクトを作成し、onStatus() 関数を定義する
var my_ns:NetStream = new NetStream(my_nc);
my_ns.onStatus = function(infoObject:Object):Void {
   status_txt.text += "status (" + this.time + " seconds)\n";
   status_txt.text += "\t Level: " + infoObject.level + "\n";
   status_txt.text += "\t Code: " + infoObject.code + "\n\n";
};
// NetStream ビデオフィードを Video オブジェクトに割り当てる
my_video.attachVideo(my_ns);
// バッファ時間を設定する
my_ns.setBufferTime(5);
// FLV ファイルの再生を開始する
my_ns.play("http://www.helpexamples.com/flash/video/clouds.flv");
```

9. [制御]-[ムービープレビュー] を選択してドキュメントをテストします。

FLV ファイルのプリロードについては、507 ページの「FLV ファイルのプリロード」を参照してください。コンポーネントへの FLV ビデオの動的ロードについては、『コンポーネントリファレンスガイド』を参照してください。FLV ファイルおよびサーバーについてや、Macintosh 上の FLV ファイルおよび FLV ファイルのローカル再生については、561 ページの「FLV ファイルに合わせたサーバーの設定」を参照してください。

ビデオバナーの作成

バナーやその他の Flash 広告のビデオコンテンツは、Flash ムービープレビューやテレビ広告などに広く利用されています。次の例は、ビデオインスタンスの作成方法と、FLA ファイルへの ActionScript を追加して、ビデオコンテンツを持つバナー広告を作成する方法を説明しています。

ビデオバナーを作成するには :

1. vidBanner.fla という新しい Flash ドキュメントを作成します。
2. [修正]-[ドキュメント] を選択します。
3. FLA ファイルのサイズを変更し、[幅] テキストボックスに「468」、[高さ] テキストボックスに「60」と入力します。

4. [ライブラリ] パネルで ([ウィンドウ]-[ライブラリ])、ライブラリオプションから [新規ビデオ] を選択します。
5. [ビデオプロパティ] ダイアログボックスで、ビデオシンボルに名前を付け、[ビデオ] (ActionScript により制御) を選択します。
6. [OK] をクリックして、ビデオオブジェクトを作成します。
7. ビデオオブジェクトを [ライブラリ] パネルからステージにドラッグして、ビデオインスタンスを作成します。
8. ステージ上でビデオオブジェクトを選択し、プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) の [インスタンス名] テキストボックスに「my_video」と入力します。
9. ビデオインスタンスを選択したまま、プロパティインスペクタの [幅] テキストボックスに「105」、[高さ] テキストボックスに「60」と入力します。
10. ビデオインスタンスをステージまでドラッグするか、プロパティインスペクタを使って、ビデオインスタンスの *x* 座標および *y* 座標を設定します。
11. タイムラインでフレーム 1 を選択し、[アクション] パネルを開きます ([ウィンドウ]-[アクション])。
12. [アクション] パネルに次のコードを追加します。

```
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.setBufferTime(5);
my_ns.play("http://www.helpexamples.com/flash/video/vbanner.flv");
```

13. [挿入]-[タイムライン]-[レイヤー] を選択し、新規フォルダを作成し、button という名前を付けます。
14. [ツール] パネルから矩形ツールを選択します。
15. [ツール] パネルの [カラー] セクションで、[鉛筆] アイコンをクリックして [線のカラー] コントロールを選択します。
16. [カラーなし] を選択すると、矩形のアウトラインが無効になります。
17. ステージ上でポインタを対角線に沿ってドラッグして、矩形を作成します。

    矩形の大きさはプロパティインスペクタを使って自由に変更できるので問題ではありません。
18. [ツール] パネルで選択ツールをクリックし、次にステージ上の矩形をクリックして選択します。
19. 矩形を選択したまま、プロパティインスペクタの [幅] テキストボックスに「468」、[高さ] テキストボックスに「60」と入力します。次に X 座標と Y 座標 ([X] テキストボックスと [Y] テキストボックス) を 0 に変更します。
20. ステージ上で矩形を選択したまま、F8 キーを押して矩形をシンボルに変換します。
21. [シンボルに変換] ダイアログボックスで、[名前] テキストボックスに「invisible btn」と入力して [OK] をクリックします。

22. ステージで新しいボタンをダブルクリックし、シンボル編集モードに設定します。

    現在、矩形は作成したボタンの最初の [アップ] フレーム上にあります。これが、ボタンがステージ上にあるときに表示される、ボタンの [アップ] 状態です。ただし、ボタンをステージ上に表示しない場合は、矩形を [ヒット] フレームまで移動する必要があります。このフレームはボタンのヒット領域 (ユーザーがクリックするとボタンのアクションが有効になるアクティブ領域) です。

23. [アップ] フレームでキーフレームをクリックし、マウスボタンを押したままキーフレームを [ヒット] フレームまでドラッグします。

    これで、バナー領域全体でクリックが有効となりますが、バナー上にボタンは表示されません。

24. シーン 1 をクリックして、メインタイムラインに戻ります。

    非表示のボタンのヒット領域を示す緑青の矩形がバナー領域内に表示されます。

25. 作成したボタンを選択し、プロパティインスペクタを開き、[インスタンス名] テキストボックスに「inv_btn」と入力します。

26. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
inv_btn.onRelease = function(){
   getURL("http://www.adobe.com");
};
```

27. グラフィックやテキストなど、バナーに対して他の修正を実行します。

28. [制御]-[ムービープレビュー] を選択して、Flash Player でバナーをテストします。

この例ではバナーを作成し、そのサイズを、Interactive Advertising Bureau の指定による、既に確定済みの標準サイズに変更します。標準広告サイズ (その他の有用なガイドライン) については、www.iab.net/standards/adunits.asp にある Interactive Advertising Bureau の「Standards & Guidelines」ページを参照してください。

標準ガイドラインに関わりなく、広告サービス、顧客、または最初に広告を実行する Web サイトの広告ガイドラインを確認してください。バナーを広告会社に提出する場合、ファイルが所定のファイルサイズ、大きさ、ターゲット Flash Player バージョン、およびフレームレートガイドラインに適合していることを確認します。さらに、使用するメディアの種類や、FLA ファイルで使用するボタンコードに関する規則も考慮することが必要な場合があります。

FLV ファイルのプリロード

FLV ファイルのダウンロード進行状況を追跡するには、`NetStream.bytesLoaded` プロパティと `NetStream.bytesTotal` プロパティを使用します。FLV ファイルに対してロードされた合計バイト数と現在のバイト数を取得するには、`NetStream.bytesLoaded` プロパティと `NetStream.bytesTotal` プロパティを使用します。

次の例では、my_video というビデオオブジェクトインスタンスに "video1.flv" がロードされる状況を表示する `bytesLoaded` プロパティと `bytesTotal` プロパティを使用します。"loaded_txt" というテキストフィールドが動的に作成され、ロードの進行状況に関する情報を表示します。

FLV ファイルをプリロードするには :

1. preloadFLV.fla という新しい FLA ファイルを作成します。
2. [ライブラリ] パネル ([ウィンドウ]-[ライブラリ]) の [ライブラリ] ポップアップメニューから [新規ビデオ] を選択します。
3. [ビデオプロパティ] ダイアログボックスで、ビデオシンボルに名前を付け、[ビデオ] (ActionScript により制御) を選択します。
4. [OK] をクリックして、ビデオオブジェクトを作成します。
5. ビデオオブジェクトを [ライブラリ] パネルからステージにドラッグして、ビデオオブジェクトインスタンスを作成します。
6. ステージ上でビデオオブジェクトを選択し、プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) の [インスタンス名] テキストボックスに「my_video」と入力します。
7. ビデオインスタンスを選択したまま、プロパティインスペクタの [幅] テキストボックスに「320」、[高さ] テキストボックスに「213」と入力します。
8. タイムラインでフレーム 1 を選択し、[アクション] パネルを開きます ([ウィンドウ]-[アクション])。
9. [アクション] パネルで、次のコードを入力します。

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("http://www.helpexamples.com/flash/video/lights_short.flv");

this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10, 160,
  22);
var loaded_interval:Number = setInterval(checkBytesLoaded, 500, stream_ns);
function checkBytesLoaded(my_ns:NetStream) {
  var pctLoaded:Number = Math.round(my_ns.bytesLoaded / my_ns.bytesTotal *
  100);
  loaded_txt.text = Math.round(my_ns.bytesLoaded / 1000) + " of " +
  Math.round(my_ns.bytesTotal / 1000) + " KB loaded (" + pctLoaded + "%)";
  progressBar_mc.bar_mc._xscale = pctLoaded;
  if (pctLoaded >= 100) {
    clearInterval(loaded_interval);
  }
}
```

10. [制御]-[ムービープレビュー] を選択してコードをテストします。

メモ	プログレスバーが簡単にロードできる場合、(この例のテストか、別の手順によるロードで) ビデオは既にハードディスク上にキャッシュされています。ビデオがキャッシュされたら、FLV ファイルをサーバーにアップロードして、代わりにロードします。

FLV ファイルをプリロードするもう 1 つの方法は、NetStream.setBufferTime( ) メソッドを使用することです。このメソッドは、再生開始前に FLV ストリームのバッファにかかる秒数を表す 1 つのパラメータを取ります。詳細については、『ActionScript 2.0 リファレンスガイド』の「setBufferTime (NetStream.setBufferTime メソッド)」、「getBytesLoaded (MovieClip.getBytesLoaded メソッド)」、「getBytesTotal (MovieClip.getBytesTotal メソッド)」、「bytesLoaded (NetStream.bytesLoaded プロパティ)」、「bytesTotal (NetStream.bytesTotal プロパティ)」、および「setInterval 関数」を参照してください。

キューポイントの操作

Flash Video では数種類のキューポイントを使用できます。ActionScript を使用することで、(FLV ファイル作成時に) FLV ファイルに埋め込んだキューポイントや ActionScript を使用して作成したキューポイントを制御できます。

ナビゲーションキューポイント　FLV ファイルをエンコードするときに、FLV ストリームや FLV メタデータパケットにナビゲーションキューポイントを埋め込みます。ナビゲーションキューポイントにより、ファイルの所定の部分をシークできます。

イベントキューポイント　FLV ファイルをエンコードするときに、FLV ストリームや FLV メタデータパケットにイベントキューポイントを埋め込みます。FLV 再生中に所定のポイントでトリガされるイベントを取り扱うコードを作成できます。

ActionScript キューポイント　ActionScript コードを使用して作成する外部キューポイント。ビデオ再生に関連して、これらのキューポイントをトリガするコードを作成できます。ビデオプレーヤーはこれらのキューポイントを個別に追跡するため、埋め込まれたキューポイントに比べ、これらのキューポイントは精度が劣ります (最高 10 分の 1 秒)。

ナビゲーションキューポイントにより、指定されたキューポイント位置にキーフレームが作成されるため、コードを使ってビデオプレーヤーの再生ヘッドをその位置まで移動することができます。FLV ファイルでユーザーにシークさせたい特定のポイントを設定できます。たとえば、複数の章ないしセグメントがあるビデオの場合、ビデオファイルにナビゲーションキューポイントを埋め込むことでそのビデオを制御することが可能です。

ユーザーがキューポイントまで移動するようなアプリケーションを作成する場合、ActionScript キューポイントは使用せず、キューポイントを作成しファイルに埋め込む必要があります。動作精度が上がるため、キューポイントは FLV ファイルに埋め込みます。キューポイントによる FLV ファイルのエンコードの詳細については、『Flash ユーザーガイド』の「キューポイントの操作」を参照してください。

ActionScript を書くことで、キューポイントパラメータにアクセスできます。キューポイントパラメータは、cuePoint イベント (event.info.parameters) と一緒に受け取るイベントオブジェクトの一部です。キューポイントのアクセスおよび追跡については、『Flash ユーザーガイド』の「キューポイントの操作」を参照してください。

FLV ファイルからのキューポイントの追跡

NetStream.onMetaData を使用して FLV ドキュメントに埋め込まれたキューポイントを追跡することができます。キューポイント情報確認のために返されるメタデータの構造を反復する必要があります。

次のコードで、FLV ファイルのキューポイントを追跡することができます。

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
stream_ns.onMetaData = function(metaProp:Object) {
  trace("The metadata:");
  traceMeta(metaProp);
  // traceObject(metaProp, 0);
};
my_video.attachVideo(stream_ns);
stream_ns.play("http://www.helpexamples.com/flash/video/cuepoints.flv");

function traceMeta(metaProp:Object):Void {
  var p:String;
  for (p in metaProp) {
    switch (p) {
    case "cuePoints" :
      trace("cuePoints: ");
      // キューポイントを循環する
      var cuePointArr:Array = metaProp[p];
      for (var j:Number = 0; j < cuePointArr.length; j++) {
        // 現在のキューポイントパラメータを循環する
        trace("\t cuePoints[" + j + "]:");
        var currentCuePoint:Object = metaProp[p][j];
        var metaPropPJParams:Object = currentCuePoint.parameters;
        trace("\t\t name: " + currentCuePoint.name);
        trace("\t\t time: " + currentCuePoint.time);
        trace("\t\t type: " + currentCuePoint.type);
        if (metaPropPJParams != undefined) {
          trace("\t\t parameters:");
          traceObject(metaPropPJParams, 4);
        }
      }
      break;
    default :
      trace(p + ": " + metaProp[p]);
      break;
    }
  }
}
```

```
function traceObject(obj:Object, indent:Number):Void {
  var indentString:String = "";
  for (var j:Number = 0; j < indent; j++) {
    indentString += "\t";
  }
  for (var i:String in obj) {
    if (typeof(obj[i]) == "object") {
      trace(indentString + " " + i + ": [Object]");
      traceObject(obj[i], indent + 1);
    } else {
      trace(indentString + " " + i + ": " + obj[i]);
    }
  }
}
```

上記の結果は以下のように出力されます。

```
The metadata:
canSeekToEnd: true
cuePoints:
  cuePoints[0]:
    name: point1
    time: 0.418
    type: navigation
    parameters:
      lights: beginning
  cuePoints[1]:
    name: point2
    time: 7.748
    type: navigation
    parameters:
      lights: middle
  cuePoints[2]:
    name: point3
    time: 16.02
    type: navigation
    parameters:
      lights: end
audiocodecid: 2
audiodelay: 0.038
audiodatarate: 96
videocodecid: 4
framerate: 15
videodatarate: 400
height: 213
width: 320
duration: 16.334
```

FLVPlayback コンポーネントによるキューポイントの使用については、「FLVPlayback コンポーネントによる埋め込みキューポイントの使用」を参照してください。

FLVPlayback コンポーネントによる埋め込みキューポイントの使用

FLVPlayback コンポーネントを使用するときに、プロパティインスペクタに FLV ファイルのキューポイントを表示できます。FLVPlayback インスタンスに contentPath プロパティを設定した後、ビデオファイルに埋め込まれたすべてのキューポイントを表示できます。[パラメータ] タブを使用して、cuePoints プロパティを探し、拡大鏡アイコンをクリックして、ファイルにあるキューポイントの一覧を表示します。

メモ	キューポイントが [パラメータ] タブにあることを確認するには、コードを使用する代わりに FLV ファイルの名前を [contentPath] テキストボックスに入力して contentPath を割り当てる必要があります。

次の例では、FLVPlayback コンポーネントでのキューポイントの使用方法を説明します。

FLVPlayback コンポーネントでキューポイントを使用するには :

1. cueFlv.fla という新しい Flash ドキュメントを作成します。
2. [コンポーネント] パネル ([ウィンドウ]-[コンポーネント]) を開き、FLVPlayback および TextArea コンポーネントのインスタンスをステージにドラッグします。
3. TextArea コンポーネントを選択し、プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) の [インスタンス名] テキストボックスに「my_ta」と入力します。
4. TextArea コンポーネントを選択したまま、[幅] テキストボックスに「200」、[高さ] テキストボックスに「100」と入力します。
5. ステージで FLVPlayback インスタンスを選択し、[インスタンス名] テキストボックスに「my_flvPb」と入力します。
6. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
var my_flvPb:mx.video.FLVPlayback;
var my_ta:mx.controls.TextArea;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
  my_ta.text += "Elapsed time in seconds: " + my_flvPb.playheadTime + "\n";
};
my_flvPb.addEventListener("cuePoint",listenerObject);
```

7. [制御]-[ムービープレビュー] を選択して SWF ファイルをテストします。

   ドキュメントに埋め込まれたキューポイントを再生ヘッドが通過するたびに、経過時間が TextArea インスタンスに表示されます。

FLVPlayback コンポーネントの操作の詳細については、『ActionScript 2.0 リファレンスガイド』を参照してください。

コンポーネントと一緒に使用するための ActionScript によるキューポイントの作成

ActionScript を使ってキューポイントを作成し、ビデオオブジェクトインスタンスやビデオプレーヤーコンポーネント (FLVPlayback for Flash Player 8 以降または MediaPlayback for Flash Player 7) の 1 つと共に使用することができます。次の例では、ActionScript コードを使ったキューポイントの作成や、スクリプトを使ったキューポイントへのアクセスがどれだけ簡単に実行できるかを示しています。

メモ	アプリケーションにナビゲーション機能を追加する場合は、ナビゲーションキューポイントをドキュメントに埋め込みます。詳細については、550 ページの「キューポイントの操作」を参照してください。埋め込んだキューポイントの操作の例については、553 ページの「FLVPlayback コンポーネントによる埋め込みキューポイントの使用」を参照してください。

FLVPlayback コンポーネントでキューポイントを作成し、使用するには :

1. cueFlvPb.fla という新しい Flash ドキュメントを作成します。
2. FLVPlayback コンポーネントのインスタンスを、[コンポーネント] パネル ([ウィンドウ]-[コンポーネント]) からステージにドラッグします。

   このコンポーネントは "FLVPlayback - Player 8" フォルダにあります。
3. コンポーネントを選択し、プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) を開きます。
4. [インスタンス名] テキストボックスに「my_flvPb」と入力します。
5. TextArea コンポーネントのインスタンスを、[コンポーネント] パネルからステージにドラッグします。
6. TextArea コンポーネントを選択し、[インスタンス名] テキストボックスに「my_ta」と入力します。
7. TextArea コンポーネントを選択したまま、[幅] テキストボックスに「200」、[高さ] テキストボックスに「100」と入力します。
8. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
var my_flvPb:mx.video.FLVPlayback;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/clouds.flv";

// cuePoint オブジェクトを作成する。
var cuePt:Object = new Object();
cuePt.time = 1;
cuePt.name = "elapsed_time";
cuePt.type = "actionscript";
// AS キューポイントを追加する。
my_flvPb.addASCuePoint(cuePt);

// 別の AS キューポイントを追加する。
my_flvPb.addASCuePoint(2, "elapsed_time2");
```

```
// テキストフィールドにキューポイント情報を表示する。
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject) {
   my_ta.text += "Elapsed time in seconds: " + my_flvPb.playheadTime + "\n";
};
my_flvPb.addEventListener("cuePoint",listenerObject);
```

9. [制御]-[ムービープレビュー] を選択してコードをテストします。

   次のキューポイントにより、[出力] パネルで追跡が実行されます。

```
Elapsed time in seconds: 1.034
Elapsed time in seconds: 2.102
```

キューポイントと FLVPlayback コンポーネントの操作の詳細については、『ActionScript 2.0 リファレンスガイド』を参照してください。

次の例は、実行時のキューポイント追加方法と、FLV ファイルが MediaPlayback コンポーネントで再生されているときのキューポイントの追跡方法を示しています。

MediaPlayback コンポーネントでキューポイントを作成し、使用するには :

1. cuePointMP.fla という新しい Flash ドキュメントを作成します。
2. MediaPlayback コンポーネントのインスタンスを、[コンポーネント] パネル ([ウィンドウ]-[コンポーネント]) からステージにドラッグします。

   このコンポーネントは "Media - Player 6 - 7" フォルダにあります。
3. コンポーネントを選択し、プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) を開きます。
4. [インスタンス名] テキストボックスに「my_mp」と入力します。
5. [パラメータ] タブを選択し、[コンポーネントインスペクタの起動] をクリックします。
6. コンポーネントインスペクタで、[URL] テキストボックスに「http://www.helpexamples.com/flash/video/clouds.flv」と入力します。
7. [アクション] パネルを開き ([ウィンドウ]-[アクション])、スクリプトペインに次のコードを入力します。

```
import mx.controls.MediaPlayback;
var my_mp:MediaPlayback;
my_mp.autoPlay = false;
my_mp.addEventListener("cuePoint", doCuePoint);
my_mp.addCuePoint("one", 1);
my_mp.addCuePoint("two", 2);
my_mp.addCuePoint("three", 3);
my_mp.addCuePoint("four", 4);
function doCuePoint(eventObj:Object):Void {
     trace(eventObj.type + " = {cuePointName:" + eventObj.cuePointName + "
   cuePointTime:" + eventObj.cuePointTime + "}");
}
```

8. [制御]-[ムービープレビュー] を選択してコードをテストします。

   次のキューポイントにより、[出力] パネルで追跡が実行されます。

```
cuePoint = {cuePointName:one cuePointTime:1}
cuePoint = {cuePointName:two cuePointTime:2}
cuePoint = {cuePointName:three cuePointTime:3}
cuePoint = {cuePointName:four cuePointTime:4}
```

MediaPlayback コンポーネントの操作の詳細については、『ActionScript 2.0 リファレンスガイド』を参照してください。

キューポイントによるシーク機能の追加

FLV ファイルにナビゲーションキューポイントを埋め込んで、アプリケーションにシーク機能を追加することができます。FLVPlayback コンポーネントの `seekToNavCuePoint()` メソッドによって、キューポイントに指定の名前が与えられ、所定のタイミングないしそれ以降に FLV ファイル内に配置されます。名前をストリングとして (`"part1"` や `"theParty"` など) 指定できます。

現在の `playheadTime` を基準にして、次のナビゲーションキューポイントをシークする `seekToNextNavCuePoint()` メソッドを使用することもできます。このメソッドを、次のナビゲーションキューポイント検索の開始時間を示すパラメータ `time` に渡すこともできますデフォルト値は現在の `playheadTime` です。

この他に、`seek()` メソッドを使用して、FLV ファイルの指定された長さをシークすることも可能です。

次の例では、キューポイント間をジャンプしたり、または FLVPlayback コンポーネントを再生する FLV ファイルの指定された長さをジャンプしたりするためのボタン、および指定されたキューポイントにジャンプするためのボタンを追加します。

指定された長さをシークするには :

1. seekduration.fla という新しい Flash ドキュメントを作成します。
2. FLVPlayback コンポーネントのインスタンスを [コンポーネント] パネル ([ウィンドウ]-[コンポーネント]) からドラッグします。

   このコンポーネントは "FLVPlayback - Player 8" フォルダにあります。
3. コンポーネントを選択し、プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) を開きます。
4. [インスタンス名] テキストボックスに「my_flvPb」と入力します。
5. Button コンポーネントのインスタンスを [コンポーネント] パネルからステージにドラッグします。
6. Button コンポーネントを選択し、[インスタンス名] テキストボックスに「seek_button」と入力します。
7. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
import mx.controls.Button;
import mx.video.FLVPlayback;
var seek_button:Button;
var my_flvPb:FLVPlayback;
```

```
my_flvPb.autoPlay = false;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/sheep.flv";
seek_button.label = "Seek";
seek_button.addEventListener("click", seekFlv);
function seekFlv(eventObj:Object):Void {
   // 2 秒間シークする
   my_flvPb.seek(2);
}
```

8. [制御]-[ムービープレビュー] を選択してコードをテストします。

   ボタンをクリックすると、再生ヘッドは指定された長さ、つまりビデオが開始してから 2 秒後に移動します。

FLVPlayback 機能を使ってシーク機能を追加するには :

1. seek1.fla という新しい Flash ドキュメントを作成します。
2. FLVPlayback コンポーネントのインスタンスを [コンポーネント] パネル ([ウィンドウ]-[コンポーネント]) からドラッグします。

   このコンポーネントは "FLVPlayback - Player 8" フォルダにあります。
3. コンポーネントを選択し、プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) を開きます。
4. [インスタンス名] テキストボックスに「my_flvPb」と入力します。
5. Button コンポーネントのインスタンスを [コンポーネント] パネルからステージにドラッグします。
6. Button コンポーネントを選択し、[インスタンス名] テキストボックスに「my_button」と入力します。
7. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
import mx.video.FLVPlayback;
var my_flvPb:FLVPlayback;
my_flvPb.autoPlay = false;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
   cuepoints.flv";
my_button.label = "Next cue point";

function clickMe(){
  my_flvPb.seekToNextNavCuePoint();
}
my_button.addEventListener("click", clickMe);
```

8. [制御]-[ムービープレビュー] を選択してコードをテストします。

   "cuepoints.flv" ファイルには、ビデオファイルの先頭付近、中間、最後の 3 か所にナビゲーションキューポイントがあります。ボタンをクリックすると、FLVPlayback インスタンスが次のキューポイントをシークし、その動作をビデオファイルの最後のキューポイントに達するまで続けます。

この他に、seekToCuePoint() メソッドを使用して、FLV ファイルの指定されたキューポイントをシークすることも可能です。

指定されたキューポイントをシークするには :

1. seek2.fla という新しい Flash ドキュメントを作成します。
2. FLVPlayback コンポーネントのインスタンスを [コンポーネント] パネル ([ウィンドウ]-[コンポーネント]) からドラッグします。

   このコンポーネントは "FLVPlayback - Player 8" フォルダにあります。
3. コンポーネントを選択し、プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) を開きます。
4. [インスタンス名] テキストボックスに「my_flvPb」と入力します。
5. FLVPlayback インスタンスを選択したまま、[パラメータ] タブをクリックします。
6. [contentPath] テキストボックスに「http://www.helpexamples.com/flash/video/cuepoints.flv」と入力します。

   [contentPath] テキストボックスに URL を入力すると、キューポイントが [パラメータ] タブ (cuePoint パラメータの次) に表示されます。したがって、コード内にあるキューポイントの名前を確認できます。拡大鏡アイコンをクリックすると、ビデオファイルのすべてのキューポイントを表示し、各キューポイントの情報を表で確認できます。
7. Button コンポーネントのインスタンスを [コンポーネント] パネルからステージにドラッグします。
8. Button コンポーネントを選択し、[インスタンス名] テキストボックスに「my_button」と入力します。
9. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
import mx.video.FLVPlayback;
var my_flvPb:FLVPlayback;
my_flvPb.autoPlay = false;
my_button.label = "Seek to point2";

function clickMe(){
  my_flvPb.seekToNavCuePoint("point2");
}
my_button.addEventListener("click", clickMe);
```

10. [制御]-[ムービープレビュー] を選択してコードをテストします。

    "cuepoints.flv" ファイルには、ビデオファイルの先頭付近、中間、最後の 3 か所にナビゲーションキューポイントがあります。ボタンをクリックすると、FLVPlayback インスタンスが指定されたキューポイント (`point2`) をシークします。

キューポイントと FLVPlayback コンポーネントの詳細については、『ActionScript 2.0 リファレンスガイド』を参照してください。

メタデータの操作

onMetaData メソッドを使用して、FLV ファイルのメタデータ情報を表示することができます。メタデータには、長さ、幅、高さ、およびフレームレートなど、FLV ファイルに関する情報が含まれています。FLV ファイルに追加されるメタデータ情報は、FLV ファイルのエンコードに使用するソフトウェアや、メタデータ情報の追加に使用するソフトウェアによって異なります。

メモ	ビデオファイルにメタデータ情報がない場合、ツールを使用して追加することができます。

NetStream.onMetaData を操作するには、メタデータを含む Flash Video が必要です。Flash 8 Video Encoder を使用して FLV ファイルをエンコードすると、FLV ファイルにはメタデータが含まれることになります (次の、Flash 8 Video Encoder でエンコードした FLV ファイルのメタデータ一覧の例を参照してください)。

メモ	Flash Video Exporter 1.2 以降 (Flash 8 Video Exporter を含む) では、FLV ファイルにメタデータが追加されます。Sorenson Squeeze 4.1 以降でも、ビデオファイルにメタデータが追加されます。

次の例では、NetStream.onMetaData を使用して、Flash 8 Video Encoder でエンコードされた FLV ファイルのメタデータ情報を追跡します。

NetStream.onMetaData を使用してメタデータ情報を表示するには :

1. flvMetadata.fla という新しい FLA ファイルを作成します。
2. [ライブラリ] パネル ([ウィンドウ]-[ライブラリ]) の [ライブラリ] ポップアップメニューから [新規ビデオ] を選択します。
3. [ビデオプロパティ] ダイアログボックスで、ビデオシンボルに名前を付け、[ビデオ] (ActionScript により制御) を選択します。
4. [OK] をクリックして、ビデオオブジェクトを作成します。
5. ビデオオブジェクトを [ライブラリ] パネルからステージにドラッグして、ビデオオブジェクトインスタンスを作成します。
6. ステージ上でビデオオブジェクトを選択し、プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) の [インスタンス名] テキストボックスに「my_video」と入力します。
7. ビデオインスタンスを選択したまま、[幅] テキストボックスに「320」、[高さ] テキストボックスに「213」と入力します。
8. タイムラインでフレーム 1 を選択し、[アクション] パネルを開きます ([ウィンドウ]-[アクション])。

9. [アクション] パネルで、次のコードを入力します。

```
// NetConnection オブジェクトを作成する。
var netConn:NetConnection = new NetConnection();
// ローカルストリーミング接続を作成する。
netConn.connect(null);
// NetStream オブジェクトを作成し、onStatus() 関数を定義する。
var nStream:NetStream = new NetStream(netConn);
// NetStream ビデオフィードを Video オブジェクトに割り当てる。
my_video.attachVideo(nStream);
// バッファ時間を設定する。
nStream.setBufferTime(30);
// FLV ファイルの再生を開始する。
nStream.play("http://www.helpexamples.com/flash/video/lights_short.flv");
// メタデータを追跡する。
nStream.onMetaData = function(myMeta) {
     for (var i in myMeta) {
          trace(i + ":\t" + myMeta[i])
     }
};
```

10. [制御]-[ムービープレビュー] を選択してコードをテストします。

[出力] パネルに次の情報が表示されます。

```
canSeekToEnd:true
audiocodecid:2
audiodelay:0.038
audiodatarate:96
videocodecid:4
framerate:15
videodatarate:400
height:213
width:320
duration:8.04
```

メモ	オーディオを含まないビデオの場合、エンコード中、オーディオ情報がメタデータに追加されないため、オーディオ関連のメタデータ情報 (audiodatarate など) は undefined のままで返されます。

次の形式を使用すれば、ほとんどのメタデータ情報を表示することができます。たとえば、次のコードは FLV ファイルの長さを表します。

```
nStream.onMetaData = function(myMeta) {
  trace("FLV duration: " + myMeta.duration + " sec.");
};
```

この形式では、cuePoint メタデータ情報を追跡することはできません。キューポイントの追跡については、551 ページの「FLV ファイルからのキューポイントの追跡」を参照してください。

FLV ファイルに合わせたサーバーの設定

FLV ファイルを操作するには、サーバーを FLV ファイル形式に合わせて設定することが必要になる場合があります。MIME (Multipurpose Internet Mail Extensions) は、非 ASCII ファイルをインターネット接続を介して送信することのできる標準データ仕様です。Web ブラウザや電子メールクライアントはさまざまな MIME タイプを解釈できるよう設定されているため、ビデオ、オーディオ、グラフィック、およびフォーマット化されたテキストを送受信することができます。Web サーバーから FLV ファイルをロードするには、ファイル拡張子と MIME タイプを Web サーバーに登録することが必要な場合があります。ご使用の Web サーバーのマニュアルを参照してください。FLV ファイルの MIME タイプは `video/x-flv` です。FLV ファイルタイプの全情報を次に示します。

MIME タイプ : video/x-flv

ファイル拡張子 : .flv

必須パラメータ : なし

オプションパラメータ : なし

エンコードに関する考慮事項 : FLV ファイルはバイナリファイルです。アプリケーションによっては application/octet-stream サブタイプも設定する必要があります。

セキュリティに関する問題 : なし

パブリッシュされた仕様 : www.adobe.com/go/flashfileformat_jp

Microsoft は、Microsoft Internet Information Services (IIS) 6.0 Web サーバーでのストリーミングメディアの取り扱い方法を旧バージョンとは異なるものに変更しました。IIS の旧バージョンでは、Flash Video をストリーミングするのに、変更は一切必要ありませんでした。IIS 6.0 では、FLV ファイルがストリーミングメディアであることを、Windows 2003 に含まれるデフォルト Web サーバーに認識させるために MIME タイプが必要になります。

外部 FLV ファイルをストリーミングする SWF ファイルが Microsoft Windows 2003 サーバーにあって、ブラウザで表示される場合、SWF ファイルの再生は正常に行われますが、FLV ビデオはストリーミングされません。この問題は、Flash オーサリングツールや Macromedia Flash Video Kit for Dreamweaver MX 2004 の旧バージョンで作成したファイルを含め、Windows 2003 サーバーにあるすべての FLV ファイルに影響します。これらのファイルは、他のオペレーティングシステムでテストすると正常に動作します。

FLV ビデオストリーミング用の Microsoft Windows 2003 および Microsoft IIS Server 6.0 の設定については、http://www.adobe.com/cfusion/knowledgebase/index.cfm?id=tn_19439 を参照してください。

Macintosh 上のローカル FLV ファイルのターゲット設定について

スラッシュ (/) を使用する相対パスによって、Macintosh コンピュータの非システムドライブからローカル FLV を再生しようとしても、再生されません。非システムドライブには、CD-ROM、パーティションされたハードディスク、リムーバブルストレージメディア、および接続済みのストレージデバイスなどが含まれます。

メモ	原因は、Flash や Flash Player の限界ではなく、オペレーティングシステムの限界にあります。

Macintosh 上の非システムドライブによる FLV ファイルの再生については、スラッシュベースの表記 (/) ではなく、コロンベースの表記 (:) による絶対パスを使用します。次の一覧は、この 2 種類の表記の違いをまとめたものです。

スラッシュベース表記 myDrive/myFolder/myFLV.flv

コロンベース表記 (Macintosh) myDrive:myFolder:myFLV.flv

また、Macintosh での再生に使用する CD-ROM 用プロジェクタファイルも作成できます。Macintosh CD-ROM および FLV ファイルの最新情報については、http://www.adobe.com/cfusion/knowledgebase/index.cfm?id=3121b301 を参照してください。

メディアファイル用プログレスアニメーションの作成について

ActionScript を使用すると、さまざまな方法で外部メディアをプリロードしたり、ダウンロードの進行状況を追跡したりできます。プログレスバーやアニメーションを作成して、ロード進行状況やロードしたコンテンツの量を視覚的に確認できます。

SWF ファイルや JPEG ファイルをプリロードするには、MovieClipLoader クラスを使用します。このクラスには、ダウンロードの進行状況をチェックするためのイベントリスナーメカニズムがあります。詳細については、427 ページの「外部 SWF ファイルおよびイメージファイルのロード」を参照してください。

MP3 ファイルのダウンロード進行状況を追跡する場合は、`Sound.getBytesLoaded()` メソッドと `Sound.getBytesTotal()` メソッドを使用します。FLV ファイルのダウンロード進行状況を追跡する場合は、`NetStream.bytesLoaded` プロパティと `NetStream.bytesTotal` プロパティを使用します。詳細については、420 ページの「MP3 ファイルのプリロード」を参照してください。

メディアファイルをロードするためのプログレスバーの作成については、次のトピックを参照してください。

- 563 ページの「SWF ファイルおよびイメージファイルのロード用プログレスアニメーションの作成」
- 565 ページの「ActionScript による MP3 ファイルのロード用プログレスバーの作成」
- 567 ページの「ActionScript による FLV ファイルのロード用プログレスバーの作成」

スクリプトによるアニメーションを使用して、プログレスバーアニメーションを作成するサンプルのソースファイル "tweenProgress.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Tween Progressbar" フォルダに移動してサンプルにアクセスします。

SWF ファイルおよびイメージファイルのロード用プログレスアニメーションの作成

大きな SWF ファイルやイメージファイルをアプリケーションにロードする際、ロード進行状況を表示するアニメーションが必要になる場合があります。アニメーションのロード進行状況を表示するプログレスバーは作成することもできます。また、ファイルのロード状況につれて変化するアニメーションも作成できます。SWF ファイルおよびイメージファイルのロードについては、533 ページの「外部 SWF ファイルおよびイメージファイルのロード」を参照してください。

次の例は、MovieClipLoader クラスと Drawing API を使用して、イメージファイルのロード進行状況を表示する方法を示しています。

SWF ファイルおよびイメージファイルのロード用プログレスアニメーションを作成するには :

1. loadImage.fla という新しい Flash ドキュメントを作成します。
2. [修正]-[ドキュメント] を選択し、[幅] テキストボックスに「700」、[高さ] テキストボックスに「500」と入力して、ドキュメントサイズを変更します。
3. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
// コンテンツを納めるクリップを作成する
this.createEmptyMovieClip("progressBar_mc", 0);
progressBar_mc.createEmptyMovieClip("bar_mc", 1);
progressBar_mc.createEmptyMovieClip("stroke_mc", 2);
// 描画メソッドを使用してプログレスバーを作成する
with (progressBar_mc.stroke_mc) {
   lineStyle(0, 0x000000);
   moveTo(0, 0);
   lineTo(100, 0);
   lineTo(100, 10);
   lineTo(0, 10);
   lineTo(0, 0);
}
with (progressBar_mc.bar_mc) {
   beginFill(0xFF0000, 100);
   moveTo(0, 0);
   lineTo(100, 0);
   lineTo(100, 10);
   lineTo(0, 10);
   lineTo(0, 0);
   endFill();
   _xscale = 0;
}
progressBar_mc._x = 2;
progressBar_mc._y = 2;
```

```
// ロード進捗状況
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip) {
   progressBar_mc.bar_mc._xscale = 0;
};
mclListener.onLoadProgress = function(target_mc:MovieClip,
  bytesLoaded:Number, bytesTotal:Number) {
   progressBar_mc.bar_mc._xscale = Math.round(bytesLoaded/bytesTotal*100);
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
   progressBar_mc.removeMovieClip();
};
mclListener.onLoadInit = function(target_mc:MovieClip) {
   target_mc._height = 500;
   target_mc._width = 700;
};
// イメージを納めるクリップを作成する
this.createEmptyMovieClip("image_mc", 100);
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
/* イメージをクリップにロードする
次の URL を SWF または別のイメージファイルに変更できます。*/
image_mcl.loadClip("http://www.helpexamples.com/flash/images/gallery1/
  images/pic3.jpg", image_mc);
```

4. [制御]-[ムービープレビュー] を選択して、イメージロードを表示し、プログレスバーを確認します。

メモ	このコードを 2 度目にテストするときは、イメージがキャッシュされ、プログレスバーの表示は直ちに完了します。テストを何度か行う場合は、異なるイメージを使用し、外部ソースからロードしてください。ローカルソースの場合、コンテンツのロードが速すぎるため、アプリケーションのテストで問題を起こす可能性があります。

スクリプトによるアニメーションを使用して、プログレスバーアニメーションを作成するサンプルのソースファイル "tweenProgress.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Tween Progressbar" フォルダに移動してサンプルにアクセスします。

フォトギャラリアプリケーションのサンプルについては、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Galleries" フォルダに移動して次のサンプルにアクセスします。

- gallery_tree.fla
- gallery_tween.fla

これらのサンプルは、イメージファイルの SWF ファイルへのダウンロード中に、ActionScript を使用してムービークリップを動的に制御する方法の例を示すものです。

ActionScript による MP3 ファイルのロード用プログレスバーの作成

次の例では、SWF ファイルにいくつかの曲をロードします。Drawing API を使用して作成したプログレスバーは、ロードの進行状況を表示します。音楽が始まってロードが完了すると、[出力] パネルに情報が表示されます。MP3 ファイルのロードについては、538 ページの「MP3 ファイルのロード」を参照してください。

MP3 ファイルのロード用プログレスバーを作成するには :

1. panel_as.fla という新しい Flash ドキュメントを作成します。
2. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
var pb_height:Number = 10;
var pb_width:Number = 100;
var pb:MovieClip = this.createEmptyMovieClip("progressBar_mc",
   this.getNextHighestDepth());
pb.createEmptyMovieClip("bar_mc", pb.getNextHighestDepth());
pb.createEmptyMovieClip("vBar_mc", pb.getNextHighestDepth());
pb.createEmptyMovieClip("stroke_mc", pb.getNextHighestDepth());
pb.createTextField("pos_txt", pb.getNextHighestDepth(), 0, pb_height,
   pb_width, 22);

pb._x = 100;
pb._y = 100;

with (pb.bar_mc) {
   beginFill(0x00FF00);
   moveTo(0, 0);
   lineTo(pb_width, 0);
   lineTo(pb_width, pb_height);
   lineTo(0, pb_height);
   lineTo(0, 0);
   endFill();
   _xscale = 0;
}
with (pb.vBar_mc) {
   lineStyle(1, 0x000000);
   moveTo(0, 0);
   lineTo(0, pb_height);
}
with (pb.stroke_mc) {
   lineStyle(3, 0x000000);
   moveTo(0, 0);
   lineTo(pb_width, 0);
   lineTo(pb_width, pb_height);
   lineTo(0, pb_height);
   lineTo(0, 0);
}
```

```
var my_interval:Number;
var my_sound:Sound = new Sound();
my_sound.onLoad = function(success:Boolean) {
   if (success) {
      trace("sound loaded");
   }
};
my_sound.onSoundComplete = function() {
   clearInterval(my_interval);
   trace("Cleared interval");
}
my_sound.loadSound("http://www.helpexamples.com/flash/sound/song2.mp3",
  true);
my_interval = setInterval(updateProgressBar, 100, my_sound);

function updateProgressBar(the_sound:Sound):Void {
   var pos:Number = Math.round(the_sound.position / the_sound.duration *
  100);
   pb.bar_mc._xscale = pos;
   pb.vBar_mc._x = pb.bar_mc._width;
   pb.pos_txt.text = pos + "%";
}
```

3. [制御]-[ムービープレビュー] を選択して、MP3 ファイルをロードし、プログレスバーを確認します。

メモ	このコードを 2 度目にテストするときは、イメージがキャッシュされ、プログレスバーの表示は直ちに完了します。テストを何度か行う場合は、異なるイメージを使用し、外部ソースからロードしてください。ローカルソースの場合、コンテンツのロードが速すぎるため、アプリケーションのテストで問題を起こす可能性があります。

サウンドの使用の詳細については、『ActionScript 2.0 リファレンスガイド』の Sound クラスエントリ、Sound を参照してください。

スクリプトによるアニメーションを使用して、プログレスバーアニメーションを作成するサンプルのソースファイル "tweenProgress.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Tween Progressbar" フォルダに移動してサンプルにアクセスします。

MP3 ファイルをロードするサンプルのソースファイル "jukebox.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ComponentsAS2/Jukebox" フォルダに移動してサンプルにアクセスします。このサンプルは、データ型、一般的なコーディングの原則、および複数のコンポーネントを使用してジュークボックスを作成する方法を示しています。

ActionScript による FLV ファイルのロード用プログレスバーの作成

プログレスバーを作成して、FLV ファイルのロード進行状況を表示することができます。SWF ファイルへの FLV ファイルのロードについては、548 ページの「FLV ファイルのプリロード」を参照してください。FLV ファイルおよび Flash についてのその他の情報は、543 ページの「FLV ビデオの使用について」を参照してください。

次の例では、Drawing API を使用してプログレスバーを作成しています。また、`bytesLoaded` および `bytesTotal` プロパティを使用して、`my_video` というビデオオブジェクトインスタンスに "video1.flv" がロードされる状況を表示します。`loaded_txt` というテキストフィールドが動的に作成され、ロードの進行状況に関する情報を表示します。

ロード進行状況を表示するプログレスバーを作成するには :

1. flvProgress.fla という新しい FLA ファイルを作成します。
2. [ライブラリ] パネル ([ウィンドウ]-[ライブラリ]) の [ライブラリ] ポップアップメニューから [新規ビデオ] を選択します。
3. [ビデオプロパティ] ダイアログボックスで、ビデオシンボルに名前を付け、[ビデオ] (ActionScript により制御) を選択します。
4. [OK] をクリックして、ビデオオブジェクトを作成します。
5. ビデオオブジェクトを [ライブラリ] パネルからステージにドラッグして、ビデオオブジェクトインスタンスを作成します。
6. ステージ上でビデオオブジェクトを選択し、プロパティインスペクタ ([ウィンドウ]-[プロパティ]-[プロパティ]) の [インスタンス名] テキストボックスに「my_video」と入力します。
7. ビデオインスタンスを選択したまま、[幅] テキストボックスに「320」、[高さ] テキストボックスに「213」と入力します。
8. タイムラインのフレーム 1 を選択し、[アクション] パネルに次のコードを入力します。

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("http://www.helpexamples.com/flash/video/typing_short.flv");

this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10, 160,
   22);
this.createEmptyMovieClip("progressBar_mc", this.getNextHighestDepth());
progressBar_mc.createEmptyMovieClip("bar_mc",
   progressBar_mc.getNextHighestDepth());
with (progressBar_mc.bar_mc) {
    beginFill(0xFF0000);
    moveTo(0, 0);
```

```
    lineTo(100, 0);
    lineTo(100, 10);
    lineTo(0, 10);
    lineTo(0, 0);
    endFill();
    _xscale = 0;
}
progressBar_mc.createEmptyMovieClip("stroke_mc",
   progressBar_mc.getNextHighestDepth());
with (progressBar_mc.stroke_mc) {
    lineStyle(0, 0x000000);
    moveTo(0, 0);
    lineTo(100, 0);
    lineTo(100, 10);
    lineTo(0, 10);
    lineTo(0, 0);
}

var loaded_interval:Number = setInterval(checkBytesLoaded, 500, stream_ns);
function checkBytesLoaded(my_ns:NetStream) {
    var pctLoaded:Number = Math.round(my_ns.bytesLoaded / my_ns.bytesTotal *
   100);
    loaded_txt.text = Math.round(my_ns.bytesLoaded / 1000) + " of " +
   Math.round(my_ns.bytesTotal / 1000) + " KB loaded (" + pctLoaded + "%)";
    progressBar_mc.bar_mc._xscale = pctLoaded;
    if (pctLoaded>=100) {
       clearInterval(loaded_interval);
    }
}
```

9. [制御]-[ムービープレビュー] を選択してコードをテストします。

   ビデオがロードされ、アニメーションバーや変化するテキスト値がロード進行状況を表します。これらのエレメントとビデオが重なった場合は、ステージ上にビデオオブジェクトを移動します。前のコードにある beginFill と lineStyle を変更することで、プログレスバーの色を変えることができます。

メモ	プログレスバーが簡単にロードできる場合、(この例のテストか、別の手順によるロードで) ビデオは既にハードディスク上にキャッシュされています。ビデオがキャッシュされたら、FLV ファイルをサーバーにアップロードして、代わりにロードします。

スクリプトによるアニメーションを使用して、プログレスバーアニメーションを作成するサンプルのソースファイル "tweenProgress.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Tween Progressbar" フォルダに移動してサンプルにアクセスします。

第 15 章

外部データの操作

15

Flash CS3 Professional では、ActionScript を使用して外部ソースから SWF ファイルにデータをロードすることができます。また、ユーザーやサーバーによって提供されるデータを SWF ファイルから、(ColdFusion や JRun などの) アプリケーションサーバー、または PHP や Perl などの別のタイプのサーバーサイドスクリプトに送信することもできます。Flash Player では、HTTP または HTTPS 経由でデータを送信またはロードすることや、ローカルなテキストファイルからデータをロードすることができます。また、チャットアプリケーションや株価情報サービスなど、待ち時間を短くすることが求められるアプリケーションには、永続的な TCP/IP ソケット接続を確立できます。Flash Player 8 以降では、ユーザーのコンピュータからサーバーへのデータのアップロード、およびサーバーからユーザーのコンピュータへのデータのダウンロードが可能です。

Flash ムービーにデータをロードする場合、または SWF ファイルからデータを送信する場合、データ形式は XML (Extensible Markup Language : 拡張可能マークアップ言語)、または名前と値のペアになります。

また、Flash Player は、Web ブラウザなどのホスト環境との間でデータを送受信することができます。データの送受信は、同じコンピュータまたは Web ページで動作する Flash Player の別のインスタンスとの間でも可能です。

デフォルトでは、SWF ファイルからアクセスできるデータは、まったく同じドメイン (www.adobe.com など) に存在するものだけに制限されています。詳細については、627 ページの「ネットワーク API の制限」を参照してください。

外部データの操作の詳細については、次のトピックを参照してください。

変数の送信とロード 570
HTTP を使用したサーバーサイドスクリプトへの接続 574
ファイルのアップロードとダウンロードについて 579
XML について 586
Flash Player とのメッセージの送受信 595
External API について 599

変数の送信とロード

SWF ファイルは、HTML ページのように情報の収集と表示を行うためのウィンドウです。ただし、ブラウザ内にいったんロードされると、ページ全体を再ロードしなくても継続して新しい情報に更新することができます。ActionScript の関数とメソッドを使用すると、サーバーサイドスクリプトとの間で情報を送受信したり、テキストファイルや XML ファイルから情報を受信したりできます。

さらに、サーバーサイドスクリプトは、データベースに特定の情報を要求し、SWF ファイルに情報を中継できます。サーバーサイドスクリプトは、さまざまな言語で書くことができます。最も一般的な言語として、CFML、Perl、ASP (Microsoft Active Server Pages)、PHP があります。データベースに格納された情報を検索することにより、ダイナミックでパーソナライズされたコンテンツを作成できます。たとえば、メッセージボードやユーザー別のプロファイルを作成したり、ユーザーが購入したものを追跡するショッピングカートを作成できます。

いくつかの ActionScript 関数とメソッドでは、SWF ファイルとの情報のやり取りを行うことができます。各関数とメソッドは、プロトコルを使用して情報を転送します。そのため、決められた方法で情報をフォーマットする必要があります。

- HTTP または HTTPS プロトコルを使用して URL エンコード形式で情報を送る関数と MovieClip メソッドには、`getURL()`、`loadVariables()`、`loadVariablesNum()`、`loadMovie()`、`loadMovieNum()` があります。
- HTTP または HTTPS プロトコルを使用して URL エンコード形式で情報を送信し、ロードする LoadVars メソッドには、`load()`、`send()`、および `sendAndLoad()` があります。
- HTTP または HTTPS プロトコルを使用して XML 形式で情報を送信し、ロードするメソッドには、`XML.send()`、`XML.load()`、および `XML.sendAndLoad()` があります。
- TCP/IP ソケット接続を生成し、その接続を使用して XML 形式で情報を送信し、ロードするメソッドには、`XMLSocket.connect()` と `XMLSocket.send()` があります。

詳細については、次のトピックを参照してください。

- 570 ページの「ロードされたデータの確認」
- 571 ページの「データロードの進行状況を表示するプログレスバーの作成」

ロードされたデータの確認

データを SWF ファイルにロードする関数やメソッド (`XMLSocket.send()` を除く) はどれも " 非同期 " です。つまり、アクションの結果が返る時間は一定ではありません。

SWF ファイルにロードされたデータを使用する前に、データがロードされているかどうかを確認する必要があります。たとえば、変数をロードするスクリプトと同じスクリプト内でそれらの変数の値を処理することはできません。ロードされるまでは操作するデータがファイルに存在しないためです。次のスクリプトでは、ファイル "myData.txt" からロードされたことが確認されるまで、変数 `lastSiteVisited` は使用できません。"myData.txt" ファイルに、次のようなテキストが格納されているとします。

```
lastSiteVisited=www.adobe.com
```

次のコードを使用しても、ロード中のデータをトレースすることはできません。

```
loadVariables("myData.txt", 0);
trace(lastSiteVisited); // undefined
```

各関数またはメソッドには、ロードされたデータを確認する方法があります。loadVariables または loadMovie function を使用すると、ムービークリップターゲットに情報をロードし、onData ハンドラを使用してスクリプトを実行できます。loadVariables 関数を使用してデータをロードすると、最後の変数がロードされたときに onData ハンドラが実行されます。loadMovie 関数を使用してデータをロードすると、SWF ファイルの一部が Flash Player にストリームされるたびに onData ハンドラが実行されます。

たとえば、次の ActionScript では、ファイル "myData.txt" からムービークリップ loadTarget_mc に変数がロードされます。loadTarget_mc インスタンスに割り当てられた onData() ハンドラでは、ファイル "myData.txt" からロードされる変数 lastSiteVisited を使用します。次のトレースアクションは、lastSiteVisited などすべての変数がロードされた後にだけ実行されます。

```
this.createEmptyMovieClip("loadTarget_mc", this.getNextHighestDepth());
this.loadTarget_mc.onData = function() {
  trace("Data Loaded");
  trace(this.lastSiteVisited);
};
loadVariables("myData.txt", this.loadTarget_mc);
```

XML.load()、XML.sendAndLoad()、および XMLSocket.connect() の各メソッドを使用する場合には、データを着信時に処理するハンドラを定義してください。このハンドラは、定義した関数を割り当てる XML または XMLSocket オブジェクトのプロパティです。情報を受信すると、ハンドラが自動的に呼び出されます。XML オブジェクトの場合は、XML.onLoad() または XML.onData() を使用します。XMLSocket オブジェクトの場合は、XMLSocket.onConnect() を使用します。

詳細については、588 ページの「XML クラスの使用」および 593 ページの「XMLSocket クラスの使用」を参照してください。LoadVars を使用して、受信後に処理されるデータを送受信する方法については、575 ページの「LoadVars クラスの使用」を参照してください。

データロードの進行状況を表示するプログレスバーの作成

次の実習では、Drawing API (アプリケーションプログラミングインターフェイス) を使用してシンプルなプリローダを動的に作成し、XML ドキュメントのロード進行状況を表示します。

ヒント	リモート XML ファイルのロードが速すぎてプリロード効果が確認できない場合は、大きな XML ファイルをインターネットにアップロードしてから、ダウンロードしてみてください。

Drawing API を使用してのプログレスバーの作成 :

1. 新しい Flash ドキュメントを作成し、drawapi.fla という名前で保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var barWidth:Number = 200;
var barHeight:Number = 6;

this.createEmptyMovieClip("pBar_mc", 9999);
var bar:MovieClip = pBar_mc.createEmptyMovieClip("bar_mc", 10);
bar.beginFill(0xFF0000, 100);
bar.moveTo(0, 0);
bar.lineTo(barWidth, 0);
bar.lineTo(barWidth, barHeight);
bar.lineTo(0, barHeight);
bar.lineTo(0, 0);
bar.endFill();
bar._xscale = 0;

var stroke:MovieClip = pBar_mc.createEmptyMovieClip("stroke_mc", 20);
stroke.lineStyle(0, 0x000000);
stroke.moveTo(0, 0);
stroke.lineTo(barWidth, 0);
stroke.lineTo(barWidth, barHeight);
stroke.lineTo(0, barHeight);
stroke.lineTo(0, 0);

pBar_mc.createTextField("label_txt", 30, 0, barHeight, 100, 21);
pBar_mc.label_txt.autoSize = "left";
pBar_mc.label_txt.selectable = false;

pBar_mc._x = (Stage.width - pBar_mc._width) / 2;
pBar_mc._y = (Stage.height - pBar_mc._height) / 2;

var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  pBar_mc.onEnterFrame = undefined;
  if (success) {
    trace("XML loaded successfully");
  } else {
    trace("Unable to load XML");
  }
};
my_xml.load("http://www.helpexamples.com/flash/xml/ds.xml");

pBar_mc.onEnterFrame = function() {
  var pctLoaded:Number = Math.floor(my_xml.getBytesLoaded() /
  my_xml.getBytesTotal() * 100);
  if (!isNaN(pctLoaded)) {
    pBar_mc.bar_mc._xscale = pctLoaded;
```

```
      pBar_mc.label_txt.text = pctLoaded + "% loaded";
      if (pctLoaded >= 100) {
        pBar_mc.onEnterFrame = undefined;
      }
   }
};
```

前述のコードは 7 つのセクションに分解されます。最初のセクションは、ステージに描画されたときのプログレスバーの幅と高さを定義します。次のセクションによって、プログレスバーはステージ上で中央揃えされます。さらに次のセクションでは、pBar_mc と bar_mc という 2 つのムービークリップが作成されます。bar_mc ムービークリップが pBar_mc ムービークリップ内にネストされ、ステージ上に赤い矩形が描画されます。リモート Web サイトから外部 XML ファイルがロードされると、bar_mc インスタンスの _xscale プロパティが変更されます。

次に、2 つ目のムービークリップ stroke_mc が、pBar_mc の中にネストされます。stroke_mc ムービークリップにより、最初のセクションで定義された変数 barHeight と barWidth により指定したサイズに一致するアウトラインがステージ上に描画されます。コードの 4 番目のセクションは、XML ファイルのロード率を表示する、ProgressBar コンポーネントのラベルに似たテキストフィールドをムービークリップ内に作成します。次に、pBar_mc ムービークリップ (ネストされた bar_mc、stroke_mc、および label_txt インスタンスを含む) がステージ上の中央に配置されます。

コードの 6 番目のセクションは、外部 XML ファイルをロードするために使用される新しい XML オブジェクトを定義します。onLoad イベントハンドラが定義され、[出力] パネルまでメッセージをトレースします。onLoad イベントハンドラは、pBar_mc ムービークリップ用の (次のセクションで定義される) onEnterFrame イベントハンドラを削除します。最後のセクションでは、pBar_mc ムービークリップ用の onEnterFrame イベントハンドラが定義されます。このイベントハンドラが、外部 XML ファイルのロード率を監視し、_xscale プロパティと bar_mc ムービークリップを変更します。最初に、onEnterFrame イベントハンドラによって、ファイルのダウンロード率が計算されます。ファイルのロード率が有効である場合、bar_mc の _xscale プロパティが設定され、pBar_mc 内のテキストフィールドにファイルのロード率が表示されます。ファイルのロードが完了すると (ロード率が 100% に達すると)、onEnterFrame イベントハンドラが削除され、ダウンロード進行状況の監視が終了します。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

外部 XML ファイルがロードされると、ネストされた bar_mc ムービークリップのサイズが変更され、その XML ファイルのダウンロード進行状況が表示されます。XML ファイルのロードが完了すると、onEnterFrame イベントハンドラが削除され、ダウンロード進行状況の計算が終了します。ダウンロードの速さによっては、bar_mc と stroke_mc の 2 つのムービークリップが同じ幅になるまで、バーが変化するのを視認することができます。ダウンロードが速すぎると、プログレスバーは 0% から 100% にすぐに達してしまい、視認することが困難です。その場合は、より大きな XML ファイルをダウンロードしてみてください。

HTTP を使用したサーバーサイドスクリプトへの接続

loadVariables、loadVariablesNum、getURL、loadMovie、loadMovieNum の各関数と、loadVariables (MovieClip.loadVariables メソッド)、loadMovie (MovieClip.loadMovie メソッド)、および getURL (MovieClip.getURL メソッド) の各メソッドは、HTTP または HTTPS プロトコルを使用して、サーバーサイドスクリプトと通信することができます。これらの関数およびメソッドは、関数が割り当てられるタイムラインからすべての変数を送ります。MovieClip オブジェクトのメソッドとして使用する場合、`loadVariables()`、`getURL()`、および `loadMovie()` は指定されたムービークリップ内のすべての変数を送ります。それぞれの関数 (またはメソッド) は、応答を次のように処理します。

- `getURL()` 関数は、ブラウザウィンドウに情報を返し、Flash Player には情報を返しません。
- `loadVariables()` メソッドは、Flash Player 内の指定されたタイムラインまたは指定されたレベルに変数をロードするメソッドです。
- `loadMovie()` メソッドは、Flash Player 内の指定されたレベルまたはムービークリップに SWF ファイルをロードするメソッドです。

`loadVariables()`、`getURL()`、または `loadMovie()` では、パラメータを指定できます。

- *URL* は、リモートに変数を設定するファイルです。
- *Location* は、変数を受け取る SWF ファイル内のレベルまたはターゲットです。`getURL()` 関数は、このパラメータを取りません。

  レベルおよびターゲットの詳細については、『Flash ユーザーガイド』を参照してください。
- *Variables* は、HTTP メソッドの `GET` (変数を URL の最後に追加) または `POST` (別の HTTP ヘッダーで変数を送信) を指定し、これらのメソッドによって変数が送られます。このパラメータを省略した場合、デフォルトで `GET` を使用しますが、変数は送信されません。

たとえば、ゲームの高得点を追跡する場合、サーバーに得点を格納し、誰かがゲームを行うたびに `loadVariables()` を使用して SWF ファイルに得点をロードすることができます。関数呼び出しは次のようになります。

```
this.createEmptyMovieClip("highscore_mc", 10);
loadVariables("http://www.helpexamples.com/flash/highscore.php", highscore_mc,
  "GET");
```

この例では、`GET` HTTP メソッドを使用して、high_score.cfm と呼ばれる ColdFusion スクリプトからムービークリップインスタンス `scoreClip` に変数をロードします。

`loadVariables()` 関数によりロードされる変数はすべて、標準の MIME 形式 application/x-www-form-urlencoded (CFM および CGI スクリプトで使用される標準形式) である必要があります。`loadVariables()` の *URL* パラメータで指定したファイルは、Flash で読み込めるよう、変数と値のペアをこの形式で書き出すものでなければなりません。このファイルでは任意の数の変数を指定できます。変数と値のペアはアンパサンド (&) で区切り、値の中の語句はプラス記号 (+) で区切る必要があります。たとえば、次の例では複数の変数が定義されています。

```
highScore1=54000&playerName1=RGoulet&highScore2=53455&playerName2=
   WNewton&highScore3=42885&playerName3=TJones
```

メモ

特定の文字 (+ や & など) については、URL エンコードを行う必要があります。詳細については、www.adobe.com/go/tn_14143_jp を参照してください。

詳細については、次のトピック 575 ページの「LoadVars クラスの使用」を参照してください。また、『ActionScript 2.0 リファレンスガイド』の「loadVariables 関数」、「getURL 関数」、「loadMovie 関数」、および「LoadVars」の項目も参照してください。

LoadVars クラスの使用

Flash Player 6 以降にパブリッシュし、`loadVariables()` よりも高い柔軟性が必要な場合は、LoadVars クラスを使用すると、SWF ファイルとサーバーとの間で変数を受け渡すことができます。

Flash Player 6 で実装された LoadVars クラスは、CGI データを Web サーバーとの間で交換するという一般的な作業において、より洗練されたオブジェクト指向のインターフェイスを提供します。LoadVars クラスの利点としては次のようなことが挙げられます。

- コンテナムービークリップを作成して、クライアント/ サーバー間の通信に固有の変数でデータや既存のムービークリップを保持する必要がありません。
- このクラスインターフェイスは XML オブジェクトのインターフェイスと類似性があり、そのことが ActionScript の一貫性に貢献しています。このクラスは、`load()`、`send()`、`sendAndLoad()` の各メソッドを使用して、サーバーとの通信を行います。LoadVars クラスと XML クラスの主な違いは、LoadVars データが XML オブジェクトに格納されている XML DOM (Document Object Model) ツリーのプロパティではなく、LoadVars オブジェクトのプロパティであるという点です。
- このクラスインターフェイスは、従来の loadVariables インターフェイスと比べ、`load`、`send`、`sendAndLoad` といった、より直観的なメソッドを備えています。
- `getBytesLoaded` メソッドや `getBytesTotal` メソッドを使用すると、通信に関する追加情報を取得できます。
- データのダウンロードに関する進捗情報を取得できます (ただし、完全にダウンロードされるまではデータにアクセスすることはできません)。
- コールバックインターフェイスは、loadVariables に必要とされていた従来の `onClipEvent` (data) を使用したアプローチではなく、ActionScript メソッド (`onLoad`) を介して実装されます。

- エラー通知を備えています。
- カスタム HTTP 要求ヘッダーを追加できます。

LoadVars オブジェクトを作成した上でメソッドを呼び出す必要があります。このオブジェクトは、ロードされたデータを保持しておくための入れ物です。

次の手順は、ColdFusion と LoadVars クラスを使用して、SWF ファイルから電子メールを送信する方法を示しています。

メモ	この例を実行するためには、Web サーバーに ColdFusion がインストールされている必要があります。

LoadVars オブジェクトでデータをロードするには :

1. Macromedia Dreamweaver または任意のテキストエディタで CFM ファイルを作成します。ファイルに次のテキストを追加します。

```
<cfif StructKeyExists(Form, "emailTo")>
<cfmail to="#Form.emailTo#" from="#Form.emailFrom#"
  subject="#Form.emailSubject#">#Form.emailBody#</cfmail>
&result=true
<cfelse>
&result=false
</cfif>
```

2. このファイルを email.cfm という名前で保存し、Web サイトにアップロードします。
3. Flash で、新規ドキュメントを作成します。
4. ステージ上で 4 つのテキスト入力フィールドを作成し、それぞれに emailFrom_txt、emailTo_txt、emailSubject_txt、および emailBody_txt というインスタンス名を指定します。
5. ステージ上でダイナミックテキストフィールドを 1 つ作成し、そのフィールドにインスタンス名 debug_txt を指定します。
6. ボタンシンボルを作成し、インスタンスをステージにドラッグし、そのインスタンスに submit_btn という名前を付けます。
7. タイムラインでフレーム 1 を選択し、まだ [アクション] パネルを開いていない場合は開きます ([ウィンドウ]-[アクション])。
8. [アクション] パネルで、次のコードを入力します。

```
this.submit_btn.onRelease = function() {
  var emailResponse:LoadVars = new LoadVars();
  emailResponse.onLoad = function(success:Boolean) {
  if (success) {
       debug_txt.text = this.result;
```

```
    } else {
         debug_txt.text = "error downloading content";
    }
    };
    var email:LoadVars = new LoadVars();
    email.emailFrom = emailFrom_txt.text;
    email.emailTo = emailTo_txt.text;
    email.emailSubject = emailSubject_txt.text;
    email.emailBody = emailBody_txt.text;
    email.sendAndLoad("http://www.yoursite.com/email.cfm", emailResponse,
    "POST");
};
```

この ActionScript は、新しい LoadVars オブジェクトのインスタンスを作成し、テキストフィールドの値をこのインスタンスにコピーして、データをサーバーに送信します。CFM ファイルが電子メールを送信すると、SWF ファイルに `result` という変数 (`true` または `false`) が返され、この変数が `debug_txt` テキストフィールドに表示されます。

メモ	なお、www.yoursite.com という URL は、適切なドメインに変更してください。

9. ドキュメントを sendEmail.fla という名前で保存し、[ファイル]-[パブリッシュ] を選択してこのファイルをパブリッシュします。
10. sendEmail.swf を email.cfm (手順 2 で保存およびアップロードした ColdFusion ファイル) と同じディレクトリにアップロードします。
11. ブラウザで SWF ファイルを表示し、テストします。

詳細については、『ActionScript 2.0 リファレンスガイド』の「LoadVars」を参照してください。

Flash Player 8 以降では、LoadVars クラス、XML クラス、および MovieClipLoader クラスの `onHTTPStatus` イベントハンドラがサポートされるため、ユーザーが HTTP 要求からのステータスコードにアクセスできます。これにより、開発者は単にロード処理が失敗したという結果についてだけでなく、ロード処理が失敗した理由を確認できるようになります。

次の例は、LoadVars クラスの `onHTTPStatus` イベントハンドラを使用して、サーバーからのテキストファイルのダウンロードが正常に完了したかどうかや、HTTP 要求から戻されたステータスコードを確認することができます。

LoadVars オブジェクトで HTTP ステータスを確認するには :

1. 新しい Flash ドキュメントを作成し、loadvars.fla という名前で保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
this.createTextField("params_txt", 10, 10, 10, 100, 21);
params_txt.autoSize = "left";

var my_lv:LoadVars = new LoadVars();
my_lv.onHTTPStatus = function(httpStatus:Number) {
  trace("HTTP status is: " + httpStatus);
};
my_lv.onLoad = function(success:Boolean) {
  if (success) {
    trace("text file successfully loaded");
    params_txt.text = my_lv.dayNames;
  } else {
    params_txt.text = "unable to load text file";
  }
};
my_lv.load("http://www.helpexamples.com/flash/404.txt");
/* 出力 :
  URL "http://www.helpexamples.com/flash/404.txt" を開くときにエラーが発生。
  HTTP ステータス : 404
*/
```

前述のコードでは、新しいテキストフィールドをステージ上に作成し、テキストフィールドのオートサイジングを有効にできます。次に、LoadVars オブジェクトが 1 つと、`onHTTPStatus` と `onLoad` という 2 つのイベントハンドラが作成されます。`onHTTPStatus` イベントハンドラは、Flash Player 8 以降でサポートされる機能で、`LoadVars.load()` または `LoadVars.sendAndLoad()` 処理が完了すると呼び出されます。`onHTTPStatus` イベントハンドラ関数 (前のコードの `httpStatus`) に渡される値には、現在のロード処理用の HTTP ステータスコード定義が含まれています。SWF ファイルによるテキストファイルのロードが正常に完了すると、`httpStatus` の値が 200 ([OK] の HTTP ステータスコード) に設定されます。このファイルがサーバーにない場合、`httpStatus` の値は 404 (" 見つかりません " の HTTP ステータスコード) に設定されます。2 番目のイベントハンドラである `LoadVars.onLoad()` は、ファイルのロードが終了した後で呼び出されます。ロードが成功した場合、`success` パラメータの値は `true` に設定され、失敗した場合は `false` に設定されます。最後に、外部ファイルが、`LoadVars.load()` メソッドを使用してロードされます。

3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

Flash では、サーバーに存在しないためにロードできないイメージがあると、そのことを示すエラーメッセージが [出力] パネルに表示されます。サーバー上でファイルが見つからないため、`onHTTPStatus` イベントハンドラはステータスコード 404 をトレースします。また、`onLoad` イベントハンドラは、`params_txt` テキストフィールドのテキストプロパティを " テキストファイルのロード不可 " に設定します。

注意	Web サーバーからステータスコードが戻ってこない場合は、`onHTTPStatus` イベントハンドラは数字の 0 を受け取ることになります。

ファイルのアップロードとダウンロードについて

FileReference クラスには、クライアントコンピュータとサーバーとの間でファイルのアップロードおよびダウンロードを行う機能を追加するための手段があります。ユーザーは自分のコンピュータとサーバーとの間で、ファイルを自由にアップロード、ダウンロードできるようになります。ユーザーはアップロードするファイルや、(Windows オペレーティングシステムの [開く] ダイアログボックスなど) ダイアログボックス内でダウンロードする場所を指定するよう要求されます。

ActionScript で作成した各 FileReference オブジェクトは、ユーザーのハードディスクにある同じファイルを参照します。このオブジェクトには、ファイルのサイズ、種類、名前、作成日、および変更日に関する情報が含まれています。Macintosh の場合は、さらにファイルのクリエータタイプに関するプロパティが含まれます。

FileReference クラスのインスタンスを作成する方法は 2 種類あります。次の new 演算子を使用することができます。

```
import flash.net.FileReference;
var myFileReference:FileReference = new FileReference();
```

あるいは、FileReferenceList.browse() メソッドを呼び出します。このメソッドは、ユーザーによるファイルの選択が正常に行われた場合に、ユーザーのシステム上でダイアログボックスを開き、アップロードするファイルを選択し、次に FileReference オブジェクトの配列を作成します。各 FileReference オブジェクトは、ユーザーがダイアログボックスで選択したファイルを表します。FileReference.browse() メソッドまたは FileReferenceList.browse() メソッドが呼び出され、ユーザーがファイルピッカーからファイルを選択するか、FileReference.download() メソッドを使用してファイルピッカーからファイルを選択するまで、FileReference オブジェクトの FileReference プロパティには (name、size、または modificationDate などの) データはありません。

メモ

FileReference.browse() により、ユーザーは 1 つのファイルを選択できます。
FileReferenceList.browse() ではユーザーは複数のファイルを選択できます。

browse() メソッドの呼び出しが正常に終了した後、FileReference.upload() を呼び出して、1 回に 1 つずつファイルをアップロードします。

Flash アプリケーションにダウンロード機能を追加することもできます。FileReference.download() メソッドでは、エンドユーザーはサーバーからのファイルを保存する場所をハードディスクに設定することを要求されます。このメソッドでは、リモート URL からのダウンロードも開始されます。download() メソッドを使用する場合、onSelect イベントが送出されると、FileReference.name プロパティだけにアクセスできるようになります。残りのプロパティについては、onComplete イベントが送出されるまでアクセスできません。

メモ	エンドユーザーのコンピュータにダイアログボックスが表示される場合、ダイアログボックスに表示されるデフォルトの場所は、(その場所が特定できる場合) 最後に選択されたフォルダか、(最後に選択されたフォルダが特定できない場合) デスクトップになります。FileReference API と FileReferenceList API では、デフォルトのファイル位置を設定できません。

FileReference API の機能およびセキュリティの詳細については、580 ページの「FileReference API の機能とセキュリティについて」を参照してください。FileReference API を使用するアプリケーションの例については、581 ページの「アプリケーションへのファイルアップロード機能の追加」を参照してください。この例のサンプルソースファイルである "FileUpload.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/FileUpload" フォルダに移動してサンプルにアクセスします。

FileReference API の各メソッド、プロパティ、およびイベントについては、『ActionScript 2.0 リファレンスガイド』の「`FileReference (flash.net.FileReference)`」および「`FileReferenceList (flash.net.FileReferenceList)`」を参照してください。

FileReference API の機能とセキュリティについて

Flash Player および FileReference API (579 ページの「ファイルのアップロードとダウンロードについて」を参照) ではファイルのアップロードとダウンロードがサポートされます。Flash Player ではアップロードまたはダウンロードできるファイルのサイズに制限はありませんが、このプレーヤーが正式にサポートしているアップロードまたはダウンロードのサイズは最大 100 MB です。FileReference API では、ファイル転送を開始した Flash アプリケーションで次のことを実行することはできません。

- アップロードまたはダウンロードしたファイルにアクセスします。
- ユーザーのコンピュータ上でファイルパスにアクセスします。

サーバーが認証を必要とする場合、唯一正常に実行できる可能性があるのは、Flash Player ブラウザプラグインを使ったファイルのダウンロードです。すべての Flash Player でのアップロードや、スタンドアローンもしくは外部の Flash Player を使ってのダウンロードは、認証を必要とするサーバーでは成功しません。FileReference イベントリスナーを使用して、処理が正常に終了したかどうかを確認し、またはエラー処理を行います。

ファイルのアップロードとダウンロードはいずれも、クロスドメインポリシーファイルを使用して指定したドメインを含む SWF のドメインに制限されています。アップロードないしダウンロードを開始した SWF ファイルがサーバーと同じドメインにある場合、ポリシーファイルはサーバーに置かなければなりません。クロスドメインポリシーファイルおよびセキュリティの詳細については、627 ページの「ネットワーク API の制限」を参照してください。

Mac OS X Flash Player ブラウザプラグイン、Macintosh 外部 Flash Player、および Mac OS X 10.1 以前の Macintosh スタンドアローンプレーヤーでは、`FileReference.browse()`、`FileReferenceList.browse()`、または `FileReference.download()` の呼び出しを実行している間、SWF の再生が一時停止します。SWF ファイルは、すべての Windows プレーヤーと、Mac OS X 10.2 以降の Macintosh スタンドアローン Flash Player で引き続き再生することができます。

警告	ユーザーによるサーバーへのファイルアップロードを認めた場合、ファイルをハードディスクに保存する前に、必ずファイルタイプをチェックする必要があります。たとえば、サーバー上のフォルダやファイルの削除に使用可能なサーバーサイドスクリプトがサーバーにアップロードされることを防止しなければなりません。イメージファイルのアップロードだけをユーザーに許可する場合、ファイルをアップロードするサーバーサイドスクリプトは、アップロード対象のファイルが有効なイメージであることをチェックできるものであることが必要です。

FileReference API を使用するアプリケーションの例については、581 ページの「アプリケーションへのファイルアップロード機能の追加」を参照してください。

アプリケーションへのファイルアップロード機能の追加

次の手順は、サーバーにイメージファイルをアップロードできるアプリケーションの構築方法を説明したものです。このアプリケーションを使用すると、ハードディスク上でイメージを選択し、それをアップロードして、サーバーに送ることができます。アップロードしたイメージは、アップロードに使用した SWF ファイルに表示されます。

Flash アプリケーションを構築する例に続いては、サーバーサイドコードの詳細を説明する例を取り上げます。イメージファイルには、最大で 200K というサイズ制限があることに注意してください。

FileReference API を使用して FLA アプリケーションを構築するには :

1. 新しい Flash ドキュメントを作成し、fileref.fla という名前を付けて保存します。
2. [コンポーネント] パネルを開いて、ScrollPane コンポーネントをステージ上にドラッグし、imagePane というインスタンス名を付けます。ScrollPane インスタンスは、この後のステップで ActionScript を使用してサイズ設定および位置変更を行います。
3. Button コンポーネントをステージまでドラッグし、uploadBtn というインスタンス名を付けます。
4. Label コンポーネント 2 個をステージまでドラッグし、imageLbl および statusLbl というインスタンス名を付けます。
5. ComboBox コンポーネントをステージまでドラッグし、imagesCb というインスタンス名を付けます。
6. TextArea コンポーネントをステージまでドラッグし、statusArea というインスタンス名を付けます。
7. ステージ上に新しいムービークリップシンボルを作成し、そのシンボルを開いて編集を行います (インスタンスをダブルクリックするとシンボル編集モードになります)。

8. ムービークリップ内に新しい静止テキストフィールドを作成し、次のテキストを追加します。

ダウンロードしようとしたファイルはサーバーにありません。

完成したアプリケーションでは、この警告は特に次の理由によって表示されることがあります。

- 他のイメージがアップロードされたため、そのイメージはサーバーのキューから削除されました。
- ファイルサイズが 200K を超えているため、そのイメージはコピーできません。
- ファイルの種類が有効な JPEG、GIF、または PNG ではありません。

メモ	テキストフィールドの幅は ScrollPane インスタンスの幅 (400 ピクセル) よりも小さくなければならず、これを超えた場合、エラーメッセージを確認するには画面を縦にスクロールする必要があります。

9. [ライブラリ] パネルでシンボルを右クリックし、コンテキストメニューから [リンケージ] を選択します。

10. [ActionScript に書き出し] チェックボックスと [最初のフレームに書き出し] チェックボックスを選択し、[識別子] テキストボックスに「Message」と入力します。[OK] をクリックします。

11. タイムラインのフレーム 1 に次の ActionScript を追加します。

メモ	コードコンテンツには、この機能に関する詳細が含まれています。コードの概要はこの例に準拠しています。

```
import flash.net.FileReference;

imagePane.setSize(400, 350);
imagePane.move(75, 25);
uploadBtn.move(75, 390);
uploadBtn.label = "Upload Image";
imageLbl.move(75, 430);
imageLbl.text = "Select Image";
statusLbl.move(210, 390);
statusLbl.text = "Status";
imagesCb.move(75, 450);
statusArea.setSize(250, 100);
statusArea.move(210, 410);

/* リスナーオブジェクトは FileReference イベントを受け取る。*/
var listener:Object = new Object();

/* ユーザーがファイルを選択すると、onSelect() メソッドが呼び出され、FileReference オブ
   ジェクトに渡される。*/
listener.onSelect = function(selectedFile:FileReference):Void {
   /* TextArea をアップロードして、Flash がイメージをアップロードしようとしていることを
   ユーザーに通知する。*/
   statusArea.text += "Attempting to upload " + selectedFile.name + "\n";
   /* ファイルをサーバーの PHP スクリプトにアップロードする。*/
```

```
  selectedFile.upload("http://www.helpexamples.com/flash/file_io/
  uploadFile.php");
};

/* ファイルのアップロードが開始されると、onOpen() メソッドが呼び出され、ファイルのアップ
  ロードが開始されたことがユーザーに通知される。*/
listener.onOpen = function(selectedFile:FileReference):Void {
  statusArea.text += "Opening " + selectedFile.name + "\n";
};

/* ファイルアップロードが終了すると、onComplete() メソッドが呼び出される。*/
listener.onComplete = function(selectedFile:FileReference):Void {
  /* Flash がイメージのアップロードを開始したことをユーザーに通知する。*/
  statusArea.text += "Downloading " + selectedFile.name + " to player\n";
  /* ComboBox コンポーネントにイメージを追加する。*/
  imagesCb.addItem(selectedFile.name);
  /* ComboBox の選択したインデックスを最後に追加したイメージのインデックスに設定する。*/
  imagesCb.selectedIndex = imagesCb.length - 1;
  /* カスタム関数 downloadImage() を呼び出す。*/
  downloadImage();
};

var imageFile:FileReference = new FileReference();
imageFile.addListener(listener);

imagePane.addEventListener("complete", imageDownloaded);
imagesCb.addEventListener("change", downloadImage);
uploadBtn.addEventListener("click", uploadImage);

/* イメージのダウンロードが行われない場合、イベントオブジェクトのトータルプロパティは -1 に
  等しくなる。この場合、ユーザーにメッセージを表示する。*/
function imageDownloaded(event:Object):Void {
  if (event.total == -1) {
    imagePane.contentPath = "Message";
  }
}

/* ユーザーが ComboBox からイメージを選択するか、downloadImage() 関数が
  listener.onComplete() メソッドにより直接呼び出された場合、イメージのダウンロードを開
  始するために、ScrollPane の contentPath が設定される。*/
function downloadImage(event:Object):Void {
  imagePane.contentPath = "http://www.helpexamples.com/flash/file_io/images/
  " + imagesCb.value;
}

/* ユーザーがボタンをクリックすると、uploadImage() 関数が呼び出され、[ ファイルブラウザ ]
  ダイアログボックスが開く。*/
function uploadImage(event:Object):Void {
  imageFile.browse([{description: "Image Files", extension:
  "*.jpg;*.gif;*.png"}]);
}
```

この ActionScript コードにより、最初に FileReference クラスが読み込まれ、ステージ上で各コンポーネントの初期化、配置、およびサイズ変更が実行されます。次に、リスナーオブジェクトが 1 つと、`onSelect`、`onOpen`、および `onComplete` という 3 つのイベントハンドラが定義されます。さらに、リスナーオブジェクトは、`imageFile` という新しい FileReference オブジェクトに追加されます。次に、イベントリスナーが、`imagePane` ScrollPane インスタンス、`imagesCb` ComboBox インスタンス、および `uploadBtn` Button インスタンスに追加されます。各イベントリスナー関数は、このコードセクションを順守するコードに定義されます。

最初の関数 `imageDownloaded()` は、ダウンロードしたイメージの合計バイト数が -1 かどうかをチェックし、-1 である場合、ScrollPane インスタンスの `contentPath` を、前のステップで作成したメッセージのリンケージ識別子を持つムービークリップに設定します。2 番目の関数 `downloadImage()` は、アップロードされた最新のイメージを ScrollPane インスタンスにダウンロードするためのものです。イメージのダウンロードが終了すると、あらかじめ定義してあった `imageDownloaded()` 関数がトリガされ、イメージが正常にダウンロードされたかどうかのチェックが実行されます。最後の関数 `uploadImage()` は、すべての JPEG、GIF、および PNG イメージをフィルタリングするファイルブラウザダイアログボックスを開きます。

12. ドキュメントの変更を保存します。

13. [ファイル]-[パブリッシュ] を選択し、次に [形式] タブを選択し、Flash と HTML が共に選択されていることを確認します。

14. (オプション) [パブリッシュ設定] ダイアログボックスで [Flash] タブを選択し、次に [ローカルでの再生に関するセキュリティ] ポップアップメニューから [ネットワークにのみアクセスする] を選択します。

    このステップが完了すると、ローカルブラウザでドキュメントをテストしても、セキュリティ制限に違反することはなくなります。

15. [パブリッシュ設定] ダイアログボックスで、[パブリッシュ] をクリックして HTML ファイルと SWF ファイルを作成します。

    作成が完了したら次の手順に進み、SWF ファイル用のコンテナを作成します。

この例のサンプルソースファイルである "FileUpload.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/FileUpload" フォルダに移動してサンプルにアクセスします。

次の手順では、Web サーバーに PHP をインストールし、"images" と "temporary" という名前のサブフォルダへの書き込み許可を得ていることが必要です。最初に、前の手順を完了するか、前述のフォルダにある完成した SWF ファイルを使用します。

イメージアップロードアプリケーションのサーバーサイドスクリプトを作成するには :

1. Dreamweaver や Notepad などのテキストエディタを使用して、新しい PHP ドキュメントを作成します。

2. ドキュメントに次の PHP コードを追加します。コードの概要はこのスクリプトに準拠しています。

```
<?php

$MAXIMUM_FILESIZE = 1024 * 200; // 200KB
$MAXIMUM_FILE_COUNT = 10; // サーバー上のファイル数を最大 10 にする
echo exif_imagetype($_FILES['Filedata']);
if ($_FILES['Filedata']['size'] <= $MAXIMUM_FILESIZE) {
  move_uploaded_file($_FILES['Filedata']['tmp_name'], "./temporary/
   ".$_FILES['Filedata']['name']);
  $type = exif_imagetype("./temporary/".$_FILES['Filedata']['name']);
  if ($type == 1 || $type == 2 || $type == 3) {
    rename("./temporary/".$_FILES['Filedata']['name'], "./images/
   ".$_FILES['Filedata']['name']);
  } else {
    unlink("./temporary/".$_FILES['Filedata']['name']);
  }
}
$directory = opendir('./images/');
$files = array();
while ($file = readdir($directory)) {
  array_push($files, array('./images/'.$file, filectime('./images/'.$file)));
}
usort($files, sorter);
if (count($files) > $MAXIMUM_FILE_COUNT) {
  $files_to_delete = array_splice($files, 0, count($files) -
   $MAXIMUM_FILE_COUNT);
  for ($i = 0; $i < count($files_to_delete); $i++) {
    unlink($files_to_delete[$i][0]);
  }
}
print_r($files);
closedir($directory);

function sorter($a, $b) {
  if ($a[1] == $b[1]) {
    return 0;
  } else {
    return ($a[1] < $b[1]) ? -1 : 1;
  }
}
?>
```

この PHP コードにより、最初に $MAXIMUM_FILESIZE と $MAXIMUM_FILE_COUNT という 2 つの定数変数が定義されます。これらの変数は、サーバーにアップロードされるイメージの最大サイズ (200KB) のほか、"images" フォルダに保存する最新のアップロードファイルの件数 (10) を決定します。最新のアップロードファイルのサイズが $MAXIMUM_FILESIZE の値以下である場合、そのイメージは "temporary" フォルダに移されます。

次に、アップロードファイルのタイプがチェックされ、イメージが JPEG、GIF、または PNG であることの確認が行われます。互換性のあるイメージタイプである場合、そのイメージは "temporary" フォルダから "images" フォルダにコピーされます。イメージタイプが認められていないものである場合、そのイメージはファイルシステムから削除されます。

次に、"images" フォルダのディレクトリ一覧が作成され、while ループによってループ処理が行われます。"images" フォルダの各ファイルは配列に追加され、ソートされます。現在、"images" フォルダにあるファイルの数が `$MAXIMUM_FILE_COUNT` 値を超えている場合、この値と同じになるまでファイルが削除されます。こうすることで、最大ファイル数が 10 で、各イメージのサイズが 200KB 以下 (したがって、イメージの総バイト数は約 2MB) と定められている "images" フォルダが管理不能になるまで大きくなるのを防止することができます。

3. PHP ドキュメントの変更を保存します。
4. Web サーバーに、SWF、HTML、および PHP ファイルをアップロードします。
5. リモート HTML ドキュメントを Web ブラウザに表示し、SWF ファイルの [イメージをアップロード] ボタンをクリックします。
6. ハードディスクにイメージファイルを配置し、ダイアログボックスで [開く] を選択します。

   イメージファイルがリモート PHP ドキュメントにアップロードされ、ScrollPane に表示されます (必要に応じて、スクロールバーが追加されます)。既にアップロードしてあるイメージを表示する場合は、ステージの ComboBox インスタンスから [ファイル名] を選択します。認められていないタイプ (認められているのは JPEG、GIF、PNG の 3 種類のみ) のイメージをアップロードしようとした場合や、ファイルサイズが大きすぎる (200KB を超える) 場合、[ライブラリ] の [メッセージ] ムービークリップからエラーメッセージが発信されます。

この例のサンプルソースファイルである "FileUpload.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/FileUpload" フォルダに移動してサンプルにアクセスします。

ローカルファイルセキュリティの詳細については、611 ページの「ローカルファイルセキュリティ と Flash Player について」を参照してください。

PHP の記述の詳細については、www.php.net/ を参照してください。

XML について

XML (拡張マークアップ言語) は、インターネットアプリケーションにおける構造化データ交換の標準になりつつあります。XML テクノロジーを使用してチャットシステムや株式仲買システムなどの高度アプリケーションを構築するサーバーに、Flash 内のデータを統合することができます。

XML では、HTML の場合と同様に、タグを使用してテキスト本体を " マークアップ "、つまり指定することができます。HTML では、定義済みのタグを使用して、Web ブラウザにテキストを表示する方法を指示することができます。たとえば、`<b>` タグでは、テキストをボールドにすることを指示します。XML では、データのタイプ (`<password>VerySecret</password>` など) を識別するタグを定義します。XML では、情報の構造と表示方法とを分けており、異なる環境で同じ XML ドキュメントを使用および再利用できます。

すべての XML タグは、" ノード " またはエレメントと呼ばれます。各ノードにはタイプ (1 - XML エレメントを示す、または 3 - テキストノードを示す) があり、またエレメントにも属性があります。ノードにネストされたノードは、" 子ノード " と呼ばれます。このノードの階層ツリー構造は、XML DOM (Document Object Model : ドキュメントオブジェクトモデル) と呼ばれます。これは、Web ブラウザ内のエレメントの構造の JavaScript DOM とよく似ています。

次の例で、`<portfolio>` は親ノードです。このノードには属性がなく、`symbol`、`qty`、`PRICE`、および `value` という属性を持つ子ノード `<holding>` があります。

```
<portfolio>
  <holding symbol="rich"
    qty="75"
    price="245.50"
    value="18412.50" />
</portfolio>
```

詳細については、次に示す各トピックを参照してください。

- 588 ページの「XML クラスの使用」
- 593 ページの「XMLSocket クラスの使用」

XML の詳細については、www.w3.org/XML を参照してください。

XML データをロードし、解析し、および操作する Web ログトラッカーの作成方法を示すサンプルのソースファイル "xml_blogTracker.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/XML_BlogTracker" フォルダに移動してサンプルにアクセスします。

XML とネストされた配列を使用し、さまざまな言語のストリングを選択してテキストフィールドに入力する方法を示す、サンプルのソースファイル "xml_languagePicker.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/XML_LanguagePicker" フォルダに移動してサンプルにアクセスします。

XML データを使用して動的にメニューを作成する方法を示すサンプルのソースファイル "xmlmenu.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/XML_Menu" フォルダに移動してサンプルにアクセスします。このサンプルでは、ActionScript `XmlMenu()` コンストラクタを呼び出し、XML メニューファイルへのパスおよび現在のタイムラインへの参照をパラメータとして渡します。残りの機能は、カスタムクラスファイル "XmlMenu.as" に存在します。

XML クラスの使用

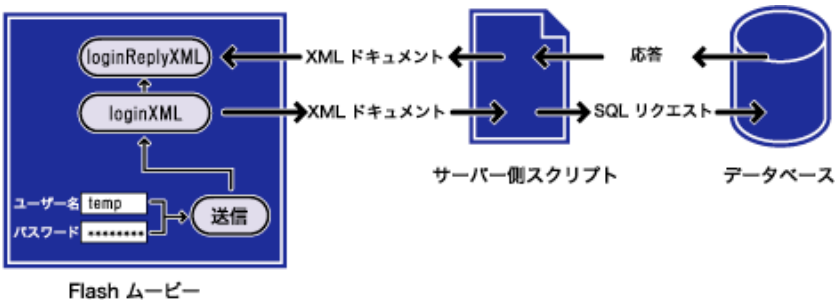
ActionScript XML クラスのメソッド (`appendChild()`、`removeNode()`、`insertBefore()` など) を使用すると、サーバーに送る XML データを Flash 内で構造化したり、ダウンロードされた XML データを処理および解釈することができます。

次の XML クラスメソッドでは、HTTP `POST` メソッドを介してサーバーに XML データを送信およびロードすることができます。

- `load()` メソッドは、URL から XML をダウンロードして ActionScript XML オブジェクトに設定します。
- `send()` メソッドは、XML オブジェクトを XML ドキュメントにエンコードし、`POST` メソッドを使用して指定された URL に送ります。指定されている場合、返されたデータがブラウザウィンドウに表示されます。
- `sendAndLoad()` メソッドは、URL に XML オブジェクトを送ります。返された情報は、ActionScript XML オブジェクトに設定されます。

たとえば、すべての情報 (ユーザー名、パスワード、セッション ID、ポートフォリオの所有権、取引情報など) をデータベースに格納する株式仲買システムを作成できます。

Flash とデータベースの間で情報を受け渡しするサーバーサイドスクリプトは、XML 形式でデータを読み書きします。ActionScript を使用すると、SWF ファイルに収集された情報 (ユーザー名やパスワードなど) を XML オブジェクトに変換し、次に XML ドキュメントとしてデータをサーバーサイドスクリプトに送ることができます。また、ActionScript を使用すると、サーバーが返す XML ドキュメントを SWF ファイルで使用される XML オブジェクトにロードすることもできます。



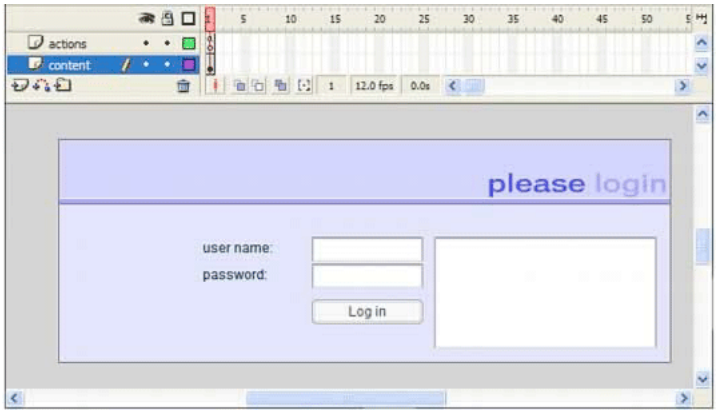
SWF ファイル、サーバーサイドスクリプト、およびデータベース間でのデータのフローと変換

株式仲買システムのパスワード検証では、2 つのスクリプトを必要とします。つまり、フレーム 1 上で定義された関数と、ドキュメント内で作成された XML オブジェクトを送り出すスクリプトです。

ユーザーが自分の情報を変数 username および password として SWF ファイルのテキストフィールドに入力するときは、サーバーに変数を渡す前に、変数を XML に変換する必要があります。スクリプトの最初のセクションでは、新たに作成された XML オブジェクト loginXML に変数がロードされます。ユーザーがログイン用のボタンをクリックすると、loginXML オブジェクトが XML のストリングに変換され、サーバーに送られます。

次の ActionScript はタイムライン上に配置され、XML 形式のデータをサーバーに送信するときに使用されます。このスクリプトを理解するには、次の // で示されるコメント行を読んでください。

```
//XML の空白を無視
XML.prototype.ignoreWhite = true;
// サーバーの応答を保持する XML オブジェクトを作成する
var loginReplyXML:XML = new XML();
// サーバーから XML パケットを受け取った場合はこの関数がトリガされる
loginReplyXML.onLoad = function(success:Boolean) {
  if (success) {
    //( オプション ) ステータス / デバッグ用に 2 つのテキストフィールドを作成
    // status_txt.text = this.firstChild.attributes.status;
    // debug_txt.text = this.firstChild;
    switch (this.firstChild.attributes.STATUS) {
    case 'OK' :
      _global.session = this.firstChild.attributes.SESSION;
      trace(_global.session);
      gotoAndStop("welcome");
      break;
    case 'FAILURE' :
      gotoAndStop("loginfailure");
      break;
    default :
      // 次のコードが実行されてはならない
      trace("Unexpected value received for STATUS.");
    }
  } else {
    trace("an error occurred.");
  }
};
// login_btn がクリックされると次の関数が呼び出される
login_btn.onRelease = function() {
  var loginXML:XML = new XML();
  // サーバーに送信する XML 形式のデータを作成
  var loginElement:XMLNode = loginXML.createElement("login");
  loginElement.attributes.username = username_txt.text;
  loginElement.attributes.password = password_txt.text;
  loginXML.appendChild(loginElement);
  // XML 形式のデータをサーバーに送信
  loginXML.sendAndLoad("http://www.flash-mx.com/mm/main.cfm", loginReplyXML);
};
```



このコードは、ユーザー名 JeanSmith、パスワード VerySecret でテストできます。スクリプトの最初のセクションでは、ユーザーがログインボタンを押したときに、次の XML が生成されます。

```
<login username="JeanSmith" password="VerySecret" />
```

サーバーは XML を受信し、XML 応答を生成し、SWF ファイルに送り返します。パスワードが受け入れられると、サーバーは次のように応答します。

```
<LOGINREPLY STATUS="OK" SESSION="4D968511" />
```

XML には、ランダムに生成された一意のセッション ID を含む session 属性があります。このセッション ID は、以降のセッションでクライアントとサーバー間のすべての通信に使用されます。パスワードが拒否されると、サーバーは次のメッセージを返します。

```
<LOGINREPLY STATUS="FAILURE" />
```

loginreply XML ノードは、SWF ファイル内の空白の XML オブジェクトにロードする必要があります。次のステートメントでは、XML ノードを受信するための XML オブジェクト loginreplyXML が作成されます。

```
// サーバーの応答を保持する XML オブジェクトを作成する
var loginReplyXML:XML = new XML();
loginReplyXML.onLoad = function(success:Boolean) {
```

この ActionScript の 2 番目のステートメントは、onLoad イベントが発生したときに呼び出される匿名 (インライン) 関数を定義しています。

ログインボタン (login_btn インスタンス) を使用して、ユーザー名とパスワードを XML としてサーバーに送信し、返ってきた XML 応答を SWF ファイルにロードします。これを行うには、次の例のように sendAndLoad() メソッドを使用します。

```
loginXML.sendAndLoad("http://www.flash-mx.com/mm/main.cfm", loginReplyXML);
```

最初に、ユーザーが SWF ファイルで入力した値を使用して XML 形式のデータが作成され、次に sendAndLoad メソッドで、その XML オブジェクトが送信されます。loadVariables() 関数のデータと同様、loginreply XML エレメントは非同期的 (つまり、結果が返されるまで待機しない) に着信し、loginReplyXML オブジェクトにロードされます。データが着信すると、loginReplyXML オブジェクトの onLoad ハンドラが呼び出されます。onLoad ハンドラがトリガされたときに呼び出される loginReplyXML 関数を定義し、loginreply エレメントを処理できるようにしておく必要があります。

メモ	この関数は必ず、ログインボタンの ActionScript を格納するフレームに配置する必要があります。

ログインに成功した場合、SWF ファイルが、welcome フレームラベルに進みます。ログインに失敗した場合、再生ヘッドが、loginfailure フレームラベルに移動します。この処理は、条件分岐ステートメントを使用して行われています。

CFM ファイルには次のコードが含まれます。

```
<cfif (Compare(Form.username, "Herbert") EQ 0) AND (Compare(Form.password,
   "glasses") EQ 0)>
   <cfoutput>&isValidLogin=1&session=#URLEncodedFormat(CreateUUID())#</cfoutput>
   <cfelse>
   <cfoutput>&isValidLogin=0</cfoutput>
</cfif>
```

case ステートメントおよび break ステートメントの詳細については、『ActionScript 2.0 リファレンスガイド』の「case ステートメント」および「break ステートメント」を参照してください。条件の詳細については、『ActionScript 2.0 リファレンスガイド』の「if ステートメント」および「else ステートメント」を参照してください。

メモ	このデザインは一例にすぎません。アドビ システムズ社では、提供するセキュリティのレベルについて一切主張しません。安全なパスワード保護システムを実現するには、ネットワークセキュリティについて十分理解する必要があります。

詳細については、http://www.adobe.com/support/flash/applications/xml/、および『ActionScript 2.0 リファレンスガイド』の「XML」を参照してください。ローカルファイルセキュリティの詳細については、611 ページの「ローカルファイルセキュリティ と Flash Player について」を参照してください。

ActionScript 2.0 を使用して Web サイトに簡単なログイン機能を追加する方法を示す、サンプルのソースファイル "login.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/Login" フォルダに移動してサンプルにアクセスします。このサンプルでは、ActionScript とコンポーネントを使用して、ユーザー名とパスワードを入力した後、ボタンをクリックしてサイトに入る小さなフォームを作成します。

Flash Player 8 以降では、XML クラス、LoadVars クラス、および MovieClipLoader クラスの onHTTPStatus イベントハンドラがサポートされるため、ユーザーが HTTP 要求からのステータスコードにアクセスできます。これにより、開発者は単にロード処理が失敗したという結果についてだけでなく、ロード処理が失敗した理由を確認できるようになります。

次の例は、XML クラスの onHTTPStatus イベントハンドラを使用して、サーバーからの XML ファイルのダウンロードが正常に完了したかどうかや、HTTP 要求から戻されたステータスコードを確認することができます。

XML クラスを使用した HTTP ステータスコードの確認

1. 新しい Flash ドキュメントを作成し、xmlhttp.fla という名前で保存します。
2. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onHTTPStatus = function(httpStatus:Number) {
   trace("HTTP status is: " + httpStatus);
};
my_xml.onLoad = function(success:Boolean) {
   if (success) {
      trace("XML successfully loaded");
      // 0 エラーなし。解析が正常に終了しました。
      trace("XML status is: " + my_xml.status);
   } else {
      trace("unable to load XML");
   }
};
my_xml.load("http://www.helpexamples.com/crossdomain.xml");
```

   前記のコードにより、変数 my_xml のある新しい XML オブジェクトが 1 つと、2 つのイベントハンドラ (onHTTPStatus と onLoad) が定義され、外部 XML ファイルが 1 つロードされます。onLoad イベントハンドラが、XML ファイルが正常にロードされたかどうかをチェックし、正常にロードされた場合は、[出力] パネルにメッセージを送り、XML オブジェクトのステータスプロパティをトレースします。onHTTPStatus イベントリスナーが、Web サーバーから返ってきたステータスコードを送り返すのに対し、XML.status プロパティには、XML オブジェクトが正常に解析できるかどうかを示す数値が含まれている点に注意する必要があります。
3. [制御]-[ムービープレビュー] を選択して Flash ドキュメントをテストします。

> **警告**
> HTTP httpStatus コードと、XML クラスの status プロパティとを混同しないよう注意してください。onHTTPStatus イベントハンドラは、HTTP 要求からのサーバーステータスコードを返し、status プロパティは、XML ドキュメントの XML オブジェクトへの解析が成功したかどうかを示す数値を自動的に設定し、返します。

> **注意**
> Web サーバーから status コードが返ってこない場合、onHTTPStatus イベントハンドラは数字 0 を受け取ることになります。

XML データをロードし、解析し、および操作する Web ログトラッカーの作成方法を示すサンプルのソースファイル "xml_blogTracker.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/XML_BlogTracker" フォルダに移動してサンプルにアクセスします。

XML とネストされた配列を使用し、さまざまな言語のストリングを選択してテキストフィールドに入力する方法を示す、サンプルのソースファイル "xml_languagePicker.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/XML_LanguagePicker" フォルダに移動してサンプルにアクセスします。

XML データを使用して動的にメニューを作成する方法を示すサンプルのソースファイル "xmlmenu.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/XML_Menu" フォルダに移動してサンプルにアクセスします。このサンプルでは、ActionScript `XmlMenu()` コンストラクタを呼び出し、XML メニューファイルへのパスおよび現在のタイムラインへの参照をパラメータとして渡します。残りの機能は、カスタムクラスファイル "XmlMenu.as" に存在します。

XMLSocket クラスの使用

ActionScript には、サーバーとの連続的な接続を開くことのできる XMLSocket ビルトインクラスがあります。ソケット接続により、サーバーは情報が使用可能になった時点でクライアントに情報をパブリッシュ (または " プッシュ ") することができます。連続的な接続がない場合、サーバーは HTTP 要求を待つ必要があります。この開いた接続は待ち時間の問題を解消し、一般にはチャットなどのリアルタイムアプリケーションに使用されます。データは 1 つのストリングとしてソケット接続を介して送られますが、XML 形式にする必要があります。XML クラスを使用すると、データを構造化することができます。

ソケット接続を構築するには、ソケット接続の要求を待って SWF ファイルに応答を送るためのサーバーサイドアプリケーションを作成する必要があります。このようなサーバーサイドアプリケーションは、Java などのプログラミング言語で書くことができます。

メモ	XMLSocket クラスは、ファイアウォールをトンネル化によって自動的に通過することはできません。RTMP プロトコルとは異なり、XMLSocket には HTTP トンネル化機能が備わっていないためです。HTTP トンネル化が必要な場合は、Flash Remoting または (RTMP をサポートする) Flash Media Server を使用するようにしてください。

XMLSocket クラスの `connect()` メソッドと `send()` メソッドを使用すると、ソケット接続を介してサーバーとの間で XML を転送することができます。`connect()` メソッドは、Web サーバーポートとのソケット接続を確立します。`send()` メソッドは、ソケット接続で指定されたサーバーに XML オブジェクトを渡します。

connect() メソッドを呼び出すと、Flash Player はサーバーへの TCP/IP 接続を開き、次のいずれかのイベントが発生するまでその接続を開いたままにします。

- XMLSocket クラスの close() メソッドが呼び出される。
- XMLSocket オブジェクトへの参照がそれ以上存在しなくなる。
- Flash Player が終了する。
- 接続が切れる (モデムの切断など)。

次の例では、XML ソケット接続を構築し、XML オブジェクト myXML からデータを送ります。このスクリプトを理解するには、次の // で示されるコメント行を読んでください。

```
// XMLSocket オブジェクトを作成する
var theSocket:XMLSocket = new XMLSocket();
// サイト (1024 番以降の未使用ポート ) に対し、connect() メソッドを使用して接続する
// ローカルテストの場合は「localhost」または「127.0.0.1」と入力する
// 実運用サーバーの場合は、ドメイン www.yourdomain.com を入力する
theSocket.connect("localhost", 12345);
// 接続に関するテキストを表示する
theSocket.onConnect = function(myStatus) {
  if (myStatus) {
    conn_txt.text = "connection successful";
  } else {
    conn_txt.text = "no connection made";
  }
};
// 送信するデータ
function sendData() {
  var myXML:XML = new XML();
  var mySend = myXML.createElement("thenode");
  mySend.attributes.myData = "someData";
  myXML.appendChild(mySend);
  theSocket.send(myXML);
}
// データ送信用ボタン
sendButton.onRelease = function() {
  sendData();
};
// ソケット接続から返されたデータをトレースする
theSocket.onData = function(msg:String):Void {
  trace(msg);
};
```

詳細については、『ActionScript 2.0 リファレンスガイド』の「XMLSocket」を参照してください。

ローカルファイルセキュリティの詳細については、611 ページの「ローカルファイルセキュリティ と Flash Player について」を参照してください。

Flash Player とのメッセージの送受信

メッセージを SWF ファイルからホスト環境 (Web ブラウザ、Macromedia Director ムービー、スタンドアローン Flash Player など) に送るには、`fscommand()` 関数を使用します。この関数を使用すると、ホストの機能を使用して SWF ファイルを拡張できます。たとえば、HTML ページの JavaScript 関数に `fscommand()` 関数を渡し、特定のプロパティを持つ新しいブラウザウィンドウを開くことができます。

JavaScript、VBScript、Microsoft JScript などの Web ブラウザスクリプト言語から Flash Player 内の SWF ファイルをコントロールするには、ホスト環境から SWF ファイルにメッセージを送る関数の Flash Player メソッドを使用します。たとえば、特定のフレームに SWF ファイルを送るリンクを HTML ページに設定することができます。

詳細については、次のトピックを参照してください。

- 595 ページの「fscommand() 関数の使用」
- 598 ページの「JavaScript を使用した Flash アプリケーションの制御について」
- 598 ページの「Flash Player メソッドについて」

fscommand() 関数の使用

メモ

External API は、Flash 8 以降の HTML ページまたはコンテナアプリケーションとの相互動作用 `fscommand()` に代わる機能です。同じ状況であれば、External API の方が `fscommand()` よりも確実に機能します。詳細については、599 ページの「External API について」を参照してください。

Web ブラウザなど、Flash Player をホスティングするプログラムにメッセージを送るには、`fscommand()` 関数を使用します。

メモ

`fscommand()` を使用した JavaScript 呼び出しは、Safari や Macintosh 版の Internet Explorer では正しく動作しません。

`fscommand()` 関数には、*`command`* および *`arguments`* の 2 つのパラメータがあります。Flash Player のスタンドアローンバージョンにメッセージを送るには、コマンドと引数を使用する必要があります。たとえば、次のイベントハンドラは、ボタンを離したときに SWF ファイルをフルスクリーンに拡大するようにスタンドアローンプレーヤーを設定します。

```
my_btn.onRelease = function() {
  fscommand("fullscreen", true);
};
```

次の表は、プロジェクタなどのスタンドアローンプレーヤーで SWF ファイルの再生および外観を制御する場合に、fscommand( ) のパラメータ *command* と *arguments* に対して指定できる値の一覧です。

メモ	プロジェクタは、スタンドアローンアプリケーションとして実行可能な形式で保存された SWF ファイルです。つまり、コンテンツと共に実行可能ファイルに埋め込まれた Flash Player です。

コマンド	引数	用途
quit	None	プロジェクタを終了します。
fullscreen	true または false	true を指定すると、Flash Player はフルスクリーンモードに設定されます。false に設定すると、プレーヤーはノーマルメニュービューになります。
allowscale	true または false	false を指定すると、SWF ファイルは常に元のサイズで表示され、拡大・縮小されないようにプレーヤーが設定されます。true を指定すると、SWF ファイルは強制的にプレーヤーのウィンドウサイズに対して 100% に拡大・縮小されます。
showmenu	true または false	true を指定すると、すべてのコンテキストメニュー項目が有効になります。false を指定すると、[設定] および [Flash Player について] 以外のすべてのコンテキストメニュー項目が省略されます。
exec	アプリケーションへのパス	プロジェクタの内部からアプリケーションを実行します。

fscommand( ) を使用して、Web ブラウザの JavaScript などのスクリプト言語にメッセージを送るには、パラメータ *command* と *arguments* 内に 2 つのパラメータを渡します。これらのパラメータはストリングと式のどちらでもよく、fscommand( ) 関数を " キャッチする "、つまり処理する JavaScript 関数で使用されます。

fscommand( ) 関数は、SWF を埋め込んだ HTML ページで JavaScript 関数 *moviename*_DoFSCommand を呼び出します。ここで、*moviename* は embed タグの name 属性または object タグの id 属性で割り当てられた Flash Player の名前です。SWF ファイルに myMovie という名前が割り当てられている場合、呼び出される JavaScript 関数は myMovie_DoFSCommand です。

JavaScript 経由で HTML ページの SWF ファイルからメッセージボックスを開くために fscommand() を使用するには :

1. 新しい FLA ファイルを作成し、myMovie.fla という名前で保存します。
2. Button コンポーネントの 2 つのインスタンスをステージ上にドラッグし、それぞれ window_btn および alert_btn というインスタンス名を付け、ラベルを Open Window と Alert と指定します。
3. タイムラインに新しいレイヤーを挿入し、名前を Actions に変更します。

4. Actions レイヤーのフレーム 1 を選択し、次の ActionScript を [アクション] パネルに追加します。

```
window_btn.onRelease = function() {
   fscommand("popup", "http://www.adobe.com/");
};
alert_btn.onRelease = function() {
   fscommand("alert", "You clicked the button.");
};
```

5. [ファイル]-[パブリッシュ設定] を選択し、[HTML] タブの [テンプレート] メニューで [Flash (FSCommand サポート)] が選択されていることを確認します。

6. [ファイル]-[パブリッシュ] を選択して、SWF ファイルおよび HTML ファイルを生成します。

7. HTML エディタまたはテキストエディタを使用し、手順 6 で生成された HTML ファイルを開いてコードを検証します。[パブリッシュ設定] ダイアログボックスの [HTML] タブで Flash (FSCommand サポート) テンプレートを使用して SWF ファイルをパブリッシュした場合、HTML ファイルに一部コードが追加されます。SWF ファイルの NAME と ID 属性には、ファイル名が使用されます。たとえば、ファイル "myMovie.fla" の場合、これらの属性は myMovie に設定されます。

8. HTML ファイルの、"// コードをここに配置します。" と表示された箇所に、次の JavaScript コードを追加します。

```
if (command == "alert") {
   alert(args);
} else if (command == "popup") {
   window.open(args, "mmwin", "width=500,height=300");
}
```

パブリッシュの詳細については、『Flash ユーザーガイド』を参照してください。

その他、Microsoft Internet Explorer アプリケーションの場合には、次の例のように、<SCRIPT> タグでイベントハンドラを直接割り当てることができます。

```
<script Language="JavaScript" event="FSCommand (command, args)"
   for="theMovie">
...
</script>
```

9. HTML ファイルを保存して閉じます。

このように、Flash の外部で HTML ファイルを編集する場合は、[ファイル]-[パブリッシュ設定] で [HTML] チェックボックスの選択を解除する必要があります。選択を解除しないと、再度パブリッシュしたときに、HTML コードが Flash によって上書きされます。

10. Web ブラウザで、HTML ファイルを開いて表示します。[ウィンドウを開く] ボタンをクリックすると、ウィンドウが開き、Macromedia の Web サイトが表示されます。[警告] ボタンをクリックすると、警告ウィンドウが開きます。

fscommand() 関数は、Lingo によりストリング、イベント、実行可能 Lingo コードのいずれかと解釈されるメッセージを Macromedia Director に送ることができます。メッセージがストリングまたはイベントである場合、fscommand() 関数からメッセージを受信するための Lingo コードを書き、Director 内でアクションを実行する必要があります。詳細については、www.adobe.com/support/director の Director サポートセンターを参照してください。

ActiveX コントロールに対応する Visual Basic や Visual C++ などのプログラムでは、fscommand() は、その環境のプログラミング言語で処理できる 2 つのストリングを含む VB イベントを送信します。詳細については、www.adobe.com/support/flash の Flash サポートセンターで「Flash メソッド」というキーワードを使用して検索してください。

JavaScript を使用した Flash アプリケーションの制御について

Flash Player 6 (6.0.40.0) 以降では、Flash アプリケーションに固有の JavaScript メソッドのほか、Netscape 6.2 以降で動作する FSCommand もサポートしています。以前のバージョンは、Netscape 6.2 以降に対して、これらの JavaScript メソッドと FSCommand をサポートしません。詳細については、Adobe サポートセンターの記事「Scripting With Flash」(www.adobe.com/support/flash/publishexport/scriptingwithflash/) を参照してください。

Netscape 6.2 以降の場合は、swliveconnect を true に設定する必要はありません。ただし、swLiveConnect を true に設定しても、SWF ファイルに対する影響はありません。詳細については、『Flash ユーザーガイド』の swLiveConnect 属性に関する記述を参照してください。

Flash Player メソッドについて

Flash Player メソッドを使用すると、JavaScript や VBScript などの Web ブラウザスクリプト言語から Flash Player 内の SWF ファイルをコントロールできます。他のメソッドと同様に、Flash Player メソッドを使用して、ActionScript 以外のスクリプト環境から SWF ファイルへ呼び出しを送信することができます。各メソッドには名前があり、ほとんどのメソッドはパラメータを取ります。パラメータは、メソッドが処理する値を指定します。メソッドで計算を実行すると、スクリプト環境で使用できる値が返されます。

ブラウザと Flash Player との通信を可能にするテクノロジーには、LiveConnect (Windows 95/98/2000/NT/XP または Power Macintosh の Netscape Navigator 3.0 以降)、および ActiveX (Windows 95/98/2000/NT/XP の Internet Explorer 3.0 以降) の 2 種類があります。スクリプティングのテクニックはすべてのブラウザおよび言語と同様ですが、ActiveX コントロールで使用できるプロパティとイベントが追加されています。

すべての Flash Player スクリプトメソッドの一覧とその他の詳細については、www.adobe.com/jp/support/flash の Flash サポートセンターから「Flash メソッド」というキーワードで検索してください。

External API について

ExternalInterface クラスは External API とも呼ばれます。これは、External API であり、JavaScript により ActionScript や Flash Player コンテナ、Flash Player を埋め込んだデスクトップアプリケーションと HTML ページ間の通信を可能にする新しいサブシステムです。

メモ

この機能は、HTML ページやコンテナアプリケーションとの相互動作用の古い機能 `fscommand()` に代わるものです。同じ状況であれば、External API の方が `fscommand()` よりも確実に機能します。詳細については、599 ページの「External API について」を参照してください。

ExternalInterface クラスは次の状況でのみ使用できます。

- Windows 用 Internet Explorer の全サポートバージョン (5.0 以降)。
- Flash Player ActiveX コントロールを埋め込むデスクトップアプリケーションなどの埋め込みカスタム ActiveX コンテナ。
- NPRuntime インターフェイスをサポートする全ブラウザ。現時点では、以下のブラウザが含まれます。
  - Firefox 1.0 以降
  - Mozilla 1.7.5 以降
  - Netscape 8.0 以降
  - Safari 1.3 以降

その他の状況では、`ExternalInterface.available` プロパティから false が返されてきます。

ActionScript から、HTML ページの JavaScript 関数を呼び出すことができます。`fscommand()` に比べ、External API の方が次のような高度な機能を備えています。

- `fscommand` 関数で使用可能な関数以外の、あらゆる JavaScript 関数を使用できます。
- 任意の数の引数を任意の名前で渡すことができます。渡すコマンドや引数に制限はありません。
- (ブール、数字、ストリングなど) さまざまなデータタイプを渡すことができます。ストリングパラメータだけに限定されません。
- 呼び出し値を受け取ることができ、その値は (呼び出しの戻り値として) 直ちに ActionScript に戻されます。

HTML ページの JavaScript から、ActionScript 関数を呼び出すことができます。詳細については、「`ExternalInterface (flash.external.ExternalInterface)`」を参照してください。ローカルファイルセキュリティの詳細については、611 ページの「ローカルファイルセキュリティ と Flash Player について」を参照してください。

この後の項では、External API を使用する例を取り上げます。

- 600 ページの「External API を使用したインタラクティブ機能の作成」
- 603 ページの「External API を使用した Flash Video の制御」

External API を使用したインタラクティブ機能の作成

ブラウザと、Web ページに埋め込まれた SWF ファイルとのインタラクションを作成できます。次の手順で、実行時に、SWF ファイルを含む HTML ページにテキストが送られ、HTML ページから SWF ファイルにメッセージ送り返されます。

Flash アプリケーションを作成するには :

1. 新しい Flash ドキュメントを作成し、extint.fla という名前で保存します。
2. TextInput コンポーネント 2 個をステージまでドラッグし、それぞれに in_ti と out_ti というインスタンス名を付けます。
3. Label コンポーネントをステージまでドラッグし、out_lbl というインスタンス名を付け、`out_ti` TextInput インスタンスの上に配置し、プロパティインスペクタの [パラメータ] タブのテキストプロパティを Sending to JS: に設定します。
4. Button コンポーネントをステージまでドラッグし、`out_lbl` ラベルの隣に配置し、send_button というインスタンス名を付けます。
5. Label コンポーネントをステージまでドラッグし、in_lbl というインスタンス名を付け、`in_ti` TextInput インスタンスの上に配置し、[パラメータ] タブのテキストプロパティを Receiving from JS: に設定します。
6. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.external.ExternalInterface;

ExternalInterface.addCallback("asFunc", this, asFunc);
function asFunc(str:String):Void {
  in_ti.text = "JS > Hello " + str;
}

send_button.addEventListener("click", clickListener);
function clickListener(eventObj:Object):Void {
  trace("click > " + out_ti.text);
  ExternalInterface.call("jsFunc", out_ti.text);
}
```

このコードは、3 つのセクションに分かれています。最初のセクションは ExternalInterface クラスを書き込むため、完全修飾のクラス名を使用する必要はありません。2 番目のセクションは、この後で取り上げる例で作成する HTML ドキュメントの JavaScript から呼び出されるコールバック関数 `asFunc()` を定義します。この関数は、ステージ上の TextInput コンポーネント内でテキストを設定するものです。3 番目のセクションは関数を定義し、ユーザーがステージ上で Button コンポーネントインスタンスをクリックしたとき、その関数をイベントリスナーに割り当てるものです。ボタンがクリックされた場合は必ず、SWF ファイルによって HTML ページの `jsFunc()` JavaScript 関数が呼び出され、`out_ti` テキスト入力インスタンスのテキストプロパティが渡されます。

7. [ファイル]-[パブリッシュ設定] を選択し、次に [形式] タブを選択して、Flash と HTML が共に選択されていることを確認します。
8. [パブリッシュ] をクリックして、HTML ファイルと SWF ファイルを作成します。

   パブリッシュが完了したら次の手順に進み、SWF ファイル用のコンテナを作成します。

前記の Flash ドキュメントをテストするには、まず、生成した HTML コードを変更し、新たにいくつかの HTML および JavaScript を追加する必要があります。次の手順は、SWF ファイル用 HTML コンテナの変更方法で、これにより 2 つのファイルが同じブラウザで実行された場合にインタラクションが可能になります。

SWF ファイル用の HTML コンテナを作成するには :

1. 前の手順を完了してください。
2. アプリケーションをパブリッシュしたときに Flash によって作成された "extint.html" ファイルを開きます。

   このファイルは Flash ドキュメントと同じフォルダにあります。
3. 開始と終了の `head` タグ間に、次の JavaScript コードを追加します。

```
<script language="JavaScript">
<!--
  function thisMovie(movieName) {
    var isIE = navigator.appName.indexOf("Microsoft") != -1;
    return (isIE) ?window[movieName] : document[movieName];
  }

  function makeCall(str) {
    thisMovie("extint").asFunc(str);
  }

  function jsFunc(str) {
    document.inForm.inField.value = "AS > Hello " + str;
  }
// -->
</script>
```

   この JavaScript コードは 3 種類のメソッドを定義します。最初のメソッドは埋め込まれた SWF ファイルに、ユーザーのブラウザが Microsoft Internet Explorer (IE) か Mozilla かに基づいて、参照を返します。2 番目の関数 `makeCall()` は、前の例の Flash ドキュメントで定義した `asFunc()` メソッドを呼び出します。`thisMovie()` 関数の `"extint"` パラメータは、埋め込まれた SWF ファイルのオブジェクト ID と埋め込み名を参照します。Flash ドキュメントに別の名前を付けて保存した場合、オブジェクトと埋め込みタグの値に合わせてこのストリングを変更する必要があります。3 番目の関数 `jsFunc()` は、HTML ドキュメントの `inField` テキストフィールドの値を設定します。この関数は、ユーザーが `send_button` Button コンポーネントをクリックしたときに、Flash ドキュメントから呼び出されます。

4. 終了の </body> タグの前に、次の HTML コードを追加します。

```
<form name="outForm" method="POST"
  action="javascript:makeCall(document.outForm.outField.value);">
  Sending to AS:<br />
  <input type="text" name="outField" value="" /><br />
  <input type="submit" value="Send" />
</form>

<form name="inForm" method="POST" action="">
  Receiving from AS:<br />
  <input type="text" name="inField">
</form>
```

この HTML コードは、前の実習の Flash 環境で作成したフォームに類似した 2 つの HTML フォームを作成します。最初のフォームは、outField テキストフィールドの値を、前のステップで定義した makeCall() 関数に送信します。2 番目のフォームは、ユーザーが send_button インスタンスをクリックすると、SWF ファイルから送られてくる値を表示するのに使用します。

5. HTML ドキュメントを保存し、HTML ファイルと SWF ファイルの両方を同じ Web サーバーにアップロードします。

6. Web ブラウザに HTML ファイルを表示し、out_ti TextInput インスタンスにストリングを入力し、[送信] ボタンをクリックします。

   Flash が jsFunc() JavaScript 関数を呼び出し、out_ti テキストフィールドのコンテンツを渡します。その結果、コンテンツは、HTML フォーム inForm の inField 入力テキストフィールドに表示されます。

7. outField HTML テキストフィールドに値を入力し、[送信] ボタンをクリックします。

   Flash が SWF ファイルの asFunc() 関数を呼び出し、その関数によって、in_ti TextInput インスタンスにストリングが表示されます。

サンプルのソースファイル "ExtInt.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードして、"ActionScript2.0/ExternalAPI" フォルダに移動してサンプルにアクセスします。

External API の使用に関する詳しい例については、603 ページの「External API を使用した Flash Video の制御」を参照してください。ローカルファイルセキュリティの詳細については、611 ページの「ローカルファイルセキュリティ と Flash Player について」を参照してください。

メモ	document.getElementById("pluginName") や document.all.pluginName など、プラグインオブジェクトにアクセスするその他のメソッドは使用しないでください。これは、こうしたメソッドが一部のブラウザに対して異なる動作を実行するためです。

External API を使用した Flash Video の制御

次の手順は、HTML ページのコントロールと、HTML テキストフィールドのビデオに関する情報の表示を使用して Flash Video (FLV) ファイルを制御する方法を示しています。この手順では External API を使用して、機能を実現しています。

External API を使用して FLA アプリケーションを構築するには :

1. 新しい Flash ドキュメントを作成し、video.fla という名前を付けて保存します。
2. [ライブラリ] パネルのポップアップメニューから [新規ビデオ] を選択し、新しいビデオシンボルをライブラリに追加します。
3. ビデオシンボルをステージまでドラッグし、selected_video というインスタンス名を付けます。
4. selected_video インスタンスを選択し、次にプロパティインスペクタを選択して、そのインスタンスのサイズを幅 320 ピクセル、高さ 240 ピクセルに変更します。
5. ビデオの位置の *x* 座標と *y* 座標を 0 に設定します。
6. ステージを選択し、次にプロパティインスペクタを使用して、サイズを 320 x 240 ピクセルに変更します。

   これで、ステージとビデオインスタンスのサイズが同じになります。
7. メインタイムラインのフレーム 1 に次の ActionScript を追加します。

```
import flash.external.ExternalInterface;

/* playVideo() と pauseResume() を登録すると、
コンテナ HTML ページの JavaScript から呼び出すことができるようになる。*/
ExternalInterface.addCallback("playVideo", null, playVideo);
ExternalInterface.addCallback("pauseResume", null, pauseResume);

/* ビデオは NetConnection と NetStream オブジェクトを必要とする。*/
var server_nc:NetConnection = new NetConnection();
server_nc.connect(null);
var video_ns:NetStream = new NetStream(server_nc);

// NetStream オブジェクトをステージ上の Video オブジェクトに割り当てると、
NetStream データが Video オブジェクトに表示される。*/
selected_video.attachVideo(video_ns);

/* onStatus() メソッドの自動呼び出しが、
NetStream オブジェクトのステータスアップデート時 ( ビデオ再生の開始など ) に実行される。
自動呼び出しが実行されると、コードプロパティの値が
ExternalInterface 経由で JavaScript updateStatus() 関数を呼び出すことで送られる。*/
video_ns.onStatus = function(obj:Object):Void {
  ExternalInterface.call("updateStatus", "  " + obj.code);
};

function playVideo(url:String):Void {
  video_ns.play(url);
```

```
}

function pauseResume():Void {
   video_ns.pause();
}
```

この ActionScript コードの最初のセクションにより、`playVideo()` と `pauseResume()` という 2 つのコールバック関数が定義されます。この 2 つの関数は、次の手順で JavaScript から呼び出されます。コードの 2 番目のセクションは、ビデオインスタンスと共に FLV ファイルの動的再生に使用する、新しい NetConnection オブジェクトと NetStream オブジェクトを作成するためのものです。

次の手順のコードは、`video_ns` NetStream オブジェクト用の `onStatus` イベントハンドラを定義するためのものです。NetStream オブジェクトのステータスが変更されるたびに、`ExternalInterface.call()` メソッドによりカスタム JavaScript 関数 `updateStatus()` がトリガされます。最後の 2 つの関数 `playVideo()` と `pauseResume()` は、ステージ上でのビデオインスタンス再生の制御用です。この 2 つの関数は、次の手順で JavaScript から呼び出されます。

8. Flash ドキュメントを保存します。
9. [ファイル]-[パブリッシュ設定] を選択し、次に [形式] タブを選択し、HTML と Flash が共に選択されていることを確認します。
10. [パブリッシュ] をクリックして、ハードディスクに SWF ファイルと HTML ファイルをパブリッシュします。

    パブリッシュが完了したら次の手順に進み、SWF ファイル用のコンテナを作成します。

サンプルのソースファイル "external.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/ExternalAPI" フォルダに移動してサンプルにアクセスします。

次の手順で、前の手順において Flash によって生成された HTML コードを変更します。この手順により、SWF ファイルで FLV ファイルを再生するのに必要な JavaScript と HTML が作成されます。

SWF ファイル用のコンテナを作成するには :

1. 前の手順を完了してください。
2. 前の手順の最後のステップでパブリッシュした video.html ドキュメントを開きます。
3. 次のコードに合致するように、既存のコードを変更します。

メモ	次の例で、コードコメントをレビューします。コードの概要はこの例に準拠しています。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
  www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>ExternalInterface</title>

<script language="JavaScript">
  // 変数を使用して、埋め込み SWF ファイルを参照する。
  var flashVideoPlayer;

  /* (<body> タグの onLoad イベントを通じて ) HTML ページがロードされると、
  initialize() 関数が呼び出される。*/
  function initialize() {
    // ブラウザが IE かどうかをチェックする。IE の場合、flashVideoPlayer は
window.videoPlayer である。IE でない場合は document.videoPlayer である。
videoPlayer <object> タグおよび <embed> タグに割り当てられる ID である。*/
    var isIE = navigator.appName.indexOf("Microsoft") != -1;
    flashVideoPlayer = (isIE) ? window['videoPlayer'] :
  document['videoPlayer'];
  }

  /* フォームの再生ボタンをクリックすると、videoStatus テキスト領域がアップデートされ、
  SWF ファイル内で playVideo() 関数が呼び出され、それに FLV ファイルの URL が渡される。
  */
  function callFlashPlayVideo() {
    var comboBox = document.forms['videoForm'].videos;
    var video = comboBox.options[comboBox.selectedIndex].value;
    updateStatus("____" + video + "____");
    flashVideoPlayer.playVideo("http://www.helpexamples.com/flash/video/" +
  video);
  }

  // SWF ファイル内で pauseResume() 関数が呼び出される。
  function callFlashPlayPauseVideo() {
    flashVideoPlayer.pauseResume();
  }

  /* updateStatus() 関数は、NetStream オブジェクトの onStatus() メソッドにより、SWF
  ファイルから呼び出される。*/
  function updateStatus(message) {
    document.forms['videoForm'].videoStatus.value += message + "\n";
  }
</script>
</head>
<body bgcolor="#ffffff" onLoad="initialize();">
```

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.adobe.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="320" height="240" id="videoPlayer"
  align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="video.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="video.swf" quality="high" bgcolor="#ffffff" width="320"
  height="240" name="videoPlayer" align="middle"
  allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
  pluginspage="http://www.adobe.com/go/getflashplayer" />
</object>

<form name="videoForm">
  Select a video:<br />
  <select name="videos">
    <option value="lights_long.flv">lights_long.flv</option>
    <option value="clouds.flv">clouds.flv</option>
    <option value="typing_long.flv">typing_long.flv</option>
    <option value="water.flv">water.flv</option>
  </select>
  <input type="button" name="selectVideo" value="play"
  onClick="callFlashPlayVideo();" />
  <br /><br />

  Playback <input type="button" name="playPause" value="play/pause"
  onClick="callFlashPlayPauseVideo();" />

  <br /><br />
  Video status messages <br />
  <textarea name="videoStatus" cols="50" rows="10"></textarea>
</form>

</body>
</html>
```

この HTML コードは、initialize()、callFlashPlayVideo()、callFlashPlayPauseVideo()、および updateStatus() という 4 種類の JavaScript 関数を定義します。initialize() 関数は、onLoad イベントの body タグ内に呼び出されます。callFlashPlayVideo() 関数と callFlashPlayPauseVideo() 関数は共に、ユーザーが HTML ドキュメント内で [再生] ボタンか [再生 / 一時停止] ボタンを押した場合に呼び出され、SWF ファイルで playVideo() 関数と pauseResume() 関数をトリガします。

最後の関数 `updateStatus()` は、`video_ns` NetStream オブジェクトの `onStatus` イベントハンドラがトリガされると必ず、SWF ファイルによって呼び出されます。この HTML コードは、ユーザーがビデオを選択できるコンボボックスを備えたフォームも定義します。ユーザーがビデオを選択し、[再生] ボタンをクリックするたびに、`callFlashPlayVideo()` JavaScript 関数が呼び出され、それによって SWF ファイルに `playVideo()` 関数が呼び出されます。この関数により、ビデオインスタンスにロードする SWF ファイルの URL が指定されます。ビデオが再生され、NetStream オブジェクトのステータスが変更されると、ステージ上の HTML テキスト領域のコンテンツがアップデートされます。

4. HTML ドキュメントの変更を保存し、HTML ファイルと SWF ファイルの両方を同じ Web サイトにアップロードします。

5. Web サイトからリモート video.html ドキュメントを開き、コンボボックスからビデオを選択し、[再生] ボタンをクリックします。

   選択した FLV ファイルが再生され、HTML ドキュメント内の videoStatus テキスト領域のコンテンツがアップデートされます。

サンプルのソースファイル "external.fla" については、Flash サンプルページ (www.adobe.com/go/learn_fl_samples_jp) を参照してください。"Samples" zip ファイルをダウンロードし解凍して、"ActionScript2.0/ExternalAPI" フォルダに移動してサンプルにアクセスします。

External API の詳細については、『ActionScript 2.0 リファレンスガイド』の「`ExternalInterface (flash.external.ExternalInterface)`」を参照してください。

ローカルファイルセキュリティの詳細については、611 ページの「ローカルファイルセキュリティ と Flash Player について」を参照してください。

メモ	document.getElementById("pluginName") や document.all.pluginName など、プラグインオブジェクトにアクセスするその他のメソッドは使用しないでください。これは、こうしたメソッドが一部のブラウザに対して異なる動作を実行するためです。

第 16 章

セキュリティについて

16

Flash CS3 Professional では、ActionScript を使用して外部ソースから SWF ファイルにデータをロードすることや、サーバーにデータを送信することができます。SWF ファイルにデータをロードするには、Flash セキュリティモデルを理解し、設定する必要があります。ハードディスクで SWF ファイルを開くには、ファイルをローカルでテストできるよう、特殊な設定が必要になる場合があります。

ローカルファイルのセキュリティについては、611 ページの「ローカルファイルセキュリティ と Flash Player について」を参照してください。Flash Player 7 と Flash Player 8 以降のセキュリティモデルの違いについては、610 ページの「Flash Player の旧セキュリティモデルとの互換性について」を参照してください。サーバーからデータをロードし、解析する方法については、569 ページ、第 15 章の「外部データの操作」. を参照してください。セキュリティの詳細については、www.adobe.com/devnet/security および www.adobe.com/software/flashplayer/security/ を参照してください。

Flash 8 以降のセキュリティの詳細については、次のトピックを参照してください。

Flash Player の旧セキュリティモデルとの互換性について 610
ローカルファイルセキュリティ と Flash Player について 611
ネットワーク API の制限. 627
サーバーサイドポリシーファイルによるデータへのアクセス許可 636
SWF ファイル間の HTTP プロトコルから HTTPS プロトコルへのアクセス 641

Flash Player の旧セキュリティモデルとの互換性について

Flash Player 7 でのセキュリティ機能の変更の結果として、Flash Player 6 以前で適切に実行されるコンテンツが Flash Player の新しいバージョンでは適切に実行されない可能性があります。たとえば、Flash Player 6 の場合、www.adobe.com にある SWF ファイルは、data.adobe.com にあるサーバー上のデータにアクセスできます。このように Flash Player 6 では、あるドメインの SWF ファイルが、" 類似する " 別のドメインからデータをロードすることが許されています。

Flash Player 7 以降では、バージョン 6 以前の SWF ファイルが別のドメイン上に存在するサーバーからデータをロードしようとして、そのサーバーがその SWF ファイルのドメインからのアクセスを許可するポリシーファイルを提供しない場合、Flash Player の [設定] ダイアログボックスが表示されます。このダイアログボックスは、クロスドメインデータアクセスを許可するか拒否するかをユーザーに尋ねます。

ユーザーが [許可] をクリックすると、SWF ファイルは要求されたデータにアクセスできます。ユーザーが [拒否] をクリックすると、SWF ファイルは要求されたデータにアクセスできません。

このダイアログボックスを表示しないようにするには、データを提供するサーバー上にセキュリティポリシーファイルを作成します。詳細については、637 ページの「ドメイン間のデータロード許可」を参照してください。

Flash Player 7 以降では、セキュリティポリシーファイルがないと、クロスドメインにアクセスできません。

Flash Player 8 以降では、`System.security.allowDomain` の取り扱い方法が変更されました。任意の引数で `System.security.allowDomain` を呼び出す Flash SWF ファイル (Flash 8 以降)、またはワイルドカード (*) 値を使用するその他の SWF ファイルでは、それ自体へのアクセスだけが認められます。たとえば、`System.security.allowDomain("*")` や `System.security.allowInsecureDomain("*")` などのワイルドカード (*) 値がサポートされています。バージョン 7 以前の SWF ファイルが、ワイルドカード (*) 以外の引数で `System.security.allowDomain` または `System.security.allowInsecureDomain` を呼び出す場合、呼び出した SWF ファイルのドメインにあるバージョン 7 以下の全 SWF ファイルが Flash Player 7 の場合と同様に影響を受けることになります。ただし、この種の呼び出しは、呼び出している SWF ファイルのドメインにある Flash Player 8 (以降) の SWF ファイルには影響を与えません。その結果、Flash Player 内での古いコンテンツの中断を最小限に抑えることができます。

詳細については、627 ページの「ネットワーク API の制限」、「allowDomain (security.allowDomain method)」、および「allowInsecureDomain (security.allowInsecureDomain method)」を参照してください。

Flash Player 8 以降では、ユーザーのコンピュータに具体的設定がない場合、ローカル SWF ファイルがインターネットと通信することはできません。これらの制限が適用される前にパブリッシュされた古いコンテンツが残っている場合を想定してみます。その古いコンテンツが、ネットワークまたはローカルファイルシステムの一方もしくは両方と通信しようとすると、その通信は Flash Player によって停止されます。アプリケーションを意図したとおりに機能させるには、ユーザーが明示的に許可を与える必要があります。詳細については、611 ページの「ローカルファイルセキュリティ と Flash Player について」を参照してください。

ローカルファイルセキュリティ と Flash Player について

Flash Player のセキュリティモデルでは、デフォルトで、ローカルコンピュータの Flash アプリケーションと SWF ファイルが、インターネットとローカルファイルシステムの両方とは通信できない設定になっています。ローカル SWF ファイルとは、Web サイトから提供されたものではなく、ユーザーのコンピュータにローカルでインストールされた SWF ファイルであり、プロジェクタ (EXE) ファイルを含みません。

メモ	この項で取り上げる制限は、インターネット上の SWF ファイルには適用されません。

FLA ファイルを作成する場合、SWF ファイルがネットワークまたはローカルファイルシステムのどちらと通信できるようにするかを指定することができます。Flash Player の旧バージョン (7 以前) では、ローカル SWF ファイルが他の SWF ファイルと対話して、リモートまたはローカルの任意のロケーションからデータをロードすることが可能でした。Flash Player 8 以降では、SWF ファイルはローカルファイルシステムとインターネットの両方には接続できません。これは安全のための変更であり、SWF ファイルがハードディスク上のファイルを読み込み、そのコンテンツをインターネットに送信するのを防ぐことができます。

このセキュリティ制限は、古いコンテンツであるか (Flash の旧バージョンで作成された FLA ファイル)、または Flash 8 以降で作成されたコンテンツであるかに関係なく、ローカルに展開されたすべてのコンテンツに適用されます。Flash MX 2004 以前を使用して、ローカルで実行され、インターネットにもアクセスする Flash アプリケーションを展開したとします。Flash Player 8 以降では、このアプリケーションは、インターネットと通信するための許可をユーザーに要求するプロンプトを表示します。

ハードディスクのファイルをテストする場合、そのファイルが信頼できるローカルドキュメントか、潜在的に信頼できないドキュメントかを判定する一連の手順が存在します。そのファイルを Flash オーサリング環境で作成した場合 (たとえば、[制御]-[ムービープレビュー] を選択した場合など)、テスト環境にあることから、そのファイルは信頼性があります。

Flash Player 7 以前では、ローカル SWF ファイルは、ローカルファイルシステムと (インターネットなどの) ネットワークの両方にアクセスすることができました。Flash Player 8 では、ローカル SWF ファイルには、次のレベルの許可を指定できます。

ローカルファイルシステム専用アクセス (デフォルト)　ローカル SWF ファイルはローカルファイルシステムと UNC (Universal Naming Convention : 汎用命名規則) ネットワークパスにアクセスすることができますが、インターネットとの通信はできません。ローカルファイルアクセス SWF ファイルの詳細については、619ページの「ローカルファイルにのみアクセス (デフォルト)」を参照してください。

ネットワーク専用アクセス　ローカル SWF ファイルは、インターネットなどのネットワークにはアクセスできますが、インストールされているローカルファイルシステムにはアクセスできません。ネットワーク専用 SWF ファイルの詳細については、620 ページの「ネットワークにのみアクセス」を参照してください。

ローカルファイルシステムとネットワークへのアクセス　ローカル SWF ファイルはインストールされているローカルファイルシステムからの読み取りと、サーバーからの読み取りおよび書き込み、ネットワークまたはローカルファイルシステムのその他の SWF ファイルへのクロススクリプトを実行することができます。これらのファイルには信頼性があり、Flash Player 7 の場合と同様に動作します。ローカルおよびネットワークアクセス SWF ファイルの詳細については、620 ページの「ファイルシステムとネットワークへのアクセス」を参照してください。

オーサリングツールとの関係における Flash 8 以降でのローカルファイルセキュリティの詳細については、以下の各項を参照してください。

- 613 ページの「ローカルセキュリティサンドボックスについて」
- 614 ページの「Flash Player のセキュリティ設定について」
- 616 ページの「ローカルファイルセキュリティとプロジェクタファイルについて」
- 617 ページの「古い SWF ファイルのトラブルシューティングについて」
- 617 ページの「ローカルコンピュータに展開された古いコンテンツの修正」
- 618 ページの「ローカル展開のためのファイル再パブリッシュ」

ユーザー用のローカルファイルセキュリティについては、614 ページの「Flash Player のセキュリティ設定について」を参照してください。セキュリティの詳細については、www.adobe.com/devnet/security/ および www.adobe.com/software/flashplayer/security/ を参照してください。

ローカルセキュリティサンドボックスについて

Flash Player には数種類のセキュリティサンドボックスがあります。それぞれが、SWF ファイルとローカルファイルシステムまたはネットワークの一方との通信、または両方との同時通信を設定するためのものです。ファイルとローカルファイルシステムあるいはネットワークとの通信方法を制限することで、コンピュータやファイルの安全を確保することができます。セキュリティサンドボックスを理解することは、予想外のエラーに遭遇することなく、コンピュータ上で Flash アプリケーションを開発し、テストする際の助けになります。

ローカル - ファイルシステムにアクセス許可

セキュリティのために、Flash Player では、古いローカル SWF ファイルを含めすべてのローカル SWF ファイルをローカルファイルシステムのサンドボックスに入れるようデフォルト設定されています (異なる設定の場合を除く)。一部の古い SWF ファイル (Flash Player 7 以前) については、アクセスに制限を加えることで (外部ネットワークのアクセス禁止)、動作に影響を与えることができますが、これはユーザー保護に関して最も安全なデフォルト設定になります。

このサンドボックスから、SWF ファイルはローカルファイルシステムや UNC ネットワークパスに (たとえば、`XML.load()` メソッドを使用して) ファイルを読み取ることができますが、ネットワークとの通信はできません。その結果、ローカルデータがネットワークに漏れ出したり、不正に共有されたりすることがなくなります。

ローカル - ネットワークにアクセス許可

ローカル SWF ファイルがネットワークにアクセス許可されたローカルファイルのサンドボックスに割り当てられると、ローカルファイルシステムへのアクセス権が失われます。代わりに、SWF ファイルはネットワークにアクセスできるようになります。ただし、ネットワークにアクセス許可されたローカル SWF ファイルは、そのための許可を得ない限り、ネットワークで派生したデータを読み取ることはできません。したがって、ネットワークにアクセス許可されたローカル SWF ファイルにはローカルアクセス権はないものの、ネットワーク経由でデータを送信したり、サイト固有のアクセス許可を与えられたサイトからネットワークデータを読み取ったりすることができます。

ローカル - 信頼する

信頼性のあるローカルファイルのサンドボックスに割り当てられた SWF ファイルは他の SWF ファイルと自由に通信したり、どこからでも (リモートでもローカルでも) データをロードしたりすることができます。

Flash Player のセキュリティ設定について

Flash Player は、ユーザーがアクセスを明示的に認めたり拒否したりする必要がほとんどないセキュリティ設定を採用しています。時に、Flash Player 7 以前の古いセキュリティ規則に従って作成された古い Flash コンテンツに遭遇する場合があります。そのような場合、Flash Player では、古いセキュリティ規則を使って開発者の意図するとおりにコンテンツが動作するのを認めるか、新しい、より厳格な規則を適用するかを選べるようになっています。後者を選択した場合、表示したり、再生したりできるのは最も新しいセキュリティ標準に準拠したコンテンツだけとなり、古い Flash コンテンツが適切に動作しなくなることがあります。

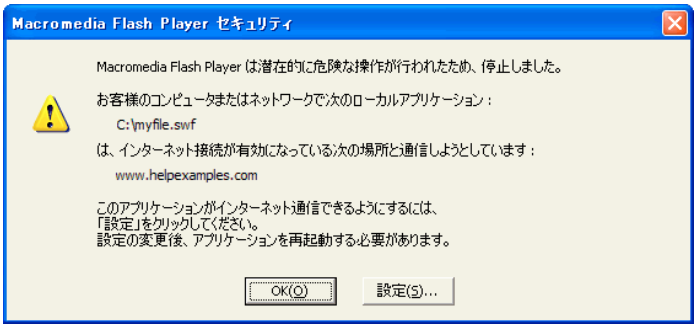
SWF ファイルを表示するユーザーは全員 (Flash 開発者以外を含む)、Flash Player の [設定マネージャ] (次の図参照) の [グローバルセキュリティ設定] パネルでグローバルに許可を設定できます。

古いコンテンツを Flash Player の新しいバージョンで実行する場合、ユーザーは新しい規則を適用するかどうかを決定しなければならず、次のポップアップダイアログボックスのいずれかが表示されます。古い Flash コンテンツとインターネット上の他の場所との通信を許可するには、これらのダイアログボックスで設定を行う必要があります。

- ダイアログボックスが表示されて、使用中の Flash コンテンツが古いセキュリティ規則を使ってドメイン外のサイトの情報にアクセスしようとして、情報が 2 つのサイトで共有されている可能性があるという警告を受けることがあります。Flash Player では、そうしたアクセスを許可するかどうかをユーザーが設定できます。

  ダイアログボックスに応答する以外に、[グローバルセキュリティ設定] パネルを使用して、他のサイトやドメインとのアクセスに関して、Flash Player にダイアログボックスを通じてユーザーの許可を事前に求めさせるか、許可を求めることなく常にアクセスを拒否させるか、あるいは許可を求めることなく常にアクセスを許可させるかを設定することができます。

■ (Flash Player 8 以降) SWF ファイルがインターネットと通信を行おうとしているという警告を発するダイアログボックスが表示されることがあります。Flash Player では、デフォルト設定で、ローカル Flash コンテンツがインターネットと通信できないようになっています。



[設定] をクリックして [グローバルセキュリティ設定] パネルにアクセスし、インターネットとの通信を許可するコンピュータ上の Flash アプリケーションを指定することができます。

セキュリティ設定を変更したり、オプションについての情報を得るには、[グローバルセキュリティ設定] パネルを使用します。このパネルを使用して、Adobe Flash Player のプライバシー設定をリセットします。

■ [常に拒否] を選択して、選択内容を確認すると、ユーザーのカメラやマイクの使用を必要とする Web サイトのアクセスはすべて拒否されます。Web サイトにカメラやマイクを使わせるかどうかを再度確認されることはありません。このアクションは、既にアクセスしたことのある Web サイトにも、アクセスしたことのない Web サイトにも適用されます。

■ [常に確認] を選択して、選択内容を確認すると、ユーザーのカメラやマイクを使おうとする Web サイトのアクセスについてはすべて許可が必要となります。このアクションは、既にアクセスしたことのある Web サイトにも、アクセスしたことのない Web サイトにも適用されます。

以前に、Web サイトのアクセスを永続的に許可ないし拒否するために、[プライバシー設定] パネルで [後で確認] を選択してあった場合 (次の図参照)、[常に確認] または [常に拒否] を選択すると、すべての Web サイトに関して [後で確認] は解除されることになります。つまり、このパネルで選択した内容は、[プライバシー設定] パネルで設定した内容を上書きすることになります。次の図を参照してください。

[常に確認] または [常に拒否] を選択した後 (または実際に確認ないし拒否を行った後)、既にアクセスした個々の Web サイトについてプライバシー設定を行うことができます。たとえば、[常に拒否] を選択し、次に [Web サイトプライバシー設定] パネルを使用して、信頼できる Web サイトごとに [常に許可] を選択することができます。

ローカルで展開されたコンテンツおよびローカルデータについて、ユーザーは別のオプションを使用できます。ユーザーは [グローバルセキュリティ設定] パネルを使用して、インターネットにアクセスできる SWF ファイルを指定できます。[グローバルセキュリティ設定] パネルでの設定の詳細については、621 ページの「設定マネージャを使用しての信頼性のあるファイルの指定」を参照してください。[グローバルセキュリティ設定] パネルの詳細については、www.adobe.com/support/documentation/jp/flashplayer/help/settings_manager04a.html を参照してください。

メモ	ユーザーが [グローバルセキュリティ設定] パネルで行った選択は、ポップアップダイアログボックスで行ったすべての決定をオーバーライドします。

ローカルファイルセキュリティとプロジェクタファイルについて

SWF ファイルを使用するのにエンドユーザーは実行可能ファイルを実行しなければならないため、プロジェクタファイルと、それに含まれる SWF ファイル、実行時にプロジェクタにロードされる SWF ファイルは、ローカルファイルセキュリティ制限の影響を受けません。Flash Player 8 以降のセキュリティファイルやプロジェクタファイルに変更はありません。アクセスやとセキュリティのレベルは、Flash Player の旧バージョンと同じです。

ユーザーはプロジェクタファイルを実行するときは慎重になります。プロジェクタファイルとは、実行可能な EXE または Macintosh アプリケーションであり、ユーザーは自らのコンピュータでそうしたファイルを実行する場合には慎重に行う必要があります。プロジェクタファイルを使用してアプリケーションを配布する場合、ユーザーによってはそのアプリケーションをインストールしないことがあります。

さらに、プロジェクタファイルは特定バージョンの Flash Player をプロジェクタの内側に埋め込みます。その場合のプロジェクタは、アドビ システムズ社の Web サイトからダウンロード可能な Flash Player の最新バージョンより古い可能性があります。プロジェクタが古いバージョンの Flash で作成されたものである場合や、Flash Player のエディションが Flash オーサリングツールの現行バージョンより後にリリースされたものである場合、プロジェクタファイル内に埋め込まれた Flash Player が古いバージョンである可能性があります。このような理由から、可能な限りアプリケーションの配布には SWF ファイルを使用してください。

古い SWF ファイルのトラブルシューティングについて

(Flash MX 2004 以前で作成した) 古い FLA ファイルや SWF ファイルの中には、ローカルの (ハードディスク上での) テストや展開を実行した場合、Flash 8 以降のセキュリティ変更のために、適切に動作しないものがあります。SWF ファイルがドメイン外のサイトにアクセスしようとした場合に発生します。また、そのような場合には、クロスドメインポリシーファイルを実装する必要があります。

Flash MX 2004 以前で作成された FLA ファイルや SWF ファイルが、Flash オーサリングツールを所有しておらず Flash Player 8 以降にアップグレードしたユーザーに配布された可能性もあります。ローカルで (SWF ファイルをユーザーのハードディスクに置いて) テストないし展開した古いコンテンツが、Flash Player 8 以降による再生中にインターネットと通信を行うため中断された場合、適切に (ダイアログボックスのボタンをクリックすることで) 再生するためには、コンテンツを明示的に信頼するかどうかをユーザーに任せなければなりません。

ローカルコンピュータでの再生用に古いコンテンツを修正する方法については、617 ページの「ローカルコンピュータに展開された古いコンテンツの修正」を参照してください。

ローカルコンピュータに展開された古いコンテンツの修正

ローカルコンピュータに展開される Flash Player 7 以前用に SWF ファイルをパブリッシュし、それがインターネットと通信する場合、ユーザーはインターネット接続を明示的に許可する必要があります。ユーザーは、ローカルコンピュータ上の SWF ファイルの場所を設定マネージャの、信頼性のあるサンドボックスに追加することで、コンテンツの中断を防止することができます。

ローカル再生のために SWF ファイルを修正するには、次のオプションのいずれかを使用します。

再展開　Local Content Updater を実行します。Local Content Updater は、SWF ファイルを再設定し、セキュリティモデル (Flash Player 8 以降) との互換性を持たせます。ローカル SWF ファイルを再設定すると、アクセスできるのはネットワークかローカルファイルシステムのどちらか一方になります。Local Content Updater 詳細およびダウンロードについては、www.adobe.com/support/flashplayer/downloads.html を参照してください。

再パブリッシュと再展開　Flash によってファイルを再パブリッシュします。オーサリングツールにより、Flash 8 では、ローカル SWF ファイルによるネットワークまたはローカルファイルシステムのどちらか一方へのアクセスを許可するかどうかを、[パブリッシュ設定] ダイアログボックスで指定することを求められます。ローカル SWF ファイルによるネットワークアクセスを認めた場合は、さらにローカル SWF ファイルがアクセスする SWF、HTML、データ、およびサーバーファイル内で、その SWF ファイル (およびすべてのローカル SWF ファイル) の許可を有効にする必要があります。詳細については、618ページの「ローカル展開のためのファイル再パブリッシュ」を参照してください。

新規コンテンツの展開　"#Security/FlashPlayerTrust" フォルダの設定 (.cfg) ファイルを使用します。このファイルを使用して、ネットワークアクセスおよびローカルアクセスの許可を設定できます。詳細については、623 ページの「Flash 開発のための設定ファイルの作成」を参照してください。

メモ	これらのオプションはいずれも、SWF ファイルの再パブリッシュまたは再展開を必要とします。

ローカル展開のためのファイル再パブリッシュ

テストないし承認のために Flash FLA または SWF ファイルをユーザーに送り、インターネットにアクセスするようアプリケーションに求めることができます。ローカルシステムで再生されたドキュメントが (たとえば、XML のロードや変数の送信で) インターネット上のファイルにアクセスした場合、コンテンツが適切に機能するためには設定ファイルが必要になったり、パブリッシュした SWF ファイルがネットワークにアクセスできるよう FLA ファイルを設定する必要が生じたりします。または、FlashPlayerTrust ディレクトリ内に設定ファイルを設定することもできます。設定ファイルの設定の詳細については、623 ページの「Flash 開発のための設定ファイルの作成」を参照してください。

Flash を使用して、Flash Player ローカルファイルセキュリティと連携して動作するコンテンツを作成し、ローカルで展開します。Flash 8 以降の [パブリッシュ設定] で、ローカルコンテンツがネットワークまたはローカルファイルシステムのどちらにアクセスできるかを指定する必要があります。両方にアクセスできるように指定することはできません。

[パブリッシュ設定] ダイアログボックスで、FLA ファイルの許可レベルを設定できます。ここで設定した許可レベルは、ハードディスク上での FLA ファイルのローカル再生に影響を与えます。

メモ	ローカルファイルに対してネットワークアクセスを指定する場合は、さらにローカル SWF ファイルがアクセスする SWF、HTML、データ、およびサーバーファイル内で許可を有効にする必要があります。

ネットワーク SWF ファイル　オンラインサーバーなど、ネットワークからダウンロードした SWF ファイルは、一意の Web サイト基準ドメインに対応する個別のサンドボックスに配置されます。ネットワークアクセスがあることを指定するローカル SWF ファイルは、ネットワークにアクセス許可されたファイルのサンドボックスに配置されます。デフォルトにより、これらのファイルは、そのファイル自体と同じサイトにあるデータだけを読み取ることができます。これらのファイルには、ドメイン完全一致規則が適用されます。適切な許可があれば、ネットワーク SWF ファイルは他のドメインからのデータにアクセスすることができます。ネットワーク SWC ファイルの詳細については、620 ページの「ネットワークにのみアクセス」を参照してください。

ローカル SWF ファイル　ローカルファイルシステムまたは UNC ネットワークパスと連動する SWF ファイルは Flash Player 8 以降の 3 種類のサンドボックスのいずれか 1 つに配置されます。デフォルトでは、ローカルファイルシステムにアクセス許可されたサンドボックスに配置されます。設定ファイルを使用して、信頼できるとして登録されたローカル SWF ファイルは、信頼性のあるローカルファイルのサンドボックスに配置されます。3 種類のサンドボックスの詳細については、619 ページの「ローカルファイルにのみアクセス (デフォルト)」を参照してください。

セキュリティサンドボックスの詳細については、613 ページの「ローカルセキュリティサンドボックスについて」を参照してください。

最初の 2 つの許可レベルは Flash オーサリング環境で設定され、3 番目は [グローバルセキュリティ設定] パネルまたは "FlashAuthor.cfg" ファイルを使用して設定されます。次の例は、ローカルハードディスクでのテストのためにファイルをパブリッシュする際に使用可能なオプションを示しています。

特定の許可レベルでドキュメントをパブリッシュするには :

1. 許可レベルを設定する FLA ファイルを開きます。
2. [ファイル]-[パブリッシュ設定]-[Flash] を選択します。
3. [ローカルでの再生に関するセキュリティ] ダイアログボックスを開き、ポップアップメニューから次のいずれかのオプションを選択します。
   - ローカルファイルにのみアクセスします。「ローカルファイルにのみアクセス (デフォルト)」を参照してください。
   - ネットワークにのみアクセスします。「ネットワークにのみアクセス」を参照してください。
4. [OK] をクリックして FLA ファイルのオーサリングを続けるか、[パブリッシュ] をクリックして SWF ファイルを作成します。

アプリケーションに対して設定可能な許可レベルの詳細については、619 ページの「ローカルファイルにのみアクセス (デフォルト)」、620 ページの「ネットワークにのみアクセス」および 620 ページの「ファイルシステムとネットワークへのアクセス」を参照してください。

ローカルファイルにのみアクセス (デフォルト)

この許可レベルを設定するには、[パブリッシュ設定]-[Flash] を選択し、次に [ローカルでの再生に関するセキュリティ] ポップアップメニューから [ローカルファイルにのみアクセスする] を選択します。この許可レベルでは、ローカル SWF ファイルは、そのファイルが実行されるローカルファイルシステムだけにアクセスできます。SWF ファイルは、ローカルディスク上の既知のファイルを無制限に読み取ることができます。ただし、ネットワークにアクセスしているアプリケーションには次の制限が適用されます。

- SWF ファイルはネットワークには一切アクセスできません。SWF ファイルはネットワーク SWF をクロススクリプトすることも、ネットワーク SWF ファイルによってクロススクリプトされることもできません。
- SWF ファイルはネットワークだけを対象としたアクセス許可を受けているローカル SWF ファイルとは通信できません。また、HTML ページとも通信できません。ただし、その HTML ページが信頼でき、`allowScriptAccess` が `always` に設定されている場合や、`allowScriptAccess` が設定されておらず、SWF ファイルが Flash Player 7 以前の場合など、通信が認められるケースもあります。

ネットワークにのみアクセス

この許可レベルを設定するには、[パブリッシュ設定]-[Flash] を選択し、次に [ローカルでの再生に関するセキュリティ] ポップアップメニューから [ネットワークにのみアクセスする] を選択します。ネットワークにアクセスできるローカル SWF ファイルは、`<allow-access-from-domain= "*">` に設定されたクロスドメインポリシーファイルがサーバーに含まれる場合に、サーバーから読み取ることができます。ネットワークにアクセスできるローカル SWF ファイルは、他の SWF ファイル (アクセス先の SWF ファイル) が `System.security.allowDomain("*")` を含んでいる場合に、他の SWF ファイルとクロススクリプトすることができます。ネットワークにアクセスできるローカル SWF ファイルは、このローカル SWF ファイルが `allowDomain("*")` を含んでいる場合に、ネットワーク SWF ファイルからクロススクリプトすることができます。SWF ファイルはローカルファイルを読み取ることはできません。場合によっては、SWF ファイルのタイプがアクセスに影響を与えることがあります。詳細については、『ActionScript 2.0 リファレンスガイド』の「allowDomain (`security.allowDomain method`)」を参照してください。

ワイルドカード (*) 値は、ローカルホストを含む *すべて*のドメインにアクセスできます。ワイルドカード引数を使用する前に、それだけの広範なレベルのアクセスが必要かどうかを確認してください。

上で説明したどのアクセス許可もない場合、ネットワークにアクセスできるローカル SWF ファイルは、ネットワークにアクセスできる他のローカル SWF ファイルとのみ通信可能であり、たとえば `XML.send()` を使用してサーバーにデータを送信できます。HTML ファイルが信頼できる場合、アクセスが許可されることがあります。

ファイルシステムとネットワークへのアクセス

これは最も高いレベルの許可です。このアクセス許可を持つローカル SWF ファイルは、信頼性のあるローカル SWF ファイルです。信頼性のあるローカル SWF ファイルは他の SWF ファイルを読み取ることや、任意のサーバーと通信することができ、明示的に禁止されていない他の SWF ファイルや HTML ファイル (`allowScriptAccess="none"` となっているファイルなど) に ActionScript を書き込むことができます。このレベルの許可は、ユーザーや Flash 開発者が次の方法で与えることができます。

- 設定マネージャの [グローバルセキュリティ設定] パネルの使用
- グローバル設定ファイルの使用

設定ファイルは、SWF ファイルでインストールすることや、Flash 開発者が作成することや、すべてのユーザーまたは現在のユーザー用に管理者が追加することや、現在のユーザー用に Flash 開発者が追加することができます。

設定ファイルおよび [グローバルセキュリティ設定] パネルの詳細については、614 ページの「Flash Player のセキュリティ設定について」、621 ページの「設定マネージャを使用しての信頼性のあるファイルの指定」、および 623 ページの「Flash 開発のための設定ファイルの作成」を参照してください。

Flash 8 ローカルファイルセキュリティ制限を使用したローカルでのコンテンツのテスト

Flash 開発者は、ローカルで頻繁に Flash アプリケーションをテストするため、ローカル Flash アプリケーションがインターネットと通信しようとした場合にダイアログボックスプロンプトを目にすることがあります。また、SWF ファイルがネットワークにアクセスできない場合に Flash Player で SWF ファイルをテストすると、このダイアログボックスが表示されることがあります。指定された許可レベルでの SWF ファイルのパブリッシュの詳細については、618 ページの「ローカル展開のためのファイル再パブリッシュ」を参照してください。これらのオプションのいずれかを使用して SWF ファイルをパブリッシュするということは、ネットワークまたはローカルファイルシステムのどちらかと通信が可能であることを意味します。

ドキュメントのテストを行うときは、ローカルファイルシステムとネットワークの両方と通信することが必要になる場合があります。新しいセキュリティモデルでは、Flash アプリケーションのオーサリング中に作業が中断されることがあるため、Flash Player の設定マネージャで [グローバルセキュリティ設定] パネルを使用して、コンピュータ上のどの Flash アプリケーションに、常にインターネットとローカルファイルシステム両方との通信を許可するかを設定できます。また、設定ファイルを修正して、信頼性のあるディレクトリをハードディスクに指定することもできます。

詳細については、次の項を参照してください。

- 621 ページの「設定マネージャを使用しての信頼性のあるファイルの指定」
- 623 ページの「Flash 開発のための設定ファイルの作成」

設定マネージャを使用しての信頼性のあるファイルの指定

コンピュータにある Flash コンテンツに対して、そのコンテンツの場所を Flash Player 設定マネージャの [グローバルセキュリティ設定] パネルに追加することで、常に古いセキュリティ規則の使用を認めるコンテンツを指定できます。コンピュータ内の場所を [グローバルセキュリティ設定] パネルに追加すると、その場所のコンテンツは " 信頼性のある " コンテンツになります。Flash Player がユーザーに許可を求めてくることはありません。Flash Player は、[グローバルセキュリティ設定] パネルで [常に拒否] が指定されている場合でも、古いセキュリティ規則を使用することができます。[これらのファイルとフォルダを常に信頼する] リストは [設定] パネルのオプションをオーバーライドします。たとえば、すべてのローカルコンテンツおよび Web コンテンツで古いセキュリティ規則の使用を許可しないよう設定した場合でも、[これらのファイルとフォルダを常に信頼する] リストにあるローカルファイルだけは古いセキュリティ規則を使用できます。

パネルの一番下にある、常に信頼できるファイルの一覧は、Web サイトにアクセスしているときに使用するコンテンツではなく、ユーザーが自分のコンピュータにダウンロードした Flash コンテンツに対して適用されます。

次の例は、ローカル SWF ファイルとインターネットの通信の許可をどのように指定するかを示したものです。ブラウザでファイルをローカルにテストする場合 ([ファイル]-[パブリッシュプレビュー]-[HTML])、セキュリティダイアログボックスが表示されます。[設定] をクリックすると、設定マネージャの [グローバルセキュリティ設定] パネルが表示されます。

SWF ファイルがインターネットおよびローカルファイルシステムとの通信をできるよう設定するには :

1. [グローバルセキュリティ設定] パネルで、ポップアップメニューをクリックし、[場所の追加] を選択します。

   [場所の追加] ボックスが表示されます。

   ダイアログボックスの [設定] ボタンをクリックして設定マネージャを開いた場合、[場所の追加] ボックスには、C:¥‹ ディレクトリ名 ›¥‹ ファイル名 .swf› または /‹ ユーザー ›/‹ ディレクトリ名 ›/‹ ファイル名 .swf› のようなパスが表示されます。このパスは、インターネットとの通信を試みて、Flash Player のセキュリティ機能によって中断させられたファイルを表しています。このパスに、インターネットと通信させたいコンテンツが含まれている場合は、そのパスをコピーして [信頼する場所] ボックスにペーストします。または、参照ボタンのいずれかをクリックして、コンテンツを探します。

個々のファイルまたはディレクトリ全体を追加できます。ディレクトリ全体を追加すると、その中にあるすべてのファイルとサブディレクトリが信頼できるファイルおよび信頼できるサブディレクトリとなります。Flash コンテンツの中には複数の関連ファイルで構成されているものがあり、その関連ファイルがすべて含まれるディレクトリは、全体を信頼することが必要になります。一般に、トップレベルのディレクトリについては信頼できるディレクトリにするのは避けてください。

2. [確認] をクリックします。

   その場所が、[セキュリティ設定] パネルに表示されます。一覧表示されている場所は、[セキュリティ] パネルの [常に否定] または [常に確認] オプションが選択されている場合でも、常にインターネットとの通信ができます。

   信頼できる場所を追加した後で、ブラウザを更新するかプレーヤーを再起動することによって、ローカル Flash コンテンツを再起動する必要があります。

[常に許可] を選択した場合、古いコンテンツ (Flash Player 7 以前) はすべて許可されるようになりますが、それだけでは Flash Player 8 以降のコンテンツも許可されるとは限りません。インターネットとローカルファイルシステムの両方と通信できる Flash アプリケーションとディレクトリをコンピュータ上で指定することをお勧めします。

Flash 開発のための設定ファイルの作成

Flash オーサリングツールでは、ハードディスク上にフラグを設定してユーザーを開発者として識別し、ユーザー向けの [グローバルセキュリティ設定] パネルではなく、同パネルの開発者用バージョンを表示します。このフラグはハードディスク上の FlashAuthor.cfg ファイル内にあり、Flash オーサリングツールのインストール時に自動的にインストールされます。

"FlashAuthor.cfg" ファイルは次のようなディレクトリ中にあります。

Windows　‹ ブートドライブ ›¥Documents and Settings¥‹ ユーザー名 ›¥Application Data¥Adobe¥Flash Player¥#Security

Macintosh　/ ユーザ /‹ ユーザー名 ›/Library/Preferences/Adobe/Flash Player/#Security/

デフォルトにより、このファイルは `LocalSecurityPrompt=Author` に設定されます。つまり、コンピュータに表示される警告では、オーサリングツールがインストールされていないユーザーとしてではなく、Flash 開発者として取り扱われることになります。

ローカルアプリケーションについては、エンドユーザーとしてテストすることができ、エンドユーザーに対する警告ダイアログボックスが表示されます。そのためには、テキストエディタで "FlashAuthor.cfg" を開き、"FlashAuthor.cfg" ファイルの LocalSecurityPrompt を次のように変更します。

```
LocalSecurityPrompt=User
```

LocalSecurityPrompt を Author に設定した状態で、"FlashAuthor.cfg" ファイルを、設計プロセスないし開発プロセスに関係する開発者や、Flash オーサリングツールをインストールしていない状態でローカルハードディスク上の Flash アプリケーションをテストするユーザーに提供できます。こうすることで、ローカルに展開したコンテンツに関するエンドユーザーの経験を模倣することができます。

メモ	"FlashAuthor.cfg" ファイルが削除された場合、ユーザーが Flash オーサリングツールを起動すると再作成されます。

ハードディスクにある #Security ディレクトリに、ユニークな設定ファイルを保存する FlashPlayerTrust ディレクトリを作成できます。これらのファイル内で、ハードディスク上で信頼するディレクトリやアプリケーションを指定することもできます。このディレクトリには管理者権限が不要なため、管理許可のないユーザーでも SWF ファイルの許可を設定し、アプリケーションをテストすることができます。

ディレクトリを指定しないと、コンテンツが所定の機能を果たさないことがあります。FlashPlayerTrust ディレクトリ内の設定ファイルにはディレクトリパスが含まれています。ファイルにはディレクトリの一覧を含めることができ、また新しいパスを追加することができます。Flash Player では、設定ファイルの 1 行につき 1 つのパスを想定しています。その前にスペースがなく、# で始まる行はすべてコメントとして取り扱われます。

ディレクトリに信頼性を付与するための設定ファイルを作成するには :

1. ハードディスクの "#Security" フォルダを確認します。
2. "#Security" フォルダ内に FlashPlayerTrust というフォルダを作成します。
3. テキストエディタを使用して、FlashPlayerTrust ディレクトリに新しいファイルを作成し、myTrustFiles.cfg として保存します。

   設定ファイルには一意の名前を割り当てることができます。
4. Flash アプリケーションをテストするディレクトリを確認します。
5. 各ディレクトリパス (ハードディスクの任意のディレクトリパス) を入力し、ファイルの新しい行にペーストします。異なる行に対して、複数のディレクトリパスをペーストすることができます。最終的に、パスは次のようになります。

   ```
   C:¥Documents and Settings¥< ユーザー名 >¥My Documents¥files¥
   C:¥Documents and Settings¥< ユーザー名 >¥My Documents¥testapps¥
   ```
6. "myTrustFiles.cfg" に変更内容を保存します。
7. ファイルに追加したディレクトリからローカルファイルやネットワークファイルにアクセスするドキュメントをテストします。

   このディレクトリに保存した Flash アプリケーションがローカルファイルやネットワークにアクセスできるようになります。

各設定ファイルには多くのディレクトリパスが保存され、FlashPlayerTrust ディレクトリには多くの *.cfg ファイルが保存される場合があります。

エンドユーザーのハードディスクにインストールされるアプリケーションを作成した場合、そのアプリケーションに対する信頼できるディレクトリを指定するために、FlashPlayerTrust ディレクトリに設定ファイルを作成する必要があります。FlashPlayerTrust ディレクトリに、信頼できるアプリケーションの場所を指定する設定ファイルを作成できます。この FlashPlayerTrust ディレクトリおよび設定ファイルの作成については、前の手順を参照してください。

メモ	インストーラは管理許可のあるユーザーが実行します。

このディレクトリにファイルをインストールする他のアプリケーションとのコンフリクトを回避するため、重複することのない命名スキームを設定する必要があります。たとえば、会社名やソフトウェア名をファイル名に組み込んで、コンフリクトを回避することができます。

ヒント	設定ファイルを使用しない場合は、クライアントまたは他の開発者にローカルハードディスク上で実行する SWF ファイルを提供する代わりに、Flash アプリケーションを独立したテストサーバーに対してパブリッシュすることができます。

設定ファイルの詳細については、www.adobe.com/go/flashauthorcfg_jp を参照してください。設定ファイルを 1 つ作成して、1 つないし複数のディレクトリを信頼性のあるディレクトリとすることも可能です。セキュリティの詳細については、www.adobe.com/devnet/security/ および www.adobe.com/software/flashplayer/security/ を参照してください。

sandboxType プロパティについて

Flash Player 8 以降の `System.security.sandboxType` プロパティは、呼び出し元の SWF ファイルが実行されるセキュリティサンドボックスのタイプを返します。

`sandboxType` プロパティの値は、次の 4 つのうちのいずれかになります。

remote　SWF ファイルはインターネットにホストされ、ドメインベースのサンドボックス規則に従って実行されます。

localTrusted　SWF ファイルは、ユーザーがグローバルセキュリティ設定マネージャか FlashPlayerTrust 設定ファイルを使用して信頼性を設定したローカルファイルです。この SWF ファイルはローカルデータソースからの読み取りと、インターネットなどのネットワークとの通信の両方を実行できます。

localWithFile　SWF ファイルはローカルファイルであり、信頼性はなく、ネットワークを指定してパブリッシュされているわけではありません。この SWF ファイルはローカルデータソースからの読み取りを実行できますが、インターネットなどのネットワークとの通信はできません。

localWithNetwork　SWF ファイルはユーザーによって信頼されておらず、また [パブリッシュ設定] ダイアログボックス ([Flash] タグ) で選択した [ネットワークにのみアクセスする] によるパブリッシュも行われていないローカルファイルです。この SWF ファイルはネットワークとの通信はできますが、ローカルデータソースからの読み取りはできません。

sandboxType プロパティはあらゆる SWF ファイルについてチェックすることができますが、値が返ってくるのは、Flash Player 8 以降用にパブリッシュされたファイルだけです。つまり、7 以前の Flash Player に関しては、ファイルのパブリッシュ実行時点で、sandboxType プロパティがサポートされているかどうかを知ることはできません。プロパティがサポートされていない場合、値は undefined になります。これは、(System.capabilities.version プロパティに表示される) Flash Player のバージョンが 8 より古い場合に発生します。値が undefined の場合、SWF ファイルの URL がローカルかローカルでないかによって、サンドボックスタイプを決定できます。SWF ファイルがローカルファイルの場合、localTrusted として分類されます (これは、8 より古い Flash Player でのローカルコンテンツの処理方法です)。SWF ファイルがローカルファイルでない場合、remote に分類されます。

ローカルファイルシステムの制限について

ローカルファイルシステムのファイルは、設定マネージャの [グローバルセキュリティ設定] パネルにある FlashPlayerTrust ディレクトリ内の設定ファイルを使用して登録されていないか、Flash オーサリング環境の [パブリッシュ設定] ダイアログボックスでネットワーク許可を受けていません。

メモ

セキュリティサンドボックスについては、613 ページの「ローカルセキュリティサンドボックスについて」を参照してください。

これらのファイルには、Flash Player 8 以降で再生される古いコンテンツが含まれています。Flash 9 で開発しているコンテンツや、次のカテゴリのいずれかに属するコンテンツである場合、そのファイルは信頼できるファイルとして登録する必要があります。信頼できるファイルとしての保存の詳細については、621 ページの「設定マネージャを使用しての信頼性のあるファイルの指定」を参照してください。設定ファイルを使用したローカルファイル再生の許可付与については、623 ページの「Flash 開発のための設定ファイルの作成」を参照してください。

ローカルファイルシステムの SWF ファイルには次の制限があります。

- 次を含め、ネットワークにアクセスできません。
  - ネットワークからの他の SWF ファイルのロード (インターネット以外の UNC パスの使用を除く)
  - HTTP 要求の送信
  - XMLSocket、Flash Remoting、または NetConnection を使用した接続
  - getURL() の呼び出し (getURL("file:...") または getURL("mailto:...") を使用する場合を除く)
- ローカルファイルシステムのファイルにはアクセスできますが、次に対しては制限があります。
  - クロススクリプト (他の SWF ファイルにあるオブジェクトへの ActionScript アクセスなど)
  - System.security.allowDomain の呼び出し

- LocalConnection.allowDomain ハンドラとは無関係な LocalConnection の送信側または受信側として使用

メモ	ローカルファイルシステム SWF ファイルは他のローカルファイルシステムの非ネットワーク SWF ファイルと通信できます。ただし、ネットワークにアクセス許可されたローカル SWF ファイルとは通信できません。

ローカルファイルシステム SWF ファイルは、ローカルファイルシステムの既知のファイルを読み取りアクセスします。たとえば、インターネットではなくローカルファイルシステムからロードした場合は、ローカルファイルシステム SWF ファイルで XML.load() を使用できます。

- ローカルファイルシステムの SWF ファイルは HTML ページと通信できません。それには次の制限が含まれます。
  - 受信スクリプト (外部 API、ActiveX、LiveConnect、XPConnect など)
  - 発信スクリプト (fscommand 呼び出し、getURL("javascript:...") など)

メモ	HTML ページが信頼できる場合の例外。

ネットワーク API の制限

SWF コンテンツを含む HTML ページで <object> および <embed> タグに allowNetworking パラメータを設定することによって、ネットワーク機能に対する SWF ファイルのアクセスを制御できます。

allowNetworking の有効な値は次のとおりです。

- "all" (デフォルト) : SWF ではすべてのネットワーク API が許可されます。
- "internal" : SWF ファイルは、以下に示すブラウザナビゲーション API またはブラウザインタラクション API を呼び出すことはできませんが、他のすべてのネットワーク API を呼び出すことができます。
- "none" : SWF ファイルは、以下に示すいずれのネットワーク API も呼び出すことができません。また、以下に示すいずれの SWF 対 SWF コミュニケーション API も使用できません。

allowNetworking パラメータを設定するには、次の例に示すように、SWF コンテンツを含む HTML ページで <object> および <embed> タグに allowNetworking パラメータを追加し、その値を設定します。

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
 codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
   swflash.cab#version=9,0,18,0"
 width="600"  height="400" id="test" align="middle">
<param name="allowNetworking" value="none" />
<param name="movie" value="test.swf" />
<param name="bgcolor" value="#333333" />
<embed src="test.swf" allowNetworking="none" bgcolor="#333333" width="600"
   height="400"
 name="test" align="middle" type="application/x-shockwave-flash"
 pluginspage="http://www.adobe.com/shockwave/download/
   download.cgi?P1_Prod_Version=ShockwaveFlash" />
</object>
```

allowNetworking が "internal" に設定されている場合は、以下の API を使用できません。

- getURL
- MovieClip.getURL
- fscommand()
- ExternalInterface.call()

allowNetworking が "none" に設定されている場合は、上記の API に加えて以下の API も使用できません。

- XML.load
- XML.send
- XML.sendAndLoad
- LoadVars.load()
- LoadVars.send
- LoadVars.sendAndLoad
- loadVariables
- loadVariablesNum
- MovieClip.loadVariables
- NetConnection.connect
- NetStream.play
- loadMovie
- loadMovieNum
- MovieClip.loadMovie
- MovieClipLoader.loadClip
- Sound.loadSound
- LocalConnection.connect
- LocalConnection.send
- SharedObject.getLocal
- SharedObject.getRemote
- FileReference.upload
- FileReference.download
- System.security.loadPolicyFile
- XMLSocket.connect

選択された allowNetworking 設定により SWF ファイルがネットワーク API を使用することが許可されている場合でも、この章で説明するように、セキュリティサンドボックスの制限に基づいて他の制限が適用されることがあります。

allowNetworking が "none" に設定されている場合、TextField の htmlText プロパティの <img> タグはネットワークコンテンツを表示しません。

ドメイン、ドメイン間のセキュリティ、および SWF ファイルについて

Flash Player 7 以降の場合、デフォルトでは、あるドメインのサービスを利用している SWF ファイルが別のドメインのサービスを利用している SWF ファイルのデータ、オブジェクト、変数にアクセスすることはできません。また、セキュリティで保護されていないプロトコル (HTTPS 以外のプロトコル) 経由でロードされたコンテンツは、セキュリティで保護されているプロトコル (HTTPS プロトコル) 経由でロードされたコンテンツにアクセスできません。これは、両方のコンテンツがまったく同じドメインにある場合でも同じです。たとえば、http://www.adobe.com/main.swf にある SWF ファイルは、明示的な許可がないと、https://www.adobe.com/data.txt からデータをロードできません。また、あるドメインのサービスを受けている SWF ファイルは、`loadVars()` などで他のドメインからデータをロードすることはできません。

同じ数値の IP アドレス間には互換性があります。一方、ドメイン名が同じ IP アドレスに解決される場合であっても、ドメイン名と IP アドレスの間には互換性がありません。

互換性のあるドメインの例を次の表にまとめます。

www.adobe.com	www.adobe.com
data.adobe.com	data.adobe.com
65.57.83.12	65.57.83.12

互換性のないドメインの例を次の表にまとめます。

www.adobe.com	data.adobe.com
adobe.com	www.adobe.com
www.adobe.com	adobe.com
65.57.83.12	www.adobe.com (このドメインが 65.57.83.12 に解決される場合を含む)
www.adobe.com	65.57.83.12 (www.adobe.com がこの IP アドレスに解決される場合を含む)

Flash Player 8 以降では、適切な設定なしに、ローカル SWF ファイルがインターネットと通信することはできません。コンテンツのローカルテスト用に設定ファイルを設定する方法については、623 ページの「Flash 開発のための設定ファイルの作成」を参照してください。

セキュリティの詳細については、www.adobe.com/devnet/security/ および www.adobe.com/software/flashplayer/security/ を参照してください。

詳細については、次のトピックを参照してください。

- 630 ページの「設定およびローカルデータのドメイン名規則」
- 630 ページの「SWF ファイル間のクロスドメインおよびサブドメインのアクセス」
- 637 ページの「ドメイン間のデータロード許可」

設定およびローカルデータのドメイン名規則

Flash Player 6 では、ローカル設定 (カメラやマイクのアクセス許可など) またはローカルに永続化されたデータ (共有オブジェクト) にアクセスする場合に、スーパードメイン一致規則がデフォルトで使用されます。つまり、one.adobe.com、two.adobe.com、および adobe.com でホストされた SWF ファイルの設定やデータは共有され、すべて adobe.com に保存されます。

Flash Player 7 では、ドメイン完全一致規則がデフォルトで使用されます。つまり、one.adobe.com でホストされたファイルの設定やデータは one.adobe.com に、two.adobe.com でホストされたファイルの設定やデータは two.adobe.com に保存されます。`System.exactSettings` で、使用する規則を指定できます。このプロパティは、Flash Player 6 以降用にパブリッシュされたファイルのためにサポートされています。Flash Player 6 用にパブリッシュされたファイルの場合、デフォルト値は `false` になり、スーパードメイン一致規則が使用されます。Flash Player 7 以降用にパブリッシュされたファイルの場合、デフォルト値は `true` になり、ドメイン完全一致規則が使用されます。設定データやローカルの永続データを使用して、Flash Player 7 以降用に Flash Player 6 の SWF ファイルをパブリッシュする場合は、移植済みファイルでこの値を `false` に設定する必要があります。詳細については、『ActionScript 2.0 リファレンスガイド』の「`exactSettings (System.exactSettings property)`」を参照してください。

SWF ファイル間のクロスドメインおよびサブドメインのアクセス

オンラインで互いに通信する一連の SWF ファイルを開発するとき、たとえば `loadMovie()`、`MovieClip.loadMovie()`、`MovieClipLoader.LoadClip()`、Local Connection オブジェクトなどを使用する場合は、それらの SWF ファイルを異なるドメイン内でホストすることも、単一スーパードメイン内の異なるサブドメイン内でホストすることもできます。

Flash Player 5 またはそれ以前のバージョン用にパブリッシュされたファイルでは、クロスドメインアクセスまたはサブドメインアクセスに対する制限はありません。

Flash Player 6 用にパブリッシュされたファイルでは、`LocalConnection.allowDomain` ハンドラまたは `System.security.allowDomain()` メソッドを使用して、許可済みのクロスドメインアクセス (例 : helpexamples.com のファイルから adobe.com のファイルへアクセス可能にする) を指定できます。サブドメインアクセスを許可するには (例 : something.adobe.com のファイルから www.adobe.com のファイルへアクセスするには)、コマンドは必要ありません。

Flash Player 7 用にパブリッシュされたファイルは、以前のバージョンとは異なる 2 つの方法で SWF ファイル間のアクセスを実装します。1 つ目は、Flash Player 7 にはスーパードメイン一致規則ではなく、ドメイン完全一致規則が使用されている点です。そのため、アクセス対象となるすべてのファイルで (Flash Player 7 より以前の Flash Player バージョン用にパブリッシュしたファイルも含め)、クロスドメインアクセスまたはサブドメインアクセスを明示的に許可する必要があります。詳細については、この項の説明を参照してください。2 つ目は、セキュリティ保護されたプロトコル (HTTPS) を使用するサイトでホストされたファイルでは、セキュリティ保護されていないプロトコル (HTTP または FTP) を使用するサイトでホストされたファイルからのアクセスを明示的に許可する必要がある点です。詳細については、641 ページの「SWF ファイル間の HTTP プロトコルから HTTPS プロトコルへのアクセス」を参照してください。

通常、`System.security.allowDomain` をアプリケーションに呼び出します。ただし、LocalConnection の受信側が HTTPS SWF ファイルで、送信側が HTTPS SWF ファイルでない場合、代わりに `allowInsecureDomain` が呼び出されます。

次の問題は、Flash Player 7 用にパブリッシュされた SWF ファイルにのみ影響を与えます。受信側が HTTPS で、送信側がローカル SWF ファイルの場合、本来 `allowInsecureDomain()` でなければならないときでも、`allowDomain()` が呼び出されます。ただし、Flash Player 8 以降では、HTTPS LocalConnection の受信側が Flash Player 8 以降で、送信側がローカルファイルの場合、`allowInsecureDomain()` が呼び出されます。

Flash Player 8 以降で実行されるファイルは、Flash Player 7 とは異なります。`System.security.allowDomain` を呼び出すと、アクセス対象の SWF ファイルが `System.security.allowDomain` を呼び出したファイルである場合にのみ、クロススクリプト操作が許可されます。つまり、`System.security.allowDomain` を呼び出す SWF ファイルはそのファイルだけにアクセスを許可することになります。旧バージョンでは、アクセス対象の SWF ファイルが `System.security.allowDomain` を呼び出したファイルである場合、`System.security.allowDomain` を呼び出すと、クロススクリプト操作が許可されます。クロススクリプト操作を実行すると、呼び出した SWF ファイルのドメイン全体が開かれます。

`System.security.allowDomain("*")` および `System.security.allowInsecureDomain("*")` のワイルドカード (*) 値に対するサポートが追加されました。アクセス元ファイルの種類を問わず、また、任意の場所からロードできる場合、ワイルドカード (*) 値を使用することでクロススクリプト操作が許可されます。ワイルドカード許可は便利ですが、Flash Player 8 以降の新しいローカルファイルセキュリティ規則に準拠していなければなりません。具体的には、ローカルファイルはドメインから取り込めないため、ワイルドカード値を使用することが必要になります。ただし、どのドメインからもファイルにアクセスできることから、ワイルドカード値を使用する場合は十分注意してください。詳細については、「`allowInsecureDomain (security.allowInsecureDomain メソッド)`」を参照してください。

呼び出したドメインではなく別のドメインから子 SWF ファイルをロードしてしまうことがあります。子 SWF ファイルに親 SWF ファイルをスクリプトさせようとしますが、子 SWF ファイルが派生した最終的なドメインが分りません。このような状況は、たとえばロードバランシングリダイレクトやサードパーティ製サーバーを使用する場合に発生します。このような状況では、`MovieClip._url` プロパティを引数として、このメソッドに使用できます。たとえば、SWF ファイルを `my_mc` にロードした場合は、`System.security.allowDomain(my_mc._url)` を呼び出すことができます。これを行った場合は、`my_mc` の SWF ファイルのロードが開始されるまで待ってください。これは、まだ `_url` プロパティに最終的な正しい値が入力されていないためです。子 SWF ファイルのロードがいつ開始されたかを確認するには、`MovieClipLoader.onLoadStart` を使用します。

この反対の状況が発生する場合もあります。つまり、親 SWF ファイルにスクリプトを許可する子 SWF ファイルを作成したものの、親 SWF ファイルのドメインがどれになるかが不明な場合です。つまり、その SWF ファイルはさまざまなドメインからロードされる可能性があるということになります。こうした状況では、子 SWF ファイルから `System.security.allowDomain(_parent._url)` を呼び出します。親 SWF ファイルは子 SWF ファイルより前にロードされるため、親 SWF ファイルがロードされるまで待つ必要はありません。

メモ

アクセス先のインターネット SWF ファイルが HTTPS URL からロードされる場合は、そのインターネット SWF ファイルで `System.security.allowInsecureDomain("*")` を呼び出す必要があります。

次の表は、各種バージョンの Flash Player におけるドメイン一致規則をまとめたものです。

Flash Player 用にパブリッシュされたファイル	SWF ファイル間のクロスドメインアクセス (allowDomain() が必要)	SWF ファイル間のサブドメインアクセス
5 以前	制限なし	制限なし
6	スーパードメイン一致 : スーパードメインが一致しない場合、`allowDomain()` が必要になります。	制限なし
7 以降	ドメイン完全一致 HTTPS でホストされたファイルから HTTP または FTP でホストされたファイルにアクセスするための明示的許可	ドメイン完全一致 HTTPS でホストされたファイルから HTTP または FTP でホストされたファイルにアクセスするための明示的許可

メモ

Flash Player 7 以降で HTTP-to-HTTPS アクセスを実行する場合は、ドメインが完全に一致しても、`System.security.allowInsecureDomain` が必要です。

Flash Player の動作を制御するバージョンは、Flash Player 自身のバージョンでなく、SWF バージョン (SWF ファイルの指定 Flash Player バージョン) です。たとえば、Flash Player 8 以降でバージョン 7 用にパブリッシュされた SWF ファイルを再生する場合、バージョン 7 と一貫した動作が適用されます。これによって、アップグレードしても、展開された SWF ファイル内の `System.security.allowDomain()` の動作は変更されなくなります。

Flash Player 7 以降では、スーパードメイン一致規則ではなくドメイン完全一致規則が使用されるため、Flash Player 7 以降用にパブリッシュされたファイルから既存のスクリプトにアクセスするには、既存のスクリプトを修正する必要があります。修正したファイルを Flash Player 6 用にパブリッシュすることは可能です。ファイル内で `LocalConnection.allowDomain()` ステートメントまたは `System.security.allowDomain()` ステートメントが使用され、許可対象としてスーパードメインサイトが指定されている場合は、パラメータを変更して、代わりにドメインを直接指定する必要があります。次の例は、Flash Player 6 コードの場合に実行しなければならない変更内容を示しています。

```
// www.helpexamples.com の SWF ファイルに Flash Player 6 コマンドを指定
// www.adobe.com または store.adobe.com でホストされた
// SWF ファイルからのアクセスを許可する
System.security.allowDomain("adobe.com");
my_lc.allowDomain = function(sendingDomain) {
   return(sendingDomain=="adobe.com");
}
// Flash Player 7 以降用にパブリッシュされた SWF ファイル
// によるアクセスを許可するコマンド
System.security.allowDomain("www.adobe.com", "store.adobe.com");
my_lc.allowDomain = function(sendingDomain) {
   return(sendingDomain=="www.adobe.com" ||
      sendingDomain=="store.adobe.com");
}
```

このようなステートメントを、現在使用していないファイルにも追加することが必要になる場合があります。たとえば、SWF ファイルが www.adobe.com でホストされていて、store.adobe.com にある Flash Player 7 以降用にパブリッシュされた SWF ファイルからのアクセスを許可する場合、www.adobe.com にあるアクセス対象のファイルに次のようなステートメントを追加する必要があります。ただし、このファイルは引き続き Flash Player 6 用にパブリッシュできます。

```
System.security.allowDomain("store.adobe.com");
my_lc.allowDomain = function(sendingDomain) {
   return(sendingDomain=="store.adobe.com");
}
```

また、Flash Player 6 アプリケーションを Flash Player 7 で実行する場合、ドメインが完全に一致しないデータにアプリケーションがアクセスしようとすると、Flash Player 7 以降のドメイン一致規則が適用され、ユーザーはアクセスの許可または拒否を選択するよう求められます。

つまり、次の条件を満たすファイルを Flash Player 7 以降用にパブリッシュする場合は、allowDomain ステートメントを追加または変更する必要があります。

- クロスドメイン SWF ファイルのスクリプトが実装済みです。詳細については、634 ページの「クロスドメイン SWF ファイル間のデータアクセスの許可」を参照してください。
- 呼び出し先の SWF ファイル (任意のバージョン) が、セキュリティ保護プロトコル (HTTPS) を使用するサイトでホストされていない。または、呼び出し元および呼び出し先の SWF ファイルがどちらも HTTPS サイトでホストされている。呼び出し先 SWF ファイルが HTTPS である場合の詳細については、641 ページの「SWF ファイル間の HTTP プロトコルから HTTPS プロトコルへのアクセス」を参照してください。
- SWF ファイルが異なるドメインに属しています (たとえば、1 つのファイルが www.adobe.com に、別のファイルが store.adobe.com にある場合など)。

次のように変更する必要があります。

- 呼び出し先 SWF ファイルを Flash Player 7 以降用にパブリッシュする場合、ドメイン完全一致規則を使用して System.security.allowDomain または LocalConnection.allowDomain を呼び出し先 SWF ファイルに追加します。
- 呼び出し先 SWF ファイルを Flash Player 6 用にパブリッシュする場合は、ドメイン完全一致規則を使用して呼び出し先ファイルを修正して、System.security.allowDomain または LocalConnection.allowDomain ステートメントを追加または変更します。この項の前半にあるコード例を参照してください。修正済みファイルは、Flash Player 6 用または Flash Player 7 用にパブリッシュできます。
- 呼び出し先 SWF ファイルを Flash Player 5 以前のバージョン用にパブリッシュする場合は、ドメイン完全一致規則を使用して、呼び出し先ファイルを Flash Player 6 または 7 に移植し、System.security.allowDomain ステートメントを追加します。この項の前半にあるコード例を参照してください。LocalConnection オブジェクトは、Flash Player 5 以前のバージョンではサポートされていません。

ローカルセキュリティサンドボックスについては、611 ページの「ローカルファイルセキュリティ と Flash Player について」を参照してください。

クロスドメイン SWF ファイル間のデータアクセスの許可

2 つの SWF ファイルが互いのデータ (変数とオブジェクト) にアクセスできるようにするには、2 つのファイルが同じドメインに存在する必要があります。Flash Player 7 以降では、デフォルトで 2 つの SWF ファイル間でデータを共有するには、両ファイルのドメインが完全に一致する必要があります。ただし、SWF ファイルは、LocalConnection.allowDomain または System.security.allowDomain() を呼び出すと、特定のドメインからのサービスを受けている複数の SWF ファイルが別の SWF ファイルにアクセスできるようになります。

System.security.allowDomain()allowDomain() 呼び出しを含むファイル内のオブジェクトと変数にアクセスできるドメインを、SWF ファイルおよび HTML ファイルから指定できます。

たとえば http://adobe.com/movieA.swf と http://adobe.com/movieB.swf のように、2 つの SWF ファイルが同じドメインからサービスを受けている場合、movieA.swf で movieB.swf 内の変数、オブジェクト、プロパティ、メソッドなどを調査、変更でき、movieB でも movieA に対して同様のことを実行できます。これを " クロススクリプト " と呼びます。

たとえば http://adobe.com/movieA.swf と http://helpexamples.com/movieB.swf のように、2 つの SWF ファイルが異なるドメインからサービスを受けている場合、デフォルトでは、movieA.swf で movieB.swf をスクリプトすることも、movieB で movieA をスクリプトすることも許可されません。movieB.swf では、`System.security.allowDomain("adobe.com")` を呼び出すことで、movieA.swf に movieB.swf をスクリプトする許可を与えます。SWF ファイルでは、他のドメインの SWF ファイルに対して、`System.security.allowDomain()` を呼び出すことで、自分自身をスクリプトする許可を与えます。これをクロスドメインスクリプトと呼びます。

`System.security.allowDomain()`、クロススクリプト、およびクロスドメインスクリプトの詳細については、『ActionScript 2.0 リファレンスガイド』の「allowDomain (security.allowDomain メソッド)」を参照してください。

たとえば、メイン SWF ファイルが www.adobe.com からサービスを受けていて、これが data.adobe.com から別の SWF ファイル (data.swf) を、`createEmptyMovieClip()` を使用して動的に作成されるムービークリップインスタンスにロードするとします。

```
// adobe.swf 内
this.createEmptyMovieClip("target_mc", this.getNextHighestDepth());
target_mc.loadMovie("http://data.adobe.com/data.swf");
```

さらに、"data.swf" がメインタイムライン上に `getData()` という名前のメソッドを定義しているとします。デフォルトでは、"main.swf" は "data.swf" のロード後、そのファイルで定義されている `getData()` メソッドを呼び出すことができません。2 つの SWF ファイルが同じドメインにないためです。たとえば、"data.swf" をロードした後、"main.swf" で次のメソッド呼び出しを行うとエラーになります。

```
// data.swf をロードした後の adobe.swf 内
target_mc.getData(); // このメソッド呼び出しはエラーになる
```

ただし、`LocalConnection.allowDomain` ハンドラおよび `System.security.allowDomain()` メソッドを使用すると、www.adobe.com のサービスを利用している SWF ファイルが "data.swf" にアクセスできる場合があります。"data.swf" に次のコードを追加すると、www.adobe.com のサービスを利用しているすべての SWF がこのファイルの変数とメソッドにアクセスできるようになります。

```
// data.swf の内部
this._lockroot = true;
System.security.allowDomain("www.adobe.com");
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain:String):Boolean {
  return (sendingDomain == "www.adobe.com");
};
function getData():Void {
  var timestamp:Date = new Date();
  output_txt.text += "data.swf:" + timestamp.toString() + "\n\n";
}
output_txt.text = "**INIT**:\n\n";
```

これで、ロードされた SWF ファイル内の getData 関数を、"adobe.swf" ファイルで呼び出すことができます。allowDomain() では、アクセスされる側の SWF ファイルがセキュリティ保護プロトコル (HTTPS) を使用したサイトでホストされていない限り、許可されたドメイン内の任意の SWF ファイルに対して、他の任意の SWF ファイルをスクリプトする許可が与えられる点に注意してください。

ドメイン一致規則の詳細については、630 ページの「SWF ファイル間のクロスドメインおよびサブドメインのアクセス」を参照してください。

サーバーサイドポリシーファイルによるデータへのアクセス許可

Flash ドキュメントで外部ソースからデータをロードする場合は、データロード呼び出し方法として、XML.load()、XML.sendAndLoad()、LoadVars.load()、LoadVars.sendAndLoad()、loadVariables()、loadVariablesNum()、MovieClip.loadVariables()、XMLSocket.connect()、および Flash Remoting (NetServices.createGatewayConnection) を使用します。また、SWF ファイルは、実行時にランタイム共有ライブラリ (RSL) や、他の SWF ファイルに定義されているアセットを読み込むことができます。デフォルトでは、データまたは RSL は、外部データやメディアをロードする SWF ファイルと同じドメインに属している必要があります。

ランタイム共有ライブラリのデータやアセットを異なるドメインの SWF ファイルでも利用できるようにするには、" ドメイン間ポリシーファイル " を使用する必要があります。ドメイン間ポリシーファイルは XML ファイルであり、サーバーのデータとドキュメントをどのドメインの SWF ファイルが利用できるかを示すものです。ドメインは、特定のものを指定することも、すべてを指定することもできます。サーバーのポリシーファイルに指定されたドメインが提供する SWF ファイルであれば、必ずサーバーのデータ、アセット、または RSL にアクセスすることができます。

外部データをロードしている場合は、ファイルを Flash Player 7 に移植する予定がなくてもポリシーファイルを作成する必要があります。RSL を使用している場合は、呼び出し元ファイルと呼び出し先ファイルのいずれか一方でも Flash Player 7 用にパブリッシュするのであればポリシーファイルを作成する必要があります。

詳細については、次のトピックを参照してください。

- 637 ページの「ドメイン間のデータロード許可」
- 638 ページの「カスタムポリシーファイルの場所について」
- 639 ページの「XMLSocket ポリシーファイルについて」

ドメイン間のデータロード許可

Flash ドキュメントが別のドメインのデータにアクセスしようとすると、Flash Player が自動的にそのドメインからポリシーファイルをロードしようとします。データをアクセスしようとしている Flash ドキュメントのドメインがポリシーファイルに含まれている場合、データへのアクセスが自動的に許可されます。

ポリシーファイルは必ず crossdomain.xml という名前にし、ルートディレクトリに置くか、別途 ActionScript でデータを提供しているサーバー上のディレクトリに置きます (638 ページの「カスタムポリシーファイルの場所について」を参照)。ポリシーファイルは、HTTP、HTTPS、または FTP 経由で通信するサーバーでのみ機能します。ポリシーファイルは、そのファイルが置かれているサーバーの特定のポートとプロトコルに固有です。

たとえば、https://www.adobe.com:8080/crossdomain.xml にあるポリシーファイルは、HTTPS 経由でポート 8080 から www.adobe.com に対して呼び出されるデータロード呼び出しにのみ適用されます。

ただし、別のドメインのソケットサーバーに接続するために XMLSocket オブジェクトを使用する場合は、この規則は当てはまりません。その場合、同じドメインでポート 80 でソケットサーバーとして実行されている HTTP サーバーが、メソッド呼び出し用のポリシーファイルを提供する必要があります。

XML ポリシーファイルには、`<cross-domain-policy>` タグが 1 つ含まれています。この `<cross-domain-policy>` タグには、`<allow-access-from>` タグが含まれている場合もあれば含まれていない場合もあります。各 `<allow-access-from>` タグには、`domain` という属性が 1 つ含まれています。この属性は、正確な IP アドレス、正確なドメイン、またはワイルドカードドメイン (任意のドメイン) のいずれかを指定します。ワイルドカードドメインを指定するには、アスタリスク (*) 1 つを使用するか、アスタリスクの後に接尾辞を続けます。前者はすべてのドメインとすべての IP アドレスを意味し、後者は指定の接尾辞で終わるドメインのみを意味します。接尾辞は、ドットで始める必要があります。ただし、この先頭のドットは検索には使用されません。ドットを除いた接尾辞に一致するドメインだけが検索されます。たとえば、adobe.com は *.adobe.com の一部と見なされます。IP ドメインの指定にワイルドカードは使用できません。

IP アドレスを指定した場合は、IP シンタックス (http://65.57.83.12/flashmovie.swf など) を使用してその IP アドレスからロードされた SWF ファイルにのみアクセスが付与されます。ドメイン名シンタックスを使用してロードされた SWF ファイルには付与されません。Flash Player は、DNS 解決を行いません。

ここで、ポリシーファイルの例を紹介します。adobe.com の Flash ドキュメントから、adobe.com、www.helpexamples.com、*.adobe.com、および 105.216.0.40 の Flash ドキュメントへのアクセスを許可するポリシーファイルです。

```
<?xml version="1.0"?>
<!-- http://www.adobe.com/crossdomain.xml -->
<cross-domain-policy>
  <allow-access-from domain="www.helpexamples.com" />
  <allow-access-from domain="*.adobe.com" />
  <allow-access-from domain="105.216.0.40" />
</cross-domain-policy>
```

また、次のように、すべてのドメインのドキュメントに対してアクセスを許可することもできます。

```
<?xml version="1.0"?>
<!-- http://www.adobe.com/crossdomain.xml -->
<cross-domain-policy>
  <allow-access-from domain="*" />
</cross-domain-policy>
```

各 `<allow-access-from>` タグには、オプションで `secure` 属性を指定することもできます。`secure` 属性は、デフォルトで `true` に設定されています。ポリシーファイルが HTTPS サーバー上に置かれているとき、HTTP サーバー上の SWF ファイルによって、HTTPS サーバーからデータをロードするには、この属性を `false` に設定します。

`secure` 属性を `false` に設定すると、HTTPS のセキュリティが影響を受けます。

HTTPS サーバーから SWF ファイルをダウンロードしているとき、それをロードする SWF ファイルが HTTP サーバー上に置かれている場合は、次のように、`secure="false"` 属性を `<allow-access-from>` タグに追加する必要があります。

```
<allow-access-from domain="www.adobe.com" secure="false" />
```

`<allow-access-from>` タグを含まないポリシーファイルは、サーバーにポリシーファイルを設定されていない場合と同じ結果を生みます。

カスタムポリシーファイルの場所について

Flash Player 7 (7.0.19.0) は、`System.security.loadPolicyFile` というメソッドをサポートしています。このメソッドを使用すると、クロスドメインポリシーファイルの検索先として任意の場所を指定でき、必ずしもルートディレクトリである必要がなくなります。Flash Player 7 (7.0.14.0) では、サーバーのルートに置かれているポリシーファイルしか検索されませんでしたが、サイト管理者にとっては、このファイルがルートディレクトリに格納されていると都合の悪い場合があります。`loadPolicyFile` メソッドと XMLSocket 接続の詳細については、『ActionScript 2.0 リファレンスガイド』の 639 ページの「XMLSocket ポリシーファイルについて」および「`loadPolicyFile (security.loadPolicyFile メソッド )`」を参照してください。

loadPolicyFile メソッドを使用した場合、サイト管理者は、ポリシーファイルを必要とする SWF ファイルが loadPolicyFile を呼び出して、このファイルの場所を Flash Player に通知できる限り、ポリシーファイルを任意のディレクトリに格納できます。ただし、ルートディレクトリに置かれていないポリシーファイルには、スコープの制限が適用されます。ポリシーファイルは、サーバーの階層において、自分と同じレベル以下の場所にしかアクセスを許可しません。

loadPolicyFile メソッドは、Flash Player 7 (7.0.19.0) 以降でのみ使用できます。SWF ファイルの作成者は、loadPolicyFile メソッドを使用する場合、次のいずれかが必要になります。

- Flash Player 7 (7.0.19.0) 以降を使用すること。
- データの送信元となるサイトで、デフォルトの場所 (ルートディレクトリ) とデフォルトでない場所の両方にポリシーファイルを配置すること。Flash Player の旧バージョンでは、デフォルトの場所が使用されます。

それ以外の場合、作成者は、ドメイン間の処理でエラーが発生した場合に実行される SWF ファイルを作成する必要があります。

注意	SWF ファイルが loadPolicyFile に依存している場合、Flash Player 6 以前または Flash Player 7 (7.0.19.0) 以降を使用しているユーザーには問題は生じません。ただし、Flash Player 7 (7.0.14.0) ユーザーの場合、loadPolicyFile がサポートされません。

デフォルト以外の場所にポリシーファイルを置き、そのポリシーファイルに依存する要求を行う場合は、次のように、System.security.loadPolicyFile を呼び出しておく必要があります。

```
System.security.loadPolicyFile("http://www.adobe.com/folder1/folder2/
  crossdomain.xml");
var my_xml:XML = new XML();
my_xml.load("http://www.adobe.com/folder1/folder2/myData.xml");
```

loadPolicyFile を使用して、スコープが重複する複数のポリシーファイルをロードできます。Flash Player は、すべての要求について、スコープにその要求の場所が含まれるファイルをすべて照会します。あるポリシーファイルでドメイン間アクセスが許可されていなくても、別のファイルでデータへのアクセスを許可できます。すべてのポリシーファイルでアクセスが拒否された場合、Flash Player は、crossdomain.xml ファイルのデフォルトの場所 (ルートディレクトリ) を参照します。デフォルトの場所にポリシーファイルが見つからなかった場合、その要求はエラーになります。

XMLSocket ポリシーファイルについて

XMLSocket 接続の場合、Flash Player 7 (7.0.14.0) は、接続先となるサブドメインの HTTP サーバー (ポート 80) を対象に crossdomain.xml を検索します。Flash Player 7 (7.0.14.0) 以前のバージョンでは、ポート 1024 以上に対する XMLSocket 接続は制限されていました。ただし、Flash Player 7 (7.0.19.0) 以降では、System.security.loadPolicyFile を使用して、ActionScript から Flash Player に対し、デフォルト以外の場所に置かれたポリシーファイルを指定することができます。XMLSocket のポリシーファイルは、XML ソケットサーバー上であれば、任意の場所に置くことができます。

次の例では、Flash Player は指定された URL からポリシーファイルを取得しています。

```
System.security.loadPolicyFile("http://www.adobe.com/folder/policy.xml");
```

この場所に置かれたポリシーファイルによって付与されるアクセス許可は、サーバーの階層で、ポリシーファイルと同じかそれ以下のレベルにあるすべてのコンテンツに適用されます。そのため、次のデータをロードしようとしても、特定の場所からしかロードできません。

```
myLoadVars.load("http://www.adobe.com/folder/vars.txt"); // 許可
myLoadVars.load("http://www.adobe.com/folder/dir/vars2.txt"); // 許可
myLoadVars.load("http://www.adobe.com/elsewhere/vars3.txt"); // 拒否
```

これを避けるには、`loadPolicyFile` を使用して、1つの SWF ファイルに複数のポリシーファイルをロードします。Flash Player は、ポリシーファイルのダウンロードがすべて完了した上で、ポリシーファイルが必要な要求を拒否します。Flash Player は、SWF ファイルで指定されたポリシーが他に存在しない場合、crossdomain.xml のデフォルトの場所を参照します。

特別なシンタックスによって、ポリシーファイルを XMLSocket サーバーから直接取得できます。

```
System.security.loadPolicyFile("xmlsocket://adobe.com:414");
```

この例では、Flash Player は、指定されたホストとポートから、ポリシーファイルを取得しようとします。ポリシーファイルがデフォルト (ルート) 以外のディレクトリに置かれている場合は、任意のポートを使用できます。ただし、それ以外の場合は、(従来のプレイヤーと同様) 1024 番以上のポートに限定されます。指定されたポートとの接続が確立されると、Flash Player は `<policy-file-request />` を送信します (null バイトにより終了)。

ポリシーファイルを提供する XML ソケットサーバーは、次のように構成する必要があります。

- ポリシーファイルと通常のソケット接続を同じポートを介して提供する。サーバーは `<policy-file-request />` を受け取ってからでなければ、ポリシーファイルを終了できません。
- ポリシーファイルを通常の接続とは独立したポートで提供する。この場合、ポリシーファイル専用のポートに対する接続が確立してすぐにポリシーファイルが送信されます。

サーバーは、接続を終了する前に null バイトを送信して、ポリシーファイルを終了する必要があります。Flash Player は終了 null バイトを受け取った時点で、サーバーとの接続を終了します。

XML ソケットサーバーによって提供されるポリシーファイルには、アクセスを許可するポートの指定が必要という点以外は、他のポリシーファイルと同じシンタックスを使用します。許可するポートは、`<allow-access-from>` タグの `to-ports` 属性で指定します。ポリシーファイルに 1024 番より小さいポートを指定した場合、すべてのポートに対するアクセスが許可されます。1024 番以上のポートからポリシーファイルが送られてきた場合は、1024 より大きい他のポートに対するアクセスだけが許可されます。ポート番号の指定には、単一のポート番号またはポート番号範囲を指定したり、ワイルドカードを使用したりすることもできます。XMLSocket ポリシーファイルのコード例を次に示します。

```
<cross-domain-policy>
<allow-access-from domain="*" to-ports="507" />
<allow-access-from domain="*.adobe.com" to-ports="507,516" />
<allow-access-from domain="*.helpexamples.com" to-ports="516-523" />
<allow-access-from domain="www.adobe.com" to-ports="507,516-523" />
<allow-access-from domain="www.helpexamples.com" to-ports="*" />
</cross-domain-policy>
```

Flash Player 7 (7.0.19.0) 以降では、1024 番より小さいポートに対して接続できるため、SWF ファイルがそれ自身のサブドメインに接続している場合でも、`loadPolicyFile` でロードされるポリシーファイルで常にこれを許可する必要があります。

SWF ファイル間の HTTP プロトコルから HTTPS プロトコルへのアクセス

あるドメインに存在する SWF ファイルから別のドメインに存在する SWF ファイルにアクセスできるようにするには、`allowDomain` ハンドラまたはメソッドを使用する必要があります。ただし、アクセスされる側の SWF ファイルが、セキュリティ保護プロトコル (HTTPS) を使用したサイトでホストされている場合、`allowDomain` ハンドラまたはメソッドは、セキュリティ保護されていないプロトコルのサイトでホストされた SWF ファイルからのアクセスを許可しません。このようなアクセスを許可するためには、`LocalConnection.allowInsecure Domain()` または `System.security.allowInsecureDomain()` のいずれかのステートメントを使用する必要があります。詳細については、641 ページの「SWF ファイル間の HTTP プロトコルから HTTPS プロトコルへのアクセスの許可」を参照してください。

SWF ファイル間の HTTP プロトコルから HTTPS プロトコルへのアクセスの許可

ドメイン完全一致規則に加え、セキュリティ保護されていないプロトコルを使用するサイトでホストされたファイルからのアクセスを許可するためには、セキュリティ保護プロトコル (HTTPS) を使用するサイトでホストされたファイルへのアクセスを明示的に許可する必要があります。呼び出し先ファイルを Flash Player 6 以降のどのバージョン用にパブリッシュするかによって、`allowDomain` ステートメントのいずれかを使用するか (630 ページの「SWF ファイル間のクロスドメインおよびサブドメインのアクセス」を参照)、`LocalConnection.allowInsecure Domain` ステートメントまたは `System.security.allowInsecureDomain()` ステートメントを使用するかを判断する必要があります。

たとえば、http://www.adobe.com に存在する SWF ファイルから、https://www.adobe.com/data.swf に存在する SWF ファイルへのアクセス許可を設定するためには、"data.swf" に次のようなコードを追加する必要があります。

```
// data.swf の内部
System.security.allowInsecureDomain("www.adobe.com");
my_lc.allowInsecureDomain = function(sendingDomain:String):Boolean {
   return (sendingDomain == "www.adobe.com");
};
```

警告

`allowInsecureDomain()` ステートメントを実装すると、HTTPS プロトコルによるセキュリティが損なわれます。すべての SWF ファイルが HTTPS プロトコルからサービスされるようにサイトを再編成できない場合以外は、これらの変更を適用しないでください。

次のコードは、必要となる変更の例を示したものです。

```
// https://www.adobe.com の Flash Player 6 用 SWF ファイルのコマンド
// http://www.someSite.com または http://www.someOtherSite.com
// でホストされた Flash Player 7 用 SWF ファイルによるアクセスを許可する
System.security.allowDomain("helpexamples.com");
my_lc.allowDomain = function(sendingDomain) {
   return(sendingDomain=="helpexamples.com");
}
// Flash Player 7 用 SWF ファイルのコマンド
// http://www.someSite.com または http://www.someOtherSite.com
// でホストされた Flash Player 7 用 SWF ファイルによるアクセスを許可する
System.security.allowInsecureDomain("www.adobe.com", "www.helpexamples.com");
my_lc.allowInsecureDomain = function(sendingDomain) {
   return(sendingDomain=="www.adobe.com" ||
      sendingDomain=="www.helpexamples.com");
}
```

このようなステートメントを、現在使用していないファイルにも追加することが必要になる場合があります。http://www.adobe.com のファイルから https://www.adobe.com のファイルを呼び出すなど、両方のファイルが同じドメインに属していても変更が必要になる場合があります。

つまり、次の条件を満たすファイルを Flash Player 7 以降用にパブリッシュする場合は、ファイルを修正してステートメントを追加または変更する必要があります。

- `loadMovie()`、`MovieClip.loadMovie()`、`MovieClipLoader.LoadClip()`、または Local Connection などのオブジェクトを使用したクロス SWF ファイルスクリプトが実装されている。
- 呼び出し元のファイルが HTTPS プロトコルを使用してホストされていないこと。かつ、呼び出し元ファイルが HTTPS プロトコルを使用してホストされていること。

次のように変更する必要があります。

- 呼び出し先ファイルを Flash Player 7 用にパブリッシュする場合は、この項で取り上げたコード例のように、ドメイン完全一致規則を使用して、`System.security.allowInsecureDomain` または `LocalConnection.allowInsecureDomain` を呼び出し先ファイルに追加します。
- 呼び出し元ファイルが Flash Player 6 以前用にパブリッシュされ、呼び出し元ファイルも呼び出し先ファイルも同じドメインの場合 (http://www.adobe.com のファイルが https://www.adobe.com のファイルを呼び出す場合) は、修正する必要はありません。
- 呼び出し先ファイルを Flash Player 6 用にパブリッシュする場合で、ファイルが異なるドメインに属していて、呼び出し先ファイルを Flash Player 7 に移植しない場合は、ドメイン完全一致規則を使用して呼び出し先ファイルを修正し、`System.security.allowDomain` ステートメントまたは `LocalConnection.allowDomain` ステートメントを追加または変更します。この項のコード例を参照してください。
- 呼び出し先ファイルを Flash Player 6 用にパブリッシュし、Flash Player 7 に移植する場合は、ドメイン完全一致規則を使用して、`System.security.allowInsecureDomain` または `LocalConnection.allowInsecureDomain` を呼び出し先ファイルに追加します。この項のコード例を参照してください。
- 呼び出し先ファイルを Flash Player 5 以前のバージョン用にパブリッシュし、両方のファイルが同じドメインに属していない場合は、2 つの選択肢があります。呼び出し先ファイルを Flash Player 6 に移植し、ドメイン完全一致規則を使用して、`System.security.allowDomain` ステートメントを追加または変更するか、呼び出し先ファイルを Flash Player 7 に移植し、ドメイン完全一致規則を使用して、`System.security.allowInsecureDomain` ステートメントを呼び出し先ファイルに追加するかの、いずれかの方法を選択できます。この項のコード例を参照してください。

第 17 章

ActionScript 2.0 のベストプラクティスとコーディング規則

17

Adobe Flash のデザイナーや開発者は、自分自身にとっても、また同じオブジェクトを使用する他の人々にとっても理解しやすい直観的な方法でコードを記述し、アプリケーションを構築する必要があります。大量のアセットまたは長いコードファイルを持つ FLA ファイルでは、これがとりわけ重要となります。ベストプラクティスとコーディング規則に従うことで、設計開発チームの全員がファイル構造と ActionScript コードを理解し、効率的に作業できるようになります。本マニュアルを利用すると、Flash の開発およびコーディングの作業を定型化できます。

Flash プロジェクトには複数のデザイナーや開発者が参加することが多いので、Flash の使用方法、FLA ファイルの構成、および ActionScript 2.0 コードの記述に関して標準のガイドラインを作成し、全員がそれに従うことは、チーム全体にとってのメリットになります。本章の各セクションでは、ActionScript の記述に関するベストプラクティスの概要を示します。『Flash ユーザーガイド』の一部のセクションでは、Flash オーサリングツール使用時のベストプラクティスについて説明します。

次のガイドラインは、Flash の使用方法や ActionScript コードの記述方法を学ぶ学習者に一貫性を保つよう促すものです。デザイナーか開発者かにかかわらず、また、単独の作業、チームの一員としての作業のいずれの場合でも、常に次のベストプラクティスに従うことをお勧めします。

- Flash ドキュメントまたは ActionScript ドキュメントに対して作業する場合

  一貫した効率的なプラクティスを使用することがワークフローの迅速化に役立ちます。コーディング規則を確立しておくと開発期間を短縮でき、また、ドキュメントの構成が理解しやすく覚えやすいものになるので、後から編集を加えることも容易になります。それだけでなく、大規模なプロジェクトのフレームワーク内ではコードのポータビリティがさらに高まり、より容易にコードを再利用できます。

- FLA ファイルまたは AS ファイルを共有する場合

  他人がドキュメントを編集する際に、ActionScript をすばやく検索して理解し、一貫性を持ってコードを変更し、アセットを検索および編集できます。

- アプリケーションに対して作業する場合

  複数の作成者がアプリケーションに対して作業する際にコンフリクトが発生しにくくなり、効率が向上します。ベストプラクティスとコーディング規則に従うことで、プロジェクトやサイトの管理者は、コンフリクトが発生したり冗長になったりするのを防ぎながら、複雑なプロジェクトやアプリケーションを管理および構築できます。

- Flash と ActionScript を学習または教育する場合

  ベストプラクティスと次のコーディング規則を使用してアプリケーションの作成方法を学習することで、特定の方法論を学習し直す必要性が小さくなります。Flash の学習者は、一貫性のある適切なコードの構成方法を学ぶことで、余分なストレスを経験せずに速く言語の学習を進めることができます。

一貫した手法と次に示すガイドラインは、Flash の学習やチーム環境での効率的な作業を容易にするものです。一貫した方法を使用していれば、以前扱っていた FLA ファイルを後になって自分自身が処理する際にも、ドキュメントをどのように構成したかを思い出すことができます。

これらは、ベストプラクティスを学習し、それに従う理由の一部にすぎません。ここで説明するベストプラクティスを読んで実践していくに従って、他にも多くの理由を必ず発見できるでしょう。以下の各トピックは、Flash での作業に関するガイドラインと考えてください。すべての推奨事項に従うか、一部のみ採用するかは自由です。また、自分の作業方法に合わせてガイドラインの内容を変更してもかまいません。本章に示すガイドラインの多くは、Flash での作業と ActionScript コードの記述について一貫性のある方法を確立するためのものです。

この章では、コーディング規則とベストプラクティスに関する次のトピックについて説明します。

命名規則 646
コード内のコメントの使用 656
ActionScript のコーディング規則 659
ActionScript と Flash Player の最適化 673
ActionScript シンタックスのフォーマット 675

命名規則

多くの場合、開発に要する時間の 80% は、デバッグ、トラブルシューティング、および一般的な保守作業に費やされます。特に、大規模なプロジェクトではこの種の作業時間が大きくなります。小規模なプロジェクトで作業している場合であっても、コードの分析と修正には多くの時間が費やされます。自分が作成するコードの読みやすさは、自分にとっても、チームのメンバーにとっても大切です。命名規則に従うことで読みやすさが向上すれば、ワークフローが改善され、コード内のエラーを発見して解決できるようになります。すべてのプログラマーが標準化した方法でコードを記述するようになれば、プロジェクトがさまざまな面で改善されます。

変数名に対する命名規則の使用は、次のような重要な役割を果たします。

- コードが読みやすくなり、変数のデータ型がすぐにわかります。学生、コードの学習者、またはコードに不慣れな開発者にとって役立ちます。
- 必要に応じて、変数名を簡単に検索し置換することができます。
- 予約語や言語コンストラクトとのコンフリクトを防ぐのに役立ちます。
- ローカル変数、クラスプロパティ、パラメータなど、異なるスコープの変数を区別しやすくします。

次の各セクションでは、ファイル、変数、定数、コンポーネントの命名など、ActionScript コードを記述する際の命名ガイドラインについて説明します。675 ページの「ActionScript シンタックスのフォーマット」では、ActionScript に特有のフォーマット規則と、他の言語に共通のフォーマット規則について説明します。659 ページの「ActionScript のコーディング規則」では、ActionScript の記述と Flash 8 での開発に特有のコーディング規則について説明します。

メモ

Flash Player 7 以降は、ECMAScript (ECMA-262) Edition 3 の言語仕様にほぼ準拠しています。言語の動作方法について、ECMA-262 edition 4 を参照するとよいでしょう。詳細については、www.ecma-international.org/publications/standards/Ecma-262.htm を参照してください。

このセクションでは、次のトピックを扱います。

- 648 ページの「一般的な命名ガイドライン」
- 648 ページの「予約語および言語コンストラクトの回避」
- 650 ページの「変数の命名」
- 652 ページの「定数の命名」
- 652 ページの「ブール型変数の命名」
- 653 ページの「関数とメソッドの命名」
- 653 ページの「クラスとオブジェクトの命名」
- 655 ページの「パッケージの命名」
- 655 ページの「インターフェイスの命名」
- 656 ページの「カスタムコンポーネントの命名」

一般的な命名ガイドライン

このセクションでは、ActionScript コードを記述する際の命名ガイドラインについて確認します。命名規則は、論理的なコードの記述にとって重要なものです。一番の目的は、ActionScript 2.0 コードを読みやすくすることです。すべての変数には一意の名前を指定する必要があります。Flash Player 7 以降では大文字と小文字が区別されます。ただし、大文字と小文字が異なるだけの同じ名前は使用しないでください。コードを参照するプログラマが混乱したり、大文字と小文字が区別されない Flash の以前のバージョンで問題が生じる可能性があります。Flash 内の変数、ファイル、クラスなどのアイテムに名前を付ける場合は、次のガイドラインに留意してください。

- 略語の使用方法を限定します。

  略語は一貫性を持って使用してください。略語では、明確に 1 つの意味を表すようにする必要があります。たとえば、"sec" という略語が "section" (セクション) を表すのか、"second" (秒) を表すのかが明確でないと混乱を招きます。

- 単語を連結して名前を作成します。

  単語を連結する場合は、大文字と小文字を混在させることで単語の境界を判別できるようにします。たとえば、`mypelican` という名前よりも `myPelican` の方が読みやすくなります。

- ファイル名は、`addUser` のように、プロセスやアイテムを表す名前にします。
- メソッド名や変数名は、内容を説明できる名前にします。

  たとえば、訪問者のユーザー名を表すデータを取得するメソッドには、`getData()` ではなく `getUserName()` という名前を付ける方が適当です。この例の場合は、実現方法ではなく実行する操作の意味を表すようにします。

- すべての名前は、可能な限り短くします。

  ただし、意味が通じる名前を常に使用することが重要です。

以降のセクションでは、コード内の変数、クラス、パッケージ、定数などのアイテムの命名方法について詳細に説明します。

予約語および言語コンストラクトの回避

インスタンスや変数に名前を付ける場合は、予約語を使用しないようにしてください。予約語を名前に使用すると、コードでエラーが発生する可能性があります。予約語には、ActionScript 言語の " キーワード " を含みます。

また、ActionScript 2.0 言語に含まれる単語 (" 言語コンストラクト " と呼ばれます) はインスタンス名または変数名として一切使用しないでください。ActionScript コントラストには、クラス名、コンポーネントクラス名、メソッド名とプロパティ名、およびインターフェイス名を含みます。

警告	名前の大文字と小文字を替えることで予約語とのコンフリクトを回避しないでください。たとえば、TextField クラスのインスタンスに `textfield` という名前を付けると、(Flash では大文字小文字が区別されるため、TextField とのコンフリクトは発生しませんが) 推奨できないコーティング方法です。

次の表は、ActionScript 2.0 の予約語の一覧です。これらを変数名に使用するとスクリプトのエラーとなります。

add	and	break	case
catch	class	continue	default
delete	do	dynamic	else
eq	extends	false	finally
for	function	ge	get
gt	if	ifFrameLoaded	implements
import	in	instanceof	interface
intrinsic	le	it	ne
new	not	null	on
onClipEvent	or	private	public
return	set	static	super
switch	tellTarget	this	throw
try	typeof	undefined	var
void	while	with	

次の語句は、ECMAScript (ECMA-262) edition 4 の言語仕様草案により、Flash で将来使用するために予約されています。これらの語句は Flash の将来のリリースで使用される可能性があるため、使用を避けてください。

as	abstract	Boolean	bytes
char	const	debugger	double
enum	export	final	float
goto	is	long	namespace
native	package	protected	short
synchronized	throws	transient	use
volatile			

変数の命名

変数名には、英数字とドル記号 ($) のみを使用できます。変数名の先頭に数字は指定できません。変数名は一意でなければならず、Flash Player 7 以降のバージョンでは大文字と小文字が区別されます。たとえば、次のような変数名は使用しないでください。

```
my/warthog = true;   // スラッシュを含む
my warthogs = true;  // スペースを含む
my.warthogs = true; // ドットを含む
5warthogs = 55;    // 数字で始まる
```

変数に厳密な型指定をできるだけ使用することで、次の利点が得られます。

- コードコンプリート機能が追加され、コーディングが高速化されます。
- SWF ファイルのコンパイル時に、[出力] パネルにエラーが表示されるので、エラーを見逃すことがありません。このようなエラーは、アプリケーションの問題を見つけて修正するのに役立ちます。

変数にデータ型を追加するには、`var` キーワードを使用して変数を定義する必要があります。次の例では、LoadVars オブジェクトを作成するときに、厳密な型指定を使用しています。

```
var paramsLv:LoadVars = new LoadVars();
```

厳密な型指定によってコードコンプリート機能が有効になり、`paramsLv` には値として LoadVars オブジェクトだけが格納されることを保証できます。また、LoadVars オブジェクトに数値データやストリングデータが格納されないことも保証できます。厳密な型指定は `var` キーワードに依存するため、オブジェクト内および配列内のグローバル変数やプロパティに対しては使用できません。変数の厳密な型指定の詳細については、42 ページの「データ型の割り当てと厳密な型指定について」を参照してください。

メモ	厳密な型指定によって SWF ファイルの実行速度が低下することはありません。型チェックは、実行時ではなくコンパイル時 (SWF ファイルの作成時) に行われます。

コード内の変数に名前を付ける場合は、次のガイドラインに従ってください。

- すべての変数には一意の名前を指定する必要があります。
- 大文字と小文字が異なるだけの同じ変数名は使用しないようにします。

  1 つのアプリケーション内で `firstname` と `firstName` という 2 つの変数を使用しないでください。Flash Player 7 以降では大文字と小文字が区別されますが、大文字と小文字が異なるだけの同じ変数名を使用すると、コードを参照するプログラマが混乱する可能性や、大文字と小文字が区別されない Flash の以前のバージョンで問題が生じる可能性があります。
- ActionScript 1.0 または 2.0 の言語の一部となっている単語を変数名として使用しないようにします。

  特に、キーワードをインスタンス名として使用してはなりません。コードでエラーが発生する可能性があります。大文字と小文字の区別を、コンフリクトを回避してコードを動作させるための手段としないでください。

- プログラミングコンストラクトに使用されることが多い単語は、変数名として使用しないようにします。

  他のプログラミング言語で使用される言語コンストラクトは、たとえ Flash に含まれるものやサポートされているものでなくても、使用しないでください。たとえば、次のキーワードは変数として使用しないでください。

```
textfield = "myTextField";
switch = true;
new = "funk";
```

- コードに常に型注釈を追加します。

  型注釈は、変数の " 厳密な型指定 " または " 強力な型指定 " とも呼ばれます。次のような理由から、変数に型注釈を追加することは重要です。

  - コンパイル時にエラーが生成されるので、アプリケーションが警告なく失敗することを防止できます。
  - コードコンプリート機能を有効にします。
  - ユーザーがコードを理解しやすくなります。

  型注釈の追加方法については、42 ページの「データ型の割り当てと厳密な型指定について」を参照してください。

- オブジェクト型を不必要に多用しないようにします。

  パフォーマンスを向上させるには正確な型注釈を使用してください。オブジェクト型を使用するのは、他に現実的な代替手段がない場合だけにしてください。

- 変数名は、明確さを維持しながらもできるだけ短くします。

  常に内容がわかる変数名を付ける必要がありますが、複雑すぎる名前や長すぎる名前にならないよう注意してください。

- 1 文字の変数名は、ループの最適化のためにのみ使用します。

  ループ内の一時的な変数には、必要に応じて 1 文字の変数名を使用することができます (i、j、k、m、n など)。このような 1 文字の変数名は、短いループのインデックスや、パフォーマンスの最適化と速度がきわめて重要な場合に限り使用します。次の例は、この使用方法を示します。

```
var fontArr:Array = TextField.getFontList();
fontArr.sort();
var i:Number;
for (i = 0; i<fontArr.length; i++) {
   trace(fontArr[i]);
}
```

- 変数名は小文字で開始します。

  先頭が大文字の名前は、クラス、インターフェイスなどに予約されています。

- 単語を連結する場合は大文字と小文字を混在させます。

  たとえば、myfont ではなく myFont という名前を使用します。

- 頭字語や略語を使用しないようにします。

  ただし、頭字語や略語で表すことが一般的になっている用語 (HTML や CFM など) の場合は例外です。よく使用される頭字語の場合は、newHTMLParser ではなく newHtmlParser のように先頭の大文字以外は小文字にすると読みやすくなります。

- 関連性のある変数のセットを作成する際は、相補的な意味を持つ変数名をペアにして使用します。

  たとえば、ゲームの最低スコアと最高スコアを表すには次のような相補ペアを使用します。

```
var minScoreNum:Number = 10; // 最低スコア
var maxScoreNum:Number = 500; // 最高スコア
```

定数の命名

値が決して変化しないプロパティを参照する必要がある状況では、定数を使用できます。定数を使用すると、リテラル値を使用した場合よりもタイプミスを発見しやすくなります。また、1 か所で値を変更することもできます。

変数名の場合は小文字または大文字と小文字の混在で指定するのに対し、静的定数 (変化しない変数) の場合は次のガイドラインに従って命名します。

- 定数の名前は大文字で指定します。
- 単語間はアンダースコアで区切ります。

次の ActionScript コードは、これらのガイドラインを実際に適用した例です。

```
var BASE_URL:String = "http://www.adobe.com";  // 定数
var MAX_WIDTH:Number = 10;  // 定数
```

1、0、-1 のような定数を for ループのカウンタ値などとして使用する場合を除き、コード内には数値定数を直接に記述しないでください。

ブール型変数の命名

ブール型変数の名前は "is" という語で始めます。これは、ブール値はその性質上、"is" (... である) または "isn't" (... でない) のいずれかを表すためです。たとえば、赤ん坊が女の子であるか、女の子でないか (ブール値) を示す場合は、次のようなコードを使用します。

```
isGirl
```

また、ユーザーがログインしているか、いないかを示す変数の場合は、次のようなコードを使用します。

```
isLoggedIn
```

関数とメソッドの命名

コード内の関数とメソッドに名前を付ける場合は、次のガイドラインに従ってください。関数とメソッドの記述方法については、第 5 章の「関数とメソッド」を参照してください。

- 内容がわかる名前を使用します。
- 単語を連結する場合は大文字と小文字を混在させます。

  たとえば、`singLoud()` のような名前にします。
- 関数名とメソッド名は小文字で始めます。
- 関数名は、その関数が返す戻り値の意味がわかる名前にします。

  たとえば、曲名を返す関数には `getCurrentSong()` のような名前を付けます。
- 関連性のある類似の関数には、何らかの規則を定めて関数名を付けます。

  ActionScript 2.0 ではオーバーロードをサポートしていません。" オーバーロード " とは OOP (Object-Oriented Programming : オブジェクト指向プログラミング) の概念であり、渡される値のデータ型に応じて関数に異なる処理を実行させる機能のことです。
- メソッド名には動詞を使用します。

  複数の単語を連結する場合でも、その中に動詞を含めるようにします。ほとんどの場合、メソッド名には動詞を使用します。メソッドはオブジェクトに対する操作を実行するからです。

メソッド名の例を次に示します。

```
sing();
boogie();
singLoud();
danceFast();
```

クラスとオブジェクトの命名

新しいクラスファイルを作成する場合、クラスと ActionScript ファイルの命名には次のガイドラインを使用します。適切な形式で命名したクラス名の例を示します。

```
class Widget;
class PlasticWidget;
class StreamingVideo;
```

クラスには、パブリックメンバー変数とプライベートメンバー変数を設定できます。クラスには、ユーザーから直接に設定およびアクセスされることが望ましくない変数も含むことができます。そのような変数はプライベートにして、getter/setter メソッドを使用しないとアクセスできないようにします。

クラスの命名には、次のガイドラインが適用されます。

- クラス名は大文字で始めます。
- 単語を複合または連結してクラス名を作成する場合は、大文字と小文字を混在させます。

  単語を複合または連結したクラス名は、大文字で始めます。たとえば、NewMember のような名前にします。
- クラス名には通常、名詞または修飾語を伴う名詞を使用します。

  修飾語とは、その名詞やフレーズに説明を加える言葉です。たとえば、"member" という名詞に修飾語を付けると、NewMember や OldMember のようになります。
- 名前を短くすることよりも、わかりやすくすることを重視します。
- 頭字語や略語を使用しないようにします。

  ただし、頭字語や略語で表すことが一般的になっている用語 (HTML や CFM など) の場合は例外です。よく使用される頭字語の場合は、NewHTMLParser ではなく NewHtmlParser のように先頭の大文字以外は小文字にすると読みやすくなります。
- クラスの内容を説明する、意味のわかる簡潔な名前を使用します。

  あいまいであったり、誤解を招かないように、一般的な名前を使用してください。
- クラス名は複合語である場合もあります。

  修飾語とは、その名詞やフレーズに説明を加える言葉です。たとえば、"member" という名詞に修飾語を付けると、NewMember や OldMember のようになります。
- クラス名で使用する単語は複数形にしないようにします (Witches、BaldPirates など)。

  多くの場合、単に修飾語を伴う名詞にするのが適切です。修飾語とは、その名詞やフレーズに説明を加える言葉です。たとえば、"cat" や "buckaneer" という名詞に修飾語を付けると、BlackCat や OldBuckaneer のようになります。
- 冗長になるので、クラスのプロパティ名にはクラス名を含めないようにします。

  たとえば、Cat.catWhiskers は意味がありません。Cat.whiskers の方が適切です。
- 動詞としても解釈できる名詞は使用しないようにします。

  たとえば、Running や Gardening などです。このような名詞を使用すると、メソッド、状態、またはアプリケーションの他の動作と混同する可能性があります。
- 1 つのアプリケーション内では、各クラスに一意のクラス名を使用します。
- クラス名は、Flash のビルトインクラスの名前と競合しないようにします。
- できれば、階層構造内におけるそのクラスの関係を示すクラス名にします。

  これによって、アプリケーション内におけるクラスの関係がわかりやすくなります。たとえば、Widget というインターフェイスがある場合、Widget を実装するクラスには PlasticWidget、SteelWidget、および SmallWidget のような名前を付けます。

インターフェイスの詳細については、第 8 章の「インターフェイス」を参照してください。

パッケージの命名

パッケージ名には、" 反転ドメイン " 命名規則を使用するのが一般的です。反転ドメイン名の例としては、adobe.com に対する com.adobe、yourdomain.org に対する org.yourdomain などがあります。

パッケージ名を付けるときは、次のガイドラインに従ってください。

- パッケージ名には、小文字のみの接頭辞を付けます。

  たとえば、com、mx、org などです。
- 関連するクラス (関連する機能を持つクラス) は同じパッケージに入れます。
- パッケージ名には、一貫した接頭辞を付けます。

  たとえば、一貫性を維持するために、com.adobe.projectName を使用します。別の例としては、『ActionScript 2.0 の学習』で使用する com.adobe.docs.learnAS2.Users があります。
- 説明する必要がない明白なパッケージ名を使用します。

  パッケージの役割を表す名前にすることが重要です。たとえば、Pentagons という名前のパッケージがあり、ドキュメントの例で Flash の描画 API を使用して各種の五角形を描画する役割を果たす場合、その名前は com.adobe.docs.as2.Pentagons になります。
- 単語を複合または連結してパッケージ名を作成する場合は、大文字と小文字を混在させます。

  単語を複合または連結したパッケージ名とは、たとえば packageName のような名前です。接頭辞には、小文字のみ使用してください (com、org など)。
- アンダースコアやドル記号は使用しないようにします。

インターフェイスの命名

インターフェイス名の先頭に大文字の "I" を付けると、インターフェイス名とクラス名を区別するのに便利です。次のインターフェイス名 IEmployeeRecords は、先頭が大文字で始まり、大文字と小文字を混在させて単語を連結した名前になっています。

```
interface IEmployeeRecords{}
```

また、次の命名規則も適用されます。

- インターフェイス名は大文字で始めます。

  これはクラス名と同じです。
- インターフェイス名は、通常は形容詞にします。

  たとえば、Printable のような名前にします。

インターフェイスの詳細については、第 8 章の「インターフェイス」を参照してください。

カスタムコンポーネントの命名

コンポーネント名は大文字で始まり、大文字と小文字が混在する単語の連結で表されます。たとえば、次に示すデフォルトのユーザーインターフェイスコンポーネントセットでは、名前に単語の連結を使用し、大文字と小文字を混在させています。

- CheckBox
- ComboBox
- DataGrid
- DateChooser
- DateField
- MenuBar
- NumericStepper
- ProgressBar
- RadioButton
- ScrollPane
- TextArea
- TextInput

単語の連結を使用しないコンポーネントは、大文字で始まります。

カスタムコンポーネントを作成する場合は、Adobe コンポーネントの名前と互換性のある命名規則を使用してください。コンポーネントの名前は、Flash に含まれているデフォルトセットとは異なる名前にする必要があります。独自の一定の命名規則を使用すると、名前の重複を回避できます。

このセクションに示されている命名規則はあくまでもガイドラインです。自分に適した命名スキームを一貫して使用することが最も重要です。

コード内のコメントの使用

このセクションでは、コード内のコメントの使用方法について説明します。コメントはコード内で行う判断を記録するものであり、" 方法 " と " 理由 " の両方を示します。たとえば、回避策などをコメントで説明しておくことが考えられます。それにより、他の開発者が後に更新や修正をするときに、関連するコードを簡単に見つけられます。Flash または Flash Player の将来のバージョンで回避策が必要なくなった場合にも、コードの発見が容易になります。

ActionScript コード内のコメントの記述方法の詳細については、次のセクションを参照してください。

- 657 ページの「適切なコメントの書き方」
- 658 ページの「クラスへのコメントの追加」

適切なコメントの書き方

ActionScript 2.0 コード内で一貫性のあるコメントを使用すると、コメントがなければ不明確になる複雑なコード領域や重要なインタラクションに説明を付けることができます。コメントは、コードの目的を明確に説明する必要があり、コードを単に翻訳したようなものであってはなりません。コードを一見しただけでわからない内容があれば、そこにコメントを追加してください。

コードで自動フォーマットツールを使用すると、後続コメント (97 ページの「後続コメント」を参照) が次の行に移動することがわかります。このようなコメントはコードをフォーマットした後に追加することができます。さもなければ、自動フォーマットツールを使用した後でコメントの新しい位置を変更する必要があります。

クラス内でのコメントの使用の詳細については、658 ページの「クラスへのコメントの追加」を参照してください。

コードにコメントを追加するときは、次のガイドラインに従ってください。

- コメントが複数行にわたる場合はブロックコメント (/* と */) を使用し、短いコメントには単一行コメント (//) を使用します。

  また、必要に応じて、ActionScript コードと同じ行に記述する " 後続コメント " も使用できます。
- ActionScript コードを単に翻訳したようなコメントは使用しないようにします。

  ActionScript コードを読めばわかるエレメントにコメントを付ける必要はありません。
- コードを一見しただけでわからないエレメントには、コメントを付けます。

  特に、主題が前後の段落に記述されていない場合は、コメントを追加してください。
- 雑然としたコメントは使用しないようにします。

  雑然としたコメント行には、よく等号 (=) やアスタリスク (*) が含まれています。代わりに、空白を使ってコメントと ActionScript コードを区切ります。

メモ	自動フォーマットツールを使用して ActionScript をフォーマットすると、空白が削除されます。その場合は空白を追加し直すか、空白を維持するための単一行コメント (//) を使用します。このようなコメント行は、コードをフォーマットした後で簡単に削除できます。

- プロジェクトを展開する前に、余分なコメントをコードから削除します。

  ActionScript コード内のコメントが多くなった場合は、そのコードの一部を書き直す必要がないか検討してください。ActionScript コードの動作に関して多くのコメントを残す必要があると感じる場合、それは通常、ActionScript コードの書き方が良くないことの現れです。

メモ	読者に教えることが目的の ActionScript コードの中では、コメントの使用が最も重要です。たとえば、Flash の教育目的でサンプルアプリケーションを作成する場合や、ActionScript コードのチュートリアルを作成する場合には、コードにコメントを追加します。

クラスへのコメントの追加

典型的なクラスファイルまたはインターフェイスファイルのコメントには、" ドキュメントコメント " と " 実装コメント " の 2 種類があります。

メモ	ドキュメントコメントと実装コメントは、ActionScript 言語で正式には規定されていません。ただし、開発者がクラスファイルやインターフェイスファイルを作成するときに、これらのコメントはよく使用されます。

ドキュメントコメントはコードの仕様を説明するために使用し、実装内容についての説明は含めません。実装コメントは、コードをコメントアウトする目的や、コードの特定の部分の実装について説明を加えるために使用します。ドキュメントコメントは /** と */ で区切られ、実装コメントは /* と */ で区切られます。

ドキュメントコメントは、インターフェイス、クラス、メソッド、およびコンストラクタを説明するのに使用します。クラス、インターフェイス、またはメンバーごとに 1 つのドキュメントコメントを含めてください。ドキュメントコメントは、宣言の直前に配置します。ドキュメントコメントとしては適さない追加情報がある場合は、実装コメント (ブロックコメントまたは単一行コメントの形式) を使用します。

クラスの先頭には、次の形式を使用して標準のコメントを配置します。

```
/**
  User クラス
  バージョン 1.2
  3/21/2004
  著作権 : Adobe Systems Incorporated
 */
```

ドキュメントコメントの後に、クラスを宣言します。実装コメントは、宣言の直後に配置します。

メモ	当該クラスと直接関係のないコメントは含めないでください。たとえば、対応するパッケージを説明するようなコメントは書かないようにします。

ActionScript コードにコメントを付けるには、クラスの本体内でブロックコメント、単一行コメント、後続コメントを使用します。クラスファイル内でのコメントの使用方法の詳細については、658 ページの「クラスへのコメントの追加」を参照してください。

ActionScript のコーディング規則

変数の命名スキーム (646 ページの「命名規則」で説明)、コードのフォーマット (675 ページの「ActionScript シンタックスのフォーマット」で説明)、コーディング規則、ActionScript 2.0 コードの配置場所 (このセクションで説明) のいずれにおいても、プログラミングに関して最も重要な要素の 1 つは一貫性です。コードが体系化され、規則に従っていれば、コードのデバッグと保守の作業は大幅に単純化できます。

コーティング規則の詳細については、次に示す各トピックを参照してください。

- 659 ページの「ActionScript コードを 1 か所に配置する」
- 660 ページの「オブジェクトへのコードのアタッチ」
- 661 ページの「スコープの扱い」
- 664 ページの「クラスファイルの構成」
- 671 ページの「関数の使用について」

ActionScript コードを 1 か所に配置する

ActionScript 2.0 コードは、外部 ActionScript ファイルやタイムラインのフレーム 1(タイムラインに配置されたコードは " フレームスクリプト " と呼ばれる) など、可能な限り 1 か所にまとめてください。

フレームスクリプト内に ActionScript コードを配置する場合は、タイムラインの 1 番目または 2 番目のフレームで、タイムラインの 1 番目または 2 番目のレイヤーである *Actions* というレイヤーに ActionScript コードを設定します。場合によっては、ActionScript 用に 2 つのレイヤーを作成して機能を分けるのも適切な方法です。なお、Flash アプリケーションによっては、すべてのコードを 1 か所に配置できない場合もあります (特にスクリーンやビヘイビアを使用する場合)。

このような稀な例外はありますが、通常は、すべてのコードを同じ場所に配置できます。ActionScript コードを 1 か所に配置することには、次のようなメリットがあります。

- 複雑なソースファイルであっても、コードを簡単に見つけることができます。
- コードを簡単にデバッグできます。

FLA ファイルのデバッグで最も困難な作業の 1 つは、すべてのコードを見つけることです。すべてのコードを見つけた後で、FLA ファイルやコードの他の部分とどのように関係しているかを理解する必要があります。すべてのコードを 1 つのフレームに配置すれば集中管理によってデバッグがはるかに簡単になり、そのような困難が発生しにくくなります。コードをオブジェクトにアタッチする (つまり、コードを分散する) 場合の詳細については、660 ページの「オブジェクトへのコードのアタッチ」を参照してください。ビヘイビアと分散コードの詳細については、『Flash ユーザーガイド』を参照してください。

オブジェクトへのコードのアタッチ

簡単なアプリケーションやプロトタイプアプリケーションであっても、FLA ファイル内のオブジェクト (ボタンやムービークリップインスタンスなど) に ActionScript コードをアタッチしないようにする必要があります。オブジェクトにコードをアタッチするとは、ムービークリップ、コンポーネント、またはボタンインスタンスを選択し、ActionScript エディタ ([アクション] パネルまたはスクリプトウィンドウ) を開き、`on()` または `onClipEvent()` ハンドラ関数を使用して ActionScript コードを追加することです。

この方法は、次の理由から、使用しないことを強くお勧めします。

- オブジェクトにアタッチされた ActionScript コードは見つけるのが困難で、FLA ファイルの編集も困難です。
- オブジェクトにアタッチされた ActionScript コードはデバッグが困難です。
- タイムラインまたはクラス内に記述された ActionScript コードの方が簡潔で、作成が簡単です。
- オブジェクトにアタッチされた ActionScript コードの場合、コーディングスタイルの品質が低下します。
- オブジェクトにアタッチされた ActionScript コードの場合、学習者やこれを読む人は、追加のシンタックスや、多くの場合、質が悪く限定的な複数のコーディングスタイルを学ぶことを余儀なくされます。
- 通常、ユーザーはタイムラインでの関数の記述方法などを後で学習し直す必要があります。

一部には、オブジェクトにコードをアタッチする方が簡単に ActionScript を学べると考える Flash ユーザーもいます。また、簡単なコードを追加する場合や ActionScript について説明したり教えたりする場合にこの方が簡単であると考える人もいます。ただし、2 つのコーディングスタイル (オブジェクトに配置されたコードとフレームスクリプト) が混在していると、その差異が ActionScript を学ぶ開発者を混乱させる可能性もあるので、避ける必要があります。また、ユーザーはオブジェクトにアタッチするコードの記述方法を学習しても、同等のコードをフレームスクリプトとして配置する方法を後で学習し直す必要があります。このような理由から、学習の過程では全体をとおしてフレームスクリプトの記述方法を学習し、一貫性を保つことが大切です。

`myBtn` というボタンに ActionScript コードをアタッチする場合、次のようなコードになります。ただしこの方法は望ましくありません。

```
on (release) {
  // 何らかの処理をする。
}
```

同等の ActionScript コードをタイムラインに配置する場合は、次のようなコードになります。

```
// よいコード
myBtn.onRelease = function() {
  // 何らかの処理をする。
};
```

ActionScript シンタックスの詳細については、675 ページの「ActionScript シンタックスのフォーマット」を参照してください。

メモ	ビヘイビアやスクリーンを使用する場合は、オブジェクトにコードをアタッチする必要が生じることがあるため、適用される原則も異なります。詳細については、『Flash ユーザーガイド』を参照してください。

スコープの扱い

スコープとは、SWF ファイル内で変数の参照や使用が可能な範囲のことです。タイムライン上のスコープ、アプリケーション全体に渡るグローバルなスコープ、関数内のローカルなスコープなどがあります。通常、コードの記述中にスコープを参照するには、複数の方法があります。スコープを正しく使用すれば、ポータブルで再利用可能な ActionScript コードを作成することができ、新しいモジュールを作成するにつれてアプリケーションが破綻していくのを防ぐことができます。

global スコープと root スコープの違いを理解することは重要です。root は、ロードされた SWF ファイルごとに固有のスコープです。このグローバルスコープは、SWF ファイル内のすべてのタイムラインおよびスコープに適用されます。root タイムラインを参照するのではなく、相対アドレスを使用してください。そうすることでコードの再利用性とポータビリティを実現できます。アプリケーションでのスコープの扱い方については、次のセクションを参照してください。

57 ページの「変数とスコープについて」

85 ページの「スコープとターゲット化について」

234 ページの「クラスとスコープについて」

絶対ターゲット (_root) の回避

`_root` を使用せずにインスタンスをターゲットにする方法はいくつかあります。このセクションではその方法について説明します。ActionScript 2.0 で `_root` を使用することは避けてください。使用すると、他の SWF ファイルにロードされる SWF ファイルが正しく動作しない可能性があります。`_root` 識別子は、ロードする側のベース SWF ファイルをターゲットにします。`_root` の代わりに相対アドレスを使用する SWF ファイルではありません。この問題は、他のファイルにロードされる SWF ファイル (特に、コンポーネントやムービークリップ) におけるコードの移植性を低下させます。`_lockroot` を使用すれば問題の解決は可能ですが、`_lockroot` はやむを得ない場合のみ使用してください (たとえば、SWF ファイルをロードする必要があるのに FLA ファイルがない場合など)。`_lockroot` の使用方法の詳細については、662 ページの「_lockroot の使用」を参照してください。

ActionScript 2.0 コードの場所に応じて、`this`、`this._parent`、`_parent` のいずれかのキーワードを `_root` の代わりに使用できます。次の例は、相対アドレスを示します。

```
myClip.onRelease = function() {
  trace(this._parent.myButton._x);
};
```

関数のパラメータやローカル変数を除き、必ず変数にはスコープを設定してください。変数のスコープは、現在のパスがあればそれを基準に this などのプロパティを使用して相対アドレスで指定します。this プロパティの使用方法の詳細については、『ActionScript 2.0 リファレンスガイド』の「this プロパティ」を参照してください。

_lockroot の使用

_root の不適切な使用に関連してよく発生するスコープの問題を解決する 1 つの方法として、ターゲットコンテンツに対して _lockroot を使用できます。_lockroot を使用するとアプリケーションで発生する多くの問題を解決できますが、これは _root を使用したことに起因する問題の回避手段と考えてください。コンテンツを SWF ファイルまたはコンポーネントインスタンスにロードするときに問題が発生した場合は、コンテンツをロードするムービークリップに _lockroot を適用してみてください。たとえば、myClip というムービークリップにコンテンツをロードし、ロード完了後に動作を停止する場合、次のコードをタイムラインに配置すると有効な場合があります。

```
this._lockroot = true;
```

this キーワードの使用

this キーワードは、不要な場合でも省略せずにできるだけ接頭辞として指定してください。this キーワードが付いていれば、メソッドやプロパティが特定のクラスのメンバーであることがすぐわかります。たとえば、タイムラインの関数の場合は、ActionScript 2.0 コードを次の形式で記述します。

```
circleClip.onPress = function() {
  this.startDrag();
};
circleClip.onRelease = function() {
  this.stopDrag();
};
```

クラスの場合は次の形式で記述します。

```
class User {
  private var username:String;
  private var password:String;
  function User(username:String, password:String) {
    this.username = username;
    this.password = password;
  }
  public function get username():String {
    return this.username;
  }
  public function set username(username:String):Void {
    this.username = username;
  }
}
```

このような状況で this キーワードを一貫して使用すると、非常に読んで理解しやすい ActionScript 2.0 コードになります。

クラス内のスコープについて

ActionScript 2.0 のクラスへコードを移植するとき、場合によっては this キーワードの使い方を変更する必要があります。たとえば、コールバック関数を使用するクラスメソッド (LoadVars クラスの onLoad メソッドなど) では、this キーワードがクラスを指すのか LoadVars オブジェクトを指すのか判断しにくいことがあります。その場合は、次の例のようにして現在のクラスへのポインタを作成します。

```
class Product {
  private var m_products_xml:XML;
  // コンストラクタ
  // targetXmlStr は XML ファイルへのパスを含む
  function Product(targetXmlStr:String) {
    /* 現在のクラスへのローカル参照を作成する。
      XML の onLoad イベントハンドラ内を実行している場合でも、
      XML パケットだけでなく、現在のクラスも参照できる。*/
    var thisObj:Product = this;
    // ローカル変数を作成する。これは XML のロードに使用する。
    var prodXml:XML = new XML();
    prodXml.ignoreWhite = true;
    prodXml.onLoad = function(success:Boolean) {
      if (success) {
        /* XML を適切にロードおよび解析できた場合は、
          クラスの m_products_xml 変数に、解析した
          XML ドキュメントを設定し、init 関数を呼び出す。*/
        thisObj.m_products_xml = this;
        thisObj.init();
      } else {
        /* XML ファイルのロードでエラーが発生した。*/
        trace("error loading XML");
      }
    };
    // XML ドキュメントのロードを開始する。
    prodXml.load(targetXmlStr);
  }
  public function init():Void {
    // XML パケットを表示する。
    trace(this.m_products_xml);
  }
}
```

this キーワードは、onLoad ハンドラ内の private メンバー変数を参照することになるので、Product クラスではなく実際には prodXml インスタンスを表します。このため、onLoad ハンドラからクラスを直接参照するには、このようにローカルクラスファイルへのポインタを作成する必要があります。

クラスの詳細については、234 ページの「クラスとスコープについて」を参照してください。スコープの詳細については、661 ページの「スコープの扱い」を参照してください。

クラスファイルの構成

クラスは独立した ActionScript 2.0 ファイルとして作成され、コンパイル時に SWF ファイルに読み込まれます。

クラスは独立した ActionScript 2.0 ファイルとして作成され、アプリケーションのコンパイル時に SWF ファイルに読み込まれます。クラスファイルを作成する場合は、一定の方法論と順序を意識してコードを記述してください。以下のセクションでこの方法論を説明します。

クラスファイルを構成するための次の規則は、コードの効率とわかりやすさを向上させるために、クラスの各部分をどのような順序で構成したらよいかを示すものです。

クラスファイルを構成するには、次のエレメントを使用します。

1. ドキュメントコメント (コードの概要と、作成者情報およびバージョン情報) を追加します。
2. 必要に応じて、import ステートメントを追加します。
3. 次のようなクラス宣言、またはインターフェイス宣言を記述します。

   ```
   UserClass{...}
   ```

4. クラスまたはインターフェイスの実装に関する注意事項を説明するコメントを追加します。

   このコメントには、クラスまたはインターフェイス全体に関係する情報を記述します。
5. すべての静的変数を追加します。

   最初にパブリッククラス変数を書き、その後にプライベートクラス変数を続けます。
6. インスタンス変数を追加します。

   最初にパブリックメンバー変数を書き、その後にプライベートメンバー変数を続けます。
7. 次の例に示すようなコンストラクタステートメントを追加します。

   ```
   public function UserClass(username:String, password:String) {...}
   ```

8. メソッドを記述します。

   メソッドは、アクセシビリティやスコープではなく、機能ごとにグループ化してください。メソッドをこのように整理するとコードが読みやすくなり、理解しやすくなります。
9. getter/setter メソッドをクラスファイルに記述します。

クラスを作成するためのガイドライン

クラスファイルを作成するときは、次のガイドラインに留意してください。

- 同じ行に複数の宣言を記述しないようにします。

  たとえば、宣言は次の例のような形式で記述します。

  ```
  var prodSkuNum:Number;     // 製品 SKU ( 識別 ) 番号
  var prodQuantityNum:Number;  // 製品数量
  ```

  この例のように、宣言は同じ行に記述しないようにします。これらの宣言は、コードブロックの先頭に記述します。

- ローカル変数は、宣言時に初期化します。

  クラスのプロパティは、初期化子がコンパイル時定数の場合のみ、宣言内で初期化するようにしてください。

- 変数は、使用する前に宣言します。

  これはループの変数も同様です。

- 上位の宣言が見えなくなるようなローカル宣言を使用することは避けます。

  たとえば、次のように同じ変数を 2 度宣言することは望ましくありません。

```
var counterNum:Number = 0;
function myMethod() {
   for (var counterNum:Number = 0; counterNum<=4; counterNum++) {
      // ステートメント ;
   }
}
```

  このコードでは内側のブロックで同じ変数を宣言していますが、望ましいことではありません。

- 1 つのステートメントで複数の変数に同じ値を代入することは避けます。

  次の ActionScript コードで示すようにコードがわかりにくくなるので、この規則に従ってください。

```
playBtn.onRelease = playBtn.onRollOut = playsound;
```

  または

```
class User {
   private var m_username:String, m_password:String;
}
```

- メソッドやプロパティは、適切な理由がある場合のみパブリックにします。それ以外の場合、メソッドやプロパティはプライベートにしてください。

- getter/setter 関数を多用しすぎないようにします。

  getter/setter 関数はさまざまな目的に役立ちますが (207 ページの「getter メソッドと setter メソッドについて」を参照)、これらを多用しすぎることは、アプリケーションのアーキテクチャや編成に改善の余地があることを示しています。

- メンバー変数は、パブリックにする適切な理由がない限り、原則としてプライベートにします。

  設計の観点から見た場合、メンバー変数はプライベートにして、getter/setter 関数のグループを通じてのみアクセスできるようにするのが適切です。

クラスファイル内での this 接頭辞の使用

クラス内で同じクラスのメソッドやメンバー変数を使用するときは、this キーワードを接頭辞として指定してください。必須ではありませんが、接頭辞が付いていれば、プロパティやメソッドがそのクラスのメンバーであることがすぐわかります。接頭辞が付いていないと、プロパティやメソッドをスーパークラスのメンバーと区別できません。

静的な変数とメソッドに対してはクラス内でもクラス名接頭辞を使用できます。これで参照を修飾すれば、コードが理解しやすくなります。コーディング環境によっては、接頭辞によってコードを補完するコードヒントがトリガされます。次のコードは、静的プロパティに接頭辞としてクラス名を付ける例を示しています。

```
class Widget {
  public static var widgetCount:Number = 0;
  public function Widget() {
    Widget.widgetCount++;
  }
}
```

メモ	これらの接頭辞は必須でないため、不要と考える開発者もいます。しかし、接頭辞として this キーワードを追加するとコンテキストが限定されてコードがわかりやすくなるので、使用をお勧めします。

初期化について

変数の初期値については、次のクラスの例に示すようにデフォルト値を代入するか、undefined 値を許可するようにします。プロパティをインラインで初期化するときには、代入式の右辺に必ずコンパイル時定数を指定します。つまり、右辺の式では、設定や定義が実行時に行われる値を参照することはできません。コンパイル時定数には、ストリングリテラル、数値、ブール値、null、undefined に加えて、Array、Boolean、Number、Object、String というトップレベルクラスのコンストラクタ関数があります。このクラスでは、m_username および m_password の初期値を空のストリングに設定します。

```
class User {
  private var m_username:String = "";
  private var m_password:String = "";
  function User(username:String, password:String) {
    this.m_username = username;
    this.m_password = password;
  }
}
```

変数が不要になった場合は、削除するか null を設定します。変数の値を null に設定すると、パフォーマンスが向上することがあります。この処理は、一般に、" ガベージコレクション " と呼ばれています。変数を削除すると、不要なアセットが SWF ファイルから削除されるため、実行時のメモリ使用が最適化されます。変数を null に設定するよりも、変数を削除する方が適切です。パフォーマンスの詳細については、674 ページの「コードの最適化」を参照してください。

変数の命名の詳細については、650 ページの「変数の命名」を参照してください。オブジェクトの削除の詳細については、『ActionScript 2.0 リファレンスガイド』の「delete ステートメント」を参照してください。

ActionScript 2.0 でコードを初期化する最も簡単な方法の 1 つは、クラスを使用することです。1 つのインスタンスに関するすべての初期化処理は、そのクラスのコンストラクタ関数にカプセル化することも、別のメソッド内に抽象化することもできます。別のメソッドにした場合は、次のコードのように、変数を作成した後で明示的に呼び出します。

```
class Product {
  function Product() {
    var prodXml:XML = new XML();
    prodXml.ignoreWhite = true;
    prodXml.onLoad = function(success:Boolean) {
      if (success) {
        trace("loaded");
      } else {
        trace("error loading XML");
      }
    };
    prodXml.load("products.xml");
  }
}
```

次のコードはアプリケーション内の最初の関数呼び出しで、初期化を行う唯一のものです。XML をロードする FLA ファイルでは、フレーム 1 で次の ActionScript のようなコードを使用します。

```
if (init == undefined) {
  var prodXml:XML = new XML();
  prodXml.ignoreWhite = true;
  prodXml.onLoad = function(success:Boolean) {
    if (success) {
      trace("loaded");
    } else {
      trace("error loading XML");
    }
  };
  prodXml.load("products.xml");
  init = true;
}
```

trace ステートメントの使用

FLA ファイルのオーサリング時には、コードのデバッグ用にドキュメント内で trace ステートメントを使用できます。たとえば、trace ステートメントと for ループを使用することで、ストリング、配列、オブジェクトなどの変数値を [出力] パネルに表示できます。次に例を示します。

```
var dayArr:Array = ["sun", "mon", "tue", "wed", "thu", "fri", "sat"];
var numOfDays:Number = dayArr.length;
for (var i = 0; i<numOfDays; i++) {
  trace(i+": "+dayArr[i]);
}
```

[出力] パネルに次の情報が表示されます。

```
0: sun
1: mon
2: tue
3: wed
4: thu
5: fri
6: sat
```

trace ステートメントは、ActionScript 2.0 をデバッグする効果的な手段です。

SWF ファイルをパブリッシュする際に trace ステートメントを削除すれば、再生のパフォーマンスが若干向上します。SWF ファイルをパブリッシュする前に、[パブリッシュ設定] を開き、[Flash] タブの [Trace アクションを省略] を選択します。trace の使用の詳細については、『ActionScript 2.0 リファレンスガイド』の「trace 関数」を参照してください。

ActionScript コードのデバッグには、デバッガツールも役立ちます。

super 接頭辞の使用

親クラス内のメソッドを参照する場合は、メソッドに super という接頭辞を付け、他の開発者が見てもメソッドの起動先がわかるようにします。次の ActionScript 2.0 コードは、super を使用した正しいスコープ設定の例を示しています。

次の例では、2 つのクラスを作成します。Socks クラスで super キーワードを使用し、親クラス (Clothes) の関数を呼び出します。Socks クラスにも Clothes クラスにも、getColor() というメソッドがありますが、super を使用することで、基本クラスの特定のメソッドおよびプロパティを参照することができます。"Clothes.as" という名前で新しい AS ファイルを作成し、次のコードを入力します。

```
class Clothes {
   private var color:String;
   function Clothes(paramColor) {
      this.color = paramColor;
      trace("[Clothes] I am the constructor");
   }
```

```
   function getColor():String {
      trace("[Clothes] I am getColor");
      return this.color;
   }
   function setColor(paramColor:String):Void {
      this.color = paramColor;
      trace("[Clothes] I am setColor");
   }
}
```

次に、Clothes クラスを拡張する Socks という新しいクラスを作成します。

```
class Socks extends Clothes {
   private var color:String;
   function Socks(paramColor:String) {
      this.color = paramColor;
      trace("[Socks] I am the constructor");
   }
   function getColor():String {
      trace("[Socks] I am getColor");
      return super.getColor();
   }
   function setColor(paramColor:String):Void {
      this.color = paramColor;
      trace("[Socks] I am setColor");
   }
}
```

次に、新しい AS ファイルまたは FLA ファイルを作成し、そのドキュメントに次の ActionScript を入力します。

```
import Socks;
var mySock:Socks = new Socks("maroon");
trace(" -> "+mySock.getColor());
mySock.setColor("Orange");
trace(" -> "+mySock.getColor());
```

次の結果が [出力] パネルに表示されます。

```
[Clothes] I am the constructor
[Socks] I am the constructor
[Socks] I am getColor
[Clothes] I am getColor
-> maroon
[Socks] I am setColor
[Socks] I am getColor
[Clothes] I am getColor
-> Orange
```

Socks クラスの getColor() メソッドに super キーワードを指定し忘れると、getColor() メソッドは自分自身を繰り返し呼び出すため、無限反復の問題によってスクリプトが失敗します。super キーワードを使用しなかった場合、[出力] パネルに次のエラーが表示されます。

```
[Socks] I am getColor
[Socks] I am getColor
...
[Socks] I am getColor
256 levels of recursion were exceeded in one action list.
This is probably an infinite loop.
Further execution of actions has been disabled in this SWF file.
```

with ステートメントの回避

ActionScript 2.0 の学習者にとって、with ステートメントの使い方は非常にわかりにくい概念の 1 つです。with ステートメントを使用している次のコードを見てみましょう。

```
this.attachMovie("circleClip", "circle1Clip", 1);
with (circle1Clip) {
  _x = 20;
  _y = Math.round(Math.random()*20);
  _alpha = 15;
  createTextField("labelTxt", 100, 0, 20, 100, 22);
  labelTxt.text = "Circle 1";
  someVariable = true;
}
```

このコードでは、ライブラリからムービークリップインスタンスをアタッチし、with ステートメントを使用してプロパティを変更しています。変数のスコープを指定しないと、どこでプロパティを設定しているかわからなくなる場合があるので、難解なコードになります。上のコードで、someVariable は circle1Clip ムービークリップに対して設定されるように見えますが、実際には SWF ファイルのメインタイムライン内に設定されます。

with ステートメントを使用するのではなく、変数のスコープを明示的に指定した方が、コード内で何が起きているかを簡単に追うことができます。次の例は多少長くなっていますが、より適切な、変数スコープを指定した ActionScript の例です。

```
this.attachMovie("circleClip", "circle1Clip", 1);
circle1Clip._x = 20;
circle1Clip._y = Math.round(Math.random()*20);
circle1Clip._alpha = 15;
circle1Clip.createTextField("labelTxt", 100, 0, 20, 100, 22);
circle1Clip.labelTxt.text = "Circle 1";
circle1Clip.someVariable = true;
```

このルールの例外として、シェイプを描画するために描画 API を使用する場合に、描画 API の機能上、同じメソッド (lineTo や curveTo など) に対して類似の呼び出しを何度も実行することがあります。たとえば、単純な矩形を描画する場合でも、次のコードに示すように、lineTo メソッドを別々に 4 回呼び出す必要があります。

```
this.createEmptyMovieClip("rectangleClip", 1);
with (rectangleClip) {
  lineStyle(2, 0x000000, 100);
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(300, 0);
  lineTo(300, 200);
  lineTo(0, 200);
  lineTo(0, 0);
  endFill();
}
```

このような場合は、個々の lineTo メソッドや curveTo メソッドをすべて完全修飾インスタンス名で記述すると、コードの外見がすぐに混み合ってしまい、解読やデバッグが困難になります。

関数の使用について

コードは、できるだけ再利用してください。コードを再利用する基本は、毎回異なるコードを作成せずに、同じ関数を何度も呼び出すことです。汎用的なコードを関数にすれば、SWF ファイル内で、同じブロックのコードを複数の多少異なる目的に使用できます。コードを再利用することで、効率のよいアプリケーションを作成できると共に、ActionScript コードの記述量を減らすことができ、結果として開発期間の短縮につながります。関数は、タイムライン上およびクラスファイル内に作成する方法と、コードを中心とするコンポーネントの ActionScript として作成する方法があります。これらの関数はさまざまな方法で再利用できます。

ActionScript 2.0 を使用している場合は、タイムライン上に関数を記述することは避けてください。ActionScript 2.0 では、次のように関数をできるだけクラスファイル内に配置してください。

```
class Circle {
public function area(radius:Number):Number {
  return (Math.PI*Math.pow(radius, 2));
}
public function perimeter(radius:Number):Number {
  return (2 * Math.PI * radius);
}
public function diameter(radius:Number):Number {
  return (radius * 2);
}
}
```

関数を作成するときは、次のシンタックスを使用します。

```
function myCircle(radius:Number):Number {
  //...
}
```

次のシンタックスは、読みにくいので使用しないでください。

```
myCircle = function(radius:Number):Number {
   //...
}
```

次の例では、関数をクラスファイル内に配置しています。ActionScript 2.0 を選択する場合、こうすることでコードの再利用性を最大化できるので、この方法がベストプラクティスだといえます。関数を他のアプリケーションで再利用する場合には、コードを最初から書き直す代わりに既存のクラスを読み込むか、新しいアプリケーション内に関数を複写することができます。

```
class mx.site.Utils {
   static function randomRange(min:Number, max:Number):Number {
      if (min>max) {
         var temp:Number = min;
         min = max;
         max = temp;
      }
      return (Math.floor(Math.random()*(max-min+1))+min);
   }
   static function arrayMin(numArr:Array):Number {
      if (numArr.length == 0) {
         return Number.NaN;
      }
      numArr.sort(Array.NUMERIC | Array.DESCENDING);
      var min:Number = Number(numArr.pop());
      return min;
   }
   static function arrayMax(numArr:Array):Number {
      if (numArr.length == 0) {
         return undefined;
      }
      numArr.sort(Array.NUMERIC);
      var max:Number = Number(numArr.pop());
      return max;
   }
}
```

これらの関数を使用するには、たとえば、次のような ActionScript を FLA ファイルに追加します。

```
import mx.site.Utils;
var randomMonth:Number = Utils.randomRange(0, 11);
var min:Number = Utils.arrayMin([3, 3, 5, 34, 2, 1, 1, -3]);
var max:Number = Utils.arrayMax([3, 3, 5, 34, 2, 1, 1, -3]);
trace("month: "+randomMonth);
trace("min: "+min);
trace("max: "+max);
```

コードの繰り返し実行の停止について

onEnterFrame イベントハンドラを使用すると、SWF ファイルのフレームレートでコードを繰り返し実行できるので便利です。しかし、パフォーマンスに影響を与えないよう、Flash ファイル内で繰り返し実行されるコードはできるだけ減らしてください。たとえば、再生ヘッドがフレームに入るたびに繰り返されるようなコードは CPU に負荷をかけます。この動作は、SWF ファイルを再生するコンピュータでパフォーマンスの問題を引き起こす可能性があります。SWF ファイル内で、何らかのアニメーションや繰り返しのために onEnterFrame イベントハンドラを使用する場合は、使用を終了したとき onEnterFrame ハンドラを削除してください。次の ActionScript 2.0 コードでは、onEnterFrame イベントハンドラを削除することで繰り返しを停止しています。

```
circleClip.onEnterFrame = function() {
  circleClip._alpha -= 5;
  if (circleClip._alpha<=0) {
    circleClip.unloadMovie();
    delete this.onEnterFrame;
    trace("deleted onEnterFrame");
  }
};
```

同様に、setInterval も多用しないでください。また、SWF ファイルの処理に使われる CPU 時間を減らすため、使用を終了するときに必ず間隔をクリアしてください。

ActionScript と Flash Player の最適化

ActionScript 2.0 を使用した SWF ファイルを、[パブリッシュ設定] を [Flash Player 6] と [ActionScript 1.0] に設定してコンパイルすると、コードは ActionScript 2.0 クラスを使用しない限りにおいては動作します。大文字と小文字はコードでは区別されず、Flash Player によってのみ区別されます。したがって、[パブリッシュ設定] に [Flash Player 7] 以降と [ActionScript 1.0] を設定して SWF ファイルをコンパイルすると、大文字と小文字が区別されます。

[パブリッシュ設定] に [ActionScript 2.0] を設定して Flash Player 7 以降を対象にコンパイルする場合は、コンパイル時に型注釈 (厳密な型指定) が適用されます。

アプリケーションをパブリッシュすると、ActionScript 2.0 は ActionScript 1.0 のバイトコードにコンパイルされます。そのため、ActionScript 2.0 を使用しながら、Flash Player 6 以降を対象にすることができます。

アプリケーションの最適化の詳細については、「コードの最適化」を参照してください。

コードの最適化

コードを最適化するときは、次のガイドラインに留意してください。

- ループ内から関数を繰り返し呼び出さないようにします。

  小さな関数なら、その内容をループに含めた方が効率的です。

- できるだけネイティブの関数を使用します。

  ネイティブの関数はユーザー定義の関数よりも高速です。

- オブジェクト型を不必要に多用しないようにします。

  パフォーマンスを向上させるには正確な型注釈を使用してください。オブジェクト型を使用するのは、他に現実的な代替手段がない場合だけにしてください。

- `eval()` 関数または配列アクセス演算子を使わないようにします。

  1 回だけローカル参照を設定する方が、多くの場合は効率的で望ましい方法です。

- ループの前に `Array.length` を変数に割り当てます。

  `myArr.length` 自体を使用するよりも、ループの前に `Array.length` を変数に割り当て、ループ条件として使用します。次に例を示します。

  ```
  var fontArr:Array = TextField.getFontList();
  var arrayLen:Number = fontArr.length;
  for (var i:Number = 0; i < arrayLen; i++) {
    trace(fontArr[i]);
  }
  ```

  次のようにはしません。

  ```
  var fontArr:Array = TextField.getFontList();
  for (var i:Number = 0; i < fontArr.length; i++) {
    trace(fontArr[i]);
  }
  ```

- ループや繰り返しアクションについて重点的に最適化します。

  Flash Player ではループの処理に多くの時間がかかります (`setInterval()` 関数を使用するループなど)。

- 変数の宣言時に `var` キーワードを追加します。
- ローカル変数で十分な場合は、クラス変数やグローバル変数を使用しないようにします。

ActionScript シンタックスのフォーマット

ActionScript 2.0 コードを標準化された方法でフォーマットすることは、保守しやすいコードを記述するために不可欠なだけでなく、それによってコードが別の開発者にとって理解しやすく、変更も簡単なものとなります。たとえば、インデントやコメントが一切なく、命名規則やフォーマットにも一貫性がないような FLA ファイルでは、ロジックを追うことは非常に困難です。コードのブロック (ループや if ステートメントなど) をインデントすることで、コードが読みやすくなり、デバッグもしやすくなります。

コードのフォーマットの詳細については、次に示す各トピックを参照してください。

- 675 ページの「一般的なフォーマットのガイドライン」
- 678 ページの「条件ステートメントの書き方」
- 679 ページの「複合ステートメントの書き方」
- 680 ページの「for ステートメントの書き方」
- 680 ページの「while および do..while ステートメントの書き方」
- 681 ページの「return ステートメントの書き方」
- 681 ページの「switch ステートメントの書き方」
- 682 ページの「try..catch および try..catch..finally ステートメントの記述」
- 682 ページの「リスナーシンタックスの使用について」

一般的なフォーマットのガイドライン

スペース、改行、およびタブインデントを使用してコードに空白を追加すると、コードが読みやすくなります。空白を使用すると、コードの階層構造を示すことができ、読みやすさが向上します。読みやすく理解しやすい ActionScript 2.0 コードにすることは、学習者だけでなく、複雑なプロジェクトで作業している経験豊富なユーザーにとっても重要です。コードの読みやすさは、ActionScript コードをデバッグする際にも重要です。コードが正しくフォーマットされ、適切な間隔で配置されていれば、エラーを簡単に突き止めることができます。

ActionScript 2.0 コードでは、同じ内容にも複数の記述方法があります。違いが生じるのは、ActionScript エディタ ([アクション] パネルまたはスクリプトウィンドウ) で複数の行にわたるシンタックスの記述方法に関する部分です。たとえば、ブラケット ({ }) や丸かっこ [( )] については何とおりかの配置方法があります。

ActionScript コードの読みやすさを向上するためのポイントは次のとおりです。

- ActionScript の段落 (モジュール) 間に 1 行分の空白行を空けます。

  ActionScript コードの段落とは、論理的に関係しているコードの集まりです。段落間に 1 行分の空白行を空けることで、ActionScript コードを読むユーザーはロジックを理解しやすくなります。

- コードの階層構造を示すために、コード内では一貫したインデントを使用します。

  ActionScript コード全体で同一のインデントスタイルを使用し、中カッコ ({ }) の位置を適切に揃えるようにしてください。中カッコの位置が揃うと、コードが読みやすくなります。正しい ActionScript シンタックスを使用していれば、Enter (Windows) または Return (Macintosh) を押すと自動的にコードの正しいインデントが設定されます。また、ActionScript エディタ ([アクション] パネルまたはスクリプトウィンドウ) で [自動フォーマット] ボタンをクリックすると、シンタックスが正しければ ActionScript コードのインデントが調整されます。

- 改行を使用して複雑なステートメントを読みやすくします。

  条件ステートメントなど、ステートメントによっては、いくつかの形式でフォーマットできます。場合によっては、1 行ではなく複数行にわたってステートメントをフォーマットする方が読みやすくなります。

- キーワードと後続のカッコ [( )] の間にスペースを 1 つ入れます。

  その例を次の ActionScript コードに示します。

```
do {
   // 処理
} while (condition);
```

- メソッド名と後続の括弧との間には、スペースを入れないようにします。

  その例を次の ActionScript コードに示します。

```
function checkLogin():Boolean {
   // ステートメント ;
}
checkLogin();
```

  または

```
printSize("size is " + foo + "\n");
```

- 引数リストでは、カンマの後ろにスペースを 1 つ入れます。

  次の例に示すように、カンマの後ろにスペースを入れることで、メソッドの呼び出しとキーワードを簡単に区別できます。

```
function addItems(item1:Number, item2:Number):Number {
   return (item1 + item2);
}
var sum:Number = addItems(1, 3);
```

- 個々の演算子とオペランドの間はスペースで区切ります。

  次の例に示すように、スペースを入れることで、メソッドの呼び出しとキーワードを簡単に区別できます。

```
// よい例
var sum:Number = 7 + 3;
// よくない例
var sum:Number=7+3;
```

  このガイドラインの例外としては、ドット (.) 演算子があります。

- 単項演算子とオペランドの間には、スペースを入れないようにします。

  たとえば、インクリメント (++) やデクリメント (--) は次の例のように記述します。

```
while (d++ = s++)
-2, -1, 0
```

- 左カッコの後と右カッコの前にはスペースを入れないようにします。

  その例を次の ActionScript コードに示します。

```
// よくない例
( "size is " + foo + "\n" );
// よい例
("size is " + foo + "\n");
```

- ActionScript コードを読みやすくするには、1行に1つのステートメントのみを書くようにします。

  その例を次の ActionScript コードに示します。

```
theNum++;        // 正解
theOtherNum++;   // 正解
aNum++; anOtherNum++;  // これは誤り
```

- 代入ステートメントの埋め込みは行わないようにします。

  SWF ファイルで実行時のパフォーマンスを改善するためにステートメントの埋め込みが行われることがあります。しかし、パフォーマンスの改善効果以上にコードの読みやすさが大きく低下し、デバッグが困難になります。その例を次の ActionScript コードに示します。ただし、実際のコードでは、単一文字の名前は避けてください。

```
var myNum:Number = (a = b + c) + d;
```

- 変数は別個のステートメントとして代入します。

  その例を次の ActionScript コードに示します。ただし、実際のコードでは、単一文字の名前は避けてください。

```
var a:Number = b + c;
var myNum:Number = a + d;
```

- 演算子の前で改行します。
- カンマの後ろで改行します。
- 次の行を、前の行の式の先頭に揃えます。

メモ	自動インデントおよびインデント設定を制御するには、[編集]-[環境設定] (Windows) または [Flash]-[環境設定] (Macintosh) を選択し、[ActionScript] タブを選択します。

条件ステートメントの書き方

条件ステートメントを記述するときは、次のガイドラインに従ってください。

- if、else..if、および if..else ステートメントでは、個々の条件を別の行に分けて記述します。
- if ステートメントでは中カッコ ({}) を使用します。
- 次の例に示すように、中カッコをフォーマットします。

```
//if ステートメント
if (condition) {
  // ステートメント
}

// if..else ステートメント
if (condition) {
  // ステートメント
} else {
  // ステートメント
}

// else..if ステートメント
if (condition) {
  // ステートメント
} else if (condition) {
  // ステートメント
} else {
  // ステートメント
}
```

複雑な条件を書くときは、カッコ [()] を使用して条件をグループ化することをお勧めします。カッコを使用しないと、自分やこの ActionScript 2.0 コードを使用する他の人が演算子の優先順位を間違えることも考えられます。

たとえば、次のコードは条件を囲むカッコを使用していません。

```
if (fruit == apple && veggie == leek) {}
```

次のように条件をカッコで囲む方が適切です。

```
if ((fruit == apple) && (veggie == leek)) {}
```

ブール値を返す条件ステートメントは、2 つの方法で書くことができます。2 番目の例に示す方法をお勧めします。

```
if (cartArr.length>0) {
  return true;
} else {
  return false;
}
```

この例を前の例と比較してください。

```
// よりよい例
return (cartArr.length > 0);
```

2 番目の例の方が簡潔で、評価する式も少なくて済みます。また、読みやすさが向上し、理解しやすくなります。

次の例では、変数 y が 0 より大きければ x/y を返し、そうでなければ値 0 を返します。

```
return ((y > 0) ? x/y : 0);
```

このコードは、次の形式で書くこともできます。この例の方法をお勧めします。

```
if (y>0) {
  return x/y;
} else {
  return 0;
}
```

最初の例の短縮された if ステートメントシンタックスを条件演算子 (?:) と呼びます。これを使用すると簡単な if..else ステートメントを 1 行のコードに書き換えることができます。このように短縮されたシンタックスはわかりにくくなります。

条件演算子を使用する場合は、先頭の条件 (疑問符 [?] の前の条件) をカッコで囲み、コードを読みやすくしてください。前のコード例ではそのようになっています。

複合ステートメントの書き方

複合ステートメントには、中カッコ ({}) で囲んだ複数のステートメントが付加されます。この中カッコ内のステートメントは、複合ステートメント自体よりもインデントを下げて記述します。その例を次の ActionScript コードに示します。

```
if (a == b) {
  // このコードはインデントされる
  trace("a == b");
}
```

制御構造 (if..else または for) の要素となるステートメントは、たとえ 1 つだけでも中括弧で囲むようにしてください。次の例は、よくない書き方です。

```
// よくない例
if (numUsers == 0)
  trace("no users found.");
```

このコードは間違いではありませんが、ステートメントを中カッコで囲んでいないので、推奨される書き方とは言えません。この trace ステートメントの後に別のステートメントを追加すると、そのコードは、numUsers 変数が 0 かどうかに関係なく実行されます。

```
// よくない例
var numUsers:Number = 5;
if (numUsers == 0)
  trace("no users found.");
  trace("I will execute");
```

numUsers 変数の値にかかわらずコードが実行されることによって、予期しない結果となる可能性があります。そこで、次の例に示すように中カッコを追加します。

```
var numUsers:Number = 0;
if (numUsers == 0) {
  trace("no users found");
}
```

条件を記述する場合は、次のように、コードに無駄な ==true を追加しないようにしてください。

```
if (something == true) {
  // ステートメント
}
```

false と比較する場合は、if (something==false) または if(!something) を使用できます。

for ステートメントの書き方

for ステートメントは次の形式で記述できます。

```
for (init; condition; update) {
  // ステートメント
}
```

次の構造は for ステートメントの使用例です。

```
var i:Number;
for (var i = 0; i<4; i++) {
  myClip.duplicateMovieClip("newClip" + i + "Clip", i + 10, {_x:i*100, _y:0});
}
```

なお、for ステートメント内では各式の後ろに、スペースを 1 つずつ入れるようにしてください。

while および do..while ステートメントの書き方

while ステートメントは次の形式で記述できます。

```
while (condition) {
  // ステートメント
}
```

do-while ステートメントは次の形式で記述できます。

```
do {
  // ステートメント
} while (condition);
```

return ステートメントの書き方

値を返す return ステートメントでは、丸カッコ [()] を使用しないでください。カッコは、return ステートメントでどのような値を返すのか明確にする場合に限り使用してください。次の ActionScript コードの 3 行目がその例です。

```
return;
return myCar.paintColor;
// 戻り値を明示するためにカッコを使用
return ((paintColor)? paintColor: defaultColor);
```

switch ステートメントの書き方

- switch ステートメントは必ず default ケースを持ちます。

  default ケースを書く位置は switch ステートメントの最後にします。default ケースには、別のケースを追加した場合のフォールスルーエラーを回避する break ステートメントを含める必要があります。

- ケースに break ステートメントがない場合は、" フォールスルー " と呼ばれる状態が発生します (次のコードの例の case A を参照)。

  次の例の case A に示すように、通常なら break ステートメントがある位置にコメントを入れてください。この例では、条件が case A に等しい場合、case A と case B の両方のステートメントが実行されます。

switch ステートメントは次の形式で記述できます。

```
switch (condition) {
case A :
   // ステートメント
   // フォールスルー
case B :
   // ステートメント
   break;
case Z :
   // ステートメント
   break;
default :
   // ステートメント
   break;
}
```

try..catch および try..catch..finally ステートメントの記述

try..catch および try..catch..finally ステートメントは、次のフォーマットを使用して記述します。

```
var myErr:Error;
// try..catch
try {
  // ステートメント
} catch (myErr) {
  // ステートメント
}

// try..catch..finally
try {
  // ステートメント
} catch (myErr) {
  // ステートメント
} finally {
  // ステートメント
}
```

リスナーシンタックスの使用について

Flash 8 以降ではイベントのリスナーを何とおりかの方法で書くことができます。以下の例は一般的な方法を示しています。最初の例は、Loader コンポーネントを使用するリスナーシンタックスです。この例では、コンテンツが SWF ファイルにロードされます。コンテンツのロード開始時に progress イベントが開始され、ロード終了時に complete が示されます。

```
var boxLdr:mx.controls.Loader;
var ldrListener:Object = new Object();
ldrListener.progress = function(evt:Object) {
  trace("loader loading:" + Math.round(evt.target.percentLoaded) + "%");
};
ldrListener.complete = function(evt:Object) {
  trace("loader complete:" + evt.target._name);
};
boxLdr.addEventListener("progress", ldrListener);
boxLdr.addEventListener("complete", ldrListener);
boxLdr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

次の例は handleEvent メソッドを使用する方法で、このセクションの最初の例を少し変形したものですが、この方法はやや面倒です。キャッチされたイベントの種類を調べるために、if..else ステートメントの連続や switch ステートメントを使用する必要があるので、この方法はお勧めしません。

```
var boxLdr:mx.controls.Loader;
var ldrListener:Object = new Object();

ldrListener.handleEvent = function(evt:Object) {
  switch (evt.type) {
  case "progress" :
    trace("loader loading:" + Math.round(evt.target.percentLoaded) + "%");
    break;
  case "complete" :
    trace("loader complete:" + evt.target._name);
    break;
  }
};
boxLdr.addEventListener("progress", ldrListener);
boxLdr.addEventListener("complete", ldrListener);
boxLdr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

付録 A

エラーメッセージ

A

Adobe CS3 Professional では、ActionScript 2.0 でパブリッシュする場合 (デフォルト) に、コンパイル時のエラー報告機能が提供されます。次の表に、Flash コンパイラが生成できるエラーメッセージの一覧を示します。

エラー番号	メッセージテキスト
1093	クラス名が必要です。
1094	基本クラス名が、'extends' キーワードの後に必要です。
1095	メンバー属性が正しく使用されていません。
1096	同じメンバー名を繰り返し使用することはできません。
1097	すべてのメンバー機能には名前が必要です。
1099	このステートメントは、クラス定義で許可されていません。
1100	クラスまたはインターフェイスが既にこの名前で定義されています。
1101	タイプが一致しません。
1102	'<ClassName>' というの名前のクラスはありません。
1103	'<propertyName>' というの名前のプロパティはありません。
1104	関数以外で関数の呼び出しが実行されました。
1105	代入ステートメントでタイプが一致しません。[lhs-type] が見つかりましたが、[rhs-type] が必要です。
1106	メンバーがプライベートで、アクセスできません。
1107	変数宣言はインターフェイスでは許可されません。
1108	イベント宣言はインターフェイスでは許可されません。
1109	getter/setter 宣言はインターフェイスでは許可されません。
1110	プライベートメンバーはインターフェイスでは許可されません。
1111	関数本体はインターフェイスでは許可されません。
1112	クラスはそれ自体を拡張できません。

エラー番号	メッセージテキスト
1113	インターフェイスはそれ自体を拡張できません。
1114	この名前で定義されているインターフェイスはありません。
1115	クラスはインターフェイスを拡張できません。
1116	インターフェイスはクラスを拡張できません。
1117	インターフェイス名が、'implements' キーワードの後に必要です。
1118	クラスはクラスを実装できません。インターフェイスのみ実装できます。
1119	クラスはメソッド 'methodName' をインターフェイス 'interfaceName' から実装する必要があります。
1120	インターフェイスメソッドの実装は、プロパティでなくメソッドである必要があります。
1121	クラスは同じインターフェイスを複数回拡張できません。
1122	インターフェイスメソッドの実装が、その定義と一致しません。
1123	このコンストラクトは ActionScript 1.0 でのみ使用できます。
1124	このコンストラクトは ActionScript 2.0 でのみ使用できます。
1125	静的メンバーはインターフェイスでは許可されません。
1126	返された式は、関数の返されたタイプと一致する必要があります。
1127	return ステートメントがこの関数で必要です。
1128	属性はクラスの外で使用されます。
1129	戻り値タイプ Void を持つ関数は、値を返すことができません。
1130	'extends' 節は 'implements' 節の前に出現する必要があります。
1131	タイプ識別子は ':' の後に必要です。
1132	インターフェイスは、'implements' ではなく 'extends' キーワードを使用する必要があります。
1133	クラスは複数のクラスを拡張できません。
1134	インターフェイスは複数のインターフェイスを拡張できません。
1135	'‹methodName›' という名前のメソッドはありません。
1136	このステートメントは、インターフェイス定義で許可されていません。
1137	設定関数には、パラメータが 1 つだけ必要です。
1138	get 関数にはパラメータが不要です。
1139	クラスは、外部 ActionScript 2.0 クラスのスクリプトでのみ定義される必要があります。

エラー番号	メッセージテキスト
1140	ActionScript 2.0 のクラスのスクリプトは、クラスまたはインターフェイスコンストラクトのみを定義することができます。
1141	このクラスの名前 '‹A.B.C›' は、ロードされた別のクラスの名前 '‹A.B›' とコンフリクトします。 クラス名の一部が既存クラスのフルネームとコンフリクトしているのが原因で ActionScript 2.0 コンパイラがクラスをコンパイルできない場合に、このエラーが発生します。たとえば、`mx.com` クラスがコンパイル済みクラスである場合、クラス `mx.com.util` をコンパイルするとエラー 1141 を生成します。
1142	クラスまたはインターフェイス '‹ クラス名またはインターフェイス名 ›' をロードできませんでした。
1143	インターフェイスは、外部 ActionScript 2.0 クラスのスクリプトでのみ定義される必要があります。
1144	静的関数の中ではインスタンス変数にアクセスできない
1145	クラスおよびインターフェイス定義はネストできません。
1146	参照されているプロパティには、静的属性がありません。
1147	super の呼び出しが superconstructor に一致しません。
1148	パブリック属性のみが、インターフェイスメソッドで許可されます。
1149	import キーワードをディレクティブとして使用できません。
1150	このアクションを使用するにはムービーを Flash 7 形式で書き出す必要があります。
1151	この式を使用するにはムービーを Flash 7 形式で書き出す必要があります。
1152	この例外節は適切に配置されていません。
1153	クラスにはコンストラクタが 1 つだけ必要です。
1154	コンストラクタは値を返せません。
1155	コンストラクタは戻り値タイプを指定できません。
1156	変数を Void タイプにすることはできません。
1157	関数のパラメータを Void タイプにすることはできません。
1158	静的メンバーには、クラスを通してのみ直接アクセスできます。
1159	複数の実装されたインターフェイスには、異なるタイプを持つ同じメソッドがあります。
1160	この名前で定義されているクラスまたはインターフェイスが既にあります。
1161	クラス、インターフェイス、およびビルトインタイプは削除できません。
1162	この名前のクラスはありません。

エラー番号	メッセージテキスト
1163	キーワード '‹keyword›' は ActionScript 2.0 で予約されているため、ここでは使用できません。
1164	カスタム属性の定義が終了していません。
1165	ActionScript 2.0 の .as ファイル 1 つにつき、クラスまたはインターフェイスを 1 つだけ定義できます。
1166	コンパイルされたクラス '‹A.b›' が読み込まれたクラス '‹A.B›' に一致しません。クラス名が、読み込まれたクラスと大文字小文字が異なって綴られている場合、このエラーが発生します。たとえば、`import mx.Com` ステートメントが "util.as" ファイルにある場合、クラス `mx.com.util` をコンパイルするとエラー 1166 を生成します。
1167	クラス名を入力してください。
1168	入力されたクラス名にはシンタックスエラーがあります。
1169	入力されたインターフェイス名にはシンタックスエラーがあります。
1170	入力された基本クラス名にはシンタックスエラーがあります。
1171	入力された基本インターフェイス名にはシンタックスエラーがあります。
1172	インターフェイス名を入力してください。
1173	クラス名またはインターフェイス名を入力してください。
1174	入力されたクラス名またはインターフェイス名には、シンタックスエラーがあります。
1175	このスコープから 'variable' にアクセスできません。
1176	複数の 'get/set/private/public/static' 属性が見つかりました。
1177	クラス属性が正しく使用されていません。
1178	静的変数を初期化するために、インスタンス変数と関数を使用できません。
1179	ランタイム循環が、‹ ユーザー定義クラスの一覧 › のクラス間で見つかりました。このランタイムエラーは、カスタムクラスの相互参照が誤っていることを示しています。
1180	現在ターゲットにしている Flash Player は、デバッグをサポートしません。
1181	現在ターゲットにしている Flash Player は、releaseOutside イベントをサポートしません。
1182	現在ターゲットにしている Flash Player は、dragOver イベントをサポートしません。
1183	現在ターゲットにしている Flash Player は、dragOut イベントをサポートしません。
1184	現在ターゲットにしている Flash Player は、ドラッグアクションをサポートしません。

エラー番号	メッセージテキスト
1185	現在ターゲットにしている Flash Player は、loadMovie アクションをサポートしません。
1186	現在ターゲットにしている Flash Player は、getURL アクションをサポートしません。
1187	現在ターゲットにしている Flash Player は、FSCommand アクションをサポートしません。
1188	読み込みステートメントは、クラスまたはインターフェイス定義内では許可されません。
1189	クラス '‹A.B›' を読み込めません。そのリーフ名が、定義済みのクラス '‹C.B›' で既に解決されています。 たとえば、`import mx.Com` ステートメントが "util.as" ファイルにある場合、クラス `util` をコンパイルするとエラー 1189 を生成します。
1190	クラス '‹A.B›' を読み込めません。そのリーフ名が、読み込み済みのクラス '‹C.B›' で既に解決されています。 たとえば、`import mx.util` ステートメントも "AS" ファイルにある場合、クラス `import jv.util` をコンパイルするとエラー 1190 を生成します。
1191	クラスのインスタンス変数は、コンパイル時の定数式にのみ初期化できます。
1192	クラスメンバー関数は、スーパークラスのコンストラクタ関数と同じ名前をもつことができません。
1193	このクラスの名前 '‹ClassName›' は、ロードされた別のクラスの名前とコンフリクトします。
1194	superconstructor をコンストラクタ本体で最初に呼び出す必要があります。
1195	識別子 '‹className›' は、実行時、ビルトインオブジェクト '‹ClassName›' に解決されません。
1196	クラス '‹A.B.ClassName›' を相対パスが '‹A.B›' のファイル内で定義する必要があります。
1197	ワイルドカード文字 "*" がクラス名 '‹ClassName›' で誤って使用されています。
1198	メンバー関数 '‹classname›' は、定義されたクラス '‹ClassName›' の名前と大文字小文字が異なり、実行時にクラスコンストラクタとして扱われません。
1199	for-in ループに許可されるタイプはストリングだけです。
1200	setter 関数は値を返すことができません。
1201	コンストラクタ関数に許可される属性は public と private だけです。
1202	ActionScript 2.0 の型チェックに必要なファイル 'toplevel.as' が見つかりませんでした。ディレクトリ '$(LocalData)/Classes' が ActionScript 環境設定のグローバルクラスパスにリストされていることを確認してください。

エラー番号	メッセージテキスト
1203	‹spanStart› と ‹spanEnd› の間のブランチが 32K の範囲を超えています。
1204	パッケージ '‹PackageName›' には '‹packageName›' という名前のクラスまたはパッケージはありません。
1205	現在ターゲットにしている Flash Player は、FSCommand2 アクションをサポートしません。
1206	メンバー関数 '‹functionName›' が 32K より大きくなっています。
1207	行 ‹lineNumber› の匿名関数が 32K の範囲を超えています。
1208	行 ‹lineNumber› のコードが 32K の範囲を超えています。
1210	パッケージ名 '‹PackageName›' をさらにメソッド名として使用することはできません。
1211	パッケージ名 '‹PackageName›' をさらにプロパティ名として使用することはできません。
1212	クラス '‹ClassName›' の ASO ファイルを作成できませんでした。ASO ファイル名 '‹ClassName.aso›' が 255 文字未満になるように、完全修飾されたクラス名が短いことを確認してください。
1213	このタイプの引用符は、ActionScript では許可されていません。標準 (ストレート) の二重引用符に変更してください。

付録 B

使用を避ける Flash 4 演算子

B

次の表に、ActionScript 2.0 では使用されなくなった Flash 4 専用の演算子を示します。これらの演算子は、Flash Player 4 以前用にパブリッシュする場合以外は使用しないでください。

演算子	説明	結合性
`not`	論理否定 (NOT)	右から左へ
`and`	論理積 (AND)	左から右へ
`or`	論理和 (OR) (Flash 4 スタイル)	左から右へ
`add`	ストリングの連結 (以前の &)	左から右へ
`lt`	より小さい (ストリングバージョン)	左から右へ
`le`	より小さいか等しい (ストリングバージョン)	左から右へ
`gt`	より大きい (ストリングバージョン)	左から右へ
`ge`	より大きいか等しい (ストリングバージョン)	左から右へ
`eq`	等しい (ストリングバージョン)	左から右へ
`ne`	等しくない (ストリングバージョン)	左から右へ

付録 C

キーボードのキーとキーコードの値

C

次の表は、標準キーボード上のすべてのキーと、ActionScript でそれらのキーを識別するために使用するキーコード値および ASCII キーコード値の一覧です。

- 693 ページの「英字 A ～ Z と標準数字 0 ～ 9」
- 696 ページの「テンキー上のキー」
- 696 ページの「ファンクションキー」
- 697 ページの「その他のキー」

キー定数を使用すると、キーの押下動作のビルトインビヘイビアを取得できます。on() ハンドラの詳細については、『ActionScript 2.0 リファレンスガイド』の「on ハンドラ」を参照してください。SWF ファイルを使用して、押されたキーのキーコード値と ASCII キーコード値をキャプチャするには、次の ActionScript コードを使用します。

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("DOWN -> Code: " + Key.getCode() + "\tACSII: " + Key.getAscii() +
  "\tKey: " + chr(Key.getAscii()));
};
Key.addListener(keyListener);
```

Key クラスの詳細については、『ActionScript 2.0 リファレンスガイド』の「Key」を参照してください。[制御]-[ムービープレビュー] を選択してオーサリング環境で SWF ファイルをテストする場合、キーをキャプチャするには、必ず [制御]-[キーボードショートカットを無効] を選択してください。

英字 A ～ Z と標準数字 0 ～ 9

次の表に、英字 A ～ Z と標準数字 0 ～ 9 の標準キーと、ActionScript でそれらのキーを識別するために使用するキーコード値を示します。

英字キーまたは数字キー	キーコード	ASCII キーコード
A	65	65
B	66	66
C	67	67

英字キーまたは数字キー	キーコード	ASCII キーコード
D	68	68
E	69	69
F	70	70
G	71	71
H	72	72
I	73	73
J	74	74
K	75	75
L	76	76
M	77	77
N	78	78
O	79	79
P	80	80
Q	81	81
R	82	82
S	83	83
T	84	84
U	85	85
V	86	86
W	87	87
X	88	88
Y	89	89
Z	90	90
0	48	48
1	49	49
2	50	50
3	51	51
4	52	52
5	53	53
6	54	54

英字キーまたは数字キー	キーコード	ASCII キーコード
7	55	55
8	56	56
9	57	57
a	65	97
b	66	98
c	67	99
d	68	100
e	69	101
f	70	102
g	71	103
h	72	104
i	73	105
j	74	106
k	75	107
l	76	108
m	77	109
n	78	110
o	79	111
p	80	112
q	81	113
r	82	114
s	83	115
t	84	116
u	85	117
v	86	118
w	87	119
x	88	120
y	89	121
z	90	122

テンキー上のキー

次の表は、テンキー上のキーと、ActionScript でそれらのキーを識別するために使用するキーコード値の一覧です。

テンキー上のキー	キーコード	ASCII キーコード
テンキー 0	96	48
テンキー 1	97	49
テンキー 2	98	50
テンキー 3	99	51
テンキー 4	100	52
テンキー 5	101	53
テンキー 6	102	54
テンキー 7	103	55
テンキー 8	104	56
テンキー 9	105	57
* (乗算)	106	42
+ (加算)	107	43
Enter	13	13
- (減算)	109	45
. (小数点)	110	46
/ (除算)	111	47

ファンクションキー

次の表は、標準キーボード上のファンクションキーと、ActionScript でそれらのキーを識別するために使用するキーコード値の一覧です。

ファンクションキー	キーコード	ASCII キーコード
F1	112	0
F2	113	0
F3	114	0
F4	115	0
F5	116	0

ファンクションキー	キーコード	ASCII キーコード
F6	117	0
F7	118	0
F8	119	0
F9	120	0
F10	このキーはシステムで予約されており、ActionScript で使用することはできません。	このキーはシステムで予約されており、ActionScript で使用することはできません。
F11	122	0
F12	123	0
F13	124	0
F14	125	0
F15	126	0

その他のキー

次の表は、文字、数字、テンキー、ファンクションキー以外の標準キーボード上のキーと、ActionScript でそれらのキーを識別するために使用するキーコード値の一覧です。

キー	キーコード	ASCII キーコード
BackSpace	8	8
Tab	9	9
Enter	13	13
Shift	16	0
Ctrl	17	0
CapsLock	20	0
Esc	27	27
スペースバー	32	32
PageUp	33	0
PageDown	34	0
End	35	0
Home	36	0

キー	キーコード	ASCII キーコード
← (左矢印)	37	0
↑ (上矢印)	38	0
→ (右矢印)	39	0
↓ (下矢印)	40	0
Insert	45	0
Delete	46	127
NumLock	144	0
ScrLk	145	0
Pause/Break	19	0
; :	186	59
= +	187	61
- _	189	45
/ ?	191	47
` ~	192	96
[{	219	91
\ |	220	92
] }	221	93
" '	222	39
,	188	44
.	190	46
/	191	47

その他のキーについては、この付録の最初に示した ActionScript を使用して目的のキーを押すと、そのキーコード値と ASCII キーコード値を出力できます。

付録 D

以前のバージョンの Flash Player 用のスクリプト作成

D

Adobe Flash オーサリングツールおよび Flash Player のリリースごとに、ActionScript は大幅に変更されてきました。ActionScript の可能性を最大限に活用できるのは、Adobe Flash Player 8 用にコンテンツを作成したときです。Flash を使用して引き続き以前のバージョンの Flash Player 用のコンテンツを作成することは可能ですが、すべての ActionScript エレメントを使用できるわけではありません。

この章では、対象とする Flash Player のバージョンに合った、正しいシンタックスのスクリプトを作成するためのガイドラインを示します。

メモ

アドビ システムズ社の Web サイトで Flash Player バージョンの普及に関する調査を閲覧できます。www.adobe.com/software/player_census/flashplayer/ を参照してください。

以前のバージョンの Flash Player を対象にすることについて

スクリプトを作成する際は、『ActionScript 2.0 リファレンスガイド』で各エレメントの対応 Player 情報を参照し、対象となる Flash Player のバージョンで、使用するエレメントがサポートされているかどうかを確認します。また、[アクション] ツールボックスを表示することでも、使用できるエレメントを確認できます。対象とするバージョンでサポートされていないエレメントは、黄色で表示されます。

Flash Player 6 以降用のコンテンツを作成している場合には、ActionScript 2.0 を使用してください。ActionScript 2.0 は、向上したコンパイルエラーやより堅固なオブジェクト指向プログラミング機能など、ActionScript 1.0 では利用できない多くの重要な機能を提供しています。

ドキュメントをパブリッシュするときに使用する Flash Player と ActionScript のバージョンを指定するには、[ファイル]-[パブリッシュ設定] を選択してから、[Flash] タブで選択を行います。Flash Player 4 をターゲットにする場合には、次のセクションを参照してください。

Flash Player 4 用のコンテンツ作成での Flash の使用

Flash を使って Flash Player 4 用のコンテンツを作成するには、[パブリッシュ設定] ダイアログボックス ([ファイル]-[パブリッシュ設定]) の [Flash] タブで [Flash Player 4] を選択します。

Flash Player 4 の ActionScript で使用されるデータ型は、数値とストリングの両方の処理で使用される基本プリミティブ型のみです。Flash Player 4 用のアプリケーションを作成する場合、[ActionScript] ツールボックスの [使用を避ける機能] カテゴリのストリング演算子を使う必要があります。

Flash Player 4 用にパブリッシュするときに次の Flash の機能を使用できます。

- 配列アクセス演算子とオブジェクトアクセス演算子 (`[ ]`)
- ドット演算子 (`.`)
- 論理演算子、代入演算子、前インクリメント演算子、後インクリメント演算子、デクリメント演算子
- 剰余演算子 (`%`)、Math クラスのすべてのメソッドとプロパティ

次の言語エレメントは、Flash Player 4 では本来サポートされていません。Flash では近似の演算子と関数が書き出されるため、結果の数値の正確さが劣ります。さらに、SWF ファイルに級数近似が組み込まれるので、これらの言語エレメントを使用すると、Flash 4 Player の SWF ファイルは Flash Player 5 以降よりもサイズが大きくなります。

- `for`、`while`、`do..while`、`break`、`continue` アクション
- `print()` および `printAsBitmap()` アクション
- `switch` アクション

詳細については、699 ページの「以前のバージョンの Flash Player を対象にすることについて」を参照してください。

Flash 4 ファイルを開くための Flash の使用

Flash 4 の ActionScript で使用されるデータ型はストリングのみでした。また、式ではいろいろなタイプの演算子を使用して、値をストリングとして処理するか、数値として処理するかを指定することができました。Flash の今後のリリースでは、すべてのデータ型について、同じ演算子を使用することができます。

Flash 4 で作成されたファイルを Flash 5 以降のバージョンで開くと、新しいシンタックスとの互換性を保つように、AcrionScript の式が自動的に変換されます。Flash では、データ型と演算子が次のように変換されます。

- = 演算子は、Flash 4 では数値が等価であることを表すために使用されました。Flash 5 以降では、等価演算子は == で、= は代入演算子です。Flash 4 の = 演算子はすべて、自動的に == に変換されます。
- Flash では、演算子が予想どおりに機能するように型が自動的に変換されます。複数のデータ型が導入されたため、次の演算子の意味が変更されました。

  +, ==, !=, <>, <, >, >=, <=

  Flash 4 の ActionScript では、これらの演算子は常に数値演算子でしたが、Flash 5 以降ではオペランドのデータ型によって意味が異なります。読み込まれたファイルで演算子の意味が変わってしまうことを防ぐために、これらの演算子のすべてのオペランドに対して、`Number()` 関数が挿入されます。ただし、定数値はどのような場合でも明らかに数値であるとわかるので、`Number()` では囲まれません。これら演算子の詳細については、138 ページの「演算子の優先順位と結合性について」および 691 ページの「使用を避ける Flash 4 演算子」の演算子の表を参照してください。
- Flash 4 では、エスケープシーケンス `\n` は復帰文字 (ASCII 13) を表しました。Flash 5 以降では、ECMA-262 規格に準拠して、`\n` は改行文字 (ASCII 10) を表します。そのため、Flash 4 の FLA ファイルの `\n` は、自動的に `\r` に変換されます。
- & 演算子は、Flash 4 ではストリングの結合に使用されていました。Flash 5 以降では、& はバイト単位の AND 演算子です。現在使用されているストリング結合演算子は `add` です。Flash 4 のファイルの & 演算子はすべて、自動的に `add` 演算子に変換されます。
- `Get Timer`、`Set Variable`、`Stop`、`Play` など、Flash 4 の関数の多くでは、閉じカッコが必要ありませんでした。シンタックスに統一性を持たせるため、`getTimer` 関数とすべてのアクションでは、カッコ (`()`) が必要になりました。Flash 4 ファイルの変換中に、必要な閉じカッコが自動的に追加されます。
- Flash 5 以降では、`getProperty` 関数が存在しないムービークリップで実行されると、0 ではなく `undefined` の値が返されます。Flash 4 より後の ActionScript では、`undefined == 0` ステートメントは `false` です。Flash 4 では、`undefined == 1` です。Flash 5 以降では、等価比較で `Number()` 関数を使用して、Flash 4 ファイルを変換するときのこの問題を解決します。次の例では、`Number()` により `undefined` が 0 に変換されるため、比較が成功します。

```
getProperty("clip", _width) == 0
Number(getProperty("clip", _width)) == Number(0)
```

メモ	Flash 4 の ActionScript で、Flash 5 以降のキーワードを変数名として使用している場合、Flash でコンパイルするとシンタックスエラーになります。この問題を解決するには、該当する変数が出現するすべての箇所で変数名を変更します。詳細については、101 ページの「予約語について」および 52 ページの「変数の命名について」を参照してください。

スラッシュシンタックスの使用法

Flash 3 と 4 では、ムービークリップや変数のターゲットパスを表すためにスラッシュシンタックス (/) が使用されていました。スラッシュシンタックスでは、次の例のようにスラッシュがドットの代わりに使用され、変数の前にコロンが 1 つあります。

```
myMovieClip/childMovieClip:myVariable
```

これと同じターゲットパスを、Flash Player 5 以降のバージョンでサポートされているドットシンタックスに従って記述すると、次のシンタックスのようになります。

```
myMovieClip.childMovieClip.myVariable
```

スラッシュシンタックスは主に `tellTarget` アクションと併せて使用されていました。しかし、この tellTarget アクションも使用しないようお勧めします。`with` アクションの方がドットシンタックスとの互換性が高いため、こちらを使用してください。詳細については、『ActionScript 2.0 リファレンスガイド』の「`tellTarget` 関数」および「`with` ステートメント」を参照してください。

付録 E

ActionScript 1.0 でのオブジェクト指向プログラミング

E

この付録は、ActionScript 1.0 オブジェクトモデルを使ってスクリプトを作成するための情報を Adobe Flash のドキュメントから抜粋したものです。この情報がここに挙げられているのは、次の理由からです。

- Flash Player 5 に対応したオブジェクト指向のスクリプトを書くためには、ActionScript 1.0 を使う必要があります。
- 既に ActionScript 1.0 を使ってオブジェクト指向のスクリプトを作成しており、ActionScript 2.0 に切り替える準備ができていない場合も、スクリプトを作成する際に必要な情報をこの付録から得ることができます。

ActionScript を使ってオブジェクト指向スクリプトを作成したことがなく、Flash Player 5 への対応も必要ないのであれば、この付録内の情報を使用すべきではありません。ActionScript 1.0 でのオブジェクト指向スクリプト作成は使用を避けつつあるためです。代わりに、ActionScript 2.0 の利用の詳細について、179 ページ、第 6 章の「クラス」を参照してください。

この章では、次のセクションについて説明します。

ActionScript 1.0 について 704
ActionScript 1.0 でのカスタムオブジェクトの作成 706
ActionScript 1.0 でのカスタムオブジェクトへのメソッドの割り当て 706
ActionScript 1.0 でのイベントハンドラメソッドの定義. 708
ActionScript 1.0 での継承の作成. 710
ActionScript 1.0 でのオブジェクトへの getter/setter プロパティの追加 711
ActionScript 1.0 での Function オブジェクトプロパティの使用 712

メモ

この付録内の一部の例では、Object.registerClass() メソッドを使用しています。このメソッドは Flash Player 6 以降のバージョンでのみサポートされているため、Flash Player 5 を対象にする場合には、このメソッドは使用しないでください。

ActionScript 1.0 について

メモ

ActionScript 2.0 を使用すると、特に複雑なアプリケーションの場合は、多くの利点があります。ActionScript 2.0 の使用方法の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

ActionScript は、オブジェクト指向プログラミング言語です。オブジェクト指向プログラミングでは、" オブジェクト " というデータ構造を使って、オブジェクトの動作や外観を制御するプロパティとメソッドをグループ化します。オブジェクトを利用することで、コードの整理と再利用が可能になります。オブジェクトは、一度定義すれば、名前で参照することができ、使うたびに再定義する必要はありません。

" クラス " は、オブジェクトの包括的なカテゴリです。クラスは、共通のプロパティを持ち、同じ方法で制御できる一連のオブジェクトを定義します。プロパティは、サイズ、位置、色、透明度など、オブジェクトを定義する属性です。プロパティはクラスに対して定義され、プロパティの値はクラス内の各オブジェクトに対して設定されます。メソッドは、オブジェクトのプロパティを設定および取得する関数です。たとえば、オブジェクトのサイズを計算するメソッドを定義することができます。プロパティと同様、メソッドはオブジェクトクラスに対して定義され、そのクラス内の各オブジェクトに対して呼び出されます。

ActionScript には、MovieClip クラス、Sound クラスなどいくつかのビルトインクラスが用意されています。アプリケーションのオブジェクトにカテゴリを定義するカスタムクラスを作成することもできます。

ActionScript のオブジェクトは、単なるデータコンテナにすることも、ムービークリップ、ボタン、またはテキストフィールドとしてステージ上にグラフィカルに表すこともできます。すべてのムービークリップはビルトイン MovieClip クラスのインスタンスであり、すべてのボタンはビルトイン Button クラスのインスタンスです。各ムービークリップインスタンスには、MovieClip クラスのすべてのプロパティ (`_height`、`_rotation`、`_totalframes` など) とすべてのメソッド (`gotoAndPlay()`、`loadMovie()`、`startDrag()` など) が含まれます。

オブジェクトのクラスを定義するには、" コンストラクタ関数 " と呼ばれる特別な関数を作成します。ビルトインクラスにはビルトインコンストラクタ関数があります。たとえば、自転車に乗る人の情報がアプリケーションで必要な場合は、`time` (時間) および `distance` (距離) のプロパティと、`getSpeed()` (自転車の速度の取得) のメソッドを持つコンストラクタ関数 `Biker()` を作成します。

```
function Biker(t, d) {
  this.time = t;
  this.distance = d;
  this.getSpeed = function() {return this.time / this.distance;};
}
```

この例では、2 つの " パラメータ " と呼ばれる情報 (t と d) に基づいてジョブを実行する関数を作成します。関数を呼び出してオブジェクトのインスタンスを新たに作成するときに、これらのパラメータが関数に渡されます。次のコードは emma および hamish という名前の Biker オブジェクトのインスタンスを作成し、上記の ActionScript の getSpeed() メソッドを使用して emma インスタンスの速度をトレースします。

```
emma = new Biker(30, 5);
hamish = new Biker(40, 5);
trace(emma.getSpeed()); // 6 をトレースする
```

オブジェクト指向のスクリプト作成では、特定の順序に従ってクラスが他のクラスのプロパティやメソッドを受け継ぐことができます。これを " 継承 " と言います。継承を使用すると、クラスのプロパティやメソッドを拡張または再定義することができます。別のクラスからプロパティやメソッドを継承するクラスを " サブクラス " と言います。別のクラスにプロパティやメソッドを渡すクラスを " スーパークラス " と言います。クラスは、サブクラスまたはスーパークラスのいずれかになります。

オブジェクトの構成は複雑で、プロパティとメソッドがないこともあれば、複数存在することもあります。変数と同様、プロパティにはそれぞれ名前と値があります。プロパティはオブジェクトごとに割り当てられ、変更や取り出しが可能な値が設定されます。これらの変数には、ストリング、数値、ブール値、オブジェクト、ムービークリップ、未定義 (undefined) などの任意のデータ型を指定できます。たとえば、次のようなデータ型の値を設定できます。

```
customer.name = "Jane Doe";
customer.age = 30;
customer.member = true;
customer.account.currentRecord = 609;
customer.mcInstanceName._visible = true;
```

オブジェクトのプロパティがオブジェクトであることもあります。上記の例の 4 行目では、account はオブジェクト customer のプロパティで、currentRecord はオブジェクト account のプロパティです。そして、currentRecord プロパティのデータ型は Number です。

ActionScript 1.0 でのカスタムオブジェクトの作成

メモ

ActionScript 2.0 を使用すると、特に複雑なアプリケーションの場合は、多くの利点があります。ActionScript 2.0 の使用方法の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

カスタムオブジェクトを作成するには、コンストラクタ関数を定義します。コンストラクタ関数には常に、作成するオブジェクトのタイプと同じ名前が付けられます。コンストラクタ関数の本体でキーワード this を使用すると、コンストラクタによって作成されるオブジェクトを参照できます。コンストラクタ関数を呼び出すと、this が隠しパラメータとして関数に渡されます。たとえば、プロパティ radius の値に従って円を作成するコンストラクタ関数は次のコードのようになります。

```
function Circle(radius) {
  this.radius = radius;
}
```

コンストラクタ関数を定義した後、オブジェクトのインスタンスを作成する必要があります。コンストラクタ関数の前に new 演算子を指定して、新しいインスタンスに変数名を割り当てます。たとえば、次のコードでは、new 演算子によって半径 5 の Circle オブジェクトが作成され、変数 myCircle に割り当てられます。

```
myCircle = new Circle(5);
```

メモ

オブジェクトのスコープは、そのオブジェクトが割り当てられる変数のスコープと同じです。

ActionScript 1.0 でのカスタムオブジェクトへのメソッドの割り当て

メモ

ActionScript 2.0 を使用すると、特に複雑なアプリケーションの場合は、多くの利点があります。ActionScript 2.0 の使用方法の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

オブジェクトのコンストラクタ関数内でオブジェクトのメソッドを定義できます。ただし、この方法はお勧めできません。コンストラクタ関数を使用するたびにメソッドを定義するためです。次の例では、getArea() メソッドと getDiameter() メソッドを作成し、構築した myCircle インスタンスの面積と直径を、半径を 55 に設定してトレースします。

```
function Circle(radius) {
  this.radius = radius;
  this.getArea = function(){
    return Math.PI * this.radius * this.radius;
  };
  this.getDiameter = function() {
    return 2 * this.radius;
  };
}
var myCircle = new Circle(55);
trace(myCircle.getArea());
trace(myCircle.getDiameter());
```

コンストラクタ関数を定義するときには、prototype プロパティが自動的に割り当てられます。prototype プロパティは、関数で作成されたオブジェクトに対するデフォルトプロパティ値を表します。オブジェクトの新しいインスタンスは、それぞれ __proto__ プロパティを持っています。このプロパティは、コンストラクタ関数で作成された prototype プロパティを参照します。したがって、オブジェクトの prototype プロパティにメソッドを割り当てると、新たに作成されたそのオブジェクトのすべてのインスタンスでメソッドを使用できます。コンストラクタ関数の prototype プロパティが定義されるのは 1 か所のみで、オブジェクトやクラスの新しいインスタンスによって参照されるので、このプロパティにメソッドを割り当てることをお勧めします。prototype プロパティと __proto__ プロパティを使用すると、オブジェクト指向のプログラムでコードを再利用できるように、オブジェクトの拡張性を向上させることができます。詳細については、710 ページの「ActionScript 1.0 での継承の作成」を参照してください。

次の手順は、カスタムオブジェクト Circle に getArea() メソッドを割り当てる方法を示します。

カスタムオブジェクトにメソッドを割り当てるには :

1. 次のように、コンストラクタ関数 Circle() を定義します。

```
function Circle(radius) {
  this.radius = radius;
}
```

2. Circle オブジェクトの getArea() メソッドを定義します。getArea() メソッドでは円の面積が計算されます。次の例では、関数リテラルを使用して getArea() メソッドを定義し、getArea プロパティを円のプロトタイプオブジェクトに割り当てます。

```
Circle.prototype.getArea = function () {
     return Math.PI * this.radius * this.radius;
};
```

3. 次の例では、Circle オブジェクトのインスタンスを作成します。

```
var myCircle = new Circle(4);
```

4. 次のコードのように、新しく作成した myCircle オブジェクトの getArea() メソッドを呼び出します。

```
var myCircleArea = myCircle.getArea();
trace(myCircleArea); //50.265... をトレースする
```

myCircle オブジェクトの getArea() メソッドが検索されます。このオブジェクトには getArea() メソッドがないので、prototype オブジェクト Circle.prototype で getArea() が検索されます。メソッドが見つかると、呼び出して、myCircleArea をトレースします。

ActionScript 1.0 でのイベントハンドラメソッドの定義

メモ

ActionScript 2.0 を使用すると、特に複雑なアプリケーションの場合は、多くの利点があります。ActionScript 2.0 の使用方法の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

ムービークリップの ActionScript クラスを作成して、そのクラスの prototype オブジェクトにイベントハンドラメソッドを定義できます。prototype オブジェクトにイベントハンドラメソッドを定義すると、そのシンボルのすべてのインスタンスがそれらのイベントに対して同じ応答を返します。

また、イベントハンドラメソッド onClipEvent() または on() を個々のインスタンスに追加して、そのインスタンスのイベントが発生したときにのみ実行するよう個別に指示することもできます。onClipEvent() メソッドと on() メソッドが、イベントハンドラメソッドを無効にすることはありません。したがって、どちらのイベントもスクリプトを実行します。ただし、prototype オブジェクトにイベントハンドラメソッドを定義している上に、個々のインスタンスのイベントハンドラメソッドも定義している場合は、インスタンスの定義が prototype の定義よりも優先されます。

オブジェクトの prototype オブジェクトにイベントハンドラメソッドを定義するには :

1. ムービークリップシンボルを作成します。[ライブラリ] パネルでシンボルを選択し、[ライブラリ] パネルのポップアップメニューから [リンケージ] を選択して、リンケージ識別子を theID に設定します。

2. [アクション] パネル ([ウィンドウ]-[アクション]) で、次の例のように、function ステートメントを使って新しいクラスを定義します。

```
// クラスの定義
function myClipClass() {}
```

タイムラインによってアプリケーションに追加されるか、attachMovie() メソッドまたは duplicateMovieClip() メソッドでアプリケーションに追加される、ムービークリップのすべてのインスタンスにこの新しいクラスが割り当てられます。ムービークリップが組み込みの MovieClip オブジェクトのメソッドおよびプロパティにアクセスできるようにする場合は、新しいクラスに MovieClip クラスから継承させる必要があります。

3. 次の例のようにコードを入力します。

```
// MovieClip クラスから継承
myClipClass.prototype = new MovieClip();
```

これで、クラス myClipClass は MovieClip クラスのすべてのプロパティとメソッドを継承します。

4. 次の例のようなコードを入力して、新しいクラスのイベントハンドラメソッドを定義します。

```
// myClipClass クラスのイベントハンドラメソッドの定義
myClipClass.prototype.onLoad = function() {trace("movie clip loaded");}
myClipClass.prototype.onEnterFrame = function() {trace("movie clip entered
   frame");}
```

5. [ライブラリ] パネルが開いていない場合は、[ウィンドウ]-[ライブラリ] を選択して開きます。

6. 新しいクラスに関連付けるシンボルを選択し、[ライブラリ] パネルのポップアップメニューから [リンケージ] を選択します。

7. [リンケージプロパティ] ダイアログボックスで、[ActionScript に書き出し] を選択します。

8. [識別子] テキストボックスにリンケージ識別子を入力します。

リンケージ識別子は、新しいクラスを関連付けるすべてのシンボルで同じであることが必要です。myClipClass の例では、識別子は theID です。

9. [アクション] パネルで、次のようなコードを入力します。

```
// クラスの登録
Object.registerClass("theID", myClipClass);
this.attachMovie("theID","myName",1);
```

これによって、リンケージ識別子が theID であるすべてのシンボルが、クラス myClipClass に登録されます。myClipClass のすべてのインスタンスは、手順 4 で定義したように動作するイベントハンドラメソッドを持ちます。また、手順 3 で新しいクラスがクラス MovieClip クラスから継承するように指定したので、MovieClip クラスのすべてのインスタンスと同じように動作します。

コード全体は、次の例のようになります。

```
function myClipClass(){}

myClipClass.prototype = new MovieClip();
myClipClass.prototype.onLoad = function(){
   trace("movie clip loaded");
}
myClipClass.prototype.onPress = function(){
   trace("pressed");
}

myClipClass.prototype.onEnterFrame = function(){
   trace("movie clip entered frame");
}
```

```
myClipClass.prototype.myfunction = function(){
   trace("myfunction called");
}

Object.registerClass("myclipID",myClipClass);
this.attachMovie("myclipID","clipName",3);
```

ActionScript 1.0 での継承の作成

メモ

ActionScript 2.0 を使用すると、特に複雑なアプリケーションの場合は、多くの利点があります。ActionScript 2.0 の使用方法の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

継承によって、機能を体系化、拡張、および再利用できるようになります。サブクラスでは、スーパークラスのプロパティとメソッドが継承され、固有のプロパティとメソッドが追加されます。たとえば、現実の世界と同じように、Bike はスーパークラスになり、MountainBike と Tricycle はそのサブクラスになります。どちらのサブクラスも、スーパークラスのメソッドとプロパティ (例 : `wheels`) を継承します。サブクラスには、スーパークラスとは別の、固有のプロパティとメソッドがあります。たとえば、MountainBike サブクラスには、`gears` プロパティがあります。ActionScript では、`prototype` エレメントと `__proto__` エレメントを使用してプロパティとメソッドを継承します。

コンストラクタ関数を定義するときには、`prototype` プロパティが自動的に割り当てられます。`prototype` プロパティは、関数で作成されたオブジェクトに対するデフォルトプロパティ値を表します。`prototype` プロパティを使用して、クラスにプロパティとメソッドを割り当てることができます。詳細については、706 ページの「ActionScript 1.0 でのカスタムオブジェクトへのメソッドの割り当て」を参照してください。

クラスのすべてのインスタンスでは、`__proto__` プロパティに、プロパティやメソッドを継承する元のオブジェクトが設定されています。コンストラクタ関数を使用してオブジェクトを作成すると、`__proto__` プロパティは、コンストラクタ関数の `prototype` プロパティを参照するように設定されます。

オブジェクトのプロパティやメソッドは、明確な階層構造に従って継承されます。プロパティやメソッドを呼び出すと、オブジェクトに該当するエレメントが存在するかどうかが確認されます。存在しない場合、オブジェクトの `__proto__` プロパティの情報 (`myObject.__proto__`) が調べられます。`__proto__` プロパティ以外のプロパティを呼び出している場合は、さらに `myObject.__proto__.__proto__` などが調べられます。

次の例では、コンストラクタ関数 `Bike()` が定義されます。

```
function Bike(length, color) {
  this.length = length;
  this.color = color;
  this.pos = 0;
}
```

次のコードでは、Bike クラスに roll() メソッドが追加されます。

```
Bike.prototype.roll = function() {return this.pos += 20;};
```

こうすると、次のコードを使って Bike の位置をトレースできます。

```
var myBike = new Bike(55, "blue");
trace(myBike.roll());   // 20 をトレースする
trace(myBike.roll());   // 40 をトレースする
```

次のように Bike をスーパークラスとして MountainBike クラスを作成することもできます。その場合は、MountainBike クラスや Tricycle クラスに roll() を追加する必要はありません。

```
MountainBike.prototype = new Bike();
```

これで、次に示すように、MountainBike の roll() メソッドを呼び出せるようになります。

```
var myKona = new MountainBike(20, "teal");
trace(myKona.roll());   // 20 をトレースする
```

ムービークリップ間では、プロパティやメソッドは継承されません。ムービークリップ間での継承を実現するには、Object.registerClass() を使用して、MovieClip クラス以外のクラスをムービークリップに割り当てます。

ActionScript 1.0 でのオブジェクトへの getter/setter プロパティの追加

メモ

ActionScript 2.0 を使用すると、特に複雑なアプリケーションの場合は、多くの利点があります。ActionScript 2.0 の使用方法の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

オブジェクトに対する getter/setter プロパティは、Object.addProperty() メソッドを使用して作成できます。

getter 関数はパラメータがない関数です。戻り値のタイプは不特定です。戻り値のタイプは呼び出しごとに変わることがあります。戻り値は、プロパティの現在の値として扱われます。

setter 関数のパラメータは 1 つであり、プロパティの新しい値です。たとえば、プロパティ x がステートメント x = 1 によって割り当てられた場合、setter 関数には Nubmer 型のパラメータ 1 が渡されます。setter 関数の戻り値は無視されます。

Flash は getter/setter プロパティを読み込むと、getter 関数を呼び出します。この関数の戻り値は prop の値になります。Flash は getter/setter プロパティを書き込むと、setter 関数を呼び出し、それに新しい値をパラメータとして渡します。指定された名前のプロパティが既に存在する場合は、新しいプロパティによって上書きされます。

getter/setter プロパティはプロトタイプオブジェクトに追加できます。getter/setter プロパティをプロトタイプオブジェクトに追加すると、プロトタイプオブジェクトを継承するすべてのオブジェクトインスタンスは getter/setter プロパティを継承します。つまり、1 つのプロトタイプオブジェクトに getter/setter プロパティを追加するだけで、クラスのすべてのインスタンスに同じプロパティを追加できます (プロトタイプオブジェクトにメソッドを追加するのと似ています)。継承先のプロトタイプオブジェクトの getter/setter プロパティに対して呼び出される getter/setter 関数には、このプロトタイプオブジェクトへの参照ではなく、継承元のオブジェクトへの参照が渡されます。

プレビューモードの [デバッグ]-[変数のリストアップ] コマンドでは、`Object.addProperty()` を使ってオブジェクトに追加した getter/setter プロパティがサポートされています。この方法でオブジェクトに追加したプロパティは、他のプロパティとともに [出力] パネルに表示されます。[出力] パネル内で、getter/setter プロパティは `[getter/setter]` というプレフィックス (接頭辞) で識別されます。[変数のリストアップ] コマンドの詳細については、『Flash ユーザーガイド』を参照してください。

ActionScript 1.0 での Function オブジェクトプロパティの使用

メモ

ActionScript 2.0 を使用すると、特に複雑なアプリケーションの場合は、多くの利点があります。ActionScript 2.0 の使用方法の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

Function オブジェクトの `call()` メソッドと `apply()` メソッドを利用すると、関数が適用されるオブジェクトと、関数に渡されるパラメータの値を指定することができます。ActionScript のすべての関数は Function オブジェクトで表されます。したがって、すべての関数は `call()` および `apply()` をサポートしています。コンストラクタ関数を使ってカスタムクラスを作成するときや、関数を使ってカスタムクラスにメソッドを定義するときに、その関数を `call()` メソッドや `apply()` で呼び出すことができます。

ActionScript 1.0 での Function.call() メソッドを使った関数の呼び出し

メモ

ActionScript 2.0 を使用すると、特に複雑なアプリケーションの場合は、多くの利点があります。ActionScript 2.0 の使用方法の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

`Function.call()` メソッドは、Function オブジェクトによって表される関数を呼び出します。

通常、関数呼び出し演算子 (`()`) を `call()` メソッドの代わりに使用できます。関数呼び出し演算子を使用すると、簡潔で読みやすいコードを作成できます。`call()` メソッドは、主に関数呼び出しの `this` パラメータを明示的に制御する場合に役立ちます。通常、関数をオブジェクトのメソッドとして、関数の本体内で呼び出すと、次のように `this` が `myObject` に設定されます。

```
myObject.myMethod(1, 2, 3);
```

`this` が他の異なる場所をポイントするように設定する場合もあります。たとえば、オブジェクトのメソッドとして呼び出す関数が、実際にはそのオブジェクトのメソッドとして格納されていない場合などです。例を次に示します。

```
myObject.myMethod.call(myOtherObject, 1, 2, 3);
```

関数をオブジェクトのメソッドとして呼び出さずに通常の関数として呼び出すには、*thisObject* パラメータに値 `null` を渡します。たとえば、次の 2 つの関数呼び出しは同じです。

```
Math.sin(Math.PI / 4)
Math.sin.call(null, Math.PI / 4)
```

Function.call() メソッドを使って関数を呼び出すには :

- これには次のシンタックスを使用します。

  myFunction`.call(`*thisObject*`, `*parameter1*`, ..., `*parameterN*`)`

  このメソッドは次のパラメータを取ります。

  - パラメータ *thisObject* は、関数本体の内部で `this` の値を指定します。
  - パラメータ *parameter1*..., *parameterN* は、*myFunction* に渡すパラメータを指定します。指定できるパラメータの数は 0 個以上です。

ActionScript 1.0 での Function.apply() を使って関数を適用したオブジェクトの指定

メモ

ActionScript 2.0 を使用すると、特に複雑なアプリケーションの場合は、多くの利点があります。ActionScript 2.0 の使用方法の詳細については、179 ページ、第 6 章の「クラス」を参照してください。

`Function.apply()` メソッドは、ActionScript が呼び出す関数内で使用される `this` の値を指定します。このメソッドは、呼び出される関数に渡されるパラメータも指定します。

パラメータは Array オブジェクトとして指定されます。これは、スクリプトが実際に実行されるまで、渡されるパラメータ数が不明である場合にも便利です。

詳細については、『ActionScript 2.0 リファレンスガイド』の「`apply (Function.apply method)`」を参照してください。

Function.apply() を使って、関数を適用するオブジェクトを指定するには :

- これには次のシンタックスを使用します。

  myFunction`.apply(`*thisObject*`, `*argumentsObject*`)`

  このメソッドは次のパラメータを取ります。

  - パラメータ *thisObject* は、*myFunction* を適用する対象のオブジェクトを指定します。
  - パラメータ *argumentsObject* は、*myFunction* にパラメータとして渡されるエレメントで構成される配列を定義します。

索引

記号

\n 405
\r 405
\t 405
\" 405
\b 405
\f 405
\unnnn 405
\xnn 405
\' 405
_lockroot、使用 662
_root スコープ 86
_root プロパティとロードされたムービークリップ 305

A

ActionScript
　Flash Player 673
　キューポイントの作成 554
　説明 29、30
　バージョンの比較 30
ActionScript 2.0
　ActionScript 2.0 クラスのムービークリップへの割り当て 327
　コンパイルエラーメッセージ 685
ActiveX コントロール 598
Adaptively sampled Distance Field (ADF) 356、359
ADF 356、359
allowNetworking 628
antiAliasType プロパティ 357、360、363
ARGB (アルファ付き RGB) 466
ASCII 値
　キーボードのキー 693
　その他のキー 697
　テンキー上のキー 696
　ファンクションキー 696
ASCII、定義 397
ASO ファイル 233
　削除 234
　使用 233

B

bitmap
　テキスト 356
BitmapData クラス
　置き換えマップフィルタ 477
　使用 476、542
　説明 475
　ノイズ効果 476
　フィルタの適用 447
Bounce イージングクラス 432

C

cacheAsBitmap プロパティ 318
clone () メソッド
　使用 474
　説明 474
CSM
　説明 356
　パラメータについて 356
CSS
　ActionScript でのスタイルの定義 373
　HTML タグでの例 377
　TextField.StyleSheet クラス 371
　XML タグでの例 380
　サポートするプロパティ 370
　新規タグを定義するための使用 379
　スタイルクラスの適用 376
　スタイルの組み合わせ 375
　テキストのフォーマット 369

テキストフィールドへの適用 374
ビルトイン HTML タグへのスタイルの適用 376
ロード 372
CustomFormatter クラス
使用 526
説明 525

D

Delegate クラス
使用 299
説明 298
depth
インスタンスの判断 316
管理 314
次に利用可能なものの判断 315
定義 314
ムービークリップ用の確認 316
do..while ループ 121
Drawing API
円の描画 486
角丸矩形の描画 485
曲線の描画 483
矩形の描画 484
三角形の描画 483、487
使用 572
説明 481
線と塗り 518
線のスタイル 489
線、曲線、シェイプの描画 482
特定のシェイプの描画 481、484
複雑なグラデーションの塗り 488
プログレスバー 572
drawingAPI
Tween クラスと TransitionManager クラス 495

E

ECMA-262 仕様 77
EnterFrame、フレームレート 417
extends キーワード 254
シンタックス 255
説明 254
External API
使用 600
説明 599
ExternalInterface クラス
使用 600
説明 599

F

FileReference クラス
download() メソッド 579
アプリケーションの構築 581
セキュリティ 580
説明 579
Flash 4 ファイル、Flash 8 で開く 700
Flash 8、ActionScript の新機能と変更点 18
Flash Play 9.x、ActionScript 2.0 の新機能と変更点 17
Flash Player
ActionScript 673
displaying full screen 596
displaying or dimming the context menu 596
normal menu view 596
scaling SWF files to 596
クラス、説明 238
コーディング規則 673
通信 595
パブリッシュ設定 31
メソッド 598
Flash Player 4
コンテンツ作成 700
Flash Player 7
新しいセキュリティモデル 630、636、641
既存スクリプトの移植 610
Flash Player 8
使用を避けるされなくなった言語エレメント 25
新規および変更された ActionScript エディタ機能 25
新規および変更された言語エレメント 20
Flash ビデオ
「ビデオ」を参照
FlashVars
説明 341
テキストの表示のための使用 342
FlashVars プロパティ
使用 64
説明 341
FLV ビデオ、「ビデオ」を参照
FLV ファイル
「ビデオ」も参照
FLV に合わせたサーバーの設定 561
FLV バナーの作成 546
Macintosh 562
外部ビデオ 543
外部ビデオのプリロード 548
キューポイント 550、551
キューポイントの操作 553
コードによるナビゲーション 556
実行時の外部ファイルのロード 545

プリロード 548
プログレスバーの作成 567
メタデータ 559
FLVPlayback コンポーネント
seek() メソッド 556
キューポイント 553
キューポイントのシーク 557、558
キューポイントの使用 554
指定された長さのシーク 556
操作するキューポイントの作成 554
for ステートメント、記述 680
for ループ 117
例 130
for..in ループ 118
fscommand() function
commands and arguments 596
fscommand() 関数
Director との通信 598
使用 595

G

getter メソッド
使用 208
説明 207
getURL() メソッド 507
glyphRange ノード、説明 350

H

hitTest() メソッド 517
HTML
引用符で囲まれたタグ 383
サポートされているタグ 384
スタイルでの使用例 377
タグを定義するための CSS の使用 379
テキストフィールド 335
テキストフィールドでの使用 383
テキストを流し込むための <img> タグ 382、386、391
ビルトインタグのスタイル設定 376
HTTP プロトコル
ActionScript メソッド 570
サーバーサイドスクリプトとの通信 574
HTTPS プロトコル 570

I

ID3 タグ 540
if..else if ステートメントの記述 107
if..else ステートメントの記述 106
IIS 6.0 Web サーバー 561
IME (Input Method Editor)
使用 401
説明 401
import
ステートメント 444
パッケージ内の複数のクラス 444
ワイルドカードの使用 444
Input Method Editor
使用 401
説明 401
interface キーワード 267
IP アドレス
セキュリティ 629
ポリシーファイル 637

J

JavaScript
ActionScript 77
Netscape 598
国際標準 77
メッセージの送信 596
JPEG ファイル
テキストフィールドへの埋め込み 391
ムービークリップへのロード 307、533

L

LiveDocs、説明 15
loadMovie() 関数 571
loadVariables() 関数 571
LoadVars オブジェクト、作成 576
LoadVars クラス
HTTP ステータスの確認 578
使用 575
テキストの表示のための使用 343
テキストファイルからの変数のロード 344
Locale クラス
使用 399
説明 399

M

Macromedia Director、通信 598
MediaPlayback コンポーネント
　キューポイントの使用 555
MIME 形式、標準 575
MIME タイプ 561
MovieClip クラス
　blendMode プロパティ 478
　filters プロパティ 445
　scale9Grid プロパティ 498
　描画メソッド 481
　フィルタプロパティの調整 472
MovieClip データ型、定義 37
moviename_DoFSCommand 関数 596
MP3 ファイル
　ID3 タグ 540
　ID3 タグの読み取り 540
　プリロード 539、548
　プログレスバーの作成 565
　ムービークリップへのロード 537
　ロード 537、538

N

Netscape、サポートする JavaScript メソッド 598
NetStream クラス
　onMetaData ハンドラ 559
　onMetaData ハンドラの使用 559

O

on() ハンドラと onClipEvent() ハンドラ 289
　スコープ 295
　ムービークリップへの割り当て 289
OOP
　インスタンスとクラスメンバー 186
　インターフェイス 187
　オブジェクト 185
　カスタムクラスの記述 188
　カプセル化 187
　継承 186
　設計 213
　説明 180、185
　ポリモーフィズム 188
opaqueBackground プロパティ
　使用 324
　定義 318

P

PDF 版のマニュアル、場所 14
projectors, executing applications from 596

R

return ステートメント 681

S

scale-9
　説明 497
scale-9、「9 スライスの拡大 / 縮小」を参照
scrollRect プロパティ 319
setInterval
　使用 419
　フレームレート 417
setRGB メソッド 513
setter メソッド
　使用 208
　説明 207
String class
　length property 408
String クラス
　charAt() メソッド 406
　concat() メソッド 410
　length プロパティ 406
　split() メソッド 411
　substr() メソッドと substring() メソッド 411
　toLowerCase() および toUpperCase() メソッド 409
　toString() メソッド 409
　説明 397、403
super 接頭辞 668
SWF files
　maintaining original size 596
　scaling to Flash Player 596
SWF ファイル
　「ムービークリップ」も参照
　Flash Player でのコントロール 598
　Web ページでの配置 507
　情報の受け渡し 570
　テキストフィールドへの埋め込み 391
　フレームまたはシーンへのジャンプ 505
　ムービークリップへのロード 533
　ロードとアンロード 304
switch ステートメント
　規則 681
　使用 108

T

TCP/IP 接続
　XMLSocket オブジェクト 594
　情報の送信 570
TextField クラス
　使用 334
　スクロールテキストの作成 395
TextField メソッド、使用 352
TextField.StyleSheet クラス 369
　CSS 371
　TextField.styleSheet プロパティ 369、374
　テキストスタイルの作成 373
TextFormat クラス
　使用 367
　説明 362
this キーワード 85、529
　クラス 214
　使用 662
　スコープ 85、214
　接頭辞 666
trace ステートメント、ActionScript の記述 668
Transition クラス
　明度レベルのアニメーション化 467
TransitionManager クラス
　Drawing API 495
　Tween クラス 440
　イージング 427
　使用 430
　説明 427
try..catch..finally ステートメントの記述 110、682
Tween クラス
　_alpha プロパティ 440
　continueTo() メソッド 438、440
　Drawing API 495
　onMotionFinished イベントハンドラ 439
　TransitionManager クラス 440
　yoyo() メソッド 439、440
　アニメーション完了時にトリガー 437
　イージング 427
　オブジェクトのフェード 436
　使用 430、435
　説明 427、434
　フレーム数での継続時間の設定 436
　ぼかしフィルタのアニメーション化 473
　明度レベルのアニメーション化 467
　読み込み 435

U

UCS (Universal Character Set)、定義 397
Unicode
　定義 397
　文字コード 397
Universal Character Set (UCS) 397
URL エンコード形式、情報の送信 570
URL 変数、説明 62
UTF-16 エンコード標準 397
UTF-8 (Unicode) 397

V

void データ型 42

W

Web アプリケーション、連続的な接続 593
while ループ 120
with ステートメント 670

X

XLIFF ファイル 399
XML 586
　DOM 587
　TCP/IP ソケット接続経由での情報の送信 570
　XML メソッドでの情報の送信 570
　階層 587
　サーバーサイドスクリプト 588
　スタイルでの使用例 380
　テキストのロードと表示 345
　変数の変換例 588
XML Localization Interchange File Format 399
XML クラス、メソッド 588
XML ファイル、Flash 8 インストール用の更新 10
XMLSocket オブジェクト
　loadPolicyFile 639
　使用 593
　データの確認 571
　メソッド 593

Z

Zoom トランジションビヘイビア 428

あ

アクション、コーディング規則 659
値
　式内での処理 135
　データ型 185
アニメーション
　グローフィルタ 442
　継続 438
　フィルタ 473
　フレームレート 417、437
　プログレスバーの作成 562
　明度 467
　連続実行 439
アニメーション、シンボル 37
アルファチャンネルマスク 326
アンチエイリアス
　Flash Player サポート 354
　アニメーションおよび読みやすさ 356
　説明 354
アンチエイリアステキスト
　advanced 値 357
　antiAliasType プロパティの設定 357
　Flash Player サポート 354
　normal 値 357
　sharpness と thickness の修正 363
　sharpness プロパティ 362
　サポート 355
　使用 357
　制限 355
　説明 354
　テーブルの作成 360
　太さ 362

い

イージング
　コード 435
　説明 432
　定義 427
イベント
　定義 279
　ブロードキャスト 293
　ムービークリップ 327
イベントハンドラメソッド
　ActionScript 2.0 297
　ActionScript クラスでの定義 281、282
　on() と onClipEvent() 289
　XML データの確認 571
　オブジェクトへの割り当て 292
　関数の割り当て 283
　スコープ 295
　定義 279
　ボタンまたはムービークリップへの割り当て 289
イベントモデル
　on() ハンドラと onClipEvent() ハンドラ 289
　イベントハンドラメソッド 282
　イベントリスナー 285
イベントリスナー 284
　スコープ 295
　ブロードキャストできるクラス 284
イメージ
　「外部メディア」も参照
　テキストフィールドへの埋め込み 391
　ブレンドモードの適用 479
　ムービークリップへのロード 307
インスタンス 418
　OOP 186
　ターゲット化 81
　ターゲット化、ダイナミック 83
　定義 247
　ネストされたもののターゲット化 82
　フィルタの適用 447
インスタンス化
　オブジェクト 248
　定義 180
インスタンス名
　ターゲットパス 81
　定義 301
　変数名との比較 336
インターフェイス
　OOP 187
　継承とインターフェイスについて 272
　作成 268
　説明 265
　定義と実装 267
　データ型としての作成 270
　複雑なインターフェイスの例 275
　命名 267
　例 273
インタラクティブ機能、SWF ファイル
　作成 503
　手法 508
インデックス付き配列 124、129
引用符、ストリング内 41

う

埋め込みフォント
TextField クラスの使用 352
フォントシンボルの埋め込み 348
埋め込み文字
作成と削除 346
テキストフィールドでの使用 347

え

エスケープシーケンス 41
エスケープ文字 405
エラー処理、フィルタ 448
エラーメッセージ 685
エレメント、配列 123
円記号、ストリング内 405
演算子
Flash での使用 156
値との組み合わせ 135
値の操作 136
オペランド 135
加算 145
関係 147
関係および等価 147
結合性 138
後置 143
条件 112、147、155
乗法 145
使用を避ける 691
使用、代入 151
数式 135
数値 145
説明 135
代入 137、150
単項 144
等価 147
ドットと配列アクセス 142
比較 141
ビット単位シフト 153
ビット論理 153、154
優先順位と結合性 138
論理 151、152
演算子の優先順位と結合性 138

お

大文字と小文字の区別
Flash Player のバージョン 78
説明 78
置き換えマップフィルタ
BitmapData クラス 477
イメージに適用 477
使用 470
説明 470
オブジェクト
Flash での作成 69
コーディング規則 660
子に対するループ 115
作成 50、69、248
データ型 40
配列内のデータの整理 70
フェードアウト 418
プロパティへのアクセス 248
メソッド呼び出し 249
オブジェクト指向プログラミング 185
オブジェクト指向プログラミング、「OOP」を参照
オブジェクトプロパティ
値の割り当て 248
オブジェクトリテラル 134
オペランド 135
親ムービークリップ 301
オンラインリソース 16

か

カーソル、カスタムの作成 508
改行文字 405
外部クラスファイル
検索にクラスパスを使用する 193
外部ソース、Flash の接続 569、609
外部メディア 531
FLV ファイルの再生 543
MP3 ファイル 537
MP3 ファイルのロード 565
ProgressBar コンポーネント 535
SWF ファイルおよびイメージファイルのロード 563
SWF ファイルと JPEG ファイルのロード 533
イメージと外部 SWF ファイルのロード 533
使用理由 531
プリロード 548、562
プログレスバーアニメーションの作成 562
ルートタイムライン 537
ロードの説明 532

改ページ文字 405
カウンタ、アクションの繰り返し 114、115
拡張マークアップ言語、「XML」を参照
カスケーディングスタイルシート、「CSS」を参照
カスタムアンチエイリアス
　定義 356
カスタム関数 157
カスタムフォーマッタ
　使用 524
　説明 524
カスタム文字セット、作成 350、351
型チェック
　dynamic 46
　定義 45
　例 45
括弧、説明 92
カプセル化
　使用 213
　説明 187
ガベージコレクション 666
カラー
　値、設定 513
カラーマトリックスフィルタ
　使用 421、467
　説明 466
関係演算子 147
関数
　Flash での使用 169
　返される値 175
　カスタム 157
　関数ブロック 162、163
　関数リテラル 165
　クラスファイル 171
　グローバル関数とタイムライン関数の定義 166
　コールバック 163
　コンストラクタ 166、704
　再利用 169
　作成と呼び出し 169
　種類 159
　説明 157
　定義 166
　匿名関数の記述 163
　トップレベル 160
　トップレベル関数の呼び出し 161
　名前付き関数の記述 162
　名前付き関数の使用 162
　名前付き関数のシンタックス 158
　名前付き関数の標準形式 162
　名前付きと匿名の比較 170
　ネスト 176
　パラメータの受け渡し 173
　非同期 570
　ビルトインとトップレベル 161
　ブラックボックス 158
　ベストプラクティス 671
　変換 34
　変数の使用 172
　ムービークリップの制御 303
　命名 168
　メソッドとの比較 178
　ユーザー定義関数の指定と呼び出し 167
関数リテラル
　再定義 165
　説明 165
完全修飾名
　使用 444
　定義 443

き

キーコード、ASCII
　その他のキー 697
　テンキー 696
　ファンクションキー 696
　文字と数字のキー 693
キーボード
　ASCII キーコード値 693
キーボードコントロール
　ムービークリップの作成 511
キーワード
　_root 86
　extends 254
　interface 267
　this 85
　使用 101
　説明 98
記述、ActionScript
　super 接頭辞 668
　trace 668
　with ステートメント 670
記述、シンタックスとステートメント
　return 681
　switch 681
　リスナー 682
基準点、ロードされたイメージ 307
規則、命名 646
キャスト、オブジェクト 71

キャップのスタイル
設定 489
説明 489
キャップのスタイル、設定 489
旧バージョンの Flash Player、対象にする 699
キューポイント
作成 554
使用 550
操作 553
追跡 551
ナビゲーション、イベント、ActionScript 550
表示 553
共有、フォント
説明 353
強力な型指定 43、47

く

区切子
括弧 92
説明 87
セミコロンおよびコロン 87
中括弧 89
クラス
ASO ファイル 233
Flash でのインスタンスへの割り当て 230
Flash でのカスタムクラスの使用 227
flash.display のクラス 243
flash.external のクラス 243
flash.filters のクラス 244
flash.geom のクラス 245
flash.net のクラス 245
flash.text のクラス 246
getter メソッドおよび setter メソッド 207
getter メソッドと setter メソッドの使用 208
mx.lang のクラス 246
System および TextField のクラス 246
インスタンス化 180
インスタンスの作成 206、229
インスタンス変数 224
インターフェイスとの比較 265
カスタムクラスの記述 188
カスタムクラスの記述例 214
カスタムクラスの操作 191
カプセル化 213
記述のためのベストプラクティス 216
クラス参照の解決 196
クラスパス 193
クラスファイルの作成 197
クラスファイルの命名 217
クラスメンバー 203
クラスメンバーの使用 206
継承 253
継承の例 256
コンストラクタ関数 220
コンパイラによる参照の解決 196
コンパイルと書き出し 231
作成とパッケージ化 217
サブクラスの作成 255
実行時のプロパティの初期化 328
使用する利点 181
スーパークラス 255
スコープ 214、234
スコープの設定 663
静的メソッドと静的プロパティ 201
設計図 183
ダイナミッククラスの作成 211
定義 247
データ型としてのインターフェイス 180
ドキュメンテーション 226
トップレベル 239
パッケージでの編成 181
パッケージとの比較 183
パブリック、プライベート、および静的なメソッドとプロパティ 199
ビルトイン 180
ビルトインオブジェクトメソッドの呼び出し 249
ビルトインクラスの新しいインスタンスの作成 248
ビルトインクラスの使用 247
ビルトインクラスの除外 250
ビルトインクラスの静的メンバー 249
ビルトインとトップレベル 237
ビルトインプロパティの参照 248
プライベートメソッドとプライベートプロパティ 201
プリロード 251
プロパティ 199
ポリモーフィズム 260
ムービークリップへの割り当て 327
メソッドとプロパティ 197
メソッドとプロパティのオーバーライド 258
メソッドとプロパティの記述 221
メンバーアクセスの制御 224
読み込み 192
読み込みとパッケージ化 228

クラスパス
グローバル 195
検索順 196
説明 31
定義 193
ディレクトリの削除 195
ドキュメントレベル 195
クラスファイル
構造化 664
編成のガイドライン 664
クラスメンバー 186、249
説明 186
グラデーショングローフィルタ
使用 458
説明 457
グラデーションベベルフィルタ
blurX と blurY プロパティ 461
knockout と type プロパティ 461
strength プロパティ 461
色分布 462
カラーの配列 462
使用 463
説明 461
適用 465
ハイライト 464
比率と角度 464
比率の配列 462
フィル 461
ムービークリップの塗り 464
ムービークリップへの適用 465
繰り返しアクション、ループの使用 113
グリッドフィットタイプ、使用 364
グレースケールイメージ 421
グローバル変数 58
グローフィルタ
アニメーション化 442
使用 456
説明 456

け

継承
OOP 186
サブクラス 254
説明 253
例 256
結合性、演算子 138
結合配列、説明 132
厳密な型指定 47

こ

子
ノード 587
ムービークリップ、定義 301
効果
イメージのパン 424
グレースケール 421
ノイズ 476
フェード 418
ブレンドモード 479
明度 467
明度トゥイーン 422
明度とカラー 420
効果、「フィルタ」を参照
構成、スクリプト
ActionScript 1.0 と ActionScript 2.0 30
オブジェクトへの割り当て 660
コーディング規則 659
コーディング規則
ActionScript 659
コンポーネント 656
コード
例、コピーとペースト 13
コールバック関数
記述 163
コメント
クラス内部 97
クラスファイル 226
クラスファイルでの記述 658
後続 97
雑然または密集 95
シンタックスの色付け 95
説明 94
単一行 95
複数行 96
ベストプラクティス 657
コロン、説明 87
コンストラクタ関数
記述 166
サンプル 704
コンパイル時、定義 13
コンポーネントのバインディング、ActionScript 527
コンポーネントベースアーキテクチャ、定義 301
コンポーネント、コーディング規則 656

さ

サーバーサイドスクリプト
　XML 形式 588
　言語 570
　作成 585
サーバー、連続的な接続のオープン 593
サーフェス
　定義 318
再生、ムービークリップ 506
サウンド
　「外部メディア」も参照
　タイムラインへの割り当て 514
　バランス制御 516
削除
　ムービークリップ 311
　ロードされた SWF ファイル 304
作成、オブジェクト 248
作成、ストリング 404
サブクラス
　記述 255
　ムービークリップ用の作成 327
　例 256
サブクラス、説明 254
サンプルファイル、説明 14

し

式
　値の処理 135
システム
　イベント、定義 279
　要件、ActionScript 2.0 10
実行順序 (演算子)
　演算子の結合性 138
　演算子の優先順位 138
ジャンプ、URL 507
終点 524
取得、情報、リモートファイル 570
取得、マウスポインタ位置 509
条件
　記述 104
条件演算子 112
条件式 112
条件ステートメント
　記述 678
条件、説明 103
衝突の判定 517
衝突、判定 517
　ムービークリップ間 518
　ムービークリップとステージ上の点の間 517
情報源、その他 14
情報の送信
　TCP/IP 経由 570
　URL エンコード形式 570
　XML 形式 570
　リモートファイル 570
情報、SWF ファイル間での受け渡し 570
使用を避ける Flash 4 演算子 691
初期化、ActionScript の記述 666
初期化、ムービークリッププロパティ 328
処理、数値 39
シングルトンパターン 204
シンタックス
　大文字と小文字の区別 78、79
　スラッシュ 87

す

数値、メソッドを使用した処理 39
スーパークラス 255
スクリプト
　Flash Player 7 への移植 610
　イベントの概要 280
　キーボードイベント 280
　クリップのイベント 281
　フレームスクリプト 281
スクリプトによるアニメーション
　Drawing API 495
　Tween クラス 473
　Tween クラスと TransitionManager クラス 427
　イメージの移動 424
　イメージのパン 424
　オブジェクトの移動 423
　説明 416
　フィルタ 473
　プログレスバーの作成 562
　ぼかしフィルタ 473
　明度トゥイーン 422
スクロール
　テキスト 395
　ビットマップのキャッシュ 426
スコープ
　this キーワード 298
　クラス 663
　説明 85
　ベストプラクティス 661

スタイル
線 489
線とキャップ 489
スタイルシート、「CSS」を参照
スタイル、線とキャップ 489
ステージ、割り当て、シンボル 311
ステートメント
for 680
if 105
if..else 106
if..else if 107
switch 108
try..catch..finally 110、682
while と do while 680
with 670
記述方法に関するガイドライン 102
条件 104、678
説明 102
定義 76
複合 103、679
読み込み 184
ストリング 41
大文字小文字の変換 409
返されるサブストリング 411
型比較の実行 408
作成 404
サブストリングの検索 412
サブストリングの配列の作成 411
使用 404
説明 397
定義 398
長さの特定 406
比較 406
分析 406
変換と連結 409
他のデータ型との比較 408
文字位置の検索 412
文字を調べる 406
ループ処理 407
ストリングパネル 398
スラッシュシンタックス
ActionScript 2.0 は非サポート 87
使用 702
説明 87

せ

静止テキスト 333
静的メンバー 249
静的メンバー、「クラスメンバー」を参照
セキュリティ
Flash Player 7 へのスクリプトの移植 630、636、641
Flash Player 互換性 610
loadPolicyFile 638、639
ドメイン間 629
ドメイン間のデータアクセス 634
ネットワーク API の制限 627
ポリシーファイル 636
接頭辞、super 668
セミコロン、説明 87
線 489
スタイルの設定 489
パラメータの設定 490
線スタイル
capsStyle と jointStyle 492
Drawing API 489
miterLimit 495
pixelHinting 492
アルファ 492
拡大 / 縮小 492
カラー 491
説明 489
線とキャップのスタイル 489
パラメータ 490
太さ 491
線のスタイル 489

そ

操作順序 481
相対パス 86
ソケット接続
サンプルスクリプト 594
説明 593

た

ターゲット化
スコープ 85
ロードされたコンテンツ 83
ターゲットパス
インスタンスのターゲット化 81
使用 167
挿入 86
ドットシンタックス 80
ネストされたインスタンス 82
ボタンの使用 86
ダイナミッククラス 211

ダイナミックテキスト 333
タイムライン変数、説明 59
多次元配列、説明 129
畳み込みフィルタ
　使用 469
　説明 468
　適用 468
タブ文字 405

ち

チェック
　ロードされたデータ 571
中括弧、説明 89

つ

通信、Flash Player 595

て

停止、ムービークリップ 506
定数
　使用 99
　説明 98
　ベストプラクティス 652
データ
　オブジェクトへの分類 69
　説明 33
　定義 33
　変数 33
　ロードとプログレスバー 571
データ型
　MovieClip 37
　null 39
　String 41
　undefined 42
　void 42
　値 185
　オブジェクト 40
　型の判定 46
　基本的 34
　自動割り当て 42
　数値 39
　注釈 47
　定義 34
　ブール値 36
　複合 35
　プリミティブ 35
　変換 34
　割り当て 43
データ、外部 569、609
　LoadVars オブジェクト 576
　SWF でのクロスドメイン間のアクセス 634、638
　XML 586
　XMLSocket オブジェクト 593
　サーバーサイドスクリプト 574
　セキュリティ機能 629
　送信とロード 570
　メッセージ 595
　ロードされた場合のチェック 570
テキスト
　「テキストフィールド」も参照
　イメージの周りに流し込むための <img> タグの使用 386
　実行時のテキストフィールドへの割り当て 335
　スクロール 395
　用語 331
　ロードと表示 342、343、345
テキストコンポーネント 333
テキスト入力 333
テキストのフォーマット
　説明 362
テキストフィールド
　「TextField クラス」、「TextFormat クラス」、「TextField.StyleSheet クラス」も参照
　CSS でのフォーマット 369
　CSS の適用 374
　dynamic 333
　HTML 形式 335
　HTML テキスト 376
　SWF ファイルまたはイメージファイルの埋め込み 391
　位置の変更 339
　イメージサイズの指定 393
　インスタンスと変数名の比較 336
　インスタンス名 336
　埋め込みイメージの周りへのテキストの流し込み 382、386
　埋め込みメディアの制御 393
　外部テキストの埋め込み 343
　クリック可能なイメージの埋め込み 395
　サイズ変更 339
　実行時に動的に作成する 335、337
　処理 338
　深度の設定 352
　説明 333
　デフォルトプロパティ 368
　フォーマット 367

変数のロード 341
変数名のコンフリクトの回避 337
ムービークリップの埋め込み 392
テキストレイアウト 362
テキストレンダリングオプション 356
デザインパターン
カプセル化 213
シングルトン 204
デバイスフォント
定義 356
マスク 326
デバッグ
コンパイルエラーメッセージ 685
テンキー、ASCII キーコード値 696
転送、ムービーとサーバー間の変数 576

と

トゥイーン
ActionScript による追加 430
ビヘイビアの追加 428
等価演算子 147
透明度、マスク 326
ドキュメントオブジェクトモデル (DOM)、XML 587
特殊文字 41
匿名関数
記述 163
使用 165
ドットシンタックス (ドット表記) 81
ドメイン名とセキュリティ 629
トランジション
ActionScript による追加 430
定義 429
ビヘイビアの追加 428
ドロップシャドウフィルタ
clone () メソッド 474
アニメーション化 454
使用 452
説明 452
透明イメージに適用 455

な

9 スライスの拡大 / 縮小
scale9Grid プロパティ 498
使用 499
説明 497
有効化 498
理解 497
ナビゲーション
制御 503
フレームまたはシーンへのジャンプ 505
名前付き関数 163

に

二重引用符、ストリング内 405

ぬ

ヌル (null) データ型 39

ね

ネストしたムービークリップ、定義 301
ネットワーク
ネットワーク API の制限 627
ネットワーク API の制限 627

の

ノイズ効果 476
ノード 587

は

配列
Array コンストラクタを使用した結合配列 134
Object クラス 135
Object を使用した結合配列 133
sortOn() メソッド 177
値の割り当て 50
インデックス付き 128
エレメント 123
エレメントの作成と削除 127
オブジェクトの作成 70
簡易シンタックス 123
簡易シンタックスを使用した作成 51
結合 132
結合配列 132
作成 50
参照渡しによる受け渡し 56
使用 124
説明 123
多次元 129
多次元配列 129
多次元配列による繰り返し処理 131
多次元、for ループの使用 130

長さの参照と検索 127
比喩 123
変更 123、126
例 123、125
配列リテラル 129
バインディング
ActionScript による作成 520
双方向バインディングの作成 522
単方向バインディングの作成 521
パッケージ
クラスとの比較 183
説明 181
操作 183、443
命名 181
読み込み 184
パフォーマンス
ビットマップキャッシュ 426
フィルタ 448
フレームレート 417
パブリッシュ設定
Flash Player のバージョンの選択 31
パラメータ 162
バランス (サウンド)、制御 516
ハンドラ、「イベントハンドラ」を参照

ひ

比較演算子 147
引数
名前付き関数 163
ビットマップキャッシュ
opaqueBackground プロパティ 318
scrollRect 319
アルファチャンネルマスク 326
サーフェス 318
使用すべきでない状況 321
使用する状況 320
説明 318、426
定義 318
フィルタ 445
有効化 318
ビデオ
FLV に合わせたサーバーの設定 561
FLV ファイルの作成 544
FLV ファイルのナビゲーション 556
Macintosh 562
onMetaData ハンドラの使用 559
外部 FLV ファイルの説明 544
キューポイント 550
キューポイントのシーク 557、558
キューポイントの操作 553
キューポイントの追跡 551
シーク機能の追加 556
実行時の FLV ファイル再生 545
指定された長さのシーク 556
説明 543
バナーの作成 546
ビデオオブジェクトの作成 545
プリロード 548
プログレスバーの作成 567
メタデータ 559
ビデオ、読み込みの代替方法 543
非同期アクション 570
一重引用符、ストリング内 405
ビヘイビア
Zoom トランジション 428
描画
コード 481
描画メソッド
「Drawing API」も参照
表記規則 12
ビルトイン関数 161

ふ

ファイル、アップロード 579
ファンクションキー、ASCII キーコード値 696
フィルタ
ActionScript 449
アニメーション化 473
インスタンスへの適用 447
エラー処理 448
回転と傾斜 446
拡大 / 縮小、傾斜、回転 447
グローフィルタ 442
コードによる操作 471
取得と設定 445
定義 442
透明 449
ノイズ 476
配列 472
パッケージの理解 443
パフォーマンス 448
プロパティの調整 472
プロパティの変更 446
明度レベルの変更 467
メモリ使用量 448
メモリ不足エラー 448
ブール値
データ型 36

フェード、オブジェクト 418
フォーマット、テキスト
　使用 363
フォント
　カットオフ値 359
　共有 353
　作成と削除 346
　説明 346
　定義 346
フォントアウトライン 359
フォントシンボル、埋め込み 348
フォントテーブル
　カットオフ値 359
　作成 360
　設定 359
フォントレンダリング
　options 356
　説明 354
　メソッド 355
複合ステートメント 103
　記述 679
複合データ型 (データ値) 35
複合リテラル 93
複製、ムービークリップ 311
プリミティブデータ型 (データ値) 35
フレームスクリプト
　説明 281
フレームレート
　onEnterFrame 417
　Tween クラス 437
　説明 417
　選択 417
ブレンドモード
　説明 478
　適用 479
ブレンドモード、「ブレンドモード」を参照
ブロードキャスターオブジェクト 284
プログレスバー
　Drawing API 572
　コードによる作成 562
　データロード 571
プロパティ
　private 201
　public 200
　static 201
　オブジェクト、アクセス 248
　実行時の初期化 328
　ムービークリップ 307

へ

ベストプラクティス
　ActionScript 1.0 と ActionScript 2.0 30
　インターフェイスの命名 655
　関数 671
　関数とメソッドの命名 653
　クラスとオブジェクトの命名 653
　クラス内のコメント 658
　コーディング規則 646
　コメント 657
　スコープ 661
　定数の命名 652
　パッケージの命名 655
　ブール型の命名 652
　変数の命名 650
ベベルフィルタ
　使用 459
　説明 459
変換、データ型 34
変数
　FlashVars を使用した受け渡し 64
　HTML からの受け渡し 342
　URL エンコード 62
　URL ストリングからの値の受け渡し 62
　URL への送信 507
　XML への変換 589
　値の変更 50
　値の割り当て 49
　アプリケーションでの使用 53
　インスタンス 224
　演算子 51
　外部テキストファイルからのロード 344
　参照渡しによる受け渡し 55
　使用 54
　スコープ 57
　宣言 48
　タイムライン 59
　定義 47
　テキストフィールドへのロード 341
　デフォルト値 49
　名前のコンフリクトの回避 337
　パスを使用した設定 85
　プロジェクトでの使用 67
　未定義と定義済みの比較 53
　ムービークリップとサーバー間の転送 576
　命名規則とガイドライン 52
　ローカル 59
　ロード 61、66
変数、グローバル 58

ほ

ポインタ、「カーソル」を参照
ぼかしフィルタ
　Tween クラスによるアニメーション化 473
　使用とアニメーション 451
　説明 450
ポストコロンシンタックス、定義 43
ポリシーファイル
　「セキュリティ」も参照
　必ず crossdomain.xml という名前にする 637
　定義 636
ポリモーフィズム
　使用 260
　説明 188
ボリューム、スライドコントロールの作成 515

ま

マウス位置、取得 509
マウスポインタ、「カーソル」を参照
マスク 325
　アルファチャンネルマスク 326
　スクリプトで作成 495
　線は無視される 325、481
　デバイスフォント 326
マニュアル、その他のリソース 16

み

未定義 (undefined) データ型 42

む

ムービークリップ
　_root プロパティ 305
　「SWF ファイル」も参照
　background 324
　filters プロパティ 464
　MP3 ファイルのロード 537
　on() ハンドラと onClipEvent() ハンドラの割り当て 289
　SWF ファイルと JPEG ファイルのロード 533
　with ステートメント 304
　インスタンス名、定義 301
　親、定義 301
　開始と停止 506
　カスタムクラスの割り当て 230
　カラーと明度の変更 420
　カラーの調整 513
　関数 303
　キーボードでの起動 511
　共有 311
　グローフィルタの適用 442
　コードによるフェード 418
　子に対するループ 115
　子、定義 301
　再生中のプロパティの変更 307
　削除 311
　サブクラスの作成 327
　実行時の作成 309
　実行時のプロパティの初期化 328
　衝突の判定 517
　深度の確認 316
　深度の管理 314
　ステージ上のシンボルへの割り当て 311
　制御 302
　空インスタンスの作成 310
　次に利用可能な深度の判断 315
　データ型 37
　テキストフィールドへの埋め込み 391
　動的に作成したもののターゲット化 83
　ドラッグ 308
　ネスト、定義 301
　パラメータの追加 313
　複数メソッドの呼び出し 304
　複製 311
　プロパティ 307
　プロパティ、実行時の初期化 328
　ボタン状態の割り当て 293
　マスクの使用 325
　メソッドと関数の比較 302
　メソッドの呼び出し 302
　メソッド、一覧 303
　メソッド、シェイプの描画での使用 481
ムービークリップのドラッグ 308

め

命名規則 646
　インターフェイス 655
　関数とメソッド 653
　クラスとオブジェクト 653
　パッケージ 181、655
　ブール (Boolean) 652
　変数 650
命名、インターフェイス、ベストプラクティス 655
命名、クラスとオブジェクト、ベストプラクティス 653
命名、パッケージ、ベストプラクティス 655

メソッド
private 201
public 200
static 201
オブジェクト、呼び出し 249
関数との比較 178
種類 159
説明 157、176
定義 176
配列 176
非同期 570
ムービークリップの制御 303
命名 178
メソッド呼び出し 37
メタデータ
使用 559
説明 559
メッセージボックス、表示 596
メモリ不足エラー 448
メンバー (メソッドとプロパティ)
パブリック、プライベート、および静的 199

も

文字
埋め込みの作成と削除 346
文字エンコーディング 397
文字セット
カスタムセットの作成 351
文字の埋め込みダイアログボックス
使用 350
文字の連続、「ストリング」を参照

ゆ

ユーザーイベント 279
ユーザー定義関数
記述 167

よ

読み込み
クラスファイル 192
予約語
説明 101
その他の推奨事項 102

ら

ランタイムデータバインディング
CheckBox 523
説明 520
双方向バインディングの作成 522
ランタイム、定義 13

り

リスナーオブジェクト 284
登録解除 285
リスナーシンタックス 682
リテラル
説明 93
複合 93
リモート
サイト、連続的な接続 593
ファイル、接続 570
リンク、ムービークリップ 311
リンケージ
identifier 311、327
コーディング規則 656
リンケージプロパティダイアログボックス 311、327

る

ループ
do..while 121
for 117
for..in 118
while 120
作成と終了 116
使用 113
ネスト 122
ループステートメント 114、115

れ

レベル
ロード 304
レベル、深度の識別 84
連結、ストリング 41
連続的な線変調 356

ろ

ローカル変数、説明 59
ロード
　XML ファイルの表示 345
　外部メディア 532
ロードされた SWF ファイル
　削除 304
　識別 84
ロードされたデータ、確認 571
ロード、データ
　サーバー 66
　変数 67

わ

割り当て、サウンド 514