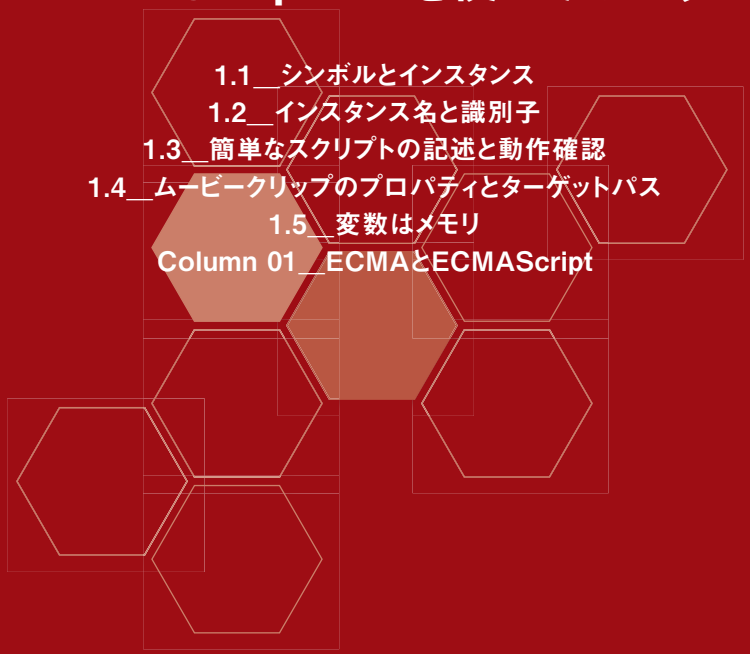
Chapter 1
ActionScript 3.0を使ってみよう



1.1 シンボルとインスタンス

Flashでは、ステージ上にさまざまな素材を配置して、ムービーを作成します。［ツール］パネルのツールを使って作成するシェイプ（Shape）やテキストフィールド（TextField）、外部画像を読込んだビットマップ（Bitmap）、それらを用いて作成するシンボルなど、Flashムービーで使われる目に見える素材を「ビジュアルエレメント」と呼びます。それらビジュアルエレメントの中で、ActionScriptにかぎらず、Flashで重要な役割を果たすのが「シンボル」です。まずは、ムービークリップシンボルを、ひとつつくってみましょう。

なお、新規Flashムービー（FLA）ファイルを作成する際には、［新規ドキュメント］ダイアログボックスで必ず［Flashファイル（AS 3.0）］を選んでください（図01-001）。この選択により、新規ファイルの［パブリッシュ設定］は、［ActionScriptのバージョン］が［ActionScript 3.0］に設定されます。

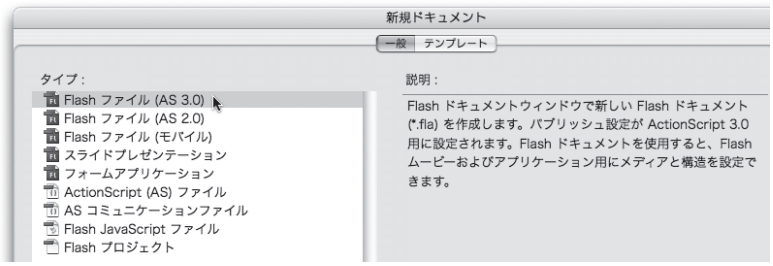


図01-001■［新規ドキュメント］ダイアログボックスで［Flashファイル（AS 3.0）］を選ぶ
［パブリッシュ設定］の［ActionScriptのバージョン］が［ActionScript 3.0］に設定され、ActionScript 3.0が使用可能になる。

1.1.1 シンボルの作成

ステージ上に、何かビジュアルエレメント、たとえば矩形や楕円のシェイプを描画・配置します。そのビジュアルエレメントを［選択ツール］で選び、右クリックして［シンボルに変換］を選ぶか、ショートカットキー［F8］を押します。［シンボルに変換］のダイアログボックスが開いたら、［タイプ］は［ムービークリップ］を選びます（図01-002）。シンボルの［名前］は、自分で管理しやすい名前をつけます。シンボル名は、入力が必須です。けれど、スクリプトのコントロール対象にはなりません。

図01-002■ビジュアルエレメントを［シンボルに変換］してムービークリップにする
［タイプ］は［ムービークリップ］を選び、任意の［名前］を入力する。

Tips 01-001 ■スクリプトの操作対象にならない名前

スクリプトはSWFにパブリッシュされて、Flash Player上で動作します。しかし、シンボルの［名前］は、SWFに書出しされません。したがって、スクリプトでコントロールする対象にはならないのです。ほかには、レイヤー名もSWFには書出されず、スクリプトで操作することはできません。

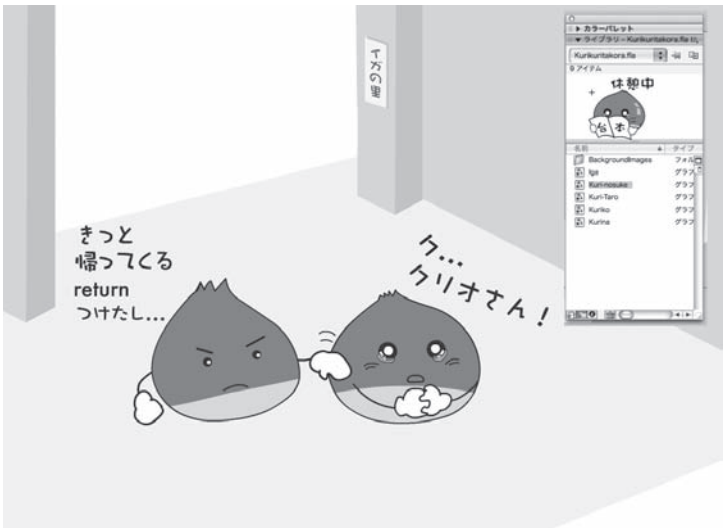
その代わり、名前のつけ方は自由で、漢字などの全角やスペースも使えます。スクリプトで操作するための名前（後掲Word01-001「識別子」参照）のような命名の決まりはとくにありません。

1.1.2　シンボルは役者でインスタンスは配役

作成したシンボルは、［ライブラリ］パネルに格納されます（図01-003）。シンボルがステージに配置されると、それは「シンボルのインスタンス」あるいは単に「インスタンス」と呼ばれます。シンボルと、そのインスタンスとの区別は重要です。ステージを「舞台」と考えれば、シンボルは「役者」に当たります。［ライブラリ］は、いわば役者の控え室です。役者は舞台のうえでは、「配役」を与えられます。この配役・キャストがインスタンスになるのです。

図01-003 ■シンボルとシンボルのインスタンス

シンボルからインスタンスをいくつつくっても、グラフィックなどのおもなデータはシンボルひとつ分で済む。



「ステージ」は舞台で「シンボル」が役者、「インスタンス」はキャストになる。

ひとつのシンボルから、インスタンスはいくつでもつくれます。同じフレームに複数同時に配置することもできますし、別のフレームで何度でも使い回すことが可能です。そのとき、もっともデータサイズを費やすグラフィックデータは、シンボルひとつ分しか要しません。ですから、同じシンボルを再利用すれば、データサイズの節約になるのです。いってみれば、ひとりの役者に何役演じてもらっても、ギャラはひとり分で済むようなものです。

1.1.3　シンボルとインスタンスのタイプ

シンボルには、[ムービークリップ] と [ボタン]、それに [グラフィック] の３つの [タイプ] があります（図01-004）。それぞれのタイプによって、ステージ上のアニメーションの動作や、ActionScriptにおける扱いが異なります。シンボルの [タイプ] を確認・変更するには、[ライブラリ] パネルでシンボルを選んで、右クリックしてコンテキストメニュー、または [ライブラリ] パネルのオプションポップアップメニューから [プロパティ] を選択するか、もしくは（プロパティ）ボタンをクリックして、[シンボルプロパティ] ダイアログボックスを開きます。

図01-004■シンボルの３つのタイプ
[ライブラリ] パネルでシンボルを選んで、右クリックしてコンテキストメニュー、またはパネルのオプションメニューから [プロパティ] を選択。あるいは、[ライブラリ] パネル下部の（プロパティ）ボタンをクリックする。

［ムービークリップ］や［ボタン］シンボルのインスタンスは、スクリプトでコントロールすることが可能です。それに対して、［グラフィック］シンボルのインスタンスは、ActionScriptで操作できません。役者というより、大道具・小道具あるいは衣装のような扱いです。［グラフィック］インスタンスをスクリプトでコントロールしたいときは、通常［ムービークリップ］シンボルの中に入れ子にして、その［ムービークリップ］ごと操作します。

［ライブラリ］から［ムービークリップ］シンボルをステージにドラッグ＆ドロップすれば、［ムービークリップ］インスタンスが配置されます。同様に、［ボタン］シンボルをドラッグ＆ドロップすると、［ボタン］インスタンスができあがります。しかし、これらインスタンスの［タイプ］は、［プロパティ］インスペクタで変更することが可能です（図01-005）。シンボルの［タイプ］が何であっても、Flashムービー上のふるまいはインスタンスの［タイプ］によって決まります。

図01-005■インスタンスの［タイプ］は［プロパティ］インスペクタで変更可能
シンボルの［タイプ］にかかわらず、ムービー上のふるまいはインスタンスの［タイプ］で決まる。

シンボルの［タイプ］は、役者の性別のようなものでしょう。普通、男優は男役をやり、女優は女役を演じます。けれども、そうでない舞台はあります。宝塚では女性が男役をやり、歌舞伎の女形は男性です。そして、舞台上の物語は、あくまで配役の設定を前提として展開するのです。

1.2 インスタンス名と識別子

スクリプト（script）には、英語で「脚本」という意味があります。脚本は、役者に対して書くものではありません。キャスト（配役）のセリフや演技を指示するものです。ActionScriptも、原則としてインスタンスをコントロールするものです。キャストに指示するには、役名が必要です。たとえそれが単なる通行人であったとしても、「通行人A」という名前がなければセリフも演技も指示できません。同じように、インスタンスには「インスタンス名」を設定する必要があります。

1.2.1 識別子とは

インスタンスには、「識別子」（しきべつし）と呼ばれる以下のWord 01-001のような決まりで名前をつけなければなりません。識別子といういかめしい用語は、英語ではidentifierといいます。identifierは、語源が「ID」と同じで、自分が何者であるかを示すものという意味です。それは要するに「名前」で、プログラムでコントロールするための名前が「識別子」なのです。したがって、識別子はインスタンス名だけでなく、後で述べる変数や関数（function）、その他ActionScriptで制御するすべての名前に用いられます。

Word 01-001 ■識別子

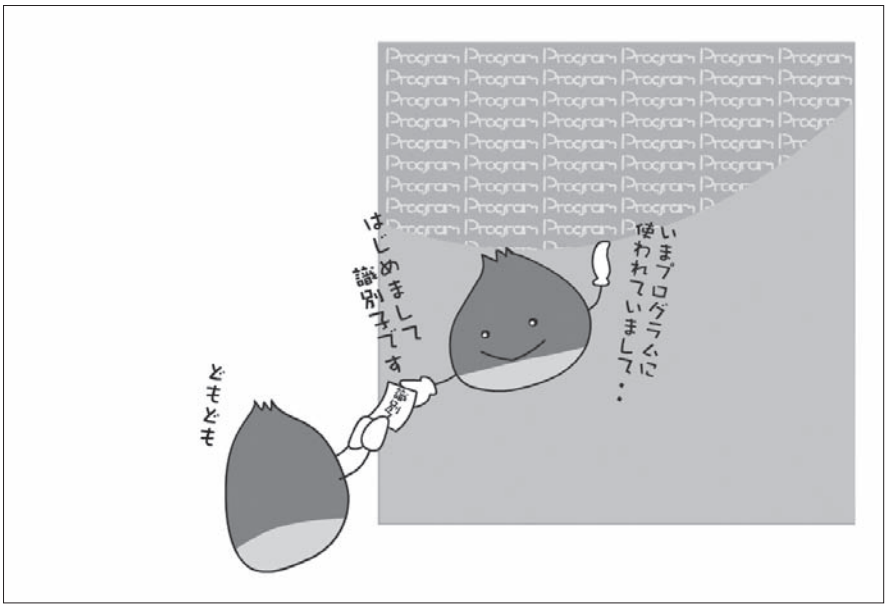
プログラムでコントロールするインスタンスや変数、関数などに対してつける名前です。以下の4つの決まりがあります。

1. **すべて半角で、英数字か「_」（アンダースコア）または「$」（ドル記号）のみを用います。ただし、「$」は通常は使われません。**
2. **識別子の最初の文字に、数字を使うことはできません。**
3. **英字の大文字小文字は区別されます。**
4. **ActionScriptで現在使用され、あるいは将来使われる可能性のあるキーワードとして定められている予約語を使うことはできません。**

識別子は第1に、半角の英数字を用います。他に記号としては、半角の「_」（アンダースコア）が使えます。もうひとつだけ「$」（ドル記号）も使用可能ですが、通常使われませんし、その必要性も少ないでしょう。「-」（マイナス）などの記号やスペースも使えませんので注意しましょう。

Maniac! 01-001 ■識別子への「$」の使用

ActionScriptが準拠するECMAScript（後述Column 01「ECMAとECMAScript」参照）の仕様には、「ドル記号はもっぱら機械的に生成されるコード中で使用されることを目的とする」（"The dollar sign is intended for use only in mechanically generated code."）とあります（ECMA-262第3版「7.6 Identifiers」）。つまり、プログラムの内部処理で使われるものなので、ユーザーが「$」を使用することは必ずしも意図されていないと考えられます。



「識別子」?! 難しそうに響くけど、要はプログラムで使われる「名前」。

第2に、識別子の最初の文字には、数字が使えません。ですから、たとえば「1-001 Start」というインスタンス名は、3つ問題があります。(1) 最初に数字「1」が使われており、(2)「-」(マイナス) や (3)「 」(半角スペース) という識別子に使えない文字が含まれているからです。この場合、たとえば「no1_001_Start」という名前に変更すればよいでしょう。

第3に、識別子に用いた英字の大文字小文字は区別されます。たとえば、ムービークリップシンボルのインスタンスに「my_mc」(すべて小文字) と名前をつけておいて、スクリプトから「My_mc」(頭が大文字) や「MY_MC」(すべて大文字) を制御しようとしても、これらの識別子はすべて別物として認識されるため、正しくコントロールすることができません。大文字小文字の違いも含めて、識別子は正確に設定・記述する必要があります。

AS1&2 Note 01-001 ■大文字小文字の区別

ActionScript 1.0/2.0 の場合、Flash Player 7以降で書出すと、識別子の大文字小文字が区別されます。Flash Player 6以前の書出しでは、大文字小文字の違いは認識されません。もっとも、大文字小文字を正確に区別しない識別子が使われたスクリプトは、大変読みにくいものになります。したがって、Flash Player 6以前であっても、大文字小文字は区別して書くことをお勧めします。

第4に、ActionScriptですでに使われているキーワードは、識別子として使用することができません。たとえば、「if」という名前をインスタンスにつけようとすると、それは予約されている旨の警告ダイアログが表示されて、設定できません (図01-006)。このようなキーワードを「予約語」と呼びます (後述 Tips 02-009「キーワードと予約語」参照)。

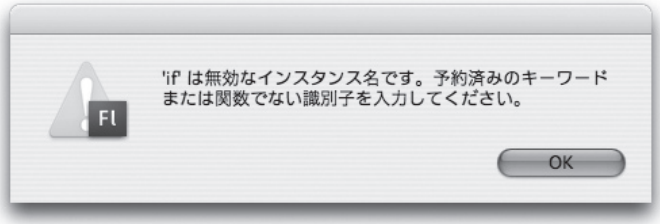


図 01-006 ■インスタンス名に予約語を入力したときの警告ダイアログ
予約語であっても警告ダイアログが表示されないものもある。

予約語には、ActionScriptで現在使用されていなくても、将来使われる可能性があるキーワードも含まれます。また、ActionScriptに定義されているすべてのプロパティや命令（関数・メソッド）の名前が、予約語になっている訳ではありません。ActionScript定義済みの名前であっても、スクリプトで使用して問題なく動作するものもあります。とはいえ、ActionScriptに定義済みの名前は、基本的に使うのは避けた方が無難でしょう。予約語については、また次Chapter2.2「変数を使う」で、もう少し詳しく説明します。

1.2.2　インスタンス名の設定

インスタンス名を設定するには、まず［選択ツール］で対象となるMovieClipインスタンスをクリックし、［プロパティ］インスペクタの［インスタンス名］のテキストボックスに識別子を入力します（図01-007）。ここでは、メインタイムラインに配置したムービークリップシンボルのインスタンスに、たとえば「my_mc」というインスタンス名を設定しておきましょう。なお、メインタイムラインは、新規のムービーにデフォルトで［シーン 1］と表示されるタイムラインです。

図 01-007 ■［プロパティ］インスペクタでインスタンス名を設定
［インスタンス名］のテキストボックスに識別子を入力する。

Tips 01-002 ■タイムラインとシーン
Flashでは、［シーン 1］と書かれたメインのタイムラインにムービークリップインスタンスを配置し、さらにそのシンボル内に別のムービークリップを幾重にも入れ子にすることができます。ムービークリップもまた独自のタイムラインをもちますので、Flash（SWF）ムービーはタイムラインの階層構造になっているということができます。メインタイムラインは、その階層の最上位にあります。
シーンは、［シーン］パネルを使って名前を変えたり、別のシーンを追加してそれらの再生順序を入替えることも可能です。ただし、［パブリッシュ］すると、すべてのシーンがその再生順序にしたがってつなげられ、1本のメインタイムラインとして書出されます。

1.3 簡単なスクリプトの記述と動作確認

スクリプトは、タイムラインのキーフレームに記述します。タイムラインというのは、メインタイムラインだけではありません。ムービークリップシンボル内のタイムラインも含みます。ですから、スクリプトの記述場所は、メインタイムライン以外に、ムービークリップインスタンスの数だけ存在します。今回は、ムービークリップインスタンスを配置したメインタイムラインにスクリプトを書くことにします。

1.3.1 フレームアクション

スクリプトを示すとき、どのタイムラインのどのフレームに記述するのかという情報は重要です。どのタイムラインかというのは、メインタイムラインなのかムービークリップなのか、そしてムービークリップの場合にはどのインスタンスなのかを特定する必要があるということです。どのフレームかというのは、キーフレームのフレーム番号の情報です。

キーフレームに記述したスクリプトを、「フレームアクション」と呼びます。ですから、たとえば「メインタイムラインの第1フレームアクション」とか、「ムービークリップインスタンスmy_mcの第10フレームアクション」と表現すれば、記述場所が明らかになります。

AS1&2 Note 01-002 ■スクリプトの記述場所
ActionScript 1.0/2.0ではキーフレームの他に、ムービークリップやボタンのインスタンスを選択して、インスタンスにスクリプトを記述することができます。ActionScript 3.0では、FLAムービー内のスクリプトは、キーフレームにしか書けません。

キーフレームでさえあれば、スクリプトはどこにでも記述できます。ただ、あちこち無頓着にスクリプトを書くと、後でわからなくなります。そこで、スクリプトのための専用レイヤーを用意した方がよいでしょう（図01-008）。その際、せっかくつくった専用レイヤーにうっかりビジュアルエレメントを置いてしまわないように、レイヤーはロックしておくと安心です。レイヤーはロックしてあっても、スクリプトを記述することは可能です。

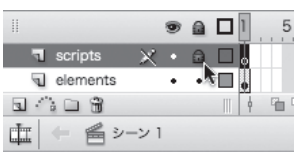


図01-008 ■スクリプト専用レイヤーを作成
スクリプト専用レイヤーは、ロックしておくと安心だ。

Tips 01-003 ■専用レイヤーの作成
スクリプトやフレームラベル、サウンドなど、ステージ上に表示されないエレメントや素材は、専用レイヤーを作成して、それぞれ別個に管理することをお勧めします。

1.3.2　プロパティの設定

Flashアプリケーションを使って、ムービーを作成する作業は「オーサリング」と呼ばれます。オーサリング時に、ムービークリップシンボルのインスタンスに数値で座標を指定するには、[プロパティ] インスペクタか [情報] パネルのxまたはy座標のテキストボックスに値を入力します（図01-009）。

図01-009■［プロパティ］インスペクタで座標値を設定
xy座標のテキストボックスに値を入力する。

この操作をスクリプトで処理するには、ムービークリップインスタンスのプロパティに数値を設定します。

まず、メインタイムラインに作成したスクリプト専用レイヤーのキーフレームを選択します。現在はフレームはひとつのみで、第1フレームはつねにキーフレームです。ですから、とくにキーフレームを作成する操作は要りません。つぎに、［ウィンドウ］メニューから［アクション］を選択して、［アクション］パネルを表示します。

Tips 01-004 ■［アクション］パネル表示のショートカット

［アクション］パネルの表示はよく使う操作ですので、ショートカットを覚えると便利です。

ショートカットキーはWindowsが［F9］、Macintoshは［option］+［F9］です。キーフレームを右クリック（右クリックの使えないMacintoshでは［control］+クリック）しても、［アクション］が選択できます。さらに、Windowsは［Alt］、Macintoshは［option］キーを押しながらキーフレームをダブルクリックしても、［アクション］パネルが開きます。

Mac OS Xでは［F9］がExposéのショートカットで使われているため、［アクション］パネルのショートカットは［option］+［F9］に設定されています。Mac OS Xの［システム環境設定］から［キーボードとマウス］を選択すれば、［キーボードショートカット］でExposéのショートカットは変更できます。

また、Flashアプリケーションの側は、Windowsは［編集］メニュー、Macintochは［Flash］アプリケーションメニューの［キーボードショートカット］で、［アクション］パネルその他のショートカットが変えられます。

スクリプトを書く前には、必ず記述場所が正しく選ばれていることを確認しましょう。書き場所を間違うと、エラーが発生したり、スクリプトが意図した動作をしなくなります。今回は、メインタイムラインのスクリプト専用レイヤー第1フレームを選択しておきます。そして、キーボードでスクリプト01-001のとおり1行の命令文を入力します。スクリプトを設定したキーフレームには、小さいa記号が表示されます（図01-010）。

スクリプト01-001■メインタイムラインのスクリプト専用レイヤーのキーフレームに記述するスクリプト

```
my_mc.x = 100;
```

スクリプトは、半角で入力します。「=」記号の前後の半角スペースは、あってもなくても問題はありません。半角スペースは、いくつ挿入しても、スクリプト内では無視されるからです。ただし、全角スペースは、スクリプトには使えません。テキストとして扱う場合以外に全角文字を用いれば、シンタックス（文法）エラー（後述Word 01-002「シンタックス」参照）になります。半角と全角のスペースは、一見してすぐに区別がつきにくいので、間違えて入力しないように注意しましょう。

図01-010■キーフレームにステートメントを記述
スクリプトを記述したキーフレームにはa記号が表示される。

Tips 01-005 ■不可視文字の表示

［アクション］パネルのオプションメニューで［隠し文字］をオンに設定すると、全角と半角のスペースや改行コードが識別できるように表示されます（図01-011）。

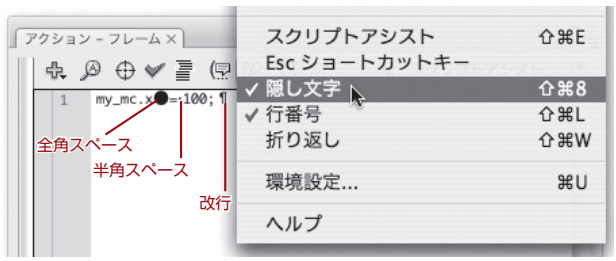


図01-011■［アクション］パネルの［隠し文字］をオンに設定
全角スペースや半角スペース、改行などが異なった記号で表示される。

命令文を「ステートメント」と呼びます。スクリプト01-001のステートメントは、スクリプトを書いたメインタイムラインに配置されているムービークリップインスタンスmy_mcの、水平座標のプロパティxに数値100を設定します。

my_mc.xの「x」は、ムービークリップインスタンスのx座標をコントロールするプロパティです。インスタンス名my_mcに続く「.」（ドット）は、インスタンスmy_mcがドットの後に指定するプロパティxの操作対象（ターゲット）であることを示します。つまり、「my_mc.x」は、ムービークリップインスタンスmy_mcのxプロパティをコントロールするという意味です。

「=」記号は、右辺の値を左辺のプロパティに設定する処理です。この処理をプログラミングでは、「代入」と呼びます。行の最後の「;」（セミコロン）は、ひとつのステートメントの終了を意味します。

【プロパティへの値の代入】
操作対象インスタンス . プロパティ　=　値 ;

スクリプト 01-001 の 1 行のステートメントは、オーサリング時に［プロパティ］インスペクタでインスタンス my_mc を選んで、x 座標のテキストボックスに 100 を入力した処理に相当します。

Tips 01-006 ■基準点と変形点

厳密には、スクリプトによる座標の操作は、基準点（＋印）を対象とします（図 01-012）。それに対して、オーサリング時の座標は、［情報］パネルの座標グリッドの設定により、変形点（○印）かシンボルの基準点のいずれかが原点になります。

図 01-012 ■ MovieClip インスタンスの基準点と変形点
変形点はデフォルトでインスタンスの中央に設定される。

変形点は、デフォルトでインスタンスの中心に設定されます。変形点を後から基準点の位置に設定するには、［自由変形ツール］で変形点をダブルクリックします。

AS1&2Note 01-003 ■プロパティ名先頭のアンダースコア（_）

ActionScript 3.0 のプロパティには、名前の先頭にアンダースコア「_」がつきません。ですから、ActionScript 1.0/2.0 の _root やムービークリップインスタンスの _x プロパティは、それぞれ root および x に変わります。
また一部には、アンダースコア（「_」）を取去るだけではなく、名称が新たになったプロパティもあります。ムービークリップの _xmouse や _xscale プロパティがそれで、それぞれ mouseX と scaleX プロパティに名前が変わります（表 01-001）。

表 01-001　ActionScript 1.0/2.0 から ActionScript 3.0 へのプロパティ名変更の例

変更方法	ActionScript 1.0/2.0	ActionScript 3.0
「_」削除	_root、_alpha、_height、_parent、_rotation、_visible、_width、_x、_y	root、alpha、height、parent、rotation、visible、width、x、y
名称変更	_xmouse、_xscale、_ymouse、_yscale	mouseX、scaleX、mouseY、scaleY

さらに、プロパティの取りうる値についても、注意が必要です。ActionScript 3.0 では、2.0/1.0 でパーセンテージを単位とする多くのプロパティが、小数値を取るよう改められました。上記表 01-001 では、alpha や scaleX/scaleY プロパティは値が、従来のパーセンテージの 100% を 1.0 とする小数値に変更されました。

1.3.3　スクリプトの動作確認

スクリプトはオーサリング環境でなく、SWFに書出されて、Flash Player上で動作します。Flashアプリーションから動作を確認する方法は、ふたつあります。

もっとも簡単な動作確認の方法は、［制御］メニューから［ムービープレビュー］を実行することです。ショートカットキーは、Windowsが［Ctrl］＋［Enter］、Macintoshは［command］＋［return］です。すると、FlashはSWFを書出したうえで、それをFlashアプリケーションに同梱されているFlash Playerで再生します（図01-013）。［ムービープレビュー］では、ブラウザを開きません。

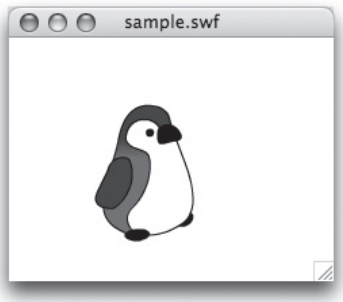


図01-013■［ムービープレビュー］で表示したSWF
ブラウザを起動せずFlashアプリケーション内のFlash Playerで再生する。

もうひとつの動作確認方法は、［ファイル］メニューから［パブリッシュプレビュー］の［デフォルト －（HTML）］（ショートカットはWindowsが［Ctrl］＋［F12］、Macintoshは［command］＋［F12］）を選択することです。FlashアプリケーションはSWFとHTMLファイルを書出して、SWFの埋込まれたHTMLをブラウザで開きます。

ただ、ローカルでの動作確認は、あくまで簡易な方法です。サイトで閲覧するコンテンツは、最終的には必ずサーバーにアップロードして確認しましょう。SWFを再生する必要はなく、SWFとHTMLファイルを書出すだけでよい場合には、［ファイル］メニューから［パブリッシュ］を行っても結構です。

Tips 01-007 ■ローカルとサーバーの動作の違い
サーバーでは、ローカルと比べてSWFや外部ファイルを読込むのに時間がかかります。ロード待ちが必要な処理は、ローカルとサーバーとでは、動作結果が異なることは少なくありません。また、ファイル名の大文字小文字の違いは、ローカルでは認識されないものの、通常サーバーでは区別されます。さらに、Flash Playerのセキュリティによる制限は、ローカルとサーバーとで仕様が違います。

1.3.4　プレビューの前に［シンタックスチェック］

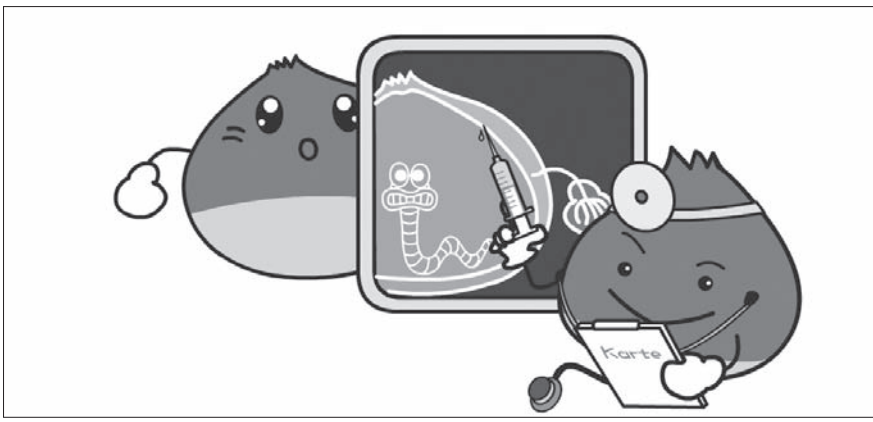
実際にムービーをつくり込んでフレーム数が増え、スクリプトの記述されたキーフレームも数多くなってくると、エラーが発生した場合どのフレームのスクリプトに問題があるのかすぐに見つけにくくなります。とくに、初心者にはエラーの発見は難しいものです。ですから、SWFを書出す前に、必ずスクリプトのチェックをするようお勧めします。

スクリプトは、［アクション］パネルの［シンタックスチェック］ボタンで文法チェックが行えます（図01-014）。ショートカットキーは、Windowsが［Ctrl］+T、Macintoshは［command］+Tです。

Word 01-002 ■シンタックス
英語のsyntaxは、構文あるいは文法を意味します。プログラミング言語の文法や記述の規則を、シンタックスといいます。文法や規則にしたがわないために発生するエラーが、シンタックスエラーです。

図01-014■［アクション］パネルの［シンタックスチェック］ボタンで構文チェック
ショートカットキーは［Ctrl］+T（Windows）または［command］+T（Macintosh）。

［ムービープレビュー］や［パブリッシュプレビュー］を行う前に［シンタックスチェック］をしておくと、エラーが生じたときにも、その記述中のフレームアクションが問題だとすぐに特定できます。スクリプトのエラーも、人間の病気と同じく早期発見・早期治療が大切です。



スクリプトのエラーも早期発見・早期治療が大切だ。

［シンタックスチェック］は、あくまで構文・文法上のテストを行うだけです。たとえば、ステートメント終了の記号「;」（セミコロン）を間違えて、「:」（コロン）を入力した場合、［シンタックスチェック］でエラーが表示されます。しかし、ムービークリップインスタンスのプロパティ「y」を誤って「z」と記入しても、「z」という名前のプロパティはないにもかかわらず、エラーにはなりません。

1.4 ムービークリップのプロパティとターゲットパス

つぎは、メインタイムラインに複数のムービークリップインスタンスを置いて、コントロールしてみます。車のボディのムービークリップシンボルCarと、タイヤのムービークリップシンボルTireを作成しました。

1.4.1 タイムラインに直接配置したインスタンスをコントロール

Carのインスタンスcar_mcをひとつと、Tireのインスタンスをふたつ、tire0_mcとtire1_mcというインスタンス名にして、メインタイムラインに配置しました（図01-015）。

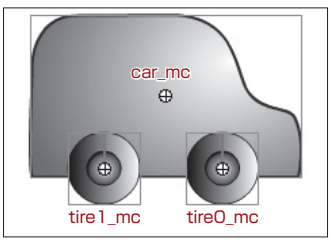


図01-015■メインタイムラインに3つのムービークリップインスタンスを配置
ムービークリップシンボルCarのインスタンスにはcar_mc、Tireのふたつのインスタンスにはtire0_mcとtire1_mcというインスタンス名を設定する。

メインタイムラインの第1フレームアクションで、これらのインスタンスをスクリプトにより操作します。ムービークリップインスタンスcar_mcを100ピクセル右に移動し、インスタンスtire0_mcを90度回転してみましょう（tire1_mcはひとまずおきます）。水平座標のプロパティはxでした。回転角はrotationプロパティで設定できます。そして、プロパティのコントロールには、ターゲット（操作対象）となるインスタンスの指定が必要でした。すると、メインタイムラインの第1フレームアクションは、つぎのようになります（スクリプト01-002および図01-016）。

スクリプト01-002■メインタイムラインの第1フレームアクション

```
car_mc.x += 100;
tire0_mc.rotation += 90;
```

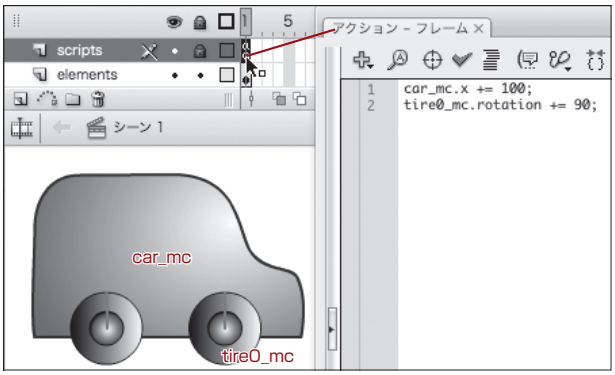


図 01-016 ■メインタイムラインの第1フレームにスクリプトを記述
+= は、現在値に値を加算する演算子。

+= は、左辺の現在値に右辺の値を加算する演算記号（演算子）で、「加算後代入演算子」と呼びます。したがって、ムービークリップインスタンス car_mc は右に 100 ピクセル移動し、インスタンス tire0_mc は時計方向に 90 度回転します。ただし、3 つのムービークリプインスタンスはメインタイムラインにバラバラに（独立して）置かれていますので、ボディのインスタンス car_mc が水平移動しても、タイヤの tire0_mc および tier1_mc はその場を動きません（図 01-017）。

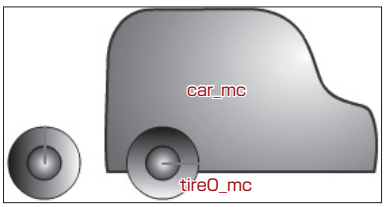


図 01-017 ■水平移動したボディにタイヤは置き去り
car_mc は水平移動し、tire0_mc はその場で回転する。

Word 01-003 ■演算子
プログラミングで使われる演算記号のことを、「演算子」（operator）と呼びます。四則演算の +、-、*、/ や代入の = などが代表的です。しかし、それだけでなく、たとえば数学の演算記号にはない % （剰余）や、見た目は英単語であっても演算子に分類される typeof など、さまざまなものがあります。

Tips 01-008 ■代入の演算子

加算後代入演算子 += を使ったステートメントは、通常の代入の演算子 = を用いた記述に書替えることができます。つぎのふたつのステートメントは、どちらもプロパティ x の値に 10 加算する処理になります。

```
x += 10
x = x + 10
```

数学好きな人は、後者が方程式のように見えて、違和感を覚えるかもしれません。右辺の x はプロパティの現在値で、左辺は新しい値になりますので、両辺の変数が同じ値を意味する方程式とは異なります。そういう人は、数列の式を想い起こすとよいでしょう。

$$x_n = x_{n-1} + 10$$

現在の値に算術演算を施して、新しい値として代入する演算子はまとめて「算術複合代入演算子」と呼ばれます。加算の += のほか、減算 -= や乗算 *=、除算 /=、剰余 %= などがあります。

1.4.2 ムービークリップシンボル内に入れ子にしたインスタンスをコントロール

ボディとタイヤを一緒に動かすには、タイヤの方にも水平移動のステートメントを加えるのも一法でしょう。けれど、Flash には、シンボルを入れ子にできるという便利な機能があります。車のムービークリップシンボル Car の中にタイヤのインスタンス tire0_mc と tire1_mc を入れ子にしてしまえば、インスタンス car_mc と一緒にふたつのタイヤも移動します（図 01-018）。

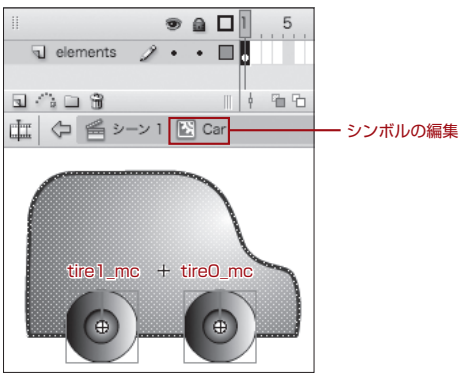


図 01-018 ■ボディのムービークリップシンボル内にタイヤのインスタンスを配置
ムービークリップシンボル Car を編集して、その中にムービークリップインスタンス tire0_mc と tire1_mc を置く。

しかし、もとのスクリプト 01-002 のままでは、車のボディ（car_mc）とタイヤ（tire0_mc と tire1_mc）は一緒に水平移動するものの、タイヤ（tire0_mc）が回転しなくなります。そして、つぎのようなエラーメッセージが［出力］されます。

```
1120: 未定義のプロパティ tire0_mc へのアクセスです。
```

これは、インスタンスtire0_mcがcar_mcのシンボル内に移動したため、rotationプロパティのターゲット指定が正しくなくなったからです。ちょうどHTMLドキュメントにハイパーリンク（<a>タグのhref属性）を設定した後、リンク先のファイルを別のフォルダに移動した状態に似ています。いわば「リンク切れ」が起こってしまったのです。

ムービークリップインスタンスcar_mcの中に配置したインスタンスtire0_mcを指定するターゲットは、car_mc.tire0_mcと記述します。したがって、スクリプト01-002をつぎのように書替えれば、car_mcと一緒に挿入・移動したtire0_mcが回転します（スクリプト01-003）。

スクリプト01-003 ■ターゲットを修正したメインタイムラインの第1フレームアクション

```
car_mc.x += 100;
car_mc.tire0_mc.rotation += 90;
```

入れ子になったムービークリップインスタンスをターゲット指定するには、親のタイムラインからその子へと、インスタンス名を順にドット（.）で結んで記述します。これはHTMLでURLを指定するときに、フォルダ（ディレクトリ）の階層を順にスラッシュ（/）で結ぶ、パスの指定と同じ考え方です。ターゲットのインスタンスもドット（.）区切りのパスで記述するので、「ターゲットパス」の指定と呼ばれることもあります。パスの起点は、スクリプトを記述しているタイムラインです。

Tips 01-009 ■インスタンスは親のプロパティ

スクリプトを記述しているタイムラインに配置したインスタンスmy_mcの水平座標は、プロパティxのターゲットを指定して、my_mc.xでコントロールします。プロパティ（property）には「所有物」という意味があります。ですから、プロパティxは、ムービークリップインスタンスmy_mcの持ち物なのです。

他方、my_mcのシンボル内に配置した子のインスタンスchild_mcのターゲットパスは、my_mc.child_mcで指定します。これは見方を変えると、インスタンスchild_mcがmy_mcの持ち物すなわちプロパティであることを示しています。つまり、タイムラインに配置された子のインスタンスは、ActionScriptでは親のタイムラインのプロパティとして扱われるのです。

したがって、スクリプト01-003の2行目のステートメントは、インスタンスcar_mcのプロパティのtire0_mcのxプロパティに、値90を加算するという意味になります。ところで、car_mcの前には、ターゲットの指定がありません。この場合、スクリプトを書いているタイムラインが、自動的にターゲットとして認識されます。スクリプト01-003はメインタイムラインに記述していますので、インスタンスcar_mcはメインタイムラインのプロパティだということになります。

1.5 変数はメモリ

スクリプトを使った、ごく簡単なアニメーションを作成してみましょう。ActionScript 3.0のスタンダードなアニメーションのスクリプティングは、つぎのChapter2「スクリプトによるアニメーション」で解説します。ここでは、Flash 4から使われている、タイムラインを用いた手法で実現します。

1.5.1 タイムラインを使ったアニメーション

ムービークリップインスタンスが配置されたレイヤーを、12フレームに拡げてみます。そのためには、第12フレームを選択して、［フレームを挿入］（ショートカット［F5］）します（図01-019）。

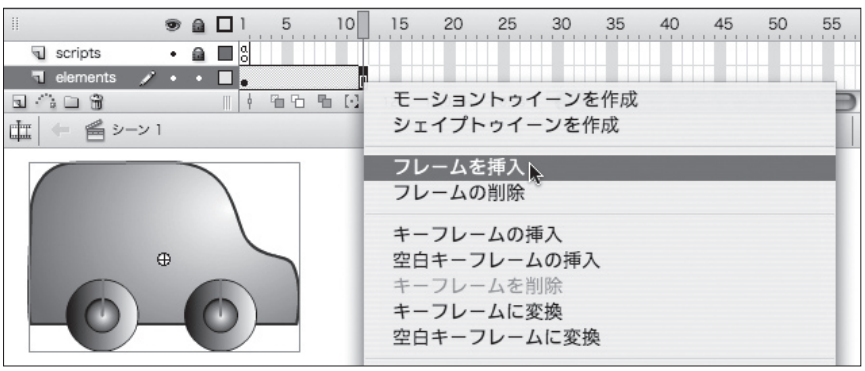


図01-019■ムービークリップインスタンスのあるレイヤーを12フレームに拡げる
ムービークリップが配置されたレイヤーの第12フレームで、右クリック（右クリックの使えないMacintoshでは［control］＋クリック）で［フレームを挿入］もしくはショートカット［F5］。

［ムービープレビュー］（ショートカットWindows［Ctrl］＋［Enter］/Macintosh［command］＋［return］）すると、フレームレートがデフォルトのまま12fpsであれば、1秒ごとに車のcar_mcは100ピクセルずつ右に移動し、前輪のtire0_mcは90度ずつ回転します。

フレームアクションは、そのフレームを描画する直前に実行されます。Flash Playerは、SWFのメインタイムラインの12フレームを再生し終えると、デフォルトでは再生ヘッドをまた第1フレームに戻します。第1フレームアクションは、再生ヘッドが第1フレームに入るたびに実行されます。したがって、フレームレートが12fpsであれば、1秒間隔で第1フレームアクションが処理されて、アニメーションが行われることになります。

Tips 01-010 ■フレームアクションと画面の描画

フレームアクションはそのフレームを表示する前に実行され、スクリプトの処理が終わってからフレームのイメージが画面に描画されます。

したがって、フレームアクションの中でインスタンスの座標を何度も動かしたり、その表示状態をさまざまに変更しても、その経過がアニメーションとして表示されることはありません。すべてのスクリプトの処理が終わったとき、最終的な座標や表示状態で、画面の描画が更新（リフレッシュ）されるからです。

また、フレームアクションは、再生ヘッドが移動してそのフレームに入ってきたときに実行されます。タイムラインに1フレームしかない場合には、再生ヘッドの移動が起こりませんので、そのフレームアクションは最初に1度しか処理されません。

ただし、画面の描画はフレームレートの頻度で更新されます。ですから、その描画更新時（厳密にはその直前）に実行されるスクリプトを書けば、1フレームだけでもアニメーションの処理を行うことは可能です（後述Chapter2「スクリプトによるアニメーション」で解説します）。

スクリプトが1秒に1回の実行では、1fpsのフレームアニメーションと同じで、動きが滑らかになりません。連続するフレームの長さを、縮めることにします。ただし、1フレームではスクリプトが繰返し処理されません（前述Tips 01-010参照）。そこで、フレームの長さは2フレームとします。フレームレート12fpsに対して2フレームごとにスクリプトを実行すれば、6fpsに相当するアニメーションになります。

Tips 01-011 ■連続したフレームの長さを変える

連続したフレームの長さを変えるには、削除あるいは追加したいフレームの範囲を選択して右クリック（右クリックの使えないMacintoshでは［control］＋クリック）から［フレームを挿入］（ショートカット［F5］）したり［フレームの削除］（ショートカット［Shift］＋［F5］）をする以外に、マウスで操作する方法もあります。

Windowsは［Ctrl］キー、Macintoshは［command］キーを押したまま、長さを変えたいフレームの終端にマウスポインタを合わせると、カーソルが左右の矢印（←→）に変わります。そのまま水平にドラッグすると、連続したフレームの長さを伸ばしたり、縮めたりすることができます（図01-020）。ただし、1フレームしかない場合には、この方法でフレームを伸ばすことはできません。

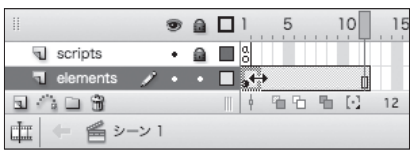


図01-020■［Ctrl］（Windows）または［command］（Macintosh）キーを押しながらフレームの終端をドラッグする
マウスポインタは、左右の矢印（←→）に変わる。ただし、1フレームしかない場合は、伸ばすことはできない。

スクリプト01-003における車のインスタンスcar_mcの移動や前輪のtire0_mcの回転は、処理頻度の増加に対応して値を減らすことにしましょう。また、ここで後輪のインスタンスtire1_mcにも、前輪と同じ処理を加えます。試しに、車の移動距離を5ピクセル、タイヤの回転角を10度としてみました（図01-021）。

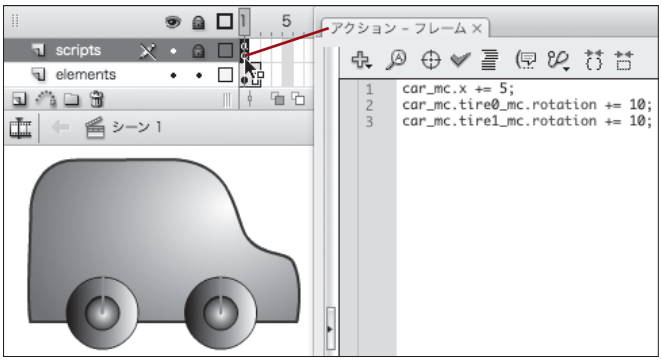


図 01-021 ■車を 5 ピクセル水平移動してタイヤを 10 度回転
車のムービークリップインスタンス car_mc を右に 5 ピクセル移動し、タイヤのインスタンス tire0_mc と tire1_mc を 10 度回転する。

1.5.2 アニメーションのパラメータを変数で指定する

実際のアニメーションコンテンツでは、［ムービープレビュー］で動きを確認しながら、スピードなどの処理に用いられる値を調整することが少なくありません。しかし、その処理があちこちのステートメントにばらばらに使われていると、修正箇所を漏れなく探し出すのに手間がかかります。

また、数値の間に一定の関係をもたせたい場合もあります。たとえば、今回の例（図 01-021）では、前輪と後輪は同じ回転角で動かないと不自然です。また、車の移動距離に対するタイヤの回転角は、アニメーションとして適切な一定の比率が存在するでしょう。

このようなときは、「変数」を使うと、記述場所をまとめることができ、また数値間の関係を定めることも可能です。「変数」は、一種のメモリです。メモリは、電卓にもついています（図 01-022）。たとえば、「2×3＋4×5」を電卓で計算するとき、「2×3」の答えを一旦メモリに取っておく必要があります。

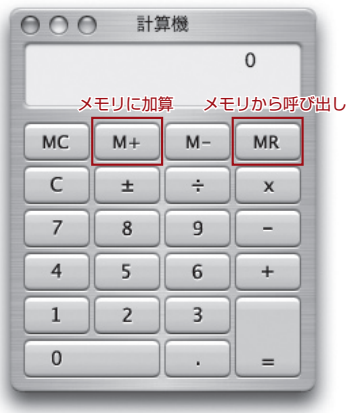


図 01-022 ■電卓のメモリ
［M+］でメモリに加算、［MR］で値を呼出す。

電卓の操作は、つぎのようになります。

1. [2]［×］[3]［=]：6が表示される
2. [M+]　　：6がメモリされる
3. [C]　　：表示がクリアされて0になる
4. [4]［×］[5]［=]：20が表示される
5. [+]［MR］[=]　：メモリされていた6が加算されて26と表示される

この同じ計算を、ActionScript 3.0で行ってみます。電卓のメモリはひとつしかないのに対して、スクリプトではいくつでも使うことが可能です。その代わり、変数には名前をつけなければなりません。名前は、もちろん「識別子」（Word 01-001）で設定します。

変数名は、varというキーワードを使って定義（宣言）します。その際、変数にメモリするデータの種類（データ型）も指定します。たとえば、数値の場合には、Numberというのがその指定になります。そして、変数に値をメモリするには、プロパティに値を設定する場合と同じく、代入を行います。

Word 01-004 ■データ型

プログラムが扱うデータの形式のことを「データ型」と呼びます。データ型によって、そのデータに対して行うことのできる処理が決まります。

変数にデータ型を指定すると、そのデータ型以外のデータを代入しようとしたり、そのデータに対して許されない処理を行おうとすると、エラーが発生します。それにより、プログラムで誤って不正な処理を書いてしまうことが防げます。

ActionScript 3.0では変数を定義するとき、データ型を指定しないことも可能です。ただし、データ型を指定しなければ、そのデータのチェックは行われず、誤った処理に対するエラーも発生しなくなります。データ型を指定することにより、プログラムのミスが発見しやすくなり、変数の中身が明らかになってプログラム自体も見やすくなります。また、データ型に応じた処理の最適化もはかられます。

AS1&2 Note 01-004 ■ ActionScript 2.0 のデータ型指定

データ型を指定して変数を定義することは、ActionScript 2.0でも可能です（1.0では型指定はできません）。ただし、ActionScript 2.0ではデータ型をSWFの書出し（コンパイル）時にしかチェックしないのに対して、3.0では実行（ランタイム）時にもチェックが行われます。

変数を定義して、値を代入するときのスクリプトの構文は、つぎのようになります。

```
var 変数名:データ型;
変数名 = 値;
```

この2行のステートメントは、1行で記述することも可能です。

```
var 変数名:データ型 = 値;
```

変数に値が代入されれば、その変数名を値の代わりに使って、スクリプトを記述することができます。たとえば、「変数名+1」という式は、変数に入っている値（数値が代入されていると期待されます）を取出して、それに1を加算する処理になります。

さて、電卓と同じ手順で「2×3＋4×5」を計算するフレームアクションは、つぎのとおりです。変数はふたつ用意しました。「2×3」の結果を変数nに、最終的な答えを変数nAnswerに代入しています（スクリプト01-004）。なお、trace()関数は、［ムービープレビュー］のとき［出力］パネルを開いて、括弧()内に指定した変数などの値を表示します（図01-023）。

```
trace（出力したい変数・式）；
```

Word 01-005 ■関数

「関数」は、狭い意味では、指定された値をもとに予め決められた処理を行い、その結果を示すプログラミング上のひとつにまとめられた命令のことを指します。Microsoft Excelの関数は、その典型です。

関数をスクリプトから実行することは、関数を「呼出す」と表現することもあります。関数の括弧()内に値を指定することは値を「渡す」といい、渡される値を「引数」（ひきすう）あるいは「パラメータ」と呼びます。関数が処理結果の値を示すことは値を「返す」と表現し、返される値は「戻り値」と称されます。なお、クラスに定義された関数を「メソッド」と呼びます（「クラス」や「メソッド」については、また改めて解説します）。

広い意味では、関数にはつねに値を渡す必要はなく、関数が値を返すことも必須ではありません。つまり、予め定められた一連の処理を行う命令群の定義が、「関数」だといえます。ActionScriptは、この意味の関数（メソッド）とプロパティで構成されています。

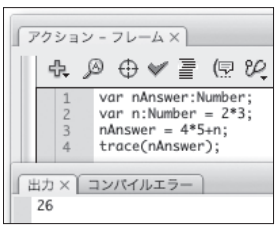


図01-023■変数を使ったフレームアクションと［出力］パネルの表示
変数nAnswerに代入された計算結果を［出力］パネルに表示

スクリプト01-004 ■変数を使って計算するフレームアクション

```
var nAnswer:Number;
var n:Number = 2*3;
nAnswer = 4*5+n;
trace(nAnswer);
```

スクリプト01-004の処理内容は、つぎのとおりです。

［1］第1ステートメントは、変数nAnswerをNumber(数値)型で宣言しています。ここではまだ値を設定せず、後のステートメント（［3］）で最終的な計算結果を代入します。

［2］第２ステートメントは、変数ｎをやはりNumber型で宣言し、電卓の場合と同じく、途中経過の式「2×3」の値を代入します。なお、掛け算の「×」には*（アスタリスク）、割り算の「÷」には/（スラッシュ）を演算子として用います。

［3］第３ステートメントは、変数nAnswerに最終的な計算結果を代入します。代入の右辺は、「4×5」に変数ｎから取出した値（［2］で代入された数値6）を加算して、式4*5+nとしています。

［4］第４ステートメントは、trace()関数を使って、変数nAnswerの値を［出力］パネルに表示します。

それでは、変数の使い方がわかったところで、車のアニメーションの例（図01-020）に戻って、スクリプトを書いてみましょう（図01-024）。車のインスタンスcar_mcの水平移動ピクセル数とタイヤのインスタンスtire0_mcとtire1_mcの回転角の比率は1:2とします。また、前輪と後輪は、同じ速度で回転させます。取りあえず、フレームアクションの実行１回ごとのcar_mcの水平移動距離は、５ピクセルで設定してみます（スクリプト01-005）。

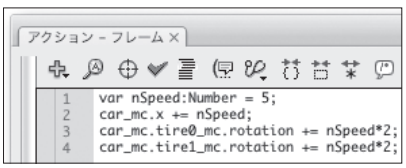


図01-024■変数を使って計算するフレームアクション
冒頭に変数nSpeedを宣言して、アニメーションの処理に利用。

スクリプト01-005■変数を使ったフレームアクションに修正

```
var nSpeed:Number = 5;
car_mc.x += nSpeed;
car_mc.tire0_mc.rotation += nSpeed*2;
car_mc.tire1_mc.rotation += nSpeed*2;
```

［1］第１ステートメントは、変数nSpeedをNumber型で宣言し、同時に水平移動距離として用いる数値５を代入しています。

［2］第２ステートメントでは、［1］で設定した変数nSpeedの値を、ムービークリップインスタンスcar_mcのxプロパティに加算しています。すると、車のインスタンスcar_mcの水平座標は、右に５ピクセル移動します。

［3］第３ステートメントは、ムービークリップインスタンスcar_mc内に配置されたtire0_mcのrotationプロパティに、変数nSpeedの２倍の値を加算しています。したがって、前輪のインスタンスtire0_mcは、10度回転します。

［4］第４ステートメントはムービークリップインスタンスcar_mc内のインスタンスtire1_mcに対して、［3］と同じ処理を加えています。ですから、後輪も前輪と同じ角度回転します。

このように前掲スクリプト01-005では、［1］で設定した変数nSpeedの値を使って、［2］の車の移動と［3］［4］のタイヤの回転が処理されています。つまり、冒頭の［1］の変数の値を変えるだけで、その数値の大きさにしたがって車の移動距離とタイヤの回転角が変わり、かつ移動距離と回転角との間にはつねに1:2の比率が保たれます。

Column 01　ECMAとECMAScript

ActionScriptは、Flash 5から導入されたオブジェクト指向プログラミング（スクリプト）言語です。ActionScriptは、ECMA-262仕様に準拠します。ECMA-262は、JavaScriptの国際標準とされています。したがって、基本的な文法は、JavaScriptと同一です。実際、ArrayやDate、Math、Objectなど、JavaScriptとまったく同じクラス（オブジェクト）も、数多く実装されています。ECMA-262に準拠したスクリプトは、とくに欧米ではECMAScript（エクマスクリプト）と呼ばれます。

Word 01-006 ■ ECMA

「ECMA」は、1961年に「European Computer Manufacturer Association」として、ヨーロッパのコンピュータメーカーを中心に設立された組織です。日本語では、ヨーロッパ（欧州）電子計算機工業会と訳されます。1994年に、名称を「Ecma International」と改めました。おもに、情報通信技術に関する標準を策定しています。現在では、世界各地に会員企業を持つ国際団体となっています。2003年11月17日に、Macromedia社（2005年12月Adobe Systems社により買収）も正式に参加しました。

JavaScriptは、Netscape Communications社（1998年に現AOL Time Warner社により買収）とSun Microsystems社が共同開発したスクリプト言語です。それに対抗したMicrosoft社が自社技術を追加・拡張して、JScriptを開発しました。NetscapeとMicrosoftの仕様は細部で異なっていたため、両社も参加して、ECMAがECMAScriptとして標準化しました。

こうした経緯ですので、JavaScriptもJScpritも、厳密にはECMA-262と一致していない部分があります。これはActionScriptでも同様で、ECMA-262に完全に準拠している訳ではありません。

Flash MX 2004からは、ActionScript 2.0が実装されました。Flash MXまでのActionScriptは、バージョンが1.0ということになります。Flash Player 9以降も、ActionScript 2.0および1.0はサポートされます。ActionScript 2.0の仕様も、やはりECMA-262によって定められています。ただし、ActionScript 1.0がECMA-262第3版に準拠するのに対して、ActionScript 2.0は第4版にもとづきます。

Word 01-007 ■ ECMA-262 第 3 版と ECMAScript 4

ECMA-262第3版 は、「Standard Ecma-262 ECMAScript Language Specification 3rd edition」（1999年12月）という文書により規定されました。前述のとおり、JavaScriptやJScriptが、この標準に準拠しています。

これに対して第4版は、「ECMAScript 4 Netscape Proposal」という提案書からスタートした仕様です。ECMAScript 4に準拠する言語としては、ActionScript 3.0と2.0のほかJavaScript 2.0とJScript.NETが挙げられます。

その後若干の曲折を経て、現在ではAdobeとMozillaが共同開発で、ECMAScript 4の策定を進めています。そして、ECMAScript.org(<http://www.ecmascript.org/>)には、言語の概要として「Proposed ECMAScript 4th Edition - Language Overview」が公開されました。

ECMAScript 4に則るActionScript 2.0は、データ型の指定とクラスをベースとしたプログラミングスタイルに大きな特徴があります。ActionScript 3.0も、同じくECMAScript 4に準拠します。したがって、ActionScript 3.0の基本的な文法や構文は、2.0と変わりません。

ただし、ActionScript 3.0ではクラスの言語体系が整備され、プロパティやメソッドなどの細かな仕様に変更が加えられています。そのため、ActionScript 2.0で記述したスクリプトは、そのままでは3.0で動作しない場合が多いでしょう。