4.8 メソッドもプロパティ

この機会に、若干突っ込んだ解説を加えておきます。スクリプト初心者は、必ずしもこの節を理解できなくても心配いりません。ただ、中級にステップアップする際に知っておくとスクリプティングの幅が広がります。その段階で、この節を読み直しても結構です。

今まで学習したメソッドやプロパティを設定するステートメントを、3つ取り出して並べてみます。

```
this.stop();

this._x = 0;

this.onRelease = xTrace;
```

1番目と3番目(xTraceは「function」名)は、MovieClipのメソッドを使ったステートメントです。2番目は、MovieClipのプロパティを設定しています。しかし、その構造を比べると、2番目と3番目が同じで、1番目とは異なるように見えます。2番目と3番目は、代入式の形を採っています。それに対して、1番目は代入ではなく、またメソッドのうしろに()がついています。

スクリプトの構造的に、2番目と3番目が同じという見方は、正しい側面があります。

「MovieClip.onRelease」は、MovieClipのプロパティととらえることもできるからです。さらにいえば、ActionScriptでは、メソッドも一種のプロパティとして扱われているといえます(「3.3 オブジェクトとは」で、メソッドも広い意味のプロパティだと解説したことを思い出してください)。

メソッドもプロパティなので代入ができるのです。ただ、「_x」プロパティでは単純な数値しか設定できないのに対して、「onRelease」メソッドは何らかの処理を行う「function」がその内容になるという違いがあるだけです。

メソッドはプロパティとして「function」が代入できるということは、設定する「function」を必要に応じて切り替えられるということを意味します。中・上級者が使うテクニックに、たとえば「onRelease」メソッドの処理を状況に応じて(条件判定して)切り替えるという手法があります。こうした技術も、メソッドがプロパティだということを理解していると、習得しやすいでしょう。

注意しなければならないのは、イベントハンドラメソッドに対してプロパティとして「function」を設定するときは、()をつけないということです。()をつけると、「function」の実行になってしまいます。

```
this.onRelease = xTrace();
```

このステートメントでは、「function」として「xTrace」を実行し、その結果を「this.onRelease」に設定します。ユーザー定義関数「xTrace」が、「return」アクションで値を返していればその値が、返していなければ未定義[*10]という値が返され、その値が「this.onRelease」メソッドに設定されるということになります。

「function」定義、つまり「function」の内容そのものを設定したい場合には、「function」名を()なしで設定するか、名前のない関数を使って直接「function」定義を代入する必要があるのです。

なお、「function」は、プロパティだけでなく変数[*11]に代入することもできます。

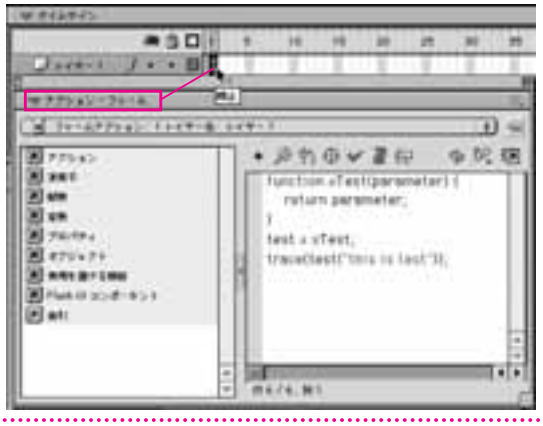
たとえば、引数として渡した値をそのまま[出力]ウィンドウに表示する「function」を定義して、その「function」を変数「test」に代入するスクリプトは、次のようになります。

```
function xTest(parameter) {
     return parameter;
}
test = xTest;
```

こうすると、「function」名「xTest」の代わりに「test」を使って、ユーザー定義関数を実行することができます。「実行」する場合には、()が必要なことに注意してください。「xTest」は引数を取りますので、()の中には任意の数値あるいはストリング(文字列)を入れます。ストリングを指定するときには、「"」(ダブルクォーテーション)で囲います。

```
test ("this is test") ;
```

たとえば、新規ムービーを作成して、次のフレームアクションをメインのタイムラインの第1フレームに記述すれば、[出力]ウィンドウには「this is test」と表示されます。



```
function xTest(parameter){
    return parameter;
}
test = xTest;
trace(test("this is test"));
```

ところで、イベントハンドラメソッドを使用するときに、「function」定義とその設定だけは行いました。しかし、実行のステートメントを記述していません。つまり、「this.onRelease()」という()つきのステートメントがありませんでした。これは、イベント(「MovieClip.onRelease」の場合はマウスクリック)が発生したとき、Flashが内部的にこの実行を行ってくれるからです。

いってみれば、イベントハンドラメソッドを設定することは、携帯の着メロを設定するようなものです。着信(イベント発生)したら、設定した曲を再生(「function」実行)してくれるのです。

ここでひとつ注意しておきたい点があります。「function」をプロパティや変数に代入した場合、その「function」が複製されるわけではありません。そのプロパティや変数に設定した「function」を実行すると、代入もととなった「function」が呼び出されることになります。いわば、転送電話のような状況です。かけられた電話番号(変数名)は単なる窓口で、呼び出されるのはおおもとの電話(function)だということです。

あるいは、銀行口座を考えてもいいでしょう。銀行に口座を開くと、どこの支店でも預金の出し入れができます。ただ、どの支店からも、同じ口座の情報を参照しているということが重要です。新宿支店で預金を引き出せば、渋谷支店で見てもちゃんと残高が減っているはずです。

「function」を複数の変数に代入したときも、それらの変数はあくまで窓口・支店です。参照しているのは、1つの同じ「function」になります。したがって、もとの「function」に変更を加えれば、そのあとはすべての変数がその変更後の「function」を参照することになるわけです。

これに対して、数値やストリングなどを変数に代入する場合は処理が異なります。変数aに「1」を代入し、変数aを別の変数bに代入した場合には、変数aの数値「1」がコピーされて格納されます。ですから、あとから変数aに「2」を代入しても、変数bの内容は「1」のままということになります。

「function」のようなタイプ(型)のデータを「参照型」といい、数値やストリングは「基本型」と呼ばれます。銀行預金口座は、残念ながら「参照型」だということでしょう。

*10：値が設定されていないときは、ActionScriptは「undefined」という特別な値を返します。前記サンプルスクリプトの「xTrace」は、「return」アクションで値を返していません。したがって、次の「trace」アクションを実行すれば、[出力]ウィンドウに「undefined」と表示されます(これはFlash MXの場合です。Flash 5では「undefined」は表示されません)。

```
trace(xTrace());
```

*11：「2.3 変数を使う」で述べたとおり、変数はそのMovieClipのタイムラインに設定されます。ですから、変数はMovieClipに設定されたユーザー定義プロパティだと考えることもできます。