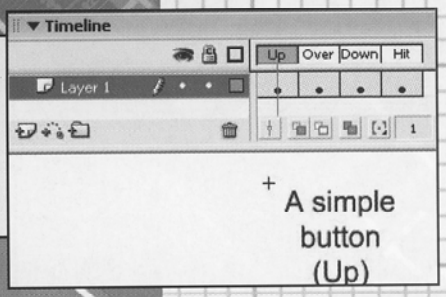
60 ACTIONSCRIPT TIPS

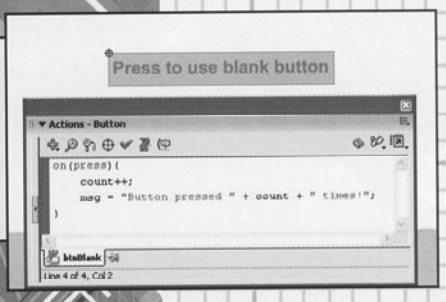
TIP 1

You can create a button using a ovieClip comprising four frames.



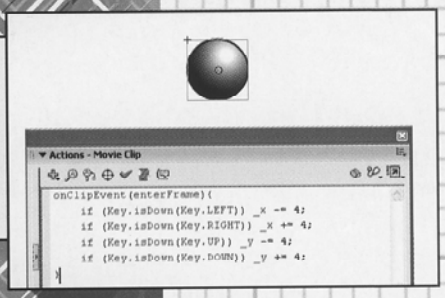
TIP 2

blank button is the simplest type to create, using just one frame.



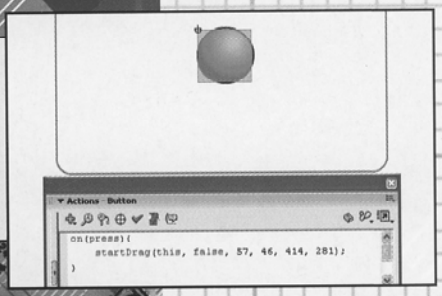
TIP 5

Add this simple code to move objects with the arrow keys.



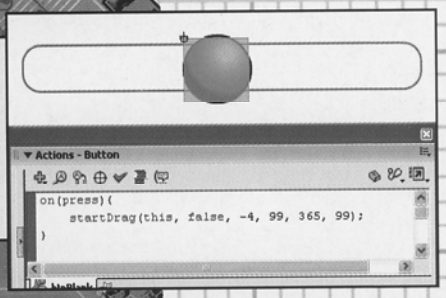
TIP 7

nit the drag region of an object by ing values to the call to startDrag.



TIP 8

You can create a simple scroll bar by limiting the y values.



1. Creating a button

Buttons are just a special kind of four-frame MovieClip. The four frames are the up, over and down images and the hit area, in that order. You can add a script to these frames by clicking the button and adding events for press, release, rollover and rollout.

2. Using a blank button

The simplest kind of button you can create contains only the fourth frame: the hit area. A blank button is displayed in turquoise at design time, but is blank at runtime. You can simply place them over the top of other objects to create a clickable button.

3. A multistate button

Sometimes you'll need a button that can be set to multiple states. To do this, create a button, select it, then press F8, and create a MovieClip symbol from the button. Now when the button is clicked, you'll be taken to a new frame inside the clip.

4. Jumping with a button

Often a button will be used to navigate a movie. Simply add the following code:

```
on(press){
  gotoAndStop("MyMarker");
}
```

This causes the playback of the timeline where the button is placed to jump to the frame labelled MyMarker.

5. Moving with the arrow keys

To move an object on screen it should be a MovieClip. You can add code to the MovieClip by adding an onClipEvent handler. In this example, we add an onClipEvent(enterFrame) handler. We check the keyboard using the Key object. If the arrow keys are pressed, we alter the position of the MovieClip by adjusting the _x and _y values.

6. Moving by dragging

Create a MovieClip with a blank button inside. Add the following code to the button and you'll have a draggable clip:

```
on(press){
  startDrag(this, false);
}

on(release){
  stopDrag();
}
```

7. Limiting the drag region

Often you'll want to limit the region in which the dragged object can move. This is done by adding four additional parameters to the call to startDrag: left, top, right and bottom values. To find these values, use the following code to display the current position of your clip:

```
trace(_x + ", " + _y);
```

Write down the values at the top-left and bottom-right corners and use these in the call to startDrag.

8. Simple scroll bar

You can use the draggable clip to create a simple scroll bar. Just limit the y values to the same value for top and bottom and you have a horizontal scroll bar. You can use the _x value to give a value for the scroll bar using the following code, where left and right are the values used in startDrag, and limit is the maximum value:

```
value = ((_x - left)/(right -
left)) * limit;
```

9. Link the scroll bar to a text box

To display the value, create a text box. Use the Properties window to set it to Dynamic and enter a variable name, in this case 'scroll.value'. 'scroll' is the instance name for the scroll bar. The value 'value' is set when the button is released.

10. Moving in a circle

To move in a circle, use the Math object methods sin and cos. If the radius of the circle is x, then the position of the object at angle theta is:

```
_x = x * Math.cos(theta);
_y = x * Math.sin(theta);
```

To centre this in the Movie, add half the width to the _x value and half the height to the _y value.

11. Creating a function

Functions are very useful for keeping a block of code in one place. Use the keyword function followed by your choice of name and then brackets. The code for the function is contained between curly braces.

```
function myfunction(){
  trace("Called myfunction");
}
```

You call the function simply by using the name of the function and brackets.

```
myfunction();
```

You can pass parameters to a function using this code:

```
myfunction("Hello world");

function myfunction(str){
  trace(str);
}
```

12. Returning a value from a function

To return a value from a function, use the return keyword. In this example, the output from trace will be 16.

```
trace (sum(11, 5));

function sum(a, b){
  return (a + b);
}
```

13. Creating an Array

Arrays are useful for storing your information. You can create an Array using the new keyword:

```
var emptyArray = new Array()
var populatedArray = new
Array(1, 2, 3, 4, "a", "b",
false);
```

14. Accessing Array elements

```
var myArray = new
Array("Macromedia", "Flash",
"MX", "2004");
```

To access a member of myArray use myArray[i], where 'i' can be any value from 0 to one less than the number of elements in the Array.

```
myArray[0] is "Macromedia"
myArray[1] is "Flash" etc.
The number of elements in an
Array is myArray.length.
```

15. Deleting elements

To remove an element from an Array at index i, use the following code, where deleteCount is the number of elements to remove:

```
myArray.splice(i, deleteCount);
```

You can use the same method to add elements by setting the value of deleteCount to 0 and then passing further parameters. In this instance, the element "Flash" will be added at index 1.

```
myArray.splice(1, 0, "Flash");
```

16. Multidimensional Arrays

```
var subArray1 = new
Array("Macromedia", "Flash",
"MX", "2004");
var subArray2 = new
Array("Macromedia", "Director",
"MX", "2004");
var myArray = new Array("Hello",
subArray1, subArray2);
```

Using the above myArray[0][1] is 'Flash': 0 to choose the first element of myArray and then 1 to choose the second element of the embedded Array.

17. Joining an Array

You can join an Array together and return it as a String, a line of text. Each element is separated from the previous one with the parameter passed to the method join.

```
var myArray = new
Array("Macromedia", "Flash",
"MX", "2004");
trace(myArray.join(" "));
```

In this instance, 'Macromedia Flash MX 2004' is sent to the Output Window.

18. Creating an Array from a String

```
str = "Macromedia Flash MX
2004";
myArray = str.split(" ");
```

Using the split method of a String object, you can create an Array. You pass the symbol used to break the String into separate elements – in this case, a space character. The result of this is an Array of four elements: "Macromedia", "Flash", "MX", "2004".

19. Variable scope

Flash can contain MovieClips embedded into MovieClips. The root level timeline is itself a MovieClip. If you have a variable at the root called x then this is different from a variable called x inside a MovieClip called mc.

To access variables in the MovieClip mc from the root use: mc.x

To access variables in the root from mc use: _root.x or this._parent.x

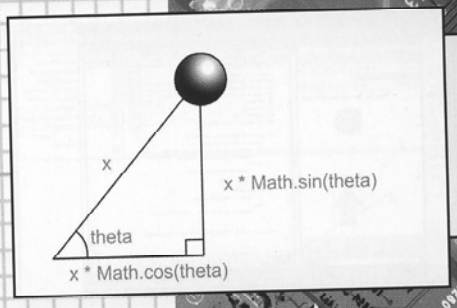
20. PC or Mac

To find out if the *Flash* movie is running on a PC or a Mac use:

```
os = $version;
if (os.substr(0, 1)=="W"){
  os = "Windows";
}else{
  os = "MAC";
}
```

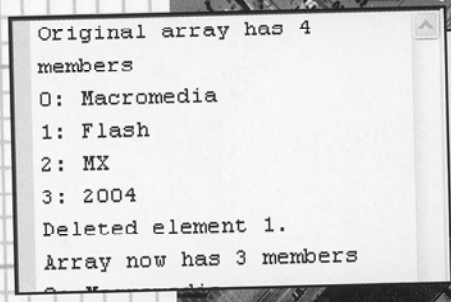
TIP 09

You can link a text box to a scroll bar in the Properties window.



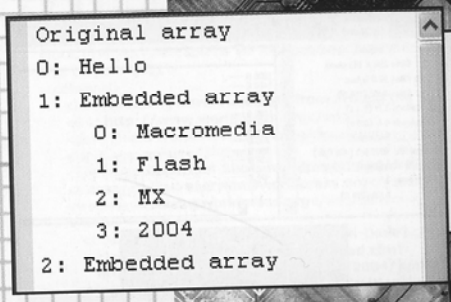
TIP 10

The Math object methods sin and cos enable movement in a circle.



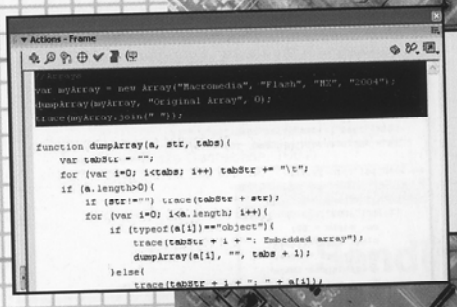
TIP 15

Use code to delete elements from, and add elements to, an Array.



TIP 16

Multiple elements can be added to create multidimensional Arrays.



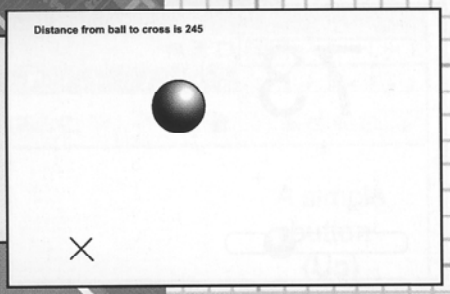
TIP 17

Arrays can be joined together to create a String – that is, a line of text

60 ACTIONSCRIPT TIPS

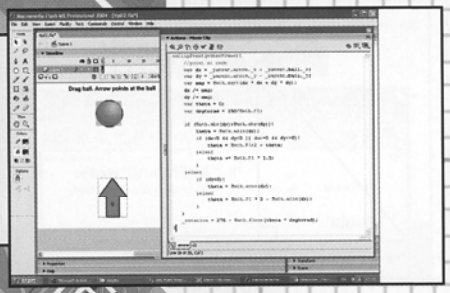
TIP 1

Work out the distance between two clips with some basic calculations.



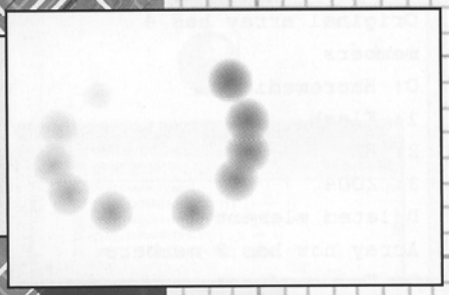
TIP 2

Determine the angle between the two clips before rotating the arrow.



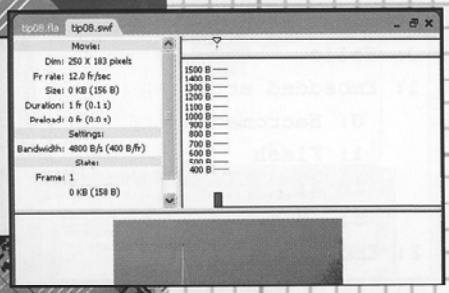
TIP 3

Make trails by using two MovieClips: individual and overall controller.



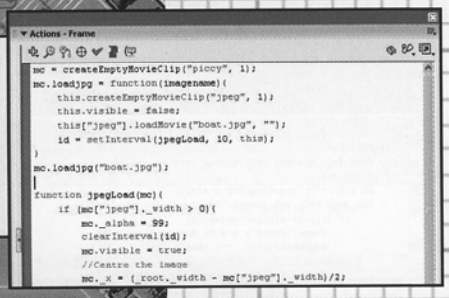
TIP 8

Create an empty MovieClip to make loading JPEG images easy.



TIP 9

A progress indicator enables you to see how long it takes JPEGs to load.



1. Calculating distances

You can calculate the distance between two MovieClips.

```
var dx = _parent.ball._x - _
parent.cross._x;
var dy = _parent.ball._y - _
parent.cross._y;
var dist = Math.sqrt(dx * dx +
dy * dy);
```

Use the square root of the sum of the square of the distance apart in the x and the square of the distance apart in the y.

2. Point one clip at another

This tip first creates two variables, dx and dy, which are the distances apart of the two clips in the x and y axes. Then you get the distance between these points. Next, divide dx and dy by this distance to get what is called a unit length vector. You can use this vector to determine the angle between the two clips and hence rotate the arrow clip to point at the ball.

3. Creating trails

Use two MovieClips: one clip is the individual part of the trail and the second is the overall controller. In this example, the individual part is set to fade out over time using the _alpha property. The controller creates new clips, starting at level 1, on the fly, at the existing mouse position. Once the limit of clips is reached, the controller loops back to level 1.

4. Using loadVariables

```
loadVariables("tip04.txt", this);
```

The above code segment loads the contents of the file 'tip04.txt' from the cover disc into named variables at the _ root of the Movie. tip04.txt is simply:

```
&count=100&
&loaded=true
```

Each variable name/value pair is separated by an ampersand.

5. Using the LoadVars object

```
lc = new LoadVars();
lc.onLoad = function(success){
  trace(success + " " + this.
count);
}
lc.load("tip05.txt");
```

The LoadVars object is a more elegant way of doing the same thing as in the previous tip. The principal benefit is you can define an onLoad function that's called when the data is loaded. To start the load, call the load method with a single parameter, the file to load.

6. Using the name Array

Every MovieClip has an array of names that can be used to access variables and Objects.

```
//Access any variable or Object
within the MovieClip
this[myName];
//Access any variable or Object
within the MovieClip s parent
this._parent[myName];
//Access and variable or Object
within the _root timeline.
_root[myName];
```

7. Using setInterval

To define something that occurs at a certain time interval or a timeout event, use setInterval. You pass a function that is called when the number of milliseconds in the second parameter elapses.

```
var id = setInterval(timeOut,
1000);
```

Use the return value from setInterval to clear the event using:

```
clearInterval(id);
```

8. Loading a JPEG image

Just create an empty MovieClip and load the image using the code below:

```
mc = createEmptyMovieClip("piccy
", 1);
mc.loadjpg =
function(imagename){
  loadMovie("boat.jpg", "");
}
mc.loadjpg("boat.jpg");
```

9. Adding a progress indicator

Use the following code to add a progress indicator to the JPEG Loader.

```
mc = createEmptyMovieClip("piccy
", 1);
mc.loadjpg =
function(imagename){
  this.createEmptyMovieClip("jpe
g", 1);
  this.visible = false;
  this["jpeg"].loadMovie("boat.
jpg", "");
  id = setInterval(jpegLoad, 10,
this);
}
mc.loadjpg("boat.jpg");
```

The trick here is to create an empty MovieClip, called 'jpeg' within the MovieClip mc. Set the MovieClip mc to invisible and then load the jpeg into the sub clip. Set an interval caller so that you can use this to update the progress. Look at the code on the cover disc for details.

10. Opening a file requester

This code opens a file requester:

```
os = $version.substr(0, 1);
if (os == "W"){
 getURL(".");
}else{
 fscommand("exec", "browseCD");
}
```

11. Repeating sections of code

The onEnterFrame function is good for code that needs to repeat at the pace of the movie's fps. Create it dynamically using:

```
this.onEnterFrame = function(){
//Things to do each frame
}
```

12. Starting with the drawing API

The code below creates a MovieClip and then uses the drawing API to draw a black line that's two pixels thick from the point (275, 200) to the current mouse position.

```
this.onEnterFrame = function(){
 removeMovieClip("mc");
 mc = this.createEmptyMovieClip
("mc", 0);
 mc.lineStyle(2, 0x000000,
100);
 mc.moveTo(275, 200);
 mc.lineTo(_root._xmouse, _
root._ymouse);
}
```

13. Draw rectangle function

Flash has the ability to extend the functionality of the built-in objects. You can add a prototype, in this case a function to draw a rectangle, to the MovieClip object and then every MovieClip has the additional function, drawRect.

```
MovieClip.prototype.drawRect =
function(l, t, w, h, scol, fcol){
 this.lineStyle(1, scol, 100);
 this.beginFill(fcol, 100);
 this.moveTo(l, t);
 this.lineTo(l + w, t);
 this.lineTo(l + w, t + h);
 this.lineTo(l, t + h);
 this.lineTo(l, t);
 this.endFill();
}
```

14. Drawing a curve

Set the starting position of a curve using the drawing API method moveTo. Then determine two points: the end point and a control point that determines the bend.

```
 mc = createEmptyMovieClip
("curves", 1);
 mc.lineStyle(2, 0xFF0000,
100);
 mc.moveTo(10,200);
 mc.curveTo(275, 0, 540, 200);
```

15. Drawing a circle

Use the drawing API method, curveTo, seen in the previous tip. Build a circle in 45-degree sections. Setting the end points uses the same method as beginners tip 10. The control points are halfway between the start and end points.

16. Drawing a pie chart

Extend the drawCircle method to create drawCircleSegment. Then generate a batch of data and use this to draw a pie chart. The code for this tip is on the cover disc.

17. Embedding fonts

Create a dynamic text box at design time and set this up to include the font outlines needed for the dynamically created text fields. Once the movie has the outlines required, set up a text field using createTextField to use the outlines just by setting the value for embed fonts to true.

```
newTextField.embedFonts = true;
```

18. Dynamic masking

Here, we create an empty clip and draw a rectangle in it. Then we create a second empty clip setting the first clip as a mask. By setting the mask as the drag item and loading an image into the second clip, we can move a window over the image.

```
mask = createEmptyMovieClip("box
", 2);
mask.drawRect(0, 0, 100, 100,
0xFF0000, 0x00FF00);
mc = createEmptyMovieClip("piccy
", 1);
mc.setMask(mask);
```

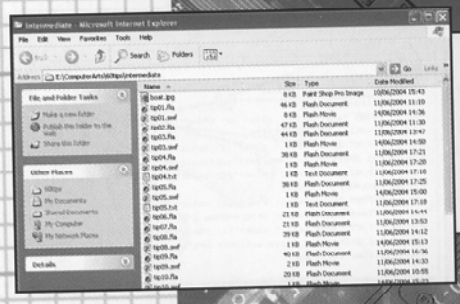
19. Using #include

You can write ActionScript in a text editor or use Flash MX 2004 Pro to add an ActionScript file (.as). To use the ActionScript in your movies, use this:

```
#include "tip19.as"
```

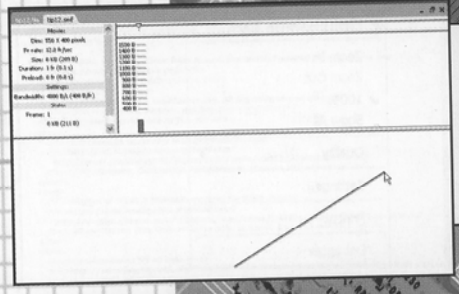
20. Email validation

The ActionScript File 'tip20.as' on the cover disc contains the code to validate an email address. Using simple String parsing, it checks if the '@' and '.' symbols are present by ensuring that there's a name before the '@' symbol and a domain after it. The function getValidationCode returns a value showing any errors that may have occurred.



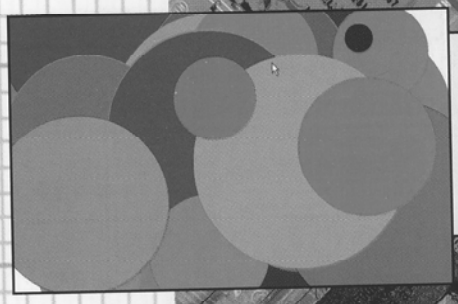
TIP 10

Enable users to browse files using this handy piece of code.



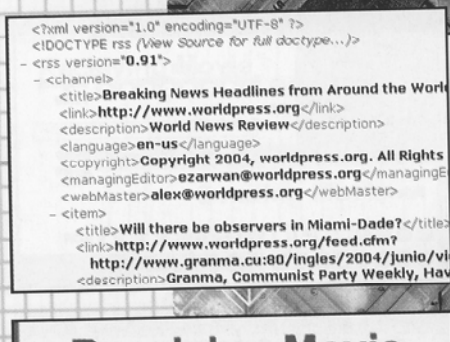
TIP 12

Dynamically draw shapes using the Flash ActionScript drawing API.



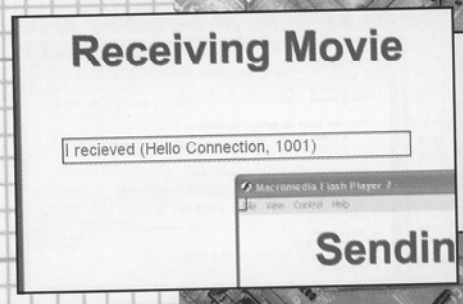
TIP 14

Draw a curve using the drawing API method moveTo.



TIP 16

Create a pie chart including a colour guide and text fields.



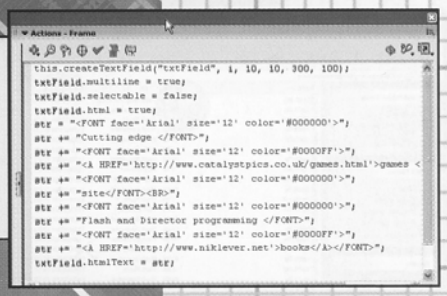
TIP 18

Dynamically mask an image by drawing a shape in an empty clip

60 ACTIONSCRIPT TIPS

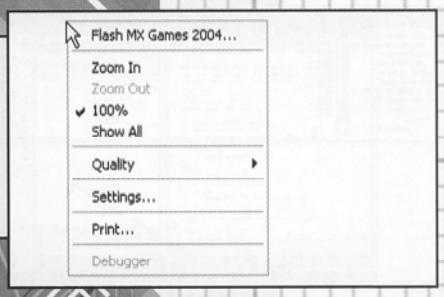
TIP 2

You can add HTML formatted text by setting the text field to an HTML field.



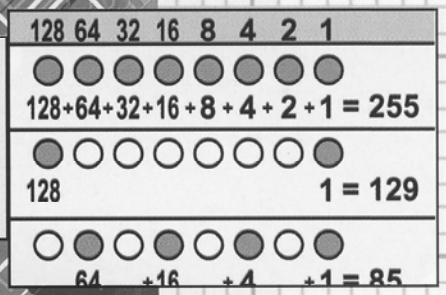
TIP 3

Creating a ContextMenu object enables you to add items to the menu.



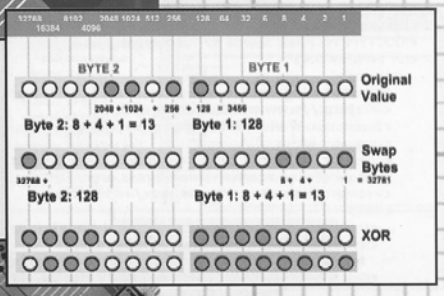
TIP 5

Manipulate data at bit level using bitwise operators.



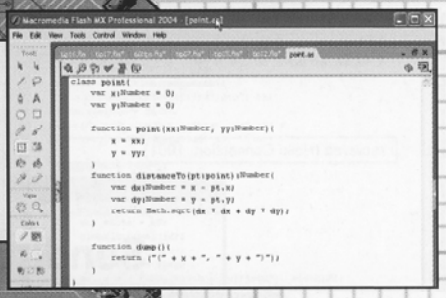
TIP 7

Bitwise operators can be used to encode info you pass over the web.



TIP 12

Get wise to using classes within an ActionScript file.



1. Additional TextField formatting

There are many properties of a text field created with createTextField. You can add a border using:

```
txtField.border = true;
```

You can size the border to fit the current text using:

```
txtField.autoSize = true;
```

You can set its type to an Input Field using:

```
txtField.type = "input";
```

2. Using HTML text

By setting the text field to be an HTML field, you can add HTML formatted text. In this example, we add two lines each with formatted text and a Hyperlink. The txtField has selectable set to false to make sure the correct cursor is displayed, and it has multiline set to true.

3. Altering the Context menu

To add items to the Context menu, you first need to create a custom ContextMenu object. Then you use the customItems Array to add content to the menu:

```
myArray.push(myContextMenuItem);
```

A ContextMenuItem is created using two parameters — the text to display and function to call. Finally, you set the context menu as the menu property of an Object on screen, in this case the _root Object.

4. Using multiple Context menus

You can attach a ContextMenu object to a specific button, MovieClip, or text field object, or to the entire movie. Use the menu property of the Button, MovieClip, TextField or _root classes to do this.

5. Using bitwise operators

At the most basic level, the data stored on a computer breaks down to either 1 or 0. This is a bit. You can manipulate data at the bit level using the bitwise operators.

```
n <<= 1;
```

The above code has the effect of moving the bits in n one to the left, which is the same as doubling its value.

6. Bitwise and, or and xor

For these operations, the bits in two variables are compared individually:

And (&) — if both bits are on, the resulting bit is set to on.

Or (|) — if either bit is on, the resulting bit is set to on.

Xor (^) — if either bit is on but not both, the resulting bit is set to on.

7. Encoding and decoding

Using the bitwise operators it's possible to encode information that you pass across the web. For example, if a variable can only have values up to 65535 then it can be stored in two bytes. Get the first byte and the second byte and then swap them round. Xor the result and you have a garbled value. To restore it, do the operation backwards.

8. Using Listeners

Assuming that txtField is an Input Text Field then changing the content will have the result of calling the onChanged method of the Listener Object.

```
myListener = new Object();
myListener.onChanged =
function(){
  trace("Contents changed");
}
txtField.
addListener(myListener);
```

9. Responding to code events

When the text of a TextField is set in code, the onChanged event is not triggered. To get around this, you can add a setText method to the TextField Object.

```
TextField.prototype.setText =
function(str){
  this.text = str;
  this.broadcastMessage("onCha
nged");
};
```

10. Creating a point class

Classes are an excellent way of keeping your code well structured. They are easily created by writing a constructor function:

```
function point(){
  this.x = 0;
  this.y = 0;
}
```

To create a point simply call:

```
pt = new point();
```

The object will be created with the variables x and y set to zero.

11. Adding methods

You can add methods to the point class by using the prototype keyword followed by a method name.

```
point.prototype.init =
function (x, y){
  this.x = x;
  this.y = y;
};
```

12. Using a class file

An alternative approach is to create an ActionScript file and then create the class in the file. You declare any member variables and write a constructor function that has the same name as the class.

```
class point{
 var x:Number = 0;
 var y:Number = 0;

 function point(xx:Number,
yy:Number){
       x = xx;
       y = yy;
 }
}
```

13. Externally declared classes

To use an externally declared class, the class file needs to be in the same folder as the project file.

```
v1 = new vector(0, 1);
v2 = new vector(1, 1);
trace("Angle between " +
v1.dump() + " and " +
     v2.dump() + " is " +
v1.angleBetween(v2) + "
degrees");
```

14. Making class variables private

The keyword 'private' means that the direct value of a variable in the class is not exposed. You can write methods to get and set them. It makes the code more robust.

```
class triangle{
 private var pt1:point;
 private var pt2:point;
 private var pt3:point;
 private var triarea:Number =
0;
 ...
```

15. Using simple physics

The bat moves and rotates using basic physics calculations. First, create a point 100 pixels up from the bat's centre point. Then calculate a force based on this position and the current mouse position. This is used to move the bat. Finally, we calculate a rotational force based on the current movement force.

16. Starting with XML parsing

The XML object is useful for accessing xml feeds from the web. While powerful, it can be a little confusing. The purpose of this small example is to generate a list of data from any XML feed so that you can see the names and values of the many nodes. This takes a feed from a free news supplier.

17. Generating HTML from an XML feed

Once you've examined the feed, you can extend the parsing to generate anything your movie requires. In this example, we convert the news feed into HTML with hyperlinks. Only the nodes that have the nodeName, item, have the actual data required.

18. Using the LocalConnection object

The LocalConnection object is useful for communicating between two swfs running on the same PC. In this example, we create a receiving movie (tip18a.fla) and a sending movie (tip18b.fla). The sender communicates with the receiver using the connect name; it can then call methods on the receiver.

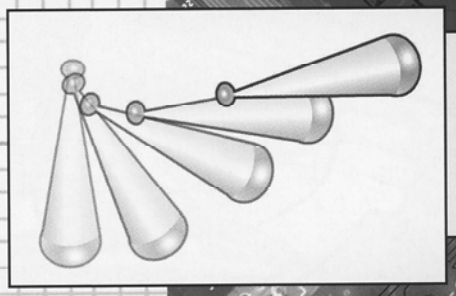
19. Using the SharedObject

The SharedObject is a good alternative to using cookies. You can store data on the client's machine or the remote server that can be accessed later.

```
so = SharedObject.
getLocal("myInfo");
if (so.data.favoriteColor==und
efined){
 //No saved data
 so.data.favoriteColor =
"orange";
 colors = new Array("red",
"blue", "green", "blue");
 so.data.possibleColors =
colors;
}else{
 //Show the saved data
 trace(so.data.
favoriteColor);
 trace(so.data.
possibleColors);
}
```

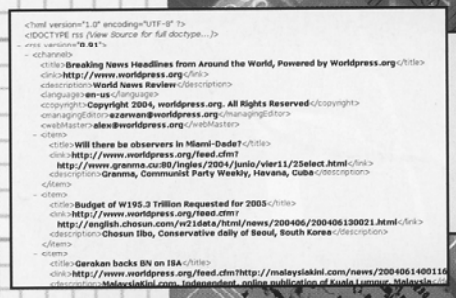
20. Using the ASV application

If you were to lose your project but still had the swf, you may well be able to recover the code using the ActionScript Viewer, which is a terrific application. The most current demo version of ASV can always be found at http://www.swf2fla.com/asvdemo.zip.



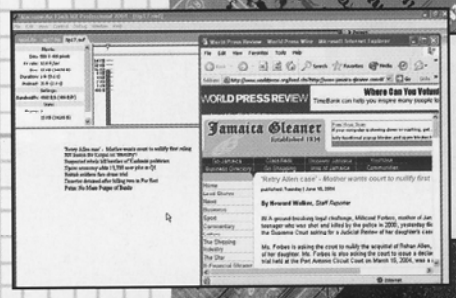
TIP 15

Move and rotate an object using simple physics calculations.



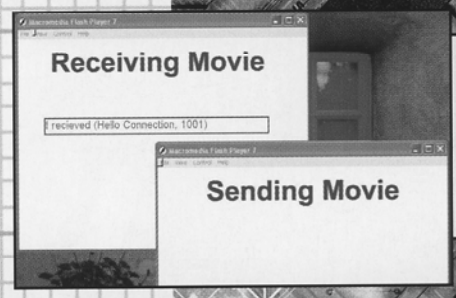
TIP 16

Access XML feeds from the web using the XML object...



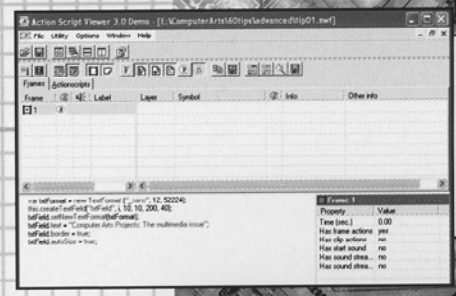
TIP 17

...and then use the feed to generate anything your movie requires.



TIP 18

Communicate between two SWF movies on the same machine.



TIP 20

The ActionScript Viewer can recover code from lost projects.