ActionScripting

Adding simple interactivity to your movies

In this section

In this section, you will learn how to:

- control the main *Timeline* using a button
- control a *Movie clip*'s *Timeline*
- control a *Movie Clip*'s properties
- set and use variables
- use a conditional statement
- create a preloader
- set dynamic text
- create drag and drop actions on *Movie clips*

ActionScript is the name of the scripting (coding) language that Flash uses to add interactivity to a movie. **ActionScript** is similar to **JavaScript**, the language that some web pages use.

ActionScript can produce some **very complex** results and this chapter will skim lightly over the most basic techniques—don't expect to become an ActionScript expert overnight!

In the previous three sections you made components and movies that could all have used some ActionScript. However, in those sections your focus was on using graphics. In this section the focus is on ActionScript, not graphics. Bring the two ideas together in your own time to produce attractive, interactive movies.

ActionScript 3.0

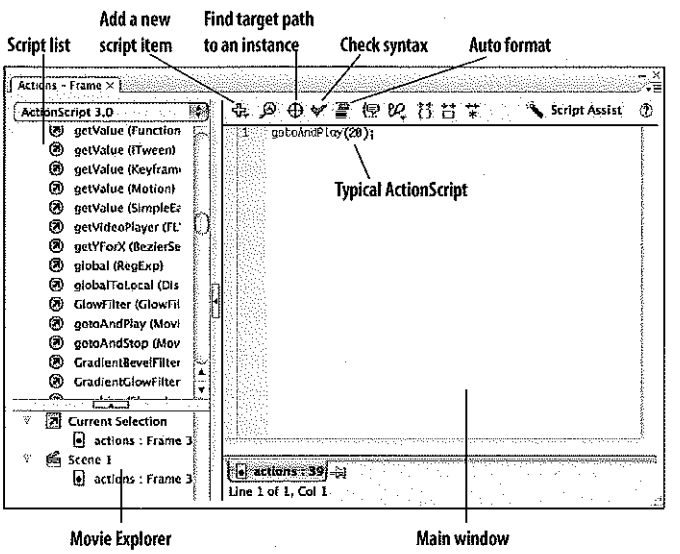
Flash CS3 introduces a new standard for ActionScript—version 3.0. Its implementation differs quite significantly from earlier versions of ActionScript and will mean that even at this basic level some techniques will need to change. This manual will use the new ActionScript 3.0 standards.

ActionScript basics

The Actions panel

ActionScript is written directly into the main window of the *Actions* panel. Call up the *Actions* panel by choosing *Window* > *Actions* (*Option/Alt F9*). Although this is a complex tool, there are a few parts that are used most often:

- Use the main window to write actions.
- Add parts of scripts from a list with the button or by dragging-and-dropping from the script list.
- 'Point' to symbol instances with the target path button.
- The *Movie Explorer* (bottom left corner) shows the movie parts.



Syntax and lettercase

ActionScript must be written in a certain way. Although the rules of ActionScript syntax are too complex for this introductory manual, be aware that the slightest mistake can make your movie stop working.

For example, there are rules about letter-case:

- ActionScript is **case-sensitive**. That means it matters whether something is written with upper or lower-case letters.
- Words are written in 'intercap' format. That means that words start with a **lowercase** letter. If a few words are run together, any word after the first word starts with an uppercase letter. For example: `play`; `gotoAndPlay`; `nextFrame`; `nextScene`.

There are also rules about brackets:

- **Parentheses** `( )` follow words to show that those words are instructions to Flash. For example: `stop( );` `gotoAndPlay(3);` The purpose of the brackets is to give us something to put extra instructions in.
- **Curly braces** `{ }` surround some blocks of code. The most important thing to remember is that for every opening { brace there must be a closing } brace somewhere later.

tip Copy-pasting code

You can copy and paste code in the *Actions* window, but you will have to use the shortcut keys (⌘/*Ctrl C* and ⌘/*Ctrl V*).

And one more rule about semi-colons:

- Where there is more than one line of code between your curly braces, each line must be separated by a **;** (see right)

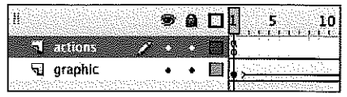
Semi-colons to separate lines of code.

```
function startClick(event){
    play();
    kettle_mc._alpha = .5;
}
```

To help you see what is right or wrong about a block of ActionScript code, notice that the code is coloured to identify different parts. It might also help to click the *Auto format* button — this makes your code nice and neat, which in turn makes it easier for you to spot mistakes, and will also let you know if it finds any errors.

ActionScript good practice

It is good practice to always keep your scripts on their own separate layer and to keep this layer at the top of the *Timeline* stack (see right). That way scripts are much easier to keep track of.



Controlling the main Timeline

Stopping the Timeline

The most basic ActionScript you will ever need (and you will need it often) is the **stop** action. In its simplest form it will stop the playback of the *Timeline* that it is placed on.

1 Make a simple animation in a new file.

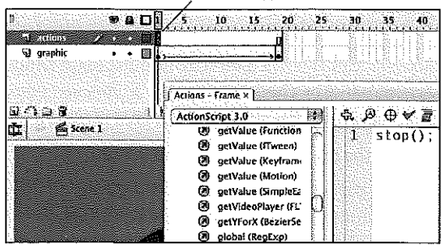
2 Make a new layer on the main *Timeline* and name it 'actions' (see right).

3 Select the first keyframe of the actions layer and open the *Actions* window.

4 In the main part of the window, type;
`stop();`

5 You will see a small 'a' appear in the keyframe (see right), indicating that an action is applied to that frame.

There is an action applied to this frame



6 Preview your movie; you'll see that the animation does not play.

This is a really useful action to add to the *Timelines* of *Movie clip* symbols; where you don't want them to loop forever (eg the animated button state on page 37).

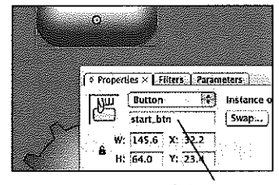
Playing the Timeline with a button

The main purpose of *Button* symbols is to trigger actions, and the main purpose of those actions is to stop or start a *Timeline* somewhere. To make a script which runs when a button is clicked, first you have to identify which button will be pressed.

1 Keep adding to the movie you made in the previous example.

2 Place a button (any button) on the *Stage* in a new layer.

3 Select the button on the *Stage* and in the *Properties* window, give the button an **instance name** (see right). Always start instance names with a lowercase letter and don't use special characters other than _ (underscore). Be descriptive but brief! Don't forget that ActionScript is case sensitive, so you will need to take note of where you have used lowercase and uppercase.



The script that you create will have two sections:

- A **listener** that waits for something to happen, which triggers:
- A **function** that tells Flash what to do

tip _btn and _mc

If you end your instance names with _btn (for a button) or _mc (for a *Movie clip*), Flash will give you code hints in the *Actions* window.

4 Select the first keyframe of the **actions** layer again (the same place where you placed the **stop** action) and open the *Actions* window. You will see the **stop** action still there.

5 Create the **listener:** Make a new line underneath `stop();` and type the following line of code:

```
start_btn.addEventListener(MouseEvent.CLICK, startClick);
```

The first part—`start_btn`—must match the instance name you assigned to the button previously. The last part—`startClick`—is a made up name, as long as it matches up with the name of the function in the next step, the code will work. Just make sure whatever name you use makes sense to you!

```
start_btn.addEventListener(MouseEvent.CLICK, startClick);
```

- This must match the name of your button instance from step 3.
- This is ActionScript code which waits for something to happen.
- This is ActionScript code which detects a mouse click.
- This is the (made up) name of the function that will be triggered in the next section.

6 Create the **function**: Make another new line and add the following lines of code:

```
function startClick(event) {
  play();
}
```

- This tells Flash that you are starting a function.
- This is the function name; it must match up with the function name in the listener.
- ActionScript gets event — in this case a mouse CLICK — from the listener.
- This is ActionScript code which makes the current Timeline play — it's just the opposite of the stop action.

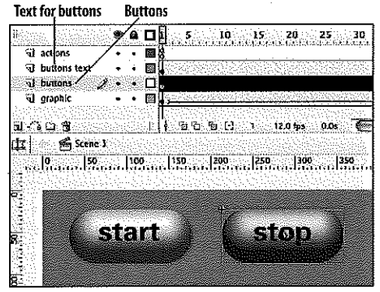
7 Now test the movie. Your movie should be stopped when it first loads and should play when you press the button.

Starting the main Timeline again

To start a *Timeline* that has been stopped:

1 Add another button instance to your buttons layer. So that you know which button is which, it's a good idea to label them—add a new layer with button text on it (see right).

Text for buttons | Buttons



2 Select the new button and give it an *instance* name (*Properties* window).

3 Select the first keyframe of the **actions** layer (you are putting all the actions in the same place!) and open the *Actions* window.

4 Add another **listener** to wait for the button to be clicked which triggers another **function** to stop the timeline. It will be almost identical to the previous **listener-function** pair, but you will need to make sure that the instance names match up, that the function has a new name (different from the previous function) and that you have set the function to `stop();` rather than `play();` (see right).

```
stop(); — Stops the Timeline first of all.

                                  Listens for start_btn to be clicked,
                                  triggers startClick function.
start_btn.addEventListener(MouseEvent.CLICK, startClick);

function startClick(event) {      — Plays the timeline.
    play();
}

                                  Listens for start_btn to be clicked,
                                  triggers startClick function.
stop_btn.addEventListener(MouseEvent.CLICK, stopClick);

function stopClick(event) {       — Stops the timeline.
    stop();
}
```

5 Test the movie now. You should be able to stop and start the *Timeline* at will.

Controlling a different Timeline

In the example above, you did not tell Flash which *Timeline* to stop or play. Flash assumed you meant the *Timeline* that the script is sitting on. To stop a different *Timeline*, a path to the *Timeline* is written using **dot syntax**.

Dot syntax is a type of addressing system for *Timelines*-within-*Timelines*. We might write:

```
movieA_mc.movieB_mc.stop();
```

This suggests that there is a *Movie clip* instance called 'movieA_mc', and inside it is a *Movie clip* instance called 'movieB_mc' that we are stopping.

A real-life version of this might be:

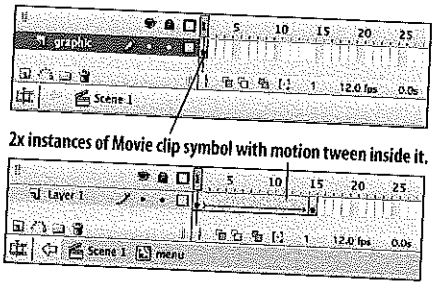
```
myHouse.kitchen.cupboardUnderSink.flySpray.play();
```

Which will play (spray!) the fly spray **inside** the cupboard under the sink **inside** the kitchen **inside** my house.

All of the *Movie Clip* instances which are involved in this kind of script need to have *instance* names so that ActionScript can take to them.

Try this now:

1. Create a new file and in it, make a new *Movie clip* symbol.
2. **Inside** that *Movie clip* symbol, create an animation of some sort.
3. Place that Movie *clip symbol* on the *Stage* twice (see right). When you test this movie, both animations will play.
4. On the main *Stage*, give each instance of the *Movie Clip* symbol its own instance name.
5. Create a new 'actions' layer and open the *Actions* window.
6. Instead of adding a simple `stop();` action (which stops the *Timeline* it is sitting on), add the following script:
   ```
   menu1_mc.stop();
   ```
   which stops the *Timeline* of the *Movie clip* symbol with the instance name 'menu1'. Of course you must make sure that your instance name matches up with the script!
7. When you test this movie, you will see that only one of the instances has stopped.
8. To make both of the instances stop, add another line so that your script reads:
   ```
   menu1_mc.stop();
   menu2_mc.stop();
   ```
9. So with that knowledge, it should be straightforward to add a button to the *Stage* which will make only one instance of the *Movie clip* play:
   ```
   menu1_btn.addEventListener(MouseEvent.CLICK, startMenu1);
   function startMenu1(event){
     menu1_mc.play();
   }
   ```



2x instances of Movie clip symbol with motion tween inside it.

tip Reserved names

When you are naming instances and functions, beware of any name that turns blue in the *Actions* window—this indicates a word that is already part of ActionScript, so you should change your name. For instance **myName** is fine, **name** is not.

Commenting code

From this point onwards, your ActionScripts will get more and more complex. Commenting is a feature common to all programming languages and allows you to add notes to yourself into the code so that when you look at it again in several weeks' time, you can easily see what you meant to do.

To add comments to your code, simply add // to the start of a line of text (see above). The comment will go grey and Flash will ignore it. You can write whatever you like as comments and you can write as many comments as you like if it helps you to understand the ActionScript!

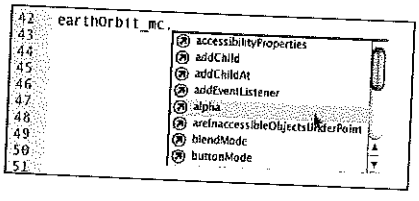
```
//Make both menus stop initially
menu1_mc.stop();
menu2_mc.stop();

//listener-function to play menu1
menu1_btn.addEventListener(MouseEvent.CLICK, startMenu1);

function startMenu1(event){
    menu1_mc.play();
}

//listener-function to play menu2
menu2_btn.addEventListener(MouseEvent.CLICK, startMenu2);

function startMenu2(event){
    menu2_mc.play();
}
```



As you type code, Flash will pop-up a list of related code if you have used '_mc' and '_btn' as instance names. You can either: press Return/Enter to insert the highlighted code, double-click another option in the list, or just ignore it and continue typing.