Chapter 2: Writing Scripts

Introduction to script writing

This section introduces you to some simple examples of writing movie clip scripts. It emphasizes where you place scripts, as script placement determines when a script gets called. Scripts are placed at three locations. These are:

- Script keyframes
- Event handlers
- State change handlers

In addition, this section discusses labels, which are frequently used in conjunction with scripting.

The section begins with a brief overview of the Script Editor user interface. To acquaint you with the functionality provided by the Script Editor, each example is presented as an exercise that you can work through yourself. You are also introduced to target addressing and *some* basic JavaScript syntax, although a tutorial on JavaScript basics is beyond the scope of this guide. Understanding JavaScript is a prerequisite if you want to do any serious LiveMotion scripting.

Script Editor overview

???This section will be updated at UI freeze.

You will be using the Script Editor to write your scripts and to locate information. Figure 2.1shows the Script Editor window. The callouts identify its main functionality.

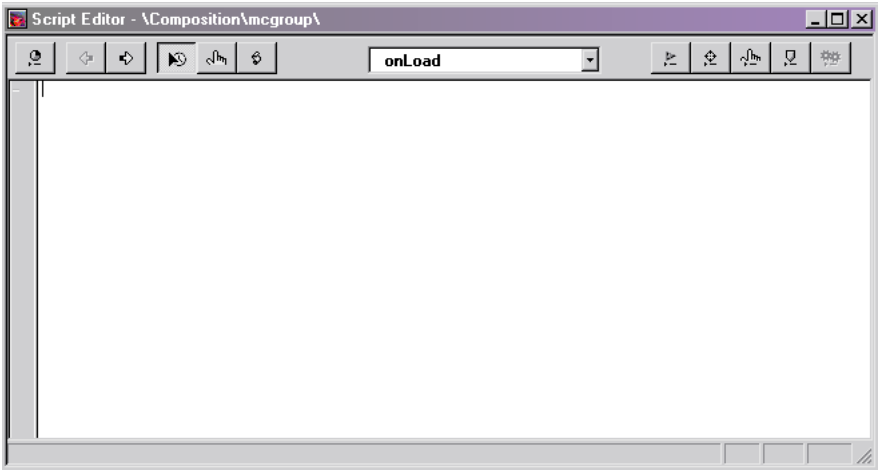

Figure 2.1 *Script editor main window*

Table 3.1 briefly describes each of the control buttons and windows shown in the Script Editor window.

Table 2.1 *Script editor controls and windows*

Button or window	Description
Movie Clip Browser	Lists all the movie clips in a composition in the movie clip hierarchy.
Previous Script	Takes you to the previous script.
Next Script	Takes you to the next script.
Event Handlers	A drop-down list of all the event handlers for which you can write scripts.
State Script	A drop-down list of all states defined for the *current* movie clip (movie clip whose timeline is open). The list contains the normal state, and it can include the rollover states over, down, and out, plus any custom states defined for the movie clip.

Button or window	Description
Keyframe Script	Used to write scripts to the keyframe displayed in the drop-down list.
Drop-down list	Displays the keyframes, event handlers, or states for the current movie clip.
Player Scripting DOM	Lists and describes: • all the global objects and properties in the JavaScript core that are supported by Player scripting • all predefined objects, their methods, and properties in the Player scripting DOM • all the LM 1.0 behaviors
Target Browser	Lists all the movie clips in hierarchical order. Composition is at the top. The movie clips on the composition timeline are one indent from the left margin. Any movie clips on the timelines of these movie clips are two indents from the left margin, and so on.
State Browser	Lists all the movie clips with all of their associated states including the normal (default) state.
Label Browser	Lists all the labels in a composition by the movie clip with which they are associated.
Automation Scripting DOM	Lists and describes all the global objects and properties in the JavaScript core that are supported by automation scripting and all predefined objects, their methods, and properties in the Automation scripting DOM. For details on automation scripts, see the LiveMotion 2.0 SDK.
Composition Window	Window in which you write and edit scripts.

Using labels

What are labels?

A label is a string identifier, or name, that references a frame in a timeline. You can use labels as arguments in scripts that you write. You could, for example, create a label called "right here" on a particular frame. With the label in place, you can write a script that sets the current frame of a timeline to the frame marked with the label "right here." Labels don't have to be used in scripts; they can be used simply to annotate a timeline. For example, you could apply the label “Accelerate” to a frame to identify where an object appears to pick up speed.

How are labels created?

To create a label,

1. Display the timeline to which you want to add a label.
2. Move the current-time marker to the frame to which you want to attach a label.
3. Click the Labels button in the timeline.
4. Enter a name for the label. The label appears on the timeline at that frame.

You can duplicate, rename, move, or delete labels. See the User Guide for details.

Using a label in a script

For examples of using labels in scripts, see “Using script keyframes on the composition timeline” on page 25 and “Writing a keyframe script to the movie clip timeline” on page 27.

Using script keyframes

What are script keyframes?

A *script keyframe* is a frame in a timeline to which a Player script is added. When the Player head enters that frame during playback, the script executes.

How do you create a script keyframe?

To add a script to a keyframe,

1 Navigate to the timeline where you want to add the script keyframe.

2 In the Timeline window, move the current-time marker to the specified frame.

3 Optionally, click the Labels button, and enter a name for the point in time where the script will be attached to the timeline. See Figure 3.2.

4 Click the Scripts button on the timeline to create a script keyframe at the current-time marker. This also opens the Script Editor.

Using script keyframes on the composition timeline

This example uses script keyframes and a label to move a movie clip horizontally across the composition.

To use script keyframes on the composition timeline,

1 Create a new document in LiveMotion.

2 Bring up the Timeline window by selecting Timeline>Show Timeline Window from the main menu. Alternately, you can use Ctrl+T (Windows) or Command+T (Mac OS).

3 Create an ellipse in the Composition window. Select the object in the Composition window or from the Timeline window.

4 Click the "Make Movie Clip" button in the Timeline window to make this object into a movie clip. Alternately, you can select Timeline>Movie clip from the main menu. A movie clip icon appears to the left of Ball in the Timeline window. To be scriptable, an object must be converted into a movie clip.

5 Provide a name for the movie clip. Select the object, press Enter, and type in the name "Ball." Press OK or Enter.

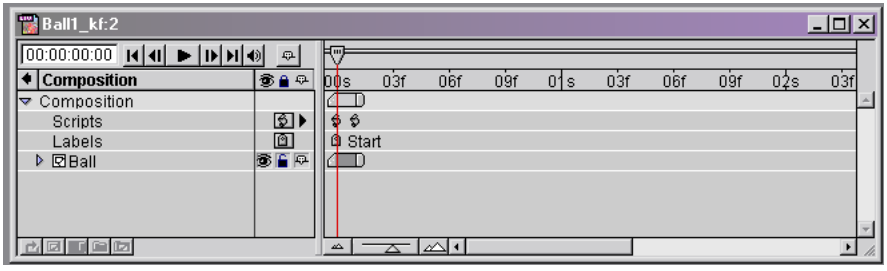

Figure 2.2 Composition timeline showing Scripts and Labels buttons and Ball movie clip

6 In the composition timeline, be sure the current-time marker is set to frame 0.

7 Click the Scripts button in the timeline to add a script keyframe at frame 0. This also brings up the Script Editor.

In the Script Editor window displayed, you can add scripts to the script keyframe you just created.

8 Write a script to the script keyframe that will move the Ball movie clip 5 pixels to the right. Here is a script that does this:

```
_root.Ball._x += 5;
```

In the script, `_root.Ball` is the absolute path to `Ball`. `_root` represents the composition timeline. All movie clips placed on `_root`'s timeline can be accessed by name as properties of `_root`. Thus we can access `Ball` by saying `_root.Ball`. (For details on `_root` and absolute path, see "Movie clip addressing" on page 47.) `_x` is the horizontal position property of `Ball`. It is one of several movie clip built-in properties. (For details, see "Movie clip properties" on page 55.) The operator `(+=)` is just a shorthand way to write the code:

```
_root.Ball._x = _root.Ball._x + 5;
```

9 With the current-time marker still at frame 0, click the Labels button in composition timeline. Enter Start in the text box and click OK to create a label named "Start" at frame 0.

10 Move the current-time marker to frame 1, create a script keyframe, and enter the following code in the Script Editor window:

```
_root.gotoAndPlay("Start");
```

`gotoAndPlay()` is a function that jumps a movie clip's timeline to a specific label and plays the timeline. In this case, it jump s _root's timeline to the label Start.

11 Preview the movie clip.

When the composition is previewed, the script you added at frame 0 moves Ball 5 pixels to the right on the screen. When execution reaches frame 1, the `gotoAndPlay` statement moves the current-time marker to the frame labeled "Start" (in this case frame 0) and plays the timeline. At this point the script on frame 0 executes again.

You can adjust the speed of the ball by changing the value added to _x in the script to a new value.

This concludes your first Player script!

Writing a keyframe script to the movie clip timeline

This example displays the same result. The difference is that you write the script to a different location: to the movie clip's own timeline rather than to the composition timeline. In the previous example, `_root` moved the Ball movie clip. In this case, the movie clip moves itself.

To write a keyframe script to the timeline,

1 Repeat steps 1 through 5 of "Using script keyframes on the composition timeline" on page 25 to create a movie clip and name it Ball.

2 Double click Ball in the composition timeline to open its own timeline. In the movie clip's timeline, be sure the current-time marker is set to frame 0. **???Figure to be corrected and updated at UI freeze**

.

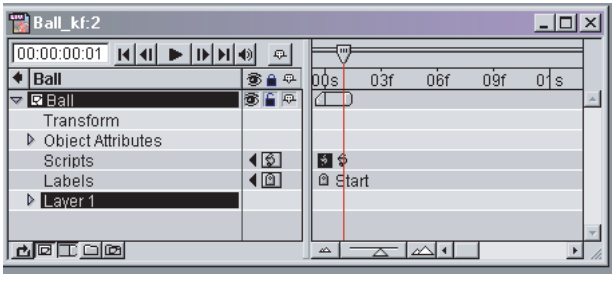

Figure 2.3 Ball movie clip timeline

3 Click the Scripts button in the movie clip's timeline to insert a scripts keyframe at frame 0. This also brings up the Script Editor

4 Write a script in the Editor window that will move Ball 5 pixels to the right.

```
this._x += 5;
```

The keyword `this` refers to the movie clip to which the script is attached—in this case, the movie clip Ball. Thus, the statement is incrementing Ball's horizontal position property.

You can also use the absolute path just as you could in the previous example in "Using script keyframes on the composition timeline" on page 25:

```
_root.Ball._x += 5;
```

However, if the object hierarchy for Ball changes (that is, Ball is placed in a movie clip *group*), the absolute path would no longer be valid. (For details on how movie clip groups change the object hierarchy, see "Some differences between movie clips and movie clip groups" on page 46.)

5 With the current-time marker still at frame 0, click the Labels button in the movie clip's timeline. Enter Start in the text box, and click OK to add the label "Start" to frame 0.

6 Move the current-time marker to frame 1, create a script keyframe, and enter the following code in the Script Editor window:

```
this.gotoAndPlay("Start");
```

7 Preview the movie clip. `Ball` moves across the screen just as it did in the previous example. The movie clip advances its horizontal position with each successive execution of the script.

Using event handlers

What are event handlers?

An event handler is script that is run as a result of a user action or a system-based event. For example, you can write an event handler that executes every time the user presses the mouse button or passes the mouse over the movie clip. System-based events such as onLoad and onData occur as a result of composition playback or loading variables into a movie clip.

Table 2.2 lists all the event handlers and describes the events they handle.

Table 2.2 Movie clip events

Event handler	Event
onLoad	First appearance of a movie clip in the composition. You can write scripts here to initialize and declare variables and to write functions.
onUnload	Removal of movie clip from the composition.
onEnterFrame	When the playhead first enters a frame before the frame is rendered.
onMouseMove	Any movement of the mouse pointer while the movie clip is in the composition.
onMouseDown	Pressing the mouse button while the movie clip is in the composition.
onMouseUp	Releasing the mouse button while the movie clip is in the composition.
onKeyDown	Pressing a key while the movie clip is in the composition.
onKeyUp	Releasing a key while the movie clip is in the composition.
onData	When the loading of variables into a movie clip is complete or a portion of a loaded movie completes loading into a movie clip.
onButtonPress	Clicking the mouse button while on the movie clip.
onButtonRelease	Releasing the mouse button while on the movie clip.
onButtonReleaseOutside	After pressing the mouse button and holding it while on a movie clip, moving the mouse outside the movie clip and releasing the button.
onButtonRollOver	Moving the mouse on the movie clip.
onButtonRollOut	Moving the mouse off the movie clip.
onButtonDragOver	Pressing the mouse button while moving the mouse on the movie clip.
onButtonDragOut	After pressing the mouse button while on a movie clip and holding it, moving the mouse off the movie clip.

How do you attach a script to an event handler?

To attach a script to an event handler,

1 Select a movie clip in the timeline or in the composition.

2 Choose Scripts>Editor to open the Script Editor. Alternately, you can use Ctrl+J (Windows) or Command+J (Mac OS).

3 Select the script handlers button.

4 Select the handler for which you want to write a script.

5 Write the script in the Script Editor window.

Event handler example.

This example adds the same movement to the movie clip Ball as the previous keyframe script examples did. See "Using script keyframes on the composition timeline" on page 25 and "Writing a keyframe script to the movie clip timeline" on page 27. However, it uses an event handler to call the script that moves Ball.

To use an event handler,

1 Repeat steps 1 through 5 of "Using script keyframes on the composition timeline" on page 25 to create a movie clip and name it Ball.

2 Double click Ball to open the movie clip's timeline.

3 Choose Scripts>Editor to open the Script Editor.

4 In the Script Editor, click the Event Handlers button to display the drop-down list of event handlers.

5 Select onEnterFrame, and enter this script to move Ball horizontally.

```
this._x += 5;
```

This script causes Ball to move itself each time the playhead enters a frame.

6 Preview the composition. The ball moves horizontally across the Composition window.

7 Save this `.liv` file for the next exercise.

Building on the previous example

This example builds on the previous one. It uses Ball's onLoad handler to explicitly set the horizontal starting position of Ball and to initialize a property containing the speed that Ball will move. For this example, open the `.liv` file that you created in the previous exercise.

To build on the example above,

1 Select Ball and open the Script Editor.

2 Click the Event Handlers button and select the onLoad handler from the drop-down list. Enter this script:

```
this._x  = 100; //sets the initial position of Ball
this.speed = 5;
```

The first statement in this script sets the initial horizontal position of Ball to 100. The second creates a new property of Ball called `speed` and assigns it the value 5.

3 With the event handlers button still toggled down, select onEnterFrame from the same drop-down list that contained onLoad. This brings up the event handling script that moves the Ball clip.

```
this._x += 5;
```

Change the script to:

```
this._x += speed;
```

4 Preview the results. Except for setting Ball's initial position, the behavior is the same as in the previous exercise: Ball moves horizontally across the Composition window.

5 As another variation, you can modify the onEnterFrame event handler to do a bounds check to make sure `Ball` doesn't move out of the Composition window. Here is the code:

```
this._x += speed;

if(this._x > 550)
    this.speed = -5;
if(this._x < 0)
    this.speed = 5;
```

Preview the results. The ball moves back and forth horizontally across the Composition window. You should adjust the value 550 to reflect your Composition window's actual width. Check Composition Settings to determine the width.

What are state scripts?

Thus far, the examples in this section have illustrated attaching scripts to:

- The composition timeline using its Labels and Scripts buttons
- Movie clip timelines using its Labels and Scripts buttons

- Event handlers

From working with the user interface, recall that you can create rollover states for an object. Scripts also can be attached to these states. A script is executed each time the object changes to the state to which the script is attached.

How do you attach scripts to states?

To attach a script to a state,

1 Select the object.

2 Open the States palette to view the movie clip states.

3 In the States palette, select the movie clip state to which you want to add a script.

4 Click the Scripts button in the palette.

This opens the Script Editor, with the correct state script displayed.

5 Write the script in the Script Editor window.

State script example

This example is similar to the keyframe examples you have created so far. Using the Rollovers palette, you create an over state, which, for effect, you can change to a different color. Then you write a script that moves the Ball one direction in the normal state and another, in the over state

To create the keyframe example,

1 Repeat steps 1 through 5 of "Using script keyframes on the composition timeline" on page 25 to create and name a movie clip named Ball.

2 Using the States palette, create an over state for the movie clip. Give it a different fill color, so you can more easily recognize the movement in the over state during playback.

3 In the Rollovers palette, select the over state; then click the Scripts button.

This opens the Script Editor, with the state scripts button down and the script for the over state displayed (in this case there's nothing there).

4 In the Script Editor window, enter the following code to move the movie clip 5 pixels to the right:

```
this._x += 5;
```

5 Select the normal state, and enter the following code:

```
this._y += 20;
```

This moves the movie clip vertically.

6 Preview the composition.

Ball first appears in its normal state. It does not move until you first pass the mouse over it. Try this a few times. Each time the mouse is moved over Ball, Ball moves five pixels to the right. Moving the mouse off Ball causes the movie clip to return to its normal state. Each time Ball enters its normal state, it moves vertically downward 20 pixels.