How to get started with ActionScript

ActionScript 3.0 is the scripting language of Adobe Flash CS4. You can use ActionScript to add complex interactivity, playback control, and data display to your application. For example, you might want to animate a picture of a boy walking. By adding ActionScript, you could have the animated boy follow the pointer around the screen, stopping whenever he collides with a piece of animated furniture.

ActionScript is an object-oriented programming language. *Object-oriented programming* is a way to organize the code in a program, using code to define objects and then sending messages back and forth between those objects.

You don't have to be a programmer to take advantage of ActionScript (see "Using Script Assist mode" later in this guide). But the following concepts will help:

- *Class:* The code that defines an object. This code consists of properties, methods, and events. Think of the blueprint of a house: you can't live in the blueprint, but you need it so you can build the house. You use classes to create *objects*.
- *Object:* An instance of a class. When you *instantiate* an object (create an instance of it), you declare what class it is created from and set its properties. You can create as many objects as you want from a single class—if you have a `bicycle` class, you can create many bicycle objects, each with its own properties (one bicycle might be red while another might be green).
- *Property:* One of the pieces of data bundled together in an object. A property helps define an object—it provides the object's characteristics. A song object might have properties named `melody` and `title`. You set the properties of an object when you create the object, but you can change them later as needed. A property is a *variable* that belongs to an object.
- *Variable:* A name that represents a value in the computer's memory. As you write code, you write the variable's name as a placeholder for the value. This allows you to write code even if you don't know all the possible values a visitor might provide. If you create a variable `firstName`, you can tell your program to display the visitor's first name without knowing in advance what the visitor's first name is.
- *Method:* An action that can be performed by an object. For example, the class `horse` might have a method called `gallop()`. When the method `gallop()` is called, it shows an animation of the horse galloping from one point to another.
- *Function:* A block of code that carries out specific tasks and can be reused in your program. For example, you might create a function called `checkEmail()` to verify that text typed by a visitor is a valid e-mail address. Each time a visitor provides an e-mail address, you can call `checkEmail()` to make sure the visitor has provided text that can actually be used as an e-mail address. If you ever want to update the function, you only have to do it once instead of in each place where e-mail addresses must be validated. You can also think of a method as a *function* attached to an object.
- *Event:* Something that happens in a Flash movie that ActionScript is aware of and can respond to. Many events are related to user interaction—for example, a user clicking a button or pressing a key on the keyboard. The technique for specifying certain actions that should be performed in response to particular events is known as *event handling*.

If you've worked with symbols in Flash, you're already used to working with objects. Imagine you've defined a movie clip symbol—say a drawing of a rectangle—and you've placed a copy of it on the Stage. That movie clip symbol is also an object in ActionScript; it's an *instance* of the MovieClip *class*. The main timeline of a Flash movie also belongs to the Movie Clip class.

You can modify various characteristics of any movie clip. When a movie clip is selected, the Property inspector shows some of the characteristics you can change, such as its X coordinate or its width. Or you can make color adjustments such as changing the clip's alpha (transparency). Other Flash tools let you make more changes, such as using the Free Transform tool to rotate the rectangle. All of the ways you can modify a movie clip symbol in the Flash authoring environment are also things you can do in ActionScript. In ActionScript, you use the *methods* of the MovieClip class to manipulate or change the *properties* of your movie clip.

For more about object-oriented programming, see *Programming ActionScript 3.0*, "Object-oriented Programming in ActionScript" (in Flash, select Help > Flash Help).

Using Script Assist mode

You can add ActionScript in the authoring environment by using the Actions panel (**Figure 1**). The Actions panel provides Script Assist mode to simplify the coding process.

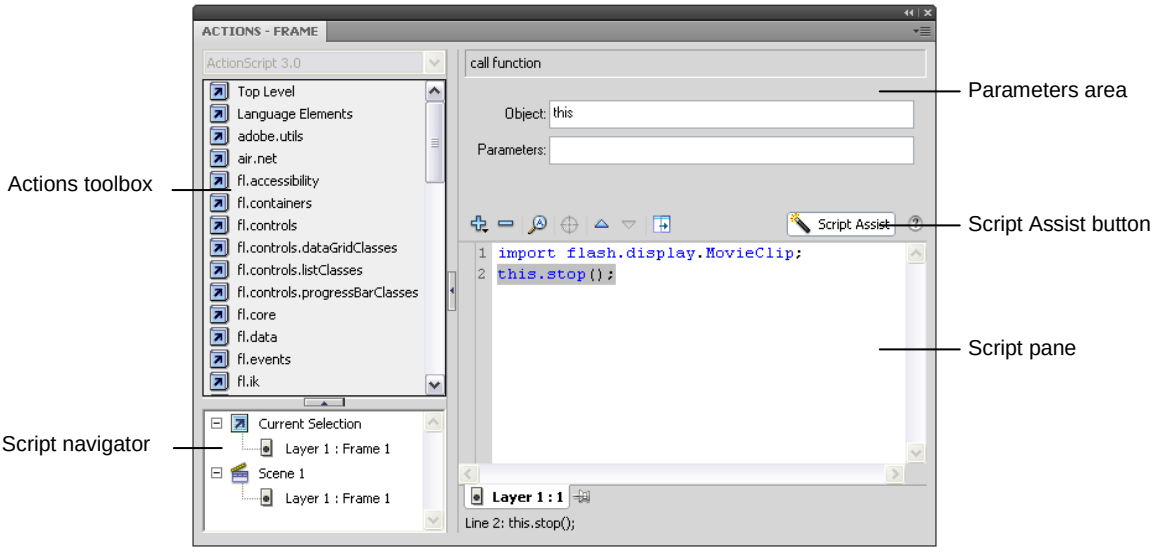

Figure 1 Actions panel in Script Assist mode

In Script Assist mode, you can add ActionScript to your Flash document without writing the code yourself. You select actions from the Actions toolbox and set the options for each action in the parameters area. You must know a little about what functions to use to accomplish specific tasks, but you don't have to learn syntax (the grammar of ActionScript). Many designers and nonprogrammers use this mode.

One of the first things to learn is how to stop your movie at a certain spot. You will also learn how to send the playhead to a particular frame in the movie.

Using ActionScript to stop a movie

1. Start Flash and open a movie. Create a layer in your movie named **actions**. In the frame that corresponds to the end of your movie, insert a new keyframe.
2. Select Window > Actions to display the Actions panel.
3. If you don't see the parameters area in the Actions panel, click the Script Assist button in the upper-right corner (**Figure 2**).

   Classes are organized into packages. You want to add a `stop()` action to a movie clip (in this case, your timeline is the movie clip), so you must locate the Movie Clip class. The Movie Clip class is part of the Flash.Display package.
4. In the Actions toolbox on the left side of the Actions panel, scroll down and click the Flash.Display package to display the classes it contains.
5. Scroll down again to find the Movie Clip class and click to expand it (**Figure 3**).
6. Click Methods to view the methods available for the Movie Clip class.

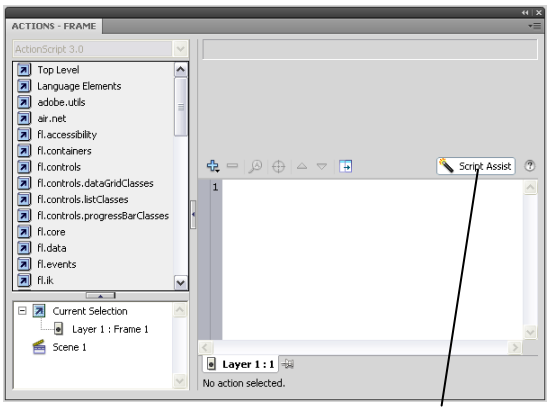

Figure 2 Actions panel

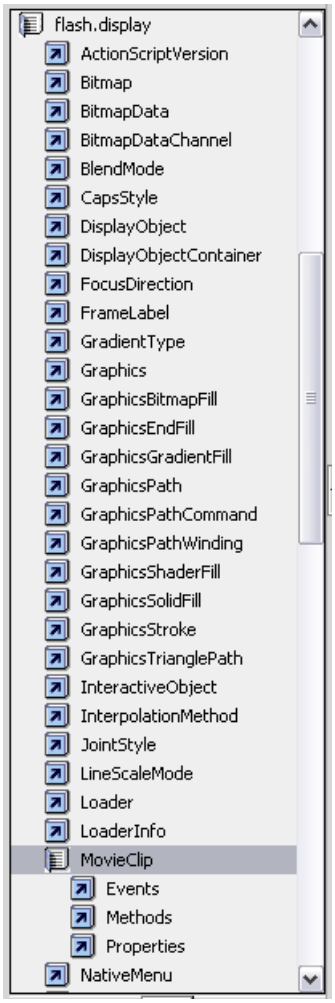

Figure 3 Movie Clip class

7. Scroll down to the Stop method. Do one of the following:

   - Double-click the Stop method.
   - Drag the Stop method into the Script pane.

   Code for applying the Stop method appears in the Script pane (**Figure 4**). The first line (`import flash.display.MovieClip();`) imports the code necessary for the Movie Clip class. The second line is the Stop action itself. The red code `not_set_yet` indicates you should use Script Assist to finish the code.

8. Click in the Object field in the parameters area of the Actions panel.

   The Insert Target Path button is now active. The target path helps you locate the object you are trying to control.

9. Click the Insert Target Path button.

   The Insert Target Path dialog box appears (**Figure 5**).

10. Select the Relative option and click Root.

    This sets the target path to `this`.

11. Click OK to close the Insert Target Path dialog box.

    The completed script for the Stop method appears in the Script pane (**Figure 6**). This code will cause your movie to stop playing at the end of the movie, frame 40.

    Note: In ActionScript, `this` is used the same way that you would refer to yourself as "me" instead of using your full name. Remember that the main timeline is an instance of the MovieClip class. In Figure 6, Flash uses `this` to refer to the movie clip that frame 40 belongs to.

12. Close the Actions panel.

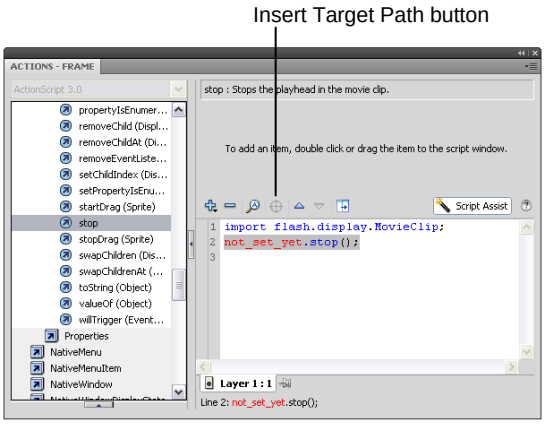

Figure 4 Stop method

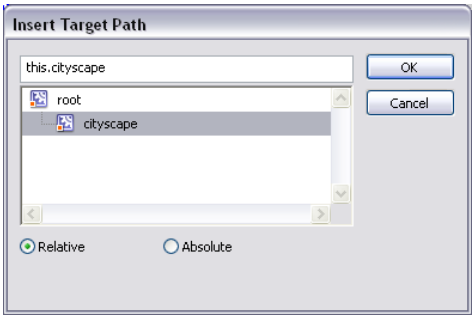

Figure 5 Insert Target Path dialog box

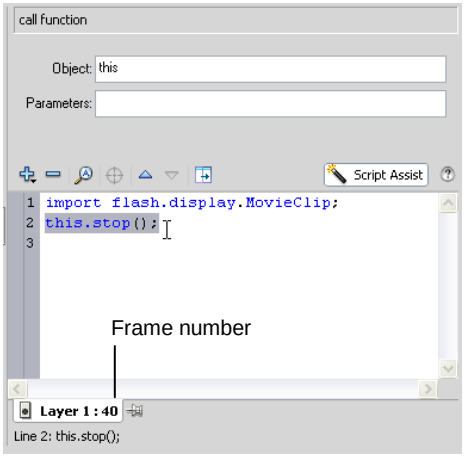

Figure 6 Stop method applied to frame 40

Event handling

The technique for specifying certain actions that should be performed in response to particular events is known as *event handling*. Event handling consists of three important elements:

- *The event source:* Which object will trigger the event? For example, which button will be clicked, or which Loader object is loading the image?
- *The event:* What is going to happen, what interaction do you want to respond to? Identifying the event is important, because an object can trigger (and listen for) several events.
- *The response:* What action(s) do you want performed when the event happens?

When an ActionScript program is running, Adobe Flash Player just sits and waits for certain events to happen, and when those events happen, the player runs the specific ActionScript code you've specified for those events. For the program to know what events are important, you must create an event listener. An *event listener* is a function Flash Player executes in response to specific events.

Adding an event listener is a two-step process:

- First, you create a function or class method for Flash Player to execute in response to the event. This function is sometimes called the *listener function*.
- Second, you use the addEventListener() method to connect the listener function with the target of the event. The addEventListener() function tells Flash what object to listen to, what event to listen for, and what function to execute in response.

Using ActionScript to go to another frame

1. Create a button users can click to go to a particular frame in your movie. Make sure you place the button on the Stage (**Figure 7**).
2. Select the button and use the Property inspector to give the button a unique instance name (**Figure 8**).
3. In the main timeline of your movie, create a layer named **actions**.
4. Create a keyframe in the actions layer that corresponds to the keyframe where your button first appears on the Stage. Select this keyframe in the actions layer.

   Note: If your button doesn't appear in this frame, Flash will generate an error message because you are referring to an object that isn't on the Stage yet.
5. Select Window > Actions to display the Actions panel.
6. If you don't see the parameters area in the Actions panel, click the Script Assist button.
7. In the Actions toolbox on the left side of the panel, select the AddEventListener method from the IEventDispatcher class.

   To find the AddEventListener method, open flash.events, and then open IEventDispatcher.

Figure 7 Button instance

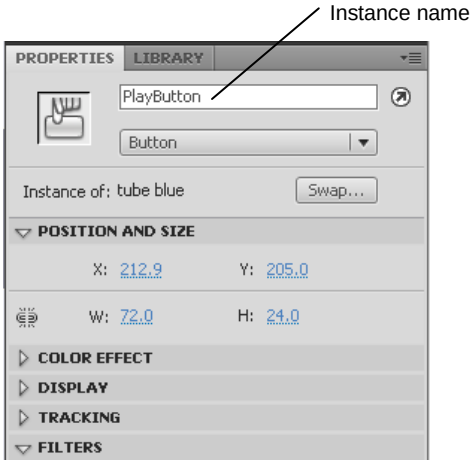

Figure 8 Button instance

8. Double-click the AddEventListener method to add it to the Script pane (**Figure 9**).
9. Click in the Object field in the parameters area of the Actions panel.

   The Insert Target Path button is now active.
10. Click the Insert Target Path button.

    The Insert Target Path dialog box appears (**Figure 10**).
11. Select the Relative option and select the instance name of your button. Click OK to close the dialog box.

    The event listener is attached to the instance of your button (**Figure 11**).

    Next, you select an event to listen for.

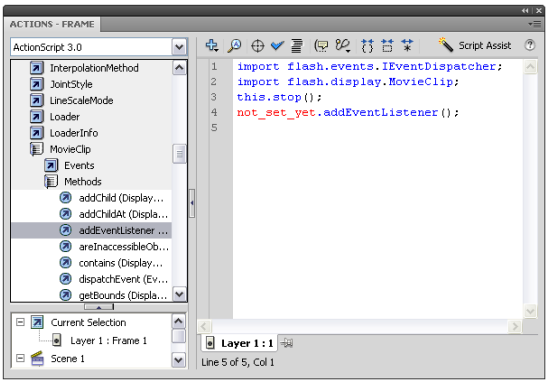

Figure 9 AddEventListener method in the Actions toolbox

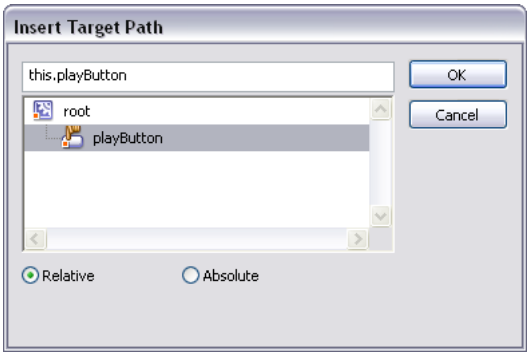

Figure 10 Insert Target Path dialog box

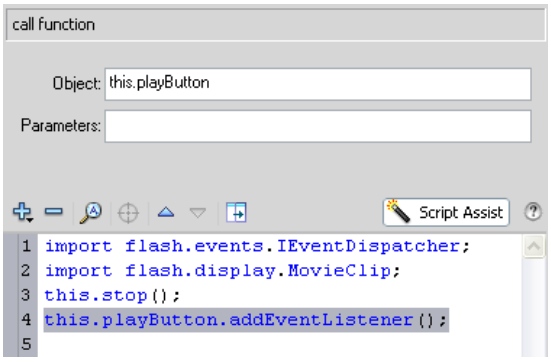

Figure 11 Event listener code in the Script pane

12. In the Actions toolbox on the left side of the panel, select the CLICK property from the MouseEvent class (**Figure 12**).

To find the CLICK property, open Flash.Events, open MouseEvent, and open Properties.

13. In the Script pane, select the AddEventListener() line to display the parameters for this method. Then, in the parameters area, click in the Type field.

14. In the Actions toolbox, double-click the CLICK property.

Script Assist adds the property to your code as `MouseEvent.CLICK` (**Figure 13**).

Now your code will listen for a click on the button. To tell the event listener how to respond when that click occurs, you next create a function.

15. In the parameters area, type a name for your function in the Listener field. You can use any name you like, but make sure the name of the function is unique and contains no spaces (**Figure 14**).

The function name appears in the Script pane as you type. You have named the function, but you haven't created it yet.

16. In the Actions toolbox, select the `function` keyword from Language Elements.

To find the `function` keyword, expand Language Elements. Then expand Statements, Keywords & Directives, and expand Definition Keyword.

17. Double-click the **function** keyword in the Actions toolbox.

The code for creating a function appears in the Script pane (**Figure 15**).

18. In the parameters area of the Script pane, type the name of your function in the Name field.

Note: The function name must be typed exactly as you typed it for the AddEventListener function in step 15. Function names are case-sensitive.

19. In the parameters area of the Script pane, type **event:MouseEvent** in the Parameters field.

In this field, you are naming a variable (**event**) and indicating what type of variable it is (**MouseEvent**).

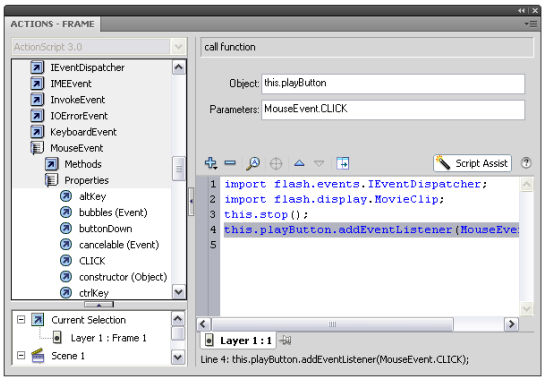

Figure 12 CLICK property in the Actions toolbox

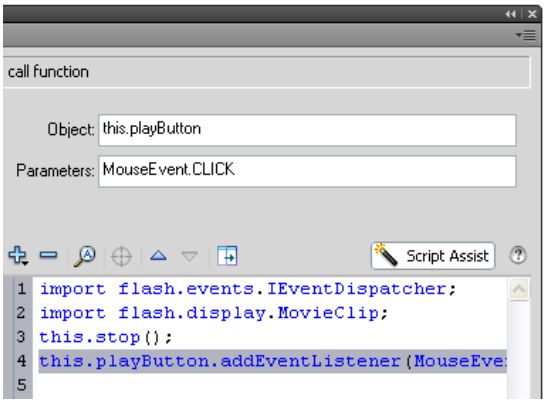

Figure 13 CLICK property in the Script pane

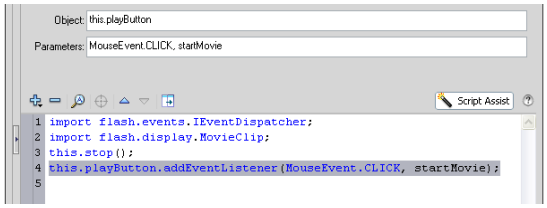

Figure 14 Function name in the Listener field

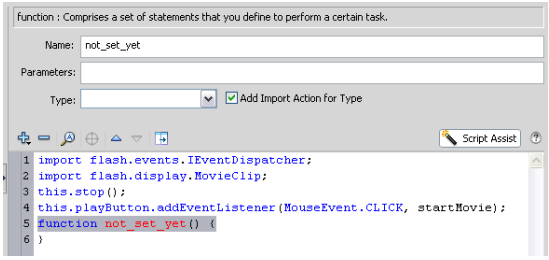

Figure 15 Function code

20. In the parameters area of the Script pane, select Void from the Type pop-up menu.

Some functions return a value when called. The keyword `void` indicates that this function does not return a value.

Now you can tell the function what you want it to do when the CLICK event occurs.

21. In the Actions toolbox on the left side of the panel, select the GotoAndPlay method for the Flash.Display class (**Figure 16**).

To find the GotoAndPlay method, expand Flash.Display, and then expand MovieClip.

22. Select the function in the Script pane and double-click the GotoAndPlay method in the Actions toolbox.

The method is added to the function (**Figure 17**).

23. Click in the Object field in the parameters area of the Actions Panel.

The Insert Target Path button is now active.

24. Click the Insert Target Path button to display the Insert Target Path dialog box.

25. Select the Relative option and select the movie clip you want to play when the button is clicked. If you want the movie in the main timeline to play, select Root to set the target path to `this`.

26. Click OK to close the Insert Target Path dialog box.

27. In the Frame field, type the number of the frame you want to send the playhead to. For example, if you want the movie to start from the beginning, type the number **1** to play the movie's first frame.

28. Close the Actions panel.

29. Save the movie.

30. Select Control > Test Movie to test the movie.

31. Select File > Close to close the preview window.

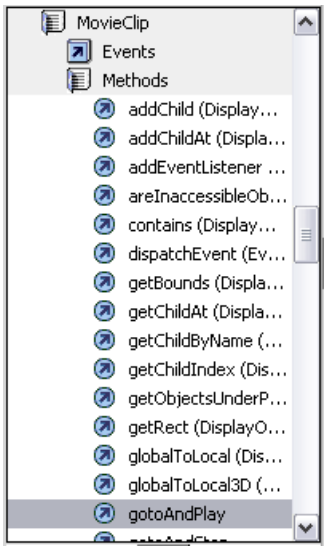

Figure 16 GotoAndPlay method

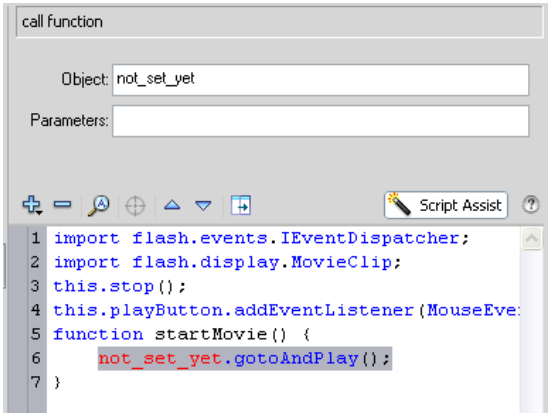

Figure 17 GotoAndPlay method in the Script pane

For more about ActionScript, see *Programming ActionScript 3.0*, "Getting Started with ActionScript" (in Flash, select Help > Flash Help).

