

ADOBE® FLASH® CS4 PROFESSIONAL



CLASSROOM IN A BOOK™

The official training workbook from Adobe Systems

CD-ROM Included for Windows and Mac OS



Chapter 6. Creating Interactive Navigation.....	1
Getting Started.....	3
About Interactive Movies.....	4
Designing a Layout.....	4
Creating Buttons.....	8
Understanding ActionScript 3.0.....	20
Adding a Stop Action.....	24
Creating Event Handlers for Buttons.....	25
Creating Destination Keyframes.....	30
Playing Animation at the Destination.....	35
Animated Buttons.....	41
Review Questions.....	0
Review Answers.....	0

6

CREATING INTERACTIVE NAVIGATION

Lesson Overview

In this lesson, you'll learn how to do the following:

- Create button symbols
- Add sound effects to buttons
- Duplicate symbols
- Swap symbols and bitmaps
- Name button instances
- Write ActionScript to create nonlinear navigation
- Create and use frame labels
- Create animated buttons



This lesson will take approximately two and a half hours to complete. If needed, remove the previous lesson folder from your hard drive and copy the Lesson06 folder onto it.

Leon Kritch Photography



Let your viewers explore your site and become active participants. Button symbols and ActionScript work together to create engaging, user-driven, interactive experiences.

Getting Started

To begin, view the photography portfolio page that you'll create as you learn to make interactive projects in Flash.

- 1 Double-click the 06End.swf file in the Lesson06/06End folder to play the animation.



The project is an interactive Web page for a fictional photographer. Viewers can click a button to see an enlarged version of a photo. In this lesson, you'll create interactive buttons and structure the Timeline properly. You'll learn to write ActionScript to provide instructions for what each button will do.

- 2 Close the 06End.swf file.
- 3 Double-click the 06Start.fla file in the Lesson06/06Start folder to open the initial project file in Flash. The file includes several assets already in the Library and on the Stage.
- 4 Choose File > Save As. Name the file **06_workingcopy.fla** and save it in the 06Start folder. Saving a working copy ensures that the original start file will be available if you want to start over.

About Interactive Movies

Interactive movies change based on the viewer's actions. For example, when the viewer clicks a button, a larger version of an image is displayed. Interactivity can be simple, such as a button click, or it can be complex, receiving inputs from a variety of sources, such as the movements of the mouse, key presses from the keyboard, or even data from databases.

In Flash, you use ActionScript to achieve most interactivity. ActionScript provides the instructions that tell each button what to do when the user clicks one of them. In this lesson, you'll learn to create a nonlinear navigation—one in which the movie doesn't have to play straight from the beginning to the end. ActionScript can tell the Flash playhead to jump around and to go to different frames of the Timeline based on which button the user clicks. Different frames on the Timeline contain different content. The user doesn't actually know that the playhead is jumping around the Timeline—all the user sees (or hears) is different content appearing as the buttons are clicked on the Stage.

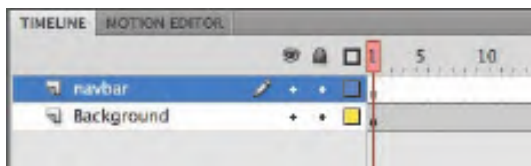
Designing a Layout

The first task when creating an interactive project like this is to design a layout that will accommodate the buttons and the content. The buttons are usually laid out in one section, called a navigation bar, or nav bar, for short. The nav bar remains constant while the content area changes, depending on which button is clicked. Space should also be allotted for a title and other global information. For this lesson, the Stage has already been set and a pleasant gradient shape has been added in a layer called Background.

Adding a Simple Design Element

You'll create a black rectangle that will serve to visually separate the title and the nav bar from the main content.

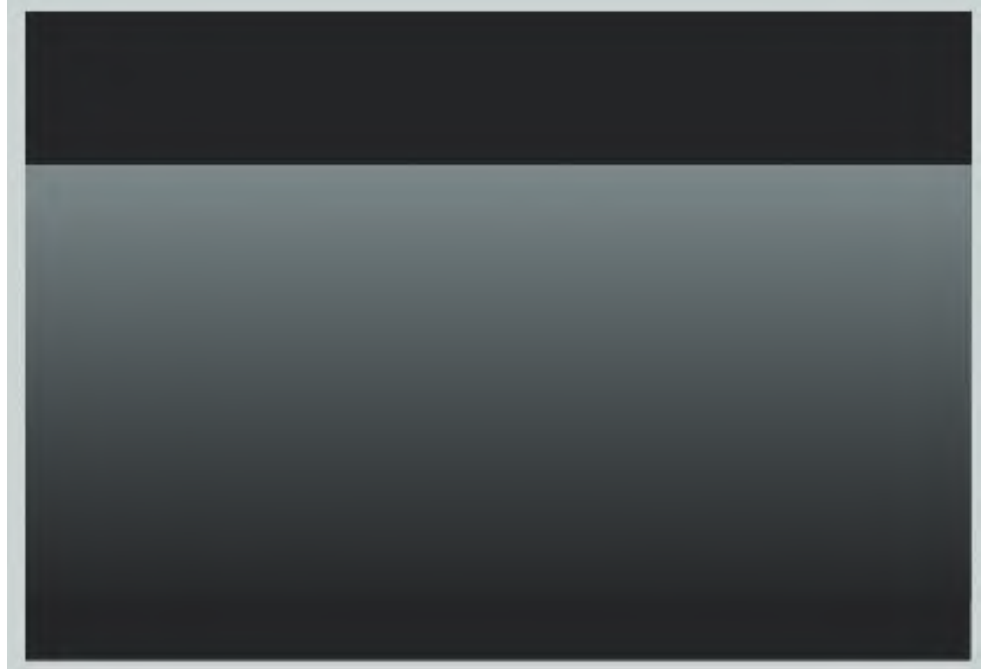
- 1 Select the existing layer.
- 2 Click the Insert New Layer button and rename it **navbar**.



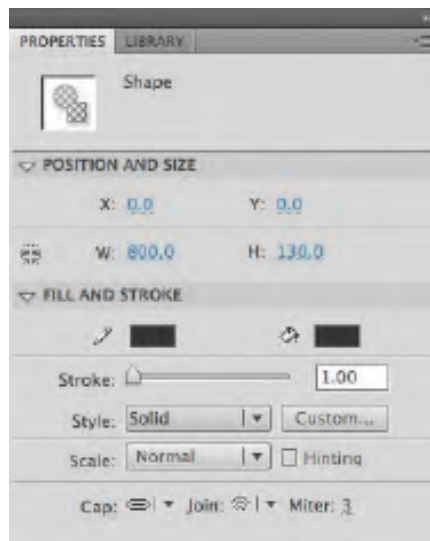
- 3 In the Tools panel, select the Rectangle tool.



- 4 Select a black fill and black for a stroke color.
- 5 Draw a rectangle on the Stage starting at the top corner of the Stage.



- 6 In the Property inspector, set the width to **800**, the height to **130**, the X value to **0**, and the Y value to **0**.

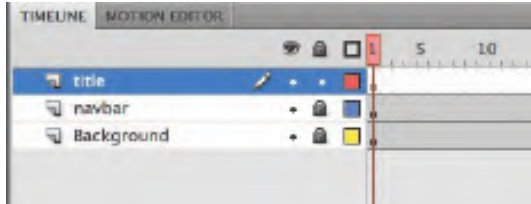


- 7 Lock the navbar layer so you don't accidentally move it.
A horizontal black bar sits atop a large gradient.

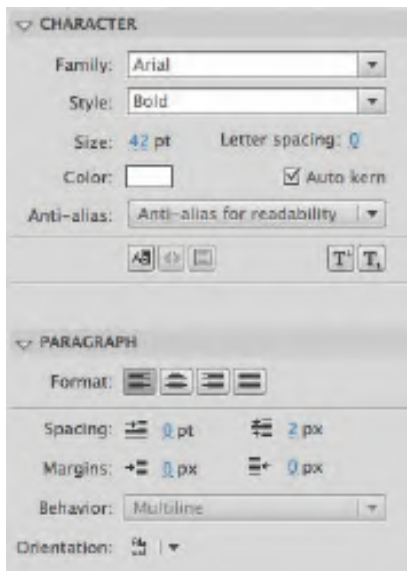
Adding a Title and Introductory Text

In this project, the title appears at the top of the Stage, no matter what else is displayed.

- 1 Select the topmost layer.
- 2 Click the Insert New Layer button and rename it **title**.

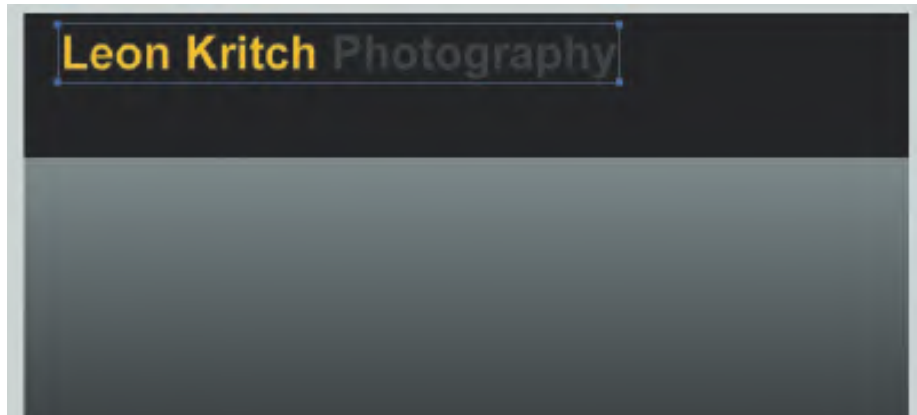


- 3 Select the Text tool and click an insertion point in the upper-left corner of the Stage.
- 4 Type **Leon Kritch Photography**.
- 5 Select all the text. In the Property inspector, select Static Text from the pop-up menu, select Arial for the typeface, and select 42 for the text size. Click the Left Align icon.

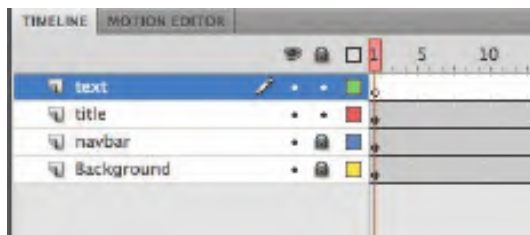


- 6 Select just *Leon Kritch* and change the text color to #FFCC00.
- 7 Select just *Photography* and change the text color to #333333.
- 8 Select the text with the Selection tool. In the Property inspector, set the X value to 30 and the Y value to 10.

The finished title appears with gold and gray lettering at the top of the black rectangle.



- 9 Click the Insert New Layer button again and rename it **text**.

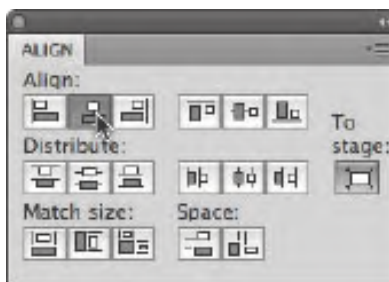


- 10 Select the Text tool again and drag out a text box on the Stage in the gradient area.
- 11 Enter some introductory text as in the 06End.swf file and as shown in the following figure. Make the title of the exhibition italic and a different color.



12 Select the text with the Selection tool and use the Align panel (Window > Align) to position the introductory text in the middle of the Stage. Make sure the To Stage option is selected before aligning the text.

13 Lock the title layer so you don't accidentally move it.



Creating Buttons

A button is the visual indicator of what the user can interact with. The user usually clicks a button, but many other types of interactions are possible. For example, something can happen when the user rolls the mouse over a button.

Buttons are a kind of symbol that have four special keyframes that determine how the button appears. Buttons can look like virtually anything—they don't have to be those typical pill-shaped gray rectangles that you see on many Web sites.

Creating a Button Symbol

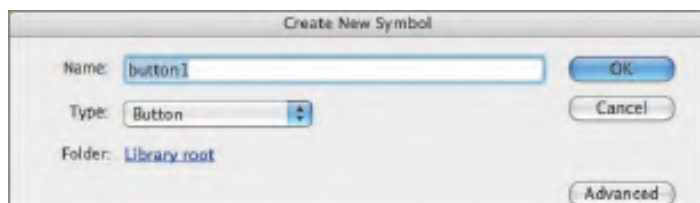
In this lesson, you'll create buttons with small thumbnail images. A button symbol's four special keyframes include:

- Up keyframe. Shows the button as it appears when the mouse is not interacting with it.
- Over keyframe. Shows the button as it appears when the mouse is hovering over the button.
- Down keyframe. Shows the button as it appears when the mouse button is depressed.
- Hit keyframe. Indicates the clickable area of the button.

You'll understand the relationship between these keyframes and the button appearance as you work through this lesson.

1 Choose Insert > New Symbol.

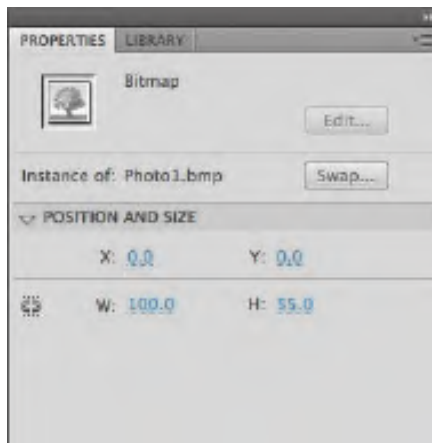
2 In the Create New Symbol dialog box, select Button and name the symbol **button1**. Click OK.



- 3 Flash brings you to symbol-editing mode for your new button.
- 4 In the Up keyframe of the Timeline, drag the small image Photo1.bmp from the Library to the middle of the Stage.

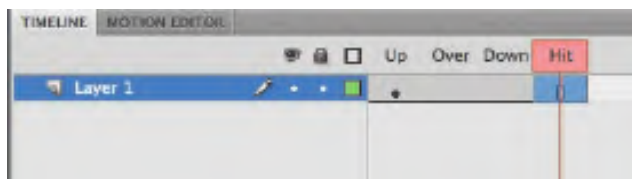


- 5 In the Property inspector, set the X value to 0 and the Y value to 0.



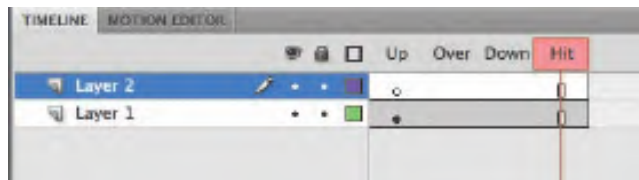
The upper-left corner of the small thumbnail image is now aligned to the center point of the symbol.

- 6 Select the Hit keyframe in the Timeline and choose Insert > Timeline > Frame to extend the Timeline.

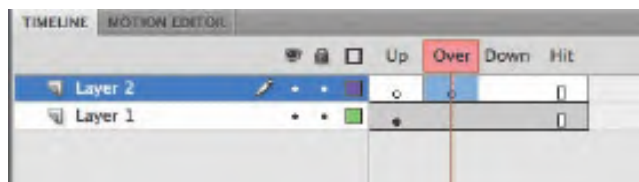


The peacock image now extends through the Up, Over, Down, and Hit keyframes.

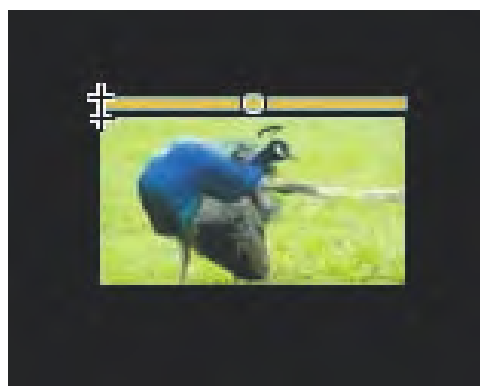
- 7 Insert a new layer.



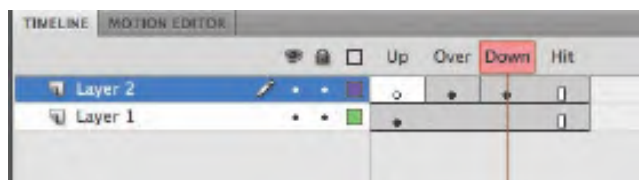
- 8 Select the Over keyframe and choose Insert > Timeline > Keyframe.
A new keyframe is inserted in the Over keyframe of the top layer.



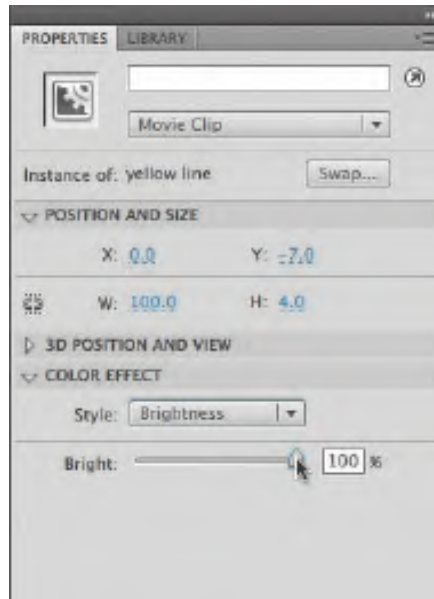
- 9 Drag the yellow line movie clip symbol from the Library to the Stage.



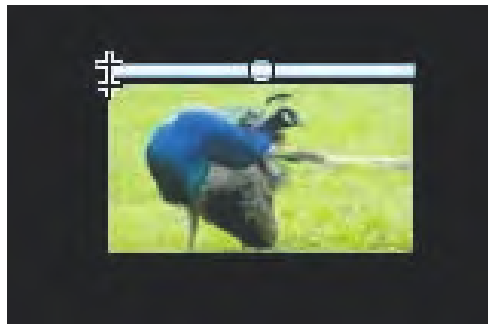
- 10 In the Property inspector, set the X value to 0 and the Y value to -7.
The yellow horizontal line appears over the thumbnail image whenever the mouse cursor rolls over the button.
- 11 Select the Down keyframe and choose Insert > Timeline > Keyframe.



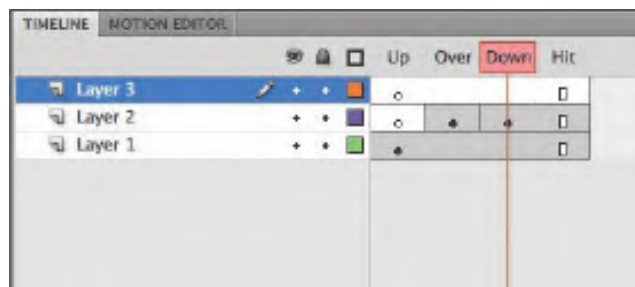
- 12 Select the yellow horizontal line, and in the Property inspector, expand the Color Effect section.
- 13 Choose Brightness from the Style pull-down menu and change the brightness value to 100%.



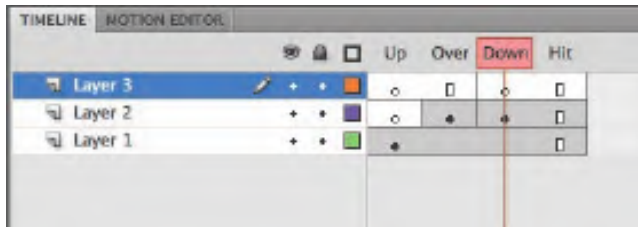
A bright, white, horizontal line appears whenever the mouse button is pressed over the button.



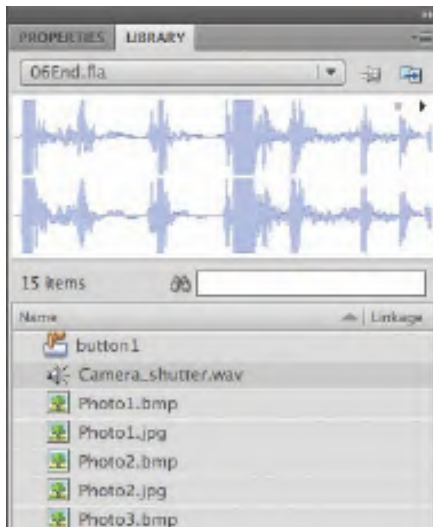
- 14 Insert a third layer.



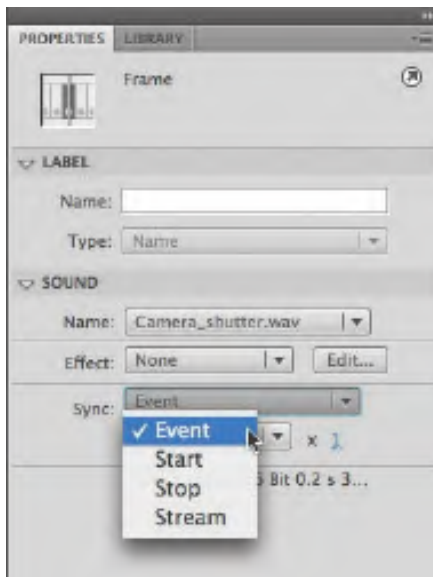
- 15 Select the Down keyframe on the new layer and choose Insert > Timeline > Keyframe.



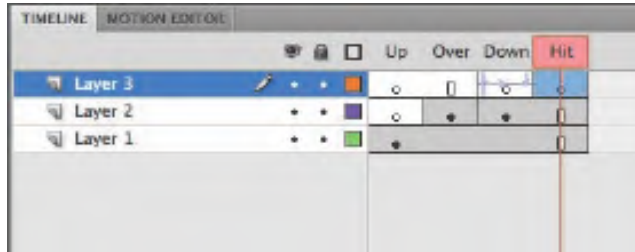
- 16 Drag the sound file called Camera_shutter.wav from the Library to the Stage.



- 17 Select the Down keyframe where the sound form appears, and in the Property inspector, make sure that the Sync is set to Event.



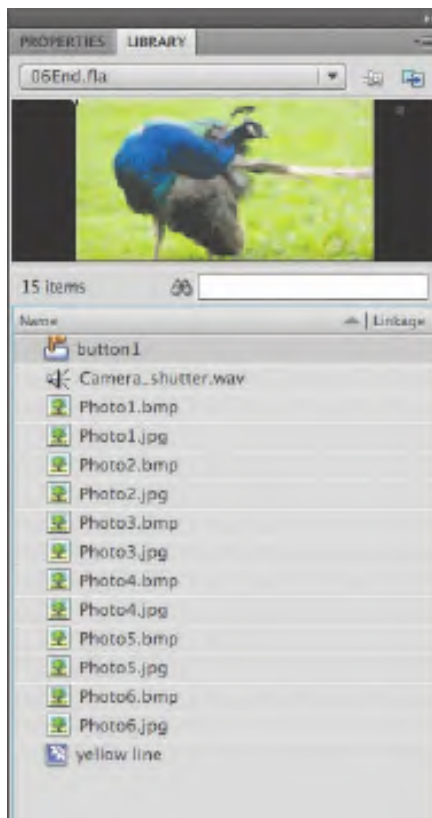
- 18 Select the Hit keyframe and choose Insert > Timeline > Blank Keyframe.



● **Note:** You'll learn more about sound in Chapter 7, "Working with Sound and Video."

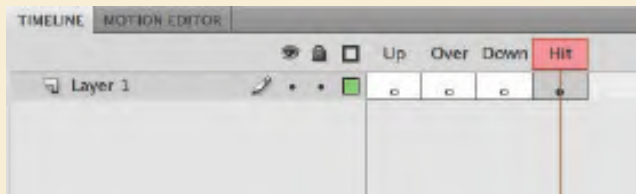
The sound of a camera shutter will play only when a viewer clicks the button.

- 19 Click Scene 1 above the Stage to exit symbol-editing mode and return to the main Timeline. Your first button symbol is complete! Look in your Library to see the new button symbol stored there.



Invisible Buttons and the Hit Keyframe

Your button symbol's Hit keyframe indicates the area that is "hot," or clickable, to the user. Normally, the Hit keyframe contains a shape that is the exact same size and location as the shape in your Up keyframe. In most cases, you want the graphics that users see to be the same area where they click. However, in certain advanced applications, you may want the Hit keyframe and the Up keyframe to be different. If your Up keyframe is empty, the resulting button is known as an invisible button.

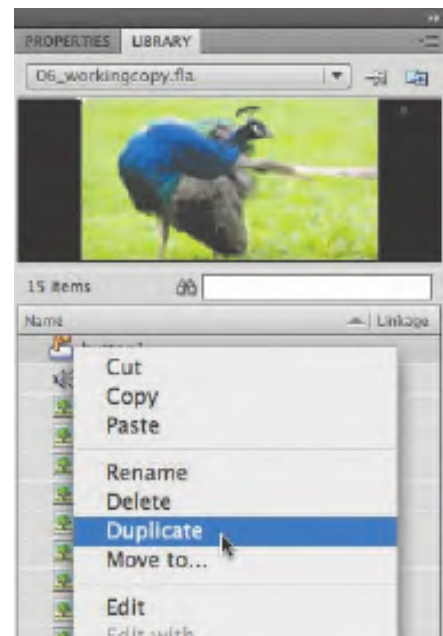
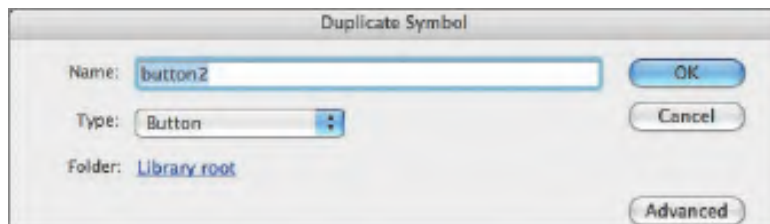


Users can't see invisible buttons, but because the Hit keyframe still defines a clickable area, invisible buttons remain active. Hence, you can place invisible buttons over any part of the Stage and use ActionScript to program them to respond to users. Invisible buttons are useful for creating generic hotspots. For example, placing them on top of different photos can help you make each photo respond to a mouse click without having to make each photo a different button symbol.

Duplicating Buttons

Now that you've created one button, the others will be easier to create. You'll duplicate the button and then modify each one to display a different thumbnail image.

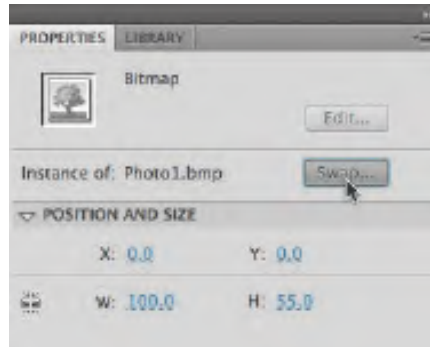
- 1 In the Library panel, right-click/Ctrl-click the `button1` symbol. Select Duplicate from the context menu. You can also click the options menu at the top-right corner of the Library and select Duplicate.
- 2 In the Duplicate Symbol dialog box, select Button, and name it **button2**. Click OK.



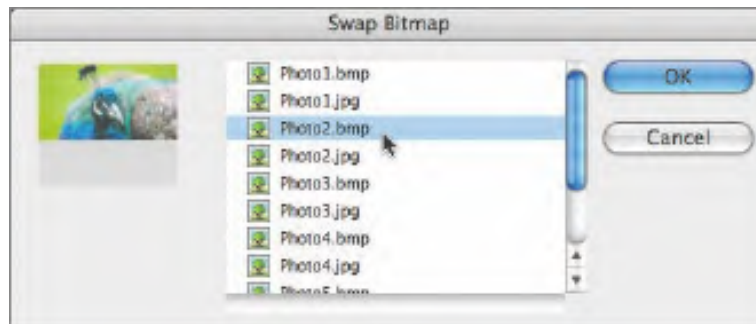
Swapping Bitmaps

Bitmaps and symbols are easy to swap on the Stage and can significantly speed up your workflow.

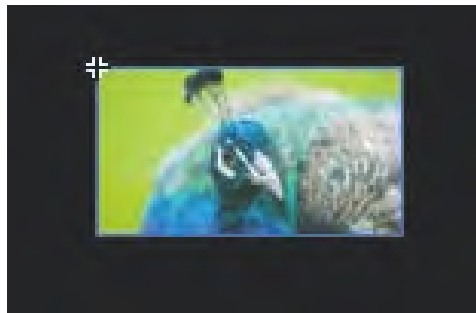
- 1 In the Library panel, double-click the icon for the button2 symbol to edit it.
- 2 Select the thumbnail image of the peacock.
- 3 In the Property inspector, click Swap.



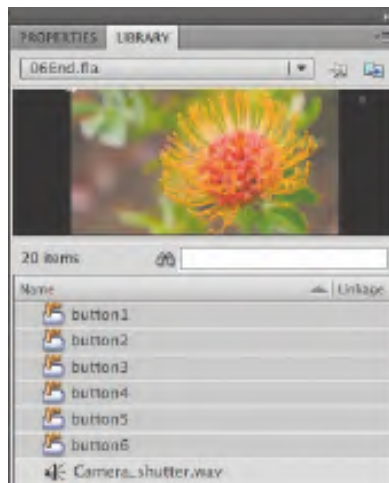
- 4 In the Swap Bitmap dialog box, select the next thumbnail image, called Photo2.bmp, and click OK.



The original thumbnail is swapped for the one you selected. Because they are both the same size, the replacement is seamless.



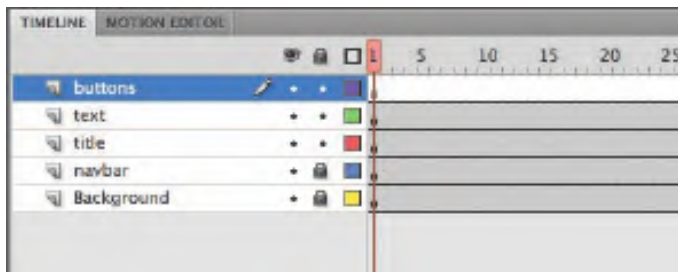
- 5 Click Scene 1 to return to the main Timeline.
- 6 Continue duplicating your buttons and swapping the bitmaps until you have six different button symbols in your Library.



Placing the Button Instances

The buttons need to be put on the Stage and given names in the Property inspector so that ActionScript can refer to them.

- 1 Insert a new layer and rename it **buttons**.



- 2 Drag each of your buttons from the Library to the Stage, placing them in a horizontal row. Don't worry about their exact position because you'll align them nicely in the next few steps.



- 3 Select the first button, and in the Property inspector, set the X value to 35.
- 4 Select the last button, and in the Property inspector, set the X value to 664.

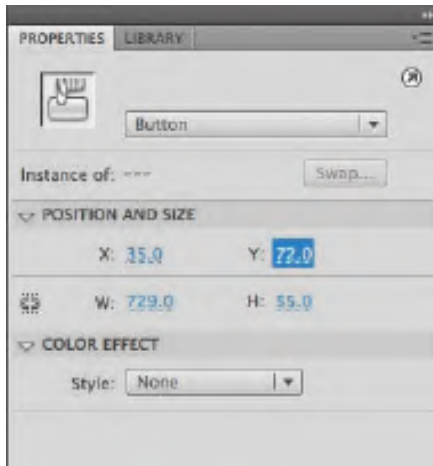


- 5 Select all six buttons. In the Align panel (Window > Align), deselect the To Stage option, click the Space Evenly Horizontally button, and then click the Align Top Edge button.



All six buttons are now evenly distributed.

- 6 With all the buttons still selected, in the Property inspector, enter 72 for the Y value.



All six buttons are aligned horizontally.

- 7 You can now test your movie to see how the buttons behave. Choose Control > Test Movie. Roll over and click each button to see how the Over and Down keyframes of each button appear. At this point, however, you haven't provided any instructions for the buttons to actually do anything. That part comes after you name the buttons and learn a little about ActionScript.



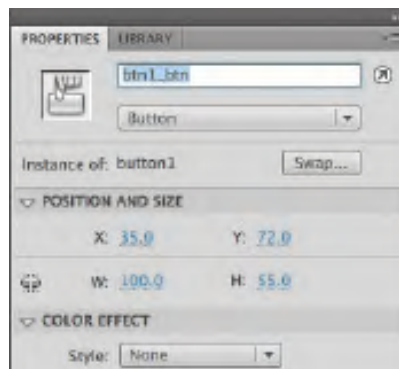
Naming the Button Instances

Name each button instance so that it can be referenced by ActionScript. This is a crucial step that many beginners forget to do.

- 1 Select the first button on the Stage.



- 2 Type **btn1_btn** in the Instance Name field in the Property inspector.



- 3 Name each of the other buttons **btn2_btn**, **btn3_btn**, **btn4_btn**, **btn5_btn**, and **btn6_btn**.
- 4 Lock all the layers.

Naming Rules

Naming instances is a critical step in creating interactive Flash projects. The most common mistake made by novices is not to name, or to incorrectly name, a button instance.

The instance names are important because ActionScript uses the names to reference those objects. Instance names are not the same as the symbol names in the Library. The names in the Library are simply organizational reminders.

Instance naming follows these simple rules:

- 1 Do not use spaces or special punctuation. Underscores are okay to use.
- 2 Do not begin a name with a number.
- 3 End your button name with **_btn**. Although it is not required, it helps identify those objects as buttons.
- 4 Do not use any word that is reserved for a Flash ActionScript command.

Understanding ActionScript 3.0

Adobe Flash CS4 uses ActionScript 3.0, a robust scripting language, to extend the functionality of Flash. Although ActionScript 3.0 may seem intimidating to you if you're new to scripting, you can get great results with some very simple scripts. As with any language, it's best if you take the time to learn the syntax and some basic terminology.

About ActionScript

ActionScript, which is similar to JavaScript, lets you add more interactivity to Flash animations. In this lesson, you'll use ActionScript to attach behaviors to buttons. You'll also learn how to use ActionScript for such simple tasks as stopping an animation.

You don't have to be a scripting expert to use ActionScript. In fact, for common tasks, you may be able to copy script that other Flash users have shared. However, you'll be able to accomplish much more in Flash—and feel more confident using the application—if you understand how ActionScript works.

This lesson isn't designed to make you an ActionScript expert. Instead, it introduces common terms and syntax, walks you through a simple script, and provides an introduction to the ActionScript language.

If you've used scripting languages before, the documentation included in the Flash Help menu may provide all the additional guidance you need to use ActionScript proficiently. If you're new to scripting and want to learn ActionScript, you may find an ActionScript 3.0 book for beginners helpful.

Understanding Scripting Terminology

Many of the terms used in describing ActionScript are similar to terms used for other scripting languages. The following terms are used frequently in ActionScript documentation.

Variable

A *variable* represents a specific piece of data that may or may not be constant. When you create, or *declare*, a variable, you also assign a data type, which determines what kind of data the variable can represent. For example, a String variable holds any string of alphanumeric characters, whereas a Number variable must contain a number.

● **Note:** Variable names must be unique, and they are case sensitive. The variable `mypassword` is not the same as the variable `MyPassword`. Variable names can contain only numbers, letters, and underscores; they cannot begin with a number. These are the same naming rules for instances. (In fact, they are conceptually the same.)

Keyword

In ActionScript, a *keyword* is a reserved word that is used to perform a specific task. For example, *var* is a keyword that is used to create a variable.

You can find a complete list of keywords in Flash Help. Because these words are reserved, you can't use them as variable names or in other ways. ActionScript always uses them to perform their assigned tasks.

Parameters

Parameters provide specific details for a particular command and are the values between parentheses () in a line of code. For example, in the code `gotoAndPlay(3)`; the parameter instructs the script to go to frame 3.

Function

A *function* is a group of statements that you can refer to by name. Using a function makes it possible to run the same set of statements without having to type them repeatedly.

Objects

In ActionScript 3.0, you work with objects, which are abstract types of data that help you do certain tasks. A Sound object, for example, helps you control sound, and a Date object can help you manipulate time-related data. The button symbols that you created earlier in this lesson are also objects—they are Button objects.

Every object should be named. An object that has a name can be referenced and controlled with ActionScript. Buttons on the Stage are referred to as instances, and in fact, *instances* and *objects* are synonymous.

Methods

Methods are the keywords that result in action. Methods are the doers of ActionScript, and each kind of object has its own set of methods. Much of learning ActionScript is learning the methods for each kind of object. For example, two methods associated with a MovieClip object are `stop()` and `gotoAndPlay()`.

Properties



Properties describe an object. For example, the properties of a movie clip include its height and width, *x* and *y* coordinates, and scale. Many properties can be changed, whereas other properties can only be “read,” meaning they simply describe the object.

Using Proper Scripting Syntax


If you're unfamiliar with program code or scripting, ActionScript code may be challenging to decipher. Once you understand the basic *syntax*, which is the grammar and punctuation of the language, you'll find it easier to follow a script.

- The **semicolon** at the end of the line tells ActionScript that it has reached the end of the code line and to go to the next line in the code.
- As in English, every open **parenthesis** must have a corresponding close parenthesis, and the same is true for **brackets** and **curly brackets**. If you open something, you must close it. Very often, the curly brackets in ActionScript code will be separated on different lines. This makes it easier to read what's inside the curly brackets.
- The **dot** operator (.) provides a way to access the properties and methods of an object. Type the instance name, followed by a dot, and then the name of the property or method. Think about the dot as a way to separate objects, methods, and properties.
- Whenever you're entering a string or the name of a file, use **quotation marks**.
- You can add **comments** that ActionScript will ignore to remind you or others what you are accomplishing with different parts of the script. To add a comment for a single line, start it with two slashes (//). To type a multiline comment, start it with /* and end it with */.

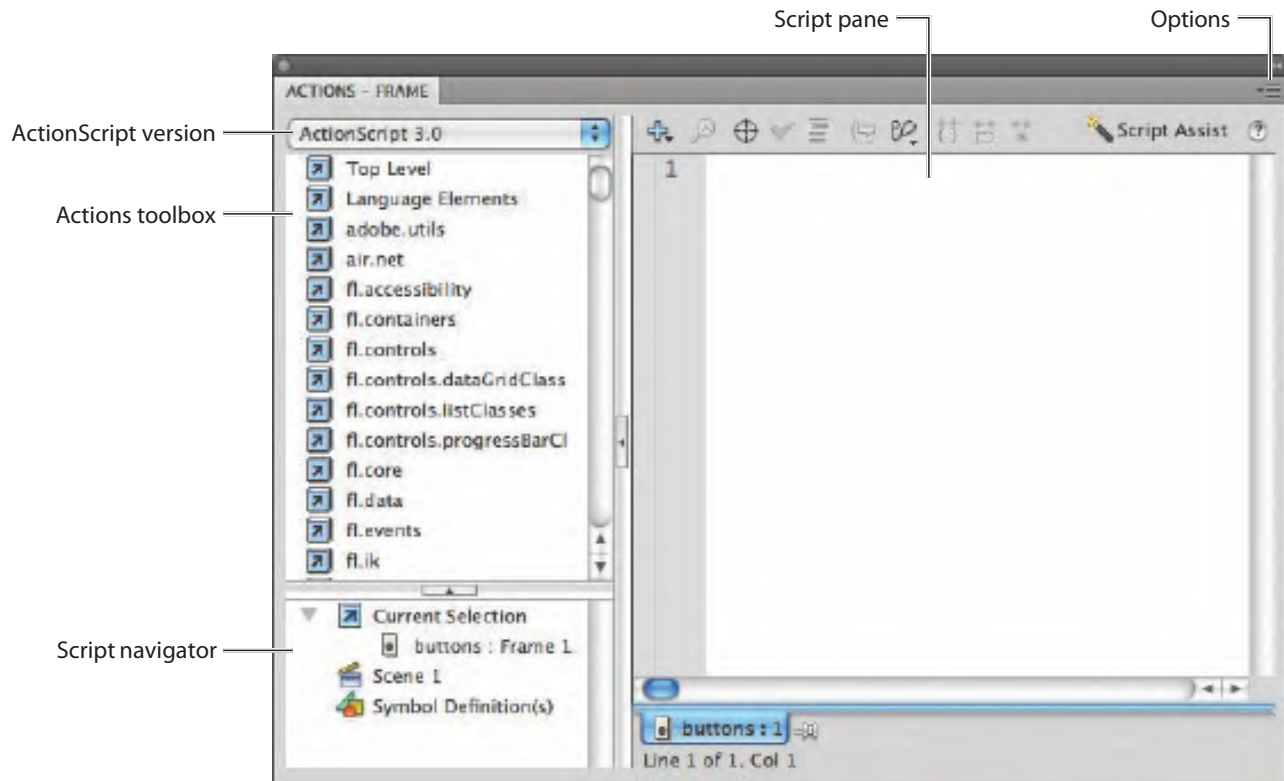
Flash provides assistance in the following ways as you write scripts in the Actions panel:

- Words that have specific meanings in ActionScript, such as keywords and statements, appear in blue as you type them in the Actions panel. Words that are not reserved in ActionScript, such as variable names, are in black. Strings are in green. Comments, which ActionScript ignores, are in gray.
- As you work in the Actions panel, Flash can detect the action you are entering and display a code hint. There are two types of code hints: a tooltip that contains the complete syntax for that action and a pop-up menu that lists possible ActionScript elements.
- To check the syntax of a script you're writing, click the AutoFormat icon  (which will also format the script according to conventions so that it is easier for others to read) or click the Check Syntax icon . Syntax errors are listed in the Compiler Errors panel.

Navigating the Actions Panel

The Actions panel is where you write all your code. Open the Actions panel by choosing Window > Actions or select a keyframe on the Timeline and click the Actions panel icon  on the top right of the Property inspector. You can also right-click/Ctrl-click on any keyframe and select Actions from the context menu.

The Actions panel gives you quick access to the core elements of ActionScript as well as provides you with different options to help you write, debug, format, edit, and find your code.



The Actions panel is divided into several panes. At the top-left corner is the Actions toolbox. Several categories are listed in the Actions toolbox, which organizes all the ActionScript code. At the top of the Actions toolbox is a pull-down menu that displays only the code for the version of ActionScript you select. You should select ActionScript 3.0, the latest version. At the very bottom of the Actions toolbox categories is a yellow Index category that lists, in alphabetical order, all the language elements. You don't need to use the toolbox to add code to your script, but it can help to ensure that you're using the code correctly.

At the top right of the Actions panel is the Script pane—the blank slate in which all your code appears. You enter ActionScript in the Script pane just as you would in a text-editing application.

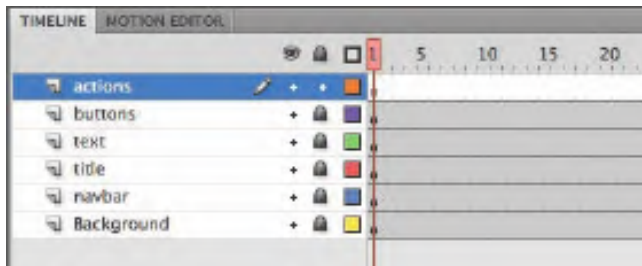
At the bottom left of the Actions panel is the Script navigator, which can help you find a particular piece of code. ActionScript is placed on keyframes on the Timeline, so the Script navigator can be particularly useful if you have lots of code scattered in different keyframes and on different Timelines.

All the panes in the Actions panel can be resized to suit your working style. They can even be collapsed completely to maximize the pane that you are working in. To resize a pane, click and drag the horizontal or vertical dividers.

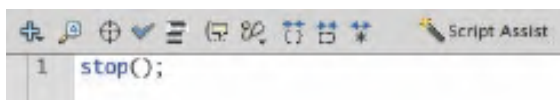
Adding a Stop Action

You've used stop actions in previous lessons. A stop action simply stops the movie from continuing by halting the playhead; in this case, you'll use it to stop the movie from showing any of the content along the Timeline until the user actually clicks a button.

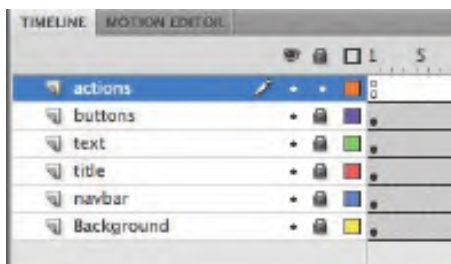
- 1 Insert a new layer and rename it **actions**.



- 2 Select the first keyframe of the actions layer and open the Actions panel (Window > Actions).
- 3 In the Script pane, enter `stop()` ;



The code appears in the Script pane and a tiny lowercase “a” appears in the first keyframe of the actions layer to indicate it contains some ActionScript. The movie will now stop at frame 1.

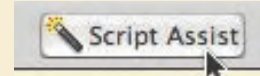


About Script Assist

Some actions are simple, such as stop actions. But others require parameters, and if you're unfamiliar with ActionScript, you may have difficulty remembering which options are available—or required—for each type of action. Script Assist mode prompts you to add the methods, parameters, or variables related to the action, and then it assembles the information using the correct syntax for you.

Script Assist won't write the script for you. You still have to know what you want to do and which variables, methods, or functions to use. However, it can help you put the pieces together coherently, so that you don't have to return to the script multiple times to troubleshoot syntax errors.

To use Script Assist, click Script Assist in the Actions panel.



Then double-click an item in the Actions toolbox to add it to the Script pane. The upper section of the Actions panel displays fields and options that are available for that item. Select options and enter values as appropriate for your script.

Creating Event Handlers for Buttons

Events are occurrences that happen in the Flash environment that Flash can detect and respond to. For example, a mouse click, a mouse movement, and a key press on the keyboard are all events. These events are produced by the user, but some events can happen independently of the user, like the successful loading of a piece of data or the completion of a sound. With ActionScript, you can write code that detects events and respond to them with an event handler.

The first step in event handling is to create a listener that will detect the event. A listener looks something like this:

```
wheretolisten.addEventListener(whatevent, responsetoevent);
```

The actual command is `addEventListener()`. The other words are placeholders for objects and parameters for your situation. *Wheretolisten* is the object where the event occurs (usually a button), *whatevent* is the specific kind of event (such as a mouse click), and *responsetoevent* is a function that is triggered when the event happens. So if you want to listen for a mouse click over a button called `btn1_btn`, and the response is to trigger a function called `showimage1`, the code would look like this:

```
btn1_btn.addEventListener(MouseEvent.CLICK, showimage1);
```

The next step is to create the function that will respond to the event—in this case, the function called `showimage1`. A function simply groups a bunch of actions

together; you can then trigger that function by referencing its name. A function looks something like this:

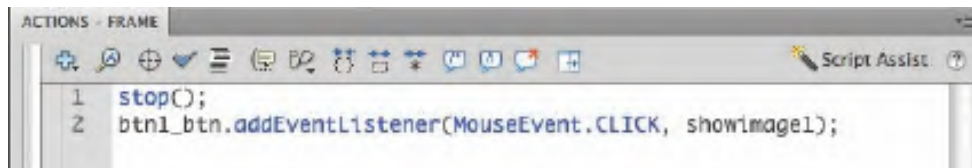
```
function showimage1 (myEvent:MouseEvent){ };
```

Function names, like button names, are arbitrary. You can name functions whatever makes sense to you. In this particular example, the name of the function is called `showimage1`. It receives one parameter (within the parentheses) called `myEvent`, which is an event that involves a mouse event. The colon describes what type of object it is. If this function is triggered, all the actions between the curly brackets are executed.

Adding the Event Listener and Function

You'll add ActionScript code to listen for a mouse click on each button. The response will make Flash go to a particular frame on the Timeline to show different content.

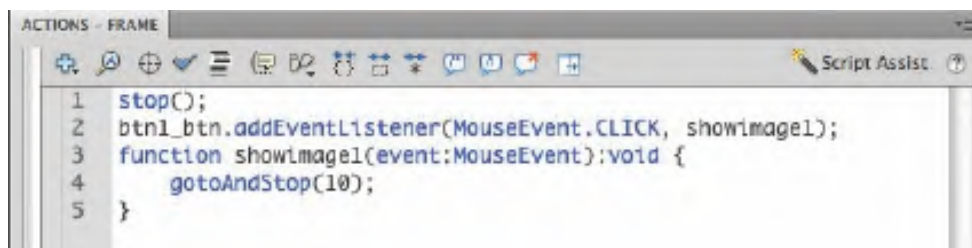
- 1 Select the first frame of the actions layer.
- 2 Open the Actions panel.
- 3 In the Script pane of the Actions panel, beginning on the second line, type `btn1_btn.addEventListener(MouseEvent.CLICK, showimage1);`



The listener listens for a mouse click over the `btn1_btn` object on the Stage. If the event happens, the function called `showimage1` is triggered.

- 4 On the next line of the Script pane, type

```
function showimage1(event:MouseEvent):void {  
    gotoAndStop(10);  
}
```



● **Note:** Be sure to include the final curly bracket, or the function won't work.

The function called `showimage1` contains instructions to go to frame 10 and stop there. The code for your button called `btn1_btn` is complete.

- 5 On the next line of the Script pane, enter additional code for the remaining five buttons. You can copy and paste lines 2 and 3, and simply change the names of the button, the name of the function (in two places), and the destination frame. The full script should be as follows:

```
stop();
btn1_btn.addEventListener(MouseEvent.CLICK, showimage1);
function showimage1(event:MouseEvent):void {
    gotoAndStop(10);
}
btn2_btn.addEventListener(MouseEvent.CLICK, showimage2);
function showimage2(event:MouseEvent):void {
    gotoAndStop(20);
}
btn3_btn.addEventListener(MouseEvent.CLICK, showimage3);
function showimage3(event:MouseEvent):void {
    gotoAndStop(30);
}
btn4_btn.addEventListener(MouseEvent.CLICK, showimage4);
function showimage4(event:MouseEvent):void {
    gotoAndStop(40);
}
btn5_btn.addEventListener(MouseEvent.CLICK, showimage5);
function showimage5(event:MouseEvent):void {
    gotoAndStop(50);
}
btn6_btn.addEventListener(MouseEvent.CLICK, showimage6);
function showimage6(event:MouseEvent):void {
    gotoAndStop(60);
}
```

Mouse Events

The following list contains the ActionScript codes for common mouse events. Use these codes when you create your listener, and make sure that you pay attention to lowercase and uppercase letters. For most users, the first event (MouseEvent.CLICK) will be sufficient for all projects. That event happens when the user clicks the mouse button.

- MouseEvent.CLICK
- MouseEvent.DOUBLE_CLICK
- MouseEvent.MOUSE_MOVE
- MouseEvent.MOUSE_DOWN
- MouseEvent.MOUSE_UP
- MouseEvent.MOUSE_OVER
- MouseEvent.MOUSE_OUT

ActionScript Commands for Navigation

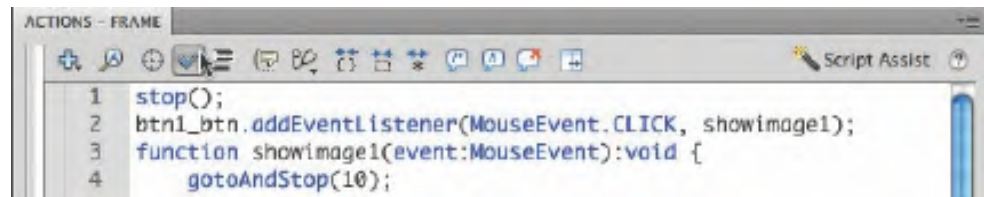
The following list contains the ActionScript codes for common navigation commands. Use these codes when you create buttons to stop the playhead, start the playhead, or move the playhead to different frames. The gotoAndStop and gotoAndPlay commands require additional information, or parameters, within their parentheses as indicated.

- stop();
- play();
- gotoAndStop(framenumber or framelabel);
- gotoAndPlay(framenumber or framelabel);
- nextFrame();
- prevFrame();

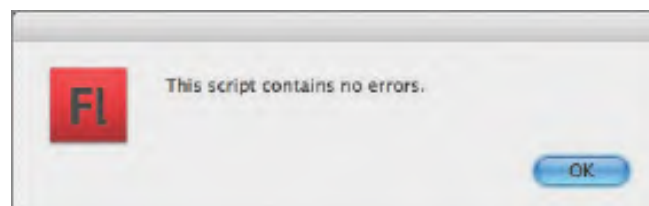
Checking Syntax and Formatting Code

ActionScript can be very picky, and a single misplaced period can cause your entire project to grind to a halt. Fortunately, the Actions panel provides a few tools to help you identify errors and help you fix them.

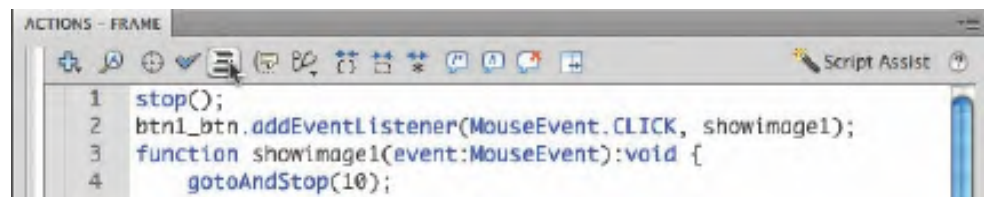
- 1 Select the first frame of your actions layer and open the Actions panel if it is not already open.
- 2 Click the Check Syntax button at the top of the Actions panel.



Flash checks the syntax of your ActionScript code. Flash notifies you if there are errors or if your code is error free.



- 3 Click the AutoFormat icon at the top of the Actions panel.



Flash formats your code so it conforms to standard spacing and line breaks.

● **Note:** Change the automatic formatting by selecting Preferences from the upper right options menu. Choose Auto Format from the left menu and select the various options for formatting your code.

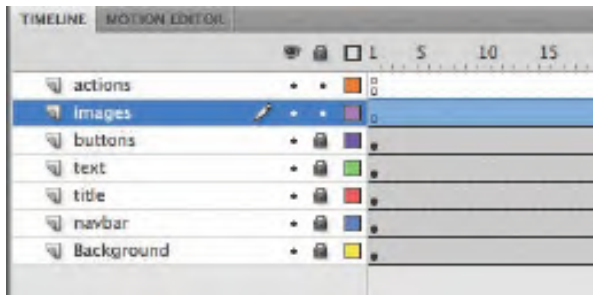
Creating Destination Keyframes

When the user clicks each button, Flash moves the playhead to a new spot on the Timeline. However, you haven't yet placed anything different at those particular frames. That's the next step.

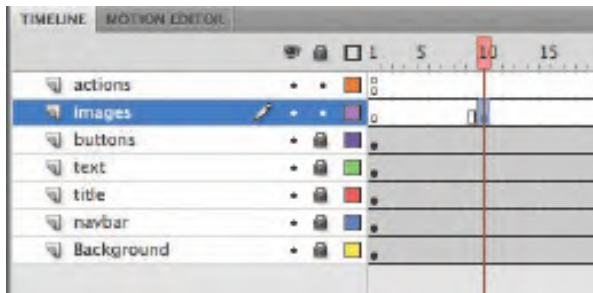
Inserting Keyframes with Different Content

You will create six keyframes in a new layer and place a different large photo in each of the keyframes.

- 1 Insert a new layer at the top of the layer stack but below the actions layer and rename it **images**.

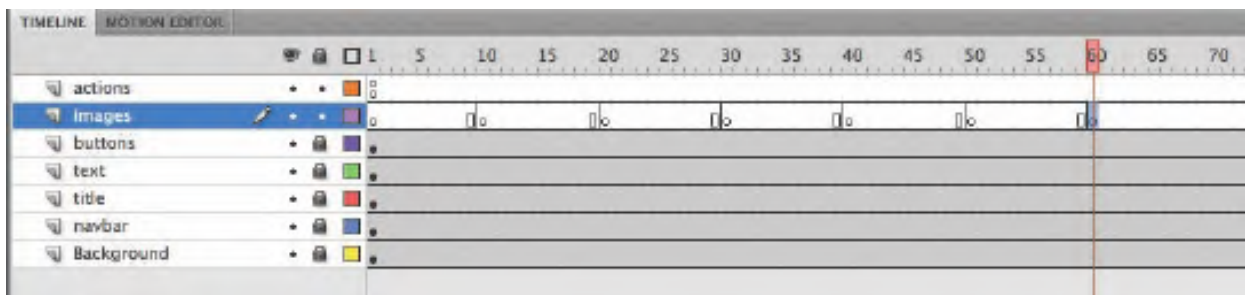


- 2 Select frame 10 of the images layer.
- 3 Insert a new keyframe at frame 10 (Insert > Timeline > Keyframe, or F6).



- 4 Insert new keyframes at frames 20, 30, 40, 50, and 60.

Your Timeline has seven empty keyframes in the images layer.



- 5 Select the keyframe at frame 10.
- 6 Drag the bitmap image called Photo1.jpg from the Library to the Stage. Photo1.jpg is a large photo of a peacock.
- 7 In the Property inspector, set the X value to **35** and the Y value to **138**.
The keyframe at frame 10 contains a large photo of a peacock.

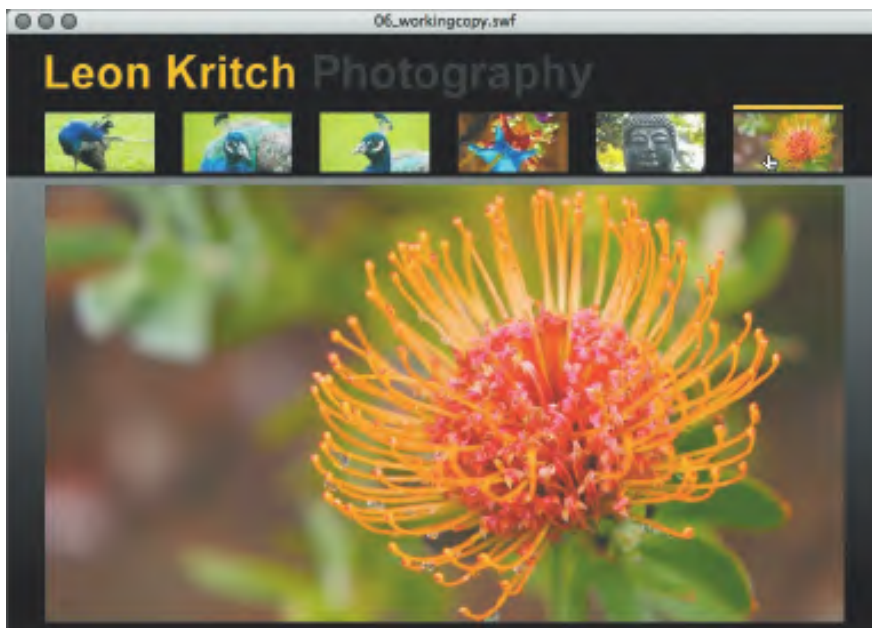


- 8 Select the keyframe at frame 20.
- 9 Drag the bitmap image called Photo2.jpg from the Library to the Stage.
- 10 In the Property inspector, set the X value to **35** and the Y value to **138**.
The keyframe at frame 20 contains a large photo of the head of a peacock (see figure at top of next page).
- 11 Place each of the large bitmaps from the Library in the corresponding keyframes in the images layer.
Each keyframe should contain a different photo from this photographer's portfolio.



12 Choose Control > Test Movie and click any of the buttons.

Each button on the top row moves the playhead to a different frame on the Timeline, where a different photo is displayed. The user can choose to see any photo in any order.

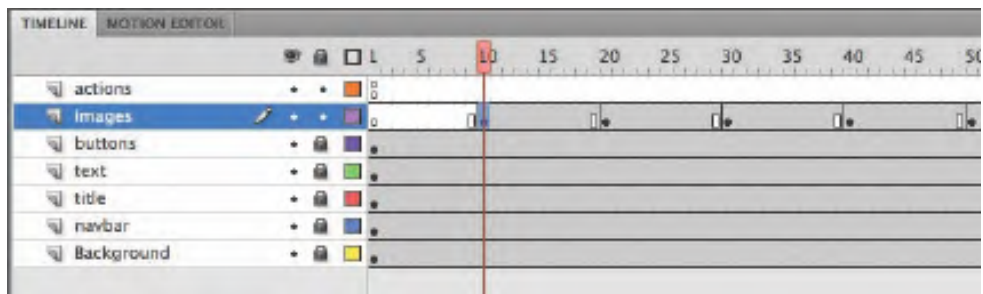


Using Labels on Keyframes

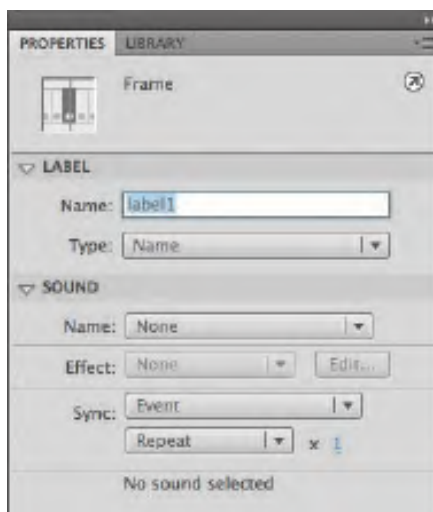
Your ActionScript code tells Flash to go to a different frame number when the user clicks each of the buttons. However, if you decide to edit your Timeline and add or delete a few frames, you'll need to go back into your ActionScript and change your code so the frame numbers match.

An easy way to avoid this problem is to use frame labels instead of fixed frame numbers. Frame labels are names that you give to keyframes. Instead of referring to keyframes by their frame number, you refer to them by their label. So, even if you move your destination keyframes as you edit, the labels remain with their keyframes. To reference frame labels in ActionScript, you must enclose them in quotation marks. The command `gotoAndStop("label1")` makes the playhead go to the keyframe with the label called label1.

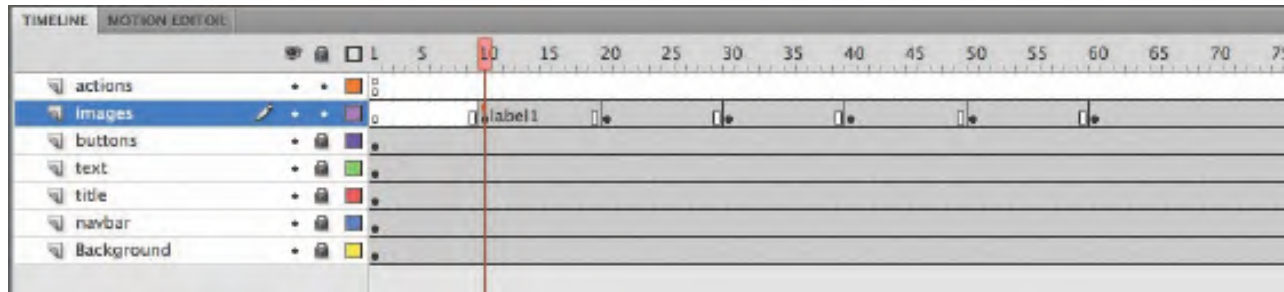
- 1 Select frame 10 on the images layer.



- 2 In the Property inspector, enter **label1** in the Label Name field.

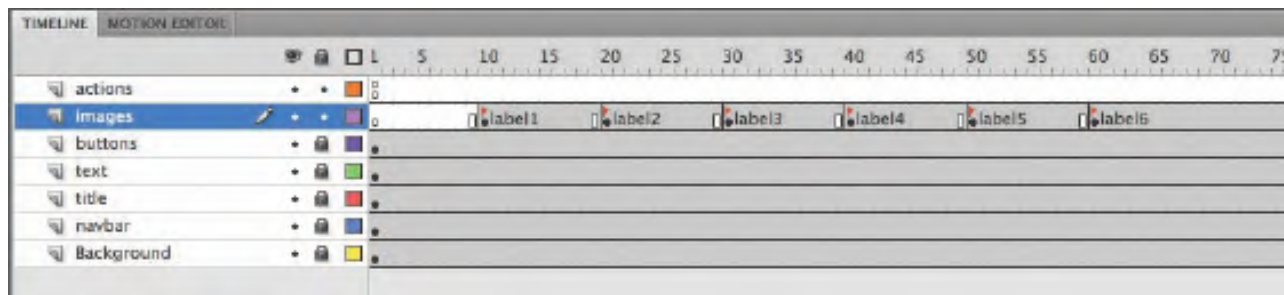


A tiny flag icon appears on each of the keyframes that have labels.



- 3 Select frame 20 on the images layer.
- 4 In the Property inspector, enter **label2** in the Label Name field.
- 5 Select frames 30, 40, 50, and 60, and in the Property inspector, enter corresponding names in the Label Name field: **label3**, **label4**, **label5**, and **label6**.

A tiny flag icon appears on each of the keyframes that have labels.



- 6 Select the first frame of the actions layer and open the Actions panel.
- 7 In your ActionScript code, change all the fixed frame numbers in each of the `gotoAndStop()` commands to the corresponding frame labels:
 - `gotoAndStop(10)`; should be changed to `gotoAndStop("label1")`;
 - `gotoAndStop(20)`; should be changed to `gotoAndStop("label2")`;
 - `gotoAndStop(30)`; should be changed to `gotoAndStop("label3")`;
 - `gotoAndStop(40)`; should be changed to `gotoAndStop("label4")`;
 - `gotoAndStop(50)`; should be changed to `gotoAndStop("label5")`;
 - `gotoAndStop(60)`; should be changed to `gotoAndStop("label6")`;

```
1 stop();
2 btn1_btn.addEventListener(MouseEvent.CLICK, showimage1);
3 function showimage1(event:MouseEvent):void {
4     gotoAndStop("label1");
5 }
6 btn2_btn.addEventListener(MouseEvent.CLICK, showimage2);
7 function showimage2(event:MouseEvent):void {
8     gotoAndStop("label2");
9 }
10 btn3_btn.addEventListener(MouseEvent.CLICK, showimage3);
11 function showimage3(event:MouseEvent):void {
12     gotoAndStop("label3");
13 }
14 btn4_btn.addEventListener(MouseEvent.CLICK, showimage4);
15 function showimage4(event:MouseEvent):void {
16     gotoAndStop("label4");
17 }
18 btn5_btn.addEventListener(MouseEvent.CLICK, showimage5);
19 function showimage5(event:MouseEvent):void {
20     gotoAndStop("label5");
21 }
22 btn6_btn.addEventListener(MouseEvent.CLICK, showimage6);
23 function showimage6(event:MouseEvent):void {
24     gotoAndStop("label6");
25 }
```

The ActionScript code now directs the playhead to a particular frame label instead of a particular frame number.

- 8 Test your movie by choosing Control > Test Movie.

The functionality of your interactive movie remains the same, but future edits to the Timeline will be easier to make.

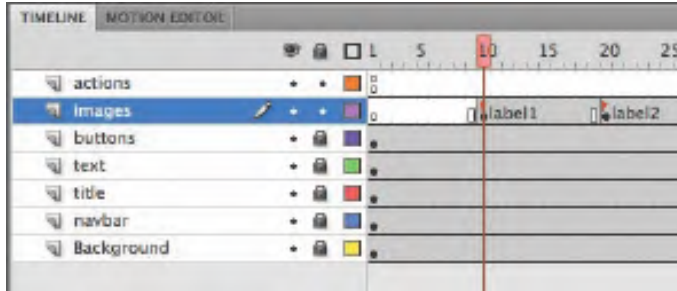
Playing Animation at the Destination

So far, this interactive portfolio works by using the `gotoAndStop()` command to show images in different keyframes along the Timeline. But how would you play an animation after a user clicks a button? The answer is to use the command `gotoAndPlay()`, which moves the playhead to the frame number or frame label specified by its parameter and plays from that point.

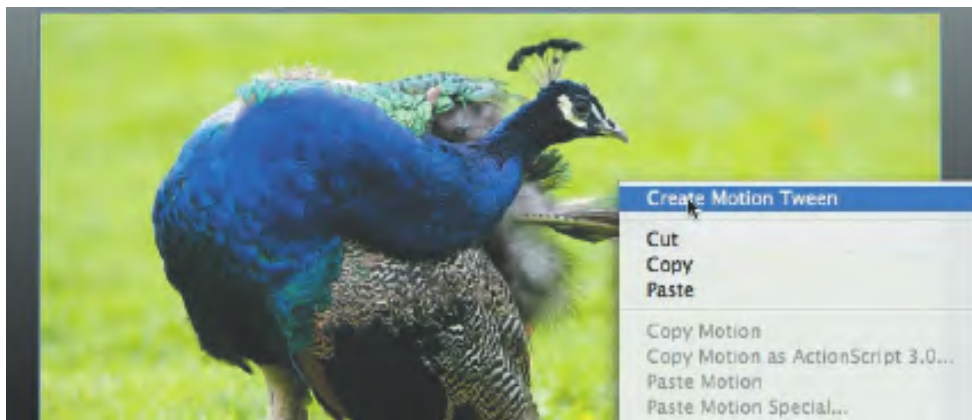
Creating Transition Animations

Next, you will create a short transition animation for each of your photos. Then you'll change your ActionScript code to direct Flash to go to each of the keyframes and play the animation.

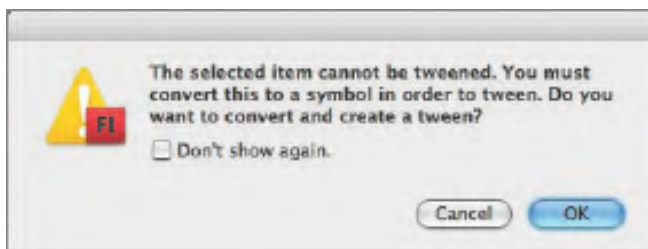
- 1 Move the playhead to the label1 frame label.



- 2 Right-click/Ctrl-click on the photo on the Stage and choose Create Motion Tween.

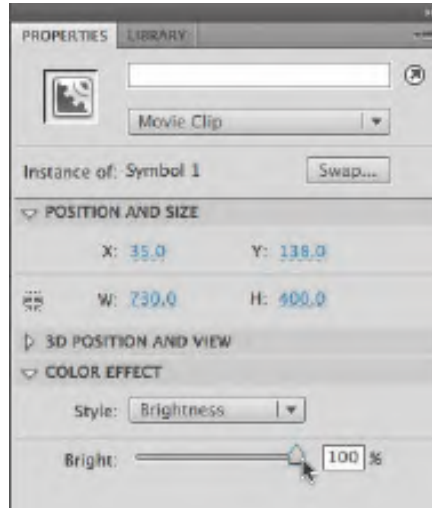


Flash asks to convert the photo to a symbol so that it can proceed with the motion tween. Click OK.



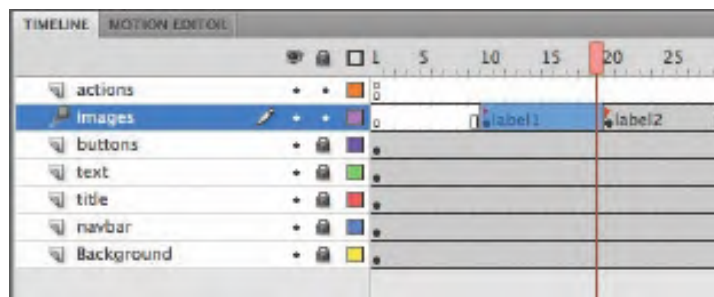
- 3 In the Property inspector, select Brightness from the Style pull-down menu in the Color Effect section.

- 4 Set the Bright slider to 100%.

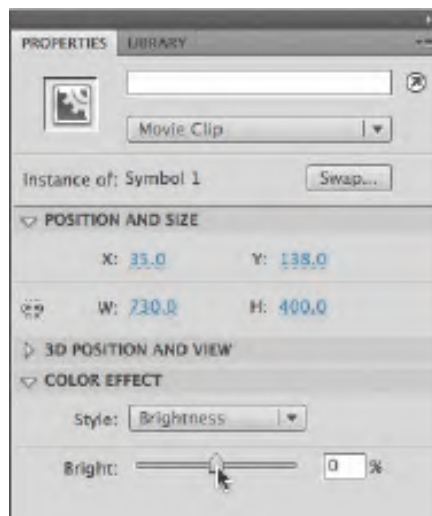


The instance on the Stage becomes bright white.

- 5 Move the playhead to the end of the tween span at frame 19.



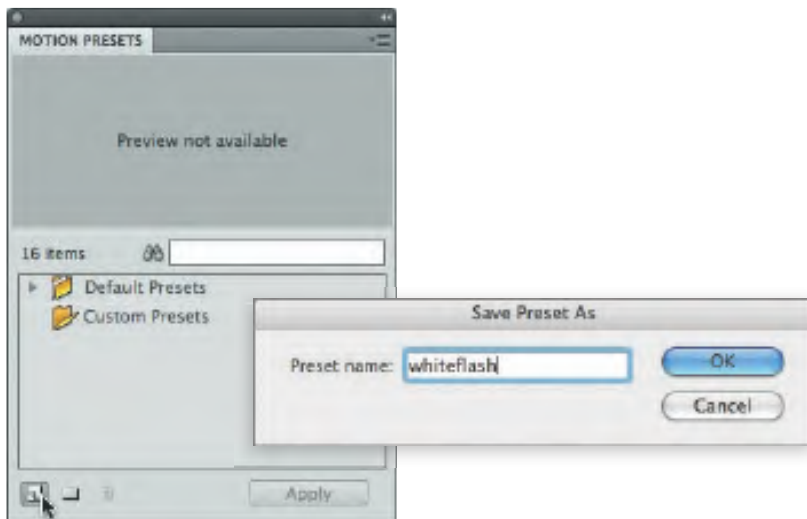
- 6 Select the bright white instance on the Stage.
7 In the Property inspector, set the Bright slider to 0%.



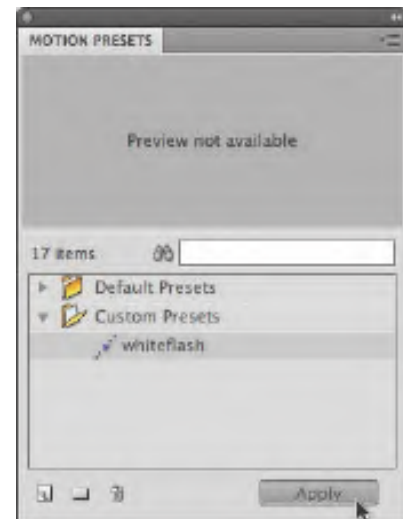
The instance is displayed at a normal brightness level. The motion tween from frame 10 to frame 19 shows a smooth transition from a bright white photo to a normal photo.

- 8 Create similar motion tweens for the remaining photos in the keyframes labeled label2, label3, label4, label5, and label6.

Note: Recall that you can use the Motion Presets panel to save a motion tween and apply it to other objects to save you time and effort. Select the first motion tween on the Timeline and click Save selection as preset.



Provide a Preset name and click OK.



Click on the second photo, select the preset that you just saved, and click Apply.

Using the gotoAndPlay Command

The gotoAndPlay command makes the Flash playhead move to a specific frame on the Timeline and begin playing from that point.

- 1 Select the first frame of the actions layer and open the Actions panel.
- 2 In your ActionScript code, change all the gotoAndStop() commands to gotoAndPlay() commands. Leave the parameter unchanged:
 - gotoAndStop("label1"); should be changed to gotoAndPlay("label1");
 - gotoAndStop("label2"); should be changed to gotoAndPlay("label2");
 - gotoAndStop("label3"); should be changed to gotoAndPlay("label3");
 - gotoAndStop("label4"); should be changed to gotoAndPlay("label4");
 - gotoAndStop("label5"); should be changed to gotoAndPlay("label5");
 - gotoAndStop("label6"); should be changed to gotoAndPlay("label6");

```

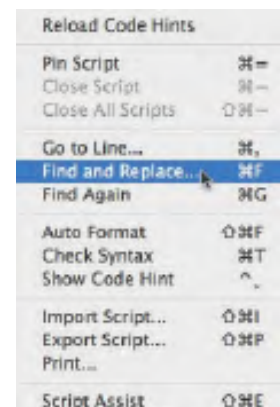
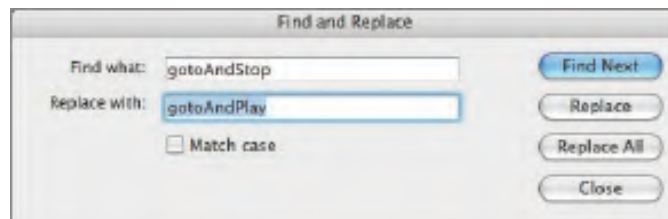
1 stop();
2 btn1_btn.addEventListener(MouseEvent.CLICK, showImage1);
3 function showImage1(event:MouseEvent):void {
4     gotoAndPlay("label1");
5 }
6 btn2_btn.addEventListener(MouseEvent.CLICK, showImage2);
7 function showImage2(event:MouseEvent):void {
8     gotoAndPlay("label2");
9 }
10 btn3_btn.addEventListener(MouseEvent.CLICK, showImage3);
11 function showImage3(event:MouseEvent):void {
12     gotoAndPlay("label3");
13 }
14 btn4_btn.addEventListener(MouseEvent.CLICK, showImage4);
15 function showImage4(event:MouseEvent):void {
16     gotoAndPlay("label4");
17 }
18 btn5_btn.addEventListener(MouseEvent.CLICK, showImage5);
19 function showImage5(event:MouseEvent):void {
20     gotoAndPlay("label5");
21 }
22 btn6_btn.addEventListener(MouseEvent.CLICK, showImage6);
23 function showImage6(event:MouseEvent):void {
24     gotoAndPlay("label6");
25 }

```

The ActionScript code now directs the playhead to a particular frame label and begins playing at that point.

● **Note:** A fast and easy way of doing multiple replacements is to use the Find and Replace command in the Actions panel. From the options menu in the upper-right corner, select Find and Replace.

In the Find field, enter **gotoAndStop**, and in the Replace field, enter **gotoAndPlay**.

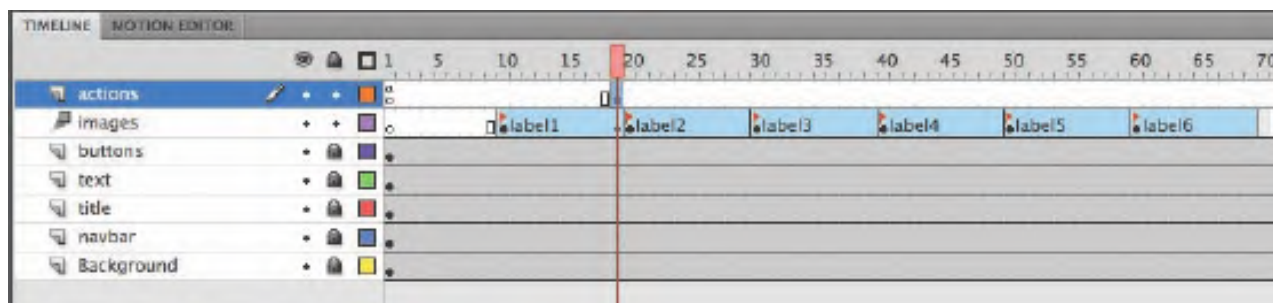


Click Replace All and Flash will make the substitutions in the code.

Stopping the Animations

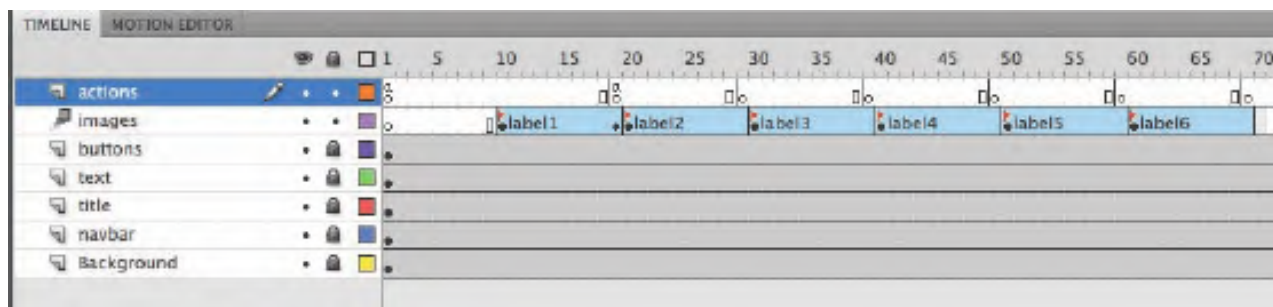
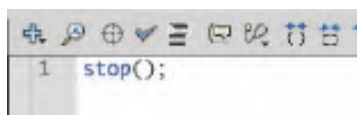
If you test your movie now (Control > Test Movie), you'll see that each button goes to its corresponding frame label and plays from that point, but it keeps playing, showing all the remaining animations in the Timeline. The next step is to tell Flash when to stop.

- 1 Select frame 19 of the actions layer, the frame just before the label2 keyframe on the image layer.
- 2 Right-click/Ctrl-click and choose Insert Keyframe.
A new keyframe is inserted in frame 19 of the actions layer.



- 3 Open the Actions panel.
The Script pane in the Actions panel is blank. Don't panic! Your code has not disappeared. Your code for the event listeners is on the first keyframe of the actions layer. You have selected a new keyframe in which you will add a stop command.

- 4 In the Script pane, enter `stop();`
Flash will stop playing when it reaches frame 19.
- 5 Insert keyframes at frames 29, 39, 49, 59, and 69.



- 6 In each of those keyframes, add a stop command in the Actions panel.
- 7 Test your movie by choosing Control > Test Movie.
Each button takes you to a different keyframe and plays a short animation. At the end of the animation, the movie stops and waits for you to click another button.

● **Note:** If you want a quick and easy way to duplicate the keyframe containing the stop command, hold down the Alt/Option key while you move it to a new location on the Timeline.

Animated Buttons

Animated buttons display an animation in the Up, Over, or Down keyframes. When you hover your mouse cursor over one of the portfolio buttons, the yellow horizontal bar appears. But imagine if that yellow horizontal bar was animated. It would give more life and whimsy to the interaction between the user and the button.

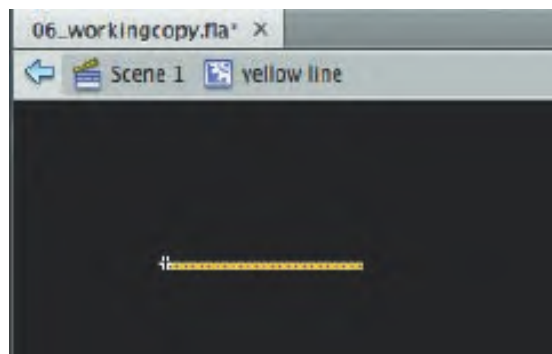
The key to creating an animated button is to create an animation inside a movie clip symbol, and then place that movie clip symbol inside the Up, Over, or Down keyframes of a button symbol. When one of those button keyframes is displayed, the animation in the movie clip plays.

Creating the Animation in a Movie Clip Symbol

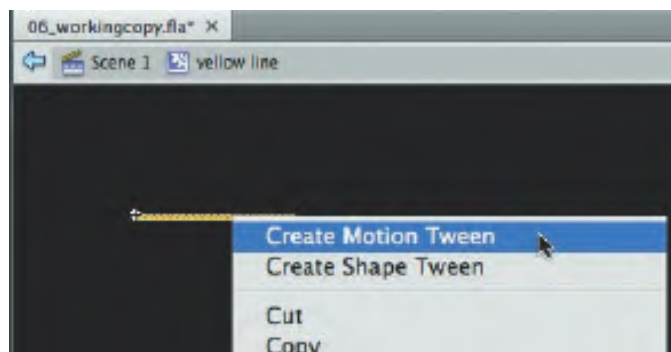
Your button symbols in this portfolio project already contain the yellow line movie clip symbol. You will edit the movie clip symbol to add an animation inside it.

- 1 In the Library, double-click the yellow line movie clip symbol icon.

Flash puts you in symbol-editing mode for the yellow line movie clip symbol.

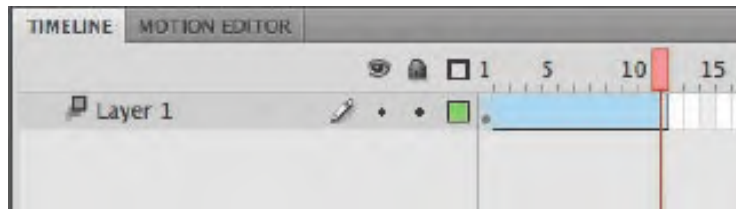


- 2 Right-click/Ctrl-click on the yellow rectangular shape on the Stage and choose Create Motion Tween.

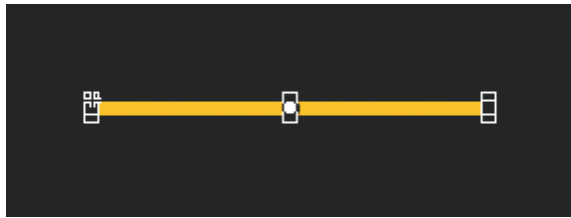


- 3 In the dialog box that appears asking for confirmation to convert the selection to a symbol, click OK.

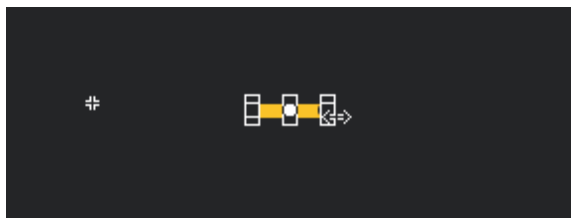
Flash creates a Tween layer and adds one second worth of frames to the movie clip Timeline.



- 4 Select the Free Transform tool.
- 5 Click on the yellow rectangle on the Stage.



- 6 Drag the control handles on the side to reduce the width to about 10 pixels.

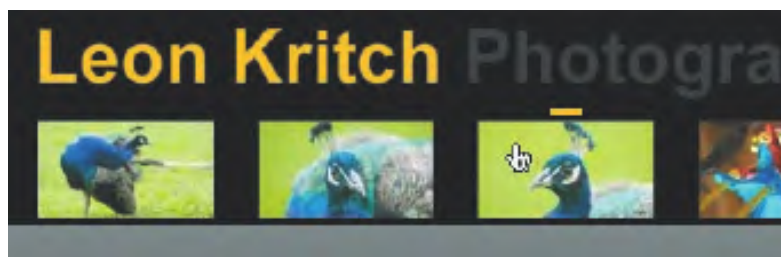


Flash creates a smooth transition between the long rectangle and the short rectangle.

- 7 Exit symbol-editing mode by clicking the Scene 1 button above the Stage.
- 8 Choose Control > Test Movie.

As your mouse cursor hovers over any of the buttons, the yellow rectangle pulses from wide to narrow. The motion tween inside the movie clip symbol plays continuously in a loop, and the movie clip symbol is placed inside the button symbol.

● **Note:** If you want your animated button to play its animation only once, you must add a stop command at the end of the movie clip's Timeline.



Review Questions

- 1 How and where do you add ActionScript code?
- 2 How do you name an instance, and why is it necessary?
- 3 How can you label frames, and when is it useful?
- 4 What is a function?
- 5 What is an event? What is an event listener?
- 6 How do you create an animated button?

Review Answers

- 1 ActionScript code resides in keyframes on the Timeline. Keyframes that contain ActionScript are indicated by a small lowercase “a”. You add ActionScript through the Actions panel. Choose Window > Actions, or select a keyframe and click the Actions icon in the Property inspector, or right-click/Ctrl-click and select Actions. You enter code directly in the Script pane in the Actions panel, or you can select commands from the categories in the Actions toolbox.
- 2 To name an instance, select it on the Stage, and then type in the Instance Name field in the Property inspector. You need to name an instance to reference it in ActionScript.
- 3 To label a frame, select a keyframe on the Timeline, and then type a name in the Frame Label box in the Property inspector. You can label frames in Flash to make it easier to reference frames in ActionScript and to give you more flexibility. If you want to change the destination of a `gotoAndStop` or `gotoAndPlay` command, you can move the label rather than having to locate every reference to the frame number in the script.
- 4 A *function* is a group of statements that you can refer to by name. Using a function makes it possible to run the same set of statements without having to type them repeatedly into the same script. When an event is detected, a function is executed in response.
- 5 An event is an occurrence that is initiated by a button click, a key press, or any number of inputs that Flash can detect and respond to. An event listener, also called an event handler, is a function that is executed in response to specific events.
- 6 Animated buttons display an animation in the Up, Over, or Down keyframes. To create an animated button, make an animation inside a movie clip symbol, and then place that movie clip symbol inside the Up, Over, or Down keyframes of a button symbol. When one of those button keyframes is displayed, the animation in the movie clip plays.

This page intentionally left blank