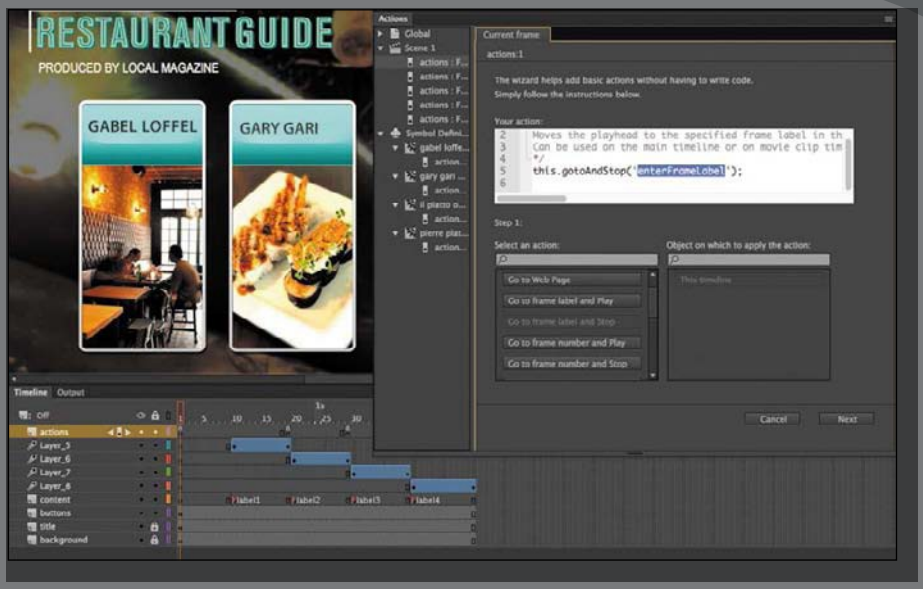
10 CREATING INTERACTIVE NAVIGATION

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
- Understand how ActionScript 3.0 and JavaScript are used in Animate documents
- Use the wizard in the Actions panel to quickly add JavaScript for interactivity
- Create and use frame labels
- Create animated buttons

This lesson will take about two hours to complete. Please log in to your account on peachpit.com to download the lesson files for this chapter, or go to the Getting Started section at the beginning of this book and follow the instructions under "Accessing the Lesson Files and Web Edition."



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

Getting Started

To begin, view the interactive restaurant guide that you'll create as you learn to make interactive projects in Adobe Animate CC.

Note If you have not already downloaded the project files for this lesson to your computer from your Account page, make sure to do so now . See "Getting Started" at the beginning of the book .

Note This project contains buttons and bitmaps that may generate security errors when you try to play the HTML file locally . Your browser may be blank or simply show a static picture when you double-click the HTML file to open it in a browser . Upload all required files to your server (see Lesson 12), or test the movie within Animate .

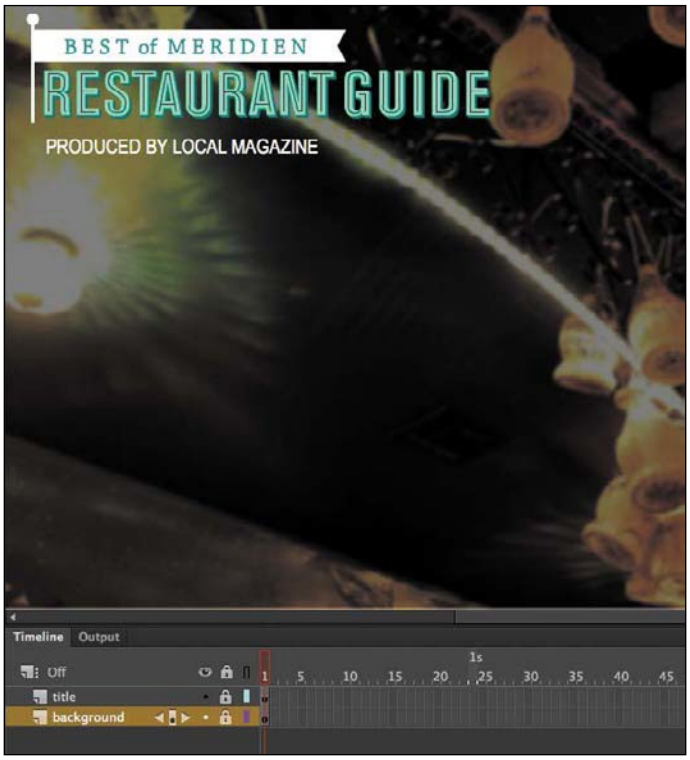
1 Double-click the 10End.fla file in the Lesson10/10End folder to open the animation in Animate. Choose Control > Test to see the final project.

The project opens in your default browser. You can ignore any warnings that appear in the Output panel.

The project is an imaginary city's interactive restaurant guide that runs in the browser. Viewers can click any button to see more information about a particular restaurant. In this lesson, you'll be working in an HTML5 Canvas document to create interactive buttons and structure the Timeline properly. You'll learn to add JavaScript code to provide instructions for what each button will do.

2 Close the 10End.fla file.

3 Double-click the 10Start.fla file in the Lesson10/10Start folder to open the initial project file in Animate. The file is an HTML5 Canvas document that will play in a browser. The document includes several assets already in the Library panel, and the Stage has already been sized properly.



4 Choose File > Save As. Name the file **10_workingcopy.fla** and save it in the 10Start folder. Saving a working copy ensures that the original start file will be available if you want to start over.

Note Animate may warn you if your computer doesn't have the same fonts contained in a FLA file. Choose substitute fonts, or simply click Use Default to have Animate automatically make the substitutions.

About Interactive Movies

Interactive movies change based on the viewer's actions. For example, when the viewer clicks a button, a different graphic with more information 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, keystrokes from the keyboard, or even the tilting of a mobile device.

ActionScript and JavaScript

In Animate, you add interactivity with either ActionScript 3.0 or JavaScript, depending on the kind of document you're working in.

If you're working in an ActionScript 3.0, AIR for Desktop, or AIR for iOS or Android document, you use ActionScript to achieve interactivity. ActionScript

provides the instructions that enable an animation to respond to the user. Those instructions could be to play a sound, skip to a keyframe on the Timeline where new graphics appear, or make a calculation.

In an HTML5 Canvas or WebGL document, you use JavaScript, the same code that drives interactivity for web pages in a browser.

While ActionScript 3.0 and JavaScript are very similar (in fact, both are based on an ECMA coding language standard), ActionScript contains language that controls Animate-specific features, and JavaScript contains language that controls browser-specific features such as scrolling and HTML elements on the page.

In this lesson, you'll use JavaScript in an HTML5 Canvas document to learn to create a nonlinear navigation—one in which the movie doesn't have to play straight from the beginning of the Timeline to the end. You'll add JavaScript to tell the Animate playhead to jump around and 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; the user just sees (or hears) different content appear as the buttons are clicked on the Stage.

Don't worry if you don't think you're good at programming! You don't have to be a code ninja, because Animate provides an easy-to-use menu-driven wizard in the Actions panel that allows you to add JavaScript quickly and simply.

Creating Buttons

A button is a basic visual indicator of something the user can interact with. The user often clicks a button with the mouse or taps a button with their finger, but many other types of interactions are possible. For example, something can happen when the user rolls the mouse pointer over a button.

A button is a kind of symbol that has four special states, or keyframes, that determine how the button appears. Buttons can look like virtually anything—an image, a graphic, or a bit of text. They don't have to be those typical pill-shaped gray rectangles that you see on many websites.

Creating a button symbol

In this lesson, you'll create buttons with small thumbnail images and restaurant names. A button symbol's four special states are represented on the button's timeline as frames, just like on the main Timeline. The four frames include the following:

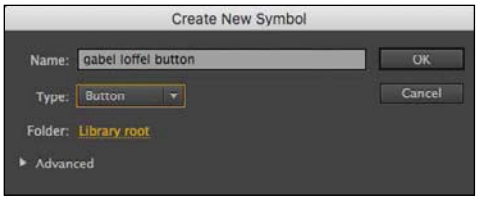
- The Up state shows the button as it appears when the mouse is not interacting with it.

- The Over state shows the button as it appears when the mouse cursor is hovering over it.
- The Down state shows the button as it appears when the mouse cursor is hovering over it and the mouse button is depressed.
- The Hit state indicates the clickable area of the button.

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

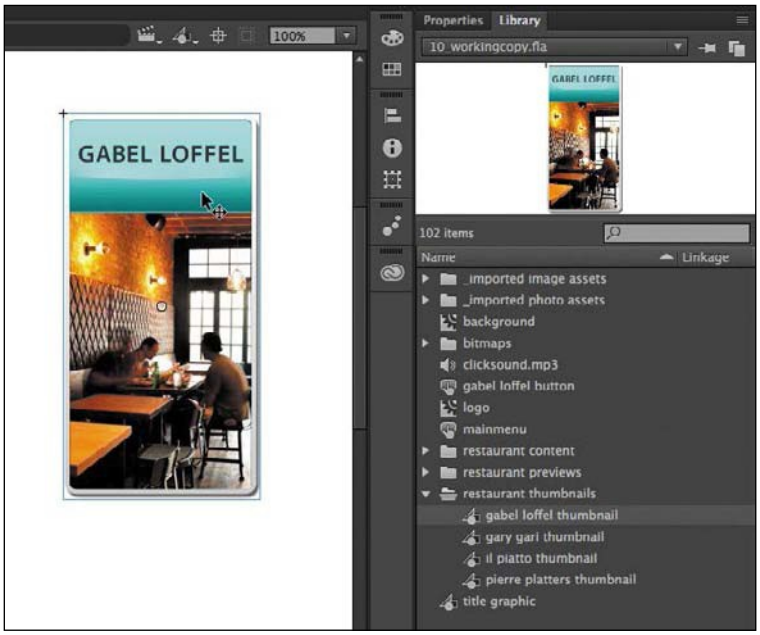
1 Choose Insert > New Symbol.

2 In the Create New Symbol dialog box, choose Button from the Type menu and name the symbol **gabel loffel button**. Click OK.



Animate puts you in symbol-editing mode for your new button.

3 In the Library panel, expand the restaurant thumbnails folder and drag the graphic symbol gabel loffel thumbnail to the middle of the Stage.

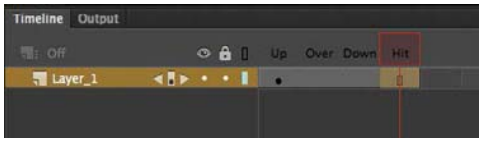


4 In the Properties panel, set the X value to **0** and the Y value to **0**.

The upper-left corner of the small gabel loffel restaurant image is now aligned to the registration point of the symbol (marked by the small cross in the center of the screen).

5 Select the Hit frame in the Timeline and choose Insert > Timeline > Frame to extend the Timeline.

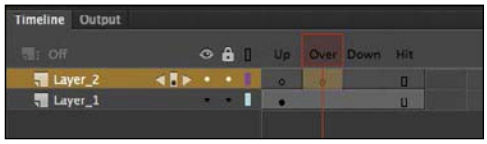
The gabel loffel image now extends through the Up, Over, Down, and Hit states.



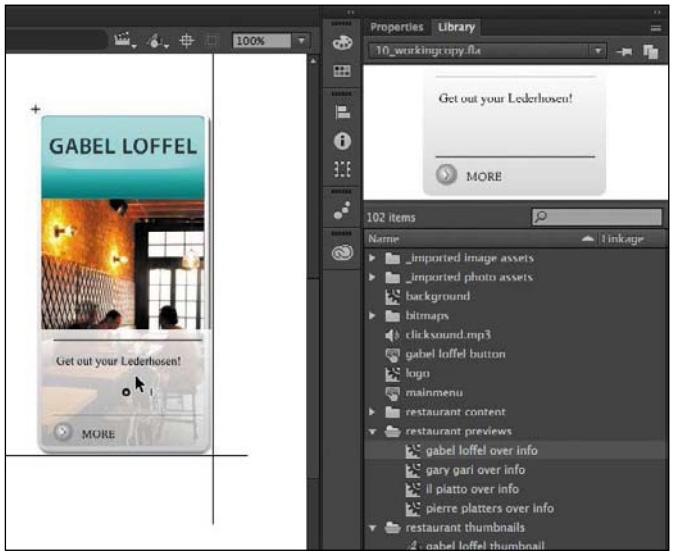
6 Insert a new layer. The new layer will accommodate an image that appears when the user's mouse cursor hovers over the button.

7 In the new layer, select the Over frame and choose Insert > Timeline > Keyframe.

Animate inserts a new keyframe in the Over state of the top layer.



8 In the Library panel, expand the restaurant previews folder, and drag the "gabel loffel over info" movie clip symbol onto the Stage.



9 Move the move clip symbol until it's over the lower part of the button. When guides appear along the bottom and side of the object, let it snap into place. In the Properties panel, the Position and Size section should show approximately X=0 and Y=217.

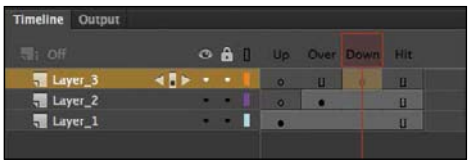
The gray information box will appear over the restaurant image whenever the mouse cursor rolls over the button.

10 Insert a third layer above the first two.

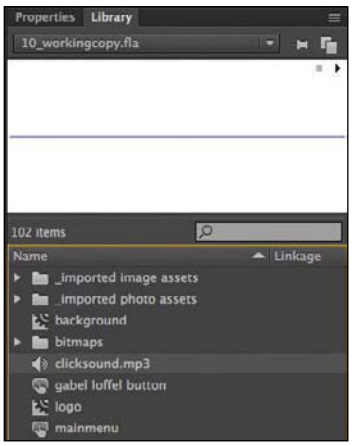
We'll use this layer to bind a sound file to the Down state so that the button will make a click sound when it's depressed.

11 Select the Down frame on the new layer and choose Insert > Timeline > Keyframe.

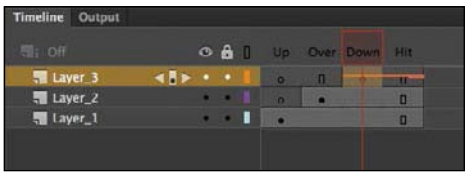
A new keyframe is inserted in the Down state of the new layer.



12 Drag the sound file called clicksound.mp3 from the Library panel to the Stage.

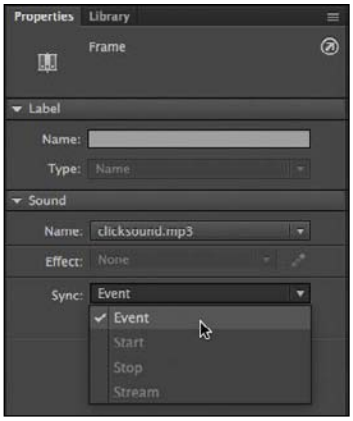


The beginning of the sound's waveform (appearing as a straight line) appears in the Down keyframe of the top layer of your button symbol.



Note You'll learn more about sound in Lesson 11.

13 Select the Down keyframe where the waveform appears, and in the Sound section of the Properties panel, note that Event is chosen in the Sync menu.



A clicking sound will play only when a viewer depresses the button.

14 Click Scene 1 in the Edit bar above the Stage to exit symbol-editing mode and return to the main Timeline. Your first button symbol is complete! Look in your Library panel 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 by the user . Normally, the Hit keyframe contains a shape that is the same size and location as the shape in your Up keyframe . In most cases, you want the graphics that users see to be in 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 . You can place invisible buttons over any part of the Stage and use ActionScript to program them to respond to users.

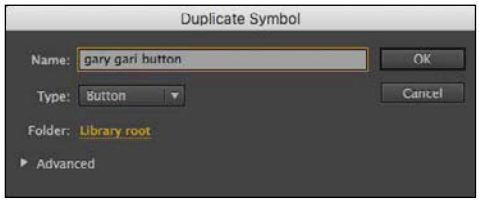
Invisible buttons are useful for creating generic hotspots . For example, placing them on top of different photos can help you make each photo respond to a mouse click without having to make each photo a different button symbol .

Duplicating buttons

Now that you've created one button, you'll be able to create others more easily. You'll duplicate one button here, change the image in the next section, and then continue to duplicate buttons and modify images for the remaining restaurants.

1 In the Library panel, right-click the gabel loffel button symbol and choose Duplicate. You can also choose Duplicate from the Library panel menu.

2 In the Duplicate Symbol dialog box, choose Button from the Type menu, and name the symbol **gary gari button**. Click OK.



Swapping bitmaps

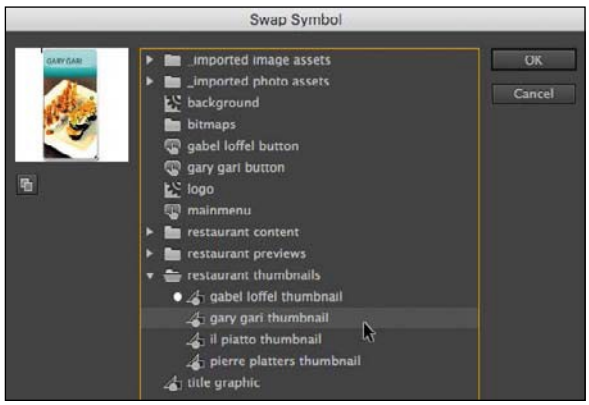
Bitmaps and symbols are easy to swap on the Stage and can significantly speed up your workflow.

1 In the Library panel, double-click the icon for your newly duplicated symbol (gary gari button) to edit it.

2 Select the restaurant image on the Stage.

3 In the Properties panel, click Swap.

4 In the Swap Symbol dialog box, select the next thumbnail image, gary gari thumbnail, in the restaurant thumbnails folder, and click OK.



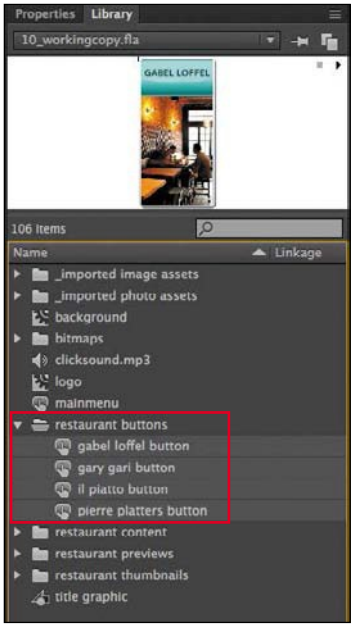
The original thumbnail (shown with a dot next to the symbol name) is swapped out for the one you selected. Because they are both the same size, the replacement is seamless.

5 Now select the Over keyframe on Layer 2, and click the gray information box on the Stage.

6 In the Properties panel, click Swap and swap the selected symbol with the symbol called "gary gari over info."

The instance of your button in the Over keyframe is replaced with one that is appropriate for the second restaurant. Because the symbol was duplicated, all other elements, such as the sound in the top layer, remain the same.

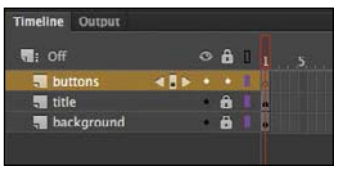
7 Continue duplicating your buttons and swapping the two instances inside them until you have four different button symbols in your Library panel, each representing a different restaurant (gabel loffel button, gary gari button, il piatto button, and pierre platters button). When you're done, take a moment to organize all your restaurant buttons in a folder in your Library panel.



Placing the button instances

Now you'll put the buttons on the Stage and give them names in the Properties panel so that your code can identify them.

1 On the main Timeline, insert a new layer and rename it **buttons**.

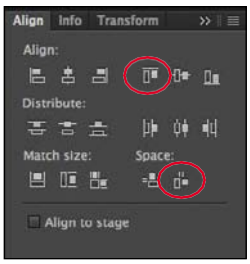


2 Drag each of your buttons from the Library panel to the middle of the Stage, placing them in a horizontal row. Don't worry about their exact position, because you'll align them nicely in the next few steps.

3 Select the first button, and in the Properties panel, set the X value to **100**.

4 Select the last button, and in the Properties panel, set the X value to **680**.

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



All four buttons are now evenly distributed and aligned horizontally.

6 With all the buttons still selected, in the Properties panel, enter **170** for the Y value.

All four buttons are positioned on the Stage correctly.

Note If your browser displays a blank screen when you test your movie (Control > Test), make sure you're connected to the Internet . If you're not, then open the Publish settings (File > Publish Settings). Select the Advanced tab and deselect Hosted Libraries. The Hosted Libraries option links to external JavaScript code, so your published files don't have to include them, but you need to be connected to the Internet for your project to work .

7 Now you can test your movie to see how the buttons behave. Choose Control > Test.

You can ignore any warnings that show up in the Output panel.

Note that the gray information box in the Over keyframe appears when your mouse cursor hovers over each button, and that pressing your mouse button over each button triggers the clicking sound. 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 coding.

Naming button instances

Next you'll name each button instance so that your code can reference it. Beginners often forget this crucial step.

1 Click in an empty part of the Stage to deselect all the buttons, and then select just the first button.

2 Type **gabelloffel_btn** in the Instance Name field in the Properties panel.

3 Name the other buttons **garygari_btn**, **ilpiatto_btn**, and **pierreplatters_btn**.

Animate is very picky, and one typo will prevent your entire project from working correctly! See the sidebar "Naming rules" for information about instance names.

4 Lock all the layers.

Naming rules

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

Instance names are important because ActionScript and JavaScript use the names to reference those objects . Instance names are not the same as the symbol names in the Library panel . The names in the Library panel are simply organizational reminders .

Instance naming follows these simple rules and best practices:

1 Do not use spaces or special punctuation . Underscores are OK to use .

2 Do not begin a name with a number .

3 Be aware of uppercase and lowercase letters . ActionScript and JavaScript are case-sensitive .

4 End your button name with _btn . Although it is not required, it helps identify those objects as buttons .

5 Do not use any word that is reserved for an Animate ActionScript or JavaScript command .

Preparing the Timeline

Every new Animate project begins with just a single frame. To create room on the Timeline to add more content, you'll have to add more frames to at least one layer.

1 Select a later frame in all three layers. In this example, select frame 50.

2 Choose Insert > Timeline > Frame (F5). You can also right-click and choose Insert Frame.

Animate adds frames in all of the selected layers up to the selected point, frame 50.

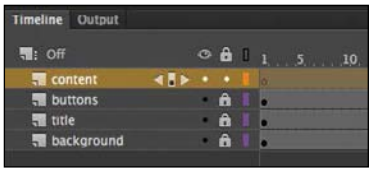
Creating Destination Keyframes

When the user clicks each button, Animate will move the playhead to a new spot on the Timeline according to code that you'll insert. Before adding the code, you'll create all the different options on the Timeline that your user might choose.

Inserting keyframes with different content

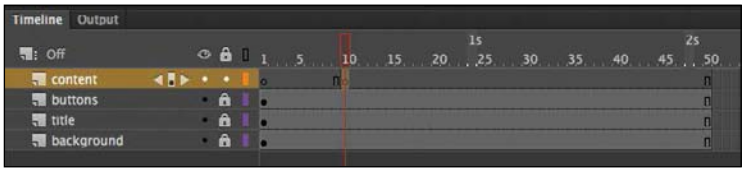
You will create four keyframes in a new layer and place information about each of the restaurants in the new keyframes.

1 Insert a new layer at the top of the layer stack and rename it **content**.



2 Select frame 10 of the content layer.

3 Insert a new keyframe at frame 10 (choose Insert > Timeline > Keyframe, or press F6).



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

Your Timeline has four empty keyframes in the content layer.

5 Select the keyframe at frame 10.

6 In the Library panel, expand the folder called restaurant content. Drag the symbol called "gabel and loffel" from the Library panel to the Stage. The symbol named gabel and loffel is a movie clip symbol that contains a photo, graphics, and text about the restaurant.

7 Position the symbol in the center of the Stage but out of the way of the heading. The Properties panel Position and Size section should show X=60 and Y=150.

The restaurant information about gabel and loffel is centered on the Stage and covers all the buttons.

8 Select the keyframe at frame 20.

9 Drag the symbol called gary gari from the Library panel to the Stage so that it also covers all the buttons. This is another movie clip symbol that contains a photo, graphics, and text about this restaurant.

10 In the Properties panel, make sure the X value is set to 60 and the Y value is set to 150.

11 Place each of the movie clip symbols from the restaurant content folder in the Library panel at the corresponding keyframes in the content layer.

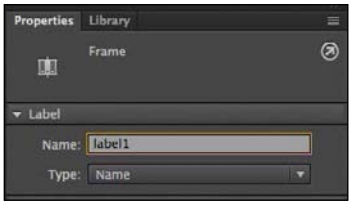
Each keyframe should contain a different movie clip symbol about a restaurant.

Using labels on keyframes

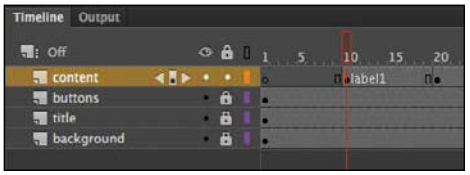
Frame labels are names that you give to keyframes. Instead of referring to keyframes by their frame number, you refer to them by their label, which makes coding easier to read, write, and edit.

1 Select frame 10 on the content layer.

2 In the Properties panel Label section, enter **label1** in the Name field.



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



3 Select frame 20 on the content layer.

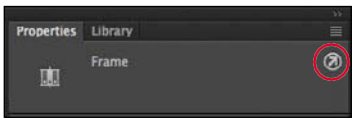
4 In the Properties panel, in the Label section, enter **label2** in the Name field.

5 Select frames 30 and 40, and in the Properties panel, enter corresponding names in the Name field for each: **label3** and **label4**.

A tiny flag icon appears on each of the keyframes that has a label.

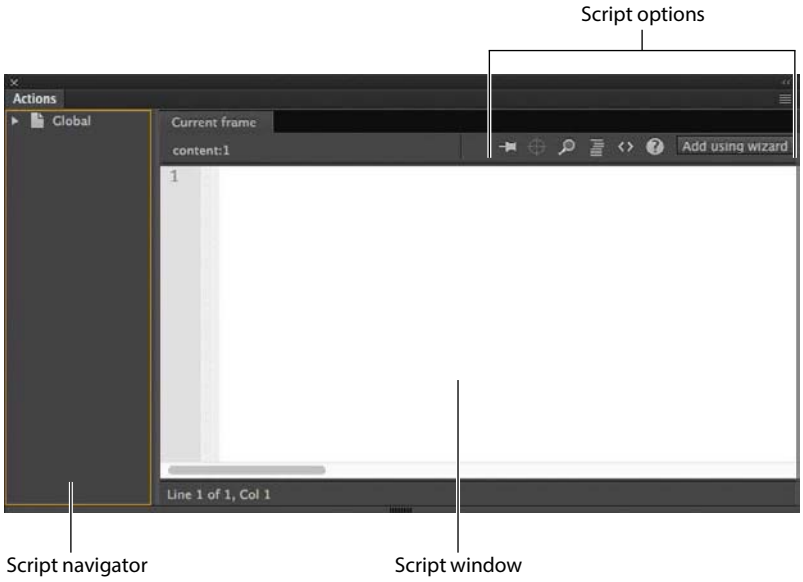
Navigating the Actions Panel

The Actions panel is where you enter all your code, whether JavaScript for HTML5 Canvas documents or ActionScript for targeting Flash Player or AIR. Open the Actions panel by choosing Window > Actions or by selecting a keyframe on the Timeline and clicking the Actions panel button on the top right of the Properties panel.



You can also right-click any keyframe and choose Actions.

The Actions panel gives you a flexible environment for entering code, as well as different options to help you write, edit, and view your code.



The Actions panel is divided into two parts. On the right of the Actions panel is the Script window—the blank slate where you can write code. You enter ActionScript or JavaScript in the Script window just as you would in a text-editing application.

On the left is the Script navigator, which tells you where your code is located. Animate places code 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.

At the bottom of the Actions panel, Animate displays the line number and column number (or character in the row) of the current position of the text insertion point.

The upper-right corner of the Actions panel contains options for finding, replacing, and inserting code. The Add Using Wizard button is also located there.

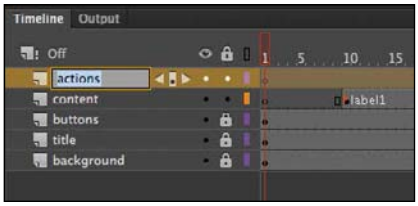
Add JavaScript Interactivity with the Actions Panel Wizard

Now that you have multiple keyframes on the Timeline, your movie will play linearly from frame 1 to frame 50, showing all the restaurant choices. However, with this interactive restaurant guide, you'll want to pause the movie at the very first frame to wait for your viewers to choose a restaurant.

Stopping the Timeline

Use a stop action to pause your Animate movie. A stop action simply stops the movie from continuing by halting the playhead.

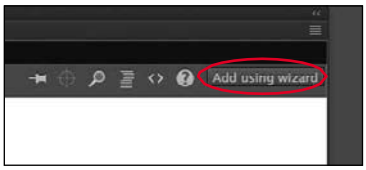
1 Insert a new layer at the top and rename it **actions**.

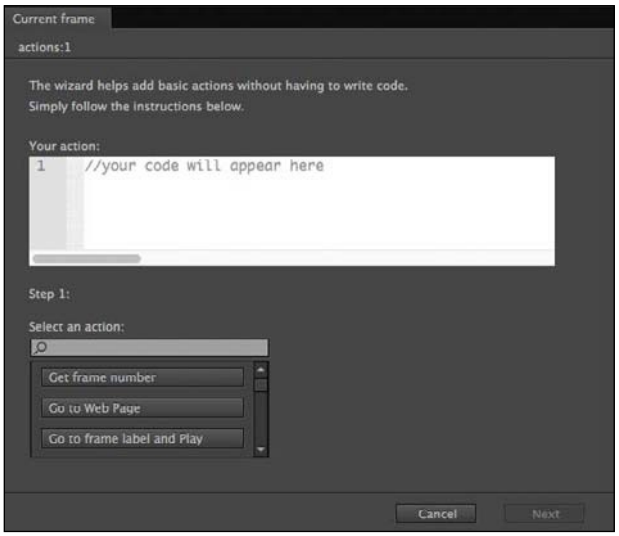


JavaScript and ActionScript code are generally placed on keyframes on the Timeline.

2 Select the first keyframe of the actions layer and open the Actions panel (Window > Actions).

3 Click the Add Using Wizard button.



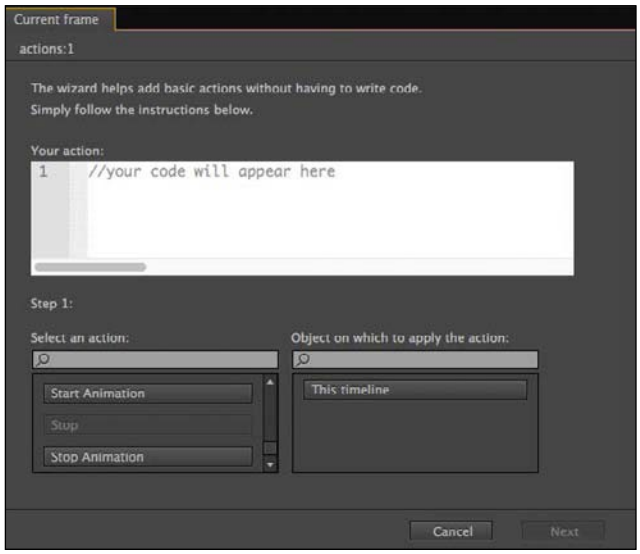


The wizard opens within the Actions panel. The wizard guides you through the code-writing process in a step-by-step process. The code that you generate with the wizard appears in the first field. The wizard is available to insert JavaScript

into HTML5 Canvas documents. For ActionScript, you can use the Code Snippets panel.

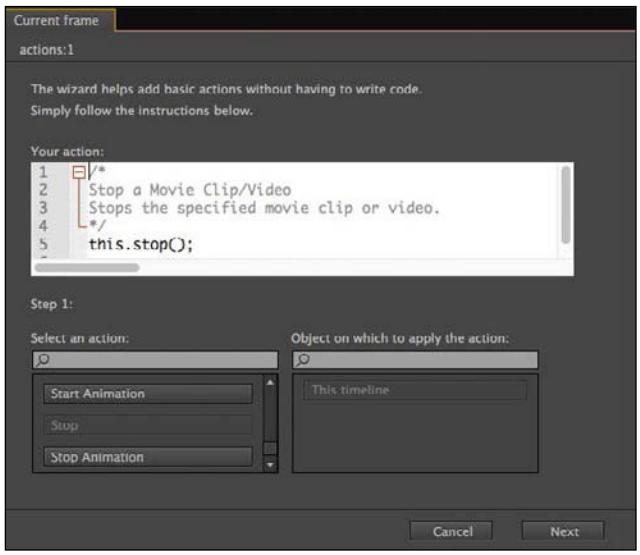
4 Step 1 asks you to choose the action, or the behavior that you want Animate to perform, from a list. Scroll down and select Stop.

Another menu appears to the right.



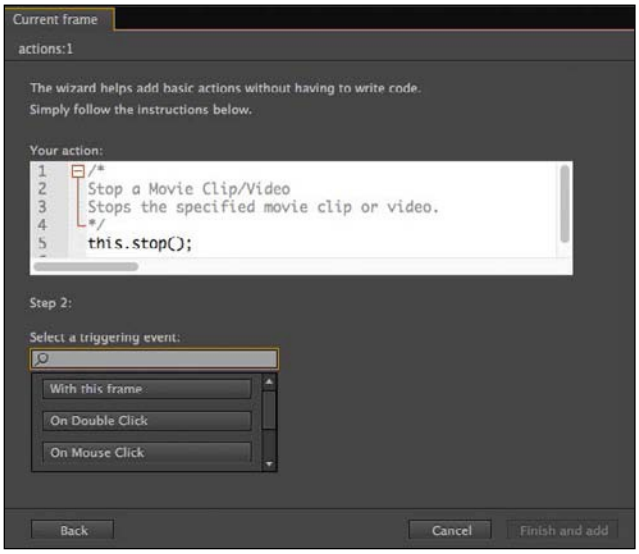
5 In the next menu, choose This Timeline.

Code appears in the action window. The stop action will apply to the current timeline.



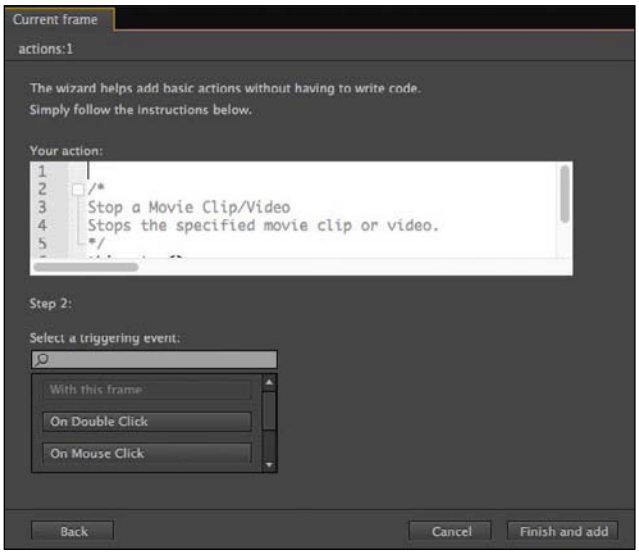
6 Click Next.

Step 2 appears in the wizard.



7 Step 2 asks you to select the trigger that will produce your selected action.

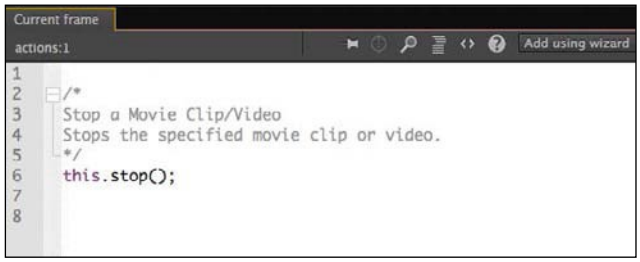
8 Select With This Frame.



You want the stop action to be executed as soon as the timeline begins, so the appropriate trigger is when the playhead encounters the current frame.

9 Click Finish And Add.

The finished code is added to the Script window in the Actions panel.



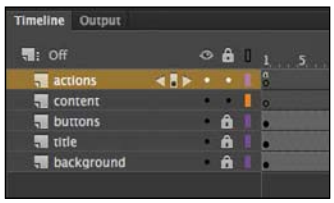
The code is:

```
this.stop();
```

The first word, `this`, refers to the current timeline, and the stop action is `stop()`. The semicolon at the end of the statement acts as a period and indicates the end of the command.

The code in light gray that begins with `/*` and ends with `*/` is called a multiline comment, and it is simply a description of what the code does. It acts as a reminder for you and for other developers. Well-commented code is essential—it will save you hours of headache when you return to a project, and it is considered best practice for coders.

10 In the timeline, a tiny letter "a" is added to the first frame in the Actions panel to indicate that code has been added there.



Adding actions for a button click

So far you have code that stops the Timeline at the first frame. Now you'll add an action for a button click. The button click is called a *trigger* in the wizard, but in JavaScript and ActionScript it is known as an *event*.

Events are occurrences in your movie that Animate can detect and respond to. For example, a mouse click, a mouse movement, and a key press are all events. Pinch and swipe gestures on mobile devices are also 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.

1 Select the first frame of the actions layer.

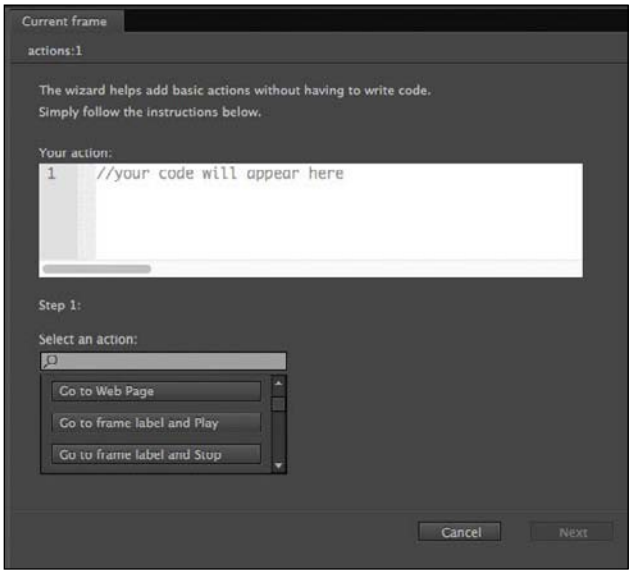
2 Open the Actions panel, if it's not already open.

3 Place your text cursor at the last line of your Script window. You'll add additional code to the stop code that's already there.

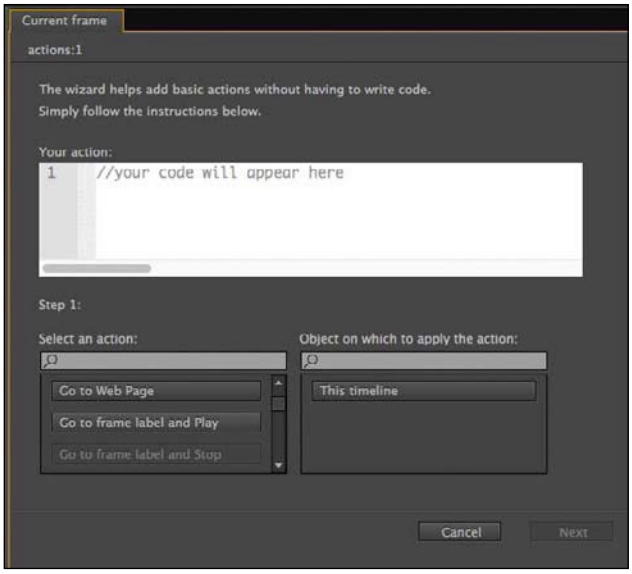
4 Click the Add Using Wizard button.

The wizard opens within the Actions panel.

5 Step 1 asks for the action. Scroll down and select Go To Frame Label And Stop.

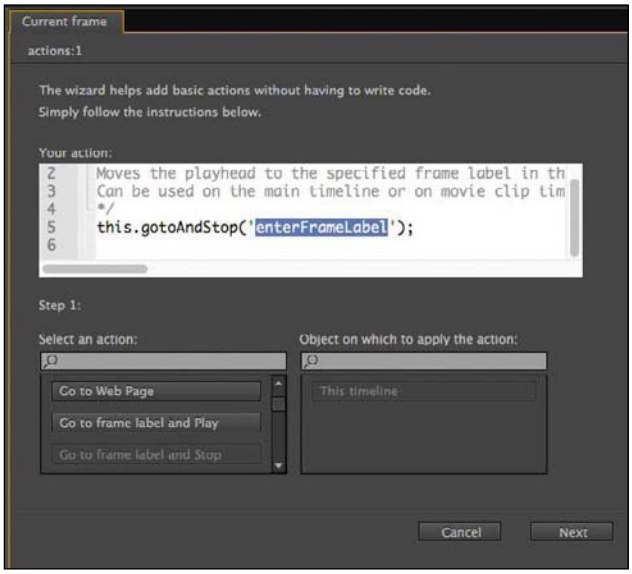


Another menu appears to the right.

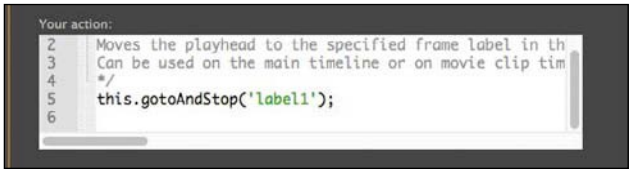


6 In the next menu, choose This Timeline.

Code appears in the action window. The action will apply to the current timeline.



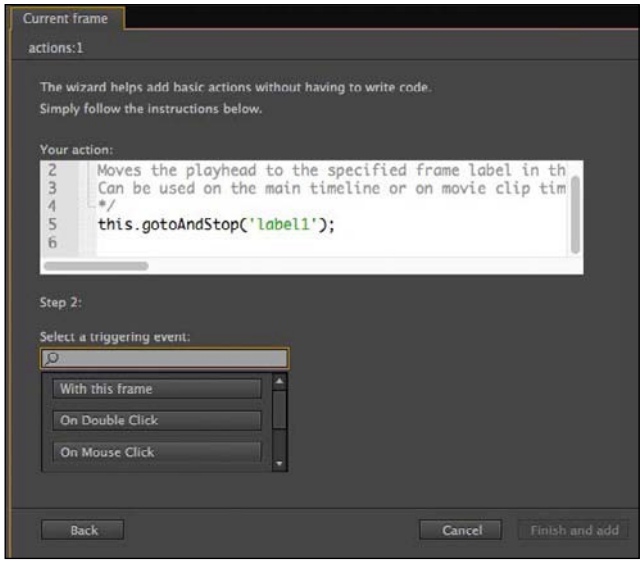
7 Replace the blue highlighted letters in the action window with the name of the label you want the playhead to go to. To replace `enterFrameLabel`, enter **label1**.



The frame label name appears in green and should be between a set of single quotes.

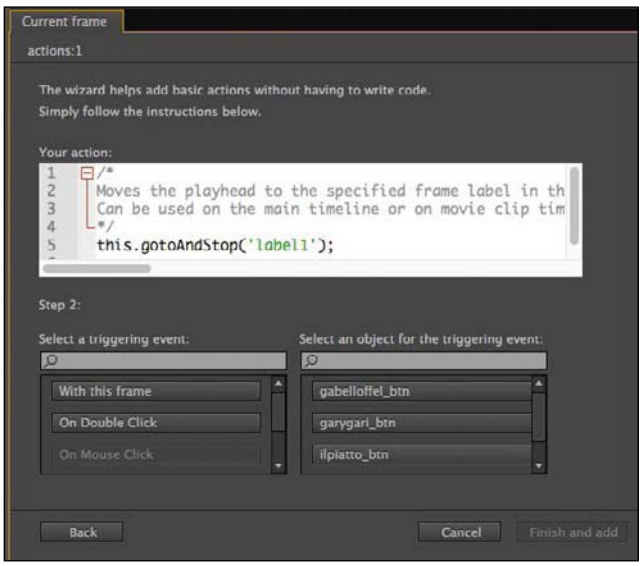
8 Click Next.

Step 2 appears in the wizard.

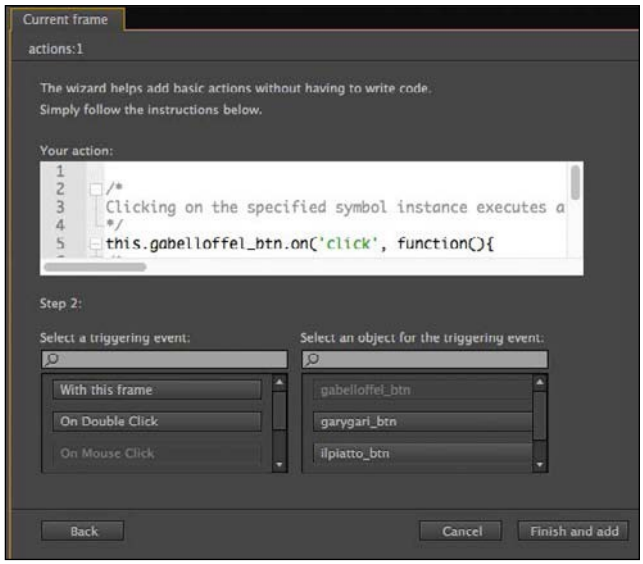


9 Step 2 asks for the trigger that will produce your selected action. Select On Mouse Click.

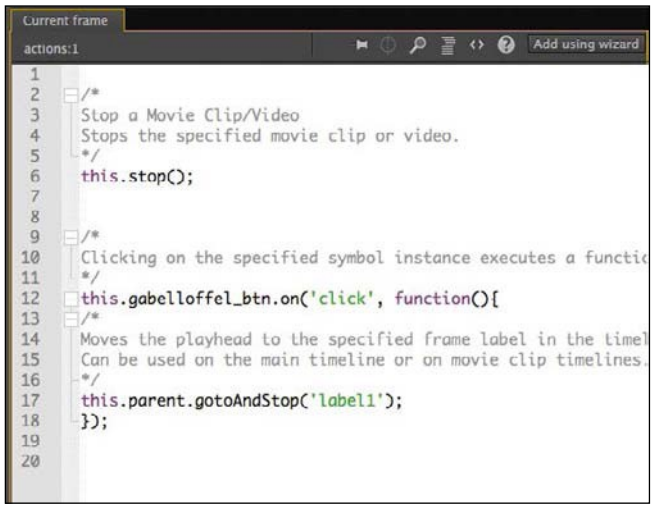
On Mouse Click is an event that happens when the user's mouse is pressed and then released over the button. Another menu appears to the right.



10 The wizard asks for the object for the triggering event. Select gabelloffel_btn, which is the button that corresponds to the Gabel and Loffel restaurant, whose information is displayed in the keyframe labeled "label1."



11 Click Finish And Add.



The finished code is added to the Script window in the Actions panel. The code consists of the trigger `'click'` and a function that groups all the code that is executed when the trigger happens. What's important for you to identify are the opening and closing curly braces of the function. This function has only one statement (a `gotoAndStop` action, which moves the playhead), but functions can contain many statements.

12 Choose Control > Test.

Animate opens your browser to show your project. Click the Gabel Loffel button. Animate detects the click trigger on the button and moves the playhead to the keyframe labeled "label1," where the Stage shows information about the Gabel and Loffel restaurant.

13 Close your browser and return to Animate.

14 Select the first frame of the actions layer and open the Actions panel again.

15 Continue adding additional actions and triggers to the existing code for the other three buttons. Each button should trigger a `gotoAndStop` action to a different keyframe.

Note If you're feeling confident, you can try to copy and paste code in the Script window and just change the name of the button and the name of the frame label . It will be faster than going through the wizard, and it will be your first step in identifying and learning different parts of JavaScript code so that you can eventually write code yourself.

Checking for errors

Debugging is a necessary process, even for veteran coders . Even if you're extra careful, errors will creep into your code . Fortunately, the wizard helps reduce typos and common errors . If you do enter code by hand, a few tips can help prevent, catch, and identify errors:

1 If you're working in an ActionScript 3.0 document, Animate automatically displays code errors in the Compiler Errors panel (Window > Compiler Errors) with a description of the error and its location . None of your code will be functional if there is a compiler error in any part of the code.

2 Take advantage of color hinting in the code . Animate colors keywords, variables, comments, and other language elements differently . You don't need to know why they are different, but the different colors can give you clues as to where there may be some missing punctuation.

3 Click the Format Code button at the upper-right corner of the Actions panel to tidy up your code and make it easier to read . You can change the formatting settings in Edit > Preferences > Code Editor (Windows) or Animate CC > Preferences > Code Editor (Mac).

Creating a Home Button

Since the information for each restaurant covers the buttons, readers currently can't make another selection after they choose their first. You'll need to provide another button that will return users to the first frame, which you'll do in the next section.

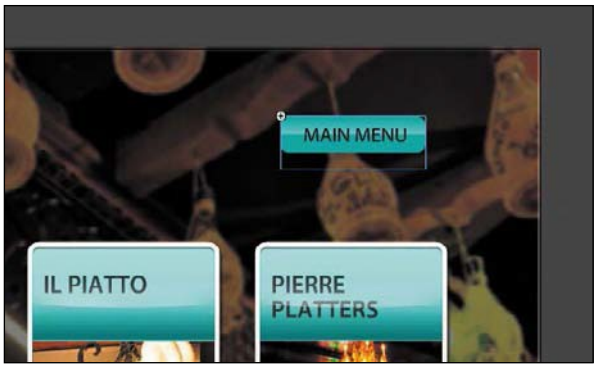
A home button simply makes the playhead go back to the first frame of the Timeline, or to a keyframe where an original set of choices, or the main menu, are presented to the viewer. You can create a button that goes to the first frame using the same process you followed to create the four restaurant buttons.

Adding another button instance

The sample lesson file provides a home, or mainmenu, button for you in the Library panel.

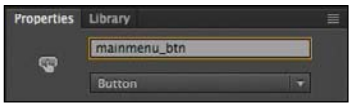
1 Select the buttons layer and unlock it if it is locked.

2 Drag the button called mainmenu from the Library panel to the Stage. Position the button instance at the upper-right corner.



3 In the Properties panel, set the X value to **726** and the Y value to **60**.

4 In the Properties panel, name the instance **mainmenu_btn**.



Adding code for the home button

The action will be Go To Frame Number And Stop, and the trigger will be a button click.

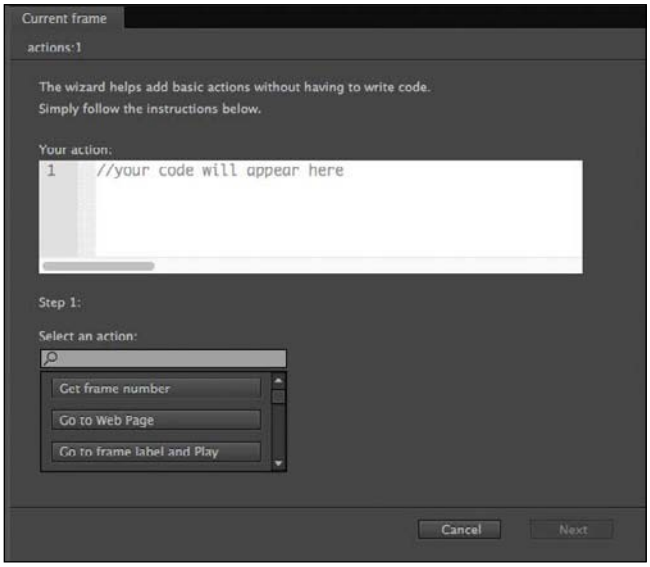
1 Select the first frame of the actions layer.

2 Open the Actions panel, if it's not already open.

3 Place your text cursor at a new line after the last line of code of your Script window. You'll add additional code to the stop code that's already there.

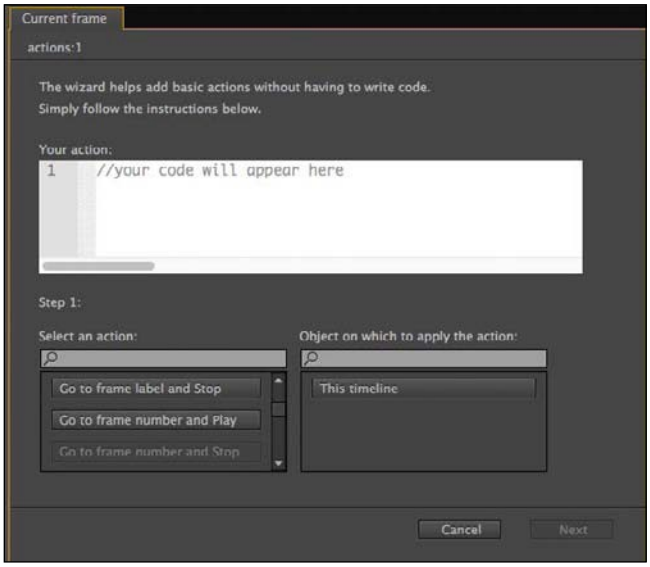
4 Click the Add Using Wizard button.

The wizard opens within the Actions panel.

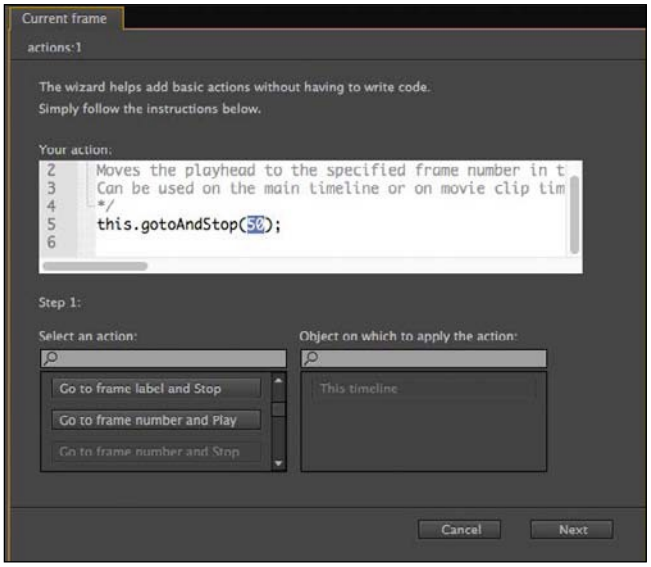


5 Step 1 asks for the action. Scroll down and select Go To Frame Number And Stop.

Another menu appears to the right.

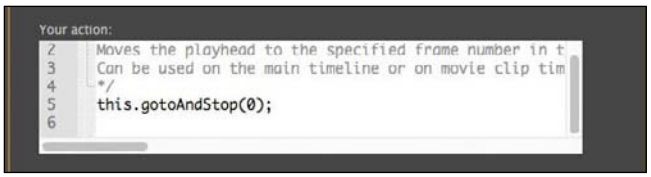


6 In the next menu, select This Timeline.



Code appears in the action window. The action will apply to the current timeline.

7 Replace the blue highlighted letters in the action window with the frame number you want the playhead to go to. Instead of 50, enter **0**.

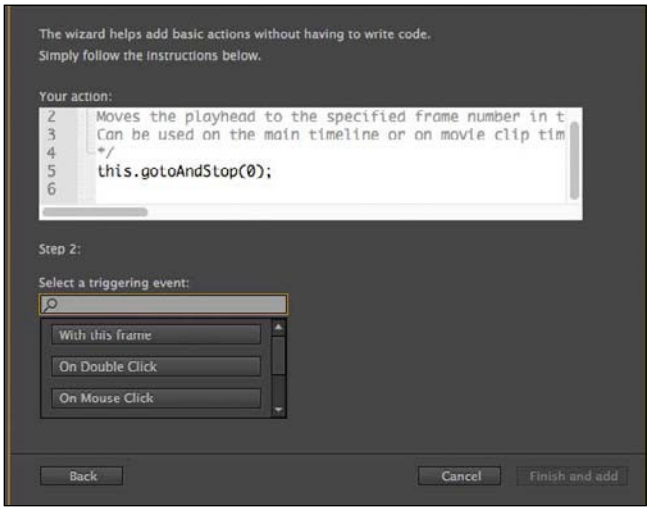


Why 0 and not 1? JavaScript begins counting frames at 0, so the first frame of your timeline is 0 and not 1. ActionScript, on the other hand, begins counting the timeline frames at 1, so be extra careful when coding frame numbers. For this reason, it's best to use frame labels whenever possible.

Notice, too, that frame numbers are not enclosed between single quotes. Frame labels, however, are.

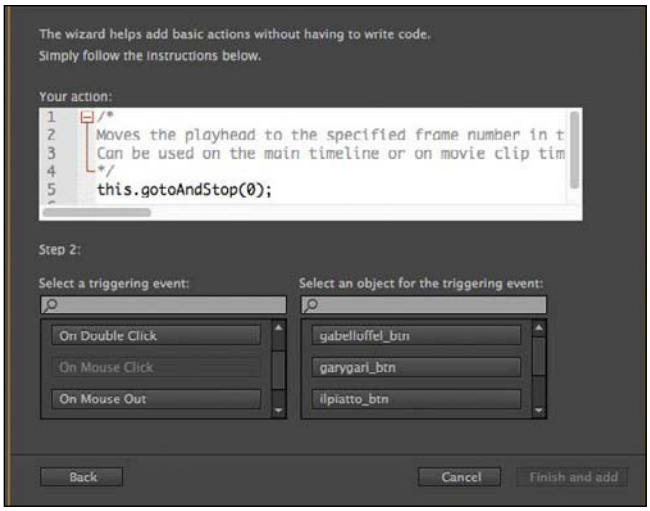
8 Click Next.

Step 2 appears in the wizard.

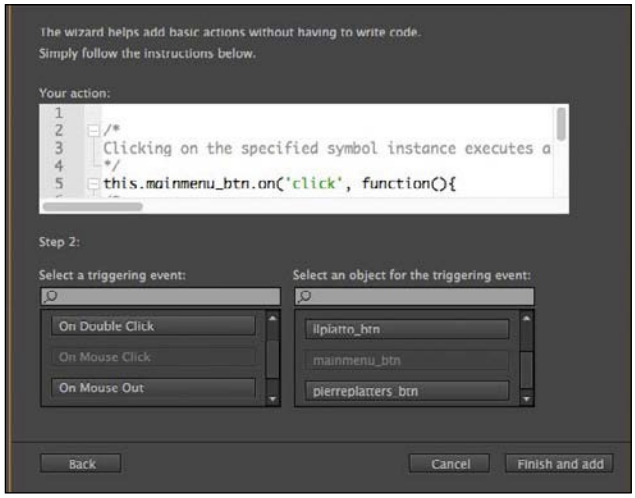


9 Step 2 asks for the trigger that will produce your selected action. Select On Mouse Click.

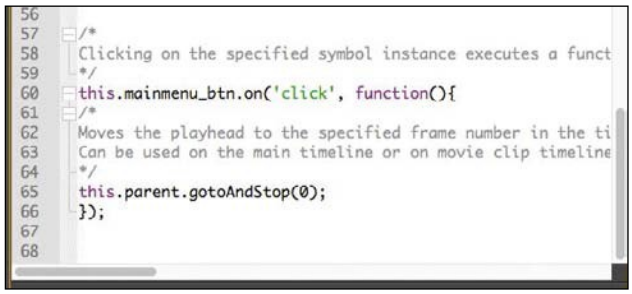
On Mouse Click is an event that happens when the user's mouse is pressed and then released over the button. Another menu appears to the right.



10 The wizard asks for the object for the triggering event. Select mainmenu_btn.

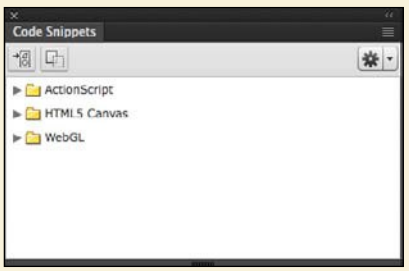


11 Click Finish And Add.



Code Snippets panel

Animate provides another panel, called the Code Snippets panel (Window > Code Snippets), that offers a way to add ActionScript 3.0 or JavaScript code. The panel is organized into folders for different kinds of interactivity . Simply expand the folder that you want and select the action . Animate guides you for any additional information .



The Code Snippets panel also provides a way for you to save your own code and share it with other developers .

For beginners, it's best to use the Add Using Wizard option in the Actions panel for JavaScript .

Playing Animation at the Destination

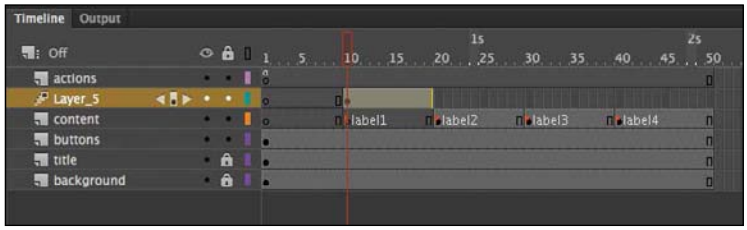
So far, this interactive restaurant guide works by using the `gotoAndStop()` command to show information in different keyframes along the Timeline. But suppose you wanted an image to fade in rather than appear suddenly—how would you play an animation after a user clicks a button? One way is to use the command `gotoAndPlay()`, which moves the playhead to the frame number or frame label and plays from that point forward.

Creating transition animations

Next, you'll create a short transition animation for each of the restaurant guides. The transition animation will show the restaurant guides slowly increase in opacity to fade up. Then you'll change your code to direct Animate to go to each of the beginning keyframes and play the animation.

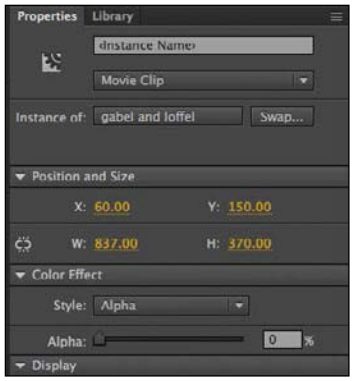
1. Move the playhead to the label1 frame label.
2. Right-click the instance of the restaurant information on the Stage and choose Create Motion Tween.

Animate creates a separate tween layer for the instance so that it can proceed with the motion tween.



3 In the Properties panel's Color Effect section, choose Alpha from the Style menu.

4 Set the Alpha slider to **0**%.



The instance on the Stage becomes totally transparent.

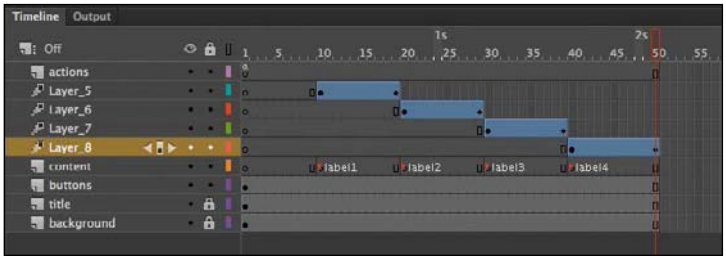
5 Move the playhead to the end of the tween span at frame 19.

6 Select the transparent instance on the Stage.

7 In the Properties panel, set the Alpha slider to **100**%.

The instance is displayed at a normal opacity level. The motion tween from frame 10 to frame 19 produces a smooth fade-in effect.

8 Create similar motion tweens for the remaining restaurants in the keyframes labeled label2, label3, and label4.



Using the gotoAndPlay command

The `gotoAndPlay` command makes the Animate 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 the first four `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');`.

For each of the restaurant buttons, the JavaScript code now directs the playhead to a particular frame label and begins playing at that point.

Make sure you keep the function for your Home button unchanged. You'll want that function to remain a `gotoAndStop()` command.

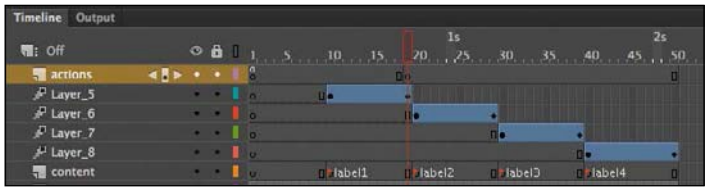
Tip A fast and easy way of doing multiple replacements is to use the Find And Replace command in the Actions panel . Click the Find button in the upper-right corner, select Find, and then choose Find And Replace from the menu to the right of the Find Text field .

Stopping the animations

If you test your movie now (Control > Test), 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 Animate when to stop.

1 Select frame 19 of the actions layer, the frame just before the label2 keyframe on the content layer.

2 Right-click and choose Insert Keyframe.



We'll use the new keyframe to add a stop action just before the second animation starts to play.

3 Open the Actions panel.

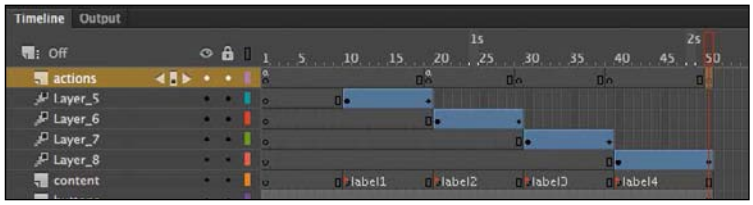
The Script window in the Actions panel is blank. Don't panic! Your code has not disappeared. Your code for the event listeners is on the first keyframe of the actions layer. You have selected a new keyframe in which you will add a stop command.

4 In the Script window, enter `this.stop();`.

Animate will stop playing when it reaches frame 19.

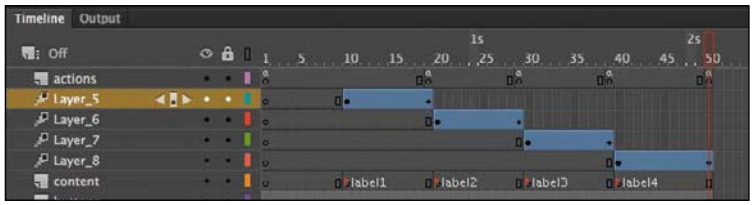
5 Insert keyframes at frames 29, 39, and 50.

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



6 In each of those keyframes, add a stop command in the Actions panel.

Tip If you wish, you could also use the Add Using Wizard panel to add the stop command for each of the keyframes .



7 Test your movie by choosing Control > Test.

Each button takes you to a different keyframe and plays a short fade-in animation. At the end of the animation, the movie stops and waits for the viewer to click the Home button.

Pinning code in the Actions panel

When you have code scattered in multiple keyframes on the Timeline, it's sometimes difficult to go back and forth to edit or view your code . The Actions panel provides a way for you to keep the code from particular keyframes "pinned" to the Actions panel . Click the Pin Script button at the top of the Actions panel and Animate will create a separate tab for the code currently displayed in the Script window .

The tab will be labeled with the frame number where your code resides . You can pin multiple scripts, allowing you to navigate easily between them .

Animated Buttons

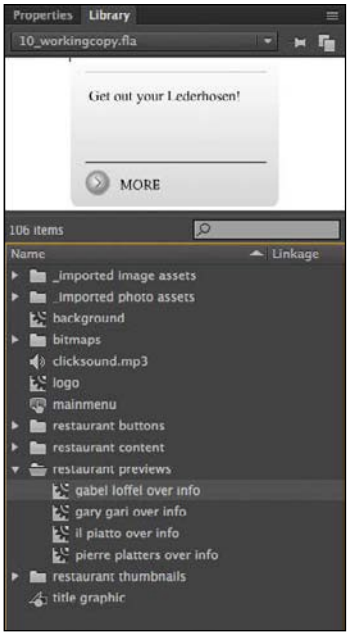
Currently, when you hover your mouse cursor over one of the restaurant buttons, the gray "additional information" box suddenly appears. But imagine if that gray information box were animated. It would give more life and sophistication to the interaction between the user and the button.

Animated buttons display an animation in the Up, Over, or Down keyframes. 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 interactive restaurant guide already contain a movie clip symbol of a gray information box in their Over states. You will edit each movie clip symbol to add an animation inside it.

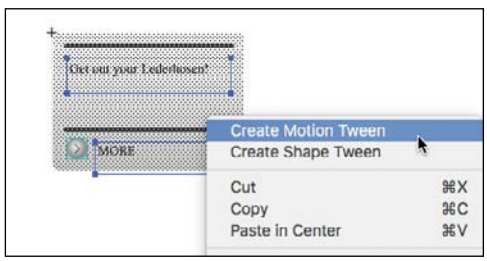
1 In the Library panel, expand the restaurant previews folder. Double-click the movie clip symbol icon for "gabel loffel over info."



You enter symbol-editing mode for the movie clip symbol called gabel loffel over info.

2 Select all the visual elements on the Stage (Ctrl+A/Command+A).

3 Right-click and choose Create Motion Tween.



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

Layer 1 is converted to a motion tween layer and adds 1 second's worth of frames to the movie clip Timeline.

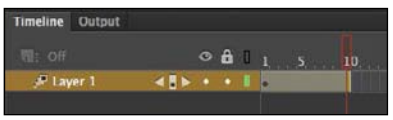
5 Drag the end of the tween span back so that the Timeline has only 10 frames.

6 Move the playhead to frame 1 and select the instance on the Stage.

7 In the Properties panel's Color Effect section, choose Alpha from the Style menu and set the Alpha value to **0**%.

The instance on the Stage becomes totally transparent.

8 Move the playhead to the end of the tween span at frame 10.

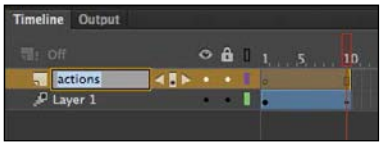


9 Select the transparent instance on the Stage.

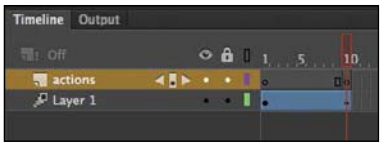
10 In the Properties panel, set the Alpha value to **100**%.

Animate creates a smooth transition between the transparent and opaque instance in the 10-frame tween span.

11 Insert a new layer and rename it **actions**.



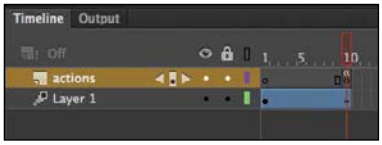
12 Insert a new keyframe in the last frame (frame 10) of the actions layer.



13 Open the Actions panel (Window > Actions) and enter `this.stop();` in the Script window.

Adding the stop action in the last frame ensures that the fade-in effect plays only once. The last keyframe on frame 10 of the actions layer shows a tiny "a," indicating that it has code attached there.

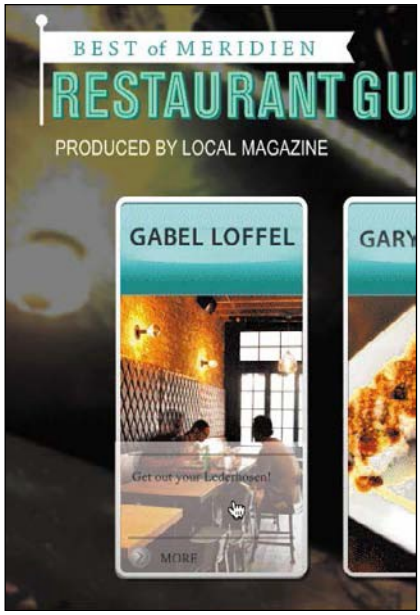
Tip If you want an animated button to repeat its animation, leave out the stop command at the end of the movie clip's Timeline.



14 Exit symbol-editing mode by clicking the Scene 1 button in the Edit bar above the Stage.

15 Choose Control > Test.

When your mouse cursor hovers over the first restaurant button, the gray information box fades in. The motion tween inside the movie clip symbol plays the fade-in effect, and the movie clip symbol is placed in the Over state of the button symbol.



16 Create identical motion tweens for the other gray information box movie clips to animate all of the restaurant buttons, and add a stop action to the end of those tweens.