Flash MX Tutorials

macromedia

Acknowledgments

Director: Erick Vera

Producer: Wayne Wieseler

Writing: Jody Bleyle, JuLee Burdekin, Mary Burger, Dale Crawford, Marcelle Taylor

Instructional Design: Stephanie Gowin, Barbara Nelson

Editing: Rosana Francescato, Lisa Stanziano, Anne Szabla

Multimedia Design and Production: Aaron Begley, Benjamin Salles, Noah Zilberberg

Print Design and Production: Chris Basmajian, Caroline Branch

First Edition: February 2002

Macromedia, Inc.
600 Townsend St.
San Francisco, CA 94103

CONTENTS

CHAPTER 1
Introduction to Flash MX Tutorial

This tutorial guides you through the process of creating a compelling Web experience with Macromedia Flash MX. By completing the tutorial, you'll learn how to design a movie, from opening a new document to publishing the movie for Web playback. The tutorial takes approximately three hours to complete, depending on your experience, and will teach you how to do the following tasks:

- Analyze a completed movie
- Define document properties and create a gradient
- Create and mask vector art
- Tween bitmap effects within a movie clip
- Load dynamic text
- Modify buttons and add navigation
- Add streaming and event sounds
- Test and publish the movie

We recommend that you complete the eight sections that comprise the tutorial in sequence, although you may choose to review only the sections that interest you. If you do complete the tutorial out of sequence, keep in mind that later sections assume you've mastered skills introduced in earlier sections.

What you should know

Before taking the tutorial, complete the seven lessons found in Flash Help. These interactive lessons created in Flash introduce you to the concepts you need to know to complete the tutorial. Lesson topics include the following:

- Getting Started with Flash MX
- Illustrating in Flash
- Adding and Editing Text
- Creating and Editing Symbols
- Understanding Layers
- Creating Buttons
- Creating Tweened Animation

To take a lesson, choose Help > Lessons, then select from the list.

View the completed movie

You can open a completed version of the tutorial movie to better understand how your finished file will appear.

In this section, you'll accomplish the following tasks:

- Analyze the completed movie using the Property inspector and Movie Explorer
- Examine a movie clip and discern its relationship to the main movie
- View the types of assets included in the movie

1. Within your Flash MX application folder, browse to Tutorials/FlashIntro and double-click stiletto.swf to open the completed movie in the stand-alone Flash Player.

   Published Flash movies have the SWF extension; documents in the authoring environment have the FLA extension.

2. When the movie opens, watch the three views of the car fade in and out.

   You'll create this animation by tweening bitmap effects within a movie clip.

3. Listen to the sound that plays continuously while the movie plays. This is an example of a streaming sound.

4. Roll over the three buttons along the lower right edge of the window to view the rollover effect, and to hear the event sounds included in each button.

5. Click a button to see where it links, then close the browser that opened and return to the SWF file.

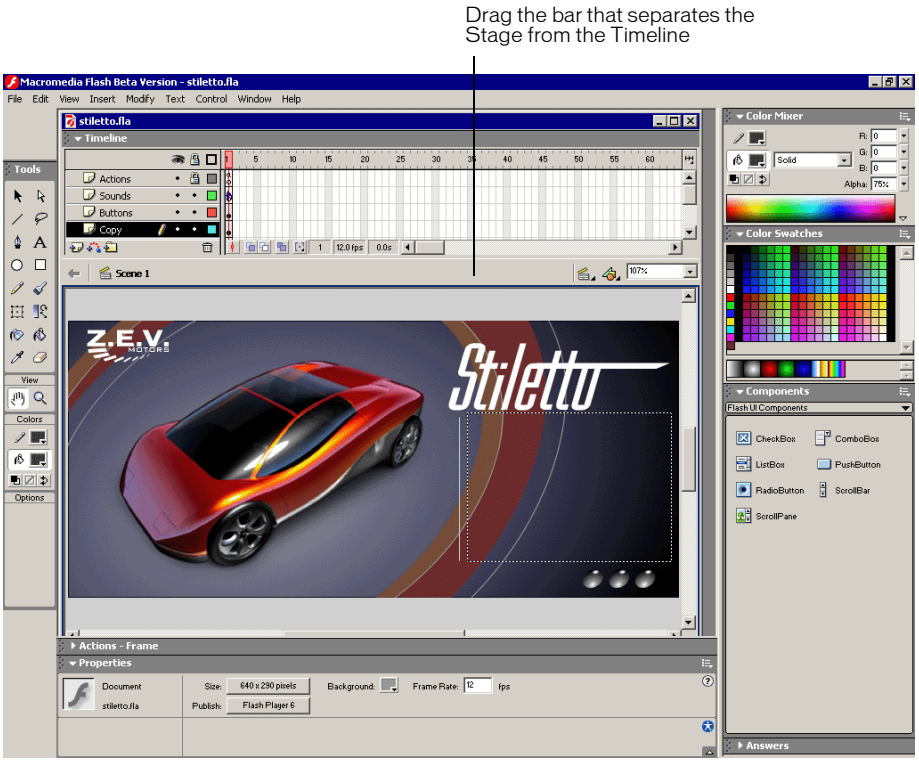
6. After viewing the movie, click its close box.

Analyze the stiletto.fla file

It's helpful to analyze the completed FLA file to see how the author designed the document. To analyze the file, you can view the properties for an object, view the Timeline and Stage, look at library assets, and use the Movie Explorer.

1 In Flash, choose File > Open. Navigate to your Flash application folder and open Tutorials/FlashIntro/stiletto.fla.

You now see the completed tutorial movie in the authoring environment.



2 To view all layers in the main Timeline, drag down the bar that separates the Stage from the Timeline.

3 In the Timeline, unlock the Copy layer and the Images layer.

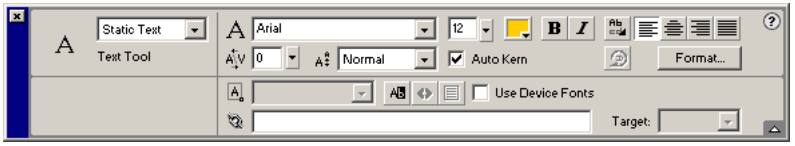
View document properties

The Property inspector lets you view specifications for selected objects. The specifications depend on the type of object selected. If you select a text object, for example, settings to view and modify text attributes appear.

1 If the Property inspector isn't open, choose Window > Properties.

2 On the Stage, scroll down if necessary and select the rectangular block where descriptive text appeared in the finished SWF file. The text does not appear in the FLA file because it loads from an external TXT file into a dynamic text field.

In the Property inspector, you can view the size, style, and color of the text, among other attributes.



If the Property inspector is not fully expanded, click the white triangle in lower right corner.

3 On the Stage, select the car.

Movie clip settings replace the text settings. Movie clips are symbols with their own independent Timelines. You can think of them as movies within movies.

View the movie clip

Now you'll open symbol-editing mode to view the Timeline for a movie clip.

1 On the Stage, double-click the movie clip of the car.

As you learned in the Creating Tweened Animation lesson, you define changes in animation in keyframes. When you scroll around the Timeline, note which layers have keyframes in them and which frames are keyframes.

Beginning and intermediate keyframes that include content are indicated in the Timeline by solid circles, ending keyframes appear as small outlined rectangles.

2 In the Timeline, select the playhead and drag it slowly across the frames.

Watch how changes in action on the Stage correspond to changes in the Timeline. As you drag the playhead, the movie plays sequentially. You can add ActionScript, the Flash scripting language, to movies to make the playhead jump to specific frames.

3 When you finish viewing the movie clip, do one of the following to return to the main movie:

- Choose Edit > Edit Document.
- Click the Back button.
- Click Scene 1 above the Stage.

View library assets

The Library panel contains the symbols and imported objects in your document.

1 If the Library panel isn't open, choose Window > Library.

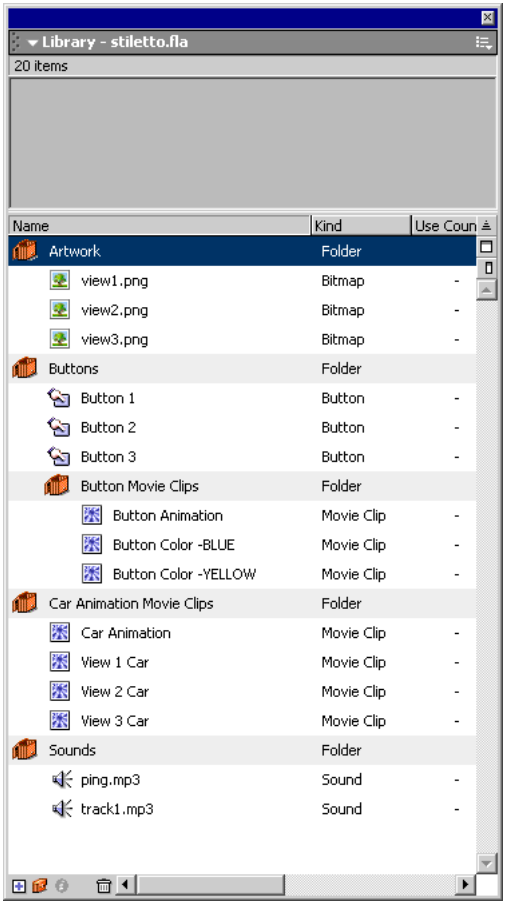
2 Drag the Library panel to enlarge it, if necessary, to view the objects within the library.

3 If the Artwork folder is not expanded, double-click it to view the objects in the folder.

4 Click view1.png to view the image in the preview area at the top of the Library panel.

5 Expand the other folders in the Library panel to view the assets included in the document, such as buttons and movie clips.

6 When you finish viewing the assets, close the Library panel.



Analyze the movie structure with the Movie Explorer

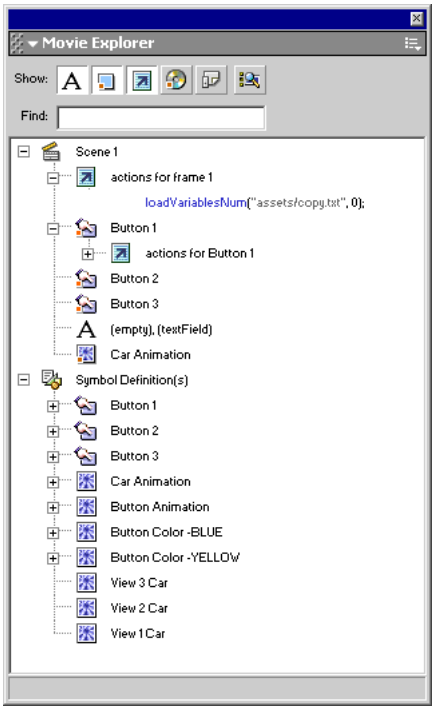
The Movie Explorer helps you arrange, locate, and edit media. With its hierarchical tree structure, the Movie Explorer provides information about the organization and flow of a movie, especially useful when you analyze a movie authored by someone else.

1 If the Movie Explorer is not already open, choose Window > Movie Explorer.

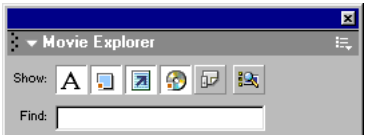
2 If necessary, enlarge the Movie Explorer to view the tree structure within the pane.

The Movie Explorer filtering buttons display or hide information.

3 Click the pop-up menu in the title bar of the Movie Explorer, and verify that Show Movie Elements and Show Symbol Definitions are selected.



4 Deselect the Show Frames and Layers button along the top of the Movie Explorer. Verify that the only filtering buttons selected are Show Text; Show Buttons, Movie Clips, and Graphics; Show ActionScripts; and Show Video, Sounds, and Bitmaps.



5 Examine the list to view some of the assets included in the movie and to see their relationship to other assets.

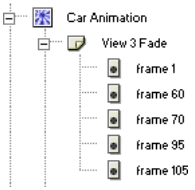
6 To expand an object or category, click the Plus (+) button to the left of the name

7 Select the Show Frames and Layers filtering button. Scroll down to the Symbol Definitions category. With the category expanded, double-click the Car Animation movie clip.

You're now in symbol-editing mode for the movie clip.

8 In the Movie Explorer, with the Car Animation category selected and expanded, expand the View 3 Fade icon, then double-click Frame 60.

In the Timeline for the movie clip, the playhead moves to Frame 60 of the View 3 Fade layer.



To view an item listed in the hierarchical tree, click the corresponding icon. If you click a frame icon, the playhead moves to that frame in the Timeline. If you click an asset, such as a bitmap image, the Property inspector displays the image settings. Double-clicking an icon that represents a symbol opens symbol-editing mode.

9 Close the Movie Explorer. To close the document, choose File > Close.

If you've made changes to the file, do not save them.

Define properties for a new document and create a gradient background

To learn how to create a movie in Flash, let's start from the very first step in the process: opening a new file. Then, by completing this section, you'll learn how to complete the following tasks:

- Open a new file and define document properties
- Create and transform a gradient background

Open a new file

Now you're ready to create your own version of the tutorial movie.

1 Choose File > New.

2 Choose File > Save As.

3 Name the file mystiletto.fla and save it within your Flash MX application folder, in the Tutorials/FlashIntro/My_Stiletto sub folder.

Note: As you complete the tutorial, remember to save your work frequently.

Define document properties

Configuring document properties is a common first step in authoring. You can use the Property inspector and Document Properties dialog box to specify settings that affect the entire movie, such as the frames per second (fps) playback rate, and the Stage size and background color.

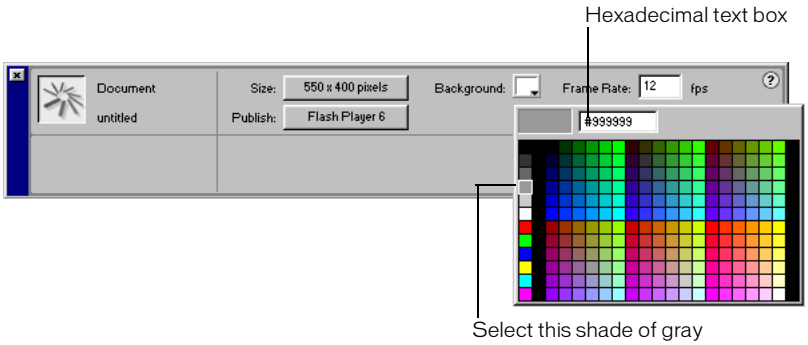
1 If the Property inspector isn't open, choose Window > Properties. In the Property inspector, verify that 12 is the number in the Frame Rate text box.

The movie will play at 12 frames per second, an optimal frame rate for playing animations on the Web.

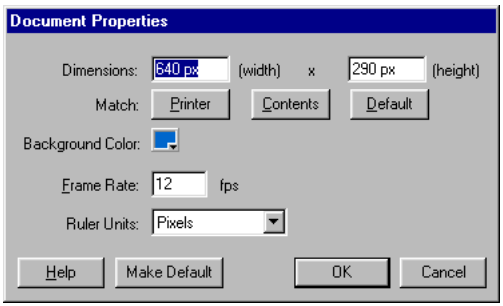
Note: If the Property inspector is not fully expanded, click the white triangle in lower right corner.

2 The Background Color box indicates the color of the Stage. Click the down arrow on the Background Color box, then move the Eyedropper tool over the color swatches to view their hexadecimal values in the Hexadecimal text box.

3 Find and click the gray color swatch with the hexadecimal value of 999999.



4 To resize the Stage, click the Size button, which indicates the size of the Stage. In the Document Properties dialog box, type **640 px** in the first Dimensions text box and **290 px** in the second Dimensions text box. Verify that Pixels is selected in the pop-up menu, and click OK.



The Stage dimensions change to reflect the new settings of 640 x 290 pixels.

Specify grid settings

On the Stage, you can align objects along horizontal and vertical grid lines. Even when the grid is not visible, you can snap objects to it. Now you'll modify the distance between the horizontal and vertical grid lines, and create a grid in alignment with the Stage borders.

1 Choose View > Grid > Edit Grid.

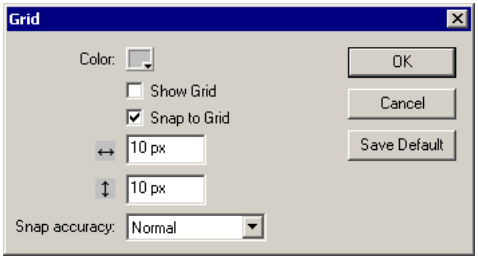
2 In the Grid dialog box, type **10 px** in the grid width text box and **10 px** in the grid height text box.

3 Select Snap to Grid and verify that Show Grid is not selected.

Objects will now snap to the grid, even when the grid is not visible.

4 Verify that Normal is selected for Snap Accuracy, and click OK.

Snap accuracy determines how close an object must be to a grid line before snapping to it.



Create a gradient background

A gradient displays subtle variations of a color, or transitions between two or more colors. In the finished tutorial file, the background is a gradient that blends black and dark blue with a transparent area that allows part of the gray Stage color to be displayed. You achieve this effect using the Color Mixer.

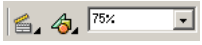
Note: While gradients offer interesting effects in movies, overuse of gradients can adversely affect computer processor speeds and decrease the speed at which an animation plays. When designing a movie, consider both your artistic and performance requirements to determine the best use of gradients.

Draw a rectangle

Earlier in the tutorial, you created a grid that aligned against the Stage and allowed you to snap objects to the grid lines. Now you'll draw a rectangle that snaps to the area of the grid bordering the Stage.

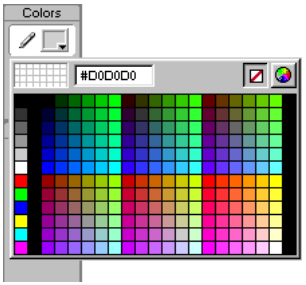
1 In the pop-up menu above the right side of the Stage, enter 75% to view the entire Stage.

This setting indicates your magnified or scaled-down view of the Stage. The setting does not affect the actual size of the Stage in your movie, which you specified in the Document Properties dialog box.



2 In the toolbox, select the Rectangle tool.

3 In the toolbox, click the Stroke Color control. Select No Stroke (the button with the red diagonal line above the color palette).



The selected fill color of the shape is unimportant; you'll soon change the color.

4 Starting from the upper left corner of the Stage, drag to the lower right corner of the Stage to draw a rectangle that covers the Stage.

When you release the pointer, the rectangle snaps to the edges of the Stage, along the hidden grid.

Note: While completing the tutorial, you may find it useful to undo a change you've made. Flash can undo several of your recent changes, depending on the number of undo levels you have set in Preferences. To undo, choose Edit › Undo or press Control+Z (Windows) or Command+Z (Macintosh). Conversely, you can redo what you've undone by choosing Edit › Redo or pressing Control+Y (Windows) or Command+Y (Macintosh).

Specify a color for the gradient

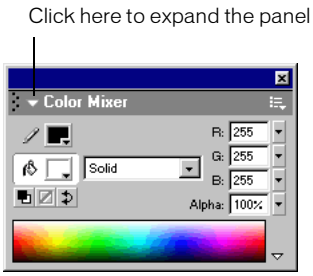
Dark blue is the first color you'll add to your gradient.

1 In the toolbox, select the Arrow tool. On the Stage, click inside the rectangle to select the fill.

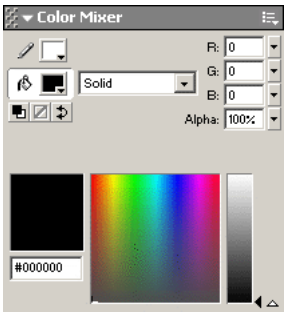
When you drew the rectangle, the Property inspector displayed properties for the Rectangle tool. When you select a shape that has already been created, the Property inspector displays properties for the shape.

2 If the Color Mixer is not open, choose Window > Color Mixer.

3 To expand the Color Mixer, click the white arrow in the panel title bar.

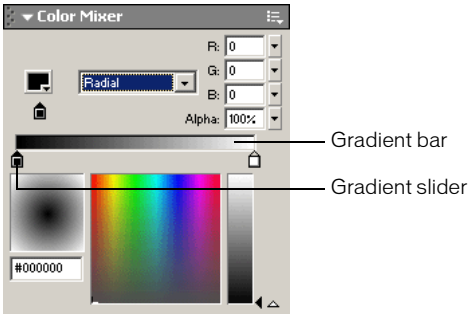


4 If the Color Mixer is not fully expanded, click the arrow in the lower right of the panel.



5 In the Fill Style pop-up menu, select Radial.

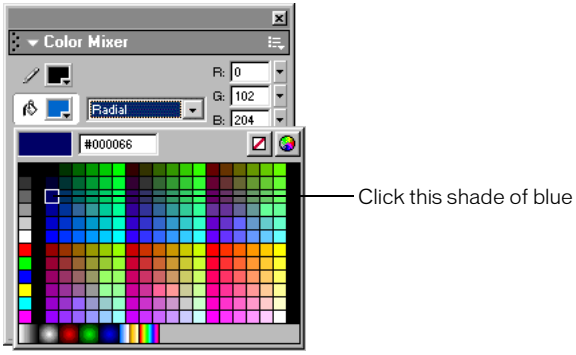
6 Click the gradient slider to the left of the gradient bar to select it.



7 Click the color box in the upper left corner of the window to open the color palette. Use one of the following methods to select dark blue:

- Type **000066** in the hexadecimal value text box and press Enter or Return.

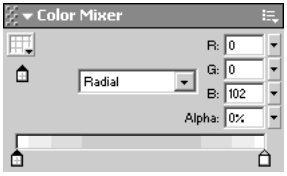
- Move the eyedropper over the color swatches until you find the dark blue with the hexadecimal value of 000066, then click the swatch to select it.



Change the alpha value

In the Color Mixer, the Alpha text box indicates the transparency of the color, with 0% indicating that the color is completely transparent, and 100% indicating the color is completely opaque. You'll specify an alpha value of 0% to create a gradient that includes dark blue and black along with the gray Stage color that shows through the transparent areas of the gradient.

- In the Color Mixer, either type **0** in the Alpha text box and press Enter or Return, or move the Alpha slider to 0.



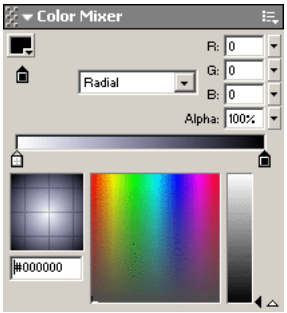
Add a second gradient color

You'll now add black to the gradient.

1 In the Color Mixer, click the gradient slider to the right of the gradient bar to select it.

2 Click the color box to open the color palette and select the black with a hexadecimal value of 000000.

Remember, you can either type the hexadecimal value in the Hexadecimal text box and press Enter or Return, or you can find and select the color swatch with the same hexadecimal value.

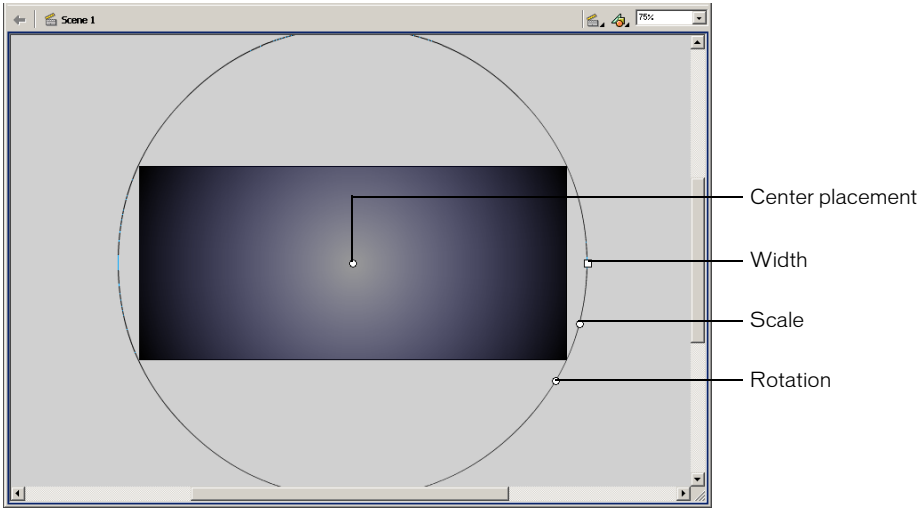


Transform the gradient fill

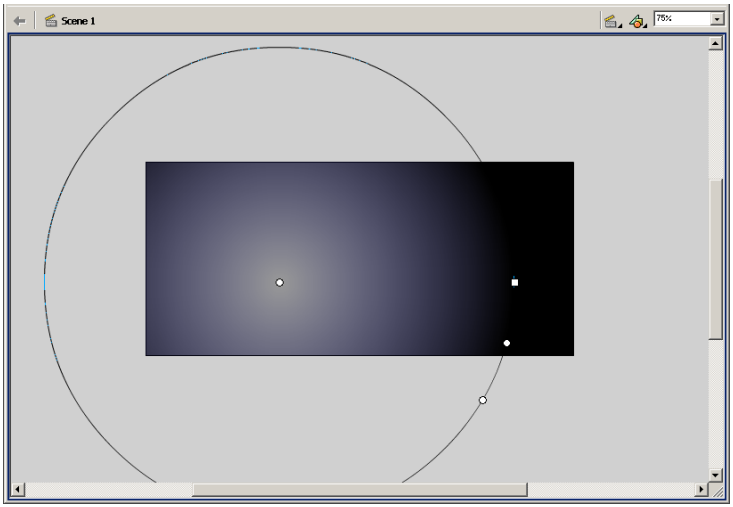
When you transform an object, you rotate, scale, or skew it. You can transform a fill using the Fill Transform tool.

1 In the toolbox, select the Fill Transform tool. On the Stage, click anywhere inside the rectangle.

An ellipse that indicates the shape and location of the gradient appears around the Stage. The ellipse has controls for the location, width, scale, and rotation of the radial gradient.

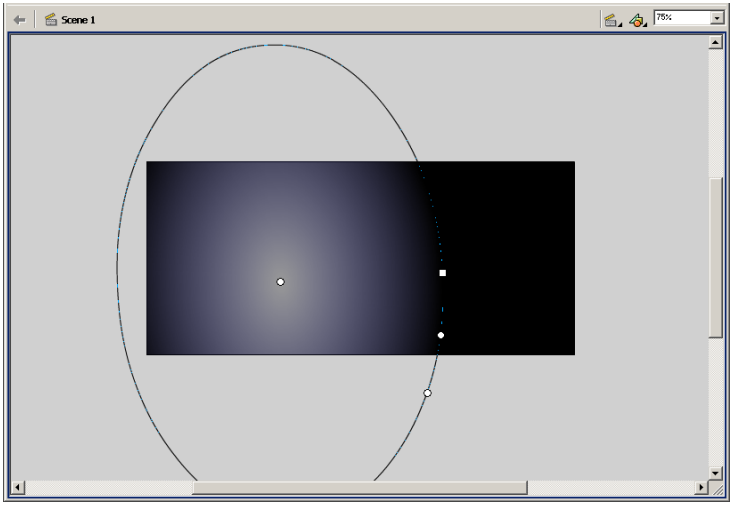


2 Drag the center control to the left of the Stage so that it's approximately 1/3 of the distance from the left edge of the Stage, as shown in the following illustration.

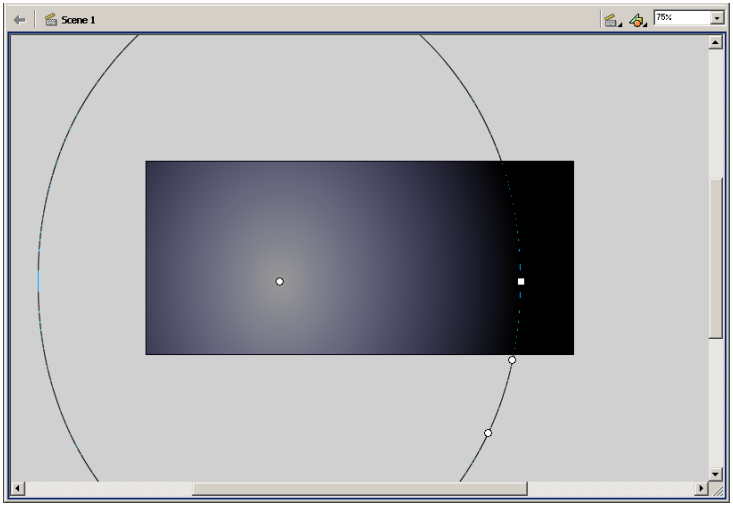


As you change the shape and placement of the ellipse, the shape and placement of the color transitions on the Stage change accordingly.

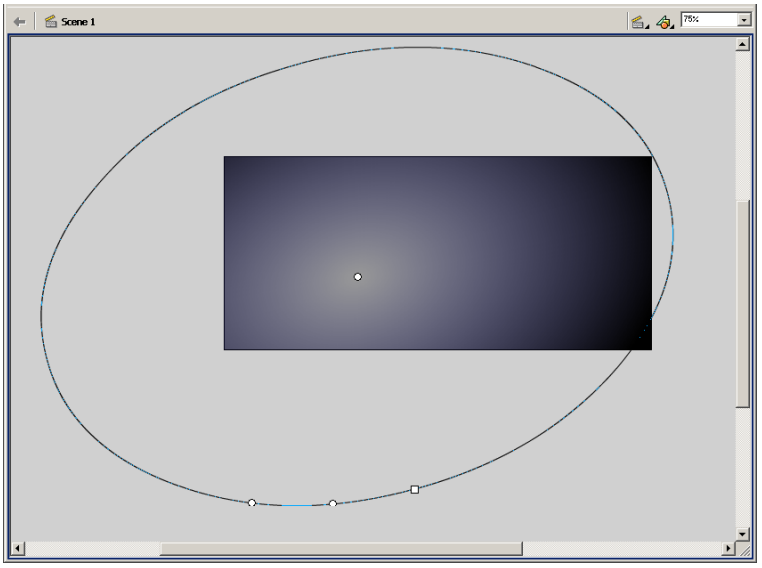
3 Drag the square handle on the ellipse, which controls gradient width, to the left to make the ellipse narrower; the approximate shape of the ellipse is shown in the following illustration.



4 Drag the circular middle handle, which controls the size of the gradient, to the right to make the ellipse larger, as shown in the following illustration.



5 Drag the bottom circular handle, which controls the rotation of the ellipse, toward the left to rotate the ellipse to the approximate angle shown in the following illustration.



Name and lock a layer

The gradient shape appears on Layer 1 in the Timeline, currently the only layer in your document. In preparation for adding and moving additional objects on the Stage, you'll rename and lock the layer. In the next section of the tutorial, you'll create a new layer.

As you learned in the Understanding Layers lesson, by locking the layer, you ensure that you or other users don't make inadvertent changes to objects on the layer.

1 In the Timeline, double-click the Layer 1 name and type **Background** to rename the layer.

2 Click away from the layer name, then click the padlock icon to lock the layer.

Create and mask vector art

When you draw in Flash, you create vector art, which is a mathematical representation of lines, curves, color, and position. Vector art is resolution-independent; you can rescale the art to any size or display it at any resolution without losing clarity. Additionally, vector art downloads faster than comparable bitmap images.

Along with the gradient, the finished file contains background shapes. You'll use the Oval tool to create the shapes.

Specifically, in this section you'll learn how to complete the following tasks:

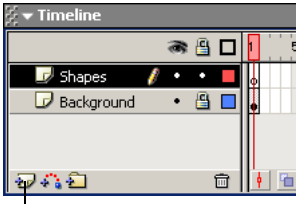
- Add a layer
- Create and "cut out" shapes
- Mask the layer containing the shapes

To complete this section, you can either continue to work on your mystiletto.fla file, or you can browse to your Flash MX application folder and open Tutorials/FlashIntro/stiletto2.fla. If you do use the stiletto2.fla file, save the file with a new name in your My_Stiletto folder to maintain an unadulterated version of the original file.

Add a layer

Rather than create the shapes on the Background layer, you'll add a new layer on which you can draw the shapes.

1 To add a new layer, choose Insert > Layer, or click the Insert Layer button. Name the new layer **Shapes**.



Insert Layer button

2 In the toolbox, select the Oval tool.

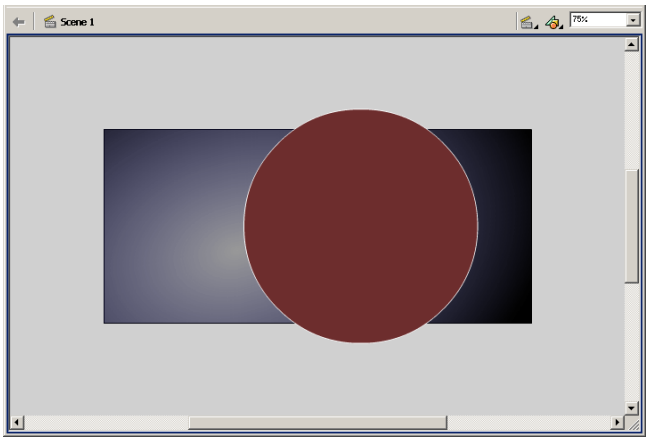
3 In the Property inspector, select Hairline from the Stroke Style pop-up menu. Click the Stroke Color control. In the color palette, select the gray with a hexadecimal value of 999999.

Remember, you can either enter the hexadecimal value in the hexadecimal text box, or find and click the color swatch with the same hexadecimal value.

4 If the Color Mixer isn't open, choose Window > Color Mixer. In the Fill Style pop-up menu, select Solid. Click the Fill Color control to select it. In the R (red) field, type **109**. In the G (green) and B (blue) text boxes, type **45**, then press Enter or Return.

You are specifying the values for the amount of red, green, and blue within a color.

5 In the Timeline, verify that the Shapes layer is selected. On the Stage, constrain the oval to a circle by pressing Shift as you drag, to draw a circle that extends from the canvass area above the Stage to the canvass below the Stage.

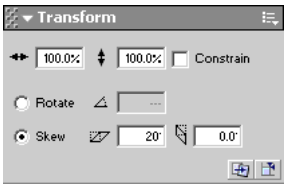


Note: If you make a mistake, choose Edit › Undo and try again.

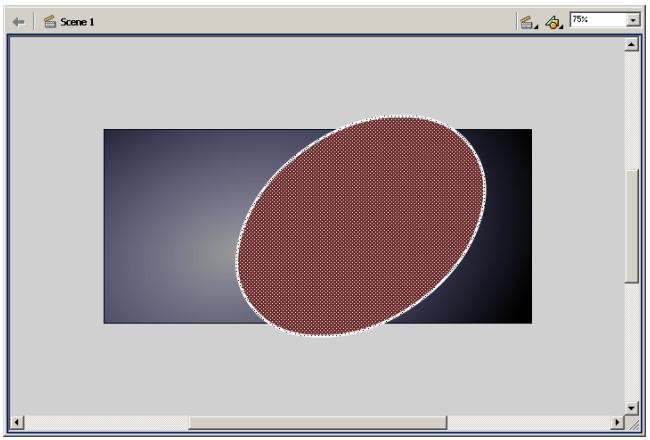
6 Select the Arrow tool in the toolbox, then double-click the circle on the Stage to select both the fill and the stroke.

7 If the Transform panel isn't open, choose Window > Transform.

8 To expand the Transform panel, click the white arrow in the upper left corner. Select Skew and type **20.0** in the Skew Horizontally text box, then press Enter or Return.



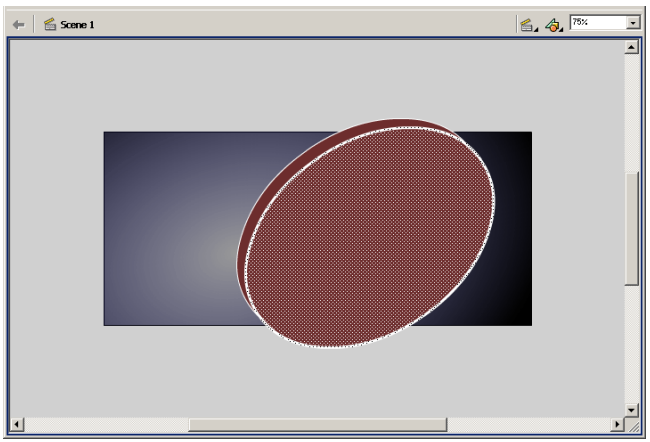
The circle that you drew changes into a skewed oval.



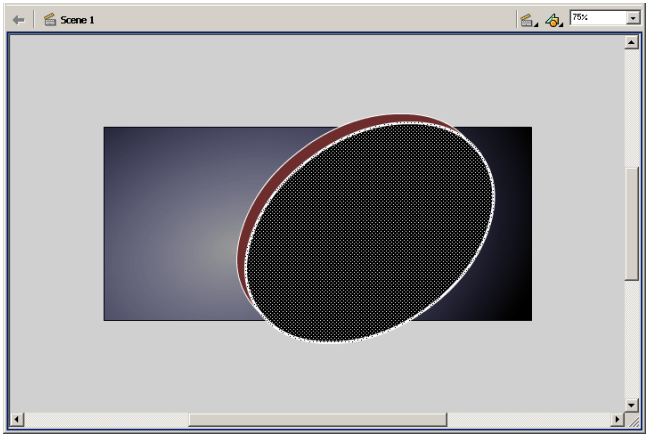
Create and transform a duplicate shape

You'll now create and transform a duplicate oval.

1 With the oval fill and stroke still selected on the Stage, choose Edit > Duplicate.



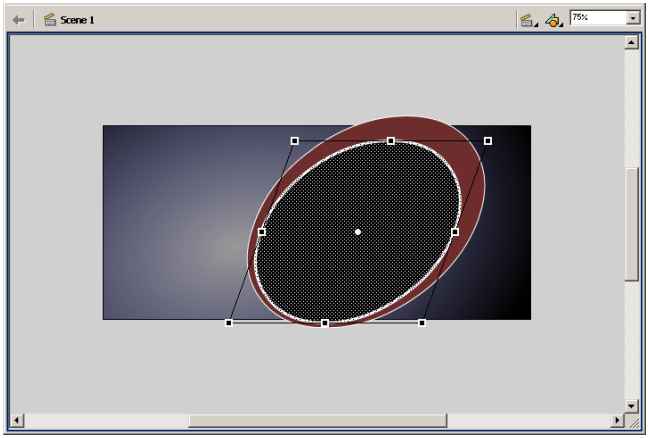
2 In the Property inspector, use the Fill Color control to select black.



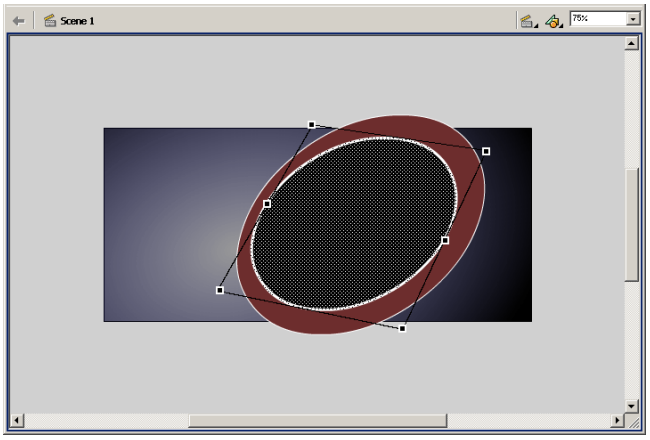
3 In the toolbox, select the Free Transform tool.

A guide with handles appears around the duplicate oval.

4 Move the pointer over a corner handle until a diagonal indicator with arrows at both ends appears. Drag the corner handle inward to make the oval smaller. Verify that the stroke of the inside oval is not touching the stroke of the outside oval.



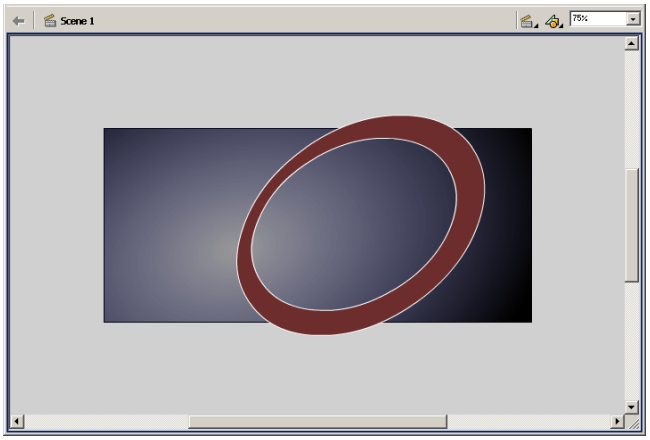
5 Move your pointer above the corner handle until the pointer changes into a circular rotation indicator. Drag the rotation indicator to the left to rotate the oval to the approximate position shown in the following illustration.



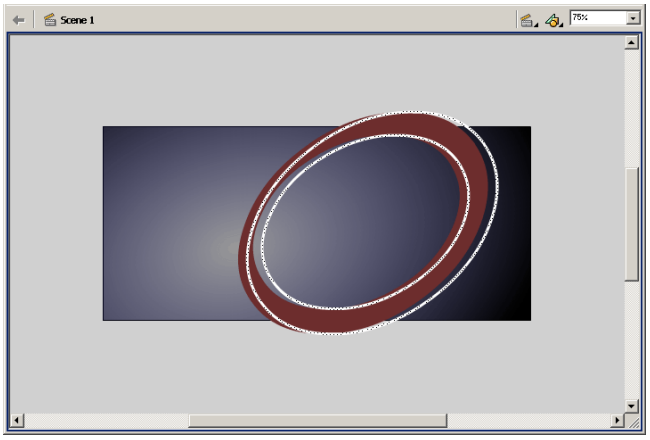
Create a "cut out" with the duplicate

When you create one shape on top of another on the same layer, and the two shapes are different colors as well as ungrouped, the shape on top "cuts out" the area of the shape underneath. You'll delete the duplicate oval to view the cut out effect.

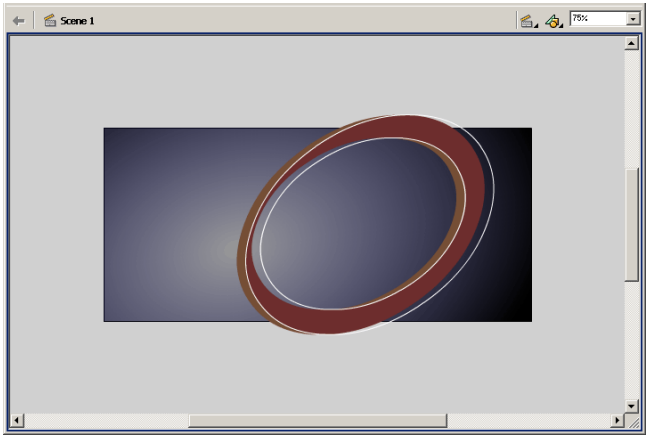
1 With the Free Transform tool guide around the duplicate oval, click anywhere on the Stage or canvass away from the shapes, then click the fill of the inner oval. Press Delete on your keyboard to delete the fill.



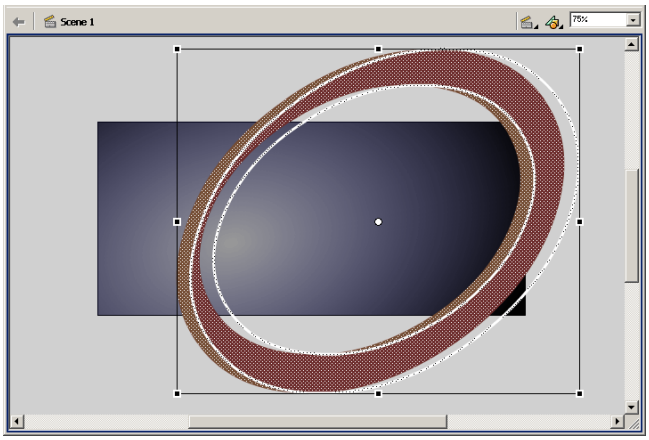
2 The oval now has an outer and an inner stroke. With the Arrow tool, click the outermost stroke on the oval to select it, then Shift-click to select the innermost stroke as well. Drag the strokes slightly to the right of the fill, as shown in the following illustration.



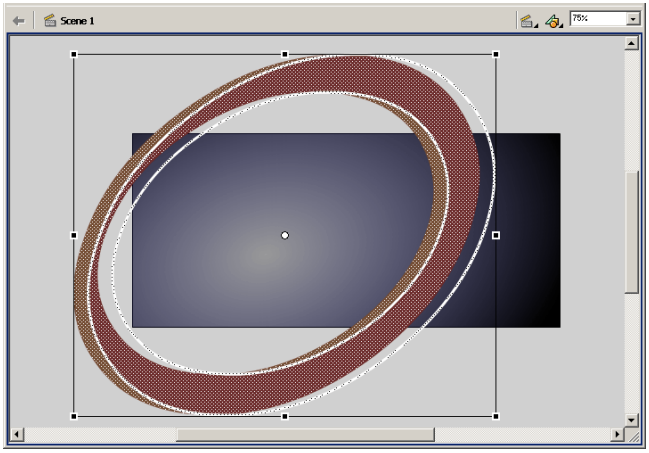
3 Each area of fill color within the area bisected by the stroke now represents a discrete segment that you can color individually. Select an area of the shape fill, and use the Color Mixer to change the fill color to a brown shade with an R (red) value of 117, a G (green) value of 78, and a B (blue) value of 53. If desired, repeat this step to change another fill area to the lighter shade of brown, as shown in the following illustration:



4 Select the Free Transform tool. Drag around the oval background shapes to select all the shapes, then drag the right corner handle of the guide to enlarge the shapes, as shown in the following illustration.



5 Drag the shapes on the Stage so that part of the curve shows on the upper left corner and right side of the Stage.



Note: As you complete the tutorial, remember to save your file frequently.

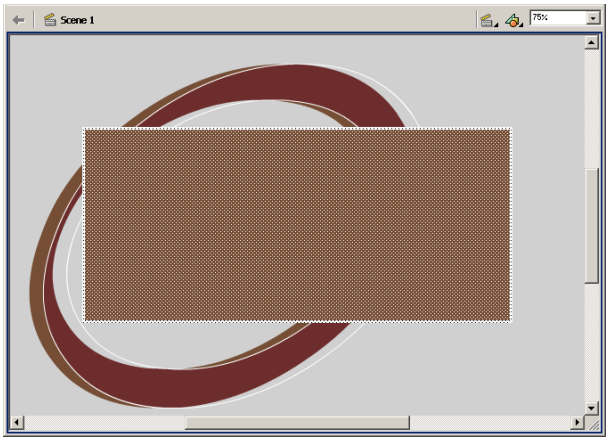
Create a mask

The art that you created on the Shapes layer extends beyond the Stage, well into the canvass area. Although the area on the canvass won't appear in your published movie, the art beyond the Stage can be distracting in the authoring environment. While you can erase the part of the shapes that extend into the canvass, a better solution is to apply a mask over the Stage so that only the area under the mask—the entire Stage, in this case—remains visible. This way, if you'd like to return to the shapes to modify them, they will be intact.

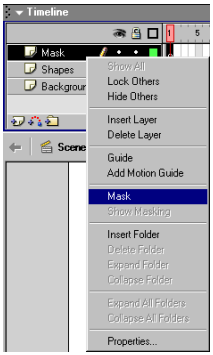
1 With the Shapes layer selected, add a new layer to the Timeline and name it **Mask**.

2 In the toolbox, select the Rectangle tool and draw a rectangle that extends from the upper left corner of the Stage to the lower right corner.

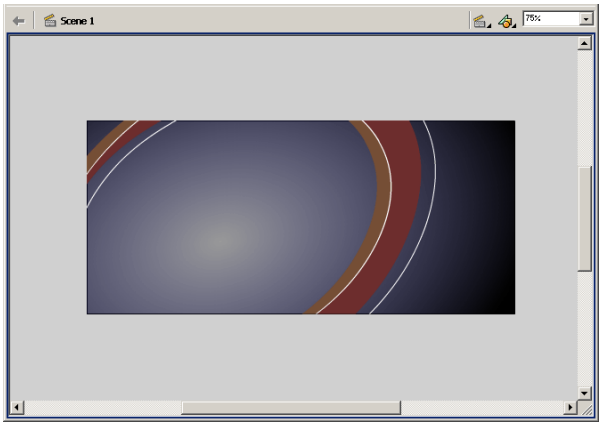
This rectangle is the shape of your mask. Anything under the rectangle will be visible.



3 Right-click (Windows) or Control-click (Macintosh) the Mask layer name in the Timeline and choose Mask from the context menu.



The layer is converted to a mask layer, indicated by a down arrow icon. The layer immediately below it is linked to the mask layer, and its contents show throughout the filled area on the mask. The masked layer name is indented, and its icon changes to a right-pointing arrow. The art on the canvass is no longer visible on the Stage.



Mask layers must be locked for the Mask effect to show. To edit the shapes, you can unlock the Mask and Background Shapes layers. When you finish editing the art, lock the layers again to invoke masking.

4 Save your file.

Tween bitmap effects within a movie clip

In addition to creating vector art in Flash, you can import bitmap images, which use pixels to display graphics, into your Flash movie and apply various color effects. In this section, you'll complete the following tasks:

- Import bitmap images
- Modify bitmap compression
- Create and edit a movie clip symbol
- Tween bitmap effects to fade views of the car in and out

To complete this section, you can either continue to work on your mystiletto.fla file, or you can browse to your Flash MX application folder and open Tutorials/FlashIntro/stiletto3.fla. If you do use the stiletto3.fla file, save the file with a new name in your My_Stiletto folder to maintain an unadulterated version of the original file.

Import images into the library

When you import a file into Flash, you can import it directly into the library.

1 On the Timeline, add a new layer and name it **Images**.

2 Choose File > Import to Library.

When you select Import to Library rather than Import, the images must be placed on the Stage before they will appear.

3 Browse to your Tutorials/FlashIntro/Assets folder within your Flash MX application folder and select view1.png, then Shift-click to add view2.png and view3.png to the selection. Click Open.

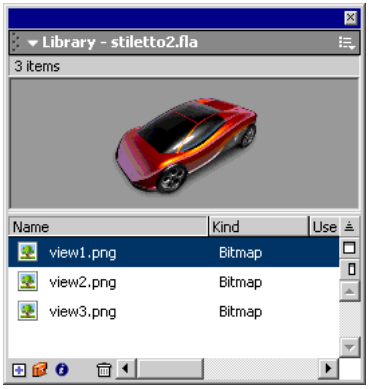
4 In the Fireworks PNG Import Settings dialog box, click Import as a Single Flattened Bitmap, then click OK.

The three images are now in the library.

Modify bitmap compression

When you import an image, you can check and modify settings that compress the image. While compressing images reduces the file size of your movie, compression can affect image quality; the goal is to strike a balance between compression settings and image quality.

1 If the Library panel isn't open, choose Window > Library. Enlarge the window, if necessary, to see the three files you imported.



2 Double-click the view1.png file.

JPEG compression is the default selection.

3 In the Compression pop-up menu, select Lossless (PNG/GIF) for higher image quality.

4 To test how the image appears with the new setting, click Test. If necessary, drag the car into view in the preview window. When you finish previewing the image, click OK.

5 Return to the Library panel. Double-click view2.png and repeat step 3, then click OK.

6 In the Library panel, double-click view3.png and specify Lossless (PNG/GIF), then click OK.

Create a movie clip symbol

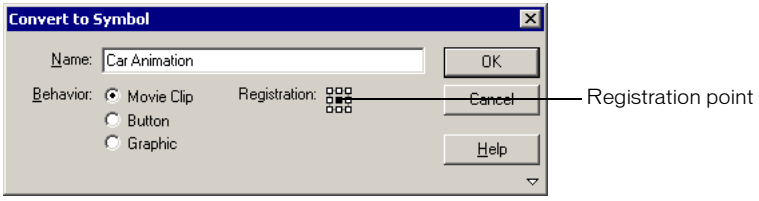
In the finished file, three views of the electric car fade in and out in the opening scene. This effect is achieved by creating a movie clip symbol that has a Timeline independent of the main Timeline. Next, you tween the alpha transparency between three views of the car to create a fade in/fade out effect. To begin to create the effect, you'll create the movie clip.

1 With the Images layer still selected in the Timeline, drag the view1.png object from the Library panel to the Stage, placing the car within the area where the gradient background is lightest.

2 Choose Insert > Convert to Symbol, or press F8.

3 In the Convert to Symbol dialog box, name the symbol **Car Animation**. Verify that Movie Clip is selected and that the center square is selected in the Registration indicator, and click OK.

Bitmaps, like other Flash objects, have registration points used for positioning and transformation. When you align the three views of the car within the movie clip, all three views should align relative to a center registration point.



Edit a symbol

To view the Timeline of the movie clip, you must be in symbol-editing mode. You can open symbol-editing mode by double-clicking the symbol either on the Stage or in the Library panel.

1 On the Stage, double-click the car to open symbol-editing mode.

The name of the symbol appears above the canvass area, along with a Scene 1 link that returns you to the main movie.

In symbol-editing mode, you're now viewing the Timeline for the movie clip rather than the Timeline for the main movie.

2 Rename Layer 1 **View 1 Fade**.

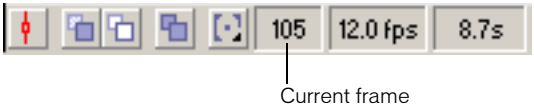
3 The car that you see on the Stage is a bitmap image, not a symbol, within the Car Animation symbol. Make the car a symbol by selecting it on the Stage and pressing F8.

4 In the Convert to Symbol dialog box, name the symbol **View 1 Car**, then verify that Movie Clip is selected.

5 Verify that the center square is selected in the Registration indicator and click OK.

6 Scroll horizontally across the Timeline until you get to Frame 105. Select the frame and choose Insert > Keyframe, or press F6 to add a keyframe.

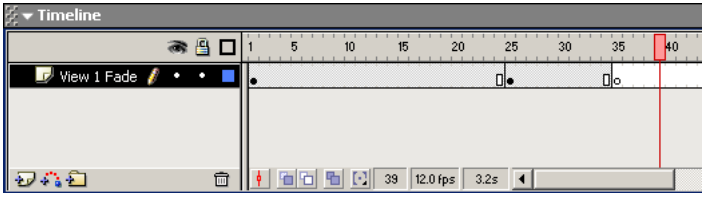
The Current Frame indicator displays the selected frame.



7 Add keyframes to Frames 25 and 35.

8 Add a keyframe to Frame 34, then click anywhere on the layer between Frames 36 and 104, and press Delete on your keyboard.

An empty keyframe appears in Frame 35, and the car does not appear on the Stage from frame 35 on.



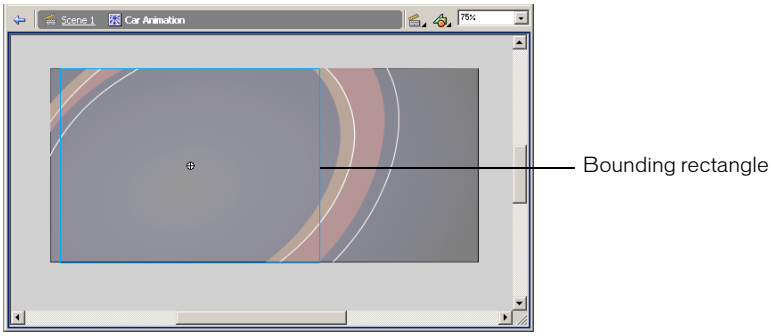
Note: If you make an error in adding keyframes and want to delete them, select one or more frames and right-click (Windows) or Control-click (Macintosh), then choose Clear Keyframe from the context menu.

Tween bitmap effects

Creating a bitmap effect tween is similar to creating a straight motion tween: you specify settings for beginning and ending keyframes, then specify tweening for those frames and the frames in between. Flash creates the transitional animation from the first keyframe in the animation to the last.

1 In the Car Animation Timeline, select Frame 34, then click the View 1 Car on the Stage so that the Property inspector appears displaying movie clip properties.

2 In the Color pop-up menu of the Property inspector, select Alpha. In the Alpha Amount pop-up menu, either type **0%** in the text box and press Enter or Return, or use the pop-up slider to select 0%.



3 In the Timeline, select any frame between Frames 25 and 34. In the Property inspector, select Motion from the Tween pop-up menu.

An arrow with a solid line spans the tweened keyframes. A dashed line between keyframes indicates the tweening is not implemented correctly, which often occurs when a beginning or ending keyframe is missing.

Fade in the second car

As the View 1 Car fades out, another view of the car should fade in.

1 Add a new layer to the Car Animation Timeline and name it **View 2 Fade**.

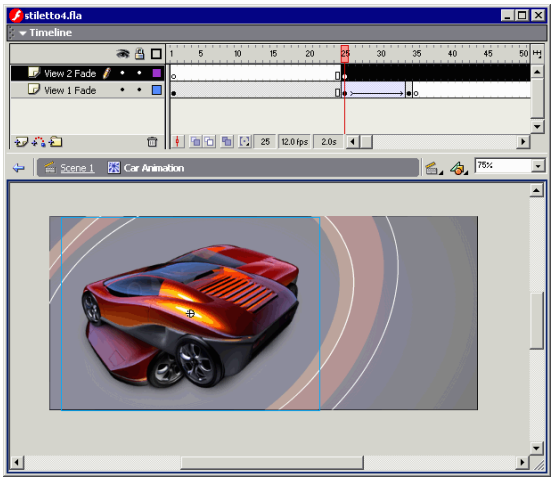
2 On the View 2 Fade layer, add a keyframe to Frame 25.

3 With the playhead still on Frame 25, drag view2.png from the Library panel to the Stage.

4 If the Info panel isn't open, choose Window > Info. Verify that the center square is selected in the Registration indicator, then type **0** in the X coordinate text box and type **0** in the Y coordinate text box. Press Enter or Return.

The Property inspector also has X and Y text boxes; however, those coordinates are relative to a registration point in the upper left corner of the movie clip.

5 Select the view2.png on the Stage and press F8 to make it a symbol. In the Convert to Symbol dialog box, name the symbol **View 2 Car**. Verify that Movie Clip is selected, and click OK.



6 In the movie clip Property inspector, select Alpha in the Color pop-up menu and type **0%** in the Alpha Amount text box.

7 Add a keyframe to Frame 35 of the View 2 Fade layer.

8 On the Stage, click inside the bounding rectangle of the transparent car. In the movie clip Property inspector, enter **100%** in the Alpha Amount text box.

9 On the View 2 Fade layer, select any frame between Frame 25 and Frame 34. In the Property inspector, select Motion from the Tween pop-up menu.

Fade out the second car

Now you'll create the animation that fades out the second car.

1 On the View 2 Fade layer, add a keyframe to Frame 60.

2 On the View 2 Fade layer, add a keyframe to Frame 70, and another keyframe to Frame 69.

3 Select the keyframe in Frame 69 of the View 2 Fade layer. Select the View 2 Car on the Stage and use the Property inspector to select an alpha transparency of 0%.

4 On the View 2 Fade layer, select any frame between Frames 60 and 68. In the Property inspector, select Motion from the Tween pop-up menu.

5 Click any frame on the View 2 Fade layer between Frames 71 and 105, and press Delete.

Note: As you complete the tutorial, remember to save your work frequently.

Fade in the third car

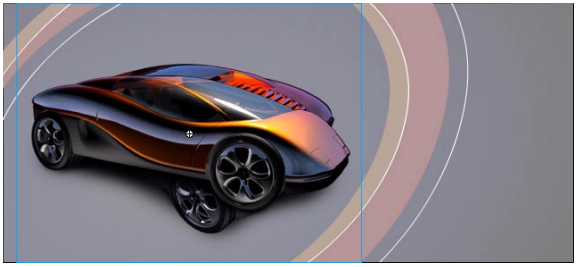
As the second car fades out, the third car fades in. You'll create that animation now.

1 With the View 2 Fade layer selected, add a new layer to Timeline and name it **View 3 Fade**.

2 On the View 3 Fade layer, add a keyframe to Frame 60.

3 With Frame 60 still selected, drag the view3.png from the Library panel to the Stage. Use the Info panel (choose Window > Info if the panel is closed) to specify **0** for both the X and Y coordinates, and to verify the registration point is centered, as you did for the view2.png.

4 Select view3.png on the Stage and press F8 to make it a symbol. In the Convert to Symbol dialog box, name the symbol **View 3 Car**. Verify that Movie Clip is selected, and click OK.



5 In the Property inspector, select Alpha in the Color pop-up menu and type **0%** in the Alpha Amount text box.

6 Add a keyframe to Frame 70 of the View 3 Fade layer.

7 On the Stage, select inside the bounding rectangle of the View 3 Car. In the Property inspector, enter **100%** in the Alpha Amount text box.

8 On the View 3 Fade layer, select any frame between Frames 60 and 69. In the Property inspector, select Motion from the Tween pop-up menu.

Fade out the third car

You'll now create the animation that fades out the third car.

1 On the View 3 Fade layer, add a keyframe to Frames 95 and 105.

2 With Frame 105 selected in the View 3 Fade layer, select the View 3 Car on the Stage and use the Property inspector to select an alpha transparency of 0%.

3 On the View 3 Fade layer, select any frame between Frames 95 and 104. In the Property inspector, select Motion from the Tween pop-up menu.

Fade in the first car

As the third car fades out, the first car must fade in to complete the animation.

1 On the View 1 Fade layer, add a keyframe to Frame 95.

2 With Frame 95 still selected, drag the View 1 Car movie clip (not view1.png) from the Library panel to the Stage.

3 In the Info panel, type **0** in the X coordinate text box and type **0** in the Y coordinate text box. Press Enter or Return.

4 In the Property inspector, select Alpha in the Color pop-up menu and enter 0% in the Alpha Amount text box.

5 Select Frame 104 of the View 1 Fade layer.

6 Click inside the bounding rectangle of the View 1 Car movie clip on the Stage. In the Property inspector, enter 100% in the Alpha Amount text box.

7 On the View 1 Fade layer, select any frame between Frames 95and 104. In the Property inspector, select Motion from the Tween pop-up menu.

Note: As you complete the tutorial, remember to save your work frequently.

Test the movie

At any point during authoring, you can test how your movie will play as a SWF file.

1 Save your movie and choose Control > Test Movie.

Flash exports a SWF copy of your movie.

In the SWF movie, the animation automatically plays in a continuous loop.

2 When you finish viewing the movie, close the SWF file by clicking its close box. In your Flash document, choose Edit > Edit Document or click Scene 1 to return to the main Timeline.

Load dynamic text at runtime

In the lesson Adding and Editing Text, you practiced typing text directly on the Stage. You can also design your movie to include text from external files. One of the easiest ways to accomplish this is to use the `loadVariables` action to load text from a text file into a dynamic text field at runtime. In the FLA file, you can specify text attributes, such as font style, size, and color, for the dynamic text field. An advantage of keeping text in external files is that anyone who wants to modify the text can work with the text file rather than the FLA file.

In this section, you'll learn how to accomplish the following tasks:

- Import and align a logo
- Create a dynamic text field
- Use the Property inspector to assign a text variable name
- Use the `loadVariables` action to load text from an external file

To complete this section, you can either continue to work on your mystiletto.fla file, or you can browse to your Flash MX application folder and open Tutorials/FlashIntro/stiletto4.fla. If you do use the stiletto4.fla file, save the file with a new name in your My_Stiletto folder to maintain an unadulterated version of the original file.

Import the logo

Before creating the dynamic text field, you'll import the logo, a Macromedia FreeHand file for which Flash automatically adds a layer on the Timeline.

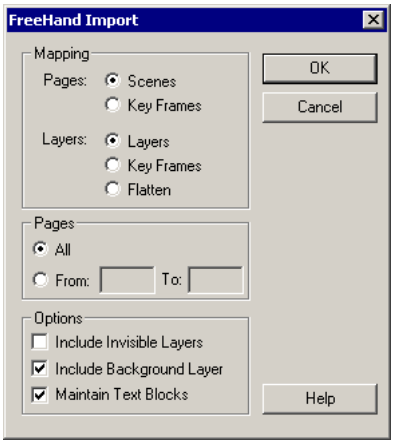
1 In the Timeline, select the Images layer and add a new layer above it. Name the new layer **Copy**.

2 With the Copy layer selected, choose File > Import.

Earlier in the tutorial, you imported objects into the library. Now you'll import the logo so that it appears on the Stage.

3 Browse within your Flash MX application folder to the Tutorials/FlashIntro/Assets folder and click logo.fh10, then click Open.

4 In the FreeHand Import dialog box, verify that Scenes, Layers, and All are selected. Also verify that Include Background Layer and Maintain Text Blocks is selected, and then click OK.



5 In the Timeline, Flash created a layer named Logo. Drag the Logo name to move the layer below the Copy layer.

6 You can specify Stage coordinates for the logo. In the Property inspector, with the logo selected, type **10** in the X text box and **20** in the Y text box. Then press Enter or Return.

7 In the Timeline, lock the Logo layer.

Create a dynamic text field

Now you'll create a dynamic text field. Instead of typing text into the field, you'll specify the variable text that loads into the field at runtime.

1 In the Timeline, select the Copy layer. In the toolbox, select the Text tool. In the Property inspector, select Dynamic Text from the Text Type pop-up menu.

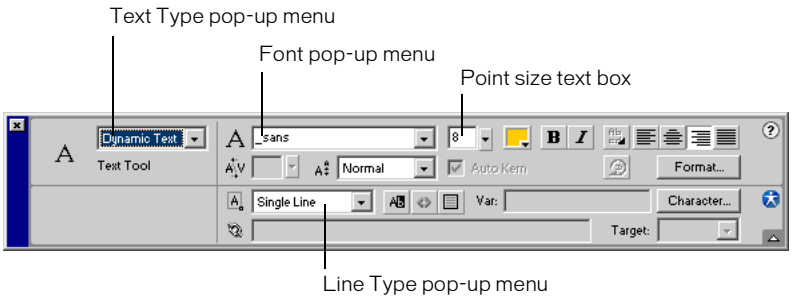
2 In the Font pop-up menu, select _sans.

_sans is a device font appropriate for small text that appears on multiple computer platforms. For more information about device fonts, see "Using device fonts (horizontal text only)," under Help > Using Flash.

3 In the Point Size text box, type **12**.

4 Click the Text (fill) Color box and select yellow, with a hexadecimal value of FFCC00.

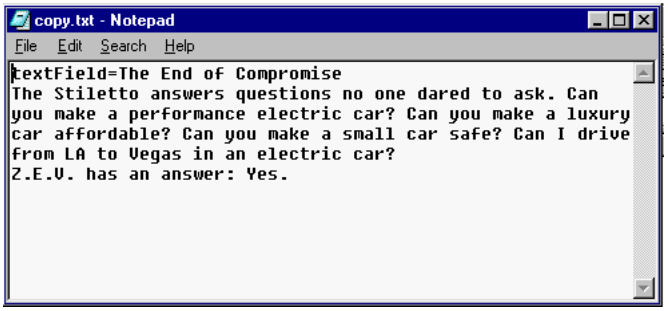
5 In the Line Type pop-up menu, select Multiline, which is for multiple lines of text that will wrap.



6 On the Stage, click below the logo. Drag the pointer to create a text field the width of the logo text and the approximate depth of the vertical line that's grouped with the logo, as shown in the following illustration.

7 In the Property inspector, type **textField** in the Var text box.

The text file that will load into the dynamic text field, as seen in the following illustration, includes text that names the variable: `textField=`. When you enter this name in the Var text box, you are naming the variable that the movie should load.



Use the loadVariables action to load text

The `loadVariables` action includes a parameter to specify the path to the variable text. The text is in a file named copy.txt, within your Tutorials/FlashIntro/Assets folder.

1 In the Timeline, add a new layer and name it **Actions**. If necessary, drag the Actions layer to the top of the Timeline.

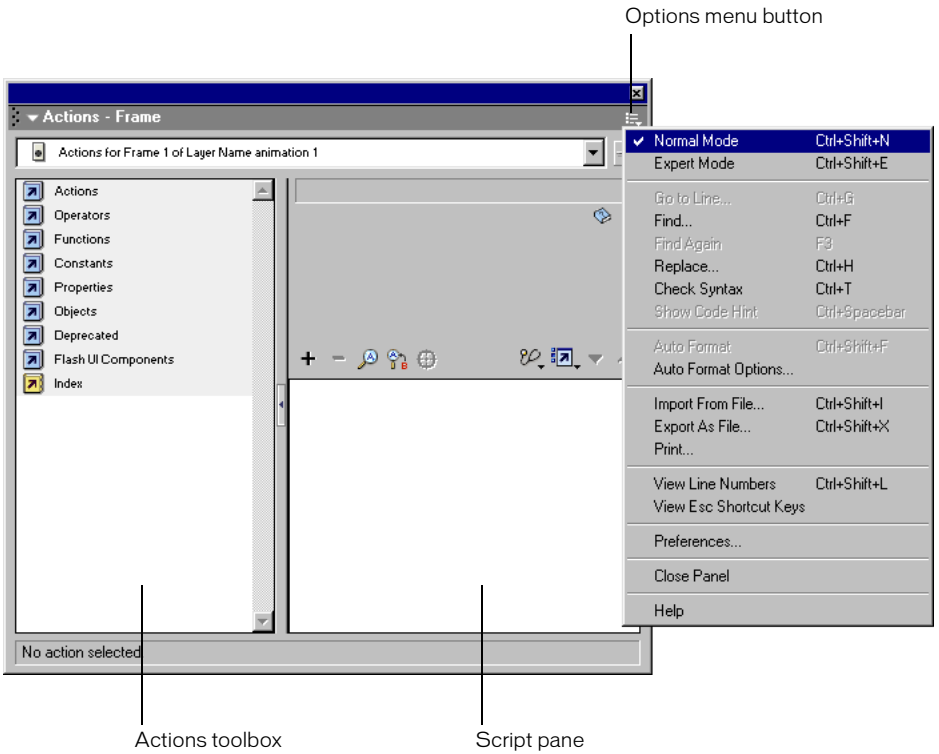
A good practice is to keep actions on the top layer in a Timeline.

2 If the Actions panel is not open, choose Window > Actions. Enlarge the Actions panel, if necessary, by clicking the white arrow in the title bar to expand the window, and by dragging the lower right corner of the panel to view the Actions toolbox and Script pane.

The type of Actions panel that is displayed depends on the object to which you're adding an action. If you've selected a frame, for example, the Actions panel displays actions for frames. If you've selected a button, the Actions panel displays actions for buttons.

3 Click the triangle in the upper right corner of the panel title bar to display the pop-up menu. Verify that normal mode, rather than expert mode, is selected.

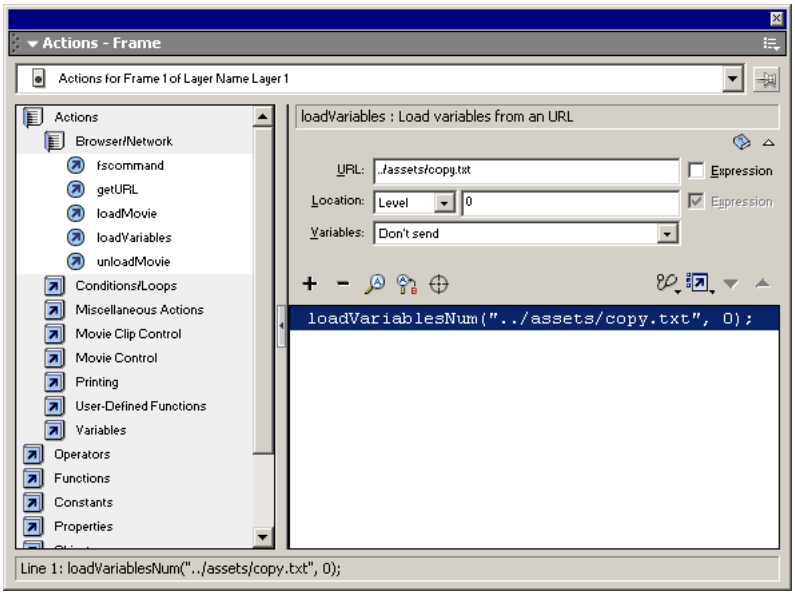
Expert mode offers features useful to those experienced with ActionScript. In normal mode, parameter fields and controls guide you in creating ActionScript.



4 In the Actions toolbox, choose Actions > Browser/Network, then double-click `loadVariables`.

The ActionScript is added to the Script pane. Parameters for the action appear above the Script pane.

5 In the URL text box, type the path to the text file: **../assets/copy.txt**.



Test your movie

- Save your movie, then choose Control > Test movie.

  You can also press Control+Enter (Windows) or Command+Return (Macintosh).

Add animation and navigation to buttons

When you specify that a new symbol is a button, Flash creates the Timeline for the button states. In the Creating Buttons lesson, you learned how to change the fill color of a shape within a button state. In this section, you'll learn more about modifying buttons, including adding animation to a button.

Specifically, in this section you'll learn how to complete the following tasks:

- Import a library from another FLA file
- Align buttons
- Add animation to a button state
- Add navigation to a button to link to a Web site
- Use the Enable Simple Buttons feature
- Add button navigation

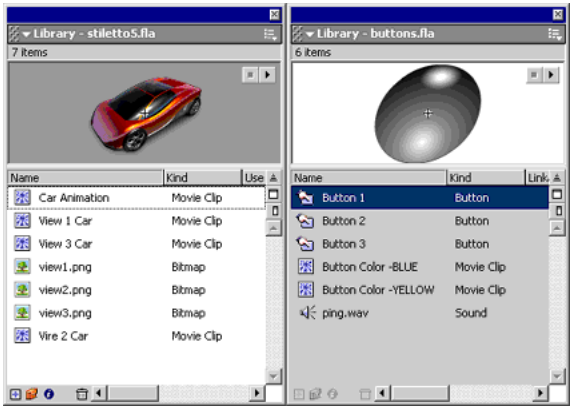
To complete this section, you can either continue to work on your mystiletto.fla file, or you can browse to your Flash MX application folder and open Tutorials/FlashIntro/stiletto5.fla. If you do use the stiletto5.fla file, save the file with a new name in your My_Stiletto folder to maintain an unadulterated version of the original file.

Import the library of another FLA file

The buttons that you'll use in your movie reside in the library of another FLA file. To use the buttons, you open the FLA file containing the buttons as a library.

1 With your Library panel open, choose File > Open as Library. Browse within your Flash MX application folder to the Tutorials/FlashIntro/Assets folder and double-click buttons.fla.

The library for the buttons.fla file appears in addition to the library for your document.



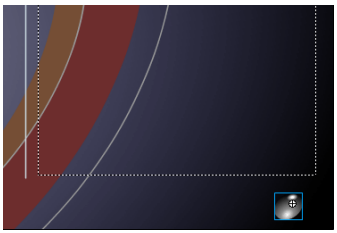
Align buttons

You can align the three buttons along horizontal and vertical axes using the Align panel.

1 In the Timeline, with the Copy layer selected, add a new layer and name it **Buttons**. Lock all the layers except the Buttons layer.

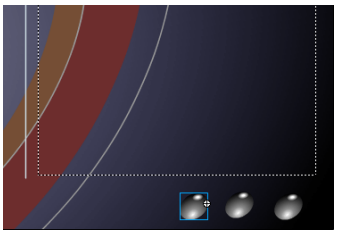
2 In the Stage View pop-up menu, on the right above the Stage, enter **150%** to enlarge your view of the Stage. Then scroll to the lower right side of the Stage.

3 Drag Button 1 from the buttons.fla Library panel and place it under the lower right corner of the dynamic text field.



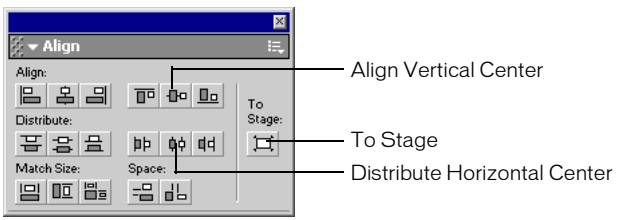
When you drag a button from the buttons.fla Library panel, the button becomes part of the library for your document.

4 Drag Button 2 and Button 3 from the buttons.fla Library panel, placing them to the left of Button 1. Use the approximate spacing shown in the following illustration:



5 With the Arrow tool, drag to select all three buttons.

6 To open the Align panel, choose Window > Align. Verify that To Stage is not selected. You do not want to align the buttons relative to the Stage. Click Align Vertical Center, and then click Distribute Horizontal Center.



The buttons align on the Stage.

Enable simple buttons

When the Enable simple buttons feature is active, you can hear the sounds included with Buttons 2 and 3, and you can view the colors used for the button states. More complex button design, such as animation, does not play.

1. Choose Control > Enable Simple Buttons, then roll over and click each button.

   The right button, Button 1, is not finished. You will modify that button next.

2. When you finish testing the buttons, choose Control > Enable Simple Buttons to deselect that feature.

Modify a button state

You'll create a movie clip within the Over state of Button 1, then create a shape tween in the movie clip. The shape tween creates an effect that changes the color from gray to red.

1. On the Stage, double-click the right button, Button 1, to open symbol-editing mode.
2. In the Button 1 Timeline, hide all layers except the Color layer. In the Color layer, select the Over keyframe.
3. On the Stage, select the black oval shape for the right button. Press F8 to make the oval a symbol. In the Convert to Symbol dialog box, name the symbol **Button Animation**. Select Movie Clip, and click OK.
4. On the Stage, double-click the Button Animation symbol to open symbol-editing mode.
5. Rename Layer 1 **Color Change**, and add a keyframe to Frame 15.
6. With the playhead still on Frame 15, select the button shape on Stage and choose a bright shade of red from the Fill Color pop-up menu in the toolbox.
7. In the Timeline, click any frame between Frames 1 and 13. In the Property inspector, select Shape from the Tween pop-up menu.

   Drag the playhead from frames 1 to 15 to see the color change.

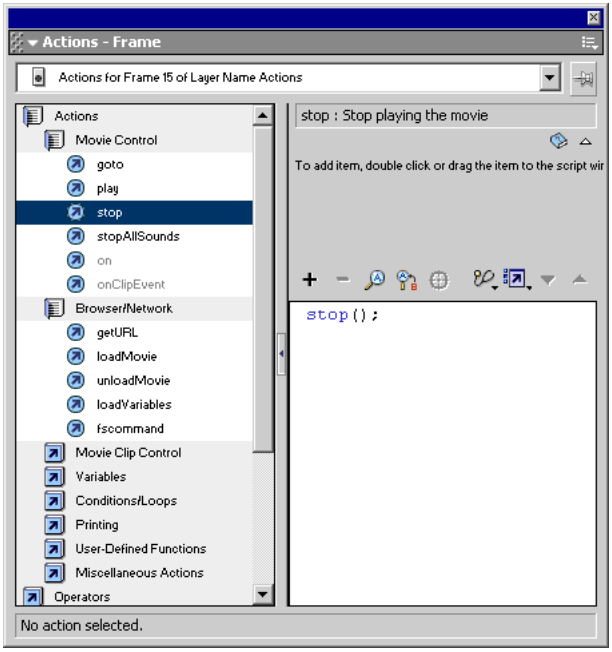
Add actions to buttons

When the user clicks the button and the tweened animation plays, you want the playhead to move to the end of the Button Animation Timeline, then stop. You use ActionScript, the Flash scripting language, to control playhead movement in a Timeline.

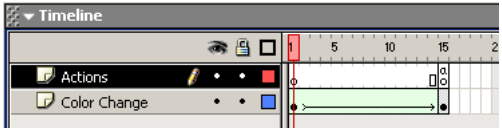
1. Add a new layer to the Button Animation Timeline and name it **Actions**.
2. On the Actions layer, add a keyframe to Frame 15.
3. If the Actions panel is not open, choose Window > Actions. Enlarge the panel, if necessary, to view both the Actions toolbox and the Script pane.

4 With Frame 15 of the Actions layer selected, go to the Actions > Movie Control category of the Actions toolbox and double-click `stop`.

The `stop` action lets you specify that the playhead will stop when it reaches Frame 15.



In the Button Animation Timeline, Frame 15 of the Actions layer now displays a small *a,* which indicates that an action is attached to that frame.



Note: As you complete the tutorial, remember to save your work frequently.

Add button navigation

You use the `getURL` action to add navigation to a button that opens a Web site.

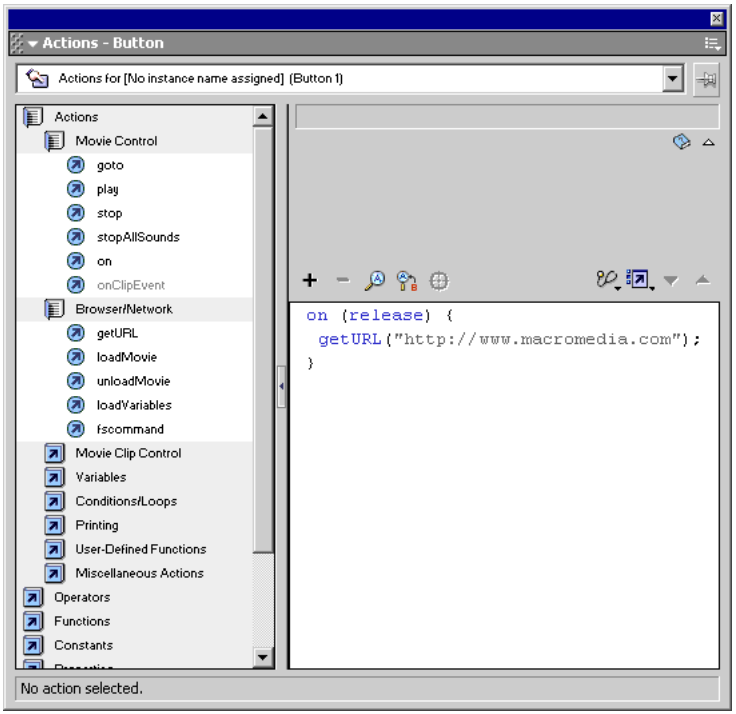
1 Choose Edit > Edit Document or click Scene 1 to return to the main movie.

2 On the Stage, click Button 1, the right button.

3 In the Actions panel, choose Actions > Browser/Network and double-click `getURL`.

4 In the URL text box, type any complete URL, such as **http://www.macromedia.com**.

5 In the Window pop-up menu, select _blank to launch a new browser window when the user clicks Button 1.



6 Save your file, then choose Control > Test movie. Click Button 1 to go to the Web site you specified in step 4.

7 Close the browser and the SWF file, and then return to the Flash authoring environment. If desired, you can select Button 2 on the Stage and repeat steps 3 through 7 to link it to a different Web site, then repeat the steps for Button 3. When you finish linking the buttons, close the Actions panel.

Add streaming and event sounds

When a movie is downloading from an Internet source, a streaming sound can begin to play as soon as the beginning of the sound file has downloaded. Such sounds are especially suited for continuous background sounds.

Event sounds must download completely and load into RAM before playing; event sounds are useful for buttons. In this section, you'll learn how to accomplish the following tasks:

- Add a streaming sound to your movie
- Add an event sound to a button

To complete this section, you can either continue to work on your mystiletto.fla file, or you can browse to your Flash MX application folder and open Tutorials/FlashIntro/stiletto6.fla. If you do use the stiletto6.fla file, save the file with a new name in your My_Stiletto folder to maintain an unadulterated version of the original file.

Add a streaming sound

You can include sound in your movie by dragging the sound to the Stage. You'll add background music that streams and plays for a specified number of times.

1. In the Timeline, with the Buttons layer selected, add a new layer and name it **Sounds**.
2. Choose File > Import. Within your Flash MX application folder, browse to Tutorials/FlashIntro/Assets and click track1.mp3. Control-click (Windows) or Command-click (Macintosh) to add ping.mp3 to the selection, then click Open.

   The files are imported into the library.
3. With the Sounds layer selected, drag the track1.mp3 sound from the Library panel to the Stage.

   In the Timeline, a small representation of sound waves appears in the frame.
4. In the Timeline, select the first frame of the Sounds layer. In the Property inspector, type **999** in the Loop text box to specify the number of times the sound can play continuously.

Test the movie

1. Save your file, then choose Control > Test Movie to hear the sound.
2. When you finish playing the movie, click the movie's close box.

Add an event sound to a button

In addition to dragging a sound to the Stage, you can select a sound from the Property inspector. You will use this method to add an event sound to a button.

As you learned in the Creating Buttons lesson, when you create a button symbol, Flash creates frames for the different button states in relation to the mouse pointer. The Over frame, for example, represents the button's state when the pointer is over the button. Other button frames/states are Up, Down, and Hit.

Now you'll add an event sound to a button, which causes the sound to play during the Over state. Because you're adding the sound to the button symbol in the library, not just to an instance of the symbol, the sound will play for each instance of the button.

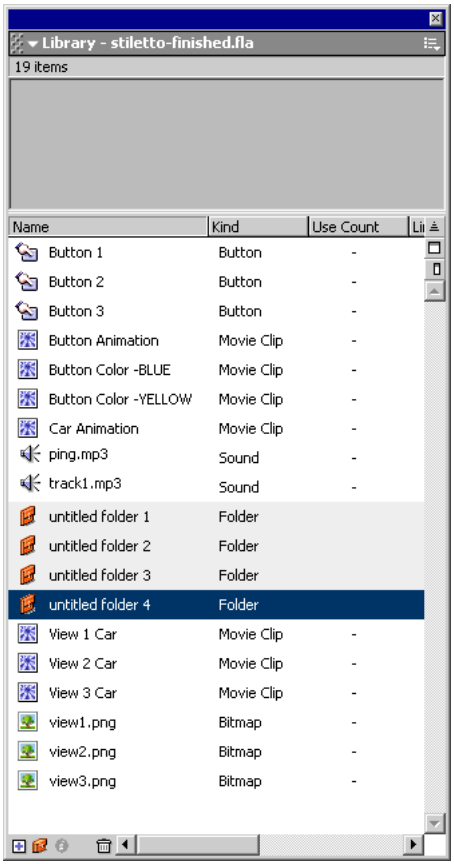
1. In the Library panel, double-click the Button 1 instance to open symbol-editing mode.
2. In the Timeline for Button 1, add a new layer and name the new layer **Sound**.
3. Select the Over frame (Frame 2) of the Sound layer and choose Insert > Keyframe or press F6.
4. To define the sound properties, click Frame 2 of the Sound layer. In the Property inspector, select Ping.mp3 from the Sound pop-up menu. Verify that Event is selected in the Sync pop-up menu.
5. Save your document and choose Control > Test Movie to hear the button sounds. When you finish viewing the SWF file, close its window to return to the document.

Organize your Library panel

You currently have quite a few assets in your Library panel. To keep these assets organized, easy to find, and categorized by type, you'll create folders, then move the assets into the folders.

Tip: Keeping your library panel organized is a good practice for any movie that you author, especially because others working on the same file can then locate assets easily.

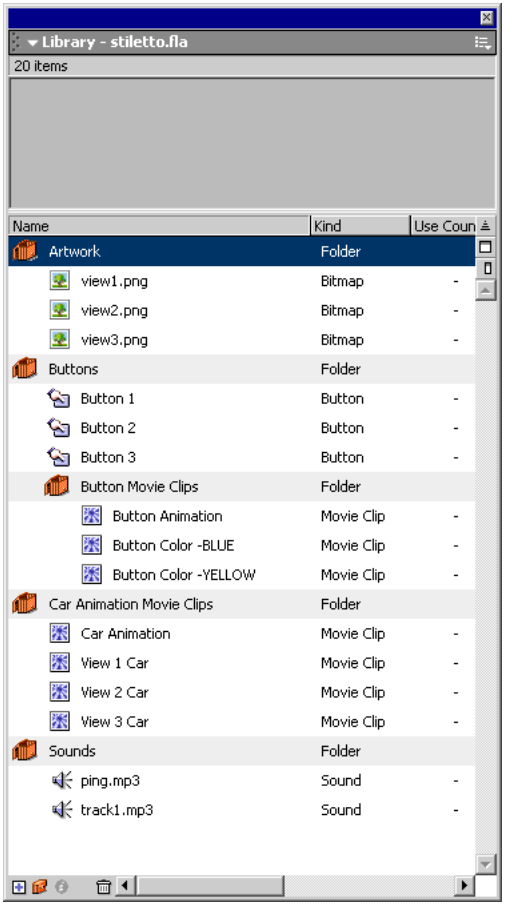
1. If the Library panel is not open, choose Window > Library.
2. Expand and enlarge the window, as necessary, to view all the assets in the Library panel. At the bottom of the Library panel, click the New Folder button four times, to create four new folders.



3. Double-click the name of untitled folder 1, and rename it **Car Animation Movie Clips**.
4. Rename the other three folders **Sounds**, **Artwork**, and **Buttons**.
5. Drag View 1 Car, View 2 Car, View 3 Car, and Car Animation to the Car Animation Movie Clips folder.
6. Drag ping.mp3 and track1.mp3 to the Sounds folder.
7. Drag view1.png, view2.png, and view3.png to the Artwork folder.

8 Drag Button 1, Button 2, and Button 3 to the Buttons folder. With the Buttons folder selected, click New Folder again to create a subfolder with the Buttons folder.

9 Name the new folder **Button Movie Clips**, then drag the remaining button movie clip assets to this folder.



10 Close the Library panel and save your document.

Test download performance and publish the movie

To complete your document, you will use the Publish command to create a Web-compatible movie with the SWF extension.

If you use the Publish command with the default settings, Flash prepares your file for the Web. Flash will Publish the SWF and create an HTML file with the tags necessary to display the SWF.

Once you define the necessary Publish Settings options, you can repeatedly export to all selected formats at once by simply choosing File > Publish. Flash stores the publish settings you specify with the document, so each movie can have its own settings.

In this section, you'll learn how to complete the following tasks:

- Use the Bandwidth Profiler to test movie download performance
- Publish your movie in one step
- Review and modify publish settings
- View your movie in a Web browser

To complete this section, you can either continue to work on your mystiletto.fla file, or you can browse to your Flash MX application folder and open Tutorials/FlashIntro/stiletto7.fla. If you do use the stiletto7.fla file, save the file with a new name in your My_Stiletto folder to maintain an unadulterated version of the original file.

Test movie download performance

For a Flash movie to play correctly over the Internet, a frame must download before the movie reaches that frame. If the movie reaches a frame that hasn't downloaded yet, it pauses until the data arrives. The low bandwidth of Flash files promotes fast downloads.

You can use the Bandwidth Profiler to test your movie and identify where pauses might occur. The Bandwidth Profiler graphically shows how much data is sent from each frame in the movie, according to the selected modem speed.

1. Save your document and choose Control > Test Movie.
2. From the Debug menu, select a modem speed to determine the download rate that Flash will simulate.

   You can also choose Customize to enter a download rate.
3. Choose View > Bandwidth Profiler to see the SWF with a download performance chart.

The shaded bar represents the first and only frame in your main movie. Movies with multiple frames will have multiple shaded bars. The height of the bar represents the frame's size in bytes and kilobytes. Bars that extend above the red line, especially if the bar represents a frame other than the first frame, indicate there could be delays in movie playback. You can optimize your movie for faster downloads. For details, see "Optimizing movies" under Help > Using Flash.

4 When you finish viewing the Bandwidth Profiler, choose View > Bandwidth Profiler to deselect it. Close the SWF window to return to the authoring environment.

Use the Publish command

You can publish your Flash document for Web playback in one step.

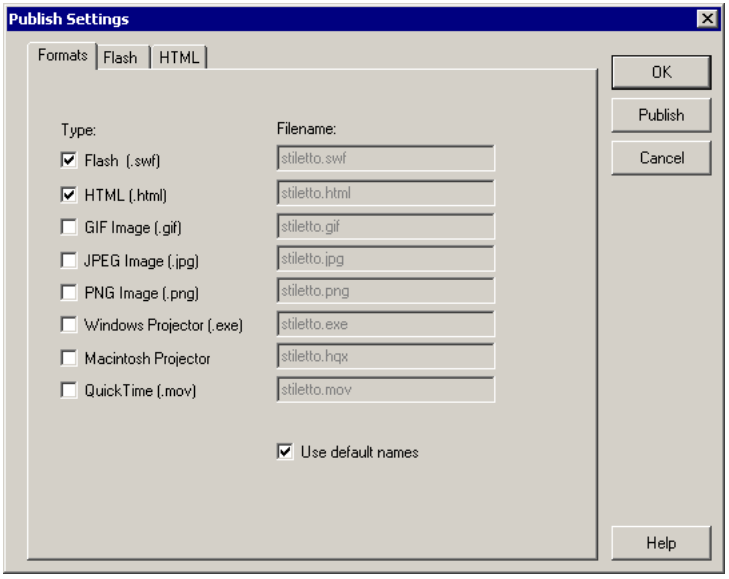
- With your document saved, choose File > Publish.

  Flash publishes your movie by creating a SWF file, and possibly additional files, based on the attributes in the Publish Settings dialog box. You'll find the published files, by default, in the same folder where you saved your FLA.

View publish settings

Using the Publish Settings dialog box, it's easy to reconfigure the way your file publishes.

1 To view your publish settings, choose File > Publish Settings.



Flash is configured, by default, to create a supporting HTML file that displays the Flash movie.

When you select a format that requires additional settings, a new tab appears.

2 On the Formats tab, verify that Flash (.swf) and HTML (.html) are selected. Click the Flash tab.

By default, the movie publishes for the Flash Player. The publishing process also applies movie and JPEG compression.

3 Click the HTML tab.

By default, the publishing process creates an HTML document that inserts your SWF file in a browser window. Settings on the HTML tab of the Publish Settings dialog box determine how the movie appears in the browser.

Change publish settings

By default, Flash gives the SWF file the same name as the FLA file. You can tell Flash to change the name.

1 On the Formats tab of the Publish Settings dialog box, deselect Use Default Names.

2 In the HTML (.html) text box, select the existing text and type a new name, such as **stilettoElectric.html**. Then click Publish. When the Publishing status window closes, click OK in the Publish Settings dialog box.

View your published movie in a browser

You can view the HTML file and SWF movie, which you just published, in your browser.

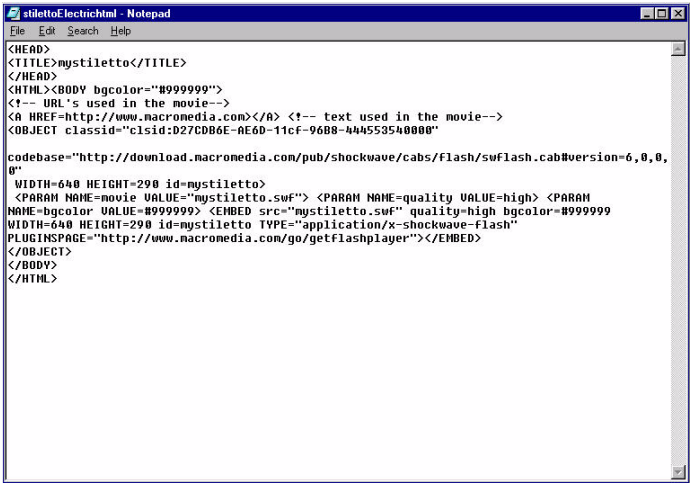
1 Open your browser, then open the HTML file that you created.

By default, the HTML file is in the same folder as your FLA file.

When you open the HTML file, the SWF movie plays within your browser.

2 In your browser, you can use a command such as View > Page Source or View > Source to view the HTML.

OBJECT and EMBED tags ensure that the SWF movie is displayed within the browser.



For additional information about Flash HTML templates, see “About HTML publishing templates” under Help > Using Flash.

The next steps

By completing all eight sections of the tutorial, you've learned much about creating Flash movies, including how to complete the following tasks:

- Analyze a completed movie
- Define document properties and create a gradient
- Create and mask vector art
- Tween bitmap effects within a movie clip
- Load dynamic text
- Modify buttons and add navigation
- Add streaming and event sounds
- Test and publish a movie

Continue learning more about Flash capabilities by taking the Introduction to ActionScript Tutorial under Help > Tutorials. Designed for ActionScript novices, the tutorial introduces you to scripting concepts while allowing you to build a jigsaw puzzle with actions. Additionally, you can search for articles and Tech Notes about Flash MX in the Macromedia award-winning Support Center. To access the site, go to www.macromedia.com and click Support.

CHAPTER 2
Introduction to ActionScript Tutorial

ActionScript is the scripting language of Macromedia Flash MX. A scripting language is a way to communicate with a program; you can use it to tell Flash what to do and to ask Flash what is happening as a movie runs. This two-way communication lets you create interactive movies. In this tutorial, you will examine the tasks involved in creating an interactive jigsaw puzzle.

This tutorial is designed for Flash users who are ActionScript beginners but who want to work toward advanced abilities. You should already be familiar with basic actions and know how to assign them in the Actions panel. To get the most out of this tutorial, you should first complete the Introduction to Flash MX Tutorial, under Help > Tutorials. You should also be comfortable with the concepts presented in "Writing Scripts with ActionScript" and "Creating Interaction with ActionScript," under Help > Using Flash.

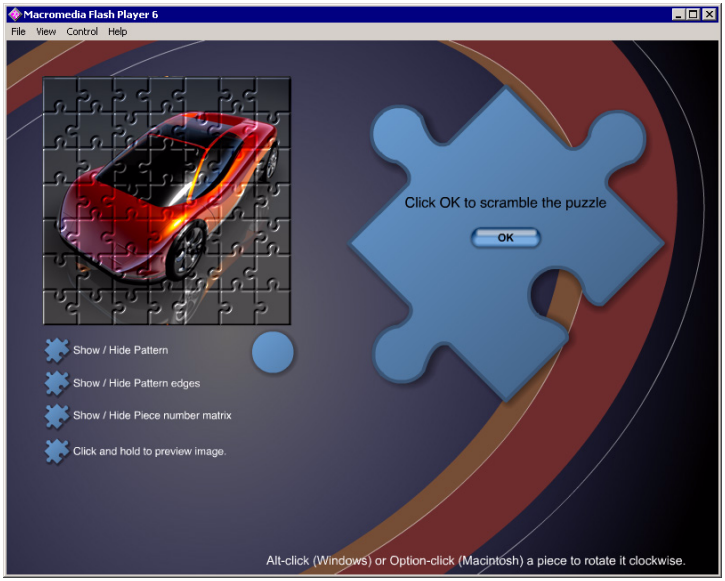
This tutorial takes approximately one hour to complete, depending upon your experience, and will teach you how to do the following tasks:

- Initialize the movie
- Save and retrieve information
- Display information in a dynamic text box
- Write an expression
- Control the flow of the movie
- Create commands and reuse code
- Use a built-in object
- Test the movie

View a completed movie

Before you begin work on your own movie, view a completed version of this tutorial to get an idea of what you'll create. Additionally, the completed tutorial lets you examine the Timeline, Movie Explorer, Stage, and Actions panel to understand authoring practices.

1 Within your Flash MX application folder, browse to Tutorials/ActionScript/Finished. Double-click puzzle.swf to open the completed movie in the stand-alone Flash Player.



2 In the puzzle.swf movie, click the OK button.

The puzzle pieces scramble.

3 Click all the Show/Hide buttons.

Notice how the different patterns and piece numbers are displayed to guide you in completing the puzzle.

4 Click a puzzle piece and drag it to the solution area.

The piece snaps into place.

5 Shift-click a puzzle piece.

The piece number appears in the circle under the solution area. You can match the piece number to its location in the piece number guide if you get stuck.

6 Alt-click (Windows) or Option-click (Macintosh) a puzzle piece.

The piece rotates clockwise.

7 When you finish viewing the SWF file, you can either close the window or leave it open to serve as a reference.

Analyze the puzzle.fla file

It's helpful to analyze the completed FLA file to determine how the author put it together and where the ActionScript elements are located.

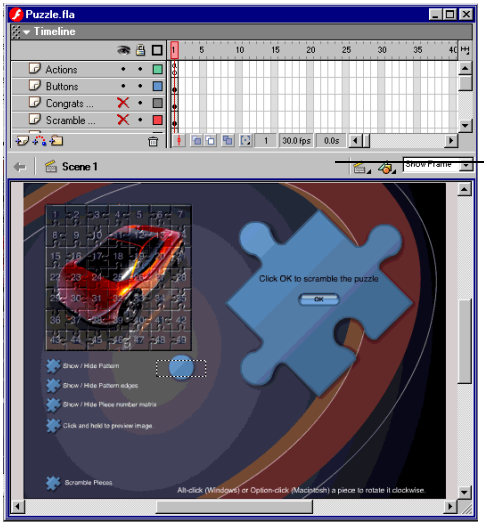
1. In Flash, choose File > Open. Within your Flash MX application folder, browse to Tutorials/ActionScript/Finished and open the puzzle.fla file.

   You now see the completed tutorial movie in the authoring environment.

2. To see all the contents on the Stage, choose View > Magnification > Show Frame.

   The movie contains only one frame, displayed in the main Timeline of the puzzle.fla file. You will use ActionScript to show and hide the various dialog boxes and pattern guides that appear in the puzzle.

3. To resize the Timeline and Stage, drag the bar that separates the Stage from the Timeline up and down. Scroll through the Timeline to see how the layers are organized.



Move this bar to resize the Timeline.

4. To see the dialog boxes and guides on the Stage in the authoring environment, click the red X in the Eye column to the right of a layer's name. A red X indicates a hidden layer.

   You click in the Lock column to lock a layer, which prevents it from being selected. This is useful if you are selecting an item on the Stage that is underneath an item in another layer. The Outline column, indicated by a square above the column, turns on outlines of all the elements in a layer; this can make it easier to see shape edges and can cause Flash to work faster.

5. Select Frame 1 of the Actions layer.

   Frame 1 has a lowercase *a,* which indicates actions are associated with the frame.

   To add an ActionScript element to a movie, you must assign it to either a button, a frame, or a movie clip. Frame scripts are indicated by a lowercase *a* on a frame in the Timeline. To locate button and movie clip scripts, do one of the following:

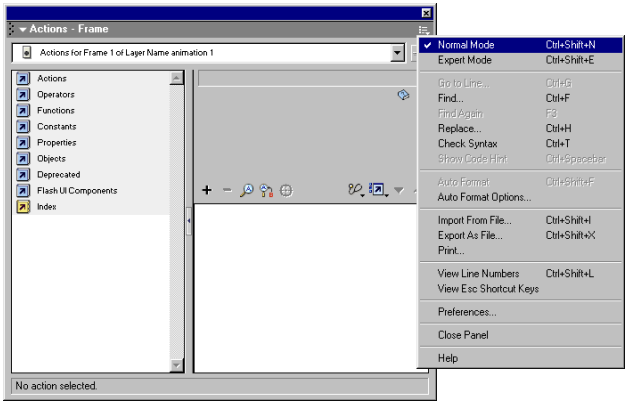
- Open the Actions panel and select a script from the Jump bar—the pop-up list under the Actions panel title bar.

- Select buttons and movie clips on the Stage with the Actions panel open.
- Use the filtering buttons or the Find text box to search for the script in the Movie Explorer.

6 To view the actions, choose Window > Actions.

The Actions panel opens. Expand the panel, if necessary, to see the actions attached to the frame.

The Actions panel has two modes, normal and expert. This tutorial explains how to add actions in normal mode.



View actions in the Movie Explorer

1 To locate all of the actions in the movie, use the Movie Explorer. If it's not open, select Window > Movie Explorer.

In the Movie Explorer, deselect all the filtering buttons except the ActionScript button. In addition to the actions in Frame 1 of the Actions layer, actions are also included in each of the Show/Hide buttons, the buttons in the various dialogs boxes that appear, and each puzzle piece.

2 Select File > Close to close the puzzle.fla movie when you're finished. Do not save changes to the finished file.

Initialize the movie

All movies have an initial state. This shows how a movie looks before it runs and anyone interacts with it. Sometimes you must set variables and movie clip properties to establish this first state. For example, in the puzzle.fla file, certain dialog boxes and pattern guides must be hidden in the initial state.

Each movie clip in a Flash movie has a set of qualities, or *properties,* that you can manipulate with ActionScript. Each of those properties is identified by a name preceded by an underscore (_) character. For example, each movie clip has an `_xscale` property, a `_yscale` property, and a `_rotation` property, among others.

ActionScript uses variables to store information. For example, the variable `myName` might hold the value `"Jody Singer"`. You can learn more about variables in the section "Save and retrieve information" on page 61.

You initialize properties and variables in the first frame of a movie. You can assign a frame action in the Timeline, or attach an object action to a movie clip. In the puzzle movie, the properties of the pattern guide and dialog box movie clips are initialized in the first frame of the main Timeline.

Open the starting file

Now you're ready to create your own version of the tutorial movie.

1. Choose File > Open.
2. In the Flash MX application folder, browse to Tutorials/ActionScript/My_Puzzle and open mypuzzle.fla.
3. If you receive a font substitution message, click Use Default.

   You see a partially completed tutorial movie. The movie may look finished because all the symbols are in place on the Stage. However, you'll still need to add quite a few scripts to make the movie interactive.
4. Choose File > Save As and save the file with a new and sequential name, such as mypuzzle2.fla, in the same folder as mypuzzle.fla.

   Making a copy of the file allows you or another user to complete the tutorial again using the original mypuzzle.fla file. Additionally, if you regularly save your file with a new and sequential name, you can revert to an earlier file if you make an error that you're not able to resolve in your current file.

Set movie clip properties

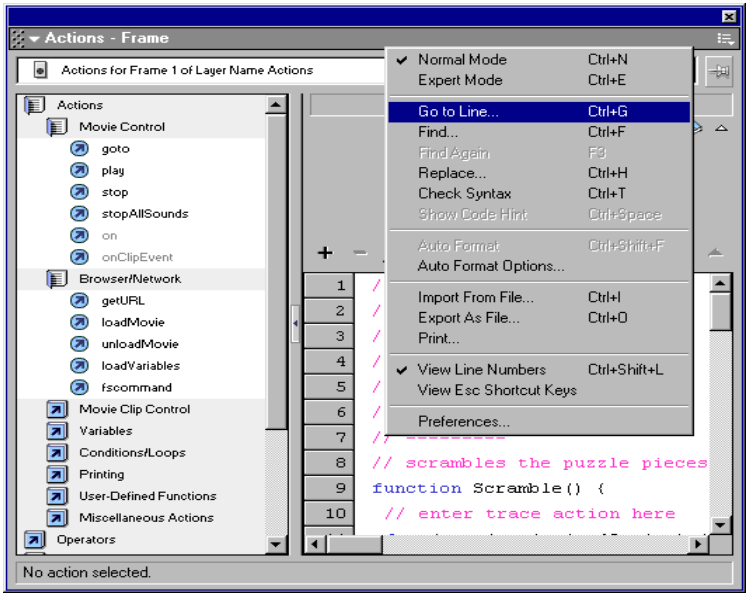
The puzzle.fla movie has three dialog boxes: one alerts you when the puzzle is completed, and the other two ask if you want to scramble the puzzle pieces. Additionally, several patterns and guides overlay the solution area to help users solve the puzzle. Each of these dialog boxes, patterns, and guides is a movie clip.

To initialize the movie, you must hide several of the movie clips so that only the start dialog box and the puzzle pieces are showing. You'll do this by setting their `_visible` properties to `false`.

1. Select Frame 1 in the Actions layer. If the Actions panel isn't open, choose Window > Actions.

   The Actions panel shows actions associated with the selected frame. Text after double slashes (//) is commented text, which offers information helpful in understanding the scripts.

2 Click the pop-up menu in the upper right corner of the Actions panel title bar. Verify that Normal Mode and View Line Numbers are selected.



3 In the Script pane, click line 4 to select the commented code that reads `//ENTER code here`.

4 From the Actions > Miscellaneous Actions category in the Actions toolbox on the left, double-click the `evaluate` action to add an empty line of code ending with a semicolon.

A semicolon (;) in an ActionScript statement is like a period (.) in an ordinary sentence.

5 With the insertion point in the Expression text box of the Actions panel, click the Insert Target Path button.

The Insert Target Path dialog box appears.

6 Verify that Dots, meaning dots notation, and Absolute, meaning absolute path, are selected.

In the dialog box, you see a list of movie clips from which you can select.

7 Select the `edges` movie clip from the movie clip tree, then click OK.

The following code appears in the Expression text box:

```
_root.edges
```

A target path tells ActionScript the location of a movie clip within the overall structure of a Flash movie. The `_root` property refers to the main Timeline and the `edges` movie clip lives on the Stage of the main Timeline. Any target path that begins with `_root` is called an *absolute path* because it gives the complete path to a movie clip from the main Timeline.

8 With the insertion point after `_root.edges`, enter **._visible = false** in the Expression text box.

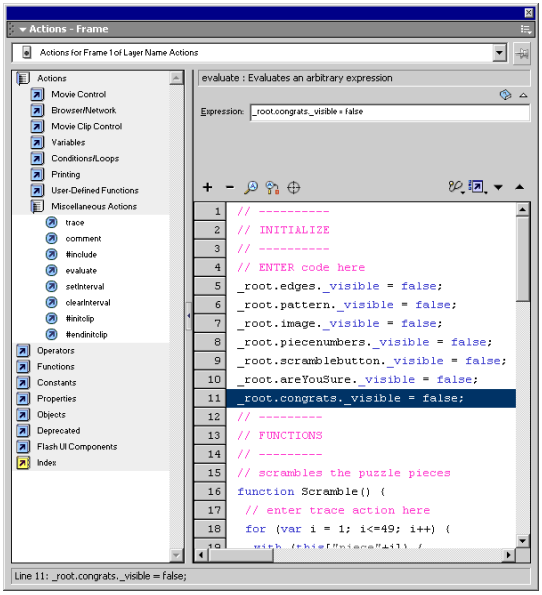
This line of code makes the movie clip invisible on the Stage.

You can type the code manually, or you can choose ActionScript items from the Properties and Operators folders in the Actions toolbox.

If you make an error entering an action and decide you'd like to delete it, select the action in the Script pane and click the Minus (-) button in the Actions panel.

9 In the Actions toolbox, again double-click the `evaluate` action to add an empty line of code ending with a semicolon. Repeat steps 5 and 6, then choose the `pattern` movie clip in the Insert Target Path dialog box. Repeat Step 8, again typing **._visible = false** in the Expression text box.

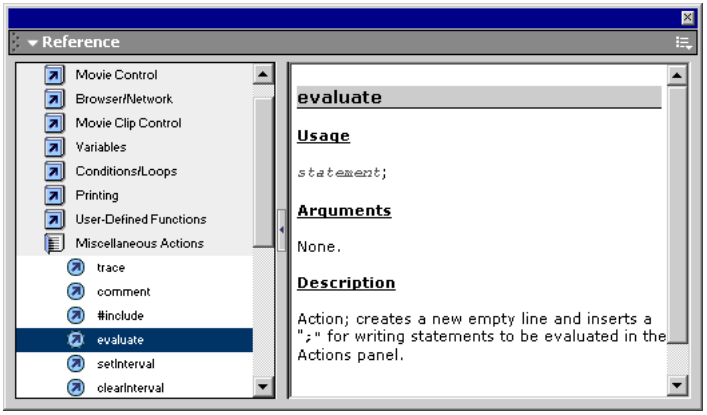
10 Continue to repeat steps 4 through 8, choosing the `image`, `piecenumbers`, `scramblebutton`, `areYouSure`, and `congrats` movie clips in the Insert Target Path dialog box, and typing **._visible = false** for each movie clip.When you finish, your script should appear as follows:



11 Choose File > Save As and enter a new filename. Use a consecutive naming scheme, such as mypuzzle3.fla, so you can revert to earlier versions of the file, if necessary.

Use the Reference panel

During authoring, if you'd like additional information about the ActionScript that you enter, you can select the action in the Actions toolbox or Script pane, then click the Reference button. The Reference panel, a help system organized similarly to the Actions panel, displays information about the selected action.



Test your syntax

ActionScript, like written language, depends on correct syntax. If the syntax is incorrect, the action will not execute correctly. Flash offers a variety of ways for you to test your syntax.

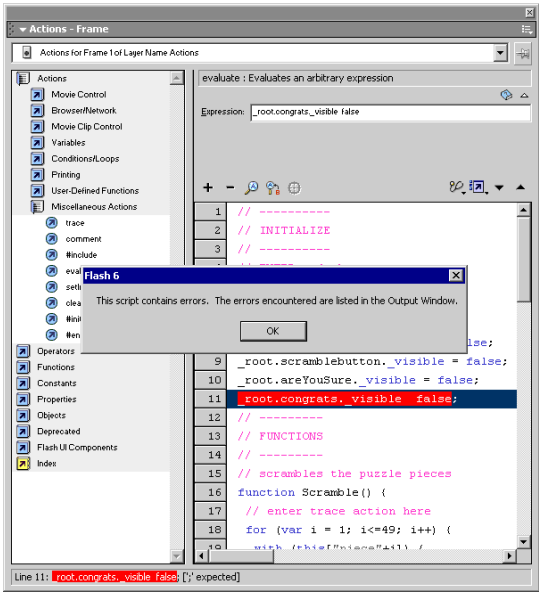
1. Click the Options menu in the upper right corner of the Actions panel title bar and choose Check Syntax.

   If the syntax is correct, a message appears stating that the script contains no errors.

   If the syntax is incorrect, a message appears stating the script contains errors; the Output window opens and displays information about the error.

2. Click OK to close the syntax message.

In normal mode, ActionScript syntax errors appear highlighted in red in the Script pane. If you move your mouse pointer over the error, a tooltip displays the error message. Syntax error messages also appear highlighted along the bottom status area of the Actions panel.



Additionally, as you learned in the Introduction to Flash MX Tutorial, you can choose Control > Test Movie throughout authoring to test if your movie plays as expected.

Save and retrieve information

To create a complex interactive Flash movie, you need a way for Flash to keep track of information and user activity: buttons that have been pressed, a user's name, a score, or what sections a user has visited. ActionScript uses variables to store pieces of information that you can retrieve and use again. You can declare a variable in a script on any Timeline and use it in any other Timeline in the same movie. You must write a target path to a variable in order to use the variable in a script, just as you must write a target path to use a movie clip in a script.

In the puzzle.fla file, ActionScript uses the `dialog` variable to keep track of whether or not a dialog box is visible. When a dialog box appears, the `dialog` variable is set to `true`; when a user clicks a button on a dialog box, the `dialog` variable is set to `false`. This variable doesn't affect the visibility of the dialog boxes themselves, it is simply a container that holds information that you can use in scripts throughout the movie. In the puzzle.fla file, if `dialog` is set to `true`, a user cannot move a puzzle piece.

Declare a variable and assign it a value

When you need a variable, you must name it, or *declare* it. You must also assign it a value. You can either do both things at once, or you may declare a variable in one statement and then assign it a value in a later statement.

ActionScript uses three types of variables: local variables, global variables, and Timeline variables. You can use the `var` action inside a block of code (designated by curly brackets `{}`) to create a local variable, which disappears when the code block finishes running. You can use the `set variable` action to create a Timeline variable attached to the Timeline of a movie clip, which can be used in any script in the document. For more information about variables, see "Understanding the ActionScript Language" under Help > Using Flash.

The puzzle.fla file uses the `var` action and the `set variable` action depending on the situation. When a variable is only needed within a block of code, the `var` action is used. The `dialog` variable is set and assigned using the `set variable` action.

Now you'll declare and assign a value to the `dialog` variable:

1. Select File > Open and choose the version of mypuzzle.fla that you last saved.

   Note: You can also browse to your Flash MX application folder and open Tutorials/ActionScript/Finished/puzzle2.fla. If you do use the puzzle2.fla file, save the file with a new name in your My_Puzzle folder to maintain an unadulterated version of the original file.

2. Select Frame 1 of the Actions layer. If the Actions panel isn't open, choose Window > Actions.

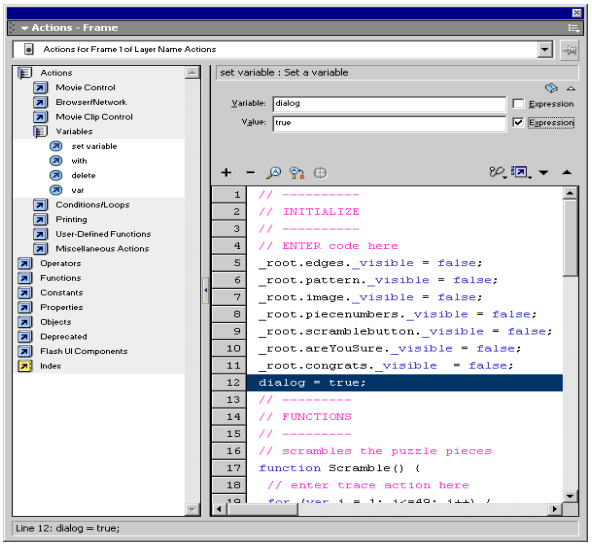
3. In the Script pane of the Actions panel, select line 11, which is the last line of code in the Initialize section. In the Actions toolbox, choose Actions > Variables and double-click the `set variable` action.

4. Type **dialog** in the Variable text box.

5. Type **true** in the Value text box. Select Expression, to the right of the Value text box.

   By selecting Expression, you are telling Flash that `true` is not a literal string of characters.

   In the movie's initial state, a dialog box is visible on the Stage. Therefore, the `dialog` variable must be set to `true` at the start of the movie—otherwise, a user can move the puzzle pieces before they are scrambled.



6. Choose File > Save As and enter a new filename. Use a consecutive naming scheme so you can revert to earlier versions of the file, if necessary.

Display information in a dynamic text box

Dynamic text boxes can display changing content in a Flash movie. You use the Property inspector to create a dynamic text box and assign it a variable name. The value of the variable is displayed in the text field.

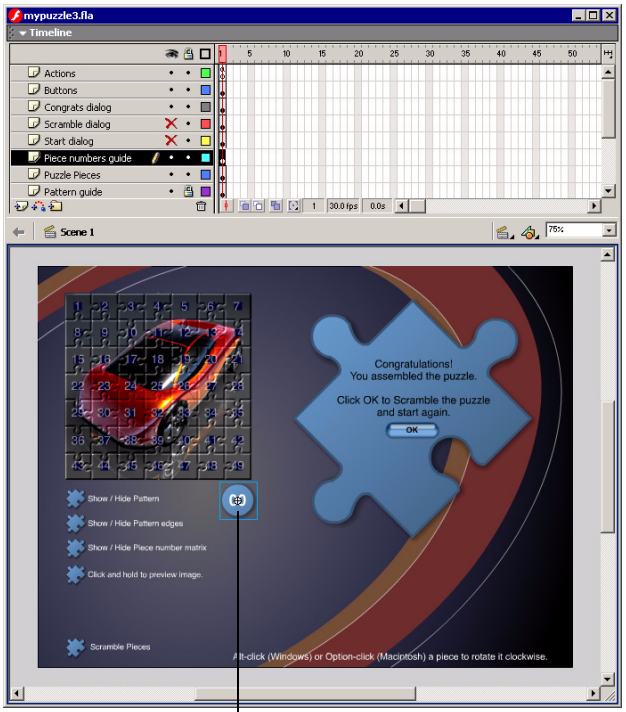
In the puzzle.fla file, a dynamic text box displays puzzle piece numbers when a user Shift-clicks a piece. Now you'll assign a variable name for the dynamic text box.

1 If necessary, choose File > Open and choose the version of the mypuzzle.fla file that you last saved.

Note: You can also browse to your Flash MX application folder and open Tutorials/ActionScript/Finished/puzzle3.fla. If you do use the puzzle3.fla file, save the file with a new name in your My_Puzzle folder to maintain an unadulterated version of the original file.

2 In the Timeline, unlock the Piece numbers guide layer if it is locked.

3 Double-click the Piece number circle movie clip on the Stage under the lower right corner of the puzzle solution area.



Piece number circle movie clip

This opens the Piece number circle in symbol-editing mode.

4 Select the Text layer in the Timeline. On the Stage, click the center of the Piece number circle to select the text field.

5 If the Property inspector is not open, choose Window > Properties. In the Property inspector, select Dynamic Text from the Text Type pop-up menu.

6 In the Line Type pop-up menu, verify that Single Line is selected.

7 Type **_root.pieceNumber** in the Var text box and press Enter or Return.

All variables, like functions and movie clips, must be referenced by their paths. The `pieceNumber` variable is declared and updated in the `RotateDisplayOrDrag` function on the main Timeline, but the variable text box is in the Piece number circle Timeline. When you enter the full path to the `pieceNumber` variable, the value updates and displays in the text field on the Stage whenever the variable's value changes on the main Timeline.

8 Either choose Edit > Edit Document, click the Back button, or click Scene 1 to return to the main Timeline.

9 Choose File > Save As and enter a new filename. Use a consecutive naming scheme so you can revert to earlier versions of the file if necessary.

Write an expression

If you've added two numbers in a math equation, you've used an operator. An *operator* is a symbol that performs a task, or *operation,* on a piece, or pieces, of data, or *operands*. For example, in the expression `2 + 2`, the plus sign (`+`) is the operator and each number is an operand.

An *expression* is any ActionScript code that can be evaluated to produce a single value. For example, the code `myAge + 17` is an expression because when the code runs, ActionScript looks up the value of the `myAge` variable, adds it to the number `17`, and produces a new single value. If the value of `myAge` is `47`, the new value would be `64`.

Operators allow you to take the information you collect and store in variables and manipulate it in expressions to create or determine other values. For example, you may know that a user has dropped a puzzle piece onto the solution area, but how do you determine if the piece is in the correct place? And if the piece is in the correct place, how do you determine if the puzzle has been solved? To examine expressions built with arithmetic operators in such a scenario, select Frame 1 of the main Timeline, open the Actions panel, and look at the `IsItDone` function on line 50.

ActionScript has numeric, or *arithmetic,* operators, but it also has other types of operators that perform different types of operations. For example, a comparison operator compares values to determine if one operand is greater than, less than, or equal to the other, and a logical operator calculates a `true` or `false` value, also called a *Boolean* value, for an expression.

Now you'll use an operator called the logical NOT operator to write an expression that shows and hides the puzzle pattern. The logical NOT operator, which is represented by an exclamation mark (`!`), calculates the opposite Boolean value of its operand. For example, the expression `!true` yields the value `false`.

1. If necessary, choose File > Open and choose the version of the mypuzzle.fla file that you last saved.

   Note: You can also browse to your Flash MX application folder and open Tutorials/ActionScript/Finished/puzzle4.fla. If you do use the puzzle4.fla file, save the file with a new name in your My_Puzzle folder to maintain an unadulterated version of the original file.

2. Click the Show/Hide Pattern edges button on the Stage. If the Actions panel isn't open, choose Window > Actions.

   The Actions panel shows actions associated with the button.

3. Select the following line of code in the Script pane:

   ```
   //  ENTER code here
   ```

4. In the Actions toolbox, choose Actions > Miscellaneous Actions and drag the `evaluate` action to the Script pane.

   When you add the action, it is enclosed in code called an *event handler*. The code looks like this:

   ```
   on (release) {
     ;
   }
   ```

5. In the Actions toolbox, double-click the `evaluate` action to add another empty line of code.

   The code now looks like this:

   ```
   on (release) {
     ;
     ;
   }
   ```

6. Select the first empty line—the line with the first semicolon—and place the insertion point in the Expression text box.

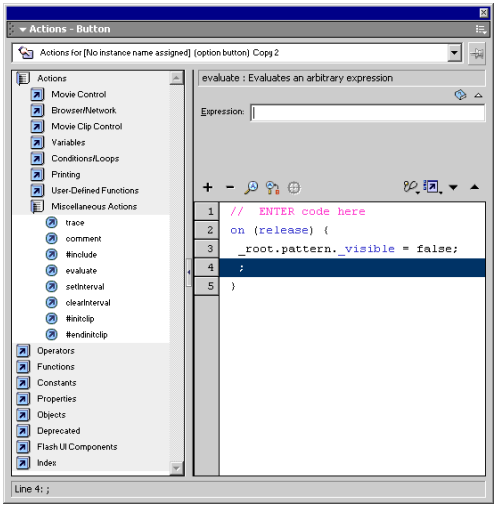
   You'll now enter code that hides the pattern movie clip; a user should see either the pattern movie clip or the edges movie clip, but not both.

7. Enter **_root.pattern._visible = false** in the Expression text box.

   You can type the path directly in the Expression text box or use the Insert Target Path dialog box to select the `pattern` movie clip. If you use the Insert Target Path dialog box, select Dots and Absolute.

   Note: As you enter parameters and properties in normal mode, you might notice code hints–tooltips that appear suggesting the complete syntax for an action; you can click a code hint to enter the syntax. For more information about code hints, include enabling and disabling them, see "Using code hints" under Help › Using Flash.

8 Select the second empty line and place the insertion point in the Expression text box.



9 Type **_root.edges._visible = !** in the Expression text box.

Ignore the syntax error message that appears.

The exclamation mark (!) is a logical NOT operator. In addition to simply typing it in the Expressions text box, you can add it to the text box from the Actions toolbox by choosing Operators > Logical Operators and double-clicking the exclamation mark.

10 Enter **_root.edges._visible** again in the Expression text box after the operator.

Your code should look like this:

```
on (release){
  _root.pattern._visible = false;
  _root.edges._visible = !_root.edges._visible;
}
```

The first line of code inside the event handler sets the visibility of the `pattern` movie clip to `false`. The second line of code sets the visibility of the `edges` movie clip to the opposite of what it currently is. This creates a toggle that either shows or hides the movie clip.

11 Choose File > Save As and enter a new filename. Use a consecutive naming scheme so you can revert to earlier versions of the file, if necessary.

Control the flow of the movie

Without ActionScript, a movie plays from Frame 1 to the last frame and either repeats from Frame 1 or stops. You can use ActionScript to control the progression of a movie more precisely; you can also use it to give the user control. For example, you could place an action in Frame 5 that stops the movie until a user presses a Play button. This is a simple example of controlling the flow of a movie.

You can use the `if`, `else`, and `else if` actions (also called *statements*) to create a more complex movie flow called *logic*. These three actions perform the following tasks:

- The `if` action lets Flash check a condition in the movie and run certain actions if that condition is true.
- The `else` statement tells Flash to run a different set of actions if the `if` condition is false.
- The `else if` statement lets Flash check for another condition before running yet a different set of actions.

Write a conditional statement

You've already used an operator to show and hide a movie clip. Now you'll use an `if` statement to create logic that shows and hides the `piece numbers` movie clip. For the sake of learning, this example uses a different ActionScript element to achieve the same result.

1. If necessary, choose File > Open and choose the version of the mypuzzle.fla file that you last saved.

   Note: You can also browse to your Flash MX application folder and open Tutorials/ActionScript/Finished/puzzle5.fla. If you do use the puzzle5.fla file, save the file with a new name in your My_Puzzle folder to maintain an unadulterated version of the original file.

2. On the Stage, click the Show/Hide Piece number matrix button. If the Actions panel isn't open, choose Window > Actions. In the Actions toolbox, choose the Actions > Conditions/Loops category.

3. Double-click the `if` action. The following code appears in the Actions panel:

```
on (release) {
    if (<not set yet>) {
    }
}
```

4. With the line of code that contains the `if` action selected, double-click the `else` action in the Actions toolbox.

   The code looks like the following:

```
on (release) {
    if (<not set yet>) {
    } else {
    }
}
```

5. Select line 3, which begins with `if`, and with the insertion point in the Condition text box, click the Insert Target Path button. Select the `piecenumbers` movie clip, Dots, and Absolute, and click OK. The following code appears in the Condition text box:

```
_root.piecenumbers
```

6. With the insertion point in the Condition text box, enter **._visible** after `_root.piecenumbers`.

7. From the Actions > Miscellaneous Actions category in the Actions toolbox, double-click the `evaluate` action to add an empty line of code.

8 Enter **_root.piecenumbers._visible = false** in the Expression text box.

You can use the Insert Target Path button or type the code manually. The code now looks like the following:

```
on (release) {
    if (_root.piecenumbers._visible) {
        _root.piecenumbers._visible = false;
    } else {
    }
}
```

When the movie plays, Flash evaluates the expression inside the condition parentheses. The expression must equal one of the Boolean values: `true` or `false`. This example uses the condition of the `if` action to check if the `piecenumbers` movie clip is visible on the Stage. If the `_visible` property is `true`, ActionScript runs the code inside the curly brackets and sets the `_visible` property to `false`, which hides the movie clip on the Stage.

9 In the Script pane, select the line of code with the `else` action and double-click the `evaluate` action.

10 Enter **_root.piecenumbers._visible = true** in the Expression text box.

The final ActionScript code looks like this:

```
on (release) {
    if (_root.piecenumbers._visible) {
        _root.piecenumbers._visible = false;
    } else {
       _root.piecenumbers._visible = true;
    }
}
```

11 Choose File > Save As and enter a new filename. Use a consecutive naming scheme so you can revert to earlier versions of the file if necessary.

Create commands and reuse code

ActionScript has actions that let you control a movie in specific ways. For example, the `play` action moves the playhead forward in the Timeline, and the `loadMovie` action loads another Flash movie into the Flash Player. Each of these actions instructs Flash to perform a certain task. You may want to create your own commands in your movies. For example, in puzzle.fla, you need a command to scramble the puzzle pieces. To figure out how to write such a command with ActionScript, you must determine each step required to scramble the puzzle pieces and determine which ActionScript elements can be used to achieve those goals.

First, the pieces must be spread out within a certain area on the Stage; each movie clip has an `_x` and `_y` property that you can use to set its position and a `_rotation` property that you can use to set its rotation. To place and rotate each piece randomly, you need to generate a random number to use in an expression. ActionScript has a built-in `Math` object with a `random` method that you can use for this purpose.

A command in ActionScript is called a *function*. A function is a script that you can use over and over again in a movie to perform a certain task. For example, in puzzle.fla, every time a user clicks a Scramble Pieces button, the function `Scramble` is run, or *called*. This function places the puzzle pieces in random positions on the Stage. Instead of rewriting that same script on each of the two Scramble Pieces buttons, the function is written, or *declared,* once and called from each button. To examine the `Scramble` function, select Frame 1 in the main Timeline and open the Actions panel. Scroll down the Script pane until you see the `Scramble` function.

You can think of a function as a machine that does extra work for you. The machine can produce different results depending on what you put into it. For example, if you put bananas in a blender, you get mashed bananas, not mashed peaches. The elements you pass to a function to work on are called *parameters* or *arguments*. Parameters are passed inside the parentheses that follow the function. For example, the function `RotateDisplayOrDrag(whichPiece)` is passed the name of a puzzle piece, and it operates only on that piece. Parameters allow you to reuse functions in many different situations.

Functions are usually declared in the first frame of a movie. In the puzzle.fla files, the functions are declared in Frame 1.

Write a function

Now you'll declare a function that will rotate, display, or drag each puzzle piece when the user clicks it.

1. If necessary, choose File > Open and choose the version of the mypuzzle.fla file that you last saved.

   Note: You can also browse to your Flash MX application folder and open Tutorials/ActionScript/Finished/puzzle6.fla. If you do use the puzzle6.fla file, save the file with a new name in your My_Puzzle folder to maintain an unadulterated version of the original file.

2. Select the first frame of the Actions layer and open the Actions panel if it's not already open.
3. Scroll down the Script pane and select line 31. The following commented line should be highlighted:

   ```
   // ENTER RotateDisplayOrDrag() function here
   ```

4. From the Actions > User Defined Functions category in the Actions toolbox, double-click the `function` action.

   Type **RotateDisplayOrDrag** in the Name text box. Type **whichPiece** in the Parameters text box. The code for line 32 now looks like this:

   ```
   function RotateDisplayOrDrag (whichPiece) {
       }
   ```

   The `whichPiece` parameter, which identifies the selected puzzle piece, will be called three times in the body of the function. When the function is called, the passed parameter is substituted for `whichPiece` in each statement.

5. From the Actions > Conditions/Loop category in the Actions toolbox, double-click the `if` action, the `else if` action, and the `else` action.

   Note: You can also select the actions from the Plus (+) pop-up menu

The code looks like this:

```
function RotateDisplayOrDrag (whichPiece) {
        if (<not set yet>) {
        } else if (<not set yet>) {
        } else {
        }
    }
```

This code creates the logical structure of the function. You will fill in the conditions to be checked in each `if` statement. You will also fill in the code within each set of curly brackets that is executed when the conditions are true.

6 Select the `if` statement line of code. Enter **Key.isDown(18)** in the Condition text box.

Key is a built-in ActionScript object, which you can also find in the Objects > Movie > Key > Methods category. Key lets you determine what key a user pressed on the keyboard. The number 18 is the numerical representation of the Alt (Windows) and Option (Macintosh) keys. This line of code checks whether a user pressed those keys.

To learn more about built-in objects, see "Use a built-in object" on page 73.

7 From the Actions > Miscellaneous Actions category in the Actions toolbox, double-click the `evaluate` action to enter a new line of code. Type **_root[whichPiece]._rotation += 90** in the Expression text box, with no space between the + and =.

This line of code rotates the selected piece 90° if the user presses the Alt (Windows) or Option (Macintosh) key. The brackets let you dynamically retrieve the value of an instance name. For more information, see "Dot and array access operators" under Help > Using Flash.

8 Select the `else if` line of code. Type **Key.isDown(Key.SHIFT)** in the Condition text box.

This line of code checks whether a user pressed the Shift key.

9 In the Actions > Miscellaneous Actions category in the Actions toolbox, double-click the `evaluate` action to enter a new line of code. Type **pieceNumber = whichPiece.slice(5)** in the Expression text box.

This line of code displays the piece number in the text box on the Stage when a user presses the Shift key. The `slice` method of the String object removes a specified number of characters (in this case, 5) from the piece number's instance, so that just the number of the piece appears. In effect, the method "slices" off the first five characters, then assigns the resulting number to the `pieceNumber` variable, which is in turn assigned to the text field on the Stage.

10 Select the `else` line of code. In the Actions > Movie Clip Control category in the Actions toolbox, double-click the `startDrag` action.

11 Type **_root[whichPiece]** in the Target text box and click Expression.

12 Select Constrain to rectangle. Type **20** in the L (left) and T (top) text boxes. In the R (right) and B (bottom) text boxes, type **780** and **580**, respectively.

The word `false` in the script indicates that `lockCenter` (which indicates that the puzzle piece will always snap to the center of the mouse pointer when clicked) is not specified. The numbers `20, 20, 780,` and `580` specify the left, top, right, and bottom coordinates of the text box on the Stage.

When a user clicks a piece, the following function that you wrote is called. The function uses the Key object to determine if the Shift, Alt, or Option key is pressed when a piece is clicked. If the Shift key is pressed while a piece is clicked, a dynamic text box displays the puzzle piece number; if the key pressed is Alt (Windows) or Option (Mac), the puzzle piece rotates 90°. If Shift, Alt, or Option keys are not pressed, the user can drag the piece. Your code should look like this:

```
function RotateDisplayOrDrag (whichPiece) {
  if (Key.isDown(18)) {
    _root[whichPiece]._rotation += 90;
  } else if (Key.isDown(Key.SHIFT)) {
    pieceNumber = whichPiece.slice(5);
  } else {
    startDrag(_root[whichPiece],false, 20, 20, 780, 580);
  }
}
```

Note: As you complete the tutorial, remember to save your work frequently.

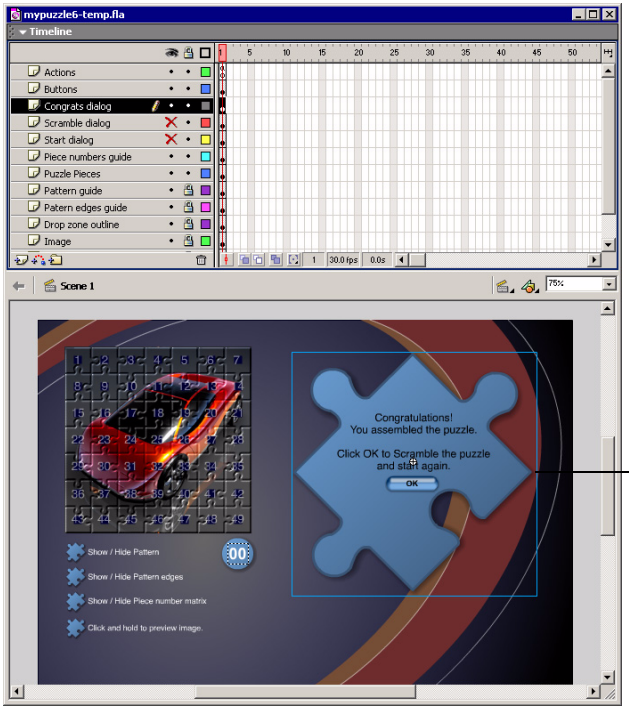
Call a function

Functions can be called from any frame in a movie where you need a task completed. You must use a target path to call a function, just as you must use a path to access a variable or a movie clip. All functions in the puzzle.fla movie are declared in the first frame of the Actions layer, in the main Timeline, so the absolute path to each begins with `_root`.

Now you'll call the function that scrambles the puzzle pieces on the Stage.

1 In the Timeline, hide the Scramble dialog and Start dialog layers, if they're not already hidden. Select the Congrats dialog layer.

2 On the Stage, double-click the Dialog—Congratulations symbol.



3 On the Stage, select the OK button. If the Actions panel isn't open, choose Window > Actions.

4 In the Actions > User-Defined Functions category of the Actions toolbox, double-click the `call function` action.

5 With the insertion point in the Object text box of the Actions panel, click the Insert Target Path button. Verify Dots and Absolute are selected. Select `_root` and click OK.

6 Type **Scramble** in the Method text box.

The `Scramble` function doesn't require any parameters; you can leave the Parameters text box empty.

The code now looks like this:

```
on (release) {
    _root.Scramble();
}
```

This is the ActionScript that calls the function. You'll also need to add a few additional lines of code that belong inside the `on(release)` handler. You'll do that in the next steps.

With the `_root.Scramble...` line of code selected in the Script pane, double-click evaluate from the Actions > Miscellaneous Actions category in the Actions toolbox. Type **_root.scramblebutton._visible = true** in the Expression text box.

7 Double-click the `evaluate` action again, to add another blank line. Type **_root.dialog = false** in the Expression text box.

8 From the Actions > Miscellaneous Actions category of the Actions toolbox, double-click the `evaluate` action.

9 From the Properties category in the Actions toolbox, double-click `_visible`. With the insertion point after `_visible`, type **= 0** in the Expression text box.

This code specifies that the dialog box is not visible after the user clicks the OK button.

The final code appears as follows:

```
on (release) {
  _root.Scramble();
  _root.scramblebutton._visible = true;
  _root.dialog = false;
  _visible = 0
}
```

10 Do one of the following to return to the main Timeline:

- Choose Edit > Edit Document.
- Click the Back button.
- Click Scene 1.

11 Choose File > Save As and enter a new filename. Use a consecutive naming scheme so you can revert to earlier versions of the file, if necessary.

Note: As you complete the tutorial, remember to save your work frequently.

Use a built-in object

ActionScript has *variables* that let you store information; it has *functions* that let you create special commands and reuse code; it has *actions* that let you control the flow of a movie; and it has movie clips with *properties* that you can change.

ActionScript also has another type of element called a built-in object. *Objects* provide a way of grouping information so that you can use it in a script. Objects can have properties, methods (which are like functions), and constants (for example, the numeric value of Pi).

In the `RotateDisplayOrDrag` function that you created in "Create commands and reuse code" on page 68, you used the Key object to determine the last key a user pressed on the keyboard. The Key object is built into ActionScript to allow you to access information about the keyboard.

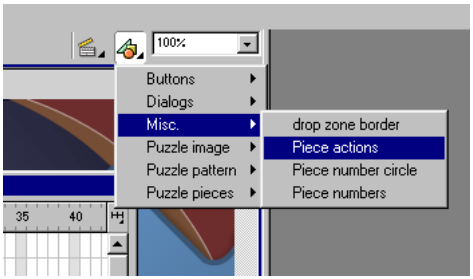
Another ActionScript object is the MovieClip object. The MovieClip object is a collection of methods that you can use to manipulate movie clips, which are the most fundamental and powerful elements of Flash. To learn more about the special characteristics of the MovieClip object and using movie clips, see "Working with Movie Clips and Buttons" under Help > Using Flash.

You will now use the `onPress` method of the MovieClip object to check whether the mouse pointer is touching a puzzle piece.

1 If necessary, choose File > Open and choose the version of the mypuzzle.fla file that you last saved.

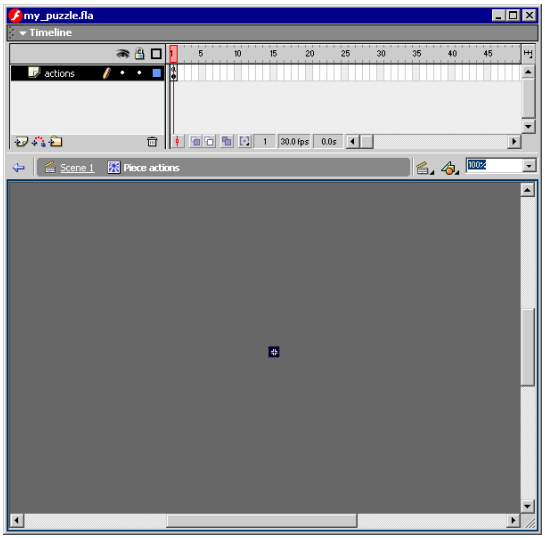
Note: You can also browse to your Flash MX application folder and open Tutorials/ActionScript/Finished/puzzle7.fla. If you do use the puzzle7.fla file, save the file with a new name in your My_Puzzle folder to maintain an unadulterated version of the original file.

2 Piece actions is a movie clip nested within each instance of a puzzle piece. To select the Piece actions movie clip from the Library panel hierarchy, click the Edit Symbols button in the lower right corner of the Timeline and choose Misc > Piece actions.



The Piece actions movie clip opens in symbol-editing mode.

3 In the Piece actions Timeline, select Frame 1 of the Actions layer.



4 If the Actions panel is not already open, choose Window > Actions. In the Script pane, select line 3 with the following line of code:

```
// ENTER code here
```

5 In the Objects > Movie > MovieClip > Events category of the Actions toolbox, double-click the `onPress` action.

The `onPress` method is a special type of action called an *event handler,* or simply an *event*. An event allows you to write a function that runs when a certain event occurs. Types of events include when the mouse button is pressed, when the playhead enters a frame, and when a movie clip loads.

In this procedure, the code within the curly brackets that follow `onPress` runs when a user presses the mouse button in the movie.

6 Type **_parent** in the Object text box.

Because Piece actions is nested within the movie clip, `_parent` specifies that `onPress` should execute on code one level up, at the same level as the puzzle piece.

7 In the Actions > Conditions/Loops category of the Actions toolbox, double-click the `if` action.

8 Type **!_root.dialog** in the Condition text box. The code appears as follows:

```
_parent.onPress = function(){
  if (!_root.dialog) {
  {
};
```

The code that you added in this step tests whether the value of the variable named `dialog` is true (visible) or false (not visible). If the value is `true`, then the next script to rotate and drag the puzzle piece will not run. If the value of the variable is `false`, then the following script will run. Users can't rotate or drag a piece, or display its piece number, if the dialog box is showing.

9 In the Actions toolbox, double-click the `evaluate` action in the Actions > Miscellaneous Actions category to add it inside the curly brackets of the `if` statement.

10 Type **_root.RotateDisplayOrDrag(_parent._name)** in the Expression box.

`_name` is the property for the name of the puzzle piece instance that the user clicks.

The final code looks like this:

```
_parent.onPress = function(){
  if (!_root.dialog) {
  _root.RotateDisplayOrDrag(_parent._name);
  }
};
```

11 Do one of the following to return to the main Timeline:

- Choose Edit > Edit Document.
- Click the Back button.
- Click Scene 1.

12 Choose File > Save As and enter a new filename. Use a consecutive naming scheme so you can revert to earlier versions of the file, if necessary.

Test the movie

"Test early and test often" is a mantra for Web developers. The sooner you test your movie, the easier it is to determine the cause of unexpected results. It's a good idea to save multiple versions of your file using sequential names—for example, mypuzzle1.fla, mypuzzle2.fla, and so on, as you've been doing in this tutorial. This way, the highest numbered file is always the newest and it's easy to revert to an earlier version.

As you learned in the Introduction to Flash MX Tutorial, the Flash authoring tool includes a version of the Flash Player that lets you test your movie at any point during authoring by choosing Control > Test Movie. This version of the Flash Player has several commands and tools to help you troubleshoot your movie.

The most common reason for unexpected results in a Flash movie is an incorrect path to a variable, function, object, or movie clip. This can cause a variable to contain the wrong value, prevent a function from being called, or prevent a movie clip from playing correctly.

The `trace` action allows you to write notes and expressions in your scripts whose results are sent to a window called the Output window.

Now you'll add a `trace` action to test if a function was called successfully.

Note: Flash also includes a Debugger, which lets you examine the values of global and local variables, including when they change as the movie plays. Additionally, with the Debugger you can use breakpoints to stop the movie and test ActionScript line by line. For more information, see "Using the Debugger" under Help › Using Flash.

1. Choose File > Open and choose the version of the mypuzzle.fla file that you last saved.

   Note: You can also browse to your Flash MX application folder and open Tutorials/ActionScript/Finished/puzzle8.fla. If you do use the puzzle8.fla file, save the file with a new name in your My_Puzzle folder to maintain an unadulterated version of the original file.

   When the user clicks the OK button in the SWF movie, the puzzle pieces should scramble. You can use a `trace` action to test if the `Scramble` function is being called.

2. In the Actions layer in the Timeline, select Frame 1 and open the Actions panel.
3. In the Script pane, scroll to and select line 18, the commented line that says `// enter trace action here`.
4. In the Actions toolbox, double-click the `trace` action from the Actions > Miscellaneous Actions category.
5. Type **Scramble function has been called** in the Message text box.

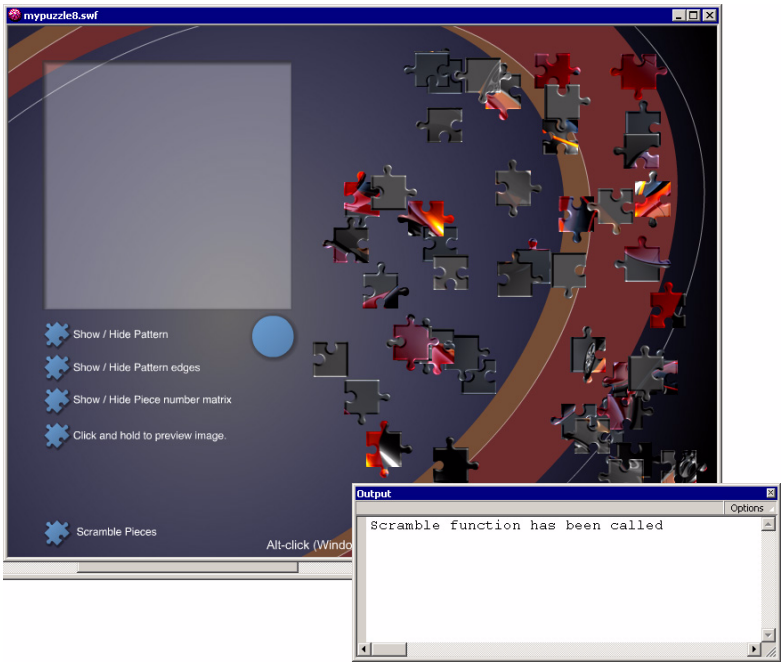
   You are placing this message within the Scramble function.

6. Save your document, then choose Control > Test Movie.

7 Click the OK button in the SWF movie.

The Output window appears, which traces events in your movie. Enlarge the window, as necessary, to read the messages.

The message Scramble function has been called appears in the Output window.



If the message did not appear, your next step would be to determine the reason why. The most likely reason is that you specified an incorrect path to the function.

8 Close the Output window and SWF file and return to authoring mode.

9 Choose File > Save As and enter a new filename. Use a consecutive naming scheme so you can revert to earlier versions of the file, if necessary.

You've made great progress in learning ActionScript!

The next steps

By completing the tutorial, you've accomplished a great deal in a relatively short amount of time. You've learned how to use ActionScript to set up a Flash movie's starting point, how to create commands and reuse code to make actions recur, and how to precisely control the flow of the movie. Using ActionScript variables and expressions, you know how to keep track of user activity in your movie and how to display changing content to your users. Finally, you've learned how to test your movie.

To continue learning about ActionScript, see the online ActionScript Dictionary in the Help menu and articles in the Flash Support Center. From the Samples folder within your Flash MX application folder, you can also open an advanced version of the puzzle.fla movie and deconstruct the ActionScript that was used to create a timer and animated puzzle piece.

CHAPTER 3
Introduction to Components Tutorial

You can use Macromedia Flash UI components to quickly and easily add simple user interface elements to your Flash document.

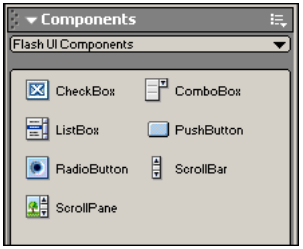
This tutorial is designed to introduce components to beginner and intermediate Flash users and show how they can be used to easily create a simple application. Before taking this tutorial, you should complete the Introduction to Flash MX Tutorial and the Introduction to ActionScript Tutorial, which you can access by selecting Help > Tutorials.

After completing this tutorial, you will know how to do the following tasks:

- Add components to a Flash document
- Configure the components
- Add ActionScript to make the components work

Types of components

Flash MX provides the following components:



Name	Action
Radio button	Represents a single choice in a group of mutually exclusive choices
Check box	Represents a single choice
Push button	Carries out a command when the user clicks it or presses Enter or Return
Combo box	Displays a list of choices
List box	Displays a list of choices
Scroll pane	Provides a scrollable window pane for viewing movie clips
Text scroll bar	Adds a scroll bar to a text field

How you can use components

Here are some ways you can use components:

Web form Design a form that asks for a user's address, telephone, e-mail address, and other personal information, then validates the data before submitting it to a server.

Build your car Create a Web form that allows users to order a car by selecting various graphic options instead of typing in the information. The fields are generated from the graphical interactions.

Survey Build a multiple-question, multiple-part survey that instantly tallies the results and charts the resulting survey data.

Photo album Create a personal photo album that takes a directory of images and thumbnails and wraps them with the Flash interface and navigation elements.

Loan calculator Design an online amortization calculator that calculates loan payments.

Web-based presentation template Create a template for slide-based presentations with simple, prebuilt navigation elements and a layout for images and text.

View the completed form

You can navigate through components in a form by doing the following:

- Click the component to select it.
- Press Tab to move forward; press Shift+Tab to move backward.
- Use the arrow keys to navigate through menu items.

1. Choose File > Open and navigate to the Flash MX program directory. Open Tutorials/Components/Finished/sweepstakes.swf. This is the completed movie.
2. Use the navigation techniques described above to test each of the buttons and boxes on the form.
3. Now open sweepstakes.fla. Choose File > Open and navigate to the Flash MX program directory. Open Tutorials/Components/Finished/sweepstakes.swf. This is the Flash document that generated the movie. Two keyframes are included in the Timeline.
4. Examine page 1. It contains the check box, combo box, and push buttons that are used to gather information. It also has two input text fields for the user's name and e-mail address.
5. Examine page 2. It is the results page and contains several dynamic text boxes to show the results of the information gathered on page 1. To save time, we have already created them for you.

Create a form

In this tutorial, you will use components to create a simple two-page form to enter a sweepstakes to win a free Stiletto electric car. The first page is used to gather the information, and the second page is used to display the results. You will follow these three steps to complete the form:

- Add components to the form
- Configure the components
- Add ActionScript to the document to make the components work

To get you started, we have included the background, text fields, and graphics for the form. You will add and configure components and ActionScript to make it an interactive form.

Add components

The first step is to add the components to the Stage and place them on the form. You will add a check box, a combo box, and a push button to the first page of the form. You will also add a push button to the second page.

To add components to a document, you can either drag elements from the Components panel to the Stage, or double-click them in the Components panel to place them in the center of the Stage. After you add a component to a document, it appears in the document's Library panel.

It is a good idea to create a new layer for the components.

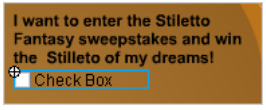
1. Choose File > Open and navigate to the Flash MX program directory. Open Tutorials/Components/my_sweepstakes/mysweepstakes.fla.
2. Choose File > Save As and save the file with a new name, such as newsweepstakes.
3. Create a new layer and name it UI. You will place the components on this layer.
4. Click Frame 6 in the UI layer of the Timeline. Choose Insert > Blank keyframe to add a blank keyframe. This will be used for components on the second page.
5. Make sure the following panels are open:

- Library panel (Window > Library)
- Components panel (Window > Components)
- Property inspector (Window > Properties)

Add a check box

Use the CheckBox component to create a box with a value of either true or false.

1. Select Frame 1 in the UI layer.
2. Drag the CheckBox component from the Components panel to the Stage. Place it under the paragraph that asks if the user wants to enter the sweepstakes.



The component appears in the Flash UI Components folder in the Library panel.

Add a combo box

Use the ComboBox component to create a simple drop-down menu of items that can be selected by users. You can also use a ComboBox to build a more complex drop-down menu that automatically scrolls to a menu item starting with the letter or letters entered into the text field by the user.

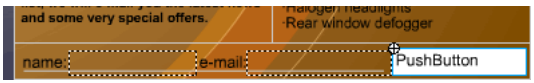
- Drag the ComboBox component from the Components panel to the Stage. Place it under "Select your favorite color:".

The component appears in the Flash UI Components folder in the Library panel.

Add push buttons

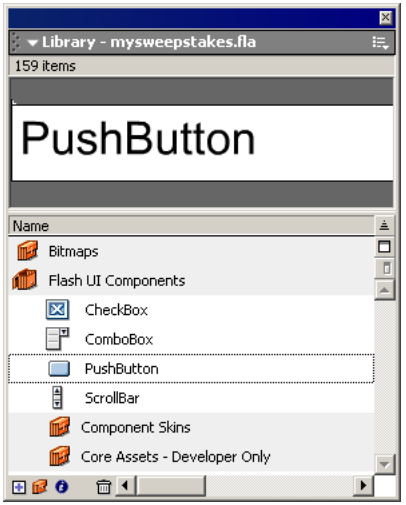
Use the PushButton component to create two push buttons. One button will be on the first page and will be used to submit the information on the form. The next button will be on the second page and will be used to return to the first page and repopulate it with the values that were previously submitted.

1 Drag the PushButton component from the Components panel to the Stage. Place it in the lower right corner of the form so it lines up horizontally with the name and e-mail text fields.

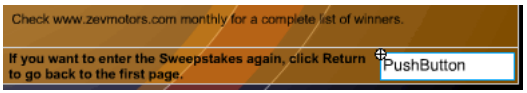


The component appears in the Flash UI Components folder in the Library panel.

After you have dragged a component from the Components panel to the Stage, you select additional instances of it from the Library panel.



2 Go to the blank keyframe at Frame 6. In the Library panel, open the Flash UI Components folder and drag the PushButton component from the Library panel to the Stage. Place it in the lower right corner.



Configure the components

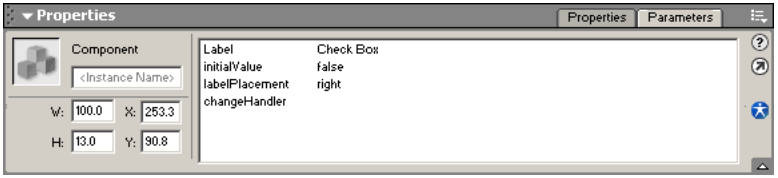
The next step is to configure the components so they are set up with the correct information for a user to select.

You set the parameters for a component using the Parameters tab of the Property inspector or the Components Parameters panel.

Advanced users can use API methods and properties for each component to set the parameters, size, appearance, and other properties of the component. The methods and properties available for each component element are listed in the ActionScript Dictionary under the name of the component.

Configure the check box

1 Select Frame 1 on the UI layer, then select the CheckBox component on the Stage. Its parameters are displayed in the Property inspector.



2 Type **sweepstakes_box** in the Instance Name text box.

3 Type **Absolutely!** in the Label text box.

4 In the Initial Value parameter pop-up menu, select True. This specifies whether the initial state of the CheckBox component is selected (True) or unselected (False).

5 In the Label Placement parameter pop-up menu, leave the default value set to right alignment. The label will be displayed to the right of the check box.

6 Do not alter the Change Handler parameter.

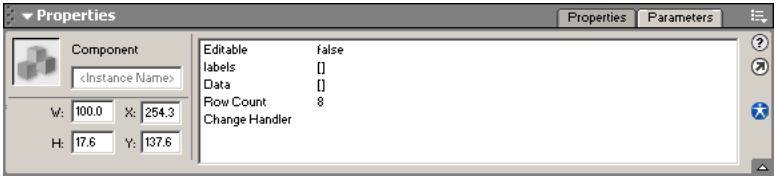
The Change Handler parameter is the function that you want to execute when the user selects an item. This function must be defined in the same Timeline as the component instance. This parameter is optional and needs to be specified only if you want an action to take place as soon as the user accesses a component.

When you finish, the Property inspector should look like the following:

For more information about using the API methods of the FCheckBox component to set additional options for the component, see the FCheckBox (component) entry in the online ActionScript Dictionary in Flash Help.

Configure the combo box

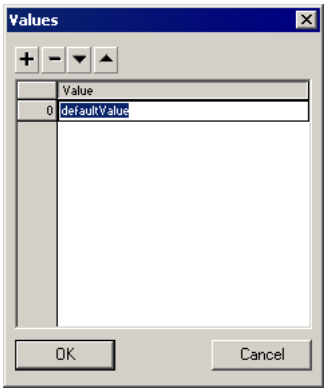
1 Select the ComboBox component on the Stage. Its parameters are displayed in the Property inspector.



2 Type **color_box** in the Instance Name text box.

3 Make sure the Editable parameter is set to False. This prevents users from typing in their own text.

4 The Labels parameter displays a list of values users can select. Click the Labels field, then click the magnifying glass to open the Values pop-up window. Click the Plus (+) button to enter a new value.

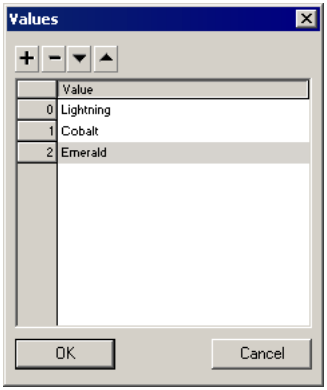


5 Click in the `default value` field, then type **Lightning** for the first value.

6 Click the Plus (+) button to enter the next value. Click in the `default value` field, then type **Cobalt** for the next value.

7 In the same manner, add **Emerald** to the list.

When you are finished, the Values pop-up window should look like the following:



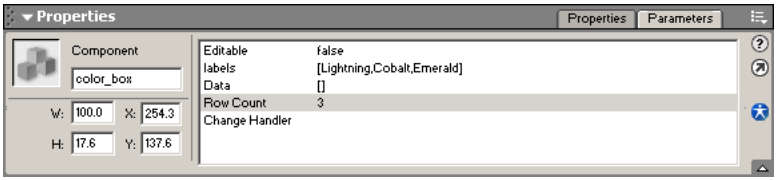
8 Click OK to close the Values pop-up window.

The Data parameter is optional. It is used to specify the values associated with the items (labels) in the list box. There is no need to do that for this tutorial.

9 The Row Count parameter specifies how many rows are displayed in the window. Since there are three options, change the value to 3.

There is no need to enter a Change Handler parameter name.

When you are finished, the Property inspector should look like the following:

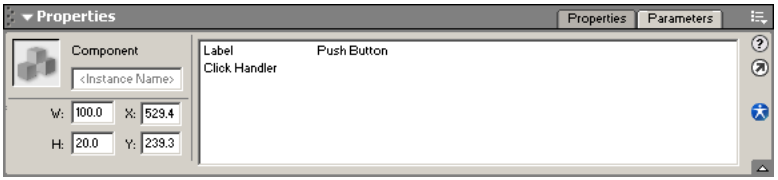


For more information about using API methods to change additional properties, see the FComboBox (component) entry in the online ActionScript Dictionary in Flash Help.

Configure the push buttons

1 Select the PushButton component in Frame 1.

The component's parameters are displayed in the Property inspector.



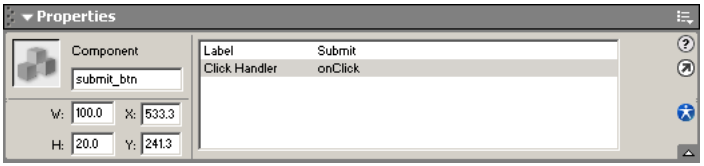
2 Type **submit_btn** in the Property inspector Instance Name text box.

3 Type **Submit** in the Property inspector Label text box.

This text is displayed in the body of the button.

4 Type **onClick** for the Click Handler name. Later you will write ActionScript to define what the Click Handler should do.

When you are finished, the Property inspector should look like the following:



5 Select the PushButton component in Frame 6.

6 Type **return_btn** in the Property inspector Instance Name text box.

7 Type **Return** in the Property inspector Label text box.

8 Type **onClick** for the Click Handler name.

For more information about using the API methods of the FPushButton component to change additional properties, see the FPushButton (component) entry in the online ActionScript Dictionary in Flash Help.

Write ActionScript to gather the data

Some of the ActionScript will be for the entire movie, while some will be related to a particular frame. The following table will help you keep track of instance names.

Instance name	Description
color_box	Color combo box on page 1
sweepstakes_box	Sweepstakes check box on page 1
submit_btn	Submit push button on page 1
name	Input text box for name on page 1
email	Input text box for e-mail address on page 1
return_btn	Return push button on page 2
name_result	Dynamic text box on page 2 to display the user's name
email_result	Dynamic text box on page 2 to display the user's e-mail address.
color_result	Dynamic text box on page 2 to display the user's color selection
sweepstakes_text	Dynamic text box on page 2 that displays different text if the user has chosen or not chosen to enter the sweepstakes

Write ActionScript for the entire movie

ActionScript for components is placed in keyframes. The Click Handler parameter specifies what happens when the PushButton component is activated. The default value is `onClick`, which means that when the user clicks one of the push buttons, it is activated. We will begin by creating a function for `onClick`. This will be a branching function that will first determine whether the button pushed was the Submit or the Return button, then carry out actions accordingly.

1. Create a new layer and name it **All Actions**. This will be used for ActionScript that should run throughout the movie.
2. If the Actions panel is not open, choose Window > Actions.
3. Switch to expert mode by pressing Control+Shift+E (Windows) or Command+Shift+E (Macintosh), or by clicking the control in the upper right corner (a triangle with a check mark above it) and selecting Expert Mode from the pop-up menu.

4 First, enter the callback function for the push buttons. This is a conditional statement that branches depending on which button is clicked. If the Submit button is clicked, it will branch to the `getResults` function and go to page 2. If the Return button is clicked, it will go to page 1.

Enter the following code in the Actions panel.

```
// push button callback
function onClick(btn) {
  if (btn == submit_btn) {
    getResults();
    gotoAndStop("pg2");
  } else if (btn == return_btn) {
    gotoAndStop("pg1");
  }
}
```

Note: Although it is not recommended, if you don't want to write the ActionScript, you can copy the text from this tutorial and paste it into the Actions panel.

5 Now, write the `getResults` function. This gets the results from the sweepstakes check box and the color combo box. It gets the results from the combo box as a label so it can show the results.

```
// get results from pg 1
function getResults() {
  sweepstakes_result = sweepstakes_box.getValue();
  color_result = color_box.getSelectedItem().label;
  selectedItem = color_box.getSelectedIndex();
}
```

6 Next, write the `initValues` function. This initializes the values on page 1 with the values the user has previously selected. It is run when the user clicks the Return button.

```
// initialize the values on pg 1 with the values the user has previously
  selected
function initValues() {
  sweepstakes_box.setValue(sweepstakes_result);
  if (!started) {
    color_box.setSelectedIndex(0);
    started = true;
  } else {
    color_box.setSelectedIndex(selectedItem);
  }
}
```

7 Finally, add a call to the `initValues` function to the beginning of the ActionScript. When you finish, the ActionScript should look like this:

```
initValues();
// push button callback
function onClick(btn) {
  if (btn == submit_btn) {
    getResults();
    gotoAndStop("pg2");
  } else if (btn == return_btn) {
    gotoAndStop("pg1");
  }
}
// initialize the values on pg 1 with the values the user has previously
  selected
function initValues() {
  sweepstakes_box.setValue(sweepstakes_result);
  if (!started) {
    color_box.setSelectedIndex(0);
    started = true;
  } else {
    color_box.setSelectedIndex(selectedItem);
  }
}
// get results from pg 1
function getResults() {
  sweepstakes_result = sweepstakes_box.getValue();
  color_result = color_box.getSelectedItem().label;
  selectedItem = color_box.getSelectedIndex();
}
```

You have completed the script that needs to run for the entire movie. However, you still need to add script for the frames in page 1 and page 2.

Add ActionScript to each keyframe

Some of the ActionScript needs to be executed only when a user selects a particular frame. In order to make the ActionScript work, you need to name each keyframe.

1 Create a new layer and name it Frame Actions. Select Frame 1, then choose Insert > Blank Keyframe to insert a keyframe. Use the Property inspector to name the keyframe **pg1**. In the same manner, insert a keyframe at Frame 6 and name it **pg2**.

2 Select the keyframe in Frame 1 of the Frame Actions layer and write the following ActionScript to stop the movie at Frame 1 (pg1):

```
stop();
```

3 (Optional) If you want to display certain text if the user has selected the sweepstakes check box and other text if the user has not, you can write a conditional statement with text that will go into the sweepstakes_text dynamic text field on page 2. Select the keyframe named pg2 in the Frame Actions layer and enter the following in the Actions panel:

```
// sweepstakes text
if (sweepstakes_result==true) {

  sweepstakes_text = "You have been entered in the Stiletto Fantasy
  sweepstakes. Winners are announced at the end of each month.";

} else {

  sweepstakes_text = "You have not been entered in the Stiletto Fantasy
  sweepstakes.";

}
```

Note: Do not cut and paste this ActionScript into the Actions panel. It will not work properly, because there are line breaks between the first and second lines of text.

Test your movie

To see what the components look like, run your movie in the Flash Player 6.

1 Select Control > Test Movie.

The movie is run in the Flash Player.

2 Select and deselect the check box to be sure it works.

3 Select a color and make sure it appears on page 2.

4 When you are finished, select File > Close to close the movie in the player.

5 If you noticed any errors, use the Arrow tool to select the component, then make the corrections in the Property inspector. If necessary, retest the movie.

The next steps

This tutorial has presented the basic steps for creating a simple group of components and then displaying the results. For more information, see Using Flash and the ActionScript Dictionary, both found in the Help menu.