Using Flash

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

INTRODUCTION
Getting Started

Macromedia Flash MX is the professional standard authoring tool for producing high-impact Web experiences. Whether you are creating animated logos, Web site navigation controls, long-form animations, entire Flash Web sites, or Web applications, you'll find the power and flexibility of Flash ideal for your own creativity.

System requirements for Flash authoring

The following hardware and software are required to author Flash movies:

- For Microsoft® Windows: An Intel Pentium 200 MHz or equivalent processor running Windows 98 SE, Windows ME, Windows NT 4.0, Windows 2000, or Windows XP; 64 MB of RAM (128 MB recommended); 85 MB of available disk space; a 16-bit color monitor capable of 1024 x 768 resolution; and a CD-ROM drive.
- For the Macintosh: A Power Macintosh running Mac OS 9.1 (or later) or Mac OS X version 10.1 (or later); 64 MB RAM free application memory (128 MB recommended), plus 85 MB of available disk space; a color monitor capable of displaying 16-bit (thousands of colors) at 1024 x 768 resolution; and a CD-ROM drive.

System requirements for the Flash Player

The following hardware and software are required to play Flash movies in a browser:

- Microsoft Windows 95, Windows 98, Windows ME, Windows NT 4.0, Windows 2000, or Windows XP or later; or a Macintosh PowerPC with System 8.6 or later (including OS X 10.1 or later).
- Netscape plug-in that works with Netscape 4 (or later) in Windows, or works with Netscape 4.5 (or later) or Internet Explorer 5.0 (or later) on the Mac OS.
- To run ActiveX controls, Microsoft® Internet Explorer 4 or later (Windows 95, Windows 98, Windows Me, Windows NT4, Windows 2000, Windows XP, or later).
- AOL 7 on Windows, AOL 5 on the Mac OS
- Opera 6 on Windows, Opera 5 on the Mac OS

Installing Flash

Follow these steps to install Flash on either a Windows or a Macintosh computer.

To install Flash on a Windows or a Macintosh computer:

1. Insert the Flash CD into the computer's CD-ROM drive.
2. Do one of the following:

- In Windows, choose Start > Run. Click Browse and choose the Flash MX Installer.exe file on the Flash MX CD. Click OK in the Run dialog box to begin the installation.
- On the Macintosh, double-click the Flash MX Installer icon.

3. Follow the onscreen instructions.
4. If prompted, restart your computer.

What's new in Flash MX

New features in Flash MX enhance the approachability, creativity, and power of Flash. Designers who require a higher level of control and integration with industry-standard design tools now have an unparalleled creative application for creating media-rich content.

Powerful new features build on this creativity, giving application developers access to new capabilities that make Flash MX a robust and exciting application development environment. Developers can work with advanced scripting and debugging tools, built-in code reference, and predefined components to rapidly deploy rich Web applications.

For all Flash users

The ability to save Flash MX documents in Flash 5 format lets you upgrade now and still collaborate with designers who are working on Flash 5 projects. See "Saving Flash documents" on page 43.

Accessible content that can be seen and heard by persons with disabilities is now easy to develop, expanding the audience for Flash movies and applications. See "About accessible objects in Flash movies" under Help > Using Flash.

Korean and Chinese language support reaches audiences in more of the world. Features like vertical text fields and Unicode support make it easy to create Asian-language content. See "Creating text" on page 136.

For the designer

Flash MX enhances creativity by providing designers with a higher level of control and expanded integration capabilities with a rich set of design tools. New features help designers quickly create a broad range of content. Instead of focusing on how Flash works, they can give more attention to their designs.

Timeline enhancements such as folders for organizing layers, improved pointer feedback, and the ability to resize, cut, and paste multiple frames make it easier to use the Timeline, helping you work faster and with less effort. See "Using the Timeline" on page 28.

Enhanced editing of symbols in place makes document creation easier by letting designers work on symbols in the context of their movies. New controls above the Stage make it easier than ever to edit symbols in place. See "Editing symbols" on page 157.

Library improvements eliminate production bottlenecks by simplifying the creation and manipulation of library symbols. Moving symbols or folders between Flash documents or creating new library symbols is now as easy as dragging and dropping. See "Working with common libraries" on page 58. The new Resolve Library Conflict dialog box simplifies adding library symbols to a document that has an existing library symbol with the same name. See "Resolving conflicts between library assets" on page 168.

Shared library assets improve Flash movie authoring by letting you share library assets with other Flash documents, either while authoring, or when a movie is played with the Flash Player. Shared runtime libraries help you create smaller files and easily make updates to multiple documents simultaneously by letting your document show library symbols and shared objects that are stored on an intranet or the Internet. Shared author-time libraries improve your work pace by letting you track, update, and swap symbols in any Flash document available on your computer or network. See "Using shared library assets" on page 165.

Workspace enhancements make the Flash MX workspace more manageable and easier to understand for new and veteran designers. The most commonly used features now appear in one context-sensitive Property inspector, eliminating the need to access many other windows, panels, and dialog boxes. See "Panels and the Property inspector" on page 48. Other frequently used features now appear in easily collapsible panels that dock and undock as necessary to conserve screen space. Designers can even save custom panel layouts to personalize their Flash workspace. See "Using panels" on page 48.

New starter templates included with Flash MX simplify the creation of new documents by eliminating many of the common tasks required to start a new document. See "Creating a new document" on page 21. You can also create your own templates from documents. See "Saving Flash documents" on page 43.

Color Mixer improvements make creating, editing, and using colors and gradients easier than ever. See "Working with solid colors and gradient fills in the Color Mixer" on page 80.

Complete lessons that address the new features in Flash MX make it easy to become familiar with its powerful tools and features. To get started with the lessons, choose Help > Lessons > Getting Started with Flash.

Video support expands the creative possibilities for Flash movies by letting you import video clips in a variety of formats. See "Importing video" under Help > Using Flash.

The Free Transform tool opens new possibilities for your creative expression by letting you combine the effects of multiple object transformations at once. See "Transforming objects freely" on page 126.

The Envelope modifier lets you easily create otherwise-difficult graphic objects by letting you warp and distort the shape of the bounding box that surrounds them. See "Modifying shapes with the Envelope modifier" on page 128.

Pixel-level editing adds precision and polish to your work by letting you align objects with pixel-level precision in your Flash documents. Precisely place objects or points of objects where you want them to appear in your final movie. See "Pixel snapping" on page 74.

The Break Apart feature makes it easy to make creative edits to individual text characters without having to convert the text to symbols, simplifying the creation of complex designs and animation. See "About transforming text" on page 144.

The Distribute to Layers command quickly and automatically distributes any number of selected objects to their own layers. See "Distributing objects to layers for tweened animation" on page 172.

Movie clip mask layers let you create animated masks by placing a movie clip on a mask layer. See "Using mask layers" on page 183. You can also use ActionScript to create an animated mask with a movie clip. See "Using movie clips as masks" under Help > Using Flash.

Enhanced sound controls enhance the production quality of your movies by letting you synchronize movie events with the start or end of sound clips. See "About the onSoundComplete event" under Help > Using Flash.

For the developer

The powerful Flash MX environment includes enhanced scripting and debugging tools, built-in code reference, and predefined components you can use to rapidly develop rich Web applications.

Enhanced ActionScript gives you the ability to dynamically load JPEG and MP3 sound files at runtime, and lets you update your files at any time without having to republish your movie.See "Placing artwork into Flash" and "Importing sounds" under Help > Using Flash.See "Placing artwork into Flash" and "Importing sounds" under Help > Using Flash.

Anchor points enhance navigation in Flash movies by letting users use the Forward and Back buttons in their browsers to jump from anchor to anchor. See "Using named anchors" on page 33.

The improved ActionScript editor makes it easier for new and veteran authors to access the full potential of ActionScript. See "About scripting in ActionScript" on page 204.

Code hints speed content development of ActionScript by automatically detecting what command the user is typing and offering hints to reveal the exact syntax of the command. See "Using code hints" under Help > Using Flash.

Flash components accelerate Web application development by providing reusable drag-and-drop interface elements for Flash content, such as list boxes, radio buttons, and scroll bars. See Chapter 15, "Using Components," on page 289.

The improved debugger combines the debugging capabilities already in existence with an ActionScript debugger by allowing you to set breakpoints and single-step through the code as it executes. See "Testing a movie" under Help > Using Flash.

The object model integrates movie clips, buttons, and text fields into the ActionScript object-oriented scripting language. See "Working with Movie Clips and Buttons" and "Controlling text with ActionScript" under Help > Using Flash.

The event model makes ActionScript event handling more powerful and easier to understand. The event model now allows for more sophisticated control over user events, such as mouse movement and keyboard input. See "Controlling when ActionScript runs" on page 207.

The Live Preview feature for components makes it possible to actively view changes in user interface components within the authoring environment. See "Working with components in Live Preview" on page 293.

Enhanced text support allows for detailed control using ActionScript over every property of a text object, including its formatting, size, and layout. See Chapter 8, "Working with Text," on page 135.

The new drawing API enhances the object-oriented programming power of ActionScript by offering a set of shape-drawing capabilities through the MovieClip object, allowing for programmatic control over the Flash rendering engine. See "About the MovieClip object" on page 206.

Strict equality and switch statements allow for concise definition of flow control statements such as if, then, and else, further increasing ActionScript support for ECMA-262. See the entries for these statements in the online ActionScript Dictionary in the Help menu.

SetInterval and clearInterval functions allow designers to set up a generic routine that will be called at periodic intervals throughout the lifetime of a movie. See the entries for these functions in the online ActionScript Dictionary in the Help menu.

Conversion of String, Array, and XML objects to native objects increases performance by optimizing the Number, Boolean, Object, String, Array, and XML ActionScript objects. Performance in the Flash Player is increased as much as 100 times. See the entries for these objects in the online ActionScript Dictionary in the Help menu.

SWF compression uses existing Z-lib compression code to improve download times for complex Flash content. See Chapter 20, "Publishing," on page 365.

Guide to instructional media

The Flash package contains a variety of media to help you learn the program quickly and become proficient in creating your own Flash movies. These media include a printed manual, an expanded electronic version of the manual, online help that appears in your Web browser, a built-in ActionScript Reference panel, interactive lessons, and a regularly updated Web site. In addition, there are many third-party resources available to Flash designers and developers.

About the printed and electronic manuals

Information that appears in the printed version of *Using Flash* is primarily intended for users who are in their first three to six months of learning Flash. Online lessons and tutorials supplement this information.

The electronic version of *Using Flash* contains all of the information in the printed version, as well as additional chapters with instructions and information for using Flash tools and commands. It also includes chapters on ActionScript, which explain how to write and create interactions with the Flash scripting language.

Using Flash Help

Flash Help contains two main sections: Using Flash and the ActionScript Dictionary. For the best experience with Flash Help, Macromedia strongly recommends that you use a browser with Java player support, such as Internet Explorer 4.5 or later. Flash Help also supports Netscape Navigator 6.1 or later on Windows and Macintosh. Running Flash and Flash Help simultaneously on a Macintosh may require up to 32 MB of memory, depending on your browser's memory needs.

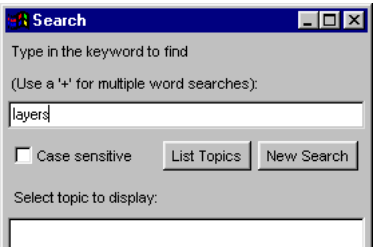
Note: The first time you access Flash Help when running Windows XP, you may be prompted to install the Java player. Follow the onscreen instructions to install the Java player.

To use Flash Help:

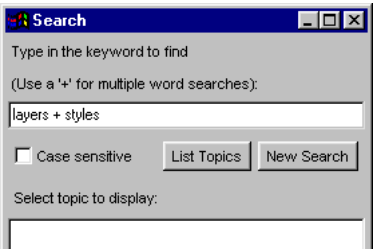
1. Choose one of the help systems from the Help menu.
2. Navigate the help topics using any of these features:

- Contents organizes information by subject. Click top-level entries to view subtopics.
- Index organizes information like a traditional printed index. Click a term to jump to a related topic.

- Search finds any character string in all topic text. Search requires a 6.1 or later browser with Java enabled. To search for a phrase, type it into the text entry box.



To search for files that contain two keywords (for example, **layers** and **style**), separate the words with a plus (+) sign.



To search for files that contain a complete phrase, separate the words with a space.

- Previous and Next buttons let you move through the topics within a section.

- The Flash icon links you to the Flash Support Center Web site.

Using Flash lessons and tutorials

Flash lessons provide quick interactive instruction that introduces you to the main features of Flash, letting you practice on isolated examples. If you are new to Flash, or if you have used only a limited set of its features, start with the lessons.

Flash tutorials provide in-depth interactive instruction that helps you familiarize yourself with Flash and provides detailed instruction on some powerful Flash features.

The Introduction to Flash MX Tutorial introduces the workflow in Flash by showing you how to create a basic movie. The tutorial assumes an understanding of the topics covered in the lessons.

The Introduction to ActionScript Tutorial teaches you the basic principles of ActionScript, the object-oriented language Flash uses to add interactivity to movies.

The Introduction to Components Tutorial is designed to introduce components to beginner and intermediate Flash users and show how they can be used to quickly create a simple application. Before taking this tutorial, you should complete the Flash lessons, the Introduction to Flash MX tutorial, and the Introduction to ActionScript tutorial or be familiar with ActionScript.

To start the lessons:

Choose Help > Lessons > Getting Started with Flash.

To start a tutorial, do one of the following:

- Choose Help > Tutorials > Introduction to Flash MX.
- Choose Help > Tutorials > Introduction to ActionScript.
- Choose Help > Tutorials > Introduction to Components.

Using additional Macromedia resources

The Flash Support Center Web site is updated regularly with the latest information on Flash, plus advice from expert users, advanced topics, examples, tips, and other updates. Check the Web site often for the latest news on Flash and how to get the most out of the program at www.macromedia.com/support/flash. Check the Web site often for the latest news on Flash and how to get the most out of the program at www.macromedia.com/support/flash.

The ActionScript Reference panel provides detailed information on ActionScript syntax and usage. The hierarchical structure of the information lets you easily scroll down to the specific information you need.

To display the ActionScript Reference panel:

Choose Window > Reference.

Third-party resources

Macromedia recommends several Web sites with links to third-party resources on Flash.

Macromedia Flash community sites:

www.macromedia.com/support/flash/ts/documents/flash_websites.htm

www.macromedia.com/support/flash/ts/documents/tn4148-flashmaillists.html

Macromedia Flash books:

www.macromedia.com/software/flash/productinfo/books/

Object-oriented programming concepts:

http://java.sun.com/docs/books/tutorial/java/concepts

Launching Flash on a network

If you encounter a license infringement warning message when launching Flash, you may have exceeded the number of licensed copies for that serial number.

Flash detects unauthorized copies of itself (under the same serial number) on a local area network. By enumerating currently running copies of Flash through network communication, Flash detects if the number of copies currently running exceeds a license count for the serial number.

To prevent license infringement warnings, do one of the following:

- Purchase additional licensed copies of Flash from Macromedia.
- Uninstall Flash from one or more computers on your local area network, then launch Flash again on your computer.

CHAPTER 1
Working in Flash

Macromedia Flash MX movies are graphics, text, animation, and applications for Web sites. They consist primarily of vector graphics, but they can also contain imported video, bitmap graphics, and sounds. Flash movies can incorporate interactivity to permit input from viewers, and you can create nonlinear movies that can interact with other Web applications. Web designers use Flash to create navigation controls, animated logos, long-form animations with synchronized sound, and even complete, sensory-rich Web sites. Flash movies use compact vector graphics, so they download rapidly and scale to the viewer's screen size.

You've probably watched and interacted with Flash movies on many Web sites. Millions of Web users have received the Flash Player with their computers, browsers, or system software; others have downloaded it from the Macromedia Web site. The Flash Player resides on the local computer, where it plays back movies in browsers or as stand-alone applications. Viewing a Flash movie on the Flash Player is similar to viewing a DVD on a DVD player—the Flash Player is the device used to display the movies you create in the Flash authoring application.

Flash documents, which have the .fla filename extension, contain all the information required to develop, design, and test interactive content. Flash documents are not the movies the Flash Player displays. Instead, you publish your FLA documents as Flash movies, which have the .swf filename extension and contain only the information needed to display the movie.

For an interactive introduction to Flash, choose Help > Lessons > Getting Started with Flash.

Artwork in Flash

Flash provides a variety of methods for creating original artwork and importing artwork from other applications. You can create objects with the drawing and painting tools, as well as modify the attributes of existing objects. See Chapter 3, "Drawing," on page 59 and Chapter 4, "Working with Color," on page 77.

You can also import vector graphics, bitmap graphics, and video from other applications and modify the imported graphics in Flash.See "Using Imported Artwork and Video" under Help > Using Flash.

Note: You can also import sound files, as described in "Importing sounds" under Help › Using Flash.

Animation in Flash

Using Flash, you can animate objects to make them appear to move across the Stage and/or change their shape, size, color, opacity, rotation, and other properties. You can create frame-by-frame animation, in which you create a separate image for each frame. You can also create tweened animation, in which you create the first and last frames of an animation and direct Flash to create the frames in between. See Chapter 10, "Creating Animation," on page 169.

You can also use ActionScript, an object-oriented programming language, to create animation in Flash. See Chapter 12, “Understanding the ActionScript Language,” on page 203.

Interactive movies in Flash

Flash lets you create interactive movies, in which your audience can use the keyboard or the mouse to jump to different parts of a movie, move objects, enter information in forms, and perform many other operations.

You create interactive movies by scripting actions using ActionScript. For more information, see Chapter 14, “Creating Interaction with ActionScript,” on page 267. For complete information on using ActionScript to create advanced interactivity, see the online ActionScript Dictionary in the Help menu.

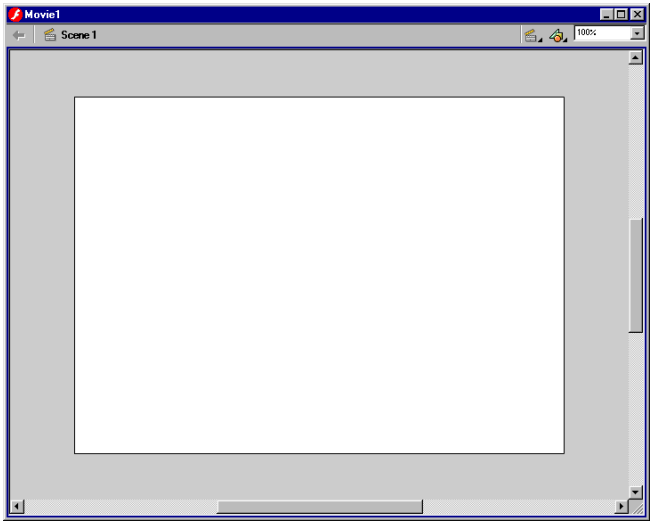
Application development in Flash

Flash provides movie clips with defined parameters, called components, to aid in developing rich user experiences in Flash movies. Each built-in Flash component has its own unique set of ActionScript methods that allow you to set and change the authoring parameters and additional options at runtime. By combining the easy drop-in capabilities of the predefined components with the powerful capabilities of ActionScript, you can create fully functional applications on the Web. For more information on components, see Chapter 15, “Using Components,” on page 289.

For an interactive introduction to components, choose Help > Tutorials > Introduction to Components.

The Stage and workspace

Like films, Flash movies divide lengths of time into frames. The Stage is where you compose the content for individual frames in the movie, drawing artwork on it directly or arranging imported artwork on it. For more information on frames, see “Using frames and keyframes” on page 31.



The Stage is where you compose individual frames in a movie.

Viewing the Stage

You can change your view of the Stage by changing the magnification level or moving the Stage within the Flash work environment. You can also adjust your view of the Stage using the View commands.

Zooming

To view the entire Stage on the screen, or to view just a particular area of your drawing at high magnification, you can change the magnification level. The maximum magnification depends on the resolution of your monitor and the document size.

To magnify or reduce your view of the Stage, do one of the following:

- To zoom in on a certain element, select the Zoom tool and click the element. To switch the Zoom tool between zooming in or out, use the Enlarge or Reduce modifiers or Alt-click (Windows) or Option-click (Macintosh).

- To zoom in on a specific area of your drawing, drag a rectangular selection marquee with the Zoom tool. Flash sets the magnification level so that the specified rectangle fills the window.
- To zoom in on or out of the entire Stage, choose View > Zoom In or View > Zoom Out.
- To zoom in or out by a specified percentage, choose View > Magnification and select a percentage from the submenu, or select a percentage from the Zoom control at the lower left corner of the application window.
- To display the contents of the current frame, choose View > Magnification > Show All, or choose Show All from the Zoom control at the lower left corner of the application window. If the scene is empty, the entire Stage is displayed.
- To display the entire Stage, choose View > Magnification > Show Frame or choose Show Frame from the Zoom control at the lower left corner of the application window.
- To display the work area surrounding the Stage, choose View > Work Area. The work area is shown in light gray. Use the Work Area command to view elements in a scene that are partly or completely outside of the Stage. For example, to have a bird fly into a frame, you would initially position the bird outside of the Stage in the work area.

Moving the view of the Stage

When the Stage is magnified, you may not be able to see all of it. The Hand tool lets you move the Stage to change the view without having to change the magnification.

To move the Stage view:

1. In the toolbox, select the Hand tool. To temporarily switch between another tool and the Hand tool, hold down the Spacebar and click the tool in the toolbox.
2. Drag the Stage.

Using the grid, guides, and rulers

Flash comes with rulers and guides that help you draw and lay out objects precisely. You can place guides in a document and snap objects to those guides, or turn on the grid and snap objects to it.

Using the grid

When the grid is displayed in a document, it appears as a set of lines behind the artwork in all scenes. You can snap objects to the grid, and you can modify the grid size and grid line color.

To display or hide the drawing grid:

Choose View > Grid > Show Grid.

To turn snapping to grid lines on or off:

Choose View > Grid > Snap to Grid.

To set grid preferences:

1. Choose View > Grid > Edit Grid.
2. For Color, click the triangle in the color box and select a grid line color from the palette.

   The default grid line color is gray.
3. Select or deselect Show Grid to display or hide the grid.
4. Select or deselect Snap to Grid to turn snapping to grid lines on or off.
5. For grid spacing, enter values in the text boxes to the right of the horizontal and vertical arrows.
6. For Snap Accuracy, select an option from the pop-up menu.
7. If you want to save the current settings as the default, click Save Default.

Using guides

You can drag horizontal and vertical guides from the rulers onto the Stage when the rulers are displayed. You can move guides, lock guides, hide guides, and remove guides. You can also snap objects to guides, and change guide color and snap tolerance (how close objects must be to snap to a guide). Draggable guides appear only in the Timeline in which they were created.

To create custom guides or irregular guides, you use guide layers. See "Using guide layers" on page 38.

To display or hide the drawing guides:

Choose View > Guides > Show Guides.

Note: If the grid is visible and Snap to Grid is turned on when you create guides, guides will snap to the grid.

To turn snapping to guides on or off:

Choose View > Guides > Snap to Guides.

Note: Snapping to guides takes precedence over snapping to the grid in places where guides fall between grid lines.

To move a guide:

Use the Arrow tool to drag the guide.

To remove a guide:

With guides unlocked, use the Arrow tool to drag the guide to the horizontal or vertical ruler. For information on locking and unlocking guides, see the following procedure.

To set guide preferences:

1. Choose View > Guides > Edit Guides.
2. For Color, click the triangle in the color box and select a guide line color from the palette.

   The default guide color is green.
3. Select or deselect Show Guides to display or hide guides.
4. Select or deselect Snap to Guides to turn snapping to guides on or off.
5. Select or deselect Lock Guides to lock or unlock guides.
6. For Snap Accuracy, select an option from the pop-up menu.
7. If you want to remove all guides, click Clear All.

   Note: Clear All removes all guides from the current scene.
8. If you want to save the current settings as the default, click Save Default.

Using rulers

When rulers are displayed, they appear along the top and left sides of the document. You can change the unit of measure used in the rulers from the default of pixels. When you move an element on the Stage with the rulers displayed, lines indicating the element's dimensions appear on the rulers.

To display or hide rulers:

Choose View > Rulers.

To specify the rulers' unit of measure for a document:

Choose Modify > Document, and then select an option from the pop-up menu at the upper right.

Creating a new document

Each time you open Flash, the application creates a new file with the FLA extension. You can create additional new Flash documents as you work. To set the size, frame rate, background color, and other properties of a new document, you use the Document Properties dialog box.

You can also open a template as a new document. You can choose from standard templates that ship with Flash, or open a template you have saved previously. For information on saving a document file as a template, see "Saving Flash documents" on page 43.

To create a new document and set its properties:

1. Choose File > New.
2. Choose Modify > Document.

   The Document Properties dialog box appears.

3 For Frame Rate, enter the number of animation frames to be displayed every second. For most computer-displayed animations, especially those playing from a Web site, 8 fps (frames per second) to 12 fps is sufficient. (12 fps is the default frame rate.)

4 For Dimensions, do one of the following:

- To specify the Stage size in pixels, enter values in the Width and Height text boxes.

  The default movie size is 550 x 400 pixels. The minimum size is 1 x 1 pixels; the maximum is 2880 x 2880 pixels.

- To set the Stage size so that there is equal space around the content on all sides, click the Contents button to the right of Match. To minimize movie size, align all elements to the upper left corner of the Stage, and then click Contents.

- To set the Stage size to the maximum available print area, click Printer. This area is determined by the paper size minus the current margin selected in the Margins area of the Page Setup dialog box (Windows) or the Print Margins dialog box (Macintosh).

- To set the Stage size to the default size, click Default.

5 To set the background color of your movie, click the triangle in the Background Color box and select a color from the palette.

6 To specify the unit of measure for rulers that you can display along the top and side of the application window, select an option from the pop-up menu in the upper right. See "Using rulers" on page 21. (This setting also determines the units used in the Info panel.)

7 Do one of the following:

- To make the new settings the default properties for your new document only, click OK.

- To make these settings the default properties for all new documents, click Make Default.

To open a template as a new document:

1 Choose File > New from Template.

2 In the New Document dialog box, select a category from the Category list, and select a document from the Category Items list.

3 Click OK.

Setting preferences in Flash

Flash lets you set preferences for general application operations, editing operations, and Clipboard operations. See also "Choosing drawing settings" on page 75.

To set preferences:

1 Choose Edit > Preferences.

2 Click the General, Editing, Clipboard, Warning, or ActionScript Editor tab, and choose from the respective options as described in the procedures that follow. For more information on ActionScript Editor preferences, see "Setting Actions panel preferences" under Help > Using Flash.

To set general preferences, choose from the following options:

- For Undo Levels, enter a value from 0 to 200 to set the number of undo/redo levels. Undo levels require memory; the more undo levels you use, the more system memory is taken up. The default is 100.
- For Printing Options (Windows only), select Disable PostScript to disable PostScript output when printing to a PostScript printer. By default, this option is deselected. Select this option if you have problems printing to a PostScript printer, but keep in mind that this will slow down printing.
- For Selection Options, select or deselect Shift Select to control how Flash handles selection of multiple elements. When Shift Select is off, clicking additional elements adds them to the current selection. When Shift Select is on, clicking additional elements deselects other elements unless you hold down the Shift key.
- Select Show Tooltips to display tooltips when the pointer pauses over a control. Deselect this option if you don't want to see the tooltips.
- For Timeline Options, select Disable Timeline Docking to keep the Timeline from attaching itself to the application window once it has been separated into its own window. For more information, see "Using the Timeline" on page 28.
- Select Span Based Selection to use span-based selection in the Timeline, rather than the default frame-based selection (Flash 5 used span-based selection). For more information on span-based and frame-based selection, see "Working with frames in the Timeline" on page 31.
- Select Named Anchor on Scenes to have Flash make the first frame of each scene in a movie a named anchor. Named anchors let you use the Forward and Back buttons in a browser to jump from scene to scene in a movie. For more information, see "Using named anchors" on page 33.
- For Highlight Color, select Use This Color and select a color from the palette, or select Use Layer Color to use the current layer's outline color.
- For Font Mapping Default, select a font to use when substituting missing fonts in movies you open in Flash. See "Substituting missing fonts" on page 145.

To set editing preferences, choose from the following options:

- For Pen Tool options, see "Setting Pen tool preferences" on page 64.
- For Vertical Text options, select Default Text Orientation to make the default orientation of text vertical, which is useful for some Asian language fonts. By default, this option is deselected.
- Select Right to Left Text Flow to reverse the default text display direction. This option is deselected by default.
- Select No Kerning to turn off kerning for vertical text. This option is deselected by default, but is useful to improve spacing for some fonts that use kerning tables.
- For Drawing Settings, see "Choosing drawing settings" on page 75.

To set Clipboard preferences, choose from the following options:

- For Bitmaps (Windows only), select options for Color Depth and Resolution to specify these parameters for bitmaps copied to the Clipboard. Select Smooth to apply anti-aliasing. Enter a value in the Size Limit text box to specify the amount of RAM that is used when placing a bitmap image on the Clipboard. Increase this value when working with large or high-resolution bitmap images. If your computer has limited memory, choose None.
- For Gradients (Windows only), choose an option to specify the quality of gradient fills placed in the Windows Metafile. Choosing a higher quality increases the time required to copy artwork. Use this setting to specify gradient quality when pasting items to a location outside of Flash. When you are pasting within Flash, the full gradient quality of the copied data is preserved regardless of the Gradients on Clipboard setting.
- For PICT Settings (Macintosh only), for Type, select Objects to preserve data copied to the Clipboard as vector artwork, or select one of the bitmap formats to convert the copied artwork to a bitmap. Enter a value for Resolution. Select Include PostScript to include PostScript data. For Gradients, choose an option to specify gradient quality in the PICT. Choosing a higher quality increases the time required to copy artwork. Use the Gradients setting to specify gradient quality when pasting items to a location outside of Flash. When you are pasting within Flash, the full gradient quality of the copied data is preserved regardless of the Gradient setting.
- For FreeHand Text, select Maintain Text as Blocks to keep text editable in a pasted FreeHand file.

To set warning preferences, choose one of the following options:

- Select Warn on Save for Macromedia Flash 5 Compatibility to have Flash warn you when you try to save documents with Flash MX–specific content to a Flash 5 file. This option is selected by default.
- Select Warn on Missing Fonts to have Flash warn you when you open a Flash document that uses fonts that are not installed on your computer. This option is selected by default.
- Select Warn on Loss of Expert Mode Formatting to have Flash warn you of any expert mode formatting that will be lost when you switch to normal mode in the Actions panel. This option is selected by default.
- Select Warn on Reading Generator Content to have Flash display a red "X" over any Generator objects, as a reminder that Generator objects are not supported in Flash MX.
- Select Warn on Inserting Frames when Importing Content to have Flash alert you when it inserts frames in your document to accommodate audio or video files that you import.

Using the Property inspector to change document attributes

The Property inspector makes it easy to access and change the most commonly used attributes of a document. You can make changes to document attributes in the Property inspector without accessing the menus or panels that contain these features. For more information on the Property inspector, see "Panels and the Property inspector" on page 48.

To change document properties with the Property inspector:

1. Deselect all assets, then select the Pointer tool.
2. If the Property inspector is not visible, choose Window > Properties.

3 Click the Size control to display the Document Properties dialog box and access its settings. For more information on the Document Properties dialog box, see "Creating a new document" on page 21.

4 To choose a background color, click the triangle in the Background color box and select a color from the palette.

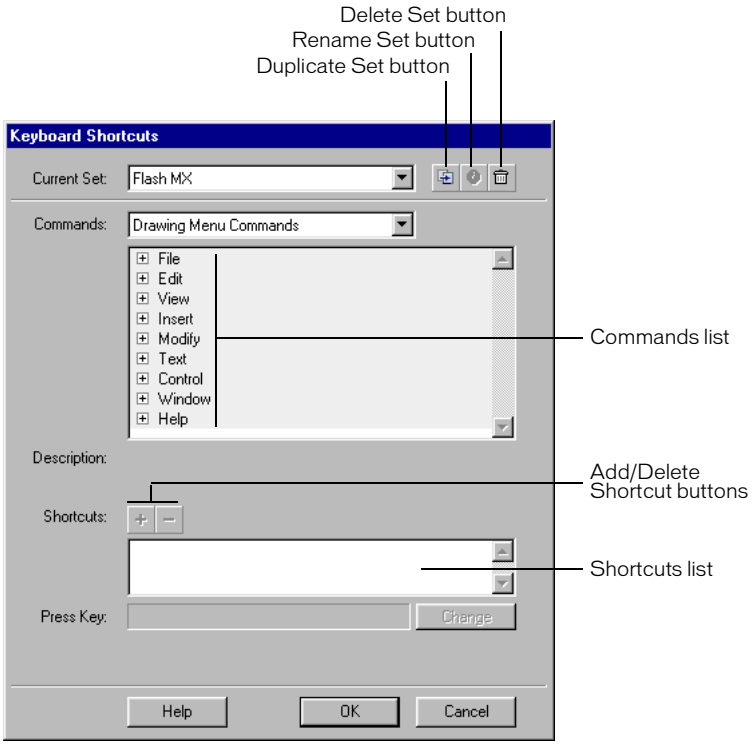
5 For Frame Rate, enter the number of animation frames to be displayed every second.

6 Click the Publish control to display the Publish Settings dialog box with the Flash tab selected. For more information on the Publish Settings dialog box, see "Publishing Flash documents" on page 367.

Customizing keyboard shortcuts

You can choose keyboard shortcuts in Flash to match the shortcuts you use in other applications, or to streamline your Flash workflow. By default, Flash uses built-in keyboard shortcuts designed for the Flash application. You can also select a built-in keyboard shortcut set from one of several popular graphics applications, including Fireworks, Adobe Illustrator, and Adobe Photoshop.

To create a custom keyboard shortcut set, you duplicate an existing set, and then add or remove shortcuts from the new set. You can delete custom shortcut sets.



To select a keyboard shortcut set:

1. Choose Edit > Keyboard Shortcuts.
2. In the Keyboard Shortcuts dialog box, choose a shortcut set from the Current Set pop-up menu.

To create a new keyboard shortcut set:

1. Select a keyboard shortcut set as described above.
2. Click the Duplicate Set button.
3. Enter a name for the new shortcut set and click OK.

To rename a custom keyboard shortcut set:

1. In the Keyboard Shortcuts dialog box, choose a shortcut set from the Current Set pop-up menu.
2. Click the Rename Set button.
3. In the Rename dialog box, enter a new name and click OK.

To add or remove a keyboard shortcut:

1. Choose Edit > Keyboard Shortcuts and select the set that you want to modify.
2. From the Commands pop-up menu, select Drawing Menu Commands, Drawing Tools, or Test Movie Menu Commands to view shortcuts for the selected category.
3. In the Commands list, select the command for which you want to add or remove a shortcut.

   An explanation of the selected command appears in the Description area in the dialog box.
4. Do one of the following:

- To add a shortcut, click the Add Shortcut (+) button.
- To remove a shortcut, click the Remove Shortcut (-) button and proceed to step 6.

5. If you are adding a shortcut, enter the new shortcut key combination in the Press Key text box.

   Note: To enter the key combination, simply press the keys on the keyboard. You do not need to spell out key names, such as Control, Option, and so on.
6. Click Change.
7. Repeat this procedure to add or remove additional shortcuts.
8. Click OK.

To delete a keyboard shortcut set:

1. Choose Edit > Keyboard Shortcuts. In the Keyboard Shortcuts dialog box, click the Delete Set button.
2. In the Delete Set dialog box, choose a shortcut set and click Delete.

   Note: You cannot delete the built-in keyboard shortcut sets that ship with Flash.

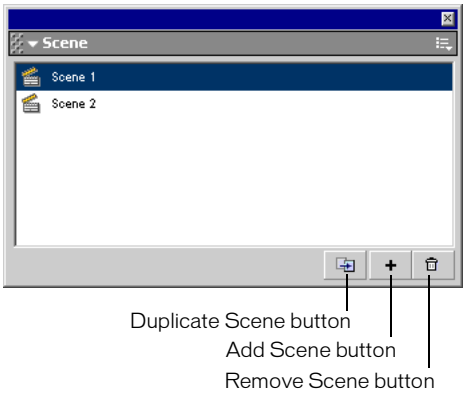
Using scenes and the Scene panel

To organize a movie thematically, you can use scenes. For example, you might use separate scenes for an introduction, a loading message, and credits.

When you publish a Flash movie that contains more than one scene, the scenes in the movie play back in the order they are listed in the Scene panel in the Flash document. Frames in the movie are numbered consecutively through scenes. For example, if a movie contains two scenes with ten frames each, the frames in Scene 2 are numbered 11–20.

You can add, delete, duplicate, rename, and change the order of scenes.

To stop or pause a movie after each scene, or to let users navigate the movie in a nonlinear fashion, you use actions. See Chapter 14, “Creating Interaction with ActionScript,” on page 267.



Scene panel

To display the Scene panel:

Choose Window > Scene.

To view a particular scene:

Choose View > Go To and then choose the name of the scene from the submenu.

To add a scene, do one of the following:

- Click the Add Scene button in the Scene panel.
- Choose Insert > Scene.

To delete a scene, do one of the following:

- Click the Delete Scene button in the Scene panel.
- Open the scene you want to delete and choose Insert > Remove Scene.

To change the name of a scene:

Double-click the scene name in the Scene panel and enter the new name.

To duplicate a scene:

Click the Duplicate Scene button in the Scene panel.

To change the order of a scene in the movie:

Drag the scene name to a different location in the Scene panel.

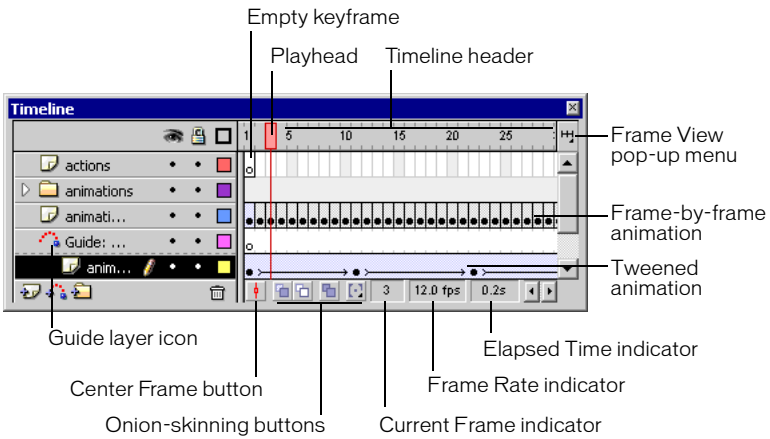
Using the Timeline

The Timeline organizes and controls a movie's content over time in layers and frames. Like films, Flash movies divide lengths of time into frames. Layers are like multiple film strips stacked on top of each other, each containing a different image that appears on the Stage. The major components of the Timeline are layers, frames, and the playhead.

Layers in a document are listed in a column on the left side of the Timeline. Frames contained in each layer appear in a row to the right of the layer name. The Timeline header at the top of the Timeline indicates frame numbers. The playhead indicates the current frame displayed on the Stage.

The Timeline status display at the bottom of the Timeline indicates the selected frame number, the current frame rate, and the elapsed time to the current frame.

Note: When an animation is played, the actual frame rate is displayed; this may differ from the movie frame rate if the computer can't display the animation quickly enough.



You can change the way frames are displayed, as well as display thumbnails of frame content in the Timeline. The Timeline shows where there is animation in a movie, including frame-by-frame animation, tweened animation, and motion paths. For more information on animation, see Chapter 10, "Creating Animation," on page 169.

Controls in the layers section of the Timeline let you hide or show, lock, or unlock layers, as well as display layer contents as outlines. See "Editing layers and layer folders" on page 36.

You can insert, delete, select, and move frames in the Timeline. You can also drag frames to a new location on the same layer or to a different layer. See "Working with frames in the Timeline" on page 31.

Changing the appearance of the Timeline

By default, the Timeline appears at the top of the main application window, above the Stage. To change its position, you can dock the Timeline to the bottom or either side of the main application window, or display the Timeline as its own window. You can also hide the Timeline.

You can resize the Timeline to change the number of layers and frames that are visible. When there are more layers than can be displayed in the Timeline, you can view additional layers by using the scroll bars on the right side of the Timeline.

To move the Timeline:

Drag from the area above the Timeline header.

Drag the Timeline to the edge of the application window to dock it. Control-drag to prevent the Timeline from docking.

To lengthen or shorten layer name fields:

Drag the bar separating the layer names and the frames portion of the Timeline.

To resize the Timeline, do one of the following:

- If the Timeline is docked to the main application window, drag the bar separating the Timeline from the application window.
- If the Timeline is not docked to the main application window, drag the lower right corner (Windows) or the Size box in the lower right corner (Macintosh).

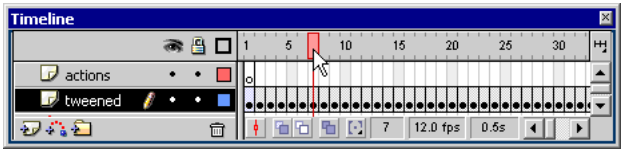
Moving the playhead

The playhead moves through the Timeline to indicate the current frame displayed on the Stage. The Timeline header shows the frame numbers of the animation. To display a frame on the Stage, you move the playhead to the frame in the Timeline.

When you're working with a large number of frames that can't all appear in the Timeline at once, you can move the playhead along the Timeline to easily locate the current frame.

To go to a frame:

Click the frame's location in the Timeline header, or drag the playhead to the desired position.



To center the Timeline on the current frame:

Click the Center Frame button at the bottom of the Timeline.

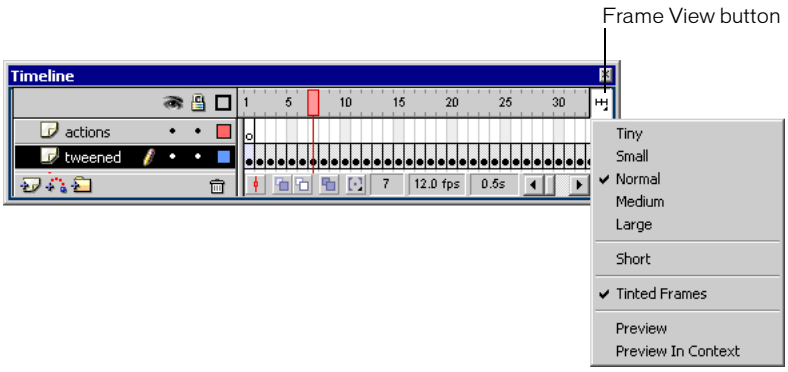
Changing the display of frames in the Timeline

You can change the size of frames in the Timeline, and display sequences of frames with tinted cells. You can also include thumbnail previews of frame content in the Timeline. These thumbnails are useful as an overview of the animation, but they require extra screen space.

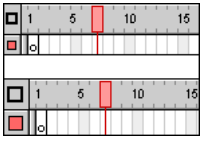
To change the display of frames in the Timeline:

1. Click the Frame View button in the upper right corner of the Timeline to display the Frame View pop-up menu.
2. Choose from the following options:

- To change the width of frame cells, choose Tiny, Small, Normal, Medium, or Large. (The Large frame-width setting is useful for viewing the details of sound waveforms.)
- To decrease the height of frame cell rows, choose Short.
- To turn tinting of frame sequences on or off, choose Tinted Frames.
- To display thumbnails of the content of each frame scaled to fit the Timeline frames, choose Preview. This can cause the apparent content size to vary.
- To display thumbnails of each full frame (including empty space), choose Preview in Context. This is useful for viewing the way elements move within their frames over the course of the animation, but previews are generally smaller than with the Preview option.



Frame View pop-up menu



Short and Normal frame view options

Using frames and keyframes

A keyframe is a frame in which you define a change in an animation or include frame actions to modify a movie. Flash can tween, or fill in, the frames between keyframes to produce fluid animations. Because keyframes let you produce animation without drawing each frame, they make creating movies easier. You can change the length of a tweened animation by dragging a keyframe in the Timeline.

The order in which frames and keyframes appear in the Timeline determines the order in which they are displayed in a movie. You can arrange keyframes in the Timeline to edit the sequence of events in a movie.

Working with frames in the Timeline

In the Timeline, you work with frames and keyframes, placing them in the order you want the objects in the frames to appear. You can change the length of a tweened animation by dragging a keyframe in the Timeline.

You can perform the following modifications on frames or keyframes:

- Insert, select, delete, and move frames or keyframes
- Drag frames and keyframes to a new location on the same layer or on a different layer
- Copy and paste frames and keyframes
- Convert keyframes to frames
- Drag an item from the Library panel onto the Stage to add the item to the current keyframe

The Timeline provides a view of tweened frames in an animation. For information on editing tweened frames, see Chapter 10, "Creating Animation," on page 169.

Flash offers two different methods for selecting frames in the Timeline. In frame-based selection (the default) you select individual frames in the Timeline. In span-based selection, the entire frame sequence, from one keyframe to the next, is selected when you click any frame in the sequence. For information on using span-based selection, see "Setting preferences in Flash" on page 22.

To insert frames in the Timeline, do one of the following:

- To insert a new frame, choose Insert > Frame.
- To create a new keyframe, choose Insert > Keyframe, or right-click (Windows) or Control-click (Macintosh) the frame where you want to place a keyframe, and choose Insert Keyframe from the context menu.
- To create a new blank keyframe, choose Insert > Blank Keyframe, or right-click (Windows) or Control-click (Macintosh) the frame where you want to place the keyframe, and choose Insert Blank Keyframe from the context menu.

To delete or modify a frame or keyframe, do one of the following:

- To delete a frame, keyframe, or frame sequence, select the frame, keyframe, or sequence and choose Insert > Remove Frame, or right-click (Windows) or Control-click (Macintosh) the frame, keyframe, or sequence and choose Remove Frame from the context menu. Surrounding frames remain unchanged.
- To move a keyframe or frame sequence and its contents, drag the keyframe or sequence to the desired location.

- To extend the duration of a keyframe, Alt-drag (Windows) or Option-drag (Macintosh) the keyframe to the final frame of the new sequence duration.
- To copy a keyframe or frame sequence by dragging, Alt-click (Windows) or Option-click (Macintosh) and drag the keyframe to the new location.
- To copy and paste a frame or frame sequence, select the frame or sequence and choose Edit > Copy Frames. Select a frame or sequence that you want to replace, and choose Edit > Paste Frames.
- To convert a keyframe to a frame, select the keyframe and choose Insert > Clear Keyframe, or right-click (Windows) or Control-click (Macintosh) the keyframe and choose Clear Keyframe from the context menu. The cleared keyframe and all frames up to the subsequent keyframe are replaced with the contents of the frame preceding the cleared keyframe.
- To change the length of a tweened sequence, drag the beginning or ending keyframe left or right. To change the length of a frame-by-frame sequence, see "Creating frame-by-frame animations" on page 180.
- To add an item from the library to the current keyframe, drag the item from the Library panel onto the Stage.

Using the Property inspector to set frame attributes

The Property inspector simplifies document creation by making it easy to edit frame attributes. The contents of the Property inspector change to reflect the contents of the frame, letting you edit a frame without accessing the menus or panels that contain these features.

In addition to changing the name of a frame and making a keyframe a named anchor, you can use the Property inspector to set animation and sound attributes. To edit animation settings, you use the Tween, Scale, Ease, Rotate, Orient to Path, Sync, and Snap options in the Property inspector. For more information, see "Tweening instances, groups, and type" on page 173. You use the Sound, Effect, Edit, Sync, and Loop options to edit sound settings. See "Adding sounds to a movie" under Help > Using Flash.

To edit the name of a frame:

1. If the Property inspector is not visible, choose Window > Properties.
2. Type a new name for the frame in the Frame text box in the Property inspector.

Creating frame labels and comments

Frame labels are useful for identifying keyframes in the Timeline and should be used instead of frame numbers when targeting frames in actions such as Go To. If you add or remove frames, the label moves with the frame it was originally attached to, whereas frame numbers can change. Frame labels are included when you publish a document as a Flash movie, so avoid long names to minimize file size.

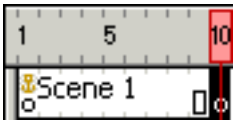
Frame comments are useful for making notes to yourself and others working on the same document. Frame comments are not exported when you publish a document as a Flash movie, so you can make them as long as you want.

To create a frame label or comment:

1. Select a frame.
2. If the Property inspector is not visible, choose Window > Properties.
3. In the Property inspector, enter the frame label or comment in the Frame Label text box. To make the text a comment, enter two slashes (//) at the beginning of each line of the text.

Using named anchors

Named anchors simplify navigation in Flash movies by letting viewers use the Forward and Back buttons in a browser to jump from frame to frame or scene to scene. Named anchor keyframes are indicated in the Timeline by an anchor icon. If you prefer to have Flash automatically make the first keyframe of each scene a named anchor, see "Setting preferences in Flash" on page 22.



A named anchor keyframe in Scene 1

To take advantage of named anchor keyframes in your final Flash movie, select the Flash w/ Named Anchors option in the Template pop-up menu on the HTML tab of the Publish Settings dialog box. For more information on the Publish Settings dialog box, see "Choosing publish settings Flash movie format" on page 370.

To use Flash movies with named anchors, you must be running Flash Player 6 on your browser.

Note: If you save a document with named anchor keyframes as a Flash 5 document, the named anchor keyframes are converted to regular labeled frames.

To make a selected keyframe a named anchor:

1. If the Property inspector is not visible, choose Window > Properties.
2. Type a name for the keyframe in the text box in the Property inspector.
3. Select the Named Anchor option.

To make a named anchor keyframe to a regular keyframe:

1. Select the named anchor keyframe in the Timeline.
2. Deselect the Named Anchor option in the Property inspector.

Using layers

Layers are like transparent sheets of acetate stacked on top of each other. Layers help you organize the artwork in your document. You can draw and edit objects on one layer without affecting objects on another layer. Where there is nothing on a layer, you can see through it to the layers below.

To draw, paint, or otherwise modify a layer or folder, you select the layer to make it active. A pencil icon next to a layer or folder name indicates that the layer or folder is active. Only one layer can be active at a time (although more than one layer can be selected at a time).

When you create a new Flash document, it contains one layer. You can add more layers to organize the artwork, animation, and other elements in your document. The number of layers you can create is limited only by your computer's memory, and layers do not increase the file size of your published movie. You can hide, lock, or rearrange layers.

You can also organize and manage layers by creating layer folders and placing layers in them. You can expand or collapse layers in the Timeline without affecting what you see on the Stage. It's a good idea to use separate layers or folders for sound files, actions, frame labels, and frame comments. This helps you find these items quickly when you need to edit them.

In addition, you can use special guide layers to make drawing and editing easier, and mask layers to help you create sophisticated effects.

For an interactive introduction to layers, choose Help > Lessons > Understanding Layers.

Creating layers and layer folders

When you create a new layer or folder, it appears above the selected layer. A newly added layer becomes the active layer.

To create a layer, do one of the following:

- Click the Add Layer button at the bottom of the Timeline.
- Choose Insert > Layer.
- Right-click (Windows) or Control-click (Macintosh) a layer name in the Timeline and choose Insert Layer from the context menu.

To create a layer folder, do one of the following:

- Select a layer or folder in the Timeline, then choose Insert > Layer Folder.
- Right-click (Windows) or Control-click (Macintosh) a layer name in the Timeline, then choose Insert Folder from the context menu.

  The new folder appears above the layer or folder you selected.

Viewing layers and layer folders

As you work, you may want to show or hide layers or folders. A red X next to a the name of layer or folder name indicates that it is hidden. Hidden layers are not preserved when a movie is published.

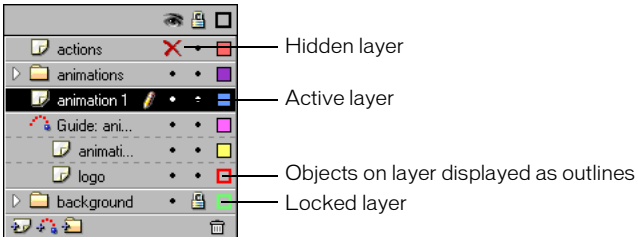
To help you distinguish which layer objects belong to, you can display all objects on a layer as colored outlines. You can change the outline color used by each layer.

You can change the height of layers in the Timeline in order to display more information (such as sound waveforms) in the Timeline. You can also change the number of layers displayed in the Timeline.

To show or hide a layer or folder, do one of the following:

- Click in the Eye column to the right of the layer or folder name in the Timeline to hide that layer or folder. Click in it again to show the layer or folder.
- Click the eye icon to hide all the layers and folders. Click it again to show all layers and folders.
- Drag through the Eye column to show or hide multiple layers or folders.

- Alt-click (Windows) or Option-click (Macintosh) in the Eye column to the right of a layer or folder name to hide all other layers and folders. Alt-click or Option-click it again to show all layers and folders.



To view the contents of a layer as outlines, do one of the following:

- Click in the Outline column to the right of the layer's name to display all objects on that layer as outlines. Click in it again to turn off outline display.
- Click the outline icon to display objects on all layers as outlines. Click it again to turn off outline display on all layers.
- Alt-click (Windows) or Option-click (Macintosh) in the Outline column to the right of a layer's name to display objects on all other layers as outlines. Alt-click or Option-click in it again to turn off outline display for all layers.

To change a layer's outline color:

1 Do one of the following:

- Double-click the layer's icon (the icon to the left of the layer name) in the Timeline.
- Right-click (Windows) or Control-click (Macintosh) the layer name and choose Properties from the context menu.
- Select the layer in the Timeline and choose Modify > Layer.

2 In the Layer Properties dialog box, click the Outline Color box and select a new color, enter the hexadecimal value for a color, or click the Color Picker button and choose a color.

3 Click OK.

To change layer height in the Timeline:

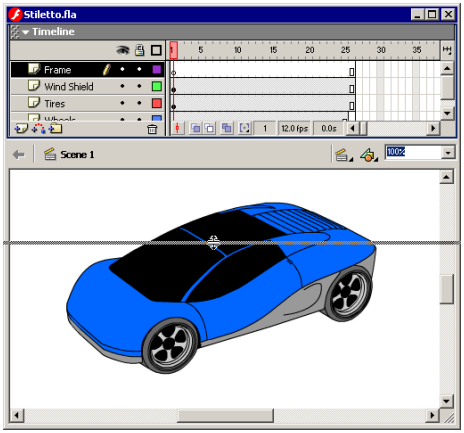
1 Do one of the following:

- Double-click the layer's icon (the icon to the left of the layer name) in the Timeline.
- Right-click (Windows) or Control-click (Macintosh) the layer name and choose Properties from the context menu.
- Select the layer in the Timeline and choose Modify > Layer.

2 In the Layer Properties dialog box, choose an option for Layer Height and click OK.

To change the number of layers displayed in the Timeline:

Drag the bar that separates the Timeline from the Stage.



Editing layers and layer folders

You can rename, copy, and delete layers and folders. You can also lock layers and folders to prevent them from being edited.

By default, new layers are named by the order in which they are created: Layer 1, Layer 2, and so on. You can rename layers to better reflect their contents.

To select a layer or folder, do one of the following:

- Click the name of a layer or folder in the Timeline.
- Click a frame in the Timeline of the layer you want to select.
- Select an object on the Stage that is located on the layer you want to select.

To select two or more layers or folders, do one of the following:

- To select contiguous layers or folders, Shift-click their names in the Timeline.
- To select discontiguous layers or folders, Control-click (Windows) or Command-click (Macintosh) their names in the Timeline.

To rename a layer or folder, do one of the following:

- Double-click the name of a layer or folder and enter a new name.
- Right-click (Windows) or Control-click (Macintosh) the name of a layer or folder and choose Properties from the context menu. Enter the new name in the Name text box and click OK.
- Select the layer or folder in the Timeline and choose Modify > Layer. In the Layer Properties dialog box, enter the new name in the Name text box and click OK.

To lock or unlock one or more layers or folders, do one of the following:

- Click in the Lock column to the right of the name of a layer or folder to lock it. Click in the Lock column again to unlock the layer or folder.
- Click the padlock icon to lock all layers and folders. Click it again to unlock all layers and folders.
- Drag through the Lock column to lock or unlock multiple layers or folders.
- Alt-click (Windows) or Option-click (Macintosh) in the Lock column to the right of a layer or folder name to lock all other layers or folders. Alt-click or Option-click in the Lock column again to unlock all layers or folders.

To copy a layer:

1. Click the layer name to select the entire layer.
2. Choose Edit > Copy Frames.
3. Click the Add Layer button to create a new layer.
4. Click the new layer and choose Edit > Paste Frames.

To copy the contents of a layer folder:

1. Click the triangle to the left of the folder name to collapse it, if necessary.
2. Click the folder name to select the entire folder.
3. Choose Edit > Copy Frames.
4. Choose Insert > Layer Folder to create a new folder.
5. Click the new folder and choose Edit > Paste Frames.

To delete a layer or folder:

1. Select the layer or folder.
2. Do one of the following:

- Click the Delete Layer button in the Timeline.
- Drag the layer or folder to the Delete Layer button.
- Right-click (Windows) or Control-click (Macintosh) the layer or folder name and choose Delete Layer from the context menu.

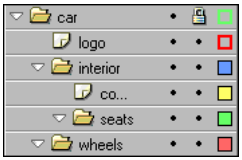
Note: When you delete a layer folder, all the enclosed layers and all their contents are also deleted.

Organizing layers and layer folders

You can rearrange layers and folders in the Timeline to organize your document.

Layer folders help organize your workflow by letting you place layers in a tree structure. You can expand or collapse a folder to see the layers it contains without affecting which layers are visible on the Stage. Folders can contain both layers and other folders, allowing you to organize layers in much the same way you organize files on your computer.

The layer controls in the Timeline affect all layers within a folder. For example, locking a layer folder locks all layers within that folder.



To move a layer or layer folder into a layer folder:

Drag the layer or layer folder name to the destination layer folder name.

The layer or layer folder appears inside the destination layer folder in the Timeline.

To change the order of layers or folders:

Drag one or more layers or folders in the Timeline to the desired position.

To expand or collapse a folder:

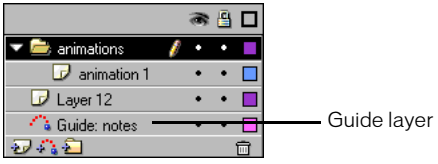
Click the triangle to the left of the folder name.

To expand or collapse all folders:

Right-click (Windows) or Control-click (Macintosh) and choose Expand All Folders or Collapse All Folders from the context menu.

Using guide layers

For help in aligning objects when drawing, you can create guide layers. You can then align objects on other layers to the objects you create on the guide layers. Guide layers do not appear in a published Flash movie. You can make any layer a guide layer. Guide layers are indicated by a guide icon to the left of the layer name.



You can also create a motion guide layer to control the movement of objects in a motion tweened animation. See “Tweening motion along a path” on page 176.

Note: Dragging a normal layer onto a guide layer converts the guide layer to a motion guide layer. To prevent accidentally converting a guide layer, place all guide layers at the bottom of the layer order.

To designate a layer as a guide layer:

Select the layer and right-click (Windows) or Control-click (Macintosh) and select Guide from the context menu. Select Guide again to change the layer back to a normal layer.

Previewing and testing movies

As you create a movie, you'll need to play it back to preview animation and test interactive controls. You can preview and test movies within the Flash authoring environment, in a separate test window in Flash, or in a Web browser.

Previewing movies in the authoring environment

To preview movies, you use commands in the Control menu, buttons on the Controller, or keyboard commands.

To preview the current scene, do one of the following:

- Choose Control > Play.
- Choose Window > Toolbars > Controller (Windows) or Window > Controller (Macintosh) and click Play.
- Press Enter (Windows) or Return (Macintosh). The animation sequence plays at the frame rate you specified for the document.
- To step through the frames of the animation, use the Step Forward and Step Backward buttons on the Controller, or choose those commands from the Control menu. You can also press the < and > keys on the keyboard.
- To jump to the first or last frame in a movie using the Controller, use the First Frame or Last Frame button.

Note: You can also drag the playhead to view frames in a document. See "Moving the playhead" on page 29.

You can modify movie playback using commands in the Control menu. When using the following commands, you must also choose Control > Play to preview the movie.

To play the movie in a continuous loop:

Choose Control > Loop Playback.

To play all the scenes in a movie:

Choose Control > Play All Scenes.

To play a movie without sound:

Choose Control > Mute Sounds.

To enable frame actions or button actions:

Choose Control > Enable Simple Frame Actions or Enable Simple Buttons.

Previewing movies with the Test Movie command

Although Flash can play movies in the authoring environment, many animation and interactive functions cannot work unless the document is exported to its final Flash movie format. Using commands in the Control menu, you can export the current document as a Flash movie and immediately play it using the Test Movie command. The exported movie uses the options set in the Publish Settings dialog box. You can also use the Test Movie command to test downloading performance. See "Testing movie download performance" under Help > Using Flash.

In addition, you can test actions in a movie using the Debugger. See “Using the Debugger” under Help > Using Flash.

To test all interactive functions and animation:

Choose Control > Test Movie or Control > Test Scene.

Flash creates a Flash movie (a SWF file), opens it in a separate window, and plays it with the Flash Player. The SWF file is placed in the same folder as the FLA file.

Previewing movies in a Web browser

For the most accurate representation of a Flash movie, you should preview it in your default Web browser.

To test the movie in a Web browser:

Choose File > Publish Preview > HTML.

Flash creates a Flash movie (a SWF file), opens it in your default Web browser, and plays it with the Flash Player. The SWF file is placed in the same folder as the FLA file. For more information, see “About HTML publishing templates” on page 382.

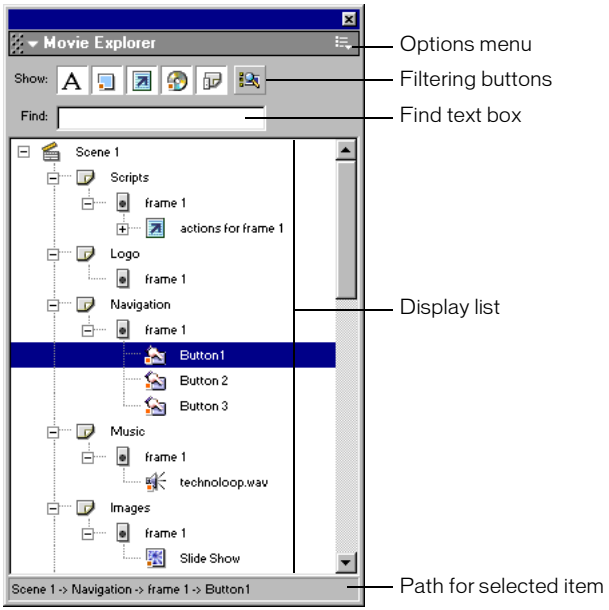
Using the Movie Explorer

The Movie Explorer provides an easy way for you to view and organize the contents of a document and select elements in the document for modification. It contains a display list of currently used elements, arranged in a navigable hierarchical tree. You can filter which categories of items in the document are displayed in the Movie Explorer, choosing from text, graphics, buttons, movie clips, actions, and imported files. You can display the selected categories as movie elements (scenes), symbol definitions, or both. You can expand and collapse the navigation tree.

The Movie Explorer offers many features to streamline the workflow for creating movies. For example, you can use the Movie Explorer to do the following:

- Search for an element in a document by name
- Familiarize yourself with the structure of a Flash document created by another developer
- Find all the instances of a particular symbol or action
- Replace all occurrences of a font in a document with another font
- Copy all text to the Clipboard to paste into an external text editor for spell checking
- Print the navigable display list currently displayed in the Movie Explorer

The Movie Explorer has an options menu as well as a context menu with options for performing operations on selected items or modifying the Movie Explorer display. The options menu is indicated by a check mark with a triangle below it in the title bar of the Movie Explorer.



To view the Movie Explorer:

Choose Window > Movie Explorer.

To filter the categories of items displayed in the Movie Explorer:

- To show text, symbols, ActionScript, imported files, or frames and layers, click one or more of the filtering buttons to the right of the Show option. To customize which items to show, click the Customize button. Select options in the Show area of the Movie Explorer Settings dialog box to view those elements.
- From the options menu in Movie Explorer, choose Show Movie Elements to display items in scenes, and choose Show Symbol Definitions to display information about symbols. (Both options can be active at the same time.)

To search for an item using the Find text box:

In the Find text box, enter the item name, font name, ActionScript string, or frame number. The Find feature searches all items currently displayed in the Movie Explorer.

To select an item in the Movie Explorer:

Click the item in the navigation tree. Shift-click to select more than one item.

The full path for the selected item appears at the bottom of the Movie Explorer. Selecting a scene in the Movie Explorer displays the first frame of that scene on the Stage. Selecting an element in the Movie Explorer selects that element on the Stage if the layer containing the element is not locked.

To use the Movie Explorer options menu or context menu commands:

1 Do one of the following:

- To view the options menu, click the options menu control in the Movie Explorer's title bar.
- To view the context menu, right-click (Windows) or Control-click (Macintosh) an item in the Movie Explorer navigation tree.

2 Select an option from the menu:

- Go to Location jumps to the selected layer, scene, or frame in the document.
- Go to Symbol Definition jumps to the symbol definition for a symbol that is selected in the Movie Elements area of the Movie Explorer. The symbol definition lists all the files associated with the symbol. (The Show Symbol Definitions option must be selected. See option definition below.)
- Select Symbol Instances jumps to the scene containing instances of a symbol that is selected in the Symbol Definitions area of the Movie Explorer. (The Show Movie Elements option must be selected.)
- Find in Library highlights the selected symbol in the document's library (Flash opens the Library panel if it is not already visible).
- Rename lets you enter a new name for a selected element.
- Edit in Place lets you edit a selected symbol on the Stage.
- Edit in New Window lets you edit a selected symbol in a new window.
- Show Movie Elements displays the elements in your movie, organized into scenes.
- Show Symbol Definitions displays all the elements associated with a symbol.
- Copy All Text to Clipboard copies selected text to the Clipboard. You can paste the text into an external text editor for spell checking or other editing.
- Cut, Copy, Paste, and Clear perform these common functions on a selected element. Modifying an item in the display list modifies the corresponding element in the movie.
- Expand Branch expands the navigation tree at the selected element.
- Collapse Branch collapses the navigation tree at the selected element.
- Collapse Others collapses the branches in the navigation tree not containing the selected element.
- Print prints the hierarchical display list currently displayed in the Movie Explorer.

Speeding up movie display

To speed up the movie display, you can use commands in the View menu to turn off rendering-quality features that require extra computing and slow down movies.

None of these commands have any effect on how Flash exports a movie. To specify the display quality of Flash movies in a Web browser, you use the `OBJECT` and `EMBED` parameters. The Publish command can do this for you automatically. For more information, see "Publishing Flash documents" on page 367.

To change the movie display speed:

Choose View and select from the following options:

- Outlines displays only the outlines of the shapes in your scene and causes all lines to appear as thin lines. This makes it easier to reshape your graphic elements and to display complex scenes faster.
- Fast turns off anti-aliasing and displays all the colors and line styles of your drawing.
- Antialias turns on anti-aliasing for lines, shapes, and bitmaps. It displays shapes and lines so that their edges appear smoother on the screen. This option draws more slowly than the Fast option. Anti-aliasing works best on video cards that provide thousands (16-bit) or millions (24-bit) of colors. In 16- or 256-color mode, black lines are smoothed, but colors might look better in fast mode.
- Antialias Text smooths the edges of any text. This command works best with large font sizes and can be slow with large amounts of text. This is the most common mode in which to work.

Saving Flash documents

You can save a Flash FLA document using its current name and location, or save the document using a different name or location. You can revert to the last saved version of a document. You can also save Flash MX content as a Flash 5 document.

You can save a document as a template, in order to use the document as the starting point for a new Flash document (this is similar to how you would use templates in word-processing or Web page–editing applications). For information on using templates to create new documents, see "Creating a new document" on page 21.

When you save a document using the Save command, Flash performs a quick save, which appends new information to the existing file. When you save using the Save As command, Flash arranges the new information into the file, creating a smaller file on disk.

To save a Flash document:

1. Do one of the following:

- To overwrite the current version on the disk, choose File > Save.
- To save the document in a different location and/or with a different name, or to compress the document, choose File > Save As.

2. If you chose the Save As command, or if the document has never been saved before, enter the file name and location.
3. To save the document in Flash MX format, choose Flash MX Document from the Format pop-up menu. If an alert message indicates that content will be deleted if you save in Flash MX format, click Save As Flash MX if you wish to continue.
4. Click Save.

To revert to the last saved version of a document:

Choose File > Revert.

To save a document as a template:

1. Choose File > Save As Template.
2. In the Save As Template dialog box, enter a name for the template in the Name text box.
3. Choose a category from the Category pop-up menu, or enter a name to create a new category.
4. Enter a description of the template in the Description text box (up to 255 characters). The description will be displayed when the template is selected in the New Document dialog box (see "Previewing and testing movies" on page 39).
5. Click OK.

To save as a Flash 5 document:

1. Choose File > Save As.
2. Enter the file name and location.
3. Choose Flash 5 Document from the Format pop-up menu. If an alert message indicates that content will be deleted if you save in Flash 5 format, click Save As Flash 5 to continue.
4. Click Save.

Configuring a server for the Flash Player

For a user to view your Flash movie on the Web, the Web server must be properly configured to recognize the SWF file as a Flash movie.

Your server may already be configured properly. If your server is not properly configured, follow the procedure below to configure it.

Configuring a server establishes the appropriate Multipart Internet Mail Extension (MIME) types for the server to identify files with the SWF suffix as Shockwave Flash files.

A browser that receives the correct MIME type can load the appropriate plug-in, control, or helper application to process and properly display the incoming data. If the MIME type is missing or not properly delivered by the server, the browser might display an error message or a blank window with a puzzle piece icon.

Note: When you publish a Flash movie, you must configure the movie for the Flash Player in order for users to view the movie. See Chapter 20, "Publishing," on page 365.

To configure a server for the Flash Player, do one of the following:

- If your site is established through an Internet service provider, contact them and request that the MIME type application/x-shockwave-flash with the SWF suffix be added to the server.
- If you are administering your own server, consult the documentation for your Web server software for instructions on adding or configuring MIME types.

Printing Flash documents as you edit

You can print frames from Flash documents as you work, to preview and edit your movies.

You can also specify frames to be printable from the Flash Player by a viewer displaying the Flash movie. See “Creating Printable Movies” under Help > Using Flash.

When printing frames from a Flash document, you use the Print dialog box to specify the range of scenes or frames you want to print, as well as the number of copies. In Windows, the Page Setup dialog box specifies paper size, orientation, and various print options—including margin settings and whether all frames are to be printed for each page. On the Macintosh, these options are divided between the Page Setup and the Print Margins dialog boxes.

The Print and Page Setup dialog boxes are standard within either operating system, and their appearance depends on the printer driver selected.

To set printing options:

1. Choose File > Page Setup (Windows) or File > Print Margins (Macintosh).
2. Set page margins. Select both Center options to print the frame in the center of the page.
3. In the Frames pop-up menu, choose to print all frames in the movie or only the first frame of each scene.
4. In the Layout pop-up menu, choose from the following options:

- Actual Size prints the frame at full size. Enter a value for Scale to reduce or enlarge the printed frame.
- Fit on One Page reduces or enlarges each frame so it fills the print area of the page.
- Storyboard options print several thumbnails on one page. Enter the number of thumbnails per page in the Frames text box. Set the space between the thumbnails in the Story Margin text box. Select Label to print the frame label as a thumbnail.

To preview how your scene is arranged on the printer paper:

Choose File > Print Preview.

To print frames:

Choose File > Print.

CHAPTER 2
Working with Flash assets

Macromedia Flash MX assets are the various elements you use to create a movie. Assets include objects on the Stage, symbols and symbol instances, sound clips, and other imported files. Flash provides tools to help you organize and optimize assets. These tools improve your workflow by keeping the most frequently used features of Flash easily accessible.

Assets and asset management

Most Flash assets are objects on the Stage or symbols stored in the document's library. Other assets include files on local or remote computers. The library makes organization easier and helps optimize file size. The toolbox, inspectors, panels, and library work help you work efficiently with the assets of any document, whether simple or complex.

Symbols and instances

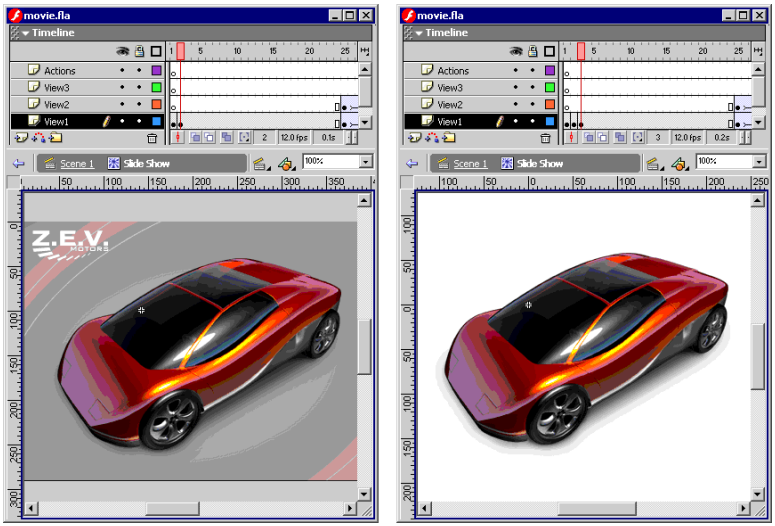
Symbols are reusable elements that you use with a document. Symbols can include graphics, buttons, video clips, sound files, or fonts. When you create a symbol, the symbol is stored in the file's library. When you place a symbol on the Stage, you create an *instance* of that symbol.

Symbols reduce file size because, regardless of how many instances of a symbol you create, Flash stores the symbol in the file only once. It is a good idea to use symbols, animated or otherwise, for every element that appears more than once in a document. You can modify the properties of an instance without affecting the master symbol, and you can edit the master symbol to change all instances.

You can edit symbols in several ways: in place on the Stage, in a new window, or in symbol-editing mode. When you edit a symbol, the Timeline window displays only the Timeline of the symbol you are editing. For more information on editing symbols, see "Editing symbols" on page 157.

You can locate and open a symbol in the library from within the Movie Explorer, using the Find in Library command. See "Using the Movie Explorer" on page 40.

For more information on symbols and instances, see the Symbols lesson, located under Help > Lessons > Creating and Editing Symbols, and Chapter 9, “Using Symbols, Instances, and Library Assets,” on page 149.



Editing a symbol in its context in the document (left) and editing a symbol in isolation (right)

Symbols and interactive movies

Symbols are an integral part of creating interactive movies; you can use instances of symbols to create interactivity in a movie. For example, you can create a button symbol that changes in response to mouse actions and place an instance of the symbol on the Stage. You use another type of symbol, called a movie clip, to create sophisticated interactive movies. See “Working with Movie Clips and Buttons” under Help > Using Flash.

Panels and the Property inspector

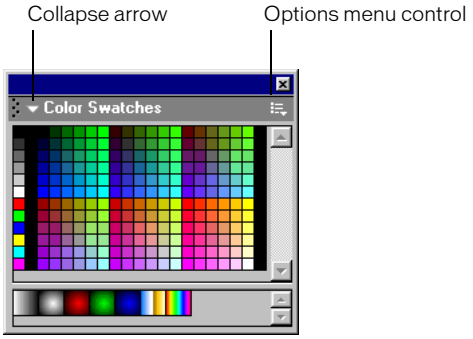
Flash offers many ways to customize the workspace to your needs. Using panels and the Property inspector, you can view, organize, and change assets and their attributes. You can show, hide, and resize panels. You can also group panels and save custom panel sets to make managing your workspace easier. The Property inspector changes to reflect the tool or asset you are working with, giving you quick access to frequently used features.

Using panels

Panels in Flash help you view, organize, and change elements in a document. The options available on panels control the characteristics of symbols, instances, colors, type, frames, and other elements. You can use panels to customize the Flash interface, by displaying the panels you need for a specific task and hiding other panels.

Panels let you work with objects, colors, text, instances, frames, scenes, and entire documents. For example, you use the Color Mixer to create colors, and the Align panel to align objects to each other or the Stage. To view the complete list of panels available in Flash, see the Window menu.

Most panels include a pop-up menu with additional options. The options menu is indicated by a control in the panel's title bar. (If no options menu control appears, there is no options menu for that panel.)



To open a panel:

Select the desired panel from the Window menu.

To close a panel, do one of the following:

- Select the desired panel from the Window menu.
- Right-click (Windows) or Control-click (Macintosh) the panel's title bar and choose Close Panel from the context menu.

To use a panel's options menu:

1. Click the control in the panel's title bar to view the options menu.
2. Click an item in the menu.

To resize a panel:

Drag the panel's border (Windows) or drag the size box at the panel's lower right corner (Macintosh).

To expand or collapse a panel to its title bar:

Click the collapse arrow in the title bar. Click the collapse arrow again to expand the panel to its previous size.

To show or hide all panels:

Press Tab.

To close all panels:

Choose Window > Close All Panels.

Arranging panels

You can rearrange the order in which panels appear within panel groups. You can also create new panel groups and dock panels to existing panel groups.

To move a panel:

Drag the panel by its title bar.

To add a panel to an existing panel group:

Drag the panel by its title bar onto another panel.

To create a new panel group:

Drag the panel by its title bar, away from other panel groups.

Using panel sets

You can create panel sets in custom arrangements, and you can save these custom panel layouts. You can reset panel display to the default layout (displaying the Color Swatches, Actions, and Components panels and the Color Mixer to the right of the application window) or to a custom layout that you have saved previously.

To save a custom panel set:

1. Choose Window > Save Panel Layout.
2. Enter a name for the layout and click OK.

To select a panel layout:

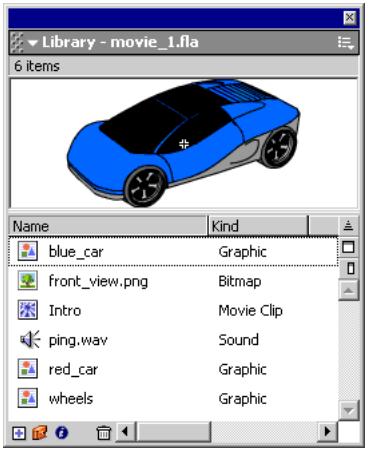
1. Choose Window > Panel Sets.
2. From the submenu, choose Default Layout to reset panels to the default layout, or choose a custom layout that you have saved previously.

To delete custom layouts:

Open the Panel Sets folder inside the Flash MX application folder on your hard drive and delete the Panel Sets file.

Working with the Library panel

The Library panel is where you store and organize symbols created in Flash, as well as imported files, including bitmap graphics, sound files, and video clips. The Library panel lets you organize library items in folders, see how often an item is used in a document, and sort items by type. See "Using the library" on page 54.



To display or hide the Library panel:

Choose Window > Library.

Working with the Actions panel

The Actions panel lets you create and edit actions for an object or frame. Selecting a frame, button, or movie clip instance makes the Actions panel active. The Actions panel title changes to Button Actions, Movie Clip Actions, or Frame Actions, depending on what is selected.

For information on using the Actions panel, including switching between editing modes, see "Using the Actions panel" under Help > Using Flash.

To display or hide the Actions panel:

Choose Window > Actions.

Using the Property inspector

The Property inspector simplifies document creation by making it easy to access the most commonly used attributes of the current selection, either on the Stage or in the Timeline. You can make changes to the object or document attributes in the Property inspector without accessing the menus or panels that contain these features.

Depending on what is currently selected, the Property inspector displays information and settings for the current document, text, symbol, shape, bitmap, video, group, frame, or tool. When two or more different types of objects are selected, the Property inspector displays the total number of objects selected.

The Property inspector displaying text options

To display or hide the Property inspector:

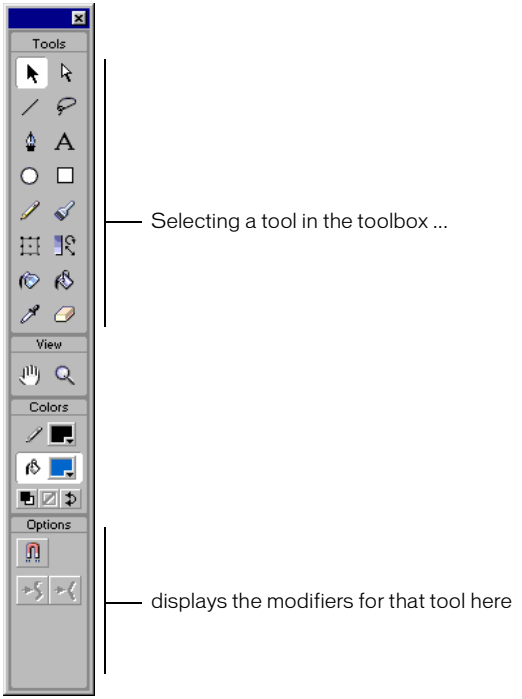
Choose Window > Properties.

Using the toolbox

The tools in the toolbox let you draw, paint, select, and modify artwork, as well as change the view of the Stage. The toolbox is divided into four sections:

- The Tools section contains drawing, painting, and selection tools.
- The View section contains tools for zooming and panning in the application window.
- The Colors section contains modifiers for stroke and fill colors.
- The Options section displays modifiers for the selected tool, which affect the tool's painting or editing operations.

For information on using the drawing and painting tools, see “Flash drawing and painting tools” on page 61. For information on using the selection tools, see “Selecting objects” on page 119. For information on using the view modification tools, see “Viewing the Stage” on page 19.



To show or hide the toolbox:

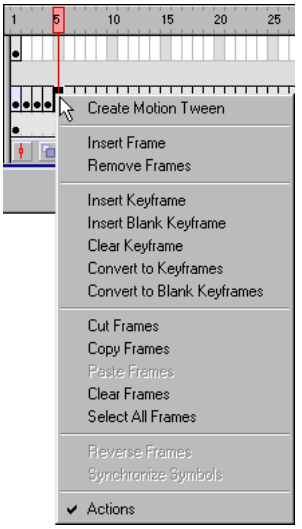
Choose Window > Tools.

To select a tool, do one of the following:

- Click the tool you want to use. Depending on the tool you select, a set of modifiers may be displayed in the Options area at the bottom of the toolbox.
- Press the tool’s keyboard shortcut.

Using context menus

Context menus contain commands relevant to the current selection. For example, when you select a frame in the Timeline window, the context menu contains commands for creating, deleting, and modifying frames and keyframes. Context menus exist for many items and controls in many locations, including on the Stage, in the Timeline, in the Library panel, and in the Actions panel.



Context menu for a selected frame

To open a context menu:

Right-click (Windows) or Control-click (Macintosh) an item.

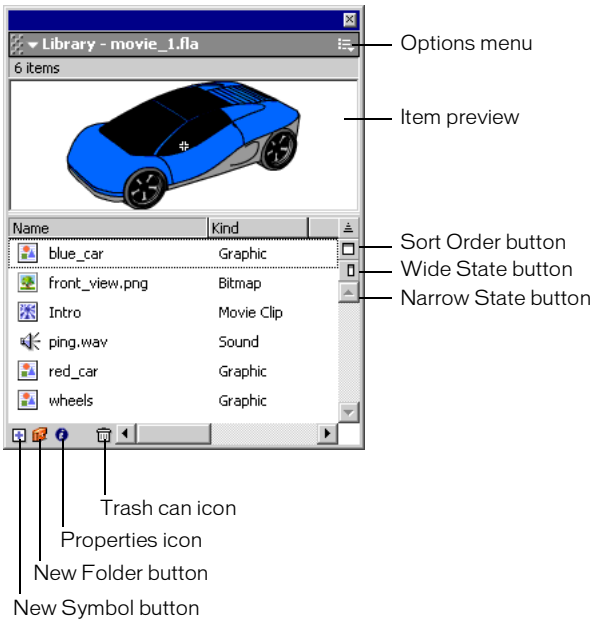
Using the library

The library in a Flash document stores symbols created in Flash, plus imported files such as video clips, sound clips, bitmaps, and imported vector artwork. The Library panel displays a scroll list with the names of all items in the library, allowing you to view and organize these elements as you work. An icon next to an item's name in the Library panel indicates the item's file type.

You can open the library of any Flash document while you are working in Flash, to make the library items from that file available for the current document.

You can create permanent libraries in your Flash application that will be available whenever you launch Flash. Flash also includes several sample libraries containing buttons, graphics, movie clips, and sounds that you can add to your own Flash documents. The sample Flash libraries and permanent libraries that you create are listed in the Window > Common Libraries submenu. See "Working with common libraries" on page 58.

You can export library assets as a SWF file to a URL to create a runtime-shared library. This allows you to link to the library assets from Flash movies that import symbols using runtime sharing. See “Using shared library assets” on page 165.



To display the Library panel:

Choose Window > Library.

To open the library from another Flash file:

1 Choose File > Open as Library.

2 Navigate to the Flash file whose library you want to open, and click Open.

The selected file’s library opens in the current document, with the file’s name at the top of the Library panel. To use items from the selected file’s library in the current document, drag the items to the current document’s Library panel or to the Stage.

To resize the Library panel, do one of the following:

- Drag the lower right corner.
- Click the Wide State button to enlarge the Library panel so that it displays all the columns.
- Click the Narrow State button to reduce the width of the Library panel.

To change the width of columns:

Position the pointer between column headers and drag to resize.

You cannot change the order of columns.

To use the Library options menu:

1. Click the control in the Library panel's title bar to view the options menu.
2. Click an item in the menu.

Working with library items

When you select an item in the Library panel, a thumbnail preview of the item appears at the top of the Library panel. If the selected item is animated or is a sound file, you can use the Play button in the Library preview window or the Controller to preview the item.

To use a library item in the current document:

Drag the item from the Library panel onto the Stage.

The item is added to the current layer.

To convert an object to a symbol in the library:

Drag the item from the Stage onto the current Library panel.

To use a library item from the current document in another document:

Drag the item from the library or Stage into the library or Stage for another document.

To move an item between folders:

Drag the item from one folder to another. If an item with the same name exists in the new location, Flash prompts you to replace the item you are moving.

To replace a symbol on the Stage with another symbol:

1. Select the symbol you want to replace.
2. Choose Modify > Swap Symbol.
3. Select a new symbol from the Swap Symbol dialog box, then click OK.

Working with folders in the Library panel

You can organize items in the Library panel using folders, much like in the Windows Explorer or the Macintosh Finder. When you create a new symbol, it is stored in the selected folder. If no folder is selected, the symbol is stored at the root of the library.

The Library panel columns list the name of an item, its type, the number of times it's used in the file, its linkage status and identifier (if the item is associated with a shared library or is exported for ActionScript), and the date on which it was last modified. You can sort items in the Library panel by any column. The Library panel also contains an options pop-up menu with options for modifying library items.

To create a new folder:

Click the New Folder button at the bottom of the Library panel.

To open or close a folder, do one of the following:

- Double-click the folder.
- Select the folder and choose Expand Folder or Collapse Folder from the Library options menu.

To open or close all folders:

Choose Expand All Folders or Collapse All Folders from the Library options menu.

Sorting items in the Library panel

You can sort items in the Library panel alphanumerically by any column. Sorting items lets you view related items together. Items are sorted within folders.

To sort items in the Library panel:

Click the column header to sort by that column. Click the triangle button to the right of the column headers to reverse the sort order.

Editing items in the library

To edit library items, including imported files, you choose options from the Library options menu. You can update imported files after editing them in an external editor, using the Update option in the Library options menu.

To edit a library item:

1. Select the item in the Library panel.
2. Choose one of the following from the Library options menu:

- Choose Edit to edit an item in Flash.
- Choose Edit With and select an application to edit the item in an external editor.

Note: When launching a supported external editor, Flash opens the original imported document.

Renaming library items

You can rename items in the library. Changing the library item name of an imported file does not change the file name.

To rename a library item, do one of the following:

- Double-click the item's name and enter the new name in the text field.
- Select the item and click the properties icon at the bottom of the Library panel. Enter the new name in the Symbol Properties dialog box and click OK.
- Select the item and choose Rename from the Library options menu, and then enter the new name in the text field.
- Right-click (Windows) or Control-click (Macintosh) the item and choose Rename from the context menu, and then enter the new name in the text field.

Deleting library items

When you delete an item from the library, all instances or occurrences of that item in the document are also deleted by default. The Use Count column in the Library panel indicates whether an item is in use.

To delete a library item:

1. Select the item and click the trash can icon at the bottom of the Library panel.
2. In the warning box that appears, select Delete Symbol Instances (the default) to delete the library item and all instances of it. Deselect the option to delete only the symbol, leaving the instances on the Stage.
3. Click Delete.

Finding unused library items

To make organizing a document easier, you can locate unused library items and delete them.

Note: It is not necessary to delete unused library items to reduce a Flash movie's file size, because unused library items are not included in the SWF file.

To find unused library items, do one of the following:

- Choose Select Unused Items from the Library options menu.
- Sort library items by the Use Count column. See "Sorting items in the Library panel" on page 57.

Updating imported files in the Library panel

If you use an external editor to modify files that you have imported into Flash, such as bitmaps or sound files, you can update the files in Flash without reimporting them. You can also update symbols that you have imported from external Flash documents. Updating an imported file replaces its contents with the contents of the external file.

To update an imported file:

Select the imported file in the Library panel and choose Update from the Library options menu.

Working with common libraries

You can use the sample libraries included with Flash to add symbols, buttons, or sounds to your documents. You can also create your own sample libraries, which you can then use with any documents that you create.

To create a sample library for your Flash application:

1. Create a Flash file with a library containing the symbols that you want to include in the permanent library.
2. Place the Flash file in the Libraries folder located in the Flash application folder on your hard drive.

To use an item from a common library in a document:

1. Choose Window > Common Libraries, and select a library from the submenu.
2. Drag an item from the common library into the library for the current document.

About components

Flash components are movie clips with defined parameters, each with its own unique set of ActionScript methods that allow you to set and change the authoring parameters and additional options at runtime. See Chapter 15, "Using Components," on page 289.

CHAPTER 3
Drawing

The drawing tools in Macromedia Flash MX let you create and modify shapes for the artwork in your movies. For an interactive introduction to drawing in Flash, choose Help > Lessons > Illustrating in Flash.

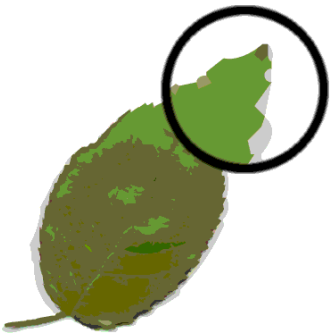
Before you draw and paint in Flash, it is important to understand how Flash creates artwork, how drawing tools work, and how drawing, painting, and modifying shapes can affect other shapes on the same layer.

About vector and bitmap graphics

Computers display graphics in either vector or bitmap format. Understanding the difference between the two formats can help you work more efficiently. Using Flash, you can create and animate compact vector graphics. Flash also lets you import and manipulate vector and bitmap graphics that have been created in other applications.

Vector graphics

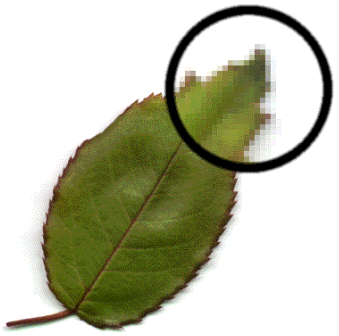
Vector graphics describe images using lines and curves, called *vectors,* that also include color and position properties. For example, the image of a leaf is described by points through which lines pass, creating the shape of the leaf's outline. The color of the leaf is determined by the color of the outline and the color of the area enclosed by the outline.



When you edit a vector graphic, you modify the properties of the lines and curves that describe its shape. You can move, resize, reshape, and change the color of a vector graphic without changing the quality of its appearance. Vector graphics are resolution-independent, meaning they can be displayed on output devices of varying resolutions without losing any quality.

Bitmap graphics

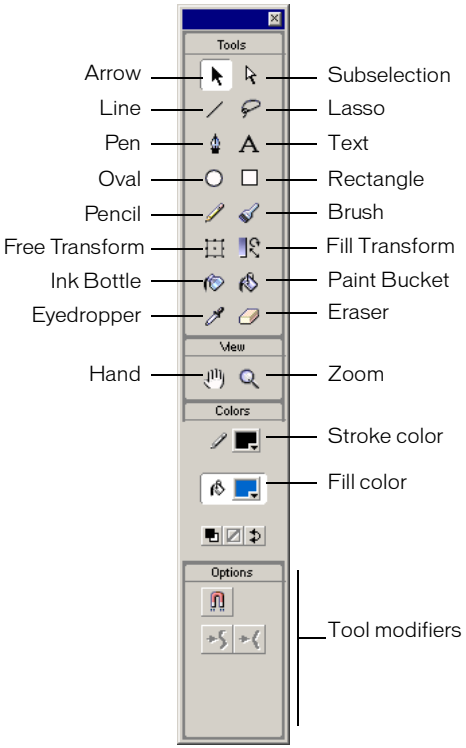
Bitmap graphics describe images using colored dots, called *pixels,* arranged within a grid. For example, the image of a leaf is described by the specific location and color value of each pixel in the grid, creating an image in much the same manner as a mosaic.



When you edit a bitmap graphic, you modify pixels, rather than lines and curves. Bitmap graphics are resolution-dependent, because the data describing the image is fixed to a grid of a particular size. Editing a bitmap graphic can change the quality of its appearance. In particular, resizing a bitmap graphic can make the edges of the image ragged as pixels are redistributed within the grid. Displaying a bitmap graphic on an output device that has a lower resolution than the image itself also degrades the quality of its appearance.

Flash drawing and painting tools

Flash provides various tools for drawing freeform or precise lines, shapes, and paths, and for painting filled objects.



- To draw freeform lines and shapes as if drawing with a real pencil, you use the Pencil tool. See "Drawing with the Pencil tool" on page 63.
- To draw precise paths as straight or curved lines, you use the Pen tool. See "Using the Pen tool" on page 64.
- To draw basic geometric shapes, you use the Line, Oval, and Rectangle tools. See "Drawing straight lines, ovals, and rectangles" on page 63.
- To create brushlike strokes as if painting with a brush, you use the Brush tool. See "Painting with the Brush tool" on page 69.

When you use most Flash tools, the Property inspector changes to present the settings associated with that tool. For example, if you choose the Text tool, the Property inspector displays text properties, making it easy to select the text attributes you want. For more information on the Property inspector, see "Panels and the Property inspector" on page 48.

When you use a drawing or painting tool to create an object, the tool applies the current stroke and fill attributes to the object. To change the stroke and fill attributes of existing objects, you can use the Paint Bucket and Ink Bottle tools in the toolbox or the Property inspector. See "Using the Stroke Color and Fill Color controls in the toolbox" on page 77 or "Using the Stroke Color and Fill Color controls in the Property inspector" on page 79.

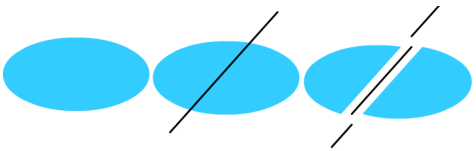
You can reshape lines and shape outlines in a variety of ways after you create them. Fills and strokes are treated as separate objects. You can select fills and strokes separately to move or modify them. See “Reshaping lines and shape outlines” on page 70.

You can use snapping to automatically align elements with each other and with the drawing grid or guides. See “Snapping” on page 74 and “Using the grid, guides, and rulers” on page 20.

About overlapping shapes in Flash

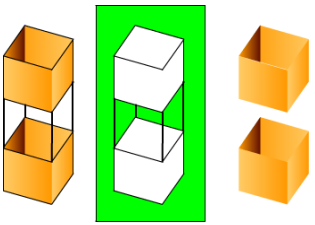
When you use the Pencil, Line, Oval, Rectangle, or Brush tool to draw a line across another line or painted shape, the overlapping lines are divided into segments at the intersection points. You can use the Arrow tool to select, move, and reshape each segment individually.

Note: Overlapping lines that you create with the Pen tool do not divide into individual segments at intersection points, but remain connected. See “Using the Pen tool” on page 64.



A fill; the fill with a line drawn through it; and the two fills and three line segments created by segmentation

When you paint on top of shapes and lines, the portion underneath is replaced by whatever is on top. Paint of the same color merges together. Paint of different colors remains distinct. You can use these features to create masks, cutouts, and other negative images. For example, the cutout below was made by moving the ungrouped kite image onto the green shape, deselecting the kite, and then moving the filled portions of the kite away from the green shape.



To avoid inadvertently altering shapes and lines by overlapping them, you can group the shapes or use layers to separate them. See “Grouping objects” on page 122 and “Using layers” on page 33.

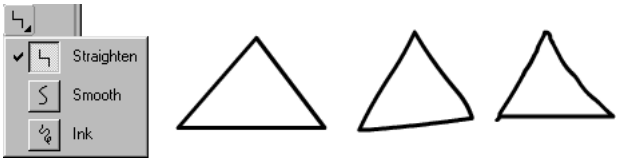
Drawing with the Pencil tool

To draw lines and shapes, you use the Pencil tool, in much the same way that you would use a real pencil to draw. To apply smoothing or straightening to the lines and shapes as you draw, you can select a drawing mode for the Pencil tool.

To draw with the Pencil tool:

1. Select the Pencil tool.
2. Select Window > Properties and select a stroke color, line weight, and style in the Property inspector. See "Using the Stroke Color and Fill Color controls in the Property inspector" on page 79.
3. Choose a drawing mode under Options in the toolbox:

- Choose Straighten to draw straight lines and convert approximations of triangles, ovals, circles, rectangles, and squares into these common geometric shapes.
- Choose Smooth to draw smooth curved lines.
- Choose Ink to draw freehand lines with no modification applied.



Lines drawn with Straighten, Smooth, and Ink mode, respectively

4. Drag on the Stage to draw with the Pencil tool. Shift-drag to constrain lines to vertical or horizontal directions.

Drawing straight lines, ovals, and rectangles

You can use the Line, Oval, and Rectangle tools to easily create these basic geometric shapes. The Oval and Rectangle tools create stroked and filled shapes. The Rectangle tool lets you create rectangles with square or rounded corners.

To draw a straight line, oval, or rectangle:

1. Select the Line, Oval, or Rectangle tool.
2. Select Window > Properties and select stroke and fill attributes in the Property inspector. See "Using the Stroke Color and Fill Color controls in the Property inspector" on page 79.

   Note: You cannot set fill attributes for the Line tool.

3. For the Rectangle tool, specify rounded corners by clicking the Round Rectangle modifier and entering a corner radius value. A value of zero creates square corners.

4 Drag on the Stage. If you are using the Rectangle tool, press the Up and Down Arrow keys while dragging to adjust the radius of rounded corners.

For the Oval and Rectangle tools, Shift-drag to constrain the shapes to circles and squares.

For the Line tool, Shift-drag to constrain lines to multiples of 45°.

Using the Pen tool

To draw precise paths as straight lines or smooth, flowing curves, you can use the Pen tool. You can create straight or curved line segments and adjust the angle and length of straight segments and the slope of curved segments.

When you draw with the Pen tool, you click to create points on straight line segments, and click and drag to create points on curved line segments. You can adjust straight and curved line segments by adjusting points on the line. You can convert curves to straight lines and the reverse. You can also display points on lines that you create with other Flash drawing tools, such as the Pencil, Brush, Line, Oval, or Rectangle tools, to adjust those lines. See "Reshaping lines and shape outlines" on page 70.

Setting Pen tool preferences

You can specify preferences for the appearance of the Pen tool pointer, for previewing line segments as you draw, or for the appearance of selected anchor points. Selected line segments and anchor points are displayed using the outline color of the layer on which the lines and points appear.

To set Pen tool preferences:

1 Choose Edit > Preferences and click the Editing tab.

2 Under Pen Tool, set the following options:

- Select Show Pen Preview to preview line segments as you draw. Flash displays a preview of the line segment as you move the pointer around the Stage, before you click to create the end point of the segment. If this option is not selected, Flash does not display a line segment until you create the end point of the segment.
- Select Show Solid Points to specify that unselected anchor points appear as solid points and selected anchor points appear as hollow points (this option is selected by default). Deselect this option to display unselected anchor points as hollow points and selected anchor points as solid points.
- Select Show Precise Cursors to specify that the Pen tool pointer appear as a cross-hair pointer, rather than the default Pen tool icon, for more precise placement of lines. Deselect the option to display the default Pen tool icon with the Pen tool.

  Note: Press the Caps Lock key while working to toggle between the cross-hair pointer and the default Pen tool icon.

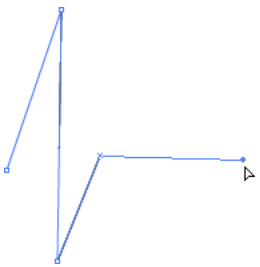
3 Click OK.

Drawing straight lines with the Pen tool

To draw straight line segments with the Pen tool, you create anchor points—points on the line that determine the length of individual line segments.

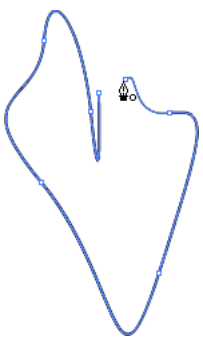
To draw straight lines with the Pen tool:

1. Select the Pen tool.
2. Select Window > Properties and select stroke and fill attributes in the Property inspector. See "Using the Stroke Color and Fill Color controls in the Property inspector" on page 79.
3. Position the pointer on the Stage where you want the straight line to begin, and click to define the first anchor point.
4. Click again where you want the first segment of the straight line to end. Shift-click to constrain lines to multiples of 45°.
5. Continue clicking to create additional straight segments.



6. To complete the path as an open or closed shape, do one of the following:

- To complete an open path, double-click the last point, click the Pen tool in the toolbox, or Control-click (Windows) or Command-click (Macintosh) anywhere away from the path.
- To close a path, position the Pen tool over the first anchor point. A small circle appears next to the pen tip when it is positioned correctly. Click or drag to close the path.



- To complete the shape as is, choose Edit > Deselect All or select a different tool in the toolbox.

Drawing curved paths with the Pen tool

You create curves by dragging the Pen tool in the direction you want the curve to go to create the first anchor point, and then dragging the Pen tool in the opposite direction to create the second anchor point.

When you use the Pen tool to create a curved segment, the anchor points of the line segment display tangent handles. The slope and length of each tangent handle determine the slope and the height, or depth, of the curve. Moving the tangent handles reshapes the curves of the path. See "Adjusting segments" on page 68.

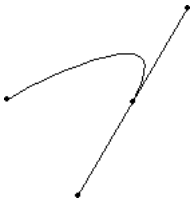
To draw a curved path:

1. Select the Pen tool.
2. Position the Pen tool on the Stage where you want the curve to begin, and hold down the mouse button.

   The first anchor point appears, and the pen tip changes to an arrowhead.
3. Drag in the direction you want the curve segment to be drawn. Shift-drag to constrain the tool to multiples of 45°.

   As you drag, the tangent handles of the curve appear.
4. Release the mouse button.

   The length and slope of the tangent handles determine the shape of the curve segment. You can move the tangent handles later to adjust the curve.
5. Position the pointer where you want the curve segment to end, hold down the mouse button, and drag in the opposite direction to complete the segment. Shift-drag to constrain the segment to multiples of 45°.

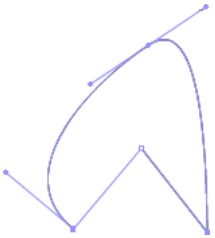


6. To draw the next segment of a curve, position the pointer where you want the next segment to end, and drag away from the curve.

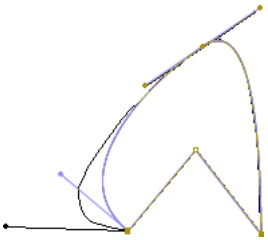
Adjusting anchor points on paths

When you draw a curve with the Pen tool, you create curve points—anchor points on a continuous, curved path. When you draw a straight line segment, or a straight line connected to a curved segment, you create corner points—anchor points on a straight path or at the juncture of a straight and a curved path.

By default, selected curve points appear as hollow circles, and selected corner points appear as hollow squares.



To convert segments in a line from straight segments to curve segments or the reverse, you convert corner points to curve points or the reverse.



You can also move, add, or delete anchor points on a path. You move anchor points using the Subselection tool to adjust the length or angle of straight segments or the slope of curved segments. You can nudge selected anchor points to make small adjustments.

Deleting unneeded anchor points on a curved path optimizes the curve and reduces the file size.

To move an anchor point:

Drag the point with the Subselection tool.

To nudge an anchor point or points:

Select the point or points with the Subselection tool and use the arrow keys to move the point or points.

To convert an anchor point, do one of the following:

- To convert a corner point to a curve point, use the Subselection tool to select the point, then Alt-drag (Windows) or Option-drag (Macintosh) the point to place the tangent handles.
- To convert a curve point to a corner point, click the point with the Pen tool.

To add an anchor point:

Click a line segment with the Pen tool.

To delete an anchor point, do one of the following:

To delete a corner point, click the point once with the Pen tool.

- To delete a curve point, click the point twice with the Pen tool. (Click once to convert the point to a corner point, and once more to delete the point.)
- Select the point with the Subselection tool and press Delete.

Adjusting segments

You can adjust straight segments to change the angle or length of the segment, or adjust curved segments to change the slope or direction of the curve.

When you move a tangent handle on a curve point, the curves on both sides of the point adjust. When you move a tangent handle on a corner point, only the curve on the same side of the point as the tangent handle adjusts.

To adjust a straight segment:

1. Select the Subselection tool, and select a straight segment.
2. Use the Subselection tool to drag an anchor point on the segment to a new position.

To adjust a curve segment:

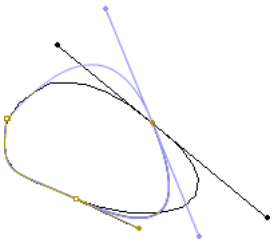
Select the Subselection tool and drag the segment.

Note: When you click the path, anchor points are revealed. Adjusting a segment with the Subselection tool may add points to the path.

To adjust points or tangent handles on a curve:

1. Select the Subselection tool, and select an anchor point on a curved segment.

   A tangent handle appears for the point you selected.
2. To adjust the shape of the curve on either side of the anchor point, drag the anchor point, or drag the tangent handle. Shift-drag to constrain the curve to multiples of 45°. Alt-drag (Windows) or Option-drag (Macintosh) to drag tangent handles individually.



Painting with the Brush tool

The Brush tool draws brushlike strokes, as if you were painting. It lets you create special effects, including calligraphic effects. You can choose a brush size and shape using the Brush tool modifiers. On most pressure-sensitive tablets, you can vary the width of the brush stroke by varying pressure on the stylus.

Brush size for new strokes remains constant even when you change the magnification level for the Stage, so the same brush size appears larger when the Stage magnification is lower. For example, suppose you set the Stage magnification to 100% and paint with the Brush tool using the smallest brush size. Then, you change the magnification to 50% and paint again using the smallest brush size. The new stroke that you paint appears 50% thicker than the earlier stroke. (Changing the magnification of the Stage does not change the size of existing brush strokes.)

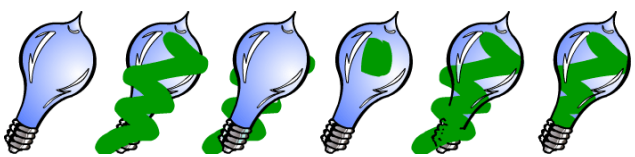
You can use an imported bitmap as a fill when painting with the Brush tool. See “Breaking apart groups and objects” on page 133.

A variable-width brush stroke drawn with a stylus

To paint with the Brush tool:

1. Select the Brush tool.
2. Select Window > Properties and select a fill color in the Property inspector. See “Using the Stroke Color and Fill Color controls in the Property inspector” on page 79.
3. Click the Brush Mode modifier and choose a painting mode:

- Paint Normal paints over lines and fills on the same layer.
- Paint Fills paints fills and empty areas, leaving lines unaffected.
- Paint Behind paints in blank areas of the Stage on the same layer, leaving lines and fills unaffected.
- Paint Selection applies a new fill to the selection when you select a fill in the Fill modifier or the Fill box of the Property inspector. (This option is the same as simply selecting a filled area and applying a new fill.)
- Paint Inside paints the fill in which you start a brush stroke and never paints lines. This works much like a smart coloring book that never allows you to paint outside the lines. If you start painting in an empty area, the fill doesn't affect any existing filled areas.



Original image, Paint Normal, Paint Behind, Paint Selection, Paint Fills, and Paint Inside

4. Choose a brush size and brush shape from the Brush tool modifiers.

5 If a pressure-sensitive tablet is attached to your computer, you can select the Pressure modifier to vary the width of your brush strokes by varying the pressure on your stylus.

6 Drag on the Stage. Shift-drag to constrain brush strokes to horizontal and vertical directions.

Reshaping lines and shape outlines

You can reshape lines and shape outlines created with the Pencil, Brush, Line, Oval, or Rectangle tools by dragging with the Arrow tool, or by optimizing their curves.

You can also use the Subselection tool to display points on lines and shape outlines and modify the lines and outlines by adjusting the points. For information on adjusting anchor points, see "Using the Pen tool" on page 64.

To display anchor points on a line or shape outline created with the Pencil, Brush, Line, Oval, or Rectangle tools:

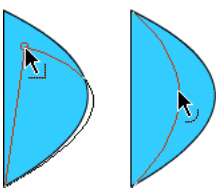
1 Select the Subselection tool.

2 Click the line or shape outline.

Reshaping using the Arrow tool

To reshape a line or shape outline, you can drag on any point on a line using the Arrow tool. The pointer changes to indicate what type of reshaping it can perform on the line or fill.

Flash adjusts the curve of the line segment to accommodate the new position of the moved point. If the repositioned point is an end point, you can lengthen or shorten the line. If the repositioned point is a corner, the line segments forming the corner remain straight as they become longer or shorter.



When a corner appears next to the pointer, you can change an end point. When a curve appears next to the pointer, you can adjust a curve.

Some brush stroke areas are easier to reshape if you view them as outlines.

If you are having trouble reshaping a complex line, you can smooth it to remove some of its details, making reshaping easier. Increasing the magnification can also make reshaping easier and more accurate; see "Optimizing curves" on page 72 or "Viewing the Stage" on page 19.

To reshape a line or shape outline using the Arrow tool:

1 Select the Arrow tool.

2 Do one of the following:

- Drag from any point on the segment to reshape it.
- Control-drag (Windows) or Option-drag (Macintosh) a line to create a new corner point.

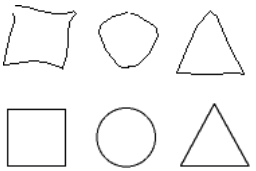
Straightening and smoothing lines

You can reshape lines and shape outlines by straightening or smoothing them.

Note: You can adjust the degree of automatic smoothing and straightening by choosing preferences for drawing settings. See "Choosing drawing settings" on page 75.

Straightening makes small straightening adjustments to lines and curves you have already drawn. It has no effect on already straight segments.

You can also use the straightening technique to make Flash recognize shapes. If you draw any oval, rectangular, or triangular shapes with the Recognize Shapes option turned off, you can use the Straightening option to make the shapes geometrically perfect. (For information on the Recognize Shapes option, see "Choosing drawing settings" on page 75.) Shapes that are touching, and thus connected to other elements, are not recognized.



Shape recognition turns the top shapes into the bottom shapes.

Smoothing softens curves and reduces bumps or other variations in a curve's overall direction. It also reduces the number of segments in a curve. Smoothing is relative, however, and has no effect on straight segments. It is particularly useful when you are having trouble reshaping a number of very short curved line segments. Selecting all the segments and smoothing them reduces the number of segments, producing a gentler curve that is easier to reshape.

Repeated application of smoothing or straightening makes each segment smoother or straighter, depending on how curved or straight each segment was originally.

To smooth the curve of each selected fill outline or curved line:

Select the Arrow tool and click the Smooth modifier in the Options section of the toolbox, or choose Modify > Smooth.

To make small straightening adjustments on each selected fill outline or curved line:

Select the Arrow tool and click the Straighten modifier in the Options section of the toolbox, or choose Modify > Straighten.

To use shape recognition:

Select the Arrow tool and click the Straighten modifier, or choose Modify > Straighten.

Optimizing curves

Another way to smooth curves is to optimize them. This refines curved lines and fill outlines by reducing the number of curves used to define these elements. Optimizing curves also reduces the size of the Flash document (FLA file) and the exported Flash movie (SWF file). As with the Smooth or Straighten modifiers or commands, you can apply optimization to the same elements multiple times.

To optimize curves:

1. Select the drawn elements to be optimized and choose Modify > Optimize.
2. In the Optimize Curves dialog box, drag the Smoothing slider to specify the degree of smoothing.

   The exact results depend on the curves selected. Generally, optimizing produces fewer curves, with less resemblance to the original outline.
3. Set the additional options:

- Select Use Multiple Passes to repeat the smoothing process until no further optimization can be accomplished; this is the same as repeatedly choosing Optimize with the same elements selected.
- Select Show Totals Message to display an alert box that indicates the extent of the optimization when smoothing is complete.

4. Click OK.

Erasing

Erasing with the Eraser tool removes strokes and fills. You can quickly erase everything on the Stage, erase individual stroke segments or filled areas, or erase by dragging.

You can customize the Eraser tool to erase only strokes, only filled areas, or only a single filled area. The Eraser tool can be either round or square, and it can have one of five sizes.

To quickly delete everything on the Stage:

Double-click the Eraser tool.

To remove stroke segments or filled areas:

1. Select the Eraser tool and then click the Faucet modifier.
2. Click the stroke segment or filled area that you want to delete.

To erase by dragging:

1. Select the Eraser tool.
2. Click the Eraser Mode modifier and choose an erasing mode:

- Erase Normal erases strokes and fills on the same layer.
- Erase Fills erases only fills; strokes are not affected.
- Erase Lines erases only strokes; fills are not affected.
- Erase Selected Fills erases only the currently selected fills and does not affect strokes, selected or not. (Select the fills you want to erase before using the Eraser tool in this mode.)

- Erase Inside erases only the fill on which you begin the eraser stroke. If you begin erasing from an empty point, nothing will be erased. Strokes are unaffected by the eraser in this mode.

3 Click the Eraser Shape modifier and choose an eraser shape and size. Make sure that the Faucet modifier is not selected.

4 Drag on the Stage.

Modifying shapes

You can modify shapes by converting lines to fills, expanding the shape of a filled object, or softening the edges of a filled shape by modifying the curves of the shape.

The Convert Lines to Fills feature changes lines to fills, which allows you to fill lines with gradients or to erase a portion of a line. The Expand Shape and Soften Edges features allow you to expand filled shapes and blur the edges of shapes.

The Expand Fill and Soften Fill Edges features work best on small shapes that do not contain many small details. Applying Soften Edges to shapes with extensive detail can increase the file size of a Flash document and the resulting SWF file.

To convert lines to fills:

1 Select a line or multiple lines.

2 Choose Modify > Shape > Convert Lines to Fills.

Selected lines are converted to filled shapes. Converting lines to fills can make file sizes larger, but it can also speed up drawing for some animations.

To expand the shape of a filled object:

1 Select a filled shape. This command works best on a single filled color shape with no stroke.

2 Choose Modify > Shape > Expand Fill.

3 In the Expand Path dialog box, enter a value in pixels for Distance and select Expand or Inset for Direction. Expand enlarges the shape, and Inset reduces it.

To soften the edges of an object:

1 Select a filled shape.

Note: This feature works best on a single filled shape that has no stroke.

2 Choose Modify > Shape > Soften Fill Edges.

3 Set the following options:

- Distance is the width in pixels of the soft edge.
- Number of Steps controls how many curves will be used for the soft edge effect. More steps will provide a smoother effect but will also create larger files and be slower to draw.
- Expand or Inset controls whether the shape will be enlarged or reduced to soften the edges.

Snapping

To automatically align elements with one another, you can use snapping. Flash lets you align objects by snapping to other objects or by snapping to individual pixels.

Note: You can also snap to the grid or to guides. For more information, see "Using the grid, guides, and rulers" on page 20.

Object snapping

Object snapping can be turned on using the Snap modifier for the Arrow tool, or the Snap to Objects command in the View menu.

If the Snap modifier for the Arrow tool is on, a small black ring appears under the pointer when you drag an element. The small ring changes to a larger ring when the object is within snapping distance of another object.

To turn object snapping on or off:

Choose View > Snap to Objects. A check mark is displayed next to the command when it is on.

When you move or reshape an object, the position of the Arrow tool on the object provides the reference point for the snap ring. For example, if you move a filled shape by dragging near its center, the center point snaps to other objects. This is particularly useful for snapping shapes to motion paths for animating.

Note: For better control of object placement when snapping, begin dragging from a corner or center point.

To adjust object snapping tolerances:

1. Choose Edit > Preferences and click the Editing tab.
2. Under Drawing Settings, adjust the Connect Lines setting. See "Choosing drawing settings" on page 75.

Pixel snapping

You can turn on pixel snapping using the Snap to Pixels command in the View menu. If Snap to Pixels is on, a pixel grid appears when the view magnification is set to 400% or higher. The pixel grid represents the individual pixels that will appear in your movie. When you create or move an object, it is constrained to the pixel grid.

To turn pixel snapping on or off:

Choose View > Snap to Pixels.

If magnification is set to 400% or higher, a pixel grid is displayed. A check mark is displayed next to the command when it is on.

To turn pixel snapping on or off temporarily:

Press the C key. When you release the C key, pixel snapping returns to the state you selected with View > Snap to Pixels.

To temporarily hide the pixel grid:

Press the X key. When you release the X key, the pixel grid reappears.

Choosing drawing settings

You can set drawing settings to specify snapping, smoothing, and straightening behaviors when you use Flash drawing tools. You can change the Tolerance setting for each option, and turn each option off or on. Tolerance settings are relative, depending on the resolution of your computer screen and the current magnification of the scene. By default, each option is turned on and set to Normal tolerance.

To set drawing settings:

1. Choose Edit > Preferences and click the Editing tab.
2. Under Drawing Settings, choose from the following options:

- Connect Lines determines how close the end of a line being drawn must be to an existing line segment before the end point snaps to the nearest point on the other line. The available options are Must Be Close, Normal, and Can Be Distant. This setting also controls horizontal and vertical line recognition—that is, how nearly horizontal or vertical a line must be drawn before Flash makes it exactly horizontal or vertical. When Snap to Objects is turned on, this setting controls how close objects must be to snap to one another.
- Smooth Curves specifies the amount of smoothing applied to curved lines drawn with the Pencil tool when the drawing mode is set to Straighten or Smooth. (Smoother curves are easier to reshape, while rougher curves match more closely the original line strokes.) The selections are Off, Rough, Normal, and Smooth.

  Note: You can further smooth existing curved segments using Modify › Smooth and Modify › Optimize.
- Recognize Lines defines how nearly straight a line segment drawn with the Pencil tool must be before Flash recognizes it and makes it perfectly straight. The selections are Off, Strict, Normal, and Tolerant. If Recognize Lines is off while you draw, you can straighten lines later by selecting one or more line segments and choosing Modify > Straighten.
- Recognize Shapes controls how precisely you must draw circles, ovals, squares, rectangles, and 90° and 180° arcs for them to be recognized as geometric shapes and redrawn accurately. The options are Off, Strict, Normal, and Tolerant. If Recognize Shapes is off while you draw, you can straighten lines later by selecting one or more shapes (for example, connected line segments) and choosing Modify > Straighten.
- Click Accuracy specifies how close to an item the pointer must be before Flash recognizes the item. The options are Strict, Normal, and Tolerant.

CHAPTER 4
Working with Color

Macromedia Flash MX provides a variety of ways to apply, create, and modify colors. Using the default palette or a palette you create, you can choose colors to apply to the stroke or fill of an object you are about to create, or one already on the Stage. Applying a stroke color to a shape paints the outline of the shape with that color. Applying a fill color to a shape paints the interior space of the shape with that color.

When applying a stroke color to a shape, you can select any solid color, and you can select the style and weight of the stroke. For a shape's fill, you can apply a solid color, gradient, or bitmap. To apply a bitmap fill to a shape, you must import a bitmap into the current file. You can also create an outlined shape with no fill by using None as a fill, or a filled shape with no outline by using None as an outline. And you can apply a solid color fill to text. See "Setting text attributes" on page 139.

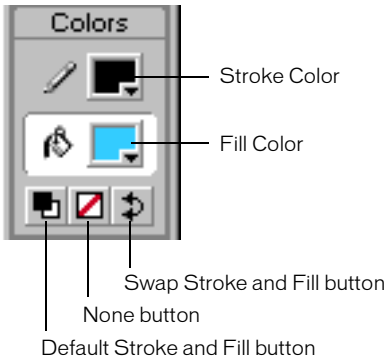
You can modify stroke and fill attributes in a variety of ways using the Paint Bucket, Ink Bottle, Eyedropper, and Fill Transform tools, and the Lock Fill modifier for the Brush or Paint Bucket tools.

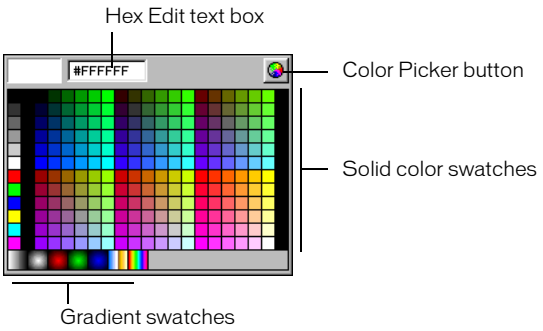
With the Color Mixer you can easily create and edit solid colors and gradient fills in RGB and HSB modes. You can import, export, delete, and otherwise modify the color palette for a file using the Color Swatches panel. You can select colors in Hexadecimal mode in the mixer, as well as in the Stroke and Fill pop-up windows in the toolbox or Property inspector.

Using the Stroke Color and Fill Color controls in the toolbox

The Stroke Color and Fill Color controls in the toolbox let you select a solid stroke color or a solid or gradient fill color, switch the stroke and fill colors, or select the default stroke and fill colors (black stroke and white fill). Oval and rectangle objects (shapes) can have both stroke and fill colors. Text objects and brush strokes can have only fill colors. Lines drawn with the Line, Pen, and Pencil tools can have only stroke colors.

The toolbox Stroke Color and Fill Color controls set the painting attributes of new objects you create with the drawing and painting tools. To use these controls to change the painting attributes of existing objects, you must first select the objects on the Stage.





Note: Gradient swatches appear only in the Fill Color control.

To apply stroke and fill colors using the toolbox controls, do one of the following:

- Click the triangle next to the Stroke or Fill color box and choose a color swatch from the pop-up window. Gradients can be selected for the fill color only.
- Click the Color Picker button in the color pop-up window and choose a color from the Color Picker.
- Type a color's hexadecimal value in the text box in the color pop-up window.
- Click the Default Fill and Stroke button in the toolbox to return to the default color settings (white fill and black stroke).
- Click the None button in the color pop-up window to remove any stroke or fill.

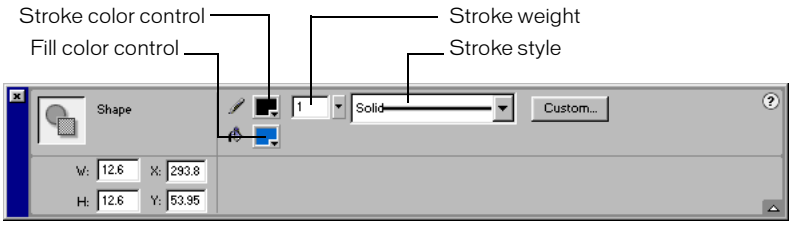
  Note: The None button appears only when you are creating a new oval or rectangle. You can create a new object without a stroke or fill, but you cannot use the None button with an existing object. Instead, select the existing stroke or fill and delete it.

- Click the Swap Fill and Stroke button in the toolbox to swap colors between the fill and the stroke.

Using the Stroke Color and Fill Color controls in the Property inspector

To change the stroke color, style, and weight for a selected object, you can use the Stroke Color controls in the Property inspector. For stroke style, you can choose from styles that are preloaded with Flash, or you can create a custom style.

To select a solid color fill, you can use the Fill Color control in the Property inspector.



To select a stroke color, style, and weight using the Property inspector:

1. Select an object or objects on the Stage.
2. If the Property inspector is not visible, choose Window > Properties.
3. To select a color, click the triangle next to the Stroke color box and do one of the following:

- Choose a color swatch from the palette.
- Type a color's hexadecimal value in the text box.

4. To select a stroke style, click the triangle next to the Style pop-up menu and choose an option from the menu. To create a custom style, choose Custom from the Property inspector, then choose options in the Stroke Style dialog box and click OK.

   Note: Choosing a stroke style other than Solid can increase file size.

5. To select a stroke weight, click the triangle next to the Weight pop-up menu and set the slider at the desired weight.

To apply a solid color fill using the Property inspector:

1. Select an object or objects on the Stage.
2. Choose Window >Properties.
3. To select a color, click the triangle next to the Fill color box and do one of the following:

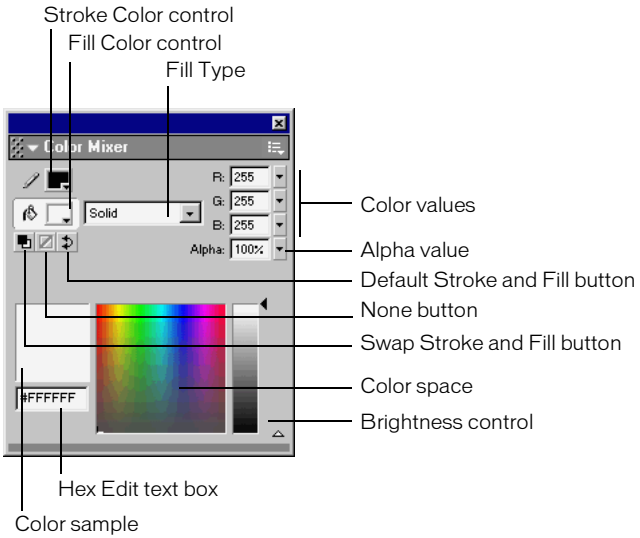
- Choose a color swatch from the palette.
- Type a color's hexadecimal value in the text box.

Working with solid colors and gradient fills in the Color Mixer

To create and edit solid colors and gradient fills, you can use the Color Mixer. If an object is selected on the Stage, the color modifications you make in the Color Mixer are applied to the selection.

You can create any color using the Color Mixer. You can choose colors in RGB, HSB, or expand the panel to use hexadecimal mode. You can also specify an alpha value to define the degree of transparency for a color. In addition, you can select a color from the existing color palette.

You can expand the Color Mixer to display a larger color space in place of the color bar, a split color swatch showing the current and previous colors, and a Brightness control to modify color brightness in all color modes.



To create or edit a solid color with the Color Mixer:

1. To apply the color to existing artwork, select an object or objects on the Stage.
2. Choose Window > Color Mixer.
3. To select a color mode display, choose RGB (the default setting) or HSB from the pop-up menu in the upper right corner of the Color Mixer.
4. Click the Stroke or Fill icon to specify which attribute is to be modified.

   Note: Be sure to click the icon, not the color box, or the color pop-up window will open.
5. If you selected the Fill icon in step 4, verify that Solid is selected in the Fill Type pop-up menu in the center of the Mixer.
6. Click the arrow in the lower right corner to expand the Color Mixer.

7 Do one of the following:

- Click in the color space in the Color Mixer to select a color. Drag the Brightness control to adjust the brightness of the color.

  Note: To create colors other than black or white, make sure the Brightness control is not set to either extreme.

- Enter values in the color value boxes: Red, Green, and Blue values for RGB display; Hue, Saturation, and Brightness values for HSB display; or hexadecimal values for hexadecimal display. Enter an Alpha value to specify the degree of transparency, from 0 for complete transparency to 100 for complete opacity.
- Click the Default Stroke and Fill button to return to the default color settings (white fill and black stroke).
- Click the Swap Stroke and Fill button to swap colors between the fill and the stroke.
- Click the None button to apply no color to the fill or stroke.

  Note: You cannot apply a stroke or fill of None to an existing object. Instead, select the existing stroke or fill and delete it.

- Click the Stroke or Fill color box and choose a color from the pop-up window.

8 To add the color defined in step 5 to the color swatch list for the current document, choose Add Swatch from the pop-up menu in the upper right corner of the Color Mixer.

To create or edit a gradient fill with the Color Mixer:

1 To apply a gradient fill to existing artwork, select an object or objects on the Stage.

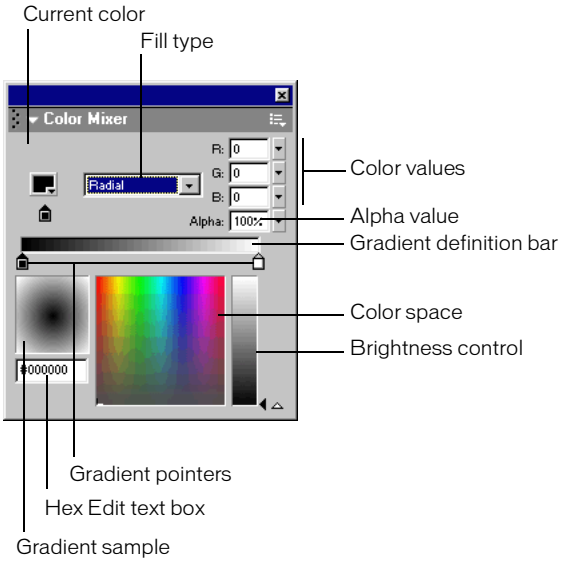
2 If the Color Mixer is not visible, choose Window > Color Mixer.

3 To select a color mode display, choose RGB (the default setting) or HSB.

4 Select a gradient type from the Fill Type pop-up menu in the center of the Color Mixer:

- Linear Gradient creates a gradient that shades from the starting point to the end point in a straight line.
- Radial Gradient creates a gradient that shades from the starting point to the end point in a circular pattern.

The gradient definition bar appears in place of the color bar in the Color Mixer, with pointers below the bar indicating each color in the gradient.



5 Click the arrow in the lower right corner to expand the Color Mixer.

6 To change a color in the gradient, click one of the pointers below the gradient definition bar and click in the color space that appears directly below the gradient bar in the expanded Color Mixer. Drag the Brightness control to adjust the lightness of the color.

7 To add a pointer to the gradient, click on or below the gradient definition bar. Select a color for the new pointer as described in step 6.

8 To reposition a pointer on the gradient, drag the pointer along the gradient definition bar. Drag a pointer down and off of the gradient definition bar to remove it.

9 To save the gradient, click the triangle in the upper right corner of the Color Mixer and choose Add Swatch from the pop-up menu. The gradient is added to the Color Swatches panel for the current document.

Modifying strokes with the Ink Bottle tool

To change the stroke color, width, and style of lines or shape outlines, you can use the Ink Bottle tool. You can apply only solid colors, not gradients or bitmaps, to lines or shape outlines.

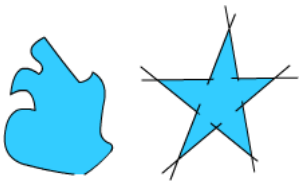
Using the Ink Bottle tool, rather than selecting individual lines, makes it easier to change the stroke attributes of multiple objects at one time.

To use the Ink Bottle tool:

1. Select the Ink Bottle tool from the toolbox.
2. Choose a stroke color as described in "Using the Stroke Color and Fill Color controls in the toolbox" on page 77.
3. Choose a stroke style and stroke width from the Property inspector. See "Using the Stroke Color and Fill Color controls in the Property inspector" on page 79.
4. Click an object on the Stage to apply the stroke modifications.

Applying solid, gradient, and bitmap fills with the Paint Bucket tool

The Paint Bucket tool fills enclosed areas with color. It can both fill empty areas and change the color of already painted areas. You can paint with solid colors, gradient fills, and bitmap fills. You can use the Paint Bucket tool to fill areas that are not entirely enclosed, and you can specify that Flash close gaps in shape outlines when you use the Paint Bucket tool. For information on applying a bitmap fill, see "Working with imported bitmaps" under Help > Using Flash.



The shape on the left is not fully enclosed but can still be filled. The star shape consists of individual lines that enclose an area that can be filled.

To use the Paint Bucket tool to fill an area:

1. Select the Paint Bucket tool from the toolbox.
2. Choose a fill color and style, as described in "Using the Stroke Color and Fill Color controls in the Property inspector" on page 79.
3. Click the Gap Size modifier and choose a gap size option:

- Choose Don't Close Gaps if you want to close gaps manually before filling the shape. Closing gaps manually can be faster for complex drawings.
- Choose a Close option to have Flash fill a shape that has gaps.

Note: If gaps are too large, you might have to close them manually.

4. Click the shape or enclosed area that you want to fill.

Transforming gradient and bitmap fills

You can transform a gradient or bitmap fill by adjusting the size, direction, or center of the fill. To transform a gradient or bitmap fill, you use the Fill Transform tool.

To adjust a gradient or bitmap fill with the Fill Transform tool:

1 Select the Fill Transform tool.

2 Click an area filled with a gradient or bitmap fill.

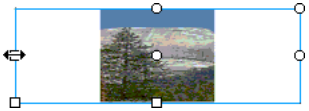
When you select a gradient or bitmap fill for editing, its center point appears, and its bounding box is displayed with editing handles. When the pointer is over any one of these handles, it changes to indicate the function of the handle.

Press Shift to constrain the direction of a linear gradient fill to multiples of 45°.

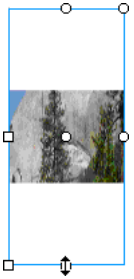
3 Reshape the gradient or fill in any of the following ways:

- To reposition the center point of the gradient or bitmap fill, drag the center point.

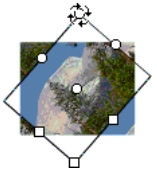
- To change the width of the gradient or bitmap fill, drag the square handle on the side of the bounding box. (This option resizes only the fill, not the object containing the fill.)



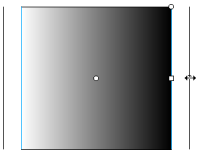
- To change the height of the gradient or bitmap fill, drag the square handle at the bottom of the bounding box.



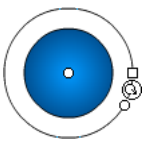
- To rotate the gradient or bitmap fill, drag the circular rotation handle at the corner. You can also drag the lowest handle on the bounding circle of a circular gradient or fill.



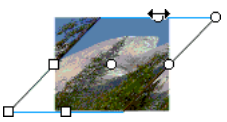
- To scale a linear gradient or a fill, drag the square handle at the center of the bounding box.



- To change the radius of a circular gradient, drag the middle circular handle on the bounding circle.



- To skew or slant a fill within a shape, drag one of the circular handles on the top or right side of the bounding box.



- To tile a bitmap inside a shape, scale the fill.

Note: To see all of the handles when working with large fills or fills close to the edge of the Stage, choose View › Work Area.

Copying strokes and fills with the Eyedropper tool

You can use the Eyedropper tool to copy fill and stroke attributes from one object and immediately apply them to another object. The Eyedropper tool also lets you sample the image in a bitmap to use as a fill. See "Breaking apart groups and objects" on page 133.

To use the Eyedropper tool to copy and apply stroke or fill attributes:

1. Select the Eyedropper tool and click the stroke or filled area whose attributes you want to apply to another stroke or filled area.

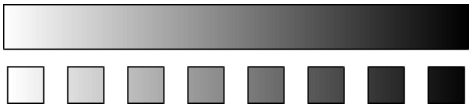
   When you click a stroke, the tool automatically changes to the Ink Bottle tool. When you click a filled area, the tool automatically changes to the Paint Bucket tool and the Lock Fill modifier is turned on. See "Locking a gradient or bitmap to fill the Stage" on page 86.

2. Click another stroke or filled area to apply the new attributes.

Locking a gradient or bitmap to fill the Stage

You can lock a gradient or bitmap fill to make it appear that the fill extends over the entire Stage and that the objects painted with the fill are masks revealing the underlying gradient or bitmap. For information on applying a bitmap fill, see "Working with imported bitmaps" under Help > Using Flash.

When you select the Lock Fill modifier with the Brush or Paint Bucket tool and paint with the tool, the bitmap or gradient fill extends across the objects you paint on the Stage.



Using the Lock Fill modifier creates the appearance of a single gradient or bitmap fill being applied to separate objects on the Stage.

To use a locked gradient fill:

1. Select the Brush or Paint Bucket tool and choose a gradient or bitmap as a fill.
2. Select a Linear Gradient or Radial Gradient from the Fill Type pop-up menu in the center of the Color Mixer before selecting the Brush or Paint Bucket tool.
3. Click the Lock Fill modifier.
4. First paint the areas where you want to place the center of the fill, and then move to other areas.

To use a locked bitmap fill:

1. Select the bitmap you want to use.
2. Select Bitmap from the Fill Type pop-up menu in the center of the Color Mixer before selecting the Brush or Paint Bucket tool.
3. Select the Brush or Paint Bucket tool.
4. Click the Lock Fill modifier.
5. First paint the areas where you want to place the center of the fill, and then move to other areas.

Modifying color palettes

Each Flash file contains its own color palette, stored in the Flash document. Flash displays a file's palette as swatches in the Fill Color and Stroke Color controls and in the Color Swatches panel. The default color palette is the Web-safe palette of 216 colors. You can add colors to the current color palette using the Color Mixer. See "Working with solid colors and gradient fills in the Color Mixer" on page 80.

To import, export, and modify a file's color palette, you use the Color Swatches panel. You can duplicate colors, remove colors from the palette, change the default palette, reload the Web-safe palette if you have replaced it, or sort the palette according to hue.

You can import and export both solid and gradient color palettes between Flash files, as well as between Flash and other applications, such as Macromedia Fireworks and Adobe Photoshop.

Duplicating and removing colors

You can duplicate colors in the palette, delete individual colors, or clear all colors from the palette.

To duplicate a color or delete a color:

1. If the Color Swatches panel is not visible, choose Window > Color Swatches.
2. Click the color that you want to duplicate or delete.
3. Choose Duplicate Swatch or Delete Swatch from the pop-up menu in the upper right corner.

To clear all colors from the color palette:

In the Color Swatches panel, choose Clear Colors from the pop-up menu in the upper right corner. All colors are removed from the palette except black and white.

Using the default palette and the Web-safe palette

You can save the current palette as the default palette, replace the current palette with the default palette defined for the file, or load the Web-safe palette to replace the current palette.

To load or save the default palette:

In the Color Swatches panel, choose one of the following commands from the pop-up menu in the upper right corner:

- Load Default Colors replaces the current palette with the default palette.
- Save as Default saves the current color palette as the default palette. The new default palette is used when you create new files.

To load the Web-safe 216-color palette:

In the Color Swatches panel, choose Web 216 from the pop-up menu in the upper right corner.

Sorting the palette

To make it easier to locate a color, you can sort colors in the palette by hue.

To sort colors in the palette:

1. In the Color Swatches panel, choose Sort by Color from the pop-up menu in the upper right corner.

Importing and exporting color palettes

To import and export both RGB colors and gradients between Flash files, you use Flash Color Set files (CLR files). You can import and export RGB color palettes using Color Table files (ACT files) that can be used with Macromedia Fireworks and Adobe Photoshop. You can also import color palettes, but not gradients, from GIF files. You cannot import or export gradients from ACT files.

To import a color palette:

1. In the Color Swatches panel, choose one of the following commands from the pop-up menu in the upper right corner:

- To append the imported colors to the current palette, choose Add Colors.
- To replace the current palette with the imported colors, choose Replace Colors.

2. Navigate to the desired file and select it.
3. Click OK.

To export a color palette:

1. In the Color Swatches panel, choose Save Colors from the pop-up menu in the upper right corner.
2. In the dialog box that appears, enter a name for the color palette.
3. For Save As Type (Windows) or Format (Macintosh), choose Flash Color Set or Color Table. Click Save.

CHAPTER 5
Using Imported Artwork and Video

Macromedia Flash MX can use artwork created in other applications. You can import vector graphics and bitmaps in a variety of file formats. If you have QuickTime 4 or later installed on your system, you can import additional vector or bitmap file formats. For more information, see "Import file formats for vector or bitmap files" on page 91. You can import FreeHand files (version 10 or earlier) and Fireworks PNG files directly into Flash, preserving attributes from those formats.

When you import a bitmap, you can apply compression and anti-aliasing, place the bitmap directly in a Flash document, use the bitmap as a fill, edit the bitmap in an external editor, break the bitmap apart into pixels and edit it in Flash, or convert the bitmap to vector artwork. See "Working with imported bitmaps" on page 96.

You can also import video into Flash. You can import files in Macromedia Flash Video format (FLV files) directly into Flash. For information on the FLV file format, see Chapter 21, "Exporting," on page 395.

If you have QuickTime 4 or later (Windows or Macintosh) or DirectX 7 or later (Windows only) installed on your system, you can import video in MOV, AVI, or MPEG format. Additional formats may be supported for import, depending on your system. Video clips can be imported as linked or embedded files. You can publish movies with imported video as SWF files or QuickTime movies. See "Importing video" on page 100.

For information on importing sound files in WAV (Windows), AIFF (Macintosh), and MP3 (both platforms) formats, see "Importing sounds" on page 109.

Placing artwork into Flash

Flash recognizes a variety of vector and bitmap formats. You can place artwork into Flash by importing it onto the Stage in the current Flash document or into the library for the current document. You can also import bitmaps by pasting them on the Stage in the current document. All bitmaps that you import directly into a Flash document are automatically added to the document's library.

Graphic files that you import into Flash must be at least 2 pixels by 2 pixels in size.

You can load JPEG files into a movie during runtime using the `loadMovie` action or method. For detailed information, see `loadMovie` in the online ActionScript Dictionary in the Help menu.

Flash imports vector graphics, bitmaps, and sequences of images as follows:

- When you import vector images into Flash from FreeHand, you can choose options for preserving FreeHand layers, pages, and text blocks. See "Importing FreeHand files" on page 93.
- When you import PNG images from Fireworks, you can import files as editable objects that you can modify in Flash, or as flattened files that you can edit and update in Fireworks.
- You can choose options for preserving images, text, and guides. See "Importing Fireworks PNG files" on page 92.

  Note: If you import a PNG file from Fireworks by cutting and pasting, the file is converted to a bitmap.

- When you import vector images into Flash from Adobe Illustrator, you can choose options for preserving Illustrator layers. See "Importing Adobe Illustrator files" on page 95.
- Vector images from SWF and Windows Metafile Format (WMF) files that you import directly into a Flash document (instead of into a library) are imported as a group in the current layer. See "Import file formats for vector or bitmap files" on page 91 and "Importing Adobe Illustrator files" on page 95.
- Bitmaps (scanned photographs, BMP files) that you import directly into a Flash document are imported as single objects in the current layer. Flash preserves the transparency settings of imported bitmaps. Because importing a bitmap can increase the file size of a Flash movie (SWF file), you may want to compress imported bitmaps. See "Setting bitmap properties" on page 97.

  Note: Bitmap transparency may not be preserved when bitmaps are imported by dragging and dropping. To preserve transparency, use the File › Import command for importing.

- Any sequence of images (for example, a PICT and BMP sequence) that you import directly into a Flash document is imported as successive keyframes of the current layer.

For information on specific file formats, see "Import file formats for vector or bitmap files" on page 91.

To import a file into Flash:

1 Do one of the following:

- To import a file directly into the current Flash document, choose File > Import.
- To import a file into the library for the current Flash document, choose File > Import to Library. (To use a library item in a document, drag it onto the Stage. See Chapter 9, "Using Symbols, Instances, and Library Assets," on page 149.

2 In the Import dialog box, choose a file format from the Files of Type (Windows) or Show (Macintosh) pop-up menu.

3 Navigate to the desired file and select it.

If an imported file has multiple layers, Flash may create new layers (depending on the import file type). Any new layers will be displayed in the Timeline.

Note: If you are importing a Fireworks PNG file, see "Importing Fireworks PNG files" on page 92. If you are importing a FreeHand file, see "Importing FreeHand files" on page 93. If you are importing an Adobe Illustrator file, see "Importing Adobe Illustrator files" on page 95.

4 Do one of the following:

- In Windows or Macintosh OS 10 or later, click Open.
- In Macintosh OS 9.X or earlier, click Add to add the selected file to the Import list, and click Import to import the file or files in the Import list.

5 If the name of the file you are importing ends with a number, and there are additional sequentially numbered files in the same folder, choose whether to import the sequence of files.

- Click Yes to import all of the sequential files.
- Click No to import only the specified file.

The following are examples of filenames that can be used as a sequence:

Frame001.gif, Frame002.gif, Frame003.gif

Bird 1, Bird 2, Bird 3

Walk-001.ai, Walk-002.ai, Walk-003.ai

To paste a bitmap from another application directly into the current Flash document:

1 Copy the image in the other application.

2 In Flash, choose Edit > Paste.

Import file formats for vector or bitmap files

Flash MX can import different vector or bitmap file formats depending on whether QuickTime 4 or later is installed on your system. Using Flash with QuickTime 4 installed is especially useful for collaborative projects in which authors work on both Windows and Macintosh platforms. QuickTime 4 extends support for certain file formats (including Adobe Photoshop, PICT, QuickTime Movie, and others) to both platforms.

The tables in this section list import file formats supported for vector or bitmap files. For information on import file formats supported for video clips, see "Importing video" on page 100.

The following vector or bitmap file formats can be imported into Flash MX, regardless of whether QuickTime 4 is installed:

File type	Extension	Windows	Macintosh
Adobe Illustrator (version 8 or earlier; see "Importing Adobe Illustrator files" on page 95)	.eps, .ai	✔	✔
AutoCAD DXF (see "AutoCAD DXF files" on page 95)	.dxf	✔	✔
Bitmap	.bmp	✔	✔ (Using QuickTime)
Enhanced Windows Metafile	.emf	✔	
FreeHand	.fh7, .fh7, .fh8, .fh8, .fh9, .fh9, .fh10	✔	✔
FutureSplash Player	.spl	✔	✔
GIF and animated GIF	.gif	✔	✔
JPEG	.jpg	✔	✔

File type	Extension	Windows	Macintosh
PICT	.pct, .pic		✔
PNG	.png	✔	✔
Flash Player 6	.swf	✔	✔
Windows Metafile	.wmf	✔	

The following vector or bitmap file formats can be imported into Flash MX only if QuickTime 4 or later is installed:

File type	Extension	Windows	Macintosh
MacPaint	.pntg	✔	✔
Photoshop	.psd	✔	✔
PICT	.pct, .pic	✔ (As bitmap)	
QuickTime Image	.qtif	✔	✔
Silicon Graphics Image	.sgi	✔	✔
TGA	.tga	✔	✔
TIFF	.tif	✔	✔

Importing Fireworks PNG files

You can import Fireworks PNG files into Flash as flattened images or as editable objects. When you import a PNG file as a flattened image, the entire file (including any vector artwork) is *rasterized,* or converted to a bitmap image. When you import a PNG file as editable objects, vector artwork in the file is preserved in vector format. You can choose to preserve placed bitmaps, text, and guides in the PNG file when you import it as editable objects.

If you import the PNG file as a flattened image, you can launch Fireworks from within Flash and edit the original PNG file (with vector data). See "Editing bitmaps in an external editor" on page 98.

When you import multiple PNG files in a batch, you choose import settings one time. Flash MX uses the same settings for all files in the batch.

Note: You can edit bitmap images in Flash by converting the bitmap images to vector artwork or by breaking apart the bitmap images. See "Converting bitmaps to vector graphics" on page 99 and "Breaking apart a bitmap" on page 98.

To import a Fireworks PNG file:

1. Choose File > Import.
2. In the Import dialog box, choose PNG Image from the Files of Type (Windows) or Show (Macintosh) pop-up menu.
3. Navigate to a Fireworks PNG image and select it.
4. Do one of the following:

- In Windows or Macintosh OS 10 or later, click Open.
- In Macintosh OS 9.X or earlier, click Add to add the selected file to the Import list, and click Import to import the file or files in the Import list.

5 In the Fireworks PNG Import Settings dialog box, select one of the following for File Structure:

- Select Import as Movie Clip and Retain Layers to import the PNG file as a movie clip, with all of its frames and layers intact inside the movie clip symbol.
- Select Import into New Layer in Current Scene to import the PNG file into the current Flash document in a single new layer at the top of the stacking order. The Fireworks layers are flattened into the single layer. The Fireworks frames are contained in the new layer.

6 For Objects, select one of the following:

- Select Rasterize if Necessary to Maintain Appearance to preserve Fireworks fills, strokes, and effects in Flash.
- Select Keep All Paths Editable to keep all objects as editable vector paths. Some Fireworks fills, strokes, and effects will be lost on import.

7 For Text, select one of the following:

- Select Rasterize if Necessary to Maintain Appearance to preserve Fireworks fills, strokes, and effects in text imported into Flash.
- Select Keep All Paths Editable to keep all text editable. Some Fireworks fills, strokes, and effects will be lost on import.

8 Select Import as a Single Flattened Image to flatten the PNG file into a single bitmap image. When this option is selected, all other options are dimmed.

9 Click OK.

Importing FreeHand files

You can import FreeHand files (version 10 or earlier) directly into Flash. FreeHand is the best choice for creating vector graphics for import into Flash, because you can preserve FreeHand layers, text blocks, library symbols, and pages, and choose a page range to import. If the imported FreeHand file is in CMYK color mode, Flash converts the file to RGB.

Keep the following guidelines in mind when importing FreeHand files:

- When importing a file with overlapping objects that you want to preserve as separate objects, place the objects on separate layers in FreeHand, and choose Layers in the FreeHand Import dialog box in Flash when importing the file. (If overlapping objects on a single layer are imported into Flash, the overlapping shapes will be divided at intersection points, just as with overlapping objects that you create in Flash.)
- When you import files with gradient fills, Flash can support up to eight colors in a gradient fill. If a FreeHand file contains a gradient fill with more than eight colors, Flash creates clipping paths to simulate the appearance of a gradient fill. Clipping paths can increase file size. To minimize file size, use gradient fills with eight colors or fewer in FreeHand.
- When you import files with blends, Flash imports each step in a blend as a separate path. Thus, the more steps a blend has in a FreeHand file, the larger the imported file size will be in Flash.
- When you import files with strokes that have square caps, Flash converts the caps to round caps.
- When you import files with placed grayscale images, Flash converts the grayscale images to RGB images. This conversion can increase the imported file's size.

- When importing files with placed EPS images, you must select the Convert Editable EPS when Imported option in FreeHand Import Preferences before you place the EPS into FreeHand. If you do not select this option, the EPS image will not be viewable when imported into Flash. In addition, Flash does not display information for an imported EPS image (regardless of the Preferences settings used in FreeHand).

To import a FreeHand file:

1. Choose File > Import.
2. In the Import dialog box, choose FreeHand from the Files of Type (Windows) or Show (Macintosh) pop-up menu.
3. Navigate to a FreeHand file and select it.
4. Do one of the following:

- In Windows or Macintosh OS 10 or later, click Open.
- In Macintosh OS 9.X or earlier, click Add to add the selected file to the Import list, and click Import to import the file or files in the Import list.

5. In the FreeHand Import Settings dialog box, for Mapping Pages, choose a setting:

- Scenes converts each page in the FreeHand document to a scene in the Flash document.
- Keyframes converts each page in the FreeHand document to a keyframe in the Flash document.

6. For Mapping Layers, select one of the following:

- Layers converts each layer in the FreeHand document to a layer in the Flash document.
- Keyframes converts each layer in the FreeHand document to a keyframe in the Flash document.
- Flatten converts all layers in the FreeHand document to a single flattened layer in the Flash document.

7. For Pages, do one of the following:

- Choose All to import all pages from the FreeHand document.
- Enter page numbers for From and To to import a page range from the FreeHand document.

8. For Options, choose any of the following options:

- Include Invisible Layers imports all layers (visible and hidden) from the FreeHand document.
- Include Background Layer imports the background layer with the FreeHand document.
- Maintain Text Blocks preserves text in the FreeHand document as editable text in the Flash document.

9. Click OK.

Importing Adobe Illustrator files

Flash can import and export Adobe Illustrator files in version 8.0 format or earlier. (For information on exporting Illustrator files, see "Adobe Illustrator" on page 397.) When you import an Illustrator file into Flash, you must ungroup all the Illustrator objects on all layers. Once all the objects are ungrouped, they can be manipulated like any other Flash object.

To import an Adobe Illustrator file:

1. Choose File > Import.
2. In the Import dialog box, choose Adobe Illustrator from the Files of Type (Windows) or Show (Macintosh) pop-up menu.
3. Navigate to an Illustrator file and select it.
4. Do one of the following:

- In Windows or Macintosh OS 10 or later, click Open.
- In Macintosh OS 9.X or earlier, click Add to add the selected file to the Import list, then click Import to import the file in the Import list.

   The Illustrator Import Settings dialog box appears.

5. For Convert Layers, select one of the following:

- Layers converts each layer in the Illustrator document to a layer in the Flash document.
- Keyframes converts each layer in the Illustrator document to a keyframe in the Flash document.
- Flatten converts all layers in the Illustrator document to a single flattened layer in the Flash document.

6. Choose Include Invisible Layers to import all layers (visible and hidden) from the Illustrator document.
7. Click OK.

AutoCAD DXF files

Flash supports the AutoCAD DXF format in the release 10 version.

DXF files do not support the standard system fonts. Flash tries to map fonts appropriately, but the results can be unpredictable, particularly for the alignment of text.

Since the DXF format does not support solid fills, filled areas are exported as outlines only. For this reason, the DXF format is most appropriate for line drawings, such as floor plans and maps.

You can import two-dimensional DXF files into Flash. Flash does not support three-dimensional DXF files.

Although Flash doesn't support scaling in a DXF file, all imported DXF files produce 12-inch x 12-inch movies that you can scale using the Modify > Transform > Scale command. Also, Flash supports only ASCII DXF files. If your DXF files are binary, you must convert them to ASCII before importing them into Flash.

Working with imported bitmaps

When you import a bitmap into Flash, you can modify that bitmap and use it in your Flash movie in a variety of ways. You can apply compression and anti-aliasing to imported bitmaps to control the size and appearance of bitmaps in movies. See "Setting bitmap properties" on page 97. You can apply an imported bitmap as a fill to an object. See "Applying a bitmap fill" on page 97.

Flash lets you break apart a bitmap into editable pixels. The bitmap retains its original detail but is broken into discrete areas of color. When you break a bitmap apart, you can select and modify areas of the bitmap with the Flash drawing and painting tools. Breaking apart a bitmap also lets you sample the bitmap with the Eyedropper tool to use it as a fill. See "Breaking apart a bitmap" on page 98.

You can edit an imported bitmap in Fireworks or another external image editor by launching the editing application from within Flash. See"Editing bitmaps in an external editor" on page 98. To convert a bitmap's image to a vector graphic, you can trace the bitmap. Performing this conversion enables you to modify the graphic as you do other vector artwork in Flash. See "Converting bitmaps to vector graphics" on page 99.

If a Flash movie displays an imported bitmap at a larger size than the original, the image may be distorted. Preview imported bitmaps to be sure that images are displayed properly.

Using the Property inspector to work with bitmaps

When you select a bitmap on the Stage, the Property inspector displays the bitmap's symbol name and its pixel dimensions and position on the Stage. Using the Property inspector, you can assign a new name to the bitmap, and you can *swap* an instance of a bitmap—that is, replace the instance with an instance of another bitmap in the current document.

To display the Property inspector:

1. Select an instance of a bitmap on the Stage.
2. Choose Window > Properties.

To assign a new name to a bitmap:

1. Select the bitmap in the Library panel.
2. Choose Window > Properties if the Property inspector is not visible. Select an instance of the bitmap on the Stage to view the bitmap properties.
3. In the Property inspector, enter a new name in the Name text box.
4. Click OK.

To replace an instance of a bitmap with an instance of another bitmap:

1. Select a bitmap instance on the Stage.
2. Choose Window > Properties if the Property inspector is not visible.
3. In the Property inspector, click Swap.
4. In the Swap Bitmap dialog box, select a bitmap to replace the one currently assigned to the instance.

Setting bitmap properties

You can apply anti-aliasing to an imported bitmap to smooth the edges in the image. You can also select a compression option to reduce the bitmap file size and format the file for display on the Web.

To select bitmap properties, you use the Bitmap Properties dialog box.

To set bitmap properties:

1. Select a bitmap in the Library panel.
2. Do one of the following:

- Click the properties icon at the bottom of the Library panel.
- Right-click (Windows) or Control-click (Macintosh) the bitmap's icon and choose Properties from the context menu.
- Choose Properties from the options menu in the upper right corner of the Library panel.

3. In the Bitmap Properties dialog box, select Allow Smoothing to smooth the edges of the bitmap with anti-aliasing.
4. For Compression, choose one of the following options:

- Choose Photo (JPEG) to compress the image in JPEG format. To use the default compression quality specified for the imported image, select Use Document Default Quality. To specify a new quality compression setting, deselect Use Document Default Quality and enter a value between 1 and 100 in the Quality text box. (A higher setting preserves greater image integrity but yields a larger file size.)
- Choose Lossless (PNG/GIF) to compress the image with lossless compression, in which no data is discarded from the image.

Note: Use Photo compression for images with complex color or tonal variations, such as photographs or images with gradient fills. Use Lossless compression for images with simple shapes and relatively few colors.

5. Click Test to determine the results of the file compression. Compare the original file size to the compressed file size to determine if the selected compression setting is acceptable.
6. Click OK.

Note: JPEG Quality settings that you select in the Publish Settings dialog box do not specify a quality setting for imported JPEG files. You must specify a quality setting for imported JPEG files in the Bitmap Properties dialog box.

Applying a bitmap fill

You can apply a bitmap as a fill to a graphic object using the Color Mixer. Applying a bitmap as a fill tiles the bitmap to fill the object. The Fill Transform tool allows you to scale, rotate, or skew an image and its bitmap fill. See "Transforming gradient and bitmap fills" on page 84.

To apply a bitmap as a fill using the Color Mixer:

1. To apply the fill to existing artwork, select a graphic object or objects on the Stage.
2. Choose Window > Color Mixer.
3. In the Color Mixer, choose Bitmap from the pop-up menu in the center of the panel.

4 If you need a larger preview window to display more bitmaps in the current document, click the arrow in the lower right corner to expand the Color Mixer.

5 Click a bitmap to select it.

The bitmap becomes the current fill color. If you selected artwork in step 1, the bitmap is applied as a fill to the artwork.

Editing bitmaps in an external editor

If you have Fireworks 3 or later or another image-editing application installed on your system, you can launch the application from within Flash to edit an imported bitmap.

If you are editing a Fireworks PNG file imported as a flattened image, you can choose to edit the PNG source file for the bitmap, when available.

Note: You cannot edit bitmaps from Fireworks PNG files imported as editable objects in an external image editor.

To edit a bitmap with Fireworks 3 or later:

1 In the Library panel, right-click (Windows) or Control-click (Macintosh) the bitmap's icon.

2 In the bitmap's context menu, select Edit with Fireworks 3.

3 In the Edit Image dialog box, specify whether the PNG source file or the bitmap file is to be opened.

4 Perform the desired modifications to the file in Fireworks.

5 Choose File > Update.

The file is automatically updated in Flash.

To edit a bitmap with another external editing application:

1 In the Library panel, right-click (Windows) or Control-click (Macintosh) the bitmap's icon.

2 In the bitmap's context menu, select Edit With.

3 Choose an image-editing application to open the bitmap file, and click OK.

4 Perform the desired modifications to the file in the image-editing application.

5 Save the file in the image-editing application.

The file is automatically updated in Flash.

6 Return to Flash to continue editing the document.

Breaking apart a bitmap

Breaking apart a bitmap separates the pixels in the image into discrete areas that can be selected and modified separately. When you break apart a bitmap, you can modify the bitmap with the Flash drawing and painting tools. Using the Lasso tool with the Magic Wand modifier, you can select areas of a bitmap that has been broken apart.

You can paint with a broken-apart bitmap by selecting the bitmap with the Eyedropper tool and applying the bitmap as a fill with the Paint Bucket tool or another drawing tool.

To break apart a bitmap:

1. Select a bitmap in the current scene.
2. Choose Modify > Break Apart.

To change the fill of selected areas of a broken-apart bitmap:

1. Select the Lasso tool and click the Magic Wand modifier.

2. Click the Magic Wand Settings modifier and set the following options:

- For Threshold, enter a value between 1 and 200 to define how closely the color of adjacent pixels must match to be included in the selection. A higher number includes a broader range of colors. If you enter 0, only pixels of the exact same color as the first pixel you click are selected.
- For Smoothing, select an option from the pop-up menu to define how much the edges of the selection will be smoothed.

3. Click the bitmap to select an area. Continue clicking to add to the selection.
4. Select the fill that you want to use to fill the selected areas in the bitmap. See "Using the Stroke Color and Fill Color controls in the toolbox" on page 77.
5. Select the Paint Bucket tool and click anywhere in the selected area to apply the new fill.

To apply a broken-apart bitmap as a fill using the Eyedropper tool:

1. Select the Eyedropper tool and click the broken-apart bitmap on the Stage.

   The Eyedropper tool sets the bitmap to be the current fill and changes the active tool to the Paint Bucket.

2. Do one of the following:

- Click an existing graphic object with the Paint Bucket tool to apply the bitmap as a fill.
- Select the Oval, Rectangle, or Pen tool and draw a new object. The object is filled with the broken-apart bitmap.

  You can use the Paint Bucket tool to scale, rotate, or skew the bitmap fill.

Converting bitmaps to vector graphics

The Trace Bitmap command converts a bitmap into a vector graphic with editable, discrete areas of color. This command lets you manipulate the image as a vector graphic; it is also useful if you wish to reduce file size.

When you convert a bitmap to a vector graphic, the vector graphic is no longer linked to the bitmap symbol in the Library panel.

Note: If the imported bitmap contains complex shapes and many colors, the converted vector graphic may have a larger file size than the original bitmap. Try a variety of settings in the Trace Bitmap dialog box to find a balance between file size and image quality.

You can also break apart a bitmap in order to modify the image using Flash drawing and painting tools. See "Breaking apart a bitmap" on page 98.

To convert a bitmap to a vector graphic:

1. Select a bitmap in the current scene.
2. Choose Modify > Trace Bitmap.
3. Enter a Color Threshold value between 1 and 500.

   When two pixels are compared, if the difference in the RGB color values is less than the color threshold, the two pixels are considered the same color. As you increase the threshold value, you decrease the number of colors.
4. For Minimum Area, enter a value between 1 and 1000 to set the number of surrounding pixels to consider when assigning a color to a pixel.
5. For Curve Fit, select an option from the pop-up menu to determine how smoothly outlines are drawn.
6. For Corner Threshold, select an option from the pop-up menu to determine whether sharp edges are retained or smoothed out.

   To create a vector graphic that looks most like the original bitmap, enter the following values:

- Color Threshold: 10
- Minimum Area: 1 pixel
- Curve Fit: Pixels
- Corner Threshold: Many Corners

The results of using the Trace Bitmap command, with low settings (more like the original image) and higher settings (more distorted)

Importing video

You can import video clips into Flash. Depending on the video format and the import method you choose, you can publish the movie with video as a Flash movie (SWF file) or a QuickTime movie (MOV file).

If you have QuickTime 4 or later (Windows or Macintosh) or DirectX 7 or later (Windows only) installed on your system, you can import video clips in a variety of file formats, including MOV (QuickTime movie), AVI (Audio Video Interleaved file), and MPG/MPEG (Motion Picture Experts Group file). For information on supported file formats, see “Video import file formats” on page 101.

You can import video clips as embedded files or linked files. See “Importing video clips as embedded files” on page 103 and “Importing QuickTime video clips as linked files” on page 105.

You can apply the following actions to imported video objects in movie clips: `goTo`, `play`, `stop`, `toggleHighQuality`, `stopAllSounds`, `getURL`, `FScommand`, `loadMovie`, `unloadMovie`, `ifFrameLoaded`, and `onMouseEvent`. To apply actions to a video object, you must first convert the video object to a movie clip. For information on using ActionScript, see Chapter 12, "Understanding the ActionScript Language," on page 203.

Video import file formats

The following video file formats are supported for import if QuickTime 4 is installed (Windows and Macintosh):

File type	Extension	Windows	Macintosh
Audio Video Interleaved	.avi,	✔	✔
Digital Video	.dv	✔	✔
Motion Picture Experts Group	.mpg, .mpeg	✔	✔
QuickTime Movie	.mov	✔	✔

The following video file formats are supported for import if DirectX 7 or higher is installed (Windows only):

File type	Extension	Windows
Audio Video Interleaved	.avi,	✔
Motion Picture Experts Group	.mpg, .mpeg	✔
Windows Media File	.wmv, .asf	✔

By default, Flash imports and exports video using the Sorenson Spark *codec*. A codec is a compression/decompression algorithm that controls how multimedia files are compressed and decompressed during import and export. Additional video import formats may be supported, depending on what codecs are installed on your system. For information on the Sorenson Spark codec, see "About the Sorenson Spark codec" on page 101.

If you attempt to import a file format that is not supported on your system, Flash displays a warning message indicating that the operation cannot be completed. In some cases, Flash may be able to import the video but not the audio in a file. For example, audio is not supported in MPG/MPEG files imported with QuickTime 4. In such cases, Flash displays a warning indicating that the audio portion of the file cannot be imported. You can still import the video without sound.

Note: Imported audio is published or exported as streamed audio, using the global audio streaming settings selected in the Publish Settings dialog box. See "Choosing publish settings Flash movie format" on page 370.

About the Sorenson Spark codec

Sorenson Spark is a motion video codec included in Flash MX that enables you to add video content to Flash. Spark is a high-quality video encoder and decoder that dramatically lowers the bandwidth required to deliver video into Flash while simultaneously increasing the video quality. With the inclusion of Spark, Flash takes an important leap forward in video capability. In previous versions of Flash, you could only simulate video using sequential bitmap images. Two versions of Sorenson Spark are available: Sorenson Spark Standard Edition is included in Flash MX and Flash Player 6. The Spark Standard edition codec produces good-quality video for low-motion content, such as a person speaking.

The Spark video codec is comprised of an encoder and a decoder. The encoder (or compressor) is the component in Spark that compresses your content. The decoder (or decompressor) is the component that decompresses the compressed content so that it can be viewed. The decoder is included in the Flash Player.

About compression

There are two different types of compression that can be applied to digital media: *spatial* and *temporal*.

Temporal compression identifies the differences between frames and stores only those differences, so that frames are described based on their difference from the preceding frame. Unchanged areas are simply repeated from the previous frame(s). A temporally compressed frame is often referred to as an *interframe*.

Spatial compression, on the other hand, is applied to a single frame of data, independent of any surrounding frames. Spatial compression can be lossless (in which no data is discarded from the image) or lossy (in which data is selectively discarded). A spatially compressed frame is often referred to as an *intraframe*.

Sorenson Spark is an interframe codec. Sorenson Spark's efficient interframe compression is part of what separates it from other compression technologies, requiring a much lower data rate than most other codecs to produce good-quality video. Many other codecs use intraframe compression; for example, JPEG is an intraframe codec.

However, interframe codecs also use intraframes. The intraframes are used as the reference frames (keyframes) for the interframes. Sorenson Spark always begins with a keyframe. Each keyframe becomes the main reference frame for the following interframes. Whenever the next frame is significantly different from the previous frame, the codec compresses a new keyframe.

Tips for creating Flash video with Sorenson Spark

How you compress your video is largely determined by the content of the video. A video clip of a talking head with very little action and only short bursts of moderate motion compresses very differently than footage of a soccer match. Following are some tips on delivering the best possible Flash video:

Strive for simplicity. Avoid elaborate transitions—they don't compress well and may make your final compressed video look "chunky" during the change. Hard cuts are usually best, or quick cross-fades. Videos that zoom out from behind the first track, do a "page turn," or wrap around a ball and then fly off the screen may look cool, but they usually don't compress well and should be used sparingly.

Know your audience data rate. When you deliver video over the Internet, you should produce files at lower intranet data rates. Users with fast Internet connections can view the files with little or no wait, while dialup users will have to wait for the files to download. In these situations, it is best to make the clips short to keep the download times within acceptable limits for dialup users.

Select the proper frame rate. Frame rate indicates how many frames are played each second. If you have a higher data rate clip, a lower frame rate can improve playback on lower-end computers. For example, if you are compressing a talking head clip with little motion, cutting the frame rate in half will probably only save you 20% of the data rate. However, if you are compressing high-motion video, reducing the frame rate has a much greater effect on the data rate.

Since video looks much better at native frame rates, we recommend leaving it high if your delivery channels and playback platforms allow. However, if you need to reduce the frame rate, the best results come from dividing the frame rate by whole numbers.

Select a frame size that fits your data rate. Like frame rate, the frame size for your movie is important for producing high-quality video. At a given data rate (connection speed), increasing the frame size results in decreased video quality. When you select the frame size for your video you must also consider frame rate, source material, and personal preferences. The following list should be used as a guideline. Experiment to find the best setting for your project.

Common frame sizes:

Modem: 160 x 120

Dual ISDN: 192 x 144

T1/DSL/Cable: 320 x 240

Know progressive download. You should know how long it is going to take to download your video. While your video clip is downloading, you might want to have other content that appears and "disguises" the download. For short clips you can use the following formula: Pause = Download time – Play time + 10% of play time. For example, If your clip is 30 seconds long and it takes one minute to download, you should give your clip a 33-second buffer: 60 seconds – 30 seconds + 3 seconds = 33 seconds.

Use clean video. The higher the quality of the original, the better the final movie. While frame rates and sizes of Internet video are usually less than what you see on a television, computer monitors have much better color fidelity, saturation, sharpness, and resolution than a conventional television. Even with a small window, image quality can be more important for digital video than for standard analog television. Artifacts and noise that would hardly be noticeable on TV can be painfully obvious on a computer.

Remove noise and interlace. After you capture your video content, you might need to remove noise and interlace.

Follow the same guidelines for audio. The same considerations exist for audio production as for video production. In order to achieve good audio compression, you must begin with clean audio. If you are encoding material from a CD, try to record the file using direct digital transfer instead of through the analog input of your sound card. The sound card introduces an unnecessary digital-to-analog and analog-to-digital conversion that can create noise in your source audio. Direct digital transfer tools are available for both Mac and PC platforms. If you must record from an analog source, be sure to use the highest quality sound card available.

Importing video clips as embedded files

You can embed a video clip when importing it into Flash. The video clip becomes part of the movie, like an imported bitmap or vector artwork file. You can publish a movie with embedded video as a Flash movie. You can also publish a movie with embedded video as a QuickTime movie, with a Flash track that contains embedded video. You can import any supported video file format as an embedded video.

You can synchronize the frame rate of an embedded video to match the frame rate of the main movie Timeline. You can also adjust the ratio of the video frame rate to the main Timeline frame rate, to drop frames from the imported video during playback.

There are situations in which you may not want to synchronize the embedded video with the Flash movie. Some examples are the following:

- You want to prevent frames in the embedded video from being dropped or duplicated. Deselecting the Synchronize option accomplishes this. For example, suppose you want to import a video that has a slightly different frame rate than the Flash movie (such as an NTSC video clip with a frame rate of 29.94 fps, imported into a Flash movie with a frame rate of 30 fps). Deselecting the Synchronize option prevents frames from being dropped in the embedded video and prevents the hiccup effect that this causes during playback.
- You want to drop frames from a video that has a lower frame rate than the Flash movie. If you synchronize this video, the option for dropping frames is disabled. You must deselect the Synchronize option in order to drop frames.

You can update an imported video that you have edited in an external application, or import another video to replace an embedded video. You can also assign a different symbol to an instance of a video clip. Assigning a different symbol to an instance displays a different instance on the Stage but leaves all the original instance properties (such as color, rotation, and so on) intact.

You can create a video object on the Stage by dragging an instance of an imported video clip from the Library panel onto the Stage. As with symbols, you can create multiple instances of an imported video clip without adding to the Flash movie file size.

To import a video as an embedded clip:

1 Do one of the following:

- To import the video clip directly to the Stage in the current Flash document, choose File > Import.
- To import the video clip into the library for the current Flash document, choose File > Import to Library.

2 In the Import Video dialog box, select Embed Video in Macromedia Flash Movie.

3 In the Import Video Settings dialog box, drag the slider or enter a value for Quality to control the amount of compression applied to the video clip. A lower Quality setting produces a smaller file size but may reduce image integrity.

4 Drag the slider or enter a value for Keyframe Interval to control the frequency of keyframes (frames with complete data) in the video clip. For example, a keyframe interval of 30 stores a complete frame at every 30 frames in the clip. Frames between intervals store only the data that changes from the preceding frame. A smaller interval stores more complete frames. This enables faster seeking in the video but yields a larger file size.

Note: A keyframe interval of 1 stores a complete frame for each frame of the video. This setting is recommended only for very small video files.

5 Drag the slider or enter a value for Scale to reduce the pixel dimensions of the video. A smaller pixel size reduces file size and can improve playback performance.

For example, a Digital Video (DV) file can be 640 x 480 pixels. Reducing the scale of this file to 25% would improve the performance of the video in the Flash movie.

6 Select Synchronize Video to Macromedia Flash Movie Frame Rate to match the playback speed of the imported video to the playback speed of the main Flash movie Timeline.

Deselect this option to prevent frame rate synchronization.

7 Select a value for Number of Video Frames to Encode Per Number of Flash Frames to specify the ratio of imported video frames to main Flash Timeline frames. For example, to play one imported video frame per one main Flash Timeline frame, choose 1:1; to play one imported video frame per every two main Timeline frames, choose 1:2; and so on.

Dropping frames from the imported video does not slow down the motion of the video. Instead, it displays fewer frames per second, so that the video appears more choppy in playback.

8 Select Import Audio to include the audio track (if present) in the imported video clip.

Deselect this option to omit the audio track from the imported video clip.

Note: If the audio codec used in the audio track is not supported on your system, Flash displays a warning when you click OK in the Import Video Settings dialog box. You can continue the procedure and import the video without sound, or return to the video authoring application and resave the video with an audio codec that is supported on your system.

9 Click OK.

10 If you import the video clip directly to the Stage in step 1, a warning appears if the imported clip contains more frames than the span in which you are placing it in the current Flash document. Do one of the following:

- Click Yes to extend the span the required number of frames.
- Click No to keep the span at its current size. Frames in the imported clip that exceed the frames in the span will not be displayed unless you subsequently add frames to the span.

To update an embedded video clip after editing it in an external editor:

1 Select the video clip in the Library panel.

2 In the options menu in the upper right corner of the Library panel, choose Properties.

3 In the Embedded Video Properties dialog box, click Update.

The embedded video clip is updated with the edited file.

To replace an embedded video clip with another video clip:

1 In the Library panel, select the embedded video clip that you want to replace.

2 In the options menu in the upper right corner of the Library panel, choose Properties.

3 In the Embedded Video Properties dialog box, click Import.

4 In the Import dialog box, select a video clip to replace the embedded clip in the Library panel.

Importing QuickTime video clips as linked files

If you are importing a QuickTime video clip, you can choose to link to the video from the Flash movie, rather than embed the video. A linked QuickTime movie imported into Flash does not become part of the Flash file. Instead, Flash maintains a pointer to the source file.

If you link to a QuickTime video, you must publish the movie as a QuickTime movie. You cannot display a linked QuickTime movie in SWF format. The QuickTime movie contains a Flash track, but the linked video clip remains in QuickTime format. For more information on publishing your Flash file as a QuickTime movie, see “Choosing publish settings for QuickTime 4 movies” on page 381.

You can scale, rotate, and animate a linked QuickTime movie in Flash. However, you cannot tween linked QuickTime movie content in Flash.

To import a QuickTime video as a linked file:

1 Do one of the following:

- To link the video clip directly to the current Flash document, choose File > Import.
- To link the video clip to the library for the current Flash document, choose File > Import to Library.

2 In the Import Video dialog box, select Link to External Video File.

Previewing a linked QuickTime movie

When you import a linked QuickTime movie, only the first frame of the movie is displayed. You must add frames to the imported movie's Timeline in order to view additional frames of the movie in Flash.

To preview a linked QuickTime movie:

1 Add the number of frames to the Timeline that correspond to the length of the QuickTime movie you want to play.

2 Choose Control > Play.

Note: You cannot preview linked QuickTime movie content using the Test Movie command.

Setting the directory path of a linked QuickTime movie

You can set the directory path of a linked QuickTime video clip in the library for the current Flash document.

To set the directory path of a linked QuickTime video clip:

1 Choose Window > Library and select the linked QuickTime movie you want to edit.

2 In the options menu in the upper right corner of the Library panel, choose Properties.

3 Click Set Path in the Linked Video Properties dialog box.

4 In the Open dialog box, navigate to the file for the linked video clip and select it, then click Open.

5 In the Linked Video Properties dialog box, click OK.

Working with imported video files

When you select an instance of an embedded or linked video clip on the Stage and open the Property inspector, the Property inspector displays the clip's symbol name and its pixel dimensions and position on the Stage. Using the Property inspector, you can assign a new name to the video clip, and you can *swap* an instance of a video clip—that is, replace the instance with an instance of another video clip in the current document.

The Embedded Video Properties dialog box allows you to view information about an imported video clip, including its name, path, creation date, pixel dimensions, length, and file size.

Note: You can preview frames of an imported video by dragging the playhead along the Timeline. However, the sound will not play back. To preview the video with sound, use the Test Movie command. See "Previewing and testing movies" on page 39.

To display the Property inspector:

1. Select an instance of an embedded or linked video clip on the Stage.
2. Choose Window > Properties.

To assign a new name to a video clip:

1. Select the video clip in the Library panel.
2. Select an instance of the video clip on the Stage.
3. Choose Window > Properties if the Property inspector is not visible.
4. In the Property inspector, enter a new name in the Name text box.
5. Click OK.

To replace an instance of a video clip with an instance of another video clip:

1. Select an embedded or linked video clip instance on the Stage.
2. Choose Window > Properties if the Property inspector is not visible.
3. In the Property inspector, click Swap.
4. In the Swap Embedded Video dialog box, select a video clip to replace the one currently assigned to the instance.

   Note: You can swap an embedded video clip only with another embedded video clip, and you can swap a linked video clip only with another linked video clip.

To display the Embedded Video Properties dialog box:

1. Select an imported video clip in the Library panel.
2. Choose Properties from the options menu in the upper right of the Library panel.

About creating a video object for a live stream

You can create a video object to display a live video stream from a camera. To create a video object, you choose New Video Object from the Library panel options menu. Next, you assign a name for the object in the Property inspector. For information on assigning a name to a video object, see "Working with imported video files" on page 106.

CHAPTER 6
Adding Sound

Macromedia Flash MX offers a number of ways to use sounds. You can make sounds that play continuously, independent of the Timeline, or you can synchronize animation to a sound track. You can add sounds to buttons to make them more interactive, and make sounds fade in and out for a more polished sound track.

There are two types of sounds in Flash: event sounds and stream sounds. An event sound must download completely before it begins playing, and it continues playing until explicitly stopped. Stream sounds begin playing as soon as enough data for the first few frames has been downloaded; stream sounds are synchronized to the Timeline for playing on a Web site.

You select compression options to control the quality and size of sounds in exported movies. You can select compression options for individual sounds using the Sound Properties dialog box, or define settings for all sounds in the movie in the Publish Settings dialog box.

You can use sounds in shared libraries, to link a sound from one library to multiple movies. See "Using shared library assets" on page 165. You can also use the ActionScript `onSoundComplete` event to trigger an event based on the completion of a sound. See "About the onSoundComplete event" on page 114.

Note: You can also use actions to load sounds dynamically. See the entries for `Sound.attachSound` and `Sound.loadSound` in the online ActionScript Dictionary in the Help menu.

Importing sounds

You place sound files into Flash by importing them into the library for the current document.

Note: When placing a sound on the Timeline, it is recommended that you place it on a separate layer. See "Adding sounds to a movie" on page 110 for more information.

You can import the following sound file formats into Flash:

- WAV (Windows only)
- AIFF (Macintosh only)
- MP3 (Windows or Macintosh)

If you have QuickTime 4 or later installed on your system, you can import these additional sound file formats:

- AIFF (Windows or Macintosh)
- Sound Designer II (Macintosh only)
- Sound Only QuickTime Movies (Windows or Macintosh)
- Sun AU (Windows or Macintosh)

- System 7 Sounds (Macintosh only)
- WAV (Windows or Macintosh)

Flash stores sounds in the library along with bitmaps and symbols. As with graphic symbols, you need only one copy of a sound file to use that sound in any number of ways in your movie.

If you want to share sounds among Flash movies, you can include the sounds in shared libraries. See "Working with common libraries" on page 58. To use a sound in a shared library, you assign the sound file an identifier string in the Symbol Linkage Properties dialog box. The identifier can also be used to access the sound as an object in ActionScript. For information on objects in ActionScript, see Chapter 12, "Understanding the ActionScript Language," on page 203.

Sounds can use considerable amounts of disk space and RAM. However, MP3 sound data is compressed and smaller than WAV or AIFF sound data. Generally, when using WAV or AIFF files, it's best to use 16-bit 22 kHz mono sounds (stereo uses twice as much data as mono), but Flash can import either 8- or 16-bit sounds at sample rates of 11 kHz, 22 kHz, or 44 kHz. Flash can convert sounds to lower sample rates on export. See "Compressing sounds for export" on page 115.

Note: Sounds recorded in formats that are not multiples of 11 kHz (such as 8, 32, or 96 kHz) are resampled when imported into Flash.

If you want to add effects to sounds in Flash, it's best to import 16-bit sounds. If you have limited RAM, keep your sound clips short or work with 8-bit sounds instead of 16-bit sounds.

To import a sound:

1. Choose File > Import to Library.
2. In the Import dialog box, locate and open the desired sound file.

   Note: You can also drag a sound from a common library into the library for the current document. See "Working with common libraries" on page 58.

Adding sounds to a movie

To add a sound to a movie from the library, you assign the sound to a layer and set options in the Sound controls in the Property inspector. It is recommended that you place each sound on a separate layer.

You can load a sound into a movie during runtime, using the `loadSound` method of the Sound object. For specific information on the Sound object and its methods, see its entry in the ActionScript Dictionary in the Help menu.

To test sounds that you add to a movie, you can use the same methods you use to preview frames or test movies: drag the playhead over the frames containing the sound, or use commands in the Controller or the Control menu. See "Previewing and testing movies" on page 39.

To add a sound to a movie:

1. Import the sound into the library if it has not already been imported. See "Importing sounds" on page 109.
2. Choose Insert > Layer to create a layer for the sound.
3. With the new sound layer selected, drag the sound from the Library panel onto the Stage. The sound is added to the current layer.

   You can place multiple sounds on one layer, or on layers containing other objects. However, it is recommended that each sound be placed on a separate layer. Each layer acts like a separate sound channel. The sounds on all layers are combined when you play back the movie.

4 In the Timeline, select the first frame that contains the sound file.

5 Choose Window > Properties and click the arrow in the lower right corner to expand the Property inspector.

6 In the Property inspector, choose the sound file from the Sound pop-up menu.

7 Choose an effect option from the Effects pop-up menu:

- None applies no effects to the sound file. Choose this option to remove previously applied effects.
- Left Channel/Right Channel plays sound in the left or right channel only.
- Fade Left to Right/Fade Right to Left shifts the sound from one channel to the other.
- Fade In gradually increases the amplitude of a sound over its duration.
- Fade Out gradually decreases the amplitude of a sound over its duration.
- Custom lets you create your own In and Out points of sound using the Edit Envelope. See "Using the sound-editing controls" on page 113.

8 Choose a synchronization option from the Sync pop-up menu:

- Event synchronizes the sound to the occurrence of an event. An event sound plays when its starting keyframe is first displayed and plays in its entirety, independently of the Timeline, even if the movie stops. Event sounds are mixed when you play your published movie.

  An example of an event sound is a sound that plays when a user clicks a button. If an event sound is playing and the sound is instantiated again (for example, by the user clicking the button again) the first instance of the sound continues to play and another instance begins to play simultaneously.
- Start is the same as Event, except that if the sound is already playing, no new instance of the sound is played.
- Stop silences the specified sound.
- Stream synchronizes the sound for playing on a Web site. Flash forces animation to keep pace with stream sounds. If Flash can't draw animation frames quickly enough, it skips frames. Unlike event sounds, stream sounds stop if the movie stops. Also, a stream sound can never play longer than the length of the frames it occupies. Stream sounds are mixed when you publish your movie.

  An example of a stream sound is the voice of a character in an animation that plays in multiple frames.

  Note: If you use an MP3 sound as a stream sound, you must recompress the sound for export. You can choose to export the sound as an MP3 file, with the same compression settings that it had on import. See "Compressing sounds for export" on page 115.

9 Enter a value for Loop to specify the number of times the sound should loop.

  For continuous play, enter a number large enough to play the sound for an extended duration. For example, to loop a 15-second sound for 15 minutes, enter 60.

  Note: Looping stream sounds is not recommended. If a stream sound is set to loop, frames are added to the movie and the file size is increased by the number of times the sound is looped.

Adding sounds to buttons

You can associate sounds with the different states of a button symbol. Because the sounds are stored with the symbol, they work for all instances of the symbol.

To add sound to a button:

1. Select the button in the Library panel.
2. Choose Edit from the options menu in the upper right corner.
3. In the button's Timeline, add a layer for sound.
4. In the sound layer, create a regular or blank keyframe to correspond to the button state to which you want to add a sound.

   For example, to add a sound that plays when the button is clicked, create a keyframe in the frame labeled Down.
5. Click the keyframe you have just created.
6. Choose Window > Properties.
7. In the Property inspector, choose a sound file from the Sound pop-up menu.
8. Choose Event from the Synchronization pop-up menu.

   To associate a different sound with each of the button's keyframes, create a blank keyframe and add another sound file for each keyframe. You can also use the same sound file and apply a different sound effect for each button keyframe. See "Using the sound-editing controls" on page 113.

Using sounds with Sound objects

You can use the Sound object in ActionScript to add sounds to a movie and to control sound objects in a movie. Controlling sounds includes adjusting the volume or the right and left balance while a sound is playing. See "Creating sound controls" on page 281.

To use a sound in a Sound action, you assign an identifier string to the sound in the Symbol Linkage dialog box.

To assign an identifier string to a sound:

1. Select the sound in the Library panel.
2. Do one of the following:

- Choose Linkage from the options menu in the upper right corner of the panel.
- Right-click (Windows) or Control-click (Macintosh) the sound name in the Library panel, and choose Linkage from the context menu.

3. Under Linkage in the Symbol Linkage Properties dialog box, select Export for ActionScript.
4. Enter an identifier string in the text box, and then click OK.

Using the sound-editing controls

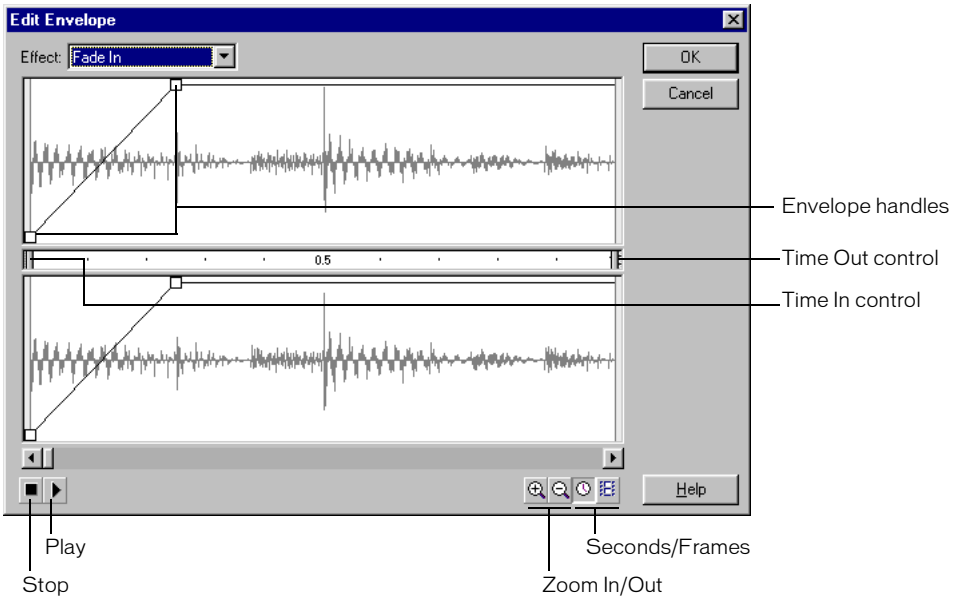
To define the starting point of a sound or to control the volume of the sound as it plays, you use the sound-editing controls in the Property inspector.

Flash can change the point at which a sound starts and stops playing. This is useful for making sound files smaller by removing unused sections.

To edit a sound file:

1. Add a sound to a frame (see "Adding sounds to a movie" on page 110), or select a frame already containing a sound.
2. Choose Window > Properties.
3. Click the Edit button on the right side of the Property inspector.
4. Do any of the following:

- To change the start and end points of a sound, drag the Time In and Time Out controls in the Edit Envelope.



- To change the sound envelope, drag the envelope handles to change levels at different points in the sound. Envelope lines show the volume of the sound as it plays. To create additional envelope handles (up to eight total), click the envelope lines. To remove an envelope handle, drag it out of the window.
- To display more or less of the sound in the window, click the Zoom In or Out buttons.
- To switch the time units between seconds and frames, click the Seconds and Frames buttons.

5. To hear the edited sound, click the Play button.

Starting and stopping sounds at keyframes

The most common sound-related task in Flash is starting and stopping sounds at keyframes in synchronization with animation.

To stop and start a sound at a keyframe:

1. Add a sound to a movie.

   To synchronize this sound with an event in the scene, choose a beginning keyframe that corresponds to the keyframe of the event in the scene. You can choose any of the synchronization options. See "Adding sounds to a movie" on page 110.

2. Create a keyframe in the sound layer's Timeline at the frame where you want the sound to end.

   A representation of the sound file appears in the Timeline.

3. Choose Window > Properties and click the arrow in the lower right corner to expand the Property inspector.

4. In the Property inspector, choose the same sound from the Sound pop-up menu.

5. Choose Stop from the Synchronization pop-up menu.

   When you play the movie, the sound stops playing when it reaches the ending keyframe.

6. To play back the sound, simply move the playhead.

About the onSoundComplete event

The `onSoundComplete` event of the ActionScript Sound object enables you to trigger an event in a movie based on the completion of an attached sound file. The Sound object is a built-in object that lets you control sounds in a movie. For general information on objects, see Chapter 12, "Understanding the ActionScript Language," on page 203. For specific information on the Sound object and its methods, see its entry in the ActionScript Dictionary in the Help menu.

The `onSoundComplete` event of a Sound object is invoked automatically when the attached sound file finishes playing. If the sound is looped a finite number of times, the event is triggered when the sound finishes looping.

The Sound object has two properties that you can use in conjunction with the `onSoundComplete` event. The `duration` property is a read-only property representing the duration in milliseconds of the sound sample attached to the sound object. The `position` property is a read-only property representing the number of milliseconds the sound has been playing in each loop.

The `onSoundComplete` event enables you to manipulate sounds in a variety of powerful ways, such as the following:

- Creating a dynamic playlist or sequencer
- Creating a multimedia presentation that checks for narration completion before advancing to the next frame or scene
- Building a game that synchronizes sounds to particular events or scenes and transitions smoothly between different sounds
- Timing an image change to a sound—for example, changing an image when a sound is half over

Compressing sounds for export

You can select compression options for individual event sounds and export the sounds with those settings. You can also select compression options for individual stream sounds. However, all stream sounds in a movie are exported as a single stream file, using the highest setting of all those applied to individual stream sounds. This includes stream sounds in video objects.

You choose compression options for individual sounds in the Sound Properties dialog box. You can also choose global compression settings for event sounds or stream sounds in the Publish Settings dialog box. These global settings are applied to individual event sounds or all stream sounds if you do not select compression settings for the sounds in the Sound Properties dialog box. See "Publishing Flash documents" on page 367.

You can also override export settings specified in the Sound Properties dialog box by selecting Override Sound Settings in the Publish Settings dialog box. This option is useful if you want to create a larger high-fidelity audio movie for local use and a smaller low-fidelity version for the Web. See "Choosing publish settings Flash movie format" on page 370.

The sampling rate and degree of compression make a significant difference in the quality and size of sounds in exported movies. The more you compress a sound and the lower the sampling rate, the smaller the size and the lower the quality. You should experiment to find the optimal balance between sound quality and file size.

When working with imported MP3 files, you can choose to export the files in MP3 format using the same settings that the file had when imported.

Note: In Windows, you can also export all the sounds from a movie as a WAV file using File › Export Movie. See "Exporting movies and images" on page 395.

To set export properties for an individual sound:

1. Do one of the following:

- Double-click the sound's icon in the Library panel.
- Right-click (Windows) or Control-click (Macintosh) a sound file in the Library panel and choose Properties from the context menu.
- Select a sound in the Library panel and choose Properties from the options menu in the upper right corner of the panel.
- Select a sound in the Library panel and click the properties icon at the bottom of the Library panel.

2. If the sound file has been edited externally, click Update.
3. For Compression, choose Default, ADPCM, MP3, Raw, or Speech. To select options for the compression format you choose, see the section below corresponding to the selected format.

- "The Default compression option" on page 116
- "Using the ADPCM compression option" on page 116
- "Using the MP3 compression option" on page 116
- "Using the Raw compression option" on page 117
- "Using the Speech compression option" on page 117

4. Set export settings.

5 Click Test to play the sound once. Click Stop if you want to stop testing the sound before it has finished playing.

6 Adjust export settings if necessary until the desired sound quality is achieved.

7 Click OK.

The Default compression option

The Default compression option uses the global compression settings in the Publish Settings dialog box when you export your movie. If you select Default, no additional export settings are available.

Using the ADPCM compression option

The ADPCM compression option sets compression for 8-bit or 16-bit sound data. Use the ADPCM setting when you are exporting short event sounds such as button clicks.

To use ADPCM compression:

1 In the Sound Properties dialog box, choose ADPCM from the Compression menu.

2 For Preprocessing, select Convert Stereo to Mono to convert mixed stereo sounds to mono (monaural). (Mono sounds are unaffected by this option.)

3 For Sample Rate, select an option to control sound fidelity and file size. Lower rates decrease file size but can also degrade sound quality. Rate options are as follows:

- 5 kHz is barely acceptable for speech.
- 11 kHz is the lowest recommended quality for a short segment of music and is one-quarter of the standard CD rate.
- 22 kHz is a popular choice for Web playback and is half the standard CD rate.
- 44 kHz is the standard CD audio rate.

Note: Flash cannot increase the kHz rate of an imported sound above the rate at which it was imported.

Using the MP3 compression option

The MP3 compression option lets you export sounds with MP3 compression. Use MP3 when you are exporting longer stream sounds such as music sound tracks.

If you are exporting a file that was imported in MP3 format, you can choose to export the file using the same settings the file had on import.

To export an imported MP3 file with the same settings the file had on import:

1 In the Sound Properties dialog box, choose MP3 from the Compression menu.

2 Select Use Imported MP3 Quality (the default setting). Deselect this option to choose other MP3 compression settings, as defined in the procedure below.

To use MP3 compression:

1 In the Sound Properties dialog box, choose MP3 from the Compression menu.

2 Deselect Use Imported MP3 Quality (the default setting).

3 For Bit Rate, select an option to determine the bits per second in the exported sound file. Flash supports 8 Kbps through 160 Kbps CBR (constant bit rate). When you are exporting music, set the bit rate to 16 Kbps or higher for the best results.

4 For Preprocessing, select Convert Stereo to Mono to convert mixed stereo sounds to mono (monaural). (Mono sounds are unaffected by this option.)

Note: The Preprocessing option is available only if you select a bit rate of 20 Kbps or higher.

5 For Quality, select an option to determine the compression speed and sound quality:

- Fast yields faster compression but lower sound quality.
- Medium yields somewhat slower compression but higher sound quality.
- Best yields the slowest compression and the highest sound quality.

Using the Raw compression option

The Raw compression option exports sounds with no sound compression.

To use raw compression:

1 In the Sound Properties dialog box, choose Raw from the Compression menu.

2 For Preprocessing, select Convert Stereo to Mono to convert mixed stereo sounds to mono (monaural). (Mono sounds are unaffected by this option.)

3 For Sample Rate, select an option to control sound fidelity and file size. Lower rates decrease file size but can also degrade sound quality. Rate options are as follows:

- 5 kHz is barely acceptable for speech.
- 11 kHz is the lowest recommended quality for a short segment of music and is one-quarter of the standard CD rate.
- 22 kHz is a popular choice for Web playback and is half the standard CD rate.
- 44 kHz is the standard CD audio rate.

Note: Flash cannot increase the kHz rate of an imported sound above the rate at which it was imported.

Using the Speech compression option

The speech compression option exports sounds using a compression specially adapted to speech.

To use speech compression:

1 In the Sound Properties dialog box, choose Speech from the Compression menu.

2 For Sample Rate, select an option to control sound fidelity and file size. A lower rate decreases file size but can also degrade sound quality. Choose from the following options:

- 5 kHz is acceptable for speech.
- 11 kHz is recommended for speech.
- 22 kHz is acceptable for most types of music on the Web.
- 44 kHz is the standard CD audio rate. However, since compression is applied, the sound will not be of CD quality in the Flash movie.

Guidelines for exporting sound in Flash movies

Besides sampling rate and compression, there are several ways to use sound efficiently in a movie and keep file size down:

- Set the in and out points to prevent silent areas from being stored in the Flash file and to reduce the size of the sound.
- Get more out of the same sounds by applying different effects for sounds (such as volume envelopes, looping, and in/out points) at different keyframes. You can get a number of sound effects using only one sound file.
- Loop short sounds for background music.
- Do not set streaming sound to loop.
- When exporting audio in embedded video clips, keep in mind that the audio is exported using the global streaming settings selected in the Publish Settings dialog box.
- Use stream synchronization to keep the animation synchronized to your sound track when you preview your animation in the editor. If your computer is not fast enough to draw the animation frames so that they keep up with your sound track, Flash skips frames.
- When exporting QuickTime movies, use as many sounds and channels as you want without worrying about file size. The sounds are combined into a single sound track when you export as a QuickTime file. The number of sounds you use has no effect on the final file size.

CHAPTER 7
Working with Graphic Objects

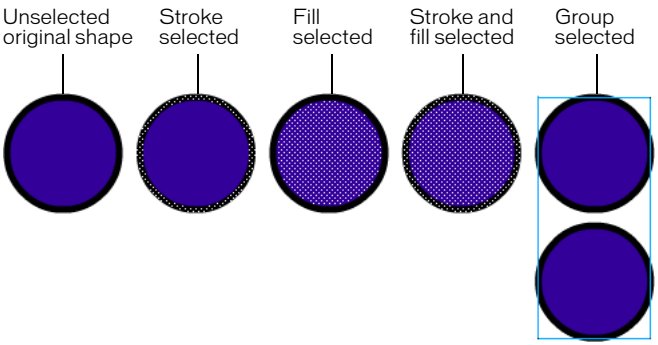
In Macromedia Flash MX, graphic objects are items on the Stage. Flash lets you move, copy, delete, transform, stack, align, and group graphic objects. You can also link a graphic object to a URL. Keep in mind that modifying lines and shapes can alter other lines and shapes on the same layer. See Chapter 3, "Drawing," on page 59.

Note: Graphic objects in Flash are different from ActionScript objects, which are part of the ActionScript programming language. Be careful not to confuse the two uses of the term. For more on objects in the programming language, see "About object-oriented scripting" on page 205.

Selecting objects

To modify an object, you must first select it. Macromedia Flash MX provides a variety of methods for making selections, including the Arrow tool, the Lasso tool, and keyboard commands. You can group individual objects to manipulate them as a single object. See "Grouping objects" on page 122.

Flash highlights objects and strokes that have been selected with a dot pattern. Selected groups are highlighted with bounding boxes in the color used for the outline of the layer that contains the selected group. You can change the layer outline color in the Layer Properties dialog box. See "Viewing layers and layer folders" on page 34.



You can choose to select only an object's strokes or only its fills. You can hide selection highlighting in order to edit objects without viewing highlighting.

When you select an object, the Property inspector displays the object's stroke and fill, its pixel dimensions, and the *x* and *y* coordinates of the object's transformation point.

If you select multiple items of different types on the Stage, such as an object, a button, and a movie clip, the Property inspector indicates a mixed selection. The Property inspector for a mixed selection displays the pixel dimensions and *x* and *y* coordinates of the selected set of items.

You can use the Property inspector for a shape to change the object's stroke and fill. See Chapter 4, "Working with Color," on page 77.

You might want to prevent a group or symbol from being selected and accidentally changed. To do this, you can lock the group or symbol. See "Modifying selections" on page 121.

Selecting objects with the Arrow tool

The Arrow tool lets you select entire objects by clicking an object or dragging to enclose the object within a rectangular selection marquee.

Note: To select the Arrow tool, you can also press the V key. To temporarily switch to the Arrow tool when another tool is active, hold down the Control key (Windows) or Command key (Macintosh).

To select a stroke, fill, group, instance, or text block:

Select the Arrow tool and click the object.

To select connected lines:

Select the Arrow tool and double-click one of the lines.

To select a filled shape and its stroked outline:

Select the Arrow tool and double-click the fill.

To select objects within a rectangular area:

Select the Arrow tool and drag a marquee around the object or objects that you want to select. Instances, groups, and type blocks must be completely enclosed to be selected.

Modifying selections

You can add to selections, select or deselect everything on every layer in a scene, select everything between keyframes, or lock and unlock selected symbols or groups.

To add to a selection:

Hold down the Shift key while making additional selections.

Note: To disable the Shift-selecting option, deselect the option in Flash General Preferences. See "Setting preferences in Flash" on page 22.

To select everything on every layer of a scene:

Choose Edit > Select All, or press Control+A (Windows) or Command+A (Macintosh). Select All doesn't select objects on locked or hidden layers, or layers not on the current Timeline.

To deselect everything on every layer:

Choose Edit > Deselect All, or press Control+Shift+A (Windows) or Command+Shift+A (Macintosh).

To select everything on one layer between keyframes:

Click a frame in the Timeline. For more information, see "Using the Timeline" on page 28.

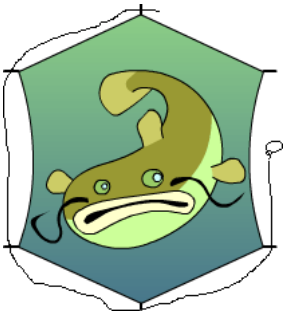
To lock a group or symbol:

Select the group or symbol and choose Modify > Arrange > Lock.

Choose Modify > Arrange > Unlock All to unlock all locked groups and symbols.

Selecting objects with the Lasso tool

To select objects by drawing either a freehand or a straight-edged selection area, you can use the Lasso tool and its Polygon Mode modifier. When using the Lasso tool, you can switch between the freeform and straight-edged selection modes.



To select objects by drawing a freehand selection area:

Select the Lasso tool and drag around the area. End the loop approximately where you started, or let Flash automatically close the loop with a straight line.

To select objects by drawing a straight-edged selection area:

1. Select the Lasso tool and select the Polygon Mode modifier in the Options section of the toolbox.
2. Click to set the starting point.
3. Position the pointer where you want the first line to end, and click. Continue setting end points for additional line segments.
4. To close the selection area, double-click.

To select objects by drawing both freehand and straight-edged selection areas:

1. Select the Lasso tool and deselect the Polygon Mode modifier.
2. To draw a freehand segment, drag on the Stage.
3. To draw a straight-edged segment, hold down Alt-click (Windows) or Option-click (Macintosh) to set start and end points. You can continue switching between drawing freehand and straight-edged segments.
4. To close the selection area, do one of the following:

- If you are drawing a freehand segment, release the mouse button.
- If you are drawing a straight-edged segment, double-click.

Hiding selection highlighting

You can hide selection highlights in order to edit objects without viewing their highlighting. Hiding highlights enables you to see how artwork will appear in its final state while you are selecting and editing objects.

To hide selection highlighting:

Choose View > Hide Edges. Choose the command again to deselect the feature.

Grouping objects

To manipulate elements as a single object, you need to group them. For example, after creating a drawing such as a tree or flower, you might group the elements of the drawing so that you can easily select and move the drawing as a whole.

When you select a group, the Property inspector displays the *x* and *y* coordinates of the group and its pixel dimensions.

You can edit groups without ungrouping them. You can also select an individual object in a group for editing, without ungrouping the objects.

To create a group:

1. Select the objects on the Stage that you want to group.

   You can select shapes, other groups, symbols, text, and so on.

2. Choose Modify > Group, or press Control+G (Windows) or Command+G (Macintosh).

To ungroup objects:

Choose Modify > Ungroup, or press Control+Shift+G (Windows) or Command+Shift+G (Macintosh).

To edit a group or an object within a group:

1. With the group selected, choose Edit > Edit Selected, or double-click the group with the Arrow tool.

   Everything on the page that is not part of the group is dimmed, indicating it is inaccessible.

2. Edit any element within the group.

3. Choose Edit > Edit All, or double-click a blank spot on the Stage with the Arrow tool.

   Flash restores the group to its status as a single entity, and you can work with other elements on the Stage.

Moving, copying, and deleting objects

You can move objects by dragging them on the Stage, cutting and pasting them, using the arrow keys, or using the Property inspector to specify an exact location for them. You can also move objects between Flash and other applications using the Clipboard. You can copy objects by dragging or pasting them, or while transforming them. When you move an object, the Property inspector indicates the new position.

When moving an object with the Arrow tool, you can use the Snap modifier for the Arrow tool to quickly align the object with points on other objects.

Moving objects

To move an object, you can drag the object, use the arrow keys, use the Property inspector, or use the Info panel.

To move objects by dragging:

1. Select an object or multiple objects.

2. Select the Arrow tool, position the pointer over the object, and drag to the new position. To copy the object and move the copy, Alt-drag (Windows) or Option-drag (Macintosh). To constrain the object's movement to multiples of 45°, Shift-drag.

To move objects using the arrow keys:

1. Select an object or multiple objects.

2. Press the arrow key for the direction in which you want the object to move 1 pixel at a time. Press Shift+arrow key to move the selection 10 pixels at a time.

   Note: When Snap to Pixels is selected, the arrow keys move objects by pixel increments on the movie's pixel grid, not by pixels on the screen. See "Pixel snapping" on page 74.

To move objects using the Property inspector:

1. Select an object or multiple objects.
2. If the Property inspector is not visible, choose Window > Properties.
3. Enter *x* and *y* values for the location of the upper left corner of the selection. The units are relative to the upper left corner of the Stage.

   Note: The Property inspector uses the units specified for the Ruler Units option in the Document Properties dialog box. To change the units, see "Using the Property inspector to change document attributes" on page 24.

To move objects using the Info Panel:

1. Select an object or multiple objects.
2. If the Info Panel is not visible, choose Window > Info.
3. Enter *x* and *y* values for the location of the upper left corner of the selection. The units are relative to the upper left corner of the Stage.

Moving and copying objects by pasting

When you need to move or copy objects between layers, scenes, or other Flash files, you should use the pasting technique. You can paste an object in the center of the Stage or in a position relative to its original position.

To move or copy objects by pasting:

1. Select an object or multiple objects.
2. Choose Edit > Cut or Edit > Copy.
3. Select another layer, scene, or file and do one of the following:

- Choose Edit > Paste to paste the selection in the center of the Stage.
- Choose Edit > Paste in Place to paste the selection in the same position relative to the Stage.

About copying artwork with the Clipboard

Elements copied to the Clipboard are anti-aliased, so they look as good in other applications as they do in Flash. This is particularly useful for frames that include a bitmap image, gradients, transparency, or a mask layer.

Graphics pasted from other Flash documents or programs are placed in the current frame of the current layer. How a graphic element is pasted into a Flash scene depends on the type of element it is, its source, and the preferences you have set:

- Text from a text editor becomes a single text object.
- Vector-based graphics from any drawing program become a group that can be ungrouped and edited like any other Flash element.
- Bitmaps become a single grouped object just like imported bitmaps. You can break apart pasted bitmaps or convert pasted bitmaps to vector graphics.

  For information on applying a bitmap fill, see "Working with imported bitmaps" under Help > Using Flash.

  Note: Before pasting graphics from FreeHand into Flash, set your FreeHand export preferences to convert colors to CMYK and RGB for Clipboard formats.

Copying transformed objects

To create a scaled, rotated, or skewed copy of an object, you can use the Transform panel.

To create a transformed copy of an object:

1. Select an object.
2. Choose Window > Transform.
3. Enter scale, rotation, or skew values. See "Scaling objects" on page 128, "Rotating objects" on page 129, and "Skewing objects" on page 130.
4. Click the Create Copy button in the Transform panel (the left button in the lower right corner of the panel).

Deleting objects

Deleting an object removes it from the file. Deleting an instance on the Stage does not delete the symbol from the library.

To delete objects:

1. Select an object or multiple objects.
2. Do one of the following:

- Press Delete or Backspace.
- Choose Edit > Clear.
- Choose Edit > Cut.
- Right-click (Windows) or Control-click (Macintosh) the object and select Cut from the context menu.

Stacking objects

Within a layer, Flash stacks objects based on the order in which they were created, placing the most recently created object at the top of the stack. The stacking order of objects determines how they appear when they are overlapping.

Drawn lines and shapes always appear below groups and symbols on the stack. To move them up the stack, you must group them or make them into symbols. You can change the stacking order of objects at any time.

Layers also affect the stacking order. Everything on Layer 2 appears on top of everything on Layer 1, and so on. To change the order of layers, drag the layer name in the Timeline to a new position. See "Using layers" on page 33.

To change the stacking order of an object:

1. Select the object.
2. Use one of the following commands:

- Choose Modify > Arrange > Bring to Front or Send to Back to move the object or group to the top or bottom of the stacking order.

- Choose Modify > Arrange > Bring Forward or Send Backward to move the object or group up or down one position in the stacking order.

  If more than one group is selected, the groups move in front of or behind all unselected groups, while maintaining their order relative to each other.

Transforming objects

You can transform graphic objects, as well as groups, text blocks, and instances, by using the Free Transform tool or the options in the Modify > Transform submenu. Depending on the type of element you select, you can freely transform, rotate, skew, scale, or distort the element. You can change or add to a selection during a transformation operation.

When you transform an object, group, text box, or instance, the Property inspector for that item displays any changes made to the item's dimensions or position.

A bounding box is displayed during transform operations that involve dragging. The bounding box is rectangular (unless it has been modified with the Distort command or the Envelope modifier; see "Distorting objects" on page 127 and "Modifying shapes with the Envelope modifier" on page 128) with its edges initially aligned to the edges of the Stage. Transformation handles are located on each corner and in the middle of each side. As you drag, the bounding box previews the transformations.

Working with the center point during transformations

During a transformation, a transformation point appears at the center of a selected element. The transformation point is initially aligned with the object's registration point. You can move the transformation point, and you can return it to its default location.

For scaling, skewing, or rotating graphic objects, groups, and text blocks, the point opposite the point you drag is the point of origin by default. For instances, the transformation point is the point of origin by default. You can move the default point of origin for a transformation.

To move the transformation point during a transform operation:

Drag the transformation point.

To realign the transformation point with the element's registration point:

Double-click the transformation point.

To switch the point of origin for a scale or skew transformation:

Alt-drag (Windows) or Option-drag (Macintosh) during the transformation.

Transforming objects freely

You can use the Free Transform tool to freely transform objects, groups, instances, or text blocks. You can perform individual transformations or combine several transformations, such as moving, rotating, scaling, skewing, and distortion.

To transform freely:

1 Select a graphic object, instance, group, or text block on the Stage.

2 Click the Free Transform tool.

Moving the pointer over and around the selection changes the pointer to indicate which transformation function is available.

3 Drag the handles to transform the selection, as follows:

- To move the selection, position the pointer over the object within the bounding box, and drag the object to a new position. Do not drag the transformation point.
- To set the center of rotation or scaling, drag the transformation point to a new location.
- To rotate the selection, position the pointer just outside a corner handle and drag. The selection rotates around the transformation point.

  Shift-drag to rotate in 45° increments.

  Alt-drag (Windows) or Option-drag (Macintosh) to rotate around the opposite corner.
- To scale the selection, drag a corner handle diagonally to scale in two dimensions. Drag a corner handle or a side handle horizontally or vertically to scale in the respective direction only. Shift-drag to resize proportionally.
- To skew the selection, position the pointer on the outline between the transformation handles and drag.

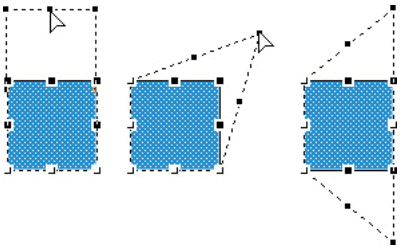
4 To distort shapes, press Control (Windows) or Command (Macintosh) and drag a corner handle or a side handle. Shift-Control-drag (Windows) or Shift-Command-drag (Macintosh) a corner handle to *taper* the object—to move the selected corner and the adjoining corner equal distances from their origins. For more information on distorting objects, see the following section.

Note: The Free Transform tool cannot transform symbols, bitmaps, video objects, sounds, gradients, object groups, or text. If a multiple selection contains any of these, only the shape objects are distorted. To transform text, first convert the characters to shape objects.

5 To end the transformation, click outside the selected object, instance, or text block.

Distorting objects

When you apply a Distort transformation to a selected object, dragging a corner handle or an edge handle on the bounding box moves the corner or edge and realigns the adjoining edges. Shift-dragging a corner point *tapers* the object—that is, it moves that corner and the adjoining corner an equal distance and opposite direction from each other. The adjoining corner is the corner opposite the direction you drag. Control-dragging (Windows) or Command-dragging a middle point on an edge moves the entire edge freely.



Distorting by edge, corner, and taper, respectively

You can distort graphic objects by using the Distort command. You can also distort objects when freely transforming them. See “Transforming objects freely” on page 126.

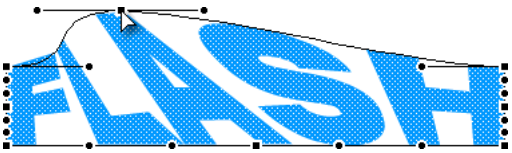
Note: The Distort command cannot modify symbols, bitmaps, video objects, sounds, gradients, object groups, or text. If a multiple selection contains any of these, only the shape objects are distorted. To modify text, first convert the characters to shape objects.

To distort graphic objects:

1. Select a graphic object or objects on the Stage.
2. Choose Modify > Transform > Distort.
3. Place the pointer on one of the transformation handles and drag.
4. To end the transformation, click outside the selected object or objects.

Modifying shapes with the Envelope modifier

The Envelope modifier lets you warp and distort objects. An envelope is a bounding box that contains one or more objects. Changes made to an envelope’s shape affect the shape of the objects contained within the envelope. You edit the shape of an envelope by adjusting its points and tangent handles. See “Adjusting segments” on page 68.



To modify a shape with the Envelope modifier:

1. Select a shape on the Stage.
2. Choose Modify > Transform > Envelope.
3. Drag the points and tangent handles to modify the envelope.

Note: The Envelope modifier cannot modify symbols, bitmaps, video objects, sounds, gradients, object groups, or text. If a multiple selection contains any of these, only the shape objects are distorted. To modify text, first convert the characters to shape objects.

Scaling objects

Scaling an object enlarges or reduces the object horizontally, vertically, or both. You can scale an object by dragging or by entering values in the Transform panel.

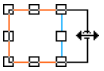
To scale objects by dragging:

1. Select a graphic object or objects on the Stage.
2. Choose Modify > Transform > Scale.

3 Do one of the following:

- To scale the object both horizontally and vertically, drag one of the corner handles. Proportions are maintained as you scale. Shift-drag to scale nonuniformly.

- To scale the object either horizontally or vertically, drag a center handle.



4 To end the transformation, click outside the selected object or objects.

Note: When you increase the size of a number of items, those near the edges of the bounding box might be moved off of the Stage. If this occurs, choose View > Work Area to see the elements that are beyond the edges of the Stage.

To scale an object with the Transform panel:

1 Select the object or objects.

2 Choose Window > Transform.

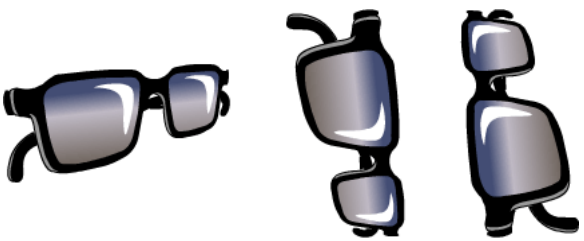
3 Enter a scale value between 1 and 1000 for vertical, horizontal, or both.

4 Select Constrain to maintain proportions.

5 Press Enter (Windows) or Return (Macintosh).

Rotating objects

Rotating an object turns it around its transformation point. The transformation point is aligned with the registration point, which defaults to the center of the object, but you can move the point by dragging it. You can rotate an object by using the Rotate commands, by dragging with the Free Transform tool, or by specifying an angle in the Transform panel. When you rotate an object by dragging, you can also skew and scale the object in the same operation. When you rotate an object using the Transform panel, you can scale the object in the same operation.



Original, rotated right, and rotated left, respectively

To rotate and skew objects by dragging:

1. Select the object or objects on the Stage.
2. Choose Modify > Transform > Rotate and Skew.
3. Do one of the following:

- Drag a corner handle to rotate the object.
- Drag a center handle to skew the object.

4. To end the transformation, click outside the selected object or objects.

To rotate objects by 90°:

1. Select the object or objects.
2. Choose Modify > Transform > Rotate 90° CW to rotate clockwise, or Rotate 90° CCW to rotate counterclockwise.

To rotate objects using the Transform panel:

1. Select the object or objects.
2. Choose Window > Transform.
3. Click Rotate.
4. Enter a rotation angle.
5. Press Enter (Windows) or Return (Macintosh) to apply the rotation.

To rotate and scale an object simultaneously:

1. Select the object or objects.
2. Choose Modify > Transform > Scale and Rotate.
3. In the Scale and Rotate dialog box, enter values for Scale and Rotation.
4. Click OK.

Skewing objects

Skewing an object transforms it by slanting it along one or both axes. You can skew an object by dragging or by entering a value in the Transform panel. To skew an object by dragging, see the procedure for rotating and skewing an object by dragging, under "Rotating objects" on page 129.

To skew an object using the Transform panel:

1. Select the object or objects.
2. Choose Window > Transform.
3. Click Skew.
4. Enter angles for the horizontal and vertical values.

Flipping objects

You can flip objects across their vertical or horizontal axis without moving their relative position on the Stage.

Original, flipped horizontally, and flipped vertically, respectively

To flip an object:

1. Select the object.
2. Choose Modify > Transform > Flip Vertical or Flip Horizontal.

Restoring transformed objects

When you scale, rotate, and skew instances, groups, and type with the Transform panel, Flash saves the original size and rotation values with the object. This allows you to remove the transformations you applied and restore the original values.

You can undo only the most recent transformation performed in the Transform panel by choosing Edit > Undo. You can reset all transformations performed in the Transform panel by clicking the Reset button in the panel before you deselect the object.

To restore a transformed object to its original state:

1. Select the transformed object.
2. Choose Modify > Transform > Remove Transform.

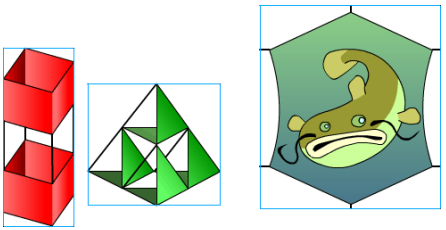
To reset a transformation performed in the Transform panel:

With the transformed object still selected, click the Reset button in the Transform panel.

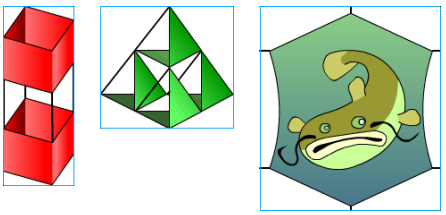
Aligning objects

The Align panel enables you to align selected objects along the horizontal or vertical axis. You can align objects vertically along the right edge, center, or left edge of the selected objects, or horizontally along the top edge, center, or bottom edge of the selected objects. Edges are determined by the bounding boxes enclosing each selected object.

Using the Align panel, you can distribute selected objects so that their centers or edges are evenly spaced. You can resize selected objects so that the horizontal or vertical dimensions of all objects match those of the largest selected object. You can also align selected objects to the Stage. You can apply one or more Align options to selected objects.



Original



Objects aligned to the top edge of the uppermost object

To align objects:

1. Select the objects to align.
2. Choose Window > Align.
3. In the Align panel, select To Stage to apply alignment modifications relative to stage dimensions.
4. Select alignment buttons to modify the selected objects:

- For Align, select Align Left, Align Horizontal Center, Align Right, Align Top, Align Vertical Center, or Align Bottom.
- For Distribute, select Distribute Top, Distribute Horizontal Center, Distribute Bottom, Distribute Left, Distribute Vertical Center, or Distribute Right.
- For Match Size, select Match Width, Match Height, or Match Width and Height.
- For Space, select Space Horizontally or Space Vertically.

Breaking apart groups and objects

To separate groups, instances, and bitmaps into ungrouped, editable elements, you use the Break Apart command. Breaking apart significantly reduces the file size of imported graphics.

Although you can choose Edit > Undo immediately after breaking apart a group or object, breaking apart is not entirely reversible. It affects objects as follows:

- It severs a symbol instance's link to its master symbol.
- It discards all but the current frame in an animated symbol.
- It converts a bitmap to a fill.
- It places each character into a separate text block when applied to text blocks.
- It converts characters to outlines when applied to a single text character. See "Breaking text apart" on page 144.

The Break Apart command should not be confused with the Ungroup command. The Ungroup command separates grouped objects, returning grouped elements to the state they were in prior to grouping. It does not break apart bitmaps, instances, or type, or convert type to outlines.

To break apart groups or objects:

1. Select the group, bitmap, or symbol that you want to break apart.
2. Choose Modify > Break Apart.

Note: Breaking apart animated symbols, or groups within an interpolated animation, is not recommended and might have unpredictable results. Breaking apart complex symbols and large blocks of text can take a long time. You might need to increase the application's memory allocation to properly break apart complex objects.

CHAPTER 8
Working with Text

You can include text in your Macromedia Flash MX movies in a variety of ways. You can create text blocks containing *static* text, text whose contents and appearance you determine when you author the movie. You can also create *dynamic* or *input* text fields. Dynamic text fields display dynamically updating text, such as sports scores or stock quotes. Input text fields allow users to enter text for forms, surveys, or other purposes.

You can orient text horizontally, with left-to-right flow, or vertically (static text only), with left-to-right or right-to-left flow. You can choose the following attributes for text: font, point size, style, color, tracking, kerning, baseline shift, alignment, margins, indents, and line spacing. "Setting text attributes" on page 139.

You can transform text like an object—rotating, scaling, skewing, and flipping it—and still edit its characters. See "About transforming text" on page 144. When you're working with horizontal text, you can link text blocks to URLs. See "Linking text to a URL (horizontal text only)" on page 145.

When you work with Flash FLA files, Flash substitutes fonts in the FLA file with other fonts installed on your system if the specified fonts are not on your system. You can choose options to control which fonts are used in substitution. Substitute fonts are used for display on your system only. The font selection in the FLA file remains unchanged. See "Substituting missing fonts" on page 145.

Flash also lets you create a symbol from a font so that you can export the font as part of a shared library and use it in other Flash movies. See "Creating font symbols" on page 143.

You can break text apart and reshape its characters. For additional text-handling capabilities, you can manipulate text in FreeHand and import the FreeHand file into Flash, or export the file from FreeHand as a SWF file. See "Breaking text apart" on page 144.

Flash movies can use Type 1 PostScript fonts, TrueType, and bitmap fonts (Macintosh only). You can spell-check text by copying text to the Clipboard using the Movie Explorer and pasting the text into an external text editor. See "Using the Movie Explorer" on page 40.

You can create text fields in your Flash movies for user input or for displaying text that updates dynamically. Just like movie clip instances, text field instances are ActionScript objects that have properties and methods. Once you give a text field an instance name, you can manipulate it with ActionScript. However, unlike movie clips, you cannot write ActionScript code inside a text instance because they don't have Timelines.

You can preserve rich text formatting in text fields.

You can format static, input, and dynamic text using the Property inspector. You can also format input and dynamic text using ActionScript.

ActionScript has events for dynamic and input text fields that you can capture and use to trigger scripts.

You can also use text fields to create scrolling text.

For an interactive introduction to creating text in Flash, choose Help > Lessons > Adding and Editing Text.

About embedded fonts and device fonts

When you use a font installed on your system in a Flash movie, Flash embeds the font information in the Flash SWF file, ensuring that the font is displayed properly in the Flash Player. Not all fonts displayed in Flash can be exported with a movie. To verify that a font can be exported, use the View > Antialias Text command to preview the text; jagged type indicates that Flash does not recognize that font's outline and will not export the text.

You can use special fonts in Flash called device fonts as an alternative to embedding font information (for horizontal text only). Device fonts are not embedded in the Flash SWF file. Instead, the Flash Player uses whatever font on the local computer most closely resembles the device font. Because device font information is not embedded, using device fonts yields a somewhat smaller Flash movie file size. In addition, device fonts can be sharper and more legible than embedded fonts at small point sizes (below 10 points). However, because device fonts are not embedded, if users do not have a font installed on their system that corresponds to the device font, text may look different than expected on a user's system.

Flash includes three device fonts, named _sans (similar to Helvetica or Arial), _serif (similar to Times Roman), and _typewriter (similar to Courier). To specify a font as a device font, you select one of the Flash device fonts in the Property inspector. During movie playback, Flash selects the first device font that is located on the user's system. You can specify text set in a device font to be selectable, so that users can copy and paste text that appears in your movie. See "Using device fonts (horizontal text only)" on page 142.

You can use device fonts for static text (text that you create when authoring a movie and that does not change when the movie is displayed) or dynamic text (text that updates periodically through input from a file server, such as sports scores or weather data). For more information on dynamic text, see "Setting dynamic and input text options" on page 142.

Creating text

You can create three types of text fields: static, dynamic, and input. All text fields support Unicode. See "Unicode text encoding in Flash movies" on page 366.

- Static text fields display text that doesn't change characters dynamically.
- Dynamic text fields display dynamically updating text, such as sports scores, stock quotes, or weather reports.
- Input text fields enable users to enter text in forms or surveys.

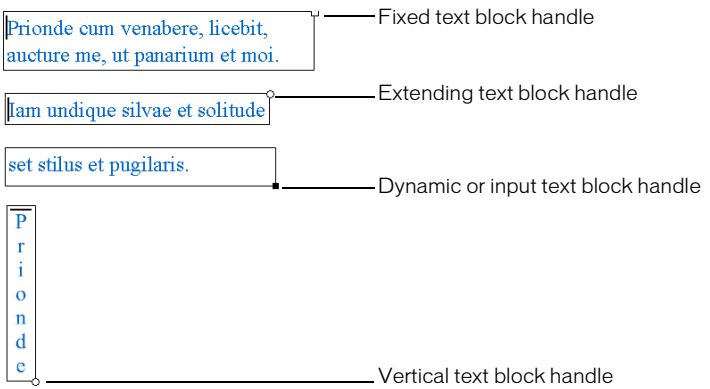
You can create horizontal text (with a left-to-right flow) or static vertical text (with either a right-to-left or left-to-right flow) in Flash. By default, text is created with horizontal orientation. You can choose preferences to make vertical text the default orientation and to set other options for vertical text.

You can also create scrolling text fields. See "Creating scrolling text" under Help > Using Flash.

To create text, you place text blocks on the Stage using the Text tool. When creating static text, you can place text on a single line that expands as you type, or in a fixed-width block (for horizontal text) or fixed-height block (for vertical text) that expands and wraps words automatically. When creating dynamic or input text, you can place text on a single line, or create a text block with a fixed width and height.

Flash displays a handle on the corner of a text block to identify the type of text block:

- For static horizontal text blocks that extend, a round handle appears at the upper right corner of the text block.
- For static horizontal text blocks with a defined height, a square handle appears at the upper right corner of the text block.
- For static vertical text with right-to-left orientation that extends, a round handle appears at the lower left corner of the text block.
- For static vertical text with right-to-left orientation and a fixed height, a square handle appears at the lower left corner of the text block.
- For static vertical text with left-to-right orientation that extends, a round handle appears at the lower right corner of the text block.
- For static vertical text with left-to-right orientation and a fixed height, a square handle appears at the lower right corner of the text block.
- For dynamic or input text blocks that extend, a round handle appears at the lower right corner of the text block.
- For dynamic or input text with a defined height and width, a square handle appears at the lower right corner of the text block.
- For dynamic scrollable text blocks, the round or square handle becomes solid black instead of hollow. See “Creating scrolling text” under Help > Using Flash.



You can shift-double-click the handle of dynamic and input text fields to create text blocks that don’t expand when you enter text on the Stage. This allows you to create a text block of a fixed size and fill it with more text than it can display to create scrolling text. See “Creating scrolling text” under Help > Using Flash.

After you use the Text tool to create a text field, you use the Property inspector to indicate which type of text field you want and set values to control the way the text field and its contents appear in the Flash movie.

To set preferences for vertical text:

1. Choose Edit > Preferences and click the Editing tab in the Preferences dialog box.
2. Under Vertical Text, select Default Text Orientation to make new text blocks automatically orient vertically.
3. Select Right to Left Text Flow to make vertical text automatically flow right-to-left.
4. Select No Kerning to prevent kerning from being applied to vertical text. (Kerning remains enabled for horizontal text.)

To create text:

1. Select the Text tool.
2. Choose Window > Property inspector.
3. In the Property inspector, choose a text type from the pop-up menu to specify the type of text field:

- Choose Dynamic Text to create a field that displays dynamically updating text.
- Choose Input Text to create a field in which users can enter text.
- Choose Static Text to create a field which cannot update dynamically.

4. For static text only: in the Property inspector, click the Text Direction button and select an option to specify the orientation of the text:

- Select Horizontal to flow text left to right horizontally (the default setting).
- Select Vertical Left-to-Right to flow text vertically, left to right.
- Select Vertical Right-to-Left to flow text vertically, right to left.

Note: Layout options for vertical text are disabled if the text is dynamic or input. Only static text can be vertical.

5. Do one of the following:

- To create a text block that displays text in a single line, click where you want the text to start.
- To create a text block with a fixed width (for horizontal text) or fixed height (for vertical text), position the pointer where you want the text to start and drag to the desired width or height.

Note: If you create a text block that extends past the edge of the Stage as you type, the text isn't lost. To make the handle accessible again, add line breaks, move the text block, or choose View › Work Area.

6. Select text attributes in the Property inspector as described in the following section.

To change the dimensions of a text block:

Drag its resize handle.

To switch a text block between fixed-width or fixed-height and extending:

Double-click the resize handle.

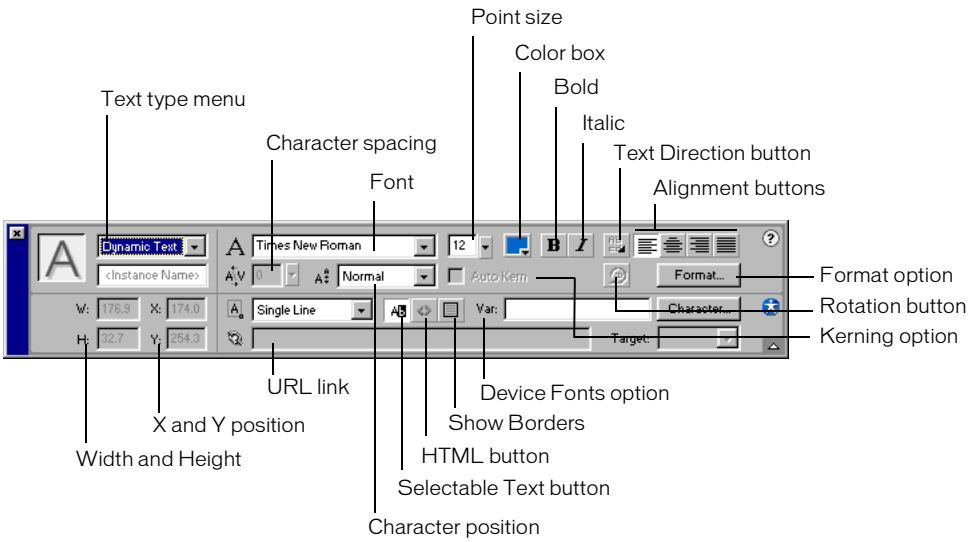
Setting text attributes

You can set the font and paragraph attributes of text. A font is an assortment of alphanumeric characters in a particular typeface design. Font attributes include font family, point size, style, color, character spacing, auto kerning, and character position. Paragraph attributes include alignment, margins, indents, and line spacing.

By default, font information is embedded in a published Flash movie (SWF file). You can choose to use device fonts, rather than embedding font information (horizontal text only). See "About embedded fonts and device fonts" on page 136.

When text is selected, you use the Property inspector to change font and paragraph attributes, and to direct Flash to use device fonts rather than embedding font information.

When creating new text, Flash uses the current text attributes. To change the font or paragraph attributes of existing text, you must first select the text.



Choosing a font, point size, style, and color

You can set the font, point size, style, and color for selected text using the Property inspector.

When setting the text color, you can use only solid colors, not gradients. To apply a gradient to text, you must convert the text to its component lines and fills. See "Breaking text apart" on page 144.

To choose a font, point size, style, and color with the Property inspector:

1 Select the Text tool.

To apply settings to existing text, use the Text tool to select a text block or text blocks on the Stage.

2 If the Property inspector is not already displayed, choose Window > Property inspector.

3 In the Property inspector, click the triangle next to the Font text box and select a font from the list, or enter a font name.

Note: The fonts _sans, _serif, and _typewriter are device fonts. Font information for these fonts is not embedded in the Flash SWF file. Device fonts can be used only with horizontal text. See "About embedded fonts and device fonts" on page 136.

4 Click the triangle next to the Point Size value and drag the slider to select a value, or enter a font size value.

Text size is set in points, regardless of the current ruler units.

5 To apply bold or italic style, click the Bold button or the Italic button.

6 To choose a fill color for text, click the color box and do one of the following:

- Choose a color from the color pop-up window.
- Type a color's hexadecimal value in the text box in the color pop-up window.
- Click the Color Picker button in the upper right corner of the pop-up window and choose a color from the system Color Picker.

For more information on selecting colors, see Chapter 4, "Working with Color," on page 77.

Setting character spacing, kerning, and character position

Character spacing inserts a uniform amount of space between characters. You use character spacing to adjust the spacing of selected characters or entire blocks of text.

Kerning controls the spacing between pairs of characters. Many fonts have built-in kerning information. For example, the spacing between an *A* and a *V* is often less than the spacing between an *A* and a *D*. To use a font's built-in kerning information to space characters, you use the Kern option.

For horizontal text, tracking and kerning set the horizontal distance between characters. For vertical text, tracking and kerning set the vertical distance between characters.

For vertical text, you can set kerning to be off by default in Flash Preferences. When kerning is turned off for vertical text in Preferences, you can leave the option selected in the Property inspector, and kerning will be applied to horizontal text only. To set Preferences for vertical text, see "Creating text" on page 136.

Character position controls where text appears in relation to its baseline. For horizontal text, character position moves characters up or down (above or below the baseline). For vertical text, character position moves characters to the left or right of the baseline.

To set character spacing, kerning, and character position:

1 Select the Text tool.

To apply settings to existing text, use the Text tool to select a text block or text blocks on the Stage.

2 If the Property inspector is not already displayed, choose Window > Properties.

3 In the Property inspector, set the following options:

- To specify character spacing, click the triangle next to the Character Spacing value and drag the slider to select a value, or enter a value in the text box.
- To use a font's built-in kerning information, select Kern.

- To specify character position, click the triangle next to the Character Position option and select a position from the menu: Normal places text on the baseline, Superscript places text above the baseline (horizontal text) or to the right of the baseline (vertical text), and Subscript places text below the baseline (horizontal text) or to the left of the baseline (vertical text).

Setting alignment, margins, indents, and line spacing

Alignment determines the position of each line of text in a paragraph relative to edges of the text block. Horizontal text is aligned relative to the left and right edges of the text block, and vertical text is aligned relative to the top and bottom edges of the text block. Text can be aligned to one edge of the text block, centered within the text block, or aligned to both edges of the text block (full justification).

Margins determine the amount of space between the border of a text block and a paragraph of text. Indents determine the distance between the margin of a paragraph and the beginning of the first line. For horizontal text, indents move the first line to the right the specified distance. For vertical text, indents move the first line down the specified distance.

Line spacing determines the distance between adjacent lines in a paragraph. For vertical text, line spacing adjusts the space between vertical columns.

To set alignment, margins, indents, and line spacing for horizontal text:

1. Select the Text tool.

To apply settings to existing text, use the Text tool to select a text block or text blocks on the Stage.

2. Choose Window > Properties.
3. In the Property inspector, click Format Options and set the following options:

- To set alignment, click the Left, Center, Right, or Full Justification button.
- To set left or right margins, click the triangle next to the Left Margin or Right Margin value and drag the slider to select a value, or enter a value in the numeric field.
- To specify indents, click the triangle next to the Indent value and drag the slider to select a value, or enter a value in the numeric field. (Either the right or left line is indented, depending on whether the text flows right to left or left to right.)
- To specify line spacing, click Format options. Click the triangle next to the Line Spacing value and drag the slider to select a value, or enter a value in the numeric field.

To set alignment, margins, indents, and line spacing for vertical text:

1. Select the Text tool.
2. To apply settings to existing text, select a text block or text blocks on the Stage.
3. Choose Window > Properties.
4. In the Property inspector, click Format Options and set the following options:

- To set alignment, click the Top, Center, Bottom, or Full Justification button.
- To set top or bottom margins, use the Left or Right margin control. Click the triangle next to the Left Margin value to set the top margin or the Right Margin value to set the bottom margin and drag the slider to select a value, or enter a value in the numeric field.

- To specify indents, click the triangle next to the Indent value and drag the slider to select a value, or enter a value in the numeric field.
- To specify line spacing, click the triangle next to the Line Spacing value and drag the slider to select a value, or enter a value in the numeric field.

Using device fonts (horizontal text only)

When you create text, you can specify that the Flash Player use device fonts to display certain text blocks, so that Flash does not embed the font for that text. This can decrease the file size of the movie and increase legibility at text sizes below 10 points.

When working with horizontal text, you can specify that text set in device fonts be selectable by users viewing your movie. See "About embedded fonts and device fonts" on page 136.

To specify that text be displayed using a device font:

1. Select text blocks on the Stage containing text that you want to display using a device font.
2. Choose Window > Properties.
3. In the Property inspector, choose Static Text from the pop-up menu.
4. Select Use Device Fonts.

To make horizontal text selectable by a user:

1. Select the horizontal text that you want to make selectable by a user.
2. Choose Window > Properties.
3. In the Property inspector, choose Static Text from the pop-up menu.
4. If the text is not already specified as using a device font, select Use Device Fonts.
5. Click Selectable.

Setting dynamic and input text options

The Property inspector lets you specify options that control the way dynamic and input text appears in the Flash movie.

To set options for dynamic and input text:

1. Click inside an existing dynamic text field.

   To create a new dynamic text field, see "Creating text" on page 136.

2. In the Property inspector, make sure Dynamic or Input is displayed in the pop-up menu. Set any of the following options:

- For Instance Name, enter the instance name for the text field.
- Choose Multiline to display the text in multiple lines, Single Line to display the text as one line, or Multiline No Wrap to display text in multiple lines that break only if the last character is a breaking character, such as Enter (Windows) or Return (Macintosh).
- Select the Render text as HTML button to preserve rich text formatting, such as fonts and hyperlinks, with the appropriate HTML tags. For more information, see "Preserving rich text formatting" under Help > Using Flash.

Select the Show Border button to display a black border and white background for the text field.

- Select the Selectable button to enable users to select dynamic text. Deselect this option to prevent users from selecting the dynamic text.
- For Variable, enter the variable name for the text field.

Creating font symbols

To use a font as a shared library item, you can create a font symbol in the Library panel. You then assign the symbol an identifier string and a URL where the movie containing the font symbol will be posted. This enables you to link to the font and use it in a Flash movie without having to embed the font in the movie.

For information on linking to a shared font symbol from other movies, see "Using shared library assets" on page 165.

To create a font symbol:

1. Open the library to which you want to add a font symbol.
2. Choose New Font from the Options menu in the upper right corner of the Library panel.

In the Font Symbol Properties dialog box, enter a name for the font symbol in the Name text box.

3. Select a font from the Font menu or enter the name of a font in the Font text box.
4. If desired, select Bold or Italic to apply the selected style to the font.
5. Click OK.

To assign an identifier string to a font symbol:

1. Select the font symbol in the Library panel.
2. Do one of the following:

- Choose Linkage from the Options menu in the upper right corner of the Library panel.
- Right-click (Windows) or Control-click (Macintosh) the font symbol name in the Library panel, and choose Linkage from the context menu.

3. Under Linkage in the Linkage Properties dialog box, select Export for Runtime Sharing.
4. In the Identifier text box, enter a string to identify the font symbol.
5. In the URL text box, enter the URL where the SWF movie file that contains the font symbol will be posted.
6. Click OK.

Editing text

You can use most common word-processing techniques to edit text in Flash. You use the Cut, Copy, and Paste commands to move text within a Flash file as well as between Flash and other applications.

To check the spelling of text, you can copy it to the Clipboard using the Movie Explorer, and paste the text into an external text editor, then run the spell checker. See "Using the Movie Explorer" on page 40.

Selecting text

When editing text or changing text attributes, you must first select the characters you want to change.

To select characters within a text block:

1 Select the Text tool.

2 Do one of the following:

- Drag to select characters.
- Double-click to select a word.

Click to specify the beginning of the selection and Shift-click to specify the end of the selection.

- Press Ctrl+A (Windows) or Command+A (Macintosh) to select all the text in the block.

To select text blocks:

Select the Arrow tool and click a text block. Shift-click to select multiple text blocks.

About transforming text

You can transform text blocks in the same ways you can other objects. You can scale, rotate, skew, and flip text blocks to create interesting effects. When you scale a text block as an object, increases or decreases in point size are not reflected in the Property inspector.

The text in a transformed text block can still be edited, although severe transformations may make it difficult to read.

For more information about transforming text blocks, see Chapter 7, "Working with Graphic Objects," on page 119.

Breaking text apart

You can break apart text to place each character in a separate text block. Once you break text apart, you can quickly distribute the text blocks to separate layers to easily animate each block separately. For information on distributing objects to layers, see "Distributing objects to layers for tweened animation" on page 172. For general information on animation, see Chapter 10, "Creating Animation," on page 169.

Note: You cannot break apart text in scrollable text fields.

Text broken apart into separate text blocks

You can also convert text to its component lines and fills to reshape, erase, and otherwise manipulate it. As with any other shape, you can individually group these converted characters, or change them to symbols and animate them. Once you've converted text to lines and fills, you can no longer edit them as text.

Text broken apart into shapes, with modifications applied

To break apart text:

1. Select the Arrow tool and click a text block.
2. Choose Modify > Break Apart. Each character in the selected text is placed into a separate text block. The text remains in the same position on the Stage.
3. Choose Modify > Break Apart again to convert the characters to shapes on the Stage.

Note: The Break Apart command applies only to outline fonts such as TrueType fonts. Bitmap fonts disappear from the screen when you break them apart. PostScript fonts can be broken apart only on Macintosh systems.

Linking text to a URL (horizontal text only)

You can link horizontal text to a URL to let users jump to other files by clicking the text.

To link horizontal text to a URL:

1. Do one of the following:

- Use the Text tool to select text in a text block.
- Use the Arrow tool to select a text block on the Stage, to link all the text in the block to a URL.

2. If the Property inspector is not already displayed, choose Window > Properties.
3. For Link, enter the URL to which you want to link the text block.

Note: To create a link to an e-mail address, use the mailto: URL. For example, for the Macromedia Flash Wish URL, enter mailto:wish-flash@macromedia.com.

Substituting missing fonts

If you work with a document containing fonts that aren't installed on your system (for example, a document you received from another designer), Flash substitutes the missing fonts with fonts available on your system. You can select the fonts on your system to be substituted for the missing fonts, or you can let Flash substitute missing fonts with the Flash System Default Font (specified in General Preferences).

Note: Substituting missing fonts while editing a Flash document does not change the fonts that are specified in the Flash document.

If you install a previously missing font on your system and restart Flash, the font will be displayed in any documents using the font, and the font will be removed from the Missing Fonts dialog box.

Selecting substitute fonts

An alert box indicating missing fonts in a document appears the first time a scene containing a missing font is displayed on the Stage. If you publish or export the document without displaying any scenes containing the missing fonts, the alert box appears during the publish or export operation. If you choose to select substitute fonts, the Font Mapping dialog box appears, listing all missing fonts in the document and letting you select a substitute for each.

Note: If the document contains many missing fonts, a delay may occur while Flash generates the list of missing fonts.

You can apply the missing font to new or existing text in the current document. The text is displayed on your system using the substitute font, but the missing font information is saved with the document. If the document is reopened on a system that includes the missing font, the text is displayed using that font.

Text attributes such as font size, leading, kerning, and so on may need to be adjusted when the text is displayed in the missing font, because the formatting you apply is based on the appearance of the text in the substitute font.

To specify font substitution:

1. When the Missing Fonts alert appears, do one of the following:

- Click Choose Substitute Fonts to select substitute fonts from fonts installed on your system and proceed to step 2.
- Click Use Default to use the Flash System Default Font to substitute all missing fonts and to dismiss the Missing Fonts alert.

2. In the Font Mapping dialog box, click on a font in the Missing Fonts column to select it. Shift-click to select multiple missing fonts, to map them all to the same substitute font.

   The default substitute fonts are displayed in the Mapped To column, until you select substitute fonts.

3. Choose a font from the Substitute Font pop-up menu.
4. Repeat steps 2–4 for all missing fonts.
5. Click OK.

Working with substitute fonts

You can use the Font Mapping dialog box to change the substitute font mapped to a missing font, to view all the substitute fonts you have mapped in Flash on your system, and to delete a substitute font mapping from your system. You can also turn off the Missing Fonts alert to prevent the alert from appearing.

When you work with a document that includes missing fonts, the missing fonts are displayed in the font list in the Property inspector. When you select substitute fonts, the substitute fonts are also displayed in the font list.

To view all the missing fonts in a document and reselect substitute fonts:

1. With the document active in Flash, Choose Edit > Font Mapping.
2. Select a substitute font, as described beginning in step 2 in the preceding procedure.

To view all the font mappings saved on your system and delete font mappings:

1. Close all documents in Flash.
2. Choose Edit > Font Mapping.
3. To delete a font mapping, select the mapping and press Delete.
4. Click OK.

To turn off the Missing Fonts alert, do one of the following:

- To turn the alert off for the current document, in the Missing Fonts alert box select Don't Show Again For This Document, Always Use Substitute Fonts. Choose Edit > Font Mapping to view mapping information for the document again.
- To turn the alert off for all documents, choose Edit > Preferences and click the Warnings tab. Deselect Warn on Missing Font and click OK. Reselect the option to turn alerts on again.

CHAPTER 9
Using Symbols, Instances, and Library Assets

A *symbol* is a graphic, button, or movie clip that you create once in Macromedia Flash MX and can reuse throughout your movie or in other movies. A symbol can include artwork that you import from another application. Any symbol you create automatically becomes part of the library for the current document. (For more information on the library, see "Using the library" on page 54.)

An *instance* is a copy of a symbol located on the Stage or nested inside another symbol. An instance can be very different from its symbol in color, size, and function. Editing the symbol updates all of its instances, but applying effects to an instance of a symbol updates only that instance. You can also create font symbols in Flash. See "Creating font symbols" on page 143.

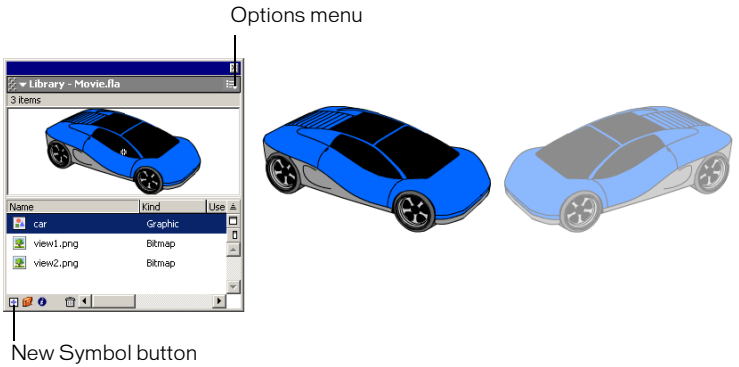
Using symbols in your movies dramatically reduces file size; saving several instances of a symbol requires less storage space than saving multiple copies of the contents of the symbol. For example, you can reduce the file size of your movies if you convert static graphics such as background images into symbols that you then reuse. Using symbols can also speed movie playback, because a symbol needs to be downloaded to the Flash Player only once.

You can share symbols among Flash movies as runtime or author-time shared library assets. For runtime shared assets, you can link assets in a source movie to any number of destination movies, without importing the assets into the destination movies. For author-time shared assets, you can update or replace a symbol with any other symbol available on your local network. See "Using shared library assets" on page 165.

If you import library assets that have the same name as assets already in the library, you can resolve naming conflicts without accidentally overwriting existing assets. See "Resolving conflicts between library assets" on page 168.

You can also add ActionScript actions to symbols. See "Writing Scripts with ActionScript" under Help > Using Flash.

For an interactive introduction to using symbols and instances, choose Help > Lessons > Symbols.



A symbol in the library and two instances on the Stage, with effects applied to the instance on the right

Types of symbol behavior

Each symbol has a unique Timeline and Stage, complete with layers. When you create a symbol you choose the symbol type, depending on how you want to use the symbol in the movie.

- Use graphic symbols for static images and to create reusable pieces of animation that are tied to the Timeline of the main movie. Graphic symbols operate in sync with the movie's Timeline. Interactive controls and sounds won't work in a graphic symbol's animation sequence.
- Use button symbols to create interactive buttons in the movie that respond to mouse clicks, rollovers or other actions. You define the graphics associated with various button states, and then assign actions to a button instance. See "Assigning actions to a button" under Help > Using Flash.
- Use movie clip symbols to create reusable pieces of animation. Movie clips have their own multiframe Timeline that plays independent of the main movie's Timeline—think of them as mini-movies inside a main movie that can contain interactive controls, sounds, and even other movie clip instances. You can also place movie clip instances inside the Timeline of a button symbol to create animated buttons.
- Use font symbols to export a font and use it in other Flash movies. See "Creating font symbols" on page 143.

  Flash provides built-in *components*, movie clips with defined parameters, that allow you to easily add user interface elements, such as buttons, check boxes, or scroll bars, to your movies. For more information, see Chapter 15, "Using Components," on page 289.

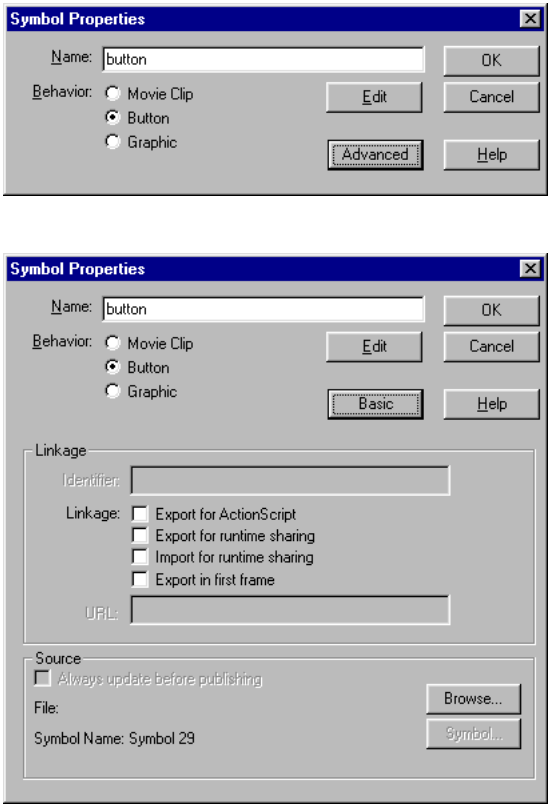
Note: To preview interactivity and animation in movie clip symbols in the Flash authoring environment, you must choose Control › Enable Live Preview. See "Working with Movie Clips and Buttons" under Help › Using Flash.

Creating symbols

You can create a symbol from selected objects on the Stage, or you can create an empty symbol and make or import the content in symbol-editing mode. Symbols can have all the functionality that you can create with Flash, including animation.

By using symbols that contain animation, you can create movies with a lot of movement while minimizing file size. Consider creating animation in a symbol when there is a repetitive or cyclic action—the up-and-down motion of a bird's wings, for example.

You can also add symbols to your movie by using runtime or author-time shared library assets. See "Using shared library assets" on page 165.



Symbol Properties dialog box in basic and advanced views. The dialog box is titled Create New Symbol if you are creating a new symbol, and Convert to Symbol if you are converting a graphic to a symbol.

To convert selected elements to a symbol:

1. Select an element or several elements on the Stage and do one of the following:

- Choose Insert > Convert to Symbol.
- Drag the selection to the Library panel.
- Right-click (Windows) or Control-click (Macintosh) and choose Convert to Symbol from the context menu.

2. In the Convert to Symbol dialog box, type the name of the symbol and choose the behavior—Graphic, Button, or Movie Clip. See "Types of symbol behavior" on page 150.
3. For Registration, click on a square in the diagram to place the registration point for the symbol.
4. Click OK.

   Flash adds the symbol to the library. The selection on the Stage becomes an instance of the symbol. You cannot edit an instance directly on the Stage—you must open it in symbol-editing mode; see "Editing symbols" on page 157.

To create a new empty symbol:

1. Make sure that nothing is selected on the Stage and do one of the following:

- Choose Insert > New Symbol.
- Click the New Symbol button at the bottom left of the Library panel.
- Choose New Symbol from the Library options menu in the upper right corner of the Library panel.

2. In the Create New Symbol dialog box, type the name of the symbol and choose the behavior—Graphic, Button, or Movie Clip. See "Types of symbol behavior" on page 150.
3. Click OK.

   Flash adds the symbol to the library and switches to symbol-editing mode. In symbol-editing mode, the name of the symbol appears above the upper left corner of the Stage, and a cross hair indicates the symbol's registration point.
4. To create the symbol content, use the Timeline, draw with the drawing tools, import media, or create instances of other symbols.
5. When you have finished creating the symbol content, do one of the following to return to movie-editing mode:

- Click the Back button at the left side of the information bar above the Stage.
- Choose Edit > Edit Document.
- Click the scene name in the information bar above the Stage.

Converting animation on the Stage into a movie clip

If you've created an animated sequence on the Stage and want to reuse it elsewhere in the movie, or if you want to manipulate it as an instance, you can select it and save it as a movie clip symbol.

To convert animation on the Stage into a movie clip:

1. On the main Timeline, select every frame in every layer of the animation on the Stage that you want to use.

   Note: For information on selecting frames, see"Using the Timeline" on page 28.

2. Do one of the following to copy the frames:

- Right-click (Windows) or Control-click (Macintosh) any selected frame and choose Copy Frames from the context menu. Choose Cut if you want to delete the sequence after converting it to a movie clip.
- Choose Edit > Copy Frames. Choose Cut Frames if you want to delete the sequence after converting it to a movie clip.

3. Deselect your selection and make sure nothing on the Stage is selected. Choose Insert > New Symbol.
4. In the Create New Symbol dialog box, name the symbol. For Behavior, choose Movie Clip, then click OK.

   Flash opens a new symbol for editing in symbol-editing mode.

5. On the Timeline, click Frame 1 on Layer 1, and choose Edit > Paste Frames.

   This pastes the frames (and any layers and layer names) you copied from the main Timeline to the Timeline of this movie clip symbol. Any animation, buttons, or interactivity from the frames you copied now becomes an independent animation (a movie clip symbol) that you can reuse throughout your movie.

6. When you have finished creating the symbol content, do one of the following to return to movie-editing mode:

- Click the Back button at the left side of the information bar above the Stage.
- Choose Edit > Edit Document.
- Click the scene name in the information bar above the Stage.

Duplicating symbols

Duplicating a symbol lets you use an existing symbol as a starting point for creating a new symbol.

You can also use instances to create versions of the symbol with different appearances. See "Creating instances" on page 154.

To duplicate a symbol using the Library panel:

1. Select a symbol in the Library panel.
2. Do one of the following to duplicate the symbol:

- Right-click (Windows) or Control-click (Macintosh) and choose Duplicate from the context menu.
- Choose Duplicate from the Library options menu.

To duplicate a symbol by selecting an instance:

1. Select an instance of the symbol on the Stage.
2. Choose Modify > Duplicate Symbol.

   The symbol is duplicated and the instance is replaced with an instance of the duplicate symbol.

Creating instances

Once you've created a symbol, you can create instances of that symbol wherever you like throughout the movie, including inside other symbols. When you modify the symbol, all instances of the symbol are updated.

Instances are given default names when you create them. You can apply custom names to instances in the Property inspector.

To create a new instance of a symbol:

1. Select a layer in the Timeline.

   Flash can place instances only in keyframes, always on the current layer. If you don't select a keyframe, the instance will be added to the first keyframe to the left of the current frame.

   Note: A keyframe is a frame in which you define a change in the animation. For more information, see "Working with frames in the Timeline" on page 31.

2. Choose Window > Library to open the library.
3. Drag the symbol from the library to the Stage.
4. If you created an instance of a graphic symbol, choose Insert > Frame to add the number of frames that will contain the graphic symbol.

To apply a custom name to an instance:

1. Select the instance on the Stage.
2. Choose Window > Properties if the Property inspector is not visible.
3. Enter a name in the Instance Name text box on the left side of the Property inspector (below the Symbol Behavior pop-up list).

After creating an instance of a symbol, use the Property inspector to specify color effects, assign actions, set the graphic display mode, or change the behavior of the instance. The behavior of the instance is the same as the symbol behavior, unless you specify otherwise. Any changes you make affect only the instance and not the symbol. See "Changing the color and transparency of an instance" on page 160.

Creating buttons

Buttons are actually four-frame interactive movie clips. When you select the button behavior for a symbol, Flash creates a Timeline with four frames. The first three frames display the button's three possible states; the fourth frame defines the active area of the button. The Timeline doesn't actually play, it simply reacts to pointer movement and actions by jumping to the appropriate frame.

To make a button interactive in a movie, you place an instance of the button symbol on the Stage and assign actions to the instance. The actions must be assigned to the instance of the button in the movie, not to frames in the button's Timeline.

Each frame in the Timeline of a button symbol has a specific function:

- The first frame is the Up state, representing the button whenever the pointer is not over the button.
- The second frame is the Over state, representing the button's appearance when the pointer is over it.
- The third frame is the Down state, representing the button's appearance as it is clicked.
- The fourth frame is the Hit state, defining the area that will respond to the mouse click. This area is invisible in the movie.

You can also create buttons using the ActionScript MovieClip object. See "Using button events with movie clips to trigger scripts" on page 296. You can add buttons to your movie using button components. See "The PushButton component" on page 299 and "The RadioButton component" on page 300.

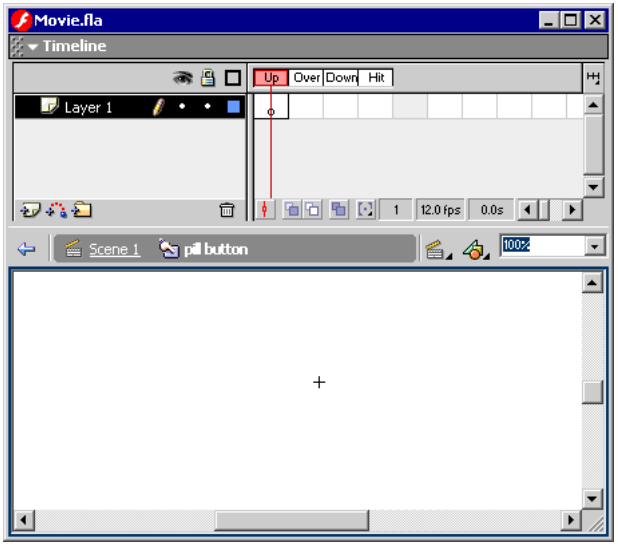
For an interactive lesson on creating buttons in Flash, choose Help > Lessons > Buttons.

To create a button:

1. Choose Edit > Deselect All to ensure that nothing is selected on the Stage.
2. Choose Insert > New Symbol, or press Control+F8 (Windows) or Command+F8 (Macintosh).

   To create the button, you convert the button frames to keyframes.
3. In the Create New Symbol dialog box, enter a name for the new button symbol, and for Behavior choose Button.

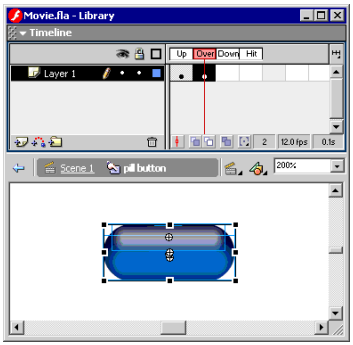
   Flash switches to symbol-editing mode. The Timeline header changes to display four consecutive frames labeled Up, Over, Down, and Hit. The first frame, Up, is a blank keyframe.



4. To create the Up state button image, use the drawing tools, import a graphic, or place an instance of another symbol on the Stage.

You can use a graphic or movie clip symbol in a button, but you cannot use another button in a button. Use a movie clip symbol if you want the button to be animated.

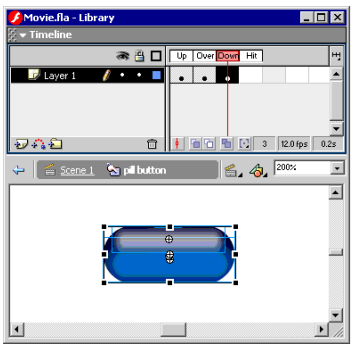
5 Click the second frame, labeled Over, and choose Insert > Keyframe.

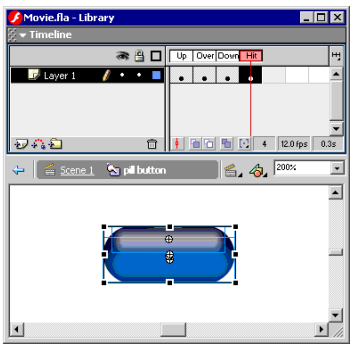


Flash inserts a keyframe that duplicates the contents of the Up frame.

6 Change the button image for the Over state.

7 Repeat steps 5 and 6 for the Down frame and the Hit frame.





The Hit frame is not visible on the Stage, but it defines the area of the button that responds when clicked. Make sure that the graphic for the Hit frame is a solid area large enough to encompass all the graphic elements of the Up, Down, and Over frames. It can also be larger than the visible button. If you do not specify a Hit frame, the image for the Up state is used as the Hit frame.

You can create a disjoint rollover, in which rolling over a button changes another graphic on the Stage. To do this, you place the Hit frame in a different location than the other button frames.

8 To assign a sound to a state of the button, select that state's frame in the Timeline, choose Window > Properties, and then select a sound from the Sound menu in the Property inspector. See "Adding sounds to buttons" on page 120.

9 When you've finished, choose Edit > Edit Document. Drag the button symbol out of the Library panel to create an instance of it in the movie.

Enabling, editing, and testing buttons

By default, Flash keeps buttons disabled as you create them, to make it easier to select and work with them. When a button is disabled, clicking the button selects it. When a button is enabled, it responds to the mouse events that you've specified as if the movie were playing. You can still select enabled buttons, however. In general, it is best to disable buttons as you work, and enable buttons to quickly test their behavior.

To enable and disable buttons:

Choose Control > Enable Simple Buttons. A check mark appears next to the command to indicate buttons are enabled. Choose the command again to disable buttons.

Any buttons on the Stage now respond. As you move the pointer over a button, Flash displays the Over frame; when you click within the button's active area, Flash displays the Down frame.

To select an enabled button:

Use the Arrow tool to drag a selection rectangle around the button.

To move or edit an enabled button:

1. Select the button, as described above.
2. Do one of the following:

- Use the arrow keys to move the button.
- If the Property inspector is not visible, choose Window > Properties to edit the button in the Property inspector, or Alt-double-click (Windows) or Option-double-click the button (Macintosh).

To test a button, do one of the following:

- Choose Control > Enable Simple Buttons. Move the pointer over the enabled button to test it.
- Select the button in the Library panel and click the Play button in the Library preview window.

  Movie clips in buttons are not visible in the Flash authoring environment. See "Previewing and testing movies" on page 39.
- Choose Control > Test Scene or Control > Test Movie.

Editing symbols

When you edit a symbol, Flash updates all the instances of that symbol in the movie. Flash provides three ways for you to edit symbols. You can edit the symbol in context with the other objects on the Stage using the Edit in Place command. Other objects are dimmed to distinguish them from the symbol you are editing. The name of the symbol you are editing is displayed in an information bar at the top of the Stage, to the right of to the current scene name.

You can also edit a symbol in a separate window, using the Edit in New Window command. Editing a symbol in a separate window lets you see both the symbol and the main Timeline at the same time. The name of the symbol you are editing is displayed in the information bar at the top of the Stage.

You edit the symbol by changing the window from the Stage view to a view of only the symbol, using symbol-editing mode. The name of the symbol you are editing is displayed in the information bar at the top of the Stage, to the right of the current scene name.

When you edit a symbol, all instances of the symbol throughout the movie are updated to reflect your edits. While editing a symbol, you can use any of the drawing tools, import media, or create instances of other symbols.

To edit a symbol in place:

1 Do one of the following:

- Double-click an instance of the symbol on the Stage.
- Select an instance of the symbol on the Stage and right-click (Windows) or Control-click (Macintosh), and choose Edit in Place from the context menu.
- Select an instance of the symbol on the Stage and choose Edit > Edit in Place.

2 Edit the symbol as needed.

3 To exit Edit in Place mode and return to movie-editing mode, do one of the following:

- Click the Back button at the left side of the information bar at the top of the Stage.
- Choose the current scene name from the Scene pop-up menu in the information bar at the top of the Stage.
- Choose Edit > Edit Document.

To edit a symbol in a new window:

1 Select an instance of the symbol on the Stage and right-click (Windows) or Control-click (Macintosh), and choose Edit in New Window from the context menu.

2 Edit the symbol as needed.

3 Click the Close box in the upper right corner (Windows) or upper left corner (Macintosh) to close the new window, and click in the main movie window to return to editing the main movie.

To edit a symbol in symbol-editing mode:

1 Do one of the following

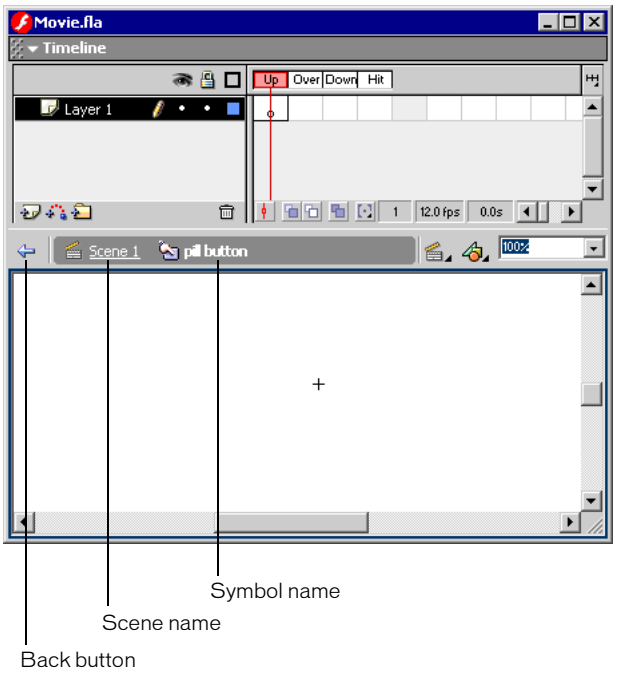
- Double-click the symbol's icon in the Library panel.
- Select an instance of the symbol on the Stage and right-click (Windows) or Control-click (Macintosh) and choose Edit from the context menu.
- Select an instance of the symbol on the Stage and choose Edit > Edit Symbols.
- Select the symbol in the Library panel and choose Edit from the Library options menu, or right-click (Windows) or Control-click (Macintosh) the symbol in the Library panel and choose Edit from the context menu.

2 Edit the symbol as needed on the Stage.

3 To exit symbol-editing mode and return to editing the movie, do one of the following:

- Click the Back button at the left side of the information bar at the top of the Stage.
- Choose Edit > Edit Document.

- Click the scene name in the information bar at the top of the Stage.



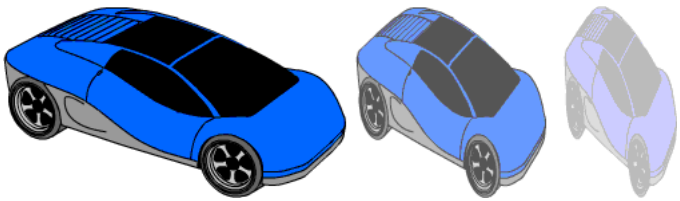
Changing instance properties

Each symbol instance has its own properties that are separate from the symbol. You can change the tint, transparency, and brightness of an instance; redefine how the instance behaves (for example, change a graphic to a movie clip); and set how animation plays inside a graphic instance. You can also skew, rotate, or scale an instance without affecting the symbol.

In addition, you can name a movie clip or button instance so that you can use ActionScript to change its properties. See "Working with Movie Clips and Buttons" under Help > Using Flash.

To edit instance properties, you use the Property inspector (Windows > Properties).

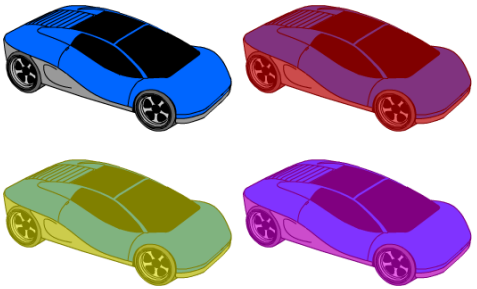
The properties of an instance are saved with it. If you edit a symbol or relink an instance to a different symbol, any instance properties you've changed still apply to the instance.



Original symbol and two modified instances

Changing the color and transparency of an instance

Each instance of a symbol can have its own color effect. To set color and transparency options for instances, you use the Property inspector. Settings on the Property inspector also affect bitmaps placed within symbols.



Symbol instances, each with its own color effect

When you change the color and transparency for an instance in a specific frame, Flash makes the change as soon as it displays that frame. To make gradual color changes, you must apply a motion tween. When tweening color, you enter different effect settings in starting and ending keyframes of an instance, and then tween the settings to make the instance's colors shift over time. See "Tweening instances, groups, and type" on page 173.

Note: If you apply a color effect to a movie clip symbol that includes multiple frames, Flash applies the effect to every frame in the movie clip symbol.

To change the color and transparency of an instance:

1. Select the instance on the Stage and choose Window > Properties.
2. In the Property inspector, choose one of the following options from the Color pop-up menu:

- Brightness adjusts the relative lightness or darkness of the image, measured on a scale from black (–100%) to white (100%). Click on the triangle and drag the slider or enter a value in the text box to adjust Brightness.
- Tint colors the instance with the same hue. Use the Tint slider in the Property inspector to set the tint percentage, from transparent (0%) to completely saturated (100%). Click on the triangle and drag the slider or enter a value in the text box to adjust Tint. To select a color, enter red, green, and blue values in the respective text boxes, or click on the color box and select a color from the pop-up window or click the Color Picker button.
- Alpha adjusts the transparency of the instance, from transparent (0%) to completely saturated (100%). Click on the triangle and drag the slider or enter a value in the text box to adjust Alpha.

- Advanced separately adjusts the red, green, blue, and transparency values of an instance. This is most useful when you want to create and animate subtle color effects on objects such as bitmaps. The controls on the left let you reduce the color or transparency values by a specified percentage. The controls on the right let you reduce or increase the color or transparency values by a constant value.

  The current red, green, blue, and alpha values are multiplied by the percentage values, and then added to the constant values in the right column, producing the new color values. For example, if the current red value is 100, setting the left slider to 50% and the right slider to 100 produces a new red value of 150 ([100 x .5] + 100 = 150).

  Note: The advanced settings in the Effect panel implement the function $(a * y + b) = x$ where **a** is the percentage specified in the left set of text boxes, **y** is the color of the original bitmap, **b** is the value specified in the right set of text boxes, and **x** is the resulting effect (between 0 and 255 for RGB, and 0 and 100 for alpha transparency).

You can also change the color of an instance using the ActionScript Color object. For detailed information on the Color object, see its entry in the online ActionScript Dictionary.

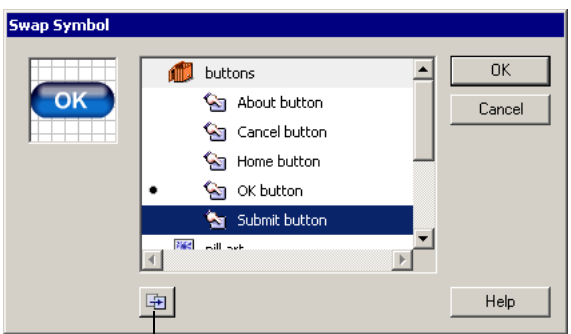
Assigning a different symbol to an instance

You can assign a different symbol to an instance to display a different instance on the Stage and preserve all the original instance properties, such as color effects or button actions.

For example, say you're creating a cartoon with a Rat symbol for your character, but decide to change the character to a Cat. You could switch the Cat for the Rat symbol and have the updated character appear in roughly the same location in all of your frames.

To assign a different symbol to an instance:

1. Select the instance on the Stage and choose Window > Properties.
2. Click the Swap button in the Property inspector.
3. In the Swap Symbol dialog box, select a symbol that will replace the one currently assigned to the instance. To duplicate a selected symbol, click the Duplicate Symbol button at the bottom of the dialog box.



Duplicate Symbol button

Duplicating lets you base a new symbol on an existing one in the library and minimizes copying if you're making several symbols that differ just slightly.

4. Click OK.

To replace all instances of a symbol:

1. Drag a symbol with the same name as the one you are replacing into the Library panel.
2. In the Resolve Library Item Conflict dialog box, click Replace. For more information, see "Resolving conflicts between library assets" on page 168.

Changing an instance's type

You can change an instance's type to redefine its behavior in a movie. For example, if a graphic instance contains animation that you want to play independently of the main movie's Timeline, you could redefine the graphic instance as a movie clip instance.

To change an instance's type:

1. Select the instance on the Stage and choose Window > Properties.
2. Choose Graphic, Button, or Movie Clip from the pop-up menu in the upper left corner of the Property inspector.

Setting the animation for graphic instances

You can determine how animation sequences inside a graphic instance play during the movie by setting options in the Property inspector.

An animated graphic symbol is tied to the Timeline of the movie in which the symbol is placed. In contrast, a movie clip symbol has its own independent Timeline. Animated graphic symbols, because they use the same Timeline as the main movie, display their animation in movie-editing mode. Movie clip symbols appear as static objects on the Stage and do not appear as animations in the Flash editing environment.

To set the animation of a graphic instance:

1. Select a graphic instance on the Stage and choose Window > Properties.
2. In the Property inspector, choose an animation option from the pop-up menu below the instance name:

- Loop loops all the animation sequences contained in the current instance for as many frames as the instance occupies.
- Play Once plays the animation sequence beginning from the frame you specify to the end of the animation and then stops.
- Single Frame displays one frame of the animation sequence. Specify which frame to display.

Breaking apart instances

To break the link between an instance and a symbol and make the instance into a collection of ungrouped shapes and lines, you "break apart" the instance. This is useful for changing the instance substantially without affecting any other instance. If you modify the source symbol after breaking apart the instance, the instance is not updated with the changes.

To break apart an instance of a symbol:

1. Select the instance on the Stage.
2. Choose Modify > Break Apart.

   This breaks the instance into its component graphic elements.
3. Use the painting and drawing tools to modify these elements as desired.

Getting information about instances on the Stage

As you create a movie, it can be difficult to identify a particular instance of a symbol on the Stage, particularly if you are working with multiple instances of the same symbol. You can identify instances using the Property inspector, the Info panel, or the Movie Explorer.

The Property inspector and Info panel display the symbol name of the selected instance and an icon that indicate its type—graphic, button, or movie clip. In addition, you can view the following information:

- In the Property inspector, you can view the instance's behavior and settings—for all instance types, color settings, location, size, and registration point; for graphics, the loop mode and first frame that contains the graphic; for buttons, the instance name (if assigned) and tracking option; for movie clips, the instance name (if assigned).
- In the Info panel, you can view the location and size of a selected instance.
- In the Movie Explorer, you can view the contents of the current movie, including instances and symbols. See "Using the Movie Explorer" on page 40.

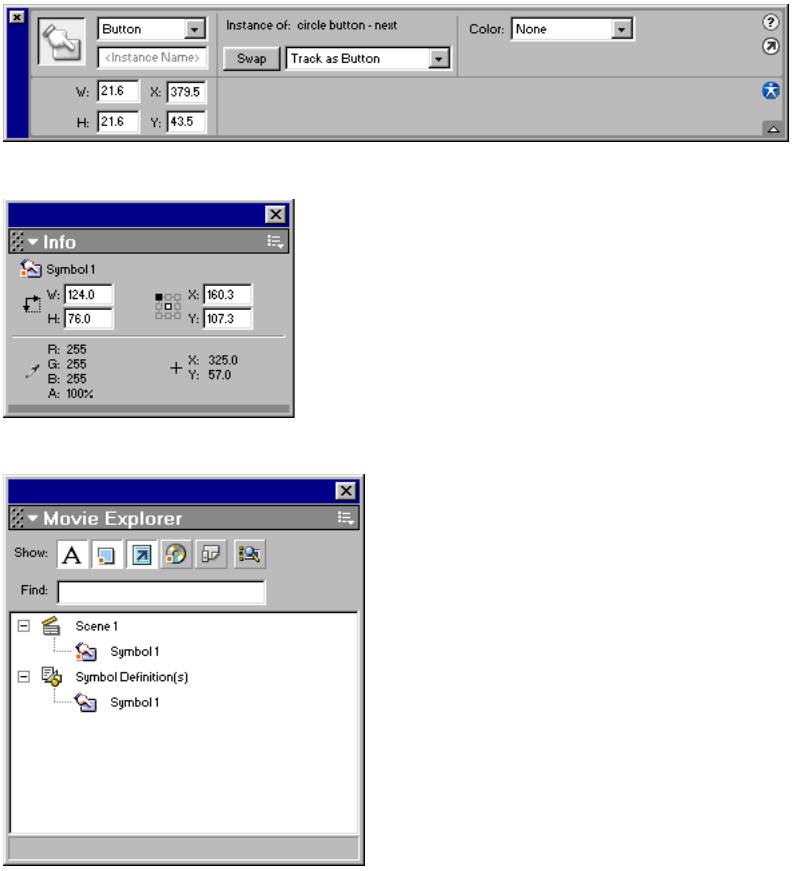
In addition, in the Actions panel, you can view any actions assigned to a button or movie clip.

To get information about an instance on the Stage:

1. Select the instance on the Stage.
2. Display the Property inspector or panel you want to use:

- To display the Property inspector, choose Window > Properties.
- To display the Info panel, choose Window > Info.
- To display the Movie Explorer, choose Window > Movie Explorer.

- To display the Actions panel, choose Window > Actions.For more information on the Movie Explorer, see “Using the Movie Explorer” on page 40.



Information for a selected button instance displayed in the Property inspector, Info panel, and Movie Explorer.

To view the symbol definition for the selected symbol in the Movie Explorer:

1. Click the Show Buttons, Movie Clips, and Graphics button at the top of the Movie Explorer.
2. Right-click (Windows) or Control-click (Macintosh) and choose Show Symbol Instances and Go to Symbol Definition from the context menu; or choose these options from the pop-up menu in the upper right corner of the Movie Explorer.

To jump to the scene containing instances of a selected symbol:

1. Display the symbol definitions as described in the previous procedure.
2. Right-click (Windows) or Control-click (Macintosh) and choose Show Movie Elements and Go to Symbol Definition from the context menu; or choose these options from the pop-up menu in the upper right corner of the Movie Explorer.

Copying library assets between movies

You can copy library assets from a source movie into a destination movie in a variety of ways: by copying and pasting the asset, by dragging and dropping the asset, or by opening the library of the source movie in the destination movie and dragging the source movie assets into the destination movie.

You can also share symbols between movies as runtime or author-time shared library assets. See "Using shared library assets" on page 165.

If you attempt to copy assets that have the same name as existing assets in the destination movie, the Resolve Library Conflicts dialog box lets you choose whether to overwrite the existing assets or to preserve the existing assets and add the new assets with modified names. See "Resolving conflicts between library assets" on page 168. You can organize library assets in folders to minimize name conflicts when copying assets between movies. See "Working with folders in the Library panel" on page 56.

To copy a library asset by copying and pasting:

1. Select the asset on the Stage in the source movie.
2. Select Edit > Copy.
3. Make the destination movie the active movie.
4. Place the pointer on the Stage and select Edit > Paste. Choose Edit > Paste in Place to place the asset in the same location as it was in the source movie.

To copy a library asset by dragging:

1. With the destination movie open in Flash, select the asset in the Library panel in the source movie.
2. Drag the asset into the Library panel in the destination movie.

To copy a library asset by opening the source movie library in the destination movie:

1. With the destination movie active in Flash, choose File > Open As Library.
2. Select the source movie in the Open As Library dialog box and click Open.
3. Drag an asset from the source movie library onto the Stage or into the library of the destination movie.

Using shared library assets

Shared library assets enable you to use assets from a source movie in multiple destination movies.You can share library assets in two different ways:

- For *runtime* shared assets, assets from a source movie are linked as external files in a destination movie. Runtime assets are loaded into the destination movie during movie playback—that is, at runtime. The source movie containing the shared asset does not need to be available on your local network when you author the destination movie. However, the source movie must be posted to a URL in order for the shared asset to be available to the destination movie at runtime.
- For *author-time* shared assets, you can update or replace any symbol in a movie you are authoring, with any other symbol available on your local network. The symbol in the destination movie can be updated as you author the movie. The symbol in the destination movie retains its original name and properties, but its contents are updated or replaced with those of the symbol you select.

Using shared library assets can optimize your workflow and movie asset management in numerous ways. For example, you can use shared library assets to share a font symbol across multiple sites, provide a single source for elements in animations used across multiple scenes or movies, or create a central resource library to use for tracking and controlling revisions.

Working with runtime shared assets

Using runtime shared library assets involves two procedures: First, the author of the source movie defines a shared asset in the source movie, and enters an identifier string for the asset and a URL where the source movie will be posted.

Second, the author of the destination movie defines a shared asset in the destination movie and enters an identifier string and URL identical to those used for the shared asset in the source movie. Alternatively, the destination movie author can drag the shared assets from the posted source movie into the destination movie library.

In either scenario, the source movie must be posted to the specified URL in order for the shared assets to be available for the destination movie.

Defining runtime shared assets in a source movie

You use the Symbol Properties dialog box or the Linkage Properties dialog box to define sharing properties for an asset in a source movie, to make the asset accessible for linking to destination movies.

To define a runtime shared asset in a source movie:

1. With the source movie open, choose Window > Library to display the Library panel.
2. Do one of the following:

- Select a movie clip, button, or graphic symbol in the Library panel and choose Properties from the Library options menu. Click the Advanced button to expand the Properties dialog box.
- Select a font symbol, sound, or bitmap and choose Linkage from the Library options menu.

3. For Linkage, select Export for Runtime Sharing to make the asset available for linking to the destination movie.
4. Enter an identifier for the symbol in the Identifier text field. Do not include spaces. This is the name Flash will use in identifying the asset when linking to the destination movie.

   Note: The Linkage Identifier is also used by Flash to identify a movie clip or button that is used as an object in ActionScript. See "Working with Movie Clips and Buttons" under Help › Using Flash.

5. Enter the URL where the SWF file containing the shared asset will be posted.
6. Click OK.

   When you publish the movie, you must post the SWF file to the URL specified in step 5, so that the shared assets will be available to destination movies.

Linking to runtime shared assets from a destination movie

You use the Symbol Properties dialog box or the Linkage Properties dialog box to define sharing properties for an asset in a destination movie, to link the asset to a shared asset in a source movie. If the source movie is posted to a URL, you can also link a shared asset to a destination movie by dragging the asset from the source movie to the destination movie.

You can turn off sharing for a shared asset in the destination movie, to embed the symbol in the destination movie.

To link a shared asset to a destination movie by entering the identifier and URL:

1. In the destination movie, choose Window > Library to display the Library panel.
2. Do one of the following:

- Select a movie clip, button, or graphic symbol in the Library panel and choose Properties from the Library options menu. Click the Advanced button to expand the Properties dialog box.
- Select a font symbol and choose Linkage from the Library options menu.

3. For Linkage, select Import for Runtime Sharing to link to the asset in the source movie.
4. Enter an identifier for the symbol in the Identifier text field that is identical to the identifier used for the symbol in the source movie. Do not include spaces.
5. Enter the URL where the SWF source file containing the shared asset is posted.
6. Click OK.

To link a shared asset to a destination movie by dragging:

1. In the destination movie, choose File > Open or Open as Library.
2. In the Open or Open as Library dialog box, select the source movie and click Open.
3. Drag the shared asset from the source movie Library panel into the Library panel or onto the Stage in the destination movie.

To turn off linkage for a symbol in a destination movie:

1. In the destination movie, select the linked symbol in the Library panel and do one of the following:

- If the asset is a movie clip, button, or graphic symbol, choose Properties from the Library options menu.
- If the asset is a font symbol, choose Linkage from the Library options menu.

2. In the Symbol Properties dialog box or the Linkage Properties dialog box, deselect Import for Runtime Sharing.
3. Click OK.

Updating or replacing symbols using author-time sharing

You can update or replace a movie clip, button, or graphic symbol in a movie with any other symbol in a FLA file accessible on your local network. The original name and properties of the symbol in the destination movie are preserved, but the contents of the symbol are replaced with the contents of the symbol you select. Any assets that the selected symbol uses are also copied into the destination movie.

To update or replace a symbol:

1. With the movie open, select movie clip, button, or graphic symbol and choose Properties from the Library options menu.
2. To select a new FLA file, under Source in the Symbol Properties dialog box, click Browse.
3. In the Open dialog box, navigate to a FLA file containing the symbol that will be used to update or replace the selected symbol in the Library panel, and click Open.

4 To select a new symbol in the FLA file, under Source, click Symbol.

5 Navigate to a symbol and click Open.

6 In the Symbol Properties dialog box, under Source, select Always Update Before Publishing to automatically update the asset if a new version is found at the specified source location.

7 Click OK to close the Symbol Properties or Linkage Properties dialog box.

Resolving conflicts between library assets

If you import or copy a library asset into a movie that already contains a different asset of the same name, you can choose whether to replace the existing item with the new item. This option is available with all the methods for importing or copying library assets, including:

- Copying and pasting an asset from a source movie
- Dragging an asset from a source movie or a source movie library
- Importing an asset
- Adding a shared library asset from a source movie
- Using a component from the components panel

The Resolve Library Items dialog box appears when you attempt to place items that conflict with existing items in a movie. A conflict exists when you copy an item from a source movie that already exists in the destination movie and the items have different modification dates. You can avoid having naming conflicts by organizing your assets inside folders in your movie's library. The dialog box also appears when you paste a symbol or component into your movie's Stage and you already have a copy of the symbol or component that has a different modification date from the one you're pasting.

If you choose not to replace the existing items, Flash attempts to use the existing item instead of the conflicting item that you are pasting. For example, if you copy a symbol named Symbol 1 and paste the copy into the stage of a movie that already contains a symbol named Symbol 1, an instance of the existing Symbol 1 is created.

If you choose to replace the existing items, the existing items (and all their instances) are replaced with the new items of the same name. If you cancel the Import or Copy operation, the operation is canceled for all items (not just those items that conflict in the destination movie).

Only identical library item types may be replaced with each other. That is, you cannot replace a sound named Test with a bitmap named Test. In such cases, the new items are added to the library with the word Copy appended to the name.

Note: Replacing library items using this method is not undoable. Be sure to save a backup of your FLA file before performing complex paste operations that are resolved by replacing conflicting library items.

To resolve naming conflicts between library assets:

If the Resolve Library Conflict dialog box appears when you are importing or copying library assets into a movie, do one of the following:

- Click Don't Replace Existing Items to preserve the existing assets in the destination movie.
- Click Replace Existing Items to replace the existing assets and their instances with the new items of the same name.

CHAPTER 10
Creating Animation

You create animation in a Macromedia Flash MX document by changing the contents of successive frames. You can make an object move across the Stage, increase or decrease its size, rotate, change color, fade in or out, or change shape. Changes can occur independently of, or in concert with, other changes. For example, you can make an object rotate and fade in as it moves across the Stage.

There are two methods for creating an animation sequence in Flash: *tweened animation,* and *frame-by-frame animation*. In tweened animation, you create starting and ending frames and let Flash create the frames in between. Flash varies the object's size, rotation, color, or other attributes evenly between the starting and ending frames to create the appearance of movement. See "About tweened animation" on page 169. In frame-by-frame animation, you create the image in every frame. See "About frame-by-frame animation" on page 170.

To simplify the process of creating tweened animation, you can distribute multiple objects to separate layers. See "Distributing objects to layers for tweened animation" on page 172.

You can use a mask layer to create a hole through which the contents of one or more underlying layers are visible. Using an animated movie clip, you can create a dynamic layer mask. See "Using mask layers" on page 183.

For an interactive introduction to animation, choose Help > Lessons > Creating Tweened Animation.

Note: You can also create animation programmatically using ActionScript to change the properties of an object, symbol, or instance. For more information about the ActionScript language, see Chapter 12, "Understanding the ActionScript Language," on page 203. For detailed information about using ActionScript elements, see the online ActionScript Dictionary in the Help menu.

About tweened animation

Flash can create two types of tweened animation, *motion tweening* and *shape tweening.*

- In motion tweening, you define properties such as position, size, and rotation for an instance, group, or text block at one point in time, and then you change those properties at another point in time. You can also apply a motion tween along a path. See "Tweening instances, groups, and type" on page 173 and "Tweening motion along a path" on page 176.
- In shape tweening, you draw a shape at one point in time, and then you change that shape or draw another shape at another point in time. Flash interpolates the values or shapes for the frames in between, creating the animation. See "Tweening shapes" on page 178.

Tweened animation is an effective way to create movement and changes over time while minimizing file size. In tweened animation, Flash stores only the values for the changes between frames.

To quickly prepare elements in a document for tweened animation, distribute objects to layers. See “Distributing objects to layers for tweened animation” on page 172.

You can apply tweened animation to an object on a mask layer to create a dynamic mask. For information on mask layers, see “Using mask layers” on page 183.

About frame-by-frame animation

Frame-by-frame animation changes the contents of the Stage in every frame and is best suited to complex animation in which an image changes in every frame instead of simply moving across the Stage. Frame-by-frame animation increases file size more rapidly than tweened animation. In frame-by-frame animation, Flash stores the values for each complete frame. For information on frame-by-frame animations, see “Creating frame-by-frame animations” on page 180.

About layers in animation

Each scene in a Flash document can consist of any number of layers. As you animate, you use layers and layer folders to organize the components of an animation sequence and to separate animated objects so they don't erase, connect, or segment each other. If you want Flash to tween the movement of more than one group or symbol at once, each must be on a separate layer. Typically, the background layer contains static artwork, and each additional layer contains one separate animated object.

When a document has several layers, tracking and editing the objects on one or two of them can be difficult. This task is easier if you work with the contents of one layer at a time. Layer folders help you organize layers into manageable groups that you can expand and collapse to view only the layers relevant to your current task. See “Using layers” on page 33.

Creating keyframes

A keyframe is a frame where you define changes in the animation. When you create frame-by-frame animation, every frame is a keyframe. In tweened animation, you define keyframes at significant points in the animation and let Flash create the contents of frames in between. Flash displays the interpolated frames of a tweened animation as light-blue or light-green with an arrow drawn between keyframes. Because Flash documents save the shapes in each keyframe, you should create keyframes only at those points in the artwork where something changes.

Keyframes are indicated in the Timeline: a keyframe with content on it is represented by a solid circle, and an empty keyframe is represented by an empty circle before the frame. Subsequent frames that you add to the same layer will have the same content as the keyframe.

To create a keyframe, do one of the following:

- Select a frame in the Timeline and choose Insert > Keyframe.
- Right-click (Windows) or Control-click (Macintosh) a frame in the Timeline and choose Insert Keyframe.

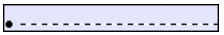
Representations of animations in the Timeline

Flash distinguishes tweened animation from frame-by-frame animation in the Timeline as follows:

- Motion tweens are indicated by a black dot at the beginning keyframe; intermediate tweened frames have a black arrow with a light-blue background.

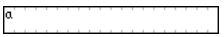
- Shape tweens are indicated by a black dot at the beginning keyframe; intermediate frames have a black arrow with a light-green background.

- A dashed line indicates that the tween is broken or incomplete, such as when the final keyframe is missing.

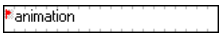


- A single keyframe is indicated by a black dot. Light-gray frames after a single keyframe contain the same content with no changes and have a black line with a hollow rectangle at the last frame of the span.

- A small *a* indicates that the frame has been assigned a frame action with the Actions panel.



- A red flag indicates that the frame contains a label or comment.



- A gold anchor indicates that the frame is a named anchor.

About frame rates

The frame rate, the speed at which the animation is played, is measured in number of frames per second. A frame rate that's too slow makes the animation appear to stop and start; a frame rate that's too fast blurs the details of the animation. A frame rate of 12 frames per second (fps) usually gives the best results on the Web. QuickTime and AVI movies generally have a frame rate of 12 fps, while the standard motion-picture rate is 24 fps.

The complexity of the animation and the speed of the computer on which the animation is being played affect the smoothness of the playback. Test your animations on a variety of machines to determine optimum frame rates.

Because you specify only one frame rate for the entire Flash document, it's a good idea to set this rate before you begin creating animation. See "Using the Property inspector to change document attributes" on page 24.

Extending still images

When you create a background for animation, it's often necessary that a still image remain the same for several frames. Adding a span of new frames (not keyframes) to a layer extends the contents of the last keyframe in all the new frames.

To extend a still image through multiple frames:

1. Create an image in the first keyframe of the sequence.
2. Select a frame to the right, marking the end of the span of frames that you want to add.
3. Choose Insert > Frame.

To use a shortcut to extend still images:

1. Create an image in the first keyframe.
2. Alt-drag (Windows) or Option-drag (Macintosh) the keyframe to the right. This creates a span of new frames, but without a new keyframe at the end point.

Distributing objects to layers for tweened animation

You can quickly distribute selected objects in a frame to separate layers to apply tweened animation to the objects. The objects can be on one or more layers initially. Flash distributes each object to a new, separate layer. Any objects that you don't select (including objects in other frames) are preserved in their original positions.

You can apply the Distribute to Layers command to any type of element on the Stage, including graphic objects, instances, bitmaps, video clips, and broken-apart text blocks.

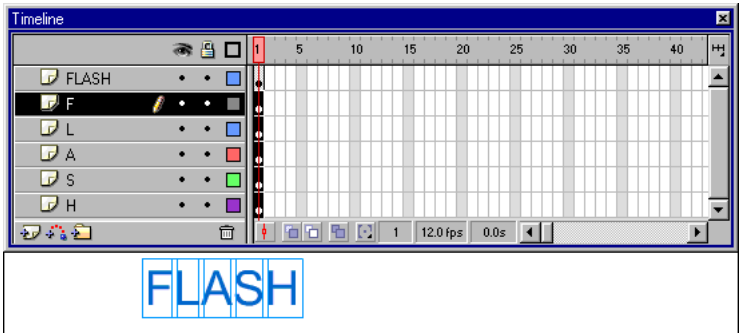
Applying the Distribute to Layers command to broken-apart text makes it easy to create animated text. The characters in the text are placed in separate text blocks during the Break Apart operation, and each text block is placed on a separate layer during the Distribute to Layers process. For information on breaking text apart, see "Breaking text apart" on page 144.

About new layers

New layers created during the Distribute to Layers operation are named according to the name of the element that each contains:

- A new layer containing a library asset (such as a symbol, bitmap, or video clip) is given the same name as the asset.
- A new layer containing a named instance is given the name of the instance.
- A new layer containing a character from a broken-apart text block is named with the character.
- A new layer containing a graphic object (which has no name) is named Layer1 (or Layer2, and so on), because graphic objects do not have names.

Flash inserts new layers below the any selected layers in the Timeline. The new layers are arranged top to bottom, in the order in which the selected elements were originally created. For broken-apart text, the layers are arranged in the order of the characters, whether left-to-right, right-to-left, or top-to-bottom. For example, if you break apart the text *FLASH* and distribute it to layers, the new layers, named F, L, A, S, and H, are arranged top to bottom, immediately below the layer initially containing the text.



Distributing objects to layers

To distribute objects to layers, you select the objects in one or more layers and choose Distribute to Layers from the Modify menu or from the context menu.

To tween distributed objects, follow the procedure in "Tweening instances, groups, and type" on page 173 or "Tweening shapes" on page 178.

To distribute objects to layers:

1. Select the objects that you want to distribute to layers. The objects can be in a single layer, or in several layers, including noncontiguous layers.
2. Do one of the following:

- Choose Modify > Distribute to Layers.
- Right-click (Windows) or Control-click (Macintosh) one of the selected objects and choose Distribute to Layers from the context menu.

Tweening instances, groups, and type

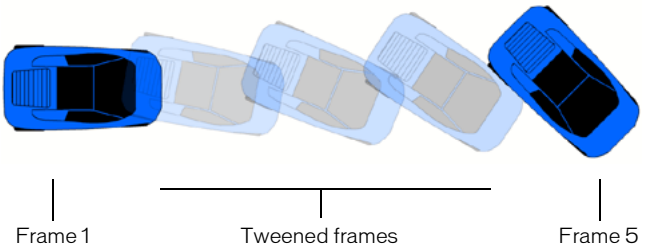
To tween the changes in properties of instances, groups, and type, you use motion tweening. Flash can tween position, size, rotation, and skew of instances, groups, and type. Additionally, Flash can tween the color of instances and type, creating gradual color shifts or making an instance fade in or out. To tween the color of groups or type, you must make them into symbols. See "Creating symbols" on page 151. To animate individual characters in a block of text separately, you place each character in a separate text block; see "Breaking text apart" on page 144.

If you apply a motion tween and then change the number of frames between the two keyframes, or move the group or symbol in either keyframe, Flash automatically tweens the frames again.

You can create a motion tween using one of two methods:

- Create the starting and ending keyframes for the animation and use the Motion Tweening option in the Property inspector.
- Create the first keyframe for the animation, insert the number of frames you want on the Timeline, choose Insert > Create Motion Tween, and move the object to the new location on the Stage. Flash automatically creates the ending keyframe.

When tweening position, you can make the object move along a nonlinear path. See "Tweening motion along a path" on page 176.



The second, third, and fourth frames result from tweening the first and last keyframes.

To create a motion tween using the Motion Tweening option:

1. Click a layer name to make it the current layer, and select an empty keyframe in the layer where you want the animation to start.
2. To create the first frame of the motion tween, do one of the following:

- Create a graphic object with the Pen, Oval, Rectangle, Pencil, or Brush tool, then convert it to a symbol. For more information on converting objects to symbols, see "Creating symbols" on page 151.
- Create an instance, group, or text block on the Stage.
- Drag an instance of a symbol from the Library panel.

3. Create a second keyframe where you want the animation to end, then select the ending frame (immediately to the left of the second keyframe on the Timeline).
4. Do any of the following to modify the instance, group, or text block in the ending frame:

- Move the item to a new position.
- Modify the item's size, rotation, or skew.
- Modify the item's color (instance or text block only).

  To tween the color of elements other than instances or text blocks, use shape tweening. See "Tweening shapes" on page 178.

5. If the Property inspector is not visible, choose Window > Properties.
6. Double-click the ending frame in the Timeline.
7. Select Motion from the Tween pop-up menu in the Property inspector.
8. If you modified the size of the item in step 4, select Scale to tween the size of the selected item.

9 Drag the arrow next to the Easing value or enter a value to adjust the rate of change between tweened frames:

- To begin the motion tween slowly and accelerate the tween toward the end of the animation, drag the slider up or enter a negative value between -1 and -100.
- To begin the motion tween rapidly and decelerate the tween toward the end of the animation, drag the slider down or enter a positive value between 1 and 100.

  By default, the rate of change between tweened frames is constant. Easing creates a more natural appearance of acceleration or deceleration by gradually adjusting the rate of change.

10 To rotate the selected item while tweening, choose an option from the Rotate menu:

- Choose None (the default setting) to prevent rotation.
- Choose Auto to rotate the object once in the direction requiring the least motion.
- Choose Clockwise (CW) or Counterclockwise (CCW) to rotate the object as indicated, and then enter a number to specify the number of rotations.

Note: The rotation in step 9 is in addition to any rotation you applied to the ending frame in step 4.

11 If you're using a motion path, select Orient to Path to orient the baseline of the tweened element to the motion path. (See "Tweening motion along a path" on page 176.)

12 Select the Sync checkbox in the Property inspector to synchronize the animation of graphic symbol instances with the main Timeline.

Note: Modify › Frames › Synchronize Symbols and the Sync checkbox both recalculate the number of frames in a tween to match the number of frames allotted to it in the Timeline.

13 If you're using a motion path, select Snap to attach the tweened element to the motion path by its registration point.

To create a motion tween using the Create Motion Tween command:

1 Select an empty keyframe and draw an object on the Stage, or drag an instance of a symbol from the Library panel.

Note: In order to create a tween, you must have only one item on the layer.

2 Choose Insert > Create Motion Tween.

If you drew an object in step 1, Flash automatically converts the object to a symbol and assigns it the name tween1.

3 Click inside the frame where you want the animation to end, and choose Insert > Frame.

4 Move the object, instance, or type block on the Stage to the desired position. Adjust the size of the element if you want to tween its scale. Adjust the rotation of the element if you want to tween its rotation. Deselect the object when you have completed adjustments.

A keyframe is automatically added to the end of the frame range.

5 Drag the arrow next to the Easing value or enter a value to adjust the rate of change between tweened frames:

- To begin the motion tween slowly and accelerate the tween toward the end of the animation, drag the slider up or enter a value between -1 and -100.
- To begin the motion tween rapidly and decelerate the tween toward the end of the animation, drag the slider down or enter a positive value between 1 and 100.

  By default, the rate of change between tweened frames is constant. Easing creates a more natural appearance of acceleration or deceleration by gradually adjusting the rate of change.

6 To rotate the selected item while tweening, choose an option from the Rotate menu:

- Choose Auto to rotate the object once in the direction requiring the least motion.
- Choose Clockwise (CW) or Counterclockwise (CCW) to rotate the object as indicated, and then enter a number to specify the number of rotations.

  Note: The rotation in step 6 is in addition to any rotation you applied to the ending frame in step 4.

7 If you're using a motion path, select Orient to Path to orient the baseline of the tweened element to the motion path. (See the following section.)

8 Select Synchronize to ensure that the instance loops properly in the main movie.

  Use the Synchronize command if the number of frames in the animation sequence inside the symbol is not an even multiple of the number of frames the graphic instance occupies in the movie.

9 If you're using a motion path, select Snap to attach the tweened element to the motion path by its registration point.

Tweening motion along a path

Motion guide layers let you draw paths along which tweened instances, groups, or text blocks can be animated. You can link multiple layers to a motion guide layer to have multiple objects follow the same path. A normal layer that is linked to a motion guide layer becomes a guided layer.

To create a motion path for a tweened animation:

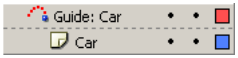
1 Create a motion-tweened animation sequence as described in "Tweening instances, groups, and type" on page 173.

  If you select Orient to Path, the baseline of the tweened element will orient to the motion path. If you select Snap, the registration point of the tweened element will snap to the motion path.

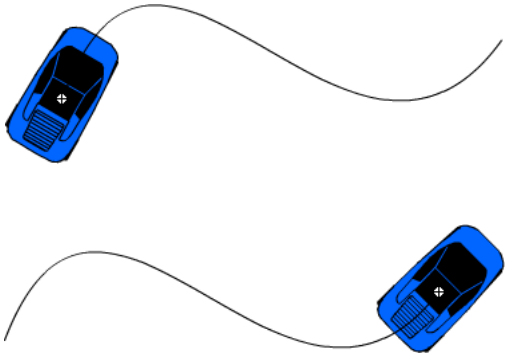
2 Do one of the following:

- Select the layer containing the animation and choose Insert > Motion Guide.
- Right-click (Windows) or Control-click (Macintosh) the layer containing the animation and choose Add Motion Guide from the context menu.

  Flash creates a new layer above the selected layer with a motion guide icon to the left of the layer name.



3 Use the Pen, Pencil, Line, Circle, Rectangle, or Brush tool to draw the desired path.

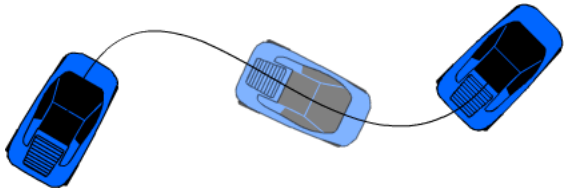


4 Snap the center to the beginning of the line in the first frame, and to the end of the line in the last frame.

Note: For best snapping results, drag the symbol by its registration point.

5 To hide the motion guide layer and the line so that only the object's movement is visible while you work, click in the Eye column on the motion guide layer.

The group or symbol follows the motion path when you play the animation.



To link layers to a motion guide layer, do one of the following:

- Drag an existing layer below the motion guide layer. The layer is indented under the motion guide layer. All objects on this layer automatically snap to the motion path.
- Create a new layer under the motion guide layer. Objects you tween on this layer are automatically tweened along the motion path.
- Select a layer below a motion guide layer. Choose Modify > Layer and select Guided in the Layer Properties dialog box.

To unlink layers from a motion guide layer:

1 Select the layer you want to unlink.

2 Do one of the following:

- Drag the layer above the motion guide layer.
- Choose Modify > Layer and select Normal as the layer type in the Layer Properties dialog box.

Tweening shapes

By tweening shapes, you can create an effect similar to morphing, making one shape appear to change into another shape over time. Flash can also tween the location, size, and color of shapes.

Tweening one shape at a time usually yields the best results. If you tween multiple shapes at one time, all the shapes must be on the same layer.

To apply shape tweening to groups, instances, or bitmap images, you must first break these elements apart. See "Breaking apart groups and objects" on page 133. To apply shape tweening to text, you must break the text apart twice to convert the text to objects. See "Breaking text apart" on page 144.

To control more complex or improbable shape changes, you use shape hints, which control how parts of the original shape move into the new shape. See "Using shape hints" on page 179.

To tween a shape:

1. Click a layer name to make it the current layer, and create or select a keyframe where you want the animation to start.
2. Create or place the artwork for the first frame of the sequence. For best results, the frame should contain only one item (a graphic object or broken-apart group, bitmap, instance, or text block).
3. Select the keyframe in the Timeline.
4. Choose Window > Properties.
5. In the Property inspector, select Shape from the Tween pop-up menu.
6. Drag the arrow next to the Easing value or enter a value to adjust the rate of change between tweened frames:

- To begin the shape tween gradually and accelerate the tween toward the end of the animation, drag the slider down or enter a negative value between -1 and -100.
- To begin the shape tween rapidly and decelerate the tween toward the end of the animation, drag the slider up or enter a positive value between 1 and 100.

  By default, the rate of change between tweened frames is constant. Easing creates a more natural appearance of transformation by gradually adjusting the rate of change.

7. Choose an option for Blend:

- Distributive creates an animation in which the intermediate shapes are smoother and more irregular.
- Angular creates an animation that preserves apparent corners and straight lines in the intermediate shapes.

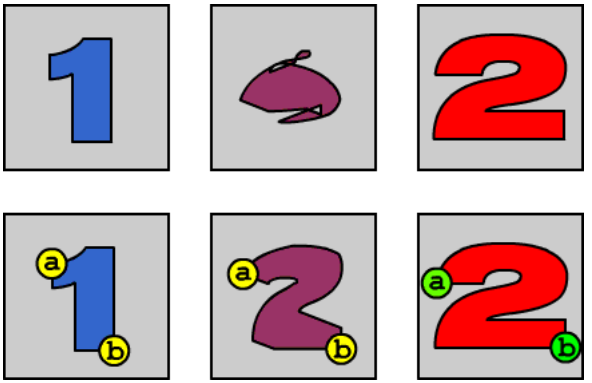
  Note: Angular is appropriate only for blending shapes with sharp corners and straight lines. If the shapes you choose do not have corners, Flash reverts to distributive shape tweening.

8. Create a second keyframe the desired number of frames after the first keyframe.
9. With the second keyframe selected, select the artwork you placed in the first keyframe and do one of the following:

- Modify the shape, color, or position of the artwork.
- Delete the artwork and place new artwork in the second keyframe.

Using shape hints

To control more complex or improbable shape changes, you can use shape hints. Shape hints identify points that should correspond in starting and ending shapes. For example, if you are tweening a drawing of a face as it changes expression, you can use a shape hint to mark each eye. Then, instead of the face becoming an amorphous tangle while the shape change takes place, each eye remains recognizable and changes separately during the shift.



The same shape tween, without (top) and with (bottom) shape hints, respectively.

Shape hints contain letters (*a* through *z*) for identifying which points correspond in the starting and ending shape. You can use up to 26 shape hints.

Shape hints are yellow in a starting keyframe, green in an ending keyframe, and red when not on a curve.

For best results when tweening shapes, follow these guidelines:

- In complex shape tweening, create intermediate shapes and tween them instead of just defining a starting and ending shape.
- Make sure that shape hints are logical. For example, if you're using three shape hints for a triangle, they must be in the same order on the original triangle and the triangle to be tweened. The order cannot be *abc* in the first keyframe and *acb* in the second.
- Shape hints work best if you place them in counterclockwise order beginning at the top left corner of the shape.

To use shape hints:

1. Select the first keyframe in a shape-tweened sequence.
2. Choose Modify > Transform > Add Shape Hint.

   The beginning shape hint appears as a red circle with the letter *a* somewhere on the shape.
3. Move the shape hint to a point that you want to mark.
4. Select the last keyframe in the tweening sequence.

   The ending shape hint appears somewhere on the shape as a green circle with the letter *a*.
5. Move the shape hint to the point in the ending shape that should correspond to the first point you marked.
6. Run the movie again to see how the shape hints change the shape tweening. Move the shape hints to fine-tune the tweening.
7. Repeat this process to add additional shape hints. New hints appear with the letters that follow (*b, c,* and so on).

While working with shape hints, you can also do the following:

- To see all shape hints, choose View > Show Shape Hints. The layer and keyframe that contain shape hints must be current for Show Shape Hints to be available.
- To remove a shape hint, drag it off the Stage.
- To remove all shape hints, choose Modify > Transform > Remove All Hints.

Creating frame-by-frame animations

To create a frame-by-frame animation, you define each frame as a keyframe and create a different image for each frame. Each new keyframe initially contains the same contents as the keyframe preceding it, so you can modify the frames in the animation incrementally.

To create a frame-by-frame animation:

1. Click a layer name to make it the current layer, and select a frame in the layer where you want the animation to start.
2. If the frame isn't already a keyframe, choose Insert > Keyframe to make it one.
3. Create the artwork for the first frame of the sequence.

   You can use the drawing tools, paste graphics from the Clipboard, or import a file.
4. Click the next frame to the right in the same row and choose Insert > Keyframe, or right-click (Windows) or Control-click (Macintosh) and choose Insert Keyframe from the context menu.

   This adds a new keyframe whose contents are the same as those of the first keyframe.
5. Alter the contents of this frame on the Stage to develop the next increment of the animation.
6. To complete your frame-by-frame animation sequence, repeat steps 4 and 5 until you've built the motion you want.
7. To test the animation sequence, choose Control > Play or click the Play button on the Controller.

Editing animation

After you create a frame or a keyframe, you can move it elsewhere in the current layer or to another layer, remove it, and make other changes. Only keyframes are editable. You can view tweened frames, but you can't edit them directly. To edit tweened frames, you change one of the defining keyframes or insert a new keyframe between the beginning and ending keyframes. You can drag items from the Library panel onto the Stage to add the items to the current keyframe.

To display and edit more than one frame at a time, you use onion skinning. See "Onion skinning" on page 182.

To insert frames in the Timeline, do one of the following:

- To insert a new frame, choose Insert > Frame.
- To create a new keyframe, choose Insert > Keyframe, or right-click (Windows) or Control-click (Macintosh) the frame where you want to place a keyframe, and choose Insert Keyframe from the context menu.
- To create a new blank keyframe, choose Insert > Blank Keyframe, or right-click (Windows) or Control-click (Macintosh) the frame where you want to place the keyframe, and choose Insert Blank Keyframe from the context menu.

To delete or modify a frame or keyframe, do one of the following:

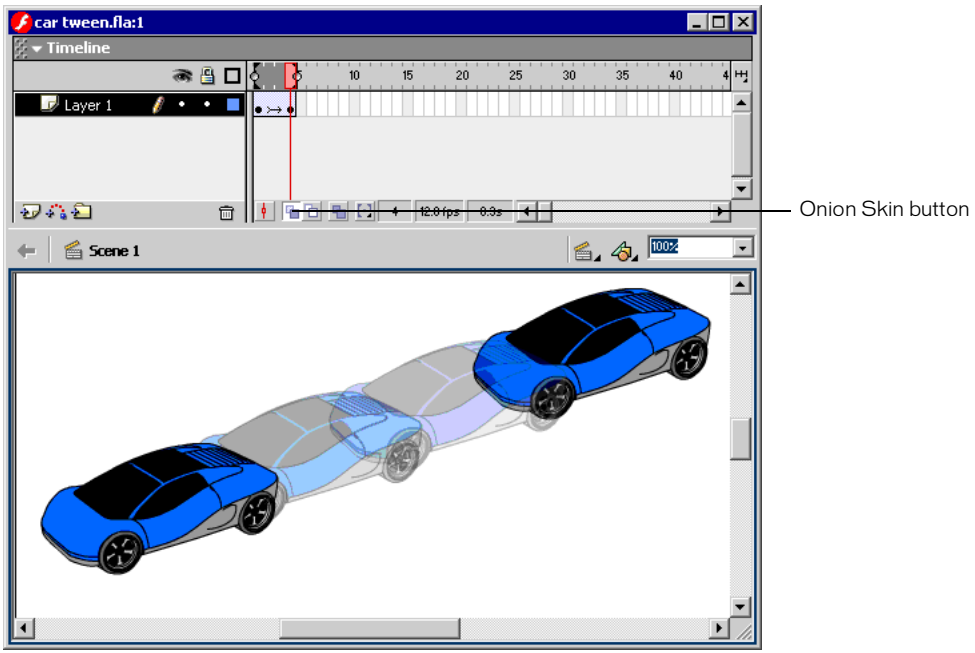
- To delete a frame, keyframe, or frame sequence, select the frame, keyframe, or sequence and choose Insert > Remove Frame, or right-click (Windows) or Control-click (Macintosh) the frame, keyframe, or sequence and choose Remove Frame from the context menu. Surrounding frames remain unchanged.
- To move a keyframe or frame sequence and its contents, select the keyframe or sequence then drag to the desired location.
- To extend the duration of a keyframe, Alt-drag (Windows) or Option-drag (Macintosh) the keyframe to the final frame of the new sequence.
- To copy a keyframe or frame sequence by dragging, select the keyframe or sequence, then Alt-drag (Windows) or Option-drag (Macintosh) to the new location.
- To copy and paste a frame or frame sequence, select the frame or sequence and choose Edit > Copy Frames. Select a frame or sequence that you want to replace, and choose Edit > Paste Frames.
- To convert a keyframe to a frame, select the keyframe and choose Insert > Clear Keyframe, or right-click (Windows) or Control-click (Macintosh) the keyframe and choose Clear Keyframe from the context menu. The cleared keyframe and all frames up to the subsequent keyframe are replaced with the contents of the frame preceding the cleared keyframe.
- To change the length of a tweened sequence, drag the beginning or ending keyframe left or right. To change the length of a frame-by-frame sequence, see "Creating frame-by-frame animations" on page 180.
- To add a library item to the current keyframe, drag the item from the Library panel onto the Stage.
- To reverse an animation sequence, select the appropriate frames in one or more layers and choose Modify > Frames > Reverse. There must be keyframes at the beginning and end of the sequence.

Onion skinning

Normally, Flash displays one frame of the animation sequence at a time on the Stage. To help you position and edit a frame-by-frame animation, you can view two or more frames on the Stage at once. The frame under the playhead appears in full color, while surrounding frames are dimmed, making it appear as if each frame were drawn on a sheet of translucent onion-skin paper and the sheets were stacked on top of each other. Dimmed frames cannot be edited.

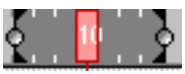
To simultaneously see several frames of an animation on the Stage:

Click the Onion Skin button. All frames between the Start Onion Skin and End Onion Skin markers (in the Timeline header) are superimposed as one frame in the Movie window.



To control onion skinning display, do any of the following:

- To display onion skinned frames as outlines, click the Onion Skin Outlines button.
- To change the position of either onion skin marker, drag its pointer to a new location. (Normally, the onion skin markers move in conjunction with the current frame pointer.)



- To enable editing of all frames between onion skin markers, click the Edit Multiple Frames button. Usually onion skinning lets you edit only the current frame. However, you can display the contents of each frame between the onion skin markers normally, and make each available for editing, regardless of which is the current frame.

Note: Locked layers (those with a padlock icon) aren't displayed when onion skinning is turned on. To avoid a multitude of confusing images, you can lock or hide the layers you don't want onion skinned.

To change the display of onion skin markers:

Click the Modify Onion Markers button and choose an item from the menu:

- Always Show Markers displays the onion skin markers in the Timeline header whether or not onion skinning is on.
- Anchor Onion Marks locks the onion skin markers to their current position in the Timeline header. Normally, the Onion Skin range is relative to the current frame pointer and the Onion Skin markers. By anchoring the Onion Skin markers, you prevent them from moving with the current frame pointer.
- Onion 2 displays two frames on either side of the current frame.
- Onion 5 displays five frames on either side of the current frame.
- Onion All displays all frames on either side of the current frame.

Moving an entire animation

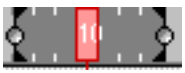
If you need to move an entire animation on the Stage, you must move the graphics in all frames and layers at once to avoid realigning everything.

To move the entire animation to another location on the Stage:

1. Unlock all layers.

   To move everything on one or more layers but nothing on other layers, lock or hide all the layers you don't want to move.

2. Click the Edit Multiple Frames button in the Timeline.
3. Drag the onion skin markers so that they enclose all the frames you want to select, or click Modify Onion Markers and choose Onion All.



4. Choose Edit > Select All.
5. Drag the entire animation to the new location on the Stage.

Using mask layers

For spotlight effects and transitions, you can use a mask layer to create a hole through which underlying layers are visible. A mask item can be a filled shape, a type object, an instance of a graphic symbol, or a movie clip. You can group multiple layers together under a single mask layer to create sophisticated effects.

To create dynamic effects, you can animate a mask layer. For a filled shape used as a mask, you use shape tweening; for a type object, graphic instance, or movie clip, you use motion tweening. When using a movie clip instance as a mask, you can animate the mask along a motion path.

To create a mask layer, you place a mask item on the layer that you want to use as a mask. Instead of having a fill or stroke, the mask item acts as a window that reveals the area of linked layers that lie beneath it. The rest of the mask layer conceals everything except what shows through the mask item. A mask layer can contain only one mask item. You cannot have a mask layer inside a button, and you cannot apply a mask to another mask.

You can also use ActionScript to create a mask layer from a movie clip. A mask layer created with ActionScript can only be applied to another movie clip. See "Using movie clips as masks" under Help > Using Flash.

To create a mask layer:

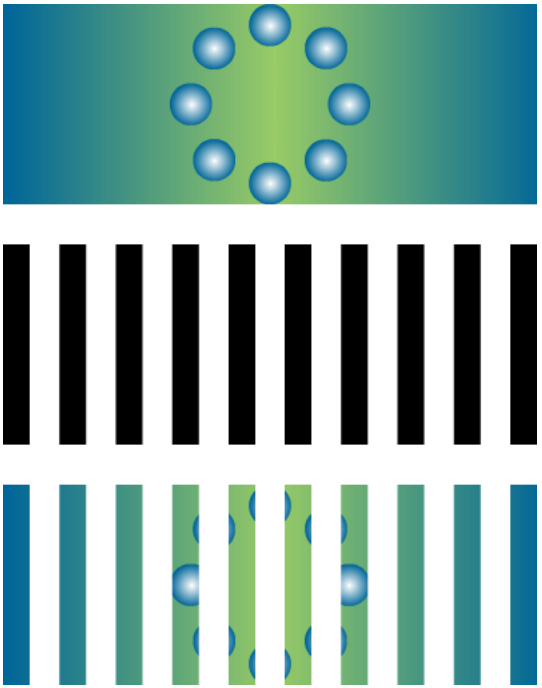
1. Select or create a layer containing the objects that will appear inside the mask.
2. With the layer selected, choose Insert > Layer to create a new layer above it.

   A mask layer always masks the layer immediately below it, so be sure to create the mask layer in the proper place.
3. Place a filled shape, type, or an instance of a symbol on the mask layer.

   Flash ignores bitmaps, gradients, transparency, colors, and line styles in a mask layer. Any filled area will be completely transparent in the mask; any nonfilled area will be opaque.
4. Right-click (Windows) or Control-click (Macintosh) the mask layer's name in the Timeline, and choose Mask from the context menu.

   The layer is converted to a mask layer, indicated by a mask layer icon. The layer immediately below it is linked to the mask layer, and its contents show through the filled area on the mask. The masked layer name is indented, and its icon changes to a masked layer icon.

   To display the mask effect in Flash, lock the mask layer and the masked layer.



A masked layer; the filled shape that will be transparent in the mask; and the final mask effect

To mask additional layers after creating a mask layer, do one of the following:

- Drag an existing layer directly below the mask layer.
- Create a new layer anywhere below the mask layer.
- Choose Modify > Layer and select Masked in the Layer Properties dialog box.

To unlink layers from a mask layer:

1. Select the layer you want to unlink.
2. Do one of the following:

- Drag the layer above the mask layer.
- Choose Modify > Layer and select Normal.

To animate a filled shape, type object, or graphic symbol instance on a mask layer:

1. Select the mask layer in the Timeline.
2. Click in the Lock column to unlock the mask layer.
3. Do one of the following:

- If the mask object is a filled shape, apply shape tweening to the object as described in "Tweening shapes" on page 178.
- If the mask object is a type object or graphic symbol instance, apply motion tweening to the object as described in "Tweening instances, groups, and type" on page 173.

4. When you've completed the animation operation, click in the Lock column for the mask layer to lock the layer again.

To animate a movie clip on a mask layer:

1. Select the mask layer in the Timeline.
2. Double-click the movie clip on the Stage to edit the clip in place and to display the movie clip's Timeline.
3. Apply motion tweening to the movie clip as described in "Tweening instances, groups, and type" on page 173. To animate the movie clip on a motion path, see "Tweening motion along a path" on page 176.
4. When you've completed the animation procedure, click the Back button in the Edit in Place window to return to movie-editing mode.
5. Click in the Lock column for the mask layer to lock the layer again.

CHAPTER 11
Writing Scripts with ActionScript

ActionScript, the scripting language of Macromedia Flash MX, lets you add interactivity to a movie. ActionScript provides elements, such as actions, operators, and objects, that you put together in scripts that tell your movie what to do; you set up your movie so that events, such as button clicks and keypresses, trigger these scripts. For example, you can use ActionScript to create navigation buttons for your movie.

In Flash, you use the Actions panel to write scripts with ActionScript. Using the panel in normal editing mode, you build scripts by choosing options from menus and lists. Using the panel in expert editing mode, you enter text directly into the Script pane. In both modes, code hints help you complete actions and insert properties and events. Once you have a script, you can attach it to a button, movie clip, or frame to create the interactivity you need.

You can write simple scripts without a full understanding of ActionScript. To begin working with ActionScript right away, complete the ActionScript tutorial in Help > Tutorials > Introduction to ActionScript. To learn more about the ActionScript language, see Chapter 12, "Understanding the ActionScript Language," on page 203.

Using the Actions panel

To add an action to a Flash document, you must attach it to a button or movie clip, or to a frame in the Timeline. The Actions panel allows you to select, drag and drop, rearrange, and delete actions.

You can use the Actions panel in two different editing modes: normal and expert. In normal mode you write actions by filling in parameter text boxes. In expert mode you write and edit actions directly in a Script pane, much like writing scripts with a text editor.

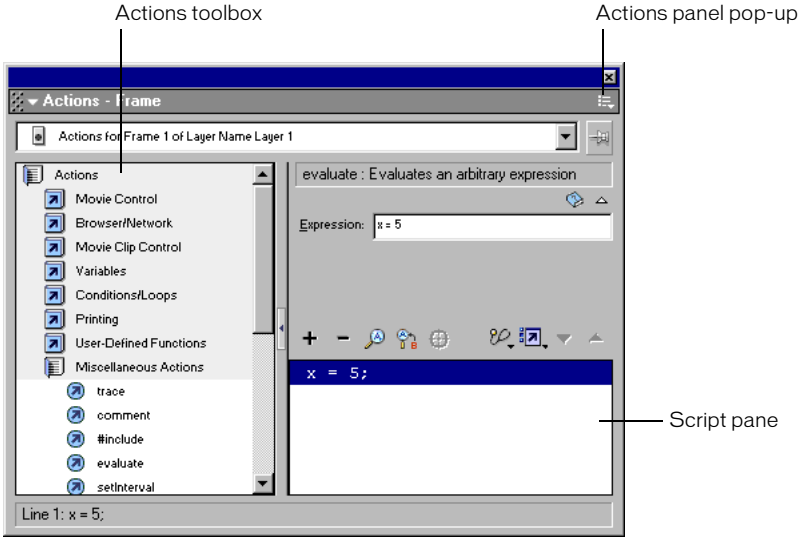
To display the Actions panel, do one of the following:

- Choose Window > Actions.
- Press F2.

To activate the Actions panel:

Select an instance of a button, movie clip, or frame.

The Actions panel title changes to reflect the selection.



To navigate through the Actions toolbox, do the following:

- To select the first item in the Actions toolbox, press Home.
- To select the last item in the Actions toolbox, press End.
- To select the previous item in the Actions toolbox, press the Up Arrow or Left Arrow key.
- To select the next item in the Actions toolbox, press the Down Arrow or Right Arrow key.
- To expand or collapse a folder, press Enter or Spacebar.
- To insert an item into a script, press Enter or Spacebar. (This is the same as selecting Add to Script from the item's context menu.)
- To scroll up a page of items, press Page Up.
- To scroll down a page of items, press Page Down.
- To search for an Actions toolbox item by initial character, type the character. This search is not case-sensitive. You can type the character multiple times to cycle through all the items that start with that character.

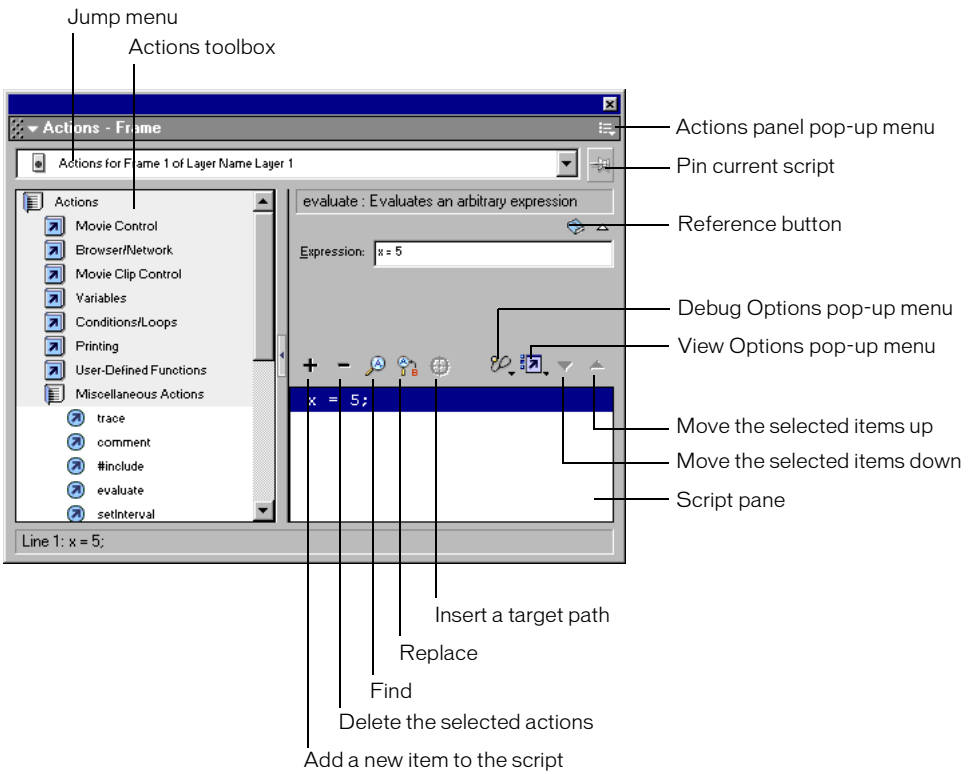
Working in normal mode

In normal mode you build scripts by selecting items from the Actions toolbox, the list on the left side of the Actions panel. (You can also select actions from the Add (+) button pop-up menu.) The Actions toolbox separates items into categories such as Actions, Properties, and Objects, and also provides an Index category that lists all items alphabetically. When you click an item once, its description appears at the upper right of the panel. When you double-click an item, it appears on the right side of the panel, in the Script pane.

In normal mode you can add, delete, or change the order of statements in the Script pane; you can also enter parameters for actions in text boxes above the Script pane. The Actions panel also lets you find and replace text, view script line numbers, and "pin" a script—that is, keep a script in the Script pane when you click away from the object or frame. You can also use the jump menu to go to any script on any object in the current frame.

To use the Actions panel to insert target paths, see Chapter 13, "Working with Movie Clips and Buttons," on page 245.

To work with the Actions panel's debugging breakpoint features, see "Setting and removing breakpoints" on page 359.



The Actions panel in normal mode

To display the Actions panel in normal mode:

1 Select Windows > Actions.

2 Do one of the following:

- Click the arrow in the upper right corner of the Actions panel to display the pop-up menu, and choose Normal Mode.
- Click anywhere in the Actions panel. Then press Control+Shift+N (Windows) or Command+Shift+N (Macintosh).

To view a description of an action, do one of the following:

- Click a category in the Actions toolbox to display the actions in that category, and click an action.
- Select a line of code in the Script pane.

The description appears at the upper right of the Actions panel.

To add an action to the Script pane, do one of the following:

- Click a category in the Actions toolbox to display the actions in that category. Then either double-click an action, drag it to the Script pane, or right-click (Windows) or Control-click (Macintosh) and select Add to Script.
- Click the Add (+) button and select an action from the pop-up menu.
- Press Escape and a shortcut key. For example, Escape+st adds a `stop` action. (To view a list of shortcut keys, select View Esc Shortcut Keys in the Actions panel pop-up menu; select this option again to hide the list.)

To delete an action:

1 Select a statement in the Script pane.

2 Click the Delete (-) button or press the Delete key.

To move a statement up or down in the Script pane:

Select a statement in the Script pane, then do one of the following:

- Click the Up or Down Arrow button.
- Select the statement and drag it up or down.

To work with parameters:

1 Add an action to or select a statement in the Script pane.

2 Enter values in the parameter text boxes above the Script pane.

For more information about code hints that can help you insert parameters, see “Using code hints” on page 197.

To search for text in a script, do one of the following:

- To go to a specific line in a script, choose GoTo Line from the Actions panel pop-up menu or press Control+G (Windows) or Command+G (Macintosh); then enter the line number.
- To find text, click the Find button above the Script pane, choose Find from the Actions panel pop-up menu, or press Control+F (Windows) or Command+F (Macintosh). Enter the text you want to find in the dialog box that appears.
- To find text again, press F3 or choose Find Again from the Actions panel pop-up menu.
- To replace text, click the Replace button above the Script pane, choose Replace from the Actions panel pop-up menu, or press Control+H (Windows) or Command-H (Macintosh). Enter the text you want to find and the text you want to replace it with in the dialog box that appears.

  In expert mode, Replace scans the entire body of text in a script. In normal mode, Replace searches and replaces text only in the parameter box of each action. For example, in normal mode you cannot replace all `gotoAndPlay` actions with `gotoAndStop`.

These find and replace features search the current Script pane. To search through text in every script in a movie, use the Movie Explorer (see "Using the Movie Explorer" on page 40).

To navigate between scripts:

Use the jump menu at the top of the Actions panel and choose a script from the list.

To pin a script to the Actions panel:

Click the Script Pin button.

The Actions panel displays the script in the Script pane even when you click away from the object or frame.

To resize the Actions toolbox or Script pane, do one of the following:

- Drag the vertical splitter bar that appears between the Actions toolbox and Script pane.
- Double-click the splitter bar to collapse the Actions toolbox; double-click the bar again to redisplay the Actions toolbox.
- Click the arrow button on the splitter bar to expand or collapse the Actions toolbox.

  When the Actions toolbox is hidden, you can still use the Add (+) button to access its items.

To view line numbers in the Script pane, do one of the following:

- Select View Line Numbers from the View Options pop-up menu above the Script pane.
- Select View Line Numbers from the Actions panel pop-up menu.
- Press Control+Shift+L (Windows) or Command+Shift+L (Macintosh).

To print actions:

1. From the Actions panel pop-up menu, choose Print.

   The Print dialog box appears.
2. Choose options and click Print.

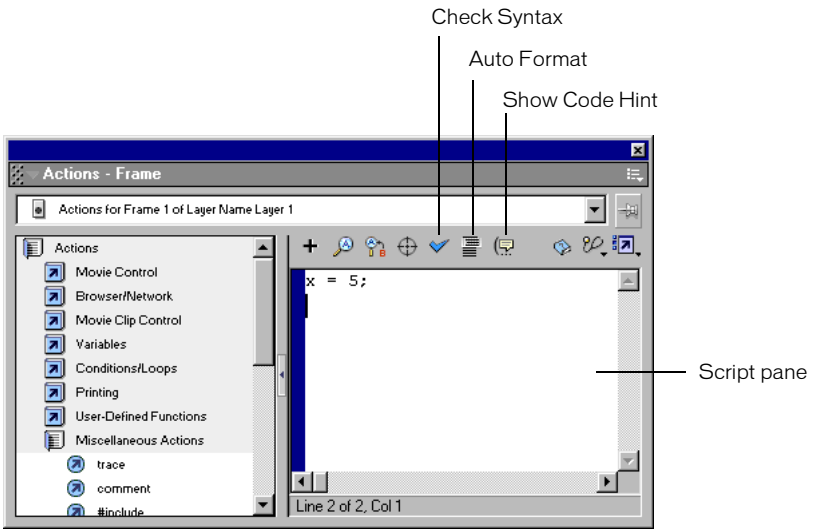
Because the printed file won't include information about its originating Flash file, it's a good idea to include this information in a `comment` action in the script.

Working in expert mode

In expert mode you create scripts by entering ActionScript directly into the Script pane on the right side of the Actions panel. You edit actions, enter parameters for actions, or delete actions directly in the Script pane, much as you would create a script in a text editor. You can also use the Actions toolbox (the left side of the Actions panel) and the Add (+) button to add actions to the Script pane, but the parameter text boxes don't appear. You cannot move statements with the Up Arrow and Down Arrow buttons or delete statements with the Delete (-) button.

Like normal mode, expert mode lets you use the buttons above the Script pane to find and replace text, set and remove debugging breakpoints, view line numbers, and insert target paths; it also allows you to use the jump menu and the Script Pin button.

In expert mode you can also check syntax for errors, automatically format code, and use code hints to help you complete syntax. In addition, the punctuation balance feature helps you pair parentheses, braces, or brackets.



The Actions panel in expert mode

To display the Actions panel in expert mode:

1 Select Windows > Actions.

2 Do one of the following:

- Click the arrow in the upper right corner of the Actions panel to display the pop-up menu, and choose Expert Mode.
- Click anywhere in the Actions panel. Then press Control+Shift+E (Windows) or Command+Shift+E (Macintosh).

To check syntax, do one of the following:

- Click the Check Syntax button.
- Choose Check Syntax from the Actions panel pop-up menu (at the upper right of the panel). Syntax errors are listed in Output window.
- Press Control+T (Windows) or Command+T (Macintosh).

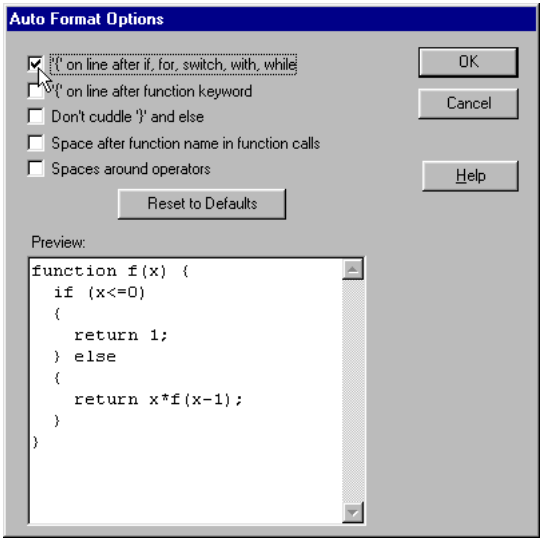
To format code using the Flash formatting style, do one of the following:

- Click the Auto Format button.
- Choose Auto Format from the Actions panel pop-up menu.
- Press Control+Shift+F (Windows) or Command+Shift+F (Macintosh).

To set options for the Flash formatting style, do the following:

1 Choose Auto Format Options from the Actions panel pop-up menu.

The Auto Format Options dialog box appears.



2 Select any of the check boxes. To see the effect of each selection, look in the Preview pane.

To use automatic indentation:

Automatic indentation is turned on by default. To turn it off, deselect Automatic Indentation in ActionScript Editor preferences.

When automatic indentation is turned on, the text you type after (or { is automatically indented according to the Tab Size setting in ActionScript Editor preferences. To indent another line in the Script pane, select the line and press Tab. To remove the indent, press Shift+Tab.

To use punctuation balance, do the following:

1. Click between braces, brackets, or parentheses in your script.
2. Press Control+' (Windows) or Command+' (Macintosh) to highlight the text between braces, brackets, or parentheses.

   The highlighting helps you check whether opening punctuation has correct corresponding closing punctuation.

To display code hints:

Click the Show Code Hint button.

For more information about code hints, see "Using code hints" on page 197. To change Flash preferences so that code hints always appear, see "Setting Actions panel preferences" on page 196.

Switching between editing modes

While working in the Actions panel, you can switch between normal mode and expert mode at any time. When you switch modes, Flash maintains your script's formatting unless you change the script. For example, if you write a script in expert mode with your own style of indentation and switch to normal mode to view it, but make no changes, the formatting does not change. If, however, you modify the script in normal mode, Flash removes your custom indentation and formats the script using the Auto Format Options settings (see "Working in expert mode" on page 192). Flash warns you before changing any formatting.

To switch editing modes, do one of the following:

- Choose Normal Mode or Expert Mode from the Actions panel pop-up menu (at the upper right of the panel) or from the View Options pop-up menu above the Script pane. A check mark indicates the selected mode.
- To switch to normal mode, press Control+Shift+N (Windows) or Command+Shift+N (Macintosh).
- To switch to expert mode, press Control+Shift+E (Windows) or Command+Shift+E (Macintosh).

The Actions panel remains in the selected mode until you choose the other mode, even if you select a script on a different button, movie clip, or frame.

You cannot use normal mode to view an expert mode script that contains errors. If you try, you'll receive the message "This script contains syntax errors. It must be edited in expert mode."

You can use normal mode to view an expert mode script that uses ActionScript elements that are not supported by the current publish settings. However, if you export such a script, you'll receive a warning message.

Using the Reference panel

You can use the Reference panel to view detailed descriptions of the actions listed in the Actions toolbox. The panel also displays sample code, which you can copy and paste into the Script pane of the Actions panel. You can also adjust the font size and print the contents of Reference panel.

To view a description and sample code for an action, do one of the following:

- Click an action in the Actions toolbox, then click the Reference button.
- Choose Window > Reference.
- Press Control+Alt+Shift+F1 (Windows) or Command+Alt+Shift+F1 (Macintosh).
- In normal mode, select an action in the Script pane and click the Reference button at the upper right above the Script pane. <<Localization: we deleted fifth bullet, which was a leftover comment--IMD>>

To copy and paste sample code:

1. Highlight the code and right-click (Windows) or Control-click (Macintosh), then select Copy from the context menu.
2. In the Script pane, right-click (Windows) or Control-click (Macintosh) and select Paste from the context menu or from the pop-up menu in the upper right.

To change the font size:

Select Large, Medium, or Small Font from the pop-up menu in the upper right of the Reference panel.

To print the contents of the Reference panel:

Select Print from either the context menu or the pop-up menu in the upper right of the Reference panel.

Using an external text editor

You can use a text editor to write and edit ActionScript outside the Actions panel. You can export actions from the Actions panel to a text file, import a text file into the Actions panel, or use the `include` action to add an external script file at runtime.

For more information about using the ActionScript language, see Chapter 12, “Understanding the ActionScript Language,” on page 203.

To export actions as a text file:

1. From the Actions panel pop-up menu (at the upper right of the panel), choose Export as File or press Control+Shift+X (Windows) or Command+Shift+X (Macintosh).
2. Choose a location where the file will be saved, and click Save.

You can then edit the file in an external text editor.

To import a text file containing ActionScript:

1. From the Actions panel pop-up menu, choose the Import from File command or press Control+Shift+I (Windows) or Command+Shift+I (Macintosh).
2. Select a text file containing ActionScript, and click Open.

Scripts with syntax errors can be imported only in expert mode. In normal mode, you’ll receive an error message.

To add an external script to a script within Flash when the movie is exported:

1. Click in the Script pane (at the right of the Actions panel) to place the insertion point where you want the external script to be included.
2. In the Actions toolbox (at the left of the Actions panel), select the Actions category; then select the Miscellaneous Actions category.
3. Double-click the `include` action to add it to the Script pane.
4. Enter the path to the external file in the Path box.

   The path should be relative to the FLA file. For example, suppose your FLA file is myDoc.fla and your external script is called externalfile.as. If myDoc.fla and externalfile.as are in the same folder, the path is externalfile.as. If externalfile.as is in a subfolder called Scripts, the path is scripts/externalfile.as. The text of the external script replaces the `include` action when the document is exported as a Flash movie (SWF) file.

About syntax highlighting

In ActionScript, as in any language, syntax is the way elements are put together to create meaning. If you use incorrect ActionScript syntax, your scripts will not work. (For more information, see "Using ActionScript syntax" on page 213.)

In normal mode, syntax errors are red in the Script pane. If you move the mouse pointer over an action with incorrect syntax, a tooltip displays the associated error message; if you select the red highlighted action, the error message appears in the status bar at the bottom of the Actions panel.

In both expert and normal mode, ActionScript export version incompatibilities are yellow in the Script pane. For example, if the Flash Player export version is set to Flash 5, ActionScript that is supported only by Flash Player 6 is yellow. For information on setting the Flash Player export version, see Chapter 12, "Understanding the ActionScript Language," on page 203

For a complete list of error messages, see Appendix D, "Error Messages," on page 411.

Setting Actions panel preferences

To set preferences for the Actions panel, you use the ActionScript Editor section of Flash preferences. You can change settings such as indentation, code hints, font, and syntax coloring, or restore the settings to their defaults.

To set Actions panel preferences:

1. Do one of the following:

- Choose Preferences from the Actions panel pop-up menu (at the upper right of the Actions panel).
- Choose Edit > Preferences and click the ActionScript Editor tab.

2. Set any of the following preferences:

- For Editing Options, select Automatic Indentation to automatically indent ActionScript in the Script pane in expert mode, and enter an integer in the Tab Size box to set an indentation tab size for expert mode (the default is 4). Select Code Hints to turn on syntax, method, and event completion tips in both expert and normal mode. Move the Delay slider to set the amount of time Flash waits before displaying a code hint (the default is 0).
- For Text, select a font or size from the pop-up menu to change the appearance of text in the Script pane.

- For Syntax Coloring, choose a color for the Script pane's foreground and background and for keywords (for example, `new`, `if`, `while`, and `on`), built-in identifiers (for example `play`, `stop`, and `gotoAndPlay`), comments, and strings. For more information about these ActionScript language elements, see Chapter 12, "Understanding the ActionScript Language," on page 203.

To restore the default ActionScript Editor preferences:

Click the Reset to Defaults button.

Using code hints

When you work in the Actions panel, Flash can detect what action you are entering and display a *code hint*—a tooltip containing the complete syntax for that action or a pop-up menu listing possible method or property names. In expert mode, code hints appear for parameters, properties, and events when you enter certain characters in the Script pane. In normal mode, code hints appear for parameters and properties (but not events) in the parameter text boxes when the Expression box is selected.

Code hints are enabled by default. By setting preferences, you can disable code hints or determine how quickly they appear. (See "Setting Actions panel preferences" on page 196.) When code hints are disabled in preferences, you can turn them on manually.

To enable automatic code hints:

1. Choose Preferences from the Actions panel pop-up menu (at the upper right of the panel).
2. On the ActionScript Editor tab, select Code Hints.

To enable manual code hints in expert mode, do one of the following:

- Click the Show Code Hint button above the Script pane (at the right side of the Actions panel).
- From the Actions panel pop-up menu (at the upper right of the panel), choose Show Code Hints.
- Press Control+Spacebar (Windows) or Command+Spacebar (Macintosh).

To work with tooltip-style code hints:

1. Type an open parenthesis [`(`] after an action name.<<Localization: we changed numbering here and indentation in next sentence--IMD>>

   The code hint appears.
2. Enter a value for the parameter. If there is more than one parameter, separate the values with commas.
3. To dismiss the code hint, do one of the following:

- Type a closing parenthesis [`)`].
- Click outside the statement.
- Press Escape.

To work with menu-style code hints:

1. Display the code hint by doing one of the following:

- Type a dot after the suffix of an object name.
- Type an open parenthesis [`(`] after an event handler name.

2 To navigate through the code hint, use the Up and Down Arrow keys.

3 To select an item in the menu, press Return or Tab, or double-click the item.

4 To dismiss the code hint, do one of the following:

- Choose one of the menu items.
- Click outside the statement.
- Type a closing parenthesis [)] if you've already typed an open parenthesis.
- Press Escape.

Many ActionScript objects require you to create a new instance of the object in order to use its methods and properties. For example, in the code `myMovieClip.gotoAndPlay(3)`, the `gotoAndPlay` method tells the instance `myMovieClip` to go to a specific frame and begin playing the movie clip. The Actions panel doesn't know that the instance `myMovieClip` is of the object class MovieClip, and therefore doesn't know which code hints to display.

If you want the Actions panel to display code hints for object instances, you must add a special class suffix to each instance name. For example, to display code hints for the class MovieClip, you must name all MovieClip objects with the suffix `_mc`, as in the following examples:

```
Circle_mc.gotoAndPlay(1);
Sqaure_mc.stop();
Block_mc.duplicateMovieClip("NewBlock_mc", 100);
```

The following table shows the suffixes and their corresponding object classes:

Suffix	Object class
`_mc`	MovieClip
`_array`	Array
`_str`	String
`_btn<<Loc: this is a change--IMD>>`	Button
`_txt`	TextField
`_fmt`	TextFormat
`_date`	Date
`_sound`	Sound
`_xml`	XML
`_xmlsocket`	XMLSocket
`_color`	Color
`_camera`	Camera
`_mic`	Microphone
`_stream`	NetStream
`_connection`	NetConnection
`_so`	SharedObject
`_video`	Video

You can also use ActionScript comments to specify an object's class for code hinting. The following example tells ActionScript that the class of the instance `theObject` is Object, and so on. If you were to enter the code `mc` after these comments, a code hint would display the list of MovieClip methods and properties; if you were to enter the code `theArray`, a code hint would display a list of Array methods and properties.

```
// Object theObject;
// Array theArray;
// MovieClip mc;
```

For more information about object classes in ActionScript, see "About object-oriented scripting" on page 205.

Assigning actions to a frame

To make a movie perform an action when the playhead reaches a frame in the Timeline, you assign a frame action. For example, to create a loop in the Timeline between frames 20 and 10, you would add an action to frame 20 that sends the playhead to frame 10, as shown here:

```
gotoAndPlay (10);
```

Some actions are commonly assigned to the first frame of a movie—for example, actions that define functions and set variables that create the initial state of your movie. In general, you can assign to the first frame any action you want to execute right as the movie starts. For more information about the order in which actions are executed, see "Controlling when ActionScript runs" on page 207.

It's a good idea to place all frame actions in a layer named Actions; that way, you can always find the actions in the Timeline. Once you've assigned an action, test the movie to see whether it works. The following instructions describe how to assign frame actions using the Actions panel in normal mode.

To assign an action to a frame:

1. Select a keyframe in the Timeline, and choose Window > Actions or press F2.

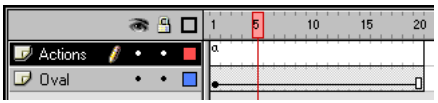
   If a selected frame is not a keyframe, the action will be assigned to the previous keyframe.

2. To assign an action, do one of the following:

- Click a folder in the Actions toolbox (at the left of the Actions panel) to open it. Double-click an action to add it to the Script pane (the right side of the panel).
- Drag an action from the Actions toolbox to the Script pane.
- Click the Add (+) button and choose an action from the pop-up menu.
- Use the keyboard shortcut listed next to an action in the Add (+) button pop-up menu.

3. Enter parameters for the action in the text boxes as needed.
4. To assign additional actions, repeat steps 2 and 3.

Frames with actions display a small *a* in the Timeline.



To test a frame action:

Choose Control > Test Movie.

For more information on using simple actions, see “Controlling movie playback” on page 267. For more information on creating interactivity, see “Creating complex interactivity” on page 271.

Assigning actions to a button

To make a movie perform an action when a button is clicked or rolled over, you can assign an action to the button. You must assign actions to an instance of the button; other instances of the symbol aren't affected.

When you assign an action to a button, you must nest the action inside an `on` handler and specify the mouse or keyboard events that trigger the action. When you assign an action to a button in normal mode, the `on` handler is automatically inserted and you can choose an event from a list. (For more information about event handlers, see “Controlling when ActionScript runs” on page 207.)

You can also use events of the ActionScript Button object to execute scripts when a button event occurs. For more information, see “Handling events with ActionScript” on page 260.

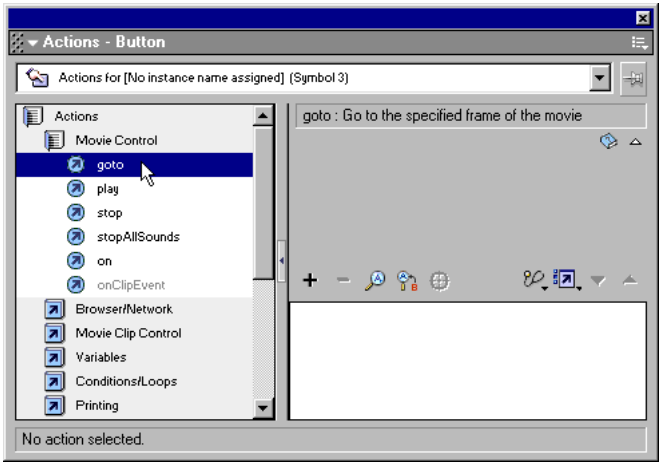
The following instructions describe how to assign actions to a button using the Actions panel in normal mode. Once you've assigned an action, you test the movie to see whether it works.

To assign an action to a button:

1. Select a button and choose Window > Actions to open the Actions panel if it isn't already open. Alternatively, you can select the button or movie clip instance from the jump menu in the Actions panel.
2. To assign an action, do one of the following:

- Click a folder in the Actions toolbox (at the left side of the panel). Double-click an action to add it to the Script pane (at the right side of the panel).
- Drag an action from the Actions toolbox to the Script pane.
- Click the Add (+) button and choose an action from the pop-up menu.

- Use the keyboard shortcut listed next to an action in the Add (+) button pop-up menu.



Selecting an action from the Actions toolbox in normal mode

3 In the parameter text boxes at the top of the panel, enter parameters for the action as needed.

Parameters vary depending on the action you choose. For detailed information on the required parameters for each action, see the online ActionScript Dictionary in the Help menu. To insert a target path for a movie clip into a parameter text box, click the Insert Target Path button above the Script pane. For more information, see Chapter 13, "Working with Movie Clips and Buttons," on page 245.

4 Repeat steps 2 and 3 to assign additional actions as necessary.

To test a button action:

Choose Control > Test Movie.

For more information on using simple actions, see "Controlling movie playback" on page 267. For more information on creating interactivity, see "Creating complex interactivity" on page 271.

Assigning actions to a movie clip

By assigning an action to a movie clip, you can make a movie perform an action when the movie clip is loaded or receives data. You must assign actions to an instance of the movie clip; other instances of the symbol aren't affected.

When you assign an action to a movie clip, you must nest the action inside an `onClipEvent` handler and specify the clip event that triggers the action. When you assign an action to a movie clip in normal mode, the `onClipEvent` handler is automatically inserted. You can choose an event from a list. (For more information about event handlers, see "Controlling when ActionScript runs" on page 207.)

You can also use events of the ActionScript MovieClip object and Button object to execute scripts. For more information on MovieClip and Button events, see "Handling events with ActionScript" on page 260.

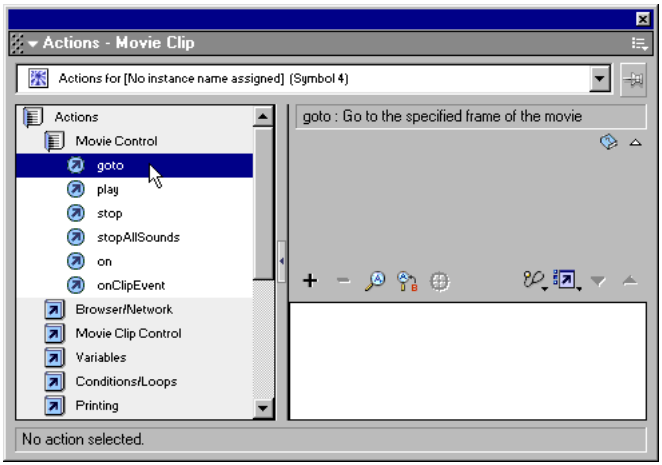
The following instructions describe how to assign actions to movie clips using the Actions panel in normal mode. Once you've assigned an action, you test the movie to see whether it works.

To assign an action to a movie clip:

1 Select a movie clip instance and choose Window > Actions.

2 To assign an action, do one of the following:

- Click a folder in the Actions toolbox (at the left side of the panel). Double-click an action to add it to the Script pane (at the right side of the panel).
- Drag an action from the Actions toolbox to the Script pane.
- Click the Add (+) button and choose an action from the pop-up menu.
- Use the keyboard shortcut listed next to an action in the Add (+) button pop-up menu.



Selecting an action from the Actions toolbox in normal mode

3 In the parameter boxes at the top of the panel, select parameters for the action as needed.

Parameters vary depending on the action you choose. For detailed information on the required parameters for each action, see the online ActionScript Dictionary in the Help menu.To insert a target path for a movie clip into a parameter text box, click the Insert Target Path button above the Script pane. For more information, see Chapter 13, "Working with Movie Clips and Buttons," on page 245.

4 Repeat steps 2 and 3 to assign additional actions as necessary.

To test a movie clip action:

Choose Control > Test Movie.

For more information on using simple actions, see "Controlling movie playback" on page 267. For more information on creating interactivity, see "Creating complex interactivity" on page 271.

CHAPTER 12
Understanding the ActionScript Language

ActionScript, the scripting language of Macromedia Flash MX, allows you to create a movie that behaves exactly as you want. You don't need to understand every ActionScript element to begin scripting; if you have a clear goal, you can start building scripts with simple actions. You can incorporate new elements of the language as you learn them to accomplish more complicated tasks.

Like other scripting languages, ActionScript follows its own rules of syntax, reserves keywords, provides operators, and allows you to use variables to store and retrieve information. ActionScript includes built-in objects and functions and allows you to create your own objects and functions.

The ActionScript syntax and style closely resemble that of JavaScript. Flash MX understands ActionScript written in any previous version of Flash.

This chapter introduces you to ActionScript as an object-oriented scripting language and provides an overview of ActionScript terms and basic programming concepts such as functions, variables, statements, operators, conditionals, and loops. It also deconstructs a sample script so that you can begin to understand ActionScript syntax. The online ActionScript Dictionary contains a detailed entry for every ActionScript element.

To begin scripting with ActionScript right away, complete the ActionScript tutorial (Help > Tutorials > Introduction to ActionScript).

Differences between ActionScript and JavaScript

ActionScript is similar to the core JavaScript programming language. You don't need to know JavaScript to use and learn ActionScript; however, if you know JavaScript, ActionScript will appear familiar to you.

This manual does not attempt to teach programming in general. There are many resources available that provide more information about general programming concepts and the JavaScript language.

- The European Computers Manufacturers Association (ECMA) document ECMA-262 is derived from JavaScript and serves as the international standard for the JavaScript language. ActionScript is based on the ECMA-262 specification, which is available from www.ecma.ch.
- Netscape DevEdge Online has a JavaScript Developer Central site (http://developer.netscape.com/tech/javascript/index.html) that contains documentation and articles useful for understanding ActionScript. The most valuable resource is the *Core JavaScript Guide.*

Some of the differences between ActionScript and JavaScript are as follows:

- ActionScript does not support browser-specific objects such as Document, Window, and Anchor.
- ActionScript does not completely support all of the JavaScript built-in objects.
- ActionScript supports syntax constructs that are not permitted in JavaScript (for example, the `tellTarget` and `ifFrameLoaded` actions and slash syntax). However, the use of these syntax constructs is not recommended; instead, use ActionScript elements that are like those in JavaScript (for example, `with`, `_framesloaded`, and dot syntax).
- ActionScript does not support some JavaScript syntax constructs, such as `try`, `catch`, `throw`, and statement labels.
- ActionScript does not support the JavaScript `Function` constructor.
- In ActionScript, the `eval` action can only perform variable references.
- In JavaScript, `toString` of `undefined` is `undefined`. In Flash 5 and Flash MX, for Flash 4 compatibility, `toString` of `undefined` is `""`.
- In JavaScript, evaluating `undefined` in a numeric context results in `NaN`. In Flash 5 and Flash MX, for Flash 4 compatibility, evaluating `undefined` results in 0.
- In JavaScript, when a string is evaluated in a Boolean context and the string has a nonzero length, the result is `true`; if the string doesn't have a nonzero length, the result is `false`. In ActionScript, the string is converted to a number. If the number is nonzero, the result is `true`; otherwise, the result is `false`.

About scripting in ActionScript

You can start writing simple scripts without knowing much about ActionScript. All you need is a goal; then it's just a matter of choosing the right actions. The best way to learn ActionScript is to create a script. You can use the Actions panel to guide you through setting up simple scripts. (See "Writing Scripts with ActionScript" under Help > Using Flash.) Once you're comfortable adding basic actions such as `play` and `stop` to your document, you can begin to learn more about the language. To use the full power of ActionScript, it is important to understand how the language works: the concepts, elements, and rules that the language uses to organize information and create interactive movies.

This section explains the ActionScript workflow, the fundamental concepts of object-oriented scripting, Flash objects, and script flow. It also describes where scripts reside in a Flash movie.

About planning and debugging scripts

Before you begin writing scripts, formulate your goal and understand what you want to achieve. This is good practice whether you want to achieve something simple such as a button that opens a new Web page, or something complex such as an entire Flash Web site. Planning your scripts is as important as developing storyboards for your work. Start by writing out what you want to happen in the movie, as in this example:

- I want to create my whole site using Flash.
- Site visitors will be asked for their name, which will be reused in messages throughout the site.
- The site will have a draggable navigation bar with buttons that link to each section of the site.
- When a navigation button is clicked, the new section will fade in at the center of the Stage.
- One scene will have a contact form with the user's name already filled in.

When you know what you want, you can use the ActionScript that you need to accomplish the tasks.

Getting scripts to work the way you want takes time—often more than one cycle of writing, testing, and debugging. The best approach is to start simple and test your work frequently. When you get one section of a script working, choose Save As to save a version of the file (for example, myDoc01.fla) and start writing the next section. This approach will help you identify trouble spots efficiently and ensure that your ActionScript is solid as you begin to write more complex scripts.

For more information, see "Testing a movie" under Help > Using Flash.

About object-oriented scripting

In object-oriented scripting, information is organized into groups called *classes*. You can create multiple instances of a class, called *objects,* to use in your scripts. You can create your own classes and use the built-in ActionScript classes; the built-in classes are located in the Objects folder of the Actions panel.

When you create a class, you define all the *properties* (characteristics) and *methods* (behaviors) of each object it creates, just as real-world objects are defined. For example, a person could be said to have properties such as gender, height, and hair color and methods such as talk, walk, and throw. In this example, Person would be a class, and each individual person would be an object, or an *instance* of that class.

Objects in ActionScript can be pure containers for data, or they can be graphically represented on the Stage as movie clips, buttons, or text fields. All movie clips are instances of the built-in class MovieClip, and all buttons are instances of the built-in class Button. Each movie clip instance contains all the properties (for example, `_height`, `_rotation`, `_totalframes`) and all the methods (for example, `gotoAndPlay`, `loadMovie`, `startDrag`) of the MovieClip class.

To define a class, you create a special function called a *constructor function*. (Built-in classes have built-in constructor functions.) For example, if you want information about a bicycle rider in your movie, you could create a constructor function, `Biker`, with the properties `time` and `distance` and the method `getSpeed`, which tells you how fast the biker is traveling:

```
function Biker(t, d) {
  this.time = t;
  this.distance = d;
  this.getSpeed = function() {return this.time / this.distance;};
}
```

In this example, you create a function that needs two pieces of information, or parameters, to do its job: `t` and `d`. When you call the function to create new instances of the object, you pass it the parameters. The following code creates instances of the object Biker called `emma` and `hamish`.

```
emma = new Biker(30, 5);
hamish = new Biker(40, 5);
```

In object-oriented scripting, classes can receive properties and methods from each other according to a specific order; this is called *inheritance*. You can use inheritance to extend or redefine the properties and methods of a class. A class that inherits from another class is called a *subclass*. A class that passes properties and methods to another class is called a *superclass*. A class can be both a subclass and a superclass.

For more information, see "Using a built-in object" on page 236 and "Creating inheritance" on page 241.

About the MovieClip object

In the Actions panel, the built-in ActionScript classes are called *objects*. You can think of an object as a class instance that allows you to access a certain type of information. For example, a Date object has methods that allow you to read information from the system clock (for example, `getFullYear`, `getMonth`). A Sound object has methods that allow you to control a sound in a movie (for example, `setVolume`, `setPan`). A MovieClip object has methods that allow you to control movie clip instances (for example, `play`, `stop`, and `getURL`) and get and set their properties (for example, `_alpha`, `_framesloaded`, `_visible`).

Movie clips are the most important objects of a Flash movie because they are mini-Flash movies: they have Timelines that run independently of each other. For example, if the main Timeline only has one frame and a movie clip in that frame has ten frames, each frame in the movie clip plays when you play the main movie. This allows instances to act as autonomous objects that can communicate with each other.

Movie clip instances each have a unique instance name so that you can target them with an action. For example, you may have multiple instances on the Stage (for example, `leftClip` and `rightClip`) and only want one to play at a time. To assign an action that tells one particular instance to play, you need to use its name. In the following example, the movie clip's name is `leftClip`:

```
leftClip.play();
```

Instance names also allow you to duplicate, remove, and drag movie clips while a movie plays. Movie clips have properties whose values you can set and retrieve dynamically with ActionScript. Changing and reading these properties can alter the appearance and identity of a movie clip and is the key to creating interactivity. For example, the following script uses the `setProperty` action to set the transparency (alpha setting) of the `navigationBar` instance to 10:

```
setProperty("navigationBar", _alpha, 10);
```

Buttons and text fields in a Flash movie are also objects that you can manipulate with ActionScript. For more information, see "Working with Movie Clips and Buttons" under Help > Using Flash.

How scripts flow

Flash executes ActionScript statements starting with the first statement and continuing in order until it reaches the final statement or a statement that instructs ActionScript to go somewhere else.

Some actions that send ActionScript somewhere other than the next statement are `if` statements, `do..while` loops, and the `return` action.

A flow chart of the `if..else` action

A flow chart of the `do..while` action

An `if` statement is called a conditional statement or a "logical branch" because it controls the flow of a script based on the evaluation of a certain condition. For example, the following code checks to see if the value of the `number` variable is less than or equal to 10. If the check returns `true` (for example, the value of `number` is 5), the variable `alert` is set and displays its value, as shown here:

```
if (myNumber <= 10) {
    alert = "The number is less than or equal to 10";
}
```

You can also add `else` statements to create a more complicated conditional statement. In the following example, if the condition returns `true` (for example, the value of `number` is 3), the statement between the first set of curly braces runs and the `alert` variable is set in the second line. If the condition returns `false` (for example, the value of `number` is 30), the first block of code is skipped and the statement between the curly braces after the `else` statement runs, as in the following:

```
if (number <= 10) {
    alert = "The number is less than or equal to 10";
} else {
    alert = "The number is greater than 10";
}
```

For more information, see "Controlling flow in scripts" on page 230.

Loops repeat an action a certain number of times or until a certain condition is met. In the following example, a movie clip is duplicated five times:

```
i = 0;
do {
    duplicateMovieClip ("myMovieClip", "newMovieClip" + i, i);
    newName = eval("newMovieClip" + i);
    setProperty(newName, _x, getProperty("myMovieClip", _x) + (i * 5));
    i = i + 1;
} while (i <= 5);
```

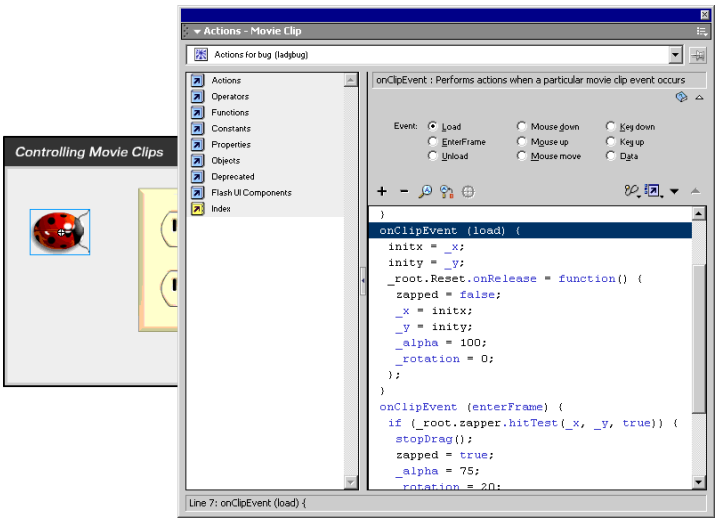
For detailed information, see "Repeating an action" on page 231.

Controlling when ActionScript runs

When you write a script, you use the Actions panel to attach the script to a frame on a Timeline, or to a button or movie clip on the Stage. Scripts attached to a frame run, or *execute*, when the playhead enters that frame. However, scripts attached to the first frame of a movie may behave differently than those attached to subsequent frames because the first frame in a movie is rendered incrementally--objects are drawn on the Stage as they download into the Flash Player--and this can effect when actions execute. All frames after the first frame are rendered all at once when every object in the frame is available.

Scripts attached to movie clips or buttons execute when an event occurs. An event is an occurrence in the movie such as a mouse movement, a keypress, or a movie clip being loaded. You can use ActionScript to find out when these events occur and execute specific scripts depending on the event.

Actions attached to a button or movie clip are enclosed in special actions called *handlers*. The `onClipEvent` and `on` actions are called handlers because they "handle" or manage an event. You can specify one or more events for each handler. Movie clip and button actions execute when the event specified by the handler occurs. You can attach more than one handler to an object if you want different actions to execute when different events occur.



Several `onClipEvent` handlers attached to a movie clip on the Stage

The `onClipEvent` action handles movie clip events, and the `on` action handles button events. You can also use the `on` action with movie clips to create a *button movie clip*, a movie clip that receives button events.

Movie clip events and button events can also be handled by methods of the MovieClip and Button objects. You must define a function and assign it to the event handler method; the function executes when the event occurs. You can use event methods to handle events for dynamically created movie clips. Event methods are also useful for handling all events in a movie in one script: you don't have to attach the script to the object whose event you are handling.

For example, if you have a button on the Stage and you use the Actions panel to add a `trace` action, the following code appears:

```
on (release) {
  trace("You clicked me!");
}
```

You could use a method to create the same effect, as in the following:

```
myMovieClip.onRelease = function() {
  trace("You clicked me!");
}
```

For more information, see "Working with Movie Clips and Buttons" under Help > Using Flash.

The following table lists button event handlers and methods:

Event handler actions	Event handler methods
`on (press)`	`onPress`
`on (release)`	`onRelease`
`on (releaseOutside)`	`onReleaseOutside`
`on (rollOver)`	`onRollOver`
`on (rollOut)`	`onRollOut`
`on (dragOver)`	`onDragOver`
`on (dragOut)`	`onDragOut`
`on (keyPress"...")`	`onKeyDown, onKeyUp`

The following table lists movie clip event handlers and methods:

Event handler actions	Event handler methods
`onClipEvent (load)`	`onLoad`
`onClipEvent (unload)`	`onUnload`
`onClipEvent (enterFrame)`	`onEnterFrame`
`onClipEvent (mouseDown)`	`onMouseDown`
`onClipEvent (mouseUp)`	`onMouseUp`
`onClipEvent (mouseMove)`	`onMouseMove`
`onClipEvent (keyDown)`	`onKeyDown`
`onClipEvent (keyUp)`	`onKeyUp`
`onClipEvent (data)`	`onData`

ActionScript also allows you to handle events for text fields and other ActionScript objects. For more information, see the online ActionScript Dictionary in the Help menu.

ActionScript terminology

Like any scripting language, ActionScript uses its own terminology. The following list provides an introduction to important ActionScript terms in alphabetical order.

Actions are statements that instruct a movie to do something while it is playing. For example, `gotoAndStop` sends the playhead to a specific frame or label. In this manual, the terms *action* and *statement* are interchangeable.

Boolean is a true or false value.

Classes are data types that you can create to define a new type of object. To define a class, you create a constructor function.

Constants are elements that don't change. For example, the constant `Key.TAB` always has the same meaning: it indicates the Tab key on a keyboard. Constants are useful for comparing values.

Constructors are functions that you use to define the properties and methods of a class. For example, the following code creates a new Circle class by creating a constructor function called `Circle`:

```
function Circle(x, y, radius){
  this.x = x;
  this.y = y;
  this.radius = radius;
}
```

Data types are a sets of values and the operations that can be performed on them. The ActionScript data types are string, number, boolean, object, movieclip, function, null, and undefined. For more details on these language elements, see "About data types" on page 216.

Events are actions that occur while a movie is playing. For example, different events are generated when a movie clip loads, the playhead enters a frame, the user clicks a button or movie clip, or the user types at the keyboard.

Event handlers are special actions that manage events such as `mouseDown` or `load`. There are two kinds of ActionScript event handlers: actions and methods. There are only two event handler actions, `on` and `onClipEvent`. In the Actions toolbox, each ActionScript object that has event handler methods has a subcategory called Events.

Expressions are any legal combination of ActionScript symbols that represent a value. An expression consists of operators and operands. For example, in the expression `x + 2`, `x` and `2` are operands and + is an operator.

Functions are blocks of reusable code that can be passed parameters and can return a value. For example, the `getProperty` function is passed the name of a property and the instance name of a movie clip, and it returns the value of the property. The `getVersion` function returns the version of the Flash Player currently playing the movie.

Identifiers are names used to indicate a variable, property, object, function, or method. The first character must be a letter, underscore (_), or dollar sign ($). Each subsequent character must be a letter, number, underscore, or dollar sign. For example, `firstName` is the name of a variable.

Instances are objects that belong to a certain class. Each instance of a class contains all the properties and methods of that class. All movie clips are instances with properties (for example, `_alpha` and `_visible`) and methods (for example, `gotoAndPlay` and `getURL`) of the MovieClip class.

Instance names are unique names that allow you to target movie clip and button instances in scripts. You use the Property inspector to assign instance names to instances on the Stage. For example, a master symbol in the library could be called `counter` and the two instances of that symbol in the movie could have the instance names `scorePlayer1` and `scorePlayer2`. The following code sets a variable called `score` inside each movie clip instance by using instance names:

```
_root.scorePlayer1.score += 1;
_root.scorePlayer2.score -= 1; <<Loc: semicolons added to each of these lines-
  -IMD>>
```

Keywords are reserved words that have special meaning. For example, `var` is a keyword used to declare local variables. You cannot use a keyword as an identifier. For example, `var` is not a legal variable name.

Methods are functions assigned to an object. After a function is assigned, it can be called as a method of that object. For example, in the following code, `clear` becomes a method of the `controller` object:

```
function reset(){
  this.x_pos = 0;
  this.x_pos = 0;
}
controller.clear = reset;
controller.clear();
```

Objects are collections of properties and methods; each object has its own name and is an instance of a particular class. Built-in objects are predefined in the ActionScript language. For example, the built-in Date object provides information from the system clock.

Operators are terms that calculate a new value from one or more values. For example, the addition (+) operator adds two or more values together to produce a new value. The values that operators manipulate are called *operands*.

Parameters (also called *arguments*) are placeholders that let you pass values to functions. For example, the following `welcome` function uses two values it receives in the parameters `firstName` and `hobby`:

```
function welcome(firstName, hobby) {
  welcomeText = "Hello, " + firstName + "I see you enjoy " + hobby;
}
```

Properties are attributes that define an object. For example, `_visible` is a property of all movie clips that defines whether a movie clip is visible or hidden.

Target paths are hierarchical addresses of movie clip instance names, variables, and objects in a movie. <<Loc: we changed “hierarchical names” to “hierarchical addresses in previous sentence -- IMD>>You name a movie clip instance in the movie clip Property inspector. (The main Timeline always has the name `_root`.) You can use a target path to direct an action at a movie clip or to get or set the value of a variable. For example, the following statement is the target path to the variable `volume` inside the movie clip `stereoControl`:

```
_root.stereoControl.volume
```

Variables are identifiers that hold values of any data type. Variables can be created, changed, and updated. The values they store can be retrieved for use in scripts. In the following example, the identifiers on the left side of the equal signs are variables:

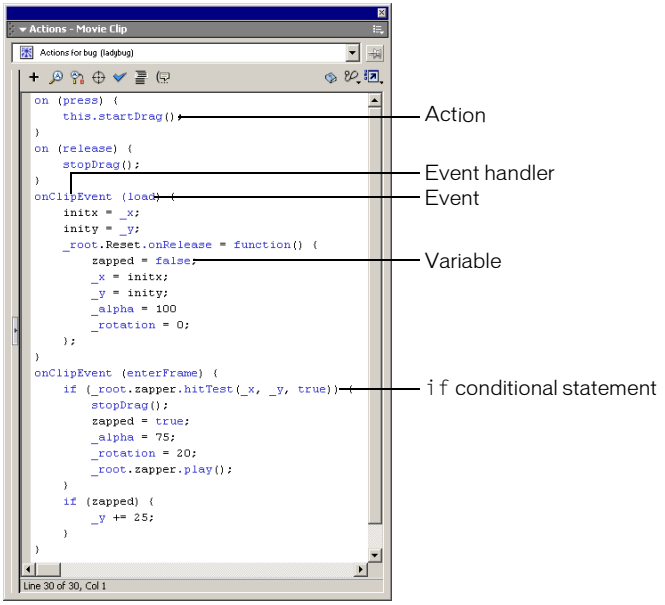
```
x = 5;
name = "Lolo";
customer.address = "66 7th Street";
c = new Color(mcinstanceName);
```

Deconstructing a sample script

In the sample movie zapper.swf, which you can view in Flash Help,when a user drags the bug to the electrical outlet the bug falls and the outlet shakes. The main Timeline has only one frame and contains three objects: the ladybug, the outlet, and a reset button.Each of these objects is a movie clip instance.

The bug and zapper movie clip instances on the Stage in frame 1

There is one script in the movie; it's attached to the `bug` instance, as in the Actions panel below:



The Actions panel with the script attached to the `bug` instance

The bug's instance name is `bug` and the outlet's instance name is `zapper`. In the script the bug is referred to as `this` because the script is attached to the bug and the reserved word `this` refers to the object that contains it.<<Loc: we changed "calls" to "contains" in previous sentence --IMD>>

There are two `onClipEvent` handlers with two different events: `load` and `enterFrame`. The actions in the `onClipEvent(load)` statement execute only once, when the movie loads. The actions in the `onClipEvent(enterFrame)` statement execute every time the playhead enters a frame. Even in a one-frame movie, the playhead still enters that frame repeatedly and the script executes repeatedly. The following actions occur within each `onClipEvent` handler:

onClipEvent(load) Two variables, `initx` and `inity`, are defined to store the inital x and y positions of the `bug` movie clip instance. A function is defined and assigned to the `onRelease` event of the Reset instance. This function is called each time the mouse is pressed and released on the Reset button. The function places the ladybug back in its starting position on the Stage, resets its rotation and alpha values, and resets the `zapped` variable to false.

onClipEvent(enterFrame) A conditional `if` statement uses the `hitTest` method to check whether the bug instance is touching the outlet instance (`_root.zapper`). There are two possible outcomes of the evaluation, `true` or `false`:

```
onClipEvent (load) {
  initx = _x;
  inity = _y;
  _root.Reset.onRelease = function() {
    zapped = false;
    _x = initx;
    _y = inity;
    _alpha = 100
    _rotation = 0;
  };
}
```

If the `hitTest` method returns `true`, the `stopDrag` method is called, the `zapper` variable is set to `true`, the alpha and rotation properties are changed, and the `zapped` instance is told to play.

If the `hitTest` method returns `false`, none of the code within the `{}` immediately following the `if` statement runs.

There are two `on` handlers attached to the `bug` isntance with two different events: `press` and `release`. The actions in the `on(press)` statement execute when the mouse is pressed over the `bug` instance. The actions in the `on(release)` statement execute when the mouse is released over the `bug` instance. The following actions occur within each `onClipEvent` handler:

on(press) A `startDrag` action makes the ladybug draggable. Because the script is attached to the `bug` instance, the keyword `this` indicates that it is the `bug` instance that is draggable:

```
on (press) {
  this.startDrag();
}
```

on(release) A `stopDrag` action stops the drag action:

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

To watch the movie play, see Flash Help.

Using ActionScript syntax

ActionScript has rules of grammar and punctuation that determine which characters and words are used to create meaning and in which order they can be written. For example, in English, a period ends a sentence. In ActionScript, a semicolon ends a statement.

The following general rules apply to all ActionScript. Most ActionScript terms also have their own individual requirements; for the rules for a specific term, see its entry in the online ActionScript Dictionary in the Help menu.

Dot syntax

In ActionScript, a dot (`.`) is used to indicate the properties or methods related to an object or movie clip. It is also used to identify the target path to a movie clip, variable, function, or object. A dot syntax expression begins with the name of the object or movie clip followed by a dot, and ends with the element you want to specify.

For example, the _x movie clip property indicates a movie clip's *x* axis position on the Stage. The expression `ballMC._x` refers to the _x property of the movie clip instance `ballMC`.

As another example, `submit` is a variable set in the `form` movie clip, which is nested inside the movie clip `shoppingCart`. The expression `shoppingCart.form.submit = true` sets the `submit` variable of the instance `form` to `true`.

Expressing a method of an object or movie clip follows the same pattern. For example, the `play` method of the `ballMC` instance moves the playhead in the Timeline of `ballMC`, as in the following statement:

```
ballMC.play();
```

Dot syntax also uses two special aliases, `_root` and `_parent`. The alias `_root` refers to the main Timeline. You can use the `_root` alias to create an absolute target path. For example, the following statement calls the function `buildGameBoard` in the movie clip `functions` on the main Timeline:

```
_root.functions.buildGameBoard();
```

You can use the alias `_parent` to refer to a movie clip in which the current movie clip is nested. You can also use `_parent` to create a relative target path. For example, if the movie clip `dog` is nested inside the movie clip `animal`, the following statement on the instance `dog` tells `animal` to stop:

```
_parent.stop();
```

See "Working with Movie Clips and Buttons" under Help > Using Flash.

Curly braces

ActionScript statements are grouped together into blocks with curly braces (`{ }`), as in the following script:

```
on(release) {
  myDate = new Date();
  currentMonth = myDate.getMonth();
}
```

Semicolons

An ActionScript statement is terminated with a semicolon. For example, the following statements are terminated with semicolons:

```
column = passedDate.getDay();
row    = 0;
```

If you omit the terminating semicolon, Flash will still compile your script successfully. However, using semicolons is good scripting practice.

Parentheses

When you define a function, place any parameters inside parentheses:

```
function myFunction (name, age, reader){
  ...
}
```

When you call a function, include any parameters passed to the function in parentheses, as shown here:

```
myFunction ("Steve", 10, true);
```

You can also use parentheses to override the ActionScript order of precedence or to make your ActionScript statements easier to read. (See “Operator precedence” on page 224.)

You also use parentheses to evaluate an expression on the left side of a dot in dot syntax. For example, in the following statement, the parentheses cause `new Color(this)` to evaluate and create a new Color object:<<Loc: we deleted last closing parenthesis in line 2 below --IMD>>

```
onClipEvent(enterFrame) {
  (new Color(this)).setRGB(0xffffff);
}
```

If you didn't use parentheses, you would need to add a statement to evaluate the expression:

```
onClipEvent(enterFrame) {
  myColor = new Color(this);
  myColor.setRGB(0xffffff);
}
```

Uppercase and lowercase letters

Only keywords in ActionScript are case sensitive; with the rest of ActionScript, you can use uppercase and lowercase letters interchangeably. For example, the following statements are equivalent:

```
cat.hilite = true;
CAT.hilite = true;
```

However, it's good practice to follow consistent capitalization conventions, such as those used in this manual, to make it is easier to identify names of functions and variables when reading ActionScript code.

Because ActionScript is not case sensitive, you must not use variable names that match built-in ActionScript objects. For example, the following is not allowed:

```
date = new Date();
```

Instead, use the variable names `myDate`, `theDate`, and so on.

If you don't use correct capitalization with keywords, your script will have errors. When Colored Syntax is turned on in the Actions panel, keywords written with correct capitalization are blue by default. For more information, see “Keywords” on page 216 and “About syntax highlighting” under Help > Using Flash.

Comments

In the Actions panel, use comments to add notes to scripts. Comments are useful for keeping track of what you intended, and for passing information to other developers if you work in a collaborative environment or are providing samples.

When you choose the `comment` action, the characters `//` are inserted into the script. Even a simple script is easier to understand if you make notes as you create it:

```
on(release) {
  // create new Date object
  myDate = new Date();
  currentMonth = myDate.getMonth();
  // convert month number to month name
  monthName = calcMonth(currentMonth);
  year = myDate.getFullYear();
  currentDate = myDate.getDate();<<Loc: changed “getDat” to “getDate” here>>
}
```

When Colored Syntax is turned on in the Actions panel, comments are pink by default in the Script pane. Comments can be any length without affecting the size of the exported file, and they do not need to follow rules for ActionScript syntax or keywords

Keywords

ActionScript reserves words for specific use within the language, so you can't use them as variable, function, or label names. The following table lists all ActionScript keywords:

break	else	instanceof	typeof
case	for	new	var
continue	function	return	void
default	if	switch	while
delete	in	this	with

For more information about a specific keyword, see its entry in the online ActionScript Dictionary in the Help menu.

Constants

A constant is a property whose value never changes.

For example, the constants `BACKSPACE`, `ENTER`, `QUOTE`, `RETURN`, `SPACE`, and `TAB` are properties of the Key object and refer to keyboard keys. To test whether the user is pressing the Enter key, you could use the following statement:

```
if(Key.getCode() == Key.ENTER) {
  alert = "Are you ready to play?";
  controlMC.gotoAndStop(5);
}
```

About data types

A data type describes the kind of information a variable or ActionScript element can hold. There are two kinds of data types: primitive and reference. The primitive data types—string, number, and boolean—have a constant value and therefore can hold the actual value of the element they represent. The reference data types—movie clip and object—have values that can change and therefore contain references to the actual value of the element. Variables containing primitive data types behave differently in certain situations than those containing reference types. (See "Using variables in a script" on page 222.) There are also two special data types: null and undefined.

Each data type has its own rules and is described here. References are included for data types that are discussed in more detail.

String

A string is a sequence of characters such as letters, numbers, and punctuation marks. You enter strings in an ActionScript statement by enclosing them in single or double quotation marks. Strings are treated as characters instead of as variables. For example, in the following statement, `"L7"` is a string:

```
favoriteBand = "L7";
```

You can use the addition (+) operator to *concatenate*, or join, two strings. ActionScript treats spaces at the beginning or end of a string as a literal part of the string. The following expression includes a space after the comma:

```
greeting = "Welcome," + firstName;
```

Although ActionScript does not distinguish between uppercase and lowercase in references to variables, instance names, and frame labels, literal strings are case sensitive. For example, the following two statements place different text in the specified text field variables, because `"Hello"` and `"HELLO"` are literal strings.

```
invoice.display = "Hello";
invoice.display = "HELLO";
```

To include a quotation mark in a string, precede it with a backslash character (\). This is called "escaping" a character. There are other characters that cannot be represented in ActionScript except by special escape sequences. The following table provides all the ActionScript escape characters:

Escape sequence	Character
`\b`	Backspace character (ASCII 8)
`\f`	Form-feed character (ASCII 12)
`\n`	Line-feed character (ASCII 10)
`\r`	Carriage return character (ASCII 13)
`\t`	Tab character (ASCII 9)
`\"`	Double quotation mark
`\'`	Single quotation mark
`\\`	Backslash
`\000 - \377`	A byte specified in octal
`\x00 - \xFF`	A byte specified in hexadecimal
`\u0000 - \uFFFF`	A 16-bit Unicode character specified in hexadecimal

Number

The number data type is a double-precision floating-point number. You can manipulate numbers using the arithmetic operators addition (+), subtraction (-), multiplication (*), division (/), modulo (%), increment (++), and decrement (--). You can also use methods of the built-in Math object to manipulate numbers. The following example uses the `sqrt` (square root) method to return the square root of the number 100:

```
Math.sqrt(100);
```

See "Numeric operators" on page 224.

Boolean

A Boolean value is one that is either true or false. ActionScript also converts the values `true` and `false` to 1 and 0 when appropriate. Boolean values are most often used with logical operators in ActionScript statements that make comparisons to control the flow of a script. For example, in the following script, the movie plays if the variable `password` is `true`:

```
onClipEvent(enterFrame) {
  if (userName == true && password == true){
    play();
  }
}
```

See "Controlling flow in scripts" on page 230 and "Logical operators" on page 226.

Object

An object is a collection of properties. Each property has a name and a value. The value of a property can be any Flash data type, even the object data type. This allows you to arrange objects inside each other, or "nest" them. To specify objects and their properties, you use the dot (`.`) operator. For example, in the following code, `hoursWorked` is a property of `weeklyStats`, which is a property of `employee`:

```
employee.weeklyStats.hoursWorked
```

You can use the built-in ActionScript objects to access and manipulate specific kinds of information. For example, the Math object has methods that perform mathematical operations on numbers you pass to them. This example uses the `sqrt` method:

```
squareRoot = Math.sqrt(100);
```

The ActionScript MovieClip object has methods that let you control movie clip symbol instances on the Stage. This example uses the `play` and `nextFrame` methods:

```
mcInstanceName.play();
mc2InstanceName.nextFrame();
```

You can also create your own objects to organize information in your movie. To add interactivity to a movie with ActionScript, you'll need many different pieces of information: for example, you might need a user's name, the speed of a ball, the names of items in a shopping cart, the number of frames loaded, the user's zip code, or the key that was pressed last. Creating custom objects allows you to organize this information into groups, simplify your scripting, and reuse your scripts.

Movieclip

Movie clips are symbols that can play animation in a Flash movie. They are the only data type that refers to a graphic element. The movieclip data type allows you to control movie clip symbols using the methods of the MovieClip object. You call the methods using the dot (`.`) operator, as shown here:

```
myClip.startDrag(true);
parentClip.getURL("http://www.macromedia.com/support/" + product);
```

Null

The null data type has only one value, `null`. This value means "no value"—that is, a lack of data. The `null` value can be used in a variety of situations. Here are some examples:

- To indicate that a variable has not yet received a value

- To indicate that a variable no longer contains a value
- As the return value of a function, to indicate that no value was available to be returned by the function
- As a parameter to a function, to indicate that a parameter is being omitted

Undefined

The undefined data type has one value, `undefined`, and is used for a variable that hasn't been assigned a value.

About variables

A variable is a container that holds information. The container itself is always the same, but the contents can change. By changing the value of a variable as the movie plays, you can record and save information about what the user has done, record values that change as the movie plays, or evaluate whether a condition is true or false.

It's a good idea always to assign a variable a known value the first time you define the variable. This is known as *initializing a variable* and is often done in the first frame of the movie. Initializing variables helps you track and compare the variable's value as the movie plays.

Variables can hold any type of data: number, string, Boolean, object, or movie clip. The type of data a variable contains affects how the variable's value changes when it is assigned in a script.

Typical types of information you can store in a variable include a URL, a user's name, the result of a mathematical operation, the number of times an event occurred, or whether a button has been clicked. Each movie and movie clip instance has its own set of variables, with each variable having its own value independent of variables in other movies or movie clips.

Naming a variable

A variable's name must follow these rules:

- It must be an identifier.
- It cannot be a keyword or an ActionScript literal such as `true`, `false`, `null`, or `undefined`.
- It must be unique within its scope. (See "Scoping a variable" on page 220.)

Typing a variable

In Flash, you do not need to explicitly define a variable as holding either a number, a string, or other data type. Flash determines the data type of a variable when the variable is assigned:

```
x = 3;
```

In the expression `x = 3`, Flash evaluates the element on the right side of the operator and determines that it is of type number. A later assignment may change the type of `x`; for example, `x = "hello"` changes the type of `x` to a string. A variable that hasn't been assigned a value has a type of `undefined`.

ActionScript converts data types automatically when an expression requires it. For example, when you pass a value to the `trace` action, `trace` automatically converts the value to a string and sends it to the Output window. In expressions with operators, ActionScript converts data types as needed; for example, when used with a string, the + operator expects the other operand to be a string:

```
"Next in line, number " + 7
```

ActionScript converts the number 7 to the string `"7"` and adds it to the end of the first string, resulting in the following string:

```
"Next in line, number 7"
```

When you debug scripts, it's often useful to determine the data type of an expression or variable to understand why it is behaving a certain way. You can do this with the `typeof` operator, as in this example:

```
trace(typeof(variableName));
```

To convert a string to a numerical value, use the `Number` function. To convert a numerical value to a string, use the `String` function. For detailed information on each action, see the online ActionScript Dictionary in the Help menu.

Scoping a variable

A variable's "scope" refers to the area in which the variable is known and can be referenced. There are three types of variable scope in ActionScript:

- *Local variables* are available within their own block of code (delineated by curly braces).
- *Timeline variables* are available to any Timeline if you use a target path.
- *Global variables* are available to any Timeline even if you do not use a target path.

You can use the `var` statement to declare a local variable inside a script. For example, the variables `i` and `j` are often used as loop counters. In the following example, `i` is used as a local variable; it only exists inside the function `makeDays`:

```
function makeDays() {
  var i;
  for( i = 0; i < monthArray[month]; i++ ) {

    _root.Days.attachMovie( "DayDisplay", i, i + 2000 );

    _root.Days[i].num = i + 1;
    _root.Days[i]._x = column * _root.Days[i]._width;
    _root.Days[i]._y = row * _root.Days[i]._height;

    column = column + 1;

    if (column == 7 ) {

      column = 0;
      row = row + 1;
    }
  }
}
```

Local variables can also help prevent name collisions, which can cause errors in your movie. For example, if you use `name` as a local variable, you could use it to store a user name in one context and a movie clip instance name in another; because these variables would run in separate scopes, there would be no collision.

It's good practice to use local variables in the body of a function so that the function can act as an independent piece of code. A local variable is only changeable within its own block of code. If an expression in a function uses a global variable, something outside the function could change its value, which would change the function.

Variable declaration

To declare Timeline variables, use the `set variable` action or the assignment (=) operator. Both methods achieve the same results. <<Loc: "regular" changed to "Timeline" in first sentence --IMD>>

To declare local variables, use the `var` statement inside the body of a function. A local variable is scoped to the block, and expires at the end of the block. A local variable not declared within a block expires at the end of its script.

To declare global variables, use the `_global` identifier before the variable name. The following example creates the global variable `myName`:

```
_global.myName = "George";
```

To test the value of a variable, use the `trace` action to send the value to the Output window. For example, `trace(hoursWorked)` sends the value of the variable `hoursWorked` to the Output window in test mode. You can also check and set the variable values in the Debugger in test mode. For more information, see "Testing a movie" under Help > Using Flash.

Using variables in a script

You must declare a variable in a script before you can use it in an expression. If you use an undeclared variable, as in the following example, the variable's value will be `undefined` and your script will generate an error:

```
getURL(myWebSite);
myWebSite = "http://www.shrimpmeat.net";
```

The statement declaring the variable `myWebSite` must come first so that the variable in the `getURL` action can be replaced with a value.

You can change the value of a variable many times in a script. The type of data that the variable contains affects how and when the variable changes. Primitive data types, such as strings and numbers, are passed by value. This means that the actual content of the variable is passed to the variable.

In the following example, `x` is set to 15 and that value is copied into `y`. When `x` is changed to 30 in line 3, the value of `y` remains 15 because `y` doesn't look to `x` for its value; it contains the value of `x` that it received in line 2.

```
var x = 15;
var y = x;
var x = 30;
```

As another example, the variable `inValue` contains a primitive value, 3, so the actual value is passed to the `sqrt` function and the returned value is 9:

```
function sqrt(x){
  return x * x;
}

var inValue = 3;
var out = sqr(in);
```

The value of the variable `inValue` does not change.

The object data type can contain such a large and complex amount of information that a variable with this type doesn't hold the actual value; it holds a reference to the value. This reference is like an alias that points to the contents of the variable. When the variable needs to know its value, the reference asks for the contents and returns the answer without transferring the value to the variable.

The following is an example of passing by reference:

```
var myArray = ["tom", "dick"];
var newArray = myArray;
myArray[1] = "jack";
trace(newArray);
```

The above code creates an Array object called `myArray` that has two elements. The variable `newArray` is created and is passed a reference to `myArray`. When the second element of `myArray` is changed, it affects every variable with a reference to it. The `trace` action sends `tom, jack` to the Output window.

In the next example, `myArray` contains an Array object, so it is passed to function `zeroArray` by reference. The `zeroArray` function changes the content of the array in `myArray`.

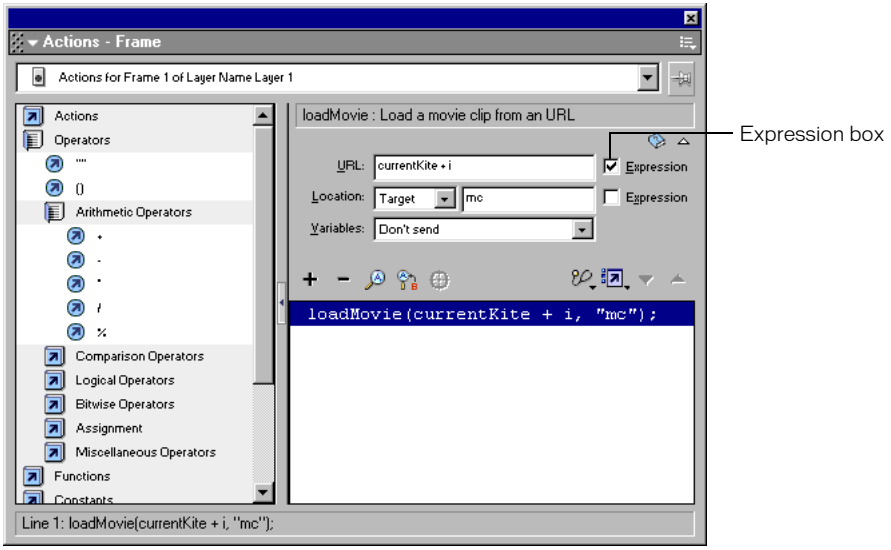
```
function zeroArray (theArray){
  var i;
  for (i=0; i < theArray.length; i++) {
    theArray[i] = 0;
  }
}

var myArray = new Array();
myArray[0] = 1;
myArray[1] = 2;
myArray[2] = 3;
zeroArray(myArray);
```

The function `zeroArray` accepts an Array object as a parameter and sets all the elements of that array to 0. It can modify the array because the array is passed by reference.

Using operators to manipulate values in expressions

An expression is any statement that Flash can evaluate and that returns a value. You can create an expression by combining operators and values, or by calling a function. When you write an expression in the Actions panel in normal mode, make sure the Expression box is selected in the parameters area; otherwise, the parameter text box contains the literal value of a string.



Operators are characters that specify how to combine, compare, or modify the values of an expression. The elements that the operator performs on are called *operands*. For example, in the following statement, the + operator adds the value of a numeric literal to the value of the variable `foo`; `foo` and `3` are the operands:

```
foo + 3
```

This section describes general rules about common types of operators. For detailed information on each operator mentioned here, as well as special operators that don't fall into these categories, see the online ActionScript Dictionary in the Help menu.

Operator precedence

When two or more operators are used in the same statement, some operators take precedence over others. ActionScript follows a precise hierarchy to determine which operators to execute first. For example, multiplication is always performed before addition; however, items in parentheses take precedence over multiplication. So, without parentheses, ActionScript performs the multiplication in the following example first:

```
total = 2 + 4 * 3;
```

The result is 14.

But when parentheses surround the addition operation, ActionScript performs the addition first:

```
total = (2 + 4) * 3;
```

The result is 18.

For a table of all operators and their precedence, see "Operator Precedence and Associativity" under Help > Using Flash.

Operator associativity

When two or more operators share the same precedence, their associativity determines the order in which they are performed. Associativity can be either left-to-right or right-to-left. For example, the multiplication operator has an associativity of left-to-right; therefore, the following two statements are equivalent:

```
total = 2 * 3 * 4;
total = (2 * 3) * 4;
```

For a table of all operators and their associativity, see "Operator Precedence and Associativity" under Help > Using Flash.

Numeric operators

Numeric operators add, subtract, multiply, divide, and perform other arithmetic operations.

The most common usage of the increment operator is `i++` instead of the more verbose `i = i+1`. You can use the increment operator before or after an operand. In the following example, `age` is incremented first and then tested against the number 30:

```
if (++age >= 30)
```

In the following example, `age` is incremented after the test is performed:

```
if (age++ >= 30)
```

The following table lists the ActionScript numeric operators:

Operator	Operation performed
+	Addition
*	Multiplication
/	Division
%	Modulo (remainder of division)
-	Subtraction
++	Increment
--	Decrement

Comparison operators

Comparison operators compare the values of expressions and return a Boolean value (`true` or `false`). These operators are most commonly used in loops and in conditional statements. In the following example, if the variable `score` is `100`, a certain movie loads; otherwise, a different movie loads:

```
if (score > 100){
  loadMovieNum("winner.swf", 5);
} else {
  loadMovieNum("loser.swf", 5);
}
```

The following table lists the ActionScript comparison operators:

Operator	Operation performed
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal

String operators

The + operator has a special effect when it operates on strings: it concatenates the two string operands. For example, the following statement adds `"Congratulations,"` to `"Donna!"`:

```
"Congratulations, " + "Donna!"
```

The result is `"Congratulations, Donna!"` If only one of the + operator's operands is a string, Flash converts the other operand to a string.

The comparison operators >, >=, <, and <= also have a special effect when operating on strings. These operators compare two strings to determine which is first in alphabetical order. The comparison operators only compare strings if both operands are strings. If only one of the operands is a string, ActionScript converts both operands to numbers and performs a numeric comparison.

Logical operators

Logical operators compare Boolean (`true` and `false`) values and return a third Boolean value. For example, if both operands evaluate to `true`, the logical AND operator (`&&`) returns `true`. If one or both of the operands evaluate to `true`, the logical OR operator (`||`) returns `true`. Logical operators are often used in conjunction with comparison operators to determine the condition of an `if` action. For example, in the following script, if both expressions are true, the `if` action will execute:

```
if (i > 10 && _framesloaded > 50){
  play();
}
```

The following table lists the ActionScript logical operators:

Operator	Operation performed
`&&`	Logical AND
`||`	Logical OR
`!`	Logical NOT

Bitwise operators

Bitwise operators internally manipulate floating-point numbers to change them into 32-bit integers. The exact operation performed depends on the operator, but all bitwise operations evaluate each binary digit (bit) of the 32-bit integer individually to compute a new value.

The following table lists the ActionScript bitwise operators:

Operator	Operation performed
`&`	Bitwise AND
`|`	Bitwise OR
`^`	Bitwise XOR
`~`	Bitwise NOT
`<<`	Shift left
`>>`	Shift right
`>>>`	Shift right zero fill

Equality operators

You can use the equality (==) operator to determine whether the values or identities of two operands are equal. This comparison returns a Boolean (`true` or `false`) value. If the operands are strings, numbers, or Boolean values, they are compared by value. If the operands are objects or arrays, they are compared by reference.

It is a common mistake to use the assignment operator to check for equality. For example, the following code compares *x* to 2:

```
if (x == 2)
```

In that same example, the expression `x = 2` is incorrect because it doesn't compare the operands, it assigns the value of 2 to the variable `x`.

The strict equality (===) operator is like the equality operator, with one important difference: the strict equality operator does not perform type conversion. If the two operands are of different types, the strict equality operator returns `false`. The strict inequality (`!==`) operator returns the inversion of the strict equality operator.

The following table lists the ActionScript equality operators:

Operator	Operation performed
`==`	Equality
`===`	Strict equality
`!=`	Inequality
`!==`	Strict inequality

Assignment operators

You can use the assignment (=) operator to assign a value to a variable, as in the following:

```
password = "Sk8tEr";
```

You can also use the assignment operator to assign multiple variables in the same expression. In the following statement, the value of `a` is assigned to the variables `b`, `c` and `d`:

```
a = b = c = d;
```

You can also use compound assignment operators to combine operations. Compound operators perform on both operands and then assign that new value to the first operand. For example, the following two statements are equivalent:

```
x += 15;
x = x + 15;
```

The assignment operator can also be used in the middle of an expression, as in the following:

```
// If the flavor is not vanilla, output a message.
if ((flavor = getIceCreamFlavor()) != "vanilla") {
  trace ("Flavor was " + flavor + ", not vanilla.");
}
```

This code is equivalent to the slightly more verbose code that follows:

```
flavor = getIceCreamFlavor();
if (flavor != "vanilla") {
  trace ("Flavor was " + flavor + ", not vanilla.");
}
```

The following table lists the ActionScript assignment operators:

Operator	Operation performed
=	Assignment
+=	Addition and assignment
-=	Subtraction and assignment
*=	Multiplication and assignment
%=	Modulo and assignment
/=	Division and assignment
<<=	Bitwise shift left and assignment
>>=	Bitwise shift right and assignment
>>>=	Shift right zero fill and assignment
^=	Bitwise XOR and assignment
|=	Bitwise OR and assignment
&=	Bitwise AND and assignment

Dot and array access operators

You can use the dot operator (`.`) and the array access operator (`[]`) to access built-in or custom ActionScript object properties, including those of a movie clip.

The dot operator uses the name of an object on its left side and the name of a property or variable on its right side. The property or variable name can't be a string or a variable that evaluates to a string; it must be an identifier. The following examples use the dot operator:

```
year.month = "June";
year.month.day = 9;
```

The dot operator and the array access operator perform the same role, but the dot operator takes an identifier as its property, whereas the array access operator evaluates its contents to a name and then accesses the value of that named property. For example, the following expressions access the same variable `velocity` in the movie clip `rocket`:

```
rocket.velocity;
rocket["velocity"];
```

You can use the array access operator to dynamically set and retrieve instance names and variables. For example, in the following code, the expression inside the `[]` operator is evaluated and the result of the evaluation is used as the name of the variable to be retrieved from movie clip `name`:

```
name["mc" + i]
```

You can also use the `eval` function, as shown here:

```
eval("mc" + i)<<Loc: deleted semicolon here --IMD>>
```

The array access operator can also be used on the left side of an assignment statement. This allows you to dynamically set instance, variable, and object names, as in the following example:

```
name[index] = "Gary";
```

You create multidimensional arrays in ActionScript by constructing an array, the elements of which are also arrays. To access elements of a multidimensional array, you can nest the array access operator with itself, as in the following:

```
var chessboard = new Array();
for (var i=0; i<8; i++) {
  chessboard.push(new Array(8));
}
function getContentsOfSquare(row, column){
  chessboard[row][column];
}
```

Using actions

Actions are ActionScript statements, or commands. Multiple actions assigned to the same frame or object create a script. Actions can act independently of each other, as in the following statements:

```
mc1.swapDepths(mc2);
gotoAndPlay(15);
```

You can also nest actions by using one action inside another; this allows actions to affect each other. In the following example, the `if` action tells the `gotoAndPlay` action when to execute:

```
if (i >= 25) {
  gotoAndPlay(10);
}
```

Actions can move the playhead in the Timeline (`gotoAndPlay`), control the flow of a script by creating loops (`do while`) or conditional logic (`if`), or create new functions and variables (`function`, `setVariable`). The following table lists all ActionScript actions:

break	#endinitclip	loadMovie	printAsBitmap	switch
call	evaluate	loadMovieNum	printAsBitmapNum	tellTarget
call function	for	loadVariables	printNum	toggleHighQuality
case	for..in	loadVariablesNum	removeMovieClip	trace
clearInterval	fsCommand	method	return	unloadMovie
comment	function	nextFrame	set variable	unloadMovieNum
continue	getURL	nextScene	setInterval	updateAfterEvent
default	gotoAndPlay	on	setProperty	var
delete	gotoAndStop	onClipEvent	startDrag	with
do while	if	play	stop	while
duplicate MovieClip	ifFrameLoaded	prevFrame	stopAllSounds	
else	include	prevScene	stopDrag	
else if	#initclip	print	swapDepths	

For syntax and usage examples of each action, see individual entries in the online ActionScript Dictionary in the Help menu.

Note: In this manual, the ActionScript term *action* is synonymous with the JavaScript term *statement*.

Writing a target path

To use an action to control a movie clip or loaded movie, you must specify its name and its address, called a *target path*.

In ActionScript you identify a movie clip by its instance name. For example, in the following statement, the `_alpha` property of the movie clip named `star` is set to 50% visibility:

```
star._alpha = 50;
```

To give a movie clip an instance name:

1. Select the movie clip on the Stage.
2. Enter an instance name in the Property inspector.

To identify a loaded movie:

Use `_level`*X*, where *X* is the level number specified in the `loadMovie` action that loaded the movie.

For example, a movie loaded into level 5 has the target path `_level5`. In the following example, a movie is loaded into level 5 and its visibility is set to `false`:

```
onClipEvent(load) {
  loadMovieNum("myMovie.swf", 5);
}
onClipEvent(enterFrame) {
  _level5._visible = false;
}
```

To enter a movie's target path:

In the Actions panel (Window > Actions), click the Insert Target Path button and select a movie clip from the list that appears.

For more information about writing target paths, see "Working with Movie Clips and Buttons" under Help > Using Flash.

Controlling flow in scripts

ActionScript uses `if`, `else`, `else if`, `for`, `while`, `do..while`, `for..in`, and `switch` actions to perform an action depending on whether a condition exists.<<Loc: deleted second sentence here --IMD>>

Checking a condition

Statements that check whether a condition is `true` or `false` begin with the term `if`. If the condition exists, ActionScript executes the statement that follows. If the condition doesn't exist, ActionScript skips to the next statement outside the block of code.

To optimize your code's performance, check for the most likely conditions first.

The following statements test several conditions. The term `else if` specifies alternative tests to perform if previous conditions are false.

```
if (password == null || email == null) {
  gotoAndStop("reject");
} else if (password == userID){
  gotoAndPlay("startMovie");
}
```

Repeating an action

ActionScript can repeat an action a specified number of times or while a specific condition exists. Use the `while`, `do..while`, `for`, and `for..in` actions to create loops.

To repeat an action while a condition exists:

Use the `while` statement.

A `while` loop evaluates an expression and executes the code in the body of the loop if the expression is `true`. After each statement in the body is executed, the expression is evaluated again. In the following example, the loop executes four times:

```
i = 4;
while (i > 0) {
  myMC.duplicateMovieClip("newMC" + i, i );
  i--;
}
```

You can use the `do..while` statement to create the same kind of loop as a `while` loop. In a `do..while` loop, the expression is evaluated at the bottom of the code block so the loop always runs at least once, as in the following:

```
i = 4;
do {
  myMC.duplicateMovieClip("newMC" +i, i );
  i--;
} while (i > 0);
```

To repeat an action using a built-in counter:

Use the `for` statement.

Most loops use a counter of some kind to control how many times the loop executes. Each execution of a loop is called an *iteration.* You can declare a variable and write a statement that increases or decreases the variable each time the loop executes. In the `for` action, the counter and the statement that increments the counter are part of the action. In the following example, the first expression (`i = 4`) is the initial expression that is evaluated before the first iteration. The second expression (`i > 0`) is the condition that is checked each time before the loop runs. The third expression (`i--`) is called the *post expression* and is evaluated each time after the loop runs.

```
for (i = 4; i > 0; i--){
  myMC.duplicateMovieClip("newMC" + i, i + 10);
}
```

To loop through the children of a movie clip or object:

Use the `for..in` statement.

Children include other movie clips, functions, objects, and variables. The following example uses `trace` to print its results in the Output window:

```
myObject = { name:'Joe', age:25, city:'San Francisco' };
for (propertyName in myObject) {
  trace("myObject has the property: " + propertyName + ", with the value: " +
  myObject[propertyName]);
}
```

This example produces the following results in the Output window:

```
myObject has the property: name, with the value: Joe
myObject has the property: age, with the value: 25
myObject has the property: city, with the value: San Francisco
```

You may want your script to iterate over a particular type of child—for example, over only movie clip children. You can do this with `for..in` in conjunction with the `typeof` operator.

```
for (name in myMovieClip) {
  if (typeof (myMovieClip[name]) == "movieclip") {
    trace("I have a movie clip child named " + name);
  }
}
```

Note: The `for..in` statement iterates over properties of objects in the iterated prototype chain of the object. If a child object's prototype is `parent`, `for..in` will also iterate over the properties of `parent`. See "Creating inheritance" on page 241.

For more information on each action, see individual entries in the online ActionScript Dictionary in the Help menu.

Using built-in functions

A function is a block of ActionScript code that can be reused anywhere in a movie. If you pass values as parameters to a function, the function will operate on those values. A function can also return values.

Flash has built-in functions that allow you to access certain information and perform certain tasks, such as getting the version number of the Flash Player hosting the movie (`getVersion`). Functions that belong to an object are called *methods*. Functions that don't belong to an object are called *top-level functions* and are found in the Functions category of the Actions panel.

Each function has its own characteristics, and some functions require you to pass certain values. If you pass more parameters than the function requires, the extra values are ignored. If you don't pass a required parameter, the empty parameters are assigned the `undefined` data type, which can cause errors when you export a script. To call a function, it must be in a frame that the playhead has reached.

The top-level built-in Flash functions are listed in the following table.

`Boolean`	`getVersion`	`parseInt`
`escape`	`isFinite`	`String`
`eval`	`isNaN`	`targetPath`
`getProperty`	`Number`	`unescape`
`getTimer`	`parseFloat`	

Note: Because string functions are deprecated, they are not listed in this table.

To call a function, you use the Actions panel in expert mode or normal mode. For more information about these modes, see "Working in expert mode" and "Working in normal mode" under Help > Using Flash.

To call a built-in function:

Choose the Functions category in the Actions toolbox, and double-click a function name to add to a script.

Creating functions

You can define functions to execute a series of statements on passed values. Your functions can also return values. Once a function is defined, it can be called from any Timeline, including the Timeline of a loaded movie.

A well-written function can be thought of as a "black box." If it has carefully placed comments about its input, output, and purpose, a user of the function does not need to understand exactly how the function works internally.

Defining a function

Functions, like variables, are attached to the Timeline of the movie clip that defines them, and you must use a target path to call them. As with variables, you can use the `_global` identifier to declare a global function that is available to all Timelines without using a target path. To define a global function, precede the function name with the identifier `_global`, as in the following:

```
_global.myFunction = function (x) {
   return (x*2)+3;
}
```

To define a Timeline function, use the `function` action followed by the name of the function, any parameters to be passed to the function, and the ActionScript statements that indicate what the function does.

The following is a function named `areaOfCircle` with the parameter `radius`:

```
function areaOfCircle(radius) {
  return Math.PI * radius * radius;
}
```

Note: The keyword `this`, used in a function body, is a reference to the movie clip that the function belongs to.

You can also define a function by creating a *function literal*—an unnamed function that is declared in an expression instead of in a statement. You can use a function literal to define a function, return its value, and assign it to a variable in one expression, as in the following:

```
area = (function() {return Math.PI * radius *radius;})(5);
```

When a function is redefined, the new definition replaces the old definition.

Passing parameters to a function

Parameters are the elements on which a function executes its code. (In this manual, the terms *parameter* and *argument* are interchangeable.) For example, the following function takes the parameters `initials` and `finalScore`:

```
function fillOutScorecard(initials, finalScore) {
  scorecard.display = initials;
  scorecard.score = finalScore;
}
```

When the function is called, the required parameters must be passed to the function. The function substitutes the passed values for the parameters in the function definition. In this example, `scorecard` is the instance name of a movie clip; `display` and `score` are input text fields in the instance. The following function call assigns the value `"JEB"` to the variable `display` and the value `45000` to the variable `score`:

```
fillOutScorecard("JEB", 45000);
```

The parameter `initials` in the function `fillOutScorecard` is similar to a local variable; it exists while the function is called and ceases to exist when the function exits. If you omit parameters during a function call, the omitted parameters are passed as `undefined`. If you provide extra parameters in a function call that are not required by the function declaration, they are ignored.

Using variables in a function

Local variables are valuable tools for organizing code and making it easier to understand. When a function uses local variables, it can hide its variables from all other scripts in the movie; local variables are scoped to the body of the function and are destroyed when the function exits. Any parameters passed to a function are also treated as local variables.

You can also use global and regular variables in a function. However, if you modify global or regular variables, it is good practice to use script comments to document these modifications.

Returning values from a function

Use the `return` action to return values from functions. The `return` action stops the function and replaces it with the value of the `return` action. If Flash doesn't encounter a `return` action before the end of a function, an empty string is returned. For example, the following function returns the square of the parameter `x`:

```
function sqr(x) {
  return x * x;
}
```

Some functions perform a series of tasks without returning a value. For example, the following function initializes a series of global variables:

```
function initialize() {
  boat_x = _root.boat._x;
  boat_y = _root.boat._y;
  car_x = _root.car._x;
  car_y = _root.car._y;
}
```

Calling a user-defined function

You can use a target path to call a function in any Timeline from any Timeline, including from the Timeline of a loaded movie. If a function was declared using the `_global` identifier, you do not need to use a target path to call it.

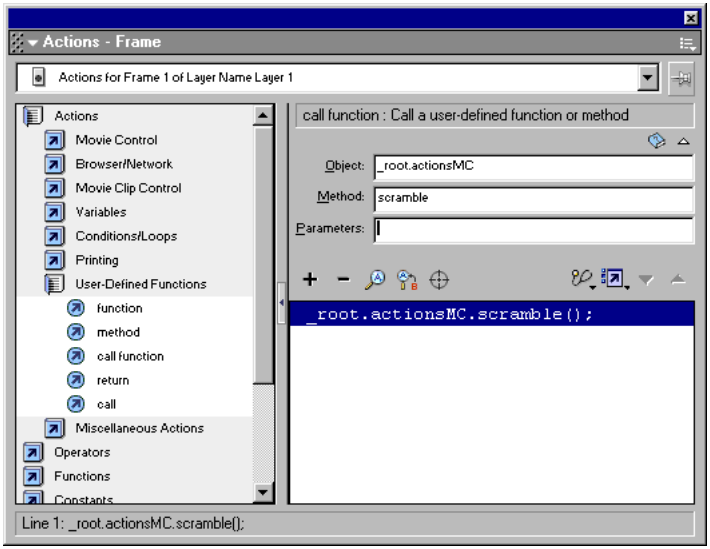
To invoke a function using the Actions panel in normal mode, you use the `call function` action. Pass the required parameters inside parentheses. You can call a function in any Timeline from any Timeline, including a loaded movie. For example, the following statement invokes the function `sqr` in the movie clip `MathLib` on the main Timeline, passes it the parameter 3, and stores the result in the variable `temp`:

```
var temp = _root.MathLib.sqr(3);
```

To call a user-defined function in normal mode:

1. Choose Window > Actions to open the Actions panel.
2. In the Actions toolbox (at the left of the panel), click the Actions folder, then click the User-Defined Functions folder.
3. Double-click the `call function` action.

4 In the Object box, enter the target path of the movie clip in which the function was defined. You can use the Insert Target Path button to enter the target path.



5 In the Method box, enter the name of the function.

6 In the Parameters box, enter the names of parameters, if any, separated by commas.

For information on target paths, see "Writing a target path" on page 230. For more information on each function, see individual entries in the online ActionScript Dictionaryin the Help menu.

To call a function in expert mode:

Enter the target path to the name of the function. Pass any required parameters inside parentheses.

The following example uses an absolute path to call the `initialize` function that was defined on the main Timeline and requires no parameters:

```
_root.initialize();
```

The following example uses a relative path to call the `list` function that was defined in the `functionsClip` movie clip:

```
_parent.functionsClip.list(6);
```

For more information about using the Actions panel, see "Working in normal mode" and "Working in expert mode" under Help > Using Flash.

About built-in objects

You can use built-in Flash objects to access and manipulate certain kinds of information. Most built-in objects have *methods* (functions assigned to an object) that you can call to return a value or perform an action. For example, the Date object returns information from the system clock and the Sound object lets you control sound elements in your movie.

Some built-in objects have properties whose values you can read. For example, the Key object has constant values that represent keys on the keyboard. Each object has its own characteristics and abilities that make it useful in a movie.

The built-in Flash objects are divided into four categories within the Objects folder in the Actions panel: Core, Movie, Client/Server, and Authoring.

- The Core objects are also core objects in the ECMA specification on which ActionScript is based. The ActionScript Core objects are Arguments, Array, Boolean, Date, Function, Math, Number, Object, and String.
- The Movie objects are specific to ActionScript. They are Accessibility, Button , Capabilities, Color, Key, Mouse, MovieClip, Selection, Sound, Stage, System, TextField, and TextFormat.
- The Client/Server objects are ActionScript objects you can use to communicate between a client and a server. They are LoadVars, XML, and XMLSocket.
- The Authoring objects are for customizing the Flash authoring application. They are CustomActions and Live Preview.

Movie clip instances are represented as objects in ActionScript; their default object class is MovieClip. To change the class of movie clips, see "Creating inheritance" on page 241. You can call built-in movie clip methods just as you would call the methods of any other ActionScript object.

For detailed information on each object, see its entry in the online ActionScript Dictionary in the Help menu.

Using a built-in object

Some built-in Flash objects are top-level objects; you can use the methods and properties of a top-level object without creating a new instance of the object. For example, to use the methods and properties of the top-level Math object, you use the name of the built-in object followed by the method or property, as in the following:

```
area = Math.PI * radius * radius;
```

Other built-in objects, like the Date object, require you to create a new instance of the object to use its methods and properties. You use the `new` operator with a constructor function to create an object. (A constructor function is a function that creates a new instance of an object.) The ActionScript built-in objects are prewritten constructor functions. When you create a new instance of a built-in object, all the properties and methods of that object are copied into the instance. This is similar to dragging a movie clip from the library to the Stage. For example, the following statement creates a new Date object called `currentDate` and then calls the `getMinutes` method:

```
currentDate = new Date();
currentMinute = currentDate.getMinutes();
```

In the following code, the object `c` is created from the constructor `Color`:

```
c = new Color(this);
```

Each object that requires a constructor function has a corresponding `new` element in its folder in the Actions panel—for example, `new Color`, `new Date`, `new String`, and so on.

You can also use the object initializer operator (`{}`) to create an object of the generic type Object.

To create an object with the `new` operator in normal mode:

1. Choose Window > Actions to open the Actions panel if it isn't already open.
2. In the Actions toolbox (at the left of the panel), click the Actions folder to open it, then open the Variables folder.
3. Double-click the `set variable` action.
4. Enter an identifier in the Variable box; this is the name of the new object.
5. Click in the Value box to place the insertion point. Then browse in the Actions toolbox to the object you want to create, and double-click `new Date`, `new Color`, and so on.
6. Select the Expression option next to the Value box.

   If you don't select the Expression option, the entire value will be a string literal.

To use the object initializer operator (`{}`) in normal mode:

1. Choose Window > Actions to open the Actions panel if it isn't already open.
2. In the Actions toolbox, click the Actions folder to open it. Click the Variables folder to open it.
3. Double-click the `set variable` action.
4. Enter an identifier in the Variable box; this is the name of the new object.
5. Select the Expression option next to the Value box.
6. In the Value box, enter the property name and value pairs separated by a colon inside the object initializer operator (`{}`).

For example, in this statement the property names are `radius` and `area` and their values are 5 and the value of an expression:

```
myRadius = 5;
myCircle = {radius: myRadius, area:(Math.PI * myRadius * myRadius)};
```

The parentheses cause the expression inside them to evaluate. The returned value is assigned to the variable `area`.

You can also nest array and object initializers, as in this statement:

```
newObject = {name: "John Smith", projects: ["Flash", "Dreamweaver"]};
```

For more information on the Actions panel, see "Writing Scripts with ActionScript" under Help > Using Flash. For detailed information on each object, see its entry in the online ActionScript Dictionary in the Help menu.

Accessing object properties

Use the dot (.) operator to access the value of a property in an object. The name of the object goes on the left side of the dot, and the name of the property goes on the right side. For example, in the following statement, `myObject` is the object and `name` is the property:

```
myObject.name
```

To assign a value to a property in normal mode:

Use the `set variable` action.

```
myObject.name = "Allen";
```

To change the value of a property:

Assign a new value as shown here:

```
myObject.name = "Homer";
```

You can also use the array access operator (`[]`) to access the properties of an object. See "Dot and array access operators" on page 228.

Calling object methods

You can call an object's method by using the dot (.) operator followed by the method. For example, the following example calls the `setVolume` method of the Sound object:

```
mySound = new Sound(this);
mySound.setVolume(50);
```

To call the method of a built-in object in the Actions panel in normal mode, use the `evaluate` action.

To call a method in normal mode:

1. Choose Window > Actions to open the Actions panel if it's not already open.
2. Click the Actions category in the Actions toolbox (at the left of the panel), and then click the Miscellaneous Actions category.
3. Double-click the `evaluate` action.
4. In the Expression box, enter the name of the object, for example, `mySound`.
5. Click the Objects category in the Actions toolbox, and click the category of the object you want to create. When you find the method you want to use, double-click it.

Using the MovieClip object

You can use the methods of the built-in MovieClip object to control movie clip symbol instances on the Stage. The following example tells the movie clip instance `dateCounter` to play:

```
dateCounter.play();
```

For detailed information on the MovieClip object, see its entry in the online ActionScript Dictionary in the Help menu; see also "Working with Movie Clips and Buttons" under Help > Using Flash.

Using the Array object

The Array object is a commonly used built-in ActionScript object that stores its data in numbered properties instead of named properties. An array element's name is called an *index*. Arrays are useful for storing and retrieving certain types of information such as lists of students or a sequence of moves in a game.

You can assign elements of the Array object just as you would assign the property of any object:

```
move[0] = "a2a4";
move[1] = "h7h5";
move[2] = "b1c3";
...
move[100] = "e3e4";
```

To access the second element of the above array, use the expression `move[2]`.

The Array object has a built-in `length` property that is the value of the number of elements in the array. When an element of the Array object is assigned and the element's index is a positive integer such that `index >= length`, `length` is automatically updated to `index + 1`.

For detailed information on the Array object, see its entry in the online ActionScript Dictionary in the Help menu.

About custom objects

You can create a custom object with properties and methods to organize information in your scripts for easier storage and access. After you create an object or class, you can create or *instantiate* copies of that object in a movie.

An object is a complex data type containing zero or more properties and methods. Each property, like a variable, has a name and a value. Properties are attached to the object and contain values that can be changed and retrieved. These values can be of any data type: string, number, Boolean, object, movie clip, or undefined. The following properties are of various data types:

```
customer.name = "Jane Doe";
customer.age = 30;
customer.member = true;
customer.account.currentRecord = 000609;
customer.mcInstanceName._visible = true;
```

The property of an object can also be an object. In line 4 of the previous example, `account` is a property of the object `customer` and `currentRecord` is a property of the object `account`. The data type of the `currentRecord` property is number.

Creating a custom object

To create a custom object, you define a constructor function. A constructor function is always given the same name as the type of object it creates. You can use the keyword `this` inside the body of the constructor function to refer to the object that the constructor creates; when you call a constructor function, Flash passes it `this` as a hidden parameter. For example, the following is a constructor function that creates a circle with the property `radius`:

```
function Circle(radius) {
  this.radius = radius;
}
```

After you define the constructor function you must create a new instance of the object. Use the `new` operator before the name of the constructor function and assign the new instance a variable name. For example, the following code uses the `new` operator to create a new Circle object with a radius of 5, and assigns it to the variable `myCircle`:

```
myCircle = new Circle(5);
```

Note: An object has the same scope as the variable to which it is assigned. See "Scoping a variable" on page 220.

For more information about creating and using objects, see "About built-in objects" on page 236.

Assigning methods to a custom object

You can define the methods of an object inside the object's constructor function. However, this technique is not recommended because it defines the method every time you use the constructor function, as in the following example, which creates the methods `area` and `diameter`:

```
function Circle(radius) {
  this.radius = radius;
  this.area = Math.PI * radius * radius;
  this.diameter = function() {return 2 * this.radius;}
}
```

Each constructor function has a `prototype` property that is created automatically when you define the function. The `prototype` property indicates the default property values for objects created with that function. Each new instance of an object has a `__proto__` property that refers to the `prototype` property of the constructor function that created it. Therefore, if you assign methods to an object's `prototype` property, they are available to any newly created instance of that object. It's best to assign a method to the `prototype` property of the constructor function because it exists in one place and is referenced by new instances of the object (or class). You can use the `prototype` and `__proto__` properties to extend objects so that you can reuse code in an object-oriented manner. (For more information, see "Creating inheritance" on page 241.)

The following procedure shows how to assign an `area` method to a custom Circle object.

To assign a method to a custom object:

1. Define the constructor function `Circle`, as follows:

   ```
   function Circle(radius) {
     this.radius = radius;
   }
   ```

2. Define the `area` method of the Circle object. The `area` method calculates the area of the circle. You can use a function literal to define the `area` method and assign the `area` property to the circle's prototype object, as follows:

   ```
   Circle.prototype.area = function () {
       return Math.PI * this.radius * this.radius;
   };
   ```

3. Create an instance of the Circle object, as follows:

   ```
   var myCircle = new Circle(4);
   ```

4 Call the `area` method of the new `myCircle` object, as follows:

```
var myCircleArea = myCircle.area()
```

ActionScript searches the `myCircle` object for the `area` method. Since the object doesn't have an `area` method, its prototype object `Circle.prototype` is searched for the `area` method. ActionScript finds it and calls it.

Creating inheritance

Inheritance is a means of organizing, extending, and reusing functionality. Subclasses inherit properties and methods from superclasses and add their own specialized properties and methods. For example, reflecting the real world, Bike would be a superclass and MountainBike and Tricycle would be subclasses of the superclass. Both subclasses contain, or *inherit,* the methods and properties of the superclass (for example, `wheels`). Each subclass also has its own properties and methods that extend the superclass (for example, the MountainBike subclass would have a `gears` property). You can use the elements `prototype` and `__proto__` to create inheritance in ActionScript.

All constructor functions have a `prototype` property that is created automatically when the function is defined. The `prototype` property indicates the default property values for objects created with that function. You can use the `prototype` property to assign properties and methods to a class. (For more information, see "Assigning methods to a custom object" on page 240.)

All instances of a class have a `__proto__` property that tells you what object they inherit from. When you use a constructor function to create a new object, the `__proto__` property is set to refer to the `prototype` property of its constructor function.

Inheritance proceeds according to a definite hierarchy. When you call an object's property or method, ActionScript looks at the object to see if such an element exists. If it doesn't exist, ActionScript looks at the object's `__proto__` property for the information (`myObject.__proto__`). If the property is not a property of the object's `__proto__` object, ActionScript looks at `myObject.__proto__.__proto__`, and so on.

The following example defines the constructor function `Bike`:

```
function Bike (length, color) {
  this.length = length;
  this.color = color;
}
```

The following code adds the `roll` method to the Bike class:

```
Bike.prototype.roll = function() {this._x = _x + 20;};
```

Instead of adding a `roll` method to the MountainBike class and the Tricycle class, you can create the MountainBike class with Bike as its superclass:

```
MountainBike.prototype = new Bike();
```

Now you can call the `roll` method of MountainBike, as in the following:

```
MountainBike.roll();
```

Movie clips do not inherit from each other. To create inheritance with movie clips, you can use the `Object.registerClass` method to assign a class other than the MovieClip class to movie clips. See `Object.registerClass` in the online ActionScript Dictionary in the Help menu.

For more information on inheritance, see the `Object.__proto__`, `#initclip`, `#endinitclip`, and `super` entries in the online ActionScript Dictionary in the Help menu.

Using Flash MX ActionScript with older versions of Flash

ActionScript changed considerably with the release of Flash 5 and has become more robust with the release of Flash MX. To take advantage of the full power of ActionScript, you must create content for Flash Player 6. If you need to create content for an earlier version of the Flash Player, you won't be able to use every ActionScript element.

Using Flash MX to create content for Flash Player 5

To use Flash MX to create content for Flash Player 5, set the export version to Flash 5 when you publish your movie.

The strict equality operator (===) and the `switch` action are new to Flash MX and are supported by Flash Player 5. (Flash Player 5 doesn't natively support these elements, but Flash uses appropriate emulation code.)

Using Flash MX to create content for Flash Player 4

To use Flash MX to create content for Flash Player 4, set the export version to Flash 4 when you publish your movie. Flash 4 ActionScript has only one basic primitive data type, which is used for both numeric and string manipulation. When you author a movie for Flash Player 4, you must use the deprecated string operators located in the Deprecated > Operators category in the Actions toolbox.

You can use the following Flash 5 and Flash MX features when you export to the Flash 4 SWF file format:

- The array and object access operator (`[]`)
- The dot operator (`.`)
- Logical operators, assignment operators, and pre-increment and post-increment/decrement operators
- The modulo operator (`%`), and all methods and properties of the Math object

  These operators and functions are not supported natively by Flash Player 4. Flash MX exports them as series approximations, which creates results that are less numerically accurate. In addition, because of the inclusion of series approximations in the SWF file, these functions take up more room in Flash 4 SWF files than they do in Flash 5 or later SWF files.
- The `for`, `while`, `do..while`, `break`, and `continue` actions
- The `print` and `printAsBitmap` actions
- The `switch` action

The following ActionScript features can't be used in movies exported to the Flash Player 4 file format:

Built-in objects (except Math)	`isFinite`	`localToGlobal`	`parseFloat`	`typeof`
Custom functions	`isNaN`	Local variables	`parseInt`	`unescape`
`delete`	`for..in`	Movie clip methods	`_quality`	XML elements
`escape`	`globalToLocal`	Multiple data types	`return`	`_xmouse`
`eval` with dot syntax—for example, `eval("_root.movieclip.variable")`	`hitTest`	`new`	`targetPath`	`_ymouse`

Using Flash MX to open Flash 4 files

Flash 4 ActionScript had only one true data type: string. It used different types of operators in expressions to indicate whether the value should be treated as a string or as a number. In Flash 5 and Flash MX, you can use one set of operators on all data types.

When you use Flash 5 or later to open a file that was created in Flash 4, Flash automatically converts ActionScript expressions to make them compatible with the new syntax. You'll see the following data type and operator conversions in your ActionScript code:

- The = operator in Flash 4 was used for numeric equality. In Flash 5 and Flash MX, == is the equality operator and = is the assignment operator. Any = operators in Flash 4 files are automatically converted to ==.
- Flash automatically performs type conversions to ensure that operators behave as expected. Because of the introduction of multiple data types, the following operators have new meanings:

  +, ==, !=, <>, <, >, >=, <=

  In Flash 4 ActionScript, these operators were always numeric operators. In Flash 5 and Flash MX, they behave differently depending on the data types of the operands. To prevent any semantic differences in imported files, the `Number` function is inserted around all operands to these operators. (Constant numbers are already obvious numbers, so they are not enclosed in `Number`).
- In Flash 4, the escape sequence `\n` generated a carriage return character (ASCII 13). In Flash 5 and Flash MX, to comply with the ECMA-262 standard, `\n` generates a line-feed character (ASCII 10). An `\n` sequence in Flash 4 FLA files is automatically converted to `\r`.
- The `&` operator in Flash 4 was used for string addition. In Flash 5 and Flash MX, `&` is the bitwise AND operator. The string addition operator is now called `add`. Any `&` operators in Flash 4 files are automatically converted to `add` operators.
- Many functions in Flash 4 did not require closing parentheses, for example, `Get Timer`, `Set Variable`, `Stop`, and `Play`. To create consistent syntax, the `getTimer` function and all actions now require closing parentheses. These parentheses are automatically added during the conversion.
- In Flash 5 and Flash MX, when the `getProperty` function is executed on a movie clip that doesn't exist, it returns the value `undefined`, not 0. The statement `undefined == 0` is `false` in ActionScript. Flash fixes this problem when converting Flash 4 files by introducing `Number` functions in equality comparisons. In the following example, `Number` forces `undefined` to be converted to 0 so the comparison will succeed:

```
getProperty("clip", _width) == 0
Number(getProperty("clip", _width)) == Number(0)
```

Note: If you used any Flash 5 or Flash MX keywords as variable names in your Flash 4 ActionScript, the syntax returns an error in Flash MX. To fix this, rename your variables in all locations. See "Keywords" on page 216.

About slash syntax

Slash syntax was used in Flash 3 and 4 to indicate the target path of a movie clip or variable. This syntax is still supported by Flash Player 6, but its use is not recommended. However, if you are creating content intended specifically for Flash Player 4, you need to use slash syntax.

In slash syntax, slashes are used instead of dots; also, to indicate a variable, you precede it with a colon:

```
myMovieClip/childMovieClip:myVariable
```

To write the same target path in dot syntax, which is supported by Flash 5 and later, you would use the following code:

```
myMovieClip.childMovieClip.myVariable
```

Slash syntax was most commonly used with the `tellTarget` action, whose use is also no longer recommended. The `with` action is now preferred over `tellTarget` because it is more compatible with dot syntax. For detailed information on these actions, see the online ActionScript Dictionary in the Help menu.

CHAPTER 13
Working with Movie Clips and Buttons

A movie clip is like a mini-movie in Macromedia Flash MX: it has its own Timeline and properties. A movie clip symbol in the library can be used multiple times in a Flash document; each use is called an *instance* of the movie clip. To distinguish instances from each other, you must assign each instance a name. Movie clip instances can be nested inside each other to create a hierarchy.

Each movie clip has a position in the hierarchical tree of Timelines called the *display list*. Movies that are loaded into the Flash Player with the `loadMovie` action also have independent Timelines and a position in the display list. You can use ActionScript to send messages between movie clips so that they can control each other. For example, an action on the last frame of one movie clip's Timeline could tell another movie clip to play.

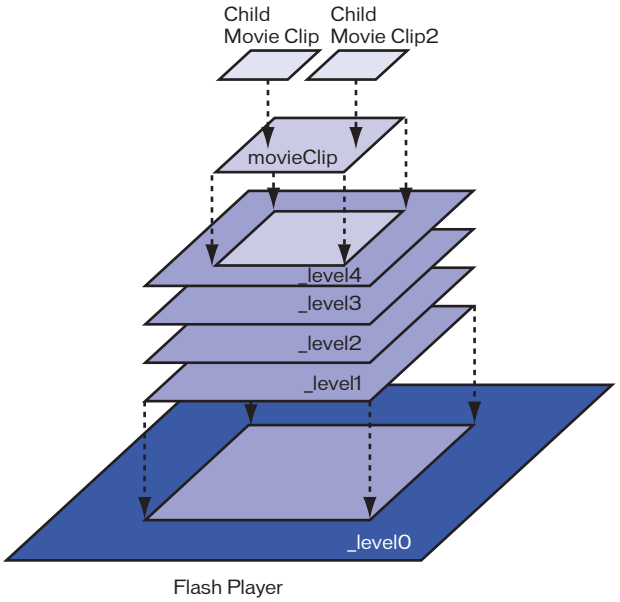
You control movie clips using actions and methods of the MovieClip object. To control a movie clip, you must address it by using a *target path*, which indicates its unique location in the display list. You can use the methods of the MovieClip object to drag a movie clip, dynamically add a movie clip to a document, turn a movie clip into a mask, and draw lines and fills on the Stage.

Just like each movie clip instance, each button instance is an ActionScript object with its own properties and methods. You can give a button an instance name and manipulate it with ActionScript. Each movie clip and button in a Flash document are objects with properties and methods that can be changed by ActionScript to create complex, nonlinear animation and powerful interactivity.

About multiple Timelines

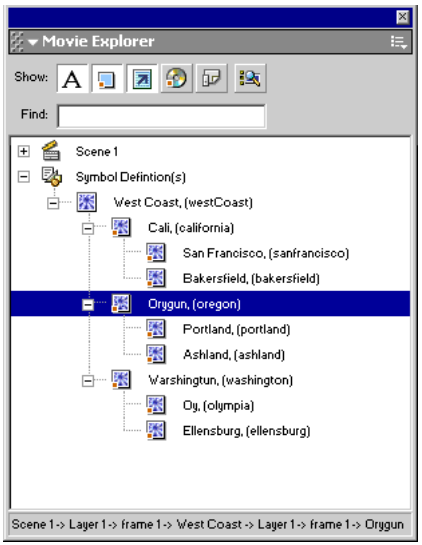
The Flash Player has a stacking order of levels. Every Flash movie has a main Timeline located at level 0 in the Flash Player. You can use the `loadMovie` action to load other Flash movies (SWF files) into the Flash Player at different levels. If you load movies into levels above level 0, the movies lie on top of each other like drawings on transparent paper; where there is no content on the Stage, you can see through to the content on lower levels. If you load a movie into level 0, it replaces the main Timeline. Each movie loaded into a level of the Flash Player has its own Timeline.

Flash movies at any level can have movie clip instances on their Timelines. Each movie clip instance also has a Timeline and can contain other movie clips that also have Timelines. In the Flash Player, levels and Timelines are arranged hierarchically so that you can organize and easily control the objects in your movie.



The hierarchy of levels and movie clips in the Flash Player

In Flash, this hierarchy of levels and movie clips is called the *display list*. When you author in Flash, you can view the display list in the Movie Explorer; when you play the movie in test mode, the stand-alone Flash Player, or a Web browser, you can view the display list in the Debugger.



The Movie Explorer shows the display list.

Depending on their locations in the display list, Timelines have specific relationships with each other. A child Timeline nested inside another Timeline is affected by changes made to the parent Timeline. For example, if `portland` is a child of `oregon` and you change the `_xscale` property of `oregon`, `portland` will also scale.

Timelines can also send messages to each other. For example, an action on the last frame of one movie clip can tell another movie clip to play.

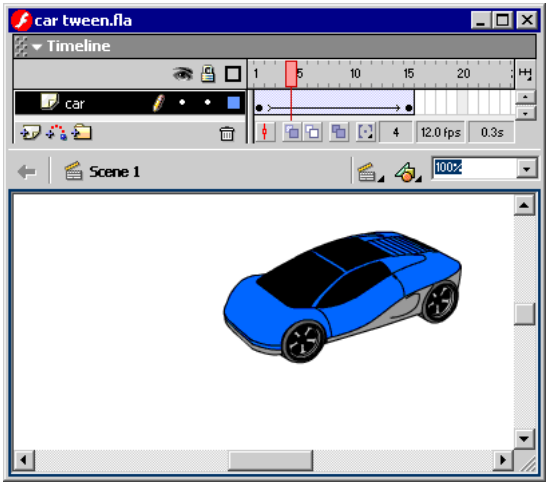
About movie clip hierarchy

When you place a movie clip instance on another movie clip's Timeline, the placed movie clip is the *child* and the other movie clip is the *parent*. The parent instance contains the child instance. The root Timeline for each level is the parent of all the movie clips on its level, and because it is the topmost Timeline, it has no parent.

The parent-child relationships of movie clips are hierarchical. To understand this hierarchy, consider the hierarchy on a computer: the hard drive has a root directory (or folder) and subdirectories. The root directory is analogous to the main Timeline of a Flash movie: it is the parent of everything else. The subdirectories are analogous to movie clips.

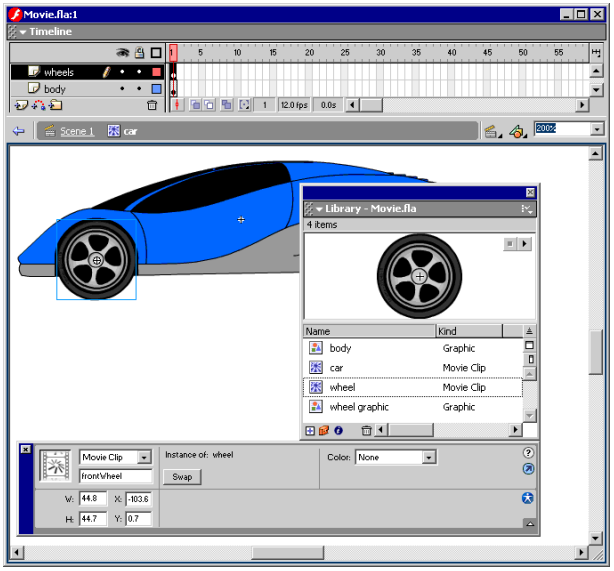
You can use the movie clip hierarchy in Flash to organize related visual objects. Any change you make to a parent movie clip is also performed on its children.

For example, you could create a Flash movie of a car that moves across the Stage. You could use a movie clip symbol to represent the car and set up a motion tween to move it across the Stage.



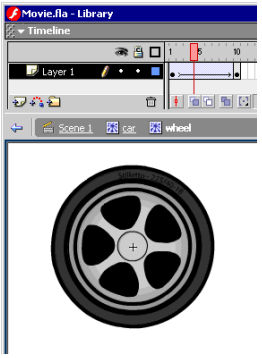
A motion tween moves the car movie clip on the main Timeline.

To add wheels that rotate, you create a movie clip for a car wheel, and create two instances of this movie clip, named `frontWheel` and `backWheel`. Then you place the wheels on the car movie clip's Timeline—not on the main Timeline. As children of `car`, `frontWheel` and `backWheel` are affected by any changes made to `car`; they will move with the car as it tweens across the Stage.



The wheel instances are placed on the Timeline of the car parent movie clip.

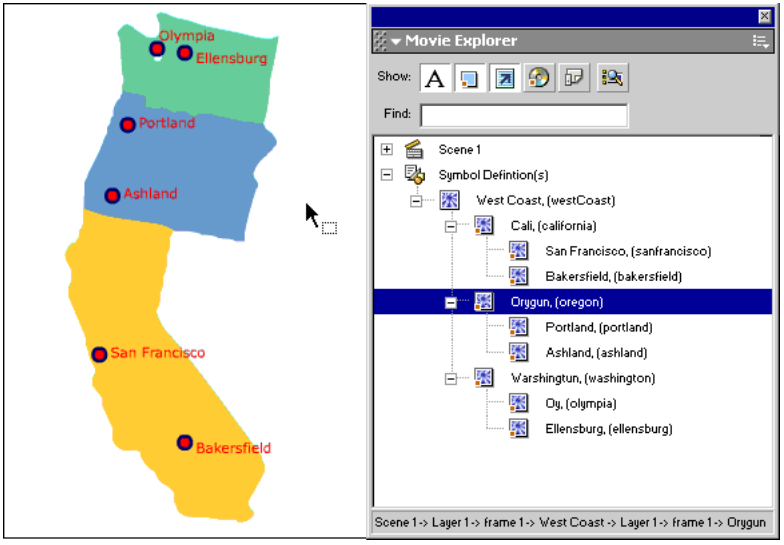
To make both wheel instances spin, you set up a motion tween that rotates the wheel symbol. Even after you change `frontWheel` and `backWheel`, they continue to be affected by the tween on their parent movie clip, `car`; the wheels spin, but they also move with the parent movie clip `car` across the Stage.



The wheel symbol in symbol-editing mode

About absolute and relative target paths

You can use actions to send messages from one Timeline to another. The Timeline that contains the action is called the *controlling Timeline*, and the Timeline that receives the action is called the *target Timeline*. For example, there could be an action on the last frame of one Timeline that tells another Timeline to play. To refer to a target Timeline, you must use a target path, which indicates the location of a movie clip in the display list.



The display list of movie clips in authoring mode

The hierarchy of movie clips in this display list is as follows:

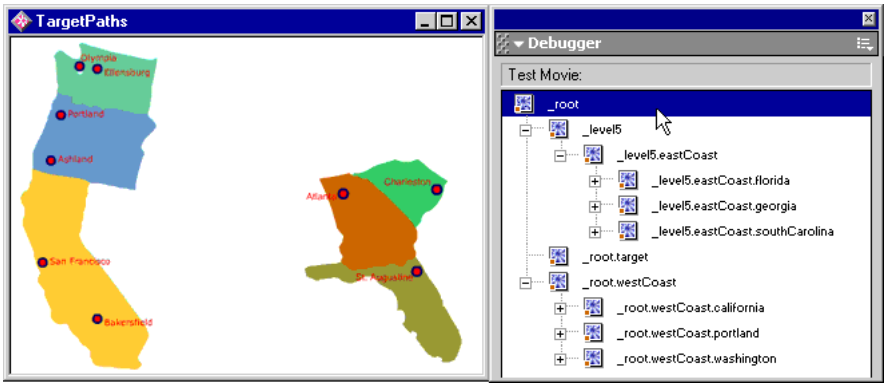
```
_level0
     westCoast
          california
               sanfrancisco
               bakersfield
          oregon
               portland
               ashland
          washington
               olympia
               ellensburg
```

Just as on a Web server, each Timeline in Flash can be addressed in two ways: with an absolute path or a relative path. The absolute path of an instance is always a full path from a level name, regardless of which Timeline calls the action; for example, the absolute path to the instance `california` is `_level0.westCoast.california`. A relative path is different when called from different locations; for example, the relative path to `california` from `sanfrancisco` is `_parent`, but from `portland`, it's `_parent._parent.california`.

An absolute path starts with the name of the level into which the movie is loaded and continues through the display list until it reaches the target instance. You can also use the alias `_root` to refer to the topmost Timeline of the current level. For example, an action in the movie clip `california` that refers to the movie clip `oregon` could use the absolute path `_root.westCoast.oregon`.

The first movie to be opened in the Flash Player is loaded at level 0. You must assign each additional loaded movie a level number. When you use an absolute reference in ActionScript to reference a loaded movie, use the form `_levelX`, where *X* is the level number into which the movie is loaded. For example, the first movie opened in the Flash Player is called `_level0`; a movie loaded into level 3 is called `_level3`.

In the following example, two movies have been loaded into the player: TargetPaths.swf at level 0, and EastCoast.swf at level 5. The levels are indicated in the Debugger, with level 0 indicated as `_root`.



To communicate between movies on different levels, you must use the level name in the target path. For example, the `portland` instance would address the `atlanta` instance as follows:

```
_level5.georgia.atlanta
```

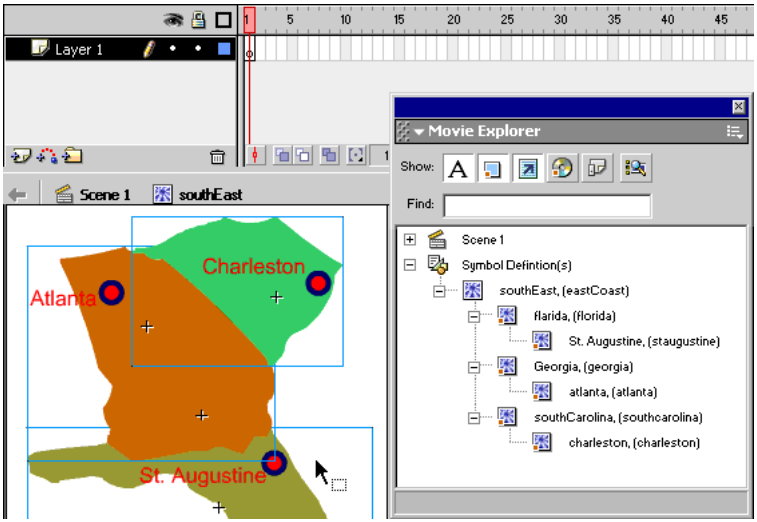
You can use the alias `_root` to refer to the main Timeline of the current level. For the main Timeline, the `_root` alias stands for `_level0` when targeted by a clip also on `_level0`. For a movie loaded into `_level5`, `_root` is equal to `_level5` when targeted by a movie clip also on level 5. For example, because `southcarolina` and `florida` are both loaded into the same level, an action called from the instance `southcarolina` could use the following absolute path to target the instance `florida`:

```
_root.eastCoast.florida
```

A relative path depends on the relationship between the controlling Timeline and the target Timeline. Relative paths can address targets only within their own level of the Flash Player. For example, you can't use a relative path in an action on `_level0` that targets a Timeline on `_level5`.

In a relative path, use the keyword `this` to refer to the current Timeline in the current level; use the alias `_parent` to indicate the parent Timeline of the current Timeline. You can use the `_parent` alias repeatedly to go up one level in the movie clip hierarchy within the same level of the Flash Player. For example, `_parent._parent` controls a movie clip up two levels in the hierarchy. The topmost Timeline at any level in the Flash Player is the only Timeline with a `_parent` value that is undefined.

In the following example, each city (`charleston`, `atlanta`, and `staugustine`) is a child of a state instance, and each state (`southcarolina`, `georgia`, and `florida`) is a child of the `eastCoast` instance.



An action on the Timeline of the instance `charleston` could use the following target path to target the instance `southcarolina`:

```
_parent
```

To target the instance `eastCoast` from an action in `charleston`, you could use the following relative path:

```
_parent._parent
```

To target the instance `atlanta` from an action on the Timeline of `charleston`, you could use the following relative path:

```
_parent._parent.georgia.atlanta
```

Relative paths are useful for reusing scripts. For example, you could attach a script to a movie clip that magnifies its parent by 150%, as follows:

```
onClipEvent (load) {
  _parent._xscale = 150;
  _parent._yscale = 150;
}
```

You could then reuse this script by attaching it to any movie clip instance.

Whether you use an absolute or relative path, you identify a variable on a Timeline or a property of an object with a dot (`.`) followed by the name of the variable or property. For example, the following statement sets the variable `name` in the instance `form` to the value `"Gilbert"`:

```
_root.form.name = "Gilbert";
```

Writing target paths

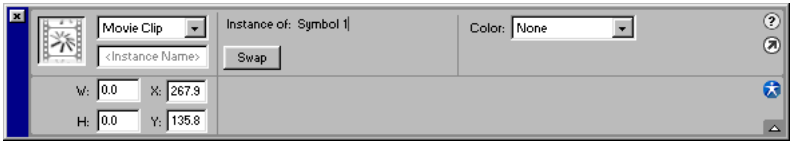
To control a movie clip, loaded movie, or button, you must specify a target path. Before you can specify a target path to a movie clip or button, you must assign it an instance name. A loaded movie doesn't require an instance name, because you use its level number as an instance name (for example, `_level5`).

You can specify a target path in several different ways:

- Use the Insert Target Path button (and dialog box) in the Actions panel.
- Enter the target path manually.
- Create an expression that evaluates to a target path. You can use the built-in functions `targetPath` and `eval`.

To assign an instance name:

1. Select a movie clip or button on the Stage.
2. Enter an instance name in the Property inspector.



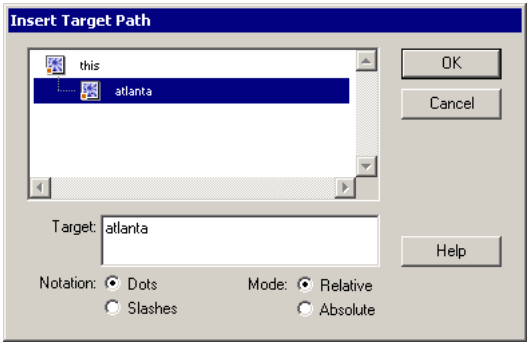
To insert a target path using the Insert Target Path dialog box:

1. Select the movie clip, frame, or button instance to which you want to assign the action.

   This will be the controlling Timeline.
2. Choose Window > Actions to display the Actions panel if it's not already open.
3. In the Actions toolbox (at the left of the panel), choose an action or method that requires a target path.
4. Click the parameter box or location in the script where you want to insert the target path.

5 Click the Insert Target Path button above the Script pane.

6 In the Insert Target Path dialog box, choose a syntax: Dots (the default) or Slashes.



7 Choose Absolute or Relative for the target path mode.

See "About absolute and relative target paths" on page 249.

8 Select a movie clip in the Insert Target Path display list.

9 Click OK.

To insert a target path manually:

Follow steps 1-4 above and enter an absolute or relative target path in the Actions panel.

To use an expression as a target path:

1 Follow steps 1-3 above.

2 Do one of the following:

- Enter an expression that evaluates to a target path in a parameter box.
- Click to place the insertion point in the script. Then, in the Functions category of the Actions toolbox, double-click the `targetPath` function.

  The `targetPath` function converts a reference to a movie clip into a string.
- Click to place the insertion point in the script. Then, in the Functions category of the Actions toolbox, choose the `eval` function.

  The `eval` function converts a string to a movie clip reference that can be used to call methods such as `play`.

  The following script assigns the value 1 to the variable `i`. It then uses the `eval` function to create a reference to a movie clip instance and assigns it to the variable `x`. The variable `x` is now a reference to a movie clip instance and can call the MovieClip object methods, as in the following:

```
i = 1;
x = eval("mc"+i);
x.play();
// this is equivalent to mc1.play();
```

  You can also use the `eval` function to call methods directly, as in the following:

```
eval("mc" + i).play();
```

Using actions and methods to control movie clips

You can use ActionScript actions or the methods of the MovieClip object to perform tasks on movie clips. Some MovieClip methods perform the same tasks as the actions of the same name; other MovieClip object methods, such as `hitTest` and `swapDepths`, don't have corresponding actions.

When an action and a method offer similar behaviors, you can choose to control movie clips by using either one. The choice depends on your preference and familiarity with writing scripts in ActionScript. Whether you use an action or a method, the target Timeline must be loaded in the Flash Player when the action or method is called.

The following actions target movie clips: `loadMovie`, `unloadMovie`, `loadVariables`, `setProperty`, `startDrag`, `duplicateMovieClip`, and `removeMovieClip`. To use these actions, you must enter a target path for the action's *`target`* parameter to indicate the target of the action.

The following MovieClip methods can control movie clips or loaded levels and do not have equivalent actions: `attachMovie`, `createEmptyMovieClip`, `createTextField`, `getBounds`, `getBytesLoaded`, `getBytesTotal`, `getDepth`, `globalToLocal`, `localToGlobal`, `hitTest`, `setMask`, `swapDepths`.

To use a method, invoke it by using the target path of the instance name, a dot, and then the method name and parameters, as in the following statements:

```
myMovieClip.play();
parentClip.childClip.gotoAndPlay(3);
```

In the first statement, the `play` method moves the playhead in the `myMovieClip` instance. In the second statement, the `gotoAndPlay` method sends the playhead in `childClip` (which is a child of the instance `parentClip`) to frame 3 and continues to move the playhead.

Actions that control a Timeline have a *`target`* parameter that allows you to specify the target path to the instance that you want to control. For example, in the following script the `startDrag` action targets the `customCursor` instance and makes it draggable:

```
on(press){
  startDrag("customCursor");
}
```

The following example illustrates the difference between using a method and using an action. Both statements duplicate the instance `myMovieClip`, name the new clip `newClip`, and place it at a depth of 5.

```
myMovieClip.duplicateMovieClip("newClip", 5);
duplicateMovieClip("myMovieClip", "newClip", 5);
```

For more information about these actions and methods, see the online ActionScript Dictionary in the Help menu.

Calling multiple methods on a single movie clip

You can use the `with` action to address a movie clip once, and then execute a series of methods on that clip. The `with` action works on all ActionScript objects (for example, Array, Color, and Sound), not just movie clips.

The `with` action takes an object as a parameter. The object you specify is added to the end of the current target path. All actions nested inside a `with` action are carried out inside the new target path, or scope. For example, in the following script, the `with` action is passed the object `donut.hole` to change the properties of `hole`:

```
with (donut.hole){
  _alpha = 20;
  _xscale = 150;
  _yscale = 150;
}
```

It is as if the statements inside the `with` action were called from the Timeline of the `hole` instance. The above code is equivalent to the following:

```
donut.hole._alpha = 20;
donut.hole._xscale = 150;
donut.hole._yscale = 150;
```

The above code is also equivalent to the following:

```
with (donut){
  hole._alpha = 20;
  hole._xscale = 150;
  hole._yscale = 150;
}
```

Loading and unloading additional movies

To play additional movies without closing the Flash Player, or to switch movies without loading another HTML page, you can use the `loadMovie` action or method. You can also use `loadMovie` to send variables to a CGI script, which generates a SWF file as its CGI output. When you load a movie, you can specify a level or movie clip target into which the movie will load. If you load a movie into a target, the loaded movie inherits the properties of the targeted movie clip. Once the movie is loaded, you can change those properties.

The `unloadMovie` action and method remove a movie previously loaded by `loadMovie`. Explicitly unloading movies with `unloadMovie` ensures a smooth transition between movies and may decrease the memory required by the Flash Player.

Use the `loadMovie` action to do any of the following:

- Play a sequence of banner ads that are SWF files by placing a `loadMovie` action at the end of each SWF file to load the next movie.
- Develop a branching interface that lets the user choose among several different SWF files.
- Build a navigation interface with navigation controls in level 0 that load other levels. Loading levels produces smoother transitions than loading new HTML pages in a browser.

Loading images and sounds dynamically

You can use the `loadMovie` action or method to load JPEG image files into a Flash movie as it plays. You can use the `loadSound` method of the Sound object to load MP3 sound files into a Flash movie as it plays.

When you load an image, the upper left corner of the image is placed on the registration point of the movie clip. Because this registration point is often the center of the movie clip, the loaded image may not appear centered. Also, when you load an image to a root Timeline, the upper left corner of the image is placed on the upper left corner of the Stage. The loaded image inherits rotation and scaling from the movie clip, but the original content of the movie clip is removed.

For more information, see the `loadMovie` and `loadSound` entries in the online ActionScript Dictionary in the Help menu.

Changing movie clip position and appearance

To change the properties of a movie clip as it plays, write a statement that assigns a value to a property or use the `setProperty` action. For example, the following code sets the rotation of instance `mc` to 45:

```
mc._rotation = 45;
```

This is equivalent to the following code, which uses the `setProperty` action:

```
setProperty("mc", _rotation, 45);
```

Some properties, called *read-only properties,* have values that you can read but not set. (These properties are specified as read-only in their ActionScript Dictionary entries.) The following are read-only properties: `_currentframe`, `_droptarget`, `_framesloaded`, `_parent`, `_target`, `_totalframes`, `_url`, `_xmouse`, and `_ymouse`.

You can write statements to set any property that is not read-only. The following statement sets the `_alpha` property of the movie clip instance `wheel`, which is a child of the `car` instance:

```
car.wheel._alpha = 50;
```

In addition, you can write statements that get the value of a movie clip property. For example, the following statement gets the value of the `_xmouse` property on the current level's Timeline and sets the `_x` property of the `customCursor` instance to that value:

```
onClipEvent(enterFrame){
  customCursor._x = _root._xmouse;
}
```

This is equivalent to the following code, which uses the `getProperty` function:

```
onClipEvent(enterFrame){
  customCursor._x = getProperty(_root, _xmouse);
}
```

The `_x`, `_y`, `_rotation`, `_xscale`, `_yscale`, `_height`, `_width`, `_alpha`, and `_visible` properties are affected by transformations on the movie clip's parent, and transform the movie clip and any of the clip's children. The `_focusrect`, `_highquality`, `_quality`, and `_soundbuftime` properties are global; they belong only to the level 0 main Timeline. All other properties belong to each movie clip or loaded level.

For a list of movie clip properties, see the MovieClip entry in the online ActionScript Dictionary in the Help menu.

Dragging movie clips

You can use the `startDrag` action or method to make a movie clip draggable while a movie is playing. For example, you can make a draggable movie clip for games, drag-and-drop functions, customizable interfaces, scroll bars, and sliders.

A movie clip remains draggable until explicitly stopped by `stopDrag`, or until another movie clip is targeted with `startDrag`. Only one movie clip can be dragged at a time.

To create more complicated drag-and-drop behavior, you can evaluate the `_droptarget` property of the movie clip being dragged. For example, you might examine the `_droptarget` property to see if the movie was dragged to a specific movie clip (such as a "trash can" movie clip) and then trigger another action. For detailed information about `startDrag`, see its entry in the online ActionScript Dictionary in the Help menu.

Duplicating and removing movie clips

To duplicate or remove movie clip instances as a movie is playing, use `duplicateMovieClip` or `removeMovieClip`, respectively. The `duplicateMovieClip` action and method dynamically create a new instance of the movie clip, assign it a new instance name, and give it a depth. A duplicated movie clip always starts at frame 1 even if the original movie clip was on another frame when duplicated, and is always on top of all previously defined movie clips placed on the Timeline.

To delete a movie clip you created with `duplicateMovieClip`, use `removeMovieClip`. Duplicated movie clips also are removed if the parent movie clip is deleted.

For more information, see `duplicateMovieClip` and `removeMovieClip` in the online ActionScript Dictionary in the Help menu.

Dynamically adding a movie clip or sound to the Stage

To retrieve a copy of a movie clip or sound from the library and play it as part of your movie, you use the `attachMovie` method of the MovieClip object or the `attachSound` method of the Sound object. The `attachMovie` method loads a movie clip as a child of the clip that loads it and plays it as the movie runs. The `attachSound` method attaches a sound to an instance of the Sound object.

To use ActionScript to attach a movie clip or sound from the library, you must assign a unique linkage identifier to the movie clip or sound. You can assign this name in the Linkage Properties dialog box.

When a movie plays, Flash loads all movie clips and sounds that are added with `attachMovie` or `attachSound` before the first frame of the movie. This can create a delay before the first frame plays. When you assign a linkage identifier to an element, you can also specify whether this content should be added before the first frame. If it isn't added in the first frame, you must include an instance of it in some other frame of the movie; if you don't, the element will not be exported to the SWF file.

To name a movie clip:

1. Choose Window > Library to open the Library panel.
2. Select a movie clip in the Library panel.
3. In the Library panel, choose Linkage from the Library panel options menu.

   The Linkage Properties dialog box appears.
4. For Linkage, select Export for ActionScript.

5 For Identifier, enter an ID for the movie clip.

6 If you don't want the movie clip or sound to load before the first frame, deselect the Export in First Frame option.

7 Click OK.

To attach a movie clip to another movie clip:

1 With the Actions panel open, select a frame in the Timeline.

2 In the Actions toolbox (at the left of the Actions panel), click the Objects category, the Movie category, and the MovieClip category, and double-click the `attachMovie` method.

3 For the *`object`* parameter, enter the instance name of a movie clip on the Stage.

4 Enter values for the following parameters:

- For `idName`, specify the identifier you entered in the Linkage Properties dialog box.
- For `newName`, enter an instance name for the attached clip so that you will be able to target it.
- For `depth`, enter the level at which the duplicate movie will be attached to the movie clip. Each attached movie has its own stacking order, with level 0 as the level of the originating movie. Attached movie clips are always on top of the original movie clip. Here is an example:

```
myMovieClip.attachMovie("calif", "california", 10);
```

Dynamically creating an empty movie clip

To create an empty movie clip on the Stage while a movie plays, use the `createEmptyMovieClip` method of the MovieClip object. This method creates a movie clip as a child of the clip that calls the method. The registration point for a newly created empty movie clip is the upper left corner. Although the `createEmptyMovieClip` method behaves similarly to `attachMovie`, you don't need to provide a linkage identifier because you aren't adding a symbol from the library.

To create an empty movie clip:

1 Select a frame, button, or movie clip to which to assign the action.

2 Choose Window > Actions to open the Actions panel if it isn't already open.

3 In the Actions toolbox (at the left of the Actions panel), click the Objects category, the Movie category, the MovieClip category, and the Methods category, and double-click `createEmptyMovieClip`.

4 For the *`object`* parameter, enter the instance name of a movie clip on the Stage, or click the Insert Target Path button to browse to an instance.

5 Enter values for the following parameters:

- For `instanceName`, specify an identifier.
- For `depth`, enter the level at which the duplicate movie will be attached to the movie clip. Each created movie has its own stacking order, with level 0 as the level of the originating movie. Newly created movie clips are always on top of the original movie clip. Here is an example:

```
myMovieClip.createEmptyMovieClip("newMC", 10);
```

Drawing shapes with ActionScript

You can use methods of the MovieClip object to draw lines and fills on the Stage as the movie plays. This allows you to create drawing tools for users and to draw shapes in the movie in response to events. The drawing methods are `beginFill`, `beginGradientFill`, `clear`, `curveTo`, `endFill`, `lineTo`, `lineStyle`, and `moveTo`.

You can use the drawing methods with any movie clip. However, if you use the drawing methods with a movie clip that was created in authoring mode, the drawing methods execute before the clip is drawn. In other words, content that is created in authoring mode is drawn on top of content drawn with the drawing methods.

You can use movie clips with drawing methods as masks; however, as with all movie clip masks, strokes are ignored.

To draw a shape:

1. Use the `createEmptyMovieClip` method to create an empty movie clip on the Stage.

   The new movie clip is a child of an existing movie clip or of the main Timeline, as in the following example:

   ```
   _root.createEmptyMovieClip ("triangle", 1);
   ```

2. Use the empty movie clip to call drawing methods.

   The following example draws a triangle with 5-point magenta lines and no fill:

   ```
   with (_root.triangle) {
        lineStyle (5, 0xff00ff, 100);
        moveTo (200, 200);
        lineTo (300, 300);
        lineTo (100, 300);
        lineTo (200, 200);
   }
   ```

For detailed information on these methods, see their entries in the online ActionScript Dictionary in the Help menu.

Using movie clips as masks

You can use a movie clip as a mask to create a hole through which the contents of another movie clip are visible. The mask movie clip plays all the frames in its Timeline, just like a regular movie clip. You can make the mask movie clip draggable, animate it along a motion guide, use separate shapes within a single mask, or resize a mask dynamically. You can also use ActionScript to turn a mask on and off while a movie plays.

You cannot use a mask to mask another mask. You cannot set the `_alpha` property of a mask movie clip. Only fills are used in a movie clip that is used as a mask; strokes are ignored.

To create a mask:

1. On the Stage, choose a movie clip to be masked.
2. In the Property inspector, enter an instance name for the movie clip, such as `image`.
3. Create a movie clip to be a mask. Give it an instance name in the Property inspector, such as `mask`.
4. Select frame 1 in the Timeline.
5. Choose Window > Actions to open the Actions panel if it isn't already open.

6 In the Actions toolbox (at the left of the panel), click the Objects category, the Movie category, the MovieClip category, and the Methods category, and double-click `setMask`.

7 In the parameters area, enter the instance name of the mask movie clip.

The code should look like this:

```
image.setMask(mask);
```

For complete information on the `setMask` method, see the online ActionScript Dictionary in the Help menu.

Handling events with ActionScript

Certain events occur while a movie plays. Some of these events always occur in the same order (for example, `load`, `enterFrame`, `unload`), and some occur when a user initiates them (for example, `mouseDown`, `mouseUp`, `mouseMove`, `keyDown`, and `keyUp`). One event, `data`, occurs when data is received by the movie from an external source. You can use these events to cause scripts to run; this is called *triggering* a script. Your response to the event is called *event handling*. For example, you could write a script that tells a movie clip to play. If you want the movie clip to play when it receives information from an external text file, you could use the `data` event to trigger the script. There are two ways to handle events using ActionScript: you can use the `onClipEvent` and `on` event handler actions, or you can use the event handler methods of the MovieClip and Button objects.

In the Actions toolbox, the `onClipEvent` and `on` event handler actions are in the Movie Control category within the Actions folder. When you use one of these actions, you pass an event to the action as a parameter (for example, `on(press)`). In the Actions toolbox, the MovieClip and Button objects have Events categories that contain methods that correspond to each movie clip and button event, such as `onLoad`, `onEnterFrame`, `onUnload`, `onMouseDown`, `onMouseUp`, `onMouseMove`, `onKeyDown`, `onKeyUp`, and `onData`. You can use these methods to define a function that runs when the event occurs. The event handler methods don't conflict with their corresponding actions; both events cause their scripts to run.

You can attach `onClipEvent` and `on` actions only to movie clip instances that have been placed on the Stage in authoring mode. You cannot attach `onClipEvent` or `on` actions to movie clip instances that are created at runtime using the `attachMovie` method. For example, the following code is attached to a movie clip instance on the Stage:

```
onClipEvent(onLoad){
  trace("loaded");
}
```

When you use the MovieClip or Button event handler methods, you don't have to assign the script to the instance whose event you are handling; for example, you can assign the script to a frame. This allows you to control movie clips and buttons placed on the Stage in authoring mode, as well as movie clips that ActionScript creates while a movie plays. To use the event handler methods, you assign a function directly to an instance. The function executes when the event specified by the method occurs. For example, the following code triggers the `trace` action when the instance `mc` loads:

```
mc.onLoad = function (){
  trace("loaded");<<Loc: semicolon added here --IMD>>
};
```

For more information about using the `onClipEvent` and `on` event handler actions, see "Assigning actions to a movie clip" on page 201 and "Assigning actions to a button" on page 200. For detailed information about each MovieClip event handler method, see the MovieClip object entry in the online ActionScript Dictionary in the Help menu.

Using movie clip event handler methods to trigger scripts

You can use the methods in the Events category of the MovieClip object to handle movie clip events. You must define a function and assign it to the event handler method. Without a function assigned to it, the event handler method has no effect on the movie.

You can either call an event handler method from the instance of the movie clip whose event you want to handle, or create a new ActionScript class and define the methods in the prototype object of the class. (For more information, see "Defining event handler methods in the prototype object" on page 264.)

To use a movie clip event handler method to trigger a script:

1. On the Stage, select the movie clip whose event you want to handle.
2. Enter an instance name in the Property inspector.
3. Select a frame, button, or movie clip to which to attach the method.
4. Choose Window > Actions to open the Actions panel if it isn't already open.
5. In the Actions toolbox (at the left of the panel), click the Objects category, then click the Movie category, the MovieClip category, and the Events category, and double-click one of the MovieClip event handler methods.
6. Enter values for the following parameters:

- For the *`object`* parameter, enter the target path for the movie clip whose event you want to handle.
- Pass any parameters needed by the function you will define. (In normal mode, enter these parameters in the Parameters text box.)

7. Add actions inside the function to define the function.

   The following code defines a function for the `onPress` method of the instance `mc` that sets the `_alpha` property of `mc` when `mc` loads:

```
mc.onPress = function() {
    this._alpha = 50;
};
```

Note: The keyword `this` refers to the instance that calls the event handler method. In this example, the instance is `mc`.

Using button event handler methods to trigger scripts

Just as a set of events is associated with movie clip symbols, a set of events is also associated with button symbols. You can use button event handler methods with button instances. (You can also use button events with movie clips; see "Using button events with movie clips to trigger scripts" on page 262.)

You can either call an event handler method from the instance of the button whose event you want to handle, or create a new ActionScript class and define the methods in the prototype object of the class. For information about defining a method in the prototype object, see "Defining event handler methods in the prototype object" on page 264.

When you use an event handler method with a button, the keyword `this` refers to the button instance that calls the method. For example, the following code sends `_level0.myButton` to the Output window:

```
myButton.onPress = function() {
  trace(this);
}
```

To use a button event handler method to trigger a script:

1. On the Stage, select the button instance whose event you want to handle.
2. Enter an instance name in the Property inspector.
3. Select a frame, button, or movie clip to which to attach the method.
4. Choose Window > Actions to open the Actions panel if it isn't already open.
5. In the Actions toolbox (at the left of the panel), click the Objects category, then click the Movie category, the MovieClip category, and the Events category, and double-click one of the MovieClip event handler methods.
6. Enter values for the following parameters:

- For the *`object`* parameter, enter the target path for the button whose event you want to handle.
- Pass any parameters needed by the function you will define. (In normal mode, enter these parameters in the Parameters text box.)

7. Add actions inside the function to define the function.

   The following code defines a function for the `onPress` method of the instance `myButton` that triggers a `trace` action:

```
myButton.onPress = function() {
    trace("onPress called!");
};
```

Using button events with movie clips to trigger scripts

You can use button events with button instances, but you can also use them with movie clip instances to create button movie clips. Button movie clips combine the power of movie clips with the control of button events. You can turn a movie clip into a button movie clip by assigning an `on` handler to the movie clip instance, or by defining button event handler methods for an instance. You can also create a new class and define event handler methods in the prototype object of that class. (For information about defining methods in the prototype object, see "Defining event handler methods in the prototype object" on page 264.)

All button events are triggered by user interaction: `press`, `release`, `releaseOutside`, `rollOver`, `rollOut`, `dragOver`, `dragOut`, and `keyPress`. In the Actions toolbox, the MovieClip object has an Events category containing methods that correspond to each button event, such as `onPress`, `onRelease`, `onReleaseOutside`, `onRollOver`, `onRollOut`, `onDragOver`, `onDragOut`, and `onKeyPress`.

A button movie clip has a full movie clip Timeline, not the four-frame Timeline of a button. You can use the frame labels _up, _over, and _down to create the Up, Over, and Down states of a button movie clip. When the user moves the mouse over a button movie clip or clicks it, the gotoAndStop action causes the playhead to go to the appropriate frame label and display the appropriate image on the Stage. If you want the playhead to start playing at the frame label, you can put a play action on the frame.

To designate a movie clip to use as the hit area of a button movie clip, you use the hitArea property of the MovieClip object.

For information about using button events with buttons, see "Using button event handler methods to trigger scripts" on page 261.

To use the on action to create a button movie clip:

1. Select a movie clip on the Stage.
2. Choose Window > Actions to open the Actions panel if it isn't already open.
3. In the Actions toolbox (at the left of the panel), click the Actions category, click the Movie Control category, and then double-click the on action.
4. In expert mode, enter the events you want to include. In normal mode, choose these events above the Script pane.
5. Inside the on action, add actions to run when the selected events occur.

To define a movie clip event handler method to create a button movie clip:

1. On the Stage, select the movie clip that you want to turn into a button movie clip.
2. Enter an instance name in the Property inspector.
3. Select a frame, button, or movie clip to which to attach the action.
4. Choose Window > Actions to open the Actions panel if it isn't already open.
5. In the Actions toolbox, click the Objects category, the Movie category, the MovieClip category, and the Events category, and double-click one of the button event handler methods.
6. Enter values for the following parameters:

- For the *object* parameter, enter the target path for the movie clip whose event you want to handle.
- Pass any parameters needed by the function you will define. (In normal mode, enter these parameters in the Parameters text box.)

7. Add actions inside the function to define the function.

   The following code defines a function for the onPress method of the instance mc that moves the playhead of mc:

```
mc.onPress = function() {
    play();
};
```

To create states for the button movie clip:

1. Select a frame in the Timeline to use as a button state (Up, Over, or Down).
2. Enter a frame label in the Property inspector (_up, _over, or _down).
3. To add additional button states, repeat steps 1–2.

Defining event handler methods in the prototype object

You can create a new ActionScript class for movie clips and define the event handler methods in the prototype object of that new class. Defining the methods in the prototype object makes all the instances of this symbol respond the same way to these events.

You can also add an `onClipEvent` or `on` event handler action to an individual instance to provide unique instructions that run only when that instance's event occurs. The `onClipEvent` and `on` actions don't override the event handler method; both events cause their scripts to run. However, if you define the event handler methods in the prototype object and also define an event handler method for a specific instance, the instance definition overrides the prototype definition.

To define an event handler method in an object's prototype object:

1. Place a movie clip symbol with linkage ID `theID` in library.
2. Select a frame in the Timeline of the object.
3. Choose Window > Actions to open the Actions panel if it isn't already open.
4. If the Actions panel is in normal mode, choose Expert Mode from the View Options pop-up menu above the Script pane.
5. Use the `function` action to define a new class in the Script pane, as in the following:

   ```
   // define a class
   function myClipClass() {}
   ```

   This new class will be assigned to all instances of the movie clip that are added to the movie by the Timeline, or that are added to the movie with the `attachMovie` or `duplicateMovieClip` method. If you want these movie clips to have access to the methods and properties of the built-in MovieClip object, you'll need to make the new class inherit from the MovieClip class.
6. Enter code like the following in the Script pane:

   ```
   // inherit from MovieClip class
   myClipClass.prototype = new MovieClip();
   ```

   Now the class `myClipClass` inherits all the properties and methods of the MovieClip class.
7. Enter code like the following to define the event handler methods for the new class:

   ```
   // define event handler methods for myClipClass class
   myClipClass.prototype.onLoad = function() {trace ("movie clip loaded");}
   myClipClass.prototype.onEnterFrame = function() {trace ("movie clip entered
     frame");}
   ```
8. Choose Window > Library to open the Library panel if it isn't already open.
9. Select the symbols that you want to associate with your new class, and choose Linkage from the pop-up menu in the upper right of the Library panel.
10. In the Linkage Properties dialog box, select Export for ActionScript.
11. Enter an identifier in the Identifier box.

    The identifier must be the same for all symbols that you want to associate with the new class. In the `myClipClass` example, the identifier is `theID`.
12. Enter code like the following in the Script pane:

    ```
    // register class
    Object.registerClass("theID", myClipClass);
    _root.attachMovie("theID","myName",1);
    ```

This registers any symbol whose linkage identifier is `theID` with the class `myClipClass`. All instances of `myClipClass` have event handler methods that behave as you defined them in step 6. They also behave like all instances of the class MovieClip, because you told the new class to inherit from the class MovieClip in step 5.

```
function myClipClass(){}

myClipClass.prototype = new MovieClip();
myClipClass.prototype.onLoad = function(){
  trace("movie clip loaded");
}
myClipClass.prototype.onPress = function(){
  trace("pressed");
}

myClipClass.prototype.onEnterFrame = function(){
  trace("movie clip entered frame");<<Loc: removed // from this line--IMD>>
}

myClipClass.prototype.myfunction = function(){
  trace("myfunction called");
}

Object.registerClass("myclipID",myClipClass);
_root.attachMovie("myclipID","ablue2",3);
```

Manipulating buttons with ActionScript

Each button in a Flash movie is an ActionScript object of the class Button and has its own properties and methods. Buttons have the same properties as movie clips, but several properties (`_currentframe`, `_droptarget`, `_framesloaded`, and `_totalframes`) are not supported and return the value `undefined`. <<Loc: changed beginning of next sentence --IMD>>The Button class has two additional properties: `useHandCursor`, which lets you decide whether the cursor turns into a hand when it passes over a button, and `enabled`, which lets you specify whether the button is active or not.

You can give a button an instance name in the Property inspector and use a target path to manipulate it with ActionScript. To write a target path for a button instance, write the target path to the movie clip in which the button is located, and add a dot (`.`) and the instance name of the button. The following example disables the button instance `myButton` on the Timeline of the movie clip `childClip`, which is on the Timeline of the movie clip `parentClip`:

```
parentClip.childClip.myButton.enabled = false;
```

For a complete list of methods and properties of the Button object, see the online ActionScript Dictionary in the Help menu.

CHAPTER 14
Creating Interaction with ActionScript

In simple animation, Macromedia Flash MX plays the scenes and frames of a movie sequentially. In an interactive movie, your audience uses the keyboard and mouse to jump to different parts of a movie, move objects, enter information in forms, and perform many other interactive operations.

You use ActionScript to create scripts that tell Flash what action to perform when an event occurs. Some events that can trigger a script occur when the playhead reaches a frame, when a movie clip loads or unloads, or when the user clicks a button or presses keys on the keyboard.

Scripts can consist of a single action, such as instructing a movie to stop playing, or a series of actions, such as first evaluating a condition and then performing an action. Many actions are simple and let you create basic controls for a movie. Other actions require some familiarity with programming languages and are intended for advanced development.

Controlling movie playback

The following are basic actions that let you control the playhead in the Timeline and load a new Web page into a browser window:

- The `goto` action jumps to a frame or scene.
- The `play` and `stop` actions play and stop movies.
- The `getURL` action jumps to a different URL.

This section explains the simplest way to use the Actions panel in normal mode to assign these actions to frames, buttons, and movie clips in your document.

Jumping to a frame or scene

To jump to a specific frame or scene in the movie, you use the `goto` action. When the movie jumps to a frame, you can choose parameters that either play the movie from the new frame (the default) or stop at the frame. In expert mode, the `goto` action is listed as two actions in the Actions toolbox: `gotoAndPlay` and `gotoAndStop`. The movie can also jump to a scene and play a specified frame or the first frame of the next or previous scene.

To jump to a frame or scene:

1. Select a frame, button instance, or movie clip instance to which you will assign the action.
2. Choose Window > Actions to display the Actions panel if it is not already visible. If the Actions panel is not in normal mode, choose Normal Mode from the View Options pop-up menu.

3 In the Actions toolbox, click the Actions category and then click the Movie Control category, and double-click the `goto` action.

Flash inserts the `gotoAndPlay` action in the Script pane.

4 To keep playing the movie after the jump, leave the Go To and Play option (the default) selected in the parameters pane. To stop the movie after the jump, select the Go To and Stop option.

5 In the Scene pop-up menu in the parameters pane, specify the destination scene.

If you select Next or Previous, the playhead jumps to the first frame in the next or previous scene. If you select the current scene or a scene that you have named, you must provide a frame for the playhead to jump to.

6 In the Type pop-up menu in the parameters pane, choose a destination frame:

- Next Frame or Previous Frame sets the destination frame to the next or previous frame.
- Frame Number, Frame Label, or Expression allows you to specify a frame. An expression is any part of a statement that produces a value, such as `1 + 1`.

7 If you chose Frame Number, Frame Label, or Expression in step 6, in the Frame parameter box enter the frame number, the frame label, or an expression that evaluates to a frame number or label.

The following action jumps the playhead to frame 50, where play continues:

```
gotoAndPlay(50);
```

The following action jumps the playhead to a frame that is five frames ahead of the frame that contains the action:

```
gotoAndStop(_currentframe + 5);
```

For more information on writing expressions, see "Using operators to manipulate values in expressions" on page 223.

For more information about using the Actions panel in normal and expert modes, see Chapter 14, "Creating Interaction with ActionScript," on page 267.

Playing and stopping movies

Unless instructed otherwise, once a movie starts, it plays through every frame in the Timeline. You can stop or start a movie by using the `play` and `stop` actions. For example, you can use the `stop` action to stop a movie at the end of a scene before proceeding to the next scene. Once stopped, a movie must be explicitly started again, by means of the `play` action.

You can use the `play` and `stop` actions to control the main Timeline or the Timeline of any movie clip or loaded movie. The movie clip you want to control must have an instance name and must be present in the Timeline. (For more information, see "Working with Movie Clips and Buttons" under Help > Using Flash.)

To stop a movie:

1. Select a frame, button instance, or movie clip instance to which you will assign the action.
2. Choose Window > Actions to display the Actions panel if it's not already visible. If the Actions panel is not in normal mode, choose Normal Mode from the View Options pop-up menu.
3. In the Actions toolbox, click the Actions category, then click the Movie Control category, and select the `stop` action.

   If the action is attached to a frame, the following code appears in the Script pane:

   ```
   stop();
   ```

   If the action is attached to a button, the action is automatically enclosed in an `on (mouse event)` handler, as shown here:

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

   If the action is attached to a movie clip, the action is automatically enclosed in an `onClipEvent` handler, as shown here:

   ```
   onClipEvent (load) {
     stop();
   }
   ```

Note: Empty parentheses after an action indicate that it has no parameters.

To play a movie:

1. Select the frame, button, or movie clip to which you will assign the action.
2. Choose Window > Actions to display the Actions panel if it's not already visible. If the Actions panel is not in normal mode, choose Normal Mode from the View Options pop-up menu.
3. In the Actions toolbox, click the Actions category, select the Movie Control category, and double-click the `play` action.

   If the action is attached to a frame, the following code appears in the Script pane:

   ```
   play();
   ```

   If the action is attached to a button, the action is automatically enclosed in an `on (mouse event)` handler, as shown here:

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

   If the action is attached to a movie clip, the action is automatically enclosed in an `onClipEvent` handler, as shown here:

   ```
   onClipEvent (load) {
      play();
   }
   ```

Jumping to a different URL

To open a Web page in a browser window, or to pass data to another application at a defined URL, you can use the `getURL` action. For example, you can have a button that links to a new Web site, or you can send data to a CGI script for processing in the same way as you would an HTML form.

In the following procedure, the requested file must be at the specified location and absolute URLs must have a network connection (for example, http://www.myserver.com/).

For information on passing variables, see "Connecting with External Sources" under Help > Using Flash.

To jump to a URL:

1. Select the frame, button instance, or movie clip instance to which you will assign the action.
2. Choose Window > Actions to display the Actions panel if it's not already visible. If the Actions panel is not in normal mode, choose Normal Mode from the View Options pop-up menu.
3. In the Actions toolbox, click the Actions category, then click the Browser/Network category, and double-click the `getURL` action.
4. In the parameters pane, enter the URL from which to get the document or to which you are sending data, following these guidelines:

- Use either a relative path, such as mypage.html, or an absolute path, such as http://www.mydomain.com/mypage.html.

  A relative path lets you describe one file's location in relation to another; it tells Flash to move up and down the hierarchy of nested files and folders, starting from the file where you issued the `getURL` instruction. An absolute path is the complete address that specifies the name of the server on which the file resides, the path (the nested hierarchy of directories, volumes, folders, and so on), and the name of the file itself. For more information about writing paths, see "About absolute and relative target paths" under Help > Using Flash.

- To get a URL based on the value of an expression, select Expression and enter an expression that evaluates to the URL's location.

  For example, the following statement indicates that the URL is the value of the variable `dynamicURL`:

  ```
  getURL(dynamicURL);
  ```

  For information on writing expressions, see Chapter 12, "Understanding the ActionScript Language," on page 203.

5. For Window, specify the window or HTML frame into which the document will be loaded, as follows.

- Choose from the following reserved target names:

  `_self` specifies the current frame in the current window.

  `_blank` specifies a new window.

  `_parent` specifies the parent of the current frame.

  `_top` specifies the top-level frame in the current window.

- Enter the name of a specific window or frame as it is named in the HTML file.
- Select Expression and enter the expression that evaluates to the window's location.

6 For Variable, choose a method for sending variables for the loaded movie to the location listed in the URL text box:

- Choose Send Using Get to append a small number of variables to the end of the URL. For example, use this option to send the values of the variables in a Flash movie to a server-side script.
- Choose Send Using Post to send variables separate from the URL, as longer strings in a separate header; this allows you to send more variables and lets you post information collected from a form to a CGI script on the server.
- Choose Don't Send to prevent variables from being passed.

  Your code would look similar to the following line:

```
getUrl ("page2.html", "blank");
```

  The `getURL` action loads the HTML file page2.html into a new browser window.

For more information on the `getURL` action, see its entry in the online ActionScript Dictionary in the Help menu.

Creating complex interactivity

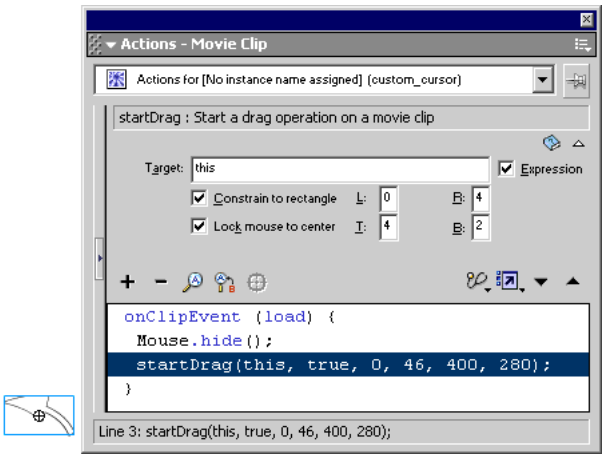
To create complex interactivity, you need to understand the following techniques:

- Creating a custom cursor
- Getting the mouse position
- Capturing keypresses
- Setting color values
- Creating sound controls
- Detecting collisions

Creating a custom cursor

A standard cursor is the operating system's onscreen representation of the user's mouse pointer. By replacing the standard cursor with one you design in Flash, you can integrate the user's mouse movement within the movie environment more closely. The sample in this section uses a custom cursor that looks like a large arrow. The power of this feature, however, lies in your ability to make the custom cursor look like anything—for example, a football to be carried to the goal line or a swatch of fabric pulled over a couch to change its color.

To create a custom cursor, you design the cursor movie clip on the stage. Then in ActionScript you hide the standard cursor and track the movement of the custom cursor. To hide the standard cursor, you use the `hide` method of the built-in Mouse object. To use a movie clip as the custom cursor, you use the `startDrag` action.



Actions attached to a movie clip to create a custom cursor (see customCursor.fla)

To create a custom cursor:

1. Create a movie clip to use as a custom cursor.
2. Select the movie clip instance on the Stage.
3. Choose Window > Actions to open the Actions panel if it is not already visible.
4. To hide the standard cursor, in the Actions toolbox, click the Objects category, click the Movie category, click Mouse, click Methods, and double-click `hide`.

   The code should look like this:

   ```
   onClipEvent(load){
     Mouse.hide();
   }
   ```

5. To apply the new cursor, in the Actions toolbox, click the Actions category, then click Movie Clip Control and double-click `startDrag`.

6 To limit the mouse movement, select the Expression box and type `this` for the target. Then select Lock Mouse to Center and Constrain to Rectangle, and enter values. For example, you might enter the following:

L: 0

T: 46

R: 400

B: 280

Your code should look like this:

```
onClipEvent (load) {
  Mouse.hide();
  startDrag(this, true, 0, 46, 400, 280);
}
```

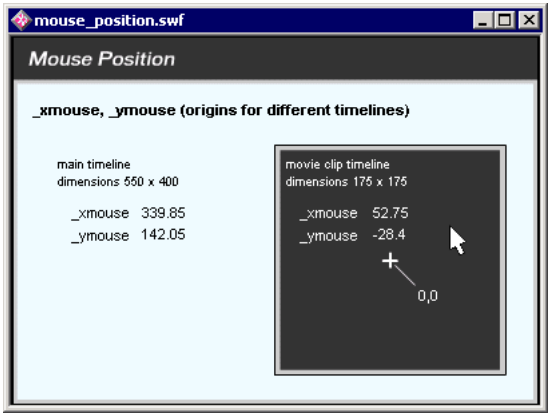
7 Choose Control > Test Movie to test your custom cursor.

Buttons still function when you use a custom cursor. It's a good idea to put the custom cursor on the top layer of the Timeline so that it moves in front of buttons and other objects as you move the mouse in the movie.

For more information about the methods of the Mouse object, see the online ActionScript Dictionary in the Help menu.

Getting the mouse position

Tracking the mouse position gives you information about user movement in your movie. This information allows you to tie user behavior to movie events. You can use the `_xmouse` and `_ymouse` properties to find the location of the mouse pointer (cursor) in a movie. Each Timeline has an `_xmouse` and `_ymouse` property that returns the location of the mouse within its coordinate system. The position is always relative to the registration point. For the main Timeline (`_level0`), the registration point is the upper left corner.



The `_xmouse` and `_ymouse` properties within the main Timeline and a movie clip Timeline (mouse_position.fla)

The following procedures show two ways to get the mouse position.

To get the current mouse position within the main Timeline:

1. Create two dynamic text boxes and name them `x_pos` and `y_pos`.
2. Choose Window > Actions to open the Actions panel if it is not already visible.
3. To return the mouse position within the main Timeline, add the following code to any frame in the `_level0` movie:

```
x_pos = _root._xmouse;
y_pos = _root._ymouse;
```

The variables `x_pos` and `y_pos` are used as containers to hold the values of the mouse positions. You could use these variables in any script in your document. In the following code, the values of `x_pos` and `y_pos` update every time the user moves the mouse.

```
onClipEvent(mouseMove){
  x_pos = _root._xmouse;
  y_pos = _root._ymouse;
}
```

To get the current mouse position within a movie clip:

1. Create a movie clip.
2. Select the movie clip instance on the Stage. Using the Property inspector, name it `myMovieClip`.
3. Choose Window > Actions to open the Actions panel if it is not already visible.
4. Use the movie clip's instance name to return the mouse position within the main Timeline.

   For example, the following statement could be placed on any Timeline in the `_level0` movie to return the `_ymouse` position in the `myMovieClip` instance:

   ```
   x_pos = _root.myMovieClip._xmouse
   y_pos = _root.myMovieClip._ymouse
   ```

   The code returns the `_xpos` and `_ypos` of the mouse relative to the registration point.
5. Choose Control > Test Movie to test the movie.

You can also determine the mouse position within a movie clip by using the `_xmouse` and `_ymouse` properties in a clip event, as in the following code:

```
onClipEvent(enterFrame){
  xmousePosition = _xmouse;
  ymousePosition = _ymouse;
}
```

For more information about the `_xmouse` and `_ymouse` properties, see the online ActionScript Dictionary in the Help menu.

Capturing keypresses

You can use the methods of the built-in Key object to detect the last key the user pressed. The Key object does not require a constructor function; to use its methods, you simply call the object itself, as in the following example:

```
Key.getCode();
```

You can obtain either virtual key codes or ASCII values of keypresses:

- To obtain the virtual key code of the last key pressed, use the `getCode` method.
- To obtain the ASCII value of the last key pressed, use the `getAscii` method.

A virtual key code is assigned to every physical key on a keyboard. For example, the Left Arrow key has the virtual key code 37. By using a virtual key code, you ensure that your movie's controls are the same on every keyboard, regardless of language or platform.

ASCII (American Standard Code for Information Interchange) values are assigned to the first 127 characters in every character set. ASCII values provide information about a character on the screen. For example, the letter "A" and the letter "a" have different ASCII values.

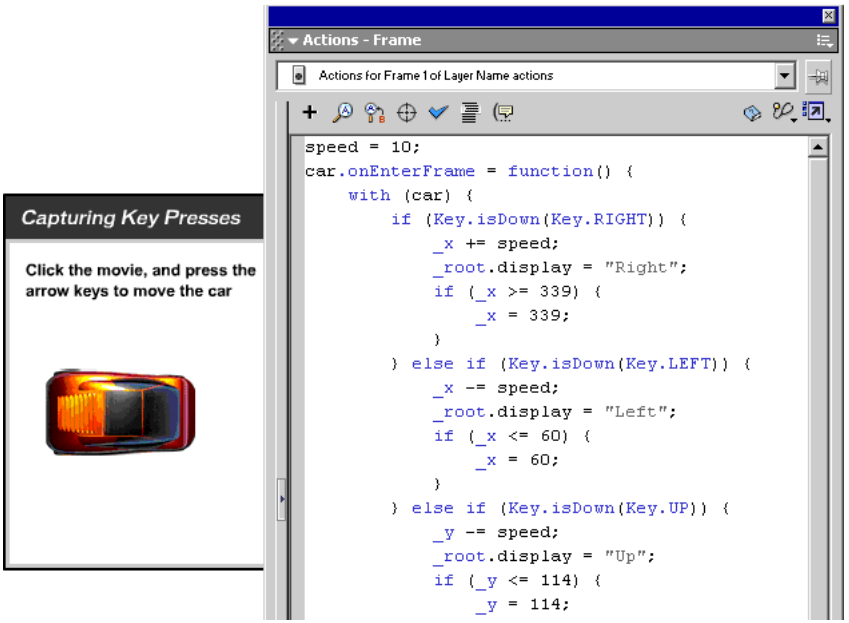
To decide which keys to use and determine their virtual key codes, use one of these approaches:

- See the list of key codes in "Keyboard Keys and Key Code Values" under Help > Using Flash.
- Use a Key object constant. (In the Actions toolbox, click the Objects category, click Movie, click Key, and click Constants.)
- Assign the following clip action, then choose Control > Test Movie and press the desired key:

```
onClipEvent(keyDown) {
  trace(Key.getCode());
}
```

The key code of the desired key displays in the Output window.

A common place to use Key object methods is within an event handler. In the following example, the user moves the car using the arrow keys. The `Key.isDown` method indicates whether the key being pressed is the right, left, up, or down arrow. The event handler, `onEnterFrame`, determines the `Key.isDown(keyCode)` value from the `if` statements. Depending on the value, the handler instructs the Flash Player to update the position of the car and to display the direction.



The input from the keyboard keys moves the car (keyCode.fla).

The following procedure shows how to capture keypresses using a car example.

To create a keyboard-activated movie clip:

1. On the Stage, create a movie clip that will move in response to keyboard arrow activity.

   In this example, the movie clip instance name is `car`.

2. On the Stage, create a dynamic text box that will be updated with the direction of the car. Using the Property inspector, give it a variable name of `display`.

   Note: Don't confuse variable names with instance names.

3. Select frame 1 in the Timeline; then choose Window > Actions to open the Actions panel if it is not already visible.

4. To set how quickly the car moves across the screen with each keypress, in the Actions toolbox, click the Actions category, click Variables, double-click `set variable`, and name the variable `speed`. Then select the Expression option for Value and enter a value of 10.

5 To create the event handler that processes the event and subsequent behavior, in the Actions toolbox, click the Objects category, then click Movie, Movie Clip, and Events, and double-click `onEnterFrame`. Enter `car` as the object name.

6 Click in the Parameters text box to place the insertion point. Then click the Actions category, click the Variables category, and double-click `with`. Enter `car` as the object name.

Your code should look like this:

```
speed = 10;
car.onEnterFrame = function() {
    with (car) {
    }
};
```

7 To add the conditions to the event handler, in the Actions toolbox, click the Actions category, click Conditions/Loops, and double-click `if`.

8 Click in the Condition text box to place the insertion point. Click the Objects category, click Movie, Key, and Methods, and double-click `isDown`. Then click the Objects category, click Movie, Key, and Constants, and double-click `RIGHT` for the key code.

```
speed = 10;
car.onEnterFrame = function() {
    with (car) {
        if (Key.isDown(Key.RIGHT)) {
        }
    }
};
```

Next, the `if` statement needs parameters to carry in case `Key.isDown(Key.RIGHT)` evaluates to `true`. In other words, if the Right Arrow key is down, the car should move to the right—the `_x` property should increase. Also, the word *Right* should be displayed in the movie, so the dynamic text box needs to be updated.

9 To enter the conditional statements, in the Actions Toolbox, click the Operators category; then click Assignment, and drag += onto line 5 in the Script pane (between the `if` statement brackets). Enter the following code in the Expression text box:

```
_x += speed
```

10 To limit the car to the right edge of the movie, add a nested `if` statement. In the Actions toolbox, click the Actions category, then click Conditions/Loops and drag `if` to line 6 in the Script pane. Enter the following code in the Condition text box:

```
_x > 339
```

11 Click the Actions category, click Variables, and double-click `set variable`. Enter `_x = 339` in the Expression text box.

12 To update the dynamic text box, in the Actions toolbox, click the Actions category, click Variables, and drag `set variable` to line 9 in the Script pane. Enter `_root.display` in the Variable text box and `Right` in the Value text box.

Your code should look like this:

```
speed = 10;
car.onEnterFrame = function() {
  with (car) {
    if (Key.isDown(Key.RIGHT)) {
      _x += speed;
      if (_x >= 339) {
        _x = 339;
      }
      _root.display = "Right";
    }
  }
};
```

You can take the time now to test the movie (Control > Test Movie), but the car will only move to the right.

13 To add the left, up, and down movement, in the Actions Toolbox, click the Actions category, click Conditions/Loops, and drag `else if` to line 10 in the Script pane. Then repeat steps 8 through 11, changing the parameter details as in the following code:

```
} else if (Key.isDown(Key.LEFT)) {
  _x -= speed;
  if (_x < 60) {
    _x = 60;
  }
  _root.display = "Left";
} else if (Key.isDown(Key.UP)) {
  _y -= speed;
  if (_y < 114) {
    _y = 114;
  }
  _root.display = "Up";
} else if (Key.isDown(Key.DOWN)) {
  _y += speed;
  if (_y > 244) {
    _y = 244;
  }
  _root.display = "Down";
}
```

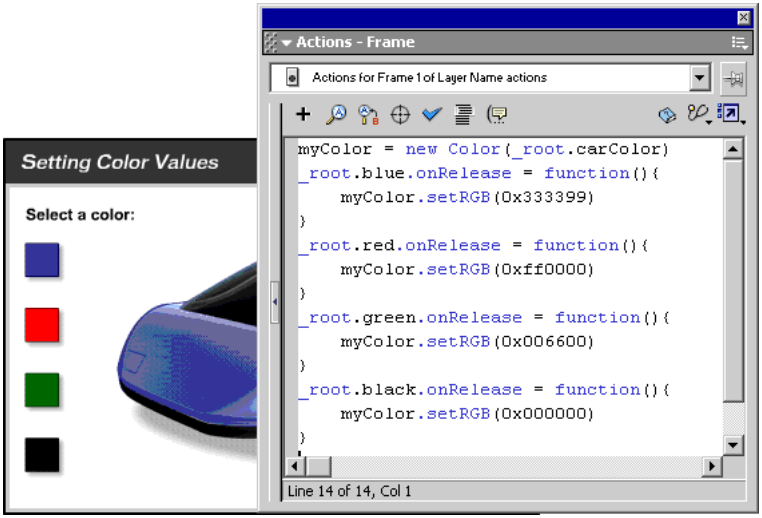
Make sure to place your code on the correct lines (lines 10 through 28). The bracket that closes the outer `if` statement and the bracket that closes the event handler should follow on lines 29 and 30.

14 Choose Control > Test Movie to test the movie.

For more information about the methods of the Key object, see the online ActionScript Dictionary in the Help menu.

Setting color values

You can use the methods of the built-in Color object to adjust the color of a movie clip. The `setRGB` method assigns hexadecimal RGB (red, green, blue) values to the object. The following example uses `setRGB` to change an object's color based on user input.



The button action creates a Color object and changes the color of the car based on user input (see setRGB.fla)

To set the color value of a movie clip:

1. Select a movie clip on the Stage.
2. In the Property inspector, enter `carColor` as the instance name.
3. Create a button named `color chip`, place four instances of the button on the Stage, and name them `red`, `green`, `blue`, and `black`.
4. Select frame 1 in the main Timeline and choose Window > Actions.
5. To create a new Color object, in the Actions toolbox, click the Objects category, then click Movie and Color, double-click `new Color`, and choose `_root.carColor` for the target. Enter `myColor =` in the Expression text box.

   Your code should look like this:

   ```
   myColor = new Color(_root.carColor);
   ```

6. To associate an event with an object, in the Actions toolbox, click the Objects category, then click Movie, Movie Clip, and Events, and double-click `onRelease`. Enter the button instance name—either `_root.red`, `_root.green`, `_root.blue`, or `_root.black`—in the Object text box.

7 In the Actions toolbox, click the Objects category; then click Movie, Color, Methods, and double-click `setRGB`. Enter the Color object name `myColor` in the Object text box. Enter the hexadecimal representation for the color in the Parameter text box:

Color	Hexadecimal value
Red	`0xff0000`
Green	`0x00ff00`
Blue	`0x0000ff`
Black	`0x000000`

8 Repeat steps 6 and 7 for all four colors, so that your code looks like this:

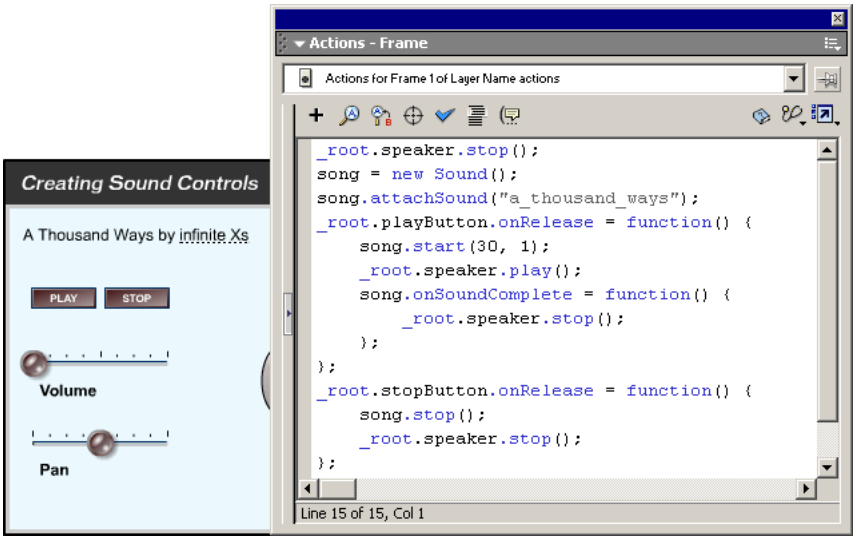
```
myColor = new Color(_root.carColor)
_root.blue.onRelease = function(){
  myColor.setRGB(0x0000ff)
}
_root.red.onRelease = function(){
  myColor.setRGB(0xff0000)<<Loc: changed hexadecimal value here--IMD>>
}
_root.green.onRelease = function(){
  myColor.setRGB(0x00ff00)<<Loc: changed hexadecimal value here--IMD>>
}
_root.black.onRelease = function(){
  myColor.setRGB(0x000000)
}
```

9 Choose Control > Test Movie to change the color of the movie clip.

For more information about the methods of the Color object, see the online ActionScript Dictionary in the Help menu.

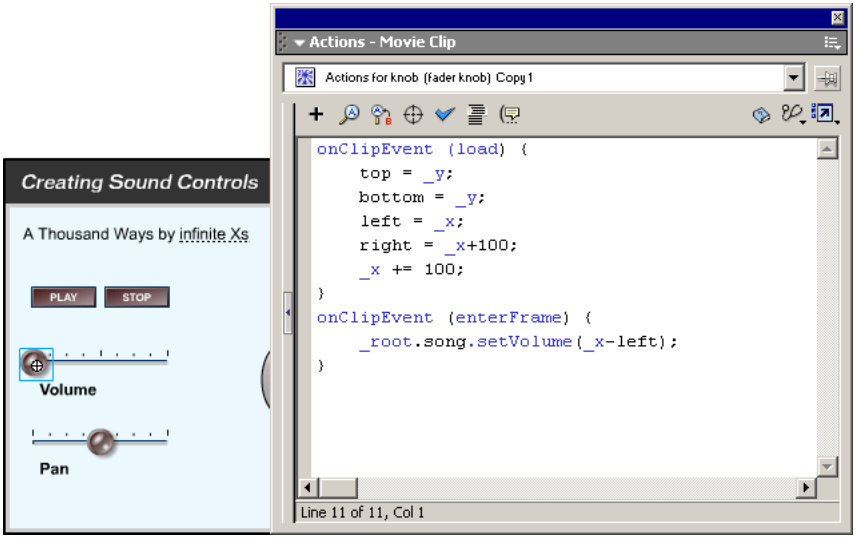
Creating sound controls

You use the built-in Sound object to control sounds in a movie. To use the methods of the Sound object, you must first create a new Sound object. Then you can use the `attachSound` method to insert a sound from the library into a movie while the movie is running.



When the user releases the Play button, a song plays through the speaker (see sound.fla).

The Sound object's `setVolume` method controls the volume, and the `setPan` method adjusts the left and right balance of a sound.



When the user drags the volume slider, the `setVolume` method is called (see sound.fla).

The following procedures show how to create sound controls like the ones shown above.

To attach a sound to a Timeline:

1. Choose File > Import to import a sound.
2. Select the sound in the library, right-click, and choose Options > Linkage.
3. Select Export for ActionScript and Export in first frame; then give it the identifier `a_thousand_ways`.
4. Add a button to the Stage and name it `playButton`.
5. Add a button to the Stage and name it `stopButton`.
6. Add a movie clip to the Stage and name it `speaker`.
7. Select frame 1 in the main Timeline and choose Window > Actions.
8. To pause the movie until the user selects Play, in the Actions toolbox, click the Objects category, click Movie, Sound, and Methods, and double-click `stop`. Enter `_root.speaker` in the Object text box.
9. To create a new Sound object, in the Actions toolbox, click the Objects category, click Movie, click Sound, and double-click `new Sound`. Enter `song =` in the Expression text box.
10. In the Actions toolbox, click the Objects category, click Movie, Sound, and Methods, and double-click `attachSound`. Enter `song` in the Object text box and `"a_thousand_ways"` (including the quotation marks) in the Parameters text box.

11 To start the song, in the Actions toolbox, click the Objects category, then click Movie, Sound, and Methods, and double-click `start`.

12 To activate the speaker, in the Actions toolbox, click the Objects category, then click Movie, Movie Clip, and Methods, and double-click `play`. Enter `_root.speaker` in the Object text box.

Your code should look like this:

```
_root.speaker.stop();
song = new Sound();
song.attachSound("a_thousand_ways");
_root.playButton.onRelease = function() {
    song.start();
    _root.speaker.play();
};
```

13 To stop the speaker when the song ends, click the Objects category, then click Movie, Sound, and Events, and double-click `onSoundComplete`. Enter `song` in the Object text box. Enter `onSoundComplete` in the Method text box.

14 In the Actions toolbox, click the Objects category, click Movie, Sound, and Methods, and double-click `stop`. Enter `_root.speaker` in the Object text box.

Your code should look like this:

```
_root.speaker.stop();
song = new Sound();
song.attachSound("a_thousand_ways");
_root.playButton.onRelease = function() {
    song.start();
    _root.speaker.play();
    song.onSoundComplete = function() {
        _root.speaker.stop();
    };
};
```

15 Choose Control > Test Movie to hear the sound.

To create a sliding volume control:

1 Drag a button to the Stage.

2 Select the button and choose Insert > Convert to Symbol. Be careful to choose the movie clip behavior.

This creates a movie clip with the button on its first frame.

3 Select the movie clip and choose Edit > Edit Symbol.

4 Select the button and choose Window > Actions.

5 Enter the following actions:

```
on (press) {
    startDrag("", false, left, top, right, bottom);
}
on (release) {
    stopDrag();
}
```

The `startDrag` parameters `left`, `top`, `right`, and `bottom` are variables set in a clip action.

6 Choose Edit > Edit Document to return to the main Timeline.

7 Select the movie clip on the Stage.

8 Enter the following actions:

```
onClipEvent (load) {
    top = _y;
    bottom = _y;
    left = _x;
    right = _x+100;
    _x += 100;
}
onClipEvent (enterFrame) {
    _root.song.setVolume(_x-left);
}
```

9 Choose Control > Test Movie to use the volume slider.

To create a sliding balance control:

1 Drag a button to the Stage.

2 Select the button and choose Insert > Convert to Symbol. Choose the movie clip property.

3 Select the movie clip and choose Edit > Edit Symbol.

4 Select the button and choose Window > Actions.

5 Enter the following actions:

```
on (press) {
  startDrag ("", false, left, top, right, bottom);
  dragging = true;
}
on (release, releaseOutside) {
  stopDrag ();
  dragging = false;
}
```

The `startDrag` parameters `left`, `top`, `right`, and `bottom` are variables set in a clip action.

6 Choose Edit > Edit Document to return to the main Timeline.

7 Select the movie clip on the Stage.

8 Enter the following actions:

```
onClipEvent(load){
  top=_y;
  bottom=_y;
  left=_x-50;
  right=_x+50;
  center=_x;
}

onClipEvent(enterFrame){
  if (dragging==true){
    _root.s.setPan((_x-center)*2);
  }
}
```

9 Choose Control > Test Movie to use the balance slider.

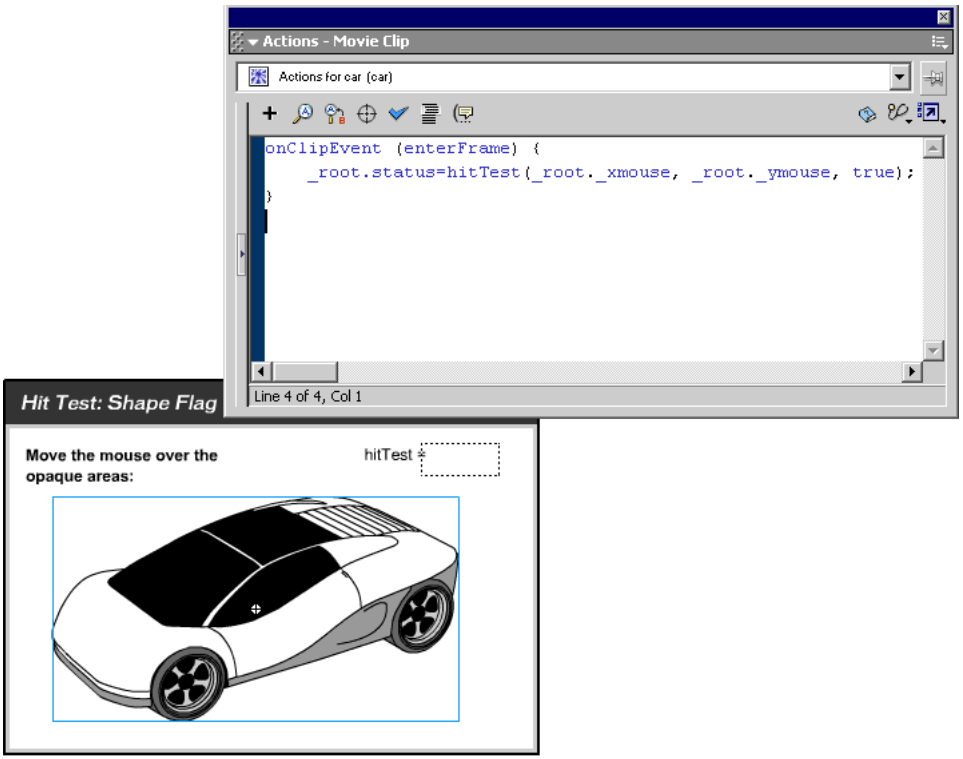
For more information about the methods of the Sound object, see the online ActionScript Dictionary in the Help menu.

Detecting collisions

The `hitTest` method of the MovieClip object detects collisions in a movie. It checks to see if an object has collided with a movie clip and returns a Boolean value (`true` or `false`).

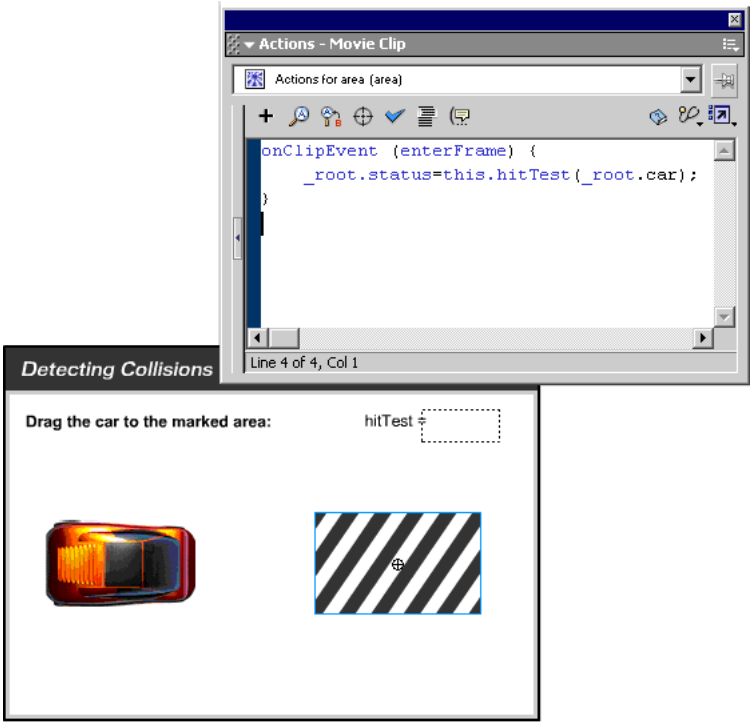
There are two cases in which you would want to know whether a collision has occurred: to test if the user has arrived at a certain static area on the Stage, and to determine when one movie clip has reached another. With the `hitTest` method, you can determine these results.

You can use the parameters of the `hitTest` method to specify the *x* and *y* coordinates of a hit area on the Stage, or use the target path of another movie clip as a hit area. When specifying *x* and *y*, `hitTest` returns `true` if the point identified by (*x*, *y*) is a nontransparent point. When a target is passed to `hitTest`, the bounding boxes of the two movie clips are compared. If they overlap, `hitTest` returns `true`. If the two boxes do not intersect, `hitTest` returns `false`.



"True" appears in the text field whenever the mouse pointer is over the car body (see shape_flag.fla).

You can also use the `hitTest` method to test a collision between two movie clips.



"True" appears in the text field whenever one movie clip touches the other (see hit_test.fla).

The following procedures show how to detect collision using the car example.

To perform collision detection between a movie clip and a point on the Stage (see shape_flag.fla):

1. Select a movie clip on the Stage.
2. Create a dynamic text box on the Stage and enter `status` as the instance name in the Property inspector.
3. Choose Window > Actions to open the Actions panel if it is not already visible.
4. In the Actions toolbox, click the Actions category, click Variables, double-click `set variable`, and name the variable `_root.status`. Then select the Expression check box for Value and enter the following code in the Value text box:

   ```
   hitTest(_root._xmouse, _root._ymouse, true)
   ```

   Flash automatically adds the `onClipEvent` handler.
5. Highlight the `onClipEvent` action and select `enterFrame` as the event.
6. Choose Control > Test Movie and move the mouse over the movie clip to test the collision.

   The value `true` is displayed whenever the mouse is over a nontransparent pixel.

To perform collision detection on two movie clips (see hit_test.fla):

1. Drag two movie clips to the Stage and give them the instance names `car` and `area`.
2. Create a dynamic text box on the Stage and enter `status` as the instance name in the Property inspector.
3. Select `area` and choose Window > Actions if the Actions panel is not already visible.
4. To apply the `hitTest` test, in the Actions toolbox, click the Actions category, click Miscellaneous Actions, and double-click `evaluate`. Enter the following code in the Expression text box:

   ```
   _root.status=this.hitTest(_root.car);
   ```

   Flash automatically adds the `onClipEvent` handler.
5. Highlight the `onClipEvent` action and select `enterFrame` as the event.
6. Select `car` from the jump menu at the top of the Actions panel.
7. To apply movement to the car, in the Actions toolbox, click the Actions category, click Movie Clip Control, and double-click `startDrag`.
8. To limit the car's movement, select the Lock Mouse to Center and Constrain to Rectangle options, and enter 4 for Left, 70 for Top, 396 for Right, and 273 for Bottom.

   Flash automatically adds the `onClipEvent` handler.
9. Highlight the `onClipEvent` action in the Script pane and choose the `Mouse down` event.

   Your code should look like this:

   ```
   onClipEvent (mouseDown) {
     startDrag("", true, 4, 70, 396, 273);
   }
   ```
10. To stop the car, in the Actions toolbox, click the Actions category, click Movie Clip Control, and double-click `stopDrag`.

    Flash automatically adds the `onClipEvent` handler.
11. Highlight the `onClipEvent` action in the Script pane and choose the `Mouse up` event.

    Your code should look like the following code:

    ```
    onClipEvent (mouseDown) {
      startDrag("", true, 4, 70, 396, 273);
    }
    onClipEvent (mouseUp) {
        stopDrag();
    }
    ```
12. Choose Control > Test Movie and drag the movie clip to test the collision detection.

    Whenever the bounding box of the car intersects the bounding box of the area, the status is `true`.

For more information about the `hitTest` method, see the online ActionScript Dictionary in the Help menu.

CHAPTER 15
Using Components

Components are complex movie clips that have defined parameters, which are set during document authoring, and a unique set of ActionScript methods that allow you to set parameters and additional options at runtime. Components replace and extend Smart Clips introduced in earlier versions of Macromedia Flash.

Macromedia Flash MX includes seven Flash UI components: CheckBox, ComboBox, ListBox, PushButton, RadioButton, ScrollBar and ScrollPane. You can use these components separately to add simple user interaction to Flash movies, or together to create a complete user interface for Web forms or applications.

You can customize the appearance of components in several ways:

- Edit the color and text formatting properties defined for all Flash UI components in the global style format
- Edit the component skins
- Create new custom style formats using the ActionScript FStyleFormat object
- Replace the skin elements that make up the component's skin with new skin elements that you create

You should complete the Introduction to Components Tutorial (Help > Tutorials > Introduction to Components) before using components.

You can also create custom components using the ActionScript language. For more information, see the online Flash Support Center at http://www.macromedia.com/flash/support.

Working with components in Flash MX

You use the Components panel to view components and add them to a document during authoring. You can view properties for components that have been added to a document using the Property inspector or the Component Parameters panel.

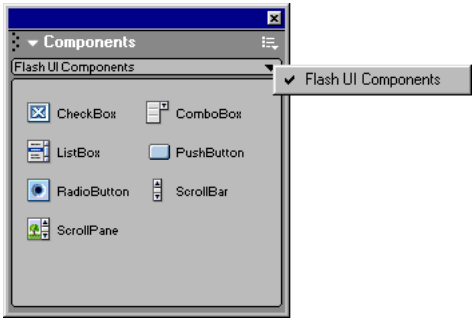
When you add a component to a document, several items for the component are added to the library, including the component movie clip, skins (graphic elements) that control the component's appearance, and core assets for developers who want to modify the component.

The Components panel

All components are stored in the Components panel. When you install the Flash program and launch it for the first time, the Flash UI components are the only components displayed in the Components panel. You can display movie clips with defined parameters or custom components that you import or create in the Components panel by placing the FLA file containing the component movie clips in the Flash MX/First Run/Components folder.

To display the Components panel:

Choose Window > Components.



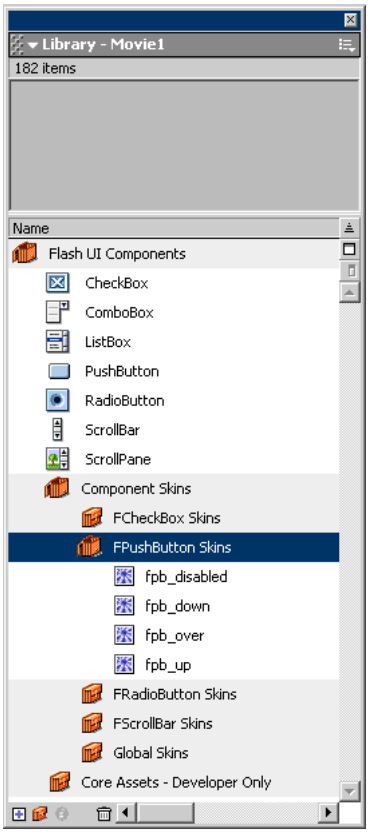
The Components panel displaying the Flash UI components

Components in the Library panel

When you add one or more components to a Flash document, a Flash UI Components folder is added to the Library panel, with the following items inside:

- The component movie clip, represented by an icon for the component type
- A Component Skins folder, with a Global Skins folder containing graphic elements that apply to all components, and a skins folder for the individual component type

- A Core Assets folder with assets for advanced developers, including a Data Provider API and the class hierarchy used by components.



The Flash UI Components folder in the library, with component movie clips, the Component Skins folder, and the Core Assets folder

The library folder structure of custom components you create or import depends on how the components were developed. You can add more instances of a component by dragging the component icon from the library to the Stage.

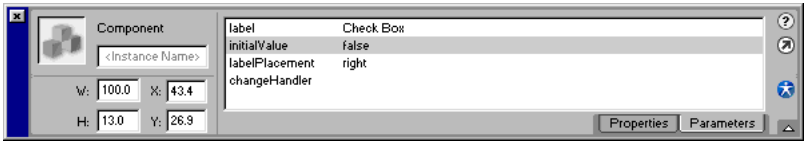
Skins folders contain the graphic symbols, called skins, used to display a component type in a Flash document. All components use the skins in the Global Skins folder. In addition, components have skins folders specific to the component type. Components that use scroll bars share the skins in the FScrollBar Skins folder, and the ListBox component uses the skins in the ComboBox Skins folder. You can edit the skins in the folders to change the appearance of components. However, you cannot edit the components themselves by double-clicking the instance.

Components in the Property inspector and Component Parameters panel

After you add an instance of a component to a Flash document, you use the Property inspector to set and view information for the instance. You create an instance of a component by dragging it from the Components panel onto the Stage, then you name the instance in the Property inspector and specify the parameters for the instance using the fields on the Parameters tab. You can also set parameters for a component instance using the Component Parameters panel.

To view information for a component instance in the Property inspector:

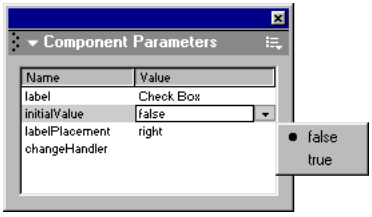
1. Choose Window > Properties.
2. Select an instance of a component on the Stage.
3. To view parameters, click the Parameters tab.



The component view in the Property inspector

To view parameters for a component instance in the Component Parameters panel:

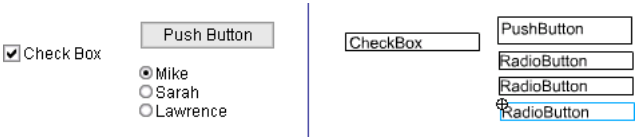
1. Choose Window > Component Parameters.
2. Select an instance of a component on the Stage.



The Component Parameters panel

Working with components in Live Preview

The Live Preview feature lets you view components on the Stage as they will appear in the published movie. Live Preview, enabled by default, lets you see the approximate size and appearance the component will have in the published movie. Live Preview does not reflect changes you make to component property settings or to component skins. Components in Live Preview are not functional. To test component functionality, you can use the Control > Test Movie command.



Push Button component with Live Preview enabled, and with Live Preview disabled

To turn Live Preview on or off:

Choose Control > Enable Live Preview. A check mark next to the option indicates that it is enabled.

Adding components to Flash documents

You can add components to Flash documents using the Components panel, or using the `MovieClip.attachMovie` ActionScript method.

- Beginning Flash users can use the Components panel to add components to Flash documents, specify basic parameters using the Property inspector or the Component Parameters panel, and then use the Actions panel to write ActionScript to control the component.
- Intermediate Flash users can use the Components panel to add components to Flash documents and then use the Property inspector, ActionScript methods, or a combination of the two to specify the parameters.
- Advanced Flash programmers can use a combination of the Components panel and ActionScript to add components and specify properties, or choose to completely implement component instances at runtime using ActionScript.

If you edit the skins of a component and then add another version of the component, or a component that shares the same skins, you can choose to use the edited skins or replace the edited skins with a new set of default skins. If you replace the edited skins, all components using those skins are updated with default versions of the skins. For more information on how to edit skins, see “Customizing component skins” on page 309.

The following section outlines the general steps for adding Flash UI components to your Flash document. These instructions may also apply to custom components that you create or acquire, depending on how the component is authored. For detailed instructions on adding components in the authoring environment, refer to the individual component sections in this chapter. For additional information on using the component ActionScript methods, refer to the component entries in the online ActionScript Dictionary in the Help menu.

Adding a component using the Components panel

When you add a component to a document using the Components panel, several items for the component are added to the Library panel, inside a Flash Components UI folder. See "Components in the Library panel" on page 290.

After you add a component to a document using the Components panel, you can add additional instances of the component to the document by dragging the component from the Library panel to the Stage. You can set properties for additional instances in the Parameters tab of the Property inspector or in the Component Parameters panel, as described in the individual components sections in this chapter.

To add a component to a Flash document using the Components panel:

1. Choose Window > Components.
2. Do one of the following:
   - Drag a component from the Components panel to the Stage.
   - Double-click a component in the Components panel.
3. If you have edited skins for another instance of the same component, or for a component that shares skins with the component you are adding, do one of the following:
   - Choose Don't Replace Existing Items to preserve the edited skins and apply the edited skins to the new component.
   - Choose Replace Existing Items (not undoable) to replace all the skins with default skins. The new component and all previous versions of the component, or of components that share its skins, will use the default skins.
4. Select the component on the Stage.
5. Choose Window > Properties.
6. In the Property inspector, enter an instance name for the component instance.
7. Click the Parameters tab and specify parameters for the instance, as described in the individual component section in this chapter.
8. Change the size and scale of the component as desired. For more information on sizing specific component types, see the individual component sections.
9. Change the color and text formatting of a component as desired, by doing one or more of the following:
   - Set or change a specific property value for a component instance using the `setStyleProperty` method available to all components. For detailed information on the `setStyleProperty` method, see its entry in the online ActionScript Dictionary in the Help menu.
   - Edit multiple properties in the global style format assigned to all Flash UI components.
   - If desired, create a custom style format for specific component instances. For more information, see "Customizing component colors and text" on page 305.
10. Customize the appearance of the component if desired, by doing one of the following:
    - Edit the skins in the Component Skins folder for a component type.
    - Register new skin elements that you create to the component skins. For more information, see "Customizing component skins" on page 309.

To add a component to your Flash document using ActionScript:

Note: The instructions in this section assume an intermediate or advanced knowledge of ActionScript and the Actions panel.

1. Select the frame in the Timeline where you want to place the component.
2. Open the Actions panel in expert mode.
3. Write a function to create the component instance, using the following code as a template:

```
_root.attachMovie("FCheckBoxSymbol", "checkBox1", Z);
_root.checkBox1.setValue(false);
_root.checkBox1.setLabel("myCheckbox");
```

4. Use the ActionScript methods of the component to specify additional options or override parameters set during authoring.

   For more information on writing ActionScript for components, see "Writing change handler functions for components" on page 304 and "Creating forms using components" on page 312. For detailed information on the ActionScript methods available to each component type, see entries for each component in the online ActionScript Dictionary in the Help menu. Entries in the dictionary are referenced using the format FCheckBox (component), FRadioButton (component), and so on.

Deleting components from Flash documents

To delete a component's instances from a Flash document, you delete the component from the Library by deleting the component type icon and the associated Component Skins folder.

If the component you want to delete shares a skins folder with other components (for example, the ScrollBar, ScrollPane, ListBox, and ComboBox components, which share the skins in the FScrollBar skins folder), you should not delete the Component Skins folder, unless you want to replace the skins elements with a new set of default skins. Do not delete skins in the Global Skins folder.

To delete a component from a document:

1. In the Library panel, open the Flash Components UI folder.
2. In the Flash Components UI folder, select the component movie clip that you want to delete.
3. Click the Delete button at the bottom of the Library panel, or choose Delete from the Library panel options menu.
4. In the Delete dialog box, click Delete to confirm the deletion.
5. Open the Component Skins folder inside the Flash Components UI folder.
6. Select the skins folder for the component that you want to delete. For information on which skins are used by a component, see the section for the individual component type in this chapter.

   Note: If the skins folder is used by any other components in the document, you should not delete the skins folder.

7. Click the Delete button at the bottom of the Library panel, or choose Delete from the Library panel options menu.
8. In the Delete dialog box, click Delete.

About component label size and component width and height

If a component instance is not large enough to display its label, the label text is truncated. If a component instance is larger than the text, the hit area extends beyond the label.

If you use the ActionScript `_width` and `_height` properties to adjust the width and height of a component, the component is resized but the layout of the content remains the same. This may cause the component to be distorted in movie playback. You should use the Free Transform tool or the `setSize` or `setWidth` methods of the individual component objects. For more information about sizing components, see information on the individual component types in this chapter.

The CheckBox component

The CheckBox component lets you add check boxes to Flash movies with a minimum of authoring and scripting.

CheckBox parameters

You can set the following parameters for each check box instance in your Flash document using the Parameters tab on the Property inspector or the Component Parameters panel.

Change Handler is a text string specifying the name of a function to call when the value of the check box changes. The function must be defined in the same Timeline as the check box instance. This parameter is optional and needs to be specified only if you want an action to occur when a user selects or deselects a check box. For more information, see "Writing change handler functions for components" on page 304 and the `FComboBox.setChangeHandler` entry in the online ActionScript Dictionary in the Help menu.

Initial Value specifies whether the check box is initially selected (`true`) or unselected (`false`).

Label is the name that appears next to the check box.

Label Placement specifies whether the label appears to the left or to the right of the check box. By default the label is displayed to the right of the check box.

You can set additional options and functionality for check box instances using the methods of the FCheckBox (component)

For more information, see the FCheckBox (component) entry in the online ActionScript Dictionary in the Help menu.

Sizing CheckBox components

You can set the width, but not the height, of a CheckBox component during authoring using the Free Transform tool, or at runtime using the `FCheckBox.setSize` method. The *hit area* of the check box—the area that will respond to the mouse click—is the combined size of the check box and check box label.

CheckBox skins

The CheckBox component uses the skins in the FCheckBox Skins folder and the FLabel skin in the Global Skins folder in the Component Skins folder in the library. Customizing the CheckBox component skins affects all check box instances in the Flash document. For more information about component skins, see "Customizing component skins" on page 309.

The ComboBox component

The ComboBox component lets you add scrollable single-selection drop-down lists to Flash movies with a minimum of authoring and scripting. The ComboBox component creates both *static* and *editable* combo boxes. A static combo box is a scrollable drop-down list that lets the user select items in the list. An editable combo box is a scrollable drop-down list with an input text field (similar to a search field) in which a user can enter text to scroll to the matching menu item in the scroll list.

The ComboBox component uses a zero-based index, where the item with index 0 is the first item displayed. When adding, removing, or replacing list items using the FComboBox methods, you may need to specify the index of the list item.

For more information, see the following entries in the online ActionScript Dictionary in the Help menu:`FComboBox.addItemAt`, `FComboBox.removeItemAt`, and `FComboBox.replaceItemAt`.

The ComboBox component has the following built-in mouse and keyboard controls:

- Up Arrow key moves the selection up one line in the scroll list.
- Down Arrow key moves the selection down one line in the scroll list.
- PageUp moves the selection up one page. The size of the page is determined by the Row Count parameter.
- PageDown moves the selection down one page.
- Home selects the item at the top of the list.
- End selects the item at the bottom of the list.

For more information, see the FComboBox (component) entry in the online ActionScript Dictionary in the Help menu.

ComboBox parameters

You can set the following parameters for each combo box instance in your Flash document using the Parameters tab on the Property inspector or the Component Parameters panel.

Change Handler is a text string specifying a function to call when a user selects an item or enters text in the input field. The function must be defined in the same Timeline as the combo box instance, and may accept the instance name of the combo box as a parameter. This parameter is optional and needs to be specified only if you want an action to take place when the user selects an item or enters text in the combo box and uses the Enter key. For more information, see "Writing change handler functions for components" on page 304 and the `FComboBox.setChangeHandler` entry in the online ActionSript Dictionary in the Help menu.

Data is an array of text strings specifying the values associated with the items (labels) in the combo box. You enter the text strings for the array using the Values dialog box.

Editable determines whether the combo box is editable or static. Editable combo boxes allow users to enter text in a field to search for the matching item in the scroll list, as well as to scroll through the list and select items. Static combo boxes allow users only to scroll through the list and select items.

Labels is an array of text strings specifying the items displayed in the combo box. You enter the text strings for the array using the Values dialog box.

Row Count is the number of items displayed in the combo box before the scroll bar is displayed. The default value for this parameter is 8.

You can set additional options and functionality for combo box instances using the methods of the FComboBox (component). For more information, see the entry for the FComboBox component in the online ActionScript Dictionary.

Sizing ComboBox components

You can set the width but not the height of ComboBox components during authoring using the Free Transform tool, or at runtime using the `FComboBox.setSize` method.If the text of a list item is longer than the width of the combo box, the text is truncated to fit inside the box. The height of a ComboBox component is determined by the size of the font displaying the list items and the Row Count parameter, which specifies the number of items visible in the drop-down list at one time.

ComboBox skins

The ComboBox component shares the skins in the FScrollBar Skins and Global Skins folders (in the Component Skin folder in the library), with all other components that use scroll bars and bounding boxes. Customizing any of the skins in the FScrollBar or Global skins folders affects all Combo Box, ListBox, ScrollBar and ScrollPane component instances in the Flash document. For more information, see "Customizing component skins" on page 309.

The ListBox component

The ListBox component lets you add scrollable single- and multiple-selection list boxes to Flash movies. You add the items displayed in the ListBox using the Values dialog box that appears when you click in the labels or data parameter fields. You can also use the `FListBox.addItem` and `FListBox.addItemAt` methods to add items.

The ListBox component uses a zero-based index, where the item with index 0 is the first item displayed. When adding, removing, or replacing list items using the FListBox methods, you may need to specify the index of the list item.For more information, see the following entries in the online ActionScript Dictionary in the Help menu: `FListBox.addItemAt`, `FListBox.removeItemAt`, `FListBox.replaceItemAt`.

The ListBox component has the following standard built-in mouse and keyboard controls:

- The Up Arrow key moves the selection up one position.
- The Down Arrow key moves the selection down one position.
- PageUp moves the selection up one page. The size of the page is determined by the height of the list box instance.
- PageDown moves the selection down one page.
- Home selects the item at the top of the list.
- End selects the item at the bottom of the list.

ListBox parameters

You set the following parameters for each list box instance in your Flash document using the Parameters tab on the Property inspector or the Component Parameters panel.

Change Handler is the name of the function that you want to call when the user selects an item in the list box. This function must be defined in the same Timeline as the instance of the list box. This parameter is optional and only needs to be specified if you want an action to occur when the user selects an item in the combo box. For more information, see "Writing change handler functions for components" on page 304 and the `FListBox.setChangeHandler` method entry in the online ActionScript Dictionary in the Help menu..

Data is an array of text strings specifying the values associated with the items (labels) in the list box. You enter the text strings for the array using the Values dialog box or using the `FListBox.addItem` or `FListBox.addItemAt` methods to add items at runtime.

Labels is an array of text strings specifying the items in the list box. You enter the text strings for the array using the Values dialog box or using the `FListBox.addItem` or `FListBox.addItemAt` methods to add items at runtime.

Select Multiple specifies whether the user is allowed to select more than one item in the list box (`true`) or not (`false`). The default setting is `false`.

You can set additional options and functionality for list box instances using the methods of the FListBox (component) in the online ActionScript Dictionary.

Sizing ListBox components

You can change the width and height of list box instances using the Free Transform tool during authoring, or at runtime using the `FListBox.setSize` or `FListBox.setWidth` methods. The width of the list box instance is determined by the width of the bounding box. If the text of the list items is too long to fit inside the bounding box, the text is truncated. The height of the list box instance is automatically adjusted to display entire lines of text without increasing the size of the box, which may result in the list box being slightly smaller than the specified height, but never larger.

ListBox skins

The ListBox component shares the skins in the FScrollBar Skins and Global Skins folders in the Component Skin folder in the library with all other components that use scroll bars and bounding boxes. Customizing any of the skins in the FScrollBar or Global Skins folders affects all ComboBox, ListBox, ScrollBar and ScrollPane component instances in the Flash document. For more information, see "Customizing component skins" on page 309.

The PushButton component

The PushButton component lets you add simple push buttons to your Flash movie. The PushButton component accepts all standard mouse and keyboard interactions, and has an `onClick` parameter that allows you to easily specify a handler to execute actions when the button is released. You can use the methods of the FPushButton component to disable or enable the button, and resize the button without distortion at runtime. For more information, see the FPushButton (component) entry in the online ActionScript Dictionary in the Help menu.

PushButton parameters

You set the following parameters for each push button instance in your Flash document using the Parameters tab on the Property inspector or the Component Parameters panel.

Click Handler is a text string specifying the function to call when a user presses and releases the push button. The function must be defined in the same Timeline as the push button instance, and may accept the instance name of the push button as a parameter. For more information, see "Writing change handler functions for components" on page 304" and the `FPushButton.setClickHandler` entry in the online ActionScript Dictionary in the Help menu.

Label(s) is the text that appears on the push button.

You can set additional options and functionality for push button instances using the methods of the FPushButton (component). For more information, see the FPushButton (component) in the online ActionScript Dictionary in the Help menu.

Sizing PushButton components

You can size the height and width of push button instances using the Free Transform tool. If the push button is not large enough to display the label, the label text is truncated. You can set the size of push button instances at runtime using the `FPushButton.setSize` method.

If you use the ActionScript `_width` and `_height` properties to adjust the width and height of components, the component is resized but the layout of the label remains the same, which may result in distortion.

PushButton skins

The PushButton component uses the skins in the FPushButton Skins folder and the FLabel skin in the Global Skins folder in the Component Skins folder in the library. Customizing any of the PushButton component skins affects all push button instances in the Flash document. For more information, see "Customizing component skins" on page 309.

The RadioButton component

The RadioButton component lets you add groups of radio buttons to your Flash document. The `groupName` parameter logically groups radio button instances together and prevents more than one radio button in the same group from being selected at the same time.

RadioButton parameters

You can set the following parameters for each radio button instance in your Flash document using the Parameters tab on the Property inspector or the Component Parameters panel.

Change Handler is the name of the function that you want to execute when the user selects one of the radio buttons in a group. This function must be defined in the same Timeline as the radio button instances in the group. This parameter is optional and needs to be specified only if you want an action to take place as soon as the user selects a radio button. For more information, see "Writing change handler functions for components" on page 304 and the `FRadioButton.setChangeHandler` entry in the ActionScript Dictionary in the Help menu.

Data is the data associated with the radio button label. There is no default setting for this parameter.

Group Name specifies the radio button as one of a group of radio buttons.

Initial State specifies whether the radio button is initially selected (`true`), or clear (`false`). Only one radio button in a group (all having the same Group Name parameter) can have an initial state of `true` (selected). If more than one radio button instance has `true` specified for this parameter, the last radio button instance with an initial state parameter of `true` is selected. The default value for this parameter is `false`.

Label is the name of the radio button. By default the label is set to the right.

Label Placement specifies whether the label appears to the left or the right of the radio button. By default, the label is displayed to the right of the check box.

You can set additional options and functionality for radio button instances and groups using the methods of the FRadioButton (component) in the online ActionScript Dictionary in the Help menu.

Sizing RadioButton components

You can set the width, but not the height of RadioButton components during authoring using the Free Transform tool, or at runtime using the `FRadioButton.setSize` method. The hit area of the radio button instance is the size of the radio button and radio button label area. If the radio button instance is not large enough to display the label, the label text is truncated; if the instance is larger than the text, the hit area extends beyond the label.

RadioButton skins

The RadioButton component uses the skins in the FRadioButton Skins folder and the FLabel skin in the Global Skins folder in the Component Skins folder. Customizing any of the RadioButton component skins affects all radio button instances in the Flash document. For more information, see “Customizing component skins” on page 309.

The ScrollBar component

The ScrollBar component provides drag-and-drop functionality for adding vertical and horizontal scroll bars to dynamic and input text fields. Adding scroll bars to dynamic and input text fields allows the text field to accept large amounts of text without requiring that it all be displayed simultaneously.

The ScrollBar component is used by the ComboBox, ListBox, and ScrollPane components. Adding any of these components to a Flash document automatically adds the ScrollBar component to the library. If an instance of the ScrollBar component is already in the library, you can add instances of the ScrollBar component to the document by dragging them from the library.

Note: You should not add another copy of the ScrollBar component to the document by dragging it from the Components panel.

Advanced users and programmers can use the ScrollBar component with movie elements other than text fields when building applications or custom components in Flash. The ComboBox, ListBox, and ScrollPane components are examples of how the ScrollBar component can be used to build another component.

Adding scroll bars to input and dynamic text fields

When you drag a scroll bar onto a dynamic or input text field on the Stage, the scroll bar automatically snaps to the nearest side at the vertical or horizontal position where you place it. The scroll bar and the text field must be in the same Timeline.

Note: The ScrollBar component cannot be used with static text fields.

Once the scroll bar is snapped to the text field, Flash enters the instance name of the text field for the `targetTextField` parameter for the scroll bar instance in the Property inspector. Although the scroll bar automatically snaps to the text field, it is not grouped with the text field. Therefore, if you move or delete the text field, you must also move or delete the scroll bar.

To create a scrollable text field:

1. Use the Text tool to create a text field on the Stage.
2. Choose Window > Properties to open the Property inspector.
3. Choose Dynamic Text or Input Text from the Text type menu in the Property inspector.
4. Enter a name in the Instance Name field.
5. Choose Window > Components to open the Components panel.

   Note: If you have already used a ScrollBar or a component that uses scroll bars in your Flash document and have modified the scroll bar properties or skins in any way, drag the ScrollBar from the library, not the Components panel.
6. Drag a ScrollBar component onto the text field bounding box close to the side where you want to place a scroll bar.
7. Repeat step 6 to add additional ScrollBar components to the text field.
8. If you resize the text field, drag the ScrollBar off the text field and then onto it again to resize the scroll bar.

ScrollBar parameters

You can set the following parameters for each scroll bar instance in your Flash document using the Parameters tab on the Property inspector or the Component Parameters panel.

Horizontal specifies whether the scroll bar is horizontal (`true`) or vertical (`false`).

Target Text Field is a string specifying the instance name of the text field for the scroll bar. This parameter is automatically filled in with the instance name of the text field when the scroll bar snaps to the text field on the Stage. Changing or deleting this parameter disassociates the scroll bar from the text field on the Stage. You use the `FScrollBar.setScrollTarget` method to specify this parameter at runtime.

You can set additional options and functionality for scroll bar instances using the methods of the FScrollBar (component). Some of the methods of the FScrollBar component are not recommended for use with scroll bars attached to text fields.

Sizing ScrollBar components

Scroll bars added to text fields are automatically sized to fit the text field. If you resize the text field, the easiest way to resize a scroll bar instance is to drag it off the text field and then back on again. You should not use the `FScrollBar.setSize` method to size scroll bars attached to text fields.

Advanced users and programmers using the ScrollBar component to add scroll bars to movie elements other than text fields can size scroll bar instances using the Free Transform tool during authoring, or using the `FScrollBar.setSize` method at runtime.

ScrollBar skins

The ScrollBar component shares the skins in the FScrollBar Skins folder in the Component Skins folder in the library with all other components that use scroll bars. Customizing any of the skins in the FScrollBar Skins folder affects all ComboBox, ListBox, ScrollBar, and ScrollPane component instances in the Flash document. For more information, see "Customizing component skins" on page 309.

The ScrollPane component

The ScrollPane component lets you add window panes with vertical and horizontal scroll bars to display movie clips in Flash documents. The ScrollPane component is useful for displaying large areas of content without taking up a lot of Stage space. The ScrollPane component only displays movie clips. To add scroll bars to text fields, you use the ScrollBar component.

To display movie clips (or JPEG files that have been converted to movie clips) inside the ScrollPane, you specify the symbol linkage ID of the movie clip for the `scrollContent` parameter. The movie clip must be in the library to set the `scrollContent` parameter, but it need not be on the Stage. Also, the movie clip must also have the Export for ActionScript option selected in the Linkage Properties dialog box. For more information, see "Using movie clip event handler methods to trigger scripts" under Help > Using Flash.

You can display JPEG and SWF files loaded from a server in the scroll pane using the `FScrollPane.loadScrollContent()` method to specify the content for the scroll pane.

Note: Text in scroll panes must be displayed using embedded fonts. You cannot use device fonts to display text in scroll panes. See "About embedded fonts and device fonts" on page 136.

ScrollPane parameters

You can set the following parameters for each scroll pane instance in your Flash document using the Parameters tab on the Property inspector or the Component Parameters panel.

Drag Content specifies whether a user can drag the content in the scroll pane to change the view (`true`), or use the scroll bars to change the view (`false`). The default setting is `false`.

Horizontal Scroll determines whether a horizontal scroll bar is displayed (`true`), not displayed (`false`), or displayed only when necessary (`auto`). The default setting is `auto`.

Scroll Content is a text string specifying the symbol linkage ID of the movie clip to be displayed in the scroll pane.

Vertical Scroll determines whether a vertical scroll bar is displayed (`true`), not displayed (`false`), or displayed only when necessary (`auto`). The default setting is `auto`.

Sizing ScrollPane components

You can change the width and height of scroll pane instances during authoring using the Free Transform tool or at runtime using the `FScrollPane.setSize` method.

ScrollPane skins

The ScrollPane component shares the skins in the FScrollBar Skins folder (in the Component Skins folder in the library) with all other components that use scroll bars. Customizing any of the skins in the FScrollBar folder affects all ComboBox, ListBox, ScrollBar, and ScrollPane component instances in the Flash document. For more information about editing skins, see "Customizing component skins" on page 309.

Writing change handler functions for components

All components have a Change Handler parameter for specifying a change handler function that is called when the user selects a menu item, a radio button, or a check box. The Click Handler parameter for the PushButton component is equivalent to the Change Handler parameter. Specifying a function for the Change Handler parameter is optional. Using this function depends on the requirements and layout of your form or user interface, and the purpose of the component.

You can write Change Handler and Click Handler functions in a variety of ways. It is good coding practice to create a single handler function that specifies the actions for the components in your document, and then use the name of this handler function as the Change Handler parameter for the components. This ensures that conflicting actions are not assigned, and makes it easier to update and change the code. A Change Handler or Click Handler function always accepts at least one parameter, which is the instance of the component that has changed.

Single-selection forms

In the following example, `onChange` is a handler function specified for two CheckBox components. The handler function accepts an instance of a changed component as a parameter, uses a series of `if/else if` statements to determine which check box instance is selected and enables either `listBox1` or `listBox2` depending on the value of the check box instance.

```
function onChange(component)
{
if (component._name=="check1") {
  listBox1_mc.setEnabled(component.getValue());
} else if (component._name=="check2") {
  listBox2_mc.setEnabled(component.getValue());
}
}
```

Another way of accomplishing the same thing is to specify a different `changeHandler` function for each CheckBox component, as shown in the following example:

For the `check1` instance, specify `onCheck1` as the Change Handler parameter on the Parameters tab in the Property inspector. You must define the `onCheck1` function in the same Timeline as the `check1` component instance. If the user selects the `check1` instance of the check box, the list box instance `listBox1` is enabled.

```
function onCheck1(component)
{
  listBox1_mc.setEnabled(component.getValue());
}
```

For the check box instance `check2`, specify `onCheck2` for the Change Handler parameter on the Parameters tab in the Property inspector, and define the `onCheck2` function in the same Timeline as the `check2` component. If the user selects the `check2` instance of the check box, the list box instance `listBox2` is enabled.

```
function onCheck2(component)
{
  listBox2_mc.setEnabled(component.getValue());
}
```

Multiple-selection forms

In a form where the user makes multiple inputs or selections using various components and then submits the completed form, you need only specify a function for the Change Handler parameter for the component responsible for submitting the form data and exiting the form. The function needs to accept an instance of the component as a parameter, create an object with properties for storing the data, specify actions for gathering the data from all of the components in the form, and then perform an output, submit, or exit page action.

The following example is an `onClick` function specified for a Submit button on a form that has a check box, a group of radio buttons, and a list box. The user makes choices before pressing the Submit button to submit the form. The labels of the selected components are written to the Output window.

```
function onClick( component ) {
if ( component._name == "submit"){

// create the object to store values
formData = new Object();
formdata.checkValue = "";
formData.radioValue = "";
formData.listValue = "";

// gather the data
formData.checkValue = checkBox_mc.getValue();
formData.radioValue = radioGroup.getValue();
formData.listValue = listBox_mc.getValue();

// output the results
trace(formData.listValue);
trace(formData.radioValue);
trace(formData.checkValue);
  }
}
```

Customizing component colors and text

You can change color and text properties of a single instance of a Flash UI component using the `setStyleProperty` method. You can globally change the appearance of all components in a document using the `globalStyleFormat` object. You can also create new custom style formats for individual components using the FStyleFormat object.

Changes made to style format properties are not displayed when viewing components on the Stage using the Live Preview feature. For more information, see “Working with components in Live Preview” on page 293.

Changing properties of a component instance

You can specify a `FStyleFormat` property for a specific instance of a Flash UI component using the `setStyleProperty` method available to all Flash UI components. Calling the `setStyleProperty` method to set a property overrides the setting for that property only. All other property settings are unchanged. It is recommended that you use a separate layer in the document Timeline to set properties.

It is not recommended to call the `setStyleProperty` method multiple times to set more than one property. If you want to change multiple properties, or change properties for multiple component instances, you should create a custom style format. See “Changing the properties of specific components” on page 307.

To set or change a property for a single component instance:

1. Select the component instance on the Stage.
2. Create a new layer in the Timeline and give it a name.
3. Select any frame in the new layer.
4. Open the Actions panel in expert mode.
5. Use the following syntax to specify a property and value for the instance:

   ```
   componentInstance.setStyleProperty("property", value);
   ```

   For example, to make the arrow of the `grapeComboBox` instance purple:

   ```
   grapeComboBox.setStyleProperty("arrow", 0x800080);
   ```
6. Choose Control > Test Movie to view the changes.

Changing properties of all Flash UI components

The `globalStyleFormat` object is assigned to all Flash UI components. If you change a property setting of the `globalStyleFormat` object, the change is applied to all components in your Flash document. The global style format is accessible if you have placed at least one component instance on the Stage. It is recommended that you use a separate layer in the document Timeline to set properties.

To change one or more properties in the global style format:

1. Make sure the document contains at least one component instance. See "Adding components to Flash documents" on page 293.
2. Create a new layer in the Timeline and give it a name.
3. Select any frame in the new layer.
4. Open the Actions panel in expert mode.
5. Use the following syntax to change any or all of the properties listed in the style format properties table. You only need to list the properties whose values you want to change.

   ```
   globalStyleFormat.darkshadow = 0x333300;
   globalStyleFormat.shadow = 0x99cc00;
   globalStyleFormat.highlight3D = 0x333300;
   globalStyleFormat.highlight = 0x99cc00;
   globalStyleFormat.face = 0x99cc99;
   globalStyleFormat.background = 0xffffff;
   globalStyleFormat.text = 0x000000;
   globalStyleFormat.radioDot = 0x333300;
   globalStyleFormat.check = 0x333300;
   globalStyleFormat.arrow = 0x333300;
   ```
6. Following the list of object properties, use the following syntax to insert the method for the `globalStyleFormat` object:

- To update all properties in the `globalStyleFormat` object (including those whose values you are not changing), enter `globalStyleFormat.applyChanges();`. Properties whose values you are not changing will be updated with the same values.

- To update only those properties in the `globalStyleFormat` object whose values you are changing, enter `globalStyleFormat.applyChanges("propertyname1", "propertyname2");`, where propertyname1, propertyname2, and so on, refer to the names of the properties being updated. For example, if you changed the properties `check` and `arrow` in step 5, enter `globalStyleFormat.applyChanges("check", "arrow");`.

7 Choose Control > Test Movie to see the changes.

Changing the properties of specific components

You can create custom style formats to specify a unique set of properties for specific components in your Flash document. You use the FStyleFormat object constructor to create a new instance of the FStyleFormat object, to define your custom style format, and to specify the properties and values for the format. The FStyleFormat object is accessible if you have placed at least one component instance on the Stage. It is recommended that you use a separate layer in the document Timeline to set properties.

You make changes to a custom style format in the same way that you edit the properties in the global style format. Instead of the `globalStyleFormat` object name, use the `FStyleFormat` object name. See "Changing properties of all Flash UI components" on page 306.

The FStyleFormat object has three methods in addition to the constructor method:

- `FStyleFormat.applyChanges` applies changes you make to the properties of a custom style format you have created.
- `FStyleFormat.addListener` assigns components the style format.
- `FStyleFormat.removeListener` removes components from the style format.

For more information about using the methods and defining the properties of the FStyleFormat object, see its entry in the online ActionScript Dictionary in the Help menu.

The following table summarizes the properties of the FStyleFormat object. For a complete description of each property, see the individual entries in the online ActionScript Dictionary. You can set or change any of the FStyleFormat object properties in the global style format, or in custom style formats that you create.

Property summary for the FStyleFormat object

Property	Description
`FStyleFormat.arrow`	The color of the arrow used in scroll bars and drop-down lists.
`FStyleFormat.background`	The color of the background portion of components.
`FStyleFormat.backgroundDisabled`	The color of the background portion of disabled components.
`FStyleFormat.check`	The color of the check in a selected check box.
`FStyleFormat.darkshadow`	The color of the inner border or shadow portion of a component.
`FStyleFormat.face`	The main color of the component.
`FStyleFormat.foregroundDisabled`	The foreground color of a disabled component.
`FStyleFormat.highlight`	The color of the inner border or darker shadow portion of a component when it is selected.
`FStyleFormat.highlight3D`	The color for the outer border or light shadow portion of a component when it is selected.

Property	Description
FStyleFormat.radioDot	The color of the dot in a selected radio button.
FStyleFormat.scrollTrack	The color of the track in a scroll bar.
FStyleFormat.selection	The color of the selection bar highlighting an item in a list of a component.
FStyleFormat.selectionDisabled	The color of the selection bar that highlights an item in a list of a disabled component.
FStyleFormat.selectionUnfocused	The color of the selection bar (highlighting) when the component does not have keyboard focus.
FStyleFormat.shadow	The color of the outer border or light shadow portion of a component.
FStyleFormat.textAlign	Specifies left, right, or center alignment for text displayed in or on a component.
FStyleFormat.textBold	Specifies whether text is bold (`true`) or not (`false`).
FStyleFormat.textColor	The color of list items in a component when they are not selected.
FStyleFormat.textDisabled	The color of the text in a disabled component.
FStyleFormat.textFont	The name of the font to display text.
FStyleFormat.textIndent	The indentation of the text from the left margin to the first text character in pixels.
FStyleFormat.textItalic	Specifies whether text is italic (true) or not (false).
FStyleFormat.textLeftMargin	The left paragraph margin in text in pixels.
FStyleFormat.textRightMargin	The right paragraph margin for text in pixels.
FStyleFormat.textSelected	The color of a selected list item in a component.
FStyleFormat.textSize	The size of the text in points.
FStyleFormat.textUnderline	Specifies whether text is underlined (true) or not (false).

To create a custom style format for specific components:

1. Make sure the document contains at least one component instance. See "Adding components to Flash documents" on page 293.
2. Create a new layer in the Timeline and give it a name.
3. Select any frame in the new layer.
4. Open the Actions panel in expert mode.
5. Use the following syntax to create an instance of the FStyleFormat object to define the new custom style format:

```
var myStyleFormat = new FStyleFormat();
```

6 In the same Script pane as above, use the following syntax to specify the properties you want to define for the `myStyleFormat` object:

```
myStyleFormat.arrow  = 0x333300;
myStyleFormat.background = 0xffffff;
myStyleFormat.backgroundDisabled  = 0xccffcc;
myStyleFormat.darkshadow = 0x333300;
myStyleFormat.foregroundDisabled  = 0x999999;
myStyleFormat.face = 0x99cc99;
myStyleFormat.textSize = 12
myStyleFormat.highlight = 0x99cc00;
myStyleFormat.highlight3D  = 0x333300;
myStyleFormat.radioDot = 0x333300;
myStyleFormat.scrollTrack = 0x99cc99;
myStyleFormat.selection = 0x333300;
myStyleFormat.selectionDisabled = 0x999999;
myStyleFormat.selectionUnfocused = 0x999999;
myStyleFormat.shadow = 0x99cc00;
myStyleFormat.textColor = 0x000000;
myStyleFormat.textDisabled  = 0x999999;
myStyleFormat.textSelected = 0x000000;
```

In the same Script pane, use the following syntax to assign the format style to specific components.

```
myStyleFormat.addListener(myComboBox, myListBox);
```

7 Use the following syntax to remove a component from a style format:

```
myStyleFormat.removeListener(myComboBox);
```

Customizing component skins

The Component Skins folder in the library has a Skins folder containing the skins used by all types of component in your Flash document. Some components share skins. Components that use scroll bars—including ComboBox, ListBox, ScrollBar, and ScrollPane—share the skins in the FScrollBar Skins folder, and the ListBox component uses the skins in the ComboBox Skins folder.

You can customize component skins in two ways:

- Edit the skins on the Stage.
- Create new graphics and break the graphics into skin elements. Then register the elements to a component by editing the code in the first frame of the Read Me layer of each skin in the library.

Both methods of customizing component graphics update all instances of the components that use those skins. It is not possible to customize the skins for only one instance of a component.

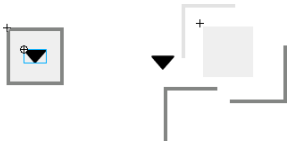
The method you use to customize component graphics—editing the symbols in the library or registering new skin elements to the component—depends on how you plan to use the components. If you do not plan to change the color or text properties of the component in the future, editing the symbols in the library is the easier option. If you want to use custom style formats to change the color and text properties of a customized component, creating new graphics and registering the skin elements is the more efficient option.

Changes made to component skins are not displayed when viewing components on the Stage using Live Preview.

About creating and registering skin elements

The best way to understand the process of creating graphics and registering skin elements is to dissect one of the graphic symbols in the library, and view the Read Me file on the first layer of the Timeline. Double-click the symbol in the library, open the Read Me file on the first layer of the Timeline, and pull apart the symbol on the Stage.

The following image shows the `fcb_downArrow` symbol as it appears whole and as separate skin elements.



In the dissected version, you can see that the `fcb_downArrow` symbol is made up of several skin elements. You can view the names of selected skin elements in the Property inspector. The `fcb_downArrow` symbol has six skin elements: `arrow_mc`, `shadow_mc`, `darkshadow_mc`, `highlight_mc`, and `highlight3D_mc`.

Each skin symbol is a movie clip registered to the component and associated with properties of the FStyleFormat object in the Read Me file for the movie clip. To register a skin element to a component, you enter the name for the skin element in the first frame of the Read Me layer for the skin symbol which contains the skin element.

The following code from the Read Me file for the `fcb_downArrow` symbol shows how the movie clip skin elements are registered to properties of the FStyleFormat object.

```
component.registerSkinElement(arrow_mc, "arrow");
component.registerSkinElement(face_mc, "face");
component.registerSkinElement(shadow_mc, "shadow");
component.registerSkinElement(darkshadow_mc, "darkshadow");
component.registerSkinElement(highlight_mc, "highlight");
component.registerSkinElement(highlight3D_mc, "highlight3D");
```

Editing component skins in the library

When you edit a skin, you must maintain the same registration point as the skin had originally in order for the skin to display correctly. The upper left corner of all edited symbols must be at (0,0).

To edit a graphic symbol in the Component Skins folder:

1. Open the Component Skins folder in the library.
2. Open the Skins folder of the component you want to edit.
3. Double-click the skin movie clip you want to edit.
4. Modify the movie clip or delete it and create a new one.
5. When you have finished editing the skin movie clip, click the Back button at the left side of the information bar at the top of the Stage to return to edit document mode.
6. Place an instance of the component that uses the edited skin onto the Stage.
7. Choose Control > Test Movie to view the component with the edited skin.

Creating and registering new skin elements for a component

To create custom component skins that you can update using style formats, you need to update the code in the first frame of the Read Me layer with the instances of the skin elements you create, and the `FStyleFormat` property to assign to the element.

Registering a skin element to a property applies the value assigned to that property in the style format to the skin element. If the property is a new property, you need to define the property and a value in the component's style format. The `registerSkinElement` method is available to all Flash UI components. For more information, see the `registerSkinElement` entry for each component in the online ActionScript Dictionary in the Help menu.

To create new skin elements and register them to a component:

1. Open the Skins folder for a component in the library.
2. Double-click the skin movie clip for which you want to create new skin elements.
3. Modify the movie clip or delete it and create a new one.
4. Break the graphic into separate skin elements, and save each element as a movie clip symbol. Give each skin element a unique name. For more information, see "About creating and registering skin elements" on page 310.
5. Click on the first frame of the ReadMe layer of the skin movie clip that you selected in step 2.
6. Open the Actions panel in expert mode.
7. Replace the name of the original skin elements in the movie clip with the names of the new skin elements that you created in step 4.

Restoring default component skins

If you want to restore the default skins used by a Flash UI component, use the Components panel to first add a new component of the same type to your Flash document and then choose Replace Existing Items in the warning dialog box that appears; your edited graphics are replaced with a new set of the default component graphic symbols. If you want to preserve your edited symbols, be sure to choose Use Existing Items when adding new components using the Components panel.

1. Choose Window > Components.
2. Do one of the following:

- Drag a component from the Components panel to the Stage.
- Double-click a component in the Components panel.

3. Choose Replace Existing Items (not undoable) to replace all the skins with default skins. The new component and all previous versions of the component, or of components that share its skins, will use the default skins.

Creating forms using components

Flash forms provide an advanced type of interactivity—a combination of user interface elements, movies, and text fields that let you pass information to another application on a local or remote server. Using the Flash UI components in conjunction with other Flash functionality, you can create sophisticated Web forms and user interfaces for intranets or applications.

For introductory information on using components to create a form, see the Introduction to Components Tutorial (Help > Tutorials > Introduction to Components). You should complete the tutorial before using the information in this chapter.

About the FormExample.fla example

The information in this section references the FormExample.fla file in the folder Flash MX/Tutorials/Components/ and assumes you have completed the Introduction to Components Tutorial and understand the basics of assembling a form using the Flash UI components.

The FormExample.fla is a three-page form:

- On page 1, users enter information about themselves using a combination of Flash UI components and text fields. The page includes a text input field for entering the name, RadioButton component instances for specifying gender, a ComboBox component for selecting a city, and a PushButton component to go to page 2.
- On page 2, a CheckBox component lets users request to receive more information. A ListBox component lets users indicated their interests. PushButton components allow users to return to page 1 or proceed to page 3.
- On page 3, the information the user has entered on the previous pages is displayed in a series of dynamic text fields.

Planning the form

You should determine the following criteria before you begin building a form:

- Which elements your form needs
- What data each form element displays and gathers
- Where each user interface element appears in the form
- How the user will navigate the form

Once you have determined the elements in your form, you can develop a data model to gather and store the data. The gathered data is displayed in the form and enables form elements as the user navigates through the form. When the user submits the form, the data is transmitted to a database.

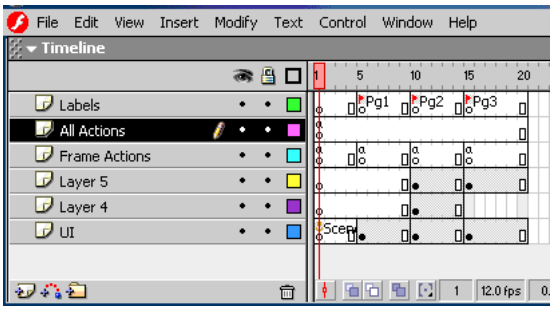
The following table lays out all of the code elements used in the FormExample.fla, including the functions they perform and the ActionScript elements they reference in the code. This table is an aid for examining the code in the in the sections that follow it, and an example of how to plan and organize forms you create.

Form element	Function of element	ActionScript reference
Page 1	Gather user information	`pg1` `updateUIFromDataPg1`
Name input field	Gather user name	`name_txt.text` - field `nameField` - object property
Gender radio buttons	Gather gender info	`genderGroup` - component `gender` - object property
City drop-down list	Gather city info	`city_mc` - component; `cityIndex` -object property; `cityTable` - array for component
Next button	Take user to page two	`pg1next`
Page 2	Gather interest information	`pg2` `updateUIFromDataPg2`
Information request check box	Enable interest list menu and flag user for more info	`junkCheck_mc` - component `junkMail` - object property
Interest list menu	Gather interest data	`interest_mc` -component `interestIndex` - object property; `interestTable` - array for component
Next button	Take user to page three	`pg2next`
Previous button	Return user to page one	`pg2prev`
Page 3	Display user and interest information	`pg3` `updateUIFromDataPg3`
Title – static text	Display "Finished" message	N/A
Name – dynamic text	Display user name	`resultsName_txt.txt` - display field name
Gender – dynamic text	Display Gender radio button selection	`resultsGender_txt.txt` - display field name
City – dynamic text	Display City combo box selection	`resultsCity_txt.txt` - display field name
Interests – dynamic text	Display Interest list menu selection	`resultsInterest_txt.txt` - display field name
Previous button	Return user to page two	`pg3prev`

Storing form data

An essential part of all forms is storing and updating the data entered by the user. The form data needs to be current and available to all pages of the form at all times, which means that the ActionScript for initializing the form and storing the data must precede all form pages in the Timeline.

In the FormExample.fla file, all of the code is defined in an Actions layer in frame 1 of the Timeline.



The Timeline showing the layer structure of the form

The following code initializes the form, creates the object defining the properties to store the data and set the initial values for each form element, and creates arrays to populate the list and combo boxes used in the form:

```
function initData()
{
// this function is called in frame 1 of the Frame Actions layer
// the following code ensures the form is only initialized once

  if ( inited )
  return;
  inited = true;

// create an object with properties to store the data and set the initial
// values or each UI element

  loginData = new Object();
  loginData.nameField ="";
  loginData.gender = "Female";
  loginData.cityIndex = 1;
  loginData.junkMail = true;
  loginData.interestIndex = undefined;

// define the arrays to populate the list and combo boxes in the form

cityTable = new Array("Berkeley", "San Francisco", "San Jose", "Oakland",
"El Cerrito", "Walnut Creek");
interestTable = new Array("Golf", "Ski", "Flash Programming", "Hiking");

}
```

The `initData` function is called from frame 1 in a Frame Actions layer using the following code:

```
initData();
```

Once you have initialized your form and established a means for storing the data entered by the user, you use the data to navigate and display the form pages.

Managing and monitoring the data

The data entered on each page of a multipage form can affect what form elements or pages are displayed and how the pages are displayed. To use the data entered by the user to display the form pages with the current data, you need to create functions for each page that retrieve the data and update the page display.

Example code from FormExample.fla

In the FormExample.fla file, the data stored in the `loginData` object properties is maintained by `getDataFromUI` and `updateUI` functions defined for each form page in frame 1 of the Actions layer. The `getDataFromUI` and `updateUI` functions for each form page are defined in the first frame of the Actions layer along with the `loginData` object in order to keep all the form actions together. The FormExample.fla file splits the getting and setting of the data into two functions—`getDataFromUI` and `updateUI`—in order to highlight the necessary code clearly, but in an actual form, these two functions could be combined into a single function for each page.

The `updateUI` function for each form page is called from the first frame of the form page in the Frame Actions layer, as shown in the following code.

In the Frame Actions layer, on the first frame of page 1 of the form:

```
stop();
updateUIFromDataPg1();
```

In the Frame Actions layer, on the first frame of page 2 of the form:

```
stop();
updateUIFromDataPg2();
```

In the Frame Actions layer, on the first frame of page 3 of the form:

```
stop();
updateUIFromDataPg3();

// get the data from the UI elements on page 1

function getDataFromUIPg1()
{
  loginData.nameField = name_txt.text;
  loginData.gender = genderGroup.getValue().getLabel();
  loginData.cityIndex = city_mc.getSelectedIndex();
}

// get the data from the UI elements on page 2

function getDataFromUIPg2()
{
  loginData.junkMail = junkCheck_mc.getValue();
  loginData.interestIndex = interest_mc.getSelectedIndex();
}

// get the data from the UI elements on page 3

function getDataFromUIPg3()
{
// page 3 only displays data, so there is no data to get
}

// set the state of UI elements on page 1 using the loginData object values

function updateUIFromDataPg1()
{
  name_txt.text = loginData.nameField;
  for (var i=0; i<cityTable.length; i++) {
  city_mc.addItem(cityTable[i]);
}
  city_mc.setSelectedIndex(loginData.stateIndex);
  genderGroup.setValue(loginData.gender + "_mc");
}

// set the state of UI elements on page 2 using the loginData object values

function updateUIFromDataPg2()
{
  for (var i=0; i<interestTable.length; i++) {
  interest_mc.addItem(interestTable[i]);
}
  interest_mc.setSelectedIndex(loginData.interestIndex);
  junkCheck_mc.setValue(loginData.junkMail);
  onChange();
}

// display the results data on page 3 using loginData object values

function updateUIFromDataPg3()
{
  resultsName_txt.text = loginData.nameField;
  resultsGender_txt.text = loginData.gender;
  resultsState_txt.text = stateTable[loginData.stateIndex];
  resultsInterests_txt.text = interestTable[loginData.interestIndex];
}
```

Once you have established a means of managing your data, you can set up the navigation of the form.

Using the data to navigate and display the form pages

The Next and Previous buttons you create to navigate a multipage form need to contain actions to take the user to the correct page and display the page using the data entered by the user. The code for a Previous button returns the user to the previous page and displays the page with the information entered. The code for a Next button takes the user to the next page.

The `onClick` handler in the first frame of the Actions layer defines the actions for the Next and Previous push buttons on all of the form pages. The handler uses `if` and `else if` statements to determine which push button is currently being released, and specifies the appropriate navigation action. The navigation actions call the `getDataFromUI` function, discussed in "Managing and monitoring the data" on page 315. The `onClick` function is specified for the Click Handler parameter on the Parameters tab of the Property inspector for each push button instance.

In the example below, the `onClick` handler is used to navigate the pages in the form:

```
function onClick(btn)
{
if ( btn == pg1next ) {

// next button on page 1
  getDataFromUIPg1();//get data from UI components on page 1
  gotoAndStop("pg2");// go to pg 2
} else if ( btn == pg2prev ) {

// prev button on page 2
  getDataFromUIPg2();// get data from UI components on page 2
  gotoAndStop("pg1");//go to pg 1
} else if ( btn == pg2next ) {

// next button on page 2
  getDataFromUIPg2();//get data from UI components on page 2
  gotoAndStop("pg3");//go to pg 3
} else if ( btn == pg3prev ) {

// prev button on page 3
  getDataFromUIPg3();//get data from UI components on page 3
  gotoAndStop("pg2");// go to pg 2
}
}
```

The `onChange` handler in frame 1 of the Actions layer defines the actions for the check box instance on page 2 of the form. This is an example of a component controlling the enabled state of another component. The check box is selected by default. If the user leaves the check box selected, the list box allows the user to make selections. If the user deselects the check box, the list box is disabled. The `onChange` function is specified for the Change Handler parameter on the Parameters tab of the Property inspector for the check box instance.

In the example below, the `onChange` handler is used to navigate the pages in the form:

```
function onChange(control)
{
if ( control == junkCheck_mc ) {
// enable and disable the list box based on check box value
  interest_mc.setEnabled(junkCheck_mc.getValue());
}
}
```

CHAPTER 16
Connecting with External Sources

Macromedia Flash MX movies can send information to and load information from external sources. For example, you can use actions and methods to communicate with server-side scripts, text files, and XML files.

You can load JPEG images and MP3 sound files into your movie as it plays. This allows you to update an image or sound without having to republish the original Flash Player (SWF) file.

To extend Flash so that it can send and receive messages from the movie's host environment—for example, the Flash Player or a JavaScript function in a Web browser—you can use `fscommand` and Flash Player methods.

Flash also provides components that you can drag and drop to create Web applications. Like built-in objects, components have predefined methods and properties, but they are reusable movie clips. For more information, see Chapter 15, "Using Components," on page 289.

Sending and loading variables to and from a remote source

A Flash movie is a window for capturing and displaying information, much like an HTML page. However, Flash movies can stay loaded in the browser and continuously update with new information without having to reload the entire page. Using Flash actions and object methods, you can send information to and receive information from server-side scripts, text files, and XML files. You can also load JPEG and MP3 files from a remote source into a Flash movie while the movie plays.

In addition, server-side scripts can request specific information from a database and relay it to a Flash movie. Server-side scripts can be written in many different languages: some of the most common are Perl, ASP (Microsoft Active Server Pages), and PHP. By storing information in a database and retrieving it, you can create dynamic and personalized content for your movie. For example, you could create a message board, personal profiles for users, or a shopping cart that keeps track of a user's purchases so that it can determine the user's preferences.

Several ActionScript actions and methods allow you to pass information into and out of a movie. Each action and method uses a protocol to transfer information, and requires information to be formatted in a certain way.

- The MovieClip object methods that use HTTP or HTTPS protocol to send information in URL-encoded format are `getURL`, `loadVariables`, `loadVariablesNum`, `loadMovie`, and `loadMovieNum`.
- The LoadVars object methods that use HTTP or HTTPS protocol to send information in URL-encoded format are `load`, `send`, and `sendAndLoad`.
- The Sound object method that uses HTTP and HTTPS protocol to load sounds is `loadSound`.

- The ActionScript elements that use HTTP or HTTPS protocol to load JPEG images into a Flash movie are `loadMovie` and `loadMovieNum`.
- The methods that use HTTP or HTTPS protocol to send information as XML are `XML.send`, `XML.load`, and `XML.sendAndLoad`.
- The methods that create and use a TCP/IP socket connection to send information as XML are `XMLSocket.connect` and `XMLSocket.send`.

Loading data securely

When playing a Flash document in a Web browser, you can load data into the document only from a file that is on a server in the same subdomain. This prevents Flash documents from being able to download information from other people's servers.

To determine the subdomain of a URL consisting of one or two components, use the entire domain:

Domain	Subdomain
http://macromedia	macromedia
http://macromedia.com	macromedia.com

To determine the subdomain of a URL consisting of more than two components, remove the last level:

Domain	Subdomain
http://x.y.macromedia.com	y.macromedia.com
http://www.macromedia.com	macromedia.com

The following chart shows how the Flash Player determines whether to permit an HTTP request:

When you use the XMLSocket object to create a socket connection with a server, you must use a port numbered 1024 or higher. (Ports with lower numbers are commonly used for Telnet, FTP, the World Wide Web, or Finger.)

Flash relies on standard browser and HTTP and HTTPS security features. Essentially, Flash offers the same security that is available with standard HTML. You should follow the same rules that you follow when building secure HTML Web sites. For example, to support secure passwords in Flash, establish your password authentication with a request to a Web server.

To create a password, use a text field to request a password from the user. Submit it to a server in a `loadVariables` action or in an `XML.sendAndLoad` method using an HTTPS URL with the `POST` method. The Web server can then verify whether the password is valid. This way, the password will never be available in the SWF file.

Checking for loaded data

Each action and method that loads data into a movie (except `XMLSocket.send`) is *asynchronous*: the results of the action are returned at an indeterminate time.

Before you can use loaded data in a movie, you must check to see if it has been loaded. For example, you can't load variables and manipulate their values in the same script. In the following script, you can't use the variable `lastFrameVisited` until you're sure the variable has loaded from the file `myData.txt`:

```
loadVariables("myData.txt", 0);
gotoAndPlay(lastFrameVisited);
```

Each action and method has a specific technique you can use to check data it has loaded. If you use the `loadVariables` or `loadMovie` action you can load information into a movie clip target and use the `data` event of the `onClipEvent` action to execute a script. If you use the `loadVariables` action to load the data, the `onClipEvent(data)` action executes when the last variable is loaded. If you use the `loadMovie` action to load the data, the `onClipEvent(data)` action executes each time a fragment of the movie is streamed into the Flash Player.

For example, the following button action loads the variables from the file `myData.txt` into the movie clip `loadTargetMC`:

```
on(release){
  loadVariables("myData.txt", _root.loadTargetMC);
}
```

An action assigned to the `loadTargetMC` instance uses the variable `lastFrameVisited`, which is loaded from the file `myData.txt`. The following action will execute only after all the variables, including `lastFrameVisited`, are loaded:

```
onClipEvent(data) {
  goToAndPlay(lastFrameVisited);
}
```

If you use the `XML.load` and `XMLSocket.connect` methods, you can define a handler that will process the data when it arrives. This handler is a property of the XML or XMLSocket object to which you assign a function you have defined. The handlers are called automatically when the information is received. For the XML object, use `XML.onLoad`. For the XMLSocket object, use `XMLSocket.onConnect`.

For more information, see "Using the XML object" on page 325 and "Using the XMLSocket object" on page 328.

Using HTTP to connect to server-side scripts

The `loadVariables`, `loadVariablesNum`, `getURL`, `loadMovie`, and `loadMovieNum` actions all communicate with server-side scripts using the HTTP protocol. These actions send all the variables from the Timeline to which the action is attached. When used as methods of the MovieClip object, `loadVariables`, `getURL`, and `loadMovie` send all the variables of the specified movie clip; each action (or method) handles its response as follows:

- `getURL` returns any information to a browser window, not to the Flash Player.
- `loadVariables` loads variables into a specified Timeline or level in the Flash Player.
- `loadMovie` loads a movie into a specified level or movie clip in the Flash Player.

When you use the `loadVariables`, `getURL`, or `loadMovie` actions, you can specify several parameters:

- *`URL`* is the file in which the remote variables reside.
- *`Location`* is the level or target in the movie that receives the variables. (The `getURL` action does not take this parameter.)

  For more information about levels and targets, see "About multiple Timelines" on page 246.
- *`Variables`* sets the HTTP method, either `GET` or `POST`, by which the variables will be sent. When omitted, the Player defaults to `GET`, but no variables are sent.

For example, if you wanted to track the high scores for a game, you could store the scores on a server and use a `loadVariables` action to load them into the movie each time someone played the game. The action might look like this:

```
loadVariables("http://www.mySite.com/scripts/high_score.php", _root.scoreClip,
  GET);
```

This loads variables from the PHP script called `high_score.php` into the movie clip instance `scoreClip` using the `GET` HTTP method.

Any variables loaded with the `loadVariables` action must be in the standard MIME format *application/x-www-urlformencoded* (a standard format used by CGI scripts). The file you specify in the *`URL`* parameter of the `loadVariables` action must write out the variable and value pairs in this format so that Flash can read them. This file can specify any number of variables; variable and value pairs must be separated with an ampersand (`&`), and words within a value must be separated with a plus (+). For example, this phrase defines several variables:

For more information on `loadVariables`, `getURL`, `loadMovie`, and the LoadVars object, see their entries in the online ActionScript Dictionary in the Help menu.

Using the LoadVars object

You can use the LoadVars object instead of `loadVariables` to transfer variables between a Flash movie and a server. The LoadVars object lets you send all the variables in an object to a specified URL and load all the variables at a specified URL into an object. The response from the server triggers the `LoadVars.onLoad` method and sets variables in the target. You can use LoadVars to obtain error information and progress indications and to stream the data while it downloads.

The LoadVars object is similar to the XML object; it uses the methods `load`, `send`, and `sendAndLoad` to initiate communication with the server. The main difference between the LoadVars and XML objects is that the LoadVars data is a property of the LoadVars object, rather than an XML DOM (Document Object Model) tree stored in the XML object.

You must create a new instance of the LoadVars object to call its methods. This instance is a container to hold the loaded data.

To load data with the LoadVars object:

1. Choose a frame, button, or movie clip to which to assign the action.
2. Choose Window > Actions to open the Actions panel if it isn't already open.
3. In the Actions toolbox, click the Actions category, click Variables, and double-click the `set variable` action to add it to the Script pane.

4 In the Variable parameter box, enter an instance name for the new object, for example, `myLoadVars`.

5 With the insertion point in the Value parameter box, from the Actions toolbox, click the Objects category, then click Client/Server, click LoadVars, and double-click `new LoadVars` to add it to the Script pane. Select the Expression box.

The code should look like this:

```
myLoadVars = new LoadVars();<<Loc: added empty parens here --IMD>>
```

6 In the Actions toolbox, click the Objects category, click Client/Server, LoadVars, and Methods, and double-click the `load` method to add it to the Script pane.

7 In the Object parameter box, enter the instance name of the LoadVars object into which the data will load—in this example, `myLoadVars`.

8 In the Parameters box, enter the URL from which to download data.

The URL must be enclosed in quotation marks, for example, `"http://www.myserver.com/data.txt"`. The finished code should look like this:

```
myLoadVars = new LoadVars();<<Loc: added empty parens here --IMD>>
myLoadVars.load("http://www.myserver.com/data.txt");
```

For more information, see the LoadVars object in the online ActionScript Dictionary in the Help menu.

Loading an image or sound dynamically

If you import an image or a sound while you author a Flash document, the image and sound are then packaged and stored in the SWF file when you publish the movie. To load JPEG images at runtime, you use the `loadMovie` or `loadMovieNum` method of the MovieClip object. To load MP3 sounds at runtime, you use the `loadSound` method of the Sound object. To return the number of bytes that have downloaded and the expected number of bytes for the image or sound file being downloaded, you use the `getBytesLoaded` and `getBytesTotal` methods of the MovieClip and Sound objects.

To load an image into a level in the Flash Player, you must use the `loadMovieNum` method or action. To load an image into a movie clip target in the Flash Player, you must use the `loadMovie` action or method. The loaded image replaces all the contents of the target movie clip.

To load a sound, you must create a new instance of the Sound object. You can use the new instance to call the `loadSound` method to load an event or a streaming sound. Event sounds are loaded completely before being played; streaming sounds are played as they are downloaded. You can set the `isStreaming` parameter of the `loadSound` method to specify a sound as an event sound or a streaming sound. After you load an event sound, you must call the `start` method of the Sound object to make the sound play. Streaming sounds begin playing when sufficient data is loaded into the movie; you don't need to use the `start` method.

Note: For image files, Flash supports only the standard JPEG image file type, not progressive JPEG files. For sound files, Flash supports only the MP3 sound file type.

To load an image dynamically:

1 Choose a frame, button, or movie clip to which to assign the action.

2 Choose Window > Actions to open the Actions panel if it isn't already open.

3 In the Actions toolbox, click the Objects category, click Movie, MovieClip, and Methods, and double-click the `loadMovie` method to add it to the Script pane.

4 In the Object parameter box, enter the instance name of the movie clip into which the image will load—in this example, `myMC`.

   If the movie clip is not a child of the same parent as the Timeline that calls the action, you must use a target path. You can use an absolute or relative path.

5 In the Parameters box, enter the URL at which the image is located. Enter a comma (,) after the URL.

6 After the comma in the Parameters box, enter the HTTP method `"GET"` or `"POST"` (in quotation marks), or leave the parameter blank.

   For example, the following code loads an image into a movie clip on the Timeline where the movie clip that calls the action is located.

```
myMC.loadMovie("http://www.foo.com/ImagesToLoad/image1.jpg")
```

To load a sound dynamically:

1 Choose a frame, button, or movie clip to which to assign the action.

2 Choose Window > Actions to open the Actions panel if it isn't already open.

3 In the Actions toolbox, click the Actions category, click Variables, and double-click the `set variable` action to add it to the Script pane.

4 In the Variable parameter box, enter an instance name for the new object, for example, `mySound`.

5 With the insertion point in the Value parameter box, from the Actions toolbox, click the Objects category, click Movie, click Sound, and double-click `new Sound` to add it to the Script pane. Select the Expression box.

   The code should look like this:

```
mySound = new Sound();
```

6 In the Actions toolbox, click the Objects category, click Movie, Sound, and Methods, and double-click the `loadSound` method to add it to the Script pane.

7 In the Object parameter box, enter the instance name of the movie clip into which the sound will load—in this example, `mySound`.

8 In the Parameters box, enter the URL at which the sound is located. Enter a comma (,) after the URL.

   The URL must be enclosed in quotation marks, for example, `"http://www.foo.com/SoundsToLoad/sound14.mp3"`.

9 After the comma in the Parameters box, enter the value `false` for the `isStreaming` parameter to indicate that the sound is an event.

   For example, the following code loads an event sound:<<Loc: in next line of code, added `/sound14.mp3` --IMD>>

```
this.loadSound("http://www.foo.com/SoundsToLoad/sound14.mp3", false)
```

10 In the Actions toolbox, click the Objects category, click Movie, Sound, and Methods, and double-click the `start` method to add it to the Script pane.

11 In the Object parameter box, enter the instance name of the sound to start—in this example, `mySound`.

The code should look like this:

```
mySound = new Sound();
mySound.loadSound("http://www.foo.com/SoundsToLoad/sound14.mp3", true);
mySound.start();
```

For more information, see the MovieClip and Sound objects in the online ActionScript Dictionary in the Help menu.

About XML

XML (*Extensible Markup Language*) is becoming the standard for the interchange of structured data in Internet applications. You can integrate data in Flash with servers that use XML technology to build sophisticated applications, such as a chat system or a brokerage system.

In XML, as with HTML, you use tags to *mark up*, or specify, a body of text. In HTML, you use predefined tags to indicate how text should appear in a Web browser (for example, the `<b>` tag indicates that text should be bold). In XML, you define tags that identify the type of a piece of data (for example, `<password>VerySecret</password>`). XML separates the structure of the information from the way it's displayed, so the same XML document can be used and reused in different environments.<<Loc: changed "to be used" to "can be used" in last sentence --IMD>>

Every XML tag is called a *node*, or an element. Each node has a type (1, which indicates an XML element, or 3, which indicates a text node), and elements may also have attributes. A node nested in a node is called a *child node*. This hierarchical tree structure of nodes is called the XML Document Object Model (DOM)—much like the JavaScript DOM, which is the structure of elements in a Web browser.

In the following example, `<PORTFOLIO>` is the parent node; it has no attributes and contains the child node `<HOLDING>`, which has the attributes `SYMBOL`, `QTY`, `PRICE`, and `VALUE`:

```
<PORTFOLIO>
  <HOLDING SYMBOL="RICH"
    QTY="75"
    PRICE="245.50"
    VALUE="18412.50" />
</PORTFOLIO>
```

Using the XML object

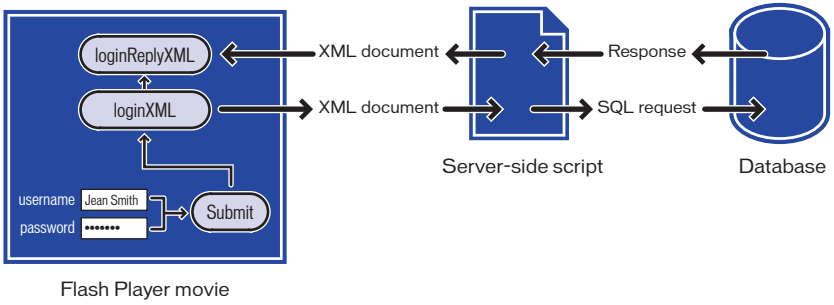
The methods of the ActionScript XML object (for example, `appendChild`, `removeNode`, and `insertBefore`) let you structure XML data in Flash to send to a server and manipulate and interpret downloaded XML data.

The following XML object methods send and load XML data to a server by using the HTTP `POST` method:

- The `load` method downloads XML from a URL and places it in an ActionScript XML object.
- The `send` method passes an XML object to a URL. Any returned information is sent to another browser window.
- The `sendAndLoad` method sends an XML object to a URL. Any returned information is placed in an ActionScript XML object.

For example, you could create a brokerage system that stores all its information (user names, passwords, session IDs, portfolio holdings, and transaction information) in a database.

The server-side script that passes information between Flash and the database reads and writes the data in XML format. You can use ActionScript to convert information collected in the Flash movie (for example, a user name and password) to an XML object and then send the data to the server-side script as an XML document. You can also use ActionScript to load the XML document that the server returns into an XML object to be used in the movie.



The flow and conversion of data between a Flash movie, a server-side script, and a database.

The password validation for the brokerage system requires two scripts: a function defined on frame 1, and a script that creates and sends the XML objects attached to the Submit button in the form.

When users enter their information into text fields in the Flash movie with the variables `username` and `password`, the variables must be converted to XML before being passed to the server. The first section of the script loads the variables into a newly created XML object called `loginXML`. When a user clicks the Submit button, the `loginXML` object is converted to a string of XML and sent to the server.

The following script is attached to the Submit button. To understand this script, read the commented lines (indicated by the characters `//`):

```
on (release) {
  // A. Construct an XML document with a LOGIN element
  loginXML = new XML();
  loginElement = loginXML.createElement("LOGIN");
  loginElement.attributes.username = username;
  loginElement.attributes.password = password;
  loginXML.appendChild(loginElement);

  // B. Construct an XML object to hold the server's reply
  loginReplyXML = new XML();
  loginReplyXML.onLoad = onLoginReply;

  // C. Send the LOGIN element to the server,
  //    place the reply in loginReplyXML
  loginXML.sendAndLoad("https://www.imexstocks.com/main.cgi",
               loginReplyXML);
}
```

The first section of the script generates the following XML when the user clicks the Submit button:

```
<LOGIN USERNAME="JeanSmith" PASSWORD="VerySecret" />
```

The server receives the XML, generates an XML response, and sends it back to the Flash movie. If the password is accepted, the server responds with the following:

```
<LOGINREPLY STATUS="OK" SESSION="rnr6f7vkj2oe14m7jkkycilb" />
```

This XML includes a `SESSION` attribute that contains a unique, randomly generated session ID, which will be used in all communications between the client and server for the rest of the session. If the password is rejected, the server responds with the following message:

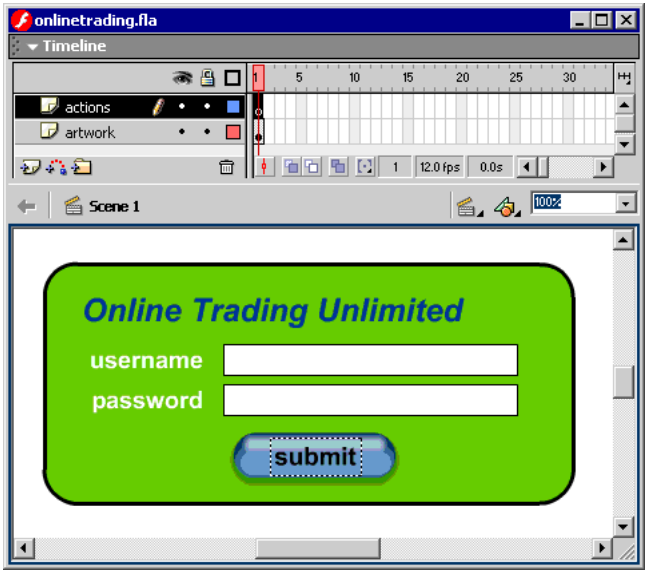
```
<LOGINREPLY STATUS="FAILED" />
```

The `LOGINREPLY` XML node must load into a blank XML object in the Flash movie. The following statement creates the XML object `loginreplyXML` to receive the XML node:

```
// B. Construct an XML object to hold the server's reply
loginReplyXML = new XML();
loginReplyXML.onLoad = onLoginReply;
```

The second statement assigns the `onLoginReply` function to the `loginReplyXML.onLoad` handler.

The `LOGINREPLY` XML element arrives asynchronously, much like the data from a `loadVariables` action, and loads into the `loginReplyXML` object. When the data arrives, the `onLoad` method of the `loginReplyXML` object is called. You must define the `onLoginReply` function and assign it to the `loginReplyXML.onLoad` handler so that it can process the `LOGINREPLY` element. You must also assign the `onLoginReply` function to the frame that contains the Submit button.



The `onLoginReply` function is defined in the first frame of the movie. (To understand this script, read the commented lines.)

```
function onLoginReply() {
  // Get the first XML element
  var e = this.firstChild;
  // If the first XML element is a LOGINREPLY element with
  // status OK, go to the portfolio screen. Otherwise,
  // go to the login failure screen and let the user try again.
  if (e.nodeName == "LOGINREPLY" && e.attributes.status == "OK") {
// Save the session ID for future communications with server
  sessionID = e.attributes.session;
// Go to the portfolio viewing screen
    gotoAndStop("portfolioView");
  } else {
    // Login failed!  Go to the login failure screen.
    gotoAndStop("loginFailed");
  }
}
```

The first line of this function, `var e = this.firstChild`, uses the keyword `this` to refer to the XML object `loginReplyXML` that has just been loaded with XML from the server. You can use `this` because `onLoginReply` has been invoked as `loginReplyXML.onLoad`, so even though `onLoginReply` appears to be a normal function, it actually behaves as a method of `loginReplyXML`.

To send the user name and password as XML to the server and to load an XML response back into the Flash movie, you can use the `sendAndLoad` method, as shown here:

```
// C. Send the LOGIN element to the server,
//    place the reply in loginReplyXML
  loginXML.sendAndLoad("https://www.imexstocks.com/main.cgi", loginReplyXML);
```

Note: This design is only an example, and Macromedia can make no claims about the level of security it provides. If you are implementing a secure password-protected system, make sure you have a good understanding of network security.

For more information on using XML to build Web applications, see "Integrating XML and Flash in a Web Application at www.macromedia.com/support/flash/interactivity/xml/. For more information on the XML object, see its entry in the online ActionScript Dictionary in the Help menu.

Using the XMLSocket object

ActionScript provides a built-in XMLSocket object that allows you to open a continuous connection with a server. A socket connection allows the server to publish (or "push") information to the client as soon as that information is available. Without a continuous connection, the server must wait for an HTTP request. This open connection removes latency issues and is commonly used for real-time applications such as chats. The data is sent over the socket connection as one string and should be in XML format. You can use the XML object to structure the data.

To create a socket connection, you must create a server-side application to wait for the socket connection request and send a response to the Flash movie. This type of server-side application can be written in a programming language such as Java.

You can use the ActionScript XMLSocket object's `connect` and `send` methods to transfer XML to and from a server over a socket connection. The `connect` method establishes a socket connection with a Web server port. The `send` method passes an XML object to the server specified in the socket connection.

When you invoke the XMLSocket object's `connect` method, the Flash Player opens a TCP/IP connection to the server and keeps that connection open until one of the following happens:

- The `close` method of the XMLSocket object is called.
- No more references to the XMLSocket object exist.
- The Flash Player quits.
- The connection is broken (for example, the modem disconnects).

The following example creates an XML socket connection and sends data from the XML object `myXML`. To understand the script, read the commented lines (indicated by the characters `//`):

```
// Create a new XMLSocket object
sock = new XMLSocket();
// Call its connect method to establish a connection with port 1024
// of the server at the URL
sock.connect("http://www.myserver.com", 1024);
// Define a function to assign to the sock object that handles
// the server's response. If the connection succeeds, send the
// myXML object. If it fails, provide an error message in a text
// field.
function onSockConnect(success){
  if (success){
    sock.send(myXML);
  } else {
    msg="There has been an error connecting to "+serverName;
  }
}
// Assign the onSockConnect function to the onConnect property
sock.onConnect = onSockConnect;
```

For more information, see the entry for the XMLSocket object in the online ActionScript Dictionary in the Help menu.

Sending messages to and from the Flash Player

To send messages from a Flash movie to its host environment (for example, a Web browser, a Macromedia Director movie, or the stand-alone Flash Player), you can use the `fscommand` action. This action lets you extend your movie by using the capabilities of the host. For example, you could pass an `fscommand` action to a JavaScript function in an HTML page that opens a new browser window with specific properties.

To control a movie in the Flash Player from Web browser scripting languages such as JavaScript, VBScript, and Microsoft JScript, you can use Flash Player methods—functions that send messages from a host environment to the Flash movie. For example, you could have a link in an HTML page that sends your Flash movie to a specific frame.

Using fscommand

Use the `fscommand` action to send a message to whichever program is hosting the Flash Player. The `fscommand` action has two parameters: *`command`* and *`arguments`*. To send a message to the stand-alone version of the Flash Player, you must use predefined commands and arguments (parameters). For example, the following action sets the stand-alone player to scale the movie to the full monitor screen size when the button is released:

```
on(release){
  fscommand("fullscreen", "true");
}
```

The following table shows the values you can specify for the *command* and *arguments* parameters of the fscommand action to control a movie playing in the stand-alone player (including projectors):

Command	Arguments	Purpose
quit	None	Closes the projector.
fullscreen	true or false	Specifying true sets the Flash Player to full-screen mode. Specifying false returns the player to normal menu view.
allowscale	true or false	Specifying false sets the player so that the movie is always drawn at its original size and never scaled. Specifying true forces the movie to scale to 100% of the player.
showmenu	true or false	Specifying true enables the full set of context menu items. Specifying false dims all the context menu items except About Flash Player.
exec	Path to application	Executes an application from within the projector.

To use fscommand to send a message to a scripting language such as JavaScript in a Web browser, you can pass any two parameters in the *command* and *arguments* parameters. These parameters can be strings or expressions and will be used in a JavaScript function that "catches," or handles, the fscommand action.

An fscommand action invokes the JavaScript function *moviename*_DoFSCommand in the HTML page that embeds the Flash movie, where *moviename* is the name of the Flash Player as assigned by the NAME attribute of the EMBED tag or the ID attribute of the OBJECT tag. If the Flash Player is assigned the name myMovie, the JavaScript function invoked is myMovie_DoFSCommand.

To use the fscommand action to open a message box from a Flash movie in the HTML page through JavaScript:

1 In the HTML page that embeds the Flash movie, add the following JavaScript code:

```
function theMovie_DoFSCommand(command, args) {
  if (command == "messagebox") {
    alert(args);
  }
}
```

If you publish your movie using the Flash with FSCommand template in the HTML Publish Settings dialog box, this code is inserted automatically. The movie's NAME and ID attributes will be the file name. For example, for the file myMovie.fla, the attributes would be set to myMovie. (For more information about publishing, see Chapter 20, "Publishing," on page 365.)

Alternatively, for Internet Explorer applications, you can attach an event handler directly in the <SCRIPT> tag, as shown in this example:

```
<Script Language = "JavaScript" event="FSCommand (command, args)" for=
  "theMovie">
...
</Script>
```

2 In the Flash document, add the fscommand action to a button, as shown in this example:

```
fscommand("messagebox", "This is a message box invoked from within Flash.")
```

You can also use expressions for the fscommand action and parameters, as in this example:<<Loc: changed & to + twice in next code line --IMD>>

```
fscommand("messagebox", "Hello, " + name + ", welcome to our Web site!")
```

3 Choose File > Publish Preview > HTML to test the document.

The `fscommand` action can send messages to Macromedia Director that are interpreted by Lingo as strings, events, or executable Lingo code. If the message is a string or an event, you must write the Lingo code to receive it from the `fscommand` action and carry out an action in Director. For more information, see the Director Support Center at www.macromedia.com/support/director.

In Visual Basic, Visual C++, and other programs that can host ActiveX controls, `fscommand` sends a VB event with two strings that can be handled in the environment's programming language. For more information, use the keywords *Flash method* to search the Flash Support Center at www.macromedia.com/support/flash.

About Flash Player methods

You can use Flash Player methods to control a movie in the Flash Player from Web browser scripting languages such as JavaScript and VBScript. As with other methods, you can use Flash Player methods to send calls to Flash movies from a scripting environment other than ActionScript. Each method has a name, and most methods take parameters. A parameter specifies a value that the method operates upon. The calculation performed by some methods returns a value that can be used by the scripting environment.

There are two different technologies that enable communication between the browser and the Flash Player: LiveConnect (Netscape Navigator 3.0 or later on Windows 95/98/2000/NT or Power Macintosh) and ActiveX (Microsoft Internet Explorer 3.0 and later on Windows 95/98/2000/NT). Additionally, in Microsoft Internet Explorer 5.5 and later, the Flash Player can be hosted as a binary behavior or custom element tag. Although the techniques for scripting are similar for all browsers and languages, there are additional properties and events available for use with ActiveX controls.

For more information, including a complete list of the Flash Player scripting methods, use the keywords *Flash method* to search the Flash Support Center at www.macromedia.com/support/flash.

CHAPTER 17
Creating Printable Movies

Once you have completed the contents of your Macromedia Flash MX movie, you can specify that certain frames be printable with the Flash Player. You can use the Flash Player printing feature to allow users to print catalogs, coupons, information sheets, receipts, invoices, or other documents in your Flash movies.

With the Flash Player, you can print Flash content as vector graphics at the high resolutions available from printers and other output devices. Printing content as vector graphics scales Flash artwork so that it prints clearly at any size without the pixelated effects that can occur when printing low-resolution bitmap images.

Printing movies from the Flash Player instead of from the browser gives Flash authors several advantages. You can do the following:

- Specify which frames in a Flash movie can be printed. This lets you design pages with layouts appropriate for printing and protect material from unauthorized printing.
- Determine the print area of frames. For example, if the printable material takes up only a portion of the frame, you can designate only that area of the frame as printable.
- Specify whether frames are printed as vectors (to take advantage of higher resolutions) or as bitmaps (to preserve transparency and color effects).
- Assign Print actions to print frames from movie clips, even if the movie clips are not visible. This lets you provide printable material without using valuable browser space.

Printing from the Flash Player

Users can print movies directly from the Flash Player in a browser in two ways: either using the Flash Player context menu and its Print command, or using the Print action to create a button or other trigger in the movie that activates printing. A Print action gives you more control over how a Flash movie can be printed and eliminates the need to use the Flash Player context menu.

The Print action can print frames in any Timeline, including the main Timeline, or the Timeline of any movie clip or loaded movie level. The Print action also lets you specify a print area and lets you print color effects, including transparency.

The Flash Player context menu is more limited in its printing capabilities: it only prints frames in the main Timeline and does not let you print transparency or color effects.

Note: Flash Player versions earlier than 4.0.25 (Windows) or 4.0.20 (Macintosh) do not support direct printing of frames.

Preparing movies for printing

To set up printing from the Flash Player, you can specify which frames to print and set their print area. To best control what users can print, keep the following in mind as you set up movies and movie clips for printing:

- Adjust the page layout in any frames that you'll designate as printable to match the desired printed output. Using the Flash Player, you can print all shapes, symbols, bitmaps, text blocks, and text fields. Levels in a Flash movie are not composited on print output.
- The Flash Player printer driver uses the HTML settings for dimension, scale, and alignment in the Publish Settings dialog box. Use these settings to control the print layout.
- The selected frames print as they appear in the movie clip symbol. You can let users print a movie clip that is not visible in a browser by setting the movie clip's `_visible` property to `false` using the Actions panel. Changing the property of a movie clip with the Set Property action, tweening, or any transformation tool does not affect how a movie clip prints.
- For a movie clip to be printable, it must be on the Stage or work area and it must be given an instance name.
- All elements must be fully loaded to print. You can use the `_framesloaded` property to check whether the printable content is loaded. For more information, see this term in the online ActionScript Dictionary in the Help menu.

Supported printers

With the Flash Player, you can print to both PostScript and non-PostScript printers. For a list of supported Flash Player printing platforms, see "Flash Web Printing for eBusiness"on the Macromedia Web site (www.macromedia.com/software/flash/open/webprinting/faq.html).

Designating printable frames

All frames in the specified Timeline print by default. You may want to limit the number of frames that can print—for example, if you have a lengthy animation of dozens of frames. You can designate specific frames in a movie as printable in order to print only those frames; unspecified frames won't print.

To specify frames as printable, you label the frames.

To designate printable frames:

1. Open or make active the movie that you want to publish.
2. Select the desired frame in the Timeline that you want to make printable.
3. Choose Window > Properties to view the Property inspector.
4. In the Property inspector, for Label enter **#p** to specify the frame as printable.
5. Repeat steps 3 and 4 for each frame you want to designate as printable.

Specifying a print area

By default, the movie's Stage determines the print area. Any object that extends off the Stage is clipped and does not print. Loaded movies use their own Stage size for the print area, not the main movie's Stage size.

As an alternative to using a movie's Stage size, you can set three different print areas:

- For either the Flash Player context menu or the Print action, you can designate the movie's bounding box as the print area for all frames by selecting an object in one frame as the bounding box. This option is useful, for example, if you want to print a full-page data sheet from a Web banner.
- With the Print action, you can use the composite bounding box of all printable frames in a Timeline as the print area—for example, to print multiple frames that share a registration point. To use the composite bounding box, select the Max argument in the Print action parameters. See "Adding a Print action" on page 336.
- With the Print action, you can change the print area for each frame, scaling objects to fit the print area—for example, to have objects of different sizes in each frame fill the printed page. To change the bounding box per frame, use the Frame parameter in the Print action parameters. See "Adding a Print action" on page 336.

To specify a print area:

1. Open the Flash document (FLA) containing the frames you will set to print.
2. Choose a frame that you have not specified to print with a #p frame label.

   To organize your work, you can select the next frame after one labeled #p.
3. Create a shape on the Stage the size of the desired print area.

   You can also choose a frame with any object of the appropriate print area size to use that frame's bounding box.
4. Select the frame in the Timeline that contains the shape you'll use for the bounding box.
5. If the Property inspector is not displayed, choose Window > Properties.
6. In the Property inspector, enter **#b** for Label to specify the selected shape as the bounding box for the print area.

   You can enter only one #b label per Timeline. This option is the same as selecting the Movie bounding box option with the Print action.

Changing the printed background color

With the Flash Player, you can print the background color set in the Document Properties dialog box. You can change the background color for only the frames to be printed by placing a colored object on the lowest layer of the Timeline being printed.

To change the printed background color:

1. Place a filled shape that covers the Stage on the lowest layer of the Timeline that will print.
2. Select the shape and choose Modify > Document. Select a color for the printing background.

   This changes the entire movie's background color, including that of movie clips and loaded movies.

3 Choose from the following options:

- To print that color as the movie's background, make sure that the frame in which you placed the shape is designated to print. For instructions, see "Designating printable frames" on page 334.
- To maintain a different background color for nonprinting frames, repeat steps 2 and 3.Then place the shape on the lowest layer of the Timeline, in all frames that are not designated to print. For instructions, see the following section.

Disabling printing

If you don't want any of the frames in the main Timeline to be printable, you label a frame as !#p to make the entire movie nonprintable. Labeling a frame as !#p dims the Print command in the Flash Player context menu. You can also remove the Flash Player context menu.

If you disable printing from the Flash Player, the user can still print frames using the browser Print command. Because this command is a browser feature, you cannot control or disable it using Flash.

To disable printing in the Flash Player context menu by dimming the Print command:

1 Open or make active the Flash document (FLA) that you want to publish.

2 Select the first keyframe in the main Timeline.

3 Choose Window > Properties to view the Property inspector.

4 In the Property inspector, for Label enter **!#p** to specify the frame as nonprinting.

You need to specify only one !#p label to dim the Print command in the context menu.

Note: Alternatively, you can select a blank frame and label it #p to prevent printing from the Flash Player context menu.

To disable printing by removing the Flash Player context menu:

1 Open or make active the Flash document (FLA) that you want to publish.

2 Choose File > Publish Settings.

3 Select the HTML tab and deselect Display Menu.

4 Click OK.

For more information on publishing options, see "Publishing Flash documents" on page 367.

Adding a Print action

You can add a Print action to a button or other element in your movie to let users print the movie. You assign the Print action to a button, frame, or movie clip. If you assign a Print action to a frame, the action executes when the playhead reaches the designated frame.

The Print action lets you print frames in other movie clips in addition to the main Timeline. Each Print action sets only one Timeline for printing, but the action lets you specify any number of frames within the Timeline to print. If you attach more than one Print action to a single button or frame, the Print dialog box appears for each action executed.

To assign a Print action to a button, frame, or movie clip:

1. Open the Flash document (FLA) containing the frames you will set to print.
2. Select the desired keyframe in the Timeline that you want to be able to print and make sure that it is labeled #p. See the instructions in "Designating printable frames" on page 334.

   If you don't specify which frames to print, all the frames in the movie print by default.
3. Select the frame, button instance, or movie clip instance to which you will assign the Print action.

   Each Print action sets only one Timeline to be printable.
4. Choose Window > Actions to display the Actions panel.
5. In the Actions toolbox, click the Actions category to display the actions, and double-click to select the Print action. Flash inserts the Print action in the Actions list.
6. For Print, choose to print the frame as vectors or as a bitmap:

- As Vectors prints the frame at a higher quality, but without transparency.

  Objects containing transparency or color effects cannot be printed as vector data. (The printer cannot interpret the alpha channel that defines the effect as vector data.)
- As Bitmap prints transparency in an alpha channel or color effect.

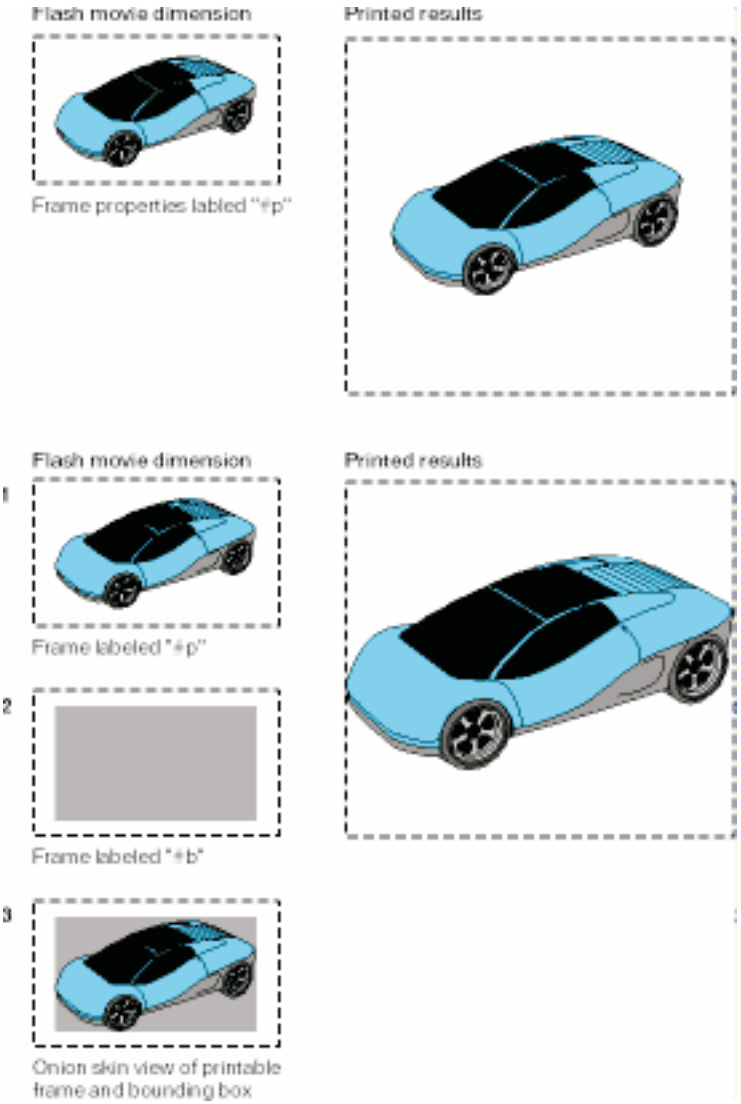
  This option prints at the highest available resolution of the printer.

7. To specify which movie Timeline to print, choose a Location option:

- For Level, specify the level number of the main Timeline or loaded movie. To use an expression to evaluate to the level, select Expression and enter an expression. For more information on levels, see "Loading and unloading additional movies" on page 255.
- For Target, enter the path to the target movie, or click the Target Path button in the lower right corner and use the Insert Target Path dialog box to locate and select the target movie. To use an expression to evaluate to the target, select Expression and enter an expression.

8. To set the printing boundaries, select a Bounding Box option:

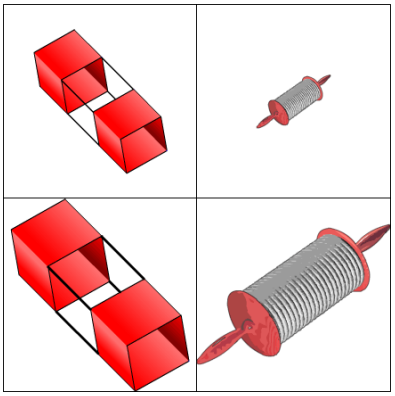
- Movie uses the bounding box of an object in the frame labeled #b as the print area for all frames as set in "Specifying a print area" on page 335. For example, choose this option to print a full-page data sheet from a Web banner.



Top: Frame labeled #p (left) prints the Stage area (right).
Bottom: Frame labeled #p (1) and frame labeled #b (2), with onion skin view (3), print the object's bounding box (right).

- Max uses the composite bounding box of all printable frames in a Timeline as the print area.

- Frame uses the bounding box of the objects in each printable frame of a Timeline as the print area, changing the print area for each frame and the scaling objects to fit the print area. For example, use Frame if you have different-sized objects in each frame and you want each object to fill the printed page.



Frame option sets the bounding box of each frame as the print area (top), scaling artwork to fit (bottom).

Note: Choosing the Max or Frame bounding box options in the Print action overrides any frames labeled #b for the movie's bounding box.

Printing from the Flash Player context menu

You can use the Print command in the Flash Player context menu to print frames from any Flash movie.

The context menu's Print command cannot print transparency or color effects and cannot print frames from other movie clips; for these printing capabilities, use the Print action instead. See "Adding a Print action" on page 336.

To print movie frames using the Flash Player context menu Print command:

1. Open the movie whose frames you will print.

   The command prints the frames labeled #b using the Stage for the print area or the specified bounding box. See "Designating printable frames" on page 334 and "Specifying a print area" on page 335.

   If you haven't designated specific frames to print, all frames in the movie's main Timeline print.
2. Choose File > Publish Preview > Default or press F12 to view your Flash movie in a browser.
3. Right-click (Windows) or Control-click (Macintosh) in the Flash movie in the browser window to display the Flash Player context menu.
4. Choose Print from the Flash Player context menu to display the Print dialog box.

5 In Windows, choose the print range to select which frames to print:

- Choose All to print all frames in the movie if no frames are labeled.
- Choose Pages and enter a range to print the labeled frames in that range.
- Choose Selection to print the current frame.

6 On the Macintosh, in the Print dialog box, select the pages to print:

- Choose All to print the current frame if no frames are labeled or to print all labeled frames.
- Choose From and enter a range to print the labeled frames in that range.

7 Select other print options, according to your printer's properties.

8 Click OK (Windows) or Print (Macintosh).

About publishing a movie with printable frames

You can publish a Flash movie with printable frames to the Web using the Publish command to generate the necessary Flash HTML templates. For more information, see "Publishing Flash documents" on page 367.

Users must have the Flash Player 4.0.25 (Windows) or 4.0.20 (Macintosh) or later to take advantage of any print functionality you have added and to be able to print the designated frames in Flash. You can set up a detection scheme to check for the proper Flash Player version. See "Screening traffic to your Web site" on page 394.

CHAPTER 18
Creating Accessible Content

A growing requirement for Web content is that it should be accessible—usable for people with a variety of disabilities. Visual content in Macromedia Flash movies can be made accessible to the visually impaired with the use of screen reader software, which provides a spoken audio description of the contents of the screen.

Accessible Flash movie content is supported by the Flash Player 6. Users must have a Windows operating system that supports Flash accessibility, and the appropriate screen reader software (including the Flash Player 6) in order to take advantage of accessible Flash content. For more information, see the Flash Accessibility Web page at www.macromedia.com/software/Flash/productinfo/accessibility/.

Screen reader technology is designed primarily to convey information about static user interfaces. Making your movie accessible will be most successful if you keep dynamic content to a minimum, and emphasize text and user interface features. You can select which objects in a movie to expose to a screen reader, and can omit animations or visually oriented movie clips to increase accessibility.

About the Macromedia Flash Accessibility Web page

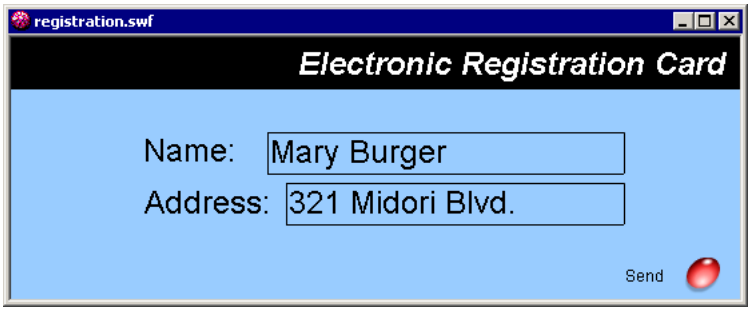
The Macromedia Flash Accessibility Web page is located at www.macromedia.com/software/Flash/productinfo/accessibility/.

Consult this page for the latest information on creating and viewing accessible Flash content, including supported platforms, screen reader compatibility, accessible examples, and more.

About screen reader technology

The Flash Player communicates with screen reader software to provide information on the visual content of a Flash movie. The screen reader in turn generates a spoken audio description of the contents of the screen.

Different screen reader applications use different methods for translating information into speech, so you can't know exactly how your movie will be presented to each user. For the simple Flash movie below, the accompanying text is one possible audio version of the movie that a screen reader might present:



Possible screen reader version:
"Electronic registration card. Textfield name. Textfield address. Button send."

From this version, a user can understand what the movie contains. The screen reader software will probably provide a keyboard option for pressing the button (for example, the user might use the Tab key to advance through the objects in the movie, as the screen reader describes them, and use the Enter key to press the button).

Note: In this example, the screen reader software enables keyboard access to the button. No special keyboard scripting is required in the Flash movie to support this kind of interaction.

Screen readers are complicated applications. This example shows only a simple introduction to screen reader capabilities. But these basics are all you need to know about screen readers in order to create Flash movies that will work well with screen readers.

About accessible objects in Flash movies

By default, the following objects are defined as accessible in all Flash movies and are included in the information that the Flash Player provides to screen reader software:

- Text
- Input text fields
- Buttons
- Movie clips
- Entire movies

Screen readers are primarily intended to help users navigate through the user interfaces of traditional applications–menus, toolbars, dialog boxes, and so on. Thus, text, buttons, and input text fields are types of objects that screen readers understand well, and that translate well to a spoken representation.

Since graphics can't easily be translated into spoken words, the Flash Player does not include individual graphic objects in the information provided, or exposed, to the screen reader. The Flash Player does include certain movie clips, especially those with descriptions that you provide. In addition, Flash Player includes the Flash movie itself in the information provided to the screen reader (even if the movie contains no other accessible objects).

Note: For accessibility purposes, button movie clips are considered buttons, not movie clips, by the Flash Player. See "Using button events with movie clips to trigger scripts" on page 262.

Flash MX allows you to provide some customization of your accessible objects, which can greatly improve the accessibility of your Flash movies for screen reader users. For each of the five kinds of accessible objects, you can set descriptive properties that will be provided to screen readers. The most important of these is the Name property, which screen readers will almost always say aloud when speaking an object. You can also control how the Flash Player decides which objects to expose to screen readers—for example, you can specify that certain accessible objects should not be exposed to screen readers at all.

Objects in Flash movies must have instance names in order for you to apply accessibility options to them. Flash provides default instance names when the instances are created. You can also apply custom names to instances. See "Creating instances" on page 154 or "Setting dynamic and input text options" on page 142.

Static text blocks do not have instance names. The contents of static text blocks is exposed to screen readers by default. To apply any other accessibility options to static text, you must convert the text block to a dynamic text field.

Supported configurations

The Flash Player uses a technology called Microsoft Active Accessibility (MSAA) to communicate with screen readers. MSAA provides a highly descriptive and standardized way for applications and screen readers to communicate. MSAA is available on Windows operating systems only. See the Macromedia Flash Accessibility Web page for more information about Windows operating systems that support MSAA.

The Windows ActiveX (Internet Explorer plug-in) version of Flash Player 6 supports MSAA, but the Windows Netscape and Windows stand-alone players do not.

MSAA is currently not supported in the opaque windowless and transparent windowless modes. (These modes are options in the HTML Publish Settings panel, available for use with the Windows version of Internet Explorer 4.0 or later, with the Flash ActiveX control.) For information about these modes, see "Setting publish settings for HTML documents accompanying Flash movies" on page 371. If you need your Flash movies to be accessible to screen readers, avoid using these modes.

Specifying basic accessibility

The most important step you can take to make your Flash movies accessible to screen readers is to make sure that every accessible object in your movie has a name. Screen readers identify objects by reading their names aloud. When accessible objects don't have names, screen readers say something generic, such as "Button." This can be confusing for screen reader users.

The Flash Player automatically provides names for static and dynamic text objects. The names of these objects are simply the contents of the text. Thus, you don't need to provide names for text objects—they name themselves automatically. The most important task in accessible Flash authoring is to provide names for buttons and input text fields.

About button and text field labeling

You can assign labels to buttons and input text fields so that they are identified appropriately by the screen reader. Often, an appropriate name for a button or input text field already appears in your movie, as a text label that you have placed on top of, inside, or near a button or text field. When the Flash Player discovers an arrangement like this, it will assume that the text object is a label for the button or text field.

You can turn off automatic labeling if it is inconvenient. See "Specifying advanced accessibility options" on page 344.

If you provide a name for an object, any text that would normally serve as a label is instead exposed to screen readers as a text object. This can cause confusion for users, because a screen reader reads both the text object and the name you have provided. In such cases you can hide the text object from the screen reader.

Choosing names for buttons, text fields, and entire movies

When you have a button or input text field that doesn't have a label, or when your label is in a location that the Flash Player labeling rules can't detect, it's a good idea to specify a name for the button or text field. You can also specify a name if you have text in a label position near a button or text field, but you don't want that text to be used as that object's name.

Sometimes it's useful to provide a name for your entire Flash movie. If you do, a screen reader will probably read it aloud. However, often it is sufficient just to name all of the accessible objects inside the movie.

Note: An object's accessibility name is unrelated to the ActionScript instance name or ActionScript variable name associated with the object. For information on instance names and variable names, see Chapter 12, "Understanding the ActionScript Language," on page 203.

To specify a name for a button, text field, or entire movie:

1 Do one of the following:

- To provide a name for a button or text field, select the object on the Stage.
- To provide a name for an entire movie, deselect all objects on the Stage.

2 Do one of the following:

- Choose Window > Property inspector if the inspector is not visible. In the Property inspector, click the Accessibility button.
- Choose Window > Accessibility.

3 In the Accessibility panel, make sure that Make Object Accessible (for buttons or text fields) or Make Movie Accessible (for entire movies) is selected (the default setting).

4 Enter a name for the button, text field, or movie in the Name text box.

Specifying advanced accessibility options

Flash provides several accessibility authoring features that go beyond simply providing names for objects. You can provide a description for text or text fields, buttons, or movie clips, and keyboard shortcuts for input text fields or buttons. You can also turn off automatic labeling behavior for your movie.

You can choose to hide a selected object from screen readers. For example, you may choose to hide animated movie clips if you think the verbal description does not enhance the accessible version of the movie. You may also decide to hide accessible objects that are contained inside a movie clip or movie, and expose only the movie clip or movie itself to screen readers.

For keyboard shortcuts, use the following conventions:

- Spell out key names, such as Ctrl or Alt
- Use capital letters for alphabetic characters
- Use a plus sign (+) between key names, with no spaces—for example, Ctrl+A

Note: If you provide a keyboard shortcut for an input text field or button, you must also use the ActionScript Key object to detect the key the user presses during movie playback. See"Capturing keypresses" on page 275.Keyboard shortcut functionality also depends on the screen reader software used.

To define the accessibility for a selected object in a movie:

1. Select the object on the Stage.
2. Do one of the following:

- Choose Window > Property inspector if the inspector is not visible. In the Property inspector, click the Accessibility button.
- Choose Window > Accessibility.

3. In the Accessibility panel, do one of the following:

- Select Make Object Accessible (the default setting) to expose the object to screen readers, and to enable other options in the panel.
- Deselect the option to hide the object from screen readers.

4. If you selected Make Object Accessible in step 3, enter information for the selected object as needed:

- For dynamic text, enter a name for the text object. Enter a description of the text in the Description field. (To provide a description for static text, you must convert it to dynamic text.)
- For input text fields or buttons, enter a name for the object. Enter a description of the object in the Description text box. Enter a keyboard shortcut in the Shortcut text box.
- For movie clips, enter a name for the object. Enter a description in the Description text box. Select Make Child Objects Accessible to expose the objects inside the movie clip to screen readers. Deselect this option to hide any accessible objects contained in the movie clip from screen readers.

5. Click OK.

To turn off an automatic label for an individual object:

1. On the Stage, select the button or input text field for which you want to control labeling.
2. Do one of the following:

- Choose Window > Property inspector if the inspector is not visible. In the Property inspector, click the Accessibility button.
- Choose Window > Accessibility.

3. In the Accessibility panel, select Make Object Accessible (the default setting).

4 Enter a name in the Name text box.

The name will be read as the label for the button or text field. The text string that was the automatic label is read as a regular text object, unless you turn off accessibility for the text string.

5 To turn off accessibility for the automatic label (and hide it from screen readers), select the text object on the Stage.

6 If the text object is static text, convert it to dynamic text: in the Property inspector, choose Dynamic Text from the Text type pop-up menu.

7 In the Accessibility panel, deselect Make Object Accessible.

To define accessibility for an entire movie:

1 When the Flash document is complete and ready to be published or exported, deselect all elements in the movie.

2 Do one of the following:

- Choose Window > Property inspector if the inspector is not visible. In the Property inspector, click the Accessibility button.
- Choose Window > Accessibility.

3 In the Accessibility dialog box, do one of the following:

- Select Make Movie Accessible (the default setting) to expose the movie to screen readers.
- Deselect the option to hide the movie from screen readers.

4 Select Make Children Accessible to expose the accessible objects contained in the movie to screen readers. Deselect this option to omit any accessible objects contained in the movie clip from screen readers.

5 If you selected Make Movie Accessible in step 3, enter information for the movie as needed:

- Enter a title for the movie in the Title text box.
- Enter a description of the movie in the Description text box.

6 Select Auto Label (the default setting) to use text objects as automatic labels for accessible buttons or input text fields contained in the movie. Deselect this option to turn off automatic labeling and expose text objects to screen readers as text objects. See “About animation and accessibility” on page 346.

7 Click OK.

About animation and accessibility

In some situations you may want to change the property of an accessible object during the course of movie playback. For example, you may want to indicate changes that take place on a keyframe in an animation.

To update properties for an accessible object, display the frame in which you want to change the properties, and change the properties for that object as needed.

Different screen readers treat new objects on frames differently. Some screen readers may read only the new object. Some screen readers may re-read the entire movie.

Custom tabs for accessible objects

You can control tab order in Flash movies using ActionScript properties. If you create a custom tab order for a movie, the accessible objects follow the specified tab order. You should include all accessible objects in the tab order, even when those objects do not represent tab stops. For example, dynamic text and movie clips should be included in the tab order so that screen readers know when to read these objects.

Tab order can be assigned to dynamic text objects, buttons, movie clips, and input text fields. To assign tab order to a static text object, you must first convert it to a dynamic text object. You can use the ActionScript `tabIndex`, `tabChildren`, or `tabEnabled` methods to assign custom tab order. For more information about these methods, see the online ActionScript Dictionary in the Help menu.

If you provide a custom tab order for a given frame in your movie, and you do not specify a tab position for one or more of the accessible objects in that frame, the Flash Player will disregard your custom tab order when users are using a screen reader.

About the Accessibility.isActive method

If you want to make your movie behave in customized ways when a screen reader is present, you can use the ActionScript method `Accessibility.isActive`, which returns a value of `true` if a screen reader is running during movie runtime, and `false` otherwise. For detailed information on the `Accessibility.isActive` method, see its entry in the online ActionScript Dictionary in the Help menu.

About components

Flash components that represent user interface elements have special requirements in order to work with screen readers. The components must contain ActionScript that defines their accessible behavior. For information on which built-in components work with screen readers, see the Macromedia Flash Accessibility homepage. For general information on components, see Chapter 15, “Using Components,” on page 289.

Suggestions for creating effective accessibility

To give yourself the greatest chance of creating accessible content, you need to be aware of certain rules of thumb, and some things to avoid. These considerations can require some design compromises. Flash is primarily a visual medium, and sometimes you may need to sacrifice some of the complexity of your visual presentation in order to accommodate users of accessibility aids.

Here are some suggestions to keep in mind:

- Screen reader users will not perceive the graphics in your Flash content. If you use graphics to convey information, that information will be lost to screen reader users. Keep in mind that graphical text is a common example of this problem—if you’re using a feature like Text Break Apart to animate text, the Flash Player won’t be able to determine the actual text content of your movie any more. Other examples of information-carrying graphics include icons and gestural animations. You can remedy problems like this by providing names or descriptions for certain accessible objects within your movie, or for the movie as a whole. You can also add supplementary text into your movie, or shift your important information content from graphics to text.

- Consider whether a screen reader user is better off hearing about the individual objects in your movie, or simply hearing a description of the movie as a whole. If you think you can convey the meaning or message of your movie with a single phrase of text, turn off the Make Children Accessible option for your movie, and type in a suitable description. This can often simplify and clarify the experience of a screen reader user.
- Try to avoid animating the text, buttons, and input text fields in your movie. If you keep these kinds of objects stable, you reduce the chance of causing a screen reader to emit extra "chatter" that may annoy users. Also, try to avoid making your movies loop.
- Remember that sound is the most important medium for most screen reader users. Consider how the sounds in your movie, if any, will interact with the text spoken aloud by screen readers. If you have a lot of loud sound, it may be difficult for screen reader users to hear what their screen readers are saying. On the other hand, some quieter or well-placed sound can greatly enhance the experience of a visually impaired user. You can also include recorded speech in your movie, augmenting the information that a screen reader will speak.
- If you are creating an interactive movie, try to make sure that users can navigate through your movie effectively using only the keyboard. This can be an especially challenging requirement, because different screen readers may interfere in different ways with the processing of input from the keyboard—meaning that your Flash movie might not receive keystrokes as you intended. Testing with screen readers is the best option.
- Try not to present information in your movie that only lasts a short time. For example, if you have a series of scenes that show different pieces of text in rapid succession (perhaps one scene every three seconds or so), a screen reader may have a hard time keeping up with your changing content, with the result that some of your text may end up being skipped. You can resolve problems like this by adding "Next" buttons that control scene movement, or by including the full string of all of your text as a description for your entire movie.

Be sure also to look at the Flash accessibility page at www.macromedia.com/software/Flash/productinfo/accessibility/, which contains up-to-date information on the Flash Player, screen readers, tools, downloadable assets, and links to articles and sites from the accessibility community.

Testing accessible content

Screen reader technology is not included in the Test Movie Player (inside the Flash authoring tool), so it is not possible to test how accessible content functions in your movie using the test mode.

If you have access to a screen reader application, you can test your movie's accessibility by playing the movie in the screen reader. Several screen reader applications provide a demonstration version of the software as a free download. You can install this demonstration software on a Windows system (Windows 95 or Windows NT 4.0 or later) to test your Flash movie.

CHAPTER 19
Testing a movie

As you work on a Macromedia Flash MX document, test it often to ensure that it plays as smoothly as possible and that it plays as expected. To test movies, you use a special version of the Flash Player whose tools let you access information that helps you optimize animations and troubleshoot ActionScript. If you use good authoring techniques in your ActionScript, your scripts will be easier to troubleshoot when something behaves unexpectedly.

Flash provides several tools for testing ActionScript in your movies:

- The Debugger shows a hierarchical display list of movie clips currently loaded in the Flash Player. Using the Debugger, you can display and modify variable and property values as the movie plays, and you can use breakpoints to stop the movie and step through ActionScript code line by line.
- The Output window displays error messages and lists of variables and objects.
- The `trace` action sends programming notes and values of expressions to the Output window.

Optimizing movies

As your movie file size increases, so does its download time and playback speed. You can take a number of steps to prepare your movie for optimal playback. As part of the publishing process, Flash automatically performs some optimization on movies: for example, it detects duplicate shapes on export and places them in the file only once, and it converts nested groups into single groups.

Before exporting a movie, you can optimize it further by using various strategies to reduce the file size. You can also compress a SWF file as you publish it. (See Chapter 20, “Publishing,” on page 365.) As you make changes, run your movie on a variety of different computers, operating systems, and Internet connections.

To optimize movies in general:

- Use symbols, animated or otherwise, for every element that appears more than once.
- Whenever possible, use tweened animations, which take up less file space than a series of keyframes.
- For animation sequences, use movie clips instead of graphic symbols.
- Limit the area of change in each keyframe; make the action take place in as small an area as possible.
- Avoid animating bitmap elements; use bitmap images as background or static elements.
- For sound, use MP3, the smallest sound format, whenever possible.

To optimize elements and lines:

- Group elements as much as possible.
- Use layers to separate elements that change over the course of the animation from those that do not.
- Use Modify > Curves > Optimize to minimize the number of separate lines that are used to describe shapes.
- Limit the number of special line types such as dashed, dotted, ragged, and so on. Solid lines require less memory. Lines created with the Pencil tool require less memory than brush strokes.

To optimize text and fonts:

- Limit the number of fonts and font styles. Use embedded fonts sparingly, because they increase file size.
- For Embed Fonts options, select only the characters needed instead of including the entire font.

To optimize colors:

- Use the Color menu in the symbol Property inspector to create many instances of a single symbol in different colors.
- Use the Color Mixer (Window > Color Mixer) to match the color palette of the movie to a browser-specific palette.
- Use gradients sparingly. Filling an area with gradient color requires about 50 bytes more than filling it with solid color.
- Use alpha transparency sparingly; it can slow playback.

To optimize ActionScript:

- Select Omit Trace Actions in the Flash tab of Publish Settings to leave `trace` actions out of a movie when it's published.
- Define functions for frequently repeated code. See "Creating functions" on page 233.
- Use local variables when possible. See "About variables" on page 219.

Testing movie download performance

The Flash Player attempts to meet the frame rate you set; the actual frame rate during playback can vary on different computers. If a movie that is downloading reaches a particular frame before the frame's required data has downloaded, the movie pauses until the data arrives.

To view downloading performance graphically, you can use the Bandwidth Profiler, which shows how much data is sent for each frame according to the modem speed you specify. In simulating the downloading speed, Flash uses estimates of typical Internet performance, not the exact modem speed. For example, if you choose to simulate a modem speed of 28.8 Kbps, Flash sets the actual rate to 2.3 Kbps to reflect typical Internet performance. It's helpful to test your movie at each speed you intend to support, and on each computer you intend to support. This allows you to make sure the movie doesn't overburden the slowest connection and computer it is designed for.

You can also generate a report of frames that are slowing playback, and then optimize or eliminate some of the content in those frames. See "Optimizing movies" on page 349.

To change the settings for the SWF file created by Test Movie and Test Scene, use File > Publish Settings. See "Previewing and testing movies" on page 39.

To test downloading performance:

1 Do one of the following:

- Choose Control > Test Scene or Control > Test Movie.

  If you test a scene or movie, Flash publishes the current selection as a SWF file using the settings in the Publish Settings dialog box. (See "Publishing Flash documents" on page 367.) The SWF file opens in a new window and begins playing immediately.

- Choose File > Open, and select a SWF file.

2 Choose Debug and select a downloading speed to determine the streaming rate that Flash simulates: 14.4 Kbps, 28.8 Kbps, or 56 Kbps. To enter your own settings, choose Customize.

3 When viewing the SWF file, choose View > Bandwidth Profiler to display a graph of the downloading performance:

- The left side of the profiler displays information on the movie, its settings, and its state.
- The right section of the profiler shows the Timeline header and graph. In the graph, each bar represents an individual frame of the movie. The size of the bar corresponds to that frame's size in bytes. The red line beneath the Timeline header indicates whether a given frame streams in real time with the current modem speed set in the Control menu. If a bar extends above the red line, the movie must wait for that frame to load.

4 Choose View > Show Streaming to turn streaming off or on.

  If you turn streaming off, the movie starts over without simulating a Web connection.

5 Click a bar on the graph to display settings for the corresponding frame in the left window and stop the movie.

6 If you want, adjust the view of the graph:

- Choose View > Streaming Graph to show which frames will cause pauses.

  This default view displays alternating light and dark gray blocks representing each frame. The side of each block indicates its relative byte size. The first frame stores a symbol's contents, so is often larger than other frames.

- Choose View > Frame by Frame Graph to display the size of each frame.

  This view helps you see which frames contribute to streaming delays. If any frame block extends above the red line in the graph, the Flash Player halts playback until the entire frame downloads.

7 Close the test window to return to the normal authoring environment.

Once you've set up a test environment incorporating the Bandwidth Profiler, you can open any SWF directly in test mode. The file opens in a player window, using the Bandwidth Profiler and other selected viewing options.

For more information on debugging your movies, see "Using the Debugger" on page 353.

To generate a report listing the amount of data in the final Flash Player file:

1 Choose File > Publish Settings.

2 Select Generate Size Report.

3 Click Publish.

Flash generates a text file with the extension .txt. (If the movie file is myMovie.fla, the text file is myMovie Report.txt.) The report lists the amount of data in the final Flash Player file by frame.

Authoring and scripting guidelines

If you use good authoring practices when you author your movie and write scripts, your movies will have fewer problems. Use the following guidelines to help prevent problems and to fix them quickly when they do occur.

Using good authoring practices

It's a good idea to save multiple versions of your document as you work. Choose File > Save As to save a version with a different name every half hour. You can then determine when a problem began by using your version history to find the most recent file without the problem. Using this approach, you'll always have a functioning version, even if one file becomes corrupted.

Another important authoring practice is to test early, test often, and test on all target platforms to find problems as soon as they develop. Use Control > Test Movie to run your movie in test mode whenever you make a significant change or before saving a version. In test mode, the movie runs in the authoring application's version of Flash Player.

If your target audience will be viewing the movie on the Web, it's important to test the movie in a browser as well. In certain situations (for example, if you're developing an intranet site) you may know the browser and platform of your target audience. If you're developing for a Web site, however, test your movie in all browsers on all potential platforms.

Using good scripting practices

It's a good idea to follow these scripting practices:

- Use the `trace` action to send information to the Output window. (See "Using the trace action" on page 364.)
- Use the `comment` action to document how your ActionScript is supposed to work.
- Use consistent naming conventions to identify elements in a script. Start variable and function names with a lowercase letter and use a capital letter for each new word (`myVariableName`, `myFunctionName`). Start constructor function names with a capital letter (`MyConstructorFunction`). Most important, pick a style that makes sense to you and use it consistently.
- Use meaningful variable names that reflect what kind of information a variable contains. For example, a variable containing information about the last button pressed could be called `lastButtonPressed`. A name like `foo` would make it difficult to remember what the variable contains.
- Use the Movie Explorer to view the display list and view all ActionScript scripts in a movie. See "Using the Movie Explorer" on page 40.

Using an ActionScript troubleshooting checklist

In ActionScript, as with every scripting environment, coders commonly make several types of mistakes. The following list is a good place to start troubleshooting your movie:

- Check all target paths to make sure they are correct.
- Make sure you do not have frame actions on multiple layers that conflict with each other.

- If you're working with the Actions panel in normal mode, make sure the Expression check box is selected if your statement shouldn't have quotation marks around it.<<Loc: wording changed in this bullet to clarify meaning -- IMD>>

  If you're passing an expression in an action and haven't selected the Expression box, the value will be passed as a string. (See "String operators" on page 225.)

- Make sure multiple ActionScript elements do not have the same name.

  It's a good idea to give every variable, function, object, and property a unique name. Local variables are exceptions, though: they only need to be unique within their scope and are often reused as counters. See "Scoping a variable" on page 220.

- Use the `for..in` action to loop through the properties of movie clips, including child movie clips. You can use the `for..in` action with the `trace` action to send a list of properties to the Output window. (See "Repeating an action" on page 231.)

In addition, if some actions aren't working properly, make sure you're in test mode (Control > Test Movie). Only simple button and frame actions (for example, `gotoAndPlay` and `stop`) work in authoring mode.

For more tips on troubleshooting a Flash movie, see the Flash Support Center at www.macromedia.com/support/flash.

Using the Debugger

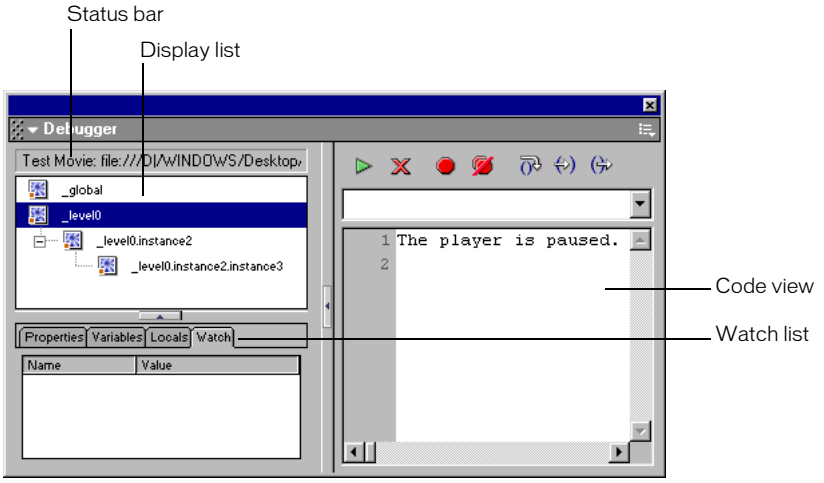
The Flash Debugger allows you to find errors in a movie while it's running in the Flash Player. You can use the Debugger in test mode with local files, or you can use the Debugger to test files on a Web server in a remote location. The Debugger lets you set breakpoints in your ActionScript that stop the Flash Player and step through the code as it runs. You can then go back to your scripts and edit them so that they produce the correct results.

Once activated, the Debugger status bar displays the URL or local file path of the movie, tells whether the Debugger is running in test mode or from a remote location, and shows a live view of the movie clip display list. When movie clips are added to or removed from the movie, the display list reflects the changes immediately. You can resize the display list by moving the horizontal splitter.

To activate the Debugger in test mode:

Choose Control > Debug Movie.

This opens the Debugger. It also opens the movie in test mode.



Debugging a movie from a remote location

You can debug a remote Flash movie using the stand-alone, ActiveX, or plug-in versions of the Flash Player. When exporting a Flash movie, you can enable debugging in your movie and create a debugging password. If you don't enable debugging, the Debugger will not activate.

To ensure that only trusted users can run your movies in the Flash Debug Player, you can publish your movie with a debugging password. As in JavaScript or HTML, it's possible for users to view client-side variables in ActionScript. To store variables securely, you must send them to a server-side application instead of storing them in the movie. However, as a Flash developer, you may have other trade secrets, such as movie clip structures, that you do not want revealed. You can use a debugging password to protect your work.

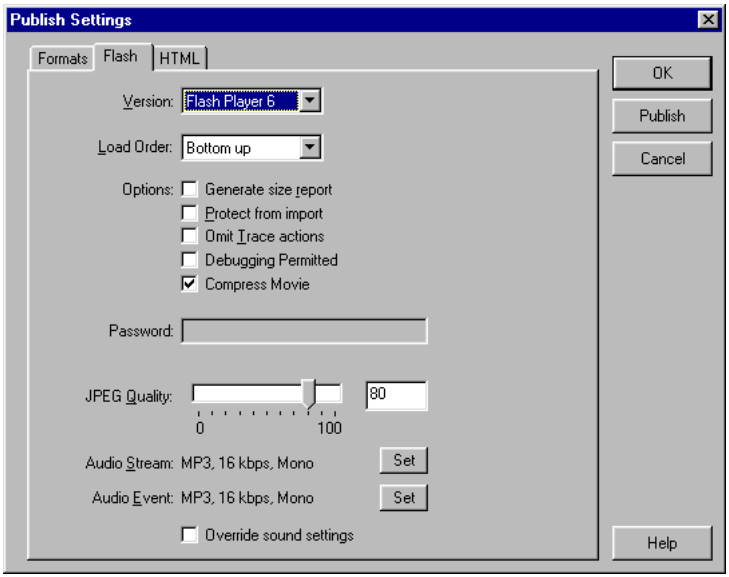
When you export, publish, or test a movie, Flash creates a SWD file that contains debug information. To debug remotely, you must place the SWD file in the same folder as the SWF file on the server.

Note: You cannot debug a movie from Flash Player 5 in the Flash MX authoring application. You cannot debug a movie from Flash Player 6 in the Flash 5 authoring application.

To enable remote debugging of a Flash movie:

1 Select File > Publish Settings.

2 On the Flash tab of the Publish Settings dialog box, select Debugging Permitted.



3 To set a password, enter a password in the Password box.

Once you set this password, no one can download information to the Debugger without the password. However, if you leave the password field blank, no password is required.

4 Select one of the following commands:

- Control > Debug Movie
- File > Export Movie
- File > Publish Settings > Publish

Flash creates a debugging file with the file extension .swd and saves it alongside the SWF file. The SWD file contains information that allows you to use breakpoints and step through code.

5 Place the movie's SWD file in the same directory as the SWF file on the server.

If the SWD file is not in the same directory as the SWF file, you can still debug remotely, but the Debugger will ignore breakpoints and you won't be able to step through code.

6 In Flash, choose Window > Debugger.

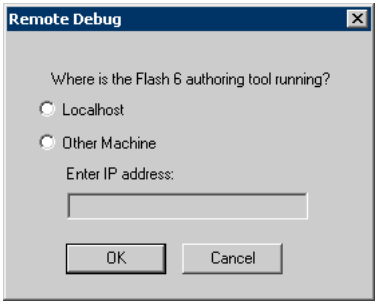
7 In the Debugger, from the Options pop-up menu, select the Enable Remote Debugging option.

To activate the Debugger from a remote location:

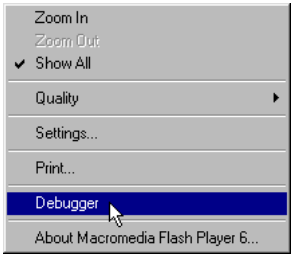
1 Open the Flash authoring application.

2 In a browser or in the stand-alone player, open the published movie (the SWF file) from the remote location.

The Remote Debug dialog box appears.



If that dialog box doesn't appear, Flash couldn't find the SWD file. In that case, right-click (Windows) or Control-click (Macintosh) in the movie to display the context menu, and select Debugger.



3 In the Remote Debug dialog box, select Localhost or Other Machine:

- Select Localhost if the Debug player and the Flash authoring application are on the same computer.
- Select Other Machine if the Debug player and the Flash authoring application are not on the same computer. Enter the IP address of the computer running the Flash authoring application.

4 When a connection is established, a password prompt appears. Enter your debugging password if you set one.

The display list of the movie appears in the Debugger.

Displaying and modifying variables

The Variables tab in the Debugger displays the names and values of any global and Timeline variables in the movie. If you change the value of a variable on the Variables tab, you can see the change reflected in the movie while it runs. For example, to test collision detection in a game, you could enter the variable value to position a ball in the correct location next to a wall.

The Locals tab in the Debugger displays the names and values of any local variables that are available wherever the movie has stopped at a breakpoint or anywhere else within a user-defined function.

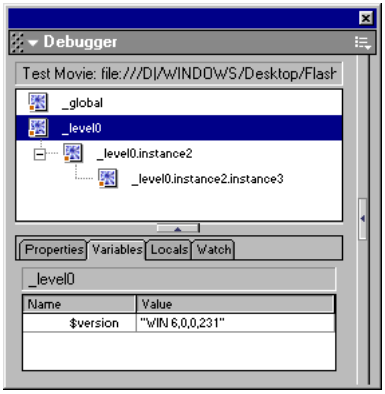
To display a variable:

1 Select the movie clip containing the variable from the display list.

To display global variables, select the `_global` clip in the display list.

2 Click the Variables tab.

The display list updates automatically as the movie plays. If a movie clip is removed from the movie at a specific frame, that movie clip, along with its variable and variable name, is also removed from the display list in the Debugger. However, if you mark a variable for the Watch list, that variable is not removed.



To modify a variable value:

Double-click the value and enter a new value.

The value cannot be an expression. For example, you can use `"Hello"`, `3523`, or `"http://www.macromedia.com"`, and you cannot use `x + 2` or `eval("name:" +i)`. The value can be a string (any value surrounded by quotation marks), a number, or a Boolean value (`true` or `false`).

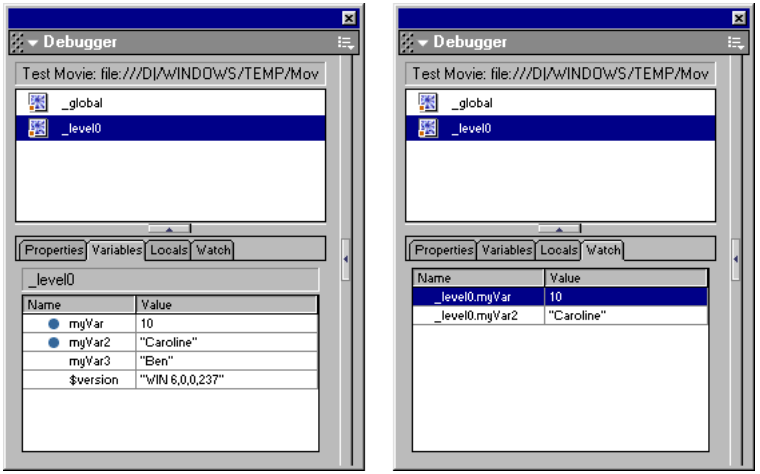
Note: To write the value of an expression to the Output window in test mode, use the `trace` action. See "Using the trace action" on page 364.

Using the Watch list

To monitor a set of critical variables in a manageable way, you can mark variables to appear in the Watch list. The Watch list displays the absolute path to the variable and the value. You can also enter a new variable value in the Watch list the same way you can in the Variables tab.

If you add a local variable to the Watch list, its value appears only when the player is stopped at a line of ActionScript where that variable is in scope. All other variables appear while the movie is playing. If the Debugger can't find the value of the variable, the value is listed as "Undefined."

The Watch list can display only variables, not properties or functions.



Variables marked for the Watch list and variables in the Watch list

To add variables to the Watch list, do one of the following:

- On the Variables or Locals tab, right-click (Windows) or Control-click (Macintosh) a selected variable and choose Watch from the context menu. A blue dot appears next to the variable.
- On the Watch tab, right-click (Windows) or Control-click (Macintosh) and choose Add from the context menu. Enter the target path to the variable name and the value in the fields.

To remove variables from the Watch list:

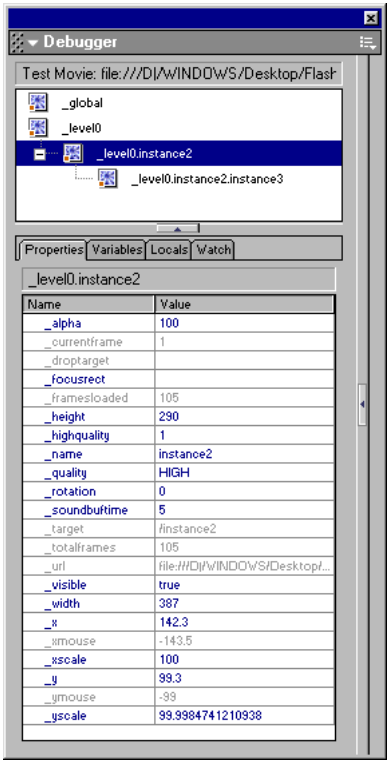
On the Watch tab, right-click (Windows) or Control-click (Macintosh) and choose Remove from the context menu.

Displaying movie properties and changing editable properties

The Debugger's Properties tab displays all the property values of any movie clip on the Stage. You can change a value and see its effect in the movie while it runs. Some movie clip properties are read-only and cannot be changed.

To display a movie clip's properties in the Debugger:

1. Select a movie clip from the display list.
2. Click the Properties tab in the Debugger.



To modify a property value:

Double-click the value and enter a new value.

The value cannot be an expression. For example, you can enter `50` or `"clearwater"` but you cannot enter `x + 50`. The value can be a string (any value surrounded by quotation marks), a number, or a Boolean value (`true` or `false`). You can't enter object or array values (for example, `{id: "rogue"}` or `[1, 2, 3]`) in the Debugger.

For more information, see "String operators" on page 225 and "Using operators to manipulate values in expressions" on page 223.

Note: To write the value of an expression to the Output window in test mode, use the `trace` action. See "Using the trace action" on page 364.

Setting and removing breakpoints

A breakpoint allows you to stop a movie running in the Flash Player at a specific line of ActionScript. You can use breakpoints to test possible trouble spots in your code. For example, if you've written a set of `if..else if` statements and can't determine which one is executing, you can add a breakpoint before the statements and step through them one by one in the Debugger.

You can set breakpoints in the Actions panel or in the Debugger. Breakpoints set in the Actions panel are saved with the Flash document (FLA) file. Breakpoints set in the Debugger are not saved in the FLA file and are valid only for the current debugging session.

To set and remove breakpoints in the Actions panel:<<Loc: text deleted -IMD>>

1. In the Script pane, select the line of code on which you want to set or remove a breakpoint.
2. Do one of the following:

- Click the Debug Options button above the Script pane.
- Right-click (Windows) or Control-click (Macintosh) to display the context menu.
- Press Control+Shift+B (Windows) or Command+Shift+B (Macintosh).

3. Choose Set Breakpoint, Remove Breakpoint, or Remove All Breakpoints.

To set and remove breakpoints in the Debugger:<<Loc: text deleted -IMD>>

1. In the Actions panel's Script pane, select the line of code on which you want to set or remove a breakpoint.
2. Do one of the following:

- In the Debugger, click the Toggle Breakpoint or Remove All Breakpoints button above the code view.
- In the Debugger, right-click (Windows) or Control-click (Macintosh) to display the context menu, and choose Breakpoint, Remove Breakpoint, or Remove All Breakpoints.
- Click in the Debugger window, and press Control+Shift+B (Windows) or Command+Shift+B (Macintosh).

  Once the Flash Player is stopped at a breakpoint, you can step into, step over, or step out of that line of code. If you set a breakpoint in a comment or on an empty line in the Actions panel, the breakpoint is ignored.

Stepping through lines of code

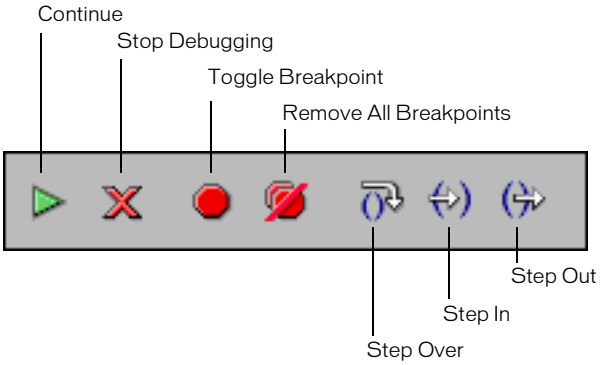
When you start a debugging session, the Flash Player is paused. If you set breakpoints in the Actions panel, you can simply click the Continue button to play the movie until it hits a breakpoint. For example, in the following code, suppose a breakpoint is set inside a button on the line `myFunction()`:

```
on(press){
  myFunction();
}
```

When you click the button, the breakpoint is reached and the Flash Player pauses. You can now step in to bring the Debugger to the first line of the `myFunction` function wherever it is defined in the document. You can also step through or out of the function.

If you didn't set breakpoints in the Actions panel, you can use the jump menu in the Debugger to select any script in the movie. Once you've selected a script, you can add breakpoints to it. After adding breakpoints, you must click the Continue button to start the movie. The Debugger stops when it reaches the breakpoint.

As you step through lines of code, the values of variables and properties change in the Watch list and in the Variables, Locals, and Properties tabs. A yellow arrow along the left side of the Debugger's code view indicates the line at which the Debugger stopped. Use the following buttons along the top of the code view:



Step In advances the Debugger (indicated by the yellow arrow) into a function. Step In works only for user-defined functions.

In the following example, if you place a breakpoint at line 7 and click Step In, the Debugger advances to line 2, and a subsequent click of Step In will advance you to line 3. Clicking Step In for lines that do not have user-defined functions in them advances the Debugger over a line of code. For example, if you stop at line 2 and choose Step In, the Debugger advances to line 3, as in the following example:

```
1 function myFunction() {
2 x = 0;
3 y = 0;
4 }
5
6 mover = 1;
7 myFunction();
8 mover = 0;
```

Step Out advances the Debugger out of a function. This button works only if you are currently stopped in a user-defined function; it moves the yellow arrow to the line after the one where that function was called. In the example above, if you place a breakpoint at line 3 and click Step Out, the Debugger moves to line 8. Clicking Step Out at a line that is not within a user-defined function is the same as clicking Continue. For example, if you stop at line 6 and click Step Out, the player continues executing the script until it encounters a breakpoint.

Step Over advances the Debugger over a line of code. This button moves the yellow arrow to the next line in the script and ignores any user-defined functions. In the example above, if you are stopped at line 7 and click Step Over, you go directly to line 8, and the function `myFunction` is called in the process.

Continue leaves the line at which the player is stopped and continues playing until a breakpoint is reached.

Stop Debugging makes the Debugger inactive, but continues to play the movie in the Flash Player.

Using the Output window

In test mode, the Output window displays information to help you troubleshoot your movie. Some information, such as syntax errors, is displayed automatically. You can display other information by using the List Objects and List Variables commands. (See "Listing a movie's objects" on page 362 and "Listing a movie's variables" on page 363.)

If you use the `trace` action in your scripts, you can send specific information to the Output window as the movie runs. This could include notes about the movie's status or the value of an expression. (See "Using the trace action" on page 364.)

To display the Output window:

1. If your movie is not running in test mode, choose Control > Test Movie.
2. Choose Window > Output.

   The Output window appears.

   Note: If there are syntax errors in a script, the Output window appears automatically.

3. To work with the contents of the Output window, use the Options pop-up menu in the upper right corner:

- Choose Options > Copy to copy the contents of the Output window to the Clipboard.
- Choose Options > Clear to clear the contents of the Output window.
- Choose Options > Save to File to save the window contents to a text file.
- Choose Options > Print to print the window contents.
- Choose Options > Find to search for a string of text.
- Choose Options > Find Again to search again for the same string of text.

Listing a movie's objects

In test mode, the List Objects command displays the level, frame, object type (shape, movie clip, or button), target paths, and instance names of movie clips, buttons, and text fields in a hierarchical list. This is especially useful for finding the correct target path and instance name. Unlike the Debugger, the list does not update automatically as the movie plays; you must choose the List Objects command each time you want to send the information to the Output window.

The List Objects command does not list all ActionScript data objects. In this context, an object is considered to be a shape or symbol on the Stage.

To display a list of objects in a movie:

1. If your movie is not running in test mode, choose Control > Test Movie.
2. Choose Debug > List Objects.

A list of all the objects currently on the Stage is displayed in the Output window, as in this example:

```
Level #0: Frame=1 Label="Scene_1"
  Button: Target="_level0.myButton"
    Shape:
  Movie Clip: Frame=1 Target="_level0.myMovieClip"
    Shape:
  Edit Text: Target="_level0.myTextField" Text="This is sample text."
```

Listing a movie's variables

In test mode, the List Variables command displays a list of all the variables currently in the movie. This is especially useful for finding the correct variable target path and variable name. Unlike the Debugger, the list does not update automatically as the movie plays; you must choose the List Variables command each time you want to send the information to the Output window.

The List Variables command also displays global variables declared with the `_global` identifier. The global variables are displayed at the top of the List Variables output in a section titled "Global Variables," and each variable is prefixed with `_global`.

In addition, the List Variables command displays getter/setter properties—properties that are created with the `Object.addProperty` method and invoke "get" or "set" methods. A getter/setter property is displayed alongside any other properties in the object it belongs to. To make these properties easily distinguishable from ordinary variables, the value of a getter/setter property is prefixed with the string `[getter/setter]`. The value displayed for a getter/setter property is determined by evaluating the "get" function of the property.

To display a list of variables in a movie:

1. If your movie is not running in test mode, choose Control > Test Movie.
2. Choose Debug > List Variables.

A list of all the variables currently in the movie is displayed in the Output window, as in this example:

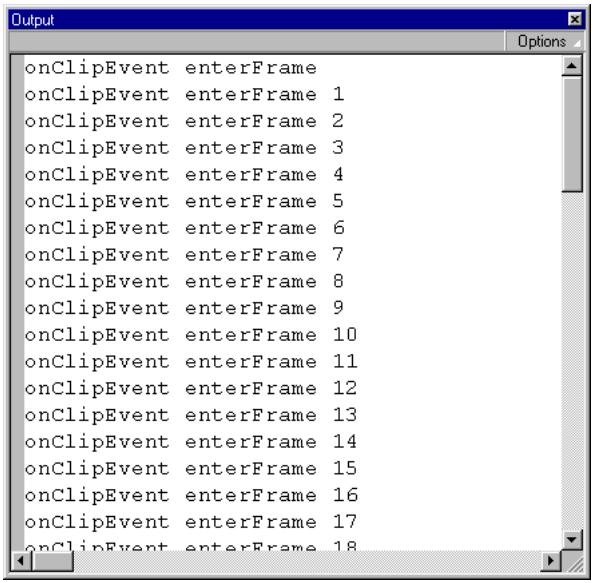
```
Global Variables:
  Variable _global.MyGlobalArray = [object #1] [
    0:1,
    1:2,
    2:3
  ]
Level #0:
  Variable _level0.$version = "WIN 6,0,0,101"
  Variable _level0.RegularVariable = "Gary"
  Variable _level0.AnObject = [object #1] {
    MyProperty: [getter/setter] 3.14159
  }
```

Using the trace action

When you use the `trace` action in a script, you can send information to the Output window. For example, while testing a movie or scene, you can send specific programming notes to the window or have specific results appear when a button is pressed or a frame is played. The `trace` action is similar to the JavaScript `alert` statement.

When you use the `trace` action in a script, you can use expressions as parameters. The value of an expression is displayed in the Output window in test mode, as in the following:

```
onClipEvent(enterFrame){
  trace("onClipEvent enterFrame " + enterFrame++)
}
```



The `trace` action returns values that are displayed in the Output window.

Updating the Flash Player for testing

You can download the latest version of the Flash Player from the Macromedia Web site and use it to test your movies in Flash MX with the most recent version of the Flash Player.

CHAPTER 20
Publishing

When you're ready to deliver your movie to an audience, you can publish the Macromedia Flash MX document (FLA file) for playback. By default, the Publish command creates the Flash SWF file and an HTML document that inserts your Flash movie in a browser window.

When you export a Flash movie file in Flash MX format, text is encoded in Unicode format, providing support for international character sets, including double-byte fonts. Likewise, your Flash Player 6 supports Unicode encoding. See "Unicode text encoding in Flash movies" on page 366.

You can also publish the FLA file in alternative file formats—GIF, JPEG, PNG, and QuickTime—with the HTML needed to display them in the browser window. Alternative formats enable a browser to display your movie's animation and interactivity for users who don't have the Flash Player 6 installed. When you publish a FLA file in alternative file formats, the settings for each file format are stored with the FLA file.

You can export the FLA file in a variety of formats as well. Exporting FLA files is similar to publishing FLA files in alternative file formats, except that the settings for each file format are not stored with the FLA file. See Chapter 21, "Exporting," on page 395.

As an alternative to using the Publish command, if you're proficient in HTML, you can create your own HTML document with any HTML editor and include the tags required to display a Flash movie. See "Configuring a Web server for Flash" on page 394.

Before you publish your movie, it's important to test how the movie works using the Test Movie and Test Scene commands. For more information, see "Testing movie download performance" under Help > Using Flash.

Playing your Flash movies

The Macromedia Flash file format (SWF) is the format for deploying Flash content.

You can play a Flash movie in the following ways:

- In Internet browsers such as Netscape Navigator and Internet Explorer that are equipped with the Flash Player 6
- With the Flash Xtra in Director and Authorware
- With the Flash ActiveX control in Microsoft Office and other ActiveX hosts
- As part of a QuickTime movie
- As a stand-alone movie called a projector

The Flash movie format, SWF, is an open standard that is supported by other applications. For more information about Flash file formats, see the Macromedia Web site at www.macromedia.com/software/flashplayer/.

Unicode text encoding in Flash movies

The Macromedia Flash (SWF) file format published in Flash MX format uses Unicode encoding for text and user interface strings. Flash Player 6 supports Unicode-encoded content in Flash movies.

About Unicode

Unicode is the universal character encoding standard for text representation in computer processing. Unicode provides a consistent way of encoding multilingual plain text, assigning each character a unique numeric value and name. Unicode defines codes for characters used in the major languages written today. Scripts include the European alphabetic scripts, Middle Eastern right-to-left scripts, and scripts of Asian languages. Unicode also includes punctuation marks, diacritics, mathematical symbols, technical symbols, and so on.

The two most common forms of Unicode encoding are *UTF-16* (where UTF stands for Unicode Transformation Format) and *UTF-8*. UTF-16 encoding is a 16-bit format that represents each code point (each text character, non-spacing accent, or other character representation) as a sequence of two bytes. UTF-8 is a scheme for representing the 16-bit code point as a sequence of one to four bytes that can be stored, retrieved, and transmitted over a network.

About Unicode support in Flash movies

Text and user interface strings in Flash MX documents (FLA files) are created using double-byte character set (DBCS) encoding. When Flash movies are published or exported in Flash MX format or later, text and UI strings are encoded using Unicode UTF-8, an 8-bit encoding format. The Flash Player stores characters in both UTF-8 and UTF-16 format.

Flash movies in Flash 5 format or earlier use mixed multibyte encoding (the Latin-1 character set for European languages, and the Shift-JIS character set for Asian languages). Text encoding in these files is supported by Flash Player 6, just as it is in earlier versions of the player.

Flash Player versions earlier than Flash Player 6 do not support Unicode. Players in these versions may not be able to read text or UI strings in SWF files that are of the Flash MX format.

When SWF files in Flash MX format are imported back into Flash MX, text and UI strings are converted back into DBCS format. These files can be edited in the Flash authoring environment.

Selecting an encoding language

The encoding conversion for export or import uses the language selected in the Regional Languages control panel (Windows 2000 or later) or the Fonts tab of the Appearance control panel (Macintosh).

To choose an encoding language (Windows):

1. In the Control Panel, select Regional Options.
2. With the General tab selected, under the Setting for the current user, choose a language from the Your Locale (location) pop-up menu.
3. Still on the General tab, under Language settings for the system, click the Set default button.
4. In the Select System Locale dialog box, choose a language for the default language.
5. Click OK.

To choose an encoding language (Macintosh OS 9.x):

When you install the Macintosh Language Kit, the encoding is automatically chosen.

To choose an encoding language (Macintosh OS X.x):

1. In System Preferences, select International.
2. Select your primary language.
3. Click OK.

About procedures supported with Unicode

Unicode support in Flash MX enables a variety of text-related procedures:

- Users can display Flash movies using a different language than that used by their operating systems. For example, a user with an English-based operating system can view a Flash movie containing Korean text. To take advantage of multilanguage support, you must have fonts capable of rendering the text on your system. For example, suppose you are viewing a Flash movie with Korean text on an English-based system. Either you must have Korean fonts installed on your computer or the fonts must be embedded into the Flash movie.
- Line breaking is interpreted for each language supported by Flash.
- The Flash Player can interpret externally loaded ActionScript files (for example, files loaded using the `#include` action), regardless of what language was used to create those files. For example, a Flash Player 6 on an English-based operating system can interpret an external ActionScript file created on a Japanese operating system.
- Flash Player 6 can interpret XML content as UTF-16, UTF-8, Latin-1, or Shift-JIS encoded text.

Publishing Flash documents

Publishing a Flash document is a two-step process. First, you choose publishing file formats and select file format settings with the Publish Settings command. Then you publish the Flash document using the Publish command.

Depending on the options you specify in the Publish Settings dialog box, the Publish command creates the following files:

- The Flash movie.
- Alternate images in a variety of formats that appear automatically when the Flash Player is not available (GIF, JPEG, PNG, and QuickTime).
- The supporting HTML document required to display a movie (or an alternate image) in a browser and control browser settings.
- Stand-alone projector files for both Windows and Macintosh systems and QuickTime videos from Flash movies (EXE, HQX, or MOV files, respectively).

Note: To alter or update a Flash movie created with the Publish command, you must edit the original Flash document and then use the Publish command again to preserve all authoring information. Importing a Flash movie into Flash removes some of the authoring information.

You can also publish a Flash document using default or previously selected settings.

To set general publish settings for a Flash document:

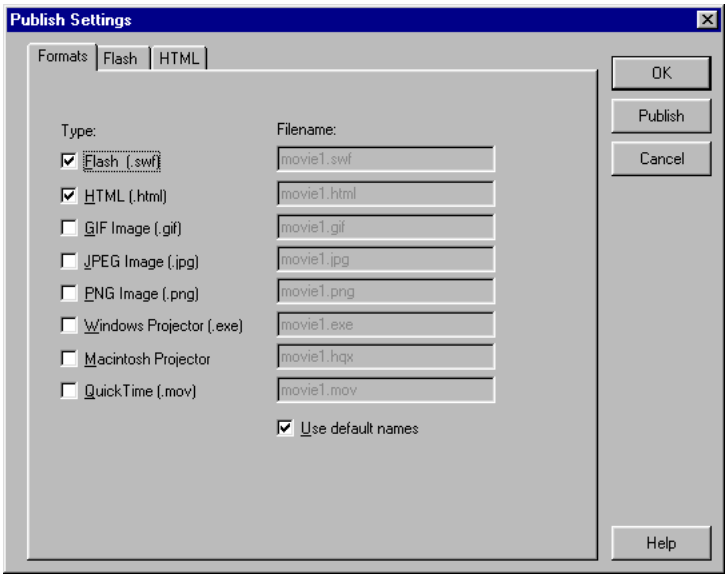
1 Do one of the following to specify where you will save the published files:

- Create the folder where you want to save the published files, and save your Flash document.
- Browse to and open an existing folder, and save your Flash document.

2 Choose File > Publish Settings.

3 In the Publish Settings dialog box, select the option for each file format you want to create.

The Flash SWF format is selected by default. The HTML format is also selected by default, because an HTML file is required to display a SWF file in a browser. Tabs corresponding to the selected file formats appear above the current panel in the dialog box (except for Windows or Macintosh projector formats, which have no settings). For more information on publishing settings for individual file formats, see the sections that follow.



Publish Settings dialog box, with Flash and HMTL file types selected

4 For Filename, choose one of the following options:

- Select Use Default Names (the default setting).
- Deselect Use Default Name and enter a new filename for each selected file format.

  You can enter a path with the filename to specify where you want to publish the file. You can set a different path for each file format (for example, if you want to publish the Flash SWF file in one location and the HTML file in another location). In Windows, use backslashes (\) to specify the directory-folder-file hierarchy; on the Macintosh, use colons (:). Specify the drive name for an absolute path. In Windows only: for a relative path, use ..\ to indicate the path to the hard drive. For example:

  In Windows, specify an absolute path as `C:\Folder\filename.swf` where `C:` is the drive name, `\Folder` specifies the folder name, and `filename.swf` is the name of the file. Specify the relative path as `..\Folder\filename.swf`.

  On the Macintosh, specify an absolute path as `HardDrivename:Folder:filename.swf`.

5 To create a stand-alone projector file, select Windows Projector or Macintosh Projector.

Note: The Windows version of Flash names a Macintosh projector file with the .hqx extension. You can create a Macintosh projector using the Windows versions of Flash, but you must use a file translator such as BinHex to make the resulting file appear as an application file in the Macintosh Finder.

6 Click the tab for the format options you want to change. Specify publish settings for each format, as described in the following sections.

7 When you have finished setting options, do one of the following:

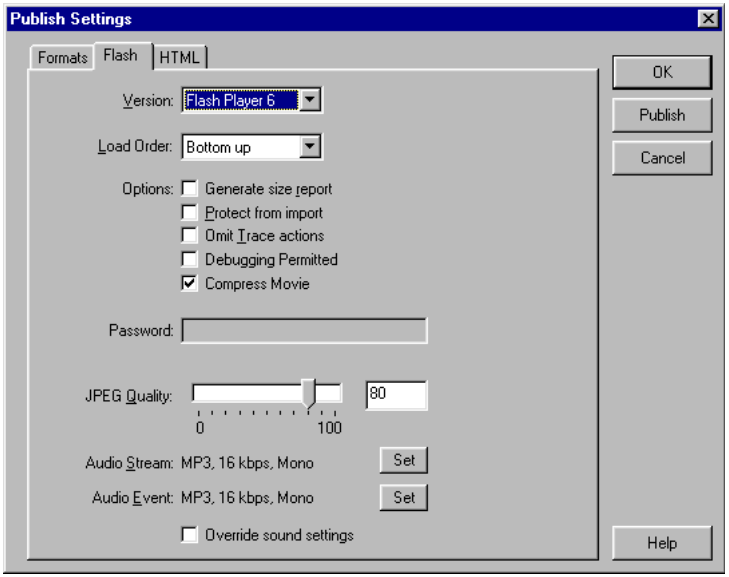
- To generate all the specified files and close the dialog box, click Publish.
- To save the settings with the FLA file and close the dialog box without publishing, click OK.

To publish a Flash document without choosing new publish settings:

Choose File > Publish to create the files in the formats and location specified in the Publish Settings dialog box (either the default settings, or settings you selected previously).

Choosing publish settings Flash movie format

When publishing a Flash movie, you can set image and sound compression options, and an option to protect your movie from being imported. Use the controls in the Flash panel of the Publish Settings dialog box to change the following settings.



Publish settings for a Flash movie

To choose publish settings for a Flash movie:

1. Choose File > Publish Settings and click the Flash tab.
2. Choose a Player Version from the pop-up menu.
3. Choose a Load Order to set the order in which Flash loads a movie's layers for displaying the first frame of your movie: Bottom Up or Top Down.

   This option controls which parts of the movie Flash draws first over a slow network or modem connection.
4. Select Generate Size Report to generate a report listing the amount of data in the final Flash movie by file. See "Testing movie download performance" under Help > Using Flash.
5. To enable debugging of the published Flash movie, select any of the following options:

- Omit Trace Actions causes Flash to ignore Trace actions (`trace`) in the current movie. When you select this option, information from Trace actions is not displayed in the Output window.

  For more information, see "Using the Output window" under Help > Using Flash.
- Protect from Import prevents others from importing the Flash movie and converting it back into a Flash (FLA) document.
- Debugging Permitted activates the Debugger and allows debugging a Flash movie remotely. If you select this option, you can choose to password-protect your Flash movie.

- Flash Player 6 Version only: Compress Movie compresses the Flash movie to reduce file size and download time. This option is on by default and is most beneficial when a file has a lot of text or ActionScript. A compressed file only plays in Flash Player 6.

6. If you selected Debugging Permitted in step 5, enter a password in the Password text box to prevent unauthorized users from debugging a Flash movie. If you add a password, others must enter the password before they can debug the movie. To remove the password, clear the Password field.

   For more information on the Debugger, see "Using the Debugger" under Help > Using Flash.

7. To control bitmap compression, adjust the JPEG Quality slider or enter a value.

   Lower image quality produces smaller files; higher image quality produces larger files. Try different settings to determine the best trade-off between size and quality; 100 provides the highest quality and least compression.

8. To set the sample rate and compression for all stream sounds or event sounds in the movie, click the Set button next to Audio Stream or Audio Event and choose options for Compression, Bit Rate, and Quality in the Sound Settings dialog box. Click OK when you are finished.

   Note: A stream sound begins playing as soon as enough data for the first few frames has been downloaded; it is synchronized to the Timeline. An event sound must download completely before it begins playing and continues playing until explicitly stopped.

   For more information on sound, see Chapter 6, "Adding Sound," under Help > Using Flash.

9. Select Override Sound Settings to use the settings selected in step 8 to override settings for individual sounds selected in the Sound section of the Property inspector. You may want to choose this option to create a smaller low-fidelity version of a movie.

   Note: If the Select Override Sound Settings option is deselected, Flash scans all stream sounds in the movie (including sounds in imported video) and publishes all stream sounds at the highest individual setting. This can increase file size, if one or more stream sounds has a high export setting.

10. Choose a Flash Player version. Not all Flash MX features work in movies published earlier than Flash Player 6.

11. To save the settings with the current file, click OK.

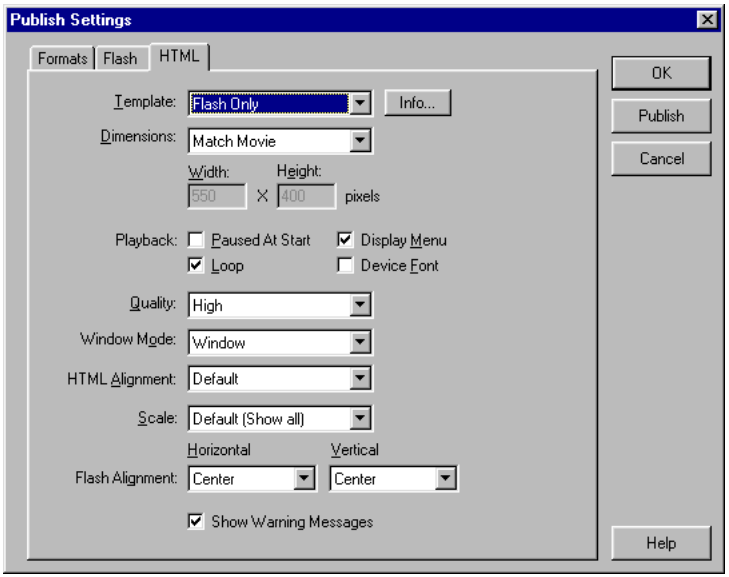
Setting publish settings for HTML documents accompanying Flash movies

Playing a Flash movie in a Web browser requires an HTML document that activates the movie and specifies browser settings. This document is generated automatically by the Publish command, from HTML parameters in a template document.

The template document can be any text file that contains the appropriate template variables—including a plain HTML file, one that includes code for special interpreters such as ColdFusion or Active Server Pages (ASP), or a template included with Flash (for more information, see "Configuring a Web server for Flash" on page 394).

You can customize a built-in template (see "Customizing HTML publishing templates" on page 383), or manually enter HTML parameters for Flash using any HTML editor (see "Editing Flash HTML settings" on page 386).

HTML parameters determine where the Flash movie appears in the window, the background color, the size of the movie, and so on, and set attributes for the OBJECT and EMBED tags. You can change these and other settings in the HTML panel of the Publish Settings dialog box. Changing these settings overrides options you've set in your movie.



Publish settings for HTML format

To publish HTML for displaying the Flash file:

1. Choose File > Publish Settings.

   The HTML file type is selected by default.

2. Enter a unique name for HTML Filename, or select Use Default Name to create a file with the Flash filename plus the .html extension.

3. Click the HTML tab to display HTML settings.

4. Choose an installed template to use from the Template pop-up menu; click the Info button to the right to display a description of the selected template. If you don't choose a template, Flash uses the Default.html template. If that template isn't present, Flash uses the first template in the list.

5. Choose a Dimensions option to set the values of the WIDTH and HEIGHT attributes in the OBJECT and EMBED tags:

- Choose Match Movie (the default) to use the size of the movie.
- Choose Pixels to enter the number of pixels for the width and height in the Width and Height field.
- Choose Percent to use a percentage of the browser window relative to the browser window.

6 Select Playback options to control the movie's play and features, as follows:

- Paused at Start pauses the movie until a user clicks a button in the movie or chooses Play from the shortcut menu. By default, the option is deselected and the movie begins to play as soon as it is loaded (the PLAY parameter is true).
- Loop repeats the movie when it reaches the last frame. Deselect this option to stop the movie when it reaches the last frame. (The LOOP parameter is on by default.)
- Display Menu displays a shortcut menu when users right-click (Windows) or Control-click (Macintosh) the movie. Deselect this option to display only About Flash in the shortcut menu. By default, this option is selected (the MENU parameter is true).
- For Windows only, select Device Font to substitute anti-aliased (smooth-edged) system fonts for fonts not installed on the user's system. Using device fonts increases the legibility of type at small sizes and can decrease the movie's file size. This option only affects movies containing static text (text that you created when authoring a movie and that does not change when the movie is displayed) set to display with device fonts. For more information, see "Using device fonts (horizontal text only)" on page 142.

7 Select Quality to determine the trade-off between processing time and applying anti-aliasing to smooth each frame before it is rendered on the viewer's screen, as follows. This option sets the QUALITY parameter's value in the OBJECT and EMBED tags.

- Low favors playback speed over appearance and does not use anti-aliasing.
- Auto Low emphasizes speed at first but improves appearance whenever possible. Playback begins with anti-aliasing turned off. If the Flash Player detects that the processor can handle it, anti-aliasing is turned on.
- Auto High emphasizes playback speed and appearance equally at first but sacrifices appearance for playback speed if necessary. Playback begins with anti-aliasing turned on. If the actual frame rate drops below the specified frame rate, anti-aliasing is turned off to improve playback speed. Use this setting to emulate the View > Antialias setting in Flash.
- Medium applies some anti-aliasing, but does not smooth bitmaps. It produces a better quality than the Low setting, but lower quality than the High setting.
- High (the default) favors appearance over playback speed and always uses anti-aliasing. If the movie does not contain animation, bitmaps are smoothed; if the movie has animation, bitmaps are not smoothed.
- Best provides the best display quality and does not consider playback speed. All output is anti-aliased and bitmaps are always smoothed.

8 For the Windows version of Internet Explorer 4.0 with the Flash ActiveX control, choose a Window Mode option for transparency, positioning, and layering.This option specifies the ALIGN attribute for the OBJECT, EMBED, and IMG tags.

- Window plays a Flash movie in its own rectangular window on a Web page, for the fastest animation. The option sets the WMODE parameter of the OBJECT tag to WINDOW.
- Opaque Windowless moves elements behind Flash movies (for example, with dynamic HTML) to prevent them from showing through, setting the WMODE parameter to OPAQUE.
- Transparent Windowless shows the background of the HTML page on which the movie is embedded through all transparent areas of the movie, but may slow animation. The option sets the WMODE parameter to TRANSPARENT.

9 Choose an HTML Alignment option to position the Flash movie window within the browser window:

- Default centers the movie in the browser window and crops edges if the browser window is smaller than the movie.
- Left, Right, Top or Bottom aligns movies along the corresponding edge of the browser window and crop the remaining three sides as needed.

10 Choose a Scale option to place the movie within specified boundaries, if you've changed the movie's original width and height. The Scale option sets the `SCALE` parameter in the `OBJECT` and `EMBED` tags.

- Default (Show All) displays the entire movie in the specified area without distortion while maintaining the original aspect ratio of the movies. Borders may appear on two sides of the movie.
- No Border scales the movie to fill the specified area and keeps the movie's original aspect ratio without distortion, cropping if needed.
- Exact Fit displays the entire movie in the specified area without preserving the original aspect ratio, which may cause distortion.
- No Scale prevents the movie from scaling when the Flash Player window is resized.

11 Choose a Flash Alignment option to set how the movie is placed within the movie window and how it is cropped, if necessary. This option sets the `SALIGN` parameter of the `OBJECT` and `EMBED` tags.

- For Horizontal alignment, choose Left, Center, or Right.
- For Vertical alignment, choose Top, Center, or Bottom.

12 Select Show Warning Messages to display error messages if tag settings conflict—for example, if a template has code referring to an alternate image that has not been specified.

13 To save the settings with the current file, click OK.

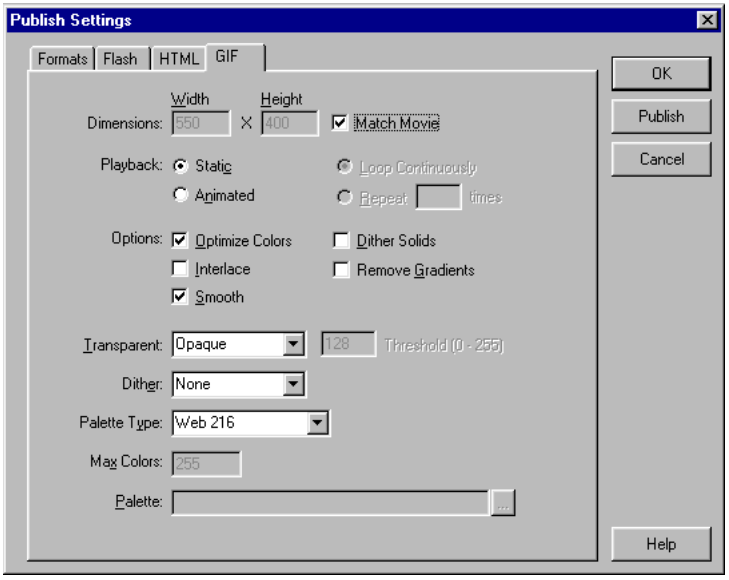
Choosing publish settings for GIF files

GIF files provide an easy way to export drawings and simple animations for use in Web pages. Standard GIF files are simply compressed bitmaps.

An animated GIF file (sometimes referred to as a GIF89a) offers a simple way to export short animation sequences. Flash optimizes an animated GIF, storing only frame-to-frame changes.

Flash exports the first frame in the movie as a GIF, unless you mark a different keyframe for export by entering the frame label #Static. Flash exports all the frames in the current movie to an animated GIF unless you specify a range of frames for export by entering the frame labels #First and #Last in the appropriate keyframes.

Flash can generate an image map for a GIF to maintain URL links for buttons in the original movie. Place the frame label #Map in the keyframe in which you want to create the image map. If you don't create a frame label, Flash creates an image map using the buttons in the last frame of the movie. You can create an image map only if the `$IM` template variable is present in the template you select. See "Creating an image map" on page 385.



Publish settings for GIF format

To publish a GIF file with the Flash file:

1. Choose File > Publish Settings.
2. Select the GIF Image type. Enter a unique name for Filename, or select Use Default Name to create a file with the Flash filename plus the .gif extension.
3. Click the GIF panel to display its settings.
4. For Dimensions, enter a Width and Height in pixels for the exported bitmap image, or select Match Movie to make the GIF the same size as the Flash movie and maintain the aspect ratio of your original image.
5. Choose a Playback option to determine whether Flash creates a still (Static) image or an animated GIF (Animation). If you choose Animation, select Loop Continuously or enter the number of repetitions.
6. Choose an option to specify a range of appearance settings for the exported GIF:

- Optimize Colors removes any unused colors from a GIF file's color table. This option reduces the file size by 1000 to 1500 bytes without affecting image quality, but slightly increases the memory requirements. This option has no effect on an adaptive palette. (An adaptive palette analyzes the colors in the image and creates a unique color table for the selected GIF.)

- Interlace makes the exported GIF display in a browser incrementally as it downloads. An interlaced GIF provides the user with basic graphic content before the file has completely downloaded and may download faster over a slow network connection. Do not interlace an animated GIF.
- Smooth applies anti-aliasing to an exported bitmap to produce a higher-quality bitmap image and improve text display quality. However, smoothing may cause a halo of gray pixels to appear around an anti-aliased image placed on a colored background, and it increases the GIF file size. Export an image without smoothing if a halo appears or if you're placing a GIF transparency on a multicolored background.
- Dither Solids applies dithering to solid colors as well as gradients. See Dither options in step 8.
- Remove Gradients, turned off by default, converts all gradients fills in the movie to solid colors using the first color in the gradient. Gradients increase the size of a GIF and often are of poor quality. If you use this option, choose the first color of your gradients carefully to prevent unexpected results.

7 Choose a Transparent option to determine the transparency of the movie's background and the way alpha settings are converted to GIF:

- Opaque to make the background a solid color.
- Transparent to make the background transparent.
- Alpha to set partial transparency. Then enter a Threshold value between 0 and 255. A lower value results in greater transparency. A value of 128 corresponds to 50% transparency.

8 Choose a Dither option to specify how pixels of available colors are combined to simulate colors not available in the current palette. Dithering can improve color quality, but it increases the file size. Choose from the following options:

- None turns off dithering and replaces colors not in the basic color table with the solid color from the table that most closely approximates the specified color. Not dithering can produce smaller files but unsatisfactory colors.
- Ordered provides good-quality dithering with the smallest increase in file size.
- Diffusion provides the best-quality dithering but increases file size and processing time more than ordered dithering. It also only works with the Web 216 color palette selected.

9 Choose a Palette Type to define the image's color palette:

- Web 216 uses the standard 216-color browser-safe palette to create the GIF image, for good image quality and the fastest processing on the server.
- Adaptive analyzes the colors in the image and creates a unique color table for the selected GIF. This option is best for systems displaying thousands or millions of colors; it creates the most accurate color for the image but results in a file size larger than a GIF created with the Web 216 palette. To reduce the size of a GIF with an adaptive palette, use the Max Colors option in step 10 to decrease the number of colors in the palette.
- Web Snap Adaptive is the same as the Adaptive palette option except that it converts very similar colors to the Web 216 color palette. The resulting color palette is optimized for the image, but when possible, Flash uses colors from Web 216. This produces better colors for the image when the Web 216 palette is active on a 256-color system.

- Custom to specify a palette that you have optimized for the selected image. This option has the same processing speed as the Web 216 palette. To use this option, you should know how to create and use custom palettes. To choose a custom palette, click the Ellipsis (...) button to the right of the Palette box at the bottom of the dialog box and select a palette file. Flash supports palettes saved in the ACT format, exported by Macromedia Fireworks and other leading graphics applications; for more information, see "Importing and exporting color palettes" on page 88.

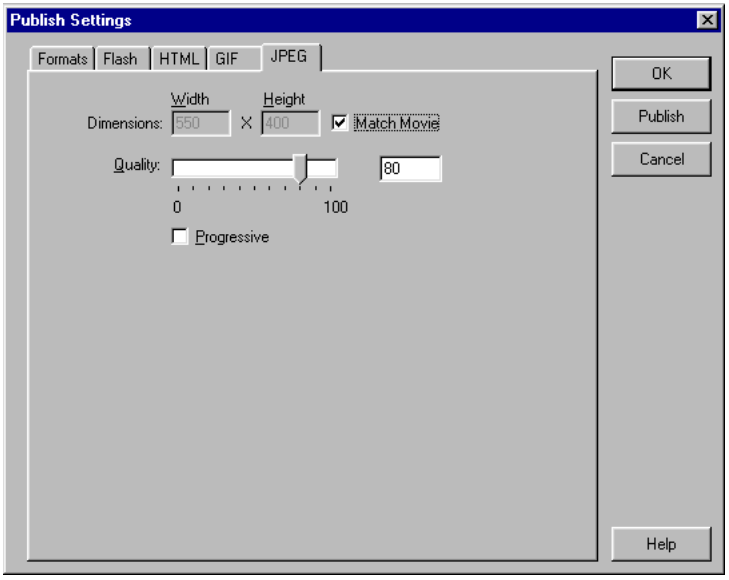
10 If you selected the Adaptive or Web Snap Adaptive palette in step 9, enter a value for Max Colors to set the number of colors used in the GIF image. Choosing a smaller number of colors can produce a smaller file but may degrade the colors in the image.

11 To save the settings with the current file, click OK.

Choosing publish settings for JPEG files

The JPEG format lets you save an image as a highly compressed, 24-bit bitmap. Generally, GIF is better for exporting line art, while JPEG is better for images that include continuous tones like photographs, gradients, or embedded bitmaps.

Flash exports the first frame in the movie as a JPEG, unless you mark a different keyframe for export by entering the frame label #Static.



Publish settings for JPEG format

To publish a JPEG file with the Flash movie:

1 Choose File > Publish Settings.

2 Select the JPEG Image type. Enter a unique name for Filename, or select Use Default Name to create a file with the Flash filename plus the .jpg extension.

3 Click the JPEG panel to display its settings.

4 For Dimensions, enter a Width and Height in pixels for the exported bitmap image, or select Match Movie to make the JPEG the same size as the Flash movie and maintain the aspect ratio as your original image.

5 For Quality, drag the slider or enter a value to control the amount of JPEG file compression used.

Lower image quality produces smaller files, while higher image quality produces larger files. Try different settings to determine the best trade-off between size and quality.

Note: You can set the bitmap export quality per object using the Bitmap Properties dialog box to change the object's compression setting. Selecting the default compression option in the Bitmap Properties dialog box applies the Publish Settings' JPEG Quality option. See "Setting bitmap properties" under Help › Using Flash.

6 Select Progressive to display Progressive JPEG images incrementally in a Web browser, to make images appear faster when loaded over a slow network connection.

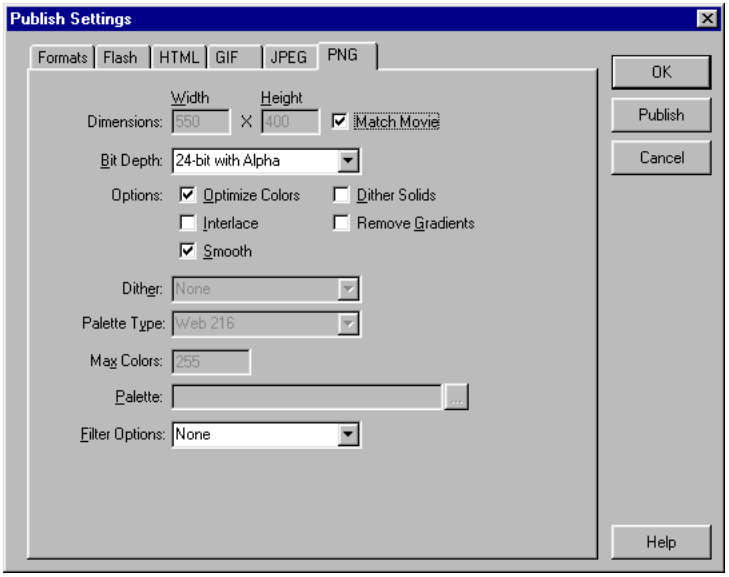
This option is similar to interlacing in GIF and PNG images.

7 To save the settings with the current file, click OK.

Choosing publish settings for PNG files

PNG is the only cross-platform bitmap format that supports transparency (an alpha channel). It is also the native file format for Macromedia Fireworks.

Flash exports the first frame in the movie as a PNG, unless you mark a different keyframe for export by entering the frame label #Static.



Publish settings for PNG format

To publish a PNG file with the Flash file:

1. Choose File > Publish Settings.
2. Select the PNG Image type. Enter a unique name for Filename, or select Use Default Name to create a file with the Flash filename plus the .png extension.
3. Click the PNG panel to display its settings.
4. For Dimensions, enter a Width and Height in pixels for the exported bitmap image, or select Match Movie to make the PNG the same size as the Flash movie and maintain the aspect ratio as your original image.
5. Choose a Bit Depth to set the number of bits per pixel and colors to use in creating the image:

- Choose 8-bit for a 256-color image.
- Choose 24-bit for thousands of colors.
- Choose 24-bit with Alpha for thousands of colors with transparency (32 bits).

The higher the bit depth, the larger the file.

6. Choose Options to specify appearance settings for the exported PNG:

- Optimize Colors removes any unused colors from a PNG file's color table. This option reduces the file size by 1000 to 1500 bytes without affecting image quality, but slightly increases the memory requirements. This option has no effect on an adaptive palette.
- Interlace makes the exported PNG display in a browser incrementally as it downloads. An interlaced PNG provides the user with basic graphic content before the file has completely downloaded and may download faster over a slow network connection. Do not interlace an animated PNG.
- Smooth applies anti-aliasing to an exported bitmap to produce a higher-quality bitmap image and improve text display quality. However, smoothing may cause a halo of gray pixels to appear around an anti-aliased image placed on a colored background, and it increases the PNG file size. Export an image without smoothing if a halo appears or if you're placing a PNG transparency on a multicolored background.
- Dither Solids applies dithering to solid colors and gradients. See Dither options in step 7.
- Remove Gradients, turned off by default, converts all gradient fills in the movie to solid colors using the first color in the gradient. Gradients increase the size of a PNG and often are of poor quality. If you use this option, choose the first color of your gradients carefully to prevent unexpected results.

7. If you chose 8-bit for Bit Depth in step 5, choose a Dither option to specify how pixels of available colors are mixed to simulate colors not available in the current palette. Dithering can improve color quality, but it increases the file size. Choose from the following options:

- None turns off dithering and replaces colors not in the basic color table with the solid color from the table that most closely approximates the specified color. Not dithering can produce smaller files but unsatisfactory colors.
- Ordered provides good-quality dithering with the smallest increase in file size.
- Diffusion provides the best-quality dithering but increases file size and processing time more than ordered dithering. It also only works with the Web 216 color palette selected.

8 Choose Palette Type to define the color palette for the PNG image:

- Web 216 uses the standard 216-color browser-safe palette to create the PNG image, for good image quality and the fastest processing on the server.
- Adaptive analyzes the colors in the image and creates a unique color table for the selected PNG. This option is best for systems displaying thousands or millions of colors; it creates the most accurate color for the image but results in a file size larger than a PNG created with the Web 216 palette.
- Web Snap Adaptive is the same as the Adaptive palette option except that it converts very similar colors to the Web 216 color palette. The resulting color palette is optimized for the image, but when possible, Flash uses colors from Web 216. This produces better colors for the image when the Web 216 palette is active on a 256-color system.

  To reduce the size of a PNG with an adaptive palette, use the Max Colors option to decrease the number of palette colors, as described in the next step.
- Custom specifies a palette that you have optimized for the selected image. This option has the same processing speed as the Web 216 palette. To use this option, you should know how to create and use custom palettes. To choose a custom palette, click the Ellipsis (...) button to the right of the Palette box at the bottom of the dialog box and select a palette file. Flash supports palettes saved in the ACT format, exported by Macromedia Fireworks and other leading graphics applications; for more information, see "Importing and exporting color palettes" on page 88.

9 If you selected the Adaptive or Web Snap Adaptive palette in step 8, enter a value for Max Colors to set the number of colors used in the PNG image. Choosing a smaller number of colors can produce a smaller file but may degrade the colors in the image.

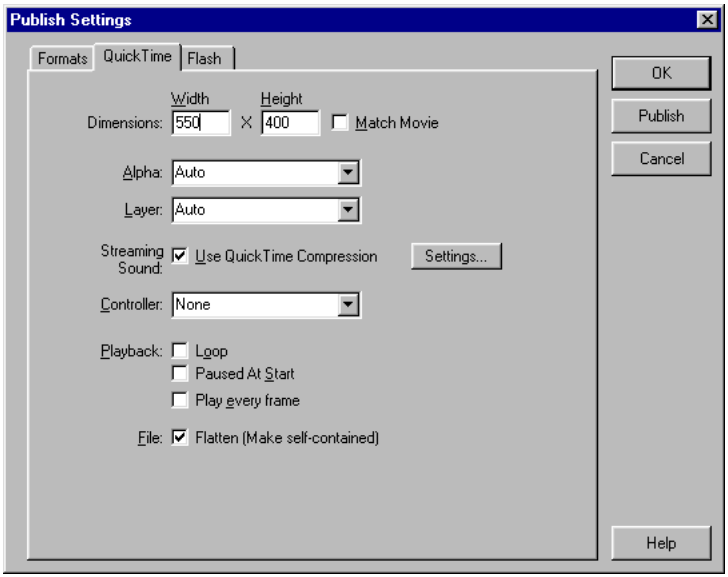
10 Choose Filter Options to select a line-by-line filtering method to make the PNG file more compressible, and experiment with the different options for a particular image:

- None turns off filtering.
- Sub transmits the difference between each byte and the value of the corresponding byte of the prior pixel.
- Up transmits the difference between each byte and the value of the corresponding byte of the pixel immediately above.
- Average uses the average of the two neighboring pixels (left and above) to predict the value of a pixel.
- Path computes a simple linear function of the three neighboring pixels (left, above, upper left), and then chooses as a predictor the neighboring pixel closest to the computed value.
- Adaptive analyzes the colors in the image and creates a unique color table for the selected PNG. This option is best for systems displaying thousands or millions of colors; it creates the most accurate color for the image but results in a file size larger than a PNG created with the Web 216 palette. You can reduce the size of a PNG created with an adaptive palette by decreasing the number of colors in the palette.

11 To save the settings with the current file, click OK.

Choosing publish settings for QuickTime 4 movies

The QuickTime Publish option creates movies in the QuickTime 4 format, copying the Flash movie onto a separate QuickTime track. The Flash movie plays in the QuickTime movie exactly as it does in the Flash Player, retaining all of its interactive features. If the Flash movie also contains a QuickTime movie, Flash copies it to its own track in the new QuickTime file. For more information on QuickTime movies, see your QuickTime documentation.



Publish settings for QuickTime format

To publish a QuickTime 4 movie with the Flash file:

1. Choose File > Publish Settings.
2. Select the QuickTime Image type. Enter a unique name for Filename, or select Use Default Name to create a file with the Flash filename plus the .mov extension.
3. Click the QuickTime panel to display its settings.
4. For Dimensions, enter a Width and Height in pixels for the exported QuickTime movie, or select Match Movie to make the QuickTime movie the same size as the Flash movie and keep its aspect ratio.
5. Choose an Alpha option to control the transparency (alpha) mode of the Flash track in the QuickTime movie without affecting any alpha settings in the Flash movie:

- Alpha Transparent to make the Flash track transparent and show any content in tracks behind the Flash track.
- Copy to make the Flash track opaque and mask all content in tracks behind the Flash track.
- Auto to make the Flash track transparent if it is on top of any other tracks, but opaque if it is the bottom or only track in the movie.

6 Choose a Layer option to control where the Flash track plays in the stacking order of the QuickTime movie:

- Top to place the Flash track always on top of other tracks in the QuickTime movie.
- Bottom to place the Flash track always behind other tracks.
- Auto to place the Flash track in front of other tracks if Flash objects are in front of video objects within the Flash movie, and behind all other tracks if Flash objects are not in front.

7 Select Streaming Sound to have Flash export all of the streaming audio in the Flash movie to a QuickTime sound track, recompressing the audio using the standard QuickTime audio settings. To change these options, click Audio Settings; see your QuickTime documentation for more information.

8 Choose Controller to specify the type of QuickTime controller used to play the exported movie—None, Standard, or QuickTime VR.

9 Select Playback options to control how QuickTime plays a movie:

- Looping repeats the movie when it reaches the last frame.
- Paused at Start pauses the movie until a user clicks a button in the movie or chooses Play from the shortcut menu. By default, the option is deselected and the movie begins to play as soon as it is loaded.
- Play Every Frame displays every frame of the movie without skipping to maintain time and does not play sound.

10 Choose File Flatten (Make Self-Contained) to combine the Flash content and imported video content into a single QuickTime movie. Deselecting this option makes the QuickTime movie refer to the imported files externally; the movie won't work properly if these files are missing.

11 To save the settings with the current file, click OK.

About HTML publishing templates

Flash HTML templates let you control what movie goes on a Web page and how it looks and plays back in the Web browser. A Flash template is a text file that contains both unchanging HTML code and template code or variables (which differ from ActionScript variables). When you publish a Flash movie, Flash replaces the variables in the template you selected in the Publish Settings dialog box with your HTML settings, and produces an HTML page with your movie embedded.

Flash includes various templates, suitable for most users' needs, that eliminate the need to edit an HTML page with the Flash movie. For example, one template simply places a Flash movie on the generated HTML page so that users can view it through a Web browser if the plug-in is installed. Another template does the same thing except it first detects whether the plug-in has been installed, and if not, installs it.

You can easily use the same template, change the settings, and publish a new HTML page. If you're proficient in HTML, you can also create your own templates using any HTML editor. Creating a template is the same as creating a standard HTML page, except that you replace specific values pertaining to a Flash movie with variables that begin with a dollar ($) sign.

Flash HTML templates have these characteristics:

- A one-line title that appears on the Template pop-up menu
- A longer description that appears when you click the Info button
- Template variables beginning with $ that specify where parameters values should be substituted when Flash generates the output file

  Note: Use `\$` if you need to use a $ for another purpose in the document.

- HTML `OBJECT` and `EMBED` tags that follow the tag requirements of Microsoft Internet Explorer and Netscape Communicator/Navigator, respectively. To display a movie properly on an HTML page, you must follow these tag requirements. Internet Explorer opens a Flash movie using the `OBJECT` HTML tag; Netscape uses the `EMBED` tag. For more information, see "Using OBJECT and EMBED tags" on page 387.

Customizing HTML publishing templates

If you're familiar with HTML, you can modify HTML template variables to create an image map, a text report, or a URL report, or to insert your own values for some of the most common Flash `OBJECT` and `EMBED` parameters (for Internet Explorer and Netscape Communicator/Navigator, respectively).

Flash templates can include any HTML content for your application, or even code for special interpreters such as Cold Fusion, ASP, and the like.

To modify an HTML publishing template:

1. Using an HTML editor, open the Flash HTML template you want to change, located in the Macromedia Flash MX/HTML folder.
2. Edit the template as needed. To use the default values, leave the variables empty.

   For information on variables supported in Flash, see the following table.

   For information on creating an image map or a text or URL report, or to insert your own values for `OBJECT` and `EMBED` parameters, see the sections for those topics, following this procedure.
3. When you have finished editing the variables, save the template in the Macromedia Flash MX/HTML folder.
4. To apply the template settings to your Flash movie, choose File > Publish Settings, select the HTML panel, and select the template you modified.

   Flash changes only the template variables in the template selected in the Publish Settings dialog box.
5. Choose your remaining publishing settings, and click OK. For more information, see "Publishing Flash documents" on page 367.

The following table lists the template variables that Flash recognizes. For a definition of all the tags these variables work with, see “Editing Flash HTML settings” on page 386.

Parameter	Template Variable
Template title	$TT
Template description start	$DS
Template description finish	$DF
Flash (SWF) movie title	$T1
Width	$WI
Height	$HE
Movie	$MO
HTML alignment	$HA
Looping	$LO
Parameters for OBJECT	$PO
Parameters for EMBED	$PE
Play	$PL
Quality	$QU
Scale	$SC
Salign	$SA
Wmode	$WM
Devicefont	$DE
Bgcolor	$BG
Movie text (area to write movie text)	$MT
Movie URL (location of movie URLs)	$MU
Image width (unspecified image type)	$IW
Image height (unspecified image type)	$IH
Image file name (unspecified image type)	$IS
Image map name	$IU
Image map tag location	$IM
QuickTime width	$QW
QuickTime height	$QH
QuickTime file name	$QN
GIF width	$GW
GIF height	$GH
GIF file name	$GN
JPEG width	$JW
JPEG height	$JH
JPEG file name	$JN

Parameter	Template Variable
PNG width	$PW
PNG height	$PH
PNG file name	$PN

Creating an image map

Flash can generate an image map using any image and maintain the function of buttons that link to URLs, even if another image is substituted. On encountering the $IM template variable, Flash inserts the image map code in a template. The $IU variable identifies the name of the GIF, JPEG, or PNG file.

To create an image map:

1. In the Flash document, select the keyframe to be used for the image map and label it **#Map** in the Frame Properties inspector (choose Window > Properties if the Properties inspector is not visible). You can use any keyframe with buttons that have attached Get URL actions.

   If you don't create a frame label, Flash creates an image map using the buttons in the last frame of the movie. This option generates an embedded image map, not an embedded Flash movie.

2. To select the frame to be used for displaying the image map, do one of the following:

- For PNG or GIF files, label the frame to be used for display as **#Static**.
- For JPEG, during the publish operation, place the playhead on the frame to be used for display.

3. In an HTML editor, open the HTML template you'll modify. Flash stores HTML templates in the Macromedia Flash MX/HTML folder.
4. Save your template.
5. Choose File > Publish Settings, click the Format tab, and select a format for the image map—GIF, JPEG, or PNG.
6. Click OK to save your settings.

As an example, inserting the following code in a template:

```
$IM
<IMG SRC=$IS usemap=$IU WIDTH=$IW HEIGHT=$IH BORDER=0>
```

might produce this code in the HTML document created by the Publish command:

```
<MAP NAME="mymovie">
<AREA COORDS="130,116,214,182" HREF="http://www.macromedia.com">
</MAP>
<IMG SRC="mymovie.gif" usemap="#mymovie" WIDTH=550 HEIGHT=400 BORDER=0>
```

Creating a text report

The $MT template variable causes Flash to insert all the text from the current Flash movie as a comment in the HTML code. This is useful for indexing the content of a movie and making it visible to search engines.

Creating a URL report

The $MU template variable makes Flash generate a list of the URLs referred to by actions in the current movie and insert it at the current location as a comment. This enables link verification tools to see and verify the links in the movie.

Using shorthand template variables

The $PO (for OBJECT tags) and $PE (for EMBED tags) template variables are useful shorthand elements. Both variables cause Flash to insert into a template any nondefault values for some of the most common Flash OBJECT and EMBED parameters, including PLAY ($PL), QUALITY ($QU), SCALE ($SC), SALIGN ($SA), WMODE ($WM), DEVICEFONT ($DE), and BGCOLOR ($BG). See the sample template in the following section for an example of these variables.

Sample template

The Default.html template file in Flash, shown here as a sample, includes many of the commonly used template variables.

```
$TTFlash Only (Default)
$DS
Use an OBJECT and EMBED tag to display Flash.
$DF
<HEAD>
<TITLE>$TI</TITLE>
</HEAD>
<HTML><BODY bgcolor="$BG">

<!-- URL's used in the movie-->
$MU
<!-- text used in the movie-->
$MT

<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
 codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/
  swflash.cab#version=6,0,0,0"
 WIDTH=$WI HEIGHT=$HE id=$TI>
 $PO
<EMBED $PE WIDTH=$WI HEIGHT=$HE id=$TI
 TYPE="application/x-shockwave-flash"
 PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer">
</EMBED>
</OBJECT>

</BODY>
</HTML>
```

Editing Flash HTML settings

An HTML document is required to play a Flash movie in a Web browser and specify browser settings. If you are experienced with HTML, you can change or enter HTML parameters manually in an HTML editor, or create your own HTML files to control a Flash movie.

For information on having Flash create the HTML document automatically when you publish a movie, see "Publishing Flash documents" on page 367. For information on customizing HTML templates included in Flash, see "Customizing HTML publishing templates" on page 383.

Using OBJECT and EMBED tags

To display a Flash movie in a Web browser, an HTML document must use the OBJECT and EMBED tags with the proper parameters.

For OBJECT, four settings (HEIGHT, WIDTH, CLASSID, and CODEBASE) are attributes that appear within the OBJECT tag; all others are parameters that appear in separate, named PARAM tags. For example:

```
<OBJECT CLASSID="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" WIDTH="100"
HEIGHT="100" CODEBASE="http://active.macromedia.com/flash6/cabs/
swflash.cab#version=6,0,0,0">
<PARAM NAME="MOVIE" VALUE="moviename.swf">
<PARAM NAME="PLAY" VALUE="true">
<PARAM NAME="LOOP" VALUE="true">
<PARAM NAME="QUALITY" VALUE="high">
</OBJECT>
```

For the EMBED tag, all settings (such as HEIGHT, WIDTH, QUALITY, and LOOP) are attributes that appear between the angle brackets of the opening EMBED tag. For example:

```
<EMBED SRC="moviename.swf" WIDTH="100" HEIGHT="100" PLAY="true"
LOOP="true" QUALITY="high"
PLUGINSPAGE="http://www.macromedia.com/shockwave/download/
  index.cgi?P1_Prod_Version=ShockwaveFlash">
</EMBED>
```

To use both tags together, position the EMBED tag just before the closing OBJECT tag, as follows:

```
<OBJECT CLASSID="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" WIDTH="100"
HEIGHT="100" CODEBASE="http://active.macromedia.com/flash6/cabs/
swflash.cab#version=6,0,0,0">
<PARAM NAME="MOVIE" VALUE="moviename.swf">
<PARAM NAME="PLAY" VALUE="true">
<PARAM NAME="LOOP" VALUE="true">
<PARAM NAME="QUALITY" VALUE="high">

<EMBED SRC="moviename.swf" WIDTH="100" HEIGHT="100" PLAY="true"
LOOP="true" QUALITY="high"
PLUGINSPAGE="http://www.macromedia.com/shockwave/download/
  index.cgi?P1_Prod_Version=ShockwaveFlash">
</EMBED>


</OBJECT>
```

Note: If you use both the OBJECT and the EMBED tags, use identical values for each attribute or parameter to ensure consistent playback across browsers. The parameter swflash.cab#version=6,0,0,0 is optional, and you can omit it if you don't want to check for version number.

The following tag attributes and parameters describe the HTML code created by the Publish command. You can refer to this list as you write your own HTML to insert in Flash movies. Unless noted, all items apply to both OBJECT and EMBED tags. Optional entries are noted. When customizing a template, you can substitute a template variable listed here for the value. See "Customizing HTML publishing templates" on page 383.

SRC

Value

movieName.swf

Template variable: $MO

Description

Specifies the name of the movie to be loaded. Applies to EMBED only.

MOVIE

Value

movieName.swf

Template variable: $MO

Description

Specifies the name of the movie to be loaded. Applies to OBJECT only.

CLASSID

Value

clsid:D27CDB6E-AE6D-11cf-96B8-444553540000

Description

Identifies the ActiveX control for the browser. The value must be entered exactly as shown. Applies to OBJECT only.

WIDTH

Value

n or *n%*

Template variable: $WI

Description

Specifies the width of the movie in either pixels or percentage of the browser window.

HEIGHT

Value

n or *n%*

Template variable: $HE

Description

Specifies the height of the movie in either pixels or percentage of the browser window.

Because Flash movies are scalable, their quality won't degrade at different sizes if the aspect ratio is maintained. (For example, the following sizes all have a 4:3 aspect ratio: 640 by 480 pixels, 320 by 240 pixels, and 240 by 180 pixels.)

CODEBASE

Value

`http://active.macromedia.com/flash6/cabs/swflash.cab#version=6,0,0,0"`

Description

Identifies the location of the Flash Player ActiveX control so that the browser can automatically download it if it is not already installed. The value must be entered exactly as shown. Applies to `OBJECT` only.

PLUGINSPAGE

Value

```
http://www.macromedia.com/shockwave/download/
  index.cgi?P1_Prod_Version=ShockwaveFlash
```

Description

Identifies the location of the Flash Player plug-in so that the user can download it if it is not already installed. The value must be entered exactly as shown. Applies to `EMBED` only.

SWLIVECONNECT

Value

`true` | `false`

Description

(Optional) Specifies whether the browser should start Java when loading the Flash Player for the first time. The default value is `false` if this attribute is omitted. If you use JavaScript and Flash on the same page, Java must be running for the FSCommand to work. However, if you are using JavaScript only for browser detection or another purpose unrelated to FSCommand actions, you can prevent Java from starting by setting `SWLIVECONNECT` to `false`. You can also force Java to start when you are not using JavaScript with Flash by explicitly setting `SWLIVECONNECT` to `true`. Starting Java substantially increases the time it takes to start a movie; set this tag to `true` only when necessary. Applies to `EMBED` only.

Use the Exec FSCommand actions to start Java from a stand-alone projector file. See "Sending messages to and from the Flash Player" under Help > Using Flash.

PLAY

Value

`true` | `false`

Template variable: `$PL`

Description

(Optional) Specifies whether the movie begins playing immediately on loading in the browser. If your Flash movie is interactive, you may want to let the user initiate play by clicking a button or performing some other task. In this case, set the `PLAY` attribute to `false` to prevent the movie from starting automatically. The default value is `true` if this attribute is omitted.

LOOP

Value

`true` | `false`

Template variable: `$LO`

Description

(Optional) Specifies whether the movie repeats indefinitely or stops when it reaches the last frame. The default value is `true` if this attribute is omitted.

QUALITY

Value

`low` | `high` | `autolow` | `autohigh` | `best`

Template variable: `$QU`

Description

(Optional) Specifies the level of anti-aliasing to be used during playback of your movie. Because anti-aliasing requires a faster processor to smooth each frame of the movie before it is rendered on the viewer's screen, choose a value based on whether speed or appearance is your top priority:

- Low favors playback speed over appearance and never uses anti-aliasing.
- Autolow emphasizes speed at first but improves appearance whenever possible. Playback begins with anti-aliasing turned off. If the Flash Player detects that the processor can handle it, anti-aliasing is turned on.
- Autohigh emphasizes playback speed and appearance equally at first but sacrifices appearance for playback speed if necessary. Playback begins with anti-aliasing turned on. If the actual frame rate drops below the specified frame rate, anti-aliasing is turned off to improve playback speed. Use this setting to emulate the View > Antialias setting in Flash.
- Medium applies some anti-aliasing and does not smooth bitmaps. It produces a better quality than the Low setting, but lower quality than the High setting.
- High favors appearance over playback speed and always applies anti-aliasing. If the movie does not contain animation, bitmaps are smoothed; if the movie has animation, bitmaps are not smoothed.
- Best provides the best display quality and does not consider playback speed. All output is anti-aliased and all bitmaps are smoothed.

The default value for Quality is `high` if this attribute is omitted.

BGCOLOR

Value

#RRGGBB (hexadecimal RGB value)

Template variable: `$BG`

Description

(Optional) Specifies the background color of the movie. Use this attribute to override the background color setting specified in the Flash movie. This attribute does not affect the background color of the HTML page.

SCALE

Value

showall | noborder | exactfit

Template variable: $SC

Description

(Optional) Defines how the movie is placed within the browser window when WIDTH and HEIGHT values are percentages.

- Default (Show all) makes the entire movie visible in the specified area without distortion, while maintaining the original aspect ratio of the movie. Borders may appear on two sides of the movie.
- No Border scales the movie to fill the specified area, without distortion but possibly with some cropping, while maintaining the original aspect ratio of the movie.
- Exact Fit makes the entire movie visible in the specified area without trying to preserve the original aspect ratio. Distortion may occur.

The default value is showall if this attribute is omitted (and WIDTH and HEIGHT values are percentages).

ALIGN

Value

L | R | T | B

Template variable: $HA

Description

Specifies the ALIGN attribute for the OBJECT, EMBED, and IMG tags and determines how the Flash movie window is positioned within the browser window.

- Default centers the movie in the browser window and crops edges if the browser window is smaller than the movie.
- Left, Right, Top, and Bottom align the movie along the corresponding edge of the browser window and crop the remaining three sides as needed.

SALIGN

Value

`L | R | T | B | TL | TR | BL | BR`

Template variable: `$SA`

Description

(Optional) Specifies where a scaled Flash movie is positioned within the area defined by the `WIDTH` and `HEIGHT` settings. See "SCALE" on page 391 for more information about these conditions.

- `L`, `R`, `T`, and `B` align the movie along the left, right, top or bottom edge, respectively, of the browser window and crop the remaining three sides as needed.
- `TL` and `TR` align the movie to the top left and top right corner, respectively, of the browser window and crop the bottom and remaining right or left side as needed.
- `BL` and `BR` align the movie to the bottom left and bottom right corner, respectively, of the browser window and crop the top and remaining right or left side as needed.

If this attribute is omitted, the movie is centered in the browser window. Cropping may occur or borders may appear on any side, as needed.

BASE

Value

`base directory or URL`

Description

(Optional) Specifies the base directory or URL used to resolve all relative path statements in the Flash movie. This attribute is helpful when your SWF files are kept in a different directory from your other files.

MENU

Value

`true | false`

Template variable: `$ME`

Description

(Optional) Specifies what type of menu is displayed when the viewer right-clicks (Windows) or Command-clicks (Macintosh) the movie area in the browser.

- `true` displays the full menu, allowing the user a variety of options to enhance or control playback.
- `false` displays a menu that contains only the About Macromedia Flash Player 6 option and the Settings option.

The default value is `true` if this attribute is omitted.

WMODE

Value

`Window` | `Opaque` | `Transparent`

Template variable: `$WM`

Description

(Optional) Lets you take advantage of the transparent movie, absolute positioning, and layering capabilities available in Internet Explorer 4.0. This tag works only in Windows with the Flash Player ActiveX control.

- `Window` plays the movie in its own rectangular window on a Web page.
- `Opaque` makes the movie hide everything behind it on the page.
- `Transparent` makes the background of the HTML page show through all the transparent portions of the movie, and may slow animation performance.

The default value is `Window` if this attribute is omitted. Applies to `OBJECT` only.

Previewing the publishing format and settings

To preview your Flash movie with the publishing format and settings you've selected, you can use the Publish Preview command. This command exports the file and opens the preview in the default browser. If you preview a QuickTime movie, Publish Preview launches the QuickTime Movie Player. If you preview a projector, Flash launches the projector.

To preview a file with the Publish Preview command:

1. Define the file's export options using the Publish Settings command; see "Publishing Flash documents" on page 367.
2. Choose File > Publish Preview, and from the submenu choose the file format you want to preview.

   Using the current Publish Settings values, Flash creates a file of the specified type in the same location as the Flash document (FLA). This file remains in this location until you overwrite or delete it.

Using the stand-alone player

The stand-alone player plays Flash movies exactly as they appear in a Web browser or an ActiveX host application. The stand-alone player is installed along with Flash (named stand-alone Flash Player in Windows and Flash Player on the Macintosh). When you double-click a Flash movie, the operating system starts the stand-alone player, which in turn runs the movie. You can use the stand-alone player to make movies viewable for users who aren't using a Web browser or an ActiveX host application.

You can control movies in the stand-alone player using menu commands and the FScommand action. For example, to make the stand-alone player take over the whole screen, you assign the FScommand action to a frame or button and then select the `Fullscreen` command with the `True` argument. For more information, see "Using actions and methods to control movie clips" under Help > Using Flash.

You can also print movie frames using the stand-alone player context menu. See "Printing from the Flash Player context menu" under Help > Using Flash.

To control movies from the stand-alone player, choose from the following options:

- Open a new or existing file by choosing File > New or File > Open.
- Change your view of the movie by choosing View > Magnification, and from the submenu choose Show All, Zoom In, Zoom Out, or 100%.
- Control movie playback by choosing Control > Play, Rewind, or Loop.

Configuring a Web server for Flash

When your files are accessed from a Web server, the server must properly identify them as Flash movies in order to display them. If the MIME type is missing or not properly delivered by the server, the browser may display error messages or a blank window with a puzzle piece icon.

Your server may already be configured properly. To test server configuration, see TechNote #12696 on the Macromedia Flash Support Center, www.macromedia.com.If your server is not properly configured, you (or your server's administrator) must add the Flash movie MIME types to the server's configuration files and associate the following MIME types with the SWF file extensions:

- MIME type application/x-shockwave-flash has the .swf file extension.
- MIME type application/futuresplash has the .spl file extension.

If you are administering your own server, consult your server software documentation for instructions on adding or configuring MIME types. If you are not administering your own server, contact your Internet service provider, Web master, or server administrator to add the MIME type information.

If your site is on a Macintosh server, you must also set the following parameters: Action: Binary; Type: SWFL; and Creator: SWF2.

Screening traffic to your Web site

When publishing Flash content on the Web, you can configure a Web server to make it easier to play Flash movies, using a script-based detector to determine whether a user has the Flash Player plug-in or ActiveX control installed. Called the Macromedia Flash Dispatcher, this detector is included in the Macromedia Flash (SWF) Deployment Kit, in the Macromedia Flash MX/ Goodies folder.

The Dispatcher is a combination of JavaScript, VBScript, and Flash movie that screens incoming traffic to your Web site. The Dispatcher detects whether the user's Web browser has the Flash Player installed, and if so, what version. You can configure the Dispatcher to load a document with Flash content, load alternate content, or oversee the updating or installation of the player.

CHAPTER 21
Exporting

The Export Movie command in Macromedia Flash MX lets you create content that can be edited in other applications and export a movie directly into a single format. For example, you can export an entire movie as a Flash movie; as a series of bitmap images; as a single frame or image file; and as moving and still images in various formats, including GIF, JPEG, PNG, BMP, PICT, QuickTime, or AVI.

When you export a Flash movie in Flash MX format, text is encoded in Unicode format, providing support for international character sets, including double-byte fonts. Likewise, Flash Player 6 supports Unicode encoding. See "Unicode text encoding in Flash movies" on page 366.

If you have Macromedia Dreamweaver, you can add a Flash movie to your Web site easily. Dreamweaver generates all the needed HTML code. You can launch and edit Flash from within Dreamweaver to update the Flash movie. See "Updating Flash movies for Dreamweaver UltraDev" on page 401.

Exporting movies and images

To prepare Flash content for use in other applications or to export the contents of the current Flash movie in a particular file format, you use the Export Movie and Export Image commands. The Export commands do not store export settings separately with each file, as does the Publish command. (Use Publish to create all the files you need to put a Flash movie on the Web. See "Publishing Flash documents" on page 367.)

The Export Movie command lets you export a Flash movie to a still-image format and create a numbered image file for every frame in the movie. You can also use Export Movie to export the sound in a movie to a WAV file (Windows only).

To export the contents of the current frame or the currently selected image to one of the still-image formats, or to a single-frame Flash Player movie, you use the Export Image command.

- When you export a Flash image as a vector-graphic file (in Adobe Illustrator format), you preserve its vector information. You can edit these files in other vector-based drawing programs, but you can't import these images into most page-layout and word-processing programs.
- When you save a Flash image as a bitmap GIF, JPEG, PICT (Macintosh), or BMP (Windows) file, the image loses its vector information and is saved with pixel information only. You can edit Flash images exported as bitmaps in image editors such as Adobe Photoshop, but you can no longer edit them in vector-based drawing programs.

To export a movie or image:

1. Open the Flash movie you want to export, or if you are exporting an image from the movie, select the frame or image in the current movie that you want to export.
2. Choose File > Export Movie or File > Export Image.
3. Enter a name for the output file.
4. Choose the file format from the Format pop-up menu.
5. Click Save.

   If the format you selected requires more information, an Export dialog box appears.
6. Set the export options for the format you selected. See the following section.
7. Click OK, then click Save.

Note: Before exporting a Flash Video format (FLV) file from Mac OS 9.1, you must increase your memory allocation to at least 37,000 Kb.

About export file formats

You can export Flash movies and images in more than a dozen different formats, as shown in the table that follows. Movies are exported as sequences, and images as individual files. PNG is the only cross-platform bitmap format that supports transparency (as an alpha channel). Some nonbitmap export formats do not support alpha (transparency) effects or mask layers.

Note: You can export video clips in FLV format. See "Importing video" under Help › Using Flash.

For more information on a specific file format, see the sections that follow.

File type	Extension	Windows	Macintosh
"Adobe Illustrator" on page 397	.ai	✔	✔
Animated GIF, GIF Sequence, and GIF Image	.gif	✔	✔
"Bitmap (BMP)" on page 397	.bmp	✔	
DXF Sequence and AutoCAD DXF Image	.dxf	✔	✔
Enhanced Metafile	.emf	✔	
EPS (Version 6.0 or earlier)	.eps	✔	✔
Flash movie	.swf	✔	✔
FutureSplash Player	.spl	✔	✔
"JPEG Sequence and JPEG Image" on page 398	.jpg	✔	✔
PICT Sequence (Macintosh)	.pct		✔
"PNG Sequence and PNG Image" on page 399	.png	✔	✔
"Choosing publish settings for QuickTime 4 movies" on page 381	.mov	✔	✔
"QuickTime Video (Macintosh)" on page 400	.mov		✔
"WAV audio (Windows)" on page 400	.wav	✔	

File type	Extension	Windows	Macintosh
"Windows AVI (Windows)" on page 400	.avi	✔	
"Windows Metafile" on page 401	.wmf	✔	

Adobe Illustrator

The Adobe Illustrator format is ideal for exchanging drawings between Flash and other drawing applications such as FreeHand. This format supports very accurate conversion of curve, line style, and fill information. Flash supports import and export of the Adobe Illustrator 88, 3.0, 5.0, 6.0, and 8.0 formats. (See "Importing Adobe Illustrator files" under Help > Using Flash.) Flash does not support the Photoshop EPS format or EPS files generated using the Print command.

Versions of the Adobe Illustrator format before 5 do not support gradient fills, and only version 6 supports bitmaps.

The Export Adobe Illustrator dialog box lets you choose the Adobe Illustrator version—88, 3.0, 5.0, 6.0, or 8.0

To make exported Flash files compatible with Adobe Illustrator 8.0 or later, use the Macromedia Flashwriter for Adobe Illustrator plug-in, included in the Flash product.

Animated GIF, GIF Sequence, and GIF Image

This option lets you export files in the GIF format. The settings are the same as those available on the GIF tab in the Publish Settings dialog box, with the following exceptions:

Resolution is set in dots per inch (dpi). You can enter a resolution or click Match Screen to use the screen resolution.

Include lets you choose to export the minimum image area or specify the full document size.

Colors lets you set the number of colors that can be used to create the exported image—black-and-white; 4-, 6-, 16-, 32-, 64-, 128-, or 256- colors; or Standard Color (the standard 216-color, browser-safe palette).

You can also choose to interlace, smooth, make transparent, or dither solid colors. For information on these options, see "Choosing publish settings for GIF files" on page 374.

Animation is available for the Animated GIF export format only and lets you enter the number of repetitions, where 0 repeats endlessly.

Bitmap (BMP)

This format lets you create bitmap images for use in other applications. The Bitmap Export Options dialog box has these options:

Dimensions sets the size of the exported bitmap image in pixels. Flash ensures that the size you specify always has the same aspect ratio as your original image.

Resolution sets the resolution of the exported bitmap image in dots per inch (dpi) and has Flash automatically calculate width and height based on the size of your drawing. To set the resolution to match your monitor, select Match Screen.

Color Depth specifies the bit depth of the image. Some Windows applications do not support the newer 32-bit depth for bitmap images; if you have problems using a 32-bit format, use the older 24-bit format.

Smooth applies anti-aliasing to the exported bitmap. Anti-aliasing produces a higher-quality bitmap image, but it may create a halo of gray pixels around an image placed on a colored background. Deselect this option if a halo appears.

DXF Sequence and AutoCAD DXF Image

This 3D format lets you export elements of your movie as AutoCAD DXF release 10 files, so that they can be brought into a DXF-compatible application for additional editing.

This format has no definable export options.

Enhanced Metafile (Windows)

Enhanced Metafile Format (EMF) is a graphics format available in Windows 95 and Windows NT that saves both vector and bitmap information. EMF supports the curves used in Flash drawings better than the older Windows Metafile format. However, many applications do not yet support this newer graphics format.

This format has no definable export options.

EPS 3.0 with Preview

You can export the current frame as an EPS 3.0 file for placement in another application, such as a page layout application. An EPS (encapsulated PostScript) file can be printed by a PostScript printer. As an option, you can include a bitmap preview with the exported EPS file for applications that can import and print the EPS files (such as Microsoft Word and Adobe PageMaker), but that can't display them onscreen.

Flash has no definable exporting options for EPS files.

Flash movie

You can export the entire document as a Flash movie, to place the movie in another application, such as Dreamweaver. You can select the same options for exporting a Flash movie as you can for publishing a Flash movie. See "Choosing publish settings Flash movie format" on page 370.

FutureSplash Player

This file format was used by Flash prior to its acquisition by Macromedia. The export options match the Flash publish settings options. See "Choosing publish settings Flash movie format" on page 370.

JPEG Sequence and JPEG Image

The JPEG export options match the JPEG Publish Settings options with one exception: the Match Screen export option makes the exported image match the size of the movie as it appears on your screen. The Match Movie publishing option makes the JPEG image the same size as the movie and maintains the aspect ratio of the original image.

For more information, see "Choosing publish settings for JPEG files" on page 377.

PICT (Macintosh)

PICT is the standard graphics format on the Macintosh and can contain bitmap or vector information. Use the Export PICT dialog box to set the following options:

Dimensions sets the size of the exported bitmap image specified in pixels. Flash ensures that the size you specify always has the same aspect ratio as your original image.

Resolution sets the resolution in dots per inch (dpi) and has Flash automatically calculate width and height based on the size of your drawing. To set the resolution to match your monitor, select Match Screen. Bitmap PICT images usually look best onscreen with 72-dpi resolution.

Include sets the portion of the document to be exported, either Minimum Image Area or Full Document Size.

Color Depth designates whether the PICT file is object-based or bitmap. Object-based images generally look better when printed, and scaling doesn't affect their appearance. Bitmap PICT images normally look best displayed onscreen and can be manipulated in applications such as Adobe Photoshop. You can also choose a variety of color depths with bitmap PICT files.

Include Postscript is available only for an object-based PICT file to include information that optimizes printing on a PostScript printer. This information makes the file larger and may not be recognized by all applications.

Smooth Bitmap is available only for a bitmap PICT. This option applies anti-aliasing to smooth jagged edges of a bitmap image.

PNG Sequence and PNG Image

These export settings are similar to the PNG Publish Settings options ("Choosing publish settings for PNG files" on page 378), with the following exceptions:

Dimensions sets the size of the exported bitmap image to the number of pixels you enter in the Width and Height fields.

Resolution lets you enter a resolution in dots per inch (dpi). To use the screen resolution and maintain the aspect ratio of your original image, select Match Screen.

Colors is the same as the Bit Depth option in the PNG Publish Settings tab and sets the number of bits per pixel to use in creating the image. For a 256-color image, choose 8-bit; for thousands of colors, choose 24-bit; for thousands of colors with transparency (32 bits) choose 24-bit with Alpha. The higher the bit depth, the larger the file.

Include lets you choose to export the minimum image area or specify the full document size.

Filter options match those in the PNG Publish Settings tab.

When exporting a PNG sequence or PNG image, you can also apply other options in the PNG Publish Settings, such as Interlace, Smooth, and Dither Solid Colors.

QuickTime

The QuickTime export option creates a movie with a Flash track in the QuickTime 4 format. This export format lets you combine the interactive features of Flash MX with the multimedia and video features of QuickTime in a single QuickTime 4 movie, which can be viewed by anyone with the QuickTime 4 plug-in.

If you have imported a video clip (in any format) into a movie as an embedded file, you can publish the movie as a QuickTime movie. If you have imported a video clip in QuickTime format into a movie as a linked file, you can also publish the movie as a QuickTime movie. For information on importing video clips, see "Importing video" under Help > Using Flash.

When you export a Flash movie as a QuickTime movie, all layers in the Flash project are exported as a single Flash track, unless the Flash movie contains an imported QuickTime movie. The imported QuickTime movie remains in QuickTime format in the exported movie.

These export options are identical to QuickTime publish options. See "Choosing publish settings for QuickTime 4 movies" on page 381.

QuickTime Video (Macintosh)

The QuickTime Video format converts the Flash movie into a sequence of bitmaps embedded in the file's video track. The Flash content is exported as a bitmap image without any interactivity. This format is useful for editing Flash movies in a video-editing application.

The Export QuickTime Video dialog box contains the following options:

Dimensions specifies a width and height in pixels for the frames of a QuickTime movie. By default, you can specify only the width or the height, and the other dimension is automatically set to maintain the aspect ratio of your original movie. To set both the width and the height, deselect Maintain Aspect Ratio.

Format selects a color depth. Options are black-and-white; 4-, 8-, 16-, or 24-bit color; and 32-bit color with alpha (transparency).

Smooth applies anti-aliasing to the exported QuickTime movie. Anti-aliasing produces a higher-quality bitmap image, but it may cause a halo of gray pixels to appear around images when placed over a colored background. Deselect the option if a halo appears.

Compressor selects a standard QuickTime compressor. See your QuickTime documentation for more information.

Quality controls the amount of compression applied to your movie. The effect depends on the compressor selected.

Sound Format sets the export rate for sounds in the movie. Higher rates yield better fidelity and larger files. Lower rates save space.

WAV audio (Windows)

The WAV Export Movie option exports only the sound file of the current movie to a single WAV file. You can specify the sound format of the new file.

Choose Sound Format to determine the sampling frequency, bit rate, and stereo or mono setting of the exported sound. Select Ignore Event Sounds to exclude events sounds from the exported file.

Windows AVI (Windows)

This format exports a movie as a Windows video, but discards any interactivity. The standard Windows movie format, Windows AVI is a good format for opening a Flash animation in a video-editing application. Because AVI is a bitmap-based format, movies that contain long or high-resolution animations can quickly become very large.

The Export Windows AVI dialog box has the following options:

Dimensions specifies a width and height, in pixels, for the frames of an AVI movie. Specify only the width or the height; the other dimension is automatically set to maintain the aspect ratio of your original movie. Deselect Maintain Aspect Ratio to set both the width and the height.

Video Format selects a color depth. Many applications do not yet support the Windows 32-bit image format. If you have problems using this format, use the older 24-bit format.

Compress Video displays a dialog box for choosing standard AVI compression options.

Smooth applies anti-aliasing to the exported AVI movie. Anti-aliasing produces a higher-quality bitmap image, but it may cause a halo of gray pixels to appear around images when placed over a colored background. Deselect the option if a halo appears.

Sound Format lets you set the sample rate and size of the sound track, and whether it will be exported in mono or stereo. The smaller the sample rate and size, the smaller the exported file, with a possible trade-off in sound quality. For more information on exporting sound to the AVI format, see "Compressing sounds for export" under Help > Using Flash.

Windows Metafile

Windows Metafile format is the standard Windows graphics format and is supported by most Windows applications. This format yields good results for importing and exporting files. It has no definable export options. See "Enhanced Metafile (Windows)" on page 398.

Updating Flash movies for Dreamweaver UltraDev

If you have Dreamweaver UltraDev installed on your system, you can export Flash movie files directly to a Dreamweaver UltraDev site. For more information on working with Dreamweaver UltraDev, see the Dreamweaver UltraDev documentation.

In Dreamweaver UltraDev, you can add the Flash movie to your page. With a single click, you can update the Flash document (FLA) for the export Flash movie in Flash and re-export the updated movie to UltraDev automatically.

To update a Flash movie for Dreamweaver UltraDev:

1. In Dreamweaver UltraDev, open the HTML page that contains the Flash movie.
2. Do one of the following:

- Select the Flash movie and click Edit in the Property inspector.
- In Design View, press Control (Windows) or Command (Macintosh) and double-click the Flash movie.
- In Design View, right-click (Windows) or Control-click (Macintosh) the Flash movie in Design View and choose Edit with Flash from the context menu.
- In the Site Window, right-click (Windows) or Control-click (Macintosh) the Flash movie in Design View and choose Open with Flash MX from the context menu.

The Flash application is launched on your system.

3. If the Flash document (FLA) for the exported movie does not open, a file locator dialog box will pop up. Navigate to the FLA in the Open File dialog box and click Open.

4. If the user has used the "Change Link Sitewide" feature in Dreamweaver UltraDev, a warning will be shown. Click OK to apply link changes to the Flash movie. Click Don't Warn Me Again to prevent the warning message from appearing when you update the Flash movie.
5. Update the Flash document (FLA) as needed in Flash.
6. To save the Flash document (FLA) and re-export the Flash movie to Dreamweaver, do one of the following to save:

- To update the file and close Flash, click the Done button above the upper left corner of the Stage.
- To update the file and keep Flash open, choose File > Update for Dreamweaver.

APPENDIX A
Keyboard shortcuts

The following sections contain tables listing the keyboard shortcuts for Macromedia Flash MX.

Navigation keys

The following table lists the keyboard shortcuts for navigating Flash MX.

Description	Mac	Windows
Move word left	Option+Left Arrow	Control+Left Arrow
Move word right	Option+Right Arrow	Control+Right Arrow
Move to start of line	Command+Left Arrow	Home
Move to end of line	Command+Right Arrow	End
Move to top of page	Option+Up Arrow	Page Up
Move to bottom of page	Option+Down Arrow	Page Down
Move to top of file	Command+Home	Control+Home
Move to bottom of file	Command+End	Control+End
Select word left	Shift+Option+Left Arrow	Shift+Control+Left Arrow
Select word right	Shift+Option+Right Arrow	Shift-Control+Right Arrow
Select to start of line	Shift+Command+Left Arrow	Shift+Home
Select to end of line	Shift+Command+Right Arrow	Shift+End
Select to start of page	Shift+Option+Up Arrow	Shift+Page Up
Select to end of page	Shift+Option+Down Arrow	Shift+Page Down
Select to start of file	Shift+Command+Up Arrow	Shift+Control+Home
Select to end of file	Shift+Command+Down Arrow	Shift+Control+End
Scroll line up	Control+Up Arrow	n/a
Scroll line down	Control+Down Arrow	n/a
Scroll page up	Page Up	n/a
Scroll page down	Page Down	n/a
Scroll to top of file	Home	n/a
Scroll to end of file	End	n/a

Action keys

The following table lists the keyboard shortcuts for performing specific actions in Flash MX.

Description	Mac	Windows
Balance Punctuation	Command+'	Control+'
Block Tab Left	Command+[	Shift+Tab
Block Tab Right	Command+]	Tab
Find Next	F3	F3
Replace	Command+H	Control+H
Go to Line...	Command+G	Control+G
Set Breakpoint	Command+Shift+B	Control+Shift+B
Remove Breakpoint	Command+Shift+B	Control+Shift+B
Remove all Bkpt.	Command+Shift+A	Control+Shift+A

Mouse actions

The following table lists the keyboard shortcuts for using the mouse to perform actions in Flash MX.

Description	Mac	Windows
select line	Triple-click	click in line# gutter
context menu	Control-click	Right mouse button

Menu items

The following table lists the keyboard shortcuts for performing specific context menu actions in Flash MX.

Description	Location	Mac Keys	Windows Keys
Undo	Context menu	Command+Z	Control+Z
Redo	Context menu	Command+Y	Control+Y
Cut	Context menu	Command+X	Control+X
Copy	Context menu	Command+C	Control+C
Paste	Context menu	Command+V	Control+V
Select All	Context menu	Command+A	Control+A
Set Breakpoint	Context menu	Command+Shift+B	Control+Shift+B
Remove Breakpoint	Context menu	Command+Shift+B	Control+Shift+B
Remove All Brkpt.	Context menu	Command+Shift+A	Control+Shift+A

APPENDIX B
Operator Precedence and Associativity

This table lists all of the ActionScript operators and their associativity, from highest to lowest precedence.

Operator	Description	Associativity
	Highest Precedence	
+	Unary plus	Right to left
-	Unary minus	Right to left
~	Bitwise one's complement	Right to left
!	Logical NOT	Right to left
not	Logical NOT (Flash 4 style)	Right to left
++	Post-increment	Left to right
--	Post-decrement	Left to right
( )	Function call	Left to right
[ ]	Array element	Left to right
.	Structure member	Left to right
++	Pre-increment	Right to left
--	Pre-decrement	Right to left
new	Allocate object	Right to left
delete	Deallocate object	Right to left
typeof	Type of object	Right to left
void	Returns undefined value	Right to left
*	Multiply	Left to right
/	Divide	Left to right
%	Modulo	Left to right
+	Add	Left to right
add	String concatenation (formerly &)	Left to right
-	Subtract	Left to right
<<	Bitwise left shift	Left to right
>>	Bitwise right shift	Left to right

Operator	Description	Associativity
>>>	Bitwise right shift (unsigned)	Left to right
<	Less than	Left to right
<=	Less than or equal to	Left to right
>	Greater than	Left to right
>=	Greater than or equal to	Left to right
lt	Less than (string version)	Left to right
le	Less than or equal to (string version)	Left to right
gt	Greater than (string version)	Left to right
ge	Greater than or equal to (string version)	Left to right
==	Equal	Left to right
!=	Not equal	Left to right
eq	Equal (string version)	Left to right
ne	Not equal (string version)	Left to right
&	Bitwise AND	Left to right
^	Bitwise XOR	Left to right
|	Bitwise OR	Left to right
&&	Logical AND	Left to right
and	Logical AND (Flash 4)	Left to right
||	Logical OR	Left to right
or	Logical OR (Flash 4)	Left to right
?:	Conditional	Right to left
=	Assignment	Right to left
*=, /=, %o=, +=, -=, &=, |=, ^=, <<=, >>=, >>>=	Compound assignment	Right to left
,	Multiple evaluation	Left to right
	Lowest Precedence	

APPENDIX C
Keyboard Keys and Key Code Values

The following tables list all of the keys on a standard keyboard and the corresponding ASCII key code values that are used to identify the keys in ActionScript. For more information, see the description of the Key object in the online ActionScript Dictionary in the Help menu.

Letters A to Z and standard numbers 0 to 9

The following table lists the keys on a standard keyboard for the letters A to Z and the numbers 0 to 9, with the corresponding ASCII key code values that are used to identify the keys in ActionScript.

Letter or number key	Key code
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84

Letter or number key	Key code
U	85
V	86
W	87
X	88
Y	89
Z	90
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

Keys on the numeric keypad

The following table lists the keys on a numeric keypad, with the corresponding ASCII key code values that are used to identify the keys in ActionScript.

Numeric keypad key	Key code
Numbpad 0	96
Numbpad1	97
Numbpad 2	98
Numbpad 3	99
Numbpad 4	100
Numbpad 5	101
Numbpad 6	102
Numbpad 7	103
Numbpad 8	104
Numbpad 9	105
Multiply	106
Add	107
Enter	108
Subtract	109

Numeric keypad key	Key code
Decimal	110
Divide	111

Function keys

The following table lists the function keys on a standard keyboard, with the corresponding ASCII key code values that are used to identify the keys in ActionScript.

Function key	Key code
F1	112
F2	113
F3	114
F4	115
F5	116
F6	117
F7	118
F8	119
F9	120
F10	121
F11	122
F12	123
F13	124
F14	125
F15	126

Other keys

The following table lists keys on a standard keyboard other than letters or numbers, with the corresponding ASCII key code values that are used to identify the keys in ActionScript.

Key	Key code
Backspace	8
Tab	9
Clear	12
Enter	13
Shift	16
Control	17
Alt	18
Caps Lock	20
Esc	27
Spacebar	32
Page Up	33
Page Down	34
End	35
Home	36
Left Arrow	37
Up Arrow	38
Right Arrow	39
Down Arrow	40
Insert	45
Delete	46
Help	47
Num Lock	144
; :	186
= +	187
- _	189
/ ?	191
‘ ~	192
[{	219
\ |	220
] }	221
" ‘	222

APPENDIX D
Error Messages

The following table contains a list of error messages returned by the Flash compiler. An explanation of each message is provided to aid you in troubleshooting your movie files.

Error message	Description
Property ‹property› does not exist	A property that does not exist was encountered. For example, `x = _green` is invalid, because there is no `_green` property.
Operator ‹operator› must be followed by an operand	An operator without an operand was encountered. For example, `x = 1 +` requires an operand after the + operator.
Syntax error	This message is issued whenever a nonspecific syntax error is encountered.
Expected a field name after '.' operator	You must specify a valid field name when using the `object.field` syntax.

‹token› Expected	An invalid or unexpected token was encountered. For example, in the syntax below, the token `foo` is not valid. The expected token is `while`.
	`do {`
	`  trace (i)`
	`} foo (i < 100)`

Initializer list must be terminated by ‹terminator›	An object or array initializer list is missing the closing] or }.
Identifier expected	An unexpected token was encountered in place of an identifier. In the example below, 3 is not a valid identifier. `var 3 = 4;`
The JavaScript '‹construct›' construct is not supported	A JavaScript construct that is not supported by ActionScript was encountered. This message appears if any of the following JavaScript constructs are used: `void`, `try`, `catch`, or `throw`.
Left side of assignment operator must be variable or property	An assignment operator was used, but the left side of the assignment was not a legal variable or property.
Statement block must be terminated by '}'	A group of statements was declared within curly braces, but the closing brace is missing.
Event expected	An `on(MouseEvent)` handler was declared, but no event was specified, or an unexpected token was encountered where an event should appear.
Invalid mouse event specified	The script contains an invalid mouse event in a handler. For a list of valid mouse events, see the `On(MouseEvent)` entry in the online ActionScript Dictionary.
Key code identifier expected	You need to specify a key code. See Appendix B for a list of key codes.

Error message	Description
Invalid key code	The specified key code does not exist.
Trailing garbage found	The script or expression parsed correctly but contained additional trailing characters that could not be parsed.
Function name expected	The name specified for this function is not a valid function name.
Parameter name expected	A parameter (argument) name was expected in a function declaration, but an unexpected token was encountered.
'else' encountered without matching 'if'	An `else` statement was encountered, but no `if` statement appeared before it. You can use `else` only in conjunction with an `if` statement.
Scene name must be a quoted string	The scene argument of a `gotoAndPlay`, `gotoAndStop`, or `ifFrameLoaded` action was of the wrong type. The scene argument must be a string constant.
Internal error	An internal error occurred in the ActionScript compiler. Please send the FLA file that generated this error to Macromedia, with detailed instructions on how to reproduce the message.
Hexadecimal digits expected after 0x	The sequence `0x` was encountered, but the sequence was not followed by valid hexadecimal digits.
Error opening include file:file not found	There was an error opening a file included with the `include` directive. The file was not present.
Malformed #include directive	An `include` directive was not written correctly. An `include` directive must use the following syntax: `#include "somefile.as"`
Multi-line comment was not terminated	A multi-line comment started with /* is missing the closing */ tag.
String literal was not properly terminated	A string literal started with an opening quotation mark (single or double) is missing the closing quotation mark.
Wrong number of parameters; <function> requires between <low> and <high>	A function was called, but an unexpected number of parameters were encountered.
Property name expected in getProperty	The `getProperty` function was called, but the second parameter was not the name of a movie clip property.
Parameter <parameter> cannot be declared multiple times	A parameter name appeared multiple times in the parameter list of a function declaration. All parameter names must be unique.
Variable <variable> cannot be declared multiple times	A variable name appeared multiple times in a `var` statement. All variable names in a single `var` statement must be unique.
'on' handlers may not nest within other 'on' handlers	An on handler was declared inside another `on` handler. All `on` handlers must appear at the top level of an action list.
'onClipEvent' handlers may not nest within other 'onClipEvent' handlers	An onClipEvent handler was declared inside another onClipEvent handler. All onClipEvent handlers must appear at the top level of an action list.
Statement must appear within on handler (Message appears for Flash 5 format)	In the actions for a button instance, a statement was declared without a surrounding `on` block. All actions for a button instance must appear inside an `on` block.
Statement must appear within onClipEvent handler (Message appears for Flash 5 format)	In the actions for a movie clip instance, a statement was declared without a surrounding `onClipEvent` block. All actions for a movie clip instance must appear inside an `onClipEvent` block.

Error message	Description
Statement must appear within on or onClipEvent handler (Message appears for Flash MX format)	In the actions for a movie clip instance, a statement was declared without a surrounding on or `onClipEvent` block. All actions for a movie clip instance must appear inside an on or `onClipEvent` block.
Mouse events are permitted only for button instances (Message appears for Flash 5 format)	A button event handler was declared in a frame action list or a movie clip instance's action list. Button events are permitted only in the action lists of button instances.
Clip events are permitted only for movie clip instances	A clip event handler was declared in a frame action list or a button instance's action list. Clip events are permitted only in the action lists of movie clip instances.
Function declaration not permitted here	Cannot use a named function when declaring it in an assignment.
Duplicate in event list	An on handler was specified with a duplicate event.
Invalid movie clip event specified	An `onClipEvent` handler can only accept `load`, `enterForm`, `unload`, `mouseMove`, `mouseDown`, `mouseUp`, `keyDown`, `keyUp`, or `data` events.
Case-insensitive identifier <identifier> will obscure built-in object <object name>	Since ActionScript is case-insensitive, a case-insensitive identifier would obscure a built-in object.
Case statements can only be used inside a switch statement	Case statements must be used inside of switch statements.
Case statements must be terminated with a ':'	Every case statement within a switch statement must end with a colon.

INDEX

B

C

D

E

F

G

H

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z