CONTENTS

GETTING STARTED

About Classroom in a Book ........................................ 1
Prerequisites ............................................................. 1
Installing Flash ......................................................... 2
Optimizing performance .............................................. 2
Copying the lesson files .............................................. 2
How to use these lessons ............................................. 3
Standard elements in the book ............................... 3
Checking for updates .................................................. 4
Flash Player version .................................................. 5
Additional resources ................................................... 5
Adobe certification ..................................................... 7

INTRODUCTION TO ACTIONSCRIPT 3.0

A brief history of Flash and ActionScript ................... 9
ActionScript 3.0 for new programmers ....................... 10
For users of ActionScript 1.0 and 2.0 ......................... 11
Formats for playing back Flash and ActionScript 3.0 files ... 12
Flash CS5, Flash Builder 4, and Flex ......................... 12
ActionScript in the Flash Timeline vs. external
ActionScript files ..................................................... 13
That’s enough background for now... ................... 13

1 USING CODE SNIPPETS AND NAVIGATING THE FLASH TIMELINE

Lesson overview ....................................................... 14
Getting started ........................................................ 16
Adding ActionScript using code snippets .................. 16
Placing code on the Flash Timeline ......................... 20
Some suggestions to try on your own ..................... 28
2 WORKING WITH EVENTS AND FUNCTIONS
Lesson overview ........................................ 30
Working with event-handling functions ............ 32
Using code snippets to create navigation .......... 34
Creating event listeners ............................ 40
Some suggestions to try on your own ............. 46

3 CREATING ANIMATION WITH ACTIONSCRIPT
Lesson overview ........................................ 48
Reviewing the starting file .......................... 50
Controlling movie clip properties with ActionScript .... 50
Creating animation using ActionScript tweens .... 57
Some suggestions to try on your own ............. 62

4 CREATING ACTIONSCRIPT IN EXTERNAL FILES
Lesson overview ........................................ 66
Creating an ActionScript file ....................... 68
Creating instances of a class file in Flash ......... 74
Some suggestions to try on your own ............. 81

5 USING ACTIONSCRIPT AND COMPONENTS TO LOAD CONTENT
Lesson overview ........................................ 84
Creating a List component instance and setting its parameters .......... 86
Adding an instance of the UILoader component .... 88
Adding a CHANGE event listener to the List component ... 89
Loading SWF files into a UILoader component .... 89
Creating the gallery file ............................ 91
Adding a scroll bar to the text field .............. 97
Some suggestions to try on your own ............. 98

6 CREATING PRELOADERS IN ACTIONSCRIPT 3.0
Lesson overview ........................................ 100
Tools in the testing environment ................... 102
Creating a text field and progress bar to track loading .... 106
<table>
<thead>
<tr>
<th>7</th>
<th>USING ARRAYS AND LOOPS IN ACTIONSCRIPT 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lesson overview</td>
<td>120</td>
</tr>
<tr>
<td>Examining the starting file</td>
<td>122</td>
</tr>
<tr>
<td>Adding MovieClip instances to the Stage from the Flash Library</td>
<td>123</td>
</tr>
<tr>
<td>Adding the initial variables</td>
<td>127</td>
</tr>
<tr>
<td>Generating multiple instances with a for loop</td>
<td>129</td>
</tr>
<tr>
<td>Generating drag-and-drop functionality</td>
<td>132</td>
</tr>
<tr>
<td>Creating the game play in an ENTER_FRAME listener</td>
<td>134</td>
</tr>
<tr>
<td>Using hitTestObject() to check for collisions</td>
<td>137</td>
</tr>
<tr>
<td>Some suggestions to try on your own</td>
<td>147</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8</th>
<th>CREATING AND FORMATTING TEXT WITH ACTIONSCRIPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lesson overview</td>
<td>150</td>
</tr>
<tr>
<td>Examining the starting file</td>
<td>153</td>
</tr>
<tr>
<td>Creating a TLF text field with ActionScript</td>
<td>156</td>
</tr>
<tr>
<td>Loading an external text file into a TLF text field</td>
<td>158</td>
</tr>
<tr>
<td>Using the TextFormat class</td>
<td>159</td>
</tr>
<tr>
<td>Giving the user a custom panel to format text</td>
<td>160</td>
</tr>
<tr>
<td>Creating a scroll bar using ActionScript</td>
<td>168</td>
</tr>
<tr>
<td>Some suggestions to try on your own</td>
<td>173</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9</th>
<th>CONTROLLING SOUND WITH ACTIONSCRIPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lesson overview</td>
<td>176</td>
</tr>
<tr>
<td>Examining the starting file</td>
<td>178</td>
</tr>
<tr>
<td>Adding sliders to the project</td>
<td>180</td>
</tr>
<tr>
<td>The Sound, SoundChannel, and SoundTransform classes</td>
<td>183</td>
</tr>
<tr>
<td>Setting the song titles using a for loop</td>
<td>185</td>
</tr>
<tr>
<td>Making the sliders invisible until needed</td>
<td>188</td>
</tr>
<tr>
<td>Programming the buttons to select songs</td>
<td>189</td>
</tr>
<tr>
<td>Controlling the visibility of the volume and pan controls</td>
<td>193</td>
</tr>
<tr>
<td>Adding a listener for the ID3 tags of an MP3 file</td>
<td>195</td>
</tr>
</tbody>
</table>
Accessing the user’s webcam or video camera using ActionScript .................................................. 277
Using the Bitmap and BitmapData classes .................................. 282
Examining the Pixel Bender Toolkit ............................................. 285
Suggestions to try on your own .................................................... 296

13 PRINTING AND SENDING EMAIL WITH ACTIONSCRIPT 3.0
Lesson overview ................................................................. 300
Examining the starting file ....................................................... 302
Adding a simple email link ..................................................... 302
Sending email from Flash ..................................................... 304
Adding printing capabilities with the PrintJob class ............. 309
Some suggestions to try on your own ............................... 317

14 CREATING ADOBE AIR APPLICATIONS WITH FLASH AND ACTIONSCRIPT
Lesson overview ................................................................. 320
Using AIR-specific ActionScript ........................................... 322
Specifying Flash publish settings for an AIR project .......... 324
Creating an AIR application ................................................. 328
Examining the starting file ....................................................... 330
Controlling printing with AIR .............................................. 332
Listening for drag events ..................................................... 333
Some suggestions to try on your own ............................... 340

15 USING THIRD-PARTY LIBRARIES TO EXTEND ACTIONSCRIPT
Lesson overview ................................................................. 342
Downloading and installing third-party ActionScript libraries ......................................................... 344
Examining the starting files ..................................................... 348
Using the Papervision3D BasicView class ......................... 351
Creating the constructor function for the Cube3D class .... 353
Creating and modifying a 3D cube ...................................... 355
Animating the 3D cube ......................................................... 358
Adding a movie clip as a material on a 3D object ............ 363
Adding video as a material on a 3D object ...................... 366
Some suggestions to try on your own ............................... 375

Index ................................................................................... 377
GETTING STARTED

Adobe Flash Professional CS5 provides a comprehensive authoring environment with tools for working with 2D and 3D animation, sound, vector and bitmap graphics, text, and video. Adobe ActionScript 3.0 is a sophisticated programming language that is totally integrated into Flash CS5 to develop rich interactive projects. The ActionScript techniques that you will learn in this book can be used with the great design and animation tools in the Flash interface to create rich interactive applications, games, e-learning content, and e-commerce tools for the web, the desktop, and mobile devices.

About Classroom in a Book

*ActionScript 3.0 for Adobe Flash Professional CS5 Classroom in a Book* is part of the official training series for the Adobe Flash platform, developed with the support of Adobe product experts. The lessons are designed so that you can learn at your own pace. If you’re new to ActionScript, you’ll learn the fundamental concepts and features you’ll need to accomplish a wide range of techniques covered in the book, but also gain enough understanding of the language to be able to learn additional techniques on your own. Each lesson in the book includes suggestions for continuing to develop your skills. Classroom in a Book teaches many advanced features, including tips and techniques for using the latest versions of ActionScript and Flash.

The lessons in this book include opportunities to use new features in Flash Professional CS5 such as Code Snippets, working with TLF Text, interacting with Adobe’s Pixel Bender Toolkit 2, and developing for the Adobe AIR 2 platform.

Prerequisites

Before you begin to use *ActionScript 3.0 for Adobe Flash Professional CS5 Classroom in a Book*, make sure that your system is set up correctly and that you’ve installed the required software. You should know how to use the mouse and standard menus and commands, and also how to open, save, and close files. If you need to review these techniques, see the printed or online documentation included with your Microsoft Windows or Apple Mac OS software.
This book is geared toward Flash users who are already comfortable with the interface and basic design and animation features of Flash. If you are new to Flash entirely, you may want to go through the lessons in Adobe Flash Professional CS5 Classroom in a Book.

This book assumes no programming experience. If you are a designer wishing to learn ActionScript 3.0, but perhaps are a little daunted by code, this book is for you. It also is helpful for Flash users who may have worked with earlier versions of ActionScript but have not yet made the transition to ActionScript 3.0.

Installing Flash

You must purchase the Adobe Flash Professional CS5 software either as a stand-alone application or as part of Adobe Creative Suite. Both products come with Flash Player 10, Adobe AIR 2, Adobe Media Encoder CS5, Adobe Extension Manager, Adobe Device Central, Adobe Bridge CS5, and the Pixel Bender Toolkit 2 in addition to the actual Adobe Flash CS5 application. Flash CS5 requires Apple QuickTime 7.6.2 or later. For system requirements and complete instructions on installing the Flash software, see the Adobe Flash ReadMe.pdf file on the application DVD.

Install Flash from the Adobe Flash Professional CS5 application DVD onto your hard disk. You cannot run the program from the DVD. Follow the onscreen instructions. Make sure that your serial number is accessible before installing the application. You can find the serial number on the registration card or on the back of the DVD case.

Optimizing performance

Flash Professional CS5 requires a minimum of 1 GB of RAM. The more RAM available to Flash, the faster the application will work for you. A broadband Internet connection is required for access to the online services offered by Adobe.

Copying the lesson files

The lessons in this book all revolve around a Flash project contained in an FLA file. Most of the lessons use additional resources such as audio, video, image, and text files. To complete the lessons in this book, you must copy these files from the ActionScript 3.0 for Adobe Flash Professional CS5 Classroom in a Book CD (located inside the back cover of this book) to your computer.

Copy the Lessons folder (which contains folders named Lesson01, Lesson02, and so on) from the ActionScript 3.0 for Adobe Flash Professional CS5 Classroom in a Book CD onto your computer by dragging it to your hard drive.
When you begin each lesson, you will be instructed where to navigate within the Lessons folder to find all the assets you need to complete the lesson.

If you have limited storage space on your computer, you can copy each Lesson folder individually as you need it and delete it afterward if desired. As mentioned before, some lessons build on preceding lessons but even then, the assets in each lesson folder are self-contained and don't require materials from other lesson folders. You do not have to save any finished project if you don’t want to or if you have limited hard disk space.

How to use these lessons

Each lesson in this book provides step-by-step instructions for creating a project that illustrates essential ActionScript techniques. Some lessons build on projects created in preceding lessons; others stand alone. All the lessons build on each other in terms of concepts and skills, so the best way to learn from this book is to proceed through the lessons in sequential order. Some techniques and processes are explained and described in detail only the first few times you perform them. Many of the most essential ActionScript processes are repeated throughout the exercises so that you can build a familiarity as well as a level of comfort with the basic tools in the language.

Each of the lesson folders contains a Start folder with the files that you will use to create the lesson as well as a Complete folder with a sample version of the completed lesson for reference; you can compare your work in progress against these samples of finished, working ActionScript. Some of the lessons also include other files and folders with media and resources needed to complete the lesson’s project. Be sure to keep each folder’s contents together.

Standard elements in the book

**Boldface text:** Words that appear in boldface indicate text that you must type while working through the steps in the lessons.

**Boldface code:** Lines of code that appear in boldface within code blocks help you easily identify changes in the block that you are to make in a step.

```actionscript
function moveUp(e:Event):void {
    if (jt0.position.y>165) {
        var pt0:Point=new Point(jt0.position.x-5,jt0.position.y-5);
        mover0.moveTo(pt0);
    } else {
        stage.removeEventListener(Event.ENTER_FRAME, moveUp);
        snapshot_btn.visible = true;
    }
}
```
Code in text: Code or keywords appear slightly different from the rest of the text so you can identify them.

Code and wrapped code lines: To help you easily identify ActionScript, XML, and HTML code within the book, the code has been styled in a special font that’s unique from the rest of the text. Single lines of code that are longer than the margins of the page allow wrap to the next line. They are designated by an arrow at the beginning of the continuation of a broken line and are indented under the line from which they continue. For example:

```javascript
var variables:URLVariables = new URLVariables();
var mailAddress:URLRequest = new
 URLRequest("http://www.actionscript.tv/email.php");
```

Italicized text: Words that appear in *italics* are either for *emphasis* or are *new vocabulary*.

Italics are also used for placeholders, in which the exact entry may change depending on your situation. For example:

```javascript
mailto:yourName@yourISP.com?subject=From Lesson 13 link&Body=
"This message was sent from Flash"
```

Menu commands and keyboard shortcuts: Menu commands are shown with angle brackets between the menu names and commands: Menu > Command > Subcommand. Keyboard shortcuts are shown with a plus sign between the names of keys to indicate that you should press the keys simultaneously; for example, Shift+Tab means that you should press the Shift and Tab keys at the same time.

Checking for updates

Adobe periodically provides updates to software. You can easily obtain these updates through Adobe Updater, as long as you have an active Internet connection.

1. In Flash Professional CS5, choose Help > Updates. The Adobe Updater automatically checks for updates available for your Adobe software.

2. In the Adobe Application Manager dialog box, select and download the updates you want to install. A message tells you if your application is already up to date. If the application is up to date, click Quit to close the Application Manager dialog box and return to Flash.
Flash Player version

The lessons in this book (with the exception of Lesson 14 which uses Adobe AIR) are created to work with Flash Player 10 or higher. While most web users have a recent version of the Flash Player, it is always a good idea before beginning your own Flash based projects to identify the target audience and determine which version of the Flash Player to develop for before starting the process. For information on Flash Player version penetration visit:

www.adobe.com/products/player_census/flashplayer/

Additional resources

ActionScript 3.0 for Adobe Flash Professional CS5 Classroom in a Book is not meant to replace documentation that comes with the program or to be a comprehensive reference for every feature. Only the commands and options used in the lessons are explained in this book. For comprehensive information about program features and tutorials, refer to these resources:

Adobe Community Help: Community Help brings together active Adobe product users, Adobe product team members, authors, and experts to give you the most useful, relevant, and up-to-date information about Adobe products. Whether you’re looking for a code sample or an answer to a problem, have a question about the software, or want to share a useful tip or recipe, you’ll benefit from Community Help. Search results will show you not only content from Adobe, but also from the community.

With Adobe Community Help you can:

- Access up-to-date definitive reference content online and offline
- Find the most relevant content contributed by experts from the Adobe community, on and off Adobe.com
- Comment on, rate, and contribute to content in the Adobe community
- Download Help content directly to your desktop for offline use
- Find related content with dynamic search and navigation tools

To access Community Help: If you have any Adobe CS5 product, then you already have the Community Help application. To invoke Help, choose Help > Flash Help. This companion application lets you search and browse Adobe and community content, plus you can comment on and rate any article just like you would in the browser. However, you can also download Adobe Help and language reference content for use offline. You can also subscribe to new content updates (which can be automatically downloaded) so that you’ll always have the most up-to-date content for your Adobe product at all times. You can download the application from www.adobe.com/support/chc/index.html.
Adobe content is updated based on community feedback and contributions. You can contribute in several ways: add comments to content or forums, including links to web content; publish your own content using Community Publishing; or contribute Cookbook Recipes. Find out how to contribute: www.adobe.com/community/publishing/download.html.

See http://community.adobe.com/help/profile/faq.html for answers to frequently asked questions about Community Help.

**Adobe Flash Professional CS5 Help and Support:** www.adobe.com/support/flash where you can find and browse Help and Support content on adobe.com.

**Adobe TV:** http://tv.adobe.com is an online video resource for expert instruction and inspiration about Adobe products, including a How To channel to get you started with your product.

**Adobe Design Center:** www.adobe.com/designcenter offers thoughtful articles on design and design issues, a gallery showcasing the work of top-notch designers, tutorials, and more.

**Adobe Developer Connection:** www.adobe.com/devnet is your source for technical articles, code samples, and how-to videos that cover Adobe developer products and technologies.

**ActionScript Technology Center:** www.adobe.com/devnet/actionscript is a special section of the Adobe Developer Connection designed specifically for ActionScript users.

**Resources for educators:** www.adobe.com/education includes three free curriculums that use an integrated approach to teaching Adobe software and can be used to prepare for the Adobe Certified Associate exams.

Also check out these useful links:

**Adobe Forums:** http://forums.adobe.com lets you tap into peer-to-peer discussions, questions, and answers on Adobe products.

**Adobe Marketplace & Exchange:** www.adobe.com/cfusion/exchange is a central resource for finding tools, services, extensions, code samples, and more to supplement and extend your Adobe products.

**Adobe Flash Professional CS5 product home page:** www.adobe.com/products/flash.
Adobe Labs: http://labs.adobe.com gives you access to early builds of cutting-edge technology, as well as forums where you can interact with both the Adobe development teams building that technology and other like-minded members of the community.

Adobe certification

The Adobe training and certification programs are designed to help Adobe customers improve and promote their product-proficiency skills. There are four levels of certification:

- Adobe Certified Associate (ACA)
- Adobe Certified Expert (ACE)
- Adobe Certified Instructor (ACI)
- Adobe Authorized Training Center (AATC)

The Adobe Certified Associate (ACA) credential certifies that individuals have the entry-level skills to plan, design, build, and maintain effective communications using different forms of digital media.

The Adobe Certified Expert program is a way for expert users to upgrade their credentials. You can use Adobe certification as a catalyst for getting a raise, finding a job, or promoting your expertise.

If you are an ACE-level instructor, the Adobe Certified Instructor program takes your skills to the next level and gives you access to a wide range of Adobe resources.


For information on the Adobe Certified programs, visit www.adobe.com/support/certification/main.html.
Accelerate your workflow with Adobe CS Live

Adobe CS Live is a set of online services that harness the connectivity of the web and integrate with Adobe Creative Suite 5 to simplify the creative review process, speed up website compatibility testing, deliver important web user intelligence and more, allowing you to focus on creating your most impactful work. CS Live services are complimentary for a limited time* and can be accessed online or from within Creative Suite 5 applications.

Adobe BrowserLab is for web designers and developers who need to preview and test their web pages on multiple browsers and operating systems. Unlike other browser compatibility solutions, BrowserLab renders screenshots virtually on demand with multiple viewing and diagnostic tools, and can be used with Dreamweaver CS5 to preview local content and different states of interactive pages. Being an online service, BrowserLab has fast development cycles, with greater flexibility for expanded browser support and updated functionality.

Adobe CS Review is for creative professionals who want a new level of efficiency in the creative review process. Unlike other services that offer online review of creative content, only CS Review lets you publish a review to the web directly from within InDesign, Photoshop, Photoshop Extended, and Illustrator and view reviewer comments back in the originating Creative Suite application.

Acrobat.com is for creative professionals who need to work with a cast of colleagues and clients in order to get a creative project from creative brief to final product. Acrobat.com is a set of online services that includes web conferencing, online file sharing and workspaces. Unlike collaborating via email and attending time-consuming in-person meetings, Acrobat.com brings people to your work instead of sending files to people, so you can get the business side of the creative process done faster, together, from any location.

Adobe Story is for creative professionals, producers, and writers working on or with scripts. Story is a collaborative script development tool that turns scripts into metadata that can be used with the Adobe CS5 Production Premium tools to streamline workflows and create video assets.

SiteCatalyst NetAverages is for web and mobile professionals who want to optimize their projects for wider audiences. NetAverages provides intelligence on how users are accessing the web, which helps reduce guesswork early in the creative process. You can access aggregate user data such as browser type, operating system, mobile device profile, screen resolution and more, which can be shown over time. The data is derived from visitor activity to participating Omniture SiteCatalyst customer sites. Unlike other web intelligence solutions, NetAverages innovatively displays data using Flash, creating an engaging experience that is robust yet easy to follow.

You can access CS Live three different ways:

1. Set up access when you register your Creative Suite 5 products and get complimentary access that includes all of the features and workflow benefits of using CS Live with CS5.

2. Set up access by signing up online and get complimentary access to CS Live services for a limited time. Note, this option does not give you access to the services from within your products.

3. Desktop product trials include a 30-day trial of CS Live services.

*CS Live services are complimentary for a limited time. See www.adobe.com/go/cslive for details.
Before you begin working through the lessons, it is worth taking a little time to understand the history of ActionScript and address a few topics that may clarify for you how Adobe ActionScript 3.0 works with Adobe Flash and the Flash platform.

A brief history of Flash and ActionScript

Flash and ActionScript have evolved together since Flash was originally released in 1996. Today, the combination of the design and animation tools in Flash CS5 and the advanced interactive capabilities of ActionScript 3.0 offers one of the most powerful, most versatile, and certainly most popular development environments available, but the origins of ActionScript as part of Flash were fairly humble.

In the first three versions of Flash, there were no programming tools available, and interactivity meant selecting from a few simple drag-and-drop options in the Actions panel. These actions allowed for navigation of the Flash Timeline and creating links to URLs, but not much more.

Flash 4 was the first version that allowed for entry of code using a simple scripting language, which became informally known as ActionScript. In Flash 5, ActionScript evolved even more and became an official scripting language. With each version of Flash since that time, the capabilities of ActionScript have become richer, offering interactive control of animation, text, sound, video, data, and much more. In 2003 ActionScript 2.0 was introduced, and its capabilities were on par with object-oriented languages such as Java and C#. You will learn more about object-oriented programming (OOP) starting in Lesson 4, “Creating ActionScript in External Files.”

Serious programmers started becoming more interested in ActionScript as a development tool, but they found that even though ActionScript 2.0 rivaled the features of other languages, it did not rival their performance. This was because each version of ActionScript was built on the foundation of the previous one, going all the way back to its very simple beginnings. Flash Player was
not originally designed for creating high-performance applications and games, but developers began using it for those purposes. It became clear that a new version of ActionScript needed to be written from the ground up.

In 2006, Adobe introduced ActionScript 3.0, which offered significant new functionality as well as dramatic performance increases. Flash CS3 was the first version of Flash to incorporate ActionScript 3.0. Flash CS4 added functionality to ActionScript 3.0, including new 3D capabilities, new animation controls, and ActionScript classes for working with Adobe AIR (see Lesson 14, “Creating Adobe AIR Applications with Flash and ActionScript”). Flash CS5 continues the evolution of ActionScript 3.0 and has added lots of new ActionScript for working with advanced text features; enhancing the AIR platform; and working with a variety of devices and controllers, including multitouch and touch-screen devices. Flash CS5 also has a number of new features to help you learn and work with ActionScript, including the new Code Snippets panel, which lets you reuse common ActionScript code with the click of a mouse. Other new ActionScript features, such as code completion and tooltips for custom classes, will show their worth as you begin working with the language.

ActionScript 3.0 for new programmers

Having the power and sophistication of ActionScript 3.0 within Flash is wonderful, but with these capabilities comes more complexity and a steeper learning curve. Many designers and animators who use Flash regularly are daunted by the prospect of learning ActionScript 3.0, and the majority of books on the subject are written for those with programming experience. The truth is that with a little patience at the beginning, you can quickly learn enough ActionScript to be able to add lots of interactive features to your Flash work.

The lessons in this book are geared toward designers who have little or no programming experience. Some knowledge of ActionScript 1.0 or 2.0 is of course useful, but should not be necessary to successfully complete the lessons.

By working through these lessons, you will gain a comfort with the syntax of ActionScript 3.0. More importantly, you will gain a large repertoire of interactive tools to add to your existing Flash skills. You’ll also build a foundation that will allow you to continue your ActionScript education using the material at the Adobe Flash Developer Center (adobe.com/devnet/flash) and the many other books and resources available.
For users of ActionScript 1.0 and 2.0

Much has changed in ActionScript 3.0 compared with ActionScript 1.0 and ActionScript 2.0, and some advanced ActionScript 1.0 and 2.0 programmers are still intimidated by the prospect of learning ActionScript 3.0. The next sections cover some points that may help you to make the transition and convince you that the benefits of ActionScript 3.0 will justify the effort.

First, the bad news

There is no doubt that ActionScript 3.0 is more verbose than earlier versions of the language; this means that, especially in the beginning, you have to type more code to get the same results. The payoff becomes apparent fairly quickly, but at first glance, ActionScript 3.0 can be a little scary for new users.

Also, Flash applications written in ActionScript 3.0 cannot be simply integrated with Flash projects created with earlier versions of ActionScript. This is because there are actually two ActionScript players inside Flash Player 9 and later. Flash Player contains ActionScript Virtual Machine 1 (AVM1), which plays files created with ActionScript 1.0 and ActionScript 2.0, and Virtual Machine 2 (AVM2), which plays files created with ActionScript 3.0. While it is possible for files to communicate between the two virtual machines, it is not as simple as communicating with files created with the same version of AVM. In this book, we will focus exclusively on ActionScript 3.0, but if you plan on integrating new ActionScript 3.0 projects into older Flash websites or applications, you should thoroughly study the resources in Flash Help on integrating ActionScript 3.0 with older files.

...and now the good news

ActionScript 1.0 and ActionScript 2.0 developers who have made the transition to ActionScript 3.0 very quickly appreciate its advantages, especially:

- Better performance. As mentioned, ActionScript 3.0 code executes much faster than earlier versions of the language—usually 2 to 10 times, but sometimes up to 100 times, faster. This makes Flash viable for creating high-performance games, simulations, 3D interfaces, and data-driven applications.

- More consistent syntax. Because everything up to ActionScript 2.0 was built on top of previous versions, there were often many ways to do similar things. This could be extremely confusing. For example, in ActionScript 1.0 and 2.0 something as simple as responding to an event or creating a new object could be dramatically different, depending on what the event or object was. As you will see beginning in Lesson 2, “Working with Events and Functions,” once you learn how to do something in ActionScript 3.0, the syntax will remain consistent throughout the language. For example, there is one way to listen for and respond to an event in ActionScript 3.0, regardless of the type of event.
• Better error checking and feedback. Everyone makes mistakes, so it is a blessing that ActionScript 3.0 offers much better feedback to help you identify and correct errors in your code.

• Lots of new features. ActionScript 3.0 has introduced dozens of new classes that offer functionality that was previously unavailable, including ways of working with sound, video, text, XML, 3D, and lots more. As you progress through the lessons you will become comfortable with many of these features.

Formats for playing back Flash and ActionScript 3.0 files

Usually, creating a website or application in Flash means publishing your finished work as a SWF file that can be played using Flash Player, most typically in a web browser. This is the most common use of Flash for most developers.

Flash has also always provided the option of creating platform-specific projector versions of your projects. These are self-running executable files that can be created for either Macintosh or Windows.

Not long ago, Adobe introduced its Adobe AIR technology, which allows creation of true cross-platform desktop applications that run on Macintosh, Windows, or Linux. Adobe AIR applications can be made using Flash CS5, which includes a number of new features for AIR 2.0. In Lesson 14, “Creating Adobe AIR applications with Flash and ActionScript,” you will learn to use ActionScript 3.0 to create desktop applications that can access the user’s operating system and printer.

Flash CS5, Flash Builder 4, and Flex

Many Flash users have heard of Adobe Flash Builder and Flex but are not sure how or if they fit into their development process. Flash CS5 and Flash Builder 4 are both commercial applications from Adobe. Flash Builder 4 is the new name for what formerly was called Flex Builder. You can use Flash CS5 and/or Flash Builder 4 to create SWF files for Flash Player as well as stand-alone Adobe AIR applications. Another option for experienced programmers is to use the free Flex SDK that is available from Adobe at www.adobe.com/products/flex/flex_framework/.

All of these programs support the entire ActionScript 3.0 language. Flash Builder is more geared toward people with a programming background and includes a number of features that support the development of large-scale rich-media applications and data-driven projects. Flash CS5 on the other hand includes tools and an interface adapted to the needs of designers and animators.
If you do projects that integrate a lot of design, video, animation, and media but also have lots of interactivity that requires serious amounts of coding, you may wish to consider developing your Flash projects using both Flash CS5 and Flash Builder 4. Both tools ship with the Adobe CS5 Web Collection and are very well integrated. Many developers or teams will create the visual parts of an application in Flash CS5 and then from within Flash CS5, launch and use Flash Builder 4 to write their ActionScript code. Of course this is an optional step—code can also be written exclusively in Flash CS5.

This book focuses on the use of ActionScript 3.0 in Flash CS5, but all the concepts and nearly all the code would work equally well in Flash Builder 4.

**ActionScript in the Flash Timeline vs. external ActionScript files**

Traditionally, ActionScript in Flash has been placed on keyframes in the Timeline. In earlier versions of Flash, ActionScript could also be placed directly on an object such as a button or a movie clip, but this is no longer the case with ActionScript 3.0.

ActionScript can also be used in a more standardized OOP (Object Oriented Programming) environment. ActionScript 3.0 is based on the ECMA standard and has many similarities to other languages including Java, C#, and C++. ActionScript is a true object-oriented language, which makes it very good for building larger and more complex projects. While this book does not put an emphasis on OOP, the later lessons will lay a foundation that will allow you to delve more deeply into OOP development in ActionScript 3.0 if you wish. An alternative to placing code on the Timeline is to create dedicated ActionScript files that can be used in any Flash project. This is the foundation for OOP in Flash.

In the early lessons of this book, you will be placing all your code in the Flash Timeline. Starting in Lesson 4, “Creating ActionScript in External Files,” you will begin working with external ActionScript class files and begin to learn to take advantage of OOP principles.

**That’s enough background for now...**

Let’s get started in Lesson 1, “Using Code Snippets and Navigating the Flash Timeline,” where, as you may guess by the title, you will learn to work with the new Code Snippets features in Flash CS5 and use ActionScript 3.0 to navigate the Flash Timeline.
Lesson overview

In this lesson, you will learn to do the following:

- Create an ActionScript file using the tools in Flash CS5.
- Create an ActionScript class that extends the MovieClip class.
- Create a constructor function.
- Define parameters for class methods.
- Use ActionScript code to create vector graphics.
- Use code to create instances of a custom class file in the Flash Timeline.
- Use the MOUSE_MOVE event.
- Turn off an event listener.
- Generate random color.

This lesson will take approximately 2 hours.

External ActionScript files can be convenient for reusing code, or they can be the foundation for large applications that use object-oriented programming (OOP) practices.
Each mouse movement in the Lesson 4 project produces a trail of ellipses with a different random color.
Up until this point, all the lessons have created ActionScript in frames of the Flash Timeline. This is a very useful way to work, and many Flash developers create all their projects exclusively using Timeline code. For simpler projects, this approach has the benefit of keeping all the graphics and media in the same file as the ActionScript. However, for Flash projects that are more complex, it is often cumbersome to have hundreds or even thousands of lines of code in the Timeline.

The alternative is to store the ActionScript for a project in one or more external files that contain only code. These ActionScript files can then be integrated with graphics, animation, and other Flash content to create the final project. External ActionScript files are really just plain text files saved with the .as file extension.

There are many benefits to developing larger projects in this manner. Most important, this approach allows you to divide the functionality of your applications into reusable chunks of code.

One common characteristic of a successful ActionScript file is that it’s written to be as versatile as possible. For example, suppose you create an ActionScript file called Scoring.as that contains code to keep track of a user’s score. By writing the code so that some of its properties can be individually modified, you can use the code in a game in which the user gets 10 points for shooting space aliens and needs 1000 points to win, or in a history quiz in which the user gets 1 point for each correct answer and proceeds to the next lesson when the score reaches 20 points.

In this lesson, you will get some experience creating an external ActionScript file designed to generate a simple graphic. You will then use this external .as file in a very simple Flash painting application as a source of generated brushstrokes.

### Creating an ActionScript file

As mentioned, an ActionScript file is just a plain text file. Therefore, ActionScript files can be created with any software that can create text files, including TextEdit on the Mac and Notepad in Windows. Of course, it is preferable to create ActionScript files in tools that offer features like color-coding and error checking. Adobe tools like Flash Professional CS5, Flash Builder, and Dreamweaver offer full support for creation of ActionScript files, as do a number of third-party tools, some of which are free. So you have a lot of options, and over time you’ll find the ones that are right for you. In this lesson, you will create an ActionScript file right in Flash CS5.

### Creating an ActionScript class in Flash

In this lesson, you will use the tools in Flash CS5 to create a new ActionScript class file. When you create an external class file, you can place code in this file that can be used by other Flash projects.
In fact, you can think of the ActionScript 3.0 language as a large collection of classes that together offer the wide range of functionality available in the language.

If you have worked in Flash at all, then you have already worked with classes. For example, the MovieClip class is a built-in ActionScript class that describes the functionality that is unique to movie clips in Flash. In the previous lesson, you were able to manipulate many movie clip properties such as rotation and alpha, because the MovieClip class references code that describes those properties. Similarly, there are built-in ActionScript classes for working with text, sound, video, and other features available in the Flash platform.

When you create a movie clip in Flash, you are actually creating an instance of the MovieClip class. When you work with a text field, you are using an instance of the TextField class; with video, you use the Video class, and so on.

The functions that are contained in an ActionScript class file are referred to as methods of that class, and the variables that are in a class file are referred to as properties of the class.

By creating custom ActionScript classes, as you will do in this lesson, you are expanding the collection of classes that are available to you in your Flash projects. And many beginners find that once they become comfortable with the way classes work in ActionScript 3.0, then the entire language begins to make a lot more sense.

In previous versions of Flash, to create a new ActionScript class you had to create the new class from scratch, creating a blank ActionScript file and typing all of the code. In Flash CS5, you now have the option of creating a new ActionScript 3.0 Class file. This approach creates an ActionScript file with the basic code structure for an ActionScript class already added for you. You will do this now.

1. In Flash, choose File > New.

2. On the General tab of the New Document dialog box, choose ActionScript 3.0 Class and then click OK.
You will be presented with a dialog box offering you the choice of using Flash Professional to create your new ActionScript file, or launching Flash Builder (if it is installed on your machine) and creating the ActionScript class file in Flash Builder. There are a number of benefits to working in Flash Builder, especially when debugging, testing, and profiling complex applications, but for these lessons we will work exclusively in Flash Professional CS5.

3 Choose Flash Professional in the dialog box.

4 In the Class Name field, type **Ellipse** and click OK to close the dialog box.

   A new ActionScript file is created with basic code for a class named Ellipse.

5 Choose File > Save, and name your new file **Ellipse.as**.

6 Save the new file in the Lesson04 > Start folder of your lessons folder.

   The name of the class created in an ActionScript file must match the name of the file. Thus, in this case, the class is named **Ellipse**, and you saved the file as Ellipse.as. This name structure is what allows class files to be referenced by other files.

### Basic structure of an ActionScript 3.0 Class file

In your new Ellipse.as file, look at the basic structure common to most .as class files. This is the code that was created by Flash in your new **Ellipse** class:

```actionscript
package {

    public class Ellipse {

        public function Ellipse() {
            // constructor code
        }

    }

}
```

All ActionScript class files begin with the `package` keyword. You can think of the package as describing the path to your ActionScript file. For example, if the Ellipse.as file was inside a folder named scripts, and the scripts folder was in a folder named com, then the first line of code in the file would read:

```actionscript
package com.scripts {
```

For simplicity, in this example you will keep all of the Flash files and ActionScript files for the lesson together in the same directory. When this is the case, then no path needs to be described after the `package` keyword. However, the `package` reference still needs to be there. The right brace at the end of the code closes the `package` tag. All the code for the **Ellipse** class will go in between the `package` braces.
After the opening **package** tag is the code that creates a new ActionScript class named **Ellipse**. All the code for this class will go between this line and the second from the last right brace:

```
public class Ellipse{
```

1. Change this line so that it reads:

```
public class Ellipse extends MovieClip {
```

When one class extends another class, the new class has all the capabilities of the original parent class in addition to whatever new capabilities are added in the new class file.

In this case, the purpose of the **Ellipse** class will be to draw a simple ellipse. Here the keyword **extends** makes the **Ellipse** class a descendent, or child, of the **MovieClip** class. This allows instances of the **Ellipse** class to do all the things that can be done with movie clips, such as have their position, scale, and rotation set.

To make the **MovieClip** class available to your new **Ellipse** class, you need to write code that imports the **MovieClip** class into your file.

2. Place the following line between the opening **package** tag and the line that creates the public class **Ellipse**:

```
import flash.display.MovieClip;
```

The file so far should read:

```
package  {

import flash.display.MovieClip;

public class Ellipse extends MovieClip {

   public function Ellipse() {
       // constructor code
   }
}
}
```

The final bit of code that was created when you created the new class file is a function named **Ellipse()**:

```
public function Ellipse() {
   // constructor code
}
```

Although a class file will often contain many functions, each class must have one function that has the same name as the class file. This is called the constructor function, and because it is mandatory, it is created automatically in Flash CS5 when you create a new class file. The constructor function is automatically called every time an instance of the class is created. You will put this into practice soon.
About access modifiers

There is another term in the line that creates the Ellipse class that you may not have encountered before:

```actionscript
public class Ellipse extends MovieClip {
```

The term `public` is what is known as an access modifier. By setting the class to `public`, you are indicating that this class can be accessed from any other file.

There are three other access modifiers:

- **private**: Methods and properties that are labeled `private` are available only from within the class file.
- **protected**: Methods and properties that are labeled `protected` are available only from within the class file and its descendents.
- **internal**: Methods and properties that are labeled `internal` are available from within the class file and to all other files within the same package.

For simplicity’s sake, these lessons will mostly use only the `public` and `private` modifiers, but in other class files, you may see the other access modifiers used on methods (functions) and properties (variables) within the class. As you get more comfortable with ActionScript, it is a good idea to make a deeper study of how to use these modifiers in your applications. For more information, see the ActionScript 3.0 Help files or Colin Moock’s excellent book, *Essential ActionScript 3.0*, from O’Reilly Media.

As mentioned, it is not uncommon for a class file to contain many functions, but because the function you’re creating here is a simple function that will perform only one task, you will put all of the code for this class file in the `Ellipse()` constructor function.

3. Modify the `Ellipse()` function so that it reads:

```actionscript
public function Ellipse(w:Number=40,h:Number=40,color:Number=0xff0000) {
  graphics.beginFill(color);
  graphics.drawEllipse(0, 0, w, h);
  graphics.endFill();
}
```

Notice the three parameters you’ve added between the parentheses of the `Ellipse()` function. These will be used to set the width, height, and color of a new ellipse that will be created when the `Ellipse()` function is called. The parameters for width (w) and height (h) are given default values of 40, and the `color` parameter is given a default value of red.

The three lines of code inside the brackets of the `Ellipse()` function are used to create an ellipse. These lines call methods that are part of the extensive and
powerful ActionScript toolset for creating and manipulating vector graphics. In the `Ellipse()` function, the first of these lines indicates that the vector graphics that are about to be created will have a specific fill color:

```actionscript
graphics.beginFill(color);
```

The `color` parameter determines the color of the ellipse. Remember that this parameter was set to a default of red when you created the function, but can be overridden when called.

The second line of code draws an ellipse using a built-in function called `drawEllipse()`:

```actionscript
graphics.drawEllipse(0, 0, w, h);
```

This function, or *method*, takes four parameters. The first two parameters set the position of the ellipse, in this case to 0 horizontally and 0 vertically (the upper-left corner). The next two use the `w` and `h` parameters of the `Ellipse()` function to set the width and height of the ellipse.

---

### Required versus optional parameters

If a function has parameters that are given default values, as in the example in step 3, then when the function is called, references to those parameters do not need to be included. These are called *optional* parameters. If references to these parameters are included with new values, they will override the default values. You will see this in action soon.

If a function has parameters that are not given initial values, you need to assign these values when calling the function. These are called *required* parameters.

The third line inside the `Ellipse()` function ends the fill and completes the drawing:

```actionscript
graphics.endFill();
```

4. Save your file. Your entire `Ellipse` class file should now read:

```actionscript
package {
    import flash.display.MovieClip;
    public class Ellipse extends MovieClip {
        public function Ellipse(w:Number=0, h:Number=40, color:Number=0xFF0000) {
            graphics.beginFill(color);
            graphics.drawEllipse(0, 0, w, h);
            graphics.endFill();
        }
    }
}
```

You’ll soon get to test your handiwork.
**ActionScript 3.0 and hexadecimal color**

ActionScript 3.0 can describe colors in a variety of ways, but the most common is as numeric hexadecimal values. This system is very easy once you are used to it. The characters “0x” before a color description tell ActionScript that a hexadecimal value is to follow. Then a six-digit number describes the amount of red, green, and blue in the color. (Optionally, an eight-digit number can be used; in addition to the color values, it would include transparency information.)

If you have worked with hexadecimal colors in web design, you know that each digit can range from 0 to 15, with the letters A, B, C, D, E, and F representing the numbers 10, 11, 12, 13, 14, and 15, respectively. In this example, the color red is described as 0xFF0000, which has the greatest possible amount of red (FF) and no green (00) or blue (00). The hexadecimal color 0x0000FF would be a color with no red (00) or green (00) and the full amount of blue (FF).

To find the hexadecimal value of a specific color in Flash, you can open the Color panel (Window > Color). You can select a color in a variety of ways in this panel. The hexadecimal value of the selected color will be displayed in the lower right of the panel. If you are using a value from the Color panel in your ActionScript, replace the initial pound symbol (#) shown in the color panel with “0x” before typing the hexadecimal value in your code.

For more information about hexadecimal colors, see Flash Help or any basic web design book.

---

**Creating instances of a class file in Flash**

Without further ado, let’s put your new class file to work.

1. Open the lesson04_start.fla file from the Lessons > Lesson04 > Start folder. This should be the same location where your ActionScript file is saved.

   Notice that this file is simply made up of a background layer with a full-screen bitmap image and an empty actions layer with no code added (yet).

2. With Frame 1 of the actions layer selected, open the Actions panel and select the first line, where you’ll begin adding code.

3. To create a single instance of your Ellipse class, add the following code:
   ```actionscript
   var ellipse:Ellipse = new Ellipse();
   ```

4. To add the ellipse to the Stage, on a new line type the following code:
   ```actionscript
   addChild(ellipse);
   ```
Using the keyword new to create instances

To create a new instance from any ActionScript class, you use the keyword new. This is consistent across the entire ActionScript 3.0 language, whether you are creating instances of built-in classes as in:

```actionscript
var myClip:MovieClip = new MovieClip();
```

and:

```actionscript
var userForm:TextField = new TextField();
```

or, as in this lesson, you are creating a new instance of a custom class as in:

```actionscript
var ellipse:Ellipse = new Ellipse();
```

Many newcomers to ActionScript find that this consistency makes ActionScript much easier than they expected once they get comfortable with learning the foundations of the language.

About addChild() and the display list

In the background of every Flash file, every visual object that is onstage is tracked in what is called the **display list**. This is true whether a visual object was placed onstage using the tools in the Flash interface, imported to the stage as an external file, or created from scratch using ActionScript.

All visual objects in a Flash project, including movie clips, shapes, buttons, text fields, bitmaps, and video, are considered **display objects** and are added to the display list when they are made viewable.

When a visual object is created with ActionScript, it may exist in code, but that does not mean that it will automatically be visible onstage. To place something in the display list, and therefore onstage, you call the method `addChild()`. A common mistake for ActionScript beginners is to forget to use `addChild()` and then wonder why the expected graphics do not appear onstage. You will be delving deeper into display objects and the display list in later lessons.
5 Save and test your movie. You should see a single red ellipse in the upper-left corner of the Stage.

A single red ellipse is not too exciting, so next you will add a few things to make more interesting use of the Ellipse class.

First, instead of having a single instance of the Ellipse generated automatically, you will let the user generate multiple instances, creating a new instance whenever the mouse is moved.

6 Select all the existing code in the Actions panel and cut it to the clipboard.

7 On the first line of the now empty Actions panel, add an event listener for an event called MOUSE_MOVE:

   stage.addEventListener(MouseEvent.MOUSE_MOVE, makeShapes);

   This event takes place whenever the user moves the mouse. This movement will call a function called makeShapes().

8 On a new line, create the makeShapes() function:

   function makeShapes(e:MouseEvent):void {
   }

9 Paste the code from the clipboard in between the curly braces of the makeShapes() function so that the function now reads:

   function makeShapes(e:MouseEvent):void {
     var ellipse:Ellipse = new Ellipse();
     addChild(ellipse);
   }

   If you tested your movie now, every time the mouse was moved, a new ellipse would be added to the stage—but they would all be in the exact same spot in the upper left. As with the parent MovieClip class, each Ellipse class instance
has an X and Y property with a default location of 0,0. To give each new ellipse a unique location, you will set each new ellipse to be placed at the current mouse location using the mouseX and mouseY properties.

10 Add two new lines to the makeShapes() function so that it now reads:

```actionscript
function makeShapes(e:MouseEvent):void {
    var ellipse:Ellipse = new Ellipse();
    addChild(ellipse);
    ellipse.x = mouseX;
    ellipse.y = mouseY;
}
```

11 Save and test your movie. Move the mouse around. A trail of red circles should be created that follow your mouse path. Congratulations, you have created a virtual paintbrush that uses big red ellipses (which are circles because the w and h parameter default values were set equal). More important, you have succeeded in creating and using a custom ActionScript class in a Flash file!

12 Close the lesson04_start.swf file to exit the testing environment.

**Overriding the parameters of each ellipse instance**

At this point, your Flash file is creating nothing but big red ellipses from your class file—but remember, they are big and red because those are the defaults you placed in the constructor function. Each time a new ellipse is created, those defaults can be overridden by passing new parameters. Let’s change the parameters to create smaller green ellipses.

1 In the makeShapes() function, change the line of code that currently reads:

```actionscript
var ellipse:Ellipse = new Ellipse();
```

so that it reads:

```actionscript
var ellipse:Ellipse = new Ellipse(10, 10, 0x00FF00);
```

2 Save and test your movie.
Now, moving the mouse should produce a trail of 10-pixel-by-10-pixel green circles. If you want, you can experiment by trying different sizes and colors and test the results.

Turning the makeShapes() function on and off

Even software that does nothing but paint green trails should give users control over when they paint. So far, you have added event listeners using the `addEventListener()` method; you can also remove a listener using a similar method called `removeEventListener()`. Here, you'll alter your code so that the listener for mouse movement is added when the user clicks onstage and removed when the mouse is released.

1. In the Actions panel, click to place the mouse pointer before the first line of code and press the Enter (Windows) or Return (Mac) key a few times to insert a few lines of space before the beginning of the code.

2. On the first line of the Actions panel, above the existing code, add two new `addEventListener()` methods to listen for the `MOUSE_UP` and `MOUSE_DOWN` events by typing the following code:

   ```actionscript
   stage.addEventListener(MouseEvent.MOUSE_DOWN, startDrawing);
   stage.addEventListener(MouseEvent.MOUSE_UP, stopDrawing);
   ```

   The `MOUSE_DOWN` event will call a function named `startDrawing()`, and the `MOUSE_UP` event will call a function named `stopDrawing()`, so next add those two new functions.

3. On the lines below the event listeners, add this code:

   ```actionscript
   function startDrawing(e:MouseEvent):void {
   }
   function stopDrawing(e:MouseEvent):void {
   }
   ```
Next, find and select the line in your code that reads:

```actionscript
stage.addEventListener(MouseEvent.MOUSE_MOVE, makeShapes);
```

Cut this line of code (Edit > Cut) to place it on the clipboard.

Place the mouse pointer between the curly braces of the new `startDrawing()` function and paste the code from the clipboard. The function should now read:

```actionscript
function startDrawing(e:MouseEvent):void {
    stage.addEventListener(MouseEvent.MOUSE_MOVE, makeShapes);
}
```

Place the mouse pointer between the curly braces of the `stopDrawing()` function and paste the same code from the clipboard.

In your newly pasted code in the `stopDrawing()` function, change `addEventListener` to `removeEventListener`. The function should now read:

```actionscript
function stopDrawing(e:MouseEvent):void {
    stage.removeEventListener(MouseEvent.MOUSE_MOVE, makeShapes);
}
```

The result of these changes is that the function that draws the ellipses when the mouse moves will occur only when the user clicks the mouse and will stop occurring when the mouse is released.

Save and test your movie. Click the Stage and move the mouse. Ellipses should be created that follow the mouse. Release the mouse, and the ellipses should stop being generated.

### Randomizing the color of the ellipses

To generate a random number in ActionScript 3.0, you use the `random` method of the `Math` class. The syntax for that is:

```actionscript
Math.random();
```

This code will return a random number between 0 and 1, usually with multiple decimal places. To control the range that `Math.random` generates, you perform some math on the resulting random number. For example, if you want to generate a random number between 0 and 50, you multiply the `Math.random` result by 50:

```actionscript
Math.random() * 50;
```

If you want to generate a random number from among the full range of possible hexadecimal colors, you write:

```actionscript
Math.random() * 0xFFFFFF;
```

Now you’ll use this technique to add random colors to the ellipses.
1 Add a variable to your file to store a numeric color value: At the top of the Actions panel, above the existing code, add a new line and create a new variable with this code:

   var color:Number;

2 Locate the startDrawing() function and add to the code so that it now reads:

   function startDrawing(e:MouseEvent):void {
       stage.addEventListener(MouseEvent.MOUSE_MOVE, makeShapes);
       color = Math.random() * 0xFFFFFF;
   }

Now each time the user clicks to begin drawing, a new random color will be chosen.
To assign that color to the ellipses, you will use the new color variable as the parameter that is passed to the Ellipse() constructor function.

3 Locate the makeShapes() function and change the line that currently reads:

   var ellipse:Ellipse = new Ellipse(10,10,0x00FF00);

so that it reads:

   var ellipse:Ellipse = new Ellipse(10,10,color);

4 Save and test your movie. Each mouse movement produces a trail of ellipses with a different random color.

The completed code in Flash should now read:

   var color:Number;
   stage.addEventListener(MouseEvent.MOUSE_DOWN, startDrawing);
   stage.addEventListener(MouseEvent.MOUSE_UP, stopDrawing);
   function startDrawing(e:MouseEvent):void {
       stage.addEventListener(MouseEvent.MOUSE_MOVE, makeShapes);
       color = Math.random() * 0xFFFFFF;
   }
   function stopDrawing(e:MouseEvent):void {
       stage.removeEventListener(MouseEvent.MOUSE_MOVE, makeShapes);
   }
function makeShapes(e:MouseEvent):void {
    var ellipse:Ellipse = new Ellipse(10, 10, color);
    addChild(ellipse);
    ellipse.x = mouseX;
    ellipse.y = mouseY;
}

By learning to create external ActionScript files and integrate them into your Flash projects, you can begin to make your rich interactive applications much more modular. It can take some time to get comfortable with this way of working, but the efforts will be very rewarding.

In the coming lessons, you will get more practice working with ActionScript classes.

Give your brain a rest between each lesson, and go back to earlier lessons for review as many times as you need to. You may be surprised how much more sense ActionScript concepts make after you are exposed to them a few times.

Some suggestions to try on your own

There are many, many ways to enhance the application you created in this lesson using techniques that we have already covered.

The Lesson04 folder has an Addendum folder containing a tutorial that goes through the steps of creating a class that is a simple variation of the Ellipse class, but that creates rectangles instead of ellipses. Use the Lesson 4 addendum file “Creating Animation with ActionScript—Addendum,” in the Lesson04 > Addendum folder, to create the second class file, and then try experimenting with some of the following techniques:

- Change your Flash file so that mouse movements paint rectangles instead of ellipses.
- Create buttons that allow users to switch between painting ellipses and painting rectangles.
- Create buttons that let users set the size of the shapes that they paint.
- Create buttons that let users choose the color they paint.
- Look in the Flash Help files and explore some of the other possible shapes you can create with the drawing capabilities in ActionScript. See if you can create additional ActionScript files that create new shapes and then incorporate them into your Flash file.

You will learn more about generating visual elements with ActionScript in upcoming lessons. In the next lesson, you will learn to import external content into a Flash application at runtime using ActionScript and Flash components.
Review questions

1. When creating an ActionScript class file, how should the file be named?
2. How does the constructor function in an ActionScript class file need to be named?
3. Define an ActionScript method and an ActionScript property.
4. What is the difference between a required parameter and an optional parameter in an ActionScript method?
5. How do you create an instance of an external class in ActionScript?
6. How is a display object added to the display list in ActionScript?
7. What is one way to generate a random color in ActionScript?

Review answers

1. An ActionScript class file must have the same name as the class that it contains, followed by the suffix .as. For example, if a file contains an ActionScript class called ScoringSystem, then the filename needs to be ScoringSystem.as.

2. The constructor function in an ActionScript class file is the function in that file with the same name as the class. For example, in a class named ScoringSystem, the constructor function would look like this:
   
   ```actionscript
   public function ScoringSystem(parameters){
   //code that does something goes here
   }
   ```

3. A method in ActionScript 3.0 is a function that is contained in a class. A property in ActionScript 3.0 is a variable contained in a class.

4. When a function is created in an ActionScript class file, it can be given any number of parameters. If those parameters are given initial default values when they are created, then they are considered optional parameters, and it is not necessary to pass parameters to the function when calling it. If a parameter does not have a default value, then a value must be passed when the function is called, and these are required parameters. For example, in the following example, the finalScore parameter has no initial value, so it is a required parameter. However, the startingScore parameter has an initial value of 0, so it is an optional parameter.
public function ScoringSystem(finalScore:Number,  
  startingScore:Number = 0){
  //code that does something goes here
}

5 To create an instance of an external class in ActionScript, you can use the keyword new followed by the class name. For example, to create a new instance of the Rocket class in a variable named rocket1, you can write:
  var rocket1:Rocket = new Rocket();

6 To add an object to the display list with ActionScript and make it appear onstage, you use the addChild() method. For example, to add an instance named rocket1 to the Flash Stage, you can write:
  addChild(rocket1);
  or
  stage.addChild(rocket1);

7 You can generate a random color value by calling the Math.random() method and multiplying the result by the full range of hexadecimal colors, as in:
  var color:Number = Math.random() * 0xFFFFFF;
INDEX

SYMBOLS

* (asterisk), 79, 269
: (colon), 23, 33
{ } (curly braces), 33
= (equal sign), 23, 43
= =, 43
( ) (parentheses), 33
+ (plus sign), 42, 47
++ characters, 23
+=, 53
“ ” (quotation marks), 27
"/n/n", 306
./, 190
/* and */ tags, 184
//, 184

A

about this book, 1–2, 3–4
access modifiers, about, 72
Actions panel, 16–18
ActionScript. See also external ActionScript files; third-party ActionScript libraries; writing ActionScript
ActionScript cue points, 244–246, 260
adding toggle functionality, 161–164
AIR-specific classes in, 322–323
case sensitivity of, 33
casting data for, 241
changes in version 3.0, 11–12
class creation for, 68–70
coding custom menus for AIR, 330
comments in, 19–20, 184
concatenating text in, 42, 47
controlling movie clip instances with,
124–126, 148
creating variables, 22–23
easing classes, 58, 59–62
empty lines between code sections, 40
evolution of Flash and, 9–10
external code files for, 13, 36, 66, 68
files in, 68
generating cue points automatically, 247
hexadecimal color and, 74
hints for debugging, 62
ID3 tag compatibility with, 196
learning version 3.0, 10
listening to and responding to events, 32–34
locating XML elements in, 214, 227
managing FLVPlayback properties in,
236–238
manipulating IK systems from, 264, 265
naming rules in, 33
placing in Timeline frames, 13, 16–17
playing back files for Flash and, 12
preventing runtime errors with, 312–313, 318
runtime IK in, 266
sending variables to PHP script, 304–305
setting FLVPlayback source property, 242
streaming video using classes in, 367–374, 376
TLF formatting with, 152
tweens for animation, 57–58
using third-party libraries with, 344
working with live camera feeds, 277–282
writing for CLICK events, 34–39
XML support in, 206
ActionScript for Adobe Flash CS5 Classroom in a Book CD, 2
ActionScript Virtual Machine 1 (AVM1), 11
addChild() method, 75, 298, 299, 317, 318
addEventListener() method, 32, 200
addPage() method, 309, 317, 318
Adobe
Community Help, 5–7
training and certification, 7
updating software, 4
Adobe CS Live, 8
Adobe Flash. See Flash CS5
Adobe Integrated Runtime. See AIR applications
Adobe Media Encoder, 243
Adobe Updater, 4
AIR applications, 320–341
about, 322
ActionScript classes for, 322–323, 341
coding custom menus for, 330
creating, 328–330
digital signature for, 326–328, 341
drag-and-drop functionality for, 333–340
AIR applications (continued)
  Flash publish settings for, 324–328, 341
  icons for, 325–326, 341
  looping through file-type arrays, 336–339
  printing from, 332–333
  reviewing lesson file for, 330–332
  using, 12
  working with clipboard, 334
album information, 197–199
alternate video scripts, 281, 284
animation, 48–65. See also cube animation; Fruit Basket game; IK
  ActionScript tweens for, 57–62
  activating cube animation with spacebar, 358–361
  authortime and runtime IK for, 266
  ENTER_FRAME event for, 53–54
  falling fruit at random speeds, 130–131
  looping, 21–22
  modifying text fields in, 41–42
  moving joints in IK armature, 270–275
  restart button for, 40–41
  rotating 3D cube, 360, 361–363
  sound added to armature, 275–277
  storing clip references in variables, 54–56
  triggering armature, 271–275
  tweens and easing classes, 58, 59–62
appendText() method, 203
Application & Installer Settings dialog box, 326, 327
armature
  about IK, 264–265, 298
  moving joints in, 270–275
  sound effects for animation, 275–277
  triggering animation of, 271–275
  working with multiple, 269
array notation, 227
arrays
  about, 127
  adding elements to, 132, 149
  adding songs to playlist, 185
  creating and referring to, 127–128
  index values of, 128, 149
  reviewing lesson file for, 122–123
  searching file types with for loops, 336–339
  splicing elements from, 140
  storing in variable, 149
artist information
  formatting, 197–199
  modifying for XML playlist, 216–218
.as files. See external ActionScript files
asterisk (*), 79, 269
audio. See sound; music player
authortime IK, 266

B
Bandwidth Profiler tool, 103–105, 119
base class, 125

BasicView class
  extending, 351–352
  making instances interactive, 353–354
  onRenderTick() function of, 358–361
Bitmap class, 282–284
bitmap graphics
  classes for, 282
  creating in ActionScript, 299
  placing on 3D cube, 355–358, 376
  storing image in variable, 282–284
BitmapData class, 282–284, 299
BitmapFileMaterial instance, 355–356
Boolean data type, 161, 174
buttons
  adding event listeners to song, 189
  changing movie clip properties with, 56–57
  changing variable value with, 42–45
  event-handling functions for, 32–34
  restart, 40–41
  Take A Snapshot, 271
  triggering navigation with, 34–39
  updating song labels on, 224–225
  URL navigation with, 17–18
  XML playlist, 223–225

C
calling functions, 37
camel case, 33
Camera class, 277, 278
cameras. See live camera feeds
casting data, 241
Caurina Transitions library
  about, 344
  testing installation of, 346–347
  Tweener class of, 361–363, 376
certification programs, 7
CHANGE event
  adding to List, 89, 99
  associated with ColorPicker, 239, 240
  changeHandler() function, 240, 241
  changeVid() function, 253–256
chooseSong() function, 189–190, 219–221
class files
  adding new instances of, 59, 74, 76–77
  assigning for external libraries, 349
  constructor functions and, 71, 82, 353–354
  creating, 68–70
  easing, 58, 59–62
  functions and variables in, 69
  naming, 33, 82
  structure of, 70–73
classes. See also class files; and specific classes
  access modifiers for, 72
  adding instance to Flash Stage, 126
  AIR-specific ActionScript, 322–323, 341
dynamic, 131, 306
extending, 71, 125
importing IK, 268–269
importing to Flash, 182, 350
inheritance and base, 125
methods and properties of, 69
MovieClip and Object, 131
overriding functions from parent, 359–360
Papervision3D animation, 351–352
used by ActionScript for streaming video, 367, 376
working with sound, 178, 183, 203
CLICK events
adding event listeners for, 52–53
adding to nested movie clips, 365–366
changing movie clip properties with, 56–57
responding to MOUSE_DOWN and MOUSE_UP events, 133
using in interactive 3D events, 356–357
writing ActionScript for, 34–39
c1ient property, 367
ClipboardFormat feature, 334
code completion in Flash, 26
code snippets, 16–20
adding to button instance, 36–37
applying, 18–19
comments for, 19–20
creating navigation with, 34–39
creating tween with, 63
defined, 17
modifying, 19–20, 38–39
navigating to URL with, 17–18
Code Snippets panel, 17
collision detection, 137–140
colon (:), 23, 33
color
adjusting skin, 234–235
formatting text, 166–167
generating random, 79–81, 83
hexadecimal, 74, 241
ColorPicker component, 154, 239–240
comments
about code snippet, 19–20
accessing XML, 207
adding to ActionScript, 184
COMPLETE event
about, 227
acknowledging completed loading with, 94–96, 99
completeHandler() function for, 109, 111–112
PROGRESS vs., 109, 119
timing of FLVPlayback component, 252, 260
URLLoader response to, 211–212
completeHandler() function, 109, 111–112
Component Inspector, 18
Component Parameters list, 233, 234, 236
concatenating text, 42, 47, 306
conditional statements, 25–28
about, 27
adding, 25–26
checking key presses with, 162–164
determining game outcome with, 141–143, 148
switch statements, 189–190
updating home frame text, 26
constructor functions
about, 71, 82
creating for Cube3D class, 353–354

copying lesson files, 2–3
count variables, 22–25
CS Live services, 8
cube animation. See also Cube3D class
about cube surfaces, 355
activating zooming with, 358–362
coding as interactive 3D event, 356–357
completed version of, 347
creating and modifying 3D cube, 354, 355–357
Cube3D constructor function, 353–354
displaying movie clips on cube, 363–367, 376
elements needed for, 351
extending BasicView class, 351–352
including streaming video, 366–374, 376
lesson files provided for, 348–349
navigating to URL from movie link, 365–366
placing cube on Flash Stage, 357–358
rotating cube, 360, 361–363
showing new cube face on click, 362–363
variables for Cube3D, 352
Cube3D class
constructor function for, 353–354
extending BasicView class, 351–352
variables for, 352
Cube3D.as file, 350–351
cue points, 242–246
ActionScript, 244–246, 260
defined, 242–243
Event, 244, 260
generating automatically, 247
Navigation, 244, 260
types of, 244, 260
uses for, 246
CUE_POINT event, 245
curly braces ({ }), 33
cube animation. See also Cube3D class
date class, 310
digital signatures, 326–328, 341
display list, 75, 83
display objects
about, 75
adding to display list, 83
using as Papervision3D material, 363
video instance as, 279
DisplayObject class, 367
dot notation, 214, 227
downloading
  estimating time for, 103–105
  external libraries, 344–345
  simulating, 105–106, 119
drag-and-drop functionality
  adding to formatCHIP panel, 165–166
  making draggable objects, 132–134
  setting up AIR applications, 333–340
draw() method, 282, 283
dynamic classes, 131, 306

E
  easing classes, 58, 59–62
  elements
    accessing data from XML, 214
    defined, 207, 227
    parent and child tags, 207–208
  ellipses. See trail of ellipses
e-mail, 300–318
  creating pages to print, 311–312, 318
  examining lesson files for, 302
  linking to external email applications, 302–304
  mailto: syntax for, 303
  printing, 309–317, 318
  sending to external URL, 305
  user-activated variables for sending, 306–308
  using date and time on, 310
  ENTER_FRAME event
    animating movie clip with, 53–54
    catching fruit with, 134–135
    triggering armature animation with, 271–275
equal sign (=), 23, 43
errorHandling function, 212
errors
  checking for loading, 191
  improved checking and feedback for, 12
  IO_ERROR events, 211–212, 227
  preventing ActionScript runtime, 312–313, 318
  reviewing during testing, 147
Event cue points, 244, 260
  event-handling functions, 32, 33
  event listeners
    adding restart button with, 40–41
    adding to thumbnails gallery, 91–93
    button instance with, 36–37
    CHANGE, 89, 240
    CLICK events with, 52–53
    ID3 tag, 195–196
    including in mouse events, 32–33
    keyboard, 161–164, 174
    removing, 78–79
    responding to completed loading, 94–96, 99
    song button, 189
    triggering with snapshot, 271
    using PROGRESS and COMPLETE with loading, 109, 119
  events. See also CLICK events; event listeners; keyboard events; mouse events
    animation using ENTER_FRAME, 53–54
    creating interactive 3D, 356–357
    listening for FLVPlayback COMPLETE, 252
    listening to and responding to, 32–34
    MouseEvent, 32–33
    naming conventions for, 32–33
    PROGRESS, 119, 212
    responding to MOUSE_DOWN and MOUSE_UP, 133
    syntax for, 32
    THUMB_DRAG, 293–296, 299
    URLLoader response to COMPLETE and IO_ERROR, 211–212
    working with, 32
  extending classes, 71, 125
  extends keyword, 71
external ActionScript files, 66–83
  class file structure, 70–73
  confirming loading of, 94–96
  creating class file instances, 74, 76–77
  creating custom classes, 68–70
  importing classes to Flash, 182
  loading to TLF text fields, 158–159
  overriding instance parameters, 77–78
  randomizing ellipse color, 79–81, 83
  required vs. optional parameters, 73
  text loading from, 93–94
  using, 13, 36, 66, 68
  workflow with, 349

external libraries. See third-party ActionScript libraries
F
  F4V files, 231, 260
  feeds. See live camera feeds
  File class, 335, 341
  files. See also class files; external ActionScript files; MP3 files; SWF files
    ActionScript .as, 70–73
    connecting Timed Text to FLVPlaybackCaptioning, 248–249
    creating AIR, 328–330
    downloading external library, 344–345
    dragging and dropping in AIR applications, 335–336
    F4V, 231, 260
    FLA, 231, 260, 324–325
    lesson, 2–3, 5
    loading SWF with URLoader component, 89–90
    loading thumbnail, 91–96
    PBFfilter.as, 288–289
    PHP, 304–305, 307, 308
    PNG, 325
    SVN, 344–345
    SWC, 345
    XML, 206–208
    ZIP, 344–345
FileStream class, 336–339, 341
filter_btn instance, 291–293
filters
  adding live filtering, 289–296
  applying from Pixel Bender, 285–287
  changing parameters with slider, 293–296, 299
  creating PBFilter class interactively, 291–293
  reviewing PBFilter.as file, 288–289
  testing Pixel Bender, 285
FLA files, 231, 260, 324–325
Flash Builder 4, 12–13
Flash CS5. See also display objects; Flash Stage; Flash Timeline; and specific components
  about import statements, 182, 350
  ActionScript class for, 68–70
  adding email link to projects in, 302–304
  adding FLVPlayback component, 231–232
  animation features of, 48
  assigning class file for external libraries, 349
  automatic insertion of import statements, 54, 350
  Bandwidth Profiler tool, 103–105, 119
  class file instances in, 74, 76–77
  code completion feature, 26
  creating AIR applications with, 288–289
  cue points, 242–246
  display objects and display list, 75
  elements needed for Papervision3D projects in, 351
  evolution of ActionScript and, 9–10
  F4V files, 231, 260
  FLA files, 231, 260, 324–325
  hardware requirements for, 2
  installing and optimizing, 2
  live camera access in, 277, 281
  opening XML files in, 206
  playing back ActionScript files for, 12
  ProgressBar component, 107, 110, 113
  sending email from, 304–308
  setting FLVPlayback properties, 233–235
  setting preferences for external libraries, 345–346
  Simulate Download tool, 105–106, 119
  streaming video using ActionScript classes, 367–374
  text in, 50, 150, 152
  tools for inverse kinematics, 264
  updating, 4
  using ActionScript version 3.0 with, 11–12
  using movie clips from library, 123–125
  working with List component, 86–88
Flash Player. See also live camera feeds
  camera access to, 281
  version used, 5
Flash Stage
  adding class instance to, 126
  adding UILoader instance to, 88
  measurements of, 51
  placing external library objects on, 357–358
  width of, 54
Flash Timeline
  ActionScript in frames vs. external files, 13, 36
  adding ActionScript to frames on, 13, 16–17
  animating with IK, 266
  controlling, 25–28
  keyframe labels on, 20–21
  placing code on, 20–26
  scrubbing through, 16
  testing code snippets from, 19, 20
Flex Builder, 12
FLV files, 231, 260
FLVPlayback component
  about, 260
  adding instance of, 231–232
  bypassing with Papervision3D materials, 366–367
  controlling properties with ActionScript, 236–238
  enabling full-screen viewing, 256–258
  listening for COMPLETE event, 252, 260
  property settings for, 233–235
  setting skin color and transparency, 234–235
  skin property for, 234
  source property of, 242
FLVPlaybackCaptioning component, 247–249, 260
font formatting, 155, 166–167
for loops
  about, 129
  creating, 130–131
  searching dragged file name suffixes with, 336–339
  syntax for, 129, 148
  using trace statements in, 130
formatClip panel
  creating instance of Formatter movie clip for, 160–161
  making draggable, 165–166
  responding to user’s keyboard events, 161–164, 174
  text formatting with, 166–168
Formatter movie clip
  about, 153
  creating instance of, 160–161
  responding to user’s keyboard events, 161–164, 174
  formatting
    artist and album properties, 198–199
    fonts, 155, 166–167
frames
  adding ActionScript to Timeline, 13, 16–17
  adding labels to, 20–21
  tying to loading progress, 114–118
Fruit Basket game
  adding class instance to Stage, 126
  adding fruits to, 132
  adding visual feedback to, 143–147
  catching fruit with ENTER_FRAME event, 134–135
  collision detection in, 137–140, 149
  creating for loops for, 130–131
  detecting when fruit hits basket, 136–137
  determining game outcome, 141–143, 148
  dragging and dropping basket, 132–133
Fruit Basket game (continued)
keeping score in, 127, 128
layers in Basket clip, 122–123
listing fruit movie clips as arrays, 127–128
splicing elements from arrays, 140
tracking clips leaving Stage, 136–140
functions. See also specific functions and methods
calling, 37
constructor, 71, 82, 353–354
defined, 33
event-handling, 32–34
naming, 33
overriding, 359–360
parameters in, 71–73, 82
syntax for writing, 33–34

G
getCamera() method, 278, 298
gotoAndPlay() method, 21, 22
gotoAndStop() method, 22

graphics. See also bitmap graphics; IK; trail of ellipses
ActionScript bitmap, 299
bitmap classes, 282
creating impression snapshot taken, 271–275
icons for AIR applications, 325–326, 341
IK, 264–266
placing on 3D cube, 355–358, 376
storing in variable, 282–284
vector, 72–73

H
hardware requirements for Flash, 2
help resources, 5–7
hexadecimal color, 74, 241

hiding/showing
interface elements when filtering snapshots, 290–291
movie clips, 56–57
sliders, 188–189
hitTestObject() method, 137–140, 149
home frame
adding conditional statement to, 25–26
updating text for, 26–28

I
icons for AIR applications, 325–326, 341
ID3 tags
adding listener for, 195, 197
checking and setting in iTunes, 196
compatible versions of, 196
using artist and album information from, 197–198
id3Handler() function, 195, 197
if...else statements. See conditional statements
ignoreComments() method, 207
IK (inverse kinematics). See also armature
about, 264
adding moveDown() function, 271–274, 276–277
armature in, 265, 269, 298
authortime and runtime, 266
coding moveUp() function, 274–275, 276, 277
connecting graphics in, 264–265
importing classes for, 268–269
moving joint in armature, 270–275
reviewing files for, 266–268
sound effects for, 275–277
IKArmature class, 269–270
IKManager class, 269, 270
IKMover class, 270–271
import statements
about, 182, 350
automatic insertion of, 54, 350
importing
external ActionScript classes to Flash, 182, 350
IK classes, 268–269
Tween class, 59–62
index values of arrays, 128, 149
inheritance, 125
installing third-party ActionScript libraries, 344–345
instances. See also specific instances
adding new ActionScript class, 59, 75
checking for existing Sound class, 191
creating class file, 74, 76–77
naming, 33, 36
syntax for new tween, 58
working with Sound, SoundChannel, and
SoundTransform, 183–184, 191–193
interactive content, controlling user-activated tweens, 362–363
interactive features. See also cube animation; Fruit Basket
game; trail of ellipses
class parameters detecting interactivity, 353
creating PBFilter class interactively, 291–293
live camera feeds, 277–282
live tracking of filter values, 293–296, 299
sound and, 176
using user-activated cue points in video, 242–246
internal access modifier, 72
inverse kinematics. See IK
IO_ERROR events
about, 227
URLLoader response to, 211–212
iTunes
checking and setting ID3 tags, 196
linking songs to, 209, 221–223

J
joints in IK armature, 270–275
K

keyboard events
  - animating 3D cube with spacebar, 347, 358–362
  - formatting text with user’s key presses, 161–164, 174
keyCheck function, 162
KEY_DOWN event, 161, 162, 354
keyframes. See frames
KEY_UP event, 161, 354

L

labels
  - adding to keyframes, 20–21
  - adding to XML playlist, 213, 215
  - updating song, 224–225
languages, 42–45
lesson files
  - copying, 2–3
  - Flash Player version used with, 5
  - using, 3
libraries. See also third-party ActionScript libraries
  - about third-party ActionScript, 344
  - Flash CS5, 124–126
links
  - adding email, 302–304
  - linking songs to iTunes, 209, 221–223
  - mailto: syntax for email, 303
  - setting movie clip properties, 124–125
List component
  - about, 86
  - adding CHANGE event listener to, 89
  - setting parameters for, 86–88
live camera feeds, 277–282
  - checking for online cameras, 278–282
  - displaying, 298
  - filtering snapshots from, 289–296
  - Flash access for, 281
  - single camera access only in Flash, 277
  - taking snapshots from, 282–284
load() method, 227
loading content, 84–99. See also prealoaders
  - adding external text to text field, 158–159
  - adding SWF files into UILoader, 89–90
  - checking for loading errors, 191
  - completing loading, 111–112
  - confirming loading of external content, 94–96, 99
  - loading text from external files, 93–94
  - monitoring loading progress, 109–112
  - prealoaders and, 102
  - reviewing and uploading thumbnail files, 91–96
  - showing loading progress, 113–118
  - tracking loading progress, 108–112
loops. See also conditional statements; for loops
  - conditional statements controlling, 25–28
  - counting number of, 22–25
  - creating animation, 21–22
  - modifying text fields dynamically, 41–42
  - reviewing lesson file for, 122–123

M

makeCube() method
  - making cube with, 354, 355–357
  - using streaming video instances within, 367–370
materials. See 3D materials
menus in AIR, 330
methods. See also functions
  - defining parameters for, 72–73
microphone access, 281
moreSongs() function, 224
mouse events
  - adding green ellipses in trail, 77–78
  - creating graphic trail with, 74, 76–77
  - including event listeners in, 32–33
  - MOUSE_DOWN and MOUSE_UP, 133
  - MOUSE_MOVE event, 76
  - navigating to URL from 3D material with, 365, 370, 373
  - providing random color for trail, 79–81
  - removing event listeners with, 78–79
  - MOUSE_DOWN and MOUSE_UP events, 133
  - MOUSE_MOVE event, 76
moveDown() function, 271–274, 276–277
moveUp() function, 274–275, 276, 277
movie clips
  - adding CLICK event to nested, 365–366
  - buttons for changing properties, 56–57
  - changing properties of, 50–51
  - checking for collisions of, 137–140, 149
  - connecting for IK, 264–265
  - controlling instances with ActionScript, 124–126, 148
  - games with visual feedback, 143–147
  - hiding/showing, 56–57
  - making draggable object, 132–134
  - playing animation while new content loads, 114–118
  - setting linkage properties of, 124–125
  - storing references in variable, 54–56
  - tracking clips leaving Stage, 136–137
  - triggering by loading process, 113–114
  - using with 3D object, 355, 363–367, 376
MovieAssetMaterial class
  - adding CLICK events to movie in, 365–366
  - adding 3D materials with, 363–365
  - parameters of, 364
  - using symbol from Flash library in, 363
MovieClip class, 69, 131
MP3 files
  - adding sound effects, 275
  - ID3 tags in, 195–198
  - reading text data in, 183
  - removing file suffix from title, 187–188
music player. See also playlists
  adding artist and album data in, 197–198
  adding sliders to, 180–181
  controlling song volume and panning, 200–201
  creating playlist array, 185
  declaring variables for current songs, 185
  displaying song titles, 185–187
  hiding sliders, 188–189
  illustrated, 177
  removing file suffix from title, 187–188
  reviewing files for, 178–180
  selecting songs with buttons, 189–193
  stopping current and starting new song, 191

N
/n tag, 198
naming
  class files, 33, 82
  constructor functions, 71, 82, 353
  events, 32–33
  functions and classes, 33
  instances, 33, 36
  variables, 23, 33
NATIVE_DRAG_DROP event, 333–334, 341
NATIVE_DRAG_ENTER event, 333–334, 341
NativeDragEvent class, 333, 341
navigation
  buttons on XML playlist, 223–225
  labels and, 20–21
  using cue points for, 243
Navigation cue points, 244, 260
NetConnection class, 367–374
NetStream class, 367–374, 376
new keyword, 59, 75
NumericStepper component, 154

O
Object class, 131
objects
  about Papervision3D, 355, 356
  placing external library objects on Flash Stage, 357–358
onClick() function, 362–363
onDragDrop() function, 335, 336
online resources, 8
onRenderTick() method, 358–361
onSnapshot() function, 271
operating system
  communicating printing capabilities of, 309
  using clipboard for AIR applications, 334
optional parameters, 73
overriding functions, 359–361

P
package keyword, 70, 71, 182
pageSprite instance, 313–315
panChange() function, 200–201
panning
  controlling current song's, 200–201
  creating slider for, 180–181
  displaying sliders for, 193–194
  hiding slider for, 188–189
  setting values for, 192
SoundTransform instances and, 180–181, 192
paper width and height, 312–313
Papervision3D
  about, 344
  activating animation with spacebar, 347, 358–361
  adding movies to 3D object, 363–367, 376
  bypassing FLVPlayback component with, 366–367
  classes for, 351–352
  creating 3D cube with, 355–357
  extending BasicView class, 351–352
  materials in, 344, 355, 376
  objects in, 355, 356
  testing installation of, 346–347
parameters
  defined, 33
  including in functions, 71–73, 82
MovieAssetMaterial class, 364
overriding instance, 77–78
replace() method's, 187
required vs. optional, 73
setting Cube3D class constructor function, 353–354
setting List component, 86–88
startDrag() m, 134
parent and child tags, 207–208
parentheses ( ), 33
PBFilter.as file, 288–289
PHP files
  confirming received email, 307, 308
  sending ActionScript values using URLVariables class, 305
  using ActionScript variables in, 304–305
Pixel Bender Toolkit 2
  about, 285, 299
  adding live filtering to snapshots, 289–296
  applying filters from, 285–287
  examining PBFilter.as file, 288–289
playlists. See also XML playlist
  adding songs to, 185
  removing file suffix from titles in, 187–188
  replacing songList array with XML instance, 209–210
plus sign (+), 42, 47
PNG format, 325
Preferences dialog box, 345
preloaders, 100–119
  completing loading, 111–112
  defined, 102
Flash troubleshooting tools, 102–106
monitoring loading progress, 109–112
progress bars and text field for, 106–107
reflecting loading progress, 113–118
tracking progress of UILoader component, 108–112
previewing Flash projects, 19, 20
print_btn button, 309
printing, 309–317
about PrintJob class, 309, 311, 332–333
adding date and time to email, 310
from AIR applications, 332–333
preventing runtime errors when, 312–313, 318
sending jobs to printer, 313–317
starting and adding pages to, 311–312, 318
PrintJob class
AIR-only properties for, 332–333
using with Flash-based emails, 309, 311
private access modifier, 72
progress bars, 106–107, 110, 113
PROGRESS event, 109, 119, 212, 227
progressHandler() function, 109–112
properties
ActionScript control of FLVPlayback, 236–238
buttons changing movie clip, 56–57
changing movie clip, 50–53
changing values for, 51–52
ENTER_FRAME event animation for movie clip, 53–54
formatting with TextFormat class, 159–160, 174
setting FLVPlayback, 233–235
setting TLFField, 157, 174
Properties panel, 87
Property Inspector
about, 18
Component Parameters list, 233, 234, 236
protected access modifier, 72
public access modifier, 72
Publish Settings dialog box, 257
publish settings for AIR applications, 324–328, 341
push() method, 132, 149

Q
quotation marks (" "), 27

R
random color, 79–81, 83
removeEventListener() method, 78
removing event listeners, 78–79
replace() method, 187–188, 203
required parameters, 73
resources
help, 5–7
online, 8
rotating 3D cube, 360, 361–363
rotation, 51, 56
runtime. See also AIR applications
formatting text at, 160–161
preventing printing errors at, 312–313, 318
runtime IK, 266

S
scroll bar
adding for text fields, 97–98, 168
showing/hiding, 169–172
scrubbing through Flash Timeline, 16
Select Skin dialog box, 235
send() method, 309, 313, 314, 318
sending email from Flash, 304–308
servers, confirming loading from, 95
setLanguage() function, 43–44
setSongs() function, 213, 215, 218–219
Shader classes, 285
Simulate Download tool, 105–106, 119
skin. See video skin
skin property, 234
sliders
adding to music player, 180–181
adjusting video transparency, 236–238
controlling song volume and panning, 200–201
displaying, 193–194
hiding, 188–189
live tracking of filter values with, 293–296, 299
snapshots
hiding interface elements when filtering, 290–291
initiating animation with, 271
live filtering of, 289–296
sound effects triggered by, 276–277
taking from video feed, 282–284
triggering graphic and animation with, 271–275
songList array, 185, 209–210
songs
accessing title and artist data, 216
adding event listeners to buttons for, 189
controlling volume and panning of, 200–201
creating songList array for, 185
declaring variables for current, 185
displaying titles in music player, 185–187
formatting artist and album data, 198–199
functions for selecting, 189–190
linking to iTunes, 209, 221–223
removing file suffix from title, 187–188
selecting more, 224
setting title and artist fields in XML, 216–218
setting title with for loop, 185–187
stopping and starting, 191
sound. See also songs
about, 176
adding to animation, 275–277
classes working with, 178, 183–185, 203
declaring variables for current songs, 185
sound (continued)
    loading to Sound instance, 191
    working with Sound, SoundChannel, and
    SoundTransform classes, 183–184, 191–193

Sound class
    about, 178, 183, 203
    creating armature sound effects, 275–277
    creating instances of, 183–185, 191
    loading sound to instance, 191

SoundChannel class
    about, 178, 183, 203
    creating armature sound effects, 275–277
    creating instances of, 183–185, 191–193

SoundTransform class
    about, 178, 183, 203
    creating instance for, 191–193
    creating instances of, 183–185

source property, 233, 242

spacebar key press, 347, 358–362

speed property, 131, 134

splice() method, 138, 140

Sprite instance, 311, 312

Stage. See Flash Stage

start() method, 309

startDrag() method, 132–134

startRendering() method, 354

stopDrag() method, 132–133

streaming video, 366–374, 376

strings
    casting data as, 241
    concatenating, 42, 47
    recasting text fields to, 25

substitute, 247

SVN files, 344–345

SWC files, 345

SWF files
    AIR applications capabilities vs., 322, 323, 333
    loading, 89–90
    publishing as AIR project, 324–325
    skin files as, 235

    viewing in browsers, 320

switch statement, 189–190, 191

Symbol Properties dialog box, 124, 125

symbols in IK systems, 264–265

syntax
    array, 127
    changing movie clip properties, 51
    describing Stage width, 54
    event, 32
    for loop, 129, 148
    improved ActionScript 3.0, 11
    loading files into UILoader, 89
    trace statement, 130
    tween variable, 58
    writing functions, 33–34

tags
    /* and */, 184
    ID3, 195–198
    XML, 207–208

Take A Snapshot button
    adding event listener for, 271

    triggering sound effects with, 276–277

testing
    about, 102
    Bandwidth Profiler tool for, 103–105, 119
    code snippets from Timeline, 19, 20
    Flash tools for, 102–106
    hints for debugging and, 62
    installed external libraries, 346–347
    looping animation, 21
    Pixel Bender filters, 285
    reviewing errors during, 147
    Simulate Download tool for, 105–106, 119

Simulate Download tool for, 105–106, 119

text. See also text fields

    about Flash, 152
    concatenating, 42, 47, 306
    displaying scroll bar based on display of, 169–172
    formatting with TextFormat panel, 166–168
    keyboard events formatting, 161–164, 174
    loading from external files, 93–94
    reviewing lesson file for, 152–156
    runtime formatting of, 160–161
    TLF and Classic, 50, 150

    TLF and Classic, 50, 150

TextLayout Framework. See TLF

TextFormat class, 159–160, 174

textLoaded() function, 158

third-party ActionScript libraries
    about, 344
    animating cube with spacebar, 358–362
    assigning class file for, 349
    constructor function for Cube3D class, 353–354
    creating 3D cube from Papervision3D, 355–357
downloading and installing, 344–345
examining ending lesson files for, 347
extending Papervision3D BasicView class, 351–352
including streaming video using, 366–374, 376
setting Flash preferences to, 345–346
reviewing starting files for, 348–349
variables for Cube3D class, 352

3D materials. See also cube animation
adding with MovieAssetMaterial class, 363–365
placing bitmap graphics as, 355–358, 376
setting interactive property for, 356
streaming video, 355, 366–374, 376
using Papervision3D, 344, 355, 376

3D materials. See also cube animation
adding with MovieAssetMaterial class, 363–365
placing bitmap graphics as, 355–358, 376
setting interactive property for, 356
streaming video, 355, 366–374, 376
using Papervision3D, 344, 355, 376

THUMB_DRAG event, 293–296, 299

thumbnails
adding event listeners to loading, 91–93
examining, 91
time printed on emails, 310

Timed Text format
about, 247–248, 260
connecting files to FLVPlaybackCaptioning component, 248–249
illustrated, 247
titles
accessing from XML data, 216
adding text field for video player, 244
displaying, 185–187
modifying for XML playlist, 216–218
removing MP3 suffix from, 187–188

TLF (Text Layout Framework)
about, 50, 150
creating TLF text fields, 156–157
formatting with ActionScript, 152
loading external files to TLF text fields, 158–159

TLFTextField class
importing and creating instance of, 156–157
properties of, 157, 174
toggle functionality, 161–164

trace statements, 130, 310

trail of ellipses
creating class file for, 70–73
creating multiple ellipse instances, 74, 76–77
overriding color of, 77–78
randomizing color of, 79–81, 83
stopping when mouse released, 78–79
training and certification programs, 7

transparency
adjusting values for, 52
skin, 234–235
slide adjusting video, 236–238

try/catch statements, 312–313, 318

Tweener class, 350, 361–363, 376
tweens, 57–62
about, 57–58
adding, 59–62
controlling user-activated, 362–363
creating with code snippet, 63
easing classes, 58, 59–62
new instances of, 59
rotating 3D cube with, 361–363

U

UILayouter component
adding instance to Stage, 88
loading SWF files into, 89–90
PROGRESS event for, 119
review of, 99
tracking percent loaded, 110
tracking progress of, 108–112

UIScrollBar component, 97, 168

URLs
navigating from nested movie clip to, 365–366
omitting spaces between names and values in, 303
sending email to server’s, 304–308

URLVariables class, 305–308, 318

users
additional resources for, 5–7
formatting text at runtime, 160–161
score in Fruit Basket game, 127, 128

V

values
Boolean data, 161, 174
changing movie clip property, 51–52
sending using URLVariables class, 305–308
setting volume and pan, 192
spaces between ActionScript names and, 303
working with true, 164

Values dialog box, 87, 88

var keyword, 23
variables
arrays, 127–129, 149
bitmap image stored in, 282–284
Boolean data type, 161, 174
comparing values of, 43, 47
creating count, 22–23
Cube3D class, 352
declaring for current songs, 185
determining language of text with, 42–45
displaying count, 24
naming, 23, 33
sending email using URLVariables class, 306–308
storing movie clip references in, 54–56
syntax for tween, 58
tracking user’s score with, 128
using wildcards as, 278
vector graphics, 72–73. See also trail of ellipses video
streaming to 3D object, 366–374, 376
using as 3D object material, 355, 376
video cameras. See live camera feeds
Video class, 367, 376
video player, 228–260
adding instance of FLVPlayback component, 231–232
adding title text field, 244
full-screen view for, 256–258
identifying next video in playlist, 253–256
loading next video on playlist in, 252–253
playing videos from playlist, 250–256
reviewing folder for, 230
setting source property of FLVPlayback component, 242
skin background color for, 240–242
storing captions in XML file, 247–249
viewing code in vidlist.xml file, 250–251
video skin
about skin files, 235
background color of, 240–242
color and transparency settings for, 234–235
VideoAssetMaterial class, 366–367, 372
videos. See also video player
selecting from XML playlist, 250–256
using cue points, 242–246
viewing full-screen, 256–258
VideoStreamMaterial class, 367, 369–370, 372
vidlist.xml file
using xmlLoaded() function with, 252–253
viewing code in, 250–251
volume
controlling song, 200–201
creating slider for, 180–181
displaying sliders for, 193–194
hiding slider for, 188–189
values for, 192
volumeChange() function, 200–201

W
webcams. See live camera feeds
wildcard (*), 269, 278
writing ActionScript. See also external ActionScript files;
naming
calling functions, 37
case sensitivity in, 33
casting data, 241
changes in version 3.0, 11–12
creating new instances, 59, 75
dot and array notation in, 214, 227
empty lines and, 40
learning version 3.0, 10
modifying code snippets, 19–20, 38–39
organizing addEventListener() calls together, 200
spaces between names and values, 303

X
XML. See also XML class; XML playlist
about, 206
accessing comments via ActionScript, 207
accessing data from elements, 214
elements for, 207, 227
file structure of, 206–208
storing captions in, 247–249
XML class
about, 227
creating instances of, 210–211
ignoreComments() method of, 207
XML playlist, 204–227
accessing title and artist from XML data, 216
adding call to setSongs() function, 218–219
creating XML and URLLoader class instances, 210–211
identifying next video in, 253–256
illustrated, 205
labeling onstage song clips, 213, 215
linking songs to iTunes, 209, 221–223
loading, 211
moving event listeners to xmlLoaded() function, 213
navigating through, 223–225
providing multiple video files in, 250–256
replacing songList array with XML instance, 209–210
reviewing starting file for, 208–209
selecting more songs, 224
setting title and artist fields in XML, 216–218
updating chooseSong() function for, 219–221
updating song buttons, 224–225
xmlLoaded() function, 212, 213, 252–253

Z
ZIP files, 344, 345
zooming variable, 352, 360, 361